

# Contents

[Documentation sur la recherche cognitive Azure](#)

[Vue d'ensemble](#)

[Qu'est-ce que la Recherche cognitive ?](#)

[Fonctionnalités de la Recherche cognitive](#)

[Nouveautés de la Recherche cognitive](#)

[Démarrages rapides](#)

[Portail](#)

[Créer un service](#)

[Création d'un index](#)

[Créer une application de démonstration](#)

[Créer un ensemble de compétences](#)

[Créer une base de connaissances](#)

[Exécuter des requêtes avec l'Explorateur de recherche](#)

[Modèle ARM](#)

[C#](#)

[Java](#)

[Node.js](#)

[postman](#)

[PowerShell](#)

[Python](#)

[Tutoriels](#)

[Créer une application C#](#)

[1 - Page de recherche de base](#)

[2 - Ajouter la pagination des résultats](#)

[3 - Ajouter type-ahead](#)

[4 - Ajouter des facettes](#)

[5 - Ajouter un classement des résultats](#)

[Indexer des données Azure](#)

[Indexer une base de données Azure SQL](#)

- [Indexer des objets blob Azure JSON](#)
- [Indexer plusieurs sources de données Azure](#)
- [Indexer des données](#)
- [Utiliser l'IA pour créer du contenu](#)
- [C#](#)
- [REST](#)
- [Python](#)
- [Déboguer un ensemble de compétences](#)
- [Créer un analyseur personnalisé](#)
- [Interroger des données à partir de Power Apps](#)
- [Exemples](#)
  - [Exemples C#](#)
  - [Exemples relatifs à Java](#)
  - [Exemples JavaScript](#)
  - [Exemples Python](#)
  - [Exemples REST](#)
- [Concepts](#)
  - [Recherche en texte intégral](#)
  - [Indexeurs](#)
  - [Enrichissement de l'IA](#)
  - [Ensemble de compétences](#)
  - [Sessions de débogage](#)
  - [Magasin de connaissances](#)
  - [Projections](#)
  - [Enrichissement incrémentiel](#)
- [Sécurité](#)
  - [Présentation de la sécurité](#)
  - [Contrôles de sécurité par Azure Policy](#)
  - [Sécurisation des ressources de l'indexeur](#)
  - [Base de référence de la sécurité](#)
- [Guides pratiques](#)
  - [Créer des index et des constructions](#)

[Création d'un index](#)

[Créer un index multilingue](#)

[Analyseurs](#)

[Ajout d'analyseurs](#)

[Ajouter un analyseur de langage](#)

[Ajouter un analyseur personnalisé](#)

[Synonymes](#)

[Ajouter des synonymes](#)

[Exemple de synonymes](#)

[Modéliser les types de données complexes](#)

[Modèle de données relationnelles](#)

[Importer des données](#)

[Vue d'ensemble de l'importation des données](#)

[Importer des données \(portail\)](#)

[Regénérer un index](#)

[Indexer de grands jeux de données](#)

[Gérer des mises à jour simultanées](#)

[Importer des données Azure \(indexeurs\)](#)

[Utiliser des indexeurs](#)

[Superviser les indexeurs](#)

[Planifier des indexeurs](#)

[Mapper des champs](#)

[Se connecter à des identités managées](#)

[Vue d'ensemble des identités managées](#)

[Stockage Azure](#)

[Azure Cosmos DB](#)

[SQL Database](#)

[Objets blob Azure](#)

[Rechercher sur des objets blob](#)

[Comprendre les objets blob avec l'IA](#)

[Configurer un indexeur d'objets blob](#)

[Indexer des objets blob changés et supprimés](#)

- [Indexer des objets blob de type un-à-plusieurs](#)
- [Indexer du texte brut](#)
- [Indexer des objets blob CSV](#)
- [Indexer des objets blob JSON](#)
- [Indexer des objets blob chiffrés](#)
- [Tables Azure](#)
- [Azure Cosmos DB](#)
- [Azure Data Lake Storage Gen2](#)
- [Bases de données SQL Azure](#)
- [Instances Azure SQL Managed Instance](#)
- [Machines virtuelles Azure SQL Server](#)
- [Utiliser des compétences \(IA\)](#)
  - [Attacher une ressource Cognitive Services](#)
  - [Ensemble de compétences
    - \[Définir des compétences\]\(#\)
    - \[Référencer une annotation\]\(#\)
    - \[Mapper vers les champs d'index\]\(#\)
    - \[Traiter des fichiers image\]\(#\)
    - \[Enrichissement \\(incrémentiel\\) du cache\]\(#\)](#)
  - [Compétences personnalisées
    - \[Intégrer des compétences personnalisées\]\(#\)
    - \[Exemple - Azure Functions \\(Python\\)\]\(#\)
    - \[Exemple - Azure Functions\]\(#\)
    - \[Exemple - Azure Form Recognizer\]\(#\)
    - \[Exemple - Azure Machine Learning\]\(#\)](#)
  - [Conseils de conception](#)
  - [Magasin de connaissances
    - \[Créer avec REST et Postman\]\(#\)
    - \[Voir avec l'Explorateur Stockage\]\(#\)
    - \[Se connecter avec Power BI\]\(#\)
    - \[Exemples de projection\]\(#\)](#)
  - [Requête](#)

- [Types de requêtes et composition](#)
- [Créer une requête simple](#)
- [Utiliser la syntaxe Lucene complète](#)
- [Termes partiels et caractères spéciaux](#)
- [Recherche partielle](#)
- [Autocomplétion « Recherche-en-cours-de-frappe »](#)
- [Créer un suggesteur](#)
- [Ajouter l'autocomplétion et les suggestions](#)
- [Informations de référence sur la syntaxe](#)
  - [Syntaxe de requête simple](#)
  - [Syntaxe de requête complète Lucene](#)
  - [moreLikeThis \(préversion\)](#)
- [Filtrer](#)
  - [Vue d'ensemble du filtre](#)
  - [Filtres de facette](#)
  - [Ajouter la navigation par facettes](#)
  - [Résoudre les problèmes liés aux filtres de collection](#)
  - [Présentation des filtres de collection](#)
  - [Filtrer par langage](#)
- [Résultats](#)
  - [Utiliser les résultats](#)
  - [Scoring de pertinence](#)
  - [Profils de score](#)
  - [Algorithme de similarité](#)
- [Plan](#)
  - [Choisir un niveau](#)
  - [Limites du service](#)
  - [Ajuster la capacité](#)
  - [Mettre à l'échelle pour les performances](#)
  - [Architecture multilocataire](#)
- [Développer](#)
  - [Versions d'API](#)

## Liste des fonctionnalités en préversion

### Développer dans .NET

Utiliser Microsoft.Azure.Search (v10)

Bien démarrer (v10)

### Mettre à niveau le kit de développement logiciel (SDK)

.NET SDK 11.0

.NET SDK 10.0

.NET SDK 9.0

Kit de développement logiciel (SDK) .NET 5.0

Kit de développement logiciel (SDK) .NET 3.0

Kit de développement logiciel (SDK) .NET 1.1

Kit de développement logiciel (SDK) .NET Management

### Mettre à niveau l'API REST

#### Sécurité

Chiffrement des données

Clés gérées par le client

Récupérer les informations de clé

#### Sécurité de trafic entrant

Clés d'accès d'API

Configurer un pare-feu IP

Créer un Private Endpoint

#### Accès basé sur l'identité

Filtres de sécurité

Filtrer sur les identités utilisateur

#### Sécurité sortante (indexeurs)

Accès via une plage d'adresses IP

Accès via une exception de service approuvée

Accès via des points de terminaison privés

#### Gérer

Portail

PowerShell

Déplacer un service d'une région à l'autre

Accès administrateur basé sur le rôle

Superviser

Notions de base

la journalisation des diagnostics.

Visualiser les journaux de diagnostic

Superviser l'activité des requêtes

Rechercher l'analyse du trafic

Dépanner

Problèmes de l'indexeur

Erreurs et avertissements de l'indexeur

Informations de référence

Azure CLI

Azure PowerShell

Modèle Resource Manager

REST

API de recherche (stable)

API de recherche (préversion)

API de gestion (stable)

API de gestion (préversion)

.NET

API de recherche

API de gestion

Java

API de recherche

API de gestion

JavaScript

API de recherche

API de gestion

Python

API de recherche

API de gestion

Informations de référence sur le langage OData

[Vue d'ensemble](#)

[\\$filter](#)

[\\$orderby](#)

[\\$select](#)

[any, all](#)

[eq, ne, gt, lt, ge, le](#)

[search.ismatch, search.ismatchscoring](#)

[geo.distance, geo.intersects](#)

[and, or, not](#)

[search.in](#)

[search.score](#)

[Grammaire du langage OData](#)

[Informations de référence sur les compétences](#)

[Compétences prédéfinies](#)

[Vue d'ensemble](#)

[Logique conditionnelle](#)

[Recherche d'entité personnalisée](#)

[Extraction de document](#)

[Reconnaissance d'entité](#)

[Analyse d'image](#)

[Extraction d'expressions clés](#)

[Détection de la langue](#)

[OCR](#)

[Détection PII](#)

[Sentiments](#)

[Modélisateur](#)

[Fusion de texte](#)

[Fractionnement de texte](#)

[Traduction de texte](#)

[Compétences personnalisées](#)

[Azure Machine Learning \(AML\)](#)

[API web personnalisée](#)

- Compétences dépréciées
- Reconnaissance d'entité nommée
- Éléments intégrés Azure Policy
- Ressources
  - Support de la communauté Azure
  - Mises à jour Azure
  - Forum de commentaires (UserVoice)
  - Conformité et certifications
  - Liens vers la documentation (exploration de connaissances)
  - Sites de démonstration
    - Présentation des fichiers financiers
    - Présentation des fichiers JFK
  - Entrainement
    - Introduction (Microsoft Learn)
    - Ajouter la recherche dans les applications (Pluralsight)
    - Cours pour développeurs (Pluralsight)
  - Questions fréquentes (FAQ)
  - Tarifs
  - Vidéos

# Qu'est-ce que la Recherche cognitive Azure ?

04/10/2020 • 31 minutes to read • [Edit Online](#)

La Recherche cognitive Azure ([anciennement « Recherche Azure »](#)) est une solution cloud de recherche en tant que service, qui offre aux développeurs des API et des outils permettant d'ajouter une expérience de recherche riche concernant du contenu privé et hétérogène dans les applications web, mobiles et d'entreprise.

Dans une solution personnalisée, un service de recherche se situe entre deux charges de travail principales : ingestion de contenu et requêtes. Votre code ou un outil définit un schéma et appelle l'ingestion de données (indexation) pour charger un index dans Recherche cognitive Azure. Si vous le souhaitez, vous pouvez ajouter des compétences cognitives pour appliquer des processus IA pendant l'indexation. Cela permet de créer des informations et structures utiles pour les scénarios de recherche et d'exploration des connaissances.

Une fois qu'un index existe, votre code d'application émet des demandes de requête à un service de recherche, puis gère les réponses. L'expérience de recherche est définie dans votre client via la fonctionnalité de la Recherche cognitive Azure, avec l'exécution de requête sur un index persistant que vous créez, possédez et stockez dans votre service.

Cette fonctionnalité est exposée par le biais d'une [API REST](#) ou d'un [SDK.NET](#) simple, qui masque la complexité inhérente de la récupération d'informations. En plus des API, le portail Azure offre la prise en charge de l'administration et de la gestion de contenu, avec des outils de prototypage et d'interrogation de vos index. Étant donné que le service s'exécute dans le cloud, infrastructure et la disponibilité sont gérées par Microsoft.

## Quand utiliser la Recherche cognitive Azure

La Recherche cognitive Azure est adaptée aux scénarios d'application suivants :

- Consolidation de types de contenu hétérogènes dans un index privé, unique et pouvant faire l'objet d'une recherche. Les requêtes sont toujours sur un index que vous créez et chargez avec des documents, et l'index réside toujours dans le cloud de votre service de Recherche cognitive Azure. Vous pouvez remplir un index avec des flux de documents JSON à partir de n'importe quelle source ou plateforme. Autrement pour du contenu provenant d'Azure, vous pouvez utiliser un *indexeur* pour extraire des données dans un index. La définition et la gestion/propriété d'index est un motif clé de l'utilisation de la Recherche cognitive Azure.
- Le contenu brut est constitué d'un long texte indifférencié, de fichiers image ou de fichiers d'application, tels que des types de contenu Office sur une source de données Azure, comme le stockage Blob Azure ou Cosmos DB. Vous pouvez appliquer des compétences cognitives pendant l'indexation pour ajouter une structure ou extraire du texte pouvant faire l'objet de recherches à partir de fichiers image et d'application.
- Implémentation facile des fonctionnalités de recherche. Les API de Recherche cognitive Azure simplifient la construction des requêtes, la navigation par facettes, les filtres (dont la recherche spatiale), la mise en correspondance des synonymes, les requêtes TypeAhead et le paramétrage de la pertinence. À l'aide des fonctionnalités intégrées, vous pouvez répondre aux attentes des utilisateurs finaux en matière de recherche, de la même façon que les moteurs de recherche Web commerciaux.
- Indexation de texte non structuré ou extraction de texte et d'informations à partir de fichiers image. La fonctionnalité d'[enrichissement de l'IA](#) de la Recherche cognitive Azure ajoute un traitement de l'IA à un pipeline d'indexation. Certains cas d'utilisation courants incluent la reconnaissance optique des documents numérisés, la reconnaissance d'entités et l'extraction d'expressions clés sur les documents volumineux, la détection de la langue et la traduction de texte, et l'analyse des sentiments.

- Les exigences linguistiques sont respectées à l'aide des analyseurs personnalisés et linguistiques de la Recherche cognitive Azure. Si vous disposez d'un contenu autre que l'anglais, la Recherche cognitive Azure prend en charge les analyseurs Lucene et les processeurs de langage naturel de Microsoft. Vous pouvez également configurer des analyseurs pour obtenir un traitement spécialisé de contenu brut, tel que le filtrage des signes diacritiques.

## Description des fonctionnalités

RECHERCHE DE BASE	FONCTIONNALITÉS
Recherche de texte de forme libre	<p>La <b>recherche en texte intégral</b> est le principal cas d'utilisation de la plupart des applications basées sur la recherche. Vous pouvez formuler des requêtes à l'aide d'une syntaxe prise en charge.</p> <p>La <b>syntaxe de requête simple</b> offre des opérateurs logiques, de recherche d'expression, de suffixe et de précédence.</p> <p>La <b>syntaxe de requête Lucene</b> inclut toutes les opérations de la syntaxe simple, avec en plus des extensions pour la recherche partielle, la recherche de proximité, la promotion de termes et les expressions régulières.</p>
Pertinence	<p><b>Notation simple</b> est l'un des principaux avantages de la Recherche cognitive Azure. Les profils de score permettent de modéliser la pertinence en fonction des valeurs dans les documents eux-mêmes. Par exemple, vous souhaiterez peut-être que les nouveaux produits ou les produits en promotion apparaissent en priorité dans les résultats de la recherche. Vous pouvez également créer des profils de score à l'aide de balises pour obtenir un score personnalisé en fonction de préférences de recherche de client que vous avez suivies et stockées séparément.</p>
Recherche basée sur la localisation	<p>La Recherche cognitive Azure traite, filtre et affiche les lieux géographiques. Les utilisateurs peuvent ainsi explorer les données selon la proximité d'un résultat de recherche par rapport à un emplacement physique. <a href="#">Regardez cette vidéo</a> ou <a href="#">étudiez cet exemple</a> pour en savoir plus.</p>
Filtres et facettes	<p>La <b>navigation par facettes</b> est activée par le biais d'un seul paramètre de requête. La Recherche cognitive Azure retourne une structure de navigation par facettes que vous pouvez utiliser en tant que code de liste de catégories pour le filtrage autonome (par exemple, pour filtrer les éléments de catalogue par fourchette de prix ou par marque).</p> <p>L'option <b>Filtres</b> permet d'intégrer la navigation par facettes dans l'interface utilisateur de votre application, d'améliorer la formulation des requêtes et d'appliquer un filtre en fonction de critères spécifiés par les utilisateurs ou les développeurs. Créez des filtres à l'aide de la syntaxe OData.</p>

RECHERCHE DE BASE	FONCTIONNALITÉS
Fonctionnalités de l'expérience utilisateur	<p>L'<a href="#">Autocomplétion</a> peut être activée sur une barre de recherche pour les requêtes prédictives.</p> <p><a href="#">Suggestions de recherche</a> fonctionne sur les entrées de texte partielle dans une barre de recherche, mais les résultats sont également les documents dans vos conditions d'index plutôt que de requête.</p> <p><a href="#">Synonymes</a> Dans les moteurs de recherche, les synonymes associent des termes équivalents qui élargissent implicitement l'étendue d'une requête, sans que l'utilisateur ait à fournir le terme.</p> <p>Le <a href="#">surlieur d'éléments</a> applique la mise en forme de texte à un mot clé correspondant dans les résultats de la recherche. Vous pouvez choisir quels champs renvoient les extraits de texte en surbrillance.</p> <p><a href="#">Tri</a> est proposée pour plusieurs champs par le biais du schéma d'index, et elle est activée ou désactivée au moment de la requête avec un paramètre de recherche unique.</p> <p>La <a href="#">pagination</a> et la limitation des résultats de la recherche sont d'un fonctionnement très simple avec le contrôle finement ajusté qu'offre la Recherche cognitive Azure sur vos résultats de recherche.</p>
ENRICHISSEMENT IA	FONCTIONNALITÉS
Traitement de l'IA pendant l'indexation	Un <a href="#">enrichissement de l'IA</a> à des fins d'analyse de texte et d'images peut être appliqué à un pipeline d'indexation afin d'extraire des informations textuelles à partir de contenus bruts. Parmi les <a href="#">compétences intégrées</a> , on peut citer la reconnaissance de caractères optiques (ce qui rend possible la recherche de JPEG), la reconnaissance d'entité (identifiant une organisation, un nom ou un emplacement), et la reconnaissance de phrase clé. Vous pouvez aussi <a href="#">coder des compétences personnalisées</a> à attacher au pipeline. Vous pouvez également <a href="#">intégrer des compétences créées Azure Machine Learning</a> .
Stockage de contenu enrichi pour l'analyse et la consommation dans des scénarios de non-recherche	La <a href="#">base de connaissances</a> est une extension de l'indexation basée sur l'IA. Avec Stockage Azure en tant que back-end, vous pouvez enregistrer les enrichissements créés lors de l'indexation. Ces artefacts peuvent vous aider à concevoir des ensembles de compétences plus performants ou à créer une forme et une structure à partir de données amorphes ou ambiguës. Vous pouvez créer des projections de ces structures afin de cibler des charges de travail ou des utilisateurs spécifiques. Vous pouvez également analyser directement les données extraites ou les charger dans d'autres applications.
Contenu mis en cache	L'option <a href="#">Enrichissement incrémentiel (préversion)</a> limite le traitement uniquement aux documents modifiés par la modification spécifique du pipeline, à l'aide du contenu mis en cache pour les parties du pipeline qui ne changent pas.

IMPORTATION/INDEXATION DE DONNÉES	FONCTIONNALITÉS
Sources de données	<p>Les index Recherche cognitive Azure acceptent les données de n'importe quelle source, sous réserve qu'elles soient soumises en tant que structure de données JSON.</p> <p>Les <b>indexeurs</b> automatisent l'ingestion des données pour les sources de données Azure prises en charge et gèrent la sérialisation JSON. Connectez-vous à <a href="#">Azure SQL Database</a>, <a href="#">Azure Cosmos DB</a> ou <a href="#">Stockage Blob Azure</a> pour extraire le contenu pouvant faire l'objet d'une recherche dans les magasins de données primaires. Les indexeurs d'objets blob Azure peuvent effectuer une <i>recherche de document</i> pour <a href="#">extraire le texte présentant la plupart des formats de fichier</a>, y compris Microsoft Office, ainsi que les documents PDF et HTML.</p>
Structures de données hiérarchiques et imbriquées	<p>Les <b>types complexes</b> et les collections vous permettent de modéliser pratiquement tout type de structure JSON en tant qu'index Recherche cognitive Azure. La cardinalité « un à plusieurs » et « plusieurs à plusieurs » peut être exprimée de manière native à travers des collections, des types complexes et des collections de types complexes.</p>
Analyse linguistique	<p>Les analyseurs sont des composants utilisés pour le traitement au cours des opérations d'indexation et de recherche de texte. Il existe deux types de clé API.</p> <p>Les <b>analyseurs lexicaux personnalisés</b> servent pour des requêtes de recherche complexes en utilisant la mise en correspondance phonétique et des expressions régulières.</p> <p>Les <b>analyseurs de langage</b> de Lucene ou de Microsoft sont utilisés pour gérer intelligemment les caractéristiques linguistiques propres à la langue, notamment les temps des verbes, le masculin et le féminin, les noms au pluriel irrégulier (par exemple, « cheval » et « chevaux »), la décomposition des mots, la césure des mots (pour les langues sans espaces), etc.</p>
NIVEAU DE LA PLATE-FORME	FONCTIONNALITÉS
Outils de prototypage et d'inspection	Dans le portail, vous pouvez utiliser l' <b>Assistant Importation de données</b> pour configurer des indexeurs, le concepteur d'index pour créer un index, et l' <b>Explorateur de recherche</b> pour tester les requêtes et affiner les profils de score. Vous pouvez également ouvrir un index pour consulter son schéma.
Surveillance et diagnostics	<b>Activez les fonctionnalités de surveillance</b> pour aller au-delà des métriques visibles en un coup d'œil sur le portail. Des mesures sur les requêtes par seconde, la latence et la limitation sont capturées et affichées sur les pages du portail sans aucune configuration supplémentaire.

NIVEAU DE LA PLATE-FORME	FONCTIONNALITÉS
Chiffrement côté serveur	Le <a href="#">chiffrement au repos géré par Microsoft</a> est intégré dans la couche de stockage interne et est irréversible. Vous pouvez également compléter le chiffrement par défaut avec des <a href="#">clés de chiffrement gérées par le client</a> . Les clés que vous créez et gérez dans Azure Key Vault permettent de chiffrer les index et les mappages de synonymes dans Recherche cognitive Azure.
Infrastructure	<p>La <a href="#">plateforme haute disponibilité</a> garantit une expérience de service de recherche extrêmement fiable. Lorsqu'il est correctement mis à l'échelle, <a href="#">Recherche cognitive Azure offre un SLA de 99,9 %</a>.</p> <p>En tant que solution complète <a href="#">entièrement gérée et extensible</a>, Recherche cognitive Azure n'exige absolument aucune gestion d'infrastructure. Votre service peut être adapté à vos besoins avec la mise à l'échelle en deux dimensions pour gérer plus de stockage de documents, plus de charge de requêtes, ou les deux.</p>

## Utilisation de la Recherche cognitive Azure

### Étape 1 : Configuration du service

Vous pouvez provisionner un service Recherche cognitive Azure dans le [portail Azure](#) ou via l'[API Gestion des ressources Azure](#). Vous pouvez opter pour le service gratuit partagé avec d'autres abonnés ou pour un [niveau payant](#) qui dédie les ressources que seul votre service utilise. Pour les niveaux payants, vous pouvez mettre à l'échelle un service dans deux dimensions :

- Ajoutez des répliques pour accroître votre capacité à traiter de lourdes charges de requêtes.
- Ajoutez des partitions pour accroître votre capacité à stocker plus de documents.

En gérant le stockage de documents et le débit de requêtes séparément, vous pouvez étalonner l'allocation de ressources en fonction des besoins de production.

### Étape 2 : Créer un index

Avant de pouvoir charger du contenu pouvant être recherché, vous devez d'abord définir un index Recherche cognitive Azure. Un index est comparable à une table de base de données qui conserve vos données et peut accepter des requêtes de recherche. Vous devez définir le schéma d'index à mapper pour refléter la structure des documents dans lesquels vous voulez effectuer des recherches, de la même façon que les champs d'une base de données.

Il est possible de créer un schéma dans le portail Azure ou par programmation à l'aide du [Kit de développement logiciel \(SDK\) .NET](#) ou de l'[API REST](#).

### Étape 3 : Charger les données

Une fois que vous avez défini un index, vous êtes prêt à charger du contenu. Vous pouvez utiliser un modèle push ou pull.

Le modèle push extrait des données auprès de sources de données externes. Ce modèle est pris en charge grâce aux *indexeurs* qui simplifient et automatisent certains aspects de l'ingestion de données, comme la connexion aux données, leur lecture et leur sérialisation. Des *indexeurs* sont disponibles pour Azure Cosmos DB, Azure SQL Database, Stockage Blob Azure et SQL Server hébergé dans une machine virtuelle Azure. Vous pouvez configurer un indexeur pour une actualisation de données à la demande ou planifiée.

Le modèle d'émission est fourni via le Kit de développement logiciel (SDK) ou les API REST permettant d'envoyer les documents mis à jour à un index. Vous pouvez émettre des données à partir de n'importe quel groupe de données à l'aide du format JSON. Pour obtenir des conseils sur le chargement des données, consultez [Ajout, mise à jour ou suppression de documents](#) ou [Utilisation du Kit de développement logiciel \(SDK\) .NET](#).

#### Étape 4 : Recherche

Après avoir rempli une index, vous pouvez [émettre des requêtes de recherche](#) à destination du point de terminaison du service en utilisant des requêtes HTTP simples avec l'[API REST](#) ou le [SDK .NET](#).

Parcourez la page [Créer votre première application de recherche](#) pour générer et étendre une page web qui collecte les entrées d'utilisateur et gère les résultats. Vous pouvez également utiliser [Postman pour les appels REST interactifs](#) ou l'[Explorateur de recherche](#) intégré dans le portail Azure pour interroger un index existant.

## Comparaison

Les clients souhaitent souvent comparer les performances de la Recherche cognitive Azure par rapport à d'autres solutions de recherche. Le tableau suivant récapitule ces différences clés.

PAR RAPPORT À	DIFFÉRENCES CLÉS
Bing	<p><a href="#">API Recherche Web Bing</a> recherche dans les index de Bing.com les termes correspondant à votre requête. Les index sont créés à partir de contenus HTML, XML et tout autre contenu web disponible sur les sites publics. Conçu autour des mêmes principes, la <a href="#">Recherche personnalisée Bing</a> offre la même technologie robot pour les types de contenu web, élargie aux sites web individuels.</p> <p>La Recherche cognitive Azure recherche dans un index que vous définissez, alimenté par les données et documents que vous possédez, provenant souvent de sources diverses. La Recherche cognitive Azure dispose de capacités robot pour certaines sources de données à travers des <a href="#">indexeurs</a>, mais vous pouvez envoyer tout document JSON conforme à votre schéma d'index vers une ressource unique consolidée avec possibilité de recherche.</p>

PAR RAPPORT À	DIFFÉRENCES CLÉS
Recherche de base de données	<p>De nombreuses plateformes de base de données incluent une expérience de recherche intégrée. SQL Server dispose d'une <a href="#">recherche de texte intégral</a>. Cosmos DB et autres technologies similaires disposent d'index requétables. Lorsque vous évaluez des produits qui combinent recherche et stockage, il peut être difficile de déterminer quelle fonction utiliser. De nombreuses solutions utilisent les deux : SGBD pour le stockage et la recherche cognitive Azure pour des fonctionnalités de recherche spécialisées.</p> <p>Par rapport à la recherche SGBD, la Recherche cognitive Azure stocke du contenu de sources variées et offre des fonctionnalités de traitement de texte spécialisées telles que du traitement de texte orienté linguistique (recherche de radical, lemmatisation, formes de mots) dans <a href="#">56 langues</a>. Elle prend également en charge la correction automatique de mots, <a href="#">synonymes</a>, <a href="#">suggestions</a>, <a href="#">contrôles de scoring</a>, <a href="#">facettes</a> et <a href="#">segmentation du texte en unités lexicales personnalisée</a>. Le <a href="#">moteur de recherche en texte intégral</a> dans la Recherche cognitive Azure est basé sur Apache Lucene, une norme du secteur en matière de récupération d'informations. Bien que la Recherche cognitive Azure conserve des données sous la forme d'index inversé, il s'agit rarement d'un remplacement de véritable stockage de données. Pour en savoir plus, consultez ce <a href="#">sujet du forum</a>.</p> <p>L'utilisation des ressources est un autre point d'infexion dans cette catégorie. L'indexation et quelques opérations de requête sont souvent intenses en termes de calcul. La recherche de déchargement de DBMS vers une solution dédiée dans le cloud conserve les ressources système pour le traitement transactionnel. De plus, en externalisant la recherche, vous pouvez facilement ajuster l'échelle afin de correspondre au volume de la requête.</p>
Solution de recherche dédiée	<p>En partant du postulat que vous avez choisi la recherche dédiée avec une fonctionnalité complète, une dernière comparaison catégorielle se fera entre des solutions locales et un service cloud. De nombreuses technologies de recherche offrent un contrôle sur l'indexation et les pipelines de requêtes, l'accès à une syntaxe de filtrage et de requête plus riche, un contrôle sur le classement et la pertinence, et des fonctionnalités de recherche intelligente et autonome.</p> <p>Choisissez un service cloud si vous cherchez une solution clé en main ajustable avec une surcharge et une maintenance minimales.</p> <p>Dans le paradigme du cloud, plusieurs fournisseurs proposent des fonctionnalités de base comparables, notamment une recherche en texte intégral, une recherche en fonction de la localisation et une capacité à gérer un certain niveau d'ambiguité dans les entrées de recherche. En général, c'est une <a href="#">fonctionnalité spécialisée</a> ou l'ergonomie et la simplicité globales des API, des outils et de la gestion qui déterminent la meilleure option.</p>

Parmi les fournisseurs de services cloud, la Recherche cognitive Azure s'avère plus efficace pour ce qui est des charges de travail de recherche en texte intégral sur les bases de données et les magasins de contenu sur Azure, dans le cas des applications qui s'appuient principalement sur la recherche pour récupérer les informations et

parcourir le contenu.

Voici les principaux atouts :

- Intégration de données Azure (robots) au niveau de la couche d'indexation
- Portail Azure de gestion centralisée
- Mise à l'échelle Azure, fiabilité et disponibilité de premier ordre
- Traitement IA de données brutes pour faciliter la recherche, y compris le texte dans des images ou la recherche de séquences dans un contenu non structuré.
- Analyse linguistique et personnalisée, incluant des analyseurs de recherche en texte intégral solide en 56 langues
- [Principales fonctionnalités communes aux applications centrées sur les recherches](#) : notation, recherche par facettes, suggestions, synonymes, recherche basée sur la localisation, etc.

#### NOTE

Les sources de données non Azure sont entièrement prises en charge, mais elles s'appuient non pas sur des indexeurs mais sur une méthodologie Push qui fait largement appel au code. En utilisant les API, vous pouvez diriger n'importe quelle collection de documents JSON vers un index Recherche cognitive Azure.

Parmi nos clients, ceux capables d'exploiter le plus large éventail de fonctionnalités de la Recherche cognitive Azure sont les catalogues en ligne, les programmes métier et les applications de découverte de documents.

## API REST | SDK .NET

Bien qu'il soit possible d'effectuer de nombreuses tâches dans le portail, la Recherche cognitive Azure s'adresse avant tout aux développeurs désireux d'intégrer la fonctionnalité de recherche dans des applications existantes. Les interfaces de programmation suivantes sont disponibles.

PLATEFORME	DESCRIPTION
<a href="#">REST</a>	Commandes HTTP prises en charge par tous les langages et toutes les plateformes de programmation, y compris Java, Python et JavaScript
<a href="#">Kit de développement logiciel (SDK) .NET</a>	Le wrapper .NET pour l'API REST offre un codage efficace en C# et d'autres langages de code générés ciblant .NET Framework

## Essai gratuit

Les abonnés Azure peuvent [configurer un service dans le niveau Gratuit](#).

Si vous n'êtes pas abonné, vous pouvez [ouvrir gratuitement un compte Azure](#). Vous obtenez alors des crédits pour tester les services Azure payants. Une fois ceux-ci épuisés, vous pouvez conserver le compte et utiliser les [services Azure gratuits](#). Votre carte de crédit n'est pas débitée tant que vous n'avez pas explicitement modifié vos paramètres pour demander à l'être.

Vous pouvez également [activer les avantages d'abonnement MSDN](#) : Votre abonnement MSDN vous donne droit chaque mois à des crédits dont vous pouvez vous servir pour les services Azure payants.

## Bien démarrer

1. Créez un [service gratuit](#). Tous les démarrages rapides et les didacticiels peuvent être effectués sur le service gratuit.

2. Passez au travers du [didacticiel sur l'utilisation d'outils intégrés pour l'indexation et les requêtes](#). Découvrez les concepts importants et familiarisez-vous avec les informations fournies par le portail.

3. Allez plus loin avec le code en utilisant l'API REST ou .NET :

- [Utilisation du kit de développement logiciel \(SDK\) .NET](#) illustre le flux de travail lié au code managé.
- [Getting started with Azure Search using the REST API](#) (Prise en main de Recherche Azure à l'aide de l'API REST) présente les mêmes étapes avec l'API REST. Vous pouvez également utiliser ce démarrage rapide pour appeler des API REST à partir de Postman ou Fiddler : [Explorer les API REST de la Recherche cognitive Azure](#).

## Regardez cette vidéo

Les moteurs de recherche servent généralement à récupérer les informations sur les applications mobiles, sur le web et dans les magasins de données d'entreprise. Avec la Recherche cognitive Azure, vous disposez d'outils permettant de créer une expérience de recherche semblable à celle des grands sites web commerciaux.

Dans cette vidéo de 15 minutes, le Chef de programme Luis Cabrera présente Recherche cognitive Azure.

# Nouveauté dans Recherche cognitive Azure

04/10/2020 • 19 minutes to read • [Edit Online](#)

Découvrir les nouveautés du service. Marquez cette page pour rester au fait des nouveautés du service.

## Annonces de fonctionnalités en 2020

### Septembre 2020

Créez une identité pour un service de recherche dans Azure Active Directory, puis utilisez les autorisations RBAC pour accorder à l'identité les autorisations en lecture seule sur les sources de données Azure. Vous avez l'option de choisir la fonctionnalité d'[exception de service approuvé](#) si des règles IP doivent être appliquées.

FONCTIONNALITÉ	CATEGORY	DESCRIPTION	DISPONIBILITÉ
<a href="#">Identité MSI (Managed Service Identity)</a>	Indexeurs, sécurité	<p>Créez une identité pour un service de recherche dans Azure Active Directory, puis utilisez les autorisations RBAC pour accorder l'accès aux sources de données Azure. Cette approche élimine la nécessité d'utiliser des informations d'identification dans la chaîne de connexion.</p> <p>Une identité MSI peut aussi être utilisée avec une <a href="#">exception de service approuvé</a> si des règles IP doivent être appliquées.</p>	<p>En disponibilité générale. Accédez à cette fonctionnalité quand vous utilisez le portail ou <a href="#">créez une source de données (REST)</a> avec api-version=2020-06-30.</p>
<a href="#">Requêtes sortantes utilisant une liaison privée</a>	Indexeurs, sécurité	<p>Créez une ressource de liaison privée partagée que les indexeurs pourront utiliser pour accéder aux ressources Azure sécurisées par Azure Private Link. Pour plus d'informations sur toutes les méthodes de sécurisation des connexions des indexeurs, consultez <a href="#">Sécuriser les ressources d'indexeur à l'aide des fonctionnalités de sécurité réseau d'Azure</a>.</p>	<p>En disponibilité générale. Accédez à cette fonctionnalité quand vous utilisez le portail ou la <a href="#">ressource de liaison privée partagée</a> avec api-version=2020-08-01.</p>
<a href="#">API REST Gestion (2020-08-01)</a>	REST	<p>La nouvelle API REST stable autorise la création de ressources de liaison privée partagée.</p>	<p>En disponibilité générale.</p>

FONCTIONNALITÉ	CATEGORY	DESCRIPTION	DISPONIBILITÉ
<a href="#">API REST Gestion (2020-08-01-Preview)</a>	REST	Ajoute une ressource de liaison privée partagée pour Azure Functions et les bases de données Azure SQL pour MySQL.	Préversion publique.
<a href="#">SDK .NET de gestion 4.0</a>	Kit de développement logiciel (SDK) .NET	Mise à jour du SDK Azure pour le SDK de gestion, ciblant la version 2020-08-01 de l'API REST.	En disponibilité générale.

## Août 2020

FONCTIONNALITÉ	CATEGORY	DESCRIPTION	DISPONIBILITÉ
<a href="#">double chiffrement</a>	Sécurité	Activez le double chiffrement au niveau de la couche de stockage en configurant le chiffrement de clé gérée par le client (CMK) sur les nouveaux services de recherche. Créez un service, <a href="#">configurez et appliquez des clés gérées par le client</a> à des index ou à des cartes de synonymes, et bénéficiez du double chiffrement sur ce contenu.	En disponibilité générale sur tous les services de recherche créés après le 1er août 2020 dans les régions suivantes : USA Ouest 2, USA Est, USA Centre Sud, US Gov Virginie, US Gov Arizona. Utilisez le portail, les API REST de gestion ou les SDK pour créer le service.

## Juillet 2020

FONCTIONNALITÉ	CATEGORY	DESCRIPTION	DISPONIBILITÉ
<a href="#">Bibliothèque cliente Azure.Search.Documents</a>	Kit SDK Azure pour .NET	Bibliothèque cliente .NET publiée par l'équipe du SDK Azure, conçue pour assurer la cohérence avec d'autres bibliothèques clientes .NET.  La version 11 cible l'API REST Recherche version 2020-06-30, mais ne prend pas encore en charge la base de connaissances, les Types géospatiaux ou <a href="#">FieldBuilder</a> .  Pour plus d'informations, consultez <a href="#">Démarrage rapide : Créer un index</a> et <a href="#">Mettre à niveau vers Azure.Search.documents (v11)</a> .	En disponibilité générale. Installez le <a href="#">package Azure.Search.Documents</a> à partir de NuGet.

FONCTIONNALITÉ	CATEGORY	DESCRIPTION	DISPONIBILITÉ
Bibliothèque cliente azure.search.documents	Kit SDK Azure pour Python	<p>Bibliothèque cliente Python publiée par l'équipe du SDK Azure, conçue pour assurer la cohérence avec d'autres bibliothèques clientes Python.</p> <p>La version 11 cible l'API REST Recherche version 2020-06-30.</p>	En disponibilité générale. Installez le <a href="#">package azure-search-documents</a> à partir de PyPI.
Bibliothèque cliente @azure/search-documents	Kit SDK Azure pour JavaScript	<p>Bibliothèque cliente JavaScript publiée par l'équipe du SDK Azure, conçue pour assurer la cohérence avec d'autres bibliothèques clientes JavaScript.</p> <p>La version 11 cible l'API REST Recherche version 2020-06-30.</p>	En disponibilité générale. Installez le <a href="#">package @azure/search-documents</a> à partir de npm.

## Juin 2020

FONCTIONNALITÉ	CATEGORY	DESCRIPTION	DISPONIBILITÉ
Base de connaissances	Enrichissement par IA	Sortie d'un indexeur enrichi par IA, avec stockage du contenu dans Stockage Azure pour une utilisation dans d'autres applications et processus.	En disponibilité générale. Utilisez l' <a href="#">API REST Recherche 2020-06-30</a> ou version ultérieure, ou le portail.
API REST Recherche 2020-06-30	REST	Nouvelle version stable des API REST. En plus de la base de connaissances, cette version comprend des améliorations du scoring et de la pertinence des recherches.	En disponibilité générale.
Algorithme de pertinence Okapi BM25	Requête	Nouvel algorithme de classement de pertinence utilisé automatiquement pour tous les nouveaux services de recherche créés après le 15 juillet. Pour les services créés plus tôt, vous pouvez choisir d'utiliser le nouvel algorithme en définissant la propriété <code>similarity</code> sur les champs d'index.	En disponibilité générale. Utilisez l' <a href="#">API REST Recherche 2020-06-30</a> ou l'API REST 2019-05-06.

FONCTIONNALITÉ	CATEGORY	DESCRIPTION	DISPONIBILITÉ
<b>executionEnvironment</b>	Sécurité (indexeurs)	Affectez explicitement la valeur <code>private</code> à cette propriété de configuration d'indexeur pour forcer toutes les connexions à des sources de données externes sur un point de terminaison privé. S'applique uniquement aux services de recherche qui tirent parti d'Azure Private Link.	En disponibilité générale. Utilisez l' <a href="#">API REST</a> <a href="#">Recherche 2020-06-30</a> pour définir ce paramètre de configuration générale.

## Mai 2020 (Microsoft Build)

FONCTIONNALITÉ	CATEGORY	DESCRIPTION	DISPONIBILITÉ
<a href="#">Sessions de débogage</a>	Enrichissement de l'IA	Les sessions de débogage offrent une interface sur basée sur le portail pour examiner et résoudre les problèmes liés à un ensemble de compétences existant. Les correctifs créés dans la session de débogage peuvent être enregistrés dans des ensembles de compétences de production. Pour bien démarrer, suivez <a href="#">ce tutoriel</a> .	Préversion publique, dans le portail.
<a href="#">Règles IP pour la prise en charge de pare-feu entrant</a>	Sécurité	Limitez l'accès à un point de terminaison de service de recherche à des adresses IP spécifiques.	En disponibilité générale. Utilisez l' <a href="#">API REST</a> <a href="#">Gestion 2020-03-13</a> ou version ultérieure, ou le portail.
<a href="#">Azure Private Link pour un point de terminaison de recherche privé</a>	Sécurité	Protégez un service de recherche de l'Internet public en l'exécutant en tant que ressource de liaison privée, accessible uniquement aux applications clientes et aux autres services Azure sur le même réseau virtuel.	En disponibilité générale. Utilisez l' <a href="#">API REST</a> <a href="#">Gestion 2020-03-13</a> ou version ultérieure, ou le portail.

FONCTIONNALITÉ	CATEGORY	DESCRIPTION	DISPONIBILITÉ
<a href="#">Identité managée par le système (préversion)</a>	Sécurité (indexeurs)	Inscrivez un service de recherche en tant que service approuvé auprès d'Azure Active Directory afin de configurer des connexions aux sources de données Azure prises en charge pour l'indexation. S'applique aux <a href="#">indexeurs</a> qui ingèrent le contenu de sources de données Azure comme Azure SQL Database, Azure Cosmos DB et le Stockage Azure.	Préversion publique. Utilisez le portail pour inscrire le service de recherche.
<a href="#">Paramètre de requête sessionId, Paramètre scoringStatistics=global</a>	Requête (pertinence)	Ajoutez sessionID à une requête afin d'établir une session pour le calcul des scores de recherche, avec scoringStatistics=global pour collecter des scores à partir de toutes les partitions, et bénéficiez ainsi de calculs de score de recherche plus cohérents.	En disponibilité générale. Utilisez l' <a href="#">API REST Recherche 2020-06-30</a> ou l' <a href="#">API REST 2019-05-06</a> .
<a href="#">featuresMode (préversion)</a>	Requête	Ajoutez ce paramètre de requête pour développer un score de pertinence afin d'afficher plus de détails : score de similarité par champ, fréquence des termes par champ et nombre de jetons uniques correspondants par champ. Vous pouvez consommer ces points de données dans des algorithmes de scoring personnalisés. Pour obtenir un exemple qui illustre cette fonctionnalité, consultez <a href="#">Ajouter le machine learning (LearnToRank) pour améliorer la pertinence de la recherche</a> .	Préversion publique. Utilisez l' <a href="#">API REST Recherche 2020-06-30-Preview</a> ou l' <a href="#">API REST 2019-05-06-Preview</a> .

## Mars 2020

FONCTIONNALITÉ	CATEGORY	DESCRIPTION	DISPONIBILITÉ
----------------	----------	-------------	---------------

FONCTIONNALITÉ	CATEGORY	DESCRIPTION	DISPONIBILITÉ
<a href="#">Suppression réversible native de blobs (préversion)</a>	Indexeurs	Un indexeur Stockage Blob Azure dans Recherche cognitive Azure reconnaît les objets blob qui sont dans un état de suppression réversible, et supprime le document de recherche correspondant durant l'indexation.	Préversion publique. Utilisez l' <a href="#">API REST Recherche 2020-06-30-Preview</a> et l' <a href="#">API REST 2019-05-06-Preview</a> , avec Exécuter l'indexeur sur une source de données d'objets blob Azure pour laquelle la « suppression réversible » native est activée.
<a href="#">API REST Gestion (2020-03-13)</a>	REST	Nouvelle API REST stable pour la création et la gestion d'un service de recherche. Ajoute la prise en charge de pare-feu IP et de Private Link	En disponibilité générale.

## Février 2020

FONCTIONNALITÉ	CATEGORY	DESCRIPTION	DISPONIBILITÉ
<a href="#">Détection d'informations d'identification personnelle (préversion)</a>	Enrichissement de l'IA	Nouvelle compétence cognitive utilisée lors de l'indexation, qui extrait les informations d'identification personnelle d'un texte d'entrée et vous donne la possibilité de les y masquer de différentes façons.	Préversion publique. Utilisez le portail ou l' <a href="#">API REST Recherche 2020-06-30-Preview</a> ou l' <a href="#">API REST 2019-05-06-Preview</a> .
<a href="#">Recherche d'entité personnalisée (préversion)</a>	Enrichissement par IA	Nouvelle compétence cognitive qui recherche du texte dans une liste personnalisée de mots et d'expressions définie par l'utilisateur. À l'aide de cette liste, elle étiquete tous les documents contenant des entités correspondantes. La compétence prend également en charge un degré de correspondance approximative qui peut être appliqué pour rechercher des correspondances similaires mais non exactes.	Préversion publique. Utilisez le portail ou l' <a href="#">API REST Recherche 2020-06-30-Preview</a> ou l' <a href="#">API REST 2019-05-06-Preview</a> .

## Janvier 2020

FONCTIONNALITÉ	CATEGORY	DESCRIPTION	DISPONIBILITÉ

FONCTIONNALITÉ	CATEGORY	DESCRIPTION	DISPONIBILITÉ
<a href="#">Clés de chiffrement gérées par le client</a>	Sécurité	Ajoute une couche supplémentaire de chiffrement en plus du chiffrement intégré à la plateforme. À l'aide d'une clé de chiffrement que vous créez et gérez, vous pouvez chiffrer le contenu de l'index et les correspondances de synonymes avant que la charge utile n'atteigne un service de recherche.	En disponibilité générale. Utilisez l'API REST Recherche 2019-05-06 ou une version ultérieure. Pour le code managé, le bon package est toujours la <a href="#">version préliminaire 8.0 SDK .NET</a> même si la fonctionnalité n'est plus en préversion.
<a href="#">Règles IP pour la prise en charge de pare-feu entrant (préversion)</a>	Sécurité	Limitez l'accès à un point de terminaison de service de recherche à des adresses IP spécifiques. L'API en préversion comprend de nouvelles propriétés <b>IpRule</b> et <b>NetworkRuleSet</b> dans <a href="#">API CreateOrUpdate</a> . Cette fonctionnalité d'évaluation est disponible dans les régions sélectionnées.	Préversion publique utilisant api-version=2019-10-01-Preview.
<a href="#">Azure Private Link pour un point de terminaison de recherche privé (préversion)</a>	Sécurité	Protégez un service de recherche de l'Internet public en l'exécutant en tant que ressource de liaison privée, accessible uniquement aux applications clientes et aux autres services Azure sur le même réseau virtuel.	Préversion publique utilisant api-version=2019-10-01-Preview.

## Annonces de fonctionnalités en 2019

### Décembre 2019

- Dans le portail, [Créer une application de démonstration \(préversion\)](#) est un nouvel Assistant qui génère un fichier HTML téléchargeable avec accès par requête (lecture seule) à un index. Le fichier est fourni avec un script incorporé qui restitue une application web opérationnelle de type « localhost », qui est liée à un index de votre service de recherche. Les pages sont configurables dans l'Assistant et peuvent contenir une barre de recherche, une zone de résultats, une navigation dans la barre latérale et la prise en charge des requêtes TypeAhead. Vous pouvez modifier le code HTML hors connexion pour étendre ou personnaliser le workflow ou l'apparence. Il est difficile d'étendre une application de démonstration pour qu'elle inclue les couches de sécurité et d'hébergement qui sont généralement nécessaires dans les scénarios de production. Vous devez la considérer comme un outil de validation et de test plutôt que comme un raccourci vers une application cliente complète.
- La section [Créer un point de terminaison privé pour les connexions sécurisées \(version préliminaire\)](#) explique comment configurer une liaison privée pour sécuriser les connexions de votre service Search. Cette fonctionnalité d'évaluation est disponible à la demande et utilise [Azure Private Link](#) et le [Réseau virtuel Azure](#) dans le cadre de la solution.

### Novembre 2019 - Conférence Ignite

- [L'enrichissement incrémentiel \(version préliminaire\)](#) ajoute la mise en cache et l'état à un pipeline

d'enrichissement afin que vous puissiez travailler sur des étapes ou des phases spécifiques sans perdre le contenu déjà traité. Auparavant, toute modification apportée à un pipeline d'enrichissement nécessitait une régénération complète. Grâce à l'enrichissement incrémentiel, la sortie de l'analyse coûteuse, en particulier l'analyse des images, est préservée.

- L'[Extraction de document \(préversion\)](#) est une compétence cognitive utilisée lors de l'indexation, qui vous permet d'extraire le contenu d'un fichier à partir d'un ensemble de compétences. Auparavant, le craquage de document avait exclusivement lieu avant l'exécution de l'ensemble de compétences. Avec l'ajout de cette compétence, vous pouvez également effectuer cette opération dans le cadre de l'exécution de l'ensemble de compétences.
- La [Traduction de texte](#) est une compétence cognitive utilisée pendant l'indexation, qui évalue le texte et, pour chaque enregistrement, retourne le texte traduit dans la langue cible spécifiée.
- Les [Modèles Power BI](#) peuvent lancer rapidement vos visualisations et l'analyse de contenu enrichi dans un magasin de connaissances dans Power BI Desktop. Ce modèle est conçu pour les projections de tables Azure créées à l'aide de l'[Assistant Importer des données](#).
- [Azure Data Lake Storage Gen2 \(préversion\)](#), l'[API Gremlin de Cosmos DB \(préversion\)](#) et l'[API Cassandra de Cosmos DB \(préversion\)](#) sont désormais pris en charge dans les indexeurs. Vous pouvez vous inscrire avec [ce formulaire](#). Vous recevez un e-mail de confirmation une fois que vous avez été accepté dans le programme de préversion.

#### Juillet 2019

- Mise à la disposition générale dans le [Cloud Azure Government](#).

## Nom du nouveau service

Recherche Azure est maintenant renommé **Recherche cognitive Azure** pour refléter l'utilisation étendue (mais facultative) de compétences cognitives et du traitement par IA dans les opérations de base. Les versions d'API, les packages NuGet, les espaces de noms et les points de terminaison ne sont pas modifiés. Les solutions de recherche, nouvelles et existantes, ne sont pas affectées par le changement de nom du service.

## Mises à jour de service

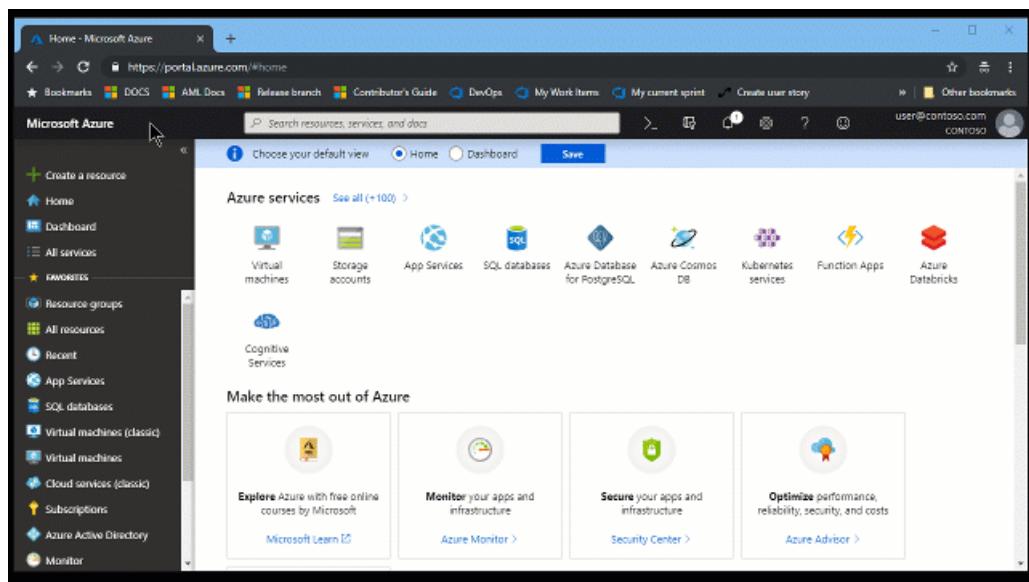
[Les annonces de mise à jour de service](#) pour Recherche cognitive Azure sont disponibles sur le site web Azure.

# Démarrage rapide : Créez un service Recherche cognitive Azure dans le portail

04/10/2020 • 18 minutes to read • [Edit Online](#)

La Recherche cognitive Azure est une ressource autonome utilisée pour raccorder une expérience de recherche à des applications personnalisées. La Recherche cognitive s'intègre facilement à de nombreux autres services Azure, à des applications situées sur des serveurs réseau ou à des logiciels s'exécutant sur d'autres plateformes cloud.

Dans cet article, découvrez comment créer une ressource dans le [portail Azure](#).



Vous préférez PowerShell ? Utilisez le [modèle de service](#) Azure Resource Manager. Pour obtenir de l'aide et bien démarrer, consultez [Gérer la Recherche cognitive Azure avec PowerShell](#).

## Avant de commencer

Les propriétés de service suivantes sont fixes pendant la durée de vie du service et leur modification nécessite un nouveau service. Dans la mesure où elles sont fixes, songez aux implications quand vous remplissez chaque propriété :

- Le nom du service fait désormais partie du point de terminaison de l'URL ([Passez en revue ces conseils](#) pour des noms de service explicites).
- Le niveau de service **affecte la facturation** et définit une limite supérieure sur la capacité. Certaines fonctionnalités ne sont pas disponibles sur le niveau gratuit.
- La région du service peut déterminer la disponibilité de certains scénarios. Si vous avez besoin de **fonctionnalités de haute sécurité** ou d'**enrichissement par IA**, vous devez placer Recherche cognitive Azure dans la même région que les autres services ou dans des régions qui fournissent la fonctionnalité en question.

## S'abonner (payant ou gratuit)

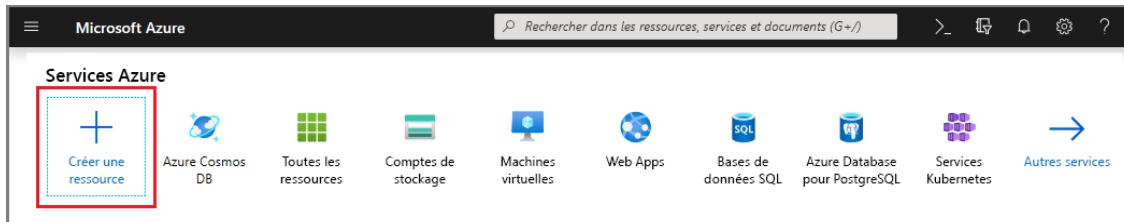
Ouvrez un [compte Azure gratuit](#) et utilisez les crédits gratuits pour essayer les services payants d'Azure. Une fois les crédits épuisés, conservez le compte et continuez à utiliser les services Azure gratuits, tels que les sites Web. Votre carte de crédit n'est pas débitée tant que vous n'avez pas

explicitement modifié vos paramètres pour demander à l'être.

Vous pouvez également [activer les avantages d'abonnement MSDN](#). Un abonnement MSDN vous donne droit chaque mois à des crédits dont vous pouvez vous servir pour les services Azure payants.

## Localiser la Recherche cognitive Azure

1. Connectez-vous au [portail Azure](#).
2. Cliquez sur le signe plus (« + Créer une ressource ») en haut à gauche.
3. Utilisez la barre de recherche pour trouver la « Recherche cognitive Azure », ou accédez à la ressource via Web > Recherche cognitive Azure.



## Sélectionnez un abonnement

Si vous avez plusieurs abonnements, choisissez-en un pour votre service de recherche. Si vous implémentez le [double chiffrement](#) ou d'autres fonctionnalités qui dépendent d'identités de service managées, choisissez le même abonnement que celui utilisé pour Azure Key Vault ou d'autres services pour lesquels des identités managées sont utilisées.

## Définir un groupe de ressources

Un groupe de ressources est un conteneur qui contient des ressources associées de votre solution Azure. Il est nécessaire pour le service de recherche. Il est également utile pour gérer l'ensemble des ressources, dont les coûts. Un groupe de ressources peut se composer d'un service ou de plusieurs services utilisés ensemble. Par exemple, si vous utilisez la Recherche cognitive Azure pour indexer une base de données Azure Cosmos DB, vous pouvez associer les deux services au sein du même groupe de ressources à des fins de gestion.

Si vous ne combinez pas des ressources dans un même groupe, ou si les groupes de ressources existants sont remplis de ressources utilisées dans des solutions non liées, créez un groupe de ressources uniquement pour votre ressource Recherche cognitive Azure.

Accueil > Nouveau > Recherche cognitive Azure > Nouveau service Recherche

## Nouveau service Recherche

De base Échelle Balises Vérifier + créer

**Détails du projet**

Abonnement \* <your-subscription-name-appears-here>

Groupe de ressources \* Sélectionner l'élément existant... **Créer nouveau**

Créer nouveau

Un groupe de ressources est un conteneur qui inclut les ressources associées à une solution Azure.

Nom \* my-prototype-resources

OK Annuler

**Détails de l'instance**

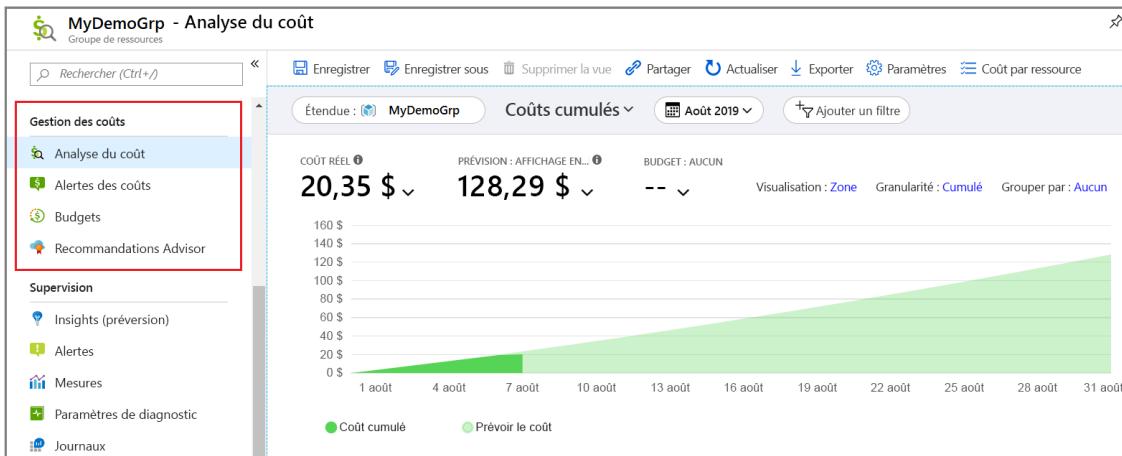
URL \* ⓘ

Emplacement \*

Niveau tarifaire \* ⓘ

Vérifier + créer Précédent Suivant : Échelle

Au fil du temps, vous pouvez effectuer le suivi des coûts actuels et prévus ou consulter les frais des différentes ressources. La capture d'écran suivante montre le type d'informations relatives aux coûts que vous pouvez vous attendre à voir quand vous combinez plusieurs ressources dans un groupe.



### TIP

Les groupes de ressources simplifient le nettoyage, car la suppression d'un groupe supprime tous les services qu'il contient. Pour les projets de prototype utilisant plusieurs services, le fait de les placer tous dans le même groupe de ressources facilite le nettoyage une fois le projet terminé.

## Nommer le service

Dans Détails de l'instance, indiquez un nom de service dans le champ URL. Le nom fait partie du point de terminaison URL par le biais duquel les appels d'API sont émis :

`https://your-service-name.search.windows.net`. Par exemple, si vous souhaitez que le point de terminaison soit `https://myservice.search.windows.net`, vous devez entrer `myservice`.

Configuration requise du nom du service :

- il doit être unique dans l'espace de noms search.windows.net ;
- Il doit comprendre entre 2 et 60 caractères
- Vous devez utiliser des lettres minuscules, des chiffres ou des tirets (« - »)
- N'utilisez pas de tirets (« - ») pour les 2 premiers caractères ou le dernier
- Vous ne pouvez pas utiliser de tirets consécutifs (« -- »)

#### TIP

Si vous pensez utiliser plusieurs services, nous vous recommandons d'inclure la région (ou l'emplacement) dans le nom du service comme convention d'affectation de noms. Les services d'une même région peuvent échanger des données gratuitement. Ainsi, si la Recherche cognitive Azure est située dans la région USA Ouest et si vous disposez d'autres services également dans cette région, un nom tel que `mysearchservice-westus` peut vous éviter de passer par la page de propriétés quand vous décidez de la façon d'associer ou d'attacher des ressources.

## Choisir un emplacement

La Recherche cognitive Azure est disponible dans la plupart des régions. Vous trouverez la liste des régions prises en charge dans la [page de tarification](#).

#### NOTE

Les régions Inde Centre et Émirats arabes unis Nord sont actuellement indisponibles pour les nouveaux services. Pour les services déjà présents dans ces régions, vous pouvez effectuer un scale-up sans aucune restriction, et votre service est entièrement pris en charge dans cette région. Les restrictions sont temporaires et ne concernent que les nouveaux services. Nous supprimerons cette note lorsque la restriction ne s'appliquera plus.

Le double chiffrement est disponible seulement dans certaines régions. Pour plus d'informations, consultez [Double chiffrement](#).

## Spécifications

Si vous utilisez des enrichissements par IA, créez votre service de recherche dans la même région que Cognitive Services. *La colocalisation de la Recherche cognitive Azure et de Cognitive Services dans la même région est obligatoire pour l'enrichissement par IA.*

Les clients ayant des exigences en matière de continuité d'activité et reprise d'activité (BCDR) doivent créer leurs services dans des [paires régionales](#). Par exemple, si vous opérez en Amérique du Nord, vous pouvez choisir USA Est et USA Ouest, ou USA Centre Nord et USA Centre Sud, pour chaque service.

## Recommandations

Si vous utilisez plusieurs services Azure, choisissez une région qui héberge également votre service de données ou d'application. Cela réduit au minimum voire évite les frais de bande passante pour les données sortantes. Il n'y a aucun frais liés aux données sortantes lorsque les services se trouvent dans la même région.

## Sélectionner un niveau tarifaire (SKU)

[La Recherche cognitive Azure est proposée à plusieurs niveaux tarifaires](#) : Gratuit, De base ou Standard. Chaque niveau a ses propres [capacité et limites](#). Pour obtenir de l'aide, voir [Choisir un niveau tarifaire ou une référence \(SKU\)](#).

De base et Standard sont les options les plus courantes pour les charges de production, mais la plupart des clients démarrent avec le service gratuit. Les principales différences entre les niveaux sont la taille et la vitesse des partitions, ainsi que les limites du nombre d'objets que vous pouvez créer.

N'oubliez pas que vous ne pouvez pas changer de niveau tarifaire une fois le service créé. Si vous avez besoin d'un niveau plus élevé ou moins élevé, vous devez recréer le service.

## Créer votre service

Une fois que vous avez fourni les entrées nécessaires, continuez et créez le service.

Accueil > Nouveau > Recherche cognitive Azure > Nouveau service Recherche

### Nouveau service Recherche

De base Échelle Balises Vérifier + créer

Détails du projet

Abonnement \* <your-subscription-name-appears-here>

Groupe de ressources \* my-new-resource-group

Créer nouveau

Détails de l'instance

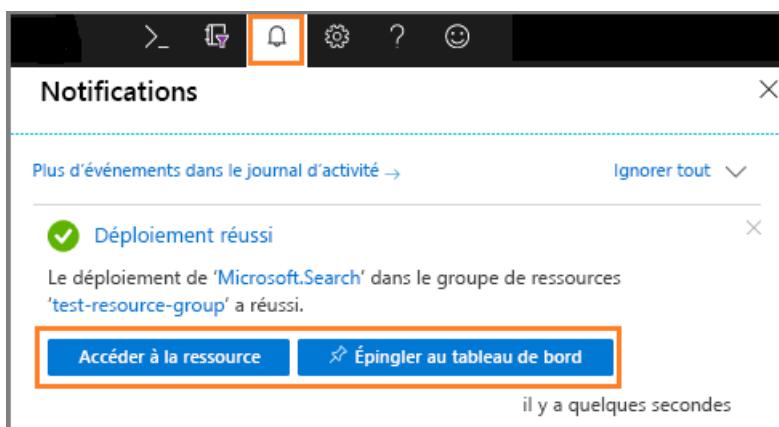
URL \* my-new-search-service

Emplacement \* USA Ouest 2

Niveau tarifaire \* Standard  
25 Go/Partition\*, 12 replicas max., 12 partitions max., 36 unités de recherche max.  
Modifier le niveau tarifaire

Vérifier + créer Précédent Suivant : Échelle

Votre service est déployé en quelques minutes. Vous pouvez superviser la progression par le biais de notifications Azure. Vous pouvez épinglez le service à votre tableau de bord afin d'y accéder plus facilement la prochaine fois.



## Obtenir une clé et un point de terminaison d'URL

Si vous n'utilisez pas le portail, l'accès programmatique à votre nouveau service nécessite de spécifier le point de terminaison d'URL et une clé d'API d'authentification.

1. Dans la page **Vue d'ensemble**, recherchez et copiez le point de terminaison d'URL sur le côté droit de la page.
2. Dans la page **Clés**, copiez l'une des clés d'administration (elles sont équivalentes). Les clés d'API d'administrateur sont nécessaires pour la création, la mise à jour et la suppression d'objets sur votre service. Les clés de requête, quant à elles, fournissent un accès en lecture seule au contenu indexé.

The screenshot shows the Azure portal interface for a search service named 'my-new-search-service'. In the top navigation bar, the URL is listed as `https://my-new-search-service.search.windows.net`. Below the URL, there is a placeholder key value: `<placeholder-for-autogenerated-alphanumeric-string>`. A red box highlights the URL, and a red circle with the number 1 points to it. Another red box highlights the placeholder key value, and a red circle with the number 2 points to it. A 'Copier dans le Presse-papiers' (Copy to Clipboard) button is located in the top right corner of the keys section.

Un point de terminaison et une clé ne sont pas nécessaires pour les tâches effectuées via le portail. Le portail est déjà lié à votre ressource Recherche cognitive Azure avec des droits d'administrateur. Pour obtenir une procédure pas à pas à effectuer dans le portail, commencez par [Guide de démarrage rapide : Créez un index de Recherche cognitive Azure dans le portail](#).

## Mettre à l'échelle le service

Une fois votre service approvisionné, vous pouvez le mettre à l'échelle en fonction de vos besoins. Si vous avez choisi le niveau Standard pour votre service Recherche cognitive Azure, vous pouvez le mettre à l'échelle dans deux dimensions : réplicas et partitions. Si vous choisissez le niveau De base, vous pouvez uniquement ajouter des réplicas. Si vous configurez le service gratuit, la mise à l'échelle n'est pas disponible.

Les *partitions* permettent à votre service de stocker plus de documents et d'effectuer des recherches dans un plus grand nombre de documents.

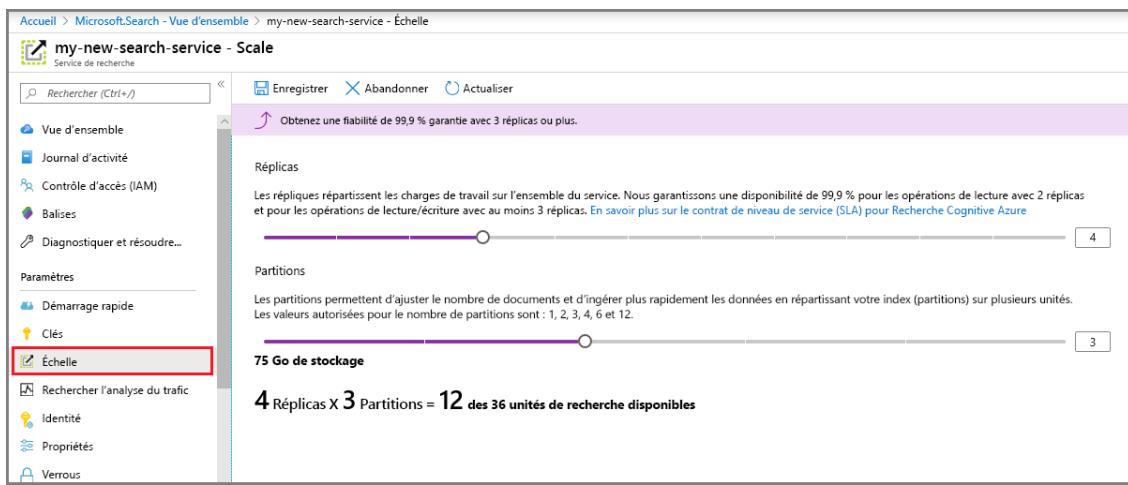
Les *réplicas* permettent à votre service de gérer une charge supérieure de requêtes de recherche.

L'ajout de ressources augmente votre facture mensuelle. Le [calculatrice de prix](#) peut vous aider à comprendre les conséquences de l'ajout de ressources pour la facturation. N'oubliez pas que vous pouvez ajuster les ressources en fonction de la charge. Par exemple, vous pouvez augmenter les ressources pour créer un index initial complet, puis les réduire ultérieurement à un niveau plus approprié pour l'indexation incrémentielle.

### IMPORTANT

Un service doit avoir **2 réplicas pour SLA en lecture seule et 3 réplicas pour SLA en lecture/écriture**.

1. Accédez à la page du service de recherche dans le portail Azure.
2. Dans le volet de navigation de gauche, sélectionnez **Paramètres > Mise à l'échelle**.
3. Utilisez le curseur pour ajouter des ressources de chaque type.



#### NOTE

Le stockage par partition et la vitesse sont plus élevés dans les niveaux de service supérieurs. Pour plus d'informations, consultez [Capacité et limitations](#).

## Quand ajouter un deuxième service

La plupart des clients n'utilisent qu'un seul service provisionné à un niveau qui fournit le **bon équilibre des ressources**. Un service peut héberger plusieurs index, soumis aux **limites maximales du niveau sélectionné**, chaque index étant isolé des autres. Dans la Recherche cognitive Azure, les requêtes ne peuvent être dirigées que vers un seul index, ce qui réduit les risques d'extraction accidentelle ou intentionnelle de données à partir d'autres index du même service.

Bien que la plupart des clients utilisent un seul service, une redondance des services peut être nécessaire en cas d'exigences opérationnelles particulières, notamment :

- **Continuité d'activité et reprise d'activité (BCDR).** La Recherche cognitive Azure ne fournit pas de basculement instantané en cas de panne.
- Les **architectures mutualisées** appellent parfois deux ou plusieurs services.
- Les applications déployées à l'échelle mondiale peuvent nécessiter des services de recherche dans chaque zone géographique pour réduire au minimum la latence.

#### NOTE

Dans la Recherche cognitive Azure, vous ne pouvez pas séparer les opérations d'indexation et d'interrogation. Vous ne créez donc jamais plusieurs services pour des charges de travail distinctes. Un index est toujours interrogé sur le service dans lequel il a été créé (vous ne pouvez pas créer un index dans un service et le copier dans un autre).

Il n'est pas nécessaire de disposer d'un second service pour la haute disponibilité. La haute disponibilité des requêtes est atteinte si vous utilisez au moins deux réplicas dans le même service. Les mises à jour des réplicas sont séquentielles, ce qui signifie qu'au moins l'un d'eux est opérationnel lors du déploiement d'une mise à jour de service. Pour plus d'informations sur la disponibilité, consultez la page [Contrats de niveau de service](#).

## Étapes suivantes

Après avoir approvisionné un service, vous pouvez rester dans le portail et créer votre premier index.

[Démarrez rapidement : Créer un index de Recherche cognitive Azure dans le portail](#)

Vous souhaitez optimiser et réduire vos coûts de cloud ?

[Démarrer l'analyse des coûts avec Cost Management](#)

# Démarrage rapide : Créer un index Recherche cognitive Azure dans le portail Azure

04/10/2020 • 27 minutes to read • [Edit Online](#)

L'**Assistant Importer des données** est un outil du portail Azure qui vous guide lors de la création d'un index de recherche afin que vous puissiez écrire des requêtes intéressantes en quelques minutes.

L'Assistant comporte également des pages pour l'enrichissement basé sur l'IA afin que vous puissiez extraire le texte et la structure des fichiers image et du texte non structuré. Le traitement du contenu avec l'IA inclut la reconnaissance optique de caractères (OCR), l'extraction d'expressions clé et d'entités, ainsi que l'analyse des images.

## Prérequis

Avant de commencer la lecture cet article, vous devez disposer des éléments suivants :

- Compte Azure avec un abonnement actif. [Créez un compte gratuitement](#).
- Service Recherche cognitive Azure. [Créez un service ou recherchez un service existant](#) dans votre abonnement actuel. Vous pouvez utiliser un service gratuit pour ce guide de démarrage rapide.

### Vérifier l'espace disponible

De nombreux clients commencent avec le service gratuit. Cette version est limitée à trois index, trois sources de données et trois indexeurs. Avant de commencer, assurez-vous de disposer d'assez d'espace pour stocker des éléments supplémentaires. Ce didacticiel crée une occurrence de chaque objet.

Les sections figurant sur le tableau de bord des services indiquent le nombre d'index, d'indexeurs et de sources de données dont vous disposez déjà.

The screenshot shows the Microsoft Azure portal interface for a search service named 'my-new-search-service'. On the left, there's a sidebar with various service management options like 'Vue d'ensemble', 'Journal d'activité', and 'Paramètres'. The main content area displays general service information such as resource group ('my-new-resource-group'), location ('USA Ouest 2'), and replication settings ('1 (aucun SLA)'). Below this, the 'Indexeurs' tab is selected in a navigation bar. A table lists index details, showing one entry: 'Nom' (Name) is 'Aucun index trouvé' (No index found), 'Nombre de documents' (Number of documents) is blank, and 'Taille du stockage' (Storage size) is also blank. A red box highlights this table.

## Créer un index et charger des données

Les requêtes de recherche se répètent sur un [index](#) contenant les données de recherche, les métadonnées et les constructions supplémentaires utilisées pour l'optimisation de certains comportements de recherche.

Pour les besoins de ce tutoriel, nous utilisons un exemple de jeu de données intégré qu'il est possible d'analyser à l'aide d'un [indexeur](#) par le biais de l'**Assistant Importer des données**. Un indexeur est un robot d'indexation spécifique à la source qui peut lire les métadonnées et le contenu des sources de données prises en charge Azure. Normalement, les indexeurs sont utilisés par programmation, mais dans le portail, vous

pouvez y accéder via l'Assistant Importation de données.

### Étape 1 : démarrer l'Assistant Importation de données et créer une source de données

1. Connectez-vous au [portail Azure](#) avec votre compte Azure.
2. **Recherchez votre service de recherche.** Ensuite, dans la page Vue d'ensemble, cliquez sur Importer des données dans la barre de commandes pour créer et remplir un index de recherche.



3. Dans l'Assistant, cliquez sur Se connecter aux données > Exemples > hotels-sample. Cette source de données est intégrée. Si vous avez créé votre propre source de données, vous devez spécifier un nom, un type et des informations de connexion. Une fois créée, elle devient une « source de données existante » qui peut être réutilisée dans d'autres opérations d'importation.

A screenshot of the 'Importer des données' (Import data) step in the Azure portal. The URL in the address bar is 'Accueil &gt; Microsoft.Search - Vue d'ensemble &gt; my-new-search-service &gt; Importer des données'. The main area shows a table with two rows. The first row has a 'Type' column with a 'SQL' icon and a 'Nom' column with 'realestate-us-sample'. The second row has a 'Type' column with a 'CSV' icon and a 'Nom' column with 'hotels-sample' (circled with a red number 2). A dropdown menu labeled 'Exemples' (circled with a red number 1) is open above the table. Other options in the menu include 'Enrichir le contenu (facultatif)', 'Personnaliser l'index cible', and 'Créer un indexeur'. A note at the bottom says 'Créez et chargez un index de recherche en utilisant les données d'une source de données Azure existante dans votre abonnement actuel. Le service Recherche Cognitive Azure analyse la structure des données que vous fournissez, extrait le contenu pouvant être recherché, l'enrichit éventuellement avec des qualifications cognitives et le charge dans un index.' followed by a link 'En savoir plus'.

4. Passez à la page suivante.

### Étape 2 : Ignorer la page « Contenu enrichi »

L'Assistant prend en charge la création d'un [pipeline d'enrichissement par intelligence artificielle](#), qui permet d'incorporer les algorithmes d'IA Cognitive Services dans l'indexation.

Nous allons pour le moment ignorer cette étape et passer directement à l'étape de **personnalisation de l'index cible**.



#### TIP

Vous pouvez consulter un exemple d'indexation IA dans un [guide de démarrage rapide](#) ou [didacticiel](#).

### Étape 3 : configurer l'index

En règle générale, la création d'index exige d'utiliser du code avant le chargement des données. Comme indiqué dans ce didacticiel, l'Assistant peut cependant générer un index de base pour n'importe quelle source de données à analyser. Un index requiert au minimum un nom et une collection de champs. L'un de ces champs servira de clé du document pour identifier chaque document de façon unique. En outre, vous pouvez spécifier des analyseurs de langage ou des suggesteurs si vous souhaitez bénéficier d'une autocomplétion ou de suggestions de requêtes.

Les champs comportent des types de données et des attributs. Les cases à cocher figurant dans la partie supérieure sont des *attributs d'index* qui contrôlent le mode d'utilisation du champ.

- **Récupérable** signifie que le champ s'affiche dans la liste des résultats de recherche. En décochant cette case, vous pouvez marquer des champs comme étant hors limites pour les résultats de recherche, par

exemple lorsqu'un champ est utilisé uniquement dans les expressions de filtre.

- **Clé** désigne l'identificateur unique du document. Il est toujours présenté sous forme d'une chaîne et est obligatoire.
- Les options **Filtrable**, **Triable** et **À choix multiples** déterminent si les champs sont utilisés dans une structure de filtre, de tri ou de navigation à facettes.
- **Possibilité de recherche** signifie que le champ est inclus dans la recherche en texte intégral. Les chaînes sont utilisables dans une recherche. Les champs numériques et booléens sont souvent marqués comme ne pouvant pas faire l'objet d'une recherche.

Votre sélection n'a aucune influence sur les besoins en stockage. Par exemple, si vous définissez l'attribut **Récupérable** sur plusieurs champs, les besoins en stockage n'augmentent pas.

Par défaut, l'Assistant analyse la source de données pour y rechercher des identificateurs uniques comme base pour le champ de clé. Les *chaînes* sont dotées des attributs **Récupérable** et **Possibilité de recherche**. Les *entiers* sont dotés des attributs **Récupérable**, **Filtrable**, **Triable** et **À choix multiples**.

## 1. Acceptez les valeurs par défaut.

Si vous réexécutez l'Assistant à l'aide d'une source de données « hotels » existante, l'index ne sera pas configuré avec les attributs par défaut. Vous devrez sélectionner manuellement les attributs lors des prochaines importations.

Accueil > Microsoft.Search – Vue d'ensemble > my-new-search-service > Importer des données

### Importer des données

Se connecter à vos données Enrichir le contenu (facultatif) Personnaliser l'index cible \* Créer un indexeur

Nous avons fourni un index par défaut pour vous. Vous pouvez supprimer les champs dont vous n'avez pas besoin. Tout est modifiable, mais une fois l'index généré, la suppression ou la modification de champs existants nécessite la réindexation de vos documents.

Nom d'index \* ⓘ  
hotels-sample-index

Clé \* ⓘ  
HotellId

Nom du générateur de suggestions Mode de recherche ⓘ  
sg

+ Ajouter un champ + Ajouter un sous-champ Supprimer

Nom du champ	Type	Récupérable	Filtrable	Triable	À choix...	Recherche...	Analyseur	Suggesteur	...
HotellId	Edm.String	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	...
HotelName	Edm.String	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Anglais - Microsoft	<input type="checkbox"/>	...
Description	Edm.String	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Anglais - Microsoft	<input type="checkbox"/>	...

## 2. Passez à la page suivante.

### Étape 4 : configurer l'indexeur

Toujours dans l'Assistant **Importer des données**, cliquez sur **Indexeur** > **Nom**, puis tapez un nom pour l'indexeur.

Cet objet définit un processus exécutable. Vous pouvez le configurer en planification récurrente, mais pour l'instant, utilisez l'option par défaut qui exécute immédiatement l'indexeur une seule fois.

Cliquez sur **Envoyer** pour créer et exécuter simultanément l'indexeur.

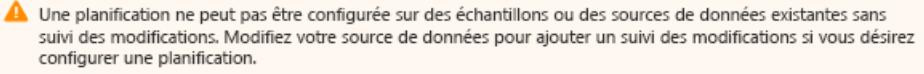
Accueil > Microsoft.Search - Vue d'ensemble > my-new-search-service > Importer des données

## Importer des données

Se connecter à vos données Enrichir le contenu (facultatif) Personnaliser l'index cible \* **Créer un indexeur \***

**Indexeur**

Nom \*

**Planification** 

Une fois  Toutes les heures  Quotidienne  Personnalisé

Description

Options avancées 

Précédent : Personnaliser l'index cible **Envoyer**

## Surveiller la progression

L'Assistant doit vous rediriger vers la liste des indexeurs où vous pourrez en surveiller la progression. Pour une navigation automatique, accédez à la page Vue d'ensemble et cliquez sur **Indexeurs**.

Il faut parfois plusieurs minutes au portail pour actualiser la page, mais l'indexeur que vous venez de créer devrait apparaître dans la liste, avec un état indiquant que l'opération est en cours ou qu'elle a réussi, ainsi que le nombre de documents indexés.

Utilisation	Supervision	Index	Indexeurs	Sources de données	Ensemble de compétences
État	↑↓ Nom	↑↓ Dernière exécution	↑↓ Documents ayant réussi		
En cours	hotels-sample-indexer	À l'instant	0/0		

## Afficher l'index

La page principale du service fournit des liens vers les ressources créées dans votre service Recherche cognitive Azure. Pour voir l'index que vous venez de créer, cliquez sur **Index** dans la liste des liens.

Attendez que la page du portail s'actualise. Après quelques minutes, vous devriez voir l'index avec un nombre de documents et une taille de stockage.

Utilisation	Supervision	Index	Indexeurs	Sources de données	Ensemble de compétences
Nom		Nombre de documents		Taille du stockage	
hotels-sample-index		50		535,41 Ko	

Dans cette liste, vous pouvez cliquer sur l'index *hotels-sample* que vous venez de créer pour voir le schéma de l'index. Ajoutez éventuellement de nouveaux champs.

L'onglet **Champs** montre le schéma d'index. Faites défiler la liste vers le bas pour entrer un nouveau champ. Dans la plupart des cas, vous ne pouvez pas modifier les champs existants. Les champs existants ont une représentation physique dans la Recherche cognitive Azure et ne sont donc pas modifiables, pas même dans du code. Pour modifier considérablement un champ existant, créez un nouvel index en supprimant l'original.

Accueil > Microsoft.Search - Vue d'ensemble > my-new-search-service > hotels-sample-index

hotels-sample-index  
index

Enregistrer Abandonner Actualiser Crée une application de recherche (préversion) Supprimer

Documents 50 Stockage 535,41 Ko

Explorateur de recherche Champs CORS Profils de score Définition d'index (JSON)

Nom du générateur de suggestions	Mode de recherche
sg	

+ Ajouter un champ + Ajouter un sous-champ Supprimer

Nom du champ	Type	Récupér...	Filtrable	Triable	À choix...	Recherche...	Analyseur	Suggesteur
HotelId	Edm.String	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
HotelName	Edm.String	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Anglais - Microsoft	
Description	Edm.String	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Anglais - Microsoft	

D'autres constructions, telles que des profils de score et des options CORS, peuvent être ajoutées à tout moment.

Pour comprendre clairement ce que vous pouvez et ne pouvez pas modifier lors de la conception d'index, prenez une minute pour consulter les options de définition d'index. Les options grisées indiquent qu'une valeur ne peut pas être modifiée ou supprimée.

## Lancer des requêtes à l'aide de l'Explorateur de recherche

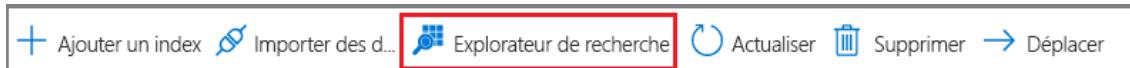
Vous devriez maintenant avoir un index de recherche prêt à lancer des requêtes à l'aide de la page de requête **Explorateur de recherche** intégrée. Il fournit une zone de recherche afin que vous puissiez tester les chaînes de requête arbitraires.

L'**Explorateur de recherche** est uniquement équipé pour gérer des [demandes d'API REST](#), mais il accepte à la fois une syntaxe de [requête simple](#) et celle de l'[analyseur complet de requêtes Lucene](#), ainsi que tous les paramètres de recherche disponibles dans des opérations d'[API REST de recherche dans des documents](#).

### TIP

Les étapes suivantes font l'objet d'une démonstration à 6:08 dans la [vidéo de présentation de la Recherche cognitive Azure](#).

1. Cliquez sur **Explorateur de recherche** dans la barre de commandes.



2. Dans la liste déroulante **Index**, choisissez *hotels-sample-index*. Cliquez sur la liste déroulante **Version d'API** pour voir les API REST disponibles. Pour les requêtes ci-après, utilisez la version mise à la disposition générale (2020-06-30).

Accueil > Microsoft.Search - Vue d'ensemble > my-new-search-service > Explorateur de recherche

**Explorateur de recherche**  
my-new-search-service

**Index**: hotels-sample-ind... | **Version d'API**: 06/05/2019

**Chaîne de requête** ⓘ  
Exemples : \*, \$top=10, \$top=10&\$skip=10&\$search=\*

**Rechercher**

3. Dans la barre de recherche, collez les chaînes de requête ci-dessous, puis cliquez sur **Rechercher**.

**Chaîne de requête** ⓘ  
search=beach&\$filter=Rating gt 4&\$count=true

**Rechercher**

## Exemples de requêtes

Vous pouvez entrer soit des termes et expressions, de la même manière que dans une recherche Bing ou Google, soit des expressions de requête entièrement spécifiées. Les résultats sont retournés sous forme de documents JSON détaillés.

### Requête simple avec les N premiers résultats

**Exemple (requête sous forme de chaîne)** : `search=spa`

- Le paramètre **search** permet d'entrer une recherche par mot clé pour une recherche en texte intégral. Dans ce cas précis, il retourne les hôtels dont l'un des champs contient le mot *spa*.
- L'**Explorateur de recherche** renvoie les résultats au format JSON, qui est particulièrement détaillé et difficile à lire si les documents présentent une structure dense. Cela est intentionnel ; la visibilité du document entier est importante en matière de développement, en particulier lors des tests. Pour une meilleure expérience utilisateur, vous devrez écrire le code qui [gère les résultats de recherche](#) pour mettre en évidence les éléments importants.
- Les documents sont composés de tous les champs marqués comme « récupérables » dans l'index. Pour visualiser les attributs d'index dans le portail, cliquez sur *hotels-sample* dans la liste **Index**.

**Exemple (requête paramétrable)** : `search=spa&$count=true&$top=10`

- Le symbole **&** permet d'ajouter des paramètres de recherche, qui peuvent être spécifiés dans n'importe quel ordre.
- Le paramètre **\$count=true** récupère une valeur indiquant le nombre total de documents retournés. Cette valeur s'affiche en haut des résultats de recherche. Vous pouvez vérifier les requêtes de filtre en surveillant les modifications signalées par **\$count=true**. Des petits nombres indiquent que votre filtre fonctionne.
- La chaîne **\$top=10** retourne les 10 documents les mieux classés parmi tous les documents. Par défaut, la Recherche cognitive Azure retourne les 50 meilleures correspondances. Vous pouvez augmenter ou diminuer ce nombre par le biais du paramètre **\$top**.

### Filtrer la requête

Les filtres sont inclus dans les demandes de recherche lorsque vous ajoutez le paramètre **\$filter**.

**Exemple (filtré)** : `search=beach&$filter=Rating gt 4`

- Le paramètre **\$filter** renvoie les résultats correspondant aux critères que vous avez spécifiés. Dans ce cas précis, ce sont les évaluations supérieures à 4.
- La syntaxe de filtre est une construction OData. Pour plus d'informations, consultez l'article [Filter OData syntax](#) (Syntaxe d'expression de filtre OData).

### « Facetter » la requête

Les filtres de facettes sont inclus dans les demandes de recherche. Vous pouvez utiliser le paramètre de facette pour retourner un nombre agrégé des documents qui correspondent à la valeur de facette que vous fournissez.

**Exemple (par facettes avec une étendue réduite) :** `search=*&facet=Category&$top=2`

- `search=*` est une recherche vide. Les recherches vides portent sur tous les éléments. L'un des motifs possibles de l'exécution d'une requête vide est l'application de filtres ou de facettes au jeu complet de documents. Par exemple, vous souhaitez obtenir une structure de navigation par facettes constituée de tous les hôtels de l'index.
- `facet` renvoie une structure de navigation que vous pouvez transmettre à un contrôle d'interface utilisateur. Il renvoie des catégories ainsi qu'un nombre. Dans ce cas, les catégories sont basées sur un champ nommé *Catégorie*. Il n'existe pas d'agrégation dans la Recherche cognitive Azure, mais vous pouvez bénéficier d'une fonctionnalité quasiment comparable via `facet`, qui retourne un nombre de documents dans chaque catégorie.
- `$top=2` renvoie deux documents, illustrant ainsi la possibilité d'utiliser `top` pour réduire ou augmenter les résultats.

**Exemple (par facettes sur des valeurs numériques) :** `search=spa&facet=Rating`

- Cette requête définit une facette pour l'évaluation dans une recherche de texte portant sur le mot *spa*. Le terme *évaluation* peut être spécifié en tant que facette, car ce champ est désigné comme récupérable, filtrable et « facettable » dans l'index. De plus, les valeurs qu'il contient (valeur numérique de 1 à 5) sont adaptées à un classement des entrées en différents groupes.
- Seuls les champs filtrables peuvent être désignés comme étant à facettes. Les résultats ne peuvent renvoyer que les champs récupérables.
- Le champ *Évaluation* est un champ à virgule flottante et double précision, et le regroupement se fera par valeur précise. Pour plus d'informations sur le regroupement par intervalle (par exemple « 3 étoiles », « 4 étoiles », etc.), consultez [Guide pratique pour implémenter une navigation par facettes dans la Recherche cognitive Azure](#).

## Mettre en surbrillance les termes de recherche

La mise en surbrillance des correspondances fait référence au formatage du texte qui correspond au mot clé, lorsque des correspondances sont trouvées dans un champ spécifique. Si votre terme de recherche est profondément enfoui dans une description, vous pouvez définir une mise en surbrillance des correspondances pour le localiser plus facilement.

**Exemple (surlieur) :** `search=beach&highlight=Description`

- Dans cet exemple, le mot mis en forme *beach* (plage) est plus facile à repérer dans le champ de description.

**Exemple (analyse linguistique) :** `search=beaches&highlight=Description`

- La recherche en texte intégral reconnaît les variations de base au niveau du format des mots. Dans ce cas, en réponse au mot clé de recherche « *beaches* », les résultats de recherche contiennent le mot « *beach* » mis en surbrillance pour les hôtels qui ont ce mot dans leurs champs de recherche. Les résultats peuvent afficher différentes formes du même mot grâce à l'exécution d'une analyse linguistique.
- La Recherche cognitive Azure prend en charge 56 analyseurs Lucene et Microsoft. Par défaut, la Recherche cognitive Azure utilise l'analyseur Lucene standard.

## Essayer la recherche partielle

Par défaut, dans une recherche classique, aucune correspondance n'est retournée pour les termes de requête mal orthographiés, par exemple *seatle* pour « Seattle ». L'exemple suivant ne retourne aucun résultat.

**Exemple (terme mal orthographié, non pris en charge) :** `search=seattle`

Pour gérer les fautes d'orthographe, vous pouvez utiliser une recherche partielle. La recherche partielle est activée lorsque vous utilisez la syntaxe de requête complète Lucene, ce qui arrive lorsque vous effectuez deux actions : définir la requête sur **queryType=full** et ajouter le ~ à la chaîne de recherche.

**Exemple (terme mal orthographié, pris en charge) :** `search=seattle~&queryType=full`

Cet exemple retourne désormais les documents qui contiennent des correspondances pour « Seattle ».

Lorsque l'élément **queryType** n'est pas spécifié, l'analyseur de requêtes simples par défaut est utilisé. L'analyseur de requêtes simples fonctionne plus rapidement, mais si vous avez besoin d'utiliser des recherches partielles, des expressions régulières, des recherches de proximité ou d'autres types de requêtes avancées, vous devrez recourir à la syntaxe complète.

La recherche partielle et la recherche par caractères génériques ont des conséquences sur les résultats de la recherche. L'analyse linguistique n'est pas effectuée sur ces formats de requête. Avant d'utiliser la recherche partielle et la recherche par caractères génériques, consultez [Fonctionnement de la recherche en texte intégral dans la Recherche cognitive Azure](#), puis recherchez la section sur les exceptions relatives à l'analyse lexicale.

Pour plus d'informations sur les scénarios de requête permis par l'analyseur de requêtes complètes, consultez [Syntaxe de requête Lucene dans la Recherche cognitive Azure](#).

### Essayez la recherche géospatiale

La recherche géographique est prise en charge par le biais du [type de données edm.GeographyPoint](#) sur un champ contenant des coordonnées. La recherche géographique est un type de filtre, spécifié dans l'article [Filter OData syntax](#) (Syntaxe d'expression de filtre OData).

**Exemple (filtres géo-coordonnées) :** `search=*&$count=true&$filter=geo.distance(Location,geography'POINT(-122.12 47.67)') le 5`

L'exemple de requête ci-dessus filtre tous les résultats sur la base de données positionnelles et renvoie les résultats situés à moins de 5 kilomètres d'un point donné (spécifié sous la forme de coordonnées de latitude et de longitude). L'ajout du paramètre **\$count** vous permet de connaître le nombre de résultats renvoyés lorsque vous modifiez la distance ou les coordonnées.

La recherche géographique est utile si votre application de recherche dispose d'une fonctionnalité « rechercher à proximité » ou qu'elle utilise la navigation dans les cartes. Toutefois, cette fonction de recherche n'est pas disponible en texte intégral. Si vos utilisateurs doivent rechercher une ville ou un pays par son nom, ajoutez des champs contenant des noms de ville ou de pays, en plus des coordonnées.

## Éléments importants à retenir

Ce tutoriel a fourni une brève présentation de la Recherche cognitive Azure à l'aide du portail Azure.

Vous avez appris à créer un index de recherche à l'aide de l'Assistant **Importer des données**. Vous avez découvert les [indexeurs](#), ainsi que le flux de travail de base pour la conception d'index, y compris les [modifications prises en charge pour un index publié](#).

À l'aide de l'**Explorateur de recherche** dans le portail Azure, vous avez découvert la syntaxe de requête par le biais d'exemples pratiques qui illustrent des fonctionnalités clés comme les filtres, la mise en surbrillance des correspondances, la recherche partielle et la recherche basée sur la géolocalisation.

Vous avez également appris à rechercher des index, des indexeurs et des sources de données dans le portail. À l'avenir, pour toute nouvelle source de données et afin de limiter vos efforts, vous pourrez utiliser le portail pour vérifier rapidement ses définitions ou les collections de champs.

## Nettoyer les ressources

Lorsque vous travaillez dans votre propre abonnement, il est recommandé, à la fin de chaque projet, de déterminer si vous avez toujours besoin des ressources que vous avez créées. Les ressources laissées en cours d'exécution peuvent vous coûter de l'argent. Vous pouvez supprimer les ressources une par une, ou choisir de supprimer le groupe de ressources afin de supprimer l'ensemble des ressources.

Vous pouvez rechercher et gérer les ressources dans le portail à l'aide des liens **Toutes les ressources** ou **Groupes de ressources** situés dans le volet de navigation de gauche.

Si vous utilisez un service gratuit, n'oubliez pas que vous êtes limité à trois index, indexeurs et sources de données. Vous pouvez supprimer des éléments un par un dans le portail pour ne pas dépasser la limite.

## Étapes suivantes

Utilisez un Assistant du portail pour générer une application web prête à l'emploi qui s'exécute dans un navigateur. Vous pouvez essayer cet Assistant sur le petit index que vous venez de créer ou utiliser l'un des exemples de jeux de données intégrés pour une expérience de recherche plus riche.

[Créer une application de démonstration dans le portail](#)

# Démarrage rapide : Créer une application de démonstration dans le portail (Recherche cognitive Azure)

04/10/2020 • 9 minutes to read • [Edit Online](#)

Utilisez l'Assistant **Créer une application de démonstration** du portail Azure pour générer une application web de type « localhost » téléchargeable qui s'exécute dans un navigateur. En fonction de sa configuration, l'application générée est opérationnelle dès la première utilisation, avec une connexion en lecture seule active à un index distant. Une application par défaut peut inclure une barre de recherche, une zone de résultats, des filtres dans une barre latérale et prendre en charge la saisie semi-automatique.

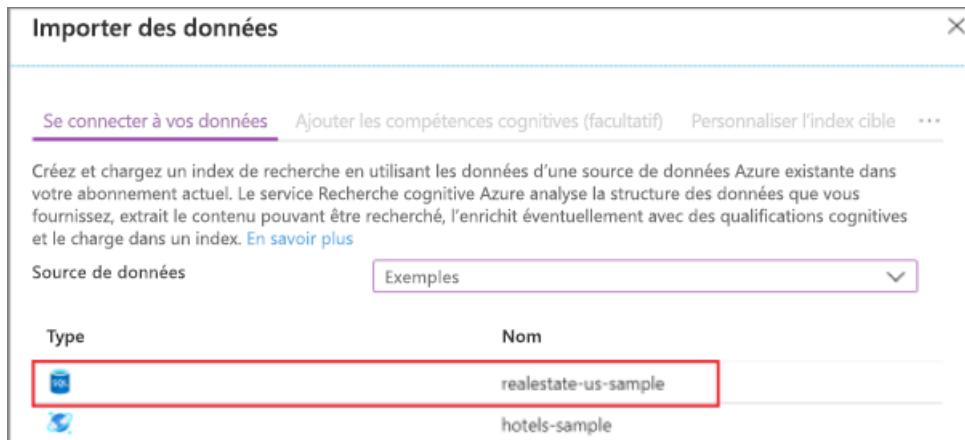
L'application de démonstration peut vous aider à visualiser la manière dont un index fonctionne dans une application cliente, mais elle n'est pas destinée aux scénarios de production. Les applications clientes doivent inclure la logique de sécurité, de gestion des erreurs et d'hébergement que la page HTML générée ne fournit pas. Quand vous êtes prêt à créer une application cliente, consultez [Créer votre première application de recherche à l'aide du SDK .NET](#) pour connaître les étapes suivantes.

## Prérequis

Avant de commencer la lecture cet article, vous devez disposer des éléments suivants :

- Compte Azure avec un abonnement actif. [Créez un compte gratuitement](#).
- Service Recherche cognitive Azure. [Créez un service](#) ou [recherchez un service existant](#) dans votre abonnement actuel. Vous pouvez utiliser un service gratuit pour ce guide de démarrage rapide.
- [Microsoft Edge \(dernière version\)](#) ou Google Chrome.
- Un [index de recherche](#) à utiliser comme base de votre application générée.

Ce guide de démarrage rapide utilise l'exemple prédéfini de données et d'index Real Estate, car il contient des images miniatures (l'Assistant prend en charge l'ajout d'images à la page de résultats). Pour créer l'index utilisé dans cet exercice, exécutez l'Assistant **Importer des données**, en choisissant la source de données *realestate-us-sample*.



Quand l'index est prêt à être utilisé, passez à l'étape suivante.

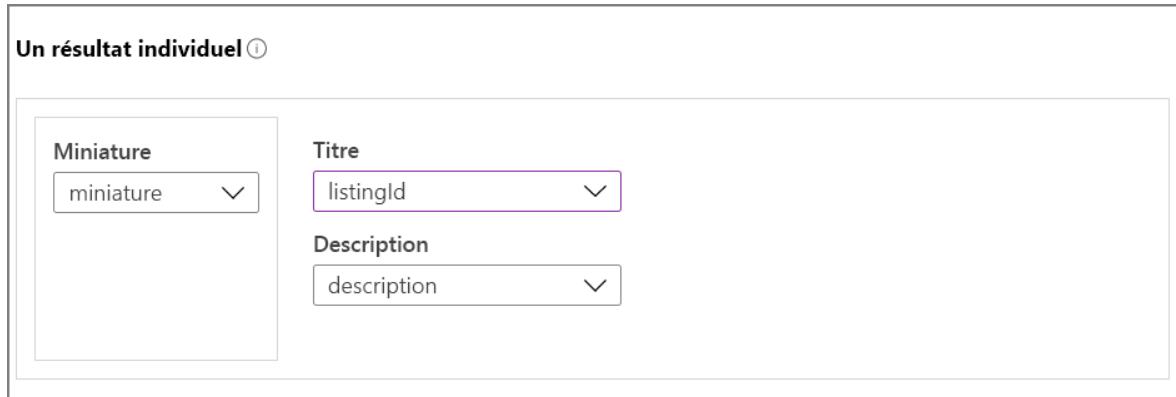
## Démarrer l'Assistant

1. Connectez-vous au [portail Azure](#) avec votre compte Azure.
2. **Recherchez votre service de recherche.** Ensuite, dans la page Vue d'ensemble, à partir des liens situés au milieu de la page, sélectionnez **Index**.
3. Choisissez *realestate-us-sample-index* dans la liste des index existants.
4. Dans la page de l'index, en haut, sélectionnez **Créer une application de démonstration (préversion)** pour démarrer l'Assistant.
5. Dans la première page de l'Assistant, sélectionnez **Activer le partage des ressources inter-origines (CORS)** pour ajouter la prise en charge du mécanisme CORS à votre définition d'index. Cette étape est facultative, mais votre application web locale ne se connecte pas à l'index distant sans elle.

## Configurer les résultats de la recherche

L'Assistant fournit une disposition de base pour l'affichage des résultats de la recherche qui inclut un espace pour une image miniature, un titre et une description. En complément de tous ces éléments se trouve un champ dans votre index qui fournit les données.

1. Dans Miniature, choisissez le champ *miniature* dans l'index *realestate-us-sample*. Cet exemple inclut des miniatures d'image sous la forme d'images contenant des adresses URL stockées dans un champ appelé *miniature*. Si votre index n'a pas d'images, laissez ce champ vide.
2. Dans Titre, choisissez un champ qui reflète l'unicité de chaque document. Dans cet exemple, l'ID de liste est une sélection acceptable.
3. Dans Description, choisissez un champ qui fournit des détails pouvant faciliter la décision de cliquer ou non pour accéder à ce document particulier.



## Ajouter une barre latérale

Le service de recherche prend en charge la navigation par facettes, souvent rendue sous forme de barre latérale. Les facettes reposent sur des champs filtrables et à choix multiples, comme exprimé dans le schéma d'index.

Dans Recherche cognitive Azure, la navigation par facettes est une expérience de filtrage cumulatif. Au sein d'une catégorie, la sélection de plusieurs filtres développe les résultats (par exemple, en sélectionnant Seattle et Bellevue dans Ville). Parmi les catégories, la sélection de plusieurs filtres affine les résultats.

### TIP

Vous pouvez voir le schéma d'index complet dans le portail. Recherchez le lien **Définition d'index (JSON)** dans la page de vue d'ensemble de chaque index. Les champs éligibles à la navigation par facettes ont des attributs « `filtrable : true` » et « `à choix multiples : true` ».

Acceptez la sélection actuelle des facettes et passez à la page suivante.

## Ajouter la saisie semi-automatique

La fonctionnalité de saisie semi-automatique est disponible sous la forme de suggestions d'autocomplétion et de requête. L'Assistant prend en charge les suggestions de requête. En fonction des séquences de touches entrées par l'utilisateur, le service de recherche retourne la liste des chaînes de requête « complétées » pouvant être sélectionnées comme entrée.

Les suggestions sont activées sur des définitions de champs spécifiques. L'Assistant vous laisse la possibilité de configurer la quantité d'informations incluses dans une suggestion.

La capture d'écran suivante montre les options de l'Assistant, en regard d'une page rendue dans l'application. Vous pouvez voir comment les sélections de champs sont utilisées et comment l'option « Afficher le nom du champ » est utilisée pour inclure ou exclure un étiquetage au sein de la suggestion.

The screenshot shows the 'Create a search application (preview)' interface. The 'Suggestions' tab is selected. A search bar at the top right contains the text 'lake wa'. Below it, a list of suggestions is displayed, with the first item being 'numéro : 5019 rue : Lake Washington Boulevard South ville : Seattle région : wa codePays : us'. To the left of the suggestions, a red box highlights the 'Choisissez et personnalisez les champs' (Select and customize fields) section. Another red box highlights the 'Nom du champ' (Field name) column in the 'Style' section, which lists fields like 'numéro', 'rue', 'ville', 'région', and 'codePays'. To the right of the field names, a column labeled 'Afficher le nom du...' (Display the name of...) contains several checkboxes, some of which are checked.

## Créer, télécharger et exécuter

1. Sélectionnez **Créer une application de démonstration** pour générer le fichier HTML.
2. À l'invite, sélectionnez **Télécharger votre application** pour télécharger le fichier.
3. Ouvrez le fichier. Une page similaire à la capture d'écran suivante doit s'afficher. Entrez un terme et utilisez des filtres pour affiner les résultats.

L'index sous-jacent se compose de données générées et fictives qui ont été dupliquées dans tous les documents, et les descriptions ne correspondent parfois pas à l'image. Vous pouvez vous attendre à une expérience plus cohérente quand vous créez une application basée sur vos propres index.

[effacer le\(s\) filtre\(s\)](#)**▼ type**

- Appartement
- Maison (249)

1 - 50 sur 249



9383442

Il s'agit d'une maison de ville à vendre à découvert. Située à proximité d'un cours d'eau, cette propriété possède également une vue sur la mer, une piscine, un dressing spacieux et des moulures de plafond.



9382448

Il s'agit d'un duplex neuf situé au sein d'une résidence. Cette propriété possède une vue sur la mer et est située à proximité d'établissements scolaires. Elle possède également une piscine, un étage donnant sur de magnifiques chambres et des plafonds à voûte.



9383199

Il s'agit d'une maison sur plusieurs étages à vendre à découvert. Située dans un lotissement aux abords d'une étendue d'eau, cette propriété dispose d'appareils ménagers haut de gamme, de portes-fenêtres et d'un palier couvert.

## Nettoyer les ressources

Lorsque vous travaillez dans votre propre abonnement, il est recommandé, à la fin de chaque projet, de déterminer si vous avez toujours besoin des ressources que vous avez créées. Les ressources laissées en cours d'exécution peuvent vous coûter de l'argent. Vous pouvez supprimer les ressources une par une, ou choisir de supprimer le groupe de ressources afin de supprimer l'ensemble des ressources.

Vous pouvez rechercher et gérer les ressources dans le portail à l'aide des liens **Toutes les ressources** ou **Groupes de ressources** situés dans le volet de navigation de gauche.

Si vous utilisez un service gratuit, n'oubliez pas que vous êtes limité à trois index, indexeurs et sources de données. Vous pouvez supprimer des éléments un par un dans le portail pour ne pas dépasser la limite.

## Étapes suivantes

L'application par défaut est pratique pour une première exploration et les petites tâches, tandis que l'examen préalable des API vous aidera à comprendre les concepts et le workflow à un niveau plus approfondi :

[Création d'un index à l'aide du Kit de développement logiciel .NET](#)

# Démarrage rapide : Créer un ensemble de compétences cognitives pour la Recherche cognitive Azure dans le portail Azure

04/10/2020 • 18 minutes to read • [Edit Online](#)

Un ensemble de compétences est une fonctionnalité basée sur l'IA (intelligence artificielle) qui extrait des informations et une structure à partir de fichiers texte ou de fichiers image non différenciés et volumineux, et qui rend le contenu indexable et offrant des possibilités de recherche dans Recherche cognitive Azure.

Dans ce guide de démarrage rapide, vous allez combiner les services et les données du cloud Azure pour créer l'ensemble de compétences. Une fois que tout est en place, vous exécutez l'Assistant Importer des données dans le portail Azure pour tout préparer. Le résultat final est un index de recherche qui contient des données créées par le traitement IA, que vous pouvez interroger dans le portail ([Explorateur de recherche](#)).

## Prérequis

Avant de commencer la lecture cet article, vous devez disposer des éléments suivants :

- Compte Azure avec un abonnement actif. [Créez un compte gratuitement](#).
- Service Recherche cognitive Azure. [Créez un service ou recherchez un service existant](#) dans votre abonnement actuel. Vous pouvez utiliser un service gratuit pour ce guide de démarrage rapide.
- Un compte de stockage Azure avec un [stockage blob](#).

### NOTE

Ce guide de démarrage rapide utilise également [Azure Cognitive Services](#) pour l'intelligence artificielle. Parce que la charge de travail est vraiment petite, Cognitive Services est utilisé en arrière-plan pour traiter gratuitement jusqu'à 20 transactions. Cela signifie que vous pouvez effectuer cet exercice sans avoir à créer une ressource Cognitive Services supplémentaire.

## Configurer vos données

Dans les étapes suivantes, configurez un conteneur d'objets blob dans Stockage Azure pour stocker des fichiers de contenu hétérogènes.

1. [Téléchargez les exemples de données](#) consistant en un petit ensemble de fichiers de types différents.  
Décompressez les fichiers
2. [Créez un compte de stockage Azure ou recherchez un compte existant](#).
  - Choisissez la même région que celle de la Recherche cognitive Azure pour éviter des frais de bande passante.
  - Choisissez le type de compte StorageV2 (V2 à usage général) si vous souhaitez tester la fonctionnalité de base de connaissances plus tard, au cours d'une autre procédure pas à pas. Sinon, choisissez n'importe quel type.
3. Ouvrez les pages des services BLOB et créez un conteneur. Vous pouvez utiliser le niveau d'accès public par défaut.

4. Dans le conteneur, cliquez sur **Charger** pour charger les exemples de fichiers que vous avez téléchargés au cours de la première étape. Notez que vous disposez d'un large éventail de types de contenu, notamment des images et des fichiers d'application qui ne peuvent pas faire l'objet de recherches en texte intégral dans leurs formats natifs.

NOM	DERNIÈRE MODIFICATION	TYPE D'OBJET BLOB	Type de contenu
10-K-FY16.html	2018-04-02T21:10:35.000Z	Bloquer un objet BLOB	text/html
5074.clip_image002_6FE27E85.png	2018-04-02T21:10:39.000Z	Bloquer un objet BLOB	image/png
Cognitive Services and Content Intelligence.pptx	2018-04-02T21:10:50.000Z	Bloquer un objet BLOB	application/vnd.openxmlformat
Cognitive Services and Bots (spanish).pdf	2018-04-02T21:10:41.000Z	Bloquer un objet BLOB	application/pdf
guthrie.jpg	2018-04-02T21:10:23.000Z	Bloquer un objet BLOB	image/jpeg
MSFT_cloud_architecture_contoso.pdf	2018-04-02T21:10:35.000Z	Bloquer un objet BLOB	application/pdf
MSFT_FY17_10K.docx	2018-04-02T21:10:36.000Z	Bloquer un objet BLOB	application/vnd.openxmlformat
NYSE_LNKD_2015.PDF	2018-04-02T21:10:35.000Z	Bloquer un objet BLOB	application/pdf
redshirt.jpg	2018-04-02T21:10:31.000Z	Bloquer un objet BLOB	image/jpeg
satyanadellalinux.jpg	2018-04-02T21:10:33.000Z	Bloquer un objet BLOB	image/jpeg
satyasletter.txt	2018-04-02T21:10:22.000Z	Bloquer un objet BLOB	texte/brut

Vous êtes maintenant prêt à passer à l'Assistant Importation de données.

## Exécuter l'Assistant Importation de données

- Connectez-vous au [portail Azure](#) avec votre compte Azure.
- Recherchez votre service de recherche.** Ensuite, dans la page Vue d'ensemble, cliquez sur **Importer des données** sur la barre de commandes pour configurer l'enrichissement cognitif en quatre étapes.



### Étape 1 : Créer une source de données

- Dans **Connexion à vos données**, choisissez **Stockage Blob Azure**, sélectionnez le compte de stockage et le conteneur que vous avez créés. Donnez un nom à la source de données et utilisez les valeurs par défaut pour le reste.

Importer des données

Se connecter à vos données \* Ajouter les compétences cognitives (facultatif) Personnaliser l'index cible Créez un indexeur

Créez et chargez un index de recherche en utilisant les données d'une source de données Azure existante dans votre abonnement actuel. Le service Recherche Cognitive Azure analyse la structure des données que vous fournissez, extrait le contenu pouvant être recherché, l'enrichit éventuellement avec des qualifications cognitives et le charge dans un index. [En savoir plus](#)

Source de données	Stockage Blob Azure
Name *	demo-skillset-ds
Données à extraire	Contenu et métadonnées
Mode d'analyse	Par défaut
Chaîne de connexion *	DefaultEndpointsProtocol=https;AccountName=[accountName];AccountKe Choisir une connexion existante
Nom du conteneur *	
Dossier d'objets blob	votre/dossier/ici

Passez à la page suivante.

### Étape 2 : Ajouter des compétences cognitives

Ensuite, configurez l'enrichissement par IA pour appeler l'OCR, l'analyse des images et le traitement en langage

naturel.

1. Pour ce guide de démarrage rapide, nous utilisons la ressource Cognitive Services au niveau **Gratuit**. Les exemples de données se composent de 14 fichiers. L'allocation gratuite de 20 transactions sur Cognitive Services est donc suffisante pour ce guide de démarrage rapide.

### Importer des données

Se connecter à vos données \* [Ajouter les compétences cognitives \(facultatif\)](#) [Personnaliser l'index cible](#) [Créer un indexeur](#)

**i** Enrichissez et extrayez la structure de vos documents par le biais de qualifications cognitives qui utilisent les mêmes algorithmes IA que Cognitive Services. Sélectionnez les options de décodage de document et les qualifications cognitives à appliquer à vos documents. Si besoin, enregistrez les documents enrichis dans le stockage Azure pour une utilisation dans des scénarios autres que des recherches. [En savoir plus](#)

^ Attacher Cognitive Services

Pour utiliser vos qualifications cognitives, sélectionnez une ressource Cognitive Services existante ou créez-en une. La ressource Cognitive Services doit être dans la même région que votre service de recherche cognitive Azure. L'exécution des qualifications cognitives est facturée sur la ressource sélectionnée. Sinon, le nombre d'enrichissements exécutés est limité. [En savoir plus](#)

[Actualiser](#)

Nom de la ressource Cognitive services	↑↓	Région	↑↓
Gratuit (enrichissements limités)			
<a href="#">Créer une ressource Cognitive Services</a>			

2. Développez **Ajouter des enrichissements** et effectuez quatre sélections.

Activez l'OCR pour ajouter des compétences d'analyse d'images à la page de l'Assistant.

Définissez la précision sur Pages pour scinder le texte en morceaux plus petits. Plusieurs compétences en texte sont limitées à des entrées de 5 Ko.

Choisissez des compétences de reconnaissance d'entité (personnes, organisations, emplacements) et d'analyse d'image.

## Importer des données

▲ Ajouter des enrichissements

Exécutez des qualifications cognitives sur un champ de données source pour créer des champs de recherche supplémentaires. [Découvrez d'autres qualifications et l'extensibilité ici.](#)

Nom de l'ensemble de qualifications \* (1)

azureblob-skillset

Activer l'OCR et fusionner tout le texte dans le **champmerged\_content** (1)

Champ de données source \*

merged\_content

Niveau de précision d'enrichissement (1)

Pages (segments de 5 000 caractères)

Compétences cognitives de texte	Paramètre	Nom du champ
<input checked="" type="checkbox"/> Extraire les noms de personne		contacts
<input checked="" type="checkbox"/> Extraire les noms d'organisation		organisations
<input checked="" type="checkbox"/> Extraire les noms d'emplacement		emplacements
<input type="checkbox"/> Extraire les phrases clés		phrases clés
<input type="checkbox"/> Déetecter la langue		language
<input type="checkbox"/> Traduire le texte	Langue cible <span style="border: 1px solid #ccc; padding: 2px;">Anglais</span>	translated_text
<input type="checkbox"/> Extraire des informations d'identification personnelle		pii_entities
<input type="checkbox"/> Déetecter le sentiment		sentiment

Compétences cognitives d'image	Nom du champ
<input checked="" type="checkbox"/> Générer les balises des images	imageTags
<input checked="" type="checkbox"/> Générer les légendes des images	imageCaption
<input type="checkbox"/> Identifier des célébrités dans les images	imageCelebrities

Passez à la page suivante.

### Étape 3 : Configurer l'index

Un index contient le contenu pouvant faire l'objet de recherches. L'Assistant **Importation de données** peut généralement créer le schéma à votre place en échantillonnant la source de données. Au cours de cette étape, examinez le schéma généré, puis modifiez éventuellement les paramètres. Vous trouverez ci-dessous le schéma par défaut créé pour le jeu de données d'objets blob de démonstration.

Pour ce guide de démarrage rapide, l'Assistant effectue un travail de qualité en termes de définition de valeurs par défaut raisonnables :

- Les champs par défaut sont basés sur les propriétés des objets blob existants ainsi que sur les nouveaux champs destinés à contenir une sortie d'enrichissement (par exemple, `people`, `organizations`, `locations`). Les types de données sont déduits à partir des métadonnées et des échantillonnages de données.
- La clé de document par défaut est `metadata_storage_path` (ce champ est sélectionné, car il contient des valeurs uniques).
- Les attributs par défaut sont **Récupérable** et **PossibilitéRecherche**. **PossibilitéRecherche** permet la recherche en texte intégral dans un champ. **Récupérable** signifie que les valeurs des champs peuvent être retournées dans les résultats. L'Assistant suppose que vous souhaitez ces champs récupérables et interrogables, car vous les avez créés par l'intermédiaire de compétences.

NOM DU CHAMP	TYPE	RÉCUPÉRABLE	FILTERABLE	SORTABLE	À CHOIX MUL...	RECHERCHE...	ANALYSEUR	SUGGESTEUR
contenu	Edm.String	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Standard - Lucene	<input type="checkbox"/>
metadata_storage_content_type	Edm.String	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		<input type="checkbox"/>
metadata_storage_size	Edm.Int64	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		<input type="checkbox"/>
metadata_storage_last_modified	Edm.DateTime...	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		<input type="checkbox"/>
metadata_storage_name	Edm.String	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		<input type="checkbox"/>
metadata_storage_path	Edm.String	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		<input type="checkbox"/>
metadata_content_encoding	Edm.String	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		<input type="checkbox"/>
metadata_content_type	Edm.String	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		<input type="checkbox"/>
metadata_language	Edm.String	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		<input type="checkbox"/>
metadata_title	Edm.String	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		<input type="checkbox"/>
contacts	Collection(Ed...	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
organisations	Collection(Ed...	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
emplacements	Collection(Ed...	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Remarquez la zone barrée et le point d'interrogation sur l'attribut **Récupérable** près du champ `content`. Pour les documents d'objets blob comportant beaucoup de texte, le champ `content` contient la majeure partie du fichier qui peut atteindre des milliers de lignes. Un champ comme celui-ci pouvant alourdir les résultats de la recherche, vous devez l'exclure de cette démonstration.

Toutefois, si vous devez transmettre le contenu du fichier au code client, assurez-vous que **Récupérable** reste sélectionné. Sinon, pensez à décocher cet attribut sur `content` si les éléments extraits (tels que `people`, `organizations`, `locations`, etc.) sont suffisants.

Marquer un champ comme étant **Récupérable** ne signifie pas que le champ *doit* être présent dans les résultats de recherche. Vous pouvez contrôler avec précision la composition des résultats de recherche à l'aide du paramètre de requête `$select` pour spécifier les champs à inclure. Pour les champs comportant beaucoup de texte, tels que `content`, le paramètre `$select` constitue votre solution pour fournir aux utilisateurs humains de votre application des résultats de recherche faciles à gérer, tout en garantissant au code client l'accès à toutes les informations dont il a besoin via l'attribut **Récupérable**.

Passez à la page suivante.

#### Étape 4 : Configurer l'indexeur

L'indexeur est une ressource de niveau supérieur qui gère le processus d'indexation. Il spécifie le nom de la source de données, un index cible et la fréquence d'exécution. L'Assistant **Importation de données** crée plusieurs objets, dont un indexeur qui est toujours présent et que vous pouvez exécuter à plusieurs reprises.

1. Dans la page **Indexeur**, vous pouvez accepter le nom par défaut et cliquer sur l'option de planification **Une fois** pour l'exécuter immédiatement.

## Importer des données

Se connecter à vos données \* Ajouter les compétences cognitives... Personnaliser... \* Créer un indexeur \*

Indexeur

Nom \*

Planification ⓘ Une fois Toutes les heures Quotidienne Personnalisé

Description

Options avancées

Précédent : Personnaliser l'index cible Envoyer

2. Cliquez sur Envoyer pour créer et exécuter simultanément l'indexeur.

## Superviser l'état

L'indexation des compétences cognitives prend plus de temps que l'indexation textuelle classique, notamment l'OCR et l'analyse d'image. Pour superviser la progression, accédez à la page Vue d'ensemble, puis cliquez sur Indexeurs au centre de la page.

Utilisation	Supervision	Index	Indexeurs	Sources de données	Ensemble de compétences
État	↑↓	Nom	↑↓	Dernière exécution ↑↓	Documents ayant... ↑↓
⚠ Avertissement		azureblob-indexer		À l'instant	14/14

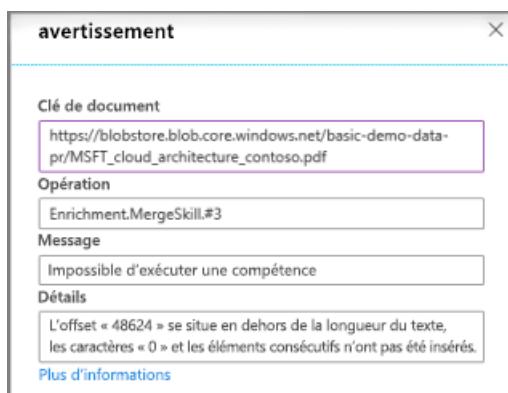
Les avertissements sont normaux compte tenu de la large gamme de types de contenu. Certains types de contenu ne sont pas valides pour certaines compétences et, à des niveaux inférieurs, il est courant de rencontrer des [limites d'indexeur](#). Par exemple, les notifications de troncation de 32 000 caractères sont une limite d'indexeur au niveau Gratuit. Si vous exécutez cette démonstration à un niveau supérieur, de nombreux avertissements de troncation disparaîtront.

Pour vérifier les avertissements ou les erreurs, cliquez sur l'état Avertissement dans la liste Indexeurs afin d'ouvrir la page Historique d'exécution.

Dans cette page, recliquez sur l'état Avertissement pour afficher la liste des avertissements similaires à celui illustré ci-dessous.

Documents ayant réussi	14/14			
Erreurs/Avertissement	0/6			
<input type="button"/> Rechercher par clé de document, opération, mes...				
Grouper par : <b>Gravité</b>	Gravité : <b>Tous</b>	Clé de document : <b>Tous</b>	Opération : <b>Tous</b>	Message : <b>Tous</b>
<b>Gra...</b> <b>Clé de document</b> <b>Opération</b> <b>Message</b>				
<b>avertissement</b>				
<a href="https://blobstore.blob.core.windows.net/basic-demo-data/pr/MSFT_cloud_architecture_contoso.pdf">https://blobstore.blob.core.windows.net/basic-demo-data-pr/MSFT_cloud_architecture_contoso.pdf</a>	DocumentExtraction.azureblob...	Texte extrait tronqué à « 32768 » caractères.		
<a href="https://blobstore.blob.core.windows.net/basic-demo-data/pr/MSFT_cloud_architecture_contoso.pdf">https://blobstore.blob.core.windows.net/basic-demo-data-pr/MSFT_cloud_architecture_contoso.pdf</a>	DocumentExtraction.azureblob...	Texte extrait tronqué à « 32768 » caractères.		
<a href="https://blobstore.blob.core.windows.net/basic-demo-data/pr/MSFT_cloud_architecture_contoso.pdf">https://blobstore.blob.core.windows.net/basic-demo-data-pr/MSFT_cloud_architecture_contoso.pdf</a>	DocumentExtraction.azureblob...	Texte extrait tronqué à « 32768 » caractères.		
<a href="https://blobstore.blob.core.windows.net/basic-demo-data/pr/MSFT_cloud_architecture_contoso.pdf">https://blobstore.blob.core.windows.net/basic-demo-data-pr/MSFT_cloud_architecture_contoso.pdf</a>	DocumentExtraction.azureblob...	Texte extrait tronqué à « 32768 » caractères.		
<a href="https://blobstore.blob.core.windows.net/basic-demo-data/pr/MSFT_cloud_architecture_contoso.pdf">https://blobstore.blob.core.windows.net/basic-demo-data-pr/MSFT_cloud_architecture_contoso.pdf</a>	Enrichment.MergeSkill.#3	Impossible d'exécuter une compétence		
<a href="https://blobstore.blob.core.windows.net/basic-demo-data/pr/MSFT_cloud_architecture_contoso.pdf">https://blobstore.blob.core.windows.net/basic-demo-data-pr/MSFT_cloud_architecture_contoso.pdf</a>	Enrichment.SplitSkill.#1	Impossible d'exécuter la compétence parce qu'une entrée...		

Les détails s'affichent quand vous cliquez sur une ligne d'état spécifique. Cet avertissement indique que la fusion s'est arrêtée après avoir atteint un seuil maximal (le fichier PDF en question est volumineux).



## Requête dans l'Explorateur de recherche

Après la création d'un index, vous pouvez exécuter des requêtes pour retourner des résultats. Dans le portail, utilisez l'**Explorateur de recherche** pour cette tâche.

- Sur la page de tableau de bord du service de recherche, cliquez sur **Explorateur de recherche** sur la barre de commandes.
- Cliquez sur **Modifier l'index** en haut pour sélectionner l'index créé.
- Entrez une chaîne de recherche pour interroger l'index, telle que  
`search=Microsoft&$select=people,organizations,locations,imageTags`.

Les résultats sont retournés au format JSON, qui peut être long et difficile à lire, en particulier dans des documents volumineux provenant d'objets blob Azure. Voici quelques conseils pour effectuer des recherches dans cet outil :

- Ajoutez `$select` pour spécifier les champs à inclure dans les résultats.
- Utilisez CTRL-F pour rechercher des termes ou des propriétés spécifiques dans les résultats au format JSON.

Les chaînes de requête respectent la casse. Ainsi, si vous recevez un message « champ inconnu », consultez l'onglet **Champs** ou **Définition d'index (JSON)** pour vérifier le nom et la casse.

The screenshot shows the Azure Cognitive Search Explorer interface. At the top, there are tabs for 'Explorateur de recherche', 'Champs', 'CORS', 'Profils de score', and 'Définition d'index (JSON)'. Below these are sections for 'Chaîne de requête' (containing 'search=Microsoft&searchFields=organizations&\$select=people,organizations,locations,imageTags') and 'Version d'API' (set to '06/05/2019'). A 'Rechercher' button is also present. The main area is titled 'Résultats' and displays a JSON response. The response starts with a document containing an array of objects, one of which is highlighted in yellow and labeled 'Microsoft'. The JSON structure includes '@odata.context', 'value', '@search.score', 'people', 'organizations', 'locations', and 'imageTags' fields. The 'organizations' field contains an array with 'Microsoft' as the first item. The 'imageTags' field contains several tags like 'person', 'man', etc. The results page shows '1 sur 73' items.

```

1 {
2   "@odata.context": "https://mydemo.search.windows.net/indexes('azureblob-index2')/$metadata#docs(*)",
3   "value": [
4     {
5       "@search.score": 1.2411621,
6       "people": [],
7       "organizations": [
8         "Microsoft"
9       ],
10      "locations": [],
11      "imageTags": [
12        "person",
13        "man",
14        "human face",
15        "clothing",
16        "shirt",
17        "indoor",
18        "glasses"
19     ]
20   },

```

## Éléments importants à retenir

Vous avez créé votre premier ensemble de compétences et appris d'importants concepts utiles pour le prototypage d'une solution de recherche enrichie à l'aide de vos propres données.

Parmi les concepts clés, que nous espérons que vous avez compris, figurent la dépendance à l'égard des sources de données Azure. Un ensemble de compétences est lié à un indexeur. Les indexeurs sont spécifiques à Azure et à la source. Bien que ce guide de démarrage rapide utilise le stockage Blob Azure, d'autres sources de données Azure sont possibles. Pour plus d'informations, consultez [Indexeurs dans la Recherche cognitive Azure](#).

Un autre concept important est que les compétences fonctionnent sur les types de contenu et que, lors de l'utilisation de contenu hétérogène, certaines entrées sont ignorées. En outre, les fichiers ou les champs volumineux peuvent dépasser les limites d'indexeur de votre niveau de service. Il est normal de voir des avertissements quand ces événements se produisent.

La sortie est dirigée vers un index de recherche. Il existe un mappage entre les paires nom-valeur créées durant l'indexation et les champs individuels de votre index. En interne, le portail configure les [annotations](#) et définit un [ensemble de compétences](#), en établissant l'ordre des opérations et le flux général. Ces étapes sont masquées dans le portail, mais lorsque vous démarrez l'écriture de code, ces concepts deviennent importants.

Enfin, vous avez appris que vous pouvez vérifier le contenu en interrogeant l'index. Pour finir, la Recherche cognitive Azure fournit un index offrant des possibilités de recherche, que vous pouvez interroger à l'aide de la [syntaxe de requête simple](#) ou de la [syntaxe de recherche entièrement étendue](#). Un index contenant des champs enrichis ressemble à n'importe quel autre index. Si vous souhaitez intégrer des analyseurs standard ou personnalisés, des [profils de scoring](#), des [synonymes](#), des [filtres à facettes](#), une recherche géographique ou toute autre fonctionnalité liée à la Recherche cognitive Azure, vous pouvez le faire sans problème.

## Nettoyer les ressources

Lorsque vous travaillez dans votre propre abonnement, il est recommandé, à la fin de chaque projet, de déterminer si vous avez toujours besoin des ressources que vous avez créées. Les ressources laissées en cours d'exécution peuvent vous coûter de l'argent. Vous pouvez supprimer les ressources une par une, ou choisir de supprimer le groupe de ressources afin de supprimer l'ensemble des ressources.

Vous pouvez rechercher et gérer les ressources dans le portail à l'aide des liens [Toutes les ressources](#) ou

**Groupes de ressources** situés dans le volet de navigation de gauche.

Si vous utilisez un service gratuit, n'oubliez pas que vous êtes limité à trois index, indexeurs et sources de données. Vous pouvez supprimer des éléments un par un dans le portail pour ne pas dépasser la limite.

## Étapes suivantes

Vous pouvez créer des ensembles de compétences à l'aide du portail, du kit SDK .NET ou de l'API REST. Pour approfondir vos connaissances, essayez l'API REST avec Postman et d'autres exemples de données.

[Tutoriel : Extraire du texte et une structure à partir d'objets blob JSON à l'aide d'API REST](#)

### TIP

Si vous souhaitez répéter cet exercice ou essayer une autre procédure pas à pas d'enrichissement par IA, supprimez l'indexeur du portail. La suppression de l'indexeur réinitialise le compteur de transactions quotidiennes gratuites à zéro pour le traitement Cognitive Services.

# Démarrage rapide : Créer une base de connaissances Recherche cognitive Azure dans le portail Azure

04/10/2020 • 11 minutes to read • [Edit Online](#)

La base de connaissances est une fonctionnalité de la Recherche cognitive Azure qui conserve la sortie d'un pipeline de traitement de contenu en vue d'analyses ultérieures ou d'un traitement en aval.

Un pipeline accepte du contenu texte et image non structuré, applique l'intelligence artificielle alimentée par Cognitive Services (par exemple, la reconnaissance optique de caractères et le traitement en langage naturel) et génère de nouvelles structures et des informations qui n'existaient pas auparavant. L'un des artefacts physiques créés par un pipeline est une [base de connaissances](#), à laquelle vous pouvez accéder par le biais d'outils pour analyser et explorer le contenu.

Dans ce guide de démarrage rapide, vous combinez des services et des données dans le cloud Azure pour créer une base de connaissances. Une fois tout en place, vous allez exécuter l'Assistant **Importation de données** dans le portail pour tout rassembler. Vous affichez ensuite le contenu du texte d'origine et le contenu généré par l'intelligence artificielle dans le portail (avec l'[Explorateur Stockage](#)).

## Prérequis

Avant de commencer la lecture cet article, vous devez disposer des éléments suivants :

- Compte Azure avec un abonnement actif. [Créez un compte gratuitement](#).
- Service Recherche cognitive Azure. [Créez un service](#) ou [recherchez un service existant](#) dans votre abonnement actuel. Vous pouvez utiliser un service gratuit pour ce guide de démarrage rapide.
- Un compte de stockage Azure avec un [stockage blob](#).

### NOTE

Ce guide de démarrage rapide utilise également [Azure Cognitive Services](#) pour l'intelligence artificielle. Parce que la charge de travail est vraiment petite, Cognitive Services est utilisé en arrière-plan pour traiter gratuitement jusqu'à 20 transactions. Cela signifie que vous pouvez effectuer cet exercice sans avoir à créer une ressource Cognitive Services supplémentaire.

## Configurer vos données

Dans les étapes suivantes, configurez un conteneur d'objets blob dans Stockage Azure pour stocker des fichiers de contenu hétérogènes.

1. [Téléchargez le fichier HotelReviews\\_Free.csv](#). Ce fichier CSV contient des données d'avis d'hôtel (issues de Kaggle.com). Il rassemble 19 commentaires de clients relatifs à un seul hôtel.
2. [Créez un compte de stockage Azure](#) ou [recherchez un compte existant](#) dans votre abonnement actuel. Vous utilisez le stockage Azure pour le contenu brut à importer, mais aussi pour la base de connaissances qui est le résultat final.
  - Choisissez le type de compte **StorageV2 (usage général v2)** .
3. Ouvrez les pages des services Blob et créez un conteneur nommé *hotel-reviews*.

4. Cliquez sur Télécharger.



5. Sélectionnez le fichier HotelReviews-Free.csv que vous avez téléchargé à la première étape.

The screenshot shows the Azure Storage Blob container interface for the 'hotelreviews' container. On the left, there's a sidebar with options like Overview, Access Control (IAM), Settings (Access policy, Properties, Metadata, Editor (preview)), and a search bar. The main area shows blob details: Authentication method: Access key (Switch to Azure AD User Account), Location: hotelreviews-free. A table lists blobs: NAME, MODIFIED, ACCESS TIER, BLOB TYPE, SIZE, LEASE STATE. One row is shown: HotelReviews\_Free.csv, 7/29/2019, 11:57:58 AM, Hot (Infe...), Block blob, 8.84 KiB, Available, ...

6. Avant de quitter les pages du Stockage Blob, utilisez un lien dans le volet de navigation gauche pour ouvrir la page Clés d'accès. Obtenez une chaîne de connexion pour récupérer les données du Stockage Blob. Une chaîne de connexion ressemble à l'exemple suivant :

```
DefaultEndpointsProtocol=https;AccountName=<YOUR-ACCOUNT-NAME>;AccountKey=<YOUR-ACCOUNT-KEY>;EndpointSuffix=core.windows.net
```

Vous êtes maintenant prêt à passer à l'Assistant Importation de données.

## Exécuter l'Assistant Importation de données

1. Connectez-vous au [portail Azure](#) avec votre compte Azure.
2. [Trouvez votre service de recherche](#). Ensuite, dans la page Vue d'ensemble, cliquez sur Importer des données dans la barre de commandes pour créer une base de connaissances en quatre étapes.



### Étape 1 : Création d'une source de données

1. Dans Se connecter à vos données, choisissez Stockage Blob Azure, puis sélectionnez le compte et le conteneur que vous avez créés.
2. Pour Nom, entrez hotel-reviews-ds .
3. Pour Mode d'analyse, sélectionnez Texte délimité, puis cochez la case La première ligne contient l'en-tête. Vérifiez que le Caractère délimiteur est une virgule (,).
4. Dans Chaîne de connexion, collez la chaîne de connexion que vous avez copiée à partir de la page Clés d'accès dans Stockage Azure.
5. Dans Conteneurs, entrez le nom du conteneur d'objets blob contenant les données.

Votre page doit ressembler à la capture d'écran suivante.

## Import data

\* Connect to your data Add cognitive search (Optional) Customize target index Create an indexer

Create and load a search index using data from an existing Azure data source in your current subscription. Azure Search crawls the data structure you provide, extracts searchable content, optionally enriches it with cognitive skills, and loads it into an index. [Learn more](#)

Data Source

Azure Blob Storage

\* Name

hotel-reviews-ds

Data to extract i

Content and metadata

Parsing mode

Delimited text

First Line Contains Header i



Delimiter Character i

,

\* Connection string

DefaultEndpointsProtocol=https;AccountName=mydemoblobstore;...



[Choose an existing connection](#)

\* Container name i

hotel-reviews

**Next: Add cognitive search (Optional)**

6. Passez à la page suivante.

### Étape 2 : Ajouter des compétences cognitives

Dans cette étape de l'Assistant, vous allez créer un ensemble de compétences par enrichissement des compétences cognitives. Les données sources sont constituées des évaluations des clients dans plusieurs langues. Les compétences pertinentes pour ce jeu de données incluent l'extraction d'expressions clés, la détection de sentiments et la traduction de texte. Dans une étape ultérieure, ces enrichissements seront « projetés » dans une base de connaissances en tant que tables Azure.

1. Développez **Attacher Cognitive Services**. **Gratuit (enrichissements limités)** est sélectionné par défaut. Vous pouvez utiliser cette ressource, car le nombre d'enregistrements dans HotelReviews-Free.csv est de 19, et cette ressource gratuite autorise jusqu'à 20 transactions par jour.

2. Développez **Ajouter des enrichissements**.

3. Pour **Nom de l'ensemble de compétences**, entrez `hotel-reviews-ss`.

4. Pour **Champ de données source**, sélectionnez `reviews_text`.

5. Pour **Niveau de précision d'enrichissement**, sélectionnez **Pages (segments de 5 000 caractères)**

6. Sélectionnez les compétences cognitives suivantes :

- Extraire des expressions clés
- Traduire le texte
- Déetecter le sentiment

#### ▲ Ajouter des enrichissements

Exécutez des qualifications cognitives sur un champ de données source pour créer des champs de recherche...  
[Découvrez d'autres qualifications et l'extensibilité ici.](#)

\* Nom de l'ensemble de qualifications [i](#)

hotel-reviews-ss ✓

Activer l'OCR et fusionner tout le texte dans le champ **merged\_content** [i](#)

\* Champ de données source

reviews\_text

Niveau de précision d'enrichissement [i](#)

Pages (segments de 5 000 caractères)

##### Compétences cognitives de texte

Nom du champ

Extraire les noms de personne

contacts

Extraire les noms d'organisation

organisations

Extraire les noms d'emplacement

emplacements

Extraire les phrases clés

phrases clés

Déetecter la langue

langue

Traduire le texte

Langue cible Ang... ▾

translated\_text

Déetecter le sentiment

tendance

7. Développez **Enregistrer les enrichissements dans une base de connaissances**.

8. Sélectionnez les **projections de table Azure** suivantes :

- Documents
- Pages
- Phrases clés

9. Entrez la **Chaîne de connexion du compte de stockage**, que vous avez enregistrée à une étape précédente.

## Import data

- ▼ Attach Cognitive Services
- ▼ Add enrichments
- ^ Save enrichments to a knowledge store (Preview)

A knowledge store allows you to project your enriched documents into tables and blobs. [Learn more about knowledge store](#)

\* Storage account connection string

DefaultEndpointsProtocol=https;AccountName=mydemoblobstore;AccountKey=UEX7p07/DfdorTEBUGfR... ✓

[Choose an existing connection](#)

### Azure table projections

- Documents
- Pages
- Key phrases

### Azure blob projections

- Document

[Previous: Connect to your data](#)

[Next: Customize target index](#)

10. Éventuellement, téléchargez un modèle Power BI. Quand vous accédez au modèle à partir de l'Assistant, le fichier .pbix local est adapté pour refléter la forme de vos données.

11. Passez à la page suivante.

### Étape 3 : Configurer l'index

Dans cette étape de l'Assistant, vous allez configurer un index pour d'éventuelles requêtes de recherche en texte intégral. L'Assistant va échantillonner votre source de données pour en déduire des champs et des types de données. Il vous suffit de sélectionner les attributs correspondant au comportement souhaité. Par exemple, l'attribut **Récupérable** permet au service de recherche de retourner une valeur de champ, alors que l'attribut **Possibilité de recherche** active la recherche en texte intégral sur le champ.

1. Pour **Nom de l'index**, entrez `hotel-reviews-idx`.

2. Pour les attributs, acceptez les sélections par défaut : **Récupérable** et **Possibilité de recherche** pour les champs que le pipeline crée.

Votre index doit ressembler à l'image suivante. Dans la mesure où la liste est longue, tous les champs ne sont pas visibles dans l'image.

### Importer des données

Ajouter un champ		Ajouter un sous-champ		Supprimer						
NOM DU CHAMP	TYPE	RÉCUPÉRABLE	FILTRABLE	TRIABLE	À CHOIX...	RECHERCHE...	ANALYSEUR			
phrases clés	Collection(E...)	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Standard - Luce...	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
translated_text	Collection(E...)	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Anglais - Lucene	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
tendance	Collection(E...)	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

[Précédent : Ajouter la recherche cognitive \(facultatif\)](#)

[Suivant : Crée un indexeur](#)

3. Passez à la page suivante.

#### Étape 4 : Configurer l'indexeur

Dans cette étape de l'Assistant, vous allez configurer un indexeur qui doit rassembler la source de données, l'ensemble de compétences et l'index que vous avez définis dans les étapes précédentes de l'Assistant.

1. Pour **Nom**, entrez `hotel-reviews-idxr`.

2. Pour **Planification**, conservez la valeur par défaut **Une fois**.

3. Cliquez sur **Envoyer** pour exécuter l'indexeur. Les opérations d'extraction de données, d'indexation et d'application des compétences cognitives se produisent toutes à cette étape.

## Superviser l'état

L'indexation des compétences cognitives prend plus de temps que l'indexation standard basée sur du texte. L'Assistant doit ouvrir la liste de l'indexeur sur la page Vue d'ensemble afin que vous puissiez suivre la progression. Pour une navigation automatique, accédez à la page Vue d'ensemble et cliquez sur **Indexeurs**.

Dans le portail Azure, vous pouvez également superviser le journal d'activité Notifications, et rechercher le lien d'état cliquable **Notification Recherche cognitive Azure**. L'exécution peut prendre plusieurs minutes.

## Étapes suivantes

Une fois que vous avez enrichi vos données à l'aide de Cognitive Services et que vous avez projeté les résultats dans une base de connaissances, vous pouvez utiliser l'Explorateur Stockage ou Power BI pour explorer votre jeu de données enrichi.

Vous pouvez afficher le contenu dans l'Explorateur Stockage, ou aller un peu plus loin avec Power BI pour obtenir des insights par le biais de la visualisation.

[Afficher avec l'Explorateur Stockage](#) [Se connecter à Power BI](#)

#### TIP

Si vous souhaitez répéter cet exercice ou essayer une autre procédure pas à pas d'enrichissement par IA, supprimez l'indexeur `hotel-reviews-idxr`. La suppression de l'indexeur réinitialise le compteur de transactions quotidiennes gratuites à zéro pour le traitement Cognitive Services.

# Démarrage rapide : Utiliser l'Explorateur de recherche pour exécuter des requêtes dans le portail

04/10/2020 • 10 minutes to read • [Edit Online](#)

L'**Explorateur de recherche** est un outil de requête intégré utilisé pour exécuter des requêtes sur un index de recherche dans Recherche cognitive Azure. Cet outil facilite l'apprentissage de la syntaxe des requêtes, le test d'une requête ou d'une expression de filtre, ou la confirmation de l'actualisation des données en vérifiant si du nouveau contenu existe dans l'index.

Ce guide de démarrage rapide utilise un index existant pour illustrer l'explorateur de recherche. Les requêtes sont formulées à l'aide de [l'API REST Search](#), avec les réponses retournées sous forme de documents JSON.

## Prérequis

Avant de commencer la lecture cet article, vous devez disposer des éléments suivants :

- Compte Azure avec un abonnement actif. [Créez un compte gratuitement](#).
- Service Recherche cognitive Azure. [Créez un service](#) ou [recherchez un service existant](#) dans votre abonnement actuel. Vous pouvez utiliser un service gratuit pour ce guide de démarrage rapide.
- *realestate-us-sample-index* est utilisé pour ce guide de démarrage rapide. Utilisez l'Assistant [Importer des données](#) pour créer l'index. Dans la première étape, lorsque vous êtes invité à entrer la source de données, choisissez **Exemples** puis sélectionnez la source de données **realestate-us-sample**. Acceptez tous les paramètres par défaut de l'Assistant pour créer l'index.

## Démarrer l'Explorateur de recherche

1. Dans le [portail Azure](#), ouvrez la page du service de recherche à partir du tableau de bord, ou [recherchez votre service](#).
2. Ouvrez l'Explorateur de recherche dans la barre de commandes :



Ou utilisez l'onglet **Explorateur de recherche** intégré sur un index ouvert :

A screenshot of the 'Search explorer' interface within the Azure portal. The top navigation bar shows 'Home &gt; mydemo &gt; realestate-us-sample-index'. The main area has tabs for 'Search explorer' (which is highlighted with a red box), 'Fields', 'CORS', 'Scoring profiles', and 'Index Definition (JSON)'. Below the tabs, there's a 'Query string' input field containing 'Examples: \*, \$top=10, \$skip=10&amp;\$search='\*, an 'API version' dropdown set to '2019-05-06', and a 'Search' button. At the bottom, there's a 'Request URL' field with the value 'https://mydemo.search.windows.net/indexes/realestate-us-sample-index/docs?api-version=2019-05-06&amp;search='\*'. There are also 'Save' and 'Discard' buttons at the bottom left.

## Requête non spécifiée

Pour un premier aperçu du contenu, exécutez une recherche vide en cliquant sur **Rechercher** sans indiquer de termes. Une recherche vide est utile en tant que première requête, car elle renvoie des documents entiers pour vous permettre d'examiner leur composition. Une recherche vide ne présente aucun classement et les documents sont renvoyés dans une ordre arbitraire ( "`@search.score": 1` pour tous les documents). Par défaut, 50 documents sont renvoyés dans une requête de recherche.

La syntaxe équivalente d'une recherche vide est `*` ou `search=*`.

```
search=*
```

### Résultats

The screenshot shows the Azure portal's search interface. In the top-left, there's a "Query string" input field containing "search=\*". To its right is a "Search" button. On the far right, it says "Index: realestate-us-sample" and "API version: 2017-11-11". Below these, the "Request URL" is shown as "https://[REDACTED].search.windows.net/indexes/realestate-us-sample/docs?api-version=2017-11-11&search=\*". The main area is titled "Results" and contains a JSON snippet:

```
1 {  
2   "@odata.context": "https://[REDACTED].search.windows.net/indexes  
('realestate-us-sample')/$metadata#docs(*)",  
3   "value": [  
4     {  
5       "@search.score": 1,  
6       "listingId": "9382104",  
7       "beds": 3,  
8       "baths": 3,  
9       "description": "This is a multi-storey house and is priced to sell. This home  
provides coastal views located within walking distance of a beach and features a lap pool,  
heated towel racks and vaulted ceilings.",  
10      "location": {  
11        "lat": 47.613221,  
12        "long": -122.332071  
13      },  
14      "photos": [  
15        "https://[REDACTED].blob.core.windows.net/[REDACTED]/[REDACTED].jpg",  
16        "https://[REDACTED].blob.core.windows.net/[REDACTED]/[REDACTED].jpg",  
17        "https://[REDACTED].blob.core.windows.net/[REDACTED]/[REDACTED].jpg",  
18        "https://[REDACTED].blob.core.windows.net/[REDACTED]/[REDACTED].jpg",  
19        "https://[REDACTED].blob.core.windows.net/[REDACTED]/[REDACTED].jpg"  
20      ]  
21    }  
22  ]  
23}
```

## Recherche en texte libre

Les requêtes de forme libre, avec ou sans opérateurs, permettent de simuler les requêtes définies par l'utilisateur envoyées à partir d'une application personnalisée vers la Recherche cognitive Azure. Seuls les champs attribués en tant que **Recherche possible** dans la définition de l'index sont analysés pour rechercher les correspondances.

Notez que lorsque vous indiquez des critères de recherche, comme des termes ou expressions de requête, le classement de la recherche intervient. L'exemple suivant illustre une recherche de texte libre.

```
Seattle apartment "Lake Washington" miele OR thermador appliance
```

### Résultats

Vous pouvez utiliser Ctrl+F pour rechercher des termes spécifiques dans les résultats.

Query string ⓘ  
Seattle apartments "Lake Washington" miele OR thermador appliance

Search Index: **realestate-us-sample**  
API version: **2017-11-11**

Request URL  
<https://...search.windows.net/indexes/realestate-us-sample/docs?api-version=2017-11-11&search=Seattle%20apartments%20...>

Results

```

5   "@search.score": 0.6959,
6   "listingId": "9383988",
7   "beds": 2,
8   "baths": 2,
9   "description": "This is a flatlet but has a small mouse problem. This home
provides lakefront property located in a culs-de-sac and features thermador appliances,
wood floors and a guest house."

```

## Nombre de documents correspondants

Ajoutez **\$count=true** pour obtenir le nombre de correspondances trouvées dans un index. Dans une recherche vide, ce nombre correspond au nombre total de documents dans l'index. Dans une recherche qualifiée, il correspond au nombre de documents correspondant à l'entrée de requête.

```
$count=true
```

### Résultats

Query string ⓘ  
**\$count=true**

Search Index: **realestate-us-sample**  
API version: **2017-11-11**

Request URL  
<https://...search.windows.net/indexes/realestate-us-sample/docs?api-version=2017-11-11&search=&%24count=true>

Results

```

1  {
2   "@odata.context": "https://...search.windows.net/indexes('realestate-us-sample')/$metadata#docs(*)",
3   "@odata.count": 4959,
4   "value": [
5     {
6       "@search.score": 1,
7       "listingId": "9382104",
8       "beds": 3,
9       "baths": 3

```

## Limiter les champs dans les résultats de la recherche

Ajoutez **\$select** pour limiter les résultats aux champs explicitement nommés et disposer d'une sortie plus lisible dans l'**Explorateur de recherche**. Pour conserver la chaîne de recherche et **\$count = true**, faites précéder les arguments de & .

```
search=seattle condo&$select=listingId,beds,baths,description,street,city,price&$count=true
```

### Résultats

Query string ⓘ  
search=seattle condo&\$select=listingId,beds,baths,description,street,city,price&\$count=true

Search Index: **realestate-us-sample**  
API version: **2017-11-11**

Request URL  
https://search.windows.net/indexes/realestate-us-sample/docs?api-version=2017-11-11&search=seattle%20condo&%24select...

Results

```

1  {
2      "@odata.context": "https://[REDACTED].search.windows.net/indexes
('realestate-us-sample')/$metadata#docs(*)",
3      "@odata.count": 2056,
4      "value": [
5          {
6              "@search.score": 0.4378676,
7              "listingId": "9382100",
8              "beds": 5,
9              "baths": 5,
10             "description": "This is a condo and is brand new. This property has lake
access located close to schools and features a steam room, crown mouldings and a guest
room.",
11             "street": "23rd Avenue South",
12             "city": "Seattle",
13             "price": 4665600
14         },

```

## Retourner le jeu de résultats suivant

La Recherche cognitive Azure retourne les 50 premières correspondances selon le classement de la recherche. Pour obtenir le jeu de documents correspondants suivant, ajoutez **\$top = 100 & \$skip = 50** afin d'augmenter le jeu de résultats à 100 documents (50 par défaut, 1 000 maximum), en ignorant les 50 premiers documents. N'oubliez pas qu'il vous faut fournir des critères de recherche, terme ou expression de requête, pour obtenir des résultats classés. Notez que les scores de recherche diminuent au fil des résultats de la recherche.

```
search=seattle
condo&$select=listingId,beds,baths,description,street,city,price&$count=true&$top=100&$skip=50
```

## Résultats

Query string ⓘ  
\$select=listingId,beds,baths,description,street,city,price&\$count=true&\$top=100&\$skip=50

Search Index: **realestate-us-sample**  
API version: **2017-11-11**

Request URL  
https://[REDACTED].search.windows.net/indexes/realestate-us-sample/docs?api-version=2017-11-11&search=seattle%20condo&%24select...

Results

```

1  {
2      "@odata.context": "https://[REDACTED].search.windows.net/indexes
('realestate-us-sample')/$metadata#docs(*)",
3      "@odata.count": 2056,
4      "value": [
5          {
6              "@search.score": 0.38382116,
7              "listingId": "9383937",
8              "beds": 1,
9              "baths": 1,

```

## Filtrer les expressions (supérieur à, inférieur à, égal à)

Utilisez le paramètre **\$filter** lorsque vous souhaitez spécifier des critères précis plutôt qu'une recherche de texte libre. Le champ doit être attribué comme **Filtrable** dans l'index. Cet exemple recherche plus de 3 chambres :

```
search=seattle condo&$filter=beds gt 3&$count=true
```

## Résultats

Query string ⓘ  
search=seattle condo&\$filter=beds gt 3&\$count=true

Request URL  
[https://\[REDACTED\].search.windows.net/indexes/realestate-us-sample/docs?api-version=2017-11-11&search=seattle%20condo&%24filter=...](https://[REDACTED].search.windows.net/indexes/realestate-us-sample/docs?api-version=2017-11-11&search=seattle%20condo&%24filter=...)

Results

```
1  {
2    "@odata.context": "https://[REDACTED].search.windows.net/indexes('realestate-us-sample')/$metadata#docs(*)",
3    "@odata.count": 805,
4    "value": [
5      {
6        "@search.score": 0.4378676,
7        "listingId": "9382100",
8        "beds": 5,
9        "baths": 5,
10       "description": "This is a condo and is brand new. This property has lake access located close to schools and features a steam room, crown mouldings and a guest room.",
```

## Expressions OrderBy

Ajoutez **\$orderby** pour trier les résultats selon un autre champ, à côté du score de recherche. Le champ doit être attribué comme **Triable** dans l'index. Pour le tester, vous pouvez utiliser l'exemple d'expression suivant :

```
search=seattle condo&$select=listingId,beds,price&$filter=beds gt 3&$count=true&$orderby=price asc
```

## Résultats

Query string ⓘ  
e condo&\$select=listingId,beds,price&\$filter=beds gt 3&\$count=true**&\$orderby=price asc**

Request URL  
[https://\[REDACTED\].search.windows.net/indexes/realestate-us-sample/docs?api-version=2017-11-11&search=seattle%20condo&%24select...](https://[REDACTED].search.windows.net/indexes/realestate-us-sample/docs?api-version=2017-11-11&search=seattle%20condo&%24select...)

Results

```
11  {
12    "@search.score": 0.05322655,
13    "listingId": "9385214",
14    "beds": 4,
15    "price": 518400
16  },
17  {
18    "@search.score": 0.05157484,
19    "listingId": "9382920",
20    "beds": 4,
21    "price": 520992
22  },
```

Les expressions **\$filter** et **\$orderby** sont des constructions OData. Pour plus d'informations, consultez l'article [Filter OData syntax](#) (Syntaxe d'expression de filtre OData).

## Éléments importants à retenir

Dans ce guide de démarrage rapide, vous avez utilisé l'**Explorateur de recherche** pour interroger un index à l'aide de l'API REST.

- Les résultats sont renvoyés sous forme de documents JSON détaillés afin de vous permettre de les

consulter dans leur intégralité. Vous pouvez utiliser des expressions de requête, comme indiqué dans les exemples, pour limiter les champs renvoyés.

- Les documents sont composés de tous les champs marqués comme **récupérables** dans l'index. Pour visualiser les attributs d'index dans le portail, cliquez sur *realestate-us-sample* dans la liste **Index** de la page de vue d'ensemble de la recherche.
- Les requêtes de forme libre, semblables à celles que vous entrez dans un navigateur web commercial, permettent de tester une expérience d'utilisateur final. Par exemple, dans le cas de l'exemple d'index *realestate* intégré, vous pourriez entrer « Seattle apartments lake washington », puis utiliser Ctrl+F pour rechercher des termes dans les résultats de la recherche.
- Les expressions de requête et de filtre sont articulées dans une syntaxe prise en charge par la Recherche cognitive Azure. Par défaut, cela correspond à une **syntaxe simple**, mais vous pouvez également utiliser **Lucene complète** pour de plus puissantes requêtes. Les **expressions de filtre** présentent une syntaxe OData.

## Nettoyer les ressources

Lorsque vous travaillez dans votre propre abonnement, il est recommandé, à la fin de chaque projet, de déterminer si vous avez toujours besoin des ressources que vous avez créées. Les ressources laissées en cours d'exécution peuvent vous coûter de l'argent. Vous pouvez supprimer les ressources une par une, ou choisir de supprimer le groupe de ressources afin de supprimer l'ensemble des ressources.

Vous pouvez rechercher et gérer les ressources dans le portail à l'aide des liens **Toutes les ressources** ou **Groupes de ressources** situés dans le volet de navigation de gauche.

Si vous utilisez un service gratuit, n'oubliez pas que vous êtes limité à trois index, indexeurs et sources de données. Vous pouvez supprimer des éléments un par un dans le portail pour ne pas dépasser la limite.

## Étapes suivantes

Pour en savoir plus sur la syntaxe et les structures des requêtes, utilisez Postman ou un outil équivalent pour créer des expressions de requête qui utilisent d'autres parties de l'API. **L'API REST Recherche** est particulièrement utile pour l'apprentissage et l'exploration.

[Créer une requête de base dans Postman](#)

# Démarrage rapide : Déployer Recherche cognitive à l'aide d'un modèle Resource Manager

04/10/2020 • 5 minutes to read • [Edit Online](#)

Cet article vous guide tout au long du processus d'utilisation d'un modèle Resource Manager (Azure Resource Manager) pour déployer une ressource Recherche cognitive Azure dans le portail Azure.

Un [modèle ARM](#) est un fichier JSON (JavaScript Object Notation) qui définit l'infrastructure et la configuration de votre projet. Le modèle utilise la syntaxe déclarative, qui vous permet d'indiquer ce que vous envisagez de déployer sans avoir à écrire la séquence de commandes de programmation pour le créer.

Si votre environnement remplit les prérequis et que vous êtes déjà familiarisé avec l'utilisation des modèles ARM, sélectionnez le bouton **Déployer sur Azure**. Le modèle s'ouvre dans le portail Azure.

## Prérequis

Si vous n'avez pas d'abonnement Azure, créez un [compte gratuit](#) avant de commencer.

## Vérifier le modèle

Le modèle utilisé dans ce démarrage rapide est tiré des [modèles de démarrage rapide Azure](#).

```
{  
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",  
  "contentVersion": "1.0.0.0",  
  "parameters": {  
    "name": {  
      "type": "string",  
      "minLength": 2,  
      "maxLength": 60,  
      "metadata": {  
        "description": "Service name must only contain lowercase letters, digits or dashes, cannot use dash as  
        the first two or last one characters, cannot contain consecutive dashes, and is limited between 2 and 60  
        characters in length."  
      }  
    },  
    "sku": {  
      "type": "string",  
      "defaultValue": "standard",  
      "allowedValues": [  
        "free",  
        "basic",  
        "standard",  
        "standard2",  
        "standard3",  
        "storage_optimized_l1",  
        "storage_optimized_l2"  
      ],  
      "metadata": {  
        "description": "The pricing tier of the search service you want to create (for example, basic or  
        standard)."  
      }  
    },  
    "replicaCount": {  
      "type": "int",  
      "defaultValue": 1,  
      "minValue": 1,  
      "maxValue": 10,  
      "metadata": {  
        "description": "The number of replicas you want to create for your search service."  
      }  
    }  
  },  
  "variables": {},  
  "resources": [],  
  "outputs": []  
}
```

```

    "defaultValue": 1,
    "minValue": 1,
    "maxValue": 12,
    "metadata": {
        "description": "Replicas distribute search workloads across the service. You need at least two replicas to support high availability of query workloads (not applicable to the free tier)."
    }
},
"partitionCount": {
    "type": "int",
    "defaultValue": 1,
    "allowedValues": [
        1,
        2,
        3,
        4,
        6,
        12
    ],
    "metadata": {
        "description": "Partitions allow for scaling of document count as well as faster indexing by sharding your index over multiple search units."
    }
},
"hostingMode": {
    "type": "string",
    "defaultValue": "default",
    "allowedValues": [
        "default",
        "highDensity"
    ],
    "metadata": {
        "description": "Applicable only for SKUs set to standard3. You can set this property to enable a single, high density partition that allows up to 1000 indexes, which is much higher than the maximum indexes allowed for any other SKU."
    }
},
"location": {
    "type": "string",
    "defaultValue": "[resourceGroup().location]",
    "metadata": {
        "description": "Location for all resources."
    }
},
"resources": [
{
    "type": "Microsoft.Search/searchServices",
    "apiVersion": "2020-03-13",
    "name": "[parameters('name')]",
    "location": "[parameters('location')]",
    "sku": {
        "name": "[toLowerCase(parameters('sku'))]"
    },
    "properties": {
        "replicaCount": "[parameters('replicaCount')]",
        "partitionCount": "[parameters('partitionCount')]",
        "hostingMode": "[parameters('hostingMode')]"
    }
}
]
}
}

```

La ressource Azure définie dans ce modèle :

- [Microsoft.Search/searchServices](#) : créer un service Recherche cognitive Azure

# Déployer le modèle

Cliquez sur l'image ci-après pour vous connecter à Azure et ouvrir un modèle. Le modèle crée une ressource Recherche cognitive Azure.

Le portail affiche un formulaire qui vous permet de spécifier facilement des valeurs de paramètre. Certains paramètres sont préremplis avec les valeurs par défaut du modèle. Vous devrez spécifier votre abonnement, le groupe de ressources, l'emplacement et le nom du service. Si vous souhaitez utiliser Cognitive Services dans un pipeline d'[enrichissement par IA](#), par exemple pour analyser le texte de fichiers image binaires, choisissez un emplacement qui offre à la fois le service Recherche cognitive et Cognitive Services. Les deux services doivent se trouver dans la même région pour les charges de travail d'enrichissement par IA. Quand vous avez rempli le formulaire, vous devez accepter les conditions générales, puis sélectionner le bouton Acheter pour terminer votre déploiement.

Tableau de bord > Service Recherche Azure

## Service Recherche Azure

Modèle de démarrage rapide Azure

---

### MODÈLE

101-azure-search-create  
1 ressource

Modifier un modèle Modifier le paramètre En savoir plus

---

### CONCEPTS DE BASE

Abonnement \*

Groupe de ressources \*   
Créer

Emplacement \*

---

### PARAMÈTRES

Name \*

Référence SKU  standard

Nombre de réplicas  1

Nombre de divisions  1

Mode d'hébergement  valeur par défaut

Emplacement  [resourceGroup().location]

---

### CONDITIONS GÉNÉRALES

Informations sur le modèle | Conditions de la Place de marché Azure | Place de marché Azure

En cliquant sur « Acheter », (a) j'accepte les conditions juridiques applicables associées à l'offre ; (b) j'autorise Microsoft à facturer ma méthode de paiement actuelle pour les frais associés aux offres, y compris les taxes applicables, avec la même fréquence de facturation que mon abonnement Azure, jusqu'à ce que j'arrête d'utiliser les offres ; et (c) j'accepte, en cas de déploiement incluant des offres de tiers, que Microsoft puisse partager mes informations de contact et d'autres détails d'un tel déploiement avec l'éditeur de cette offre.

J'accepte les termes et conditions mentionnés ci-dessus

---

**Purchase**

## Vérifier les ressources déployées

Quand votre déploiement est terminé, vous pouvez accéder au nouveau groupe de ressources et au nouveau service de recherche dans le portail.

## Nettoyer les ressources

D'autres guides de démarrage rapide et tutoriels sur le service Recherche cognitive reposent sur ce guide de démarrage rapide. Si vous prévoyez d'utiliser les guides de démarrage rapide et tutoriels suivants, il peut être utile de conserver ces ressources. Si vous n'en avez plus besoin, vous pouvez supprimer le groupe de ressources. Dans ce cas, le service Recherche cognitive et les ressources associées seront également supprimés.

## Étapes suivantes

Dans ce guide de démarrage rapide, vous avez créé un service Recherche cognitive en utilisant un modèle Resource Manager, puis vous avez validé le déploiement. Pour plus d'informations sur le service Recherche cognitive et Azure Resource Manager, consultez les articles ci-dessous.

- Consultez la [vue d'ensemble de Recherche cognitive Azure](#).
- [Créer un index](#) pour votre service de recherche.
- [Créer une application de démonstration](#) à l'aide de l'Assistant du portail.
- [Créer un ensemble de compétences](#) pour extraire des informations de vos données.

# Démarrage rapide : Créez un index de recherche à l'aide de la bibliothèque de client Azure.Search.Documents

04/10/2020 • 16 minutes to read • [Edit Online](#)

Utilisez la nouvelle [bibliothèque de client Azure.Search.Documents \(version 11\)](#) pour créer une application console .NET Core en C#, qui crée, charge et interroge un index de recherche.

Téléchargez le [code source](#) pour commencer avec un projet terminé ou suivez les étapes décrites dans cet article pour créer votre propre projet.

## NOTE

Vous recherchez une version antérieure ? Consultez [Créer un index de recherche à l'aide de Microsoft.Azure.Search v10](#) à la place.

## Prérequis

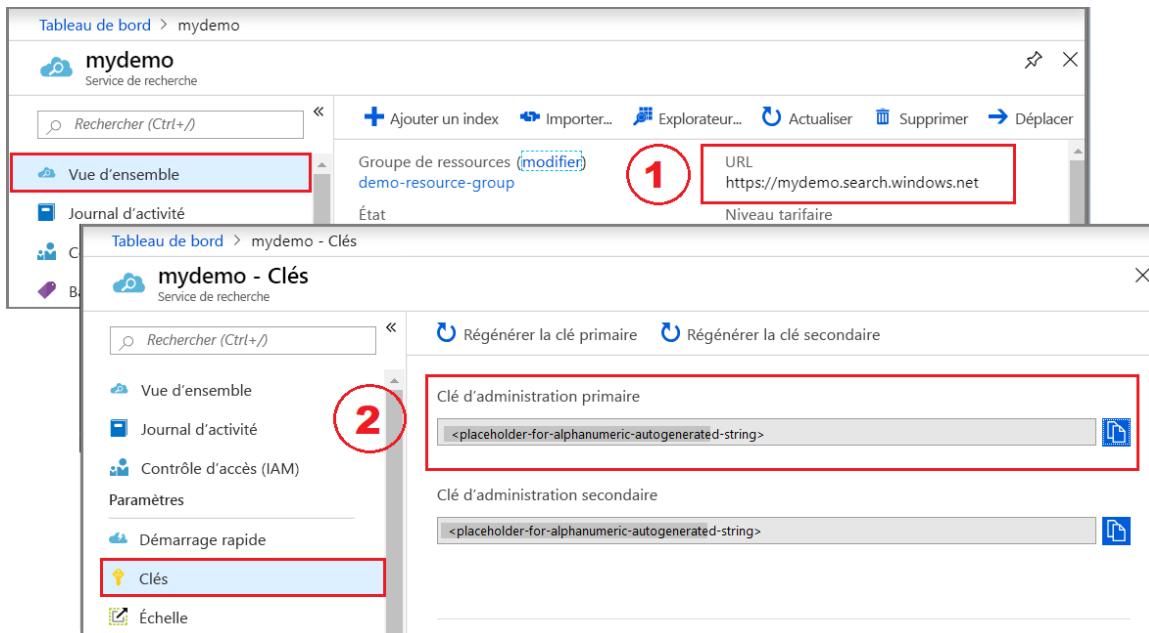
Avant de commencer, vous devez disposer des outils et services suivants :

- Compte Azure avec un abonnement actif. [Créez un compte gratuitement](#).
- Service Recherche cognitive Azure. [Créez un service](#) ou [recherchez un service existant](#). Vous pouvez utiliser un service gratuit pour ce guide de démarrage rapide.
- [Visual Studio](#), toute édition. L'exemple de code a été testé sur l'édition Community gratuite de Visual Studio 2019.

## Obtenir un clé et un point de terminaison

Les appels au service nécessitent un point de terminaison d'URL et une clé d'accès pour chaque requête. Un service de recherche est créé avec les deux. Ainsi, si vous avez ajouté la Recherche cognitive Azure à votre abonnement, effectuez ce qui suit pour obtenir les informations nécessaires :

1. [Connectez-vous au portail Azure](#), puis dans la page **Vue d'ensemble** du service de recherche, récupérez l'URL. Voici un exemple de point de terminaison : `https://mydemo.search.windows.net`.
2. Dans **Paramètres > Clés**, procurez-vous une clé d'administration pour disposer de droits complets sur le service, indispensables pour créer ou supprimer des objets. Il existe deux clés, primaire et secondaire, interchangeables. Vous pouvez utiliser celle de votre choix.



Toutes les demandes nécessitent une clé API sur chaque demande envoyée à votre service. L'utilisation d'une clé valide permet d'établir, en fonction de chaque demande, une relation de confiance entre l'application qui envoie la demande et le service qui en assure le traitement.

## Configuration de votre projet

Démarrez Visual Studio et créez un projet d'application console pouvant s'exécuter sur .NET Core.

### Installez le package NuGet

Une fois le projet créé, ajoutez la bibliothèque de client. Le [package Azure.Search.Documents](#) consiste en une bibliothèque de client qui fournit toutes les API utilisées pour utiliser un service de recherche dans .NET.

1. Dans **Outils > Gestionnaire de package NuGet**, sélectionnez **Gérer les packages NuGet pour la solution...**.
2. Cliquez sur **Parcourir**.
3. Recherchez **Azure.Search.Documents** et sélectionnez la version 11.0.0.
4. Cliquez sur **Installer** à droite pour ajouter l'assembly à vos projet et solution.

### Créer un client de recherche

1. Dans **Program.cs**, remplacez l'espace de noms par **AzureSearch.SDK.Quickstart.v11**, puis ajoutez les directives **using** suivantes.

```
using Azure;
using Azure.Search.Documents;
using Azure.Search.Documents.Indexes;
using Azure.Search.Documents.Indexes.Models;
using Azure.Search.Documents.Models;
```

2. Créez deux clients : Le client **SearchIndexClient** crée l'index et le client **SearchClient** fonctionne avec un index existant. Tous deux ont besoin du point de terminaison de service et d'une clé API d'administration pour l'authentification avec des droits de création/suppression.

```

static void Main(string[] args)
{
    string serviceName = "<YOUR-SERVICE-NAME>";
    string indexName = "hotels-quickstart-v11";
    string apiKey = "<YOUR-ADMIN-API-KEY>";

    // Create a SearchIndexClient to send create/delete index commands
    Uri serviceEndpoint = new Uri($"https://'{serviceName}'.search.windows.net/");
    AzureKeyCredential credential = new AzureKeyCredential(apiKey);
    SearchIndexClient idxclient = new SearchIndexClient(serviceEndpoint, credential);

    // Create a SearchClient to load and query documents
    SearchClient qryclient = new SearchClient(serviceEndpoint, indexName, credential);
}

```

## 1 – Créer un index

Ce démarrage rapide crée un index Hotels que vous allez charger à l'aide de données d'hôtel et sur lequel vous allez exécuter des requêtes. Dans cette étape, définissez les champs de l'index. Chaque définition de champ comprend un nom, un type de données et des attributs qui déterminent la façon dont le champ est utilisé.

Dans cet exemple, des méthodes synchrones de la bibliothèque Azure.Search.Documents sont utilisées à des fins de simplicité et de lisibilité. En revanche, dans des scénarios de production, vous devez utiliser des méthodes asynchrones pour maintenir la scalabilité et la réactivité de votre application. Par exemple, vous pouvez utiliser [CreateIndexAsync](#) au lieu de [CreateIndex](#).

1. Ajoutez une définition de classe vide à votre projet : **Hotel.cs**
2. Dans **Hotel.cs**, définissez la structure d'un document d'hôtel.

```

using System;
using System.Text.Json.Serialization;

namespace AzureSearch.SDK.Quickstart.v11
{
    public class Hotel
    {
        [JsonPropertyName("hotelId")]
        public string Id { get; set; }

        [JsonPropertyName("hotelName")]
        public string Name { get; set; }

        [JsonPropertyName("hotelCategory")]
        public string Category { get; set; }

        [JsonPropertyName("baseRate")]
        public Int32 Rate { get; set; }

        [JsonPropertyName("lastRenovationDate")]
        public DateTime Updated { get; set; }
    }
}

```

3. Dans **Program.cs**, spécifiez les champs et les attributs. [SearchIndex](#) et [CreateIndex](#) sont utilisés pour créer un index.

```

// Define an index schema using SearchIndex
// Create the index using SearchIndexClient
SearchIndex index = new SearchIndex(indexName)
{
    Fields =
    {
        new SimpleField("hotelId", SearchFieldDataType.String) { IsKey = true, IsFilterable =
true, IsSortable = true },
        new SearchableField("hotelName") { IsFilterable = true, IsSortable = true },
        new SearchableField("hotelCategory") { IsFilterable = true, IsSortable = true },
        new SimpleField("baseRate", SearchFieldDataType.Int32) { IsFilterable = true,
IsSortable = true },
        new SimpleField("lastRenovationDate", SearchFieldDataType.DateTimeOffset) {
IsFilterable = true, IsSortable = true }
    }
};

Console.WriteLine("{0}", "Creating index...\n");
idxclient.CreateIndex(index);

```

Les attributs du champ déterminent la façon dont il est utilisé dans une application. Par exemple, l'attribut `IsFilterable` doit être assigné à chaque champ qui prend en charge une expression de filtre.

Contrairement aux versions précédentes du kit de développement logiciel (SDK) .NET qui nécessitent la méthode `IsSearchable` sur des champs de chaîne offrant une possibilité de recherche, vous pouvez utiliser `SearchableField` et `SimpleField` pour simplifier les définitions de champs.

À l'instar des versions précédentes, d'autres attributs sont toujours requis sur la définition elle-même. Par exemple, les attributs `IsFilterable`, `IsSortable` et `IsFacetable` doivent être explicitement attribués, comme illustré dans l'exemple ci-dessus.

## 2 – Charger des documents

Le service Recherche cognitive Azure recherche dans le contenu stocké dans le service. Au cours de cette étape, vous allez charger des documents JSON conformes à l'index de l'hôtel que vous venez de créer.

Dans la Recherche cognitive Azure, les documents sont des structures de données qui sont à la fois des entrées pour l'indexation et des sorties de requêtes. Selon une source de données externe, les entrées de documents peuvent être des lignes dans une base de données, des objets blob dans le Stockage Blob ou des documents JSON sur le disque. Dans cet exemple, nous prenons un raccourci et incorporons des documents JSON pour cinq hôtels dans le code lui-même.

Lors du chargement de documents, vous devez utiliser un objet `IndexDocumentsBatch`. Un objet `IndexDocumentsBatch` contient une collection d'`Actions` contenant chacune un document et une propriété indiquant au service Recherche cognitive Azure l'action à effectuer (`upload`, `merge`, `delete` et `mergeOrUpload`).

1. Dans `Program.cs`, créez un tableau de documents et d'actions d'index, puis transmettez le tableau à `ndexDocumentsBatch`. Les documents ci-dessous sont conformes à l'index `hotels-quickstart-v11`, tel que défini par la classe `Hotel`.

```

// Load documents (using a subset of fields for brevity)
IndexDocumentsBatch<Hotel> batch = IndexDocumentsBatch.Create(
    IndexDocumentsAction.Upload(new Hotel { Id = "78", Name = "Upload Inn", Category = "hotel", Rate
= 279, Updated = new DateTime(2018, 3, 1, 7, 0, 0) }),
    IndexDocumentsAction.Upload(new Hotel { Id = "54", Name = "Breakpoint by the Sea", Category =
"motel", Rate = 162, Updated = new DateTime(2015, 9, 12, 7, 0, 0) }),
    IndexDocumentsAction.Upload(new Hotel { Id = "39", Name = "Debug Motel", Category = "motel",
Rate = 159, Updated = new DateTime(2016, 11, 11, 7, 0, 0) }),
    IndexDocumentsAction.Upload(new Hotel { Id = "48", Name = "NuGet Hotel", Category = "hotel",
Rate = 238, Updated = new DateTime(2016, 5, 30, 7, 0, 0) }),
    IndexDocumentsAction.Upload(new Hotel { Id = "12", Name = "Renovated Ranch", Category = "motel",
Rate = 149, Updated = new DateTime(2020, 1, 24, 7, 0, 0) }));

IndexDocumentsOptions idxoptions = new IndexDocumentsOptions { ThrowOnAnyError = true };

Console.WriteLine("{0}", "Loading index...\n");
qryclient.IndexDocuments(batch, idxoptions);

```

Une fois que vous avez initialisé l'objet [IndexDocumentsBatch](#), vous pouvez l'envoyer à l'index en appelant [IndexDocuments](#) sur votre objet [SearchClient](#).

- Étant donné qu'il s'agit d'une application console qui exécute toutes les commandes de manière séquentielle, ajoutez un délai d'attente de 2 secondes entre l'indexation et les requêtes.

```

// Wait 2 seconds for indexing to complete before starting queries (for demo and console-app
purposes only)
Console.WriteLine("Waiting for indexing...\n");
System.Threading.Thread.Sleep(2000);

```

Le retard de 2 secondes compense l'indexation, qui est asynchrone, afin que tous les documents puissent être indexés avant l'exécution des requêtes. Le codage dans un retard n'est nécessaire que dans les démonstrations, les tests et les exemples d'applications.

## 3 – Rechercher dans un index

Vous pouvez obtenir les résultats de la requête dès que le premier document est indexé, mais les tests réels de votre index doivent attendre que tous les documents soient indexés.

Cette section ajoute deux éléments de fonctionnalité : logique de requête et résultats. Pour les requêtes, utilisez la méthode [Search](#). Cette méthode prend le texte de recherche (la chaîne de requête) ainsi que d'autres [options](#).

La classe [SearchResults](#) représente les résultats.

- Dans [Program.cs](#), créez une méthode [WriteDocuments](#) qui affiche les résultats de la recherche sur la console.

```

private static void WriteDocuments(SearchResults<Hotel> searchResults)
{
    foreach ( SearchResult<Hotel> response in searchResults.GetResults() )
    {
        Hotel doc = response.Document;
        var score = response.Score;
        Console.WriteLine($"Name: {doc.Name}, Type: {doc.Category}, Rate: {doc.Rate}, Last-update:
{doc.Updated}, Score: {score}");
    }

    Console.WriteLine();
}

```

2. Créez une méthode RunQueries pour exécuter des requêtes et retourner des résultats. Les résultats sont des objets Hotel.

```
private static void RunQueries(SearchClient qryclient)
{
    SearchOptions options;
    SearchResults<Hotel> response;

    Console.WriteLine("Query #1: Search on the term 'motel' and list the relevance score for each
match...\n");

    options = new SearchOptions()
    {
        Filter = "",
        OrderBy = { "" }
    };

    response = qryclient.Search<Hotel>("motel", options);
    WriteDocuments(response);

    Console.WriteLine("Query #2: Find hotels where 'type' equals hotel...\n");

    options = new SearchOptions()
    {
        Filter = "hotelCategory eq 'hotel'",
    };

    response = qryclient.Search<Hotel>("*", options);
    WriteDocuments(response);

    Console.WriteLine("Query #3: Filter on rates less than $200 and sort by when the hotel was last
updated...\n");

    options = new SearchOptions()
    {
        Filter = "baseRate lt 200",
        OrderBy = { "lastRenovationDate desc" }
    };

    response = qryclient.Search<Hotel>("*", options);
    WriteDocuments(response);
}
```

Cet exemple montre les deux [façons de mettre en correspondance des termes dans une requête](#) : la recherche en texte intégral et les filtres.

- Une recherche en texte intégral effectue une requête sur un ou plusieurs termes dans des champs de votre index offrant une possibilité de recherche. La première requête est une recherche en texte intégral. La recherche en texte intégral produit des scores de pertinence utilisés pour classer les résultats.
- Un filtre est une expression booléenne évaluée sur des champs [IsFilterable](#) dans un index. Les requêtes de filtre incluent ou excluent des valeurs. Par conséquent, aucune note de pertinence n'est associée à une requête de filtre. Les deux dernières requêtes illustrent une recherche de filtre.

Vous pouvez utiliser la recherche en texte intégral et les filtres conjointement ou séparément.

Les recherches et les filtres sont effectués à l'aide de la méthode [SearchClient.Search](#). Une requête de recherche peut être transmise dans la chaîne `searchText`, tandis qu'une expression de filtre peut être transmise dans la propriété `Filter` de la classe `SearchOptions`. Pour filtrer sans effectuer de recherche, transmettez simplement `"*"` pour le paramètre `searchText` de la méthode `Search`. Pour effectuer une recherche sans filtrage, ne définissez pas la propriété `Filter` et ne transmettez aucune instance `SearchOptions`.

## Exécuter le programme

Appuyez sur F5 pour regénérer l'application et exécuter le programme dans son intégralité.

La sortie comprend des messages de la méthode `Console.WriteLine`, avec l'ajout d'informations et de résultats de requête.

## Nettoyer les ressources

Lorsque vous travaillez dans votre propre abonnement, il est recommandé, à la fin de chaque projet, de déterminer si vous avez toujours besoin des ressources que vous avez créées. Les ressources laissées en cours d'exécution peuvent vous coûter de l'argent. Vous pouvez supprimer les ressources une par une, ou choisir de supprimer le groupe de ressources afin de supprimer l'ensemble des ressources.

Vous pouvez rechercher et gérer les ressources dans le portail à l'aide des liens **Toutes les ressources** ou **Groupes de ressources** situés dans le volet de navigation de gauche.

Si vous utilisez un service gratuit, n'oubliez pas que vous êtes limité à trois index, indexeurs et sources de données. Vous pouvez supprimer des éléments un par un dans le portail pour ne pas dépasser la limite.

## Étapes suivantes

Dans ce guide de démarrage rapide C#, vous avez effectué une série de tâches pour créer un index, le charger avec des documents et exécuter des requêtes. À différents stades, nous avons pris des raccourcis afin de simplifier le code pour une meilleure lisibilité et compréhension. Si vous êtes familiarisé avec les concepts de base, nous vous recommandons l'article suivant pour explorer d'autres approches et concepts afin d'approfondir vos connaissances.

[Guide pratique pour développer dans .NET](#)

Vous souhaitez optimiser et réduire vos coûts de cloud ?

[Démarrer l'analyse des coûts avec Cost Management](#)

# Démarrage rapide : Créer un index Recherche cognitive Azure en Java à l'aide des API REST

04/10/2020 • 28 minutes to read • [Edit Online](#)

Générez une application console Java qui crée, charge et interroge un index de recherche à l'aide de [IntelliJ](#), du [SDK Java 11](#) et de l'[API REST Recherche cognitive Azure](#). Cet article fournit des instructions pas à pas pour créer l'application. Vous pouvez aussi [télécharger et exécuter l'application complète](#).

Si vous n'avez pas d'abonnement Azure, créez un [compte gratuit](#) avant de commencer.

## Prérequis

Nous avons utilisé les services et logiciels suivants pour générer et tester ce démarrage rapide :

- [IntelliJ IDEA](#)
- [Kit SDK Java 11](#)
- [Créez un service Recherche cognitive Azure](#) ou [recherchez un service existant](#) dans votre abonnement actuel. Vous pouvez utiliser un service gratuit pour ce guide de démarrage rapide.

## Obtenir une clé et une URL

Les appels au service nécessitent un point de terminaison d'URL et une clé d'accès pour chaque requête. Un service de recherche est créé avec les deux. Ainsi, si vous avez ajouté la Recherche cognitive Azure à votre abonnement, effectuez ce qui suit pour obtenir les informations nécessaires :

1. [Connectez-vous au portail Azure](#), puis dans la page **Vue d'ensemble** du service de recherche, récupérez l'URL. Voici un exemple de point de terminaison : `https://mydemo.search.windows.net`.
2. Dans **Paramètres > Clés**, obtenez une clé d'administration pour avoir des droits d'accès complets sur le service. Il existe deux clés d'administration interchangeables, fournies pour assurer la continuité de l'activité au cas où vous deviez en remplacer une. Vous pouvez utiliser la clé primaire ou secondaire sur les demandes d'ajout, de modification et de suppression d'objets.

Créez également une clé de requête. Il est recommandé d'émettre des demandes de requête avec un accès en lecture seule.

The screenshot shows the Microsoft Azure portal interface. The left sidebar has a tree view with 'mydemo' selected. The main content area is titled 'mydemo - Clés'. It shows two key fields: 'Clé d'administration primaire' containing '<placeholder-for-alphanumeric-autogenerated-string>' and 'Clé d'administration secondaire' which is empty. Below this is a table for managing request keys, with one row showing 'NAME' and 'CLÉ' both as '<placeholder-for-alphanumeric-autogenerated-string>'.

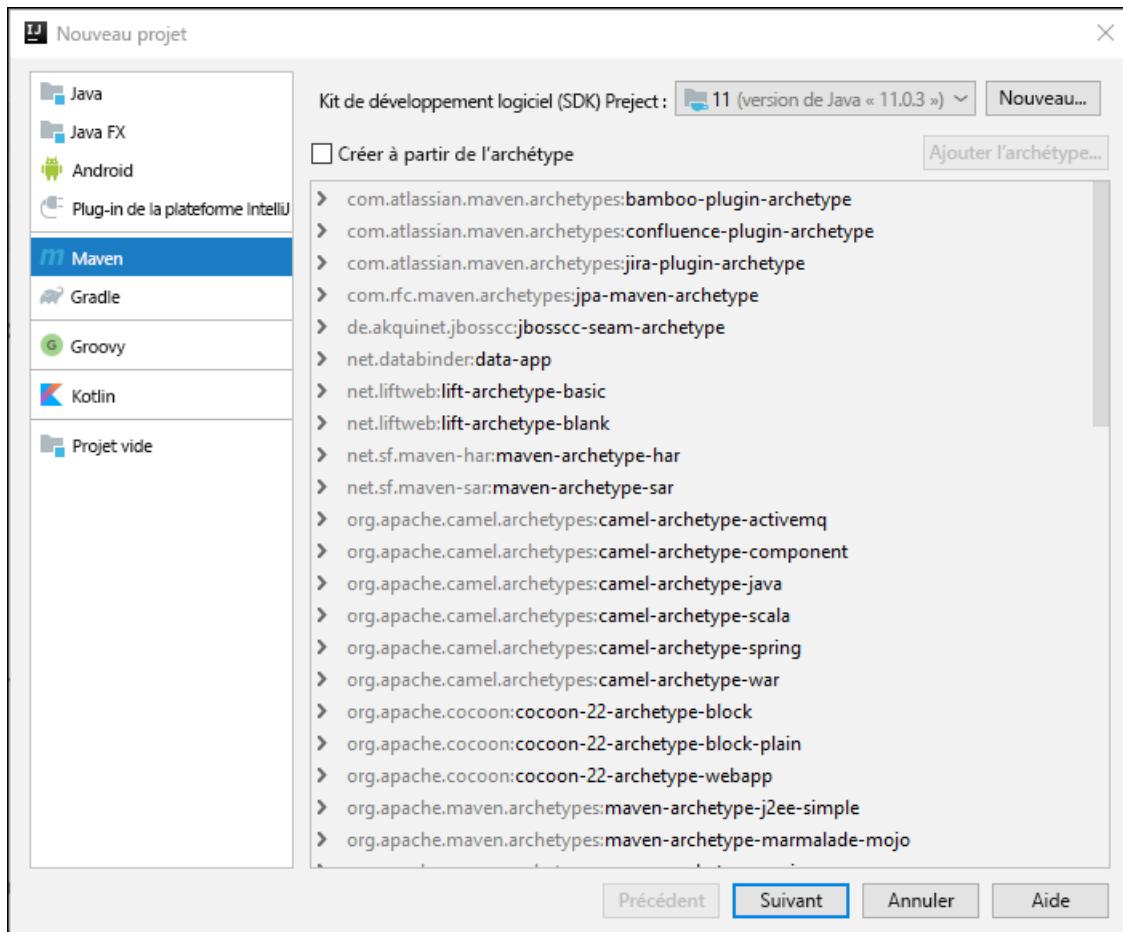
Chaque demande envoyée à votre service nécessite une clé API. L'utilisation d'une clé valide permet d'établir, en fonction de chaque demande, une relation de confiance entre l'application qui envoie la demande et le service qui en assure le traitement.

## Configurer votre environnement

Commencez par ouvrir IntelliJ IDEA et configurer un nouveau projet.

### Créer le projet

1. Ouvrez IntelliJ IDEA et sélectionnez **Create New Project** (Créer un projet).
2. Sélectionnez **Maven**.
3. Dans la liste **Project SDK**, sélectionnez le kit SDK Java 11.

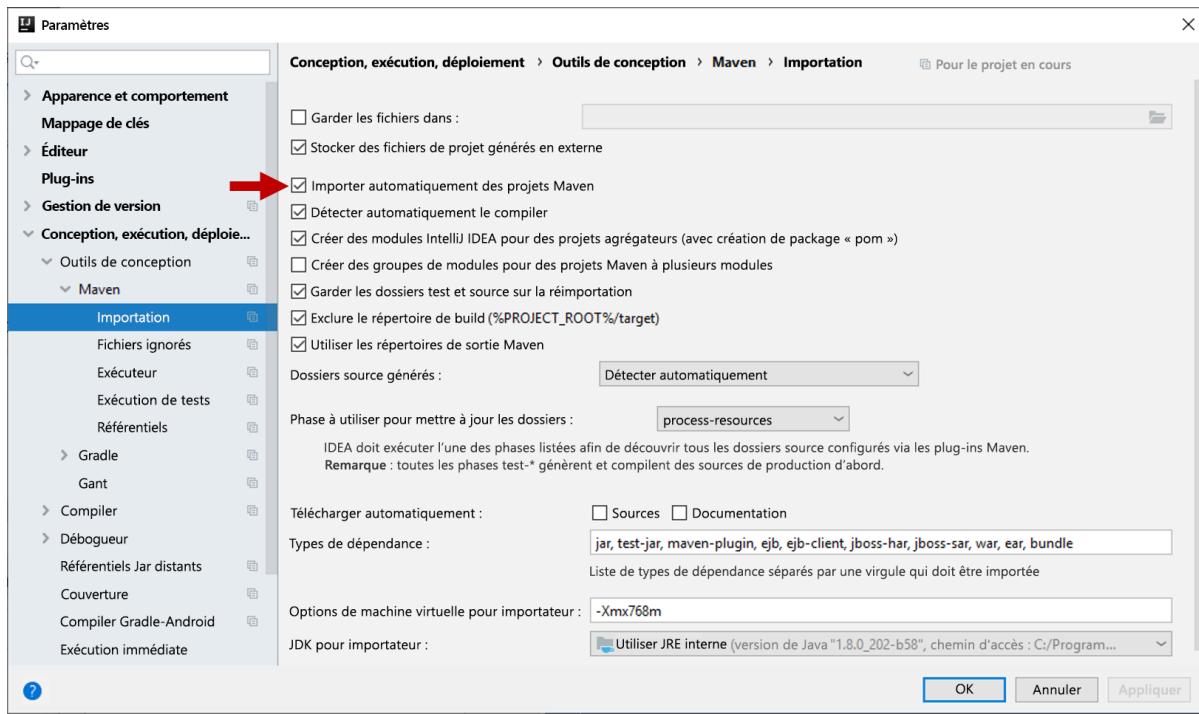


4. Pour **GroupId** et **ArtifactId**, entrez `AzureSearchQuickstart`.

5. Acceptez les valeurs par défaut restantes pour ouvrir le projet.

### Spécifier les dépendances Maven

1. Sélectionnez Fichier > Paramètres.
2. Dans la fenêtre Paramètres, sélectionnez Build, Execution, Deployment > Build Tools > Maven > Importing (Build, Exécution, Déploiement > Outils de build > Maven > Importation).
3. Cochez la case Import Maven projects automatically (Importer les projets Maven automatiquement), puis cliquez sur OK pour fermer la fenêtre. Les plug-ins Maven et les autres dépendances sont désormais automatiquement synchronisés quand vous mettez à jour le fichier pom.xml à l'étape suivante.



- Ouvrez le fichier pom.xml et remplacez le contenu par les détails de configuration Maven suivants. Ceux-ci incluent des références au [plug-in Exec Maven](#) et une [API d'interface JSON](#)

```

<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-
4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

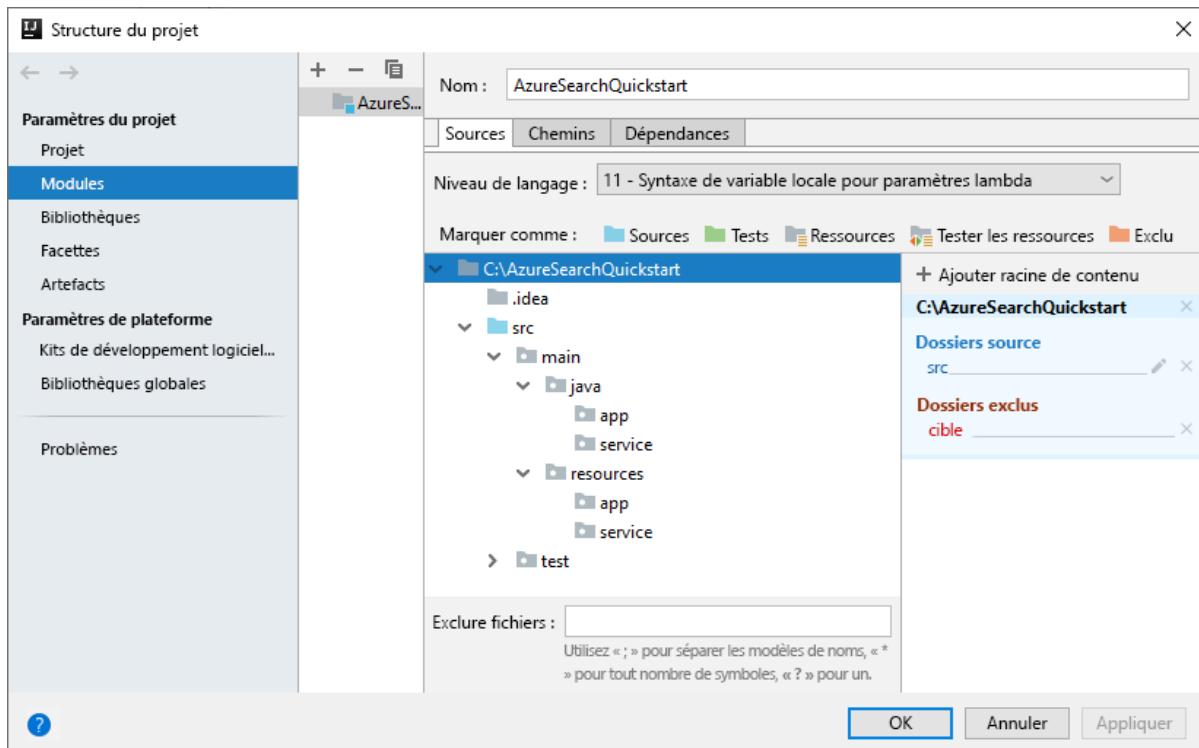
    <groupId>AzureSearchQuickstart</groupId>
    <artifactId>AzureSearchQuickstart</artifactId>
    <version>1.0-SNAPSHOT</version>
    <build>
        <sourceDirectory>src</sourceDirectory>
        <plugins>
            <plugin>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.1</version>
                <configuration>
                    <source>11</source>
                    <target>11</target>
                </configuration>
            </plugin>
            <plugin>
                <groupId>org.codehaus.mojo</groupId>
                <artifactId>exec-maven-plugin</artifactId>
                <version>1.6.0</version>
                <executions>
                    <execution>
                        <goals>
                            <goal>exec</goal>
                        </goals>
                    </execution>
                </executions>
                <configuration>
                    <mainClass>main.java.app.App</mainClass>
                    <cleanupDaemonThreads>false</cleanupDaemonThreads>
                </configuration>
            </plugin>
        </plugins>
    </build>
    <dependencies>
        <dependency>
            <groupId>org.glassfish</groupId>
            <artifactId>javax.json</artifactId>
            <version>1.0.2</version>
        </dependency>
    </dependencies>
</project>

```

## Configurer la structure de projet

1. Sélectionnez Fichier > Structure du projet.
2. Sélectionnez **Modules**, puis développez l'arborescence source pour accéder au contenu du dossier `src` > `main`.
3. Dans le dossier `src` > `main` > `java`, ajoutez les dossiers `app` et `service`. Pour ce faire, sélectionnez le dossier `java`, appuyez sur Alt+Insertion, puis entrez le nom du dossier.
4. Dans le dossier `src` > `main` > `resources`, ajoutez les dossiers `app` et `service`.

Quand vous avez terminé, l'arborescence de projet doit ressembler à l'image suivante.



5. Cliquez sur **OK** pour fermer la fenêtre.

#### Ajouter des informations relatives au service Recherche cognitive Azure

1. Dans la fenêtre **Projet**, développez l'arborescence source pour accéder au dossier `src > main > resources` > `app`, puis ajoutez un fichier `config.properties`. Pour ce faire, sélectionnez le dossier `app`, appuyez sur Alt+Insertion, sélectionnez **Fichier**, puis entrez le nom du fichier.
2. Copiez les paramètres suivants dans le nouveau fichier et remplacez `<YOUR-SEARCH-SERVICE-NAME>`, `<YOUR-ADMIN-KEY>` et `<YOUR-QUERY-KEY>` par les clés et le nom de votre service. Si le point de terminaison de votre service est `https://mydemo.search.windows.net`, le nom du service serait « mydemo ».

```
SearchServiceName=<YOUR-SEARCH-SERVICE-NAME>
SearchServiceAdminKey=<YOUR-ADMIN-KEY>
SearchServiceQueryKey=<YOUR-QUERY-KEY>
IndexName=hotels-quickstart
ApiVersion=2020-06-30
```

#### Ajouter la méthode principale

1. Dans le dossier `src > main > java > app`, ajoutez une classe `App`. Pour ce faire, sélectionnez le dossier `app`, appuyez sur Alt+Insertion, sélectionnez **Java Class** (Classe Java), puis entrez le nom de la classe.
2. Ouvrez la classe `App` et remplacez le contenu par le code suivant. Ce code contient la méthode `main`.

Le code sans marques de commentaire lit les paramètres du service de recherche et les utilise pour créer une instance du client du service de recherche. Le code client du service de recherche sera ajouté dans la section suivante.

Le code commenté dans cette classe verra ses marques de commentaire supprimées dans une section ultérieure de ce guide de démarrage rapide.

```
package main.java.app;

import main.java.service.SearchServiceClient;
import java.io.IOException;
import java.util.Properties;
```

```

public class App {

    private static Properties loadPropertiesFromResource(String resourcePath) throws IOException {
        var inputStream = App.class.getResourceAsStream(resourcePath);
        var configProperties = new Properties();
        configProperties.load(inputStream);
        return configProperties;
    }

    public static void main(String[] args) {
        try {
            var config = loadPropertiesFromResource("/app/config.properties");
            var client = new SearchServiceClient(
                config.getProperty("SearchServiceName"),
                config.getProperty("SearchServiceAdminKey"),
                config.getProperty("SearchServiceQueryKey"),
                config.getProperty("ApiVersion"),
                config.getProperty("IndexName")
            );
        }

        //Uncomment the next 3 lines in the 1 - Create Index section of the quickstart
        //        if(client.indexExists()){ client.deleteIndex();}
        //        client.createIndex("/service/index.json");
        //        Thread.sleep(1000L); // wait a second to create the index

        //Uncomment the next 2 lines in the 2 - Load Documents section of the quickstart
        //        client.uploadDocuments("/service/hotels.json");
        //        Thread.sleep(2000L); // wait 2 seconds for data to upload

        //Uncomment the following 5 search queries in the 3 - Search an index section of the quickstart
        //        // Query 1
        //        client.logMessage("\n*QUERY
1*****";
        //        client.logMessage("Search for: Atlanta'");
        //        client.logMessage("Return: All fields'");
        //        client.searchPlus("Atlanta");
        //
        //        // Query 2
        //        client.logMessage("\n*QUERY
2*****";
        //        client.logMessage("Search for: Atlanta");
        //        client.logMessage("Return: HotelName, Tags, Address");
        //        SearchServiceClient.SearchOptions options2 = client.createSearchOptions();
        //        options2.select = "HotelName,Tags,Address";
        //        client.searchPlus("Atlanta", options2);
        //
        //        //Query 3
        //        client.logMessage("\n*QUERY
3*****";
        //        client.logMessage("Search for: wifi & restaurant");
        //        client.logMessage("Return: HotelName, Description, Tags");
        //        SearchServiceClient.SearchOptions options3 = client.createSearchOptions();
        //        options3.select = "HotelName,Description,Tags";
        //        client.searchPlus("wifi,restaurant", options3);
        //
        //        // Query 4 -filtered query
        //        client.logMessage("\n*QUERY
4*****";
        //        client.logMessage("Search for: all");
        //        client.logMessage("Filter: Ratings greater than 4");
        //        client.logMessage("Return: HotelName, Rating");
        //        SearchServiceClient.SearchOptions options4 = client.createSearchOptions();
        //        options4.filter="Rating%20gt%204";
        //        options4.select = "HotelName,Rating";
        //        client.searchPlus("",options4);
        //
        //        // Query 5 - top 2 results, ordered by
        //        client.logMessage("\n*QUERY

```

```
5*****");
//        client.logMessage("Search for: boutique");
//        client.logMessage("Get: Top 2 results");
//        client.logMessage("Order by: Rating in descending order");
//        client.logMessage("Return: HotelId, HotelName, Category, Rating");
//        SearchServiceClient.SearchOptions options5 = client.createSearchOptions();
//        options5.top=2;
//        options5.orderby = "Rating%20desc";
//        options5.select = "HotelId,HotelName,Category,Rating";
//        client.searchPlus("boutique", options5);

    } catch (Exception e) {
        System.err.println("Exception:" + e.getMessage());
        e.printStackTrace();
    }
}
}
```

## Ajouter les opérations HTTP

1. Dans le dossier `src` > `main` > `java` > `service`, ajoutez une classe `SearchServiceClient`. Pour ce faire, sélectionnez le dossier `service`, appuyez sur Alt+Insertion, sélectionnez **Java Class** (Classe Java), puis entrez le nom de la classe.
  2. Ouvrez la classe `SearchServiceClient` et remplacez son contenu par le code suivant. Ce code fournit les opérations HTTP nécessaires pour utiliser l'API REST de la Recherche cognitive Azure. Des méthodes supplémentaires pour la création d'un index, le chargement de documents et l'interrogation de l'index seront ajoutées dans une section ultérieure.

```
package main.java.service;

import javax.json.Json;
import javax.net.ssl.HttpsURLConnection;
import java.io.IOException;
import java.io.StringReader;
import java.net.HttpURLConnection;
import java.net.URI;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.nio.charset.StandardCharsets;
import java.util.Formatter;
import java.util.function.Consumer;

/* This class is responsible for implementing HTTP operations for creating the index, uploading
documents and searching the data*/
public class SearchServiceClient {

    private final String _adminKey;
    private final String _queryKey;
    private final String _apiVersion;
    private final String _serviceName;
    private final String _indexName;
    private final static HttpClient client = HttpClient.newHttpClient();

    public SearchServiceClient(String serviceName, String adminKey, String queryKey, String apiVersion,
String indexName) {
        this._serviceName = serviceName;
        this._adminKey = adminKey;
        this._queryKey = queryKey;
        this._apiVersion = apiVersion;
        this._indexName = indexName;
    }

    private static HttpResponse<String> sendRequest(HttpRequest request) throws IOException,
InterruptedException {

```

```

        logMessage(String.format("%s: %s", request.method(), request.uri()));
        return client.send(request, HttpResponse.BodyHandlers.ofString()));
    }

    private static URI buildURI(Consumer<Formatter> fmtFn)
    {
        Formatter strFormatter = new Formatter();
        fmtFn.accept(strFormatter);
        String url = strFormatter.out().toString();
        strFormatter.close();
        return URI.create(url);
    }

    public static void logMessage(String message) {
        System.out.println(message);
    }

    public static boolean isSuccessResponse(HttpResponse<String> response) {
        try {
            int responseCode = response.statusCode();

            logMessage("\n Response code = " + responseCode);

            if (responseCode == HttpURLConnection.HTTP_OK || responseCode ==
HttpURLConnection.HTTP_ACCEPTED
                || responseCode == HttpURLConnection.HTTP_NO_CONTENT || responseCode ==
HttpsURLConnection.HTTP_CREATED) {
                return true;
            }

            // We got an error
            var msg = response.body();
            if (msg != null) {
                logMessage(String.format("\n MESSAGE: %s", msg));
            }
        } catch (Exception e) {
            e.printStackTrace();
        }

        return false;
    }

    public static HttpRequest httpRequest(URI uri, String key, String method, String contents) {
        contents = contents == null ? "" : contents;
        var builder = HttpRequest.newBuilder();
        builder.uri(uri);
        builder.setHeader("content-type", "application/json");
        builder.setHeader("api-key", key);

        switch (method) {
            case "GET":
                builder = builder.GET();
                break;
            case "HEAD":
                builder = builder.GET();
                break;
            case "DELETE":
                builder = builder.DELETE();
                break;
            case "PUT":
                builder = builder.PUT(HttpRequest.BodyPublishers.ofString(contents));
                break;
            case "POST":
                builder = builder.POST(HttpRequest.BodyPublishers.ofString(contents));
                break;
            default:
                throw new IllegalArgumentException(String.format("Can't create request for method '%s'", method)));
        }
    }
}

```

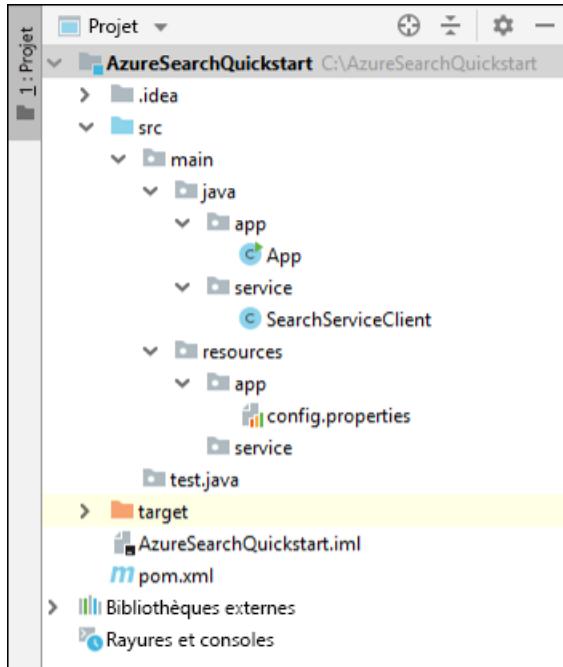
```

        }
        return builder.build();
    }
}

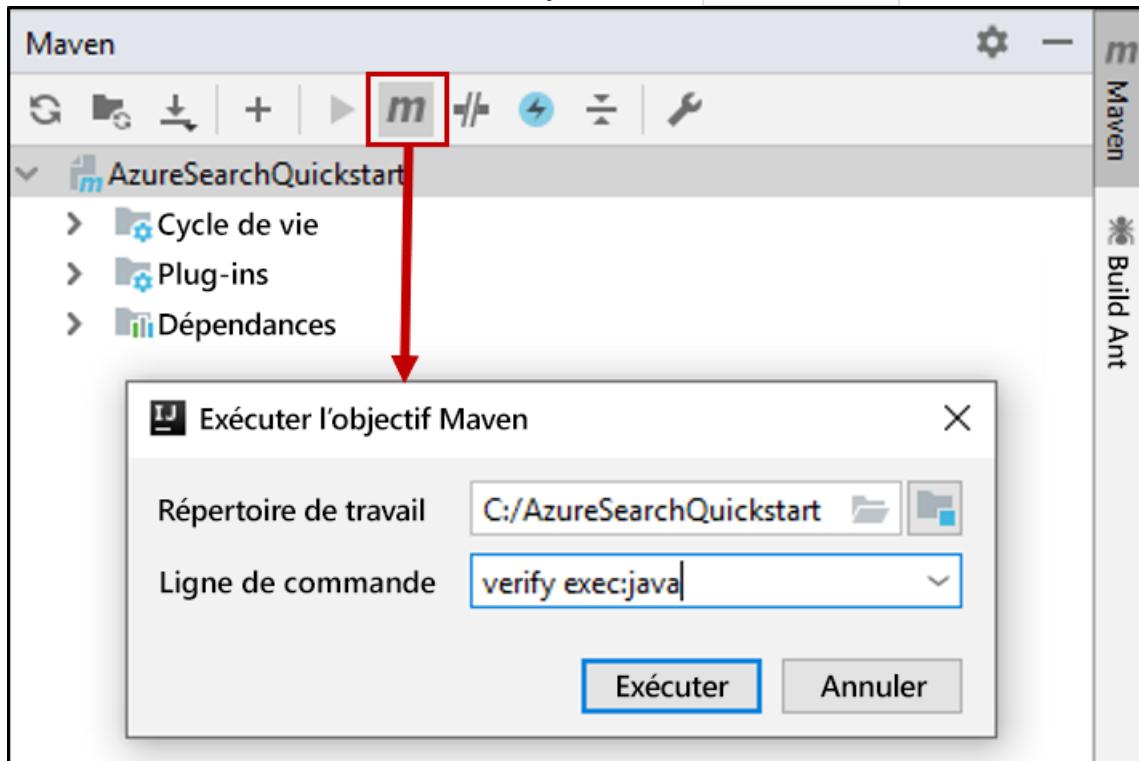
```

## Créer le projet

- Vérifiez que votre projet a la structure suivante.



- Ouvrez la fenêtre Outil Maven et exécutez cet objectif Maven : `verify exec:java`



Une fois le traitement terminé, recherchez un message BUILD SUCCESS suivi d'un code de sortie zéro (0).

## 1 - Créer un index

La définition de l'index des hôtels contient des champs simples et un champ complexe. « HotelName » ou « Description » sont des exemples de champs simples. Le champ « Address » est un champ complexe, car il contient

des sous-champs, tels que « Street Address » et « City ». Dans ce guide de démarrage rapide, la définition d'index est spécifiée à l'aide de JSON.

1. Dans la fenêtre **Projet**, développez l'arborescence source pour accéder au dossier `src > main > resources` > `service`, puis ajoutez un fichier `index.json`. Pour ce faire, sélectionnez le dossier `app`, appuyez sur Alt+Insertion, sélectionnez **Fichier**, puis entrez le nom du fichier.
2. Ouvrez le fichier `index.json` et insérez la définition d'index suivante.

```
{  
  "name": "hotels-quickstart",  
  "fields": [  
    {  
      "name": "HotelId",  
      "type": "Edm.String",  
      "key": true,  
      "filterable": true  
    },  
    {  
      "name": "HotelName",  
      "type": "Edm.String",  
      "searchable": true,  
      "filterable": false,  
      "sortable": true,  
      "facetable": false  
    },  
    {  
      "name": "Description",  
      "type": "Edm.String",  
      "searchable": true,  
      "filterable": false,  
      "sortable": false,  
      "facetable": false,  
      "analyzer": "en.lucene"  
    },  
    {  
      "name": "Description_fr",  
      "type": "Edm.String",  
      "searchable": true,  
      "filterable": false,  
      "sortable": false,  
      "facetable": false,  
      "analyzer": "fr.lucene"  
    },  
    {  
      "name": "Category",  
      "type": "Edm.String",  
      "searchable": true,  
      "filterable": true,  
      "sortable": true,  
      "facetable": true  
    },  
    {  
      "name": "Tags",  
      "type": "Collection(Edm.String)",  
      "searchable": true,  
      "filterable": true,  
      "sortable": false,  
      "facetable": true  
    },  
    {  
      "name": "ParkingIncluded",  
      "type": "Edm.Boolean",  
      "filterable": true,  
      "sortable": true,  
      "facetable": true  
    }  
  ]  
}
```

```

        },
        {
            "name": "LastRenovationDate",
            "type": "Edm.DateTimeOffset",
            "filterable": true,
            "sortable": true,
            "facetable": true
        },
        {
            "name": "Rating",
            "type": "Edm.Double",
            "filterable": true,
            "sortable": true,
            "facetable": true
        },
        {
            "name": "Address",
            "type": "Edm.ComplexType",
            "fields": [
                {
                    "name": "StreetAddress",
                    "type": "Edm.String",
                    "filterable": false,
                    "sortable": false,
                    "facetable": false,
                    "searchable": true
                },
                {
                    "name": "City",
                    "type": "Edm.String",
                    "searchable": true,
                    "filterable": true,
                    "sortable": true,
                    "facetable": true
                },
                {
                    "name": "StateProvince",
                    "type": "Edm.String",
                    "searchable": true,
                    "filterable": true,
                    "sortable": true,
                    "facetable": true
                },
                {
                    "name": "PostalCode",
                    "type": "Edm.String",
                    "searchable": true,
                    "filterable": true,
                    "sortable": true,
                    "facetable": true
                },
                {
                    "name": "Country",
                    "type": "Edm.String",
                    "searchable": true,
                    "filterable": true,
                    "sortable": true,
                    "facetable": true
                }
            ]
        }
    ]
}

```

Le nom de l'index sera « hotels-quickstart ». Les attributs des champs d'index déterminent la manière dont les données indexées peuvent faire l'objet d'une recherche dans une application. Par exemple, l'attribut `IsSearchable` doit être affecté à tous les champs qui doivent être inclus dans une recherche en texte intégral.

Pour en savoir plus sur les attributs, consultez [Collection et attributs de champs](#).

Le champ `Description` de cet index utilise la propriété `analyzer` facultative pour remplacer l'analyseur de langage Lucene par défaut. Le champ `Description_fr` utilise l'analyseur Lucene en français (`fr.lucene`), car il stocke le texte en français. Le champ `Description` utilise l'analyseur de langage Microsoft facultatif `en.lucene`. Pour en savoir plus sur les analyseurs, consultez [Analyseurs pour le traitement de texte dans la Recherche cognitive Azure](#).

3. Ajoutez le code suivant à la classe `SearchServiceClient`. Ces méthodes génèrent des URL de service REST pour la Recherche cognitive Azure, qui créent et suppriment un index, et qui déterminent son existence. Les méthodes constituent également la requête HTTP.

```
public boolean indexExists() throws IOException, InterruptedException {
    logMessage("\n Checking if index exists...");
    var uri = buildURI(strFormatter -> strFormatter.format(
        "https://$s.search.windows.net/indexes/$s/docs?api-version=%s&search=*",
        _serviceName,_indexName,_apiVersion));
    var request = httpRequest(uri, _adminKey, "HEAD", "");
    var response = sendRequest(request);
    return isSuccessResponse(response);
}

public boolean deleteIndex() throws IOException, InterruptedException {
    logMessage("\n Deleting index...");
    var uri = buildURI(strFormatter -> strFormatter.format(
        "https://$s.search.windows.net/indexes/$s?api-version=%s",
        _serviceName,_indexName,_apiVersion));
    var request = httpRequest(uri, _adminKey, "DELETE", "*");
    var response = sendRequest(request);
    return isSuccessResponse(response);
}

public boolean createIndex(String indexDefinitionFile) throws IOException, InterruptedException {
    logMessage("\n Creating index...");
    //Build the search service URL
    var uri = buildURI(strFormatter -> strFormatter.format(
        "https://$s.search.windows.net/indexes/$s?api-version=%s",
        _serviceName,_indexName,_apiVersion));
    //Read in index definition file
    var inputStream = SearchServiceClient.class.getResourceAsStream(indexDefinitionFile);
    var indexDef = new String(inputStream.readAllBytes(), StandardCharsets.UTF_8);
    //Send HTTP PUT request to create the index in the search service
    var request = httpRequest(uri, _adminKey, "PUT", indexDef);
    var response = sendRequest(request);
    return isSuccessResponse(response);
}
```

4. Supprimez les marques de commentaire du code suivant dans la classe `App`. Ce code supprime l'index « hotels-quickstart », s'il existe, et crée un index basé sur la définition de l'index dans le fichier « index.json ».

Une pause d'une seconde est insérée après la demande de création d'index. Cette pause permet de s'assurer que l'index est créé avant de charger des documents.

```
if (client.indexExists()) { client.deleteIndex();}
client.createIndex("/service/index.json");
Thread.sleep(1000L); // wait a second to create the index
```

5. Ouvrez la fenêtre Outil Maven et exécutez cet objectif Maven : `verify exec:java`

Lors de l'exécution du code, recherchez un message de type « Crédit d'index » suivi d'un code de

réponse 201. Ce code de réponse confirme que l'index a été créé. L'exécution doit se terminer par un message BUILD SUCCESS et un code de sortie zéro (0).

## 2 – Charger des documents

1. Dans la fenêtre **Projet**, développez l'arborescence source pour accéder au dossier `src > main > resources > service`, puis ajoutez un fichier `hotels.json`. Pour ce faire, sélectionnez le dossier `app`, appuyez sur Alt+Insertion, sélectionnez **Fichier**, puis entrez le nom du fichier.
2. Insérez les documents d'hôtel suivants dans le fichier.

```
{
  "value": [
    {
      "@search.action": "upload",
      "HotelId": "1",
      "HotelName": "Secret Point Motel",
      "Description": "The hotel is ideally located on the main commercial artery of the city in the heart of New York. A few minutes away is Time's Square and the historic centre of the city, as well as other places of interest that make New York one of America's most attractive and cosmopolitan cities.",
      "Description_fr": "L'hôtel est idéalement situé sur la principale artère commerciale de la ville en plein cœur de New York. A quelques minutes se trouve la place du temps et le centre historique de la ville, ainsi que d'autres lieux d'intérêt qui font de New York l'une des villes les plus attractives et cosmopolites de l'Amérique.",
      "Category": "Boutique",
      "Tags": [ "pool", "air conditioning", "concierge" ],
      "ParkingIncluded": "false",
      "LastRenovationDate": "1970-01-18T00:00:00Z",
      "Rating": 3.60,
      "Address": {
        "StreetAddress": "677 5th Ave",
        "City": "New York",
        "StateProvince": "NY",
        "PostalCode": "10022",
        "Country": "USA"
      }
    },
    {
      "@search.action": "upload",
      "HotelId": "2",
      "HotelName": "Twin Dome Motel",
      "Description": "The hotel is situated in a nineteenth century plaza, which has been expanded and renovated to the highest architectural standards to create a modern, functional and first-class hotel in which art and unique historical elements coexist with the most modern comforts.",
      "Description_fr": "L'hôtel est situé dans une place du XIXe siècle, qui a été agrandie et rénovée aux plus hautes normes architecturales pour créer un hôtel moderne, fonctionnel et de première classe dans lequel l'art et les éléments historiques uniques coexistent avec le confort le plus moderne.",
      "Category": "Boutique",
      "Tags": [ "pool", "free wifi", "concierge" ],
      "ParkingIncluded": "false",
      "LastRenovationDate": "1979-02-18T00:00:00Z",
      "Rating": 3.60,
      "Address": {
        "StreetAddress": "140 University Town Center Dr",
        "City": "Sarasota",
        "StateProvince": "FL",
        "PostalCode": "34243",
        "Country": "USA"
      }
    },
    {
      "@search.action": "upload",
      "HotelId": "3",
      "HotelName": "Triple Landscape Hotel",
      "Description": "The Hotel stands out for its gastronomic excellence under the management of William Dough, who advises on and oversees all of the Hotel's restaurant services."
    }
  ]
}
```

```

    "Description_fr": "L'hôtel est situé dans une place du XIXe siècle, qui a été agrandie et rénovée aux plus hautes normes architecturales pour créer un hôtel moderne, fonctionnel et de première classe dans lequel l'art et les éléments historiques uniques coexistent avec le confort le plus moderne.",  

    "Category": "Resort and Spa",  

    "Tags": [ "air conditioning", "bar", "continental breakfast" ],  

    "ParkingIncluded": "true",  

    "LastRenovationDate": "2015-09-20T00:00:00Z",  

    "Rating": 4.80,  

    "Address": {  

        "StreetAddress": "3393 Peachtree Rd",  

        "City": "Atlanta",  

        "StateProvince": "GA",  

        "PostalCode": "30326",  

        "Country": "USA"  

    }  

},  

{  

    "@search.action": "upload",  

    "HotelId": "4",  

    "HotelName": "Sublime Cliff Hotel",  

    "Description": "Sublime Cliff Hotel is located in the heart of the historic center of Sublime in an extremely vibrant and lively area within short walking distance to the sites and landmarks of the city and is surrounded by the extraordinary beauty of churches, buildings, shops and monuments. Sublime Cliff is part of a lovingly restored 1800 palace.",  

    "Description_fr": "Le sublime Cliff Hotel est situé au coeur du centre historique de sublime dans un quartier extrêmement animé et vivant, à courte distance de marche des sites et monuments de la ville et est entouré par l'extraordinaire beauté des églises, des bâtiments, des commerces et Monuments. Sublime Cliff fait partie d'un Palace 1800 restauré avec amour.",  

    "Category": "Boutique",  

    "Tags": [ "concierge", "view", "24-hour front desk service" ],  

    "ParkingIncluded": "true",  

    "LastRenovationDate": "1960-02-06T00:00:00Z",  

    "Rating": 4.60,  

    "Address": {  

        "StreetAddress": "7400 San Pedro Ave",  

        "City": "San Antonio",  

        "StateProvince": "TX",  

        "PostalCode": "78216",  

        "Country": "USA"  

    }  

}  

]  

}

```

3. Insérez le code ci-après dans la classe `SearchServiceClient`. Ce code génère l'URL de service REST pour charger les documents d'hôtel dans l'index, puis crée la requête HTTP POST.

```

public boolean uploadDocuments(String documentsFile) throws IOException, InterruptedException {  

    logMessage("\n Uploading documents...");  

    //Build the search service URL  

    var endpoint = buildURI(strFormatter -> strFormatter.format(  

        "https://%.search.windows.net/indexes/%s/docs/index?api-version=%s",  

        _serviceName,_indexName,_apiVersion));  

    //Read in the data to index  

    var inputStream = SearchServiceClient.class.getResourceAsStream(documentsFile);  

    var documents = new String(inputStream.readAllBytes(), StandardCharsets.UTF_8);  

    //Send HTTP POST request to upload and index the data  

    var request = httpRequest(endpoint, _adminKey, "POST", documents);  

    var response = sendRequest(request);  

    return isSuccessResponse(response);  

}

```

4. Supprimez les marques de commentaire du code suivant dans la classe `App`. Ce code charge les documents dans « hotels.json » dans l'index.

```
client.uploadDocuments("/service/hotels.json");
Thread.sleep(2000L); // wait 2 seconds for data to upload
```

Une pause de deux secondes est insérée après la demande de chargement pour vérifier que le processus de chargement de documents se termine avant d'interroger l'index.

5. Ouvrez la fenêtre **Outil Maven** et exécutez cet objectif Maven : `verify exec:java`

Comme vous avez créé un index « hotels-quickstart » à l'étape précédente, le code le supprime et le recrée avant de charger les documents d'hôtel.

Lors de l'exécution du code, recherchez un message de type « Chargement de documents » suivi d'un code de réponse 200. Ce code de réponse confirme que les documents ont été chargés dans l'index. L'exécution doit se terminer par un message BUILD SUCCESS et un code de sortie zéro (0).

## 3 – Rechercher dans un index

Maintenant que vous avez chargé les documents d'hôtels, vous pouvez créer des requêtes de recherche pour accéder aux données d'hôtels.

1. Ajoutez le code suivant à la classe `SearchServiceClient`. Ce code génère des URL de service REST pour la Recherche cognitive Azure afin de rechercher les données indexées et d'imprimer les résultats de la recherche.

La classe `SearchOptions` et la méthode `createSearchOptions` vous permettent de spécifier un sous-ensemble des options de requête disponibles de l'API REST de la Recherche cognitive Azure. Pour plus d'informations sur les options de requête de l'API REST, consultez [Rechercher des documents \(API REST de la Recherche cognitive Azure\)](#).

La méthode `searchPlus` crée l'URL de requête de recherche, effectue la demande de recherche, puis imprime les résultats dans la console.

```

public SearchOptions createSearchOptions() { return new SearchOptions();}

//Defines available search parameters that can be set
public static class SearchOptions {

    public String select = "";
    public String filter = "";
    public int top = 0;
    public String orderby= "";
}

//Concatenates search parameters to append to the search request
private String createOptionsString(SearchOptions options)
{
    String optionsString = "";
    if (options != null) {
        if (options.select != "")
            optionsString = optionsString + "&$select=" + options.select;
        if (options.filter != "")
            optionsString = optionsString + "&$filter=" + options.filter;
        if (options.top != 0)
            optionsString = optionsString + "&$top=" + options.top;
        if (options.orderby != "")
            optionsString = optionsString + "&$orderby=" + options.orderby;
    }
    return optionsString;
}

public void searchPlus(String queryString)
{
    searchPlus( queryString, null);
}

public void searchPlus(String queryString, SearchOptions options) {

    try {
        String optionsString = createOptionsString(options);
        var uri = buildURI(strFormatter -> strFormatter.format(
            "https://%s.search.windows.net/indexes/%s/docs?api-version=%s&search=%s",
            _serviceName, _indexName, _apiVersion, queryString, optionsString));
        var request = httpRequest(uri, _queryKey, "GET", null);
        var response = sendRequest(request);
        var jsonReader = Json.createReader(new StringReader(response.body()));
        var jsonArray = jsonReader.readObject().getJSONArray("value");
        var resultsCount = jsonArray.size();
        logMessage("Results:\nCount: " + resultsCount);
        for (int i = 0; i <= resultsCount - 1; i++) {
            logMessage(jsonArray.get(i).toString());
        }

        jsonReader.close();
    }
    catch (Exception e) {
        e.printStackTrace();
    }
}
}

```

2. Dans la classe `App`, supprimez les marques de commentaire du code suivant. Ce code configure cinq requêtes différentes, dont le texte de recherche, les paramètres de requête et les champs de données à retourner.

```

// Query 1
client.logMessage("\n*QUERY 1*****");
client.logMessage("Search for: Atlanta");
client.logMessage("Return: All fields'");
client.searchPlus("Atlanta");

// Query 2
client.logMessage("\n*QUERY 2*****");
client.logMessage("Search for: Atlanta");
client.logMessage("Return: HotelName, Tags, Address");
SearchServiceClient.SearchOptions options2 = client.createSearchOptions();
options2.select = "HotelName,Tags,Address";
client.searchPlus("Atlanta", options2);

//Query 3
client.logMessage("\n*QUERY 3*****");
client.logMessage("Search for: wifi & restaurant");
client.logMessage("Return: HotelName, Description, Tags");
SearchServiceClient.SearchOptions options3 = client.createSearchOptions();
options3.select = "HotelName,Description,Tags";
client.searchPlus("wifi,restaurant", options3);

// Query 4 -filtered query
client.logMessage("\n*QUERY 4*****");
client.logMessage("Search for: all");
client.logMessage("Filter: Ratings greater than 4");
client.logMessage("Return: HotelName, Rating");
SearchServiceClient.SearchOptions options4 = client.createSearchOptions();
options4.filter="Rating%20gt%204";
options4.select = "HotelName,Rating";
client.searchPlus("*",options4);

// Query 5 - top 2 results, ordered by
client.logMessage("\n*QUERY 5*****");
client.logMessage("Search for: boutique");
client.logMessage("Get: Top 2 results");
client.logMessage("Order by: Rating in descending order");
client.logMessage("Return: HotelId, HotelName, Category, Rating");
SearchServiceClient.SearchOptions options5 = client.createSearchOptions();
options5.top=2;
options5.orderby = "Rating%20desc";
options5.select = "HotelId,HotelName,Category,Rating";
client.searchPlus("boutique", options5);

```

Il existe deux [façons de mettre en correspondance des termes dans une requête](#) : la recherche en texte intégral et les filtres. Une requête de recherche en texte intégral recherche un ou plusieurs termes dans les champs `IsSearchable` de votre index. Un filtre est une expression booléenne évaluée sur les champs `IsFilterable` dans un index. Vous pouvez utiliser la recherche en texte intégral et les filtres conjointement ou séparément.

### 3. Ouvrez la fenêtre Outil Maven et exécutez cet objectif Maven : `verify exec:java`

Recherchez un résumé de chaque requête et ses résultats. L'exécution doit se terminer par un message BUILD SUCCESS et un code de sortie zéro (0).

## Nettoyer les ressources

Lorsque vous travaillez dans votre propre abonnement, il est judicieux à la fin d'un projet de supprimer les ressources dont vous n'avez plus besoin. Les ressources laissées en cours d'exécution peuvent vous coûter de l'argent. Vous pouvez supprimer les ressources une par une, ou choisir de supprimer le groupe de ressources afin de supprimer l'ensemble des ressources.

Vous pouvez rechercher et gérer les ressources dans le portail à l'aide des liens **Toutes les ressources** ou

**Groupes de ressources** situés dans le volet de navigation de gauche.

Si vous utilisez un service gratuit, n'oubliez pas que vous êtes limité à trois index, indexeurs et sources de données. Vous pouvez supprimer des éléments un par un dans le portail pour ne pas dépasser la limite.

## Étapes suivantes

Dans ce guide de démarrage rapide Java, vous avez effectué une série de tâches pour créer un index, le charger avec des documents et exécuter des requêtes. Si vous êtes familiarisé avec les concepts de base, nous vous recommandons l'article suivant qui liste les opérations d'indexeur dans REST.

[Opérations d'indexeur](#)

# Démarrage rapide : Créer un index Recherche cognitive Azure dans Node.js à l'aide des API REST

04/10/2020 • 32 minutes to read • [Edit Online](#)

Créez une application Nodejs qui crée, charge et interroge un index Recherche cognitive Azure. Cet article explique comment créer l'application pas à pas. Vous pouvez aussi [télécharger le code source et les données](#), puis exécuter l'application à partir de la ligne de commande.

Si vous n'avez pas d'abonnement Azure, créez un [compte gratuit](#) avant de commencer.

## Prérequis

Nous avons utilisé les services et logiciels suivants pour générer et tester ce guide de démarrage rapide :

- [NodeJS](#)
- [NPM](#) doit être installé par Nodejs
- Un exemple de structure d'index et les documents correspondants sont fournis dans cet article et se trouvent dans le répertoire [quickstart du dépôt](#)
- [Créez un service Recherche cognitive Azure](#) ou [recherchez un service existant](#) dans votre abonnement actuel. Vous pouvez utiliser un service gratuit pour ce guide de démarrage rapide.

Recommandé :

- [Visual Studio Code](#)
- Extensions [Prettier](#) et [ESLint](#) pour VSCode.

## Obtenir des clés et des URL

Les appels au service nécessitent un point de terminaison d'URL et une clé d'accès pour chaque requête. Un service de recherche est créé avec les deux. Ainsi, si vous avez ajouté la Recherche cognitive Azure à votre abonnement, effectuez ce qui suit pour obtenir les informations nécessaires :

1. [Connectez-vous au Portail Azure](#), puis dans la page **Vue d'ensemble** du service de recherche, récupérez le nom de votre service de recherche. Vous pouvez confirmer le nom de votre service en passant en revue l'URL du point de terminaison. Si votre URL de point de terminaison est `https://mydemo.search.windows.net`, le nom du service doit être `mydemo`.
2. Dans **Paramètres > Clés**, obtenez une clé d'administration pour avoir des droits d'accès complets sur le service. Il existe deux clés d'administration interchangeables, fournies pour assurer la continuité de l'activité au cas où vous deviez en remplacer une. Vous pouvez utiliser la clé primaire ou secondaire sur les demandes d'ajout, de modification et de suppression d'objets.  
Obtenez aussi la clé de requête. Il est recommandé d'émettre des demandes de requête avec un accès en lecture seule.

The screenshot shows the Azure portal interface for managing keys. On the left, there's a sidebar with various service management options like 'Vue d'ensemble', 'Journal d'activité', and 'Paramètres'. The 'Clés' (Keys) option is selected. In the main pane, under 'Clé d'administration primaire', the placeholder value is shown. Below it, there's a table for managing request keys. A single row is present in the table, with the 'NAME' column showing the placeholder and the 'CLÉ' column also showing the placeholder.

Une clé API est nécessaire dans l'en-tête de chaque requête envoyée à votre service. Une clé valide permet d'établir, en fonction de chaque demande, une relation de confiance entre l'application qui envoie la demande et le service qui en assure le traitement.

## Configurer votre environnement

Commencez par ouvrir une console PowerShell ou un autre environnement dans lequel vous avez installé Node.js.

- Créez un répertoire de développement, que vous appellerez `quickstart` :

```
mkdir quickstart
cd quickstart
```

- Initialisez un projet vide avec NPM en exécutant `npm init`. Acceptez les valeurs par défaut, sauf pour la licence, que vous devez définir sur « MIT ».

- Ajoutez des packages qui seront dépendants du code et faciliteront le développement :

```
npm install nconf node-fetch
npm install --save-dev eslint eslint-config-prettier eslint-config-airbnb-base eslint-plugin-import
prettier
```

- Confirmez que vous avez configuré les projets et leurs dépendances en vérifiant que votre fichier `package.json` ressemble à ce qui suit :

```
{
  "name": "quickstart",
  "version": "1.0.0",
  "description": "Azure Cognitive Search Quickstart",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [
    "Azure",
    "Azure_Search"
  ],
  "author": "Your Name",
  "license": "MIT",
  "dependencies": {
    "nconf": "^0.10.0",
    "node-fetch": "^2.6.0"
  },
  "devDependencies": {
    "eslint": "^6.1.0",
    "eslint-config-airbnb-base": "^13.2.0",
    "eslint-config-prettier": "^6.0.0",
    "eslint-plugin-import": "^2.18.2",
    "prettier": "^1.18.2"
  }
}
```

5. Créez un fichier `azure_search_config.json` pour stocker les données de votre service de recherche :

```
{
  "serviceName" : "[SEARCH_SERVICE_NAME]",
  "adminKey" : "[ADMIN_KEY]",
  "queryKey" : "[QUERY_KEY]",
  "indexName" : "hotels-quickstart"
}
```

Remplacez la valeur `[SERVICE_NAME]` par le nom de votre service de recherche. Remplacez `[ADMIN_KEY]` et `[QUERY_KEY]` par les valeurs de clé que vous avez enregistrées précédemment.

## 1 - Créer un index

Créez un fichier `hotels_quickstart_index.json`. Ce fichier définit le fonctionnement de la Recherche cognitive Azure avec les documents que vous allez charger à l'étape suivante. Chaque champ est identifié par un `name` et a un `type` spécifié. Chaque champ contient également une série d'attributs d'index qui spécifient si la Recherche cognitive Azure peut effectuer des opérations de recherche, de tri, de filtrage et de définition de facettes sur le champ. La plupart des champs sont des types de données simples, mais certains, comme `AddressType`, sont des types complexes qui vous permettent de créer des structures de données riches dans votre index. Vous pouvez en savoir plus sur les [types de données pris en charge](#) et les [attributs d'index](#).

Ajoutez l'élément suivant à `hotels_quickstart_index.json` ou [téléchargez le fichier](#).

```
{
  "name": "hotels-quickstart",
  "fields": [
    {
      "name": "HotelId",
      "type": "Edm.String",
      "key": true,
      "filterable": true
    },
    {
      "name": "Name",
      "type": "Edm.String",
      "filterable": true
    },
    {
      "name": "Address",
      "type": "Edm.String",
      "filterable": true
    },
    {
      "name": "City",
      "type": "Edm.String",
      "filterable": true
    },
    {
      "name": "StateProvince",
      "type": "Edm.String",
      "filterable": true
    },
    {
      "name": "PostalCode",
      "type": "Edm.String",
      "filterable": true
    },
    {
      "name": "Country",
      "type": "Edm.String",
      "filterable": true
    },
    {
      "name": "Distance",
      "type": "Edm.Double"
    }
  ]
}
```

```
    },
    {
        "name": "HotelName",
        "type": "Edm.String",
        "searchable": true,
        "filterable": false,
        "sortable": true,
        "facetable": false
    },
    {
        "name": "Description",
        "type": "Edm.String",
        "searchable": true,
        "filterable": false,
        "sortable": false,
        "facetable": false,
        "analyzer": "en.lucene"
    },
    {
        "name": "Description_fr",
        "type": "Edm.String",
        "searchable": true,
        "filterable": false,
        "sortable": false,
        "facetable": false,
        "analyzer": "fr.lucene"
    },
    {
        "name": "Category",
        "type": "Edm.String",
        "searchable": true,
        "filterable": true,
        "sortable": true,
        "facetable": true
    },
    {
        "name": "Tags",
        "type": "Collection(Edm.String)",
        "searchable": true,
        "filterable": true,
        "sortable": false,
        "facetable": true
    },
    {
        "name": "ParkingIncluded",
        "type": "Edm.Boolean",
        "filterable": true,
        "sortable": true,
        "facetable": true
    },
    {
        "name": "LastRenovationDate",
        "type": "Edm.DateTimeOffset",
        "filterable": true,
        "sortable": true,
        "facetable": true
    },
    {
        "name": "Rating",
        "type": "Edm.Double",
        "filterable": true,
        "sortable": true,
        "facetable": true
    },
    {
        "name": "Address",
        "type": "Edm.ComplexType",
        "fields": [
            {
                "name": "StreetAddress",
                "type": "Edm.String"
            }
        ]
    }
]
```

```

        "type": "Edm.String",
        "filterable": false,
        "sortable": false,
        "facetable": false,
        "searchable": true
    },
    {
        "name": "City",
        "type": "Edm.String",
        "searchable": true,
        "filterable": true,
        "sortable": true,
        "facetable": true
    },
    {
        "name": "StateProvince",
        "type": "Edm.String",
        "searchable": true,
        "filterable": true,
        "sortable": true,
        "facetable": true
    },
    {
        "name": "PostalCode",
        "type": "Edm.String",
        "searchable": true,
        "filterable": true,
        "sortable": true,
        "facetable": true
    },
    {
        "name": "Country",
        "type": "Edm.String",
        "searchable": true,
        "filterable": true,
        "sortable": true,
        "facetable": true
    }
],
},
"suggesters": [
{
    "name": "sg",
    "searchMode": "analyzingInfixMatching",
    "sourceFields": [
        "HotelName"
    ]
}
]
}

```

Il est recommandé de distinguer les spécificités d'un scénario particulier du code qui seront largement applicables. La classe `AzureSearchClient` définie dans le fichier `AzureSearchClient.js` saura comment construire des URL de requête, effectuer une requête à l'aide de l'API Fetch et réagir au code d'état de la réponse.

Commencez à travailler sur le fichier `AzureSearchClient.js` en important le package `node-fetch` et en créant une classe simple. Isolez les parties modifiables de la classe `AzureSearchClient` en transmettant les différentes valeurs de configuration à son constructeur :

```

const fetch = require('node-fetch');

class AzureSearchClient {
    constructor(searchServiceName, adminKey, queryKey, indexName) {
        this.searchServiceName = searchServiceName;
        this.adminKey = adminKey;
        // The query key is used for read-only requests and so can be distributed with less risk of abuse.
        this.queryKey = queryKey;
        this.indexName = indexName;
        this.apiVersion = '2020-06-30';
    }

    // All methods go inside class body here!
}

module.exports = AzureSearchClient;

```

La première responsabilité de la classe est de savoir comment construire des URL auxquelles envoyer les diverses requêtes. Générez ces URL avec des méthodes d'instance qui utilisent les données de configuration transmises au constructeur de classe. Notez que l'URL qu'elles construisent est spécifique à une version de l'API et doit disposer d'un argument spécifiant cette version ( `2020-06-30` dans cette application).

La première de ces méthodes retourne l'URL de l'index lui-même. Ajoutez la méthode suivante à l'intérieur du corps de votre classe :

```

getIndexUrl() { return `https://${this.searchServiceName}.search.windows.net/indexes/${this.indexName}?api-
version=${this.apiVersion}`; }

```

La responsabilité suivante de `AzureSearchClient` consiste à effectuer une requête asynchrone avec l'API Fetch. La méthode statique asynchrone `request` utilise une URL, une chaîne spécifiant la méthode HTTP (« GET », « PUT », « POST », « DELETE »), la clé à utiliser dans la requête et un objet JSON facultatif. La variable `headers` mappe la `queryKey` (la clé d'administration ou la clé de requête en lecture seule) à l'en-tête de requête HTTP « api-key ». Les options de requête contiennent toujours le `method` à utiliser et le `headers`. Si `bodyJson` n'est pas `null`, le corps de la requête HTTP est défini sur la représentation de chaîne de `bodyJson`. La méthode `request` retourne la promesse de l'API Fetch d'exécuter la requête HTTP.

```

static async request(url, method, apiKey, bodyJson = null) {
    // Uncomment the following for request details:
    /*
    console.log(`\n${method} ${url}`);
    console.log(`\nKey ${apiKey}`);
    if (bodyJson !== null) {
        console.log(`\ncontent: ${JSON.stringify(bodyJson, null, 4)})`);
    }
    */

    const headers = {
        'content-type' : 'application/json',
        'api-key' : apiKey
    };
    const init = bodyJson === null ?
    {
        method,
        headers
    } :
    {
        method,
        headers,
        body : JSON.stringify(bodyJson)
    };
    return fetch(url, init);
}

```

À des fins de démonstration, levez une exception si la requête HTTP n'est pas un succès. Dans une application réelle, vous effectuerez probablement une journalisation et un diagnostic du code d'état HTTP dans le `response` à partir de la requête du service de recherche.

```

static throwOnHttpError(response) {
    const statusCode = response.status;
    if (statusCode >= 300){
        console.log(`Request failed: ${JSON.stringify(response, null, 4)})`);
        throw new Error(`Failure in request. HTTP Status was ${statusCode}`);
    }
}

```

Enfin, ajoutez les méthodes permettant de détecter, supprimer et créer l'index Recherche cognitive Azure. Ces méthodes ont toutes la même structure :

- Obtenir le point de terminaison auquel la requête sera envoyée.
- Générez la requête avec le point de terminaison, le verbe HTTP, la clé d'API appropriés, et, le cas échéant, un corps JSON. `indexExistsAsync()` et `deleteIndexAsync()` n'ont pas de corps JSON, contrairement à `createIndexAsync(definition)`.
- `await` la réponse à la requête.
- Action sur le code d'état de la réponse.
- Retourne la promesse d'une valeur appropriée (une valeur booléenne, `this`, ou les résultats de la requête).

```

async indexExistsAsync() {
    console.log("\n Checking if index exists...");
    const endpoint = this.getIndexUrl();
    const response = await AzureSearchClient.request(endpoint, "GET", this.adminKey);
    // Success has a few likely status codes: 200 or 204 (No Content), but accept all in 200 range...
    const exists = response.status >= 200 && response.status < 300;
    return exists;
}

async deleteIndexAsync() {
    console.log("\n Deleting existing index...");
    const endpoint = this.getIndexUrl();
    const response = await AzureSearchClient.request(endpoint, "DELETE", this.adminKey);
    AzureSearchClient.throwOnHttpError(response);
    return this;
}

async createIndexAsync(definition) {
    console.log("\n Creating index...");
    const endpoint = this.getIndexUrl();
    const response = await AzureSearchClient.request(endpoint, "PUT", this.adminKey, definition);
    AzureSearchClient.throwOnHttpError(response);
    return this;
}

```

Confirmez que vos méthodes sont contenues dans la classe et que vous exportez la classe. L'étendue la plus à l'extérieur de `AzureSearchClient.js` doit être :

```

const fetch = require('node-fetch');

class AzureSearchClient {
    // ... code here ...
}

module.exports = AzureSearchClient;

```

Une classe orientée objet représente un bon choix pour le module `AzureSearchClient.js` potentiellement réutilisable, mais n'est pas nécessaire pour le programme principal, que vous devez placer dans un fichier nommé `index.js`.

Créez `index.js` et commencez en introduisant :

- Le package `nconf`, qui vous offre la possibilité de spécifier la configuration avec JSON, les variables d'environnement ou les arguments de ligne de commande.
- Les données du fichier `hotels_quickstart_index.json`.
- Le module `AzureSearchClient`.

```

const nconf = require('nconf');

const indexDefinition = require('./hotels_quickstart_index.json');
const AzureSearchClient = require('./AzureSearchClient.js');

```

Le package `nconf` permet de spécifier des données de configuration dans différents formats, tels que des variables d'environnement ou la ligne de commande. Cet exemple utilise `nconf` de manière simple pour lire le fichier `azure_search_config.json` et retourner le contenu de ce fichier sous la forme d'un dictionnaire. À l'aide de la fonction `get(key)` de `nconf`, vous pouvez vérifier rapidement que les informations de configuration ont été correctement personnalisées. Enfin, la fonction retourne la configuration :

```

function getAzureConfiguration() {
    const config = nconf.file({ file: 'azure_search_config.json' });
    if (config.get('serviceName') === '[SEARCH_SERVICE_NAME]' ) {
        throw new Error("You have not set the values in your azure_search_config.json file. Change them to match your search service's values.");
    }
    return config;
}

```

La fonction `sleep` crée un `Promise` qui se résout après un laps de temps spécifié. L'utilisation de cette fonction permet à l'application de s'interrompre en attendant que les opérations d'index asynchrones se terminent et deviennent disponibles. L'ajout d'un délai n'est nécessaire que dans les démonstrations, les tests et les exemples d'applications.

```

function sleep(ms) {
    return(
        new Promise(function(resolve, reject) {
            setTimeout(function() { resolve(); }, ms);
        })
    );
}

```

Enfin, spécifiez et appelez la fonction `run` asynchrone principale. Cette fonction appelle les autres fonctions dans l'ordre, en attendant de résoudre `Promise`.

- Récupérer la configuration avec le `getAzureConfiguration()` que vous avez écrit précédemment
- Créer une nouvelle instance `AzureSearchClient`, en transmettant des valeurs à partir de votre configuration
- Vérifier si l'index existe et, le cas échéant, le supprimer
- Créer un index à l'aide du `indexDefinition` chargé à partir de `hotels_quickstart_index.json`

```

const run = async () => {
    try {
        const cfg = getAzureConfiguration();
        const client = new AzureSearchClient(cfg.get("serviceName"), cfg.get("adminKey"), cfg.get("queryKey"),
        cfg.get("indexName"));

        const exists = await client.indexExistsAsync();
        await exists ? client.deleteIndexAsync() : Promise.resolve();
        // Deleting index can take a few seconds
        await sleep(2000);
        await client.createIndexAsync(indexDefinition);
    } catch (x) {
        console.log(x);
    }
}

run();

```

N'oubliez pas l'appel final à `run()` ! C'est le point d'entrée de votre programme lorsque vous exécuterez `node index.js` à l'étape suivante.

Notez que `AzureSearchClient.indexExistsAsync()` et `AzureSearchClient.deleteIndexAsync()` ne prennent pas de paramètres. Ces fonctions appellent `AzureSearchClient.request()` sans argument `bodyJson`. Dans `AzureSearchClient.request()`, étant donné que `bodyJson === null` est `true`, la structure `init` est configurée pour comprendre uniquement le verbe HTTP (« GET » pour `indexExistsAsync()` et « DELETE » pour `deleteIndexAsync()`) et les en-têtes, qui spécifient la clé de requête.

En revanche, la méthode `AzureSearchClient.createIndexAsync(indexDefinition)` prend un paramètre. La fonction

run dans `index.js` transmet le contenu du fichier `hotels_quickstart_index.json` à la méthode `AzureSearchClient.createIndexAsync(indexDefinition)`. La méthode `createIndexAsync()` transmet cette définition à `AzureSearchClient.request()`. Dans `AzureSearchClient.request()`, étant donné que `bodyJson === null` est désormais `false`, la structure `init` comprend non seulement le verbe HTTP (« PUT ») et les en-têtes, mais configure la valeur `body` sur les données de définition d'index.

## Préparer et exécuter l'exemple

Utilisez une fenêtre de terminal pour exécuter les commandes suivantes.

1. Accédez au dossier qui contient le fichier `package.json` et le reste du code.
2. Installez les packages de l'exemple avec `npm install`. Cette commande permet de télécharger les packages dont dépend le code.
3. Exécutez votre programme avec `node index.js`.

Vous devez voir une série de messages décrivant les actions effectuées par le programme. Si vous souhaitez afficher plus de détails concernant les requêtes, vous pouvez supprimer les marques de commentaire des lignes au début de la méthode `AzureSearchClient.request()` (<https://github.com/Azure-Samples/azure-search-javascript-samples/blob/master/quickstart/AzureSearchClient.js#L21-L27>) dans `AzureSearchClient.js`.

Ouvrez la **Vue d'ensemble** de votre service de recherche dans le Portail Azure. Sélectionnez l'onglet **Index**. Un résultat tel que celui-ci doit s'afficher :

The screenshot shows the Azure portal's service overview for a search service. The left sidebar has a 'Vue d'ensemble' section with various settings like activity log, IAM, and diagnostics. Below it is a 'Paramètres' section with options for quick start, keys, scale, traffic analysis, identity, properties, locks, and model export. The main content area is titled 'Obtenez une fiabilité de 99,9 % garantie avec 3 répliques ou plus.' It displays resource group, URL (https://[REDACTED].search.windows.net), standard tier, 1 replica, 1 partition, and 1 unit of search. It also shows 0 documents and 0B storage. The 'Index' tab is selected in the navigation bar. A table at the bottom lists the index 'hotels-quickstart' with 0 documents and 0B storage.

NOM	NOMBRE DE DOCUMENTS	TAILLE DU STOCKAGE
hotels-quickstart	0	0 B

À l'étape suivante, vous ajouterez des données à l'index.

## 2 - Charger des documents

Dans la Recherche cognitive Azure, les documents sont des structures de données qui sont à la fois des entrées pour l'indexation et des sorties de requêtes. Vous devez poster (« POST ») ces données dans l'index. Cette opération nécessite un autre point de terminaison que celui des opérations effectuées à l'étape précédente. Ouvrez `AzureSearchClient.js` et ajoutez la méthode suivante après `getIndexUrl()` :

```
getPostDataUrl() { return  
`https://${this.searchServiceName}.search.windows.net/indexes/${this.indexName}/docs/index?api-  
version=${this.apiVersion}`; }
```

Comme `AzureSearchClient.createIndexAsync(definition)`, vous avez besoin d'une fonction qui appelle `AzureSearchClient.request()` et transmet les données d'hôtel pour en faire le corps. Dans `AzureSearchClient.js`,

ajoutez `postDataAsync(hotelsData)` après `createIndexAsync(definition)` :

```
async postDataAsync(hotelsData) {
    console.log("\n Adding hotel data...");
    const endpoint = this.getPostDataUrl();
    const response = await AzureSearchClient.request(endpoint, "POST", this.adminKey, hotelsData);
    AzureSearchClient.throwOnHttpError(response);
    return this;
}
```

Les entrées de documents peuvent être des lignes dans une base de données, des objets blob dans le Stockage Blob ou, comme dans cet exemple, des documents JSON sur le disque. Vous pouvez télécharger [hotels.json](#) ou créer votre propre fichier **hotels.json** avec le contenu suivant :

```
{
  "value": [
    {
      "HotelId": "1",
      "HotelName": "Secret Point Motel",
      "Description": "The hotel is ideally located on the main commercial artery of the city in the heart of New York. A few minutes away is Time's Square and the historic centre of the city, as well as other places of interest that make New York one of America's most attractive and cosmopolitan cities.",
      "Description_fr": "L'hôtel est idéalement situé sur la principale artère commerciale de la ville en plein cœur de New York. A quelques minutes se trouve la place du temps et le centre historique de la ville, ainsi que d'autres lieux d'intérêt qui font de New York l'une des villes les plus attractives et cosmopolites de l'Amérique.",
      "Category": "Boutique",
      "Tags": ["pool", "air conditioning", "concierge"],
      "ParkingIncluded": false,
      "LastRenovationDate": "1970-01-18T00:00:00Z",
      "Rating": 3.6,
      "Address": {
        "StreetAddress": "677 5th Ave",
        "City": "New York",
        "StateProvince": "NY",
        "PostalCode": "10022"
      }
    },
    {
      "HotelId": "2",
      "HotelName": "Twin Dome Motel",
      "Description": "The hotel is situated in a nineteenth century plaza, which has been expanded and renovated to the highest architectural standards to create a modern, functional and first-class hotel in which art and unique historical elements coexist with the most modern comforts.",
      "Description_fr": "L'hôtel est situé dans une place du XIXe siècle, qui a été agrandie et rénovée aux plus hautes normes architecturales pour créer un hôtel moderne, fonctionnel et de première classe dans lequel l'art et les éléments historiques uniques coexistent avec le confort le plus moderne.",
      "Category": "Boutique",
      "Tags": ["pool", "free wifi", "concierge"],
      "ParkingIncluded": "false",
      "LastRenovationDate": "1979-02-18T00:00:00Z",
      "Rating": 3.6,
      "Address": {
        "StreetAddress": "140 University Town Center Dr",
        "City": "Sarasota",
        "StateProvince": "FL",
        "PostalCode": "34243"
      }
    },
    {
      "HotelId": "3",
      "HotelName": "Triple Landscape Hotel",
      "Description": "The Hotel stands out for its gastronomic excellence under the management of William Dough, who advises on and oversees all of the Hotel's restaurant services.",
      "Description_fr": "L'hôtel est situé dans une place du XIXe siècle, qui a été agrandie et rénovée
```

```

aux plus hautes normes architecturales pour créer un hôtel moderne, fonctionnel et de première classe dans
lequel l'art et les éléments historiques uniques coexistent avec le confort le plus moderne.",

    "Category": "Resort and Spa",
    "Tags": ["air conditioning", "bar", "continental breakfast"],
    "ParkingIncluded": "true",
    "LastRenovationDate": "2015-09-20T00:00:00Z",
    "Rating": 4.8,
    "Address": {
        "StreetAddress": "3393 Peachtree Rd",
        "City": "Atlanta",
        "StateProvince": "GA",
        "PostalCode": "30326"
    }
},
{
    "HotelId": "4",
    "HotelName": "Sublime Cliff Hotel",
    "Description": "Sublime Cliff Hotel is located in the heart of the historic center of Sublime in
an extremely vibrant and lively area within short walking distance to the sites and landmarks of the city and
is surrounded by the extraordinary beauty of churches, buildings, shops and monuments. Sublime Cliff is part
of a lovingly restored 1800 palace.",
    "Description_fr": "Le sublime Cliff Hotel est situé au cœur du centre historique de sublime dans
un quartier extrêmement animé et vivant, à courte distance de marche des sites et monuments de la ville et est
entouré par l'extraordinaire beauté des églises, des bâtiments, des commerces et Monuments. Sublime Cliff fait
partie d'un Palace 1800 restauré avec amour.",
    "Category": "Boutique",
    "Tags": ["concierge", "view", "24-hour front desk service"],
    "ParkingIncluded": true,
    "LastRenovationDate": "1960-02-06T00:00:00Z",
    "Rating": 4.6,
    "Address": {
        "StreetAddress": "7400 San Pedro Ave",
        "City": "San Antonio",
        "StateProvince": "TX",
        "PostalCode": "78216"
    }
}
]
}

```

Pour charger ces données dans votre programme, modifiez le fichier **index.js** en ajoutant la ligne faisant référence  
à `hotelData` dans la partie supérieure :

```

const nconf = require('nconf');

const hotelData = require('./hotels.json');
const indexDefinition = require('./hotels_quickstart_index.json');

```

À présent, modifiez la fonction `run()` dans le fichier **index.js**. Il faut parfois attendre plusieurs secondes avant que  
l'index ne soit disponible. Ajoutez donc une pause de 2 secondes avant d'appeler

```
AzureSearchClient.postDataAsync(hotelData) :
```

```

const run = async () => {
  try {
    const cfg = getAzureConfiguration();
    const client = new AzureSearchClient(cfg.get("serviceName"), cfg.get("adminKey"), cfg.get("queryKey"),
      cfg.get("indexName"));

    const exists = await client.indexExistsAsync();
    await exists ? client.deleteIndexAsync() : Promise.resolve();
    // Deleting index can take a few seconds
    await sleep(2000);
    await client.createIndexAsync(indexDefinition);
    // Index availability can take a few seconds
    await sleep(2000);
    await client.postDataAsync(hotelData);
  } catch (x) {
    console.log(x);
  }
}

```

Exécutez à nouveau le programme avec `node index.js`. Vous devriez voir un ensemble de messages légèrement différents de ceux que vous avez vus à l'étape 1. Cette fois-ci, l'index *existe* et un message doit s'afficher pour vous permettre de le supprimer avant que l'application ne crée le nouvel index et n'y envoie des données.

### 3 – Rechercher dans un index

Revenez à l'onglet **Index** dans la **Vue d'ensemble** de votre service de recherche dans le Portail Azure. Votre index contient désormais quatre documents et consomme une certaine quantité de stockage (l'interface utilisateur peut prendre quelques minutes pour refléter correctement l'état sous-jacent de l'index). Cliquez sur le nom de l'index à utiliser dans l'**Explorateur de recherche**. Cette page vous permet d'effectuer des tests avec les requêtes de données. Essayez d'effectuer une recherche sur une chaîne de requête de `*&$count=true`. Vous devriez récupérer tous vos documents et le nombre de résultats. Essayez avec la chaîne de requête

`historic&highlight=Description&$filter=Rating gt 4`. Vous devriez récupérer un document unique, avec le mot « historic » encapsulé dans des balises `<em></em>`. En savoir plus sur la [composition d'une requête dans la Recherche cognitive Azure](#).

Reproduisez ces requêtes dans le code en ouvrant le fichier `index.js` et en ajoutant ce code dans la partie supérieure :

```

const queries = [
  "*&$count=true",
  "historic&highlight=Description&$filter=Rating gt 4&"
];

```

Dans le même fichier `index.js`, écrivez la fonction `doQueriesAsync()` indiquée ci-dessous. Cette fonction prend un objet `AzureSearchClient` et applique la méthode `AzureSearchClient.queryAsync` à chacune des valeurs du tableau `queries`. Elle utilise la fonction `Promise.all()` pour retourner un `Promise` unique, qui est résolu uniquement lorsque toutes les requêtes ont été résolues. L'appel à `JSON.stringify(body, null, 4)` met en forme le résultat de la requête pour qu'il soit plus lisible.

```

async function doQueriesAsync(client) {
    return Promise.all(
        queries.map( async query => {
            const result = await client.queryAsync(query);
            const body = await result.json();
            const str = JSON.stringify( body, null, 4);
            console.log(`Query: ${query} \n ${str}`);
        })
    );
}

```

Modifiez la fonction `run()` afin qu'elle s'interrompe suffisamment longtemps pour que l'indexeur fonctionne, puis appelez la fonction `doQueriesAsync(client)` :

```

const run = async () => {
    try {
        const cfg = getAzureConfiguration();
        const client = new AzureSearchClient(cfg.get("serviceName"), cfg.get("adminKey"), cfg.get("queryKey"),
        cfg.get("indexName"));

        const exists = await client.indexExistsAsync();
        await exists ? client.deleteIndexAsync() : Promise.resolve();
        // Deleting index can take a few seconds
        await sleep(2000);
        await client.createIndexAsync(indexDefinition);
        // Index availability can take a few seconds
        await sleep(2000);
        await client postDataAsync(hotelData);
        // Data availability can take a few seconds
        await sleep(5000);
        await doQueriesAsync(client);
    } catch (x) {
        console.log(x);
    }
}

```

Pour implémenter `AzureSearchClient.queryAsync(query)`, modifiez le fichier `AzureSearchClient.js`. La recherche requiert un point de terminaison différent, et les termes de recherche deviennent des arguments d'URL. Ajoutez donc la fonction `getSearchUrl(searchTerm)` avec les méthodes `getIndexUrl()` et `getPostDataUrl()` que vous avez déjà écrites.

```

getSearchUrl(searchTerm) { return
`https://${this.searchServiceName}.search.windows.net/indexes/${this.indexName}/docs?api-
version=${this.apiVersion}&search=${searchTerm}&searchMode=all`;
}

```

La fonction `queryAsync(searchTerm)` est également transmise à `AzureSearchClient.js` et suit la même structure que `postDataAsync(data)` et les autres fonctions de requête :

```

async queryAsync(searchTerm) {
    console.log("\n Querying...")
    const endpoint = this.getSearchUrl(searchTerm);
    const response = await AzureSearchClient.request(endpoint, "GET", this.queryKey);
    AzureSearchClient.throwOnHttpError(response);
    return response;
}

```

La recherche s'effectue avec le verbe « GET », et sans corps, car le terme de recherche fait partie de l'URL. Notez que `queryAsync(searchTerm)` utilise `this.queryKey`, contrairement aux autres fonctions qui utilisaient la clé

d'administration. Comme leur nom l'indique, les clés de requête ne peuvent être utilisées que pour interroger l'index, et pas pour modifier l'index de quelque façon que ce soit. Les clés de requête sont donc plus sûres à distribuer aux applications clientes.

Exécutez le programme avec `node index.js`. À présent, en plus des étapes précédentes, les requêtes sont envoyées et les résultats sont écrits dans la console.

## À propos de l'exemple

L'exemple utilise une petite quantité de données relatives à un hôtel. Elle est suffisante pour illustrer les bases de la création et de l'interrogation d'un index Recherche cognitive Azure.

La classe `AzureSearchClient` encapsule la configuration, les URL et les requêtes HTTP de base pour le service de recherche. Le fichier `index.js` charge les données de configuration du service Recherche cognitive Azure ainsi que les données relatives aux hôtels, qui vont être chargées pour l'indexation ; en outre, dans sa fonction `run`, il ordonne et exécute les différentes opérations.

Le comportement global de la fonction `run` consiste à supprimer l'index Recherche cognitive Azure s'il existe, à créer l'index, à ajouter des données et à effectuer des requêtes.

## Nettoyer les ressources

Lorsque vous travaillez dans votre propre abonnement, il est recommandé, à la fin de chaque projet, de déterminer si vous avez toujours besoin des ressources que vous avez créées. Les ressources laissées en cours d'exécution peuvent vous coûter de l'argent. Vous pouvez supprimer les ressources une par une, ou choisir de supprimer le groupe de ressources afin de supprimer l'ensemble des ressources.

Vous pouvez rechercher et gérer les ressources dans le portail à l'aide des liens **Toutes les ressources** ou **Groupes de ressources** situés dans le volet de navigation de gauche.

Si vous utilisez un service gratuit, n'oubliez pas que vous êtes limité à trois index, indexeurs et sources de données. Vous pouvez supprimer des éléments un par un dans le portail pour ne pas dépasser la limite.

## Étapes suivantes

Dans ce guide de démarrage rapide Node.js, vous avez effectué une série de tâches pour créer un index, le charger avec des documents et exécuter des requêtes. Nous avons effectué certaines étapes, telles que la lecture de la configuration et la définition des requêtes, de la manière la plus simple possible. Dans une application réelle, vous résoudrez ces problèmes dans des modules distincts qui offrent flexibilité et encapsulation.

Si vous connaissez déjà la Recherche cognitive Azure, vous pouvez utiliser cet exemple comme tremplin pour tester des suggesteurs (requêtes prédictives ou d'autocomplétion), des filtres et la navigation par facettes. Si vous débutez avec la Recherche cognitive Azure, nous vous recommandons de suivre d'autres tutoriels pour mieux comprendre ce que vous pouvez créer. Consultez les autres ressources disponibles dans notre [page de documentation](#).

[Appeler la Recherche cognitive Azure à partir d'une page web en JavaScript](#)

# Démarrage rapide : Créer un index Recherche cognitive Azure dans Postman à l'aide des API REST

04/10/2020 • 17 minutes to read • [Edit Online](#)

Cet article explique comment formuler des requêtes API REST de façon interactive à l'aide des [API REST de la Recherche cognitive Azure](#) et d'un API client pour envoyer et recevoir les requêtes. Avec un API client et ces instructions, vous pouvez envoyer des requêtes et afficher les réponses avant d'écrire un code.

L'article utilise l'application Postman. Vous pouvez [télécharger et importer une collection Postman](#) si vous préférez utiliser des requêtes prédéfinies.

Si vous n'avez pas d'abonnement Azure, créez un [compte gratuit](#) avant de commencer.

## Prérequis

Les services et outils suivants sont indispensables dans ce guide de démarrage rapide.

- L'[application de bureau Postman](#) permet d'envoyer des requêtes à la Recherche cognitive Azure.
- [Créez un service Recherche cognitive Azure](#) ou [recherchez un service existant](#) dans votre abonnement actuel. Vous pouvez utiliser un service gratuit pour ce guide de démarrage rapide.

## Obtenir une clé et une URL

Les appels REST requièrent l'URL du service et une clé d'accès et ce, sur chaque demande. Un service de recherche est créé avec les deux. Ainsi, si vous avez ajouté la Recherche cognitive Azure à votre abonnement, effectuez ce qui suit pour obtenir les informations nécessaires :

1. [Connectez-vous au portail Azure](#), puis dans la page **Vue d'ensemble** du service de recherche, récupérez l'URL. Voici un exemple de point de terminaison : `https://mydemo.search.windows.net`.
2. Dans **Paramètres > Clés**, obtenez une clé d'administration pour avoir des droits d'accès complets sur le service. Il existe deux clés d'administration interchangeables, fournies pour assurer la continuité de l'activité au cas où vous deviez en remplacer une. Vous pouvez utiliser la clé primaire ou secondaire sur les demandes d'ajout, de modification et de suppression d'objets.

The screenshot shows two overlapping Azure portal windows. The top window is titled 'mydemo' and displays the 'Vue d'ensemble' (Overview) page for a search service. It includes a search bar, navigation links like 'Ajouter un index', 'Import...', 'Explorateur...', 'Actualiser', 'Supprimer', and 'Déplacer'. A red box highlights the 'Vue d'ensemble' link in the sidebar, and another red box highlights the URL 'https://mydemo.search.windows.net' in the top right. A red circle with the number '1' is positioned near the URL. The bottom window is titled 'mydemo - Clés' (Keys) and shows the 'Clés' (Keys) section. It has a sidebar with links like 'Vue d'ensemble', 'Journal d'activité', 'Contrôle d'accès (IAM)', 'Paramètres', 'Démarrage rapide', and 'Clés'. A red box highlights the 'Clés' link in the sidebar, and another red circle with the number '2' is positioned near this link. The main content area shows two fields: 'Clé d'administration primaire' containing '<placeholder-for-alphanumeric-autogenerated-string>' and 'Clé d'administration secondaire' also containing '<placeholder-for-alphanumeric-autogenerated-string>'. Both fields have a blue download icon.

Toutes les demandes nécessitent une clé API sur chaque demande envoyée à votre service. L'utilisation d'une clé valide permet d'établir, en fonction de chaque demande, une relation de confiance entre l'application qui envoie la demande et le service qui en assure le traitement.

## Se connecter à la Recherche cognitive Azure

Dans cette section, utilisez l'outil web de votre choix pour configurer les connexions à la Recherche cognitive Azure. Chaque outil conserve les informations d'en-tête de requête de la session, ce qui signifie que vous n'avez à indiquer la clé d'API et le type de contenu qu'une seule fois.

Quel que soit l'outil, vous devez choisir une commande (GET, POST, PUT, etc.), fournir un point de terminaison d'URL et, pour certaines tâches, fournir du code JSON dans le corps de la requête. Remplacez le nom du service de recherche (YOUR-SEARCH-SERVICE-NAME) par une valeur valide. Ajoutez `$select=name` pour retourner uniquement le nom de chaque index.

```
https://<YOUR-SEARCH-SERVICE-NAME>.search.windows.net/indexes?api-version=2020-06-30&$select=name
```

Remarquez le préfixe HTTPS, le nom du service, le nom de l'objet (dans ce cas, la collection d'index) et la [version de l'API](#). La version de l'API est une chaîne en minuscules obligatoire, spécifiée au format `?api-version=2020-06-30` pour la version actuelle. Les versions d'API sont régulièrement mises à jour. Le fait d'inclure la version d'API sur chaque demande vous permet de bénéficier du contrôle absolu sur la version utilisée.

Un en-tête de requête se compose de deux éléments : `Content-Type` et la `api-key` utilisée pour s'authentifier auprès de la Recherche cognitive Azure. Remplacez la clé API administrateur (YOUR-AZURE-SEARCH-ADMIN-API-KEY) par une valeur valide.

```
api-key: <YOUR-AZURE-SEARCH-ADMIN-API-KEY>
Content-Type: application/json
```

Dans Postman, formulez une requête similaire à celle de la capture d'écran suivante. Choisissez **GET** comme commande, indiquez l'URL, puis cliquez sur **Envoyer**. Cette commande se connecte à la Recherche cognitive Azure, lit la collection d'index et retourne le code d'état HTTP 200 en cas de réussite de la connexion. Si votre service a déjà des index, la réponse inclut également les définitions de ces index.

GET https://mydemo.search.windows.net/indexes?api-version=2020-06-30

En-têtes (2)

Clé	Valeur	Description
<input checked="" type="checkbox"/> Content-Type	application/json	
<input checked="" type="checkbox"/> api-key	<placeholder-api-key-for-your-service>	

Nouvelle clé Valeur Description

Corps Cookies En-têtes (14) Etat : 200 OK Heure : 540 ms

## 1 – Créez un index

Dans la Recherche cognitive Azure, vous créez généralement l'index avant de le charger avec des données. L'[API REST Create Index](#) est utilisée pour cette tâche.

L'URL est étendue pour inclure le nom d'index `hotels`.

Pour faire cela dans Postman :

1. Définissez la commande sur **PUT**.

2. Copiez dans cette URL

```
https://<YOUR-SEARCH-SERVICE-NAME>.search.windows.net/indexes/hotels-quickstart?api-version=2020-06-30
```

3. Fournissez la définition d'index (un code prêt à copier est fourni ci-dessous) dans le corps de la demande.

4. Cliquez sur **Envoyer**.

```

1  {
2    "name": "hotels-quickstart",
3    "fields": [
4      {"name": "HotelId", "type": "Edm.String", "key": true, "filterable": true},
5      {"name": "HotelName", "type": "Edm.String", "searchable": true, "filterable": false, "sortable": true, "facetable": false},
6      {"name": "Description", "type": "Edm.String", "searchable": true, "filterable": false, "sortable": false, "facetable": false, "analyzer": "en.lucene"},
7      {"name": "Category", "type": "Edm.String", "searchable": true, "filterable": true, "sortable": true, "facetable": true},
8      {"name": "Tags", "type": "Collection(Edm.String)", "searchable": true, "filterable": true, "sortable": false, "facetable": true},
9      {"name": "ParkingIncluded", "type": "Edm.Boolean", "filterable": true, "sortable": true, "facetable": true},
10     {"name": "LastRenovationDate", "type": "Edm.DateTimeOffset", "filterable": true, "sortable": true, "facetable": true},
11     {"name": "Rating", "type": "Edm.Double", "filterable": true, "sortable": true, "facetable": true},
12     {"name": "Address", "type": "Edm.ComplexType",
13       "fields": [
14         {"name": "StreetAddress", "type": "Edm.String", "filterable": false, "sortable": false, "facetable": false, "searchable": true},
15         {"name": "City", "type": "Edm.String", "searchable": true, "filterable": true, "sortable": true, "facetable": true},
16         {"name": "StateProvince", "type": "Edm.String", "searchable": true, "filterable": true, "sortable": true, "facetable": true},
17         {"name": "PostalCode", "type": "Edm.String", "searchable": true, "filterable": true, "sortable": true, "facetable": true},
18         {"name": "Country", "type": "Edm.String", "searchable": true, "filterable": true, "sortable": true, "facetable": true}
19       ]
20     }
21   }
22 }
```

### Définition de l'index

La collection de champs définit la structure du document. Chaque document doit comporter ces champs, et chaque champ doit avoir un type de données. Les champs de chaîne sont utilisés dans la recherche en texte intégral. Si vous avez besoin de données numériques pouvant faire l'objet d'une recherche, vous devez caster les données numériques comme chaînes.

Les attributs du champ déterminent l'action autorisée. Les API REST autorisent de nombreuses actions par défaut. Par exemple, toutes les chaînes peuvent être soumises à des recherches, récupérables et filtrables et sont à choix multiples par défaut. Bien souvent, il suffit de définir des attributs pour désactiver un comportement.

```
{
  "name": "hotels-quickstart",
  "fields": [
    {"name": "HotelId", "type": "Edm.String", "key": true, "filterable": true},
    {"name": "HotelName", "type": "Edm.String", "searchable": true, "filterable": false, "sortable": true, "facetable": false},
    {"name": "Description", "type": "Edm.String", "searchable": true, "filterable": false, "sortable": false, "facetable": false, "analyzer": "en.lucene"},
    {"name": "Category", "type": "Edm.String", "searchable": true, "filterable": true, "sortable": true, "facetable": true},
    {"name": "Tags", "type": "Collection(Edm.String)", "searchable": true, "filterable": true, "sortable": false, "facetable": true},
    {"name": "ParkingIncluded", "type": "Edm.Boolean", "filterable": true, "sortable": true, "facetable": true},
    {"name": "LastRenovationDate", "type": "Edm.DateTimeOffset", "filterable": true, "sortable": true, "facetable": true},
    {"name": "Rating", "type": "Edm.Double", "filterable": true, "sortable": true, "facetable": true},
    {"name": "Address", "type": "Edm.ComplexType",
      "fields": [
        {"name": "StreetAddress", "type": "Edm.String", "filterable": false, "sortable": false, "facetable": false, "searchable": true},
        {"name": "City", "type": "Edm.String", "searchable": true, "filterable": true, "sortable": true, "facetable": true},
        {"name": "StateProvince", "type": "Edm.String", "searchable": true, "filterable": true, "sortable": true, "facetable": true},
        {"name": "PostalCode", "type": "Edm.String", "searchable": true, "filterable": true, "sortable": true, "facetable": true},
        {"name": "Country", "type": "Edm.String", "searchable": true, "filterable": true, "sortable": true, "facetable": true}
      ]
    }
  ]
}
```

Lorsque vous envoyez cette demande, vous devez obtenir une réponse HTTP 201, indiquant que l'index a bien été créé. Vous pouvez vérifier cette action dans le portail, mais n'oubliez pas que la page du portail est actualisée à intervalles réguliers. Il se peut donc que le résultat de l'action s'affiche au bout d'une minute ou deux.

#### TIP

Si vous obtenez HTTP 504, vérifiez que l'URL spécifie HTTPS. Si vous voyez HTTP 400 ou 404, contrôlez le corps de la demande pour vérifier l'absence d'erreurs de copier-coller. HTTP 403 indique normalement qu'il y a un problème avec la clé API (soit la clé n'est pas valide, soit il y a un problème de syntaxe avec la façon dont elle est spécifiée).

## 2 – Charger des documents

Les étapes de création et de remplissage d'index sont des opérations distinctes. Dans la Recherche cognitive Azure, l'index contient toutes les données pouvant faire l'objet de recherches. Dans ce scénario, les données sont fournies sous forme de documents JSON. L'[API REST Add, Update ou Delete Documents](#) est utilisée pour cette tâche.

L'URL est étendue pour inclure les collections `docs` et l'opération `index`.

Pour faire cela dans Postman :

1. Définissez la commande sur POST.
2. Copiez dans cette URL

```
https://<YOUR-SEARCH-SERVICE-NAME>.search.windows.net/indexes/hotels-quickstart/docs/index?api-version=2020-06-30
```

3. Fournissez les documents JSON (un code prêt à copier est fourni ci-dessous) dans le corps de la demande.

4. Cliquez sur Envoyer.

```
POST https://<YOUR-SEARCH-SERVICE-NAME>.search.windows.net/indexes/hotels-quickstart/docs/index?api-version=2020-06-30
```

Body Content-Type: application/json

```
1 {  
2   "value": [  
3     {  
4       "@search.action": "upload",  
5       "HotelId": "1",  
6       "BaseRate": 199.0,  
7       "Description": "Best hotel in town",  
8       "Description_fr": "Meilleur h\u00f4tel en ville",  
9       "HotelName": "Fancy Stay",  
10      "Category": "Luxury",  
11      "Tags": ["pool", "view", "wifi", "concierge"],  
12      "ParkingIncluded": false,  
13      "SmokingAllowed": false,  
14      "LastRenovationDate": "2010-06-27T00:00:00Z",  
15      "Rating": 5,  
16      "Location": { "type": "Point", "coordinates": [-122.131577, 47.678581] }  
17    },  
18    {  
19      "@search.action": "upload",  
20      "HotelId": "2",  
21      "HotelName": "Secret Point Motel",  
22      "Description": "The hotel is ideally located on the main commercial artery of the city in the heart of New York. A few minutes away is Time's Square and the historic centre of the city, as well as other places of interest that make New York one of America's most attractive and cosmopolitan cities.",  
23      "Category": "Boutique",  
24      "Tags": [ "pool", "air conditioning", "concierge" ],  
25      "ParkingIncluded": false,  
26      "LastRenovationDate": "1970-01-18T00:00:00Z",  
27      "Rating": 3.60,  
28      "Address":  
29        {  
30          "StreetAddress": "677 5th Ave",  
31          "City": "New York",  
32          "StateProvince": "NY",  
33          "PostalCode": "10022",  
34          "Country": "USA"  
35        }  
36    },  
37    {  
38      "@search.action": "upload",  
39      "HotelId": "3",  
40      "HotelName": "Twin Dome Motel",  
41      "Description": "The hotel is situated in a nineteenth century plaza, which has been expanded and renovated to the highest architectural standards to create a modern, functional and first-class hotel in which art and unique historical elements coexist with the most modern comforts.",  
42      "Category": "Boutique",  
43      "Tags": [ "pool", "free wifi", "concierge" ],  
44      "ParkingIncluded": false,  
45      "LastRenovationDate": "1979-02-18T00:00:00Z",  
46      "Rating": 3.60,  
47      "Address":  
48        {  
49          "StreetAddress": "140 University Town Center Dr",  
50          "City": "Sarasota",  
51        }  
52    }  
53  ]  
54}
```

### Documents JSON à charger dans l'index

Le corps de la demande contient quatre documents à ajouter à l'index des hôtels.

```
{  
  "value": [  
    {  
      "@search.action": "upload",  
      "HotelId": "1",  
      "HotelName": "Secret Point Motel",  
      "Description": "The hotel is ideally located on the main commercial artery of the city in the heart of New York. A few minutes away is Time's Square and the historic centre of the city, as well as other places of interest that make New York one of America's most attractive and cosmopolitan cities.",  
      "Category": "Boutique",  
      "Tags": [ "pool", "air conditioning", "concierge" ],  
      "ParkingIncluded": false,  
      "LastRenovationDate": "1970-01-18T00:00:00Z",  
      "Rating": 3.60,  
      "Address":  
        {  
          "StreetAddress": "677 5th Ave",  
          "City": "New York",  
          "StateProvince": "NY",  
          "PostalCode": "10022",  
          "Country": "USA"  
        }  
    },  
    {  
      "@search.action": "upload",  
      "HotelId": "2",  
      "HotelName": "Twin Dome Motel",  
      "Description": "The hotel is situated in a nineteenth century plaza, which has been expanded and renovated to the highest architectural standards to create a modern, functional and first-class hotel in which art and unique historical elements coexist with the most modern comforts.",  
      "Category": "Boutique",  
      "Tags": [ "pool", "free wifi", "concierge" ],  
      "ParkingIncluded": false,  
      "LastRenovationDate": "1979-02-18T00:00:00Z",  
      "Rating": 3.60,  
      "Address":  
        {  
          "StreetAddress": "140 University Town Center Dr",  
          "City": "Sarasota",  
        }  
    }  
  ]  
}
```

```

        "StateProvince": "FL",
        "PostalCode": "34243",
        "Country": "USA"
    }
},
{
"@search.action": "upload",
"HotelId": "3",
"HotelName": "Triple Landscape Hotel",
"Description": "The Hotel stands out for its gastronomic excellence under the management of William Dough, who advises on and oversees all of the Hotel's restaurant services.",
"Category": "Resort and Spa",
"Tags": [ "air conditioning", "bar", "continental breakfast" ],
"ParkingIncluded": true,
"LastRenovationDate": "2015-09-20T00:00:00Z",
"Rating": 4.80,
"Address":
{
    "StreetAddress": "3393 Peachtree Rd",
    "City": "Atlanta",
    "StateProvince": "GA",
    "PostalCode": "30326",
    "Country": "USA"
}
},
{
"@search.action": "upload",
"HotelId": "4",
"HotelName": "Sublime Cliff Hotel",
"Description": "Sublime Cliff Hotel is located in the heart of the historic center of Sublime in an extremely vibrant and lively area within short walking distance to the sites and landmarks of the city and is surrounded by the extraordinary beauty of churches, buildings, shops and monuments. Sublime Cliff is part of a lovingly restored 1800 palace.",
"Category": "Boutique",
"Tags": [ "concierge", "view", "24-hour front desk service" ],
"ParkingIncluded": true,
"LastRenovationDate": "1960-02-06T00:00:00Z",
"Rating": 4.60,
"Address":
{
    "StreetAddress": "7400 San Pedro Ave",
    "City": "San Antonio",
    "StateProvince": "TX",
    "PostalCode": "78216",
    "Country": "USA"
}
}
]
}

```

Après quelques secondes, la réponse HTTP 201 apparaît dans la liste de sessions. Cela indique que les documents ont été correctement créés.

Si vous obtenez HTTP 207, cela signifie qu'au moins un document n'a pas pu être chargé. Si l'erreur HTTP 404 s'affiche, cela signifie que vous avez fait une erreur de syntaxe dans l'en-tête ou le corps de la demande. Vérifiez que le point de terminaison a bien été modifié de manière à inclure `/docs/index`.

#### TIP

Pour les sources de données sélectionnées, vous pouvez choisir l'approche *d'indexeur*, qui simplifie et réduit la quantité de code nécessaire pour l'indexation. Pour en savoir plus, consultez la section relative aux [opérations de l'indexeur](#).

## 3 – Rechercher dans un index

Maintenant qu'un index et qu'un ensemble de documents sont chargés, vous pouvez émettre des requêtes les concernant à l'aide de l'[API REST Search Documents](#).

L'URL est étendue pour inclure une expression de requête spécifiée à l'aide de l'opérateur de recherche.

Pour faire cela dans Postman :

1. Définissez la commande sur GET.

2. Copiez dans cette URL

```
https://<YOUR-SEARCH-SERVICE-NAME>.search.windows.net/indexes/hotels-quickstart/docs?search=*&$count=true&api-version=2020-06-30
```

3. Cliquez sur Envoyer.

Cette requête est vide et renvoie le nombre des documents dans les résultats de recherche. La requête et la réponse doivent ressembler à la capture d'écran suivante pour Postman, une fois que vous avez cliqué sur Envoyer. Le code d'état doit être 200.



Essayez quelques autres exemples de requête pour avoir un aperçu de la syntaxe. Vous pouvez effectuer une recherche de chaîne, textualiser les requêtes \$filter, limiter le jeu de résultats, restreindre la recherche à des champs spécifiques, etc.

Remplacez l'URL actuelle par celles ci-dessous, en cliquant sur Envoyer à chaque fois pour afficher les résultats.

```
# Query example 1 - Search on restaurant and wifi
# Return only the HotelName, Description, and Tags fields
https://<YOUR-SEARCH-SERVICE>.search.windows.net/indexes/hotels-quickstart/docs?search=restaurant
wifi&$count=true&$select=HotelName,Description,Tags&api-version=2020-06-30

# Query example 2 - Apply a filter to the index to find hotels rated 4 or higher
# Returns the HotelName and Rating. Two documents match
https://<YOUR-SEARCH-SERVICE>.search.windows.net/indexes/hotels-quickstart/docs?search=*&$filter=Rating
gt 4&$select=HotelName,Rating&api-version=2020-06-30

# Query example 3 - Take the top two results, and show only HotelName and Category in the results
https://<YOUR-SEARCH-SERVICE>.search.windows.net/indexes/hotels-quickstart/docs?
search=boutique&$top=2&$select=HotelName,Category&api-version=2020-06-30

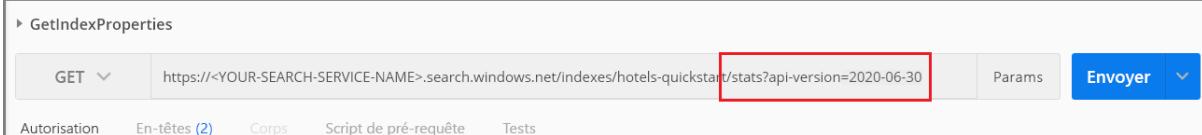
# Query example 4 - Sort by a specific field (Address/City) in ascending order
https://<YOUR-SEARCH-SERVICE>.search.windows.net/indexes/hotels-quickstart/docs?
search=pool&$orderby=Address/City asc&$select=HotelName, Address/City, Tags, Rating&api-version=2020-06-
30
```

## Obtenez les propriétés de l'index

Vous pouvez également utiliser la commande [Obtenir des statistiques](#) pour interroger les nombres de documents et la taille de l'index :

```
https://<YOUR-SEARCH-SERVICE-NAME>.search.windows.net/indexes/hotels-quickstart/stats?api-version=2020-
06-30
```

L'ajout de `/stats` à votre URL retourne des informations d'index. Dans Postman, votre requête doit ressembler à ce qui suit ; la réponse inclut un nombre de documents et la quantité d'espace utilisé, en octets.



The screenshot shows the Postman interface with a 'GetIndexProperties' collection. A GET request is selected with the URL `https://<YOUR-SEARCH-SERVICE-NAME>.search.windows.net/indexes/hotels-quickstart/stats?api-version=2020-06-30`. The 'Params' tab is visible, and a blue 'Envoyer' (Send) button is at the bottom right. Below the main interface, there are tabs for 'Autorisation', 'En-têtes (2)', 'Corps', 'Script de pré-requête', and 'Tests'. The URL path 'stats?api-version=2020-06-30' is highlighted with a red box.

Notez que la syntaxe de la version d'API diffère. Pour cette requête, utilisez `?`  pour ajouter la version d'API. `?` sépare le chemin de l'URL de la chaîne de requête, alors que le signe `&` sépare chaque paire « nom=valeur » dans la chaîne de requête. Pour cette requête, la version d'API est le premier et seul élément de la chaîne de requête.

## Nettoyer les ressources

Lorsque vous travaillez dans votre propre abonnement, il est recommandé, à la fin de chaque projet, de déterminer si vous avez toujours besoin des ressources que vous avez créées. Les ressources laissées en cours d'exécution peuvent vous coûter de l'argent. Vous pouvez supprimer les ressources une par une, ou choisir de supprimer le groupe de ressources afin de supprimer l'ensemble des ressources.

Vous pouvez rechercher et gérer les ressources dans le portail à l'aide des liens **Toutes les ressources** ou **Groupes de ressources** situés dans le volet de navigation de gauche.

Si vous utilisez un service gratuit, n'oubliez pas que vous êtes limité à trois index, indexeurs et sources de données. Vous pouvez supprimer des éléments un par un dans le portail pour ne pas dépasser la limite.

## Étapes suivantes

Maintenant que vous savez effectuer les tâches de base, vous pouvez aller plus loin avec les appels d'API REST supplémentaires pour des fonctionnalités plus avancées, telles que les indexeurs ou la [configuration d'un pipeline de recherche cognitive](#). Pour les étapes suivantes, nous vous recommandons le lien suivant :

[Tutoriel REST : Indexer et rechercher des données semi-structurées \(objets blob JSON\) dans la Recherche cognitive Azure](#)

# Démarrage rapide : Créer un index Recherche cognitive Azure dans PowerShell à l'aide des API REST

04/10/2020 • 16 minutes to read • [Edit Online](#)

Cet article décrit le processus de création, de chargement et d'interrogation d'un index Recherche cognitive Azure à l'aide de PowerShell et des [API REST de la Recherche cognitive Azure](#). Cet article explique comment exécuter des commandes PowerShell de manière interactive. Vous pouvez également [télécharger et exécuter un script PowerShell](#) qui effectue les mêmes opérations.

Si vous n'avez pas d'abonnement Azure, créez un [compte gratuit](#) avant de commencer.

## Prérequis

Les services et outils suivants sont indispensables dans ce guide de démarrage rapide.

- [PowerShell 5.1 ou version ultérieure](#) avec `Invoke-RestMethod` pour connaître les étapes séquentielles et interactives.
- [Créez un service Recherche cognitive Azure](#) ou [recherchez un service existant](#) dans votre abonnement actuel. Vous pouvez utiliser un service gratuit pour ce guide de démarrage rapide.

## Obtenir une clé et une URL

Les appels REST requièrent l'URL du service et une clé d'accès et ce, sur chaque demande. Un service de recherche est créé avec les deux. Ainsi, si vous avez ajouté la Recherche cognitive Azure à votre abonnement, effectuez ce qui suit pour obtenir les informations nécessaires :

1. [Connectez-vous au portail Azure](#), puis dans la page **Vue d'ensemble** du service de recherche, récupérez l'URL. Voici un exemple de point de terminaison : `https://mydemo.search.windows.net`.
2. Dans **Paramètres > Clés**, obtenez une clé d'administration pour avoir des droits d'accès complets sur le service. Il existe deux clés d'administration interchangeables, fournies pour assurer la continuité de l'activité au cas où vous deviez en remplacer une. Vous pouvez utiliser la clé primaire ou secondaire sur les demandes d'ajout, de modification et de suppression d'objets.

The screenshot shows two overlapping Azure portal windows. The top window is titled 'mydemo' and displays the 'Vue d'ensemble' (Overview) page. It includes a search bar, navigation links like 'Ajouter un index', 'Importer...', 'Explorateur...', 'Actualiser', 'Supprimer', and 'Déplacer'. A red box highlights the URL 'https://mydemo.search.windows.net'. The bottom window is titled 'mydemo - Clés' (Keys) and shows the 'Clés' (Keys) section. It has a sidebar with 'Vue d'ensemble', 'Journal d'activité', 'Contrôle d'accès (IAM)', 'Paramètres', 'Démarrage rapide', and 'Clés' (highlighted with a red box). The main area contains two fields: 'Clé d'administration primaire' containing '<placeholder-for-alphanumeric-autogenerated-string>' and 'Clé d'administration secondaire' containing another placeholder. Red circles labeled '1' and '2' point to the URL and the primary key field respectively.

Toutes les demandes nécessitent une clé API sur chaque demande envoyée à votre service. L'utilisation d'une clé valide permet d'établir, en fonction de chaque demande, une relation de confiance entre l'application qui envoie la demande et le service qui en assure le traitement.

## Se connecter à la Recherche cognitive Azure

1. Dans PowerShell, créez un objet `$headers` pour stocker le type de contenu et la clé API. Remplacez la clé API d'administration (YOUR-ADMIN-API-KEY) par une clé valide pour votre service de recherche. Vous définissez cet en-tête une seule fois pour toute la durée de la session, mais vous l'ajoutez à chaque requête.

```
$headers = @{
    'api-key' = '<YOUR-ADMIN-API-KEY>'
    'Content-Type' = 'application/json'
    'Accept' = 'application/json' }
```

2. Créez un objet `$url` qui spécifie la collection d'index du service. Remplacez le nom du service de recherche (YOUR-SEARCH-SERVICE-NAME) par un service de recherche valide.

```
$url = "https://<YOUR-SEARCH-SERVICE-NAME>.search.windows.net/indexes?api-version=2020-06-30&$select=name"
```

3. Exécutez `Invoke-RestMethod` pour envoyer une requête GET au service et vérifier la connexion. Ajoutez `ConvertTo-Json` afin d'afficher les réponses renvoyées par le service.

```
Invoke-RestMethod -Uri $url -Headers $headers | ConvertTo-Json
```

Si le service est vide et ne contient aucun index, les résultats se présenteront comme suit. Sinon, vous verrez une représentation JSON des définitions d'index.

```
{
    "@odata.context": "https://mydemo.search.windows.net/$metadata#indexes",
    "value": [
        ]
}
```

## 1 – Créer un index

Sauf si vous utilisez le portail, le service doit contenir un index pour vous permettre de charger des données. Cette étape définit l'index et l'envoie (push) au service. L'[API REST Create Index](#) est utilisée pour cette étape.

Un index doit contenir un nom et une collection de champs. La collection de champs définit la structure d'un *document*. Chaque champ a un nom, un type et des attributs qui déterminent la façon dont il est utilisé (par exemple, s'il permet d'effectuer une recherche en texte intégral, et s'il est filtrable ou récupérable dans les résultats de la recherche). Dans un index, l'un des champs de type `Edm.String` doit être désigné comme *clé* pour l'identité du document.

Cet index est nommé « hotels-quickstart » et contient les définitions de champ que vous voyez ci-dessous. Il s'agit d'un sous-ensemble d'un [index Hotels](#) plus grand et utilisé dans d'autres articles pas à pas. Par souci de concision, les définitions de champs ont été découpées dans ce démarrage rapide.

1. Collez cet exemple dans PowerShell pour créer un objet `$body` contenant le schéma d'index.

```
$body = @@
{
    "name": "hotels-quickstart",
    "fields": [
        {"name": "HotelId", "type": "Edm.String", "key": true, "filterable": true},
        {"name": "HotelName", "type": "Edm.String", "searchable": true, "filterable": false,
    "sortable": true, "facetable": false},
        {"name": "Description", "type": "Edm.String", "searchable": true, "filterable": false,
    "sortable": false, "facetable": false, "analyzer": "en.lucene"},
        {"name": "Category", "type": "Edm.String", "searchable": true, "filterable": true, "sortable": true,
    "facetable": true},
        {"name": "Tags", "type": "Collection(Edm.String)", "searchable": true, "filterable": true,
    "sortable": false, "facetable": true},
        {"name": "ParkingIncluded", "type": "Edm.Boolean", "filterable": true, "sortable": true,
    "facetable": true},
        {"name": "LastRenovationDate", "type": "Edm.DateTimeOffset", "filterable": true, "sortable": true,
    "facetable": true},
        {"name": "Rating", "type": "Edm.Double", "filterable": true, "sortable": true, "facetable": true},
        {"name": "Address", "type": "Edm.ComplexType",
            "fields": [
                {"name": "StreetAddress", "type": "Edm.String", "filterable": false, "sortable": false,
    "facetable": false, "searchable": true},
                {"name": "City", "type": "Edm.String", "searchable": true, "filterable": true, "sortable": true,
    "facetable": true},
                {"name": "StateProvince", "type": "Edm.String", "searchable": true, "filterable": true,
    "sortable": true, "facetable": true},
                {"name": "PostalCode", "type": "Edm.String", "searchable": true, "filterable": true,
    "sortable": true, "facetable": true},
                {"name": "Country", "type": "Edm.String", "searchable": true, "filterable": true, "sortable": true,
    "facetable": true}
            ]
        }
    ]
}
```

- Définissez l'URI vers la collection d'index de votre service et l'index *hotels-quickstart*.

```
$url = "https://<YOUR-SEARCH-SERVICE>.search.windows.net/indexes/hotels-quickstart?api-version=2020-06-30"
```

- Exécutez la commande avec **\$url**, **\$headers** et **\$body** pour créer l'index sur le service.

```
Invoke-RestMethod -Uri $url -Headers $headers -Method Put -Body $body | ConvertTo-Json
```

Les résultats doivent se présenter comme suit (tronqués au niveau des deux premiers champs par souci de concision) :

```
{
    "@odata.context": "https://mydemo.search.windows.net/$metadata#indexes/$entity",
    "@odata.etag": "\"0x8D6EDE28CFEABDA\"",
    "name": "hotels-quickstart",
    "defaultScoringProfile": null,
    "fields": [
        {
            "name": "HotelId",
            "type": "Edm.String",
            "searchable": true,
            "filterable": true,
            "retrievable": true,
            "sortable": true,
            "facetable": true,
            "key": true,
            "indexAnalyzer": null,
            "searchAnalyzer": null,
            "analyzer": null,
            "synonymMaps": ""
        },
        {
            "name": "HotelName",
            "type": "Edm.String",
            "searchable": true,
            "filterable": false,
            "retrievable": true,
            "sortable": true,
            "facetable": false,
            "key": false,
            "indexAnalyzer": null,
            "searchAnalyzer": null,
            "analyzer": null,
            "synonymMaps": ""
        },
        ...
    ]
}
```

**TIP**

À des fins de vérification, vous pouvez également consulter la liste des index dans le portail.

## 2 – Charger des documents

Pour envoyer des documents, utilisez une requête HTTP POST au point de terminaison de l'URL de votre index. Pour cette tâche, l'API REST est destinée à l'[ajout, la mise à jour ou la suppression de documents](#).

- Collez cet exemple dans PowerShell pour créer un objet **\$body** contenant les documents que vous souhaitez charger.

Cette requête comprend deux enregistrements intégraux et un enregistrement partiel. L'enregistrement partiel indique que vous pouvez charger des documents incomplets. Le paramètre `@search.action` spécifie la manière dont l'indexation est effectuée. Les valeurs valides incluent `upload`, `merge`, `mergeOrUpload` et `delete`. Le comportement `mergeOrUpload` crée un nouveau document pour `hotelId = 3` ou met à jour le contenu, le cas échéant.

```
$body = @"
{
    "value": [
        {
            "@search.action": "upload",
            "HotelId": "1",
            "HotelName": "Secret Point Motel",
            "Description": "The hotel is ideally located on the main commercial artery of the city in the heart of New York. A few minutes away is Time's Square and the historic centre of the city, as well as other places of interest that make New York one of America's most attractive and cosmopolitan cities.",
            "Category": "Boutique",
            "Tags": [ "pool", "air conditioning", "concierge" ],
            "ParkingIncluded": false,
            "LastRenovationDate": "1970-01-18T00:00:00Z",
            "Rating": 3.60,
            "Address":
            {
                "StreetAddress": "677 5th Ave",
                "City": "New York",
                "StateProvince": "NY",
                "PostalCode": "10022",
                "Country": "USA"
            }
        },
        {
            "@search.action": "upload",
            "HotelId": "2",
            "HotelName": "Twin Dome Motel",
            "Description": "The hotel is situated in a nineteenth century plaza, which has been expanded and renovated to the highest architectural standards to create a modern, functional and first-class hotel in which art and unique historical elements coexist with the most modern comforts.",
            "Category": "Boutique",
            "Tags": [ "pool", "free wifi", "concierge" ],
            "ParkingIncluded": false,
            "LastRenovationDate": "1979-02-18T00:00:00Z",
            "Rating": 3.60,
            "Address":
            {
                "StreetAddress": "140 University Town Center Dr",
                "City": "Sarasota",
                "StateProvince": "FL",
                "PostalCode": "34243",
                "Country": "USA"
            }
        },
        {
            "@search.action": "upload",
            "HotelId": "3",
            "HotelName": "Triple Landscape Hotel",
            "Description": "The Hotel stands out for its gastronomic excellence under the management of William Dough, who advises on and oversees all of the Hotel's restaurant services.",
            "Category": "Resort and Spa",
            "Tags": [ "air conditioning", "bar", "continental breakfast" ],
            "ParkingIncluded": true,
            "LastRenovationDate": "2015-09-20T00:00:00Z",
            "Rating": 4.80,
            "Address":
            {
                "StreetAddress": "3393 Peachtree Rd",
                "City": "Atlanta"
            }
        }
    ]
}
```

```

        "City": "Atlanta",
        "StateProvince": "GA",
        "PostalCode": "30326",
        "Country": "USA"
    }
},
{
"@search.action": "upload",
"HotelId": "4",
"HotelName": "Sublime Cliff Hotel",
"Description": "Sublime Cliff Hotel is located in the heart of the historic center of Sublime in an extremely vibrant and lively area within short walking distance to the sites and landmarks of the city and is surrounded by the extraordinary beauty of churches, buildings, shops and monuments. Sublime Cliff is part of a lovingly restored 1800 palace.",
"Category": "Boutique",
"Tags": [ "concierge", "view", "24-hour front desk service" ],
"ParkingIncluded": true,
"LastRenovationDate": "1960-02-06T00:00:00Z",
"Rating": 4.60,
"Address":
{
    "StreetAddress": "7400 San Pedro Ave",
    "City": "San Antonio",
    "StateProvince": "TX",
    "PostalCode": "78216",
    "Country": "USA"
}
}
]
}
"@
```

2. Définissez le point de terminaison sur la collection de documents *hotels-quickstart* et incluez l'opération d'indexation (`indexes/hotels-quickstart/docs/index`).

```
$url = "https://<YOUR-SEARCH-SERVICE>.search.windows.net/indexes/hotels-quickstart/docs/index?api-version=2020-06-30"
```

3. Exécutez la commande avec `$url`, `$headers` et `$body` pour charger des documents dans l'index *hotels-quickstart*.

```
Invoke-RestMethod -Uri $url -Headers $headers -Method Post -Body $body | ConvertTo-Json
```

Les résultats doivent ressembler à l'exemple suivant. Vous devez voir un [code d'état de 201](#).

```
{
    "@odata.context": "https://mydemo.search.windows.net/indexes(\u0027hotels-quickstart\u0027)/$metadata#Collection(Microsoft.Azure.Search.V2019_05_06.IndexResult)",
    "value": [
        {
            "key": "1",
            "status": true,
            "errorMessage": null,
            "statusCode": 201
        },
        {
            "key": "2",
            "status": true,
            "errorMessage": null,
            "statusCode": 201
        },
        {
            "key": "3",
            "status": true,
            "errorMessage": null,
            "statusCode": 201
        },
        {
            "key": "4",
            "status": true,
            "errorMessage": null,
            "statusCode": 201
        }
    ]
}
```

## 3 – Rechercher dans un index

Cette étape explique comment interroger un index à l'aide de [l'API Rechercher des documents](#).

Utilisez des guillemets simples pour la recherche \$urls. Les chaînes de requêtes contiennent des caractères \$ et vous n'êtes pas tenu de les placer en échappement si l'intégralité de la chaîne est placée entre guillemets simples.

1. Définissez le point de terminaison vers la collection de documents *hotels-quickstart* et ajoutez un paramètre **search** à transmettre dans une chaîne de requête.

Cette chaîne exécute une recherche vide (recherche=\*), renvoyant une liste non classée (résultat de la recherche = 1.0) de documents arbitraires. Par défaut, la Recherche cognitive Azure retourne 50 correspondances à la fois. Telle qu'elle est structurée, cette requête renvoie la structure et les valeurs d'un document entier. Ajoutez **\$count = true** pour obtenir le nombre de tous les documents dans les résultats.

```
$url = 'https://<YOUR-SEARCH-SERVICE>.search.windows.net/indexes/hotels-quickstart/docs?api-version=2020-06-30&search=*&$count=true'
```

2. Exécutez la commande pour envoyer **\$url** au service.

```
Invoke-RestMethod -Uri $url -Headers $headers | ConvertTo-Json
```

Les résultats se présentent comme suit.

```
{
  "@odata.context": "https://mydemo.search.windows.net/indexes(\u0027hotels-quickstart\u0027)/$metadata#docs(*)",
  "@odata.count": 4,
  "value": [
    {
      "@search.score": 0.1547872,
      "HotelId": "2",
      "HotelName": "Twin Dome Motel",
      "Description": "The hotel is situated in a nineteenth century plaza, which has been expanded and renovated to the highest architectural standards to create a modern, functional and first-class hotel in which art and unique historical elements coexist with the most modern comforts.",
      "Category": "Boutique",
      "Tags": "pool free wifi concierge",
      "ParkingIncluded": false,
      "LastRenovationDate": "1979-02-18T00:00:00Z",
      "Rating": 3.6,
      "Address": "{StreetAddress=140 University Town Center Dr; City=Sarasota; StateProvince=FL; PostalCode=34243; Country=USA}"
    },
    {
      "@search.score": 0.009068266,
      "HotelId": "3",
      "HotelName": "Triple Landscape Hotel",
      "Description": "The Hotel stands out for its gastronomic excellence under the management of William Dough, who advises on and oversees all of the Hotel's restaurant services.",
      "Category": "Resort and Spa",
      "Tags": "air conditioning bar continental breakfast",
      "ParkingIncluded": true,
      "LastRenovationDate": "2015-09-20T00:00:00Z",
      "Rating": 4.8,
      "Address": "{StreetAddress=3393 Peachtree Rd; City=Atlanta; StateProvince=GA; PostalCode=30326; Country=USA}"
    },
    ...
  ]
}
```

Essayez quelques autres exemples de requête pour avoir un aperçu de la syntaxe. Vous pouvez effectuer une recherche de chaîne, textualiser les requêtes \$filter, limiter le jeu de résultats, restreindre la recherche à des champs spécifiques, etc.

```
# Query example 1
# Search the entire index for the terms 'restaurant' and 'wifi'
# Return only the HotelName, Description, and Tags fields
$url = 'https://<YOUR-SEARCH-SERVICE>.search.windows.net/indexes/hotels-quickstart/docs?api-version=2020-06-30&search=restaurant wifi&$count=true&$select=HotelName,Description,Tags'

# Query example 2
# Apply a filter to the index to find hotels rated 4 or higher
# Returns the HotelName and Rating. Two documents match.
$url = 'https://<YOUR-SEARCH-SERVICE>.search.windows.net/indexes/hotels-quickstart/docs?api-version=2020-06-30&search=*==$filter=Rating gt 4&$select=HotelName,Rating'

# Query example 3
# Take the top two results, and show only HotelName and Category in the results
$url = 'https://<YOUR-SEARCH-SERVICE>.search.windows.net/indexes/hotels-quickstart/docs?api-version=2020-06-30&search=boutique&$top=2&$select=HotelName,Category'

# Query example 4
# Sort by a specific field (Address/City) in ascending order
$url = 'https://<YOUR-SEARCH-SERVICE>.search.windows.net/indexes/hotels-quickstart/docs?api-version=2020-06-30&search=pool&$orderby=Address/City asc&$select=HotelName, Address/City, Tags, Rating'
```

## Nettoyer les ressources

Lorsque vous travaillez dans votre propre abonnement, il est recommandé, à la fin de chaque projet, de déterminer si vous avez toujours besoin des ressources que vous avez créées. Les ressources laissées en cours d'exécution peuvent vous coûter de l'argent. Vous pouvez supprimer les ressources une par une, ou choisir de supprimer le groupe de ressources afin de supprimer l'ensemble des ressources.

Vous pouvez rechercher et gérer les ressources dans le portail à l'aide des liens **Toutes les ressources** ou **Groupes de ressources** situés dans le volet de navigation de gauche.

Si vous utilisez un service gratuit, n'oubliez pas que vous êtes limité à trois index, indexeurs et sources de données. Vous pouvez supprimer des éléments un par un dans le portail pour ne pas dépasser la limite.

## Étapes suivantes

Dans ce guide de démarrage rapide, vous avez utilisé PowerShell pour parcourir le workflow de base permettant de créer et d'accéder à du contenu dans la Recherche cognitive Azure. En prenant ces concepts en compte, nous vous recommandons de passer à des scénarios plus avancés, tels que l'indexation à partir de sources de données Azure ;

[Tutoriel REST : Indexer et rechercher des données semi-structurées \(objets blob JSON\) dans la Recherche cognitive Azure](#)

# Démarrage rapide : Créer un index Recherche cognitive Azure en Python à l'aide de notebooks Jupyter

04/10/2020 • 16 minutes to read • [Edit Online](#)

Générez un notebook Jupyter qui crée, charge et interroge un index Recherche cognitive Azure en Python à l'aide des [API REST de la Recherche cognitive Azure](#). Cet article décrit la procédure à suivre pour créer un notebook. Vous pouvez [également télécharger et exécuter un notebook Jupyter Python complet](#).

Si vous n'avez pas d'abonnement Azure, créez un [compte gratuit](#) avant de commencer.

## Prérequis

Les services et outils suivants sont indispensables dans ce guide de démarrage rapide.

- [Anaconda 3.x](#), qui fournit des notebooks Python 3.x et Jupyter.
- [Créez un service Recherche cognitive Azure](#) ou [recherchez un service existant](#) dans votre abonnement actuel. Vous pouvez utiliser le niveau gratuit pour ce démarrage rapide.

## Obtenir une clé et une URL

Les appels REST requièrent l'URL du service et une clé d'accès et ce, sur chaque demande. Un service de recherche est créé avec les deux. Ainsi, si vous avez ajouté la Recherche cognitive Azure à votre abonnement, effectuez ce qui suit pour obtenir les informations nécessaires :

1. [Connectez-vous au portail Azure](#), puis dans la page **Vue d'ensemble** du service de recherche, récupérez l'URL. Voici un exemple de point de terminaison : `https://mydemo.search.windows.net`.
2. Dans **Paramètres > Clés**, obtenez une clé d'administration pour avoir des droits d'accès complets sur le service. Il existe deux clés d'administration interchangeables, fournies pour assurer la continuité de l'activité au cas où vous deviez en remplacer une. Vous pouvez utiliser la clé primaire ou secondaire sur les demandes d'ajout, de modification et de suppression d'objets.

The screenshot shows two overlapping Azure portal windows. The top window is titled 'mydemo' and displays the 'Vue d'ensemble' (Overview) page. It includes a search bar, navigation links like 'Ajouter un index', 'Importer...', 'Explorateur...', 'Actualiser', 'Supprimer', and 'Déplacer'. A red box highlights the URL 'https://mydemo.search.windows.net'. The bottom window is titled 'mydemo - Clés' (Keys) and shows the 'Clés' (Keys) section. It has a sidebar with 'Vue d'ensemble', 'Journal d'activité', 'Contrôle d'accès (IAM)', 'Paramètres', 'Démarrage rapide', 'Clés' (highlighted with a red box), and 'Échelle'. The main area shows two fields: 'Clé d'administration primaire' containing '<placeholder-for-alphanumeric-autogenerated-string>' and 'Clé d'administration secondaire' also containing '<placeholder-for-alphanumeric-autogenerated-string>'. Red circles labeled '1' and '2' point to the URL in the top window and the primary key field in the bottom window respectively.

Toutes les demandes nécessitent une clé API sur chaque demande envoyée à votre service. L'utilisation d'une clé valide permet d'établir, en fonction de chaque demande, une relation de confiance entre l'application qui envoie la demande et le service qui en assure le traitement.

## Se connecter à la Recherche cognitive Azure

Dans cette tâche, démarrez un notebook Jupyter, puis vérifiez que vous pouvez vous connecter à la Recherche cognitive Azure. Pour ce faire, demandez une liste d'index de votre service. Sur Windows avec Anaconda3, vous pouvez utiliser Anaconda Navigator pour lancer un notebook.

1. Créez un notebook Python3.
2. Dans la première cellule, chargez les bibliothèques utilisées pour travailler avec JSON et formuler des requêtes HTTP.

```
import json
import requests
from pprint import pprint
```

3. Dans la deuxième cellule, entrez les éléments de requête qui seront des constantes sur chaque requête. Remplacez le nom du service de recherche (YOUR-SEARCH-SERVICE-NAME) et la clé API d'administration (YOUR-ADMIN-API-KEY) par des valeurs valides.

```
endpoint = 'https://<YOUR-SEARCH-SERVICE-NAME>.search.windows.net/'
api_version = '?api-version=2020-06-30'
headers = {'Content-Type': 'application/json',
           'api-key': '<YOUR-ADMIN-API-KEY>'}
```

Si vous recevez `ConnectionError "Failed to establish a new connection"`, vérifiez que la clé API est une clé d'administration principale ou secondaire, et que tous les caractères de début et de fin (`?` et `/`) sont en place.

4. Dans la troisième cellule, formulez la requête. Cette requête GET cible la collection d'index de votre service de recherche et sélectionne la propriété de nom des index existants.

```

url = endpoint + "indexes" + api_version + "&$select=name"
response = requests.get(url, headers=headers)
index_list = response.json()
pprint(index_list)

```

5. Exécutez chaque étape. Si des index existent, la réponse contient la liste des noms d'index. Dans la capture d'écran ci-dessous, le service a déjà un index azureblob et un index realestate-us-sample.

```

In [1]: import json
import requests
from pprint import pprint

In [2]: endpoint = 'https://mydemo.search.windows.net/'
api_version = '?api-version=2019-05-06'
headers = {'Content-Type': 'application/json',
           'api-key': 'AA11BB22CC33DD44FF55EE66GG77HH88II99' }

In [3]: url = endpoint + "indexes" + api_version + "&$select=name"
response = requests.get(url, headers=headers)
index_list = response.json()
pprint(index_list)

{'@odata.context': 'https://mydemo.search.windows.net/$metadata#indexes(name)',
 'value': [{'name': 'azureblob-index'}, {'name': 'realestate-us-sample'}]}

```

En revanche, une collection d'index vide retourne cette réponse :

```
{'@odata.context': 'https://mydemo.search.windows.net/$metadata#indexes(name)', 'value': []}
```

## 1 – Créer un index

Sauf si vous utilisez le portail, le service doit contenir un index pour vous permettre de charger des données. Cette étape utilise l'[API REST Créer un index](#) pour envoyer un schéma d'index au service.

Les éléments requis d'un index sont un nom, une collection de champs et une clé. La collection de champs définit la structure d'un *document*. Chaque champ a un nom, un type et des attributs qui déterminent la façon dont il est utilisé (par exemple, s'il permet d'effectuer une recherche en texte intégral, et s'il est filtrable ou récupérable dans des résultats de recherche). Dans un index, l'un des champs de type `Edm.String` doit être désigné comme *clé* pour l'identité du document.

Cet index est nommé « hotels-quickstart » et contient les définitions de champ que vous voyez ci-dessous. Il s'agit d'un sous-ensemble d'un [index Hotels](#) plus grand utilisé dans d'autres procédures pas à pas. Nous l'avons volontairement tronqué dans ce démarrage rapide par souci de concision.

1. Dans la cellule suivante, collez l'exemple suivant dans une cellule pour fournir le schéma.

```

index_schema = {
    "name": "hotels-quickstart",
    "fields": [
        {"name": "HotelId", "type": "Edm.String", "key": "true", "filterable": "true"}, 
        {"name": "HotelName", "type": "Edm.String", "searchable": "true", "filterable": "false", 
        "sortable": "true", "facetable": "false"}, 
        {"name": "Description", "type": "Edm.String", "searchable": "true", "filterable": "false", 
        "sortable": "false", "facetable": "false", "analyzer": "en.lucene"}, 
        {"name": "Description_fr", "type": "Edm.String", "searchable": "true", "filterable": "false", 
        "sortable": "false", "facetable": "false", "analyzer": "fr.lucene"}, 
        {"name": "Category", "type": "Edm.String", "searchable": "true", "filterable": "true", "sortable": 
        "true", "facetable": "true"}, 
        {"name": "Tags", "type": "Collection(Edm.String)", "searchable": "true", "filterable": "true", 
        "sortable": "false", "facetable": "true"}, 
        {"name": "ParkingIncluded", "type": "Edm.Boolean", "filterable": "true", "sortable": "true", 
        "facetable": "true"}, 
        {"name": "LastRenovationDate", "type": "Edm.DateTimeOffset", "filterable": "true", "sortable": 
        "true", "facetable": "true"}, 
        {"name": "Rating", "type": "Edm.Double", "filterable": "true", "sortable": "true", "facetable": 
        "true"}, 
        {"name": "Address", "type": "Edm.ComplexType", 
        "fields": [
            {"name": "StreetAddress", "type": "Edm.String", "filterable": "false", "sortable": "false", 
            "facetable": "false", "searchable": "true"}, 
            {"name": "City", "type": "Edm.String", "searchable": "true", "filterable": "true", "sortable": 
            "true", "facetable": "true"}, 
            {"name": "StateProvince", "type": "Edm.String", "searchable": "true", "filterable": "true", 
            "sortable": "true", "facetable": "true"}, 
            {"name": "PostalCode", "type": "Edm.String", "searchable": "true", "filterable": "true", 
            "sortable": "true", "facetable": "true"}, 
            {"name": "Country", "type": "Edm.String", "searchable": "true", "filterable": "true", "sortable": 
            "true", "facetable": "true"} 
        ] 
    ] 
}

```

2. Dans une autre cellule, formulez la requête. Cette requête POST cible la collection d'index de votre service de recherche et crée un index basé sur le schéma d'index que vous avez fourni dans la cellule précédente.

```

url = endpoint + "indexes" + api_version
response = requests.post(url, headers=headers, json=index_schema)
index = response.json()
pprint(index)

```

3. Exécutez chaque étape.

La réponse inclut la représentation JSON du schéma. La capture d'écran suivante présente uniquement une partie de la réponse.

```
In [4]: M index_schema = {
    "name": "hotels-quickstart",
    "fields": [
        {"name": "HotelId", "type": "Edm.String", "key": "true", "filterable": "true"}, 
        {"name": "HotelName", "type": "Edm.String", "searchable": "true", "filterable": "false", "sortable": "true", "facetable": "true"}, 
        {"name": "Description", "type": "Edm.String", "searchable": "true", "filterable": "false", "sortable": "false", "facetable": "true"}, 
        {"name": "Description_fr", "type": "Edm.String", "searchable": "true", "filterable": "false", "sortable": "false", "facetable": "true"}, 
        {"name": "Category", "type": "Edm.String", "searchable": "true", "filterable": "true", "sortable": "true", "facetable": "true"}, 
        {"name": "Tags", "type": "Collection(Edm.String)", "searchable": "true", "filterable": "true", "sortable": "false", "facetable": "false"}, 
        {"name": "ParkingIncluded", "type": "Edm.Boolean", "filterable": "true", "sortable": "true", "facetable": "true"}, 
        {"name": "LastRenovationDate", "type": "Edm.DateTimeOffset", "filterable": "true", "sortable": "true", "facetable": "true"}, 
        {"name": "Rating", "type": "Edm.Double", "filterable": "true", "sortable": "true", "facetable": "true"}, 
        {"name": "Address", "type": "Edm.ComplexType", "fields": [
            {"name": "StreetAddress", "type": "Edm.String", "filterable": "false", "sortable": "false", "facetable": "false", "searchable": "true"}, 
            {"name": "City", "type": "Edm.String", "searchable": "true", "filterable": "true", "sortable": "true", "facetable": "true"}, 
            {"name": "StateProvince", "type": "Edm.String", "searchable": "true", "filterable": "true", "sortable": "true", "facetable": "true"}, 
            {"name": "PostalCode", "type": "Edm.String", "searchable": "true", "filterable": "true", "sortable": "true", "facetable": "true"}, 
            {"name": "Country", "type": "Edm.String", "searchable": "true", "filterable": "true", "sortable": "true", "facetable": "true"}]}]
```

```
In [5]: M url = endpoint + "indexes" + api_version
response = requests.post(url, headers=headers, json=index_schema)
index = response.json()
pprint(index)
```

```
{'@odata.context': 'https://mydemo.search.windows.net/$metadata#indexes$/entity',
 '@odata.etag': '"0x8D6EDC5DE036D1A"',
 'analyzers': [],
 'charFilters': [],
 'corsOptions': None,
 'defaultScoringProfile': None,
 'fields': [{"analyzer': None,
   'facetable': True,
   'filterable': True,
   'indexAnalyzer': None,
   'key': True,
   'name': 'HotelId',
   'retrievable': True,
   'searchAnalyzer': None,
   'searchable': True,
   'sortable': True}]}]
```

### TIP

Une autre manière de vérifier la création d'index consiste à consulter la liste Index dans le portail.

## 2 – Charger des documents

Pour envoyer (push) des documents, utilisez une requête HTTP POST au point de terminaison de l'URL de votre index. L'API REST est destinée à l'[ajout, la mise à jour ou la suppression de documents](#). Les documents proviennent de [HotelsData](#) sur GitHub.

- Dans une nouvelle cellule, fournissez quatre documents conformes au schéma d'index. Spécifiez une action de chargement pour chaque document.

```
documents = {
    "value": [
        {
            "@search.action": "upload",
            "HotelId": "1",
            "HotelName": "Secret Point Motel",
            "Description": "The hotel is ideally located on the main commercial artery of the city in the heart of New York. A few minutes away is Time's Square and the historic centre of the city, as well as other places of interest that make New York one of America's most attractive and cosmopolitan cities.",
            "Description_fr": "L'hôtel est idéalement situé sur la principale artère commerciale de la ville en plein cœur de New York. A quelques minutes se trouve la place du temps et le centre historique de la ville, ainsi que d'autres lieux d'intérêt qui font de New York l'une des villes les plus attractives et cosmopolites de l'Amérique.",
            "Category": "Boutique",
            "Tags": [ "pool", "air conditioning", "concierge" ],
            "ParkingIncluded": "false",
            "LastRenovationDate": "1970-01-18T00:00:00Z",
            "Rating": 3.60,
            "Address": {
                "StreetAddress": "677 5th Ave",
                "City": "New York",
                "StateProvince": "NY",
                "PostalCode": "10001"
            }
        }
    ]
}
```

```
"PostalCode": "10022",
"Country": "USA"
},
{
"@search.action": "upload",
"HotelId": "2",
"HotelName": "Twin Dome Motel",
"Description": "The hotel is situated in a nineteenth century plaza, which has been expanded and renovated to the highest architectural standards to create a modern, functional and first-class hotel in which art and unique historical elements coexist with the most modern comforts.",
"Description_fr": "L'hôtel est situé dans une place du XIXe siècle, qui a été agrandie et rénovée aux plus hautes normes architecturales pour créer un hôtel moderne, fonctionnel et de première classe dans lequel l'art et les éléments historiques uniques coexistent avec le confort le plus moderne.",
"Category": "Boutique",
"Tags": [ "pool", "free wifi", "concierge" ],
"ParkingIncluded": "false",
"LastRenovationDate": "1979-02-18T00:00:00Z",
"Rating": 3.60,
"Address": {
    "StreetAddress": "140 University Town Center Dr",
    "City": "Sarasota",
    "StateProvince": "FL",
    "PostalCode": "34243",
    "Country": "USA"
},
{
"@search.action": "upload",
"HotelId": "3",
"HotelName": "Triple Landscape Hotel",
"Description": "The Hotel stands out for its gastronomic excellence under the management of William Dough, who advises on and oversees all of the Hotel's restaurant services.",
"Description_fr": "L'hôtel est situé dans une place du XIXe siècle, qui a été agrandie et rénovée aux plus hautes normes architecturales pour créer un hôtel moderne, fonctionnel et de première classe dans lequel l'art et les éléments historiques uniques coexistent avec le confort le plus moderne.",
"Category": "Resort and Spa",
"Tags": [ "air conditioning", "bar", "continental breakfast" ],
"ParkingIncluded": "true",
"LastRenovationDate": "2015-09-20T00:00:00Z",
"Rating": 4.80,
"Address": {
    "StreetAddress": "3393 Peachtree Rd",
    "City": "Atlanta",
    "StateProvince": "GA",
    "PostalCode": "30326",
    "Country": "USA"
},
{
"@search.action": "upload",
"HotelId": "4",
"HotelName": "Sublime Cliff Hotel",
"Description": "Sublime Cliff Hotel is located in the heart of the historic center of Sublime in an extremely vibrant and lively area within short walking distance to the sites and landmarks of the city and is surrounded by the extraordinary beauty of churches, buildings, shops and monuments. Sublime Cliff is part of a lovingly restored 1800 palace.",
"Description_fr": "Le sublime Cliff Hotel est situé au cœur du centre historique de sublime dans un quartier extrêmement animé et vivant, à courte distance de marche des sites et monuments de la ville et est entouré par l'extraordinaire beauté des églises, des bâtiments, des commerces et Monuments. Sublime Cliff fait partie d'un Palace 1800 restauré avec amour.",
"Category": "Boutique",
"Tags": [ "concierge", "view", "24-hour front desk service" ],
"ParkingIncluded": "true",
"LastRenovationDate": "1960-02-06T00:00:00Z",
"Rating": 4.60,
"Address": {
    "StreetAddress": "7400 San Pedro Ave",
    "City": "San Antonio",
```

```

        "StateProvince": "TX",
        "PostalCode": "78216",
        "Country": "USA"
    }
}
]
}

```

- Dans une autre cellule, formulez la requête. Cette requête POST cible la collection de documents de l'index de démarrage rapide hotels, et envoie les documents fournis à l'étape précédente.

```

url = endpoint + "indexes/hotels-quickstart/docs/index" + api_version
response = requests.post(url, headers=headers, json=documents)
index_content = response.json()
pprint(index_content)

```

- Exécutez chaque étape pour envoyer les documents à un index dans votre service de recherche. Les résultats doivent ressembler à l'exemple suivant.

```

In [6]: url = endpoint + "indexes/hotels-quickstart/docs/index" + api_version
response = requests.post(url, headers=headers, json=documents)
index_content = response.json()
pprint(index_content)

{'@odata.context': "https://mydemo.search.windows.net/indexes('hotels-quickstart')/$metadata#Collection(Microsoft.Azure.Sea
rch.V2019_05_06.IndexResult)",
'value': [{"errorMessage": None,
'key': '1',
'status': True,
'statusCode': 201},
{'errorMessage': None,
'key': '2',
'status': True,
'statusCode': 201},
{'errorMessage': None,
'key': '3',
'status': True,
'statusCode': 201},
{'errorMessage': None,
'key': '4',
'status': True,
'statusCode': 201}]}

```

## 3 – Rechercher dans un index

Cette étape vous montre comment interroger un index à l'aide de l'[API REST Rechercher des documents](#).

- Dans une cellule, fournir une expression de requête qui exécute une recherche vide (recherche=\*), retournant une liste non classée (résultat de la recherche=1.0) de documents arbitraires. Par défaut, la Recherche cognitive Azure retourne 50 correspondances à la fois. Telle qu'elle est structurée, cette requête renvoie la structure et les valeurs d'un document entier. Ajoutez \$count=true pour obtenir le nombre de tous les documents dans les résultats.

```

searchstring = '&search=*&$count=true'

url = endpoint + "indexes/hotels-quickstart/docs" + api_version + searchstring
response = requests.get(url, headers=headers, json=searchstring)
query = response.json()
pprint(query)

```

- Dans une nouvelle cellule, fournissez l'exemple suivant pour rechercher les termes « hotels » et « wifi ». Ajoutez \$select pour spécifier les champs à inclure dans les résultats de recherche.

```

searchstring = '&search=hotels wifi&$count=true&$select=HotelId,HotelName'

url = endpoint + "indexes/hotels-quickstart/docs" + api_version + searchstring
response = requests.get(url, headers=headers, json=searchstring)
query = response.json()
pprint(query)

```

Les résultats se présentent comme suit.

```

In [11]: M searchstring = '&search=hotels wifi&$count=true&$select=HotelId,HotelName'

In [12]: M url = endpoint + "indexes/hotels-quickstart/docs" + api_version + searchstring
          response = requests.get(url, headers=headers, json=searchstring)
          query = response.json()
          pprint(query)

{'@odata.context': "https://mydemo.search.windows.net/indexes('hotels-quickstart')/$metadata#docs(*)",
 '@odata.count': 4,
 'value': [{"@search.score": 0.16548625,
            'HotelId': '2',
            'HotelName': 'Twin Dome Motel'},
           {"@search.score": 0.016361875,
            'HotelId': '3',
            'HotelName': 'Triple Landscape Hotel'},
           {"@search.score": 0.008899688,
            'HotelId': '1',
            'HotelName': 'Secret Point Motel'},
           {"@search.score": 0.008899688,
            'HotelId': '4',
            'HotelName': 'Sublime Cliff Hotel'}]}

```

3. Ensuite, appliquez une expression \$filter qui sélectionne uniquement les hôtels dont l'évaluation est supérieure à 4.

```

searchstring = '&search=*&$filter=Rating gt 4&$select=HotelId,HotelName,Description,Rating'

url = endpoint + "indexes/hotels-quickstart/docs" + api_version + searchstring
response = requests.get(url, headers=headers, json=searchstring)
query = response.json()
pprint(query)

```

4. Par défaut, le moteur de recherche retourne les 50 premiers documents. Cependant, vous pouvez utiliser top et skip pour ajouter la pagination et choisir le nombre de documents retournés pour chaque résultat. Cette requête retourne deux documents dans chaque jeu de résultats.

```

searchstring = '&search=boutique&$top=2&$select=HotelId,HotelName,Description'

url = endpoint + "indexes/hotels-quickstart/docs" + api_version + searchstring
response = requests.get(url, headers=headers, json=searchstring)
query = response.json()
pprint(query)

```

5. Dans ce dernier exemple, utilisez \$orderby pour trier les résultats par ville. Cet exemple comprend les champs de la collection Address.

```

searchstring = '&search=pool&$orderby=Address/City&$select=HotelId, HotelName, Address/City,
Address/StateProvince'

url = endpoint + "indexes/hotels-quickstart/docs" + api_version + searchstring
response = requests.get(url, headers=headers, json=searchstring)
query = response.json()
pprint(query)

```

## Nettoyer

Lorsque vous travaillez dans votre propre abonnement, il est recommandé, à la fin de chaque projet, de

déterminer si vous avez toujours besoin des ressources que vous avez créées. Les ressources laissées en cours d'exécution peuvent vous coûter de l'argent. Vous pouvez supprimer les ressources une par une, ou choisir de supprimer le groupe de ressources afin de supprimer l'ensemble des ressources.

Vous pouvez rechercher et gérer les ressources dans le portail à l'aide des liens **Toutes les ressources** ou **Groupes de ressources** situés dans le volet de navigation de gauche.

Si vous utilisez un service gratuit, n'oubliez pas que vous êtes limité à trois index, indexeurs et sources de données. Vous pouvez supprimer des éléments un par un dans le portail pour ne pas dépasser la limite.

## Étapes suivantes

Par souci de simplification, ce démarrage rapide utilise une version abrégée de l'index Hotels. Vous pouvez créer la version complète pour essayer des requêtes plus intéressantes. Pour obtenir la version complète et les 50 documents, exécutez l'Assistant **Importation de données** en sélectionnant *hotels-sample* à partir des exemples de sources de données intégrés.

[Démarrage rapide : Créer un index dans le portail Azure](#)

# Tutoriel : Créer votre première application de recherche à l'aide du SDK .NET

04/10/2020 • 25 minutes to read • [Edit Online](#)

Découvrez comment créer une interface web pour interroger un index et présenter les résultats de la recherche en utilisant Recherche cognitive Azure. Ce tutoriel s'appuyant sur un index hébergé existant, vous pouvez vous concentrer sur la création d'une page de recherche. L'index contient des données d'hôtels fictives. Une fois que vous avez une page de base, vous pourrez l'améliorer dans les leçons suivantes pour inclure une pagination, des facettes et une expérience d'auto-complétion.

Dans ce tutoriel, vous allez apprendre à :

- Configurer un environnement de développement
- Modéliser des structures de données
- Créer une page web
- Définir des méthodes
- Test de l'application

Vous allez également découvrir à quel point il est simple d'effectuer un appel de recherche. Les instructions essentielles du code que vous allez développer sont encapsulées dans les quelques lignes suivantes.

```
var parameters = new SearchParameters
{
    // Enter Hotel property names into this list, so only these values will be returned.
    Select = new[] { "HotelName", "Description" }
};

DocumentSearchResult<Hotel> results = await _indexClient.Documents.SearchAsync<Hotel>("search text",
parameters);
```

Cet appel lance une recherche de données Azure et retourne les résultats.



16 Results

## Travel Resort

The Best Gaming Resort in the area. With elegant rooms & suites, pool, cabanas, spa, brewery & world-class gaming. This is the best place to play, stay & dine.

## Nova Hotel & Spa

1 Mile from the airport. Free WiFi, Outdoor Pool, Complimentary Airport Shuttle, 6 miles from the beach & 10 miles from downtown.

## King's Palace Hotel

Newest kid on the downtown block. Steps away from the most popular destinations in downtown, enjoy free WiFi, an indoor rooftop pool & fitness center, 24 Grab'n'Go & drinks at the bar

## Prérequis

Pour suivre ce didacticiel, vous devez effectuer les opérations suivantes :

[Installez Visual Studio](#) à utiliser comme IDE.

### Installer et exécuter le projet à partir de GitHub

1. Recherchez l'exemple sur GitHub : [Créer votre première application](#).
2. Sélectionnez **Clone or download** (Cloner ou télécharger) et créez votre copie privée locale du projet.
3. À l'aide de Visual Studio, accédez à la solution pour la page de recherche de base et ouvrez-la, puis sélectionnez **Démarrer sans débogage** (ou appuyez sur F5).
4. Tapez des mots (par exemple « wifi », « view », « bar », « parking ») et examiner les résultats !



19 Results

### Super Deluxe Inn & Suites

Complimentary Airport Shuttle & WiFi. Book Now and save - Spacious All Suite Hotel, Indoor/Outdoor Pool, Fitness Center, Florida Green certified, Starbucks Coffee, HDTV

### Double Sanctuary Resort

5\* Luxury Hotel - Biggest Rooms in the city. #1 Hotel in the area listed by Conde Nast Traveler. Free WiFi, Flexible check in/out, Fitness Center & Nespresso in room.

### Veteran Right Track

Free Shuttle to the airport and casinos. Free breakfast and WiFi.

### Regal Orb Resort & Spa

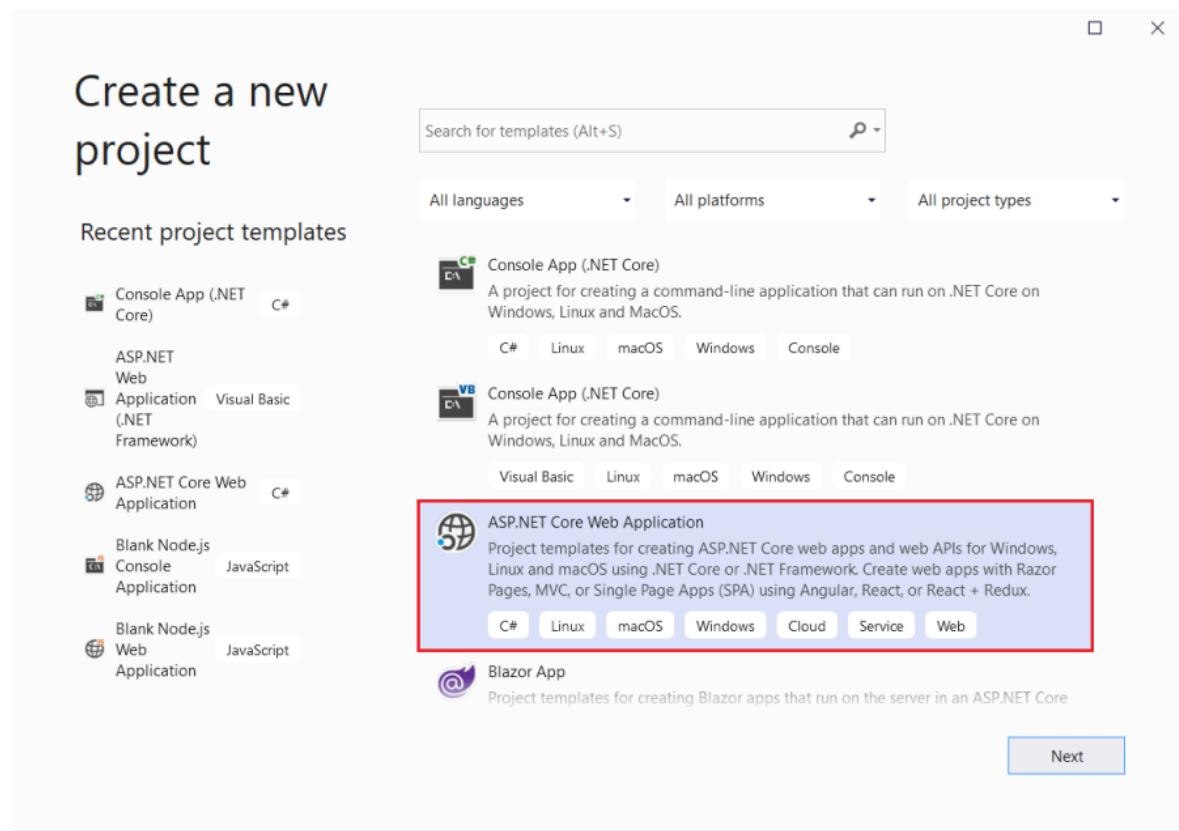
Your home away from home. Brand new fully equipped premium rooms, fast WiFi, full kitchen, washer & dryer, fitness center

Nous espérons que ce projet se déroule normalement et que l'application Azure est opérationnelle. De nombreux composants essentiels pour des recherches plus sophistiquées étant inclus dans cette application, il est judicieux de la suivre et de la recréer étape par étape.

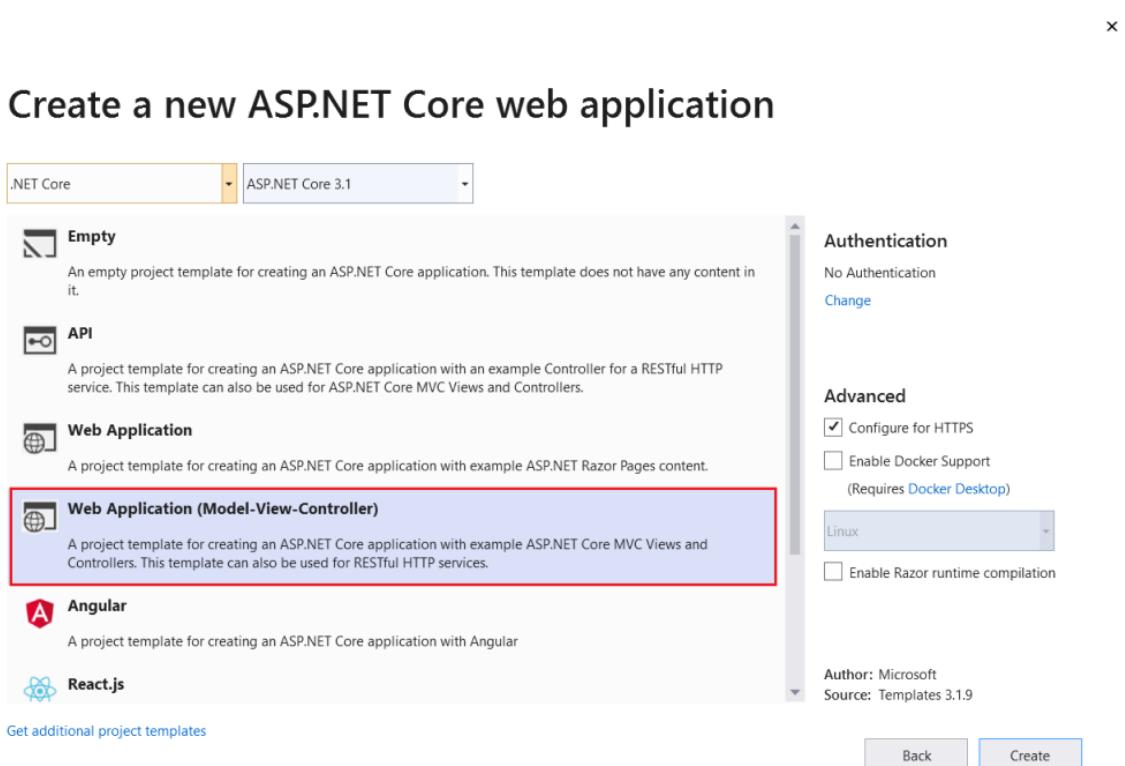
Pour créer ce projet à partir de zéro et ainsi mieux connaître les composants de Recherche cognitive Azure, effectuez les étapes suivantes.

## Configurer un environnement de développement

1. Dans Visual Studio 2017 ou version ultérieure, sélectionnez **Nouveau/Projet** puis **Application web ASP.NET Core**. Attribuez un nom au projet, par exemple « FirstAzureSearchApp ».



2. Une fois que vous avez cliqué sur OK pour ce type de projet, un second ensemble d'options s'appliquant à ce projet vous est présenté. Sélectionnez **Application web (Model-View-Controller)**.



3. Ensuite, dans le menu Outils, sélectionnez Gestionnaire de package NuGet, puis Gérer les packages NuGet pour la solution... . Nous devons installer un seul package. Sélectionnez l'onglet Parcourir puis, dans la zone de recherche, tapez « Recherche cognitive Azure ». Installez Microsoft.Azure.Search quand il apparaît dans la liste (version 9.0.1 ou ultérieure). Vous devrez valider quelques boîtes de dialogue supplémentaires pour terminer l'installation.

Azure.Search.Documents by Microsoft, v11.1.1  
This is the Azure Cognitive Search client library for developing .NET

Microsoft.Azure.Search.Service by Microsoft, v10.1.0  
Use this package if you're developing automation in .NET to manage Azure Cognitive Search index...

Microsoft.Azure.Search.Data by Microsoft, v10.1.0  
Use this package if you're developing a .NET application using Azure Cognitive Search, and y...

Citolab.Azure.BlobStorage.Document by Microsoft, v2020.1.29.2  
Helper to upload documents to Azure blob storage. Makes it easy to do full text search...

Citolab.Azure.BlobStorage.Document by Microsoft, v2020.1.29.2  
Helper to upload documents to Azure blob storage. Makes it easy to do full text search...

Microsoft.Azure.Management.Search.F by Microsoft, v1.34.0  
Provides search service accounts management

**Azure.Search.Documents** by Microsoft, v11.1.1  
nuget.org

Versions - 0

Project	Version
FirstSearchApp	

Installed: not installed      Uninstall

Version: Latest stable 11.1.1      **Install**

Options

Description

## Initialiser Recherche cognitive Azure

Pour cet exemple, nous utilisons des données d'hôtels disponibles publiquement. Ces données sont une collection arbitraire de 50 noms d'hôtels fictifs et de descriptions, créée uniquement à des fins de démonstration. Pour accéder à ces données, vous devez spécifier un nom et la clé correspondante.

- Ouvrez le fichier appsettings.json dans votre nouveau projet et remplacez les lignes par défaut par les nom et clé suivants. La clé d'API indiquée ici n'est pas un exemple de clé ; c'est *exactement* la clé dont vous avez besoin pour accéder aux données d'hôtels. Votre fichier appsettings.json doit maintenant ressembler à ceci.

```
{
  "SearchServiceName": "azs-playground",
  "SearchServiceQueryApiKey": "EA4510A6219E14888741FCFC19BFBB82"
}
```

- Nous n'en avons pas encore terminé avec ce fichier ; accédez à ses propriétés et définissez le paramètre Copier dans le répertoire de sortie sur Copier si plus récent.

Properties

appsettings.json File Properties

Advanced

Build Action	Content
Copy to Output Director	<b>Copy if newer</b>
Custom Tool	
Custom Tool Namespace	

Misc

File Name	appsettings.json
-----------	------------------

## Modéliser des structures de données

Les modèles (classes C#) permettent de communiquer des données entre le client (vue), le serveur (contrôleur) et également le cloud Azure à l'aide de l'architecture MVC (modèle, vue, contrôleur). En règle générale, ces modèles reflètent la structure des données sollicitées. De plus, nous avons besoin d'un modèle pour gérer les communications entre la vue et le contrôleur.

1. Ouvrez le dossier **Modèles** de votre projet à l'aide de l'Explorateur de solutions ; vous y verrez un modèle par défaut : **ErrorViewModel.cs**.

2. Cliquez avec le bouton droit sur le dossier **Modèles** et sélectionnez **Ajouter**, puis **Nouvel élément**.

Ensuite, dans la boîte de dialogue qui s'affiche, sélectionnez **ASP.NET Core**, puis la première option **Classe**. Renommez le fichier .cs en Hotel.cs, puis cliquez sur **Ajouter**. Remplacez tout le contenu de Hotel.cs par le code suivant. Notez les membres **Address** et **Room** de la classe ; ces champs étant eux-mêmes des classes, nous aurons également besoin de modèles pour eux.

```
using System;
using Microsoft.Azure.Search;
using Microsoft.Azure.Search.Models;
using Microsoft.Spatial;
using Newtonsoft.Json;

namespace FirstAzureSearchApp.Models
{
    public partial class Hotel
    {
        [System.ComponentModel.DataAnnotations.Key]
        [IsFilterable]
        public string HotelId { get; set; }

        [IsSearchable, IsSortable]
        public string HotelName { get; set; }

        [IsSearchable]
        [Analyzer(AnalyzerName.AsString.EnLucene)]
        public string Description { get; set; }

        [IsSearchable]
        [Analyzer(AnalyzerName.AsString.FrLucene)]
        [JsonProperty("Description_fr")]
        public string DescriptionFr { get; set; }

        [IsSearchable, IsFilterable, IsSortable, IsFacetable]
        public string Category { get; set; }

        [IsSearchable, IsFilterable, IsFacetable]
        public string[] Tags { get; set; }

        [IsFilterable, IsSortable, IsFacetable]
        public bool? ParkingIncluded { get; set; }

        [IsFilterable, IsSortable, IsFacetable]
        public DateTimeOffset? LastRenovationDate { get; set; }

        [IsFilterable, IsSortable, IsFacetable]
        public double? Rating { get; set; }

        public Address Address { get; set; }

        [IsFilterable, IsSortable]
        public GeographyPoint Location { get; set; }

        public Room[] Rooms { get; set; }
    }
}
```

3. Suivez la même procédure de création de modèle pour la classe **Address**, mais nommez le fichier Address.cs. Remplacez le contenu par ce qui suit.

```
using Microsoft.Azure.Search;

namespace FirstAzureSearchApp.Models
{
    public partial class Address
    {
        [IsSearchable]
        public string StreetAddress { get; set; }

        [IsSearchable, IsFilterable, IsSortable, IsFacetable]
        public string City { get; set; }

        [IsSearchable, IsFilterable, IsSortable, IsFacetable]
        public string StateProvince { get; set; }

        [IsSearchable, IsFilterable, IsSortable, IsFacetable]
        public string PostalCode { get; set; }

        [IsSearchable, IsFilterable, IsSortable, IsFacetable]
        public string Country { get; set; }
    }
}
```

4. Là encore, suivez le même processus pour créer la classe **Room**, mais nommez le fichier Room.cs. Remplacez également le contenu par ce qui suit.

```

using Microsoft.Azure.Search;
using Microsoft.Azure.Search.Models;
using Newtonsoft.Json;

namespace FirstAzureSearchApp.Models
{
    public partial class Room
    {
        [IsSearchable]
        [Analyzer(AnalyzerName.AsString.EnMicrosoft)]

        public string Description { get; set; }

        [IsSearchable]
        [Analyzer(AnalyzerName.AsString.FrMicrosoft)]
        [JsonProperty("Description_fr")]
        public string DescriptionFr { get; set; }

        [IsSearchable, IsFilterable, IsFacetable]
        public string Type { get; set; }

        [IsFilterable, IsFacetable]
        public double? BaseRate { get; set; }

        [IsSearchable, IsFilterable, IsFacetable]
        public string BedOptions { get; set; }

        [IsFilterable, IsFacetable]

        public int SleepsCount { get; set; }

        [IsFilterable, IsFacetable]
        public bool? SmokingAllowed { get; set; }

        [IsSearchable, IsFilterable, IsFacetable]
        public string[] Tags { get; set; }
    }
}

```

5. L'ensemble de classes **Hotel**, **Address** et **Room** constitue ce qui dans Azure est connu sous le terme de *type complexe*, une fonctionnalité importante de Recherche cognitive Azure. Les types complexes peuvent avoir de nombreux niveaux de classes et de sous-classes et permettent de représenter des structures de données beaucoup plus complexes qu'au moyen de *types simples* (classe contenant uniquement des membres primitifs). Comme nous avons besoin d'un modèle supplémentaire, réeffectuez le processus de création de classe de modèle, mais cette fois appelez la classe `SearchData.cs` et remplacez le code par défaut par le code suivant.

```

using Microsoft.Azure.Search.Models;

namespace FirstAzureSearchApp.Models
{
    public class searchData
    {
        // The text to search for.
        public string searchText { get; set; }

        // The list of results.
        public DocumentSearchResult<Hotel> resultList;
    }
}

```

Cette classe contient l'entrée de l'utilisateur (`searchText`), et la sortie de la recherche (`resultList`). Le type

de la sortie, `DocumentSearchResult<Hotel>`, est critique ; en effet, il correspond exactement aux résultats de la recherche et nous devons passer cette référence à la vue.

## Créer une page web

Le projet que vous avez créé génère par défaut une série de vues clientes. Les vues exactes dépendent de la version de .NET Core que vous utilisez (nous utilisons la version 2.1 dans cet exemple). Elles sont regroupées dans le dossier `Vues` du projet. Il vous suffira de modifier le fichier `Index.cshtml` (dans le dossier `Vues/Accueil`).

Supprimez le contenu d'`Index.cshtml` dans son intégralité et regénérez le fichier dans les étapes suivantes.

1. Nous utilisons deux petites images dans la vue. Vous pouvez utiliser vos propres images ou copier les images à partir du projet GitHub : `azure-logo.png` et `search.png`. Ces deux images doivent être placées dans le dossier `wwwroot/images`.
2. La première ligne d'`Index.cshtml` doit référencer le modèle que nous allons utiliser pour communiquer des données entre le client (vue) et le serveur (contrôleur), qui correspond au modèle `SearchData` que nous avons créé. Ajoutez cette ligne au fichier `Index.cshtml`.

```
@model FirstAzureSearchApp.Models.SearchData
```

3. Comme il est courant d'entrer un titre pour la vue, les lignes suivantes doivent être :

```
@{  
    ViewData["Title"] = "Home Page";  
}
```

4. Après le titre, entrez une référence à une feuille de style HTML, que nous allons créer sous peu.

```
<head>  
    <link rel="stylesheet" href="~/css/hotels.css" />  
</head>
```

5. Attaquons-nous à présent au cœur de la vue. Un point clé à retenir est que la vue doit gérer deux situations. Tout d'abord, elle doit gérer l'affichage quand l'application est lancée et que l'utilisateur n'a pas encore entré de texte de recherche. Deuxièmement, elle doit gérer l'affichage des résultats, en plus de la zone de texte de recherche, à chaque utilisation par l'utilisateur. Pour gérer ces deux situations, nous devons vérifier si le modèle fourni à la vue est Null ou non. Un modèle Null indique que nous nous trouvons dans la première des deux situations (exécution initiale de l'application). Ajoutez le code suivant au fichier `Index.cshtml` et lisez les commentaires.

```

<body>
    <h1 class="sampleTitle">
        
        Hotels Search
    </h1>

    @using (Html.BeginForm("Index", "Home", FormMethod.Post))
    {
        // Display the search text box, with the search icon to the right of it.
        <div class="searchBoxForm">
            @Html.TextBoxFor(m => m.searchText, new { @class = "searchBox" }) <input
            class="searchBoxSubmit" type="submit" value="">
        </div>

        @if (Model != null)
        {
            // Show the result count.
            <p class="sampleText">
                @Html.DisplayFor(m => m.resultList.Results.Count) Results
            </p>

            @for (var i = 0; i < Model.resultList.Results.Count; i++)
            {
                // Display the hotel name and description.
                @Html.TextAreaFor(m => Model.resultList.Results[i].Document.HotelName, new { @class =
                "box1" })
                @Html.TextArea($"desc{i}", Model.resultList.Results[i].Document.Description, new { @class
                = "box2" })
            }
        }
    </body>

```

6. Enfin, nous ajoutons la feuille de style. Dans Visual Studio, dans le menu **Fichier**, sélectionnez **Nouveau/Fichier** puis **Feuille de style** (avec **Général** en surbrillance). Remplacez le code par défaut par ce qui suit. Nous n'allons pas nous attarder davantage sur ce fichier ; les styles correspondent à une syntaxe HTML standard.

```

textarea.box1 {
    width: 648px;
    height: 30px;
    border: none;
    background-color: azure;
    font-size: 14pt;
    color: blue;
    padding-left: 5px;
}

textarea.box2 {
    width: 648px;
    height: 100px;
    border: none;
    background-color: azure;
    font-size: 12pt;
    padding-left: 5px;
    margin-bottom: 24px;
}

.sampleTitle {
    font: 32px/normal 'Segoe UI Light',Arial,Helvetica,Sans-Serif;
    margin: 20px 0;
    font-size: 32px;
    text-align: left;
}

```

```

.sampleText {
    font: 16px/bold 'Segoe UI Light',Arial,Helvetica,Sans-Serif;
    margin: 20px 0;
    font-size: 14px;
    text-align: left;
    height: 30px;
}

.searchBoxForm {
    width: 648px;
    box-shadow: 0 0 0 1px rgba(0,0,0,.1), 0 2px 4px 0 rgba(0,0,0,.16);
    background-color: #fff;
    display: inline-block;
    border-collapse: collapse;
    border-spacing: 0;
    list-style: none;
    color: #666;
}

.searchBox {
    width: 568px;
    font-size: 16px;
    margin: 5px 0 1px 20px;
    padding: 0 10px 0 0;
    border: 0;
    max-height: 30px;
    outline: none;
    box-sizing: content-box;
    height: 35px;
    vertical-align: top;
}

.searchBoxSubmit {
    background-color: #fff;
    border-color: #fff;
    background-image: url(/images/search.png);
    background-repeat: no-repeat;
    height: 20px;
    width: 20px;
    text-indent: -99em;
    border-width: 0;
    border-style: solid;
    margin: 10px;
    outline: 0;
}

```

7. Enregistrez le fichier de feuille de style sous le nom `hotels.css`, dans le dossier `wwwroot/css`, avec le fichier `site.css` par défaut.

Notre vue est à présent complète. Nous sommes en bonne voie. Les modèles et les vues sont terminés ; seul reste à définir le contrôleur, qui relie tout cela.

## Définir des méthodes

Nous devons modifier le contenu d'un contrôleur (**contrôleur Home**), qui est créé par défaut.

1. Ouvrez le fichier `HomeController.cs` et remplacez les instructions **using** par ce qui suit.

```
using System;
using System.Diagnostics;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using FirstAzureSearchApp.Models;
using Microsoft.Extensions.Configuration;
using Microsoft.Azure.Search;
using Microsoft.Azure.Search.Models;
```

## Ajouter des méthodes Index

Nous avons besoin de deux méthodes **Index**, l'une ne prenant aucun paramètre (quand l'application est ouverte) et l'autre prenant un modèle en guise de paramètre (quand l'utilisateur a entré du texte de recherche). La première de ces méthodes est créée par défaut.

1. Ajoutez la méthode suivante, après la méthode **Index()** par défaut.

```
[HttpPost]
public async Task<ActionResult> Index(SearchData model)
{
    try
    {
        // Ensure the search string is valid.
        if (model.searchText == null)
        {
            model.searchText = "";
        }

        // Make the Azure Cognitive Search call.
        await RunQueryAsync(model);
    }

    catch
    {
        return View("Error", new ErrorViewModel { RequestId = "1" });
    }
    return View(model);
}
```

Notez la déclaration **async** de la méthode et l'appel **await** à **RunQueryAsync**. Ces mots clés ont pour rôle de rendre nos appels asynchrones et d'éviter, par là même, le blocage de threads sur le serveur.

Le bloc **catch** utilise le modèle d'erreur qui a été créé pour nous par défaut.

### Noter la gestion des erreurs et les autres méthodes et vues par défaut

L'ensemble de vues créées par défaut diffère légèrement suivant la version de .NET Core que vous utilisez. Pour .NET Core 2.1, les vues par défaut sont Index, About, Contact, Privacy et Error. Pour .NET Core 2.2, par exemple, les vues par défaut sont Index, Privacy et Error. Dans les deux cas, vous pouvez afficher ces pages par défaut lors de l'exécution de l'application et examiner la façon dont elles sont gérées dans le contrôleur.

Nous testerons la vue Error plus loin dans ce tutoriel.

Dans l'exemple GitHub, nous avons supprimé les vues non utilisées et leurs actions associées.

## Ajouter la méthode RunQueryAsync

L'appel de Recherche cognitive Azure est encapsulé dans notre méthode **RunQueryAsync**.

1. Tout d'abord, ajoutez des variables statiques pour configurer le service Azure et un appel pour les lancer.

```

private static SearchServiceClient _serviceClient;
private static ISearchIndexClient _indexClient;
private static IConfigurationBuilder _builder;
private static IConfigurationRoot _configuration;

private void InitSearch()
{
    // Create a configuration using the appsettings file.
    _builder = new ConfigurationBuilder().AddJsonFile("appsettings.json");
    _configuration = _builder.Build();

    // Pull the values from the appsettings.json file.
    string searchServiceName = _configuration["SearchServiceName"];
    string queryApiKey = _configuration["SearchServiceQueryApiKey"];

    // Create a service and index client.
    _serviceClient = new SearchServiceClient(searchServiceName, new
    SearchCredentials(queryApiKey));
    _indexClient = _serviceClient.Indexes.GetClient("hotels");
}

```

## 2. Maintenant, ajoutez la méthode `RunQueryAsync` proprement dite.

```

private async Task<ActionResult> RunQueryAsync(SearchData model)
{
    InitSearch();

    var parameters = new SearchParameters
    {
        // Enter Hotel property names into this list so only these values will be returned.
        // If Select is empty, all values will be returned, which can be inefficient.
        Select = new[] { "HotelName", "Description" }
    };

    // For efficiency, the search call should be asynchronous, so use SearchAsync rather than
    Search.
    model.resultList = await _indexClient.Documents.SearchAsync<Hotel>(model.searchText,
    parameters);

    // Display the results.
    return View("Index", model);
}

```

Dans cette méthode, nous nous assurons d'abord que notre configuration Azure est lancée, puis nous définissons certains paramètres de recherche. Les noms des champs dans le paramètre **Select** correspondent exactement aux noms de propriété dans la classe **hotel**. Il est possible d'omettre le paramètre **Select** ; dans ce cas, toutes les propriétés sont retournées. Toutefois, l'affectation daucun paramètre **Select** est inefficace si seul un sous-ensemble des données nous intéresse. En spécifiant les propriétés qui nous intéressent, seules ces propriétés sont retournées.

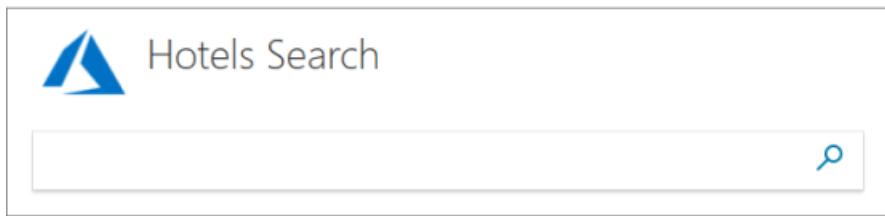
L'appel asynchrone à la fonctionnalité de recherche (`model.resultList = await _indexClient.Documents.SearchAsync<Hotel>(model.searchText, parameters);`) constitue le cœur de ce tutoriel et de cette application. La classe **DocumentSearchResult** est particulièrement intéressante ; une bonne idée (quand l'application est en cours d'exécution) consiste à y définir un point d'arrêt et à utiliser un débogueur pour examiner le contenu de `model.resultList`. Vous devriez trouver cela intuitif, les résultats contenant pour l'essentiel les données demandées.

Le moment de vérité est arrivé.

### Test de l'application

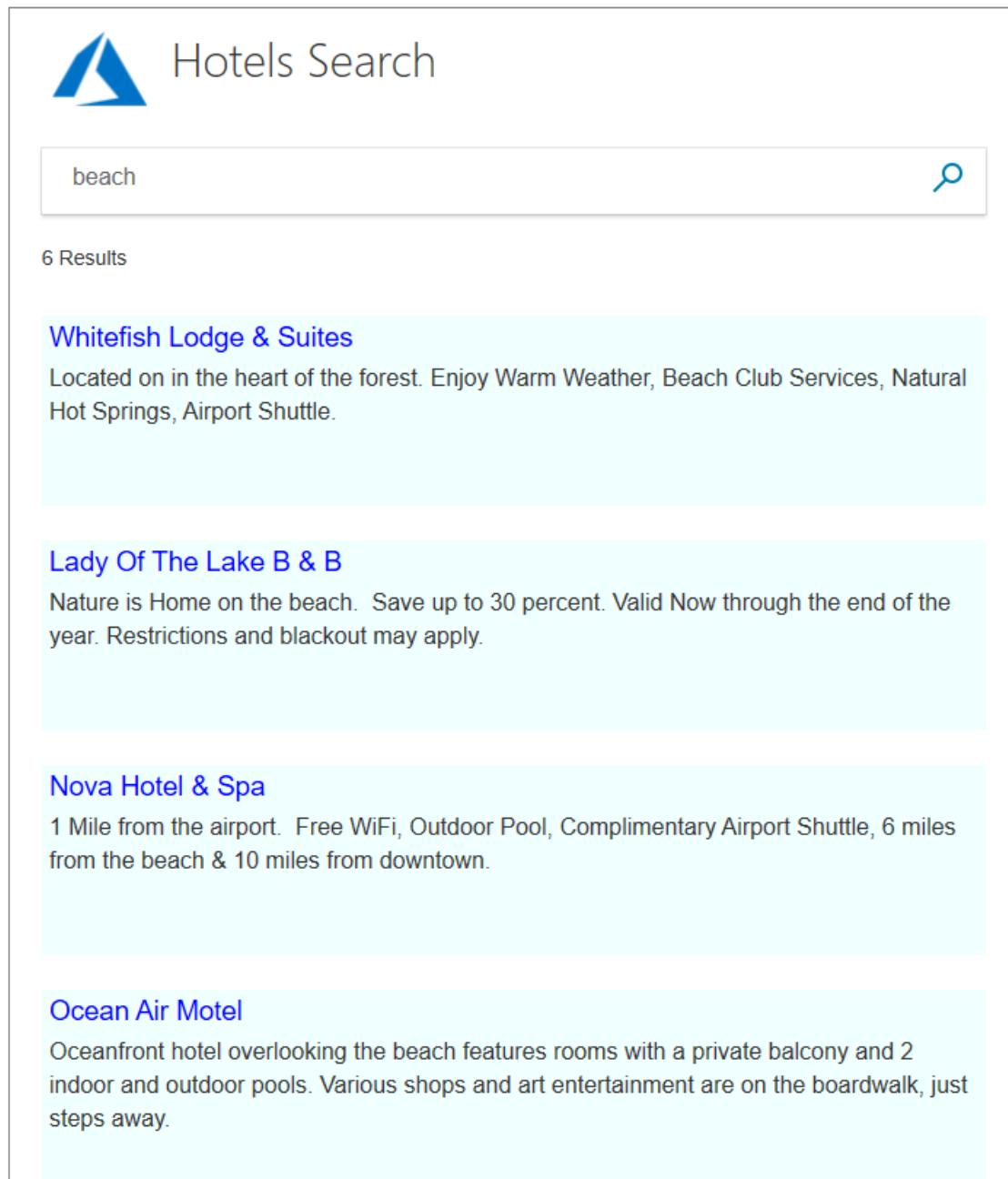
À présent, nous allons vérifier que l'application s'exécute correctement.

1. Sélectionnez Déboguer/Démarrer sans débogage ou appuyez sur la touche F5. Si vous avez tout codé correctement, vous obtenez la vue Index initiale.



The screenshot shows the 'Hotels Search' application's home screen. It features a blue triangle logo on the left, followed by the text 'Hotels Search'. Below this is a search bar containing a placeholder text area and a magnifying glass icon on the right. The entire interface is contained within a light gray border.

2. Entrez du texte tel que « beach » (ou n'importe quel texte qui vous vient à l'esprit), puis cliquez sur l'icône de recherche. Vous devriez obtenir des résultats.



The screenshot displays the search results for the query 'beach'. At the top, it says '6 Results'. Below this, there are four hotel listings, each enclosed in a light green box:

- Whitefish Lodge & Suites**  
Located on in the heart of the forest. Enjoy Warm Weather, Beach Club Services, Natural Hot Springs, Airport Shuttle.
- Lady Of The Lake B & B**  
Nature is Home on the beach. Save up to 30 percent. Valid Now through the end of the year. Restrictions and blackout may apply.
- Nova Hotel & Spa**  
1 Mile from the airport. Free WiFi, Outdoor Pool, Complimentary Airport Shuttle, 6 miles from the beach & 10 miles from downtown.
- Ocean Air Motel**  
Oceanfront hotel overlooking the beach features rooms with a private balcony and 2 indoor and outdoor pools. Various shops and art entertainment are on the boardwalk, just steps away.

3. Essayez d'entrer « five star ». Comme vous pouvez le constater, vous n'obtenez aucun résultat. Une recherche plus sophistiquée traiterait « five star » (cinq étoiles) comme un synonyme de « luxury » (luxe) et retournerait les résultats correspondants. L'utilisation de synonymes est disponible dans Recherche cognitive Azure, mais nous n'allons pas l'aborder dans les premiers tutoriels.
4. Essayez d'entrer « hot » comme texte de recherche. Vous n'obtenez *aucune* entrée contenant le mot « hotel ». Notre recherche porte uniquement sur les mots entiers, même si quelques résultats sont retournés.

5. Essayez d'autres mots, tels que « pool », « sunshine » ou « view ». Vous verrez Recherche cognitive Azure fonctionner à son niveau le plus simple, mais néanmoins convaincant.

## Tester des erreurs et conditions limites

Il est important de vérifier que nos fonctionnalités de gestion des erreurs fonctionnent comme prévu, même quand les choses fonctionnent parfaitement.

1. Dans la méthode **Index**, après l'appel `try {`, entrez la ligne `Throw new Exception()`. Cette exception force une erreur quand nous effectuons une recherche sur du texte.
2. Exécutez l'application, entrez « bar » comme texte de recherche, puis cliquez sur l'icône de recherche. L'exception doit aboutir à la vue d'erreur.

Error.

An error occurred while processing your request.

Request ID: [1](#)

Development Mode

Swapping to **Development** environment will display more detailed information about the error that occurred.

**Development environment should not be enabled in deployed applications**, as it can result in sensitive information from exceptions being displayed to end users. For local debugging, development environment can be enabled by setting the **ASPNETCORE\_ENVIRONMENT** environment variable to **Development**, and restarting the application.

© 2019 - FirstAzureSearchApp

### IMPORTANT

Il est considéré comme risqué du point de vue de la sécurité de retourner des numéros d'erreur interne dans les pages d'erreur. Si votre application est prévue pour une utilisation générale, effectuez quelques recherches concernant la sécurité et les bonnes pratiques relatives à ce qu'il faut retourner en cas d'erreur.

3. Supprimez `Throw new Exception()` quand vous jugez que la gestion des erreurs fonctionne correctement.

## Éléments importants à retenir

Retenez les points importants suivants de ce projet :

- Un appel de Recherche cognitive Azure est concis et il est facile d'interpréter les résultats.
- Les appels asynchrones ajoutent une petite dose de complexité au contrôleur, mais constituent la meilleure approche si vous avez l'intention de développer des applications de qualité.
- Cette application a effectué une recherche de texte simple, définie par ce qui est configuré dans `searchParameters`. Toutefois, cette classe peut être remplie avec de nombreux membres qui rendent une recherche plus sophistiquée. Il n'est pas nécessaire d'effectuer un travail supplémentaire considérable pour rendre cette application beaucoup plus puissante.

## Étapes suivantes

Pour offrir à l'utilisateur la meilleure expérience de Recherche cognitive Azure, nous devons ajouter des fonctionnalités, notamment la pagination (au moyen de numéros de page ou d'un défilement infini) ainsi que l'auto-complétion et les suggestions. Nous devons aussi envisager des paramètres de recherche plus complexes (par exemple, des recherches géographiques permettant de trouver des hôtels dans un rayon spécifique autour d'un point donné et le tri des résultats de la recherche).

Ces étapes suivantes sont traitées dans une série de tutoriels. Commençons par la pagination.



# Tutoriel : Ajouter la pagination aux résultats de recherche à l'aide du SDK .NET

04/10/2020 • 27 minutes to read • [Edit Online](#)

Découvrez comment implémenter deux systèmes de pagination différents, le premier basé sur des numéros de page et le second sur un défilement infini. Les deux systèmes de pagination sont largement utilisés, et le choix de l'un ou de l'autre dépend de l'expérience utilisateur que vous souhaitez associer aux résultats. Ce tutoriel intègre les systèmes de pagination au projet créé dans le [Tutoriel C# : Créer votre première application - Recherche cognitive Azure](#).

Dans ce tutoriel, vous allez apprendre à :

- Étendre votre application avec une pagination numérotée
- Étendre votre application avec un défilement infini

## Prérequis

Pour suivre ce didacticiel, vous devez effectuer les opérations suivantes :

Disposer pleinement du projet [Tutoriel C# : Créer votre première application - Recherche cognitive Azure](#). Ce projet peut être votre propre version ou vous pouvez l'installer à partir de GitHub : [Créer votre première application](#).

## Étendre votre application avec une pagination numérotée

La pagination numérotée est le système de pagination de choix pour les principaux moteurs de recherche Internet et la plupart des autres sites web de recherche. La pagination numérotée inclut généralement des options « page suivante » et « page précédente » en plus d'une plage de numéros de pages réels. De plus, des options « première page » et « dernière page » peuvent également être disponibles. Ces options donnent certainement à l'utilisateur la possibilité de contrôler l'exploration des résultats basés sur des pages.

Nous allons ajouter un système qui inclut les options de première page, de page précédente, de page suivante et de dernière page ainsi que des numéros de page qui, au lieu de commencer par 1, entourent la page sur laquelle se trouve actuellement l'utilisateur (ainsi, par exemple, si l'utilisateur consulte la page 10, les numéros de page 8, 9, 10, 11 et 12 peuvent être affichés).

Le système est suffisamment flexible pour permettre de définir le nombre de numéros de page visibles dans une variable globale.

Le système traite les boutons de numéro de page les plus à gauche et à droite comme des boutons spéciaux, en ce sens qu'ils déclenchent le changement de la plage des numéros de page affichés. Par exemple, si les numéros de page 8, 9, 10, 11 et 12 sont affichés et que l'utilisateur clique sur 8, c'est la plage de numéros de page 6, 7, 8, 9 et 10 qui est alors affichée. Un décalage similaire vers la droite a lieu s'il sélectionne 12.

### Ajouter des champs de pagination au modèle

Si ce n'est déjà fait, ouvrez la solution de page de recherche de base.

1. Ouvrez le fichier de modèle SearchData.cs.
2. Tout d'abord, ajoutez des variables globales. Dans MVC, les variables globales sont déclarées dans leur propre classe statique. `ResultsPerPage` définit le nombre de résultats par page. `MaxPageRange` détermine le nombre de numéros de page visibles sur la vue. `PageRangeDelta` détermine de quel

nombre de pages à gauche ou à droite la plage de pages doit être décalée quand le numéro de page le plus à gauche ou à droite est sélectionné. En général, ce dernier nombre est égal à environ la moitié de **MaxPageRange**. Ajoutez le code suivant à l'espace de noms.

```
public static class GlobalVariables
{
    public static int ResultsPerPage
    {
        get
        {
            return 3;
        }
    }

    public static int MaxPageRange
    {
        get
        {
            return 5;
        }
    }

    public static int PageRangeDelta
    {
        get
        {
            return 2;
        }
    }
}
```

#### TIP

Si vous exécutez ce projet sur un appareil doté d'un écran plus petit, par exemple un ordinateur portable, envisagez de définir **ResultsPerPage** sur 2.

3. Ajoutez des propriétés de pagination à la classe **SearchData**, par exemple, après la propriété **searchText**.

```
// The current page being displayed.
public int currentPage { get; set; }

// The total number of pages of results.
public int pageCount { get; set; }

// The left-most page number to display.
public int leftMostPage { get; set; }

// The number of page numbers to display - which can be less than MaxPageRange towards the end of
// the results.
public int pageRange { get; set; }

// Used when page numbers, or next or prev buttons, have been selected.
public string paging { get; set; }
```

#### Ajouter un tableau d'options de pagination à la vue

1. Ouvrez le fichier `index.cshtml` et ajoutez le code suivant juste avant la balise `</body>` fermante. Ce nouveau code présente un tableau d'options de pagination : première page, page précédente, pages 1, 2, 3, 4, 5, page suivante, dernière page.

```
@if (Model != null && Model.pageCount > 1)
{
```

```

// If there is more than one page of results, show the paging buttons.


|                                                                                                                                                                                                      |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| @if (Model.currentPage > 0) {     <p class="pageButton">         @Html.ActionLink(" <", "Page", "Home", new { paging = "0" }, null)     </p> } else {     <p class="pageButtonDisabled"> &lt;</p> }  |
| @if (Model.currentPage > 0) {     <p class="pageButton">         @Html.ActionLink("<", "Page", "Home", new { paging = "prev" }, null)     </p> } else {     <p class="pageButtonDisabled">&lt;</p> } |


```

```

        <p class="pageButtonDisabled">&gt; | </p>
    }
</td>
</tr>
</table>
}

```

Nous utilisons un tableau HTML pour aligner parfaitement les informations. Toutefois, l'action provient en totalité des instructions @Html.ActionLink, chacune appelant le contrôleur avec un **nouveau** modèle créé avec des entrées différentes pour la propriété **paging** que nous avons ajoutée.

Les options relatives aux première et dernière pages n'envoient pas de chaînes telles que « first » (première) et « last » (dernière), mais les numéros de page corrects.

2. Ajoutez des classes de pagination à la liste des styles HTML dans le fichier hotels.css. La classe **pageSelected** identifie la page que l'utilisateur est en train de consulter (en affichant le numéro en gras) dans la liste des numéros de page.

```

.pageButton {
    border: none;
    color: darkblue;
    font-weight: normal;
    width: 50px;
}

.pageSelected {
    border: none;
    color: black;
    font-weight: bold;
    width: 50px;
}

.pageButtonDisabled {
    border: none;
    color: lightgray;
    font-weight: bold;
    width: 50px;
}

```

### Ajouter une action Page au contrôleur

1. Ouvrez le fichier HomeController.cs, puis ajoutez l'action **Page**. Cette action répond aux options de page sélectionnées.

```

public async Task<ActionResult> Page(SearchData model)
{
    try
    {
        int page;

        switch (model.paging)
        {
            case "prev":
                page = (int) TempData["page"] - 1;
                break;

            case "next":
                page = (int) TempData["page"] + 1;
                break;

            default:
                page = int.Parse(model.paging);
                break;
        }

        // Recover the leftMostPage.
        int leftMostPage = (int) TempData["leftMostPage"];

        // Recover the search text and search for the data for the new page.
        model.searchText = TempData["searchfor"].ToString();

        await RunQueryAsync(model, page, leftMostPage);

        // Ensure Temp data is stored for next call, as TempData only stored for one call.
        TempData["page"] = (object)page;
        TempData["searchfor"] = model.searchText;
        TempData["leftMostPage"] = model.leftMostPage;
    }

    catch
    {
        return View("Error", new ErrorViewModel { RequestId = "2" });
    }
    return View("Index", model);
}

```

La méthode **RunQueryAsync** affiche une erreur de syntaxe, en raison du troisième paramètre ; nous y reviendrons un peu plus tard.

#### NOTE

Les appels **TempData** stockent une valeur (un **objet**) dans un stockage temporaire, bien que ce stockage persiste pendant *uniquement* un seul appel. Si nous stockons une information dans des données temporaires, elle est disponible pour le prochain appel d'une action de contrôleur, mais certainement pas pour l'appel suivant ! En raison de cette courte durée de vie, nous stockons le texte de recherche et les propriétés de pagination dans un stockage temporaire à chaque appel de **Page**.

2. L'action **Index(model)** doit être mise à jour pour stocker les variables temporaires et pour ajouter le paramètre de page la plus à gauche à l'appel **RunQueryAsync**.

```
public async Task<ActionResult> Index(SearchData model)
{
    try
    {
        // Ensure the search string is valid.
        if (model.searchText == null)
        {
            model.searchText = "";
        }

        // Make the search call for the first page.
        await RunQueryAsync(model, 0, 0);

        // Ensure temporary data is stored for the next call.
        TempData["page"] = 0;
        TempData["leftMostPage"] = 0;
        TempData["searchfor"] = model.searchText;
    }

    catch
    {
        return View("Error", new ErrorViewModel { RequestId = "1" });
    }
    return View(model);
}
```

3. La méthode **RunQueryAsync** doit être mise à jour de manière significative. Nous utilisons les champs **Skip**, **Top** et **IncludeTotalResultCount** de la classe **SearchParameters** pour ne demander que l'équivalent d'une page de résultats, à partir du paramètre **Skip**. Nous devons également calculer les variables de pagination pour notre vue. Remplacez la méthode entière par le code suivant.

```

private async Task<ActionResult> RunQueryAsync(SearchData model, int page, int leftMostPage)
{
    InitSearch();

    var parameters = new SearchParameters
    {
        // Enter Hotel property names into this list so only these values will be returned.
        // If Select is empty, all values will be returned, which can be inefficient.
        Select = new[] { "HotelName", "Description" },
        SearchMode = SearchMode.All,

        // Skip past results that have already been returned.
        Skip = page * GlobalVariables.ResultsPerPage,

        // Take only the next page worth of results.
        Top = GlobalVariables.ResultsPerPage,

        // Include the total number of results.
        IncludeTotalResultCount = true,
    };

    // For efficiency, the search call should be asynchronous, so use SearchAsync rather than
    Search.
    model.resultList = await _indexClient.Documents.SearchAsync<Hotel>(model.searchText,
parameters);

    // This variable communicates the total number of pages to the view.
    model.pageCount = ( (int)model.resultList.Count + GlobalVariables.ResultsPerPage - 1 ) /
GlobalVariables.ResultsPerPage;

    // This variable communicates the page number being displayed to the view.
    model.currentPage = page;

    // Calculate the range of page numbers to display.
    if (page == 0)
    {
        leftMostPage = 0;
    }
    else
    {
        if (page <= leftMostPage)
        {
            // Trigger a switch to a lower page range.
            leftMostPage = Math.Max(page - GlobalVariables.PageRangeDelta, 0);
        }
        else
        {
            if (page >= leftMostPage + GlobalVariables.MaxPageRange - 1)
            {
                // Trigger a switch to a higher page range.
                leftMostPage = Math.Min(page - GlobalVariables.PageRangeDelta, model.pageCount -
GlobalVariables.MaxPageRange);
            }
            model.leftMostPage = leftMostPage;
        }
    }

    // Calculate the number of page numbers to display.
    model.pageRange = Math.Min(model.pageCount - leftMostPage, GlobalVariables.MaxPageRange);

    return View("Index", model);
}

```

- Enfin, nous devons apporter une petite modification à la vue. La variable **resultsList.Results.Count** contient maintenant le nombre de résultats renvoyés dans une page (3 dans notre exemple), pas le nombre total. Comme nous avons défini **IncludeTotalResultCount** sur true, la variable **resultsList.Count** contient à présent le nombre total de résultats. Recherchez l'emplacement auquel est affiché le nombre de résultats dans la vue et remplacez-le par le code suivant.

```
// Show the result count.  
<p class="sampleText">  
    @Html.DisplayFor(m => m.resultList.Count) Results  
</p>
```

#### NOTE

La définition d'`IncludeTotalResultCount` sur true entraîne une légère diminution des performances, car ce total doit être calculé par Recherche cognitive Azure. Avec des jeux de données complexes, un avertissement s'affiche, indiquant que la valeur renvoyée est une *approximation*. Dans le cas de nos données d'hôtel, la valeur sera précise.

### Compiler et exécuter l'application

Sélectionnez à présent Démarrer sans débogage (ou appuyez sur la touche F5).

- Effectuez une recherche portant sur un texte qui donne beaucoup de résultats (par exemple, « wifi »). Pouvez-vous parcourir les résultats de façon claire ?

The screenshot shows the 'Hotels Search - Paging' application. A search bar at the top contains the word 'wifi'. Below the search bar, the text '19 Results' is displayed. Three hotel results are listed:

- Super Deluxe Inn & Suites**  
Complimentary Airport Shuttle & WiFi. Book Now and save - Spacious All Suite Hotel, Indoor/Outdoor Pool, Fitness Center, Florida Green certified, Starbucks Coffee, HDTV
- Double Sanctuary Resort**  
5\* Luxury Hotel - Biggest Rooms in the city. #1 Hotel in the area listed by Conde Nast Traveler. Free WiFi, Flexible check in/out, Fitness Center & Nespresso in room.
- Veteran Right Track**  
Free Shuttle to the airport and casinos. Free breakfast and WiFi.

At the bottom of the page, there is a navigation bar with page numbers from 1 to 5 and arrows for navigation.

- Essayez de cliquer sur le numéro de page le plus à droite, puis sur celui le plus à gauche. Les numéros de page s'ajustent-ils en conséquence afin que le numéro de la page actuelle se trouve au centre ?
- Les options de première et dernière pages sont-elles utiles ? Certaines recherches web populaires utilisent ces options, d'autres pas.

4. Accédez à la dernière page de résultats. La dernière page est la seule page qui peut contenir moins de **ResultsPerPage** résultats.

The screenshot shows a search interface with a blue logo icon on the left. The main title is "Hotels Search - Paging". Below it is a search bar containing the text "wifi" and a magnifying glass icon. Underneath the search bar, the text "19 Results" is displayed. A list of hotel results is shown, starting with "Whitefish Lodge & Suites", which is described as being located in the heart of the forest and offering various amenities like warm weather, beach club services, natural hot springs, and an airport shuttle. Below this result, there is a pagination control with buttons labeled "< < 3 4 5 6 7 > >|".

5. Tapez « town » et cliquez sur l'icône de recherche. Aucune option de pagination n'est affichée s'il y a moins de l'équivalent d'une page de résultats.

The screenshot shows a search interface with a blue logo icon on the left. The main title is "Hotels Search - Paging". Below it is a search bar containing the text "town" and a magnifying glass icon. Underneath the search bar, the text "2 Results" is displayed. A list of hotel results is shown, starting with "Twin Dome Motel", which is described as being situated in a nineteenth century plaza and renovated to modern standards. Below this result, there is another hotel entry for "Commander Hotel", described as being right in the heart of campus and having a prime location between the union and stadium. There is no visible pagination control.

À présent, enregistrez ce projet et essayons une autre forme de pagination.

## Étendre votre application avec un défilement infini

Le défilement infini est déclenché quand l'utilisateur fait défiler une barre de défilement verticale jusqu'au dernier

des résultats affichés. Quand cela se produit, un appel au serveur est effectué pour obtenir la page de résultats suivante. S'il n'y a plus aucun résultat, rien n'est retourné et la barre de défilement verticale ne change pas. S'il existe plus de résultats, ils sont ajoutés à la page actuelle et la barre de défilement change pour indiquer que des résultats supplémentaires sont disponibles.

Le point important ici est que la page en cours d'affichage n'est pas remplacée, mais enrichie des nouveaux résultats. Un utilisateur peut toujours faire défiler l'affichage jusqu'aux premiers résultats de la recherche.

Pour implémenter un défilement infini, nous allons partir du projet tel qu'il existait avant que ne soient ajoutés des éléments de défilement de numéro de page. Ainsi, le cas échéant, créez une autre copie de la page de recherche de base à partir de GitHub : [Créer votre première application](#).

### Ajouter des champs de pagination au modèle

1. Tout d'abord, ajoutez une propriété **paging** à la classe **SearchData** (dans le fichier de modèle **SearchData.cs**).

```
// Record if the next page is requested.  
public string paging { get; set; }
```

Cette variable est une chaîne qui conserve « `next` » si la page de résultats suivante doit être envoyée, ou `Null` pour la première page d'une recherche.

2. Dans le même fichier et dans l'espace de noms, ajoutez une classe de variable globale avec une seule propriété. Dans MVC, les variables globales sont déclarées dans leur propre classe statique. **ResultsPerPage** définit le nombre de résultats par page.

```
public static class GlobalVariables  
{  
    public static int ResultsPerPage  
    {  
        get  
        {  
            return 3;  
        }  
    }  
}
```

### Ajouter une barre de défilement verticale à la vue

1. Dans le fichier `index.cshtml`, recherchez la section qui affiche les résultats (elle commence par `@if (Model != null)` ).
2. Remplacez la section par le code ci-dessous. La nouvelle section `<div>` englobe la zone que l'utilisateur doit être en mesure de faire défiler et ajoute un attribut `overflow-y` et un appel à une fonction `onscroll` appelée « `scrolled()` », comme suit.

```

@if (Model != null)
{
    // Show the result count.
    <p class="sampleText">
        @Html.DisplayFor(m => m.resultList.Count) Results
    </p>

    <div id="myDiv" style="width: 800px; height: 450px; overflow-y: scroll;" onscroll="scrolled()">

        <!-- Show the hotel data. -->
        @for (var i = 0; i < Model.resultList.Results.Count; i++)
        {
            // Display the hotel name and description.
            @Html.TextAreaFor(m => Model.resultList.Results[i].Document.HotelName, new { @class =
"box1" })
            @Html.TextArea($"desc{i}", Model.resultList.Results[i].Document.Description, new {
@class = "box2" })
        }
    </div>
}

```

3. Juste sous la boucle, après la balise `</div>`, ajoutez la fonction `scrolled`.

```

<script>
    function scrolled() {
        if (myDiv.offsetHeight + myDiv.scrollTop >= myDiv.scrollHeight) {
            $.getJSON("/Home/Next", function (data) {
                var div = document.getElementById('myDiv');

                // Append the returned data to the current list of hotels.
                for (var i = 0; i < data.length; i += 2) {
                    div.innerHTML += '\n<textarea class="box1">' + data[i] + '</textarea>';
                    div.innerHTML += '\n<textarea class="box2">' + data[i + 1] +
                    '</textarea>';
                }
            });
        }
    }
</script>

```

L'instruction `if` dans le script ci-dessus vérifie si l'utilisateur a fait défiler jusqu'au bas de la barre de défilement verticale. Si tel le cas, un appel est effectué à destination d'une action appelée **Next** par le biais du contrôleur **Home**. Aucune autre information n'est nécessaire pour le contrôleur, qui retourne la page de données suivante. Ces données sont ensuite mises en forme à l'aide des mêmes styles HTML que la page d'origine. Si aucun résultat n'est retourné, rien n'est ajouté et les choses en restent là.

## Gérer l'action **Next**

Seules trois actions doivent être envoyées au contrôleur : la première exécution de l'application, qui appelle **Index()** , la première recherche par l'utilisateur, qui appelle **Index(model)** , puis les appels suivants qui permettent d'obtenir d'autres résultats, par le biais de **Next(model)** .

- Ouvrez le fichier du contrôleur Home et supprimez la méthode **RunQueryAsync** du tutoriel d'origine.
- Remplacez l'action **Index(model)** par le code suivant. Il gère désormais le champ **paging** quand il est Null ou défini sur « next », et gère l'appel à Recherche cognitive Azure.

```

public async Task<ActionResult> Index(SearchData model)
{
    try
    {
        InitSearch();

        int page;

        if (model.paging != null && model.paging == "next")
        {
            // Increment the page.
            page = (int)TempData["page"] + 1;

            // Recover the search text.
            model.searchText = TempData["searchfor"].ToString();
        }
        else
        {
            // First call. Check for valid text input.
            if (model.searchText == null)
            {
                model.searchText = "";
            }
            page = 0;
        }

        // Setup the search parameters.
        var parameters = new SearchParameters
        {
            // Enter Hotel property names into this list so only these values will be returned.
            // If Select is empty, all values will be returned, which can be inefficient.
            Select = new[] { "HotelName", "Description" },
            SearchMode = SearchMode.All,

            // Skip past results that have already been returned.
            Skip = page * GlobalVariables.ResultsPerPage,

            // Take only the next page worth of results.
            Top = GlobalVariables.ResultsPerPage,

            // Include the total number of results.
            IncludeTotalResultCount = true,
        };

        // For efficiency, the search call should be asynchronous, so use SearchAsync rather than
        Search.
        model.resultList = await _indexClient.Documents.SearchAsync<Hotel>(model.searchText,
parameters);

        // Ensure TempData is stored for the next call.
        TempData["page"] = page;
        TempData["searchfor"] = model.searchText;
    }
    catch
    {
        return View("Error", new ErrorViewModel { RequestId = "1" });
    }
    return View("Index", model);
}

```

Comme dans le cas de la méthode de pagination numérotée, nous utilisons les paramètres de recherche **Skip** et **Top** pour demander uniquement les données qui doivent être retournées.

3. Ajoutez l'action **Next** au contrôleur Home. Comme vous pouvez le constater, il retourne une liste, avec pour chaque hôtel deux éléments : son nom et sa description. Ce format est conçu pour être compatible avec

l'utilisation de la fonction **scrolled** appliquée aux données retournées dans la vue.

```
public async Task<ActionResult> Next(SearchData model)
{
    // Set the next page setting, and call the Index(model) action.
    model.paging = "next";
    await Index(model);

    // Create an empty list.
    var nextHotels = new List<string>();

    // Add a hotel name, then description, to the list.
    for (int n = 0; n < model.resultList.Results.Count; n++)
    {
        nextHotels.Add(model.resultList.Results[n].Document.HotelName);
        nextHotels.Add(model.resultList.Results[n].Document.Description);
    }

    // Rather than return a view, return the list of data.
    return new JsonResult(nextHotels);
}
```

4. Si vous obtenez une erreur de syntaxe sur **List<string>** , ajoutez la directive **using** suivante au début du fichier du contrôleur.

```
using System.Collections.Generic;
```

### Compiler et exécuter votre projet

Sélectionnez à présent Démarrer sans débogage (ou appuyez sur la touche F5).

1. Entrez un terme qui donne beaucoup de résultats (par exemple, « pool »), puis testez la barre de défilement verticale. Une nouvelle page de résultats est-elle déclenchée ?



16 Results

## Twin Dome Motel

The hotel is situated in a nineteenth century plaza, which has been expanded and renovated to the highest architectural standards to create a modern, functional and first-class hotel in which art and unique historical elements coexist with the most modern comforts.

## Gacc Capital

Chic hotel near the city. High-rise hotel in downtown, walking distance to theaters, restaurants and shops, complete with wellness programs.

## Pull'r Inn Motel

The hotel rooms and suites offer the perfect blend of beauty and elegance. Our rooms will elevate your stay, whether you're traveling for business, celebrating a honeymoon, or just looking for a remarkable getaway. With views of the valley or the iconic fountains right from your suite, your stay will be nothing short of unforgettable.

### TIP

Pour qu'une barre de défilement s'affiche dans la première page de résultats, la hauteur de cette page doit être légèrement supérieure à la hauteur de la zone dans laquelle les résultats sont affichés. Dans notre exemple .box1 a une hauteur de 30 pixels, .box2 une hauteur de 100 pixels et une marge inférieure de 24 pixels. Ainsi, chaque entrée utilise 154 pixels. Trois entrées occupent  $3 \times 154 = 462$  pixels. Pour qu'une barre de défilement verticale s'affiche, la hauteur de la zone d'affichage doit être définie sur une valeur inférieure à 462 pixels ; même 461 peut faire l'affaire. Ce problème ne concerne que la première page ; pour les pages suivantes, une barre de défilement apparaît systématiquement. La ligne à mettre à jour est : `<div id="myDiv" style="width: 800px; height: 450px; overflow-y: scroll;" onscroll="scrolled()">`.

2. Faites défiler jusqu'au bas des résultats. Comme vous pouvez le constater, toutes les informations se trouvent désormais sur la page de la vue. Vous pouvez faire défiler jusqu'en haut sans déclencher d'appels de serveur.

Des systèmes de défilement infini plus sophistiqués peuvent utiliser la roulette de la souris, ou un autre mécanisme similaire, pour déclencher le chargement d'une nouvelle page de résultats. Le défilement infini, qui évite des clics de souris supplémentaires, présente un certain charme, mais nous n'allons pas l'approfondir dans ces tutoriels, laissant la voie libre à l'examen d'autres options !

## Éléments importants à retenir

Retenez les points importants suivants de ce projet :

- La pagination numérotée est appropriée pour les recherches où l'ordre des résultats est quelque peu arbitraire, ce qui signifie que ce qui intéresse l'utilisateur peut se trouver dans les dernières pages.
- Le défilement infini convient quand l'ordre des résultats est particulièrement important. C'est, par exemple, le cas si les résultats sont triés en fonction de la distance par rapport au centre d'une ville de destination.

- Dans une certaine mesure, la pagination numérotée permet une meilleure navigation. Par exemple, l'utilisateur peut se souvenir qu'un résultat intéressant se trouvait dans la page 6, facilité de référence que n'offre pas le défilement infini.
- Le défilement infini, dépourvu de numéros de page sur lesquels il est nécessaire de cliquer, présente un certain charme.
- Une caractéristique du défilement infini, et qui fait son efficacité, est que les résultats sont ajoutés à une page existante ; ils ne remplacent pas cette page.
- Le stockage temporaire persiste le temps d'un appel et doit être réinitialisé pour être conservé d'un appel à l'autre.

## Étapes suivantes

La pagination est essentielle pour les recherches sur Internet. La pagination ayant été bien couverte, l'étape suivante consiste à améliorer l'expérience utilisateur en ajoutant des recherches avec auto-complétion.

[Tutoriel C# : Ajouter l'auto-complétion et des suggestions - Recherche cognitive Azure](#)

# Tutoriel : Ajouter l'autocomplétion et les suggestions à l'aide du SDK .NET

04/10/2020 • 19 minutes to read • [Edit Online](#)

Découvrez comment implémenter l'autocomplétion (requêtes en saisie semi-automatique et suggestions de documents) quand un utilisateur commence à taper dans une zone de recherche. Dans ce tutoriel, nous allons montrer les requêtes autocomplétées et les résultats de suggestions séparément, puis ensemble. Il suffit à l'utilisateur de taper deux ou trois caractères pour obtenir tous les résultats disponibles.

Dans ce tutoriel, vous allez apprendre à :

- Ajouter des suggestions
- Mettre en évidence les suggestions
- Ajouter l'autocomplétion
- Combiner l'auto-complétion et les suggestions

## Prérequis

Ce tutoriel fait partie d'une série et repose sur le projet de pagination créé dans le [Tutoriel C# : Pagination des résultats de la recherche - Recherche cognitive Azure](#).

Vous pouvez également télécharger et exécuter la solution pour ce tutoriel spécifique : [3-add-typeahead](#).

## Ajouter des suggestions

Nous allons commencer par le cas le plus simple de proposition de choix à l'utilisateur : une liste déroulante de suggestions.

1. Dans le fichier index.cshtml, remplacez l'`@id` de l'instruction `TextBoxFor` par `azureautosuggest`.

```
@Html.TextBoxFor(m => m.searchText, new { @class = "searchBox", @id = "azureautosuggest" }) <input value="" class="searchBoxSubmit" type="submit">
```

2. Après cette instruction, après la balise `</div>` fermante, entrez ce script. Ce script tire parti du [widget d'autocomplétion](#) de la bibliothèque de l'interface utilisateur open source jQuery pour présenter la liste déroulante des résultats suggérés.

```
<script>
    $("#azureautosuggest").autocomplete({
        source: "/Home/Suggest?highlights=false&fuzzy=false",
        minLength: 2,
        position: {
            my: "left top",
            at: "left-23 bottom+10"
        }
    });
</script>
```

L'ID « `azureautosuggest` » connecte le script ci-dessus à la zone de recherche. L'option `source` du widget est définie sur une méthode `Suggest` qui appelle l'API `Suggest` avec deux paramètres de requête : `highlights` et `fuzzy`, tous deux ayant la valeur `false` dans cette instance. De plus, un minimum de deux caractères sont

nécessaires pour déclencher la recherche.

## Ajouter des références aux scripts jQuery à la vue

1. Pour accéder à la bibliothèque jQuery, remplacez la section <head> du fichier de la vue par le code suivant :

```
<head>
    <meta charset="utf-8">
    <title>Typeahead</title>
    <link href="https://code.jquery.com/ui/1.12.1/themes/start/jquery-ui.css"
          rel="stylesheet">
    <script src="https://code.jquery.com/jquery-1.10.2.js"></script>
    <script src="https://code.jquery.com/ui/1.12.1/jquery-ui.js"></script>

    <link rel="stylesheet" href="~/css/hotels.css" />
</head>
```

2. Étant donné que nous introduisons une nouvelle référence jQuery, nous devons également supprimer (ou commenter) la référence jQuery par défaut dans le fichier \_Layout.cshtml (dans le dossier **Views/Shared**). Recherchez les lignes suivantes et commentez la première ligne de script comme indiqué. Cette modification permet d'éviter les conflits de références à jQuery.

```
<environment include="Development">
    <!-- <script src="~/lib/jquery/dist/jquery.js"></script> -->
    <script src="~/lib/bootstrap/dist/js/bootstrap.js"></script>
    <script src="~/js/site.js" asp-append-version="true"></script>
</environment>
```

Maintenant, nous pouvons utiliser les fonctions jQuery d'autocomplétion prédéfinies.

## Ajouter l'action Suggest au contrôleur

1. Dans le contrôleur Home, ajoutez l'action **Suggest** (par exemple, après l'action **Page**).

```
public async Task<ActionResult> Suggest(bool highlights, bool fuzzy, string term)
{
    InitSearch();

    // Setup the suggest parameters.
    var parameters = new SuggestParameters()
    {
        UseFuzzyMatching = fuzzy,
        Top = 8,
    };

    if (highlights)
    {
        parameters.HighlightPreTag = "<b>";
        parameters.HighlightPostTag = "</b>";
    }

    // Only one suggester can be specified per index. It is defined in the index schema.
    // The name of the suggester is set when the suggester is specified by other API calls.
    // The suggester for the hotel database is called "sg", and simply searches the hotel name.
    DocumentSuggestResult<Hotel> suggestResult = await _indexClient.Documents.SuggestAsync<Hotel>
(term, "sg", parameters);

    // Convert the suggest query results to a list that can be displayed in the client.
    List<string> suggestions = suggestResult.Results.Select(x => x.Text).ToList();

    // Return the list of suggestions.
    return new JsonResult(suggestions);
}
```

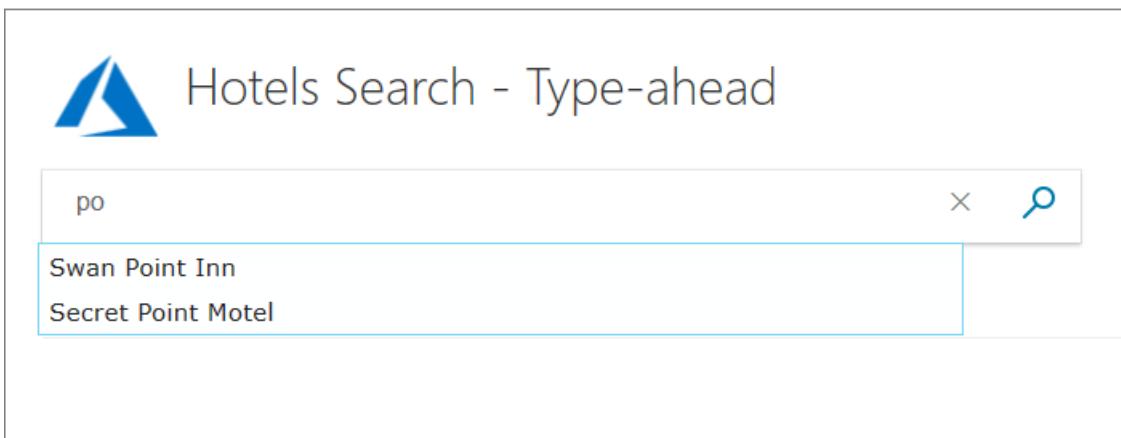
Le paramètre **Top** spécifie le nombre de résultats à retourner (s'il n'est pas spécifié, le nombre par défaut est 5). Un *suggesteur* est spécifié sur l'index Azure, au moment où les données sont configurées et non par une application cliente, comme ce tutoriel. Dans ce cas, le suggesteur est appelé « sg », et il recherche uniquement dans le champ **HotelName**.

La correspondance approximative permet d'inclure les « correspondances proches » dans la sortie, à une distance maximale d'une modification. Si le paramètre **highlights** est défini sur true, des balises HTML en gras sont ajoutées à la sortie. Nous allons définir ces deux paramètres sur true dans la section suivante.

2. Vous pouvez obtenir des erreurs de syntaxe. Si c'est le cas, ajoutez les deux instructions **using** suivantes au début du fichier.

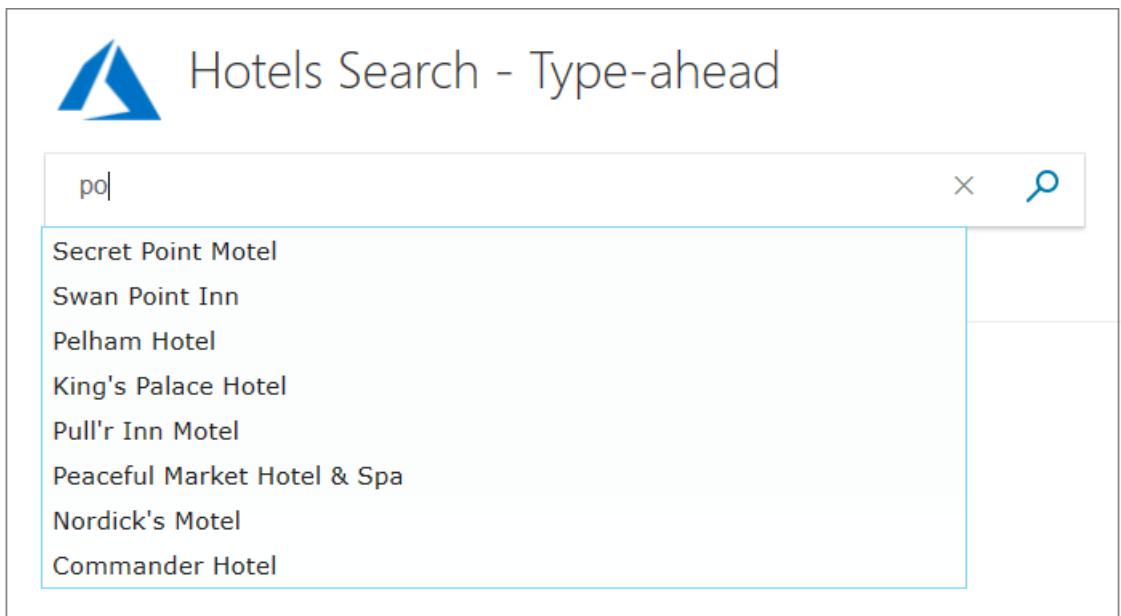
```
using System.Collections.Generic;
using System.Linq;
```

3. Exécutez l'application. Obtenez-vous un éventail d'options quand vous entrez « po », par exemple ? Essayez à présent « pa ».



Notez que les lettres que vous entrez *doivent* commencer un mot et pas simplement être incluses dans le mot.

4. Dans le script de la vue, définissez **&fuzzy** sur true et réexécutez l'application. À présent, entrez « po ». Comme vous pouvez le constater, la recherche part du principe qu'une des lettres tapées peut être incorrecte.



Si cela vous intéresse, la [Syntaxe des requêtes Lucene dans Recherche cognitive Azure](#) décrit la logique

utilisée dans les recherches approximatives.

## Mettre en évidence les suggestions

Nous pouvons améliorer l'apparence des suggestions proposées à l'utilisateur en affectant la valeur true au paramètre **highlights**. Toutefois, nous devons d'abord ajouter du code à la vue pour afficher le texte en gras.

1. Dans la vue (index.cshtml), ajoutez le script suivant après le script **azureautosuggest** que vous avez entré plus haut.

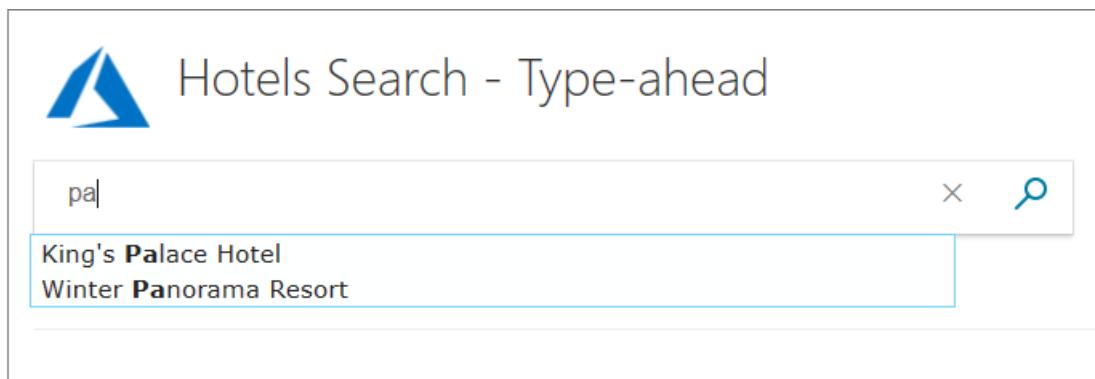
```
<script>
    var updateTextbox = function (event, ui) {
        var result = ui.item.value.replace(/<[^>]+(>|$)/g, "");
        $("#azuresuggesthighlights").val(result);
        return false;
    };

    $("#azuresuggesthighlights").autocomplete({
        html: true,
        source: "/home/suggest?highlights=true&fuzzy=false",
        minLength: 2,
        position: {
            my: "left top",
            at: "left-23 bottom+10"
        },
        select: updateTextbox,
        focus: updateTextbox
    }).data("ui-autocomplete")._renderItem = function (ul, item) {
        return $(<li></li>")
            .data("item.autocomplete", item)
            .append("<a>" + item.label + "</a>")
            .appendTo(ul);
    };
</script>
```

2. À présent, modifiez l'ID de la zone de texte comme suit.

```
@Html.TextBoxFor(m => m.searchText, new { @class = "searchBox", @id = "azuresuggesthighlights" })
<input value="" class="searchBoxSubmit" type="submit">
```

3. Réexécutez l'application ; le texte que vous avez entré doit apparaître en gras dans les suggestions. Par exemple, essayez de taper « pa ».



4. La logique utilisée dans le script de mise en évidence ci-dessus n'est pas infaillible. Si vous entrez un terme qui apparaît deux fois dans le même nom, les résultats en gras ne sont pas tout à fait ce que vous souhaiteriez. Essayez de taper « mo ».

Il appartient au développeur de déterminer si un script fonctionne « suffisamment bien » ou s'il doit faire

l'objet d'ajustements. Nous n'allons pas approfondir la mise en évidence dans ce tutoriel, mais il peut être important de trouver un algorithme précis si la mise en évidence n'est pas efficace pour vos données. Pour plus d'informations, consultez [Mise en surbrillance des correspondances](#).

## Ajouter l'autocomplétion

Une autre variante, qui diffère légèrement des suggestions, est l'autocomplétion, qui permet d'entrer automatiquement un terme de requête. Là encore, nous allons commencer par l'implémentation la plus simple, avant d'améliorer l'expérience utilisateur.

1. Entrez le script suivant dans la vue, après vos scripts précédents.

```
<script>
    $("#azureautocompletebasic").autocomplete({
        source: "/Home/Autocomplete",
        minLength: 2,
        position: {
            my: "left top",
            at: "left-23 bottom+10"
        }
    });
</script>
```

2. À présent, changez l'ID de la zone de texte comme suit.

```
@Html.TextBoxFor(m => m.searchText, new { @class = "searchBox", @id = "azureautocompletebasic" })
<input value="" class="searchBoxSubmit" type="submit">
```

3. Dans le contrôleur Home, nous devons entrer l'action **Autocomplete**, par exemple, sous l'action **Suggest**.

```
public async Task<ActionResult> Autocomplete(string term)
{
    InitSearch();

    // Setup the autocomplete parameters.
    var ap = new AutocompleteParameters()
    {
        AutocompleteMode = AutocompleteMode.OneTermWithContext,
        Top = 6
    };
    AutocompleteResult autocompleteResult = await _indexClient.Documents.AutocompleteAsync(term,
    "sg", ap);

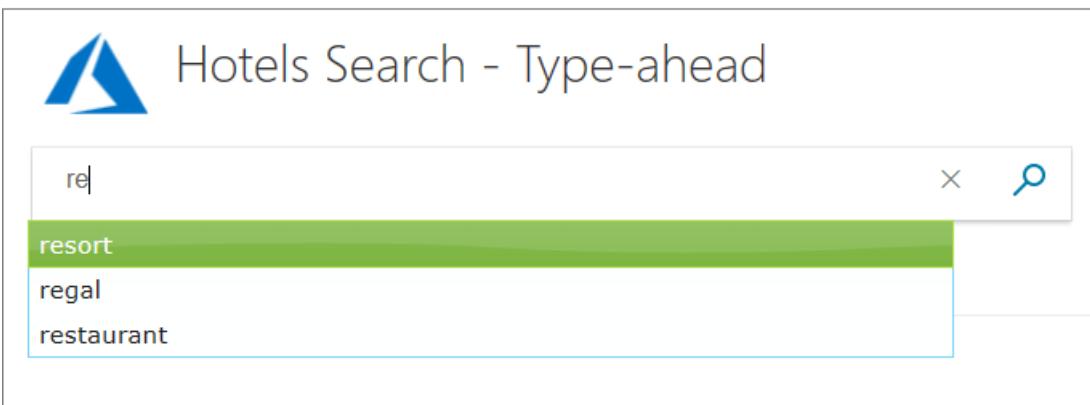
    // Convert the results to a list that can be displayed in the client.
    List<string> autocomplete = autocompleteResult.Results.Select(x => x.Text).ToList();

    // Return the list.
    return new JsonResult(autocomplete);
}
```

Notez que nous utilisons la même fonction de *suggesteur*, appelée « sg », dans la recherche avec auto-complétion, que pour les suggestions (l'auto-complétion ne porte donc que sur les noms d'hôtel).

Parmi les paramètres **AutocompleteMode** disponibles, nous utilisons **OneTermWithContext**. Pour obtenir une description des options supplémentaires, consultez [API Autocomplete](#).

4. Exécutez l'application. Notez que la plage d'options affichées dans la liste déroulante est constituée de mots uniques. Essayez de taper des mots commençant par « re ». Comme vous pouvez le constater, le nombre d'options diminue à mesure que vous tapez des lettres.



Tel quel, le script de suggestions que vous avez exécuté est probablement plus utile que ce script d'auto-complétion. Pour rendre l'auto-complétion plus conviviale, il convient de l'ajouter à la recherche de suggestions.

## Combiner l'auto-complétion et les suggestions

La combinaison de l'auto-complétion et des suggestions est l'option la plus complexe et fournit probablement la meilleure expérience utilisateur. Nous voulons afficher, conjointement avec le texte en cours de frappe, le premier choix de Recherche cognitive Azure pour compléter le texte automatiquement. De plus, nous voulons proposer une plage de suggestions sous forme de liste déroulante.

Il existe des bibliothèques qui offrent cette fonctionnalité, souvent appelée « auto-complétion inline » ou nom similaire. Toutefois, nous allons implémenter cette fonctionnalité en mode natif afin que vous puissiez voir ce qui se passe. Dans cet exemple, nous allons d'abord nous pencher sur le contrôleur.

1. Nous devons ajouter au contrôleur une action qui retourne simplement un résultat d'auto-complétion ainsi qu'un nombre spécifié de suggestions. Nous appellerons cette action **AutocompleteAndSuggest**. Dans le contrôleur Home, ajoutez l'action suivante, après vos autres nouvelles actions.

```

public async Task<ActionResult> AutocompleteAndSuggest(string term)
{
    InitSearch();

    // Setup the type-ahead search parameters.
    var ap = new AutocompleteParameters()
    {
        AutocompleteMode = AutocompleteMode.OneTermWithContext,
        Top = 1,
    };
    AutocompleteResult autocompleteResult = await _indexClient.Documents.AutocompleteAsync(term,
    "sg", ap);

    // Setup the suggest search parameters.
    var sp = new SuggestParameters()
    {
        Top = 8,
    };

    // Only one suggester can be specified per index. The name of the suggester is set when the
    suggester is specified by other API calls.
    // The suggester for the hotel database is called "sg", and it searches only the hotel name.
    DocumentSuggestResult<Hotel> suggestResult = await _indexClient.Documents.SuggestAsync<Hotel>
    (term, "sg", sp);

    // Create an empty list.
    var results = new List<string>();

    if (autocompleteResult.Results.Count > 0)
    {
        // Add the top result for type-ahead.
        results.Add(autocompleteResult.Results[0].Text);
    }
    else
    {
        // There were no type-ahead suggestions, so add an empty string.
        results.Add("");
    }
    for (int n = 0; n < suggestResult.Results.Count; n++)
    {
        // Now add the suggestions.
        results.Add(suggestResult.Results[n].Text);
    }

    // Return the list.
    return new JsonResult(results);
}

```

Une option d'auto-complétion est retournée en haut de la liste **results**, suivie de toutes les suggestions.

2. Dans la vue, nous implémentons d'abord une astuce qui permet d'afficher un mot d'auto-complétion en gris clair juste au-dessous du texte plus foncé entré par l'utilisateur. HTML inclut un positionnement relatif à cet effet. Modifiez comme suit l'instruction **TextBoxFor** (et les instructions **<div>** qui l'entourent) ; l'opération consiste à placer une deuxième zone de recherche nommée **underneath** juste en dessous de notre zone de recherche normale, en décalant celle-ci vers le haut de 39 pixels par rapport à son emplacement par défaut !

```

<div id="underneath" class="searchBox" style="position: relative; left: 0; top: 0">
</div>

<div id="searchinput" class="searchBoxForm" style="position: relative; left: 0; top: -39px">
    @Html.TextBoxFor(m => m.searchText, new { @class = "searchBox", @id = "azureautocomplete" }) <input
    value="" class="searchBoxSubmit" type="submit">
</div>

```

Notez que nous rechargeons l'ID, en **azureautocomplete** dans ce cas.

- Également dans la vue, entrez le script suivant, après tous les scripts que vous avez entrés jusqu'à présent. Il est assez fourni.

```

<script>
    $('#azureautocomplete').autocomplete({
        delay: 500,
        minLength: 2,
        position: {
            my: "left top",
            at: "left-23 bottom+10"
        },

        // Use Ajax to set up a "success" function.
        source: function (request, response) {
            var controllerUrl = "/Home/AutoCompleteAndSuggest?term=" + $("#azureautocomplete").val();
            $.ajax({
                url: controllerUrl,
                dataType: "json",
                success: function (data) {
                    if (data && data.length > 0) {

                        // Show the autocomplete suggestion.
                        document.getElementById("underneath").innerHTML = data[0];

                        // Remove the top suggestion as it is used for inline autocomplete.
                        var array = new Array();
                        for (var n = 1; n < data.length; n++) {
                            array[n - 1] = data[n];
                        }

                        // Show the drop-down list of suggestions.
                        response(array);
                    } else {

                        // Nothing is returned, so clear the autocomplete suggestion.
                        document.getElementById("underneath").innerHTML = "";
                    }
                }
            });
        }
    });

    // Complete on TAB.
    // Clear on ESC.
    // Clear if backspace to less than 2 characters.
    // Clear if any arrow key hit as user is navigating the suggestions.
    $("#azureautocomplete").keydown(function (evt) {

        var suggestedText = document.getElementById("underneath").innerHTML;
        if (evt.keyCode === 9 /* TAB */ && suggestedText.length > 0) {
            $("#azureautocomplete").val(suggestedText);
            return false;
        } else if (evt.keyCode === 27 /* ESC */) {
            document.getElementById("underneath").innerHTML = "";
            $("#azureautocomplete").val("");
        } else if (evt.keyCode === 8 /* Backspace */) {

```

```

        , close it. (evt.keyCode == 8, backspace, etc)
        if ($("#azureautocomplete").val().length < 2) {
            document.getElementById("underneath").innerHTML = "";
        }
    } else if (evt.keyCode >= 37 && evt.keyCode <= 40 /* Any arrow key */) {
        document.getElementById("underneath").innerHTML = "";
    }
});

// Character replace function.
function setCharAt(str, index, chr) {
    if (index > str.length - 1) return str;
    return str.substr(0, index) + chr + str.substr(index + 1);
}

// This function is needed to clear the "underneath" text when the user clicks on a suggestion, and
to
// correct the case of the autocomplete option when it does not match the case of the user input.
// The interval function is activated with the input, blur, change, or focus events.
$("#azureautocomplete").on("input blur change focus", function (e) {

    // Set a 2 second interval duration.
    var intervalDuration = 2000,
        interval = setInterval(function () {

        // Compare the autocorrect suggestion with the actual typed string.
        var inputText = document.getElementById("azureautocomplete").value;
        var autoText = document.getElementById("underneath").innerHTML;

        // If the typed string is longer than the suggestion, then clear the suggestion.
        if (inputText.length > autoText.length) {
            document.getElementById("underneath").innerHTML = "";
        } else {

            // If the strings match, change the case of the suggestion to match the case of the
            typed input.
            if (autoText.toLowerCase().startsWith(inputText.toLowerCase())) {
                for (var n = 0; n < inputText.length; n++) {
                    autoText = setCharAt(autoText, n, inputText[n]);
                }
                document.getElementById("underneath").innerHTML = autoText;
            } else {
                // The strings do not match, so clear the suggestion.
                document.getElementById("underneath").innerHTML = "";
            }
        }

        // If the element loses focus, stop the interval checking.
        if (!$input.is(':focus')) clearInterval(interval);

    }, intervalDuration);
});
</script>

```

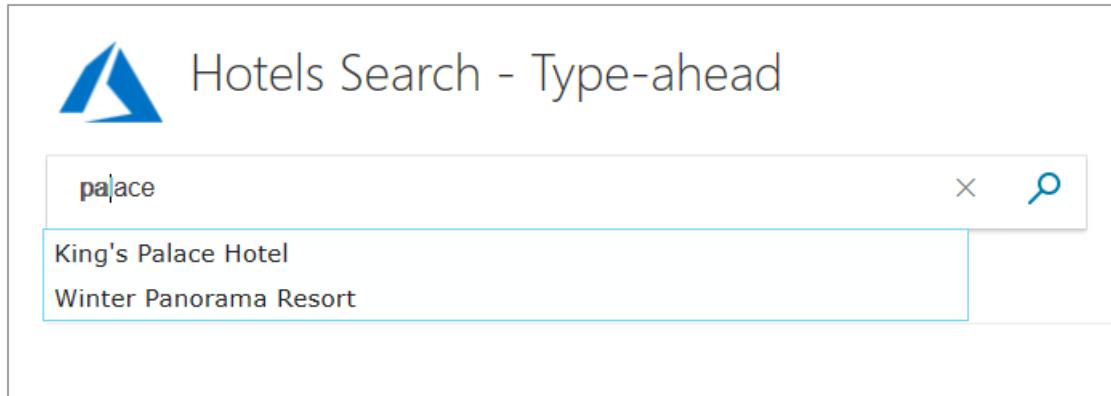
Remarquez l'utilisation intelligente de la fonction **interval** pour effacer le texte sous-jacent quand il ne correspond plus à ce que tape l'utilisateur et définir la même casse (lettres majuscules ou lettres minuscules) que ce que tape l'utilisateur (« pa » pouvant correspondre à « PA », « pA » et « Pa » pendant la recherche), afin que le texte superposé soit net.

Lisez les commentaires dans le script pour mieux comprendre ce dernier.

4. Enfin, nous devons apporter un ajustement mineur à deux classes HTML pour les rendre transparentes. Ajoutez la ligne suivante aux classes **searchBoxForm** et **searchBox** dans le fichier hotels.css.

```
background: rgba(0,0,0,0);
```

5. À présent, exécutez l'application. Entrez « pa » dans la zone de recherche. Obtenez-vous « palace » en tant que suggestion d'auto-complétion ainsi que deux noms d'hôtels contenant « pa » ?



6. Essayez d'accepter la suggestion d'auto-complétion à l'aide de la touche Tab et de sélectionner des suggestions à l'aide des touches de direction et de la touche Tab. Renouvez l'opération à l'aide de la souris et d'un simple clic. Vérifiez que le script gère parfaitement toutes ces situations.

Il peut vous sembler plus simple de charger une bibliothèque qui offre cette fonctionnalité, mais vous connaissez désormais au moins une façon de faire fonctionner l'auto-complétion inline !

## Éléments importants à retenir

Retenez les points importants suivants de ce projet :

- L'auto-complétion et les suggestions peuvent permettre à l'utilisateur de trouver exactement ce qu'il souhaite au moyen de quelques frappes de clavier.
- Combinées, l'auto-complétion et les suggestions peuvent offrir une expérience utilisateur complète.
- Testez toujours les fonctions d'auto-complétion avec toutes les formes d'entrée.
- L'utilisation de la fonction `setInterval` peut être utile pour vérifier et corriger les éléments d'interface utilisateur.

## Étapes suivantes

Dans le tutoriel suivant, nous abordons une autre façon d'améliorer l'expérience utilisateur, consistant à utiliser des facettes pour affiner les recherches au moyen d'un simple clic.

[Tutoriel C# : Utiliser des facettes pour faciliter la navigation - Recherche cognitive Azure](#)

# Tutoriel : Ajouter la navigation à facettes à l'aide du SDK .NET

04/10/2020 • 15 minutes to read • [Edit Online](#)

Les facettes facilitent la navigation, en fournissant à l'utilisateur un ensemble de liens qui l'aident à cibler sa recherche. Les facettes sont des attributs des données (par exemple, la catégorie, ou une fonctionnalité spécifique, d'un hôtel dans nos exemples de données).

Ce tutoriel repose sur le projet de pagination créé dans le [Tutoriel C# : Pagination des résultats de la recherche - Recherche cognitive Azure](#).

Dans ce tutoriel, vous allez apprendre à :

- Définir des propriétés de modèle en tant que *IsFacetable*
- Ajouter la navigation à facettes à votre application

## Prérequis

Pour suivre ce didacticiel, vous devez effectuer les opérations suivantes :

Disposer pleinement du projet [Tutoriel C# : Pagination des résultats de la recherche - Recherche cognitive Azure](#). Ce projet peut être votre propre version ou vous pouvez l'installer à partir de GitHub : [Créer votre première application](#).

## Définir les propriétés de modèle en tant que **IsFacetable**

Pour qu'une propriété de modèle puisse être trouvée dans une recherche à facettes, elle doit être étiquetée avec **IsFacetable**.

1. Examinez la classe **Hotel**. **Category** et **Tags**, par exemple, sont marqués comme **IsFacetable**, contrairement à **HotelName** et **Description**.

```

public partial class Hotel
{
    [System.ComponentModel.DataAnnotations.Key]
    [IsFilterable]
    public string HotelId { get; set; }

    [IsSearchable, IsSortable]
    public string HotelName { get; set; }

    [IsSearchable]
    [Analyzer(AnalyzerName.AsString.EnLucene)]
    public string Description { get; set; }

    [IsSearchable]
    [Analyzer(AnalyzerName.AsString.FrLucene)]
    [JsonProperty("Description_fr")]
    public string DescriptionFr { get; set; }

    [IsSearchable, IsFilterable, IsSortable, IsFacetable]
    public string Category { get; set; }

    [IsSearchable, IsFilterable, IsFacetable]
    public string[] Tags { get; set; }

    [IsFilterable, IsSortable, IsFacetable]
    public bool? ParkingIncluded { get; set; }

    [IsFilterable, IsSortable, IsFacetable]
    public DateTimeOffset? LastRenovationDate { get; set; }

    [IsFilterable, IsSortable, IsFacetable]
    public double? Rating { get; set; }

    public Address Address { get; set; }

    [IsFilterable, IsSortable]
    public GeographyPoint Location { get; set; }

    public Room[] Rooms { get; set; }
}

```

2. Comme nous n'allons pas changer d'étiquettes dans le cadre de ce tutoriel, fermez le fichier hotel.cs sans le modifier.

**NOTE**

Une recherche à facettes génère une erreur si un champ demandé dans la recherche n'est pas correctement étiqueté.

## Ajouter la navigation à facettes à votre application

Pour cet exemple, nous allons permettre à l'utilisateur de sélectionner une catégorie d'hôtels, ou une commodité, dans des listes de liens affichés à gauche des résultats. L'utilisateur commence par entrer du texte de recherche, puis peut affiner les résultats de la recherche en sélectionnant successivement une catégorie et une commodité (il peut aussi sélectionner la commodité en premier, l'ordre n'étant pas important).

Le contrôleur doit passer les listes de facettes à la vue. Nous devons conserver les sélections de l'utilisateur à mesure que la recherche progresse ; là encore, nous utilisons un stockage temporaire comme mécanisme de conservation des données.



wifi

**Category:**

Budget (5)  
Luxury (5)  
Resort and Spa (5)  
Boutique (3)  
Extended-Stay (1)

**Amenities:**

free wifi (11)  
laundry service (6)  
24-hour front desk service (5)  
pool (5)  
restaurant (5)  
concierge (4)  
continental breakfast (4)  
free parking (4)  
view (4)  
coffee in lobby (3)  
air conditioning (2)  
bar (1)

19 Results

**Super Deluxe Inn & Suites**

Complimentary Airport Shuttle & WiFi. Book Now and save - Spacious All Suite Hotel, Indoor/Outdoor Pool, Fitness Center, Florida Green certified, Starbucks Coffee, HDTV

Category: Boutique

Amenities: bar, free wifi, free wifi

**Double Sanctuary Resort**

5\* Luxury Hotel - Biggest Rooms in the city. #1 Hotel in the area listed by Conde Nast Traveler. Free WiFi, Flexible check in/out, Fitness Center & Nespresso in room.

Category: Resort and Spa

Amenities: view, laundry service, free wifi

**Pull'r Inn Motel**

The hotel rooms and suites offer the perfect blend of beauty and elegance. Our rooms will elevate your stay, whether you're traveling for business, celebrating a honeymoon, or just looking for a remarkable getaway. With views of the valley or the iconic fountains right from your suite, your stay will be nothing short of unforgettable.

|< < 1 2 3 4 5 > >|

**Ajouter des chaînes de filtre au modèle SearchData**

- Ouvrez le fichier `SearchData.cs` et ajoutez des propriétés de chaîne à la classe `SearchData` destinées à contenir les chaînes de filtre des facettes.

```
public string categoryFilter { get; set; }
public string amenityFilter { get; set; }
```

**Ajouter la méthode d'action Facet**

Le contrôleur Home a besoin d'une nouvelle action, `Facet`, et de mises à jour pour ses actions `Index` et `Page` existantes ainsi que pour la méthode `RunQueryAsync`.

- Ouvrez le fichier du contrôleur Home et ajoutez l'instruction `using` suivante pour activer la construction `List<string>`.

```
using System.Collections.Generic;
```

- Remplacez la méthode d'action `Index(SearchData model)`.

```
public async Task<ActionResult> Index(SearchData model)
{
    try
    {
        // Ensure the search string is valid.
        if (model.searchText == null)
        {
            model.searchText = "";
        }

        // Make the search call for the first page.
        await RunQueryAsync(model, 0, 0, "", "");
    }

    catch
    {
        return View("Error", new ErrorViewModel { RequestId = "1" });
    }
    return View(model);
}
```

3. Remplacez la méthode d'action **Page(SearchData model)** .

```

public async Task<ActionResult> Page(SearchData model)
{
    try
    {
        int page;

        // Calculate the page that should be displayed.
        switch (model.paging)
        {
            case "prev":
                page = (int)TempData["page"] - 1;
                break;

            case "next":
                page = (int)TempData["page"] + 1;
                break;

            default:
                page = int.Parse(model.paging);
                break;
        }

        // Recover the leftMostPage.
        int leftMostPage = (int)TempData["leftMostPage"];

        // Recover the filters.
        string catFilter = TempData["categoryFilter"].ToString();
        string ameFilter = TempData["amenityFilter"].ToString();

        // Recover the search text.
        model.searchText = TempData["searchfor"].ToString();

        // Search for the new page.
        await RunQueryAsync(model, page, leftMostPage, catFilter, ameFilter);
    }

    catch
    {
        return View("Error", new ErrorViewModel { RequestId = "2" });
    }
    return View("Index", model);
}

```

- Ajoutez un méthode d'action **Facet(SearchData model)** à activer quand l'utilisateur clique sur un lien à facette. Le modèle contient un filtre de recherche de catégorie ou un filtre de recherche de commodité. Ajoutez-le éventuellement après l'action **Page**.

```

public async Task<ActionResult> Facet(SearchData model)
{
    try
    {
        // Filters set by the model override those stored in temporary data.
        string catFilter;
        string ameFilter;
        if (model.categoryFilter != null)
        {
            catFilter = model.categoryFilter;
        } else
        {
            catFilter = TempData["categoryFilter"].ToString();
        }

        if (model.amenityFilter != null)
        {
            ameFilter = model.amenityFilter;
        } else
        {
            ameFilter = TempData["amenityFilter"].ToString();
        }

        // Recover the search text.
        model.searchText = TempData["searchfor"].ToString();

        // Initiate a new search.
        await RunQueryAsync(model, 0, 0, catFilter, ameFilter);
    }

    catch
    {
        return View("Error", new ErrorViewModel { RequestId = "2" });
    }
    return View("Index", model);
}

```

## Configurer le filtre de recherche

Quand un utilisateur sélectionne une facette, il peut, par exemple, cliquer sur la catégorie **Resort and Spa** ; dans ce cas, seuls des hôtels spécifiés comme appartenant à cette catégorie doivent être retournés dans les résultats. Pour affiner une recherche de cette façon, nous devons configurer un *filtre*.

1. Remplacez la méthode **RunQueryAsync** par le code suivant. En gros, elle prend une chaîne de filtre de catégories et une chaîne de filtre de commodités et définit le paramètre **Filter** de **SearchParameters**.

```

private async Task<ActionResult> RunQueryAsync(SearchData model, int page, int leftMostPage, string
catFilter, string ameFilter)
{
    InitSearch();

    string facetFilter = "";

    if (catFilter.Length > 0 && ameFilter.Length > 0)
    {
        // Both facets apply.
        facetFilter = $"{catFilter} and {ameFilter}";
    } else
    {
        // One, or zero, facets apply.
        facetFilter = $"{catFilter}{ameFilter}";
    }

    var parameters = new SearchParameters
    {
        ...
    };

```

```

        Filter = facetFilter,
        // Return information on the text, and number, of facets in the data.
        Facets = new List<string> { "Category,count:20", "Tags,count:20" },
        // Enter Hotel property names into this list, so only these values will be returned.
        Select = new[] { "HotelName", "Description", "Category", "Tags" },
        SearchMode = SearchMode.All,
        // Skip past results that have already been returned.
        Skip = page * GlobalVariables.ResultsPerPage,
        // Take only the next page worth of results.
        Top = GlobalVariables.ResultsPerPage,
        // Include the total number of results.
        IncludeTotalResultCount = true,
    };

    // For efficiency, the search call should be asynchronous, so use SearchAsync rather than
    Search.
    model.resultList = await _indexClient.Documents.SearchAsync<Hotel>(model.searchText,
parameters);

    // This variable communicates the total number of pages to the view.
    model.pageCount = ((int)model.resultList.Count + GlobalVariables.ResultsPerPage - 1) /
GlobalVariables.ResultsPerPage;

    // This variable communicates the page number being displayed to the view.
    model.currentPage = page;

    // Calculate the range of page numbers to display.
    if (page == 0)
    {
        leftMostPage = 0;
    }
    else
        if (page <= leftMostPage)
    {
        // Trigger a switch to a lower page range.
        leftMostPage = Math.Max(page - GlobalVariables.PageRangeDelta, 0);
    }
    else
        if (page >= leftMostPage + GlobalVariables.MaxPageRange - 1)
    {
        // Trigger a switch to a higher page range.
        leftMostPage = Math.Min(page - GlobalVariables.PageRangeDelta, model.pageCount -
GlobalVariables.MaxPageRange);
    }
    model.leftMostPage = leftMostPage;

    // Calculate the number of page numbers to display.
    model.pageRange = Math.Min(model.pageCount - leftMostPage, GlobalVariables.MaxPageRange);

    // Ensure Temp data is stored for the next call.
    TempData["page"] = page;
    TempData["leftMostPage"] = model.leftMostPage;
    TempData["searchfor"] = model.searchText;
    TempData["categoryFilter"] = catFilter;
    TempData["amenityFilter"] = ameFilter;

    // Return the new view.
    return View("Index", model);
}

```

Nous avons ajouté les propriétés **Category** et **Tags** à la liste des éléments **Select** à retourner. Cet ajout n'est

pas obligatoire pour que la navigation à facettes fonctionne, mais nous utilisons ces informations pour vérifier que le filtrage est correct.

## Ajouter des listes de liens à facette à la vue

La vue nécessite certaines modifications significatives.

1. Commencez par ouvrir le fichier hotels.css (dans le dossier wwwroot/css) et ajoutez les classes suivantes.

```
.facetlist {
    list-style: none;
}

.facetchecks {
    width: 250px;
    display: normal;
    color: #666;
    margin: 10px;
    padding: 5px;
}

.facetheader {
    font-size: 10pt;
    font-weight: bold;
    color: darkgreen;
}
```

2. Pour la vue, nous organisons la sortie sous la forme d'un tableau, afin d'aligner correctement les listes de facettes sur la gauche et les résultats sur la droite. Ouvrez le fichier index.cshtml. Remplacez tout le contenu des balises HTML <body> par le code suivant.

```
<body>

@using (Html.BeginForm("Index", "Home", FormMethod.Post))
{
    <table>
        <tr>
            <td></td>
            <td>
                <h1 class="sampleTitle">
                    
                    Hotels Search - Facet Navigation
                </h1>
            </td>
        </tr>

        <tr>
            <td></td>
            <td>
                <!-- Display the search text box, with the search icon to the right of it.-->
                <div class="searchBoxForm">
                    @Html.TextBoxFor(m => m.searchText, new { @class = "searchBox" }) <input value="" 
class="searchBoxSubmit" type="submit">
                </div>
            </td>
        </tr>

        <tr>
            <td valign="top">
                <div id="facetplace" class="facetchecks">

                    @if (Model != null && Model.resultList != null)
                    {
                        List<string> categories = Model.resultList.Facets["Category"].Select(x =>
x.Value.ToString()).ToList();
                    }
                </div>
            </td>
        </tr>
    </table>
}
```

```

        if (categories.Count > 0)
        {
            <h5 class="facethader">Category:</h5>
            <ul class="facetlist">
                @for (var c = 0; c < categories.Count; c++)
                {
                    var facetLink = $"{categories[c]}"
({Model.resultList.Facets["Category"]}[c].Count}");
                    <li>
                        @Html.ActionLink(facetLink, "Facet", "Home", new {
categoryFilter = $"Category eq '{categories[c]}'" }, null)
                    </li>
                }
            </ul>
        }

        List<string> tags = Model.resultList.Facets["Tags"].Select(x =>
x.Value.ToString()).ToList();

        if (tags.Count > 0)
        {
            <h5 class="facethader">Amenities:</h5>
            <ul class="facetlist">
                @for (var c = 0; c < tags.Count; c++)
                {
                    var facetLink = $"{tags[c]} ({Model.resultList.Facets["Tags"]}[c].Count}";
                    <li>
                        @Html.ActionLink(facetLink, "Facet", "Home", new { amenityFilter
= $"Tags/any(t: t eq '{tags[c]}')" }, null)
                    </li>
                }
            </ul>
        }
    </div>
</td>
<td valign="top">
    <div id="resultsplace">
        @if (Model != null && Model.resultList != null)
        {
            // Show the result count.
            <p class="sampleText">
                @Html.DisplayFor(m => m.resultList.Count) Results
            </p>

            @for (var i = 0; i < Model.resultList.Results.Count; i++)
            {
                string amenities = string.Join(", ",
Model.resultList.Results[i].Document.Tags);

                string fullDescription = Model.resultList.Results[i].Document.Description;
                fullDescription += $"{Category:
{Model.resultList.Results[i].Document.Category}}";
                fullDescription += $"{\nAmenities: {amenities}}";

                // Display the hotel name and description.
                @Html.TextAreaFor(m => Model.resultList.Results[i].Document.HotelName, new {
@class = "box1" })
                @Html.TextArea($"desc{i}", fullDescription, new { @class = "box2" })
            }
        }
    </div>
</td>
</tr>
<tr>
    <td></td>

```

```

<td valign="top">
    @if (Model != null && Model.pageCount > 1)
    {
        // If there is more than one page of results, show the paging buttons.
        <table>
            <tr>
                <td class="tdPage">
                    @if (Model.currentPage > 0)
                    {
                        <p class="pageButton">
                            @Html.ActionLink("|<", "Page", "Home", new { paging = "0" },
null)
                        </p>
                    }
                    else
                    {
                        <p class="pageButtonDisabled">|&lt;</p>
                    }
                </td>

                <td class="tdPage">
                    @if (Model.currentPage > 0)
                    {
                        <p class="pageButton">
                            @Html.ActionLink("<", "Page", "Home", new { paging = "prev" },
null)
                        </p>
                    }
                    else
                    {
                        <p class="pageButtonDisabled">&lt;</p>
                    }
                </td>

                @for (var pn = Model.leftMostPage; pn < Model.leftMostPage +
Model.pageRange; pn++)
                {
                    <td class="tdPage">
                        @if (Model.currentPage == pn)
                        {
                            // Convert displayed page numbers to 1-based and not 0-based.
                            <p class="pageSelected">@(pn + 1)</p>
                        }
                        else
                        {
                            <p class="pageButton">
                                @Html.ActionLink((pn + 1).ToString(), "Page", "Home", new {
paging = @pn }, null)
                            </p>
                        }
                    </td>
                }

                <td class="tdPage">
                    @if (Model.currentPage < Model.pageCount - 1)
                    {
                        <p class="pageButton">
                            @Html.ActionLink(">", "Page", "Home", new { paging = "next" },
null)
                        </p>
                    }
                    else
                    {
                        <p class="pageButtonDisabled">&gt;</p>
                    }
                </td>

                <td class="tdPage">
                    @if (Model.currentPage < Model.pageCount - 1)

```

```

        {
            <p class="pageButton">
                @Html.ActionLink(">|", "Page", "Home", new { paging =
Model.pageCount - 1 }, null)
            </p>
        }
        else
        {
            <p class="pageButtonDisabled">&gt;|</p>
        }
    </td>
</tr>
</table>
}
</td>
</tr>
</table>
}
</body>

```

Remarquez l'utilisation de l'appel `Html.ActionLink`. Cet appel communique les chaînes de filtre valides au contrôleur quand l'utilisateur clique sur un lien à facette.

## Exécuter et tester l'application

La navigation à facettes permet à l'utilisateur d'affiner les recherches d'un simple clic, comme l'illustre la séquence suivante.

1. Exécutez l'application, puis tapez « airport » comme texte de recherche. Vérifiez que la liste de facettes apparaît clairement à gauche. Ces facettes sont toutes celles qui s'appliquent aux hôtels dont le descriptif comporte le terme « airport », et chacune est accompagnée de son nombre d'occurrences.

Hotels Search - Facet Navigation

airport

**Category:**

- Luxury (4)
- Resort and Spa (4)
- Boutique (1)

**Amenities:**

- continental breakfast (5)
- 24-hour front desk service (4)
- free parking (3)
- free wifi (3)
- view (2)
- air conditioning (1)
- bar (1)
- coffee in lobby (1)
- concierge (1)
- laundry service (1)
- pool (1)
- restaurant (1)

9 Results

**Veteran Right Track**  
Free Shuttle to the airport and casinos. Free breakfast and WiFi.  
Category: Resort and Spa  
Amenities: 24-hour front desk service, restaurant, concierge

**Nova Hotel & Spa**  
1 Mile from the airport. Free WiFi, Outdoor Pool, Complimentary Airport Shuttle, 6 miles from the beach & 10 miles from downtown.  
Category: Resort and Spa  
Amenities: pool, continental breakfast, free parking

**Suites At Bellevue Square**  
Luxury at the mall. Located across the street from the Light Rail to downtown. Free shuttle to the mall and airport.  
Category: Resort and Spa  
Amenities: continental breakfast, air conditioning, 24-hour front desk service

|< < 1 2 3 > >|

2. Cliquez sur la catégorie **Resort and Spa**. Vérifiez que tous les résultats appartiennent à cette catégorie.



airport

**Category:**

Resort and Spa (4)

**Amenities:**

- 24-hour front desk service (2)
- continental breakfast (2)
- air conditioning (1)
- coffee in lobby (1)
- concierge (1)
- free parking (1)
- laundry service (1)
- pool (1)
- restaurant (1)
- view (1)

4 Results

**Veteran Right Track**

Free Shuttle to the airport and casinos. Free breakfast and WiFi.

Category: Resort and Spa

Amenities: 24-hour front desk service, restaurant, concierge

**Nova Hotel & Spa**

1 Mile from the airport. Free WiFi, Outdoor Pool, Complimentary Airport Shuttle, 6 miles from the beach &amp; 10 miles from downtown.

Category: Resort and Spa

Amenities: pool, continental breakfast, free parking

**Suites At Bellevue Square**

Luxury at the mall. Located across the street from the Light Rail to downtown. Free shuttle to the mall and airport.

Category: Resort and Spa

Amenities: continental breakfast, air conditioning, 24-hour front desk service

|&lt; &lt; 1 2 &gt; &gt;|

3. Cliquez sur la commodité **continental breakfast**. Vérifiez que tous les résultats appartiennent toujours à la catégorie « **Resort and Spa** » et correspondent à la commodité sélectionnée.



airport

**Category:**

Resort and Spa (2)

**Amenities:**

- continental breakfast (2)
- 24-hour front desk service (1)
- air conditioning (1)
- free parking (1)
- pool (1)

2 Results

**Nova Hotel & Spa**

1 Mile from the airport. Free WiFi, Outdoor Pool, Complimentary Airport Shuttle, 6 miles from the beach &amp; 10 miles from downtown.

Category: Resort and Spa

Amenities: pool, continental breakfast, free parking

**Suites At Bellevue Square**

Luxury at the mall. Located across the street from the Light Rail to downtown. Free shuttle to the mall and airport.

Category: Resort and Spa

Amenities: continental breakfast, air conditioning, 24-hour front desk service

4. Essayez de sélectionner une autre catégorie, puis une commodité, et observez la façon dont les résultats sont affinés. Ensuite, faites l'opération inverse, en choisissant d'abord une commodité, puis une catégorie.

**NOTE**

Quand une sélection est effectuée dans une liste de facettes (par exemple, une catégorie), elle remplace toute sélection antérieure dans la liste des catégories.

## Éléments importants à retenir

Retenez les points importants suivants de ce projet :

- Il est impératif de marquer chaque propriété comme `IsFacetable` si elle doit être incluse dans la navigation à facettes.
- La navigation à facettes fournit à un utilisateur un moyen facile et intuitif d'affiner une recherche.
- Il convient d'organiser la navigation à facettes en sections (catégories d'hôtel, commodités d'un hôtel, gammes de prix, plages d'évaluation, etc.) portant chacune un titre approprié.

## Étapes suivantes

Dans le tutoriel suivant, nous examinons le classement des résultats. À ce stade, les résultats sont simplement triés dans l'ordre dans lequel ils sont stockés dans la base de données.

[Tutoriel C# : Classer les résultats - Recherche cognitive Azure](#)

# Tutoriel : Trier des résultats de recherche à l'aide du SDK .NET

04/10/2020 • 36 minutes to read • [Edit Online](#)

Jusqu'à présent, dans notre série de tutoriels, les résultats sont retournés et affichés dans un ordre par défaut. Il peut s'agir de l'ordre dans lequel se trouvent les données ou bien dépendant d'un éventuel *profil de score* qui a été défini et qui est utilisé quand aucun paramètre de classement n'est spécifié. Dans ce tutoriel, nous allons découvrir comment classer les résultats en fonction d'une propriété principale puis, pour les résultats qui ont la même propriété principale, comment classer cette sélection sur une propriété secondaire. Comme alternative au classement sur la base de valeurs numériques, le dernier exemple montre comment classer selon un profil de score personnalisé. Nous allons également explorer plus avant l'affichage de *types complexes*.

Afin de comparer facilement les résultats retournés, ce projet s'appuie sur le projet avec défilement infini créé dans le [Tutoriel C# : Pagination des résultats de la recherche - Recherche cognitive Azure](#).

Dans ce tutoriel, vous allez apprendre à :

- Classer des résultats en fonction d'une seule propriété
- Classer des résultats en fonction de plusieurs propriétés
- Filtrer les résultats en fonction de la distance d'un point géographique
- Classer des résultats en fonction d'un profil de score

## Prérequis

Pour suivre ce didacticiel, vous devez effectuer les opérations suivantes :

Disposer de la version avec défilement infini opérationnelle du [Tutoriel C# : Pagination des résultats de la recherche - Recherche cognitive Azure](#). Ce projet peut être votre propre version ou vous pouvez l'installer à partir de GitHub : [Créer votre première application](#).

## Classer des résultats en fonction d'une seule propriété

Quand nous classons des résultats en fonction d'une seule propriété, par exemple l'évaluation des hôtels, nous voulons non seulement obtenir les résultats classés, mais aussi la confirmation que l'ordre est bien correct. En d'autres termes, si nous classons sur l'évaluation, nous devons afficher l'évaluation dans la vue.

Dans ce tutoriel, nous allons également ajouter un peu plus d'informations à l'affichage des résultats : le prix de la chambre la moins chère et celui de la chambre la plus chère pour chaque hôtel. Comme nous nous intéressons au classement, nous allons également ajouter des valeurs pour garantir que ce sur quoi nous classons figure aussi dans la vue.

Il n'est pas nécessaire de modifier les modèles pour activer le classement. La vue et le contrôleur doivent être mis à jour. Commencez par ouvrir le contrôleur home.

### Ajouter la propriété OrderBy aux paramètres de recherche

1. Pour classer les résultats en fonction d'une seule propriété numérique, il suffit de définir le paramètre **OrderBy** sur le nom de la propriété. Dans la méthode **Index(SearchData model)** , ajoutez la ligne suivante aux paramètres de recherche.

```
OrderBy = new[] { "Rating desc" },
```

#### NOTE

L'ordre par défaut est croissant, même si vous pouvez ajouter **asc** à la propriété pour rendre cela explicite. Vous spécifiez un ordre décroissant en ajoutant **desc**.

2. Maintenant, exécutez l'application et entrez un terme de recherche courant. Les résultats peuvent ou non être dans l'ordre correct, car ni vous en tant que développeur ni l'utilisateur n'avez un moyen simple de vérifier les résultats !
3. Nous allons faire apparaître que les résultats sont classés sur l'évaluation. Remplacez d'abord les classes **box1** et **box2** dans le fichier `hotels.css` par les classes suivantes (ces classes sont les seules nouvelles classes dont nous avons besoin pour ce tutoriel).

```

textarea.box1A {
    width: 324px;
    height: 32px;
    border: none;
    background-color: azure;
    font-size: 14pt;
    color: blue;
    padding-left: 5px;
    text-align: left;
}

textarea.box1B {
    width: 324px;
    height: 32px;
    border: none;
    background-color: azure;
    font-size: 14pt;
    color: blue;
    text-align: right;
    padding-right: 5px;
}

textarea.box2A {
    width: 324px;
    height: 32px;
    border: none;
    background-color: azure;
    font-size: 12pt;
    color: blue;
    padding-left: 5px;
    text-align: left;
}

textarea.box2B {
    width: 324px;
    height: 32px;
    border: none;
    background-color: azure;
    font-size: 12pt;
    color: blue;
    text-align: right;
    padding-right: 5px;
}

textarea.box3 {
    width: 648px;
    height: 100px;
    border: none;
    background-color: azure;
    font-size: 12pt;
    padding-left: 5px;
    margin-bottom: 24px;
}

```

#### TIP

Les navigateurs mettent généralement en cache les fichiers css : ceci peut faire qu'un ancien fichier css est utilisé et que vos modifications sont ignorées. Un bon moyen de contourner ceci est d'ajouter au lien une chaîne de requête avec un paramètre de version. Par exemple :

```
<link rel="stylesheet" href="~/css/hotels.css?v1.1" />
```

Mettez à jour le numéro de version si vous pensez qu'un ancien fichier css est utilisé par votre navigateur.

4. Ajoutez la propriété **Rating** (Évaluation) au paramètre **Select** dans la méthode **Index(SearchData model)**

```
Select = new[] { "HotelName", "Description", "Rating"},
```

5. Ouvrez la vue (index.cshtml) et remplacez la boucle de rendu (`<!-- Show the hotel data. -->`) par le code suivant.

```
<!-- Show the hotel data. -->
@for (var i = 0; i < Model.resultList.Results.Count; i++)
{
    var ratingText = $"Rating: {Model.resultList.Results[i].Document.Rating}";

    // Display the hotel details.
    @Html.TextArea($"name{i}", Model.resultList.Results[i].Document.HotelName, new { @class = "box1A" })
        @Html.TextArea($"rating{i}", ratingText, new { @class = "box1B" })
        @Html.TextArea($"desc{i}", Model.resultList.Results[i].Document.Description, new {
@class = "box3" })
}
```

6. L'évaluation doit être disponible dans la première page affichée et dans les pages suivantes qui sont appelées via le défilement infini. Pour cette dernière situation, nous devons mettre à jour à la fois l'action **Next** dans le contrôleur et la fonction **scrolled** dans la vue. En commençant par le contrôleur, remplacez la méthode **Next** par le code suivant. Ce code crée et communique le texte de l'évaluation.

```
public async Task<ActionResult> Next(SearchData model)
{
    // Set the next page setting, and call the Index(model) action.
    model.paging = "next";
    await Index(model);

    // Create an empty list.
    var nextHotels = new List<string>();

    // Add a hotel details to the list.
    for (int n = 0; n < model.resultList.Results.Count; n++)
    {
        var ratingText = $"Rating: {model.resultList.Results[n].Document.Rating}";

        // Add three strings to the list.
        nextHotels.Add(model.resultList.Results[n].Document.HotelName);
        nextHotels.Add(ratingText);
        nextHotels.Add(model.resultList.Results[n].Document.Description);
    }

    // Rather than return a view, return the list of data.
    return new JsonResult(nextHotels);
}
```

7. Maintenant, mettez à jour la fonction **scrolled** dans la vue, de façon à afficher le texte de l'évaluation.

```

<script>
    function scrolled() {
        if (myDiv.offsetHeight + myDiv.scrollTop >= myDiv.scrollHeight) {
            $.getJSON("/Home/Next", function (data) {
                var div = document.getElementById('myDiv');

                // Append the returned data to the current list of hotels.
                for (var i = 0; i < data.length; i += 3) {
                    div.innerHTML += '\n<textarea class="box1A">' + data[i] + '</textarea>';
                    div.innerHTML += '\n<textarea class="box1B">' + data[i + 1] + '</textarea>';
                    div.innerHTML += '\n<textarea class="box3">' + data[i + 2] + '</textarea>';
                }
            });
        }
    }
</script>

```

8. Vous pouvez maintenant réexécuter l'application. Recherchez un terme courant, comme « wifi », et vérifiez que les résultats sont classés par ordre décroissant de l'évaluation des hôtels.

Hotels Search - Ordering Results

wifi

19 Results

Hotel Name	Rating
Summer Wind Resort	Rating: 4.9
Pull'r Inn Motel	Rating: 4.7
Motteler's Thunderbird Motel	Rating: 4.4

Vous remarquerez que plusieurs hôtels ont une évaluation identique : leur apparition dans l'affichage correspond donc à nouveau à l'ordre dans lequel les données sont trouvées, ce qui est arbitraire.

Avant de passer à l'ajout d'un deuxième niveau de classement, ajoutons du code pour afficher la plage de prix des chambres. Nous ajoutons ce code pour deux raisons : montrer l'extraction de données à partir d'un *type complexe* et présenter le classement des résultats en fonction des prix (peut-être le moins cher en premier).

#### Ajouter la plage de prix des chambres à la vue

1. Ajoutez les propriétés qui contiennent le prix de la chambre la moins chère et la plus chère au modèle

## Hotel.cs.

```
// Room rate range  
public double cheapest { get; set; }  
public double expensive { get; set; }
```

2. Calculez les prix des chambres à la fin de l'action **Index(SearchData model)** dans le contrôleur home. Ajoutez les calculs après le stockage des données temporaires.

```
// Ensure TempData is stored for the next call.  
TempData["page"] = page;  
TempData["searchfor"] = model.searchText;  
  
// Calculate the room rate ranges.  
for (int n = 0; n < model.resultList.Results.Count; n++)  
{  
    // Calculate room rates.  
    var cheapest = 0d;  
    var expensive = 0d;  
  
    for (var r = 0; r < model.resultList.Results[n].Document.Rooms.Length; r++)  
    {  
        var rate = model.resultList.Results[n].Document.Rooms[r].BaseRate;  
        if (rate < cheapest || cheapest == 0)  
        {  
            cheapest = (double)rate;  
        }  
        if (rate > expensive)  
        {  
            expensive = (double)rate;  
        }  
    }  
    model.resultList.Results[n].Document.cheapest = cheapest;  
    model.resultList.Results[n].Document.expensive = expensive;  
}
```

3. Ajoutez la propriété **Rooms** au paramètre **Select** dans la méthode d'action **Index(SearchData model)** du contrôleur.

```
Select = new[] { "HotelName", "Description", "Rating", "Rooms" },
```

4. Modifiez la boucle de rendu de la vue pour afficher la plage des prix pour la première page de résultats.

```
<!-- Show the hotel data. -->  
@for (var i = 0; i < Model.resultList.Results.Count; i++)  
{  
    var rateText = $"Rates from ${Model.resultList.Results[i].Document.cheapest} to  
${Model.resultList.Results[i].Document.expensive}";  
    var ratingText = $"Rating: {Model.resultList.Results[i].Document.Rating}";  
  
    // Display the hotel details.  
    @Html.TextArea($"name{i}", Model.resultList.Results[i].Document.HotelName, new { @class  
= "box1A" })  
    @Html.TextArea($"rating{i}", ratingText, new { @class = "box1B" })  
    @Html.TextArea($"rates{i}" , rateText, new { @class = "box2A" })  
    @Html.TextArea($"desc{i}" , Model.resultList.Results[i].Document.Description, new {  
@class = "box3" })  
}
```

5. Changez la méthode **Next** du contrôleur home de façon à communiquer la plage de prix pour les pages de

résultats suivantes.

```
public async Task<ActionResult> Next(SearchData model)
{
    // Set the next page setting, and call the Index(model) action.
    model.paging = "next";
    await Index(model);

    // Create an empty list.
    var nextHotels = new List<string>();

    // Add a hotel details to the list.
    for (int n = 0; n < model.resultList.Results.Count; n++)
    {
        var ratingText = $"Rating: {model.resultList.Results[n].Document.Rating}";
        var rateText = $"Rates from ${model.resultList.Results[n].Document.cheapest} to
${model.resultList.Results[n].Document.expensive}";

        // Add strings to the list.
        nextHotels.Add(model.resultList.Results[n].Document.HotelName);
        nextHotels.Add(ratingText);
        nextHotels.Add(rateText);
        nextHotels.Add(model.resultList.Results[n].Document.Description);
    }

    // Rather than return a view, return the list of data.
    return new JsonResult(nextHotels);
}
```

6. Mettez à jour la fonction `scrolled` dans la vue, de façon à gérer le texte des prix des chambres.

```
<script>
function scrolled() {
    if (myDiv.offsetHeight + myDiv.scrollTop >= myDiv.scrollHeight) {
        $.getJSON("/Home/Next", function (data) {
            var div = document.getElementById('myDiv');

            // Append the returned data to the current list of hotels.
            for (var i = 0; i < data.length; i += 4) {
                div.innerHTML += '\n<textarea class="box1A">' + data[i] + '</textarea>';
                div.innerHTML += '\n<textarea class="box1B">' + data[i + 1] + '</textarea>';
                div.innerHTML += '\n<textarea class="box2A">' + data[i + 2] + '</textarea>';
                div.innerHTML += '\n<textarea class="box3">' + data[i + 4] + '</textarea>';
            }
        });
    }
}
</script>
```

7. Exécutez l'application et vérifiez que les plages de prix des chambres sont affichés.



## Hotels Search - Ordering Results



16 Results

### Pull'r Inn Motel

Rating: 4.7

Rates from \$64.99 to \$268.99

The hotel rooms and suites offer the perfect blend of beauty and elegance. Our rooms will elevate your stay, whether you're traveling for business, celebrating a honeymoon, or just looking for a remarkable getaway. With views of the valley or the iconic fountains right from your suite, your stay will be nothing short of unforgettable.

### Travel Resort

Rating: 4.2

Rates from \$72.99 to \$259.99

The Best Gaming Resort in the area. With elegant rooms & suites, pool, cabanas, spa, brewery & world-class gaming. This is the best place to play, stay & dine.

### Days Hotel

Rating: 4.2

Rates from \$60.99 to \$266.99

La propriété **OrderBy** des paramètres de recherche n'accepte pas d'entrée telle que **Rooms.BaseRate** pour fournir le prix de la chambre la moins chère, même si les chambres sont déjà triées en fonction du prix. Dans ce cas, les chambres ne sont pas triées en fonction du prix. Pour afficher les hôtels de l'exemple de jeu de données, classés en fonction du prix des chambres, il faudrait de trier les résultats dans votre contrôleur home et envoyer ces résultats à la vue dans l'ordre souhaité.

## Classer des résultats en fonction de plusieurs valeurs

La question est maintenant de savoir comment distinguer entre les hôtels ayant la même évaluation. Un bon moyen serait de classer en fonction de la date de la dernière rénovation de l'hôtel. En d'autres termes, plus la rénovation de l'hôtel est récente, plus l'hôtel apparaît haut dans les résultats.

1. Pour ajouter un deuxième niveau de classement, modifiez les propriétés **OrderBy** et **Select** dans la méthode **Index(SearchData model)** pour y inclure la propriété **LastRenovationDate**.

```
OrderBy = new[] { "Rating desc", "LastRenovationDate desc" },
Select = new[] { "HotelName", "Description", "Rating", "Rooms", "LastRenovationDate" },
```

### TIP

Vous pouvez entrer un nombre quelconque de propriétés dans la liste **OrderBy**. Si des hôtels avaient le même classement et la même date de rénovation, une troisième propriété pourrait ainsi être entrée pour les distinguer.

2. Là encore, nous avons besoin de voir la date de rénovation dans la vue, juste pour être certain que le classement est correct. Pour quelque chose comme une rénovation, il est probable que seule l'année soit nécessaire. Remplacez la boucle de rendu de la vue par le code suivant.

```

<!-- Show the hotel data. -->
@for (var i = 0; i < Model.resultList.Results.Count; i++)
{
    var rateText = $"Rates from ${Model.resultList.Results[i].Document.cheapest} to
${Model.resultList.Results[i].Document.expensive}";
    var lastRenovatedText = $"Last renovated: {
Model.resultList.Results[i].Document.LastRenovationDate.Value.Year}";
    var ratingText = $"Rating: ${Model.resultList.Results[i].Document.Rating}";

    // Display the hotel details.
    @Html.TextArea($"name{i}", Model.resultList.Results[i].Document.HotelName, new { @class
= "box1A" })
        @Html.TextArea($"rating{i}", ratingText, new { @class = "box1B" })
        @Html.TextArea($"rates{i}" , rateText, new { @class = "box2A" })
        @Html.TextArea($"renovation{i}", lastRenovatedText, new { @class = "box2B" })
        @Html.TextArea($"desc{i}", Model.resultList.Results[i].Document.Description, new {
@class = "box3" })
}

```

3. Modifiez la méthode **Next** du contrôleur home pour transférer le composant « année » de la dernière date de rénovation.

```

public async Task<ActionResult> Next(SearchData model)
{
    // Set the next page setting, and call the Index(model) action.
    model.paging = "next";
    await Index(model);

    // Create an empty list.
    var nextHotels = new List<string>();

    // Add a hotel details to the list.
    for (int n = 0; n < model.resultList.Results.Count; n++)
    {
        var ratingText = $"Rating: ${model.resultList.Results[n].Document.Rating}";
        var rateText = $"Rates from ${model.resultList.Results[n].Document.cheapest} to
${model.resultList.Results[n].Document.expensive}";
        var lastRenovatedText = $"Last renovated:
{model.resultList.Results[n].Document.LastRenovationDate.Value.Year}";

        // Add strings to the list.
        nextHotels.Add(model.resultList.Results[n].Document.HotelName);
        nextHotels.Add(ratingText);
        nextHotels.Add(rateText);
        nextHotels.Add(lastRenovatedText);
        nextHotels.Add(model.resultList.Results[n].Document.Description);
    }

    // Rather than return a view, return the list of data.
    return new JsonResult(nextHotels);
}

```

4. Changez la fonction **scrolled** dans la vue de façon à afficher le texte de la rénovation.

```

<script>
    function scrolled() {
        if (myDiv.offsetHeight + myDiv.scrollTop >= myDiv.scrollHeight) {
            $.getJSON("/Home/Next", function (data) {
                var div = document.getElementById('myDiv');

                // Append the returned data to the current list of hotels.
                for (var i = 0; i < data.length; i += 5) {
                    div.innerHTML += '\n<textarea class="box1A">' + data[i] + '</textarea>';
                    div.innerHTML += '\n<textarea class="box1B">' + data[i + 1] + '</textarea>';
                    div.innerHTML += '\n<textarea class="box2A">' + data[i + 2] + '</textarea>';
                    div.innerHTML += '\n<textarea class="box2B">' + data[i + 3] + '</textarea>';
                    div.innerHTML += '\n<textarea class="box3">' + data[i + 4] + '</textarea>';
                }
            });
        }
    }
</script>

```

- Exécuter l'application. Effectuez une recherche sur un terme courant, comme « pool » ou « view », et vérifiez que les hôtels avec la même évaluation sont maintenant affichés dans l'ordre décroissant de la date de rénovation.

Hotels Search - Ordering Results

pool

16 Results

<b>King's Palace Hotel</b> Rates from \$67.99 to \$257.99 Newest kid on the downtown block. Steps away from the most popular destinations in downtown, enjoy free WiFi, an indoor rooftop pool & fitness center, 24 Grab'n'Go & drinks at the bar	Rating: 3.5 Last renovated: 2005
<b>Gacc Capital</b> Rates from \$64.99 to \$265.99 Chic hotel near the city. High-rise hotel in downtown, walking distance to theaters, restaurants and shops, complete with wellness programs.	Rating: 3.5 Last renovated: 2000
<b>Pelham Hotel</b> Rates from \$62.99 to \$168.99	Rating: 3.5 Last renovated: 1983

## Filtrer les résultats en fonction de la distance d'un point géographique

L'évaluation et la date de rénovation, sont des exemples de propriétés dont l'affichage par ordre décroissant est ce qui convient le mieux. Une liste alphabétique serait un exemple d'une bonne utilisation de l'ordre croissant (par exemple, s'il n'y avait qu'une propriété **OrderBy** et si elle était définie sur **HotelName**, les données apparaîtraient par ordre alphabétique). Cependant, pour nos exemples de données, la distance depuis un point géographique serait plus appropriée.

Pour afficher les résultats en fonction de la distance géographique, plusieurs étapes sont nécessaires.

1. Excluez tous les hôtels qui sont en dehors d'un rayon spécifié à partir du point donné en entrant un filtre avec les paramètres de longitude, de latitude et de rayon. La longitude est donnée en premier à la fonction POINT. Le rayon est exprimé en kilomètres.

```
// "Location" must match the field name in the Hotel class.  
// Distance (the radius) is in kilometers.  
// Point order is Longitude then Latitude.  
Filter = $"geo.distance(Location, geography'POINT({model.lon} {model.lat})') le {model.radius}",
```

2. Le filtre ci-dessus ne classe *pas* les résultats en fonction de la distance, il supprime simplement les valeurs non conformes. Pour classer les résultats, entrez un paramètre OrderBy qui spécifie la méthode geoDistance.

```
OrderBy = new[] { $"geo.distance(Location, geography'POINT({model.lon} {model.lat})') asc" },
```

3. Bien que les résultats aient été retournés par Recherche cognitive Azure avec un filtre de distance, la distance calculée entre les données et le point spécifié n'est *pas* retournée. Recalcullez cette valeur dans la vue ou dans le contrôleur si vous voulez l'afficher dans les résultats.

Le code suivant calcule la distance entre deux points définis par leur latitude/longitude.

```
const double EarthRadius = 6371;  
  
public static double Degrees2Radians(double deg)  
{  
    return deg * Math.PI / 180;  
}  
  
public static double DistanceInKm( double lat1,  double lon1, double lat2, double lon2)  
{  
    double dlon = Degrees2Radians(lon2 - lon1);  
    double dlat = Degrees2Radians(lat2 - lat1);  
  
    double a = (Math.Sin(dlat / 2) * Math.Sin(dlat / 2)) + Math.Cos(Degrees2Radians(lat1)) *  
    Math.Cos(Degrees2Radians(lat2)) * (Math.Sin(dlon / 2) * Math.Sin(dlon / 2));  
    double angle = 2 * Math.Atan2(Math.Sqrt(a), Math.Sqrt(1 - a));  
    return angle * EarthRadius;  
}
```

4. Vous devez à présent relier ces concepts. Cependant, ces extraits de code s'éloignent de l'objectif de notre tutoriel : la création d'une application basée sur une carte est laissée au lecteur à titre d'exercice. Pour aller plus loin avec cet exemple, considérez l'entrée d'un nom de ville avec un rayon, ou la localisation d'un point sur une carte et la sélection d'un rayon. Pour examiner ces options en détail, consultez les ressources suivantes :

- [Documentation sur Azure Maps](#)
- [Rechercher une adresse à l'aide du service Recherche Azure Maps](#)

## Classer des résultats en fonction d'un profil de score

Les exemples donnés jusqu'à présent dans le tutoriel montrent comment classer sur des valeurs numériques (évaluation, date de rénovation, distance géographique), fournissant un processus de classement *exact*. Cependant, certaines recherches et certaines données ne se prêtent pas à cette comparaison facile entre deux éléments de données. Recherche cognitive Azure inclut le concept de *scoring*. Des *profils de score* peuvent être spécifiés pour un jeu de données qui peut être utilisé pour fournir des comparaisons plus complexes et qualitatives, ce qui doit

être plus utile par exemple lors de la comparaison de données texte pour déterminer ce qui doit être affiché en premier.

Les profils de score ne sont pas définis par les utilisateurs, mais ils le sont en général par les administrateurs d'un jeu de données. Plusieurs profils de score ont été définis sur les données des hôtels. Voyons comment un profil de score est défini, puis essayons d'écrire du code pour effectuer des recherches sur ces profils.

### Comment les profils de score sont définis

Examinons trois exemples de profils de score et regardons comment chacun *devrait* affecter l'ordre des résultats. En tant que développeur d'applications, vous n'écrivez pas ces profils : ils sont écrits par l'administrateur des données. Il est cependant utile d'en examiner la syntaxe.

1. Ceci est le profil de score par défaut pour le jeu de données des hôtels, utilisé quand vous ne spécifiez pas de paramètre **OrderBy** ou **ScoringProfile**. Ce profil améliore le *score* pour un hôtel si le texte recherché est présent dans le nom, la description ou la liste d'étiquettes (équipements) de l'hôtel. Notez comment les pondérations du score favorisent certains champs. Si le texte de recherche apparaît dans un autre champ, non listé ci-dessous, il aura une pondération de 1. Bien sûr, plus le score est élevé, plus un résultat apparaît haut dans la vue.

```
{  
    "name": "boostByField",  
    "text": {  
        "weights": {  
            "HotelName": 2,  
            "Description": 1.5,  
            "Description_fr": 1.5,  
            "Tags": 3  
        }  
    }  
}
```

2. Le profil de score suivant améliore le score de façon significative si un paramètre fourni contient un ou plusieurs des éléments de la liste d'étiquettes (que nous appelons « équipements »). Le point clé de ce profil est qu'un paramètre *doit* être fourni, contenant du texte. Si le paramètre est vide ou n'est pas fourni, une erreur est levée.

```
{  
    "name": "boostAmenities",  
    "functions": [  
        {  
            "type": "tag",  
            "fieldName": "Tags",  
            "boost": 5,  
            "tag": {  
                "tagsParameter": "amenities"  
            }  
        }  
    ]  
}
```

3. Dans ce troisième exemple, l'évaluation donne une amélioration significative au score. La date de la dernière rénovation augmente également le score, mais seulement si cette donnée n'est pas antérieure de plus de 730 jours (2 ans) à la date actuelle.

```

{
  "name": "renovatedAndHighlyRated",
  "functions": [
    {
      "type": "magnitude",
      "fieldName": "Rating",
      "boost": 20,
      "interpolation": "linear",
      "magnitude": {
        "boostingRangeStart": 0,
        "boostingRangeEnd": 5,
        "constantBoostBeyondRange": false
      }
    },
    {
      "type": "freshness",
      "fieldName": "LastRenovationDate",
      "boost": 10,
      "interpolation": "quadratic",
      "freshness": {
        "boostingDuration": "P730D"
      }
    }
  ]
}

```

Maintenant, voyons si ces profils fonctionnent comme nous le pensons !

### Ajouter du code à la vue pour comparer les profils

- Ouvrez le fichier index.cshtml et remplacez la section <body> par le code suivant.

```

<body>

@using (Html.BeginForm("Index", "Home", FormMethod.Post))
{
    <table>
        <tr>
            <td></td>
            <td>
                <h1 class="sampleTitle">
                    
                    Hotels Search - Order Results
                </h1>
            </td>
        </tr>
        <tr>
            <td></td>
            <td>
                <!-- Display the search text box, with the search icon to the right of it. -->
                <div class="searchBoxForm">
                    @Html.TextBoxFor(m => m.searchText, new { @class = "searchBox" }) <input
                    class="searchBoxSubmit" type="submit" value="">
                </div>

                <div class="searchBoxForm">
                    <b>&nbsp;Order:&nbsp;</b>
                    @Html.RadioButtonFor(m => m.scoring, "Default") Default&nbsp;&nbsp;
                    @Html.RadioButtonFor(m => m.scoring, "RatingRenovation") By numerical
                    Rating&nbsp;&nbsp;
                    @Html.RadioButtonFor(m => m.scoring, "boostAmenities") By Amenities&nbsp;&nbsp;
                    @Html.RadioButtonFor(m => m.scoring, "renovatedAndHighlyRated") By Renovated
                    date/Rating profile&nbsp;&nbsp;
                </div>
            </td>
        </tr>
    </table>
}

```

```

</tr>

<tr>
    <td valign="top">
        <div id="facetplace" class="facetchecks">

            @if (Model != null && Model.facetText != null)
            {
                <h5 class="facetheader">Amenities:</h5>
                <ul class="facetlist">
                    @for (var c = 0; c < Model.facetText.Length; c++)
                    {
                        <li> @Html.CheckBoxFor(m => m.facetOn[c], new { @id = "check" +
c.ToString() }) @Model.facetText[c] </li>
                    }
                </ul>
            }
        </div>
    </td>
    <td>
        @if (Model != null && Model.resultList != null)
        {
            // Show the total result count.
            <p class="sampleText">
                @Html.DisplayFor(m => m.resultList.Count) Results <br />
            </p>

            <div id="myDiv" style="width: 800px; height: 450px; overflow-y: scroll;" onscroll="scrolled()">

                <!-- Show the hotel data. -->
                @for (var i = 0; i < Model.resultList.Results.Count; i++)
                {
                    var rateText = $"Rates from ${Model.resultList.Results[i].Document.cheapest} to ${Model.resultList.Results[i].Document.expensive}";
                    var lastRenovatedText = $"Last renovated: {Model.resultList.Results[i].Document.LastRenovationDate.Value.Year}";
                    var ratingText = $"Rating: {Model.resultList.Results[i].Document.Rating}";

                    string amenities = string.Join(", ", Model.resultList.Results[i].Document.Tags);
                    string fullDescription = Model.resultList.Results[i].Document.Description;
                    fullDescription += $"{Environment.NewLine}Amenities: {amenities}";

                    // Display the hotel details.
                    @Html.TextArea($"name{i}", Model.resultList.Results[i].Document.HotelName, new { @class = "box1A" })
                    @Html.TextArea($"rating{i}", ratingText, new { @class = "box1B" })
                    @Html.TextArea($"rates{i}", rateText, new { @class = "box2A" })
                    @Html.TextArea($"renovation{i}", lastRenovatedText, new { @class = "box2B" })
                }
                @Html.TextArea($"desc{i}", fullDescription, new { @class = "box3" })
            }
        </div>

        <script>
            function scrolled() {
                if (myDiv.offsetHeight + myDiv.scrollTop >= myDiv.scrollHeight) {
                    $.getJSON("/Home/Next", function (data) {
                        var div = document.getElementById('myDiv');

                        // Append the returned data to the current list of hotels.
                        for (var i = 0; i < data.length; i += 5) {
                            div.innerHTML += '\n<textarea class="box1A">' + data[i] +
'</textarea>';
                            div.innerHTML += '<textarea class="box1B">' + data[i + 1] +
'</textarea>';
                            div.innerHTML += '\n<textarea class="box2A">' + data[i + 2] +
'</textarea>';
                        }
                    });
                }
            }
        </script>
    </td>

```

```

'</textarea>';
                    div.innerHTML += '<textarea class="box2B">' + data[i + 3] +
'</textarea>';
                    div.innerHTML += '\n<textarea class="box3">' + data[i + 4] +
'</textarea>';
                }
            });
        }
    
```

2. Ouvrez le fichier SearchData.cs et remplacez la classe **SearchData** par le code suivant.

```

public class searchData
{
    public searchData()
    {
    }

    // Constructor to initialize the list of facets sent from the controller.
    public searchData(List<string> facets)
    {
        facetText = new string[facets.Count];

        for (int i = 0; i < facets.Count; i++)
        {
            facetText[i] = facets[i];
        }
    }

    // Array to hold the text for each amenity.
    public string[] facetText { get; set; }

    // Array to hold the setting for each amenity.
    public bool[] facetOn { get; set; }

    // The text to search for.
    public string searchText { get; set; }

    // Record if the next page is requested.
    public string paging { get; set; }

    // The list of results.
    public DocumentSearchResult<Hotel> resultlist;

    public string scoring { get; set; }
}

```

3. Ouvrez le fichier hotels.css et ajoutez les classes HTML suivantes.

```

.facetlist {
    list-style: none;
}

.facetchecks {
    width: 250px;
    display: normal;
    color: #666;
    margin: 10px;
    padding: 5px;
}

.facetheader {
    font-size: 10pt;
    font-weight: bold;
    color: darkgreen;
}

```

## Ajouter du code au contrôleur pour spécifier un profil de score

- Ouvrez le fichier du contrôleur home. Ajoutez l'instruction **using** suivante (pour faciliter la création de listes).

```
using System.Linq;
```

- Pour cet exemple, nous avons besoin de l'appel initial à **Index** pour en faire un peu plus que retourner seulement la vue initiale. La méthode recherche maintenant jusqu'à 20 équipements à afficher dans la vue.

```

public async Task<ActionResult> Index()
{
    InitSearch();

    // Set up the facets call in the search parameters.
    SearchParameters sp = new SearchParameters()
    {
        // Search for up to 20 amenities.
        Facets = new List<string> { "Tags,count:20" },
    };

    DocumentSearchResult<Hotel> searchResult = await _indexClient.Documents.SearchAsync<Hotel>("*",
sp);

    // Convert the results to a list that can be displayed in the client.
    List<string> facets = searchResult.Facets["Tags"].Select(x => x.Value.ToString()).ToList();

    // Initiate a model with a list of facets for the first view.
    searchData model = new searchData(facets);

    // Save the facet text for the next view.
    SaveFacets(model, false);

    // Render the view including the facets.
    return View(model);
}

```

- Nous avons besoin de deux méthodes privées pour enregistrer les facettes dans un stockage temporaire, et pour les récupérer auprès d'un stockage temporaire et remplir un modèle.

```

// Save the facet text to temporary storage, optionally saving the state of the check boxes.
private void SaveFacets(SearchData model, bool saveChecks = false)
{
    for (int i = 0; i < model.facetText.Length; i++)
    {
        TempData["facet" + i.ToString()] = model.facetText[i];
        if (saveChecks)
        {
            TempData["facetOn" + i.ToString()] = model.facetOn[i];
        }
    }
    TempData["facetcount"] = model.facetText.Length;
}

// Recover the facet text to a model, optionally recovering the state of the check boxes.
private void RecoverFacets(SearchData model, bool recoverChecks = false)
{
    // Create arrays of the appropriate length.
    model.facetText = new string[(int)TempData["facetcount"]];
    if (recoverChecks)
    {
        model.facetOn = new bool[(int)TempData["facetcount"]];
    }

    for (int i = 0; i < (int)TempData["facetcount"]; i++)
    {
        model.facetText[i] = TempData["facet" + i.ToString()].ToString();
        if (recoverChecks)
        {
            model.facetOn[i] = (bool)TempData["facetOn" + i.ToString()];
        }
    }
}

```

4. Nous devons définir les paramètres **OrderBy** et **ScoringProfile** en fonction des besoins. Remplacez la méthode **Index(SearchData model)** existante par ce qui suit.

```

public async Task<ActionResult> Index(SearchData model)
{
    try
    {
        InitSearch();

        int page;

        if (model.paging != null && model.paging == "next")
        {
            // Recover the facet text, and the facet check box settings.
            RecoverFacets(model, true);

            // Increment the page.
            page = (int)TempData["page"] + 1;

            // Recover the search text.
            model.searchText = TempData["searchfor"].ToString();
        }
        else
        {
            // First search with text.
            // Recover the facet text, but ignore the check box settings, and use the current model
            settings.
            RecoverFacets(model, false);

            // First call. Check for valid text input, and valid scoring profile.
            if (model.searchText == null)
            {

```

```

        model.searchText = "";
    }
    if (model.scoring == null)
    {
        model.scoring = "Default";
    }
    page = 0;
}

// Set empty defaults for ordering and scoring parameters.
var orderby = new List<string>();
string profile = "";
var scoringParams = new List<ScoringParameter>();

// Set the ordering based on the user's radio button selection.
switch (model.scoring)
{
    case "RatingRenovation":
        orderby.Add("Rating desc");
        orderby.Add("LastRenovationDate desc");
        break;

    case "boostAmenities":
    {
        profile = model.scoring;
        var setAmenities = new List<string>();

        // Create a string list of amenities that have been clicked.
        for (int a = 0; a < model.facetOn.Length; a++)
        {
            if (model.facetOn[a])
            {
                setAmenities.Add(model.facetText[a]);
            }
        }
        if (setAmenities.Count > 0)
        {
            // Only set scoring parameters if there are any.
            var sp = new ScoringParameter("amenities", setAmenities);
            scoringParams.Add(sp);
        }
        else
        {
            // No amenities selected, so set profile back to default.
            profile = "";
        }
    }
    break;

    case "renovatedAndHighlyRated":
        profile = model.scoring;
        break;

    default:
        break;
}

// Setup the search parameters.
var parameters = new SearchParameters
{
    // Set the ordering/scoring parameters.
    OrderBy = orderby,
    ScoringProfile = profile,
    ScoringParameters = scoringParams,

    // Select the data properties to be returned.
    Select = new[] { "HotelName", "Description", "Tags", "Rooms", "Rating",
        "LastRenovationDate" },
    SearchMode = SearchMode.All
}

```

```

SearchMode = SearchMode.All,
Skip = page * GlobalVariables.ResultsPerPage,
Top = GlobalVariables.ResultsPerPage,
IncludeTotalResultCount = true,
};

// For efficiency, the search call should be asynchronous, so use SearchAsync rather than
Search.
model.resultList = await _indexClient.Documents.SearchAsync<Hotel>(model.searchText,
parameters);

// Ensure TempData is stored for the next call.
TempData["page"] = page;
TempData["searchfor"] = model.searchText;
TempData["scoring"] = model.scoring;
SaveFacets(model,true);

// Calculate the room rate ranges.
for (int n = 0; n < model.resultList.Results.Count; n++)
{
    var cheapest = 0d;
    var expensive = 0d;

    for (var r = 0; r < model.resultList.Results[n].Document.Rooms.Length; r++)
    {
        var rate = model.resultList.Results[n].Document.Rooms[r].BaseRate;
        if (rate < cheapest || cheapest == 0)
        {
            cheapest = (double)rate;
        }
        if (rate > expensive)
        {
            expensive = (double)rate;
        }
    }
    model.resultList.Results[n].Document.cheapest = cheapest;
    model.resultList.Results[n].Document.expensive = expensive;
}
}
catch
{
    return View("Error", new ErrorViewModel { RequestId = "1" });
}
return View("Index", model);
}

```

Lisez les commentaires pour chacune des sélections de **switch**.

5. Nous n'avons pas besoin d'apporter des modifications à l'action **Next** si vous avez ajouté le code supplémentaire pour la section précédente sur le classement basé sur plusieurs propriétés.

## Exécuter et tester l'application

1. Exécutez l'application. Vous devez voir un ensemble complet d'équipements dans la vue.
2. Pour le classement, sélectionner « By numerical Rating » (Par évaluation numérique) vous donnera le classement numérique que vous avez déjà implémenté dans ce tutoriel, avec la date de rénovation déterminant le classement des hôtels ayant la même évaluation.

## Hotels Search - Order Results

beach

Order:  Default  By numerical Rating  By Amenities  By Renovated date/Rating profile

**Amenities:**

- view
- 24-hour front desk service
- laundry service
- air conditioning
- concierge
- pool
- continental breakfast
- free wifi
- restaurant
- coffee in lobby
- free parking
- bar

**Johnson's Resort** Rating: 4.8  
**Rates from \$65.99 to \$268.99** Last renovated: 1978  
 been built with all the comforts of home. Sporting a huge beach with multiple water toys for those sunny summer days and a Lodge full of games for when you just can't swim anymore, there's always something for the family to do. A full marina offers watercraft rentals, boat launch, powered dock slips, canoes (free to use), & fish cleaning facility.  
Rent pontoon, 14' fishing boats, 16' fishing rigs or jet skis for a fun day or week on the lake.

**Lady Of The Lake B & B** Rating: 4.7  
**Rates from \$60.99 to \$265.99** Last renovated: 1987  
 Nature is Home on the beach. Save up to 30 percent. Valid Now through the end of the year. Restrictions and blackout may apply.  
 Amenities: laundry service, concierge, view

**Nova Hotel & Spa** Rating: 3.6  
**Rates from \$60.99 to \$269.99** Last renovated: 1973  
1 Mile from the airport. Free WiFi, Outdoor Pool, Complimentary Airport Shuttle, 6 miles from the beach & 10 miles from downtown.

3. Essayez maintenant le profil « By amenities » (Par équipements). Effectuez différentes sélections d'équipements et vérifiez que les hôtels avec ces équipements sont promus en haut de la liste des résultats.

## Hotels Search - Order Results

beach

Order:  Default  By numerical Rating  By Amenities  By Renovated date/Rating profile

**Amenities:**

- view
- 24-hour front desk service
- laundry service
- air conditioning
- concierge
- pool
- continental breakfast
- free wifi
- restaurant
- coffee in lobby
- free parking
- bar

**Nova Hotel & Spa** Rating: 3.6  
**Rates from \$60.99 to \$269.99** Last renovated: 1973  
 1 Mile from the airport. Free WiFi, Outdoor Pool, Complimentary Airport Shuttle, 6 miles from the beach & 10 miles from downtown.  
 Amenities: pool, continental breakfast, free parking

**Whitefish Lodge & Suites** Rating: 2.4  
**Rates from \$92.99 to \$257.99** Last renovated: 1987  
 Located on in the heart of the forest. Enjoy Warm Weather, Beach Club Services, Natural Hot Springs, Airport Shuttle.  
 Amenities: continental breakfast, free parking, free wifi

**Ocean Air Motel** Rating: 3.5  
**Rates from \$63.99 to \$254.99** Last renovated: 1951

4. Essayez « By Renovated date/Rating profile » (Par profil de Date de rénovation/Évaluation) pour voir si vous obtenez ce que vous attendez. Seuls les hôtels rénovés récemment doivent obtenir une amélioration *freshness* (fraîcheur).

### Ressources

Pour plus d'informations, consultez [Ajouter des profils de score à un index Recherche cognitive Azure](#).

# Éléments importants à retenir

Retenez les points importants suivants de ce projet :

- Les utilisateurs s'attendent à ce que les résultats de la recherche soient classés, les plus pertinents apparaissant en premier.
- Les données doivent être structurées pour que le classement soit facile. Nous n'avons pas pu effectuer facilement un tri sur « cheapest » (le moins cher) pour commencer, car les données ne sont pas structurées pour permettre un classement sans du code supplémentaire.
- Il peut y avoir de nombreux niveaux de classement pour faire la distinction entre les résultats qui ont la même valeur à un niveau plus élevé du classement.
- Il est naturel que certains résultats soient classés par ordre croissant (par exemple la distance séparant d'un point) et certains autres par ordre décroissant (par exemple l'évaluation des invités).
- Des profils de score peuvent être définis quand des comparaisons numériques ne sont pas disponibles ou pas assez pertinentes pour un jeu de données. Attribuer un score à chaque résultat aide à classer et à afficher les résultats de façon intelligente.

## Étapes suivantes

Vous avez terminé cette série de tutoriels C# et vous avez normalement acquis une bonne connaissance des API de Recherche cognitive Azure.

Pour plus d'informations de référence et des tutoriels, parcourez [Microsoft Learn](#) ou les autres tutoriels de la [Documentation de Recherche cognitive Azure](#).

# Tutoriel : Indexer des données Azure SQL à l'aide du SDK .NET

04/10/2020 • 17 minutes to read • [Edit Online](#)

Configurez un [indexeur](#) pour extraire des données interrogeables d'Azure SQL Database, en les envoyant à un index de recherche dans Recherche cognitive Azure.

Ce tutoriel utilise C# et le [SDK .NET](#) pour effectuer les tâches suivantes :

- Créer une source de données qui se connecte à une base de données Azure SQL
- Créer un indexeur
- Exécuter un indexeur pour charger des données dans un index
- Interroger un index dans le cadre d'une étape de vérification

Si vous n'avez pas d'abonnement Azure, créez un [compte gratuit](#) avant de commencer.

## Prérequis

- [Azure SQL Database](#)
- [Visual Studio](#)
- [Créer ou rechercher un service de recherche existant](#)

### NOTE

Vous pouvez utiliser le service gratuit pour ce tutoriel. Avec un service de recherche gratuit, vous êtes limité à trois index, trois indexeurs et trois sources de données. Ce didacticiel crée une occurrence de chaque élément. Avant de commencer, veillez à disposer de l'espace suffisant sur votre service pour accepter les nouvelles ressources.

## Télécharger les fichiers

Le code source pour ce tutoriel se trouve dans le dossier [DotNetHowToIndexer](#) du dépôt GitHub [Azure-Samples/search-dotnet-getting-started](#).

## 1 - Créer les services

Ce tutoriel utilise Recherche cognitive Azure pour l'indexation et les requêtes, et Azure SQL Database comme source de données externe. Si possible, créez les deux services dans la même région et le même groupe de ressources pour des raisons de proximité et de facilité de gestion. En pratique, Azure SQL Database peut se trouver dans n'importe quelle région.

### Démarrer avec Azure SQL Database

Au cours de cette étape, vous allez créer, sur Azure SQL Database, une source de données externe qu'un indexeur peut analyser. Vous pouvez utiliser le portail Azure et le fichier *hotels.sql* de l'exemple téléchargé pour créer le jeu de données dans Azure SQL Database. Recherche cognitive Azure utilise des ensembles de lignes aplatis, comme celui généré à partir d'une vue ou d'une requête. Le fichier SQL de l'exemple de solution crée et remplit une table unique.

Si vous disposez d'une ressource Azure SQL Database, vous pouvez y ajouter la table *hotels* en commençant à l'étape 4.

1. Connectez-vous au portail Azure.
2. Recherchez ou créez une **base de données SQL**. Vous pouvez utiliser les valeurs par défaut et le niveau de tarification le moins élevé. Un des avantages lié à la création d'un serveur est de pouvoir spécifier un nom d'utilisateur administrateur et le mot de passe nécessaire pour la création et le chargement des tables lors d'une étape ultérieure.

The screenshot shows the 'Create SQL Database' wizard in the Azure portal. The 'Basics' tab is selected. The 'Project details' section asks to select a subscription and resource group. The 'Database details' section asks to enter database settings, including a name ('hotels'), server ('(new)'), and compute/storage configuration ('General Purpose, Gen5, 2 vCores, 32 GB storage'). The 'Compute + storage' section also includes a 'Configure database' link. At the bottom, there are 'Review + create' and 'Next : Networking >' buttons.

3. Cliquez sur **Vérifier + créer** pour déployer le nouveau serveur et la nouvelle base de données. Patientez jusqu'au déploiement du serveur et de la base de données.
4. Dans le volet de navigation, cliquez sur **Éditeur de requêtes (préversion)** et entrez le nom d'utilisateur et le mot de passe de l'administrateur du serveur.  
Si l'accès est refusé, copiez l'adresse IP du client à partir du message d'erreur, puis cliquez sur le lien **Définir le pare-feu du serveur** pour ajouter une règle autorisant l'accès à partir de votre ordinateur client, en utilisant l'adresse IP de votre client pour la plage. La prise en compte de la règle peut prendre plusieurs minutes.
5. Dans l'Éditeur de requête, cliquez sur **Ouvrir la requête** et accédez à l'emplacement du fichier *hotels.sql* sur votre ordinateur local.

6. Sélectionnez le fichier et cliquez sur **Ouvrir**. Le script doit ressembler à la capture d'écran suivante :

```
1 ALTER DATABASE CURRENT
2 SET CHANGE_TRACKING = ON
3 (CHANGE_RETENTION = 2 DAYS, AUTO_CLEANUP = ON)
4
5 CREATE TABLE Hotels (
6     [HotelId] nvarchar(450) NOT NULL PRIMARY KEY,
7     [BaseRate] float NULL,
8     [Category] nvarchar(max) NULL,
9     [Description] nvarchar(max) NULL,
10    [Description_fr] nvarchar(max) NULL,
11    [HotelName] nvarchar(max) NULL,
12    [Tags] nvarchar(max) NULL,
13    [IsDeleted] bit NOT NULL,
14    [LastRenovationDate] DateTime NULL,
```

Results Messages

Query succeeded: Affected rows: 0Affected rows: 3

7. Cliquez sur **Exécuter** pour exécuter la requête. Dans le volet Résultats, vous devez voir un message de réussite de la requête pour 3 lignes.
8. Pour renvoyer un ensemble de lignes de cette table, vous pouvez exécuter la requête suivante comme étape de vérification :

```
SELECT * FROM Hotels
```

9. Copiez la chaîne de connexion ADO.NET pour la base de données. Sous **Paramètres > Chaînes de connexion**, copiez la chaîne de connexion ADO.NET, similaire à celle présentée dans l'exemple ci-dessous.

```
Server=tcp:{your_dbname}.database.windows.net,1433;Initial Catalog=hotels-db;Persist Security
Info=False;User ID={your_username};Password=
{your_password};MultipleActiveResultSets=False;Encrypt=True;TrustServerCertificate=False;Connection
Timeout=30;
```

Vous aurez besoin de cette chaîne de connexion dans l'exercice suivant, lors de la configuration de votre environnement.

### Recherche cognitive Azure

Le composant suivant est Recherche cognitive Azure, que vous pouvez [créer dans le portail](#). Vous pouvez utiliser le niveau gratuit pour effectuer cette procédure pas à pas.

#### Obtenir une clé API d'administration et une URL pour Recherche cognitive Azure

Les appels d'API nécessitent l'URL du service et une clé d'accès. Un service de recherche est créé avec les deux. Ainsi, si vous avez ajouté la Recherche cognitive Azure à votre abonnement, effectuez ce qui suit pour obtenir les informations nécessaires :

1. [Connectez-vous au portail Azure](#), puis dans la page **Vue d'ensemble** du service de recherche, récupérez l'URL. Voici un exemple de point de terminaison : <https://mydemo.search.windows.net>.
2. Dans **Paramètres > Clés**, obtenez une clé d'administration pour avoir des droits d'accès complets sur le service. Il existe deux clés d'administration interchangeables, fournies pour assurer la continuité de l'activité.

au cas où vous deviez en remplacer une. Vous pouvez utiliser la clé primaire ou secondaire sur les demandes d'ajout, de modification et de suppression d'objets.

The screenshot shows two overlapping Azure portal windows. The top window is titled 'mydemo' and displays a 'Vue d'ensemble' card. A red box highlights the 'Vue d'ensemble' card, and a red circle with the number '1' highlights the 'URL' field which contains 'https://mydemo.search.windows.net'. The bottom window is titled 'mydemo - Clés' and shows a list of service principals. A red box highlights the 'Clés' card, and a red circle with the number '2' highlights the 'Clé d'administration primaire' field, which contains '<placeholder-for-alphanumeric-autogenerated-string>'.

## 2 - Configurer votre environnement

1. Démarrez Visual Studio et ouvrez le fichier **DotNetHowToIndexers.sln**.
2. Dans l'Explorateur de solutions, ouvrez **appsettings.json** pour fournir les informations de connexion.
3. Pour `searchServiceName`, si l'URL complète est « <https://my-demo-service.search.windows.net> », le nom de service à spécifier est « my-demo-service ».
4. Pour `AzureSqlConnectionString`, le format de la chaîne est similaire à celui-ci :

```
"Server=tcp:{your_dbname}.database.windows.net,1433;Initial Catalog=hotels-db;Persist Security Info=False;User ID={your_username};Password={your_password};MultipleActiveResultSets=False;Encrypt=True;TrustServerCertificate=False;Connection Timeout=30;"
```

```
{  
  "SearchServiceName": "<placeholder-Azure-Search-service-name>",  
  "SearchServiceAdminApiKey": "<placeholder-admin-key-for-Azure-Search>",  
  "AzureSqlConnectionString": "<placeholder-ADO.NET-connection-string>"  
}
```

5. Vérifiez que la chaîne de connexion contient un mot de passe valide. Le nom de l'utilisateur et celui de la base de données sont copiés dans votre chaîne de connexion, mais le mot de passe doit être entré manuellement.

## 3 - Créer le pipeline

Les indexeurs nécessitent un objet de source de données et un index. Le code approprié se trouve dans deux fichiers :

- **hotel.cs**, contenant un schéma qui définit l'index
- **Program.cs**, contenant les fonctions permettant de créer et de gérer des structures dans votre service

### Dans Hotel.cs

Le schéma d'index définit la collection de champs, y compris les attributs spécifiant les opérations autorisées, pour savoir, par exemple, si un champ peut faire l'objet d'une interrogation en texte intégral, d'un filtrage ou d'un tri

comme indiqué dans la définition de champ suivante pour HotelName.

```
    ...
    [IsSearchable, IsFilterable, IsSortable]
    public string HotelName { get; set; }
    ...
```

Un schéma peut également inclure d'autres éléments, y compris des profils de notation pour améliorer un score de recherche, des analyseurs personnalisés et d'autres constructions. Toutefois, en ce qui nous concerne, le schéma est peu défini et comporte uniquement des champs trouvés dans les exemples de jeux de données.

## Dans Program.cs

Le programme principal inclut une logique pour la création d'un client, d'un index, d'une source de données et d'un indexeur. Le code recherche et supprime les ressources existantes du même nom, en supposant que vous pouvez exécuter ce programme plusieurs fois.

L'objet de source de données est configuré avec des paramètres propres aux ressources Azure SQL Database, notamment l'[indexation partielle ou incrémentielle](#), pour tirer parti des [fonctionnalités intégrées de détection des modifications](#) d'Azure SQL. La base de données des hôtels de démonstration dans Azure SQL a une colonne « suppression réversible » nommée **IsDeleted**. Quand cette colonne est définie sur true dans la base de données, l'indexeur supprime le document correspondant dans l'index de Recherche cognitive Azure.

```
Console.WriteLine("Creating data source...");

DataSource dataSource = DataSource.AzureSql(
    name: "azure-sql",
    sqlConnectionString: configuration["AzureSQLConnectionString"],
    tableOrViewName: "hotels",
    deletionDetectionPolicy: new SoftDeleteColumnDeletionDetectionPolicy(
        softDeleteColumnName: "IsDeleted",
        softDeleteMarkerValue: "true"));
dataSource.DataChangeDetectionPolicy = new SqlIntegratedChangeTrackingPolicy();

searchService.DataSources.CreateOrUpdateAsync(dataSource).Wait();
```

Un objet de l'indexeur est indépendant de la plateforme, où la configuration, la planification et l'appel sont les mêmes quelle que soit la source. Cet exemple d'indexeur inclut une planification, une option de réinitialisation qui efface l'historique de l'indexeur, et appelle une méthode pour créer et exécuter immédiatement l'indexeur.

```

Console.WriteLine("Creating Azure SQL indexer...");
Indexer indexer = new Indexer(
    name: "azure-sql-indexer",
    dataSourceName: dataSource.Name,
    targetIndexName: index.Name,
    schedule: new IndexingSchedule(TimeSpan.FromDays(1)));
// Indexers contain metadata about how much they have already indexed
// If we already ran the sample, the indexer will remember that it already
// indexed the sample data and not run again
// To avoid this, reset the indexer if it exists
exists = await searchService.Indexers.ExistsAsync(indexer.Name);
if (exists)
{
    await searchService.Indexers.ResetAsync(indexer.Name);
}

await searchService.Indexers.CreateOrUpdateAsync(indexer);

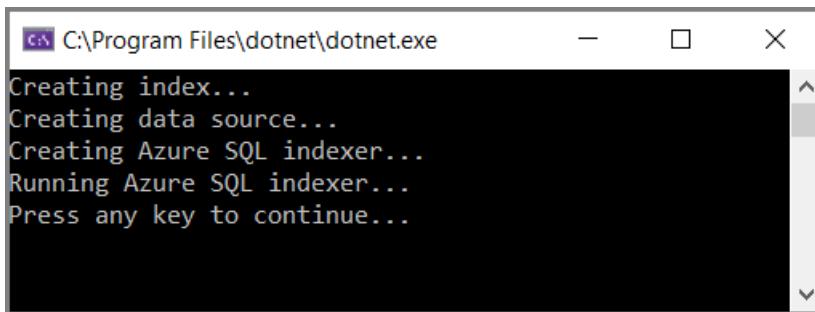
// We created the indexer with a schedule, but we also
// want to run it immediately
Console.WriteLine("Running Azure SQL indexer...");

try
{
    await searchService.Indexers.RunAsync(indexer.Name);
}
catch (CloudException e) when (e.Response.StatusCode == (HttpStatusCode)429)
{
    Console.WriteLine("Failed to run indexer: {0}", e.Response.Content);
}

```

## 4 - Générer la solution

Appuyez sur F5 pour générer et exécuter la solution. Le programme s'exécute en mode débogage. Une fenêtre de console signale l'état de chaque opération.



Votre code s'exécute localement dans Visual Studio en se connectant à votre service de recherche sur Azure qui, à son tour, se connecte à Azure SQL Database et récupère le jeu de données. Étant donné le grand nombre d'opérations, il existe plusieurs points de défaillance potentiels. Si une erreur survient, commencez par vérifier les conditions suivantes :

- Les informations de connexion du service de recherche que vous fournissez se limitent au nom du service de ce didacticiel. Si vous avez entré l'URL complète, les opérations s'arrêtent à la création d'index avec une erreur signalant un échec de connexion.
- Informations de connexion à la base de données dans **appsettings.json**. Il doit s'agir de la chaîne de connexion ADO.NET obtenue à partir du portail et modifiée pour inclure un nom d'utilisateur et un mot de passe valides pour votre base de données. Le compte d'utilisateur doit avoir l'autorisation de récupérer des données. L'accès à partir de l'adresse IP du client local doit être autorisé.
- Limites des ressources. N'oubliez pas que le niveau Gratuit a une limite de 3 index, indexeurs et sources de

données. Un service ayant atteint la limite maximale ne peut pas créer de nouveaux objets.

## 5 - Recherche

Utilisez le portail Azure pour vérifier la création des objets, puis utilisez l'**Explorateur de recherche** pour interroger l'index.

1. **Connectez-vous au portail Azure.** Dans la page **Vue d'ensemble** de votre service de recherche, ouvrez chaque liste successivement pour vérifier que l'objet est créé. Sous **Index, Indexeurs et Sources de données**, vous devez voir « hotels », « azure-sql-indexer » et « azure-sql » respectivement.

Status ↑↓	Name ↑↓	Last run ↑↓	Docs succeeded
✓ Success	azure-sql-indexer	7 min ago	3/3

2. Sélectionnez l'index hotels. Dans la page hotels, le premier onglet est celui de l'**Explorateur de recherche**.
3. Cliquez sur **Rechercher** pour émettre une requête vide.

Les trois entrées de votre index sont renvoyées en tant que documents JSON. L'explorateur de recherche renvoie des documents au format JSON afin que vous puissiez afficher l'ensemble de la structure.

```
1 {  
2   "@odata.context": "https://.search.windows.net/indexes('hotels')/$metadata#docs(*)",  
3   "value": [  
4     {  
5       "@search.score": 1,  
6       "hotelId": "3",  
7       "baseRate": 129.99,  
8       "description": "Close to town hall and the river",  
9     }  
10    ]  
11  }  
12 }  
13 }  
14 }  
15 }  
16 }  
17 }  
18 }  
19 }  
20 }  
21 }  
22 }  
23 }  
24 }  
25 }  
26 }  
27 }  
28 }  
29 }  
30 }  
31 }  
32 }  
33 }  
34 }  
35 }  
36 }  
37 }  
38 }  
39 }  
40 }  
41 }  
42 }  
43 }  
44 }  
45 }  
46 }  
47 }  
48 }  
49 }  
50 }  
51 }  
52 }  
53 }  
54 }  
55 }  
56 }  
57 }  
58 }  
59 }  
60 }  
61 }  
62 }  
63 }  
64 }  
65 }  
66 }  
67 }  
68 }  
69 }  
70 }  
71 }  
72 }  
73 }  
74 }  
75 }  
76 }  
77 }  
78 }  
79 }  
80 }  
81 }  
82 }  
83 }  
84 }  
85 }  
86 }  
87 }  
88 }  
89 }  
90 }  
91 }  
92 }  
93 }  
94 }  
95 }  
96 }  
97 }  
98 }  
99 }  
100 }  
101 }  
102 }  
103 }  
104 }  
105 }  
106 }  
107 }  
108 }  
109 }  
110 }  
111 }  
112 }  
113 }  
114 }  
115 }  
116 }  
117 }  
118 }  
119 }  
120 }  
121 }  
122 }  
123 }  
124 }  
125 }  
126 }  
127 }  
128 }  
129 }  
130 }  
131 }  
132 }  
133 }  
134 }  
135 }  
136 }  
137 }  
138 }  
139 }  
140 }  
141 }  
142 }  
143 }  
144 }  
145 }  
146 }  
147 }  
148 }  
149 }  
150 }  
151 }  
152 }  
153 }  
154 }  
155 }  
156 }  
157 }  
158 }  
159 }  
160 }  
161 }  
162 }  
163 }  
164 }  
165 }  
166 }  
167 }  
168 }  
169 }  
170 }  
171 }  
172 }  
173 }  
174 }  
175 }  
176 }  
177 }  
178 }  
179 }  
180 }  
181 }  
182 }  
183 }  
184 }  
185 }  
186 }  
187 }  
188 }  
189 }  
190 }  
191 }  
192 }  
193 }  
194 }  
195 }  
196 }  
197 }  
198 }  
199 }  
200 }  
201 }  
202 }  
203 }  
204 }  
205 }  
206 }  
207 }  
208 }  
209 }  
210 }  
211 }  
212 }  
213 }  
214 }  
215 }  
216 }  
217 }  
218 }  
219 }  
220 }  
221 }  
222 }  
223 }  
224 }  
225 }  
226 }  
227 }  
228 }  
229 }  
230 }  
231 }  
232 }  
233 }  
234 }  
235 }  
236 }  
237 }  
238 }  
239 }  
240 }  
241 }  
242 }  
243 }  
244 }  
245 }  
246 }  
247 }  
248 }  
249 }  
250 }  
251 }  
252 }  
253 }  
254 }  
255 }  
256 }  
257 }  
258 }  
259 }  
260 }  
261 }  
262 }  
263 }  
264 }  
265 }  
266 }  
267 }  
268 }  
269 }  
270 }  
271 }  
272 }  
273 }  
274 }  
275 }  
276 }  
277 }  
278 }  
279 }  
280 }  
281 }  
282 }  
283 }  
284 }  
285 }  
286 }  
287 }  
288 }  
289 }  
290 }  
291 }  
292 }  
293 }  
294 }  
295 }  
296 }  
297 }  
298 }  
299 }  
300 }  
301 }  
302 }  
303 }  
304 }  
305 }  
306 }  
307 }  
308 }  
309 }  
310 }  
311 }  
312 }  
313 }  
314 }  
315 }  
316 }  
317 }  
318 }  
319 }  
320 }  
321 }  
322 }  
323 }  
324 }  
325 }  
326 }  
327 }  
328 }  
329 }  
330 }  
331 }  
332 }  
333 }  
334 }  
335 }  
336 }  
337 }  
338 }  
339 }  
340 }  
341 }  
342 }  
343 }  
344 }  
345 }  
346 }  
347 }  
348 }  
349 }  
350 }  
351 }  
352 }  
353 }  
354 }  
355 }  
356 }  
357 }  
358 }  
359 }  
360 }  
361 }  
362 }  
363 }  
364 }  
365 }  
366 }  
367 }  
368 }  
369 }  
370 }  
371 }  
372 }  
373 }  
374 }  
375 }  
376 }  
377 }  
378 }  
379 }  
380 }  
381 }  
382 }  
383 }  
384 }  
385 }  
386 }  
387 }  
388 }  
389 }  
390 }  
391 }  
392 }  
393 }  
394 }  
395 }  
396 }  
397 }  
398 }  
399 }  
400 }  
401 }  
402 }  
403 }  
404 }  
405 }  
406 }  
407 }  
408 }  
409 }  
410 }  
411 }  
412 }  
413 }  
414 }  
415 }  
416 }  
417 }  
418 }  
419 }  
420 }  
421 }  
422 }  
423 }  
424 }  
425 }  
426 }  
427 }  
428 }  
429 }  
430 }  
431 }  
432 }  
433 }  
434 }  
435 }  
436 }  
437 }  
438 }  
439 }  
440 }  
441 }  
442 }  
443 }  
444 }  
445 }  
446 }  
447 }  
448 }  
449 }  
450 }  
451 }  
452 }  
453 }  
454 }  
455 }  
456 }  
457 }  
458 }  
459 }  
460 }  
461 }  
462 }  
463 }  
464 }  
465 }  
466 }  
467 }  
468 }  
469 }  
470 }  
471 }  
472 }  
473 }  
474 }  
475 }  
476 }  
477 }  
478 }  
479 }  
480 }  
481 }  
482 }  
483 }  
484 }  
485 }  
486 }  
487 }  
488 }  
489 }  
490 }  
491 }  
492 }  
493 }  
494 }  
495 }  
496 }  
497 }  
498 }  
499 }  
500 }  
501 }  
502 }  
503 }  
504 }  
505 }  
506 }  
507 }  
508 }  
509 }  
510 }  
511 }  
512 }  
513 }  
514 }  
515 }  
516 }  
517 }  
518 }  
519 }  
520 }  
521 }  
522 }  
523 }  
524 }  
525 }  
526 }  
527 }  
528 }  
529 }  
530 }  
531 }  
532 }  
533 }  
534 }  
535 }  
536 }  
537 }  
538 }  
539 }  
540 }  
541 }  
542 }  
543 }  
544 }  
545 }  
546 }  
547 }  
548 }  
549 }  
550 }  
551 }  
552 }  
553 }  
554 }  
555 }  
556 }  
557 }  
558 }  
559 }  
560 }  
561 }  
562 }  
563 }  
564 }  
565 }  
566 }  
567 }  
568 }  
569 }  
570 }  
571 }  
572 }  
573 }  
574 }  
575 }  
576 }  
577 }  
578 }  
579 }  
580 }  
581 }  
582 }  
583 }  
584 }  
585 }  
586 }  
587 }  
588 }  
589 }  
590 }  
591 }  
592 }  
593 }  
594 }  
595 }  
596 }  
597 }  
598 }  
599 }  
600 }  
601 }  
602 }  
603 }  
604 }  
605 }  
606 }  
607 }  
608 }  
609 }  
610 }  
611 }  
612 }  
613 }  
614 }  
615 }  
616 }  
617 }  
618 }  
619 }  
620 }  
621 }  
622 }  
623 }  
624 }  
625 }  
626 }  
627 }  
628 }  
629 }  
630 }  
631 }  
632 }  
633 }  
634 }  
635 }  
636 }  
637 }  
638 }  
639 }  
640 }  
641 }  
642 }  
643 }  
644 }  
645 }  
646 }  
647 }  
648 }  
649 }  
650 }  
651 }  
652 }  
653 }  
654 }  
655 }  
656 }  
657 }  
658 }  
659 }  
660 }  
661 }  
662 }  
663 }  
664 }  
665 }  
666 }  
667 }  
668 }  
669 }  
670 }  
671 }  
672 }  
673 }  
674 }  
675 }  
676 }  
677 }  
678 }  
679 }  
680 }  
681 }  
682 }  
683 }  
684 }  
685 }  
686 }  
687 }  
688 }  
689 }  
690 }  
691 }  
692 }  
693 }  
694 }  
695 }  
696 }  
697 }  
698 }  
699 }  
700 }  
701 }  
702 }  
703 }  
704 }  
705 }  
706 }  
707 }  
708 }  
709 }  
710 }  
711 }  
712 }  
713 }  
714 }  
715 }  
716 }  
717 }  
718 }  
719 }  
720 }  
721 }  
722 }  
723 }  
724 }  
725 }  
726 }  
727 }  
728 }  
729 }  
730 }  
731 }  
732 }  
733 }  
734 }  
735 }  
736 }  
737 }  
738 }  
739 }  
740 }  
741 }  
742 }  
743 }  
744 }  
745 }  
746 }  
747 }  
748 }  
749 }  
750 }  
751 }  
752 }  
753 }  
754 }  
755 }  
756 }  
757 }  
758 }  
759 }  
760 }  
761 }  
762 }  
763 }  
764 }  
765 }  
766 }  
767 }  
768 }  
769 }  
770 }  
771 }  
772 }  
773 }  
774 }  
775 }  
776 }  
777 }  
778 }  
779 }  
780 }  
781 }  
782 }  
783 }  
784 }  
785 }  
786 }  
787 }  
788 }  
789 }  
790 }  
791 }  
792 }  
793 }  
794 }  
795 }  
796 }  
797 }  
798 }  
799 }  
800 }  
801 }  
802 }  
803 }  
804 }  
805 }  
806 }  
807 }  
808 }  
809 }  
810 }  
811 }  
812 }  
813 }  
814 }  
815 }  
816 }  
817 }  
818 }  
819 }  
820 }  
821 }  
822 }  
823 }  
824 }  
825 }  
826 }  
827 }  
828 }  
829 }  
830 }  
831 }  
832 }  
833 }  
834 }  
835 }  
836 }  
837 }  
838 }  
839 }  
840 }  
841 }  
842 }  
843 }  
844 }  
845 }  
846 }  
847 }  
848 }  
849 }  
850 }  
851 }  
852 }  
853 }  
854 }  
855 }  
856 }  
857 }  
858 }  
859 }  
860 }  
861 }  
862 }  
863 }  
864 }  
865 }  
866 }  
867 }  
868 }  
869 }  
870 }  
871 }  
872 }  
873 }  
874 }  
875 }  
876 }  
877 }  
878 }  
879 }  
880 }  
881 }  
882 }  
883 }  
884 }  
885 }  
886 }  
887 }  
888 }  
889 }  
890 }  
891 }  
892 }  
893 }  
894 }  
895 }  
896 }  
897 }  
898 }  
899 }  
900 }  
901 }  
902 }  
903 }  
904 }  
905 }  
906 }  
907 }  
908 }  
909 }  
910 }  
911 }  
912 }  
913 }  
914 }  
915 }  
916 }  
917 }  
918 }  
919 }  
920 }  
921 }  
922 }  
923 }  
924 }  
925 }  
926 }  
927 }  
928 }  
929 }  
930 }  
931 }  
932 }  
933 }  
934 }  
935 }  
936 }  
937 }  
938 }  
939 }  
940 }  
941 }  
942 }  
943 }  
944 }  
945 }  
946 }  
947 }  
948 }  
949 }  
950 }  
951 }  
952 }  
953 }  
954 }  
955 }  
956 }  
957 }  
958 }  
959 }  
960 }  
961 }  
962 }  
963 }  
964 }  
965 }  
966 }  
967 }  
968 }  
969 }  
970 }  
971 }  
972 }  
973 }  
974 }  
975 }  
976 }  
977 }  
978 }  
979 }  
980 }  
981 }  
982 }  
983 }  
984 }  
985 }  
986 }  
987 }  
988 }  
989 }  
990 }  
991 }  
992 }  
993 }  
994 }  
995 }  
996 }  
997 }  
998 }  
999 }
```

4. Ensuite, entrez une chaîne de recherche : `search=river&$count=true`.

Cette requête appelle la recherche en texte intégral sur le terme `river` et le résultat inclut le nombre de documents correspondants. Connaître le nombre de documents correspondants est utile dans les scénarios de test lorsque vous avez un index de grande taille comportant des milliers, voire des millions de documents. Dans ce cas, un seul document correspond à la requête.

5. Enfin, entrez une chaîne de recherche qui limite la sortie JSON aux champs pertinents :

```
search=river&$count=true&$select=hotelId, baseRate, description
```

La réponse à la requête se réduit aux champs sélectionnés, ce qui entraîne une sortie plus concise.

## Réinitialiser et réexécuter

Dans les premières étapes expérimentales du développement, l'approche la plus pratique pour les itérations de

conception consiste à supprimer les objets de Recherche cognitive Azure et à autoriser votre code à les reconstruire. Les noms des ressources sont uniques. La suppression d'un objet vous permet de le recréer en utilisant le même nom.

L'exemple de code pour ce tutoriel recherche les objets existants et les supprime pour vous permettre de réexécuter votre code.

Vous pouvez également utiliser le portail pour supprimer les index, les indexeurs et les sources de données.

## Nettoyer les ressources

Lorsque vous travaillez dans votre propre abonnement, il est judicieux à la fin d'un projet de supprimer les ressources dont vous n'avez plus besoin. Les ressources laissées en cours d'exécution peuvent vous coûter de l'argent. Vous pouvez supprimer les ressources une par une, ou choisir de supprimer le groupe de ressources afin de supprimer l'ensemble des ressources.

Vous pouvez rechercher et gérer les ressources dans le portail à l'aide des liens Toutes les ressources ou Groupes de ressources situés dans le volet de navigation de gauche.

## Étapes suivantes

Maintenant que vous êtes familiarisé avec les principes fondamentaux de l'indexation de SQL Database, examinons de plus près la configuration de l'indexeur.

[Configurer un indexeur SQL Database](#)

# Tutoriel : Indexer des objets blob JSON à partir de Stockage Azure avec REST

04/10/2020 • 18 minutes to read • [Edit Online](#)

La recherche cognitive Azure peut indexer des tableaux et documents JSON dans le stockage d'objets blob Azure à l'aide d'un [indexeur](#) qui sait comment lire des données semi-structurées. Les données semi-structurées contiennent des balises ou des marquages qui séparent le contenu au sein des données. Elles séparent les données non structurées, qui doivent être indexées entièrement, des données formellement structurées qui respectent un modèle de données (tel qu'un schéma de base de données relationnelle), qui peuvent être indexées par champ.

Ce tutoriel utilise Postman et les [API REST de Recherche](#) pour effectuer les tâches suivantes :

- Configurer une source de données de recherche cognitive Azure pour un conteneur d'objets blob Azure
- Créer un index de recherche cognitive Azure où stocker le contenu pouvant faire l'objet d'une recherche
- Configurer et exécuter un indexeur pour lire le conteneur et extraire le contenu de recherche à partir du stockage d'objets blob Azure
- Effectuer une recherche dans l'index que vous venez de créer

Si vous n'avez pas d'abonnement Azure, créez un [compte gratuit](#) avant de commencer.

## Prérequis

- [Stockage Azure](#)
- [Application de bureau Postman](#)
- [Créer ou rechercher un service de recherche existant](#)

### NOTE

Vous pouvez utiliser le service gratuit pour ce tutoriel. Avec un service de recherche gratuit, vous êtes limité à trois index, trois indexeurs et trois sources de données. Ce didacticiel crée une occurrence de chaque élément. Avant de commencer, veillez à disposer de l'espace suffisant sur votre service pour accepter les nouvelles ressources.

## Télécharger les fichiers

[Clinical-trials-json.zip](#) contient les données utilisées dans ce tutoriel. Téléchargez et décompressez ce fichier dans son propre dossier. Les données proviennent de [clinicaltrials.gov](#), converties au format JSON pour ce tutoriel.

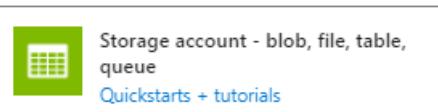
## 1 - Créer les services

Ce tutoriel utilise Recherche cognitive Azure pour l'indexation et les requêtes, et Stockage Blob Azure pour fournir les données.

Si possible, créez les deux services dans la même région et le même groupe de ressources pour des raisons de proximité et de facilité de gestion. Dans la pratique, votre compte de stockage Azure peut être dans une région quelconque.

### Démarrer avec le stockage Azure

1. [Connectez-vous au portail Azure](#) et cliquez sur + **Créer une ressource**.
2. Recherchez un *compte de stockage* et sélectionnez l'offre de compte de stockage Microsoft.



3. Sous l'onglet Bases, les éléments suivants sont obligatoires. Acceptez les valeurs par défaut pour tout le reste.

- **Groupe de ressources.** Sélectionnez un groupe existant ou créez-en un, mais utilisez le même groupe pour tous les services afin de pouvoir les gérer collectivement.
- **Nom du compte de stockage.** Si vous pensez que vous pouvez avoir plusieurs ressources du même type, utilisez le nom pour lever l'ambiguïté par type et par région, par exemple *blobstoragewestus*.
- **Emplacement.** Si possible, choisissez le même emplacement que celui utilisé pour Recherche cognitive Azure et Cognitive Services. Un emplacement unique annule les frais liés à la bande passante.
- **Type de compte.** Choisissez la valeur par défaut, *StorageV2 (v2 universel)*.

4. Cliquez sur **Vérifier + créer** pour créer le service.

5. Une fois qu'il est créé, cliquez sur **Accéder à la ressource** pour ouvrir la page Vue d'ensemble.

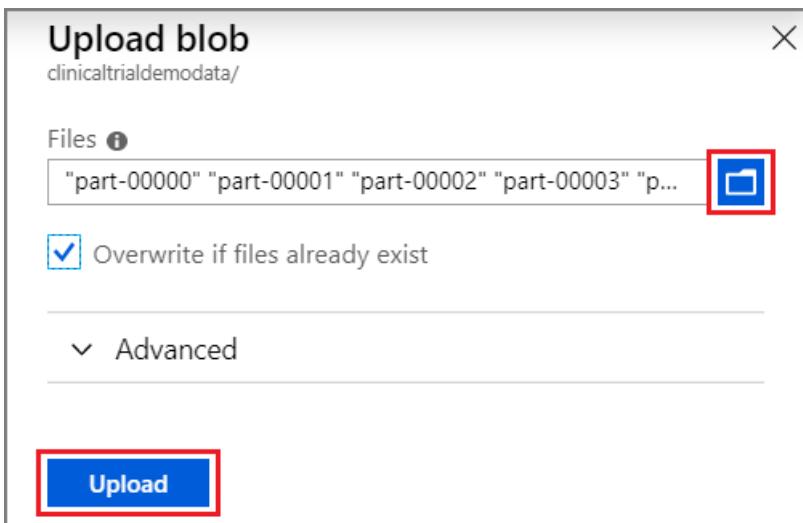
6. Cliquez sur le service **Objets blob**.

7. **Créez un conteneur d'objets blob** pour contenir des exemples de données. Vous pouvez définir le niveau d'accès public sur l'une de ses valeurs valides.

8. Une fois le conteneur créé, ouvrez-le et sélectionnez **Charger** dans la barre de commandes.



9. Accédez au dossier contenant les exemples de fichiers. Sélectionnez-les tous, puis cliquez sur **Charger**.



Une fois le chargement terminé, les fichiers doivent apparaître dans leur propre sous-dossier au sein du conteneur de données.

#### Recherche cognitive Azure

La ressource suivante est Recherche cognitive Azure, que vous pouvez [créer dans le portail](#). Vous pouvez utiliser le niveau gratuit pour effectuer cette procédure pas à pas.

Comme avec le stockage Blob Azure, prenez un moment pour collecter la clé d'accès. Par ailleurs, lorsque vous commencez à structurer les demandes, vous devez fournir le point de terminaison et la clé API d'administration utilisés pour authentifier chaque demande.

### Obtenir une clé et une URL

Les appels REST requièrent l'URL du service et une clé d'accès et ce, sur chaque demande. Un service de recherche est créé avec les deux. Ainsi, si vous avez ajouté la Recherche cognitive Azure à votre abonnement, effectuez ce qui suit pour obtenir les informations nécessaires :

1. [Connectez-vous au portail Azure](#), puis dans la page **Vue d'ensemble** du service de recherche, récupérez l'URL. Voici un exemple de point de terminaison : `https://mydemo.search.windows.net`.
2. Dans **Paramètres > Clés**, obtenez une clé d'administration pour avoir des droits d'accès complets sur le service. Il existe deux clés d'administration interchangeables, fournies pour assurer la continuité de l'activité au cas où vous deviez en remplacer une. Vous pouvez utiliser la clé primaire ou secondaire sur les demandes d'ajout, de modification et de suppression d'objets.

The screenshot shows two overlapping Azure portal pages. The top page is titled 'Tableau de bord > mydemo' and displays the 'Vue d'ensemble' (Overview) section. It includes a search bar, navigation links like 'Ajouter un index', 'Importeur...', 'Explorateur...', 'Actualiser', 'Supprimer', and 'Déplacer'. A red box highlights the 'Vue d'ensemble' link in the left sidebar. To its right, a red box highlights the 'URL' field containing 'https://mydemo.search.windows.net'. A red circle with the number '1' is positioned over the URL field. The bottom page is titled 'Tableau de bord > mydemo - Clés' and displays the 'Clés' (Keys) section. It has a search bar and two buttons: 'Régénérer la clé primaire' and 'Régénérer la clé secondaire'. Below these are two fields: 'Clé d'administration primaire' and 'Clé d'administration secondaire', both containing placeholder text '<placeholder-for-alphanumeric-autogenerated-string>'. A red box highlights the 'Clés' link in the left sidebar. A red circle with the number '2' is positioned over the 'Clés' link. The left sidebar of the bottom page also lists 'Vue d'ensemble', 'Journal d'activité', 'Contrôle d'accès (IAM)', 'Paramètres', 'Démarrage rapide', and 'Échelle'.

Toutes les demandes nécessitent une clé API sur chaque demande envoyée à votre service. L'utilisation d'une clé valide permet d'établir, en fonction de chaque demande, une relation de confiance entre l'application qui envoie la demande et le service qui en assure le traitement.

## 2 - Configurer Postman

Démarrez Postman et paramétrez une requête HTTP. Si vous ne connaissez pas bien cet outil, consultez [Explorer les API REST de Recherche cognitive Azure avec Postman](#).

Les méthodes de requête utilisées pour chaque appel dans ce tutoriel sont **POST** et **GET**. Vous allez faire trois appels d'API à votre service de recherche pour créer une source de données, un index et un indexeur. La source de données inclut un pointeur vers votre compte de stockage et vos données JSON. Votre service de recherche établit la connexion lors du chargement des données.

Dans les en-têtes, définissez « Content-type » sur `application/json` et `api-key` sur la clé API d'administration de votre service Recherche cognitive Azure. Une fois que vous avez défini les en-têtes, vous pouvez les utiliser pour chaque demande de cet exercice.

GET https://mydemo.search.windows.net/indexes?api-version=2020-06-30

Params Envoyer Enregistrer

Autorisation En-têtes (2) Corps Script de pré-requête Tests Coder

Clé	Valeur	Description	... Modifier en bloc	Présélections ▾
<input checked="" type="checkbox"/> Content-Type	application/json			
<input checked="" type="checkbox"/> api-key	<placeholder-api-key-for-your-service>			

Nouvelle clé Valeur Description

Corps Cookies En-têtes (14)

État : 200 OK Heure : 540 ms

Les URI doivent spécifier un élément « api-version » et chaque appel doit retourner une réponse 201 Créé. L'api-version généralement disponible pour utiliser les tableaux JSON est 2020-06-30.

### 3 - Créer une source de données

L'[API Create Data Source](#) crée un objet Recherche cognitive Azure qui spécifie les données à indexer.

- Définissez le point de terminaison de cet appel sur

`https://[service name].search.windows.net/datasources?api-version=2020-06-30`. Remplacez [service name] par le nom de votre service de recherche.

- Copiez le code JSON suivant dans le corps de la requête.

```
{
    "name" : "clinical-trials-json-ds",
    "type" : "azureblob",
    "credentials" : { "connectionString" : "DefaultEndpointsProtocol=https;AccountName=[storage account name];AccountKey=[storage account key];" },
    "container" : { "name" : "[blob container name]" }
}
```

- Remplacez la chaîne de connexion par une chaîne valide pour votre compte.

- Remplacez « [blob container name] » par le conteneur que vous avez créé pour l'exemple de données.

- Envoyez la demande. La réponse doit ressembler à ce qui suit :

```
{
    "@odata.context": "https://exampleurl.search.windows.net/$metadata#datasources/$entity",
    "@odata.etag": "\"0x8D505FBC3856C9E\"",
    "name": "clinical-trials-json-ds",
    "description": null,
    "type": "azureblob",
    "subtype": null,
    "credentials": {
        "connectionString": "DefaultEndpointsProtocol=https;AccountName=[mystorageaccounthere];AccountKey=[[myaccountkeyhere]];" },
    "container": {
        "name": "[mycontainernamehere]",
        "query": null
    },
    "dataChangeDetectionPolicy": null,
    "dataDeletionDetectionPolicy": null
}
```

### 4 - Créer un index

Le deuxième appel est celui de l'[API de création d'index](#), qui permet de créer un index de recherche cognitive Azure

pour stocker toutes les données pouvant faire l'objet de recherches. Un index spécifie tous les paramètres et leurs attributs.

1. Définissez le point de terminaison de cet appel sur

`https://[service name].search.windows.net/indexes?api-version=2020-06-30`. Remplacez `[service name]` par le nom de votre service de recherche.

2. Copiez le code JSON suivant dans le corps de la requête.

```
{
  "name": "clinical-trials-json-index",
  "fields": [
    {"name": "FileName", "type": "Edm.String", "searchable": false, "retrievable": true, "facetable": false, "filterable": false, "sortable": true},
    {"name": "Description", "type": "Edm.String", "searchable": true, "retrievable": false, "facetable": false, "filterable": false, "sortable": false},
    {"name": "MinimumAge", "type": "Edm.Int32", "searchable": false, "retrievable": true, "facetable": true, "filterable": true, "sortable": true},
    {"name": "Title", "type": "Edm.String", "searchable": true, "retrievable": true, "facetable": false, "filterable": true, "sortable": true},
    {"name": "URL", "type": "Edm.String", "searchable": false, "retrievable": false, "facetable": false, "filterable": false, "sortable": false},
    {"name": "MyURL", "type": "Edm.String", "searchable": false, "retrievable": true, "facetable": false, "filterable": false, "sortable": false},
    {"name": "Gender", "type": "Edm.String", "searchable": false, "retrievable": true, "facetable": true, "filterable": true, "sortable": false},
    {"name": "MaximumAge", "type": "Edm.Int32", "searchable": false, "retrievable": true, "facetable": true, "filterable": true, "sortable": true},
    {"name": "Summary", "type": "Edm.String", "searchable": true, "retrievable": true, "facetable": false, "filterable": false, "sortable": false},
    {"name": "NCTID", "type": "Edm.String", "key": true, "searchable": true, "retrievable": true, "facetable": false, "filterable": true, "sortable": true},
    {"name": "Phase", "type": "Edm.String", "searchable": false, "retrievable": true, "facetable": true, "filterable": true, "sortable": false},
    {"name": "Date", "type": "Edm.String", "searchable": false, "retrievable": true, "facetable": false, "filterable": false, "sortable": true},
    {"name": "OverallStatus", "type": "Edm.String", "searchable": false, "retrievable": true, "facetable": true, "filterable": true, "sortable": false},
    {"name": "OrgStudyId", "type": "Edm.String", "searchable": true, "retrievable": true, "facetable": false, "filterable": true, "sortable": false},
    {"name": "HealthyVolunteers", "type": "Edm.String", "searchable": false, "retrievable": true, "facetable": true, "filterable": true, "sortable": false},
    {"name": "Keywords", "type": "Collection(Edm.String)", "searchable": true, "retrievable": true, "facetable": true, "filterable": false, "sortable": false},
    {"name": "metadata_storage_last_modified", "type": "Edm.DateTimeOffset", "searchable": false, "retrievable": true, "facetable": true, "filterable": true, "sortable": false},
    {"name": "metadata_storage_size", "type": "Edm.String", "searchable": false, "retrievable": true, "facetable": true, "filterable": false, "sortable": false},
    {"name": "metadata_content_type", "type": "Edm.String", "searchable": true, "retrievable": true, "facetable": true, "filterable": true, "sortable": false}
  ]
}
```

3. Envoyez la demande. La réponse doit ressembler à ce qui suit :

```
{
    "@odata.context": "https://exampleurl.search.windows.net/$metadata#indexes/$entity",
    "@odata.etag": "\0x8D505FC00EDD5FA\"",
    "name": "clinical-trials-json-index",
    "fields": [
        {
            "name": "FileName",
            "type": "Edm.String",
            "searchable": false,
            "filterable": false,
            "retrievable": true,
            "sortable": true,
            "facetable": false,
            "key": false,
            "indexAnalyzer": null,
            "searchAnalyzer": null,
            "analyzer": null,
            "synonymMaps": []
        },
        {
            "name": "Description",
            "type": "Edm.String",
            "searchable": true,
            "filterable": false,
            "retrievable": false,
            "sortable": false,
            "facetable": false,
            "key": false,
            "indexAnalyzer": null,
            "searchAnalyzer": null,
            "analyzer": null,
            "synonymMaps": []
        },
        ...
    }
}
```

## 5 - Créer et exécuter un indexeur

Un indexeur se connecte à la source de données, importe les données dans l'index de recherche cible, et peut fournir une planification afin d'automatiser l'actualisation des données. L'API REST est celle qui consiste à [créer un indexeur](#).

- Définissez l'URI pour cet appel sur

`https://[service name].search.windows.net/indexers?api-version=2020-06-30`. Remplacez `[service name]` par le nom de votre service de recherche.

- Copiez le code JSON suivant dans le corps de la requête.

```
{
    "name" : "clinical-trials-json-indexer",
    "dataSourceName" : "clinical-trials-json-ds",
    "targetIndexName" : "clinical-trials-json-index",
    "parameters" : { "configuration" : { "parsingMode" : "jsonArray" } }
}
```

- Envoyez la demande. La requête est traitée immédiatement. Une fois la réponse reçue, vous aurez un index de recherche en texte intégral consultable. La réponse doit ressembler à ce qui suit :

```
{  
    "@odata.context": "https://exampleurl.search.windows.net/$metadata#indexers/$entity",  
    "@odata.etag": "\"0x8D505FDE143D164\"",  
    "name": "clinical-trials-json-indexer",  
    "description": null,  
    "dataSourceName": "clinical-trials-json-ds",  
    "targetIndexName": "clinical-trials-json-index",  
    "schedule": null,  
    "parameters": {  
        "batchSize": null,  
        "maxFailedItems": null,  
        "maxFailedItemsPerBatch": null,  
        "base64EncodeKeys": null,  
        "configuration": {  
            "parsingMode": "jsonArray"  
        }  
    },  
    "fieldMappings": [],  
    "enrichers": [],  
    "disabled": null  
}
```

## 6- Rechercher dans vos fichiers JSON

Vous pouvez démarrer la recherche dès que le premier document est chargé.

1. Remplacez le verbe par GET.
2. Définissez l'URI pour cet appel sur

```
https://[service name].search.windows.net/indexes/clinical-trials-json-index/docs?search=*&api-version=2020-06-30&$count=true
```

. Remplacez [service name] par le nom de votre service de recherche.

3. Envoyez la demande. Il s'agit d'une requête de recherche en texte intégral non spécifiée qui retourne tous les champs marqués comme récupérables dans l'index, ainsi qu'un nombre de documents. La réponse doit ressembler à ce qui suit:

```
{
    "@odata.context": "https://exampleurl.search.windows.net/indexes('clinical-trials-json-index')/$metadata#docs(*)",
    "@odata.count": 100,
    "value": [
        {
            "@search.score": 1.0,
            "FileName": "NCT00000102.txt",
            "MinimumAge": 14,
            "Title": "Congenital Adrenal Hyperplasia: Calcium Channels as Therapeutic Targets",
            "MyURL": "https://azure.storagedemos.com/clinical-trials/NCT00000102.txt",
            "Gender": "Both",
            "MaximumAge": 35,
            "Summary": "This study will test the ability of extended release nifedipine (Procardia XL), a blood pressure medication, to permit a decrease in the dose of glucocorticoid medication children take to treat congenital adrenal hyperplasia (CAH).",
            "NCTID": "NCT00000102",
            "Phase": "Phase 1/Phase 2",
            "Date": "ClinicalTrials.gov processed this data on October 25, 2016",
            "OverallStatus": "Completed",
            "OrgStudyId": "NCRR-M01RR01070-0506",
            "HealthyVolunteers": "No",
            "Keywords": [],
            "metadata_storage_last_modified": "2019-04-09T18:16:24Z",
            "metadata_storage_size": "33060",
            "metadata_content_type": null
        },
        ...
    ]
}
```

4. Ajoutez le paramètre de requête `$select` pour limiter les résultats à moins de champs :

```
https://[service name].search.windows.net/indexes/clinical-trials-json-index/docs?search=*&$select=Gender,metadata_storage_size&api-version=2020-06-30&$count=true
```

. Pour cette requête, 100 documents correspondent, mais par défaut, Recherche cognitive Azure retourne seulement 50 dans les résultats.

Line Number	Content
3	"@odata.count": 100,
4	"value": [
5	{
6	"@search.score": 1.0,
7	"Gender": "Both",
8	"metadata_storage_size": "33060"
9	},
10	{
11	"@search.score": 1.0,
12	"Gender": "Both",
13	"metadata_storage_size": "34219"
14	},

5. `$filter=MinimumAge ge 30 and MaximumAge lt 75` est un exemple de requête plus complexe qui retourne seulement les résultats pour lesquels le paramètre MinimumAge est supérieur ou égal à 30 et le paramètre MaximumAge inférieur à 75. Remplacez l'expression `$select` par l'expression `$filter`.

The screenshot shows a GET request to [https://exampleurl.search.windows.net/indexes/cClinical-trials-json-index/docs?search=\\*&\\$filter=MinimumAge ge 30 and Maxim...](https://exampleurl.search.windows.net/indexes/cClinical-trials-json-index/docs?search=*&$filter=MinimumAge ge 30 and Maxim...). The results are displayed in JSON format, showing 10 items. One item is highlighted with a red box, which contains the following data:

```
3     "@odata.count": 10,
4     "value": [
5         {
6             "@search.score": 1.0,
7             "FileName": "NCT00000161.txt",
8             "MinimumAge": 45,
9             "Title": "Randomized Trials of Vitamin Supplements and Eye Disease",
10            "MyURL": "https://azure.storagedemos.com/clinical-trials/NCT00000161.txt",
11            "Gender": "Female",
12            "MaximumAge": 0,
13            "Summary": "To determine whether vitamin E supplementation reduces the risk of cataract and age-related macular degeneration in women. To determine whether vitamin C supplementation reduces the risk of cataract and AMD in women. To determine whether beta-carotene supplementation reduces the risk of cataract and AMD in women. To determine whether alternative treatments reduce the risk of cataract and AMD in women. To identify potential risk factors for cataract and AMD including alcohol intake, blood pressure, blood cholesterol, cardiovascular disease, height, body mass index, and dietary factors.",
14            "NCTID": "NCT00000161",
15            "Phase": "Phase 3",
16            "Date": "ClinicalTrials.gov processed this data on October 25, 2016",
17            "OverallStatus": "Active, not recruiting",
18            "OrgStudyId": "NEI-63",
19            "HealthyVolunteers": "Accepts Healthy Volunteers",
20            "Keywords": [
21                "Age-Related Macular Degeneration"
22            ],
23        }
24    ]
25 }
```

Vous pouvez aussi utiliser des opérateurs logiques (AND, OR, NOT) et des opérateurs de comparaison (eq, ne, gt, lt, ge, le). Les comparaisons de chaînes sont sensibles à la casse. Pour plus d'informations et d'exemples, consultez [Créer une requête simple](#).

#### NOTE

Le paramètre `$filter` fonctionne uniquement avec des métadonnées qui ont été marquées comme « filtrables » lors de la création de l'index.

## Réinitialiser et réexécuter

Dans les premières étapes expérimentales du développement, l'approche la plus pratique pour les itérations de conception consiste à supprimer les objets de Recherche cognitive Azure et à autoriser votre code à les reconstruire. Les noms des ressources sont uniques. La suppression d'un objet vous permet de le recréer en utilisant le même nom.

Vous pouvez utiliser le portail pour supprimer les index, les indexeurs et les sources de données. Vous pouvez aussi utiliser **DELETE** et fournir des URL vers chaque objet. La commande suivante supprime un indexeur.

```
DELETE https://[YOUR-SERVICE-NAME].search.windows.net/indexers/cClinical-trials-json-indexer?api-version=2020-06-30
```

Le code d'état 204 est retourné lorsque la suppression réussit.

## Nettoyer les ressources

Lorsque vous travaillez dans votre propre abonnement, il est judicieux à la fin d'un projet de supprimer les ressources dont vous n'avez plus besoin. Les ressources laissées en cours d'exécution peuvent vous coûter de l'argent. Vous pouvez supprimer les ressources une par une, ou choisir de supprimer le groupe de ressources afin de supprimer l'ensemble des ressources.

Vous pouvez rechercher et gérer les ressources dans le portail à l'aide des liens Toutes les ressources ou Groupes de ressources situés dans le volet de navigation de gauche.

## Étapes suivantes

Maintenant que vous êtes familiarisé avec les principes fondamentaux de l'indexation de Stockage Blob Azure, examinons de plus près la configuration de l'indexeur pour les objets blob JSON dans Stockage Azure.

[Configurer l'indexation d'objets blob JSON](#)

# Tutoriel : Indexer à partir de plusieurs sources de données à l'aide du SDK .NET

04/10/2020 • 26 minutes to read • [Edit Online](#)

Recherche cognitive Azure peut importer, analyser et indexer des données provenant de plusieurs sources de données dans un même index de recherche consolidé. Cette fonctionnalité prend en charge les situations où des données structurées sont agrégées avec des données moins structurées, voire de texte brut, provenant d'autres sources, telles que des documents de texte, HTML ou JSON.

Ce tutoriel explique comment indexer des données d'hôtels provenant d'une source de données Azure Cosmos DB et les fusionner avec des détails sur les chambres d'hôtel tirés de documents Stockage Blob Azure. Le résultat est un index de recherche d'hôtel combiné contenant des types de données complexes.

Ce tutoriel utilise C# et le [SDK .NET](#). Dans ce tutoriel, vous allez effectuer les tâches suivantes :

- Charger des exemples de données et créer des sources de données
- Identifier la clé de document
- Définir et créer l'index
- Indexer les données d'hôtels issues d'Azure Cosmos DB
- Fusionner les données des chambres d'hôtel provenant du stockage d'objets blob

Si vous n'avez pas d'abonnement Azure, créez un [compte gratuit](#) avant de commencer.

## Prérequis

- [Azure Cosmos DB](#)
- [Stockage Azure](#)
- [Visual Studio 2019](#)
- [Créer ou rechercher un service de recherche existant](#)

### NOTE

Vous pouvez utiliser le service gratuit pour ce tutoriel. Avec un service de recherche gratuit, vous êtes limité à trois index, trois indexeurs et trois sources de données. Ce didacticiel crée une occurrence de chaque élément. Avant de commencer, veillez à disposer de l'espace suffisant sur votre service pour accepter les nouvelles ressources.

## Télécharger les fichiers

Le code source pour ce tutoriel se trouve dans le dépôt GitHub [azure-search-dotnet-samples](#), dans le dossier [multiple-data-sources](#).

## 1 - Créer les services

Ce tutoriel utilise Recherche cognitive Azure pour l'indexation et les requêtes, Azure Cosmos DB pour un des jeux de données et Stockage Blob Azure pour le deuxième jeu de données.

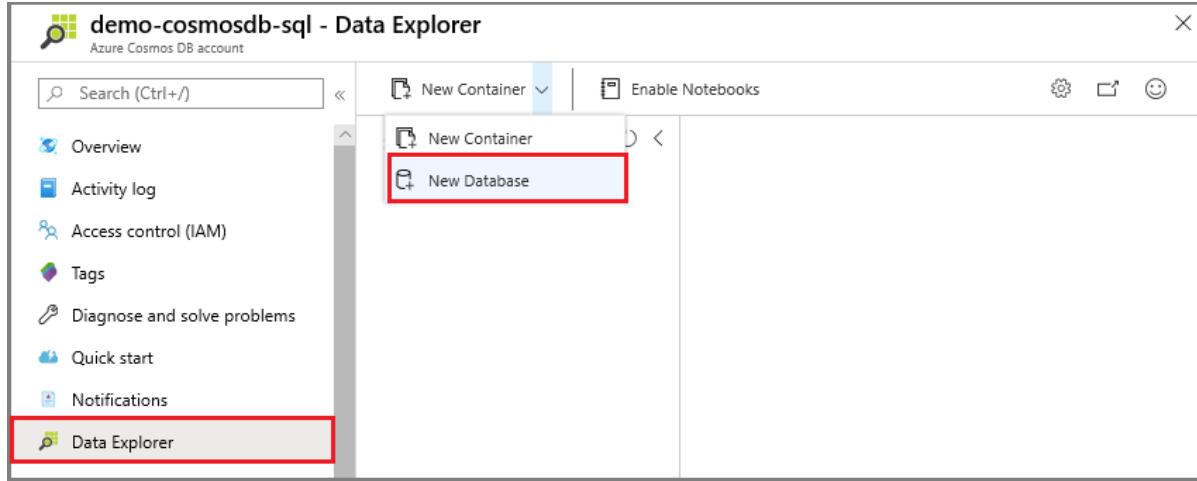
Si possible, créez tous les services dans la même région et le même groupe de ressources pour des raisons de proximité et de facilité de gestion. Dans la pratique, vos services peuvent se trouver dans n'importe quelle région.

Cet exemple utilise deux petits ensembles de données qui décrivent sept hôtels fictifs. Un ensemble décrit les hôtels

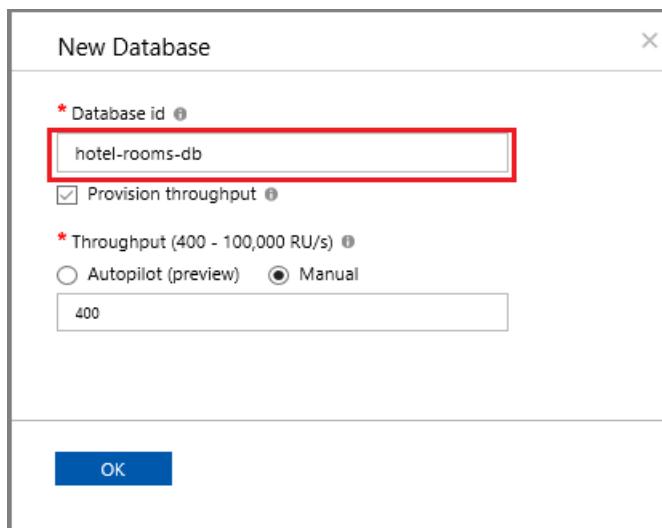
eux-mêmes et est chargé dans une base de données Azure Cosmos DB. L'autre ensemble contient les détails des chambres d'hôtel et est fourni sous la forme de sept fichiers JSON distincts à charger dans Stockage Blob Azure.

## Démarrer avec Cosmos DB

1. Connectez-vous au [portail Azure](#), puis accédez à la page Vue d'ensemble de votre compte Azure Cosmos DB.
2. Sélectionnez **Explorateur de données**, puis sélectionnez **Nouvelle base de données**.



3. Entrez le nom **hotel-rooms-db**. Acceptez les valeurs par défaut pour les autres paramètres.



4. Créez un conteneur. Utilisez la base de données existante que vous venez de créer. Entrez **hotels** comme nom de conteneur et utilisez **/HotelId** comme clé de partition.

Add Container

\* Database id ⓘ  
 Create new  Use existing  
 hotel-rooms-db

\* Container id ⓘ

\* Partition key ⓘ

My partition key is larger than 100 bytes

Provision dedicated throughput for this container ⓘ

Unique keys ⓘ  
 + Add unique key

OK

5. Sélectionnez Éléments sous Hôtels, puis cliquez sur Charger un élément dans la barre de commandes. Accédez au fichier **cosmosdb/HotelsDataSubset\_CosmosDb.json** dans le dossier du projet, puis sélectionnez-le.

SQL API

hotel-rooms-db

hotels

Items

Upload Items

Select JSON Files ⓘ

Upload

6. Utilisez le bouton Actualiser pour actualiser l'affichage des éléments dans la collection hotels. Sept nouveaux documents doivent être listés.

### Stockage Blob Azure

- Connectez-vous au [portail Azure](#), accédez à votre compte de stockage Azure, cliquez sur **Objets blob**, puis sur **+ Conteneur**.
- [Créez un conteneur d'objets blob](#) nommé **hotel-rooms** pour stocker les exemples de fichiers JSON des chambres d'hôtel. Vous pouvez définir le niveau d'accès public sur l'une de ses valeurs valides.

New container

Name \*

hotel-rooms

Public access level ⓘ

Private (no anonymous access)

OK Cancel

- Créer un
- Une fois le conteneur créé, ouvrez-le et sélectionnez **Charger** dans la barre de commandes. Accédez au dossier contenant les exemples de fichiers. Sélectionnez-les tous, puis cliquez sur **Charger**.

hotel-rooms

Upload blob

Files ⓘ

"Rooms1.json" "Rooms10.json" "Rooms11.json" "Roo..."

Overwrite if files already exist

Advanced

Upload

Une fois le chargement terminé, les fichiers doivent apparaître dans la liste liée au conteneur de données.

## Recherche cognitive Azure

Le troisième composant est Recherche cognitive Azure, que vous pouvez [créer dans le portail](#). Vous pouvez utiliser le niveau gratuit pour effectuer cette procédure pas à pas.

### Obtenir une clé API d'administration et une URL pour Recherche cognitive Azure

Pour interagir avec votre service Recherche cognitive Azure, vous devrez disposer de l'URL du service et d'une clé d'accès. Un service de recherche est créé avec les deux. Ainsi, si vous avez ajouté la Recherche cognitive Azure à votre abonnement, effectuez ce qui suit pour obtenir les informations nécessaires :

- Connectez-vous au [portail Azure](#), puis dans la page **Vue d'ensemble** du service de recherche, récupérez l'URL. Voici un exemple de point de terminaison : <https://mydemo.search.windows.net>.
- Dans **Paramètres > Clés**, obtenez une clé d'administration pour avoir des droits d'accès complets sur le service. Il existe deux clés d'administration interchangeables, fournies pour assurer la continuité de l'activité au cas où vous deviez en remplacer une. Vous pouvez utiliser la clé primaire ou secondaire sur les demandes d'ajout, de modification et de suppression d'objets.

Obtenez aussi la clé de requête. Il est recommandé d'émettre des demandes de requête avec un accès en lecture seule.

L'utilisation d'une clé valide permet d'établir, en fonction de chaque demande, une relation de confiance entre l'application qui envoie la demande et le service qui en assure le traitement.

## 2 - Configurer votre environnement

1. Démarrez Visual Studio 2019 puis, dans le menu Outils, sélectionnez **Gestionnaire de package NuGet**, puis **Gérer les packages NuGet pour la solution...**.
2. Sous l'onglet **Parcourir**, recherchez et installez **Microsoft.Azure.Search** (version 9.0.1 ou ultérieure). Vous devrez valider une série de boîtes de dialogue supplémentaires pour terminer l'installation.

3. Recherchez le package NuGet **Microsoft.Extensions.Configuration.Json** et installez-le également.

4. Ouvrez le fichier solution **AzureSearchMultipleDataSources.sln**.
5. Dans l'Explorateur de solutions, modifiez le fichier **appsettings.json** en y ajoutant les informations de connexion.

```
{
  "SearchServiceName": "Put your search service name here",
  "SearchServiceAdminApiKey": "Put your primary or secondary API key here",
  "BlobStorageAccountName": "Put your Azure Storage account name here",
  "BlobStorageConnectionString": "Put your Azure Blob Storage connection string here",
  "CosmosDBConnectionString": "Put your Cosmos DB connection string here",
  "CosmosDBDatabaseName": "hotel-rooms-db"
}
```

Les deux premières entrées utilisent l'URL et les clés d'administration de votre service Recherche cognitive Azure. Si le point de terminaison est `https://mydemo.search.windows.net`, le nom du service à fournir est `mydemo`.

Les entrées suivantes spécifient les noms de compte et les informations de chaîne de connexion pour les sources de données Stockage Blob Azure et Azure Cosmos DB.

### 3 - Mapper les champs clés

La fusion de contenu nécessite que les deux flux de données ciblent les mêmes documents dans l'index de recherche.

Dans Recherche cognitive Azure, le champ de clé identifie chaque document de façon univoque. Chaque index de recherche doit comporter exactement un champ de clé de type `Edm.String`. Ce champ de clé doit être présent pour chaque document d'une source de données qui est ajouté à l'index. (il constitue en fait le seul champ obligatoire).

Quand vous indexez des données provenant de plusieurs sources de données, vérifiez que chaque ligne ou document entrant contient une clé de document commune pour les fusionner à partir de deux documents sources physiquement distincts dans un nouveau document de recherche dans l'index combiné.

Il est souvent nécessaire de passer par une planification préalable pour identifier une clé de document significative pour votre index et vous assurer qu'elle existe dans les deux sources de données. Dans cette démonstration, la clé `HotelId` de chaque hôtel dans Cosmos DB est également présente dans les objets blob JSON des chambres dans Stockage Blob.

Les indexeurs de la Recherche cognitive Azure peuvent utiliser des mappages de champs pour renommer, voire remettre en forme, les champs de données pendant le processus d'indexation, afin que les données sources puissent être redirigées vers le champ de l'index approprié. Par exemple, dans Cosmos DB, l'identificateur d'hôtel est appelé `HotelId`. Mais dans les fichiers d'objets blob JSON pour les chambres d'hôtel, l'identificateur d'hôtel est nommé `Id`. Le programme gère cette situation en mappant le champ `Id` des objets blob au champ de clé `HotelId` de l'index.

#### NOTE

Dans la plupart des cas, les clés de document générées automatiquement, comme celles créées par défaut par certains indexeurs, ne constituent pas des clés de document appropriées pour les index combinés. En général, il est préférable d'utiliser une valeur de clé unique explicite qui existe déjà dans vos sources de données ou qui peut y être facilement ajoutée.

### 4 - Explorer le code

Une fois que les données et les paramètres de configuration sont en place, l'exemple de programme dans **AzureSearchMultipleDataSources.sln** doit être opérationnel.

Cette application console C#/.NET simple effectue les tâches suivantes :

- Elle crée un index basé sur la structure de données de la classe Hotel C# (qui référence également les classes Address et Room).
- Elle crée une source de données et un indexeur qui mappe les données Azure Cosmos DB aux champs d'index. Il s'agit de deux objets dans le service Recherche cognitive Azure.
- Elle exécute l'indexeur pour charger les données des hôtels à partir de Cosmos DB.
- Elle crée une seconde source de données et un indexeur qui mappe les données blob JSON aux champs d'index.
- Elle exécute le second indexeur pour charger des données des chambres à partir du stockage d'objets blob.

Avant d'exécuter le programme, prenez une minute pour étudier le code et les définitions d'index et d'indexeur de cet exemple. Le code qui convient se trouve dans deux fichiers :

- **Hotel.cs** contient le schéma qui définit l'index
- **Program.cs** contient des fonctions qui créent l'index, les sources de données et les indexeurs de la Recherche cognitive Azure, et chargent les résultats combinés dans l'index.

## Création d'un index

Cet exemple de programme utilise le SDK .NET pour définir et créer un index de Recherche cognitive Azure. Il tire parti de la classe [FieldBuilder](#) pour générer une structure d'index à partir d'une classe de modèle de données C#.

Le modèle de données est défini par la classe Hotel, qui contient également des références aux classes Address et Room. La classe FieldBuilder explore plusieurs définitions de classe pour générer une structure de données complexes pour l'index. Des étiquettes de métadonnées sont utilisées pour définir les attributs de chaque champ, par exemple s'il peut faire l'objet d'une recherche ou d'un tri.

Les extraits de code suivants tirés du fichier **Hotel.cs** montrent comment spécifier un champ unique et une référence à une autre classe de modèle de données.

```
    ...
[IsSearchable, IsFilterable, IsSortable]
public string HotelName { get; set; }
    ...
public Room[] Rooms { get; set; }
    ...
```

Dans le fichier **Program.cs**, l'index est défini avec un nom et une collection de champs générés par la méthode `FieldBuilder.BuildForType<Hotel>()`, puis créé comme suit :

```
private static async Task CreateIndex(string indexName, SearchServiceClient searchService)
{
    // Create a new search index structure that matches the properties of the Hotel class.
    // The Address and Room classes are referenced from the Hotel class. The FieldBuilder
    // will enumerate these to create a complex data structure for the index.
    var definition = new Index()
    {
        Name = indexName,
        Fields = FieldBuilder.BuildForType<Hotel>()
    };
    await searchService.Indexes.CreateAsync(definition);
}
```

## Créer l'indexeur et la source de données Azure Cosmos DB

Ensuite, le programme principal inclut une logique destinée à créer la source de données Azure Cosmos DB pour les données d'hôtels.

Tout d'abord, il concatène le nom de la base de données Azure Cosmos DB à la chaîne de connexion. Il définit

ensuite l'objet de source de données, y compris les paramètres propres aux sources Azure Cosmos DB, tels que la propriété [useChangeDetection].

```
private static async Task CreateAndRunCosmosDbIndexer(string indexName, SearchServiceClient searchService)
{
    // Append the database name to the connection string
    string cosmosConnectionString =
        configuration["CosmosDBConnectionString"]
        + ";Database="
        + configuration["CosmosDBDatabaseName"];

    DataSource cosmosDbDataSource = DataSource.CosmosDb(
        name: configuration["CosmosDBDatabaseName"],
        cosmosDbConnectionString: cosmosConnectionString,
        collectionName: "hotels",
        useChangeDetection: true);

    // The Azure Cosmos DB data source does not need to be deleted if it already exists,
    // but the connection string might need to be updated if it has changed.
    await searchService.DataSources.CreateOrUpdateAsync(cosmosDbDataSource);
```

Une fois la source de données créée, le programme configure un indexeur Azure Cosmos DB nommé **hotel-rooms-cosmos-indexer**.

```
Indexer cosmosDbIndexer = new Indexer(
    name: "hotel-rooms-cosmos-indexer",
    dataSourceName: cosmosDbDataSource.Name,
    targetIndexName: indexName,
    schedule: new IndexingSchedule(TimeSpan.FromDays(1)));

// Indexers keep metadata about how much they have already indexed.
// If we already ran this sample, the indexer will remember that it already
// indexed the sample data and not run again.
// To avoid this, reset the indexer if it exists.
bool exists = await searchService.Indexers.ExistsAsync(cosmosDbIndexer.Name);
if (exists)
{
    await searchService.Indexers.ResetAsync(cosmosDbIndexer.Name);
}
await searchService.Indexers.CreateOrUpdateAsync(cosmosDbIndexer);
```

Le programme supprime les indexeurs portant le même nom avant de créer le nouveau, au cas où vous souhaiteriez exécuter cet exemple plusieurs fois.

Cet exemple définit une planification pour l'indexeur, afin qu'il s'exécute une fois par jour. Si vous ne souhaitez pas que l'indexeur soit automatiquement réexécuté à l'avenir, vous pouvez supprimer la propriété de planification de cet appel.

## Indexer les données Azure Cosmos DB

Une fois la source de données et l'indexeur créés, le code qui exécute l'indexeur est simple :

```
try
{
    await searchService.Indexers.RunAsync(cosmosDbIndexer.Name);
}
catch (CloudException e) when (e.Response.StatusCode == (HttpStatusCode)429)
{
    Console.WriteLine("Failed to run indexer: {0}", e.Response.Content);
}
```

Cet exemple inclut un bloc try-catch simple pour signaler les erreurs susceptibles de se produire pendant

l'exécution.

Une fois l'indexeur Azure Cosmos DB exécuté, l'index de recherche contient un ensemble complet d'exemples de documents d'hôtel. Toutefois, le champ des chambres pour chaque hôtel est un tableau vide, car la source de données Azure Cosmos DB ne contenait aucun détail sur les chambres. Ensuite, le programme extrait les données des chambres du stockage d'objets blob et les fusionne.

### Créer l'indexeur et la source de données du stockage blob

Pour obtenir les détails des chambres, le programme configure tout d'abord une source de données de stockage blob pour référencer un ensemble de fichiers d'objets blob JSON individuels.

```
private static async Task CreateAndRunBlobIndexer(string indexName, SearchServiceClient searchService)
{
    DataSource blobDataSource = DataSource.AzureBlobStorage(
        name: configuration["BlobStorageAccountName"],
        storageConnectionString: configuration["BlobStorageConnectionString"],
        containerName: "hotel-rooms");

    // The blob data source does not need to be deleted if it already exists,
    // but the connection string might need to be updated if it has changed.
    await searchService.DataSources.CreateOrUpdateAsync(blobDataSource);
```

Une fois la source de données créée, le programme configure un indexeur d'objets blob nommé **hotel-rooms-blob-indexer**.

```
// Add a field mapping to match the Id field in the documents to
// the HotelId key field in the index
List<FieldMapping> map = new List<FieldMapping> {
    new FieldMapping("Id", "HotelId")
};

Indexer blobIndexer = new Indexer(
    name: "hotel-rooms-blob-indexer",
    dataSourceName: blobDataSource.Name,
    targetIndexName: indexName,
    fieldMappings: map,
    parameters: new IndexingParameters().ParseJson(),
    schedule: new IndexingSchedule(TimeSpan.FromDays(1)));

// Reset the indexer if it already exists
bool exists = await searchService.Indexers.ExistsAsync(blobIndexer.Name);
if (exists)
{
    await searchService.Indexers.ResetAsync(blobIndexer.Name);
}
await searchService.Indexers.CreateOrUpdateAsync(blobIndexer);
```

Les objets blob JSON contiennent un champ de clé nommé `Id` au lieu de `HotelId`. Le code utilise la classe `FieldMapping` pour indiquer à l'indexeur de rediriger la valeur du champ `Id` vers la clé de document `HotelId` dans l'index.

Les indexeurs de stockage d'objets blob peuvent utiliser des paramètres qui identifient le mode d'analyse à utiliser. Le mode d'analyse diffère selon que les objets blob représentent un document unique ou plusieurs documents au sein du même objet blob. Dans cet exemple, comme chaque objet blob représente un document d'index unique, le code utilise le paramètre `IndexingParameters.ParseJson()`.

Pour plus d'informations sur les paramètres d'analyse d'indexeur pour les objets blob JSON, consultez [Indexer des objets blob JSON](#). Pour plus d'informations sur la spécification de ces paramètres à l'aide du SDK .NET, consultez la classe [IndexerParametersExtension](#).

Le programme supprime les indexeurs portant le même nom avant de créer le nouveau, au cas où vous souhaiteriez exécuter cet exemple plusieurs fois.

Cet exemple définit une planification pour l'indexeur, afin qu'il s'exécute une fois par jour. Si vous ne souhaitez pas que l'indexeur soit automatiquement réexécuté à l'avenir, vous pouvez supprimer la propriété de planification de cet appel.

### Indexer des données blob

Une fois l'indexeur et la source de données de stockage d'objets blob créés, le code qui exécute l'indexeur est simple :

```
try
{
    await searchService.Indexers.RunAsync(cosmosDbIndexer.Name);
}
catch (CloudException e) when (e.Response.StatusCode == (HttpStatusCode)429)
{
    Console.WriteLine("Failed to run indexer: {0}", e.Response.Content);
}
```

Étant donné que l'index a déjà été rempli avec les données d'hôtels à partir de la base de données Azure Cosmos DB, l'indexeur d'objets blob met à jour les documents existants dans l'index et ajoute les détails des chambres.

#### NOTE

Si vous avez les mêmes champs non-clé dans vos deux sources de données et que les données au sein de ces champs ne correspondent pas, l'index contient les valeurs de l'indexeur qui a été exécuté le plus récemment. Dans notre exemple, les deux sources de données contiennent un champ **HotelName**. Si, pour une raison quelconque, les données de ce champ sont différentes pour des documents ayant la même valeur de clé, les données **HotelName** de la source de données qui a été indexée le plus récemment sont la valeur stockée dans l'index.

## 5 - Recherche

Une fois le programme exécuté, vous pouvez explorer l'index de recherche rempli en utilisant l'[Explorateur de recherche](#) dans le portail.

Dans le portail Azure, ouvrez la page **Vue d'ensemble** du service de recherche, puis recherchez l'index **hotel-rooms-sample** dans la liste **Index**.

Usage	Monitoring	Indexes	Indexers	Data sources	Skillsets
NAME			DOCUMENT COUNT		STORAGE SIZE
hotel-rooms-sample			7		114.83 kB

Cliquez sur l'index **hotel-rooms-sample** dans la liste. Vous verrez une interface de l'Explorateur de recherche pour l'index. Entrez une requête portant sur un terme, par exemple « Luxury » (Luxe). Vous devriez voir au moins un document dans les résultats, et ce document doit afficher une liste d'objets de chambre dans son tableau de chambres.

## Réinitialiser et réexécuter

Dans les premières étapes expérimentales du développement, l'approche la plus pratique pour les itérations de conception consiste à supprimer les objets de Recherche cognitive Azure et à autoriser votre code à les

reconstruire. Les noms des ressources sont uniques. La suppression d'un objet vous permet de le recréer en utilisant le même nom.

L'exemple de code pour ce tutoriel recherche les objets existants et les supprime pour vous permettre de réexécuter votre code.

Vous pouvez également utiliser le portail pour supprimer les index, les indexeurs et les sources de données.

## Nettoyer les ressources

Lorsque vous travaillez dans votre propre abonnement, il est judicieux à la fin d'un projet de supprimer les ressources dont vous n'avez plus besoin. Les ressources laissées en cours d'exécution peuvent vous coûter de l'argent. Vous pouvez supprimer les ressources une par une, ou choisir de supprimer le groupe de ressources afin de supprimer l'ensemble des ressources.

Vous pouvez rechercher et gérer les ressources dans le portail à l'aide des liens Toutes les ressources ou Groupes de ressources situés dans le volet de navigation de gauche.

## Étapes suivantes

Maintenant que vous êtes familiarisé avec le concept d'investigation de données provenant de plusieurs sources, examinons de plus près la configuration de l'indexeur, en commençant par Cosmos DB.

[Configurer un indexeur Azure Cosmos DB](#)

# Tutoriel : Optimiser l'indexation avec l'API Push

04/10/2020 • 24 minutes to read • [Edit Online](#)

Recherche cognitive Azure prend en charge [deux approches de base](#) pour importer des données dans un index de recherche : *envoyer (push)* les données dans l'index par programme ou pointer un [indexeur Recherche cognitive Azure](#) vers une source de données prise en charge pour *extraire (pull)* les données.

Ce tutoriel explique comment indexer efficacement des données à l'aide du [modèle Push](#) en utilisant le traitement par lot des demandes et en utilisant une stratégie de nouvelle tentative d'interruption exponentielle. Vous pouvez [Télécharger et exécuter l'application](#). Cet article explique les aspects clés de l'application et les facteurs à prendre en compte lors de l'indexation des données.

Ce tutoriel utilise C# et le [SDK .NET](#) pour effectuer les tâches suivantes :

- Création d'un index
- Tester différentes tailles de lot pour déterminer la taille la plus efficace
- Indexer les données de façon asynchrone
- Utiliser plusieurs threads pour augmenter les vitesses d'indexation
- Utiliser une stratégie de nouvelle tentative d'interruption exponentielle pour réessayer les éléments ayant échoué

Si vous n'avez pas d'abonnement Azure, créez un [compte gratuit](#) avant de commencer.

## Prérequis

Les services et outils suivants sont indispensables dans ce tutoriel.

- [Visual Studio](#), toute édition. L'exemple de code et les instructions ont été testés dans l'édition Communauté gratuite.
- [Créez un service Recherche cognitive Azure](#) ou [recherchez un service existant](#) dans votre abonnement actuel.

## Télécharger les fichiers

Le code source pour ce tutoriel se trouve dans le dossier [optimzize-data-indexing](#) du dépôt GitHub [Azure-Samples/azure-search-dotnet-samples](#).

## Considérations relatives aux clés

Lors de l'envoi (push) de données dans un index, plusieurs considérations importantes affectent les vitesses d'indexation. Pour plus d'informations sur ces facteurs, consultez l'article [Indexer de grands jeux de données](#).

Les six facteurs clés à prendre en compte sont les suivants :

- **Niveau de service et nombre de partitions/réplicas** : L'ajout de partitions et l'augmentation de votre niveau augmentent les vitesses d'indexation.
- **Schéma d'index** : L'ajout de champs et l'ajout de propriétés supplémentaires à des champs (comme `searchable`, `facetable` ou `filterable`) réduisent les vitesses d'indexation.
- **Taille de lot** : La taille de lot optimale varie en fonction du schéma et du jeu de données de votre index.
- **Nombre de threads/workers** : Un seul thread ne tirera pas pleinement parti des vitesses d'indexation

- **Stratégie de nouvelle tentative** : Une stratégie de nouvelle tentative d'interruption exponentielle doit être utilisée pour optimiser l'indexation.
- **Vitesse de transfert des données du réseau** : La vitesse de transfert des données peut être un facteur limitatif. Indexez les données à partir de votre environnement Azure pour augmenter la vitesse de transfert des données.

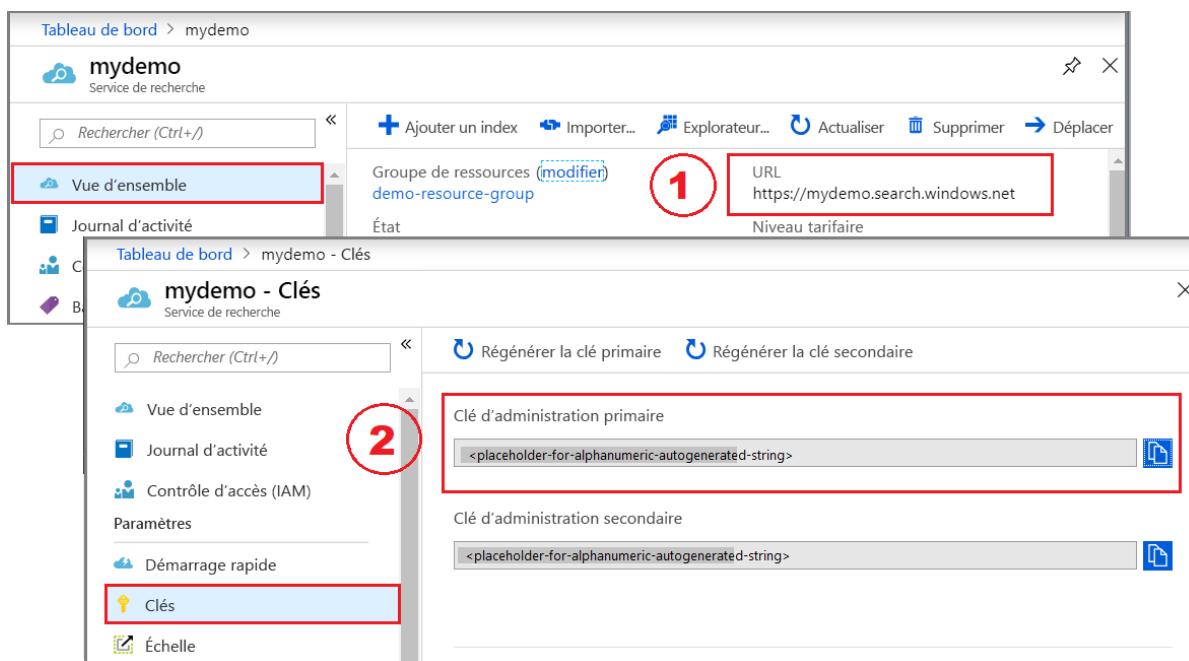
## 1 - Créer un service Recherche cognitive Azure

Pour suivre ce didacticiel, vous avez besoin d'un service Recherche cognitive Azure, que vous pouvez [créer dans le portail](#). Nous vous recommandons d'utiliser le même niveau que celui que vous prévoyez d'utiliser en production pour pouvoir tester et optimiser avec précision les vitesses d'indexation.

### Obtenir une clé API d'administration et une URL pour Recherche cognitive Azure

Les appels d'API nécessitent l'URL du service et une clé d'accès. Un service de recherche est créé avec les deux. Ainsi, si vous avez ajouté la Recherche cognitive Azure à votre abonnement, effectuez ce qui suit pour obtenir les informations nécessaires :

1. [Connectez-vous au portail Azure](#), puis dans la page **Vue d'ensemble** du service de recherche, récupérez l'URL. Voici un exemple de point de terminaison : <https://mydemo.search.windows.net>.
2. Dans **Paramètres > Clés**, obtenez une clé d'administration pour avoir des droits d'accès complets sur le service. Il existe deux clés d'administration interchangeables, fournies pour assurer la continuité de l'activité au cas où vous deviez en remplacer une. Vous pouvez utiliser la clé primaire ou secondaire sur les demandes d'ajout, de modification et de suppression d'objets.



## 2 - Configurer votre environnement

1. Démarrez Visual Studio et ouvrez **OptimizeDataIndexing.sln**.
2. Dans l'Explorateur de solutions, ouvrez **appsettings.json** pour fournir les informations de connexion.
3. Pour `searchServiceName`, si l'URL complète est « <https://my-demo-service.search.windows.net> », le nom de service à spécifier est « my-demo-service ».

```
{  
  "SearchServiceName": "<YOUR-SEARCH-SERVICE-NAME>",  
  "SearchServiceAdminApiKey": "<YOUR-ADMIN-API-KEY>",  
  "SearchIndexName": "optimize-indexing"  
}
```

## 3 - Explorer le code

Une fois que vous avez mis à jour `appSettings.json`, l'exemple de programme dans `OptimizeDataIndexing.sln` doit être prêt pour la génération et l'exécution.

Ce code est dérivé du [Démarrage rapide de C#](#). Vous trouverez des informations plus détaillées sur les principes de base de l'utilisation du kit de développement logiciel (SDK) .NET dans cet article.

Cette application console C#/NET simple effectue les tâches suivantes :

- Elle crée un index basé sur la structure de données de la classe Hotel C# (qui référence également la classe Address).
- Teste différentes tailles de lot pour déterminer la taille la plus efficace
- Indexe les données de façon asynchrone
  - Utilisation de plusieurs threads pour augmenter les vitesses d'indexation
  - Utilisation d'une stratégie de nouvelle tentative d'interruption exponentielle pour réessayer les éléments ayant échoué

Avant d'exécuter le programme, prenez une minute pour étudier le code et les définitions d'index de cet exemple. Le code qui convient se trouve dans plusieurs fichiers :

- **Hotel.cs** et **Address.cs** contiennent le schéma qui définit l'index
- **DataGenerator.cs** contient une classe simple pour faciliter la création de grandes quantités de données d'hôtel
- **ExponentialBackoff.cs** contient du code pour optimiser le processus d'indexation, comme décrit ci-dessous
- **Program.cs** contient des fonctions qui créent et suppriment l'index Recherche cognitive Azure, indexent des lots de données et testent différentes tailles de lot

### Création de l'index

Cet exemple de programme utilise le SDK .NET pour définir et créer un index de Recherche cognitive Azure. Il tire parti de la classe `FieldBuilder` pour générer une structure d'index à partir d'une classe de modèle de données C#.

Le modèle de données est défini par la classe Hotel, qui contient également des références à la classe Address. La classe FieldBuilder explore plusieurs définitions de classe pour générer une structure de données complexes pour l'index. Des étiquettes de métadonnées sont utilisées pour définir les attributs de chaque champ, par exemple s'il peut faire l'objet d'une recherche ou d'un tri.

Les extraits de code suivants tirés du fichier `Hotel.cs` montrent comment spécifier un champ unique et une référence à une autre classe de modèle de données.

```
    . . .  
    [IsSearchable, IsSortable]  
    public string HotelName { get; set; }  
    . . .  
    public Address Address { get; set; }  
    . . .
```

Dans le fichier `Program.cs`, l'index est défini avec un nom et une collection de champs générés par la méthode `FieldBuilder.BuildForType<Hotel>()`, puis créé comme suit :

```

private static async Task CreateIndex(string indexName, SearchServiceClient searchService)
{
    // Create a new search index structure that matches the properties of the Hotel class.
    // The Address class is referenced from the Hotel class. The FieldBuilder
    // will enumerate these to create a complex data structure for the index.
    var definition = new Index()
    {
        Name = indexName,
        Fields = FieldBuilder.BuildForType<Hotel>()
    };
    await searchService.Indexes.CreateAsync(definition);
}

```

## Génération des données

Une classe simple est implémentée dans le fichier **DataGenerator.cs** pour générer des données à des fins de test. Le seul but de cette classe est de faciliter la génération d'un grand nombre de documents avec un ID unique pour l'indexation.

Pour obtenir une liste de 100 000 hôtels avec des ID uniques, vous devez exécuter les deux lignes de code suivantes :

```

DataGenerator dg = new DataGenerator();
List<Hotel> hotels = dg.GetHotels(100000, "large");

```

Deux tailles d'hôtels sont disponibles pour le test dans cet exemple : **small** et **large**.

Le schéma de votre index peut avoir un impact significatif sur les vitesses d'indexation. En raison de cet impact, il est logique de convertir cette classe afin de générer des données correspondant à votre schéma d'index prévu après l'exécution de ce didacticiel.

## 4 - Tester les tailles de lot

Recherche cognitive Azure prend en charge les API suivantes pour charger un ou plusieurs documents dans un index :

- [Ajout, mise à jour ou suppression de documents \(API REST\)](#)
- [classe indexAction ou classe indexBatch](#)

L'indexation de documents par lots améliore considérablement les performances d'indexation. Ces lots peuvent comporter jusqu'à 1000 documents, ou jusqu'à 16 Mo par lot.

La détermination de la taille de lot optimale pour vos données est un composant clé de l'optimisation des vitesses d'indexation. Les deux principaux facteurs qui influencent la taille de lot optimale sont les suivants :

- Le schéma de votre index
- La taille de vos données

Étant donné que la taille de lot optimale dépend de votre index et de vos données, la meilleure approche consiste à tester différentes tailles de lot pour déterminer ce qui produit les vitesses d'indexation les plus rapides pour votre scénario.

La fonction suivante illustre une approche simple pour tester les tailles de lot.

```

public static async Task TestBatchSizes(IPageIndexClient indexClient, int min = 100, int max = 1000, int
step = 100, int numTries = 3)
{
    DataGenerator dg = new DataGenerator();

    Console.WriteLine("Batch Size \t Size in MB \t MB / Doc \t Time (ms) \t MB / Second");
    for (int numDocs = min; numDocs <= max; numDocs += step)
    {
        List<TimeSpan> durations = new List<TimeSpan>();
        double sizeInMb = 0.0;
        for (int x = 0; x < numTries; x++)
        {
            List<Hotel> hotels = dg.GetHotels(numDocs, "large");

            DateTime startTime = DateTime.Now;
            await UploadDocuments(indexClient, hotels);
            DateTime endTime = DateTime.Now;
            durations.Add(endTime - startTime);

            sizeInMb = EstimateObjectSize(hotels);
        }

        var avgDuration = durations.Average(TimeSpan => TimeSpan.TotalMilliseconds);
        var avgDurationInSeconds = avgDuration / 1000;
        var mbPerSecond = sizeInMb / avgDurationInSeconds;

        Console.WriteLine("{0} \t{1} \t{2} \t{3} \t{4}", numDocs, Math.Round(sizeInMb, 3),
Math.Round(sizeInMb / numDocs, 3), Math.Round(avgDuration, 3), Math.Round(mbPerSecond, 3));

        // Pausing 2 seconds to let the search service catch its breath
        Thread.Sleep(2000);
    }
}

```

Étant donné que tous les documents ne sont pas de la même taille (même si c'est le cas dans cet exemple), nous estimons la taille des données que nous envoyons au service de recherche. Pour cela, nous allons utiliser la fonction ci-dessous, qui convertit d'abord l'objet en JSON, puis détermine sa taille en octets. Cette technique nous permet de déterminer les tailles de lot les plus efficaces en termes de vitesses d'indexation en Mo/s.

```

public static double EstimateObjectSize(object data)
{
    // converting data to json for more accurate sizing
    var json = JsonConvert.SerializeObject(data);

    // converting object to byte[] to determine the size of the data
    BinaryFormatter bf = new BinaryFormatter();
    MemoryStream ms = new MemoryStream();
    byte[] Array;

    bf.Serialize(ms, json);
    Array = ms.ToArray();

    // converting from bytes to megabytes
    double sizeInMb = (double)Array.Length / 1000000;

    return sizeInMb;
}

```

La fonction requiert un `IPageIndexClient`, ainsi que le nombre de tentatives que vous souhaitez tester pour chaque taille de lot. Comme il peut y avoir une certaine variabilité dans les temps d'indexation pour chaque lot, nous essayons chaque lot trois fois par défaut afin que les résultats soient plus significatifs.

```
await TestBatchSizes(indexClient, numTries: 3);
```

Lorsque vous exécutez la fonction, vous devez voir une sortie semblable à celle-ci dans votre console :

```
Deleting index...
Creating index...
Finding optimal batch size...
Batch Size      Size in MB      MB / Doc      Time (ms)      MB / Second
100            0.29             0.003          241.517        1.202
200            0.58             0.003          279.182        2.079
300            0.871            0.003          434.859        2.003
400            1.161            0.003          506.927        2.291
500            1.452            0.003          593.79         2.445
600            1.742            0.003          752.854        2.314
700            2.032            0.003          862.523        2.356
800            2.323            0.003          929.534        2.499
900            2.614            0.003          1082.359       2.415
1000           2.904            0.003          1255.456       2.313
Complete. Press any key to end application...
```

Identifiez la taille de lot la plus efficace, puis utilisez cette taille de lot à l'étape suivante du didacticiel. Vous pouvez voir un plateau de Mo/s sur différentes tailles de lot.

## 5 - Indexer des données

Maintenant que nous avons identifié la taille de lot que nous envisageons d'utiliser, l'étape suivante consiste à commencer à indexer les données. Pour indexer les données efficacement, cet exemple :

- utilise plusieurs threads/workers.
- Implémente une stratégie de nouvelle tentative d'interruption exponentielle.

### Utiliser plusieurs threads/workers

Pour tirer pleinement parti des vitesses d'indexation de Recherche cognitive Azure, vous devrez probablement utiliser plusieurs threads pour envoyer simultanément des demandes d'indexation par lot au service.

Plusieurs des considérations clés mentionnées ci-dessus ont un impact sur le nombre optimal de threads. Vous pouvez modifier cet exemple et effectuer des tests avec différents nombres de threads pour déterminer le nombre de threads optimal pour votre scénario. Toutefois, tant que plusieurs threads s'exécutent simultanément, vous devriez être en mesure de bénéficier de la majeure partie des gains d'efficacité.

Au fur et à mesure que les requêtes atteignent le service de recherche, vous pouvez rencontrer des [codes d'état HTTP](#) indiquant que la demande n'a pas abouti. Pendant l'indexation, deux codes d'état HTTP courants sont :

- **503 Service indisponible** : Cette erreur signifie que le système est surchargé et que votre requête ne peut pas être traitée pour le moment.
- **207 Multi-état** : Cette erreur signifie que certains documents ont réussi, mais qu'au moins un a échoué.

### Implémenter une stratégie de nouvelle tentative d'interruption exponentielle

En cas d'échec, les requêtes doivent être retentées à l'aide d'une [stratégie de nouvelle tentative d'interruption exponentielle](#).

Le kit de développement logiciel (SDK) .NET de Recherche cognitive Azure retente automatiquement lors des erreurs 503 et autres requêtes ayant échoué, mais vous devez implémenter votre propre logique pour réessayer en cas de code 207. Des outils open source tels que [Polly](#) peuvent également être utilisés pour mettre en œuvre

une stratégie de nouvelle tentative.

Dans cet exemple, nous implémentons notre propre stratégie de nouvelle tentative d'interruption exponentielle. Pour implémenter cette stratégie, nous commençons par définir certaines variables, dont `maxRetryAttempts` et la valeur `delay` initiale pour une requête ayant échoué :

```
// Create batch of documents for indexing
IndexBatch<Hotel> batch = IndexBatch.Upload(hotels);

// Define parameters for exponential backoff
int attempts = 0;
TimeSpan delay = delay = TimeSpan.FromSeconds(2);
int maxRetryAttempts = 5;
```

Il est important d'intercepter `IndexBatchException`, car ces exceptions indiquent que l'opération d'indexation a été partiellement réussie (207). Les éléments ayant échoué doivent être retentés à l'aide de la méthode `FindFailedActionsToRetry` qui permet de créer facilement un nouveau lot contenant uniquement les éléments ayant échoué.

Les exceptions autres que `IndexBatchException` doivent également être interceptées et indiquer que la requête a échoué complètement. Ces exceptions sont moins courantes, en particulier avec le kit de développement logiciel (SDK) .NET, car ce dernier retente automatiquement en cas d'erreur 503.

```
// Implement exponential backoff
do
{
    try
    {
        attempts++;
        var response = await indexClient.Documents.IndexAsync(batch);
        break;
    }
    catch (IndexBatchException ex)
    {
        Console.WriteLine("[Attempt: {0} of {1} Failed] - Error: {2}", attempts, maxRetryAttempts,
ex.Message);

        if (attempts == maxRetryAttempts)
            break;

        // Find the failed items and create a new batch to retry
        batch = ex.FindFailedActionsToRetry(batch, x => x.HotelId);
        Console.WriteLine("Retrying failed documents using exponential backoff...\n");

        Task.Delay(delay).Wait();
        delay = delay * 2;
    }
    catch (Exception ex)
    {
        Console.WriteLine("[Attempt: {0} of {1} Failed] - Error: {2} \n", attempts, maxRetryAttempts,
ex.Message);

        if (attempts == maxRetryAttempts)
            break;

        Task.Delay(delay).Wait();
        delay = delay * 2;
    }
} while (true);
```

À partir de là, nous encapsulons le code d'interruption exponentielle dans une fonction afin qu'il puisse être facilement appelé.

Une autre fonction est ensuite créée pour gérer les threads actifs. Par souci de simplicité, cette fonction n'est pas incluse ici, mais se trouve dans [ExponentialBackoff.cs](#). La fonction peut être appelée à l'aide de la commande suivante, où `hotels` correspond aux données que nous voulons télécharger, `1000` à la taille du lot et `8` au nombre de threads simultanés :

```
ExponentialBackoff.IndexData(indexClient, hotels, 1000, 8).Wait();
```

Lorsque vous exécutez la fonction, vous devez voir une sortie semblable à celle ci-dessous :

```
Deleting index...
Creating index...
Uploading using exponential backoff...
Uploading 100000 documents...
Creating 8 threads...
Sending a batch of 1000 docs starting with doc 0...
Sending a batch of 1000 docs starting with doc 1000...
Sending a batch of 1000 docs starting with doc 2000...
Sending a batch of 1000 docs starting with doc 3000...
Sending a batch of 1000 docs starting with doc 4000...
Sending a batch of 1000 docs starting with doc 5000...
Sending a batch of 1000 docs starting with doc 6000...
Sending a batch of 1000 docs starting with doc 7000...
Finished a thread, kicking off another...
Sending a batch of 1000 docs starting with doc 8000...
```

En cas d'échec d'un lot de documents, une erreur s'affiche pour indiquer qu'un échec s'est produit et que le lot est retenté :

```
BATCH STARTING AT DOC 37000:
[Attempt: 1 of 5 Failed] - Error: 71 of 1000 indexing actions in the batch failed. The remaining actions succeeded and modified the index. Check the IndexResponse property for the status of each index action.
[Status Code 503] - 71 documents
[Status Code 201] - 929 documents
Retrying failed documents using exponential backoff...
```

Une fois l'exécution de la fonction terminée, vous pouvez vérifier que tous les documents ont été ajoutés à l'index.

## 6 - Explorer l'index

Une fois le programme exécuté, vous pouvez explorer l'index de recherche rempli, par programme ou en utilisant l'[Explorateur de recherche](#) dans le portail.

### Par programme

Il existe deux options principales pour vérifier le nombre de documents dans un index : l'[API Nombre de documents](#) et l'[API Obtention de statistiques d'index](#). Les deux peuvent nécessiter du temps supplémentaire pour la mise à jour. Par conséquent, ne vous inquiétez pas si le nombre de documents renvoyés est inférieur à celui

prévu initialement.

#### Nombre de documents

L'opération Count Documents récupère le nombre de documents dans un index de recherche :

```
long indexDocCount = indexClient.Documents.Count();
```

#### Obtention de statistiques d'index

L'opération Get Index Statistics retourne un nombre de documents pour l'index actuel, ainsi que l'utilisation du stockage. La mise à jour des statistiques d'index prendra plus de temps que celle du nombre de documents.

```
IndexGetStatisticsResult indexStats = serviceClient.Indexes.GetStatistics(configuration["SearchIndexName"]);
```

#### Portail Azure

Dans le portail Azure, ouvrez la page **Vue d'ensemble** du service de recherche, puis recherchez l'index **optimize-indexing** dans la liste **Index**.

Usage	Monitoring	Indexes	Indexers	Data sources	Skillsets
Name ↑↓				Document Count ↑↓	Storage Size ↑↓
		optimize-indexing		100,000	12.41 MB

Le *Nombre de documents* et la *Taille de stockage* sont basés sur l'[API Obtention de statistiques d'index](#) et peuvent nécessiter plusieurs minutes pour se mettre à jour.

## Réinitialiser et réexécuter

Dans les premières étapes expérimentales du développement, l'approche la plus pratique pour les itérations de conception consiste à supprimer les objets de Recherche cognitive Azure et à autoriser votre code à les reconstruire. Les noms des ressources sont uniques. La suppression d'un objet vous permet de le recréer en utilisant le même nom.

L'exemple de code pour ce tutoriel recherche les index existants et les supprime pour vous permettre de réexécuter votre code.

Vous pouvez également utiliser le portail pour supprimer des index.

## Nettoyer les ressources

Lorsque vous travaillez dans votre propre abonnement, il est judicieux à la fin d'un projet de supprimer les ressources dont vous n'avez plus besoin. Les ressources laissées en cours d'exécution peuvent vous coûter de l'argent. Vous pouvez supprimer les ressources une par une, ou choisir de supprimer le groupe de ressources afin de supprimer l'ensemble des ressources.

Vous pouvez rechercher et gérer les ressources dans le portail à l'aide des liens **Toutes les ressources** ou **Groupes de ressources** situés dans le volet de navigation de gauche.

## Étapes suivantes

Maintenant que vous avez fait connaissance avec le concept d'ingestion efficace des données, examinons de plus près l'architecture des requêtes Lucene et le fonctionnement de la recherche en texte intégral dans Recherche cognitive Azure.

[Fonctionnement de la recherche en texte intégral dans la Recherche cognitive Azure](#)



# Tutoriel : Contenu recherchable généré par l'IA issu d'objets blob Azure avec le SDK .NET

04/10/2020 • 42 minutes to read • [Edit Online](#)

Si vous avez du texte non structuré ou des images dans Stockage Blob Azure, un [pipeline d'enrichissement par IA](#) peut extraire des informations et créer du contenu utile pour les scénarios de recherche en texte intégral ou d'exploration de connaissances. Dans ce tutoriel C#, appliquez la reconnaissance optique de caractères (OCR) sur des images et effectuez un traitement en langage naturel pour créer de nouveaux champs que vous pouvez exploiter dans les requêtes, les facettes et les filtres.

Ce tutoriel utilise C# et le [SDK .NET](#) pour effectuer les tâches suivantes :

- Commencez avec des fichiers d'application et des images dans Stockage Blob Azure.
- Définissez un pipeline pour ajouter la reconnaissance optique de caractères, l'extraction de texte, la détection de la langue, la reconnaissance d'expressions d'entité et de clé.
- Définissez un index pour stocker la sortie (contenu brut et paires nom-valeur générées par le pipeline).
- Exécutez le pipeline pour démarrer des transformations et une analyse, ainsi que pour créer et charger l'index.
- Explorez les résultats à l'aide de la recherche en texte intégral et d'une syntaxe de requête enrichie.

Si vous n'avez pas d'abonnement Azure, ouvrez un [compte gratuit](#) avant de commencer.

## Prérequis

- [Stockage Azure](#)
- [Visual Studio](#)
- [Créer ou rechercher un service de recherche existant](#)

### NOTE

Vous pouvez utiliser le service gratuit pour ce tutoriel. Avec un service de recherche gratuit, vous êtes limité à trois index, trois indexeurs et trois sources de données. Ce didacticiel crée une occurrence de chaque élément. Avant de commencer, veillez à disposer de l'espace suffisant sur votre service pour accepter les nouvelles ressources.

## Télécharger les fichiers

1. Ouvrez ce [dossier OneDrive](#) et en haut à gauche, cliquez sur **Télécharger** pour copier les fichiers sur votre ordinateur.
2. Cliquez avec le bouton droit sur le fichier zip et sélectionnez **Tout extraire**. Il y a 14 fichiers de différents types. Vous en utiliserez 7 dans le cadre de cet exercice.

Vous pouvez également télécharger le code source de ce tutoriel. Le code source se trouve dans le dossier tutorial-ai-enrichment du référentiel [azure-search-dotnet-samples](#).

## 1 - Créer les services

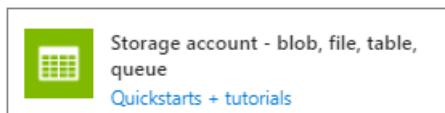
Ce tutoriel utilise Recherche cognitive Azure pour l'indexation et les requêtes, Cognitive Services pour l'enrichissement par IA sur le back-end, et Stockage Blob Azure pour fournir les données. Ce tutoriel reste sous l'allocation gratuite de 20 transactions par indexeur par jour sur Cognitive Services : les seuls services que vous

devez créer sont donc la recherche et le stockage.

Si possible, créez les deux services dans la même région et le même groupe de ressources pour des raisons de proximité et de facilité de gestion. Dans la pratique, votre compte de stockage Azure peut être dans une région quelconque.

## Démarrer avec le stockage Azure

1. [Connectez-vous au portail Azure](#) et cliquez sur **+ Créer une ressource**.
2. Recherchez un *compte de stockage* et sélectionnez l'offre de compte de stockage Microsoft.



3. Sous l'onglet **Bases**, les éléments suivants sont obligatoires. Acceptez les valeurs par défaut pour tout le reste.
  - **Groupe de ressources**. Sélectionnez un groupe existant ou créez-en un, mais utilisez le même groupe pour tous les services afin de pouvoir les gérer collectivement.
  - **Nom du compte de stockage**. Si vous pensez que vous pouvez avoir plusieurs ressources du même type, utilisez le nom pour lever l'ambiguïté par type et par région, par exemple *blobstoragewestus*.
  - **Emplacement**. Si possible, choisissez le même emplacement que celui utilisé pour Recherche cognitive Azure et Cognitive Services. Un emplacement unique annule les frais liés à la bande passante.
  - **Type de compte**. Choisissez la valeur par défaut, *StorageV2 (v2 universel)*.

4. Cliquez sur **Vérifier + créer** pour créer le service.
5. Une fois qu'il est créé, cliquez sur **Accéder à la ressource** pour ouvrir la page Vue d'ensemble.
6. Cliquez sur le service **Objets blob**.
7. Cliquez sur **+ Conteneur** pour créer un conteneur et nommez-le *cog-search-demo*.
8. Sélectionnez *cog-search-demo*, puis cliquez sur **Charger** pour ouvrir le dossier dans lequel vous avez enregistré les fichiers téléchargés. Sélectionnez les quatorze fichiers, puis cliquez sur **OK** pour charger.

NOM	DERNIÈRE MODIFICATION	TYPE D'OBJET BLOB	Type de contenu
10-K-FY16.html	2018-04-02T21:10:35.000Z	Bloquer un objet BLOB	text/html
5074.clip_image002_6FE27E85.png	2018-04-02T21:10:39.000Z	Bloquer un objet BLOB	image/png
Cognitive Services and Content Intelligence.pptx	2018-04-02T21:10:50.000Z	Bloquer un objet BLOB	application/vnd.openxmlformat
Cognitive Services and Bots (spanish).pdf	2018-04-02T21:10:41.000Z	Bloquer un objet BLOB	application/pdf
guthrie.jpg	2018-04-02T21:10:23.000Z	Bloquer un objet BLOB	image/jpeg
MSFT_cloud_architecture_contoso.pdf	2018-04-02T21:10:35.000Z	Bloquer un objet BLOB	application/pdf
MSFT_FY17_10K.docx	2018-04-02T21:10:36.000Z	Bloquer un objet BLOB	application/vnd.openxmlformat
NYSE_LNKD_2015.PDF	2018-04-02T21:10:35.000Z	Bloquer un objet BLOB	application/pdf
redshirt.jpg	2018-04-02T21:10:31.000Z	Bloquer un objet BLOB	image/jpeg
satyanadellalinux.jpg	2018-04-02T21:10:33.000Z	Bloquer un objet BLOB	image/jpeg
satyasletter.txt	2018-04-02T21:10:22.000Z	Bloquer un objet BLOB	texte/brut

9. Avant de quitter Stockage Azure, obtenez une chaîne de connexion afin de pouvoir formuler une connexion dans Recherche cognitive Azure.
  - a. Revenez à la page Vue d'ensemble de votre compte de stockage (nous avons utilisé *blobstoragewestus* comme exemple).

- b. Dans le volet de navigation gauche, sélectionnez **Clés d'accès** et copiez l'une des chaînes de connexion.

La chaîne de connexion est une URL similaire à l'exemple suivant :

```
DefaultEndpointsProtocol=https;AccountName=blobstoragewestus;AccountKey=<your account key>;EndpointSuffix=core.windows.net
```

10. Enregistrez la chaîne de connexion dans le Bloc-Notes. Vous en aurez besoin plus tard lors de la configuration de la connexion à la source de données.

## Cognitive Services

L'enrichissement par IA s'appuie sur Cognitive Services, notamment Analyse de texte et Vision par ordinateur pour le traitement des images et du langage naturel. Si votre objectif était de réaliser un prototype ou un projet réel, vous devriez à ce stade provisionner Cognitive Services (dans la même région que Recherche cognitive Azure) afin de pouvoir l'attacher aux opérations d'indexation.

Cependant, dans le cadre de cet exercice, vous pouvez ignorer le provisionnement des ressources, car Recherche cognitive Azure peut se connecter à Cognitive Services en arrière-plan et vous fournir 20 transactions gratuites par exécution de l'indexeur. Comme ce tutoriel utilise 14 transactions, l'allocation gratuite est suffisante. Pour les projets de plus grande envergure, prévoyez de provisionner Cognitive Services au niveau S0 du paiement à l'utilisation.

Pour plus d'informations, consultez [Attacher Cognitive Services](#).

## Recherche cognitive Azure

Le troisième composant est Recherche cognitive Azure, que vous pouvez [créer dans le portail](#). Vous pouvez utiliser le niveau gratuit pour effectuer cette procédure pas à pas.

### Obtenir une clé API d'administration et une URL pour Recherche cognitive Azure

Pour interagir avec votre service Recherche cognitive Azure, vous devrez disposer de l'URL du service et d'une clé d'accès. Un service de recherche est créé avec les deux. Ainsi, si vous avez ajouté la Recherche cognitive Azure à votre abonnement, effectuez ce qui suit pour obtenir les informations nécessaires :

1. [Connectez-vous au portail Azure](#), puis dans la page **Vue d'ensemble** du service de recherche, récupérez l'URL. Voici un exemple de point de terminaison : `https://mydemo.search.windows.net`.
2. Dans **Paramètres > Clés**, obtenez une clé d'administration pour avoir des droits d'accès complets sur le service. Il existe deux clés d'administration interchangeables, fournies pour assurer la continuité de l'activité au cas où vous deviez en remplacer une. Vous pouvez utiliser la clé primaire ou secondaire sur les demandes d'ajout, de modification et de suppression d'objets.

Obtenez aussi la clé de requête. Il est recommandé d'émettre des demandes de requête avec un accès en lecture seule.

L'utilisation d'une clé valide permet d'établir, en fonction de chaque demande, une relation de confiance entre l'application qui envoie la demande et le service qui en assure le traitement.

## 2 - Configurer votre environnement

Commencez par ouvrir Visual Studio et créer un nouveau projet d'application console pouvant s'exécuter sur .NET Core.

### Installer les packages NuGet

Le [SDK .NET de la Recherche cognitive Azure](#) se compose de quelques bibliothèques clientes qui vous permettent de gérer vos index, sources de données, indexeurs et compétences, ainsi que de charger et gérer des documents et d'exécuter des requêtes, sans avoir à gérer les détails de HTTP et de JSON. Ces bibliothèques clientes sont distribuées sous la forme de packages NuGet.

Pour ce projet, installez la version 9 ou ultérieure du package NuGet [Microsoft.Azure.Search](#).

1. Dans un navigateur, accédez à la [page Package NuGet Microsoft.Azure.Search](#).
2. Sélectionnez la version la plus récente (9 ou ultérieure).
3. Copiez la commande du gestionnaire de package.
4. Ouvrez la console du gestionnaire de package. Cliquez sur **Outils > Gestionnaire de package NuGet > Console du Gestionnaire de package**.
5. Collez et exécutez la commande que vous avez copiée à l'étape précédente.

Installez ensuite la version la plus récente du package NuGet [Microsoft.Extensions.Configuration.Json](#).

1. Sélectionnez **Outils > Gestionnaire de package NuGet > Gérer les packages NuGet pour la solution...**.
2. Cliquez sur **Parcourir**, puis recherchez le package NuGet [Microsoft.Extensions.Configuration.Json](#).
3. Sélectionnez le package, sélectionnez votre projet, vérifiez que la version correspond à la dernière version stable, puis cliquez sur **Installer**.

### Ajouter les informations de connexion au service

1. Cliquez avec le bouton droit sur votre projet dans l'Explorateur de solutions et sélectionnez **Ajouter > Nouvel élément...**.
2. Nommez le fichier `appsettings.json` et sélectionnez **Ajouter**.
3. Incluez ce fichier dans votre répertoire de sortie.
  - a. Cliquez avec le bouton droit sur `appsettings.json`, puis sélectionnez **Propriétés**.
  - b. Remplacez la valeur **Copier dans le répertoire de sortie** par **Copie du plus récent**.
4. Copiez le code JSON ci-dessous dans votre nouveau fichier JSON.

```
{
  "SearchServiceName": "Put your search service name here",
  "SearchServiceAdminApiKey": "Put your primary or secondary API key here",
  "SearchServiceQueryApiKey": "Put your query API key here",
  "AzureBlobConnectionString": "Put your Azure Blob connection string here",
}
```

Ajoutez les informations relatives à votre service de recherche, ainsi qu'à votre compte de stockage d'objets blob. Rappelez-vous que vous pouvez récupérer ces informations à partir des étapes de configuration du service indiquées dans la section précédente.

Pour **SearchServiceName**, entrez le nom court du service et non l'URL complète.

### Ajouter des espaces de noms

Dans `Program.cs`, ajoutez les espaces de noms suivants.

```
using System;
using System.Collections.Generic;
using Microsoft.Azure.Search;
using Microsoft.Azure.Search.Models;
using Microsoft.Extensions.Configuration;

namespace EnrichwithAI
```

### Créer un client

Créez une instance de la classe `SearchServiceClient` sous `Main`.

```
public static void Main(string[] args)
{
    // Create service client
    IConfigurationBuilder builder = new ConfigurationBuilder().AddJsonFile("appsettings.json");
    IConfigurationRoot configuration = builder.Build();
    SearchServiceClient serviceClient = CreateSearchServiceClient(configuration);
```

`CreateSearchServiceClient` crée un `SearchServiceClient` à l'aide de valeurs stockées dans le fichier config de l'application (`appsettings.json`).

```
private static SearchServiceClient CreateSearchServiceClient(IConfigurationRoot configuration)
{
    string searchServiceName = configuration["SearchServiceName"];
    string adminApiKey = configuration["SearchServiceAdminApiKey"];

    SearchServiceClient serviceClient = new SearchServiceClient(searchServiceName, new
    SearchCredentials(adminApiKey));
    return serviceClient;
}
```

#### NOTE

La classe `SearchServiceClient` gère les connexions à votre service de recherche. Pour éviter l'ouverture d'un trop grand nombre de connexions, essayez de partager une seule instance de `SearchServiceClient` dans votre application, si possible. Ses méthodes sont thread-safe pour activer le partage.

## Ajouter une fonction pour quitter le programme pendant des événements d'échec

Ce tutoriel est destiné à vous aider à comprendre chaque étape du pipeline d'indexation. En cas de problème critique empêchant le programme de créer la source de données, l'ensemble de compétences, l'index ou l'indexeur, le programme génère le message d'erreur et se ferme afin que le problème puisse être compris et résolu.

Ajoutez `ExitProgram` à `Main` pour gérer les scénarios qui nécessitent la fermeture du programme.

```
private static void ExitProgram(string message)
{
    Console.WriteLine("{0}", message);
    Console.WriteLine("Press any key to exit the program...");
    Console.ReadKey();
    Environment.Exit(0);
}
```

## 3 - Créer le pipeline

Dans Recherche cognitive Azure, le traitement de l'IA se produit pendant l'indexation (ou l'ingestion de données). Cette partie de la procédure pas à pas crée quatre objets : source de données, définition d'index, ensemble de compétences, indexeur.

### Étape 1: Création d'une source de données

`SearchServiceClient` a une propriété `DataSources`. Cette propriété fournit toutes les méthodes dont vous avez besoin pour créer, afficher, mettre à jour ou supprimer des sources de données de la Recherche cognitive Azure.

Créer une nouvelle instance `DataSource` en appelant `serviceClient.DataSources.CreateOrUpdate(dataSource)`. `DataSource.AzureBlobStorage` implique la spécification du nom de la source de données, de la chaîne de connexion, ainsi que du nom du conteneur d'objets blob.

```

private static DataSource CreateOrUpdateDataSource(SearchServiceClient serviceClient, IConfigurationRoot configuration)
{
    DataSource dataSource = DataSource.AzureBlobStorage(
        name: "demodata",
        storageConnectionString: configuration["AzureBlobConnectionString"],
        containerName: "cog-search-demo",
        description: "Demo files to demonstrate cognitive search capabilities.");

    // The data source does not need to be deleted if it was already created
    // since we are using the CreateOrUpdate method
    try
    {
        serviceClient.DataSources.CreateOrUpdate(dataSource);
    }
    catch (Exception e)
    {
        Console.WriteLine("Failed to create or update the data source\n Exception message: {0}\n", e.Message);
        ExitProgram("Cannot continue without a data source");
    }

    return dataSource;
}

```

Pour une requête réussie, la méthode renvoie la source de données créée. En cas de problème lié à la requête, comme un paramètre non valide, la méthode lève une exception.

Ajoutez maintenant une ligne dans `Main` pour appeler la fonction `CreateOrUpdateDataSource` que vous venez d'ajouter.

```

public static void Main(string[] args)
{
    // Create service client
    IConfigurationBuilder builder = new ConfigurationBuilder().AddJsonFile("appsettings.json");
    IConfigurationRoot configuration = builder.Build();
    SearchServiceClient serviceClient = CreateSearchServiceClient(configuration);

    // Create or Update the data source
    Console.WriteLine("Creating or updating the data source...");
    DataSource dataSource = CreateOrUpdateDataSource(serviceClient, configuration);
}

```

Créez et exécutez la solution. Dans la mesure où il s'agit de votre première demande, vérifiez dans le portail Azure que la source de données a bien été créée dans la Recherche cognitive Azure. Dans la page de tableau de bord du service de recherche, vérifiez que la vignette Sources de données a un nouvel élément. Vous devrez peut-être attendre quelques minutes que la page du portail soit actualisée.

Type ↑↓	Name ↑↓	Table/Collection ↑↓
	demodata	cog-search-demo
	hotels-sample	hotels
	realestate-us-sample	Listings_5K_KingCounty_WA

## Étape 2 : Créer un ensemble de compétences

Dans cette section, vous définissez un ensemble d'étapes d'enrichissement que vous souhaitez appliquer à vos données. Chaque étape d'enrichissement est appelée *compétence* et l'ensemble des étapes d'enrichissement, *ensemble de compétences*. Ce tutoriel utilise des [compétences cognitives prédéfinies](#) pour l'ensemble de compétences :

- [Reconnaissance optique de caractères](#) pour reconnaître le texte imprimé et manuscrit dans des fichiers image.
- [Fusion de texte](#) pour consolider en un champ unique du texte issu d'une collection de champs.
- [Détection de la langue](#) pour identifier la langue du contenu.
- [Fractionnement de texte](#) pour découper un texte long en plus petits morceaux avant d'appeler les compétences d'extraction de phrases clés et de reconnaissance d'entités. Les compétences d'extraction de phrases clés et de reconnaissance d'entités acceptent les entrées de 50 000 caractères maximum. Certains fichiers d'exemple doivent être fractionnés pour satisfaire cette limite.
- [Reconnaissance d'entité](#) pour extraire les noms d'organisations du contenu dans le conteneur d'objets blob.
- [Extraction de phrases clés](#) pour extraire les principales expressions clés.

Lors du traitement initial, la Recherche cognitive Azure interprète chaque document pour lire le contenu de fichiers de différents formats. Le texte trouvé originaire du fichier source est placé dans un champ `content` généré, un pour chaque document. Par conséquent, définissez l'entrée en tant que `"/document/content"` de manière à utiliser ce texte.

Les sorties peuvent être mappées à un index, utilisées comme entrée d'une compétence en aval, ou les deux, comme c'est le cas avec le code de langue. Dans l'index, un code de langue est utile pour le filtrage. En tant qu'entrée, le code de langue est utilisé par les compétences d'analyse de texte pour informer les règles linguistiques en matière de césure de mots.

Pour plus d'informations sur les principes de base des ensembles de compétences, consultez [Guide pratique pour définir un ensemble de compétences](#).

### **Compétence de reconnaissance optique des caractères**

La compétence **OCR** extrait le texte des images. Cette compétence suppose l'existence d'un champ `normalized_images`. Pour générer ce champ, plus loin dans ce didacticiel, nous définirons la configuration `"imageAction"` dans la définition de l'indexeur sur `"generateNormalizedImages"`.

```
private static OcrSkill CreateOcrSkill()
{
    List<InputFieldMappingEntry> inputMappings = new List<InputFieldMappingEntry>
    {
        new InputFieldMappingEntry(
            name: "image",
            source: "/document/normalized_images/*")
    };

    List<OutputFieldMappingEntry> outputMappings = new List<OutputFieldMappingEntry>
    {
        new OutputFieldMappingEntry(
            name: "text",
            targetName: "text")
    };

    OcrSkill ocrSkill = new OcrSkill(
        description: "Extract text (plain and structured) from image",
        context: "/document/normalized_images/*",
        inputs: inputMappings,
        outputs: outputMappings,
        defaultLanguageCode: OcrSkillLanguage.En,
        shouldDetectOrientation: true);

    return ocrSkill;
}
```

## Compétence de fusion de texte

Dans cette section, vous allez créer une compétence **Fusion** qui fusionne le champ de contenu de document avec le texte généré par la compétence de reconnaissance optique des caractères.

```
private static MergeSkill CreateMergeSkill()
{
    List<InputFieldMappingEntry> inputMappings = new List<InputFieldMappingEntry>
    {
        new InputFieldMappingEntry(
            name: "text",
            source: "/document/content"),
        new InputFieldMappingEntry(
            name: "itemsToInsert",
            source: "/document/normalized_images/*/text"),
        new InputFieldMappingEntry(
            name: "offsets",
            source: "/document/normalized_images/*/contentOffset")
    };

    List<OutputFieldMappingEntry> outputMappings = new List<OutputFieldMappingEntry>
    {
        new OutputFieldMappingEntry(
            name: "mergedText",
            targetName: "merged_text")
    };

    MergeSkill mergeSkill = new MergeSkill(
        description: "Create merged_text which includes all the textual representation of each image inserted at the right location in the content field.",
        context: "/document",
        inputs: inputMappings,
        outputs: outputMappings,
        insertPreTag: " ",
        insertPostTag: " ");
}

return mergeSkill;
}
```

## Compétence Détection de langue

La compétence **Détection de langue** détecte la langue du texte d'entrée et renvoie un code de langue unique pour chaque document soumis dans la requête. Nous allons utiliser la sortie de la compétence **Détection de langue** en tant qu'entrée pour la compétence **Fractionnement de texte**.

```

private static LanguageDetectionSkill CreateLanguageDetectionSkill()
{
    List<InputFieldMappingEntry> inputMappings = new List<InputFieldMappingEntry>
    {
        new InputFieldMappingEntry(
            name: "text",
            source: "/document/merged_text")
    };

    List<OutputFieldMappingEntry> outputMappings = new List<OutputFieldMappingEntry>
    {
        new OutputFieldMappingEntry(
            name: "languageCode",
            targetName: "languageCode")
    };

    LanguageDetectionSkill languageDetectionSkill = new LanguageDetectionSkill(
        description: "Detect the language used in the document",
        context: "/document",
        inputs: inputMappings,
        outputs: outputMappings);

    return languageDetectionSkill;
}

```

## Compétence Fractionnement de texte

La compétence **Fractionnement** ci-dessous fractionnera le texte par pages et limitera la longueur de la page à 4 000 caractères comme indiqué par `String.Length`. L'algorithme essaiera de fractionner le texte en blocs dont la taille n'excède pas `maximumPageLength`. Dans ce cas, l'algorithme fera de son mieux pour arrêter la phrase sur une limite de phrase, de sorte que la taille du bloc peut être légèrement inférieure à `maximumPageLength`.

```

private static SplitSkill CreateSplitSkill()
{
    List<InputFieldMappingEntry> inputMappings = new List<InputFieldMappingEntry>
    {
        new InputFieldMappingEntry(
            name: "text",
            source: "/document/merged_text"),
        new InputFieldMappingEntry(
            name: "languageCode",
            source: "/document/languageCode")
    };

    List<OutputFieldMappingEntry> outputMappings = new List<OutputFieldMappingEntry>
    {
        new OutputFieldMappingEntry(
            name: "textItems",
            targetName: "pages")
    };

    SplitSkill splitSkill = new SplitSkill(
        description: "Split content into pages",
        context: "/document",
        inputs: inputMappings,
        outputs: outputMappings,
        textSplitMode: TextSplitMode.Pages,
        maximumPageLength: 4000);

    return splitSkill;
}

```

## Compétence de reconnaissance d'entités

Cette instance `EntityRecognitionSkill` est configurée pour reconnaître le type de catégorie `organization`. La compétence **Reconnaissance d'entités** peut également reconnaître les types de catégories `person` et `location`.

Notez que le champ « contexte » contient la valeur `"/document/pages/*"` avec un astérisque, ce qui signifie que l'étape d'enrichissement est appelée pour chaque page sous `"/document/pages"`.

```
private static EntityRecognitionSkill CreateEntityRecognitionSkill()
{
    List<InputFieldMappingEntry> inputMappings = new List<InputFieldMappingEntry>
    {
        new InputFieldMappingEntry(
            name: "text",
            source: "/document/pages/*")
    };

    List<OutputFieldMappingEntry> outputMappings = new List<OutputFieldMappingEntry>
    {
        new OutputFieldMappingEntry(
            name: "organizations",
            targetName: "organizations")
    };

    List<EntityCategory> entityCategory = new List<EntityCategory>
    {
        EntityCategory.Organization
    };

    EntityRecognitionSkill entityRecognitionSkill = new EntityRecognitionSkill(
        description: "Recognize organizations",
        context: "/document/pages/*",
        inputs: inputMappings,
        outputs: outputMappings,
        categories: entityCategory,
        defaultLanguageCode: EntityRecognitionSkillLanguage.En);

    return entityRecognitionSkill;
}
```

### Compétence Extraction de phrases clés

Comme l'instance `EntityRecognitionSkill` précédemment créée, la compétence **Extraction de phrases clés** est appelée pour chaque page du document.

```

private static KeyPhraseExtractionSkill CreateKeyPhraseExtractionSkill()
{
    List<InputFieldMappingEntry> inputMappings = new List<InputFieldMappingEntry>();
    inputMappings.Add(new InputFieldMappingEntry(
        name: "text",
        source: "/document/pages/*"));
    inputMappings.Add(new InputFieldMappingEntry(
        name: "languageCode",
        source: "/document/languageCode"));

    List<OutputFieldMappingEntry> outputMappings = new List<OutputFieldMappingEntry>();
    outputMappings.Add(new OutputFieldMappingEntry(
        name: "keyPhrases",
        targetName: "keyPhrases"));

    KeyPhraseExtractionSkill keyPhraseExtractionSkill = new KeyPhraseExtractionSkill(
        description: "Extract the key phrases",
        context: "/document/pages/*",
        inputs: inputMappings,
        outputs: outputMappings);

    return keyPhraseExtractionSkill;
}

```

## Générer et créer l'ensemble de compétences

Générez l'`Skillset` en utilisant les compétences que vous avez créées.

```

private static Skillset CreateOrUpdateDemoSkillSet(SearchServiceClient serviceClient, IList<Skill> skills)
{
    Skillset skillset = new Skillset(
        name: "demoskillset",
        description: "Demo skillset",
        skills: skills);

    // Create the skillset in your search service.
    // The skillset does not need to be deleted if it was already created
    // since we are using the CreateOrUpdate method
    try
    {
        serviceClient.Skillsets.CreateOrUpdate(skillset);
    }
    catch (Exception e)
    {
        Console.WriteLine("Failed to create the skillset\n Exception message: {0}\n", e.Message);
        ExitProgram("Cannot continue without a skillset");
    }

    return skillset;
}

```

Ajoutez les lignes suivantes à `Main`.

```

// Create the skills
Console.WriteLine("Creating the skills...");
OcrSkill ocrSkill = CreateOcrSkill();
MergeSkill mergeSkill = CreateMergeSkill();
EntityRecognitionSkill entityRecognitionSkill = CreateEntityRecognitionSkill();
LanguageDetectionSkill languageDetectionSkill = CreateLanguageDetectionSkill();
SplitSkill splitSkill = CreateSplitSkill();
KeyPhraseExtractionSkill keyPhraseExtractionSkill = CreateKeyPhraseExtractionSkill();

// Create the skillset
Console.WriteLine("Creating or updating the skillset...");
List<Skill> skills = new List<Skill>
{
    ocrSkill,
    mergeSkill,
    languageDetectionSkill,
    splitSkill,
    entityRecognitionSkill,
    keyPhraseExtractionSkill
};

Skillset skillset = CreateOrUpdateDemoSkillSet(serviceClient, skills);

```

### Étape 3 : Création d'un index

Dans cette section, vous définissez le schéma d'index en spécifiant les champs à inclure dans l'index de recherche et les attributs de recherche pour chaque champ. Les champs ont un type et peuvent prendre des attributs qui déterminent la façon dont le champ est utilisé (pour la recherche, le tri, etc.). Les noms des champs dans un index ne sont pas tenus de correspondre exactement aux noms des champs dans la source. Dans une étape ultérieure, vous ajoutez des mappages de champs dans un indexeur pour connecter les champs sources et de destination. Pour cette étape, définissez l'index à l'aide de conventions d'affectation de noms de champs appropriées à votre application de recherche.

Cet exercice utilise les champs et les types de champ suivants :

NOMS DE CHAMPS	TYPES DE CHAMP
id	Edm.String
content	Edm.String
languageCode	Edm.String
keyPhrases	List<Edm.String>
organizations	List<Edm.String>

#### Créer la classe DemoIndex

Les champs de cet index sont définis à l'aide d'une classe de modèle. Chaque propriété de la classe de modèle comporte des attributs qui déterminent les comportements liés à la recherche du champ d'index correspondant.

Nous allons ajouter la classe de modèle à un nouveau fichier C#. Cliquez avec le bouton droit sur votre projet, sélectionnez **Ajouter > Nouvel élément...**, sélectionnez « Classe » et nommez le fichier `DemoIndex.cs`, puis sélectionnez **Ajouter**.

Veuillez à indiquer que vous souhaitez utiliser des types à partir des espaces de noms `Microsoft.Azure.Search` et `Microsoft.Azure.Search.Models`.

Ajoutez la définition de classe de modèle ci-dessous à `DemoIndex.cs` et incluez-la dans le même espace de noms que celui où vous créerez l'index.

```
using Microsoft.Azure.Search;
using Microsoft.Azure.Search.Models;

namespace EnrichwithAI
{
    // The SerializePropertyNamesAsCamelCase attribute is defined in the Azure Search .NET SDK.
    // It ensures that Pascal-case property names in the model class are mapped to camel-case
    // field names in the index.
    [SerializePropertyNamesAsCamelCase]
    public class DemoIndex
    {
        [System.ComponentModel.DataAnnotations.Key]
        [IsSearchable, IsSortable]
        public string Id { get; set; }

        [IsSearchable]
        public string Content { get; set; }

        [IsSearchable]
        public string LanguageCode { get; set; }

        [IsSearchable]
        public string[] KeyPhrases { get; set; }

        [IsSearchable]
        public string[] Organizations { get; set; }
    }
}
```

Maintenant que vous avez défini une classe de modèle, de retour dans `Program.cs`, vous pouvez créer facilement une définition d'index. Le nom de cet index sera `demoindex`. Si un index portant ce nom existe déjà, il sera supprimé.

```
private static Index CreateDemoIndex(SearchServiceClient serviceClient)
{
    var index = new Index()
    {
        Name = "demoindex",
        Fields = FieldBuilder.BuildForType<DemoIndex>()
    };

    try
    {
        bool exists = serviceClient.Indexes.Exists(index.Name);

        if (exists)
        {
            serviceClient.Indexes.Delete(index.Name);
        }

        serviceClient.Indexes.Create(index);
    }
    catch (Exception e)
    {
        Console.WriteLine("Failed to create the index\n Exception message: {0}\n", e.Message);
        ExitProgram("Cannot continue without an index");
    }

    return index;
}
```

Pendant le test, vous serez peut-être amené à tenter de créer l'index plusieurs fois. Dès lors, vérifiez si l'index que vous vous apprêtez à créer existe avant de tenter de le créer.

Ajoutez les lignes suivantes à `Main`.

```
// Create the index
Console.WriteLine("Creating the index...");
Microsoft.Azure.Search.Models.Index demoIndex = CreateDemoIndex(serviceClient);
```

Ajoutez l'instruction `using` suivante pour résoudre la référence ambiguë.

```
using Index = Microsoft.Azure.Search.Models.Index;
```

Pour en savoir plus sur la définition d'un index, consultez [Créer un index \(API REST de la Recherche cognitive Azure\)](#).

#### Étape 4 : Créer et exécuter un indexeur

À ce stade, vous avez créé une source de données, un ensemble de compétences et un index. Ces trois composants deviennent partie intégrante d'un [indexeur](#) qui extrait chaque élément pour l'insérer dans une opération unique à plusieurs phases. Pour lier ces éléments en un indexeur, vous devez définir des mappages de champs.

- Les `fieldMappings` sont traités avant l'ensemble de compétences, en mappant les champs sources de la source de données sur des champs cibles dans un index. Si les noms et types de champ sont identiques aux deux extrémités, aucun mappage n'est nécessaire.
- Les `outputFieldMappings` sont traités après l'ensemble de compétences, en référençant les `sourceFieldNames` qui n'existent pas tant que le décodage de document ou l'enrichissement ne les ont pas créés. `targetFieldName` est un champ dans un index.

En plus de lier des entrées à des sorties, vous pouvez également utiliser les mappages de champs pour aplatisir les structures de données. Pour plus d'informations, consultez [Guide pratique pour mapper des champs enrichis sur un index pouvant faire l'objet d'une recherche](#).

```
private static Indexer CreateDemoIndexer(SearchServiceClient serviceClient, DataSource dataSource, Skillset skillSet, Index index)
{
    IDictionary<string, object> config = new Dictionary<string, object>();
    config.Add(
        key: "dataToExtract",
        value: "contentAndMetadata");
    config.Add(
        key: "imageAction",
        value: "generateNormalizedImages");

    List<FieldMapping> fieldMappings = new List<FieldMapping>
    {
        new FieldMapping(
            sourceFieldName: "metadata_storage_path",
            targetFieldName: "id",
            mappingFunction: new FieldMappingFunction(
                name: "base64Encode")),
        new FieldMapping(
            sourceFieldName: "content",
            targetFieldName: "content")
    };

    List<FieldMapping> outputMappings = new List<FieldMapping>
    {
        new FieldMapping(
            sourceFieldName: "/document/pages/*/organizations/*",
            targetFieldName: "parentOrganizationId",
            mappingFunction: new FieldMappingFunction(
                name: "base64Encode"))
    };
}
```

```

        targetFieldName: "organizations",
        new FieldMapping(
        sourceFieldName: "/document/pages/*/keyPhrases/*",
        targetFieldName: "keyPhrases"),
        new FieldMapping(
        sourceFieldName: "/document/languageCode",
        targetFieldName: "languageCode")
    );
}

Indexer indexer = new Indexer(
    name: "demoindexer",
    dataSourceName: dataSource.Name,
    targetIndexName: index.Name,
    description: "Demo Indexer",
    skillsetName: skillSet.Name,
    parameters: new IndexingParameters(
        maxFailedItems: -1,
        maxFailedItemsPerBatch: -1,
        configuration: config),
    fieldMappings: fieldMappings,
    outputFieldMappings: outputMappings);

try
{
    bool exists = serviceClient.Indexers.Exists(indexer.Name);

    if (exists)
    {
        serviceClient.Indexers.Delete(indexer.Name);
    }

    serviceClient.Indexers.Create(indexer);
}
catch (Exception e)
{
    Console.WriteLine("Failed to create the indexer\n Exception message: {0}\n", e.Message);
    ExitProgram("Cannot continue without creating an indexer");
}

return indexer;
}

```

Ajoutez les lignes suivantes à `Main`.

```

// Create the indexer, map fields, and execute transformations
Console.WriteLine("Creating the indexer and executing the pipeline...");
Indexer demoIndexer = CreateDemoIndexer(serviceClient, dataSource, skillset, demoIndex);

```

Notez que la création de l'indexeur requiert un certain temps. Même si le jeu de données est petit, les compétences analytiques exigent des calculs intensifs. Certaines compétences, telles que l'analyse d'image, sont des opérations longues.

#### TIP

La création d'un indexeur appelle le pipeline. En cas de problèmes pour atteindre les données, mapper les entrées et les sorties, ou ordonner les opérations, ceux-ci apparaissent à ce stade.

## Explorer la création de l'indexeur

Le code affecte à `"maxFailedItems"` la valeur -1, ce qui indique au moteur d'indexation d'ignorer les erreurs au cours de l'importation des données. Cela est utile car très peu de documents figurent dans la source de données de démonstration. Pour une source de données plus volumineuse, vous définiriez une valeur supérieure à 0.

Notez également la définition de `"dataToExtract"` sur `"contentAndMetadata"`. Cette instruction indique à l'indexeur d'extraire automatiquement le contenu de fichiers de différents formats, ainsi que les métadonnées associées à chaque fichier.

Lorsque le contenu est extrait, vous pouvez définir `imageAction` pour extraire le texte des images trouvées dans la source de données. La configuration `"imageAction"` définie sur `"generateNormalizedImages"`, associée à la compétence de reconnaissance optique des caractères et à la compétence de fusion de texte, indique à l'indexeur d'extraire le texte des images (par exemple, le mot « stop » d'un panneau de signalisation Stop) et de l'incorporer dans le champ de contenu. Ce comportement s'applique aux images intégrées dans les documents (pensez à une image dans un fichier PDF), ainsi qu'aux images trouvées dans la source de données, par exemple un fichier JPG.

## 4 - Surveiller l'indexation

Une fois l'indexeur défini, il s'exécute automatiquement lorsque vous envoyez la demande. Selon les compétences cognitives que vous avez définies, l'indexation peut prendre plus de temps que prévu. Pour savoir si l'indexeur est toujours en cours d'exécution, utilisez la méthode `GetStatus`.

```
private static void CheckIndexerOverallStatus(SearchServiceClient serviceClient, Indexer indexer)
{
    try
    {
        IndexerExecutionInfo demoIndexerExecutionInfo = serviceClient.Indexers.GetStatus(indexer.Name);

        switch (demoIndexerExecutionInfo.Status)
        {
            case IndexerStatus.Error:
                ExitProgram("Indexer has error status. Check the Azure Portal to further understand the
error.");
                break;
            case IndexerStatus.Running:
                Console.WriteLine("Indexer is running");
                break;
            case IndexerStatus.Unknown:
                Console.WriteLine("Indexer status is unknown");
                break;
            default:
                Console.WriteLine("No indexer information");
                break;
        }
    }
    catch (Exception e)
    {
        Console.WriteLine("Failed to get indexer overall status\n Exception message: {0}\n", e.Message);
    }
}
```

`IndexerExecutionInfo` correspond à l'historique actuel d'état et d'exécution d'un indexeur.

Les avertissements sont courants avec certaines combinaisons de fichiers sources et de compétences et n'indiquent pas toujours un problème. Dans ce tutoriel, les avertissements sont sans gravité (par exemple, aucune entrée de texte à partir des fichiers JPEG).

Ajoutez les lignes suivantes à `Main`.

```
// Check indexer overall status
Console.WriteLine("Check the indexer overall status...");
CheckIndexerOverallStatus(serviceClient, demoIndexer);
```

## 5 - Recherche

Une fois l'indexation terminée, vous pouvez exécuter des requêtes qui renvoient le contenu de champs individuels. Par défaut, le service Recherche cognitive Azure retourne les 50 meilleurs résultats. Les données d'exemple sont petites si bien que la configuration par défaut fonctionne correctement. Toutefois, lorsque vous travaillez avec des jeux de données plus volumineux, vous pouvez être amené à inclure des paramètres dans la chaîne de requête pour retourner plus de résultats. Pour obtenir des instructions, consultez [Guide pratique pour présenter les résultats dans la Recherche cognitive Azure](#).

Comme étape de vérification, interrogez l'index au sujet de tous les champs.

Ajoutez les lignes suivantes à `Main`.

```
DocumentSearchResult<DemoIndex> results;

ISearchIndexClient indexClientForQueries = CreateSearchIndexClient(configuration);

SearchParameters parameters =
    new SearchParameters
    {
        Select = new[] { "organizations" }
    };

try
{
    results = indexClientForQueries.Documents.Search<DemoIndex>("*", parameters);
}
catch (Exception e)
{
    // Handle exception
}
```

`CreateSearchIndexClient` crée un `SearchIndexClient` à l'aide de valeurs stockées dans le fichier config de l'application (`appsettings.json`). Notez que la clé de l'API de requête du service de recherche est utilisée, et non pas la clé d'administration.

```
private static SearchIndexClient CreateSearchIndexClient(IConfigurationRoot configuration)
{
    string searchServiceName = configuration["SearchServiceName"];
    string queryApiKey = configuration["SearchServiceQueryApiKey"];

    SearchIndexClient indexClient = new SearchIndexClient(searchServiceName, "demoindex", new
    SearchCredentials(queryApiKey));
    return indexClient;
}
```

Ajoutez le code suivant à `Main`. Le premier try-catch retourne la définition de l'index, avec le nom, le type et les attributs de chaque champ. Le deuxième est une requête paramétrable, où `Select` spécifie les champs à inclure dans les résultats, par exemple `organizations`. Une chaîne de recherche de `"*"` retourne tout le contenu d'un même champ.

```

//Verify content is returned after indexing is finished
ISearchIndexClient indexClientForQueries = CreateSearchIndexClient(configuration);

try
{
    results = indexClientForQueries.Documents.Search<DemoIndex>("*");
    Console.WriteLine("First query succeeded with a result count of {0}", results.Results.Count);
}
catch (Exception e)
{
    Console.WriteLine("First query failed\n Exception message: {0}\n", e.Message);
}

SearchParameters parameters =
new SearchParameters
{
    Select = new[] { "organizations" }
};

try
{
    results = indexClientForQueries.Documents.Search<DemoIndex>("*", parameters);
    Console.WriteLine("Second query succeeded with a result count of {0}", results.Results.Count);
}
catch (Exception e)
{
    Console.WriteLine("Second query failed\n Exception message: {0}\n", e.Message);
}

```

Répétez ces étapes pour les champs supplémentaires : content, languageCode, keyPhrases et organizations dans cet exercice. Vous pouvez retourner plusieurs champs via la propriété **Select** en utilisant une liste délimitée par des virgules.

## Réinitialiser et réexécuter

Dans les premières étapes expérimentales du développement, l'approche la plus pratique pour les itérations de conception consiste à supprimer les objets de Recherche cognitive Azure et à autoriser votre code à les reconstruire. Les noms des ressources sont uniques. La suppression d'un objet vous permet de le recréer en utilisant le même nom.

L'exemple de code pour ce tutoriel recherche les objets existants et les supprime pour vous permettre de réexécuter votre code.

Vous pouvez aussi utiliser le portail pour supprimer les index, les indexeurs et les ensembles de compétences.

## Éléments importants à retenir

Ce tutoriel présente les étapes de base pour générer un pipeline d'indexation enrichie via la création de composants : une source de données, un ensemble de compétences, un index et un indexeur.

Les **compétences intégrées** ont été introduites, ainsi que la définition d'un ensemble de compétences et les mécanismes de chaînage de compétences via des entrées et des sorties. Vous avez également appris que `outputFieldMappings` est requis dans la définition de l'indexeur pour acheminer les valeurs enrichies du pipeline dans un index de recherche, sur un service de Recherche cognitive Azure.

Enfin, vous avez appris à tester les résultats et réinitialiser le système pour des itérations ultérieures. Vous avez appris qu'émettre des requêtes par rapport à l'index retourne la sortie créée par le pipeline d'indexation enrichie. Vous avez également appris à vérifier l'état de l'indexeur et quels objets supprimer avant de réexécuter un pipeline.

## Nettoyer les ressources

Lorsque vous travaillez dans votre propre abonnement, il est judicieux à la fin d'un projet de supprimer les ressources dont vous n'avez plus besoin. Les ressources laissées en cours d'exécution peuvent vous coûter de l'argent. Vous pouvez supprimer les ressources une par une, ou choisir de supprimer le groupe de ressources afin de supprimer l'ensemble des ressources.

Vous pouvez rechercher et gérer les ressources dans le portail à l'aide des liens Toutes les ressources ou Groupes de ressources situés dans le volet de navigation de gauche.

## Étapes suivantes

Maintenant que vous êtes familiarisé avec tous les objets d'un pipeline d'enrichissement par IA, examinons de plus près les définitions des ensembles de compétences et les compétences individuelles.

[Guide pratique pour créer un ensemble de compétences](#)

# Tutoriel : Utiliser REST et l'IA pour générer du contenu pouvant faire l'objet de recherches à partir d'objets blob Azure

04/10/2020 • 32 minutes to read • [Edit Online](#)

Si vous avez du texte non structuré ou des images dans Stockage Blob Azure, un [pipeline d'enrichissement par IA](#) peut extraire des informations et créer du contenu utile pour les scénarios de recherche en texte intégral ou d'exploration de connaissances. Bien qu'un pipeline puisse traiter des images, ce tutoriel REST se concentre sur du texte, en appliquant la détection de la langue et le traitement en langage naturel pour créer des champs exploitables dans des requêtes, des facettes et des filtres.

Ce tutoriel utilise Postman et les [API REST de Recherche](#) pour effectuer les tâches suivantes :

- Commencez avec des documents entiers (texte non structuré), comme des documents PDF, HTML, DOCX et PPTX, dans Stockage Blob Azure.
- Définissez un pipeline qui extrait du texte, détecte la langue, reconnaît les entités et détecte les expressions clés.
- Définissez un index pour stocker la sortie (contenu brut et paires nom-valeur générées par le pipeline).
- Exécutez le pipeline pour démarrer des transformations et une analyse, ainsi que pour créer et charger l'index.
- Explorez les résultats à l'aide de la recherche en texte intégral et d'une syntaxe de requête enrichie.

Si vous n'avez pas d'abonnement Azure, ouvrez un [compte gratuit](#) avant de commencer.

## Prérequis

- [Stockage Azure](#)
- [Application de bureau Postman](#)
- [Créer ou rechercher un service de recherche existant](#)

### NOTE

Vous pouvez utiliser le service gratuit pour ce tutoriel. Avec un service de recherche gratuit, vous êtes limité à trois index, trois indexeurs et trois sources de données. Ce didacticiel crée une occurrence de chaque élément. Avant de commencer, veillez à disposer de l'espace suffisant sur votre service pour accepter les nouvelles ressources.

## Télécharger les fichiers

1. Ouvrez ce [dossier OneDrive](#) et en haut à gauche, cliquez sur **Télécharger** pour copier les fichiers sur votre ordinateur.
2. Cliquez avec le bouton droit sur le fichier zip et sélectionnez **Tout extraire**. Il y a 14 fichiers de différents types. Vous en utiliserez 7 dans le cadre de cet exercice.

## 1 - Créer les services

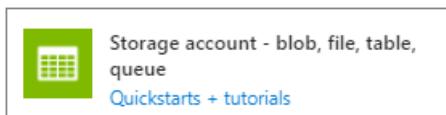
Ce tutoriel utilise Recherche cognitive Azure pour l'indexation et les requêtes, Cognitive Services pour l'enrichissement par IA sur le back-end, et Stockage Blob Azure pour fournir les données. Ce tutoriel reste sous l'allocation gratuite de 20 transactions par indexeur par jour sur Cognitive Services : les seuls services que vous

devez créer sont donc la recherche et le stockage.

Si possible, créez les deux services dans la même région et le même groupe de ressources pour des raisons de proximité et de facilité de gestion. Dans la pratique, votre compte de stockage Azure peut être dans une région quelconque.

### Démarrer avec le stockage Azure

1. [Connectez-vous au portail Azure](#) et cliquez sur + **Créer une ressource**.
2. Recherchez un *compte de stockage* et sélectionnez l'offre de compte de stockage Microsoft.



3. Sous l'onglet **Bases**, les éléments suivants sont obligatoires. Acceptez les valeurs par défaut pour tout le reste.
  - **Groupe de ressources.** Sélectionnez un groupe existant ou créez-en un, mais utilisez le même groupe pour tous les services afin de pouvoir les gérer collectivement.
  - **Nom du compte de stockage.** Si vous pensez que vous pouvez avoir plusieurs ressources du même type, utilisez le nom pour lever l'ambiguïté par type et par région, par exemple *blobstoragewestus*.
  - **Emplacement.** Si possible, choisissez le même emplacement que celui utilisé pour Recherche cognitive Azure et Cognitive Services. Un emplacement unique annule les frais liés à la bande passante.
  - **Type de compte.** Choisissez la valeur par défaut, *StorageV2 (v2 universel)*.
4. Cliquez sur **Vérifier + créer** pour créer le service.
5. Une fois qu'il est créé, cliquez sur **Accéder à la ressource** pour ouvrir la page Vue d'ensemble.
6. Cliquez sur le service **Objets blob**.
7. Cliquez sur + **Conteneur** pour créer un conteneur et nommez-le *cog-search-demo*.
8. Sélectionnez *cog-search-demo*, puis cliquez sur **Charger** pour ouvrir le dossier dans lequel vous avez enregistré les fichiers téléchargés. Sélectionnez tous les fichiers autres que des images. Vous devez disposer de 7 fichiers. Cliquez sur **OK** pour effectuer le chargement.

<input type="checkbox"/> Name	Date modified	Type	Size
<input checked="" type="checkbox"/> Cognitive Services and Bots (spanish)....	8/22/2019 10:51 PM	Adobe Acrobat D...	3,884 KB
<input checked="" type="checkbox"/> MSFT_cloud_architecture_contoso.pdf	8/22/2019 10:51 PM	Adobe Acrobat D...	1,805 KB
<input checked="" type="checkbox"/> NYSE_LNKD_2015.PDF	8/22/2019 10:51 PM	Adobe Acrobat D...	394 KB
<input checked="" type="checkbox"/> 10-K-FY16.html	8/22/2019 10:51 PM	HTML File	1,835 KB
guthrie.jpg	8/22/2019 10:51 PM	JPG File	46 KB
redshirt.jpg	8/22/2019 10:51 PM	JPG File	67 KB
satyanadellalinux.jpg	8/22/2019 10:51 PM	JPG File	108 KB
<input checked="" type="checkbox"/> Cognitive Searvices and Content Intelli...	8/22/2019 10:51 PM	Microsoft PowerPo...	11,231 KB
<input checked="" type="checkbox"/> MSFT_FY17_10K.docx	8/22/2019 10:51 PM	Microsoft Word D...	675 KB
5074.clip_image002_6FE27E85.png	8/22/2019 10:51 PM	PNG File	472 KB
create-search-collect-info.png	8/22/2019 10:51 PM	PNG File	57 KB
create-search-service.png	8/22/2019 10:51 PM	PNG File	26 KB
create-service-full-portal.png	8/22/2019 10:51 PM	PNG File	67 KB
<input checked="" type="checkbox"/> satyasletter.txt	8/22/2019 10:51 PM	Text Document	6 KB

9. Avant de quitter Stockage Azure, obtenez une chaîne de connexion afin de pouvoir formuler une connexion dans Recherche cognitive Azure.

- a. Revenez à la page Vue d'ensemble de votre compte de stockage (nous avons utilisé *blobstragewestus* comme exemple).
- b. Dans le volet de navigation gauche, sélectionnez Clés d'accès et copiez l'une des chaînes de connexion.

La chaîne de connexion est une URL similaire à l'exemple suivant :

```
DefaultEndpointsProtocol=https;AccountName=cogsrchdemostorage;AccountKey=<your account key>;EndpointSuffix=core.windows.net
```

10. Enregistrez la chaîne de connexion dans le Bloc-Notes. Vous en aurez besoin plus tard lors de la configuration de la connexion à la source de données.

## Cognitive Services

L'enrichissement par IA s'appuie sur Cognitive Services, notamment Analyse de texte et Vision par ordinateur pour le traitement des images et du langage naturel. Si votre objectif était de réaliser un prototype ou un projet réel, vous devriez à ce stade provisionner Cognitive Services (dans la même région que Recherche cognitive Azure) afin de pouvoir l'attacher aux opérations d'indexation.

Cependant, dans le cadre de cet exercice, vous pouvez ignorer le provisionnement des ressources, car Recherche cognitive Azure peut se connecter à Cognitive Services en arrière-plan et vous fournir 20 transactions gratuites par exécution de l'indexeur. Comme ce tutoriel utilise 7 transactions, l'allocation gratuite est suffisante. Pour les projets de plus grande envergure, prévoyez de provisionner Cognitive Services au niveau S0 du paiement à l'utilisation. Pour plus d'informations, consultez [Attacher Cognitive Services](#).

## Recherche cognitive Azure

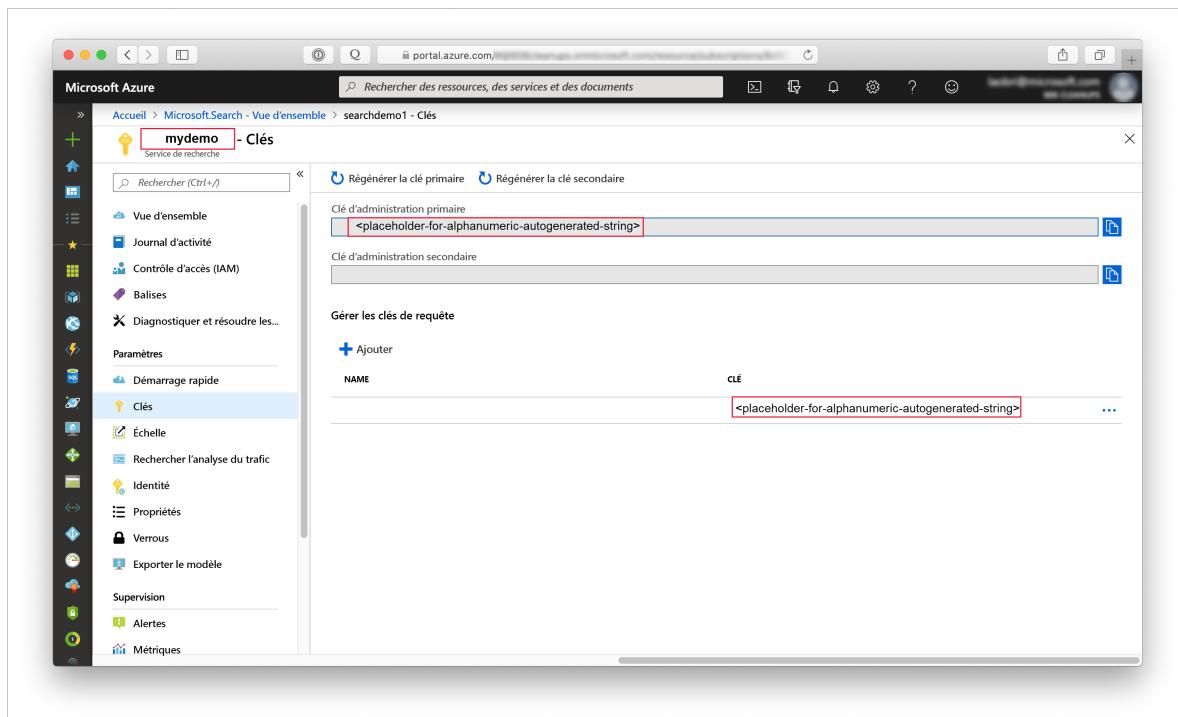
Le troisième composant est Recherche cognitive Azure, que vous pouvez [créer dans le portail](#). Vous pouvez utiliser le niveau gratuit pour effectuer cette procédure pas à pas.

Comme avec le stockage Blob Azure, prenez un moment pour collecter la clé d'accès. Par ailleurs, lorsque vous commencez à structurer les demandes, vous devez fournir le point de terminaison et la clé API d'administration utilisés pour authentifier chaque demande.

## Obtenir une clé API d'administration et une URL pour Recherche cognitive Azure

1. Connectez-vous au Portail Azure, puis dans la page Vue d'ensemble du service de recherche, récupérez le nom de votre service de recherche. Vous pouvez confirmer le nom de votre service en passant en revue l'URL du point de terminaison. Si votre URL de point de terminaison est `https://mydemo.search.windows.net`, le nom du service doit être `mydemo`.
2. Dans Paramètres > Clés, obtenez une clé d'administration pour avoir des droits d'accès complets sur le service. Il existe deux clés d'administration interchangeables, fournies pour assurer la continuité de l'activité au cas où vous deviez en remplacer une. Vous pouvez utiliser la clé primaire ou secondaire sur les demandes d'ajout, de modification et de suppression d'objets.

Obtenez aussi la clé de requête. Il est recommandé d'émettre des demandes de requête avec un accès en lecture seule.



Une clé API est nécessaire dans l'en-tête de chaque requête envoyée à votre service. Une clé valide permet d'établir, en fonction de chaque demande, une relation de confiance entre l'application qui envoie la demande et le service qui en assure le traitement.

## 2 - Configurer Postman

Démarrez Postman et paramétrez une requête HTTP. Si vous ne connaissez pas bien cet outil, consultez [Explorer les API REST de Recherche cognitive Azure avec Postman](#).

Les méthodes de requête utilisées dans ce tutoriel sont POST, PUT et GET. Vous allez utiliser les méthodes permettant d'effectuer quatre appels d'API vers votre service de recherche afin de créer une source de données, un ensemble de compétences, un index et un indexeur.

Dans les en-têtes, définissez « Content-type » sur `application/json` et `api-key` sur la clé API d'administration de votre service Recherche cognitive Azure. Une fois que vous avez défini les en-têtes, vous pouvez les utiliser pour chaque demande de cet exercice.

The screenshot shows the Postman interface with a red box highlighting the URL and the 'Headers' tab. Under 'Headers', two entries are selected: 'Content-Type: application/json' and 'api-key: <placeholder-api-key-for-your-service>'. A red box also highlights the 'Status' field showing '200 OK' and the 'Duration' field showing '540 ms'.

## 3 - Créer le pipeline

Dans Recherche cognitive Azure, le traitement de l'IA se produit pendant l'indexation (ou l'ingestion de données). Cette partie de la procédure pas à pas crée quatre objets : source de données, définition d'index, ensemble de compétences, indexeur.

### Étape 1 : Création d'une source de données

Un [objet Source de données](#) fournit la chaîne de connexion au conteneur d'objets blob contenant les fichiers.

1. Utilisez **POST** et l'URL suivante, en remplaçant YOUR-SERVICE-NAME par le véritable nom de votre service.

```
https://[YOUR-SERVICE-NAME].search.windows.net/datasources?api-version=2020-06-30
```

2. Dans le **corps** de la demande, copiez la définition JSON suivante, en remplaçant `connectionString` par la connexion réelle de votre compte de stockage.

Pensez à modifier également le nom du conteneur. Nous avons suggéré "cog-search-demo" comme nom du conteneur dans une étape précédente.

```
{
  "name" : "cog-search-demo-ds",
  "description" : "Demo files to demonstrate cognitive search capabilities.",
  "type" : "azureblob",
  "credentials" :
  { "connectionString" :
    "DefaultEndpointsProtocol=https;AccountName=<YOUR-STORAGE-ACCOUNT>;AccountKey=<YOUR-ACCOUNT-KEY>;"
  },
  "container" : { "name" : "<YOUR-BLOB-CONTAINER-NAME>" }
}
```

3. Envoyez la demande. Vous devriez voir un code d'état 201 confirmant la réussite.

Si vous avez obtenu une erreur 403 ou 404, vérifiez la construction de la demande : `api-version=2020-06-30` doit être sur le point de terminaison, `api-key` doit être dans l'en-tête après `Content-Type` et sa valeur doit être valide pour un service de recherche. Vous souhaiterez peut-être exécuter le document JSON via un validateur JSON en ligne pour vous assurer que la syntaxe est correcte.

### Étape 2 : Créer un ensemble de compétences

Un [objet Ensemble de compétences](#) est un ensemble d'étapes d'enrichissement appliquées à votre contenu.

1. Utilisez **PUT** et l'URL suivante, en remplaçant YOUR-SERVICE-NAME par le véritable nom de votre service.

```
https://[YOUR-SERVICE-NAME].search.windows.net/skillsets/cog-search-demo-sd?api-version=2020-06-30
```

2. Dans le **corps** de la demande, copiez la définition JSON ci-dessous. Cet ensemble de compétences

comprend les compétences intégrées suivantes.

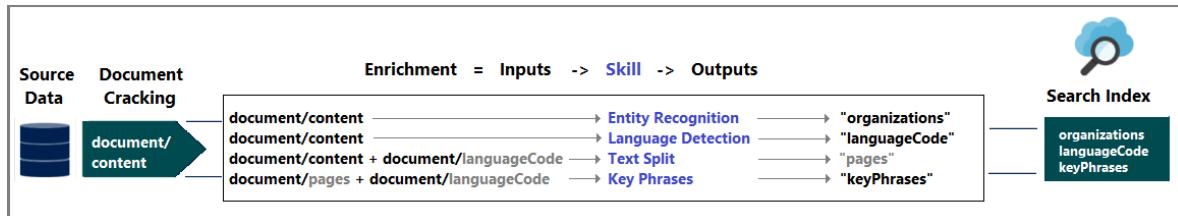
COMPÉTENCE	DESCRIPTION
Reconnaissance d'entités	Extrait les noms des personnes, des organisations et des emplacements à partir du contenu figurant dans le conteneur d'objets blob.
Détection de la langue	Déetecte la langue du contenu.
Division du texte	Divise un long contenu en plus petits morceaux avant d'appeler la compétence d'extraction de phrases clés. L'extraction de phrases clés accepte des entrées de 50 000 caractères au maximum. Certains fichiers d'exemple doivent être fractionnés pour satisfaire cette limite.
Extraction d'expressions clés	Extrait les principales expressions clés.

Chaque compétence s'exécute sur le contenu du document. Au cours du traitement, Recherche cognitive Azure ouvre chaque document pour lire le contenu de différents formats de fichiers. Le texte trouvé original du fichier source est placé dans un champ `content` généré, un pour chaque document. Par conséquent, l'entrée devient `"/document/content"`.

Pour l'extraction d'expressions clés, comme nous utilisons la compétence de division de texte pour découper des fichiers volumineux en pages, le contexte de la compétence d'extraction d'expressions clés est `"document/pages/*"` (pour chaque page du document) au lieu de `"/document/content"`.

```
{
  "description": "Extract entities, detect language and extract key-phrases",
  "skills": [
    [
      {
        "@odata.type": "#Microsoft.Skills.Text.EntityRecognitionSkill",
        "categories": [ "Person", "Organization", "Location" ],
        "defaultLanguageCode": "en",
        "inputs": [
          { "name": "text", "source": "/document/content" }
        ],
        "outputs": [
          { "name": "persons", "targetName": "persons" },
          { "name": "organizations", "targetName": "organizations" },
          { "name": "locations", "targetName": "locations" }
        ]
      },
      {
        "@odata.type": "#Microsoft.Skills.Text.LanguageDetectionSkill",
        "inputs": [
          { "name": "text", "source": "/document/content" }
        ],
        "outputs": [
          { "name": "languageCode", "targetName": "languageCode" }
        ]
      },
      {
        "@odata.type": "#Microsoft.Skills.Text.SplitSkill",
        "textSplitMode" : "pages",
        "maximumPageLength": 4000,
        "inputs": [
          { "name": "text", "source": "/document/content" },
          { "name": "languageCode", "source": "/document/languageCode" }
        ],
        "outputs": [
          { "name": "textItems", "targetName": "pages" }
        ]
      },
      {
        "@odata.type": "#Microsoft.Skills.Text.KeyPhraseExtractionSkill",
        "context": "/document/pages/*",
        "inputs": [
          { "name": "text", "source": "/document/pages/*" },
          { "name": "languageCode", "source": "/document/languageCode" }
        ],
        "outputs": [
          { "name": "keyPhrases", "targetName": "keyPhrases" }
        ]
      }
    ]
  }
}
```

Une représentation graphique de l'ensemble de compétences est présentée ci-dessous.



- Envoyez la demande. Postman doit retourner un code d'état 201 confirmant la réussite.

## NOTE

Les sorties peuvent être mappées à un index, utilisées comme entrée d'une compétence en aval, ou les deux, comme c'est le cas avec le code de langue. Dans l'index, un code de langue est utile pour le filtrage. En tant qu'entrée, le code de langue est utilisé par les compétences d'analyse de texte pour informer les règles linguistiques en matière de césure de mots. Pour plus d'informations sur les principes de base des ensembles de compétences, consultez [Guide pratique pour définir un ensemble de compétences](#).

## Étape 3 : Création d'un index

Un **index** fournit le schéma utilisé pour créer l'expression physique de votre contenu dans les index inversés et d'autres constructions dans Recherche cognitive Azure. Le plus grand composant d'un index est la collection de champs, où le type de données et les attributs déterminent le contenu et les comportements dans Recherche cognitive Azure.

1. Utilisez **PUT** et l'URL suivante, en remplaçant YOUR-SERVICE-NAME par le véritable nom de votre service, pour nommer votre index.

```
https://[YOUR-SERVICE-NAME].search.windows.net/indexes/cog-search-demo-idx?api-version=2020-06-30
```

2. Dans le **corps** de la demande, copiez la définition JSON suivante. Le champ `content` stocke le document lui-même. Les champs supplémentaires `languageCode`, `keyPhrases` et `organizations` représentent de nouvelles informations (champs et valeurs) créées par l'ensemble de compétences.

```
{  
  "fields": [  
    {  
      "name": "id",  
      "type": "Edm.String",  
      "key": true,  
      "searchable": true,  
      "filterable": false,  
      "facetable": false,  
      "sortable": true  
    },  
    {  
      "name": "metadata_storage_name",  
      "type": "Edm.String",  
      "searchable": false,  
      "filterable": false,  
      "facetable": false,  
      "sortable": false  
    },  
    {  
      "name": "content",  
      "type": "Edm.String",  
      "sortable": false,  
      "searchable": true,  
      "filterable": false,  
      "facetable": false  
    },  
    {  
      "name": "languageCode",  
      "type": "Edm.String",  
      "searchable": true,  
      "filterable": false,  
      "facetable": false  
    },  
    {  
      "name": "keyPhrases",  
      "type": "Collection(Edm.String)",  
      "searchable": true  
    }  
  ]  
}
```

```

        "searchable": true,
        "filterable": false,
        "facetable": false
    },
    {
        "name": "persons",
        "type": "Collection(Edm.String)",
        "searchable": true,
        "sortable": false,
        "filterable": true,
        "facetable": true
    },
    {
        "name": "organizations",
        "type": "Collection(Edm.String)",
        "searchable": true,
        "sortable": false,
        "filterable": true,
        "facetable": true
    },
    {
        "name": "locations",
        "type": "Collection(Edm.String)",
        "searchable": true,
        "sortable": false,
        "filterable": true,
        "facetable": true
    }
]
}

```

- Envoyez la demande. Postman doit retourner un code d'état 201 confirmant la réussite.

#### Étape 4 : Créer et exécuter un indexeur

Un [indexeur](#) pilote le pipeline. Les trois composants que vous avez créés jusqu'à présent (source de données, ensemble de compétences, index) sont les entrées d'un indexeur. La création de l'indexeur sur Recherche cognitive Azure est l'événement qui met en mouvement la totalité du pipeline.

- Utilisez **PUT** et l'URL suivante, en remplaçant YOUR-SERVICE-NAME par le véritable nom de votre service, pour nommer votre indexeur.

```
https://[servicename].search.windows.net/indexers/cog-search-demo-idxr?api-version=2020-06-30
```

- Dans le **corps** de la demande, copiez la définition JSON ci-dessous. Notez les éléments de mappage de champs : ces mappages sont importants car ils définissent le flux de données.

Les éléments `fieldMappings` sont traités avant l'ensemble de compétences, en envoyant le contenu de la source de données aux champs cibles dans un index. Vous utiliserez des mappages de champs pour envoyer du contenu existant non modifié à l'index. Si les noms et types de champ sont identiques aux deux extrémités, aucun mappage n'est nécessaire.

Les éléments `outputFieldMappings` sont pour les champs créés par les compétences et donc traités après l'exécution de l'ensemble de compétences. Les références à `sourceFieldNames` dans `outputFieldMappings` n'existent pas tant que l'enrichissement ou le craquage de document ne les a pas créées. L'élément `targetFieldName` est un champ dans un index, défini dans le schéma d'index.

```
{
  "name": "cog-search-demo-idxr",
  "dataSourceName": "cog-search-demo-ds",
  "targetIndexName": "cog-search-demo-idx",
  "skillsetName": "cog-search-demo-ss",
  "fieldMappings": [
    {
      "sourceFieldName": "metadata_storage_path",
      "targetFieldName": "id",
      "mappingFunction": {
        "name": "base64Encode"
      }
    },
    {
      "sourceFieldName": "metadata_storage_name",
      "targetFieldName": "metadata_storage_name",
      "mappingFunction": {
        "name": "base64Encode"
      }
    },
    {
      "sourceFieldName": "content",
      "targetFieldName": "content"
    }
  ],
  "outputFieldMappings": [
    {
      "sourceFieldName": "/document/persons",
      "targetFieldName": "persons"
    },
    {
      "sourceFieldName": "/document/organizations",
      "targetFieldName": "organizations"
    },
    {
      "sourceFieldName": "/document/locations",
      "targetFieldName": "locations"
    },
    {
      "sourceFieldName": "/document/pages/*/keyPhrases/*",
      "targetFieldName": "keyPhrases"
    },
    {
      "sourceFieldName": "/document/languageCode",
      "targetFieldName": "languageCode"
    }
  ],
  "parameters": {
    "maxFailedItems": -1,
    "maxFailedItemsPerBatch": -1,
    "configuration": {
      "dataToExtract": "contentAndMetadata",
      "parsingMode": "default",
      "firstLineContainsHeaders": false,
      "delimitedTextDelimiter": ","
    }
  }
}
```

3. Envoyez la demande. Postman doit retourner un code d'état 201 confirmant la réussite du processus.

Attendez-vous à ce que cette étape prenne plusieurs minutes. Même si le jeu de données est petit, les compétences analytiques exigent des calculs intensifs.

#### NOTE

La création d'un indexeur appelle le pipeline. En cas de problèmes pour atteindre les données, mapper les entrées et les sorties, ou ordonner les opérations, ceux-ci apparaissent à ce stade. Pour réexécuter le pipeline avec des modifications de code ou de script, vous pouvez être amené à supprimer d'abord des objets. Pour plus d'informations, consultez [Réinitialiser et réexécuter](#).

#### À propos des paramètres de l'indexeur

Le script affecte à `"maxFailedItems"` la valeur -1, ce qui indique au moteur d'indexation d'ignorer les erreurs au cours de l'importation des données. C'est acceptable car très peu de documents figurent dans la source de données de démonstration. Pour une source de données plus volumineuse, vous définiriez une valeur supérieure à 0.

Cette instruction `"dataToExtract": "contentAndMetadata"` indique à l'indexeur d'extraire automatiquement le contenu de fichiers de différents formats, ainsi que les métadonnées associées à chaque fichier.

Lorsque le contenu est extrait, vous pouvez définir `imageAction` pour extraire le texte des images trouvées dans la source de données. La configuration `"imageAction": "generateNormalizedImages"`, associée à la compétence de reconnaissance optique des caractères et à la compétence de fusion de texte, indique à l'indexeur d'extraire le texte des images (par exemple, le mot « stop » d'un panneau de signalisation Stop) et de l'incorporer dans le champ de contenu. Ce comportement s'applique aux images intégrées dans les documents (pensez à une image dans un fichier PDF), ainsi qu'aux images trouvées dans la source de données, par exemple un fichier JPG.

## 4 - Surveiller l'indexation

L'indexation et l'enrichissement commencent dès que vous envoyez la demande de création d'indexeur. Selon les compétences cognitives que vous avez définies, l'indexation peut durer longtemps. Pour déterminer si l'indexeur est toujours en cours d'exécution, envoyez la demande suivante pour vérifier l'état de l'indexeur.

1. Utilisez GET et l'URL suivante, en remplaçant YOUR-SERVICE-NAME par le véritable nom de votre service, pour nommer votre indexeur.

```
https://[YOUR-SERVICE-NAME].search.windows.net/indexers/cog-search-demo-idxr/status?api-version=2020-06-30
```

2. Passez en revue la réponse pour savoir si l'indexeur est en cours d'exécution ou pour voir les informations d'erreur et d'avertissement.

Si vous utilisez le niveau Gratuit, le message suivant est attendu : « Impossible d'extraire le contenu ou les métadonnées de votre document. Texte extrait tronqué à '32768' caractères ». Ce message s'affiche parce que l'indexation d'objets blob au niveau Gratuit a une [limite d'extraction de caractères égale à 32 000](#). Vous ne verrez pas ce message pour ce jeu de données sur des niveaux supérieurs.

#### NOTE

Les avertissements sont courants dans certains scénarios et n'indiquent pas toujours un problème. Par exemple, si un conteneur d'objets blob comprend des fichiers image et que le pipeline ne gère pas les images, vous obtenez un avertissement indiquant que les images n'ont pas été traitées.

## 5 - Recherche

Maintenant que vous avez créé des champs et des informations, nous allons exécuter certaines requêtes pour comprendre la valeur de la recherche cognitive par rapport à un scénario de recherche standard.

Rappelez-vous que nous avons commencé avec le contenu d'objets blob là où le document entier est empaqueté dans un champ `content` unique. Vous pouvez rechercher dans ce champ et trouver des correspondances pour vos requêtes.

1. Utilisez **GET** et l'URL suivante, en remplaçant **YOUR-SERVICE-NAME** par le nom réel de votre service, pour rechercher des instances d'un terme ou d'une expression et retourner le champ `content` et le nombre des documents correspondants.

[https://\[YOUR-SERVICE-NAME\].search.windows.net/indexes/cog-search-demo-idx/docs?search=\\*&\\$count=true&\\$select=content&api-version=2020-06-30](https://[YOUR-SERVICE-NAME].search.windows.net/indexes/cog-search-demo-idx/docs?search=*&$count=true&$select=content&api-version=2020-06-30)

Les résultats de cette requête retournent le contenu du document, ce qui donne le même résultat que si vous utilisez l'indexeur d'objets blob sans le pipeline de recherche cognitive. Ce champ peut faire l'objet d'une recherche, mais il ne peut pas être utilisé si vous souhaitez utiliser des facettes, des filtres ou la saisie semi-automatique.

2. Pour la deuxième requête, retournez certains des nouveaux champs créés par le pipeline (personnes, organisations, emplacements, languageCode). Nous omettons keyPhrases par souci de concision, mais vous devez l'inclure si vous souhaitez voir ces valeurs.

```
https://[YOUR-SERVICE-NAME].search.windows.net/indexes/cog-search-demo-idx/docs?  
search=*&$count=true&$select=metadata_storage_name,persons,organizations,locations,languageCode&api-  
version=2020-06-30
```

Les champs de l'instruction \$select contiennent de nouvelles informations créées à partir des fonctionnalités de traitement du langage naturel de Cognitive Services. Comme vous pouvez vous y attendre, il existe un certain bruit dans les résultats et une certaine variation entre les documents, mais dans de nombreuses instances, les modèles analytiques produisent des résultats précis.

L'image suivante montre les résultats pour la lettre ouverte de Satya Nadella lorsqu'il a endossé le rôle de Directeur général de Microsoft.

```

637 {
638     "@search.score": 1.0,
639     "metadata_storage_name": "satyasletter.txt",
640     "languageCode": "en",
641     "persons": [
642         "Steve",
643         "Bill",
644         "John Thompson",
645         "Qi Lu",
646         "Oscar Wilde",
647         "Satya"
648     ],
649     "organizations": [
650         "Microsoft",
651         "CEO",
652         "the Board",
653         "Nokia devices and services",
654         "Nokia",
655         "We",
656         "Next"
657     ],
658     "locations": []
659 }

```

3. Pour voir comment tirer parti de ces champs, ajoutez un paramètre de facette pour retourner une agrégation de documents correspondants par emplacement.

```
https://[YOUR-SERVICE-NAME].search.windows.net/indexes/cog-search-demo-idx/docs?
search=*&facet=locations&api-version=2020-06-30
```

Dans cet exemple, il existe 2 ou 3 correspondances pour chaque emplacement.

```

21 {
22     "count": 3,
23     "value": "UNITED STATES"
24 },
25 {
26     "count": 3,
27     "value": "Washington, D.C."
28 },
29 {
30     "count": 2,
31     "value": "Asia"
32 },
33 {
34     "count": 2,
35     "value": "JFK"
36 },
37 {
38     "count": 2,
39     "value": "ONE MICROSOFT WAY"
40 },
41 {
42     "count": 2,
43     "value": "REDMOND, WASHINGTON"
44 }

```

4. Dans ce dernier exemple, appliquez un filtre à la collection des organisations pour retourner deux correspondances pour les critères de filtre basés sur le NASDAQ.

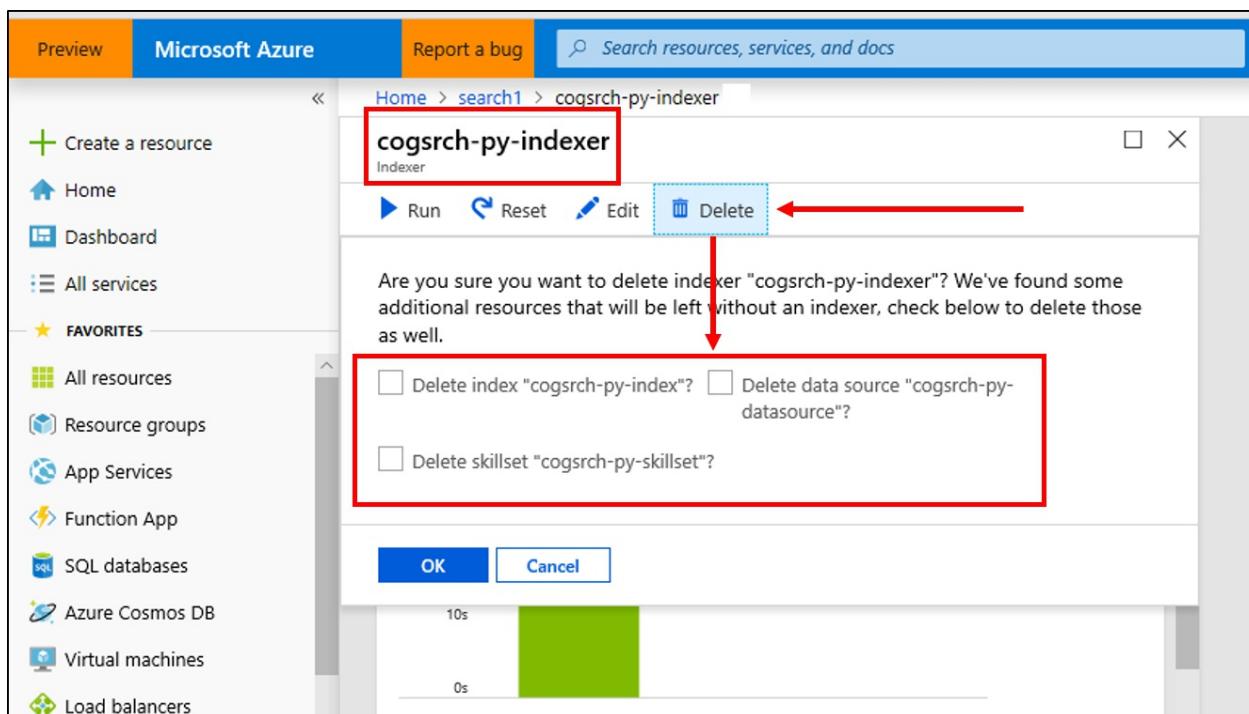
```
https://[YOUR-SERVICE-NAME].search.windows.net/indexes/cog-search-demo-idx/docs?  
search=*&$filter=organizations/any(organizations: organizations eq  
'NASDAQ')&$select=metadata_storage_name,organizations&$count=true&api-version=2020-06-30
```

Ces requêtes illustrent quelques-unes des façons dont vous pouvez travailler avec la syntaxe de requête et les filtres sur les nouveaux champs créés par Recherche cognitive. Pour obtenir plus d'exemples de requêtes, consultez [Exemples dans l'API REST de recherche de documents](#), [Exemples de requêtes de syntaxe simple](#) et [Exemples complets de requêtes Lucene](#).

## Réinitialiser et réexécuter

Dans les premières étapes expérimentales du développement, l'approche la plus pratique pour les itérations de conception consiste à supprimer les objets de Recherche cognitive Azure et à autoriser votre code à les reconstruire. Les noms des ressources sont uniques. La suppression d'un objet vous permet de le recréer en utilisant le même nom.

Vous pouvez utiliser le portail pour supprimer les index, les indexeurs et les ensembles de compétences. Quand vous supprimez l'indexeur, vous pouvez si vous le souhaitez, sélectivement supprimer l'index, l'ensemble de compétences et la source de données en même temps.



Vous pouvez aussi utiliser **DELETE** et fournir des URL vers chaque objet. La commande suivante supprime un indexeur.

```
DELETE https://[YOUR-SERVICE-NAME].search.windows.net/indexers/cog-search-demo-idxr?api-version=2020-06-30
```

Le code d'état 204 est retourné lorsque la suppression réussit.

## Éléments importants à retenir

Ce tutoriel présente les étapes de base pour générer un pipeline d'indexation enrichie via la création de composants : une source de données, un ensemble de compétences, un index et un indexeur.

Les [compétences intégrées](#) ont été introduites, ainsi que la définition d'un ensemble de compétences et les mécanismes de chaînage de compétences via des entrées et des sorties. Vous avez également appris que

`outputFieldMappings` est requis dans la définition de l'indexeur pour acheminer les valeurs enrichies du pipeline dans un index de recherche, sur un service de Recherche cognitive Azure.

Enfin, vous avez appris à tester les résultats et réinitialiser le système pour des itérations ultérieures. Vous avez appris qu'émettre des requêtes par rapport à l'index retourne la sortie créée par le pipeline d'indexation enrichie.

## Nettoyer les ressources

Lorsque vous travaillez dans votre propre abonnement, il est judicieux à la fin d'un projet de supprimer les ressources dont vous n'avez plus besoin. Les ressources laissées en cours d'exécution peuvent vous coûter de l'argent. Vous pouvez supprimer les ressources une par une, ou choisir de supprimer le groupe de ressources afin de supprimer l'ensemble des ressources.

Vous pouvez rechercher et gérer les ressources dans le portail à l'aide des liens Toutes les ressources ou Groupes de ressources situés dans le volet de navigation de gauche.

## Étapes suivantes

Maintenant que vous êtes familiarisé avec tous les objets d'un pipeline d'enrichissement par IA, examinons de plus près les définitions des ensembles de compétences et les compétences individuelles.

[Guide pratique pour créer un ensemble de compétences](#)

# Tutoriel : Utiliser Python et l'IA pour générer du contenu pouvant faire l'objet de recherches à partir d'objets blob Azure

04/10/2020 • 31 minutes to read • [Edit Online](#)

Si vous avez du texte non structuré ou des images dans Stockage Blob Azure, un [pipeline d'enrichissement par IA](#) peut extraire des informations et créer du contenu utile pour les scénarios de recherche en texte intégral ou d'exploration de connaissances. Bien qu'un pipeline puisse traiter des images, ce tutoriel Python se concentre sur du texte, en appliquant la détection de la langue et le traitement en langage naturel pour créer des champs exploitables dans des requêtes, des facettes et des filtres.

Ce tutoriel utilise Python et les [API REST de Recherche](#) pour effectuer les tâches suivantes :

- Commencez avec des documents entiers (texte non structuré), comme des documents PDF, HTML, DOCX et PPTX, dans Stockage Blob Azure.
- Définissez un pipeline qui extrait du texte, détecte la langue, reconnaît les entités et détecte les expressions clés.
- Définissez un index pour stocker la sortie (contenu brut et paires nom-valeur générées par le pipeline).
- Exécutez le pipeline pour démarrer des transformations et une analyse, ainsi que pour créer et charger l'index.
- Explorez les résultats à l'aide de la recherche en texte intégral et d'une syntaxe de requête enrichie.

Si vous n'avez pas d'abonnement Azure, ouvrez un [compte gratuit](#) avant de commencer.

## Prérequis

- [Stockage Azure](#)
- [Anaconda 3.7](#)
- [Créer ou rechercher un service de recherche existant](#)

### NOTE

Vous pouvez utiliser le service gratuit pour ce tutoriel. Avec un service de recherche gratuit, vous êtes limité à trois index, trois indexeurs et trois sources de données. Ce didacticiel crée une occurrence de chaque élément. Avant de commencer, veillez à disposer de l'espace suffisant sur votre service pour accepter les nouvelles ressources.

## Télécharger les fichiers

1. Ouvrez ce [dossier OneDrive](#) et en haut à gauche, cliquez sur **Télécharger** pour copier les fichiers sur votre ordinateur.
2. Cliquez avec le bouton droit sur le fichier zip et sélectionnez **Tout extraire**. Il y a 14 fichiers de différents types. Vous en utiliserez 7 dans le cadre de cet exercice.

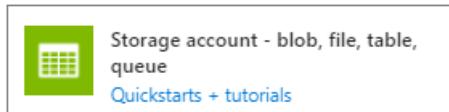
## 1 - Créer les services

Ce tutoriel utilise Recherche cognitive Azure pour l'indexation et les requêtes, Cognitive Services pour l'enrichissement par IA sur le back-end, et Stockage Blob Azure pour fournir les données. Ce tutoriel reste sous l'allocation gratuite de 20 transactions par indexeur par jour sur Cognitive Services : les seuls services que vous devez créer sont donc la recherche et le stockage.

Si possible, créez les deux services dans la même région et le même groupe de ressources pour des raisons de proximité et de facilité de gestion. Dans la pratique, votre compte de stockage Azure peut être dans une région quelconque.

## Démarrer avec le stockage Azure

1. [Connectez-vous au portail Azure](#) et cliquez sur **+ Créer une ressource**.
2. Recherchez un *compte de stockage* et sélectionnez l'offre de compte de stockage Microsoft.



3. Sous l'onglet **Bases**, les éléments suivants sont obligatoires. Acceptez les valeurs par défaut pour tout le reste.
  - **Groupe de ressources.** Sélectionnez un groupe existant ou créez-en un, mais utilisez le même groupe pour tous les services afin de pouvoir les gérer collectivement.
  - **Nom du compte de stockage.** Si vous pensez que vous pouvez avoir plusieurs ressources du même type, utilisez le nom pour lever l'ambiguïté par type et par région, par exemple *blobstoragewestus*.
  - **Emplacement.** Si possible, choisissez le même emplacement que celui utilisé pour Recherche cognitive Azure et Cognitive Services. Un emplacement unique annule les frais liés à la bande passante.
  - **Type de compte.** Choisissez la valeur par défaut, *StorageV2 (v2 universel)* .
4. Cliquez sur **Vérifier + créer** pour créer le service.
5. Une fois qu'il est créé, cliquez sur **Accéder à la ressource** pour ouvrir la page Vue d'ensemble.
6. Cliquez sur le service **Objets blob**.
7. Cliquez sur **+ Conteneur** pour créer un conteneur et nommez-le *cog-search-demo*.
8. Sélectionnez *cog-search-demo*, puis cliquez sur **Charger** pour ouvrir le dossier dans lequel vous avez enregistré les fichiers téléchargés. Sélectionnez tous les fichiers autres que des images. Vous devez disposer de 7 fichiers. Cliquez sur **OK** pour effectuer le chargement.

<input type="checkbox"/> Name	Date modified	Type	Size
<input checked="" type="checkbox"/> Cognitive Services and Bots (spanish)....	8/22/2019 10:51 PM	Adobe Acrobat D...	3,884 KB
<input checked="" type="checkbox"/> MSFT_cloud_architecture_contoso.pdf	8/22/2019 10:51 PM	Adobe Acrobat D...	1,805 KB
<input checked="" type="checkbox"/> NYSE_LNKD_2015.PDF	8/22/2019 10:51 PM	Adobe Acrobat D...	394 KB
<input checked="" type="checkbox"/> 10-K-FY16.html	8/22/2019 10:51 PM	HTML File	1,835 KB
<input type="checkbox"/> guthrie.jpg	8/22/2019 10:51 PM	JPG File	46 KB
<input type="checkbox"/> redshirt.jpg	8/22/2019 10:51 PM	JPG File	67 KB
<input type="checkbox"/> satyanadellalinux.jpg	8/22/2019 10:51 PM	JPG File	108 KB
<input checked="" type="checkbox"/> Cognitive Searvices and Content Intelli...	8/22/2019 10:51 PM	Microsoft PowerPo...	11,231 KB
<input checked="" type="checkbox"/> MSFT_FY17_10K.docx	8/22/2019 10:51 PM	Microsoft Word D...	675 KB
<input type="checkbox"/> 5074.clip_image002_6FE27E85.png	8/22/2019 10:51 PM	PNG File	472 KB
<input type="checkbox"/> create-search-collect-info.png	8/22/2019 10:51 PM	PNG File	57 KB
<input type="checkbox"/> create-search-service.png	8/22/2019 10:51 PM	PNG File	26 KB
<input type="checkbox"/> create-service-full-portal.png	8/22/2019 10:51 PM	PNG File	67 KB
<input checked="" type="checkbox"/> satyasletter.txt	8/22/2019 10:51 PM	Text Document	6 KB

9. Avant de quitter Stockage Azure, obtenez une chaîne de connexion afin de pouvoir formuler une connexion dans Recherche cognitive Azure.

- a. Revenez à la page Vue d'ensemble de votre compte de stockage (nous avons utilisé *blobstragewestus* comme exemple).
- b. Dans le volet de navigation gauche, sélectionnez **Clés d'accès** et copiez l'une des chaînes de connexion.

La chaîne de connexion est une URL similaire à l'exemple suivant :

```
DefaultEndpointsProtocol=https;AccountName=<storageaccountname>;AccountKey=<your account key>;EndpointSuffix=core.windows.net
```

10. Enregistrez la chaîne de connexion dans le Bloc-Notes. Vous en aurez besoin plus tard lors de la configuration de la connexion à la source de données.

## Cognitive Services

L'enrichissement par IA s'appuie sur Cognitive Services, notamment Analyse de texte et Vision par ordinateur pour le traitement des images et du langage naturel. Si votre objectif était de réaliser un prototype ou un projet réel, vous devriez à ce stade provisionner Cognitive Services (dans la même région que Recherche cognitive Azure) afin de pouvoir l'attacher aux opérations d'indexation.

Étant donné que ce tutoriel n'utilise que 7 transactions, vous pouvez ignorer le provisionnement des ressources, car Recherche cognitive Azure peut se connecter à Cognitive Services pour 20 transactions gratuites par exécution de l'indexeur. L'allocation gratuite est suffisante. Pour les projets de plus grande envergure, prévoyez de provisionner Cognitive Services au niveau S0 du paiement à l'utilisation. Pour plus d'informations, consultez [Attacher Cognitive Services](#).

## Recherche cognitive Azure

Le troisième composant est Recherche cognitive Azure, que vous pouvez [créer dans le portail](#). Vous pouvez utiliser le niveau Gratuit pour effectuer cette procédure pas à pas.

Comme avec le stockage Blob Azure, prenez un moment pour collecter la clé d'accès. Par ailleurs, lorsque vous commencez à structurer les demandes, vous devez fournir le point de terminaison et la clé API d'administration

utilisés pour authentifier chaque demande.

## Obtenir une clé API d'administration et une URL pour Recherche cognitive Azure

1. [Connectez-vous au Portail Azure](#), puis dans la page **Vue d'ensemble** du service de recherche, récupérez le nom de votre service de recherche. Vous pouvez confirmer le nom de votre service en passant en revue l'URL du point de terminaison. Si votre URL de point de terminaison est `https://mydemo.search.windows.net`, le nom du service doit être `mydemo`.
2. Dans **Paramètres > Clés**, obtenez une clé d'administration pour avoir des droits d'accès complets sur le service. Il existe deux clés d'administration interchangeables, fournies pour assurer la continuité de l'activité au cas où vous deviez en remplacer une. Vous pouvez utiliser la clé primaire ou secondaire sur les demandes d'ajout, de modification et de suppression d'objets.

Obtenez aussi la clé de requête. Il est recommandé d'émettre des demandes de requête avec un accès en lecture seule.

Une clé API est nécessaire dans l'en-tête de chaque requête envoyée à votre service. Une clé valide permet d'établir, en fonction de chaque demande, une relation de confiance entre l'application qui envoie la demande et le service qui en assure le traitement.

## 2 - Démarrer un notebook

Créez le bloc-notes en suivant les instructions ci-dessous ou téléchargez un notebook terminé à partir du [dépôt Azure-Search-python-samples](#).

Utilisez Anaconda Navigator pour lancer Jupyter Notebook et créer un notebook Python 3.

Dans le notebook, exécutez ce script pour charger les bibliothèques permettant d'exploiter JSON et de formuler des requêtes HTTP.

```
import json
import requests
from pprint import pprint
```

Dans le même notebook, définissez les noms de la source de données, de l'index, de l'indexeur et de l'ensemble de compétences. Exécutez ce script afin de configurer les noms pour ce tutoriel.

```
# Define the names for the data source, skillset, index and indexer
datasource_name = "cogsrch-py-datasource"
skillset_name = "cogsrch-py-skillset"
index_name = "cogsrch-py-index"
indexer_name = "cogsrch-py-indexer"
```

Dans le script suivant, remplacez les espaces réservés par vos service de recherche (YOUR-SEARCH-SERVICE-NAME) et clé API d'administration (YOUR-ADMIN-API-KEY), puis exécutez-le pour configurer le point de terminaison de service de la recherche.

```
# Setup the endpoint
endpoint = 'https://<YOUR-SEARCH-SERVICE-NAME>.search.windows.net/'
headers = {'Content-Type': 'application/json',
           'api-key': '<YOUR-ADMIN-API-KEY>'}
params = {
           'api-version': '2020-06-30'
}
```

## 3 - Créer le pipeline

Dans Recherche cognitive Azure, le traitement de l'IA se produit pendant l'indexation (ou l'ingestion de données). Cette partie de la procédure pas à pas crée quatre objets : source de données, définition d'index, ensemble de compétences, indexeur.

### Étape 1: Création d'une source de données

Un [objet Source de données](#) fournit la chaîne de connexion au conteneur d'objets blob contenant les fichiers.

Dans le script suivant, remplacez l'espace réservé YOUR-BLOB-RESOURCE-CONNECTION-STRING par la chaîne de connexion de l'objet blob que vous avez créé à l'étape précédente. Remplacez le texte de l'espace réservé pour le conteneur. Ensuite, exécutez le script pour créer une source de données nommée `cogsrch-py-datasource`.

```
# Create a data source
datasourceConnectionString = "<YOUR-BLOB-RESOURCE-CONNECTION-STRING>"
datasource_payload = {
    "name": datasource_name,
    "description": "Demo files to demonstrate cognitive search capabilities.",
    "type": "azureblob",
    "credentials": {
        "connectionString": datasourceConnectionString
    },
    "container": {
        "name": "<YOUR-BLOB-CONTAINER-NAME>"
    }
}
r = requests.put(endpoint + "/datasources/" + datasource_name,
                  data=json.dumps(datasource_payload), headers=headers, params=params)
print(r.status_code)
```

La demande doit retourner un code d'état 201 confirmant la réussite.

Dans le portail Azure, dans la page de tableau de bord du service de recherche, vérifiez que cogsrch-py-datasource apparaît dans la liste **Sources de données**. Cliquez sur **Actualiser** pour mettre à jour la page.

TYPE ↑↓	NAME	↑↓ TABLE/COLLECTION
	cogsrch-py-datasource	basic-demo-data-pr
	hotels-datasource	hotels
	realestate-us-sample	Listings 5K KingCounty WA

## Étape 2 : Créer un ensemble de compétences

Dans cette étape, vous allez définir un ensemble d'étapes d'enrichissement à appliquer à vos données. Vous appelez chaque étape d'enrichissement une *compétence* et l'ensemble des étapes d'enrichissement un *ensemble de compétences*. Ce tutoriel utilise des [compétences cognitives prédéfinies](#) pour l'ensemble de compétences :

- [Reconnaissance d'entité](#) pour extraire les noms d'organisations du contenu dans le conteneur d'objets blob.
- [Détection de la langue](#) pour identifier la langue du contenu.
- [Fractionnement de texte](#) pour découper un grand contenu en plus petits morceaux avant d'appeler la compétence d'extraction de phrases clés. L'extraction de phrases clés accepte des entrées de 50 000 caractères au maximum. Certains fichiers d'exemple doivent être fractionnés pour satisfaire cette limite.
- [Extraction de phrases clés](#) pour extraire les principales expressions clés.

Utilisez le script suivant pour créer un ensemble de compétences appelé `cogsrch-py-skillset`.

```
# Create a skillset
skillset_payload = {
    "name": skillset_name,
    "description":
        "Extract entities, detect language and extract key-phrases",
    "skills":
    [
        {
            "@odata.type": "#Microsoft.Skills.Text.EntityRecognitionSkill",
            "categories": ["Organization"],
            "defaultLanguageCode": "en",
            "inputs": [
                {
                    "name": "text",
                    "source": "/document/content"
                }
            ],
            "outputs": [
                {
                    "name": "organizations",
                    "targetName": "organizations"
                }
            ]
        },
        {
            "@odata.type": "#Microsoft.Skills.Text.LanguageDetectionSkill",
            "inputs": [
                {
                    "name": "text",
                    "source": "/document/content"
                }
            ]
        }
    ]
}
```

```

        ],
        "outputs": [
            {
                "name": "languageCode",
                "targetName": "languageCode"
            }
        ]
    },
    {
        "@odata.type": "#Microsoft.Skills.Text.SplitSkill",
        "textSplitMode": "pages",
        "maximumPageLength": 4000,
        "inputs": [
            {
                "name": "text",
                "source": "/document/content"
            },
            {
                "name": "languageCode",
                "source": "/document/languageCode"
            }
        ],
        "outputs": [
            {
                "name": "textItems",
                "targetName": "pages"
            }
        ]
    },
    {
        "@odata.type": "#Microsoft.Skills.Text.KeyPhraseExtractionSkill",
        "context": "/document/pages/*",
        "inputs": [
            {
                "name": "text",
                "source": "/document/pages/*"
            },
            {
                "name": "languageCode",
                "source": "/document/languageCode"
            }
        ],
        "outputs": [
            {
                "name": "keyPhrases",
                "targetName": "keyPhrases"
            }
        ]
    }
]

}

r = requests.put(endpoint + "/skillsets/" + skillset_name,
                  data=json.dumps(skillset_payload), headers=headers, params=params)
print(r.status_code)

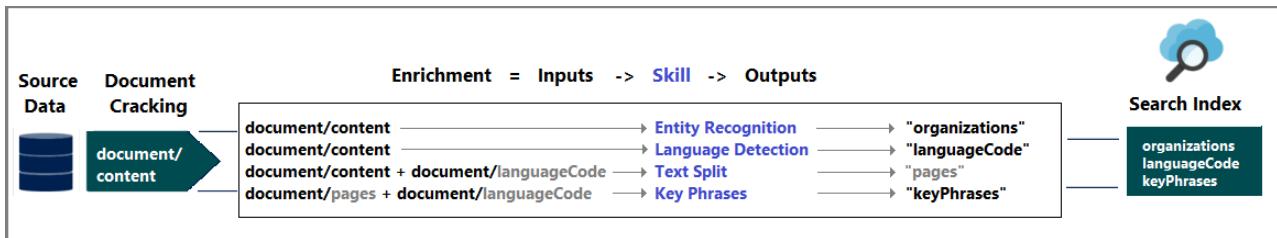
```

La demande doit retourner un code d'état 201 confirmant la réussite.

La compétence d'extraction de phrases clés est appliquée pour chaque page. En définissant le contexte sur `"document/pages/*"`, vous exécutez cette enrichisseur pour chaque membre du document/tableau de pages (pour chaque page dans le document).

Chaque compétence s'exécute sur le contenu du document. Au cours du traitement, Recherche cognitive Azure ouvre chaque document pour lire le contenu de différents formats de fichiers. Le texte trouvé dans le fichier source est placé dans un champ `content`, un pour chaque document. Ainsi, définissez l'entrée en tant que `/document/content`.

Une représentation graphique de l'ensemble de compétences est présentée ci-dessous.



Les sorties peuvent être mappées à un index, utilisées comme entrée d'une compétence en aval, ou les deux, comme c'est le cas avec le code de langue. Dans l'index, un code de langue est utile pour le filtrage. En tant qu'entrée, le code de langue est utilisé par les compétences d'analyse de texte pour informer les règles linguistiques en matière de césure de mots.

Pour plus d'informations sur les principes de base des ensembles de compétences, consultez [Guide pratique pour définir un ensemble de compétences](#).

### Étape 3 : Création d'un index

Dans cette section, vous définissez le schéma d'index en spécifiant les champs à inclure dans l'index de recherche et en définissant les attributs de recherche pour chaque champ. Les champs ont un type et peuvent prendre des attributs qui déterminent la façon dont le champ est utilisé (pour la recherche, le tri, etc.). Les noms des champs dans un index ne sont pas tenus de correspondre exactement aux noms des champs dans la source. Dans une étape ultérieure, vous ajoutez des mappages de champs dans un indexeur pour connecter les champs sources et de destination. Pour cette étape, définissez l'index à l'aide de conventions d'affectation de noms de champs appropriées à votre application de recherche.

Cet exercice utilise les champs et les types de champ suivants :

NOMS DE CHAMPS :	ID	CONTENT	LANGUAGECODE	KEYPHRASES	ORGANIZATIONS
Types de champ :	Edm.String	Edm.String	Edm.String	List<Edm.String>	List<Edm.String>

Exécutez ce script pour créer l'index nommé `cogsrch-py-index`.

```

# Create an index
index_payload = {
    "name": index_name,
    "fields": [
        {
            "name": "id",
            "type": "Edm.String",
            "key": "true",
            "searchable": "true",
            "filterable": "false",
            "facetable": "false",
            "sortable": "true"
        },
        {
            "name": "content",
            "type": "Edm.String",
            "sortable": "false",
            "searchable": "true",
            "filterable": "false",
            "facetable": "false"
        },
        {
            "name": "languageCode",
            "type": "Edm.String",
            "searchable": "true",
            "filterable": "false",
            "facetable": "false"
        },
        {
            "name": "keyPhrases",
            "type": "Collection(Edm.String)",
            "searchable": "true",
            "filterable": "false",
            "facetable": "false"
        },
        {
            "name": "organizations",
            "type": "Collection(Edm.String)",
            "searchable": "true",
            "sortable": "false",
            "filterable": "false",
            "facetable": "false"
        }
    ]
}

r = requests.put(endpoint + "/indexes/" + index_name,
                 data=json.dumps(index_payload), headers=headers, params=params)
print(r.status_code)

```

La demande doit retourner un code d'état 201 confirmant la réussite.

Pour en savoir plus sur la définition d'un index, consultez [Créer un index \(API REST de la Recherche cognitive Azure\)](#).

#### Étape 4 : Créer et exécuter un indexeur

Un [indexeur](#) pilote le pipeline. Les trois composants que vous avez créés jusqu'à présent (source de données, ensemble de compétences, index) sont les entrées d'un indexeur. La création de l'indexeur sur Recherche cognitive Azure est l'événement qui met en mouvement la totalité du pipeline.

Pour lier ces objets en un indexeur, vous devez définir des mappages de champs.

- Les `"fieldMappings"` sont traités avant l'ensemble de compétences, en mappant les champs sources de la source de données sur des champs cibles dans un index. Si les noms et types de champ sont identiques aux

deux extrémités, aucun mappage n'est nécessaire.

- Les `"outputFieldMappings"` sont traités après l'ensemble de compétences, en référençant les `"sourceFieldNames"` qui n'existent pas tant que le craquage de document ou l'enrichissement ne les ont pas créés. `"targetFieldName"` est un champ dans un index.

En plus de raccrocher des entrées à des sorties, vous pouvez également utiliser les mappages de champs pour aplatiser les structures de données. Pour plus d'informations, consultez [Guide pratique pour mapper des champs enrichis sur un index pouvant faire l'objet d'une recherche](#).

Exécutez ce script pour créer un indexeur nommé `cogsrch-py-indexer`.

```
# Create an indexer
indexer_payload = {
    "name": indexer_name,
    "dataSourceName": datasource_name,
    "targetIndexName": index_name,
    "skillsetName": skillset_name,
    "fieldMappings": [
        {
            "sourceFieldName": "metadata_storage_path",
            "targetFieldName": "id",
            "mappingFunction":
                {"name": "base64Encode"}
        },
        {
            "sourceFieldName": "content",
            "targetFieldName": "content"
        }
    ],
    "outputFieldMappings":
    [
        {
            "sourceFieldName": "/document/organizations",
            "targetFieldName": "organizations"
        },
        {
            "sourceFieldName": "/document/pages/*/keyPhrases/*",
            "targetFieldName": "keyPhrases"
        },
        {
            "sourceFieldName": "/document/languageCode",
            "targetFieldName": "languageCode"
        }
    ],
    "parameters":
    {
        "maxFailedItems": -1,
        "maxFailedItemsPerBatch": -1,
        "configuration":
        {
            "dataToExtract": "contentAndMetadata",
            "imageAction": "generateNormalizedImages"
        }
    }
}

r = requests.put(endpoint + "/indexers/" + indexer_name,
                  data=json.dumps(indexer_payload), headers=headers, params=params)
print(r.status_code)
```

La demande doit retourner sous peu un code d'état 201, bien que le traitement puisse cependant durer plusieurs minutes. Bien que le jeu de données soit petit, les compétences analytiques, telles que l'analyse d'image, sont gourmandes en ressources de calcul et en temps.

Vous pouvez [superviser l'état de l'indexeur](#) pour déterminer à quel moment il est en cours d'exécution ou terminé.

#### TIP

La création d'un indexeur appelle le pipeline. Tout problème lié à l'accès aux données, au mappage des entrées et des sorties ou à l'ordre des opérations apparaît à ce stade. Pour réexécuter le pipeline avec des modifications de code ou de script, vous pouvez être amené à supprimer d'abord des objets. Pour plus d'informations, consultez [Réinitialiser et réexécuter](#).

#### À propos du corps de la demande

Le script affecte à `"maxFailedItems"` la valeur -1, ce qui indique au moteur d'indexation d'ignorer les erreurs au cours de l'importation des données. Cela est utile car très peu de documents figurent dans la source de données de démonstration. Pour une source de données plus volumineuse, vous définiriez une valeur supérieure à 0.

Notez également l'instruction `"dataToExtract": "contentAndMetadata"` dans les paramètres de configuration. Cette instruction indique à l'indexeur d'extraire le contenu de fichiers de différents formats et les métadonnées associées à chaque fichier.

Lorsque le contenu est extrait, vous pouvez définir `imageAction` pour extraire le texte des images trouvées dans la source de données. La configuration `"imageAction": "generateNormalizedImages"`, associée à la compétence de reconnaissance optique des caractères et à la compétence de fusion de texte, indique à l'indexeur d'extraire le texte des images (par exemple, le mot « stop » d'un panneau de signalisation Stop) et de l'incorporer dans le champ de contenu. Ce comportement s'applique aux images intégrées dans les documents (pensez à une image dans un fichier PDF) et aux images trouvées dans la source de données, par exemple un fichier JPG.

## 4 - Surveiller l'indexation

Une fois l'indexeur défini, il s'exécute automatiquement lorsque vous envoyez la demande. Selon les compétences cognitives que vous avez définies, l'indexation peut prendre plus de temps que prévu. Pour savoir si le traitement de l'indexeur est terminé, exécutez le script suivant.

```
# Get indexer status
r = requests.get(endpoint + "/indexers/" + indexer_name +
                  "/status", headers=headers, params=params)
pprint(json.dumps(r.json(), indent=1))
```

Dans la réponse, supervisez `"lastResult"` pour ses valeurs `"status"` et `"endTime"`. Exécuter le script régulièrement pour vérifier l'état. Quand l'indexeur a terminé, l'état est défini sur « success », une valeur « endTime » est spécifiée et la réponse inclut les erreurs et avertissements qui se sont éventuellement produits au cours de l'enrichissement.

```
#Determine if the indexer is still running
r = requests.get(endpoint + "/indexers/" + indexer_name + "/status", headers=headers, params=params)
pprint(json.dumps(r.json(), indent=1))

'{"\n  "@odata.context": '\n    "https://mysearchsvc.search.windows.net/$metadata#Microsoft.Azure.Search.V2019_05_06.IndexerExecutionInfo",\n  '\n  "@name": "cogsrch-py-indexer",\n  '\n  "status": "running",\n  '\n  "lastResult": {\n    '\n      "status": "success",\n      '\n      "errorMessage": null,\n      '\n      "startTime": "2019-06-13T15:15:14.078Z",\n      '\n      "endTime": "2019-06-13T15:15:47.66Z",\n      '\n      "itemsProcessed": 14,\n      '\n      "itemsFailed": 0,\n      '\n      "initialTrackingState": "{\\r\\n  \"lastFullEnumerationStartTime\": '\n        \"2001-01-01T00:00:00Z\",\\r\\n  \"lastAttemptedEnumerationStartTime\": '\n        \"2001-01-01T00:00:00Z\",\\r\\n  \"nameHighWaterMark\": null\\r\\n}\",\n      '\n      "finalTrackingState": '\n        \"LastFullEnumerationStartTime\": \"2019-06-13T15:14:44.0784124+00:00\",\\\"LastAttemptedEnumerationStartTime\\\": \"201\n9-06-13T15:14:44.0784124+00:00\",\\\"NameHighWaterMark\\\":null}\",\n      '\n      "errors": [],\n      '\n  }\n}'
```

Les avertissements sont courants avec certaines combinaisons de fichiers sources et de compétences et n'indiquent pas toujours un problème. De nombreux avertissements ont une importance mineure. Par exemple, si vous indexez un fichier JPEG qui n'a pas de texte, vous voyez l'avertissement dans la capture d'écran ci-après.

```
#Determine if the indexer is still running
r = requests.get(endpoint + "/indexers/" + indexer_name + "/status", headers=headers, params=params)
pprint(json.dumps(r.json(), indent=1))
  https://mysearchsvc.blob.core.windows.net/basic-demo-data-pr/NYSE_LNKD_2015.PDF",\n
  '    "message": "Truncated extracted text to 65536 characters.\n'
  '  },\n'
  '  {\n'
  '    "key": '
"https://mysearchsvc.blob.core.windows.net/basic-demo-data-pr/NYSE_LNKD_2015.PDF",\n
  '    "message": "Truncated extracted text to 65536 characters.\n'
  '  },\n'
  '  {\n'
  '    "key": '
"https://mysearchsvc.blob.core.windows.net/basic-demo-data-pr/create-service-full-portal.png",\n
  '    "message": "Skill #1: Input text is missing. No entities will be '
'returned\r\nSkill #2: Input text is missing. No language will be '
'returned\r\\nSkill #3: Input text is missing\r\\n"\n'
  '},\n'
  '  {\n'
  '    "key": '
"https://mysearchsvc.blob.core.windows.net/basic-demo-data-pr/create-search-service.png",\n
  '    "message": "Skill #1: Input text is missing. No entities will be '
'returned\r\\nSkill #2: Input text is missing. No language will be '
```

## 5 - Recherche

Une fois l'indexation terminée, exécutez des requêtes qui retournent le contenu de champs individuels. Par défaut, le service Recherche cognitive Azure retourne les 50 meilleurs résultats. Les données d'exemple sont petites si bien que la configuration par défaut fonctionne correctement. Toutefois, lorsque vous travaillez avec des jeux de données plus volumineux, vous pouvez être amené à inclure des paramètres dans la chaîne de requête pour retourner plus de résultats. Pour obtenir des instructions, consultez [Guide pratique pour présenter les résultats dans la Recherche cognitive Azure](#).

En guise d'étape de vérification, récupérez la définition de l'index montrant tous les champs.

```
# Query the service for the index definition
r = requests.get(endpoint + "/indexes/" + index_name,
                  headers=headers, params=params)
pprint(json.dumps(r.json(), indent=1))
```

Les résultats doivent ressembler à l'exemple suivant. La capture d'écran montre uniquement une partie de la réponse.

jupyter PythonTutorial-AzureSearch-AIEnrichment

File Edit View Insert Cell Kernel Widgets Help

In [13]: #Query the index for all fields  
r = requests.get(endpoint + "/indexes/" + index\_name, headers=headers, params=params)  
print(json.dumps(r.json(), indent=1))

```
{  
    "@odata.context": "https://mysearchsvc.search.windows.net/$metadata#indexes/$entity",  
    "@odata.etag": "\\"0x8D6EE3652EC9078\\\"",  
    "name": "cogsrch-py-index",  
    "defaultScoringProfile": null,  
    "fields": [  
        {  
            "name": "id",  
            "type": "Edm.String",  
            "searchable": true,  
            "filterable": false,  
            "retrievable": true,  
            "sortable": true,  
            "facetable": false,  
            "key": true,  
            "indexAnalyzer": null,  
            "searchAnalyzer": null,  
            "analyzer": null,  
            "synonymMaps": []  
        },  
        {  
            "name": "content",  
            "type": "Edm.String",  
            "searchable": true,  
            "filterable": false,  
            "retrievable": true,  
            "sortable": true,  
            "facetable": false,  
            "key": true,  
            "indexAnalyzer": null,  
            "searchAnalyzer": null,  
            "analyzer": null,  
            "synonymMaps": []  
        }  
    ]  
}
```

La sortie est le schéma d'index, avec le nom, le type et les attributs de chaque champ.

Envoyez une deuxième requête pour `"*"` afin de retourner tout le contenu d'un champ individuel, tel que `organizations`.

```
# Query the index to return the contents of organizations  
r = requests.get(endpoint + "/indexes/" + index_name +  
    "/docs?&search=*&$select=organizations", headers=headers, params=params)  
pprint(json.dumps(r.json(), indent=1))
```

Les résultats doivent ressembler à l'exemple suivant. La capture d'écran montre uniquement une partie de la réponse.

```

In [14]: #Query the index to return the contents of organizations
#Note: Index creation may take time. If this step returns no data, wait a few minutes
# and then try again
r = requests.get(endpoint + "/indexes/" + index_name + "/docs?&search=*&$select=organizations", headers=headers, params=params)
pprint(r.json())

```

```

{'@odata.context': "https://mysearchsvc.search.windows.net/indexes('cogsrch-py-index')/$metadata#docs(*)",
 'value': [{"@search.score": 1.0,
            'organizations': ['Microsoft Cloud\n\nContoso',
                             'Microsoft',
                             'Microsoft s',
                             'Microsoft s',
                             'Microsoft Corporation',
                             'Microsoft \n\nCloud This',
                             'The Contoso Corporation\n\nContoso',
                             'Windows Server AD',
                             'Management\n\nManagement',
                             'Azure IaaS',
                             'Microsoft Cloud Identity for \n',
                             '\n',
                             'Enterprise Architects',
                             'Microsoft Cloud Networking \n',
                             '\n',
                             'for Enterprise Architects',
                             'Microsoft Cloud Security for \n',
                             '\n']}

```

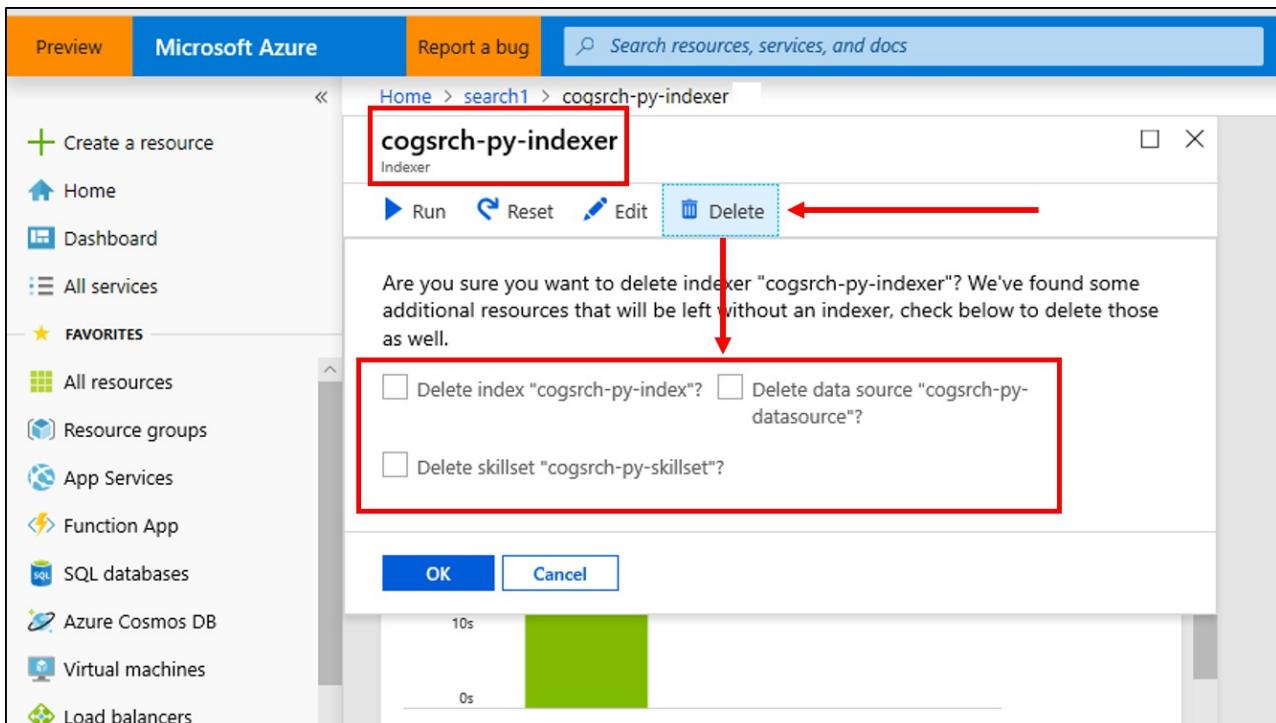
Répétez cette opération pour les champs supplémentaires : `content`, `languageCode`, `keyPhrases` et `organizations` dans cet exercice. Vous pouvez retourner plusieurs champs via `$select` à l'aide d'une liste délimitée par des virgules.

Vous pouvez utiliser GET ou POST, en fonction de la longueur et de la complexité de la chaîne de requête. Pour plus d'informations, consultez [Exécuter des requêtes à l'aide de l'API REST](#).

## Réinitialiser et réexécuter

Dans les premières étapes expérimentales du développement, l'approche la plus pratique pour les itérations de conception consiste à supprimer les objets de Recherche cognitive Azure et à autoriser votre code à les reconstruire. Les noms des ressources sont uniques. La suppression d'un objet vous permet de le recréer en utilisant le même nom.

Vous pouvez utiliser le portail pour supprimer les index, les indexeurs et les ensembles de compétences. Quand vous supprimez l'indexeur, vous pouvez si vous le souhaitez, sélectivement supprimer l'index, l'ensemble de compétences et la source de données en même temps.



Vous pouvez également les supprimer à l'aide d'un script. Le script suivant montre comment supprimer un ensemble de compétences.

```
# delete the skillset
r = requests.delete(endpoint + "/skillsets/" + skillset_name,
                     headers=headers, params=params)
pprint(json.dumps(r.json(), indent=1))
```

Le code d'état 204 est retourné lorsque la suppression réussit.

## Éléments importants à retenir

Ce tutoriel présente les étapes de base pour générer un pipeline d'indexation enrichie via la création de composants : une source de données, un ensemble de compétences, un index et un indexeur.

Des **compétences intégrées** ont été introduites, ainsi que des définitions d'ensemble de compétences et un moyen de chaîner les compétences entre elles via des entrées et des sorties. Vous avez également appris que `outputFieldMappings` est requis dans la définition de l'indexeur pour acheminer les valeurs enrichies du pipeline dans un index de recherche, sur un service de Recherche cognitive Azure.

Enfin, vous avez appris à tester les résultats et réinitialiser le système pour des itérations ultérieures. Vous avez appris qu'émettre des requêtes par rapport à l'index retourne la sortie créée par le pipeline d'indexation enrichie. Dans cette version, il existe un mécanisme permettant d'afficher les constructions internes (documents enrichis créés par le système). Vous avez également appris à vérifier l'état de l'indexeur et quels objets supprimer avant de réexécuter un pipeline.

## Nettoyer les ressources

Lorsque vous travaillez dans votre propre abonnement, il est judicieux à la fin d'un projet de supprimer les ressources dont vous n'avez plus besoin. Les ressources laissées en cours d'exécution peuvent vous coûter de l'argent. Vous pouvez supprimer les ressources une par une, ou choisir de supprimer le groupe de ressources afin de supprimer l'ensemble des ressources.

Vous pouvez rechercher et gérer les ressources dans le portail à l'aide des liens Toutes les ressources ou Groupes de ressources situés dans le volet de navigation de gauche.

## Étapes suivantes

Maintenant que vous êtes familiarisé avec tous les objets d'un pipeline d'enrichissement par IA, examinons de plus près les définitions des ensembles de compétences et les compétences individuelles.

[Guide pratique pour créer un ensemble de compétences](#)

# Tutoriel : Diagnostiquer, réparer et valider les changements apportés à votre ensemble de compétences

04/10/2020 • 24 minutes to read • [Edit Online](#)

Dans cet article, vous allez utiliser le portail Azure pour accéder à des sessions de débogage afin de résoudre les problèmes liés à l'ensemble de compétences fourni. L'ensemble de compétences comporte des erreurs qui doivent être traitées. Ce tutoriel va vous guider tout au long d'une session de débogage pour identifier et résoudre les problèmes liés aux entrées et sorties de compétences.

## IMPORTANT

Sessions de débogage est une fonctionnalité en préversion fournie sans contrat de niveau de service ; elle n'est pas recommandée pour les charges de travail de production. Pour plus d'informations, consultez [Conditions d'Utilisation Supplémentaires relatives aux Évaluations Microsoft Azure](#).

Si vous n'avez pas d'abonnement Azure, créez un [compte gratuit](#) avant de commencer.

## Prérequis

- Un abonnement Azure. Créez un [compte gratuit](#) ou utilisez votre abonnement actuel
- Une instance du service Recherche cognitive Azure
- Un compte Azure Storage
- [Application de bureau Postman](#)

## Créer des services et charger des données

Ce tutoriel utilise la Recherche cognitive Azure et les services de Stockage Azure.

- [Téléchargez les exemples de données](#) qui se composent de 19 fichiers.
- [Créez un compte de stockage Azure](#) ou [recherchez un compte existant](#).

Choisissez la même région que celle de la Recherche cognitive Azure pour éviter des frais de bande passante.

Il doit être de type StorageV2 (V2 universel).

- Ouvrez les pages des services de stockage et créez un conteneur. Une bonne pratique consiste à spécifier le niveau d'accès «privé ». Nommez votre conteneur `clinicaltrialdataset`.
- Dans le conteneur, cliquez sur **Charger** pour charger les exemples de fichiers que vous avez téléchargés et décompressés au cours de la première étape.
- [Créez un service Recherche cognitive Azure](#) ou [recherchez un service existant](#). Vous pouvez utiliser un service gratuit pour ce guide de démarrage rapide.

## Obtenir une clé et une URL

Les appels REST requièrent l'URL du service et une clé d'accès et ce, sur chaque demande. Un service de recherche

est créé avec les deux. Ainsi, si vous avez ajouté la Recherche cognitive Azure à votre abonnement, effectuez ce qui suit pour obtenir les informations nécessaires :

1. [Connectez-vous au portail Azure](#), puis dans la page **Vue d'ensemble** du service de recherche, récupérez l'URL. Voici un exemple de point de terminaison : `https://mydemo.search.windows.net`.
2. Dans **Paramètres > Clés**, obtenez une clé d'administration pour avoir des droits d'accès complets sur le service. Il existe deux clés d'administration interchangeables, fournies pour assurer la continuité de l'activité au cas où vous deviez en remplacer une. Vous pouvez utiliser la clé primaire ou secondaire sur les demandes d'ajout, de modification et de suppression d'objets.

The screenshot displays two windows from the Azure portal. The top window is titled 'Tableau de bord > mydemo' and shows the 'Vue d'ensemble' blade. It includes a search bar, a toolbar with actions like 'Ajouter un index', 'Importer...', 'Explorateur...', 'Actualiser', 'Supprimer', and 'Déplacer'. Below the toolbar, it shows a 'Groupe de ressources' named 'demo-resource-group' and an 'URL' field containing 'https://mydemo.search.windows.net'. The bottom window is titled 'Tableau de bord > mydemo - Clés' and shows the 'Clés' blade. It has a search bar and two buttons: 'Régénérer la clé primaire' and 'Régénérer la clé secondaire'. It lists two keys: 'Clé d'administration primaire' and 'Clé d'administration secondaire', both represented by placeholder text fields: '<placeholder-for-alphanumeric-autogenerated-string>'. The left sidebar of the bottom window includes 'Vue d'ensemble', 'Journal d'activité', 'Contrôle d'accès (IAM)', 'Paramètres', 'Démarrage rapide', and 'Clés' (which is highlighted with a red box and circled with a red number 2). The URL 'https://mydemo.search.windows.net' is also highlighted with a red box and circled with a red number 1.

Toutes les demandes nécessitent une clé API sur chaque demande envoyée à votre service. L'utilisation d'une clé valide permet d'établir, en fonction de chaque demande, une relation de confiance entre l'application qui envoie la demande et le service qui en assure le traitement.

## Créer une source de données, un ensemble de compétences, un index et un indexeur

Dans cette section, Postman et une collection fournie sont utilisés pour créer la source de données, l'ensemble de compétences, l'index et l'indexeur du service de recherche.

1. Si vous ne disposez pas de Postman, vous pouvez [télécharger l'application de bureau Postman ici](#).
2. [Télécharger la collection Postman Sessions de débogage](#)
3. Démarrer Postman
4. Sous **Fichiers > Nouveau**, sélectionnez la collection à importer.
5. Une fois la collection importée, développez la liste des actions (...).
6. Cliquez sur **Modifier**.
7. Entrez le nom de votre searchService (par exemple, si le point de terminaison est `https://mydemo.search.windows.net`, le nom du service est « `mydemo` »).
8. Entrez la valeur apiKey avec la clé primaire ou secondaire de votre service de recherche.
9. Entrez la valeur storageConnectionString à partir de la page des clés de votre compte de Stockage Azure.
10. Entrez la valeur containerName pour le conteneur que vous avez créé dans le compte de stockage.

**EDIT COLLECTION**

Name  
DebugSessions

Description    Authorization    Pre-request Scripts    Tests    **Variables** ●

These variables are specific to this collection and its requests. [Learn more about collection variables.](#)

	VARIABLE	INITIAL VALUE ⓘ	CURRENT VALUE ⓘ	...	Persist All	Reset All
<input checked="" type="checkbox"/>	searchService					
<input checked="" type="checkbox"/>	apiKey					
<input checked="" type="checkbox"/>	storageConnectionString					
<input checked="" type="checkbox"/>	containerName					
Add a new variable						

ⓘ Use variables to reuse values in different places. The current value is used while sending a request and is never synced to Postman's servers. The initial value is auto-updated to reflect the current value. [Change this](#) behaviour from Settings. [Learn more about variable values](#)

Cancel    **Update**

La collection contient quatre appels REST différents utilisés pour terminer cette section.

Le premier appel crée la source de données, `clinical-trials-ds`. Le deuxième appel crée l'ensemble de compétences, `clinical-trials-ss`. Le troisième appel crée l'index, `clinical-trials`. Le quatrième et dernier appel crée l'indexeur, `clinical-trials-idxr`. Une fois tous les appels de la collection terminés, fermez Postman, puis revenez au portail Azure.

POST CreateDatasource

POST https://{{(searchService)}}.search.windows.net/datasources?api-version=2019-05-06-Preview

Headers (1)

KEY VALUE DESCRIPTION

api-key {{(apiKey)}

Body

Params Authorization Headers (1) Body Pre-request Script Tests Settings

## Vérification des résultats

L'ensemble de compétences contient quelques erreurs courantes. Dans cette section, l'exécution d'une requête vide pour retourner tous les documents affiche plusieurs erreurs. Dans les étapes suivantes, les problèmes sont résolus à l'aide d'une session de débogage.

1. Accédez à votre service de recherche dans le portail Azure.
2. Sélectionnez l'onglet **Index**.
3. Sélectionnez l'index `clinical-trials`.
4. Cliquez sur **Rechercher** pour exécuter une requête vide.

Une fois la recherche terminée, deux champs sans données listées, « organizations » et « locations », sont affichés dans la fenêtre. Suivez les étapes permettant de découvrir tous les problèmes produits par l'ensemble de compétences.

1. Retournez à la page Vue d'ensemble du service de recherche.
2. Sélectionnez l'onglet **Indexeurs**.
3. Cliquez sur `clinical-trials-idxr`, puis sélectionnez la notification d'avertissement.

Il existe de nombreux problèmes avec les mappages de champs de sortie de projection, et la page 3 comporte des avertissements car une ou plusieurs entrées de compétences ne sont pas valides.

Retournez à l'écran de vue d'ensemble du service de recherche.

## Démarrer votre session de débogage

**new-debug-session**

Debug session

Save Session Refresh Delete Run Commit changes... download

**Definition** AI Enrichments Errors/Warnings

Debug session name \* new-debug-session

Description (optional)

Storage connection string \* Choose an existing connection

Indexer \* clinical-trials-idxr  
Skillset: clinical-trials-ss Data Source: clinical-trials-ds

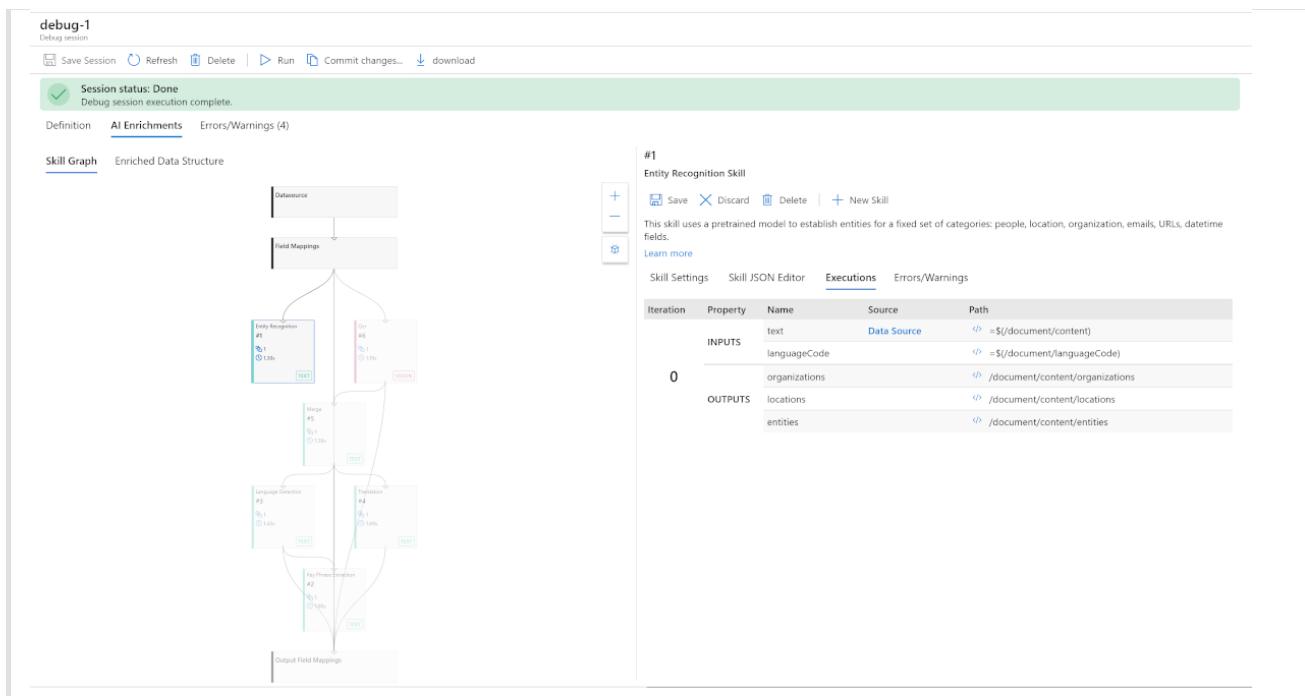
Datasource Indexer options

Document to debug Select first document Debug specific document

1. Cliquez sur l'onglet Sessions de débogage (préversion).
2. Sélectionnez +NewDebugSession.
3. Donnez un nom à la session.
4. Connectez la session à votre compte de stockage.
5. Indiquez le nom de l'indexeur. L'indexeur a des références à la source de données, à l'ensemble de compétences et à l'index.
6. Acceptez le choix de document par défaut pour le premier document de la collection.
7. **Enregistrez** la session. L'enregistrement de la session lance le pipeline d'enrichissement par IA tel que défini par l'ensemble de compétences.

### IMPORTANT

Une session de débogage ne fonctionne qu'avec un seul document. Un document spécifique du jeu de données peut être sélectionné. Si ce n'est pas le cas, la session est définie par défaut sur le premier document.



Quand l'exécution de la session de débogage est terminée, la session affiche par défaut l'onglet Enrichissements par IA, en mettant en évidence le graphe des compétences.

- Le graphe des compétences fournit une hiérarchie visuelle de l'ensemble de compétences et son ordre d'exécution de haut en bas. Les compétences qui sont côte à côte dans le graphe sont exécutées en parallèle. Le code couleur des compétences dans le graphe indique les types de compétences qui sont exécutées dans l'ensemble de compétences. Dans l'exemple, les compétences en vert sont du texte et la compétence en rouge est la vision. Si vous cliquez sur une compétence individuelle dans le graphe, les détails de cette instance de la compétence s'affichent dans le volet droit de la fenêtre de session. Les paramètres des compétences, un éditeur JSON, les détails de l'exécution et les erreurs/avertissemets sont tous disponibles pour que vous puissiez les consulter et les modifier.
- La structure de données enrichie détaille les nœuds de l'arborescence d'enrichissement générés par les compétences à partir du contenu du document source.

L'onglet Erreurs/avertissemets fournit une liste bien plus petite que celle affichée précédemment car cette liste détaille uniquement les erreurs d'un seul document. Comme avec la liste affichée par l'indexeur, vous pouvez cliquer sur un message d'avertissement pour voir les détails le concernant.

## Remédier à la valeur d'entrée de compétence manquante

Sous l'onglet Erreurs/avertissemets, une erreur est signalée pour une opération libellée

`Enrichment.NerSkillV2.#1`. Les détails de cette erreur expliquent qu'il y a un problème avec une valeur d'entrée de compétence « `/document/languageCode` ».

- Retournez à l'onglet Enrichissements par IA.
- Cliquez sur le **graphe des compétences**.
- Cliquez sur la compétence étiquetée #1 pour afficher ses détails dans le volet de droite.
- Recherchez l'entrée correspondant à « `languageCode` ».
- Selectionnez le symbole `</>` au début de la ligne, puis ouvrez l'évaluateur d'expression.
- Cliquez sur le bouton **Évaluer** pour vérifier que cette expression génère une erreur. Cette opération permet de vérifier que la propriété « `languageCode` » n'est pas une entrée valide.

The screenshot shows the AI Skills interface with a 'Skill Graph' tab selected. The graph illustrates the flow of data from a 'Datasource' through 'Field Mappings' to various 'Entity Recognition' and 'Language Detection' components. These components then feed into a 'Merge' node, which finally outputs to 'Text' and 'Text' fields. A tooltip for the 'languageCode' field indicates it is mapped from 'languageCode' in the 'document' source. To the right, an 'Expression evaluator' window is open, showing the expression '/document/languageCode'. The context is set to '/document/content'. The message 'The expression did not resolve to a valid input path.' is displayed, along with the JSON code for the skill's execution.

Il existe deux façons de rechercher cette erreur dans la session. La première consiste à examiner d'où provient l'entrée, c'est-à-dire quelle compétence de la hiérarchie est supposée produire ce résultat ? L'onglet Exécutions du volet des détails des compétences doit afficher la source de l'entrée. Si aucune source n'est affichée, cela indique une erreur de mappage de champ.

1. Cliquez sur l'onglet **Exécutions**.
2. Examinez les entrées (INPUTS) et recherchez « languageCode ». Aucune source n'est indiquée pour cette entrée.
3. Basculez vers le volet de gauche pour afficher la structure de données enrichie. Aucun chemin mappé ne correspond à « languageCode ».

The screenshot shows the AI Skills interface with the 'Enriched Data Structure' tab selected. It displays a tree view of the data path, starting from '/document/path'. The 'language' node under 'content' is highlighted with a red box. The 'Skill Executions' tab is selected, showing a table of skill executions. For iteration 0, there is one entry for the 'Entity Recognition Skill' (#1). The 'languageCode' input is mapped from the 'languageCode' source, while 'organizations', 'locations', and 'entities' are direct outputs. The 'Path' column shows the full path for each output.

Il existe un chemin mappé pour « language ». Par conséquent, il y a une faute de frappe dans les paramètres des compétences. Pour résoudre ce problème, l'expression dans « /document/language » de la compétence #1 doit être mise à jour.

1. Ouvrez l'évaluateur d'expression </> pour le chemin « language ».
2. Copiez l'expression. Fermez la fenêtre.

3. Accédez aux paramètres des compétences de la compétence #1, puis ouvrez l'évaluateur d'expression </> pour l'entrée « languageCode ».
4. Collez la nouvelle valeur « /document/language » dans la zone Expression, puis cliquez sur **Évaluer**.
5. Elle doit afficher l'entrée correcte « en ». Cliquez sur Appliquer pour mettre à jour l'expression.
6. Cliquez sur **Enregistrer** dans le volet des détails des compétences, à droite.
7. Cliquez sur **Exécuter** dans le menu Fenêtre de la session. Cela lance une autre exécution de l'ensemble de compétences avec le document.

Une fois l'exécution de la session de débogage terminée, cliquez sur l'onglet Erreurs/avertissemets. Il indique alors que l'erreur intitulée « Enrichment.NerSkillV2.#1 » a disparu. Toutefois, il y a toujours deux avertissements indiquant que le service n'a pas pu mapper les champs de sortie des organisations (organizations) et des emplacements (locations) à l'index de recherche. Des valeurs sont manquantes : « /document/merged\_content/organizations » et « /document/merged\_content(locations) ».

## Remédier aux valeurs de sortie de compétence manquantes

Session status: Done  
Debug session execution complete.

Definition AI Enrichments Errors/Warnings (2)

Errors/Warnings (2)

1 Operation ↑

Message ↑

Details

Projection.SearchIndex.OutputFieldMapping.locations Could not map output field 'locations' to search index. Check your indexer's 'outputFieldMappings' prop... Missing value '/document/merged\_content/locations'.

Projection.SearchIndex.OutputFieldMapping.organizations Could not map output field 'organizations' to search index. Check your indexer's 'outputFieldMappings' ... Missing value '/document/merged\_content/organizations'.

Il manque des valeurs de sortie d'une compétence. Pour identifier la compétence avec l'erreur, accédez à la structure de données enrichie, recherchez le nom de la valeur, puis examinez sa source d'origine. Dans le cas des valeurs « organizations » et « locations » manquantes, il s'agit de sorties de la compétence #1. Le fait d'ouvrir l'évaluateur d'expression </> pour chaque chemin affiche les expressions listées sous la forme « /document/content/organizations » et « /document/content/locations », respectivement.

debug-2

Session status: Done  
Debug session execution complete.

Definition AI Enrichments Errors/Warnings (3)

Skill Graph Enriched Data Structure

Filter Path /document/path

Path	Output	Originating Source
document	{"normalized_images": [{"type": "file", "url": "@trim..."}]	#5
merged_content	"\n\n\n[{"image": "image0.jpg", "Study of BMN 110 in ..."}]	#2
keyphrases	["Study of BMN", "Syndrome", "Pediatric Patients", "V..."]	#4
translated_text	"\n[{"image": "image0.jpg", "Study of BMN 110 in Pedia..."}]	#3
language	"en"	#1
content		#1
entities		#1
locations		#1
organizations		#1

Expression evaluator

Context /document

Expression /document/content/organizations

Evaluate

Value

Close

La sortie de ces entités est vide alors qu'elle ne doit pas l'être. Quelles sont les entrées produisant ce résultat ?

1. Accédez au **graphe des compétences**, puis sélectionnez la compétence #1.
2. Sélectionnez l'onglet **Exécutions** dans le volet des détails des compétences, à droite.
3. Ouvrez l'évaluateur d'expression </> pour l'entrée (INPUT) « text ».

The screenshot shows the configuration interface for an Entity Recognition Skill. At the top, there's a header with the skill's name (#1 Entity Recognition Skill) and standard save, discard, and delete buttons. Below the header, a note states: "This skill uses a pretrained model to establish entities for a fixed set of categories: people, location, organization, email, phone, address, date, currency, percentage, and time." A "Learn more" link is also present.

The main area has tabs for "Skill Settings", "Skill JSON Editor", "Executions" (which is selected), and "Errors/Warnings".

In the "Executions" tab, there's a table titled "INPUTS" with columns: Iteration, Property, Name, Source, and Path. One row is shown with "Name" as "text", "Source" as "Data Source", and "Path" as "</> =\$(/document/content)".

Below the table, there are sections for "Expression evaluator", "Context" (containing "/document"), and "Expression" (containing "= \$(/document/content)"). An "Evaluate" button is next to the expression input field.

On the right side, there's a sidebar titled "Value" showing the result of the evaluation: "\n \n\n[image: image0.jpg]\n\n\n". There's also a "Wrap Columns" checkbox.

Le résultat affiché pour cette entrée ne ressemble pas à une entrée de texte. Il ressemble à une image qui est entourée de nouvelles lignes. L'absence de texte signifie qu'aucune entité ne peut être identifiée. Si vous observez la hiérarchie de l'ensemble de compétences, vous pouvez voir que le contenu est d'abord traité par la compétence #6 (OCR), puis transmis à la compétence #5 (Fusion).

1. Sélectionnez la compétence #5 (Fusion) dans le **graphe des compétences**.
2. Sélectionnez l'onglet **Exécutions** dans le volet des détails des compétences, à droite, puis ouvrez l'évaluateur d'expression </> pour les sorties (OUTPUTS) « mergedText ».

The screenshot shows the DXP interface with the Expression evaluator dialog open. The expression `/document/merged_content` is evaluated, resulting in the following text:

```

1  "\n \n\n[image: image0.jpg] Study of BMN 110 in
Pediatric Patients < 5 Years of Age With
Mucopolysaccharides IVA (Morquio A Syndrome) This
open-label Phase 2 study will evaluate the safety and
efficacy of weekly 2.0 mg/kg/wk infusions of BMN 110
in pediatric patients, less than 5 years of age at
the time of administration of the first dose of study
drug, diagnosed with MPS IVA (Morquio A Syndrome) for
up to 208 weeks. \n\n\n"

```

The execution details for skill #5, Merge Skill, are shown in the background. The skill consolidates text from a collection of fields into a single field. The skill settings show the following configuration:

Iteration	Property	Name	Source
0	text	Data Source	
	INPUTS	itemsToInsert	#6 Data Source
		offsets	Data Source
OUTPUTS	mergedText		

Ici, le texte est associé à l'image. Si vous observez l'expression « `/document/merged_content` », l'erreur dans les chemins « `organizations` » et « `locations` » pour la compétence #1 est visible. Au lieu d'utiliser « `/document/content` », elle doit utiliser « `/document/merged_content` » pour les entrées « `text` ».

1. Copiez l'expression correspondant à la sortie « `mergedText` », puis fermez la fenêtre de l'évaluateur d'expression.
2. Sélectionnez la compétence #1 dans le **graphe des compétences**.
3. Sélectionnez l'onglet **Paramètres des compétences** dans le volet des détails des compétences, à droite.
4. Ouvrez l'évaluateur d'expression `</>` pour l'entrée « `text` ».
5. Collez la nouvelle expression dans la zone. Cliquez sur **Évaluer**.
6. L'entrée correcte avec le texte ajouté doit s'afficher. Cliquez sur **Appliquer** pour mettre à jour les paramètres des compétences.
7. Cliquez sur **Enregistrer** dans le volet des détails des compétences, à droite.
8. Cliquez sur **Exécuter** dans le menu Fenêtre de la session. Cela lance une autre exécution de l'ensemble de compétences avec le document.

Une fois l'exécution de l'indexeur terminée, les erreurs sont toujours présentes. Revenez à la compétence #1, puis recherchez la cause du problème. L'entrée pour la compétence a été corrigée de « `contenu` » en « `merged_content` ». Quelles sont les sorties de ces entités dans la compétence ?

1. Sélectionnez l'onglet **Enrichissements par IA**.
2. Sélectionnez le **graphe des compétences**, puis cliquez sur la compétence #1.
3. Parcourez les **paramètres des compétences** pour rechercher «outputs» (sorties).
4. Ouvrez l'évaluateur d'expression `</>` pour l'entité « `organizations` ».

The screenshot shows the Expression evaluator dialog box. The 'Entity' dropdown is set to 'Expression evaluator'. The 'Context' field contains '/document/content'. The 'Expression' field contains '/document/content/organizations'. The 'Evaluate' button is visible. The 'Value' section displays a table with three rows:

1	
2	"BMN"
3	

A 'Wrap Columns' checkbox is checked. A 'Close' button is at the bottom left. Below the dialog, a list of competence types is shown:

- `↳ — {name : "organizations", targetName : "organizations" }`
- `↳ — {name : "locations", targetName : "locations" }`
- `↳ — {name : "entities", targetName : "entities" }`
- `+`

L'évaluation du résultat de l'expression donne le résultat correct. La compétence consiste à identifier la valeur correcte pour l'entité, « organizations ». Toutefois, le mappage de sortie dans le chemin de l'entité génère toujours une erreur. En comparant le chemin de sortie de la compétence avec le chemin de sortie de l'erreur, la compétence apparaît les sorties, les organisations et les emplacements sous le nœud /document/content, tandis que le mappage de champs de sortie s'attend à ce que les résultats soient apparentés sous le nœud /document/merged\_content. À l'étape précédente, l'entrée est passée de « /document/content » à « /document/merged\_content ». Le contexte dans les paramètres des compétences doit être changé afin de garantir que la sortie est générée avec le contexte approprié.

1. Sélectionnez l'onglet **Enrichissements par IA**.
2. Sélectionnez le **graphe des compétences**, puis cliquez sur la compétence #1.
3. Accédez aux **Paramètres des compétences** pour rechercher « context » (contexte).
4. Double-cliquez sur le paramètre pour « context », puis remplacez-le pour qu'il indique « /document/merged\_content ».
5. Cliquez sur **Enregistrer** dans le volet des détails des compétences, à droite.
6. Cliquez sur **Exécuter** dans le menu Fenêtre de la session. Cela lance une autre exécution de l'ensemble de compétences avec le document.

The screenshot shows the configuration page for an Entity Recognition Skill named '#1'. The skill uses a pretrained model to establish entities for a fixed set of categories. The configuration includes:

- Skill Settings**: Shows the skill's name (#1), description (null), context (/document/merged\_content), and categories (Person, Quantity, Organization).
- Executions**: Shows 0 executions.
- Errors/Warnings**: Shows 0 errors and 0 warnings.

The JSON configuration code is displayed below:

```
{  
  "name": "#1",  
  "description": null,  
  "context": "/document/merged_content",  
  "categories": [  
    "Person",  
    "Quantity",  
    "Organization"]}
```

Toutes les erreurs ont été résolues.

## Valider les changements apportés à l'ensemble de compétences

Quand la session de débogage a été lancée, le service de recherche a créé une copie de l'ensemble de compétences. Cela a pour but que les changements apportés n'affectent pas le système de production. Maintenant que vous avez terminé le débogage de votre ensemble de compétences, les correctifs peuvent être validés (remplacer l'ensemble de compétences d'origine) pour le système de production. Si vous voulez continuer à apporter des changements à l'ensemble de compétences sans affecter le système de production, la session de débogage peut être enregistrée et rouverte ultérieurement.

1. Cliquez sur **Valider les changements** dans le menu principal Sessions de débogage.
2. Cliquez sur **OK** pour vérifier que vous souhaitez mettre à jour votre ensemble de compétences.
3. Fermez la session de débogage, puis sélectionnez l'onglet **Indexeurs**.
4. Ouvrez votre indexeur « clinical-trials-idxr ».
5. Cliquez sur **Réinitialiser**.
6. Cliquez sur **Exécuter**. Cliquez sur **OK** pour confirmer.

Une fois l'exécution de l'indexeur terminée, il doit y avoir une coche verte et le mot « Success » (Réussite) en regard de l'horodatage de la dernière exécution sous l'onglet Historique d'exécution. Pour vérifier que les changements ont été appliqués :

1. Quittez **Indexeur**, puis sélectionnez l'onglet **Index**.
2. Ouvrez l'index « clinical-trials », puis sous l'onglet Explorateur de recherche, cliquez sur **Rechercher**.
3. La fenêtre de résultats doit indiquer que les entités organizations et locations sont désormais remplies avec les valeurs attendues.

## Nettoyer les ressources

Lorsque vous travaillez dans votre propre abonnement, il est recommandé, à la fin de chaque projet, de déterminer si vous avez toujours besoin des ressources que vous avez créées. Les ressources laissées en cours d'exécution peuvent vous coûter de l'argent. Vous pouvez supprimer les ressources une par une, ou choisir de supprimer le groupe de ressources afin de supprimer l'ensemble des ressources.

Vous pouvez rechercher et gérer les ressources dans le portail à l'aide des liens **Toutes les ressources** ou **Groupes de ressources** situés dans le volet de navigation de gauche.

Si vous utilisez un service gratuit, n'oubliez pas que vous êtes limité à trois index, indexeurs et sources de données.

Vous pouvez supprimer des éléments un par un dans le portail pour ne pas dépasser la limite.

## Étapes suivantes

[En savoir plus sur les ensembles de compétences](#) [En savoir plus sur l'enrichissement incrémentiel et la mise en cache](#)

# Tutoriel : Créer un analyseur personnalisé pour les numéros de téléphone

04/10/2020 • 24 minutes to read • [Edit Online](#)

Les [analyseurs](#) sont un composant clé de toute solution de recherche. Pour améliorer la qualité des résultats de la recherche, il est important de comprendre le fonctionnement des analyseurs et leur impact sur ces résultats.

Dans certains cas, comme dans celui d'un champ de texte libre, il suffit de sélectionner l'[analyseur de langage](#) approprié pour améliorer les résultats de la recherche. Toutefois, dans d'autres scénarios comme la recherche de numéros de téléphone, d'URL ou d'e-mails, il peut être nécessaire de faire appel à des analyseurs personnalisés.

Ce tutoriel utilise Postman et les [API REST](#) de Recherche cognitive Azure pour :

- Expliquer le fonctionnement des analyseurs
- Définir un analyseur personnalisé pour rechercher des numéros de téléphone
- Tester la manière dont l'analyseur personnalisé génère des jetons à partir du texte
- Créer des analyseurs distincts pour l'indexation et la recherche afin d'améliorer les résultats

## Prérequis

Les services et outils suivants sont indispensables dans ce tutoriel.

- [Application de bureau Postman](#)
- [Créer ou rechercher un service de recherche existant](#)

## Télécharger les fichiers

Le code source pour ce tutoriel se trouve dans le dossier [custom-analyzers](#) du dépôt GitHub [Azure-Samples/azure-search-postman-samples](#).

## 1 - Créer un service Recherche cognitive Azure

Pour suivre ce didacticiel, vous avez besoin d'un service Recherche cognitive Azure, que vous pouvez [créer dans le portail](#). Vous pouvez utiliser le niveau gratuit pour effectuer cette procédure pas à pas.

Pour l'étape suivante, vous devez connaître le nom de votre service de recherche et sa clé API. Si vous ne savez pas où trouver ces éléments, consultez ce [guide de démarrage rapide](#).

## 2 - Configurer Postman

Ensuite, démarrez Postman et importez la collection que vous venez de télécharger à partir du dépôt [Azure-Samples/azure-search-postman-samples](#).

Pour importer la collection, accédez à **Fichiers > Importer**, puis sélectionnez le fichier de collection à importer.

Pour chaque demande, vous devez effectuer les étapes suivantes :

1. Remplacez `<YOUR-SEARCH-SERVICE>` par le nom de votre service de recherche.
2. Remplacez `<YOUR-ADMIN-API-KEY>` par la clé primaire ou secondaire de votre service de recherche.

The screenshot shows the Postman interface with a red box highlighting the URL and the 'Content-Type' header in the 'Headers' section. Another red box highlights the 'Status' field in the bottom right corner.

Si vous ne connaissez pas bien Postman, consultez [Explorer les API REST de Recherche cognitive Azure avec Postman](#).

### 3 - Créer un index initial

Dans cette étape, nous allons créer un index initial, charger des documents dans l'index, puis interroger les documents pour voir comment nos recherches initiales fonctionnent.

#### Créer un index

Nous allons commencer par créer un index simple appelé `tutorial-basic-index` avec deux champs : `id` et `phone_number`. Comme nous n'avons pas encore défini d'analyseur, l'analyseur `standard.lucene` est utilisé par défaut.

Pour créer l'index, nous envoyons la demande suivante :

```
PUT https://<YOUR-SEARCH-SERVICE-NAME>.search.windows.net/indexes/tutorial-basic-index?api-version=2019-05-06
Content-Type: application/json
api-key: <YOUR-ADMIN-API-KEY>

{
  "fields": [
    {
      "name": "id",
      "type": "Edm.String",
      "key": true,
      "searchable": true,
      "filterable": false,
      "facetable": false,
      "sortable": true
    },
    {
      "name": "phone_number",
      "type": "Edm.String",
      "sortable": false,
      "searchable": true,
      "filterable": false,
      "facetable": false
    }
  ]
}
```

#### Charger les données

Ensuite, nous chargeons des données dans l'index. Dans certains cas, vous ne pouvez pas contrôler le format des numéros de téléphone ingérés. Nous allons donc tester plusieurs types de formats. Dans l'idéal, une solution de recherche retourne tous les numéros de téléphone correspondants, quel que soit leur format.

Les données sont chargées dans l'index à l'aide de la demande suivante :

```

POST https://<YOUR-SEARCH-SERVICE-NAME>.search.windows.net/indexes/tutorial-basic-index/docs/index?api-version=2019-05-06
Content-Type: application/json
api-key: <YOUR-ADMIN-API-KEY>

{
  "value": [
    {
      "@search.action": "upload",
      "id": "1",
      "phone_number": "425-555-0100"
    },
    {
      "@search.action": "upload",
      "id": "2",
      "phone_number": "(321) 555-0199"
    },
    {
      "@search.action": "upload",
      "id": "3",
      "phone_number": "+1 425-555-0100"
    },
    {
      "@search.action": "upload",
      "id": "4",
      "phone_number": "+1 (321) 555-0199"
    },
    {
      "@search.action": "upload",
      "id": "5",
      "phone_number": "4255550100"
    },
    {
      "@search.action": "upload",
      "id": "6",
      "phone_number": "13215550199"
    },
    {
      "@search.action": "upload",
      "id": "7",
      "phone_number": "425 555 0100"
    },
    {
      "@search.action": "upload",
      "id": "8",
      "phone_number": "321.555.0199"
    }
  ]
}

```

Une fois les données dans l'index, nous pouvons lancer la recherche.

## Recherche

Pour rendre la recherche intuitive, il est préférable de partir du principe que les utilisateurs ne mettront pas en forme leurs requêtes d'une manière spécifique. Un utilisateur recherchant `(425) 555-0100` doit pouvoir entrer ce numéro dans n'importe lequel des formats ci-dessus et obtenir les résultats escomptés. Dans cette étape, nous allons tester quelques exemples de requêtes pour voir ce qu'elles donnent.

Commençons par rechercher `(425) 555-0100` :

```
GET https://<YOUR-SEARCH-SERVICE-NAME>.search.windows.net/indexes/tutorial-basic-index/docs?api-version=2019-05-06&search=(425) 555-0100
Content-Type: application/json
api-key: <YOUR-ADMIN-API-KEY>
```

Cette requête retourne **trois des quatre résultats attendus**, mais également deux résultats inattendus :

```
{
  "value": [
    {
      "@search.score": 0.05634898,
      "phone_number": "+1 425-555-0100"
    },
    {
      "@search.score": 0.05634898,
      "phone_number": "425 555 0100"
    },
    {
      "@search.score": 0.05634898,
      "phone_number": "425-555-0100"
    },
    {
      "@search.score": 0.020766128,
      "phone_number": "(321) 555-0199"
    },
    {
      "@search.score": 0.020766128,
      "phone_number": "+1 (321) 555-0199"
    }
  ]
}
```

Recherchons ensuite un numéro sans mise en forme (`4255550100`).

```
GET https://<YOUR-SEARCH-SERVICE-NAME>.search.windows.net/indexes/tutorial-basic-index/docs?api-version=2019-05-06&search=4255550100
api-key: <YOUR-ADMIN-API-KEY>
```

Cette requête est encore pire puisqu'elle ne retourne qu'**un des quatre numéros correspondants**.

```
{
  "value": [
    {
      "@search.score": 0.6015292,
      "phone_number": "4255550100"
    }
  ]
}
```

Si vous trouvez que ces résultats prêtent à confusion, vous n'êtes pas le seul. Dans la section suivante, nous allons examiner en détail pourquoi nous obtenons ces résultats.

## 4 - Déboguer les résultats de la recherche

Pour comprendre les résultats de la recherche, nous devons d'abord comprendre le fonctionnement des analyseurs. Nous pourrons ensuite tester l'analyseur par défaut à l'aide de l'[API d'analyse de texte](#), puis créer un analyseur qui répond à nos besoins.

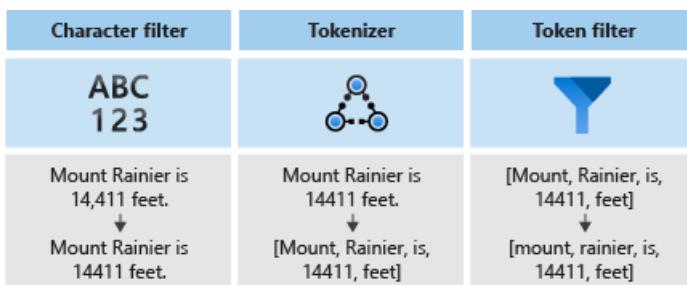
### Fonctionnement des analyseurs

Un [analyseur](#) est un composant du [moteur de recherche en texte intégral](#) chargé de traiter le texte dans les chaînes de requête et les documents indexés. La façon dont les différents analyseurs manipulent le texte varie en fonction du scénario. Pour ce scénario, nous devons créer un analyseur adapté aux numéros de téléphone.

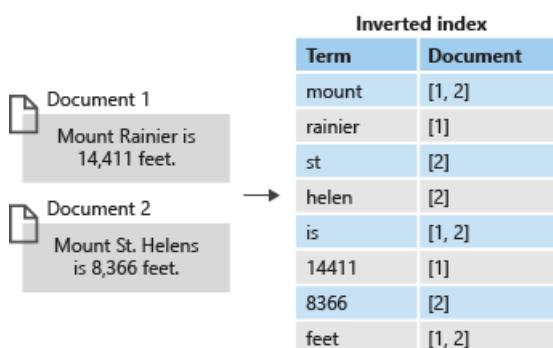
Un analyseur comprend trois composants :

- Des [filtres de caractères](#) qui suppriment ou remplacent des caractères individuels du texte d'entrée.
- Un [générateur de jetons](#) qui divise le texte d'entrée en jetons, lesquels deviennent des clés dans l'index de recherche.
- Des [filtres de jetons](#) qui manipulent les jetons produits par le générateur de jetons.

Le diagramme ci-dessous vous montre comment ces trois composants fonctionnent ensemble pour générer des jetons à partir d'une phrase :

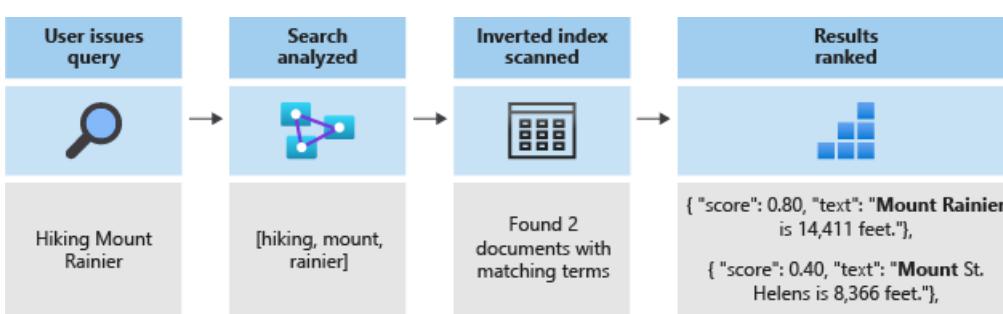


Ces jetons sont ensuite stockés dans un index inversé qui autorise des recherches rapides en texte intégral. Pour cela, un index inversé mappe tous les termes uniques extraits durant l'analyse lexicale aux documents dans lesquels ils apparaissent. Le diagramme ci-dessous vous montre un exemple :



Toute la recherche se résume à rechercher les termes stockés dans l'index inversé. Quand un utilisateur émet une requête :

1. La requête est analysée et les termes de la requête sont analysés.
2. L'index inversé est ensuite analysé à la recherche de documents contenant des termes correspondants.
3. Enfin, les documents récupérés sont classés par l'[algorithme de similarité](#).



Si les termes de la requête ne correspondent pas aux termes de votre index inversé, aucun résultat n'est retourné. Pour en savoir plus sur le fonctionnement des requêtes, consultez cet article sur la [recherche en texte intégral](#).

## NOTE

Les [requêtes sur des termes partiels](#) constituent une exception importante à cette règle. Contrairement aux requêtes sur des termes réguliers, ces requêtes (requêtes avec des préfixes, des caractères génériques ou des expressions régulières) contournent le processus d'analyse lexicale. Les termes partiels sont uniquement mis en minuscules avant d'être mis en correspondance avec les termes de l'index. Si un analyseur n'est pas configuré pour prendre en charge ces types de requêtes, vous obtenez souvent des résultats inattendus dans la mesure où les termes correspondants n'existent pas dans l'index.

## Analyseur de test utilisant l'API d'analyse de texte

Recherche cognitive Azure fournit une [API d'analyse de texte](#) qui vous permet de tester les analyseurs pour comprendre comment ils traitent le texte.

Pour appeler l'API d'analyse de texte, utilisez la requête suivante :

```
POST https://<YOUR-SEARCH-SERVICE-NAME>.search.windows.net/indexes/tutorial-basic-index/analyze?api-version=2019-05-06
Content-Type: application/json
api-key: <YOUR-ADMIN-API-KEY>

{
    "text": "(425) 555-0100",
    "analyzer": "standard.lucene"
}
```

L'API retourne ensuite une liste des jetons extraits du texte. Vous pouvez voir que l'analyseur Lucene standard divise le numéro de téléphone en trois jetons distincts :

```
{
    "tokens": [
        {
            "token": "425",
            "startOffset": 1,
            "endOffset": 4,
            "position": 0
        },
        {
            "token": "555",
            "startOffset": 6,
            "endOffset": 9,
            "position": 1
        },
        {
            "token": "0100",
            "startOffset": 10,
            "endOffset": 14,
            "position": 2
        }
    ]
}
```

À l'inverse, le numéro de téléphone `4255550100` mis en forme sans ponctuation produit un seul jeton.

```
{
    "text": "4255550100",
    "analyzer": "standard.lucene"
}
```

```
{
  "tokens": [
    {
      "token": "4255550100",
      "startOffset": 0,
      "endOffset": 10,
      "position": 0
    }
  ]
}
```

Gardez à l'esprit que les termes de la requête et les documents indexés sont analysés. Si vous repensez aux résultats de la recherche obtenus à l'étape précédente, vous pouvez commencer à comprendre pourquoi ces résultats ont été retournés.

Dans la première requête, des numéros de téléphone incorrects ont été retournés, car ils contenaient le terme `555` qui faisait partie des termes recherchés. Dans la deuxième requête, un seul numéro a été retourné parce qu'un seul enregistrement contenait un terme correspondant à `4255550100`.

## 5 - Créer un analyseur personnalisé

Maintenant que nous comprenons les résultats que nous avons obtenus, nous allons créer un analyseur personnalisé pour améliorer la logique de génération de jetons.

L'objectif est de pouvoir rechercher de manière intuitive des numéros de téléphone, quel que soit le format de la requête ou de la chaîne indexée. Pour y parvenir, nous allons spécifier un [filtre de caractères](#), un [générateur de jetons](#) et un [filtre de jetons](#).

### Filtres de caractères

Les filtres de caractères permettent de traiter le texte avant de l'envoyer au générateur de jetons. Les filtres de caractères sont couramment utilisés pour rejeter des éléments HTML ou remplacer des caractères spéciaux.

Pour les numéros de téléphone, nous voulons supprimer les espaces blancs et les caractères spéciaux, car tous les formats de numéro de téléphone ne contiennent pas les mêmes caractères spéciaux et espaces.

```
"charFilters": [
  {
    "@odata.type": "#Microsoft.Azure.Search.MappingCharFilter",
    "name": "phone_char_mapping",
    "mappings": [
      "-=>",
      "(=>",
      ")=>",
      "+=>",
      ".=>",
      "\u0020=>"
    ]
  }
]
```

Le filtre ci-dessus supprime `-` `(` `)` `+` `.` et les espaces de l'entrée.

ENTRÉE	OUTPUT
<code>(321) 555-0199</code>	<code>3215550199</code>
<code>321.555.0199</code>	<code>3215550199</code>

## Générateurs de jetons

Les générateurs de jetons divisent le texte en jetons et rejettent certains caractères, comme les signes de ponctuation, au cours du processus. Dans de nombreux cas, l'objectif de la génération de jetons est de diviser une phrase en mots individuels.

Pour ce scénario, nous allons utiliser le générateur de jetons `keyword_v2` pour capturer le numéro de téléphone comme un terme unique. Notez qu'il existe d'autres moyens de résoudre ce problème. Pour les voir, consultez la section [Autres approches](#) ci-dessous.

Les générateurs de jetons (mots clés) génèrent toujours le même texte que celui qu'ils ont reçu sous la forme d'un terme unique.

ENTRÉE	OUTPUT
The dog swims.	[The dog swims.]
3215550199	[3215550199]

## Filtres de jeton

Les filtres de jetons rejettent ou modifient les jetons générés par le générateur de jetons. Un filtre de jetons est couramment utilisé pour mettre en minuscules pour tous les caractères à l'aide d'un filtre de jeton. Une autre utilisation courante consiste à rejeter les mots vides comme `the`, `and` ou `is`.

Bien que nous n'ayons pas besoin d'utiliser l'un de ces filtres dans ce scénario, nous allons utiliser un filtre de jetons n-gramme pour pouvoir effectuer des recherches partielles de numéros de téléphone.

```
"tokenFilters": [
  {
    "@odata.type": "#Microsoft.Azure.Search.NGramTokenFilterV2",
    "name": "custom_ngram_filter",
    "minGram": 3,
    "maxGram": 20
  }
]
```

### NGramTokenFilterV2

Le [filtre de jetons nGram\\_v2](#) divise les jetons en n-grammes d'une taille donnée en fonction des paramètres `minGram` et `maxGram`.

Pour l'analyseur de numéros de téléphone, nous définissons `minGram` avec la valeur `3`, car nous ne nous attendons pas à ce que les utilisateurs recherchent des sous-chaînes plus courtes. Nous définissons `maxGram` avec la valeur `20` pour nous assurer que tous les numéros de téléphone, même ceux avec des extensions, tiennent dans un seul n-gramme.

Malheureusement, la génération de faux positifs est un effet secondaire des n-grammes. Nous corrigerais ce problème à l'étape 7 en créant un analyseur distinct pour les recherches qui n'inclut pas le filtre de jeton n-gramme.

ENTRÉE	OUTPUT
[12345]	[123, 1234, 12345, 234, 2345, 345]
[3215550199]	[321, 3215, 32155, 321555, 3215550, 32155501, 321555019, 3215550199, 215, 2155, 21555, 215550, ...]

## Analyseur

Une fois les filtres de caractères, le générateur de jetons et les filtres de jetons en place, nous sommes prêts à définir notre analyseur.

```
"analyzers": [
  {
    "@odata.type": "#Microsoft.Azure.Search.CustomAnalyzer",
    "name": "phone_analyzer",
    "tokenizer": "custom_tokenizer_phone",
    "tokenFilters": [
      "custom_ngram_filter"
    ],
    "charFilters": [
      "phone_char_mapping"
    ]
  }
]
```

ENTRÉE	OUTPUT
12345	[123, 1234, 12345, 234, 2345, 345]
(321) 555-0199	[321, 3215, 32155, 321555, 3215550, 32155501, 321555019, 3215550199, 215, 2155, 21555, 215550, ...]

Notez qu'il est possible de lancer une recherche sur n'importe lequel des jetons dans la sortie. Si notre requête contient l'un de ces jetons, le numéro de téléphone est retourné.

Une fois l'analyseur personnalisé défini, recréez l'index pour que l'analyseur personnalisé puisse être testé à l'étape suivante. Par souci de simplicité, la collection Postman crée un index nommé `tutorial-first-analyzer` avec l'analyseur que nous avons défini.

## 6 - Tester l'analyseur personnalisé

Une fois l'index créé, vous pouvez tester l'analyseur que nous avons créé à l'aide de la requête suivante :

```
POST https://<YOUR-SEARCH-SERVICE-NAME>.search.windows.net/indexes/tutorial-first-analyzer/analyze?api-version=2019-05-06
Content-Type: application/json
api-key: <YOUR-ADMIN-API-KEY>

{
  "text": "+1 (321) 555-0199",
  "analyzer": "phone_analyzer"
}
```

Vous pouvez alors voir la collection de jetons résultant du numéro de téléphone :

```
{
  "tokens": [
    {
      "token": "132",
      "startOffset": 1,
      "endOffset": 17,
      "position": 0
    },
    {
      "token": "1321",
      "startOffset": 1,
      "endOffset": 17,
      "position": 0
    },
    {
      "token": "13215",
      "startOffset": 1,
      "endOffset": 17,
      "position": 0
    },
    ...
    ...
    ...
  ]
}
```

## 7 - Créer un analyseur personnalisé pour les requêtes

Après avoir exécuté quelques exemples de requêtes sur l'index à l'aide de l'analyseur personnalisé, vous constatez que le rappel a été amélioré et que tous les numéros de téléphone correspondants sont désormais retournés. Toutefois, le filtre de jetons n-gramme retourne également quelques faux positifs. Il s'agit d'un effet secondaire courant avec les filtres de jetons n-gramme.

Pour éviter les faux positifs, nous allons créer un analyseur distinct pour l'interrogation. Cet analyseur est identique à celui que nous avons déjà créé, mais **sans** le filtre `custom_ngram_filter`.

```
{
  "@odata.type": "#Microsoft.Azure.Search.CustomAnalyzer",
  "name": "phone_analyzer_search",
  "tokenizer": "custom_tokenizer_phone",
  "tokenFilters": [],
  "charFilters": [
    "phone_char_mapping"
  ]
}
```

Dans la définition de l'index, nous spécifions un `indexAnalyzer` et un `searchAnalyzer`.

```
{
  "name": "phone_number",
  "type": "Edm.String",
  "sortable": false,
  "searchable": true,
  "filterable": false,
  "facetable": false,
  "indexAnalyzer": "phone_analyzer",
  "searchAnalyzer": "phone_analyzer_search"
}
```

Une fois ce changement effectué, tout est prêt. Recréez l'index, indexez les données et retestez les requêtes pour

vérifier que la recherche fonctionne comme prévu. Si vous utilisez la collection Postman, un troisième index nommé `tutorial-second-analyzer` est créé.

## Autres approches

L'analyseur ci-dessus a été conçu pour optimiser la flexibilité de la recherche. Toutefois, il en résulte le stockage de nombreux termes potentiellement sans importance dans l'index.

L'exemple ci-dessous montre un autre analyseur qui peut également être utilisé pour cette tâche.

L'analyseur fonctionne bien, sauf pour les données d'entrée de type `14255550100` qui rendent difficile la segmentation logique du numéro de téléphone. Par exemple, l'analyseur ne peut pas séparer l'indicatif du pays, `1`, de l'indicatif régional `425`. En raison de ce problème, le numéro ci-dessus n'est pas retourné si un utilisateur n'inclut pas l'indicatif du pays dans sa recherche.

```
"analyzers": [
  {
    "@odata.type": "#Microsoft.Azure.Search.CustomAnalyzer",
    "name": "phone_analyzer_shingles",
    "tokenizer": "custom_tokenizer_phone",
    "tokenFilters": [
      "custom_shingle_filter"
    ]
  }
],
"tokenizers": [
  {
    "@odata.type": "#Microsoft.Azure.Search.StandardTokenizerV2",
    "name": "custom_tokenizer_phone",
    "maxTokenLength": 4
  }
],
"tokenFilters": [
  {
    "@odata.type": "#Microsoft.Azure.Search.ShingleTokenFilter",
    "name": "custom_shingle_filter",
    "minShingleSize": 2,
    "maxShingleSize": 6,
    "tokenSeparator": ""
  }
]
```

Vous pouvez voir dans l'exemple ci-dessous que le numéro de téléphone est divisé en blocs que recherchent normalement les utilisateurs.

ENTRÉE	OUTPUT
(321) 555-0199	[321, 555, 0199, 321555, 5550199, 3215550199]

Selon vos besoins, cette approche peut être plus efficace pour résoudre le problème.

## Réinitialiser et réexécuter

Pour rester simple, ce tutoriel vous a demandé de créer trois index. Toutefois, il est courant de supprimer et de recréer les index au cours des premières phases du développement. Vous pouvez supprimer un index dans le portail Azure ou à l'aide de l'appel d'API suivant :

```
DELETE https://<YOUR-SEARCH-SERVICE-NAME>.search.windows.net/indexes/tutorial-basic-index?api-version=2019-05-06  
api-key: <YOUR-ADMIN-API-KEY>
```

## Éléments importants à retenir

Dans ce tutoriel, vous avez vu le processus de création et de test d'un analyseur personnalisé. Vous avez créé un index, indexé les données, puis interrogé l'index pour voir les résultats de la recherche retournés. Ensuite, vous avez utilisé l'API d'analyse de texte pour voir le processus d'analyse lexicale en action.

Bien que l'analyseur défini dans ce tutoriel offre une solution simple de recherche de numéros de téléphone, vous pouvez suivre ce même processus afin de créer un analyseur personnalisé pour n'importe quel scénario.

## Nettoyer les ressources

Quand vous travaillez dans votre propre abonnement, il est judicieux à la fin d'un projet de supprimer les ressources dont vous n'avez plus besoin. Les ressources laissées en cours d'exécution peuvent vous coûter de l'argent. Vous pouvez supprimer les ressources une par une, ou choisir de supprimer le groupe de ressources afin de supprimer l'ensemble des ressources.

Vous pouvez rechercher et gérer les ressources dans le portail à l'aide des liens Toutes les ressources ou Groupes de ressources situés dans le volet de navigation de gauche.

## Étapes suivantes

Maintenant que vous savez comment créer un analyseur personnalisé, examinons les différents filtres, générateurs de jetons et analyseurs disponibles pour créer une expérience de recherche enrichie.

[Analyseurs personnalisés dans Recherche cognitive Azure](#)

# Tutoriel : Interroger un index Recherche cognitive à partir de Power Apps

04/10/2020 • 14 minutes to read • [Edit Online](#)

Tirez parti de l'environnement de développement rapide d'applications Power Apps pour créer une application personnalisée permettant de rechercher votre contenu dans Recherche cognitive Azure.

Dans ce tutoriel, vous allez apprendre à :

- Se connecter à la Recherche cognitive Azure
- Configurer une demande de requête
- Visualiser les résultats dans une application canevas

Si vous n'avez pas d'abonnement Azure, ouvrez un [compte gratuit](#) avant de commencer.

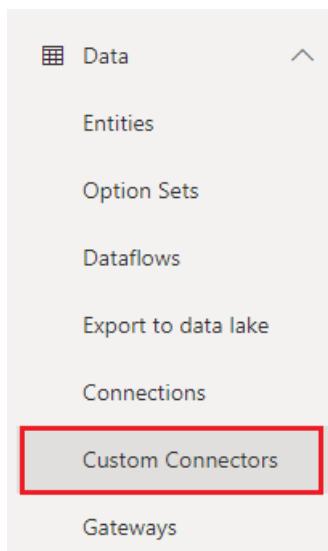
## Prérequis

- [Compte Power Apps](#)
- [Index Hotels-sample](#)
- [Clé d'API de requête](#)

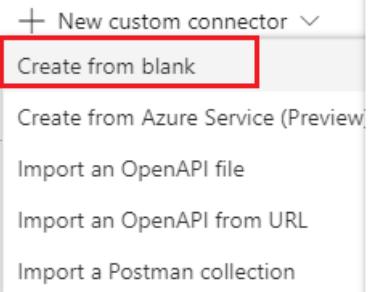
## 1 – Créez un connecteur personnalisé

Dans Power Apps, un connecteur est une connexion de source de données. Dans cette étape, vous allez créer un connecteur personnalisé pour vous connecter à un index de recherche dans le cloud.

1. [Connectez-vous](#) à Power Apps.
2. À gauche, développez **Données > Connecteurs personnalisés**.



3. Sélectionnez + Nouveau connecteur personnalisé, puis Créez entièrement.



4. Donnez un nom à votre connecteur personnalisé (par exemple, *AzureSearchQuery*), puis cliquez sur **Continuer**.

5. Entrez des informations dans la page Général :

- Couleur d'arrière-plan de l'icône (par exemple, #007ee5)
- Description (par exemple, « Connecteur vers Recherche cognitive Azure »)
- Dans l'hôte, vous devez entrer l'URL de votre service de recherche (par exemple, <your servicename>.search.windows.net )
- Pour l'URL de base, entrez simplement « / »

#### General information



Upload connector icon

Supported file formats are PNG and JPG. (< 1MB)

Icon background color

#007ee5

Description

A connector to Azure Cognitive Search

Connect via on-premises data gateway [Learn more](#)

Scheme \*

HTTPS  HTTP

Host \*

mydemo.search.windows.net

Base URL

/

6. Dans la page Sécurité, définissez *Clé de l'API* comme Type d'authentification, puis définissez l'étiquette et le nom du paramètre sur *api-key*. Pour Emplacement du paramètre, sélectionnez *En-tête* comme indiqué ci-dessous.

## Authentication type

Choose what authentication is implemented by your API \*

API Key

 Edit

## API Key

Users will be required to provide the API Key when creating a connection

Parameter label \*

api-key

Parameter name \*

api-key

Parameter location \*

Header

7. Dans la page Définitions, sélectionnez + Nouvelle action pour créer une action qui interroge l'index. Entrez la valeur « Requête » pour le résumé et le nom de l'ID d'opération. Entrez une description comme « *Interroge l'index de recherche* ».

## General

Summary [Learn more](#)

Query

Description [Learn more](#)

Queries an Azure Cognitive Search index

Operation ID \*

This is the unique string used to identify the operation.

Query

Visibility [Learn more](#)

none  advanced  internal  important

8. Faites défiler vers le bas. Dans Demandes, cliquez sur le bouton + Importer à partir de l'exemple pour configurer une demande de requête à destination de votre service de recherche :

- Sélectionnez le verbe `GET`

- Pour l'URL, entrez un exemple de requête pour votre index de recherche (`search=*` retourne tous les documents, `$select=` vous permet de choisir les champs). La version de l'API est obligatoire. Voici à quoi ressemble une URL entièrement spécifiée :

```
https://mydemo.search.windows.net/indexes/hotels-sample-index/docs?  
search=*&$select=HotelName,Description,Address/City&api-version=2020-06-30
```

- Pour En-têtes, tapez `Content-Type`.

**Power Apps** utilise la syntaxe pour extraire les paramètres de la requête. Notez que nous avons défini explicitement le champ de recherche.

## Import from sample

X

### Verb \*

- GET  DELETE  POST  PUT  HEAD  
 OPTIONS  PATCH

### URL \*

`https://mydemo.search.windows.net/indexes/hotels-sample-index/docs`

This is the request URL.

### Headers

`Content-Type application/json`

These are custom headers that are part of the request.

**Import**

**Close**

9. Cliquez sur **Importer** pour compléter automatiquement la demande. Terminez la définition des métadonnées de paramètre en cliquant sur le symbole ... en regard de chacun des paramètres. Cliquez sur **Retour** pour retourner à la page Demande après chaque mise à jour de paramètre.

**Request** + Import from sample

**Verb \***  
The verb describes the operations available on a single path.  
**GET**

**URL \***  
This is the request URL.  
`https://mydemo.search.windows.net/indexes/hotels-sample-index/docs`

**Path**  
Path is used together with Path Templating, where the parameter value is actually part of the operation's URL.

**Query**  
Query parameters are appended to the URL. For example, in `/items?id=####`, the query parameter is `id`.  
`search` ... `$select` ... `api-version` ...

**Headers**  
These are custom headers that are part of the request.  
`Content-Type` ...

**Body**  
The body is the payload that's appended to the HTTP request. There can only be one body parameter.

10. Pour `search`: définissez `*` comme valeur par défaut, définissez obligatoire sur *Faux* et définissez la visibilité sur *aucune*.

## Parameter

Name \*

search

Description [Learn more](#)

Summary [Learn more](#)

Default value

\*

Is required?

Yes  No

Visibility [Learn more](#)

none  advanced  internal  important

11. Pour *select*: définissez `HotelName,Description,Address/City` comme **valeur par défaut**, définissez **obligatoire** sur *Faux* et définissez la **visibilité** sur *aucune*.

## Parameter

Name \*

\$select

Description [Learn more](#)

Summary [Learn more](#)

Default value

`HotelName,Description,Address/City`

Is required?

Yes  No

Visibility [Learn more](#)

none  advanced  internal  important

12. Pour *api-version* : définissez `2020-06-30` comme **valeur par défaut**, définissez **obligatoire** sur *Vrai* et définissez la **visibilité** sur *interne*.

## Parameter

Name \*

Description [Learn more](#)

Summary [Learn more](#)

Default value

Is required?

Yes  No

Visibility [Learn more](#)

none  advanced  internal  important

13. Pour *Content-Type*: Défini sur `application/json`.
14. Une fois ces modifications effectuées, basculez sur la vue **Éditeur Swagger**. Dans la section Paramètres, vous devez voir la configuration suivante :

```
parameters:  
  - {name: search, in: query, required: false, type: string, default: '*'}  
  - {name: $select, in: query, required: false, type: string, default:  
    'HotelName,Description,Address/City'}  
  - {name: api-version, in: query, required: true, type: string, default: '2020-06-30',  
    x-ms-visibility: internal}  
  - {name: Content-Type, in: header, required: false, type: string}
```

15. Revenez à l'étape 3. **Demande** et descendez jusqu'à la section Réponse. Cliquez sur « **Ajouter une réponse par défaut** ». Cela est primordial dans le sens où cela permet à Power Apps de comprendre le schéma de la réponse.
16. Collez un exemple de réponse. Une façon simple de capturer un exemple de réponse est de passer par l'Explorateur de recherche sur le portail Azure. Dans l'Explorateur de recherche, vous devez entrer la même requête que pour la demande, mais en ajoutant `$top=2` pour limiter les résultats à seulement deux documents : `search=*&$select=HotelName,Description,Address/City&$top=2`.

Il suffit de quelques résultats à Power Apps pour détecter le schéma.

```
{
    "@odata.context": "https://mydemo.search.windows.net/indexes('hotels-sample-index')/$metadata#docs(*)",
    "value": [
        {
            "@search.score": 1,
            "HotelName": "Arcadia Resort & Restaurant",
            "Description": "The largest year-round resort in the area offering more of everything for your vacation – at the best value! What can you enjoy while at the resort, aside from the mile-long sandy beaches of the lake? Check out our activities sure to excite both young and young-at-heart guests. We have it all, including being named “Property of the Year” and a “Top Ten Resort” by top publications.",
            "Address": {
                "City": "Seattle"
            }
        },
        {
            "@search.score": 1,
            "HotelName": "Travel Resort",
            "Description": "The Best Gaming Resort in the area. With elegant rooms & suites, pool, cabanas, spa, brewery & world-class gaming. This is the best place to play, stay & dine.",
            "Address": {
                "City": "Albuquerque"
            }
        }
    ]
}
```

#### TIP

Sachant que vous êtes soumis à une limite de caractères dans la réponse JSON que vous pouvez entrer, vous pouvez simplifier le JSON avant de le coller. Le schéma et le format de la réponse sont plus importants que les valeurs elles-mêmes. Par exemple, le champ Description peut être simplifié à la première phrase.

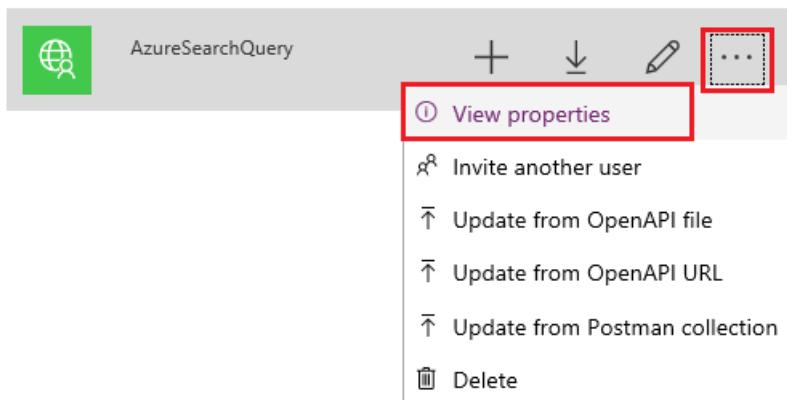
17. Cliquez sur **Créer un connecteur** en haut à droite.

## 2 – Tester la connexion

Après avoir créé le connecteur, vous devez le rouvrir à partir de la liste Connecteurs personnalisés de façon à le tester. Si, par la suite, vous effectuez des mises à jour supplémentaires, vous pourrez le tester depuis l'Assistant.

Vous aurez besoin d'une [clé d'API de requête](#) pour cette tâche. Chaque fois qu'une connexion est créée, que ce soit pour une série de tests ou pour une inclusion dans une application, le connecteur a besoin de la clé d'API de requête pour se connecter à Recherche cognitive Azure.

1. Tout à gauche, cliquez sur **Connecteurs personnalisés**.
2. Recherchez le connecteur par son nom (dans ce tutoriel, il s'agit de « AzureSearchQuery »).
3. Sélectionnez le connecteur, développez la liste d'actions, puis sélectionnez **Afficher les propriétés**.



4. Sélectionnez **Modifier** en haut à droite.
5. Sélectionnez **4. Tester** pour ouvrir la page de test.
6. Dans Opération de test, cliquez sur **+** Nouvelle connexion.
7. Entrez une clé d'API de requête. Il s'agit d'une requête Recherche cognitive Azure pour un accès en lecture seule à un index. Vous pouvez [trouver la clé](#) sur le portail Azure.
8. Dans Opérations, cliquez sur le bouton **Opération de test**. Si l'opération aboutit, vous obtenez un état 200 et un JSON apparaît dans le corps de la réponse avec une description des résultats de la recherche.

```
Request Response

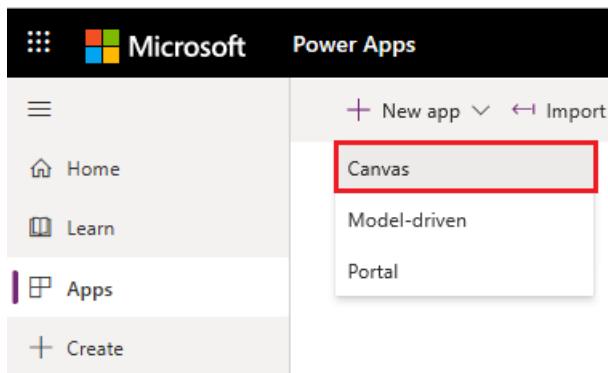
Status
(200)
Headers
{
  "cache-control": "no-cache",
  "content-length": "51163",
  "content-type": "application/json; odata.metadata=minimal",
  "date": "Sun, 19 Apr 2020 03:01:33 GMT",
  "elapsed-time": "116",
}

Body
{
  "@odata.context": "https://mydemo.search.windows.net/indexes('hotels-sample-index')/$metadata#hotels",
  "value": [
    {
      "@search.score": 1,
      "HotelName": "Arcadia Resort & Restaurant",
      "Description": "The largest year-round resort in the area offering more of everything for your vacation needs. Located in the heart of Seattle's business district, the Arcadia is the perfect place to stay while you're in town. Our resort features a variety of rooms and suites, as well as a full-service spa and fitness center. We offer a range of dining options, from casual to fine dining, and a variety of activities for all ages. Whether you're looking for a relaxing vacation or an active adventure, the Arcadia is the perfect choice.",
      "Address": {
        "City": "Seattle"
      }
    }
  ]
}
```

### 3 – Visualiser les résultats

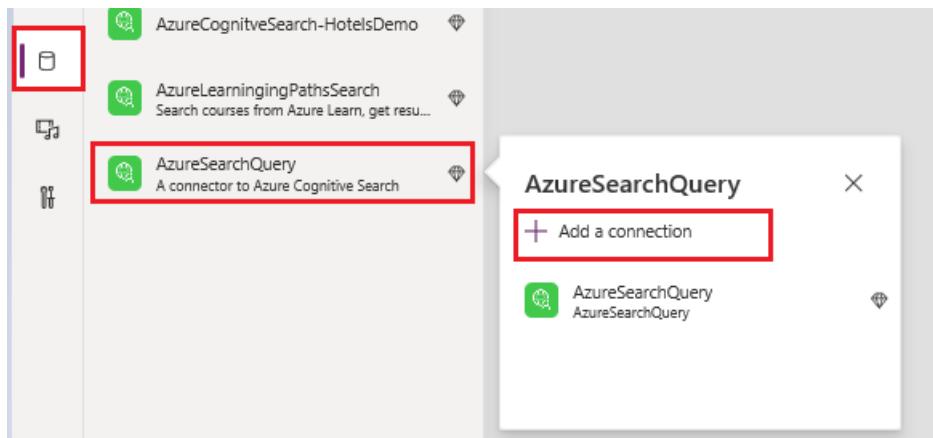
Dans cette étape, vous allez créer une application Power Apps avec une zone de recherche, un bouton de recherche et une zone d'affichage des résultats. L'application Power Apps se connecte au connecteur personnalisé récemment créé pour obtenir les données à partir de Recherche Azure.

1. À gauche, développez Apps > **+** Nouvelle application > Canevas.



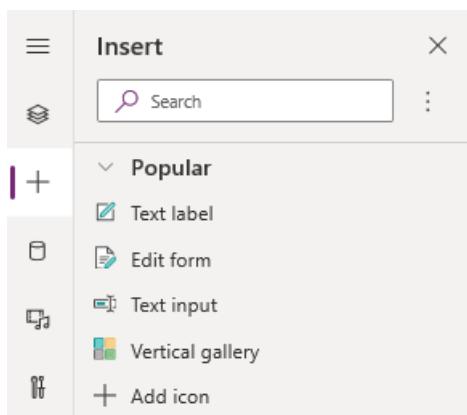
2. Sélectionnez le type d'application. Pour ce tutoriel, créez une **application vide** avec le **Mode téléphone**. **Power Apps Studio** s'affiche.
3. Une fois dans Studio, sélectionnez l'onglet **Sources de données**, puis cliquez sur le nouveau connecteur que vous venez de créer. Dans notre cas, il s'appelle *AzureSearchQuery*. Cliquez sur **Ajouter une connexion**.

Entrez la clé d'API de requête.



À présent, *AzureSearchQuery* est une source de données qui peut être utilisée à partir de votre application.

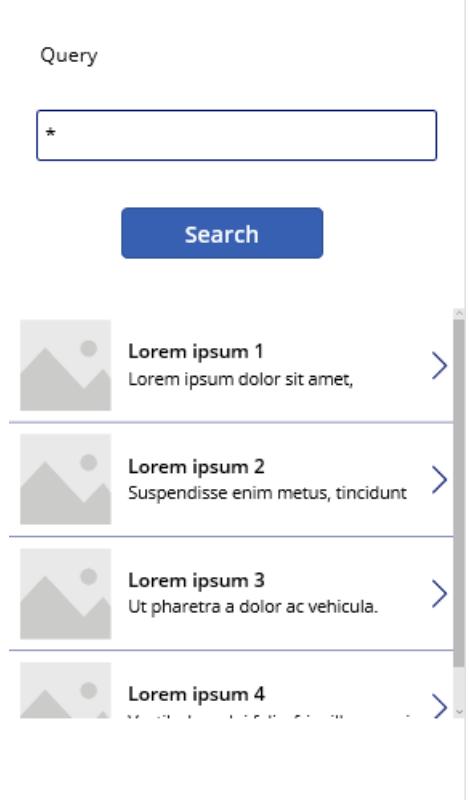
4. Sous l'**onglet Insertion**, ajoutez quelques contrôles au canevas.



5. Insérez les éléments suivants :

- Une étiquette de texte avec la valeur « Requête : »
- Un élément d'entrée de texte (appelez-le *txtQuery*, valeur par défaut : «\*»)
- Un bouton avec le texte « Rechercher »
- Une galerie verticale appelée (appelez-la *galleryResults*)

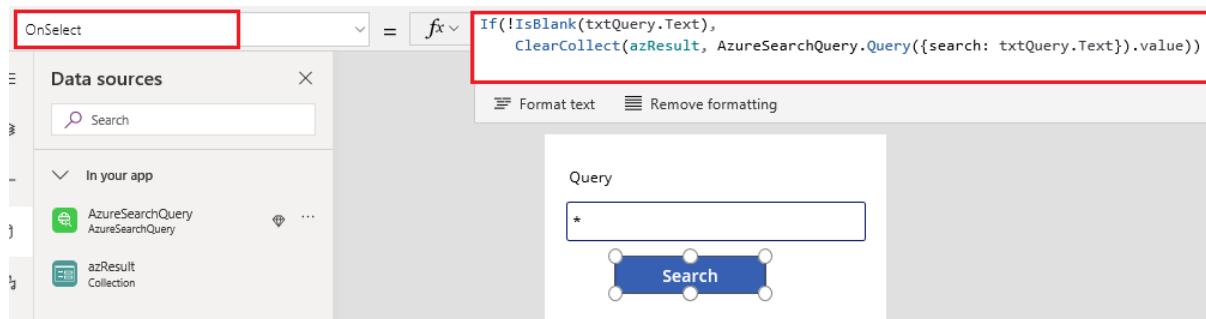
Voici comment se présente le canevas :



- Pour que le bouton Rechercher émette une requête, collez l'action suivante dans OnSelect :

```
If(!IsBlank(txtQuery.Text),
    ClearCollect(azResult, AzureSearchQuery.Query({search: txtQuery.Text}).value))
```

La capture d'écran suivante affiche la barre de formule pour l'action OnSelect.



Cette action fera que le bouton mettra à jour une nouvelle collection appelée *azResult* avec le résultat de la requête de recherche en utilisant le texte de la zone de texte *txtQuery* comme terme de la requête.

#### NOTE

Si vous obtenez une erreur de syntaxe de formule de type « La fonction 'ClearCollect' contient des fonctions non valides », essayez ceci :

- Tout d'abord, vérifiez que la référence du connecteur est correcte. Effacez le nom du connecteur et commencez à taper le nom de votre connecteur. Intellisense doit alors vous suggérer le connecteur et le verbe appropriés.
- Si l'erreur persiste, supprimez et recréez le connecteur. S'il existe plusieurs instances d'un connecteur, il se peut que l'application n'utilise pas le bon.

- Liez le contrôle Galerie verticale à la collection *azResult* que vous avez créée à l'étape précédente.

Sélectionnez le contrôle Galerie, puis effectuez les actions suivantes dans le volet Propriétés.

- Définissez **Source de données** sur *azResult*.
- Sélectionnez une **Disposition** qui vous convienne en fonction du type de données de votre index. Dans ce cas, nous avons utilisé la disposition *Title, subtitle and body* (Titre, sous-titre et corps).
- **Modifiez les champs**, puis sélectionnez les champs que vous souhaitez visualiser.

Étant donné que nous avons fourni un exemple de résultat lorsque nous avons défini le connecteur, l'application connaît les champs disponibles dans votre index.

- Appuyez sur la touche F5 pour afficher un aperçu de l'application.

## Nettoyer les ressources

Lorsque vous travaillez dans votre propre abonnement, il est recommandé, à la fin de chaque projet, de déterminer si vous avez toujours besoin des ressources que vous avez créées. Les ressources laissées en cours d'exécution peuvent vous coûter de l'argent. Vous pouvez supprimer les ressources une par une, ou choisir de supprimer le groupe de ressources afin de supprimer l'ensemble des ressources.

Vous pouvez rechercher et gérer les ressources dans le portail à l'aide des liens **Toutes les ressources** ou **Groupes de ressources** situés dans le volet de navigation de gauche.

Si vous utilisez un service gratuit, n'oubliez pas que vous êtes limité à trois index, indexeurs et sources de données. Vous pouvez supprimer des éléments un par un dans le portail pour ne pas dépasser la limite.

## Étapes suivantes

Power Apps permet de développer rapidement des applications personnalisées. Maintenant que vous savez comment vous connecter à un index de recherche, apprenez-en davantage sur la création d'une expérience de

visualisation enrichie dans une application Power Apps personnalisée.

[Catalogue de formations Power Apps](#)

# Recherche en texte intégral dans Recherche cognitive Azure

04/10/2020 • 35 minutes to read • [Edit Online](#)

Cet article est destiné aux développeurs qui ont besoin d'une compréhension approfondie du fonctionnement de la recherche en texte intégral Lucene dans la Recherche cognitive Azure. Pour les requêtes de texte, la Recherche cognitive Azure fournit en toute transparence les résultats attendus dans la plupart des scénarios, mais il se peut que vous obteniez un résultat « étrange » dans certains cas. Dans ce cas, le fait d'avoir une connaissance des quatre phases d'exécution des requêtes Lucene (analyse des requêtes, analyse lexicale, mise en correspondance des documents et notation) peut vous permettre d'identifier les modifications spécifiques des paramètres de requête ou de la configuration d'index qui permettront d'obtenir le résultat souhaité.

## NOTE

La Recherche cognitive Azure utilise Lucene pour la recherche en texte intégral, mais l'intégration Lucene n'est pas exhaustive. Nous exposons et étendons la fonctionnalité Lucene de façon sélective pour activer les scénarios importants dans la Recherche cognitive Azure.

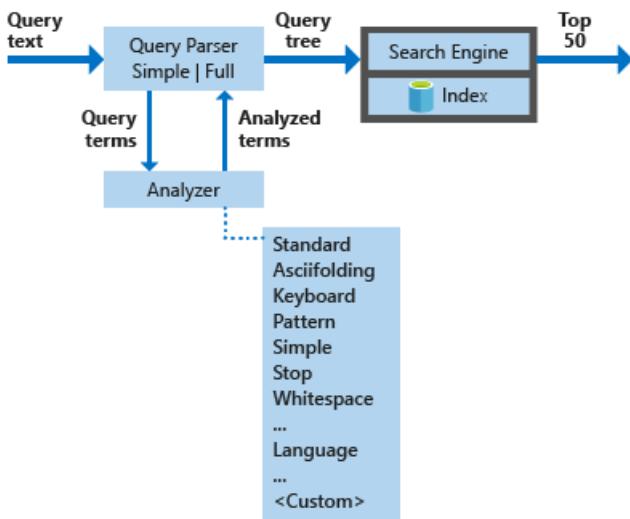
## Présentation et diagramme de l'architecture

Le traitement d'une requête de recherche en texte intégral commence par analyser le texte de la requête pour extraire des termes de recherche. Le moteur de recherche utilise un index pour extraire les documents contenant les termes correspondants. Les termes de requête individuelle sont parfois ventilés et reconstitués dans de nouveaux formulaires pour étendre la recherche sur ce qui pourrait être considéré comme une correspondance potentielle. Le jeu de résultats est ensuite trié en fonction d'un score de pertinence attribué à chaque document individuel correspondant. Les premiers résultats de la liste ordonnée sont renvoyés à l'application appelante.

Après retraitement, l'exécution des requêtes comporte quatre étapes :

1. Analyse des requêtes
2. Analyse lexicale
3. Extraction de documents
4. Notation

Le diagramme ci-dessous illustre les composants utilisés pour traiter une demande de recherche.



COMPOSANTS CLÉS	DESCRIPTION FONCTIONNELLE
<b>Analyseurs de requêtes</b>	Distinguez les termes de requête des opérateurs de requête et créez la structure de la requête (arborescence de requête) à envoyer au moteur de recherche.
<b>Analyseurs</b>	Effectuez une analyse lexicale sur les termes de requête. Ce processus peut impliquer la transformation, la suppression ou le développement des termes de requête.
<b>Index</b>	Structure de données efficace permettant de stocker et d'organiser les termes pouvant faire l'objet d'une recherche extraits à partir de documents indexés.
<b>Moteur de recherche</b>	Extrait et évalue les documents correspondants en fonction du contenu de l'index inversé.

## Anatomie d'une requête de recherche

Une requête de recherche est une spécification complète de ce qui doit être renvoyé dans un jeu de résultats. Dans sa forme la plus simple, il s'agit d'une requête vide sans aucun critère. Un exemple plus réaliste inclut des paramètres, plusieurs termes de requête, peut-être limités à certains champs, avec éventuellement une expression de filtre et des règles de classement.

L'exemple suivant est une requête de recherche que vous pourriez envoyer à la Recherche cognitive Azure à l'aide de l'[API REST](#).

```
POST /indexes/hotels/docs/search?api-version=2020-06-30
{
  "search": "Spacious, air-condition* +\"Ocean view\"",
  "searchFields": "description, title",
  "searchMode": "any",
  "filter": "price ge 60 and price lt 300",
  "orderby": "geo.distance(location, geography'POINT(-159.476235 22.227659)'),",
  "queryType": "full"
}
```

Pour cette requête, le moteur de recherche effectue les opérations suivantes :

1. Filtre les documents dont le prix est supérieur ou égal à 60 dollars et inférieur à 300 dollars.
  2. Exécute la requête. Dans cet exemple, la requête de recherche se compose d'expressions et de termes :  
"Spacious, air-condition\* +\"Ocean view\""
- (en général, les utilisateurs n'entrent pas de ponctuation,

mais l'inclure dans l'exemple nous permet d'expliquer la manière dont les analyseurs la traite). Pour cette requête, le moteur de recherche analyse les champs Description et Titre spécifiés dans `searchFields` pour les documents qui contiennent « Vue mer », mais également le terme « spacieux », ou pour les termes qui commencent par le préfixe « air condition ». Le paramètre `searchMode` est utilisé pour mettre en correspondance n'importe quel terme (valeur par défaut) ou la totalité d'entre eux, pour les cas où un terme n'est pas explicitement requis (+).

3. Trie l'ensemble d'hôtels résultant en fonction de leur proximité par rapport à un emplacement géographique donné, puis les renvoie à l'application appelante.

La majeure partie de cet article porte sur le traitement de la *requête de recherche*:

`"Spacious, air-condition* +"Ocean view"`. Le filtrage et le tri ne sont pas abordés. Pour plus d'informations, voir [Documentation de référence sur l'API de recherche](#).

## Étape 1 : Analyse des requêtes

Comme indiqué, la chaîne de requête constitue la première ligne de la requête :

```
"search": "Spacious, air-condition* +"Ocean view\"",
```

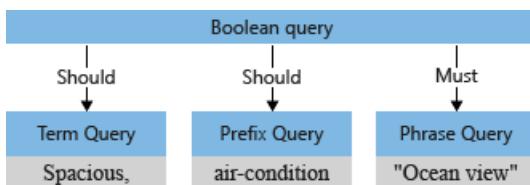
L'analyseur de requêtes distingue les opérateurs (tels que \* et + dans l'exemple) des termes de recherche et décompose la requête de recherche en *sous-requêtes* d'un type pris en charge :

- *requête de terme* pour les termes autonomes (comme spacieux)
- *requête d'expression* pour les termes entre guillemets (comme vue mer)
- *requête de préfixe* pour les termes suivis d'un opérateur de préfixe \* (comme air condition)

Pour obtenir la liste complète des types de requêtes pris en charge, consultez [Syntaxe de requête Lucene](#)

Les opérateurs associés à une sous-requête déterminent si la requête « doit être » ou « peut être » satisfaite pour qu'un document soit considéré comme correspondant. Par exemple, +"Ocean view" indique que la requête « doit être » satisfaite en raison de l'opérateur +.

L'analyseur de requêtes restructure les sous-requêtes en une *arborescence de requête* (structure interne représentant la requête) qu'il transmet au moteur de recherche. Dans la première étape d'analyse de la requête, l'arborescence de requête ressemble à ceci.



### Analyseurs pris en charge : Lucene simple et complet

La Recherche cognitive Azure expose deux langages de requête différents, `simple` (valeur par défaut) et `full`. En définissant le paramètre `queryType` avec votre requête de recherche, vous indiquez à l'analyseur de requêtes le langage de requête choisi afin qu'il sache comment interpréter les opérateurs et la syntaxe. Le [langage de requête simple](#) est intuitif et robuste, généralement adapté à l'interprétation de l'entrée d'utilisateur telle quelle, sans traitement côté client. Il prend en charge les opérateurs de requête courants des moteurs de recherche web. Le [langage de requête complet Lucene](#), que vous pouvez obtenir en définissant `queryType=full`, étend le langage de requête simple par défaut en y ajoutant la prise en charge de plusieurs opérateurs et types de requête, tels que les caractères génériques, et les requêtes partielles, d'expression régulière et portant sur des champs. Par exemple, une expression régulière envoyée en syntaxe de requête simple serait interprétée en tant que chaîne de requête et pas en tant qu'expression. L'exemple de requête de cet article utilise le langage de requête complet Lucene.

## Impact du mode de recherche sur l'Analyseur

Un autre paramètre de requête de recherche susceptible d'affecter l'analyse est le paramètre `searchMode`. Il contrôle l'opérateur par défaut pour les requêtes booléennes : quelconque (valeur par défaut) ou tout.

Lorsque `searchMode=any` (valeur par défaut), le délimiteur d'espace entre spacieux et air condition est OU (||), rendant le texte d'exemple de requête équivalent à :

```
Spacious, ||air-condition*"Ocean view"
```

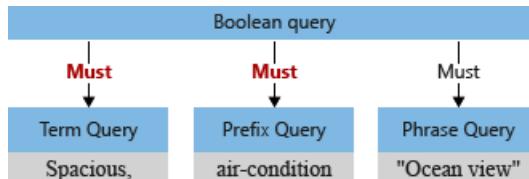
Les opérateurs explicites, tels que + dans +"Ocean view", ne sont pas équivoques dans la construction de requête booléenne (le terme *doit* correspondre). Il est moins évident d'interpréter les termes restants : spacieux et air condition. Le moteur doit-il rechercher les correspondances avec vue mer *et* spacieux *et* air condition ? Ou doit-il rechercher les correspondances avec vue mer *et* *l'un des* termes restants ?

Par défaut (`searchMode=any`), le moteur de recherche choisit l'interprétation la plus large. N'importe quel champ *doit* être mis en correspondance, reflétant la sémantique du « ou ». L'arborescence de requête initiale décrite précédemment, avec les deux opérations « doit », indique la valeur par défaut.

Supposons que nous avons maintenant défini `searchMode=all`. Dans ce cas, l'espace est interprété comme une opération « et ». Chacun des termes restants doit être présent dans le document pour être considéré comme une correspondance. L'exemple de requête résultant serait interprété comme suit :

```
+Spacious,+air-condition*"Ocean view"
```

Une arborescence de requête modifiée pour cette requête se présente comme suit, où un document correspondant représente l'intersection des trois sous-requêtes :



### NOTE

Choisir `searchMode=any` plutôt que `searchMode=all` est le meilleur moyen d'exécuter des requêtes représentatives. Les utilisateurs susceptibles d'inclure des opérateurs (fréquent lors de la recherche de magasins de documents) peuvent obtenir des résultats plus intuitifs si `searchMode=all` informe les constructions de requête booléenne. Pour plus d'informations sur l'interaction entre `searchMode` et les opérateurs, voir [Syntaxe de requête simple](#).

## Étape 2 : Analyse lexicale

Les analyseurs lexicaux traitent les *requêtes de terme* et les *requêtes d'expression* une fois l'arborescence de requête structurée. Un analyseur accepte les entrées de texte transmises par l'analyseur, traite le texte, puis renvoie les termes tokénisés à incorporer dans l'arborescence de requête.

La forme la plus courante d'analyse lexicale est *l'analyse linguistique* qui transforme les termes de requête en fonction des règles propres à un langage donné :

- Réduire un terme de requête à la racine d'un mot
- Supprimer les mots non essentiels (mots vides, tels que « le », « la », « les » ou « et » en français)
- Diviser un mot composite en composants
- Convertir en minuscules un mot en majuscules

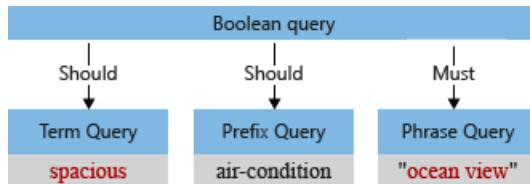
Toutes ces opérations ont tendance à gommer les différences entre l'entrée de texte fournie par l'utilisateur et les termes stockés dans l'index. Ces opérations vont au-delà du traitement de texte et nécessitent une connaissance approfondie du langage lui-même. Pour ajouter cette couche de sensibilisation linguistique, la Recherche cognitive Azure prend en charge une longue liste d'[analyseurs de langage](#) Lucene et Microsoft.

#### NOTE

Les exigences d'analyse peuvent être minimales ou élaborées, selon votre scénario. Vous pouvez contrôler la complexité de l'analyse lexicale en sélectionnant l'un des analyseurs prédefinis ou en créant votre propre [analyseur personnalisé](#). Les analyseurs sont limités aux champs pouvant faire l'objet d'une recherche et sont spécifiés dans le cadre d'une définition de champ. Cela vous permet de faire varier l'analyse lexicale en fonction du champ. L'analyseur Lucene *standard* est utilisé si aucun autre analyseur n'est spécifié.

Dans notre exemple, avant l'analyse, l'arborescence de requête initiale contient le terme « Spacieux, », avec un « S » majuscule et une virgule que l'analyseur de requêtes interprète comme faisant partie du terme de requête (une virgule n'est pas considérée comme un opérateur de langage de requête).

Lorsque l'analyseur par défaut traite le terme, « vue mer » et « spacieux » sont en minuscules et la virgule est supprimée. L'arborescence de requête modifiée se présente comme suit :



#### Test des comportements de l'analyseur

Le comportement d'un analyseur peut être testé à l'aide de l'[API Analyser](#). Saisissez le texte que vous souhaitez analyser pour voir les termes qu'un analyseur donné génère. Par exemple, pour voir comment l'analyseur standard traite le texte « air condition », vous pouvez émettre la requête suivante :

```
{  
  "text": "air-condition",  
  "analyzer": "standard"  
}
```

L'analyseur standard fractionne le texte d'entrée en deux jetons, les annotant avec des attributs tels que les décalages de début et de fin (pour la mise en surbrillance des correspondances), et annotant également leur position (pour la correspondance d'expression) :

```
{  
  "tokens": [  
    {  
      "token": "air",  
      "startOffset": 0,  
      "endOffset": 3,  
      "position": 0  
    },  
    {  
      "token": "condition",  
      "startOffset": 4,  
      "endOffset": 13,  
      "position": 1  
    }  
  ]  
}
```

## Exceptions à l'analyse lexicale

L'analyse lexicale s'applique uniquement aux types de requêtes qui nécessitent des termes complets (requête de terme ou requête d'expression). Elle ne s'applique pas aux types de requête avec des termes incomplets : requête de préfixe, requête de caractère générique, requête d'expression régulière ou requête partielle. Ces types de requête, y compris la requête de préfixe avec le terme `air-condition*` dans notre exemple, sont ajoutés directement à l'arborescence de requête, en ignorant la phase d'analyse. La seule transformation effectuée sur les termes de requête de ce type est l'utilisation de minuscules.

## Étape 3 : Extraction de documents

L'extraction de documents fait référence à la recherche de documents avec des termes correspondants dans l'index. Vous comprendrez mieux cette étape à l'aide d'un exemple. Commençons par un index des hôtels dont le schéma simple est le suivant :

```
{  
    "name": "hotels",  
    "fields": [  
        { "name": "id", "type": "Edm.String", "key": true, "searchable": false },  
        { "name": "title", "type": "Edm.String", "searchable": true },  
        { "name": "description", "type": "Edm.String", "searchable": true }  
    ]  
}
```

Supposons également que cet index contient les quatre documents suivants :

```
{  
    "value": [  
        {  
            "id": "1",  
            "title": "Hotel Atman",  
            "description": "Spacious rooms, ocean view, walking distance to the beach."  
        },  
        {  
            "id": "2",  
            "title": "Beach Resort",  
            "description": "Located on the north shore of the island of Kaua'i. Ocean view."  
        },  
        {  
            "id": "3",  
            "title": "Playa Hotel",  
            "description": "Comfortable, air-conditioned rooms with ocean view."  
        },  
        {  
            "id": "4",  
            "title": "Ocean Retreat",  
            "description": "Quiet and secluded"  
        }  
    ]  
}
```

### Comment les termes sont indexés

Pour comprendre l'extraction, il est utile de connaître quelques notions de base sur l'indexation. L'unité de stockage est un index inversé, un pour chaque champ pouvant faire l'objet d'une recherche. Un index inversé comprend la liste triée de tous les termes issus de tous les documents. Chaque terme correspond à la liste des documents dans laquelle il se trouve, comme le montre l'exemple ci-dessous.

Pour produire les termes d'un index inversé, le moteur de recherche effectue une analyse lexicale sur le contenu des documents, de façon similaire à ce qui se produit pendant le traitement des requêtes :

1. Les *entrées de texte* sont transmises à un analyseur, en minuscules, débarrassées des signes de ponctuation et ainsi de suite, en fonction de la configuration de l'analyseur.
2. Les *jetons* sont le résultat de l'analyse lexicale.
3. Les *termes* sont ajoutés à l'index.

Il est commun, mais pas obligatoire, d'utiliser les mêmes analyseurs pour les opérations de recherche et d'indexation afin que les termes de requête ressemblent davantage aux termes de l'index.

#### NOTE

La Recherche cognitive Azure vous permet de spécifier différents analyseurs pour l'indexation et la recherche via les paramètres de champ supplémentaires `indexAnalyzer` et `searchAnalyzer`. Par défaut, l'analyseur défini avec la propriété `analyzer` est utilisé pour l'indexation et la recherche.

### Index inversé pour les documents d'exemple

Dans notre exemple, pour le champ **titre**, l'index inversé ressemble à ceci :

TERME	LISTE DE DOCUMENTS
atman	1
plage	2
hôtel	1, 3
mer	4
playa	3
complexe	3
retraite	4

Dans le champ Titre, seul *hôtel* apparaît dans deux documents : 1, 3.

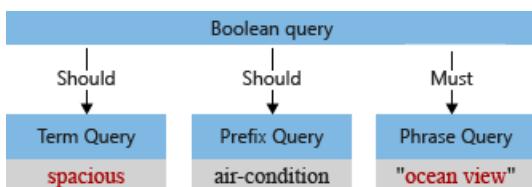
Pour le champ **Description**, l'index est le suivant :

TERME	LISTE DE DOCUMENTS
air	3
and	4
plage	1
conditionné	3
confortable	3
distance	1
île	2

TERME	LISTE DE DOCUMENTS
kauai	2
situé	2
Nord	2
mer	1, 2, 3
sur	2
sur	2
calme	4
chambres	1, 3
isolé	4
rive	2
spacieux	1
le	1, 2
to	1
vue	1, 2, 3
marche	1
par	3

### Termes de requête correspondant aux termes indexés

Étant donné les index inversés ci-dessus, nous allons revenir à l'exemple de requête et voir comment les documents correspondants sont trouvés pour notre exemple. L'arborescence de requête finale ressemble à ceci :



Pendant l'exécution de la requête, les requêtes individuelles sont exécutées indépendamment sur les champs pouvant faire l'objet d'une recherche.

- La requête TermQuery « spacious » correspond au document 1 (hôtel Atman).
- La requête PrefixQuery « air condition » ne correspond à aucun document.

Ce comportement déroute parfois les développeurs. Bien que le terme air condition existe dans le document, il est divisé en deux termes par l'analyseur par défaut. N'oubliez pas que les requêtes de

préfixe, qui contiennent des termes partiels, ne sont pas analysées. Par conséquent, les termes avec le préfixe « air condition » sont recherchés dans l'index inversé, mais demeurent introuvables.

- La requête PhraseQuery, « vue mer », recherche les termes « mer » et « vue » et vérifie la proximité des termes dans le document d'origine. Les documents 1, 2 et 3 correspondent à cette requête dans le champ Description. Le document 4 contient le terme mer dans le titre, mais n'est pas considéré comme une correspondance, puisque nous recherchons l'expression « vue mer » plutôt que des mots individuels.

#### NOTE

Une requête de recherche est exécutée indépendamment sur tous les champs pouvant faire l'objet d'une recherche dans l'index de Recherche cognitive Azure, sauf si vous limitez les champs définis avec le paramètre `searchFields`, comme illustré dans l'exemple de requête de recherche. Les documents qui correspondent à l'un des champs sélectionnés sont renvoyés.

Dans l'ensemble, pour la requête en question, les documents qui correspondent sont les documents 1, 2, 3.

## Étape 4 : Notation

Un score de pertinence est attribué à chaque document d'un jeu de résultats de recherche. La fonction du score de pertinence est d'accorder une place prioritaire aux documents qui répondent le mieux à une question de l'utilisateur exprimée par la requête de recherche. Le score est calculé en fonction des propriétés statistiques des termes qui correspondent. La base de la formule de notation est appelée [TF-IDF \(Term Frequency-Inverse Document Frequency, fréquence de terme-fréquence inverse de document\)](#). Dans les requêtes qui contiennent des termes rares et courants, TF/IDF promeut les résultats contenant le terme rare. Par exemple, dans un index hypothétique de tous les articles Wikipedia, à partir de documents correspondant à la requête *le président*, les documents correspondant au terme *président* sont considérés comme plus pertinents que les documents correspondant au terme */e*.

### Exemple de notation

Souvenez-vous des trois documents correspondant à notre exemple de requête :

```
search=Spacious, air-condition* +"Ocean view"
```

```
{
  "value": [
    {
      "@search.score": 0.25610128,
      "id": "1",
      "title": "Hotel Atman",
      "description": "Spacious rooms, ocean view, walking distance to the beach."
    },
    {
      "@search.score": 0.08951007,
      "id": "3",
      "title": "Playa Hotel",
      "description": "Comfortable, air-conditioned rooms with ocean view."
    },
    {
      "@search.score": 0.05967338,
      "id": "2",
      "title": "Ocean Resort",
      "description": "Located on a cliff on the north shore of the island of Kauai. Ocean view."
    }
  ]
}
```

Le document 1 correspond mieux à la requête, car le terme *spacieux* et l'expression requise *vue mer* se trouvent tous les deux dans le champ Description. Les documents 2 et 3 correspondent uniquement à l'expression *vue mer*. Il peut être surprenant que le score de pertinence des documents 2 et 3 soit différent, bien qu'ils correspondent à la requête de la même façon. Cela signifie que la formule de notation a plus de composants que la formule TF/IDF. Dans ce cas, un score légèrement plus élevé a été affecté au document 3, car sa description est plus courte. En savoir plus sur la [formule de notation pratique Lucene](#) pour comprendre comment la longueur de champ et d'autres facteurs peuvent influencer le score de pertinence.

Certains types de requête (caractère générique, préfixe, expression régulière) contribuent toujours à un score constant dans le score général du document. Ainsi, les correspondances trouvées par le biais de l'extension de requête sont incluses dans les résultats, sans affecter le classement.

L'exemple suivant illustre l'importance de ce facteur. Les recherches avec caractères génériques, y compris les recherches de préfixe, sont ambiguës par définition, car l'entrée est une chaîne partielle avec des correspondances potentielles sur un très grand nombre de termes disparates (imaginez l'entrée « tour\* » avec des correspondances trouvées sur « tours », « tourettes » et « tourmaline »). Étant donné la nature de ces résultats, il est impossible de déduire raisonnablement quels termes sont plus utiles que d'autres. Pour cette raison, nous ignorons les fréquences de terme lors de la notation des résultats dans les requêtes de types génériques, de préfixe et d'expression régulière. Dans une requête de recherche à parties multiples qui comprend des termes partiels et complets, les résultats de l'entrée partielle sont incorporés à un score constant pour éviter le biais vers les correspondances potentiellement inattendues.

## Paramétrage du score

Il existe deux façons de régler les scores de pertinence dans la Recherche cognitive Azure :

1. **Les profils de score** promeuvent les documents dans la liste ordonnée des résultats en fonction d'un ensemble de règles. Dans notre exemple, nous pourrions considérer que les documents correspondant au champ Titre sont plus pertinents que les documents correspondant au champ Description. En outre, si notre index comporte un champ Prix pour chaque hôtel, nous aurions pu promouvoir les documents avec un prix inférieur. Pour plus d'informations, voir [Ajouter des profils de notation à un index de recherche](#).
2. **La promotion de termes** (disponible uniquement dans la syntaxe de requête complète Lucene) fournit un opérateur de promotion `^` qui peut être appliqué à d'autres parties de l'arborescence de requête. Dans notre exemple, au lieu de rechercher le préfixe *air condition\**, on peut rechercher le terme exact *air condition* ou le préfixe, mais les documents qui correspondent au terme exact sont mieux classés lorsqu'on applique la promotion à la requête de terme : *air condition^2||air condition\**. En savoir plus sur la

promotion de termes.

### Notation dans un index distribué

Tous les index dans la Recherche cognitive Azure sont automatiquement divisés en plusieurs partitions, ce qui nous permet de distribuer rapidement l'index entre plusieurs nœuds pendant la mise à l'échelle du service (inférieure ou supérieure). Lorsqu'une requête de recherche est émise, elle est émise sur chaque partition indépendamment. Les résultats de chaque partition sont ensuite fusionnés et classés par notation (si aucun autre classement n'est défini). Il est important de savoir que la fonction de notation compare la fréquence de terme de requête à la fréquence inverse de documents dans tous les documents de la partition, et pas dans toutes les partitions.

Cela signifie qu'un score de pertinence *peut* être différent pour des documents identiques s'ils résident sur différentes partitions. Heureusement, ces différences ont tendance à disparaître à mesure que le nombre de documents dans l'index augmente et ce, grâce à une distribution des termes plus homogène. Il est impossible de deviner sur quelle partition un document sera placé. Toutefois, si la clé de document ne change pas, celui-ci sera toujours affecté à la même partition.

En général, le score du document n'est pas le meilleur attribut pour classer les documents si la stabilité du classement est importante. Prenons par exemple deux documents avec un score identique. Il est impossible de déterminer lequel apparaîtra en premier dans les exécutions suivantes de la même requête. Le score du document doit uniquement fournir une idée générale de la pertinence du document par rapport à d'autres documents dans le jeu de résultats.

## Conclusion

La réussite des moteurs de recherche Internet a suscité des attentes en termes de recherche en texte intégral sur les données privées. Quelle que soit l'expérience de recherche, nous nous attendons désormais à ce que le moteur comprenne notre intention, même lorsque les termes sont mal orthographiés ou incomplets. Nous allons même jusqu'à attendre des correspondances en fonction de termes équivalents ou synonymes jamais spécifiés.

D'un point de vue technique, la recherche en texte intégral est très complexe, nécessitant une analyse linguistique sophistiquée et une approche systématique du traitement de manière à distiller, développer et transformer les termes de requête pour renvoyer un résultat correspondant. Étant donné la complexité inhérente, de nombreux facteurs peuvent affecter le résultat d'une requête. Pour cette raison, prenez le temps nécessaire pour comprendre le fonctionnement de la recherche en texte intégral si vous essayez d'analyser des résultats inattendus.

Cet article a présenté la recherche en texte intégral dans le contexte de la Recherche cognitive Azure. Nous espérons qu'il vous a donné les bases nécessaires pour connaître les causes potentielles des problèmes les plus courants en matière de requête, ainsi que leurs résolutions.

## Étapes suivantes

- Créez l'index des exemples, essayez différentes requêtes et passez en revue les résultats. Pour obtenir des instructions, consultez la page [Générer et interroger un index dans le portail](#).
- Essayez une syntaxe de requête supplémentaire à partir de la section Exemple [Rechercher des documents](#) ou à partir de la [syntaxe de requête simple](#) dans l'Explorateur de recherche du portail.
- Passez en revue les [profils de score](#) si vous souhaitez paramétrier le classement dans votre application de recherche.
- Découvrez comment appliquer des [analyseurs lexicaux propres au langage](#).
- [Configurez des analyseurs personnalisés](#) pour un traitement minimal ou pour un traitement spécialisé

sur des champs spécifiques.

## Voir aussi

[API REST de recherche de documents](#)

[Syntaxe de requête simple](#)

[Syntaxe de requête complète Lucene](#)

[Traiter les résultats de recherche](#)

# Indexeurs dans Recherche cognitive Azure

04/10/2020 • 15 minutes to read • [Edit Online](#)

Dans Recherche cognitive Azure, un *indexeur* est un analyseur qui extrait les données et métadonnées pouvant faire l'objet d'une recherche d'une source de données Azure externe et renseigne un index en fonction des mappages champ à champ entre l'index et votre source de données. Cette approche est parfois appelée « modèle d'extraction », car le service extrait des données sans que vous ayez à écrire un code qui ajoute des données à un index.

Les indexeurs sont basés sur des types de sources de données ou des plateformes, avec des indexeurs individuels pour SQL Server sur Azure, Cosmos DB, Stockage Table Azure et Stockage Blob. Les indexeurs de stockage d'objets blob ont des propriétés supplémentaires spécifiques aux types de contenu d'objet blob.

Vous pouvez utiliser un indexeur comme seul moyen d'ingestion des données ou utiliser une combinaison de techniques qui incluent l'utilisation d'un indexeur pour charger uniquement certains champs dans l'index.

Vous pouvez exécuter des indexeurs à la demande ou en fonction d'une planification d'actualisation des données périodique qui s'exécute jusqu'à une fois toutes les cinq minutes. Des mises à jour plus fréquentes requièrent un modèle d'émission qui met à jour simultanément les données dans Recherche cognitive Azure et dans votre source de données externe.

## Méthodes de création et de gestion des indexeurs

Vous pouvez créer et gérer des indexeurs en suivant l'une de ces approches :

- [Portail > Assistant Importer des données](#)
- [API REST du service](#)
- [Kit de développement logiciel \(SDK\) .NET](#)

Au départ, un nouvel indexeur est annoncé comme une fonctionnalité d'aperçu. Les fonctionnalités d'aperçu sont introduites dans les API (REST et .NET) et sont ensuite intégrées dans le portail après la promotion vers la disponibilité générale. Lors de l'évaluation d'un indexeur, vous devez envisager d'écrire du code.

## Autorisations

Toutes les opérations liées aux indexeurs, notamment les requêtes GET d'état ou de définitions, nécessitent une [admin api-key](#).

## Sources de données prises en charge

Les indexeurs analysent les magasins de données sur Azure.

- [Stockage Blob Azure](#)
- [Azure Data Lake Storage Gen2](#) (en préversion)
- [Stockage de tables Azure](#)
- [Azure Cosmos DB](#)
- [Azure SQL Database](#)
- [SQL Managed Instance](#)

- SQL Server sur les machines virtuelles Azure

## Étapes de l'indexeur

Lors d'une exécution initiale, lorsque l'index est vide, un indexeur lit toutes les données fournies dans la table ou le conteneur. Lors des exécutions suivantes, l'indexeur peut généralement détecter et récupérer uniquement les données qui ont été modifiées. Pour les données blob, la détection des modifications est automatique. Pour d'autres sources de données comme Azure SQL ou Cosmos DB, la détection des modifications doit être activée.

Pour chacun des documents qu'il reçoit, un indexeur implémente ou coordonne plusieurs étapes, de la récupération du document à un « transfert » vers un moteur de recherche final pour indexation. Si vous le souhaitez, un indexeur peut également opérer de manière instrumentale pour gérer l'exécution d'un ensemble de compétences et des sorties, en supposant qu'un ensemble de compétences est défini.



### Étape 1 : Décodage du document

Le décodage de document est le processus d'ouverture de fichiers et d'extraction du contenu. Selon le type de source de données, l'indexeur essaiera d'effectuer différentes opérations pour extraire le contenu potentiellement indexable.

Exemples :

- Si le document est un enregistrement d'une [source de données Azure SQL](#), l'indexeur extrait chacun des champs de l'enregistrement.
- Si le document est un fichier PDF d'une [source de données Stockage Blob Azure](#), l'indexeur extrait le texte, les images et les métadonnées du fichier.
- Si le document est un enregistrement d'une [source de données Cosmos DB](#), l'indexeur extrait les champs et les sous-champs du document Cosmos DB.

### Étape 2 : Mappages de champs

Un indexeur extrait le texte d'un champ source et l'envoie à un champ de destination dans un index ou une base de connaissances. Si les noms et les types de champs coïncident, le chemin d'accès est validé. Toutefois, vous pouvez utiliser des noms ou des types différents dans la sortie, auquel cas vous devez indiquer à l'indexeur comment mapper le champ. Cette étape survient après le décodage du document, mais avant les transformations, lorsque l'indexeur lit les données des documents sources. Lorsque vous définissez un [mappage de champs](#), la valeur du champ source est envoyée telle quelle au champ de destination, sans aucune modification. Les mappages de champs sont facultatifs.

### Étape 3 : Exécution d'un ensemble de compétences

L'exécution d'un ensemble de compétences est une étape facultative qui appelle un traitement IA intégré ou personnalisé. Vous pouvez en avoir besoin pour la reconnaissance optique de caractères (OCR) sous la forme d'une analyse d'images, ou dans le cas d'une traduction linguistique. Quelle que soit la transformation, l'exécution d'un ensemble de compétences est l'endroit où l'enrichissement se produit. Si un indexeur est un pipeline, vous pouvez considérer un [ensemble de compétences](#) comme un « pipeline dans le pipeline ». Un ensemble de compétences possède sa propre séquence d'étapes appelée « compétences ».

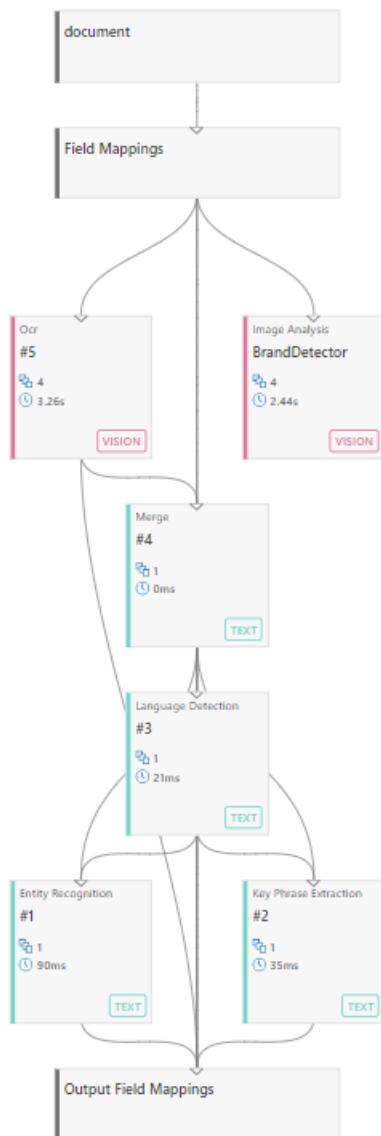
### Étape 4 : Mappages de champs de sortie

La sortie d'un ensemble de compétences est en fait une arborescence d'informations appelée

« document enrichi ». Les mappages de champs de sortie vous permettent de sélectionner les parties de cette arborescence à mapper dans les champs de votre index. Découvrez comment [définir des mappages de champs de sortie](#).

Tout comme les mappages de champs qui associent les valeurs verbatim des champs sources aux champs de destination, les mappages de champs de sortie indiquent à l'indexeur comment associer les valeurs transformées dans le document enrichi aux champs de destination dans l'index. Contrairement aux mappages de champs, qui sont considérés comme facultatifs, vous devrez toujours définir un mappage de champs de sortie pour tout contenu transformé qui figurera dans un index.

L'image suivante montre un exemple de [session de débogage](#) représentant les étapes de l'indexeur : le décodage de document, les mappages de champs, l'exécution d'un ensemble de compétences, et les mappages de champs de sortie.



## Étapes de configuration de base

Les indexeurs peuvent offrir des fonctionnalités propres à la source de données. À cet égard, certains aspects de la configuration de l'indexeur ou de la source de données varient en fonction du type d'indexeur. Cependant, tous les indexeurs présentent une composition et des exigences de base identiques. Les étapes communes à tous les indexeurs sont décrites ci-dessous.

### Étape 1 : Création d'une source de données

Un indexeur obtient une connexion de source de données à partir d'un objet *source de données*. La définition de source de données fournit une chaîne de connexion et éventuellement des informations d'identification. Appelez l'API REST de [création de source de données](#) ou la [classe DataSource](#) pour créer la ressource.

Les sources de données sont configurées et gérées indépendamment des indexeurs qui les utilisent. Autrement dit, une source de données peut être utilisée par plusieurs indexeurs pour charger plusieurs index à la fois.

### Étape 2 : Création d'un index

Un indexeur automatise certaines tâches liées à l'ingestion des données, mais la création d'un index n'en fait généralement pas partie. Au préalable, vous devez disposer d'un index prédéfini présentant des champs qui correspondent à ceux de votre source de données externe. Les champs doivent correspondre par nom et type de données. Pour plus d'informations sur la structuration d'un index, consultez l'article [Create an Index \(Azure Search REST API\)](#)(Création d'un index (API REST Recherche cognitive Azure)) ou [Index class](#) (Classe Index). Pour plus d'informations sur les associations de champ, consultez [Mappages de champs dans les indexeurs de Recherche cognitive Azure](#).

#### TIP

Bien que les indexeurs ne puissent pas générer d'index pour vous, l'Assistant **Importation des données** du portail peut vous aider. Dans la plupart des cas, l'Assistant peut déduire un schéma d'index à partir des métadonnées existantes dans la source, en présentant un schéma d'index préliminaire que vous pouvez modifier en ligne pendant que l'Assistant est actif. Une fois que l'index est créé sur le service, les modifications supplémentaires dans le portail sont principalement limitées à l'ajout de nouveaux champs. Pensez à utiliser l'Assistant pour créer un index (mais pas pour le réviser). Pour mettre vos connaissances en pratique, parcourez la [procédure pas à pas dans le portail](#).

### Étape 3 : Créez et planifiez l'indexeur

La définition de l'indexeur est une construction qui rassemble tous les éléments liés à l'ingestion des données. Les éléments obligatoires incluent une source de données et un index. Les éléments facultatifs incluent une planification et des mappages de champs. Le mappage de champs n'est facultatif que si les champs source et les champs d'index correspondent parfaitement. Pour plus d'informations sur la structuration d'un indexeur, consultez l'article [Create Indexer \(Azure Search REST API\)](#)(Création d'un indexeur (API REST Recherche cognitive Azure)).

## Exécutez des indexeurs à la demande

Bien qu'il soit courant de planifier l'indexation, un indexeur peut également être appelé à la demande à l'aide de la [commande Exécuter](#) :

```
POST https://[service name].search.windows.net/indexers/[indexer name]/run?api-version=2020-06-30  
api-key: [Search service admin key]
```

#### NOTE

Lors de l'API s'exécute avec succès, l'appel de l'indexeur a été planifié, mais le traitement réel se produit de façon asynchrone.

Vous pouvez surveiller l'état de l'indexeur dans le portail ou à l'aide de l'API Get Indexer Status.

## Get indexer status

Vous pouvez récupérer l'état et l'historique d'exécution d'un indexeur à l'aide de la [commande Get Indexer Status](#) :

```
GET https://[service name].search.windows.net/indexers/[indexer name]/status?api-version=2020-06-30  
api-key: [Search service admin key]
```

La réponse contient l'état d'intégrité global de l'indexeur, le dernier appel de l'indexeur (ou celui en cours), ainsi que l'historique des appels récents de l'indexeur.

```
{
    "status": "running",
    "lastResult": {
        "status": "success",
        "errorMessage": null,
        "startTime": "2018-11-26T03:37:18.853Z",
        "endTime": "2018-11-26T03:37:19.012Z",
        "errors": [],
        "itemsProcessed": 11,
        "itemsFailed": 0,
        "initialTrackingState": null,
        "finalTrackingState": null
    },
    "executionHistory": [
        {
            "status": "success",
            "errorMessage": null,
            "startTime": "2018-11-26T03:37:18.853Z",
            "endTime": "2018-11-26T03:37:19.012Z",
            "errors": [],
            "itemsProcessed": 11,
            "itemsFailed": 0,
            "initialTrackingState": null,
            "finalTrackingState": null
        }
    ]
}
```

L'historique d'exécution contient les 50 exécutions les plus récentes, classées par ordre chronologique inverse (la dernière exécution est répertoriée en premier dans la réponse).

## Étapes suivantes

Maintenant que vous avez la structure de base, l'étape suivante consiste à passer en revue les exigences et les tâches propres à chaque type de source de données.

- [Azure SQL Database, SQL Managed Instance ou SQL Server sur une machine virtuelle Azure](#)
- [Azure Cosmos DB](#)
- [Stockage Blob Azure](#)
- [Stockage de tables Azure](#)
- [Indexation d'objets blob CSV avec l'indexeur d'objets blob Recherche cognitive Azure](#)
- [Indexation d'objets blob JSON avec l'indexeur d'objets blob Recherche cognitive Azure](#)

# Enrichissement de l'IA dans Recherche cognitive Azure

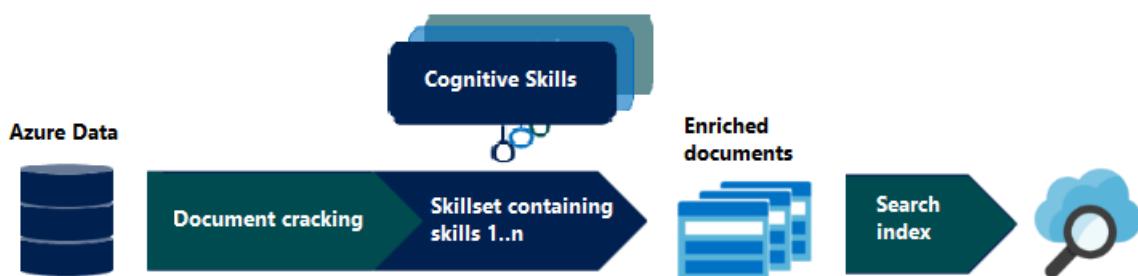
04/10/2020 • 16 minutes to read • [Edit Online](#)

L'enrichissement par IA est une extension des [indexeurs](#) qui peut être utilisée pour extraire du texte à partir d'images, de blobs et d'autres sources de données non structurées. L'enrichissement et l'extraction facilitent les recherches de votre contenu dans les objets de sortie de l'indexeur, un [index de recherche](#) ou une [base de connaissances](#).

L'extraction et l'enrichissement sont implémentés à l'aide de *compétences cognitives* rattachées au pipeline piloté par l'indexeur. Vous pouvez utiliser les compétences intégrées de Microsoft ou incorporer un traitement externe dans une [compétence personnalisée](#) que vous créez. Un module d'entité ou un classifieur de documents ciblant un domaine spécifique comme la finance, les publications scientifiques ou la médecine sont des exemples de compétence personnalisée.

Les compétences intégrées se répartissent en fonction des catégories suivantes :

- Les compétences de **traitement en langage naturel** incluent la [reconnaissance d'entité](#), la [détection de la langue](#), l'[extraction de phrases clés](#), la manipulation de texte, la [détection de sentiments](#) et la [détection d'informations d'identification personnelle](#). Grâce à ces compétences, un texte non structuré est mappé sous la forme de champs pouvant être interrogés et filtrés dans un index.
- Les compétences de **traitement d'image** incluent la [reconnaissance optique de caractères \(OCR\)](#) et l'[identification des caractéristiques visuelles](#), comme la détection des visages, l'interprétation des images, la reconnaissance des images (monuments et personnes célèbres) ou des attributs tels que l'orientation des images. Ces compétences créent des représentations textuelles du contenu des images, ce qui rend les recherches possibles grâce aux capacités d'interrogation de Recherche cognitive Azure.



Les compétences intégrées de la Recherche cognitive Azure sont basées sur les modèles Machine Learning préentraînés des API Cognitive Services : [Vision par ordinateur](#) et [Analyse de texte](#). Vous pouvez attacher une ressource Cognitive Services si vous souhaitez tirer parti de ces ressources lors du traitement du contenu.

Le traitement en langage naturel et le traitement des images sont appliqués durant la phase d'ingestion des données. Les résultats sont intégrés à la composition d'un document au sein d'un index pouvant faire l'objet de recherches dans la Recherche cognitive Azure. Les données sont fournies en tant que jeu de données Azure, puis transmises via un pipeline d'indexation à l'aide des [compétences intégrées](#) dont vous avez besoin.

## Quand utiliser l'enrichissement par IA

Vous devez envisager d'utiliser des compétences cognitives intégrées si votre contenu brut est du texte non structuré, du contenu d'image ou du contenu qui nécessite la détection de la langue et la traduction. L'application de l'intelligence artificielle via les compétences cognitives intégrées peut déverrouiller ce contenu, en renforçant sa valeur et son utilité dans vos applications de science des données et de recherche.

En outre, vous pouvez envisager d'ajouter une compétence personnalisée si vous avez du code open source, tiers ou interne que vous souhaitez intégrer dans le pipeline. Les modèles de classification qui identifient les caractéristiques importantes de différents types de documents appartiennent à cette catégorie, mais tout package qui ajoute de la valeur à votre contenu peut être utilisé.

### En savoir plus sur les compétences intégrées

Un [ensemble de compétences](#) assemblé à l'aide de compétences intégrées convient parfaitement aux scénarios d'application suivants :

- Documents numérisés (JPEG) dans lesquels effectuer une recherche en texte intégral. Vous pouvez associer une compétence de reconnaissance optique de caractères (OCR) pour identifier, extraire et ingérer du texte à partir de fichiers JPEG.
- PDF avec image et texte combinés. Le texte des fichiers PDF peut être extrait durant l'indexation sans recourir aux étapes d'enrichissement. Toutefois, l'ajout du traitement des images et du traitement en langage naturel permet souvent de produire un résultat supérieur à celui fourni par une indexation standard.
- Contenu multilingue sur lequel vous souhaitez appliquer la détection de langue et éventuellement la traduction de texte.
- Documents non structurés ou semi-structurés contenant du contenu qui a une signification ou un contexte intrinsèquement masqué dans le document plus volumineux.

En particulier, les blobs contiennent souvent un grand corps de contenu qui est compressé dans un « champ » unique. En associant des compétences en traitement des images et en langage naturel à un indexeur, vous pouvez créer de nouvelles informations qui sont existantes dans le contenu brut, mais qui ne sont pas affichées en tant que champs distincts. Voici quelques compétences cognitives intégrées prêtes à l'emploi qui peuvent aider : l'extraction d'expressions clés, l'analyse des sentiments et la reconnaissance d'entité (personnes, organisations et lieux).

De plus, les compétences prédéfinies peuvent également être utilisées pour restructurer du contenu via des opérations de découpage, de fusion et de mise en forme du texte.

### En savoir plus sur les compétences personnalisées

Les compétences personnalisées peuvent prendre en charge des scénarios plus complexes, tels que la reconnaissance de formulaires ou la détection d'entité personnalisée à l'aide d'un modèle que vous fournissez et encapsulez dans l'[interface web des compétences personnalisées](#). Plusieurs exemples de compétences personnalisées incluent [Form Recognizer](#), l'intégration de l'[API Recherche d'entités Bing](#) et la [reconnaissance d'entité personnalisée](#).

## Étapes d'un pipeline d'enrichissement

Un pipeline d'enrichissement est basé sur des [indexeurs](#). Les indexeurs remplissent un index basé sur des mappages champ à champ entre l'index et votre source de données pour la craquage de documents. Les compétences, désormais rattachées aux indexeurs, interceptent et enrichissent les documents en fonction des ensembles de compétences que vous définissez. Une fois l'indexation effectuée, vous pouvez accéder au contenu via des requêtes de recherche ainsi qu'à l'aide de tous les [types de requête pris en charge par la Recherche cognitive Azure](#). Si vous ne connaissez pas les indexeurs, cette section vous guide tout au long des étapes.

## Étape 1 : Phase de connexion et de décodage du document

Au début du pipeline, vous avez du texte non structuré ou du contenu non textuel (comme des images, des documents numérisés ou des fichiers JPEG). Les données doivent se trouver dans un service de stockage de données Azure accessible par un indexeur. Les indexeurs peuvent décoder les documents sources pour extraire le texte à partir des données sources. Le craquage de document désigne le processus d'extraction ou de création du contenu textuel à partir de sources non textuelles lors de l'indexation.

### Azure Data



Les sources prises en charge comprennent le stockage Blob Azure, le stockage Table Azure, Azure SQL Database et Azure Cosmos DB. Le contenu textuel peut être extrait des types de fichier suivants : PDF, Word, PowerPoint et CSV. Pour obtenir la liste complète, consultez [Formats de document pris en charge](#). Étant donné que l'indexation prend un certain temps, commencez par un petit ensemble de données représentatif, puis augmentez sa taille de façon incrémentielle à mesure que votre solution grandit.

## Étape 2 : Phase d'enrichissement et compétences cognitives

L'enrichissement se fait grâce aux *compétences cognitives* effectuant des opérations atomiques. Par exemple, une fois que vous avez craqué un fichier PDF, vous pouvez appliquer la reconnaissance d'entité, la détection de la langue ou l'extraction de phrases clés pour créer dans votre index des champs qui ne sont pas disponibles nativement dans la source. La collection de compétences utilisée dans votre pipeline est appelée *ensemble de compétences*.



Un ensemble de compétences est basé sur des [compétences cognitives intégrées](#) ou des [compétences personnalisées](#), que vous fournissez et que vous connectez à l'ensemble de compétences. Un ensemble de compétences peut être minimal ou très complexe, et détermine non seulement le type de traitement, mais également l'ordre des opérations. Un ensemble de compétences plus les mappages de champ définis dans un indexeur spécifient entièrement le pipeline d'enrichissement. Pour plus d'informations sur l'extraction de tous ces éléments ensemble, consultez [How to create a skillset in an enrichment pipeline](#) (Créer un ensemble de compétences dans un pipeline d'enrichissement).

En interne, le pipeline génère une collection de documents enrichis. Vous pouvez décider quelles parties des documents enrichis doivent être mappées aux champs indexables dans votre index de recherche. Par exemple, si vous avez appliqué les compétences de reconnaissance d'entité et d'extraction de phrases clés, ces nouveaux champs vont faire partie du document enrichi et peuvent donc être mappés aux champs de votre index. Consultez [How to reference annotations in a cognitive search skillset](#) (Référencer des annotations dans un ensemble de compétences de recherche cognitive) pour en savoir plus sur les formations entrée/sortie.

### Ajouter un élément knowledgeStore pour enregistrer des enrichissements

L'[API REST du service Recherche \(api-version=2020-06-30\)](#) étend les ensembles de compétences avec une définition `knowledgeStore` qui fournit une connexion de stockage Azure ainsi que des projections décrivant la manière dont les enrichissements sont stockés. Cela s'ajoute à votre index. Dans un pipeline IA standard, les documents enrichis sont temporaires, utilisés uniquement pendant l'indexation, puis ignorés. Avec la base de connaissances, les documents enrichis sont conservés. Pour plus d'informations, consultez [Base de connaissances](#).

### Étape 3 : Accès basé sur des requêtes et index de recherche

À la fin du traitement, vous disposez d'un index de recherche constitué de documents enrichis, qui peut faire l'objet de recherches textuelles dans la Recherche cognitive Azure. [L'interrogation de l'index](#) est la façon dont les développeurs et les utilisateurs accèdent au contenu enrichi généré par le pipeline.



L'index est semblable à tous ceux que vous pouvez créer pour la Recherche cognitive Azure : vous pouvez ajouter des analyseurs personnalisés, appeler des requêtes de recherche partielle, ajouter une recherche filtrée ou expérimenter des profils de scoring afin de remodeler les résultats de la recherche.

Les index sont générés à partir d'un schéma d'index qui définit les champs, les attributs et d'autres constructions jointes à un index spécifique, telles que les profils de score et les cartes de synonymes. Une fois qu'un index est défini et rempli, vous pouvez effectuer des indexations incrémentielles pour sélectionner de nouveaux documents sources et des documents sources mis à jour. Certaines modifications requièrent une régénération complète. Vous devez utiliser un petit jeu de données tant que la conception du schéma n'est pas stable. Pour plus d'informations, consultez [How to rebuild an Azure Search index](#) (Régénérer un index Recherche Azure).

#### Liste de vérification : workflow classique

1. Créez un sous-ensemble de vos données sources dans un échantillon représentatif. Étant donné que l'indexation prend un certain temps, commencez par un petit ensemble de données représentatif, puis augmentez sa taille de façon incrémentielle à mesure que votre solution grandit.
2. Créez un [objet de source de données](#) dans la Recherche cognitive Azure afin de fournir une chaîne de connexion pour l'extraction de données.
3. Créez un [ensemble de compétences](#) avec les étapes d'enrichissement.
4. Définissez le [schéma d'index](#). La collection *Champs* inclut des champs issus des données sources. Vous devez également écraser les champs supplémentaires pour stocker des valeurs générées pour le contenu créé au cours de l'enrichissement.
5. Définissez [l'indexeur](#) faisant référence à la source de données, à l'ensemble de compétences et à l'index.
6. Dans l'indexeur, ajoutez *outputFieldMappings*. Cette section mappe la sortie de l'ensemble de compétences (à l'étape 3) aux champs d'entrées dans le schéma d'index (à l'étape 4).
7. Envoyez la requête *Créer un indexeur* que vous venez de créer (requête POST avec une définition d'indexeur dans le corps de la requête) pour exprimer l'indexeur dans la Recherche cognitive Azure. Cette étape correspond à la façon dont vous exécutez l'indexeur, en appelant le pipeline.
8. Exécutez des requêtes pour évaluer les résultats et modifiez le code pour mettre à jour les ensembles de compétences, le schéma ou la configuration de l'indexeur.
9. [Réinitialisez l'indexeur](#) avant de régénérer le pipeline.

## Étapes suivantes

- [Liens vers la documentation sur l'enrichissement par IA](#)
- [Exemple : Création d'une compétence personnalisée pour l'enrichissement par IA \(C#\)](#)
- [Démarrage rapide : Procédure pas à pas d'enrichissement par IA à partir du portail](#)
- [Tutoriel : En savoir plus sur les API d'enrichissement par IA](#)

- Base de connaissances
- Créer une base de connaissances avec REST
- Conseils de dépannage

# Concepts des ensembles de compétences dans Recherche cognitive Azure

04/10/2020 • 24 minutes to read • [Edit Online](#)

Cet article est destiné aux développeurs qui ont besoin d'une compréhension plus approfondie des concepts et de la composition des compétences. Il suppose une bonne connaissance du processus d'enrichissement par IA. Si vous ne connaissez pas ce concept, commencez par [Enrichissement par IA dans Azure Recherche cognitive](#).

## Présentation des ensembles de compétences

Un ensemble de compétences est une ressource réutilisable dans Recherche cognitive Azure et qui est liée à un indexeur. Cette ressource spécifie une collection de compétences servant à analyser, transformer et enrichir du texte ou des images durant l'indexation. Les compétences ont des entrées et des sorties, et souvent la sortie d'une compétence devient l'entrée d'une autre dans une chaîne ou une séquence de processus.

Chaque ensemble de compétences possède trois propriétés principales :

- `skills`, une collection non triée de compétences dont la séquence d'exécution est déterminée par la plateforme en fonction des entrées requises pour chaque compétence.
- `cognitiveServices`, la clé d'une ressource Cognitive Services qui effectue le traitement des images et du texte pour les compétences qui incluent des compétences intégrées.
- `knowledgeStore` (facultatif), le compte de stockage Azure dans lequel seront projetés vos documents. Les documents enrichis sont également utilisés par les index de recherche.

Les ensembles de compétences sont créés dans JSON. L'exemple suivant est une version légèrement simplifiée de l'[ensemble de compétences hotel-reviews](#) utilisée pour illustrer les concepts de cet article.

Les deux premières compétences sont présentées ci-dessous :

- La compétence 1 est une [compétence de fractionnement de texte](#) qui accepte le contenu du champ « reviews\_text » comme entrée et divise ce contenu en « pages » de 5 000 caractères en sortie.
- La compétence 2 est une [compétence de détection des sentiments](#) qui accepte « pages » comme entrée et génère un nouveau champ appelé « Sentiment » qui contient les résultats de l'analyse des sentiments.

```
{
  "skills": [
    {
      "@odata.type": "#Microsoft.Skills.Text.SplitSkill",
      "name": "#1",
      "description": null,
      "context": "/document/reviews_text",
      "defaultLanguageCode": "en",
      "textSplitMode": "pages",
      "maximumPageLength": 5000,
      "inputs": [
        {
          "name": "text",
          "source": "/document/reviews_text"
        }
      ],
      "outputs": [
        {
          "name": "textItems",
          "targetName": "pages"
        }
      ]
    },
    {
      "@odata.type": "#Microsoft.Skills.Text.SentimentSkill",
      "name": "#2",
      "description": null,
      "context": "/document/reviews_text/pages/*",
      "defaultLanguageCode": "en",
      "inputs": [
        {
          "name": "text",
          "source": "/document/reviews_text/pages/*"
        }
      ],
      "outputs": [
        {
          "name": "score",
          "targetName": "Sentiment"
        }
      ]
    },
    "cognitiveServices": null,
    "knowledgeStore": {}
  }
}
```

#### NOTE

Vous pouvez créer des ensembles de compétences complexes, avec des boucles et des branches, à l'aide de la [compétence conditionnelle](#) pour créer des expressions. La syntaxe se base sur la notation du [pointeur JSON](#) pour les chemins, légèrement modifiée afin d'identifier les nœuds dans l'arborescence d'enrichissements. Un `"/"` fait passer à un niveau inférieur dans l'arborescence et `"*"` est utilisé comme un opérateur for-each dans le contexte. De nombreux exemples de cet article illustrent la syntaxe.

## Arborescence d'enrichissements

Dans la progression des [étapes du pipeline d'enrichissement](#), le traitement du contenu suit la phase de *craquage de document*, au cours de laquelle le texte et les images sont extraits de la source. Le contenu des images peut ensuite être acheminé vers des compétences qui spécifient le traitement des images, tandis que le contenu texte est mis en file d'attente pour le traitement du texte. Pour les documents sources qui contiennent de grandes quantités de texte, vous pouvez définir un *mode d'analyse* sur l'indexeur pour découper le texte en plus petits segments pour un traitement optimal.

Une fois qu'un document se trouve dans le pipeline d'enrichissement, il est représenté sous la forme d'une arborescence du contenu et des enrichissements associés. Cette arborescence est instanciée en tant que sortie du craquage du document. Le format de l'arborescence d'enrichissements permet au pipeline d'enrichissement d'attacher des métadonnées même à des types de données primitifs ; ce n'est pas un objet JSON valide, mais il peut être projeté dans un format JSON valide. Le tableau suivant indique l'état d'un document qui entre dans le pipeline d'enrichissement :

DATA SOURCE\PARSING MODE	DEFAULT	JSON, JSON LINES & CSV
Stockage Blob	/document/content /document/normalized_images/* ...	/document/{key1} /document/{key2} ...
SQL	/document/{column1} /document/{column2} ...	N/A
Cosmos DB	/document/{key1} /document/{key2} ...	N/A

À mesure que les compétences s'exécutent, elles ajoutent de nouveaux nœuds à l'arborescence d'enrichissements. Ces nouveaux nœuds peuvent ensuite être utilisés comme entrées pour les compétences en aval, en les projetant dans la base de connaissances ou en les mappant aux champs d'index. Les enrichissements ne sont pas mutables : une fois créés, les nœuds ne peuvent pas être modifiés. Plus votre ensemble de compétences est complexe, plus votre arborescence d'enrichissements l'est aussi. Toutefois, vous n'avez pas besoin d'inclure systématiquement tous les nœuds de l'arborescence d'enrichissements dans l'index ou la base de connaissances.

Vous pouvez choisir de conserver uniquement une partie des enrichissements dans l'index ou la base de connaissances.

## Context

Chaque compétence demande un contexte. Un contexte détermine :

- Le nombre de fois que la compétence est exécutée, en fonction des nœuds sélectionnés. Pour les valeurs de contexte d'une collection, l'ajout de `/*` à la fin spécifie que la compétence est appelée une fois pour chaque instance dans la collection.
- L'endroit dans l'arborescence d'enrichissements où les sorties de la compétence sont ajoutées. Les sorties sont toujours ajoutées à l'arborescence en tant qu'enfants du nœud de contexte.
- La forme des entrées. Pour les collections à plusieurs niveaux, la définition du contexte sur la collection parente détermine la forme de l'entrée de la compétence. Par exemple, dans une arborescence d'enrichissements avec une liste de pays/régions, chaque entrée est enrichie avec une liste d'États contenant elle-même une liste de codes postaux.

CONTEXT	ENTRÉE	FORME DE L'ENTRÉE	APPEL DE COMPÉTENCE
<code>/document/countries/*</code>	<code>/document/countries/*/states/*</code>	Liste de tous les codes postaux du pays/de la région	Une fois par pays/région
<code>/document/countries/*/states/*</code>	<code>/document/countries/*/states//zipcodes/*``</code>	Liste de tous les codes postaux de l'état	Une fois par combinaison pays/région et État

# Générer des données enrichies

À l'aide de l'[ensemble de compétences hotel-reviews](#) comme point de référence, nous allons examiner les éléments suivants :

- L'arborescence d'enrichissements évolue au fur et à mesure de l'exécution de chaque compétence.
- Le contexte et les entrées déterminent le nombre de fois qu'une compétence s'exécute.
- Le contexte affecte la forme de l'entrée.

Un « document » au sein du processus d'enrichissement est une ligne (un avis sur un hôtel) dans le fichier source `hotel_reviews.csv`.

## Compétence n° 1 : Division

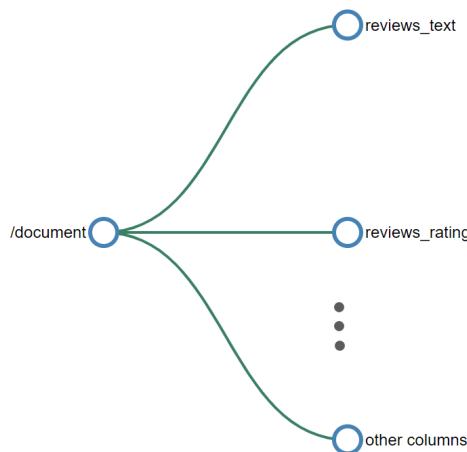
Lorsque le contenu source est constitué de gros blocs de texte, il est utile de le diviser en plus petits composants pour plus de précision dans la détection de la langue, du sentiment et des expressions clés. Deux grains sont disponibles : les pages et les phrases. Une page se compose d'environ 5000 caractères.

La compétence de fractionnement de texte se trouve généralement d'abord dans un ensemble de compétences.

```
"@odata.type": "#Microsoft.Skills.Text.SplitSkill",
"name": "#1",
"description": null,
"context": "/document/reviews_text",
"defaultLanguageCode": "en",
"textSplitMode": "pages",
"maximumPageLength": 5000,
"inputs": [
  {
    "name": "text",
    "source": "/document/reviews_text"
  }
],
"outputs": [
  {
    "name": "textItems",
    "targetName": "pages"
  }
]
```

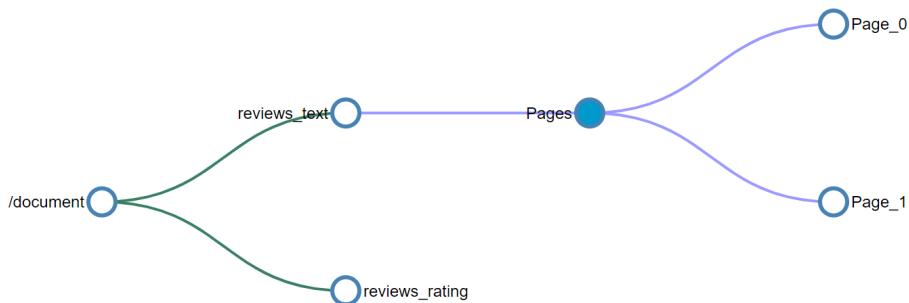
Avec le contexte de compétence `"/document/reviews_text"`, la compétence de fractionnement s'exécute une fois pour `reviews_text`. La sortie de la compétence est une liste où `reviews_text` est segmenté en 5 000 séquences de caractères. La sortie de la compétence de division est nommée `pages` et elle est ajoutée à l'arborescence d'enrichissements. Avec `targetName`, vous pouvez renommer une sortie de compétence avant de l'ajouter à l'arborescence d'enrichissements.

L'arborescence d'enrichissements comporte maintenant un nouveau nœud, situé sous le contexte de la compétence. Ce nœud peut être utilisé pour d'autres compétences, projections ou mappages de champs de sortie. Conceptuellement, l'arborescence ressemble à ce qui suit :



Le nœud racine de tous les enrichissements est `"/document"`. Quand vous utilisez des indexeurs d'objets blob, le nœud `"/document"` contient les nœuds enfants `"/document/content"` et `"/document/normalized_images"`. Si vous utilisez des données CSV, comme dans cet exemple, les noms de colonne sont mappés aux nœuds figurant sous `"/document"`.

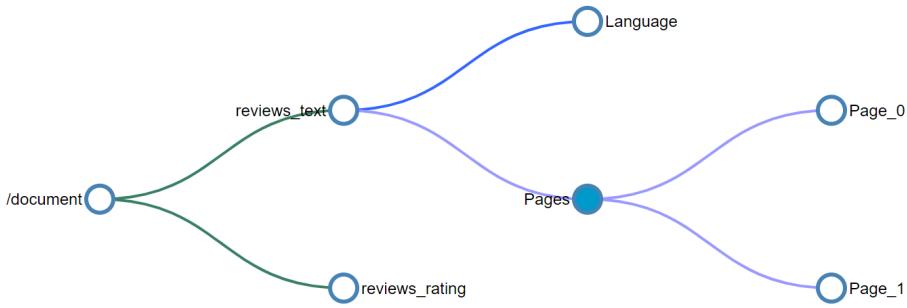
Pour accéder à un enrichissement qui a été ajouté à un nœud par une compétence, vous devez indiquer le chemin complet de l'enrichissement. Par exemple, si vous souhaitez utiliser le texte du nœud `pages` comme entrée dans une autre compétence, vous devez spécifier le chemin de cette façon : `"/document/reviews_text/pages/*"`.



## Compétence n° 2 : Détection de la langue

Les documents d'avis sur les hôtels comprennent des commentaires clients exprimés en plusieurs langues. La compétence de détection de langue détermine la langue utilisée. Le résultat est ensuite transmis à l'extraction d'expressions clés et à la détection de sentiments, en prenant en considération la langue lors de la détection des sentiments et des expressions.

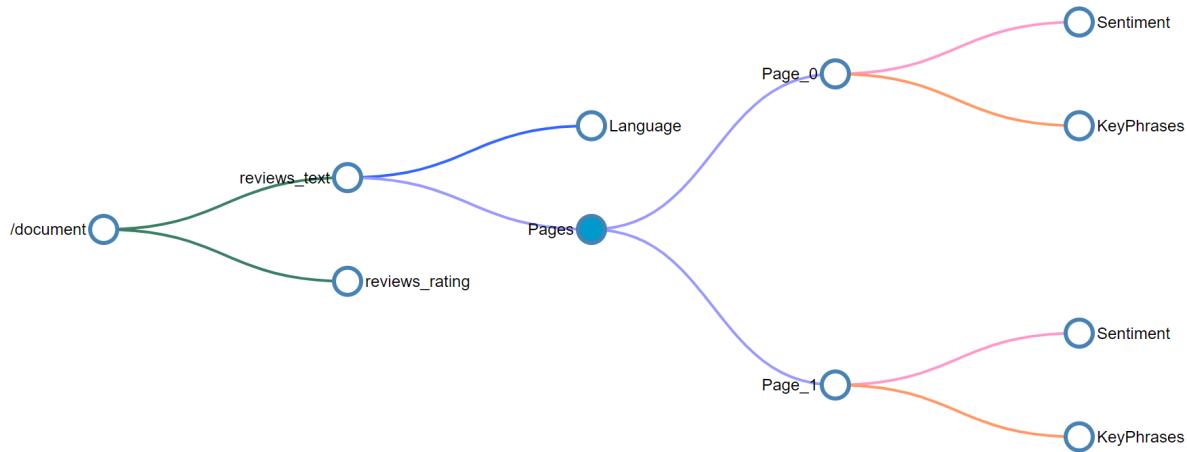
La compétence de détection de la langue est la troisième compétence (compétence n° 3) définie dans l'ensemble de compétences, mais c'est la compétence suivante à exécuter. Comme elle n'est pas bloquée dans l'attente d'entrées, elle s'exécute parallèlement à la compétence précédente. À l'instar de la compétence de division qui l'a précédée, la compétence de détection de la langue est également appelée une fois pour chaque document. L'arborescence d'enrichissements comporte désormais un nouveau nœud pour la langue.



### Compétence n° 3 : Expressions clés

Avec le contexte `/document/reviews_text/pages/*`, la compétence des expressions clés est appelée une fois pour chacun des éléments dans la collection `pages`. La sortie de la compétence est un nœud placé sous l'élément page associé.

Vous pouvez maintenant examiner le reste des compétences dans l'ensemble de compétences et regarder comment l'arborescence des enrichissements continue de croître à l'exécution de chaque compétence. Certaines compétences, telles que la compétence de fusion et la compétence de modélisation, créent également des nœuds, mais utilisent uniquement les données de nœuds existants et ne créent pas d'enrichissements supplémentaires.



Les couleurs des connecteurs dans l'arborescence ci-dessus indiquent que les enrichissements ont été créés par différentes compétences, c'est-à-dire que les nœuds devront être traités individuellement et qu'ils ne feront pas partie de l'objet retourné lors de la sélection du nœud parent.

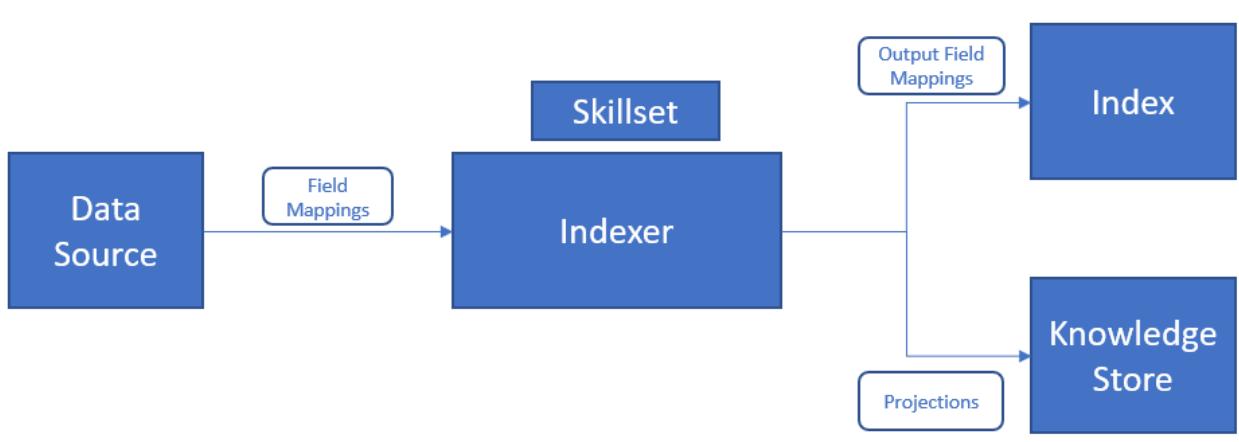
## Enregistrer les enrichissements

Dans Recherche cognitive Azure, l'indexeur enregistre la sortie qu'il crée. L'une des sorties est toujours un [index pouvant faire l'objet d'une recherche](#). La spécification d'un index est une exigence et, lorsque vous attachez un ensemble de compétences, les données ingérées par l'index incluent la substance des enrichissements. En règle générale, les sorties de compétences spécifiques, comme les expressions clés ou les scores de sentiment, sont ingérées dans l'index d'un champ créé à cet effet.

Si vous le souhaitez, l'indexeur peut également envoyer la sortie vers une [base de connaissances](#) à des fins de consommation dans d'autres outils ou processus. La base de connaissances est définie dans le cadre de l'ensemble de compétences. Sa définition détermine si vos documents enrichis sont projetés sous forme de tables ou d'objets (fichiers ou objets blob). Les projections tabulaires sont bien adaptées à l'analyse interactive dans des outils tels que Power BI, tandis que les fichiers et les objets blob sont généralement utilisés en science des données ou dans des processus similaires. Dans cette section, vous allez apprendre comment la composition des ensembles de compétences peut mettre en forme les tables ou les objets que vous souhaitez projeter.

## Projections

Pour le contenu qui cible une base de connaissances, vous devez prendre en compte la façon dont le contenu est structuré. La *projection* est le processus qui consiste à sélectionner les nœuds de l'arborescence d'enrichissements et à en créer une expression physique dans la base de connaissances. Les projections sont des formes personnalisées du document (contenu et enrichissements) qui peuvent être générées en sortie sous forme de projections de tables ou d'objets. Pour en savoir plus sur l'utilisation des projections, consultez [Utilisation de projections](#).



## SourceContext

L'élément `sourceContext` s'utilise uniquement dans les entrées de compétence et les projections. Il permet de construire des objets imbriqués à plusieurs niveaux. Vous devrez peut-être créer un objet pour le transmettre en tant qu'entrée à une compétence ou à un projet dans la base de connaissances. Étant donné que des nœuds d'enrichissement peuvent ne pas être des objets JSON valides dans l'arborescence d'enrichissement et que le référencement d'un nœud dans l'arborescence retourne uniquement cet état du nœud lors de sa création, l'utilisation d'enrichissements en guise d'entrées ou projections de compétence vous oblige à créer un objet JSON bien formé. Avec `sourceContext`, vous pouvez construire un objet hiérarchique de type anonyme, ce qui nécessiterait plusieurs compétences si vous utilisiez uniquement le contexte.

L'utilisation de `sourceContext` est illustré dans l'exemple suivant. Examinez la sortie de compétence qui a généré un enrichissement afin de déterminer s'il s'agit d'un objet JSON valide et non d'un type primitif.

## Découpage de projections

Quand vous définissez un groupe de projections de tables, un nœud de l'arborescence d'enrichissements peut être divisé pour être projeté dans plusieurs tables associées. Si vous ajoutez une table dont le chemin source est un nœud enfant d'une projection de table existante, le nœud enfant qui en résulte n'est pas un enfant de cette projection, mais il est projeté à la place dans la nouvelle table associée. Cette technique de découpage vous permet de définir un nœud unique dans une compétence de modélisation qui peut être la source de toutes vos projections de tables.

## Mise en forme de projections

Il existe deux façons de définir une projection :

- Utilisez la modélisation de texte pour créer un nœud qui est le nœud racine pour tous les enrichissements que vous projetez. Ensuite, dans vos projections, vous référez uniquement la sortie de la compétence

de modélisation.

- Utilisez la mise en forme d'une projection dans la définition de la projection.

L'approche de la modélisation est plus détaillée que la mise en forme incluse, mais elle garantit que toutes les mutations de l'arborescence d'enrichissements sont présentes dans les compétences et que la sortie est un objet réutilisable. Par contre, la mise en forme incluse vous permet de créer la forme dont vous avez besoin, mais elle constitue un objet anonyme qui est utilisable uniquement dans la projection pour laquelle elle est définie. Les approches peuvent s'utiliser ensemble ou séparément. L'ensemble de compétences créé pour vous dans le workflow du portail contient les deux. Il utilise une compétence de modélisation pour les projections de tables, mais aussi une mise en forme incluse pour projeter la table d'expressions clés.

Pour compléter l'exemple, vous pouvez choisir de supprimer la mise en forme incluse et d'utiliser une compétence de modélisation afin de créer un nœud spécifique pour les expressions clés. Pour créer une forme projetée dans trois tables (`hotelReviewsDocument`, `hotelReviewsPages` et `hotelReviewsKeyPhrases`), vous utilisez les deux options décrites dans les sections suivantes.

#### Compétence de modélisation et projection

Cette

##### NOTE

Certaines colonnes de la table de documents ont été supprimées de cet exemple par souci de concision.

```
{  
    "@odata.type": "#Microsoft.Skills.Util.ShaperSkill",  
    "name": "#5",  
    "description": null,  
    "context": "/document",  
    "inputs": [  
        {  
            "name": "reviews_text",  
            "source": "/document/reviews_text",  
            "sourceContext": null,  
            "inputs": []  
        },  
        {  
            "name": "reviews_title",  
            "source": "/document/reviews_title",  
            "sourceContext": null,  
            "inputs": []  
        },  
        {  
            "name": "AzureSearch_DocumentKey",  
            "source": "/document/AzureSearch_DocumentKey",  
            "sourceContext": null,  
            "inputs": []  
        },  
        {  
            "name": "pages",  
            "source": null,  
            "sourceContext": "/document/reviews_text/pages/*",  
            "inputs": [  
                {  
                    "name": "SentimentScore",  
                    "source": "/document/reviews_text/pages/*/Sentiment",  
                    "sourceContext": null,  
                    "inputs": []  
                },  
                {  
                    "name": "LanguageCode",  
                    "source": "/document/Language",  
                    "sourceContext": null,  
                    "inputs": []  
                }  
            ]  
        }  
    ]  
}
```

```

        "sourceContext": null,
        "inputs": []
    },
    {
        "name": "Page",
        "source": "/document/reviews_text/pages/*",
        "sourceContext": null,
        "inputs": []
    },
    {
        "name": "keyphrase",
        "sourceContext": "/document/reviews_text/pages/*/Keyphrases/*",
        "inputs": [
            {
                "source": "/document/reviews_text/pages/*/Keyphrases/*",
                "name": "Keyphrases"
            }
        ]
    }
],
"outputs": [
{
    "name": "output",
    "targetName": "tableprojection"
}
]
}

```

Avec le nœud `tableprojection` défini dans la section `outputs` ci-dessus, nous pouvons maintenant utiliser la fonctionnalité de découpage pour projeter les éléments du nœud `tableprojection` dans des tables différentes :

#### NOTE

Il s'agit uniquement d'un extrait de la projection configurée dans la base de connaissances.

```

"projections": [
{
    "tables": [
        {
            "tableName": "hotelReviewsDocument",
            "generatedKeyName": "Documentid",
            "source": "/document/tableprojection"
        },
        {
            "tableName": "hotelReviewsPages",
            "generatedKeyName": "Pagesid",
            "source": "/document/tableprojection/pages/*"
        },
        {
            "tableName": "hotelReviewsKeyPhrases",
            "generatedKeyName": "KeyPhrasesid",
            "source": "/document/tableprojection/pages/*/keyphrase/*"
        }
    ]
}
]

```

#### Mise en forme incluse pour les projections

L'approche de la mise en forme incluse ne nécessite pas de compétence de modélisation, car toutes les formes nécessaires aux projections sont créées au fur et à mesure des besoins. Pour projeter les mêmes données que l'exemple précédent, nous utilisons une option de projection incluse similaire à ceci :

```

"projections": [
    {
        "tables": [
            {
                "tableName": "hotelReviewsInlineDocument",
                "generatedKeyName": "Documentid",
                "sourceContext": "/document",
                "inputs": [
                    {
                        "name": "reviews_text",
                        "source": "/document/reviews_text"
                    },
                    {
                        "name": "reviews_title",
                        "source": "/document/reviews_title"
                    },
                    {
                        "name": "AzureSearch_DocumentKey",
                        "source": "/document/AzureSearch_DocumentKey"
                    }
                ]
            },
            {
                "tableName": "hotelReviewsInlinePages",
                "generatedKeyName": "Pagesid",
                "sourceContext": "/document/reviews_text/pages/*",
                "inputs": [
                    {
                        "name": "SentimentScore",
                        "source": "/document/reviews_text/pages/*/Sentiment"
                    },
                    {
                        "name": "LanguageCode",
                        "source": "/document/Language"
                    },
                    {
                        "name": "Page",
                        "source": "/document/reviews_text/pages/*"
                    }
                ]
            },
            {
                "tableName": "hotelReviewsInlineKeyPhrases",
                "generatedKeyName": "KeyPhraseId",
                "sourceContext": "/document/reviews_text/pages/*/Keyphrases/*",
                "inputs": [
                    {
                        "name": "Keyphrases",
                        "source": "/document/reviews_text/pages/*/Keyphrases/*"
                    }
                ]
            }
        ]
    }
]

```

Dans les deux approches, nous pouvons observer comment les valeurs de `"Keyphrases"` sont projetées à l'aide de `"sourceContext"`. Le nœud `"Keyphrases"`, qui contient une collection de chaînes, est lui-même un enfant du texte de la page. Toutefois, étant donné que les projections requièrent un objet JSON et que la page est de type primitif (chaîne), `"sourceContext"` est utilisé pour inclure l'expression clé dans un objet avec une propriété nommée. Cette technique permet aussi de projeter des primitives de manière indépendante.

## Étapes suivantes

Vous pouvez maintenant créer votre premier ensemble de compétences avec des compétences cognitives.

[Créez votre premier ensemble de compétences.](#)

# Sessions de débogage dans Recherche cognitive Azure

04/10/2020 • 11 minutes to read • [Edit Online](#)

Les sessions de débogage consistent en un éditeur visuel qui fonctionne avec un ensemble de compétences existant dans le portail Azure. Au sein d'une session de débogage, vous pouvez identifier et résoudre les erreurs, valider les modifications et les transférer vers un ensemble de compétences de production dans le pipeline d'enrichissement par IA.

## IMPORTANT

Les sessions de débogage constituent une fonctionnalité d'évaluation qui est fournie sans contrat de niveau de service et qui n'est pas recommandée pour les charges de travail de production. Pour plus d'informations, consultez [Conditions d'Utilisation Supplémentaires relatives aux Évaluations Microsoft Azure](#).

## Utilisation des sessions de débogage

Lorsque vous démarrez une session, le service crée une copie de l'ensemble de compétences et de l'indexeur, puis il indexe l'endroit où un document seul est utilisé pour tester l'ensemble de compétences. Les modifications apportées au sein de la session y sont enregistrées. Les modifications apportées dans la session de débogage n'affectent pas l'ensemble de compétences de production, sauf si elles sont validées. La validation des modifications remplacera l'ensemble de compétences de production.

Pendant une session de débogage, vous pouvez modifier un ensemble de compétences, inspecter, et valider chaque noeud de l'arborescence d'enrichissement dans le contexte d'un document spécifique. Une fois les problèmes de pipeline d'enrichissement résolus, les modifications peuvent être enregistrées dans la session et validées dans l'ensemble de compétences en production.

Si le pipeline d'enrichissement ne contient pas d'erreur, une session de débogage peut être utilisée pour enrichir un document de façon incrémentielle, tester et valider chaque modification avant la validation finale.

## Création d'une session de débogage

Pour démarrer une session de débogage, vous devez disposer d'un pipeline d'enrichissement par IA existant, incluant : une source de données, un ensemble de compétences, un indexeur et un index. Pour configurer une session de débogage, vous devez nommer la session et fournir un compte de stockage à usage général qui sera utilisé pour mettre en cache les exécutions de compétences lors de l'exécution de l'indexeur. Vous devrez également sélectionner l'indexeur qui sera exécuté. L'indexeur contient des références stockées dans la source de données, dans l'ensemble de compétences et dans l'index. La session de débogage utilise par défaut le premier document de la source de données. Sinon, vous pouvez spécifier le document à parcourir dans la source de données.

**new-debug-session**  
Debug session

Save Session Refresh Delete Run Commit changes... download

**Definition** AI Enrichments Errors/Warnings

Debug session name \*  ✓

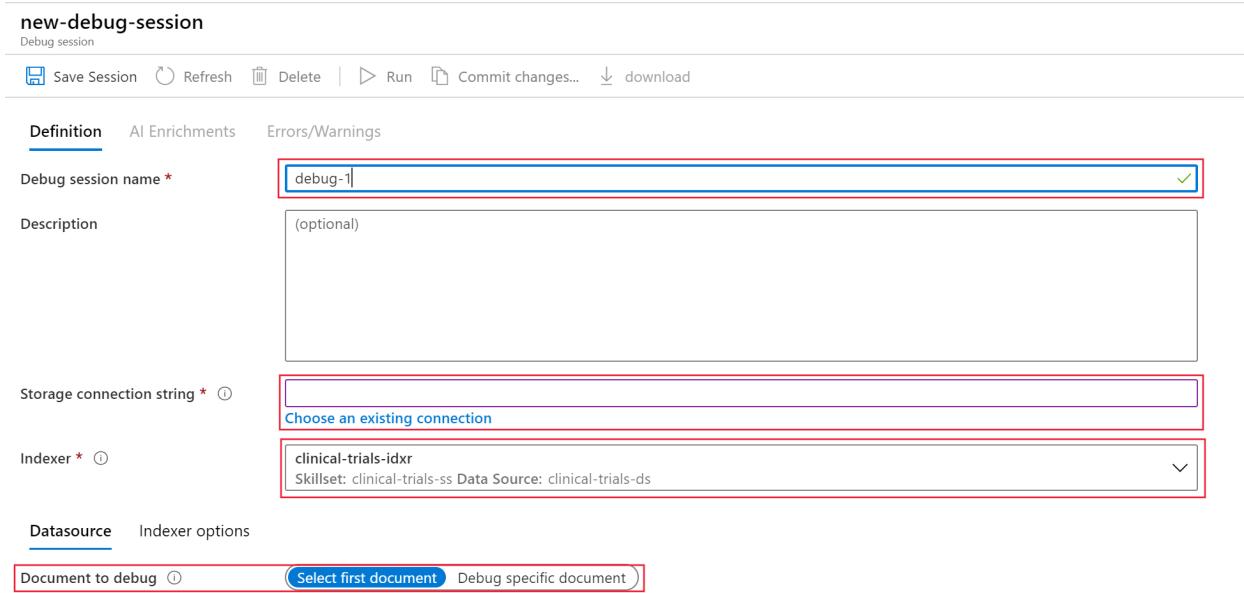
Description

Storage connection string \*

Indexer \*   
Skillset: clinical-trials-ss Data Source: clinical-trials-ds

Datasource Indexer options

Document to debug  Debug specific document



## Fonctionnalités de la session de débogage

La session de débogage commence par exécuter l'ensemble de compétences sur le document sélectionné. La session de débogage enregistre les métadonnées supplémentaires associées à chaque opération au sein de l'ensemble de compétences. Les métadonnées créées par les exécutions de compétences du pipeline indiquent l'ensemble de fonctionnalités qui suit. Elles sont ensuite utilisées pour aider à identifier et résoudre les problèmes liés à l'ensemble de compétence.

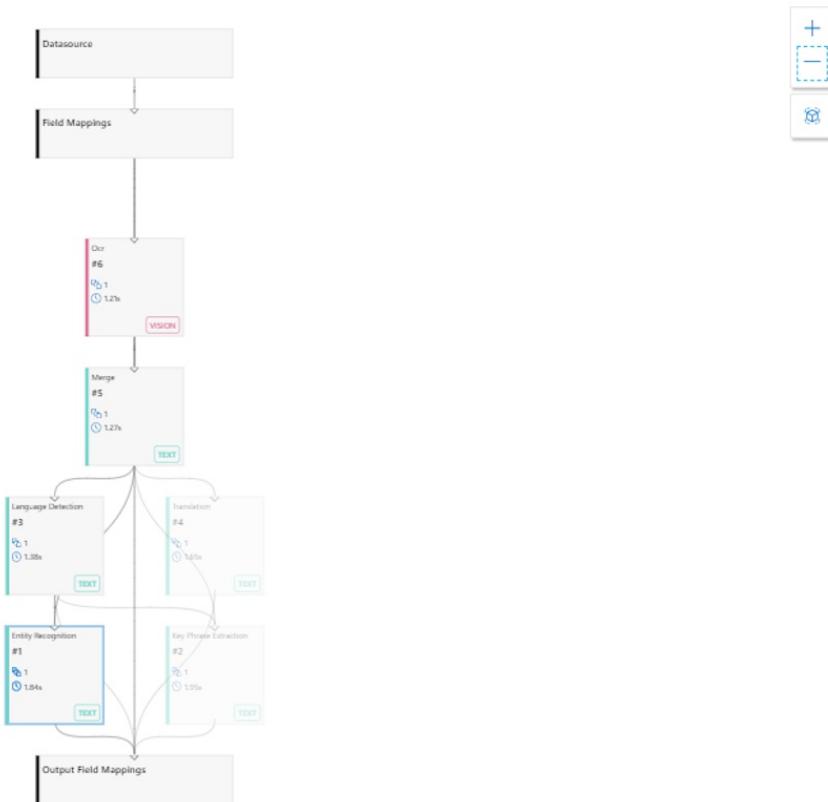
## Enrichissements par IA

À mesure que les compétences s'exécutent, un arbre d'enrichissement, représentant le document, se développe. L'utilisation d'une arborescence pour visualiser les sorties de compétences ou d'enrichissements offre une vue d'ensemble de tous les enrichissements effectués. Vous pouvez consulter l'ensemble du document et inspecter chaque nœud de l'arborescence d'enrichissement. Cette perspective facilite la modélisation d'objets. Ce format fournit également des signaux visuels pour le type, le chemin d'accès et le contenu de chaque nœud dans l'arborescence.

## Graphique des compétences

L'affichage du **graphe des compétences** fournit une représentation visuelle hiérarchique de l'ensemble de compétences. Le graphique représente l'ordre dans lequel les compétences sont exécutées, de haut en bas. Les compétences qui dépendent de la sortie d'autres compétences sont indiquées plus bas dans le graphique. Les compétences sur le même niveau hiérarchique peuvent être exécutées en parallèle.

Sélectionner une compétence dans le graphique mettra en surbrillance les compétences qui y sont connectées, les nœuds qui créent ses entrées et ceux qui acceptent ses sorties. Le type, les erreurs ou avertissements ainsi que le nombre d'exécutions de chaque nœud de compétence apparaissent. Le **graphique des compétence** vous permet de sélectionner les compétences à déboguer ou améliorer. Lorsque vous sélectionnez une compétence, les informations la concernant sont affichées dans le volet d'information de la compétence à droite du graphique.



## Détails de la compétence

Le volet d'information sur la compétence affiche un ensemble de zones pour l'utilisation d'une compétence spécifique, lorsque celle-ci est en surbrillance dans le **graphique des compétences**. Vous pouvez examiner et modifier les informations des paramètres de la compétence. La définition JSON de la compétence est fournie. Les informations de l'exécution de la compétence, ainsi que les erreurs et avertissements, sont également affichés. L'onglet **Paramètres des compétences** et l'**éditeur de compétences JSON** permettent de modifier directement la compétence. [`</>`](#) ouvre une fenêtre permettant d'afficher et de modifier les expressions des entrées et des sorties de compétences.

Les contrôles d'entrée imbriqués dans la fenêtre « paramètres des compétences » peuvent être utilisés pour créer des formes complexes pour les projections, des mappages de champs de sortie pour un champ complexe ou une entrée de compétence. Lorsqu'elles sont utilisées avec l'**évaluateur d'expression**, les entrées imbriquées constituent un test facile pour valider le générateur d'expressions.

## Historique d'exécution des compétences

Une compétence peut s'exécuter plusieurs fois dans un ensemble de compétences sur un document seul. Par exemple, la compétence de reconnaissance optique de caractères (OCR) s'exécutera une fois pour chaque image extraite de chacun des documents. Dans le volet d'information sur la compétence, l'onglet **Exécutions** affiche l'historique d'exécution de la compétence, ce qui permet un examen plus approfondi de chaque appel de la compétence.

L'historique d'exécution permet de remonter jusqu'à la compétence qui a généré un enrichissement spécifique. Cliquer sur une entrée de compétence pour accéder à la compétence qui a généré cette entrée. Cela permet d'identifier la cause racine d'un problème qui peut se manifester dans une compétence en aval.

Lorsqu'un problème potentiel est identifié, l'historique d'exécution affiche des liens vers les compétences qui ont généré les entrées spécifiques, et offre une fonctionnalité de traçage de piles similaires. Cliquer sur la compétence associée à l'entrée, accéder à la compétence pour corriger les bogues ou continuer à remonter vers le problème

spécifique.

Lors de la création d'une compétence personnalisée ou du débogage d'une erreur avec une compétence personnalisée, il est possible de générer une requête d'appel de compétence dans l'historique d'exécution.

## Structure de données enrichie

Le volet **Structure de données enrichie** montre les enrichissements du document via l'ensemble de compétences, en détaillant le contexte et la compétence d'origine pour chaque enrichissement. L'**évaluateur d'expression** peut également servir à afficher le contenu de chaque enrichissement.

The screenshot shows the 'Enriched Data Structure' view in the AI Enrichments tab. The main area displays a hierarchical tree of paths and their corresponding outputs and originating sources. A modal window titled 'Expression evaluator' is open, showing a context path of '/document' and an expression path of '/document/merged\_content/keyphrases'. The value pane displays a list of keyphrases.

Path	Output	Originating Source
document	{"normalized_images": [{"\$type": "file", "url": "{\$@trim..."}]	</>
merged_content	"\n\n[ Study of BMN 110 in ...	#5
keyphrases	["Study of BMN", "Syndrome", "Pediatric Patients", "Y...	#2
locations	["IVA"]	#1
translated_text	"\n[ Study of BMN 110 in Pedia...	#4
entities	[{"name": "BMN", "wikipediald": null, "wikipediaLang..."}]	#1
organizations	["BMN"]	#1
language	"en"	#3
normalized_images		
#		
layoutText	[{"language": "en", "text": {"@trimmed": true, "display": "..."}]	#6
text	[{"@trimmed": true, "display": "Study of BMN 110 in ..."}]	#6

Expression evaluator  
Context  
/document  
Expression  
/document/merged\_content/keyphrases  
Evaluate

Value

1	"Study of BMN",
2	"Syndrome",
3	"Pediatric Patients",
4	"Years of Age",
5	"Morgulio",
6	"dose of study drug",
7	"MPS IVA",
8	"Mucopolysaccharides IVA",
9	"wk infusions of BMN",
10	"open-label Phase",
11	"safety",
12	"time of administration",
13	"efficacy of weekly",
14	

Close

## Évaluateur d'expression

L'**évaluateur d'expression** offre un aperçu de la valeur de tout chemin d'accès. Il permet de modifier le chemin d'accès et de tester les résultats avant de mettre à jour les entrées ou le contexte d'une compétence ou d'une projection.

## Erreurs/avertissemens

Cette fenêtre affiche toutes les erreurs et les avertissements produits par l'ensemble de compétences lorsqu'il est exécuté sur le document dans la session de débogage.

## Limites

Les sessions de débogage fonctionnent avec toutes les sources de données généralement disponibles et la plupart des sources de données en préversion. Les API MongoDB et Cassandra de Cosmos DB (toutes deux en préversion) ne sont actuellement pas prises en charge.

## Étapes suivantes

Maintenant que vous avez compris les composantes des sessions de débogage, essayez le tutoriel pour mettre la théorie en pratique.

## Tutoriel : explorer les fonctionnalités des sessions de débogage

# Base de connaissances dans Recherche cognitive Azure

04/10/2020 • 16 minutes to read • [Edit Online](#)

La base de connaissances est une fonctionnalité de la Recherche cognitive Azure. Elle permet de conserver la sortie d'un [pipeline d'enrichissement par l'IA](#) en vue d'une analyse indépendante ou d'un traitement en aval. Un *document enrichi* est la sortie d'un pipeline, créée à partir d'un contenu qui a été extrait, structuré et analysé à l'aide de processus IA. Dans un pipeline IA standard, les documents enrichis sont temporaires, utilisés uniquement pendant l'indexation, puis ignorés. Le choix de créer une base de connaissances vous permettra de conserver les documents enrichis.

Si vous avez déjà utilisé des compétences cognitives par le passé, vous savez que des *ensembles de compétences* déplacent un document dans une séquence d'enrichissements. Le résultat peut être un index de recherche ou des projections dans une base de connaissances. Les deux sorties, l'index de recherche et la base de connaissances sont des produits du même pipeline, dérivés des mêmes entrées, mais qui produisent une sortie structurée, stockée et utilisée de manières très différentes.

Physiquement, une base de connaissances représente un [stockage Azure](#), soit le stockage Table Azure, soit le stockage Blob Azure, ou les deux. Tout outil ou processus pouvant se connecter au Stockage Azure peut utiliser le contenu d'un magasin de connaissances.

## Avantages de la base de connaissances

Une base de connaissances vous fournit une structure, un contexte et un contenu réel, extraits de fichiers de données non structurés et semi-structurés, tels que des objets blob, des fichiers image ayant fait l'objet d'une analyse, ou même des données structurées remodelées dans nouveaux formulaires. Une [procédure pas à pas](#) montre comment un document JSON dense de source sure est partitionné en sous-structures, reconstitué en nouvelles structures et mis à disposition pour des processus en aval, comme des charges de travail liées au Machine Learning et à la science des données.

Bien qu'il soit utile de voir ce qu'un pipeline d'enrichissement par l'IA peut produire, le véritable potentiel d'une base de connaissances réside dans sa capacité à remodeler les données. Vous pouvez commencer avec un ensemble de compétences de base, puis procéder à une itération sur celui-ci afin d'ajouter des niveaux de structure croissants, que vous pouvez ensuite combiner en nouvelles structures, utilisables dans des applications autres que la Recherche cognitive Azure.

La base de connaissances vous offre notamment les avantages suivants :

- Utilisez des documents enrichis dans des [outils d'analyse et de génération d'états](#) autres que des outils de recherche. Power BI avec Power Query est un excellent choix, mais tout outil ou application capable de se connecter à Stockage Azure peut puiser dans une base de connaissances que vous avez créée.
- Affinez un pipeline d'indexation IA lors des étapes de débogage et de la définition des ensembles de compétences. Une base de connaissances vous présente le produit de la définition d'un ensemble de compétences dans un pipeline d'indexation IA. Vous pouvez utiliser ces résultats afin de concevoir un ensemble de compétences plus performant car vous voyez exactement à quoi ressemblent les enrichissements. Vous pouvez utiliser l'[Explorateur Stockage](#) de Stockage Azure pour afficher le contenu d'une base de connaissances.

- Modelez les données pour les transformer en nouveaux formulaires. Le remodelage est codifié en ensembles de compétences, mais le fait est qu'un ensemble de compétences peut désormais fournir cette capacité. La [compétence Modélisateur](#) de la Recherche cognitive Azure a été étendue pour s'adapter à cette tâche. Le remodelage vous permet de définir une projection alignée sur l'utilisation que vous prévoyez de faire des données tout en préservant les relations.

#### NOTE

Vous ne connaissez pas l'enrichissement et les compétences cognitives de l'intelligence artificielle ? La Recherche cognitive Azure s'intègre aux fonctionnalités de vision et de langage de Cognitive Services pour extraire et enrichir les données sources à l'aide de la reconnaissance optique de caractères sur des fichiers image, de la reconnaissance d'entités et de l'extraction d'expressions clés à partir de fichiers texte et autres. Pour plus d'informations, consultez [Enrichissement de l'IA dans la Recherche cognitive Azure](#).

## Stockage physique

L'expression physique d'une base de connaissances est articulée au travers de l'élément `projections` d'une définition `knowledgeStore` dans un ensemble de compétences. La projection définit une structure de la sortie afin qu'elle corresponde à votre utilisation prévue.

Des projections peuvent être articulées en tant que tables, objets ou fichiers.

```
"knowledgeStore": {
  "storageConnectionString": "<YOUR-AZURE-STORAGE-ACCOUNT-CONNECTION-STRING>",
  "projections": [
    {
      "tables": [ ],
      "objects": [ ],
      "files": [ ]
    },
    {
      "tables": [ ],
      "objects": [ ],
      "files": [ ]
    }
  ]
}
```

Le type de projection que vous spécifiez dans cette structure détermine le type de stockage utilisé par la base de connaissances.

- Un stockage Table est utilisé lorsque vous définissez `tables`. Définissez une projection de table lorsque vous avez besoin de structures de rapports tabulaires pour des entrées dans des outils analytiques, ou les exporter en tant que trames de données dans d'autres magasins de données. Vous pouvez spécifier plusieurs `tables` pour obtenir un sous-ensemble ou une section transversale de documents enrichis. Dans le même groupe de projection, les relations entre tables sont conservées afin que vous puissiez les utiliser toutes.
- Un stockage Blob est utilisé lorsque vous définissez `objects` ou `files`. La représentation physique d'un `object` est une structure JSON hiérarchique représentant un document enrichi. Un `file` est une image extraite d'un document, transférée intacte vers un stockage Blob.

Un objet projection unique contient un ensemble de `tables`, `objects`, `files`, et, pour de nombreux scénarios, la création d'une seule projection peut suffire.

Toutefois, il est possible de créer plusieurs ensembles de projections `table` - `object` - `file`, et vous pouvez le faire si vous souhaitez des relations entre données différentes. Dans un ensemble, les données sont liées, en

supposant que ces relations existent et peuvent être détectées. Si vous créez des ensembles supplémentaires, les documents de chaque groupe ne sont jamais liés. Voici un exemple de ce que pourrait donner une utilisation de plusieurs groupes de projection si vous souhaitez que les mêmes données soient projetées pour une utilisation avec votre système en ligne, qu'elles soient représentées de manière spécifique, et que les mêmes données soient projetées pour une utilisation dans un pipeline de science des données représenté différemment.

## Spécifications

Un [Stockage Azure](#) est requis. Il fournit le stockage physique. Vous pouvez utiliser un stockage Blob, un stockage Table ou les deux. Un stockage Blob est utilisé pour des documents enrichis intacts, généralement lorsque la sortie est acheminée vers des processus en aval. Un stockage Table est destiné à des tranches de documents enrichis, couramment utilisées à des fins d'analyse et de rapport.

Un [ensemble de compétences](#) est requis. Il contient la définition de `knowledgeStore` et détermine la structure et la composition d'un document enrichi. Vous ne pouvez pas créer une base de connaissances à l'aide d'un ensemble de compétences vide. Vous devez avoir au moins une compétence dans un ensemble compétences.

Un [indexeur](#) est requis. Un ensemble de compétences est appelé par un indexeur qui pilote l'exécution. Les indexeurs sont fournis avec leur propre ensemble d'exigences et d'attributs. Plusieurs de ces attributs ont une incidence directe sur une base de connaissances :

- Les indexeurs requièrent une [source de données Azure pris en charge](#) (le pipeline qui crée finalement la base de connaissances commence par extraire des données d'une source prise en charge sur Azure).
- Les indexeurs requièrent un index de recherche. Un indexeur requiert que vous fournissiez un schéma d'index, même si vous n'envisagez pas de l'utiliser. Un index minimal comprend un champ de chaîne désigné comme clé.
- Les indexeurs fournissent des mappages de champs facultatifs, utilisés pour attribuer en tant qu'alias un champ source à un champ de destination. Si un mappage de champs par défaut nécessite une modification (pour utiliser un autre nom ou type), vous pouvez créer un [mappage de champs](#) dans un indexeur. Pour la sortie de la base de connaissances, la destination peut être un champ dans un objet blob ou une table.
- Les indexeurs ont des planifications, et d'autres propriétés telles que des mécanismes de détection des modifications fournis par diverses sources de données peuvent également être appliquées à une base de connaissances. Par exemple, vous pouvez [planifier](#) un enrichissement à intervalles réguliers pour actualiser le contenu.

## Comment créer une base de connaissances

Pour créer une base de connaissances, utilisez le portail ou l'API REST (`api-version=2020-06-30`).

### Utilisation du portail Azure

L'Assistant [Importation de données](#) contient des options pour la création d'une base de connaissances. Pour l'exploration initiale, [créez votre première base de connaissances en quatre étapes](#).

1. Sélectionnez une source de données prise en charge.
2. Spécifiez un enrichissement : joignez une ressource, sélectionnez des compétences et spécifiez une base de connaissances.
3. Créez un schéma d'index. L'Assistant l'exige et peut en inférer un pour vous.
4. Exécutez l'Assistant. L'extraction, l'enrichissement et le stockage se produisent dans cette dernière étape.

## Utiliser Créer un ensemble de compétences (API REST)]

Une `knowledgeStore` est définie au sein d'un [ensemble de compétences](#) qui est appelé à son tour par un [indexeur](#). Pendant l'enrichissement, la Recherche cognitive Azure crée un espace dans votre compte de stockage Azure et projette les documents enrichis en tant qu'objets blob ou dans des tables, en fonction de votre configuration.

L'API REST est un mécanisme qui vous permet de créer une base de connaissances par programme. Une manière facile d'explorer consiste à [créer votre première base de connaissances à l'aide de Postman et de l'API REST](#).

## Comment se connecter avec des outils et applications

Une fois les enrichissements disponibles dans le stockage, n'importe quel outil ou technologie capable de se connecter à Stockage Blob ou Table Azure peut être utilisé pour explorer, analyser ou utiliser le contenu. La liste suivante est un début :

- L'[Explorateur Stockage](#) permet d'afficher la structure et le contenu des documents enrichis. Considérez-le comme votre outil de référence pour afficher le contenu de la base de connaissances.
- Power BI pour la création de rapports et l'analyse.
- [Azure Data Factory](#) permet d'effectuer d'autres manipulations.

## Informations de référence sur l'API

La version `2020-06-30` de l'API REST fournit une base de connaissances via des définitions supplémentaires sur des ensembles de compétences. En plus de la référence, consultez [créer une base de connaissances à l'aide de Postman](#) pour plus d'informations sur la façon d'appeler les API.

- [Créer un ensemble de compétences \(api-version=2020-06-30\)](#)
- [Mettre à jour un ensemble de compétences \(api-version=2020-06-30\)](#)

## Étapes suivantes

La base de connaissances offre la persistance de documents enrichis, utile lors de la conception d'un ensemble de compétences, ou la création de structures et de contenu pour une consommation par les applications clientes qui peuvent accéder à un compte de stockage Azure.

L'approche la plus simple pour créer des documents enrichis consiste à [utiliser le portail](#), mais vous pouvez également utiliser Postman et l'API REST, ce qui est plus utile si vous souhaitez obtenir des insights sur la façon dont les objets sont créés et référencés.

### [Créer une base de connaissances à l'aide de Postman et de REST](#)

Découvrez les projections, les fonctionnalités et la façon dont vous [les définissez dans un ensemble de compétences](#).

### [Projections dans une base de connaissances](#)

Pour un tutoriel sur les concepts de projections avancées, tels que le découpage, la mise en forme inline et les relations, commencez par [définir des projections dans une base de connaissances](#).

### [Définir des projections dans une base de connaissances](#)

# « Projections » de base de connaissances dans Recherche cognitive Azure

04/10/2020 • 15 minutes to read • [Edit Online](#)

Recherche cognitive Azure permet l'enrichissement de contenu via des compétences cognitives intégrées et personnalisées dans le cadre de l'indexation. Les enrichissements créent de nouvelles informations là où aucune n'existeait précédemment : extraction d'informations à partir d'images, détection de sentiments, d'expressions clés et d'entités à partir de texte, pour n'en nommer que quelques-uns. Les enrichissements ajoutent également une structure à du texte non différencié. Tous ces processus produisent des documents qui rendent la recherche en texte intégral plus efficace. Dans de nombreux cas, les documents enrichis sont utiles pour des scénarios autres que la recherche, pour l'exploration de connaissances par exemple.

Les projections, un composant de la [base de connaissances](#), sont des vues de documents enrichis qui peuvent être enregistrés dans un stockage physique à des fins d'exploration de connaissances. Une projection vous permet de « projeter » vos données dans une forme qui répond à vos besoins, en conservant les relations afin que les outils tels que Power BI puissent lire les données sans effort supplémentaire.

Les projections peuvent être tabulaires, avec des données stockées dans des lignes et des colonnes dans le stockage Table Azure, ou des objets JSON stockés dans le stockage Blob Azure. Vous pouvez définir plusieurs projections de vos données pendant leur enrichissement. Des projections multiples sont utile lorsque vous souhaitez que les mêmes données soient mises en forme différemment pour des cas d'utilisation individuels.

La base de connaissances prend en charge trois types de projections :

- **Tables** : Pour les données qui sont mieux représentées sous forme de lignes et de colonnes, les projections de table vous permettent de définir une forme schématisée ou une projection dans le stockage Table. Seuls les objets JSON valides peuvent être projetés sous forme de tables, le document enrichi peut contenir des nœuds qui ne sont pas des objets JSON nommés. Par ailleurs, lors de la projection de ces objets, il peut créer un objet JSON valide avec une compétence de modélisateur ou une mise en forme incluse.
- **Objets** : Lorsque vous avez besoin d'une représentation JSON de vos données et enrichissements, les projections d'objet sont enregistrées comme des objets Blob. Seuls les objets JSON valides peuvent être projetés sous forme d'objets, le document enrichi peut contenir des nœuds qui ne sont pas des objets JSON nommés. Par ailleurs, lors de la projection de ces objets, il peut créer un objet JSON valide avec une compétence de modélisateur ou une mise en forme incluse.
- **Fichiers** : Quand vous devez enregistrer les images extraites des documents, les projections de fichiers vous permettent d'enregistrer les images normalisées dans un stockage d'objets blob.

Pour voir des projections définies en contexte, consultez [Créer une base de connaissances avec REST](#).

## Groupes de projections

Dans certains cas, vous devrez projeter vos données enrichies dans différentes formes pour répondre à différents objectifs. La base de connaissances vous permet de définir plusieurs groupes de projections. Les groupes de projection disposent des principales caractéristiques d'exclusivité mutuelle et de parenté suivantes.

### Exclusivité mutuelle

Tout le contenu projeté dans un même groupe est indépendant des données projetées dans d'autres groupes de projections. Cette indépendance implique que les mêmes données peuvent être mises en forme différemment,

mais répétées dans chaque groupe de projections.

### Parenté

Les groupes de projection vous permettent désormais de projeter vos documents dans différents types de projections tout en préservant les relations entre ceux-ci. Tout le contenu projeté dans un même groupe de projections conserve les relations entre les données dans les types de projections. À l'intérieur des tableaux, les relations sont basées sur une clé générée, et chaque nœud enfant conserve une référence au nœud parent. Parmi les types (tables, objets et fichiers), les relations sont préservées quand un seul nœud est projeté vers différents types. Par exemple, imaginez un scénario dans lequel vous avez un document contenant des images et du texte. Vous pouvez projeter le texte vers des tables ou des objets, et les images vers des fichiers dont l'URL figure dans une colonne/propriété dans ces tables ou objets.

## Mise en forme d'entrée

Disposer de vos données dans la bonne forme ou structure est essentiel pour les utiliser efficacement, qu'il s'agisse de tables ou d'objets. La possibilité de mettre en forme ou de structurer vos données selon la méthode d'accès et d'utilisation souhaitées est une fonctionnalité clé présentée comme la compétence de **modélisateur** de l'ensemble de compétences.

Les projections sont plus simples à définir lorsque vous avez un objet dans l'arborescence d'enrichissement qui correspond au schéma de la projection. La [compétence de modélisateur](#) mise à jour vous permet de composer un objet à partir de différents nœuds de l'arborescence d'enrichissement et de les appartenir sous un nouveau nœud. La compétence de **modélisateur** vous permet de définir des types complexes avec des objets imbriqués.

Lorsqu'une nouvelle forme définie contient tous les éléments que vous devez projeter, vous pouvez maintenant utiliser cette forme comme source pour vos projections ou comme entrée d'une autre compétence.

## Découpage de la projection

Quand vous définissez un groupe de projections, un nœud de l'arborescence d'enrichissements peut être divisé pour être projeté dans plusieurs tables ou objets associés. Si vous ajoutez une projection dont le chemin source est un nœud enfant d'une projection existante, le nœud enfant est divisé à partir du nœud parent et projeté dans la nouvelle table ou le nouvel objet associés. Cela vous permet de définir un nœud unique dans une compétence de modélisation qui peut être la source de toutes vos projections.

## Projections de table

Leur importation étant plus simple, nous vous recommandons les projections de table pour l'exploration des données avec Power BI. Les projections de table permettent également de modifier la cardinalité entre les relations de table.

Vous pouvez projeter un document de votre index dans plusieurs tables, en conservant les relations. Lors de la projection dans plusieurs tables, la forme complète est projetée dans chaque table, sauf si un nœud enfant est la source d'une autre table du même groupe.

### Définition d'une projection de table

Lorsque vous définissez une projection de la table dans l'élément `knowledgeStore` de votre ensemble de compétences, commencez par mapper un nœud dans l'arborescence d'enrichissement avec la source de table. Ce nœud est généralement la sortie d'une compétence de **modélisateur** que vous avez ajoutée à la liste des compétences pour produire une forme spécifique dont vous avez besoin pour projeter dans des tables. Le nœud que vous choisissez de projeter peut être divisé pour être projeté dans plusieurs tables. La définition de tables est une liste de tables que vous souhaitez projeter.

Chaque table requiert trois propriétés :

- tableName : Nom de la table dans le Stockage Azure.
- generatedKeyName : Nom de colonne de la clé qui identifie de façon unique cette ligne.
- source : Nœud de l'arborescence d'enrichissement, source de vos enrichissements. Ce nœud est généralement la sortie d'un modélisateur, mais peut également être la sortie d'une des compétences.

Voici un exemple des projections de table.

```
{
  "name": "your-skillset",
  "skills": [
    ...your skills
  ],
  "cognitiveServices": {
    ... your cognitive services key info
  },
  "knowledgeStore": {
    "storageConnectionString": "an Azure storage connection string",
    "projections" : [
      {
        "tables": [
          { "tableName": "MainTable", "generatedKeyName": "SomeId", "source": "/document/EnrichedShape" },
          { "tableName": "KeyPhrases", "generatedKeyName": "KeyPhraseId", "source": "/document/EnrichedShape/*/KeyPhrases/*" },
          { "tableName": "Entities", "generatedKeyName": "EntityId", "source": "/document/EnrichedShape/*/Entities/*" }
        ]
      },
      {
        "objects": [ ]
      },
      {
        "files": [ ]
      }
    ]
  }
}
```

Comme illustré dans cet exemple, les expressions et entités clés sont modélisées dans différentes tables et contiennent une référence au parent (MainTable) pour chaque ligne.

## Projections d'objet

Les projections d'objet sont des représentations JSON de l'arborescence d'enrichissement pouvant provenir de n'importe quel nœud. Dans de nombreux cas, la même compétence de **modélisateur** que celle qui crée une projection de table peut être utilisée pour générer une projection d'objet.

```
{
  "name": "your-skillset",
  "skills": [
    ...your skills
  ],
  "cognitiveServices": {
    ... your cognitive services key info
  },

  "knowledgeStore": {
    "storageConnectionString": "an Azure storage connection string",
    "projections" : [
      {
        "tables": [ ]
      },
      {
        "objects": [
          {
            "storageContainer": "hotelreviews",
            "source": "/document/hotel"
          }
        ]
      },
      {
        "files": [ ]
      }
    ]
  }
}
```

La génération d'une projection d'objet nécessite quelques attributs spécifiques à un objet :

- `storageContainer` : Conteneur de blobs dans lequel les objets seront enregistrés
- `source` : Chemin d'accès au noeud de l'arborescence d'enrichissement qui est la racine de la projection

## Projection de fichier

Les projections de fichiers sont similaires aux projections d'objets et agissent uniquement sur la collection `normalized_images`. À l'instar des projections d'objets, les projections de fichiers sont enregistrées dans le conteneur d'objets blob dont le préfixe de dossier est la valeur encodée en base64 de l'ID de document. Les projections de fichiers ne peuvent pas partager le même conteneur que les projections d'objets. Elles doivent être projetées vers un conteneur distinct.

```
{
  "name": "your-skillset",
  "skills": [
    ...your skills
  ],
  "cognitiveServices": {
    ... your cognitive services key info
  },

  "knowledgeStore": {
    "storageConnectionString": "an Azure storage connection string",
    "projections" : [
      {
        "tables": [ ]
      },
      {
        "objects": [ ]
      },
      {
        "files": [
          {
            "storageContainer": "ReviewImages",
            "source": "/document/normalized_images/*"
          }
        ]
      }
    ]
  }
}
```

## Cycle de vie de projection

Vos projections ont un cycle de vie qui est lié à la source de données dans votre source de données. Lorsque vos données sont mises à jour et réindexées, vos projections sont mises à jour avec les résultats des enrichissements en s'assurant que vos projections sont cohérentes avec les données dans votre source de données. Les projections héritent de la stratégie de suppression que vous avez configurée pour votre index. Les projections ne sont pas supprimées lors de la suppression de l'indexeur ou du service de recherche.

## Utilisation de projections

Après l'exécution de l'indexeur, vous pouvez lire les données projetées dans les conteneurs ou les tables que vous avez spécifiés par le biais de projections.

Pour l'analyse, l'exploration dans Power BI est aussi simple que de définir le stockage Table Azure comme la source de données. Vous pouvez facilement créer un ensemble de visualisations sur vos données en utilisant les relations incluses.

Si vous devez utiliser les données enrichies dans un pipeline de science des données, vous pouvez également [charger les données d'objets Blob dans un dataframe Pandas](#).

Enfin, si vous devez exporter vos données de la base de connaissances, Azure Data Factory comprend des connecteurs pour exporter les données et les placer dans la base de données de votre choix.

## Étapes suivantes

À l'étape suivante, créez votre première base de connaissances à l'aide d'exemples de données et d'instructions.

[Créer une base de connaissances avec REST](#).

Pour un tutoriel sur les concepts de projections avancées, tels que le découpage, la mise en forme inline et les

relations, commencez par définir des projections dans une base de connaissances.

Définir des projections dans une base de connaissances

# Enrichissement incrémentiel et mise en cache dans Recherche cognitive Azure

04/10/2020 • 16 minutes to read • [Edit Online](#)

## IMPORTANT

L'enrichissement incrémentiel est actuellement en version préliminaire publique. Cette préversion est fournie sans contrat de niveau de service et n'est pas recommandée pour les charges de travail de production. Pour plus d'informations, consultez [Conditions d'Utilisation Supplémentaires relatives aux Évaluations Microsoft Azure](#). Les préversions de l'API REST offrent cette fonctionnalité. Il n'y a pas de prise en charge de portail ou de SDK .NET pour l'instant.

*L'enrichissement incrémentiel* est une fonctionnalité qui cible des [ensembles de compétences](#). Elle tire parti du Stockage Azure pour enregistrer la sortie de traitement émise par un pipeline d'enrichissement en vue d'une réutilisation dans de futures exécutions de l'indexeur. Dans la mesure du possible, l'indexeur réutilise toute sortie mise en cache qui est toujours valide.

L'enrichissement incrémentiel permet non seulement de préserver votre investissement financier dans le traitement (en particulier la reconnaissance optique des caractères et le traitement des images), mais aussi d'améliorer l'efficacité d'un système. Quand des structures et du content sont mis en cache, un indexeur peut déterminer les compétences qui ont changé et exécuter uniquement celles qui ont été modifiées, ainsi que toutes les compétences dépendantes en aval.

Un flux de travail qui utilise une mise en cache incrémentielle comprend les étapes suivantes :

1. [Créer ou identifier un compte de stockage Azure](#) pour stocker le cache.
2. [Activer l'enrichissement incrémentiel](#) dans l'indexeur.
3. [Créer un indexeur](#) (et un [ensemble de compétences](#)) pour appeler le pipeline. Lors du traitement, les étapes d'enrichissement sont enregistrées pour chaque document dans le stockage d'objets blob en vue d'une utilisation ultérieure.
4. Testez votre code, puis, après avoir apporté des modifications, utilisez la commande [Mettre à jour l'ensemble de compétences](#) pour modifier une définition.
5. [Exécutez l'indexeur](#) pour appeler le pipeline, en extrayant la sortie mise en cache pour un traitement plus rapide et plus économique.

Pour plus d'informations sur les étapes et les considérations à prendre en compte lors de l'utilisation d'un indexeur existant, consultez [Configurer l'enrichissement incrémentiel](#).

## Cache d'indexeur

L'enrichissement incrémentiel ajoute un cache au pipeline d'enrichissement. Cet indexeur met en cache les résultats du craquage de document, ainsi que les résultats de chaque compétence pour chaque document. Quand un ensemble de compétences est mis à jour, seules les compétences ayant changé ou situées en aval sont réexécutées. Les résultats mis à jour sont écrits dans le cache. Le document est mis à jour dans l'index ou dans la base de connaissances.

Physiquement, le cache est stocké dans un conteneur d'objets blob de votre compte de stockage Azure. Le cache utilise également le stockage table pour un enregistrement interne des mises à jour de traitement. Tous les index d'un service de recherche peuvent partager le même compte de stockage pour le cache de l'indexeur. Chaque indexeur se voit affecté un identificateur de cache unique et non modifiable au conteneur qu'il utilise.

# Configuration du cache

Vous devez définir la propriété `cache` sur l'indexeur pour commencer à bénéficier de l'enrichissement incrémentiel. L'exemple suivant illustre un indexeur pour lequel la mise en cache est activée. Des parties spécifiques de cette configuration sont décrites dans les sections suivantes. Pour plus d'informations, consultez [Configurer l'enrichissement incrémentiel](#).

```
{  
    "name": "myIndexerName",  
    "targetIndexName": "myIndex",  
    "dataSourceName": "myDatasource",  
    "skillsetName": "mySkillset",  
    "cache" : {  
        "storageConnectionString" : "Your storage account connection string",  
        "enableReprocessing": true  
    },  
    "fieldMappings" : [],  
    "outputFieldMappings": [],  
    "parameters": []  
}
```

Si vous définissez cette propriété sur un indexeur, vous devez également le réinitialiser et le réexécuter, ce qui entraînera un nouveau traitement de tous les documents présents dans votre source de données. Cette étape est nécessaire pour éliminer les documents enrichis par les versions précédentes des ensembles de compétences.

## Gestion du cache

Le cycle de vie du cache est géré par l'indexeur. Si la propriété `cache` définie sur l'indexeur a une valeur null ou si la chaîne de connexion est modifiée, le cache existant est supprimé lors de la prochaine exécution de l'indexeur. Le cycle de vie du cache est également lié au cycle de vie de l'indexeur. Si un indexeur est supprimé, le cache associé est également supprimé.

Alors que l'enrichissement incrémentiel est conçu pour détecter les modifications et y répondre sans aucune intervention de votre part, vous pouvez utiliser certains paramètres pour remplacer les comportements par défaut :

- Classer les nouveaux documents par ordre de priorité
- Ignorer les vérifications des ensembles de compétences
- Ignorer les vérifications des sources de données
- Forcer l'évaluation des ensembles de compétences

### Classer les nouveaux documents par ordre de priorité

Définissez la propriété `enableReprocessing` pour contrôler le traitement des documents entrants déjà représentés dans le cache. Lorsque la valeur est `true` (valeur par défaut), les documents qui se trouvent déjà dans le cache sont traités à nouveau lorsque vous réexécutez l'indexeur, en supposant que la mise à jour de vos compétences affecte ce document.

Lorsque la valeur est `false`, les documents existants ne sont pas retraités, ce qui permet de hiérarchiser efficacement le nouveau contenu entrant par rapport au contenu existant. Vous devez uniquement définir `enableReprocessing` sur la valeur `false` de manière temporaire. Pour garantir la cohérence dans le corpus, la valeur de `enableReprocessing` doit être `true` la plupart du temps, afin que tous les documents, nouveaux ou existants, soient valides conformément à la définition actuelle de l'ensemble de compétences.

### Ignorer l'évaluation des ensembles de compétences

La modification d'un ensemble de compétences et le retraitement de cet ensemble de compétences vont généralement de pair. Toutefois, certaines modifications des ensembles de compétences ne n'entraînent pas de

retraitement (par exemple, le déploiement d'une compétence personnalisée vers un nouvel emplacement ou avec une nouvelle clé d'accès). La plupart du temps, il s'agit de modifications périphériques qui n'ont pas d'impact réel sur la substance des ensembles de compétences.

Si vous savez que la modification apportée à l'ensemble de compétences est en effet superficielle, vous pouvez remplacer l'évaluation des compétences en définissant le paramètre `disableCacheReprocessingChangeDetection` sur `true` :

1. Appelez Mettre à jour les compétences et modifiez la définition de l'ensemble de compétences.
2. Ajoutez le paramètre `disableCacheReprocessingChangeDetection=true` à la requête.
3. Envoyer la modification.

Ce paramètre garantit que seules les mises à jour de la définition de l'ensemble de compétences sont validées et que le changement n'est pas évalué en ce qui concerne ses effets sur le corpus existant.

L'exemple suivant illustre une requête de mise à jour de l'ensemble de compétences avec le paramètre :

```
PUT https://customerdemos.search.windows.net/skillsets/callcenter-text-skillset?api-version=2020-06-30-Preview&disableCacheReprocessingChangeDetection=true
```

### Ignorer les vérifications de validation de la source de données

La plupart des modifications apportées à la définition de source de données invalident le cache. Toutefois, dans les scénarios où vous savez qu'une modification ne doit pas invalider le cache, par exemple, la modification d'une chaîne de connexion ou la rotation de la clé sur le compte de stockage, ajoutez le paramètre `ignoreResetRequirement` à la mise à jour de la source de données. La définition de ce paramètre sur `true` permet à la validation de poursuivre sans déclencher de réinitialisation qui entraînerait la reconstruction de tous les objets et leur remplissage complet.

```
PUT https://customerdemos.search.windows.net/datasources/callcenter-ds?api-version=2020-06-30-Preview&ignoreResetRequirement=true
```

### Forcer l'évaluation des ensembles de compétences

L'objectif du cache est d'éviter les traitements inutiles, mais supposons que vous apportiez une modification à une compétence que l'indexeur ne détecte pas (par exemple, en modifiant un texte dans du code externe, par exemple une compétence personnalisée).

Dans ce cas, vous pouvez utiliser les [compétences de réinitialisation](#) pour forcer le retraitement d'une compétence en particulier, y compris les compétences en aval qui dépendent du résultat de cette compétence. Cette API accepte la requête POST avec une liste de compétences qui doit être invalidée et marquée pour retraitement. Après avoir réinitialisé les compétences, exécutez l'indexeur pour appeler le pipeline.

## Détection des changements

Une fois que vous avez activé un cache, l'indexeur évalue les modifications apportées à la composition de votre pipeline pour déterminer le contenu qui peut être réutilisé et celui qui nécessite un retraitement. Cette section énumère les modifications qui invalident directement le cache, suivies des modifications qui déclenchent un traitement incrémentiel.

### Modifications qui invalident le cache

Un changement invalidant est un changement qui rend la totalité du cache non valide. La mise à jour de votre source de données est un exemple de changement invalidant. Voici la liste complète des changements qui invalident votre cache :

- Changement de votre type de source de données

- Changement du conteneur de source de données
- Informations d'identification de la source de données
- Stratégie de détection des changements apportés à la source de données
- Stratégie de détection de la suppression des sources de données
- Mappages de champs de l'indexeur
- Paramètres d'indexeur
  - Mode d'analyse
  - Extensions de noms de fichiers exclues
  - Extensions de noms de fichiers indexées
  - Indexer les métadonnées de stockage uniquement pour les documents volumineux
  - En-têtes de texte délimité
  - Délimiteur de texte délimité
  - Racine du document
  - Action d'image (changements apportés au mode d'extraction des images)

### **Modifications qui déclenchent un traitement incrémentiel**

Le traitement incrémentiel évalue la définition de votre ensemble de compétences et détermine les compétences à réexécuter, en mettant à jour de manière selective les parties affectées de l'arborescence du document. Voici la liste complète des modifications entraînant un enrichissement incrémentiel :

- Une compétence de l'ensemble de compétences est d'un type distinct. Mise à jour du type OData de la compétence
- Mise à jour des paramètres spécifiques à une compétence, par exemple l'URL, les paramètres par défaut ou tout autre paramètre
- Changement des sorties d'une compétence, la compétence retourne des sorties supplémentaires ou distinctes
- Mises à jour de compétences entraînant un changement d'ancêtres, par exemple un changement du chaînage des compétences entrées de compétences
- Toute invalidation de compétence amont, si une compétence qui fournit une entrée à cette compétence est mise à jour
- Mises à jour de l'emplacement de projection de la base de connaissances, qui entraînent une reprojection des documents
- Apport de changements aux projections de la base de connaissances, qui entraînent une reprojection des documents
- Changements des mappages de champs de sortie d'un indexeur, qui entraînent une reprojection des documents dans l'index

## Informations de référence sur l'API

La version `2020-06-30-Preview` de l'API REST fournit un enrichissement incrémentiel par le biais de propriétés supplémentaires sur des indexeurs. Des ensembles de compétences et des sources de données peuvent utiliser la version généralement disponible. En plus de la documentation de référence, consultez [Configurer la mise en cache pour l'enrichissement incrémentiel](#) pour plus d'informations sur la façon d'appeler les API.

- [Créer un indexeur \(api-version=2020-06-30-Preview\)](#)
- [Mettre à jour un indexeur \(api-version=2020-06-30-Preview\)](#)
- [Mettre à jour un ensemble de compétences \(api-version=2020-06-30\)](#) (nouveau paramètre d'URI sur la requête)
- [Réinitialiser les compétences \(api-version=2020-06-30\)](#)

- Indexeurs de base de données (SQL Azure, Cosmos DB). Certains indexeurs récupèrent les données par le biais de requêtes. Pour les requêtes qui récupèrent des données, [Mettre à jour la source de données](#) prend en charge un nouveau paramètre de requête `ignoreResetRequirement`, qui doit avoir la valeur `true` quand votre action de mise à jour ne doit pas invalider le cache.

Utilisez `ignoreResetRequirement` avec modération, car cela peut entraîner une incohérence involontaire de vos données, qui n'est pas facilement détectée.

## Étapes suivantes

L'enrichissement incrémentiel est une fonctionnalité puissante qui étend le suivi aux ensembles de compétences et à l'enrichissement par IA. L'enrichissement incrémentiel permet de réutiliser le contenu traité existant au fur et à mesure que vous itérez au sein de la conception de compétences.

En guise d'étape suivante, activez la mise en cache sur un indexeur existant ou ajoutez un cache lors de la définition d'un nouvel indexeur.

[Configurer la mise en cache pour l'enrichissement incrémentiel](#)

# Sécurité dans Recherche cognitive Azure - Vue d'ensemble

04/10/2020 • 23 minutes to read • [Edit Online](#)

Cet article décrit les principales fonctionnalités de sécurité de Recherche cognitive Azure qui peuvent protéger le contenu et les opérations.

- Au niveau de la couche de stockage, le chiffrement au repos est intégré pour l'ensemble du contenu géré par le service enregistré sur le disque, y compris les index, les cartes de synonymes et les définitions d'indexeurs, de sources de données et d'ensembles de compétences. Recherche cognitive Azure prend également en charge l'ajout de clés gérées par le client (CMK) pour le chiffrement supplémentaire du contenu indexé. Pour les services créés après le 1er août 2020, le chiffrement CMK s'étend aux données sur les disques temporaires, pour le double chiffrement complet du contenu indexé.
- La sécurité du trafic entrant protège le point de terminaison du service Recherche à des niveaux de sécurité plus élevés : depuis des clés API sur la demande à des règles de trafic entrant dans le pare-feu et à des points de terminaison privés qui protègent intégralement votre service de l'Internet public.
- La sécurité du trafic sortant s'applique aux indexeurs qui extraient du contenu de sources externes. Pour les demandes sortantes, configurez une identité managée pour effectuer une recherche dans un service approuvé lors de l'accès aux données dans Stockage Azure, Azure SQL, Cosmos DB ou d'autres sources de données Azure. Une identité managée est un substitut pour les informations d'identification ou les clés d'accès sur la connexion. La sécurité du trafic sortant n'est pas abordée dans cet article. Pour plus d'informations sur cette fonctionnalité, consultez [Se connecter à une source de données en utilisant une identité managée](#).

Regardez cette vidéo rapide pour obtenir une vue d'ensemble de l'architecture de la sécurité et de chaque catégorie de fonctionnalités.

## Transmissions et stockage chiffrés

Dans Recherche cognitive Azure, le chiffrement commence aux connexions et aux transmissions, et s'étend au contenu stocké sur disque. Pour les services de recherche sur l'Internet public, Recherche cognitive Azure écoute sur le port HTTPS 443. Toutes les connexions client-à-service utilisent le chiffrement TLS 1.2. Les versions antérieures (1.0 et 1.1) ne sont pas prises en charge.

Pour les données gérées en interne par le service de recherche, le tableau suivant décrit les [modèles de chiffrement de données](#). Certaines fonctionnalités, telles que la base de connaissances, l'enrichissement incrémentiel et l'indexation basée sur un indexeur, lisent ou écrivent dans des structures de données dans d'autres services Azure. Ces services disposent de leur propre niveau de prise en charge du chiffrement, distinct de Recherche cognitive Azure.

MODÈLE	CLÉS	SPÉCIFICATIONS	RESTRICTIONS	S'APPLIQUE À
--------	------	----------------	--------------	--------------

Modèle	Clés	Spécifications	Restrictions	S'applique à
chiffrement côté serveur	Clés managées par Microsoft	Aucune (intégré)	Aucune, disponible pour tous les niveaux de service, dans toutes les régions, pour le contenu créé après le 24 janvier 2018.	Contenu (index et cartes de synonymes) et définitions (indexeurs, sources de données, ensembles de compétences)
chiffrement côté serveur	clés gérées par le client	Azure Key Vault	Disponible pour les niveaux de service facturables, dans toutes les régions, pour le contenu créé après janvier 2019.	Contenu (index et cartes de synonymes) sur les disques de données
double chiffrement côté serveur	clés gérées par le client	Azure Key Vault	Disponible sur les niveaux de service facturables, dans certaines régions, sur les services de recherche après le 1er août 2020.	Contenu (index et cartes de synonymes) sur les disques de données et les disques temporaires

### Clés gérées par le service

Le chiffrement géré par le service est une opération interne de Microsoft qui est basée sur [Azure Storage Service Encryption](#) et qui utilise le [chiffrement AES](#) 256 bits. Il se produit automatiquement lors de toutes les indexations, y compris lors des mises à jour incrémentielles des index qui ne sont pas entièrement chiffrés (créés avant janvier 2018).

### Clés gérées par le client (CMK)

Les clés gérées par le client nécessitent un service facturable supplémentaire, Azure Key Vault, qui peut se trouver dans une autre région que l'instance Recherche cognitive Azure, mais qui doit être sous le même abonnement. L'activation du chiffrement CMK a pour effet d'augmenter la taille de l'index et dégrader les performances des requêtes. Sur la base des observations effectuées à ce jour, vous pouvez vous attendre à une augmentation de 30 à 60 % des temps de requête, même si les performances réelles varient en fonction de la définition d'index et des types de requêtes. En raison de cet incidence sur les performances, nous vous recommandons de n'activer cette fonctionnalité que sur les index qui en ont réellement besoin. Pour plus d'informations, consultez [Configurer des clés de chiffrement gérées par le client dans Recherche cognitive Azure](#).

### Double chiffrement

Dans Recherche cognitive Azure, le double chiffrement est une extension de CMK. Il s'agit d'un chiffrement à deux reprises (une fois par CMK, et une nouvelle fois par des clés gérées par le service) et de portée globale, comprenant le stockage à long terme qui est écrit sur un disque de données et le stockage à court terme écrit sur des disques temporaires. La différence entre CMK avant et après le 1er août 2020, et ce qui fait de CMK une fonctionnalité de double chiffrement dans Recherche cognitive Azure, est le chiffrement supplémentaire de données au repos sur les disques temporaires.

Le double chiffrement est actuellement disponible sur les nouveaux services créés dans ces régions après le 1er août :

- USA Ouest 2
- USA Est
- États-Unis - partie centrale méridionale
- Gouvernement américain - Virginie

- Gouvernement des États-Unis – Arizona

## Sécurité du trafic entrant et protection des points de terminaison

Les fonctionnalités de sécurité du trafic entrant protègent le point de terminaison du service de recherche via des niveaux croissants de sécurité et de complexité. Premièrement, toutes les demandes nécessitent une clé API pour l'accès authentifié. Deuxièmement, vous avez la possibilité de définir des règles de pare-feu qui limitent l'accès à des adresses IP spécifiques. Pour une protection avancée, une troisième option consiste à activer Azure Private Link pour protéger votre point de terminaison de service de tout le trafic Internet.

### Accès public avec des clés API

Par défaut, un service de recherche est accessible via le cloud public, en utilisant une authentification basée sur des clés pour l'administration ou l'accès aux requêtes au point de terminaison du service de recherche. Une clé API est une chaîne composée de nombres et de lettres générée de manière aléatoire. Le type de clé (admin ou requête) détermine le niveau d'accès. La soumission d'une clé valide est considérée comme la preuve que la requête provient d'une entité approuvée.

Il existe deux niveaux d'accès à votre service de recherche, qui sont activés par les deux clés API suivantes :

- Clé d'administration (autorise l'accès en lecture-écriture pour les opérations [créer - lire - mettre à jour - supprimer](#) sur le service de recherche)
- Clé de requête (autorise l'accès en lecture seule à la collection documents d'un index)

Des *clés d'administration* sont créées une fois le service approvisionné. Bien qu'il existe deux clés d'administration, désignées comme *principale* et *secondaire*, celles-ci sont en fait interchangeables. Chaque service dispose de deux clés Admin que vous pouvez interchanger sans perdre l'accès à votre service. Vous pouvez [régénérer un clé d'administration](#) périodiquement conformément aux meilleures pratiques de sécurité Azure, mais ne pouvez pas en ajouter au nombre total de clés d'administration. Il y a, au maximum, deux clés d'administration par service de recherche.

Des *clés de requête* sont créées en fonction des besoins pour les applications clientes qui émettent des requêtes. Vous pouvez créer, au maximum, 50 clés de ce type. Dans le code d'application, vous spécifiez l'URL de recherche et une clé d'API de requête pour autoriser l'accès en lecture seule à la collection de documents d'un index spécifique. Ensemble, le point de terminaison, une clé API pour un accès en lecture seule et un index cible définissent le niveau de portée et d'accès de la connexion à partir de votre application cliente.

L'authentification est requise à chaque requête, chaque requête étant composée d'une clé obligatoire, d'une opération et d'un objet. Quand ils sont chaînés, les deux niveaux d'autorisation (complet ou en lecture seule) et le contexte (par exemple, une opération de requête sur un index) sont suffisants pour fournir une sécurité couvrant l'ensemble des opérations de service. Pour plus d'informations sur les clés, consultez [Créer et gérer des clés de l'api](#).

### Accès restreint à des adresses IP

Pour contrôler davantage l'accès à votre service de recherche, vous pouvez créer des règles de pare-feu de trafic entrant qui autorisent l'accès à une adresse IP spécifique ou à une plage d'adresses IP. Toutes les connexions clientes doivent être effectuées via une adresse IP autorisée, sans quoi la connexion est refusée.

Vous pouvez utiliser le portail pour [configurer l'accès du trafic entrant](#).

Vous pouvez aussi utiliser les API REST de gestion. L'API version 13-03-2020 avec le paramètre `IpRule` vous permet de restreindre l'accès à votre service en identifiant les adresses IP individuellement ou dans une plage, autorisées à accéder à votre service de recherche.

### Point de terminaison privé (pas de trafic Internet)

Un [point de terminaison privé](#) pour Recherche cognitive Azure permet à un client d'un [réseau virtuel](#) d'accéder de

façon sécurisée aux données d'un index de recherche grâce à une [liaison privée](#).

Le points de terminaison privé utilise une adresse IP de l'espace d'adressage du réseau virtuel pour les connexions à votre service de recherche. Le trafic entre le client et le service Search traverse le réseau virtuel et une liaison privée sur le réseau principal de Microsoft, ce qui élimine l'exposition sur l'Internet public. Un réseau virtuel permet une communication sécurisée entre des ressources, avec votre réseau local ainsi qu'avec Internet.

Bien que cette solution soit la plus sécurisée, l'utilisation de services supplémentaires représente un coût supplémentaire : veillez donc à avoir une compréhension claire des avantages avant de la mettre en place. Pour plus d'informations sur les coûts, consultez la [page Tarification](#). Pour plus d'informations sur la façon dont ces composants fonctionnent ensemble, regardez la vidéo en haut de cet article. L'option du point de terminaison privé est présentée à partir de 5:48 dans la vidéo. Pour obtenir des instructions sur la configuration du point de terminaison, consultez [Créer un point de terminaison privé pour Recherche cognitive Azure](#).

## Accès aux index

Dans Recherche cognitive Azure, les index individuels ne sont pas des objets sécurisables. Au lieu de cela, l'accès aux index est déterminé au niveau de la couche du service (accès en lecture ou en écriture au service) et du contexte d'une opération.

Pour l'accès de l'utilisateur final, vous pouvez structurer les demandes de requête pour établir la connexion à l'aide d'une clé de requête, qui configure toutes les demandes en mode de lecture seule et qui inclut l'index spécifique utilisé par votre application. Dans une demande de requête, il est impossible de joindre des index ou d'accéder simultanément à plusieurs index. Ainsi, toutes les demandes ciblent un index unique par définition. Par conséquent, la structure de la demande de requête proprement dite (une clé plus un index unique cible) définit la limite de sécurité.

Il n'existe aucune différence entre l'accès administrateur et l'accès développeur aux index : tous deux doivent disposer d'un accès en écriture pour pouvoir créer, supprimer et mettre à jour des objets gérés par le service. Toute personne disposant d'une clé d'administration pour votre service peut lire, modifier ou supprimer un index de ce service. En ce qui concerne la protection contre la suppression accidentelle ou malveillante d'index, votre contrôle de code source en interne pour les ressources de code est la solution appropriée pour annuler des suppressions ou des modifications d'index indésirables. Recherche cognitive Azure dispose d'un système de basculement dans le cluster pour garantir sa disponibilité, mais il ne stocke pas et n'exécute pas le code propriétaire que vous avez utilisé pour créer ou charger des index.

Pour les solutions d'architecture mutualisée qui nécessitent des limites de sécurité au niveau des index, ces solutions incluent généralement un niveau intermédiaire, que les clients utilisent pour gérer l'isolation des index. Pour plus d'informations sur les cas d'usage d'architecture mutualisée, consultez [Modèles de conception pour les applications SaaS mutualisées et Recherche cognitive Azure](#).

## Accès utilisateur

La façon dont un utilisateur accède à un index et à d'autres objets est déterminée par le type de clé API sur la demande. La plupart des développeurs créent et affectent des [clés de requête](#) pour les demandes de recherche du côté client. Une clé de requête accorde un accès en lecture au contenu pouvant faire l'objet d'une recherche dans l'index.

Si vous avez besoin d'un contrôle précis par utilisateur sur les résultats de la recherche, vous pouvez créer des filtres de sécurité sur vos requêtes, en retournant des documents associés à une identité de sécurité donnée. Au lieu des rôles prédéfinis et des attributions de rôles, le contrôle d'accès basé sur l'identité est implémenté en tant que *filtre* qui limite les résultats de recherche de documents et de contenu en fonction des identités. Le tableau suivant décrit les deux approches permettant de filtrer les résultats de recherche de contenu non autorisé.

APPROCHE	DESCRIPTION
Filtrage de sécurité basé sur les filtres d'identité	Cet article décrit le workflow de base pour l'implémentation du contrôle d'accès basé sur l'identité de l'utilisateur. Il décrit l'ajout d'identificateurs de sécurité à un index, puis le filtrage relatif à ce champ qui permet d'omettre les résultats de contenu non autorisé.
Filtrage de sécurité basé sur les identités Azure Active Directory	Cet article développe l'article précédent, en indiquant les étapes à suivre pour récupérer des identités d'Azure Active Directory (AAD), l'un des <a href="#">services gratuits</a> de la plateforme cloud Azure.

## Droits d'administration

Le [contrôle d'accès en fonction du rôle Azure \(Azure RBAC\)](#) est un système d'autorisation basé sur [Azure Resource Manager](#) pour l'approvisionnement de ressources Azure. Dans Recherche cognitive Azure, Resource Manager est utilisé pour créer ou supprimer le service, gérer les clés API et mettre à l'échelle le service. Ainsi, les attributions de rôles Azure déterminent qui peut effectuer ces tâches, qu'elles utilisent le [portail](#), [PowerShell](#) ou les [API REST de gestion](#).

En revanche, les droits d'administrateur sur le contenu hébergé sur le service, comme la possibilité de créer ou de supprimer un index, sont conférés via des clés API, comme décrit dans la [section précédente](#).

### TIP

En utilisant des mécanismes à l'échelle d'Azure, vous pouvez verrouiller un abonnement ou une ressource pour empêcher la suppression accidentelle ou non autorisée de votre service de recherche par les utilisateurs disposant de droits d'administration. Pour plus d'informations, consultez [Verrouiller les ressources pour en empêcher la suppression](#).

## Certifications et conformité

Recherche cognitive Azure a été certifié conforme à plusieurs standards mondiaux, régionaux et spécifiques à des secteurs pour le cloud public et Azure Government. Pour obtenir la liste complète, téléchargez le livre blanc [Microsoft Azure Compliance Offerings](#) depuis la page des rapports d'audit officiels.

Pour la conformité, vous pouvez utiliser [Azure Policy](#) pour mettre en œuvre les meilleures pratiques de haute sécurité d'[Azure Security Benchmark](#). Azure Security Benchmark est un ensemble de recommandations de sécurité, codifiées en contrôles de sécurité qui correspondent aux principales actions que vous devez prendre pour atténuer les menaces pesant sur les services et les données. Il existe actuellement 11 contrôles de sécurité, dont la [Sécurité réseau](#), [Journalisation et surveillance](#), et [Protection des données](#), pour n'en nommer que quelques-uns.

Azure Policy est une capacité intégrée à Azure qui vous permet de gérer la conformité de plusieurs normes, y compris celles d'Azure Security Benchmark. Pour les critères de référence bien connus, Azure Policy fournit des définitions intégrées qui fournissent à la fois des critères et une réponse actionnable en cas de non-conformité.

Pour Recherche cognitive Azure, il existe actuellement une définition intégrée. Elle concerne la journalisation des diagnostics. Grâce à cette intégration, vous pouvez attribuer une stratégie qui identifie tout service de recherche auquel il manque la journalisation des diagnostics, puis l'active. Pour plus d'informations, consultez [Contrôles de conformité réglementaire d'Azure Policy pour Recherche cognitive Azure](#).

## Voir aussi

- [Concepts de base de la sécurité Azure](#)
- [Sécurité Azure](#)

- Centre de sécurité Azure

# Contrôles de conformité réglementaire d'Azure Policy pour Recherche cognitive Azure

04/10/2020 • 5 minutes to read • [Edit Online](#)

Si vous utilisez [Azure Policy](#) pour appliquer les recommandations du [benchmark de sécurité Azure](#), vous savez probablement déjà que vous pouvez créer des stratégies pour identifier et corriger les services non conformes. Ces stratégies peuvent être personnalisées ou basées sur des définitions intégrées qui fournissent des critères de conformité et des solutions appropriées s'inscrivant dans les bonnes pratiques bien définies.

Pour le service Recherche cognitive Azure, il existe actuellement une définition intégrée, indiquée ci-dessous, que vous pouvez utiliser dans une attribution de stratégie. Elle est destinée à la journalisation et la supervision. Quand vous utilisez cette définition intégrée dans une [stratégie que vous créez](#), le système recherche les services de recherche qui n'ont pas de [journalisation des diagnostics](#), puis l'active en conséquence.

La [conformité réglementaire Azure Policy](#) fournit des définitions d'initiatives créées et gérées par Microsoft, qui sont dites *intégrées*, pour les **domaines de conformité** et les **contrôles de sécurité** associés à différents standards de conformité. Cette page liste les **domaines de conformité** et les **contrôles de sécurité** pour Recherche cognitive Azure. Vous pouvez affecter les initiatives intégrées pour un **contrôle de sécurité** individuellement, pour rendre vos ressources Azure conformes au standard spécifique.

Le titre de chaque définition de stratégie intégrée est un lien vers la définition de la stratégie dans le portail Azure. Utilisez le lien de la colonne **Version de la stratégie** pour voir la source dans le [dépôt GitHub Azure Policy](#).

## IMPORTANT

Chaque contrôle ci-dessous est associé à une ou plusieurs définitions [Azure Policy](#). Ces stratégies peuvent vous aider à évaluer la conformité avec le contrôle ; toutefois, il n'existe pas souvent de correspondance un-à-un ou parfaite entre un contrôle et une ou plusieurs stratégies. Ainsi, la **conformité** dans Azure Policy fait uniquement référence aux stratégies elles-mêmes ; cela ne garantit pas que vous êtes entièrement conforme à toutes les exigences d'un contrôle. En outre, la norme de conformité comprend des contrôles qui ne sont traités par aucune définition Azure Policy pour l'instant. Par conséquent, la conformité dans Azure Policy n'est qu'une vue partielle de l'état de conformité global. Les associations entre les contrôles et les définitions de conformité réglementaire Azure Policy pour ces normes de conformité peuvent changer au fil du temps.

## Benchmark de sécurité Azure

Le [benchmark de sécurité Azure](#) fournit des recommandations sur la façon dont vous pouvez sécuriser vos solutions cloud sur Azure. Pour voir comment ce service correspond totalement au benchmark de sécurité Azure, consultez les [fichiers de correspondance du benchmark de sécurité Azure](#).

Pour passer en revue la façon dont les composants intégrés Azure Policy disponibles pour tous les services Azure correspondent à ce standard de conformité, consultez [Conformité réglementaire Azure Policy – Benchmark de sécurité Azure](#).

DOMAIN	ID DU CONTRÔLE	TITRE DU CONTRÔLE	POLICY (PORTAIL AZURE)	VERSION DE LA STRATÉGIE (GITHUB)
Journalisation et supervision	2.3	Activer la journalisation d'audit pour les ressources Azure	<a href="#">Les journaux de diagnostic dans les services Search doivent être activés</a>	3.0.0

DOMAIN	ID DU CONTRÔLE	TITRE DU CONTRÔLE	POLICY	VERSION DE LA STRATÉGIE
--------	----------------	-------------------	--------	-------------------------

## HIPAA HITRUST 9.2

Pour voir comment les composants intégrés Azure Policy disponibles pour tous les services Azure correspondent à ce standard de conformité, consultez [Conformité réglementaire Azure Policy – HIPAA HITRUST 9.2](#). Pour plus d'informations sur cette norme de conformité, consultez [HIPAA HITRUST 9.2](#).

DOMAIN	ID DU CONTRÔLE	TITRE DU CONTRÔLE	POLICY (PORTAIL AZURE)	VERSION DE LA STRATÉGIE (GITHUB)
Journalisation d'audit	1208.09aa3System.1 - 09.aa	Les journaux d'audit sont conservés pour les activités de gestion, le démarrage, l'arrêt et les erreurs du système et des applications, les changements de fichier et les changements de stratégie de sécurité.	<a href="#">Les journaux de diagnostic dans les services Search doivent être activés</a>	3.0.0

## Étapes suivantes

- Apprenez-en davantage sur la [Conformité réglementaire d'Azure Policy](#).
- Consultez les définitions intégrées dans le [dépôt Azure Policy de GitHub](#).

# Accès de l'indexeur aux sources de données à l'aide des fonctionnalités de sécurité réseau Azure

04/10/2020 • 15 minutes to read • [Edit Online](#)

Les indexeurs Recherche cognitive Azure peuvent effectuer des appels sortants vers différentes ressources Azure lors de l'exécution. Cet article explique les concepts liés à l'accès de l'indexeur aux ressources lorsque ces ressources sont protégées par des pare-feu IP, des points de terminaison privés et d'autres mécanismes de sécurité au niveau du réseau. Les types de ressources auxquels un indexeur peut accéder dans le cadre d'une exécution classique sont répertoriés dans le tableau ci-dessous.

RESSOURCE	OBJECTIF AU SEIN DE L'EXÉCUTION DE L'INDEXEUR
Stockage Azure (objets blob, tables, ADLS Gen 2)	Source de données
Stockage Azure (objets blob, tables)	Ensembles de compétences (mise en cache de documents enrichis et stockage des projections de la base de connaissances)
Azure Cosmos DB (différentes API)	Source de données
Azure SQL Database	Source de données
SQL Server sur des machines virtuelles IaaS Azure	Source de données
Instances managées SQL	Source de données
Azure Functions	Hôte pour les compétences API web personnalisées
Cognitive Services	Joint aux compétences qui seront utilisées pour facturer l'enrichissement au-delà de la limite de 20 documents gratuits

## NOTE

La ressource Cognitive Services jointe à une compétences est utilisée pour la facturation, selon les enrichissements effectués et écrits dans l'index de recherche. Elle n'est pas utilisée pour accéder aux API Cognitive Services. L'accès du pipeline d'enrichissement d'indexeur aux API Cognitive Services intervient via un canal de communication sécurisé, où les données sont fortement chiffrées en transit et ne sont jamais stockées au repos.

Les clients peuvent sécuriser ces ressources via plusieurs mécanismes d'isolement réseau proposés par Azure. À l'exception de la ressource Cognitive Services, les indexeurs ont une capacité d'accès limitée aux autres ressources, même si elles sont isolées du réseau, comme indiqué dans le tableau ci-dessous.

RESSOURCE	RESTRICTION D'ADRESSE IP	POINT DE TERMINAISON PRIVÉ
Stockage Azure (objets blob, tables, ADLS Gen 2)	Pris en charge uniquement si le compte de stockage et le service de recherche se trouvent dans des régions différentes	Prise en charge

RESSOURCE	RESTRICTION D'ADRESSE IP	POINT DE TERMINAISON PRIVÉ
Azure Cosmos DB - API SQL	Prise en charge	Prise en charge
Azure Cosmos DB - API Cassandra, Mongo et Gremlin	Prise en charge	Non pris en charge
Azure SQL Database	Prise en charge	Prise en charge
SQL Server sur des machines virtuelles IaaS Azure	Prise en charge	N/A
Instances managées SQL	Prise en charge	N/A
Azure Functions	Prise en charge	Pris en charge uniquement pour certaines références SKU d'Azure Functions

#### NOTE

En plus des options répertoriées ci-dessus, pour les comptes Stockage Azure sécurisés sur le réseau, les clients peuvent tirer parti du fait que la Recherche cognitive Azure est un [service Microsoft approuvé](#). Cela signifie qu'un service de recherche spécifique peut contourner les restrictions de réseau virtuel ou d'adresse IP sur le compte de stockage et accéder aux données du compte de stockage moyennant l'activation du contrôle d'accès en fonction du rôle sur ce compte. Les détails sont disponibles dans le [guide pratique](#). Cette option peut être privilégiée par rapport à l'itinéraire de restriction d'adresse IP, s'il est impossible de déplacer le compte de stockage ou le service de recherche vers une autre région.

Lorsque vous choisissez le mécanisme d'accès sécurisé qu'un indexeur doit utiliser, prenez en compte les contraintes suivantes :

- [Les points de terminaison de service](#) ne sont pas pris en charge pour les ressources Azure.
- Un service de recherche ne peut pas être approvisionné dans un réseau virtuel spécifique ; cette fonctionnalité n'est pas offerte par Recherche cognitive Azure.
- Lorsque les indexeurs utilisent des points de terminaison privés (sortants) pour accéder aux ressources, des [frais de liaison privée](#) supplémentaires peuvent s'appliquer.

## Environnement d'exécution des indexeurs

Les indexeurs Recherche cognitive Azure peuvent extraire efficacement du contenu à partir de sources de données, en ajoutant des enrichissements au contenu extrait, voire en générant des projections avant d'écrire les résultats dans l'index de recherche. Selon le nombre de responsabilités attribuées à un indexeur, il peut s'exécuter dans l'un des deux environnements suivants :

- Environnement privé pour un service de recherche spécifique. Les indexeurs qui s'exécutent dans de tels environnements partagent des ressources avec d'autres charges de travail (par exemple, l'indexation initiée par le client ou l'interrogation de la charge de travail). En général, seuls les indexeurs nécessitant peu de ressources (par exemple, n'utilisant pas d'ensemble de compétences) s'exécutent dans cet environnement.
- Environnement multilocataire hébergeant des indexeurs gourmands en ressources, tels que ceux dotés d'un ensemble de compétences. Les ressources les plus gourmandes s'exécutent dans cet environnement pour offrir des performances optimales tout en garantissant la disponibilité des ressources du service de recherche pour d'autres charges de travail. Cet environnement multilocataire est géré et sécurisé par Recherche cognitive Azure, sans frais supplémentaires pour le client.

Recherche cognitive Azure détermine l'environnement le plus adapté à l'exécution de d'un indexeur donné.

## Octroi d'accès aux plages d'adresses IP de l'indexeur

Si la ressource à laquelle votre indexeur tente d'accéder est limitée à un certain nombre de plages d'adresses IP, vous devez développer l'ensemble pour inclure les plages d'adresses IP à partir desquelles une demande d'indexeur peut provenir. Comme indiqué ci-dessus, il existe deux environnements possibles dans lesquels les indexeurs s'exécutent et à partir desquels les demandes d'accès peuvent provenir. Vous devez ajouter les adresses IP des **deux environnements** pour permettre le bon fonctionnement de l'indexeur.

- Pour obtenir l'adresse IP de l'environnement privé spécifique du service de recherche, `nslookup` (ou `ping`) le nom de domaine complet (FQDN) de votre service de recherche. Le nom de domaine complet d'un service de recherche dans le cloud public peut, par exemple, être `<service-name>.search.windows.net`. Ces informations sont disponibles sur le portail Azure.
- Les adresses IP des environnements multilocataires sont disponibles via la balise de service `AzureCognitiveSearch`. Les [balises de service Azure](#) disposent d'une plage d'adresses IP publiée pour chaque service, ce qui est possible via une [API de détection \(préversion\)](#) ou un [fichier JSON téléchargeable](#). Dans les deux cas, les plages d'adresses IP sont décomposées par région. Vous pouvez choisir uniquement les plages d'adresses IP attribuées à la région dans laquelle votre service de recherche est approvisionné.

Pour certaines sources de données, la balise de service elle-même peut être utilisée directement plutôt que d'énumérer la liste de plages d'adresses IP (l'adresse IP du service de recherche doit toujours être utilisée explicitement). Ces sources de données restreignent l'accès au moyen de la configuration d'une [règle de groupe de sécurité réseau](#), qui prend en charge l'ajout d'une balise de service en mode natif, à la différence des règles IP telles que celles proposées par Stockage Azure, CosmosDB, Azure SQL, etc. Les sources de données qui prennent en charge l'utilisation de la balise de service `AzureCognitiveSearch` sont les suivantes :

- [SQL Server sur machines virtuelles IaaS](#)
- [Instances managées SQL](#)

Des détails sont fournis dans le [guide pratique](#).

## Octroi de l'accès via des points de terminaison privés

Les indexeurs peuvent utiliser [des points de terminaison privés](#) afin d'accéder à des ressources dont l'accès est verrouillé pour sélectionner des réseaux virtuels ou pour lesquels aucun accès public n'est activé. Cette fonctionnalité est réservée aux services payants, avec des limites sur le nombre de points de terminaison privés créés. Pour plus d'informations sur ces limites, consultez la [page relative aux limites Recherche Azure](#).

### Étape 1 : Créer un point de terminaison privé vers la ressource sécurisée

Les clients doivent appeler l'opération de gestion de la recherche, l'API [Créer ou mettre à jour la ressource de lien privée partagée](#) afin de créer une connexion de point de terminaison privé à leur ressource sécurisée (par exemple, un compte de stockage). Le trafic qui transite via cette connexion de point de terminaison privé (sortant) provient uniquement du réseau virtuel situé dans l'environnement d'exécution de l'indexeur « privé » spécifique du service de recherche.

Recherche cognitive Azure vérifie que les appelants de cette API disposent des autorisations pour approuver les demandes de connexion de point de terminaison privées à la ressource sécurisée. Par exemple, si vous demandez une connexion de point de terminaison privé à un compte de stockage auquel vous n'avez pas accès, cet appel est rejeté.

### Étape 2 : Approuver la connexion Private Endpoint

Lorsque l'opération (asynchrone) qui crée une ressource de lien privé partagé est terminée, une connexion de point de terminaison privé est créée avec un état « en attente ». Aucun trafic n'est actuellement transmis via la connexion. Le client est censé localiser cette demande sur sa ressource sécurisée et l'approuver. En règle générale, cette opération peut être effectuée via le portail ou l'[API REST](#).

### Étape 3 : Forcer l'exécution des indexeurs dans l'environnement privé

Un point de terminaison privé approuvé permet les appels sortants entre le service de recherche et une ressource présentant des restrictions d'accès au niveau du réseau (par exemple, une source de données de compte de stockage configurée pour être uniquement accessible à partir de certains réseaux virtuels). Cela signifie que tout indexeur capable d'atteindre une telle source de données sur le point de terminaison privé aboutira. Si le point de terminaison privé n'est pas approuvé ou si l'indexeur n'utilise pas la connexion au point de terminaison privé, l'exécution de l'indexeur se terminera par `transientFailure`.

Pour permettre aux indexeurs d'accéder aux ressources par le biais de connexions de point de terminaison privé, il est impératif de définir la valeur `executionEnvironment` de l'indexeur sur `"Private"` afin de veiller à ce que toutes les exécutions de cet indexeur utilisent le point de terminaison privé et ce, en raison du fait que les points de terminaison privés sont approvisionnés dans l'environnement spécifique du service de recherche privé.

```
{  
    "name" : "myindexer",  
    ... other indexer properties  
    "parameters" : {  
        ... other parameters  
        "configuration" : {  
            ... other configuration properties  
            "executionEnvironment": "Private"  
        }  
    }  
}
```

Cette procédure est décrite plus en détail dans le [guide pratique](#). Lorsque vous disposez d'un point de terminaison privé approuvé pour une ressource, les indexeurs définis pour être *privés* tentent d'obtenir l'accès via la connexion au point de terminaison privé.

### Limites

À des fins de performances et de stabilité optimales du service de recherche, des restrictions sont imposées (par référence SKU du service de recherche) comme suit :

- Types d'indexeurs pouvant être définis comme *privés*.
- Nombre de ressources de lien privé partagé pouvant être créées.
- Nombre de types de ressources distincts pour lesquels des ressources de lien privé partagé peuvent être créées.

Ces limites sont documentées dans [Limites de service](#).

# Créer un index de recherche de base dans Recherche cognitive Azure

04/10/2020 • 25 minutes to read • [Edit Online](#)

Dans Recherche cognitive Azure, un *index de recherche* stocke le contenu pouvant faire l'objet d'une recherche utilisé pour le texte intégral et les requêtes filtrées. Un index est défini par un schéma et enregistré dans le service, l'importation des données se faisant dans un deuxième temps.

Les index contiennent des *documents*. Conceptuellement, un document correspond à une unité de données pouvant faire l'objet d'une recherche dans un index. Un détaillant peut posséder un document pour chaque produit, un organisme de presse peut posséder un document par article, et ainsi de suite. Pour comparer avec des éléments de base de données plus familiers, un *index de recherche* correspond à une *table*, et les *documents* équivalent plus ou moins aux *lignes* d'une table.

La structure physique d'un index est déterminée par le schéma, les champs marqués « Possibilité de recherche » donnant lieu à la création d'un index inversé pour ce champ.

Vous pouvez créer un index à l'aide des outils et API suivants :

- Dans le portail Azure, utilisez l'Assistant **Ajouter un index** ou **Importer des données**
- À l'aide de [Create Index \(REST API\)](#)
- À l'aide du [SDK .NET](#)

Il est plus facile d'apprendre avec un outil du portail. Le portail applique des exigences et des règles de schéma pour des types de données spécifiques, telles que l'interdiction de la recherche en texte intégral sur les champs numériques. Une fois que vous disposez d'un index exploitable, vous pouvez effectuer la transition vers le code en extrayant la définition JSON du service à l'aide de [Get Index \(REST API\)](#) et en l'ajoutant à votre solution.

## Workflow recommandé

L'élaboration d'un index final est un processus itératif. Il est courant de commencer par le portail pour créer l'index initial, puis de basculer vers le code pour placer l'index sous le contrôle de code source.

1. Déterminez si vous pouvez utiliser **Importer des données**. L'Assistant effectue une indexation tout-en-un basée sur un indexeur si les données sources proviennent d'un [type de source de données pris en charge dans Azure](#).
2. Si vous ne pouvez pas utiliser **Importer des données**, commencez par **Ajouter un index** pour définir le schéma.



3. Fournissez un nom et une clé utilisés pour identifier de manière unique chaque document de recherche dans l'index. La clé est obligatoire et doit être de type Edm.String. Lors de l'importation, vous devez prévoir de faire correspondre à ce champ un champ unique dans les données sources.

Le portail vous donne un champ `id` pour la clé. Pour remplacer le `id` par défaut, créez un nouveau

champ (par exemple, une nouvelle définition de champ appelée `HotelId`), puis sélectionnez-le dans Clé.

The screenshot shows the 'Add index' page in the Azure portal. At the top, there's a breadcrumb navigation: Home > mydemo >. The main title is 'Add index'. Below it, there's a form with the following fields:

- Index name \***: hotel-index (highlighted with a purple border)
- Key \***: id
- Suggerer name**: (empty)
- Search mode**: (empty dropdown)

Below these are sections for 'Add field', 'Add subfield', and 'Delete'. A table lists four fields:

Field name	Type	Retrievable	Filterable	Sortable	Facetable	Searchable	Analyzer	Suggerer
id	Edm.String	<input checked="" type="checkbox"/>	Standard - Luc...	<input type="checkbox"/>				
HotelName	Edm.String	<input checked="" type="checkbox"/>	English - Micro...	<input type="checkbox"/>				
HotelDescription_EN	Edm.String	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Spanish - Micr...	<input type="checkbox"/>
HotelDescription_ES	Edm.String	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	English - Micro...	<input type="checkbox"/>

At the bottom left is a blue 'Create' button.

- Ajoutez d'autres champs. Le portail vous indique les [attributs de champ](#) disponibles en fonction des types de données. Si vous débutez dans la conception d'index, cela vous sera utile.

Si les données entrantes sont de nature hiérarchique, attribuez le type de données [Type complexe](#) pour représenter les structures imbriquées. L'exemple de jeu de données intégré, Hotels, illustre des types complexes utilisant une adresse (contenant plusieurs sous-champs) qui entretient une relation un-à-un avec chaque hôtel, et une collection complexe Rooms, où plusieurs chambres sont associées à chaque hôtel.

- Attribuez des [analyseurs](#) aux champs de type chaîne avant la création de l'index. Procédez de même pour les [suggesteurs](#) si vous souhaitez activer l'Autocomplétion sur des champs spécifiques.
- Cliquez sur **Créer** pour générer les structures physiques dans votre service de recherche.
- Après la création d'un index, utilisez des commandes supplémentaires pour vérifier les définitions ou ajouter d'autres éléments.

The screenshot shows the 'hotels-sample-index' page in the Azure portal. At the top, there's a breadcrumb navigation: Home > mydemo >. The main title is 'hotels-sample-index'. Below it, there's a toolbar with Save, Discard, Refresh, Create Search App (preview), and Delete buttons. The document count is 50 and the size is 343.67 kB.

Below the toolbar, there are tabs: Search explorer, Fields (highlighted with a red border), CORS, Scoring profiles, and Index Definition (JSON).

Under the 'Fields' tab, there's a 'Suggerer name' input field containing 'sg' and a 'Search mode' dropdown.

Below these are sections for 'Add field', 'Add subfield', and 'Delete'. A table lists ten fields:

Field name	Type	Retrievable	Filterable	Sortable	Facetable	Searchable	Analyzer	Suggerer
HotelId	Edm.String	<input checked="" type="checkbox"/>		<input type="checkbox"/>				
HotelName	Edm.String	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	English - Micro...	<input type="checkbox"/>
Description	Edm.String	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	English - Micro...	<input type="checkbox"/>
Description_fr	Edm.String	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	French - Micros...	<input type="checkbox"/>
Category	Edm.String	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	English - Micro...	<input type="checkbox"/>
Tags	Collection(Edm.String)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	English - Micro...	<input type="checkbox"/>
ParkingIncluded	Edm.Boolean	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>			<input type="checkbox"/>
LastRenovationDate	Edm.DateTi...	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>			<input type="checkbox"/>
Rating	Edm.Double	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>			<input type="checkbox"/>

8. Téléchargez le schéma d'index à l'aide de [Get Index \(REST API\)](#) et d'un outil de test web comme [Postman](#). Vous disposez maintenant d'une représentation JSON de l'index que vous pouvez adapter pour le code.
9. [Chargez votre index avec des données](#). La Recherche cognitive Azure accepte les documents JSON. Pour charger vos données par programmation, vous pouvez utiliser Postman avec des documents JSON dans la charge utile de demande. S'il n'est pas facilement d'exprimer vos données au format JSON, cette étape sera la plus fastidieuse.

Une fois qu'un index est chargé avec des données, la plupart des modifications apportées aux champs existants nécessitent la suppression et la régénération d'un index.

10. Interrogez votre index, examinez les résultats et continuez d'itérer sur le schéma d'index jusqu'à ce que vous commençiez à obtenir les résultats attendus. Pour interroger votre index, vous pouvez utiliser l'[Explorateur de recherche](#) ou Postman.

Pendant le développement, prévoyez des régénérations fréquentes. Comme les structures physiques sont créées dans le service, il est nécessaire de [supprimer et de recréer les index](#) pour la plupart des modifications apportées à une définition de champ existante. Vous pouvez envisager de travailler sur une partie de vos données pour regénérer plus rapidement.

#### TIP

Il est préférable d'utiliser du code plutôt que le portail pour travailler simultanément sur la conception de l'index et l'importation des données. Sinon, les outils tels que [Postman](#) et [l'API REST](#) s'avèrent utiles pour tester la preuve de concept aux phases initiales d'un projet de développement. Vous pouvez apporter des modifications incrémentielles à une définition d'index dans un corps de demande, puis envoyer la demande à votre service pour recréer un index en utilisant un schéma mis à jour.

## Schéma d'index

Un index doit avoir un nom et un champ clé désigné (Edm.string) dans la collection de champs. La [collection de champs](#) correspond généralement à la majeure partie de l'index, dans laquelle chaque champ est nommé, tapé et pourvu de comportements autorisés qui déterminent son utilisation.

Parmi les autres éléments, citons les [suggesteurs](#), les [profils de scoring](#), les [analyseurs utilisés](#) pour traiter les chaînes en jetons selon des règles linguistiques ou d'autres caractéristiques prises en charge par l'analyseur et des paramètres [CORS \(Cross-Origin Remote Scripting\)](#).

```
{  
    "name": (optional on PUT; required on POST) "name_of_index",  
    "fields": [  
        {  
            "name": "name_of_field",  
            "type": "Edm.String | Collection(Edm.String) | Edm.Int32 | Edm.Int64 | Edm.Double | Edm.Boolean |  
            Edm.DateTimeOffset | Edm.GeographyPoint",  
            "searchable": true (default where applicable) | false (only Edm.String and Collection(Edm.String)  
            fields can be searchable),  
            "filterable": true (default) | false,  
            "sortable": true (default where applicable) | false (Collection(Edm.String) fields cannot be  
            sortable),  
            "facetatable": true (default where applicable) | false (Edm.GeographyPoint fields cannot be  
            facetatable),  
            "key": true | false (default, only Edm.String fields can be keys),  
            "retrievable": true (default) | false,  
            "analyzer": "name_of_analyzer_for_search_and_indexing", (only if 'searchAnalyzer' and  
            'indexAnalyzer' are not set)  
            "searchAnalyzer": "name_of_search_analyzer", (only if 'indexAnalyzer' is set and 'analyzer' is not  
            set)  
    ]  
}
```

```

    "indexAnalyzer": "name_of_indexing_analyzer", (only if 'searchAnalyzer' is set and 'analyzer' is not
set)
    "synonymMaps": [ "name_of_synonym_map" ] (optional, only one synonym map per field is currently
supported)
  ],
  "suggesters": [
    {
      "name": "name of suggester",
      "searchMode": "analyzingInfixMatching",
      "sourceFields": ["field1", "field2", ...]
    }
  ],
  "scoringProfiles": [
    {
      "name": "name of scoring profile",
      "text": (optional, only applies to searchable fields) {
        "weights": {
          "searchable_field_name": relative_weight_value (positive #'s),
          ...
        }
      },
      "functions": (optional) [
        {
          "type": "magnitude | freshness | distance | tag",
          "boost": # (positive number used as multiplier for raw score != 1),
          "fieldName": "...",
          "interpolation": "constant | linear (default) | quadratic | logarithmic",
          "magnitude": {
            "boostingRangeStart": #,
            "boostingRangeEnd": #,
            "constantBoostBeyondRange": true | false (default)
          },
          "freshness": {
            "boostingDuration": "..." (value representing timespan leading to now over which boosting
occurs)
          },
          "distance": {
            "referencePointParameter": "...", (parameter to be passed in queries to use as reference
location)
            "boostingDistance": # (the distance in kilometers from the reference location where the
boosting range ends)
          },
          "tag": {
            "tagsParameter": "..." (parameter to be passed in queries to specify a list of tags to compare
against target fields)
          }
        }
      ],
      "functionAggregation": (optional, applies only when functions are specified)
      "sum (default) | average | minimum | maximum | firstMatching"
    }
  ],
  "analyzers":(optional)[ ... ],
  "charFilters":(optional)[ ... ],
  "tokenizers":(optional)[ ... ],
  "tokenFilters":(optional)[ ... ],
  "defaultScoringProfile": (optional) "...",
  "corsOptions": (optional) {
    "allowedOrigins": ["*"] | ["origin_1", "origin_2", ...],
    "maxAgeInSeconds": (optional) max_age_in_seconds (non-negative integer)
  },
  "encryptionKey":(optional){
    "keyVaultUri": "azure_key_vault_uri",
    "keyVaultKeyName": "name_of_azure_key_vault_key",
    "keyVaultKeyVersion": "version_of_azure_key_vault_key",
    "accessCredentials":(optional){
      "applicationId": "azure_active_directory_application_id",
      "applicationSecret": "azure active directory application authentication key"
    }
  }
]

```

```
    }  
}  
}
```

## Collection et attributs de champs

Les champs ont un nom, un type qui classe les données stockées et des attributs qui spécifient la façon dont le champ est utilisé.

### Types de données

TYPE	DESCRIPTION
Edm.String	Texte pour lequel un jeton peut éventuellement être généré pour la recherche en texte intégral (césure de mots, recherche de radical, etc).
Collection(Edm.String)	Liste de chaînes pouvant être éventuellement tokenisées pour la recherche en texte intégral. En théorie, il n'existe pas de limite supérieure quant au nombre d'éléments d'une collection, mais la limite supérieure de 16 Mo sur la taille de charge utile s'applique aux collections.
Edm.Boolean	Contient des valeurs true/false.
Edm.Int32	Valeurs entières 32 bits.
Edm.Int64	Valeurs entières 64 bits.
Edm.Double	Données numériques à double précision.
Edm.DateTimeOffset	Valeurs de date et heure représentées au format OData V4 (par exemple, <code>yyyy-MM-ddTHH:mm:ss.ffffZ</code> ou <code>yyyy-MM-ddTHH:mm:ss.fff[+/-]HH:mm</code> ).
Edm.GeographyPoint	Point représentant un emplacement géographique de la planète.

Pour en savoir plus, consultez les [types de données pris en charge](#).

### Attributs

Les attributs d'un champ déterminent son utilisation, par exemple s'il est utilisé dans la recherche en texte intégral, la navigation par facettes, les opérations de tri et ainsi de suite.

Les champs de type chaîne sont souvent marqués avec les attributs « Possibilité de recherche » et « Récupérable ». Vous pouvez utiliser les champs « Tirable », « Filtrable » et « À choix multiple » pour affiner les résultats de la recherche.

ATTRIBUT	DESCRIPTION
----------	-------------

ATTRIBUT	DESCRIPTION
« Possibilité de recherche »	Recherche en texte intégral, avec analyse lexicale (césure de mots) lors de l'indexation. Si vous définissez un champ avec possibilité de recherche sur une valeur comme « journée ensoleillée », cette valeur est fractionnée au niveau interne en jetons individuels « journée » et « ensoleillée ». Pour en savoir plus, consultez la rubrique <a href="#">Fonctionnement de la recherche en texte intégral</a> .
« Filtrable »	Référencé dans les requêtes <code>\$filter</code> . Les champs filtrables de type <code>Edm.String</code> ou <code>Collection(Edm.String)</code> ne font pas l'objet d'une analyse lexicale, les comparaisons ne concernent donc que les correspondances exactes. Par exemple, si vous définissez un champ avec la valeur « journée ensoleillée », la requête <code>\$filter=f eq 'sunny'</code> ne renverra aucune correspondance, contrairement à <code>\$filter=f eq 'sunny day'</code> .
« Triable »	Le système trie les résultats par score par défaut, mais vous pouvez configurer le tri en fonction des champs des documents. Les champs de type <code>Collection(Edm.String)</code> ne sont pas « triables ».
« À choix multiple »	Généralement utilisé dans une présentation des résultats de recherche qui inclut un nombre de correspondances par catégorie (par exemple, les hôtels dans une ville spécifique). Cette option ne peut pas être utilisée avec des champs de type <code>Edm.GeographyPoint</code> . Les champs de type <code>Edm.String</code> qui sont « filtrables », « triables » ou « à choix multiple » ne peuvent pas dépasser 32 Ko de longueur. Pour plus d'informations, consultez l'article <a href="#">Créer un index (API REST)</a> .
« Clé »	Identificateur unique des documents dans l'index. Un seul champ doit être choisi comme champ clé et il doit être de type <code>Edm.String</code> .
« Récupérable »	Définit si le champ peut être retourné dans un résultat de recherche. Cet attribut est utile quand vous voulez utiliser un champ (comme <i>profit margin</i> ) comme mécanisme de filtre, de tri ou de score, mais que vous ne voulez pas qu'il soit visible par l'utilisateur final. Il doit être <code>true</code> for <code>key</code> .

Même si vous pouvez ajouter de nouveaux champs à tout moment, les définitions de champ existantes sont verrouillées pour toute la durée de vie de l'index. C'est pourquoi les développeurs utilisent généralement le portail pour créer des index simples, tester des idées ou rechercher une définition de paramètre. Il est plus efficace d'effectuer des itérations fréquentes sur la conception d'un index si vous suivez une approche basée sur du code pour reconstruire l'index facilement.

#### NOTE

Les API que vous utilisez pour créer un index ont différents comportements par défaut. Pour les [API REST](#), la plupart des attributs sont activés par défaut (par exemple, « Possibilité de recherche » et « Récupérable » sont actifs pour les champs de type chaîne) et vous n'avez généralement besoin de les définir que si vous voulez les désactiver. Pour le Kit de développement logiciel (SDK) .NET, l'inverse est vrai. Pour les propriétés que vous ne définissez pas explicitement, l'option par défaut désactive le comportement de recherche correspondante, sauf si vous l'activez de façon spécifique.

## analyzers

L'élément analyseurs définit le nom de l'analyseur linguistique à utiliser pour le champ. Pour plus d'informations sur les différents analyseurs disponibles, consultez [Ajout d'analyseurs à un index Recherche cognitive Azure](#). Les analyseurs peuvent être utilisés uniquement avec les champs pouvant faire l'objet d'une recherche. Une fois que l'analyseur est affecté à un champ, il ne peut plus être modifié, à moins de régénérer l'index.

## suggesters

Un suggesteur est une section du schéma qui définit quels champs d'un index sont utilisés pour prendre en charge l'autocomplétion et les requêtes prédictives dans les recherches. En général, les chaînes de recherche partielle sont envoyées à l'[API REST Suggestions](#) pendant que l'utilisateur tape une requête de recherche, et l'API retourne un ensemble de documents ou d'expressions suggérés.

Les champs ajoutés à un suggesteur sont utilisés pour générer des termes de recherche à saisie semi-automatique (« type-ahead »). Tous les termes de recherche sont créés pendant l'indexation et stockés séparément. Pour plus d'informations sur la création d'une structure de suggesteur, consultez [Ajouter des suggesteurs](#).

## corsOptions

Le code JavaScript côté client ne peut pas appeler d'API par défaut, car le navigateur empêche toutes les requêtes cross-origin. Pour autoriser les requêtes cross-origin dans l'index, activez CORS (partage des ressources cross-origin) en définissant l'attribut **corsOptions**. Pour des raisons de sécurité, seules les API de requête prennent en charge CORS.

Les options suivantes peuvent être définies pour CORS :

- **allowedOrigins** (obligatoire) : il s'agit de la liste des origines pouvant accéder à l'index. Cela signifie que le code JavaScript distribué à partir de ces origines est autorisé à interroger l'index (s'il fournit la bonne clé API). Chaque origine se présente généralement sous la forme `protocol://<fully-qualified-domain-name>:<port>`, bien que `<port>` soit souvent omis. Pour plus d'informations, voir [Partage des ressources cross-origin \(Wikipédia\)](#).

Si vous voulez autoriser l'accès à toutes les origines, incluez uniquement l'élément `*` dans le tableau **allowedOrigins**. Si *cette pratique est déconseillée pour les services de recherche de production*, elle est souvent utile pour le développement et le débogage.

- **maxAgeInSeconds** (facultatif) : les navigateurs utilisent cette valeur pour déterminer la durée (en secondes) de mise en cache des réponses CORS préliminaires. Il doit s'agir d'un entier non négatif. Plus cette valeur est importante, meilleures sont les performances, mais plus il faut de temps pour que les modifications apportées à la stratégie CORS prennent effet. Si la valeur n'est pas définie, une durée par défaut de 5 minutes est utilisée.

## scoringProfiles

Un **profil de score** est une section du schéma qui définit des comportements de score personnalisés permettant de faire apparaître certains éléments plus haut dans les résultats de la recherche. Les profils de calcul de score sont constitués de pondérations et de fonctions de champ. Pour les utiliser, vous spécifiez un profil par nom dans la chaîne de requête.

Un profil de score par défaut fonctionne en arrière-plan pour calculer un score de recherche pour chaque élément d'un jeu de résultats. Vous pouvez utiliser le profil de score interne et sans nom. Vous pouvez aussi

définir `defaultScoringProfile` pour utiliser par défaut un profil personnalisé, appelé chaque fois qu'aucun profil personnalisé n'est spécifié dans la chaîne de requête.

## Attributs et taille de l'index (implications en matière de stockage)

La taille d'un index est déterminée par la taille des documents que vous chargez, plus la configuration de l'index, par exemple si vous incluez des suggesteurs, et la façon dont vous définissez des attributs sur des champs individuels.

La capture d'écran suivante illustre les caractéristiques du stockage d'index résultant des différentes combinaisons d'attributs. L'index est basé sur l'**exemple d'index de biens immobiliers**, que vous pouvez créer facilement à l'aide de l'Assistant Importer des données. Bien que les schémas de l'index ne soient pas montrés, vous pouvez en déduire les attributs d'après le nom de l'index. Par exemple, pour l'index `realestate-searchable`, seul l'attribut « Possibilité de recherche » est sélectionné ; pour l'index `realestate-retrievable`, seul l'index « Récupérable » est sélectionné, et ainsi de suite.

NAME	DOCUMENT COUNT	STORAGE SIZE
realestate-all-attributes-no-suggester	4,959	26.55 MiB
realestate-all-attributes-plus-suggester	4,959	49.4 MiB
realestate-filterable-facetable-sortable	4,959	20.89 MiB
realestate-no-attributes	4,959	4.99 MiB
realestate-retrievable	4,959	5.04 MiB
realestate-searchable	4,959	9.95 MiB

Bien que ces variantes d'index soient artificielles, nous pouvons nous y reporter pour nous faire une idée de la façon dont les attributs affectent le stockage. Le paramètre « Récupérable » fait-il croître l'index ? Non. L'ajout de champs à un **suggesteur** fait-il croître l'index ? Oui.

Les index qui prennent en charge le filtrage et le tri sont proportionnellement plus volumineux que ceux qui prennent en charge uniquement la recherche en texte intégral. Cela est dû au fait que les opérations de filtrage et de tri recherchent des correspondances exactes, ce qui nécessite la présence de chaînes textuelles verbatim. En revanche, les champs pouvant faire l'objet d'une recherche prenant en charge les requêtes en texte intégral utilisent des index inversés, qui sont remplis avec des termes assortis de jetons qui occupent moins d'espace que des documents entiers.

### NOTE

L'architecture de stockage est considérée comme un détail d'implémentation de Recherche cognitive Azure et est susceptible d'évoluer sans préavis. Il n'est pas garanti que le comportement actuel persistera dans l'avenir.

## Étapes suivantes

Maintenant que vous comprenez la composition de l'index, vous pouvez créer votre premier index sur le portail. Nous vous recommandons de commencer par l'**Assistant Importer des données**, en choisissant les sources de données hébergées `realestate-us-sample` ou `hotels-sample`.

### [Assistant Importer des données \(portail\)](#)

Pour les deux jeux de données, l'Assistant peut déduire un schéma d'index, importer les données et générer un

index pouvant faire l'objet d'une recherche que vous pouvez interroger à l'aide de l'explorateur de recherche. Recherchez ces sources de données dans la page **Connexion à vos données** de l'Assistant Importer des données.

Home > mydemo >

## Import data

Connect to your data   Add cognitive skills (Optional)   Customize target index   Create an indexer

Create and load a search index using data from an existing Azure data source in your current subscription. Azure Cognitive Search crawls the data structure you provide, extracts searchable content, optionally enriches it with cognitive skills, and loads it into an index. [Learn more](#)

Data Source	Samples
Type	Name
	realestate-us-sample
	hotels-sample

# Comment créer un index dans plusieurs langues dans Recherche cognitive Azure

04/10/2020 • 5 minutes to read • [Edit Online](#)

Les index peuvent comprendre des champs avec du contenu provenant de plusieurs langues, par exemple pour créer des champs pour des chaînes spécifiques à une langue. Pour obtenir des résultats optimaux lors de l'indexation et de l'interrogation, affectez un analyseur de langue qui fournit les règles linguistiques appropriées.

Recherche cognitive Azure offre une large sélection d'analyseurs de langue Lucene et Microsoft, qui peuvent être affectés à des champs individuels à l'aide de la propriété Analyzer. Vous pouvez également spécifier un analyseur de langue dans le portail, comme décrit dans cet article.

## Ajouter des analyseurs aux champs

L'analyseur de langue est spécifié lors de la création d'un champ. L'ajout d'un analyseur à une définition de champ existante nécessite le remplacement (et le rechargement) de l'index ou la création d'un nouveau champ identique à l'original, mais avec une affectation d'analyseur. Vous pouvez ensuite supprimer le champ inutilisé à votre convenance.

1. Connectez-vous au [portail Azure](#), puis trouvez votre service de recherche.
2. Cliquez sur **Ajouter un index** dans la barre de commandes en haut du tableau de bord de service pour démarrer un nouvel index ou ouvrez un index existant pour définir un analyseur sur des nouveaux champs que vous ajoutez à un index existant.
3. Démarrez une définition de champ en fournissant un nom.
4. Choisissez le type de données Edm.String. Seuls les champs de type chaîne peuvent faire l'objet d'une recherche en texte intégral.
5. Définissez l'attribut **Searchable** pour activer la propriété de l'analyseur. Le champ doit être basé sur du texte pour pouvoir utiliser un analyseur de langue.
6. Choisissez l'un des analyseurs disponibles.

Add index

\* Index name [?](#)  
test-index-2

\* Key [?](#)  
id

Suggester name [?](#) 1

Search mode [?](#)

**+ Add field** [+ Add subfield](#) [Delete](#)

FIELD NAME	TYPE	RETRIEVABLE	FILTERABLE	SORTABLE	FACTETABLE	SEARCHABLE	ANALYZER	SUGGESTER
id	Edm.String	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Spanish - Microsoft	<a href="#">...</a>
SpanishContent	Edm.String	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	French - Lucene	<a href="#">...</a>
FrenchContent	Edm.String	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Polish - Microsoft	<a href="#">...</a>
PolishContent	Edm.String	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>		<a href="#">...</a>

**Create**

1. Suggester name: id  
2. Searchable: checked  
3. Analyzer: Spanish - Microsoft, French - Lucene, Polish - Microsoft

Par défaut, tous les champs pouvant faire l'objet d'une recherche utilisent l'[analyseur Lucene Standard](#) qui est

indépendant de la langue. Pour afficher la liste complète des analyseurs pris en charge, consultez [Ajouter des analyseurs linguistiques à un index de Recherche cognitive Azure](#).

Dans le portail, les analyseurs sont conçus pour être utilisés tels quels. Si vous avez besoin d'une personnalisation ou d'une configuration spécifique des filtres et des générateurs de jetons, vous devez [créer un analyseur personnalisé](#) dans le code. Le portail ne prend pas en charge la sélection ou la configuration d'analyseurs personnalisés.

## Champs spécifiques à la langue de la requête

Une fois que l'analyseur de langage est sélectionné pour un champ, il sera utilisé à chaque demande de recherche et d'indexation pour ce champ. Lorsqu'une requête est émise sur plusieurs champs à l'aide de différents analyseurs, elle sera traitée indépendamment par les analyseurs affectés pour chaque champ.

Si la langue de l'agent d'émission d'une requête est connue, une demande de recherche peut être étendue à un champ spécifique à l'aide du paramètre de requête **searchFields**. La requête suivante sera émise uniquement sur la description en polonais :

```
https://[service name].search.windows.net/indexes/[index name]/docs?  
search=darmowy&searchFields=PolishContent&api-version=2020-06-30
```

Vous pouvez interroger votre index à partir du portail avec l'[Explorateur de recherche](#) pour coller une requête similaire à celle présentée ci-dessus.

## Améliorer les champs spécifiques à la langue

Parfois, la langue de l'agent d'émission d'une requête n'est pas connue, auquel cas la requête peut être exécutée simultanément sur tous les champs. Si nécessaire, la préférence de résultats dans une langue donnée peut être définie à l'aide des [profils de score](#). Dans l'exemple ci-dessous, les correspondances trouvées dans la description en anglais auront un score supérieur par rapport aux correspondances en polonais et en français :

```
"scoringProfiles": [  
  {  
    "name": "englishFirst",  
    "text": {  
      "weights": { "description_en": 2 }  
    }  
  }  
]
```

```
https://[service name].search.windows.net/indexes/[index name]/docs?  
search=Microsoft&scoringProfile=englishFirst&api-version=2020-06-30
```

## Étapes suivantes

Si vous êtes développeur .NET, notez que vous pouvez configurer les analyseurs de langue à l'aide du [Kit de développement logiciel \(SDK\) Recherche cognitive Azure .NET](#) et de la propriété [Analyzer](#).

# Analyseurs pour le traitement de texte dans la Recherche cognitive Azure

04/10/2020 • 21 minutes to read • [Edit Online](#)

Un *analyseur* est un composant du [moteur de recherche en texte intégral](#) chargé de traiter le texte dans les chaînes de requête et les documents indexés. Le traitement du texte (également appelé analyse lexicale) est à l'origine de transformations. Il modifie une chaîne via des actions telles que celles-ci :

- Supprimer les mots (mots vides) et la ponctuation non essentiels
- Segmenter des expressions et des mots avec tirets en différents composants
- Mettre les mots en majuscules en minuscules
- Réduire les mots dans des formes racines primitives pour une efficacité de stockage et de sorte que des correspondances puissent être trouvées, quels que soient les temps

L'analyse s'applique à des champs `Edm.String` marqués comme étant « interrogeables », ce qui indique une recherche en texte intégral. Pour les champs avec cette configuration, l'analyse a lieu pendant l'indexation lors de la création de jetons, puis à nouveau lors de l'exécution de requêtes lorsque ces dernières sont analysées et que le moteur recherche les jetons correspondants. Une correspondance est plus probable lorsque le même analyseur est utilisé à la fois pour l'indexation et pour les requêtes, mais vous pouvez définir l'analyseur de manière indépendante pour chaque charge de travail, en fonction de vos besoins.

Les types de requêtes qui ne sont pas une recherche en texte intégral, tels que l'expression régulière ou la recherche approximative, ne passent pas par la phase d'analyse du côté de la requête. Au lieu de cela, l'analyseur envoie ces chaînes directement au moteur de recherche, à l'aide du modèle que vous fournissez comme base pour la correspondance. Ces formulaires de requête nécessitent généralement des jetons de chaîne entière pour effectuer un travail de correspondance de modèle. Pour recevoir des jetons de termes complets pendant l'indexation, vous aurez peut-être besoin d'[analyseurs personnalisés](#). Pour plus d'informations sur le moment et la raison de l'analyse des termes de la requête, consultez [Recherche en texte intégral dans la Recherche cognitive Azure](#).

Pour plus d'informations sur l'analyse lexicale, regardez le clip vidéo suivant, qui donne une brève explication.

## Analyseur par défaut

Dans les requêtes de la Recherche cognitive Azure, un analyseur est automatiquement appelé sur tous les champs de chaînes marqués comme pouvant faire l'objet d'une recherche.

Par défaut, la Recherche cognitive Azure utilise l'[analyseur Apache Lucene Standard \(lucene standard\)](#), qui décompose le texte en éléments en suivant les règles de la « [Segmentation du texte Unicode](#) ». Par ailleurs, l'analyseur standard convertit tous les caractères en minuscules. Les documents indexés et les termes de recherche sont analysés pendant l'indexation et le traitement des requêtes.

Vous pouvez substituer l'analyseur par défaut champ par champ. Ces analyseurs alternatifs peuvent être un [analyseur linguistique](#) pour le traitement linguistique, un [analyseur personnalisé](#) ou un analyseur prédéfini figurant dans la [liste des analyseurs disponibles](#).

## Types d'analyseurs

La liste suivante décrit les analyseurs disponibles dans la Recherche cognitive Azure.

CATEGORY	DESCRIPTION
Analyseur Lucene standard	Par défaut. Aucune spécification ou configuration n'est nécessaire. Cet analyseur à usage général est efficace pour de nombreux scénarios et langues.
Analyseurs prédéfinis	<p>Proposés comme produits finis destinés à être utilisés tels quels.</p> <p>Il en existe deux types : spécialisé et linguistique. Ils sont dits « prédéfinis », car vous les référez par leur nom, sans aucune configuration ni personnalisation.</p> <p>Les <a href="#">analyseurs spécialisés (non dépendants de la langue)</a> sont employés quand les entrées de texte nécessitent un traitement spécialisé ou minimal. Les analyseurs prédéfinis qui ne dépendent pas de la langue sont les suivants : <b>Asciifolding, Keyword, Pattern, Simple, Stop, Whitespace</b>.</p> <p>Utilisez les <a href="#">analyseurs linguistiques</a> quand vous avez besoin d'une prise en charge linguistique avancée pour différentes langues. La Recherche cognitive Azure prend en charge 35 analyseurs linguistiques Lucene et 50 analyseurs de traitement en langage naturel Microsoft.</p>
Analyseurs personnalisés	Ils renvoient à une configuration définie par l'utilisateur constituée d'une combinaison d'éléments existants, dont un générateur de jetons (obligatoire) et des filtres facultatifs (caractères ou jetons).

Certains analyseurs prédéfinis, comme **Pattern** ou **Stop**, prennent en charge un ensemble limité d'options de configuration. Pour définir ces options, vous créez en réalité un analyseur personnalisé constitué de l'analyseur prédéfini et d'une des autres options documentées dans les [informations de référence sur les analyseurs prédéfinis](#). Comme pour toute configuration personnalisée, nommez votre nouvelle configuration (par exemple, *MonAnalyseurSéquencesOctets*) pour la distinguer de l'analyseur Lucene Pattern.

## Comment spécifier des analyseurs

La définition d'un analyseur est facultative. En règle générale, essayez d'abord d'utiliser l'analyseur Lucene standard par défaut pour voir comment il fonctionne. Si les requêtes ne renvoient pas les résultats attendus, le passage à un autre analyseur est souvent la bonne solution.

1. Lors de la création d'une définition de champ dans l'[index](#), définissez la propriété **analyzer** sur l'un des éléments suivants : un [analyseur prédéfini](#) tel que `keyword`, un [analyseur de langage](#) tel que `en.microsoft` ou un analyseur personnalisé (défini dans le même schéma d'index).

```

"fields": [
{
  "name": "Description",
  "type": "Edm.String",
  "retrievable": true,
  "searchable": true,
  "analyzer": "en.microsoft",
  "indexAnalyzer": null,
  "searchAnalyzer": null
},

```

Si vous utilisez un [analyseur de langage](#), vous devez utiliser la propriété **analyzer** pour le spécifier. Les propriétés **searchAnalyzer** et **indexAnalyzer** ne prennent pas en charge les analyseurs de langage.

- Vous pouvez également définir `indexAnalyzer` et `searchAnalyzer` afin de changer d'analyseur pour chaque charge de travail. Ces propriétés sont définies ensemble et remplacent la propriété `analyzer`, qui doit avoir la valeur Null. Vous pouvez utiliser différents analyseurs pour la préparation et la récupération des données si l'une de ces activités nécessite une transformation spécifique et l'autre non.

```
"fields": [
{
  "name": "Description",
  "type": "Edm.String",
  "retrievable": true,
  "searchable": true,
  "analyzer": null,
  "indexAnalyzer": "keyword",
  "searchAnalyzer": "whitespace"
},
```

- Pour les analyseurs personnalisés uniquement, créez une entrée dans la section [analyseurs] de l'index, puis affectez votre analyseur personnalisé à la définition de champ pour l'une ou l'autre des deux étapes précédentes. Pour plus d'informations, consultez [Créer un index](#) et [Ajouter des analyseurs personnalisés](#).

## Quand ajouter des analyseurs

La phase de développement actif est le meilleur moment pour ajouter et affecter des analyseurs, là où la suppression et la récréation d'index sont des activités courantes.

Étant donné que les analyseurs sont utilisés pour créer des jetons pour les termes, vous devez affecter un analyseur lors de la création du champ. En fait, l'affectation d'`analyzer` ou d' `indexAnalyzer` à un champ déjà créé physiquement n'est pas autorisée (bien que vous puissiez modifier la propriété `searchAnalyzer` à tout moment sans aucun impact sur l'index).

Pour modifier l'analyseur d'un champ existant, vous devez [recréer entièrement l'index](#) (vous ne pouvez pas recréer des champs individuels). Pour les index en production, vous devez retarder une régénération en créant un champ avec la nouvelle affectation d'analyseur et commencer à l'utiliser à la place de l'ancien. Utilisez [Mettre à jour l'index](#) pour incorporer le nouveau champ et [mergeOrUpload](#) pour le remplir. Par la suite, pendant l'opération de maintenance planifiée de l'index, vous pouvez le nettoyer de façon à supprimer les champs obsolètes.

Pour ajouter un nouveau champ à un index existant,appelez [Mettre à jour l'index](#) et [mergeOrUpload](#) pour le remplir.

Pour ajouter un analyseur personnalisé à un index existant, transférez l'indicateur `allowIndexDowntime` dans [Mettre à jour l'index](#) si vous souhaitez éviter cette erreur :

*« Mise à jour d'index non autorisée, car cette opération pourrait entraîner un temps d'arrêt. Pour ajouter de nouveaux analyseurs, générateurs de jetons, filtres de jetons ou filtres de caractères à un index existant, définissez le paramètre de requête « allowIndexDowntime » sur « true » dans la demande de mise à jour d'index. Notez que cette opération placera votre index hors connexion pendant au moins quelques secondes, ce qui entraînera l'échec de vos demandes d'indexation et de requête. Les performances et la disponibilité en écriture peuvent être altérées pendant plusieurs minutes après la mise à jour de l'index, voire plus longtemps pour les index de très grande taille ».*

## Suggestions pour utiliser des analyseurs

Cette section offre des conseils pour utiliser les analyseurs.

### Un même analyseur pour les opérations de lecture-écriture, sauf besoins spécifiques

La Recherche cognitive Azure vous permet de spécifier différents analyseurs pour l'indexation et la recherche par le biais des propriétés de champ supplémentaires `indexAnalyzer` et `searchAnalyzer`. Par défaut, l'analyseur défini avec la propriété `analyzer` est utilisé pour l'indexation et la recherche. Si `analyzer` n'est pas spécifié, l'analyseur Lucene standard est utilisé par défaut.

En règle générale, il est préférable d'utiliser le même analyseur pour l'indexation et l'interrogation, sauf si des besoins spécifiques vous obligent à faire autrement. Veillez à effectuer des tests approfondis. Quand il existe une divergence de traitement de texte pendant la recherche et l'indexation, le risque est que les termes de la requête et les termes indexés ne correspondent pas si la configuration de l'analyseur de recherche et celle de l'analyseur d'indexation ne sont pas conformes.

### Effectuer des tests pendant le développement actif

La substitution de l'analyseur standard nécessite une régénération de l'index. Si possible, choisissez les analyseurs à utiliser pendant le développement actif, avant de déployer l'index dans un environnement de production.

### Examiner les termes sous forme de jetons

Si une recherche ne renvoie pas les résultats attendus, cela est très probablement dû aux différences de jetons entre les termes entrés dans la requête et les termes sous forme de jetons présents dans l'index. Si les jetons ne sont pas identiques, les correspondances ne sont pas détectées. Pour examiner les résultats du générateur de jetons, nous vous recommandons d'utiliser [l'API d'analyse](#) comme outil d'investigation. La réponse se compose de jetons qui sont générés par un analyseur spécifique.

## Exemples REST

Les exemples ci-dessous montrent des définitions d'analyseur pour quelques scénarios clés.

- [Exemple d'analyseur personnalisé](#)
- [Exemple d'affectation d'analyseurs à un champ](#)
- [Combinaison d'analyseurs pour l'indexation et la recherche](#)
- [Exemple d'analyseur linguistique](#)

### Exemple d'analyseur personnalisé

Cet exemple montre une définition d'analyseur avec des options personnalisées. Les options personnalisées pour les filtres de caractères, les générateurs de jetons et les filtres de jetons sont spécifiées séparément comme des constructions nommées, puis elles sont référencées dans la définition de l'analyseur. Les éléments prédéfinis sont utilisés tels quels et sont référencés par leur nom.

Si nous suivons cet exemple :

- Les analyseurs sont une propriété de la classe d'un champ pouvant faire l'objet d'une recherche.
- Un analyseur personnalisé fait partie d'une définition d'index. Il peut faire l'objet d'une légère personnalisation (par exemple, la personnalisation d'une option d'un filtre) ou d'une personnalisation à plusieurs endroits.
- Dans ce cas, l'analyseur personnalisé est « `my_analyzer` », qui utilise alors le générateur de jetons standard personnalisé « `my_standard_tokenizer` », et deux filtres de jetons : le filtre « `lowercase` » et le filtre `asciifolding` personnalisé « `my_asciifolding` ».
- Il définit également les deux filtres de caractères personnalisés « `map_dash` » et « `remove_whitespace` ». Le premier remplace tous les tirets par des traits de soulignement et le second supprime tous les espaces. Les espaces doivent être encodés en UTF-8 dans les règles de mappage. Les filtres de caractères sont appliqués avant la segmentation du texte en unités lexicales et affectent les jetons qui en résultent (les tirets et les espaces provoquent l'arrêt du générateur de jetons, mais pas les traits de soulignement).

```
{
  "name": "myindex",
  "fields": [
    {
      "name": "id",
      "type": "Edm.String",
      "key": true,
      "searchable": false
    },
    {
      "name": "text",
      "type": "Edm.String",
      "searchable": true,
      "analyzer": "my_analyzer"
    }
  ],
  "analyzers": [
    {
      "name": "my_analyzer",
      "@odata.type": "#Microsoft.Azure.Search.CustomAnalyzer",
      "charFilters": [
        "map_dash",
        "remove_whitespace"
      ],
      "tokenizer": "my_standard_tokenizer",
      "tokenFilters": [
        "my_asciifolding",
        "lowercase"
      ]
    }
  ],
  "charFilters": [
    {
      "name": "map_dash",
      "@odata.type": "#Microsoft.Azure.Search.MappingCharFilter",
      "mappings": ["-=>_"]
    },
    {
      "name": "remove_whitespace",
      "@odata.type": "#Microsoft.Azure.Search.MappingCharFilter",
      "mappings": ["\u0020=>"]
    }
  ],
  "tokenizers": [
    {
      "name": "my_standard_tokenizer",
      "@odata.type": "#Microsoft.Azure.Search.StandardTokenizerV2",
      "maxTokenLength": 20
    }
  ],
  "tokenFilters": [
    {
      "name": "my_asciifolding",
      "@odata.type": "#Microsoft.Azure.Search.AsciiFoldingTokenFilter",
      "preserveOriginal": true
    }
  ]
}
```

## Exemple d'affectation d'analyseur par champ

L'analyseur standard est celui qui est utilisé par défaut. Supposons que vous souhaitez remplacer l'analyseur par défaut par un autre analyseur prédéfini, tel que l'analyseur de pattern. Si vous ne définissez pas d'options personnalisées, vous devez uniquement le spécifier par son nom dans la définition du champ.

L'élément « analyzer » remplace l'analyseur standard champ après champ. Aucun remplacement global n'est

possible. Dans cet exemple, `text1` utilise l'analyseur de pattern, et `text2`, qui ne spécifie pas d'analyseur, utilise celui par défaut.

```
{  
    "name": "myindex",  
    "fields": [  
        {  
            "name": "id",  
            "type": "Edm.String",  
            "key": true,  
            "searchable": false  
        },  
        {  
            "name": "text1",  
            "type": "Edm.String",  
            "searchable": true,  
            "analyzer": "pattern"  
        },  
        {  
            "name": "text2",  
            "type": "Edm.String",  
            "searchable": true  
        }  
    ]  
}
```

### Combinaison d'analyseurs pour les opérations d'indexation et de recherche

Les API comprennent des attributs d'index supplémentaires qui permettent de spécifier des analyseurs différents pour l'indexation et la recherche. Les attributs `searchAnalyzer` et `indexAnalyzer` doivent être spécifiés sous forme de paire, en remplacement de l'attribut `analyzer`.

```
{  
    "name": "myindex",  
    "fields": [  
        {  
            "name": "id",  
            "type": "Edm.String",  
            "key": true,  
            "searchable": false  
        },  
        {  
            "name": "text",  
            "type": "Edm.String",  
            "searchable": true,  
            "indexAnalyzer": "whitespace",  
            "searchAnalyzer": "simple"  
        },  
    ],  
}
```

### Exemple d'analyseur linguistique

Les champs qui contiennent des chaînes dans différentes langues peuvent utiliser un analyseur linguistique, tandis que les autres champs conservent l'analyseur par défaut (ou utilisent un autre analyseur prédéfini ou personnalisé). Si vous utilisez un analyseur linguistique, vous devez l'utiliser à la fois pour les opérations d'indexation et de recherche. Les champs qui utilisent un analyseur linguistique ne peuvent pas avoir des analyseurs différents pour l'indexation et la recherche.

```
{
  "name": "myindex",
  "fields": [
    {
      "name": "id",
      "type": "Edm.String",
      "key": true,
      "searchable": false
    },
    {
      "name": "text",
      "type": "Edm.String",
      "searchable": true,
      "indexAnalyzer": "whitespace",
      "searchAnalyzer": "simple"
    },
    {
      "name": "text_fr",
      "type": "Edm.String",
      "searchable": true,
      "analyzer": "fr.lucene"
    }
  ],
}
```

## Exemples C#

Si vous utilisez les exemples de code du SDK .NET, vous pouvez ajouter ces exemples pour utiliser ou configurer des analyseurs.

- [Attribuer un analyseur intégré](#)
- [Configurer un analyseur](#)

### Attribuer un analyseur de langage

Tout analyseur qui est utilisé tel quel, sans configuration, est spécifié sur une définition de champ. La création d'une entrée dans la section **[analyzers]** de l'index n'est pas obligatoire.

Cet exemple attribue les analyseurs Anglais et Français de Microsoft aux champs de description. C'est un extrait de code issu d'une définition plus grande de l'index des hôtels, créé à l'aide de la classe Hotel dans le fichier hotels.cs de l'exemple [DotNetHowTo](#).

Appelez la classe [Analyzer](#) en spécifiant le type [AnalyzerName](#) pour fournir un analyseur de texte pris en charge dans la Recherche cognitive Azure.

```
public partial class Hotel
{
  ...
  [IsSearchable]
  [Analyzer(AnalyzerName.AsString.EnMicrosoft)]
  [JsonProperty("description")]
  public string Description { get; set; }

  [IsSearchable]
  [Analyzer(AnalyzerName.AsString.FrLucene)]
  [JsonProperty("description_fr")]
  public string DescriptionFr { get; set; }

  ...
}
```

## Définir un analyseur personnalisé

Lorsqu'une personnalisation ou configuration est requise, vous devez ajouter une construction de l'analyseur à un index. Une fois que vous la définissez, vous pouvez l'ajouter à la définition de champ comme illustré dans l'exemple précédent.

Créez un objet [CustomAnalyzer](#). Pour plus d'exemples, consultez [CustomAnalyzerTests.cs](#).

```
{  
    var definition = new Index()  
    {  
        Name = "hotels",  
        Fields = FieldBuilder.BuildForType<Hotel>(),  
        Analyzers = new[]  
        {  
            new CustomAnalyzer()  
            {  
                Name = "url-analyze",  
                Tokenizer = TokenizerName.UaxUrlEmail,  
                TokenFilters = new[] { TokenFilterName.Lowercase }  
            }  
        },  
    };  
  
    serviceClient.Indexes.Create(definition);
```

## Étapes suivantes

- Lisez notre explication complète du [fonctionnement de la recherche en texte intégral dans la Recherche cognitive Azure](#). Cet article utilise des exemples pour expliquer les comportements qui, au premier abord, peuvent sembler contraires à la logique.
- Essayez une syntaxe de requête supplémentaire à partir de la section Exemple [Rechercher des documents](#) ou à partir de la [syntaxe de requête simple](#) dans l'Explorateur de recherche du portail.
- Découvrez comment appliquer des [analyseurs lexicaux propres au langage](#).
- [Configurez des analyseurs personnalisés](#) pour un traitement minimal ou pour un traitement spécialisé sur certains champs.

## Voir aussi

[API REST de recherche de documents](#)

[Syntaxe de requête simple](#)

[Syntaxe de requête complète Lucene](#)

[Traiter les résultats de recherche](#)

# Ajouter des analyseurs de langue à des champs de chaîne dans l'index de Recherche cognitive Azure

04/10/2020 • 10 minutes to read • [Edit Online](#)

Un *analyseur linguistique* est un type spécifique d'[analyseur de texte](#) qui effectue une analyse lexicale selon les règles linguistiques de la langue cible. Chaque champ pouvant faire l'objet d'une recherche dispose d'une propriété **analyzer**. Si votre contenu est constitué de chaînes traduites, par exemple des champs distincts pour le texte en anglais et le texte en chinois, vous pouvez spécifier des analyseurs linguistiques sur chaque champ pour accéder aux riches fonctionnalités linguistiques de ces analyseurs.

## Quand utiliser un analyseur linguistique ?

Vous devez envisager le recours à un analyseur linguistique quand la reconnaissance de la structure des mots ou des phrases ajoute de la valeur à l'analyse du texte. Un exemple courant est l'association de formes de verbes irréguliers (« bring » et « brought ») ou de noms au pluriel (« mice » et « mouse »). Sans reconnaissance linguistique, ces chaînes sont analysées uniquement sur des caractéristiques physiques, et l'association n'est pas détectée. Étant donné que les grands segments de texte sont plus susceptibles de présenter ce contenu, les champs composés de descriptions, d'évaluations ou de résumés sont de bons candidats pour un analyseur linguistique.

Vous devez également prendre en compte le recours aux analyseurs linguistiques quand le contenu est constitué de chaînes en langue non occidentale. Bien que l'[analyseur par défaut](#) soit indépendant de la langue, le concept d'utilisation d'espaces et de caractères spéciaux (traits d'union et barres obliques) pour séparer les chaînes tend à s'appliquer davantage aux langues occidentales qu'aux autres.

Par exemple, en chinois, japonais, coréen (CJK) et d'autres langues asiatiques, un espace n'est pas nécessairement un délimiteur de mots. Prenons la chaîne suivante en japonais. Étant donné qu'il n'y a pas d'espace, un analyseur indépendant de la langue analysera probablement la chaîne entière comme un seul jeton, alors qu'il s'agit en fait d'une phrase.

これは私たちの銀河系の中ではもっとも重く明るいクラスの球状星団です。

(This is the heaviest and brightest group of spherical stars in our galaxy.)

Dans l'exemple ci-dessus, une requête correcte devrait inclure le jeton complet, ou un jeton partiel avec caractère générique de suffixe, ce qui donnerait lieu à une expérience de recherche non naturelle et restrictive.

Une meilleure expérience consiste à rechercher des mots individuels : 明るい (brillant), 私たちの (notre), 銀河系 (galaxie). L'utilisation de l'un des analyseurs japonais disponibles dans Recherche cognitive est plus susceptible de déverrouiller ce comportement, car ces analyseurs sont mieux équipés pour diviser le segment de texte en mots significatifs dans la langue cible.

## Comparaison des analyseurs Lucene et Microsoft

Recherche cognitive Azure prend en charge 35 analyseurs linguistiques qui s'appuient sur Lucene et 50 analyseurs linguistiques exploitant la technologie propriétaire Microsoft de traitement du langage naturel utilisée dans Office et Bing.

Certains développeurs peuvent préférer la solution open source, plus familière et simple de Lucene. Les analyseurs linguistiques Lucene sont plus rapides, mais les analyseurs Microsoft proposent des fonctionnalités avancées, comme la lemmatisation, la décomposition en mots (dans les langues comme

l'allemand, le danois, le néerlandais, le suédois, le norvégien, l'estonien, le finnois, le hongrois et le slovaque) et la reconnaissance d'entités (URL, adresses e-mail, dates, numéros). Si possible, vous devez comparer les analyseurs Microsoft et Lucene pour savoir lequel vous convient le mieux.

L'indexation avec les analyseurs Microsoft est en moyenne trois fois plus lente qu'avec leurs équivalents Lucene, en fonction de la langue. Les performances de recherche ne doivent pas être trop affectées pour les requêtes de taille moyenne.

### Analyseurs pour l'anglais

L'analyseur par défaut est Lucene Standard, qui fonctionne bien pour l'anglais, mais peut-être pas aussi bien que l'analyseur d'anglais de Lucene ou l'analyseur d'anglais de Microsoft.

- L'analyseur d'anglais de Lucene est une extension de l'analyseur standard. Il supprime la marque du possessif (« 's ») à la fin des mots, applique l'algorithme de recherche de radical de Porter et supprime les mots vides de l'anglais.
- L'analyseur d'anglais de Microsoft procède par lemmatisation plutôt que par recherche de radical. Il gère donc beaucoup mieux les formes fléchies et irrégulières, ce qui donne des résultats de recherche plus pertinents

## Configuration des analyseurs

Les analyseurs linguistiques sont utilisés en l'état. Pour chaque champ de la définition d'index, vous pouvez attribuer à la propriété **analyzer** un nom d'analyseur qui spécifie la langue et la pile linguistique (Microsoft ou Lucene). Le même analyseur sera appliqué lors de l'indexation et de la recherche pour ce champ. Il peut par exemple y avoir des champs distincts pour des descriptions d'hôtels en anglais, en français et en espagnol côté à côté dans le même index.

#### NOTE

Il n'est pas possible d'utiliser un autre analyseur de langage au moment de l'indexation qu'au moment de la requête pour un champ. Cette fonctionnalité est réservée aux [analyseurs personnalisés](#). C'est la raison pour laquelle, si vous tentez de définir les propriétés **searchAnalyzer** ou **indexAnalyzer** sur le nom d'un analyseur de langage, l'API REST renvoie une réponse d'erreur. Vous devez utiliser la propriété **analyzer** à la place.

Utilisez le paramètre de requête **searchFields** pour spécifier le champ de langue dans vos requêtes. Vous pouvez consulter les exemples de requêtes qui contiennent la propriété **analyzer** dans [Recherche dans les documents](#).

Pour plus d'informations sur les propriétés d'index, voir [Créer un index \(API REST Recherche cognitive Azure\)](#).

Pour plus d'informations sur l'analyse dans Recherche cognitive Azure, voir [Analyseurs dans Recherche cognitive Azure](#).

## Liste d'analyseurs linguistiques

Voici la liste des langues prises en charge avec les noms d'analyseur Lucene et Microsoft.

LANGAGE	NOM DE L'ANALYSEUR MICROSOFT	NOM DE L'ANALYSEUR LUCENE
Arabe	ar.microsoft	ar.lucene
Arménien		hy.lucene
Bangla	bn.microsoft	

LANGAGE	NOM DE L'ANALYSEUR MICROSOFT	NOM DE L'ANALYSEUR LUCENE
Basque		eu.lucene
Bulgare	bg.microsoft	bg.lucene
Catalan	ca.microsoft	ca.lucene
Chinois (simplifié)	zh-Hans.microsoft	zh-Hans.lucene
Chinois traditionnel	zh-Hant.microsoft	zh-Hant.lucene
Croate	hr.microsoft	
Tchèque	cs.microsoft	cs.lucene
Danois	da.microsoft	da.lucene
Néerlandais	nl.microsoft	nl.lucene
Anglais	en.microsoft	en.lucene
Estonien	et.Microsoft	
Finnois	fi.microsoft	fi.lucene
Français	fr.Microsoft	fr.lucene
Galicien		gl.lucene
Allemand	de.Microsoft	de.lucene
Grec	el.microsoft	el.lucene
Goudjratî	gu.microsoft	
Hébreu	he.microsoft	
Hindi	hi.microsoft	hi.lucene
Hongrois	hu.microsoft	hu.lucene
Islandais	is.microsoft	
Indonésien	id.microsoft	id.lucene
Irlandais		ga.lucene
Italien	it.microsoft	it.lucene
Japonais	ja.Microsoft	ja.lucene

LANGAGE	NOM DE L'ANALYSEUR MICROSOFT	NOM DE L'ANALYSEUR LUCENE
Kannada	kn.microsoft	
Coréen	ko.microsoft	ko.lucene
Letton	lv.microsoft	lv.lucene
Lituaniens	lt.microsoft	
Malayalam	ml.microsoft	
Malais (latin)	ms.microsoft	
Marathi	mr.microsoft	
Norvégien	nb.Microsoft	no.lucene
Persan		fa.lucene
Polonais	pl.microsoft	pl.lucene
Portugais (Brésil)	pt-Br.microsoft	pt-Br.lucene
Portugais (Portugal)	pt-Pt.microsoft	pt-Pt.lucene
Pendjabi	pa.microsoft	
Roumain	ro.microsoft	ro.lucene
Russe	ru.microsoft	ru.lucene
Serbe (cyrillique)	sr-cyrillic.microsoft	
Serbe (latin)	sr-latin.microsoft	
Slovaque	sk.microsoft	
Slovène	sl.microsoft	
Espagnol	es.Microsoft	es.lucene
Suédois	sv.microsoft	sv.lucene
Tamoul	ta.microsoft	
Télougou	te.microsoft	
Thaï	th.microsoft	th.lucene
Turc	tr.microsoft	tr.lucene

LANGAGE	NOM DE L'ANALYSEUR MICROSOFT	NOM DE L'ANALYSEUR LUCENE
Ukrainien	uk.microsoft	
Ourdou	ur.microsoft	
Vietnamien	vi.microsoft	

Tous les analyseurs dont le nom est annoté **lucene** s'appuient sur les [analyseurs linguistiques d'Apache Lucene](#).

## Voir aussi

- [Créer un index \(API REST Recherche cognitive Azure\)](#)
- [Classe AnalyzerName](#)

# Ajouter des analyseurs personnalisés à des champs de chaîne dans l'index de Recherche cognitive Azure

04/10/2020 • 55 minutes to read • [Edit Online](#)

Un *analyseur personnalisé* est un type spécifique d'[analyseur de texte](#) qui se compose d'une combinaison définie par l'utilisateur du générateur de jetons existant et des filtres facultatifs. En combinant des générateurs de jetons et des filtres de manière innovante, vous pouvez personnaliser le traitement du texte dans le moteur de recherche pour obtenir des résultats spécifiques. Par exemple, vous pouvez créer un analyseur personnalisé avec un *filtre de caractères* pour supprimer le balisage HTML avant que les entrées de texte soient tokenisées.

Vous pouvez définir plusieurs analyseurs personnalisés pour varier la combinaison de filtres, mais chaque champ ne peut utiliser qu'un analyseur pour l'analyse de l'indexation et qu'un analyseur pour l'analyse de la recherche. Pour savoir à quoi ressemble un analyseur de client, consultez [Exemple d'analyseur personnalisé](#).

## Vue d'ensemble

En termes simples, le rôle d'un [moteur de recherche en texte intégral](#) est de traiter et de stocker des documents afin de permettre une interrogation et une récupération efficaces. À un niveau élevé, tout cela se résume à extraire les mots importants des documents, à les placer dans un index, puis à utiliser ce dernier pour rechercher les documents qui correspondent aux mots d'une requête donnée. Le processus d'extraction de mots des documents et de requêtes de recherche est appelé *analyse lexicale*. Les composants qui effectuent une analyse lexicale sont appelés des *analyseurs*.

Dans Recherche cognitive Azure, vous pouvez choisir parmi un ensemble prédéfini d'analyseurs indépendants des langues dans le tableau [Analyseurs](#) ou dans des analyseurs spécifiques à des langues listés dans [Analyseurs linguistiques \(API REST Recherche cognitive Azure\)](#). Vous pouvez également définir vos propres analyseurs personnalisés.

Un analyseur personnalisé vous permet de contrôler le processus de conversion du texte en jetons indexables et utilisables pour la recherche. Il s'agit d'une configuration définie par l'utilisateur constituée d'un seul générateur de jetons prédéfini, d'un ou plusieurs filtres de jetons et d'un ou plusieurs filtres de caractères. Le générateur de jetons est chargé de segmenter le texte en jetons tandis que les filtres de jeton modifient les jetons émis par le générateur de jetons. Les filtres de caractères sont appliqués pour préparer le texte en entrée avant son traitement par le générateur de jetons. Par exemple, le filtre de caractères peut remplacer certains caractères ou certains symboles.

Les scénarios courants permis par les analyseurs personnalisés sont les suivants :

- Recherche phonétique. Ajoutez un filtre phonétique pour permettre une recherche basée sur la façon dont un mot est prononcé et non pas sur la façon dont il s'écrit.
- Analyse lexicale désactivée. Utilisez l'analyseur de mots clés pour créer des champs utilisables pour la recherche qui ne sont pas analysés.
- Recherche rapide de préfixe/suffixe. Ajoutez le filtre de jetons Edge N-gram pour indexer les préfixes des mots afin de permettre la mise en correspondance rapide des préfixes. Combinez-le avec le filtre de jeton Reverse pour effectuer une correspondance de suffixes.
- Segmentation personnalisée du texte en unités lexicales. Par exemple, utilisez le générateur de jetons Whitespace pour segmenter les phrases en jetons en utilisant l'espace comme délimiteur.

- Conversion ASCII. Ajoutez le filtre de conversion ASCII Standard pour normaliser les signes diacritiques, comme ö ou ê, dans les termes de recherche.

Cette page contient la liste des analyseurs, des générateurs de jetons, des filtres de jetons et des filtres de caractères pris en charge. Vous trouverez également la description des modifications apportées à la définition d'index avec un exemple d'utilisation. Pour plus d'informations sur la technologie sous-jacente exploitée dans l'implémentation de Recherche cognitive Azure, consultez l'article [Analysis package summary \(Lucene\)](#)(Résumé du package d'analyse [Lucene]). Pour obtenir des exemples de configurations d'analyseurs, consultez [Ajouter des analyseurs dans Recherche cognitive Azure](#).

## Règles de validation

Les noms des analyseurs, des générateurs de jetons, des filtres de jetons et des filtres de caractères doivent être uniques, et ils ne peuvent pas être identiques à ceux des analyseurs, générateurs de jetons, filtres de jetons et filtres de caractères prédéfinis. Consultez les [Informations de référence sur les propriétés](#) pour les noms déjà utilisés.

## Créer des analyseurs personnalisés

Vous pouvez définir des analyseurs personnalisés au moment de la création d'index. La syntaxe de spécification d'un analyseur personnalisé est décrite dans cette section. Vous pouvez également vous familiariser avec la syntaxe en examinant les exemples de définitions dans [Ajouter des analyseurs dans Recherche cognitive Azure](#).

Une définition d'analyseur inclut un nom, un type, un ou plusieurs filtres de caractères, un générateur de jetons au maximum, et un ou plusieurs filtres de jetons pour le traitement venant après la segmentation du texte en unités lexicales. Les filtres de caractères sont appliqués avant la segmentation du texte en unités lexicales. Les filtres de jetons et les filtres de caractères sont appliqués de gauche à droite.

Le `tokenizer_name` est le nom d'un générateur de jetons, `token_filter_name_1` et `token_filter_name_2` sont les noms des filtres de jetons, et `char_filter_name_1` et `char_filter_name_2` sont les noms des filtres de caractères (consultez les tableaux [Générateurs de jetons](#), [Filtres de jetons](#) et Filtres de caractères pour accéder aux valeurs valides).

La définition de l'analyseur est une partie de l'index. Consultez [API de création d'index](#) pour plus d'informations sur le reste de l'index.

```

"analyzers":(optional)[
  {
    "name":"name of analyzer",
    "@odata.type": "#Microsoft.Azure.Search.CustomAnalyzer",
    "charFilters":[
      "char_filter_name_1",
      "char_filter_name_2"
    ],
    "tokenizer":"tokenizer_name",
    "tokenFilters":[
      "token_filter_name_1",
      "token_filter_name_2"
    ]
  },
  {
    "name":"name of analyzer",
    "@odata.type": "#analyzer_type",
    "option1":value1,
    "option2":value2,
    ...
  }
],
"charFilters":(optional)[
  {
    "name":"char_filter_name",
    "@odata.type": "#char_filter_type",
    "option1":value1,
    "option2":value2,
    ...
  }
],
"tokenizers":(optional)[
  {
    "name":"tokenizer_name",
    "@odata.type": "#tokenizer_type",
    "option1":value1,
    "option2":value2,
    ...
  }
],
"tokenFilters":(optional)[
  {
    "name":"token_filter_name",
    "@odata.type": "#token_filter_type",
    "option1":value1,
    "option2":value2,
    ...
  }
]
]

```

#### **NOTE**

Les analyseurs personnalisés que vous créez ne sont pas exposés dans le portail Azure. La seule façon d'ajouter un analyseur personnalisé est de le faire dans du code qui effectue des appels à l'API lors de la définition d'un index.

Au sein d'une définition d'index, vous pouvez placer cette section n'importe où dans le corps d'une demande de création d'index, mais elle est généralement placée à la fin :

```
{  
  "name": "name_of_index",  
  "fields": [ ],  
  "suggesters": [ ],  
  "scoringProfiles": [ ],  
  "defaultScoringProfile": (optional) "...",  
  "corsOptions": (optional) { },  
  "analyzers":(optional)[ ],  
  "charFilters":(optional)[ ],  
  "tokenizers":(optional)[ ],  
  "tokenFilters":(optional)[ ]  
}
```

Les définitions des filtres de caractères, des générateurs de jetons et des filtres de jetons sont ajoutées à l'index seulement si vous définissez des options personnalisées. Pour utiliser tel quel un filtre ou un générateur de jetons existant, spécifiez son nom dans la définition de l'analyseur.

## Tester des analyseurs personnalisés

Vous pouvez utiliser l'[opération de test d'analyseur](#) dans l'[API REST](#) pour voir comment un analyseur décompose le texte donné en jetons.

### Requête

```
POST https://[search service name].search.windows.net/indexes/[index name]/analyze?api-version=[api-version]  
Content-Type: application/json  
api-key: [admin key]  
  
{  

```

### Réponse

```
{
  "tokens": [
    {
      "token": "vis_a_vis",
      "startOffset": 0,
      "endOffset": 9,
      "position": 0
    },
    {
      "token": "vis_à_vis",
      "startOffset": 0,
      "endOffset": 9,
      "position": 0
    },
    {
      "token": "means",
      "startOffset": 10,
      "endOffset": 15,
      "position": 1
    },
    {
      "token": "opposite",
      "startOffset": 16,
      "endOffset": 24,
      "position": 2
    }
  ]
}
```

## Mettre à jour des analyseurs personnalisés

Une fois qu'un analyseur, un générateur de jetons, un filtre de jetons ou un filtre de caractères est défini, il ne peut pas être modifié. D'autres peuvent être ajoutés à un index existant à condition que l'indicateur

`allowIndexDowntime` soit défini comme true dans la demande de mise à jour de l'index :

```
PUT https://[search service name].search.windows.net/indexes/[index name]?api-version=[api-version]&allowIndexDowntime=true
```

Cette opération place votre index hors connexion pendant au moins quelques secondes, ce qui entraîne l'échec de vos demandes d'indexation et de requête. Les performances et la disponibilité en écriture de l'index peuvent être altérées pendant plusieurs minutes après la mise à jour de l'index, ou plus longtemps pour les très grands index. Ces effets sont cependant temporaires et finissent par se résoudre d'eux-mêmes.

## Référence d'analyseur

Les tableaux ci-dessous listent les propriétés de configuration de la section des analyseurs, des générateurs de jetons, des filtres de jetons et des filtres de caractères d'une définition d'index. La structure d'un analyseur, d'un générateur de jetons ou d'un filtre dans votre index est composée de ces attributs. Pour plus d'informations sur l'affectation d'une valeur, consultez les [Informations de référence sur les propriétés](#).

### Analyseurs

Pour les analyseurs, les attributs d'index varient selon que vous utilisez des analyseurs prédéfinis ou des analyseurs personnalisés.

#### Analyseurs prédéfinis

TYPE	DESCRIPTION
Nom	Il doit contenir uniquement des lettres, des chiffres, des espaces, des tirets ou des traits de soulignement. Il doit commencer et se terminer uniquement par des caractères alphanumériques, et ne doit pas dépasser 128 caractères.
Type	Type de l'analyseur provenant de la liste des analyseurs pris en charge. Consultez la colonne <b>type_analyseur</b> dans le tableau <a href="#">Analyseurs</a> ci-dessous.
Options	Il doit s'agir des options valides d'un analyseur prédéfini listées dans le tableau <a href="#">Analyseurs</a> ci-dessous.

#### Analyseurs personnalisés

TYPE	DESCRIPTION
Nom	Il doit contenir uniquement des lettres, des chiffres, des espaces, des tirets ou des traits de soulignement. Il doit commencer et se terminer uniquement par des caractères alphanumériques, et ne doit pas dépasser 128 caractères.
Type	Doit être « #Microsoft.Azure.Search.CustomAnalyzer ».
CharFilters	Défini sur un des filtres de caractères prédéfinis listés dans le tableau <a href="#">Filtres de caractères</a> ou sur un filtre de caractères personnalisé spécifié dans la définition d'index.
Générateur de jetons	Obligatoire. Défini sur un des générateurs de jetons prédéfinis listés dans le tableau <a href="#">Générateur de jetons</a> ou sur un générateur de jetons personnalisé spécifié dans la définition d'index.
TokenFilters	Défini sur un des filtres de jetons prédéfinis listés dans le tableau <a href="#">Filtres de jetons</a> ou sur un filtre de jetons personnalisé spécifié dans la définition d'index.

#### NOTE

Il est nécessaire de configurer votre analyseur personnalisé de façon à ne pas produire des jetons d'une longueur supérieure à 300 caractères. L'indexation échoue pour les documents avec de tels jetons. Pour les tronquer ou les ignorer, utilisez respectivement `TruncateTokenFilter` et `LengthTokenFilter`. Voir [Filtres de jeton](#) pour en savoir plus.

#### Filtres de caractères

Un filtre de caractères est utilisé pour préparer le texte en entrée avant son traitement par le générateur de jetons. Par exemple, il peut remplacer certains caractères ou certains symboles. Vous pouvez avoir plusieurs filtres de caractères dans un analyseur personnalisé. Les filtres de caractères s'exécutent dans l'ordre où ils sont listés.

TYPE	DESCRIPTION
------	-------------

TYPE	DESCRIPTION
Nom	Il doit contenir uniquement des lettres, des chiffres, des espaces, des tirets ou des traits de soulignement. Il doit commencer et se terminer uniquement par des caractères alphanumériques, et ne doit pas dépasser 128 caractères.
Type	Type de filtre de caractères provenant de la liste des filtres de caractères pris en charge. Consultez la colonne <b>type_filtre_caractères</b> dans le tableau <a href="#">Filtres de caractères</a> ci-dessous.
Options	Il doit s'agir des options valides d'un type de <a href="#">Filtre de caractères</a> donné.

## Générateurs de jetons

Un générateur de jetons divise un texte continu en une séquence de jetons, par exemple en divisant une phrase en mots.

Vous pouvez spécifier un seul générateur de jetons par analyseur personnalisé. Si vous avez besoin de plusieurs générateurs de jetons, vous pouvez créer plusieurs analyseurs personnalisés et les affecter champ par champ dans votre schéma d'index.

Un analyseur personnalisé peut utiliser un générateur de jetons prédéfini avec des options personnalisées ou par défaut.

TYPE	DESCRIPTION
Nom	Il doit contenir uniquement des lettres, des chiffres, des espaces, des tirets ou des traits de soulignement. Il doit commencer et se terminer uniquement par des caractères alphanumériques, et ne doit pas dépasser 128 caractères.
Type	Nom du générateur de jetons provenant de la liste des générateurs de jetons pris en charge. Consultez la colonne <b>type_générateur_jetons</b> dans le tableau <a href="#">Générateurs de jetons</a> ci-dessous.
Options	Il doit s'agir des options valides d'un type de générateur de jetons donné listé dans le tableau <a href="#">Générateurs de jetons</a> ci-dessous.

## Filtres de jeton

Un filtre de jetons est utilisé pour filtrer ou modifier les jetons générés par un générateur de jetons. Par exemple, vous pouvez spécifier un filtre lowercase qui convertit tous les caractères en minuscules.

Vous pouvez avoir plusieurs filtres de jetons dans un analyseur personnalisé. Les filtres de jetons s'exécutent dans l'ordre où ils sont listés.

TYPE	DESCRIPTION
Nom	Il doit contenir uniquement des lettres, des chiffres, des espaces, des tirets ou des traits de soulignement. Il doit commencer et se terminer uniquement par des caractères alphanumériques, et ne doit pas dépasser 128 caractères.

TYPE	DESCRIPTION
Type	Nom du filtre de jetons provenant de la liste des filtres de jetons pris en charge. Consultez la colonne <b>type_filtre_jetons</b> dans le tableau <a href="#">Filtres de jetons</a> ci-dessous.
Options	Il doit s'agir de <a href="#">Filtres de jeton</a> d'un type de filtre de jetons donné.

## Informations de référence sur les propriétés

Cette section fournit les valeurs valides pour les attributs spécifiés dans la définition d'un analyseur, d'un générateur de jetons, d'un filtre de caractères ou d'un filtre de jetons personnalisé dans votre index. Les analyseurs, les générateurs et les filtres qui sont implémentés avec Apache Lucene ont des liens vers la documentation de l'API Lucene.

### Informations de référence sur les analyseurs prédéfinis

NOM_ANALYSEUR	ANALYZER_TYPE <sup>1</sup>	DESCRIPTION ET OPTIONS
<a href="#">keyword</a>	(le type s'applique seulement quand des options sont disponibles)	Traite l'intégralité du contenu d'un champ comme un seul jeton. Ceci est utile pour les données comme les codes postaux, les ID et certains noms de produit.
<a href="#">pattern</a>	PatternAnalyzer	<p>Sépare le texte de façon flexible en termes via un modèle d'expression régulière.</p> <p><b>Options</b></p> <p><b>lowercase</b> (type : booléen) : détermine si les termes sont en minuscules. La valeur par défaut est true.</p> <p><b>pattern</b> (type : chaîne) : un modèle d'expression régulière pour mettre en correspondance les séparateurs de jetons. La valeur par défaut est <code>\w+</code>, qui correspond aux caractères non alphabétiques.</p> <p><b>flags</b> (type : chaîne) : indicateurs d'expression régulière. La valeur par défaut est une chaîne vide. Valeurs autorisées : CANON_EQ, CASE_INSENSITIVE, COMMENTS, DOTALL, LITERAL, MULTILINE, UNICODE_CASE, UNIX_LINES</p> <p><b>stopwords</b> (type : tableau de chaînes) : une liste de mots vides. La valeur par défaut est une liste vide.</p>
<a href="#">simple</a>	(le type s'applique seulement quand des options sont disponibles)	Divise le texte à l'endroit des caractères qui ne sont pas des lettres et le convertit en minuscules.

NOM_ANALYSEUR	ANALYZER_TYPE	DESCRIPTION ET OPTIONS
<a href="#">standard</a> (Également appelé standard.lucene)	StandardAnalyzer	<p>Analyseur Lucene Standard, composé du générateur de jetons standard, du filtre lowercase et du filtre stop.</p> <p><b>Options</b></p> <p>maxTokenLength (type : entier) : la longueur maximale des jetons. La valeur par défaut est 255. Les jetons dépassant la longueur maximale sont fractionnés. La longueur maximale des jetons qui peut être utilisée est de 300 caractères.</p> <p>stopwords (type : tableau de chaînes) : une liste de mots vides. La valeur par défaut est une liste vide.</p>
standardasciifolding.lucene	(le type s'applique seulement quand des options sont disponibles)	Analyseur standard avec filtre de conversion ASCII.
<a href="#">stop</a>	StopAnalyzer	<p>Divise un texte à l'endroit des caractères qui ne sont pas des lettres, applique les filtres de jetons lowercase et stopword.</p> <p><b>Options</b></p> <p>stopwords (type : tableau de chaînes) : une liste de mots vides. La valeur par défaut est une liste prédefinie pour l'anglais.</p>
<a href="#">whitespace</a>	(le type s'applique seulement quand des options sont disponibles)	Un analyseur qui utilise le générateur de jetons whitespace. Les jetons d'une longueur supérieure à 255 caractères sont fractionnés.

<sup>1</sup> Les types d'analyseurs sont toujours préfixés dans le code par « #Microsoft.Azure.Search » ; ainsi, « PatternAnalyzer » serait spécifié sous la forme « #Microsoft.Azure.Search.PatternAnalyzer ». Nous avons supprimé le préfixe par souci de concision, mais ce préfixe est obligatoire dans votre code.

Le type d'analyseur (type\_analyseur) est fourni seulement pour les analyseurs qui peuvent être personnalisés. S'il n'existe pas d'options, comme c'est le cas avec l'analyseur keyword, aucun type #Microsoft.Azure.Search n'est associé.

### Informations de référence sur les filtres de caractères

Dans le tableau ci-dessous, les filtres de caractères qui sont implémentés avec Apache Lucene sont liés à la documentation de l'API Lucene.

NOM_FILTER_CARACTÈRES	CHAR_FILTER_TYPE <sup>1</sup>	DESCRIPTION ET OPTIONS
<a href="#">html_strip</a>	(le type s'applique seulement quand des options sont disponibles)	Un filtre de caractères qui tente d'enlever retirer les constructions HTML.

NOM_FILTER_CARACTÈRES	CHAR_FILTER_TYPE	DESCRIPTION ET OPTIONS
mapping	MappingCharFilter	<p>Un filtre de caractères applique des mappages définis avec l'option mappings. La mise en correspondance est gourmande en ressources (la correspondance du modèle le plus long à un point donné l'emporte). La chaîne vide est autorisée comme remplacement.</p> <p><b>Options</b></p> <p>Mappings (type : tableau de chaînes) : une liste de mappages au format suivant : « a=&gt;b » (toutes les occurrences du caractère « a » sont remplacées par le caractère « b »). Obligatoire.</p>
pattern_replace	PatternReplaceCharFilter	<p>Un filtre de caractères qui remplace des caractères dans la chaîne d'entrée. Il utilise une expression régulière pour identifier les séquences de caractères à conserver et un modèle de remplacement pour identifier les caractères à remplacer. Par exemple, texte d'entrée = "aa bb aa bb", modèle = "(aa)\\s+(bb)" remplacement = "\$1#\$2", résultat = "aa#bb aa#bb".</p> <p><b>Options</b></p> <p>pattern (type : chaîne) : obligatoire.</p> <p>replacement (type : chaîne) : obligatoire.</p>

<sup>1</sup> Les types de filtres de caractères sont toujours préfixés dans le code par « #Microsoft.Azure.Search » ; ainsi, « MappingCharFilter » serait spécifié sous la forme « #Microsoft.Azure.Search.MappingCharFilter ». Nous avons supprimé le préfixe pour réduire la largeur du tableau, mais n'oubliez pas de l'inclure dans votre code. Notez que le type de filtre de caractères (type\_filtre\_caractères) est fourni seulement pour les filtres qui peuvent être personnalisés. S'il n'existe pas d'options, comme c'est le cas avec html\_strip, aucun type #Microsoft.Azure.Search n'est associé.

### Informations de référence sur les générateurs de jetons

Dans le tableau ci-dessous, les générateurs de jetons qui sont implémentés avec Apache Lucene sont liés à la documentation de l'API Lucene.

NOM_GÉNÉRATEUR_JETONS	TOKENIZER_TYPE <sup>1</sup>	DESCRIPTION ET OPTIONS
-----------------------	-----------------------------	------------------------

NOM_GÉNÉRATEUR_JETONS	TOKENIZER_TYPE	DESCRIPTION ET OPTIONS
classique	ClassicTokenizer	<p>Générateur de jetons basé sur la grammaire qui convient pour le traitement des documents dans la plupart des langues européennes.</p> <p><b>Options</b></p> <p>maxTokenLength (type : entier) : la longueur maximale des jetons. Valeur par défaut : 255, maximum : 300. Les jetons dépassant la longueur maximale sont fractionnés.</p>
edgeNGram	EdgeNGramTokenizer	<p>Génère des jetons à partir de l'entrée depuis une délimitation en n-grammes d'une ou plusieurs tailles données.</p> <p><b>Options</b></p> <p>minGram (type : entier) : par défaut : 1, maximum : 300.</p> <p>maxGram (type : entier) : par défaut : 2, maximum : 300. Doit être supérieur à minGram.</p> <p>tokenChars (type : tableau de chaînes) : classes de caractères à conserver dans les jetons. Valeurs autorisées : "letter", "digit", "whitespace", "punctuation", "symbol". La valeur par défaut est un tableau vide - conserve tous les caractères.</p>
keyword_v2	KeywordTokenizerV2	<p>Génère la totalité de l'entrée sous la forme d'un unique jeton.</p> <p><b>Options</b></p> <p>maxTokenLength (type : entier) : la longueur maximale des jetons. Valeur par défaut : 256, maximum : 300. Les jetons dépassant la longueur maximale sont fractionnés.</p>
letter	(le type s'applique seulement quand des options sont disponibles)	Divise un texte à l'endroit des caractères qui ne sont pas des lettres. Les jetons d'une longueur supérieure à 255 caractères sont fractionnés.
lowercase	(le type s'applique seulement quand des options sont disponibles)	Divise le texte à l'endroit des caractères qui ne sont pas des lettres et le convertit en minuscules. Les jetons d'une longueur supérieure à 255 caractères sont fractionnés.

NOM_GÉNÉRATEUR_JETONS	TOKENIZER_TYPE	DESCRIPTION ET OPTIONS
microsoft_language_tokenizer	MicrosoftLanguageTokenizer	<p>Divise le texte en utilisant des règles spécifiques à la langue.</p> <p><b>Options</b></p> <p>maxTokenLength (type : entier) : la longueur maximale des jetons. Par défaut : 255, maximum : 300. Les jetons dépassant la longueur maximale sont fractionnés. Les jetons de plus de 300 caractères sont d'abord fractionnés en jetons d'une longueur de 300 caractères, puis chacun de ces jetons est ensuite fractionné selon la valeur définie pour maxTokenLength.</p> <p>isSearchTokenizer (type : booléen) : défini sur true s'il est utilisé comme générateur de jetons de recherche, défini sur false s'il est utilisé comme générateur de jetons d'indexation.</p> <p>language (type : chaîne) : langue à utiliser, par défaut « english ». Les valeurs autorisées sont les suivantes : "bangla", "bulgarian", "catalan", "chineseSimplified", "chineseTraditional", "croatian", "czech", "danish", "dutch", "english", "french", "german", "greek", "gujarati", "hindi", "icelandic", "indonesian", "italian", "japanese", "kannada", "korean", "malay", "malayalam", "marathi", "norwegianBokmaal", "polish", "portuguese", "portugueseBrazilian", "punjabi", "romanian", "russian", "serbianCyrillic", "serbianLatin", "slovenian", "spanish", "swedish", "tamil", "telugu", "thai", "ukrainian", "urdu", "vietnamese"</p>

NOM_GÉNÉRATEUR_JETONS	TOKENIZER_TYPE	DESCRIPTION ET OPTIONS
microsoft_language_stemming_tokenizer	MicrosoftLanguageStemmingTokenizer	<p>Divise le texte en utilisant des règles spécifiques à la langue et réduit les mots à leurs formes de base</p> <p><b>Options</b></p> <p>maxTokenLength (type : entier) : la longueur maximale des jetons. Par défaut : 255, maximum : 300. Les jetons dépassant la longueur maximale sont fractionnés. Les jetons de plus de 300 caractères sont d'abord fractionnés en jetons d'une longueur de 300 caractères, puis chacun de ces jetons est ensuite fractionné selon la valeur définie pour maxTokenLength.</p> <p>isSearchTokenizer (type : booléen) : défini sur true s'il est utilisé comme générateur de jetons de recherche, défini sur false s'il est utilisé comme générateur de jetons d'indexation.</p> <p>language (type : chaîne) : langue à utiliser, par défaut « english ». Les valeurs autorisées sont les suivantes : "arabic", "bangla", "bulgarian", "catalan", "croatian", "czech", "danish", "dutch", "english", "estonian", "finnish", "french", "german", "greek", "gujarati", "hebrew", "hindi", "hungarian", "icelandic", "indonesian", "italian", "kannada", "latvian", "lithuanian", "malay", "malayalam", "marathi", "norwegianBokmaal", "polish", "portuguese", "portugueseBrazilian", "punjabi", "romanian", "russian", "serbianCyrillic", "serbianLatin", "slovak", "slovenian", "spanish", "swedish", "tamil", "telugu", "turkish", "ukrainian", "urdu"</p>

NOM_GÉNÉRATEUR_JETONS	TOKENIZER_TYPE	DESCRIPTION ET OPTIONS
nGram	NGramTokenizer	<p>Génère des jetons à partir de l'entrée en n-grammes d'une ou plusieurs tailles données.</p> <p><b>Options</b></p> <p>minGram (type : entier) : par défaut : 1, maximum : 300.</p> <p>maxGram (type : entier) : par défaut : 2, maximum : 300. Doit être supérieur à minGram.</p> <p>tokenChars (type : tableau de chaînes) : classes de caractères à conserver dans les jetons. Valeurs autorisées : "letter", "digit", "whitespace", "punctuation", "symbol". La valeur par défaut est un tableau vide - conserve tous les caractères.</p>
path_hierarchy_v2	PathHierarchyTokenizerV2	<p>Générateur de jetons pour les hiérarchies de type chemin.</p> <p><b>Options</b></p> <p>delimiter (type : chaîne) : par défaut : '/.</p> <p>replacement (type : chaîne) : s'il est défini, remplace le caractère délimiteur. Par défaut, identique à la valeur du délimiteur.</p> <p>maxTokenLength (type : entier) : la longueur maximale des jetons. Valeur par défaut : 300, maximum : 300. Les chemins plus longs que maxTokenLength sont ignorés.</p> <p>reverse (type : booléen) : si la valeur est true, génère le jeton dans l'ordre inverse. Valeur par défaut : false.</p> <p>skip (type : booléen) : jetons initiaux à ignorer. La valeur par défaut est 0.</p>

NOM_GÉNÉRATEUR_JETONS	TOKENIZER_TYPE	DESCRIPTION ET OPTIONS
pattern	PatternTokenizer	<p>Ce générateur de jetons utilise la correspondance de modèle d'expression régulière pour construire des jetons distincts.</p> <p><b>Options</b></p> <p><b>pattern</b> (type : chaîne) : un modèle d'expression régulière pour mettre en correspondance les séparateurs de jetons. La valeur par défaut est <code>\w+</code>, qui correspond aux caractères non alphabétiques.</p> <p><b>flags</b> (type : chaîne) : indicateurs d'expression régulière. La valeur par défaut est une chaîne vide. Valeurs autorisées : CANON_EQ, CASE_INSENSITIVE, COMMENTS, DOTALL, LITERAL, MULTILINE, UNICODE_CASE, UNIX_LINES</p> <p><b>group</b> (type : entier) : groupe à extraire dans les jetons. La valeur par défaut est -1 (diviser).</p>
standard_v2	StandardTokenizerV2	<p>Décompose le texte en suivant les <a href="#">règles de segmentation du texte Unicode</a>.</p> <p><b>Options</b></p> <p><b>maxTokenLength</b> (type : entier) : la longueur maximale des jetons. Valeur par défaut : 255, maximum : 300. Les jetons dépassant la longueur maximale sont fractionnés.</p>
uax_url_email	UaxUrlEmailTokenizer	<p>Génère des jetons pour des URL et des e-mails sous la forme d'un seul jeton.</p> <p><b>Options</b></p> <p><b>maxTokenLength</b> (type : entier) : la longueur maximale des jetons. Valeur par défaut : 255, maximum : 300. Les jetons dépassant la longueur maximale sont fractionnés.</p>
whitespace	(le type s'applique seulement quand des options sont disponibles)	Divise le texte au niveau des espaces. Les jetons d'une longueur supérieure à 255 caractères sont fractionnés.

<sup>1</sup> Les types de générateurs de jetons sont toujours préfixés dans le code par « #Microsoft.Azure.Search » ; ainsi, « ClassicTokenizer » serait spécifié sous la forme « #Microsoft.Azure.Search.ClassicTokenizer ». Nous avons supprimé le préfixe pour réduire la largeur du tableau, mais n'oubliez pas de l'inclure dans votre code. Notez que le type de générateur de jetons (type\_générateur\_jetons) est fourni seulement pour les générateurs de jetons qui peuvent être personnalisés. S'il n'existe pas d'options, comme c'est le cas avec le générateur de

jetons letter, aucun type #Microsoft.Azure.Search n'est associé.

## Informations de référence sur les filtres de jetons

Dans le tableau ci-dessous, les filtres de jetons qui sont implémentés avec Apache Lucene sont liés à la documentation de l'API Lucene.

NOM_FILTRE_JETONS	TOKEN_FILTER_TYPE <sup>1</sup>	DESCRIPTION ET OPTIONS
arabic_normalization	(le type s'applique seulement quand des options sont disponibles)	Un filtre de jetons qui applique le normaliseur arabe pour normaliser l'orthographe.
apostrophe	(le type s'applique seulement quand des options sont disponibles)	Supprime tous les caractères suivant une apostrophe (y compris l'apostrophe elle-même).
asciifolding	AsciiFoldingTokenFilter	<p>Convertit les caractères Unicode alphabétiques, numériques et symboliques qui ne sont pas dans les 127 premiers caractères ASCII (le bloc Unicode « Basic Latin ») en leur équivalent ASCII, s'il existe.</p> <p><b>Options</b></p> <p>preserveOriginal (type : booléen) : si la valeur est true, le jeton d'origine est conservé. La valeur par défaut est false.</p>
cjk_bigram	CjkBigramTokenFilter	<p>Forme des digrammes de termes CJC qui sont générés à partir de StandardTokenizer.</p> <p><b>Options</b></p> <p>ignoreScripts (type : tableau de chaînes) : scripts à ignorer. Les valeurs autorisées sont : "han", "hiragana", "katakana", "hangul". La valeur par défaut est une liste vide.</p> <p>outputUnigrams (type : booléen) : à définir sur true si vous voulez toujours obtenir en sortie des unigrammes et des digrammes. La valeur par défaut est false.</p>
cjk_width	(le type s'applique seulement quand des options sont disponibles)	Normalise les différences de largeur de CJC. Convertit les variantes ASCII pleine chasse en équivalent Latin de base, et les variantes Katakana demi-chasse en équivalent kana.
classique	(le type s'applique seulement quand des options sont disponibles)	Supprime les possessifs anglais et les points des acronymes.

NOM_FILTRE_JETONS	TOKEN_FILTER_TYPE	DESCRIPTION ET OPTIONS
common_grams	CommonGramTokenFilter	<p>Construit des digrammes pour les termes d'occurrence fréquente lors de l'indexation. Les termes uniques sont néanmoins aussi indexés, avec des digrammes superposés.</p> <p><b>Options</b></p> <p>commonWords (type : tableau de chaînes) : l'ensemble des mots courants. La valeur par défaut est une liste vide. Obligatoire.</p> <p>ignoreCase (type : booléen) : Si la valeur est true, la mise en correspondance ne respecte pas la casse. La valeur par défaut est false.</p> <p>queryMode (type : booléen) : génère des digrammes, puis supprime les mots courants et les termes uniques suivis d'un mot courant. La valeur par défaut est false.</p>
dictionary_decompounder	DictionaryDecompounderTokenFilter	<p>Décompose les mots composés trouvés dans beaucoup de langues germaniques.</p> <p><b>Options</b></p> <p>wordList (type : tableau de chaînes) : la liste de mots dans laquelle rechercher des correspondances. La valeur par défaut est une liste vide. Obligatoire.</p> <p>minWordSize (type : entier) : seuls les mots d'une longueur supérieure à cette taille sont traités. La valeur par défaut est 5.</p> <p>minSubwordSize (type : entier) : seuls les sous-mots d'une longueur supérieure à cette taille figurent dans les résultats. La valeur par défaut est 2.</p> <p>maxSubwordSize (type : entier) : seuls les sous-mots d'une longueur inférieure à cette taille figurent dans les résultats. La valeur par défaut est 15.</p> <p>onlyLongestMatch (type : booléen) : ajoute seulement le sous-mot correspondant le plus long à la sortie. La valeur par défaut est false.</p>

NOM_FILTRE_JETONS	TOKEN_FILTER_TYPE	DESCRIPTION ET OPTIONS
edgeNGram_v2	EdgeNGramTokenFilterV2	<p>Génère des n-grammes de la ou des tailles données en démarrant de l'avant ou de l'arrière d'un jeton d'entrée.</p> <p><b>Options</b></p> <p>minGram (type : entier) : par défaut : 1, maximum : 300.</p> <p>maxGram (type : entier) : par défaut : 2, maximum 300. Doit être supérieur à minGram.</p> <p>side (type : chaîne) : spécifie le côté de l'entrée à partir duquel le n-gramme doit être généré. Valeurs autorisées : "front", "back"</p>
elision	ElisionTokenFilter	<p>Supprime les élisions. Par exemple, « l'avion » est converti en « avion ».</p> <p><b>Options</b></p> <p>articles (type : tableau de chaînes) : un ensemble d'articles à supprimer. La valeur par défaut est une liste vide. Si aucune liste d'articles n'est définie, par défaut, tous les articles du français sont supprimés.</p>
german_normalization	(le type s'applique seulement quand des options sont disponibles)	Normalise les caractères allemands en fonction de l'heuristique de l' <a href="#">algorithme Snowball German2</a> .
hindi_normalization	(le type s'applique seulement quand des options sont disponibles)	Normalise le texte dans Hindi de façon à supprimer des différences dans les variations orthographiques.
indic_normalization	IndicNormalizationTokenFilter	Normalise la représentation Unicode du texte dans les langues indiennes.
keep	KeepTokenFilter	<p>Filtre de jetons qui conserve seulement les jetons avec du texte contenu dans la liste de mots spécifiée.</p> <p><b>Options</b></p> <p>keepWords (type : tableau de chaînes) : une liste de mots à conserver. La valeur par défaut est une liste vide. Obligatoire.</p> <p>keepWordsCase (type : booléen) : si la valeur est true, convertit d'abord tous les mots en minuscules. La valeur par défaut est false.</p>

NOM_FILTRE_JETONS	TOKEN_FILTER_TYPE	DESCRIPTION ET OPTIONS
keyword_marker	KeywordMarkerTokenFilter	<p>Marque les termes comme mots clés.</p> <p><b>Options</b></p> <p>keywords (type : tableau de chaînes) : une liste de mots à marquer comme mots clés. La valeur par défaut est une liste vide. Obligatoire.</p> <p>ignoreCase (type : booléen) : si la valeur est true, convertit d'abord tous les mots en minuscules. La valeur par défaut est false.</p>
keyword_repeat	(le type s'applique seulement quand des options sont disponibles)	Génère deux fois chaque jeton entrant : une fois comme mot clé et une fois comme non-mot clé.
kstem	(le type s'applique seulement quand des options sont disponibles)	Un filtre kstem à hautes performances pour l'anglais.
length	LengthTokenFilter	<p>Supprime les mots qui sont trop longs ou trop courts.</p> <p><b>Options</b></p> <p>min (type : entier) : le nombre minimal. Valeur par défaut : 0, maximum : 300.</p> <p>max (type : entier) : le nombre maximal. Valeur par défaut : 300, maximum : 300.</p>
limit	Microsoft.Azure.Search.LimitTokenFilter	<p>Limite le nombre de jetons lors de l'indexation.</p> <p><b>Options</b></p> <p>maxTokenCount (type : entier) : nombre maximal de jetons à produire. La valeur par défaut est 1.</p> <p>consumeAllTokens (type : booléen) : indique si tous les jetons de l'entrée doivent être utilisés même si maxTokenCount est atteint. La valeur par défaut est false.</p>
lowercase	(le type s'applique seulement quand des options sont disponibles)	Normalise le texte des jetons en minuscules.

NOM_FILTRE_JETONS	TOKEN_FILTER_TYPE	DESCRIPTION ET OPTIONS
nGram_v2	NGramTokenFilterV2	<p>Génère des n-grammes de la taille donnée.</p> <p><b>Options</b></p> <p>minGram (type : entier) : par défaut : 1, maximum : 300.</p> <p>maxGram (type : entier) : par défaut : 2, maximum 300. Doit être supérieur à minGram.</p>
pattern_capture	PatternCaptureTokenFilter	<p>Utilise des expressions régulières Java pour générer plusieurs jetons, un pour chaque groupe de capture dans un ou plusieurs modèles.</p> <p><b>Options</b></p> <p>patterns (type : tableau de chaînes) : une liste de modèles à mettre en correspondance avec chaque jeton. Obligatoire.</p> <p>preserveOriginal (type : booléen) : à définir sur true pour retourner le jeton d'origine, même si un des modèles est en correspondance. Valeur par défaut : true</p>
pattern_replace	PatternReplaceTokenFilter	<p>Un filtre de jeton qui applique un modèle à chaque jeton du flux, en remplaçant les occurrences en correspondance par la chaîne de remplacement spécifiée.</p> <p><b>Options</b></p> <p>pattern (type : chaîne) : obligatoire.</p> <p>replacement (type : chaîne) : obligatoire.</p>
persian_normalization	(le type s'applique seulement quand des options sont disponibles)	Applique la normalisation pour le persan.

NOM_FILTRE_JETONS	TOKEN_FILTER_TYPE	DESCRIPTION ET OPTIONS
phonetic	PhoneticTokenFilter	<p>Crée des jetons pour les correspondances phonétiques.</p> <p><b>Options</b></p> <p>encoder (type : chaîne) : Encodeur phonétique à utiliser. Les valeurs autorisées sont : "metaphone", "doubleMetaphone", "soundex", "refinedSoundex", "caverphone1", "caverphone2", "cologne", "nysiis", "koelnerPhonetik", "haasePhonetik", "beiderMorse". Par défaut : "metaphone". La valeur par défaut est metaphone.</p> <p>Pour plus d'informations, consultez <a href="#">encoder</a>.</p> <p>replace (type : booléen) : true si les jetons encodés doivent remplacer les jetons d'origine ; false s'ils doivent être ajoutés comme synonymes. La valeur par défaut est true.</p>
porter_stem	(le type s'applique seulement quand des options sont disponibles)	Transforme le flux de jetons selon <a href="#">l'algorithme de recherche de radical de Porter</a> .
reverse	(le type s'applique seulement quand des options sont disponibles)	Inverse la chaîne des jetons.
scandinavian_normalization	(le type s'applique seulement quand des options sont disponibles)	Normalise l'utilisation des caractères scandinaves interchangeables.
scandinavian_folding	(le type s'applique seulement quand des options sont disponibles)	Convertit les caractères scandinaves åÄäæÄÆ->a et öÖøØ->o. Il identifie aussi l'utilisation des voyelles doubles aa, ae, ao, oe et oo, et conserve seulement la première voyelle.

NOM_FILTRE_JETONS	TOKEN_FILTER_TYPE	DESCRIPTION ET OPTIONS
shingle	ShingleTokenFilter	<p>Crée des combinaisons de jetons sous la forme d'un unique jeton.</p> <p><b>Options</b></p> <p>maxShingleSize (type : entier) : la valeur par défaut est 2.</p> <p>minShingleSize (type : entier) : la valeur par défaut est 2.</p> <p>outputUnigrams (type : booléen) : si la valeur est true, le flux de sortie contient les jetons d'entrée (unigrammes) ainsi que les n-grammes composés de mots. La valeur par défaut est true.</p> <p>outputUnigramsIfNoShingles (type : booléen) : si la valeur est true, remplace le comportement de outputUnigrams==false quand aucun n-gramme composé de mots n'est disponible. La valeur par défaut est false.</p> <p>tokenSeparator (type : chaîne) : la chaîne à utiliser lors de la jonction de jetons adjacents pour former un n-gramme composé de mots. La valeur par défaut est " ".</p> <p>filterToken (type : chaîne) : la chaîne à insérer pour chaque position à laquelle il n'y a pas de jeton. La valeur par défaut est " ".</p>
snowball	SnowballTokenFilter	<p>Filtre de jetons Snowball.</p> <p><b>Options</b></p> <p>language (type : chaîne) : les valeurs autorisées sont : "armenian", "basque", "catalan", "danish", "dutch", "english", "finnish", "french", "german", "german2", "hungarian", "italian", "kp", "lovins", "norwegian", "porter", "portuguese", "romanian", "russian", "spanish", "swedish", "turkish"</p>
sorani_normalization	SoraniNormalizationTokenFilter	<p>Normalise la représentation Unicode du texte en sorani.</p> <p><b>Options</b></p> <p>Aucun.</p>

NOM_FILTRE_JETONS	TOKEN_FILTER_TYPE	DESCRIPTION ET OPTIONS
stemmer	StemmerTokenFilter	<p>Filtre de recherche de radical spécifique à la langue.</p> <p><b>Options</b></p> <p>language (type : chaîne) : les valeurs autorisées sont :</p> <ul style="list-style-type: none"> <li>- "arabic"</li> <li>- "armenian"</li> <li>- "basque"</li> <li>- "brazilian"</li> <li>- "bulgarian"</li> <li>- "catalan"</li> <li>- "czech"</li> <li>- "danish"</li> <li>- "dutch"</li> <li>- "dutchKp"</li> <li>- "english"</li> <li>- "lightEnglish"</li> <li>- "minimalEnglish"</li> <li>- "possessiveEnglish"</li> <li>- "porter2"</li> <li>- "lovins"</li> <li>- "finnish"</li> <li>- "lightFinnish"</li> <li>- "french"</li> <li>- "lightFrench"</li> <li>- "minimalFrench"</li> <li>- "galician"</li> <li>- "minimalGalician"</li> <li>- "german"</li> <li>- "german2"</li> <li>- "lightGerman"</li> <li>- "minimalGerman"</li> <li>- "greek"</li> <li>- "hindi"</li> <li>- "hungarian"</li> <li>- "lightHungarian"</li> <li>- "indonesian"</li> <li>- "irish"</li> <li>- "italian"</li> <li>- "lightItalian"</li> <li>- "sorani"</li> <li>- "latvian"</li> <li>- "norwegian"</li> <li>- "lightNorwegian"</li> <li>- "minimalNorwegian"</li> <li>- "lightNynorsk"</li> <li>- "minimalNynorsk"</li> <li>- "portuguese"</li> <li>- "lightPortuguese"</li> <li>- "minimalPortuguese"</li> <li>- "portugueseRslp"</li> <li>- "romanian"</li> <li>- "russian"</li> <li>- "lightRussian"</li> <li>- "spanish"</li> <li>- "lightSpanish"</li> <li>- "swedish"</li> <li>- "lightSwedish"</li> <li>- "turkish"</li> </ul>

NOM_FILTRE_JETONS	TOKEN_FILTER_TYPE	DESCRIPTION ET OPTIONS
stemmer_override	StemmerOverrideTokenFilter	<p>Les termes dont le radical est trouvé dans un dictionnaire sont marqués comme mots clés, ce qui empêche la recherche de radical plus avant dans la chaîne. Doit être placé avant les filtres de recherche de radical.</p> <p><b>Options</b></p> <p>rules (type : tableau de chaînes) : règles de recherche de radical au format suivant "mot =&gt; radical", par exemple "ran =&gt; run". La valeur par défaut est une liste vide. Obligatoire.</p>
stopwords	StopwordsTokenFilter	<p>Supprime les mots vides d'un flux de jetons. Par défaut, le filtre utilise une liste de mots vides prédéfinie pour l'anglais.</p> <p><b>Options</b></p> <p>stopwords (type : tableau de chaînes) : une liste de mots vides. Ne peut pas être spécifié si une liste stopwordsList est spécifiée.</p> <p>stopwordsList (type : chaîne) : une liste prédéfinie de mots vides. Ne peut pas être spécifié si stopwords est spécifié. Les valeurs autorisées sont :"arabic", "armenian", "basque", "brazilian", "bulgarian", "catalan", "czech", "danish", "dutch", "english", "finnish", "french", "galician", "german", "greek", "hindi", "hungarian", "indonesian", "irish", "italian", "latvian", "norwegian", "persian", "portuguese", "romanian", "russian", "sorani", "spanish", "swedish", "thai", "turkish" ; par défaut : "english". Ne peut pas être spécifié si stopwords est spécifié.</p> <p>ignoreCase (type : booléen) : si la valeur est true, tous les mots sont d'abord convertis en minuscules. La valeur par défaut est false.</p> <p>removeTrailing (type : booléen) : si la valeur est true, ignore le dernier terme de recherche s'il s'agit d'un mot vide. La valeur par défaut est true.</p>

NOM_FILTRE_JETONS	TOKEN_FILTER_TYPE	DESCRIPTION ET OPTIONS
synonym	SynonymTokenFilter	<p>Met en correspondance des synonymes constitués d'un ou plusieurs mots dans un flux de jetons.</p> <p><b>Options</b></p> <p>synonyms (type : tableau de chaînes) : obligatoire. Liste des synonymes dans l'un des deux formats suivants :</p> <ul style="list-style-type: none"> <li>- incroyable, inouï, fabuleux =&gt; étonnant : tous les termes du côté gauche du symbole =&gt; sont remplacés par tous les termes du côté droit.</li> <li>- incroyable, inouï, fabuleux, étonnant : une liste séparée par des virgules de mots équivalents. Définissez l'option expand pour changer la façon dont cette liste est interprétée.</li> </ul> <p>ignoreCase (type : booléen) : convertit la casse de l'entrée pour la mise en correspondance. La valeur par défaut est false.</p> <p>expand (type : booléen) : si la valeur est true, tous les mots de la liste des synonymes (si la notation « =&gt; » n'est pas utilisée) sont mappés les uns aux autres.</p> <p>La liste suivante : incroyable, inouï, fabuleux, étonnant équivaut à : incroyable, inouï, fabuleux, étonnant =&gt; incroyable, inouï, fabuleux, étonnant</p> <ul style="list-style-type: none"> <li>- Si la valeur est false, la liste suivante : incroyable, inouï, fabuleux, étonnant équivaut à : incroyable, inouï, fabuleux, étonnant =&gt; incroyable.</li> </ul>
trim	(le type s'applique seulement quand des options sont disponibles)	Supprime les espaces de début et de fin des jetons.
truncate	TruncateTokenFilter	<p>Tronque les termes à une longueur spécifique.</p> <p><b>Options</b></p> <p>length (type : entier) : par défaut : 300, maximum : 300. Obligatoire.</p>

NOM_FILTRE_JETONS	TOKEN_FILTER_TYPE	DESCRIPTION ET OPTIONS
unique	UniqueTokenFilter	<p>Élimine les jetons avec le même texte que le jeton précédent.</p> <p><b>Options</b></p> <p>onlyOnSamePosition (type : booléen) : s'il est défini, supprime les doublons seulement à la même position. La valeur par défaut est true.</p>
uppercase	(le type s'applique seulement quand des options sont disponibles)	Normalise le texte des jetons en majuscules.
word_delimiter	WordDelimiterTokenFilter	<p>Divise les mots en sous-mots et effectue des transformations facultatives sur les groupes de sous-mots.</p> <p><b>Options</b></p> <p>generateWordParts (type : booléen) : provoque la génération de parties de mots ; par exemple « AzureSearch » devient « Azure » « Search ». La valeur par défaut est true.</p> <p>generateNumberParts (type : booléen) : provoque la génération de sous-mots constitués de nombres. La valeur par défaut est true.</p> <p>catenateWords (type : booléen) : provoque la concaténation maximale des parties de mots ; par exemple « Azure-Search » devient « AzureSearch ». La valeur par défaut est false.</p> <p>catenateNumbers (type : booléen) : provoque la concaténation maximale des parties numériques ; par exemple « 1-2 » devient « 12 ». La valeur par défaut est false.</p> <p>catenateAll (type : booléen) : provoque la concaténation de tous les éléments qui sont des sous-mots ; par exemple « Azure-Search-1 » devient « AzureSearch1 ». La valeur par défaut est false.</p> <p>splitOnCaseChange (type : booléen) : si la valeur est true, fractionne les mots au niveau des changements de casse ; par exemple « AzureSearch » devient « Azure » « Search ». La valeur par défaut est true.</p> <p>preserveOriginal : fait que les mots d'origine sont conservés et ajoutés à la liste des sous-mots. La valeur par</p>

NOM_FILTER_JETONS	TOKEN_FILTER_TYPE	défaut est false. DESCRIPTION ET OPTIONS
		<p>splitOnNumerics (type : booléen) : si la valeur est true, fractionne au niveau des nombres ; par exemple « Azure1Search » devient « Azure », « 1 » « Search ». La valeur par défaut est true.</p> <p>stemEnglishPossessive (type : booléen) : provoque la suppression du « s » final pour chaque sous-mot. La valeur par défaut est true.</p> <p>protectedWords (type : tableau de chaînes) : jetons à protéger de la délimitation. La valeur par défaut est une liste vide.</p>

<sup>1</sup> Les types de filtres de jetons sont toujours préfixés dans le code par « #Microsoft.Azure.Search » ; ainsi, « ArabicNormalizationTokenFilter » serait spécifié sous la forme « #Microsoft.Azure.Search.ArabicNormalizationTokenFilter ». Nous avons supprimé le préfixe pour réduire la largeur du tableau, mais n'oubliez pas de l'inclure dans votre code.

## Voir aussi

[API REST Recherche cognitive Azure](#)

[Analyseurs dans Recherche cognitive Azure > Exemples](#)

[Créer un index \(API REST Recherche cognitive Azure\)](#)

# Synonymes dans Recherche cognitive Azure

04/10/2020 • 11 minutes to read • [Edit Online](#)

Dans les moteurs de recherche, les synonymes associent des termes équivalents qui élargissent implicitement l'étendue d'une requête, sans que l'utilisateur ait à fournir le terme. Par exemple, si l'on considère le terme « chien » et les associations de synonymes « canin » et « chiot », tous les documents contenant « chien », « canin » ou « chiot » seront pris en compte dans la requête.

Dans Recherche cognitive Azure, l'expansion des synonymes est effectuée au moment de la requête. Vous pouvez ajouter des cartes de synonymes à un service sans interrompre les opérations existantes. Vous pouvez ajouter une propriété **synonymMaps** à une définition de champ sans avoir à reconstruire l'index.

## Créer des synonymes

Le portail ne prend pas en charge la création de synonymes, mais vous pouvez utiliser l'API REST ou le kit SDK .NET. Pour prendre en main REST, nous vous recommandons d'[utiliser Postman](#) et la formulation de requêtes à l'aide de cette API : [Créer des cartes de synonymes](#). Pour les développeurs C#, vous pouvez commencer par [Ajouter des synonymes dans Recherche cognitive Azure à l'aide de C#](#).

En outre, si vous utilisez des [clés gérée par le client](#) pour le chiffrement au repos côté service, vous pouvez appliquer cette protection au contenu de votre carte de synonymes.

## Utiliser des synonymes

Dans Recherche cognitive Azure, la prise en charge des synonymes repose sur des cartes de synonymes que vous définissez et chargez sur votre service. Ces cartes constituent des ressources indépendantes (telles que des index ou des sources de données) et peuvent être utilisées par n'importe quel champ de recherche dans un index de votre service de recherche.

Les cartes de synonymes et les index sont gérés de manière indépendante. Une fois que vous définissez une carte de synonymes et la téléchargez sur votre service, vous pouvez activer la fonctionnalité de synonyme sur un champ en ajoutant une nouvelle propriété nommée **synonymMaps** dans la définition du champ. La création, la mise à jour et la suppression d'une carte de synonymes constituent des opérations qui s'appliquent au document entier, ce qui signifie que vous ne pouvez pas créer, mettre à jour ou supprimer des parties de la carte de synonyme de façon incrémentielle. Même la mise à jour d'une seule entrée nécessite un rechargeement.

L'incorporation de synonymes dans votre application de recherche est un processus en deux étapes :

1. Ajoutez une carte de synonymes à votre service de recherche via les API ci-dessous.
2. Configurez un champ pouvant faire l'objet d'une recherche pour utiliser la carte de synonymes dans la définition d'index.

Vous pouvez créer plusieurs cartes de synonymes pour votre application de recherche (par exemple, par langue si votre application prend en charge une base de clients multilingue). Actuellement, un champ peut uniquement utiliser une de ces deux méthodes. Vous pouvez mettre à jour la propriété **synonymMaps** d'un champ à tout moment.

### API de ressources **SynonymMaps**

[Ajoutez ou mettez à jour une carte de synonymes dans votre service à l'aide d'une requête POST ou PUT](#).

Les cartes de synonymes sont téléchargées sur le service via une requête POST ou PUT. Chaque règle doit être délimitée par le caractère de nouvelle ligne ('\n'). Vous pouvez définir jusqu'à 5 000 règles par carte de

synonymes dans un service gratuit et 20 000 règles par carte dans toutes les autres références SKU. Chaque règle peut avoir jusqu'à 20 extensions.

Les cartes de synonymes doivent être au format Apache Solr décrit ci-dessous. Si vous disposez d'un dictionnaire de synonymes existant dans un autre format et souhaitez l'utiliser directement, faites-le-nous savoir sur [UserVoice](#).

Vous pouvez créer une nouvelle carte de synonymes à l'aide d'une requête HTTP POST, comme dans l'exemple suivant :

```
POST https://[servicename].search.windows.net/synonymmaps?api-version=2020-06-30
api-key: [admin key]

{
    "name": "mysynonymmap",
    "format": "solr",
    "synonyms": "
        USA, United States, United States of America\n
        Washington, Wash., WA => WA\n"
}
```

Vous pouvez également utiliser une requête PUT en spécifiant le nom de la carte de synonymes sur l'URI. Si la carte de synonymes n'existe pas, elle est créée.

```
PUT https://[servicename].search.windows.net/synonymmaps/mysynonymmap?api-version=2020-06-30
api-key: [admin key]

{
    "format": "solr",
    "synonyms": "
        USA, United States, United States of America\n
        Washington, Wash., WA => WA\n"
}
```

#### Format de synonymes Apache Solr

Le format Solr prend en charge les cartes de synonymes équivalentes et explicites. Les règles de mappage respectent la spécification de filtre de synonyme open source d'Apache Solr, décrite dans ce document : [SynonymFilter](#). Voici un exemple de règle pour des synonymes équivalents.

```
USA, United States, United States of America
```

Avec la règle ci-dessus, une requête de recherche « USA » s'étendra à « USA » OR « United States » OR « United States of America ».

Un mappage explicite est indiqué par une flèche « => ». Lorsqu'elle spécifiée, une séquence de termes d'une requête de recherche qui correspond à la partie gauche de « => » est remplacée par les alternatives sur la partie droite. Étant donné la règle ci-dessous, les requêtes de recherche « Washington », « Wash. » ou « WA » seront réécrites « WA ». Le mappage explicite s'applique dans le sens spécifié uniquement et ne réécrit pas la requête « WA » en « Washington » dans ce cas.

```
Washington, Wash., WA => WA
```

Si vous devez définir des synonymes qui contiennent des virgules, vous pouvez échapper celles-ci à l'aide d'une barre oblique inverse, comme dans l'exemple suivant :

```
WA\, USA, WA, Washington
```

Étant donné que la barre oblique inverse est elle-même un caractère spécial dans d'autres langages tels que JSON et C#, vous devrez probablement effectuer un double échappement. Par exemple, le code JSON envoyé à l'API REST pour la carte de synonymes ci-dessus ressemble à ceci :

```
{
    "format": "solr",
    "synonyms": "WA\\, USA, WA, Washington"
}
```

#### Répertorier les cartes de synonymes de votre service.

```
GET https://[servicename].search.windows.net/synonymmaps?api-version=2020-06-30
api-key: [admin key]
```

#### Obtenir une carte de synonymes pour votre service.

```
GET https://[servicename].search.windows.net/synonymmaps/mysynonymmap?api-version=2020-06-30
api-key: [admin key]
```

#### Supprimer une carte de synonymes de votre service.

```
DELETE https://[servicename].search.windows.net/synonymmaps/mysynonymmap?api-version=2020-06-30
api-key: [admin key]
```

#### Configurez un champ pouvant faire l'objet d'une recherche pour utiliser la carte de synonymes dans la définition d'index.

Une nouvelle propriété de champ **synonymMaps** permet de spécifier une carte de synonymes à utiliser pour un champ pouvant faire l'objet d'une recherche. Les cartes de synonymes sont des ressources de niveau de service et peuvent être référencées selon n'importe quel champ d'index dans le service.

```

POST https://[servicename].search.windows.net/indexes?api-version=2020-06-30
api-key: [admin key]

{
  "name": "myindex",
  "fields": [
    {
      "name": "id",
      "type": "Edm.String",
      "key": true
    },
    {
      "name": "name",
      "type": "Edm.String",
      "searchable": true,
      "analyzer": "en.lucene",
      "synonymMaps": [
        "mysynonymmap"
      ]
    },
    {
      "name": "name_jp",
      "type": "Edm.String",
      "searchable": true,
      "analyzer": "ja.microsoft",
      "synonymMaps": [
        "japanesesynonymmap"
      ]
    }
  ]
}

```

**synonymMaps** peut être spécifié pour les champs pouvant faire l'objet d'une recherche de type « Edm.String » ou « Collection(Edm.String) ».

#### NOTE

Vous ne pouvez utiliser qu'une carte de synonymes par champ. Si vous souhaitez utiliser plusieurs cartes de synonymes, faites-le-nous savoir sur [UserVoice](#).

## Impact des synonymes sur les autres fonctionnalités de recherche

La fonctionnalité de synonymes réécrit la requête d'origine avec des synonymes utilisant l'opérateur OR. Pour cette raison, la mise en surbrillance des correspondances et les profils de score traitent les synonymes et le terme d'origine comme équivalents.

La fonctionnalité de synonymes s'applique aux requêtes de recherche et non aux filtres ou facettes. De même, les suggestions sont basées uniquement sur le terme d'origine : les correspondances de synonymes n'apparaissent pas dans la réponse.

Les extensions de synonymes ne s'appliquent pas aux termes de recherche génériques : les préfixes, correspondances partielles et les expressions régulières ne sont pas étendus.

Si vous devez faire une requête unique qui applique l'expansion de synonymes et des recherches avec des caractères génériques, des expressions régulières ou une correspondance approximative, vous pouvez combiner les requêtes avec la syntaxe d'OR. Par exemple, pour combiner des synonymes avec des caractères génériques pour une syntaxe de requête simple, le terme serait `<query> | <query>*`.

Si vous avez un index existant dans un environnement de déploiement (non production), faites des essais avec un petit dictionnaire pour voir comment l'ajout de synonymes modifie l'expérience de recherche, notamment son

impact sur les profils de score, la mise en surbrillance des correspondances et les suggestions.

## Étapes suivantes

[Créer une carte de synonymes](#)

# Exemple : Ajouter des synonymes pour le service Recherche cognitive Azure en C#

04/10/2020 • 9 minutes to read • [Edit Online](#)

Les synonymes développent une requête en faisant correspondre les termes considérés comme sémantiquement équivalents à l'expression entrée. Par exemple, vous souhaiterez peut-être que le terme « voiture » vous permette de trouver des documents contenant les mots « automobile » ou « véhicule ».

Dans la Recherche cognitive Azure, les synonymes sont définis dans une *carte de synonymes*, par le biais des *règles de mappage* qui associent des termes équivalents. Cet exemple décrit les étapes essentielles pour l'ajout et l'utilisation de synonymes avec un index existant. Vous allez apprendre à effectuer les actions suivantes :

- Créer une carte de synonymes en utilisant la classe [SynonymMap](#).
- Définir la propriété [SynonymMaps](#) sur des champs qui doivent prendre en charge l'extension de requête par le biais de synonymes.

Vous pouvez interroger un champ acceptant les synonymes comme vous le faites habituellement. Aucune syntaxe de requête supplémentaire n'est requise pour accéder aux synonymes.

Vous pouvez créer plusieurs cartes de synonymes, les valider en tant que ressources du service disponible pour tout index, et ensuite référencer ceux que vous souhaitez utiliser au niveau du champ. Au moment de la requête, en plus de la recherche dans un index, la Recherche cognitive Azure effectue une recherche dans une carte de synonymes, si une telle carte est spécifiée dans les champs utilisés dans la requête.

## NOTE

Les synonymes peuvent être créés par programmation, mais pas dans le portail. Si la prise en charge des synonymes par le portail Azure peut vous être utile, donnez-nous votre avis sur [UserVoice](#)

## Prérequis

La configuration requise du didacticiel est la suivante :

- [Visual Studio](#)
- [Service Recherche cognitive Azure](#)
- [Bibliothèque .NET Microsoft.Azure.Search](#)
- [Guide pratique pour utiliser la Recherche cognitive Azure à partir d'une application .NET](#)

## Vue d'ensemble

Les requêtes avant et après présentent la valeur des synonymes. Dans cet exemple, vous utilisez un exemple d'application qui exécute des requêtes et retourne des résultats sur un index d'exemples. L'exemple d'application crée un petit index nommé « hotels » comprenant deux documents. L'application exécute des requêtes de recherche à l'aide de termes et d'expressions qui n'apparaissent pas dans l'index, active la fonctionnalité de synonymes, puis lance les mêmes recherches une nouvelle fois. Le code ci-dessous montre le flux global.

```

static void Main(string[] args)
{
    SearchServiceClient serviceClient = CreateSearchServiceClient();

    Console.WriteLine("{0}", "Cleaning up resources...\n");
    CleanupResources(serviceClient);

    Console.WriteLine("{0}", "Creating index...\n");
    CreateHotelsIndex(serviceClient);

    ISearchIndexClient indexClient = serviceClient.Indexes.GetClient("hotels");

    Console.WriteLine("{0}", "Uploading documents...\n");
    UploadDocuments(indexClient);

    ISearchIndexClient indexClientForQueries = CreateSearchIndexClient();

    RunQueriesWithNonExistentTermsInIndex(indexClientForQueries);

    Console.WriteLine("{0}", "Adding synonyms...\n");
    UploadSynonyms(serviceClient);
    EnableSynonymsInHotelsIndex(serviceClient);
    Thread.Sleep(10000); // Wait for the changes to propagate

    RunQueriesWithNonExistentTermsInIndex(indexClientForQueries);

    Console.WriteLine("{0}", "Complete. Press any key to end application...\n");

    Console.ReadKey();
}

```

Les étapes pour créer et remplir l'index des exemples sont expliquées dans le [Guide pratique pour utiliser la Recherche cognitive Azure à partir d'une application .NET](#).

## Requêtes « avant »

Dans `RunQueriesWithNonExistentTermsInIndex`, émettez des requêtes de recherche avec « five star », « internet » et « economy AND hotel ».

```

Console.WriteLine("Search the entire index for the phrase \"five star\":\n");
results = indexClient.Documents.Search<Hotel>("\"five star\"", parameters);
WriteDocuments(results);

Console.WriteLine("Search the entire index for the term 'internet':\n");
results = indexClient.Documents.Search<Hotel>("internet", parameters);
WriteDocuments(results);

Console.WriteLine("Search the entire index for the terms 'economy' AND 'hotel':\n");
results = indexClient.Documents.Search<Hotel>("economy AND hotel", parameters);
WriteDocuments(results);

```

Aucun des deux documents indexés ne contient les termes, nous avons donc la sortie suivante à partir du premier `RunQueriesWithNonExistentTermsInIndex`.

```
Search the entire index for the phrase "five star":
```

```
no document matched
```

```
Search the entire index for the term 'internet':
```

```
no document matched
```

```
Search the entire index for the terms 'economy' AND 'hotel':
```

```
no document matched
```

## Activation des synonymes

L'activation des synonymes est un processus en deux étapes. Tout d'abord nous définissons et chargeons les règles de synonymes, puis nous configurons les champs pour les utiliser. Le processus est décrit dans [UploadSynonyms](#) et [EnableSynonymsInHotelsIndex](#).

1. Ajoutez une carte de synonymes à votre service de recherche. Dans [UploadSynonyms](#), nous définissons quatre règles de notre carte de synonymes « desc-synonymmap » et effectuons le téléchargement vers le service.

```
var synonymMap = new SynonymMap()
{
    Name = "desc-synonymmap",
    Format = "solr",
    Synonyms = "hotel, motel\n
                internet,wifi\n
                five star=>luxury\n
                economy,inexpensive=>budget"
};

serviceClient.SynonymMaps.CreateOrUpdate(synonymMap);
```

Une carte de synonymes doit être conforme au format [solr](#) standard Open Source. Le format est expliqué dans [Synonymes dans la Recherche cognitive Azure](#) sous la section [Apache Solr synonym format](#).

2. Configurez les champs pouvant faire l'objet d'une recherche pour utiliser la carte de synonymes dans la définition d'index. Dans [EnableSynonymsInHotelsIndex](#), nous activons les synonymes sur deux champs [category](#) et [tags](#) en affectant à la propriété [synonymMaps](#) le nom de la carte de synonymes qui vient d'être téléchargée.

```
Index index = serviceClient.Indexes.Get("hotels");
index.Fields.First(f => f.Name == "category").SynonymMaps = new[] { "desc-synonymmap" };
index.Fields.First(f => f.Name == "tags").SynonymMaps = new[] { "desc-synonymmap" };

serviceClient.Indexes.CreateOrUpdate(index);
```

Lorsque vous ajoutez une carte de synonymes, les reconstructions d'index ne sont pas requises. Vous pouvez ajouter une carte de synonymes à votre service, puis modifier les définitions de champ existantes dans n'importe quel index pour utiliser la nouvelle carte de synonymes. L'ajout de nouveaux attributs n'a aucun impact sur la disponibilité de l'index. Il en va de même pour la désactivation de synonymes pour un champ. Vous pouvez simplement affecter à la propriété [synonymMaps](#) une liste vide.

```
index.Fields.First(f => f.Name == "category").SynonymMaps = new List<string>();
```

## Requêtes « après »

Une fois que la carte de synonymes est téléchargée et l'index mis à jour pour utiliser la carte de synonymes, le deuxième appel `RunQueriesWithNonExistentTermsInIndex` affiche les éléments suivants :

```
Search the entire index for the phrase "five star":
```

```
Name: Fancy Stay Category: Luxury Tags: [pool, view, wifi, concierge]
```

```
Search the entire index for the term 'internet':
```

```
Name: Fancy Stay Category: Luxury Tags: [pool, view, wifi, concierge]
```

```
Search the entire index for the terms 'economy' AND 'hotel':
```

```
Name: Roach Motel Category: Budget Tags: [motel, budget]
```

La première requête trouve le document à partir de la règle `five star=>luxury`. La deuxième requête étend la recherche à l'aide de `internet,wifi` et la troisième avec à la fois `hotel, motel` et `economy,inexpensive=>budget` pour trouver les documents en correspondance.

L'ajout de synonymes modifie complètement l'expérience de recherche. Dans cet exemple, les requêtes d'origine n'ont pas pu retourner de résultats significatifs même si les documents dans notre index étaient pertinents. En activant les synonymes, nous pouvons développer un index pour inclure les termes communément utilisés, sans modification de données sous-jacentes dans l'index.

## Code source de l'exemple d'application

Vous trouverez le code source complet de l'exemple d'application utilisé dans cette procédure sur [GitHub](#).

## Nettoyer les ressources

Le moyen le plus rapide de procéder à un nettoyage après un exemple consiste à supprimer le groupe de ressources contenant le service Recherche cognitive Azure. Vous pouvez maintenant supprimer le groupe de ressources pour supprimer définitivement tout ce qu'il contient. Dans le portail, le nom du groupe de ressources figure dans la page Vue d'ensemble du service Recherche cognitive Azure.

## Étapes suivantes

Cet exemple a présenté la fonctionnalité de synonymes en code C# pour créer et publier des règles de mappage, puis appeler la carte de synonymes pour une requête. Vous trouverez des informations supplémentaires dans la documentation de référence du [Kit de développement logiciel \(SDK\) .NET](#) et de l'[API REST](#).

[Guide pratique pour utiliser des synonymes dans la Recherche cognitive Azure](#)

# Modélisation de types de données complexes dans Recherche cognitive Azure

04/10/2020 • 16 minutes to read • [Edit Online](#)

Les jeux de données externes utilisés pour remplir un index Recherche cognitive Azure peuvent avoir différentes formes. Ils incluent parfois des sous-structures hiérarchiques ou imbriquées. Des exemples incluent les adresses multiples pour un même client, les couleurs et les tailles multiples pour une même référence, les auteurs multiples pour un même livre, etc. En termes de modélisation, ces structures peuvent être désignées sous le nom de types de données *complexes*, *composées*, *composites* or *agrégées*. Le terme utilisé par Recherche cognitive Azure pour ce concept est **type complexe**. Dans Recherche cognitive Azure, les types complexes sont modélisés à l'aide de **champs complexes**. Un champ complexe est un champ qui contient des enfants (sous-champs) qui peuvent être des données de n'importe quel type, notamment d'autres types complexes. Ceci fonctionne d'une manière similaire aux types de données structurées dans un langage de programmation.

Les champs complexes représentent un objet unique dans le document ou un tableau d'objets, selon le type de données. Les champs de type `Edm.ComplexType` représentent des objets uniques, alors que des champs de type `Collection(Edm.ComplexType)` représentent des tableaux d'objets.

Recherche cognitive Azure prend nativement en charge les types et les collections complexes. Ces types vous permettent de modéliser presque n'importe quelle structure JSON dans un index Recherche cognitive Azure. Dans les versions précédentes d'API de Recherche cognitive Azure, seuls les jeux de lignes aplatis ont pu être importés. Dans la version la plus récente, votre index peut mieux correspondre aux données sources. En d'autres termes, si vos données sources contiennent des types complexes, votre index peut également contenir des types complexes.

Pour commencer, nous vous recommandons le [jeu de données d'hôtels](#), que vous pouvez charger dans l'Assistant **Importer des données** du portail Azure. L'Assistant détecte les types complexes dans la source et suggère un schéma d'index basé sur les structures détectées.

## NOTE

La prise en charge des types complexes a commencé à être généralement disponible dans `api-version=2019-05-06`.

Si votre solution de recherche est basée sur des solutions de contournement antérieures de jeux de données aplatis d'une collection, vous devez modifier votre index pour inclure des types complexes pris en charge dans la dernière version d'API. Pour plus d'informations sur la mise à niveau des versions d'API, consultez [Mettre à niveau vers la dernière version de l'API REST](#) ou [Mettre à niveau vers la dernière version du kit de développement logiciel \(SDK\) .NET](#).

## Exemple de structure complexe

Le document JSON suivant est composé de champs simples et de champs complexes. Les champs complexes, tels que `Address` et `Rooms`, ont des sous-champs. `Address` contient un seul jeu de valeurs pour ces sous-champs, puisqu'il s'agit d'un objet unique dans le document. En revanche, `Rooms` a plusieurs ensembles de valeurs pour ses sous-champs, soit un pour chaque objet dans la collection.

```
{
  "HotelId": "1",
  "HotelName": "Secret Point Motel",
  "Description": " Ideally located on the main commercial artery of the city in the heart of New York.",
  "Address": {
    "StreetAddress": "677 5th Ave",
    "City": "New York",
    "StateProvince": "NY"
  },
  "Rooms": [
    {
      "Description": "Budget Room, 1 Queen Bed (cityside)",
      "Type": "Budget Room",
      "BaseRate": 96.99
    },
    {
      "Description": "Deluxe Room, 2 Double Beds (City View)",
      "Type": "Deluxe Room",
      "BaseRate": 150.99
    }
  ]
}
```

## Création de champs complexes

Comme avec n'importe quelle définition d'index, vous pouvez utiliser le portail, l'[API REST](#), ou le kit de développement logiciel ([SDK](#)) .NET pour créer un schéma incluant des types complexes.

L'exemple suivant montre un schéma d'index JSON avec des champs simples, des collections et des types complexes. Notez qu'au sein d'un type complexe, chaque sous-champ a un type et peut avoir des attributs, comme c'est le cas pour les champs de niveau supérieur. Le schéma correspond à l'exemple de données ci-dessus. `Address` est un champ complexe qui n'est pas une collection (un hôtel a une adresse). `Rooms` est un champ de collection complexe (un hôtel a plusieurs chambres).

```
{
  "name": "hotels",
  "fields": [
    { "name": "HotelId", "type": "Edm.String", "key": true, "filterable": true },
    { "name": "HotelName", "type": "Edm.String", "searchable": true, "filterable": false },
    { "name": "Description", "type": "Edm.String", "searchable": true, "analyzer": "en.lucene" },
    { "name": "Address", "type": "Edm.ComplexType",
      "fields": [
        { "name": "StreetAddress", "type": "Edm.String", "filterable": false, "sortable": false,
          "facetable": false, "searchable": true },
        { "name": "City", "type": "Edm.String", "searchable": true, "filterable": true, "sortable": true,
          "facetable": true },
        { "name": "StateProvince", "type": "Edm.String", "searchable": true, "filterable": true,
          "sortable": true, "facetable": true }
      ]
    },
    { "name": "Rooms", "type": "Collection(Edm.ComplexType)",
      "fields": [
        { "name": "Description", "type": "Edm.String", "searchable": true, "analyzer": "en.lucene" },
        { "name": "Type", "type": "Edm.String", "searchable": true },
        { "name": "BaseRate", "type": "Edm.Double", "filterable": true, "facetable": true }
      ]
    }
  ]
}
```

## Mise à jour de champs complexes

Toutes les [règles de réindexation](#) qui s'appliquent à des champs s'appliquent en général toujours aux champs complexes. Dans le cadre du rappel de quelques règles principales, notons que, contrairement à la plupart des modifications, l'ajout d'un champ ne nécessite pas une reconstruction de l'index.

### Mises à jour structurelles de la définition

Vous pouvez ajouter des sous-champs à un champ complexe à tout moment sans qu'une reconstruction d'index soit nécessaire. Par exemple, l'ajout de « ZipCode » à `Address` ou d'« Amenities » (infrastructures) à `Rooms` est autorisé, tout comme l'ajout d'un champ de niveau supérieur à un index. Les documents existants ont une valeur Null pour les nouveaux champs jusqu'à ce que vous remplissiez explicitement ces champs en mettant à jour de vos données.

Notez qu'au sein d'un type complexe, chaque sous-champ a un type et peut avoir des attributs, comme c'est le cas pour les champs de niveau supérieur

### Mises à jour des données

La mise à jour de documents existants dans un index avec l'action `upload` fonctionne de la même façon pour les champs complexes et simples : tous les champs sont remplacés. Toutefois, `merge` (ou `mergeOrUpload` lorsqu'il est appliqué à un document existant) ne fonctionne pas de la même façon dans tous les champs. Plus précisément, `merge` ne prend pas en charge la fusion d'éléments dans une collection. Cette limitation existe pour les collections de types primitifs et les collections complexes. Pour mettre à jour une collection, vous devez récupérer la valeur de la collection complète, apporter des modifications, puis inclure la nouvelle collection dans la requête de l'API d'index.

## Recherche de champs complexes

Les expressions de recherche de forme libre fonctionnent comme prévu avec des types complexes. Si n'importe quel champ de recherche ou sous-champ n'importe où dans un document correspond, le document proprement est une correspondance.

Les requêtes sont plus nuancées lorsque vous avez plusieurs termes et opérateurs, et certains termes ont des noms de champs spécifiés, comme cela est possible avec la [syntaxe Lucene](#). Par exemple, cette requête essaie de faire correspondre deux termes, « Portland » et « OR » à deux sous-champs du champ adresse :

```
search=Address/City:Portland AND Address/State:OR
```

Des requêtes de ce type sont *sans corrélation* pour la recherche en texte intégral, contrairement aux filtres. Dans les filtres, les requêtes relatives aux sous-champs d'une collection complexe sont corrélées à l'aide de variables de portée dans `any` ou `all`. La requête Lucene ci-dessus retourne des documents contenant « Portland, Maine » et « Portland, Oregon », ainsi que d'autres villes d'Oregon. C'est dû au fait que chaque clause s'applique à toutes les valeurs de son champ dans le document entier. Il n'existe donc pas de concept de « sous-élément actuel ». Pour plus d'informations à ce sujet, consultez [Présentation de filtres de collection OData dans Recherche cognitive Azure](#).

## Sélection de champs complexes

Le paramètre `$select` permet de choisir quels champs retourner dans les résultats de la recherche. Pour utiliser ce paramètre afin de sélectionner des sous-champs spécifiques d'un champ complexe, incluez le champ parent et le sous-champ séparés par une barre oblique (`/`).

```
$select=HotelName, Address/City, Rooms/BaseRate
```

Les champs doivent être marqués comme récupérables dans l'index si vous souhaitez les afficher dans les résultats de la recherche. Seuls les champs marqués comme récupérable peuvent être utilisés dans une

instruction `$select`.

## Filtrer et trier des champs complexes et activer des facettes pour les champs complexes

La même [syntaxe de chemin OData](#) utilisée pour le filtrage et les recherches par champ peut également être utilisée pour activer des facettes, trier et sélectionner dans le cadre d'une requête de recherche. Pour les types complexes, des règles qui définissent quels sous-champs peuvent être marqués triables ou dotés de facettes activées s'appliquent. Pour plus d'informations sur ces règles, consultez la [référence Créez une API d'index](#).

### Activation de facettes pour les sous-champs

Des facettes peuvent être activées pour n'importe quel sous-champ, sauf s'il est de type `Edm.GeographyPoint` ou `Collection(Edm.GeographyPoint)`.

Le nombre de documents retournés dans les résultats des facettes activées est calculé pour le document parent (un hôtel), pas pour les sous-documents dans une collection complexe (des chambres). Par exemple, supposez qu'un hôtel a 20 suites. Étant donné ce paramètre de facette `facet=Rooms/Type`, le nombre de facettes sera de un pour l'hôtel, pas de 20 pour les chambres.

### Tri de champs complexes

Les opérations de tri s'appliquent aux documents (hôtels) et non aux sous-documents (chambres). Lorsque vous avez une collection de type complexe comme des chambres, il est important de savoir que vous ne pouvez pas du tout trier les chambres. En fait, vous ne pouvez trier aucune collection.

Les opérations de tri fonctionnent lorsque les champs ont une valeur unique pour chaque document, que le champ soit un champ simple ou un sous-champ dans un type complexe. Par exemple, `Address/City` est autorisé à être trié, car il n'y a qu'une seule adresse par hôtel. `$orderby=Address/City` triera donc les hôtels par ville.

### Filtrage sur des champs complexes

Vous pouvez faire référence à des sous-champs d'un champ complexe dans une expression de filtre. Utilisez simplement la même [syntaxe de chemin OData](#) qui est utilisé pour l'activation de facettes, le tri et la sélection de champs. Par exemple, le filtre suivant retourne tous les hôtels au Canada :

```
$filter=Address/Country eq 'Canada'
```

Pour filtrer sur un champ de collection complexe, vous pouvez utiliser une [expression lambda avec les opérateurs](#) `any` et `all`. Dans ce cas, la **variable de portée** de l'expression lambda est un objet avec des sous-champs. Vous pouvez consulter ces sous-champs avec la syntaxe de chemin OData standard. Par exemple, le filtre suivant retourne tous les hôtels avec au moins une chambre de luxe et toutes les chambres non-fumeurs :

```
$filter=Rooms/any(room: room/Type eq 'Deluxe Room') and Rooms/all(room: not room/SmokingAllowed)
```

Comme avec les champs simples de niveau supérieur, les sous-champs simples de champs complexe ne peuvent être inclus dans des filtres que si leur attribut **filtrable** est défini sur `true` dans la définition d'index. Pour plus d'informations, consultez la [référence Créez une API d'index](#).

## Étapes suivantes

Essayez le [jeu de données des hôtels](#) dans l'**Assistant Importer des données**. Vous avez besoin des informations de connexion de Cosmos DB fournies dans le fichier *Lisez-moi* pour accéder aux données.

Lorsque vous avez ces informations, votre première étape dans l'**Assistant** est de créer une nouvelle source de

données Azure Cosmos DB. Plus loin dans l'Assistant, lorsque vous accédez à la page d'index cible, vous voyez un index avec des types complexes. Créez et chargez cet index, puis exécutez des requêtes pour comprendre la nouvelle structure.

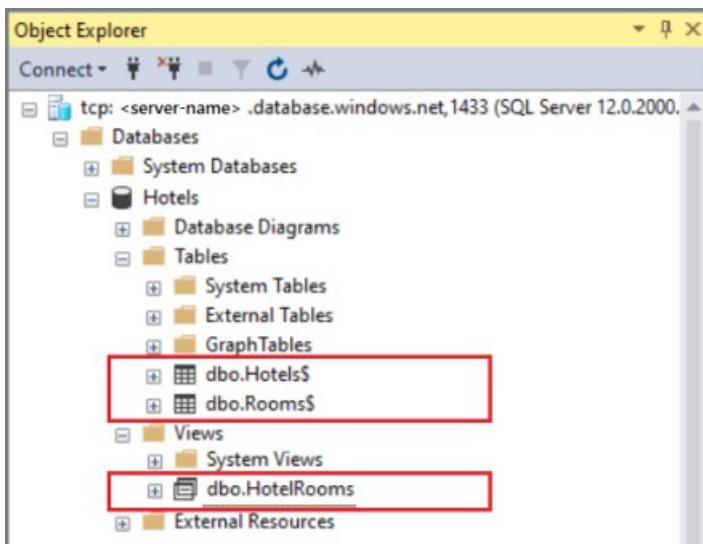
[Démarrage rapide : Assistant du portail pour l'importation, l'indexation et les requêtes](#)

# Comment modéliser des données relationnelles SQL pour l'importation et l'indexation dans la Recherche cognitive Azure

04/10/2020 • 10 minutes to read • [Edit Online](#)

La Recherche cognitive Azure accepte un ensemble de lignes plat comme entrée dans le [pipeline d'indexation](#). Si vos données sources proviennent de tables jointes dans une base de données relationnelle SQL Server, lisez cet article pour savoir comment construire le jeu de résultats et comment modéliser une relation parent-enfant dans un index Recherche cognitive Azure.

Pour illustrer nos explications, nous faisons référence à une base de données Hotels fictive dont le contenu est tiré de ces [données de démonstration](#). Supposons que la base de données se compose d'une table Hotels\$ listant 50 hôtels et d'une table Rooms\$ contenant les détails de 750 chambres au total (comme le type, l'équipement et le prix de chacune d'elles). Il y a une relation un-à-plusieurs entre les tables. Dans notre approche, nous utilisons une vue pour fournir la requête qui retourne 50 lignes (soit une ligne par hôtel), avec les détails des chambres associées incorporés dans chaque ligne.



## Le problème des données dénormalisées

L'un des problèmes que pose l'utilisation de relations un-à-plusieurs vient des requêtes standard créées sur des tables jointes qui retournent des données dénormalisées, un format qui n'est pas approprié dans un scénario Recherche cognitive Azure. Prenons l'exemple suivant qui joint les hôtels et les chambres.

```
SELECT * FROM Hotels$  
INNER JOIN Rooms$  
ON Rooms$.HotelID = Hotels$.HotelID
```

Les résultats de cette requête retournent tous les champs de la table Hotels, suivis de tous les champs de la table Rooms, avec des informations préliminaires sur l'hôtel qui sont répétées pour chaque chambre.

Fields from Hotels\$				Fields from Rooms\$					
HotelID	HotelName	Description	State	HotelID	Description	Type	BaseRate	BedOptions	
181	2	Twin Dome Motel	The hotel is situated in a nineteenth century plaza...	FL	2	Standard Room, 1 Queen Bed (City View)	Standard Room	121.99	1 Queen Bed
182	2	Twin Dome Motel	The hotel is situated in a nineteenth century plaza...	FL	2	Budget Room, 1 King Bed (Waterfront View)	Budget Room	88.99	1 King Bed
183	2	Twin Dome Motel	The hotel is situated in a nineteenth century plaza...	FL	2	Standard Room, 2 Double Beds (Cityside)	Standard Room	127.99	2 Double Beds
184	2	Twin Dome Motel	The hotel is situated in a nineteenth century plaza...	FL	2	Budget Room, 2 Double Beds (Cityside)	Budget Room	96.99	2 Double Beds
185	2	Twin Dome Motel	The hotel is situated in a nineteenth century plaza...	FL	2	Budget Room, 1 Queen Bed (Mountain View)	Budget Room	63.99	1 Queen Bed
186	2	Twin Dome Motel	The hotel is situated in a nineteenth century plaza...	FL	2	Standard Room, 1 Queen Bed (Mountain View)	Standard Room	124.99	1 Queen Bed
187	2	Twin Dome Motel	The hotel is situated in a nineteenth century plaza...	FL	2	Standard Room, 1 Queen Bed (Cityside)	Standard Room	117.99	1 Queen Bed

Cette requête semble correcte de prime abord (elle fournit bien toutes les données dans un ensemble de lignes plat), mais en réalité, elle ne donne pas la structure de document appropriée pour l'expérience de recherche attendue. Durant l'indexation, la Recherche cognitive Azure crée un document de recherche pour chaque ligne ingérée. Si vos documents de recherche présentaient une structure similaire aux résultats ci-dessus, vous verriez des doublons, comme ici où nous avons sept documents distincts pour le seul hôtel Twin Dome. Une requête de recherche sur les hôtels en Floride renverrait sept résultats rien que pour l'hôtel Twin Dome, et les autres hôtels pertinents seraient affichés beaucoup plus loin dans les résultats de la recherche.

Pour obtenir l'expérience attendue d'un seul document par hôtel, vous devez fournir un ensemble de lignes à la granularité appropriée, mais avec des informations complètes. Heureusement, vous pouvez y parvenir facilement en adoptant les techniques décrites dans cet article.

## Définir une requête qui retourne une collection JSON incorporée

Pour offrir l'expérience de recherche attendue, votre jeu de données doit contenir une ligne pour chaque document de recherche dans la Recherche cognitive Azure. Dans notre exemple, nous voulons avoir une ligne distincte par hôtel, mais nous souhaitons également que les utilisateurs puissent faire des recherches sur d'autres champs descriptifs des chambres pour trouver les renseignements dont ils ont besoin, comme le prix par nuit, le nombre et la taille des lits ou une vue sur mer, etc.

La solution consiste à capturer les détails des chambres avec une requête JSON imbriquée, puis d'insérer la structure JSON dans un champ d'une vue, comme cela est montré à la deuxième étape.

1. Supposons que vous avez deux tables Hotels\$ et Rooms\$ qui sont jointes sur le champ HotelID. Ces tables contiennent respectivement les détails de 50 hôtels et de leurs 750 chambres.

```

CREATE TABLE [dbo].[Hotels$](
    [HotelID] [nchar](10) NOT NULL,
    [HotelName] [nvarchar](255) NULL,
    [Description] [nvarchar](max) NULL,
    [Description_fr] [nvarchar](max) NULL,
    [Category] [nvarchar](255) NULL,
    [Tags] [nvarchar](255) NULL,
    [ParkingIncluded] [float] NULL,
    [SmokingAllowed] [float] NULL,
    [LastRenovationDate] [smalldatetime] NULL,
    [Rating] [float] NULL,
    [StreetAddress] [nvarchar](255) NULL,
    [City] [nvarchar](255) NULL,
    [State] [nvarchar](255) NULL,
    [ZipCode] [nvarchar](255) NULL,
    [GeoCoordinates] [nvarchar](255) NULL
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO

```

```

CREATE TABLE [dbo].[Rooms$](
    [HotelID] [nchar](10) NULL,
    [Description] [nvarchar](255) NULL,
    [Description_fr] [nvarchar](255) NULL,
    [Type] [nvarchar](255) NULL,
    [BaseRate] [float] NULL,
    [BedOptions] [nvarchar](255) NULL,
    [SleepsCount] [float] NULL,
    [SmokingAllowed] [float] NULL,
    [Tags] [nvarchar](255) NULL
) ON [PRIMARY]
GO

```

2. Créez une vue composée de tous les champs de la table parente (`SELECT * from dbo.Hotels$`) et d'un nouveau champ *Rooms* qui contient la sortie d'une requête imbriquée. La clause **FOR JSON AUTO** spécifiée dans `SELECT * from dbo.Rooms$` structure la sortie au format JSON.

```

CREATE VIEW [dbo].[HotelRooms]
AS
SELECT *, (SELECT *
            FROM dbo.Rooms$
            WHERE dbo.Rooms$.HotelID = dbo.Hotels$.HotelID FOR JSON AUTO) AS Rooms
FROM dbo.Hotels$
GO

```

La capture d'écran ci-dessous montre la vue obtenue, avec le champ *Rooms* nvarchar tout en bas. Le champ *Rooms* existe uniquement dans la vue HotelRooms.

The screenshot shows the 'dbo.HotelRooms' table structure. The 'Rooms' column is highlighted with a red box.

```

    [+] Views
      [+] System Views
      [+] dbo.HotelRooms
        [+] Columns
          [+] HotelID (nchar(10), not null)
          [+] HotelName (nvarchar(255), null)
          [+] Description (nvarchar(max), null)
          [+] Description_fr (nvarchar(max), null)
          [+] Category (nvarchar(255), null)
          [+] Tags (nvarchar(255), null)
          [+] SmokingIncluded (float, null)
          [+] LastRenovationDate (smalldatetime, null)
          [+] Rating (float, null)
          [+] StreetAddress (nvarchar(255), null)
          [+] City (nvarchar(255), null)
          [+] State (nvarchar(255), null)
          [+] ZipCode (nvarchar(255), null)
          [+] GeoCoordinates (nvarchar(255), null)
          [+] Rooms (nvarchar(max), null)
  
```

3. Exécutez `SELECT * FROM dbo.HotelRooms` pour récupérer l'ensemble de lignes. Cette requête retourne 50 lignes (soit une par hôtel) ainsi que les détails des chambres associées sous forme de collection JSON.

The screenshot shows the results of the query. The 'Rooms' column is expanded to show its JSON value.

HotelID	HotelName	Des...	De...	Ca...	T...	P...	S...	L...	R...	St...	C...	St...	Zi...	Geo...	Rooms
1	Secret Point Motel	Th...	L...	B...	p...	0	1	1...	3...	6...	N.	NY	1...	[7...	[{"HotelID": "1", "Description": "Budget Room, 1 Queen Bed (Cityside)", "Description_fr": "Chambre standard, 1 lit double (c\u00e2t\u00e9 ville)"}, {"HotelID": "10", "Description": "Suite, 1 King Bed (Amenities)", "Description_fr": "Suite, 1 lit king (amenagements)"}, {"HotelID": "11", "Description": "Deluxe Room, 1 Queen Bed (Waterfront View)", "Description_fr": "Chambre sup\u00e9rieure, 1 lit double (vue sur la baie)"}, {"HotelID": "12", "Description": "Deluxe Room, 1 King Bed (Cityside)", "Description_fr": "Chambre sup\u00e9rieure, 1 lit king (c\u00e2t\u00e9 ville)"}, {"HotelID": "13", "Description": "Standard Room, 1 King Bed (Mountain View)", "Description_fr": "Chambre standard, 1 lit king (vue montagne)"}, {"HotelID": "14", "Description": "Budget Room, 1 King Bed (Cityside)", "Description_fr": "Chambre standard, 1 lit king (c\u00e2t\u00e9 ville)"}, {"HotelID": "15", "Description": "Standard Room, 1 King Bed (Waterfront View)", "Description_fr": "Chambre standard, 1 lit king (vue sur la baie)"}, {"HotelID": "16", "Description": "Suite, 2 Queen Beds (Amenities)", "Description_fr": "Suite, 2 lits doubles (amenagements)"}, {"HotelID": "17", "Description": "Budget Room, 2 Queen Beds (Waterfront View)", "Description_fr": "Chambre standard, 2 lits doubles (vue sur la baie)"}, {"HotelID": "18", "Description": "Standard Room, 1 Queen Bed (Cityside)", "Description_fr": "Chambre standard, 1 lit double (c\u00e2t\u00e9 ville)"}, {"HotelID": "19", "Description": "Deluxe Room, 1 Queen Bed (Waterfront View)", "Description_fr": "Chambre sup\u00e9rieure, 1 lit double (vue sur la baie)"}, {"HotelID": "2", "Description": "Suite, 2 Double Beds (Mountain View)", "Description_fr": "Suite, 2 lits doubles (vue montagne)"}]
2	Countryside Hotel	Sa...	\u00c9...	B...	2...	0	1	1...	2...	6...	D.	NC	2...	[7...	
3	Regal Orb Resort & Spa	Yo...	V...	E...	fr...	1	0	1...	2...	2...	B.	W...	9...	[1...	
4	Winter Panorama Resort	Ne...	R...	R...	I...	0	0	1...	4...	9...	W.	OR	9...	[1...	
5	Historic Lion Resort	Un...	U...	B...	v...	0	1	1...	4...	3...	S.	M...	6...	[9...	
6	Twin Vertex Hotel	Ne...	N...	E...	b...	0	0	1...	4...	1...	D.	TX	7...	[9...	
7	Peaceful Market Hotel & Spa	Bo...	R...	R...	c...	1	0	2...	3...	1...	N.	NY	1...	[7...	
8	Double Sanctuary Resort	5'	5...	R...	v...	0	0	1...	4...	2...	S.	W...	9...	[1...	
9	Antiquity Hotel	Ele...	\u00c9...	B...	r...	0	0	1...	4...	8...	N.	NY	1...	[7...	
10	Oceanside Resort	Ne...	N...	B...	v...	1	1	1...	4...	5...	T.	FL	3...	[8...	
11	Universe Motel	Bo...	R...	S...	r...	0	0	1...	2...	1...	R.	W...	9...	[1...	
12	Twin Dome Motel	Th...	L...	B...	p...	0	1	1...	3...	1...	S.	FL	3...	[8...	

Query executed successfully. | tcp:azs-playground.database... | findable (111) | Hotels | 00:00:00 | 50 rows

Cet ensemble de lignes est maintenant pr\u00eet pour l'importation dans la Recherche cognitive Azure.

#### NOTE

Cette approche suppose que la collection JSON incorpor\u00e9e ne d\u00e9passe pas les [limites de tailles de colonne maximales de SQL Server](#).

## Utiliser une collection complexe pour le c\u00f4t\u00e9 « plusieurs » d'une relation un-\u00e0-plusieurs

Dans la Recherche cognitive Azure, cr\u00e9ez un sch\u00e9ma d'index qui mod\u00e9lise la relation un-\u00e0-plusieurs \u00e0 l'aide d'une requ\u00eate JSON imbriqu\u00e9e. Le jeu de r\u00e9sultats que vous avez cr\u00e9\u00e9 dans la section pr\u00e9c\u00e9dente correspond g\u00e9n\u00e9ralement au sch\u00e9ma d'index fourni ci-dessous (nous avons toutefois enlev\u00e9 certains champs par souci de concision).

L'exemple suivant est similaire \u00e0 l'exemple donn\u00e9 dans [Mod\u00e9lisation de types de donn\u00e9es complexes](#). La structure *Rooms*, qui a \u00e9t\u00e9 le sujet principal de cet article, se trouve dans la collection fields d'un index nomm\u00e9 *hotels*. Cet exemple montre \u00e9galement un type complexe pour *Address*, qui diff\u00e8re de *Rooms* dans la mesure o\u00f9 il est compos\u00e9 d'un ensemble fixe d'\u00e9l\u00e9ments, par opposition au nombre arbitraire d'\u00e9l\u00e9ments multiples autoris\u00e9s dans une collection.

```
{
  "name": "hotels",
  "fields": [
    { "name": "HotelId", "type": "Edm.String", "key": true, "filterable": true },
    { "name": "HotelName", "type": "Edm.String", "searchable": true, "filterable": false },
    { "name": "Description", "type": "Edm.String", "searchable": true, "analyzer": "en.lucene" },
    { "name": "Description_fr", "type": "Edm.String", "searchable": true, "analyzer": "fr.lucene" },
    { "name": "Category", "type": "Edm.String", "searchable": true, "filterable": false },
    { "name": "ParkingIncluded", "type": "Edm.Boolean", "filterable": true, "facetable": true },
    { "name": "Address", "type": "Edm.ComplexType",
      "fields": [
        { "name": "StreetAddress", "type": "Edm.String", "filterable": false, "sortable": false, "facetable": false, "searchable": true },
        { "name": "City", "type": "Edm.String", "searchable": true, "filterable": true, "sortable": true, "facetable": true },
        { "name": "StateProvince", "type": "Edm.String", "searchable": true, "filterable": true, "sortable": true, "facetable": true }
      ]
    },
    { "name": "Rooms", "type": "Collection(Edm.ComplexType)",
      "fields": [
        { "name": "Description", "type": "Edm.String", "searchable": true, "analyzer": "en.lucene" },
        { "name": "Description_fr", "type": "Edm.String", "searchable": true, "analyzer": "fr.lucene" },
        { "name": "Type", "type": "Edm.String", "searchable": true },
        { "name": "BaseRate", "type": "Edm.Double", "filterable": true, "facetable": true },
        { "name": "BedOptions", "type": "Edm.String", "searchable": true, "filterable": true, "facetable": true }
      ],
      "key": "HotelID"
    }
  ]
}
```

Avec le jeu de résultats précédent et le schéma d'index ci-dessus, vous avez tous les composants requis pour réussir l'indexation. Le jeu de données aplati répond aux exigences d'indexation, tout en préservant les informations détaillées nécessaires. Dans l'index de la Recherche cognitive Azure, les résultats de la recherche sont catégorisés par hôtel, tout en conservant le contexte de chacune des chambres et leurs caractéristiques.

## Étapes suivantes

Avec votre propre jeu de données, vous pouvez utiliser l'[Assistant Importer des données](#) pour créer et charger l'index. L'Assistant détecte la collection JSON incorporée, telle que celle contenue dans *Rooms*, et déduit un schéma d'index qui contient une collection de types complexes.

The screenshot shows the 'Import data' step of the Azure Cognitive Search 'Import data' wizard. It displays the configuration for the 'hotels' index. Key settings include:

- Index name:** azuresql-hotels-index
- Key:** HotelID
- Fields:** A table showing field names, types, and indexing properties (RETRIEVABLE, FILTERABLE, SORTABLE, FACETABLE, SEARCHABLE). Most fields are of type Edm.String and are marked as RETRIEVABLE, FILTERABLE, and SEARCHABLE.
- Rooms:** This row is expanded, showing it is a Collection type. It contains two sub-fields: HotelID (Edm.ComplexType) and Description (Edm.String). The HotelID field is also marked as RETRIEVABLE, FILTERABLE, and SEARCHABLE.

Essayez le guide de démarrage rapide suivant pour découvrir les étapes de base de l'Assistant Importer des données.

[Démarrage rapide : Créer un index de recherche à l'aide du portail Azure](#)

# Vue d'ensemble de l'importation des données - Recherche cognitive Azure

04/10/2020 • 14 minutes to read • [Edit Online](#)

Dans Recherche cognitive Azure, les requêtes s'exécutent sur le contenu chargé et enregistré dans un [index de recherche](#). Cet article examine les deux méthodes de base pour remplir un index : *envoyer* (push) les données dans l'index par programme ou pointer un [indexeur Recherche cognitive Azure](#) vers une source de données prise en charge pour *extraire* les données.

Ces deux approches ont pour objectif de charger des données depuis une source de données externe vers un index Recherche cognitive Azure. Recherche cognitive Azure vous permet de créer un index vide, mais ce dernier ne pourra être interrogé qu'après envoi (push) ou extraction des données.

## NOTE

Si l'[enrichissement en IA](#) est une condition de la solution, vous devez utiliser le modèle d'extraction (indexeurs) pour charger un index. Le traitement externe est pris en charge uniquement par le biais de compétences liées à un indexeur.

## Envoyer des données à un index

Le modèle d'envoi (push), utilisé pour envoyer vos données à Recherche cognitive Azure par programme, constitue l'approche la plus flexible. Tout d'abord, il n'y a pas de restrictions sur le type de source de données. Tout jeu de données composé de documents JSON peut être envoyé (push) à un index Recherche cognitive Azure, en supposant que chaque document dans le jeu de données possède des champs mappés sur des champs définis dans votre schéma d'index. En second lieu, il n'y a aucune restriction sur la fréquence d'exécution. Vous pouvez transmettre des modifications à un index aussi souvent que vous le souhaitez. Pour les applications ayant des exigences à très faible latence (par exemple, si vous devez synchroniser les opérations de recherche avec les bases de données d'inventaire dynamiques), le modèle d'émission est la seule option.

Cette approche est plus flexible que le modèle d'extraction, car vous pouvez charger des documents individuellement ou par lots (jusqu'à 1 000 par lot ou 16 Mo, quelle que soit la limite atteinte en premier). Le modèle d'envoi (push) vous permet également de charger des documents dans Recherche cognitive Azure indépendamment de l'emplacement des données.

### Comment envoyer (push) des données à un index Recherche cognitive Azure

Vous pouvez utiliser les API suivantes pour charger un ou plusieurs documents dans un index :

- [Ajout, mise à jour ou suppression de documents \(API REST\)](#)
- [classe indexAction ou classe indexBatch](#)

Il n'existe actuellement aucune prise en charge de l'outil de diffusion de données via le portail.

Pour une présentation de chaque méthode, consultez [Guide de démarrage rapide : Créer un index Recherche cognitive Azure à l'aide de PowerShell](#) ou [Guide de démarrage rapide C# : Créer un index Recherche cognitive Azure à l'aide du Kit de développement logiciel \(SDK\) .NET](#).

### Actions d'indexation : upload, merge, mergeOrUpload, supprimer

Vous pouvez contrôler le type d'action d'indexation par document, en spécifiant si le document doit être chargé en intégralité, fusionné avec du contenu de document existant ou supprimé.

Dans l'API REST, émettez des requêtes HTTP POST avec un corps de requête JSON à l'URL de point de terminaison de votre index Recherche cognitive Azure. Chaque objet JSON du tableau « valeur » contient la clé du document et spécifie qu'une action d'indexation doit ajouter, mettre à jour ou supprimer le contenu d'un document. Pour un exemple de

code, consultez [Charger des documents](#).

Dans le kit de développement logiciel (SDK) .NET, empaquetez vos données dans un objet `IndexBatch`. Un objet `IndexBatch` encapsule une collection d'objets `IndexAction`, chacun d'entre eux contenant un document et une propriété qui indique à Recherche cognitive Azure les actions à effectuer sur ce document. Pour obtenir un exemple de code, consultez le [guide de démarrage rapide C#](#).

<code>@SEARCH.ACTION</code>	DESCRIPTION	CHAMPS REQUIS POUR CHAQUE DOCUMENT	NOTES
<code>upload</code>	Une action <code>upload</code> est similaire à celle d'un « upsert », où le document est inséré s'il est nouveau et mis à jour/remplacé s'il existe déjà.	une clé, ainsi que tout autre champ que vous souhaitez définir	Lors de la mise à jour ou du remplacement d'un document existant, un champ qui n'est pas spécifié dans la requête sera défini sur la valeur <code>null</code> , y compris lorsque le champ a été précédemment défini sur une valeur non null.
<code>merge</code>	Met à jour un document existant avec les champs spécifiés. Si le document n'existe pas dans l'index, la fusion échoue.	une clé, ainsi que tout autre champ que vous souhaitez définir	N'importe quel champ que vous spécifiez dans une fusion remplace le champ existant dans le document. Dans le kit de développement logiciel (SDK), cela inclut les champs de type <code>DataType.Collection(DataType.String)</code> . Dans l'API REST, cela inclut les champs de type <code>Collection(Edm.String)</code> . Par exemple, si le document contient un champ <code>tags</code> avec la valeur <code>["budget"]</code> et que vous exécutez une fusion avec la valeur <code>["economy", "pool"]</code> pour le champ <code>tags</code> , la valeur finale du champ <code>tags</code> sera <code>["economy", "pool"]</code> , et non <code>["budget", "economy", "pool"]</code>
<code>mergeOrUpload</code>	Cette action est similaire à celle d'une action <code>merge</code> s'il existe déjà dans l'index un document comportant la clé spécifiée. Dans le cas contraire, elle exécutera une action <code>upload</code> avec un nouveau document.	une clé, ainsi que tout autre champ que vous souhaitez définir	-
<code>delete</code>	Cette action supprime de l'index le document spécifié.	clé uniquement	Tous les champs que vous spécifiez en dehors du champ de clé sont ignorés. Si vous souhaitez supprimer un champ individuel dans un document, utilisez plutôt <code>merge</code> et définissez simplement le champ de manière explicite sur la valeur null.

## Formuler votre requête

Deux méthodes permettent d'effectuer une [recherche dans un index à l'aide de l'API REST](#). L'une consiste à émettre une requête HTTP POST, dans laquelle vos paramètres de requête sont définis dans un objet JSON contenu dans le corps de la requête. L'autre consiste à émettre une requête HTTP GET, dans laquelle vos paramètres de requête seront définis à l'intérieur de l'URL de requête. Notez que les limites en matière de taille des paramètres de requête sont [plus souples](#) pour la méthode POST que pour la méthode GET. Pour cette raison, nous vous recommandons d'utiliser POST, à moins que la situation justifie l'utilisation de GET.

Pour POST et GET, vous devez fournir votre *nom de service*, *nom d'index* et une *version de l'API* dans l'URL de la demande.

Pour la méthode GET, vous renseignez les paramètres de requête au niveau de la *chaîne de requête* à la fin de l'URL. Voici le format URL à utiliser :

```
https://[service name].search.windows.net/indexes/[index name]/docs?[query string]&api-version=2019-05-06
```

La méthode POST suit un format identique, mais seule `api-version` dans les paramètres de chaîne de requête.

## Extraction de données dans un index

Le modèle d'extraction analyse une source de données prise en charge et charge automatiquement les données dans votre index. Dans Recherche cognitive Azure, cette fonctionnalité est implémentée via des *indexeurs*, actuellement disponibles pour ces plateformes :

- [Stockage Blob](#)
- [Stockage Table](#)
- [Azure Cosmos DB](#)
- [Azure SQL Database, SQL Managed Instance et SQL Server sur des machines virtuelles Azure](#)

Les indexeurs connectent un index à une source de données (généralement une table, une vue ou une structure équivalente) et mappent les champs source aux champs équivalents de l'index. Pendant l'exécution, l'ensemble de lignes est automatiquement transformé en JSON et chargé dans l'index spécifié. Tous les indexeurs prennent en charge la planification de sorte que vous puissiez spécifier la fréquence à laquelle les données sont à actualiser. La plupart des indexeurs fournissent le suivi des modifications si la source de données le prend en charge. En suivant les modifications et les suppressions effectuées dans les documents existants, et en reconnaissant les nouveaux documents, les indexeurs suppriment la nécessité de gérer activement les données de votre index.

### Comment extraire des données dans un index Recherche cognitive Azure

La fonctionnalité de l'indexeur est exposée dans le [Portail Azure](#), ainsi que dans [l'API REST](#) et le [Kit de développement logiciel \(SDK\) .NET](#).

Grâce au portail, Recherche cognitive Azure peut générer un schéma d'index par défaut pour vous en lisant les métadonnées du jeu de données source. Vous pouvez modifier l'index généré jusqu'à ce que l'index soit traité, après quoi les seules modifications de schéma autorisées sont celles qui ne nécessitent pas la réindexation. Si les modifications que vous souhaitez apporter impactent directement le schéma, vous devez reconstruire l'index.

## Vérifier l'importation de données avec l'Explorateur de recherche

Un moyen rapide d'effectuer une vérification préliminaire sur le téléchargement de document consiste à utiliser l'[Explorateur de recherche](#) dans le portail. L'Explorateur vous permet d'interroger un index sans avoir à écrire du code. L'expérience de recherche est basée sur les paramètres par défaut, tels que la [syntaxe simple](#) et le [paramètre de requête searchMode](#). Les résultats sont retournés au format JSON afin que vous puissiez inspecter le document dans son intégralité.

**TIP**

De nombreux [exemples de code Recherche cognitive Azure](#) incluent des jeux de données incorporés ou disponibles rapidement, offrant ainsi une prise en main simplifiée. Le portail fournit également un exemple d'indexeur et de source de données composée d'un petit jeu de données immobilières (nommé « realstate-us-sample »). Lorsque vous exécutez l'indexeur préconfiguré sur l'exemple de source de données, un index est créé et chargé avec des documents qui peuvent ensuite être interrogés dans l'Explorateur de recherche ou par le code que vous écrivez.

## Voir aussi

- [Présentation de l'indexeur](#)
- [Procédure pas à pas dans le portail : créer, charger, interroger un index](#)

# Assistant Importation de données pour la Recherche cognitive Azure

04/10/2020 • 20 minutes to read • [Edit Online](#)

Le tableau de bord Recherche cognitive Azure du Portail Azure propose un Assistant **Importation des données** destiné au prototypage et au chargement d'un index. Cet article s'intéresse aux avantages et aux limitations de l'utilisation de l'Assistant, aux entrées et sorties et à certains aspects liés à l'utilisation. Pour obtenir des conseils pratiques concernant l'utilisation de l'Assistant avec les exemples de données intégrés, consultez le guide de démarrage rapide [Créer un index Recherche cognitive Azure à partir du Portail Azure](#).

Cet Assistant effectue les opérations suivantes :

- 1 - Connexion à une source de données Azure prise en charge.
- 2 - Création d'un schéma d'index, déduit par l'échantillonnage des données sources.
- 3 - (Facultatif) Ajout d'enrichissements de l'IA pour extraire ou générer du contenu et la structure.
- 4 - Exécution de l'Assistant pour créer des objets, importer des données, définir une planification et d'autres options de configuration.

L'Assistant génère un certain nombre d'objets qui sont enregistrés dans votre service de recherche, auquel vous pouvez accéder par programmation ou avec d'autres outils.

## Avantages et limitations

Avant d'écrire du code, vous pouvez utiliser l'Assistant à des fins de prototypage et de test de preuve de concept. L'Assistant se connecte à des sources de données externes, échantillonne les données pour créer un index initial, puis importe les données sous forme de documents JSON dans un index dans la Recherche cognitive Azure.

L'échantillonnage est le processus par lequel un schéma d'index est déduit et présente quelques limitations. Quand la source de données est créée, l'Assistant choisit un échantillon de documents pour identifier les colonnes qui font partie de la source de données. Tous les fichiers ne sont pas lus, car l'opération pourrait durer des heures avec les sources de données très volumineuses. À partir d'une sélection de documents, les métadonnées sources, comme le nom ou le type de champ, sont utilisées pour créer une collection de champs dans un schéma d'index. Selon la complexité des données sources, vous devrez peut-être modifier le schéma initial dans un souci de précision, ou l'étendre à des fins d'exhaustivité. Vous pouvez faire en sorte que vos modifications soient incorporées dans la page de définition de l'index.

Globalement, les avantages de l'Assistant sont évidents : dans la mesure où les exigences sont respectées, vous pouvez créer un prototype d'index interrogeable en quelques minutes. Certaines complexités liées à l'indexation, comme la mise à disposition des données sous forme de documents JSON, sont gérées par l'Assistant.

En bref, les limitations connues sont les suivantes :

- L'Assistant ne prend pas en charge l'itération ou la réutilisation. Chaque exécution de l'Assistant donne lieu à la création d'une configuration d'index, d'ensemble de compétences et d'indexeur. Seules les sources de données peuvent être conservées et réutilisées dans l'Assistant. Pour modifier ou affiner d'autres objets, vous devez utiliser les API REST ou le SDK .NET pour récupérer et modifier les structures.

- Le contenu source doit résider dans une source de données Azure prise en charge.
- L'échantillonnage porte sur un sous-ensemble des données sources. Pour les sources de données volumineuses, l'Assistant peut ne pas déceler des champs. Vous serez peut-être amené à étendre le schéma ou à corriger les types de données déduits si l'échantillonnage est insuffisant.
- L'enrichissement par IA, tel que présenté sur le portail, se limite à quelques compétences intégrées.
- Une [base de connaissances](#), qui peut être créée par l'Assistant, est limitée à quelques projections par défaut. Si vous voulez enregistrer des documents enrichis créés par l'Assistant, le conteneur d'objets blob et les tables sont fournis avec des noms et une structure par défaut.

## Entrée de source de données

L'Assistant **Importation des données** se connecte à une source de données externe en utilisant la logique interne fournie par les indexeurs Recherche cognitive Azure, qui sont capables d'échantillonner la source, lire les métadonnées, décrypter les documents pour en lire le contenu et la structure et sérialiser le contenu sous forme de JSON pour une importation ultérieure dans la Recherche cognitive Azure.

Vous ne pouvez importer des données qu'à partir d'une seule table, d'une vue de base de données ou d'une structure de données équivalente, mais la structure peut inclure des sous-structures hiérarchiques ou imbriquées. Pour plus d'informations, consultez [How to model complex types](#) (Modélisation des types complexes).

Vous devez créer cette table ou vue unique avant d'exécuter l'Assistant et il/elle doit comporter du contenu. Pour des raisons évidentes, exécuter l'Assistant **Importation des données** sur une source de données vide ne présente aucun intérêt.

SÉLECTION	DESCRIPTION
<b>Source de données existante</b>	Si des indexeurs sont déjà définis dans votre service de recherche, il existe peut-être une définition de source de données que vous pouvez réutiliser. Dans la Recherche cognitive Azure, les objets de source de données sont uniquement utilisés par les indexeurs. Vous pouvez créer un objet de source de données par programmation ou via l'Assistant <b>Importation des données</b> , puis le réutiliser si nécessaire.
<b>Exemples</b>	La Recherche cognitive Azure intègre deux exemples de sources de données qui sont utilisées dans des tutoriels et autres guides de démarrage rapide : une base de données immobilières SQL et une base d'hôtels toutes deux hébergées dans Cosmos DB. Pour consulter une procédure pas à pas basée sur l'exemple Hotels, reportez-vous au guide de démarrage rapide <a href="#">Créer un index sur le portail Azure</a> .

SÉLECTION	DESCRIPTION
<a href="#">Azure SQL Database ou SQL Managed Instance</a>	<p>Le nom du service, les informations d'identification d'un utilisateur de base de données avec autorisation de lecture, ainsi que le nom de la base de données peuvent être spécifiés sur la page ou par le biais d'une chaîne de connexion ADO.NET. Choisissez l'option de chaîne de connexion permettant d'afficher ou de personnaliser les propriétés.</p> <p>La table ou la vue qui fournit l'ensemble de lignes doit être spécifiée sur la page. Cette option s'affiche une fois que la connexion aboutit : vous pouvez alors faire votre choix dans une liste déroulante.</p>
<a href="#">SQL Server dans les machines virtuelles Azure</a>	<p>Spécifiez un nom de service complet, un ID d'utilisateur et un mot de passe, ainsi qu'une base de données pour la chaîne de connexion. Afin d'utiliser cette source de données, vous devez avoir préalablement installé un certificat dans le magasin local pour chiffrer la connexion. Pour obtenir des instructions, reportez-vous à <a href="#">Connexion de machines virtuelles SQL à la Recherche cognitive Azure</a>.</p> <p>La table ou la vue qui fournit l'ensemble de lignes doit être spécifiée sur la page. Cette option s'affiche une fois que la connexion aboutit : vous pouvez alors faire votre choix dans une liste déroulante.</p>
<a href="#">Azure Cosmos DB</a>	<p>La configuration requise inclut le compte, la base de données et la collection. Tous les documents de la collection seront inclus dans l'index. Vous pouvez définir une requête pour aplatis ou filtrer l'ensemble de lignes, ou laisser la requête vide. Aucune requête n'est nécessaire dans cet Assistant.</p>
<a href="#">Stockage Blob Azure</a>	<p>La configuration requise inclut le compte de stockage et un conteneur. Si les noms d'objets blob suivent une convention d'affectation de noms virtuelle à des fins de regroupement, vous pouvez indiquer la partie de répertoire virtuel du nom comme dossier sous le conteneur. Consultez la page <a href="#">Indexation de Stockage Blob</a> pour plus d'informations.</p>
<a href="#">Stockage Table Azure</a>	<p>La configuration requise inclut le compte de stockage et un nom de table. Vous pouvez également spécifier une requête pour extraire un sous-ensemble des tables. Consultez la page <a href="#">Indexation de Stockage Table</a> pour plus d'informations.</p>

## Sortie de l'Assistant

En arrière-plan, l'Assistant crée, configure et appelle les objets suivants. Une fois l'Assistant exécuté, vous pouvez trouver sa sortie dans les pages du portail. La page Vue d'ensemble de votre service contient des listes d'index, des indexeurs, des sources de données et des ensembles de compétences. Les définitions d'index peuvent être affichées sous forme de JSON complet sur le portail. Pour les autres définitions, vous pouvez utiliser l'[API REST](#) pour obtenir des objets spécifiques via GET.

OBJECT	DESCRIPTION
--------	-------------

OBJECT	DESCRIPTION
Source de données	Conserve les informations de connexion aux données sources, notamment les informations d'identification. Un objet de source de données est utilisé exclusivement avec les indexeurs.
Index	Structure de données physique utilisée pour la recherche en texte intégral et d'autres requêtes.
Ensemble de compétences	Ensemble complet d'instructions destiné à manipuler, transformer et mettre en forme du contenu, notamment en analysant et extrayant des informations de fichiers image. À l'exception des structures très simples et limitées, il comporte une référence à une ressource Cognitive Services qui assure l'enrichissement. Il peut aussi éventuellement contenir une définition de base de connaissances.
Indexeur	Objet de configuration spécifiant une source de données, un index cible, un ensemble de compétences facultatif, une planification facultative et des paramètres de configuration facultatifs pour la gestion des erreurs et l'encodage en base 64.

## Comment démarrer l'Assistant

L'Assistant Importation des données se démarre à partir de la barre de commandes dans la page Vue d'ensemble du service.

1. Dans le [portail Azure](#), ouvrez la page du service de recherche à partir du tableau de bord, ou [recherchez votre service](#) dans la liste.
2. En haut de la page de présentation du service, cliquez sur **Importer des données**.



Vous pouvez aussi lancer l'**Importation des données** à partir d'autres services Azure, dont Azure Cosmos DB, Azure SQL Database, SQL Managed Instance et le stockage Blob Azure. Recherchez **Ajouter Recherche cognitive Azure** dans le volet de navigation de gauche de la page de présentation du service.

## Comment modifier ou terminer un schéma d'index dans l'Assistant

L'Assistant génère un index incomplet qui sera rempli avec les documents obtenus à partir de la source de données d'entrée. Pour un index fonctionnel, assurez-vous que les éléments suivants sont bien définis.

1. La liste des champs est-elle complète ? Ajoutez de nouveaux champs qui ont échappé à l'échantillonnage et supprimez ceux qui n'apportent rien à une expérience de recherche ou qui ne seront pas utilisés dans une [expression de filtre](#) ou un [profil de scoring](#).
2. Le type de données convient-il pour les données entrantes ? La Recherche cognitive Azure prend en charge les [types de données EDM \(Entity Data Model\)](#). Pour les données Azure SQL, il existe un [tableau de mappages](#) qui présente les valeurs équivalentes. Pour plus d'informations, consultez [Mappages et transformations de champs](#).
3. Avez-vous un champ qui peut faire office de *clé* ? Ce champ doit être Edm.string et doit identifier un document de manière unique. Dans le cas des données relationnelles, elles peuvent être mappées à une

clé primaire. Pour les objets blob, il peut s'agir de `metadata-storage-path`. Si des valeurs de champ comportent des espaces ou des tirets, vous devez définir l'option **Clé d'encodage en base-64** dans l'étape **Créer un indexeur**, sous **Options avancées**, pour supprimer la vérification de la validation pour ces caractères.

#### 4. Définissez des attributs pour déterminer comment ce champ est utilisé dans un index.

Prenez votre temps dans cette étape, car les attributs déterminent l'expression physique des champs dans l'index. Si, par la suite, vous souhaitez modifier des attributs, même par programmation, vous devrez presque toujours supprimer et regénérer l'index. Les attributs de base comme **Searchable** et **Retrievable** ont un [impact négligeable sur le stockage](#). L'activation de filtres et l'utilisation de suggesteurs augmentent les besoins de stockage.

- **Possibilité de recherche** permet une recherche en texte intégral. Chaque champ utilisé dans les requêtes de forme libre ou dans les expressions de requête doit avoir cet attribut. Les index inversés sont créés pour chaque champ que vous marquez comme **Possibilité de recherche**.
- **Récupérable** retourne le champ dans les résultats de la recherche. Chaque champ qui fournit du contenu aux résultats de recherche doit avoir cet attribut. La définition de ce champ n'a pas d'incidence notable sur la taille de l'index.
- **Filtrable** permet de référencer le champ dans les expressions de filtre. Chaque champ utilisé dans une expression `$filter` doit avoir cet attribut. Les expressions de filtre sont des correspondances exactes. Les chaînes de texte demeurant intactes, un stockage supplémentaire est nécessaire pour recevoir le contenu textuel.
- **À choix multiples** active le champ pour la navigation par facettes. Seuls les champs également marqués comme **Filtrables** peuvent être marqués comme **À choix multiples**.
- **Triable** permet d'utiliser le champ dans un tri. Chaque champ utilisé dans une expression `$orderby` doit avoir cet attribut.

#### 5. Avez-vous besoin d'une [analyse lexicale](#) ? Pour les champs Edm.string de type **Searchable**, vous pouvez définir un **analyseur** si vous voulez des fonctions d'indexation et d'interrogation qui offrent une prise en charge linguistique améliorée.

La valeur par défaut est *Standard Lucene*, mais vous pouvez choisir *Microsoft Anglais* si vous souhaitez utiliser l'analyseur de Microsoft pour le traitement lexical avancé, tel que la résolution des formes verbales et nominales irrégulières. Seuls des analyseurs linguistiques peuvent être spécifiés sur le portail. L'utilisation d'un analyseur personnalisé ou d'un analyseur non linguistique, comme **Keyword**, **Pattern**, etc., doit s'effectuer par programmation. Pour plus d'informations sur les analyseurs, consultez [Ajouter des analyseurs linguistiques](#).

#### 6. Avez-vous besoin de fonctionnalités TypeAhead matérialisées par la saisie semi-automatique ou les suggestions de résultats ? Cochez la case **Suggesteur** pour activer les [suggestions de requête TypeAhead](#) et la saisie semi-automatique sur les champs sélectionnés. Les suggesteurs s'ajoutent au nombre de termes tokenisés de votre index et occupent donc davantage d'espace de stockage.

## Étapes suivantes

La meilleure façon de comprendre les avantages et les limitations de l'Assistant est de le parcourir pas à pas. Le guide de démarrage rapide suivant vous guide à chaque étape.

[Créer un index Recherche cognitive Azure à l'aide du Portail Azure](#)

# Guide pratique pour régénérer un index dans la Recherche cognitive Azure

04/10/2020 • 14 minutes to read • [Edit Online](#)

Cet article explique comment régénérer un index Recherche cognitive Azure et les circonstances dans lesquelles les régénérations sont nécessaires, et il contient des suggestions pour atténuer l'impact des régénérations sur les demandes des requêtes en cours.

Une *régénération* fait référence à la suppression et à la recréation des structures de données physiques associées à un index, notamment tous les index inversés basés sur un champ. Dans Recherche cognitive Azure, vous ne pouvez pas supprimer et recréer des champs un par un. Pour régénérer un index, la totalité du stockage des champs doit être supprimé, recréé sur la base d'un schéma d'index existant ou révisé, puis à nouveau rempli avec les données envoyées à l'index ou extraites de sources externes.

Il est courant de régénérer les index pendant le développement quand vous itérez sur la conception de l'index, mais il peut également être nécessaire de les régénérer au niveau de la production pour prendre en compte des modifications structurelles, comme l'ajout de types complexes ou l'ajout de champs à des suggesteurs.

## « Régénérer » ou « actualiser » ?

Ne confondez pas la régénération avec l'actualisation du contenu d'un index avec des documents nouveaux, modifiés ou supprimés. L'actualisation d'un corpus de recherche est presque inévitable dans toute application de recherche, certains scénarios nécessitant même des mises à jour à la minute (par exemple quand un corpus de recherche doit refléter les modifications d'inventaire dans une application de vente en ligne).

Tant que vous ne changez pas la structure de l'index, vous pouvez l'actualiser en appliquant les mêmes techniques que celles appliquées pour le charger initialement :

- Pour l'indexation en mode Push,appelez [Ajouter, mettre à jour ou supprimer des documents](#) pour envoyer (push) les modifications vers un index.
- Pour les indexeurs, vous pouvez [planifier l'exécution de l'indexeur](#) et utiliser le suivi des modifications ou des horodatages pour identifier le delta. Si les mises à jour doivent être reflétées plus rapidement que ce qu'un planificateur peut gérer, vous pouvez utiliser l'indexation en mode Push à la place.

## Conditions de la recréation

Supprimez un index et recréez-en un si l'une des conditions suivantes est vraie.

CONDITION	DESCRIPTION
Modifier une définition de champ	La révision d'un nom de champ, d'un type de données ou <a href="#">d'attributs d'index</a> spécifiques (interrogeable, filtrable, triable, à choix multiples) exige une régénération complète.

CONDITION	DESCRIPTION
Affecter un analyseur à un champ	Les <a href="#">analyseurs</a> sont définis dans un index, puis affectés à des champs. Vous pouvez à tout moment ajouter une nouvelle définition d'analyseur à un index, mais il n'est possible <i>d'affecter</i> l'analyseur qu'à la création du champ. Cette condition s'applique à la fois à la propriété <b>analyzer</b> et à la propriété <b>indexAnalyzer</b> . La propriété <b>searchAnalyzer</b> fait figure d'exception (elle peut être affectée à un champ existant).
Mettre à jour ou supprimer une définition d'analyseur dans un index	Il n'est pas possible de supprimer ou de modifier une configuration d'analyseur existante (analyseur, générateur de jetons, filtre de jetons ou filtre de caractères) dans l'index, à moins de régénérer la totalité de l'index.
Ajouter un champ à un suggesteur	Pour pouvoir ajouter un champ existant à une construction <a href="#">Suggesteurs</a> , il faut régénérer l'index.
Supprimer un champ	Pour supprimer physiquement toutes les traces d'un champ, vous devez régénérer l'index. Lorsqu'une régénération immédiate n'est pas pratique, vous pouvez modifier le code de l'application pour désactiver l'accès au champ « supprimé » ou utiliser le <a href="#">paramètre de requête \$select</a> pour choisir les champs représentés dans le jeu de résultats. Physiquement, le contenu et la définition du champ restent dans l'index jusqu'à la régénération suivante, lors de laquelle est appliqué un schéma omettant le champ en question.
Changer de niveau	Si vous avez besoin de davantage de capacité, il n'y a pas de mise à niveau sur place dans le portail Azure. Vous devez créer un service et régénérer entièrement les index sur le nouveau service. Pour faciliter l'automatisation de ce processus, vous pouvez utiliser l'exemple de code <a href="#">index-backup-restore</a> dans cet <a href="#">exemple de dépôt .NET Recherche cognitive Azure</a> . Cette application va sauvegarder votre index dans une série de fichiers JSON, puis recréer l'index dans un service de recherche que vous spécifiez.

## Conditions de mise à jour

Beaucoup d'autres modifications peuvent être effectuées sans impacter les structures physiques existantes. Plus précisément, les modifications suivantes n'exigent *pas* de régénération d'index. Pour ces modifications, vous pouvez [mettre à jour une définition d'index](#) avec vos modifications.

- Ajouter un nouveau champ
- Définir l'attribut  **retrievable** sur un champ existant
- Définir un **searchAnalyzer** sur un champ existant
- Ajouter une nouvelle définition d'analyseur dans un index
- Ajouter, mettre à jour ou supprimer des profils de scoring
- Ajouter, mettre à jour ou supprimer des paramètres CORS
- Ajouter, mettre à jour ou supprimer des synonymMaps

Quand vous ajoutez un nouveau champ, les documents indexés existants reçoivent une valeur null pour le nouveau champ. Lors de l'actualisation suivante des données, les valeurs provenant des données sources externes remplacent les valeurs null ajoutées par Recherche cognitive Azure. Pour plus d'informations sur la mise à jour du contenu des index, consultez [Ajouter, mettre à jour ou supprimer des documents](#).

## Comment regénérer un index

Durant le développement, le schéma d'index change fréquemment. Anticipez cela en créant des index qui peuvent être rapidement supprimés, recréés et rechargés avec un petit jeu de données représentatif.

Pour les applications déjà en production, nous recommandons de créer un nouvel index qui s'exécute côté à côté avec un index existant pour éviter des temps d'arrêt dans les requêtes. Votre code d'application fournit la redirection vers le nouvel index.

L'indexation n'est pas exécutée en arrière-plan et le service équilibre l'indexation supplémentaire par rapport aux requêtes en cours. Pendant l'indexation, vous pouvez [surveiller les demandes de requête](#) dans le portail pour vous assurer que les demandes sont traitées en temps voulu.

1. Déterminez si une régénération est nécessaire. Si vous ajoutez juste des champs, ou modifiez une partie de l'index qui n'est pas liée à des champs, vous pouvez simplement [mettre à jour la définition](#) sans supprimer, recréer et recharger entièrement l'index.
2. [Obtenez une définition d'index](#) pour vous y référer ultérieurement au besoin.
3. [Supprimez l'index existant](#), en supposant que vous n'exécutez pas le nouvel index et l'ancien index simultanément.

Toutes les requêtes ciblant cet index sont immédiatement supprimées. Souvenez-vous que la suppression d'un index est irréversible, détruisant le stockage physique pour la collection de champs et d'autres constructions. Prenez le temps de réfléchir aux implications avant de supprimer l'index.

4. [Créez un index revu](#), où le corps de la requête contient les définitions des champs nouveaux ou modifiés.
5. [Chargez l'index avec des documents](#) d'une source externe.

Quand vous créez l'index, du stockage physique est alloué pour chaque champ dans le schéma d'index, avec un index inversé créé pour chaque champ avec possibilité de recherche. Des champs sans possibilité de recherche peuvent être utilisés dans des filtres ou des expressions, mais ils n'ont pas d'index inversé et n'autorisent pas la recherche approximative ou en texte intégral. Lors d'une régénération d'index, ces index inversés sont supprimés et recréés sur la base du schéma d'index que vous fournissez.

Quand vous chargez l'index, l'index inversé de chaque champ est rempli avec tous les mots uniques tokenisés de chaque document, avec un mappage aux ID des documents correspondants. Par exemple, lors de l'indexation d'un jeu de données avec des hôtels, un index inversé créé pour un champ Ville peut contenir des termes pour Seattle, Portland, etc. L'ID des documents qui incluent « Seattle » ou « Portland » dans le champ Ville figure à côté du terme. Lors d'une opération [Ajouter, mettre à jour ou supprimer](#), les termes et la liste des ID de document sont mis à jour en conséquence.

### NOTE

Si vous avez des exigences strictes dans le cadre d'un contrat SLA, vous pouvez envisager de provisionner un nouveau service spécifiquement pour ce travail, le développement et l'indexation se produisant dans une isolation complète d'un index de production. Un service distinct s'exécute sur son propre matériel, éliminant toute possibilité de contention des ressources. Une fois le développement terminé, vous laissez le nouvel index en place et vous redirigez les requêtes vers le nouveau point de terminaison et le nouvel index, ou bien vous exécutez le code terminé pour publier un index revu sur votre service Recherche cognitive Azure d'origine. Il n'existe actuellement aucun mécanisme pour déplacer un index prêt à l'emploi vers un autre service.

## Rechercher les mises à jour

Vous pouvez commencer à interroger un index dès que le premier document est chargé. Si vous connaissez l'ID d'un document, l'[API REST de recherche de document](#) retourne le document spécifique. Pour un test plus large,

attendez que l'index soit entièrement chargé, puis utilisez des requêtes pour vérifier le contexte que vous vous attendez à voir.

Vous pouvez utiliser l'[Explorateur de recherche](#) ou un outil de test web comme [Postman](#) pour rechercher du contenu mis à jour.

Si vous avez ajouté ou renommé un champ, utilisez `$select` pour retourner ce champ :

```
search=*&$select=document-id,my-new-field,some-old-field&$count=true .
```

## Voir aussi

- [Présentation de l'indexeur](#)
- [Indexer de grands jeux de données à grande échelle](#)
- [Indexation dans le portail](#)
- [Indexeur Azure SQL Database](#)
- [Indexeur Azure Cosmos DB](#)
- [Indexeur Stockage Blob Azure](#)
- [Indexeur Stockage Table Azure](#)
- [Sécurité dans Recherche cognitive Azure](#)

# Comment indexer des grands ensembles de données dans la Recherche cognitive Azure

04/10/2020 • 20 minutes to read • [Edit Online](#)

Recherche cognitive Azure prend en charge [deux approches de base](#) pour importer des données dans un index de recherche : *envoyer (push)* les données dans l'index par programme ou pointer un [indexeur Recherche cognitive Azure](#) vers une source de données prise en charge pour *extraire (pull)* les données.

Au fur et à mesure que les volumes de données augmentent ou que les besoins en traitement évoluent, vous trouverez peut-être que les stratégies d'indexation simples ou par défaut ne sont plus adaptées. Pour la Recherche cognitive Azure, il existe plusieurs approches pour prendre en charge les grands jeux de données, allant de la façon dont une demande de chargement de données est structurée à l'utilisation d'un indexeur spécifique à la source pour les charges de travail planifiées et distribuées.

Les mêmes techniques s'appliquent également aux processus à exécution longue. En particulier, les étapes décrites dans [Indexation parallèle](#) sont utiles pour l'indexation gourmande en ressources, comme l'analyse d'images ou le traitement en langage naturel dans un [pipeline d'enrichissement de l'IA](#).

Les sections suivantes décrivent des techniques d'indexation de grandes quantités de données à l'aide de l'API Push et des indexeurs.

## API push

Lors de l'envoi (push) de données dans un index, plusieurs considérations importantes affectent les vitesses d'indexation pour l'API Push. Ces facteurs sont décrits dans la section ci-dessous.

En plus des informations contenues dans cet article, vous pouvez également tirer parti des exemples de code dans le tutoriel [Optimiser l'indexation avec l'API Push](#).

### Niveau de service et nombre de partitions/réplicas

L'ajout de partitions ou l'augmentation du niveau de votre service de recherche augmente les vitesses d'indexation.

L'ajout de réplicas supplémentaires peut également augmenter les vitesses d'indexation, mais cela n'est pas garanti. En revanche, les réplicas supplémentaires augmentent le volume de requêtes que votre service de recherche peut gérer. Les réplicas sont également un composant clé pour l'obtention d'un contrat [SLA](#).

Avant d'ajouter des partitions/réplicas ou de procéder à une mise à niveau vers un niveau supérieur, prenez en compte des paramètres tels que le coût et le temps d'allocation. L'ajout de partitions peut augmenter considérablement la vitesse d'indexation, mais l'ajout/la suppression peut prendre de 15 minutes à plusieurs heures. Pour plus d'informations, consultez la documentation sur le [réglage de capacité](#).

### Schéma d'index

Le schéma de votre index joue un rôle important dans l'indexation des données. L'ajout de champs et l'ajout de propriétés supplémentaires à ces champs (comme *searchable*, *facetable* ou *filterable*) réduisent les vitesses d'indexation.

En général, nous vous recommandons d'ajouter des propriétés supplémentaires aux champs uniquement si vous envisagez de les utiliser.

#### **NOTE**

Pour réduire la taille du document, évitez d'ajouter des données non interrogeables à un index. Les images et autres données binaires ne peuvent pas faire l'objet de recherches directes et ne doivent pas être stockées dans l'index. Pour intégrer des données non interrogeables dans les résultats de la recherche, vous devez définir un champ sans possibilité de recherche qui stocke une référence d'URL vers la ressource.

### **Taille du lot**

Un des mécanismes les plus simples pour l'indexation d'un grand jeu de données consiste à soumettre plusieurs documents ou enregistrements dans une même demande. Tant que la charge utile entière est inférieure à 16 Mo, une demande peut gérer jusqu'à 1 000 documents dans une opération de chargement en bloc. Ces limites s'appliquent que vous utilisez l'[API REST d'ajout de documents](#) ou la [méthode Index](#) du SDK .NET. Pour l'une ou l'autre des API, vous devez empaqueter 1 000 documents dans le corps de chaque requête.

L'indexation de documents par lots améliorera considérablement les performances d'indexation. La détermination de la taille de lot optimale pour vos données est un composant clé de l'optimisation des vitesses d'indexation. Les deux principaux facteurs qui influencent la taille de lot optimale sont les suivants :

- Le schéma de votre index
- La taille de vos données

Étant donné que la taille de lot optimale dépend de votre index et de vos données, la meilleure approche consiste à tester différentes tailles de lot pour déterminer ce qui produit les vitesses d'indexation les plus rapides pour votre scénario. Ce [tutoriel](#) fournit des exemples de code pour tester les tailles de lot à l'aide du kit de développement logiciel (SDK) .NET.

### **Nombre de threads/rôles de travail**

Pour tirer pleinement parti des vitesses d'indexation de Recherche cognitive Azure, vous devrez probablement utiliser plusieurs threads pour envoyer simultanément des demandes d'indexation par lot au service.

Le nombre optimal de threads est déterminé par :

- Le niveau de votre service de recherche
- Nombre de partitions
- La taille de vos lots
- Le schéma de votre index

Vous pouvez modifier cet exemple et effectuer des tests avec différents nombres de threads pour déterminer le nombre de threads optimal pour votre scénario. Toutefois, tant que plusieurs threads s'exécutent simultanément, vous devriez être en mesure de bénéficier de la majeure partie des gains d'efficacité.

#### **NOTE**

Au fur et à mesure que vous augmentez le niveau de votre service de recherche ou que vous augmentez les partitions, vous devez également augmenter le nombre de threads simultanés.

Au fur et à mesure que les requêtes atteignent le service de recherche, vous pouvez rencontrer des [codes d'état HTTP](#) indiquant que la demande n'a pas abouti. Pendant l'indexation, deux codes d'état HTTP courants sont :

- **503 Service indisponible** : Cette erreur signifie que le système est surchargé et que votre requête ne peut pas être traitée pour le moment.
- **207 Multi-état** : Cette erreur signifie que certains documents ont réussi, mais qu'au moins un a échoué.

### **Stratégie de nouvelle tentative**

En cas d'échec, les requêtes doivent être retentées à l'aide d'une [stratégie de nouvelle tentative d'interruption exponentielle](#).

Le kit de développement logiciel (SDK) .NET de Recherche cognitive Azure retente automatiquement lors des erreurs 503 et autres requêtes ayant échoué, mais vous devez implémenter votre propre logique pour réessayer en cas de code 207. Des outils open source tels que [Polly](#) peuvent également être utilisés pour mettre en œuvre une stratégie de nouvelle tentative.

### Vitesses de transfert de données réseau

La vitesse de transfert des données peut être un facteur limitatif lors de l'indexation des données. L'indexation de données à partir de votre environnement Azure est un moyen simple d'accélérer l'indexation.

## Indexeurs

Les [indexeurs](#) sont utilisés pour analyser le contenu pouvant être recherché dans les sources de données Azure prises en charge. Bien qu'ils ne soient pas spécifiquement destinés à l'indexation à grande échelle, plusieurs fonctionnalités des indexeurs sont particulièrement utiles pour prendre en charge les grands jeux de données :

- Les planificateurs vous permettent de diviser l'indexation pour l'effectuer à intervalles réguliers : vous pouvez ainsi la répartir dans le temps.
- L'indexation planifiée peut reprendre au dernier point d'arrêt connu. Si une source de données n'est pas entièrement parcourue dans une fenêtre de 24 heures, l'indexeur reprend l'indexation au deuxième jour, là où elle s'était arrêtée.
- Le partitionnement des données en sources de données individuelles plus petites permet le traitement parallèle. Vous pouvez séparer les données sources en composants plus petits, par exemple en plusieurs conteneurs dans un stockage Blob Azure, puis créer plusieurs [objets de source de données](#) correspondants dans la Recherche cognitive Azure, qui peuvent être indexés en parallèle.

#### NOTE

Les indexeurs sont spécifiques à une source de données : l'utilisation d'une approche par indexeur est donc viable seulement pour des sources de données sélectionnées sur Azure : [SQL Database](#), [Stockage Blob](#), [Stockage Table](#), [Cosmos DB](#).

### Taille du lot

Comme avec l'API Push, les indexeurs vous permettent de configurer le nombre d'éléments par lot. Pour les indexeurs basés sur l'[API REST de création d'un indexeur](#), vous pouvez définir l'argument `batchSize` pour personnaliser ce paramètre de façon à le faire mieux correspondre aux caractéristiques de vos données.

Les tailles de lot par défaut sont spécifiques à la source de données. Azure SQL Database et Azure Cosmos DB ont une taille de lot par défaut de 1 000. À l'inverse, l'indexation des objets blob Azure définit la taille des lots à 10 documents en fonction de la taille moyenne des documents la plus élevée.

### Indexation planifiée

La planification des indexeurs est un mécanisme important pour le traitement des grands jeux de données et pour les processus à exécution longue, comme l'analyse d'images dans un pipeline de recherche cognitive. Le traitement de l'indexeur s'exécute dans une fenêtre de 24 heures. Si le traitement n'est pas terminé dans ce délai de 24 heures, les fonctions de planification de l'indexeur peuvent vous être d'une aide précieuse.

Par conception, l'indexation planifiée démarre à intervalles spécifiques. En général, les tâches sont entièrement exécutées, puis redémarrées au prochain intervalle planifié. Toutefois, si le traitement n'est pas terminé à la fin de l'intervalle, l'indexeur s'arrête (car le délai de traitement a expiré). Au prochain intervalle, le traitement reprend là où il s'était arrêté, le système gardant en mémoire l'endroit où la tâche doit redémarrée.

En pratique, pour les charges d'index réparties sur plusieurs jours, vous pouvez définir une fenêtre d'exécution de

24 heures pour l'indexeur. Quand l'indexation reprend pour le cycle suivant de 24 heures, elle redémarre au dernier document valide connu. De cette façon, un indexeur peut s'exécuter sur un backlog de documents pendant plusieurs jours jusqu'à ce que tous les documents non traités soient traités. Pour plus d'informations sur cette approche, consultez [Indexation de grands jeux de données dans Stockage Blob Azure](#). Pour plus d'informations sur la définition de planifications en général, voir [API REST de création d'indexeur](#) ou [Comment planifier des indexeurs pour la Recherche cognitive Azure](#).

## Indexation parallèle

Une stratégie d'indexation parallèle est basée sur l'indexation de plusieurs sources de données à l'unisson, où chaque définition de source de données spécifie un sous-ensemble des données.

Pour des besoins ponctuels d'indexation gourmande en ressources, comme la reconnaissance de caractères sur des documents numérisés dans un pipeline de recherche cognitive, l'analyse d'images ou un traitement en langage naturel, une stratégie d'indexation parallèle est souvent la bonne approche pour effectuer un processus à exécution longue dans le temps le plus court. Si vous pouvez éliminer ou réduire les demandes de requêtes, l'indexation parallèle sur un service qui ne gère pas simultanément des requêtes peut être un choix de stratégie valide pour traiter un grand volume de contenu dont le traitement est lent.

Un traitement parallèle se déroule comme suit :

- Répartissez vos données sources entre plusieurs conteneurs ou plusieurs dossiers virtuels au sein du même conteneur.
- Mappez chaque petit jeu de données à sa propre [source de données](#), apparée à son propre [indexeur](#).
- Pour la recherche cognitive, référez le même [ensemble de compétences](#) dans chaque définition d'indexeur.
- Écrivez dans le même index de recherche cible.
- Planifiez une exécution simultanée de tous les indexeurs.

### NOTE

Dans Recherche cognitive Azure, vous ne pouvez pas assigner des répliques ou des partitions individuels à l'indexation ou au traitement des requêtes. Le système détermine la façon dont les ressources sont utilisées. Pour comprendre l'impact sur les performances des requêtes, vous pouvez tenter une indexation parallèle dans un environnement de test avant de la déployer en production.

## Comment configurer l'indexation parallèle

Pour les indexeurs, la capacité de traitement est plus ou moins basée sur un sous-système d'indexeur pour chaque unité de service utilisée par votre service de recherche. Plusieurs indexeurs peuvent être exécutés en même temps sur les services Recherche cognitive Azure configurés sur le niveau De base ou Standard et ayant au moins deux répliques.

1. Dans le [portail Azure](#), sur la page **Vue d'ensemble** de votre tableau de bord de service de recherche, vérifiez le **niveau tarifaire** pour vous assurer qu'il est compatible avec l'indexation parallèle. Les niveaux De base et Standard fournissent plusieurs répliques.
2. Vous pouvez exécuter autant d'indexeurs en parallèle que le nombre d'unités de recherche dans votre service. Sous **Paramètres > Échelle**, [augmentez le nombre de répliques](#) ou de partitions pour le traitement parallèle : un réplica ou une partition supplémentaire pour chaque charge de travail d'indexeur. Conservez-en un nombre suffisant pour le volume de requêtes existant. Sacrifier les charges de travail de requête au profit de l'indexation n'est pas judicieux.
3. Répartissez les données dans plusieurs conteneurs à un niveau qui est accessible par les indexeurs Recherche cognitive Azure. Placez-les par exemple dans plusieurs tables dans Azure SQL Database, dans différents conteneurs du stockage Blob Azure ou dans plusieurs collections. Définissez un objet de source de données pour chaque table ou conteneur.

#### 4. Créez plusieurs indexeurs et planifiez leur exécution parallèle :

- Prenons l'exemple d'un service contenant six réplicas. Configurez les six indexeurs, chacun mappé à une source de données contenant un sixième du jeu de données afin de diviser le jeu de données en 6.
- Faites pointer chaque indexeur vers le même index. Pour les charges de travail de recherche cognitive, faites pointer chaque indexeur vers le même ensemble de compétences.
- Dans chaque définition d'indexeur, planifiez le même modèle d'exécution. Par exemple, `"schedule" : { "interval" : "PT8H", "startTime" : "2018-05-15T00:00:00Z" }` crée une planification démarrant le 15-05-2018 sur tous les indexeurs, avec un intervalle d'exécution de huit heures.

À l'heure planifiée, tous les indexeurs commencent à s'exécuter en chargeant les données, en procédant aux enrichissements (si vous avez configuré un pipeline de recherche cognitive) et en écrivant dans l'index. Le service Recherche cognitive Azure ne verrouille pas l'index pour les mises à jour. Il prend en charge les écritures simultanées, en effectuant une nouvelle tentative si une écriture spécifique échoue à la première tentative.

##### NOTE

Lorsque vous augmentez le nombre de réplicas, envisagez d'augmenter le nombre de partitions si vous anticipiez une augmentation significative de la taille de l'index. Les partitions stockent des sections de contenu indexé. Ainsi, plus vous avez de partitions, plus la section de contenu que chaque partition doit stocker est réduite.

## Voir aussi

- [Présentation de l'indexeur](#)
- [Indexation dans le portail](#)
- [Indexeur Azure SQL Database](#)
- [Indexeur Azure Cosmos DB](#)
- [Indexeur Stockage Blob Azure](#)
- [Indexeur Stockage Table Azure](#)
- [Sécurité dans Recherche cognitive Azure](#)

# Gestion de l'accès concurrentiel dans la Recherche cognitive Azure

04/10/2020 • 9 minutes to read • [Edit Online](#)

Lors de la gestion de ressources de la Recherche cognitive Azure telles que des index et des sources de données, il est important de mettre à jour les ressources en toute sécurité, surtout si elles sont accessibles simultanément par différents composants de votre application. Lorsque deux clients mettent à jour une ressource en même temps sans coordination, cela peut créer des conditions de concurrence. Pour éviter ce problème, la Recherche cognitive Azure offre un *modèle d'accès concurrentiel optimiste*. Aucun verrou n'est appliqué aux ressources. Au lieu de cela, chaque ressource présente une étiquette d'entité (ETag) qui identifie la version de la ressource afin que vous puissiez élaborer des requêtes sans risquer de remplacements accidentels.

## TIP

Le code conceptuel de cet [exemple de solution C#](#) illustre le fonctionnement du contrôle d'accès concurrentiel dans la Recherche cognitive Azure. Le code crée des conditions qui appellent le contrôle d'accès concurrentiel. La lecture du [fragment de code ci-dessous](#) est probablement suffisante pour la plupart des développeurs, mais si vous souhaitez l'exécuter, modifiez le fichier appsettings.json pour y ajouter le nom du service et une clé API d'administration. Pour une URL de service `http://myservice.search.windows.net`, le nom du service est `myservice`.

## Fonctionnement

L'accès concurrentiel optimiste est implémenté via des contrôles de conditions d'accès dans les appels d'API écrivant dans des index, des indexeurs, des sources de données et des ressources synonymMap.

Toutes les ressources présentent une [étiquette d'entité \(ETag\)](#) qui fournit des informations sur la version de l'objet. En vérifiant d'abord l'ETag, vous pouvez éviter les mises à jour simultanées dans un flux de travail classique (obtention, modification locale, mise à jour) en vous assurant que l'ETag de la ressource correspond à celui de votre copie locale.

- L'API REST utilise un [ETag](#) sur l'en-tête de demande.
- Le Kit de développement logiciel (SDK) .NET spécifie l'ETag via un objet accessCondition en définissant l'en-tête [If-Match](#) | [If-Match-None](#) sur la ressource. Tout objet héritant de l'interface [IResourceWithETag](#) (Kit de développement logiciel [SDK] .NET) présente un objet accessCondition.

À chaque fois que vous mettez à jour une ressource, son ETag change automatiquement. Lorsque vous implémentez la gestion de l'accès concurrentiel, vous placez simplement sur la requête de mise à jour une condition préalable qui exige que la ressource distante présente le même ETag que la copie de la ressource que vous avez modifiée sur le client. Si un processus simultané a déjà modifié la ressource distante, l'ETag ne correspondra pas à la condition préalable et la requête échouera avec l'erreur HTTP 412. Si vous utilisez le Kit de développement logiciel (SDK) .NET, cela se manifeste sous la forme d'une exception `CloudException`, où la méthode d'extension `IsAccessConditionFailed()` renvoie la valeur true.

## NOTE

Il n'existe qu'un seul mécanisme pour l'accès concurrentiel. Celui-ci est systématiquement utilisé, peu importe l'API employée pour les mises à jour de ressources.

# Cas d'usage et exemple de code

Le code suivant illustre les contrôles accessCondition pour les opérations de mise à jour de clé :

- Échec de la mise à jour si la ressource n'existe plus
- Échec de la mise à jour si la version de la ressource change

## Exemple de code tiré du programme DotNetETagsExplainer

```
class Program
{
    // This sample shows how ETags work by performing conditional updates and deletes
    // on an Azure Cognitive Search index.
    static void Main(string[] args)
    {
        IConfigurationBuilder builder = new ConfigurationBuilder().AddJsonFile("appsettings.json");
        IConfigurationRoot configuration = builder.Build();

        SearchServiceClient serviceClient = CreateSearchServiceClient(configuration);

        Console.WriteLine("Deleting index...\n");
        DeleteTestIndexIfExists(serviceClient);

        // Every top-level resource in Azure Cognitive Search has an associated ETag that keeps track of
        which version
        // of the resource you're working on. When you first create a resource such as an index, its ETag
        is
        // empty.
        Index index = DefineTestIndex();
        Console.WriteLine(
            $"Test index hasn't been created yet, so its ETag should be blank. ETag: '{index.ETag}'");

        // Once the resource exists in Azure Cognitive Search, its ETag will be populated. Make sure to use
        the object
        // returned by the SearchServiceClient! Otherwise, you will still have the old object with the
        // blank ETag.
        Console.WriteLine("Creating index...\n");
        index = serviceClient.Indexes.Create(index);

        Console.WriteLine($"Test index created; Its ETag should be populated. ETag: '{index.ETag}'");

        // ETags let you do some useful things you couldn't do otherwise. For example, by using an If-Match
        // condition, we can update an index using CreateOrUpdate and be guaranteed that the update will
        only
        // succeed if the index already exists.
        index.Fields.Add(new Field("name", AnalyzerName.EnMicrosoft));
        index =
            serviceClient.Indexes.CreateOrUpdate(
                index,
                accessCondition: AccessCondition.GenerateIfExistsCondition());

        Console.WriteLine(
            $"Test index updated; Its ETag should have changed since it was created. ETag:
            '{index.ETag}'");

        // More importantly, ETags protect you from concurrent updates to the same resource. If another
        // client tries to update the resource, it will fail as long as all clients are using the right
        // access conditions.
        Index indexForClient1 = index;
        Index indexForClient2 = serviceClient.Indexes.Get("test");

        Console.WriteLine("Simulating concurrent update. To start, both clients see the same ETag.");
        Console.WriteLine($"Client 1 ETag: '{indexForClient1.ETag}' Client 2 ETag:
            '{indexForClient2.ETag}'");

        // Client 1 successfully updates the index.
        indexForClient1.Fields.Add(new Field("a", DataType.Int32));
```

```

indexForClient1 =
    serviceClient.Indexes.CreateOrUpdate(
        indexForClient1,
        accessCondition: AccessCondition.IfNotChanged(indexForClient1));

Console.WriteLine($"Test index updated by client 1; ETag: '{indexForClient1.ETag}'");

// Client 2 tries to update the index, but fails, thanks to the ETag check.
try
{
    indexForClient2.Fields.Add(new Field("b", DataType.Boolean));
    serviceClient.Indexes.CreateOrUpdate(
        indexForClient2,
        accessCondition: AccessCondition.IfNotChanged(indexForClient2));

    Console.WriteLine("Whoops; This shouldn't happen");
    Environment.Exit(1);
}
catch (CloudException e) when (e.IsAccessConditionFailed())
{
    Console.WriteLine("Client 2 failed to update the index, as expected.");
}

// You can also use access conditions with Delete operations. For example, you can implement an
// atomic version of the DeleteTestIndexIfExists method from this sample like this:
Console.WriteLine("Deleting index...\n");
serviceClient.Indexes.Delete("test", accessCondition: AccessCondition.GenerateIfExistsCondition());

// This is slightly better than using the Exists method since it makes only one round trip to
// Azure Cognitive Search instead of potentially two. It also avoids an extra Delete request in
cases where
    // the resource is deleted concurrently, but this doesn't matter much since resource deletion in
    // Azure Cognitive Search is idempotent.

    // And we're done! Bye!
Console.WriteLine("Complete. Press any key to end application...\n");
Console.ReadKey();
}

private static SearchServiceClient CreateSearchServiceClient(IConfigurationRoot configuration)
{
    string searchServiceName = configuration["SearchServiceName"];
    string adminApiKey = configuration["SearchServiceAdminApiKey"];

    SearchServiceClient serviceClient =
        new SearchServiceClient(searchServiceName, new SearchCredentials(adminApiKey));
    return serviceClient;
}

private static void DeleteTestIndexIfExists(SearchServiceClient serviceClient)
{
    if (serviceClient.Indexes.Exists("test"))
    {
        serviceClient.Indexes.Delete("test");
    }
}

private static Index DefineTestIndex() =>
    new Index()
    {
        Name = "test",
        Fields = new[] { new Field("id", DataType.String) { IsKey = true } }
    };
}
}

```

## Modèle de conception

Un modèle de conception pour l'implémentation de l'accès concurrentiel optimiste doit inclure une boucle qui effectue une nouvelle tentative de contrôle de la condition d'accès, un test de la condition d'accès et récupère éventuellement une ressource mise à jour avant d'essayer de réappliquer les modifications.

Cet extrait de code illustre l'ajout d'une ressource synonymMap à un index existant. Ce code est tiré de l'[exemple C# des synonymes pour la Recherche cognitive Azure](#).

L'extrait de code obtient l'index « hotel », vérifie la version de l'objet pour une opération de mise à jour, lève une exception si la condition échoue, puis retente l'opération (jusqu'à trois fois), en commençant par extraire l'index du serveur pour obtenir sa dernière version.

```
private static void EnableSynonymsInHotelsIndexSafely(SearchServiceClient serviceClient)
{
    int MaxNumTries = 3;

    for (int i = 0; i < MaxNumTries; ++i)
    {
        try
        {
            Index index = serviceClient.Indexes.Get("hotels");
            index = AddSynonymMapsToFields(index);

            // The IfNotChanged condition ensures that the index is updated only if the ETags match.
            serviceClient.Indexes.CreateOrUpdate(index, accessCondition:
AccessCondition.IfNotChanged(index));

            Console.WriteLine("Updated the index successfully.\n");
            break;
        }
        catch (CloudException e) when (e.IsAccessConditionFailed())
        {
            Console.WriteLine($"Index update failed : {e.Message}. Attempt({i}/{MaxNumTries}).\n");
        }
    }
}

private static Index AddSynonymMapsToFields(Index index)
{
    index.Fields.First(f => f.Name == "category").SynonymMaps = new[] { "desc-synonymmap" };
    index.Fields.First(f => f.Name == "tags").SynonymMaps = new[] { "desc-synonymmap" };
    return index;
}
```

## Étapes suivantes

Examinez l'[exemple C# de synonymes](#) pour avoir une illustration de la mise à jour en toute sécurité d'un index existant.

Essayez de modifier l'un des exemples suivants pour inclure des ETags ou des objets AccessCondition.

- [Exemple d'API REST sur GitHub](#)
- [Exemple de SDK .NET sur GitHub](#). Cette solution inclut le projet « DotNetEtagsExplainer », qui contient le code présenté dans cet article.

## Voir aussi

[En-têtes de demande et de réponse HTTP communément utilisés](#) [Codes d'état HTTP Opérations d'index \(API REST\)](#)

# Comment surveiller l'état et les résultats d'un indexeur Recherche cognitive Azure

04/10/2020 • 9 minutes to read • [Edit Online](#)

Recherche cognitive Azure fournit des informations sur l'état et le suivi des exécutions actuelles et historiques de chaque indexeur.

La surveillance de l'indexeur est utile lorsque vous souhaitez :

- Suivre la progression d'un indexeur durant une exécution.
- Passer en revue les résultats de l'exécution en cours ou précédente de l'indexeur.
- Identifier les erreurs de l'indexeur de niveau supérieur et les erreurs ou avertissements concernant l'indexation de documents individuels.

## Obtenir l'état et l'historique

Vous pouvez accéder aux informations de surveillance de l'indexeur de différentes manières, notamment :

- Dans le [portail Azure](#) :
- À l'aide de l'[API REST](#)
- À l'aide du [SDK .NET](#)

Les informations de surveillance de l'indexeur disponibles comprennent toutes les informations suivantes (bien que les formats de données diffèrent en fonction de la méthode d'accès utilisée) :

- Informations d'état de l'indexeur lui-même
- Informations sur l'exécution la plus récente de l'indexeur, y compris son état, les heures de début et de fin et les erreurs et avertissements détaillés.
- Liste des exécutions de l'indexeur d'historique, ainsi que leurs états, résultats, erreurs et avertissements.

L'exécution des indexeurs qui traitent de grands volumes de données peut être longue. Par exemple, les indexeurs qui traitent des millions de documents sources peuvent fonctionner pendant 24 heures, puis redémarrer presque immédiatement. Les indexeurs traitant de grands volumes de données peuvent se trouver encore à l'état **En cours** dans le portail. Même lorsqu'un indexeur est en cours d'exécution, des informations détaillées sont disponibles sur la progression en cours et les exécutions précédentes.

## Surveiller à l'aide du portail

L'état actuel de tous vos indexeurs figure dans la liste **Indexeurs** de la page Vue d'ensemble de votre service de recherche.

Usage	Monitoring	Indexes	Indexers	Data sources	Skillsets
Last result	Name	Status		Last run	Docs succeeded
⌚	azure-sql-indexer	In progress		Just now	0/0
❗	hotel-rooms-blob-indexer	Failed		6 min ago	0/1
ℹ️	hotel-rooms-cosmos-indexer	Reset		4 min ago	7/7
✓	myindexer	Success		8 min ago	50/50

Lorsqu'un indexeur est en cours d'exécution, la liste affiche l'état **En cours** et la valeur de **Documents ayant réussi** indique le nombre de documents traités à ce jour. Le portail peut prendre quelques minutes pour mettre à jour les valeurs d'état de l'indexeur et le nombre de documents.

Un indexeur dont l'exécution la plus récente a réussi affiche **Réussite**. Une exécution de l'indexeur peut réussir même si des documents individuels contiennent des erreurs, car le nombre d'erreurs est inférieur à la valeur du paramètre **Nombre maximal d'éléments en échec** de l'indexeur.

Si la dernière exécution s'est terminée avec une erreur, l'état affiche **Échec**. **Réinitialiser** signifie que l'état de suivi des modifications de l'indexeur a été réinitialisé.

Cliquez sur un indexeur dans la liste pour afficher plus de détails sur les exécutions actuelles et récentes de l'indexeur.

## hotel-rooms-blob-indexer

Indexer

Run Reset Edit Delete

### Indexer summary

#### Execution history

Time (s)	Status
~0.05	SUCCEEDED
~0.95	SUCCEEDED
~1.0	SUCCEEDED

#### Execution details

Last Result	Last Run	Status	Docs Succeeded
!	6/28, 03:29 UTC	Failed	1/2
i	6/28, 03:28 UTC	Reset	0/0
✓	6/28, 00:00 UTC	Success	0/0
✓	6/27, 22:02 UTC	Success	7/7
i	6/27, 22:02 UTC	Reset	0/0
✓	6/27, 22:00 UTC	Success	7/7
i	6/27, 22:00 UTC	Reset	0/0
✓	6/27, 21:23 UTC	Success	7/7
i	6/27, 21:23 UTC	Reset	0/0
✓	6/27, 21:14 UTC	Success	7/7

1 2 3 4 5 < >

Le graphique Résumé de l'indexeur indique le nombre de documents traités lors des exécutions les plus récentes de l'indexeur.

La liste Détails de l'exécution affiche jusqu'à 50 des résultats d'exécution les plus récents.

Cliquez sur un résultat d'exécution dans la liste pour afficher les détails correspondants. Cela inclut les heures de début et de fin, ainsi que les erreurs et les avertissements qui se sont produits.

# myindexer

Execution

## Execution result

**0 Errors/Warnings** 

Status	Success
Datasource	test-cosmos-src
Target index	test-cosmosdb-index
Start	6/28, 03:27 UTC
End	6/28, 03:27 UTC
Documents succeeded	50
Documents failed	0

## Errors

[ ]

[ ]

Si vous rencontrez des problèmes spécifiques au document pendant l'exécution, ils seront signalés dans les champs Erreurs et Avertissements.

# hotel-rooms-blob-indexer

Execution

## Execution result

### 1 Errors/Warnings !

Status	Failed
Datasource	sampleblobstore
Target index	hotel-rooms-sample
Start	6/28, 03:29 UTC
End	6/28, 03:29 UTC
Documents succeeded	0
Documents failed	1

## Errors

```
[{"error": { "key": "https://sampleblobstore.blob.core.windows.net/hotel-rooms/Rooms11.json", "errorMessage": "Document key cannot be missing or empty.\r\n"}}]
```

## Warnings

```
[]
```

Les avertissements sont courants avec certains types d'indexeurs et n'indiquent pas toujours un problème. Par exemple, les indexeurs qui utilisent des services cognitifs peuvent signaler des avertissements lorsque les fichiers image ou PDF ne contiennent pas de texte à traiter.

Pour plus d'informations sur l'analyse des erreurs et des avertissements de l'indexeur, consultez [Résoudre les problèmes courants des indexeurs dans Recherche cognitive Azure](#).

Surveiller à l'aide d'API REST

Vous pouvez récupérer l'état et l'historique d'exécution d'un indexeur à l'aide de la [commande Get Indexer Status](#) :

```
GET https://[service name].search.windows.net/indexers/[indexer name]/status?api-version=2020-06-30  
api-key: [Search service admin key]
```

La réponse contient l'état d'intégrité global de l'indexeur, le dernier appel de l'indexeur (ou celui en cours), ainsi que l'historique des appels récents de l'indexeur.

```
{  
    "status": "running",  
    "lastResult": {  
        "status": "success",  
        "errorMessage": null,  
        "startTime": "2018-11-26T03:37:18.853Z",  
        "endTime": "2018-11-26T03:37:19.012Z",  
        "errors": [],  
        "itemsProcessed": 11,  
        "itemsFailed": 0,  
        "initialTrackingState": null,  
        "finalTrackingState": null  
    },  
    "executionHistory": [  
        {  
            "status": "success",  
            "errorMessage": null,  
            "startTime": "2018-11-26T03:37:18.853Z",  
            "endTime": "2018-11-26T03:37:19.012Z",  
            "errors": [],  
            "itemsProcessed": 11,  
            "itemsFailed": 0,  
            "initialTrackingState": null,  
            "finalTrackingState": null  
        }]  
    }]
```

L'historique d'exécution contient les 50 exécutions les plus récentes, classées par ordre chronologique inverse (la plus récente en premier).

Notez qu'il existe deux valeurs d'état différentes. L'état de niveau supérieur s'applique à l'indexeur lui-même. Un état d'indexeur **En cours** signifie que l'indexeur est correctement configuré et disponible pour être exécuté mais qu'il n'est pas actuellement en cours d'exécution.

Chaque exécution de l'indexeur possède également un état qui indique si cette exécution spécifique est en cours d'exécution (**running**) ou déjà terminée avec l'état **success**, **transientFailure** ou **persistentFailure**.

Lorsque l'on réinitialise un indexeur pour actualiser son état de suivi des modifications, une entrée distincte indiquant l'état **Réinitialiser** est ajoutée à l'historique des exécutions.

Pour plus d'informations sur les codes d'état et les données de surveillance de l'indexeur, consultez [GetIndexerStatus](#).

## Surveiller à l'aide du Kit de développement logiciel (SDK) .NET

Vous pouvez configurer la planification d'un indexeur à l'aide du Kit de développement logiciel (SDK) .NET Recherche cognitive Azure. Pour ce faire, ajoutez la propriété **schedule** lors de la création ou de la mise à jour d'un indexeur.

L'exemple C# suivant écrit des informations sur l'état d'un indexeur et les résultats de son exécution la plus récente (ou en cours) à la console.

```

static void CheckIndexerStatus(Indexer indexer, SearchServiceClient searchService)
{
    try
    {
        IndexerExecutionInfo execInfo = searchService.Indexers.GetStatus(indexer.Name);

        Console.WriteLine("Indexer has run {0} times.", execInfo.ExecutionHistory.Count);
        Console.WriteLine("Indexer Status: " + execInfo.Status.ToString());

        IndexerExecutionResult result = execInfo.LastResult;

        Console.WriteLine("Latest run");
        Console.WriteLine("  Run Status: {0}", result.Status.ToString());
        Console.WriteLine("  Total Documents: {0}, Failed: {1}", result.ItemCount, result.FailedItemCount);

        TimeSpan elapsed = result.EndTime.Value - result.StartTime.Value;
        Console.WriteLine("  StartTime: {0:T}, EndTime: {1:T}, Elapsed: {2:t}", result.StartTime.Value,
result.EndTime.Value, elapsed);

        string errorMsg = (result.ErrorMessage == null) ? "none" : result.ErrorMessage;
        Console.WriteLine("  ErrorMessage: {0}", errorMsg);
        Console.WriteLine("  Document Errors: {0}, Warnings: {1}\n", result.Errors.Count,
result.Warnings.Count);
    }
    catch (Exception e)
    {
        // Handle exception
    }
}

```

Sur la console, vous obtenez un résultat semblable à ceci :

```

Indexer has run 18 times.
Indexer Status: Running
Latest run
Run Status: Success
Total Documents: 7, Failed: 0
StartTime: 10:02:46 PM, EndTime: 10:02:47 PM, Elapsed: 00:00:01.0990000
ErrorMessage: none
Document Errors: 0, Warnings: 0

```

Notez qu'il existe deux valeurs d'état différentes. L'état de niveau supérieur est l'état de l'indexeur lui-même. Un état d'indexeur **En cours** signifie que l'indexeur est correctement configuré et disponible pour être exécuté mais qu'il n'est pas actuellement en cours d'exécution.

Chaque exécution de l'indexeur possède également un état qui indique si cette exécution spécifique est en cours d'exécution (**En cours**) ou déjà terminée avec l'état **Réussite** ou **TransientError**.

Lorsque l'on réinitialise un indexeur pour actualiser son état de suivi des modifications, une entrée distincte indiquant l'état **Réinitialiser** est ajoutée à l'historique.

Pour plus d'informations sur les codes d'état et les données de surveillance de l'indexeur, consultez [GetIndexerStatus](#) dans l'API REST.

Il est possible de récupérer les détails sur les erreurs ou les avertissements spécifiques aux documents en énumérant les listes `IndexerExecutionResult.Errors` et `IndexerExecutionResult.Warnings`.

Pour plus d'informations sur les classes du SDK .NET permettant de surveiller les indexeurs, consultez [IndexerExecutionInfo](#) et [IndexerExecutionResult](#).

# Comment planifier des indexeurs dans la Recherche cognitive Azure

04/10/2020 • 11 minutes to read • [Edit Online](#)

Un indexeur s'exécute normalement une fois, immédiatement après sa création. Vous pouvez l'exécuter à nouveau à la demande avec le portail, l'API REST ou le kit SDK .NET. Vous pouvez également configurer un indexeur pour qu'il s'exécute à intervalles périodiques.

Certaines situations dans lesquelles la planification de l'indexeur est utilisée :

- Les données sources changent au fil du temps, et vous souhaitez que les indexeurs Recherche cognitive Azure traitent automatiquement les données modifiées.
- L'index est renseigné à partir de plusieurs sources de données et vous souhaitez vous assurer que les indexeurs s'exécutent à des moments différents afin de réduire les conflits.
- Les données sources sont très volumineuses et vous souhaitez répartir le traitement de l'indexeur dans le temps. Pour plus d'informations sur l'indexation de grands volumes de données, consultez [Comment indexer des grands volumes de données dans la Recherche cognitive Azure](#).

Le planificateur est une fonctionnalité intégrée à la Recherche cognitive Azure. Vous ne pouvez pas utiliser un planificateur externe pour contrôler les indexeurs de recherche.

## Définir les propriétés de la planification

Une planification d'indexeur a deux propriétés :

- **Interval**, qui définit l'intervalle de temps entre les deux exécutions d'indexeur planifiées. Le plus petit intervalle autorisé est de 5 minutes, et le plus grand est de 24 heures.
- **Heure de début (UTC)** , qui indique la première heure à laquelle l'indexeur doit être exécuté.

Vous pouvez spécifier une planification lors de la création de l'indexeur, ou en mettant à jour les propriétés de l'indexeur ultérieurement. Les planifications de l'indexeur peuvent être définies à l'aide du [Portail](#), de l'[API REST](#) ou du [SDK .NET](#).

L'indexeur ne peut s'exécuter qu'une seule fois simultanément. Si un indexeur est déjà en cours d'exécution au moment de sa deuxième exécution planifiée, celle-ci est différée jusqu'à la prochaine date planifiée.

Pour être plus clair, prenons un exemple. Supposons que nous configurons une planification de l'indexeur avec un **intervalle** par heure et une **heure de début** le 1er juin 2019 à 08:00:00 UTC. Voici ce qui peut se produire lorsque l'exécution de l'indexeur prend plus d'une heure :

- La première exécution de l'indexeur commence à ou autour du 1er juin 2019 à 8h00 UTC. Supposons que cette exécution prend 20 minutes (ou en tout cas, moins de 1 heure).
- La deuxième exécution de l'indexeur commence à ou autour du 1er juin 2019 à 9h00 UTC. Supposons que cette exécution dure 70 minutes (plus d'une heure), et qu'elle ne se terminera pas avant 10h10 UTC.
- La troisième exécution est planifiée pour démarrer à 10h00 UTC, mais à ce moment l'exécution précédente est toujours en cours. Cette exécution planifiée est donc ignorée. La prochaine exécution de l'indexeur ne va pas démarrer avant 11h00 UTC.

## NOTE

Si un indexeur est défini sur une certaine planification, mais échoue à plusieurs reprises sur le même document chaque fois qu'il s'exécute, l'indexeur commence à s'exécuter à un intervalle moins fréquent (jusqu'à un maximum d'au moins une fois toutes les 24 heures) jusqu'à ce qu'il progresse correctement à nouveau. Si vous pensez avoir résolu le problème qui provoquait le blocage de l'indexeur à un moment donné, vous pouvez effectuer une exécution à la demande de l'indexeur, et en cas de progression, l'indexeur reprend son intervalle de planification défini.

## Planification dans le portail

L'Assistant Importation de données dans le portail vous permet de définir la planification pour un indexeur lors de sa création. Le paramètre de planification par défaut est **Par heure**, ce qui signifie que l'indexeur s'exécute une fois qu'il est créé et s'exécute à nouveau toutes les heures par la suite.

Vous pouvez modifier le paramètre de planification sur **Une fois** si vous ne souhaitez pas que l'indexeur s'exécute à nouveau automatiquement, ou sur **Une fois par jour** pour l'exécuter une fois par jour. Définissez-le sur **Personnalisé** si vous souhaitez spécifier un intervalle différent ou une heure de début suivante spécifique.

Lorsque vous définissez la planification sur **Personnalisée**, les champs qui s'affichent vous permettent de spécifier l'**intervalle** et l'**heure de début (UTC)**. Le plus court intervalle de temps autorisé est de 5 minutes, et le plus long est de 1 440 minutes (24 heures).

**Import data** X

---

\* Connect to your data   Add cognitive search (Optional)   \* Customize target index   [\\* Create an indexer](#)

**Indexer**

\* Name  ✓

Schedule i Once Hourly Daily Custom

\* Interval (minutes)  ✓

\* Start time (UTC) i  [calendrier]

✓ Change tracking automatically configured with a high watermark policy.

Track deletions i

Description

▼ Advanced options

---

[Previous: Customize target index](#) Submit

Une fois un indexeur créé, vous pouvez modifier les paramètres de planification à l'aide du panneau Modifier

de l'indexeur. Les champs de planification sont les mêmes que ceux dans l'Assistant Importation de données.

**Edit**

---

Index  
test-cosmosdb-index

Data Source  
test-cosmos-src

Schedule   
Once   Hourly   Daily   **Custom**

\* Interval (minutes)  
240

\* Start time (UTC)   
2019-06-01  8:00:00 AM

---

Advanced options >

---

Description

---

**OK**

## Planification à l'aide des API REST

Vous pouvez configurer la planification d'un indexeur à l'aide de l'API REST. Pour ce faire, ajoutez la propriété **schedule** lors de la création ou de la mise à jour de l'indexeur. L'exemple ci-dessous montre une requête PUT mettant à jour l'indexeur existant :

```

PUT https://myservice.search.windows.net/indexers/myindexer?api-version=2020-06-30
Content-Type: application/json
api-key: admin-key

{
    "dataSourceName" : "myazuresqldatasource",
    "targetIndexName" : "target index name",
    "schedule" : { "interval" : "PT10M", "startTime" : "2015-01-01T00:00:00Z" }
}

```

Le paramètre **interval** est obligatoire. Il correspond à la durée entre le début de deux exécutions consécutives de l'indexeur. L'intervalle minimal autorisé est de 5 minutes, l'intervalle maximal autorisé est d'une journée. Il doit être formaté en tant que valeur « `dayTimeDuration` » XSD (un sous-ensemble limité d'une valeur de [durée ISO 8601](#)). Le modèle est le suivant : `P(nD)(T(nH)(nM))`. Exemples : `PT15M` toutes les 15 minutes, `PT2H` toutes les deux heures.

Le paramètre **startTime** facultatif indique quand les exécutions planifiées doivent commencer. S'il est omis, l'heure UTC actuelle est utilisée. Cette heure peut être passée, auquel cas la première exécution est planifiée comme si l'indexeur s'exécutait en continu depuis l'**heure de début** originale.

Vous pouvez également exécuter un indexeur à la demande à tout moment à l'aide de l'appel `Exécuter l'indexeur`. Pour plus d'informations sur l'exécution des indexeurs et la définition des planifications d'indexeur, consultez [Exécuter l'indexeur](#), [Obtention d'indexeur](#) et [Mise à jour d'un indexeur](#) dans la référence d'API REST.

## Planification à l'aide du kit de développement logiciel (SDK) REST

Vous pouvez configurer la planification d'un indexeur à l'aide du Kit de développement logiciel (SDK) .NET Recherche cognitive Azure. Pour ce faire, ajoutez la propriété **schedule** lors de la création ou de la mise à jour d'un indexeur.

L'exemple C# suivant crée un indexeur à l'aide d'une source de données et d'un index prédefinis, et définit sa planification pour qu'il s'exécute une fois par jour dans 30 minutes à partir de maintenant :

```

Indexer indexer = new Indexer(
    name: "azure-sql-indexer",
    dataSourceName: dataSource.Name,
    targetIndexName: index.Name,
    schedule: new IndexingSchedule(
        TimeSpan.FromDays(1),
        new DateTimeOffset(DateTime.UtcNow.AddMinutes(30))
    )
);
await searchService.Indexers.CreateOrUpdateAsync(indexer);

```

Si le paramètre **planification** est omis, l'indexeur s'exécute uniquement une fois immédiatement après sa création.

Le paramètre **startTime** peut être défini sur une heure passée. Dans ce cas, la première exécution est planifiée comme si l'indexeur s'exécutait en continu depuis l'**heure de début** donnée.

La planification est définie à l'aide de la classe [IndexingSchedule](#). Le constructeur [IndexingSchedule](#) requiert un paramètre **interval** spécifié à l'aide d'un objet [TimeSpan](#). La plus petite valeur d'intervalle autorisée est de 5 minutes, et la plus grande est de 24 heures. Le second paramètre **startTime**, spécifié comme un objet [DateTimeOffset](#), est facultatif.

Le kit de développement logiciel (SDK) .NET vous permet de contrôler les opérations d'indexeur à l'aide de la classe [SearchServiceClient](#) et de la propriété [Indexers](#), qui implémente les méthodes à partir de l'interface [IIndexersOperations](#).

Vous pouvez exécuter un indexeur à la demande à tout moment en utilisant l'une des méthodes [Run](#), [RunAsync](#) ou [RunWithHttpMessagesAsync](#).

Pour plus d'informations sur la création, la mise à jour et l'exécution des indexeurs, consultez [IIndexersOperations](#).

# Mappages de champs et transformations à l'aide d'indexeurs Recherche cognitive Azure

04/10/2020 • 19 minutes to read • [Edit Online](#)



Lorsque vous utilisez des indexeurs Recherche cognitive Azure, il se peut que les données d'entrée ne correspondent pas tout à fait au schéma de votre index cible. Dans ce cas, vous pouvez utiliser des **mappages de champs** pour remodeler vos données pendant le processus d'indexation.

Quelques situations où les mappages de champs sont utiles :

- Votre source de données a un champ appelé `_id`, mais la Recherche cognitive Azure n'autorise pas les noms de champs commençant par un trait de soulignement. Un mappage de champ vous permet de renommer un champ.
- Vous souhaitez remplir plusieurs champs de l'index à partir des données de la même source de données. Par exemple, vous souhaiterez peut-être appliquer différents analyseurs à ces champs.
- Vous voulez remplir un champ d'index avec des données provenant de plusieurs sources de données, lesquelles utilisent des noms de champs différents.
- Vous avez besoin d'encoder ou de décoder vos données en Base64. Les mappages de champs prennent en charge plusieurs **fonctions de mappage**, y compris les fonctions d'encodage et de décodage en Base64.

## NOTE

Les mappages de champs dans les indexeurs sont un moyen simple de mapper des champs de données à des champs d'index, avec une certaine possibilité de conversion de données simples. Les données plus complexes devront peut-être être prétraitées pour être converties dans un format propice à l'indexation. L'une des options que vous pouvez envisager est [Azure Data Factory](#).

## Configurer des mappages de champs

Un mappage de champs se compose de trois parties :

1. Un `sourceFieldName`, qui représente un champ de votre source de données. Cette propriété est requise.
2. Un `targetFieldName` facultatif, qui représente un champ de votre index de recherche. Si omis, le nom de la source de données est utilisé.
3. Une `mappingFunction` facultative, qui peut transformer vos données à l'aide d'une des fonctions prédéfinies. Celle-ci peut être appliquée sur les mappages de champs d'entrée et de sortie. La liste complète des fonctions est présentée [ci-dessous](#).

Les mappages de champs sont ajoutés au tableau `fieldMappings` dans la définition de l'indexeur.

#### NOTE

Si aucun mappage de champs n'est ajouté, les indexeurs supposent que les champs de source de données doivent être mappés à des champs d'index portant le même nom. L'ajout d'un mappage de champs supprime ces mappages de champs par défaut pour les champs source et cible. Certains indexeurs, comme [l'indexeur de stockage d'objets blob](#), ajoutent des mappages de champs par défaut pour le champ de clé d'index.

## Mapper des champs avec l'API REST

Vous pouvez ajouter des mappages de champs lors de la création d'un indexeur avec la requête d'API [Créer un indexeur](#). Vous pouvez gérer les mappages de champs d'un indexeur existant avec la requête d'API [Mise à jour d'un indexeur](#).

Par exemple, voici comment mapper un champ source à un champ cible avec un nom différent :

```
PUT https://[service name].search.windows.net/indexers/myindexer?api-version=[api-version]
Content-Type: application/json
api-key: [admin key]
{
  "dataSourceName" : "mydatasource",
  "targetIndexName" : "myindex",
  "fieldMappings" : [ { "sourceFieldName" : "_id", "targetFieldName" : "id" } ]
}
```

Un champ source peut être référencé dans plusieurs mappages de champs. L'exemple suivant montre comment dupliquer (fork) un champ, en copiant le même champ source dans deux champs d'index différents :

```
"fieldMappings" : [
  { "sourceFieldName" : "text", "targetFieldName" : "textStandardEnglishAnalyzer" },
  { "sourceFieldName" : "text", "targetFieldName" : "textSoundexAnalyzer" }
]
```

#### NOTE

Azure Search utilise une comparaison qui ne respecte pas la casse pour résoudre les noms de champ et de fonction dans les mappages de champs. C'est pratique (vous n'avez besoin de faire attention à la casse), mais cela signifie que votre source de données et votre index ne peuvent pas comporter des champs qui diffèrent uniquement par la casse.

## Mapper des champs avec le Kit de développement logiciel (SDK) .NET

Vous définissez des mappages de champs dans le Kit de développement logiciel (SDK) .NET à l'aide de la classe [FieldMapping](#), qui possède les propriétés `SourceFieldName` et `TargetFieldName`, ainsi qu'une référence `MappingFunction` en option.

Vous pouvez spécifier des mappages de champs lors de la construction de l'indexeur, ou ultérieurement, en définissant directement la propriété `Indexer.FieldMappings`.

L'exemple C# suivant définit les mappages de champs lors de la construction d'un indexeur.

```

List<FieldMapping> map = new List<FieldMapping> {
    // removes a leading underscore from a field name
    new FieldMapping("_custId", "custId"),
    // URL-encodes a field for use as the index key
    new FieldMapping("docPath", "docId", FieldMappingFunction.Base64Encode() )
};

Indexer sqlIndexer = new Indexer(
    name: "azure-sql-indexer",
    dataSourceName: sqlDataSource.Name,
    targetIndexName: index.Name,
    fieldMappings: map,
    schedule: new IndexingSchedule(TimeSpan.FromDays(1)));

await searchService.Indexers.CreateOrUpdateAsync(indexer);

```

## Fonctions de mappage de champs

Une fonction de mappage de champ transforme le contenu d'un champ avant son stockage dans l'index. Les fonctions de mappage actuellement prises en charge sont les suivantes :

- [base64Encode](#)
- [base64Decode](#)
- [extractTokenAtPosition](#)
- [jsonArrayToStringCollection](#)
- [urlEncode](#)
- [urlDecode](#)

### Fonction base64Encode

Exécute l'encodage Base64 *sécurisé pour les URL* de la chaîne d'entrée. Suppose que l'entrée est encodée en UTF-8.

#### Exemple de recherche d'une clé de document

Seuls les caractères sécurisés pour les URL peuvent apparaître dans une clé de document Recherche cognitive Azure (car les clients doivent pouvoir traiter le document à l'aide de l'[API de recherche](#)). Si le champ source de votre clé contient des caractères non sécurisés pour les URL, vous pouvez utiliser la fonction `base64Encode` pour les convertir au moment de l'indexation. Cependant, une clé de document (avant et après la conversion) ne doit pas excéder 1 024 caractères.

Une fois que vous avez récupéré la clé encodée au moment de la recherche, vous pouvez utiliser la fonction `base64Decode` pour obtenir la valeur de clé d'origine, et l'utiliser pour récupérer le document source.

```

"fieldMappings" : [
    {
        "sourceFieldName" : "SourceKey",
        "targetFieldName" : "IndexKey",
        "mappingFunction" : {
            "name" : "base64Encode",
            "parameters" : { "useHttpServerUtilityUrlTokenEncode" : false }
        }
    }
]

```

#### Exemple : conserver les valeurs d'origine

L'[indexeur de stockage d'objets blob](#) ajoute automatiquement un mappage de champs à partir de `metadata_storage_path`, l'URI de l'objet blob, au champ de clé d'index si aucun mappage de champs n'est spécifié. Cette valeur est encodée en Base64 afin d'être utilisée en toute sécurité comme clé de document

Recherche cognitive Azure. L'exemple suivant montre comment mapper simultanément une *version* de `metadata_storage_path` en codage Base64 sécurisée pour les URL à un champ `index_key` et conserver la valeur d'origine dans un champ `metadata_storage_path` :

```
"fieldMappings": [
  {
    "sourceFieldName": "metadata_storage_path",
    "targetFieldName": "metadata_storage_path"
  },
  {
    "sourceFieldName": "metadata_storage_path",
    "targetFieldName": "index_key",
    "mappingFunction": {
      "name": "base64Encode"
    }
  }
]
```

Si vous n'incluez aucune propriété de paramètre pour votre fonction de mappage, la valeur par défaut est `{"useHttpServerUtilityUrlTokenEncode" : true}`.

Recherche cognitive Azure prend en charge deux encodages en Base64. Vous devez utiliser les mêmes paramètres lors de l'encodage et du décodage du même champ. Pour décider quels paramètres utiliser, consultez les [options relatives à l'encodage base64](#).

### Fonction base64Decode

Effectue le décodage en Base64 de la chaîne d'entrée. L'entrée est considérée comme une chaîne encodée en Base64 et *sécurisée pour les URL*.

#### Exemple : décodage de métadonnées de blob ou d'URL

Votre source de données peut contenir des chaînes encodées en Base64, telles que des chaînes de métadonnées de blob ou des URL web, que vous voulez rendre disponibles pour des recherches sous la forme de texte brut. Vous pouvez utiliser la fonction `base64Decode` pour transformer les données encodées en chaînes normales lors du remplissage de votre index de recherche.

```
"fieldMappings" : [
  {
    "sourceFieldName" : "Base64EncodedMetadata",
    "targetFieldName" : "SearchableMetadata",
    "mappingFunction" : {
      "name" : "base64Decode",
      "parameters" : { "useHttpServerUtilityUrlTokenDecode" : false }
    }
  }
]
```

Si vous n'incluez aucune propriété de paramètre, la valeur par défaut est `{"useHttpServerUtilityUrlTokenEncode" : true}`.

Recherche cognitive Azure prend en charge deux encodages en Base64. Vous devez utiliser les mêmes paramètres lors de l'encodage et du décodage du même champ. Pour décider quels paramètres utiliser, reportez-vous aux [options relatives à l'encodage base64](#).

### Options d'encodage en Base64

La Recherche cognitive Azure prend en charge l'encodage en base64 normal et sécurisé pour les URL. Une chaîne encodée en base64 lors de l'indexation devra être décodée ultérieurement avec les mêmes options d'encodage. Dans le cas contraire, le résultat ne correspondra pas à la version d'origine.

Si les paramètres `useHttpServerUtilityUrlTokenEncode` ou `useHttpServerUtilityUrlTokenDecode` d'encodage et de décodage, respectivement, sont définis sur `true`, `base64Encode` se comporte comme `HttpServerUtility.UrlTokenEncode` et `base64Decode` se comporte comme `HttpServerUtility.UrlTokenDecode`.

#### WARNING

Si `base64Encode` est utilisé pour générer des valeurs de clé, `useHttpServerUtilityUrlTokenEncode` doit être défini sur `true`. Seul l'encodage en base64 sécurisé pour les URL peut être utilisé pour les valeurs de clés. Consultez [Règles de nommage \(Recherche cognitive Azure\)](#) pour obtenir l'ensemble des restrictions appliquées aux caractères des valeurs de clé.

Les bibliothèques .NET dans la Recherche cognitive Azure utilisent l'intégralité de .NET Framework qui fournit un encodage intégré. Les options `useHttpServerUtilityUrlTokenEncode` et `useHttpServerUtilityUrlTokenDecode` tirent parti de cette fonctionnalité intégrée. Si vous utilisez .NET Core ou une autre infrastructure, nous vous recommandons de définir ces options sur `false` et d'appeler directement les fonctions d'encodage et de décodage de votre infrastructure.

Le tableau suivant compare les différents encodages base64 de la chaîne `00>00?00`. Pour déterminer quel traitement supplémentaire s'avère nécessaire (le cas échéant) pour vos fonctions base64, appliquez votre fonction d'encodage de bibliothèque à la chaîne `00>00?00` et comparez le résultat obtenu au résultat attendu `MDA-MDA_MDA`.

ENCODAGE	RÉSULTAT D'ENCODAGE BASE64	TRAITEMENT SUPPLÉMENTAIRE APRÈS L'ENCODAGE DE BIBLIOTHÈQUE	TRAITEMENT SUPPLÉMENTAIRE AVANT L'ENCODAGE DE BIBLIOTHÈQUE
Base64 avec remplissage	<code>MDA+MDA/MDA=</code>	Utiliser des caractères sécurisés pour les URL et supprimer le remplissage	Utiliser des caractères base64 standard et ajouter le remplissage
Base64 sans remplissage	<code>MDA+MDA/MDA</code>	Utiliser des caractères sécurisés pour les URL	Utiliser des caractères base64 standard
Base64 sécurisé pour les URL avec remplissage	<code>MDA-MDA_MDA=</code>	Supprimer le remplissage	Ajouter le remplissage
Base64 sécurisé pour les URL sans remplissage	<code>MDA-MDA_MDA</code>	None	None

#### Fonction extractTokenAtPosition

Divise un champ de chaîne en utilisant le séparateur spécifié et récupère le jeton à la position spécifiée dans le fractionnement résultant.

Cette fonction utilise les paramètres suivants :

- `delimiter` : une chaîne à utiliser comme séparateur lors du fractionnement de la chaîne d'entrée.
- `position` : une position entière à base zéro du jeton à choisir une fois la chaîne d'entrée fractionnée.

Par exemple, si l'entrée est `Jane Doe`, que le `delimiter` est `" "` (espace) et que la `position` est 0, le résultat est `Jane`; si la `position` est 1, le résultat est `Doe`. Si la position fait référence à un jeton qui n'existe pas, une erreur est renvoyée.

#### Exemple : extraction d'un nom

Votre source de données contient un champ `PersonName` et vous souhaitez l'indexer en tant que deux champs `FirstName` et `LastName` distincts. Vous pouvez utiliser cette fonction pour fractionner l'entrée en utilisant

l'espace comme séparateur.

```
"fieldMappings" : [
  {
    "sourceFieldName" : "PersonName",
    "targetFieldName" : "FirstName",
    "mappingFunction" : { "name" : "extractTokenAtPosition", "parameters" : { "delimiter" : " ", "position" : 0 } }
  },
  {
    "sourceFieldName" : "PersonName",
    "targetFieldName" : "LastName",
    "mappingFunction" : { "name" : "extractTokenAtPosition", "parameters" : { "delimiter" : " ", "position" : 1 } }
  }
]
```

## Fonction jsonArrayToStringCollection

Transforme une chaîne formatée en tant que tableau de chaînes JSON en un tableau de chaînes utilisable pour remplir un champ `Collection(Edm.String)` dans l'index.

Par exemple, si la chaîne d'entrée est `["red", "white", "blue"]`, le champ cible de type `Collection(Edm.String)` est rempli avec les valeurs `red`, `white` et `blue`. Pour les valeurs d'entrée qui ne peuvent pas être analysées en tant que tableaux de chaînes JSON, une erreur est renvoyée.

### Exemple : remplissage d'une collection avec des données relationnelles

Microsoft Azure SQL Database n'inclut aucun type de données intégré qui se mappe naturellement aux champs `Collection(Edm.String)` dans Recherche cognitive Azure. Pour remplir les champs de la collection de chaînes, vous pouvez prétraiter votre source de données en tant que tableau de chaînes JSON, puis utiliser la fonction de mappage `jsonArrayToStringCollection`.

```
"fieldMappings" : [
  {
    "sourceFieldName" : "tags",
    "mappingFunction" : { "name" : "jsonArrayToStringCollection" }
  }
]
```

## fonction urlEncode

Cette fonction peut être utilisée pour encoder une chaîne de sorte qu'elle soit « URL safe ». Quand elle est utilisée avec une chaîne qui contient des caractères qui ne sont pas autorisés dans une URL, cette fonction convertit ces caractères « non sécurisés » en équivalents d'entité de caractère. Cette fonction utilise le format d'encodage UTF-8.

### Exemple de recherche d'une clé de document

La fonction `urlEncode` peut être utilisée comme alternative à la fonction `base64Encode`, si seuls les caractères non sécurisés d'URL doivent être convertis, tout en conservant les autres caractères tels quels.

Par exemple, si la chaîne d'entrée est `<hello>`, le champ cible de type `(Edm.String)` est rempli avec la valeur `%3chello%3e`

Une fois que vous avez récupéré la clé encodée au moment de la recherche, vous pouvez utiliser la fonction `urlDecode` pour obtenir la valeur de clé d'origine, et l'utiliser pour récupérer le document source.

```
"fieldMappings" : [
  {
    "sourceFieldName" : "SourceKey",
    "targetFieldName" : "IndexKey",
    "mappingFunction" : {
      "name" : "urlEncode"
    }
  }
]
```

### **fonction urlDecode**

Cette fonction convertit une chaîne encodée en URL en chaîne décodée à l'aide du format d'encodage UTF-8.

#### **Exemple : décodage de métadonnées de blob**

Certains clients de stockage Azure encodent automatiquement les métadonnées Blob si elles contiennent des caractères non-ASCII. Toutefois, si vous souhaitez effectuer des recherches dans ces métadonnées (en texte brut), vous pouvez utiliser la fonction `urlDecode` pour réactiver les données encodées dans des chaînes normales lors du remplissage de votre index de recherche.

```
"fieldMappings" : [
  {
    "sourceFieldName" : "UrlEncodedMetadata",
    "targetFieldName" : "SearchableMetadata",
    "mappingFunction" : {
      "name" : "urlDecode"
    }
  }
]
```

### **Fonction fixedLengthEncode**

Cette fonction convertit une chaîne de n'importe quelle longueur en chaîne à longueur fixe.

#### **Exemple - mapper des clés de document trop longues**

Lorsque vous rencontrez une erreur indiquant que la clé du document excède 1 024 caractères, cette fonction peut être appliquée pour réduire la longueur de la clé.

```
"fieldMappings" : [
  {
    "sourceFieldName" : "metadata_storage_path",
    "targetFieldName" : "your key field",
    "mappingFunction" : {
      "name" : "fixedLengthEncode"
    }
  }
]
```

# Configurer une connexion d'indexeur à une source de données à l'aide d'une identité managée

04/10/2020 • 4 minutes to read • [Edit Online](#)

Un [indexeur](#) dans Recherche cognitive Azure est un robot qui permet d'extraire des données de votre source de données vers Recherche cognitive Azure. Un indexeur obtient une connexion à la source de données à partir de l'objet source de données que vous créez. L'objet source de données comprend généralement les informations d'identification de la source de données cible. Par exemple, l'objet source de données peut inclure une clé de compte Stockage Azure si vous souhaitez indexer les données d'un conteneur de stockage blob.

Dans de nombreux cas, le fait de fournir des informations d'identification directement dans l'objet source de données n'est pas un problème, mais quelques difficultés peuvent se présenter :

- Comment conserver les informations d'identification en toute sécurité dans mon code qui crée l'objet source des données ?
- Si la clé ou le mot de passe de mon compte sont compromis et que je dois les modifier, je dois mettre à jour mes objets sources de données avec la nouvelle clé ou le nouveau mot de passe du compte afin que mon indexeur puisse de nouveau se connecter à la source de données.

Ces problèmes peuvent être résolus en configurant votre connexion à l'aide d'une identité managée.

## Utilisation des identités managées

[Identités managées](#) est une fonctionnalité qui fournit aux services Azure une identité gérée automatiquement dans Azure Active Directory (Azure AD). Vous pouvez utiliser cette fonctionnalité dans Recherche cognitive Azure pour créer un objet source de données doté d'une chaîne de connexion qui n'inclut aucune information d'identification. Au lieu de cela, votre service de recherche disposera de l'accès à la source de données via le contrôle d'accès en fonction du rôle (RBAC).

En configurant une source de données à l'aide d'une identité managée, vous pouvez modifier vos informations d'identification de source de données le cas échéant : vos indexeurs pourront toujours se connecter à la source de données. Vous pouvez également créer des objets sources de données dans votre code sans avoir à inclure de clé de compte ni à utiliser Key Vault pour récupérer une clé de compte.

## Limites

Les sources de données suivantes prennent en charge la configuration d'une connexion d'indexeur à l'aide d'identités managées.

- [Stockage Blob Azure, Azure Data Lake Storage Gen2 \(préversion\), Stockage Table Azure](#)
- [Azure Cosmos DB](#)
- [Azure SQL Database](#)

Actuellement, les fonctionnalités suivantes ne prennent pas en charge l'utilisation des identités managées pour configurer la connexion :

- Base de connaissances
- Compétences personnalisées

## Étapes suivantes

En savoir plus sur la configuration d'une connexion d'indexeur à l'aide d'identités managées :

- [Stockage Blob Azure, Azure Data Lake Storage Gen2 \(préversion\), Stockage Table Azure](#)
- [Azure Cosmos DB](#)
- [Azure SQL Database](#)

# Configurer une connexion à un compte Stockage Azure à l'aide d'une identité managée (préversion)

04/10/2020 • 9 minutes to read • [Edit Online](#)

## IMPORTANT

La prise en charge de la configuration d'une connexion à une source de données à l'aide d'une identité managée fait actuellement l'objet d'une préversion publique. Les fonctionnalités en préversion sont fournies sans contrat de niveau de service et ne sont pas recommandées pour les charges de travail de production.

Cette page explique comment configurer une connexion d'indexeur à un compte Stockage Azure à l'aide d'une identité managée au lieu de fournir des informations d'identification dans la chaîne de connexion de l'objet source de données.

Avant d'en apprendre plus sur cette fonctionnalité, il est recommandé de comprendre ce qu'est un indexeur et savoir comment configurer un indexeur pour votre source de données. Pour plus d'informations, consultez les liens suivants :

- [Présentation de l'indexeur](#)
- [Indexeur de blobs Azure](#)
- [Indexeur Azure Data Lake Storage Gen2](#)
- [Indexeur de tables Azure](#)

## Configurer la connexion

### 1 – Activer l'identité managée affectée par le système

Quand une identité managée affectée par le système est activée, Azure crée une identité pour votre service de recherche qui peut être utilisée pour vous authentifier auprès d'autres services Azure au sein du même locataire et du même abonnement. Vous pouvez ensuite utiliser cette identité dans les attributions de contrôle d'accès en fonction du rôle (RBAC) qui autorisent l'accès aux données pendant l'indexation.

Azure Cognitive Search Service Identity settings page. The 'Identity' section is highlighted with a red box. The 'Status' switch is set to 'On'. The 'Save' button is highlighted with a red box.

Après avoir sélectionné Enregistrer, vous verrez un ID d'objet qui a été attribué à votre service de recherche.

Azure Cognitive Search Service Identity settings page. The 'Object ID' field contains the value '1da4d008-e5ce-4a03-8041-4694e5059d36', which is highlighted with a red box.

## 2 – Ajouter une attribution de rôle

Au cours de cette étape, vous allez accorder à votre service Recherche cognitive Azure l'autorisation de lire les données de votre compte de stockage.

1. Dans le portail Azure, accédez au compte de stockage qui contient les données que vous souhaitez indexer.
2. Sélectionnez Contrôle d'accès (IAM)
3. Sélectionnez Ajouter, puis Ajouter une attribution de rôle.

The screenshot shows the 'Access control (IAM)' blade for a storage account. On the left, there's a sidebar with various options like Overview, Activity log, and Access control (IAM). The 'Access control (IAM)' option is selected and highlighted with a red box. In the main content area, there's a 'Check access' section and a 'Role assignments' tab. A red box highlights the '+ Add' button under the 'Role assignments' tab. To the right, there are two cards: 'Add a role assignment' and 'View deny assignments', each with a 'Learn more' link.

4. Sélectionnez les rôles appropriés en fonction du type de compte de stockage que vous souhaitez indexer :
  - a. Stockage Blob Azure nécessite que vous ajoutiez votre service de recherche au rôle **Lecteur des données blob du stockage**.
  - b. Azure Data Lake Storage Gen2 nécessite que vous ajoutiez votre service de recherche au rôle **Lecteur des données blob du stockage**.
  - c. Stockage Table Azure nécessite que vous ajoutiez votre service de recherche au rôle **Lecteur et accès aux données**.
5. Laissez Attribuer l'accès à sur Utilisateur, groupe ou principal de service Azure AD.
6. Recherchez votre service de recherche, sélectionnez-le, puis sélectionnez Enregistrer.

Exemple pour Stockage Blob Azure et Azure Data Lake Storage Gen2 :

The screenshot shows the 'Add role assignment' dialog. It has a 'Role' dropdown set to 'Storage Blob Data Reader' (highlighted with a red box). Below it, there's a 'Assign access to' dropdown set to 'Azure AD user, group, or service principal'. A 'Select' button is followed by a search bar containing 'azure-cognitive-search'. The 'Selected members' section shows a single entry: 'azure-cognitive-search-service' (also highlighted with a red box). At the bottom, there are 'Save' and 'Discard' buttons.

Exemple pour Stockage Table Azure :

### 3 – Créer la source de données

L'[API REST](#), le portail Azure et le [Kit de développement logiciel \(SDK\) .NET](#) prennent également en charge la chaîne de connexion des identités managées. Voici un exemple de création d'une source de données pour indexer des données à partir d'un compte de stockage à l'aide de l'[API REST](#) et d'une chaîne de connexion d'identité gérée. Le format de chaîne de connexion d'identité managée est le même pour l'API REST, le kit de développement logiciel (SDK) .NET et le portail Azure.

Lors de l'indexation d'un compte de stockage, la source de données doit avoir les propriétés requises suivantes :

- **name** est le nom unique de la source de données au sein de votre service de recherche.
- **type**
  - Stockage Blob Azure : `azureblob`
  - Stockage Table Azure : `azuretable`
  - Azure Data Lake Storage Gen2 : **type** est fourni une fois que vous vous êtes inscrit à la préversion à l'aide de [ce formulaire](#).
- **credentials**
  - Le format d'**informations d'identification** est différent selon que vous utilisez ou non une identité managée. Vous indiquerez ici un Resourceld qui n'a pas de clé de compte ni de mot de passe. Le Resourceld doit inclure l'ID d'abonnement du compte de stockage, le groupe de ressources du compte de stockage et le nom du compte de stockage.
  - Format de l'identité managée :
    - *Resourceld=/subscriptions/votre ID d'abonnement/resourceGroups/le nom de votre groupe de ressources/providers/Microsoft.Storage/storageAccounts/le nom de votre compte de stockage/;*
- **container** spécifie le nom d'un conteneur ou d'une table dans votre compte de stockage. Par défaut, tous les objets blob du conteneur sont récupérables. Si vous souhaitez indexer uniquement les objets blob dans un répertoire virtuel particulier, vous pouvez spécifier ce répertoire à l'aide du paramètre facultatif **query**.

Exemple de création d'un objet source de données blob à l'aide de l'[API REST](#) :

```

POST https://[service name].search.windows.net/datasources?api-version=2020-06-30
Content-Type: application/json
api-key: [admin key]

{
    "name" : "blob-datasource",
    "type" : "azureblob",
    "credentials" : { "connectionString" : "ResourceId=/subscriptions/00000000-0000-0000-0000-
000000000000/resourceGroups/resource-group-name/providers/Microsoft.Storage/storageAccounts/storage-account-
name/;" },
    "container" : { "name" : "my-container", "query" : "<optional-virtual-directory-name>" }
}

```

## 4 – Créer l'index

L'index spécifie les champs d'un document, les attributs et d'autres constructions qui façonnent l'expérience de recherche.

Voici comment créer un index avec un champ `content` pouvant faire l'objet d'une recherche afin de stocker le texte extrait d'objets blob :

```

POST https://[service name].search.windows.net/indexes?api-version=2020-06-30
Content-Type: application/json
api-key: [admin key]

{
    "name" : "my-target-index",
    "fields": [
        { "name": "id", "type": "Edm.String", "key": true, "searchable": false },
        { "name": "content", "type": "Edm.String", "searchable": true, "filterable": false, "sortable": false, "facetable": false }
    ]
}

```

Pour plus d'informations sur la création d'index, consultez [Création d'un index](#)

## 5 – Créer l'indexeur

Un indexeur connecte une source de données à un index de recherche cible et fournit une planification afin d'automatiser l'actualisation des données.

Une fois l'index et la source de données créés, vous êtes prêt à créer l'indexeur.

Exemple de définition d'indexeur pour un indexeur de blobs :

```

POST https://[service name].search.windows.net/indexers?api-version=2020-06-30
Content-Type: application/json
api-key: [admin key]

{
    "name" : "blob-indexer",
    "dataSourceName" : "blob-datasource",
    "targetIndexName" : "my-target-index",
    "schedule" : { "interval" : "PT2H" }
}

```

Cet indexeur s'exécutera toutes les deux heures (intervalle de planification défini sur « PT2H »). Pour exécuter un indexeur toutes les 30 minutes, définissez l'intervalle sur « PT30M ». Le plus court intervalle pris en charge est de 5 minutes. La planification est facultative : en cas d'omission, un indexeur ne s'exécute qu'une seule fois lorsqu'il est créé. Toutefois, vous pouvez à tout moment exécuter un indexeur à la demande.

Pour plus d'informations sur l'API Créer un indexeur, consultez [Créer un indexeur](#).

Pour plus d'informations sur la définition des planifications de l'indexeur, consultez [Comment planifier des indexeurs pour la Recherche cognitive Azure](#).

## Voir aussi

En savoir plus sur les indexeurs de Stockage Azure :

- [Indexeur de blobs Azure](#)
- [Indexeur Azure Data Lake Storage Gen2](#)
- [Indexeur de tables Azure](#)

# Configurer une connexion d'indexeur à une base de données Cosmos DB à l'aide d'une identité managée

04/10/2020 • 7 minutes to read • [Edit Online](#)

Cette page explique comment configurer une connexion d'indexeur à une base de données Azure Cosmos DB à l'aide d'une identité managée au lieu de fournir des informations d'identification dans la chaîne de connexion de l'objet source de données.

Avant d'en apprendre plus sur cette fonctionnalité, il est recommandé de comprendre ce qu'est un indexeur et savoir comment configurer un indexeur pour votre source de données. Pour plus d'informations, consultez les liens suivants :

- [Présentation de l'indexeur](#)
- [Indexeur Azure Cosmos DB](#)

## Configurer une connexion à l'aide d'une identité managée

### 1 – Activer l'identité managée affectée par le système

Quand une identité managée affectée par le système est activée, Azure crée une identité pour votre service de recherche qui peut être utilisée pour vous authentifier auprès d'autres services Azure au sein du même locataire et du même abonnement. Vous pouvez ensuite utiliser cette identité dans les attributions de contrôle d'accès en fonction du rôle (RBAC) qui autorisent l'accès aux données pendant l'indexation.

The screenshot shows the Azure Cognitive Search Service Identity settings page. The left sidebar lists navigation options: Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Quick start, Keys, Scale, Search traffic analytics, Identity (which is selected and highlighted with a red box), Properties, and Locks. The main content area has a title 'System assigned' with a sub-instruction: 'A system assigned managed identity enables Azure resources to authenticate to cloud services (e.g. Azure managed identity is tied to the lifecycle of this resource. Additionally, each resource (e.g. Virtual Machine, Function App, etc.) can have its own managed identity).'. Below this are buttons for Save (highlighted with a red box), Discard, Refresh, and Got feedback?. Underneath is a 'Status' section with a toggle switch set to 'On' (highlighted with a red box).

Après avoir sélectionné **Enregistrer**, vous verrez un ID d'objet qui a été attribué à votre service de recherche.

The screenshot shows the 'Identity' section of the Azure Cognitive Search Service configuration. On the left, a sidebar lists various service management options like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, and Settings. Under Settings, the 'Identity' option is selected and highlighted with a red box. The main pane displays the 'System assigned' identity configuration. It includes a 'Save' button, a 'Discard' button, a 'Refresh' button, and a 'Got feedback?' link. A status switch is set to 'On'. The 'Object ID' field contains the value '1da4d008-e5ce-4a03-8041-4694e5059d36', which is also highlighted with a red box. Below this is a 'Permissions' section with a 'Azure role assignments' button. A note at the bottom states: 'This resource is registered with Azure Active Directory. You can control its access to services like Azure Resource Manager and other Azure services using roles.' A blue 'Copy' icon is located next to the Object ID.

## 2 – Ajouter une attribution de rôle

Au cours de cette étape, vous allez accorder à votre service Recherche cognitive Azure l'autorisation de lire les données de votre compte Cosmos DB.

1. Dans le portail Azure, accédez au compte Cosmos DB qui contient les données que vous souhaitez indexer.
2. Sélectionnez Contrôle d'accès (IAM)
3. Sélectionnez Ajouter, puis Ajouter une attribution de rôle.

The screenshot shows the 'Access control (IAM)' section of the Azure Cosmos DB account configuration. The left sidebar lists options like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Quick start, Notifications, Data Explorer, and Settings. The 'Access control (IAM)' option is selected and highlighted with a red box. The main pane shows a table with columns for 'Add', 'Edit columns', 'Refresh', 'Remove', and 'Got feedback?'. A dropdown menu is open over the 'Add' button, showing options: 'Add role assignment' (which is highlighted with a red box), 'Deny assignments', 'Classic administrators', and 'Roles'. To the right of the table, there are two sections: 'Check access' (with a 'Find' input field for 'Azure AD user, group, or service principal') and 'Add a role as' (with a 'Grant access to' input field and an 'Add' button). Another section, 'View deny as', is partially visible below it.

4. Sélectionner le rôle Lecteur de compte Cosmos DB
5. Laissez Attribuer l'accès à sur Utilisateur, groupe ou principal de service Azure AD.
6. Recherchez votre service de recherche, sélectionnez-le, puis sélectionnez Enregistrer.

### 3 – Créer la source de données

L'[API REST](#), le portail Azure et le [Kit de développement logiciel \(SDK\) .NET](#) prennent également en charge la chaîne de connexion des identités managées. Voici un exemple de création d'une source de données pour indexer des données à partir de Cosmos DB à l'aide de l'[API REST](#) et d'une chaîne de connexion d'identité gérée. Le format de chaîne de connexion d'identité managée est le même pour l'API REST, le kit de développement logiciel (SDK) .NET et le portail Azure.

Lorsque vous utilisez des identités managées pour l'authentification, les **informations d'identification** n'incluent pas de clé de compte.

```
POST https://[service name].search.windows.net/datasources?api-version=2020-06-30
Content-Type: application/json
api-key: [Search service admin key]

{
    "name": "cosmos-db-datasource",
    "type": "cosmosdb",
    "credentials": {
        "connectionString": "Database=sq1-test-db;ResourceId=/subscriptions/00000000-0000-0000-0000-000000000000/resourceGroups/cosmos-db-resource-group/providers/Microsoft.DocumentDB/databaseAccounts/my-cosmos-db-account/"
    },
    "container": { "name": "myCollection", "query": null },
    "dataChangeDetectionPolicy": {
        "@odata.type": "#Microsoft.Azure.Search.HighWaterMarkChangeDetectionPolicy",
        "highWaterMarkColumnName": "_ts"
    }
}
```

Le corps de la requête contient la définition de la source de données, qui doit inclure les champs suivants :

CHAMP	DESCRIPTION
<b>name</b>	Obligatoire. Choisissez un nom pour représenter votre objet source de données.
<b>type</b>	Obligatoire. Doit être <code>cosmosdb</code> .

CHAMP	DESCRIPTION
credentials	Obligatoire.  Lors de la connexion à l'aide d'une identité managée, le format des informations d'identification doit être : <i>Database=[nom-base-de-données];ResourceId=[chaîne-id-ressource];(ApiKind=[type-apil];)</i>  Format de ResourceId : <i>ResourceId=/subscriptions/votre ID d'abonnement/resourceGroups/le nom de votre groupe de ressources/providers/Microsoft.DocumentDB/databaseAccounts/le nom de votre compte cosmos db/;</i>  Pour les collections SQL, la chaîne de connexion ne requiert pas d'ApiKind.  Pour les collections MongoDB, ajoutez <b>ApiKind=MongoDb</b> à la chaîne de connexion.  Pour les graphes Gremlin et les tables Cassandra, inscrivez-vous à la <a href="#">préversion de l'indexeur contrôlé</a> pour accéder à la préversion et aux informations sur la façon de mettre en forme les informations d'identification.
container	Contient les éléments suivants : <b>nom</b> : Obligatoire. Spécifiez l'ID de la collection de bases de données à indexier. <b>query</b> : facultatif. Vous pouvez spécifier une requête pour obtenir un schéma plat à partir d'un document JSON arbitraire de manière à ce qu'Azure Search puisse procéder à l'indexation. Pour l'API MongoDB, l'API Gremlin et l'API Cassandra, les requêtes ne sont pas prises en charge.
dataChangeDetectionPolicy	Recommandé
dataDeletionDetectionPolicy	Facultatif

#### 4 – Créer l'index

L'index spécifie les champs d'un document, les attributs et d'autres constructions qui façonnent l'expérience de recherche.

Voici comment créer un index avec un champ `booktitle` pouvant faire l'objet d'une recherche :

```
POST https://[service name].search.windows.net/indexes?api-version=2020-06-30
Content-Type: application/json
api-key: [admin key]

{
  "name" : "my-target-index",
  "fields": [
    { "name": "id", "type": "Edm.String", "key": true, "searchable": false },
    { "name": "booktitle", "type": "Edm.String", "searchable": true, "filterable": false, "sortable": false,
      "facetable": false }
  ]
}
```

Pour plus d'informations sur la création d'index, consultez [Création d'un index](#)

## 5 – Créer l'indexeur

Un indexeur connecte une source de données à un index de recherche cible et fournit une planification afin d'automatiser l'actualisation des données.

Une fois l'index et la source de données créés, vous êtes prêt à créer l'indexeur.

Exemple de définition d'indexeur :

```
POST https://[service name].search.windows.net/indexers?api-version=2020-06-30
Content-Type: application/json
api-key: [admin key]

{
    "name" : "cosmos-db-indexer",
    "dataSourceName" : "cosmos-db-datasource",
    "targetIndexName" : "my-target-index",
    "schedule" : { "interval" : "PT2H" }
}
```

Cet indexeur s'exécutera toutes les deux heures (intervalle de planification défini sur « PT2H »). Pour exécuter un indexeur toutes les 30 minutes, définissez l'intervalle sur « PT30M ». Le plus court intervalle pris en charge est de 5 minutes. La planification est facultative : en cas d'omission, un indexeur ne s'exécute qu'une seule fois lorsqu'il est créé. Toutefois, vous pouvez à tout moment exécuter un indexeur à la demande.

Pour plus d'informations sur l'API Crée un indexeur, consultez [Créer un indexeur](#).

Pour plus d'informations sur la définition des planifications de l'indexeur, consultez [Comment planifier des indexeurs pour la Recherche cognitive Azure](#).

## Voir aussi

En savoir plus sur les indexeurs Cosmos DB :

- [Indexeur Azure Cosmos DB](#)

# Configurer une connexion d'indexeur à Azure SQL Database à l'aide d'une identité managée

04/10/2020 • 9 minutes to read • [Edit Online](#)

Cette page explique comment configurer une connexion d'indexeur à Azure SQL Database à l'aide d'une identité managée au lieu de fournir des informations d'identification dans la chaîne de connexion de l'objet source de données.

Avant d'en apprendre plus sur cette fonctionnalité, il est recommandé de comprendre ce qu'est un indexeur et savoir comment configurer un indexeur pour votre source de données. Pour plus d'informations, consultez les liens suivants :

- [Présentation de l'indexeur](#)
- [Indexeur Azure SQL](#)

## Configurer une connexion à l'aide d'une identité managée

### 1 – Activer l'identité managée affectée par le système

Quand une identité managée affectée par le système est activée, Azure crée une identité pour votre service de recherche qui peut être utilisée pour vous authentifier auprès d'autres services Azure au sein du même locataire et du même abonnement. Vous pouvez ensuite utiliser cette identité dans les attributions de contrôle d'accès en fonction du rôle (RBAC) qui autorisent l'accès aux données pendant l'indexation.

The screenshot shows the Azure Cognitive Search Service Identity settings page. The left sidebar lists various service settings: Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Quick start, Keys, Scale, Search traffic analytics, Identity (which is selected and highlighted with a red box), Properties, and Locks. The main content area is titled 'System assigned' and contains the following text: 'A system assigned managed identity enables Azure resources to authenticate to cloud services (e.g. Azure managed identity is tied to the lifecycle of this resource. Additionally, each resource (e.g. Virtual Machine, Function App, etc.) can have its own identity).'. Below this text are four buttons: Save (highlighted with a red box), Discard, Refresh, and Got feedback?. Underneath these buttons is a 'Status' section with a toggle switch. The switch is currently set to 'On' (highlighted with a red box), indicating that the system-assigned identity is enabled. A status message above the switch says 'Identity successfully registered'.

Après avoir sélectionné **Enregistrer**, vous verrez un ID d'objet qui a été attribué à votre service de recherche.

The screenshot shows the 'Identity' settings page for an Azure Cognitive Search service. On the left, there's a sidebar with various options like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Settings, Quick start, Keys, Scale, Search traffic analytics, Identity (which is selected and highlighted in grey), Properties, and Help. The main area has a search bar at the top. Below it, the title 'System assigned' is underlined. A status message says: 'A system assigned managed identity enables Azure resources to authenticate to cloud services (e.g. Azure Key Vault) and can be used by any Azure resource that supports managed identities. This managed identity is tied to the lifecycle of this resource. Additionally, each resource (e.g. Virtual Machine) can have its own managed identity.' There are buttons for Save, Discard, Refresh, and Got feedback? A 'Status' switch is set to 'On'. An 'Object ID' input field contains the value '1da4d008-e5ce-4a03-8041-4694e5059d36', which is also highlighted with a red box. Below that is a 'Permissions' section with a 'Azure role assignments' button. At the bottom, a note says: 'This resource is registered with Azure Active Directory. You can control its access to services like Azure Resource Manager and other Azure services using Azure Active Directory roles.'

## 2 – Approvisionner un administrateur Azure Active Directory pour SQL Server

Lors de la connexion à la base de données à l'étape suivante, vous devez vous connecter avec un compte Azure Active Directory (Azure AD) disposant d'un accès administrateur à la base de données afin de permettre à votre service de recherche d'y accéder.

Suivez les instructions indiquées [ici](#) pour accorder à votre compte Azure AD un accès administrateur à la base de données.

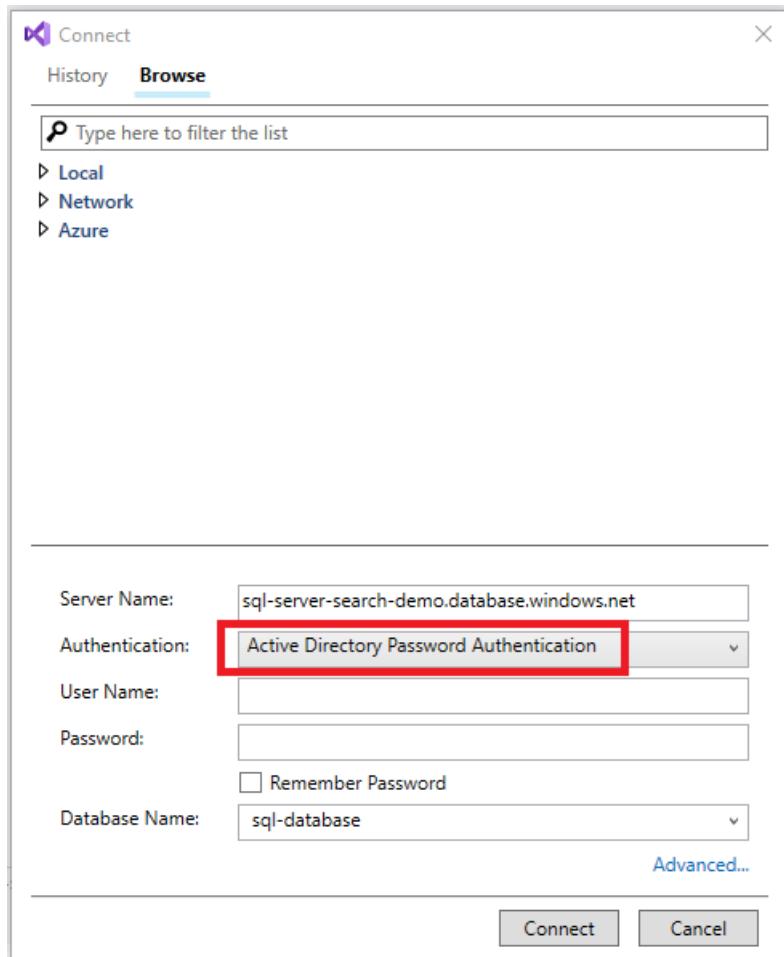
## 3 – Attribuer des autorisations au service de recherche

Suivez les étapes ci-dessous pour attribuer au service de recherche l'autorisation de lire la base de données.

1. Connectez-vous à Visual Studio.

The screenshot shows the 'sql-database' settings page for an Azure SQL database. The left sidebar includes Overview, Activity log, Tags, Diagnose and solve problems, Quick start, Query editor (preview), Power Platform (Power BI, Power Apps, Power Automate), and Settings (Configure, Geo-Replication, Connection strings, Sync to other databases). The main area has a search bar and a toolbar with Copy, Restore, Export, Set server firewall, Delete, Connect with... (which is highlighted with a red box), and Feedback. Below that, it shows resource group (Online), location (East US), subscription, and tags. It also displays compute utilization over the last hour, 24 hours, or 7 days. A note at the bottom says: 'Show data for last: 1 hour 24 hours 7 days'.

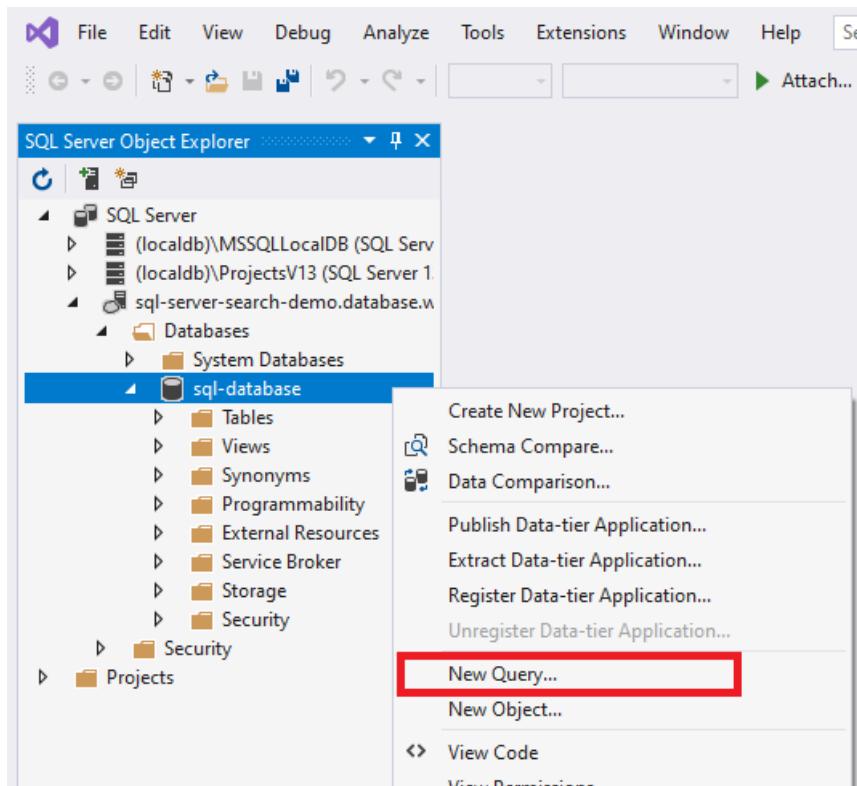
2. S'authentifier à l'aide du compte Azure AD

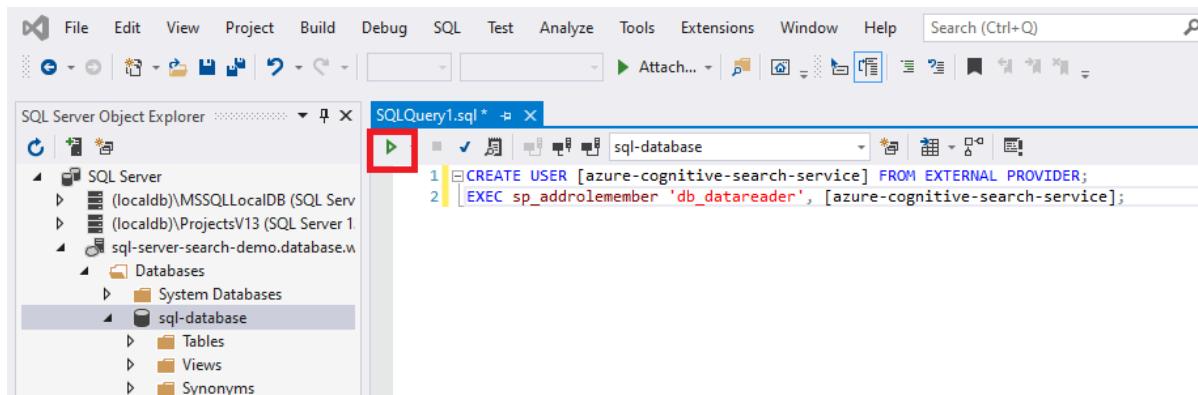


### 3. Exécutez les commandes suivantes :

Incluez les crochets autour du nom de votre service de recherche.

```
CREATE USER [your search service name here] FROM EXTERNAL PROVIDER;
EXEC sp_addrolemember 'db_datareader', [your search service name here];
```





#### NOTE

Si l'identité du service de recherche de l'étape 1 est modifiée après avoir terminé cette étape, vous devez supprimer l'appartenance au rôle et supprimer l'utilisateur dans la base de données SQL, puis rajouter les autorisations en effectuant à nouveau l'étape 3. La suppression de l'appartenance au rôle et de l'utilisateur peut être effectuée en exécutant les commandes suivantes :

```
sp_droprolemember 'db_datareader', [your search service name];
DROP USER IF EXISTS [your search service name];
```

## 4 – Ajouter une attribution de rôle

Au cours de cette étape, vous allez accorder à votre service Recherche cognitive Azure l'autorisation de lire les données de votre serveur SQL.

1. Dans le portail Azure, accédez à votre page Azure SQL Server.
2. Sélectionnez Contrôle d'accès (IAM)
3. Sélectionnez Ajouter, puis Ajouter une attribution de rôle.

4. Sélectionnez le rôle **Lecteur** approprié.
5. Laissez **Attribuer l'accès à** sur **Utilisateur, groupe ou principal de service Azure AD**.
6. Recherchez votre service de recherche, sélectionnez-le, puis sélectionnez **Enregistrer**.

## 5 – Créer la source de données

L'[API REST](#), le portail Azure et le [Kit de développement logiciel \(SDK\) .NET](#) prennent également en charge la chaîne de connexion des identités managées. Voici un exemple de création d'une source de données pour indexer des données à partir d'une instance Azure SQL Database à l'aide de l'[API REST](#) et d'une chaîne de connexion d'identité gérée. Le format de chaîne de connexion d'identité managée est le même pour l'API REST, le kit de développement logiciel (SDK) .NET et le portail Azure.

Lors de la création d'une source de données à l'aide de l'[API REST](#), la source de données doit avoir les propriétés requises suivantes :

- **name** est le nom unique de la source de données au sein de votre service de recherche.
- **type** est `azuresql`
- **credentials**
  - Le format d'**informations d'identification** est différent selon que vous utilisez ou non une identité managée. Vous indiquerez ici un nom Initial Catalog ou Database et un Resourceld qui n'a pas de clé de compte ni de mot de passe. Le Resourceld doit inclure l'ID d'abonnement d'Azure SQL Database, le groupe de ressources d'Azure SQL Database et le nom de la base de données SQL.
  - Format de la chaîne de connexion de l'identité managée :
    - *Initial Catalog/Database=nom de la base de données;Resourceld=/subscriptions/votre ID d'abonnement/resourceGroups/votre nom de groupe de ressources/providers/Microsoft.Sql/servers/votre nom de serveur SQL;/Connection Timeout=durée de la connexion;*
- **container** spécifie le nom de la table ou de l'affichage que vous souhaitez indexer.

Exemple de création d'un objet de source de données Azure SQL à l'aide de l'[API REST](#) :

```
POST https://[service name].search.windows.net/datasources?api-version=2020-06-30
Content-Type: application/json
api-key: [admin key]

{
  "name" : "sql-datasource",
  "type" : "azuresql",
  "credentials" : { "connectionString" : "Database=sql-database;ResourceId=/subscriptions/00000000-0000-0000-0000-000000000000/resourceGroups/azure-sql-resource-group/providers/Microsoft.Sql/servers/sql-server-search-demo;Connection Timeout=30;" },
  "container" : { "name" : "my-table" }
}
```

## 6 – Créer l'index

L'index spécifie les champs d'un document, les attributs et d'autres constructions qui façonnent l'expérience de recherche.

Voici comment créer un index avec un champ `booktitle` pouvant faire l'objet d'une recherche :

```
POST https://[service name].search.windows.net/indexes?api-version=2020-06-30
Content-Type: application/json
api-key: [admin key]

{
    "name" : "my-target-index",
    "fields": [
        { "name": "id", "type": "Edm.String", "key": true, "searchable": false },
        { "name": "booktitle", "type": "Edm.String", "searchable": true, "filterable": false, "sortable": false,
        "facetable": false }
    ]
}
```

Pour plus d'informations sur la création d'index, consultez [Création d'un index](#)

## 7 – Créer l'indexeur

Un indexeur connecte une source de données à un index de recherche cible et fournit une planification afin d'automatiser l'actualisation des données.

Une fois l'index et la source de données créés, vous êtes prêt à créer l'indexeur.

Exemple de définition d'indexeur pour un indexeur Azure SQL :

```
POST https://[service name].search.windows.net/indexers?api-version=2020-06-30
Content-Type: application/json
api-key: [admin key]

{
    "name" : "sql-indexer",
    "dataSourceName" : "sql-datasource",
    "targetIndexName" : "my-target-index",
    "schedule" : { "interval" : "PT2H" }
}
```

Cet indexeur s'exécutera toutes les deux heures (intervalle de planification défini sur « PT2H »). Pour exécuter un indexeur toutes les 30 minutes, définissez l'intervalle sur « PT30M ». Le plus court intervalle pris en charge est de 5 minutes. La planification est facultative : en cas d'omission, un indexeur ne s'exécute qu'une seule fois lorsqu'il est créé. Toutefois, vous pouvez à tout moment exécuter un indexeur à la demande.

Pour plus d'informations sur l'API Créer un indexeur, consultez [Créer un indexeur](#).

Pour plus d'informations sur la définition des planifications de l'indexeur, consultez [Comment planifier des indexeurs pour la Recherche cognitive Azure](#).

## Dépannage

Si vous recevez une erreur lorsque l'indexeur tente de se connecter à la source de données qui indique que le client n'est pas autorisé à accéder au serveur, consultez les [erreurs courantes de l'indexeur](#).

## Voir aussi

En savoir plus sur l'indexeur Azure SQL :

- Indexeur Azure SQL

# Ajouter la recherche en texte intégral à des données blob Azure à l'aide de Recherche cognitive Azure

04/10/2020 • 11 minutes to read • [Edit Online](#)

La recherche dans les différents types de contenu enregistrés dans le Stockage Blob Azure peut constituer un problème difficile à résoudre. Cependant, vous pouvez indexer le contenu de vos objets blob et y effectuer une recherche en quelques clics à l'aide de [Recherche cognitive Azure](#). Recherche cognitive Azure propose une intégration prédéfinie qui permet d'effectuer une indexation à partir de Stockage Blob via un [\*indexeur Blob\*](#) qui ajoute à l'indexation des capacités de reconnaissance des sources de données.

## Implications de l'ajout de la recherche en texte intégral aux données blob

Recherche cognitive Azure est un service de recherche cloud qui propose des moteurs d'indexation et de requête qui opèrent sur les index définis par l'utilisateur hébergés dans votre service de recherche. Colocaliser votre contenu recherchable avec le moteur de requête dans le cloud est une nécessité pour offrir aux utilisateurs les performances qu'ils attendent eu égard au délai d'affichage des résultats des requêtes de recherche.

Recherche cognitive Azure s'intègre avec Stockage Blob Azure au niveau de la couche d'indexation. Le contenu de vos objets blob est ainsi importé sous forme de documents de recherche indexés dans des *index inversés* et d'autres structures de requête qui prennent en charge les requêtes de texte en forme libre et les expressions de filtre. Sachant que le contenu de vos objets blob est indexé dans un index de recherche, il est possible de tirer parti de l'ensemble des fonctionnalités de requête de Recherche cognitive Azure pour accéder à ce contenu.

Une fois créé et rempli, l'index existe indépendamment de votre conteneur d'objets blob. Vous pouvez néanmoins réexécuter les opérations d'indexation de façon à actualiser l'index avec les modifications apportées au conteneur sous-jacent. Les informations d'horodatage des différents objets blob servent à détecter les modifications. Vous pouvez opter pour une exécution planifiée ou une indexation à la demande en guise de mécanisme d'actualisation.

Les entrées sont vos objets blob, dans un même conteneur, dans Stockage Blob Azure. Les objets blob peuvent correspondre à pratiquement tout type de données texte. Si vos objets blob contiennent des images, vous pouvez ajouter [l'enrichissement par IA à l'indexation d'objets blob](#) pour créer et extraire du texte des images.

La sortie est toujours un index Recherche cognitive Azure, utilisé pour la recherche, l'extraction et l'exploration rapides de texte dans les applications clientes. Au milieu se trouve l'architecture proprement dite du pipeline d'indexation. Le pipeline est basé sur la fonctionnalité d'*indexeur*, décrite plus loin dans cet article.

## Commencer avec les services

Vous avez besoin de Recherche cognitive Azure et de Stockage d'objets blob Azure. Dans Stockage Blob, vous avez besoin d'un conteneur qui fournit le contenu source.

Vous pouvez commencer directement dans votre page du portail des comptes Stockage. Dans la page de navigation de gauche, sous **Service Blob**, cliquez sur **Ajouter Recherche cognitive Azure** pour créer un nouveau service ou en sélectionner un existant.

Une fois que vous avez ajouté Recherche cognitive Azure à votre compte de stockage, vous pouvez suivre le processus standard d'indexation de données blob. Nous vous recommandons d'utiliser l'**Assistant Importation des données** de Recherche cognitive Azure pour une mise en route simple ou d'appeler les API REST à l'aide d'un outil comme Postman. Ce tutoriel vous explique pas à pas comment appeler l'API REST dans Postman : [Indexer et](#)

rechercher des données semi-structurées (objets blob JSON) dans la Recherche cognitive Azure.

## Utiliser un indexeur d'objets blob

Un *indexeur* est un sous-service qui reconnaît les sources de données. Avec sa logique interne, il échantillonne les données, lit les données de métadonnées, extrait les données et les sérialise dans des documents JSON à partir de formats natifs pour être ensuite importées.

Dans Stockage Azure, les objets blob sont indexés à l'aide de l'[indexeur de stockage d'objets blob de Recherche cognitive Azure](#). Vous pouvez appeler cet indexeur à partir de l'Assistant **Importation des données**, d'une API REST ou du kit SDK .NET. Dans le code, vous pouvez utiliser cet indexeur en définissant le type et en fournissant des informations de connexion qui incluent un compte Stockage Azure associé à un conteneur d'objets blob. Vous pouvez créer un sous-ensemble de vos objets blob en créant un répertoire virtuel, que vous pouvez ensuite transmettre comme paramètre, ou en filtrant sur une extension de type de fichier.

Un indexeur effectue le « craquage de document » en ouvrant un objet blob pour en inspecter le contenu. Une fois connecté à la source de données, il s'agit de la première étape du pipeline. Pour les données blob, c'est à ce stade que les fichiers PDF, les documents Office et d'autres types de contenu sont détectés. Le craquage de document avec extraction de texte n'est pas facturé. Si vos objets blob contiennent des images, celles-ci sont ignorées si vous [n'ajoutez pas l'enrichissement par IA](#). L'indexation standard s'applique uniquement au contenu texte.

L'indexeur d'objets blob est assorti de paramètres de configuration et prend en charge le suivi des modifications si les données sous-jacentes fournissent suffisamment d'informations. Vous trouverez des informations supplémentaires sur les fonctionnalités de base dans [Indexeur de stockage d'objets blob de Recherche cognitive Azure](#).

### Types de contenu pris en charge

En exécutant un indexeur d'objets blob sur un conteneur, vous pouvez extraire du texte et des métadonnées à partir des types de contenu suivants avec une seule requête :

- PDF
- Formats Microsoft Office : DOCX/DOC/DOCM, XLSX/XLS/XLSM, PPTX/PPT/PPTM, MSG (e-mails Outlook), XML (WORD XML 2003 et 2006)
- Formats de document ouverts : ODT, ODS, ODP
- HTML
- XML
- ZIP
- GZ
- EPUB
- EML
- RTF
- Fichiers de texte brut (voir aussi [l'indexation de texte brut](#))
- JSON (consultez [l'indexation d'objets JSON blobs](#))
- CSV (consultez [Indexation d'objets blob CSV](#))

### Indexation de métadonnées blob

Pour faciliter le tri dans les objets blob constitués de tout type de contenu, un scénario courant consiste à indexer les métadonnées personnalisées et les propriétés système pour chaque objet blob. De cette façon, les informations de tous les objets blob sont indexées indépendamment du type de document et stockées dans un index de votre service de recherche. Le nouvel index vous permet alors d'effectuer un tri, un filtrage et une facette dans l'ensemble du contenu du stockage Blob.

#### **NOTE**

Les balises d'index d'objet blob sont indexées en mode natif par le service de stockage d'objets blob et exposées pour l'interrogation. Si les attributs clé/valeur de vos objets blob nécessitent des fonctionnalités d'indexation et de filtrage, les balises d'index d'objet blob doivent être exploitées à la place des métadonnées.

Pour en savoir plus sur un index d'objets blob, consultez [Gérer et rechercher des données sur le Stockage Blob Azure avec un index d'objets blob](#).

## **Indexation d'objets JSON**

Les indexeurs peuvent être configurés pour extraire le contenu structuré des objets blob qui contiennent des objets JSON. Un indexeur peut lire les objets blob JSON et analyser le contenu structuré dans les champs adaptés d'un document de recherche. Les indexeurs peuvent également extraire les objets blob contenant des objets JSON et mapper chaque élément avec un document de recherche différent. Vous pouvez définir un mode d'analyse pour affecter le type d'objet JSON créé par l'indexeur.

## **Rechercher du contenu d'objet blob dans un index de recherche**

La sortie d'une indexation est un index de recherche, qui permet une exploration interactive en utilisant des requêtes de texte libre et filtrées dans une application cliente. Pour une exploration et une vérification initiales de contenu, nous vous recommandons de commencer avec l'[Explorateur de recherche](#) sur le portail, qui vous permet d'examiner la structure du document. L'Explorateur de recherche vous permet d'utiliser une [syntaxe de requête simple](#), une [syntaxe de requête complète](#) et une [syntaxe d'expression de filtre](#).

Une solution plus permanente consiste à regrouper les entrées de requête et à présenter la réponse sous forme de résultats de recherche dans une application cliente. Le tutoriel C# suivant explique comment créer une application de recherche : [Créer votre première application dans Recherche cognitive Azure](#).

## **Étapes suivantes**

- [Charger, télécharger et répertorier des blobs à l'aide du portail Azure \(Stockage blob Azure\)](#)
- [Configurer un indexeur blob \(Recherche cognitive Azure\)](#)

# Utiliser l'IA pour comprendre les données de stockage blob

04/10/2020 • 17 minutes to read • [Edit Online](#)

Dans Stockage Blob Azure, les données sont souvent constituées de divers contenus non structurés comme des images, du texte long, des fichiers PDF et des documents Office. En utilisant les fonctionnalités d'IA dans Recherche cognitive Azure, vous pouvez arriver à comprendre et à extraire des informations précieuses des objets blob, et ce de différentes manières. Voici des exemples d'application de l'IA au contenu d'objets blob :

- Extraction de texte d'images à l'aide de la reconnaissance optique de caractères (OCR)
- Création d'une description de scène ou d'étiquettes à partir d'une photo
- Détection de langue et traduction de texte dans différentes langues
- Traitement de texte avec la reconnaissance d'entité nommée (NER) pour rechercher des références à des personnes, dates, lieux ou organisations

Même si vous n'êtes intéressé que par une seule de ces fonctionnalités d'IA, il est courant d'en combiner plusieurs dans le même pipeline (par exemple, extraction de texte d'une image numérisée et recherche des dates et des emplacements référencés).

L'enrichissement par IA crée de nouvelles informations, capturées sous forme de texte, stockées dans des champs. Après l'enrichissement, vous pouvez accéder à ces informations à partir d'un index de recherche en effectuant une recherche en texte intégral ou renvoyer les documents enrichis à Stockage Azure pour permettre de nouvelles expériences applicatives comme notamment l'exploration de données pour des scénarios de découverte ou d'analytique.

Dans cet article, nous examinons l'enrichissement par IA à travers un objectif grand-angle pour vous permettre de saisir rapidement l'ensemble du processus, depuis la transformation des données brutes d'objets blob jusqu'aux informations requêtables dans un index de recherche ou une base de connaissances.

## Que signifie « enrichir » des données blob avec l'IA ?

L'*enrichissement par IA* fait partie de l'architecture d'indexation de Recherche cognitive Azure qui intègre l'IA (intelligence artificielle) de Microsoft ou une IA personnalisée que vous fournissez. Il permet de mettre en œuvre des scénarios de bout en bout quand il s'agit de traiter des objets blob (existants ou nouvellement obtenus ou mis à jour), d'ouvrir tous les formats de fichiers pour en extraire des images et du texte, d'extraire les informations souhaitées avec différentes fonctionnalités d'IA, puis de les indexer dans un index de recherche pour accélérer la recherche, l'extraction et l'exploration.

Les entrées sont vos objets blob, dans un même conteneur, dans Stockage Blob Azure. Les objets blob peuvent correspondre à pratiquement tout type de données texte ou image.

La sortie est toujours un index de recherche, utilisé pour la recherche, l'extraction et l'exploration rapides de texte dans les applications clientes. Par ailleurs, la sortie peut aussi être une *base de connaissances* qui projette des documents enrichis dans des objets blob Azure ou des tables Azure pour exercer une analyse en aval dans des outils comme Power BI ou des charges de travail de science des données.

Au milieu se trouve l'architecture du pipeline proprement dite. Le pipeline est basé sur la fonctionnalité d'*indexeur*, à laquelle vous pouvez affecter un *ensemble de compétences*, qui se compose d'une ou plusieurs *compétences* fournissant l'IA. L'objectif du pipeline est de produire des *documents enrichis* : le contenu brut de départ se dote d'une structure, d'un contexte et d'informations supplémentaires à mesure qu'il avance dans le pipeline. Les

documents enrichis sont consommés pendant l'indexation pour créer des index inversés et d'autres structures utilisées dans la recherche en texte intégral ou l'exploration et l'analytique.

## Commencer avec les services

Vous avez besoin de Recherche cognitive Azure et de Stockage d'objets blob Azure. Dans Stockage Blob, vous avez besoin d'un conteneur qui fournit le contenu source.

Vous pouvez commencer directement dans votre page du portail des comptes Stockage. Dans la page de navigation de gauche, sous **Service Blob**, cliquez sur **Ajouter Recherche cognitive Azure** pour créer un nouveau service ou en sélectionner un existant.

Une fois que vous avez ajouté Recherche cognitive Azure à votre compte de stockage, vous pouvez suivre le processus standard d'enrichissement de données dans une source de données Azure. Nous vous recommandons d'utiliser l'**Assistant Importation des données** de Recherche cognitive Azure pour une mise en route simple de l'enrichissement par IA. Ce guide de démarrage rapide vous accompagne tout au long de la procédure : [Créez un pipeline d'enrichissement par IA sur le portail](#).

Dans les sections suivantes, nous allons explorer d'autres composants et concepts.

## Utiliser un indexeur d'objets blob

L'enrichissement par IA est une extension du pipeline d'indexation, et dans Recherche cognitive Azure, ces pipelines reposent sur un *indexeur*. Un indexeur est un sous-service qui reconnaît les sources de données. Avec sa logique interne, il échantillonne les données, lit les données de métadonnées, extrait les données et les sérialise dans des documents JSON à partir de formats natifs pour être ensuite importées. Les indexeurs sont souvent utilisés seuls pour l'importation, indépendamment de l'IA, mais si vous voulez créer un pipeline d'enrichissement par IA, vous devrez l'associer à un indexeur et à un ensemble de compétences. Cette section met en lumière l'indexeur ; la section suivante porte sur les ensembles de compétences.

Dans Stockage Azure, les objets blob sont indexés à l'aide de l'[indexeur de stockage d'objets blob de Recherche cognitive Azure](#). Vous pouvez appeler cet indexeur à partir de l'**Assistant Importation des données**, d'une API REST ou du kit SDK .NET. Dans le code, vous pouvez utiliser cet indexeur en définissant le type et en fournissant des informations de connexion qui incluent un compte Stockage Azure associé à un conteneur d'objets blob. Vous pouvez créer un sous-ensemble de vos objets blob en créant un répertoire virtuel, que vous pouvez ensuite transmettre comme paramètre, ou en filtrant sur une extension de type de fichier.

Un indexeur effectue le « craquage de document » en ouvrant un objet blob pour en inspecter le contenu. Une fois connecté à la source de données, il s'agit de la première étape du pipeline. Pour les données blob, c'est à ce stade que les fichiers PDF, les documents Office, les images et d'autres types de contenu sont détectés. Le craquage de document avec extraction de texte n'est pas facturé. Le craquage de document avec extraction d'images est facturé aux tarifs indiqués dans la [page des tarifs](#).

Même si le craquage concerne tous les documents, l'enrichissement ne se produit que si vous fournissez explicitement les compétences nécessaires. Par exemple, si votre pipeline est constitué exclusivement par l'analyse d'images, le texte contenu dans votre conteneur ou vos documents est ignoré.

L'indexeur d'objets blob est assorti de paramètres de configuration et prend en charge le suivi des modifications si les données sous-jacentes fournissent suffisamment d'informations. Vous trouverez des informations supplémentaires sur les fonctionnalités de base dans [Indexeur de stockage d'objets blob de Recherche cognitive Azure](#).

## Ajouter des composants d'IA

L'enrichissement par AI s'appuie sur certains modules qui recherchent des modèles ou des caractéristiques, puis effectuent une opération en conséquence. La reconnaissance faciale dans les photos, les descriptions textuelles de

photos, la détection d'expressions clés dans un document et la reconnaissance optique (ou la reconnaissance de texte imprimé ou manuscrit dans des fichiers binaires) sont autant d'exemples représentatifs.

Dans Recherche cognitive Azure, les *compétences* sont les composants individuels du traitement de l'IA que vous pouvez utiliser de façon autonome ou en association avec d'autres compétences.

- Les compétences intégrées sont adossées à Cognitive Services, l'analyse d'images reposant sur Vision par ordinateur et le traitement en langage naturel sur Analyse de texte. Pour obtenir la liste complète, consultez [Compétences intégrées pour l'enrichissement de contenu](#).
- Les compétences personnalisées correspondent à du code personnalisé, encapsulé dans la [définition d'une interface](#) qui permet une intégration dans le pipeline. Dans les solutions des clients, il est courant d'utiliser les deux, les compétences personnalisées fournissant des modules d'IA open source, tiers ou internes.

L'*ensemble de compétences* est l'assortiment de compétences utilisé dans un pipeline ; il est appelé après que la phase de craquage de document a mis le contenu à disposition. Un indexeur peut consommer un seul ensemble de compétences, mais celui-ci existe indépendamment d'un indexeur. Vous pouvez donc le réutiliser dans d'autres scénarios.

Si les compétences personnalisées peuvent paraître complexes, elles peuvent s'avérer simples du point de vue de l'implémentation. Si les packages dont vous disposez fournissent des critères spéciaux ou des modèles de classification, le contenu que vous extrayez des objets blob peut être transmis à ces modèles pour traitement. Comme l'enrichissement par IA est basé sur Azure, votre modèle doit aussi se trouver dans Azure. Certaines méthodologies d'hébergement courantes incluent l'utilisation d'[Azure Functions](#) ou de [Containers](#).

Les compétences intégrées adossées à Cognitive Services nécessitent une clé d'abonnement tout-en-un [Cognitive Services](#) attachée qui vous donne accès à la ressource. Une clé tout-en-un vous fait bénéficier de l'analyse d'images, de la détection de langue, de la traduction de texte et de l'analyse de texte. Les autres compétences intégrées sont les fonctionnalités de Recherche cognitive Azure et ne nécessitent pas de service ou de clé supplémentaire. Les fonctionnalités de mise en forme, de fractionnement et de fusion de texte sont des exemples de compétences d'assistance qui sont parfois nécessaires au moment de concevoir le pipeline.

Si vous utilisez uniquement des compétences personnalisées et des compétences d'utilitaire intégrées, il n'existe aucune dépendance ou de coûts en rapport avec Cognitive Services.

## Consommer une sortie enrichie par l'IA dans des solutions en aval

La sortie de l'enrichissement par IA est soit un index de recherche dans Recherche cognitive Azure, soit une [base de connaissances](#) dans Stockage Azure.

Dans Recherche cognitive Azure, un index de recherche sert à l'exploration interactive dans une application cliente à partir de requêtes de texte libre et filtrées. Les documents enrichis créés via l'IA sont au format JSON et sont indexés comme tous les documents dans Recherche cognitive Azure, tirant ainsi parti de tous les avantages offerts par un indexeur. Par exemple, pendant l'indexation, l'indexeur d'objets blob se réfère aux paramètres de configuration pour utiliser les mappages de champs ou la logique de détection des changements. Ces paramètres sont entièrement disponibles pour l'indexation normale et les charges de travail enrichies par l'IA. Après l'indexation, quand le contenu est stocké dans Recherche cognitive Azure, vous pouvez créer des requêtes élaborées et des expressions de filtre pour comprendre votre contenu.

Dans Stockage Azure, une base de connaissances peut prendre deux formes : celle d'un conteneur d'objets blob ou celle de tables dans Stockage Table.

- Un conteneur d'objets blob capture les documents enrichis dans leur intégralité, ce qui est utile si vous souhaitez alimenter d'autres processus.
- En revanche, Stockage Table peut accueillir des projections physiques de documents enrichis. Vous pouvez créer des tranches ou des couches de documents enrichis qui incluent ou excluent des parties spécifiques.

Pour l'analyse dans Power BI, les tables stockées dans Table Azure deviennent la source de données pour une visualisation et une exploration plus poussées.

Un document enrichi à la fin du pipeline est différent de sa version d'entrée initiale du fait de la présence de champs supplémentaires contenant les nouvelles informations extraites ou générées pendant l'enrichissement. Ainsi, vous pouvez utiliser en même temps du contenu d'origine et du contenu créé, quelle que soit la structure de sortie que vous utilisez.

## Étapes suivantes

L'enrichissement par IA offre bien plus de possibilités, qui permettent d'exploiter au mieux les données contenues dans Stockage Azure. Vous pouvez notamment combiner les services Cognitive Services de différentes manières ou créer des compétences personnalisées s'il n'existe pas de service cognitif pour le scénario. Pour en savoir plus, suivez les liens ci-dessous.

- [Charger, télécharger et répertorier des blobs à l'aide du portail Azure \(Stockage blob Azure\)](#)
- [Configurer un indexeur blob \(Recherche cognitive Azure\)](#)
- [Vue d'ensemble de l'enrichissement de l'IA \(Recherche cognitive Azure\)](#)
- [Créer un ensemble de compétences \(Recherche cognitive Azure\)](#)
- [Mapper des nœuds dans une arborescence d'annotation \(Recherche cognitive Azure\)](#)

# Comment indexer des documents dans Stockage Blob Azure avec la Recherche cognitive Azure

04/10/2020 • 42 minutes to read • [Edit Online](#)

Cet article explique comment utiliser la Recherche cognitive Azure pour indexer des documents (tels que des fichiers PDF, des documents Microsoft Office et plusieurs autres formats courants) stockés dans le stockage d'objets blob Azure. Tout d'abord, il présente les concepts de base de la définition et de la configuration d'un indexeur d'objets blob. Ensuite, il offre une exploration plus approfondie des comportements et des scénarios que vous êtes susceptible de rencontrer.

## Formats de document pris en charge

L'indexeur d'objets blob peut extraire du texte à partir des formats de document suivants :

- PDF
- Formats Microsoft Office : DOCX/DOC/DOCM, XLSX/XLS/XLSM, PPTX/PPT/PPTM, MSG (e-mails Outlook), XML (WORD XML 2003 et 2006)
- Formats de document ouverts : ODT, ODS, ODP
- HTML
- XML
- ZIP
- GZ
- EPUB
- EML
- RTF
- Fichiers de texte brut (voir aussi [l'indexation de texte brut](#))
- JSON (consultez [l'indexation d'objets JSON blobs](#))
- CSV (consultez [Indexation d'objets blob CSV](#))

## Configuration de l'indexation d'objets blob

Vous pouvez configurer un indexeur de Stockage Blob Azure avec les outils suivants :

- [Azure portal](#)
- [API REST](#) de Recherche cognitive Azure
- [Kit de développement logiciel \(SDK\) .NET](#) de Recherche cognitive Azure

### NOTE

Certaines fonctionnalités (par exemple, les mappages de champs) ne sont pas encore disponibles dans le portail et doivent être utilisées par l'intermédiaire de programmes.

Ici, nous vous présentons le flux à l'aide de l'API REST.

### Étape 1 : Création d'une source de données

Une source de données spécifie les données à indexer, les informations d'identification nécessaires pour accéder aux données et les stratégies qui identifient efficacement les changements dans les données (telles que des lignes modifiées ou supprimées). Une source de données peut être utilisée par plusieurs indexeurs dans le même service de recherche.

Pour l'indexation des objets blob, la source de données doit avoir les propriétés requises suivantes :

- **name** est le nom unique de la source de données au sein de votre service de recherche.
- **type** doit être `azureblob`.
- **credentials** fournit la chaîne de connexion du compte de stockage en tant que paramètre `credentials.connectionString`. Pour plus d'informations, consultez [Comment spécifier des informations d'identification](#) ci-dessous.
- **container** spécifie un conteneur dans votre compte de stockage. Par défaut, tous les objets blob du conteneur sont récupérables. Si vous souhaitez indexer uniquement les objets blob dans un répertoire virtuel particulier, vous pouvez spécifier ce répertoire à l'aide du paramètre facultatif **query**.

Pour créer une source de données :

```
POST https://[service name].search.windows.net/datasources?api-version=2020-06-30
Content-Type: application/json
api-key: [admin key]

{
    "name" : "blob-datasource",
    "type" : "azureblob",
    "credentials" : { "connectionString" : "DefaultEndpointsProtocol=https;AccountName=<account name>;AccountKey=<account key>;" },
    "container" : { "name" : "my-container", "query" : "<optional-virtual-directory-name>" }
}
```

Pour plus d'informations sur l'API Créez une source de données, consultez [Créer une source de données](#).

#### Comment spécifier des informations d'identification

Vous pouvez fournir les informations d'identification du conteneur d'objets blob de l'une des manières suivantes :

- **Chaîne de connexion au compte de stockage avec accès complet :**

```
DefaultEndpointsProtocol=https;AccountName=<your storage account>;AccountKey=<your account key>
```

Vous pouvez obtenir la chaîne de connexion sur le portail Azure en sélectionnant le panneau du compte de stockage > Paramètres > Clés (pour les comptes de stockage Classic) ou en sélectionnant Paramètres > Clés d'accès (pour les comptes de stockage ARM).

- **Chaîne de connexion de la signature d'accès partagé (SAP) au compte de stockage :**

```
BlobEndpoint=https://<your account>.blob.core.windows.net/;SharedAccessSignature=?sv=2016-05-31&sig=<the signature>&spr=https&se=<the validity end time>&srt=co&ss=b&sp=r1
```

La SAP doit avoir les autorisations de liste et de lecture sur les conteneurs et les objets (blob en l'occurrence).

- **Signature d'accès partagé du conteneur :**

```
ContainerSharedAccessUri=https://<your storage account>.blob.core.windows.net/<container name>?sv=2016-05-31&sr=c&sig=<the signature>&se=<the validity end time>&sp=r1
```

La SAP doit avoir les autorisations de liste et lecture sur le conteneur.

Pour plus d'informations sur les signatures d'accès partagé au stockage, consultez [Utilisation des signatures d'accès partagé](#).

#### NOTE

Si vous utilisez des informations d'identification d'une SAP, vous devez mettre à jour les informations d'identification de la source de données régulièrement avec des signatures renouvelées afin d'éviter leur expiration. Si les informations d'identification de la SAP expirent, l'indexeur se bloque et affiche un message d'erreur similaire à `Credentials provided in the connection string are invalid or have expired.`.

### Étape 2 : Création d'un index

L'index spécifie les champs d'un document, les attributs et d'autres constructions qui façonnent l'expérience de recherche.

Voici comment créer un index avec un champ `content` pouvant faire l'objet d'une recherche afin de stocker le texte extrait d'objets blob :

```
POST https://[service name].search.windows.net/indexes?api-version=2020-06-30
Content-Type: application/json
api-key: [admin key]

{
    "name" : "my-target-index",
    "fields": [
        { "name": "id", "type": "Edm.String", "key": true, "searchable": false },
        { "name": "content", "type": "Edm.String", "searchable": true, "filterable": false,
    "sortable": false, "facetable": false }
    ]
}
```

Pour plus d'informations sur la création d'index, consultez [Création d'un index](#)

### Étape 3 : Créez un indexeur

Un indexeur connecte une source de données à un index de recherche cible et fournit une planification afin d'automatiser l'actualisation des données.

Une fois l'index et la source de données créés, vous êtes prêt à créer l'indexeur :

```
POST https://[service name].search.windows.net/indexers?api-version=2020-06-30
Content-Type: application/json
api-key: [admin key]

{
    "name" : "blob-indexer",
    "dataSourceName" : "blob-datasource",
    "targetIndexName" : "my-target-index",
    "schedule" : { "interval" : "PT2H" }
}
```

Cet indexeur s'exécutera toutes les deux heures (intervalle de planification défini sur « PT2H »). Pour exécuter un indexeur toutes les 30 minutes, définissez l'intervalle sur « PT30M ». Le plus court intervalle pris en charge est de 5 minutes. La planification est facultative : en cas d'omission, un indexeur ne s'exécute qu'une seule fois lorsqu'il est créé. Toutefois, vous pouvez à tout moment exécuter un indexeur à la demande.

Pour plus d'informations sur l'API Créez un indexeur, consultez [Créer un indexeur](#).

Pour plus d'informations sur la définition des planifications de l'indexeur, consultez [Comment planifier des indexeurs pour la Recherche cognitive Azure](#).

# Comment la Recherche cognitive Azure indexe les objets BLOB

En fonction de sa [configuration](#), l'indexeur d'objets blob peut indexer uniquement les métadonnées de stockage (une fonctionnalité utile lorsque vous ne vous préoccupez que des métadonnées et n'avez pas besoin d'indexer le contenu des objets blob), le stockage et le contenu des métadonnées, ou les métadonnées et le contenu textuel. Par défaut, l'indexeur extrait les métadonnées et le contenu.

## NOTE

Par défaut, les objets blob avec contenu structuré tels que JSON ou CSV sont indexés en tant que bloc de texte unique. Si vous souhaitez indexer des objets blob JSON et CSV de manière structurée, consultez [Indexation d'objets blob JSON](#) et [Indexation d'objets blob CSV](#) pour en savoir plus.

Un document composé ou incorporé (tel qu'une archive ZIP ou un document Word avec e-mail Outlook incorporé intégrant des pièces jointes, ou un fichier .MSG avec des pièces jointes) est également indexé en tant que document unique. Par exemple, toutes les images extraites des pièces jointes d'un fichier .MSG seront renvoyées dans le champ normalized\_images.

- Le contenu de texte du document est extrait dans un champ de chaîne nommé `content`.

## NOTE

La Recherche cognitive Azure limite la quantité de texte extrait en fonction du niveau tarifaire : 32 000 caractères pour le niveau Gratuit, 64 000 pour le niveau De base, 4 millions pour le niveau Standard, 8 millions pour le niveau Standard S2 et 16 millions pour le niveau Standard S3. Un avertissement est inclus dans la réponse d'état de l'indexeur pour les documents tronqués.

- Les propriétés de métadonnées spécifiées par l'utilisateur qui sont éventuellement présentes dans l'objet blob sont extraites textuellement. Notez que cela nécessite la définition d'un champ dans l'index portant le même nom que la clé de métadonnées du BLOB. Par exemple, si votre BLOB a une clé de métadonnées de `Sensitivity` avec la valeur `High`, vous devez définir un champ nommé `Sensitivity` dans votre index de recherche et il sera rempli avec la valeur `High`.
- Les propriétés de métadonnées d'objet blob standard sont extraites dans les champs suivants :
  - metadata\_storage\_name** (Edm.String) : nom de fichier de l'objet blob. Par exemple, si vous disposez de l'objet blob /my-container/my-folder/subfolder/resume.pdf, ce champ présente la valeur `resume.pdf`.
  - metadata\_storage\_path** (Edm.String) : URI complet de l'objet blob, incluant le compte de stockage. Par exemple :  
`https://myaccount.blob.core.windows.net/my-container/my-folder/subfolder/resume.pdf`
  - metadata\_storage\_content\_type** (Edm.String) : type de contenu tel que spécifié par le code que vous avez utilisé pour charger l'objet blob. Par exemple : `application/octet-stream`.
  - metadata\_storage\_last\_modified** (Edm.DateTimeOffset) : horodateur de la dernière modification de l'objet blob. La Recherche cognitive Azure utilise cet horodateur pour identifier les objets blob modifiés afin d'éviter une réindexation complète après l'indexation initiale.
  - metadata\_storage\_size** (Edm.Int64) : taille de l'objet blob en octets.
  - metadata\_storage\_content\_md5** (Edm.String) : code de hachage MD5 du contenu de l'objet blob s'il est disponible.
  - metadata\_storage\_sas\_token** (Edm.String) : jeton SAS temporaire qui peut être utilisé par des [compétences personnalisées](#) pour accéder à l'objet blob. Ce jeton ne doit pas être stocké pour une utilisation ultérieure dans la mesure où il peut expirer.
- Les propriétés de métadonnées propres à chaque format de document sont extraites dans les

champs répertoriés [ici](#).

Vous n'avez pas besoin de définir les champs relatifs à chacune des propriétés ci-dessus dans votre index de recherche. Il vous suffit de capturer les propriétés dont vous devez disposer pour votre application.

#### NOTE

Les noms de champ figurant dans votre index existant diffèrent généralement des noms de champ générés lors de l'extraction de document. Dans ce cas, vous pouvez utiliser les **mappages de champs** pour mapper les noms de propriétés fournis par la Recherche cognitive Azure sur les noms de champs de votre index de recherche. Découvrez ci-dessous une exemple d'utilisation de mappage de champ.

### Définition des clés de document et des mappages de champs

Dans la Recherche cognitive Azure, la clé de document identifie un document de manière unique. Chaque index de recherche doit comporter exactement un champ de clé de type Edm.String. Ce champ de clé est nécessaire pour chaque document ajouté à l'index (il constitue en fait le seul champ obligatoire).

Vous devez déterminer avec soin le champ extrait que vous souhaitez mapper sur le champ de clé de votre index. Les candidats sont les suivants :

- **metadata\_storage\_name** : ce champ pourrait se révéler un choix commode, mais notez que (1) les noms ne sont pas forcément uniques, car vous pouvez disposer d'objets blob portant le même nom dans différents dossiers, et (2) le nom peut contenir des caractères qui ne sont pas valides dans les clés de document, comme des tirets. Vous pouvez gérer les caractères non valides en utilisant la [fonction de mappage de champs](#) `base64Encode`. Dans ce cas, pensez à encoder les clés de documents lorsque vous les transmettez dans des appels d'API, comme l'API Lookup. (Par exemple, dans .NET, vous pouvez utiliser la [méthode UriTokenEncode](#) à cet effet).
- **metadata\_storage\_path** : l'utilisation du chemin d'accès complet garantit l'unicité, mais le chemin d'accès contient invariablement des caractères `/` qui ne sont [pas valides dans une clé de document](#). Comme ci-dessus, vous avez la possibilité d'encoder les clés à l'aide de la [fonction](#) `base64Encode`.
- Si aucune des solutions ci-dessus n'est adaptée à votre cas, vous pouvez ajouter une propriété de métadonnées personnalisée aux objets blob. Toutefois, cette approche contraint votre processus de chargement d'objets blob à ajouter cette propriété de métadonnées à tous les objets blob. Étant donné que la clé est une propriété obligatoire, tous les objets blob dépourvus de cette propriété ne seront pas indexés.

#### IMPORTANT

En l'absence de mappage explicite pour le champ de clé dans l'index, la Recherche cognitive Azure utilise automatiquement `metadata_storage_path` en guise de clé et encode les valeurs de clés en base 64 (la deuxième option ci-dessus).

Pour cet exemple, sélectionnons le champ `metadata_storage_name` en tant que clé de document.

Supposons également que votre index comporte un champ de clé nommé `key` et un champ `fileSize` pour le stockage de la taille du document. Pour obtenir le résultat souhaité, spécifiez les mappages de champs ci-après lors de la création ou de la mise à jour de votre indexeur :

```
"fieldMappings" : [
    { "sourceFieldName" : "metadata_storage_name", "targetFieldName" : "key", "mappingFunction" :
    { "name" : "base64Encode" } },
    { "sourceFieldName" : "metadata_storage_size", "targetFieldName" : "fileSize" }
]
```

Pour regrouper tous ces éléments, utilisez le code ci-après pour ajouter des mappages de champs et activer le codage base 64 des clés pour un indexeur existant :

```
PUT https://[service name].search.windows.net/indexers/blob-indexer?api-version=2020-06-30
Content-Type: application/json
api-key: [admin key]

{
  "dataSourceName" : "blob-datasource",
  "targetIndexName" : "my-target-index",
  "schedule" : { "interval" : "PT2H" },
  "fieldMappings" : [
    { "sourceFieldName" : "metadata_storage_name", "targetFieldName" : "key", "mappingFunction" :
      { "name" : "base64Encode" } },
    { "sourceFieldName" : "metadata_storage_size", "targetFieldName" : "fileSize" }
  ]
}
```

#### NOTE

Pour en savoir plus sur les mappages de champs, consultez [cet article](#).

#### Que se passe-t-il si vous devez encoder un champ à utiliser comme clé, que vous souhaitez également pouvoir rechercher ?

Il peut arriver que vous deviez utiliser une version encodée d'un champ tel que `metadata_storage_path` en tant que clé, et que vous ayez également besoin de pouvoir rechercher ce champ (sans encodage). Pour résoudre ce dilemme, vous pouvez mapper ce champ à deux champs, l'un étant utilisé pour la clé, et l'autre à des fins de recherche. Dans l'exemple ci-dessous, le champ `key` (clé) contient le chemin d'accès encodé, tandis que le champ `path` (chemin) n'est pas encodé et sera utilisé en tant que champ pouvant faire l'objet d'une recherche dans l'index.

```
PUT https://[service name].search.windows.net/indexers/blob-indexer?api-version=2020-06-30
Content-Type: application/json
api-key: [admin key]

{
  "dataSourceName" : "blob-datasource",
  "targetIndexName" : "my-target-index",
  "schedule" : { "interval" : "PT2H" },
  "fieldMappings" : [
    { "sourceFieldName" : "metadata_storage_path", "targetFieldName" : "key", "mappingFunction" :
      { "name" : "base64Encode" } },
    { "sourceFieldName" : "metadata_storage_path", "targetFieldName" : "path" }
  ]
}
```

## Contrôle les objets blob indexés

Vous pouvez contrôler les objets BLOB qui sont indexés et ignorés.

#### Indexer uniquement les objets blob avec des extensions de fichier spécifiques

Vous pouvez indexer uniquement les objets blob avec des extensions de nom de fichier que vous spécifiez à l'aide du paramètre de configuration d'indexeur `indexedFileNameExtensions`. La valeur est une chaîne contenant une liste d'extensions de fichier séparées par des virgules (précédées d'un point). Par exemple, pour indexer uniquement les objets blob .PDF et .DOCX, procédez comme suit :

```

PUT https://[service name].search.windows.net/indexers/[indexer name]?api-version=2020-06-30
Content-Type: application/json
api-key: [admin key]

{
    ... other parts of indexer definition
    "parameters" : { "configuration" : { "indexedFileNameExtensions" : ".pdf,.docx" } }
}

```

### Exclusion d'objets blob avec des extensions de fichier spécifiques

Vous pouvez exclure de l'indexation des objets blob avec des extensions de nom de fichier spécifiques à l'aide du paramètre de configuration `excludedFileNameExtensions`. La valeur est une chaîne contenant une liste d'extensions de fichier séparées par des virgules (précédées d'un point). Par exemple, pour indexer tous les objets blob, sauf ceux qui ont les extensions .PNG et .JPEG, procédez comme suit :

```

PUT https://[service name].search.windows.net/indexers/[indexer name]?api-version=2020-06-30
Content-Type: application/json
api-key: [admin key]

{
    ... other parts of indexer definition
    "parameters" : { "configuration" : { "excludedFileNameExtensions" : ".png,.jpeg" } }
}

```

Si les paramètres `indexedFileNameExtensions` et `excludedFileNameExtensions` sont tous deux présents, la Recherche cognitive Azure regarde d'abord `indexedFileNameExtensions`, puis `excludedFileNameExtensions`. Cela signifie que, si la même extension de fichier est présente dans les deux listes, elle sera exclue de l'indexation.

## Contrôle des parties de l'objet blob à indexer

Vous pouvez contrôler les parties des objets blob à indexer à l'aide du paramètre de configuration `dataToExtract`. Il peut avoir les valeurs suivantes :

- `storageMetadata` : spécifie que seuls les propriétés standard **et les métadonnées** spécifiées par l'utilisateur sont indexés.
- `allMetadata` : spécifie que les métadonnées de stockage et les **métadonnées spécifiques du type de contenu** extraites du contenu des objets blob sont indexés.
- `contentAndMetadata` : spécifie que toutes les métadonnées et tous les contenus textuels extraits de l'objet blob sont indexés. Il s'agit de la valeur par défaut.

Par exemple, pour indexer uniquement les métadonnées de stockage, utilisez :

```

PUT https://[service name].search.windows.net/indexers/[indexer name]?api-version=2020-06-30
Content-Type: application/json
api-key: [admin key]

{
    ... other parts of indexer definition
    "parameters" : { "configuration" : { "dataToExtract" : "storageMetadata" } }
}

```

### Utilisation de métadonnées d'objets blob pour contrôler la manière dont les objets blob sont indexés

Les paramètres de configuration décrits ci-dessus s'appliquent à tous les objets blob. Parfois, vous pouvez souhaiter contrôler la manière dont *differents objets blob* sont indexés. Pour cela, vous pouvez

ajouter les propriétés et valeurs de métadonnées d'objets blob suivantes :

NOM DE LA PROPRIÉTÉ	VALEUR DE LA PROPRIÉTÉ	EXPLICATION
AzureSearch_Skip	"true"	Indique à l'indexeur d'objets blob d'ignorer complètement l'objet blob. Ni l'extraction des métadonnées, ni l'extraction de contenu n'est tentée. Cette propriété est utile lorsqu'un objet blob spécifique échoue à plusieurs reprises et interrompt le processus d'indexation.
AzureSearch_SkipContent	"true"	Équivaut au réglage "dataToExtract" : "allMetadata" décrit ci-dessus au niveau d'un objet blob donné.

## Gestion des erreurs

Par défaut, l'indexeur d'objets blob s'arrête dès qu'il rencontre un objet blob avec un type de contenu non pris en charge (par exemple, une image). Vous pouvez évidemment utiliser le paramètre `excludedFileNameExtensions` pour ignorer certains types de contenu. Toutefois, vous devrez peut-être indexer des objets blob sans connaître à l'avance tous les types de contenu possibles. Pour poursuivre l'indexation lorsqu'un type de contenu non pris en charge est détecté, définissez le paramètre de configuration `failOnUnsupportedContentType` sur `false` :

```
PUT https://[service name].search.windows.net/indexers/[indexer name]?api-version=2020-06-30
Content-Type: application/json
api-key: [admin key]

{
    ... other parts of indexer definition
    "parameters" : { "configuration" : { "failOnUnsupportedContentType" : false } }
}
```

Pour certains objets blob, le service Recherche cognitive Azure ne parvient pas à déterminer le type de contenu ou à traiter un document avec un autre type de contenu pris en charge. Pour ignorer ce mode d'échec, définissez le paramètre de configuration `failOnUnprocessableDocument` sur `false` :

```
"parameters" : { "configuration" : { "failOnUnprocessableDocument" : false } }
```

La Recherche cognitive Azure limite la taille des objets blob indexés. Ces limites sont documentées dans [Limites de service de Recherche cognitive Azure](#). Par défaut, les objets blob surdimensionnés sont traités comme des erreurs. Toutefois, vous pouvez toujours indexer des métadonnées de stockage d'objets blob surdimensionnés en définissant la valeur du paramètre configuration `indexStorageMetadataOnlyForOversizedDocuments` sur `true` :

```
"parameters" : { "configuration" : { "indexStorageMetadataOnlyForOversizedDocuments" : true } }
```

Vous pouvez également poursuivre l'indexation si des erreurs se produisent à tout moment du traitement, que ce soit durant l'analyse d'objets blob ou l'ajout de documents à un index. Pour ignorer un nombre spécifique d'erreurs, définissez les paramètres de configuration `maxFailedItems` et

`maxFailedItemsPerBatch` sur les valeurs souhaitées. Par exemple :

```
{  
    ... other parts of indexer definition  
    "parameters" : { "maxFailedItems" : 10, "maxFailedItemsPerBatch" : 10 }  
}
```

## Indexation incrémentielle et détection des suppressions

Si vous configurez l'exécution planifiée d'un indexeur d'objets blob, celui-ci réindexera uniquement les objets blob modifiés d'après leur horodateur `LastModified`.

### NOTE

Vous n'êtes pas contraint de spécifier une stratégie de détection des modifications ; l'indexation incrémentielle est activée automatiquement à votre intention.

Pour prendre en charge la suppression de documents, utilisez une approche de type « suppression réversible ». Si vous supprimez complètement les objets blob, les documents correspondants ne seront pas supprimés de l'index de recherche.

Il existe deux façons d'implémenter l'approche de suppression réversible. Les deux sont décrites ci-dessous.

### Suppression réversible native de blobs (préversion)

#### IMPORTANT

La prise en charge de la suppression réversible native de blobs est disponible en préversion. Les fonctionnalités en préversion sont fournies sans contrat de niveau de service et ne sont pas recommandées pour les charges de travail de production. Pour plus d'informations, consultez [Conditions d'Utilisation Supplémentaires relatives aux Évaluations Microsoft Azure](#). L'[API REST version 2020-06-30-Preview](#) fournit cette fonctionnalité. Il n'y a actuellement pas de prise en charge du portail ou du SDK .NET.

### NOTE

Lors de l'utilisation de la stratégie de suppression réversible native de blobs, les clés de document des documents de votre index doivent être une propriété blob ou des métadonnées blob.

Dans cette méthode, vous allez utiliser la fonctionnalité de [suppression réversible native de blobs](#) offerte par le stockage Blob Azure. Si la suppression réversible native de blobs est activée sur votre compte de stockage, votre source de données a un jeu natif de stratégies de suppression réversible, et si l'indexeur trouve un blob qui a été transféré à un état Supprimé de manière réversible, l'indexeur supprime ce document de l'index. La stratégie de suppression réversible native de blobs n'est pas prise en charge lors de l'indexation de blobs à partir d'Azure Data Lake Storage Gen2.

Utiliser les étapes suivantes :

1. Activez la [suppression réversible native pour le stockage Blob Azure](#). Nous vous recommandons de définir la stratégie de rétention sur une valeur bien supérieure à celle de la planification d'intervalle de votre indexeur. Ainsi, en cas de problème lors de l'exécution de l'indexeur ou si vous avez un grand nombre de documents à indexer, l'indexeur a tout le temps nécessaire pour traiter les blobs supprimés de manière réversible. Les indexeurs Recherche cognitive Azure suppriment un document de l'index uniquement s'ils traitent le blob alors que son état est

Supprimé de manière réversible.

2. Configurez une stratégie de détection des suppressions réversibles natives de blobs sur la source de données. Voici un exemple. Étant donné qu'il s'agit d'une fonctionnalité d'évaluation, vous devez utiliser l'API REST en préversion.
3. Exécutez l'indexeur ou configurez l'indexeur pour qu'il s'exécute selon une planification. Lorsque l'indexeur s'exécute et traite le blob, le document est supprimé de l'index.

```
PUT https://[service name].search.windows.net/datasources/blob-datasource?api-version=2020-06-30-Preview
Content-Type: application/json
api-key: [admin key]
{
    "name" : "blob-datasource",
    "type" : "azureblob",
    "credentials" : { "connectionString" : "<your storage connection string>" },
    "container" : { "name" : "my-container", "query" : null },
    "dataDeletionDetectionPolicy" : {
        "@odata.type" :"#Microsoft.Azure.Search.NativeBlobSoftDeleteDeletionDetectionPolicy"
    }
}
```

#### Réindexation des blobs non supprimés

Si vous supprimez un blob du stockage Blob Azure à l'aide de la suppression réversible native qui est activée sur votre compte de stockage, le blob passera à un état Supprimé de manière réversible, vous donnant la possibilité d'annuler la suppression de ce blob pendant la période de rétention. Quand une source de données Recherche cognitive Azure dispose d'une stratégie de suppression réversible native de blobs et que l'indexeur traite un blob supprimé de manière réversible, ce document est supprimé de l'index. Si la suppression de ce blob est annulée par la suite, l'indexeur ne réindexera pas toujours le blob en question. Cela est dû au fait que l'indexeur détermine les blobs à indexer en fonction du timestamp `LastModified` du blob. Lorsque la suppression de manière réversible d'un blob est annulée, son timestamp `LastModified` n'est pas mis à jour. Par conséquent, si l'indexeur a déjà traité des blobs dotés de timestamps `LastModified` plus récents que celui du blob dont la suppression a été annulée, il ne réindexe pas ce blob. Pour vous assurer qu'un blob dont la suppression a été annulée est réindexé, vous devez mettre à jour le timestamp `LastModified` du blob. Pour ce faire, vous pouvez réenregistrer les métadonnées de ce blob. Vous n'avez pas besoin de modifier les métadonnées, mais le fait de réenregistrer les métadonnées met à jour le timestamp `LastModified` du blob afin que l'indexeur sache qu'il doit réindexer ce dernier.

#### Suppression réversible à l'aide de métadonnées personnalisées

Dans cette méthode, vous utiliserez les métadonnées d'un blob pour indiquer à quel moment un document doit être supprimé de l'index de recherche.

Utiliser les étapes suivantes :

1. Ajoutez une paire clé-valeur de métadonnées personnalisées au blob pour indiquer à Recherche cognitive Azure qu'il est logiquement supprimé.
2. Configurez une stratégie de détection des colonnes de suppression réversible sur la source de données. Voici un exemple.
3. Une fois que l'indexeur a traité le blob et supprimé le document de l'index, vous pouvez supprimer le blob du stockage Blob Azure.

Par exemple, la stratégie suivante considère qu'un objet blob est supprimé s'il présente une propriété de métadonnées `IsDeleted` avec la valeur `true` :

```

PUT https://[service name].search.windows.net/datasources/blob-datasource?api-version=2020-06-30
Content-Type: application/json
api-key: [admin key]

{
    "name" : "blob-datasource",
    "type" : "azureblob",
    "credentials" : { "connectionString" : "<your storage connection string>" },
    "container" : { "name" : "my-container", "query" : null },
    "dataDeletionDetectionPolicy" : {
        "@odata.type" :"#Microsoft.Azure.Search.SoftDeleteColumnDeletionDetectionPolicy",
        "softDeleteColumnName" : "IsDeleted",
        "softDeleteMarkerValue" : "true"
    }
}

```

#### Réindexation des blobs non supprimés

Si vous définissez une stratégie de détection des colonnes de suppression réversible sur votre source de données, puis que vous ajoutez des métadonnées personnalisées à un blob avec la valeur du marqueur et exécutez l'indexeur, ce document sera supprimé de l'index. Si vous souhaitez réindexer ce document, il vous suffit de modifier la valeur des métadonnées de suppression réversible pour ce blob et de réexécuter l'indexeur.

## Indexation de jeux de données volumineux

L'indexation d'objets blob peut être un processus long. Dans le cas où vous avez des millions d'objets blob à indexer, vous pouvez accélérer l'indexation en partitionnant les données et en utilisant plusieurs indexeurs pour traiter les données en parallèle. Par exemple, vous pouvez effectuer la configuration suivante :

- Partitionnez les données dans plusieurs conteneurs d'objets blob ou des dossiers virtuels.
- Configurez plusieurs sources de données de Recherche cognitive Azure, une par conteneur ou dossier. Pour pointer vers un dossier d'objets blob, utilisez le paramètre `query` :

```

{
    "name" : "blob-datasource",
    "type" : "azureblob",
    "credentials" : { "connectionString" : "<your storage connection string>" },
    "container" : { "name" : "my-container", "query" : "my-folder" }
}

```

- Créez un indexeur correspondant pour chaque source de données. Tous les indexeurs peuvent pointer vers le même index de recherche cible.
- Une unité de recherche dans votre service peut exécuter un indexeur à tout moment donné. La création de plusieurs indexeurs comme décrit ci-dessus est utile uniquement s'ils s'exécutent en parallèle. Pour exécuter plusieurs indexeurs en parallèle, effectuez un scale-out de votre service de recherche en créant un nombre approprié de partitions et répliques. Par exemple, si votre service de recherche a 6 unités de recherche (2 partitions x 3 répliques), 6 indexeurs peuvent s'exécuter simultanément, ce qui augmente le débit d'indexation par six. Pour plus d'informations sur la mise à l'échelle et la planification de capacité, consultez [Mettre à l'échelle des niveaux de ressources pour les requêtes et indexation des charges de travail dans la Recherche cognitive Azure](#).

## Indexation de documents et des données associées

Vous pourriez souhaiter « rassembler » des documents provenant de plusieurs sources dans votre index.

Par exemple, vous pourriez souhaiter fusionner des textes de blobs avec d'autres métadonnées stockées dans la base de données Cosmos. Vous pouvez même utiliser le push de l'indexation des API ainsi que plusieurs indexeurs pour générer des documents de recherche à partir de plusieurs parties.

Pour ce faire, tous les indexeurs et les autres composants doivent s'accorder sur la clé de document. Pour plus d'informations sur cette rubrique, consultez [Indexer plusieurs sources de données Azure](#). Pour une procédure détaillée, consultez l'article externe : [Combine documents with other data in Azure Search](#) (Associer des documents à d'autres données dans la Recherche cognitive Azure).

## Indexation en texte brut

Si tous vos objets BLOB contiennent du texte brut dans le même encodage, vous pouvez améliorer considérablement les performances d'indexation à l'aide du **mode d'analyse de texte**. Pour utiliser le mode d'analyse de texte, définissez la `parsingMode` propriété configuration à `text` :

```
PUT https://[service name].search.windows.net/indexers/[indexer name]?api-version=2020-06-30
Content-Type: application/json
api-key: [admin key]

{
    ... other parts of indexer definition
    "parameters" : { "configuration" : { "parsingMode" : "text" } }
}
```

Par défaut, le `UTF-8` encodage est possible. Pour spécifier un encodage différent, utilisez la `encoding` propriété de configuration :

```
{
    ... other parts of indexer definition
    "parameters" : { "configuration" : { "parsingMode" : "text", "encoding" : "windows-1252" } }
}
```

## Propriétés de métadonnées propres au type de contenu

Le tableau ci-après récapitule le traitement appliqué pour chaque format de document et décrit les propriétés de métadonnées extraites par la Recherche cognitive Azure.

FORMAT DE DOCUMENT/TYPE DE CONTENU	PROPRIÉTÉS DE MÉTADONNÉES PROPRES AU TYPE DE CONTENU	DÉTAILS DU TRAITEMENT
HTML (text/html)	<code>metadata_content_encoding</code> <code>metadata_content_type</code> <code>metadata_language</code> <code>metadata_description</code> <code>metadata_keywords</code> <code>metadata_title</code>	Suppression du balisage HTML et extraction du texte
PDF (application/pdf)	<code>metadata_content_type</code> <code>metadata_language</code> <code>metadata_author</code> <code>metadata_title</code>	Extraction du texte, y compris les documents incorporés (à l'exclusion des images)

FORMAT DE DOCUMENT/TYPE DE CONTENU	PROPRIÉTÉS DE MÉTADONNÉES PROPRES AU TYPE DE CONTENU	DÉTAILS DU TRAITEMENT
DOCX (application/vnd.openxmlformats-officedocument.wordprocessingml.document)	<code>metadata_content_type</code> <code>metadata_author</code> <code>metadata_character_count</code> <code>metadata_creation_date</code> <code>metadata_last_modified</code> <code>metadata_page_count</code> <code>metadata_word_count</code>	Extraction du texte, y compris les documents incorporés
DOC (application/msword)	<code>metadata_content_type</code> <code>metadata_author</code> <code>metadata_character_count</code> <code>metadata_creation_date</code> <code>metadata_last_modified</code> <code>metadata_page_count</code> <code>metadata_word_count</code>	Extraction du texte, y compris les documents incorporés
DOCM (application/vnd.ms-word.document.macroenabled.12)	<code>metadata_content_type</code> <code>metadata_author</code> <code>metadata_character_count</code> <code>metadata_creation_date</code> <code>metadata_last_modified</code> <code>metadata_page_count</code> <code>metadata_word_count</code>	Extraction du texte, y compris les documents incorporés
WORD XML (application/vnd.ms-word2006ml)	<code>metadata_content_type</code> <code>metadata_author</code> <code>metadata_character_count</code> <code>metadata_creation_date</code> <code>metadata_last_modified</code> <code>metadata_page_count</code> <code>metadata_word_count</code>	Suppression du balisage XML et extraction du texte
WORD 2003 XML (application/vnd.ms-wordml)	<code>metadata_content_type</code> <code>metadata_author</code> <code>metadata_creation_date</code>	Suppression du balisage XML et extraction du texte
XLSX (application/vnd.openxmlformats-officedocument.spreadsheetml.sheet)	<code>metadata_content_type</code> <code>metadata_author</code> <code>metadata_creation_date</code> <code>metadata_last_modified</code>	Extraction du texte, y compris les documents incorporés
XLS (application/vnd.ms-excel)	<code>metadata_content_type</code> <code>metadata_author</code> <code>metadata_creation_date</code> <code>metadata_last_modified</code>	Extraction du texte, y compris les documents incorporés
XLSM (application/vnd.ms-excel.sheet.macroenabled.12)	<code>metadata_content_type</code> <code>metadata_author</code> <code>metadata_creation_date</code> <code>metadata_last_modified</code>	Extraction du texte, y compris les documents incorporés

FORMAT DE DOCUMENT/TYPE DE CONTENU	PROPRIÉTÉS DE MÉTADONNÉES PROPRES AU TYPE DE CONTENU	DÉTAILS DU TRAITEMENT
PPTX (application/vnd.openxmlformats-officedocument.presentationml.presentation)	metadata_content_type metadata_author metadata_creation_date metadata_last_modified metadata_slide_count metadata_title	Extraction du texte, y compris les documents incorporés
PPT (application/vnd.ms-powerpoint)	metadata_content_type metadata_author metadata_creation_date metadata_last_modified metadata_slide_count metadata_title	Extraction du texte, y compris les documents incorporés
PPTM (application/vnd.ms-powerpoint.presentation.macroenabled.12)	metadata_content_type metadata_author metadata_creation_date metadata_last_modified metadata_slide_count metadata_title	Extraction du texte, y compris les documents incorporés
MSG (application/vnd.ms-outlook)	metadata_content_type metadata_message_from metadata_message_from_email metadata_message_to metadata_message_to_email metadata_message_cc metadata_message_cc_email metadata_message_bcc metadata_message_bcc_email metadata_creation_date metadata_last_modified metadata_subject	Extrayez le texte, y compris celui des pièces jointes.  metadata_message_to_email , metadata_message_cc_email et metadata_message_bcc_email sont des collections de chaînes ; les autres champs sont des chaînes.
ODT (application/vnd.oasis.opendocument.text)	metadata_content_type metadata_author metadata_character_count metadata_creation_date metadata_last_modified metadata_page_count metadata_word_count	Extraction du texte, y compris les documents incorporés
ODS (application/vnd.oasis.opendocument.spreadsheet)	metadata_content_type metadata_author metadata_creation_date metadata_last_modified	Extraction du texte, y compris les documents incorporés
ODP (application/vnd.oasis.opendocument.presentation)	metadata_content_type metadata_author metadata_creation_date metadata_last_modified title	Extraction du texte, y compris les documents incorporés

FORMAT DE DOCUMENT/TYPE DE CONTENU	PROPRIÉTÉS DE MÉTADONNÉES PROPRES AU TYPE DE CONTENU	DÉTAILS DU TRAITEMENT
ZIP (application/zip)	metadata_content_type	Extraction du texte de tous les documents figurant dans l'archive
GZ (application/gzip)	metadata_content_type	Extraction du texte de tous les documents figurant dans l'archive
EPUB (application/epub+zip)	metadata_content_type metadata_author metadata_creation_date metadata_title metadata_description metadata_language metadata_keywords metadata_identifier metadata_publisher	Extraction du texte de tous les documents figurant dans l'archive
XML (application/xml)	metadata_content_type metadata_content_encoding	Suppression du balisage XML et extraction du texte
JSON (application/json)	metadata_content_type metadata_content_encoding	Extraction du texte REMARQUE : si vous devez extraire plusieurs champs de document à partir d'un objet blob JSON, consultez <a href="#">Indexation d'objets blob JSON</a> pour plus de détails
EML (message/rfc822)	metadata_content_type metadata_message_from metadata_message_to metadata_message_cc metadata_creation_date metadata_subject	Extraction du texte, y compris les pièces jointes
RTF (application/rtf)	metadata_content_type metadata_author metadata_character_count metadata_creation_date metadata_page_count metadata_word_count	Extraction du texte
Texte brut (text/plain)	metadata_content_type metadata_content_encoding	Extraction du texte

## Aidez-nous à améliorer Recherche cognitive Azure

Si vous souhaitez nous soumettre des demandes d'ajout de fonctionnalités ou des idées d'amélioration, contactez-nous sur notre [site UserVoice](#).

# Indexation d'objets blob pour produire plusieurs documents de recherche

04/10/2020 • 5 minutes to read • [Edit Online](#)

Par défaut, un indexeur d'objets blob traite le contenu d'un objet blob comme un document de recherche unique. Certaines valeurs **parsingMode** prennent en charge les scénarios où un objet blob individuel peut entraîner plusieurs documents de recherche. Les différents types de **parsingMode** qui permettent à un indexeur d'extraire plusieurs documents de recherche à partir d'un objet blob sont les suivants :

- `delimitedText`
- `jsonArray`
- `jsonLines`

## Clé de document de type un-à-plusieurs

Chaque document qui s'affiche dans un index de la Recherche cognitive Azure est identifié par une clé de document.

En l'absence de mode d'analyse et de mappage explicite pour le champ de clé dans l'index, la Recherche cognitive Azure **mappe** automatiquement la propriété `metadata_storage_path` en guise de clé. Ce mappage garantit que chaque objet blob apparaît sous la forme d'un document de recherche distinct.

Lorsque vous utilisez un des modes d'analyse répertoriées ci-dessus, un objet blob correspond à « plusieurs » documents de recherche, et la clé de document repose uniquement sur des métadonnées d'objets blob non adaptées. Pour contourner cette contrainte, la Recherche cognitive Azure est capable de générer une clé de document de type « un-à-plusieurs » pour chaque entité extraite à partir d'un objet blob. Cette propriété est nommée `AzureSearch_DocumentKey` et ajoutée à chaque entité extraite à partir de l'objet blob. La valeur de cette propriété est garantie unique pour chaque entité *parmi les objets blob*, et les entités seront affichées sous forme de documents de recherche distincts.

Par défaut, en l'absence de mappage explicite pour le champ d'index de clé, le champ `AzureSearch_DocumentKey` est mappé à celui-ci, à l'aide de la fonction de mappage de champs `base64Encode`.

## Exemple

Supposons que vous avez une définition d'index avec les champs suivants :

- `id`
- `temperature`
- `pressure`
- `timestamp`

Et que votre conteneur d'objets blob a des objets blob avec la structure suivante :

*Blob1.json*

```
{ "temperature": 100, "pressure": 100, "timestamp": "2019-02-13T00:00:00Z" }  
{ "temperature" : 33, "pressure" : 30, "timestamp": "2019-02-14T00:00:00Z" }
```

*Blob2.json*

```
{ "temperature": 1, "pressure": 1, "timestamp": "2018-01-12T00:00:00Z" }  
{ "temperature" : 120, "pressure" : 3, "timestamp": "2013-05-11T00:00:00Z" }
```

Lorsque vous créez un indexeur et définissez **parsingMode** sur `jsonLines` (sans spécifier de mappage explicite pour le champ de clé), le mappage suivant est implicitement appliqué.

```
{  
    "sourceFieldName" : "AzureSearch_DocumentKey",  
    "targetFieldName": "id",  
    "mappingFunction": { "name" : "base64Encode" }  
}
```

Cette configuration produit un index de Recherche cognitive Azure contenant les informations suivantes (ID codé en base64 raccourci par souci de concision)

ID	TEMPÉRATURE	PRESSION	TIMESTAMP
aHR0 ... YjEuanNvbjsx	100	100	2019-02-13T00:00:00Z
aHR0 ... YjEuanNvbjsy	33	30	2019-02-14T00:00:00Z
aHR0 ... YjluanNvbjsx	1	1	2018-01-12T00:00:00Z
aHR0 ... YjluanNvbjsy	120	3	2013-05-11T00:00:00Z

## Mappage de champ personnalisé pour le champ de clé d'index

En prenant la même définition d'index que l'exemple précédent, supposons que votre conteneur d'objets blob a des objets blob avec la structure suivante :

*Blob1.json*

```
recordid, temperature, pressure, timestamp  
1, 100, 100,"2019-02-13T00:00:00Z"  
2, 33, 30,"2019-02-14T00:00:00Z"
```

*Blob2.json*

```
recordid, temperature, pressure, timestamp  
1, 1, 1,"2018-01-12T00:00:00Z"  
2, 120, 3,"2013-05-11T00:00:00Z"
```

Quand vous créez un indexeur avec `delimitedText` **parsingMode**, il peut sembler naturel de configurer une fonction de mappage pour le champ de clé comme suit :

```
{  
    "sourceFieldName" : "recordid",  
    "targetFieldName": "id"  
}
```

Toutefois, ce mappage ne permet *pas* d'afficher 4 documents dans l'index, car le champ `recordid` n'est pas unique *parmi les objets blob*. Par conséquent, nous vous recommandons d'utiliser le mappage de champs implicite appliquée à partir de la propriété `AzureSearch_DocumentKey` pour le champ d'index de clé avec les modes d'analyse

« un-à-plusieurs ».

Si vous ne souhaitez pas configurer un mappage de champs explicite, assurez-vous que la valeur *sourceField* est distincte pour chaque entité **parmi tous les objets blob**.

**NOTE**

L'approche utilisée par `AzureSearch_DocumentKey`, visant à assurer l'unicité des entités extraites, est susceptible de changer. Par conséquent, ne vous fiez pas à sa valeur pour les besoins de votre application.

## Étapes suivantes

Si vous n'êtes pas déjà familiarisé avec la structure et le workflow de base de l'indexation d'objets blob, vous devez d'abord passer en revue [Indexation de Stockage Blob Azure avec la Recherche cognitive Azure](#). Pour plus d'informations sur les modes d'analyse pour les différents types de contenu blob, consultez les articles suivants.

[Indexation d'objets blob CSV](#) [Indexation d'objets blob JSON](#)

# Comment indexer des objets blob CSV en utilisant le mode d'analyse delimitedText et des indexeurs d'objets blob dans Recherche cognitive Azure

04/10/2020 • 3 minutes to read • [Edit Online](#)

Par défaut, l'[indexeur d'objets blob Recherche cognitive Azure](#) analyse les objets blob de texte délimité comme un bloc de texte unique. Toutefois, avec des objets blob contenant des données CSV, vous souhaitez généralement traiter chaque ligne dans l'objet blob comme un document distinct. Par exemple, vous pouvez analyser le texte délimité suivant dans deux documents contenant chacun les champs « id », « datePublished » et « tags » :

```
id, datePublished, tags
1, 2016-01-12, "azure-search,azure,cloud"
2, 2016-07-07, "cloud,mobile"
```

Cet article explique comment analyser les objets blob CSV avec un indexeur d'objets blob Recherche cognitive Azure en définissant le mode d'analyse `delimitedText`.

## NOTE

Suivez les recommandations de configuration de l'indexeur dans [Indexation un-à-plusieurs](#) pour générer plusieurs documents de recherche à partir d'un objet Blob Azure.

## Configuration de l'indexation CSV

Pour indexer des objets blob CSV, créez ou mettez à jour une définition d'indexeur avec le mode d'analyse `delimitedText` sur une demande [Créer un indexeur](#) :

```
{
  "name" : "my-csv-indexer",
  ... other indexer properties
  "parameters" : { "configuration" : { "parsingMode" : "delimitedText", "firstLineContainsHeaders" :
true } }
```

`firstLineContainsHeaders` Indique que la première ligne (non vide) de chaque objet blob contient des en-têtes. Si les objets blob ne contiennent pas de ligne d'en-tête initiale, les en-têtes doivent être spécifiés dans la configuration de l'indexeur :

```
"parameters" : { "configuration" : { "parsingMode" : "delimitedText", "delimitedTextHeaders" :
"id,datePublished,tags" } }
```

Vous pouvez personnaliser le caractère délimiteur à l'aide du paramètre de configuration `delimitedTextDelimiter`. Par exemple :

```
"parameters" : { "configuration" : { "parsingMode" : "delimitedText", "delimitedTextDelimiter" : "|" } }
```

**NOTE**

Actuellement, seul le format UTF-8 est pris en charge. Si vous devez prendre en charge d'autres encodages, faites-le-nous savoir sur [UserVoice](#).

**IMPORTANT**

Lorsque vous utilisez le mode d'analyse de texte délimité, Recherche cognitive Azure suppose que tous les objets blob dans votre source de données seront de type CSV. Si vous devez prendre en charge une combinaison d'objets blob CSV et autres dans la même source de données, faites-le nous savoir sur [UserVoice](#).

## Exemples de requête

En résumé, voici des exemples complets de charges utiles.

Source de données :

```
POST https://[service name].search.windows.net/datasources?api-version=2020-06-30
Content-Type: application/json
api-key: [admin key]

{
    "name" : "my-blob-datasource",
    "type" : "azureblob",
    "credentials" : { "connectionString" : "DefaultEndpointsProtocol=https;AccountName=<account name>;AccountKey=<account key>;" },
    "container" : { "name" : "my-container", "query" : "<optional, my-folder>" }
}
```

Indexeur :

```
POST https://[service name].search.windows.net/indexers?api-version=2020-06-30
Content-Type: application/json
api-key: [admin key]

{
    "name" : "my-csv-indexer",
    "dataSourceName" : "my-blob-datasource",
    "targetIndexName" : "my-target-index",
    "parameters" : { "configuration" : { "parsingMode" : "delimitedText", "delimitedTextHeaders" : "id,datePublished,tags" } }
}
```

## Aidez-nous à améliorer Recherche cognitive Azure

Si vous avez des suggestions de fonctionnalités ou des idées d'amélioration, faites-le-nous savoir [UserVoice](#).

# Guide pratique pour indexer des objets blob JSON avec un indexeur d'objets blob dans Recherche cognitive Azure

04/10/2020 • 34 minutes to read • [Edit Online](#)

Cet article explique comment configurer un [indexeur](#) d'objets Blob Recherche cognitive Azure pour extraire le contenu structuré de documents JSON dans Stockage Blob Azure et pouvoir le rechercher dans Recherche cognitive Azure. Ce flux de travail crée un index Recherche cognitive Azure et le charge avec le texte existant extrait d'objets Blob JSON.

Vous pouvez utiliser le [portail](#), l'[API REST](#) ou le [SDK .NET](#) pour indexer du contenu JSON. Dans toutes les approches, les documents JSON sont généralement situés dans un conteneur d'objets Blob, dans un compte de Stockage Azure. Pour obtenir des conseils sur l'envoi (push) de documents JSON depuis d'autres plateformes qu'Azure, consultez [Importation de données dans Recherche cognitive Azure](#).

Les objets Blob JSON dans Stockage Blob Azure se composent généralement d'un seul document JSON (le mode d'analyse est `json`) ou d'une collection d'entités. Pour les collections JSON, l'objet blob peut avoir un **tableau** d'éléments JSON bien formés (le mode d'analyse est `jsonArray`). Les objets Blob peuvent également être composés de plusieurs entités JSON individuelles séparées par un saut de ligne (le mode d'analyse est `jsonLines`). Le paramètre **parsingMode** sur la demande détermine les structures de sortie.

## NOTE

Pour plus d'informations sur l'indexation de plusieurs documents de recherche à partir d'un seul objet blob, consultez [indexation un-à-plusieurs](#).

## Utiliser le portail

La méthode la plus simple pour l'indexation de documents JSON consiste à utiliser un Assistant dans le [portail Azure](#). En analysant les métadonnées dans le conteneur d'objets blob Azure, l'Assistant [Importation de données](#) peut créer un index par défaut, mapper des champs sources aux champs d'index cibles et charger l'index en une seule opération. Selon la taille et la complexité de la source de données, vous pouvez obtenir un index de recherche en texte intégral opérationnel en quelques minutes.

Nous vous recommandons d'utiliser la même région ou le même emplacement pour Recherche cognitive Azure et Stockage Azure pour obtenir une latence plus faible et pour éviter les frais de bande passante.

### 1 - Préparez les données sources

Connectez-vous au [portail Azure](#) et [créez un de conteneur d'objets blob](#) pour accueillir vos données. Le niveau d'accès public peut être défini sur l'une de ses valeurs valides.

Vous aurez besoin du nom du compte de stockage, du nom du conteneur et d'une clé d'accès pour récupérer vos données dans l'Assistant [Importation de données](#).

### 2 - Démarrez l'Assistant Importation de données

Dans la page Vue d'ensemble de votre service de recherche, vous pouvez [démarrer l'Assistant](#) à partir de la barre de commandes.

### 3 - Définissez la source de données

Dans la page **Source de données**, la source doit être **Stockage Blob Azure**, avec les spécifications suivantes :

- **Données à extraire** doit être *Contenu et métadonnées*. Le choix de cette option permet à l'Assistant d'inférer un schéma d'index et de mapper les champs pour l'importation.
- **Mode d'analyse** doit être défini sur *JSON*, *Tableau JSON* ou *Lignes JSON*.

*JSON* considère chaque objet blob comme un seul document de recherche, qui apparaît comme élément indépendant dans les résultats de la recherche.

*Tableau JSON* s'applique aux objets Blob contenant des données JSON bien formées. Le JSON bien formé correspond à un tableau d'objets, ou à une propriété qui est un tableau d'objets et vous voulez que chaque élément soit considéré comme un document de recherche autonome et indépendant. Si les objets blob sont complexes et que vous ne choisissez pas *Tableau JSON*, l'objet blob tout entier est ingéré sous la forme d'un seul document.

*Lignes JSON* concerne les objets Blob composés de plusieurs entités JSON séparées par un saut de ligne, où vous voulez que chaque élément soit considéré comme un document de recherche autonome et indépendant. Si les objets Blob sont complexes et que vous ne choisissez pas le mode d'analyse *Lignes JSON*, l'objet Blob tout entier est alors ingéré sous la forme d'un seul document.

- **Conteneur de stockage** doit spécifier votre compte de stockage et votre conteneur, ou une chaîne de connexion dont la résolution aboutit au conteneur. Vous pouvez obtenir des chaînes de connexion sur la page du portail Service Blob.

**New data source**

\* Name: blobdatasource

Data to extract: Content and metadata

Parsing mode: JSON array

\* Storage container: blobstore/basicdemo

Or input a connection string

Blob folder: your/folder/here

### 4 – Ignorer la page « Enrichir le contenu » de l'Assistant

L'ajout de compétences cognitives (ou enrichissement) n'est pas une condition d'importation. Si vous n'avez pas besoin d'[ajouter un enrichissement de l'IA](#) à votre pipeline d'indexation, ignorez cette étape.

Pour ignorer cette étape, cliquez sur les boutons bleus au bas de la page pour « Suivant » et « Ignorer ».

## 5 - Définissez les attributs de l'index

Dans la page **Index**, vous devez voir une liste de champs avec un type de données et une série de cases à cocher permettant de définir les attributs de l'index. L'Assistant peut générer une liste de champs basée sur les métadonnées et en échantillonnant les données sources.

Vous pouvez sélectionner des attributs en bloc en cliquant sur la case à cocher en haut de la colonne d'attribut. Choisissez **Récupérable** et **Possibilité de recherche** pour chaque champ qui doit être retourné vers une application cliente et soumis à un traitement de recherche de texte intégral. Vous remarquerez que les entiers ne peuvent pas être recherchés en texte intégral ou partiel (les nombres sont évalués textuellement et sont généralement utiles dans les filtres).

Pour plus d'informations, passez en revue la description des [attributs d'index](#) et des [analyseurs de langage](#).

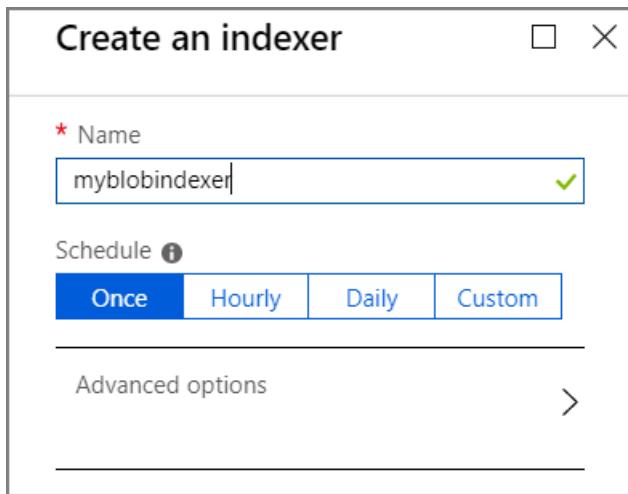
Prenez un moment pour passer en revue vos sélections. Une fois que vous exécutez l'Assistant, des structures de données physiques sont créées : vous ne pourrez donc plus modifier ces champs sans supprimer et recréer tous les objets.

Index						
Delete						
FIELD NAME	TYPE	RETRIEVABLE	FILTERABLE	SORTABLE	FACTETABLE	SEARCHABLE
content	Edm.String	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
metadata_storage_content_type	Edm.String	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
metadata_storage_size	Edm.Int64	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
metadata_storage_last_modified	Edm.DateTim...	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
metadata_storage_name	Edm.String	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
metadata_storage_path	Edm.String	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
metadata_content_encoding	Edm.String	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
metadata_content_type	Edm.String	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
metadata_language	Edm.String	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
metadata_title	Edm.String	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

## 6 - Créez un indexeur

Une fois que tout est spécifié, l'Assistant crée trois objets distincts dans votre service de recherche. Un objet source de données et un objet index sont enregistrés comme ressources nommées dans votre service Recherche cognitive Azure. La dernière étape crée un objet indexeur. Le fait de nommer l'indexeur lui permet d'exister comme ressource autonome, que vous pouvez planifier et gérer indépendamment de l'objet index et de l'objet source de données, créés dans la même séquence de l'Assistant.

Si vous n'êtes pas familiarisé avec les indexeurs, en voici une définition : un *indexeur* est une ressource dans Recherche cognitive Azure qui analyse une source de données externe et son contenu avec possibilité de recherche. La sortie de l'Assistant **Importation de données** est un indexeur qui analyse votre source de données JSON, extrait le contenu avec possibilité de recherche et l'importe dans un index sur Recherche cognitive Azure.



Cliquez sur **OK** pour exécuter l'Assistant et créer tous les objets. L'indexation commence immédiatement.

Vous pouvez surveiller l'importation des données dans les pages du portail. Des notifications de l'avancement indiquent l'état de l'indexation et le nombre de documents chargés.

Quand l'indexation est terminée, vous pouvez utiliser l'[Explorateur de recherche](#) pour interroger votre index.

#### NOTE

Si vous ne voyez pas les données attendues, vous devrez peut-être définir d'autres attributs sur d'autres champs.

Supprimez l'index et l'indexeur que vous venez de créer et réexécutez l'Assistant, en modifiant vos sélections pour les attributs d'index à l'étape 5.

## Utiliser les API REST

Vous pouvez utiliser l'API REST pour indexer des objets Blob JSON en suivant un flux de travail en trois parties commun à tous les indexeurs dans Recherche cognitive Azure : créer une source de données, créer un index, créer un indexeur. L'extraction de données du stockage d'objets Blob se produit lorsque vous envoyez la requête de création d'un indexeur. Lorsque cette requête est terminée, vous disposez d'un index pouvant être interrogé.

Vous pouvez consulter l'[exemple de code REST](#) à la fin de cette section qui montre comment créer les trois objets. Cette section fournit également des détails sur les [modes d'analyse JSON](#), les [objets Blob uniques](#), les [tableaux JSON](#) et les [tableaux imbriqués](#).

Pour l'indexation JSON basée sur le code, utilisez [Postman](#) et l'API REST pour créer ces objets :

- [index](#)
- [source de données](#)
- [indexeur](#)

L'ordre des opérations nécessite que vous créiez et appeliez des objets dans cet ordre. Contrairement au flux de travail du portail, une approche de code requiert un index pour accepter les documents JSON envoyés via la requête de **création d'un indexeur**.

Les objets Blob JSON dans Stockage Blob Azure se composent généralement d'un seul document JSON ou d'un tableau JSON. L'indexeur d'objets blob dans Recherche cognitive Azure peut analyser l'une ou l'autre de ces constructions selon la définition du paramètre **parsingMode** sur la requête.

DOCUMENT JSON	PARSING MODE	DESCRIPTION	DISPONIBILITÉ
Un seul par objet blob	json	Analyse les objets blob JSON comme un bloc de texte unique. Chaque objet blob JSON devient un document Recherche cognitive Azure unique.	Généralement disponible dans les API REST et le Kit de développement logiciel (SDK) .NET.
Plusieurs par objet blob	jsonArray	Analyse un tableau JSON dans l'objet blob, où chaque élément du tableau devient un document Recherche cognitive Azure distinct.	Généralement disponible dans les API REST et le Kit de développement logiciel (SDK) .NET.
Plusieurs par objet blob	jsonLines	Analyse un objet Blob qui contient plusieurs entités JSON (« tableau ») séparées par un saut de ligne, où chaque entité devient un document Recherche cognitive Azure distinct.	Généralement disponible dans les API REST et le Kit de développement logiciel (SDK) .NET.

## 1 - Assembler des entrées pour la requête

Pour chaque requête, vous devez fournir le nom du service et la clé d'administration pour Recherche cognitive Azure (dans l'en-tête POST), ainsi que le nom du compte de stockage et la clé pour le stockage d'objets Blob. Vous pouvez utiliser [Postman](#) pour envoyer des requêtes HTTP à Recherche cognitive Azure.

Copiez les quatre valeurs suivantes dans le bloc-notes pour pouvoir les coller dans une requête :

- Nom du service Recherche cognitive Azure
- Clé d'administration de Recherche cognitive Azure
- Nom du compte de stockage Azure
- Clé du compte de Stockage Azure

Vous pouvez trouver ces valeurs dans le portail :

1. Dans les pages du portail pour Recherche cognitive Azure, copiez l'URL du service de recherche dans la page Vue d'ensemble.
2. Dans le volet de navigation gauche, cliquez sur **Clés**, puis copiez la clé primaire ou secondaire (elles sont équivalentes).
3. Basculez vers les pages du portail pour votre compte de stockage. Dans le volet de navigation gauche, sous **Paramètres**, sélectionnez **Clés d'accès**. Cette page fournit le nom du compte et la clé. Copiez le nom du compte de stockage et une des clés dans le bloc-notes.

## 2 - Crédit d'une source de données

Cette étape consiste à fournir les informations de connexion à la source de données utilisées par l'indexeur. La source de données est un objet nommé dans Recherche cognitive Azure qui rend persistantes les informations de connexion. Le type de source de données, `azureblob`, détermine les comportements d'extraction de données appelés par l'indexeur.

Remplacez les valeurs valides des espaces réservés de nom du service, clé d'administration, compte de stockage et clé de compte.

```

POST https://[service name].search.windows.net/datasources?api-version=2020-06-30
Content-Type: application/json
api-key: [admin key for Azure Cognitive Search]

{
    "name" : "my-blob-datasource",
    "type" : "azureblob",
    "credentials" : { "connectionString" : "DefaultEndpointsProtocol=https;AccountName=<account name>;AccountKey=<account key>;" },
    "container" : { "name" : "my-container", "query" : "optional, my-folder" }
}

```

### 3 - Créer un index de recherche cible

Les indexeurs sont couplés à un schéma d'index. Si vous utilisez l'API (à la place du portail), préparez un index à l'avance pour pouvoir le spécifier sur l'opération de l'indexeur.

L'index stocke le contenu avec possibilité de recherche dans Recherche cognitive Azure. Pour créer un index, fournissez un schéma qui spécifie les champs d'un document, les attributs et d'autres constructions qui façonnent l'expérience de recherche. Si vous créez un index qui a les mêmes noms de champs et les mêmes types de données que la source, l'indexeur met en correspondance les champs sources et de destination, ce qui vous évite de devoir mapper explicitement les champs.

L'exemple suivant montre une demande [Créer un index](#). L'index aura un champ `content` avec possibilité de recherche pour stocker le texte extrait d'objets blob :

```

POST https://[service name].search.windows.net/indexes?api-version=2020-06-30
Content-Type: application/json
api-key: [admin key for Azure Cognitive Search]

{
    "name" : "my-target-index",
    "fields": [
        { "name": "id", "type": "Edm.String", "key": true, "searchable": false },
        { "name": "content", "type": "Edm.String", "searchable": true, "filterable": false,
        "sortable": false, "facetable": false }
    ]
}

```

### 4 - Configurer et exécuter l'indexeur

Comme c'est le cas pour l'index et la source de données, un indexeur est également objet nommé que vous créez et réutilisez sur un service Recherche cognitive Azure. Une requête complète pour créer un indexeur peut se présenter comme suit :

```

POST https://[service name].search.windows.net/indexers?api-version=2020-06-30
Content-Type: application/json
api-key: [admin key for Azure Cognitive Search]

{
    "name" : "my-json-indexer",
    "dataSourceName" : "my-blob-datasource",
    "targetIndexName" : "my-target-index",
    "schedule" : { "interval" : "PT2H" },
    "parameters" : { "configuration" : { "parsingMode" : "json" } }
}

```

La configuration de l'indexeur se trouve dans le corps de la requête. Elle nécessite une source de données et un index cible vide qui existe déjà dans Recherche cognitive Azure.

La planification et les paramètres sont facultatifs. Si vous les omettez, l'indexeur s'exécute immédiatement en mode d'analyse `json`.

Cet indexeur particulier n'inclut pas les mappages de champs. Dans la définition de l'indexeur, vous pouvez ignorer les **mappages de champs** si les propriétés du document JSON source correspondent aux champs de votre index de recherche cible.

## Exemple REST

Cette section est un récapitulatif de toutes les requêtes utilisées pour la création d'objets. Pour une présentation des composants, consultez les sections précédentes de cet article.

### Requête de source de données

Tous les indexeurs nécessitent un objet de source de données qui fournit des informations de connexion aux données existantes.

```
POST https://[service name].search.windows.net/datasources?api-version=2020-06-30
Content-Type: application/json
api-key: [admin key for Azure Cognitive Search]

{
    "name" : "my-blob-datasource",
    "type" : "azureblob",
    "credentials" : { "connectionString" : "DefaultEndpointsProtocol=https;AccountName=<account name>;AccountKey=<account key>;" },
    "container" : { "name" : "my-container", "query" : "optional, my-folder" }
}
```

### Requête d'index

Tous les indexeurs nécessitent un index cible qui reçoit les données. Le corps de la requête définit le schéma d'index, composé de champs, attribué pour prendre en charge les comportements souhaités dans un index pouvant faire l'objet d'une recherche. Cet index doit être vide lorsque vous exécutez l'indexeur.

```
POST https://[service name].search.windows.net/indexes?api-version=2020-06-30
Content-Type: application/json
api-key: [admin key for Azure Cognitive Search]

{
    "name" : "my-target-index",
    "fields": [
        { "name": "id", "type": "Edm.String", "key": true, "searchable": false },
        { "name": "content", "type": "Edm.String", "searchable": true, "filterable": false,
        "sortable": false, "facetable": false }
    ]
}
```

### Requête d'indexeur

Cette requête montre un indexeur complètement spécifié. Il inclut des mappages de champs, qui ont été omis dans les exemples précédents. Rappelez-vous que « `schedule` », « `parameters` » et « `fieldMappings` » sont facultatifs tant qu'une valeur par défaut est disponible. L'omission de « `schedule` » entraîne une exécution immédiate de l'indexeur. Si « `parsingMode` » est omis, l'index utilise la valeur « `json` » par défaut.

La création de l'indexeur sur Recherche cognitive Azure déclenche l'importation des données. Elle s'exécute immédiatement, puis selon une planification si vous en avez fourni une.

```

POST https://[service name].search.windows.net/indexers?api-version=2020-06-30
Content-Type: application/json
api-key: [admin key for Azure Cognitive Search]

{
    "name" : "my-json-indexer",
    "dataSourceName" : "my-blob-datasource",
    "targetIndexName" : "my-target-index",
    "schedule" : { "interval" : "PT2H" },
    "parameters" : { "configuration" : { "parsingMode" : "json" } },
    "fieldMappings" : [
        { "sourceFieldName" : "/article/text", "targetFieldName" : "text" },
        { "sourceFieldName" : "/article/datePublished", "targetFieldName" : "date" },
        { "sourceFieldName" : "/article/tags", "targetFieldName" : "tags" }
    ]
}

```

## Utilisation du Kit de développement logiciel (SDK) .NET

Le Kit de développement logiciel (SDK) .NET est totalement identique à l'API REST. Nous vous recommandons de consulter la section précédente de l'API REST pour découvrir les concepts, les workflows et les exigences. Vous pouvez alors vous référer à la documentation de référence des API .NET suivante pour implémenter un indexeur JSON dans du code managé.

- [microsoft.azure.search.models.datasource](#)
- [microsoft.azure.search.models.datasourcetype](#)
- [microsoft.azure.search.models.index](#)
- [microsoft.azure.search.models.indexer](#)

## Modes d'analyse

Les objets Blob JSON peuvent prendre plusieurs formules. Le paramètre **parsingMode** de l'indexeur JSON détermine comment le contenu d'objet Blob JSON est analysé et structuré dans un index Recherche cognitive Azure :

PARSINGMODE	DESCRIPTION
<code>json</code>	Indexez chaque objet Blob comme un seul document. Il s'agit de la valeur par défaut.
<code>jsonArray</code>	Choisissez ce mode si vos objets Blob sont constitués de tableaux JSON et s'il faut que chaque élément du tableau devienne un document distinct dans Recherche cognitive Azure.
<code>jsonLines</code>	Choisissez ce mode si vos objets Blob sont constitués de plusieurs entités JSON, qui sont séparées par un saut de ligne, et que chacune doit devenir un document distinct dans Recherche cognitive Azure.

Vous pouvez considérer un document comme un élément individuel dans les résultats de la recherche. Si vous voulez que chaque élément du tableau apparaisse dans les résultats de la recherche comme élément indépendant, utilisez l'option `jsonArray` ou `jsonLines` le cas échéant.

Dans la définition de l'indexeur, vous pouvez utiliser des [mappages de champs](#) pour choisir les propriétés du document JSON source à utiliser pour remplir votre index de recherche cible. Pour le mode d'analyse `jsonArray`, si le tableau existe en tant que propriété de plus bas niveau, vous pouvez définir une racine de

document qui indique l'emplacement du tableau dans l'objet Blob.

#### IMPORTANT

Lorsque vous utilisez `json`, `jsonArray` ou `jsonLines`, Recherche cognitive Azure suppose que tous les objets Blob dans votre source de données contiennent JSON. Si vous devez prendre en charge une combinaison d'objets blob JSON et autres dans la même source de données, faites-le nous savoir sur [notre site UserVoice](#).

## Analyser des objets Blob JSON uniques

Par défaut, [l'indexeur d'objets blob Azure Search](#) analyse les objets blob JSON comme un bloc de texte unique. Vous souhaitez généralement conserver la structure de vos documents JSON. En guise d'exemple, prenons le document JSON suivant dans Stockage Blob Azure :

```
{  
    "article" : {  
        "text" : "A hopefully useful article explaining how to parse JSON blobs",  
        "datePublished" : "2016-04-13",  
        "tags" : [ "search", "storage", "howto" ]  
    }  
}
```

L'indexeur d'objets blob analyse le document JSON en un seul document Recherche cognitive Azure. L'indexeur charge un index en mettant en correspondance les champs « text », « datePublished » et « tags » de la source avec les champs cibles de même nom et de même type.

Comme indiqué, les mappages de champs ne sont pas nécessaires. Étant donné un index avec les champs « text », « datePublished » et « tags », l'indexeur d'objets blob peut déduire le mappage correct sans qu'un mappage de champs ne soit présent dans la requête.

## Analyser des tableaux JSON

Vous pouvez également utiliser l'option de tableau JSON. Cette option est utile lorsque les objets Blob contiennent un *tableau d'objets JSON bien formés* et que vous souhaitez que chaque élément devienne un document Recherche cognitive Azure distinct. Prenons par exemple l'objet blob JSON suivant. Vous pouvez remplir l'index Recherche cognitive Azure avec trois documents distincts contenant chacun les champs « id » et « text ».

```
[  
    { "id" : "1", "text" : "example 1" },  
    { "id" : "2", "text" : "example 2" },  
    { "id" : "3", "text" : "example 3" }  
]
```

Pour un tableau JSON, la définition de l'indexeur doit être similaire à l'exemple suivant. Notez que le paramètre `parsingMode` spécifie l'analyseur `jsonArray`. Spécifier l'analyseur correct et avoir les bonnes entrées de données sont les deux seules conditions spécifiques aux tableaux pour l'indexation d'objets Blob JSON.

```

POST https://[service name].search.windows.net/indexers?api-version=2020-06-30
Content-Type: application/json
api-key: [admin key]

{
    "name" : "my-json-indexer",
    "dataSourceName" : "my-blob-datasource",
    "targetIndexName" : "my-target-index",
    "schedule" : { "interval" : "PT2H" },
    "parameters" : { "configuration" : { "parsingMode" : "jsonArray" } }
}

```

Là encore, notez que les mappages des champs peuvent être omis. En supposant un index avec des champs nommés de façon identique « id » et « text », l'indexeur d'objets blob peut inférer le mappage correct sans une liste de mappages des champs explicites.

## Analyser des tableaux imbriqués

Pour les tableaux JSON comprenant des éléments imbriqués, vous pouvez spécifier un `documentRoot` pour indiquer une structure à plusieurs niveaux. Par exemple, si vos objets blob ressemblent à ceci :

```

{
    "level1" : {
        "level2" : [
            { "id" : "1", "text" : "Use the documentRoot property" },
            { "id" : "2", "text" : "to pluck the array you want to index" },
            { "id" : "3", "text" : "even if it's nested inside the document" }
        ]
    }
}

```

Utilisez cette configuration pour indexer le tableau contenu dans la propriété `level2` :

```

{
    "name" : "my-json-array-indexer",
    ... other indexer properties
    "parameters" : { "configuration" : { "parsingMode" : "jsonArray", "documentRoot" :
    "/level1/level2" } }
}

```

## Analyser des objets Blob séparés par des sauts de ligne

Si votre objet Blob contient plusieurs entités JSON séparées par un saut de ligne et que vous souhaitez que chaque élément devienne un document de Recherche cognitive Azure distinct, vous pouvez choisir l'option de lignes JSON. Prenons, par exemple, l'objet Blob JSON suivant. Vous pouvez remplir l'index Recherche cognitive Azure avec trois documents distincts contenant chacun les champs « id » et « text ».

```

{ "id" : "1", "text" : "example 1" }
{ "id" : "2", "text" : "example 2" }
{ "id" : "3", "text" : "example 3" }

```

Pour les lignes JSON, la définition de l'indexeur doit être similaire à l'exemple suivant. Notez que le paramètre `parsingMode` spécifie l'analyseur `jsonLines`.

```

POST https://[service name].search.windows.net/indexers?api-version=2020-06-30
Content-Type: application/json
api-key: [admin key]

{
    "name" : "my-json-indexer",
    "dataSourceName" : "my-blob-datasource",
    "targetIndexName" : "my-target-index",
    "schedule" : { "interval" : "PT2H" },
    "parameters" : { "configuration" : { "parsingMode" : "jsonLines" } }
}

```

Là encore, notez que les mappages des champs peuvent être omis, comme dans le mode d'analyse `jsonArray`

## Ajouter des mappages de champs

Quand les champs sources et cibles ne sont pas parfaitement alignés, vous pouvez définir une section de mappage de champs dans le corps de la requête pour des associations champ à champ explicites.

Recherche cognitive Azure ne peut actuellement pas indexer des documents JSON arbitraires directement, car il ne prend en charge que les types de données primitifs, les tableaux de chaînes et les points GeoJSON. Toutefois, vous pouvez utiliser les **mappages de champ** pour sélectionner des parties de votre document JSON et les intégrer dans les champs de niveau supérieur du document de recherche. Pour en savoir plus sur les principes de base des mappages de champs, consultez [Mappages de champs dans les indexeurs Recherche cognitive Azure](#).

Revenons à notre exemple de document JSON :

```

{
    "article" : {
        "text" : "A hopefully useful article explaining how to parse JSON blobs",
        "datePublished" : "2016-04-13"
        "tags" : [ "search", "storage", "howto" ]
    }
}

```

Prenons un index de recherche avec les champs suivants : `text` de type `Edm.String`, `date` de type `Edm.DateTimeOffset` et `tags` de type `Collection(Edm.String)`. Notez la différence entre « `datePublished` » dans la source et le champ `date` dans l'index. Pour mapper votre document JSON à la forme souhaitée, utilisez les mappages de champ suivants :

```

"fieldMappings" : [
    { "sourceFieldName" : "/article/text", "targetFieldName" : "text" },
    { "sourceFieldName" : "/article/datePublished", "targetFieldName" : "date" },
    { "sourceFieldName" : "/article/tags", "targetFieldName" : "tags" }
]

```

Les noms de champ source dans les mappages sont spécifiés selon la notation de [pointeur JSON](#). Vous débutez par une barre oblique pour faire référence à la racine de votre document JSON, puis sélectionnez la propriété souhaitée (au niveau arbitraire de l'imbrication) en utilisant un chemin d'accès séparé par des barres obliques avant.

Vous pouvez également faire référence à des éléments de tableau en utilisant un index de base zéro. Par exemple, pour sélectionner le premier élément du tableau « `tags` » dans l'exemple ci-dessus, utilisez un mappage de champ similaire au suivant :

```
{ "sourceFieldName" : "/article/tags/0", "targetFieldName" : "firstTag" }
```

#### NOTE

Si un nom de champ source dans un chemin de mappage de champ fait référence à une propriété qui n'existe pas dans JSON, ce mappage est ignoré sans erreur. Cela nous permet de prendre en charge les documents avec un schéma différent (cas fréquent). Comme il n'y a aucune validation, vous devez veiller à éviter les fautes de frappe dans la spécification du mappage de champ.

## Voir aussi

- [Indexeurs dans Recherche cognitive Azure](#)
- [Indexation de Stockage Blob Azure avec Recherche cognitive Azure](#)
- [Indexation d'objets blob CSV avec l'indexeur d'objets blob de Recherche cognitive Azure](#)
- [Tutoriel : Rechercher des données semi-structurées à partir de Stockage Blob Azure](#)

# Comment indexer des tables à partir du stockage de tables Azure avec la Recherche cognitive Azure

04/10/2020 • 12 minutes to read • [Edit Online](#)

Cet article montre comment utiliser la Recherche cognitive Azure pour indexer les données stockées dans le stockage de tables Azure.

## Configurer l'indexation du stockage de tables Azure

Vous pouvez configurer un indexeur de stockage de tables Azure à l'aide des ressources suivantes :

- [Azure portal](#)
- [API REST](#) de Recherche cognitive Azure
- [Kit de développement logiciel \(SDK\) .NET](#) de Recherche cognitive Azure

Ici, nous vous présentons le flux à l'aide de l'API REST.

### Étape 1 : Créer une source de données

Une source de données spécifie les données à indexer, les informations d'identification nécessaires pour accéder aux données et les stratégies qui permettent à la Recherche cognitive Azure d'identifier efficacement les changements dans les données.

Pour l'indexation des tables, la source de données doit avoir les propriétés suivantes :

- **name** est le nom unique de la source de données au sein de votre service de recherche.
- **type** doit être `azuretable`.
- Le paramètre **credentials** contient la chaîne de connexion du compte de stockage. Pour plus d'informations, consultez la section [Spécifier des informations d'identification](#).
- **container** définit le nom de table et une requête facultative.
  - Spécifiez le nom de la table à l'aide du paramètre `name`.
  - Si vous le souhaitez, spécifiez une requête en utilisant le paramètre `query`.

#### IMPORTANT

Si possible, utilisez un filtre sur PartitionKey pour de meilleures performances. Toute autre requête effectue une analyse complète, ce qui entraîne une dégradation des performances pour les grandes tables. Consultez la section [Considérations relatives aux performances](#).

Pour créer une source de données :

```
POST https://[service name].search.windows.net/datasources?api-version=2020-06-30
Content-Type: application/json
api-key: [admin key]

{
    "name" : "table-datasource",
    "type" : "azuretable",
    "credentials" : { "connectionString" : "DefaultEndpointsProtocol=https;AccountName=<account name>;AccountKey=<account key>" },
    "container" : { "name" : "my-table", "query" : "PartitionKey eq '123'" }
}
```

Pour plus d'informations sur l'API Crée une source de données, consultez [Créer une source de données](#).

#### Manières de spécifier des informations d'identification

Vous pouvez fournir les informations d'identification de la table de l'une des manières suivantes :

- **Chaîne de connexion au compte de stockage avec accès complet :**

`DefaultEndpointsProtocol=https;AccountName=<your storage account>;AccountKey=<your account key>` Vous pouvez obtenir la chaîne de connexion sur le portail Azure en sélectionnant le **panneau du compte de stockage > Paramètres > Clés** (pour les comptes de stockage Classic) ou en sélectionnant **Paramètres > Clés d'accès** (pour les comptes de stockage ARM).

- **Chaîne de connexion de la signature d'accès partagé (SAP) au compte de stockage :**

`TableEndpoint=https://<your account>.table.core.windows.net/;SharedAccessSignature=?sv=2016-05-31&sig=<the signature>&spr=https&se=<the validity end time>&srt=co&ss=t&sp=r`

La SAP doit avoir les autorisations de liste et de lecture sur les conteneurs (des tables en l'occurrence) et les objets (des lignes de table).

- **Signature d'accès partagé à une table :**

`ContainerSharedAccessUri=https://<your storage account>.table.core.windows.net/<table name>?tn=<table name>&sv=2016-05-31&sig=<the signature>&se=<the validity end time>&sp=r`

La signature d'accès partagé doit disposer d'autorisations de requête (lecture) sur la table.

Pour plus d'informations sur les signatures d'accès partagé au stockage, consultez [Utilisation des signatures d'accès partagé](#).

#### NOTE

Si vous utilisez des informations d'identification d'une signature d'accès partagé, vous devez mettre à jour les informations d'identification de la source de données régulièrement avec des signatures renouvelées afin d'éviter leur expiration. Si les informations d'identification d'une signature d'accès partagé expirent, l'indexeur échoue avec un message d'erreur tel que « Les informations d'identification fournies dans la chaîne de connexion sont invalides ou ont expiré. »

## Étape 2 : Création d'un index

L'index spécifie les champs d'un document, les attributs et d'autres constructions qui façonnent l'expérience de recherche.

Pour créer un index :

```
POST https://[service name].search.windows.net/indexes?api-version=2020-06-30
Content-Type: application/json
api-key: [admin key]

{
    "name" : "my-target-index",
    "fields": [
        { "name": "key", "type": "Edm.String", "key": true, "searchable": false },
        { "name": "SomeColumnInMyTable", "type": "Edm.String", "searchable": true }
    ]
}
```

Pour plus d'informations sur la création d'index, consultez [Création d'un index](#).

### Étape 3 : Créer un indexeur

Un indexeur connecte une source de données à un index de recherche cible et fournit une planification afin d'automatiser l'actualisation des données.

Une fois l'index et la source de données créés, vous êtes prêt à créer l'indexeur :

```
POST https://[service name].search.windows.net/indexers?api-version=2020-06-30
Content-Type: application/json
api-key: [admin key]

{
    "name" : "table-indexer",
    "dataSourceName" : "table-datasource",
    "targetIndexName" : "my-target-index",
    "schedule" : { "interval" : "PT2H" }
}
```

Cet indexeur s'exécute toutes les deux heures. (L'intervalle de planification est définie sur « PT2H ».) Pour exécuter un indexeur toutes les 30 minutes, définissez l'intervalle sur « PT30M ». Le plus court intervalle pris en charge est de 5 minutes. La planification est facultative : en cas d'omission, un indexeur ne s'exécute qu'une seule fois lorsqu'il est créé. Toutefois, vous pouvez à tout moment exécuter un indexeur à la demande.

Pour plus d'informations sur l'API [Créer un indexeur](#), consultez [Créer un indexeur](#).

Pour plus d'informations sur la définition des planifications de l'indexeur, consultez [Comment planifier des indexeurs pour la Recherche cognitive Azure](#).

## Gérer différents noms de champs

Les noms de champ figurant dans votre index existant diffèrent parfois des noms de propriétés dans votre table. Dans ce cas, vous pouvez utiliser les mappages de champs pour mapper les noms de propriété de la table aux noms de champ de votre index de recherche. Pour en savoir plus sur les mappages de champs, consultez [Les mappages de champs de l'indexeur de la Recherche cognitive Azure comblient les différences entre les sources de données et les index de recherche](#).

## Gérer les clés de document

Dans la Recherche cognitive Azure, la clé de document identifie un document de manière unique. Chaque index de recherche doit comporter exactement un champ de clé de type `Edm.String`. Ce champ de clé est nécessaire pour chaque document ajouté à l'index (il constitue en fait le seul champ obligatoire).

Puisque les lignes d'une table ont une clé composée, la Recherche cognitive Azure génère un champ synthétique appelé `key` qui est une concaténation des valeurs de la clé de partition et de la clé de ligne. Par

exemple, si la valeur PartitionKey d'une ligne est `PK1` et que RowKey est `RK1`, alors la valeur du champ `Key` est `PK1RK1`.

#### NOTE

La valeur `Key` peut contenir des caractères non valides dans les clés de document, par exemple des tirets. Vous pouvez traiter les caractères non valides à l'aide de la [fonction de mappage de champ `base64Encode`](#). Si vous procédez ainsi, n'oubliez pas d'utiliser également l'encodage Base64 sécurisé pour les URL lorsque vous transmettez des clés de document dans des appels d'API tels que Recherche.

## Indexation incrémentielle et détection des suppressions

Lorsque vous configurez un indexeur de table pour l'exécuter de manière planifiée, cet indexeur répertorie uniquement les lignes nouvelles ou mises à jour, comme le détermine la valeur `Timestamp` de la ligne. Vous n'êtes pas obligé de spécifier une stratégie de détection de modification. L'indexation incrémentielle est automatiquement activée pour vous.

Pour indiquer que certains documents doivent être supprimés de l'index, vous pouvez utiliser une stratégie de suppression réversible. Plutôt que de supprimer une ligne, ajoutez une propriété pour signaler sa suppression, puis configurez une stratégie de détection des suppressions réversibles sur la source de données. Par exemple, la stratégie suivante considère qu'une ligne est supprimée si elle a une propriété `IsDeleted` avec la valeur

`"true"` :

```
PUT https://[service name].search.windows.net/datasources?api-version=2020-06-30
Content-Type: application/json
api-key: [admin key]

{
    "name" : "my-table-datasource",
    "type" : "azuretable",
    "credentials" : { "connectionString" : "<your storage connection string>" },
    "container" : { "name" : "table name", "query" : "<query>" },
    "dataDeletionDetectionPolicy" : { "@odata.type" :
        "#Microsoft.Azure.Search.SoftDeleteColumnDeletionDetectionPolicy", "softDeleteColumnName" : "IsDeleted",
        "softDeleteMarkerValue" : "true" }
}
```

## Considérations relatives aux performances

Par défaut, la Recherche cognitive Azure utilise le filtre de requête suivant : `Timestamp >= HighWaterMarkValue`. Dans la mesure où les tables Azure n'ont pas d'index secondaire sur le champ `Timestamp`, ce type de requête nécessite une analyse de table complète et est donc lente pour les tables volumineuses.

Voici deux approches possibles pour améliorer les performances d'indexation de table. Ces deux approches s'appuient sur l'utilisation de partitions de table :

- Si vos données peuvent être partitionnées naturellement en plusieurs plages de partition, créez une source de données et un indexeur correspondant pour chaque plage. Maintenant, chaque indexeur n'a à traiter qu'une plage de partition spécifique, ce qui conduit à de meilleures performances pour les requêtes. Si les données à indexer ont un petit nombre de partitions fixes, c'est encore mieux : chaque indexeur procède uniquement à une analyse de partition. Par exemple, pour créer une source de données pour le traitement d'une plage de partition avec des clés de `000` à `100`, utilisez une requête semblable à celle-ci :

```
"container" : { "name" : "my-table", "query" : "PartitionKey ge '000' and PartitionKey lt '100' " }
```

- Si vos données sont partitionnées par date (par exemple, si vous créez une nouvelle partition chaque jour ou chaque semaine), envisagez l'approche suivante :
  - Utilisez une requête sous la forme : `(PartitionKey ge <TimeStamp>) and (other filters)`.
  - Surveillez la progression de l'indexeur avec [l'API Get Indexer Status](#) et mettez régulièrement à jour la condition `<TimeStamp>` de la requête sur la base de la dernière valeur de limite supérieure réussie.
  - Avec cette approche, si vous avez besoin de déclencher une réindexation complète, vous devez réinitialiser la requête de source de données en plus de la réinitialisation de l'indexeur.

## Aidez-nous à améliorer Recherche cognitive Azure

Si vous souhaitez nous soumettre des demandes d'ajout de fonctionnalités ou des idées d'amélioration, n'hésitez pas les proposer sur notre [site UserVoice](#).

# Guide pratique pour indexer des données Cosmos DB avec un indexeur dans Recherche cognitive Azure

04/10/2020 • 30 minutes to read • [Edit Online](#)

## IMPORTANT

L'API SQL est mise à la disposition générale. La prise en charge de l'API MongoDB, de l'API Gremlin et de l'API Cassandra est actuellement en préversion publique. Les fonctionnalités en préversion sont fournies sans contrat de niveau de service et ne sont pas recommandées pour les charges de travail de production. Pour plus d'informations, consultez [Conditions d'Utilisation Supplémentaires relatives aux Évaluations Microsoft Azure](#). Vous pouvez demander l'accès aux préversions en remplissant [ce formulaire](#). L'API REST version 2020-06-30-Preview fournit des fonctionnalités en préversion. La prise en charge du portail est actuellement limitée, et il n'existe pas de prise en charge du kit SDK .NET.

## WARNING

Seuls les collections Cosmos DB dotées d'une stratégie d'indexation configurée sur [Cohérent](#) sont prises en charge par Recherche cognitive Azure. L'indexation des collections dotées d'une stratégie d'indexation différée n'est pas recommandée et peut se traduire par des données manquantes. Les collections dont l'indexation est désactivée ne sont pas prises en charge.

Cet article vous montre comment configurer l'[indexeur](#) Azure Cosmos DB pour extraire le contenu et les rendre détectables dans Recherche cognitive Azure. Ce flux de travail crée un index Recherche cognitive Azure et le charge avec le texte existant extrait d'Azure Cosmos DB.

Étant donné que la terminologie peut prêter à confusion, il est important de souligner que l'[indexation Azure Cosmos DB](#) et l'[indexation Recherche cognitive Azure](#) sont des opérations distinctes, propres à chaque service. Avant de commencer l'indexation Recherche cognitive Azure, votre base de données Azure Cosmos DB doit déjà exister et contenir des données.

L'indexeur Cosmos DB dans Recherche cognitive Azure peut analyser les [éléments d'Azure Cosmos DB](#) accessibles via différents protocoles.

- Pour l'[API SQL](#), qui est en disponibilité générale, vous pouvez utiliser le [portail](#), l'[API REST](#) ou le [Kit de développement logiciel \(SDK\) .NET](#) pour créer la source de données et l'indexeur.
- Pour l'[API MongoDB \(préversion\)](#), vous pouvez utiliser le [portail](#) ou l'[API REST version 2020-06-30-Preview](#) pour créer la source de données et l'indexeur.
- Pour l'[API Cassandra \(préversion\)](#) et l'[API Gremlin \(préversion\)](#), vous pouvez uniquement utiliser l'[API REST version 2020-06-30-Preview](#) pour créer la source de données et l'indexeur.

## NOTE

Vous pouvez voter sur User Voice pour l'[API Table](#) si vous souhaitez la voir prise en charge dans Recherche cognitive Azure.

# Utiliser le portail

## NOTE

Le portail prend actuellement en charge l'API SQL et l'API MongoDB (préversion).

La méthode la plus simple pour l'indexation d'éléments Azure Cosmos DB consiste à utiliser un Assistant dans le [portail Azure](#). En analysant les données et en lisant les métadonnées dans le conteneur, l'Assistant **Importation de données** peut créer un index par défaut, mapper des champs sources aux champs d'index cibles et charger l'index en une seule opération. Selon la taille et la complexité de la source de données, vous pouvez obtenir un index de recherche en texte intégral opérationnel en quelques minutes.

Nous vous recommandons d'utiliser la même région ou le même emplacement pour Recherche cognitive Azure et Azure Cosmos DB pour bénéficier d'une latence plus faible et éviter des frais de bande passante.

## 1 - Préparez les données sources

Vous devez disposer d'un compte Cosmos DB, d'une base de données Azure Cosmos DB mappée à l'API SQL, à l'API MongoDB (préversion) ou à l'API Gremlin (préversion), et d'un contenu dans la base de données.

Assurez-vous que votre base de données Cosmos DB contient des données. L'[Assistant Importation de données](#) lit les métadonnées et effectue un échantillonnage des données pour déduire un schéma d'index, mais il charge également des données à partir de Cosmos DB. Si les données sont manquantes, l'Assistant s'arrête avec l'erreur « Erreur lors de la détection du schéma d'index à partir de la source de données : Impossible de générer un index prototype, car la source de données 'emptycollection' n'a retourné aucune donnée ».

## 2 - Démarrez l'Assistant Importation de données

Vous pouvez [démarrer l'Assistant](#) à partir de la barre de commandes dans la page du service Recherche cognitive Azure ou, si vous vous connectez à l'API SQL de Cosmos DB, vous pouvez cliquer sur **Ajouter Recherche cognitive Azure** dans la section **Paramètres** du volet de navigation gauche de votre compte Cosmos DB.



## 3 - Définissez la source de données

Dans la page **Source de données**, la source doit être **Cosmos DB**, avec les spécifications suivantes :

- Le **Nom** correspond au nom de l'objet source de données. Une fois créé, vous pouvez le choisir pour d'autres charges de travail.
- Un **compte Cosmos DB** doit être la chaîne de connexion primaire ou secondaire à partir de Cosmos DB au format suivant :

```
AccountEndpoint=https://<Cosmos DB account name>.documents.azure.com;AccountKey=<Cosmos DB auth key>;
```

- Pour les **collections MongoDB** des versions 3.2 et 3.6, utilisez le format suivant pour le compte Cosmos DB dans le portail Azure :  

```
AccountEndpoint=https://<Cosmos DB account name>.documents.azure.com;AccountKey=<Cosmos DB auth key>;ApiKind=MongoDb
```
- Pour les **graphes Gremlin et les tables Cassandra**, inscrivez-vous à la [préversion de l'indexeur contrôlé](#) pour accéder à la préversion et aux informations sur la façon de mettre en forme les informations d'identification.
- La **Base de données** correspond à une base de données existante à partir du compte.

- La **Collection** correspond à un conteneur de documents. Des documents doivent exister pour que l'importation réussisse.
- Une **Requête** peut être vide si vous souhaitez tous les documents, sinon vous pouvez saisir une requête qui sélectionne un sous-ensemble du document. Une **requête** est disponible seulement pour l'API SQL.

**Import data**

[Connect to your data](#) [Add cognitive search \(Optional\)](#) [Customize target index](#) [...](#)

Create and load a search index using data from an existing Azure data source in your current subscription. Azure Search crawls the data structure you provide, extracts searchable content, optionally enriches it with cognitive skills, and loads it into an index. [Learn more](#)

Data Source	Cosmos DB
* Name	my-cosmosdb-datasource-name
* Cosmos DB account	AccountEndpoint=https://westus-cosmosdb.documents.azure.com:... <a href="#">Choose an existing connection</a>
* Database	hotel-demo-db
* Collection	hotel-demo-coll
Query	<code>SELECT * FROM c WHERE c._ts &gt;= @HighWaterMark ORDER BY c._ts</code>

#### 4 – Ignorer la page « Enrichir le contenu » de l’Assistant

L’ajout de compétences cognitives (ou enrichissement) n’est pas une condition d’importation. Si vous n’avez pas besoin d’[ajouter un enrichissement de l’IA](#) à votre pipeline d’indexation, ignorez cette étape.

Pour ignorer cette étape, cliquez sur les boutons bleus au bas de la page pour « Suivant » et « Ignorer ».

#### 5 - Définissez les attributs de l’index

Dans la page **Index**, vous devez voir une liste de champs avec un type de données et une série de cases à cocher permettant de définir les attributs de l’index. L’Assistant peut générer une liste de champs basée sur les métadonnées et en échantillonnant les données sources.

Vous pouvez sélectionner des attributs en bloc en cliquant sur la case à cocher en haut de la colonne d’attribut. Choisissez **Récupérable** et **Possibilité de recherche** pour chaque champ qui doit être retourné vers une application cliente et soumis à un traitement de recherche de texte intégral. Vous remarquerez que les entiers ne peuvent pas être recherchés en texte intégral ou partiel (les nombres sont évalués textuellement et sont généralement utiles dans les filtres).

Pour plus d’informations, passez en revue la description des [attributs d’index](#) et des [analyseurs de langage](#).

Prenez un moment pour passer en revue vos sélections. Une fois que vous exécutez l’Assistant, des structures de données physiques sont créées : vous ne pourrez donc plus modifier ces champs sans supprimer et recréer tous les objets.

**Import data**

\* Key [?](#)  
HotelID

Suggester name  Search mode [?](#)

[Delete](#)

FIELD NAME	TYPE	RETRIEVABLE	FILTERABLE	SORTABLE	FACTETABLE	SEARCHABLE	ANALYZER	SUGGESTER
HotelID	Edm.String	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Standard - Lucene <input type="button" value="▼"/>	<input type="button" value="..."/>
HotelName	Edm.String	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Standard - Lucene <input type="button" value="▼"/>	<input type="button" value="..."/>
Description	Edm.String	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	English - Microsoft <input type="button" value="▼"/>	<input type="button" value="..."/>
Description_fr	Edm.String	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	French - Microsoft <input type="button" value="▼"/>	<input type="button" value="..."/>
Category	Edm.String	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Standard - Lucene <input type="button" value="▼"/>	<input type="button" value="..."/>
Tags	Edm.String	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Standard - Lucene <input type="button" value="▼"/>	<input type="button" value="..."/>
ParkingIncluded	Edm.Int64	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="button" value="..."/>	<input type="button" value="..."/>
SmokingAllowed	Edm.Int64	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="button" value="..."/>	<input type="button" value="..."/>
LastRenovationDate	Edm.DateTi...	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="button" value="..."/>	<input type="button" value="..."/>
Rating	Edm.Double	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="button" value="..."/>	<input type="button" value="..."/>

[Previous: Add cognitive search \(Optional\)](#) [Next: Create an indexer](#)

## 6 - Créez un indexeur

Une fois que tout est spécifié, l'Assistant crée trois objets distincts dans votre service de recherche. Un objet source de données et un objet index sont enregistrés comme ressources nommées dans votre service Recherche cognitive Azure. La dernière étape crée un objet indexeur. Le fait de nommer l'indexeur lui permet d'exister comme ressource autonome, que vous pouvez planifier et gérer indépendamment de l'objet index et de l'objet source de données, créés dans la même séquence de l'Assistant.

Si vous n'êtes pas familiarisé avec les indexeurs, en voici une définition : un *indexeur* est une ressource dans Recherche cognitive Azure qui analyse une source de données externe et son contenu avec possibilité de recherche. La sortie de l'Assistant **Importation de données** est un indexeur qui analyse votre source de données Cosmos DB, extrait le contenu avec possibilité de recherche et l'importe dans un index sur Recherche cognitive Azure.

La capture d'écran suivante montre la configuration de l'indexeur par défaut. Vous pouvez basculer sur **Une seule fois** si vous souhaitez exécuter l'indexeur une seule fois. Cliquez sur **Envoyer** pour exécuter l'Assistant et créer tous les objets. L'indexation commence immédiatement.

## Import data

Connect to your data Add cognitive search (Optional) Customize target index [Create an indexer](#)

### Indexer

\* Name

my-cosmosdb-indexer



Schedule [i](#)

Once

Hourly

Daily

Custom



Change tracking automatically configured with a high watermark policy.

Track deletions [i](#)



Description

(optional)

▼ Advanced options

[Previous: Customize target index](#)

**Submit**

Vous pouvez surveiller l'importation des données dans les pages du portail. Des notifications de l'avancement indiquent l'état de l'indexation et le nombre de documents chargés.

Quand l'indexation est terminée, vous pouvez utiliser l'[Explorateur de recherche](#) pour interroger votre index.

#### NOTE

Si vous ne voyez pas les données attendues, vous devrez peut-être définir d'autres attributs sur d'autres champs. Supprimez l'index et l'indexeur que vous venez de créer et réexécutez l'Assistant, en modifiant vos sélections pour les attributs d'index à l'étape 5.

## Utiliser les API REST

Vous pouvez utiliser l'API REST pour indexer des données Azure Cosmos DB en suivant un flux de travail en trois parties commun à tous les indexeurs dans Recherche cognitive Azure : créer une source de données, créer un index, créer un indexeur. L'extraction de données dans Cosmos DB se produit quand vous envoyez la demande de création d'un indexeur. Lorsque cette requête est terminée, vous disposez d'un index pouvant être interrogé.

#### NOTE

Pour indexer des données à partir de l'API Gremlin ou de l'API Cassandra de Cosmos DB, vous devez d'abord demander l'accès aux préversions contrôlées en remplissant [ce formulaire](#). Une fois votre demande traitée, vous recevez des instructions sur l'utilisation de l'[API REST version 2020-06-30-Preview](#) pour créer la source de données.

Plus haut dans cet article, il est mentionné que l'[indexation d'Azure Cosmos DB](#) et l'[indexation de Recherche cognitive Azure](#) sont des opérations distinctes. Pour l'indexation Cosmos DB, par défaut, tous les documents sont indexés automatiquement, sauf avec l'API Cassandra. Si vous désactivez l'indexation automatique, les documents sont accessibles seulement via leurs liens réflexifs ou des requêtes avec l'ID de document. L'indexation Recherche cognitive Azure nécessite l'activation de l'indexation automatique Cosmos DB dans la collection qui sera indexée par Recherche cognitive Azure. Lors de l'inscription à la préversion de l'indexeur de l'API Cassandra de Cosmos DB, vous recevez des instructions sur la configuration de l'indexation Cosmos DB.

#### **WARNING**

Azure Cosmos DB est la nouvelle génération de DocumentDB. Précédemment, avec l'API version **2017-11-11**, vous pouviez utiliser la syntaxe `documentdb`. Autrement dit, vous pouviez spécifier votre type de source de données en tant que `cosmosdb` ou `documentdb`. À compter de la version d'API **2019-05-06**, les APIS Recherche cognitive Azure et le portail prennent uniquement en charge la syntaxe `cosmosdb` comme indiqué dans cet article. Cela signifie que le type de source de données doit être `cosmosdb` si vous souhaitez vous connecter à un point de terminaison Cosmos DB.

## **1 - Assembler des entrées pour la requête**

Pour chaque requête, vous devez fournir le nom du service et la clé d'administration pour Recherche cognitive Azure (dans l'en-tête POST), ainsi que le nom du compte de stockage et la clé pour le stockage d'objets Blob. Vous pouvez utiliser [Postman](#) pour envoyer des requêtes HTTP à Recherche cognitive Azure.

Copiez les quatre valeurs suivantes dans le bloc-notes pour pouvoir les coller dans une requête :

- Nom du service Recherche cognitive Azure
- Clé d'administration de Recherche cognitive Azure
- Chaîne de connexion Cosmos DB

Vous pouvez trouver ces valeurs dans le portail :

1. Dans les pages du portail pour Recherche cognitive Azure, copiez l'URL du service de recherche dans la page Vue d'ensemble.
2. Dans le volet de navigation gauche, cliquez sur **Clés**, puis copiez la clé primaire ou secondaire (elles sont équivalentes).
3. Basculez vers les pages du portail pour votre compte de stockage Cosmos. Dans le volet de navigation gauche, sous **Paramètres**, sélectionnez **Clés**. Cette page fournit un URI, deux ensembles de chaînes de connexion, et deux ensembles de clés. Copiez une des chaînes de connexion dans le bloc-notes.

## **2 - Crédit d'une source de données**

Une **source de données** spécifie les données à indexer, les informations d'identification et les stratégies pour identifier les modifications des données (par exemple, les documents modifiés ou supprimés dans votre collection). La source de données est définie en tant que ressource indépendante de manière à pouvoir être utilisée par plusieurs indexeurs.

Pour créer une source de données, formulez une requête POST :

```

POST https://[service name].search.windows.net/datasources?api-version=2020-06-30
Content-Type: application/json
api-key: [Search service admin key]

{
    "name": "mycosmosdbdatasource",
    "type": "cosmosdb",
    "credentials": {
        "connectionString": "AccountEndpoint=https://myCosmosDbEndpoint.documents.azure.com;AccountKey=myCosmosDbAuthKey;Database=myCosmosDbDatabaseId"
    },
    "container": { "name": "myCollection", "query": null },
    "dataChangeDetectionPolicy": {
        "@odata.type": "#Microsoft.Azure.Search.HighWaterMarkChangeDetectionPolicy",
        "highWaterMarkColumnName": "_ts"
    }
}

```

Le corps de la requête contient la définition de la source de données, qui doit inclure les champs suivants :

CHAMP	DESCRIPTION
<b>name</b>	Obligatoire. Choisissez un nom pour représenter votre objet source de données.
<b>type</b>	Obligatoire. Doit être <code>cosmosdb</code> .
<b>credentials</b>	<p>Obligatoire. Doit être une chaîne de connexion Cosmos DB.</p> <p>Pour les <b>collections SQL</b>, les chaînes de connexion sont au format suivant :</p> <div style="border: 1px solid black; padding: 5px;"> <code>AccountEndpoint=https://&lt;Cosmos DB account name&gt;.documents.azure.com;AccountKey=&lt;Cosmos DB auth key&gt;;Database=&lt;Cosmos DB database id&gt;</code> </div> <p>Pour les <b>collections MongoDB</b> des versions 3.2 et 3.6, utilisez le format suivant pour la chaîne de connexion :</p> <div style="border: 1px solid black; padding: 5px;"> <code>AccountEndpoint=https://&lt;Cosmos DB account name&gt;.documents.azure.com;AccountKey=&lt;Cosmos DB auth key&gt;;Database=&lt;Cosmos DB database id&gt;;ApiKind=MongoDb</code> </div> <p>Pour les <b>graphes Gremlin et les tables Cassandra</b>, inscrivez-vous à la <a href="#">préversion de l'indexeur contrôlé</a> pour accéder à la préversion et aux informations sur la façon de mettre en forme les informations d'identification.</p> <p>Évitez les numéros de port dans l'URL du point de terminaison. Si vous incluez le numéro de port, Recherche cognitive Azure ne peut pas indexer votre base de données Azure Cosmos DB.</p>

CHAMP	DESCRIPTION
<b>container</b>	Contient les éléments suivants : <b>nom</b> : Obligatoire. Spécifiez l'ID de la collection de bases de données à indexer. <b>query</b> : facultatif. Vous pouvez spécifier une requête pour obtenir un schéma plat à partir d'un document JSON arbitraire de manière à ce qu'Azure Search puisse procéder à l'indexation. Pour l'API MongoDB, l'API Gremlin et l'API Cassandra, les requêtes ne sont pas prises en charge.
<b>dataChangeDetectionPolicy</b>	Recommandé. Consultez la section <a href="#">Indexation des documents modifiés</a> .
<b>dataDeletionDetectionPolicy</b>	facultatif. Consultez la section <a href="#">Indexation des documents supprimés</a> .

### Utilisation de requêtes pour formater les données indexées

Vous pouvez spécifier une requête SQL pour aplatiser les propriétés ou les tableaux imbriqués, projeter des propriétés JSON et filtrer les données à indexer.

#### WARNING

Les requêtes personnalisées ne sont pas prises en charge pour l'**API MongoDB**, l'**API Gremlin** et l'**API Cassandra** : le paramètre `container.query` doit être défini sur null ou être omis. Si vous avez besoin d'utiliser une requête personnalisée, indiquez-le nous sur [UserVoice](#).

Exemple de document :

```
{
  "userId": 10001,
  "contact": {
    "firstName": "andy",
    "lastName": "hoh"
  },
  "company": "microsoft",
  "tags": ["azure", "cosmosdb", "search"]
}
```

Requête de filtre :

```
SELECT * FROM c WHERE c.company = "microsoft" and c._ts >= @HighWaterMark ORDER BY c._ts
```

Requête d'aplatissement :

```
SELECT c.id, c.userId, c.contact.firstName, c.contact.lastName, c.company, c._ts FROM c WHERE c._ts >=
@HighWaterMark ORDER BY c._ts
```

Requête de projection :

```
SELECT VALUE { "id":c.id, "Name":c.contact.firstName, "Company":c.company, "_ts":c._ts } FROM c WHERE
c._ts >= @HighWaterMark ORDER BY c._ts
```

Requête d'aplatissage de tableau :

```
SELECT c.id, c.userId, tag, c._ts FROM c JOIN tag IN c.tags WHERE c._ts >= @HighWaterMark ORDER BY c._ts
```

### 3 - Créer un index de recherche cible

Créez un index Recherche cognitive Azure cible si vous n'en avez pas. L'exemple suivant crée un index avec un champ ID et un champ Description :

```
POST https://[service name].search.windows.net/indexes?api-version=2020-06-30
Content-Type: application/json
api-key: [Search service admin key]

{
  "name": "mysearchindex",
  "fields": [
    {
      "name": "id",
      "type": "Edm.String",
      "key": true,
      "searchable": false
    },
    {
      "name": "description",
      "type": "Edm.String",
      "filterable": false,
      "sortable": false,
      "facetable": false,
      "suggestions": true
    }
  ]
}
```

Assurez-vous que le schéma de votre index cible est compatible avec le schéma des documents JSON source ou la sortie de votre projection de requête personnalisée.

#### NOTE

Pour les collections partitionnées, la clé de document par défaut est la propriété `_rid` d'Azure Cosmos DB, que Recherche cognitive Azure renomme automatiquement en `rid`, car les noms de champ ne peuvent pas commencer par un trait de soulignement. De même, les valeurs `_rid` d'Azure Cosmos DB contiennent des caractères non valides dans les clés de Recherche cognitive Azure. Par conséquent, les valeurs `_rid` sont codées en Base64.

Pour les collections MongoDB, Recherche cognitive Azure renomme automatiquement la propriété `_id`_id`.

### Mappage entre les types de données JSON et les types de données Azure Search

TYPE DE DONNÉES JSON	TYPES DE CHAMPS D'INDEX CIBLE COMPATIBLES
Bool	Edm.Boolean, Edm.String
Nombres qui ressemblent à des nombres entiers	Edm.Int32, Edm.Int64, Edm.String
Nombres qui ressemblent à des nombres avec points flottants	Edm.Double, Edm.String
String	Edm.String
Tableaux de types primitifs, par exemple ["a", "b", "c"]	Collection(Edm.String)

TYPE DE DONNÉES JSON	TYPES DE CHAMPS D'INDEX CIBLE COMPATIBLES
Chaînes qui ressemblent à des dates	Edm.DateTimeOffset, Edm.String
Objets GeoJSON, par exemple { "type": "Point", "coordinates": [long, lat] }	Edm.GeographyPoint
Autres objets JSON	N/A

#### 4 - Configurer et exécuter l'indexeur

Une fois l'index et la source de données créés, vous êtes prêt à créer l'indexeur :

```
POST https://[service name].search.windows.net/indexers?api-version=2020-06-30
Content-Type: application/json
api-key: [admin key]

{
  "name" : "mycosmosdbindexer",
  "dataSourceName" : "mycosmosdbdatasource",
  "targetIndexName" : "mysearchindex",
  "schedule" : { "interval" : "PT2H" }
}
```

Cet indexeur s'exécute toutes les deux heures (intervalle de planification défini sur « PT2H »). Pour exécuter un indexeur toutes les 30 minutes, définissez l'intervalle sur « PT30M ». Le plus court intervalle pris en charge est de 5 minutes. La planification est facultative : en cas d'omission, un indexeur ne s'exécute qu'une seule fois lorsqu'il est créé. Toutefois, vous pouvez à tout moment exécuter un indexeur à la demande.

Pour plus d'informations sur l'API Créer un indexeur, consultez [Créer un indexeur](#).

Pour plus d'informations sur la définition des planifications de l'indexeur, consultez [Guide pratique pour planifier les indexeurs pour Recherche cognitive Azure](#).

## Utiliser .NET

Le kit de développement logiciel (SDK) .NET mis à la disposition générale offre une parité complète avec l'API REST mise à la disposition générale. Nous vous recommandons de consulter la section précédente de l'API REST pour découvrir les concepts, les workflows et les exigences. Vous pouvez alors vous référer à la documentation de référence des API .NET suivante pour implémenter un indexeur JSON dans du code managé.

- [microsoft.azure.search.models.datasource](#)
- [microsoft.azure.search.models.datasourcetype](#)
- [microsoft.azure.search.models.index](#)
- [microsoft.azure.search.models.indexer](#)

## Indexation des documents modifiés

L'objectif d'une stratégie de détection des changements de données est d'identifier efficacement les données modifiées. La seule stratégie actuellement prise en charge est la [HighWaterMarkChangeDetectionPolicy](#) qui utilise la propriété `_ts` (timestamp) fournie par Azure Cosmos DB, définie ainsi :

```
{  
    "@odata.type" : "#Microsoft.Azure.Search.HighWaterMarkChangeDetectionPolicy",  
    "highWaterMarkColumnName" : "_ts"  
}
```

Cette stratégie est vivement recommandée pour garantir de bonnes performances pour l'indexeur.

Si vous utilisez une requête personnalisée, assurez-vous que la propriété `_ts` est projetée par la requête.

### Progression incrémentielle et requêtes personnalisées

Dans le cas où l'exécution de l'indexeur est interrompue par des défaillances passagères ou un dépassement du délai d'exécution, la progression incrémentielle pendant l'indexation permet à l'indexeur de reprendre là où il en était lors de sa dernière exécution, plutôt que d'avoir à tout réindexer depuis le début. Ceci est particulièrement important lors de l'indexation de grandes collections.

Pour activer la progression incrémentielle lors de l'utilisation d'une requête personnalisée, assurez-vous que votre requête classe les résultats par la colonne `_ts`. Ceci permet de créer des points de contrôle périodiques dont Azure Search se sert pour proposer la progression incrémentielle en cas d'erreurs.

Dans certains cas, il se peut qu'Azure Search ne déduise pas que la requête est ordonnée par `_ts`, même si elle contient une clause `ORDER BY [collection alias]._ts`. Vous pouvez indiquer à Azure Search que les résultats sont triés à l'aide de la propriété de configuration `assumeOrderByHighWaterMarkColumn`. Pour ce faire, créez ou mettez à jour l'indexeur comme suit :

```
{  
    ... other indexer definition properties  
    "parameters" : {  
        "configuration" : { "assumeOrderByHighWaterMarkColumn" : true } }  
}
```

## Indexation des documents supprimés

Lorsque des lignes sont supprimées de la collection, vous devez normalement supprimer ces lignes de l'index de recherche. L'objectif d'une stratégie de détection des suppressions de données est d'identifier efficacement les données supprimées. La seule stratégie actuellement prise en charge est la stratégie `Soft Delete` (où la suppression est signalée par un indicateur quelconque), spécifiée comme suit :

```
{  
    "@odata.type" : "#Microsoft.Azure.Search.SoftDeleteColumnDeletionDetectionPolicy",  
    "softDeleteColumnName" : "the property that specifies whether a document was deleted",  
    "softDeleteMarkerValue" : "the value that identifies a document as deleted"  
}
```

Si vous utilisez une requête personnalisée, assurez-vous que la propriété référencée par `softDeleteColumnName` est projetée par la requête.

L'exemple suivant crée une source de données avec des conseils pour une stratégie de suppression en douceur :

```
POST https://[service name].search.windows.net/datasources?api-version=2020-06-30
Content-Type: application/json
api-key: [Search service admin key]

{
    "name": "mycosmosdbdatasource",
    "type": "cosmosdb",
    "credentials": {
        "connectionString":
"AccountEndpoint=https://myCosmosDbEndpoint.documents.azure.com;AccountKey=myCosmosDbAuthKey;Database=myCosmosDbDatabaseId"
    },
    "container": { "name": "myCosmosDbCollectionId" },
    "dataChangeDetectionPolicy": {
        "@odata.type": "#Microsoft.Azure.Search.HighWaterMarkChangeDetectionPolicy",
        "highWaterMarkColumnName": "_ts"
    },
    "dataDeletionDetectionPolicy": {
        "@odata.type": "#Microsoft.Azure.Search.SoftDeleteColumnDeletionDetectionPolicy",
        "softDeleteColumnName": "isDeleted",
        "softDeleteMarkerValue": "true"
    }
}
```

## Étapes suivantes

Félicitations ! Vous avez appris à intégrer Azure Cosmos DB avec Recherche cognitive Azure à l'aide d'un indexeur.

- Pour en savoir plus sur Azure Cosmos DB, consultez la [page du service Azure Cosmos DB](#).
- Pour en savoir plus sur la Recherche cognitive Azure, consultez la [page du service Recherche](#).

# Indexation de documents dans Azure Data Lake Storage Gen2

04/10/2020 • 5 minutes to read • [Edit Online](#)

## IMPORTANT

La prise en charge d'Azure Data Lake Storage Gen2 est actuellement en préversion publique. Les fonctionnalités en préversion sont fournies sans contrat de niveau de service et ne sont pas recommandées pour les charges de travail de production. Pour plus d'informations, consultez [Conditions d'Utilisation Supplémentaires relatives aux Évaluations Microsoft Azure](#). Vous pouvez demander l'accès aux préversions en remplissant [ce formulaire](#). Les [versions d'API REST 2020-06-30-preview](#) et le portail offrent cette fonctionnalité. Le SDK .NET n'est actuellement pas pris en charge.

Lorsque vous configurez un compte de stockage Azure, vous avez la possibilité d'activer un [espace de noms hiérarchique](#). Cela permet d'organiser la collection du contenu d'un compte dans une hiérarchie de répertoires et de sous-répertoires imbriqués. En activant l'espace de noms hiérarchique, vous activez [Azure Data Lake Storage Gen2](#).

Cet article explique comment prendre en main l'indexation des documents qui se trouvent dans Azure Data Lake Storage Gen2.

## Configurer l'indexeur Azure Data Lake Storage Gen2

Vous devez effectuer quelques étapes pour indexer le contenu de Data Lake Storage Gen2.

### Étape 1 : S'inscrire à la version préliminaire

Inscrivez-vous à la préversion de l'indexeur Data Lake Storage Gen2 en remplissant [ce formulaire](#). Vous recevrez un e-mail de confirmation une fois que vous avez été accepté dans la préversion.

### Étape 2 : Suivre les étapes de configuration de l'indexation du stockage d'objets blob Azure

Une fois que vous avez reçu la confirmation de la réussite de votre inscription à la préversion, vous êtes prêt à créer le pipeline d'indexation.

Vous pouvez indexer le contenu et les métadonnées de Data Lake Storage Gen2 à l'aide de l'[API REST version 2020-06-30-preview](#) ou le portail. Il n'y a pas de prise en charge de .NET SDK pour l'instant.

L'indexation du contenu dans Data Lake Storage Gen2 est identique à l'indexation du contenu dans le stockage d'objets Blob Azure. Pour savoir comment configurer la source de données, l'index et l'indexeur Data Lake Storage Gen2, consultez [Comment indexer des documents dans Stockage Blob Azure avec la Recherche cognitive Azure](#). L'article relatif au stockage d'objets Blob fournit également des informations sur les formats de document pris en charge, les propriétés de métadonnées d'objet Blob extraites, l'indexation incrémentielle, et bien d'autres encore. Ces informations seront identiques pour Data Lake Storage Gen2.

## Contrôle d'accès

Azure Data Lake Storage Gen2 implémente un [modèle de contrôle d'accès](#) qui prend en charge le contrôle d'accès en fonction du rôle (Azure RBAC) et les listes de contrôle d'accès (ACL) POSIX. Lorsque vous indexez du contenu à partir de Data Lake Storage Gen2, Recherche cognitive Azure n'extraira pas les informations RBAC et ACL du contenu. Par conséquent, ces informations ne seront pas incluses dans votre index Recherche cognitive Azure.

Si la gestion du contrôle d'accès sur chaque document de l'index est importante, le développeur de l'application

doit implémenter un [filtrage de sécurité](#).

## Détection des changements

L'indexeur Data Lake Storage Gen2 prend en charge la détection des modifications. Cela signifie que, lorsque l'indexeur s'exécute, il réindexe uniquement les objets blob modifiés comme déterminé par l'horodateur `LastModified` de l'objet blob.

### NOTE

Data Lake Storage Gen2 permet de renommer les répertoires. Quand un répertoire est renommé, les horodateurs des objets blob qu'il contient ne sont pas mis à jour. Par conséquent, l'indexeur ne réindexe pas ces objets blob. Si vous avez besoin que les objets blob d'un répertoire soient réindexés après le changement de nom de celui-ci parce que leurs URL ont changé, vous devez mettre à jour l'horodateur `LastModified` pour tous les objets blob dans l'annuaire afin que l'indexeur sache les réindexer lors d'une exécution ultérieure.

# Se connecter à du contenu Azure SQL et l'indexer à l'aide d'un indexeur Recherche cognitive Azure

04/10/2020 • 28 minutes to read • [Edit Online](#)

Avant d'interroger un [index Recherche cognitive Azure](#), vous devez le remplir avec vos données. Si les données se trouvent dans Azure SQL Database ou SQL Managed Instance, un **indexeur Recherche cognitive Azure pour Azure SQL Database** (ou **indexeur Azure SQL**) peut automatiser le processus d'indexation. En d'autres termes, vous avez moins de code à écrire et la maintenance de l'infrastructure est moins lourde.

Cet article décrit l'utilisation des [indexeurs](#), mais aussi les fonctionnalités propres à Azure SQL Database ou SQL Managed Instance (par exemple, le suivi intégré des modifications).

En plus d'Azure SQL Database et de SQL Managed Instance, Recherche cognitive Azure fournit des indexeurs pour [Azure Cosmos DB](#), le [stockage blob Azure](#) et le [stockage de table Azure](#). Pour obtenir de l'aide concernant d'autres sources de données, indiquez vos souhaits sur le [forum Recherche cognitive Azure](#).

## Indexeurs et sources de données

Une **source de données** spécifie les données à indexer, les informations d'identification pour accéder aux données, et les stratégies qui identifient efficacement les modifications apportées aux données (comme les lignes nouvelles, modifiées ou supprimées). Elle est définie en tant que ressource indépendante utilisable par plusieurs indexeurs.

Un **indexeur** est une ressource qui connecte une source de données unique à un index de recherche cible. Un indexeur est utilisé pour :

- effectuer une copie unique des données pour remplir un index ;
- Synchroniser un index avec les modifications apportées à la source de données selon une planification donnée.
- S'exécuter à la demande afin de mettre à jour un index en fonction des besoins.

Un indexeur unique peut utiliser une seule table ou une seule vue, mais vous pouvez créer plusieurs indexeurs pour remplir plusieurs index de recherche. Pour plus d'informations sur ces concepts, consultez [Opérations d'indexeur : workflow classique](#).

Vous pouvez installer et configurer un indexeur SQL Azure avec les outils suivants :

- Assistant Importation de données sur le [Portail Azure](#)
- [Kit de développement logiciel \(SDK\) .NET](#) de Recherche cognitive Azure
- [API REST](#) de Recherche cognitive Azure

Dans cet article, nous allons utiliser l'API REST pour créer des **indexeurs** et des **sources de données**.

## Quand utiliser l'indexeur Azure SQL

Selon plusieurs facteurs relatifs à vos données, l'utilisation de l'indexeur Azure SQL peut être ou ne pas être appropriée. Si vos données répondent aux conditions suivantes, vous pouvez utiliser l'indexeur Azure SQL.

CRITÈRES	DÉTAILS
Les données proviennent d'une seule table ou d'une seule vue	Si les données sont disséminées entre plusieurs tables, vous pouvez créer une vue unique des données. Toutefois, si vous utilisez une vue, vous ne pourrez plus utiliser la fonction intégrée de détection des modifications de SQL Server pour actualiser un index avec des modifications incrémentielles. Pour plus d'informations, consultez la section <a href="#">Capture des lignes modifiées et supprimées</a> ci-dessous.
Les types de données sont compatibles	Mais certains types SQL ne sont pas pris en charge dans les index Recherche cognitive Azure. Pour obtenir une liste, consultez <a href="#">Mappage des types de données</a> .
La synchronisation de données en temps réel n'est pas requise	Un indexeur peut réindexer votre table toutes les cinq minutes au maximum. Si vos données changent fréquemment et si les modifications doivent être intégrées dans l'index en quelques secondes ou quelques minutes, nous vous recommandons d'utiliser l' <a href="#">API REST</a> ou le <a href="#">SDK .NET</a> pour émettre directement les lignes mises à jour.
Une indexation incrémentielle est possible	Si vous avez un jeu de données important et si vous comptez exécuter l'indexeur selon une planification, Recherche cognitive Azure doit être en mesure d'identifier efficacement les lignes nouvelles, modifiées ou supprimées. L'indexation non incrémentielle n'est autorisée que si vous effectuez une indexation à la demande (non planifiée) ou une indexation de moins de 100 000 lignes. Pour plus d'informations, consultez la section <a href="#">Capture des lignes modifiées et supprimées</a> ci-dessous.

#### NOTE

Recherche cognitive Azure ne prend en charge que l'authentification SQL Server. Si vous avez besoin de prise en charge pour l'authentification du mot de passe Azure Active Directory, veuillez voter pour cette [suggestion UserVoice](#).

## Créer un indexeur Azure SQL

### 1. Créez la source de données :

```
POST https://myservice.search.windows.net/datasources?api-version=2020-06-30
Content-Type: application/json
api-key: admin-key

{
    "name" : "myazuresqldatasource",
    "type" : "azuresql",
    "credentials" : { "connectionString" : "Server=tcp:<your
server>.database.windows.net,1433;Database=<your database>;User ID=<your user name>;Password=<your
password>;Trusted_Connection=False;Encrypt=True;Connection Timeout=30;" },
    "container" : { "name" : "name of the table or view that you want to index" }
}
```

Vous pouvez obtenir la chaîne de connexion auprès du [portail Azure](#). Utilisez l'option `ADO.NET connection string`.

### 2. Créez l'index Recherche cognitive Azure cible si vous n'en avez pas encore. Pour créer un index, utilisez

le [portail](#) ou l'[API Créer un index](#). Vérifiez que le schéma de votre index cible est compatible avec le schéma de la table source. Consultez [Mappage entre les types de données SQL et les types de données de Recherche cognitive Azure](#).

- Créez l'indexeur en lui attribuant un nom et en référençant les sources de données sources et cibles :

```
POST https://myservice.search.windows.net/indexers?api-version=2020-06-30
Content-Type: application/json
api-key: admin-key

{
    "name" : "myindexer",
    "dataSourceName" : "myazuresqldatasource",
    "targetIndexName" : "target index name"
}
```

Un indexeur créé de cette façon n'a pas de planification. Il s'exécute automatiquement une fois créé. Vous pouvez le réexécuter à tout moment à l'aide d'une requête **run indexer** :

```
POST https://myservice.search.windows.net/indexers/myindexer/run?api-version=2020-06-30
api-key: admin-key
```

Vous pouvez personnaliser différents aspects du comportement des indexeurs, notamment la taille du lot et le nombre de documents pouvant être ignorés avant que l'exécution d'un indexeur n'échoue. Pour plus d'informations, consultez [Créer une API d'indexeur](#).

Il se peut que vous deviez autoriser des services Azure pour vous connecter à votre base de données. Pour plus d'informations sur la marche à suivre, consultez la section [Connexion à partir de Azure](#).

Pour surveiller l'état et l'historique d'exécution de l'indexeur (nombre d'éléments indexés, échecs, etc.), utilisez une requête **indexer status** :

```
GET https://myservice.search.windows.net/indexers/myindexer/status?api-version=2020-06-30
api-key: admin-key
```

La réponse doit être semblable à ce qui suit :

```
{
    "@odata.context": "https://myservice.search.windows.net/$metadata#Microsoft.Azure.Search.V2015_02_28.IndexerExecutionInfo",
    "status": "running",
    "lastResult": {
        "status": "success",
        "errorMessage": null,
        "startTime": "2015-02-21T00:23:24.957Z",
        "endTime": "2015-02-21T00:36:47.752Z",
        "errors": [],
        "itemsProcessed": 1599501,
        "itemsFailed": 0,
        "initialTrackingState": null,
        "finalTrackingState": null
    },
    "executionHistory": [
        {
            "status": "success",
            "errorMessage": null,
            "startTime": "2015-02-21T00:23:24.957Z",
            "endTime": "2015-02-21T00:36:47.752Z",
            "errors": [],
            "itemsProcessed": 1599501,
            "itemsFailed": 0,
            "initialTrackingState": null,
            "finalTrackingState": null
        },
        ... earlier history items
    ]
}
```

L'historique d'exécution contient jusqu'à 50 exécutions les plus récentes, classées par ordre antichronologique (la dernière exécution apparaît en premier dans la réponse). Vous trouverez des informations supplémentaires sur la réponse dans [Obtenir l'état de l'indexeur](#)

## Exécuter des indexeurs selon une planification

Vous pouvez également configurer l'indexeur pour qu'il s'exécute à intervalles périodiques. Pour ce faire, ajoutez la propriété **schedule** lors de la création ou de la mise à jour de l'indexeur. L'exemple ci-dessous montre une requête PUT mettant à jour l'indexeur :

```
PUT https://myservice.search.windows.net/indexers/myindexer?api-version=2020-06-30
Content-Type: application/json
api-key: admin-key

{
    "dataSourceName" : "myazuresqldatasource",
    "targetIndexName" : "target index name",
    "schedule" : { "interval" : "PT10M", "startTime" : "2015-01-01T00:00:00Z" }
}
```

Le paramètre **interval** est obligatoire. Il correspond à la durée entre le début de deux exécutions consécutives de l'indexeur. L'intervalle minimal autorisé est de 5 minutes, l'intervalle maximal autorisé est d'une journée. Il doit être formaté en tant que valeur « `dayTimeDuration` » XSD (un sous-ensemble limité d'une valeur de [durée ISO 8601](#)). Le modèle est le suivant : `P(nD)(T(nH))(nM)`. Exemples : `PT15M` toutes les 15 minutes, `PT2H` toutes les deux heures.

Pour plus d'informations sur la définition des planifications de l'indexeur, consultez [Comment planifier des indexeurs pour la Recherche cognitive Azure](#).

# Capturer des lignes nouvelles, modifiées et supprimées

La Recherche cognitive Azure utilise l'**indexation incrémentielle** pour éviter d'avoir à réindexer toute la table ou toute la vue à chaque exécution d'un indexeur. Recherche cognitive Azure fournit deux stratégies de détection des modifications pour la prise en charge de l'indexation incrémentielle.

## Stratégie de suivi intégré des modifications SQL

Si votre base de données SQL prend en charge le [suivi des modifications](#), nous recommandons d'utiliser la **stratégie de suivi intégré des modifications SQL**. Il s'agit de la stratégie la plus efficace. De plus, elle permet à la Recherche cognitive Azure d'identifier les lignes supprimées, sans avoir à ajouter une colonne « suppression réversible » explicite à votre table.

### Spécifications

- Configuration requise pour la version de base de données :
  - SQL Server 2012 SP3 et versions ultérieures, si vous utilisez SQL Server sur des machines virtuelles Azure.
  - Azure SQL Database ou SQL Managed Instance.
- Tables uniquement (aucune vue).
- Dans la base de données, [activez le suivi](#) de la table.
- Aucune clé primaire composite (clé primaire contenant plusieurs colonnes) dans la table.

### Usage

Pour utiliser cette stratégie, créez ou mettez à jour votre source de données comme suit :

```
{  
    "name" : "myazuresqldatasource",  
    "type" : "azuresql",  
    "credentials" : { "connectionString" : "connection string" },  
    "container" : { "name" : "table or view name" },  
    "dataChangeDetectionPolicy" : {  
        "@odata.type" : "#Microsoft.Azure.Search.SqlIntegratedChangeTrackingPolicy"  
    }  
}
```

Si vous utilisez le suivi intégré des modifications SQL, ne spécifiez pas une stratégie de détection des lignes supprimées. Elle intègre la prise en charge de l'identification des lignes supprimées. Toutefois, pour les suppressions détectées automatiquement, la clé de document de votre index de recherche doit être identique à la clé primaire de la table SQL.

### NOTE

Lorsque vous utilisez [TRUNCATE TABLE](#) pour supprimer un grand nombre de lignes dans une table SQL, l'indexeur doit être [reset](#) pour réinitialiser l'état de suivi des modifications et récupérer les suppressions de lignes.

## Stratégie de détection des modifications de limite supérieure

Cette stratégie de détection des modifications s'appuie sur une colonne « Limite supérieure » qui capture la version ou l'heure pour laquelle une ligne a été mise à jour. Si vous utilisez une vue, vous devez vous servir d'une stratégie de limite supérieure. La colonne de limite supérieure doit remplir les conditions suivantes.

### Spécifications

- Toutes les insertions spécifient une valeur pour la colonne.
- Toutes les mises à jour d'un élément modifient également la valeur de la colonne.
- La valeur de cette colonne augmente à chaque insertion ou mise à jour.
- Les requêtes utilisant les clauses WHERE et ORDER BY suivantes peuvent être exécutées efficacement :

```
WHERE [High Water Mark Column] > [Current High Water Mark Value] ORDER BY [High Water Mark Column]
```

## IMPORTANT

Nous vous recommandons d'utiliser le type de données `rowversion` pour la colonne dédiée à la limite supérieure. Si un autre type de données est utilisé, il n'est pas garanti que le suivi des modifications capture toutes les modifications en présence de transactions qui s'exécutent en même temps qu'une requête de l'indexeur. Lorsque vous utilisez `rowversion` dans une configuration avec des répliques en lecture seule, vous devez pointer l'indexeur sur le réplica principal. Seul un réplica principal peut être utilisé dans les scénarios de synchronisation de données.

## Usage

Pour utiliser une stratégie de limite supérieure, créez ou mettez à jour votre source de données comme suit :

```
{
    "name" : "myazuresqldatasource",
    "type" : "azuresql",
    "credentials" : { "connectionString" : "connection string" },
    "container" : { "name" : "table or view name" },
    "dataChangeDetectionPolicy" : {
        "@odata.type" : "#Microsoft.Azure.Search.HighWaterMarkChangeDetectionPolicy",
        "highWaterMarkColumnName" : "[a rowversion or last_updated column name]"
    }
}
```

## WARNING

Si la table source n'a pas d'index dans la colonne de limite supérieure, les requêtes utilisées par l'indexeur SQL risquent d'expirer. En particulier, la clause `ORDER BY [High Water Mark Column]` a besoin d'un index pour pouvoir s'exécuter efficacement lorsque la table contient de nombreuses lignes.

### convertHighWaterMarkToRowVersion

Si vous utilisez un type de données `rowversion` pour la colonne de limite supérieure, envisagez d'utiliser le paramètre de configuration de l'indexeur `convertHighWaterMarkToRowVersion`.

`convertHighWaterMarkToRowVersion` effectue deux opérations :

- Utilisez le type de données `rowversion` pour la colonne de limite supérieure dans la requête sql de l'indexeur. L'utilisation du type de données correct améliore le niveau de performance de requête de l'indexeur.
- Soustrayez 1 de la valeur `rowversion` avant l'exécution de la requête de l'indexeur. Les affichages comportant de 1 à plusieurs jointures peuvent contenir des lignes avec des valeurs `rowversion` en double. Soustraire 1 garantit que la requête de l'indexeur n'ignore pas ces lignes.

Pour activer cette fonctionnalité, créez ou mettez à jour l'indexeur avec la configuration suivante :

```
{
    ...
    ... other indexer definition properties
    "parameters" : {
        "configuration" : { "convertHighWaterMarkToRowVersion" : true } }
}
```

### queryTimeout

Si vous rencontrez des erreurs de temporisation, vous pouvez utiliser le paramètre de configuration d'indexeur `queryTimeout` pour donner une valeur plus élevée que les 5 minutes par défaut au délai d'expiration de la requête. Par exemple, pour fixer un délai d'expiration de 10 minutes, créez ou mettez à jour l'indexeur avec la configuration suivante :

```
{
    ...
    ... other indexer definition properties
    "parameters" : {
        "configuration" : { "queryTimeout" : "00:10:00" } }
}
```

#### `disableOrderByHighWaterMarkColumn`

Vous pouvez également désactiver la clause `ORDER BY [High Water Mark Column]`. Toutefois, cette action est déconseillée car, si l'exécution de l'indexeur est interrompue par une erreur, l'indexeur doit traiter à nouveau toutes les lignes quand son exécution reprend, même s'il avait déjà traité la quasi-totalité des lignes au moment de l'interruption. Pour désactiver la clause `ORDER BY`, utilisez le paramètre

`disableOrderByHighWaterMarkColumn` dans la définition de l'indexeur :

```
{
    ...
    ... other indexer definition properties
    "parameters" : {
        "configuration" : { "disableOrderByHighWaterMarkColumn" : true } }
}
```

## Stratégie de détection des colonnes à suppression réversible

Lorsque des lignes sont supprimées de la table source, vous devez également supprimer ces lignes de l'index de recherche. Si vous utilisez la stratégie de suivi intégré des modifications SQL, cette opération est prise en charge à votre place. Mais la stratégie de suivi des modifications de limite supérieure ne vous est d'aucune aide pour les lignes supprimées. Que faire, alors ?

Si des lignes sont physiquement supprimées de la table, Recherche cognitive Azure n'a aucun moyen de déduire la présence d'enregistrements qui n'existent plus. Toutefois, vous pouvez utiliser la technique de la « suppression réversible » pour supprimer des lignes logiquement sans les supprimer de la table. Ajoutez une colonne à votre table ou votre vue et marquez les lignes comme supprimées à l'aide de cette colonne.

Lorsque vous utilisez la technique de suppression réversible, vous pouvez spécifier cette stratégie réversible comme suit lors de la création ou de la mise à jour de la source de données :

```
{
    ...
    ...
    "dataDeletionDetectionPolicy" : {
        "@odata.type" : "#Microsoft.Azure.Search.SoftDeleteColumnDeletionDetectionPolicy",
        "softDeleteColumnName" : "[a column name]",
        "softDeleteMarkerValue" : "[the value that indicates that a row is deleted]"
    }
}
```

**softDeleteMarkerValue** doit être une chaîne. Utilisez la représentation au format chaîne de votre valeur. Par exemple, si vous avez une colonne d'entiers dans laquelle les lignes supprimées sont marquées avec la valeur 1, utilisez `"1"`. Si vous avez une colonne BIT dans laquelle les lignes supprimées sont marquées avec la valeur booléenne True, utilisez le littéral de chaîne `True` ou `true`, la casse ne comptant pas.

## Mappage entre les types de données SQL et Recherche cognitive Azure

TYPE DE DONNÉES SQL	TYPES DE CHAMPS D'INDEX CIBLE AUTORISÉS	NOTES
bit	Edm.Boolean, Edm.String	

TYPE DE DONNÉES SQL	TYPES DE CHAMPS D'INDEX CIBLE AUTORISÉS	NOTES
int, smallint, tinyint	Edm.Int32, Edm.Int64, Edm.String	
bigint	Edm.Int64, Edm.String	
real, float	Edm.Double, Edm.String	
smallmoney, money decimal numeric	Edm.String	Recherche cognitive Azure ne prend pas en charge la conversion de types décimaux en Edm.Double, car elle entraîne une perte de précision
char, nchar, varchar, nvarchar	Edm.String Collection(Edm.String)	Une chaîne SQL peut être utilisée pour remplir un champ Collection(Edm.String) si la chaîne représente un tableau JSON de chaînes : ["red", "white", "blue"]
smalldatetime, datetime, datetime2, date, datetimeoffset	Edm.DateTimeOffset, Edm.String	
uniqueidentifier	Edm.String	
Geography	Edm.GeographyPoint	Seules les instances Geography de type POINT avec SRID 4326 (valeur par défaut) sont prises en charge
rowversion	N/A	Les colonnes de version de ligne ne peuvent pas être stockées dans l'index de recherche, mais peuvent être utilisées pour le suivi des modifications
time, timespan, binary, varbinary, image, xml, geometry, types CLR	N/A	Non pris en charge

## Paramètres de configuration

L'indexeur SQL expose plusieurs paramètres de configuration :

PARAMÈTRE	TYPE DE DONNÉES	OBJECTIF	VALEUR PAR DÉFAUT
queryTimeout	string	Définit le délai d'expiration de l'exécution de la requête SQL	5 minutes ("00:05:00")
disableOrderByHighWaterMarkColumn	bool	Indique que la requête SQL utilisée par la stratégie de limite supérieure doit omettre la clause ORDER BY. Consultez <a href="#">Stratégie de limite supérieure</a>	false

Ces paramètres sont utilisés dans l'objet `parameters.configuration`, dans la définition de l'indexeur. Par

exemple, pour fixer un délai d'expiration de la requête de 10 minutes, créez ou mettez à jour l'indexeur avec la configuration suivante :

```
{  
    ... other indexer definition properties  
    "parameters" : {  
        "configuration" : { "queryTimeout" : "00:10:00" } }  
}
```

## Questions fréquentes (FAQ)

**Q : Puis-je utiliser l'indexeur SQL Azure avec des bases de données SQL exécutées sur des machines virtuelles IaaS dans Azure ?**

Oui. Toutefois, vous devez autoriser votre service de recherche à se connecter à votre base de données. Pour plus d'informations, consultez l'article [Configurer une connexion d'un indexeur de Recherche cognitive Azure à SQL Server sur une machine virtuelle Azure](#).

**Q : Puis-je utiliser l'indexeur Azure SQL avec des bases de données SQL exécutées localement ?**

Pas directement. La connexion directe n'est pas prise en charge, ni recommandée, car elle vous oblige à ouvrir vos bases de données au trafic Internet. Les clients ont réussi à l'aide de technologies de pont telles qu'Azure Data Factory. Pour plus d'informations, consultez [Envoyer des données à un index Recherche cognitive Azure à l'aide d'Azure Data Factory](#).

**Q : Puis-je utiliser l'indexeur Azure SQL avec des bases de données autres que SQL Server exécutées en IaaS sur Azure ?**

Non. Ce cas de figure n'est pas pris en charge, car nous n'avons pas testé l'indexeur avec des bases de données autres que SQL Server.

**Q : Puis-je créer plusieurs indexeurs qui s'exécutent selon une planification ?**

Oui. Cependant, seul un indexeur peut s'exécuter sur un nœud à la fois. Si vous avez besoin d'exécuter plusieurs indexeurs simultanément, envisagez d'ajouter d'autres unités de recherche à votre service de recherche.

**Q : L'exécution d'un indexeur affecte-t-elle la charge de travail de mes requêtes ?**

Oui. L'indexeur s'exécute sur un des nœuds de votre service de recherche, et les ressources de ce nœud sont partagées entre l'indexation et le traitement du trafic de requêtes d'une part, et d'autres requêtes d'API d'autre part. Si vous exécutez des charges de travail intensives d'indexation et de requête et si vous rencontrez un taux élevé d'erreurs 503 ou une augmentation des délais de réponse, [redimensionnez votre service de recherche](#).

**Q : Puis-je utiliser un réplica secondaire dans un cluster de basculement comme source de données ?**

Cela dépend. Pour l'indexation intégrale d'une table ou d'une vue, vous pouvez utiliser un réplica secondaire.

Pour l'indexation incrémentielle, Recherche cognitive Azure prend en charge deux stratégies de détection des modifications : le suivi des modifications intégré SQL et la limite supérieure.

Sur les réplicas en lecture seule, SQL Database ne prend pas en charge le suivi des modifications intégré. Par conséquent, vous devez utiliser la stratégie de limite supérieure.

Nous vous recommandons d'utiliser le type de données rowversion pour la colonne dédiée à la limite supérieure. Toutefois, l'utilisation de rowversion repose sur la fonction `MIN_ACTIVE_ROWVERSION`, qui n'est pas prise en charge sur les réplicas en lecture seule. Par conséquent, vous devez pointer l'indexeur sur un réplica

principal si vous utilisez rowversion.

Si vous essayez d'utiliser rowversion sur un réplica en lecture seule, l'erreur suivante s'affiche :

« L'utilisation d'une colonne rowversion pour le suivi des modifications n'est pas prise en charge sur les réplicas de disponibilité secondaires (en lecture seule). Veuillez mettre à jour la source de données et spécifier une connexion au réplica de disponibilité principal. La propriété de « capacité de mise à jour » de la base de données actuelle est « READ\_ONLY ».

**Q : Puis-je utiliser une colonne autre que rowversion pour le suivi des modifications de la limite supérieure ?**

Cela n'est pas recommandé. Seule la colonne **rowversion** permet une synchronisation fiable des données. Toutefois, en fonction de votre logique d'application, cette opération peut être sécurisée si :

- Vous pouvez vous assurer que pendant l'exécution de l'indexeur, aucune transaction n'est en attente sur la table en cours d'indexation (par exemple, toutes les mises à jour de la table s'effectuent de manière planifiée par lot, et la planification de l'indexeur Recherche cognitive Azure est définie de manière à éviter tout chevauchement avec la planification de la mise à jour de la table).
- Vous procédez régulièrement à une réindexation complète pour sélectionner toutes les lignes manquantes.

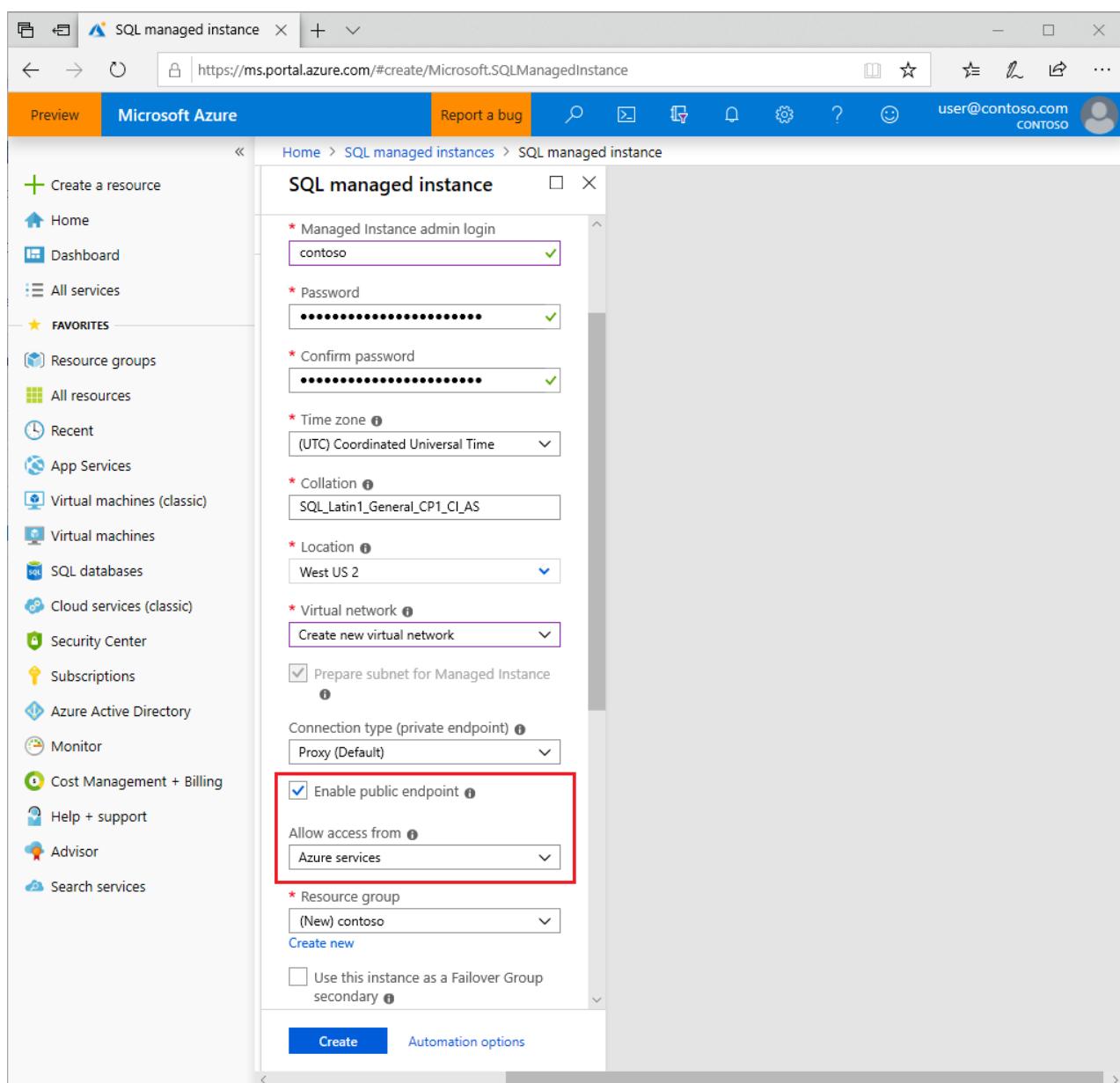
# Configurer une connexion entre un indexeur Recherche cognitive Azure et SQL Managed Instance

04/10/2020 • 3 minutes to read • [Edit Online](#)

Comme indiqué dans [Connexion d'Azure SQL Database à Recherche cognitive Azure à l'aide d'indexeurs](#), la création d'indexeurs sur **SQL Managed Instances** est prise en charge par Recherche cognitive Azure via le point de terminaison public.

## Créer une instance Azure SQL Managed Instance avec un point de terminaison public

Créez une instance SQL Managed Instance avec l'option **Activer le point de terminaison public** sélectionnée.



The screenshot shows the Microsoft Azure portal interface for creating a new SQL Managed Instance. The left sidebar lists various service options like Home, Dashboard, and Resource groups. The main central area is titled 'SQL managed instance' and contains several configuration fields:

- Managed Instance admin login: contoso
- Password and Confirm password: both fields contain masked text.
- Time zone: (UTC) Coordinated Universal Time
- Collation: SQL\_Latin1\_General\_CI\_AS
- Location: West US 2
- Virtual network: Create new virtual network
- A checked checkbox labeled 'Prepare subnet for Managed Instance'
- A dropdown for 'Connection type (private endpoint)': Proxy (Default)
- A checked checkbox labeled 'Enable public endpoint' (which is highlighted with a red box).
- An 'Allow access from' dropdown set to 'Azure services' (also highlighted with a red box).
- A dropdown for 'Resource group': (New) contoso
- A unchecked checkbox for 'Use this instance as a Failover Group secondary'

At the bottom are two buttons: 'Create' (in blue) and 'Automation options'.

## Activer un point de terminaison public Azure SQL Managed Instance

Vous pouvez également activer un point de terminaison public sur une instance SQL Managed Instance existante

sous Sécurité > Réseau virtuel > Point de terminaison public > Activer.

The screenshot shows the Microsoft Azure portal interface. The left sidebar contains a list of services and resources. The main content area is titled "contoso - Virtual network" and shows the "Public endpoint (data)" settings. A red box highlights the "Enable" button. Below it, a note states: "This option requires port 3342 to be open for inbound traffic. You will need to configure NSG rule for this port separately." The "Host" field is set to "contoso.public.000000000000.database.windows.net" and the "Port" is set to "3342". The "Connection type (private endpoint)" dropdown is set to "Proxy (Default)".

## Vérifier les règles du groupe de sécurité réseau

Vérifiez que le groupe de sécurité réseau contient des **règles de sécurité de trafic entrant** appropriées qui autorisent les connexions à partir des services Azure.

The screenshot shows the Microsoft Azure portal interface. The left sidebar contains a list of services and resources. The main content area is titled "nsg-contoso - Inbound security rules" and displays a table of rules. A red box highlights the row for "public\_endpoint\_inbound" with priority 1300, which allows TCP port 3342 from AzureCloud to Any destination. Other rows include "allow\_management\_inbound", "allow\_missubnet\_inbound", "allow\_health\_probe\_inbound", "allow\_tds\_inbound", "allow\_redirect\_inbound", "allow\_geodr\_inbound", and several deny rules.

PRIORITY	NAME	PORT	PROTOCOL	SOURCE	DESTINATION	ACTION
100	allow_management_inbound	9000,9003,14... Any	TCP	Any	Any	Allow
200	allow_missubnet_inbound	Any	Any	10.0.0/16	Any	Allow
300	allow_health_probe_inbound	Any	Any	AzureLoadBa...	Any	Allow
1000	allow_tds_inbound	1433	TCP	VirtualNetwork	Any	Allow
1100	allow_redirect_inbound	11000-11999	TCP	VirtualNetwork	Any	Allow
1200	allow_geodr_inbound	5022	TCP	VirtualNetwork	Any	Allow
1300	public_endpoint_inbound	3342	TCP	AzureCloud	Any	Allow
4096	deny_all_inbound	Any	Any	Any	Any	Deny
65000	AllowVnetInBound	Any	Any	VirtualNetwork	VirtualNetwork	Allow
65001	AllowAzureLoadBalanceri...	Any	Any	AzureLoadBa...	Any	Allow
65500	DenyAllInBound	Any	Any	Any	Any	Deny

## NOTE

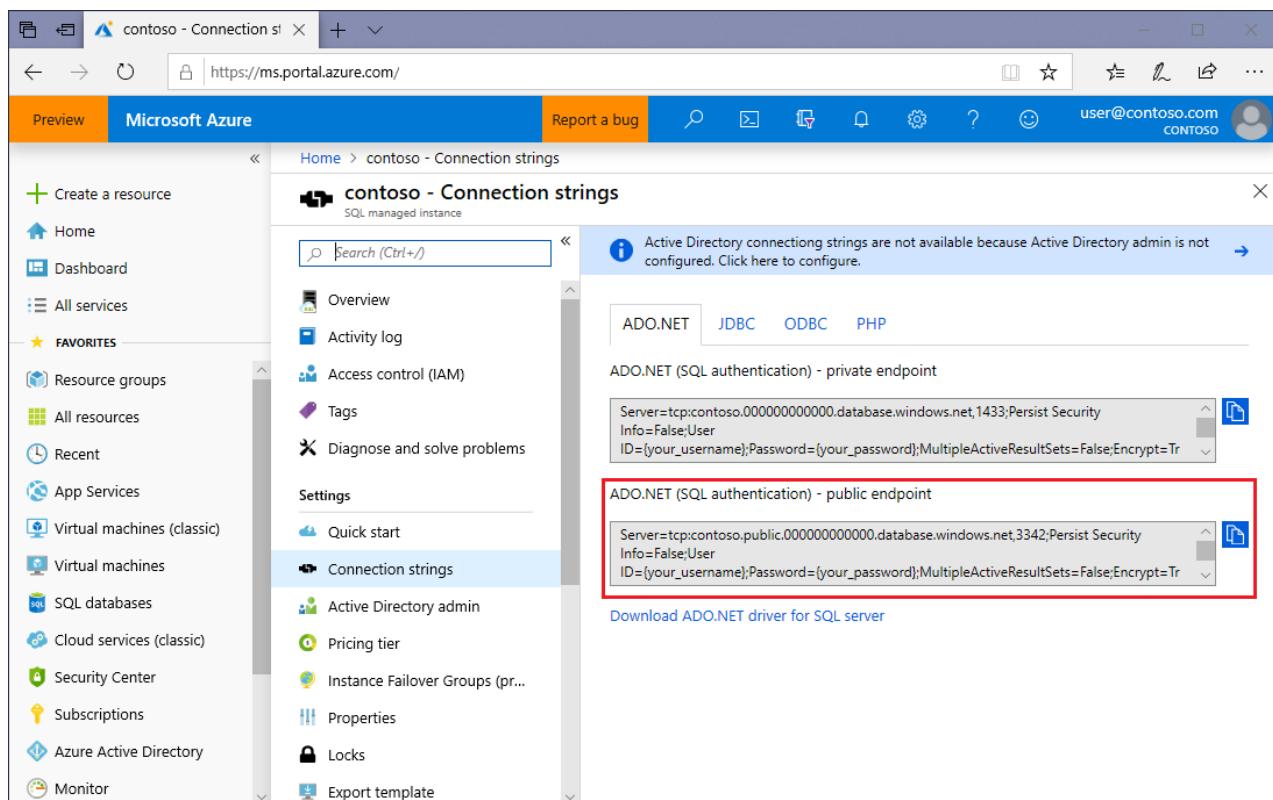
Les indexeurs requièrent toujours que SQL Managed Instance soit configuré avec un point de terminaison public pour pouvoir lire les données. Toutefois, vous pouvez choisir de restreindre l'accès entrant de ce point de terminaison public en remplaçant la règle actuelle (`public_endpoint_inbound`) par les deux règles suivantes :

- Autorisation de l'accès entrant à partir de la [balise de service](#) `AzureCognitiveSearch` ("SOURCE" = `AzureCognitiveSearch`, "NAME" = `cognitive_search_inbound`)
- Autorisation de l'accès entrant à partir de l'adresse IP du service de recherche, qui peut être obtenue en exécutant une commande ping sur son nom de domaine complet (par exemple, `<your-search-service-name>.search.windows.net`). ("SOURCE" = `IP address`, "NAME" = `search_service_inbound`)

Pour chacune de ces deux règles, définissez "PORT" = `3342`, "PROTOCOL" = `TCP`, "DESTINATION" = `Any`, "ACTION" = `Allow`

## Obtenir la chaîne de connexion du point de terminaison public

Veuillez à utiliser la chaîne de connexion pour le [point de terminaison public](#) (le port 3342, et non le port 1433).



## Étapes suivantes

Une fois la configuration résolue, vous pouvez spécifier une instance SQL Managed Instance comme source de données pour un indexeur Recherche cognitive Azure à l'aide du portail ou de l'API REST. Pour plus d'informations, consultez [Connexion d'Azure SQL Database à Recherche cognitive Azure à l'aide d'indexeurs](#).

# Configurer une connexion d'un indexeur de Recherche cognitive Azure à SQL Server sur une machine virtuelle Azure

04/10/2020 • 12 minutes to read • [Edit Online](#)

Comme indiqué dans [Connexion d'Azure SQL Database à Recherche cognitive Azure à l'aide d'indexeurs](#), la création d'indexeurs dans **SQL Server sur des machines virtuelles Azure** (ou des **machines virtuelles Azure SQL** pour faire plus court) est prise en charge par le service Recherche cognitive Azure, mais il existe quelques conditions préalables liées à la sécurité qu'il faut résoudre en premier.

Les connexions entre la Recherche cognitive Azure et SQL Server sur une machine virtuelle constituent des connexions Internet publiques. Toutes les mesures de sécurité que vous suivriez normalement pour ces connexions s'appliquent également ici :

- Obtenez un certificat auprès d'un [fournisseur d'autorité de certification](#) pour le nom de domaine complet de l'instance SQL Server de la machine virtuelle Azure.
- Installez le certificat sur la machine virtuelle, puis activez et configurez les connexions chiffrées sur la machine virtuelle en suivant les instructions de cet article.

## Activer des connexions chiffrées

Le service Recherche cognitive Azure requiert un canal chiffré pour toutes les demandes d'indexeur via une connexion internet publique. Cette section répertorie les étapes pour y parvenir.

1. Vérifiez les propriétés du certificat pour vous assurer que le nom du sujet est le nom de domaine complet (FQDN) de la machine virtuelle Azure. Vous pouvez utiliser un outil tel que CertUtils ou le composant logiciel enfichable Certificats pour afficher les propriétés. Vous pouvez obtenir le nom de domaine complet à partir de la section Essentials du panneau du service de la machine virtuelle dans le champ **Adresse IP publique/Étiquette du nom DNS** dans le [portail Azure](#).

- Pour les machines virtuelles créées à l'aide du modèle **Resource Manager** le plus récent, le nom de domaine complet est au format `<your-VM-name>.<region>.cloudapp.azure.com`.
- Pour les machines virtuelles plus anciennes créées comme machines virtuelles **Classic**, le nom de domaine complet est au format `<your-cloud-service-name.cloudapp.net>`.

2. Configurez SQL Server de manière à utiliser le certificat à l'aide de l'Éditeur du Registre (regedit).

Bien que le Gestionnaire de Configuration SQL Server soit souvent utilisé pour cette tâche, vous ne pouvez pas l'utiliser pour ce scénario. Il ne trouve pas le certificat importé, car le nom de domaine complet de la machine virtuelle sur Azure ne correspond pas au nom de domaine complet tel que déterminé par la machine virtuelle (il identifie le domaine en tant qu'ordinateur local ou domaine du réseau auquel il est joint). Lorsque les noms ne correspondent pas, utilisez regedit pour spécifier le certificat.

- Dans regedit, accédez à la clé de Registre suivante :  
`HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Microsoft SQL Server\[MSSQL13.MSSQLSERVER]\MSSQLServer\SuperSocketNetLib\Certificate`

La partie `[MSSQL13.MSSQLSERVER]` varie en fonction du nom de version et de l'instance.

- Définissez la valeur de la clé de **Certificat** sur l'**empreinte numérique** du certificat TLS/SSL que

vous avez importé sur la machine virtuelle.

Il existe plusieurs manières d'obtenir l'empreinte numérique, certaines étant mieux que d'autres. Si vous la copiez à partir du composant logiciel enfichable **certificats** dans MMC, vous allez probablement choisir un caractère de début invisible [comme décrit dans cet article du support technique](#), ce qui entraîne une erreur lors de la tentative de connexion. Il existe plusieurs solutions de contournement pour résoudre ce problème. La plus simple consiste à reculer et retaper le premier caractère de l'empreinte numérique pour supprimer le caractère de début dans le champ de la valeur de clé dans regedit. Vous pouvez également utiliser un autre outil pour copier l'empreinte numérique.

### 3. Accordez des autorisations pour le compte de service.

Assurez-vous que le compte de service SQL Server a l'autorisation appropriée sur la clé privée du certificat TLS/SSL. Si vous ignorez cette étape, SQL Server ne démarre pas. Vous pouvez utiliser le composant logiciel enfichable **Certificats** ou **CertUtils** pour cette tâche.

### 4. Redémarrez le service SQL Server.

## Configurer la connectivité SQL Server dans la machine virtuelle

Après avoir configuré la connexion chiffrée requise par la Recherche cognitive Azure, vous devez réaliser des étapes de configuration supplémentaires intrinsèques à SQL Server sur les machines virtuelles Azure. Si vous ne l'avez pas déjà fait, l'étape suivante consiste à terminer la configuration en utilisant l'un de ces articles :

- Pour une machine virtuelle **Resource Manager**, consultez l'article [Connect to a SQL Server Virtual Machine on Azure using Resource Manager](#)(Se connecter à une machine virtuelle SQL Server sur Azure à l'aide de Resource Manager).
- Pour une machine virtuelle **classique**, consultez [Connexion à une machine virtuelle sur Azure Classic](#).

En particulier, examinez la section de chaque article pour la « connexion via internet ».

## Configurer le groupe de sécurité réseau

Il n'est pas inhabituel de configurer le groupe de sécurité réseau et le point de terminaison correspondant ou la liste de contrôle d'accès Azure correspondant(e) pour rendre votre machine virtuelle Azure accessible à d'autres parties. Vous avez sans doute déjà effectué cela pour permettre à votre propre logique d'application de se connecter à votre machine virtuelle SQL Azure. C'est la même chose pour une connexion de la Recherche cognitive Azure à votre machine virtuelle SQL Azure.

Les liens ci-dessous fournissent des instructions sur la configuration du groupe de sécurité réseau pour les déploiements de machines virtuelles. Utilisez ces instructions pour faire figurer un point de terminaison Recherche cognitive Azure dans la liste de contrôle d'accès en fonction de son adresse IP.

### NOTE

Pour obtenir des informations générales, consultez [Présentation du groupe de sécurité réseau](#)

- Pour une machine virtuelle **Resource Manager**, consultez [How to create NSGs for ARM deployments](#)(Procédure de création des groupes de sécurité réseau pour les déploiements ARM).
- Pour une machine virtuelle **classique**, consultez [How to create NSGs for Classic deployments](#)(Procédure de création des groupes de sécurité réseau pour les déploiements classiques).

L'adressage IP peut poser quelques problèmes qui sont facilement surmontés si vous êtes conscient du problème et des solutions de contournement possibles. Les sections restantes fournissent des recommandations pour la

gestion des problèmes liés aux adresses IP dans la liste de contrôle d'accès.

#### **Restreindre l'accès à Recherche cognitive Azure**

Nous vous recommandons fortement de restreindre l'accès à l'adresse IP de votre service de recherche et la plage d'adresses IP de l'[étiquette de service](#) `AzureCognitiveSearch` dans la liste de contrôle d'accès au lieu de rendre vos machines virtuelles SQL Azure disponibles pour toutes les demandes de connexion.

Vous pouvez trouver l'adresse IP en effectuer un test ping sur le nom de domaine complet (par exemple, `<your-search-service-name>.search.windows.net`) de votre service de recherche.

Vous pouvez trouver la plage d'adresses IP de l'[étiquette de service](#) `AzureCognitiveSearch` en utilisant des [fichiers JSON téléchargeables](#) ou via l'[API de détection d'étiquettes de service](#). La plage d'adresses IP est mise à jour chaque semaine.

#### **Gestion des fluctuations d'adresse IP**

Si votre service de recherche n'a qu'une seule unité de recherche (autrement dit, un réplica et une partition), l'adresse IP change lors d'un redémarrage du service de routine, ce qui invalide une liste de contrôle d'accès existante avec votre adresse IP du service de recherche.

Pour éviter l'erreur de connectivité qui s'ensuit, utilisez plusieurs réplicas et une partition dans la Recherche cognitive Azure. Cela augmente le coût, mais permet également de résoudre le problème d'adresse IP. Dans la Recherche cognitive Azure, les adresses IP ne changent pas lorsque vous avez plusieurs unités de recherche.

Une deuxième approche consiste à autoriser l'échec de la connexion et de reconfigurer les listes de contrôle d'accès dans le groupe de sécurité réseau. En moyenne, vous pouvez vous attendre à ce que les adresses IP soient modifiées à quelques semaines d'intervalle. Pour les clients qui procèdent à une indexation contrôlée de manière irrégulière, cette approche peut être viable.

Une troisième approche viable (mais pas particulièrement sécurisée) consiste à spécifier la plage d'adresses IP de la région Azure où votre service de recherche est configuré. La liste des plages d'adresses IP à partir desquelles les adresses IP publiques sont allouées à des ressources Azure est publiée dans [Plages d'adresses IP du centre de données Azure](#).

#### **Inclure les adresses IP du portail de la Recherche cognitive Azure**

Si vous utilisez le Portail Azure pour créer un indexeur, la logique du portail de la Recherche cognitive Azure doit également pouvoir accéder à votre machine virtuelle SQL Azure lors de la création. Exécutez la commande ping sur `stamp2.search.ext.azure.com` pour trouver les adresses IP du portail de la Recherche cognitive Azure.

## **Étapes suivantes**

Une fois la configuration résolue, vous pouvez maintenant spécifier un serveur SQL Server sur une machine virtuelle Azure comme source de données pour un indexeur de la Recherche cognitive Azure. Pour plus d'informations, consultez [Connexion d'Azure SQL Database à Recherche cognitive Azure à l'aide d'indexeurs](#).

# Attacher une ressource Cognitive Services à un ensemble de compétences dans Recherche cognitive Azure

04/10/2020 • 12 minutes to read • [Edit Online](#)

Lors de la configuration d'un pipeline d'enrichissement dans la Recherche cognitive Azure, vous pouvez enrichir un nombre limité de documents gratuitement. Vous pouvez également associer une ressource Cognitive Services facturable pour des charges de travail plus volumineuses et plus fréquentes.

Dans cet article, vous allez apprendre à joindre une ressource en affectant une clé à un ensemble de compétences qui définit un pipeline d'enrichissement.

## Ressources utilisées pendant l'enrichissement

La Recherche cognitive Azure a une dépendance vis-à-vis de Cognitive Services, y compris [Vision par ordinateur](#), pour l'analyse d'images et la reconnaissance optique de caractères (OCR), [Analyse de texte](#) pour le traitement en langage naturel et d'autres enrichissements tels que la [Traduction de texte](#). Dans le contexte de l'enrichissement dans la Recherche cognitive Azure, ces algorithmes d'IA sont encapsulés dans une *qualification*, placés dans un *ensemble de qualifications* et référencés par un *indexeur* lors de l'indexation.

## Comment la facturation fonctionne

- La Recherche cognitive Azure utilise la clé de ressource Cognitive Services que vous fournissez sur un ensemble de compétences pour facturer l'enrichissement des images et du texte. L'exécution de qualifications facturables est facturée au [tarif de paiement à l'utilisation de Cognitive Services](#).
- L'extraction d'images est une opération de la Recherche cognitive Azure qui se produit lorsque les documents sont décodés avant l'enrichissement. L'extraction d'images est facturable. Pour connaître les prix appliqués à l'extraction d'images, voir la [page de tarification du service Recherche cognitive Azure](#).
- L'extraction de texte se produit également lors de la phrase de décodage de document. Elle n'est pas facturable.
- Les qualifications qui n'appellent pas Cognitive Services, y compris les compétences conditionnelles, le modélisateur, la fusion de texte et le fractionnement du texte, ne sont pas facturables.

## Exigence de même région

Recherche cognitive Azure et Azure Cognitive Services doivent obligatoirement exister au sein de la même région. Autrement, vous obtenez ce message lors de l'exécution :

`"Provided key is not a valid CognitiveServices type key for the region of your search service."`

Il n'existe aucun moyen de changer un service de région. Si vous obtenez cette erreur, vous devez créer une ressource Cognitive Services située dans la même région que le service Recherche cognitive Azure.

#### NOTE

Certaines compétences intégrées sont basées sur des services cognitifs non régionaux (par exemple la [compétence de traduction de texte](#)). L'utilisation d'une compétence non régionale signifie que votre requête peut être desservie dans une région autre que la région de la Recherche cognitive Azure. Pour plus d'informations sur les services non régionaux, consultez la page [Produit Cognitive Services par région](#).

## Utiliser des ressources gratuites

Vous pouvez utiliser une option de traitement gratuite, limitée aux exercices des guides de démarrage rapide et des tutoriels d'enrichissement par IA.

Les ressources du niveau tarifaire Gratuit (enrichissements limités) sont limitées à 20 documents par jour, par indexeur. Vous pouvez supprimer et recréer l'indexeur pour réinitialiser le compteur.

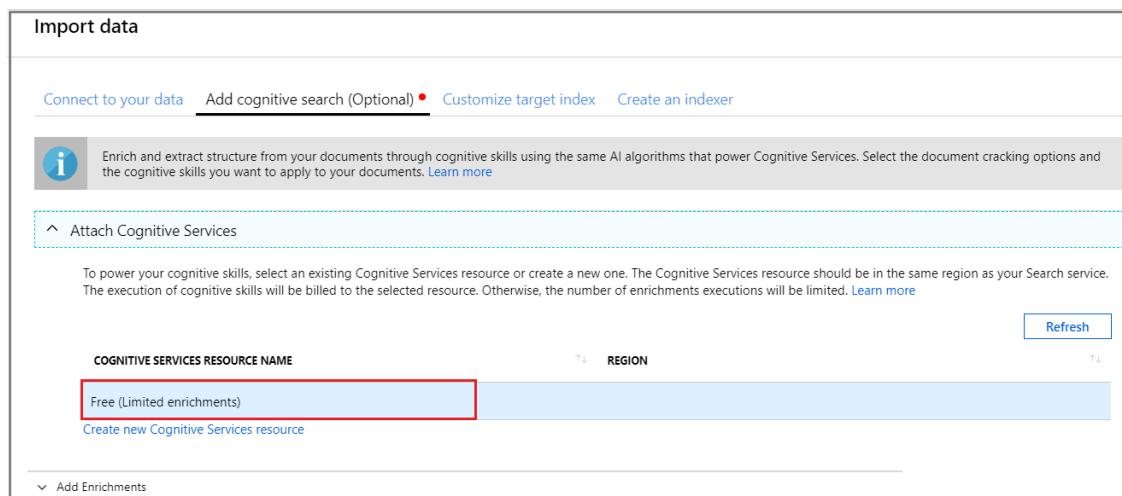
1. Ouvrez l'Assistant Importation de données :



2. Choisissez une source de données, puis passez à Ajouter l'enrichissement par IA (facultatif) .

Pour suivre une procédure pas à pas de cet Assistant, consultez [Créer un index dans le Portail Azure](#).

3. Développez Attacher Cognitive Services, puis sélectionnez Gratuit (enrichissements limités) .



4. Vous pouvez maintenant passer aux étapes suivantes, notamment Ajouter les compétences cognitives.

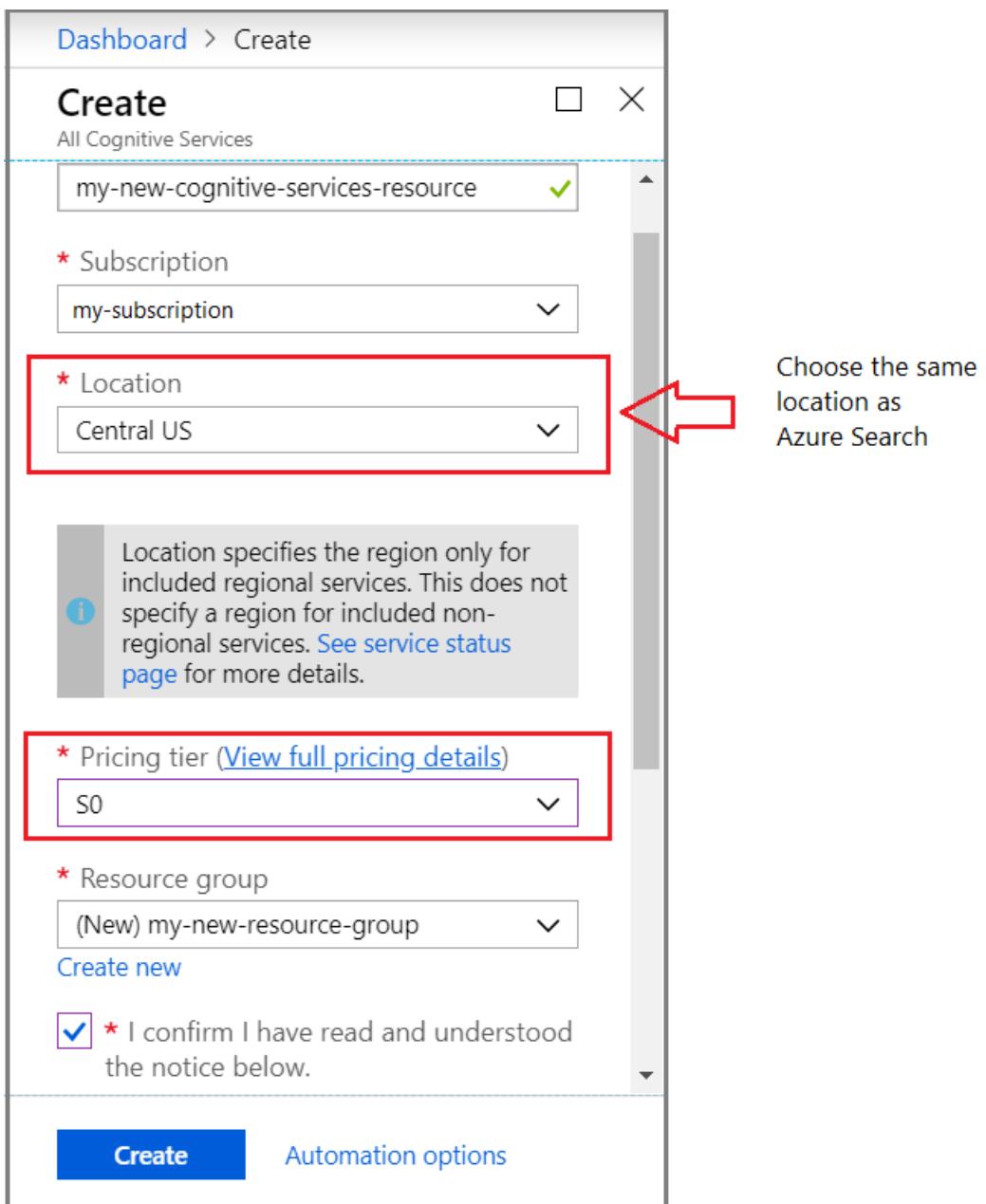
## Utiliser des ressources facturables

Pour les charges de travail créant plus de 20 enrichissements par jour, veillez à attacher une ressource Cognitive Services facturable. Nous vous recommandons de toujours attacher une ressource Cognitive Services facturable, même si vous n'avez aucune intention d'appeler les API Cognitive Services. L'attachement d'une ressource modifie la limite quotidienne.

Vous n'êtes facturé que pour les qualifications qui appellent les API Cognitive Services. Vous n'êtes pas facturé pour les [qualifications personnalisées](#) ou des qualifications telles que [Fusion de texte](#), [Séparateur de texte](#) et [Modélisateur](#) qui ne sont pas basées sur des API.

1. Ouvrez l'Assistant Importation de données, choisissez une source de données, puis passez à Ajouter l'enrichissement par IA (facultatif) .

- Développez **Attacher Cognitive Services**, puis sélectionnez **Créer une ressource Cognitive Services**. Un nouvel onglet s'ouvre pour vous permettre de créer la ressource :



- Dans la liste **Emplacement**, sélectionnez la région où se trouve votre service Recherche cognitive Azure. Veillez à utiliser cette région à des fins de performances. L'utilisation de cette région évite également les frais de bande passante sortante entre les différentes régions.
  - Dans la liste **Niveau tarifaire**, sélectionnez **S0** pour obtenir la collection complète de fonctionnalités Cognitive Services, y compris les fonctionnalités Vision et Langue qui sous-tendent les compétences prédéfinies fournies par Recherche cognitive Azure.
- Pour le niveau S0, vous pouvez trouver les tarifs des charges de travail spécifiques dans la [page de tarification de Cognitive Services](#).
- Dans la liste **Sélectionner une offre**, assurez-vous que **Cognitive Services** est sélectionné.
  - Sous les fonctionnalités **Langue**, les tarifs d'Analyse de texte standard s'appliquent à l'indexation basée sur l'intelligence artificielle.
  - Sous les fonctionnalités **Vision**, les tarifs de **Vision par ordinateur S1** s'appliquent.
- Sélectionnez **Créer** pour approvisionner la nouvelle ressource Cognitive Services.
  - Revenez à l'onglet précédent contenant l'Assistant Importation de données. Sélectionnez **Actualiser**

pour afficher la ressource Cognitive Services, puis sélectionnez-la :

^ Attach Cognitive Services

To power your cognitive skills, select an existing Cognitive Services resource or create a new one. The Cognitive Services resource should be in the same region as your Search service.  
The execution of cognitive skills will be billed to the selected resource. Otherwise, the number of enrichments executions will be limited. [Learn more](#)

Refresh

COGNITIVE SERVICES RESOURCE NAME	REGION
Free (Limited enrichments)	
my-cog-services-resource	westus2

[Create new Cognitive Services resource](#)

7. Développez la section **Ajouter des compétences cognitives** pour sélectionner les qualifications cognitives spécifiques à exécuter sur vos données. Suivez les recommandations restantes de l'Assistant.

## Attacher un ensemble de qualifications à une ressource Cognitive Services existante

Si vous avez un ensemble de qualifications existant, vous pouvez l'attacher à une ressource Cognitive Services nouvelle ou différente.

1. Dans la page **Vue d'ensemble du service**, sélectionnez l'onglet **Ensembles de qualifications** :

Usage	Monitoring	Indexes (2)	Indexers (2)	Data sources (2)	Skillsets
NAME					
myskillset				1	...

2. Sélectionnez le nom de l'ensemble de qualifications, puis choisissez une ressource existante ou créez-en une. Cliquez sur **OK** pour confirmer vos modifications.

**myskillset**

Skills

Delete skillset

To power your cognitive skills, select an existing Cognitive Services resource or create a new one. The Cognitive Services resource should be in the same region as your Search service. The execution of cognitive skills will be billed to the selected resource. Otherwise, the number of enrichments executions will be limited. [Learn more](#)

[Refresh](#)

COGNITIVE SERVICES RESOURCE NAME	REGION
Free (Limited enrichments)	
my-cog-services-resource	westus2

[Create new Cognitive Services resource](#)

N'oubliez pas que l'option **Gratuit (enrichissements limités)** vous limite à 20 documents par jour et que vous pouvez utiliser l'option **Créer une ressource Cognitive Services** pour approvisionner une nouvelle ressource facturable. Si vous avez créé une ressource, sélectionnez **Actualiser** pour rafraîchir la liste des ressources Cognitive Services, puis sélectionnez la ressource.

## Attacher Cognitive Services par programme

Lorsque vous définissez l'ensemble de qualifications par programme, ajoutez une section `cognitiveServices` à l'ensemble de qualifications. Dans cette section, incluez la clé de la ressource Cognitive Services que vous souhaitez associer à l'ensemble de qualifications. Rappelez-vous que la ressource doit se trouver dans la même région que votre ressource Recherche cognitive Azure. Incluez également `@odata.type`, que vous définissez sur `#Microsoft.Azure.Search.CognitiveServicesByKey`.

L'exemple ci-après illustre ce modèle. Notez la section `cognitiveServices` à la fin de la définition.

```
PUT https://[servicename].search.windows.net/skillsets/[skillset name]?api-version=2020-06-30
api-key: [admin key]
Content-Type: application/json
```

```
{
  "name": "skillset name",
  "skills": [
    [
      {
        "@odata.type": "#Microsoft.Skills.Text.EntityRecognitionSkill",
        "categories": [ "Organization" ],
        "defaultLanguageCode": "en",
        "inputs": [
          {
            "name": "text", "source": "/document/content"
          }
        ],
        "outputs": [
          {
            "name": "organizations", "targetName": "organizations"
          }
        ]
      }
    ],
    "cognitiveServices": {
      "@odata.type": "#Microsoft.Azure.Search.CognitiveServicesByKey",
      "description": "mycogsvcs",
      "key": "<your key goes here>"
    }
  }
}
```

## Exemple : Estimer les coûts

Pour estimer les coûts associés à l'indexation de recherche cognitive, commencez par vous représenter à quoi ressemble un document moyen afin de pouvoir donner une approximation. Par exemple, vous pourriez faire une approximation :

- 1 000 fichiers PDF.
- Six pages par fichier.
- Une image par page (6 000 images).
- 3 000 caractères par page.

Supposons un pipeline consistant en un craquage de document PDF, une extraction d'image et de texte, une reconnaissance optique de caractères (OCR) d'images, et une reconnaissance d'entités d'organisations.

Les prix indiqués dans cet article sont hypothétiques. Ils sont utilisés pour illustrer le processus d'estimation. Vos coûts peuvent être inférieurs. Pour obtenir les prix réels des transactions, voir [Tarification Cognitive Services](#).

1. Pour le décodage de documents avec contenu de texte et d'image, l'extraction de texte est actuellement gratuite. Pour 6 000 images, supposez un prix de 1 USD pour chaque tranche de 1 000 images extraites. Cela revient à un coût de 6,00 USD pour cette étape.
2. Pour la reconnaissance optique de caractères (OCR) de 6 000 images en anglais, la qualification cognitive OCR utilise le meilleur algorithme (DescribeText). À raison de 2,50 \$ par lot de 1 000 images à analyser, cette étape vous coûterait 15 \$.
3. Pour l'extraction d'entité, vous auriez un total de trois enregistrements texte par page. Chaque enregistrement contient 1 000 caractères. Trois enregistrements texte par page multipliés par 6 000 pages équivalent à 18 000 enregistrements texte. À raison de 2 \$ par lot de 1 000 enregistrements texte, cette étape vous coûterait 36 \$.

En additionnant tout cela, l'ingestion de 1 000 documents PDF de ce type avec l'ensemble de qualifications

décrit vous reviendrait à environ 57 USD.

## Étapes suivantes

- [Page de tarification de Recherche cognitive Azure](#)
- [Guide pratique pour définir un ensemble de compétences](#)
- [Créer un jeu de compétences \(REST\)](#)
- [Guide pratique pour mapper des champs enrichis](#)

# Comment créer un ensemble de compétences dans un pipeline d'enrichissement de l'IA dans la Recherche cognitive Azure

04/10/2020 • 16 minutes to read • [Edit Online](#)



Un ensemble de compétences définit les opérations qui extraient et enrichissent des données afin qu'elles puissent faire l'objet de recherches. Un ensemble de compétences est exécuté une fois que le contenu du texte et de l'image est extrait des documents sources et que les champs du document source sont (éventuellement) mappés aux champs de destination d'un index ou d'une base de connaissances.

Un ensemble de compétences contient une ou plusieurs *compétences cognitives* qui représentent une opération d'enrichissement spécifique, par exemple la traduction d'un texte, l'extraction d'expressions clés ou l'exécution d'une reconnaissance optique de caractères à partir d'un fichier image. Pour créer un ensemble de compétences, vous pouvez utiliser les [compétences intégrées](#) de Microsoft ou des compétences personnalisées contenant des modèles ou une logique de traitement que vous fournissez (consultez [Exemple : Création d'une compétence personnalisée dans un pipeline d'enrichissement de l'intelligence artificielle](#) pour plus d'informations).

Dans cet article, vous allez découvrir comment créer un pipeline d'enrichissement pour les compétences que vous souhaitez utiliser. Un ensemble de compétences est attaché à un [indexeur](#) de Recherche cognitive Azure. Une partie de la conception du pipeline, traitée dans cet article, constitue le jeu de compétences proprement dit.

## NOTE

Une autre partie de la conception du pipeline spécifie un indexeur, décrit dans [l'étape suivante](#). Une définition d'indexeur inclut une référence au jeu de compétences, ainsi que les mappages de champs utilisés pour la connexion des entrées aux sorties dans l'index cible.

Points importants à retenir :

- Vous ne pouvez avoir qu'un jeu de compétences par indexeur.
- Un ensemble de compétences doit avoir au moins une compétence.
- Vous pouvez créer plusieurs compétences du même type (par exemple, les variantes d'une compétence d'analyse d'image).

## Commencer en ayant la fin à l'esprit

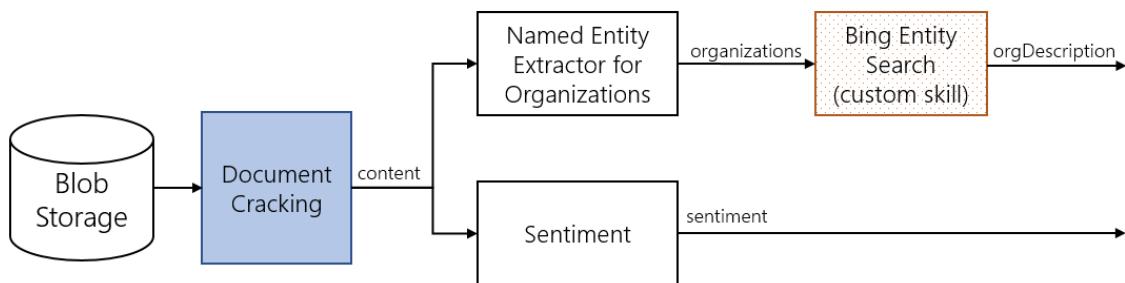
Nous vous recommandons de choisir d'abord les données à extraire de vos données brutes et le mode d'utilisation souhaité pour ces données dans une solution de recherche. Il peut être utile de créer une illustration du pipeline d'enrichissement complet pour vous aider à identifier les étapes nécessaires.

Par exemple, vous souhaitez traiter un ensemble de commentaires d'analystes financiers. Pour chaque fichier, vous souhaitez extraire les noms de sociétés et la tendance générale des commentaires. Vous

souhaiterez peut-être également écrire un enrichisseur personnalisé qui utilise le service Recherche d'entités Bing pour rechercher des informations supplémentaires sur la société, comme le type d'activité de l'entreprise. Sur le fond, vous souhaitez extraire des informations telles que celles qui suivent, qui sont indexées pour chaque document :

DOSSIER-TEXTE	SOCIÉTÉS	TENDANCE	DESCRIPTION DE SOCIÉTÉS
exemple de dossier	[« Microsoft », « LinkedIn »]	0,99	[« Microsoft Corporation est une multinationale informatique américaine... », « LinkedIn est un réseau social professionnel... »]

Le diagramme suivant illustre un pipeline d'enrichissement hypothétique :



Une fois que vous avez une idée assez claire de ce que vous souhaitez inclure dans le pipeline, vous pouvez représenter le jeu de compétences qui correspond à ces étapes. Au niveau fonctionnel, le jeu de compétences est représenté lorsque vous chargez la définition de l'indexeur dans la Recherche cognitive Azure. Pour en savoir plus sur le chargement de l'indexeur, consultez la [documentation relative à l'indexeur](#).

Dans le diagramme, l'étape de *décodage de document* a lieu automatiquement. La Recherche cognitive Azure sait comment ouvrir des fichiers connus, et crée un champ de *contenu* dont le texte est extrait à partir de chaque document. Les cases blanches sont des enrichisseurs intégrés, et la case « Recherche d'entités Bing » en pointillé représente un enrichisseur personnalisé que vous créez. Comme illustré sur le diagramme, le jeu de compétences contient trois compétences.

## Définition du jeu de compétences dans REST

Un jeu de compétences est défini comme un tableau de compétences. Chaque compétence définit la source de ses entrées et le nom des sorties générées. À l'aide de l'[API REST Create Skillset](#), vous pouvez définir un jeu de compétences qui correspond au diagramme précédent :

```

PUT https://[servicename].search.windows.net/skillsets/[skillset name]?api-version=2020-06-30
api-key: [admin key]
Content-Type: application/json
  
```

```
{
  "description": "Extract sentiment from financial records, extract company names, and then find additional information about each company mentioned.",
  "skills": [
    {
      "@odata.type": "#Microsoft.Skills.Text.EntityRecognitionSkill",
      "context": "/document",
      "categories": [ "Organization" ],
      "defaultLanguageCode": "en",
      "inputs": [
        {
          "name": "text",
          "source": "/document/content"
        }
      ],
      "outputs": [
        {
          "name": "organizations",
          "targetName": "organizations"
        }
      ]
    },
    {
      "@odata.type": "#Microsoft.Skills.Text.SentimentSkill",
      "inputs": [
        {
          "name": "text",
          "source": "/document/content"
        }
      ],
      "outputs": [
        {
          "name": "score",
          "targetName": "mySentiment"
        }
      ]
    },
    {
      "@odata.type": "#Microsoft.Skills.Custom.WebApiSkill",
      "description": "Calls an Azure function, which in turn calls Bing Entity Search",
      "uri": "https://indexer-e2e-webskill.azurewebsites.net/api/InvokeTextAnalyticsV3?code=foo",
      "httpHeaders": {
        "Ocp-Apim-Subscription-Key": "foobar"
      },
      "context": "/document/organizations/*",
      "inputs": [
        {
          "name": "query",
          "source": "/document/organizations/*"
        }
      ],
      "outputs": [
        {
          "name": "description",
          "targetName": "companyDescription"
        }
      ]
    }
  ]
}
```

## Créer un ensemble de compétences

Lorsque vous créez un jeu de compétences, vous pouvez en indiquer une description pour qu'il se documente automatiquement. Une description d'un jeu de compétences est facultative, mais utile pour en conserver une trace. Comme un jeu de compétences est un document JSON, qui n'autorise pas de commentaires, vous devez utiliser un élément `description` à cet effet.

```
{  
  "description":  
    "This is our first skill set, it extracts sentiment from financial records, extract company names,  
    and then finds additional information about each company mentioned.",  
    ...  
}
```

L'élément suivant du jeu de compétences est un tableau de compétences. Vous pouvez penser chaque compétence comme étant une primitive d'enrichissement. Chaque compétence effectue une petite tâche dans ce pipeline d'enrichissement. Chacune d'elles accepte une entrée (ou un ensemble d'entrées) et retourne des sorties. Les sections suivantes sont consacrées à la spécification des compétences prédéfinies et personnalisées, chaînant ainsi les compétences ensemble via des références d'entrée et de sortie. Les entrées peuvent provenir de données sources ou d'une autre compétence. Les sorties peuvent être mappées à un champ dans un index de recherche ou utilisées en tant qu'entrée pour une compétence en aval.

## Ajouter des compétences prédéfinies

Examinons la première compétence, qui est la [compétence de reconnaissance d'entité](#) prédéfinie :

```
{  
  "@odata.type": "#Microsoft.Skills.Text.EntityRecognitionSkill",  
  "context": "/document",  
  "categories": [ "Organization" ],  
  "defaultLanguageCode": "en",  
  "inputs": [  
    {  
      "name": "text",  
      "source": "/document/content"  
    }  
  ],  
  "outputs": [  
    {  
      "name": "organizations",  
      "targetName": "organizations"  
    }  
  ]  
}
```

- Chaque compétence prédéfinie dispose des propriétés `odata.type`, `input` et `output`. Les propriétés propres à une compétence fournissent des informations supplémentaires applicables à cette compétence. Pour la reconnaissance d'entité, `categories` est une entité parmi un ensemble fixe de types d'entité que le modèle préformé peut reconnaître.
- Chaque compétence doit posséder un `"context"`. Le contexte représente le niveau auquel les opérations ont lieu. Dans la compétence ci-dessus, le contexte représente l'ensemble du document, ce qui implique que la compétence de reconnaissance d'entité est appelée une fois par document. Les sorties sont également générées à ce niveau. Plus spécifiquement, les `"organizations"` sont générées en tant que membre de `"/document"`. Dans les compétences en aval, vous pouvez faire référence à ces informations qui viennent d'être créées sous la forme `"/document/organizations"`. Si le champ `"context"` n'est pas défini explicitement, le contexte par défaut est le document.

- La compétence possède une entrée appelée « texte », avec une entrée source définie sur `"/document/content"`. La compétence (reconnaissance d'entité) fonctionne sur le champ *contenu* de chaque document. Il s'agit d'un champ standard créé par l'indexeur des objets blob Azure.
- La compétence possède une sortie appelée `"organizations"`. Les sorties existent uniquement pendant le traitement. Pour chaîner cette sortie à l'entrée d'une compétence en aval, référez la sortie en tant que `"/document/organizations"`.
- Pour un document particulier, la valeur de `"/document/organizations"` est un tableau des organisations extraites du texte. Par exemple :

```
[ "Microsoft", "LinkedIn" ]
```

Certaines situations demandent de référencer chaque élément d'un tableau séparément. Par exemple, vous souhaitez transmettre chaque élément de `"/document/organizations"` séparément à une autre compétence (par exemple, l'enrichisseur personnalisé Recherche d'entités Bing). Vous pouvez faire référence à chaque élément du tableau en ajoutant un astérisque dans le chemin d'accès :

```
"/document/organizations/*"
```

La deuxième compétence correspondant à l'extraction de la tendance suit le même modèle que le premier enrichisseur. Elle dispose de l'entrée `"/document/content"`, et retourne un score de tendance pour chaque instance de contenu. Comme vous n'avez pas défini le champ `"context"` explicitement, la sortie (`mySentiment`) est maintenant un enfant de `"/document"`.

```
{
  "@odata.type": "#Microsoft.Skills.Text.SentimentSkill",
  "inputs": [
    {
      "name": "text",
      "source": "/document/content"
    }
  ],
  "outputs": [
    {
      "name": "score",
      "targetName": "mySentiment"
    }
  ]
},
```

## Ajouter une compétence personnalisée

Rappelez la structure de l'enrichisseur personnalisé Recherche d'entités Bing :

```
{
    "@odata.type": "#Microsoft.Skills.Custom.WebApiSkill",
    "description": "This skill calls an Azure function, which in turn calls Bing Entity Search",
    "uri": "https://indexer-e2e-webskill.azurewebsites.net/api/InvokeTextAnalyticsV3?code=foo",
    "httpHeaders": {
        "Ocp-Apim-Subscription-Key": "foobar"
    },
    "context": "/document/organizations/*",
    "inputs": [
        {
            "name": "query",
            "source": "/document/organizations/*"
        }
    ],
    "outputs": [
        {
            "name": "description",
            "targetName": "companyDescription"
        }
    ]
}
```

Cette définition est une [compétence personnalisée](#) qui appelle une API web dans le cadre du processus d'enrichissement. Pour chaque organisation identifiée par la reconnaissance d'entité, cette compétence appelle une API web pour rechercher la description de cette organisation. L'orchestration du moment auquel appeler l'API web et du traitement des informations reçues est gérée en interne par le moteur d'enrichissement. Toutefois, l'initialisation nécessaire pour appeler cette API personnalisée doit être indiquée dans le fichier JSON (par exemple, l'URI, les en-têtes HTTP et les entrées attendus). Pour obtenir des conseils sur la création d'une API web personnalisée pour le pipeline d'enrichissement, consultez [Guide pratique pour définir une interface personnalisée](#).

Notez que le champ « contexte » contient la valeur `"/document/organizations/*"` avec un astérisque, ce qui signifie que l'étape d'enrichissement est appelée *pour chaque* organisation sous `"/document/organizations"`.

La sortie, dans ce cas une description de société, est générée pour chaque organisation identifiée. Lorsque vous faites référence à la description d'une étape en aval (par exemple, dans l'extraction d'expressions clés), vous utilisez le chemin d'accès `"/document/organizations/*/description"` pour ce faire.

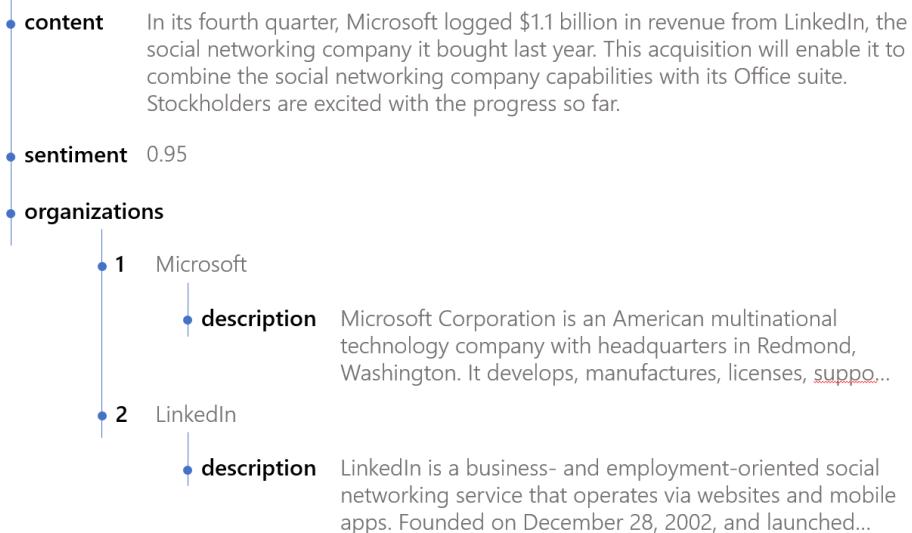
## Ajouter une structure

Le jeu de compétences génère des informations structurées à partir de données non structurées. Prenons l'exemple suivant :

*« In its fourth quarter, Microsoft logged \$1.1 billion in revenue from LinkedIn, the social networking company it bought last year. The acquisition enables Microsoft to combine LinkedIn capabilities with its CRM and Office capabilities. Stockholders are excited with the progress so far. »*

Un résultat probable serait une structure générée semblable à l'illustration suivante :

## document



Jusqu'à présent, cette structure est interne uniquement, en mémoire seulement et utilisée uniquement dans les index de Recherche cognitive Azure. L'ajout d'une base de connaissances vous permet d'enregistrer les enrichissements mis en forme pour une utilisation hors de la recherche.

## Ajouter une base de connaissances

La [Base de connaissances](#) est une fonctionnalité de Recherche cognitive Azure destinée à l'enregistrement de vos documents enrichis. La base de connaissances que vous créez à l'aide d'un compte de stockage Azure est le référentiel dans lequel vos données enrichies résident.

Une définition de la base de connaissances est ajoutée à un ensemble de compétences. Pour obtenir une procédure pas à pas de l'ensemble du processus, consultez la section [Créer une base de connaissances dans REST](#).

```
"knowledgeStore": {  
    "storageConnectionString": "<an Azure storage connection string>",  
    "projections" : [  
        {  
            "tables": [ ]  
        },  
        {  
            "objects": [  
                {  
                    "storageContainer": "containername",  
                    "source": "/document/EnrichedShape/",  
                    "key": "/document/Id"  
                }  
            ]  
        }  
    ]  
}
```

Vous pouvez choisir d'enregistrer les documents enrichis sous forme de tables avec des relations hiérarchiques conservées ou sous forme de documents JSON dans le stockage blob. Toute sortie à partir des compétences de l'ensemble de compétences peut servir de source d'entrée de la projection. Si vous cherchez à projeter les données dans une forme spécifique, la [compétence de modélisateur](#) mise à jour peut maintenant modéliser les types complexes que vous utilisez.

## Étapes suivantes

Maintenant que vous connaissez le pipeline d'enrichissement et les jeux de compétences, poursuivez avec [How to reference annotations in a skillset](#) (Référencement des annotations dans un jeu de compétences) ou [How to map outputs to fields in an index](#) (Mappage de sorties à des champs dans un index).

# Comment référencer des annotations dans un jeu de compétences Recherche cognitive Azure

04/10/2020 • 8 minutes to read • [Edit Online](#)

Cet article vous permet de découvrir comment référencer des annotations dans des définitions de compétences, en utilisant des exemples pour illustrer différents scénarios. À mesure que le contenu d'un document traverse un jeu de compétences, il s'enrichit d'annotations. Ces annotations peuvent être utilisées comme entrées pour un enrichissement supplémentaire en aval, ou mappées à un champ de sortie dans un index.

Les exemples présentés dans cet article sont basés sur le champ de *contenu* généré automatiquement par les [indexeurs d'objets blob Azure](#) durant la phase de décodage du document. Lorsque vous faites référence à des documents à partir d'un conteneur d'objets blob, utilisez un format tel que `"/document/content"`, où le champ de *contenu* fait partie du *document*.

## Concepts de base

Avant d'examiner la syntaxe, revenons sur quelques concepts importants pour mieux comprendre les exemples fournis plus loin.

TERME	DESCRIPTION
Document enrichi	<p>Un document enrichi est une structure interne créée et utilisée par le pipeline pour recueillir toutes les annotations liées à un document. Il faut imaginer un document enrichi comme une arborescence d'annotations. En règle générale, une annotation créée à partir d'une annotation précédente devient l'enfant de celle-ci.</p> <p>Des documents enrichis n'existent que pendant l'exécution d'un jeu de compétences. Une fois le contenu mappé à l'index de recherche, le document enrichi n'est plus nécessaire. Bien que vous n'interagissiez pas directement avec des documents enrichis, il est utile d'avoir un modèle mental des documents lors de la création d'un jeu de compétences.</p>
Contexte d'enrichissement	<p>Contexte dans lequel l'enrichissement a lieu, déterminant l'élément qui est enrichi. Par défaut, le contexte d'enrichissement est au niveau <code>"/document"</code>, limité à des documents individuels. Quand une compétence est appliquée, ses résultats deviennent des <a href="#">propriétés du contexte défini</a>.</p>

## Exemple 1 : référencer une annotation simple

Dans un stockage d'objets blob Azure, supposons que vous ayez une variété de fichiers contenant des références à des noms de personnes que vous souhaitez extraire à l'aide d'une reconnaissance d'entité. Dans la définition de compétence ci-dessous, `"/document/content"` est la représentation textuelle du document entier, et « personnes » une extraction de noms complets d'entités identifiées en tant que personnes.

Le contexte par défaut étant `"/document"`, la liste de personnes peut désormais être référencée comme `"/document/people"`. En l'occurrence, `"/document/people"` est une annotation qui peut maintenant être mappée à

un champ dans un index, ou utilisée dans une autre compétence du même jeu de compétences.

```
{  
    "@odata.type": "#Microsoft.Skills.Text.EntityRecognitionSkill",  
    "categories": [ "Person"],  
    "defaultLanguageCode": "en",  
    "inputs": [  
        {  
            "name": "text",  
            "source": "/document/content"  
        }  
    ],  
    "outputs": [  
        {  
            "name": "persons",  
            "targetName": "people"  
        }  
    ]  
}
```

## Exemple 2 : référencer un tableau à l'intérieur d'un document

Cet exemple s'appuie sur le précédent pour montrer comment appeler une étape d'enrichissement plusieurs fois sur le même document. Supposons que l'exemple précédent a généré un tableau de chaînes comprenant 10 noms de personnes à partir d'un seul document. L'étape suivante pourrait raisonnablement être un deuxième enrichissement extrayant le nom de famille d'un nom complet. Étant donné qu'il y a 10 noms, vous voulez que cette étape soit appelée 10 fois dans ce document, soit une fois par personne.

Pour appeler le nombre approprié d'itérations, définissez le contexte comme `"/document/people/*"`, où l'astérisque (`"*"`) représente tous les nœuds figurant dans le document enrichi en tant que descendants de `"/document/people"`. Bien que cette compétence ne soit définie qu'une seule fois dans le tableau des compétences, elle est appelée pour chaque membre figurant dans le document jusqu'à ce que tous les membres soient traités.

```
{  
    "@odata.type": "#Microsoft.Skills.Custom.WebApiSkill",  
    "description": "Fictitious skill that gets the last name from a full name",  
    "uri": "http://names.azurewebsites.net/api/GetLastName",  
    "context" : "/document/people/*",  
    "defaultLanguageCode": "en",  
    "inputs": [  
        {  
            "name": "fullname",  
            "source": "/document/people/*"  
        }  
    ],  
    "outputs": [  
        {  
            "name": "lastname",  
            "targetName": "last"  
        }  
    ]  
}
```

Lorsque les annotations sont des tableaux ou des collections de chaînes, vous pouvez cibler des membres spécifiques plutôt que le tableau dans son ensemble. L'exemple ci-dessus génère une annotation appelée `"last"` sous chaque nœud représenté par le contexte. Si vous souhaitez faire référence à cette famille d'annotations, vous pouvez utiliser la syntaxe `"/document/people/*/last"`. Si vous voulez faire référence à une annotation particulière, vous pouvez utiliser un index explicite `"/document/people/1/last"` pour faire référence au nom de famille de la

première personne identifiée dans le document. Notez que, dans cette syntaxe, les tableaux sont « indexés sur 0 ».

## Exemple 3 : référencer des membres à l'intérieur d'un tableau

Parfois, vous devez regrouper toutes les annotations d'un type particulier pour les transmettre à une compétence particulière. Imaginez une compétence personnalisée hypothétique qui identifie le nom de famille le plus courant parmi tous les noms de famille extraits dans l'Exemple 2. Pour fournir uniquement les noms de famille à la compétence personnalisée, spécifiez le contexte comme `"/document"` et l'entrée comme

```
"/document/people/*/lastname".
```

Notez que la cardinalité de `"/document/people/*/lastname"` est supérieure à celle du document. Il peut y avoir 10 nœuds de nom de famille alors qu'il n'y a qu'un seul nœud de document pour ce document. Dans ce cas, le système crée automatiquement un tableau de `"/document/people/*/lastname"` contenant tous les éléments du document.

```
{
  "@odata.type": "#Microsoft.Skills.Custom.WebApiSkill",
  "description": "Fictitious skill that gets the most common string from an array of strings",
  "uri": "http://names.azurewebsites.net/api/MostCommonString",
  "context" : "/document",
  "inputs": [
    {
      "name": "strings",
      "source": "/document/people/*/lastname"
    }
  ],
  "outputs": [
    {
      "name": "mostcommon",
      "targetName": "common-lastname"
    }
  ]
}
```

## Voir aussi

- [Comment intégrer une compétence personnalisée dans un pipeline d'enrichissement](#)
- [Guide pratique pour définir un ensemble de compétences](#)
- [Créer un jeu de compétences \(REST\)](#)
- [Guide pratique pour mapper des champs enrichis à un index](#)

# Guide pratique pour mapper des champs enrichis par IA sur un index pouvant faire l'objet d'une recherche

04/10/2020 • 6 minutes to read • [Edit Online](#)



Dans cet article, vous allez apprendre à mapper des champs d'entrée enrichis sur des champs de sortie dans un index pouvant faire l'objet d'une recherche. Une fois que vous avez [défini un ensemble de compétences](#), vous devez mapper les champs de sortie de n'importe quelle compétence qui fournit directement des valeurs sur un champ donné dans votre index de recherche.

Les mappages de champs de sortie sont nécessaires au déplacement de contenu de documents enrichis vers l'index. Le document enrichi est en fait une arborescence d'informations, et même si l'index prend en charge les types complexes, vous pouvez parfois souhaiter transformer les informations de l'arborescence enrichie en un type plus simple (par exemple, un tableau de chaînes). Les mappages de champs de sortie vous permettent d'effectuer des transformations de formes de données en aplatisant les informations. Les mappages de champs de sortie se produisent toujours après l'exécution d'un ensemble de compétences, même s'il est possible d'exécuter cette étape sans aucun ensemble de compétences défini.

Exemples de mappages de champs de sortie :

- Dans le cadre de votre ensemble de compétences, vous avez extrait les noms des organisations mentionnées dans chacune des pages de votre document. Vous souhaitez maintenant mapper chacun de ces noms d'organisation dans un champ de votre index de type Edm.Collection(Edm.String).
- Dans le cadre de votre ensemble de compétences, vous avez créé un nouveau nœud appelé « document/translated\_text ». Vous souhaitez mapper les informations de ce nœud à un champ spécifique dans votre index.
- Vous n'avez pas d'ensemble de compétences, mais vous indexez un type complexe à partir d'une base de données Cosmos DB. Vous aimeriez accéder à un nœud sur ce type complexe et le mapper à un champ de votre index.

## NOTE

Nous avons récemment activé la fonctionnalité de mappage des fonctions sur les mappages de champs de sortie. Pour plus d'informations sur les fonctions de mappage, consultez [Fonctions de mappage de champs](#).

## Utiliser outputFieldMappings

Pour mapper les champs, ajoutez `outputFieldMappings` à la définition de l'indexeur comme indiqué ci-dessous :

```
PUT https://[servicename].search.windows.net/indexers/[indexer name]?api-version=2020-06-30
api-key: [admin key]
Content-Type: application/json
```

Le corps de la demande est structuré comme suit :

```
{
  "name": "myIndexer",
  "dataSourceName": "myDataSource",
  "targetIndexName": "myIndex",
  "skillsetName": "myFirstSkillSet",
  "fieldMappings": [
    {
      "sourceFieldName": "metadata_storage_path",
      "targetFieldName": "id",
      "mappingFunction": {
        "name": "base64Encode"
      }
    }
  ],
  "outputFieldMappings": [
    {
      "sourceFieldName": "/document/content/organizations/*/description",
      "targetFieldName": "descriptions",
      "mappingFunction": {
        "name": "base64Decode"
      }
    },
    {
      "sourceFieldName": "/document/content/organizations",
      "targetFieldName": "orgNames"
    },
    {
      "sourceFieldName": "/document/content/sentiment",
      "targetFieldName": "sentiment"
    }
  ]
}
```

Pour chaque mappage de champ de sortie, définissez l'emplacement des données dans l'arborescence de documents enrichis (sourceFieldName) ainsi que le nom du champ tel que référencé dans l'index (targetFieldName).

## Aplatissement d'informations à partir de types complexes

Le chemin dans un sourceFieldName peut représenter un élément ou plusieurs éléments. Dans l'exemple ci-dessus, `/document/content/sentiment` représente une valeur numérique unique, tandis que `/document/content/organizations/*/description` représente plusieurs descriptions d'organisation.

Dans les cas où il existe plusieurs éléments, ceux-ci sont « aplatis » en un tableau qui contient chacun des éléments.

Plus concrètement, si nous considérons l'exemple `/document/content/organizations/*/description`, les données du champ *descriptions* ressembleraient à un tableau de descriptions aplati avant d'être indexées :

```
["Microsoft is a company in Seattle","LinkedIn's office is in San Francisco"]
```

S'agissant d'un principe important, nous allons examiner un autre exemple. Imaginez que vous disposez d'un tableau de types complexes parmi l'arborescence d'enrichissement. Supposons qu'un membre appelé

customEntities dispose d'un tableau de types complexes comme celui décrit ci-dessous.

```
"document/customEntities":  
[  
  {  
    "name": "heart failure",  
    "matches": [  
      {  
        "text": "heart failure",  
        "offset": 10,  
        "length": 12,  
        "matchDistance": 0.0  
      }  
    ]  
  },  
  {  
    "name": "morquio",  
    "matches": [  
      {  
        "text": "morquio",  
        "offset": 25,  
        "length": 7,  
        "matchDistance": 0.0  
      }  
    ]  
  }  
  //...  
]
```

Par ailleurs, vous possédez une index qui comporte un champ appelé « diseases » de type Collection(Edm.String) dans lequel vous souhaitez stocker chaque nom d'entité.

Cela peut se faire facilement en utilisant le symbole « \* », comme suit :

```
"outputFieldMappings": [  
  {  
    "sourceFieldName": "/document/customEntities/*/name",  
    "targetFieldName": "diseases"  
  }  
]
```

Cette opération a simplement pour effet d'« aplatisir » chaque nom d'élément customEntities dans un même tableau de chaînes, comme ceci :

```
"diseases" : ["heart failure", "morquio"]
```

## Étapes suivantes

Une fois que vous avez mappé les champs enrichis sur des champs pouvant faire l'objet d'une recherche, vous pouvez définir les attributs de chacun des champs pouvant faire l'objet d'une recherche [dans le cadre de la définition de l'index](#).

Pour en savoir plus sur le mappage de champs, consultez [Mappages de champs dans les indexeurs de Recherche cognitive Azure](#).

# Comment traiter et extraire des informations d'images dans des scénarios d'enrichissement de l'IA

04/10/2020 • 13 minutes to read • [Edit Online](#)

La Recherche cognitive Azure intègre plusieurs fonctionnalités pour l'utilisation d'images et de fichiers image. Pendant le décodage d'un document, vous pouvez utiliser le paramètre *imageAction* pour extraire du texte de photos ou d'images contenant du texte alphanumérique, tel que le mot « STOP » dans le panneau de signalisation d'arrêt. D'autres scénarios incluent la génération d'une représentation textuelle d'une image, telle que « pissenlit » pour une photo de pissenlit ou la couleur « jaune ». Vous pouvez également extraire des métadonnées de l'image, telles que sa taille.

Cet article couvre plus en détail le traitement d'image et fournit des conseils pour l'utilisation d'images dans un pipeline d'enrichissement de l'IA.

## Obtenir des images normalisées

Dans le cadre du décodage d'un document, il existe un nouvel ensemble de paramètres de configuration de l'indexeur pour gérer des fichiers image ou des images incorporées dans des fichiers. Ces paramètres permettent de normaliser des images en vue d'un traitement ultérieur en aval. La normalisation des images rend celles-ci plus uniformes. Les images de grande taille sont redimensionnées à une hauteur et une largeur maximales afin de les rendre utilisables. Pour les images fournissant des métadonnées d'orientation, la rotation est ajustée pour un chargement vertical. Les ajustements de métadonnées sont capturés dans un type complexe créé pour chaque image.

Vous ne pouvez pas désactiver la normalisation d'image. Les compétences qui itèrent sur des images attendent des images normalisées. L'activation de la normalisation d'image sur un indexeur nécessite l'attachement d'un ensemble de compétences à cet indexeur.

PARAMÈTRE DE CONFIGURATION	DESCRIPTION
imageAction	<p>Défini sur « none » si aucune action ne doit être effectuée quand des images incorporées ou des fichiers image sont rencontrés.</p> <p>Défini sur « generateNormalizedImages » pour générer un tableau d'images normalisées dans le cadre du décodage de document.</p> <p>Défini sur « generateNormalizedImages » pour générer un tableau d'images normalisées où, pour les PDF de votre source de données, chaque page s'affiche en tant qu'image de sortie unique. La fonctionnalité est la même que « generateNormalizedImages » pour les types de fichiers autres que PDF.</p> <p>Pour toute option autre que « none », les images seront exposées dans le champ <i>normalized_images</i>.</p> <p>La valeur par défaut est « none ». Cette configuration est pertinente uniquement pour des sources de données d'objet blob quand le paramètre « dataToExtract » est défini sur « contentAndMetadata ».</p> <p>Au maximum, 1 000 images seront extraites d'un document donné. Si un document contient plus de 1 000 images, les 1 000 premières seront extraites et un avertissement sera généré.</p>

PARAMÈTRE DE CONFIGURATION	DESCRIPTION
normalizedImageMaxWidth	Largeur maximale (en pixels) des images normalisées générées. La valeur par défaut est 2000. La valeur maximale autorisée est 10000.
normalizedImageMaxHeight	Hauteur maximale (en pixels) des images normalisées générées. La valeur par défaut est 2000. La valeur maximale autorisée est 10000.

#### NOTE

Si vous ne définissez pas la propriété *imageAction* sur « none », vous ne pouvez définir la propriété *parsingMode* que sur « default ». Vous ne pouvez définir qu'une seule de ces deux propriétés sur une valeur autre que la valeur par défaut dans la configuration de votre indexeur.

Définissez le paramètre **parsingMode** avec la valeur `json` (pour indexer chaque objet blob en tant que document unique) ou `jsonArray` (si vos objets blob contiennent des tableaux JSON et que vous souhaitez traiter chaque élément de tableau comme un document distinct).

La valeur par défaut de 2000 pixels pour la hauteur et la largeur maximales des images normalisées est basée sur les tailles maximales prises en charge par la [compétence de reconnaissance optique de caractères](#) et la [compétence d'analyse d'image](#). La [compétence de reconnaissance optique de caractères](#) (OCR) prend en charge une largeur et une hauteur maximales de 4200 pour les langues autres que l'anglais et de 10000 pour l'anglais. Si vous augmentez les limites maximales, le traitement des images plus volumineuses peut échouer en fonction de la définition de vos compétences et de la langue des documents.

Vous spécifiez la propriété *imageAction* dans votre [définition d'indexeur](#) comme suit :

```
{
  //...rest of your indexer definition goes here ...
  "parameters": {
    {
      "configuration": {
        {
          "dataToExtract": "contentAndMetadata",
          "imageAction": "generateNormalizedImages"
        }
      }
    }
  }
}
```

Si la propriété *imageAction* est définie sur une valeur autre que « none », le nouveau champ *normalized\_images* contiendra un tableau d'images. Chaque image est un type complexe comprenant les membres suivants :

MEMBRE DE L'IMAGE	DESCRIPTION
data	Chaîne codée en Base64 de l'image normalisée au format JPEG.
width	Largeur de l'image normalisée en pixels.
height	Hauteur de l'image normalisée en pixels.
originalWidth	Largeur d'origine de l'image avant la normalisation.

MEMBRE DE L'IMAGE	DESCRIPTION
originalHeight	Hauteur d'origine de l'image avant la normalisation.
rotationFromOriginal	Rotation dans le sens inverse des aiguilles d'une montre exprimée en degrés pour créer l'image normalisée. Valeur comprise entre 0 et 360 degrés. Cette étape lit les métadonnées de l'image générée par un appareil photo ou un scanner. La valeur est généralement un multiple de 90 degrés.
contentOffset	Offset de caractère à l'intérieur du champ de contenu dont l'image a été extraite. Ce champ est applicable uniquement aux fichiers contenant des images incorporées.
pageNumber	Si l'image a été extraite ou rendue à partir d'un PDF, ce champ contient le numéro de page du PDF à partir de laquelle elle a été extraite ou rendue, à partir de 1. Si l'image ne provient pas d'un fichier PDF, ce champ a la valeur 0.

Exemple de valeur de *normalized\_images*:

```
[
  {
    "data": "BASE64 ENCODED STRING OF A JPEG IMAGE",
    "width": 500,
    "height": 300,
    "originalWidth": 5000,
    "originalHeight": 3000,
    "rotationFromOriginal": 90,
    "contentOffset": 500,
    "pageNumber": 2
  }
]
```

## Compétences liées à l'image

Il existe deux compétences cognitives intégrées qui prennent des images en tant qu'entrée : la [reconnaissance optique des caractères](#) et [l'analyse d'image](#).

Actuellement, ces compétences fonctionnent uniquement avec des images générées à partir de l'étape de décodage du document. Par conséquent, la seule entrée prise en charge est `"/document/normalized_images"`.

### Compétence d'analyse d'image

La [compétence d'analyse d'image](#) extrait un ensemble riche de caractéristiques visuelles basées sur le contenu de l'image. Par exemple, à partir d'une image, vous pouvez générer une légende, générer des balises ou identifier des célébrités et des points de repère.

### Compétence de reconnaissance optique des caractères

La [compétence de reconnaissance optique des caractères](#) extrait du texte de fichiers image, par exemple, aux formats JPG, PNG et bitmap. Elle peut extraire du texte ainsi que des informations de disposition. Les informations de disposition fournissent des rectangles englobants pour chacune des chaînes identifiées.

## Scénario d'image incorporée

Un scénario courant implique la création d'une chaîne comprenant tous les contenus d'un fichier, tant le texte que le texte d'origine de l'image, en procédant comme suit :

1. Extraction de normalized\_images.
2. Exécution de la compétence de reconnaissance optique des caractères en utilisant `"/document/normalized_images"` en entrée.
3. Fusion de la représentation textuelle de ces images avec le texte brut extrait du fichier. Vous pouvez utiliser la compétence de [fusion de texte](#) pour combiner les deux blocs de texte dans une seule chaîne de grande taille.

L'exemple suivant de jeu de jeu de compétences crée un champ *merged\_text* comprenant le contenu textuel de votre document. Il inclut également le texte obtenu par reconnaissance optique des caractères de chacune des images incorporées.

#### Syntaxe du corps de la demande

```
{
  "description": "Extract text from images and merge with content text to produce merged_text",
  "skills": [
    {
      "description": "Extract text (plain and structured) from image.",
      "@odata.type": "#Microsoft.Skills.Vision.OcrSkill",
      "context": "/document/normalized_images/*",
      "defaultLanguageCode": "en",
      "detectOrientation": true,
      "inputs": [
        {
          "name": "image",
          "source": "/document/normalized_images/*"
        }
      ],
      "outputs": [
        {
          "name": "text"
        }
      ]
    },
    {
      "@odata.type": "#Microsoft.Skills.Text.MergeSkill",
      "description": "Create merged_text, which includes all the textual representation of each image inserted at the right location in the content field.",
      "context": "/document",
      "insertPreTag": " ",
      "insertPostTag": " ",
      "inputs": [
        {
          "name": "text", "source": "/document/content"
        },
        {
          "name": "itemsToInsert", "source": "/document/normalized_images/*/text"
        },
        {
          "name": "offsets", "source": "/document/normalized_images/*/contentOffset"
        }
      ],
      "outputs": [
        {
          "name": "mergedText", "targetName" : "merged_text"
        }
      ]
    }
  ]
}
```

À présent que vous avez un champ *merged\_text*, vous pouvez le mapper en tant que champ pouvant faire l'objet d'une recherche dans la définition de votre indexeur. Tout le contenu de vos fichiers, y compris le texte des images, sera disponible pour une recherche.

## Visualiser les rectangles englobants du texte extrait

Un autre scénario courant est la visualisation des informations de disposition des résultats de recherche. Par exemple, dans le cadre de vos résultats de recherche, vous pouvez vouloir mettre en évidence l'endroit où un élément de texte a été trouvé dans une image.

Étant donné que l'étape de reconnaissance optique des caractères est effectuée sur les images normalisées, les coordonnées de disposition figurent dans l'espace de l'image normalisée. Lorsque vous affichez l'image normalisée, la présence de coordonnées n'est généralement pas un problème mais, dans certains cas, il peut être utile d'afficher l'image d'origine. Dans ce cas, vous pouvez convertir chaque point de coordonnée dans la disposition vers le système de coordonnées de l'image d'origine.

Pour vous aider, si vous avez besoin de convertir des coordonnées normalisées vers l'espace des coordonnées d'origine, vous pouvez utiliser l'algorithme suivant :

```
/// <summary>
/// Converts a point in the normalized coordinate space to the original coordinate space.
/// This method assumes the rotation angles are multiples of 90 degrees.
/// </summary>
public static Point GetOriginalCoordinates(Point normalized,
                                            int originalWidth,
                                            int originalHeight,
                                            int width,
                                            int height,
                                            double rotationFromOriginal)
{
    Point original = new Point();
    double angle = rotationFromOriginal % 360;

    if (angle == 0 )
    {
        original.X = normalized.X;
        original.Y = normalized.Y;
    } else if (angle == 90)
    {
        original.X = normalized.Y;
        original.Y = (width - normalized.X);
    } else if (angle == 180)
    {
        original.X = (width - normalized.X);
        original.Y = (height - normalized.Y);
    } else if (angle == 270)
    {
        original.X = height - normalized.Y;
        original.Y = normalized.X;
    }

    double scalingFactor = (angle % 180 == 0) ? originalHeight / height : originalHeight / width;
    original.X = (int) (original.X * scalingFactor);
    original.Y = (int)(original.Y * scalingFactor);

    return original;
}
```

## Voir aussi

- [Créer un indexeur \(REST\)](#)
- [Compétence d'analyse d'image](#)
- [Compétence de reconnaissance optique des caractères](#)
- [Compétence de fusion de texte](#)
- [Guide pratique pour définir un ensemble de compétences](#)

- Guide pratique pour mapper des champs enrichis

# Guide de configuration de la mise en cache pour l'enrichissement incrémentiel dans Recherche cognitive Azure

04/10/2020 • 15 minutes to read • [Edit Online](#)

## IMPORTANT

L'enrichissement incrémentiel est actuellement en préversion publique. Cette préversion est fournie sans contrat de niveau de service et n'est pas recommandée pour les charges de travail de production. Pour plus d'informations, consultez [Conditions d'Utilisation Supplémentaires relatives aux Évaluations Microsoft Azure](#). Les préversions de l'API REST offrent cette fonctionnalité. Il n'y a pas de prise en charge de portail ou de SDK .NET pour l'instant.

Cet article explique comment ajouter la mise en cache à un pipeline d'enrichissement afin de pouvoir modifier les étapes de façon incrémentielle sans avoir à effectuer une régénération à chaque fois. Par défaut, un ensemble de compétences n'a pas d'état, et toute modification de sa composition nécessite une réexécution complète de l'indexeur. Grâce à l'enrichissement incrémentiel, l'indexeur peut déterminer les parties de l'arborescence de documents qui doivent être actualisées en fonction des modifications détectées dans l'ensemble de compétences ou les définitions de l'indexeur. La sortie traitée existante est conservée et réutilisée dans la mesure du possible.

Le contenu mis en cache est placé dans le stockage Azure à l'aide des informations de compte que vous fournissez. Le conteneur, nommé `ms-az-search-indexercache-<alpha-numerc-string>`, est créé lors de l'exécution de l'indexeur. Il doit être considéré comme un composant interne géré par votre service de recherche et ne doit pas être modifié.

Si le processus de configuration des indexeurs ne vous est pas familier, commencez par la [Présentation de l'indexeur](#), puis passez aux [Ensembles de compétences](#) pour en savoir plus sur les pipelines d'enrichissement. Pour plus d'informations sur les concepts clés, consultez [Enrichissement incrémentiel](#).

## Activer la mise en cache sur un indexeur existant

Si vous avez un indexeur existant qui est déjà doté d'un ensemble de compétences, suivez les étapes de cette section pour ajouter la mise en cache. Dans le cadre d'une opération ponctuelle, vous devez réinitialiser et réexécuter l'indexeur entièrement pour que le traitement incrémentiel puisse prendre effet.

## TIP

En guise de preuve de concept, vous pouvez utiliser ce [démarrage rapide du portail](#) pour créer les objets nécessaires, puis utiliser Postman ou le portail pour effectuer vos mises à jour. Vous souhaiterez peut-être joindre une ressource Cognitive Services facturable. L'exécution de l'indexeur à plusieurs reprises entraîne l'épuisement de l'allocation quotidienne gratuite avant que vous ne puissiez effectuer toutes les étapes.

### Étape 1 : Obtenir la définition de l'indexeur

Démarrez par un indexeur valide existant qui contient les composants suivants : source de données, ensemble de compétences, index. Votre indexeur doit être exécutable.

À l'aide d'un client d'API, créez une [requête GET Indexer](#) pour obtenir la configuration actuelle de l'indexeur. Lorsque vous utilisez la version préliminaire de l'API pour faire une requête GET sur l'indexeur, une propriété `cache` définie sur Null est ajoutée aux définitions.

```
GET https://[YOUR-SEARCH-SERVICE].search.windows.net/indexers/[YOUR-INDEXER-NAME]?api-version=2020-06-30-Preview  
Content-Type: application/json  
api-key: [YOUR-ADMIN-KEY]
```

Copiez la définition de l'indexeur à partir de la réponse.

### Étape 2 : Modifier la propriété du cache dans la définition de l'indexeur

Par défaut, la propriété `cache` est Null. Utilisez un client d'API pour définir la configuration du cache (le portail ne prend pas en charge cette mise à jour particulière).

Modifiez l'objet `cache` pour inclure les propriétés obligatoires et facultatives suivantes :

- La propriété `storageConnectionString` est obligatoire et doit être définie sur une chaîne de connexion de stockage Azure.
- La propriété booléenne `enableReprocessing` est facultative (`true` par défaut) et indique que l'enrichissement incrémentiel est activé. Si nécessaire, vous pouvez la définir sur `false` pour interrompre le traitement incrémentiel pendant que d'autres opérations gourmandes en ressources, telles que l'indexation de nouveaux documents, sont en cours d'exécution, puis la rétablir sur `true` plus tard.

```
{  
    "name": "<YOUR-INDEXER-NAME>",  
    "targetIndexName": "<YOUR-INDEX-NAME>",  
    "dataSourceName": "<YOUR-DATASOURCE-NAME>",  
    "skillsetName": "<YOUR-SKILLSET-NAME>",  
    "cache" : {  
        "storageConnectionString" : "<YOUR-STORAGE-ACCOUNT-CONNECTION-STRING>",  
        "enableReprocessing": true  
    },  
    "fieldMappings" : [],  
    "outputFieldMappings": [],  
    "parameters": []  
}
```

### Étape 3 : Réinitialiser l'indexeur

Une réinitialisation de l'indexeur est nécessaire lors de la configuration de l'enrichissement incrémentiel pour que les indexeurs existants puissent garantir que tous les documents sont dans un état cohérent. Vous pouvez utiliser le portail ou un client d'API, ainsi que l'[API REST Reset Indexer](#) (réinitialisation de l'indexeur) pour cette tâche.

```
POST https://[YOUR-SEARCH-SERVICE].search.windows.net/indexers/[YOUR-INDEXER-NAME]/reset?api-version=2020-06-30-Preview  
Content-Type: application/json  
api-key: [YOUR-ADMIN-KEY]
```

### Étape 4 : Enregistrer la définition mise à jour

Mettez à jour l'indexeur à l'aide d'une requête PUT. Le corps de la requête doit contenir la définition mise à jour de l'indexeur dotée de la propriété du cache. Si vous recevez une erreur 400, vérifiez la définition de l'indexeur pour vous assurer que toutes les exigences sont satisfaites (source de données, ensemble de compétences, index).

```
PUT https://[YOUR-SEARCH-SERVICE].search.windows.net/indexers/[YOUR-INDEXER-NAME]?api-version=2020-06-30-Preview
Content-Type: application/json
api-key: [YOUR-ADMIN-KEY]
{
    "name" : "<YOUR-INDEXER-NAME>",
    ...
    "cache": {
        "storageConnectionString": "<YOUR-STORAGE-ACCOUNT-CONNECTION-STRING>",
        "enableReprocessing": true
    }
}
```

Si vous émettez à présent une autre requête GET sur l'indexeur, la réponse du service inclut une propriété `ID` dans l'objet cache. La chaîne alphanumérique est ajoutée au nom du conteneur contenant tous les résultats mis en cache et l'état intermédiaire de chaque document traité par cet indexeur. L'ID sera utilisé pour nommer le cache de manière unique dans le stockage d'objets blob.

```
"cache": {
    "ID": "<ALPHA-NUMERIC STRING>",
    "enableReprocessing": true,
    "storageConnectionString": "DefaultEndpointsProtocol=https;AccountName=<YOUR-STORAGE-ACCOUNT>;AccountKey=<YOUR-STORAGE-KEY>;EndpointSuffix=core.windows.net"
}
```

## Étape 5 : Exécuter l'indexeur

Pour exécuter l'indexeur, vous pouvez utiliser le portail ou l'API. Dans le portail, à partir de la liste des indexeurs, sélectionnez l'indexeur souhaité, puis cliquez sur **Exécuter**. L'un des avantages à utiliser le portail est que vous pouvez surveiller l'état de l'indexeur, noter la durée du travail et connaître le nombre de documents traités. Les pages du portail sont actualisées à des intervalles de quelques minutes.

Vous pouvez également utiliser REST pour [exécuter l'indexeur](#) :

```
POST https://[YOUR-SEARCH-SERVICE].search.windows.net/indexers/[YOUR-INDEXER-NAME]/run?api-version=2020-06-30-Preview
Content-Type: application/json
api-key: [YOUR-ADMIN-KEY]
```

Après l'exécution de l'indexeur, vous trouverez le cache dans Stockage Blob Azure. Le nom du conteneur respecte le format suivant : `ms-az-search-indexercache-<YOUR-CACHE-ID>`

### NOTE

Une réinitialisation et une réexécution de l'indexeur entraînent une régénération complète afin que le contenu puisse être mis en cache. Tous les enrichissements cognitifs sont réexécutés sur tous les documents.

## Étape 6 : Modifier un ensemble de compétences et confirmer l'enrichissement incrémentiel

Pour modifier un ensemble de compétences, vous pouvez utiliser le portail ou l'API. Par exemple, si vous utilisez la traduction de texte, une simple modification inlined de `en` à `es` ou une autre langue suffit pour tester la preuve de concept de l'enrichissement incrémentiel.

Réexécutez l'indexeur. Seules les parties d'une arborescence enrichie de documents sont mises à jour. Si vous avez utilisé le [démarrage rapide du portail](#) comme preuve de concept, en modifiant la compétence de traduction de texte en « es », vous remarquerez que seuls 8 documents sont mis à jour au lieu des 14 originaux. Les fichiers image non concernés par le processus de traduction sont réutilisés à partir du cache.

## Activer la mise en cache sur les nouveaux indexeurs

Pour configurer l'enrichissement incrémentiel d'un nouvel indexeur, il vous suffit d'inclure la propriété `cache` dans la charge utile de la définition de l'indexeur lors de l'appel de [Create Indexer \(2020-06-30-Preview\)](#).

```
{  
    "name": "<YOUR-INDEXER-NAME>",  
    "targetIndexName": "<YOUR-INDEX-NAME>",  
    "dataSourceName": "<YOUR-DATASOURCE-NAME>",  
    "skillsetName": "<YOUR-SKILLSET-NAME>",  
    "cache": {  
        "storageConnectionString" : "<YOUR-STORAGE-ACCOUNT-CONNECTION-STRING>",  
        "enableReprocessing": true  
    },  
    "fieldMappings" : [],  
    "outputFieldMappings": [],  
    "parameters": []  
}  
}
```

## Vérification de la sortie mise en cache

Le cache est créé, utilisé et géré par l'indexeur, et son contenu n'est pas représenté dans un format lisible. Pour déterminer si le cache est utilisé, la meilleure méthode consiste à exécuter l'indexeur et à comparer les métriques avant et après du temps d'exécution et du nombre de documents.

Par exemple, supposez qu'il s'agit d'un ensemble de compétences qui commence par l'analyse des images et la reconnaissance optique de caractères (OCR) de documents numérisés, suivies de l'analyse en aval du texte résultant. Si vous modifiez une compétence de texte en aval, l'indexeur peut récupérer tout le contenu des images et de l'OCR précédemment traité à partir du cache, en mettant à jour et en traitant uniquement les modifications relatives au texte indiquées par vos modifications. Vous pouvez vous attendre à voir moins de documents dans le nombre de documents (par exemple 8/8 au lieu de 14/14 lors de l'exécution d'origine), des durées d'exécution plus courtes et moins de frais sur votre facture.

## Utilisation du cache

Une fois le cache opérationnel, les indexeurs vérifient le cache chaque fois que [Run Indexer](#) (Exécuter l'indexeur) est appelé, afin de déterminer quelles parties de la sortie existante peuvent être utilisées.

Le tableau suivant récapitule la relation entre différentes API et le cache :

API	IMPACT DU CACHE
<a href="#">Create Indexer (2020-06-30-Preview)</a>	Crée et exécute un indexeur lors de la première utilisation, y compris la création d'un cache si la définition de votre indexeur le spécifie.
<a href="#">Run Indexer</a>	Exécute un pipeline d'enrichissement à la demande. Cette API lit le cache, le cas échéant, ou crée un cache si vous avez ajouté la mise en cache à une définition d'indexeur mise à jour. Lorsque vous exécutez un indexeur pour lequel la mise en cache est activée, l'indexeur omet des étapes si la sortie mise en cache peut être utilisée. Vous pouvez utiliser la version généralement disponible ou la préversion de cette API.

API	IMPACT DU CACHE
<a href="#">Reset Indexer</a>	Efface toutes les informations d'indexation incrémentielles présentes dans l'indexeur. L'exécution suivante de l'indexeur (à la demande ou planifiée) est un retraitement complet à partir de zéro, y compris la réexécution de tous les ensembles de compétences et la recréation du cache. Sur le plan fonctionnel, cela équivaut à supprimer l'indexeur et à le recréer. Vous pouvez utiliser la version généralement disponible ou la préversion de cette API.
<a href="#">Reset Skills</a>	Spécifie les compétences à réexécuter lors de l'exécution suivante de l'indexeur, même si vous n'avez modifié aucune compétence. Le cache est mis à jour en conséquence. Les sorties, telles qu'une banque de connaissances ou un index de recherche, sont actualisées à l'aide de données réutilisables provenant du cache et du nouveau contenu selon la compétence mise à jour.

Pour plus d'informations sur le contrôle de ce qui se passe dans le cache, consultez [Gestion du cache](#).

## Étapes suivantes

L'enrichissement incrémentiel s'applique aux indexeurs qui contiennent des ensembles de compétences. Comme étape suivante, consultez la documentation relative aux ensembles de compétences pour comprendre les concepts et la composition.

En outre, une fois que vous avez activé le cache, vous souhaiterez connaître les paramètres et les API qui sont pris en compte dans la mise en cache, y compris la façon de substituer ou forcer des comportements particuliers. Pour plus d'informations, consultez les liens suivants :

- [Concepts et composition des ensembles de compétences](#)
- [Guide pratique pour créer un ensemble de compétences](#)
- [Présentation de l'enrichissement incrémentiel et de la mise en cache](#)

# Comment ajouter une compétence personnalisée à un pipeline d'enrichissement Recherche cognitive Azure

04/10/2020 • 8 minutes to read • [Edit Online](#)

Dans Recherche cognitive Azure, un [pipeline d'enrichissement](#) peut être assemblé à partir de [compétences cognitives prédéfinies](#) ainsi que de [compétences personnalisées](#) que vous avez créées et ajoutées personnellement au pipeline. Dans cet article, découvrez comment créer une compétence personnalisée qui présente une interface lui permettant d'être incluse dans un pipeline d'enrichissement d'intelligence artificielle.

Construire une compétence personnalisée vous donne un moyen d'insérer des transformations uniques dans votre contenu. Une compétence personnalisée s'exécute de façon indépendante, en appliquant l'étape d'enrichissement dont vous avez besoin. Par exemple, vous pouvez définir des entités personnalisées spécifiques de votre domaine, créer des modèles de classification personnalisés pour différencier des contrats et documents commerciaux et financiers, ou ajouter une compétence de reconnaissance vocale pour explorer plus profondément des fichiers audio afin d'en extraire du contenu pertinent. Pour obtenir un exemple pas à pas, consultez [Exemple : Création d'une compétence personnalisée pour l'enrichissement de l'IA](#).

Quelle que soit la capacité personnalisée dont vous avez besoin, il existe une interface simple et claire pour connecter une compétence personnalisée au reste du pipeline d'enrichissement. La seule exigence pour l'inclusion d'une capacité dans un [jeu de compétences](#) est la possibilité d'accepter des entrées et de générer des sorties de manières exploitables dans le jeu de compétences dans son ensemble. Cet article se concentre sur les formats d'entrée et de sortie que le pipeline d'enrichissement exige.

## Interface de compétence personnalisée d'API web

Par défaut, les points de terminaison des compétences WebAPI personnalisées expirent si aucune réponse n'est renvoyée dans les 30 secondes. Le pipeline d'indexation est synchrone et l'indexation produit une erreur de délai d'expiration si aucune réponse n'est reçue au terme de ces 30 secondes. Le paramètre d'expiration peut être allongé jusqu'à 230 secondes en définissant le paramètre `Expiration` :

```
"@odata.type": "#Microsoft.Skills.Custom.WebApiSkill",
"description": "This skill has a 230 second timeout",
"uri": "https://[your custom skill uri goes here]",
"timeout": "PT230S",
```

Assurez-vous que l'URI est sécurisé (HTTPS).

Actuellement, le seul mécanisme d'interaction avec une compétence personnalisée est l'interface d'API web. L'API web doit satisfaire les exigences décrites dans cette section.

### 1. Format d'entrée de l'API web

L'API web doit accepter un tableau d'enregistrements à traiter. Chaque enregistrement doit contenir un « jeu de propriétés » qui est l'entrée fournie à votre API web.

Supposons que vous souhaitez créer un enrichisseur simple qui identifie la première date mentionnée dans le texte d'un contrat. Dans cet exemple, la compétence accepte une entrée unique, *contractText*, en tant que texte du contrat. La compétence a également une sortie unique, qui est la date du contrat. Pour rendre l'enrichisseur plus intéressant, retournez cette *date de contrat* (*contractDate*) sous la forme d'un type complexe à parties multiples.

Votre API web doit être prête à recevoir un lot d'enregistrements d'entrée. Chaque élément du tableau *valeurs* représente l'entrée d'un enregistrement particulier. Chaque enregistrement doit comporter les éléments suivants :

- Un membre *recordId* qui est l'identificateur unique d'un enregistrement particulier. Quand votre enrichisseur retourne les résultats, il doit fournir ce *recordId* afin de permettre à l'appelant de faire correspondre les résultats de l'enregistrement à leur entrée.
- Un membre *date* qui est essentiellement un jeu de champs d'entrée pour chaque enregistrement.

Pour être plus concret, sur la base de l'exemple ci-dessus, votre API web doit attendre des demandes ressemblant à ceci :

```
{
  "values": [
    {
      "recordId": "a1",
      "data": {
        "contractText":
          "This is a contract that was issued on November 3, 2017 and that involves..."
      }
    },
    {
      "recordId": "b5",
      "data": {
        "contractText":
          "In the City of Seattle, WA on February 5, 2018 there was a decision made..."
      }
    },
    {
      "recordId": "c3",
      "data": {
        "contractText": null
      }
    }
  ]
}
```

En réalité, votre service peut être appelé avec des centaines voire des milliers d'enregistrements au lieu de seulement les trois présentés ici.

## 2. Format de sortie de l'API web

Le format de la sortie est un ensemble d'enregistrements contenant un *recordId* et un jeu de propriétés.

```
{
  "values": [
    {
      "recordId": "b5",
      "data" : {
        "contractDate": { "day" : 5, "month": 2, "year" : 2018 }
      }
    },
    {
      "recordId": "a1",
      "data" : {
        "contractDate": { "day" : 3, "month": 11, "year" : 2017 }
      }
    },
    {
      "recordId": "c3",
      "data" : {
        "errors": [ { "message": "contractText field required "} ],
        "warnings": [ {"message": "Date not found" } ]
      }
    }
  ]
}
```

Cet exemple particulier ne génère qu'une seule sortie, mais vous pouvez générer plus d'une propriété.

### **Erreurs et avertissements**

Comme le montre l'exemple précédent, vous pouvez renvoyer des messages d'erreur et d'avertissement pour chaque enregistrement.

## Utilisation des compétences personnalisées du jeu de compétences

Lorsque vous créez un enrichisseur d'API web, vous pouvez décrire des en-têtes HTTP et des paramètres dans le cadre de la demande. L'extrait de code ci-dessous montre comment des paramètres de requête et des en-têtes HTTP  *facultatifs* peuvent être décrits dans le cadre de la définition d'un jeu de compétences. Les en-têtes HTTP ne sont pas obligatoires, mais vous permettent d'ajouter des fonctionnalités de configuration supplémentaires à vos compétences et de les définir à partir de la définition de compétences.

```
{  
  "skills": [  
    {  
      "@odata.type": "#Microsoft.Skills.Custom.WebApiSkill",  
      "description": "This skill calls an Azure function, which in turn calls TA sentiment",  
      "uri": "https://indexer-e2e-webskill.azurewebsites.net/api/DateExtractor?language=en",  
      "context": "/document",  
      "httpHeaders": {  
        "DateExtractor-Api-Key": "foo"  
      },  
      "inputs": [  
        {  
          "name": "contractText",  
          "source": "/document/content"  
        }  
      ],  
      "outputs": [  
        {  
          "name": "contractDate",  
          "targetName": "date"  
        }  
      ]  
    }  
  ]  
}
```

## Étapes suivantes

Cet article a abordé les exigences d'interface nécessaires à l'intégration d'une compétence personnalisée dans un ensemble de compétences. Cliquez sur les liens suivants pour en savoir plus sur les compétences personnalisées et la composition d'un ensemble de compétences.

- [Regardez notre vidéo sur les compétences personnalisées](#)
- [Super compétences : référentiel de compétences personnalisées](#)
- [Exemple : Création d'une compétence personnalisée pour l'enrichissement par l'IA](#)
- [Guide pratique pour définir un ensemble de compétences](#)
- [Créer un jeu de compétences \(REST\)](#)
- [Guide pratique pour mapper des champs enrichis](#)

# Exemple : Créer une compétence personnalisée avec Python

04/10/2020 • 11 minutes to read • [Edit Online](#)

Dans cet exemple d'ensemble de compétences concernant la Recherche cognitive Azure, vous allez apprendre à créer une compétence personnalisée d'API web à l'aide de Python et de Visual Studio Code. L'exemple utilise une fonction Azure qui implémente l'[interface de la compétence personnalisée](#).

La conception de la compétence personnalisée est simple (elle concatène deux chaînes), ce qui vous permet de vous concentrer sur les outils et les technologies utilisés pour le développement de compétences personnalisées en Python. Une fois que vous avez réussi avec une compétence simple, vous pouvez vous baser sur celle-ci pour créer des scénarios plus complexes.

## Prérequis

- Pour connaître l'interface d'entrée et de sortie qu'une compétence personnalisée doit implémenter, examinez l'[interface de compétence personnalisée](#).
- Configurez votre environnement. Nous avons suivi [ce tutoriel de bout en bout](#) pour configurer une fonction Azure serverless à l'aide d'extensions Visual Studio Code et Python. Ce tutoriel va vous guider tout au long de l'installation des outils et des composants suivants :
  - [Python 3.75](#)
  - [Visual Studio Code](#)
  - [Extension Python pour Visual Studio Code](#)
  - [Azure Functions Core Tools](#)
  - [Extension Azure Functions pour Visual Studio Code](#)

## Création d'une fonction Azure

Cet exemple utilise une fonction Azure pour illustrer le concept d'hébergement d'une API web. Toutefois, d'autres approches sont possibles. Pour autant que vous respectiez les [exigences d'interface pour une compétence cognitive](#), l'approche que vous adoptez est sans importance. Toutefois, Azure Functions facilite la création d'une compétence personnalisée.

### Créer une application de fonction

Le modèle de projet Azure Functions dans Visual Studio Code crée un projet qui peut être publié dans une application de fonction sur Azure. Une application de fonctions vous permet de regrouper les fonctions dans une unité logique pour la gestion, le déploiement et le partage des ressources.

1. Dans Visual Studio Code, appuyez sur F1 pour ouvrir la palette de commandes. Dans la palette de commandes, recherchez et sélectionnez `Azure Functions: Create new project...`.
2. Choisissez un emplacement de répertoire pour votre espace de travail de projet et optez pour **Sélectionner**.

#### NOTE

Ces étapes ont été conçues pour être terminées en dehors d'un espace de travail. Pour cette raison, vous ne devez pas sélectionner un dossier de projet qui fait partie d'un espace de travail.

3. Sélectionnez un langage pour votre projet d'application de fonction. Pour ce tutoriel, sélectionnez **Python**.
4. Sélectionnez la version de Python (la version 3.7.5 est prise en charge par Azure Functions).
5. Sélectionnez un modèle pour la première fonction de votre projet. Sélectionnez un **déclencheur HTTP** pour créer une fonction déclenchée via HTTP dans la nouvelle application de fonction.
6. Attribuez un nom à la fonction. Dans ce cas, nous allons utiliser **Concatenator**.
7. Sélectionnez **Fonction** comme niveau d'autorisation. Cela signifie que nous allons fournir une **clé de fonction** pour appeler le point de terminaison HTTP de la fonction.
8. Sélectionnez la façon dont vous souhaitez ouvrir votre projet. Pour cette étape, sélectionnez **Ajouter à l'espace de travail** afin de créer l'application de fonction dans l'espace de travail actuel.

Visual Studio Code crée le projet d'application de fonction dans un nouvel espace de travail. Ce projet contient les fichiers de configuration `host.json` et `local.settings.json`, ainsi que des fichiers de projet spécifiques au langage.

Une nouvelle fonction déclenchée via HTTP est également créée dans le dossier **Concatenator** du projet d'application de fonction. Vous trouverez dans celui-ci un fichier nommé « `__init__.py` » dont le contenu est le suivant :

```
import logging

import azure.functions as func


def main(req: func.HttpRequest) -> func.HttpResponse:
    logging.info('Python HTTP trigger function processed a request.')

    name = req.params.get('name')
    if not name:
        try:
            req_body = req.get_json()
        except ValueError:
            pass
        else:
            name = req_body.get('name')

    if name:
        return func.HttpResponse(f"Hello {name}!")
    else:
        return func.HttpResponse(
            "Please pass a name on the query string or in the request body",
            status_code=400
        )
```

À présent, modifions ce code pour suivre l'[interface de compétence personnalisée](#). Modifiez le code avec le contenu suivant :

```
import logging
import azure.functions as func
import json


def main(req: func.HttpRequest) -> func.HttpResponse:
```

```

logging.info('Python HTTP trigger function processed a request.')

try:
    body = json.dumps(req.get_json())
except ValueError:
    return func.HttpResponse(
        "Invalid body",
        status_code=400
    )

if body:
    result = compose_response(body)
    return func.HttpResponse(result, mimetype="application/json")
else:
    return func.HttpResponse(
        "Invalid body",
        status_code=400
    )

def compose_response(json_data):
    values = json.loads(json_data)['values']

    # Prepare the Output before the loop
    results = {}
    results["values"] = []

    for value in values:
        output_record = transform_value(value)
        if output_record != None:
            results["values"].append(output_record)
    return json.dumps(results, ensure_ascii=False)

## Perform an operation on a record
def transform_value(value):
    try:
        recordId = value['recordId']
    except AssertionError as error:
        return None

    # Validate the inputs
    try:
        assert ('data' in value), "'data' field is required."
        data = value['data']
        assert ('text1' in data), "'text1' field is required in 'data' object."
        assert ('text2' in data), "'text2' field is required in 'data' object."
    except AssertionError as error:
        return (
            {
                "recordId": recordId,
                "errors": [ { "message": "Error:" + error.args[0] } ]
            }
        )

    try:
        concatenated_string = value['data']['text1'] + " " + value['data']['text2']
        # Here you could do something more interesting with the inputs
    except:
        return (
            {
                "recordId": recordId,
                "errors": [ { "message": "Could not complete operation for record." } ]
            }
        )

    return ({
        "recordId": recordId,
        "data": {
            "text": concatenated_string
        }
    })

```

})

La méthode **transform\_value** effectue une opération sur un seul enregistrement. Vous pouvez modifier la méthode en fonction de vos besoins. N'oubliez pas d'effectuer les validations d'entrée nécessaires, et de retourner une erreur ou un avertissement en cas d'échec de l'opération.

### Déboguer votre code localement

Visual Studio Code facilite le débogage de votre code. Appuyez sur F5 ou accédez au menu **Déboguer** pour sélectionner **Démarrer le débogage**.

Vous pouvez définir des points d'arrêt dans le code en appuyant sur F9 sur la ligne qui vous intéresse.

Une fois que vous avez démarré le débogage, votre fonction s'exécute localement. Vous pouvez utiliser un outil tel que Postman ou Fiddler pour envoyer la requête à localhost. Notez l'emplacement de votre point de terminaison local dans la fenêtre de terminal.

## Publier votre fonction

Lorsque vous êtes satisfait du comportement de la fonction, vous pouvez la publier.

1. Dans Visual Studio Code, appuyez sur F1 pour ouvrir la palette de commandes. Dans la palette de commandes, recherchez et sélectionnez **Déployer dans l'application de fonction...** .
2. Sélectionnez l'abonnement Azure dans lequel vous souhaitez déployer votre application.
3. Sélectionnez + **Créer une application de fonction dans Azure**.
4. Entrez un nom global unique pour votre application de fonction.
5. Sélectionnez la version de Python (Python 3.7.x est compatible avec cette fonction).
6. Sélectionnez un emplacement pour la nouvelle ressource (par exemple, USA Ouest 2).

À ce stade, les ressources nécessaires seront créées dans votre abonnement Azure pour héberger la nouvelle fonction Azure dans Azure. Attendez la fin du déploiement. La fenêtre de sortie affiche l'état du processus de déploiement.

1. Dans le [portail Azure](#), accédez à **Toutes les ressources**, puis recherchez la fonction que vous avez publiée à l'aide de son nom. Si vous l'avez nommée **Concatenator**, sélectionnez la ressource.
2. Cliquez sur le bouton </> **Obtenir l'URL de fonction**. Cela vous permettra de copier l'URL pour appeler la fonction.

## Tester la fonction dans Azure

À présent que vous disposez de la clé d'hôte par défaut, testez votre fonction comme suit :

```
POST [Function URL you copied above]
```

### Corps de la requête

```
{
  "values": [
    {
      "recordId": "e1",
      "data":
        {
          "text1": "Hello",
          "text2": "World"
        }
    },
    {
      "recordId": "e2",
      "data": "This is an invalid input"
    }
  ]
}
```

Cet exemple devrait produire le même résultat que celui que vous avez obtenu précédemment lors de l'exécution de la fonction dans l'environnement local.

## Connexion à votre pipeline

À présent que vous avez une nouvelle compétence personnalisée, vous pouvez l'ajouter à vos jeu de compétences. L'exemple ci-dessous montre comment appeler la compétence en vue de concaténer le titre et l'auteur du document dans un même champ unique que nous appellerons merged\_title\_author. Remplacez `[your-function-url-here]` par l'URL de votre nouvelle fonction Azure.

```
{
  "skills": [
    "[... your existing skills remain here]",
    {
      "@odata.type": "#Microsoft.Skills.Custom.WebApiSkill",
      "description": "Our new search custom skill",
      "uri": "https://[your-function-url-here]",
      "context": "/document/merged_content/organizations/*",
      "inputs": [
        {
          "name": "text1",
          "source": "/document/metadata_title"
        },
        {
          "name": "text2",
          "source": "/document/metadata_author"
        }
      ],
      "outputs": [
        {
          "name": "text",
          "targetName": "merged_title_author"
        }
      ]
    }
  ]
}
```

## Étapes suivantes

Félicitations ! Vous avez créé votre première qualification personnalisée. Vous pouvez maintenant suivre le même schéma pour ajouter vos propres fonctionnalités personnalisées. Pour en savoir plus, cliquez sur les liens suivants.

- [Super compétences : référentiel de compétences personnalisées](#)

- Ajouter une qualification personnalisée à un pipeline d'enrichissement par IA
- Guide pratique pour définir un ensemble de compétences
- Créer un jeu de compétences (REST)
- Guide pratique pour mapper des champs enrichis

# Exemple : Créer une qualification personnalisée à l'aide de l'API Recherche d'entités Bing

04/10/2020 • 14 minutes to read • [Edit Online](#)

Dans cet exemple, découvrez comment créer une compétence personnalisée d'API web. Cette qualification accepte des emplacements, des chiffres publics et des organisations, et renvoie leurs descriptions. L'exemple utilise une [fonction Azure](#) pour encapsuler l'[API Recherche d'entités Bing](#) de façon à ce qu'elle implémente l'interface de qualification personnalisée.

## Prérequis

- Si vous ne connaissez pas l'interface d'entrée/sortie qu'une compétence personnalisée doit implémenter, lisez l'article sur l'[interface de compétence personnalisée](#).

### • Créer une ressource Azure

Commencez à utiliser l'API Recherche d'entités Bing en créant une des ressources Azure suivantes.

#### Ressource Recherche d'entités Bing

- Disponible via le portail Azure jusqu'à ce que vous supprimiez la ressource.
- Utilisez le niveau tarifaire Gratuit pour tester le service, puis effectuez par la suite une mise à niveau vers un niveau payant pour la production.
- L'API Recherche d'entités Bing est aussi proposée à certains niveaux de la [ressource Recherche Bing v7](#).

#### Ressource multiservice

- Disponible via le portail Azure jusqu'à ce que vous supprimiez la ressource.
- Utilisez la même clé et le même point de terminaison pour vos applications, dans plusieurs services Cognitive Services.
- Installez [Visual Studio 2019](#) ou une version ultérieure, incluant la charge de travail de développement Azure.

## Création d'une fonction Azure

Bien que cet exemple utilise une Fonction Azure pour héberger une API web, elle n'est pas indispensable. Pour autant que vous respectiez les [exigences d'interface pour une compétence cognitive](#), l'approche que vous adoptez est sans importance. Toutefois, Azure Functions facilite la création d'une compétence personnalisée.

#### Créer une application de fonction

1. Dans Visual Studio, dans le menu Fichier, sélectionnez **Nouveau > Projet**.
2. Dans la boîte de dialogue Nouveau projet, sélectionnez **Installé**, développez **Visual C# > Cloud**, sélectionnez **Azure Functions**, tapez un Nom pour votre projet, puis cliquez sur OK. Le nom d'application de la fonction doit être valide en tant qu'espace de noms C#, afin de ne pas utiliser des traits d'union, des traits de soulignement ou d'autres caractères non alphanumériques.
3. Sélectionnez **Azure Functions v2 (.NET Core)**. Vous pouvez également utiliser la version 1, mais le code écrit ci-dessous est basé sur le modèle v2.
4. Sélectionnez le type **Déclencheur HTTP**

5. Pour Compte de stockage, vous pouvez sélectionner **Aucun**, car vous n'aurez besoin d'aucun stockage pour cette fonction.

6. Sélectionnez **OK** pour créer le projet de fonction et la fonction HTTP déclenchée.

### Modifier le code pour appeler le service Recherche d'entités Bing

Visual Studio crée un projet qui contient une classe qui présente un code réutilisable pour le type de fonction sélectionné. L'attribut *FunctionName* de la méthode définit le nom de la fonction. L'attribut *HttpTrigger* spécifie que la fonction est déclenchée par une requête HTTP.

À présent, remplacez tout le contenu du fichier *Function1.cs* par le code suivant :

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Net.Http;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Azure.WebJobs;
using Microsoft.Azure.WebJobs.Extensions.Http;
using Microsoft.AspNetCore.Http;
using Microsoft.Extensions.Logging;
using Newtonsoft.Json;

namespace SampleSkills
{
    /// <summary>
    /// Sample custom skill that wraps the Bing entity search API to connect it with a
    /// AI enrichment pipeline.
    /// </summary>
    public static class BingEntitySearch
    {
        #region Credentials
        // IMPORTANT: Make sure to enter your credential and to verify the API endpoint matches yours.
        static readonly string bingApiEndpoint =
"https://api.cognitive.microsoft.com/bing/v7.0/entities/";
        static readonly string key = "<enter your api key here>";
        #endregion

        #region Class used to deserialize the request
        private class InputRecord
        {
            public class InputRecordData
            {
                public string Name { get; set; }
            }

            public string RecordId { get; set; }
            public InputRecordData Data { get; set; }
        }
        #endregion

        #region Classes used to serialize the response

        private class OutputRecord
        {
            public class OutputRecordData
            {
                public string Name { get; set; } = "";
                public string Description { get; set; } = "";
            }
        }
    }
}
```

```

        public string Source { get; set; } = "";
        public string SourceUrl { get; set; } = "";
        public string LicenseAttribution { get; set; } = "";
        public string LicenseUrl { get; set; } = "";
    }

    public class OutputRecordMessage
    {
        public string Message { get; set; }
    }

    public string RecordId { get; set; }
    public OutputRecordData Data { get; set; }
    public List<OutputRecordMessage> Errors { get; set; }
    public List<OutputRecordMessage> Warnings { get; set; }
}

private class WebApiResponse
{
    public List<OutputRecord> Values { get; set; }
}

#endregion

#region Classes used to interact with the Bing API
private class BingResponse
{
    public BingEntities Entities { get; set; }
}

private class BingEntities
{
    public BingEntity[] Value { get; set; }
}

private class BingEntity
{
    public class EntityPresentationinfo
    {
        public string[] EntityTypeHints { get; set; }
    }

    public class License
    {
        public string Url { get; set; }
    }

    public class ContractualRule
    {
        public string _type { get; set; }
        public License License { get; set; }
        public string LicenseNotice { get; set; }
        public string Text { get; set; }
        public string Url { get; set; }
    }

    public ContractualRule[] ContractualRules { get; set; }
    public string Description { get; set; }
    public string Name { get; set; }
    public EntityPresentationinfo EntityPresentationInfo { get; set; }
}
#endregion

#region The Azure Function definition

[FunctionName("EntitySearch")]
public static async Task<IActionResult> Run(
    [HttpTrigger(AuthorizationLevel.Function, "post", Route = null)] HttpRequest req,
    ILogger log)
{
    log.LogInformation("Entity Search function: C# HTTP trigger function processed a request.");
}

```

```

        var response = new WebApiResponse
    {
        Values = new List<OutputRecord>()
    };

    string requestBody = new StreamReader(req.Body).ReadToEnd();
    var data = JsonConvert.DeserializeObject<WebApiRequest>(requestBody);

    // Do some schema validation
    if (data == null)
    {
        return new BadRequestObjectResult("The request schema does not match expected schema.");
    }
    if (data.Values == null)
    {
        return new BadRequestObjectResult("The request schema does not match expected schema.
Could not find values array.");
    }

    // Calculate the response for each value.
    foreach (var record in data.Values)
    {
        if (record == null || record.RecordId == null) continue;

        OutputRecord responseRecord = new OutputRecord
        {
            RecordId = record.RecordId
        };

        try
        {
            responseRecord.Data = GetEntityMetadata(record.Data.Name).Result;
        }
        catch (Exception e)
        {
            // Something bad happened, log the issue.
            var error = new OutputRecord.OutputRecordMessage
            {
                Message = e.Message
            };

            responseRecord.Errors = new List<OutputRecord.OutputRecordMessage>
            {
                error
            };
        }
        finally
        {
            response.Values.Add(responseRecord);
        }
    }

    return (ActionResult)new OkObjectResult(response);
}

#endregion

#region Methods to call the Bing API
/// <summary>
/// Gets metadata for a particular entity based on its name using Bing Entity Search
/// </summary>
/// <param name="entityName">The name of the entity to extract data for.</param>
/// <returns>Asynchronous task that returns entity data. </returns>
private async static Task<OutputRecord.OutputRecordData> GetEntityMetadata(string entityName)
{
    var uri = bingApiEndpoint + "?q=" + entityName + "&mkt=en-
us&count=10&offset=0&safesearch=Moderate";
    var result = new OutputRecord.OutputRecordData();
}

```

```

        using (var client = new HttpClient())
        using (var request = new HttpRequestMessage {
            Method = HttpMethod.Get,
            RequestUri = new Uri(uri)
        })
        {
            request.Headers.Add("Ocp-Apim-Subscription-Key", key);

            HttpResponseMessage response = await client.SendAsync(request);
            string responseBody = await response?.Content?.ReadAsStringAsync();

            BingResponse bingResult = JsonConvert.DeserializeObject<BingResponse>(responseBody);
            if (bingResult != null)
            {
                // In addition to the list of entities that could match the name, for simplicity
                let's return information
                // for the top match as additional metadata at the root object.
                return AddTopEntityMetadata(bingResult.Entities?.Value);
            }
        }

        return result;
    }

    private static OutputRecord.OutputRecordData AddTopEntityMetadata(BingEntity[] entities)
    {
        if (entities != null)
        {
            foreach (BingEntity entity in entities.Where(
                entity => entity?.EntityPresentationInfo?.EntityTypeHints != null
                && (entity.EntityPresentationInfo.EntityTypeHints[0] == "Person"
                    || entity.EntityPresentationInfo.EntityTypeHints[0] == "Organization"
                    || entity.EntityPresentationInfo.EntityTypeHints[0] == "Location")
                && !String.IsNullOrEmpty(entity.Description)))
            {
                var rootObject = new OutputRecord.OutputRecordData
                {
                    Description = entity.Description,
                    Name = entity.Name
                };

                if (entity.ContractualRules != null)
                {
                    foreach (var rule in entity.ContractualRules)
                    {
                        switch (rule._type)
                        {
                            case "ContractualRules/LicenseAttribution":
                                rootObject.LicenseAttribution = rule.LicenseNotice;
                                rootObject.LicenseUrl = rule.License.Url;
                                break;
                            case "ContractualRules/LinkAttribution":
                                rootObject.Source = rule.Text;
                                rootObject.SourceUrl = rule.Url;
                                break;
                        }
                    }
                }

                return rootObject;
            }
        }

        return new OutputRecord.OutputRecordData();
    }
}
#endifregion
}

```

Veillez à entrer votre propre valeur de clé dans la constante `key` en fonction de la clé que vous avez obtenue lors de l'inscription à l'API Recherche d'entités Bing.

Cet exemple inclut tous le code nécessaire dans un fichier unique pour plus de commodité. Vous pouvez trouver une version légèrement plus structurée de cette qualification dans le [référentiel des qualifications puissantes](#).

Vous pouvez bien sûr renommer le fichier de `Function1.cs` en `BingEntitySearch.cs`.

## Tester la fonction à partir de Visual Studio

Appuyez sur **F5** pour exécuter le programme et tester le comportement de la fonction. En l'occurrence, nous allons utiliser la fonction ci-dessous pour examiner deux entités. Utilisez Postman ou Fiddler pour émettre un appel tel que celui ci-dessous :

```
POST https://localhost:7071/api/EntitySearch
```

### Corps de la demande

```
{
  "values": [
    {
      "recordId": "e1",
      "data":
        {
          "name": "Pablo Picasso"
        }
    },
    {
      "recordId": "e2",
      "data":
        {
          "name": "Microsoft"
        }
    }
  ]
}
```

### réponse

Vous devriez voir une réponse similaire à l'exemple suivant :

```
{
  "values": [
    {
      "recordId": "e1",
      "data": {
        "name": "Pablo Picasso",
        "description": "Pablo Ruiz Picasso was a Spanish painter [...]",
        "source": "Wikipedia",
        "sourceUrl": "http://en.wikipedia.org/wiki/Pablo_Picasso",
        "licenseAttribution": "Text under CC-BY-SA license",
        "licenseUrl": "http://creativecommons.org/licenses/by-sa/3.0/"
      },
      "errors": null,
      "warnings": null
    },
    ...
  ]
}
```

## Publier la fonction sur Azure

Lorsque vous êtes satisfait du comportement de la fonction, vous pouvez la publier.

1. Dans l'**Explorateur de solutions**, cliquez avec le bouton droit sur le projet, puis sélectionnez **Publier**. Choisissez **Créer > Publier**.
2. Si vous n'avez pas encore connecté Visual Studio à votre compte Azure, sélectionnez **Ajouter un compte...** .
3. Suivez les invitations qui s'affichent à l'écran. Vous êtes invité à spécifier un nom unique pour votre service d'application, l'abonnement Azure, le groupe de ressources, le plan d'hébergement et le compte de stockage que vous souhaitez utiliser. Si vous n'en avez pas de groupe de ressources, de plan d'hébergement et de compte de stockage, vous pouvez les créer. Quand vous avez terminé, sélectionnez **Créer**.
4. Une fois le déploiement terminé, notez l'URL du site. Il s'agit de l'adresse de votre application de fonction dans Azure.
5. Dans le [portail Azure](#), accédez au groupe de ressources, puis recherchez la fonction `EntitySearch` que vous avez publiée. Dans la section **Gérer**, vous devriez voir Host Keys. Sélectionnez l'icône **Copier** pour la clé d'hôte *par défaut*.

## Tester la fonction dans Azure

À présent que vous disposez de la clé d'hôte par défaut, testez votre fonction comme suit :

```
POST https://[your-entity-search-app-name].azurewebsites.net/api/EntitySearch?code=[enter default host key here]
```

### Corps de la requête

```
{
  "values": [
    {
      "recordId": "e1",
      "data":
        {
          "name": "Pablo Picasso"
        }
    },
    {
      "recordId": "e2",
      "data":
        {
          "name": "Microsoft"
        }
    }
  ]
}
```

Cet exemple devrait produire le même résultat que celui que vous avez obtenu précédemment lors de l'exécution de la fonction dans l'environnement local.

## Connexion à votre pipeline

À présent que vous avez une nouvelle compétence personnalisée, vous pouvez l'ajouter à vos jeu de compétences. L'exemple ci-dessous montre comment appeler la qualification pour ajouter des descriptions aux organisations dans le document (il pourrait être étendu à des emplacements ou à des personnes).

Remplacez `[your-entity-search-app-name]` par le nom de votre application.

```
{
  "skills": [
    "[... your existing skills remain here]",
    {
      "@odata.type": "#Microsoft.Skills.Custom.WebApiSkill",
      "description": "Our new Bing entity search custom skill",
      "uri": "https://[your-entity-search-app-name].azurewebsites.net/api/EntitySearch?code=[enter default host key here]",
      "context": "/document/merged_content/organizations/*",
      "inputs": [
        {
          "name": "name",
          "source": "/document/merged_content/organizations/*"
        }
      ],
      "outputs": [
        {
          "name": "description",
          "targetName": "description"
        }
      ]
    }
  ]
}
```

Ici, nous comptons sur le fait que la [compétence de reconnaissance d'entité](#) intégrée soit présente dans l'ensemble de qualifications et ait enrichi le document avec la liste d'organisations. Pour référence, voici une configuration de qualification d'extraction d'entité qui suffirait pour générer les données dont nous avons besoin :

```
{  
    "@odata.type": "#Microsoft.Skills.Text.EntityRecognitionSkill",  
    "name": "#1",  
    "description": "Organization name extraction",  
    "context": "/document/merged_content",  
    "categories": [ "Organization" ],  
    "defaultLanguageCode": "en",  
    "inputs": [  
        {  
            "name": "text",  
            "source": "/document/merged_content"  
        },  
        {  
            "name": "languageCode",  
            "source": "/document/language"  
        }  
    ],  
    "outputs": [  
        {  
            "name": "organizations",  
            "targetName": "organizations"  
        }  
    ]  
},
```

## Étapes suivantes

Félicitations ! Vous avez créé votre première qualification personnalisée. Vous pouvez maintenant suivre le même schéma pour ajouter vos propres fonctionnalités personnalisées. Pour en savoir plus, cliquez sur les liens suivants.

- [Super compétences : référentiel de compétences personnalisées](#)
- [Ajouter une qualification personnalisée à un pipeline d'enrichissement par IA](#)
- [Guide pratique pour définir un ensemble de compétences](#)
- [Créer un jeu de compétences \(REST\)](#)
- [Guide pratique pour mapper des champs enrichis](#)

# Exemple : Créer une compétence personnalisée Form Recognizer

04/10/2020 • 13 minutes to read • [Edit Online](#)

Dans cet exemple d'ensemble de compétences de la Recherche cognitive Azure, vous allez apprendre à créer une compétence personnalisée Form Recognizer à l'aide de C# et de Visual Studio. Form Recognizer analyse des documents et extrait les paires clé/valeur et les données de table. En wrappant Form Recognizer dans l'[interface de compétence personnalisée](#), vous pouvez ajouter cette fonctionnalité en tant qu'étape d'un pipeline d'enrichissement de bout en bout. Le pipeline peut ensuite charger les documents et effectuer d'autres transformations.

## Prérequis

- [Visual Studio 2019](#) (toute édition).
- Au moins cinq formulaires du même type. Vous pouvez utiliser les exemples de données fournis avec ce guide.

## Créer une ressource Form Recognizer

Accédez au portail Azure et [Créer une ressource Form Recognizer](#). Dans le volet **Créer**, indiquez les informations suivantes :

Nom	Nom descriptif de votre ressource. Nous recommandons d'utiliser un nom explicite, par exemple <i>MyNameFormRecognizer</i> .
Abonnement	Sélectionnez l'abonnement Azure auquel l'accès a été accordé.
Lieu	Emplacement de votre instance Cognitive Services. Des emplacements différents peuvent entraîner une latence. Toutefois, cela n'aura pas d'impact sur la disponibilité d'exécution de votre ressource.
Niveau tarifaire	Le coût de la ressource dépend du niveau tarifaire que vous choisissez et de votre utilisation. Pour plus d'informations, consultez le <a href="#">détail des tarifs</a> de l'API.
Groupe de ressources	<a href="#">Groupe de ressources Azure</a> comprenant votre ressource. Vous pouvez créer un groupe ou l'ajouter à un groupe préexistant.

### NOTE

Normalement, lorsque vous créez une ressource Cognitive Services dans le portail Azure, vous avez la possibilité de créer une clé d'abonnement multiservice (utilisée dans plusieurs services cognitifs) ou une clé d'abonnement à un seul service (utilisée uniquement avec un service cognitif spécifique). Cependant, Form Recognizer n'est pas compris dans l'abonnement multiservice pour le moment.

Lorsque le déploiement de la ressource Form Recognizer se termine, recherchez-la et sélectionnez-la dans la liste

Toutes les ressources dans le portail. Votre clé et votre point de terminaison se trouvent dans la page de la clé et du point de terminaison de la ressource, sous Gestion des ressources. Enregistrez ces deux éléments à un emplacement temporaire avant de continuer.

## Entraîner votre modèle

Vous devez entraîner un modèle Form Recognizer avec vos formulaires d'entrée avant d'utiliser cette compétence. Suivez le [guide de démarrage rapide cURL](#) pour savoir comment entraîner un modèle. Vous pouvez utiliser les exemples de formulaire fournis dans ce guide de démarrage rapide ou utiliser vos propres données. Une fois le modèle entraîné, copiez sa valeur d'ID dans un emplacement sécurisé.

## Configurer la compétence personnalisée

Ce tutoriel utilise le projet [AnalyzeForm](#) dans le dépôt GitHub [Super compétences de la Recherche Azure](#). Clonez ce dépôt sur votre ordinateur local, puis naviguez vers [Vision/AnalyzeForm/](#) pour accéder au projet. Ouvrez ensuite *AnalyzeForm.csproj* dans Visual Studio. Ce projet crée une ressource Azure Function qui satisfait l'[interface de compétence personnalisée](#) et qui peut être utilisée pour l'enrichissement de la Recherche cognitive Azure. Il prend des documents de formulaire comme entrées et génère (sous forme de texte) les paires clé/valeur que vous spécifiez.

Tout d'abord, ajoutez des variables d'environnement au niveau du projet. Localisez le projet **AnalyzeForm** dans le volet gauche, cliquez dessus avec le bouton droit, puis sélectionnez **Propriétés**. Dans la fenêtre **Propriétés**, cliquez sur l'onglet **Débogage**, puis recherchez le champ **Variables d'environnement**. Cliquez sur **Ajouter** pour ajouter les variables suivantes :

- `FORMS_RECOGNIZER_ENDPOINT_URL` avec la valeur définie sur l'URL de votre point de terminaison.
- `FORMS_RECOGNIZER_API_KEY` avec la valeur définie sur votre clé d'abonnement.
- `FORMS_RECOGNIZER_MODEL_ID` avec la valeur définie sur l'ID du modèle que vous avez entraîné.
- `FORMS_RECOGNIZER_RETRY_DELAY` avec la valeur définie sur 1000. Cette valeur est le temps, en millisecondes, pendant lequel le programme attend avant de retester la requête.
- `FORMS_RECOGNIZER_MAX_ATTEMPTS` avec la valeur définie sur 100. Cette valeur est le nombre de fois où le programme interroge le service en tentant d'obtenir une réponse correcte.

Ensuite, ouvrez *AnalyzeForm.cs*, puis recherchez la variable `fieldMappings`, qui fait référence au fichier *field-mappings.json*. Ce fichier (et la variable qui y fait référence) définissent la liste des clés que vous voulez extraire de vos formulaires et une étiquette personnalisée pour chaque clé. Par exemple, la valeur  
`{ "Address:", "address" }, { "Invoice For:", "recipient" }` signifie que le script enregistre uniquement les valeurs définies pour les champs `Address:` et `Invoice For:` détectés, et qu'il étiquette ces valeurs avec `"address"` et `"recipient"`, respectivement.

Enfin, notez la variable `contentType`. Ce script exécute le modèle Form Recognizer donné sur des documents distants qui sont référencés par URL, de sorte que le type de contenu est `application/json`. Si vous voulez analyser des fichiers locaux en incluant leurs flux d'octets dans les requêtes HTTP, vous devez remplacer le `contentType` par le [type MIME](#) approprié pour votre fichier.

## Tester la fonction à partir de Visual Studio

Une fois que vous avez modifié votre projet, enregistrez-le et définissez le projet **AnalyzeForm** comme projet de démarrage dans Visual Studio (s'il n'est pas déjà défini). Appuyez ensuite sur F5 pour exécuter la fonction dans votre environnement local. Utilisez un service REST comme [Postman](#) pour appeler la fonction.

### Demande HTTP

Vous allez effectuer la requête suivante pour appeler la fonction.

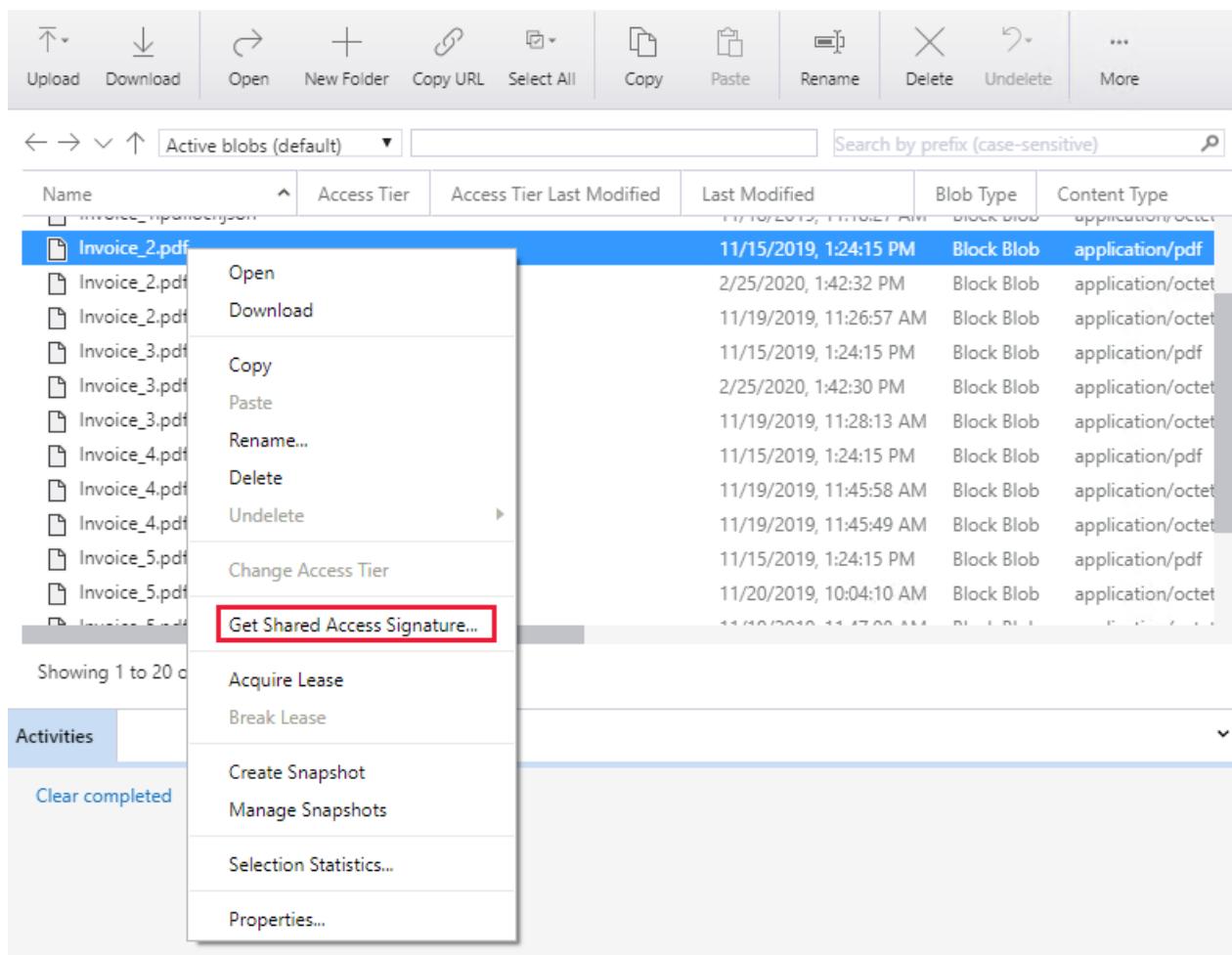
```
POST https://localhost:7071/api/analyze-form
```

## Corps de la demande

Commencez par le modèle de corps de la demande ci-dessous.

```
{  
    "values": [  
        {  
            "recordId": "record1",  
            "data": {  
                "formUrl": "<your-form-url>",  
                "formSasToken": "<your-sas-token>"  
            }  
        }  
    ]  
}
```

Ici, vous devez fournir l'URL d'un formulaire qui a le même type que les formulaires avec lesquels vous avez effectué l'entraînement. À des fins de test, vous pouvez utiliser l'un de vos formulaires d'entraînement. Si vous avez suivi le guide de démarrage rapide cURL, vos formulaires se trouvent dans un compte de stockage d'objets blob Azure. Ouvrez l'Explorateur Stockage Azure, localisez un fichier de formulaire, cliquez dessus avec le bouton droit, puis sélectionnez **Obtenir une signature d'accès partagé**. La fenêtre de boîte de dialogue suivante fournit une URL et un jeton SAS. Entrez ces chaînes dans les champs `"formUrl"` et `"formSasToken"` du corps de votre demande, respectivement.



Si vous voulez analyser un document distant qui ne se trouve pas dans le stockage d'objets blob Azure, collez son URL dans le champ `"formUrl"` et laissez le champ `"formSasToken"` vide.

#### NOTE

Quand la compétence est intégrée dans un ensemble de compétences, l'URL et le jeton sont fournis par la Recherche cognitive.

#### response

Vous devriez voir une réponse similaire à l'exemple suivant :

```
{  
    "values": [  
        {  
            "recordId": "record1",  
            "data": {  
                "address": "1111 8th st. Bellevue, WA 99501 ",  
                "recipient": "Southridge Video 1060 Main St. Atlanta, GA 65024 "  
            },  
            "errors": null,  
            "warnings": null  
        }  
    ]  
}
```

## Publier la fonction sur Azure

Lorsque vous êtes satisfait du comportement de la fonction, vous pouvez la publier.

1. Dans l'**Explorateur de solutions** dans Visual Studio, cliquez avec le bouton droit sur le projet, puis sélectionnez **Publier**. Choisissez **Créer > Publier**.
2. Si vous n'avez pas encore connecté Visual Studio à votre compte Azure, sélectionnez **Ajouter un compte...**.
3. Suivez les invitations qui s'affichent à l'écran. Spécifiez un nom unique pour votre service d'application, l'abonnement Azure, le groupe de ressources, le plan d'hébergement et le compte de stockage que vous voulez utiliser. Si vous n'avez pas encore de groupe de ressources, de plan d'hébergement et de compte de stockage, vous pouvez les créer. Quand vous avez terminé, sélectionnez **Créer**.
4. Une fois le déploiement terminé, notez l'URL du site. Cette URL est l'adresse de votre application de fonction dans Azure. Enregistrez-la dans un emplacement temporaire.
5. Dans le [portail Azure](#), accédez au groupe de ressources, puis recherchez la fonction `AnalyzeForm` que vous avez publiée. Dans la section **Gérer**, vous devriez voir Host Keys. Copiez la clé d'hôte *par défaut*, puis enregistrez-la dans un emplacement temporaire.

## Connexion à votre pipeline

Pour utiliser cette compétence dans un pipeline de Recherche cognitive, vous devez ajouter une définition de compétence à votre ensemble de compétences. Le bloc JSON suivant est un exemple de définition de compétence (vous devez mettre à jour les entrées et les sorties pour refléter l'environnement de votre scénario et votre ensemble de compétences spécifiques). Remplacez `AzureFunctionEndpointUrl` par votre URL de fonction, puis remplacez `AzureFunctionDefaultHostKey` par votre clé d'hôte.

```
{
  "description": "Skillset that invokes the Form Recognizer custom skill",
  "skills": [
    "[... your existing skills go here]",
    {
      "@odata.type": "#Microsoft.Skills.Custom.WebApiSkill",
      "name": "formrecognizer",
      "description": "Extracts fields from a form using a pre-trained form recognition model",
      "uri": "[AzureFunctionEndpointUrl]/api/analyze-form?code=[AzureFunctionDefaultHostKey]",
      "httpMethod": "POST",
      "timeout": "PT30S",
      "context": "/document",
      "batchSize": 1,
      "inputs": [
        {
          "name": "formUrl",
          "source": "/document/metadata_storage_path"
        },
        {
          "name": "formSasToken",
          "source": "/document/metadata_storage_sas_token"
        }
      ],
      "outputs": [
        {
          "name": "address",
          "targetName": "address"
        },
        {
          "name": "recipient",
          "targetName": "recipient"
        }
      ]
    }
  ]
}
```

## Étapes suivantes

Dans ce guide, vous avez créé une compétence personnalisée à partir du service Azure Form Recognizer. Pour en savoir plus sur les compétences personnalisées, consultez les ressources suivantes.

- [Super compétences de la Recherche Azure : un dépôt de compétences personnalisées](#)
- [Ajouter une qualification personnalisée à un pipeline d'enrichissement par IA](#)
- [How to create a skillset in an enrichment pipeline](#) (Créer un ensemble de compétences dans un pipeline d'enrichissement)
- [Créer un ensemble de compétences \(REST\)](#)
- [Mapper des champs enrichis](#)

# Tutoriel : Créer et déployer une compétence personnalisée avec Azure Machine Learning

04/10/2020 • 11 minutes to read • [Edit Online](#)

Dans ce tutoriel, vous utilisez le [jeu de données des avis sur les hôtels](#) (distribué dans le cadre de la licence Creative Commons [CC BY-NC-SA 4.0](#)) et vous créez une [compétence personnalisée](#) à l'aide d'Azure Machine Learning pour extraire, à partir des avis, des sentiments basés sur les aspects. L'affectation de sentiments positifs et négatifs au sein d'un même avis est ainsi possible, et une imputation correcte est alors effectuée aux entités identifiées, telles que le personnel, la chambre, la réception ou la piscine.

Pour entraîner le modèle de sentiment basé sur l'aspect dans Azure Machine Learning, vous allez utiliser le [dépôt de recettes nlp](#). Le modèle sera ensuite déployé en tant que point de terminaison sur un cluster Azure Kubernetes. Aussitôt déployé, le point de terminaison est ajouté au pipeline d'enrichissement en tant que compétence Azure Machine Learning à utiliser par le service Recherche cognitive.

Deux jeux de données sont fournis. Si vous souhaitez effectuer l'apprentissage du modèle vous-même, le fichier `hotel_reviews_1000.csv` est demandé. Vous préférez ignorer l'étape d'entraînement ? Téléchargez le fichier `hotel_reviews_100.csv`.

- Créez une instance Recherche cognitive Azure
- Créez un espace de travail Azure Machine Learning (le service de recherche et l'espace de travail doivent se trouver dans le même abonnement)
- Entraînez et déployez un modèle sur un cluster Azure Kubernetes
- Lier un pipeline d'enrichissement par IA au modèle déployé
- Ingérez la sortie du modèle déployé en tant que compétence personnalisée

## IMPORTANT

Cette compétence est actuellement en préversion publique. Les fonctionnalités en préversion sont fournies sans contrat de niveau de service et ne sont pas recommandées pour les charges de travail de production. Pour plus d'informations, consultez [Conditions d'Utilisation Supplémentaires relatives aux Évaluations Microsoft Azure](#). Le SDK .NET n'est actuellement pas pris en charge.

## Prérequis

- Abonnement Azure : procurez-vous un [abonnement gratuit](#).
- [Service Recherche cognitive](#)
- [Ressource Cognitive Services](#)
- [compte Stockage Azure](#)
- [Espace de travail Azure Machine Learning](#)

## Programme d'installation

- Clonez ou téléchargez le contenu de l'[exemple de dépôt](#).
- Procédez à l'extraction du contenu si le téléchargement est un fichier zip. Assurez-vous que les fichiers sont en lecture-écriture.
- Lors de la configuration des comptes et des services Azure, copiez les noms et les clés dans un fichier texte facilement accessible. Les noms et les clés sont ajoutés à la première cellule du notebook dans lequel les

variables d'accès aux services Azure sont définies.

- Si vous n'êtes pas familiarisé avec Azure Machine Learning et ses spécifications, vous pouvez consulter ces documents avant de commencer :
  - [Configurer un environnement de développement pour Azure Machine Learning](#)
  - [Créer et gérer des espaces de travail Azure Machine Learning dans le portail Azure](#)
- Lors de la configuration de l'environnement de développement pour Azure Machine Learning, envisagez l'utilisation de l'[instance de calcul informatique](#) pour accélérer et faciliter la prise en main.
- Chargez le fichier du jeu de données sur un conteneur dans le compte de stockage. Le plus gros fichier est nécessaire si vous souhaitez effectuer l'étape d'entraînement dans le notebook. Si vous préférez ignorer l'étape d'entraînement, le plus petit fichier est recommandé.

## Ouvrir le notebook et se connecter aux services Azure

1. Placez dans la première cellule toutes les informations nécessaires aux variables qui permettront l'accès aux services Azure, puis exécutez la cellule.
2. L'exécution de la deuxième cellule confirmera que vous êtes connecté au service de recherche pour votre abonnement.
3. Les sections 1.1 à 1.5 vont créer le magasin de données, l'ensemble de compétences, l'index et l'indexeur du service de recherche.

À ce stade, vous pouvez choisir d'ignorer l'étape de création du jeu de données d'entraînement et celle d'expérimentation dans Azure Machine Learning pour passer directement à l'enregistrement des deux modèles fournis dans le dossier des modèles du dépôt GitHub. Si vous ignorez ces étapes, dans le notebook, passez à la section 3.5, Écrire un script de scoring. Vous gagnerez du temps, car les étapes de téléchargement et de chargement des données peuvent prendre jusqu'à 30 minutes.

## Création et entraînement des modèles

La section 2 comprend six cellules qui téléchargent le fichier des incorporations de gants à partir du dépôt de recettes nlp. Après le téléchargement, le fichier est ensuite chargé dans le magasin de données Azure Machine Learning. La taille du fichier .zip est de 2 Go environ, et l'exécution de ces tâches peut demander un certain temps. Une fois chargées, les données d'entraînement sont ensuite extraites ; vous êtes prêt désormais pour la section 3.

## Entraîner le modèle de sentiment basé sur l'aspect et déployer votre point de terminaison

La section 3 du notebook entraîne les modèles qui ont été créés à la section 2 ; enregistrez ces modèles et déployez-les comme point de terminaison dans un cluster Azure Kubernetes. Si vous n'êtes pas familiarisé avec Azure Kubernetes, nous vous recommandons vivement de consulter les articles suivants avant d'essayer de créer un cluster d'inférence :

- [Présentation d'Azure Kubernetes Service](#)
- [Concepts de base de Kubernetes pour Azure Kubernetes Service \(AKS\)](#)
- [Quotas, restrictions de taille de machine virtuelle et disponibilité des régions dans Azure Kubernetes Service \(AKS\)](#)

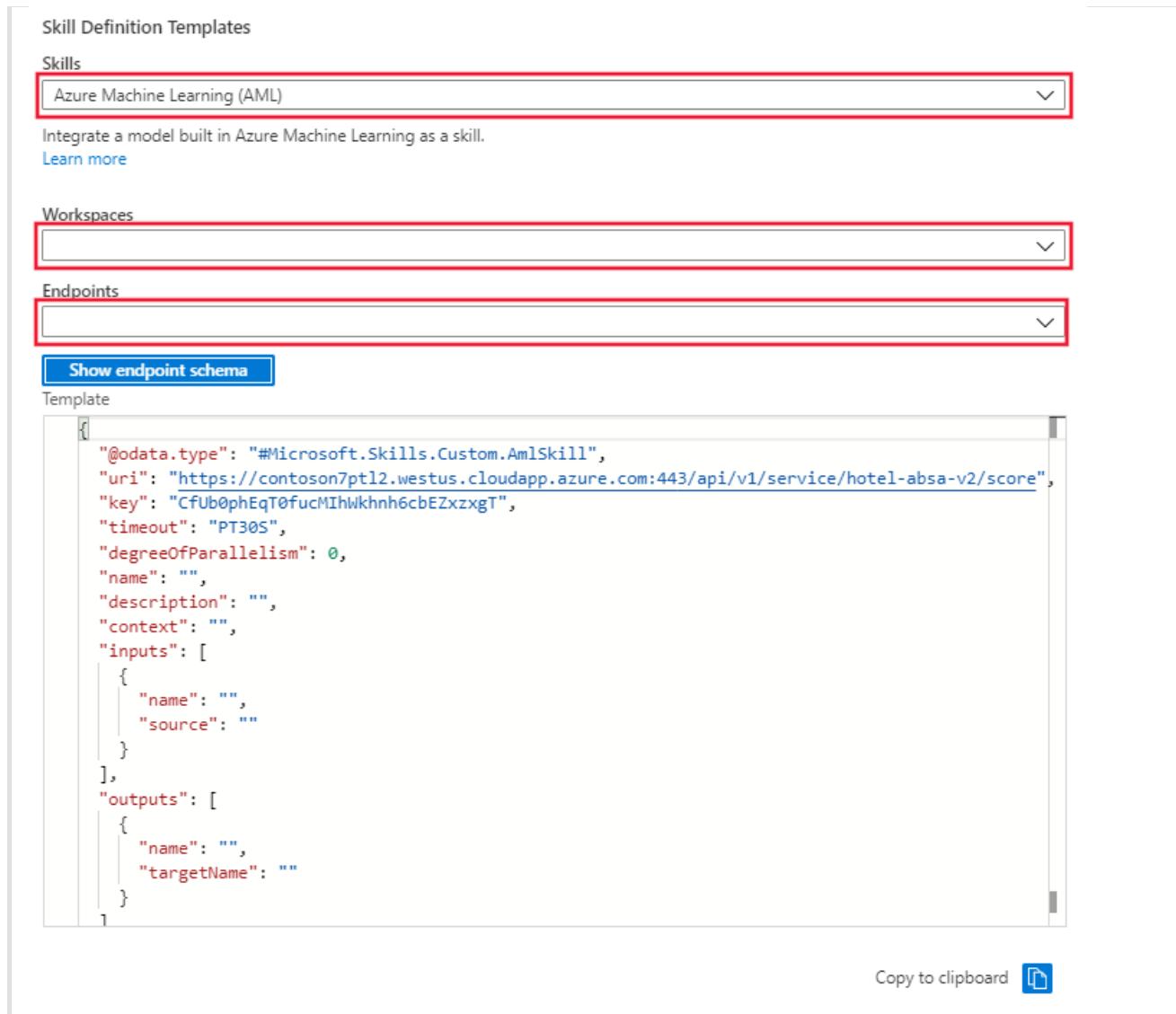
Les opérations de création et de déploiement du cluster d'inférence peuvent prendre jusqu'à 30 minutes. Pour tester le service web avant de passer aux dernières étapes, il est recommandé de mettre à jour votre ensemble de compétences et d'exécuter l'indexeur.

## Mettre à jour l'ensemble de compétences

La section 4 du notebook comporte quatre cellules qui mettent à jour l'ensemble de compétences et l'indexeur.

Vous pouvez également utiliser le portail pour sélectionner et appliquer la nouvelle compétence à l'ensemble de compétences, puis exécuter l'indexeur pour mettre à jour le service de recherche.

Dans le portail, accédez à Ensemble de compétences, puis sélectionnez le lien Définition de l'ensemble de compétences (JSON). Le portail affiche le fichier JSON de l'ensemble de compétences qui a été créé dans les premières cellules du notebook. Dans la partie droite de l'écran, un menu déroulant vous permet de sélectionner le modèle de définition de l'ensemble de compétences. Sélectionnez le modèle Azure Machine Learning (AML). Indiquez le nom de l'espace de travail Azure Machine Learning ainsi que le point de terminaison du modèle déployé sur le cluster d'inférence. Le modèle sera mis à jour avec l'URI et la clé du point de terminaison.



Skill Definition Templates

Skills

Azure Machine Learning (AML)

Integrate a model built in Azure Machine Learning as a skill.  
[Learn more](#)

Workspaces

Endpoints

Show endpoint schema

Template

```
{ "@odata.type": "#Microsoft.Skills.Custom.AmlSkill", "uri": "https://contoson7pt12.westus.cloudapp.azure.com:443/api/v1/service/hotel-absa-v2/score", "key": "CfUb0phEqT0fucMIhWkhnh6cbEZxzxgT", "timeout": "PT30S", "degreeOfParallelism": 0, "name": "", "description": "", "context": "", "inputs": [ { "name": "", "source": "" } ], "outputs": [ { "name": "", "targetName": "" } ] }
```

Copy to clipboard

À partir de la fenêtre, copiez le modèle de l'ensemble de compétences, puis collez-le dans la définition de l'ensemble de compétences, à gauche. Modifiez le modèle afin de préciser les valeurs manquantes pour :

- Nom
- Description
- Context
- Nom et source des « entrées »
- Nom et targetName des « sorties »

Enregistrez l'ensemble de compétences.

Après avoir enregistré l'ensemble de compétences, accédez à l'indexeur et sélectionnez le lien Définition de

l'indexeur (JSON). Le portail affiche le code JSON de l'indexeur créé dans les premières cellules du notebook. Les mappages des champs de sortie devront être mis à jour avec les mappages de champs supplémentaires pour garantir la gestion et le passage correct par l'indexeur. Enregistrez vos modifications, puis sélectionnez Exécuter.

## Nettoyer les ressources

Lorsque vous travaillez dans votre propre abonnement, il est recommandé, à la fin de chaque projet, de déterminer si vous avez toujours besoin des ressources que vous avez créées. Les ressources laissées en cours d'exécution peuvent vous coûter de l'argent. Vous pouvez supprimer les ressources une par une, ou choisir de supprimer le groupe de ressources afin de supprimer l'ensemble des ressources.

Vous pouvez rechercher et gérer les ressources dans le portail à l'aide des liens **Toutes les ressources** ou **Groupes de ressources** situés dans le volet de navigation de gauche.

Si vous utilisez un service gratuit, n'oubliez pas que vous êtes limité à trois index, indexeurs et sources de données. Vous pouvez supprimer des éléments un par un dans le portail pour ne pas dépasser la limite.

## Étapes suivantes

[Consulter l'API web de compétence personnalisée](#) En savoir plus sur l'ajout de compétences personnalisées au pipeline d'enrichissement

# Conseils sur l'enrichissement par IA dans Recherche cognitive Azure

04/10/2020 • 12 minutes to read • [Edit Online](#)

Cet article contient une liste de conseils et astuces destinés à vous permettre d'aller de l'avant lorsque vous commencez à utiliser les fonctionnalités d'enrichissement de l'IA dans la Recherche cognitive Azure.

Si ce n'est déjà fait, suivez le [Tutorial : Appeler des API d'enrichissement de l'IA](#) pour apprendre à appliquer des enrichissements de l'IA à une source de données Blob.

## Conseil 1 : commencez avec un petit jeu de données

La meilleure façon de détecter rapidement des problèmes consiste à augmenter la vitesse à laquelle vous pouvez résoudre ceux-ci. La meilleure façon de réduire le temps d'indexation consiste à diminuer le nombre de documents à indexer.

Commencez par créer une source de données ne contenant que quelques documents/enregistrements. Votre exemple de document doit refléter assez fidèlement la diversité des documents qui seront indexés.

Exécutez votre exemple de document via le pipeline de bout en bout et vérifiez que les résultats correspondent à vos besoins. Lorsque vous êtes satisfait des résultats, vous pouvez ajouter des fichiers à votre source de données.

## Conseil 2 : assurez-vous que les informations d'identification de la source de données sont correctes

La connexion de source de données n'est pas validée tant que vous ne définissez pas un indexeur qui l'utilise. Si vous rencontrez des erreurs signalant que l'indexeur ne peut pas accéder aux données, vérifiez les points suivants :

- Votre chaîne de connexion est correcte. En particulier lorsque vous créez des jetons SAP, veillez à utiliser le format attendu par la Recherche cognitive Azure. Pour découvrir les différents formats pris en charge, voir la section [Comment spécifier des informations d'identification](#).
- Le nom de votre conteneur dans l'indexeur est correct.

## Conseil 3 : regardez ce qui fonctionne même en présence d'erreurs

Parfois, une petite défaillance arrête un indexeur. C'est bien si vous prévoyez de résoudre les problèmes un par un. Cependant, vous pourriez vouloir ignorer un type particulier d'erreur et permettre à l'indexeur de continuer à opérer afin que vous puissiez voir quels flux fonctionnent réellement.

Dans ce cas, il se peut que vous vouliez demander à l'indexeur d'ignorer certaines erreurs. Pour ce faire, définissez les valeurs de *maxFailedItems* et *maxFailedItemsPerBatch* sur -1 dans le cadre de la définition de l'indexeur.

```
{  
    // rest of your indexer definition  
    "parameters":  
    {  
        "maxFailedItems": -1,  
        "maxFailedItemsPerBatch": -1  
    }  
}
```

#### NOTE

Il est recommandé de définir maxFailedItems, maxFailedItemsPerBatch sur 0 pour les charges de travail de production

## Conseil 4 : Utiliser des sessions de débogage pour identifier et résoudre les problèmes liés à vos compétences

Les sessions de débogage consistent en un éditeur visuel qui fonctionne avec un ensemble de compétences existant dans le portail Azure. Au sein d'une session de débogage, vous pouvez identifier et résoudre les erreurs, valider les modifications et les transférer vers un ensemble de compétences de production dans le pipeline d'enrichissement par IA. Il s'agit d'une fonctionnalité d'évaluation [lire la documentation](#). Pour plus d'informations sur les concepts et la mise en route, consultez [Sessions de débogage](#).

Les sessions de débogage qui fonctionnent sur un seul document sont un excellent moyen de créer de manière itérative des pipelines d'enrichissement plus complexes.

## Conseil 5 : examinez les documents enrichis temporaires

Les documents enrichis sont des structures temporaires créées pendant l'enrichissement, puis supprimées une fois le traitement terminé.

Pour capturer un instantané du document enrichi créé lors de l'indexation, ajoutez à votre index un champ nommé `enriched`. L'indexeur vide automatiquement dans ce champ une représentation de chaîne de tous les enrichissements de ce document.

Le champ `enriched` contiendra une chaîne qui est une représentation logique du document enrichi en mémoire dans JSON. Toutefois, la valeur du champ est un document JSON valide. Les guillemets étant échappés, vous devez remplacer `\"` par `"` afin d'afficher le document en tant que JSON formaté.

Le champ enrichi est fourni à des fins de débogage uniquement, pour vous aider à comprendre la forme logique du contenu par rapport auquel les expressions sont évaluées. Vous ne devez pas dépendre ce champ pour l'indexation.

Ajoutez un champ `enriched` dans votre définition d'index à des fins de débogage :

#### Syntaxe du corps de la demande

```
{
  "fields": [
    // other fields go here.
    {
      "name": "enriched",
      "type": "Edm.String",
      "searchable": false,
      "sortable": false,
      "filterable": false,
      "facetable": false
    }
  ]
}
```

## Conseil 6 : le contenu attendu n'apparaît pas

Un contenu manquant peut résulter d'une suppression de documents lors de l'indexation. Les limites de taille de document pour les niveaux Gratuit et De base sont basses. Tout fichier dépassant la limite est écarté lors de l'indexation. Vous pouvez vérifier la présence de documents écartés dans le portail Azure. Dans le tableau de bord

du service de recherche, double-cliquez sur la vignette Indexeurs. Examinez le taux de documents correctement indexés. S'il n'est pas de 100 %, vous pouvez cliquer dessus pour obtenir plus de détails.

Si le problème est lié à la taille du fichier, vous pouvez voir une erreur similaire à celle-ci : « L'objet blob <filename> a une taille de <file-size> octets, ce qui est supérieur à la taille maximale d'extraction de document correspondant à votre niveau de service actuel. » Pour plus d'informations sur les limites de l'indexeur, voir [Limites du service](#).

Un échec d'affichage du contenu peut également résulter d'erreurs de mappage d'entrée/sortie. Par exemple, un nom de cible de sortie est « Contacts », mais le nom du champ d'index est « contacts » en minuscules. Le système peut retourner 201 messages de réussite pour le pipeline entier, de sorte que vous pensez que l'indexation a réussi alors qu'en fait un champ est vide.

## Conseil 7 : prolongez le traitement au-delà du temps d'exécution maximal (fenêtre de 24 heures)

L'analyse d'image nécessite une grande capacité de calcul, même pour des cas simples. Ainsi, quand des images sont particulièrement volumineuses ou complexes, les temps de traitement peuvent dépasser le temps maximal imparti.

Le temps d'exécution maximal varie selon le niveau : de quelques minutes pour le niveau Gratuit, à une durée d'indexation de 24 heures pour les niveaux facturables. Si le traitement n'aboutit pas dans un délai de 24 heures pour un traitement à la demande, passez à une planification telle que l'indexeur reprenne le traitement là où il l'a laissé.

Pour les indexeurs planifiés, l'indexation reprend dans le délai prévu au dernier bon document connu. Avec une planification récurrente, l'indexeur peut opérer à sa manière dans le backlog d'images sur une série d'heures ou de jours, jusqu'à ce que toutes les images non traitées le soient. Pour plus d'informations sur la syntaxe de la planification, consultez [Étape 3 : Créer un indexeur](#) ou consultez [Guide pratique pour planifier des indexeurs pour la Recherche cognitive Azure](#).

### NOTE

Si un indexeur est défini sur une certaine planification, mais échoue à plusieurs reprises sur le même document chaque fois qu'il s'exécute, l'indexeur commence à s'exécuter à un intervalle moins fréquent (jusqu'à un maximum d'au moins une fois toutes les 24 heures) jusqu'à ce qu'il progresse correctement à nouveau. Si vous pensez avoir résolu le problème qui provoquait le blocage de l'indexeur à un moment donné, vous pouvez effectuer une exécution à la demande de l'indexeur, et en cas de progression, l'indexeur reprend son intervalle de planification défini.

Pour une indexation basée sur le portail (telle que décrite dans le démarrage rapide), le choix de l'option d'indexeur « Exécuter une fois » limite le traitement à 1 heure (`"maxRunTime": "PT1H"`). Vous pouvez étendre la fenêtre de traitement.

## Conseil 8 : augmentez le débit d'indexation

Pour une [indexation parallèle](#), placez vos données dans plusieurs conteneurs ou dans plusieurs dossiers virtuels au sein du même conteneur. Créez ensuite plusieurs paires source de données et indexeur. Tous les indexeurs pouvant utiliser le même jeu de compétences et écrire dans le même index de recherche cible, votre application de recherche n'a pas besoin d'être informée de ce partitionnement. Pour plus d'informations, voir [Indexation de jeux de données volumineux](#).

## Voir aussi

- [Démarrage rapide : Créer un pipeline d'enrichissement de l'IA dans le portail](#)

- [Tutoriel : Découvrir les API REST d'enrichissement de l'IA](#)
- [Spécification des informations d'identification de la source de données](#)
- [Indexation de jeux de données volumineux](#)
- [Guide pratique pour définir un ensemble de compétences](#)
- [Guide pratique pour mapper des champs enrichis à un index](#)

# Créer une base de connaissances à l'aide de REST et Postman

04/10/2020 • 20 minutes to read • [Edit Online](#)

Une base de connaissances contient la sortie d'un pipeline d'enrichissement Recherche cognitive Azure pour une analyse ultérieure ou tout autre traitement en aval. Un pipeline enrichi par IA accepte les fichiers image ou les fichiers texte non structurés, les index en utilisant Recherche cognitive Azure, applique des enrichissements par IA provenant de Cognitive Services (par exemple l'analyse d'images et le traitement en langage naturel), puis enregistre les résultats dans une base de connaissances dans Stockage Azure. Vous pouvez utiliser des outils comme Power BI ou l'Explorateur Stockage sur le portail Azure pour explorer la base de connaissances.

Dans cet article, vous allez utiliser l'interface de l'API REST pour ingérer, indexer et appliquer des enrichissements par IA à un ensemble d'avis sur des hôtels. Les avis sur les hôtels sont importés dans le Stockage Blob Azure. Les résultats sont enregistrés sous la forme d'une base de connaissances dans le stockage Table Azure.

Une fois la base de connaissances créée, vous pouvez apprendre à y accéder à l'aide de l'[Explorateur Stockage](#) ou de [Power BI](#).

Si vous n'avez pas d'abonnement Azure, créez un [compte gratuit](#) avant de commencer.

## TIP

Nous vous recommandons l'[application de bureau Postman](#) pour cet article. Le [code source](#) de cet article comprend une collection Postman contenant toutes les requêtes.

## Créer des services et charger des données

Ce guide de démarrage rapide utilise la Recherche cognitive Azure, le Stockage Blob Azure et [Azure Cognitive Services](#) pour l'IA.

En raison de la taille réduite de la charge de travail, Cognitive Services est utilisé en arrière-plan pour traiter gratuitement jusqu'à 20 transactions par jour. Parce que le jeu de données est vraiment petit, vous pouvez ignorer la création ou l'attachement d'une ressource Cognitive Services.

1. [Téléchargez le fichier HotelReviews\\_Free.csv](#). Ce fichier CSV contient des données d'avis d'hôtel (issues de Kaggle.com). Il rassemble 19 commentaires de clients relatifs à un seul hôtel.
2. [Créez un compte de stockage Azure](#) ou [recherchez un compte existant](#) dans votre abonnement actuel. Vous utilisez le stockage Azure pour le contenu brut à importer, mais aussi pour la base de connaissances qui est le résultat final.  
Choisissez le type de compte **StorageV2 (usage général v2)**.
3. Ouvrez les pages des services Blob et créez un conteneur nommé *hotel-reviews*.
4. Cliquez sur **Télécharger**.



5. Sélectionnez le fichier **HotelReviews-Free.csv** que vous avez téléchargé à la première étape.

6. Vous avez presque terminé avec cette ressource, mais avant de quitter ces pages, ouvrez la page **Clés d'accès** à partir du lien correspondant dans le volet de navigation de gauche. Obtenez une chaîne de connexion pour récupérer les données du Stockage Blob. Une chaîne de connexion ressemble à l'exemple suivant :

```
DefaultEndpointsProtocol=https;AccountName=<YOUR-ACCOUNT-NAME>;AccountKey=<YOUR-ACCOUNT-KEY>;EndpointSuffix=core.windows.net
```

7. Toujours dans le portail, basculez vers Recherche cognitive Azure. [Créez un service](#) ou [recherchez un service existant](#). Vous pouvez utiliser un service gratuit pour cet exercice.

## Configurer Postman

Installez et configurez Postman.

### Télécharger et installer Postman

1. Téléchargez le [code source de la collection Postman](#).
2. Sélectionnez **Fichier > Importer** pour importer le code source dans Postman.
3. Sélectionnez l'onglet **Collections**, puis le bouton ... (points de suspension).
4. Sélectionnez **Modifier**.

5. Dans la boîte de dialogue **Edit**, sélectionnez l'onglet **Variables**.

Sous l'onglet **Variables**, vous pouvez ajouter les valeurs que Postman récupère chaque fois qu'il trouve une

variable spécifique entre double accolades. Par exemple, Postman remplace le symbole `{{admin-key}}` par la valeur actuelle que vous avez définie pour `admin-key`. Postman effectue la substitution dans les URL, les en-têtes, le corps de la requête, etc.

Pour obtenir la valeur de `admin-key`, accédez au service Recherche cognitive Azure et sélectionnez l'onglet **Clés**. Remplacez les valeurs de `search-service-name` et `storage-account-name` par celles que vous avez choisies dans [Créer des services](#). Définissez `storage-connection-string` en utilisant la valeur figurant sous l'onglet **Clés d'accès** du compte de stockage. Partout ailleurs, vous pouvez conserver les valeurs par défaut.

EDIT COLLECTION X

Name  
Create a KnowledgeStore

Description    Authorization    Pre-request Scripts    Tests    **Variables** ●

These variables are specific to this collection and its requests. [Learn more about collection variables](#).

	VARIABLE	INITIAL VALUE ⓘ	CURRENT VALUE ⓘ	...	Persist All	Reset All
<input checked="" type="checkbox"/>	admin-key	<SEARCH_SERVICE_ADMIN_KEY>	[redacted]			
<input checked="" type="checkbox"/>	search-service-name	<SEARCH_SERVICE_NAME>	[redacted]			
<input checked="" type="checkbox"/>	storage-account-name	<STORAGE_ACCOUNT_NAME>	[redacted]			
<input checked="" type="checkbox"/>	storage-connection-string	<STORAGE_ACCOUNT_CONNEC...	[redacted]			
<input checked="" type="checkbox"/>	api-version	2019-05-06-Preview	2019-05-06-Preview			
<input checked="" type="checkbox"/>	datasource-name	hotel-reviews-ds	hotel-reviews-ds			
<input checked="" type="checkbox"/>	indexer-name	hotel-reviews-ixr	hotel-reviews-ixr			
<input checked="" type="checkbox"/>	index-name	hotel-reviews-ix	hotel-reviews-ix			
<input checked="" type="checkbox"/>	skillset-name	hotel-reviews-ss	hotel-reviews-ss			
<input checked="" type="checkbox"/>	storage-container-name	hotel-reviews	hotel-reviews			
	Add a new variable					

ⓘ Use variables to reuse values in different places. The current value is used while sending a request and is never synced to Postman's servers. The initial value is auto-updated to reflect the current value. [Change this](#) behaviour from Settings. X

[Learn more about variable values](#)

Cancel    Update

VARIABLE	COMMENT LES OBTENIR
admin-key	Dans la page <b>Clés</b> du service Recherche cognitive Azure.
api-version	Laisser <b>2020-06-30</b> .
datasource-name	Laisser <b>hotel-reviews-ds</b> .

VARIABLE	COMMENT LES OBTENIR
indexer-name	Laisser hotel-reviews-ixr.
index-name	Laisser hotel-reviews-ix.
search-service-name	Nom du service Recherche cognitive Azure. L'URL est <code>https://{{search-service-name}}.search.windows.net</code> .
skillset-name	Laisser hotel-reviews-ss.
storage-account-name	nom du compte de stockage.
storage-connection-string	Dans le compte de stockage, sous l'onglet Clés d'accès, sélectionnez key1 > Chaîne de connexion.
storage-container-name	Laisser hotel-reviews.

### Passer en revue la collection de requêtes dans Postman

Quand vous créez une base de connaissances, vous devez émettre quatre requêtes HTTP :

- **Une requête PUT pour créer l'index** : Cet index contient les données utilisées et retournées par Recherche cognitive Azure.
- **Une requête POST pour créer la source de données** : Cette source de données connecte le comportement du service Recherche cognitive Azure au compte de stockage des données et de la base de connaissances.
- **Une requête PUT pour créer l'ensemble de compétences** : l'ensemble de compétences spécifie les enrichissements appliqués à vos données et à la structure de la base de connaissances.
- **Une requête PUT pour créer l'indexeur** : L'exécution de l'indexeur lit les données, applique l'ensemble de compétences et stocke les résultats. Vous devez exécuter cette requête en dernier.

Le [code source](#) contient une collection Postman avec ces quatre requêtes. Pour émettre les requêtes, dans Postman, sélectionnez l'onglet correspondant à la requête. Ajoutez ensuite les en-têtes de requête `api-key` et `Content-Type`. Affectez à `api-key` la valeur `{{admin-key}}`. Affectez à `Content-type` la valeur `application/json`.

The screenshot shows the Postman interface with a collection named "Create Hotel Reviews". It contains four requests:

- Create Index**: A PUT request to `https://{{search-service-name}}.search.windows.net/indexes/{{index-name}}?api-version={{api-version}}`. Headers include `api-key: {{admin-key}}` and `Content-Type: application/json`.
- Create Datasource**: A POST request to `https://{{search-service-name}}.search.windows.net/datasources?api-version={{api-version}}`. Headers include `Content-Type: application/json`.
- Create the Skillset**: A PUT request to `https://{{search-service-name}}.search.windows.net/skillsets/{{skillset-name}}?api-version={{api-version}}`. Headers include `Content-Type: application/json`.
- Create the Indexer**: A PUT request to `https://{{search-service-name}}.search.windows.net/indexers/{{indexer-name}}?api-version={{api-version}}`. Headers include `Content-Type: application/json`.

#### NOTE

Vous devez définir les en-têtes `api-key` et `Content-type` dans toutes vos requêtes. Si Postman reconnaît une variable, celle-ci se apparaît en orange, comme pour `{{admin-key}}` dans la capture d'écran précédente. Si la variable est mal orthographiée, elle apparaît en rouge.

## Création d'un index Recherche cognitive Azure

Créez un index Recherche cognitive Azure pour représenter les données sur lesquelles vous souhaitez effectuer des recherches et des filtrages, et appliquer des améliorations. Créez l'index en émettant une requête PUT vers `https://{{search-service-name}}.search.windows.net/indexes/{{index-name}}?api-version={{api-version}}`.

Postman remplace les symboles placés entre deux accolades (par exemple, `{{search-service-name}}`, `{{index-name}}` et `{{api-version}}`) par les valeurs que vous avez définies dans [Configurer Postman](#). Si vous utilisez un autre outil pour émettre vos commandes REST, vous devez vous-même remplacer ces variables.

Définissez la structure de votre index Recherche cognitive Azure dans le corps de la requête. Dans Postman, après avoir défini les en-têtes `api-key` et `Content-type`, accédez au volet **Body** (Corps) de la requête. Le code JSON suivant doit s'afficher. Si ce n'est pas le cas, sélectionnez **Raw > JSON (application/json)**, puis collez le code suivant dans le corps :

```

{
    "name": "{{index-name}}",
    "fields": [
        { "name": "address", "type": "Edm.String", "searchable": false, "filterable": false, "sortable": false, "facetable": false },
        { "name": "categories", "type": "Edm.String", "searchable": false, "filterable": false, "sortable": false, "facetable": false },
        { "name": "city", "type": "Edm.String", "filterable": false, "sortable": false, "facetable": false },
    ],
    { "name": "country", "type": "Edm.String", "searchable": false, "filterable": false, "sortable": false, "facetable": false },
    { "name": "latitude", "type": "Edm.String", "searchable": false, "filterable": false, "sortable": false, "facetable": false },
    { "name": "longitude", "type": "Edm.String", "searchable": false, "filterable": false, "sortable": false, "facetable": false },
    { "name": "name", "type": "Edm.String", "filterable": false, "sortable": false, "facetable": false },
},
{ "name": "postalCode", "type": "Edm.String", "searchable": false, "filterable": false, "sortable": false, "facetable": false },
{ "name": "province", "type": "Edm.String", "searchable": false, "filterable": false, "sortable": false, "facetable": false },
{ "name": "reviews_date", "type": "Edm.DateTimeOffset", "searchable": false, "filterable": false, "sortable": false, "facetable": false },
{ "name": "reviews_dateAdded", "type": "Edm.DateTimeOffset", "searchable": false, "filterable": false, "sortable": false, "facetable": false },
{ "name": "reviews_rating", "type": "Edm.String", "searchable": false, "filterable": false, "sortable": false, "facetable": false },
{ "name": "reviews_text", "type": "Edm.String", "filterable": false, "sortable": false, "facetable": false },
{ "name": "reviews_title", "type": "Edm.String", "searchable": false, "filterable": false, "sortable": false, "facetable": false },
{ "name": "reviews_username", "type": "Edm.String", "searchable": false, "filterable": false, "sortable": false, "facetable": false },
{ "name": "AzureSearch_DocumentKey", "type": "Edm.String", "searchable": false, "filterable": false, "sortable": false, "facetable": false, "key": true },
{ "name": "metadata_storage_content_type", "type": "Edm.String", "searchable": false, "filterable": false, "sortable": false, "facetable": false },
{ "name": "metadata_storage_size", "type": "Edm.Int64", "searchable": false, "filterable": false, "sortable": false, "facetable": false },
{ "name": "metadata_storage_last_modified", "type": "Edm.DateTimeOffset", "searchable": false, "filterable": false, "sortable": false, "facetable": false },
{ "name": "metadata_storage_name", "type": "Edm.String", "searchable": false, "filterable": false, "sortable": false, "facetable": false },
{ "name": "metadata_storage_path", "type": "Edm.String", "searchable": false, "filterable": false, "sortable": false, "facetable": false },
{ "name": "Sentiment", "type": "Collection(Edm.Double)", "searchable": false, "filterable": true, "retrievable": true, "sortable": false, "facetable": true },
{ "name": "Language", "type": "Edm.String", "filterable": true, "sortable": false, "facetable": true },
{ "name": "Keyphrases", "type": "Collection(Edm.String)", "filterable": true, "sortable": false, "facetable": true }
]
}

```

Cette définition d'index est une combinaison de données que vous souhaitez présenter à l'utilisateur (nom de l'hôtel, contenu de l'avis, date), de métadonnées de recherche et de données d'amélioration de l'IA (sentiment, phrases clés et langue).

Sélectionnez **Send** (Envoyer) pour émettre la requête PUT. L'état `201 - Created` doit alors s'afficher. Si vous un autre état s'affiche, dans le volet **Body**, recherchez une réponse JSON qui contient un message d'erreur.

## Créer la source de données

Ensuite, connectez Recherche cognitive Azure aux données relatives aux hôtels que vous avez stockées dans le stockage Blob. Pour créer la source de données, envoyez une requête POST à

`https://{{search-service-name}}.search.windows.net/datasources?api-version={{api-version}}`. Vous devez définir les en-têtes `api-key` et `Content-Type` comme indiqué précédemment.

Dans Postman, accédez à la requête **Create Datasource** (Créer une source de données), puis au volet **Body** (Corps). Le code suivant doit s'afficher :

```
{  
    "name" : "{{datasource-name}}",  
    "description" : "Demo files to demonstrate knowledge store capabilities.",  
    "type" : "azureblob",  
    "credentials" : { "connectionString" : "{{storage-connection-string}}" },  
    "container" : { "name" : "{{storage-container-name}}" }  
}
```

Sélectionnez **Send** pour émettre la requête POST.

## Créer l'ensemble de compétences

L'étape suivante consiste à spécifier l'ensemble de compétences, qui indique à la fois les améliorations à appliquer et la base de connaissances où sont stockés les résultats. Dans Postman, sélectionnez l'onglet **Create the Skillset** (Créer l'ensemble de compétences). Cette requête envoie un PUT à

`https://{{search-service-name}}.search.windows.net/skillsets/{{skillset-name}}?api-version={{api-version}}`. Définissez les en-têtes `api-key` et `Content-type` comme vous l'avez fait précédemment.

Il existe deux grands objets de niveau supérieur : `skills` et `knowledgeStore`. Chaque objet contenu dans l'objet `skills` est un service d'enrichissement. Chaque service d'enrichissement comporte `inputs` et `outputs`. `LanguageDetectionSkill` présente une sortie `targetName` qui correspond à `Language`. La valeur de ce nœud est utilisée par la plupart des autres compétences comme entrée. La source est `document/Language`. La possibilité d'utiliser la sortie d'un nœud comme entrée d'un autre nœud est encore plus évidente dans `ShaperSkill`, qui spécifie les flux de données dans les tables de la base de connaissances.

L'objet `knowledge_store` se connecte au compte de stockage via la variable Postman `{{storage-connection-string}}`. `knowledge_store` contient un ensemble de mappages entre le document amélioré et les tables et colonnes de la base de connaissances.

Pour générer l'ensemble de compétences, sélectionnez le bouton **Send** dans Postman pour effectuer un PUT de la requête :

```
{  
    "name": "{{skillset-name}}",  
    "description": "Skillset to detect language, extract key phrases, and detect sentiment",  
    "skills": [  
        {  
            "@odata.type": "#Microsoft.Skills.Text.SplitSkill",  
            "context": "/document/reviews_text", "textSplitMode": "pages", "maximumPageLength": 5000,  
            "inputs": [  
                { "name": "text", "source": "/document/reviews_text" },  
                { "name": "languageCode", "source": "/document/Language" }  
            ],  
            "outputs": [  
                { "name": "textItems", "targetName": "pages" }  
            ]  
        },  
        {  
            "@odata.type": "#Microsoft.Skills.Text.SentimentSkill",  
            "context": "/document/reviews_text/pages/*",  
            "inputs": [  
            ]  
        }  
    ]  
}
```

```

        {
            "name": "text", "source": "/document/reviews_text/pages/*" },
            { "name": "languageCode", "source": "/document/Language" }
        ],
        "outputs": [
            { "name": "score", "targetName": "Sentiment" }
        ]
    },
    {
        "@odata.type": "#Microsoft.Skills.Text.LanguageDetectionSkill",
        "context": "/document",
        "inputs": [
            { "name": "text", "source": "/document/reviews_text" }
        ],
        "outputs": [
            { "name": "languageCode", "targetName": "Language" }
        ]
    },
    {
        "@odata.type": "#Microsoft.Skills.Text.KeyPhraseExtractionSkill",
        "context": "/document/reviews_text/pages/*",
        "inputs": [
            { "name": "text", "source": "/document/reviews_text/pages/*" },
            { "name": "languageCode", "source": "/document/Language" }
        ],
        "outputs": [
            { "name": "keyPhrases", "targetName": "Keyphrases" }
        ]
    },
    {
        "@odata.type": "#Microsoft.Skills.Util.ShaperSkill",
        "context": "/document",
        "inputs": [
            { "name": "name", "source": "/document/name" },
            { "name": "reviews_date", "source": "/document/reviews_date" },
            { "name": "reviews_rating", "source": "/document/reviews_rating" },
            { "name": "reviews_text", "source": "/document/reviews_text" },
            { "name": "reviews_title", "source": "/document/reviews_title" },
            { "name": "AzureSearch_DocumentKey", "source": "/document/AzureSearch_DocumentKey" },
            {
                "name": "pages",
                "sourceContext": "/document/reviews_text/pages/*",
                "inputs": [
                    { "name": "SentimentScore", "source": "/document/reviews_text/pages/*/Sentiment"
                },
                { "name": "LanguageCode", "source": "/document/Language" },
                { "name": "Page", "source": "/document/reviews_text/pages/*" },
                {
                    "name": "keyphrase", "sourceContext": "/document/reviews_text/pages/*/Keyphrases/*",
                    "inputs": [
                        { "name": "Keyphrases", "source": "/document/reviews_text/pages/*/Keyphrases/*"
                    ]
                }
            ]
        ],
        "outputs": [
            { "name": "output", "targetName": "tableprojection" }
        ]
    },
    "knowledgeStore": {
        "storageConnectionString": "{{storage-connection-string}}",
        "projections": [
            {
                "tables": [
                    { "tableName": "hotelReviewsDocument", "generatedKeyName": "Documentid", "source": "/document/tableprojection" },

```

```

        {
          "tableName": "hotelReviewsPages", "generatedKeyName": "Pagesid", "source": "/document/tableprojection/pages/*" },
          {
            "tableName": "hotelReviewsKeyPhrases", "generatedKeyName": "KeyPhrasesid", "source": "/document/tableprojection/pages/*/keyphrase/*" },
            {
              "tableName": "hotelReviewsSentiment", "generatedKeyName": "Sentimentid", "source": "/document/tableprojection/pages/*/sentiment/*" }
            ],
            "objects": []
          },
          {
            "tables": [
              {
                "tableName": "hotelReviewsInlineDocument", "generatedKeyName": "Documentid",
                "sourceContext": "/document",
                "inputs": [
                  { "name": "name", "source": "/document/name"}, { "name": "reviews_date", "source": "/document/reviews_date"}, { "name": "reviews_rating", "source": "/document/reviews_rating"}, { "name": "reviews_text", "source": "/document/reviews_text"}, { "name": "reviews_title", "source": "/document/reviews_title"}, { "name": "AzureSearch_DocumentKey", "source": "/document/AzureSearch_DocumentKey" }
                ]
              },
              {
                "tableName": "hotelReviewsInlinePages", "generatedKeyName": "Pagesid",
                "sourceContext": "/document/reviews_text/pages/*",
                "inputs": [
                  { "name": "SentimentScore", "source": "/document/reviews_text/pages/*/Sentiment"}, { "name": "LanguageCode", "source": "/document/Language"}, { "name": "Page", "source": "/document/reviews_text/pages/*" }
                ]
              },
              {
                "tableName": "hotelReviewsInlineKeyPhrases", "generatedKeyName": "kpidv2",
                "sourceContext": "/document/reviews_text/pages/*/Keyphrases/*",
                "inputs": [
                  { "name": "Keyphrases", "source": "/document/reviews_text/pages/*/Keyphrases/*" }
                ]
              },
              "objects": []
            }
          ]
        }
      }
    }
  }
}

```

## Créer l'indexeur

La dernière étape consiste à créer l'indexeur. L'indexeur lit les données et active l'ensemble de compétences. Dans Postman, sélectionnez la requête **Create Indexer** (Créer l'indexeur), puis examinez le corps. La définition de l'indexeur fait référence à plusieurs autres ressources que vous avez déjà créées : la source de données, l'index et l'ensemble de compétences.

L'objet `parameters/configuration` contrôle la manière dont l'indexeur ingère les données. Dans ce cas, les données d'entrée se trouvent dans un même document qui comporte une ligne d'en-tête et des valeurs séparées par des virgules. La clé du document est un identificateur unique du document. Avant l'encodage, la clé du document est l'URL du document source. Enfin, les valeurs de sortie de l'ensemble de l'ensemble de compétences, comme le code de langue, le sentiment et les phrases clés, sont mappées à leurs emplacements dans le document. Bien qu'il existe une seule valeur pour `Language`, `Sentiment` est appliqué à chaque élément du tableau de `pages`. `Keyphrases` est un tableau qui est aussi appliqué à chaque élément du tableau `pages`.

Une fois que vous avez défini les en-têtes `api-key` et `Content-type` et vérifié que le corps de la requête est similaire au code source suivant, sélectionnez **Send** dans Postman. Postman envoie une requête PUT à `https://{{search-service-name}}.search.windows.net/indexers/{{indexer-name}}?api-version={{api-version}}`. Recherche cognitive Azure crée et exécute l'indexeur.

```
{  
    "name": "{{indexer-name}}",  
    "dataSourceName": "{{datasource-name}}",  
    "skillsetName": "{{skillset-name}}",  
    "targetIndexName": "{{index-name}}",  
    "parameters": {  
        "configuration": {  
            "dataToExtract": "contentAndMetadata",  
            "parsingMode": "delimitedText",  
            "firstLineContainsHeaders": true,  
            "delimitedTextDelimiter": ","  
        }  
    },  
    "fieldMappings": [  
        {  
            "sourceFieldName": "AzureSearch_DocumentKey",  
            "targetFieldName": "AzureSearch_DocumentKey",  
            "mappingFunction": { "name": "base64Encode" }  
        }  
    ],  
    "outputFieldMappings": [  
        { "sourceFieldName": "/document/reviews_text/pages/*/Keyphrases/*", "targetFieldName": "Keyphrases" },  
        { "sourceFieldName": "/document/Language", "targetFieldName": "Language" },  
        { "sourceFieldName": "/document/reviews_text/pages/*/Sentiment", "targetFieldName": "Sentiment" }  
    ]  
}
```

## Exécuter l'indexeur

Dans le portail Azure, accédez à la page **Vue d'ensemble** du service Recherche cognitive Azure. Sélectionnez l'onglet **Indexeurs**, puis **hotels-reviews-ixr**. Si l'indexeur n'a pas déjà été exécuté, sélectionnez **Exécuter**. La tâche d'indexation peut déclencher des avertissements liés à la reconnaissance de la langue. Les données incluent des avis rédigés dans des langues qui ne sont pas encore prises en charge par les compétences cognitives.

## Étapes suivantes

Une fois que vous avez enrichi vos données à l'aide de Cognitive Services et que vous avez projeté les résultats dans une base de connaissances, vous pouvez utiliser l'Explorateur Stockage ou Power BI pour explorer votre jeu de données enrichi.

Pour savoir comment explorer cette base de connaissances à l'aide de l'Explorateur Stockage, consultez cette procédure pas à pas :

[Voir avec l'Explorateur Stockage](#)

Pour savoir comment connecter cette base de connaissances à Power BI, consultez cette procédure pas à pas :

[Connexion avec Power BI](#)

Si vous souhaitez répéter cet exercice ou essayer une autre procédure pas à pas d'enrichissement par IA, supprimez l'indexeur **hotel-reviews-idxr**. La suppression de l'indexeur remet le compteur de transactions quotidiennes gratuites à zéro.

## Voir une base de connaissances avec l'Explorateur Stockage

04/10/2020 • 3 minutes to read • [Edit Online](#)

Cet article explique, par exemple, comment se connecter à une base de connaissances et l'explorer à l'aide de l'Explorateur Stockage dans le portail Azure.

# Prérequis

- Suivez les étapes décrites dans [Créer une base de connaissances dans le portail Azure](#) pour créer l'exemple de base de connaissances utilisé dans cette procédure pas à pas.
  - Vous aurez également besoin du nom du compte de stockage Azure utilisé pour créer la base de connaissances ainsi que de sa clé d'accès dans le portail Azure.

## Visualiser, modifier et interroger une base de connaissances dans l'Explorateur Stockage

1. Dans le portail Azure, [ouvrez le compte de stockage](#) que vous avez utilisé pour créer la base de connaissances.
  2. Dans le volet de navigation de gauche du compte de stockage, cliquez sur **Explorateur Stockage**.
  3. Développez la liste TABLES pour afficher la liste des projections de tables Azure créées durant l'exécution de l'Assistant **Importation des données** sur les exemples de données d'avis des clients concernant les hôtels.

Sélectionnez une table pour voir les données enrichies, notamment les phrases clés et les scores de sentiment.

Pour changer le type de données d'une valeur de table ou pour changer des valeurs individuelles dans votre table, cliquez sur **Modifier**. Quand vous changez le type de données d'une colonne dans une ligne de table, il s'applique à toutes les lignes.

				Query	Add	Edit	More
ADDRESS		CATEGORIES	CITY				
ijc3Y7...	Riviera San Nicol 11/a	Hotels	Mableton				
ijc3Y7...	Riviera San Nicol 11/a	Hotels	Mableton				

### Edit Entity

Property Name	Type	Value	
PartitionKey	String	aHR0cHM6Ly9saXNhbGVpYnNhM	
RowKey	String	aHR0cHM6Ly9saXNhbGVpYnNhM	
Timestamp	DateTime	2019-09-09T07:14:17.3880182Z	
AzureSearch_DocumentKey	String	aHR0cHM6Ly9saXNhbGVpYnNhM	
Documentid	String	aHR0cHM6Ly9saXNhbGVpYnNhM	
address	String	Riviera San Nicol 11/a	
categories	String	Hotels	
city	String	Mableton	
country	String	US	
latitude	String	45.421611	
longitude	String	12.376187	

Pour exécuter des requêtes, cliquez sur Requête dans la barre de commandes, puis entrez vos conditions.

The screenshot shows the Power BI Query Editor interface. At the top, there's a toolbar with 'Query' (highlighted with a red box), 'Add', 'Edit', and 'More'. Below the toolbar is a table with three columns: 'ADDRESS', 'CATEGORIES', and 'CITY'. Two rows of data are visible: 'Riviera San Nicol 11/a' under ADDRESS, 'Hotels' under CATEGORIES, and 'Mableton' under CITY. A red arrow points from the 'Query' button in the top bar down to the query editor area.

**Query Editor:**

- Toolbar:** Close Query, Add, Edit, Select All, More.
- Query Structure:**
  - And/Or clause: reviews\_rating String >= 4
  - And clause: reviews\_username String = Julie
  - + Add new clause
- Advanced Options:** dropdown menu
- Table:**| ALCODE | PROVINCE | REVIEWS\_DATE | REVIEWS\_DATEADDED | REVIEWS\_RATING | REVIEWS\_TEXT |
| --- | --- | --- | --- | --- | --- |
| GA |  | 2013-10-27T00:00:00Z | 2016-10-24T00:00:25Z | 5 | We stayed h |

## Nettoyer

Lorsque vous travaillez dans votre propre abonnement, il est recommandé, à la fin de chaque projet, de déterminer si vous avez toujours besoin des ressources que vous avez créées. Les ressources laissées en cours d'exécution peuvent vous coûter de l'argent. Vous pouvez supprimer les ressources une par une, ou choisir de supprimer le groupe de ressources afin de supprimer l'ensemble des ressources.

Vous pouvez rechercher et gérer les ressources dans le portail à l'aide des liens **Toutes les ressources** ou **Groupes de ressources** situés dans le volet de navigation de gauche.

Si vous utilisez un service gratuit, n'oubliez pas que vous êtes limité à trois index, indexeurs et sources de données. Vous pouvez supprimer des éléments un par un dans le portail pour ne pas dépasser la limite.

## Étapes suivantes

Connectez cette base de connaissances à Power BI pour effectuer une analyse plus approfondie, ou allez plus loin avec le code en utilisant l'API REST et Postman pour créer une autre base de connaissances.

[Se connecter avec Power BI](#) [Créer une base de connaissances avec REST](#)

# Connecter une base de connaissances à Power BI

04/10/2020 • 7 minutes to read • [Edit Online](#)

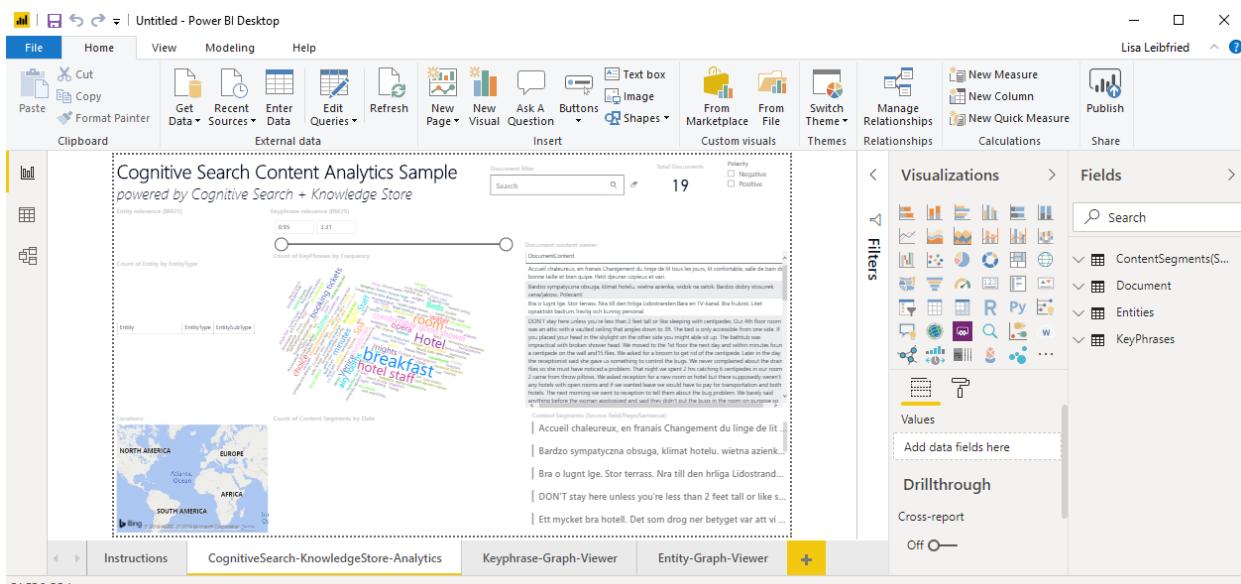
Cet article explique comment connecter et explorer une base de connaissances à l'aide de Power Query dans l'application Power BI Desktop. Vous pouvez commencer plus rapidement avec des modèles, ou créer un tableau de bord personnalisé à partir de rien. Cette courte vidéo ci-dessous montre comment vous pouvez enrichir votre expérience de vos données à l'aide de Recherche cognitive Azure en association avec Power BI.

- Suivez les étapes décrites dans [Créer une base de connaissances dans le portail Azure](#) ou [Créer une base de connaissances Recherche cognitive Azure en utilisant REST](#) pour créer l'exemple de base de connaissances utilisé dans cette procédure pas à pas. Vous aurez également besoin du nom du compte de stockage Azure utilisé pour créer la base de connaissances ainsi que de sa clé d'accès dans le portail Azure.
  - [Installer Power BI Desktop](#)

Exemple de modèle Power BI - Portail Azure uniquement

Lorsque vous créez une [base de connaissances à l'aide du portail Azure](#), vous avez la possibilité de télécharger un [modèle Power BI](#) dans la deuxième page de l'Assistant **Importer des données**. Ce modèle propose plusieurs visualisations, comme WordCloud et Network Navigator, pour le contenu textuel.

Cliquez sur **Obtenir le modèle Power BI** dans la page **Ajouter des compétences cognitives** pour récupérer et télécharger le modèle à partir de son emplacement GitHub public. L'Assistant modifie le modèle pour qu'il s'adapte à la forme de vos données, telles qu'elles sont capturées dans les projections de la base de connaissances spécifiées dans l'Assistant. C'est la raison pour laquelle le modèle que vous téléchargez varie chaque fois que vous exécutez l'Assistant, conformément aux entrées de données et sélections de compétences différentes.



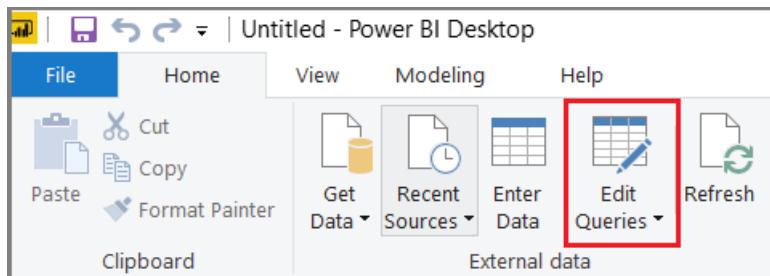
## NOTE

Bien que le modèle soit téléchargé à mi-parcours de l'Assistant, vous devez attendre que la base de connaissances soit réellement créée dans le stockage Table Azure avant de pouvoir l'utiliser.

# Se connecter avec Power BI

1. Démarrez Power BI Desktop, puis cliquez sur **Obtenir les données**.
2. Dans la fenêtre **Obtenir les données**, sélectionnez **Azure**, puis **Stockage Table Azure**.
3. Cliquez sur **Connecter**.
4. Pour **Nom du compte ou URL**, entrez le nom de votre compte Stockage Azure (l'URL complète est créée automatiquement).
5. Si vous y êtes invité, entrez la clé du compte de stockage.
6. Sélectionnez les tables contenant les données d'évaluation des hôtels créées au long des procédures pas à pas précédentes.
  - Pour la procédure pas à pas du portail, les noms de table sont *hotelReviewsSsDocument*, *hotelReviewsSsEntities*, *hotelReviewsSsKeyPhrases* et *hotelReviewsSsPages*.
  - Pour la procédure pas à pas REST, les noms de table sont *hotelReviewsDocument*, *hotelReviewsPages*, *hotelReviewsKeyPhrases* et *hotelReviewsSentiment*.
7. Cliquez sur **Charger**.

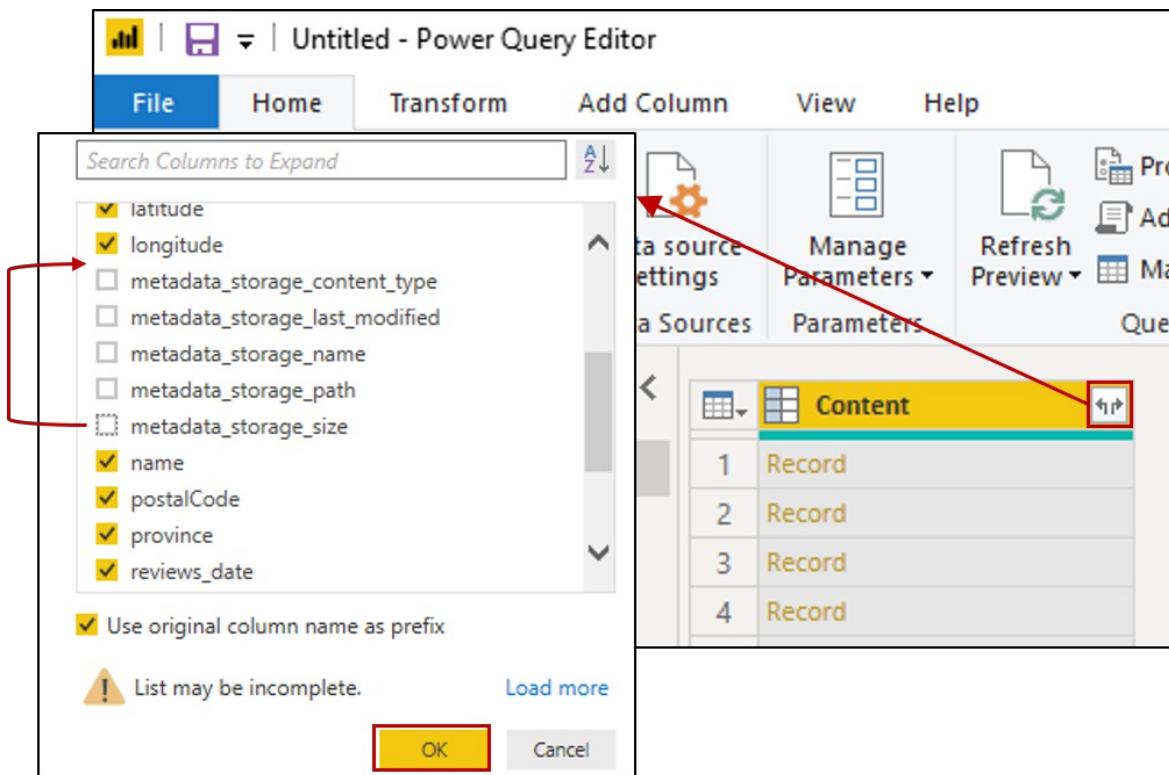
8. Dans le ruban supérieur, cliquez sur **Modifier les requêtes** pour ouvrir l'**Éditeur Power Query**.



9. Sélectionnez *hotelReviewsSsDocument*, puis supprimez les colonnes *PartitionKey*, *RowKey* et *Timestamp*.

A screenshot of the Power Query Editor window. On the left, the 'Queries [3]' pane shows three entries: 'hotelReviewsSsDocument', 'hotelReviewsSsKeyPhrases', and 'hotelReviewsSsPages'. The main area displays a table with three columns: 'PartitionKey', 'RowKey', and 'Timestamp'. Each row contains a unique identifier starting with 'ahR0chM6Ly9saXNhbGVpYn...'. Three large red 'X' marks are drawn over these three columns. The 'Transform' ribbon tab is selected at the top. On the right, the 'Query Settings' pane shows the 'Name' field set to 'hotelReviewsSsDocument'. The 'APPLIED STEPS' pane shows a single step named 'Navigation'.

10. En haut à droite de la table, cliquez sur l'icône montrant des flèches opposées pour développer le *Contenu*. Une fois que la liste des colonnes s'affiche, sélectionnez toutes les colonnes, puis désélectionnez les colonnes commençant par « métadonnées ». Cliquez sur **OK** pour afficher les colonnes sélectionnées.



11. Changez le type de données des colonnes suivantes en cliquant en haut à gauche de la colonne sur l'icône ABC-123.

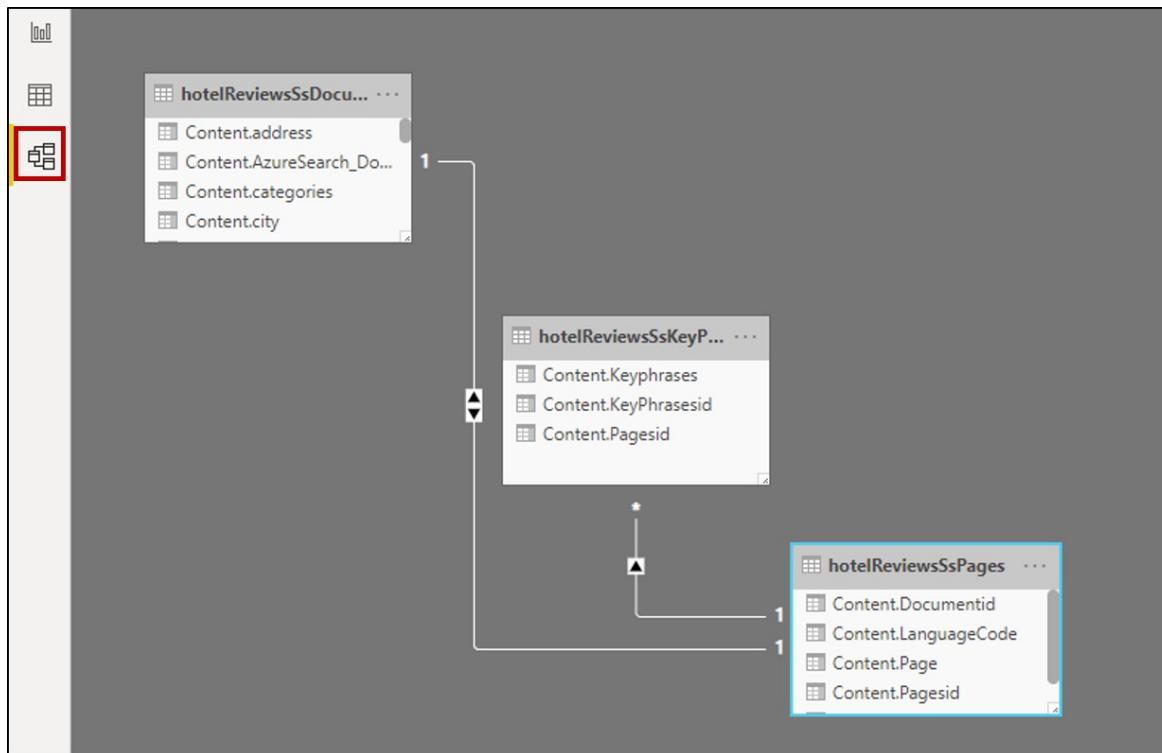
- Pour *Content.latitude* et *Content.longitude*, sélectionnez Nombre décimal.
- Pour *Content.reviews\_date* et *Content.reviews\_dateAdded*, sélectionnez Date/Heure.

1	1.2	Decimal Number	5187
2	\$	Fixed decimal number	5187
3	1 <sup>2</sup> 3	Whole Number	5187
4	%	Percentage	5187
5		Date/Time	5187
6		Date	5187
7			5187

12. Sélectionnez *hotelReviewsSsPages*, puis répétez les étapes 9 et 10 pour supprimer les colonnes et développer le *Contenu*.

13. Remplacez le type de données de *ContentSentimentScore* par Nombre décimal.

14. Sélectionnez *hotelReviewsSsKeyPhrases*, puis répétez les étapes 9 et 10 pour supprimer les colonnes et développer le *Contenu*. Il n'y a aucune modification de type de données pour cette table.
15. Dans la barre de commandes, cliquez sur **Fermer et appliquer**.
16. Cliquez sur la vignette Modèle dans le volet de navigation de gauche, puis vérifiez que Power BI montre les relations entre les trois tables.



17. Double-cliquez sur chaque relation, puis vérifiez que l'option **Direction du filtre croisé** a la valeur **Les deux**. Cela permet d'actualiser les visuels quand un filtre est appliqué.
18. Cliquez sur la vignette Rapport dans le volet de navigation gauche pour explorer les données dans des visualisations. Pour les champs de texte, les tables et les cartes s'avèrent être des visualisations utiles. Vous pouvez choisir des champs dans chacune des trois tables pour renseigner la table ou la carte.

## Nettoyer

Lorsque vous travaillez dans votre propre abonnement, il est recommandé, à la fin de chaque projet, de déterminer si vous avez toujours besoin des ressources que vous avez créées. Les ressources laissées en cours d'exécution peuvent vous coûter de l'argent. Vous pouvez supprimer les ressources une par une, ou choisir de supprimer le groupe de ressources afin de supprimer l'ensemble des ressources.

Vous pouvez rechercher et gérer les ressources dans le portail à l'aide des liens **Toutes les ressources** ou **Groupes de ressources** situés dans le volet de navigation de gauche.

Si vous utilisez un service gratuit, n'oubliez pas que vous êtes limité à trois index, indexeurs et sources de données. Vous pouvez supprimer des éléments un par un dans le portail pour ne pas dépasser la limite.

## Étapes suivantes

Pour apprendre à explorer cette base de connaissances à l'aide de l'Explorateur Stockage, consultez l'article suivant.

[Voir avec l'Explorateur Stockage](#)

# guide pratique pour mettre en forme et exporter des enrichissements

04/10/2020 • 25 minutes to read • [Edit Online](#)

Les projections sont l'expression physique de documents enrichis dans une base de connaissances. L'utilisation efficace de documents enrichis nécessite une structure. Dans cet article, vous allez explorer la structure et les relations, apprendre à créer des propriétés de projection ainsi qu'à lier des données aux différents types de projection créés.

Pour créer une projection, les données sont mises en forme à l'aide d'une [compétence Modélisateur](#) afin de créer un objet personnalisé ou à l'aide de la syntaxe de mise en forme inline dans une définition de projection.

Une forme de données contient toutes les données devant être projetées, mises en forme en tant que hiérarchie de nœuds. Cet article présente plusieurs techniques pour mettre en forme les données afin de pouvoir ensuite les projeter dans des structures physiques qui permettent la création de rapports, l'analyse ou le traitement en aval.

Les exemples étudiés dans cet article sont tirés de cet [exemple d'API REST](#), que vous pouvez télécharger et exécuter dans un client HTTP.

## Présentation des exemples de projection

Il existe trois types de [projections](#) :

- Tables
- Objets
- Fichiers

Les projections de table sont stockées dans le stockage Table Azure. Les projections d'objet et de fichier sont écrites dans le Stockage Blob, où les projections d'objet sont enregistrées en tant que fichiers JSON, et peuvent contenir le contenu du document source ainsi que les sorties ou enrichissements de compétence. Le pipeline d'enrichissement peut également extraire des binaires tels que des images ; ces binaires sont projetés en tant que projections de fichier. Quand un objet binaire est projeté en tant que projection d'objet, seules les métadonnées qui lui sont associées sont enregistrées en tant qu'objet blob JSON.

Pour comprendre l'intersection entre la mise en forme des données et les projections, nous allons utiliser l'ensemble de compétences suivant comme base pour l'exploration de différentes configurations. Cet ensemble de compétences traite le contenu d'images et de texte brut. Pour les scénarios souhaités, les projections sont définies à partir du contenu du document et des sorties des compétences.

### IMPORTANT

Quand vous expérimentez des projections, il est utile de [définir la propriété de cache de l'indexeur](#) pour garantir le contrôle des coûts. La modification des projections entraîne l'enrichissement de la totalité du document si le cache de l'indexeur n'est pas défini. Quand le cache est défini et que seules les projections sont mises à jour, les exécutions de l'ensemble de compétences pour les documents précédemment enrichis n'entraînent pas de frais supplémentaires liés à Cognitive Services.

```
{  
    "name": "azureblob-skillset",  
    "description": "Skillset created from the portal. skillsetName: azureblob-skillset; contentField:  
merged_content; enrichmentGranularity: document; knowledgeStoreStorageAccount: confdemo;"},
```

```
  "skills": [
    {
        "@odata.type": "#Microsoft.Skills.Text.EntityRecognitionSkill",
        "name": "#1",
        "description": null,
        "context": "/document/merged_content",
        "categories": [
            "Person",
            "Quantity",
            "Organization",
            "URL",
            "Email",
            "Location",
            "DateTime"
        ],
        "defaultLanguageCode": "en",
        "minimumPrecision": null,
        "includeTypelessEntities": null,
        "inputs": [
            {
                "name": "text",
                "source": "/document/merged_content"
            },
            {
                "name": "languageCode",
                "source": "/document/language"
            }
        ],
        "outputs": [
            {
                "name": "persons",
                "targetName": "people"
            },
            {
                "name": "organizations",
                "targetName": "organizations"
            },
            {
                "name": "locations",
                "targetName": "locations"
            },
            {
                "name": "entities",
                "targetName": "entities"
            }
        ]
    },
    {
        "@odata.type": "#Microsoft.Skills.Text.KeyPhraseExtractionSkill",
        "name": "#2",
        "description": null,
        "context": "/document/merged_content",
        "defaultLanguageCode": "en",
        "maxKeyPhraseCount": null,
        "inputs": [
            {
                "name": "text",
                "source": "/document/merged_content"
            },
            {
                "name": "languageCode",
                "source": "/document/language"
            }
        ],
        "outputs": [
            {
                "name": "keyPhrases",
                "targetName": "keyphrases"
            }
        ]
    }
]
```

```
        ],
    },
    {
        "@odata.type": "#Microsoft.Skills.Text.LanguageDetectionSkill",
        "name": "#3",
        "description": null,
        "context": "/document",
        "inputs": [
            {
                "name": "text",
                "source": "/document/merged_content"
            }
        ],
        "outputs": [
            {
                "name": "languageCode",
                "targetName": "language"
            }
        ]
    },
    {
        "@odata.type": "#Microsoft.Skills.Text.MergeSkill",
        "name": "#4",
        "description": null,
        "context": "/document",
        "insertPreTag": " ",
        "insertPostTag": " ",
        "inputs": [
            {
                "name": "text",
                "source": "/document/content"
            },
            {
                "name": "itemsToInsert",
                "source": "/document/normalized_images/*/text"
            },
            {
                "name": "offsets",
                "source": "/document/normalized_images/*/contentOffset"
            }
        ],
        "outputs": [
            {
                "name": "mergedText",
                "targetName": "merged_content"
            }
        ]
    },
    {
        "@odata.type": "#Microsoft.Skills.Vision.OcrSkill",
        "name": "#5",
        "description": null,
        "context": "/document/normalized_images/*",
        "textExtractionAlgorithm": "printed",
        "lineEnding": "Space",
        "defaultLanguageCode": "en",
        "detectOrientation": true,
        "inputs": [
            {
                "name": "image",
                "source": "/document/normalized_images/*"
            }
        ],
        "outputs": [
            {
                "name": "text",
                "targetName": "text"
            },
            {
                "name": "languageCode",
                "targetName": "language"
            }
        ]
    }
]
```

```

        "name": "layoutText",
        "targetName": "layoutText"
    }
],
"cognitiveServices": {
    "@odata.type": "#Microsoft.Azure.Search.CognitiveServicesByKey",
    "description": "DemosCS",
    "key": "<COGNITIVE SERVICES KEY>"
},
"knowledgeStore": null
}

```

À partir de cet ensemble de compétences, avec la valeur `knowledgeStore` Null comme base, notre premier exemple remplit l'objet `knowledgeStore`, configuré avec des projections qui créent des structures de données tabulaires utilisables dans d'autres scénarios.

## Projection dans des tables

La projection dans des tables dans le Stockage Azure est utile pour la création de rapports et l'analyse des données à l'aide d'outils comme Power BI. Power BI peut lire le contenu des tables et découvrir les relations d'après les clés qui sont générées pendant la projection. Si vous voulez créer un tableau de bord, cette tâche sera plus simple si les données sont liées entre elles.

Créons un tableau de bord dans lequel il est possible de visualiser les expressions clés extraites de documents en tant que nuage de mots clés. Pour créer la structure de données appropriée, ajoutez une compétence Modélisateur à l'ensemble de compétences en vue de créer une forme personnalisée avec les détails et les expressions clés spécifiques du document. La forme personnalisée est appelée `pbiShape` sur le nœud racine `document`.

### NOTE

Les projections de table sont des tables du Stockage Azure auxquelles s'appliquent les limites de stockage imposées par le Stockage Azure. Pour plus d'informations, consultez les [limites du stockage de table](#). Il est utile de savoir que la taille maximale respective d'une entité et d'une propriété est de 1 Mo et 64 Ko. Ces contraintes font des tables une bonne solution pour stocker un grand nombre de petites entités.

### Utilisation d'une compétence Modélisateur pour créer une forme personnalisée

Créez une forme personnalisée que vous pouvez projeter dans le stockage de table. Sans forme personnalisée, une projection ne peut référencer qu'un seul nœud (une projection par sortie). La création d'une forme personnalisée permet de regrouper différents éléments en un nouvel ensemble logique qui peut être projeté sous la forme d'une table unique, ou découpé et distribué dans une collection de tables.

Dans cet exemple, la forme personnalisée combine les métadonnées et les entités et expressions clés identifiées. L'objet est appelé `pbiShape` et est apparenté à `/document`.

## IMPORTANT

L'une des finalités de la mise en forme est de s'assurer que tous les nœuds d'enrichissement sont exprimés dans un format JSON correct, ce qui est obligatoire pour la projection dans une base de connaissances. Cela est particulièrement vrai quand une arborescence d'enrichissement contient des nœuds qui n'ont pas le bon format JSON (par exemple, si un enrichissement est apparenté à une primitive comme une chaîne).

Notez que les deux derniers nœuds, `KeyPhrases` et `Entities`, sont wrappés dans un objet JSON valide avec le `sourceContext`. Cela est nécessaire, car `keyphrases` et `entities` sont des enrichissements sur les primitives et ils doivent être convertis en JSON valide avant de pouvoir être projetés.

```
{
    "@odata.type": "#Microsoft.Skills.Util.ShaperSkill",
    "name": "ShaperForTables",
    "description": null,
    "context": "/document",
    "inputs": [
        {
            "name": "metadata_storage_content_type",
            "source": "/document/metadata_storage_content_type",
            "sourceContext": null,
            "inputs": []
        },
        {
            "name": "metadata_storage_name",
            "source": "/document/metadata_storage_name",
            "sourceContext": null,
            "inputs": []
        },
        {
            "name": "metadata_storage_path",
            "source": "/document/metadata_storage_path",
            "sourceContext": null,
            "inputs": []
        },
        {
            "name": "metadata_content_type",
            "source": "/document/metadata_content_type",
            "sourceContext": null,
            "inputs": []
        },
        {
            "name": "keyPhrases",
            "source": null,
            "sourceContext": "/document/merged_content/keyphrases/*",
            "inputs": [
                {
                    "name": "KeyPhrases",
                    "source": "/document/merged_content/keyphrases/*"
                }
            ]
        },
        {
            "name": "Entities",
            "source": null,
            "sourceContext": "/document/merged_content/entities/*",
            "inputs": [
                {
                    "name": "Entities",
                    "source": "/document/merged_content/entities/*/name"
                }
            ]
        }
    ],
    "outputs": [
        {
            "name": "output",
            "targetName": "pbIShape"
        }
    ]
}
```

Ajoutez la compétence Modélisateur à l'ensemble de compétences.

```

    "name": "azureblob-skillset",
    "description": "A friendly description of the skillset goes here.",
    "skills": [
        {
            Shaper skill goes here
        }
    ],
    "cognitiveServices": "A key goes here",
    "knowledgeStore": []
}

```

Toutes les données nécessaires à la projection dans des tables étant désormais disponibles, mettez à jour l'objet knowledgeStore avec les définitions des tables. Dans cet exemple, nous avons trois tables, définies par les propriétés `tableName`, `source` et `generatedKeyName`.

```

"knowledgeStore" : {
    "storageConnectionString": "DefaultEndpointsProtocol=https;AccountName=<Acct Name>;AccountKey=<Acct
Key>;",
    "projections": [
        {
            "tables": [
                {
                    "tableName": "pbiDocument",
                    "generatedKeyName": "Documentid",
                    "source": "/document/pbiShape"
                },
                {
                    "tableName": "pbiKeyPhrases",
                    "generatedKeyName": "KeyPhraseid",
                    "source": "/document/pbiShape/keyPhrases/*"
                },
                {
                    "tableName": "pbiEntities",
                    "generatedKeyName": "Entityid",
                    "source": "/document/pbiShape/Entities/*"
                }
            ],
            "objects": [],
            "files": []
        }
    ]
}

```

Vous pouvez effectuer votre travail en procédant comme suit :

1. Définissez la propriété `storageConnectionString` sur une chaîne de connexion de compte de stockage v2 universel valide.
2. Mettez à jour l'ensemble de compétences en envoyant la requête PUT.
3. Après la mise à jour de l'ensemble de compétences, exécutez l'indexeur.

Vous disposez maintenant d'une projection de travail avec trois tables. Quand vous importez ces tables dans Power BI, celui-ci doit être en mesure de détecter les relations automatiquement.

Avant de passer à l'exemple suivant, revoyons les aspects de la projection de table pour comprendre les mécanismes de découpage et des données associées.

## Découpage

Le découpage est une technique qui divise une forme regroupée entière en plusieurs parties constitutives. Vous obtenez des tables distinctes mais liées entre elles, que vous pouvez utiliser individuellement.

Dans l'exemple, `pbiShape` est la forme regroupée (ou nœud d'enrichissement). Dans la définition de la projection, `pbiShape` est découpé en tables supplémentaires, ce qui vous permet d'extraire des parties de la forme, `keyPhrases` et `Entities`. Dans Power BI, cette opération est utile lorsque plusieurs entités et expressions clés sont associées à chaque document : vous obtenez des insights plus complets si vous pouvez visualiser les entités et les expressions clés sous forme de données classifiées par catégorie.

Le découpage génère implicitement une relation entre les tables parent et enfant, en utilisant le `generatedKeyName` dans la table parent pour créer une colonne du même nom dans la table enfant.

### Nommage des relations

Les propriétés `generatedKeyName` et `referenceKeyName` sont utilisées pour lier des données entre des tables ou même entre des types de projection. Chaque ligne de la table enfant/projection a une propriété qui pointe vers le parent. Le nom de la colonne ou de la propriété dans l'enfant est le `referenceKeyName` du parent. Quand le `referenceKeyName` n'est pas fourni, le service le définit par défaut sur le `generatedKeyName` du parent.

Power BI utilise ces clés générées pour découvrir les relations dans les tables. S'il est nécessaire que la colonne de la table enfant soit nommée différemment, définissez la propriété `referenceKeyName` sur la table . Par exemple, vous pouvez définir le `generatedKeyName` en tant qu'ID sur la table `pbiDocument` et le `referenceKeyName` en tant que `DocumentID`. Ainsi, la colonne dans les tables `pbiEntities` et `pbiKeyPhrases` contenant l'ID de document seraient nommées `DocumentID`.

## Projection dans des objets

Les projections d'objet n'ont pas les mêmes limitations que les projections de table et elles sont mieux adaptées pour la projection de documents volumineux. Dans cet exemple, la totalité du document est envoyée en tant que projection d'objet. Les projections d'objet sont limitées à une seule projection dans un conteneur et elles ne peuvent pas être découpées.

Pour définir une projection d'objet, utilisez le tableau `objects` dans les projections. Vous pouvez générer une nouvelle forme à l'aide de la compétence Modélisateur ou utiliser la mise en forme inline de la projection d'objet. Tandis que l'exemple de tables illustre l'approche de la création d'une forme et du découpage, cet exemple illustre l'utilisation de la mise en forme inline.

La mise en forme inline permet de créer une forme dans la définition des entrées d'une projection. La mise en forme inline crée un objet anonyme identique à ce qu'une compétence Modélisateur produirait (ici, `pbiShape`). La mise en forme inline est utile si vous définissez une forme que vous n'envisagez pas de réutiliser.

La propriété `projections` est un tableau. Cet exemple comprend une nouvelle instance de projection au tableau, où la définition `knowledgeStore` contient des projections inline. Quand vous utilisez des projections inline, vous pouvez omettre la compétence Modélisateur.

```

"knowledgeStore" : {
    "storageConnectionString": "DefaultEndpointsProtocol=https;AccountName=<Acct Name>;AccountKey=<Acct Key>;",
    "projections": [
        {
            "tables": [ ],
            "objects": [
                {
                    "storageContainer": "sampleobject",
                    "source": null,
                    "generatedKeyName": "myobject",
                    "sourceContext": "/document",
                    "inputs": [
                        {
                            "name": "metadata_storage_name",
                            "source": "/document/metadata_storage_name"
                        },
                        {
                            "name": "metadata_storage_path",
                            "source": "/document/metadata_storage_path"
                        },
                        {
                            "name": "content",
                            "source": "/document/content"
                        },
                        {
                            "name": "keyPhrases",
                            "source": "/document/merged_content/keyphrases/*"
                        },
                        {
                            "name": "entities",
                            "source": "/document/merged_content/entities/*/name"
                        },
                        {
                            "name": "ocrText",
                            "source": "/document/normalized_images/*/text"
                        },
                        {
                            "name": "ocrLayoutText",
                            "source": "/document/normalized_images/*/layoutText"
                        }
                    ]
                }
            ],
            "files": []
        }
    ]
}

```

## Projection dans un fichier

Les projections de fichier sont des images extraites du document source ou des sorties d'enrichissement qui peuvent être projetées à partir du processus d'enrichissement. Les projections de fichier, comme les projections d'objet, sont implémentées sous forme d'objets blob dans le Stockage Azure et contiennent l'image.

Pour générer une projection de fichier, utilisez le tableau `files` dans l'objet de projection. Cet exemple projette toutes les images extraites du document dans un conteneur appelé `samplefile`.

```

"knowledgeStore" : {
    "storageConnectionString": "DefaultEndpointsProtocol=https;AccountName=<Acct Name>;AccountKey=<Acct Key>;",
    "projections": [
        {
            "tables": [ ],
            "objects": [ ],
            "files": [
                {
                    "storageContainer": "samplefile",
                    "source": "/document/normalized_images/*"
                }
            ]
        }
    ]
}

```

## Projection dans plusieurs types

Un scénario plus complexe peut vous obliger à projeter du contenu dans différents types de projection. Par exemple, si vous devez projeter des données telles que des expressions clés et des entités dans des tables, enregistrez les résultats OCR du texte et du texte de disposition en page sous forme d'objets, puis projetez les images en tant que fichiers.

Cet exemple comprend les mises à jour apportées à l'ensemble de compétences avec les modifications suivantes :

1. Créer une table avec une ligne pour chaque document.
2. Créer une table liée à la table de documents avec chaque expression clé identifiée en tant que ligne dans cette table.
3. Créer une table liée à la table de documents avec chaque entité identifiée en tant que ligne dans cette table.
4. Créer une projection d'objet avec le texte de disposition pour chaque image.
5. Créez une projection de fichier, en projetant chaque image extraite.
6. Créer une table de référence croisée qui contient des références à la table de document, à la projection d'objet avec le texte de disposition et à la projection de fichier.

Ces modifications sont répercutées plus loin dans la définition knowledgeStore.

### Mettre en forme les données pour la projection croisée

Pour obtenir les formes nécessaires pour ces projections, commencez par ajouter une nouvelle compétence Modélisateur qui crée un objet mis en forme appelé `crossProjection`.

```
{
    "@odata.type": "#Microsoft.Skills.Util.ShaperSkill",
    "name": "ShaperForCross",
    "description": null,
    "context": "/document",
    "inputs": [
        {
            "name": "metadata_storage_name",
            "source": "/document/metadata_storage_name",
            "sourceContext": null,
            "inputs": []
        },
        {
            "name": "keyPhrases",
            "source": null,
            "sourceContext": "/document/merged_content/keyphrases/*",
            "inputs": [
                {
                    "name": "KeyPhrases",
                    "source": "/document/merged_content/keyphrases/*"
                }
            ]
        },
        {
            "name": "entities",
            "source": null,
            "sourceContext": "/document/merged_content/entities/*",
            "inputs": [
                {
                    "name": "Entities",
                    "source": "/document/merged_content/entities/*/name"
                }
            ]
        },
        {
            "name": "images",
            "source": null,
            "sourceContext": "/document/normalized_images/*",
            "inputs": [
                {
                    "name": "image",
                    "source": "/document/normalized_images/*"
                },
                {
                    "name": "layoutText",
                    "source": "/document/normalized_images/*/layoutText"
                },
                {
                    "name": "ocrText",
                    "source": "/document/normalized_images/*/text"
                }
            ]
        }
    ],
    "outputs": [
        {
            "name": "output",
            "targetName": "crossProjection"
        }
    ]
}
```

## Définir des projections de table, d'objet et de fichier

À partir de l'objet crossProjection regroupé, découpez l'objet en plusieurs tables, capturez la sortie OCR sous forme d'objets blob, puis enregistrez l'image en tant que fichiers (également dans le Stockage Blob).

```
"knowledgeStore" : {
    "storageConnectionString": "DefaultEndpointsProtocol=https;AccountName=<Acct Name>;AccountKey=<Acct Key>;",
    "projections": [
        {
            "tables": [
                {
                    "tableName": "crossDocument",
                    "generatedKeyName": "Id",
                    "source": "/document/crossProjection"
                },
                {
                    "tableName": "crossEntities",
                    "generatedKeyName": "EntityId",
                    "source": "/document/crossProjection/entities/*"
                },
                {
                    "tableName": "crossKeyPhrases",
                    "generatedKeyName": "KeyPhraseId",
                    "source": "/document/crossProjection/keyPhrases/*"
                },
                {
                    "tableName": "crossReference",
                    "generatedKeyName": "CrossId",
                    "source": "/document/crossProjection/images/*"
                }
            ],
            "objects": [
                {
                    "storageContainer": "crossobject",
                    "generatedKeyName": "crosslayout",
                    "source": null,
                    "sourceContext": "/document/crossProjection/images/*/layoutText",
                    "inputs": [
                        {
                            "name": "OcrLayoutText",
                            "source": "/document/crossProjection/images/*/layoutText"
                        }
                    ]
                }
            ],
            "files": [
                {
                    "storageContainer": "crossimages",
                    "generatedKeyName": "crossimages",
                    "source": "/document/crossProjection/images/*/image"
                }
            ]
        }
    ]
}
```

Les projections d'objet nécessitent un nom de conteneur pour chaque projection ; les projections d'objet ou les projections de fichier ne peuvent pas partager de conteneur.

### Relations entre les projections de table, d'objet et de fichier

Cet exemple met également en lumière une autre fonctionnalité des projections. En définissant plusieurs types de projections dans le même objet de projection, il existe une relation exprimée dans et entre les différents types (tables, objets, fichiers). Cela vous permet de commencer avec une ligne de tableau pour un document et de

rechercher tout le texte OCR pour les images contenues dans ce document dans la projection de l'objet.

Si vous ne souhaitez pas que les données soient liées, définissez les projections dans différents objets de projection. Par exemple, l'extrait de code suivant entraîne la liaison des tables, sans qu'aucune relation ne soit toutefois établie entre les tables et les projections d'objet (texte OCR).

Les groupes de projection sont utiles quand vous souhaitez projeter les mêmes données dans différentes formes en fonction de besoins différents. Par exemple, un groupe de projections pour le tableau de bord Power BI et un autre groupe de projections pour la capture des données utilisées pour entraîner un modèle Machine Learning wrappé dans une compétence personnalisée.

Lors de la création de projections de types différents, les projections de fichier et d'objet sont générées en premier, et les chemins sont ajoutés aux tables.

```

"knowledgeStore" : {
    "storageConnectionString": "DefaultEndpointsProtocol=https;AccountName=<Acct Name>;AccountKey=<Acct Key>;",
    "projections": [
        {
            "tables": [
                {
                    "tableName": "unrelatedDocument",
                    "generatedKeyName": "Documentid",
                    "source": "/document/pbiShape"
                },
                {
                    "tableName": "unrelatedKeyPhrases",
                    "generatedKeyName": "KeyPhraseid",
                    "source": "/document/pbiShape/keyPhrases"
                }
            ],
            "objects": [
                ],
            "files": []
        },
        {
            "tables": [],
            "objects": [
                {
                    "storageContainer": "unrelatedocrtext",
                    "source": null,
                    "sourceContext": "/document/normalized_images/*/text",
                    "inputs": [
                        {
                            "name": "ocrText",
                            "source": "/document/normalized_images/*/text"
                        }
                    ]
                },
                {
                    "storageContainer": "unrelatedocrlayout",
                    "source": null,
                    "sourceContext": "/document/normalized_images/*/layoutText",
                    "inputs": [
                        {
                            "name": "ocrLayoutText",
                            "source": "/document/normalized_images/*/layoutText"
                        }
                    ]
                }
            ],
            "files": []
        }
    ]
}

```

## Problèmes courants

Lors de la définition d'une projection, il existe quelques problèmes courants qui peuvent entraîner des résultats imprévus. Vérifiez les points ci-dessous si la sortie dans la base de connaissances ne correspond pas à la sortie prévue.

- Absence de la mise en forme des enrichissements de chaînes dans un JSON valide. Quand des chaînes sont enrichies, par exemple la chaîne `merged_content` enrichie d'expressions clés, la propriété enrichie est représentée comme enfant de `merged_content` dans l'arborescence des enrichissements. La représentation par défaut n'est pas un JSON correctement mis en forme. Au moment de la projection, vous devez donc

transformer l'enrichissement en un objet JSON valide avec un nom et une valeur.

- Omission de `/*` à la fin d'un chemin source. Si la source d'une projection est `/document/pbiShape/keyPhrases`, le tableau d'expressions clés est projeté sous la forme d'un objet ou d'une ligne unique. À la place, définissez le chemin source sur `/document/pbiShape/keyPhrases/*` afin d'obtenir une seule ligne ou un seul objet pour chacune des expressions clés.
- Erreurs dans la syntaxe du chemin. Les sélecteurs de chemin respectent la casse et peuvent entraîner des avertissements d'entrée manquante si vous n'utilisez pas la casse exacte du sélecteur.

## Étapes suivantes

Les exemples de cet article montrent des modèles courants de création de projections. Maintenant que vous avez bien compris les concepts, vous êtes mieux armé pour créer des projections adaptées à votre scénario particulier.

À mesure que vous explorez de nouvelles fonctionnalités, envisagez l'enrichissement incrémentiel en guise d'étape suivante. L'enrichissement incrémentiel est basé sur la mise en cache, ce qui vous permet de réutiliser tous les enrichissements qui ne sont pas impactés par une modification de l'ensemble de compétences. Cela s'avère particulièrement utile pour les pipelines qui incluent l'OCR et l'analyse des images.

### [Présentation de l'enrichissement incrémentiel et de la mise en cache](#)

Pour obtenir une vue d'ensemble des projections, découvrez les fonctionnalités telles que les groupes et le découpage, et comment vous [les définissez dans un ensemble de compétences](#).

### [Projections dans une base de connaissances](#)

# Types et composition de requête dans Recherche cognitive Azure

04/10/2020 • 22 minutes to read • [Edit Online](#)

Dans Recherche cognitive Azure, une requête est une spécification complète d'une opération d'aller-retour. Dans la requête, il existe des paramètres qui fournissent des instructions d'exécution pour le moteur, ainsi que des paramètres qui forment la réponse en retour. Non spécifiée (`search=*`), sans critères de correspondance et qui utilise des paramètres nuls ou par défaut, une requête s'exécute sur tous les champs recherchables en tant qu'opération de recherche de texte intégral et retourne un jeu de résultats sans score dans un ordre arbitraire.

L'exemple suivant est une requête représentative construite dans l'[API REST](#). Cet exemple cible l'[index de démonstration des hôtels](#) et comprend des paramètres communs afin que vous puissiez vous faire une idée de ce à quoi ressemble une requête.

```
{  
    "queryType": "simple"  
    "search": "+New York +restaurant",  
    "searchFields": "Description, Address/City, Tags",  
    "select": "HotelId, HotelName, Description, Rating, Address/City, Tags",  
    "top": "10",  
    "count": "true",  
    "orderby": "Rating desc"  
}
```

- `queryType` définit l'analyseur qui peut être l'[analyseur de requêtes simple par défaut](#) (optimal pour la recherche en texte intégral), ou l'[analyseur de requêtes complet Lucene](#) utilisé pour les constructions de requêtes avancées, telles que les expressions régulières, la recherche de proximité, la recherche approximative et par caractères génériques et bien d'autres encore.
- `search` fournit le critère de correspondance, généralement des termes ou expressions, qui est néanmoins souvent accompagné d'opérateurs booléens. Les termes autonomes uniques constituent des requêtes de *terme*. Les requêtes en plusieurs parties entre guillemets sont des requêtes *d'expression*. La recherche peut être non définie, comme dans `search=*`, mais sans critère de correspondance, le jeu de résultats est composé de documents sélectionnés de manière arbitraire.
- `searchFields` limite l'exécution des requêtes à des champs spécifiques. Tout champ attribué en tant que champ avec *possibilité de recherche* dans le schéma d'index peut être utilisé avec ce paramètre.

Les réponses sont également mises en forme par les paramètres que vous incluez dans la requête :

- `select` spécifie les champs à retourner dans la réponse. Seuls les champs marqués comme *récupérables* dans l'index peuvent être utilisés dans une instruction select.
- `top` retourne le nombre de documents les mieux correspondants spécifié. Dans cet exemple, seuls 10 correspondances sont retournées. Vous pouvez utiliser Top et Skip (non affichés) pour paginer les résultats.
- `count` indique le nombre total de documents dans l'index complet, ce qui peut être supérieur à ce qui est retourné.
- `orderby` est utilisé si vous souhaitez trier les résultats en fonction d'une valeur, telle qu'une évaluation ou un emplacement. Dans le cas contraire, la valeur par défaut consiste à utiliser le score de pertinence pour

classer les résultats.

Dans Recherche cognitive Azure, la requête s'exécute toujours sur un seul index qui est authentifié à l'aide d'une clé d'API (api-key) fournie dans la requête. Dans l'API REST, les deux éléments sont fournis dans les en-têtes de la requête.

### Comment exécuter cette requête

Avant d'écrire du code, vous pouvez utiliser les outils de requête pour apprendre la syntaxe et tester différents paramètres. L'approche la plus rapide est l'outil de portail intégré, l'[Explorateur de recherche](#).

Si vous avez suivi ce [démarrage rapide pour créer l'index de démonstration des hôtels](#), vous pouvez coller cette chaîne de requête dans la barre de recherche de l'Explorateur pour exécuter votre première requête :

```
search+="New York" +restaurant&searchFields=Description, Address/City, Tags&$select=HotelId, HotelName, Description, Rating, Address/City, Tags&$top=10&$orderby=Rating desc&$count=true
```

## Comment les opérations de requête sont activées par l'index

La conception des index et la conception des requêtes sont étroitement liées dans Recherche cognitive Azure. Il est important de savoir que le *schéma d'index*, qui comporte des attributs pour chaque champ, détermine le type de requête que vous pouvez créer.

Les attributs d'index d'un champ définissent les opérations autorisées, par exemple si un champ *peut faire l'objet de recherches* dans l'index, s'il peut être *affiché* dans les résultats, s'il peut être *trié* ou *filtré*, etc. Dans l'exemple de chaîne de requête, `"$orderby": "Rating"` fonctionne uniquement parce que le champ Évaluation est marqué comme *trieable* dans le schéma d'index.

FIELD NAME	TYPE	RETRIEVABLE	FILTERABLE	SORTABLE	FACTETABLE	SEARCHABLE
🔑 HotelId	Edm.String	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
HotelName	Edm.String	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Description	Edm.String	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Description_fr	Edm.String	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Category	Edm.String	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Tags	Collection(E...	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
ParkingIncluded	Edm.Boolean	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
LastRenovationDate	Edm.DateTi...	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
Rating	Edm.Double	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	

La capture d'écran ci-dessus présente une liste partielle des attributs d'index pour l'exemple des hôtels. Vous pouvez consulter le schéma d'index complet dans le portail. Pour en savoir plus sur les attributs d'index, consultez [Création d'une API REST d'index](#).

#### NOTE

Certaines fonctionnalités de requête s'appliquent à l'ensemble de l'index plutôt qu'à des champs spécifiques. Ces fonctionnalités incluent : les [cartes de synonymes](#), les [analyseurs personnalisés](#), les [constructions de générateur de suggestions](#) (pour les requêtes de saisie semi-automatique et de suggestion automatique) et la [logique de notation](#) pour le classement des résultats.

## Éléments d'une demande de requête

Les requêtes s'appliquent toujours à un index unique. Vous ne pouvez pas joindre des index ou créer des

structures de données personnalisées ou temporaires en tant que requête cible.

Les éléments obligatoires dans une demande de requête incluent les composants suivants :

- Collection des points de terminaison de service et des documents d'index, exprimée sous la forme d'une URL contenant les composants fixes et définis par l'utilisateur :  
`https://<your-service-name>.search.windows.net/indexes/<your-index-name>/docs`
- `api-version` (API REST uniquement) est requis car plusieurs versions de l'API sont disponibles en permanence.
- `api-key`, représenté par une clé d'API de requête ou administration, authentifie la requête auprès votre service.
- `queryType`, simple ou full, qui peut être omis si vous utilisez la syntaxe simple par défaut intégrée.
- `search` ou `filter` fournit les critères de recherche, qui peuvent être non spécifiés si vous souhaitez effectuer une recherche vide. Les deux types de requêtes sont décrits dans le cadre d'un analyseur simple, mais même les requêtes avancées requièrent le paramètre de recherche pour prendre en compte les expressions de requêtes complexes.

Tous les autres paramètres de recherche sont facultatifs. Pour obtenir la liste complète des attributs, consultez la section relative à la [création d'index \(REST\)](#). Pour en savoir plus sur le rôle des paramètres lors du traitement des requêtes, consultez [Fonctionnement de la recherche en texte intégral dans Recherche cognitive Azure](#).

## Choisir les API et les outils

Le tableau suivant liste les API et les approches basées sur des outils pour envoyer des requêtes.

MÉTHODOLOGIE	DESCRIPTION
<a href="#">Navigateur de recherche (portail)</a>	Fournit une barre de recherche et des options pour les sélections d'index et de version d'API. Les résultats sont retournés sous forme de documents JSON. Il est recommandé pour l'exploration, le test et la validation. <a href="#">En savoir plus.</a>
<a href="#">Postman ou autres outils REST</a>	Les outils de test web constituent un excellent choix pour formuler des appels REST. L'API REST prend en charge toutes les opérations possibles dans Recherche cognitive Azure. Dans cet article, découvrez comment configurer un en-tête et un corps de requête HTTP pour l'envoi de requêtes à Recherche cognitive Azure.
<a href="#">SearchIndexClient (.NET)</a>	Client que vous pouvez utiliser pour interroger un index Recherche cognitive Azure. <a href="#">En savoir plus.</a>
<a href="#">Rechercher des documents (API REST)</a>	Méthodes GET ou POST sur un index, avec paramètres de requête pour une entrée supplémentaire.

## Choisir un analyseur : simple | full

Le service Recherche cognitive Azure repose sur Apache Lucene et vous offre le choix entre deux analyseurs de requêtes pour la gestion des requêtes classiques et spécialisées. Les requêtes qui utilisent l'analyseur simple sont formulées à l'aide de la [syntaxe de requête simple](#), qui est sélectionnée par défaut pour sa rapidité et son efficacité dans les requêtes de texte au format libre. Cette syntaxe prend en charge un certain nombre d'opérateurs de recherche courants, notamment les opérateurs AND, OR, NOT, les expressions, les suffixes et les opérateurs de priorité.

La [syntaxe de requête complète Lucene](#), activée lorsque vous ajoutez `queryType=full` à la requête, expose le langage de requête expressif et largement adopté développé dans le cadre [d'Apache Lucene](#). La syntaxe complète étend la syntaxe simple. Toute requête que vous écrivez avec la syntaxe simple s'exécute dans l'analyseur Lucene complet.

Les exemples suivants illustrent parfaitement la situation : la même requête, mais avec des paramètres `queryType` différents, génère des résultats différents. Dans la première requête, l'élément `^3` après `historic` est traité comme une partie du terme recherché. Pour cette requête, le résultat qui arrive en tête du classement est « Marquis Plaza & Suites », dont la description comprend le terme *ocean*.

```
queryType=simple&search=ocean historic^3&searchFields=Description, Tags&$select=HotelId, HotelName, Tags, Description&$count=true
```

La même requête exécutée à l'aide de l'analyseur Lucene complet interprète `^3` comme élément de champ prioritaire. Le changement d'analyseur modifie le classement. Ainsi, les résultats contenant le terme *historic* arrivent en tête.

```
queryType=full&search=ocean historic^3&searchFields=Description, Tags&$select=HotelId, HotelName, Tags, Description&$count=true
```

## Types de requête

Recherche cognitive Azure prend en charge un large éventail de types de requêtes.

TYPE DE REQUÊTE	USAGE	EXEMPLES ET INFORMATIONS COMPLÉMENTAIRES
Recherche de texte de forme libre	Paramètre de recherche et analyseur au choix	<p>Une recherche en texte intégral recherche un ou plusieurs termes dans tous les champs <i>pouvant faire l'objet d'une recherche</i> de votre index, et fonctionne à l'instar des moteurs de recherche Google ou Bing. L'exemple dans l'introduction est une recherche en texte intégral.</p> <p>La recherche en texte intégral fait l'objet d'une analyse lexicale à l'aide de l'analyseur Lucene standard (par défaut) pour mettre tous les termes en minuscules et supprimer les mots exclus tels que « the » (la, le, les). Vous pouvez remplacer l'analyseur par défaut en choisissant un <a href="#">analyseur non anglais</a> ou un <a href="#">analyseur spécialisé indépendant des langages</a> qui modifiera les paramètres d'analyse lexicale. Un analyseur <a href="#">mot clé</a> traite par exemple tout le contenu d'un champ comme un jeton unique. Ceci est utile pour les données comme les codes postaux, les ID et certains noms de produit.</p>

TYPE DE REQUÊTE	USAGE	EXEMPLES ET INFORMATIONS COMPLÉMENTAIRES
Recherche filtrée	Expression de filtre OData et analyseur au choix	Les requêtes de filtre évaluent une expression booléenne dans tous les champs <i>filtrables</i> d'un index. Contrairement à une recherche, une requête de filtre établit une correspondance avec le contenu exact d'un champ, y compris la casse dans les champs de type chaîne. Une autre différence est que les requêtes de filtre sont exprimées dans la syntaxe OData. <a href="#">Exemple d'expression de filtre</a>
Recherche basée sur la localisation	Champ de type Edm.GeographyPoint, expression de filtre et analyseur au choix	Les coordonnées stockées dans un champ de type Edm.GeographyPoint sont utilisées pour les recherches de type « rechercher à proximité » ou basées sur une carte. <a href="#">Exemple de recherche sur la localisation</a>
Recherche de plage	expression de filtre et analyseur simple	Dans Recherche cognitive Azure, les requêtes de plage sont créées à l'aide du paramètre de filtre. <a href="#">Exemple de filtre de plage</a>
Recherche par champ	Paramètre de recherche et analyseur complet	Crée une expression de requête composite ciblant un champ unique. <a href="#">Exemple de recherche par champ</a>
recherche approximative	Paramètre de recherche et analyseur complet	Recherche les termes ayant une construction ou une orthographe similaire. <a href="#">Exemple de recherche approximative</a>
recherche de proximité	Paramètre de recherche et analyseur complet	Recherche les termes proches les uns des autres dans un document. <a href="#">Exemple de recherche de proximité</a>
promotion de termes	Paramètre de recherche et analyseur complet	Élève le rang d'un document qui contient le terme de promotion, par rapport aux documents qui ne contiennent pas ce terme. <a href="#">Exemple de promotion de termes</a>
recherche d'expression régulière	Paramètre de recherche et analyseur complet	Exécute la recherche à partir du contenu d'une expression régulière. <a href="#">Exemple de recherche d'expression régulière</a>
recherche par préfixe ou caractères génériques	Paramètre de recherche et analyseur complet	Exécute la recherche à partir d'un préfixe et d'un tilde (~) ou d'un caractère unique (?). <a href="#">Exemple de recherche par caractères génériques</a>

## Gérer les résultats de recherche

Les résultats de la requête sont envoyés sous forme de documents JSON dans l'API REST, mais si vous utilisez des API .NET, la sérialisation est intégrée. Vous pouvez mettre en forme les résultats en définissant des paramètres sur la requête, et en sélectionnant des champs spécifiques pour la réponse.

Vous pouvez utiliser les paramètres de la requête pour structurer le jeu de résultats des manières suivantes :

- Limitation ou traitement par lot du nombre de documents dans les résultats (50 par défaut)
- Sélection des champs à inclure dans les résultats
- Définition d'un ordre de tri
- Ajout de la mise en surbrillance des correspondances pour attirer l'attention sur les termes correspondants dans le corps des résultats de recherche

### Conseils en cas de résultats inattendus

Parfois, la substance et non la structure de résultats est inattendue. Quand les résultats de requête ne sont pas à la hauteur de vos attentes, vous pouvez essayer ces modifications de requête pour voir si les résultats s'améliorent :

- Remplacez `searchMode=any` (valeur par défaut) par `searchMode=all` pour exiger des correspondances sur tous les critères plutôt que sur un seul d'entre eux. Cela s'applique particulièrement quand des opérateurs booléens sont inclus dans la requête.
- Changez la technique de requête si l'analyse lexicale ou du texte est nécessaire, mais que le type de requête exclut tout traitement linguistique. Dans la recherche en texte intégral, l'analyse lexicale ou du texte corrige automatiquement les fautes d'orthographe, les formes singulier-pluriel des noms, et même les noms ou les verbes irréguliers. Pour certaines requêtes telles que la recherche approximative ou par caractères génériques, l'analyse lexicale ne fait pas partie du pipeline d'analyse de requête. Dans certains scénarios, des expressions régulières ont été utilisées pour contourner ce problème.

### Résultats de pagination

Recherche cognitive Azure facilite l'implémentation de la pagination des résultats de recherche. À l'aide des paramètres `top` et `skip`, vous pouvez facilement émettre des requêtes de recherche pour recevoir le jeu de résultats complet dans des sous-ensembles gérables, ordonnés qui permettent de bonnes pratiques de recherche dans l'interface utilisateur. Lors de la réception de ces sous-ensembles de résultats plus petits, vous pouvez également recevoir le nombre de documents dans l'ensemble total des résultats de la recherche.

Pour plus d'informations sur la pagination des résultats de recherche, consultez l'article [Navigation dans les résultats de recherche de Recherche cognitive Azure](#).

### Classement des résultats

Lors de la réception des résultats d'une requête de recherche, vous pouvez demander que Recherche cognitive Azure produise les résultats classés par valeurs dans un champ spécifique. Par défaut, Recherche cognitive Azure classe les résultats en fonction du rang du résultat de la recherche de chaque document, qui est dérivé de la méthode [TF-IDF](#).

Si vous souhaitez que Recherche cognitive Azure retourne les résultats en les classant avec une valeur autre que le résultat de la recherche, vous pouvez utiliser le paramètre de recherche `orderby`. Vous pouvez spécifier la valeur du paramètre `orderby` pour inclure les noms de champ et les appels à la fonction `geo.distance()` pour les valeurs géospatiales. Chaque expression peut être suivie par `asc` pour indiquer que les résultats sont demandés dans l'ordre croissant, et par `desc` pour indiquer que les résultats sont demandés dans l'ordre décroissant. Le classement par défaut est l'ordre croissant.

### Mise en surbrillance des correspondances

Dans Recherche cognitive Azure, vous pouvez mettre facilement en évidence la partie exacte des résultats de recherche qui correspondent à la requête de recherche en utilisant les paramètres `highlight`, `highlightPreTag` et `highlightPostTag`. Vous pouvez spécifier les champs *utilisables dans une recherche* dont le texte

correspondant à la requête doit être mis en évidence ainsi que les balises de chaîne exactes à ajouter au début et à la fin du texte correspondant retourné par le service Recherche cognitive Azure.

## Voir aussi

- [Fonctionnement de la recherche en texte intégral dans Recherche cognitive Azure \(architecture d'analyse de requête\)](#)
- [Navigateur de recherche](#)
- [Guide pratique pour interroger dans .NET](#)
- [Guide pratique pour interroger dans REST](#)

# Créer une requête simple dans la Recherche cognitive Azure

04/10/2020 • 20 minutes to read • [Edit Online](#)

Dans la Recherche cognitive Azure, la [syntaxe de requête simple](#) appelle l'analyseur de requêtes par défaut pour l'exécution de requêtes de recherche en texte intégral sur un index. Cet analyseur rapide gère des scénarios courants, notamment la recherche en texte intégral, filtrée et à facettes ainsi que la recherche géographique.

Dans cet article, nous utilisons des exemples pour illustrer la syntaxe simple, en renseignant le paramètre `search=` d'une opération [Rechercher des documents](#).

L'autre syntaxe de requête disponible est la [syntaxe Lucene complète](#), qui prend en charge des structures de requête plus complexes comme la recherche approximative et par caractères génériques, dont le traitement peut être plus long. Pour plus d'informations et pour obtenir des exemples illustrant la syntaxe complète, consultez l'article sur [l'utilisation de la syntaxe Lucene complète](#).

## Formuler des requêtes dans Postman

Les exemples suivants utilisent un index de recherche NYC Jobs composé de postes à pourvoir sur la base d'un jeu de données fourni par l'initiative [City of New York OpenData](#). Ces données ne doivent pas être considérées comme étant à jour ou complètes. L'index se trouve sur un service de bac à sable fourni par Microsoft, ce qui signifie que vous n'avez pas besoin d'abonnement Azure ni de Recherche cognitive Azure pour essayer ces requêtes.

En revanche, vous avez besoin de Postman ou d'un outil équivalent pour émettre la requête HTTP sur GET. Pour plus d'informations, consultez [Démarrage rapide : Explorer les API REST de Recherche cognitive Azure avec Postman](#).

### Définir l'en-tête de requête

1. Dans l'en-tête de requête, affectez la valeur `application/json` à **Content-Type**.
2. Ajoutez un **api-key** et affectez-lui cette chaîne : `252044BE3886FE4A8E3BAA4F595114BB`. Il s'agit d'une clé de requête pour le service de recherche de bac à sable qui héberge l'index NYC Jobs.

Une fois que vous avez spécifié l'en-tête de requête, vous pouvez le réutiliser pour toutes les requêtes dans cet article, en remplaçant uniquement la chaîne `search=`.

Key	Value	Description
Content-Type	application/json	
api-key	252044BE3886FE4A8E3BAA4F595114BB	This is a query key for a sandbox Azure Search service used for demos

### Définir l'URL de requête

La requête est une commande GET accompagnée d'une URL contenant le point de terminaison de Recherche cognitive Azure et la chaîne de recherche.

L'URL est composée des éléments suivants :

- `https://azs-playground.search.windows.net/` est un service de recherche de bac à sable tenu à jour par l'équipe de développement de Recherche cognitive Azure.
- `indexes/nycjobs/` est l'index NYC Jobs dans la collection d'index de ce service. Le nom du service et l'index sont tous deux obligatoires dans la requête.
- `docs` est la collection de documents contenant tout le contenu disponible pour la recherche. La clé d'API de requête fournie dans l'en-tête de requête fonctionne uniquement sur les opérations de lecture ciblant la collection de documents.
- `api-version=2020-06-30` définit la version de l'API, qui est un paramètre requis dans chaque requête.
- `search=*` est la chaîne de requête qui, dans la requête initiale, est nulle, ce qui retourne les 50 premiers résultats (par défaut).

## Envoyer votre première requête

En guise d'étape de vérification, collez la requête suivante dans GET et cliquez sur **Envoyer**. Les résultats sont retournés sous forme de documents JSON détaillés. Des documents entiers sont retournés, ce qui vous permet de voir tous les champs et toutes les valeurs.

Collez cette URL dans un client REST comme étape de validation et pour afficher la structure du document.

```
https://azs-playground.search.windows.net/indexes/nycjobs/docs?api-version=2020-06-30&$count=true&search=*
```

La chaîne de requête `search=*`, est une recherche non spécifiée équivalente à une recherche nulle ou vide. Elle n'est pas particulièrement utile, mais c'est la recherche la plus simple que vous puissiez effectuer.

Si vous le souhaitez, vous pouvez ajouter `$count=true` à l'URL pour retourner le nombre de documents correspondant aux critères de recherche. Une chaîne de recherche vide correspond à tous les documents figurant dans l'index (environ 2800 dans le cas de NYC Jobs).

## Appel de l'analyse de requête simple

Pour les requêtes interactives, vous n'avez rien à spécifier : l'analyse simple est l'analyse par défaut. Dans le code, si vous avez déjà appelé `queryType=full` pour la syntaxe de requête complète, vous pouvez rétablir la valeur par défaut avec `queryType=simple`.

## Exemple 1 : Requête sur des champs

Ce premier exemple n'est pas propre à un analyseur, mais nous permet d'introduire le premier concept de requête fondamental : la contenance. Cet exemple limite l'exécution de la requête et la réponse à quelques champs spécifiques. Lors de l'utilisation de l'outil Postman ou Explorateur de recherche, il est important de connaître la structure d'une réponse JSON accessible en lecture.

Par souci de concision, la requête cible uniquement le champ `business_title` et spécifie que seuls les titres de fonctions sont retournés. La syntaxe est `searchFields` pour restreindre l'exécution de la requête au champ `business_title`, ainsi que `select` pour spécifier les champs inclus dans la réponse.

### Chaîne de requête partielle

```
searchFields=business_title&$select=business_title&search=*
```

Voici la même requête avec plusieurs champs dans une liste de valeurs séparées par des virgules.

```
search=*&searchFields=business_title, posting_type&$select=business_title, posting_type
```

## URL complète

```
https://azs-playground.search.windows.net/indexes/nycjobs/docs?api-version=2020-06-30&count=true&searchFields=business_title&$select=business_title&search=*
```

La réponse pour cette requête doit ressembler à la capture d'écran suivante.

```
1 {
2     "@odata.context": "https://azs-playground.search.windows.net/indexes('nycjobs')/$metadata#docs(business_title)",
3     "@odata.count": 2802,
4     "value": [
5         {
6             "@search.score": 1,
7             "business_title": "Business Analyst"
8         },
9         {
10            "@search.score": 1,
11            "business_title": "Limited Alteration Application (LAA) Inspector"
12        },
13        {
14            "@search.score": 1,
15            "business_title": "Deputy Director, Customer Service"
16        },
17        {
18            "@search.score": 1,
19            "business_title": "Borough Safety Specialist"
20        }
],
```

Vous avez peut-être remarqué le score de recherche dans la réponse. Des scores uniformes de 1 sont obtenus en l'absence de classement, soit parce que la recherche n'était pas une recherche en texte intégral, soit parce qu'aucun critère n'a été appliqué. Pour la recherche de valeur Null sans aucun critère, les lignes sont renvoyées dans un ordre arbitraire. Si vous incluez des critères réels, vous constaterez que les scores de recherche deviendront des valeurs significatives.

## Exemple 2 : Recherche par ID

Cet exemple est un peu atypique, mais lors de l'évaluation des comportements de recherche, vous souhaiterez peut-être inspecter l'ensemble du contenu d'un document spécifique afin de comprendre pourquoi il a été inclus dans les résultats ou exclu de ces derniers. Pour obtenir un document unique dans sa totalité, utilisez une [opération de recherche](#) afin de transmettre l'ID du document.

Tous les documents ont un identificateur unique. Pour tester la syntaxe d'une requête de recherche, retournez tout d'abord une liste d'ID de document afin d'en trouver un à utiliser. Pour NYC Jobs, les identificateurs sont stockés dans le champ `id`.

```
https://azs-playground.search.windows.net/indexes/nycjobs/docs?api-version=2020-06-30&count=true&searchFields=id&$select=id&search=*
```

L'exemple suivant est une requête de recherche qui retourne un document spécifique basé sur `id` « 9E1E3AF9-0660-4E00-AF51-9B654925A2D5 », qui est apparu en premier dans la réponse précédente. La requête suivante retourne le document entier, et pas seulement certains champs.

```
https://azs-playground.search.windows.net/indexes/nycjobs/docs/9E1E3AF9-0660-4E00-AF51-9B654925A2D5?api-version=2020-06-30&count=true&search=*
```

## Exemple 3 : Requêtes de filtre

La [syntaxe de filtre](#) est une expression OData que vous pouvez utiliser avec une **recherche** ou de façon autonome. Un filtre autonome, dépourvu de paramètre de recherche, est utile quand l'expression de filtre est en mesure de qualifier complètement les documents d'intérêt. En l'absence d'une chaîne de requête, il n'y a ni analyse lexicale ou linguistique, ni scoring (tous les scores sont égaux à 1), ni classement. Vous pouvez remarquer que la chaîne de recherche est vide.

```
POST /indexes/nycjobs/docs/search?api-version=2020-06-30
{
  "search": "",
  "filter": "salary_frequency eq 'Annual' and salary_range_from gt 90000",
  "select": "job_id, business_title, agency, salary_range_from",
  "count": "true"
}
```

Utilisés conjointement, le filtre est d'abord appliqué à la totalité de l'index et la recherche est effectuée sur les résultats du filtre. Les filtres peuvent donc être utiles pour améliorer les performances des requêtes, puisqu'ils limitent le nombre de documents que devra traiter la requête de recherche.

```
1 {
2   "@odata.context": "https://azs-playground.search.windows.net/indexes('nycjobs')/$metadata#docs(job_id,business_title
3   ,agency,salary_range_from)",
4   "@odata.count": 96,
5   "value": [
6     {
7       "@search.score": 1,
8       "job_id": "156545",
9       "business_title": "Construction Deputy Director",
10      "agency": "NYC HOUSING AUTHORITY",
11      "salary_range_from": 105000
12    },
13    {
14      "@search.score": 1,
15      "job_id": "175003",
16      "business_title": "Senior Java Developer",
17      "agency": "DEPARTMENT OF CORRECTION",
18      "salary_range_from": 100000
19    },
20    {
21      "@search.score": 1,
22      "job_id": "176839",
23      "business_title": "Internal Monitor",
24      "agency": "ADMIN FOR CHILDREN'S SVCS",
25      "salary_range_from": 110000
26    }
27  ]
28}
```

Si vous souhaitez tester ce type de requête dans Postman à l'aide de GET, vous pouvez coller la chaîne suivante :

```
https://azs-playground.search.windows.net/indexes/nycjobs/docs?api-version=2020-06-
30&$count=true&$select=job_id,business_title,agency,salary_range_from&&search=&&filter=salary_frequency eq
'Annual' and salary_range_from gt 90000
```

Une autre méthode efficace pour combiner un filtre et une recherche consiste à utiliser `search.ismatch()` dans une expression de filtre, dans laquelle vous pouvez utiliser une requête de recherche dans le filtre. Cette expression de filtre utilise un caractère générique sur *plan* afin de sélectionner les documents dont le champ `business_title` contient les termes `plan`, `planificateur`, `planification`, et ainsi de suite.

```
https://azs-playground.search.windows.net/indexes/nycjobs/docs?api-version=2020-06-
30&$count=true&$select=job_id,business_title,agency&&search=&&filter=search.ismatch('plan*',
'business_title', 'full', 'any')
```

Pour plus d'informations sur la fonction, consultez la [description de la fonction search.ismatch dans les exemples de filtre](#).

## Exemple 4 : Filtres de plage

Le filtrage de plage est pris en charge par le biais des expressions `$filter` pour n'importe quel type de données.

Les exemples ci-après effectuent des recherches sur des champs numériques et de chaîne.

Les types de données sont importants dans les filtres de plage et fonctionnent mieux lorsque les données numériques se trouvent dans des champs numériques, et les données de chaîne dans des champs de chaîne. L'utilisation de données numériques dans les champs de chaîne n'est pas adaptée aux plages, car les chaînes numériques ne sont pas comparables dans la Recherche cognitive Azure.

Les exemples ci-après sont fournis au format POST à des fins de lisibilité (plage numérique, suivie d'une plage de texte) :

```
POST /indexes/nycjobs/docs/search?api-version=2020-06-30
{
  "search": "",
  "filter": "num_of_positions ge 5 and num_of_positions lt 10",
  "select": "job_id, business_title, num_of_positions, agency",
  "orderby": "agency",
  "count": "true"
}
```

```
1 { "@odata.context": "https://azs-playground.search.windows.net/indexes('nycjobs')/$metadata#docs(job_id,business_title ,num_of_positions,agency)", "@odata.count": 105, "value": [
2   {
3     "@search.score": 1,
4     "job_id": "191070",
5     "business_title": "Agency Attorney",
6     "num_of_positions": 5,
7     "agency": "ADMIN FOR CHILDREN'S SVCS"
8   },
9   {
10    "@search.score": 1,
11    "job_id": "191070",
12    "business_title": "Agency Attorney",
13    "num_of_positions": 5,
14    "agency": "ADMIN FOR CHILDREN'S SVCS"
15  },
16  {
17    "@search.score": 1,
18    "job_id": "182154",
19    "business_title": "Investigator Level 1",
20    "num_of_positions": 8,
21    "agency": "CIVILIAN COMPLAINT REVIEW BD"
22  },
23  {
24    "@search.score": 1,
25    "job_id": "182154",
26    "business_title": "Investigator Level 1",
27    "num_of_positions": 8,
28    "agency": "CIVILIAN COMPLAINT REVIEW BD"
29  }
30]}
```

```
POST /indexes/nycjobs/docs/search?api-version=2020-06-30
{
  "search": "",
  "filter": "business_title ge 'A*' and business_title lt 'C**'",
  "select": "job_id, business_title, agency",
  "orderby": "business_title",
  "count": "true"
}
```

```

1  {
2      "@odata.context": "https://azs-playground.search.windows.net/indexes('nycjobs')/$metadata#docs(job_id,business_title
3          ,agency)",
4      "@odata.count": 377,
5      "value": [
6          {
7              "@search.score": 1,
8              "job_id": "187046",
9              "business_title": "ADMINISTRATIVE ACCOUNTANT - LEVEL I",
10             "agency": "DEPARTMENT OF SANITATION"
11         },
12         {
13             "@search.score": 1,
14             "job_id": "187046",
15             "business_title": "ADMINISTRATIVE ACCOUNTANT - LEVEL I",
16             "agency": "DEPARTMENT OF SANITATION"
17         },
18         {
19             "@search.score": 1,
20             "job_id": "123135",
21             "business_title": "ADMINISTRATIVE LAW JUDGE",
22             "agency": "TAX COMMISSION"
23     },

```

Vous pouvez également tester ces filtres dans Postman à l'aide de GET :

```
https://azs-playground.search.windows.net/indexes/nycjobs/docs?api-version=2020-06-30&search=&$filter=num_of_positions ge 5 and num_of_positions lt 10&$select=job_id, business_title, num_of_positions, agency&$orderby=agency&$count=true
```

```
https://azs-playground.search.windows.net/indexes/nycjobs/docs?api-version=2020-06-30&search=&$filter=business_title ge 'A*' and business_title lt 'C*&$select=job_id, business_title, agency&$orderby=business_title&$count=true
```

#### NOTE

L'utilisation de facettes sur des plages de valeurs est une condition d'application de recherche courante. Pour plus d'informations et d'exemples sur la génération de filtres pour les structures de navigation à facettes, consultez la section « [Filtrer sur une plage de valeurs](#) » de l'article [Implémentation de la navigation à facettes](#).

## Exemple 5 : Recherche basée sur la localisation

L'exemple d'index inclut un champ geo\_location avec des coordonnées de latitude et de longitude. Cet exemple utilise la [fonction geo.distance](#) qui applique un filtre sur les documents situés à une distance arbitraire (en kilomètres) d'un point de départ que vous spécifiez. Vous pouvez ajuster la dernière valeur de la requête (4) pour réduire ou étendre la surface de la requête.

L'exemple ci-après est fourni au format POST à des fins de lisibilité :

```
POST /indexes/nycjobs/docs/search?api-version=2020-06-30
{
    "search": "",
    "filter": "geo.distance(geo_location, geography'POINT(-74.11734 40.634384)') le 4",
    "select": "job_id, business_title, work_location",
    "count": "true"
}
```

Pour améliorer la lisibilité des résultats de la recherche, ces derniers sont tronqués afin d'inclure un ID de travail, un poste et le lieu de travail. Les coordonnées de départ ont été obtenues à partir d'un document aléatoire dans l'index (dans le cas présent, pour le lieu de travail Staten Island).

Vous pouvez également tester cette recherche dans Postman à l'aide de GET :

```
https://azs-playground.search.windows.net/indexes/nycjobs/docs?api-version=2020-06-  
30&$count=true&search=&$select=job_id, business_title, work_location&$filter=geo.distance(geo_location,  
geography'POINT(-74.11734 40.634384)') le 4
```

## Exemple 6 : Précision de la recherche

Les requêtes de termes portent sur des termes uniques, pouvant être nombreux, qui sont évalués de manière indépendante. Les requêtes d'expressions sont placées entre guillemets et évaluées comme une chaîne textuelle. La précision de la correspondance est contrôlée par des opérateurs et searchMode.

Exemple 1 : `&search=fire` retourne 150 résultats, où toutes les correspondances contiennent le mot fire quelque part dans le document.

```
https://azs-playground.search.windows.net/indexes/nycjobs/docs?api-version=2020-06-  
30&$count=true&search=fire
```

Exemple 2 : `&search=fire department` retourne 2002 résultats. Des correspondances sont retournées pour les documents contenant fire ou department.

```
https://azs-playground.search.windows.net/indexes/nycjobs/docs?api-version=2020-06-  
30&$count=true&search=fire department
```

Exemple 3 : `&search="fire department"` retourne 82 résultats. La chaîne étant entre guillemets, il s'agit d'une recherche textuelle sur les deux termes, et des correspondances sont trouvées sur les termes tokenisés dans l'index constitués des termes combinés. Cela explique pourquoi une recherche comme `search=+fire +department` n'est pas équivalente. Les deux termes sont obligatoires, mais sont analysés de manière indépendante.

```
https://azs-playground.search.windows.net/indexes/nycjobs/docs?api-version=2020-06-  
30&$count=true&search="fire department"
```

## Exemple 7 : Opérateurs booléens avec searchMode

La syntaxe simple prend en charge les opérateurs booléens sous la forme de caractères (`+`, `-`, `|`). Le paramètre searchMode indique le compromis à faire entre précision et rappel, `searchMode=any` favorisant le rappel (la mise en correspondance sur n'importe quel critère qualifie un document pour le jeu de résultats), et `searchMode=all` favorisant la précision (tous les critères doivent être mis en correspondance). La valeur par défaut est `searchMode=any`, ce qui peut être déroutant si vous placez plusieurs opérateurs dans une requête et que vous obtenez des résultats moins affinés que prévu. Cela est particulièrement vrai avec NOT, où les résultats incluent tous les documents « ne contenant pas » un terme spécifique.

Avec le searchMode par défaut (any), 2800 documents sont retournés : ceux contenant le terme en plusieurs parties « fire department », plus tous les documents ne contenant pas le terme « Metrotech Center ».

```
https://azs-playground.search.windows.net/indexes/nycjobs/docs?api-version=2020-06-  
30&$count=true&searchMode=any&search="fire department" -"Metrotech Center"
```

```

1 {  

2   "@odata.context": "https://azs-playground.search.windows.net/indexes('nycjobs')/$metadata#docs",  

3   "@odata.count": 2800,  

4   "value": [  

5     {  

6       "@search.score": 0.67712706,  

7       "id": "9896F5B1-1504-43F3-A937-C54465D4B168",  

8       "job_id": "180260",  

9       "agency": "FIRE DEPARTMENT",  

10      "posting_type": "External",  

11      "num_of_positions": 2,  

12      "business_title": "FIELD DESKTOP ASSOCIATE",  

13      "civil_service_title": "COMPUTER ASSOC (SOFTWARE)",  

14      "title_code_no": "13631",  

15      "level": "01",  

16      "salary_range_from": 58721,  

17      "salary_range_to": 81405,  

18      "salary_frequency": "Annual",  

19      "work_location": "5209 5Th Ave., Brooklyn",

```

Affecter la valeur `all` à `searchMode` applique un effet cumulé aux critères et retourne un jeu de résultats plus petit (21 documents) constitué de documents contenant l'expression entière « fire department », moins les emplois à l'adresse de Metrotech Center.

```
https://azs-playground.search.windows.net/indexes/nycjobs/docs?api-version=2020-06-30&$count=true&searchMode=all&search="fire department" -"Metrotech Center"
```

```

1 {  

2   "@odata.context": "https://azs-playground.search.windows.net/indexes('nycjobs')/$metadata#docs",  

3   "@odata.count": 21,  

4   "value": [  

5     {  

6       "@search.score": 0.67712706,  

7       "id": "9896F5B1-1504-43F3-A937-C54465D4B168",  

8       "job_id": "180260",  

9       "agency": "FIRE DEPARTMENT",  

10      "posting_type": "External",  

11      "num_of_positions": 2,  

12      "business_title": "FIELD DESKTOP ASSOCIATE",  

13      "civil_service_title": "COMPUTER ASSOC (SOFTWARE)",  

14      "title code no": "13631",

```

## Exemple 8 : Structuration des résultats

Plusieurs paramètres contrôlent quels champs figurent dans les résultats de la recherche, le nombre de documents retournés dans chaque lot, ainsi que l'ordre de tri. Cet exemple reprend quelques-uns des exemples précédents, mais limite les résultats à des champs spécifiques à l'aide de l'instruction `$select` et de critères de recherche textuelle, retournant 82 correspondances.

```
https://azs-playground.search.windows.net/indexes/nycjobs/docs?api-version=2020-06-30&$count=true&$select=job_id,agency,business_title,civil_service_title,work_location,job_description&search="fire department"
```

En ajoutant à l'exemple précédent, vous pouvez trier par titre. Ce tri fonctionne car `civil_service_title` est *trieable* dans l'index.

```
https://azs-playground.search.windows.net/indexes/nycjobs/docs?api-version=2020-06-30&$count=true&$select=job_id,agency,business_title,civil_service_title,work_location,job_description&search="fire department"&$orderby=civil_service_title
```

La pagination des résultats est implémentée à l'aide du paramètre `$top`, retournant ici les cinq premiers documents :

```
https://azs-playground.search.windows.net/indexes/nycjobs/docs?api-version=2020-06-30&$count=true&$select=job_id,agency,business_title,civil_service_title,work_location,job_description&search="fire department"&$orderby=civil_service_title&$top=5&$skip=0
```

Pour obtenir les cinq suivants, ignorez le premier lot :

```
https://azs-playground.search.windows.net/indexes/nycjobs/docs?api-version=2020-06-30&$count=true&$select=job_id,agency,business_title,civil_service_title,work_location,job_description&search="fire department"&$orderby=civil_service_title&$top=5&$skip=5
```

## Étapes suivantes

Essayez de spécifier des requêtes dans votre code. Les liens suivants expliquent comment configurer des requêtes de recherche pour .NET et l'API REST à l'aide de la syntaxe simple par défaut.

- [Interroger un index à l'aide du SDK .NET](#)
- [Interroger un index à l'aide de l'API REST](#)

Vous trouverez des informations de référence supplémentaires sur la syntaxe et sur l'architecture de requête, ainsi que des exemples, en cliquant sur les liens suivants :

- [Exemples de syntaxe de requête Lucene pour créer des requêtes avancées](#)
- [Fonctionnement de la recherche en texte intégral dans la Recherche cognitive Azure](#)
- [Syntaxe de requête simple](#)
- [Requête complète Lucene](#)
- [Syntaxe de filtre \(Filter\) et de tri \(Orderby\)](#)

# Utiliser la syntaxe de recherche Lucene « complète » (requêtes avancées dans Recherche cognitive Azure)

04/10/2020 • 20 minutes to read • [Edit Online](#)

Lors de la construction de requêtes pour Recherche cognitive Azure, vous pouvez remplacer l'[analyseur de requêtes simple](#) par défaut par l'[analyseur de requêtes Lucene dans Recherche cognitive Azure](#), plus vaste, afin de formuler des définitions de requêtes spécialisées et avancées.

L'analyseur Lucene prend en charge des constructions de requêtes complexes, telles que des requêtes portant sur des champs, la recherche approximative, la recherche par caractères génériques infixes et suffixes, la recherche de proximité, la promotion de termes et la recherche d'expression régulière. Cette fonctionnalité plus puissante nécessite des capacités de traitement supplémentaires et peut donc entraîner des temps d'exécution un peu plus longs. Dans cet article, vous pouvez parcourir des exemples décrivant les opérations de requête disponibles lors de l'utilisation de la syntaxe complète.

## NOTE

Une grande partie des constructions de requêtes spécialisées activées par le biais de la syntaxe de requête Lucene complète ne sont pas soumises à une [analyse du texte](#), ce qui peut être surprenant si vous prévoyez une recherche de radical ou une lemmatisation. L'analyse lexicale est effectuée uniquement sur des termes complets (requête sur un terme ou une expression). Les types de requête avec des termes incomplets (requête de préfixe, de caractère générique, d'expression régulière, partielle) sont ajoutés directement à l'arborescence de requête, en ignorant la phase d'analyse. La seule transformation effectuée sur les termes de requête partiels est l'utilisation de minuscules.

## Formuler des requêtes dans Postman

Les exemples suivants utilisent un index de recherche NYC Jobs composé de postes à pourvoir sur la base d'un jeu de données fourni par l'initiative [City of New York OpenData](#). Ces données ne doivent pas être considérées comme étant à jour ou complètes. L'index se trouve sur un service de bac à sable fourni par Microsoft, ce qui signifie que vous n'avez pas besoin d'abonnement Azure ni de Recherche cognitive Azure pour essayer ces requêtes.

En revanche, vous avez besoin de Postman ou d'un outil équivalent pour émettre la requête HTTP sur GET. Pour plus d'informations, consultez l'article indiquant comment [explorer avec les clients REST](#).

### Définir l'en-tête de requête

1. Dans l'en-tête de requête, affectez la valeur `application/json` à Content-Type.
2. Ajoutez un **api-key** et affectez-lui cette chaîne : `252044BE3886FE4A8E3BAA4F595114BB`. Il s'agit d'une clé de requête pour le service de recherche de bac à sable qui héberge l'index NYC Jobs.

Une fois que vous avez spécifié l'en-tête de requête, vous pouvez le réutiliser pour toutes les requêtes dans cet article, en remplaçant uniquement la chaîne `search=`.

NYCjobs

GET Enter request URL Params Send Save

Headers (2)

Key	Value	Description
Content-Type	application/json	
api-key	252044BE3886FE4A8E3BAA4F595114BB	This is a query key for a sandbox Azure Search service used for demos

## Définir l'URL de requête

La requête est une commande GET accompagnée d'une URL contenant le point de terminaison de Recherche cognitive Azure et la chaîne de recherche.

GET https://azs-playground.search.windows.net/indexes/nycjobs/docs?api-version=2020-06-30&search=\* Params Send

L'URL est composée des éléments suivants :

- `https://azs-playground.search.windows.net/` est un service de recherche de bac à sable tenu à jour par l'équipe de développement de Recherche cognitive Azure.
- `indexes/nycjobs/` est l'index NYC Jobs dans la collection d'index de ce service. Le nom du service et l'index sont tous deux obligatoires dans la requête.
- `docs` est la collection de documents contenant tout le contenu disponible pour la recherche. La clé d'API de requête fournie dans l'en-tête de requête fonctionne uniquement sur les opérations de lecture ciblant la collection de documents.
- `api-version=2020-06-30` définit la version de l'API, qui est un paramètre requis dans chaque requête.
- `search=*` est la chaîne de requête qui, dans la requête initiale, est nulle, ce qui retourne les 50 premiers résultats (par défaut).

## Envoyer votre première requête

En guise d'étape de vérification, collez la requête suivante dans GET et cliquez sur **Envoyer**. Les résultats sont renvoyés sous forme de documents JSON détaillés. Des documents entiers sont renvoyés, ce qui vous permet de voir tous les champs et toutes les valeurs.

Collez cette URL dans un client REST comme étape de validation et pour afficher la structure du document.

```
https://azs-playground.search.windows.net/indexes/nycjobs/docs?api-version=2020-06-30&$count=true&search=*
```

La chaîne de requête `search=*`, est une recherche non spécifiée équivalente à une recherche nulle ou vide. C'est la recherche la plus simple disponible.

Si vous le souhaitez, vous pouvez ajouter `$count=true` à l'URL pour retourner le nombre de documents correspondant aux critères de recherche. Une chaîne de recherche vide correspond à tous les documents figurant dans l'index (environ 2800 dans le cas de NYC Jobs).

## Appel de l'analyse Lucene complète

Ajoutez `queryType=full` pour appeler la syntaxe de requête complète et substituer la syntaxe de requête simple par défaut.

```
https://azs-playground.search.windows.net/indexes/nycjobs/docs?api-version=2020-06-30&queryType=full&search=*
```

Tous les exemples de cet article spécifient le paramètre de requête `queryType=full`, ce qui indique la syntaxe complète est traitée par l'analyseur de requêtes Lucene.

## Exemple 1 : Requête portée sur une liste de champs

Ce premier exemple n'est pas propre à Lucene, mais nous permet d'introduire le premier concept de requête fondamental : l'étendue d'un champ. Cet exemple limite l'exécution de la requête et la réponse à quelques champs spécifiques. Lors de l'utilisation de l'outil Postman ou Explorateur de recherche, il est important de connaître la structure d'une réponse JSON accessible en lecture.

Par souci de concision, la requête cible uniquement le champ `business_title` et spécifie que seuls les titres de fonctions sont retournés. Le paramètre `searchFields` restreint l'exécution de la requête au champ `business_title`, et le paramètre `select` spécifie les champs inclus dans la réponse.

### Expression de recherche

```
&search=*&searchFields=business_title&$select=business_title
```

Voici la même requête avec plusieurs champs dans une liste de valeurs séparées par des virgules.

```
search=*&searchFields=business_title, posting_type&$select=business_title, posting_type
```

Les espaces après les virgules sont facultatifs.

#### TIP

Lorsque vous utilisez l'API REST à partir du code de votre application, n'oubliez pas les paramètres d'encodage d'URL comme `$select` et `searchFields`.

## URL complète

```
https://azs-playground.search.windows.net/indexes/nycjobs/docs?api-version=2020-06-30&queryType=full&$count=true&search=*&searchFields=business_title&$select=business_title
```

La réponse pour cette requête doit ressembler à la capture d'écran suivante.

The screenshot shows the Postman interface with the following details:

- Body**: The tab is selected.
- Headers (13)**: Shows various headers including Content-Type, User-Agent, and X-Request-Id.
- Status: 200 OK Time: 262 ms**: Response status and time taken.
- JSON**: The response is displayed in JSON format.
- Response Body (Pretty)**:

```
1  {
2    "@odata.context": "https://azs-playground.search.windows.net/indexes('nycjobs')/$metadata#docs(business_title)",
3    "@odata.count": 2802,
4    "value": [
5      {
6        "@search.score": 1,
7        "business_title": "Business Analyst"
8      },
9      {
10        "@search.score": 1,
11        "business_title": "Limited Alteration Application (LAA) Inspector"
12      },
13      {
14        "@search.score": 1,
15        "business_title": "Deputy Director, Customer Service"
16      },
17      {
18        "@search.score": 1,
19        "business_title": "Borough Safety Specialist"
20      }
]
```

Vous avez peut-être remarqué le score de recherche dans la réponse. Des scores uniformes de 1 sont obtenus en l'absence de classement, soit parce que la recherche n'était pas une recherche en texte intégral, soit parce

qu'aucun critère n'a été appliqué. Pour la recherche de valeur Null sans aucun critère, les lignes sont renvoyées dans un ordre arbitraire. Si vous incluez des critères de recherche réels, vous constaterez que les scores de recherche deviendront des valeurs significatives.

## Exemple 2 : Recherche par champ

La syntaxe Lucene complète permet de restreindre des expressions de recherche individuelles à un champ spécifique. Cet exemple recherche les titres de fonctions contenant le terme « senior » mais pas « junior ».

### Expression de recherche

```
$select=business_title&search=business_title:(senior NOT junior)
```

Voici la même requête avec plusieurs champs.

```
$select=business_title, posting_type&search=business_title:(senior NOT junior) AND posting_type:external
```

### URL complète

```
https://azs-playground.search.windows.net/indexes/nycjobs/docs?api-version=2020-06-30&queryType=full&$count=true&$select=business_title&search=business_title:(senior NOT junior)
```

```
1 {  
2   "@odata.context": "https://azs-playground.search.windows.net/indexes('nycjobs')/$metadata#docs(business_title)",  
3   "@odata.count": 165,  
4   "value": [  
5     {  
6       "@search.score": 2.6672912,  
7       "business_title": "Senior Estimator"  
8     },  
9     {  
10       "@search.score": 2.465274,  
11       "business_title": "Senior Auditor"  
12     },  
13     {  
14       "@search.score": 2.3447099,  
15       "business_title": "Senior Developer"  
16     },
```

Vous pouvez définir une opération de recherche par champ avec la syntaxe **fieldName:searchExpression**, où l'expression de recherche peut être un mot ou une phrase, ou une expression plus complexe entre parenthèses, éventuellement avec des opérateurs booléens. Voici quelques exemples :

- `business_title:(senior NOT junior)`
- `state:(“New York” OR “New Jersey”)`
- `business_title:(senior NOT junior) AND posting_type:external`

Veillez à placer les chaînes entre guillemets si vous souhaitez que les deux chaînes soient évaluées comme une seule entité, comme ici où deux emplacements distincts sont recherchés dans le champ `state`. Vérifiez également que l'opérateur est en majuscules, comme c'est le cas ici avec NOT et AND.

Le champ spécifié dans **fieldName:searchExpression** doit être un champ pouvant faire l'objet d'une recherche. Pour plus d'informations sur l'utilisation des attributs d'index dans les définitions de champs, consultez [Créer un index \(API REST Recherche cognitive Azure\)](#).

## NOTE

Dans l'exemple ci-dessus, nous n'avons pas eu besoin d'utiliser le paramètre `searchFields` car chaque partie de la requête comporte un nom de champ explicitement spécifié. Cependant, vous pouvez toujours utiliser le paramètre `searchFields` si vous voulez exécuter une requête où certaines parties sont limitées à un champ spécifique, et le reste peut s'appliquer à plusieurs champs. Par exemple, la requête `search=business_title:(senior NOT junior) AND external&searchFields=posting_type` ne correspondrait à `senior NOT junior` qu'au niveau du champ `business_title`, alors qu'elle correspondrait au champ « externe » avec le champ `posting_type`. Le nom du champ fourni dans `fieldName:searchExpression` a toujours priorité sur le paramètre `searchFields`, c'est pourquoi dans cet exemple, nous n'avons pas besoin d'inclure `business_title` dans le paramètre `searchFields`.

## Exemple 3 : Recherche partielle

La syntaxe Lucene complète prend également en charge la recherche approximative, avec une mise en correspondance des termes qui ont une construction similaire. Pour effectuer une recherche partielle, ajoutez le signe tilde `~` à la fin d'un mot avec un paramètre facultatif, une valeur comprise entre 0 et 2, qui spécifie la distance de modification. Par exemple, `blue~` ou `blue~1` retournent blue, blues et glue.

### Expression de recherche

```
searchFields=business_title&$select=business_title&search=business_title:asosiate~
```

Les phrases ne sont pas prises en charge directement, mais vous pouvez spécifier une correspondance partielle pour différents éléments d'une phrase.

```
searchFields=business_title&$select=business_title&search=business_title:asosiate~ AND comm~
```

### URL complète

Cette requête recherche les postes à pourvoir contenant le terme « associate » (délibérément mal orthographié) :

```
https://azs-playground.search.windows.net/indexes/nycjobs/docs?api-version=2020-06-30&queryType=full&$count=true&searchFields=business_title&$select=business_title&search=business_title:asociate~
```

```
1  [
2    {
3      "@odata.context": "https://azs-playground.search.windows.net/indexes('nycjobs')/$metadata#docs(business_title)",
4      "@odata.count": 85,
5      "value": [
6        {
7          "@search.score": 2.9147263,
8          "business_title": "Community Associate"
9        },
10       {
11          "@search.score": 2.9147263,
12          "business_title": "Enforcement Associate"
13        },
14       {
15          "@search.score": 2.821047,
16          "business_title": "Clerical Associate"
17        }
18      ]
19    }
20  ]
```

## NOTE

Les requêtes approximatives ne sont pas [analysées](#). Les types de requête avec des termes incomplets (requête de préfixe, de caractère générique, d'expression régulière, partielle) sont ajoutés directement à l'arborescence de requête, en ignorant la phase d'analyse. La seule transformation effectuée sur les termes de requête incomplets est l'utilisation de minuscules.

## Exemple 4 : Recherche de proximité

Les recherches de proximité servent à rechercher des termes qui sont proches les uns des autres dans un document. Insérez un signe tilde « ~ » à la fin d'une expression, suivi du nombre de mots qui créent la limite de proximité. Par exemple, "hotel airport"~5 recherche les termes hotel et airport distants de cinq mots ou moins dans un document.

### Expression de recherche

```
searchFields=business_title&$select=business_title&search=business_title:%22senior%20analyst%22~1
```

### URL complète

Dans cette requête, on recherche les postes contenant le terme « senior analyst » où les deux mots sont séparés au plus par un mot :

```
https://azs-playground.search.windows.net/indexes/nycjobs/docs?api-version=2020-06-30&queryType=full&$count=true&searchFields=business_title&$select=business_title&search=business_title:%22senior%20analyst%22~1
```

```
1 + {
2   "@odata.context": "https://azs-playground.search.windows.net/indexes('nycjobs')/$metadata#docs(business_title)",
3   "@odata.count": 18,
4   "value": [
5     {
6       "@search.score": 4.174329,
7       "business_title": "Senior Analyst"
8     },
9     {
10       "@search.score": 4.174329,
11       "business_title": "Senior Analyst"
12     },
13     {
14       "@search.score": 4.090341,
15       "business_title": "Senior Analyst"
16     },
17   ]
}
```

Réessayez, mais en supprimant les mots entre « senior analyst ». Notez que huit documents sont retournés pour cette requête, par opposition à 10 pour la requête précédente.

```
https://azs-playground.search.windows.net/indexes/nycjobs/docs?api-version=2020-06-30&queryType=full&$count=true&searchFields=business_title&$select=business_title&search=business_title:%22senior%20analyst%22~0
```

## Exemple 5 : Promotion de termes

La promotion de termes signifie que vous pouvez accorder à un document un rang plus élevé s'il contient le terme promu, par rapport aux documents qui ne contiennent pas ce terme. Pour promouvoir un terme, utilisez le signe « ^ » avec un facteur de promotion (un nombre) à la fin du terme recherché.

### URL complètes

Dans cette requête « avant », on recherche les postes à pourvoir contenant le terme *computer analyst* et on constate l'absence de résultat avec les deux mots *computer* et *analyst*. Par contre, les postes *computer* figurent en haut des résultats.

```
https://azs-playground.search.windows.net/indexes/nycjobs/docs?api-version=2020-06-30&queryType=full&$count=true&searchFields=business_title&$select=business_title&search=business_title:computer%20analyst
```

```

1  {
2      "@odata.context": "https://azs-playground.search.windows.net/indexes('nycjobs')/$metadata#docs(business_title)",
3      "@odata.count": 336,
4      "value": [
5          {
6              "@search.score": 1.6003169,
7              "business_title": "Computer Associate"
8          },
9          {
10             "@search.score": 1.5311143,
11             "business_title": "Computer Associate"
12         },
13         {
14             "@search.score": 1.2886862,
15             "business_title": "Computer Associate (Software)"
16         },

```

Dans la requête « après », on répète la recherche, cette fois en promouvant les résultats contenant le terme *analyst* par rapport au terme *computer* si les deux mots ensemble n'existent pas.

```

https://azs-playground.search.windows.net/indexes/nycjobs/docs?api-version=2020-06-
30&queryType=full&$count=true&searchFields=business_title&$select=business_title&search=business_title:compu-
ter%20analyst%5e2

```

Une version plus lisible de la requête ci-dessus est `search=business_title:computer analyst^2`. Pour obtenir une requête exploitable, `^2` est encodé sous la forme `%5E2`, ce qui est plus difficile à voir.

```

1  {
2      "@odata.context": "https://azs-playground.search.windows.net/indexes('nycjobs')/$metadata#docs(business_title)",
3      "@odata.count": 336,
4      "value": [
5          {
6              "@search.score": 1.249817,
7              "business_title": "Analyst"
8          },
9          {
10             "@search.score": 1.2403976,
11             "business_title": "Computer Associate"
12         },
13         {
14             "@search.score": 1.1583551,
15             "business_title": "Computer Associate"
16         },

```

Il ne faut pas confondre la promotion de termes avec les profils de score, qui promeuvent certains champs plutôt que des termes spécifiques. L'exemple suivant permet d'illustrer les différences entre les deux.

Prenez un profil de score qui promeut les correspondances figurant dans un certain champ, par exemple **genre** dans l'exemple musicstoreindex. En utilisant la promotion de termes, vous pouvez promouvoir encore plus certains termes de recherche. Par exemple, "rock^2 electronic" promeut les documents qui contiennent les termes de recherche dans le champ **genre**, par rapport aux autres champs de recherche de l'index. Par ailleurs, les documents qui contiennent le terme de recherche « rock » bénéficient d'un classement plus élevé que ceux qui contiennent le terme de recherche « electronic » en raison de la valeur de promotion du terme (2).

Lors de la définition du niveau de facteur, plus le facteur de promotion est élevé, plus le terme est pertinent par rapport aux autres termes de recherche. Par défaut, le facteur de promotion est égal à 1. Ce facteur doit être positif, mais il peut être inférieur à 1 (par exemple 0,2).

## Exemple 6 : Expression régulière

Une recherche d'expression régulière trouve une correspondance en fonction du contenu placé entre des barres obliques « / », comme le décrit la [classe RegExp](#).

### Expression de recherche

```

searchFields=business_title&$select=business_title&search=business_title:(Sen|Jun)ior/

```

### URL complète

Dans cette requête, recherchez les postes à pourvoir contenant le terme « Senior » ou « Junior » :

```
search=business_title:/(Sen|Jun)ior/ .
```

```
https://azs-playground.search.windows.net/indexes/nycjobs/docs?api-version=2020-06-30&queryType=full&$count=true&searchFields=business_title&$select=business_title&search=business_title:/(Sen|Jun)ior/
```

```
1 + {
2     "@odata.context": "https://azs-playground.search.windows.net/indexes('nycjobs')/$metadata#docs(business_title)",
3     "@odata.count": 183,
4     "value": [
5         {
6             "@search.score": 1,
7             "business_title": "Senior Business Analyst"
8         },
9         {
10            "@search.score": 1,
11            "business_title": "Senior Software Developer"
12        },
13        {
14            "@search.score": 1,
15            "business_title": "Junior Claiming Analyst, Bureau of Budget and Revenue"
16        },
17    ]
18 }
```

#### NOTE

Les requêtes Regex ne sont pas [analysées](#). La seule transformation effectuée sur les termes de requête incomplets est l'utilisation de minuscules.

## Exemple 7 : Recherche par caractères génériques

Vous pouvez utiliser la syntaxe généralement reconnue pour effectuer des recherches avec plusieurs caractères génériques (\*) ou un caractère générique unique (?). Notez que l'Analyseur de requêtes Lucene prend en charge l'utilisation de ces symboles avec un terme unique, et non une expression.

#### Expression de recherche

```
searchFields=business_title&$select=business_title&search=business_title:prog*
```

#### URL complète

Dans cette requête, on recherche les postes à pourvoir qui contiennent le préfixe « prog », par exemple ceux contenant les termes « programmation » et « programmeur ». Vous ne pouvez pas utiliser un signe \* ou ? comme premier caractère d'une recherche.

```
https://azs-playground.search.windows.net/indexes/nycjobs/docs?api-version=2020-06-30&queryType=full&$count=true&searchFields=business_title&$select=business_title&search=business_title:prog*
```

```
1 + {
2     "@odata.context": "https://azs-playground.search.windows.net/indexes('nycjobs')/$metadata#docs(business_title)",
3     "@odata.count": 185,
4     "value": [
5         {
6             "@search.score": 1,
7             "business_title": "Deputy Program Manager"
8         },
9         {
10            "@search.score": 1,
11            "business_title": "MAINFRAME DB2 SYSTEM PROGRAMMER"
12        },
13        {
14            "@search.score": 1,
15            "business_title": "SB1 Data and Program Analyst"
16        },
17        {
18            "@search.score": 1,
19            "business_title": "PROGRAM MANAGER, CAPITAL ACCESS"
20        }
21    ]
22 }
```

#### NOTE

Les requêtes par caractères génériques ne sont pas [analysées](#). La seule transformation effectuée sur les termes de requête incomplets est l'utilisation de minuscules.

## Étapes suivantes

Essayez de spécifier l'Analyseur de requêtes Lucene dans votre code. Les liens suivants expliquent comment configurer des requêtes de recherche pour .NET et l'API REST. Ces liens utilisent la syntaxe simple par défaut. Vous devrez donc appliquer ce que vous avez appris dans cet article pour spécifier le `queryType`.

- [Interroger un index à l'aide du SDK .NET](#)
- [Interroger un index à l'aide de l'API REST](#)

Vous trouverez des informations de référence supplémentaires sur la syntaxe et sur l'architecture de requête, ainsi que des exemples, en cliquant sur les liens suivants :

- [Exemples de requête de syntaxe simple](#)
- [Fonctionnement de la recherche en texte intégral dans la Recherche cognitive Azure](#)
- [Syntaxe de requête simple](#)
- [Syntaxe de requête complète Lucene](#)

# Recherche de termes partiels et modèles avec des caractères spéciaux (caractère générique, expression régulière, modèles)

04/10/2020 • 21 minutes to read • [Edit Online](#)

Une *recherche de terme partiel* fait référence à des requêtes composées de fragments de termes où, au lieu d'un terme entier, vous pouvez avoir juste le début, le milieu ou la fin du terme (elle est parfois appelée requête de préfixe, d'infixe ou de suffixe). Une recherche de terme partiel peut inclure une combinaison de fragments, souvent avec des caractères spéciaux comme des tirets ou des barres obliques qui font partie de la chaîne de requête. Les cas d'usage courants incluent les parties d'un numéro de téléphone, une URL, des codes ou des mots composés avec trait d'union.

La recherche de terme partiel et les chaînes de requête qui comprennent des caractères spéciaux peuvent être problématiques si l'index n'a pas de jetons au format attendu. Pendant la **phase d'analyse lexicale** de l'indexation (en supposant une utilisation de l'analyseur standard par défaut), les caractères spéciaux sont ignorés, les mots composés sont fractionnés et l'espace blanc est supprimé ; ceci peut provoquer l'échec des requêtes quand aucune correspondance n'est trouvée. Par exemple, un numéro de téléphone comme `+1 (425) 703-6214` (segmenté en unités `"1"`, `"425"`, `"703"`, `"6214"`) n'apparaît pas dans une requête `"3-62"`, car ce contenu n'existe pas dans l'index en réalité.

La solution consiste à appeler pendant l'indexation un analyseur qui conserve une chaîne complète, y compris les espaces et les caractères spéciaux si nécessaire, afin que vous puissiez inclure les espaces et les caractères dans votre chaîne de requête. De même, le fait d'avoir une chaîne complète qui n'est pas tokenisée en parties plus petites autorise des critères spéciaux pour les requêtes « commence par » ou « se termine par », où le modèle que vous fournissez peut être évalué par rapport à un terme qui n'est pas transformé par l'analyse lexicale. La création d'un champ supplémentaire pour une chaîne intacte, ainsi que l'utilisation d'un analyseur à conservation du contenu qui émet des jetons de terme entier, est la solution pour les critères spéciaux et pour la correspondance sur les chaînes de requête qui incluent des caractères spéciaux.

## TIP

Si vous connaissez Postman et les API REST, [téléchargez la collection d'exemples de requêtes](#) pour interroger les termes partiels et les caractères spéciaux décrits dans cet article.

## Qu'est-ce que la recherche de terme partiel dans Recherche cognitive Azure ?

Recherche cognitive Azure recherche des termes tokenisés entiers dans l'index, et ne trouvera pas de correspondance sur un terme partiel, sauf si vous incluez des opérateurs génériques d'espace réservé (`*` et `?`) ou que vous mettez en forme la requête en tant qu'expression régulière. Les termes partiels sont spécifiés à l'aide des techniques suivantes :

- Les [requêtes d'expression régulière](#) peuvent être n'importe quelle expression régulière valide sous Apache Lucene.
- Les [opérateurs génériques avec correspondance de préfixe](#) font référence à un modèle généralement reconnu qui comprend le début d'un terme, suivi d'opérateurs de suffixe `*` ou `?`, comme pour la mise en correspondance de `search=cap*` sur « Cap'n Jack's Waterfront Inn » ou « Gacc Capital ». La mise en

correspondance de préfixe est prise en charge dans la syntaxe de requête Lucene simple et complète.

- La [mise en correspondance générique avec infix et suffixe](#) place les opérateurs `*` et `?` à l'intérieur ou au début d'un terme, et nécessite une syntaxe d'expression régulière (où l'expression est entourée de barres obliques). Par exemple, la chaîne de requête (`search=/.*numeric*/`) retourne les résultats sur « alphanumeric » et « alphanumerical » en tant que correspondances de suffixe et d'infixe.

Pour la recherche de terme partiel ou de modèle, et quelques autres formes de requête telles que la recherche approximative, les analyseurs ne sont pas utilisés au moment de la requête. Pour ces formes de requête, que l'analyseur détecte par la présence d'opérateurs et de délimiteurs, la chaîne de requête est passée au moteur sans analyse lexicale. Pour ces formes de requête, l'analyseur spécifié sur le champ est ignoré.

#### NOTE

Quand une chaîne de requête partielle inclut des caractères, tels que des barres obliques dans un fragment d'URL, vous devrez peut-être ajouter des caractères d'échappement. Dans JSON, une barre oblique `/` est placée dans une séquence d'échappement avec une barre oblique inverse `\`. Par conséquent, `search=/.*microsoft.com\azure\/*/` est la syntaxe pour le fragment d'URL « microsoft.com/azure/ ».

## Résolution des problèmes liés à la recherche partielle/de modèles

Quand vous devez effectuer une recherche sur des fragments, modèles ou caractères spéciaux, vous pouvez remplacer l'analyseur par défaut par un analyseur personnalisé qui fonctionne selon des règles de tokenisation plus simples, préservant la chaîne entière dans l'index. En prenant un peu de recul, l'approche ressemble à ceci :

- Définir un champ pour stocker une version intacte de la chaîne (en supposant que vous voulez du texte analysé et non analysé au moment de la requête)
- Évaluer et choisir parmi les différents analyseurs qui émettent des jetons au niveau de précision approprié
- Attribuer l'analyseur au champ
- Générer et tester l'index

#### TIP

L'évaluation des analyseurs est un processus itératif qui nécessite de fréquentes régénération de l'index. Vous pouvez faciliter cette étape en utilisant Postman, les API REST pour [créer un index](#), [supprimer un index](#), [charger des documents](#) et [rechercher des documents](#). Pour charger des documents, le corps de la requête doit contenir un petit jeu de données représentatif que vous souhaitez tester (par exemple, un champ avec des numéros de téléphone ou des codes de produit). Ces API étant dans la même collection Postman, vous pouvez effectuer rapidement ces étapes.

## Duplicier des champs pour différents scénarios

Les analyseurs déterminent comment les termes sont tokenisés dans un index. Les analyseurs étant affectés par champ, vous pouvez créer des champs dans votre index pour optimiser les différents scénarios. Par exemple, vous pouvez définir « featureCode » et « featureCodeRegex » pour prendre en charge une recherche en texte intégral normale sur le premier et des critères spéciaux avancés sur le second. Les analyseurs affectés à chaque champ déterminent comment le contenu de chaque champ est tokenisé dans l'index.

```
{
  "name": "featureCode",
  "type": "Edm.String",
  "retrievable": true,
  "searchable": true,
  "analyzer": null
},
{
  "name": "featureCodeRegex",
  "type": "Edm.String",
  "retrievable": true,
  "searchable": true,
  "analyzer": "my_custom_analyzer"
},
```

## Choisir un analyseur

Lors du choix d'un analyseur qui produit des jetons à terme entier, les analyseurs suivants sont des choix courants :

ANALYSEUR	COMPORTEMENTS
<a href="#">Analyseurs de langage</a>	Conserve les traits d'Union dans les mots composés, les chaînes, les mutations de voyelles et les formes verbales. Si les modèles de requête incluent des tirets, l'utilisation d'un analyseur de langage peut suffire.
<a href="#">keyword</a>	Le contenu du champ entier est segmenté en jetons en tant que terme unique.
<a href="#">whitespace</a>	Sépare sur les espaces blancs seulement. Les termes qui incluent des tirets ou d'autres caractères sont traités comme un jeton unique.
<a href="#">analyseur personnalisé</a>	<p>(recommandé) La création d'un analyseur personnalisé vous permet de spécifier à la fois le générateur de jetons et le filtre de jeton. Les analyseurs précédents doivent être utilisés tels quels. Un analyseur personnalisé vous permet de choisir les générateurs et les filtres de jetons à utiliser.</p> <p>Une combinaison recommandée est le <a href="#">générateur de jetons keyword</a> avec un <a href="#">filtre de jeton lowercase</a>. En soi, l'<a href="#">analyseur keyword</a> prédéfini ne met pas en minuscules les caractères majuscules, ce qui peut entraîner l'échec des requêtes. Un analyseur personnalisé vous donne un mécanisme pour ajouter le filtre de jeton lowercase.</p>

Si vous utilisez un outil de test de l'API Web tel que Postman, vous pouvez ajouter l'[appel REST de l'analyseur de test](#) pour inspecter la sortie segmentée en jetons.

Vous devez avoir un index rempli avec lequel travailler. À partir d'un index existant et d'un champ contenant des tirets ou des termes partiels, vous pouvez essayer différents analyseurs sur des termes spécifiques pour voir quels jetons sont émis.

1. Vérifiez d'abord l'analyseur Standard pour voir comment les termes sont tokenisés par défaut.

```
{  
  "text": "SVP10-NOR-00",  
  "analyzer": "standard"  
}
```

2. Évaluez la réponse pour voir comment le texte est segmenté en jetons dans l'index. Notez la manière dont chaque terme est en minuscules et segmenté. Seules les requêtes qui correspondent à ces jetons retourneront ce document dans les résultats. Une requête qui comprend « 10-NOR » échouera.

```
{  
  "tokens": [  
    {  
      "token": "svp10",  
      "startOffset": 0,  
      "endOffset": 5,  
      "position": 0  
    },  
    {  
      "token": "nor",  
      "startOffset": 6,  
      "endOffset": 9,  
      "position": 1  
    },  
    {  
      "token": "00",  
      "startOffset": 10,  
      "endOffset": 12,  
      "position": 2  
    }  
  ]  
}
```

3. Maintenant, modifiez la requête pour utiliser l'analyseur `whitespace` ou `keyword` :

```
{  
  "text": "SVP10-NOR-00",  
  "analyzer": "keyword"  
}
```

4. À présent, la réponse se compose d'un jeton unique, en majuscules, avec des tirets conservés en tant que partie de la chaîne. Si vous devez effectuer une recherche sur un modèle ou un terme partiel tel que « 10-NOR », le moteur de requête dispose maintenant de la base pour trouver une correspondance.

```
{  
  "tokens": [  
    {  
      "token": "SVP10-NOR-00",  
      "startOffset": 0,  
      "endOffset": 12,  
      "position": 0  
    }  
  ]  
}
```

## IMPORTANT

N'oubliez pas que les analyseurs de requêtes mettent souvent les termes en minuscules dans une expression de recherche lors de la création de l'arborescence de requêtes. Si vous utilisez un analyseur qui ne met pas les entrées de texte en minuscules durant l'indexation, et que vous n'obtenez pas les résultats attendus, cela peut être la raison. La solution consiste à ajouter un filtre de jeton lowercase, comme décrit dans la section « Utiliser des analyseurs personnalisés », ci-dessous.

# Configurer un analyseur

Que vous évaluiez des analyseurs ou que vous avanciez avec une configuration spécifique, vous devrez spécifier l'analyseur sur la définition de champ et éventuellement configurer l'analyseur lui-même si vous n'utilisez pas d'analyseur intégré. Lorsque vous changez d'analyseur, vous devez généralement reconstruire l'index (supprimer, recréer et recharger).

## Utiliser des analyseurs intégrés

Les analyseurs intégrés ou prédéfinis peuvent être spécifiés par leur nom dans la propriété `analyzer` d'une définition de champ, sans qu'une configuration supplémentaire ne soit requise dans l'index. L'exemple suivant montre comment définir l'analyseur `whitespace` sur un champ.

Pour d'autres scénarios et pour en savoir plus sur les autres analyseurs intégrés, consultez [Liste des analyseurs prédéfinis](#).

```
{
  "name": "phoneNumber",
  "type": "Edm.String",
  "key": false,
  "retrievable": true,
  "searchable": true,
  "analyzer": "whitespace"
}
```

## Utiliser des analyseurs personnalisés

Si vous utilisez un [analyseur personnalisé](#), définissez-le dans l'index avec une combinaison définie par l'utilisateur d'un générateur de jetons et d'un filtre de jeton, avec les paramètres de configuration possibles. Ensuite, référez-vous à la section [Utiliser des analyseurs personnalisés](#) pour obtenir des instructions détaillées.

Lorsque l'objectif est la segmentation du texte en termes entiers, un analyseur personnalisé qui se compose d'un **générateur de jetons keyword** et d'un **filtre de jeton lowercase** est recommandé.

- Le générateur de jetons keyword crée un jeton unique pour tout le contenu d'un champ.
- Le filtre de jeton lowercase transforme les lettres majuscules en caractères minuscules. Les analyseurs de requêtes mettent généralement en minuscules toutes les entrées de texte qui sont en majuscules. La mise en minuscules homogénéise les entrées avec les termes tokenisés.

L'exemple suivant illustre un analyseur personnalisé qui fournit le générateur de jetons keyword et un filtre de jetons lowercase.

```
{
  "fields": [
    {
      "name": "accountNumber",
      "analyzer": "myCustomAnalyzer",
      "type": "Edm.String",
      "searchable": true,
      "filterable": true,
      "retrievable": true,
      "sortable": false,
      "facetable": false
    }
  ],
  "analyzers": [
    {
      "@odata.type": "#Microsoft.Azure.Search.CustomAnalyzer",
      "name": "myCustomAnalyzer",
      "charFilters": [],
      "tokenizer": "keyword_v2",
      "tokenFilters": ["lowercase"]
    }
  ],
  "tokenizers": [],
  "charFilters": [],
  "tokenFilters": []
}
```

#### NOTE

Le générateur de jetons `keyword_v2` et le filtre de jeton `lowercase` sont connus du système et utilisent leurs configurations par défaut, c'est pourquoi vous pouvez les référencer par leur nom sans avoir à les définir au préalable.

## Générer et tester

Une fois que vous avez défini un index avec des analyseurs et des définitions de champ qui prennent en charge votre scénario, chargez des documents contenant des chaînes représentatives afin de pouvoir tester des requêtes de chaînes partielles.

Les sections précédentes ont expliqué la logique. Cette section parcourt chaque API que vous devez appeler lors du test de votre solution. Comme indiqué précédemment, si vous utilisez un outil de test web interactif comme Postman, vous pouvez parcourir ces tâches rapidement.

- [Supprimer l'index](#) supprime un index existant du même nom afin que vous puissiez le recréer.
- [Créer un index](#) crée la structure d'index sur votre service de recherche, notamment des définitions et des champs d'analyseur avec la spécification d'un analyseur.
- [Charger des documents](#) importe des documents ayant la même structure que votre index, ainsi que du contenu pouvant faire l'objet d'une recherche. Après cette étape, votre index est prêt à être interrogé ou testé.
- L'option [Tester l'analyseur](#) a été introduite dans [Choisir un analyseur](#). Testez certaines chaînes de votre index à l'aide de divers analyseurs pour comprendre comment les termes sont tokenisés.
- [Rechercher dans les documents](#) explique comment construire une demande de requête, en utilisant une [syntaxe simple](#) ou la [syntaxe Lucene complète](#) pour les expressions génériques et les expressions régulières.

Pour les requêtes de termes partiels, telles que l'interrogation « 3-6214 » pour trouver une correspondance

sur « + 1 (425) 703-6214 », vous pouvez utiliser la syntaxe simple : `search=3-6214&queryType=simple`.

Pour les requêtes d'infixe et de suffixe, telles que l'interrogation « num » ou « numérique pour trouver une correspondance sur « alphanumérique », utilisez la syntaxe Lucene complète et une expression régulière :

```
search=/.*num.*/&queryType=full
```

## Régler les performances des requêtes

Si vous implémentez la configuration recommandée qui comprend le générateur de jetons keyword\_v2 et le filtre de jeton lowercase, vous remarquerez peut-être une baisse des performances de requêtes en raison du traitement du filtre de jeton supplémentaire sur les jetons existants dans votre index.

L'exemple suivant ajoute un [EdgeNGramTokenFilter](#) pour que les correspondances de préfixe soient plus rapides. Des jetons supplémentaires sont générés pour les combinaisons de 2 à 25 caractères qui incluent des caractères : (pas seulement MS, MSF, MSFT, MSFT/, MSFT/S, MSFT/SQ, MSFT/SQL).

Comme vous pouvez l'imaginer, les résultats supplémentaires de segmentation du texte en unités lexicales entraîne un plus grand index. Si vous disposez d'une capacité suffisante pour accueillir l'index plus volumineux, cette approche offrant un temps de réponse plus rapide peut constituer une meilleure solution.

```
{
  "fields": [
    {
      "name": "accountNumber",
      "analyzer": "myCustomAnalyzer",
      "type": "Edm.String",
      "searchable": true,
      "filterable": true,
      "retrievable": true,
      "sortable": false,
      "facetable": false
    }
  ],
  "analyzers": [
    {
      "@odata.type": "#Microsoft.Azure.Search.CustomAnalyzer",
      "name": "myCustomAnalyzer",
      "charFilters": [],
      "tokenizer": "keyword_v2",
      "tokenFilters": ["lowercase", "my_edgeNGram"]
    }
  ],
  "tokenizers": [],
  "charFilters": [],
  "tokenFilters": [
    {
      "@odata.type": "#Microsoft.Azure.Search.EdgeNGramTokenFilterV2",
      "name": "my_edgeNGram",
      "minGram": 2,
      "maxGram": 25,
      "side": "front"
    }
  ]
}
```

## Étapes suivantes

Cet article explique comment les analyseurs contribuent aux problèmes de requête et à les résoudre. À l'étape suivante, examinez de plus près l'impact de l'analyseur sur l'indexation et le traitement des requêtes. En particulier, envisagez d'utiliser l'API Analyze Text pour retourner une sortie segmentée en jetons afin que vous puissiez voir exactement ce qu'un analyseur crée pour votre index.

- Analyseurs de langage
- Analyseurs pour le traitement de texte dans Recherche cognitive Azure
- API (REST) Analyze Text
- Fonctionnement de la recherche en texte intégral (architecture de requête)

# Recherche approximative pour corriger les fautes d'orthographe et de frappe

04/10/2020 • 13 minutes to read • [Edit Online](#)

Recherche cognitive Azure prend en charge la recherche approximative, type de requête qui pallie les fautes d'orthographe et les termes mal orthographiés dans la chaîne d'entrée. Pour ce faire, le service analyse les termes ayant une composition similaire. L'extension de la recherche pour couvrir les correspondances proches a pour effet de corriger automatiquement une faute de frappe quand l'écart n'est qu'un petit nombre de caractères mal placés.

## Qu'est-ce que la recherche approximative ?

Il s'agit d'un exercice d'expansion qui produit une correspondance avec les termes ayant une composition similaire. Quand une recherche approximative est spécifiée, le moteur génère un graphe (basé sur la [théorie de l'automate fini déterministe](#)) des termes composés de la même façon, pour l'ensemble des termes de la requête. Par exemple, si votre requête inclut les trois termes « university of washington », un graphe est créé pour chaque terme de la requête `search=university~ of~ washington~` (comme il n'y a pas de suppression des mots vides dans la recherche approximative, « of » obtient un graphe).

Le graphe se compose d'un maximum de 50 expansions, ou permutations, de chaque terme, capturant à la fois les variantes correctes et incorrectes dans le processus. Le moteur retourne ensuite les correspondances les plus pertinentes dans la réponse.

Pour un terme comme « university », le graphe peut comporter « unversty, universty, university, universe, inverse ». Tous les documents qui correspondent à ces termes du graphe sont inclus dans les résultats. Contrairement à d'autres requêtes qui analysent le texte pour gérer les différentes formes du même mot (« mice » et « mouse »), les comparaisons dans une requête approximative sont effectuées telles quelles sans aucune analyse linguistique du texte. Les termes « universe » et « inverse », qui sont sémantiquement différents, constituent des correspondances, car les différences syntaxiques sont minimales.

Une correspondance est établie si les différences sont limitées à, au plus, deux modifications, une modification étant un caractère inséré, supprimé, substitué ou transposé. L'algorithme de correction de chaîne qui spécifie la valeur différentielle est la métrique [Distance de Damerau-Levenshtein](#), décrite comme étant le nombre minimum d'opérations (insertions, suppressions, substitutions ou transpositions de deux caractères adjacents) nécessaires pour transformer une chaîne de caractères en une autre.

Dans Recherche cognitive Azure :

- La requête approximative s'applique aux termes entiers, mais vous pouvez prendre en charge des expressions par le biais de constructions ET. Par exemple, « Unviersté~ de~ « Wshington~ » correspondrait à « Université de Washington ».
- La distance par défaut d'une modification est 2. La valeur `~0` signifie qu'il n'y a pas d'expansion (seul le terme exact est considéré comme une correspondance), mais vous pouvez spécifier `~1` pour un degré de différence ou une modification.
- Une requête approximative peut étendre un terme jusqu'à 50 permutations supplémentaires. Cette limite n'est pas configurable, mais vous pouvez réduire efficacement le nombre d'expansions en diminuant la distance de modification à 1.
- Les réponses se composent de documents contenant une correspondance pertinente (jusqu'à 50).

Collectivement, les graphes sont soumis en tant que critères de correspondance par rapport aux jetons de l'index.

Comme vous pouvez l'imaginer, la recherche approximative est fondamentalement plus lente que les autres formes de requête. La taille et la complexité de votre index peuvent déterminer si les avantages sont suffisants pour compenser la latence de la réponse.

#### NOTE

La recherche approximative ayant tendance à être lente, il peut être utile d'examiner d'autres approches, telles que l'indexation N-gramme, avec sa progression de séquences de caractères courtes (séquences de deux et trois caractères pour les jetons bigrammes et trigrammes). En fonction de votre langue et de la surface d'exposition de la requête, l'approche N-gramme peut offrir de meilleures performances. Sachez toutefois que l'indexation N-gramme est très gourmande en stockage et génère des index beaucoup plus volumineux.

Une autre solution, que vous pouvez envisager si vous souhaitez gérer uniquement les cas les plus flagrants, est une [carte de synonymes](#). Par exemple, vous pouvez établir une correspondance entre « search » et « serach, serch, sarch » ou entre « retrieve » et « retreive ».

## Indexation pour la recherche approximative

Les analyseurs ne sont pas utilisés lors du traitement des requêtes pour créer un graphe d'expansions, mais cela ne signifie pas qu'ils doivent être ignorés dans les scénarios de recherche approximative. Après tout, les analyseurs sont utilisés pendant l'indexation pour créer les jetons vis-à-vis desquels la correspondance est effectuée, que la requête soit une forme libre, une recherche filtrée ou une recherche approximative avec un graphe comme entrée.

En général, quand vous affectez des analyseurs en fonction du champ, la décision d'affiner la chaîne d'analyse est basée sur le cas d'usage principal (un filtre ou une recherche en texte intégral) plutôt que sur des formes de requête spécialisées telles qu'une recherche approximative. Ainsi n'existe-t-il pas de recommandation d'analyseur spécifique pour la recherche approximative.

Toutefois, si les requêtes de test ne produisent pas les correspondances attendues, vous pouvez essayer de faire varier l'analyseur d'indexation, en le définissant sur un [analyseur linguistique](#), afin de voir si vous obtenez de meilleurs résultats. Certaines langues, en particulier celles qui ont des mutations de voyelle, peuvent tirer parti des formes fléchies et irrégulières générées par les processeurs de langage naturel Microsoft. Dans certains cas, l'utilisation de l'analyseur linguistique approprié peut s'avérer déterminante pour qu'un terme soit représenté sous forme de jetons d'une manière compatible avec la valeur fournie par l'utilisateur.

## Comment utiliser la recherche approximative

Les requêtes approximatives sont construites à l'aide de la syntaxe de requête Lucene complète, par le biais de l'appel de l'[analyseur de requêtes Lucene](#).

1. Définissez l'analyseur Lucene complet sur la requête (`queryType=full`).
2. Si vous le souhaitez, étendez la requête à des champs spécifiques, à l'aide de ce paramètre (`searchFields=<field1,field2>`).
3. Ajoutez l'opérateur tilde (`~`) à la fin du terme entier (`search=<string>~`).

Incluez un paramètre facultatif, un nombre compris entre 0 et 2 (valeur par défaut), si vous souhaitez spécifier la distance de modification (`~1`). Par exemple, « blue~ » ou « blue~1 » retournent « blue », « blues » et « glue ».

Dans Recherche cognitive Azure, en plus du terme et de la distance (maximum 2), il n'y a aucun paramètre supplémentaire à définir sur la requête.

#### NOTE

Pendant leur traitement, les requêtes approximatives ne sont pas soumises à une [analyse lexicale](#). L'entrée de la requête est ajoutée directement à l'arborescence de requête et développée pour créer un graphe de termes. La seule transformation effectuée est la mise en minuscules.

## Test de la recherche approximative

Pour les tests simples, nous vous recommandons d'utiliser l'[Explorateur de recherche](#) ou Postman afin d'effectuer une itération au sein d'une expression de requête. Les deux outils sont interactifs, ce qui signifie que vous pouvez rapidement parcourir plusieurs variantes d'un terme et évaluer les réponses retournées.

Quand les résultats sont ambigus, la [mise en surbrillance des correspondances](#) peut vous aider à identifier la correspondance dans la réponse.

#### NOTE

L'utilisation de la mise en surbrillance des correspondances pour identifier les correspondances approximatives a des limites et fonctionne uniquement pour la recherche approximative de base. Si votre index a des profils de scoring, ou si vous enrichissez la requête avec une syntaxe supplémentaire, la mise en surbrillance des correspondances risque d'échouer pour identifier la correspondance.

### Exemple 1 : Recherche approximative avec le terme exact

Supposons qu'un champ "Description" d'un document de recherche contient la chaîne suivante :

```
"Test queries with special characters, plus strings for MSFT, SQL and Java."
```

Commencez par une recherche approximative sur « special » et ajoutez la mise en surbrillance des correspondances au champ Description :

```
search=special~&highlight=Description
```

Dans la réponse, étant donné que vous avez ajouté la mise en surbrillance des correspondances, la mise en forme est appliquée à « special » en tant que terme correspondant.

```
@search.highlights": {  
  "Description": [  
    "Test queries with <em>special</em> characters, plus strings for MSFT, SQL and Java."  
  ]}
```

Renouvelez la demande en retirant quelques lettres (« pe ») de « special » afin qu'il soit mal orthographié :

```
search=scial~&highlight=Description
```

Jusqu'à présent, aucune modification n'est apportée à la réponse. À l'aide de la distance par défaut de 2 degrés, la suppression de deux caractères (« pe ») de « special » permet toujours une correspondance correcte sur ce terme.

```
@search.highlights": {  
  "Description": [  
    "Test queries with <em>special</em> characters, plus strings for MSFT, SQL and Java."  
  ]}
```

Pour effectuer une demande supplémentaire, modifiez davantage le terme de recherche en retirant un dernier

caractère, soit un total de trois suppressions (permettant de passer de « special » à « scal ») :

```
search=scal~&highlight=Description
```

Notez que la même réponse est retournée, mais à présent la correspondance approximative ne porte pas sur « special », mais sur « SQL ».

```
"@search.score": 0.4232868,  
"@search.highlights": {  
    "Description": [  
        "Mix of special characters, plus strings for MSFT, <em>SQL</em>, 2019, Linux, Java."  
    ]
```

L'intérêt de cet exemple développé est d'illustrer la clarté que la mise en surbrillance des correspondances peut apporter à des résultats ambigus. Dans tous les cas, le même document est retourné. Si vous aviez utilisé des ID de document pour vérifier une correspondance, vous auriez peut-être manqué le passage de « special » à « SQL ».

## Voir aussi

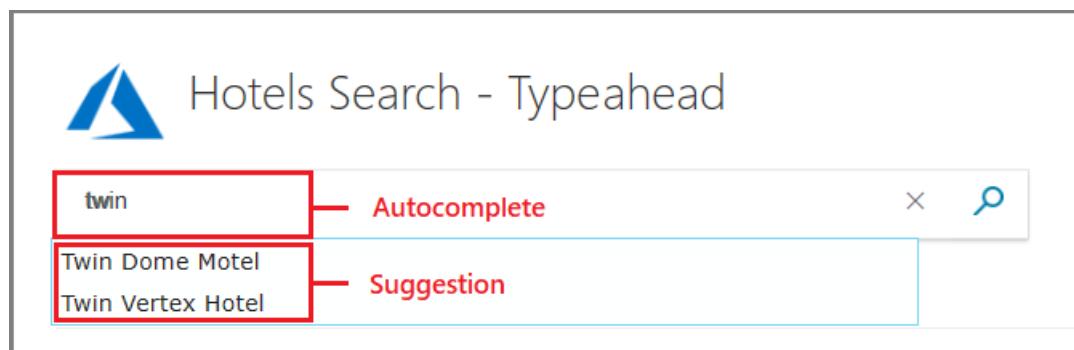
- [Fonctionnement de la recherche en texte intégral dans Recherche cognitive Azure \(architecture d'analyse de requête\)](#)
- [Navigateur de recherche](#)
- [Guide pratique pour interroger dans .NET](#)
- [Guide pratique pour interroger dans REST](#)

# Créer un suggesteur pour activer l'autocomplétion et les résultats suggérés dans une requête

04/10/2020 • 13 minutes to read • [Edit Online](#)

Dans la recherche cognitive Azure, la « search-as-you-type » (recherche pendant la saisie) est activée par le biais d'une construction de **suggesteur** ajoutée à un [index de recherche](#). Un suggesteur prend en charge deux expériences : l'*autocomplete*, qui complète une entrée partielle pour une recherche sur un terme entier, et les *suggestions*, qui invitent à cliquer sur une correspondance particulière. L'autocomplétion génère une requête. Les suggestions produisent un document correspondant.

Toutes deux sont illustrées par la capture d'écran suivante, extraite de [Créer votre première application en C#](#). L'autocomplétion anticipe un terme potentiel, en ajoutant « in » si vous avez tapé « tw ». Les suggestions sont des mini-résultats de recherche, où un champ comme Nom de l'hôtel représente un document de recherche d'hôtels correspondants à partir de l'index. Concernant les suggestions, vous pouvez couvrir n'importe quel champ fournissant des informations descriptives.



Vous pouvez utiliser ces fonctionnalités séparément ou ensemble. Pour implémenter ces comportements dans la Recherche cognitive Azure, il existe un composant de requête et d'index.

- Dans l'index, ajoutez un suggesteur à un index. Vous pouvez utiliser le portail, l'[API REST](#) ou le [kit de développement logiciel \(SDK\) .NET](#). La suite de cet article se concentre sur la création d'un suggesteur.
- Dans la requête,appelez une des [API répertoriées ci-dessous](#).

La prise en charge de l'expérience « search-as-you-type » est activée par champ pour des champs de chaînes. Vous pouvez implémenter les deux comportements prédictifs dans la même solution de recherche si vous souhaitez une expérience similaire à celle indiquée dans la capture d'écran. Les deux requêtes ciblent la collection de *documents* d'un index spécifique, et les réponses sont renvoyées après qu'un utilisateur a saisi une chaîne d'entrée avec trois caractères minimum.

## Qu'est-ce qu'un suggesteur ?

Un suggesteur est une structure de données interne prenant en charge des comportements « search-as-you-type » en stockant des préfixes en vue d'une correspondance lors de requêtes partielles. Comme avec les termes tokenisés, les préfixes sont stockés dans des index inversés, soit un pour chaque champ spécifié dans une collection de champs de suggesteurs.

## Définir un suggesteur

Pour créer un suggesteur, ajoutez-en un à un [schéma d'index](#) et [définissez chaque propriété](#). Le meilleur moment pour créer un suggesteur est lorsque vous définissez également le champ qui va l'utiliser.

- Utiliser uniquement des champs de chaîne
- Utilisez l'analyseur Lucene standard par défaut ( "analyzer": null ) ou un [analyseur de langage](#) (par exemple, "analyzer": "en.Microsoft" ) sur le champ.

## Choisir des champs

Bien qu'un suggesteur possède plusieurs propriétés, il se compose principalement d'une collection de champs de chaîne pour lesquels vous activez une expérience « search-as-you-type ». Il existe un suggesteur pour chaque index ; la liste des suggesteurs doit donc inclure tous les champs qui contribuent au contenu des suggestions et de l'autocomplétion.

L'autocomplétion bénéficie d'un plus grand nombre de champs à exploiter, car le contenu supplémentaire a un potentiel d'achèvement des termes plus important.

En revanche, les suggestions produisent de meilleurs résultats lorsque votre choix de champ est sélectif. N'oubliez pas que la suggestion est un proxy pour un document de recherche, vous voudrez donc les champs qui représentent le mieux un résultat unique. Les noms, titres ou autres champs uniques qui font la distinction entre plusieurs correspondances fonctionnent le mieux. Si les champs sont constitués de valeurs répétitives, les suggestions consistent en des résultats identiques, et l'utilisateur ne saura pas sur lequel cliquer.

Pour convenir aux deux expériences « search-as-you-type », ajoutez tous les champs dont vous avez besoin pour l'autocomplétion, mais utilisez `$select`, `$top`, `$filter` et `searchFields` pour contrôler les résultats des suggestions.

## Choisir des analyseurs

L'analyseur choisi détermine la façon dont les champs sont tokenisés et préfixés par la suite. Par exemple, pour une chaîne avec trait d'union comme « context-sensitive », l'utilisation d'un analyseur de langage entraînera les combinaisons de jetons suivantes : « context », « sensitive », « context-sensitive ». Si vous aviez utilisé l'analyseur Lucene standard, la chaîne avec trait d'union n'existerait pas.

Lors de l'évaluation des analyseurs, envisagez d'utiliser l'[API d'analyse de texte](#) pour savoir comment les termes sont tokenisés et préfixés par la suite. Une fois que vous avez créé un index, vous pouvez essayer divers analyseurs sur une chaîne pour afficher la sortie du jeton.

Les champs qui utilisent des [analyseurs personnalisés](#) ou des [analyseurs prédéfinis](#) (à l'exception de Lucene standard) sont explicitement interdits pour éviter des résultats médiocres.

### NOTE

Si vous avez besoin de contourner la contrainte de l'analyseur, par exemple, si vous avez besoin d'un analyseur de mot clé ou de ngram pour certains scénarios de requête, vous devez utiliser deux champs distincts pour le même contenu. Ceci permettra à l'un des champs d'avoir un suggesteur et à l'autre d'être configuré avec une configuration d'analyseur personnalisée.

## Quand créer un suggesteur

La création de la définition de champs est le moment idéal pour créer un suggesteur.

L'API ne vous autorise pas à créer un suggesteur à l'aide de champs préexistants. Des préfixes sont générés pendant l'indexation, lorsque des termes partiels combinant deux caractères ou plus sont tokenisés avec des termes entiers. Les champs existants étant déjà tokenisés, vous devez recréer l'index pour les ajouter à un suggesteur. Pour plus d'informations, consultez [Comment regénérer un index de recherche cognitive Azure](#).

## Créer à l'aide de REST

Dans l'API REST, ajoutez des suggesteurs via [Créer un index](#) ou [Mettre à jour l'index](#).

```
{
  "name": "hotels-sample-index",
  "fields": [
    . . .
    {
      "name": "HotelName",
      "type": "Edm.String",
      "facetable": false,
      "filterable": false,
      "key": false,
      "retrievable": true,
      "searchable": true,
      "sortable": false,
      "analyzer": "en.microsoft",
      "indexAnalyzer": null,
      "searchAnalyzer": null,
      "synonymMaps": [],
      "fields": []
    },
  ],
  "suggesters": [
    {
      "name": "sg",
      "searchMode": "analyzingInfixMatching",
      "sourceFields": ["HotelName"]
    }
  ],
  "scoringProfiles": [
    . . .
  ]
}
```

## Créer à l'aide de .NET

Dans C#, définissez un [objet de suggesteur](#). `Suggesters` est une collection, mais elle peut uniquement accepter un élément.

```
private static void CreateHotelsIndex(SearchServiceClient serviceClient)
{
    var definition = new Index()
    {
        Name = "hotels-sample-index",
        Fields = FieldBuilder.BuildForType<Hotel>(),
        Suggesters = new List<Suggerer>() {new Suggerer()
        {
            Name = "sg",
            SourceFields = new string[] { "HotelName", "Category" }
        }}
    };
    serviceClient.Indexes.Create(definition);
}
```

## Informations de référence sur les propriétés

PROPRIÉTÉ	DESCRIPTION
<code>name</code>	Nom du suggesteur.

PROPRIÉTÉ	DESCRIPTION
<code>searchMode</code>	Stratégie utilisée pour rechercher des expressions candidates. Le seul mode actuellement pris en charge est <code>analyzingInfixMatching</code> , qui établit actuellement une correspondance au début d'un terme.
<code>sourceFields</code>	<p>Liste d'un ou de plusieurs champs constituant la source du contenu pour des suggestions. Les champs doivent être de type <code>Edm.String</code> et <code>Collection(Edm.String)</code>. Tout analyseur spécifié dans le champ doit correspondre à un analyseur nommé issu de <a href="#">cette liste</a> (et non à un analyseur personnalisé).</p> <p>Il vous est conseillé de spécifier uniquement les champs qui se prêtent à une réponse attendue et appropriée, qu'il s'agisse d'une chaîne complète dans une barre de recherche ou d'une liste déroulante.</p> <p>Un nom d'hôtel est un bon candidat, car il est précis. Les champs détaillés tels que des descriptions et des commentaires sont trop denses. De même, les champs répétitifs, tels que les balises et les catégories sont moins efficaces. Dans les exemples, nous incluons tout de même « catégorie » pour montrer que vous pouvez inclure plusieurs champs.</p>

## Utiliser un suggesteur

Un suggesteur est utilisé dans une requête. Après la création d'un suggesteur,appelez l'une des API suivantes pour tester une expérience « search-as-you-type » :

- [REST API Suggestions](#)
- [API REST Autocomplétion](#)
- [Méthode SuggestWithHttpMessagesAsync](#)
- [Méthode AutocompleteWithHttpMessagesAsync](#)

Dans une application de recherche, le code client doit utiliser une bibliothèque comme [jQuery UI Autocomplete](#) pour collecter la requête partielle et fournir la correspondance. Pour plus d'informations sur cette tâche, consultez [Ajouter l'autocomplétion ou les résultats suggérés au code client](#).

L'utilisation de l'API est illustrée dans l'appel suivant de l'API REST Autocomplétion. Dans cet exemple, deux éléments importants sont à retenir. Tout d'abord, comme avec toutes les requêtes, l'opération est effectuée sur la collection de documents d'un index et la requête inclut un paramètre de **recherche**, qui, dans ce cas, fournit la requête partielle. Ensuite, vous devez ajouter **suggesterName** à la requête. Si un suggesteur n'est pas défini dans l'index, tout appel à l'autocomplétion ou aux suggestions se soldera par un échec.

```
POST /indexes/myxboxgames/docs/autocomplete?search&api-version=2020-06-30
{
  "search": "minecraft",
  "suggesterName": "sg"
}
```

## Exemple de code

- L'exemple [Créer votre première application en C# \(Leçon 3 : Ajouter « search-as-you-type »\)](#) illustre une construction de suggesteur, des requêtes suggérées, l'autocomplétion et la navigation par facettes. Cet

exemple de code s'exécute dans un service Recherche cognitive Azure de bac à sable et utilise un index des hôtels préchargé. Vous n'avez plus qu'à appuyer sur F5 pour exécuter l'application. Aucun abonnement ou connexion n'est nécessaire.

- [DotNetHowToAutocomplete](#) correspond à un précédent exemple contenant du code C# et Java. Il illustre une construction de suggesteur, des requêtes suggérées, l'autocomplétion et la navigation par facettes. Cet exemple de code utilise les exemples de données hébergées [NYCJobs](#).

## Étapes suivantes

Nous vous recommandons l'article suivant pour en savoir plus sur la formulation des requêtes.

[Ajouter l'autocomplétion et les suggestions au code client](#)

# Ajouter l'autocomplétion et les suggestions aux applications clientes

04/10/2020 • 17 minutes to read • [Edit Online](#)

La recherche en cours de frappe est une technique courante pour améliorer la productivité des requêtes initiées par l'utilisateur. Dans Recherche cognitive Azure, cette expérience est prise en charge via l'*autocomplétion*, qui termine un terme ou une expression en fonction d'une entrée partielle (en complétant « micro » par « microsoft » par exemple). Les *suggestions* sont une autre possibilité : une liste succincte de documents correspondants (retournant des titres de livres avec un ID pour pouvoir lier vers une page d'informations). L'autocomplétion et les suggestions sont basées sur une correspondance dans l'index. Le service ne propose pas de requêtes qui ne retournent aucun résultat.

Pour implémenter ces expériences dans Recherche cognitive Azure, vous avez besoin des éléments suivants :

- Un *suggesteur* sur le back end.
- Une *requête* spécifiant l'API [Autocomplétion](#) ou [Suggestions](#) sur la demande.
- Un *contrôle d'interface utilisateur* pour gérer les interactions de type recherche en cours de frappe dans votre application cliente. Nous vous recommandons d'utiliser une bibliothèque JavaScript existante à cet effet.

Dans Recherche cognitive Azure, les requêtes avec autocomplétion et les résultats suggérés sont récupérés à partir de l'index de recherche, à partir des champs sélectionnés que vous avez enregistrés avec un suggesteur. Un suggesteur fait partie de l'index. Il spécifie les champs qui fournissent du contenu qui complètent une requête et/ou suggère un résultat. Quand l'index est créé et chargé, une structure de données de suggesteur est créée en interne pour stocker les préfixes utilisés pour la correspondance sur des requêtes partielles. Pour les suggestions, le choix de champs appropriés qui sont uniques, ou au moins non répétitifs, est essentiel à l'expérience. Pour plus d'informations, consultez [Créer un suggesteur](#).

Le reste de cet article se concentre sur les requêtes et le code client. Il utilise JavaScript et C# pour illustrer des points clés. Les exemples d'API REST sont utilisés pour présenter chaque opération de façon concise. Pour obtenir des liens vers des exemples de code de bout en bout, consultez [Étapes suivantes](#).

## Configurer une demande

Les éléments d'une demande incluent l'une des API de recherche en cours de frappe, une requête partielle et un suggesteur. Le script suivant illustre les composants d'une demande, à l'aide de l'API REST d'autocomplétion.

```
POST /indexes/myxboxgames/docs/autocomplete?search&api-version=2020-06-30
{
  "search": "minecraf",
  "suggesterName": "sg"
}
```

Le **suggesterName** vous donne les champs de suggesteur utilisés pour compléter les termes ou les suggestions. Pour les suggestions en particulier, la liste de champs doit être composée de ceux qui offrent des choix clairs parmi les résultats de correspondance. Sur un site qui vend des jeux informatiques, le champ peut être le titre du jeu.

Le paramètre **search** fournit la requête partielle, où les caractères sont transmis à la demande de requête via le contrôle Autocomplete dans jQuery. Dans l'exemple ci-dessus, « minecraf » est une illustration statique de ce que le contrôle peut transmettre.

Les API n'imposent pas d'exigences de longueur minimale sur la requête partielle qui peut inclure un seul caractère. Toutefois, le contrôle Autocomplete de jQuery fournit une longueur minimale. Un minimum de deux ou trois caractères est normal.

Les correspondances se trouvent au début d'un terme n'importe où dans la chaîne d'entrée. Prenons comme exemple « The Quick Brown Fox » : l'autocomplétion et les suggestions correspondent à des versions partielles de « The », « Quick », « Brown » ou « Fox », mais pas à des termes partiels tels que « rown » ou « ox ». En outre, chaque correspondance définit l'étendue des expansions en aval. Une requête partielle « Quick br » est une correspondance de « Quick Brown » ou « Quick Bread », mais ni « Brown » ni « Bread » ne sont des correspondances, sauf si « Quick » les précède.

### API pour la recherche en cours de frappe

Cliquez sur les liens suivants pour accéder aux pages de référence des SDK REST et .NET :

- [REST API Suggestions](#)
- [API REST Autocomplétion](#)
- [Méthode SuggestWithHttpMessagesAsync](#)
- [Méthode AutocompleteWithHttpMessagesAsync](#)

## Structurer une réponse

Les réponses de l'autocomplétion et des suggestions sont celles que vous pouvez attendre pour le modèle : L'[autocomplétion](#) retourne une liste de termes et les [suggestions](#) retournent des termes plus un ID de document pour vous permettre de récupérer le document (utilisez l'API [Document de recherche](#) pour récupérer le document spécifique pour une page d'informations).

Les réponses sont mises en forme par les paramètres que vous incluez dans la demande. Pour l'autocomplétion, définissez [autocompleteMode](#) pour déterminer si la complétion du texte se produit sur un ou deux termes. Pour les suggestions, le champ que vous choisissez détermine le contenu de la réponse.

Vous devez affiner la réponse pour éviter les doublons ou les résultats non liés. Pour contrôler les résultats, incluez davantage de paramètres dans la demande. Les paramètres suivants s'appliquent à la fois à l'autocomplétion et aux suggestions. Ils sont toutefois peut-être plus utiles pour les suggestions, en particulier lorsqu'un suggesteur inclut plusieurs champs.

PARAMÈTRE	USAGE
<code>\$select</code>	Si vous avez plusieurs champs source ( <code>sourceFields</code> ) dans un suggesteur, utilisez <code>\$select</code> pour choisir le champ qui contribue aux valeurs ( <code>\$select=GameTitle</code> ).
<code>searchFields</code>	Limitez la requête à des champs spécifiques.
<code>\$filter</code>	Appliquez des critères de correspondance sur le jeu de résultats ( <code>\$filter=Category eq 'ActionAdventure'</code> ).
<code>\$top</code>	Limitez les résultats à un nombre spécifique ( <code>\$top=5</code> ).

## Ajouter un code d'interaction utilisateur

Le remplissage automatique d'un terme de requête ou l'affichage d'une liste de liens correspondants nécessite un code d'interaction utilisateur, en général JavaScript, qui peut consommer des demandes provenant de sources externes, telles que les requêtes d'autocomplétion ou de suggestion sur un index Recherche cognitive Azure.

Bien que vous puissiez écrire ce code en mode natif, il est beaucoup plus facile d'utiliser les fonctions d'une

bibliothèque JavaScript existante. Cet article en présente deux : l'une pour les suggestions et l'autre pour l'autocomplétion.

- Le [widget d'autocomplétion \(jQuery UI\)](#) est utilisé dans l'exemple de suggestion. Vous pouvez créer une zone de recherche, puis la référencer dans une fonction JavaScript qui utilise le widget d'autocomplétion. Les propriétés du widget définissent la source (fonction d'autocomplétion ou de suggestions), la longueur minimale des caractères d'entrée avant que l'action soit effectuée et le positionnement.
- Le [plug-in d'autocomplétion fourni par XDSOFT](#) est utilisé dans l'exemple d'autocomplétion.

Nous utilisons ces bibliothèques pour créer la zone de recherche prenant en charge les suggestions et l'autocomplétion. Les entrées collectées dans la zone de recherche sont associées à des suggestions et des actions d'autocomplétion.

## Suggestions

Cette section vous guide tout au long de l'implémentation des résultats suggérés, en commençant par la définition de la zone de recherche. Elle montre également un script qui appelle la première bibliothèque d'autocomplétion JavaScript référencée dans cet article.

### Créer une zone de recherche

En prenant l'exemple de la [bibliothèque d'autocomplétion jQuery UI](#) et d'un projet MVC en C#, vous pouvez définir la zone de recherche à l'aide de JavaScript dans le fichier **Index.cshtml**. La bibliothèque ajoute l'interaction de recherche en cours de frappe en effectuant des appels asynchrones au contrôleur MVC afin de récupérer les suggestions.

Dans **Index.cshtml** sous le dossier \Views\Home, une ligne pour créer une zone de recherche peut se présenter comme suit :

```
<input class="searchBox" type="text" id="searchbox1" placeholder="search">
```

Cet exemple est une simple zone de texte avec une classe pour le style, un ID qui doit être référencé par JavaScript et un texte d'espace réservé.

Dans le même fichier, incorporez le code JavaScript qui fait référence à la zone de recherche. La fonction suivante appelle l'API **Suggest**, qui demande les documents correspondants suggérés en fonction des entrées de terme partielles :

```
$(function () {
    $("#searchbox1").autocomplete({
        source: "/home/suggest?highlights=false&fuzzy=false",
        minLength: 3,
        position: {
            my: "left top",
            at: "left-23 bottom+10"
        }
    });
});
```

`source` indique à la fonction d'autocomplétion de jQuery UI où récupérer la liste des éléments à afficher sous la zone de recherche. Dans la mesure où ce projet est un projet MVC, il appelle la fonction **Suggest** dans le fichier **HomeController.cs** qui contient la logique permettant de retourner les suggestions de requête. Cette fonction transmet également quelques paramètres pour contrôler la mise en surbrillance, les correspondances approximatives et les termes. L'API JavaScript d'autocomplétion ajoute le paramètre de terme.

`minLength: 3` garantit que les recommandations ne sont affichées que lorsque plus de trois caractères sont saisis

dans la zone de recherche.

### Activer la correspondance approximative

La recherche approximative vous permet d'obtenir des résultats selon des correspondances proches, même si l'utilisateur a mal épelé un mot dans la zone de recherche. La distance d'édition est 1, ce qui signifie qu'il peut y avoir un écart maximal de 1 caractère entre l'entrée de l'utilisateur et une correspondance.

```
source: "/home/suggest?highlights=false&fuzzy=true&",


```

### Activer la mise en surbrillance

La mise en surbrillance applique le style de police aux caractères du résultat qui correspondent à l'entrée de l'utilisateur. Par exemple, si l'entrée partielle est « micro », le résultat est le suivant : **microsoft**, **microscope**, et ainsi de suite. La mise en surbrillance est basée sur les paramètres **HighlightPreTag** et **HighlightPostTag**, définis en ligne avec la fonction **Suggestion**.

```
source: "/home/suggest?highlights=true&fuzzy=true&",


```

### Fonction Suggest

Si vous utilisez C# et une application MVC, le fichier **HomeController.cs** sous le répertoire Controllers est l'endroit où vous pouvez créer une classe pour les résultats suggérés. Dans .NET, une fonction **Suggest** est basée sur la méthode [DocumentsOperationsExtensions.Suggest](#).

La méthode `InitSearch` crée un client d'index HTTP authentifié dans le service Recherche cognitive Azure. Pour plus d'informations sur le SDK .NET, consultez [Guide pratique pour utiliser la Recherche cognitive Azure à partir d'une application .NET](#).

```
public ActionResult Suggest(bool highlights, bool fuzzy, string term)
{
    InitSearch();

    // Call suggest API and return results
    SuggestParameters sp = new SuggestParameters()
    {
        Select = HotelName,
        SearchFields = HotelName,
        UseFuzzyMatching = fuzzy,
        Top = 5
    };

    if (highlights)
    {
        sp.HighlightPreTag = "<b>";
        sp.HighlightPostTag = "</b>";
    }

    DocumentSuggestResult resp = _indexClient.Documents.Suggest(term, "sg", sp);

    // Convert the suggest query results to a list that can be displayed in the client.
    List<string> suggestions = resp.Results.Select(x => x.Text).ToList();
    return new JsonResult
    {
        JsonRequestBehavior = JsonRequestBehavior.AllowGet,
        Data = suggestions
    };
}
```

La fonction **Suggest** prend deux paramètres qui déterminent si les meilleurs résultats sont renvoyés ou si la correspondance approximative est utilisée en plus de la recherche de terme. La méthode crée un objet

[SuggestParameters](#), qui est ensuite passé à l'API de suggestion. Le résultat est ensuite converti en JSON pour être visible par le client.

## Autocomplétion

Jusqu'à présent, le code de l'expérience utilisateur de recherche a été centré sur les suggestions. Le bloc de code suivant montre l'autocomplétion, en utilisant la fonction d'autocomplétion XDSoft jQuery UI, en passant une requête d'autocomplétion pour Recherche cognitive Azure. Comme avec les suggestions, dans une application C#, le code qui prend en charge l'interaction utilisateur se trouve dans `index.cshtml`.

```
$function () {
    // using modified jQuery Autocomplete plugin v1.2.6 https://xdsoft.net/jqplugins/autocomplete/
    // $.autocomplete -> $.autocompleteInline
    $("#searchbox1").autocompleteInline({
        appendMethod: "replace",
        source: [
            function (text, add) {
                if (!text) {
                    return;
                }

                $.getJSON("/home/autocomplete?term=" + text, function (data) {
                    if (data && data.length > 0) {
                        currentSuggestion2 = data[0];
                        add(data);
                    }
                });
            }
        ]
    });

    // complete on TAB and clear on ESC
    $("#searchbox1").keydown(function (evt) {
        if (evt.keyCode === 9 /* TAB */ && currentSuggestion2) {
            $("#searchbox1").val(currentSuggestion2);
            return false;
        } else if (evt.keyCode === 27 /* ESC */) {
            currentSuggestion2 = "";
            $("#searchbox1").val("");
        }
    });
}
```

### Fonction d'autocomplétion

L'autocomplétion est basée sur la [méthode DocumentsOperationsExtensions.AutoComplete](#). Comme avec les suggestions, ce bloc de code se trouve dans le fichier `HomeController.cs`.

```

public ActionResult AutoComplete(string term)
{
    InitSearch();
    //Call autocomplete API and return results
    AutocompleteParameters ap = new AutocompleteParameters()
    {
        AutocompleteMode = AutocompleteMode.OneTermWithContext,
        UseFuzzyMatching = false,
        Top = 5
    };
    AutocompleteResult autocompleteResult = _indexClient.Documents.Autocomplete(term, "sg", ap);

    // Convert the Suggest results to a list that can be displayed in the client.
    List<string> autocomplete = autocompleteResult.Results.Select(x => x.Text).ToList();
    return new JsonResult
    {
        JsonRequestBehavior = JsonRequestBehavior.AllowGet,
        Data = autocomplete
    };
}

```

La fonction d'autocomplétion prend l'entrée du terme de recherche. La méthode crée un [objet AutocompleteParameters](#). Le résultat est ensuite converti en JSON pour être visible par le client.

## Étapes suivantes

Suivez ces liens pour obtenir des instructions de bout en bout ou du code illustrant les deux expériences de recherche en cours de frappe. Les deux exemples de code incluent des implémentations hybrides des suggestions et de l'autocomplétion.

- [Tutoriel : Créer votre première application en C# \(leçon 3\)](#)
- [Exemple de code C# : azure-search-dotnet-samples/create-first-app/3-add-typeahead/](#)
- [C# et JavaScript avec l'exemple de code côté à côté REST](#)

# Syntaxe de requête simple dans la recherche cognitive Azure

04/10/2020 • 15 minutes to read • [Edit Online](#)

La recherche cognitive Azure implémente deux langages de requête basés sur Lucene : L'[analyseur de requêtes simples](#) et l'[analyseur de requêtes Lucene](#).

L'analyseur simple est plus flexible, et tente d'interpréter une requête même si elle n'est pas parfaitement composée. Pour cette raison, il s'agit de l'analyseur par défaut pour les requêtes dans Recherche cognitive Azure.

La syntaxe simple est utilisée pour les expressions de requête passées dans le paramètre `search` d'une [requête de recherche dans des documents](#), et ne doit pas être confondue avec la [syntaxe OData](#) utilisée pour le paramètre d'[expressions \\$filter](#) de la même API de recherche dans des documents. Les paramètres `search` et `$filter` ont une syntaxe différente, avec leurs propres règles pour la construction de requêtes, l'échappement de chaînes, et ainsi de suite.

Bien que l'analyseur simple soit basé sur la classe de l'[analyseur de requêtes simple Apache Lucene](#), l'implémentation dans Recherche cognitive Azure exclut la recherche approximative. Si vous avez besoin de la [recherche approximative](#) ou d'autres formes de requête avancée, utilisez plutôt la [syntaxe de requête Lucene complète](#).

## Appeler une analyse simple

La syntaxe simple est la syntaxe par défaut. Appeler celle-ci est nécessaire seulement si vous repassez de la syntaxe complète à la syntaxe simple. Pour définir explicitement la syntaxe, utilisez le paramètre de recherche `queryType`. Les valeurs valides sont `queryType=simple` ou `queryType=full`, `simple` correspondant à la valeur par défaut, et `full` appelle l'[analyseur de requêtes Lucene complètes](#) pour les requêtes plus avancées.

## Bases de la syntaxe

Tout texte avec un ou plusieurs termes est considéré comme un point de départ valide pour l'exécution des requêtes. La recherche cognitive Azure établit une correspondance avec les documents contenant tout ou partie des conditions, y compris d'éventuelles variations détectées lors de l'analyse du texte.

Aussi simple que cela puisse paraître, un aspect de l'exécution des requêtes dans la recherche cognitive Azure est qu'elles *peuvent* produire des résultats inattendus, en augmentant le nombre des résultats de la recherche au lieu de les diminuer quand vous ajoutez plus de conditions et d'opérateurs à la chaîne d'entrée. Cette expansion dépend de l'inclusion de l'opérateur NOT, combiné à une valeur de paramètre `searchMode` qui détermine comment NOT est interprété en termes de comportements AND ou OR. Pour plus d'informations, consultez [Opérateur NOT](#).

### Opérateurs de priorité (regroupement)

Vous pouvez utiliser des parenthèses pour créer des sous-requêtes, en incluant des opérateurs au sein de l'instruction entre parenthèses. Par exemple, `motel+(wifi|luxury)` recherche les documents contenant le terme « motel », et « wifi » ou « luxury » (ou les deux).

Le regroupement de champs est similaire, mais il délimite le regroupement à un seul champ. Par exemple, `hotelAmenities:(gym+(wifi|pool))` recherche « gym » et « wifi », ou « gym » et « pool », dans le champ « hotelAmenities ».

## Échappement des opérateurs de recherche

Dans la syntaxe simple, les opérateurs de recherche incluent les caractères suivants : + | " ( ) ' \

Si l'un de ces caractères fait partie d'un jeton dans l'index, échappez-le en le faisant précédé d'une barre oblique inverse (\) dans la requête. Par exemple, supposez que vous avez utilisé un analyseur personnalisé pour la création de jetons de termes entiers, et que votre index contient la chaîne « Luxury+Hotel ». Pour obtenir une correspondance exacte sur ce jeton, insérez un caractère d'échappement : `search=luxury\+hotel`.

Pour simplifier les choses pour les cas les plus classiques, il existe deux exceptions à cette règle où l'échappement n'est pas nécessaire :

- L'opérateur NOT - doit être placé en échappement seulement s'il s'agit du premier caractère après un espace. Si le - apparaît au milieu (par exemple `3352CDD0-EF30-4A2E-A512-3B30AF40F3FD`), aucun échappement n'est nécessaire.
- L'opérateur de suffixe \* doit être placé en échappement seulement s'il s'agit du dernier caractère avant un espace. Si le \* apparaît au milieu (par exemple `4*4=16`), aucun échappement n'est nécessaire.

### NOTE

Par défaut, l'analyseur standard supprime et coupe les mots sur les traits d'union, les espaces, les esperluettes et autres caractères pendant l'[analyse lexique](#). Si vous avez besoin que les caractères spéciaux restent dans la chaîne de requête, vous aurez peut-être besoin d'un analyseur qui les conserve dans l'index. Parmi les choix possibles figurent les [analyseurs de langage](#) naturel de Microsoft, qui conservent les mots avec trait d'union, ou un analyseur personnalisé pour les modèles plus complexes. Pour plus d'informations, consultez [Termes partiels, modèles et caractères spéciaux](#).

## Encodage de caractères dangereux et réservés dans les URL

Vérifiez que tous les caractères dangereux et réservés dans une URL sont encodés. Par exemple, « # » est un caractère non sécurisé, car il s'agit d'un identificateur de fragment/d'ancrage au sein d'une URL. Le caractère doit être encodé en %23 s'il est utilisé dans une URL. « & » et « = » sont des exemples de caractères réservés, car ils délimitent les paramètres et spécifient des valeurs dans la Recherche cognitive Azure. Pour plus d'informations, consultez [RFC1738 : Uniform Resource Locators \(URL\)](#).

Les caractères dangereux sont " ` < > # % { } | \ ^ ~ [ ]. Les caractères réservés sont ; / ? : @ = + & .

## Exécution d'une requête pour des caractères spéciaux

Dans certains cas, vous rechercherez un caractère spécial, tel que l'emoji « ❤ » ou le signe « € ». Le cas échéant, assurez-vous que l'analyseur que vous utilisez n'ignore pas ces caractères. L'analyseur standard ignore la plupart des caractères spéciaux de façon à ce qu'ils ne deviennent pas des jetons dans votre index.

La première étape consiste donc à vous assurer que vous utilisez un analyseur qui prendra en compte ces jetons d'éléments. Par exemple, l'analyseur « Whitespace » prend en compte toutes les séquences de caractères séparées par des espaces blancs comme des jetons, donc la chaîne « ❤ » est considérée comme un jeton. En outre, un analyseur comme Microsoft English Analyzer (« en.microsoft ») considère la chaîne « € » comme un jeton. Vous pouvez [tester un analyseur](#) pour voir quels jetons il génère pour une requête donnée.

Lorsque vous utilisez des caractères Unicode, assurez-vous que les symboles sont correctement placés dans une séquence d'échappement dans l'URL de la requête (par exemple, pour « ❤ » utilisez la séquence d'échappement `%E2%9D%A4+`). Postman effectue cette traduction automatiquement.

## Limites de taille des requêtes

Il existe une limite à la taille des requêtes que vous pouvez envoyer à la Recherche cognitive Azure. Plus précisément, vous pouvez avoir au maximum 1 024 clauses (des expressions séparées par AND, OR, etc.). Il existe également une limite d'environ 32 Ko pour la taille d'un terme individuel dans une requête. Si votre

application génère des requêtes de recherche par programmation, nous vous recommandons de la concevoir de façon à ce qu'elle ne génère pas des requêtes d'une taille illimitée.

## Recherche booléenne

Vous pouvez incorporer des opérateurs booléens (AND, OR, NOT) dans une chaîne de requête pour créer un ensemble substantiel de critères en fonction desquels les documents correspondants sont trouvés.

### Opérateur AND

L'opérateur AND est un signe plus. Par exemple, `wifi + luxury` recherche les documents contenant à la fois `wifi` et `luxury`.

### Opérateur OR

L'opérateur OR est une barre verticale. Par exemple, `wifi | luxury` recherche les documents contenant `wifi` ou `luxury`, ou les deux.

### Opérateur NOT

L'opérateur NOT est un signe moins. Par exemple, `wifi -luxury` recherche les documents qui contiennent le terme `wifi` et/ou pas le terme `luxury`.

Dans une requête, le paramètre `searchMode` détermine si un terme accompagné de l'opérateur NOT est associé à d'autres termes de la requête par les opérateurs AND ou OR (en supposant qu'il n'y ait pas d'opérateur `+` ou `|` sur les autres termes). Les valeurs valides sont `any` ou `all`.

`searchMode=any` augmente le rappel des requêtes en incluant plus de résultats, et par défaut `-` est interprété comme « OR NOT ». Par exemple, `wifi -luxury` établit une correspondance avec les documents qui contiennent le terme `wifi` ou avec ceux qui ne contiennent pas le terme `luxury`.

`searchMode=all` augmente la précision des requêtes en incluant moins de résultats et, par défaut, « - » est interprété comme « AND NOT ». Par exemple, `wifi -luxury` établit une correspondance avec les documents qui contiennent le terme `wifi` et ne contiennent pas le terme « `luxury` ». Il s'agit sans doute d'un comportement plus intuitif pour l'opérateur `-`. Ainsi, préférez `searchMode=all` à `searchMode=any` si vous souhaitez optimiser la précision des recherches plutôt que le rappel, *et si vos utilisateurs utilisent fréquemment l'opérateur `-` dans les recherches*.

Lorsque vous choisissez un paramètre `searchMode`, tenez compte des modèles d'interaction utilisateur pour les requêtes liées à différentes applications. Les utilisateurs qui recherchent des informations sont plus enclins à inclure un opérateur dans une requête, contrairement aux sites de commerce électronique qui disposent de structures de navigation intégrées.

## Correspondance du préfixe de caractère générique (\*, ?)

Pour les requêtes « commence par », ajoutez un opérateur de suffixe en tant qu'espace réservé pour le reste d'un terme. Utilisez un astérisque `*` pour plusieurs caractères ou `?` pour les caractères uniques. Par exemple, `lingui*` correspond à « linguistique » ou « linguini », en ignorant la casse.

Comme pour les filtres, une requête de préfixe recherche une correspondance exacte. Par conséquent, il n'existe aucun scoring de pertinence (tous les résultats reçoivent un score de recherche de 1.0). Sachez que les requêtes de préfixe peuvent être lentes, en particulier si l'index est volumineux et le préfixe constitué d'un petit nombre de caractères. Une autre méthodologie, telle que la segmentation du texte en unités lexicales de type edge n-gram, peut s'effectuer plus rapidement.

Pour les autres variantes de requêtes de caractères génériques, telles que la correspondance de suffixes ou d'infixes à la fin ou au milieu d'un terme, utilisez la [syntaxe Lucene complète pour la recherche de caractères génériques](#).

# Recherche d'expressions

Une recherche de termes est une requête portant sur un ou plusieurs termes et dans laquelle un des termes est considéré comme une correspondance. Une recherche d'expression est une expression exacte placée entre guillemets " ". Par exemple, si Roach Motel (sans guillemets) recherche les documents contenant Roach et/ou Motel n'importe où dans n'importe quel ordre, "Roach Motel" (avec des guillemets) établit une correspondance seulement avec les documents qui contiennent cette expression entière, avec les mots dans cet ordre (l'analyse lexicale s'applique néanmoins toujours).

## Voir aussi

- [Fonctionnement de la recherche en texte intégral dans la Recherche cognitive Azure](#)
- [Exemples de requêtes pour une recherche simple](#)
- [Exemples de requêtes pour une recherche Lucene complète](#)
- [API REST de recherche de documents](#)
- [Syntaxe de requête Lucene](#)
- [Syntaxe d'expression OData](#)

# Syntaxe de requête Lucene dans la Recherche cognitive Azure

04/10/2020 • 25 minutes to read • [Edit Online](#)

Vous pouvez écrire des requêtes sur la Recherche cognitive Azure en utilisant la syntaxe riche en fonctionnalités de l'[analyseur de requêtes Lucene](#) pour des formes de requêtes spécialisées : caractère générique, recherche approximative, recherche de proximité et expressions régulières en sont quelques exemples. La plus grande partie de la syntaxe de l'analyseur de requêtes Lucene est [implémentée telle quelle dans la Recherche cognitive Azure](#), à l'exception des *recherches de plage*, qui sont construites dans la Recherche cognitive Azure à l'aide d'expressions `$filter`.

## NOTE

La syntaxe Lucene complète est utilisée pour les expressions de requête passées dans le paramètre `search` de l'API [Recherche dans des documents](#) et ne doit pas être confondue avec la [syntaxe OData](#) utilisée pour le paramètre `$Filter` de cette API. Ces différentes syntaxes ont leurs propres règles pour la construction de requêtes, l'échappement de chaînes, etc.

## Appeler l'analyse complète

Définissez le paramètre de recherche `queryType` pour spécifier l'analyseur à utiliser. Les valeurs valides sont `simple|full`, `simple` étant la valeur par défaut et `full` la valeur pour Lucene.

### Exemple montrant la syntaxe complète

L'exemple suivant recherche les documents dans l'index avec la syntaxe de requête Lucene, ce qu'indique le paramètre `queryType=full`. Cette requête renvoie les hôtels où le champ de catégorie contient le terme « budget » et tous les champs pouvant faire l'objet d'une recherche contenant l'expression « récemment rénové ». Les documents contenant l'expression « recently renovated » sont mieux classés en raison de la valeur de promotion du terme (3).

Le paramètre `searchMode=all` est approprié dans cet exemple. Quand des opérateurs se trouvent sur la requête, vous devez généralement définir `searchMode=all` pour garantir que *tous* les critères sont satisfaits.

```
GET /indexes/hotels/docs?search=category:budget AND \"recently renovated\"^3&searchMode=all&api-version=2020-06-30&querytype=full
```

Vous pouvez aussi utiliser POST :

```
POST /indexes/hotels/docs/search?api-version=2020-06-30
{
  "search": "category:budget AND \"recently renovated\"^3",
  "queryType": "full",
  "searchMode": "all"
}
```

Pour obtenir d'autres exemples, consultez [Exemples de syntaxe de requête Lucene pour créer des requêtes dans la Recherche cognitive Azure](#). Pour plus d'informations sur la spécification de tous les paramètres des requêtes, consultez [Rechercher des documents \(API REST de Recherche cognitive Azure\)](#).

#### NOTE

La Recherche cognitive Azure prend également en charge une [syntaxe de requête simple](#) : il s'agit d'un langage de requête simple et robuste qui peut être utilisé pour la recherche directe de mots clés.

## Principes de base de la syntaxe

les principes de base suivants de la syntaxe s'appliquent à toutes les requêtes qui utilisent la syntaxe Lucene.

### Évaluation des opérateurs en contexte

L'emplacement détermine si un symbole est interprété comme un opérateur ou simplement comme un autre caractère dans une chaîne.

Par exemple, dans la syntaxe complète Lucene, le tilde (~) est utilisé pour la recherche approximative et la recherche de proximité. Quand il est placé après une expression entre guillemets, « ~ » appelle la recherche de proximité. Quand il est placé à la fin d'un terme, « ~ » appelle la recherche approximative.

Au sein d'un terme, comme « business~analyst », le caractère n'est pas évalué comme opérateur. Dans ce cas, en supposant que la requête est une requête de terme ou d'expression, la [recherche en texte intégral](#) avec [analyse lexicale](#) supprime le ~ et décompose le terme « business~analyst » en deux : business OR analyst.

L'exemple ci-dessus concerne le tilde (~), mais le même principe s'applique à chaque opérateur.

### Échappement des caractères spéciaux

Pour utiliser un opérateur de recherche dans le texte de recherche, placez le caractère dans une séquence d'échappement en le faisant précéder d'une barre oblique inverse (\). Par exemple, pour une recherche avec caractères génériques sur https://, sachant que // fait partie de la chaîne de requête, vous devez spécifier search=https\:\/\/\* . De même, un modèle de numéro de téléphone placé dans une séquence d'échappement peut se présenter comme suit : \+1 \(800\) 642\ -7676 .

Les caractères spéciaux qui doivent être placés dans une séquence d'échappement sont les suivants :

+ - & | ! ( ) { } [ ] ^ " ~ \* ? : \ /

#### NOTE

Bien que l'échappement maintienne les jetons ensemble, l'[analyse lexicale](#) pendant l'indexation peut les supprimer. Par exemple, l'analyseur Lucene standard coupe les mots sur les traits d'union, les espaces et autres caractères. Si vous avez besoin de caractères spéciaux dans la chaîne de requête, vous aurez peut-être également besoin d'un analyseur qui les conserve dans l'index. Parmi les choix possibles figurent les [analyseurs de langage](#) naturel de Microsoft, qui conservent les mots avec trait d'union, ou un analyseur personnalisé pour les modèles plus complexes. Pour plus d'informations, consultez [Termes partiels, modèles et caractères spéciaux](#).

### Encodage de caractères dangereux et réservés dans les URL

Vérifiez que tous les caractères dangereux et réservés dans une URL sont encodés. Par exemple, « # » est un caractère non sécurisé, car il s'agit d'un identificateur de fragment/d'ancrage au sein d'une URL. Le caractère doit être encodé en %23 s'il est utilisé dans une URL. « & » et « = » sont des exemples de caractères réservés, car ils délimitent les paramètres et spécifient des valeurs dans la Recherche cognitive Azure. Pour plus d'informations, consultez [RFC1738 : Uniform Resource Locators \(URL\)](#).

Les caractères dangereux sont " ` < > # % { } | \ ^ ~ [ ] . Les caractères réservés sont ; / ? : @ = + & .

### Limites de taille des requêtes

Il existe une limite à la taille des requêtes que vous pouvez envoyer à la Recherche cognitive Azure. Plus

précisément, vous pouvez avoir au maximum 1 024 clauses (des expressions séparées par AND, OR, etc.). Il existe également une limite d'environ 32 Ko pour la taille d'un terme individuel dans une requête. Si votre application génère des requêtes de recherche par programmation, nous vous recommandons de la concevoir de façon à ce qu'elle ne génère pas des requêtes d'une taille illimitée.

### Opérateurs de priorité (regroupement)

Vous pouvez utiliser des parenthèses pour créer des sous-requêtes, en incluant des opérateurs au sein de l'instruction entre parenthèses. Par exemple, `motel+(wifi||luxury)` recherche les documents contenant le terme « motel », et « wifi » ou « luxury » (ou les deux).

Le regroupement de champs est similaire, mais il délimite le regroupement à un seul champ. Par exemple, `hotelAmenities:(gym+(wifi||pool))` recherche « gym » et « wifi », ou « gym » et « pool », dans le champ « hotelAmenities ».

## Recherche booléenne

Spécifiez toujours les opérateurs booléens de texte (AND, OR, NOT) tout en majuscules.

### Opérateur OR `OR` `ou` `||`

L'opérateur OR est une barre verticale. Par exemple, `wifi || luxury` recherche les documents contenant « wifi » ou « luxury », ou les deux. Comme OR est l'opérateur de conjonction par défaut, vous pouvez aussi l'omettre : ainsi, `wifi luxury` est l'équivalent de `wifi || luxury`.

### Opérateur AND `AND`, `&&` `ou` `+`

L'opérateur AND est une esperluette ou un signe plus. Par exemple, `wifi && luxury` recherche les documents contenant à la fois « wifi » et « luxury ». Le caractère plus (+) est utilisé pour les termes obligatoirement présents. Par exemple, `+wifi +luxury` stipule que les deux termes doivent apparaître quelque part dans le champ d'un même document.

### Opérateur NOT `NOT`, `!` `ou` `-`

L'opérateur NOT est un signe moins. Par exemple, `wifi -luxury` recherche les documents qui contiennent le terme `wifi` et/ou pas le terme `luxury`.

Dans une requête, le paramètre `searchMode` détermine si un terme accompagné de l'opérateur NOT est associé à d'autres termes de la requête par les opérateurs AND ou OR (en supposant qu'il n'y ait pas d'opérateur `+` ou `|` sur les autres termes). Les valeurs valides sont `any` ou `all`.

`searchMode=any` augmente le rappel des requêtes en incluant plus de résultats, et par défaut `-` est interprété comme « OR NOT ». Par exemple, `wifi -luxury` établit une correspondance avec les documents qui contiennent le terme `wifi` ou avec ceux qui ne contiennent pas le terme `luxury`.

`searchMode=all` augmente la précision des requêtes en incluant moins de résultats et, par défaut, « - » est interprété comme « AND NOT ». Par exemple, `wifi -luxury` établit une correspondance avec les documents qui contiennent le terme `wifi` et ne contiennent pas le terme « luxury ». Il s'agit sans doute d'un comportement plus intuitif pour l'opérateur `-`. Ainsi, préférez `searchMode=all` à `searchMode=any` si vous souhaitez optimiser la précision des recherches plutôt que le rappel, *et si vos utilisateurs utilisent fréquemment l'opérateur `-` dans les recherches*.

Lorsque vous choisissez un paramètre `searchMode`, tenez compte des modèles d'interaction utilisateur pour les requêtes liées à différentes applications. Les utilisateurs qui recherchent des informations sont plus enclins à inclure un opérateur dans une requête, contrairement aux sites de commerce électronique qui disposent de structures de navigation intégrées.

## Recherche par champ

Vous pouvez définir une opération de recherche par champ avec la syntaxe `fieldName:searchExpression`, où l'expression de recherche peut être un mot ou une phrase, ou une expression plus complexe entre parenthèses, éventuellement avec des opérateurs booléens. Voici quelques exemples :

- genre:jazz NOT history
- artists:(“Miles Davis” “John Coltrane”)

Veillez à placer les chaînes multiples entre guillemets si vous voulez que les deux chaînes soient évaluées comme une seule entité, comme ici où deux artistes distincts sont recherchés dans le champ `artists`.

Le champ spécifié dans `fieldName:searchExpression` doit être un champ `searchable`. Pour plus d'informations sur l'utilisation des attributs d'index dans les définitions de champs, consultez [Créer un index](#)

#### NOTE

Lorsque vous utilisez des expressions de recherche par champ, il est inutile d'utiliser le paramètre `searchFields`, car chaque expression de recherche par champ a un nom de champ spécifié explicitement. Cependant, vous pouvez toujours utiliser le paramètre `searchFields` si vous voulez exécuter une requête où certaines parties sont limitées à un champ spécifique, et le reste peut s'appliquer à plusieurs champs. Par exemple, la requête `search=genre:jazz NOT history&searchFields=description` ne correspondrait à `jazz` qu'au niveau du champ `genre`, alors qu'elle correspondrait au champ `NOT history` avec le champ `description`. Le nom du champ fourni dans `fieldName:searchExpression` a toujours priorité sur le paramètre `searchFields`, c'est pourquoi dans cet exemple, nous n'avons pas besoin d'inclure `genre` dans le paramètre `searchFields`.

## Recherche approximative

Une recherche approximative recherche des correspondances dans des termes à la construction similaire, en développant un terme jusqu'au maximum de 50 termes qui répondent aux critères de distance de deux ou moins. Pour plus d'informations, consultez [Recherche approximative](#).

Pour effectuer une recherche approximative, utilisez le symbole « ~ » (tilde) à la fin d'un mot avec un paramètre facultatif, un nombre compris entre 0 et 2 (la valeur par défaut), qui spécifie la distance de modification. Par exemple, « blue~ » ou « blue~1 » retournent « blue », « blues » et « glue ».

La recherche partielle est applicable uniquement pour les termes, et non les expressions, mais vous pouvez ajouter le tilde à chaque terme individuel dans un nom en plusieurs parties ou une expression. Par exemple, « Univiersté~ de~ « Wshington~ » correspondrait à « Université de Washington ».

## Recherche de proximité

Les recherches de proximité servent à rechercher des termes qui sont proches les uns des autres dans un document. Insérez un signe tilde « ~ » à la fin d'une expression, suivi du nombre de mots qui créent la limite de proximité. Par exemple, `“hotel airport”~5` recherche les termes « hotel » et « airport » distants de 5 mots ou moins dans un document.

## Promotion de termes

La promotion de termes signifie que vous pouvez accorder à un document un rang plus élevé s'il contient le terme promu, par rapport aux documents qui ne contiennent pas ce terme. À ne pas confondre avec les profils de score qui promeuvent certains champs, plutôt que des termes spécifiques.

L'exemple suivant permet d'illustrer les différences entre les deux. Supposons qu'un profil de score promeut les correspondances dans un certain champ, disons `genre` dans l'[exemple musicstoreindex](#). En

utilisant la promotion de termes, vous pouvez promouvoir encore plus certains termes de recherche. Par exemple, `rock^2 electronic` promeut les documents qui contiennent les termes de recherche dans le champ genre en les classant mieux que d'autres champs de recherche de l'index. Par ailleurs, les documents qui contiennent le terme de recherche *rock* bénéficient d'un meilleur classement que ceux qui contiennent le terme de recherche *electronic* en raison de la valeur de promotion du terme (2).

Pour promouvoir un terme, utilisez le signe « ^ » (caret) avec un facteur de promotion (un nombre) à la fin du terme recherché. Vous pouvez également promouvoir des expressions. Plus le facteur de promotion est élevé, plus le terme est pertinent par rapport aux autres termes de recherche. Par défaut, le facteur de promotion est égal à 1. Ce facteur doit être positif, mais il peut être inférieur à 1 (par exemple 0,20).

## Recherche d'expression régulière

Une recherche d'expression régulière trouve une correspondance en fonction de modèles qui sont valides sous Apache Lucene, comme le décrit la [classe RegExp](#). Dans Recherche cognitive Azure, une expression régulière est placée entre des barres obliques `/`.

Par exemple, pour rechercher des documents contenant « motel » ou « hotel », spécifiez `/[mh]otel/`. Les recherches d'expression régulière se font par comparaison avec des mots individuels.

Certains outils et certaines langues imposent des exigences supplémentaires en matière de caractères d'échappement. Pour JSON, les chaînes qui incluent une barre oblique sont placées dans une séquence d'échappement avec une barre oblique inverse : « microsoft.com/azure/ » devient

`search=/.*microsoft.com\ azure\ .*/` où `search=.* <string-placeholder>.*/` configure l'expression régulière et `microsoft.com\ azure\ \` est la chaîne avec une barre oblique d'échappement.

## Recherche par caractères génériques

Vous pouvez utiliser la syntaxe généralement reconnue pour effectuer des recherches avec plusieurs caractères génériques `*` ou un caractère générique unique `?`. Par exemple, une expression de requête `search=alpha*` retourne « alphanumérique » ou « alphabétique ». Notez que l'Analyseur de requêtes Lucene prend en charge l'utilisation de ces symboles avec un terme unique, et non une expression.

La syntaxe Lucene complète prend en charge la correspondance de préfixe, d'infixe et de suffixe. Toutefois, si tout ce dont vous avez besoin est la correspondance de préfixe, vous pouvez utiliser la syntaxe simple (la correspondance de préfixe est prise en charge dans les deux).

La correspondance de suffixe, où `*` ou `?` précède la chaîne (comme dans `search=.*numeric./`), ou la correspondance d'infixe nécessite une syntaxe Lucene complète ainsi que les barres obliques `/` délimiteurs d'expressions régulières. Vous ne pouvez pas utiliser un signe `*` ou `?` comme premier caractère d'un terme, ou dans un terme, sans `/`.

### NOTE

En règle générale, les critères spéciaux sont lents ; vous préférerez donc peut-être explorer d'autres méthodes, telles que la tokenisation Edge n-Gram qui crée des jetons pour les séquences de caractères d'un terme. L'index sera plus grand, mais les requêtes pourront s'exécuter plus rapidement, en fonction de la construction du modèle et de la longueur des chaînes que vous indexez.

## Impact d'un analyseur sur les requêtes générées

Pendant l'analyse des requêtes, les requêtes qui sont formulées sous forme de préfixe, de suffixe, de caractère générique ou d'expression régulière sont transmises telles quelles à l'arborescence de requête, en ignorant [l'analyse lexicale](#). Les correspondances ne seront trouvées que si l'index contient les chaînes au format spécifié par votre requête. Dans la plupart des cas, vous aurez besoin d'un analyseur pendant

l'indexation qui préserve l'intégrité de la chaîne pour que le terme partiel et les critères spéciaux soient respectés. Pour plus d'informations, consultez [Recherche de termes partiels dans les requêtes Recherche cognitive Azure](#).

Supposons que vous souhaitez que la requête de recherche « terminat\* » retourne des résultats qui contiennent des termes tels que « terminate », « termination » et « terminates ».

Si vous deviez utiliser l'analyseur en.lucene (Lucene anglais), il appliquerait une recherche de radical agressive pour chaque terme. Par exemple, « terminate », « termination », « terminates » seraient tous réduits au jeton « termi » dans votre index. D'un côté, comme les termes dans les requêtes utilisant des caractères génériques ou une recherche approximative ne sont pas analysés du tout, la requête « terminat\* » ne donnerait aucun résultat.

D'un autre côté, les analyseurs Microsoft (dans ce cas, l'analyseur en.microsoft) sont un peu plus avancés et utilisent la lemmatisation et non la recherche de radical. Cela signifie que tous les jetons générés doivent être des mots anglais valides. Par exemple, « terminate », « terminates » et « termination » seront généralement conservés dans leur forme entière dans l'index, ce qui constitue un choix préférable pour les scénarios qui dépendent beaucoup des caractères génériques et de la recherche approximative.

## Scoring des requêtes avec des caractères génériques et des expressions régulières

La Recherche cognitive Azure utilise un scoring basé sur la fréquence ([TF-IDF](#)) pour les requêtes de texte. Cependant, pour les requêtes avec des caractères génériques et des expressions régulières où l'étendue des termes est potentiellement vaste, le facteur de fréquence est ignoré pour empêcher que le classement soit faussé par les correspondances avec des termes plus rares. Toutes les correspondances sont traitées de façon égale pour les recherches avec des caractères génériques et avec des expressions régulières.

## Voir aussi

- [Exemples de requêtes pour une recherche simple](#)
- [Exemples de requêtes pour une recherche Lucene complète](#)
- [Recherche dans des documents](#)
- [Syntaxe des expressions OData pour les filtres et le tri](#)
- [Syntaxe de requête simple dans la Recherche cognitive Azure](#)

# moreLikeThis (préversion) dans Recherche cognitive Azure

04/10/2020 • 4 minutes to read • [Edit Online](#)

## IMPORTANT

Cette fonctionnalité est actuellement disponible en préversion publique. Les fonctionnalités en préversion sont fournies sans contrat de niveau de service et ne sont pas recommandées pour les charges de travail de production. Pour plus d'informations, consultez [Conditions d'Utilisation Supplémentaires relatives aux Évaluations Microsoft Azure](#). L'[API REST version 2020-06-30-Preview](#) fournit cette fonctionnalité. Il n'y a actuellement pas de prise en charge du portail ou du SDK .NET.

`moreLikeThis=[key]` est un paramètre de requête dans [L'API Search Documents](#) (Recherche de documents) qui recherche des documents similaires au document spécifié par la clé de document. Lorsqu'une requête de recherche est formulée avec `moreLikeThis`, une demande est générée en utilisant les termes de recherche extraits du document donné et qui décrivent le mieux ce document. La demande générée est ensuite utilisée pour effectuer la requête de recherche. Par défaut, le contenu de tous les champs pouvant faire l'objet d'une recherche est pris en compte, à l'exception des champs restreints que vous avez spécifiés à l'aide du paramètre `searchFields`. Le paramètre `moreLikeThis` ne peut pas être utilisé avec le paramètre de recherche, `search=[string]`.

Par défaut, le contenu de tous les champs de niveau supérieur pouvant faire l'objet d'une recherche est pris en compte. Si vous souhaitez spécifier des champs spécifiques à la place, vous pouvez utiliser le paramètre `searchFields`.

Vous ne pouvez pas utiliser `MoreLikeThis` sur des sous-champs pouvant faire l'objet d'une recherche dans un [type complexe](#).

## Exemples

Tous les exemples suivants utilisent l'exemple Hôtels tirés du [Guide de démarrage rapide : Créer un index de recherche dans le portail Azure](#).

### Requête simple

La requête suivante recherche les documents dont les champs de description sont les plus proches du champ du document source, tel que spécifié par le paramètre `moreLikeThis`.

```
GET /indexes/hotels-sample-index/docs?moreLikeThis=29&searchFields=Description&api-version=2020-06-30-Preview
```

Dans cet exemple, la requête recherche des hôtels similaires à celui ayant l'`HotelId` 29. Au lieu d'utiliser HTTP GET, vous pouvez également appeler `MoreLikeThis` avec HTTP POST :

```
POST /indexes/hotels-sample-index/docs/search?api-version=2020-06-30-Preview
{
    "moreLikeThis": "29",
    "searchFields": "Description"
}
```

## Appliquer des filtres

`MoreLikeThis` peut être combiné avec d'autres paramètres de requête courants comme `$filter`. Par exemple, la requête peut être limitée à des hôtels dont la catégorie est « budget » et pour lesquels l'évaluation est supérieure à 3,5 :

```
GET /indexes/hotels-sample-index/docs?moreLikeThis=20&searchFields=Description&$filter=(Category eq 'Budget' and Rating gt 3.5)&api-version=2020-06-30-Preview
```

### Sélectionner des champs et limiter les résultats

Le sélecteur `$top` peut être utilisé pour limiter le nombre de résultats devant être retournés dans une requête `MoreLikeThis`. Les champs peuvent aussi être sélectionnés avec `$select`. Ici, les trois premiers hôtels sont sélectionnés avec leur ID, leur nom et leur évaluation :

```
GET /indexes/hotels-sample-index/docs?moreLikeThis=20&searchFields=Description&$filter=(Category eq 'Budget' and Rating gt 3.5)&$top=3&$select=HotelId,HotelName,Rating&api-version=2020-06-30-Preview
```

## Étapes suivantes

Vous pouvez utiliser n'importe quel outil de test web pour faire des essais avec cette fonctionnalité. Nous vous recommandons d'utiliser Postman pour cet exercice.

[Explorer les API REST de Recherche cognitive Azure avec Postman](#)

# Filtres dans la Recherche cognitive Azure

04/10/2020 • 19 minutes to read • [Edit Online](#)

Un *filtre* établit des critères pour la sélection des documents sur lesquels doit porter une requête de Recherche cognitive Azure. Une recherche non filtrée inclut tous les documents figurant dans l'index. Un filtre détermine l'étendue d'une requête de recherche en limitant celle-ci à un sous-ensemble de documents. Par exemple, un filtre peut restreindre une recherche en texte intégral à des produits d'une marque ou d'une couleur spécifiques, commercialisés à des niveaux de prix supérieurs à un seuil défini.

Certaines expériences de recherche imposent des exigences de filtre dans le cadre de leur implémentation, mais vous pouvez utiliser des filtres à chaque fois que vous souhaitez limiter la recherche à l'aide de critères *basés sur des valeurs* (par exemple, en définissant l'étendue de la recherche sur le type de produit « livres » de la catégorie « non-fiction » publiés par « Simon & Schuster »).

Si, au lieu de cela, votre objectif est une recherche ciblée sur des *structures* de données spécifiques (par exemple, en définissant l'étendue de la recherche sur un champ contenant des avis de clients), il existe des méthodes alternatives décrites ci-dessous.

## Quand utiliser un filtre

Certains filtres sont fondamentaux pour diverses expériences de recherche. C'est notamment le cas des filtres de « recherche à proximité », de navigation par facettes et de sécurité, qui montrent uniquement les documents qu'un utilisateur est autorisé à afficher. Si vous implémentez l'une de ces expériences, un filtre est requis. C'est le filtre associé à la requête de recherche qui fournit les coordonnées de géolocalisation, la catégorie de facettes sélectionnée par l'utilisateur, ou l'ID de sécurité du demandeur.

Voici des exemples de scénarios :

1. Utilisez un filtre pour découper votre index sur la base de valeurs de données figurant dans l'index. À partir d'un schéma donné, avec une ville, un type de logement et des commodités, vous pouvez créer un filtre permettant de sélectionner explicitement les documents correspondant à vos critères (par exemple, Marseille, appartement, front de mer).

Une recherche en texte intégral portant sur les mêmes entrées produit souvent des résultats similaires, mais un filtre est plus précis car il requiert une correspondance exacte entre la condition de filtre et le contenu de votre index.

2. Utilisez un filtre si l'expérience de recherche intègre une condition de filtre :

- La [navigation par facettes](#) utilise un filtre pour re-transmettre la catégorie de facettes sélectionnée par l'utilisateur.
- La recherche basée sur la géolocalisation utilise un filtre pour transmettre les coordonnées de la localisation actuelle dans des applications de « recherche à proximité ».
- Les filtres de sécurité transmettent des identificateurs de sécurité en tant que critères de filtre, où une correspondance dans l'index sert de proxy pour les droits d'accès au document.

3. Utilisez un filtre si vous souhaitez appliquer des critères de recherche sur un champ numérique.

Les champs numériques sont récupérables dans le document et peuvent apparaître dans des résultats de recherche, mais ils ne peuvent pas faire l'objet individuellement d'une recherche en texte intégral. Si vous avez besoin de critères de sélection basés sur des données numériques, utilisez un filtre.

## Autres méthodes de réduction de l'étendue de recherche

Si vous souhaitez restreindre vos résultats de recherche, les filtres ne sont pas le seul choix possible. Les solutions alternatives suivantes peuvent être plus appropriées selon votre objectif :

- Le paramètre de requête `searchFields` permet de focaliser une recherche sur des champs spécifiques. Par exemple, si votre index comprend des champs distincts pour les descriptions en anglais et en espagnol, le paramètre `searchFields` vous permet de cibler les champs à utiliser pour une recherche en texte intégral.
- Le paramètre `$select` permet de spécifier les champs à inclure dans un jeu de résultats, ce qui a pour effet de réduire la réponse avant l'envoi de celle-ci à l'application appelante. Ce paramètre n'affine pas la requête et ne réduit pas la collection de documents. En revanche, si votre objectif est d'obtenir une plus petite réponse, ce paramètre constitue une option à envisager.

Pour plus d'informations sur ces deux paramètres, voir [Rechercher des documents > Demande > Paramètres de requête](#).

## Comment les filtres sont-ils exécutés ?

Au moment de la requête, un analyseur de filtre accepte les critères en entrée, convertit l'expression en expressions booléennes atomiques sous la forme d'une arborescence de filtres qui est ensuite évaluée sur les champs filtrables dans un index.

Le filtrage se produit en même temps que la recherche. Il permet de qualifier les documents à inclure dans le traitement en aval pour la récupération de documents et le scoring de leur pertinence. En association avec une chaîne de recherche, le filtre réduit efficacement l'ensemble de rappels de l'opération de recherche suivante. Utilisé seul (par exemple, lorsque la chaîne de requête est vide, où `search=*`), le critère de filtre est la seule entrée.

## Définition des filtres

Les filtres sont des expressions OData articulées à l'aide d'un [sous-ensemble de la syntaxe OData V4 prise en charge dans la Recherche cognitive Azure](#).

Vous pouvez spécifier un filtre pour chaque opération de **recherche**, mais le filtre lui-même peut inclure plusieurs champs, plusieurs critères et, si vous utilisez une fonction `ismatch`, plusieurs expressions de recherche en texte intégral. Dans une expression de filtre comportant plusieurs parties, vous pouvez spécifier des prédictats dans n'importe quel ordre (soumis aux règles de précédence de l'opérateur). Vous n'obtenez aucun gain sensible des performances si vous tentez de réorganiser les prédictats dans une séquence particulière.

L'une des limites inconditionnelles sur une expression de filtre est la limite de taille maximale de la demande. La demande entière, filtre inclus, peut être un maximum de 16 Mo pour la commande POST ou de 8 Ko pour la commande GET. Le nombre de clauses dans votre expression de filtre est également limité. Une règle empirique est que, si vous avez des centaines de clauses, vous risquez d'atteindre la limite. Nous vous recommandons de concevoir votre application de telle sorte qu'elle ne génère pas de filtres de taille illimitée.

Les exemples suivants illustrent des définitions de filtre prototypiques dans plusieurs API.

```

# Option 1: Use $filter for GET
GET https://[service name].search.windows.net/indexes/hotels/docs?api-version=2020-06-30&search=*&$filter=Rooms/any(room: room/BaseRate lt 150.0)&$select=HotelId, HotelName, Rooms/Description, Rooms/BaseRate

# Option 2: Use filter for POST and pass it in the request body
POST https://[service name].search.windows.net/indexes/hotels/docs/search?api-version=2020-06-30
{
    "search": "*",
    "filter": "Rooms/any(room: room/BaseRate lt 150.0)",
    "select": "HotelId, HotelName, Rooms/Description, Rooms/BaseRate"
}

```

```

parameters =
    new SearchParameters()
    {
        Filter = "Rooms/any(room: room/BaseRate lt 150.0)",
        Select = new[] { "HotelId", "HotelName", "Rooms/Description" , "Rooms/BaseRate" }
    };

var results = searchIndexClient.Documents.Search("*", parameters);

```

## Filtrer les modèles d'utilisation

Les exemples suivants illustrent plusieurs modèles d'utilisation pour des scénarios de filtre. Pour d'autres idées, voir [Syntaxe d'expression OData > Exemples](#).

- **\$filter autonome**, sans chaîne de requête, utile lorsque l'expression de filtre est en mesure de qualifier complètement les documents d'intérêt. À défaut de chaîne de requête, il n'y a ni analyse lexicale ou linguistique, ni notation, ni classement. Notez que la chaîne de recherche comporte uniquement un astérisque, ce qui signifie « faire correspondre tous les documents ».

```
search=*&$filter=Rooms/any(room: room/BaseRate ge 60 and room/BaseRate lt 300) and Address/City eq 'Honolulu'
```

- Combinaison de chaîne de requête et de **\$filter**, où le filtre crée le sous-ensemble, et la chaîne de requête fournit les entrées de condition de recherche en texte intégral sur le sous-ensemble filtré. L'ajout de termes (théâtres à distance de marche) introduit des scores de recherche dans les résultats, où les documents qui correspondent le mieux aux termes sont mieux classés. Utiliser un filtre avec une chaîne de requête constitue le modèle d'utilisation le plus courant.

```
search=walking distance theaters&$filter=Rooms/any(room: room/BaseRate ge 60 and room/BaseRate lt 300) and Address/City eq 'Seattle'&$count=true
```

- Requêtes composées, séparées par « OR » (ou), chacune avec ses propres critères de filtre (par exemple, « beagle » dans « chien » ou « siamois » dans « chat »). Les expressions combinées utilisant `or` sont évaluées individuellement et la correspondance des documents joints avec chaque expression est retournée avec la réponse. Ce modèle d'utilisation est obtenu via la fonction `search.ismatchscoring`. Vous pouvez également utiliser la version sans scoring, `search.ismatch`.

```

# Match on hostels rated higher than 4 OR 5-star motels.
$filter=search.ismatchscoring('hostel') and Rating ge 4 or search.ismatchscoring('motel') and Rating
eq 5

# Match on 'luxury' or 'high-end' in the description field OR on category exactly equal to 'Luxury'.
$filter=search.ismatchscoring('luxury | high-end', 'Description') or Category eq 'Luxury'&$count=true

```

Il est également possible de combiner la recherche en texte intégral via `search.ismatchscoring` avec des filtres utilisant `and` au lieu de `or`. Toutefois, cette opération équivaut à utiliser les paramètres `search` et `$filter` dans une demande de recherche sur le plan fonctionnel. Par exemple, les deux requêtes suivantes génèrent le même résultat :

```

$filter=search.ismatchscoring('pool') and Rating ge 4

search=pool&$filter=Rating ge 4

```

Pour obtenir des instructions complètes sur des cas d'usage spécifiques, consultez les articles suivants :

- [Filtres de facette](#)
- [Filtres de langage](#)
- [Filtrage de sécurité](#)

## Conditions requises des champs pour le filtrage

Dans l'API REST, la propriété `filterable` (`filtrable`) est *activée* par défaut pour les champs simples. Les champs filtrables augmentent la taille de l'index. Veillez à définir `"filterable": false` pour les champs que vous ne prévoyez pas réellement d'utiliser dans un filtre. Pour plus d'informations sur les paramètres des définitions de champ, voir [Create Index](#) (Créer un index).

Dans le Kit de développement logiciel (SDK) .NET, la propriété `filterable` (`filtrable`) est *désactivée* par défaut. Vous pouvez rendre un champ filtrable en définissant la [propriété `IsFilterable`](#) de l'objet [Champ](#) correspondant sur `true`. Vous pouvez aussi effectuer cette opération de façon déclarative à l'aide de [l'attribut `IsFilterable`](#). Dans l'exemple ci-dessous, l'attribut est défini sur la propriété `BaseRate` d'une classe de modèle mappant vers la définition d'index.

```

[IsFilterable, IsSortable, IsFacetable]
public double? BaseRate { get; set; }

```

### Rendre filtrable un champ existant

Vous ne pouvez pas modifier des champs existants pour les rendre filtrables. À la place, vous devez ajouter un nouveau champ ou régénérer l'index. Pour plus d'informations sur la régénération d'un index ou la façon de remplir à nouveau des champs, consultez [Comment régénérer un index de Recherche cognitive Azure](#).

## Notions de base concernant les filtres de texte

Les filtres de texte comparent les champs de chaîne aux chaînes littérales que vous fournissez dans le filtre. Contrairement à la recherche en texte intégral, les filtres de texte n'appliquent aucune analyse lexicale ou césure de mots. Les comparaisons portent alors uniquement sur des correspondances exactes. Par exemple, supposons un champ `f` contenant les mots « Sunny day » (journée ensoleillée). `$filter=f eq 'sunny day'` sera une correspondance, mais pas `$filter=f eq 'Sunny'`.

Les chaînes de texte respectent la casse. Il n'y a pas de conversion en minuscules des mots contenant des majuscules. Ainsi, la chaîne `$filter=f eq 'Sunny day'` ne permet pas de trouver « sunny day ».

## Approches pour le filtrage de texte

APPROCHE	DESCRIPTION	QUAND L'UTILISER
<code>search.in</code>	Une fonction qui compare un champ à une liste délimitée de chaînes.	Cette fonction est recommandée pour les <a href="#">filtres de sécurité</a> et pour tous les filtres dans lesquels plusieurs valeurs de texte brut doivent être comparées à un champ de chaîne. La fonction <code>search.in</code> est conçue pour fonctionner rapidement. Elle est donc beaucoup plus rapide qu'une comparaison explicite du champ à chaque chaîne à l'aide de <code>eq</code> et <code>or</code> .
<code>search.ismatch</code>	Fonction permettant de combiner des opérations de recherche en texte intégral avec des opérations de filtre strictement booléen dans une même expression de filtre.	Utilisez <code>search.ismatch</code> (ou son équivalent <code>search.ismatchscoring</code> pour le scoring) lorsque vous souhaitez utiliser plusieurs combinaisons de filtres et de recherches dans une seule demande. Vous pouvez également l'utiliser pour un filtre <code>contains</code> afin de filtrer sur une chaîne partielle figurant à l'intérieur d'une chaîne de plus grande taille.
<code>\$filter=field operator string</code>	Expression définie par l'utilisateur composée de champs, d'opérateurs et de valeurs.	Utilisez-la lorsque vous souhaitez rechercher des correspondances exactes entre un champ de chaîne et une valeur de chaîne.

## Notions de base concernant les filtres numériques

Les champs numériques ne sont pas de type `searchable` (il n'est pas possible d'y effectuer une recherche) dans le contexte d'une recherche en texte intégral. Seules des chaînes peuvent faire l'objet d'une recherche en texte intégral. Par exemple, le terme de recherche 99,99 ne permet pas d'obtenir la liste des articles dont le prix est 99,99. Au lieu de cela, vous obtenez la liste des documents dont des champs de chaîne contiennent le nombre 99. Par conséquent, si vous avez des données numériques, l'hypothèse est que vous allez les utiliser pour des filtres, dont des plages, des facettes, des groupes, etc.

Les documents contenant des champs numériques (prix, taille, référence, ID, etc.) fournissent ces valeurs dans les résultats de recherche si ces champs sont marqués en tant que `retrievable`. Ce qu'il importe de souligner ici, c'est que la recherche en texte intégral proprement dite ne s'applique pas aux champs de type numérique.

## Étapes suivantes

Tout d'abord, essayez l'[Explorateur de recherche](#) dans le portail pour soumettre des requêtes avec des paramètres de `$filter`. L'[index d'exemple immobilier](#) fournit des résultats intéressants pour les requêtes filtrées suivantes lorsque vous les collez dans la barre de recherche :

```
# Geo-filter returning documents within 5 kilometers of Redmond, Washington state
# Use $count=true to get a number of hits returned by the query
# Use $select to trim results, showing values for named fields only
# Use search=* for an empty query string. The filter is the sole input

search=*&$count=true&$select=description,city,postCode&$filter=geo.distance(location,geography'POINT(-122.121513 47.673988)') le 5

# Numeric filters use comparison like greater than (gt), less than (lt), not equal (ne)
# Include "and" to filter on multiple fields (baths and bed)
# Full text search is on John Leclerc, matching on John or Leclerc

search=John Leclerc&$count=true&$select=source,city,postCode,baths,beds&$filter=baths gt 3 and beds gt 4

# Text filters can also use comparison operators
# Wrap text in single or double quotes and use the correct case
# Full text search is on John Leclerc, matching on John or Leclerc

search=John Leclerc&$count=true&$select=source,city,postCode,baths,beds&$filter=city gt 'Seattle'
```

Pour utiliser d'autres exemples, voir [Syntaxe d'expression de filtre OData > Exemples](#).

## Voir aussi

- [Fonctionnement de la recherche en texte intégral dans la Recherche cognitive Azure](#)
- [API REST de recherche de documents](#)
- [Syntaxe de requête simple](#)
- [Syntaxe de requête Lucene](#)
- [Types de données prises en charge](#)

# Comment créer un filtre de facette dans la Recherche cognitive Azure

04/10/2020 • 10 minutes to read • [Edit Online](#)

La navigation par facettes est utilisée pour le filtrage autonome sur les résultats d'une requête dans une application de recherche, dans le cadre de laquelle votre application propose des contrôles d'interface utilisateur permettant de limiter une recherche à des groupes de documents (par exemple, des catégories ou des marques) ; la Recherche cognitive Azure fournit la structure de données sur laquelle reposera l'opération. Dans cet article, nous passerons rapidement en revue les étapes de base de la procédure de création d'une structure de navigation par facettes, sur laquelle sera basée l'expérience de recherche que vous souhaitez proposer.

- Choisissez des champs pour le filtrage et la création de facettes
- Définissez des attributs sur le champ
- Créez l'index et chargez les données
- Ajoutez des filtres de facettes à une requête
- Gérez les résultats

Les facettes sont dynamiques et renvoyées sur une requête. Les réponses associées à une recherche incluent les catégories de facettes utilisées pour parcourir les résultats. Si vous n'êtes pas familiarisé avec les facettes, l'exemple suivant illustre une structure de navigation par facettes.

The screenshot shows a search interface with the following sections:

- SEARCH**: Contains a search bar labeled "Search Jobs" and a dropdown field "Any distance from" with the value "10001".
- FILTER RESULTS**: Contains a section titled "BUSINESS TITLE" with three items:
  - > Auditor (20)
  - > Project Manager (20)
  - > Agency Attorney (14)

A large yellow arrow points from the word "Facets" to the "BUSINESS TITLE" section, highlighting the facet navigation feature.

Vous découvrez ce type de navigation et souhaitez en savoir plus ? Consultez [Implémentation de la navigation par facettes dans la Recherche cognitive Azure](#).

## Choisir des champs

Vous pouvez calculer des facettes sur la base de champs à une seule valeur, ou de collections. Les champs les plus efficaces dans une navigation par facettes présentent une cardinalité faible (un petit nombre de valeurs distinctes, qui se répètent tout au long d'un document dans votre corpus de recherche, par exemple une liste de couleurs, de pays/régions ou de noms de marques).

La création de facettes est activée champ par champ lorsque vous génez l'index, si vous définissez l'attribut `facetable` sur `true`. Vous devez généralement définir également l'attribut `filterable` sur `true` pour ces champs, afin que votre application de recherche puisse filtrer sur ces champs en fonction des facettes

sélectionnées par l'utilisateur final.

Lors de la création d'un index à l'aide de l'API REST, tout **type de champ** pouvant éventuellement être utilisé dans la navigation à facettes est marqué comme **facetable** par défaut :

- `Edm.String`
- `Edm.DateTimeOffset`
- `Edm.Boolean`
- Types de champs numériques : `Edm.Int32`, `Edm.Int64`, `Edm.Double`
- Collections des types ci-dessus (par exemple, `Collection(Edm.String)` ou `Collection(Edm.Double)`)

Vous ne pouvez pas utiliser les champs `Edm.GeographyPoint` ou `Collection(Edm.GeographyPoint)` dans la navigation par facettes. Les facettes fonctionnent mieux sur les champs présentant une faible cardinalité. En raison de la résolution des coordonnées géographiques, il est rare que les deux jeux de coordonnées soient égaux dans un jeu de données. Par conséquent, elles ne sont pas prises en charge pour les coordonnées géographiques. Vous aurez besoin d'un champ de ville ou de région pour les facettes créées par lieu.

## Définir des attributs

Les attributs d'index qui contrôlent l'utilisation d'un champ sont ajoutés aux définitions de champs individuels dans l'index. Dans l'exemple suivant, les champs présentant une faible cardinalité, utiles pour les facettes, incluent une `category` (hôtel, motel, auberge de jeunesse...), des `tags` et des `rating`. Les attributs `filterable` et `facetable` de ces champs sont définis explicitement dans l'exemple suivant à titre d'illustration.

### TIP

Pour optimiser le stockage et les performances, une meilleure pratique consiste à désactiver la création de facettes pour les champs qui ne doivent pas être utilisés en tant que facettes. En particulier, les champs de chaîne pour les valeurs uniques, comme un ID ou un nom de produit, doivent être définis sur `"facetable": false` pour empêcher leur utilisation accidentelle (et inefficace) dans la navigation à facettes.

```
{
  "name": "hotels",
  "fields": [
    { "name": "hotelId", "type": "Edm.String", "key": true, "searchable": false, "sortable": false,
      "facetable": false },
    { "name": "baseRate", "type": "Edm.Double" },
    { "name": "description", "type": "Edm.String", "filterable": false, "sortable": false, "facetable": false },
    { "name": "description_fr", "type": "Edm.String", "filterable": false, "sortable": false, "facetable": false,
      "analyzer": "fr.lucene" },
    { "name": "hotelName", "type": "Edm.String", "facetable": false },
    { "name": "category", "type": "Edm.String", "filterable": true, "facetable": true },
    { "name": "tags", "type": "Collection(Edm.String)", "filterable": true, "facetable": true },
    { "name": "parkingIncluded", "type": "Edm.Boolean", "filterable": true, "facetable": true, "sortable": false },
    { "name": "smokingAllowed", "type": "Edm.Boolean", "filterable": true, "facetable": true, "sortable": false },
    { "name": "lastRenovationDate", "type": "Edm.DateTimeOffset" },
    { "name": "rating", "type": "Edm.Int32", "filterable": true, "facetable": true },
    { "name": "location", "type": "Edm.GeographyPoint" }
  ]
}
```

#### NOTE

Cette définition d'index est copiée à partir de la section relative à la [création d'un index de Recherche cognitive Azure à l'aide de l'API REST](#). Il est identique, à l'exception de légères différences dans les définitions de champ. Les attributs `filterable` et `facetable` sont ajoutés de manière explicite sur les champs `category`, `tags`, `parkingIncluded`, `smokingAllowed`, et `rating`. Dans la pratique, `filterable` et `facetable` seraient activés par défaut sur ces champs lorsque vous utilisez l'API REST. Lorsque vous utilisez le Kit de développement logiciel (SDK) .NET, ces attributs doivent être activés explicitement.

## Créer et charger un index

La [création et le remplissage de l'index](#) sont une étape intermédiaire (et peut-être évidente) avant la formulation d'une requête. Nous signalons cette étape ici par souci d'exhaustivité. Une manière de déterminer si l'index est disponible consiste à vérifier la liste des index dans le [portail](#).

## Ajoutez des filtres de facettes à une requête

Dans le code d'application, construisez une requête spécifiant toutes les parties d'une requête valide, y compris les expressions de recherche, les facettes, les filtres, les profils de score, à savoir tout ce qui sert à formuler une requête. L'exemple suivant génère une requête qui crée une navigation par facettes selon le type d'hébergement, d'évaluation et d'autres équipements.

```
var sp = new SearchParameters()
{
    ...
    // Add facets
    Facets = new[] { "category", "rating", "parkingIncluded", "smokingAllowed" }.ToList()
};
```

### Renvoyer des résultats filtrés sur les événements clic

Lorsque l'utilisateur final clique sur une valeur de facette, le gestionnaire de l'événement click doit utiliser une expression de filtre pour comprendre l'intention de l'utilisateur. Si vous disposez d'une facette `category`, un clic sur la catégorie `$filter` est implémenté avec une expression qui sélectionne les hébergements de ce type. Lorsqu'un utilisateur clique sur « motels » pour indiquer que seul ce type d'hébergement doit être affiché, la requête suivante que l'application envoie inclut la chaîne `$filter=category eq 'motel'`.

L'extrait de code suivant ajoute la catégorie au filtre si l'utilisateur sélectionne une valeur à partir de la facette « Category ».

```
if (!String.IsNullOrEmpty(categoryFacet))
    filter = "$category eq '{categoryFacet}'";
```

Si l'utilisateur clique sur une valeur de facette pour un champ de la collection comme `tags`, par exemple le « pool » de valeurs, l'application doit utiliser la syntaxe de filtre suivante : `$filter=tags/any(t: t eq 'pool')`

## Conseils et solutions de contournement

### Initialiser une page sur laquelle des facettes sont définies

Si vous souhaitez initialiser une page sur laquelle des facettes sont définies, vous pouvez envoyer une requête dans le cadre de l'initialisation de la page, afin d'amorcer la page avec une structure de facette initiale.

### Conserver une structure de navigation par facettes de manière asynchrone par rapport aux résultats filtrés

L'une des difficultés liées à la navigation par facettes dans la Recherche cognitive Azure est le fait que les facettes

existent uniquement pour les résultats actuels. Dans la pratique, il est courant de conserver un ensemble statique de facettes, afin que l'utilisateur puisse naviguer dans l'ordre inverse, en suivant les étapes déjà parcourues pour explorer les autres chemins d'accès via le contenu de la recherche.

Il s'agit d'un cas d'usage courant, mais non fourni de manière prête à l'emploi par la structure de navigation par facettes. Les développeurs qui veulent utiliser des facettes statiques contournent cette limitation en émettant deux requêtes filtrées, l'une dont la portée se limite aux résultats et l'autre, qui permet de créer une liste statique de facettes pour la navigation.

## Voir aussi

- [Filtres dans la Recherche cognitive Azure](#)
- [Création d'une API REST d'index](#)
- [API REST de recherche de documents](#)

# Implémentation de la navigation par facettes dans la Recherche cognitive Azure

04/10/2020 • 47 minutes to read • [Edit Online](#)

La navigation à facettes est un mécanisme de filtrage qui fournit une navigation autonome d'extraction dans les applications de recherche. Le terme « navigation à facettes » peut vous sembler peu familier, mais vous l'avez très certainement déjà utilisé. Comme l'indique l'exemple ci-dessous, la navigation à facettes correspond tout simplement aux catégories utilisées pour filtrer les résultats.

The screenshot shows the Azure Search Job Portal Demo interface. At the top, there's a header with the Azure Search logo and the text "AVAILABLE JOBS (2802 jobs)". Below the header, there are search and filter sections. The "SEARCH" section includes a search bar and a dropdown for "Any distance from" set to 10001. The "FILTER RESULTS" section has two expandable categories: "BUSINESS TITLE" and "LOCATION". The "BUSINESS TITLE" category is highlighted with a red border and contains a list of job titles with their counts: Auditor (20), Project Manager (20), Agency Attorney (14), College Aide (14), Assistant Corporation Counsel (12), Construction Project Manager Intern (10), Administrative Assistant (8), Community Coordinator (8), Procurement Analyst (8), and Watershed Maintainer (8). The "LOCATION" category lists Internal (1469) and External (1333). To the right of these filters is a map of New York City with several orange dots indicating job locations. A red callout bubble labeled "FACETS" points to the "BUSINESS TITLE" filter. Below the map, the text "2802 AVAILABLE JOBS" is displayed. Underneath, there's a job listing for a Java Developer at Metro Tech 4Th Flr, Rm 418, with a salary range of \$81,290 to \$100,000 Annual. The listing includes a small thumbnail image of two people, a date (Jan), and a "Read More" link. At the bottom, there's a relevance dropdown and a page navigation menu.

La navigation à facettes constitue un autre point d'entrée pour la recherche. Elle offre une alternative pratique à la saisie manuelle d'expressions de recherche complexes. Les facettes peuvent vous aider à trouver ce que vous recherchez, tout en vous assurant d'obtenir au moins un résultat. En tant que développeur, les facettes vous permettent d'exposer les critères de recherche les plus utiles pour naviguer dans votre index de recherche. Dans les applications de vente au détail en ligne, la navigation à facettes repose souvent sur les marques, les catégories (chaussures pour enfants), la taille, le prix, la popularité et les évaluations.

L'implémentation de la navigation par facettes varie en fonction des technologies de recherche. Dans la Recherche cognitive Azure, la navigation par facettes est créée au moment de la requête, à l'aide de champs précédemment attribués dans votre schéma.

- Dans les requêtes créées par votre application, une requête doit envoyer des *paramètres de requête de facette* afin de recevoir les valeurs de filtre de facette disponibles pour ce jeu de résultats du document.
- Pour réduire réellement le jeu de résultats du document, l'application doit également appliquer une expression `$filter`.

Dans le développement de votre application, l'écriture du code qui construit les requêtes constitue la majeure

partie du travail. La plupart des comportements de l'application que vous attendez de la navigation par facettes sont fournis par le service, y compris la prise en charge intégrée de la définition des plages et la récupération des décomptes pour les résultats de facettes. Le service comprend également des valeurs par défaut qui vous aident à éviter les structures de navigation difficiles à gérer.

## Exemple de code et démonstration

Cet article prend l'exemple d'un portail de recrutement. L'exemple est implémenté en tant qu'application ASP.NET MVC.

- Consultez et testez la [démonstration en ligne du portail de recrutement de la Recherche cognitive Azure](#).
- Téléchargez le code à partir du [référentiel Azure-Samples sur GitHub](#).

## Bien démarrer

Si vous êtes novice en recherche et développement, considérez que la navigation par facettes affiche les possibilités de recherche autonome. Il s'agit d'un type d'expérience de recherche détaillée, en fonction de filtres prédéfinis, utilisés pour limiter rapidement les résultats de la recherche à l'aide d'actions de type pointer et cliquer.

### Modèle d'interaction

L'expérience de recherche pour la navigation à facettes est itérative. Donc, essayons de la comprendre comme séquence de requêtes qui se déroulent en réponse aux actions de l'utilisateur.

Le point de départ est une page d'application qui offre une navigation à facettes, généralement placée sur la périphérie. La navigation à facettes est souvent présentée sous forme d'arborescence avec des cases à cocher pour chaque valeur ou du texte interactif.

1. Une requête envoyée à la Recherche cognitive Azure spécifie la structure de la navigation par facettes par le biais d'un ou plusieurs paramètres de requête de facette. Par exemple, la requête peut inclure `facet=Rating`, éventuellement avec une option `:values` ou `:sort` pour affiner la présentation.
2. La couche de présentation renvoie une page de recherche qui fournit une navigation à facettes, à l'aide des facettes spécifiées dans la requête.
3. Face à une structure de navigation à facettes qui inclut le paramètre Évaluation, vous cliquez sur « 4 » pour indiquer que seuls les produits dotés d'une évaluation minimale de 4 doivent être affichés.
4. En réponse, l'application envoie une requête qui inclut `$filter=Rating ge 4`
5. La couche de présentation met à jour la page en affichant un jeu de résultats réduit, contenant uniquement les éléments qui répondent aux nouveaux critères (dans ce cas, les produits avec une évaluation de 4 et supérieure).

Une facette est un paramètre de requête, mais ne la confondez pas avec l'entrée de requête. Elle n'est jamais utilisée comme critère de sélection dans une requête. Considérez plutôt les paramètres de requête de facette comme des entrées de la structure de navigation qui est renvoyée dans la réponse. Pour chaque paramètre de requête de facette que vous fournissez, la Recherche cognitive Azure évalue le nombre de documents dans les résultats partiels pour chaque valeur de la facette.

Notez le `$filter` à l'étape 4. Ce filtre constitue un aspect important de la navigation à facettes. Bien que les facettes et les filtres soient indépendants dans l'API, vous avez besoin des deux pour fournir l'expérience attendue.

### Modèle de conception de l'application

Dans le code d'application, le modèle consiste à utiliser les paramètres de requête de facette pour renvoyer la structure de navigation à facettes, ainsi que les résultats de la facette, mais aussi une expression `$filter`.

L'expression de filtre gère l'événement clic sur la valeur de facette. Considérez l'expression `$filter` comme le code derrière la réduction réelle des résultats de la recherche renvoyés à la couche de présentation. Dans le cas

d'une facette Couleurs, le fait de cliquer sur la couleur Rouge est implémenté par le biais d'une expression `$filter` qui sélectionne uniquement les éléments qui ont une couleur rouge.

## Principes de base des requêtes

Dans la Recherche cognitive Azure, une requête est spécifiée par le biais d'un ou de plusieurs paramètres de requête (consultez [Rechercher des documents](#) pour obtenir une description de chacun d'eux). Aucun des paramètres de requête n'est requis, mais vous devez en avoir au moins un pour qu'une requête soit valide.

La précision, interprétée comme la possibilité de filtrer les résultats non pertinents, s'effectue par le biais d'une ou de ces deux expressions :

- **search=**

La valeur de ce paramètre constitue l'expression de recherche. Il peut s'agir d'une portion de texte unique ou d'une expression de recherche complexe qui comprend plusieurs termes et opérateurs. Sur le serveur, une expression de recherche est utilisée pour la recherche en texte intégral. Elle interroge les champs pouvant faire l'objet d'une recherche dans l'index pour la correspondance des termes et renvoie les résultats classés. Si vous définissez `search` sur null, l'exécution de la requête est effectuée sur la totalité de l'index (c'est-à-dire, `search=*`). Dans ce cas, d'autres éléments de la requête, comme `$filter` ou un profil de score, sont les principaux facteurs qui influencent les documents renvoyés (`($filter)` ainsi que leur ordre (`scoringProfile` ou `$orderby`)).

- **\$filter =**

Un filtre est un mécanisme puissant pour limiter la taille des résultats de la recherche basés sur les valeurs des attributs de document spécifiques. Un `$filter` est évalué en premier, suivi de la logique de facettes qui génère les valeurs disponibles et les décomptes correspondants pour chaque valeur.

Les expressions de recherche complexes diminuent les performances de la requête. Si possible, utilisez des expressions de filtre bien construites pour accroître la précision et améliorer les performances de la requête.

Pour mieux comprendre comment un filtre ajoute plus de précision, comparez une expression de recherche complexe à une expression qui contient une expression de filtre :

- `GET /indexes/hotel/docs?search=lodging budget +Seattle -motel +parking`
- `GET /indexes/hotel/docs?search=lodging&$filter=City eq 'Seattle' and Parking and Type ne 'motel'`

Les deux requêtes sont valides, mais la seconde est préférable si vous cherchez des établissements autres que des « motels » avec stationnement à Seattle.

- La première requête repose sur ces mots spécifiques étant mentionnés ou non mentionnés dans des champs de type chaîne comme Nom, Description et tout autre champ contenant des données pouvant faire l'objet d'une recherche.
- La deuxième requête recherche des correspondances précises sur des données structurées et est susceptible d'être beaucoup plus précise.

Dans les applications qui incluent la navigation à facettes, veillez à ce que chaque action de l'utilisateur sur une structure de navigation à facettes soit accompagnée d'une réduction du champ des résultats de recherche. Pour affiner les résultats, utilisez une expression de filtre.

## Créer une application de navigation à facettes

Vous implémentez la navigation par facettes avec la Recherche cognitive Azure dans le code d'application qui vous permet de créer la requête de recherche. La navigation à facettes repose sur les éléments de schéma que vous avez définis précédemment.

L'attribut d'index `Facetable [true|false]`, prédéfini sur votre index de recherche, est défini sur des champs sélectionnés pour activer ou désactiver leur utilisation dans une structure de navigation à facettes. Sans

`"Facetable" = true`, un champ ne peut pas être utilisé dans la navigation à facettes.

La couche de présentation dans votre code fournit l'expérience utilisateur. Elle doit répertorier les éléments constitutifs de la navigation à facettes, comme l'étiquette, les valeurs, les cases à cocher et le décompte. L'API REST de Recherche cognitive Azure est indépendante de la plateforme. Vous pouvez donc utiliser la langue et la plateforme que vous souhaitez. L'important est d'inclure des éléments d'interface utilisateur qui prennent en charge l'actualisation incrémentielle, avec l'état de l'interface utilisateur mis à jour lorsque chaque facette supplémentaire est sélectionnée.

Au moment de la requête, votre code d'application crée une requête qui inclut `facet=[string]`, un paramètre de requête qui fournit le champ sur lequel baser la facette. Une requête peut avoir plusieurs facettes, comme `&facet=color&facet=category&facet=rating`, chaque facette étant séparée par un caractère d'espacement (&).

Le code d'application doit également construire une expression `$filter` pour gérer les événements de clic dans la navigation à facettes. Un `$filter` réduit les résultats de la recherche, à l'aide de la valeur de facette comme critère de filtre.

La Recherche cognitive Azure retourne les résultats de la recherche en fonction du ou des mots que vous avez saisis ainsi que les mises à jour de la structure de la navigation par facettes. Dans la Recherche cognitive Azure, la navigation par facettes est une construction à niveau unique, avec des valeurs de facettes et des décomptes des résultats trouvés pour chaque.

Dans les sections suivantes, nous allons étudier comment générer chaque partie.

## Création de l'index

Les facettes sont activées sur une base de champ par champ dans l'index, au moyen de cet attribut d'index :

`"Facetable": true`.

Tous les types de champs pouvant être utilisés dans la navigation à facettes sont `Facetable` par défaut. Ces types de champs incluent `Edm.String`, `Edm.DateTimeOffset` et tous les types de champs numériques (globalement, tous les types de champs peuvent être utilisés comme facettes, sauf `Edm.GeographyPoint` qui ne peut pas être utilisé dans la navigation à facettes).

Lorsque vous créez un index, nous vous recommandons, pour la navigation à facettes, de désactiver explicitement les facettes pour les champs qui ne doivent jamais être utilisés comme facettes. En particulier, les champs de chaîne pour les valeurs singleton, comme un ID ou un nom de produit, doivent être définis sur `"Facetable": false` pour empêcher leur utilisation accidentelle (et inefficace) dans la navigation à facettes. Le fait de désactiver les facettes inutiles permet de conserver une taille d'index réduite, ce qui améliore généralement les performances.

Vous trouverez ci-dessous une partie du schéma de l'exemple d'application de démonstration du portail de recrutement, sans certains attributs pour en réduire la taille :

```

{
  ...
  "name": "nycjobs",
  "fields": [
    { "name": "id",           "type": "Edm.String",      "searchable": false, "filterable": false, ... "facetable": false, ... },
    { "name": "job_id",        "type": "Edm.String",      "searchable": false, "filterable": false, ... "facetable": false, ... },
    { "name": "agency",         "type": "Edm.String",      "searchable": true,  "filterable": true, ... "facetable": true, ... },
    { "name": "posting_type",   "type": "Edm.String",      "searchable": true,  "filterable": true, ... "facetable": true, ... },
    { "name": "num_of_positions", "type": "Edm.Int32",     "searchable": false, "filterable": true, ... "facetable": true, ... },
    { "name": "business_title", "type": "Edm.String",      "searchable": true,  "filterable": true, ... "facetable": true, ... },
    { "name": "civil_service_title", "type": "Edm.String",      "searchable": true,  "filterable": true, ... "facetable": true, ... },
    { "name": "title_code_no",    "type": "Edm.String",      "searchable": true,  "filterable": true, ... "facetable": true, ... },
    { "name": "level",           "type": "Edm.String",      "searchable": true,  "filterable": true, ... "facetable": true, ... },
    { "name": "salary_range_from", "type": "Edm.Int32",     "searchable": false, "filterable": true, ... "facetable": true, ... },
    { "name": "salary_range_to",  "type": "Edm.Int32",     "searchable": false, "filterable": true, ... "facetable": true, ... },
    { "name": "salary_frequency", "type": "Edm.String",      "searchable": true,  "filterable": true, ... "facetable": true, ... },
    { "name": "work_location",    "type": "Edm.String",      "searchable": true,  "filterable": true, ... "facetable": true, ... },
    ...
    { "name": "geo_location",    "type": "Edm.GeographyPoint", "searchable": false, "filterable": true, ... "facetable": false, ... },
    { "name": "tags",             "type": "Collection(Edm.String)", "searchable": true,  "filterable": true, ... "facetable": true, ... }
  ],
  ...
}

```

Conformément à l'exemple de schéma, l'option `Facetable` est désactivée pour les champs de chaîne qui ne doivent pas être utilisés comme facettes, par exemple les valeurs d'ID. Le fait de désactiver les facettes inutiles permet de conserver une taille d'index réduite, ce qui améliore généralement les performances.

#### TIP

Nous vous recommandons d'inclure l'ensemble des attributs d'index pour chaque champ. Bien que l'option `Facetable` soit activée par défaut pour presque tous les champs, le fait de configurer volontairement chaque attribut peut vous aider à réfléchir aux implications de chaque décision au sein du schéma.

## Vérifier les données

La qualité de vos données a un effet direct sur la matérialisation attendue de la structure de la navigation à facettes. Elle affecte également la facilité de création des filtres afin de limiter le jeu de résultats.

Si vous souhaitez trier selon la marque ou le prix, chaque document doit contenir des valeurs `BrandName` et `ProductPrice` valides, cohérentes et productives en tant qu'option de filtre.

Voici quelques rappels des points à appliquer :

- Pour chaque champ que vous souhaitez utiliser comme facette, demandez-vous s'il contient des valeurs qui conviennent en tant que filtres dans une recherche autonome. Les valeurs doivent être courtes, descriptives et

suffisamment distinctives pour offrir un choix clair entre les différentes options.

- Fautes d'orthographe ou valeurs presque correspondantes. Si vous créez une facette Couleur et que les valeurs de champ incluent Orange et Ornage (faute d'orthographe), une facette basée sur le champ Couleur renverrait les deux options.
- Le texte à casse mixte peut également causer des dégâts dans la navigation à facettes, où orange et Orange s'afficheraient comme deux valeurs différentes.
- Les versions au singulier et au pluriel de la même valeur peuvent entraîner une facette distincte pour chacune.

Comme vous pouvez l'imaginer, la rigueur en termes de préparation des données est un aspect essentiel d'une navigation à facettes efficace.

## Créer l'interface utilisateur

Le fait de travailler à partir de la couche de présentation peut vous aider à découvrir des exigences que vous auriez pu manquer dans le cas contraire et de comprendre les capacités essentielles à l'expérience de recherche.

En termes de navigation à facettes, votre page Web ou d'application affiche la structure de navigation à facettes, détecte une entrée utilisateur sur la page et insère les éléments modifiés.

Pour les applications Web, la méthode AJAX est en général utilisée dans la couche de présentation, car elle vous permet d'actualiser les modifications incrémentielles. Vous pouvez également utiliser ASP.NET MVC ou toute autre plateforme de visualisation qui peut se connecter à un service Recherche cognitive Azure par le biais de HTTP. L'exemple d'application référencé dans cet article ([Démonstration du portail de recrutement de la Recherche cognitive Azure](#)) est une application ASP.NET MVC.

Dans l'exemple fourni, la navigation à facettes est intégrée dans la page de résultats. L'exemple suivant, extrait du fichier `index.cshtml` de l'exemple d'application, montre la structure HTML statique correspondant à l'affichage de la navigation à facettes sur la page des résultats de la recherche. La liste de facettes est créée ou recréée dynamiquement lorsque vous envoyez un terme de recherche, ou lorsque vous activez ou désactivez une facette.

```
<div class="widget sidebar-widget jobs-filter-widget">
    <h5 class="widget-title">Filter Results</h5>
    <p id="filterReset"></p>
    <div class="widget-content">

        <h6 id="businessTitleFacetTitle">Business Title</h6>
        <ul class="filter-list" id="business_title_facets">
        </ul>

        <h6>Location</h6>
        <ul class="filter-list" id="posting_type_facets">
        </ul>

        <h6>Posting Type</h6>
        <ul class="filter-list" id="posting_type_facets"></ul>

        <h6>Minimum Salary</h6>
        <ul class="filter-list" id="salary_range_facets">
        </ul>

    </div>
</div>
```

L'extrait de code suivant, tiré de la page `index.cshtml`, crée dynamiquement le code HTML permettant d'afficher la première facette, Fonction. Des fonctions similaires génèrent dynamiquement le code HTML pour les autres facettes. Chaque facette est associée à une étiquette et à un nombre indiquant combien d'éléments ont été trouvés pour ce résultat de facette.

```

function UpdateBusinessTitleFacets(data) {
    var facetResultsHTML = '';
    for (var i = 0; i < data.length; i++) {
        facetResultsHTML += '<li><a href="javascript:void(0)" onclick="ChooseBusinessTitleFacet(\'' +
data[i].Value + '\');">' + data[i].Value + ' (' + data[i].Count + ')</span></a></li>';
    }

    $("#business_title_facets").html(facetResultsHTML);
}

```

#### TIP

Lorsque vous concevez la page de résultats, pensez à ajouter un mécanisme de suppression des facettes. Si vous ajoutez des cases à cocher, il est facile de savoir comment effacer les filtres. Pour les autres dispositions, vous devrez peut-être utiliser un modèle de navigation ou une autre approche créative. Ainsi, dans l'exemple d'application du portail de recrutement, vous pouvez cliquer sur **[X]** situé en regard d'une facette pour effacer cette dernière.

## Création de la requête

Le code que vous écrivez pour la création de requêtes doit spécifier toutes les parties d'une requête valide, y compris les expressions de recherche, les facettes, les filtres, les profils de score ; tout ce qui sert à formuler une requête. Dans cette section, nous allons explorer l'emplacement où les facettes s'intègrent dans une requête, ainsi que la façon dont les filtres sont utilisés avec des facettes pour fournir un jeu de résultats réduit.

Notez que les facettes font partie intégrante de cet exemple d'application. L'expérience de recherche dans la démonstration du portail de recrutement est conçue autour de filtres et de la navigation à facettes. La position de la navigation à facettes sur la page démontre son importance.

Pour commencer, prenons un exemple. L'extrait suivant, extrait du fichier **JobsSearch.cs**, construit une requête qui crée une navigation à facettes basée sur la Fonction, le Lieu, le Type de publication et le Salaire minimal.

```

SearchParameters sp = new SearchParameters()
{
    ...
    // Add facets
    Facets = new List<String>() { "business_title", "posting_type", "level", "salary_range_from,interval:50000"
},
};

```

Un paramètre de requête à facettes est défini sur un champ et, selon le type de données, peut être davantage paramétré par une liste délimitée par des virgules qui inclut **count:<integer>**, **sort:<>**, **interval:<integer>** et **values:<list>**. Une liste de valeurs est prise en charge pour les données numériques lors de la définition de plages. Consultez [Rechercher des documents \(API de Recherche cognitive Azure\)](#) pour obtenir des détails sur l'utilisation.

En plus des facettes, la requête formulée par votre application doit également créer des filtres pour limiter le jeu de documents candidats basés sur une sélection de valeur de facette. Pour un magasin de vélos, la navigation à facettes fournit des indications aux questions du type *Quels sont les couleurs, fabricants et types de vélos disponibles ?*. Le filtrage permet de répondre à des questions du type *Quels sont les vélos de type VTT et de couleur rouge dans cette gamme de prix ?*. Lorsque vous cliquez sur « Rouge » pour indiquer que seuls les produits de couleur rouge doivent s'afficher, la requête suivante envoyée par l'application inclut **\$filter=Color eq 'Red'**.

L'extrait de code suivant tiré de la page **JobsSearch.cs** ajoute la Fonction sélectionnée au filtre si vous sélectionnez une valeur de la facette Fonction.

```
if (businessTitleFacet != "")  
    filter = "business_title eq '" + businessTitleFacet + "'";
```

## Conseils et meilleures pratiques

### Conseils d'indexation

#### Améliorer l'efficacité des index si vous n'utilisez pas de zone de recherche

Si votre application utilise exclusivement la navigation à facettes (autrement dit, aucune zone de recherche), vous pouvez marquer le champ en tant que `searchable=false`, `facettable=true` pour produire un index plus compact. En outre, l'indexation se produit uniquement sur les valeurs de facettes entières, sans césure de mots ou indexation des composants d'une valeur à plusieurs mots.

#### Spécifier les champs qui peuvent servir de facettes

N'oubliez pas que le schéma de l'index détermine quels champs sont disponibles pour être utilisés comme facettes. En supposant qu'un champ puisse être utilisé comme facette, la requête spécifie quels champs utiliser comme facettes. Le champ que vous utilisez comme facette fournit les valeurs qui apparaissent sous l'étiquette.

Les valeurs qui s'affichent sous chaque étiquette sont récupérées à partir de l'index. Par exemple, si le champ de facette est *Couleur*, les valeurs disponibles pour le filtrage supplémentaire sont les valeurs de ce champ : Rouge, Noir, etc.

Pour les valeurs de type Numérique et DateHeure uniquement, vous pouvez définir explicitement des valeurs sur le champ de facette (par exemple, `facet=Rating,values:1|2|3|4|5`). Une liste de valeurs est autorisée pour ces types de champs afin de simplifier la séparation des résultats de la facette en plages contiguës (plages basées sur des valeurs numériques ou des périodes de temps).

#### Par défaut, vous ne pouvez avoir qu'un seul niveau de navigation par facettes

Comme mentionné, il n'existe aucune prise en charge directe de l'imbrication des facettes dans une hiérarchie. Par défaut, la navigation par facettes dans la Recherche cognitive Azure ne prend en charge qu'un seul niveau de filtres. Toutefois, des solutions de contournement existent. Vous pouvez encoder une structure hiérarchique de facette dans une `Collection(Edm.String)` avec un point d'entrée par hiérarchie. L'implémentation de cette solution de contournement n'est pas abordée dans cet article.

### Conseils d'interrogation

#### Validation des champs

Si vous générez dynamiquement la liste de facettes en fonction d'une entrée utilisateur non fiable, vérifiez la validité des noms des champs à facettes. Vous pouvez également contourner les noms lors de la génération d'URL à l'aide de `Uri.EscapeDataString()` dans .NET, ou son équivalent dans la plateforme de votre choix.

### Conseils de filtrage

#### Augmenter la précision de la recherche avec des filtres

Utilisation de filtres. Si vous utilisez uniquement les expressions de recherche, la recherche de radical peut entraîner le renvoi d'un document qui ne contient pas la valeur de facette précise dans aucun de ses champs.

#### Augmenter les performances de recherche avec des filtres

Les filtres réduisent le jeu de documents candidats pour la recherche et les excluent du classement. Pour les grands jeux de documents, l'utilisation d'une exploration à facettes sélective offre souvent de meilleures performances.

#### Filtrer uniquement les champs à facettes

Dans l'exploration à facettes, vous voulez en général inclure uniquement les documents contenant la valeur de facette dans un champ spécifique (facette), et pas n'importe où dans tous les champs de recherche. L'ajout d'un filtre renforce le champ cible en indiquant au service de rechercher uniquement dans le champ à facette pour trouver une valeur correspondante.

## Tronquer les résultats de facettes avec d'autres filtres

Les résultats de la facette sont des documents trouvés dans les résultats de la recherche qui correspondent à un terme de la facette. Dans l'exemple suivant, dans les résultats de la recherche pour *cloud computing*, 254 éléments ont également la *spécification interne* comme type de contenu. Les éléments ne sont pas nécessairement mutuellement exclusifs. Si un élément répond aux critères des deux filtres, il est compté dans chacun d'eux. La duplication est possible lors de l'utilisation des facettes sur les champs `Collection(Edm.String)`, souvent utilisés pour implémenter le balisage de document.

```
Search term: "cloud computing"
Content type
Internal specification (254)
Video (10)
```

En général, si vous trouvez que les résultats de la facette sont toujours trop volumineux, nous vous recommandons d'ajouter des filtres afin d'apporter aux utilisateurs de votre application plus d'options pour affiner la recherche.

## Conseils sur le décompte des résultats

### Limiter le nombre d'éléments dans la navigation par facettes

Pour chaque champ à facettes dans l'arborescence de navigation, il existe une limite par défaut de 10 valeurs. Cette valeur par défaut est judicieuse pour les structures de navigation, car elle permet de conserver une taille gérable pour la liste des valeurs. Vous pouvez remplacer la valeur par défaut en affectant une valeur à compter.

- `&facet=city,count:5` spécifie que seules les cinq premières villes trouvées dans les résultats en tête du classement sont renvoyées en tant que résultat de la facette. Imaginez un exemple de requête affichant le terme de recherche « aéroport » et 32 correspondances. Si la requête spécifie `&facet=city,count:5`, seules les cinq premières villes uniques avec le plus de documents dans les résultats de la recherche sont incluses dans les résultats de la facette.

Notez la différence entre les résultats de la recherche et les résultats de la facette. Les résultats de la recherche sont tous les documents qui correspondent à la requête. Les résultats de la facette sont les correspondances pour chaque valeur de facette. Dans l'exemple, les résultats de la recherche incluent des noms de villes qui ne se trouvent pas dans la liste de classification de la facette (5, dans notre exemple). Les résultats filtrés par le biais de la navigation à facettes deviennent visibles lorsque vous effacez les facettes ou choisissez d'autres facettes en plus de Ville.

### NOTE

Traiter de `count` lorsqu'il existe plus d'un type peut prêter à confusion. Le tableau suivant offre un bref résumé de l'utilisation du terme dans l'API de Recherche cognitive Azure, un exemple de code et la documentation.

- `@colorFacet.count`  
Dans le code de présentation, un paramètre de décompte doit s'afficher sur la facette. Il est utilisé pour afficher le nombre de résultats de la facette. Dans les résultats de la facette, le décompte indique le nombre de documents qui correspondent au terme ou à la plage de la facette.
- `&facet=City,count:12`  
Dans une requête de facette, vous pouvez définir le décompte sur une valeur. La valeur par défaut est 10, mais

vous pouvez définir une valeur supérieure ou inférieure. Le paramètre `count:12` renvoie les 12 premières correspondances dans les résultats de la facette selon le décompte de documents.

- "`@odata.count`"

Dans la réponse de la requête, cette valeur indique le nombre d'éléments correspondants dans les résultats de la recherche. En moyenne, il est supérieur à la somme de tous les résultats de la facette combinés, en raison de la présence d'éléments qui correspondent au terme de la recherche, mais sans correspondance avec la valeur de la facette.

## Obtenir les décomptes dans les résultats de facettes

Lorsque vous ajoutez un filtre à une requête à facettes, il se peut que vous souhaitiez conserver l'instruction de facette (par exemple, `facet=Rating&$filter=Rating ge 4`). Techniquement, le paramètre `facette=Évaluation` n'est pas nécessaire, mais le fait de le conserver renvoie les décomptes des valeurs de facettes pour les évaluations de 4 et supérieures. Par exemple, si vous cliquez sur « 4 » et si la requête inclut un filtre pour une valeur supérieure ou égale à « 4 », des décomptes sont renvoyés pour chaque évaluation égale ou supérieure à 4.

## Vérifier que les nombres de facettes sont exacts

Dans certaines circonstances, il est possible que les décomptes de facettes ne correspondent pas aux jeux de résultats (consultez [Navigation par facettes dans la Recherche cognitive Azure \(page de questions Microsoft Q&A\)](#)).

Les décomptes de facettes peuvent être erronés en raison de l'architecture de partitionnement. Chaque index de recherche a plusieurs partitions et chacune d'elles indique les N premières facettes par décompte de document, qui est ensuite combiné en un résultat unique. Si certaines partitions ont beaucoup de valeurs correspondantes, tandis que d'autres en ont moins, il est possible que certaines valeurs de facettes soient manquantes ou sous-comptabilisées dans les résultats.

Ce comportement peut changer à tout moment mais si vous rencontrez ce problème aujourd'hui, vous pouvez le contourner en gonflant artificiellement le décompte :`<number>` sur un nombre élevé pour appliquer la déclaration complète à partir de chaque partition. Si la valeur de décompte : est supérieure ou égale au nombre de valeurs uniques dans le champ, vous êtes sûr d'obtenir des résultats précis. Toutefois, lorsque les décomptes de documents sont élevés, les performances baissent, alors utilisez cette option judicieusement.

## Conseils sur l'interface utilisateur

### Ajouter des étiquettes pour chaque champ dans la navigation à facettes

Les étiquettes sont généralement définies dans le code HTML ou le formulaire (`index.cshtml` dans l'exemple d'application). Il n'existe aucune API dans la Recherche cognitive Azure pour les étiquettes de navigation par facettes ou tout autre type de métadonnées.

## Filtrer sur une plage de valeurs

L'utilisation de facettes sur des plages de valeurs est une condition d'application de recherche courante. Les plages sont prises en charge pour les données numériques et les valeurs DateHeure. Vous pouvez en savoir plus sur chaque approche en consultant [Rechercher des documents \(API de Recherche cognitive Azure\)](#).

La Recherche cognitive Azure simplifie la création de plage en fournissant deux approches pour calculer une plage. Pour les deux approches, la Recherche cognitive Azure crée les plages appropriées avec les entrées que vous avez fournies. Par exemple, si vous spécifiez des valeurs de plage de 10|20|30, Recherche Azure crée automatiquement les plages 0-10, 10-20, 20-30. Votre application peut éventuellement supprimer les intervalles vides.

### Approche 1 : Utiliser le paramètre d'intervalle

Pour définir les facettes de prix par incrément de 10 \$, vous devez spécifier : `&facet=price,interval:10`

## Approche 2 : Utiliser une liste de valeurs

Pour les données numériques, vous pouvez utiliser une liste de valeurs. Prenez en compte la plage de facette pour un champ `listPrice`, indiquée comme suit :

Prices:

- 0 - 10 (13)
- 10 - 25 (11)
- 25 - 100 (66)
- 100 - 500 (68)
- 500 - 1000 (50)
- 1000 - 2500 (73)
- 2500 - more (13)

Pour spécifier une plage de facette comme celle de la capture d'écran précédente, utilisez une liste de valeurs :

```
facet=listPrice,values:10|25|100|500|1000|2500
```

Chaque plage est créée avec 0 comme point de départ, une valeur de la liste comme point de terminaison, puis la plage précédente en moins pour créer des intervalles discrets. La Recherche cognitive Azure effectue cette opération dans le cadre de la navigation par facettes. Vous n'avez pas à écrire du code pour structurer chaque intervalle.

## Créer un filtre pour une plage

Pour filtrer les documents en fonction d'une plage que vous avez sélectionnée, vous pouvez utiliser les opérateurs de filtre `"ge"` et `"lt"` dans une expression en deux parties qui définit les points de terminaison de la plage. Par exemple, si vous choisissez la plage 10-25 pour un champ `listPrice`, le filtre sera

`$filter=listPrice ge 10 and listPrice lt 25`. Dans l'exemple de code, l'expression de filtre utilise les paramètres `priceFrom` et `priceTo` pour définir les points de terminaison.

```
private string BuildFilter(string color, string category, double? priceFrom, double? priceTo)
{
    string filter = "&$filter=discontinuedDate eq null";

    if (!string.IsNullOrWhiteSpace(color))
    {
        filter += " and color eq '" + Escape0DataString(color) + "'";
    }

    if (!string.IsNullOrWhiteSpace(category))
    {
        filter += " and categoryName eq '" + Escape0DataString(category) + "'";
    }

    if (priceFrom.HasValue)
    {
        filter += " and listPrice ge " + priceFrom.Value.ToString(CultureInfo.InvariantCulture);
    }

    if (priceTo.HasValue && priceTo > 0)
    {
        filter += " and listPrice le " + priceTo.Value.ToString(CultureInfo.InvariantCulture);
    }
}

return filter;
```

## Filtrer en fonction de la distance

Il est courant de voir des filtres qui vous aident à choisir un magasin, un restaurant ou une destination en fonction de sa proximité à votre emplacement actuel. Ce type de filtre peut ressembler à la navigation à facettes, mais c'est tout simplement un filtre. Nous le mentionnons ici pour ceux d'entre vous qui recherchent spécifiquement des conseils d'implémentation pour ce problème de conception particulier.

Il existe deux fonctions géospatiales dans la Recherche cognitive Azure, `geo.distance` et `geo.intersects`.

- La fonction `geo.distance` renvoie la distance en kilomètres entre deux points. L'un est un champ et l'autre est une constante considérée comme une partie du filtre.
- La fonction `geo.intersects` renvoie true si un point donné se trouve dans un polygone donné. Le point est un champ et le polygone est spécifié sous forme de liste constante de coordonnées en tant que partie du filtre.

Vous trouverez des exemples de filtres dans [Syntaxe d'expression OData \(Recherche cognitive Azure\)](#).

## Essayer la démonstration

La démonstration du portail de recrutement de la Recherche cognitive Azure contient les exemples référencés dans cet article.

- Consultez et testez la [démonstration en ligne du portail de recrutement de la Recherche cognitive Azure](#).
- Téléchargez le code à partir du [référentiel Azure-Samples sur GitHub](#).

Lorsque vous utilisez les résultats de la recherche, observez les modifications de construction de la requête dans l'URL. Cette application ajoute des facettes à l'URI lors de la sélection de chaque.

- Pour utiliser la fonctionnalité de mappage de l'application de démonstration, obtenez une clé Bing Maps auprès du [Centre de développement Bing Maps](#). Collez-la sur la clé existante dans la page `index.cshtml`. Le paramètre `BingApiKey` du fichier `Web.config` n'est pas utilisé.
- Exécutez l'application. Suivez la visite guidée facultative ou fermez la boîte de dialogue.
- Saisissez un terme de recherche, comme « analyste », puis cliquez sur l'icône de recherche. La requête s'exécute rapidement.

Une structure de navigation à facettes est également renvoyée avec les résultats de la recherche. Dans la page de résultats de la recherche, la structure de la navigation à facettes inclut des décomptes pour chaque résultat de facette. Aucune facette n'est sélectionnée : tous les résultats correspondants sont donc renvoyés.

- Cliquez sur une Fonction, un Lieu ou un Salaire minimal. Les facettes sont égales à « null » pour la recherche initiale, mais lorsqu'elles prennent des valeurs, les éléments qui ne correspondent plus sont supprimés des résultats de la recherche.

**Azure Search**  
JOB PORTAL DEMO

AVAILABLE JOBS (6 jobs)

Home / Jobs / About Azure Search

**SEARCH**

analyst

Any distance from

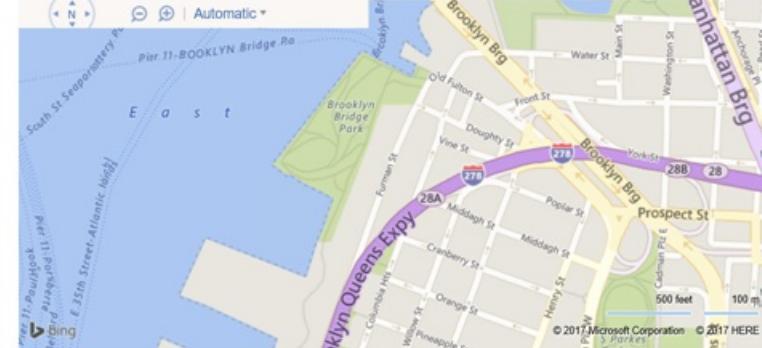
**FILTER RESULTS**

Current Filters:  
 Policy Analyst [X]  
 \$0 - \$49,999 [X]

**BUSINESS TITLE**

> Policy Analyst (6)

**LOCATION**



6 AVAILABLE JOBS

5. Pour effacer la requête à facettes afin de tester des comportements de requête différents, cliquez sur  en regard des facettes sélectionnées dans le but d'effacer ces dernières.

## En savoir plus

Regardez la [vidéo de présentation approfondie de la Recherche cognitive Azure](#). À 45:25 sur la vidéo, vous trouverez une démonstration sur la façon d'implémenter des facettes.

Pour plus d'informations sur les principes de conception pour la navigation à facettes, nous vous recommandons les liens suivants :

- [Modèles de conception : Navigation à facettes](#)
- [Problèmes de serveur frontal lors de l'implémentation de la recherche à facettes - Partie 1](#)

# Résolution de problèmes liés aux filtres de collection OData dans la Recherche cognitive Azure

04/10/2020 • 22 minutes to read • [Edit Online](#)

Pour appliquer un [filtre](#) sur des champs de collection dans la Recherche cognitive Azure, vous pouvez utiliser les **opérateurs** `any` et `all` avec des **expressions lambda**. Une expression lambda est un sous-filtre qui est appliquée à chaque élément d'une collection.

Toutes les fonctionnalités des expressions de filtre ne sont pas disponibles dans une expression lambda. Les fonctionnalités diffèrent selon le type de données du champ de la collection que vous souhaitez filtrer. Cela peut entraîner une erreur si vous essayez d'utiliser une fonctionnalité dans une expression lambda qui n'est pas prise en charge dans ce contexte. Si vous rencontrez de telles erreurs lors de la tentative d'écriture d'un filtre complexe sur des champs de collection, cet article vous aidera à résoudre le problème.

## Erreurs de filtre de collection courantes

Le tableau suivant répertorie les erreurs que vous pouvez rencontrer lorsque vous tentez d'exécuter un filtre de collection. Ces erreurs se produisent lorsque vous utilisez une fonctionnalité d'expressions de filtre qui n'est pas prise en charge dans une expression lambda. Chaque erreur donne des conseils sur la façon dont vous pouvez réécrire votre filtre afin d'éviter l'erreur. Le tableau contient également un lien vers la section appropriée de cet article, qui fournit plus d'informations sur la façon d'éviter cette erreur.

MESSAGE D'ERREUR	SITUATION	POUR PLUS D'INFORMATIONS, CONSULTEZ LA RUBRIQUE
La fonction « ismatch » n'a aucun paramètre lié à la variable de portée « s ». Seules les références de champ liées sont prises en charge dans les expressions lambda (« any » ou « all »). Veuillez modifier votre filtre afin que la fonction « ismatch » soit en dehors de l'expression lambda et réessayez.	Utilisation de <code>search.ismatch</code> ou <code>search.ismatchscoring</code> dans une expression lambda	<a href="#">Règles de filtrage des collections complexes</a>
Expression lambda non valide. Un test d'égalité ou d'inégalité où le contraire était attendu a été trouvé dans une expression lambda qui itère sur un champ de type collection (Edm.String). Pour « any », utilisez des expressions de forme « x eq y » ou « search.in(...) ». Pour « all », utilisez des expressions de forme « x ne y », « not (x eq y) » ou « not search.in(...) ».	Filtrage sur un champ de type <code>Collection(Edm.String)</code>	<a href="#">Règles de filtrage des collections de chaînes</a>

MESSAGE D'ERREUR	SITUATION	POUR PLUS D'INFORMATIONS, CONSULTEZ LA RUBRIQUE
<p>Expression lambda non valide. Une forme non prise en charge d'une expression booléenne complexe a été trouvé. Pour « any », utilisez des expressions « ORs de ANDs », également appelées forme normale disjonctive. Par exemple : « (a et b) ou (c et d) », où a, b, c et d sont des sous-expressions de comparaison ou d'égalité. Pour « all », utilisez des expressions « ANDs de ORs », également appelées forme normale conjonctive. Par exemple : « (a ou b) ou (c ou d) », où a, b, c et d sont des sous-expressions de comparaison ou d'inégalité. Exemples d'expressions de comparaison : « x gt 5 », « x le 2 ». Exemple d'expression d'égalité : « x eq 5 ». Exemple d'expression d'inégalité : « x ne 5 ».</p>	<p>Filtrage sur des champs de type  <code>Collection(Edm.DateTimeOffset)</code> ,  <code>Collection(Edm.Double)</code> ,  <code>Collection(Edm.Int32)</code> ou  <code>Collection(Edm.Int64)</code></p>	<a href="#">Règles de filtrage des collections comparables</a>
<p>Expression lambda non valide. Une utilisation non prise en charge de <code>geo.distance()</code> ou de <code>geo.intersects()</code> a été trouvée dans une expression lambda qui itère sur un champ de type <code>collection(Edm.GeographyPoint)</code>. Pour « any », vérifiez que vous comparez <code>geo.distance()</code> à l'aide des opérateurs « lt » ou « le » et qu'aucune utilisation de <code>geo.intersects()</code> n'est négative. Pour « all », vérifiez que vous comparez <code>geo.distance()</code> à l'aide des opérateurs « gt » ou « ge » et que toute utilisation de <code>geo.intersects()</code> est négative.</p>	<p>Filtrage sur un champ de type  <code>Collection(Edm.GeographyPoint)</code></p>	<a href="#">Règles de filtrage des collections GeographyPoint</a>
<p>Expression lambda non valide. Les expressions booléennes complexes ne sont pas prises en charge dans les expressions lambda qui itèrent sur des champs de type <code>Collection(Edm.GeographyPoint)</code>. Pour « any », reliez les sous-expressions avec « ou » ; « et » n'est pas pris en charge. Pour « all », reliez les sous-expressions avec « et » ; « ou » n'est pas pris en charge.</p>	<p>Filtrage sur des champs de type  <code>Collection(Edm.String)</code> ou  <code>Collection(Edm.GeographyPoint)</code></p>	<a href="#">Règles de filtrage des collections de chaînes</a> <a href="#">Règles de filtrage des collections GeographyPoint</a>
<p>Expression lambda non valide. Trouver un opérateur de comparaison (« lt », « le », « gt » ou « ge »). Seuls des opérateurs d'égalité sont autorisés dans les expressions lambda qui itèrent sur les champs de type <code>collection(Edm.String)</code>. Pour « any », utilisez des expressions de forme « x eq y ». Pour « all », utilisez des expressions de forme « x ne y » ou « not (x eq y) ».</p>	<p>Filtrage sur un champ de type  <code>Collection(Edm.String)</code></p>	<a href="#">Règles de filtrage des collections de chaînes</a>

# Comment écrire des filtres de collection valides

Les règles d'écriture de filtres de collection valides sont différentes pour chaque type de données. Les sections suivantes décrivent les règles en donnant des exemples de fonctionnalités de filtres qui sont prises en charge ou ne le sont pas :

- [Règles de filtrage des collections de chaînes](#)
- [Règles de filtrage des collections booléennes](#)
- [Règles de filtrage des collections GeographyPoint](#)
- [Règles de filtrage des collections comparables](#)
- [Règles de filtrage des collections complexes](#)

## Règles de filtrage des collections de chaînes

Dans les expressions lambda pour les collections de chaînes, les seuls opérateurs de comparaison pouvant être utilisés sont `eq` et `ne`.

### NOTE

La Recherche cognitive Azure ne prend pas en charge les opérateurs `lt` / `le` / `gt` / `ge` pour les chaînes, qu'ils soient inclus ou non dans une expression lambda.

Le corps d'un `any` peut uniquement tester l'égalité, tandis que le corps d'un `all` peut uniquement tester l'inégalité.

Il est également possible de combiner plusieurs expressions via `or` dans le corps d'un `any` et via `and` dans le corps d'un `all`. Étant donné que la fonction `search.in` est équivalente à la combinaison de vérifications d'égalité avec `or`, elle est également autorisée dans le corps d'un `any`. À l'inverse, `not search.in` est autorisé dans le corps d'un `all`.

Par exemple, ces expressions sont autorisées :

- `tags/any(t: t eq 'books')`
- `tags/any(t: search.in(t, 'books, games, toys'))`
- `tags/all(t: t ne 'books')`
- `tags/all(t: not (t eq 'books'))`
- `tags/all(t: not search.in(t, 'books, games, toys'))`
- `tags/any(t: t eq 'books' or t eq 'games')`
- `tags/all(t: t ne 'books' and not (t eq 'games'))`

alors que ces expressions ne le sont pas :

- `tags/any(t: t ne 'books')`
- `tags/any(t: not search.in(t, 'books, games, toys'))`
- `tags/all(t: t eq 'books')`
- `tags/all(t: search.in(t, 'books, games, toys'))`
- `tags/any(t: t eq 'books' and t ne 'games')`
- `tags/all(t: t ne 'books' or not (t eq 'games'))`

## Règles de filtrage des collections booléennes

Le type `Edm.Boolean` prend uniquement en charge les opérateurs `eq` et `ne`. Par conséquent, permettre la

combinaison de clauses qui vérifient la même variable de portée avec `and` / `or` n'a pas de sens, car cela entraîne toujours des tautologies ou des contradictions.

Voici quelques exemples de filtres sur les collections booléennes qui sont autorisés :

- `flags/any(f: f)`
- `flags/all(f: f)`
- `flags/any(f: f eq true)`
- `flags/any(f: f ne true)`
- `flags/all(f: not f)`
- `flags/all(f: not (f eq true))`

Contrairement aux collections de chaînes, les collections booléennes n'ont aucune limite sur laquelle l'opérateur peut être utilisé dans le type d'expression lambda. Les deux `eq` et `ne` peuvent être utilisés dans le corps de `any` ou de `all`.

Les expressions comme celles qui suivent ne sont pas autorisées pour les collections booléennes :

- `flags/any(f: f or not f)`
- `flags/any(f: f or f)`
- `flags/all(f: f and not f)`
- `flags/all(f: f and f eq true)`

## Règles de filtrage des collections GeographyPoint

Les valeurs de type `Edm.GeographyPoint` dans une collection ne peuvent pas être comparées directement entre les unes aux autres. Au lieu de cela, elles doivent être utilisées en tant que paramètres des fonctions `geo.distance` et `geo.intersects`. La fonction `geo.distance` doit à son tour être comparée à une valeur de distance à l'aide d'un des opérateurs de comparaison `lt`, `le`, `gt` ou `ge`. Ces règles s'appliquent également aux champs `Edm.GeographyPoint` autres que collection.

Comme les collections de chaînes, les collections `Edm.GeographyPoint` ont des règles relatives à la façon dont les fonctions géospatiales peuvent être utilisées et combinées dans les différents types d'expressions lambda :

- Les opérateurs de comparaison que vous pouvez utiliser avec la fonction `geo.distance` varient selon le type d'expression lambda. Pour `any`, vous pouvez utiliser uniquement `lt` ou `le`. Pour `all`, vous pouvez utiliser uniquement `gt` ou `ge`. Vous pouvez inverser les expressions qui impliquent `geo.distance`, mais vous devrez modifier l'opérateur de comparaison (`geo.distance(...)` `lt` `x` devient `not (geo.distance(...)` `ge` `x`) et `geo.distance(...)` `le` `x` devient `not (geo.distance(...)` `gt` `x`).
- Dans le corps d'un `all`, la fonction `geo.intersects` doit être négative. Dans le corps d'un `any`, la fonction `geo.intersects` doit être négative.
- Dans le corps d'un `any`, des expressions géospatiales peuvent être combinées à l'aide de `or`. Dans le corps d'un `all`, ces expressions peuvent être combinées à l'aide de `and`.

Les limitations ci-dessus existent pour des raisons similaires à la limitation d'égalités/d'inégalités sur les collections de chaînes. Consultez [Présentation des filtres de collection OData dans la Recherche cognitive Azure](#) pour en savoir plus sur ces raisons.

Voici quelques exemples de filtres autorisés sur `Edm.GeographyPoint` :

- `locations/any(l: geo.distance(l, geography'POINT(-122 49)') lt 10)`  
`locations/any(l: not (geo.distance(l, geography'POINT(-122 49)') ge 10) or geo.intersects(l, geography'POLYGON((-122.031577 47.578581, -122.031577 47.678581, -122.131577 47.678581, -122.031577 47.578581)))`

```
locations/all(l: geo.distance(l, geography'POINT(-122 49)') ge 10 and not geo.intersects(l, geography'POLYGON((-122.031577 47.578581, -122.031577 47.678581, -122.131577 47.678581, -122.031577 47.578581)))')
```

Les expressions comme celles qui suivent ne sont pas autorisées pour les collections `Edm.GeographyPoint` :

- `locations/any(l: l eq geography'POINT(-122 49)')`
- `locations/any(l: not geo.intersects(l, geography'POLYGON((-122.031577 47.578581, -122.031577 47.678581, -122.131577 47.678581, -122.031577 47.578581)))')`
- `locations/all(l: geo.intersects(l, geography'POLYGON((-122.031577 47.578581, -122.031577 47.678581, -122.131577 47.678581, -122.031577 47.578581)))')`
- `locations/any(l: geo.distance(l, geography'POINT(-122 49)') gt 10)`
- `locations/all(l: geo.distance(l, geography'POINT(-122 49)') lt 10)`
- `locations/any(l: geo.distance(l, geography'POINT(-122 49)') lt 10 and geo.intersects(l, geography'POLYGON((-122.031577 47.578581, -122.031577 47.678581, -122.131577 47.678581, -122.031577 47.578581)))')`
- `locations/all(l: geo.distance(l, geography'POINT(-122 49)') le 10 or not geo.intersects(l, geography'POLYGON((-122.031577 47.578581, -122.031577 47.678581, -122.131577 47.678581, -122.031577 47.578581)))')`
- `locations/any(l: geo.distance(l, geography'POINT(-122 49)') ge 10 and geo.intersects(l, geography'POLYGON((-122.031577 47.578581, -122.031577 47.678581, -122.131577 47.678581, -122.031577 47.578581)))')`

## Règles de filtrage des collections comparables

Cette section s'applique à tous les types de données suivants :

- `Collection(Edm.DateTimeOffset)`
- `Collection(Edm.Double)`
- `Collection(Edm.Int32)`
- `Collection(Edm.Int64)`

Les types tels que `Edm.Int32` et `Edm.DateTimeOffset` prennent en charge les six opérateurs de comparaison : `eq`, `ne`, `lt`, `le`, `gt` et `ge`. Les expressions lambda sur des collections de ces types peuvent contenir des expressions simples utilisant l'un de ces opérateurs. Cela s'applique à `any` et à `all`. Les filtres suivants sont par exemple autorisés :

- `ratings/any(r: r ne 5)`
- `dates/any(d: d gt 2017-08-24T00:00:00Z)`
- `not margins/all(m: m eq 3.5)`

Toutefois, il existe des limitations sur la façon dont ces expressions de comparaison peuvent être combinées dans des expressions plus complexes à l'intérieur d'une expression lambda :

- Règles pour `any` :
  - Des expressions d'inégalité simples ne peuvent pas être combinées de façon utile avec d'autres expressions. Par exemple, cette expression est autorisée :
    - `ratings/any(r: r ne 5)`mais cette expression ne l'est pas :
    - `ratings/any(r: r ne 5 and r gt 2)`et, bien que cette expression soit autorisée, elle n'est pas utile, car les conditions se chevauchent :
    - `ratings/any(r: r ne 5 or r gt 7)`
  - Des expressions de comparaison simples impliquant `eq`, `lt`, `le`, `gt` ou `ge` peuvent être combinées avec `and` / `or`. Par exemple :
    - `ratings/any(r: r gt 2 and r le 5)`
    - `ratings/any(r: r le 5 or r gt 7)`

- Les expressions de comparaison combinées avec `and` (conjonctions) peuvent également l'être à l'aide de `or`. Dans une logique booléenne, cette forme est appelée « [Forme normale disjonctive](#) » (FND). Par exemple :

- `ratings/any(r: (r gt 2 and r le 5) or (r gt 7 and r lt 10))`

- Règles pour `all` :

- Des expressions d'égalité simples ne peuvent pas être combinées de façon utile avec d'autres expressions. Par exemple, cette expression est autorisée :

- `ratings/all(r: r eq 5)`

mais cette expression ne l'est pas :

- `ratings/all(r: r eq 5 or r le 2)`

et, bien que cette expression soit autorisée, elle n'est pas utile, car les conditions se chevauchent :

- `ratings/all(r: r eq 5 and r le 7)`

- Des expressions de comparaison simples impliquant `ne`, `lt`, `le`, `gt` ou `ge` peuvent être combinées avec `and` / `or`. Par exemple :

- `ratings/all(r: r gt 2 and r le 5)`

- `ratings/all(r: r le 5 or r gt 7)`

- Les expressions de comparaison combinées avec `or` (disjonctions) peuvent également l'être à l'aide de `and`. Dans une logique booléenne, cette forme est appelée « [Forme normale conjonctive](#) » (FNC). Par exemple :

- `ratings/all(r: (r le 2 or gt 5) and (r lt 7 or r ge 10))`

## Règles de filtrage des collections complexes

Les expressions lambda sur des collections complexes prennent en charge une syntaxe beaucoup plus flexible que les expressions lambda sur des collections de types primitifs. Dans une telle construction lambda, vous pouvez utiliser n'importe quelle construction de filtre que vous pouvez également utiliser ailleurs, à deux exceptions près seulement.

Tout d'abord, les fonctions `search.ismatch` et `search.ismatchscoring` ne sont pas prises en charge dans les expressions lambda. Pour plus d'informations, consultez [Présentation des filtres de collection OData dans la Recherche cognitive Azure](#).

Ensuite, il n'est pas permis de faire référence à des champs qui ne sont pas *liés* à la variable de portée (appelée *variables libres*). Voyons par exemple les deux expressions de filtres OData équivalentes suivantes :

1. `stores/any(s: s/amenities/any(a: a eq 'parking')) and details/margin gt 0.5`
2. `stores/any(s: s/amenities/any(a: a eq 'parking' and details/margin gt 0.5))`

La première expression sera autorisée, alors que la seconde forme sera rejetée, car `details/margin` n'est pas lié à la variable de portée `s`.

Cette règle s'applique également aux expressions qui ont des variables liées dans une portée externe. Ces variables sont libres par rapport à la portée dans laquelle elles apparaissent. Par exemple, la première expression est autorisée, tandis que la seconde expression, équivalente, ne l'est pas, car `s/name` est libre en par rapport à la portée de la variable de portée `a` :

1. `stores/any(s: s/amenities/any(a: a eq 'parking') and s/name ne 'Flagship')`
2. `stores/any(s: s/amenities/any(a: a eq 'parking' and s/name ne 'Flagship'))`

Cette limitation ne devrait pas être un problème dans la pratique, car il est toujours possible de construire des filtres de telle façon que les expressions lambda contiennent uniquement des variables liées.

## Aide-mémoire pour les règles de filtre de collection

Le tableau suivant récapitule les règles de construction de filtres valides pour chaque type de données de collection.

TYPE DE DONNÉES	FONCTIONNALITÉS AUTORISÉES DANS LES EXPRESSIONS LAMBDA AVEC ANY	FONCTIONNALITÉS AUTORISÉES DANS LES EXPRESSIONS LAMBDA AVEC ALL
Collection(Edm.ComplexType)	Tout sauf <code>search.ismatch</code> et <code>search.ismatchscoring</code>	Identique
Collection(Edm.String)	Comparaisons avec <code>eq</code> ou <code>search.in</code>  Combinaison de sous-expressions avec <code>or</code>	Comparaisons avec <code>ne</code> ou <code>not search.in()</code>  Combinaison de sous-expressions avec <code>and</code>
Collection(Edm.Boolean)	Comparaisons avec <code>eq</code> ou <code>ne</code>	Identique
Collection(Edm.GeographyPoint)	Utilisation de <code>geo.distance</code> avec <code>lt</code> ou <code>le</code>  <code>geo.intersects</code>  Combinaison de sous-expressions avec <code>or</code>	Utilisation de <code>geo.distance</code> avec <code>gt</code> ou <code>ge</code>  <code>not geo.intersects(...)</code>  Combinaison de sous-expressions avec <code>and</code>
Collection(Edm.DateTimeOffset), Collection(Edm.Double), Collection(Edm.Int32), Collection(Edm.Int64)	Comparaisons avec <code>eq</code> , <code>ne</code> , <code>lt</code> , <code>gt</code> , <code>le</code> ou <code>ge</code>  Combinaison de comparaisons avec d'autres sous-expressions à l'aide de <code>or</code>  Combinaison de comparaisons, sauf <code>ne</code> , avec d'autres sous-expressions à l'aide de <code>and</code>  Expressions utilisant des combinaisons de <code>and</code> et <code>or</code> en <b>forme normale disjonctive (FND)</b>	Comparaisons avec <code>eq</code> , <code>ne</code> , <code>lt</code> , <code>gt</code> , <code>le</code> ou <code>ge</code>  Combinaison de comparaisons avec d'autres sous-expressions à l'aide de <code>and</code>  Combinaison de comparaisons, sauf <code>eq</code> , avec d'autres sous-expressions à l'aide de <code>or</code>  Expressions utilisant des combinaisons de <code>and</code> et <code>or</code> en <b>forme normale conjonctive (FNC)</b>

Pour obtenir des exemples montrant comment construire des filtres valides pour chaque cas, consultez [Comment écrire des filtres de collection valides](#).

Si vous écrivez souvent des filtres et qu'il vous serait plus utile de comprendre les principes fondamentaux des règles plutôt que simplement les mémoriser, consultez [Présentation des filtres de collection OData dans la Recherche cognitive Azure](#).

## Étapes suivantes

- [Présentation des filtres de collection OData dans la Recherche cognitive Azure](#)
- [Filtres dans la Recherche cognitive Azure](#)
- [Vue d'ensemble du langage d'expression OData pour Recherche cognitive Azure](#)

- [Informations de référence sur la syntaxe d'expression OData pour Recherche cognitive Azure](#)
- [Rechercher des documents \(API REST de la recherche cognitive Azure\)](#)

# Présentation des filtres de collection OData dans Recherche cognitive Azure

04/10/2020 • 14 minutes to read • [Edit Online](#)

Pour appliquer un [filtre](#) sur des champs de collection dans la Recherche cognitive Azure, vous pouvez utiliser les [opérateurs](#) `any` et `all` avec des [expressions lambda](#). Les expressions lambda sont des expressions booléennes qui font référence à une [variable de portée](#). Les opérateurs `any` et `all` sont analogues à une boucle `for` dans la plupart des langages de programmation, avec la variable de portée jouant le rôle de variable de boucle et l'expression lambda en tant que corps de la boucle. La variable de portée prend la valeur « actuelle » de la collection pendant l'itération de la boucle.

Du moins, c'est ainsi qu'elle fonctionne sur le plan conceptuel. En réalité, Recherche cognitive Azure implémente les filtres d'une manière très différente du fonctionnement des boucles `for`. Dans l'idéal, cette différence est invisible pour vous, mais elle ne l'est pas dans certaines situations. Le résultat final est qu'il existe des règles que vous devez suivre lors de l'écriture d'expressions lambda.

Cet article explique pourquoi les règles applicables aux filtres de collection existent en explorant la façon dont Recherche cognitive Azure Search exécute ces filtres. Si vous écrivez des filtres avancés avec des expressions lambda complexes, cet article peut vous être utile pour améliorer votre compréhension des possibilités offertes par les filtres et pourquoi.

Pour plus d'informations sur ce que sont les règles applicables aux filtres de collection, notamment des exemples, consultez [Résolution des problèmes liés aux filtres de collection OData dans Recherche cognitive Azure](#).

## Raisons de la limitation des filtres de collection

Il existe trois raisons sous-jacentes pour lesquelles toutes les fonctionnalités de filtre ne sont pas prises en charge pour tous les types de collections :

1. Seuls certains opérateurs sont pris en charge pour certains types de données. Par exemple, cela n'a aucun sens de comparer les valeurs booléennes `true` et `false` à l'aide de `lt`, `gt`, etc.
2. Recherche cognitive Azure ne prend pas en charge la [recherche corrélée](#) sur des champs de type `Collection(Edm.ComplexType)`.
3. Recherche cognitive Azure utilise des index inversés pour exécuter des filtres sur tous les types de données, dont les collections.

La première raison est simplement une conséquence de la façon dont le langage OData et le système de type EDM sont définis. Les deux dernières sont expliquées de façon plus détaillée dans le reste de cet article.

## Recherche corrélée et recherche non corrélée

Quand vous appliquez plusieurs critères de filtre sur une collection d'objets complexes, les critères sont [corrélés](#) car ils s'appliquent à *chaque objet de la collection*. Par exemple, le filtre suivant retourne les hôtels ayant au moins une chambre de luxe (deluxe) dont le tarif est inférieur à 100 :

```
Rooms/any(room: room/Type eq 'Deluxe Room' and room/BaseRate lt 100)
```

Si le filtrage a été *sans corrélation*, le filtre ci-dessus peut retourner les hôtels où une chambre est de luxe

(deluxe) et une autre chambre dispose d'un tarif de base inférieur à 100. Cela n'aurait pas de sens, car les deux clauses de l'expression lambda s'appliquent à la même variable de portée, à savoir `room`. C'est pourquoi ces filtres sont mis en corrélation.

Toutefois, pour la recherche en texte intégral, il n'existe aucun moyen de faire référence à une variable de portée spécifique. Si vous utilisez une recherche par champ pour émettre une [requête complète Lucene](#) comme celle-ci :

```
Rooms/Type:deluxe AND Rooms/Description:"city view"
```

vous pouvez obtenir les hôtels où une chambre est de luxe (deluxe) et où une autre chambre mentionne « city view » (vue sur la ville) dans la description. Par exemple, le document ci-dessous avec comme `Id` la valeur `1` correspondrait à la requête :

```
{
  "value": [
    {
      "Id": "1",
      "Rooms": [
        { "Type": "deluxe", "Description": "Large garden view suite" },
        { "Type": "standard", "Description": "Standard city view room" }
      ]
    },
    {
      "Id": "2",
      "Rooms": [
        { "Type": "deluxe", "Description": "Courtyard motel room" }
      ]
    }
  ]
}
```

La raison est que `Rooms/Type` fait référence à tous les termes analysés du champ `Rooms/Type` dans la totalité du document, et il en est de même pour `Rooms/Description`, comme indiqué dans les tableaux ci-dessous.

Façon dont `Rooms/Type` est stocké pour la recherche en texte intégral :

TERME DANS <code>ROOMS/TYPE</code>	ID DE DOCUMENT
deluxe	1, 2
standard	1

Façon dont `Rooms/Description` est stocké pour la recherche en texte intégral :

TERME DANS <code>ROOMS/DESCRIPTION</code>	ID DE DOCUMENT
courtyard	2
city	1
garden	1
large	1
motel	2

TERME DANS	ROOMS/DESCRIPTION	ID DE DOCUMENT
room		1, 2
standard		1
suite		1
vue		1

Par conséquent, contrairement au filtre ci-dessus, qui dit : « Trouver une correspondance avec les documents où une chambre a un **Type** égal à « Deluxe Room » (Chambre de luxe) et où **cette même chambre** a un **BaseRate** (Tarif de base) inférieur à 100 », la requête de recherche indique « Trouver une correspondance avec les documents où **Rooms/Type** (Chambres/Type) contient le terme « deluxe » (de luxe) et où **Rooms/Description** (Chambres/Description) contient l'expression « city view » (vue sur la ville). Il n'existe aucun concept de chambres individuelles dont les champs peuvent être mis en corrélation dans ce dernier cas.

#### NOTE

Si vous voulez voir la prise en charge de la recherche corrélée ajoutée à Recherche cognitive Azure, votez pour [cet élément User Voice](#).

## Index inversés et collections

Vous avez peut-être remarqué qu'il existe beaucoup moins de restrictions concernant les expressions lambda sur les collections complexes que pour les collections simples comme **Collection(Edm.Int32)**, **Collection(Edm.GeographyPoint)**, etc. En fait, Recherche cognitive Azure stocke les collections complexes en tant que collections réelles de sous-documents, tandis que les collections simples ne sont pas du tout stockées en tant que collections.

Par exemple, considérez un champ de collection de chaînes filtrable comme **seasons** dans un index destiné à un détaillant en ligne. Certains documents chargés dans cet index peuvent se présenter comme suit :

```
{
  "value": [
    {
      "id": "1",
      "name": "Hiking boots",
      "seasons": ["spring", "summer", "fall"]
    },
    {
      "id": "2",
      "name": "Rain jacket",
      "seasons": ["spring", "fall", "winter"]
    },
    {
      "id": "3",
      "name": "Parka",
      "seasons": ["winter"]
    }
  ]
}
```

Les valeurs du champ **seasons** sont stockées dans une structure appelée **index inversé**, qui ressemble à ceci :

TERME	ID DE DOCUMENT
spring	1, 2
summer	1
fall	1, 2
winter	2, 3

Cette structure de données est conçue pour répondre très rapidement à une question : Dans quels documents un terme donné apparaît-il ? Réponse à cette question équivaut davantage à un contrôle d'égalité simple qu'à une boucle sur une collection. En fait, c'est la raison pour laquelle, pour les collections de chaînes, Recherche cognitive Azure autorise uniquement `eq` comme opérateur de comparaison dans une expression lambda pour `any`.

En partant de l'égalité, nous allons ensuite voir comment il est possible de combiner plusieurs contrôles d'égalité sur la même variable de plage avec `or`. Cela fonctionne grâce à l'algèbre et à [la propriété distributive des quantificateurs](#). Cette expression :

```
seasons/any(s: s eq 'winter' or s eq 'fall')
```

équivaut à :

```
seasons/any(s: s eq 'winter') or seasons/any(s: s eq 'fall')
```

et chacune des deux sous-expressions `any` peut être exécutée efficacement à l'aide de l'index inversé. De plus, grâce à [la loi de la négation des quantificateurs](#), cette expression :

```
seasons/all(s: s ne 'winter' and s ne 'fall')
```

équivaut à :

```
not seasons/any(s: s eq 'winter' or s eq 'fall')
```

C'est pourquoi il est possible d'utiliser `all` avec `ne` et `and`.

#### NOTE

Même si ce document ne contient pas de présentation détaillée, ces mêmes principes s'étendent également aux [tests de distance et d'intersection pour les collections de points de données géospatiales](#). C'est pourquoi, dans `any` :

- `geo.intersects` ne peut pas être inversé
- `geo.distance` doit être comparé à l'aide de `lt` ou de `le`
- Les expressions doivent être combinées avec `or`, et non `and`

Les règles inverses s'appliquent pour `all`.

Un plus large choix d'expressions sont autorisées lors du filtrage sur des collections de types de données qui prennent en charge les opérateurs `lt`, `gt`, `le` et `ge`, comme `Collection(Edm.Int32)` par exemple. Plus précisément, vous pouvez utiliser `and` ainsi que `or` dans `any`, à condition que les expressions de comparaison

sous-jacentes soient combinées en **comparaisons de plages** à l'aide de `and`, qui sont ensuite combinées à l'aide de `or`. Cette structure d'expressions booléennes est appelée **forme disjonctive normale (FND)**, également appelée « ORs de ANDs ». À l'inverse, les expressions lambda pour `all` pour ces types de données doivent être en **forme normale conjonctive (FNC)**, également appelée « ANDs de ORs ». Recherche cognitive Azure permet de telles comparaisons de plages car elle peut les exécuter efficacement à l'aide d'index inversés, tout comme elle peut effectuer une recherche de termes rapide pour les chaînes.

En résumé, voici les règles générales concernant ce qui est autorisé dans une expression lambda :

- Dans `any`, les *contrôles positifs* sont toujours autorisés, comme l'égalité, les comparaisons de plages, `geo.intersects` ou `geo.distance` comparé à `1t` ou `1e` (considérez la « proximité » comme une égalité quand il s'agit d'un contrôle de distance).
- Dans `any`, `or` est toujours autorisé. Vous pouvez utiliser `and` uniquement pour les types de données qui peuvent exprimer des contrôles de plage, et uniquement si vous utilisez des ORs de ANDs (FND).
- Dans `all`, les règles sont inversées : seuls les *contrôles négatifs* sont autorisés. Vous pouvez toujours utiliser `and`, et vous pouvez utiliser `or` uniquement pour les contrôles de plage exprimés sous forme de ANDs de ORs (FNC).

Dans la pratique, voici les types de filtres que vous êtes le plus susceptible d'utiliser. Il est cependant toujours utile de comprendre les limites de ce qui est possible.

Pour obtenir des exemples spécifiques des types de filtres qui sont autorisés et de ceux qui ne le sont pas, consultez [Comment écrire des filtres de collection valides](#).

## Étapes suivantes

- [Résolution de problèmes liés aux filtres de collection OData dans Recherche cognitive Azure](#)
- [Filtres dans la Recherche cognitive Azure](#)
- [Vue d'ensemble du langage d'expression OData pour Recherche cognitive Azure](#)
- [Informations de référence sur la syntaxe d'expression OData pour Recherche cognitive Azure](#)
- [Rechercher des documents \(API REST de la recherche cognitive Azure\)](#)

# Comment filtrer par langue dans la Recherche cognitive Azure

04/10/2020 • 5 minutes to read • [Edit Online](#)

Une condition requise dans une application de recherche multilingue est la possibilité d'effectuer une recherche et de récupérer les résultats dans la langue de l'utilisateur. Dans la Recherche cognitive Azure, une manière de remplir les conditions de langue d'une application multilingue consiste à créer une série de champs dédiés au stockage de chaînes dans une langue spécifique, puis à limiter la recherche en texte intégral à ces seuls champs au moment de la requête.

Des paramètres de requête appliqués à la demande sont utilisés pour déterminer l'étendue de l'opération de recherche, puis écarter les résultats de tous les champs qui ne fournissent pas de contenu compatible avec l'expérience de recherche que vous souhaitez offrir.

PARAMÈTRES	OBJECTIF
<b>searchFields</b>	Limite la recherche en texte intégral à la liste des champs nommés.
<b>\$select</b>	Réduit la réponse pour inclure uniquement les champs que vous spécifiez. Par défaut, tous les champs récupérables sont retournés. Le paramètre <b>\$select</b> vous permet de choisir les champs à retourner.

Le succès de cette technique dépend de l'intégrité du contenu des champs. La Recherche cognitive Azure ne convertit pas les chaînes et n'effectue pas de détection de la langue. Il vous appartient de vous assurer que les champs contiennent les chaînes que vous attendez.

## Définir des champs de contenu dans différentes langues

Dans la Recherche cognitive Azure, les requêtes ciblent un index unique. Les développeurs qui souhaitent fournir des chaînes spécifiques d'une langue dans une expérience de recherche unique définissent généralement des champs dédiés pour stocker les valeurs : un champ pour les chaînes en français, un autre pour les chaînes en anglais, et ainsi de suite.

L'exemple suivant est extrait de l'[exemple d'immobilier](#), qui comporte plusieurs champs de chaîne avec du contenu dans différentes langues. Notez les assignations de l'analyseur de langue pour les champs de cet index. Les champs qui contiennent des chaînes produisent de meilleurs résultats en lien avec une recherche en texte intégral quand ils sont associés à un analyseur conçu pour traiter les règles linguistiques de la langue cible.

Fields  
realestate-us-sample

Save Discard

Basic Analyzer Suggester

FIELD NAME	TYPE	SEARCHABLE	ANALYZER
listingId	Edm.String	<input type="checkbox"/>	
beds	Edm.Int32	<input type="checkbox"/>	
baths	Edm.Int32	<input type="checkbox"/>	
description	Edm.String	<input checked="" type="checkbox"/>	English - Microsoft
description_de	Edm.String	<input checked="" type="checkbox"/>	German - Microsoft
description_fr	Edm.String	<input checked="" type="checkbox"/>	French - Microsoft
description_it	Edm.String	<input checked="" type="checkbox"/>	Italian - Microsoft
description_es	Edm.String	<input checked="" type="checkbox"/>	Spanish - Microsoft
description_pl	Edm.String	<input checked="" type="checkbox"/>	Polish - Microsoft
description_nl	Edm.String	<input checked="" type="checkbox"/>	Dutch - Microsoft

#### NOTE

Pour des exemples de code montrant des définitions de champ avec des analyseurs de langue, voir [Définir un index \(.NET\)](#) et [Définir un index \(REST\)](#).

## Créer et charger un index

La [création et le remplissage de l'index](#) sont une étape intermédiaire (et peut-être évidente) avant la formulation d'une requête. Nous signalons cette étape ici par souci d'exhaustivité. Une manière de déterminer si l'index est disponible consiste à vérifier la liste des index dans le [portail](#).

## Limiter la requête et réduire les résultats

Les paramètres de la requête sont utilisés pour limiter la recherche à des champs spécifiques, puis écarter les résultats des champs dépourvus d'utilité pour votre scénario. Si votre objectif est de limiter la recherche aux champs contenant des chaînes en français, utilisez `searchFields` pour cibler la requête sur les champs contenant des chaînes dans cette langue.

Par défaut, une recherche retourne tous les champs marqués comme récupérables. Par conséquent, il peut être utile d'exclure les champs non conformes à l'expérience de recherche spécifique de la langue que vous souhaitez offrir. Plus précisément, si vous limitez la recherche à un champ contenant des chaînes en français, vous voulez probablement exclure des résultats les champs contenant des chaînes en anglais. Le paramètre de requête `$select` vous permet de contrôler les champs qui sont retournés à l'application appelante.

```
parameters =
    new SearchParameters()
{
    searchFields = "description_fr"
    Select = new[] { "description_fr"  }
};
```

#### NOTE

Bien que la requête ne contienne pas d'argument \$filter, ce cas d'utilisation est étroitement apparenté aux concepts de filtre. C'est pourquoi il est présenté comme un scénario de filtrage.

## Voir aussi

- [Filtres dans la Recherche cognitive Azure](#)
- [Analyseurs de langage](#)
- [Fonctionnement de la recherche en texte intégral dans la Recherche cognitive Azure](#)
- [API REST de recherche de documents](#)

# Guide pratique pour utiliser les résultats de recherche dans Recherche cognitive Azure

04/10/2020 • 14 minutes to read • [Edit Online](#)

Cet article explique comment obtenir une réponse à une requête qui renvoie le nombre total de documents correspondants, les résultats paginés, les résultats triés et les termes mis en surbrillance.

La structure d'une réponse est déterminée par les paramètres de la requête : [Search Document](#) dans l'API REST, ou [DocumentSearchResult Class](#) dans le Kit de développement logiciel (SDK) .NET.

## Composition des résultats

Bien qu'un document de recherche puisse comporter un grand nombre de champs, en général, seuls quelques-uns sont nécessaires pour représenter chaque document dans le jeu de résultats. Dans une demande de requête, ajoutez `$select=<field list>` pour spécifier les champs qui s'affichent dans la réponse. Un champ doit être attribué en tant que **Récupérable** dans l'index pour être inclus dans un résultat.

Les champs qui fonctionnent le mieux incluent ceux qui distinguent et différencient les documents, en fournissant suffisamment d'informations pour inviter l'utilisateur à cliquer sur le document. Sur un site d'e-commerce, il peut s'agir d'un nom de produit, d'une description, d'une couleur, d'une taille, d'un prix et d'une évaluation. Pour l'exemple intégré hotels-sample-index, il peut s'agir de champs dans l'exemple suivant :

```
POST /indexes/hotels-sample-index/docs/search?api-version=2020-06-30
{
    "search": "sandy beaches",
    "select": "HotelId, HotelName, Description, Rating, Address/City"
    "count": true
}
```

### NOTE

Si vous souhaitez inclure des fichiers image dans un résultat, tel qu'une photo de produit ou un logo, stockez-les en dehors de Recherche cognitive Azure, mais incluez un champ dans votre index pour référencer l'URL d'image dans le document de recherche. Les exemples d'index qui prennent en charge les images dans les résultats incluent la démonstration [realestate-sample-us](#), présentée dans ce [guide de démarrage rapide](#), et l'[application de démonstration New York City Jobs](#).

## Résultats de pagination

Par défaut, le moteur de recherche retourne jusqu'aux 50 premières correspondances, comme déterminé par le score de recherche si la requête est une recherche en texte intégral, ou dans un ordre arbitraire pour les requêtes de correspondance exacte.

Pour retourner un nombre différent de documents correspondants, ajoutez les paramètres `$top` et `$skip` à la demande de requête. La liste suivante explique la logique.

- Ajoutez `$count=true` pour obtenir le nombre total de documents correspondants dans un index.
- Retournez le premier jeu de 15 documents correspondants plus un nombre total de correspondances :

```
GET /indexes/<INDEX-NAME>/docs?search=<QUERY STRING>&$top=15&$skip=0&$count=true
```

- Retournez le deuxième jeu, en ignorant les 15 premiers pour obtenir les 15 suivants : `$top=15&$skip=15` .

Procédez de la même façon pour le troisième jeu de 15 : `$top=15&$skip=30`

Il n'est pas garanti que les résultats des requêtes paginées soient stables si l'index sous-jacent est modifié. La pagination modifie la valeur de `$skip` pour chaque page, mais chaque requête est indépendante et opère sur l'affichage actuel des données telles qu'elles existent dans l'index au moment de la requête. En d'autres termes, il n'y a aucune mise en cache ni capture instantanée des résultats, comme c'est le cas dans une base de données à usage général. Voici un exemple de la façon dont vous pouvez obtenir des doublons. Imaginons un index avec quatre documents :

```
{ "id": "1", "rating": 5 }
{ "id": "2", "rating": 3 }
{ "id": "3", "rating": 2 }
{ "id": "4", "rating": 1 }
```

Supposons à présent que vous souhaitez que les résultats soient retournés par deux, classés par évaluation. Vous exécuterez cette requête pour obtenir la première page des résultats,

`$top=2&$skip=0&$orderby=rating desc`, et vous obtenez les résultats suivants :

```
{ "id": "1", "rating": 5 }
{ "id": "2", "rating": 3 }
```

Sur le service, supposez qu'un cinquième document est ajouté à l'index entre les appels de requête :

`{ "id": "5", "rating": 4 }`. Peu de temps après, vous exécutez une requête pour extraire la deuxième page, `$top=2&$skip=2&$orderby=rating desc`, et vous obtenez ces résultats :

```
{ "id": "2", "rating": 3 }
{ "id": "3", "rating": 2 }
```

Notez que le document 2 est extrait 2 fois. Cela est dû au fait que le nouveau document 5 a une plus grande valeur d'évaluation. Il est donc trié avant le document 2 et atterrit sur la première page. Bien que ce comportement puisse être inattendu, c'est généralement ainsi qu'un moteur de recherche se comporte.

## Classement des résultats

Pour les requêtes de recherche en texte intégral, les résultats sont automatiquement classés en fonction d'un score de recherche, calculé sur la base de la fréquence et de la proximité des termes dans un document, les scores les plus élevés étant attribués aux documents présentant des correspondances plus nombreuses ou plus fortes sur un terme de recherche.

Les scores de recherche donnent une impression générale de pertinence, reflétant la force de la correspondance par rapport à d'autres documents du même jeu de résultats. Les scores ne sont pas toujours cohérents d'une requête à l'autre. Par conséquent, lorsque vous utilisez des requêtes, vous pouvez remarquer des différences mineures dans l'ordre des documents recherchés. Il existe plusieurs explications à cette situation.

CAUSE	DESCRIPTION
Volatilité des données	Le contenu de l'index varie au fur et à mesure que vous ajoutez, modifiez ou supprimez des documents. La fréquence des termes changera à mesure que les mises à jour de l'index seront traitées, ce qui aura un impact sur les scores de recherche des documents correspondants.

CAUSE	DESCRIPTION
Réplicas multiples	Pour les services qui utilisent plusieurs réplicas, les requêtes sont émises en parallèle pour chaque réplica. Les statistiques d'index utilisées pour calculer un score de recherche sont calculées par réplica, les résultats étant fusionnés et classés dans la réponse de la requête. Les réplicas sont principalement des miroirs les uns des autres, mais les statistiques peuvent varier en raison de petites différences d'état. Par exemple, un réplica peut avoir supprimé des documents contribuant à leurs statistiques, qui ont été fusionnées à partir d'autres réplicas. En règle générale, les différences dans les statistiques par réplica sont plus perceptibles dans les index plus petits.
Scores identiques	Si plusieurs documents ont le même score, chacun d'entre eux peut apparaître en premier.

## Classement cohérent

Étant donné la flexibilité dans le classement des résultats, vous souhaiterez peut-être explorer d'autres options si la cohérence est une exigence de l'application. L'approche la plus simple consiste à trier en fonction d'une valeur de champ, comme une évaluation ou une date. Pour les scénarios où vous souhaitez effectuer un tri en fonction d'un champ spécifique, tel qu'une évaluation ou une date, vous pouvez définir explicitement une expression `$orderby`, qui peut être appliquée à n'importe quel champ indexé en tant que **Triable**.

Une autre option consiste à utiliser un [profil de scoring personnalisé](#). Les profils de scoring vous permettent de mieux contrôler le classement des éléments dans les résultats de recherche, avec la possibilité d'augmenter le nombre de correspondances trouvées dans des champs spécifiques. La logique de scoring supplémentaire peut aider à surmonter les différences mineures entre les réplicas, car les scores de recherche pour chaque document sont plus éloignés les uns des autres. Nous vous recommandons d'utiliser l'[algorithme de classement](#) pour cette approche.

## Mise en surbrillance des correspondances

La mise en surbrillance des correspondances fait référence à la mise en forme de texte (par exemple, caractères gras ou surlignage jaune) appliquée au terme correspondant dans un résultat, ce qui facilite le repérage de l'occurrence. Des instructions pour la mise en surbrillance des correspondances sont fournies dans la [demande de requête](#).

Pour activer la mise en surbrillance des correspondances, ajoutez

`highlight=[comma-delimited list of string fields]` pour spécifier les champs qui utiliseront la mise en surbrillance. La mise en surbrillance est utile pour des champs de contenu longs, tels qu'un champ de description, où la correspondance n'est pas immédiatement évidente. Seules les définitions de champs attribuées comme **pouvant faire l'objet d'une recherche** sont éligibles pour la mise en surbrillance des correspondances.

Par défaut, la Recherche cognitive Azure renvoie jusqu'à cinq éléments en surbrillance par champ. Vous pouvez ajuster ce nombre en ajoutant au champ un tiret suivi d'un entier. Par exemple, `highlight=Description-10` retourne jusqu'à 10 mises en surbrillance de contenu correspondant dans le champ Description.

La mise en forme est appliquée aux requêtes de termes entières. Le type de mise en forme est déterminé par des balises, `highlightPreTag` et `highlightPostTag`, et votre code gère la réponse (par exemple, en appliquant une police en gras ou un arrière-plan jaune).

Dans l'exemple suivant, les termes « sablonneux », « sable », « plages » et « plage » trouvés dans le champ Description sont balisés pour la mise en surbrillance. Les requêtes qui déclenchent une extension de requête

dans le moteur, telles que les recherches floues ou par caractères génériques, offrent une prise en charge limitée de la mise en surbrillance des correspondances.

```
GET /indexes/hotels-sample-index/docs/search=search=sandy beaches&highlight=Description?api-version=2020-06-30
```

```
POST /indexes/hotels-sample-index/docs/search?api-version=2020-06-30
{
  "search": "sandy beaches",
  "highlight": "Description"
}
```

### Nouveau comportement (à partir du 15 juillet)

Les services créés après le 15 juillet 2020 offriront une expérience de mise en surbrillance différente. Le comportement de mise en surbrillance ne varie pas pour les services créés avant cette date.

Avec le nouveau comportement :

- Seules les expressions qui correspondent à la requête d'expression complète sont renvoyées. La requête « super bowl » retourne les résultats mis en surbrillance comme suit :

```
'<em>super bowl</em> is super awesome with a bowl of chips'
```

Notez que le terme *bowl of chips* n'est pas mis en surbrillance, car il ne correspond pas à l'expression complète.

Lorsque vous écrivez du code client qui implémente la mise en surbrillance des correspondances, tenez compte de cette modification. Notez que cela n'aura aucun impact sauf si vous créez un service de recherche entièrement nouveau.

## Étapes suivantes

Pour générer rapidement une page de recherche pour votre client, envisagez les options suivantes :

- Dans le portail, le [générateur d'applications](#) crée une page HTML avec une barre de recherche, une navigation par facettes et une zone de résultats qui contient des images.
- Le tutoriel [Créer votre première application en C#](#) permet de créer un client fonctionnel. L'exemple de code illustre les requêtes paginées, la mise en surbrillance des correspondances et le tri.

Plusieurs exemples de code comportent une interface frontale web qui se trouve ici : [Application de démonstration New York City Jobs](#), [exemple de code JavaScript avec un site de démonstration en direct](#) et [CognitiveSearchFrontEnd](#).

# Similarité et scoring dans Recherche cognitive Azure

04/10/2020 • 11 minutes to read • [Edit Online](#)

Le scoring consiste à calculer un score de recherche pour chaque élément retourné dans des résultats de recherche pour des requêtes de texte intégral. Le score est un indicateur de la pertinence d'un élément dans le contexte de l'opération de recherche en cours. Plus le score est élevé, plus l'élément est pertinent. Dans des résultats de recherche, les éléments sont classés par ordre décroissant de pertinence, sur la base des résultats de recherche calculés pour chacun d'eux.

Par défaut, les 50 premiers sont retournés dans la réponse, mais vous pouvez utiliser le paramètre `$top` pour retourner un nombre inférieur ou supérieur d'éléments (jusqu'à 1000 par réponse) et le paramètre `$skip` pour obtenir le jeu de résultats suivant.

Le score de recherche est calculé sur la base de propriétés statistiques des données et de la requête. Recherche cognitive Azure trouve les documents qui contiennent des correspondances de recherche (en totalité ou en partie, selon `searchMode`), en favorisant les documents qui contiennent de nombreuses occurrences du terme recherché. Le score de recherche augmente davantage si le terme est rare dans l'index de données, mais courant au sein du document. La base de cette approche de la pertinence du calcul est appelée *TF-IDF* ou Term Frequency-Inverse Document Frequency (fréquence de terme-fréquence inverse de document).

Des valeurs de score de recherche peuvent être répétées dans un jeu de résultats. Quand plusieurs correspondances ont le même score de recherche, le classement des éléments ayant le même score n'est ni défini ni stable. Réexécutez la requête, et vous constaterez peut-être que des éléments changent de position, en particulier si vous utilisez le service gratuit ou un service facturable avec plusieurs répliques. Si deux éléments ont un score identique, il est impossible de prédire celui qui apparaîtra en première position.

Si vous souhaitez départager des scores identiques, vous pouvez ajouter une clause `$orderby` afin de trier d'abord par score, puis par un autre champ pouvant être trié (par exemple

`$orderby=search.score() desc,Rating desc`). Pour plus d'informations, consultez [\\$orderby](#).

## NOTE

Un `@search.score = 1.00` indique un jeu de résultats sans score ou non classé. Le score est uniforme parmi tous les résultats. Des résultats sans score se produisent quand le formulaire de requête est une recherche approximative, des requêtes Regex ou de caractères génériques, ou une expression `$filter`.

## Profils de score

Vous pouvez personnaliser la façon dont les différents champs sont classés en définissant un *profil de scoring* personnalisé. Les profils de score vous permettent de mieux contrôler le classement d'éléments dans des résultats de recherche. Par exemple, vous pouvez privilégier des éléments en fonction de leur revenu potentiel, promouvoir des éléments plus récents, voire en favoriser d'autres restés trop longtemps en stock.

Un profil de score fait partie de la définition d'index, composée de champs, fonctions et paramètres pondérés. Pour plus d'informations sur la définition d'un profil de scoring, consultez [Profils de scoring](#).

## Statistiques de scoring et sessions rémanentes

À des fins de scalabilité, Recherche cognitive Azure distribue chaque index horizontalement par le biais d'un processus de partitionnement, ce qui signifie que les portions d'un index sont physiquement séparées.

Par défaut, le score d'un document est calculé en fonction de propriétés statistiques des données *au sein d'une partition*. Cette approche n'est généralement pas un problème pour un corpus important de données, et elle offre de meilleures performances que le calcul du score à partir des informations de l'ensemble des partitions. Cela dit, avec cette optimisation des performances, deux documents très similaires (ou même des documents identiques) risquent de se retrouver avec des scores de pertinence différents s'ils figurent dans des partitions différentes.

Si vous préférez calculer le score à partir des propriétés statistiques sur l'ensemble des partitions, vous pouvez le faire en ajoutant *scoringStatistics=global* en tant que [paramètre de requête](#) (ou en ajoutant "*scoringStatistics*": "*global*" en tant que paramètre de corps de la [demande de requête](#)).

```
GET https://[service name].search.windows.net/indexes/[index name]/docs?scoringStatistics=global&api-version=2020-06-30&search=[search term]
Content-Type: application/json
api-key: [admin or query key]
```

L'utilisation de *scoringStatistics* garantit que toutes les partitions dans le même réplica fournissent les mêmes résultats. Cela dit, plusieurs réplicas peuvent être légèrement différents les uns des autres, car ils sont toujours mis à jour avec les dernières modifications apportées à votre index. Dans certains scénarios, vous souhaiterez peut-être que vos utilisateurs obtiennent des résultats plus cohérents pendant une « session de requête ». Dans de tels scénarios, vous pouvez fournir un `sessionId` dans le cadre de vos requêtes. Le `sessionId` est une chaîne unique que vous créez pour faire référence à une session utilisateur unique.

```
GET https://[service name].search.windows.net/indexes/[index name]/docs?sessionId=[string]&api-version=2020-06-30&search=[search term]
Content-Type: application/json
api-key: [admin or query key]
```

Tant que le même `sessionId` est utilisé, la méthode de la meilleure tentative possible est utilisée pour cibler le même réplica, ce qui améliore la cohérence des résultats présentés à vos utilisateurs.

#### NOTE

La réutilisation des mêmes valeurs `sessionId` à plusieurs reprises peut interférer avec l'équilibrage de la charge des demandes sur les réplicas et nuire aux performances du service de recherche. La valeur utilisée comme `sessionId` ne peut pas commencer par un caractère « \_ ».

## Algorithmes de classement de similarité

Recherche cognitive Azure prend en charge deux algorithmes de classement de similarité différents : un algorithme de *similarité classique* et l'implémentation officielle de l'algorithme *Okapi BM25* (actuellement en préversion). L'algorithme de similarité classique est l'algorithme par défaut, mais à compter du 15 juillet tous les nouveaux services créés utiliseront le nouvel algorithme BM25. Il s'agira du seul algorithme disponible sur les nouveaux services.

Pour le moment, vous pouvez spécifier l'algorithme de classement de similarité que vous souhaitez utiliser. Pour plus d'informations, consultez [Algorithme de classement](#).

Le segment vidéo suivant permet d'accéder rapidement à une explication des algorithmes de classement utilisés dans Recherche cognitive Azure. Vous pouvez regarder la vidéo complète pour plus d'informations.

## Paramètre `featuresMode` (préversion)

Les requêtes [Rechercher des documents](#) ont un nouveau paramètre `featuresMode` qui peut fournir des détails supplémentaires sur la pertinence au niveau du champ. Tandis que `@searchScore` est calculé pour l'ensemble du document (quelle est la pertinence de ce document dans le contexte de cette requête), par le biais de `featuresMode`, vous pouvez obtenir des informations sur des champs individuels, tels qu'ils sont exprimés dans une structure `@search.features`. La structure contient tous les champs utilisés dans la requête (soit des champs spécifiques par le biais de `searchFields` dans une requête, soit tous les champs attribués comme `interrogeables` dans un index). Pour chaque champ, vous pouvez obtenir les valeurs suivantes :

- Nombre de jetons uniques trouvés dans le champ
- Score de similarité, ou mesure de la similitude entre le contenu du champ et le terme recherché
- Fréquence du terme, ou nombre de fois où le terme recherché a été trouvé dans le champ

Pour une requête qui cible les champs « Description » et « Titre », une réponse incluant `@search.features` peut se présenter comme suit :

```
"value": [  
  {  
    "@search.score": 5.1958685,  
    "@search.features": {  
      "description": {  
        "uniqueTokenMatches": 1.0,  
        "similarityScore": 0.29541412,  
        "termFrequency" : 2  
      },  
      "title": {  
        "uniqueTokenMatches": 3.0,  
        "similarityScore": 1.75451557,  
        "termFrequency" : 6  
      }  
    }  
  }]
```

Vous pouvez utiliser ces points de données dans [des solutions de scoring personnalisées](#) ou utiliser les informations pour déboguer des problèmes de pertinence des recherches.

## Voir aussi

[Profils de scoring Référence de l'API REST](#)

[API de recherche dans des documents](#)

[Kit de développement logiciel \(SDK\) de Recherche cognitive Azure .NET](#)

# Ajouter des profils de score à un index Recherche cognitive Azure

04/10/2020 • 31 minutes to read • [Edit Online](#)

Un *scoring* détermine un score de recherche pour chaque élément dans un jeu de résultats classés par rang. Un score de recherche est attribué à chaque élément d'un jeu de résultats de recherche. Ils sont ensuite classés du rang le plus élevé au rang le plus bas.

Le service Recherche cognitive Azure utilise un score par défaut pour calculer un score initial, mais vous pouvez personnaliser le calcul à l'aide d'un *profil de score*. Les profils de score vous permettent de mieux contrôler le classement d'éléments dans des résultats de recherche. Par exemple, vous pouvez privilégier des éléments en fonction de leur revenu potentiel, promouvoir des éléments plus récents, voire en favoriser d'autres restés trop longtemps en stock.

Le segment vidéo suivant permet d'accéder rapidement au fonctionnement des profils de score dans Recherche cognitive Azure.

## Définitions de profil de score

Un profil de score fait partie de la définition d'index, composée de champs, fonctions et paramètres pondérés.

Pour vous donner une idée de l'apparence d'un profil de score, l'exemple suivant présente un simple profil « géographique ». Celui-ci privilégie les éléments dont le champ **hotelName** contient le terme recherché. Il utilise également la fonction `distance` pour favoriser les éléments qui se situent dans un rayon de dix kilomètres par rapport à l'emplacement actuel. Si quelqu'un effectue une recherche sur le terme « inn » dans un rayon de 10 km par rapport à la position actuelle, les documents d'hôtels dont le nom contient cette chaîne de caractères apparaissent en tête des résultats de la recherche.

```
"scoringProfiles": [
  {
    "name": "geo",
    "text": {
      "weights": {
        "hotelName": 5
      }
    },
    "functions": [
      {
        "type": "distance",
        "boost": 5,
        "fieldName": "location",
        "interpolation": "logarithmic",
        "distance": {
          "referencePointParameter": "currentLocation",
          "boostingDistance": 10
        }
      }
    ]
  }
]
```

Pour utiliser ce profil de score, votre requête est formulée de façon à spécifier le profil de la chaîne de

caractères de requête. Dans la requête ci-dessous, notez la présence du paramètre de requête `scoringProfile=geo`.

```
GET /indexes/hotels/docs?search=inn&scoringProfile=geo&scoringParameter=currentLocation--  
122.123.44.77233&api-version=2020-06-30
```

Cette requête effectue une recherche du terme « inn », puis transmet l'emplacement actuel. Notez que cette requête inclut d'autres paramètres, tels que `scoringParameter`. Les paramètres de requête sont décrits dans [Recherche dans des documents \(API REST de Recherche cognitive Azure\)](#).

Pour voir un exemple plus détaillé de profil de score, cliquez sur [Exemple](#)

## Qu'est-ce qu'un calcul de score par défaut ?

Un calcul de score détermine un score de recherche pour chaque élément dans un jeu de résultats classés par rang. Un score de recherche est attribué à chaque élément d'un jeu de résultats de recherche. Ils sont ensuite classés du rang le plus élevé au rang le plus bas. Les éléments dont les scores sont plus élevés sont renvoyés à l'application. Par défaut, il s'agit des 50 premiers éléments, mais le paramètre `$top` vous permet de définir le renvoi d'un nombre supérieur ou inférieur d'éléments (jusqu'à 1 000 par réponse).

Le score de recherche est calculé sur la base de propriétés statistiques des données et de la requête. Recherche cognitive Azure trouve les documents qui contiennent (en totalité ou en partie, selon le `searchMode`) les termes recherchés indiqués dans la chaîne de requête, en favorisant les documents qui contiennent de nombreuses occurrences du terme recherché. Le score de recherche augmente davantage si le terme est rare dans l'index de données, mais courant au sein du document. La base de cette approche de la pertinence du calcul est appelée [TF-IDF](#) ou Term Frequency-Inverse Document Frequency (fréquence de terme-fréquence inverse de document).

En supposant qu'il n'y a pas de tri personnalisé, les résultats sont classés par score de recherche avant d'être renvoyés à l'application appelante. Si la valeur `$top` n'est pas spécifiée, les 50 éléments dont le score de recherche est le plus élevé sont renvoyés.

Des valeurs de score de recherche peuvent être répétées dans un jeu de résultats. Par exemple, vous pouvez avoir dix éléments dont le score est 1,2, vingt éléments dont le score est 1,0, et vingt éléments dont le score est 0,5. Quand plusieurs correspondances ont le même score de recherche, le classement des éléments ayant le même score n'est ni défini ni stable. Si vous exécutez de nouveau la requête, il se peut que des éléments changent de position. Si deux éléments ont un score identique, il est impossible de prédire celui qui apparaîtra en première position.

## Quand ajouter une logique de notation

Quand le classement par défaut produit des résultats trop éloignés de vos objectifs, vous devez créer un ou plusieurs profils de calcul de score. Par exemple, vous pouvez décider que la pertinence de la recherche doit privilégier les éléments récemment ajoutés. De même, il se peut qu'un champ affiche une marge bénéficiaire, ou un autre un revenu potentiel. La volonté de privilégier les correspondances qui génèrent des bénéfices pour votre entreprise peut être un facteur important dans la décision d'utiliser des profils de calcul de score.

Un classement basé sur la pertinence peut également être mis en œuvre par l'intermédiaire de profils de calcul de score. Songez aux pages de résultats de recherche que vous avez utilisées par le passé, qui vous permettaient de trier par prix, date, évaluation ou pertinence. Dans Recherche cognitive Azure, les profils de calcul de score déterminent l'option « pertinence ». Vous contrôlez la définition de la pertinence en fonction de vos objectifs et du type d'expérience de recherche que vous souhaitez.

## Exemple

Comme indiqué précédemment, un calcul de score personnalisé est mis en œuvre à l'aide d'un ou de plusieurs profils de calcul de score définis dans un schéma d'index.

Cet exemple montre le schéma d'un index comprenant deux profils de calcul de score (`boostGenre`, `newAndHighlyRated`). Toute requête sur cet index qui comprend un profil comme paramètre de requête utilise le profil pour évaluer le jeu de résultats.

```
{
  "name": "musicstoreindex",
  "fields": [
    { "name": "key", "type": "Edm.String", "key": true },
    { "name": "albumTitle", "type": "Edm.String" },
    { "name": "albumUrl", "type": "Edm.String", "filterable": false },
    { "name": "genre", "type": "Edm.String" },
    { "name": "genreDescription", "type": "Edm.String", "filterable": false },
    { "name": "artistName", "type": "Edm.String" },
    { "name": "orderableOnline", "type": "Edm.Boolean" },
    { "name": "rating", "type": "Edm.Int32" },
    { "name": "tags", "type": "Collection(Edm.String)" },
    { "name": "price", "type": "Edm.Double", "filterable": false },
    { "name": "margin", "type": "Edm.Int32", "retrievable": false },
    { "name": "inventory", "type": "Edm.Int32" },
    { "name": "lastUpdated", "type": "Edm.DateTimeOffset" }
  ],
  "scoringProfiles": [
    {
      "name": "boostGenre",
      "text": {
        "weights": {
          "albumTitle": 1.5,
          "genre": 5,
          "artistName": 2
        }
      }
    },
    {
      "name": "newAndHighlyRated",
      "functions": [
        {
          "type": "freshness",
          "fieldName": "lastUpdated",
          "boost": 10,
          "interpolation": "quadratic",
          "freshness": {
            "boostingDuration": "P365D"
          }
        },
        {
          "type": "magnitude",
          "fieldName": "rating",
          "boost": 10,
          "interpolation": "linear",
          "magnitude": {
            "boostingRangeStart": 1,
            "boostingRangeEnd": 5,
            "constantBoostBeyondRange": false
          }
        }
      ]
    }
  ],
  "suggesters": [
    {
      "name": "sg",
      "searchMode": "analyzingInfixMatching",
      "sourceFields": [ "albumTitle", "artistName" ]
    }
  ]
}
```

## Workflow

Pour mettre en œuvre un calcul de score personnalisé, ajoutez un profil de calcul de score au schéma qui définit l'index. Vous pouvez avoir jusqu'à 100 profils de calcul de score au sein d'un même index (consultez [Limites de service](#)), mais vous ne pouvez spécifier qu'un seul profil à la fois dans une requête donnée.

Commencez par le [Modèle](#) fourni dans cette rubrique.

Donnez-lui un nom. Les profils de calcul de score sont facultatifs mais, si vous en ajoutez un, le nom est obligatoire. Veillez à respecter les conventions d'affectation des noms pour les champs (commencer par une lettre, en évitant les caractères spéciaux et les mots réservés). Consultez [Règles d'attribution de noms](#) ([Recherche cognitive Azure](#)) pour obtenir la liste complète.

Le corps du profil de calcul de score est construit à partir de champs et de fonctions pondérés.

<b>Pondérations</b>	Spécifiez les paires nom-valeur qui affectent une pondération relative à un champ. Dans l' <a href="#">exemple</a> , les valeurs de pondération des champs albumTitle, genre et artistName sont respectivement 1,5, 5 et 2. Pourquoi la pondération du champ genre est-elle beaucoup plus élevée que celle des autres champs ? Si la recherche est effectuée sur des données relativement homogènes (comme c'est le cas du « genre » dans le <code>musicstoreindex</code> ), il se peut que vous ayez besoin d'une variance plus importante dans les pondérations relatives. Par exemple, dans le <code>musicstoreindex</code> , « rock » apparaît à la fois comme genre et dans des descriptions de genre formulées de façon identique. Si vous souhaitez que le genre ait une pondération plus élevée que la description du genre, la pondération relative du champ Genre doit être sensiblement plus importante.
---------------------	---

## Fonctions

Utilisées lorsque des calculs supplémentaires sont nécessaires dans des contextes spécifiques. Les valeurs valides sont `freshness`, `magnitude`, `distance` et `tag`. Chaque fonction a des paramètres qui sont uniques à cette dernière.

- `freshness` doit être utilisée pour privilégier des résultats en fonction de la nouveauté ou de l'ancienneté d'un élément. Cette fonction peut être utilisée uniquement avec des champs `datetime` (`edm.DateTimeOffset`). Notez que l'attribut `boostingDuration` est utilisé uniquement avec la fonction `freshness`.

- `magnitude` doit être utilisée pour privilégier des résultats en fonction de l'importance ou de la faiblesse d'une valeur numérique. Parmi les scénarios qui appellent cette fonction figurent la valorisation de la marge bénéficiaire, du prix le plus élevé, du prix le plus bas ou du nombre de téléchargements. Cette fonction peut être utilisée uniquement avec des champs `double` et `integer`.

Pour la fonction `magnitude`, vous pouvez inverser la plage (de la valeur la plus élevée à la valeur la plus basse) si vous souhaitez inverser le schéma (par exemple, pour privilégier des éléments dont le prix est plus bas par rapport aux éléments dont le prix est plus élevé). Dans une gamme de prix allant de \$100USD à \$1USD, vous devez définir `boostingRangeStart` sur 100 et `boostingRangeEnd` sur 1 pour privilégier les éléments dont le prix est plus bas.

- `distance` doit être utilisée pour privilégier des résultats en fonction de la proximité ou de l'emplacement géographique. Cette fonction peut être utilisée uniquement avec des champs `Edm.GeographyPoint`.

- `tag` doit être utilisée pour privilégier des résultats en fonction de balises communes entre des documents et des requêtes de recherche. Cette fonction peut être utilisée uniquement avec les champs `Edm.String` et `Collection(Edm.String)`.

## Règles d'utilisation des fonctions

Le type de fonction (`freshness`, `magnitude`, `distance`, `tag`) doit être en lettres minuscules.

Les fonctions ne peuvent pas contenir de valeurs null ou vides. En particulier, si vous incluez la valeur `fieldname`, vous devez la spécifier.

Les fonctions ne peuvent être appliquées qu'à des champs filtrables. Consultez [Créer un index \(API REST de Recherche cognitive Azure\)](#) pour plus d'informations sur les champs filtrables.

Vous ne pouvez pas appliquer de fonctions à des champs définis dans la collection de champs d'un index.

Une fois l'index défini, générez-le en chargeant le schéma d'index, puis des documents. Consultez [Créer un index \(API REST de Recherche cognitive Azure\)](#) et [Ajouter, mettre à jour ou supprimer des documents \(API REST de Recherche cognitive Azure\)](#) pour obtenir des instructions relatives à ces opérations. Une fois l'index généré, vous disposez d'un profil de calcul de score fonctionnel qui opère avec vos données de recherche.

## Modèle

Cette section présente la syntaxe et le modèle de profils de calcul de score. Pour obtenir la description des attributs, consultez [Référence des attributs d'index](#) dans la section suivante.

```
...
"scoringProfiles": [
  {
    "name": "name of scoring profile",
    "text": (optional, only applies to searchable fields) {
      "weights": {
        "searchable_field_name": relative_weight_value (positive #'s),
        ...
      }
    },
    "functions": (optional) [
      {
        "type": "magnitude | freshness | distance | tag",
        "boost": # (positive number used as multiplier for raw score != 1),
        "fieldName": "...",
        "interpolation": "constant | linear (default) | quadratic | logarithmic",

        "magnitude": {
          "boostingRangeStart": #,
          "boostingRangeEnd": #,
          "constantBoostBeyondRange": true | false (default)
        }
      }

      // ( - or -)

      "freshness": {
        "boostingDuration": .... (value representing timespan over which boosting occurs)
      }

      // ( - or -)

      "distance": {
        "referencePointParameter": "...", (parameter to be passed in queries to use as reference
location)
        "boostingDistance": # (the distance in kilometers from the reference location where the boosting
range ends)
      }

      // ( - or -)

      "tag": {
        "tagsParameter": ....(parameter to be passed in queries to specify a list of tags to compare
against target field)
      }
    ],
    "functionAggregation": (optional, applies only when functions are specified) "sum (default) | average
| minimum | maximum | firstMatching"
  }
],
"defaultScoringProfile": (optional) "...",
...
]
```

## Référence des attributs d'index

### NOTE

Vous pouvez appliquer une fonction de calcul de score uniquement à des champs filtrables.

ATTRIBUT	DESCRIPTION
<code>name</code>	Obligatoire. Nom du profil de calcul de score. Il suit les conventions d'affectation de noms applicables aux champs. Il doit commencer par une lettre, ne peut pas contenir les signes point, deux-points ou @, et ne peut pas commencer par l'expression « azureSearch » (avec respect de la casse).
<code>text</code>	Contient la propriété weights.
<code>weights</code>	facultatif. Contient des paires nom-valeur spécifiant chacune un nom de champ et une pondération relative. La pondération relative doit être un entier positif ou un nombre à virgule flottante.  Les pondérations permettent d'indiquer l'importance d'un champ pouvant faire l'objet d'une recherche par rapport à d'autres.
<code>functions</code>	facultatif. Vous pouvez appliquer une fonction de calcul de score uniquement à des champs filtrables.
<code>type</code>	Obligatoire pour les fonctions de calcul de score. Indique le type de fonction à utiliser. Les valeurs autorisées sont magnitude, freshness, distance et tag. Vous pouvez inclure plusieurs fonctions dans chaque profil de calcul de score. Le nom de fonction doit être en lettres minuscules.
<code>boost</code>	Obligatoire pour les fonctions de calcul de score. Nombre positif utilisé comme multiplicateur pour le score brut. Il ne peut pas être égal à 1.
<code>fieldname</code>	Obligatoire pour les fonctions de calcul de score. Vous pouvez appliquer une fonction de calcul de score uniquement à des champs faisant partie de la collection de champs de l'index, et qui sont filtrables. De plus, chaque type de fonction fait l'objet de restrictions supplémentaires (la valeur freshness est utilisée avec des champs datetime, la valeur amplitude avec des champs integer ou double, et la valeur distance avec des champs location). Vous pouvez spécifier un seul champ par définition de fonction. Par exemple, pour utiliser une valeur magnitude deux fois dans le même profil, vous devez inclure deux définitions de magnitude, une pour chaque champ.
<code>interpolation</code>	Obligatoire pour les fonctions de calcul de score. Définit la pente pour laquelle la valorisation du score augmente, du début à la fin de la plage. Les valeurs autorisées sont Linear (par défaut), Constant, Quadratic et Logarithmic. Consultez la rubrique <a href="#">Définition d'interpolations</a> pour plus d'informations.

ATTRIBUT	DESCRIPTION
<pre>magnitude</pre>	<p>La fonction de calcul de score magnitude est utilisée pour modifier des classements en fonction de la plage de valeurs pour un champ numérique. Voici quelques exemples d'utilisation courants :</p> <ul style="list-style-type: none"> <li>- <b>Évaluations</b> : modifiez le calcul de score sur la base de la valeur figurant dans le champ « Évaluation ». Lorsque deux éléments sont pertinents, l'élément associé à l'évaluation la plus élevée s'affiche en premier.</li> <li>- <b>Marge</b> : lorsque deux documents sont pertinents, un détaillant peut souhaiter privilégier le document présentant les marges les plus élevées.</li> <li>- <b>Nombres de clics</b> : pour les applications qui suivent les actions de clic sur des produits ou des pages, vous pouvez utiliser la fonction magnitude pour favoriser les éléments qui drainent le plus de trafic.</li> <li>- <b>Nombres de téléchargements</b> : pour les applications qui suivent les téléchargements, la fonction magnitude permet de favoriser les éléments les plus téléchargés.</li> </ul>
<pre>magnitude   boostingRangeStart</pre>	<p>Définit la valeur de début de la plage sur la base de laquelle les scores de magnitude sont calculés. La valeur doit être un entier ou un nombre à virgule flottante. Pour des évaluations de 1 à 4, il s'agit de 1. Pour des marges de plus de 50 %, il s'agit de 50.</p>
<pre>magnitude   boostingRangeEnd</pre>	<p>Définit la valeur de fin de la plage sur laquelle les scores de magnitude sont calculés. La valeur doit être un entier ou un nombre à virgule flottante. Pour les évaluations de 1 à 4, il s'agit de 4.</p>
<pre>magnitude   constantBoostBeyondRange</pre>	<p>Les valeurs autorisées sont true ou false (par défaut). Quand la valeur est true, la valorisation complète continue de s'appliquer aux documents dont la valeur pour le champ cible est supérieure à la limite supérieure de la plage. Quand la valeur est false, la valorisation de cette fonction ne s'applique pas aux documents dont la valeur pour le champ cible se situe en dehors de la plage.</p>

ATTRIBUT	DESCRIPTION
<code>freshness</code>	<p>La fonction de calcul de score <code>freshness</code> permet de modifier les scores de classement d'éléments en fonction des valeurs des champs <code>DateTimeOffset</code>. Par exemple, un élément dont la date est plus récente peut être classé plus haut que des éléments plus anciens.</p> <p>Il est également possible de classer les éléments tels que les événements de calendrier comportant des dates ultérieures afin que les éléments plus proches de la date du jour soient classés plus haut que les éléments plus éloignés dans le temps.</p> <p>Dans la version de service actuelle, une extrémité de la plage doit être définie sur l'heure réelle. L'autre extrémité est une heure dans le passé selon l'attribut <code>boostingDuration</code>. Pour privilégier une plage d'heures à venir, utilisez un attribut <code>boostingDuration</code> négatif.</p> <p>La vitesse à laquelle la valorisation passe d'une plage maximale à une plage minimale est déterminée par l'interpolation appliquée au profil de calcul de score (voir la figure ci-dessous). Pour inverser le facteur de valorisation appliqué, choisissez un facteur inférieur à 1.</p>
<code>freshness</code>   <code>boostingDuration</code>	Définit une période d'expiration après laquelle la valorisation s'arrête pour un document spécifique. Pour plus d'informations sur la syntaxe et des exemples, consultez <a href="#">Set <code>boostingDuration</code></a> dans la section suivante.
<code>distance</code>	La fonction de calcul de score à distance est utilisée pour affecter le score de documents sur la base de leur proximité ou de l'éloignement par rapport à un emplacement géographique de référence. L'emplacement de référence est indiqué comme partie intégrante de la requête dans un paramètre (à l'aide de l'option de chaîne <code>scoringParameterquery</code> ) en tant qu'argument lon,lat.
<code>distance</code>   <code>referencePointParameter</code>	Paramètre à transmettre dans des requêtes, à utiliser comme emplacement de référence. <code>scoringParameter</code> est un paramètre de requête. Les paramètres de requête sont décrits dans <a href="#">Recherche dans des documents (API REST de Recherche cognitive Azure)</a> .
<code>distance</code>   <code>boostingDistance</code>	Nombre indiquant la distance en kilomètres par rapport à l'emplacement de référence où la valorisation se termine.
<code>tag</code>	La fonction de calcul de score de balises est utilisée pour affecter le score de documents sur la base de balises dans des documents et des requêtes de recherche. Les documents contenant des balises communes avec la requête de recherche seront privilégiés. Les balises pour la requête de recherche sont fournies en tant que paramètre de calcul de score dans chaque requête de recherche (à l'aide de l'option de chaîne <code>scoringParameterquery</code> ).

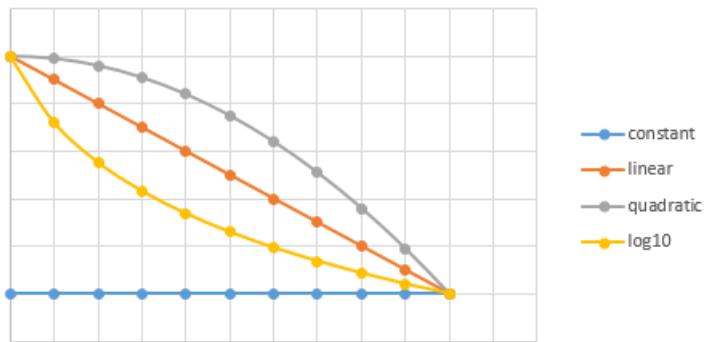
ATTRIBUT	DESCRIPTION
<code>tag</code>   <code>tagsParameter</code>	Paramètre à transmettre dans des requêtes pour spécifier des balises pour une requête spécifique. <code>scoringParameter</code> est un paramètre de requête. Les paramètres de requête sont décrits dans <a href="#">Recherche dans des documents (API REST de Recherche cognitive Azure)</a> .
<code>functionAggregation</code>	facultatif. S'applique uniquement quand des fonctions sont spécifiées. Les valeurs autorisées sont les suivantes : sum (par défaut), average, minimum, maximum et firstMatching. Un score de recherche est une valeur unique calculée à partir de plusieurs variables, notamment plusieurs fonctions. Cet attribut indique comment les valorisations de toutes les fonctions sont combinées en une valorisation agrégée qui est ensuite appliquée au score du document de base. Le score de base dépend de la valeur <code>tf-idf</code> calculée à partir du document et de la requête de recherche.
<code>defaultScoringProfile</code>	Lors de l'exécution d'une requête de recherche, le calcul de score par défaut est utilisé ( <code>tf-idf</code> uniquement) si aucun profil de calcul de score n'est spécifié.  Un nom de profil de calcul de score par défaut peut être défini ici de façon à ce que Recherche cognitive Azure utilise ce profil quand aucun profil spécifique n'est fourni dans la requête de recherche.

## Définition d'interpolations

Les interpolations vous permettent de définir la forme de la pente utilisée pour calculer les scores. Le score allant d'élevé à faible, la pente est toujours décroissante, mais l'interpolation détermine la courbe de la pente descendante. Les interpolations utilisables sont les suivantes :

INTERPOLATION	DESCRIPTION
<code>linear</code>	pour les éléments qui s'inscrivent dans la plage de max à min, la valorisation appliquée décroît de manière constante. L'interpolation de type Linear est l'interpolation par défaut pour un profil de calcul de score.
<code>constant</code>	Pour les éléments qui s'inscrivent dans la plage du début à la fin, une valorisation constante est appliquée aux résultats du classement.
<code>quadratic</code>	Par rapport à une interpolation de type Linear dont la valorisation décroît de façon constante, une interpolation de type Quadratic décroît initialement plus lentement, puis, lorsqu'elle approche de la plage de fin, beaucoup plus rapidement. Cette option d'interpolation n'est pas autorisée dans les fonctions de calcul de score de balises.
<code>logarithmic</code>	Par rapport à une interpolation de type Linear dont la valorisation décroît de façon constante, une interpolation de type Logarithmic décroît initialement plus rapidement, puis, lorsqu'elle approche de la plage de fin, beaucoup plus lentement. Cette option d'interpolation n'est pas autorisée dans les fonctions de calcul de score de balises.

## Scoring Function Interpolation



## Définition de boostingDuration

`boostingDuration` est un attribut de la fonction `freshness`. Il permet de définir une période d'expiration après laquelle la valorisation s'arrête pour un document spécifique. Par exemple, pour valoriser une ligne de produits ou une marque pendant une période promotionnelle de 10 jours, vous spécifiez la période de 10 jours en tant que « P10D » pour les documents correspondants.

`boostingDuration` doit être au format « `dayTimeDuration` » XSD (sous-ensemble limité d'une valeur de durée ISO 8601). Le modèle est le suivant : « `P[nD][T[nH][nM][nS]]` ».

Le tableau suivant fournit plusieurs exemples.

DURATION	BOOSTINGDURATION
1 jour	« P1D »
2 jours et 12 heures	« P2DT12H »
15 minutes	« PT15M »
30 jours, 5 heures 10 minutes, et 6 334 secondes	« P30DT5H10M6.334S »

Pour plus d'exemples, consultez [Schéma XML : types de données \(site web W3.org\)](#).

## Voir aussi

- [Référence d'API REST](#)
- [Créer une API d'index](#)
- [Kit de développement logiciel \(SDK\) de Recherche cognitive Azure .NET](#)

# Algorithme de classement dans la Recherche cognitive Azure

04/10/2020 • 9 minutes to read • [Edit Online](#)

## IMPORTANT

À compter du 15 juillet 2020, les services de recherche nouvellement créés utiliseront automatiquement la fonction de classement BM25, qui s'est avérée efficace, dans la plupart des cas, pour fournir des classements de recherche répondant mieux aux attentes des utilisateurs que le classement par défaut actuel. Au-delà du classement plus élevé, BM25 active également des options de configuration pour les résultats de paramétrage en fonction de facteurs tels que la taille des documents.

Avec ce changement, vous verrez probablement de légères différences dans l'ordre de vos résultats de recherche. Pour ceux qui souhaitent tester l'impact de cette modification, l'algorithme BM25 est disponible dans les versions d'API 2019-05-06-Preview et 2020-06-30.

Cet article explique comment utiliser le nouvel algorithme de classement BM25 sur les services de recherche existants pour les nouveaux index créés et interrogés à l'aide de l'API en préversion.

La Recherche cognitive Azure est en train d'adopter l'implémentation Lucene officielle de l'algorithme Okapi BM25, *BM25Similarity*, qui remplacera l'implémentation de *ClassicSimilarity* précédemment utilisée. Comme l'ancien algorithme *ClassicSimilarity*, *BM25Similarity* est une fonction de récupération semblable à TF-IDF qui utilise la fréquence du terme (TF) et la fréquence de document inverse (IDF) comme variables pour calculer les notes de pertinence pour chaque paire document-requête, qui sont ensuite utilisées pour le classement.

Bien que semblable d'un point de vue conceptuel à l'ancien algorithme de similarité classique, BM25 prend racine dans la récupération d'informations probabilistes pour les améliorer. BM25 propose également des options de personnalisation avancées, comme permettre à l'utilisateur de décider comment la note de pertinence est mise à l'échelle avec la fréquence du terme des termes correspondants.

## Comment tester BM25 aujourd'hui

Quand vous créez un index, vous pouvez définir une propriété **similarity** pour spécifier l'algorithme. Vous pouvez utiliser `api-version=2019-05-06-Preview`, comme indiqué ci-dessous, ou `api-version=2020-06-30`.

```
PUT https://[search service name].search.windows.net/indexes/[index name]?api-version=2019-05-06-Preview
```

```
{
  "name": "indexName",
  "fields": [
    {
      "name": "id",
      "type": "Edm.String",
      "key": true
    },
    {
      "name": "name",
      "type": "Edm.String",
      "searchable": true,
      "analyzer": "en.lucene"
    },
    ...
  ],
  "similarity": {
    "@odata.type": "#Microsoft.Azure.Search.BM25Similarity"
  }
}
```

La propriété **similarity** est utile pendant cette période intermédiaire lorsque les deux algorithmes sont disponibles, uniquement sur les services existants.

PROPRIÉTÉ	DESCRIPTION
similarity	facultatif. Les valeurs valides sont <code>"#Microsoft.Azure.Search.ClassicSimilarity"</code> ou <code>"#Microsoft.Azure.Search.BM25Similarity"</code> . Requiert <code>api-version=2019-05-06-Preview</code> ou version ultérieure sur un service de recherche créé avant le 15 juillet 2020.

Pour les nouveaux services créés après le 15 juillet 2020, BM25 est utilisé automatiquement et est l'unique algorithme de similarité. Si vous essayez de définir **similarity** sur `ClassicSimilarity` pour un nouveau service, une erreur 400 est renvoyée, car cet algorithme n'est pas pris en charge sur un nouveau service.

Pour les services existants créés avant le 15 juillet 2020, la similarité classique demeure l'algorithme par défaut. Si la propriété **similarity** est omise ou a la valeur null, l'index utilise l'algorithme classique. Si vous souhaitez utiliser le nouvel algorithme, vous devez définir **similarity** comme décrit ci-dessus.

## Paramètres de similarité de BM25

La similarité de BM25 ajoute deux paramètres personnalisables par l'utilisateur pour contrôler la note de pertinence calculée.

### k1

Le paramètre *k1* contrôle la fonction de mise à l'échelle entre la fréquence du terme de chaque terme correspondant et la note de pertinence finale d'une paire document-requête.

La valeur zéro représente un « modèle binaire », où la contribution d'un seul terme correspondant est identique pour tous les documents correspondants, quel que soit le nombre de fois que ce terme apparaît dans le texte, tandis qu'une valeur *k1* plus grande permet à la note de continuer à augmenter à mesure que des instances supplémentaires du même terme sont trouvées dans le document. Par défaut, la Recherche cognitive Azure utilise la valeur 1.2 pour le paramètre *k1*. L'utilisation d'une valeur *k1* supérieure peut être importante dans les cas où nous pensons que plusieurs termes peuvent faire partie d'une requête de recherche. Dans ce cas, nous pouvons souhaiter privilégier les documents qui correspondent à un grand nombre des différents termes de requête faisant l'objet d'une recherche plutôt que les documents qui n'ont une correspondance qu'avec un seul terme,

plusieurs fois. Par exemple, lors de l'interrogation de l'index pour rechercher les documents contenant les termes « Apollo Spaceflight », nous pouvons souhaiter diminuer la note d'un article sur la mythologie grecque qui contient le terme « Apollo » plusieurs dizaines de fois mais qui ne mentionne pas « Spaceflight », par rapport à un autre article qui mentionne explicitement à la fois « Apollo » et « Spaceflight » quelques fois seulement.

## b

Le paramètre *b* contrôle la manière dont la longueur d'un document affecte la note de pertinence.

La valeur 0.0 signifie que la longueur du document n'aura pas d'influence sur la note, tandis que la valeur 1.0 signifie que l'impact de la fréquence du terme sur la note de pertinence sera normalisé par la longueur du document. La valeur par défaut utilisée dans la Recherche cognitive Azure pour le paramètre *b* est 0.75. La normalisation de la fréquence du terme par la longueur du document est utile dans les cas où nous voulons pénaliser les documents plus longs. Dans certains cas, des documents plus longs (par exemple, un roman entier) sont davantage susceptibles de contenir de nombreux termes non pertinents, par rapport à des documents beaucoup plus courts.

## Définition des paramètres k1 et b

Pour personnaliser les valeurs *b* ou *k1*, ajoutez-les simplement sous forme de propriétés à l'objet de similarité lors de l'utilisation de BM25 :

```
"similarity": {  
    "@odata.type": "#Microsoft.Azure.Search.BM25Similarity",  
    "b" : 0.5,  
    "k1" : 1.3  
}
```

L'algorithme de similarité ne peut être défini qu'au moment de la création de l'index. Cela signifie que l'algorithme de similarité utilisé ne peut pas être modifié pour les index existants. Les paramètres "*b*" et "*k1*" peuvent être modifiés lors de la mise à jour de la définition d'un index existant qui utilise BM25. Le changement de ces valeurs sur un index existant placera votre index hors connexion pendant au moins quelques secondes, ce qui entraînera l'échec de vos demandes d'indexation et de requête. C'est pourquoi vous devrez définir le paramètre « allowIndexDowntime=true » dans la chaîne de requête de votre demande de mise à jour :

```
PUT https://[search service name].search.windows.net/indexes/[index name]?api-version=[api-version]&allowIndexDowntime=true
```

## Voir aussi

- [Référence d'API REST](#)
- [Ajouter des profils de scoring à votre index](#)
- [Créer une API d'index](#)
- [Kit de développement logiciel \(SDK\) de Recherche cognitive Azure .NET](#)

# Choisir un niveau tarifaire pour Recherche cognitive Azure

04/10/2020 • 34 minutes to read • [Edit Online](#)

Lorsque vous créez un service Recherche cognitive Azure, une [ressource est créée](#) à un niveau tarifaire ou une référence SKU qui sont fixes pendant toute la durée de vie du service. Les niveaux disponibles sont : Gratuit, De base, Standard et À stockage optimisé. Les niveaux Standard et À stockage optimisé sont proposés dans diverses configurations et capacités.

La plupart des clients commencent par le niveau Gratuit qui leur permet d'évaluer le service. Une fois l'évaluation terminée, il est courant de créer un deuxième service à l'un des niveaux supérieurs pour les déploiements de développement et de production.

## Disponibilité des fonctionnalités par niveau

Le tableau suivant décrit les contraintes de fonctionnalité liées aux niveaux.

FONCTIONNALITÉ	LIMITES
indexeurs	Les indexeurs ne sont pas disponibles sur S3 HD.
Enrichissement par IA	Fonctionne au niveau Gratuit, mais n'est pas recommandé.
Clés de chiffrement gérées par le client	Non disponibles au niveau Gratuit.
Accès au pare-feu IP	Non disponibles au niveau Gratuit.
Intégration à Azure Private Link	Non disponibles au niveau Gratuit.

La plupart des fonctionnalités sont disponibles à tous les niveaux, notamment au niveau Gratuit, mais les fonctionnalités gourmandes en ressources peuvent ne pas fonctionner correctement si vous ne leur accordez pas les capacités suffisantes. Par exemple, [l'enrichissement par IA](#) implique des qualifications à long terme qui dépassent le délai d'attente sur un service Gratuit, sauf si le jeu de données est restreint.

## Niveaux (références SKU)

Les niveaux se différencient par :

- Quantité d'index et d'indexeurs que vous pouvez créer
- Taille et la vitesse des partitions (stockage physique)

Le niveau que vous sélectionnez détermine le taux facturable. La capture d'écran suivante du portail Azure indique les niveaux disponibles, après la déduction indiquée sur le portail et dans la [page de tarification](#). Les niveaux les plus courants sont **Gratuit**, **De base** et **Standard**.

Le niveau **Gratuit** crée un service de recherche limité pour les projets plus petits, notamment les guides de démarrage rapides et les tutoriels. En interne, les répliques et les partitions sont partagées entre plusieurs abonnés. Vous ne pouvez pas mettre à l'échelle un service gratuit ni exécuter des charges de travail importantes.

Les niveaux **De base** et **Standard** sont les niveaux facturables les plus utilisés, **Standard** étant le niveau par

défaut. Grâce à des ressources dédiées sous votre contrôle, vous pouvez déployer des projets plus volumineux, optimiser les performances et définir la capacité.

Select Pricing Tier								
Browse available skus and their features								
Sku	Offering	Indexes	Indexers	Storage	Search units	Replicas	Partitions	
F	Free	3	3	50 MB	1	1	1	
B	Basic	15	15	2 GB	3	3	1	
S	Standard	50	50	25 GB/Partition*	36	12	12	
S2	Standard	200	200	100 GB/Partition*	36	12	12	
S3	Standard	200	200	200 GB/Partition*	36	12	12	
S3HD	High-density	1000	0	200 GB/Partition*	36	12	3	
L1	Storage Optimized	10	10	1 TB/Partition*	36	12	12	
L2	Storage Optimized	10	10	2 TB/Partition*	36	12	12	

Certains niveaux sont optimisés pour certains types de travaux. Par exemple, le niveau **Standard 3 High Density (S3 HD)** est un *mode d'hébergement* pour S3, où le matériel sous-jacent est optimisé pour un grand nombre d'index plus petits, qui est destiné aux scénarios d'architecture mutualisée. Le niveau S3 HD présente les mêmes frais à l'unité que S3, mais le matériel est optimisé pour les lectures de fichiers rapides sur un grand nombre d'index plus petits.

Les niveaux **À stockage optimisé** offrent une capacité de stockage plus importante et à moindre coût par To que les niveaux Standard. Le principal compromis impliqué par ces niveaux réside dans une latence de requête plus élevée, ce que vous devez valider pour vos exigences applicatives spécifiques. Pour plus d'informations sur les considérations en matière de performances de ce niveau, consultez l'article [Considérations sur les performances et l'optimisation](#).

Des informations complémentaires sur les différents niveaux sont disponibles sur la [page de tarification](#), dans l'article [Service limits in Azure Search](#) (Limites du service Recherche cognitive Azure), ainsi que sur la page du portail lorsque vous approvisionnez un service.

## Événements facturables

Une solution reposant sur Recherche cognitive Azure peut occasionner des coûts de l'une des manières suivantes :

- Coût du service proprement dit, exécuté 24 h sur 24 et 7 j sur 7 avec une configuration minimale (une partition et un réplica)
- Ajout de capacité (réplicas ou partitions)
- Frais de bande passante (transfert de données sortant)
- Services complémentaires requis pour des capacités ou fonctionnalités spécifiques :
  - enrichissement par IA (nécessite [Cognitive Services](#))
  - base de connaissances (nécessite [Stockage Azure](#))
  - enrichissement incrémentiel (nécessite [Stockage Azure](#), s'applique à l'enrichissement par IA)
  - clés gérées par le client et double chiffrement (nécessite [Azure Key Vault](#))
  - points de terminaison privés pour un modèle sans accès à Internet (nécessite [Azure Private Link](#))

### Coûts de service

Contrairement à des machines virtuelles ou à d'autres ressources qui peuvent être « mises en pause » pour éviter des frais, un service Recherche cognitive Azure est toujours disponible sur du matériel dédié à votre usage

exclusif. En tant que telle, la création d'un service est un événement facturable qui commence lorsque vous créez le service et se termine lorsque vous le supprimez.

Le coût minimal est la première unité de recherche (une réplique x une partition) au tarif facturable. Ces frais minimaux sont fixes pendant toute la durée de vie du service, car il est impossible d'exécuter le service sur une configuration inférieure à cette dernière. Au-delà des frais minimaux, vous pouvez ajouter des réplicas et des partitions indépendants les uns des autres. Les augmentations incrémentielles de capacité via des réplicas et partitions augmentent votre facture selon la formule suivante : ([réplicas x partitions x tarif](#)) , où le tarif facturé dépendant du niveau de tarification sélectionné.

Lorsque vous estimatez le coût d'une solution de recherche, gardez à l'esprit que la tarification et la capacité ne sont pas linéaires. (Par exemple, la multiplication par deux de la capacité coûte plus du double.) Pour découvrir un exemple du mode de fonctionnement de la formule, consultez la section [How to allocate replicas and partitions](#) (Allocation de réplicas et de partitions).

### Frais liés à la bande passante

L'utilisation [d'indexeurs](#) peut avoir une incidence sur la facturation, selon l'emplacement de vos services. Vous pouvez éliminer totalement les frais de sortie de données si vous créez le service Recherche cognitive Azure dans la même région que vos données. Voici quelques informations issues de la [page de tarification de la bande passante](#) :

- Microsoft ne facture aucune donnée entrante dans un quelconque service Azure, ni aucune donnée sortante du service Recherche cognitive Azure.
- Dans les solutions multiservice, il n'existe aucun frais pour les données qui transitent par le réseau lorsque tous les services se trouvent dans la même région.

Des frais s'appliquent pour les données sortantes si les services sont situés dans des régions différentes. Ces frais ne figurent pas sur votre facture Recherche cognitive Azure. Ils sont mentionnés ici, car si vous utilisez des données ou des indexeurs IA pour extraire les données de différentes régions, les coûts associés apparaîtront sur votre facture globale.

### Enrichissement de l'IA avec Cognitive Services

Pour l'enrichissement de l'IA, prévoyez d'[attacher une ressource Azure Cognitive Services facturable](#) dans la même région que le service Recherche cognitive Azure, au niveau tarifaire S0 pour le traitement du paiement à l'utilisation. L'attachement de Cognitive Services n'entraîne aucun coût fixe. Vous payez uniquement pour le traitement dont vous avez besoin.

OPÉRATION	IMPACT SUR LA FACTURATION
Craquage de document, extraction de texte	Gratuit
Craquage de document, extraction d'image	Facturée en fonction du nombre d'images extraites de vos documents. Dans une <a href="#">configuration d'indexeur</a> , <b>imageAction</b> est le paramètre qui déclenche l'extraction d'images. Si <b>imageAction</b> est défini sur « none » (valeur par défaut), l'extraction d'images ne vous sera pas facturée. Le tarif de l'extraction d'image est mentionné sur la page des <a href="#">détails de tarification</a> du service Recherche cognitive Azure.
Compétences cognitives prédefinies	Facturées au même tarif que si vous aviez exécuté la tâche directement avec Cognitive Services.
Compétences personnalisées	Une compétence personnalisée est une fonctionnalité que vous fournissez. Le coût d'utilisation d'une compétence personnalisée dépend entièrement du fait qu'un code personnalisé appelle d'autres services mesurés.

La fonctionnalité [d'enrichissement incrémentiel](#) (préversion) permet de fournir un cache qui augmente l'efficacité de l'indexeur en exécutant uniquement les compétences cognitives nécessaires en cas de modification ultérieure de l'ensemble de compétences, ce qui représente un gain de temps et d'argent.

## Formule de facturation ( $R \times P = SU$ )

La notion de facturation la plus importante à comprendre pour les opérations du service Recherche cognitive Azure est *l'Unité Recherche* (SU, Search Unit). Comme Recherche cognitive Azure dépend à la fois des réplicas et des partitions pour l'indexation et les requêtes, facturer uniquement par rapport à l'un ou à l'autre n'aurait aucun sens. Au lieu de cela, la facturation est basée sur un composite de ces deux facteurs.

L'Unité Recherche est le produit des *réplicas* et des *partitions* utilisés par un service : ( $R \times P = SU$ ) .

Chaque service commence avec une Unité Recherche (un réplica multiplié par une partition) au minimum. Le montant maximal de tout service est de 36 SU. Ce montant maximal peut être atteint de plusieurs façons : 6 partitions x 6 réplicas, ou 3 partitions x 12 réplicas, par exemple. La capacité utilisée est généralement inférieure à la capacité totale (par exemple, un service de 3 réplicas et 3 partitions facturé avec un montant de 9 SU). Pour connaître les combinaisons valides, consultez l'article [Partition and replica combinations](#) (Combinaisons de partitions et de réplicas).

Le tarif de facturation est un tarif horaire par SU. Le tarif augmente progressivement à chaque niveau. Les niveaux supérieurs proposent des partitions plus volumineuses et plus rapides, ce qui entraîne un tarif horaire globalement plus élevé pour ces niveaux. Vous pouvez consulter les tarifs de chaque niveau sur la page des [détails de tarification](#).

La plupart des clients mettent seulement une partie de la capacité totale en ligne et gardent le reste en réserve. Pour la facturation, le nombre de partitions et de réplicas que vous mettez en ligne, calculé à l'aide de la formule SU, détermine le tarif horaire qui vous est facturé.

## Gestion des coûts

Les suggestions suivantes peuvent vous aider à réduire les coûts ou à les gérer plus efficacement :

- Créez toutes les ressources dans la même région ou dans le moins de régions possible afin de réduire ou d'éliminer les coûts liés à la bande passante.
- Regroupez tous les services dans un seul groupe de ressources, tel que Recherche cognitive Azure, Cognitive Services et tout autre service Azure utilisé dans votre solution. Dans le portail Azure, recherchez le groupe de ressources et utilisez les commandes **Cost Management** pour obtenir des informations sur les dépenses réelles et prévues.
- Envisagez d'utiliser Application web Azure pour votre application frontale afin que les demandes et réponses restent dans les limites du centre de données.
- Montez en puissance pour les opérations gourmandes en ressources, telles que l'indexation, puis réajustez à la baisse les charges de travail de requête régulières. Commencez avec la configuration minimale pour Recherche cognitive Azure (une unité de stockage composée d'une partition et un réplica), puis surveillez l'activité de l'utilisateur pour identifier des modèles d'utilisation qui indiqueront un besoin de capacité supplémentaires. Si un modèle est prévisible, vous pouvez peut-être synchroniser l'échelle avec l'activité (vous devez écrire du code pour automatiser ce comportement).

Consultez également la page [Facturation et gestion des coûts](#) pour en savoir plus sur les outils et fonctionnalités intégrés liés aux dépenses.

L'arrêt temporaire d'un service de recherche est impossible. Les ressources dédiées sont toujours opérationnelles et sont allouées pour votre usage exclusif pendant pour la durée de vie de votre service. La suppression d'un service est définitive, et elle entraîne également la suppression des données associées à celui-

ci.

En ce qui concerne le service lui-même, le seul moyen de réduire votre facture consiste à réduire le nombre de réplicas et de partitions à un niveau offrant garantissant encore des performances acceptables et la [conformité au SLA](#), ou à créer un service de niveau inférieur (les taux horaires du niveau S1 sont inférieurs à ceux des niveaux S2 ou S3). En supposant que vous approvisionniez votre service à la limite inférieure de vos prévisions de charge, si vous dépassiez le service, vous pouvez créer un second service de niveau supérieur, régénérer vos index sur ce second service, puis supprimer le premier.

## Comment évaluer les besoins en capacité

Dans Recherche cognitive Azure, la capacité est structurée sous forme de *réplicas* et de *partitions*.

- Les réplicas sont des instances du service de recherche. Chaque réplica héberge une copie à charge équilibrée d'un index. Par exemple, un service avec six réplicas comporte six copies de chaque index chargé dans le service.
- Les partitions stockent les index et fractionnent automatiquement les données utilisables dans une requête. Deux partitions fractionnent votre index en deux, trois partitions le scindent en trois, et ainsi de suite. En termes de capacité, la *taille de partition* est la principale caractéristique qui différencie les niveaux.

### NOTE

Tous les niveaux Standard et À stockage optimisé prennent en charge des [combinaisons flexibles de réplicas et de partitions](#) afin que vous puissiez [optimiser votre système sur le plan du stockage ou de la vitesse](#) en changeant l'équilibrage. Le niveau De base offre jusqu'à trois réplicas pour la haute disponibilité, mais ne comporte qu'une seule partition. Les niveaux Gratuit ne fournissent pas de ressources dédiées : les ressources de calcul sont partagées par plusieurs abonnés.

### Évaluation de la capacité

La capacité et les coûts d'exécution du service vont de pair. Les niveaux imposent des limites sur deux plans : stockage et ressources. Vous devez considérer les deux à la fois, car la limite que vous atteignez en premier représente la limite effective.

Ce sont généralement les exigences métier qui imposent le nombre d'index dont vous aurez besoin. Par exemple, vous devrez peut-être disposer d'un index global pour un référentiel de documents volumineux. Ou vous pourrez avoir besoin de plusieurs index basés sur la région, l'application ou le créneau commercial.

Pour déterminer la taille d'un index, vous devez en [créer un](#). Sa taille repose sur les données importées et la configuration de l'index, notamment si vous activez des suggesteurs, un filtrage et un tri.

Pour la recherche en texte intégral, la structure de données principale constitue une structure d'[index inversé](#), dont les caractéristiques diffèrent de celles des données sources. Dans le cas d'un index inversé, la taille et la complexité sont déterminées par le contenu, et non nécessairement par la quantité de données qui l'alimentent. Une source de données volumineuse avec un haut niveau de redondance peut générer un index plus restreint qu'un jeu de données plus modeste présentant un contenu extrêmement variable. Il est donc généralement impossible de déduire la taille de l'index d'après celle du jeu de données d'origine.

### NOTE

Même si l'estimation des besoins futurs en matière d'index et de stockage semble très approximative, elle en vaut la peine. Si la capacité d'un niveau se révèle insuffisante, vous devrez approvisionner un nouveau service à un niveau supérieur, puis [recharger vos index](#). Un service ne peut faire l'objet d'aucune mise à niveau sur place d'une référence SKU vers une autre.

## **Estimer avec le niveau gratuit**

Une méthode possible pour l'estimation de la capacité consiste à commencer par utiliser le niveau Gratuit. Souvenez-vous que le niveau Gratuit offre jusqu'à 3 index, 50 Mo de stockage et 2 minutes de temps d'indexation. Il peut être difficile d'estimer la taille projetée de l'index avec ces contraintes, mais voici comment procéder :

- [Créez un service gratuit.](#)
- Préparez un petit jeu de données représentatif.
- [Générez un index initial sur le portail](#) et notez sa taille. Les fonctionnalités et attributs ont une incidence sur le stockage. Par exemple, l'ajout de suggesteurs (requêtes en cours de frappe) augmente les besoins en stockage. À l'aide du même jeu de données, vous pouvez tenter de créer plusieurs versions d'un index, avec des attributs différents sur chaque champ, pour voir comment les besoins de stockage varient. Pour plus d'informations, voir « [Implications au niveau du stockage](#) » dans [Créer un index de base](#).

Si vous disposez d'une estimation approximative, vous pouvez doubler cette quantité pour budgérer deux index (développement et production), puis choisir votre niveau en conséquence.

## **Estimer avec un niveau facturable**

Des ressources dédiées peuvent prendre en charge un échantillonnage et des temps de traitement plus conséquents pour produire des estimations plus réalistes de quantité d'index, de taille et de volume de requête durant le développement. Certains clients démarrent d'emblée avec un niveau facturable, puis réévaluent celui-ci à mesure que le projet de développement murit.

1. [Passez en revue les limites de service de chaque niveau](#) afin de déterminer si les niveaux inférieurs peuvent prendre en charge le nombre d'index dont vous avez besoin. Pour les niveaux De base, S1 et S2, les limites d'index sont respectivement de 15, 50 et 200. Le niveau À stockage optimisé présente une limite de 10 index, car il a été conçu pour prendre en charge un petit nombre d'index très volumineux.
2. [Créez un service à un niveau facturable](#) :
  - Si vous n'êtes pas sûr de la charge projetée, commencez modestement, au niveau De base ou S1.
  - En revanche, si vous savez que vous devrez traiter des charges de requêtes et d'indexation à grande échelle, choisissez dès le départ un niveau élevé, S2 ou même S3.
  - Enfin, si vous indexez une grande quantité de données et que la charge de requêtes est relativement faible, comme dans le cas d'une application métier interne, commencez par le niveau À stockage optimisé L1 ou L2.
3. [Générez un index initial](#) pour déterminer comment les données sources se traduisent par un index. C'est l'unique façon d'estimer la taille de l'index.
4. [Surveillez le stockage, les limites de service, le volume de requêtes et la latence](#) dans le portail. Le portail vous indique le nombre de requêtes par seconde, les requêtes limitées et la latence de recherche. Toutes ces valeurs peuvent vous aider à décider si vous avez sélectionné le niveau adéquat.

Le nombre d'index et la taille se révèlent tout aussi importants pour votre analyse. En effet, les limites maximales sont atteintes en cas d'utilisation totale du stockage (partitions) ou des limites maximales relatives aux ressources (index, indexeurs et ainsi de suite), selon ce qui se produit en premier. Le portail vous aide à effectuer le suivi de ces deux facteurs en affichant l'utilisation actuelle et les limites maximales côté à côté dans la page de présentation.

#### NOTE

La présence de données superflues dans les documents peut entraîner une augmentation des exigences de stockage. Dans l'idéal, les documents contiennent uniquement les données dont vous avez besoin pour l'expérience de recherche. Les données binaires ne sont pas utilisables dans une requête et doivent être stockées séparément (par exemple dans un stockage Table Azure ou Blob Azure). Un champ doit ensuite être ajouté dans l'index pour conserver une référence URL aux données externes. La taille maximale d'un document individuel est de 16 Mo (ou moins si vous chargez en bloc plusieurs documents dans une seule requête). Pour plus d'informations, consultez [Limites de service dans Recherche cognitive Azure](#).

## Considérations relatives au volume de requêtes

Le nombre de requêtes par seconde est une métrique importante lors du réglage des performances, mais il ne doit généralement être pris en compte pour le choix du niveau que si vous prévoyez un volume de requêtes élevé dès le départ.

Les niveaux Standard peuvent assurer un équilibre entre le nombre de réplicas et le nombre de partitions. Vous pouvez accélérer l'exécution des requêtes en ajoutant des réplicas pour l'équilibrage de charge ou ajouter des partitions à des fins de traitement parallèle. Vous pouvez ensuite régler les performances une fois le service approvisionné.

Si vous prévoyez des volumes de requêtes élevés et soutenus dès le début, vous devez envisager d'adopter des niveaux Standard plus élevés, qui s'appuient sur du matériel plus puissant. Si ces volumes de requêtes ne surviennent pas, vous pouvez alors mettre des partitions et des réplicas hors connexion, ou même passer à un service de niveau inférieur. Pour plus d'informations sur la façon de calculer le débit de requête, consultez [Performances et optimisation de Recherche cognitive Azure](#).

Les niveaux À stockage optimisé sont utiles pour les charges de travail de données intensives, car ils prennent en charge un stockage d'index total disponible plus important dans les cas où les exigences en matière de latence des requêtes sont moins essentielles. Vous devez toujours utiliser des réplicas supplémentaires pour l'équilibrage de charge, ainsi que des partitions supplémentaires à des fins de traitement parallèle. Vous pouvez ensuite régler les performances une fois le service approvisionné.

## Contrats de niveau de service

Le niveau Gratuit et les fonctionnalités d'évaluation ne fournissent pas de [Contrats de niveau de service \(SLA\)](#).

Pour tous les niveaux facturables, les SLA entrent en vigueur lorsque vous approvisionnez une redondance suffisante pour votre service. Vous devez disposer d'au moins deux réplicas pour les SLA de requête (lecture). Vous devez posséder au moins trois réplicas pour les SLA de requête et d'indexation (lecture-écriture). Le nombre de partitions n'affecte pas les SLA.

## Conseils pour l'évaluation du niveau

- Autorisez la génération de métriques autour des requêtes et recueillez des données relatives aux modèles d'utilisation (requêtes pendant les heures de bureau, indexation pendant les heures creuses). Utilisez ces données pour faciliter les décisions d'approvisionnement de service. Bien que cela ne soit pas pratique à une cadence horaire ou quotidienne, vous pouvez ajuster les partitions et les ressources de manière dynamique afin de prendre en charge les changements de volumes de requêtes planifiés. Vous pouvez également gérer les changements non planifiés mais soutenus si les niveaux se maintiennent suffisamment longtemps pour que des mesures s'imposent.
- N'oubliez pas que le seul inconvénient du sous-approvisionnement réside dans le fait que vous devrez peut-être supprimer un service si les exigences réelles sont supérieures à vos prévisions. Pour éviter toute interruption de service, vous devrez créer un service à un niveau supérieur et l'exécuter côté à côté jusqu'à ce que toutes les applications et requêtes ciblent le nouveau point de terminaison.

## Étapes suivantes

Commencez avec un niveau Gratuit et créez un index initial à l'aide d'un sous-ensemble de vos données afin de bien comprendre ses caractéristiques. La structure de données du service Recherche cognitive Azure est une structure d'index inversé. Le contenu détermine la taille et la complexité d'un index inversé. Souvenez-vous que le contenu hautement redondant a tendance à générer un index plus petit que le contenu très irrégulier. Par conséquent, les exigences en matière de stockage d'index sont déterminées par les caractéristiques du contenu, et non par la taille du jeu de données.

Après avoir effectué une estimation initiale de la taille de votre index, [approvisionnez un service facturable](#) sur l'un des niveaux présentés dans cet article : De base, Standard ou À stockage optimisé. Assouplissez les contraintes artificielles sur le dimensionnement des données et [régénérez votre index](#) afin d'y inclure toutes les données que vous souhaitez rendre utilisables dans une requête.

[Allouez des partitions et des réplicas](#) en fonction des besoins, afin d'obtenir les performances et l'échelle requises.

Si les performances et la capacité conviennent, vous avez terminé. Dans le cas contraire, recréez un service de recherche à un autre niveau correspondant mieux à vos besoins.

### NOTE

Si vous avez des questions, publiez un message sur le site [StackOverflow](#) ou [contactez le support Azure](#).

# Limites de service de la Recherche cognitive Azure

04/10/2020 • 24 minutes to read • [Edit Online](#)

Les limites maximales du stockage, des charges de travail et des quantités d'index et autres objets varient selon que vous [approvisionnez le service Recherche cognitive Azure](#) avec les niveaux tarifaires **Gratuit**, **De base**, **Standard** ou **Stockage optimisé**.

- **Gratuit** est un service partagé multi-locataire qui est fourni avec votre abonnement Azure.
- **De base** : fournit des ressources de calcul dédiées pour des charges de travail de production à plus petite échelle, mais partage une infrastructure réseau avec d'autres locataires.
- Le niveau **Standard** est exécuté sur des ordinateurs dédiés, avec une capacité de stockage et de traitement beaucoup plus grande, et ce, à chaque niveau. Le niveau Standard se décompose en quatre catégories : S1, S2, S3 et S3 HD. La catégorie S3 HD (S3 High Density) est conçue pour des [utilisateurs multiples](#) et de grandes quantités de petits index (trois mille index par service). S3 HD ne fournit pas la [fonctionnalité d'indexeur](#) et l'ingestion des données doit tirer parti des API qui envoient (push) les données de la source vers l'index.
- **Stockage optimisé** s'exécute sur des ordinateurs dédiés avec plus de stockage total, de bande passante de stockage et de mémoire que **Standard**. Ce niveau cible les index volumineux et à variation lente. Stockage optimisé est disponible en deux niveaux : L1 et L2.

## Limites d'abonnement

Vous pouvez créer plusieurs services au sein d'un abonnement. Chacun peut être approvisionné à un niveau spécifique. Vous êtes uniquement limité par le nombre de services autorisé à chaque niveau. Ainsi, vous pouvez créer jusqu'à 12 services au niveau de base et 12 autres au niveau S1 au sein du même abonnement. Pour en savoir plus sur ces niveaux, consultez [Choisir un niveau tarifaire pour Recherche cognitive Azure](#).

Les limites de service maximales peuvent être augmentées sur demande. Si vous faut davantage de services dans le même abonnement, contactez le Support Azure.

RESSOURCE	GRATUIT <sup>1</sup>	DE BASE	S1	S2	S3	S3 HD	L1	L2
Nombre de services maximum	1	16	16	8	6	6	6	6
Nombre maximal d'unités de recherche scale-in (SU) <sup>2</sup>	N/A	3 unités de recherche	36 unités de recherche					

<sup>1</sup> Le niveau Gratuit est basé sur des ressources partagées, non des ressources dédiées. Le « scale up » n'est pas pris en charge sur les ressources partagées.

<sup>2</sup> Les unités de recherche sont des unités de facturation, allouées en tant que *réplicas* ou *partitions*. Vous avez besoin des deux types de ressource pour les opérations de stockage, d'indexation et d'interrogation. Pour en savoir plus sur les calculs SU, consultez [Mettre à l'échelle les niveaux de ressources pour les charges de travail de requête et d'index](#).

## Limites de stockage

Un service de recherche est limité par l'espace disque ou le nombre maximum d'index ou d'indexeurs, selon la limite atteinte en premier. Le tableau suivant décrit les limites de stockage. Pour connaître le nombre maximal d'objets, examinez les [limites par ressource](#).

RESSOURCE	GRATUIT	DE BASE <sup>1</sup>	S1	S2	S3	S3 HD	L1	L2
Contrat de Niveau de Service (SLA) <sup>2</sup>	Non	Oui	Oui	Oui	Oui	Oui	Oui	Oui
Stockage par partition	50 Mo	2 Go	25 Go	100 Go	200 Go	200 Go	1 To	2 To
Partitions par service	N/A	1	12	12	12	3	12	12
Taille de partition	N/A	2 Go	25 Go	100 Go	200 Go	200 Go	1 To	2 To
Réplicas	N/A	3	12	12	12	12	12	12

<sup>1</sup> Le niveau De base comporte une partition fixe. Vous pouvez utiliser des unités de recherche supplémentaires pour ajouter des réplicas en vue d'exécuter des requêtes plus volumineuses.

<sup>2</sup> Des contrats de niveau de service (SLA) sont en vigueur pour les services facturables sur les ressources dédiées. Les services gratuits et les fonctionnalités en préversion n'en ont pas. En ce qui concerne les services facturables, les contrats SLA entrent en vigueur dès lors que vous provisionnez une redondance suffisante pour votre service. Un SLA de requête (lecture) nécessite au moins deux réplicas. Un SLA de requête et d'indexation (lecture-écriture) nécessite au moins trois réplicas. Le nombre de partitions n'est pas pris en compte dans les SLA.

## Limites d'index

RESSOURCE	GRATUIT	DE BASE <sup>1</sup>	S1	S2	S3	S3 HD	L1	L2
Nombre maximal d'index	3	5 ou 15	50	200	200	1 000 par partition ou 3 000 par service	10	10



<sup>1</sup> Les services de base créés avant décembre 2017 présentent des limites inférieures (5 au lieu de 15) sur les index. Le niveau De base est la seule référence soumise à une limite inférieure de 100 champs par index.

<sup>2</sup> Le fait d'avoir un très grand nombre d'éléments dans des collections complexes par document entraîne actuellement une utilisation élevée du stockage. Il s'agit d'un problème connu. En attendant, une limite de 3 000 est une limite supérieure sûre pour tous les niveaux de service. Cette limite est appliquée seulement aux opérations d'indexation qui utilisent la version de l'API en disponibilité générale la plus ancienne qui prend en charge les champs de type complexe (`2019-05-06`). Pour que les clients qui utilisent des versions antérieures de l'API en préversion (qui prennent en charge les champs de type complexe) continuent de fonctionner, nous n'appliquons pas cette limite pour les opérations d'indexation qui utilisent ces versions de l'API en préversion. Notez que les versions de l'API en préversion ne sont pas destinées à être utilisées pour des scénarios de production et que nous recommandons vivement aux clients de passer à la dernière version de l'API en disponibilité générale.

## Limites du document

À compter d'octobre 2018, il n'y a plus de limites du nombre de documents pour tout nouveau service créé, quel que soit le niveau facturable (De base, S1, S2, S3, S3 HD) et la région. Les services plus anciens créés avant octobre 2018 peuvent toujours être soumis à des limites de nombre de documents.

Pour déterminer si des limites du document sont définies pour votre service, utilisez l'[API REST GET Service Statistics](#). Les limites du document figurent dans la réponse, `null` indiquant aucune limite.

### NOTE

Bien qu'il n'y ait aucune limite de document imposée par le service, il existe une limite de partition d'environ 24 milliards de documents par index sur les services de recherche de base, S1, S2 et S3. Pour S3 HD, la limite de partition est de 2 milliards de documents par index. Chaque élément d'une collection complexe compte comme un document distinct en termes de limites de partition.

### Limites de taille de document par appel d'API

La taille maximale d'un document lors de l'appel d'une API d'index est d'environ 16 mégaoctets.

La taille du document est en fait une limite de taille du corps de requête de l'API d'index. Étant donné que vous pouvez transmettre en une seule fois un lot de plusieurs documents à l'API d'index, la limite de taille dépend en réalité du nombre de documents présents dans le lot. Pour un lot comprenant un seul document, la taille maximale du document est de 16 Mo de JSON.

Lorsque vous estimatez la taille du document, n'oubliez pas de prendre en compte uniquement les champs qui peuvent être utilisés par un service de recherche. Toutes les données binaires ou d'image des documents sources doivent être omises de vos calculs.

## Limites de l'indexeur

Les durées d'exécution maximales existent pour fournir équilibre et stabilité au service dans son ensemble, mais l'indexation des jeux de données volumineux peut prendre plus de temps que la valeur maximale ne le permet. Si un travail d'indexation ne peut pas être terminé dans le délai maximal autorisé, essayez de l'exécuter selon une planification. Le planificateur effectue le suivi de l'état de l'indexation. Si une tâche d'indexation planifiée est interrompue pour une raison quelconque, à la prochaine exécution planifiée, l'indexeur peut repartir de là où il s'était arrêté.

RESSOURCE	GRATUIT <sup>1</sup>	DE BASE <sup>2</sup>	S1	S2	S3	S3 HD <sup>3</sup>	L1	L2
Nombre maximal d'indexeurs	3	5 ou 15	50	200	200	N/A	10	10
Nombre maximal de sources de données	3	5 ou 15	50	200	200	N/A	10	10
Compétences maximales <sup>4</sup>	3	5 ou 15	50	200	200	N/A	10	10
Charge d'indexation maximale par appel	10 000 documents	Limité uniquement par le nombre maximal de documents	Limité uniquement par le nombre maximal de documents	Limité uniquement par le nombre maximal de documents	Limité uniquement par le nombre maximal de documents	N/A	Aucune limite	Aucune limite
Planification minimale	5 minutes	5 minutes	5 minutes	5 minutes	5 minutes	5 minutes	5 minutes	5 minutes
Durée maximale d'exécution	1-3 minutes	24 heures	24 heures	24 heures	24 heures	N/A	24 heures	24 heures
Durée d'exécution maximale pour les indexeurs avec un ensemble de compétences <sup>5</sup>	3-10 minutes	2 heures	2 heures	2 heures	2 heures	N/A	2 heures	2 heures
Indexeur d'objets blob : taille maximale des objets blob, en Mo	16	16	128	256	256	N/A	256	256

RESSOURCE	GRATUIT	DE BASE	S1	S2	S3	S3 HD	L1	L2
Indexeur d'objets blob : nombre maximal de caractères du contenu extrait d'un objet blob	32 000	64 000	4 millions	8 millions	16 millions	N/A	4 millions	4 millions

<sup>1</sup> Les services du niveau Gratuit bénéficient d'une durée d'exécution maximale de l'indexeur de 3 minutes pour les sources d'objets blob, et de 1 minute pour toutes les autres sources de données. Pour l'indexation de l'intelligence artificielle qui appelle la méthode dans Cognitive Services, les services gratuits sont limités à 20 transactions gratuites par jour, une transaction étant définie comme un document qui traverse le pipeline d'enrichissement.

<sup>2</sup> Les services de base créés avant décembre 2017 présentent des limites inférieures (5 au lieu de 15) sur les index, les sources de données et les ensembles de compétences.

<sup>3</sup> Les services S3 HD ne comprennent pas de prise en charge de l'indexeur.

<sup>4</sup> Nombre maximal de 30 compétences par group de compétences.

<sup>5</sup> L'enrichissement par IA et l'analyse d'images sont gourmands en ressources et consomment une quantité disproportionnée de la puissance de traitement disponible. Le temps d'exécution de ces charges de travail a été réduit pour permettre l'exécution d'autres tâches dans la file d'attente.

#### NOTE

Comme indiqué dans les [limites des index](#), les indexeurs appliquent également la limite supérieure de 3 000 éléments à toutes les collections complexes par document, en commençant par la dernière version de l'API en disponibilité générale qui prend en charge les types complexes ([2019-05-06](#)). Cela signifie que si vous avez créé votre indexeur avec une version antérieure de l'API, vous ne serez pas soumis à cette limite. Pour préserver une compatibilité maximale, un indexeur qui a été créé avec une version antérieure de l'API, puis mis à jour avec une version de l'API [2019-05-06](#) ou ultérieure, sera toujours exclu des limites. Les clients doivent être conscients de l'impact négatif dans le cas de collections complexes très grandes (comme indiqué précédemment) et nous recommandons vivement de créer les indexeurs avec la dernière version de l'API en disponibilité générale.

## Limites de ressource de liaison privée partagée

#### NOTE

Les indexeurs peuvent accéder aux ressources en toute sécurité via des points de terminaison privés gérés via l'[API de ressource de liaison privée partagée](#), comme décrit dans ce [guide pratique](#).

RESSOURCE	GRATUIT	DE BASE	S1	S2	S3	S3 HD	L1	L2
Prise en charge de l'indexeur de point de terminais on privé	Non	Oui	Oui	Oui	Oui	Non	Oui	Oui
Prise en charge du point de terminais on privé pour les indexeurs avec un ensemble de compétences <sup>1</sup>	Non	Non	Non	Oui	Oui	Non	Oui	Oui
Nombre maximal de points de terminais on privés	N/A	10 ou 30	100	400	400	N/A	20	20
Nombre maximal de types de ressource s distincts <sup>2</sup>	N/A	4	7	15	15	N/A	4	4

<sup>1</sup> L'enrichissement par IA et l'analyse d'image sont gourmands en calculs et consomment des quantités disproportionnées de puissance de traitement disponible. Par conséquent, pour des niveaux de service de recherche inférieurs, leur définition pour une exécution dans l'environnement privé peut avoir un impact négatif sur les performances et la stabilité du service de recherche.

<sup>2</sup> Le nombre de types de ressources distincts est calculé comme le nombre de valeurs de `groupId` uniques utilisées dans toutes les ressources de liaison privée partagée pour un service de recherche donné, quel que soit l'état de la ressource.

## Limites des synonymes

Le nombre maximal de cartes de synonymes varie en fonction du niveau. Chaque règle peut avoir jusqu'à 20 expansions, où une expansion est un terme équivalent. Par exemple, pour le mot « chat », l'association avec « minou », « félin » et « felis » (le genre des chats) est comptée comme 3 expansions.

RESSOURCE	GRATUIT	DE BASE	S1	S2	S3	S3-HD	L1	L2
Mappages de synonymes maximum	3	3	5	10	20	20	10	10
Nombre maximal de règles par mappage	5 000	20000	20000	20000	20000	20000	20000	20000

## Requêtes par seconde

Les estimations du nombre de requêtes par seconde doivent être développées indépendamment par chaque client. La taille et la complexité des index et des requêtes ainsi que la quantité de trafic sont les principaux facteurs qui déterminent le nombre de requêtes par seconde. Si ces facteurs sont inconnus, il est impossible d'établir des estimations significatives.

Les estimations sont plus prévisibles si elles sont calculées sur des services qui s'exécutent sur des ressources dédiées (niveaux De base et Standard). Vous pouvez mieux estimer les requêtes par seconde, car vous contrôlez davantage de paramètres. Pour obtenir de l'aide sur la manière d'aborder les estimations, consultez [Performances et optimisation de Recherche cognitive Azure](#).

Pour les niveaux à stockage optimisé (L1 et L2), attendez-vous à un plus faible débit des requêtes et à une latence plus élevée que les niveaux Standard.

## Limites de données (enrichissement de l'IA)

Un [pipeline d'enrichissement par IA](#) faisant appel à une ressource Analyse de texte pour la [reconnaissance d'entités](#), [l'extraction de phrases clés](#), [l'analyse des sentiments](#), la [détection de la langue](#) et la [détection d'informations personnelles](#) est soumis à des limites de données. La taille maximale d'un enregistrement doit être de 50 000 caractères telle que mesurée par [String.Length](#). Si vous avez besoin de découper vos données avant de les envoyer à l'Analyseur des sentiments, utilisez la [compétence Fractionnement du texte](#).

## Limites de la limitation

Les requêtes de recherche et les demandes d'indexation sont limitées dès que le système s'approche de la capacité maximale. Le comportement de la limitation varie en fonction des API. Les API de requête (recherche/suggestion/saisie semi-automatique) et les API d'indexation se limitent dynamiquement en fonction de la charge du service. Les API d'indexation sont soumises à des limites de taux de requêtes statiques.

Limites de taux de requêtes statiques pour les opérations liées à un index :

- Lister les index (GET /indexes) : 5 par seconde par unité de recherche
- Obtenir un index (GET /indexes/myindex) : 10 par seconde par unité de recherche
- Créer un index (POST /indexes) : 12 par minute par unité de recherche
- Créer ou mettre à jour un index (PUT /indexes/myindex) : 6 par seconde par unité de recherche
- Supprimer un index (DELETE /indexes/myindex) : 12 par minute par unité de recherche

## Limites de requête d'API

- 16 Mo maximum par requête<sup>1</sup>
- La longueur maximale d'une URL est de 8 Ko
- 1 000 documents maximum par lot de charges, de fusions ou de suppressions d'index
- 32 champs maximum dans la clause \$orderby
- La taille maximale des termes de recherche du texte encodé en UTF-8 est de 32 766 octets (32 Ko moins 2 octets)

<sup>1</sup> Dans la Recherche cognitive Azure, le corps d'une requête est soumis à une limite supérieure de 16 Mo. Cela signifie qu'une limite pratique est imposée au contenu des champs individuels ou des collections qui ne font pas l'objet de limites théoriques (pour plus d'informations sur la composition et les restrictions des champs, consultez [Types de données pris en charge](#)).

## Limites de réponse d'API

- 1 000 documents maximum retournés par page de résultats de recherche
- 100 suggestions maximum retournées par requête d'API de suggestion

## Limites de clés API

Les clés API sont utilisées pour l'authentification de service. Il existe deux types de clé API. Les clés d'administration sont spécifiées dans l'en-tête de la demande et accordent un accès complet en lecture et en écriture au service. Les clés de requête sont en lecture seule, spécifiées dans l'URL et généralement distribuées aux applications clientes.

- 2 clés administrateur maximum par service
- 50 clés de requête maximum par service

# Ajuster la capacité dans la Recherche cognitive Azure

04/10/2020 • 16 minutes to read • [Edit Online](#)

Avant de [provisionner un service de recherche](#) et de choisir un niveau tarifaire particulier, prenez quelques minutes pour bien comprendre le rôle des réplicas et des partitions dans un service, et la manière dont vous pouvez ajuster la capacité d'un service en fonction des variations de la demande de ressources.

La capacité est une fonction du [niveau que vous choisissez](#) (les niveaux déterminent les caractéristiques matérielles), et de la combinaison de réplicas et de partitions nécessaires pour les charges de travail projetées. Selon le niveau et la taille de l'ajustement, le processus d'ajout ou de réduction de capacité peut prendre de 15 minutes à plusieurs heures.

Si vous modifiez l'allocation des réplicas et des partitions, nous vous recommandons d'utiliser le portail Azure. Le portail applique des limites aux combinaisons autorisées inférieures aux limites maximales d'un niveau. Toutefois, si vous souhaitez suivre une approche de provisionnement basée sur un script ou du code, [Azure PowerShell](#) et l'[API REST Gestion](#) constituent des solutions alternatives.

## Terminologie : réplicas et partitions

<i>Partitions</i>	Fournissent un stockage des index et des E/S pour les opérations de lecture-écriture (par exemple, lors de la reconstruction ou de l'actualisation d'un index). Chaque partition se partage l'index total. Si vous allouez trois partitions, l'index est divisé en trois.
<i>Réplicas</i>	Instances du service de recherche, principalement utilisées pour équilibrer la charge des opérations de requête. Chaque réplica est une copie d'un index. Si vous allouez trois réplicas, vous avez trois copies d'un index disponibles pour traiter les demandes de requête.

## Quand ajouter des nœuds

Au départ, un service se voit allouer un niveau minimal de ressources consistant en une partition et un réplica.

Un seul service doit avoir suffisamment de ressources pour gérer toutes les charges de travail (indexation et requêtes). Aucune charge de travail ne s'exécute en arrière-plan. Vous pouvez planifier l'indexation à des moments où les demandes de requête sont naturellement moins fréquentes, mais à part cela, le service n'établit pas de priorité entre les tâches. De plus, un certain degré de redondance lisse les performances des requêtes lorsque les services ou les nœuds sont mis à jour en interne.

En règle générale, les applications de recherche tendent à avoir besoin de plus de réplicas que de partitions, en particulier lorsque les opérations de service favorisent les charges de travail de requête. La section [Haute disponibilité](#) explique pourquoi.

#### NOTE

L'ajout de réplicas ou de partitions augmente le coût d'exécution du service et peut introduire de légères variations dans la façon dont les résultats sont classés. N'oubliez pas d'utiliser la [calculatrice de prix](#) pour bien comprendre l'impact de l'ajout de noeuds supplémentaires sur votre facturation. Le [graphique ci-dessous](#) peut vous aider à déterminer le nombre d'unités de recherche requises pour une configuration spécifique. Pour plus d'informations sur l'impact des réplicas supplémentaires sur le traitement des requêtes, consultez [Classement des résultats](#).

## Comment allouer des réplicas et partitions

1. Connectez-vous au [portail Azure](#), puis sélectionnez le service de recherche.
2. Dans **Paramètres**, ouvrez la page **Mise à l'échelle** pour modifier les réplicas et partitions.

La capture d'écran suivante montre un service standard approvisionné avec un réplica et une partition. La formule en bas indique combien d'unités de recherche sont utilisées (1). Si le prix unitaire était de 100 (prix fictif), le coût mensuel de l'exécution de ce service serait de 100 en moyenne.

The screenshot shows the 'Scale' section of the Azure Search blade. At the top, there are 'Save' and 'Discard' buttons. Below them, under 'Replicas', it says 'Replicas distribute workloads across the service. We guarantee 99.9% availability for read operations with 2 replicas, and for read and write operations with 3 or more replicas.' A link 'Learn more about the SLA for Azure Search' is provided. A slider for 'Number of replicas' is set to 1, which is highlighted with a red box. Under 'Partitions', it says 'Partitions allow for scaling of document counts as well as faster data ingestion by spanning your index over multiple Azure Search Units (applies to Standard tiers only)'. A slider for 'Number of partitions' is also set to 1, which is highlighted with a red box. Below this, '25 GiB Storage' is mentioned. In the 'Scale Totals' section, it states 'Search Unit is the unit of pricing used for Azure Search. It is calculated as the number of partitions x number of replicas. The maximum number of search units is 36.' A red box highlights the text '1 of 36 available search units' and '1 Replicas X 1 Partitions'.

3. Utilisez le curseur pour augmenter ou diminuer le nombre de partitions. La formule en bas indique combien d'unités de recherche utilisées.

Cet exemple double la capacité, avec deux réplicas et partitions. Notez le nombre d'unités de recherche. Il est désormais de quatre, car la formule de facturation multiplie le nombre de réplicas par le nombre de partitions ( $2 \times 2$ ). Le doublement de la capacité fait plus que doubler le coût de l'exécution du service. Si le coût d'une unité de recherche était de 100, la nouvelle facture mensuelle serait désormais de 400.

Pour le coût unitaire de chaque niveau, visitez la [page Tarification](#).

**Save** **Discard**

### Replicas

Replicas distribute workloads across the service. We guarantee 99.9% availability for read operations with 2 replicas, and for read and write operations with 3 or more replicas. [Learn more about the SLA for Azure Search](#)

Number of replicas



2

---

### Partitions

Partitions allow for scaling of document counts as well as faster data ingestion by spanning your index over multiple Azure Search Units (applies to Standard tiers only).

Number of partitions



2

**50 GiB Storage**

---

### Scale Totals

Search Unit is the unit of pricing used for Azure Search. It is calculated as the number of partitions x number of replicas. The maximum number of search units is 36.

**4 of 36 available search units**  
2 Replicas X 2 Partitions

4. Cliquez sur **Enregistrer** pour confirmer les modifications.

**Save** **Discard**

### Update search service

This update may affect the billing. Are you sure you want to update this service?

**Yes** **No**

La prise d'effet des changements de capacité peut nécessiter quelques heures. Une fois que le processus a démarré, vous ne pouvez pas l'annuler, et il n'y a pas de supervision en temps réel des ajustements de réplicas et de partitions. Toutefois, le message suivant reste visible pendant que les changements prennent effet.

**Save** **Discard**

**Updating... This operation may take a while...**

#### NOTE

Une fois qu'un service a été provisionné, il ne peut pas être mis à niveau vers un niveau supérieur. Vous devez créer un service de recherche au nouveau niveau et recharger vos index. Pour obtenir des instructions sur l'approvisionnement du service, voir [Créer un service Recherche cognitive Azure dans le portail](#).

De plus, les partitions et les réplicas sont gérés de manière exclusive et en interne par le service. Il n'existe pas de concept d'affinité du processeur ni d'affectation d'une charge de travail à un nœud spécifique.

## combinaisons de partitions et de réplicas

Un service basique peut avoir exactement une partition et jusqu'à trois réplicas, pour une limite maximale de trois unités de recherche. Les seules ressources ajustables sont les réplicas. Vous devez disposer d'au moins 2 réplicas pour la haute disponibilité sur des requêtes.

Tous les services de recherche standard et à stockage optimisé peuvent supposer les combinaisons suivantes de réplicas et de partitions, soumises à la limite de 36 unités de stockage.

	1 PARTITION	2 PARTITIONS	3 PARTITIONS	4 PARTITIONS	6 PARTITIONS	12 PARTITIONS
1 réplica	1 unité de recherche	2 unités de recherche	3 unités de recherche	4 unités de recherche	6 unités de recherche	12 unités de recherche
2 réplicas	2 unités de recherche	4 unités de recherche	6 unités de recherche	8 unités de recherche	12 unités de recherche	24 unités de recherche
3 réplicas	3 unités de recherche	6 unités de recherche	9 unités de recherche	12 unités de recherche	18 unités de recherche	36 unités de recherche
4 réplicas	4 unités de recherche	8 unités de recherche	12 unités de recherche	16 unités de recherche	24 unités de recherche	N/A
5 réplicas	5 unités de recherche	10 unités de recherche	15 unités de recherche	20 unités de recherche	30 unités de recherche	N/A
6 réplicas	6 unités de recherche	12 unités de recherche	18 unités de recherche	24 unités de recherche	36 unités de recherche	N/A
12 réplicas	12 unités de recherche	24 unités de recherche	36 unités de recherche	N/A	N/A	N/A

Les unités de recherche, leur tarification et leur capacité sont détaillées sur le site web Azure. Pour plus d'informations, consultez la rubrique [Tarification](#).

#### NOTE

Le nombre de réplicas et de partitions est divisible par 12 de manière égale (plus précisément, 1, 2, 3, 4, 6, 12).

Recherche cognitive Azure divise au préalable chaque index en 12 partitions pour que celles-ci puissent être réparties équitablement sur plusieurs partitions. Par exemple, si votre service comporte trois partitions et que vous créez un index, chaque partition contiendra quatre partitions de l'index. Le partitionnement d'un index réalisé par la Recherche cognitive Azure est un détail d'implémentation susceptible d'être modifié dans des futures versions. Le nombre de partitions (12 à l'heure actuelle) peut être, à l'avenir, totalement différent.

## Haute disponibilité

Nous vous recommandons généralement de démarrer avec une partition et un ou deux réplicas, puis de monter en puissance à mesure que les volumes de requête se créent. Les charges de travail de requêtes s'exécutent principalement sur des réplicas. Si vous nécessitez une haute disponibilité ou un débit plus important, vous aurez probablement besoin de réplicas supplémentaires.

Recommandations générales pour la haute disponibilité :

- Deux réplicas pour la haute disponibilité des charges de travail en lecture seule (requêtes)
- Trois réplicas minimum pour la haute disponibilité des charges de travail en lecture/écriture (les requêtes et l'indexation en tant que documents individuels sont ajoutées, mises à jour ou supprimées)

Les contrats de niveau de service (SLA) pour la Recherche cognitive Azure sont ciblés au moment des opérations de requête et des mises à jour d'index qui se composent d'ajout, de mise à jour ou de suppression de documents.

Le niveau De base est plafonné à une partition et trois réplicas. Si vous souhaitez pouvoir répondre immédiatement aux fluctuations de la demande sur le plan de l'indexation et du débit des requêtes, songez à passer à l'un des niveaux Standard. Si vous trouvez que vos besoins de stockage augmentent beaucoup plus rapidement que votre débit de requête, envisagez l'un des niveaux de stockage optimisé.

## Récupération d'urgence

Il n'existe actuellement aucun mécanisme intégré de récupération d'urgence. L'ajout de partitions ou de réplicas ne vous permettra pas d'atteindre les objectifs de récupération d'urgence qui ont été fixés. L'approche la plus courante consiste à intégrer la redondance au niveau du service en configurant un deuxième service de recherche dans une autre région. Comme avec la disponibilité pendant une reconstruction d'index, la redirection ou la logique de basculement doit provenir de votre code.

## Estimer les réplicas

Sur un service de production, vous devez allouer trois réplicas pour les besoins de contrat SLA. Si vous rencontrez des problèmes de lenteur, vous pouvez ajouter des réplicas afin que des copies supplémentaires de l'index soient mises en ligne pour prendre en charge des charges de travail avec davantage de requêtes et équilibrer la charge des requêtes sur plusieurs réplicas.

Nous ne fournissons pas d'instructions sur le nombre de réplicas nécessaires pour prendre en charge les charges de requêtes. Les performances des requêtes dépendent de la complexité des requêtes et des charges de travail concurrentes. Bien que l'ajout de réplicas entraîne clairement une amélioration de l'évolutivité et des performances, le résultat final n'est pas strictement linéaire : si vous ajoutez trois réplicas, le débit n'est pas forcément multiplié par trois.

Pour obtenir des conseils sur l'estimation du nombre de requêtes par seconde (RPS) pour votre solution, consultez [Mettre à l'échelle pour les performances](#) et [Superviser les requêtes](#)

## Estimer les partitions

Le [niveau que vous choisissez](#) détermine la taille et la vitesse des partitions, et chaque niveau est optimisé par rapport à un ensemble de caractéristiques adaptées à différents scénarios. Si vous choisissez un niveau tarifaire supérieur, vous aurez sans doute besoin de moins de partitions que si vous optez pour le niveau S1. L'une des questions auxquelles vous devrez répondre au moyen de tests autonomes est de savoir si une partition plus grande et plus coûteuse offre de meilleures performances que deux partitions moins chères sur un service provisionné à un niveau inférieur.

Les applications de recherche nécessitant une actualisation des données en temps réel ou presque ont proportionnellement besoin de plus de partitions que de réplicas. L'ajout de partitions répartit les opérations de lecture/écriture sur un plus grand nombre de ressources de calcul. Il vous offre également davantage d'espace disque pour stocker des documents et des index supplémentaires.

Plus les index sont grands, plus ils sont longs à interroger. Par conséquent, peut-être constaterez-vous que chaque augmentation incrémentielle des partitions nécessite une augmentation plus faible mais proportionnelle des réplicas. La complexité et le volume de vos requêtes auront une incidence sur la vitesse d'exécution des requêtes.

## Étapes suivantes

[Choisir un niveau tarifaire pour Recherche cognitive Azure](#)

# Mettre à l'échelle pour les performances de la Recherche cognitive Azure

04/10/2020 • 16 minutes to read • [Edit Online](#)

Cet article décrit les meilleures pratiques applicables dans le cadre de scénarios avancés présentant des exigences complexes en matière d'extensibilité et de disponibilité.

## Commencer avec des valeurs de référence

Avant d'entreprendre un déploiement à plus grande échelle, vous devez savoir à quoi ressemble une charge de requête standard. Les instructions ci-après peuvent vous aider à atteindre les valeurs de requête de référence.

1. Choisissez une latence cible (ou durée maximale) nécessaire à l'exécution d'une recherche standard .
2. Créez et testez une charge de travail réelle avec votre service de recherche en utilisant un jeu de données réaliste pour mesurer les taux de latence.
3. Commencez par un petit nombre de requêtes par seconde (RPS), puis augmentez graduellement le nombre d'exécutions dans le test jusqu'à ce que la latence des requêtes soit inférieure à la cible prédéfinie. Il s'agit d'un test d'évaluation important qui vous aidera à planifier la mise à l'échelle à mesure que l'utilisation de votre application s'intensifie.
4. Dans la mesure du possible, réutilisez les connexions HTTP. Si vous utilisez le Kit de développement logiciel (SDK) .NET Recherche cognitive Azure, cela signifie que vous devez réutiliser une instance ou une instance [SearchIndexClient](#), et si vous utilisez l'API REST, vous devez réutiliser une instance [HttpClient](#) unique.
5. Variez la substance des requêtes afin que la recherche s'effectue sur différentes parties de votre index. Il est important de varier les requêtes, car si vous exécutez continuellement les mêmes requêtes de recherche, la mise en cache des données commencera à offrir de meilleures performances qu'avec un ensemble de requêtes plus disparates.
6. Variez la structure des requêtes afin d'obtenir différents types de requêtes. En effet, le niveau de performance varie selon les requêtes de recherche. Par exemple, une recherche de document ou une suggestion de recherche s'exécute plus rapidement qu'une requête comportant un nombre important de facettes et de filtres. La composition de test doit inclure diverses requêtes, approximativement dans les mêmes proportions que celles que vous rencontreriez en production.

Lors de la création de ces charges de travail de test, certaines caractéristiques de la Recherche cognitive Azure doivent être prises en compte :

- L'envoi (push) simultané d'un nombre excessif de requêtes de recherche risque de surcharger votre service. Dans ce cas, vous verrez les codes de réponse HTTP 503. Pour éviter l'obtention du code 503 lors du test, commencez avec différentes plages de requêtes de recherche pour visualiser les différences de taux de latence à mesure que vous ajoutez d'autres requêtes.
- Le service Recherche cognitive Azure n'exécute pas de tâches d'indexation en arrière-plan. Si votre service gère les charges de travail de requête et d'indexation simultanément, prenez cela en compte en introduisant des travaux d'indexation dans vos tests de requête ou en explorant les options d'exécution des travaux d'indexation pendant les heures creuses.

#### TIP

Vous pouvez simuler une charge de requête réaliste à l'aide des outils de test de charge. Essayez de [tester la charge avec Azure DevOps](#) ou utilisez l'une de ces [alternatives](#).

## Mettre à l'échelle pour un volume de requêtes élevé

Un service est surchargé quand les requêtes prennent trop de temps ou que le service commence à abandonner des requêtes. Dans ces cas de figure, vous pouvez résoudre le problème de l'une des manières suivantes :

- **Ajouter des réplicas**

Chaque réplica est une copie de vos données, ce qui permet au service d'équilibrer la charge des requêtes sur plusieurs copies. L'équilibrage de la charge et la réPLICATION DES DONNÉES sont entièrement gérés par la Recherche cognitive Azure, mais vous pouvez à tout moment changer le nombre de réplicas alloués à votre service. Vous pouvez allouer jusqu'à 12 réplicas dans un service de recherche Standard, et 3 dans un service de recherche de base. Les réplicas peuvent être ajustés sur le [portail Azure](#) ou via [PowerShell](#).

- **Créer un service à un niveau supérieur**

la Recherche cognitive Azure est proposée dans [plusieurs niveaux](#), chacun d'eux offrant des niveaux de performances différents. Dans certains cas, le nombre de requêtes peut être trop élevé pour qu'elles puissent être traitées dans le niveau que vous avez choisi, même si les réplicas sont optimisés. Envisagez alors de passer à un niveau supérieur plus performant, tel que le niveau S3 Standard qui est conçu pour des scénarios impliquant un grand nombre de documents et des charges de requêtes extrêmement élevées.

## Mettre à l'échelle pour les requêtes individuelles lentes

Une requête unique dont l'exécution prend trop de temps peut également entraîner des taux de latence élevés. Dans ce cas, l'ajout de réplicas n'offrira aucune amélioration. Vous pouvez tenter de résoudre ce problème de deux manières :

- **Augmenter les partitions**

Une partition divise les données entre des ressources de calcul supplémentaires. Deux partitions divisent les données en deux, trois partitions les divisent en trois, et ainsi de suite. L'un des avantages découlant de cette opération réside dans l'accélération potentielle de l'exécution des requêtes les plus lentes grâce au calcul parallèle. Nous avons constaté une parallélisation sur les requêtes qui affichent une faible sélectivité, telles que les requêtes portant sur de nombreux documents ou les facettes comptabilisant un grand nombre de documents. Dans la mesure où des calculs significatifs sont nécessaires pour évaluer la pertinence des documents ou pour comptabiliser les documents, l'ajout de partitions supplémentaires contribue à accélérer l'exécution des requêtes.

Il peut y avoir un maximum de 12 partitions dans le service de recherche de niveau Standard, et 1 partition dans le service de recherche de niveau De base. Les réplicas peuvent être ajustés sur le [portail Azure](#) ou via [PowerShell](#).

- **Limiter les champs à cardinalité élevée**

un champ à cardinalité élevée est un champ utilisable comme facette ou comme filtre et contenant un grand nombre de valeurs uniques. Ce champ consomme donc une quantité substantielle de ressources lors du calcul des résultats. Par exemple, la définition d'un champ ID produit ou Description en tant que

champ utilisable comme facette ou comme filtre est considérée comme une cardinalité élevée, car la plupart des valeurs sont uniques pour chaque document. Dans la mesure du possible, limitez le nombre de champs à cardinalité élevée.

- **Passer à un niveau de Recherche supérieur**

Choisir un niveau de Recherche cognitive Azure supérieur est une autre façon d'améliorer les performances des requêtes lentes. Chaque niveau supérieur se caractérise par un processeur plus rapide et par une plus grande quantité de mémoire. Ces deux aspects ont un impact positif sur les performances des requêtes.

## Mettre à l'échelle pour la disponibilité

Les réplicas permettent non seulement de réduire la latence des requêtes, mais ils peuvent également offrir une plus grande disponibilité. Avec un seul réplica, attendez-vous à des temps d'arrêt périodiques en raison du redémarrage du serveur après les mises à jour logicielles ou lorsque d'autres événements de maintenance se produisent. Par conséquent, il est important d'évaluer si votre application requiert une haute disponibilité des recherches (requêtes) ainsi que des écritures (événements d'indexation). La Recherche cognitive Azure offre des options de contrat de niveau de service sur tous les services de recherche payants avec les attributs suivants :

- Deux réplicas pour la haute disponibilité des charges de travail en lecture seule (requêtes)
- Trois réplicas (ou plus) pour la haute disponibilité des charges de travail en lecture-écriture (requêtes et indexation)

Pour plus d'informations, consultez le [Contrat de niveau de service de la Recherche cognitive Azure](#).

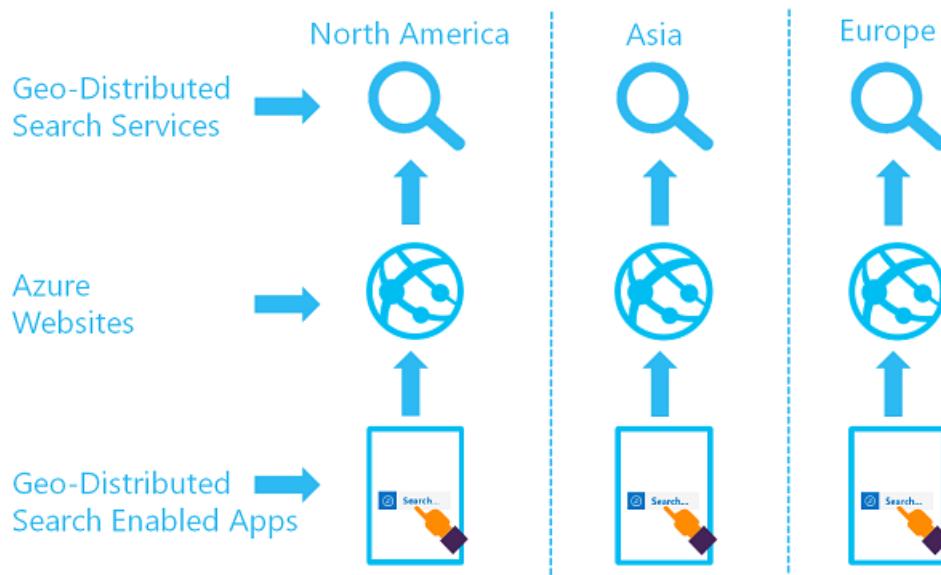
Étant donné que les réplicas sont des copies de vos données, l'utilisation de plusieurs réplicas permet au service Recherche cognitive Azure de gérer les redémarrages et la maintenance des machines pour un réplica spécifique, alors que les requêtes continuent de s'exécuter sur d'autres réplicas. À l'inverse, si vous retirez des réplicas, vous constaterez une dégradation des performances de requête, en supposant que ces réplicas constituaient des ressources sous-exploitées.

## Mettre à l'échelle pour les charges de travail géodistribuées et la géoredondance

Dans le cas des charges de travail géodistribuées, les utilisateurs éloignés du centre de données hôte font face à des taux de latence plus élevés. L'une des manières d'atténuer ce problème consiste à approvisionner plusieurs services de recherche dans des régions plus proches de ces utilisateurs.

La Recherche cognitive Azure n'offre actuellement pas de méthode automatisée de géo-réPLICATION des index de Recherche cognitive Azure sur différentes régions, mais quelques techniques permettent d'implémenter et de gérer simplement ce processus. Elles sont présentées dans les sections suivantes.

L'objectif d'un ensemble géodistribué de services de recherche est de disposer de plusieurs index disponibles dans au moins deux régions, et de rediriger l'utilisateur vers le service Recherche cognitive Azure qui offre la plus faible latence, comme illustré dans cet exemple :



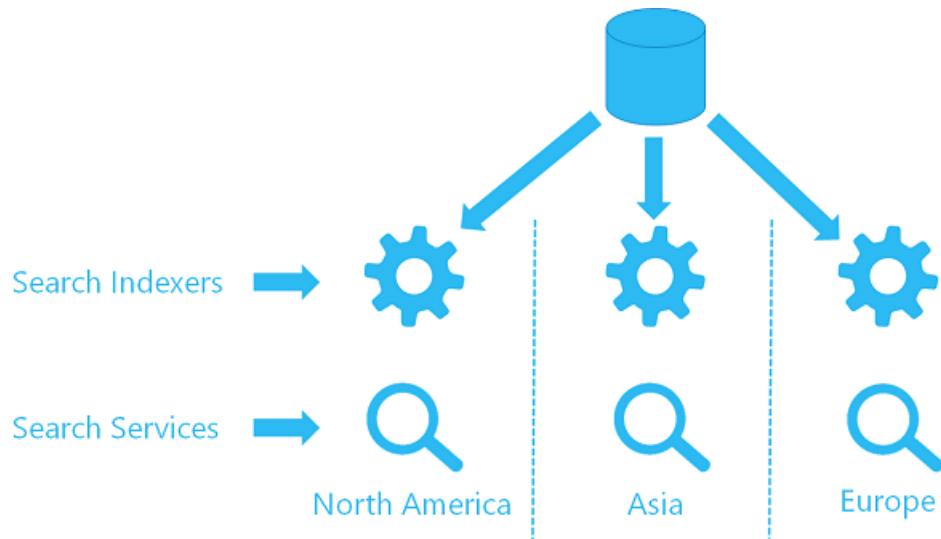
### Garantir la synchronisation des données entre plusieurs services

Il existe deux méthodes pour synchroniser vos services de recherche distribués : [l'indexeur de Recherche cognitive Azure](#) ou l'[API Push](#) (également appelée [API REST de Recherche cognitive Azure](#)).

### Mettre à jour le contenu dans plusieurs services à l'aide d'indexeurs

Si vous utilisez déjà un indexeur sur un service unique, vous pouvez configurer un second indexeur sur un deuxième service pour qu'il utilise le même objet de source de données, en extrayant les données au même emplacement. Chacun des services de chaque région dispose de son propre indexeur et d'un index cible (votre index de recherche n'est pas partagé, ce qui signifie que les données sont dupliquées), mais chaque indexeur référence la même source de données.

Voici une vue d'ensemble de l'aspect d'une telle architecture.



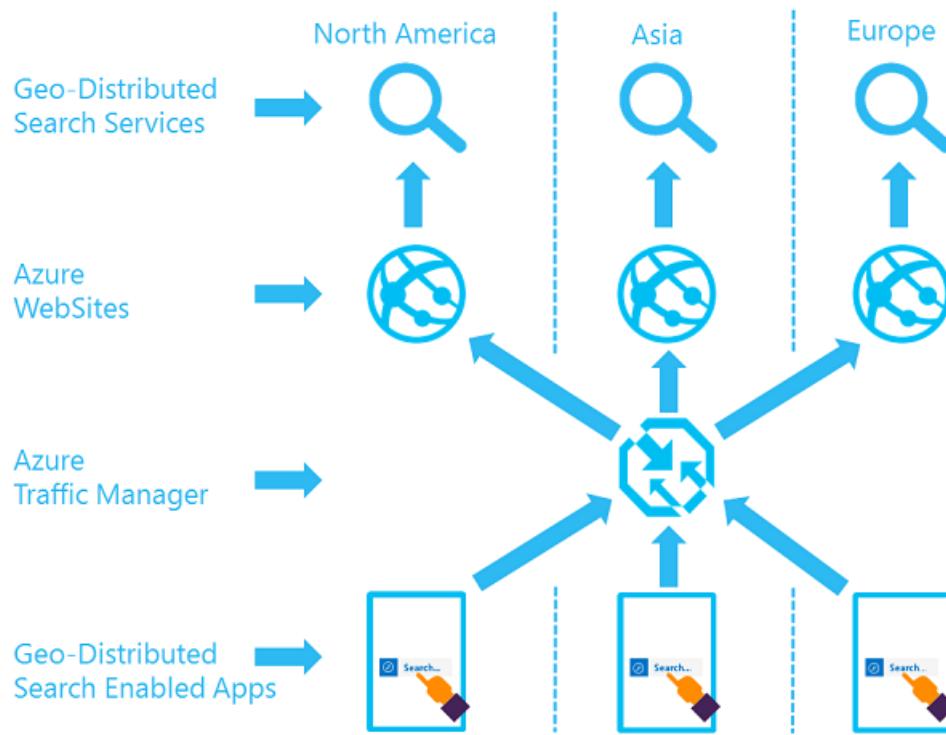
### Envoyer (push) les mises à jour de contenu à plusieurs services par le biais d'API REST

Si vous utilisez l'[API REST Recherche cognitive Azure](#) pour [envoyer \(push\) le contenu de votre index Recherche cognitive Azure](#), vous pouvez assurer la synchronisation continue de vos différents services de recherche en envoyant les modifications à tous ces services chaque fois qu'une mise à jour est nécessaire. Dans votre code, veillez à gérer les cas dans lesquels la mise à jour d'un service de recherche échoue, mais réussit pour d'autres services de recherche.

## Exploiter Azure Traffic Manager

[Azure Traffic Manager](#) vous permet d'acheminer les requêtes vers plusieurs sites géo-localisés et pris en charge

par plusieurs services de recherche. Traffic Manager offre l'avantage de pouvoir tester la Recherche cognitive Azure pour vous assurer qu'il est disponible et de rediriger les utilisateurs vers d'autres services de recherche en cas d'interruption du service. En outre, si vous acheminez des requêtes de recherche via des sites Web Azure, Azure Traffic Manager permet d'équilibrer les charges lorsque le site web est opérationnel mais pas la Recherche cognitive Azure. Voici un exemple d'architecture tirant parti de Traffic Manager.



## Étapes suivantes

Pour plus d'informations sur les niveaux tarifaires et les limites de service de chacun d'eux, consultez [Limites de service](#). Consultez la page [Planification des capacités](#) pour en savoir plus sur les combinaisons de partitions et de réplicas.

Pour une discussion sur les performances et des démonstrations des techniques présentées dans cet article, regardez la vidéo suivante :

# Modèles de conception pour les applications SaaS mutualisées et Recherche cognitive Azure

04/10/2020 • 20 minutes to read • [Edit Online](#)

Une application mutualisée est une application qui fournit les mêmes services et fonctionnalités à plusieurs clients qui ne peuvent pas voir ni partager les données d'un autre client. Ce document aborde les stratégies d'isolation de client pour les applications mutualisées conçues avec Recherche cognitive Azure.

## Concepts de Recherche cognitive Azure

En tant que solution SaaS (search-as-a-service), [Recherche cognitive Azure](#) permet aux développeurs d'ajouter des expériences de recherche enrichies dans les applications sans avoir à gérer d'infrastructure, ni devenir un expert en matière de récupération d'informations. Les données sont téléchargées vers le service, puis stockées dans le cloud. À l'aide de requêtes simples dans l'API Recherche cognitive Azure, les données peuvent ensuite être modifiées et faire l'objet de recherches.

### Rechercher des services, des index, des champs et des documents

Avant d'aborder les modèles de conception, il est important de comprendre certains concepts de base.

Lorsque vous utilisez Recherche cognitive Azure, vous vous abonnez à un *service de recherche*. Lorsque les données sont téléchargées vers Recherche cognitive Azure, elles sont stockées dans un *index* au sein du service de recherche. Un seul service peut contenir plusieurs index. Pour utiliser les concepts familiers des bases de données, le service de recherche peut être comparé à une base de données, tandis que les index au sein d'un service peuvent être comparés aux tables dans une base de données.

Chaque index au sein d'un service de recherche possède son propre schéma, qui est défini par un certain nombre de *champs* personnalisables. Les données sont ajoutées à un index Recherche cognitive Azure sous la forme de *documents* individuels. Chaque document doit être téléchargé dans un index spécifique et doit respecter le schéma de cet index. Lors de la recherche de données à l'aide de Recherche cognitive Azure, les requêtes de recherche en texte intégral sont exécutées sur un index spécifique. Pour comparer ces concepts à ceux d'une base de données, les champs peuvent être comparés aux colonnes d'une table et les documents peuvent être comparés aux lignes.

### Extensibilité

Tout service Recherche cognitive Azure dans le [niveau tarifaire](#) Standard peut évoluer en deux dimensions : stockage et disponibilité.

- *partitions* pour augmenter le stockage d'un service de recherche.
- *réplicas* à un service pour augmenter le débit des requêtes qu'un service de recherche peut gérer.

L'ajout et la suppression de partitions et de réplicas lorsque nécessaire permet à la capacité du service de recherche d'augmenter en fonction de la quantité de données et du trafic demandés par l'application. Pour qu'un service de recherche obtienne un [Contrat de niveau de service](#) en lecture, deux réplicas sont nécessaires. Pour qu'un service obtienne un [Contrat de niveau de service](#) en lecture et en écriture, trois réplicas sont nécessaires.

### Limites du service et de l'index dans Recherche cognitive Azure

Il existe différents [niveaux tarifaires](#) dans la Recherche cognitive Azure et chaque niveau présente des [limites](#) et [quotas](#) différents. Certaines de ces limites se situent au niveau du service, certaines au niveau de l'index, et d'autres au niveau de la partition.

	DE BASE	STANDARD1	STANDARD2	STANDARD3	STANDARD3 HD
Nombre maximal de réplicas par service	3	12	12	12	12
Nombre maximal de partitions par service	1	12	12	12	3
Nombre maximal d'unités de recherche (réplicas*partitions) par service	3	36	36	36	36 (3 partitions max.)
Stockage maximal par service	2 Go	300 Go	1,2 To	2,4 To	600 Go
Stockage maximal par partition	2 Go	25 Go	100 Go	200 Go	200 Go
Nombre maximal d'index par service	5	50	200	200	3000 (1 000 index max. par partition)

#### Haute densité S3

Dans le niveau tarifaire S3 de Recherche cognitive Azure, il existe une option pour le mode haute densité (HD) conçu spécifiquement pour les scénarios mutualisés. Dans de nombreux cas, il est nécessaire de prendre en charge un grand nombre de clients plus petits sous un seul service pour tirer parti des avantages de simplicité et de rentabilité.

S3 HD permet de réunir les nombreux index de petite taille sous la gestion d'un seul service de recherche en troquant la possibilité d'effectuer un scale-out des index à l'aide de partitions contre la possibilité d'héberger plus d'index dans un seul service.

Un service S3 est conçu pour héberger un nombre fixe d'index (200 maximum) et permet à chaque index d'évoluer horizontalement au fur et à mesure que de nouvelles partitions sont ajoutées au service. L'ajout de partitions à des services S3 HD augmente le nombre maximal d'index que le service peut héberger. La taille maximale idéale pour un index S3 HD est d'environ 50 à 80 Go, bien qu'il n'y ait pas de limite de taille matérielle sur chaque index imposé par le système.

## Considérations relatives aux applications mutualisées

Les applications mutualisées doivent distribuer efficacement les ressources entre les clients tout en conservant un certain niveau de confidentialité entre les différents clients. Il existe quelques considérations à prendre en compte lors de la conception de l'architecture pour ce type d'application :

- *Isolation des locataires* : les développeurs d'applications doivent prendre les mesures appropriées pour s'assurer qu'aucun locataire ne bénéficie d'un accès non autorisé ou non désiré aux données d'autres

locataires. Au-delà de la confidentialité des données, les stratégies d'isolation des clients nécessitent une gestion efficace des ressources partagées et la protection contre les voisins bruyants.

- *Coût des ressources cloud* : comme pour toute autre application, les solutions logicielles doivent rester compétitives au niveau du coût en tant que composant d'une application mutualisée.
- *Facilité des opérations* : lors du développement d'une architecture mutualisée, l'impact sur les opérations et la complexité de l'application est un facteur important. Recherche cognitive Azure propose un [Contrat de niveau de service à 99,9 %](#).
- *Envergure internationale* : les applications mutualisées devront peut-être servir efficacement des locataires qui sont répartis dans le monde entier.
- *Scalabilité* : Les développeurs d'applications doivent trouver l'équilibre entre le fait de maintenir un niveau de complexité des applications suffisamment faible et la conception de l'application de façon à ce qu'elle évolue avec le nombre de locataires, ainsi que la taille des données et la charge de travail des locataires.

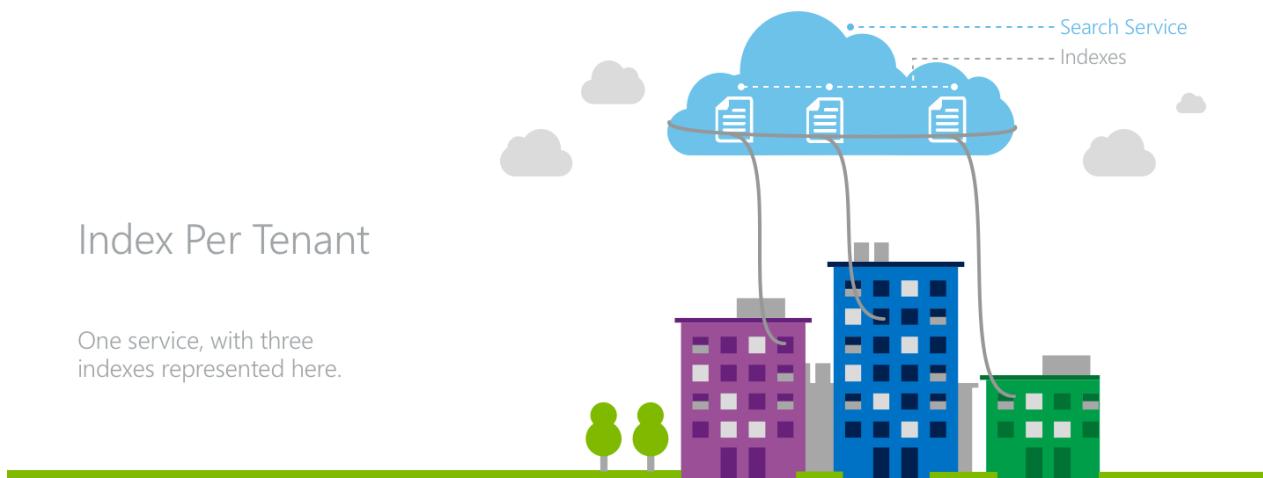
Recherche cognitive Azure propose quelques limites qui peuvent être utilisées pour isoler les données et la charge de travail des clients.

## Modélisation de la mutualisation avec Recherche cognitive Azure

Dans le cas d'un scénario mutualisé, le développeur de l'application consomme un ou plusieurs services de recherche et répartit les clients entre les services, les index ou les deux. Recherche cognitive Azure offre quelques modèles courants pour la modélisation d'un scénario mutualisé :

1. *Index par locataire* : chaque locataire a son propre index dans un service de recherche qui est partagé avec d'autres locataires.
2. *Service par locataire* : chaque locataire possède son propre service Recherche cognitive Azure dédié, pour un niveau de séparation maximal entre les données et la charge de travail.
3. *Combinaison des deux* : les locataires les plus volumineux et actifs se voient attribuer des services dédiés, tandis que les locataires plus petits se voient attribuer des index individuels au sein de services partagés.

### 1. Index par client



Dans un modèle d'index par client, plusieurs clients occupent un seul service Recherche cognitive Azure où chaque client a son propre index.

Les clients bénéficient de l'isolation des données, car toutes les requêtes de recherche et les opérations sur les documents sont émises au niveau de l'index dans Recherche cognitive Azure. Dans la couche d'application, il est entendu qu'il est nécessaire de diriger le trafic des divers clients vers les index appropriés, tout en gérant les ressources au niveau du service pour tous les clients.

Un attribut clé du modèle d'index par client est la possibilité pour le développeur d'applications d'augmenter la capacité d'un service de recherche pour les clients de l'application. Si la répartition de la charge de travail est inégale entre les clients, la combinaison optimale des clients peut être distribuée entre les index d'un service de recherche pour prendre en charge un nombre de clients très actifs, gourmands en ressources, tout en gérant simultanément une longue file de clients moins actifs. Le compromis réside dans l'incapacité du modèle à gérer les situations où chaque client est simultanément très actif.

Le modèle d'index par client sert de base à un modèle de coût variable, où un service Recherche cognitive Azure entier est acheté d'avance, puis rempli avec des clients. Cela permet d'attribuer la capacité inutilisée aux essais et aux comptes gratuits.

Pour les applications avec une envergure internationale, le modèle d'index par client n'est peut-être pas le plus efficace. Si les clients d'une application sont répartis dans le monde entier, un service distinct peut être nécessaire pour chaque région, ce qui peut dupliquer les coûts pour chacun d'eux.

Recherche cognitive Azure permet la croissance des index individuels et du nombre total d'index. Si un niveau tarifaire approprié est choisi, des réplicas et des partitions peuvent être ajoutés au service de recherche entier lorsqu'un index individuel au sein du service devient trop important en termes de stockage ou de trafic.

Si le nombre total d'index devient trop important pour un seul service, un autre service doit être configuré pour prendre en charge les nouveaux clients. Si les index doivent être déplacés entre les services de recherche lorsque de nouveaux services sont ajoutés, les données de l'index doivent être copiées manuellement d'un index vers l'autre car le déplacement d'un index n'est pas autorisé dans Recherche cognitive Azure.

## 2. Service par client



Dans une architecture de service par client, chaque client possède son propre service de recherche.

Dans ce modèle, l'application atteint le niveau maximal d'isolation pour ses clients. Chaque service bénéficie d'un débit et d'un stockage dédiés pour gérer les requêtes de recherche, ainsi que des clés d'API distinctes.

Pour les applications où chaque client a une grande envergure ou si la charge de travail varie peu d'un client à l'autre, le modèle de service par client constitue un choix efficace car les ressources ne sont pas partagées entre les charges de travail de différents clients.

Un modèle de service par client offre également l'avantage d'un modèle à coût prévisible et fixe. Il n'existe aucun investissement initial pour un service de recherche entier jusqu'à ce qu'il y ait un client pour le remplir.

Cependant, le coût par client est supérieur au modèle d'index par client.

Le modèle de service par client constitue une solution efficace pour les applications d'envergure internationale. Avec des clients répartis géographiquement, il est facile de placer le service de chaque client dans la région appropriée.

Les défis liés à la mise à l'échelle de ce modèle se présentent lorsque la croissance des clients individuels surpassé celle de leur service. Actuellement, Recherche cognitive Azure ne prend pas en charge la mise à niveau du niveau de tarification d'un service de recherche, donc toutes les données doivent être copiées manuellement vers un nouveau service.

### 3. Combinaison des deux modèles

Un autre modèle pour la modélisation mutualisée est de combiner les stratégies d'index par client et de service par client.

En mélangeant les deux modèles, les clients les plus volumineux d'une application peuvent occuper des services dédiés, tandis que la longue file de clients moins actifs et plus petits peuvent occuper des index dans un service partagé. Ce modèle garantit que les clients les plus volumineux bénéficient toujours de performances élevées du service tout en protégeant les clients plus petits des voisins bruyants.

Cependant, l'implémentation de cette stratégie repose sur la capacité à prévoir quels clients nécessiteront un service dédié au lieu d'un index dans un service partagé. La complexité de l'application augmente avec la nécessité de gérer les deux modèles d'architecture mutualisée.

### Atteindre une granularité encore plus fine

Les modèles de conception ci-dessus pour les scénarios mutualisés dans Recherche cognitive Azure supposent qu'il existe une étendue uniforme, où chaque client constitue une instance entière d'une application. Toutefois, les applications peuvent parfois gérer plusieurs étendues plus petites.

Si les modèles service par client et index par client ne sont pas des étendues suffisamment petites, il est possible de modéliser un index pour atteindre un degré de granularité encore plus fin.

Pour qu'un seul index se comporte différemment pour des points de terminaison clients différents, il est possible d'ajouter un champ à un index qui désigne une certaine valeur pour chaque client possible. À chaque fois qu'un client appelle Recherche cognitive Azure pour interroger ou modifier un index, le code de l'application cliente spécifie la valeur appropriée pour ce champ à l'aide de la fonctionnalité [filter](#) de Recherche cognitive Azure au moment de la requête.

Cette méthode peut être utilisée pour obtenir une fonctionnalité de comptes d'utilisateurs distincts, de niveaux d'autorisation distincts et même d'applications distinctes.

#### NOTE

Utiliser l'approche décrite ci-dessus pour configurer un index unique pour servir plusieurs locataires affecte la pertinence des résultats de recherche. Les notes de pertinence de la recherche sont calculées au niveau de l'index et non au niveau du locataire. Ainsi, toutes les données des locataires sont incorporées dans les statistiques sous-jacentes des notes de pertinence, comme la fréquence des termes.

## Étapes suivantes

Recherche cognitive Azure est un choix attrayant pour de nombreuses applications. Lorsque vous évaluez les différents modèles de conception pour les applications mutualisées, consultez les [différents niveaux tarifaires](#) et les [limites de service](#) respectives pour mieux adapter la Recherche cognitive Azure aux architectures et aux charges de travail de toutes tailles.

Toutes questions relatives à Recherche cognitive Azure et les scénarios partagés au sein d'une architecture mutualisée peuvent être envoyées vers [azuresearch\\_contact@microsoft.com](mailto:azuresearch_contact@microsoft.com).

# Versions d'API dans la Recherche cognitive Azure

04/10/2020 • 11 minutes to read • [Edit Online](#)

Le service Recherche cognitive Azure déploie régulièrement des mises à jour de fonctionnalités. Parfois, ces mises à jour requièrent une nouvelle version de l'API pour maintenir la compatibilité descendante. La publication d'une nouvelle version vous permet de contrôler quand et comment intégrer les mises à jour du service de recherche dans votre code.

En règle générale, l'équipe de Recherche cognitive Azure publie de nouvelles versions quand cela s'avère nécessaire uniquement. En effet, la mise à niveau de votre code pour utiliser une nouvelle version de l'API peut demander un certain travail. Une nouvelle version est requise uniquement si certains aspects de l'API ont changé d'une manière qui interrompt la compatibilité descendante. Ces changements peuvent se produire en raison de correctifs de fonctionnalités existantes ou en raison de nouvelles fonctionnalités qui modifient la surface d'exposition des API existantes.

La même règle s'applique pour les mises à jour du Kit de développement logiciel (SDK). Le SDK de Recherche cognitive Azure suit les règles de [gestion sémantique de version](#), ce qui signifie que sa version comprend trois parties : majeure, mineure et numéro de version (par exemple, 1.1.0). Une nouvelle version majeure du Kit de développement logiciel (SDK) est publiée uniquement en cas de modifications qui interrompent la compatibilité descendante. Les mises à jour de fonctionnalités sans rupture incrémentent la version mineure. Pour corriger les bogues, nous augmentons uniquement le numéro de version.

## IMPORTANT

Les Kits de développement logiciel (SDK) Azure pour .NET, Java, Python et JavaScript déplient de nouvelles bibliothèques de client pour Recherche cognitive Azure. Actuellement, aucune des bibliothèques des Kits de développement logiciel (SDK) Azure ne prend entièrement en charge les API REST de recherche (2020-06-30) ou les API REST de gestion (2020-03-13) les plus récentes, mais cela changera au fil du temps. Vous pouvez consulter régulièrement cette page ou la rubrique [Nouveautés](#) pour connaître les annonces sur les améliorations fonctionnelles.

## Versions non prises en charge

Mettez à niveau les solutions de recherche existantes vers la dernière version de l'API REST avant le 15 octobre 2020. À compter de cette date, les versions suivantes de l'API REST Recherche cognitive Azure seront retirées et ne seront plus prises en charge :

- 2015-02-28
- 2015-02-28-Preview
- 2014-07-31-Preview ;
- 2014-10-20-Preview.

Les versions du kit SDK .NET Recherche cognitive Azure antérieures à la version [3.0.0-rc](#) seront également retirées, car elles ciblent l'une de ces versions de l'API REST.

Après cette date, les applications utilisant l'API REST déconseillée ou les versions du Kit de développement logiciel (SDK) ne fonctionneront plus et nécessiteront une mise à niveau. Comme pour toute modification de ce type, nous donnons un préavis de 12 mois pour vous laisser le temps de vous adapter.

Pour continuer à utiliser le service Recherche cognitive Azure, migrez le code existant ciblant [l'API REST](#) vers [l'API REST version 2020-06-30](#) ou vers un Kit de développement logiciel (SDK) plus récent avant le 15 octobre 2020. Si vous avez des questions sur la mise à jour vers la dernière version, envoyez un courrier à

azuresearch\_contact@microsoft.com pour le 15 mai 2020 afin d'être certain de disposer de suffisamment de temps pour mettre à jour votre code.

## API REST

Une instance du service Recherche cognitive Azure prend en charge plusieurs versions de l'API REST, y compris la plus récente. Vous pouvez continuer à utiliser une version lorsqu'elle n'est pas la plus récente, mais nous vous recommandons de [migrer votre code](#) pour utiliser la dernière version. Lorsque vous utilisez l'API REST, vous devez spécifier la version de l'API dans chaque requête via le paramètre « api-version » (Version de l'API). Lorsque vous utilisez le Kit de développement logiciel (SDK) .NET, la version du Kit de développement logiciel (SDK) que vous utilisez détermine la version correspondante de l'API REST. Si vous utilisez un Kit de développement logiciel (SDK) plus ancien, vous pouvez continuer à exécuter ce code sans changement, même si le service est mis à niveau pour prendre en charge une version plus récente de l'API.

Le tableau suivant fournit l'historique des versions actuelles et précédentes de l'API REST du service de recherche. La documentation est publiée uniquement pour les versions préliminaires et celles stables les plus récentes.

### API du service de recherche

Créez et gérez du contenu sur un service de recherche.

VERSION	STATUT	DESCRIPTION
Recherche 30-06-2020	Stable	Version stable la plus récente des API REST de recherche, avec des progrès dans le scoring de la pertinence et la disponibilité générale de la base de connaissances.
Recherche 30-06-2020 - Préversion	PRÉVERSION	Préversion associée à la version stable. Comprend plusieurs <a href="#">fonctionnalités d'évaluation</a> .
Recherche 2019-05-06	Stable	Ajoute des <a href="#">types complexes</a> .
Recherche 2019-05-06-Preview	PRÉVERSION	Préversion associée à la version stable.
Recherche 11-11-2017	Stable	Ajoute des ensembles de compétences et l' <a href="#">enrichissement par IA</a> .
Recherche 11-11-2017 - Préversion	PRÉVERSION	Préversion associée à la version stable.
Recherche 01-09-2016	Stable	Ajoute des <a href="#">indexeurs</a> .
Recherche 01-09-2016 - Préversion	PRÉVERSION	Préversion associée à la version stable.
Recherche 28-02-2015	Non prise en charge après le 10/10/2020	Première version en disponibilité générale.
Recherche 28-02-2015 - Préversion	Non prise en charge après le 10/10/2020	Préversion associée à la version stable.
Recherche 20-10-2014 - Préversion	Non prise en charge après le 10/10/2020	Deuxième préversion publique.

VERSION	STATUT	DESCRIPTION
Recherche 31-07-2014 - Préversion	Non prise en charge après le 10/10/2020	Première préversion publique.

## API REST de gestion

Créez et configurez un service de recherche et gérez les clés API.

VERSION	STATUT	DESCRIPTION
Gestion 01-08-2020	Stable	Version stable la plus récente des API REST de gestion. Ajoute la prise en charge de ressource de liaison privée partagée généralement disponible pour toutes les ressources faisant l'objet d'un accès sortant, à l'exception de celles indiquées dans la préversion.
Gestion 01-08-2020 – Préversion	PRÉVERSION	Actuellement en préversion : prise en charge des ressources de liaison privée partagée pour Azure Functions et Azure Database pour MySQL.
Gestion 2020-03-13	Stable	Ajoute un <a href="#">point de terminaison privé</a> via une liaison privée et des <a href="#">règles d'IP réseau</a> pour les nouveaux services. Pour plus d'informations, consultez cette <a href="#">spécification Swagger</a> .
Gestion 2019-10-01-Preview	Préversion	Aucune fonctionnalité en préversion n'a été introduite dans cette liste. Cette préversion est fonctionnellement équivalente à la version 13-03-2020. Pour plus d'informations, consultez cette <a href="#">spécification Swagger</a> .
Gestion 19-08-2015	Stable	Première version en disponibilité générale des API REST de gestion. Fournit le provisionnement des services, le scale-up et la gestion des clés API. Pour plus d'informations, consultez cette <a href="#">spécification Swagger</a> .
Gestion 19-08-2015 - Préversion	PRÉVERSION	Première préversion des API REST de gestion. Pour plus d'informations, consultez cette <a href="#">spécification Swagger</a> .

## Kit SDK Azure pour .NET

Le tableau suivant fournit des liens vers des versions plus récentes du Kit de développement logiciel (SDK).

VERSION DU SDK	STATUT	DESCRIPTION
----------------	--------	-------------

VERSION DU SDK	STATUT	DESCRIPTION
Azure.Search.Documents 11.0	Stable	Nouvelle bibliothèque de client du Kit de développement logiciel (SDK) .NET Azure, publiée en juillet 2020. Cible l'API REST Recherche (api-version = 2020-06-30), mais ne prend pas encore en charge les filtres géographiques et <a href="#">FieldBuilder</a> .
Microsoft.Azure.Search 10.0	Stable	Publiée en mai 2019. Cible l'API REST de recherche api-version=2019-05-06.
Microsoft.Azure.Management.Search 4.0.0	Stable	Cible l'API REST Gestion-version=2020-08-01.
Microsoft.Azure.Management.Search 3.0.0	Stable	Cible l'API REST de gestion de versions 19-08-2015.

## Kit SDK Azure pour Java

VERSION DU SDK	STATUT	DESCRIPTION
Java azure-search-documents 11	Stable	Nouvelle bibliothèque de client du Kit de développement logiciel (SDK) .NET Azure, publiée en juillet 2020. Cible l'API REST de recherche api-version=2019-05-06.
Java Management Client 1.35.0	Stable	Cible l'API REST de gestion de versions 19-08-2015.

## Kit SDK Azure pour JavaScript

VERSION DU SDK	STATUT	DESCRIPTION
JavaScript azure-search 11.0	Stable	Nouvelle bibliothèque de client du Kit de développement logiciel (SDK) .NET Azure, publiée en juillet 2020. Cible l'API REST de recherche api-version=2016-09-01.
JavaScript azure-arm-search	Stable	Cible l'API REST de gestion de versions 19-08-2015.

## Kit SDK Azure pour Python

VERSION DU SDK	STATUT	DESCRIPTION
Python azure-search-documents 11.0	Stable	Nouvelle bibliothèque de client du Kit de développement logiciel (SDK) .NET Azure, publiée en juillet 2020. Cible l'API REST de recherche api-version=2019-05-06.

VERSION DU SDK	STATUT	DESCRIPTION
Python azure-mgmt-search 1.0	Stable	Cible l'API REST de gestion de versions 19-08-2015.

# Fonctionnalités d'évaluation dans Recherche cognitive Azure

04/10/2020 • 9 minutes to read • [Edit Online](#)

Cet article est une liste complète de toutes les fonctionnalités disponibles en préversion publique. Les fonctionnalités d'évaluation sont fournies sans contrat de niveau de service et ne sont pas recommandées pour les charges de travail de production. Pour plus d'informations, consultez [Conditions d'Utilisation Supplémentaires relatives aux Évaluations Microsoft Azure](#).

Les fonctionnalités d'évaluation qui passent en disponibilité générale sont supprimées de cette liste. Si une fonctionnalité n'est pas listée ci-dessous, vous pouvez supposer qu'elle est en disponibilité générale. Pour accéder aux annonces concernant la disponibilité générale, consultez [Mises à jour de service](#) ou [Nouveautés](#).

FONCTIONNALITÉ	CATEGORY	DESCRIPTION	DISPONIBILITÉ
<a href="#">Compétence Azure Machine Learning (AML)</a>	Enrichissement de l'IA	Nouveau type de compétence permettant d'intégrer un point de terminaison d'inférence à partir d'Azure Machine Learning. Pour bien démarrer, suivez <a href="#">ce tutoriel</a> .	Utilisez l' <a href="#">API REST Recherche 2020-06-30-Preview</a> ou 2019-05-06-Preview. Également disponible sur le portail, dans la conception de compétences, si les services Recherche cognitive et Azure ML sont déployés sur le même abonnement.
<a href="#">Paramètre featuresMode</a>	Pertinence (scoring)	Expansion du score de pertinence pour inclure des détails : score de similarité par champ, fréquence de terme par champ, et nombre de jetons uniques correspondants par champ. Vous pouvez consommer ces points de données dans des <a href="#">solutions de scoring personnalisées</a> .	Ajoutez ce paramètre de requête à l'aide de <a href="#">Recherche dans des documents (REST)</a> avec api-version=2020-06-30-Preview ou 2019-05-06-Preview.
<a href="#">Sessions de débogage</a>	Portail, enrichissement par IA (ensemble de compétences)	Éditeur d'ensemble de compétences dans la session utilisé pour examiner et résoudre les problèmes liés aux ensembles de compétences. Les correctifs appliqués au cours d'une session de débogage peuvent être enregistrés dans un ensemble de compétences dans le service.	Portail uniquement, à l'aide des liens en milieu de page Vue d'ensemble pour ouvrir une session de débogage.

FONCTIONNALITÉ	CATEGORY	DESCRIPTION	DISPONIBILITÉ
<a href="#">Suppression réversible native d'objets blob</a>	Indexeurs, objets blob Azure	L'indexeur Stockage Blob Azure dans Recherche cognitive Azure reconnaît les objets blob qui sont dans un état de suppression réversible, et supprime le document de recherche correspondant durant l'indexation.	Ajoutez ce paramètre de configuration à l'aide de <a href="#">Créer un indexeur (REST)</a> avec api-version=2020-06-30-Preview ou api-version=2019-05-06-Preview.
<a href="#">Compétence de recherche d'entité personnalisée</a>	Enrichissement par IA (ensemble de compétences)	Compétence cognitive qui recherche du texte dans une liste personnalisée de mots et d'expressions définie par l'utilisateur. À l'aide de cette liste, elle étiquète tous les documents contenant des entités correspondantes. La compétence prend également en charge un degré de correspondance approximative qui peut être appliquée pour rechercher des correspondances similaires sans être rigoureusement exactes.	Référez cette qualification en préversion à l'aide de l'éditeur d'ensemble de compétences dans le portail ou <a href="#">créez un ensemble de compétences (REST)</a> avec api-version=2020-06-30-Preview or api-version=2019-05-06-Preview.
<a href="#">Compétence de détection d'informations d'identification personnelle</a>	Enrichissement par IA (ensemble de compétences)	Compétence cognitive utilisée lors de l'indexation qui extrait les informations d'identification personnelle d'un texte d'entrée et vous donne la possibilité de les masquer de différentes façons.	Référez cette qualification en préversion à l'aide de l'éditeur d'ensemble de compétences dans le portail ou <a href="#">créez un ensemble de compétences (REST)</a> avec api-version=2020-06-30-Preview or api-version=2019-05-06-Preview.
<a href="#">Enrichissement incrémentiel</a>	Configuration de l'indexeur	Ajoute la mise en cache à un pipeline d'enrichissement, ce qui vous permet de réutiliser la sortie existante si une modification ciblée, telle que la mise à jour d'un ensemble de compétences ou d'un autre objet, ne change pas le contenu. La mise en cache s'applique uniquement aux documents enrichis générés par un ensemble de compétences.	Ajoutez ce paramètre de configuration à l'aide de <a href="#">Créer un indexeur (REST)</a> avec api-version=2020-06-30-Preview ou api-version=2019-05-06-Preview.

FONCTIONNALITÉ	CATEGORY	DESCRIPTION	DISPONIBILITÉ
<a href="#">Indexeur Cosmos DB : API MongoDB, API Gremlin, API Cassandra</a>	Source de données d'indexeur	Pour Cosmos DB, l'API SQL est en disponibilité générale, mais les API MongoDB, Gremlin et Cassandra sont en préversion.	Pour Gremlin et Cassandra uniquement, <a href="#">inscrivez-vous au préalable</a> afin que la prise en charge puisse être activée pour votre abonnement sur le back-end. Les sources de données MongoDB peuvent être configurées dans le portail. Autrement, la configuration de la source de données pour les trois API est prise en charge à l'aide de <a href="#">Créer une source de données (REST)</a> avec api-version=2020-06-30-Preview ou api-version=2019-05-06-Preview.
<a href="#">Indexeur Azure Data Lake Storage Gen2</a>	Source de données d'indexeur	Indexe le contenu et les métadonnées de Data Lake Storage Gen2.	L' <a href="#">inscription</a> est nécessaire pour que la prise en charge puisse être activée pour votre abonnement sur le back-end. Accédez à cette source de données à l'aide de <a href="#">Créer une source de données (REST)</a> avec api-version=2020-06-30-Preview ou api-version=2019-05-06-Preview.
<a href="#">moreLikeThis</a>	Requête	Recherche les documents correspondant à un document spécifique. Cette fonctionnalité existait dans les préversions antérieures.	Ajoutez ce paramètre de requête dans les appels à <a href="#">Recherche dans des documents (REST)</a> avec api-version=2020-06-30-Preview, 2019-05-06-Preview, 2016-09-01-Preview ou 2017-11-11-Preview.

## Comment appeler une API REST en préversion

Le service Recherche cognitive Azure publie toujours les fonctionnalités expérimentales par le biais de l'API REST, puis par le biais des préversions du SDK .NET.

Les fonctionnalités en préversion sont disponibles pour le test et l'expérimentation en vue de recueillir des commentaires sur la conception et la mise en œuvre de la fonctionnalité. Ainsi, les fonctionnalités en préversion peuvent changer au fil du temps, voire de manière à empêcher la compatibilité descendante. Ce comportement diffère de celui des fonctionnalités dans la version à disposition générale, qui sont stables et peu susceptibles de changer à l'exception d'améliorations et de correctifs peu importants pour la compatibilité descendante. En outre, les fonctionnalités en préversion ne sont pas toujours intégrées à une version à disposition générale.

Alors que certaines fonctionnalités d'évaluation peuvent être disponibles dans le portail et le SDK .NET, l'API REST dispose toujours de fonctionnalités d'évaluation.

- Pour les opérations de recherche, [2020-06-30-Preview](#) correspond à la préversion actuelle.
- Pour les opérations de gestion, [2019-10-01-Preview](#) correspond à la préversion actuelle.

Les préversions plus anciennes sont toujours opérationnelles mais deviennent obsolètes au fil du temps. Si votre code appelle `api-version=2019-05-06-Preview`, `api-version=2016-09-01-Preview` ou `api-version=2017-11-11-Preview`, ces appels sont toujours valides. Toutefois, seule la dernière préversion est actualisée avec des améliorations.

La syntaxe de l'exemple suivant illustre un appel à l'API en préversion.

```
POST https://[service name].search.windows.net/indexes/hotels-idx/docs/search?api-version=2020-06-30-Preview
Content-Type: application/json
api-key: [admin key]
```

Le service Recherche cognitive Azure est disponible sous plusieurs versions. Pour plus d'informations, consultez [Versions d'API](#).

## Étapes suivantes

Consultez la documentation de référence relative à l'API REST en préversion du service de recherche. Si vous rencontrez des problèmes, sollicitez notre aide sur [Stack Overflow](#) ou [contactez le support](#).

[Informations de référence sur l'API REST du service de recherche \(préversion\)](#)

# Utilisation de Microsoft.Azure.Search (v10) dans une application .NET

04/10/2020 • 42 minutes to read • [Edit Online](#)

Cet article explique comment créer et gérer des objets de recherche en utilisant C# et le [Kit de développement logiciel \(SDK\) .NET Recherche cognitive Azure \(v10\)](#). La version 10 est la version la plus récente du package Microsoft.Azure.Search. À l'avenir, les nouvelles fonctionnalités seront déployées dans [Azure.Search.Documents](#) par l'équipe du Kit de développement logiciel (SDK) Azure.

Si vous avez des projets de développement en cours ou existants, continuez à utiliser la version 10. Pour les nouveaux projets, ou pour utiliser de nouvelles fonctionnalités, vous devez transférer une solution de recherche existante vers la nouvelle bibliothèque.

## Contenu de la version 10

Le SDK se compose de quelques bibliothèques clientes qui vous permettent de gérer vos index, sources de données, indexeurs et cartes de synonymes, ainsi que de charger et gérer des documents et d'exécuter des requêtes, sans avoir à gérer les détails de HTTP et de JSON. Ces bibliothèques clientes sont distribuées sous la forme de packages NuGet.

Le package NuGet principal est `Microsoft.Azure.Search`, méta-package qui inclut tous les autres packages en tant que dépendances. Utilisez ce package si vous débutez, ou si vous savez que votre application a besoin de toutes les fonctionnalités de Recherche cognitive Azure.

Les autres packages NuGet dans le SDK sont les suivants :

- `Microsoft.Azure.Search.Data` : Utilisez ce package si vous développez une application .NET à l'aide de Recherche cognitive Azure et que vous devez uniquement interroger ou mettre à jour des documents dans vos index. Si vous devez également créer ou mettre à jour des index, des cartes de synonymes ou d'autres ressources de niveau service, utilisez le package `Microsoft.Azure.Search` à la place.
- `Microsoft.Azure.Search.Service` : Utilisez ce package si vous développez un processus d'automatisation en .NET pour gérer les index Recherche cognitive Azure, cartes de synonymes, indexeurs, sources de données ou autres ressources de niveau service. Si vous devez uniquement interroger ou mettre à jour des documents dans vos index, utilisez le package `Microsoft.Azure.Search.Data` à la place. Si vous avez besoin de toutes les fonctionnalités de Recherche cognitive Azure, utilisez le package `Microsoft.Azure.Search` à la place.
- `Microsoft.Azure.Search.Common` : Types courants requis par les bibliothèques .NET de Recherche cognitive Azure. Vous n'avez pas besoin d'utiliser ce package directement dans votre application. Il est uniquement destiné à être utilisé en tant que dépendance.

Les différentes bibliothèques clientes définissent des classes comme `Index`, `Field` et `Document`, ainsi que des opérations telles que `Indexes.Create` et `Documents.Search` sur les classes `SearchServiceClient` et `SearchIndexClient`. Ces classes sont organisées dans les espaces de noms suivants :

- [Microsoft.Azure.Search](#)
- [Microsoft.Azure.Search.Models](#).

Si vous souhaitez formuler des commentaires pour une prochaine mise à jour du Kit de développement logiciel (SDK), consultez notre [page de commentaires](#) ou créez un problème sur [GitHub](#) en mentionnant « Recherche cognitive Azure » dans le titre.

Le SDK .NET cible la version [2019-05-06](#) de l'[API REST de Recherche cognitive Azure](#). Cette version inclut la prise en charge des [types complexes](#), de l'[enrichissement par IA](#), de la [saisie semi-automatique](#) et du [mode d'analyse JsonLines](#) lors de l'indexation d'objets blob Azure.

Ce Kit de développement logiciel (SDK) ne prend pas en charge les [opérations de gestion](#) telles que la création et la mise à l'échelle des services de recherche, ainsi que la gestion des clés API. Si vous avez besoin de gérer vos ressources de recherche à partir d'une application .NET, vous pouvez utiliser le [Kit de développement logiciel \(SDK\) .NET de la Recherche cognitive Azure](#).

## Mise à niveau vers la dernière version du Kit de développement logiciel (SDK)

Si vous utilisez déjà une version antérieure du Kit de développement logiciel (SDK) .NET Recherche cognitive Azure et que vous souhaitez mettre à niveau vers la dernière version mise à la disposition générale, [cet article](#) vous explique comment procéder.

## Configuration requise pour le SDK

1. Visual Studio 2017 ou version ultérieure.
2. Votre propre service Recherche cognitive Azure. Pour utiliser le SDK, vous devez connaître le nom de votre service et une ou plusieurs clés API. [Créer un service dans le portail](#) vous guidera à travers ces étapes.
3. Téléchargez le [package NuGet](#) du SDK .NET Recherche cognitive Azure en utilisant « Gérer les packages NuGet » dans Visual Studio. Recherchez simplement le nom de package [Microsoft.Azure.Search](#) sur NuGet.org (ou l'un des autres noms de package ci-dessus si vous avez uniquement besoin d'un sous-ensemble des fonctionnalités).

Le SDK .NET Recherche cognitive Azure prend en charge les applications qui ciblent .NET Framework versions 4.5.2 et supérieures, ainsi que .NET Core versions 2.0 et supérieures.

## Principaux scénarios

Vous devez faire plusieurs choses dans votre application de recherche. Dans ce didacticiel, nous aborderons ces principaux scénarios :

- Création d'un index
- Remplissage de l'index avec des documents
- Recherche de documents à l'aide de filtres et de la recherche en texte intégral

L'exemple de code suivant illustre chacun de ces scénarios. N'hésitez pas à utiliser les extraits de code dans votre propre application.

### Vue d'ensemble

L'application exemple que nous allons examiner crée un index nommé « hotels », le remplit avec des documents, puis exécute des requêtes de recherche. Voici le programme principal, décrivant le flux global :

```

// This sample shows how to delete, create, upload documents and query an index
static void Main(string[] args)
{
    IConfigurationBuilder builder = new ConfigurationBuilder().AddJsonFile("appsettings.json");
    IConfigurationRoot configuration = builder.Build();

    SearchServiceClient serviceClient = CreateSearchServiceClient(configuration);

    string indexName = configuration["SearchIndexName"];

    Console.WriteLine("{0}", "Deleting index...\\n");
    DeleteIndexIfExists(indexName, serviceClient);

    Console.WriteLine("{0}", "Creating index...\\n");
    CreateIndex(indexName, serviceClient);

    ISearchIndexClient indexClient = serviceClient.Indexes.GetClient(indexName);

    Console.WriteLine("{0}", "Uploading documents...\\n");
    UploadDocuments(indexClient);

    ISearchIndexClient indexClientForQueries = CreateSearchIndexClient(configuration);

    RunQueries(indexClientForQueries);

    Console.WriteLine("{0}", "Complete. Press any key to end application...\\n");
    Console.ReadKey();
}

```

#### NOTE

Vous trouverez le code source complet de l'exemple d'application utilisé dans cette procédure sur [GitHub](#).

Nous allons le détailler, étape par étape. Tout d'abord, nous devons créer un objet `SearchServiceClient`. Cet objet vous permet de gérer les index. Pour en construire un, vous devez indiquer le nom de votre service Recherche cognitive Azure, ainsi qu'une clé API d'administration. Vous pouvez entrer ces informations dans le fichier `appsettings.json` de [l'exemple d'application](#).

```

private static SearchServiceClient CreateSearchServiceClient(IConfigurationRoot configuration)
{
    string searchServiceName = configuration["SearchServiceName"];
    string adminApiKey = configuration["SearchServiceAdminApiKey"];

    SearchServiceClient serviceClient = new SearchServiceClient(searchServiceName, new
    SearchCredentials(adminApiKey));
    return serviceClient;
}

```

#### NOTE

Si vous fournissez une clé incorrecte (par exemple, une clé de requête là où une clé d'administration était demandée), `SearchServiceClient` génère une `CloudException` avec le message d'erreur « `Forbidden` » la première fois que vous invoquez une méthode d'opération dessus, comme `Indexes.Create`. Si cette situation se produit, vérifiez la clé API.

Les quelques lignes suivantes appellent des méthodes pour créer un index nommé « `hotels` », en le supprimant s'il existe déjà. Nous étudierons ces méthodes un peu plus tard.

```
Console.WriteLine("{0}", "Deleting index...\n");
DeleteIndexIfExists(indexName, serviceClient);

Console.WriteLine("{0}", "Creating index...\n");
CreateIndex(indexName, serviceClient);
```

Ensuite, l'index doit être rempli. Pour remplir l'index, nous aurons besoin d'un `SearchIndexClient`. Il existe deux façons d'en obtenir un : en le créant ou en appelant `Indexes.GetClient` sur `SearchServiceClient`. Pour des raisons pratiques, nous allons opter pour la deuxième solution.

```
ISearchIndexClient indexClient = serviceClient.Indexes.GetClient(indexName);
```

#### NOTE

Dans une application de recherche classique, le remplissage et la gestion des index peuvent être gérés par un composant séparé des requêtes de recherche. `Indexes.GetClient` est pratique pour remplir un index car il vous évite de devoir fournir un `SearchCredentials` supplémentaire. Pour ce faire, il transmet la clé d'administration que vous avez utilisée afin de créer le `SearchServiceClient` dans le nouveau `SearchIndexClient`. Toutefois, dans la partie de votre application qui exécute des requêtes, il vaut mieux créer le `SearchIndexClient` directement afin de transmettre une clé de requête qui vous permet uniquement de lire des données au lieu d'une clé d'administration. Cela est conforme au principe du moindre privilège et vous permet de mieux sécuriser votre application. Pour en savoir plus sur les clés d'administration et les clés de requête, cliquez [ici](#).

Maintenant que nous avons un `SearchIndexClient`, nous pouvons remplir l'index. Le remplissage d'index est exécuté par une autre méthode que nous décrirons ultérieurement.

```
Console.WriteLine("{0}", "Uploading documents...\n");
UploadDocuments(indexClient);
```

Enfin, nous exécutons quelques requêtes de recherche et affichons les résultats. Cette fois-ci, nous utilisons un `SearchIndexClient` différent :

```
ISearchIndexClient indexClientForQueries = CreateSearchIndexClient(indexName, configuration);

RunQueries(indexClientForQueries);
```

Nous examinerons la méthode `RunQueries` plus en détail un peu plus tard. Voici le code permettant de créer le nouveau `SearchIndexClient` :

```
private static SearchIndexClient CreateSearchIndexClient(string indexName, IConfigurationRoot
configuration)
{
    string searchServiceName = configuration["SearchServiceName"];
    string queryApiKey = configuration["SearchServiceQueryApiKey"];

    SearchIndexClient indexClient = new SearchIndexClient(searchServiceName, indexName, new
    SearchCredentials(queryApiKey));
    return indexClient;
}
```

Cette fois, nous utilisons une clé de requête, car nous n'avons pas besoin d'obtenir un accès en écriture à l'index. Vous pouvez entrer ces informations dans le fichier `appsettings.json` de [l'exemple d'application](#).

Si vous exécutez cette application avec un nom de service et une clé API valides, la sortie doit être similaire à l'exemple suivant : (Une partie de la sortie de la console a été remplacée par « ... » pour l'illustration.)

```
Deleting index...

Creating index...

Uploading documents...

Waiting for documents to be indexed...

Search the entire index for the term 'motel' and return only the HotelName field:

Name: Secret Point Motel

Name: Twin Dome Motel

Apply a filter to the index to find hotels with a room cheaper than $100 per night, and return the hotelId and description:

HotelId: 1
Description: The hotel is ideally located on the main commercial artery of the city in the heart of New York. A few minutes away is Times Square and the historic centre of the city, as well as other places of interest that make New York one of America's most attractive and cosmopolitan cities.

HotelId: 2
Description: The hotel is situated in a nineteenth century plaza, which has been expanded and renovated to the highest architectural standards to create a modern, functional and first-class hotel in which art and unique historical elements coexist with the most modern comforts.

Search the entire index, order by a specific field (lastRenovationDate) in descending order, take the top two results, and show only hotelName and lastRenovationDate:

Name: Triple Landscape Hotel
Last renovated on: 9/20/2015 12:00:00 AM +00:00

Name: Twin Dome Motel
Last renovated on: 2/18/1979 12:00:00 AM +00:00

Search the hotel names for the term 'hotel':

HotelId: 3
Name: Triple Landscape Hotel
...
Complete. Press any key to end application...
```

Le code source complet de l'application est fourni à la fin de cet article.

Ensuite, nous allons étudier chacune des méthodes appelées par `Main`.

### Création d'un index

Après la création d'un `SearchServiceClient`, `Main` supprime l'index « hotels » s'il existe déjà. Cette suppression est effectuée par la méthode suivante :

```
private static void DeleteIndexIfExists(string indexName, SearchServiceClient serviceClient)
{
    if (serviceClient.Indexes.Exists(indexName))
    {
        serviceClient.Indexes.Delete(indexName);
    }
}
```

Cette méthode utilise `SearchServiceClient` pour vérifier si l'index existe et, dans l'affirmative, elle le supprime.

#### NOTE

L'exemple de code dans cet article utilise les méthodes synchrones du SDK .NET Recherche cognitive Azure pour plus de simplicité. Nous vous recommandons d'utiliser les méthodes asynchrones dans vos propres applications pour les rendre évolutives et réactives. Par exemple, dans la méthode ci-dessus, vous pouvez utiliser `ExistsAsync` et `DeleteAsync` au lieu de `Exists` et `Delete`.

Ensuite, `Main` crée un index « hotels » en appelant cette méthode :

```
private static void CreateIndex(string indexName, SearchServiceClient serviceClient)
{
    var definition = new Index()
    {
        Name = indexName,
        Fields = FieldBuilder.BuildForType<Hotel>()
    };

    serviceClient.Indexes.Create(definition);
}
```

Cette méthode crée un objet `Index` avec une liste d'objets `Field` qui définit le schéma du nouvel index. Chaque champ a un nom, un type de données et plusieurs attributs qui définissent son comportement de recherche. La classe `FieldBuilder` utilise la réflexion pour créer une liste d'objets `Field` pour l'index en examinant les attributs et propriétés publics de la classe de modèle `Hotel` donnée. Nous examinerons la classe `Hotel` plus en détail un peu plus tard.

#### NOTE

Vous pouvez toujours créer directement la liste des objets `Field` au lieu d'utiliser la fonctionnalité `FieldBuilder`, si nécessaire. Par exemple, vous ne souhaitez pas utiliser une classe de modèle, ou vous pouvez être amené à recourir à une classe de modèle existante, que vous ne souhaitez pas modifier en ajoutant des attributs.

En plus des champs, vous pouvez ajouter des profils de notation, des générateurs de suggestions ou des options CORS à l'index (ces paramètres sont omis de l'exemple par souci de concision). Vous trouverez plus d'informations sur l'objet `Index` et ses composants dans la page de [référence sur le Kit de développement logiciel \(SDK\)](#), ainsi que dans la page de [référence sur l'API REST de la Recherche cognitive Azure](#).

#### Remplissage de l'index

La prochaine étape dans `Main` remplit l'index nouvellement créé. Ce remplissage d'index s'effectue dans la méthode suivante : (Une partie du code a été remplacée par « ... » pour l'illustration. Pour le code complet de remplissage des données, voir l'exemple complet de la solution.)

```
private static void UploadDocuments(ISearchIndexClient indexClient)
{
```

```
var hotels = new Hotel[]
{
    new Hotel()
    {
        HotelId = "1",
        HotelName = "Secret Point Motel",
        ...
        Address = new Address()
        {
            StreetAddress = "677 5th Ave",
            ...
        },
        Rooms = new Room[]
        {
            new Room()
            {
                Description = "Budget Room, 1 Queen Bed (Cityside)",
                ...
            },
            new Room()
            {
                Description = "Budget Room, 1 King Bed (Mountain View)",
                ...
            },
            new Room()
            {
                Description = "Deluxe Room, 2 Double Beds (City View)",
                ...
            }
        }
    },
    new Hotel()
    {
        HotelId = "2",
        HotelName = "Twin Dome Motel",
        ...
    {
        StreetAddress = "140 University Town Center Dr",
        ...
    },
    Rooms = new Room[]
    {
        new Room()
        {
            Description = "Suite, 2 Double Beds (Mountain View)",
            ...
        },
        new Room()
        {
            Description = "Standard Room, 1 Queen Bed (City View)",
            ...
        },
        new Room()
        {
            Description = "Budget Room, 1 King Bed (Waterfront View)",
            ...
        }
    }
},
new Hotel()
{
    HotelId = "3",
    HotelName = "Triple Landscape Hotel",
    ...
    Address = new Address()
    {
        StreetAddress = "3393 Peachtree Rd",
        ...
    }
}
```

```

        },
        Rooms = new Room[]
        {
            new Room()
            {
                Description = "Standard Room, 2 Queen Beds (Amenities)",
                ...
            },
            new Room ()
            {
                Description = "Standard Room, 2 Double Beds (Waterfront View)",
                ...
            },
            new Room()
            {
                Description = "Deluxe Room, 2 Double Beds (Cityside)",
                ...
            }
        }
    );
}

var batch = IndexBatch.Upload(hotels);

try
{
    indexClient.Documents.Index(batch);
}
catch (IndexBatchException e)
{
    // Sometimes when your Search service is under load, indexing will fail for some of the
    // documents in
    // the batch. Depending on your application, you can take compensating actions like delaying and
    // retrying. For this simple demo, we just log the failed document keys and continue.
    Console.WriteLine(
        "Failed to index some of the documents: {0}",
        String.Join(", ", e.IndexingResults.Where(r => !r.Succeeded).Select(r => r.Key)));
}

Console.WriteLine("Waiting for documents to be indexed...\n");
Thread.Sleep(2000);
}

```

Cette méthode présente quatre parties. La première crée un tableau de 3 objets `Hotel`, comprenant chacun 3 objets `Room`, qui servent de données en entrée à charger dans l'index. Ces données sont codées en dur pour plus de simplicité. Dans votre application, vos données seront probablement issues d'une source de données externe, comme une base de données SQL.

La deuxième partie crée un `IndexBatch` contenant les documents. Vous spécifiez l'opération que vous souhaitez appliquer au lot au moment de sa création, dans ce cas en appelant `IndexBatch.Upload`. Le lot est ensuite chargé dans l'index Recherche cognitive Azure par la méthode `Documents.Index`.

#### NOTE

Dans cet exemple, nous allons simplement charger les documents. Si vous souhaitez fusionner les modifications dans les documents existants ou supprimer des documents, vous pouvez créer des lots en appelant `IndexBatch.Merge`, `IndexBatch.MergeOrUpload` ou `IndexBatch.Delete` à la place. Vous pouvez aussi combiner plusieurs opérations dans un lot unique en appelant `IndexBatch.New`, qui accepte une collection d'objets `IndexAction`, dont chacun indique à Recherche cognitive Azure d'effectuer une opération spécifique sur un document. Vous pouvez créer chaque `IndexAction` avec sa propre opération en appelant la méthode correspondante comme `IndexAction.Merge`, `IndexAction.Upload`, et ainsi de suite.

La troisième partie de cette méthode est un bloc catch qui gère un cas d'erreur important pour l'indexation. Si votre service Recherche cognitive Azure ne parvient pas à indexer certains documents du lot, `Documents.Index` génère un `IndexBatchException`. Cette exception peut se produire si vous indexez des documents lorsque votre service est surchargé. **Nous vous recommandons vivement de prendre en charge explicitement ce cas de figure dans votre code.** Vous pouvez retarder puis relancer l'indexation des documents qui ont échoué, ouvrir une session et continuer comme dans l'exemple, ou faire autre chose selon la cohérence des données requise par votre application.

#### NOTE

Vous pouvez utiliser la méthode `FindFailedActionsToRetry` pour construire un nouveau lot contenant seulement les actions qui ont échoué lors d'un précédent appel à `Index`. Vous trouverez [sur StackOverflow](#) un fil de discussion sur l'utilisation adéquate de cette méthode.

Enfin, la méthode `UploadDocuments` retarde son exécution de deux secondes. L'indexation s'exécutant en mode asynchrone dans votre service Recherche cognitive Azure, l'exemple d'application doit attendre quelque temps afin de s'assurer que les documents sont disponibles pour la recherche. Ce genre de retard n'est nécessaire que dans les démonstrations, les tests et les exemples d'applications.

#### Gestion des documents par le Kit de développement logiciel (SDK) .NET

Vous vous demandez peut-être comment le SDK .NET Recherche cognitive Azure peut charger des instances d'une classe définie par l'utilisateur, comme `Hotel`, dans l'index. Pour répondre à cette question, examinons la classe `Hotel` :

```

using System;
using Microsoft.Azure.Search;
using Microsoft.Azure.Search.Models;
using Microsoft.Spatial;
using Newtonsoft.Json;

public partial class Hotel
{
    [System.ComponentModel.DataAnnotations.Key]
    [IsFilterable]
    public string HotelId { get; set; }

    [IsSearchable, IsSortable]
    public string HotelName { get; set; }

    [IsSearchable]
    [Analyzer(AnalyzerName.AsString.EnLucene)]
    public string Description { get; set; }

    [IsSearchable]
    [Analyzer(AnalyzerName.AsString.FrLucene)]
    [JsonProperty("Description_fr")]
    public string DescriptionFr { get; set; }

    [IsSearchable, IsFilterable, IsSortable, IsFacetable]
    public string Category { get; set; }

    [IsSearchable, IsFilterable, IsFacetable]
    public string[] Tags { get; set; }

    [IsFilterable, IsSortable, IsFacetable]
    public bool? ParkingIncluded { get; set; }

    // SmokingAllowed reflects whether any room in the hotel allows smoking.
    // The JsonIgnore attribute indicates that a field should not be created
    // in the index for this property and it will only be used by code in the client.
    [JsonIgnore]
    public bool? SmokingAllowed => (Rooms != null) ? Array.Exists(Rooms, element =>
element.SmokingAllowed == true) : (bool?)null;

    [IsFilterable, IsSortable, IsFacetable]
    public DateTimeOffset? LastRenovationDate { get; set; }

    [IsFilterable, IsSortable, IsFacetable]
    public double? Rating { get; set; }

    public Address Address { get; set; }

    [IsFilterable, IsSortable]
    public GeographyPoint Location { get; set; }

    public Room[] Rooms { get; set; }
}

```

La première chose à remarquer est que le nom de chaque propriété publique dans la classe `Hotel` mappe à un champ portant le même nom dans la définition d'index. Si vous souhaitez que chaque champ commence par une lettre minuscule (« casse mixte »), vous pouvez demander au Kit de développement logiciel (SDK) de mapper les noms de propriété à une casse mixte automatiquement avec l'attribut `[SerializePropertyNamesAsCamelCase]` sur la classe. Ce scénario est courant dans les applications .NET qui effectuent des liaisons de données, où le schéma cible échappe au contrôle du développeur de l'application, sans devoir violer les directives d'affectation de noms en « casse Pascal » dans .NET.

## NOTE

Le SDK .NET Recherche cognitive Azure utilise la bibliothèque [NewtonSoft JSON.NET](#) pour sérialiser et désérialiser vos objets de modèle personnalisés vers et à partir de JSON. Vous pouvez personnaliser cette sérialisation si nécessaire. Pour plus d'informations, voir [Sérialisation personnalisée avec JSON.NET](#).

La deuxième chose à remarquer est que chaque propriété est assortie d'attributs tels que `IsFilterable`, `IsSearchable`, `Key` et `Analyzer`. Ces attributs sont mappés directement aux [attributs de champ correspondants dans un index de Recherche cognitive Azure](#). La classe `FieldBuilder` utilise ces propriétés pour construire des définitions de champ pour l'index.

La troisième chose importante concernant la classe `Hotel` a trait aux types de données des propriétés publiques. Les types .NET de ces propriétés correspondent à leurs types de champ équivalents dans la définition de l'index. Par exemple, la propriété de chaîne `Category` correspond au champ `category`, qui est de type `Edm.String`. Il existe des mappages de type similaire entre `bool?`, `Edm.Boolean`, `DateTimeOffset?` et `Edm.DateTimeOffset`, etc. Les règles spécifiques pour le mappage de type sont documentées avec la méthode `Documents.Get` dans [l'article de référence sur le Kit de développement logiciel \(SDK\) .NET du service Recherche cognitive Azure](#). La classe `FieldBuilder` effectue ce mappage pour vous, mais il peut toutefois être utile de comprendre son fonctionnement, pour les situations éventuelles de résolution des problèmes de sérialisation.

Avez-vous remarqué la propriété `SmokingAllowed` ?

```
[JsonIgnore]
public bool? SmokingAllowed => (Rooms != null) ? Array.Exists(Rooms, element => element.SmokingAllowed == true) : (bool?)null;
```

L'attribut `JsonIgnore` de cette propriété indique au `FieldBuilder` de ne pas la sérialiser sur l'index en tant que champ. C'est un excellent moyen de créer des propriétés calculées côté client que vous pouvez utiliser comme aides dans votre application. Dans ce cas, la propriété `SmokingAllowed` indique s'il est permis de fumer dans l'une des `Room` de la collection `Rooms`. Si la valeur est false partout, cela signifie qu'il est interdit de fumer dans tout l'hôtel.

Certaines propriétés, telles que `Address` et `Rooms` sont des instances de classes .NET. Ces propriétés représentent des structures de données plus complexes et, par conséquent, nécessitent des champs avec un [type de données complexe](#) dans l'index.

La propriété `Address` représente un ensemble de plusieurs valeurs dans la classe `Address` définie ci-dessous :

```
using System;
using Microsoft.Azure.Search;
using Microsoft.Azure.Search.Models;
using Newtonsoft.Json;

namespace AzureSearch.SDKHowTo
{
    public partial class Address
    {
        [IsSearchable]
        public string StreetAddress { get; set; }

        [IsSearchable, IsFilterable, IsSortable, IsFacetable]
        public string City { get; set; }

        [IsSearchable, IsFilterable, IsSortable, IsFacetable]
        public string StateProvince { get; set; }

        [IsSearchable, IsFilterable, IsSortable, IsFacetable]
        public string PostalCode { get; set; }

        [IsSearchable, IsFilterable, IsSortable, IsFacetable]
        public string Country { get; set; }
    }
}
```

Cette classe contient les valeurs standards utilisées pour décrire les adresses aux États-Unis ou au Canada. Vous pouvez utiliser des types comme celui-ci pour regrouper des champs logiques dans l'index.

La propriété `Rooms` représente une série d'objets `Room` :

```

using System;
using Microsoft.Azure.Search;
using Microsoft.Azure.Search.Models;
using Newtonsoft.Json;

namespace AzureSearch.SDKHowTo
{
    public partial class Room
    {
        [IsSearchable]
        [Analyzer(AnalyzerName.AsString.EnMicrosoft)]
        public string Description { get; set; }

        [IsSearchable]
        [Analyzer(AnalyzerName.AsString.FrMicrosoft)]
        [JsonProperty("Description_fr")]
        public string DescriptionFr { get; set; }

        [IsSearchable, IsFilterable, IsFacetable]
        public string Type { get; set; }

        [IsFilterable, IsFacetable]
        public double? BaseRate { get; set; }

        [IsSearchable, IsFilterable, IsFacetable]
        public string BedOptions { get; set; }

        [IsFilterable, IsFacetable]
        public int SleepsCount { get; set; }

        [IsFilterable, IsFacetable]
        public bool? SmokingAllowed { get; set; }

        [IsSearchable, IsFilterable, IsFacetable]
        public string[] Tags { get; set; }
    }
}

```

Votre modèle de données dans .NET et le schéma d'index qui lui correspond doivent être conçus pour prendre en charge l'expérience de recherche que vous souhaitez offrir à vos utilisateurs finaux. Chaque objet de niveau supérieur dans .NET, c'est-à-dire chaque document dans l'index, correspond à un résultat de recherche qui sera présenté dans votre interface utilisateur. Par exemple, dans une application de recherche d'hôtel, vos utilisateurs peuvent rechercher par nom d'hôtel, caractéristiques d'hôtel, ou caractéristiques de chambres spécifiques. Nous examinerons quelques exemples de requête un peu plus tard.

Cette capacité à utiliser vos propres classes pour interagir avec des documents dans l'index fonctionne dans les deux sens. Vous pouvez également récupérer les résultats de la recherche et laisser le Kit de développement logiciel (SDK) les déserialiser automatiquement à un type de votre choix, comme nous le verrons dans la section suivante.

#### **NOTE**

Le SDK .NET Recherche cognitive Azure prend également en charge les documents dynamiquement typés à l'aide de la classe `Document`, qui est un mappage de type clé/valeur entre des noms de champ et des valeurs de champ. Cela est utile dans les cas où vous ne connaissez pas le schéma de l'index lors de sa conception et où il serait peu pratique d'établir une liaison à des classes de modèles spécifiques. Toutes les méthodes du SDK qui gèrent les documents ont des surcharges qui fonctionnent avec la classe `Document`, ainsi que des surcharges fortement typées qui acceptent un paramètre de type générique. Seules ces dernières sont utilisées dans l'exemple de code de ce didacticiel. La classe `Document` hérite de `Dictionary<string, object>`.

## Pourquoi utiliser des types de données Nullable

Lorsque vous créez vos propres classes de modèles à mapper à un index Recherche cognitive Azure, nous vous recommandons de déclarer les propriétés des types de valeurs, par exemple `bool` et `int`, comme acceptant la valeur null (par exemple, `bool?` au lieu de `bool`). Si vous utilisez une propriété ne pouvant être définie sur null, vous devez garantir qu'aucun document de cet index ne contient de valeur null pour le champ correspondant. Ni le Kit de développement logiciel ni le service Recherche cognitive Azure ne vous aideront à appliquer cette recommandation.

Il ne s'agit pas d'une préoccupation hypothétique : imaginez un scénario dans lequel vous ajoutez un nouveau champ à un index existant qui est de type `Edm.Int32`. Après la mise à jour de la définition d'index, ce nouveau champ prendra la valeur null pour tous les documents (car tous les types peuvent avoir la valeur null dans Recherche cognitive Azure). Si vous utilisez ensuite une classe de modèle avec une propriété `int` ne pouvant être définie sur null pour ce champ, vous obtiendrez l'exception `JsonSerializationException` ci-dessous lorsque vous tenterez de récupérer des documents :

```
Error converting value {null} to type 'System.Int32'. Path 'IntValue'.
```

Pour cette raison, nous vous recommandons d'utiliser des types pour lesquels la valeur null est autorisée en tant que meilleure pratique.

## Sérialisation personnalisée avec JSON.NET

Le kit de développement logiciel utilise JSON.NET pour sérialiser et désérialiser les documents. Si nécessaire, vous pouvez personnaliser la sérialisation et la désérialisation en définissant votre propre `JsonConverter` ou `IContractResolver`. Pour plus d'informations, voir la [documentation sur JSON.NET](#). Cela peut s'avérer utile lorsque vous souhaitez adapter une classe de modèle existante de votre application à utiliser avec Recherche cognitive Azure et d'autres scénarios plus avancés. Par exemple, avec la sérialisation personnalisée, vous pouvez :

- Incluez ou exclure certaines propriétés de votre classe de modèle dans le stockage en tant que champs de document.
- Mappez des noms de propriété dans le code et des noms de champ de votre index.
- Créez des attributs personnalisés qui peuvent être utilisés pour le mappage des propriétés aux champs du document.

Vous pouvez trouver des exemples d'implémentation de sérialisation personnalisée dans les tests d'unités du kit de développement logiciel .NET Recherche cognitive Azure sur GitHub. [Ce dossier](#) est un bon point de départ. Il contient des classes qui sont utilisées par les tests de sérialisation personnalisés.

## Recherche de documents dans l'index

La dernière étape dans l'exemple d'application consiste à rechercher certains documents dans l'index :

```

private static void RunQueries(ISearchIndexClient indexClient)
{
    SearchParameters parameters;
    DocumentSearchResult<Hotel> results;

    Console.WriteLine("Search the entire index for the term 'motel' and return only the HotelName field:\n");

    parameters =
        new SearchParameters()
    {
        Select = new[] { "HotelName" }
    };

    results = indexClient.Documents.Search<Hotel>("motel", parameters);

    WriteDocuments(results);

    Console.Write("Apply a filter to the index to find hotels with a room cheaper than $100 per night,\n");
    Console.WriteLine("and return the hotelId and description:\n");

    parameters =
        new SearchParameters()
    {
        Filter = "Rooms/any(r: r/BaseRate lt 100)",
        Select = new[] { "HotelId", "Description" }
    };

    results = indexClient.Documents.Search<Hotel>("*", parameters);

    WriteDocuments(results);

    Console.Write("Search the entire index, order by a specific field (lastRenovationDate )\n");
    Console.Write("in descending order, take the top two results, and show only hotelName and " );
    Console.WriteLine("lastRenovationDate:\n");

    parameters =
        new SearchParameters()
    {
        OrderBy = new[] { "LastRenovationDate desc" },
        Select = new[] { "HotelName", "LastRenovationDate" },
        Top = 2
    };

    results = indexClient.Documents.Search<Hotel>("*", parameters);

    WriteDocuments(results);

    Console.WriteLine("Search the entire index for the term 'hotel':\n");

    parameters = new SearchParameters();
    results = indexClient.Documents.Search<Hotel>("hotel", parameters);

    WriteDocuments(results);
}

```

Chaque fois qu'elle exécute une requête, cette méthode crée tout d'abord un objet `SearchParameters`. Cet objet permet de spécifier des options supplémentaires pour la requête, comme le tri, le filtrage, la pagination et la génération de facettes. Dans cette méthode, nous définissons les propriétés `Filter`, `Select`, `OrderBy`, et `Top` pour différentes requêtes. Toutes les propriétés `SearchParameters` sont documentées [ici](#).

L'étape suivante consiste à exécuter la requête de recherche. La recherche est exécutée à l'aide de la méthode `Documents.Search`. Pour chaque requête, nous transmettons le texte de recherche à utiliser en tant que chaîne

(ou l'élément `"*"`, s'il n'en existe aucun), ainsi que les paramètres de recherche créés précédemment. Nous spécifions également `Hotel` comme paramètre de type pour `Documents.Search`, qui demande au SDK de déserialiser les documents figurant dans les résultats de recherche, en objets de type `Hotel`.

#### NOTE

Pour plus d'informations sur la syntaxe des requêtes de recherche, cliquez [ici](#).

Enfin, après l'exécution de chaque requête, cette méthode parcourt toutes les correspondances dans les résultats de la recherche et imprime chaque document dans la console :

```
private static void WriteDocuments(DocumentSearchResult<Hotel> searchResults)
{
    foreach (SearchResult<Hotel> result in searchResults.Results)
    {
        Console.WriteLine(result.Document);
    }

    Console.WriteLine();
}
```

Examinons de plus près chacune des requêtes exécutées. Voici le code permettant d'exécuter la première requête :

```
parameters =
    new SearchParameters()
    {
        Select = new[] { "HotelName" }
    };

results = indexClient.Documents.Search<Hotel>("motel", parameters);

WriteDocuments(results);
```

Dans ce cas, nous recherchons dans l'index entier le mot « motel », dans tout champ pouvant faire l'objet d'une recherche, et ne voulons récupérer que les noms d'hôtel, comme spécifié par le paramètre `Select`. Voici les résultats :

```
Name: Secret Point Motel
```

```
Name: Twin Dome Motel
```

La requête suivante est un peu plus intéressante. Nous souhaitons trouver tous les hôtels proposant une chambre à moins de 100 \$ la nuitée, et obtenir uniquement l'ID et la description des hôtels :

```
parameters =
    new SearchParameters()
    {
        Filter = "Rooms/any(r: r/BaseRate lt 100)",
        Select = new[] { "HotelId", "Description" }
    };

results = indexClient.Documents.Search<Hotel>("*", parameters);

WriteDocuments(results);
```

Cette requête utilise une expression `$filter` OData (`Rooms/any(r: r/BaseRate lt 100)`) pour filtrer les documents dans l'index. Cet exemple utilise l'[opérateur any](#) pour appliquer le « BaseRate lt 100 » à chaque élément de la collection Rooms (Chambres). Pour plus d'informations sur la syntaxe OData prise en charge par Recherche cognitive Azure, cliquez [ici](#).

Voici les résultats de la requête :

```
HotelId: 1
Description: The hotel is ideally located on the main commercial artery of the city in the heart of New York...

HotelId: 2
Description: The hotel is situated in a nineteenth century plaza, which has been expanded and renovated to...
```

Ensuite, nous souhaitons rechercher les deux hôtels ayant été le plus récemment rénovés, et afficher le nom et la date de la dernière rénovation de ces derniers. Voici le code :

```
parameters =
    new SearchParameters()
{
    OrderBy = new[] { "LastRenovationDate desc" },
    Select = new[] { "HotelName", "LastRenovationDate" },
    Top = 2
};

results = indexClient.Documents.Search<Hotel>("*", parameters);

WriteDocuments(results);
```

Dans ce cas, nous utilisons à nouveau la syntaxe OData pour spécifier le paramètre `OrderBy` en tant que `lastRenovationDate desc`. Nous définissons également le paramètre `Top` sur 2 pour vérifier que nous obtenons uniquement les deux premiers documents. Comme précédemment, nous définissons le paramètre `Select` pour spécifier les champs qui doivent être renvoyés.

Voici les résultats :

```
Name: Fancy Stay      Last renovated on: 6/27/2010 12:00:00 AM +00:00
Name: Roach Motel     Last renovated on: 4/28/1982 12:00:00 AM +00:00
```

Enfin, nous souhaitons trouver tous les noms d'hôtel contenant le mot « hotel » :

```
parameters = new SearchParameters()
{
    SearchFields = new[] { "HotelName" }
};
results = indexClient.Documents.Search<Hotel>("hotel", parameters);

WriteDocuments(results);
```

Voici les résultats, qui incluent tous les champs, dans la mesure où nous n'avons pas spécifié la propriété `Select` :

```
HotelId: 3
Name: Triple Landscape Hotel
...
```

Cette étape termine le didacticiel, mais ne vous arrêtez pas en si bon chemin. Les étapes suivantes fournissent des ressources supplémentaires pour en apprendre davantage sur Recherche cognitive Azure.

## Étapes suivantes

- Parcourez les références relatives au [Kit de développement logiciel \(SDK\) .NET](#) et à [l'API REST](#).
- Consultez les [conventions d'affectation de noms](#) pour apprendre les règles de dénomination des différents objets.
- Faites connaissance avec les [types de données pris en charge](#) par Recherche cognitive Azure.

# Démarrage rapide : Créer un index de recherche à l'aide de la bibliothèque de client Microsoft.Azure.Search v10

04/10/2020 • 28 minutes to read • [Edit Online](#)

Cet article est le démarrage rapide C# pour la bibliothèque de client Microsoft.Azure.Search (version 10) héritée, désormais remplacée par la bibliothèque de client Azure.Search.Documents (version 11). Si vous avez des solutions de recherche existantes qui utilisent les bibliothèques Microsoft.Azure.Search, vous pouvez utiliser ce démarrage rapide pour en savoir plus sur ces API.

Pour les nouvelles solutions, nous vous recommandons d'utiliser la nouvelle bibliothèque Azure.Search.Documents. Pour une introduction, consultez [Démarrage rapide : Créer un index de recherche à l'aide de la bibliothèque Azure.Search.Documents](#).

## À propos de ce démarrage rapide

Générez une application console .NET Core en C#, qui crée, charge et interroge un index Recherche cognitive Azure à l'aide de Visual Studio et des [bibliothèques de client Microsoft.Azure.Search](#).

Cet article explique comment créer l'application. Vous pouvez aussi [télécharger et exécuter l'application complète](#).

### NOTE

Le code de démonstration présent dans cet article utilise les méthodes synchrones du Kit de développement logiciel (SDK) .NET Recherche cognitive Azure version 10 par simplicité. Toutefois, pour les scénarios de production, nous recommandons d'utiliser les méthodes asynchrones dans vos propres applications pour les rendre scalables et réactives. Par exemple, vous pouvez utiliser `CreateAsync` et `DeleteAsync` au lieu de `Create` et `Delete`.

## Prérequis

Avant de commencer la lecture cet article, vous devez disposer des éléments suivants :

- Compte Azure avec un abonnement actif. [Créez un compte gratuitement](#).
- Service Recherche cognitive Azure. [Créez un service](#) ou [recherchez un service existant](#) dans votre abonnement actuel. Vous pouvez utiliser un service gratuit pour ce guide de démarrage rapide.
- [Visual Studio](#), toute édition. L'exemple de code et les instructions ont été testés dans l'édition Communauté gratuite.

## Obtenir une clé et une URL

Les appels au service nécessitent un point de terminaison d'URL et une clé d'accès pour chaque requête. Un service de recherche est créé avec les deux. Ainsi, si vous avez ajouté la Recherche cognitive Azure à votre abonnement, effectuez ce qui suit pour obtenir les informations nécessaires :

1. [Connectez-vous au portail Azure](#), puis dans la page **Vue d'ensemble** du service de recherche, récupérez l'URL. Voici un exemple de point de terminaison : `https://mydemo.search.windows.net`.
2. Dans **Paramètres > Clés**, obtenez une clé d'administration pour avoir des droits d'accès complets sur le

service. Il existe deux clés d'administration interchangeables, fournies pour assurer la continuité de l'activité au cas où vous deviez en remplacer une. Vous pouvez utiliser la clé primaire ou secondaire sur les demandes d'ajout, de modification et de suppression d'objets.

Obtenez aussi la clé de requête. Il est recommandé d'émettre des demandes de requête avec un accès en lecture seule.

The screenshot shows two overlapping Azure portal pages. The top page is titled 'mydemo' and displays a 'Vue d'ensemble' card. A red box highlights the 'Vue d'ensemble' card, and a red circle with the number '1' highlights the 'URL' field which contains 'https://mydemo.search.windows.net'. The bottom page is titled 'mydemo - Clés' and shows a 'Clés' card. A red box highlights the 'Clés' card, and a red circle with the number '2' highlights the 'Clé d'administration primaire' input field, which contains '<placeholder-for-alphanumeric-autogenerated-string>'.

Toutes les demandes nécessitent une clé API sur chaque demande envoyée à votre service. L'utilisation d'une clé valide permet d'établir, en fonction de chaque demande, une relation de confiance entre l'application qui envoie la demande et le service qui en assure le traitement.

## Configurer votre environnement

Commencez par ouvrir Visual Studio et créer un nouveau projet d'application console pouvant s'exécuter sur .NET Core.

### Installer les packages NuGet

Le [package Microsoft.Azure.Search](#) est constitué de quelques bibliothèques de client distribuées comme packages NuGet.

Pour ce projet, utilisez la version 10 du package NuGet `Microsoft.Azure.Search` ainsi que le dernier package NuGet `Microsoft.Extensions.Configuration.Json`.

1. Dans Outils > Gestionnaire de package NuGet, sélectionnez Gérer les packages NuGet pour la solution... .
2. Cliquez sur Parcourir.
3. Recherchez `Microsoft.Azure.Search` et sélectionnez la version 10.
4. Cliquez sur Installer à droite pour ajouter l'assembly à vos projet et solution.
5. Répétez l'opération pour `Microsoft.Extensions.Configuration.Json`, en sélectionnant la version 2.2.0 ou ultérieure.

### Ajouter des informations relatives au service Recherche cognitive Azure

1. Dans l'Explorateur de solutions, cliquez avec le bouton droit sur le projet, puis sélectionnez Ajouter > Nouvel élément... .

2. Dans Ajouter un nouvel élément, recherchez « JSON » pour retourner une liste liée à JSON de types d'éléments.
3. Choisissez **Fichier JSON**, nommez le fichier « appsettings.json », puis cliquez sur **Ajouter**.
4. Ajoutez le fichier à votre répertoire de sortie. Cliquez avec le bouton droit sur appsettings.json et sélectionnez **Propriétés**. Dans **Copier dans le répertoire de sortie**, sélectionnez **Copier si plus récent**.
5. Copiez le code JSON ci-dessous dans votre nouveau fichier JSON.

```
{
  "SearchServiceName": "<YOUR-SEARCH-SERVICE-NAME>",
  "SearchServiceAdminApiKey": "<YOUR-ADMIN-API-KEY>",
  "SearchIndexName": "hotels-quickstart"
}
```

6. Remplacez le nom du service de recherche (YOUR-SEARCH-SERVICE-NAME) et la clé API d'administration (YOUR-ADMIN-API-KEY) par des valeurs valides. Si le point de terminaison de votre service est <https://mydemo.search.windows.net>, le nom du service serait « `mydemo` ».

#### **Ajouter les fichiers de classe « .Method » à votre projet**

Cette étape est requise pour produire une sortie significative dans la console. Lors de l'impression des résultats dans la fenêtre de console, des champs individuels à partir de l'objet Hotel doivent être retournés sous forme de chaînes. Cette étape permet d'implémenter [ToString\(\)](#) pour effectuer cette tâche, en copiant le code nécessaire dans deux nouveaux fichiers.

1. Ajoutez deux définitions de classe vides à votre projet : Address.Methods.cs, Hotel.Methods.cs
2. Dans Address.Methods.cs, remplacez le contenu par défaut par le code suivant, [lignes 1 à 25](#).
3. Dans Hotel.Methods.cs, copiez les [lignes 1 à 68](#).

## 1 - Créer un index

L'index des hôtels se compose de champs simples et complexes, où un champ simple est « HotelName » ou « Description », et les champs complexes sont une adresse avec des sous-champs, ou une collection de chambres. Lorsqu'un index inclut des types complexes, isolez les définitions de champ complexes dans des classes distinctes.

1. Ajoutez deux définitions de classe vides à votre projet : Address.cs, Hotel.cs
2. Dans Address.cs, remplacez le contenu par défaut par le code suivant :

```
using System;
using Microsoft.Azure.Search;
using Microsoft.Azure.Search.Models;
using Newtonsoft.Json;

namespace AzureSearchQuickstart
{
    public partial class Address
    {
        [IsSearchable]
        public string StreetAddress { get; set; }

        [IsSearchable, IsFilterable, IsSortable, IsFacetable]
        public string City { get; set; }

        [IsSearchable, IsFilterable, IsSortable, IsFacetable]
        public string StateProvince { get; set; }

        [IsSearchable, IsFilterable, IsSortable, IsFacetable]
        public string PostalCode { get; set; }

        [IsSearchable, IsFilterable, IsSortable, IsFacetable]
        public string Country { get; set; }
    }
}
```

3. Dans Hotel.cs, la classe définit la structure globale de l'index, notamment des références à la classe d'adresses.

```

namespace AzureSearchQuickstart
{
    using System;
    using Microsoft.Azure.Search;
    using Microsoft.Azure.Search.Models;
    using Newtonsoft.Json;

    public partial class Hotel
    {
        [System.ComponentModel.DataAnnotations.Key]
        [IsFilterable]
        public string HotelId { get; set; }

        [IsSearchable, IsSortable]
        public string HotelName { get; set; }

        [IsSearchable]
        [Analyzer(AnalyzerNameAsString.EnMicrosoft)]
        public string Description { get; set; }

        [IsSearchable]
        [Analyzer(AnalyzerNameAsString.FrLucene)]
        [JsonProperty("Description_fr")]
        public string DescriptionFr { get; set; }

        [IsSearchable, IsFilterable, IsSortable, IsFacetable]
        public string Category { get; set; }

        [IsSearchable, IsFilterable, IsFacetable]
        public string[] Tags { get; set; }

        [IsFilterable, IsSortable, IsFacetable]
        public bool? ParkingIncluded { get; set; }

        [IsFilterable, IsSortable, IsFacetable]
        public DateTimeOffset? LastRenovationDate { get; set; }

        [IsFilterable, IsSortable, IsFacetable]
        public double? Rating { get; set; }

        public Address Address { get; set; }
    }
}

```

Les attributs du champ déterminent la façon dont il est utilisé dans une application. Par exemple, l'attribut `IsSearchable` doit être affecté à tous les champs qui doivent être inclus dans une recherche en texte intégral.

#### NOTE

Dans le kit de développement logiciel (SDK) .NET, les champs doivent être explicitement attribués en tant que `IsSearchable`, `IsFilterable`, `IsSortable` et `IsFacetable`. Ce comportement contraste avec l'API REST qui active implicitement l'attribution en fonction du type de données (par exemple, les champs de chaîne simple peuvent automatiquement faire l'objet d'une recherche).

Dans votre index, un seul champ de type `string` doit être désigné en tant que champ de *clé* qui identifie de façon unique chaque document. Dans ce schéma, la clé est `HotelId`.

Dans cet index, les champs de description utilisent la propriété `analyzer` facultative, spécifiée lorsque vous souhaitez remplacer l'analyseur Lucene standard par défaut. Le champ `description_fr` utilise l'analyseur Lucene en français (`FrLucene`), car il stocke le texte en français. Le champ `description` utilise l'analyseur de

langue Microsoft facultatif ([EnMicrosoft](#)).

4. Dans Program.cs, créez une instance de la classe `SearchServiceClient` pour vous connecter au service, à l'aide de valeurs qui sont stockées dans le fichier de configuration de l'application (appsettings.json).

`SearchServiceClient` a une propriété `Indexes`, qui fournit toutes les méthodes dont vous avez besoin pour créer, lister, mettre à jour ou supprimer des index Recherche cognitive Azure.

```
using System;
using System.Linq;
using System.Threading;
using Microsoft.Azure.Search;
using Microsoft.Azure.Search.Models;
using Microsoft.Extensions.Configuration;

namespace AzureSearchQuickstart
{
    class Program {
        // Demonstrates index delete, create, load, and query
        // Commented-out code is uncommented in later steps
        static void Main(string[] args)
        {
            IConfigurationBuilder builder = new ConfigurationBuilder().AddJsonFile("appsettings.json");
            IConfigurationRoot configuration = builder.Build();

            SearchServiceClient serviceClient = CreateSearchServiceClient(configuration);

            string indexName = configuration["SearchIndexName"];

            Console.WriteLine("{0}", "Deleting index...\n");
            DeleteIndexIfExists(indexName, serviceClient);

            Console.WriteLine("{0}", "Creating index...\n");
            CreateIndex(indexName, serviceClient);

            // Uncomment next 3 lines in "2 - Load documents"
            // ISearchIndexClient indexClient = serviceClient.Indexes.GetClient(indexName);
            // Console.WriteLine("{0}", "Uploading documents...\n");
            // UploadDocuments(indexClient);

            // Uncomment next 2 lines in "3 - Search an index"
            // Console.WriteLine("{0}", "Searching index...\n");
            // RunQueries(indexClient);

            Console.WriteLine("{0}", "Complete. Press any key to end application...\n");
            Console.ReadKey();
        }

        // Create the search service client
        private static SearchServiceClient CreateSearchServiceClient(IConfigurationRoot configuration)
        {
            string searchServiceName = configuration["SearchServiceName"];
            string adminApiKey = configuration["SearchServiceAdminApiKey"];

            SearchServiceClient serviceClient = new SearchServiceClient(searchServiceName, new
SearchCredentials(adminApiKey));
            return serviceClient;
        }

        // Delete an existing index to reuse its name
        private static void DeleteIndexIfExists(string indexName, SearchServiceClient serviceClient)
        {
            if (serviceClient.Indexes.Exists(indexName))
            {
                serviceClient.Indexes.Delete(indexName);
            }
        }
    }
}
```

```

    // Create an index whose fields correspond to the properties of the Hotel class.
    // The Address property of Hotel will be modeled as a complex field.
    // The properties of the Address class in turn correspond to sub-fields of the Address complex
    field.
    // The fields of the index are defined by calling the FieldBuilder.BuildForType() method.
    private static void CreateIndex(string indexName, SearchServiceClient serviceClient)
    {
        var definition = new Microsoft.Azure.Search.Models.Index()
        {
            Name = indexName,
            Fields = FieldBuilder.BuildForType<Hotel>()
        };

        serviceClient.Indexes.Create(definition);
    }
}

```

Si possible, partagez une seule instance de `SearchServiceClient` dans votre application pour éviter l'ouverture d'un trop grand nombre de connexions. Les méthodes de classe sont thread-safe pour permettre ce partage.

La classe dispose de plusieurs constructeurs. Celle que vous cherchez utilise le nom de votre service de recherche et un objet `SearchCredentials` en tant paramètres. `SearchCredentials` encapsule votre clé API.

Dans la définition d'index, le moyen le plus simple de créer les objets `Field` consiste à appeler la méthode `FieldBuilder.BuildForType` en transmettant une classe de modèle pour le paramètre de type. Une classe de modèle comporte des propriétés qui sont mappées sur les champs de votre index. Ce mappage vous permet de lier des documents à partir de votre index de recherche à des instances de votre classe de modèle.

#### NOTE

Si vous n'envisagez pas d'utiliser une classe de modèle, vous pouvez toujours définir votre index en créant les objets `Field` directement. Vous pouvez fournir le nom du champ au constructeur, ainsi que le type de données (ou un analyseur pour les champs de chaîne). Vous pouvez également définir d'autres propriétés comme `IsSearchable`, `IsFilterable`, etc.

- Appuyez sur F5 pour générer l'application et créer l'index.

Si le projet est correctement généré, une fenêtre de console s'ouvre, en affichant des messages d'état à l'écran pour la suppression et la création de l'index.

## 2 – Charger des documents

Dans la Recherche cognitive Azure, les documents sont des structures de données qui sont à la fois des entrées pour l'indexation et des sorties de requêtes. Selon une source de données externe, les entrées de documents peuvent être des lignes dans une base de données, des objets blob dans le Stockage Blob ou des documents JSON sur le disque. Dans cet exemple, nous prenons un raccourci et incorporons des documents JSON pour quatre hôtels dans le code lui-même.

Lors du chargement de documents, vous devez utiliser un objet `IndexBatch`. `IndexBatch` contient une collection d'objets `IndexAction`, chacun contenant un document et une propriété qui indiquent à la Recherche cognitive Azure l'action à effectuer (`upload`, `merge`, `delete` et `mergeOrUpload`).

- Dans Program.cs, créez un tableau des documents et des actions d'index, puis transmettez le tableau à `IndexBatch`. Les documents ci-dessous sont conformes à l'index hotels-quickstart, tel que défini par les

classes hotel et address.

```
// Upload documents as a batch
private static void UploadDocuments(ISearchIndexClient indexClient)
{
    var actions = new IndexAction<Hotel>[]
    {
        IndexAction.Upload(
            new Hotel()
            {
                HotelId = "1",
                HotelName = "Secret Point Motel",
                Description = "The hotel is ideally located on the main commercial artery of the city in the heart of New York. A few minutes away is Time's Square and the historic centre of the city, as well as other places of interest that make New York one of America's most attractive and cosmopolitan cities.",
                DescriptionFr = "L'hôtel est idéalement situé sur la principale artère commerciale de la ville en plein cœur de New York. A quelques minutes se trouve la place du temps et le centre historique de la ville, ainsi que d'autres lieux d'intérêt qui font de New York l'une des villes les plus attractives et cosmopolites de l'Amérique.",
                Category = "Boutique",
                Tags = new[] { "pool", "air conditioning", "concierge" },
                ParkingIncluded = false,
                LastRenovationDate = new DateTimeOffset(1970, 1, 18, 0, 0, TimeSpan.Zero),
                Rating = 3.6,
                Address = new Address()
                {
                    StreetAddress = "677 5th Ave",
                    City = "New York",
                    StateProvince = "NY",
                    PostalCode = "10022",
                    Country = "USA"
                }
            },
        ),
        IndexAction.Upload(
            new Hotel()
            {
                HotelId = "2",
                HotelName = "Twin Dome Motel",
                Description = "The hotel is situated in a nineteenth century plaza, which has been expanded and renovated to the highest architectural standards to create a modern, functional and first-class hotel in which art and unique historical elements coexist with the most modern comforts.",
                DescriptionFr = "L'hôtel est situé dans une place du XIXe siècle, qui a été agrandie et rénovée aux plus hautes normes architecturales pour créer un hôtel moderne, fonctionnel et de première classe dans lequel l'art et les éléments historiques uniques coexistent avec le confort le plus moderne.",
                Category = "Boutique",
                Tags = new[] { "pool", "free wifi", "concierge" },
                ParkingIncluded = false,
                LastRenovationDate = new DateTimeOffset(1979, 2, 18, 0, 0, TimeSpan.Zero),
                Rating = 3.60,
                Address = new Address()
                {
                    StreetAddress = "140 University Town Center Dr",
                    City = "Sarasota",
                    StateProvince = "FL",
                    PostalCode = "34243",
                    Country = "USA"
                }
            },
        ),
        IndexAction.Upload(
            new Hotel()
            {
                HotelId = "3",
                HotelName = "Triple Landscape Hotel",
                Description = "The Hotel stands out for its gastronomic excellence under the management"
            }
        )
    }
}
```

```

of William Dough, who advises on and oversees all of the Hotel's restaurant services.",

    DescriptionFr = "L'hôtel est situé dans une place du XIXe siècle, qui a été agrandie et
rénovée aux plus hautes normes architecturales pour créer un hôtel moderne, fonctionnel et de première
classe dans lequel l'art et les éléments historiques uniques coexistent avec le confort le plus
moderne.",

        Category = "Resort and Spa",
        Tags = new[] { "air conditioning", "bar", "continental breakfast" },
        ParkingIncluded = true,
        LastRenovationDate = new DateTimeOffset(2015, 9, 20, 0, 0, TimeSpan.Zero),
        Rating = 4.80,
        Address = new Address()
    {
        StreetAddress = "3393 Peachtree Rd",
        City = "Atlanta",
        StateProvince = "GA",
        PostalCode = "30326",
        Country = "USA"
    }
}

),
IndexAction.Upload(
    new Hotel()
{
    HotelId = "4",
    HotelName = "Sublime Cliff Hotel",
    Description = "Sublime Cliff Hotel is located in the heart of the historic center of
Sublime in an extremely vibrant and lively area within short walking distance to the sites and
landmarks of the city and is surrounded by the extraordinary beauty of churches, buildings, shops and
monuments. Sublime Cliff is part of a lovingly restored 1800 palace.",
    DescriptionFr = "Le sublime Cliff Hotel est situé au coeur du centre historique de
sublime dans un quartier extrêmement animé et vivant, à courte distance de marche des sites et
monuments de la ville et est entouré par l'extraordinaire beauté des églises, des bâtiments, des
commerces et Monuments. Sublime Cliff fait partie d'un Palace 1800 restauré avec amour.",
    Category = "Boutique",
    Tags = new[] { "concierge", "view", "24-hour front desk service" },
    ParkingIncluded = true,
    LastRenovationDate = new DateTimeOffset(1960, 2, 06, 0, 0, TimeSpan.Zero),
    Rating = 4.6,
    Address = new Address()
    {
        StreetAddress = "7400 San Pedro Ave",
        City = "San Antonio",
        StateProvince = "TX",
        PostalCode = "78216",
        Country = "USA"
    }
}
),
};

var batch = IndexBatch.New(actions);

try
{
    indexClient.Documents.Index(batch);
}
catch (IndexBatchException e)
{
    // When a service is under load, indexing might fail for some documents in the batch.
    // Depending on your application, you can compensate by delaying and retrying.
    // For this simple demo, we just log the failed document keys and continue.
    Console.WriteLine(
        "Failed to index some of the documents: {0}",
        String.Join(", ", e.IndexingResults.Where(r => !r.Succeeded).Select(r => r.Key)));
}

// Wait 2 seconds before starting queries
Console.WriteLine("Waiting for indexing... \n");
Thread.Sleep(2000);

```

```
}
```

Une fois que vous avez initialisé l'objet `IndexBatch`, vous pouvez l'envoyer à l'index en appelant `Documents.Index` sur votre objet `SearchIndexClient`. `Documents` est une propriété de `SearchIndexClient` qui fournit des méthodes pour l'ajout, la modification, la suppression ou l'interrogation de documents dans votre index.

L'élément `try / catch` entourant l'appel à la méthode `Index` intercepte les échecs d'indexation, qui peuvent se produire si votre service est surchargé. Dans le code de production, vous pouvez retarder, puis relancer l'indexation des documents qui ont échoué, ouvrir une session et continuer comme dans l'exemple, ou gérer la situation d'une autre manière qui répond à la cohérence des données requise par votre application.

Le retard de 2 secondes compense l'indexation, qui est asynchrone, afin que tous les documents puissent être indexés avant l'exécution des requêtes. Le codage dans un retard n'est nécessaire que dans les démonstrations, les tests et les exemples d'applications.

2. Dans Program.cs, dans main, supprimez les commentaires des lignes pour « 2 - Charger des documents ».

```
// Uncomment next 3 lines in "2 - Load documents"
ISearchIndexClient indexClient = serviceClient.Indexes.GetClient(indexName);
Console.WriteLine("{0}", "Uploading documents...\n");
UploadDocuments(indexClient);
```

3. Appuyez sur F5 pour régénérer l'application.

Si le projet est correctement généré, une fenêtre de console s'ouvre, en affichant des messages d'état, cette fois sur le chargement des documents. Dans le portail Azure, dans la page **Vue d'ensemble** du service de recherche, l'index hotels-quickstart doit maintenant afficher 4 documents.

Pour plus d'informations sur le traitement des documents, consultez « [Comment le SDK .NET traite les documents](#) ».

### 3 – Rechercher dans un index

Vous pouvez obtenir les résultats de la requête dès que le premier document est indexé, mais les tests réels de votre index doivent attendre que tous les documents soient indexés.

Cette section ajoute deux éléments de fonctionnalité : logique de requête et résultats. Pour les requêtes, utilisez la méthode `Search`. Cette méthode accepte le texte recherché ainsi que d'autres **paramètres**.

La classe `DocumentsSearchResult` représente les résultats.

1. Dans Program.cs, créez une méthode `WriteDocuments` qui imprime les résultats de la recherche dans la console.

```
private static void WriteDocuments(DocumentSearchResult<Hotel> searchResults)
{
    foreach (SearchResult<Hotel> result in searchResults.Results)
    {
        Console.WriteLine(result.Document);
    }

    Console.WriteLine();
}
```

2. Créez une méthode `RunQueries` pour exécuter des requêtes et retourner des résultats. Les résultats sont des

objets Hotel. Vous pouvez utiliser le paramètre select pour présenter les champs individuels. Si un champ n'est pas inclus dans le paramètre select, sa propriété Hotel correspondante aura la valeur Null.

```
private static void RunQueries(ISearchIndexClient indexClient)
{
    SearchParameters parameters;
    DocumentSearchResult<Hotel> results;

    // Query 1
    Console.WriteLine("Query 1: Search for term 'Atlanta' with no result trimming");
    parameters = new SearchParameters();
    results = indexClient.Documents.Search<Hotel>("Atlanta", parameters);
    WriteDocuments(results);

    // Query 2
    Console.WriteLine("Query 2: Search on the term 'Atlanta', with trimming");
    Console.WriteLine("Returning only these fields: HotelName, Tags, Address:\n");
    parameters =
        new SearchParameters()
    {
        Select = new[] { "HotelName", "Tags", "Address" },
    };
    results = indexClient.Documents.Search<Hotel>("Atlanta", parameters);
    WriteDocuments(results);

    // Query 3
    Console.WriteLine("Query 3: Search for the terms 'restaurant' and 'wifi'");
    Console.WriteLine("Return only these fields: HotelName, Description, and Tags:\n");
    parameters =
        new SearchParameters()
    {
        Select = new[] { "HotelName", "Description", "Tags" }
    };
    results = indexClient.Documents.Search<Hotel>("restaurant, wifi", parameters);
    WriteDocuments(results);

    // Query 4 -filtered query
    Console.WriteLine("Query 4: Filter on ratings greater than 4");
    Console.WriteLine("Returning only these fields: HotelName, Rating:\n");
    parameters =
        new SearchParameters()
    {
        Filter = "Rating gt 4",
        Select = new[] { "HotelName", "Rating" }
    };
    results = indexClient.Documents.Search<Hotel>("*", parameters);
    WriteDocuments(results);

    // Query 5 - top 2 results
    Console.WriteLine("Query 5: Search on term 'boutique'");
    Console.WriteLine("Sort by rating in descending order, taking the top two results");
    Console.WriteLine("Returning only these fields: HotelId, HotelName, Category, Rating:\n");
    parameters =
        new SearchParameters()
    {
        OrderBy = new[] { "Rating desc" },
        Select = new[] { "HotelId", "HotelName", "Category", "Rating" },
        Top = 2
    };
    results = indexClient.Documents.Search<Hotel>("boutique", parameters);
    WriteDocuments(results);
}
```

Il existe deux façons de mettre en correspondance des termes dans une requête : la recherche en texte intégral et les filtres. Une requête de recherche en texte intégral recherche un ou plusieurs termes dans les

champs `IsSearchable` de votre index. Un filtre est une expression booléenne évaluée sur les champs `IsFilterable` dans un index. Vous pouvez utiliser la recherche en texte intégral et les filtres conjointement ou séparément.

Les filtres et les recherches sont exécutés à l'aide de la méthode `Documents.Search`. Une requête de recherche peut être transmise dans le paramètre `searchText`, tandis qu'une expression de filtre peut être transmise dans la propriété `Filter` de la classe `SearchParameters`. Pour filtrer sans effectuer de recherche, transmettez simplement `"*"` pour le paramètre `searchText`. Pour effectuer une recherche sans appliquer de filtre, ne définissez pas la propriété `Filter` et ne transmettez pas une instance `SearchParameters`.

3. Dans Program.cs, dans main, supprimez les commentaires des lignes pour « 3 - Rechercher dans un index ».

```
// Uncomment next 2 lines in "3 - Search an index"
Console.WriteLine("{0}", "Searching documents...\n");
RunQueries(indexClient);
```

4. La solution est maintenant terminée. Appuyez sur F5 pour régénérer l'application et exécuter le programme dans son intégralité.

La sortie contient les mêmes messages qu'auparavant, avec l'ajout des informations et des résultats de la requête.

## Nettoyer les ressources

Lorsque vous travaillez dans votre propre abonnement, il est recommandé, à la fin de chaque projet, de déterminer si vous avez toujours besoin des ressources que vous avez créées. Les ressources laissées en cours d'exécution peuvent vous coûter de l'argent. Vous pouvez supprimer les ressources une par une, ou choisir de supprimer le groupe de ressources afin de supprimer l'ensemble des ressources.

Vous pouvez rechercher et gérer les ressources dans le portail à l'aide des liens **Toutes les ressources** ou **Groupes de ressources** situés dans le volet de navigation de gauche.

Si vous utilisez un service gratuit, n'oubliez pas que vous êtes limité à trois index, indexeurs et sources de données. Vous pouvez supprimer des éléments un par un dans le portail pour ne pas dépasser la limite.

## Étapes suivantes

Dans ce guide de démarrage rapide C#, vous avez effectué une série de tâches pour créer un index, le charger avec des documents et exécuter des requêtes. À différents stades, nous avons pris des raccourcis afin de simplifier le code pour une meilleure lisibilité et compréhension. Si vous êtes familiarisé avec les concepts de base, nous vous recommandons l'article suivant pour explorer d'autres approches et concepts afin d'approfondir vos connaissances.

L'exemple de code et l'index sont des versions développées de celui-ci. L'exemple suivant ajoute une collection Rooms, utilise différentes classes et actions, et examine de plus près le fonctionnement du traitement.

### [Guide pratique pour développer dans .NET](#)

Vous souhaitez optimiser et réduire vos coûts de cloud ?

### [Démarrer l'analyse des coûts avec Cost Management](#)

# Mettre à niveau vers la version 11 du Kit de développement logiciel (SDK) .NET Recherche cognitive Azure

04/10/2020 • 13 minutes to read • [Edit Online](#)

Si vous utilisez la version 10.0 ou une version antérieure du [Kit de développement logiciel \(SDK\) .NET](#), cet article vous aidera à le mettre à niveau vers la version 11.

La version 11 est une bibliothèque de client entièrement repensée, publiée par l'équipe de développement de Kits de développement logiciel (SDK) d'Azure (les versions précédentes ont été produites par l'équipe de développement de Recherche cognitive Azure). La bibliothèque a été repensée pour une plus grande cohérence avec les autres bibliothèques de client Azure, en dépendant d'[Azure.Core](#) et de [System.Text.Json](#) et en implémentant des approches familières pour les tâches courantes.

Voici quelques-unes des principales différences que vous noterez dans la nouvelle version :

- Un package et une bibliothèque au lieu de plusieurs
- Un nouveau nom du package : `Azure.Search.Documents` au lieu de `Microsoft.Azure.Search`
- Trois clients au lieu de deux : `SearchClient`, `SearchIndexClient`, `SearchIndexerClient`
- Des différences d'affectation de noms dans une plage d'API et de petites différences structurelles qui simplifient certaines tâches

## NOTE

Passez en revue le [journal des modifications](#) pour obtenir la liste des modifications du kit de développement logiciel (SDK) .NET version 11.

## Consolidation des packages et des bibliothèques

La version 11 regroupe plusieurs packages et bibliothèques en un seul package et une seule bibliothèque. Après la migration, vous aurez moins de bibliothèques à gérer.

- [Package Azure.Search.Documents](#)
- [Informations de référence sur l'API pour la bibliothèque de client](#)

## Différences de clients

Le cas échéant, le tableau suivant mappe les bibliothèques de client entre les deux versions.

ÉTENDUE DES OPÉRATIONS	MICROSOFT.AZURE.SEARCH (V10)	AZURE.SEARCH.DOCUMENTS (V11)
Client utilisé pour les requêtes et pour remplir un index.	<code>SearchIndexClient</code>	<code>SearchClient</code>
Client utilisé pour les index, les analyseurs et les mappages de synonymes	<code>SearchServiceClient</code>	<code>SearchIndexClient</code>

ÉTENDUE DES OPÉRATIONS	MICROSOFT.AZURE.SEARCH (V10)	AZURE.SEARCH.DOCUMENTS (V11)
Client utilisé pour les indexeurs, les sources de données et les ensembles de compétences	SearchServiceClient	SearchIndexerClient (nouveau)

### IMPORTANT

`SearchIndexClient` existe dans les deux versions, mais prend en charge des éléments différents. Dans la version 10, `SearchIndexClient` crée des index et d'autres objets. Dans la version 11, `SearchIndexClient` fonctionne avec les index existants. Pour éviter toute confusion lors de la mise à jour du code, gardez à l'esprit l'ordre dans lequel les références de client sont mises à jour. Le fait de suivre l'ordre des étapes de mise à niveau devrait aider à atténuer les éventuels problèmes de remplacement de chaînes.

## Affectation de noms et autres différences d'API

Outre les différences de clients (notées précédemment et donc omises ici), plusieurs autres API ont été renommées et, dans certains cas, redéfinies. Les différences entre les noms de classe sont résumées ci-dessous. Cette liste n'est pas exhaustive, mais elle regroupe les modifications d'API par tâche, ce qui peut être utile pour les révisions de blocs de code spécifiques. Pour obtenir la liste détaillée des mises à jour des API, consultez le [journal des modifications](#) pour `Azure.Search.Documents` sur GitHub.

### Authentification et chiffrement

VERSION 10	ÉQUIVALENT DANS LA VERSION 11
<code>SearchCredentials</code>	<code>AzureKeyCredential</code>
<code>EncryptionKey</code> (existait dans le <a href="#">Kit de développement logiciel (SDK) de préversion</a> en tant que fonctionnalité généralement disponible)	<code>SearchResourceEncryptionKey</code>

### Index, analyseurs et mappages de synonymes

VERSION 10	ÉQUIVALENT DANS LA VERSION 11
<code>Index</code>	<code>SearchIndex</code>
<code>Champ</code>	<code>SearchField</code>
<code>DataType</code>	<code>SearchFieldDataType</code>
<code>ItemError</code>	<code>SearchIndexerError</code>
<code>Analyseur</code>	<code>LexicalAnalyzer</code> (ainsi que <code>AnalyzerName</code> vers <code>LexicalAnalyzerName</code> )
<code>AnalyzeRequest</code>	<code>AnalyzeTextOptions</code>
<code>StandardAnalyzer</code>	<code>LuceneStandardAnalyzer</code>
<code>StandardTokenizer</code>	<code>LuceneStandardTokenizer</code> (ainsi que <code>StandardTokenizerV2</code> vers <code>LuceneStandardTokenizerV2</code> )

VERSION 10	ÉQUIVALENT DANS LA VERSION 11
TokenInfo	AnalyzedTokenInfo
Générateur de jetons	LexicalTokenizer (ainsi que <code>TokenizerName</code> vers <code>LexicalTokenizerName</code> )
SynonymMap.Format	Aucun. Supprimez les références à <code>Format</code> .

Les définitions de champ sont rationalisées : [SearchableField](#), [SimpleField](#) et [ComplexField](#) sont de nouvelles API permettant de créer des définitions de champ.

## Indexeurs, sources de données et ensembles de compétences

VERSION 10	ÉQUIVALENT DANS LA VERSION 11
Indexeur	SearchIndexer
DataSource	SearchIndexerDataSourceConnection
Compétence	SearchIndexerSkill
Ensemble de compétences	SearchIndexerSkillset
DataSourceType	SearchIndexerDataSourceType

## Importation des données

VERSION 10	ÉQUIVALENT DANS LA VERSION 11
IndexAction	IndexDocumentsAction
IndexBatch	IndexDocumentsBatch

## Définitions et résultats des requêtes

VERSION 10	ÉQUIVALENT DANS LA VERSION 11
DocumentSearchResult	SearchResult ou SearchResults, selon que le résultat est un document unique ou plusieurs documents.
DocumentSuggestResult	SuggestResults
SearchParameters	SearchOptions

## Contenu de la version 11

Chaque version d'une bibliothèque de client Recherche cognitive Azure cible une version correspondante de l'API REST. L'API REST est considérée comme fondamentale pour le service, avec des Kits de développement logiciel (SDK) individuels encapsulant une version de l'API REST. En tant que développeur .NET, il peut être utile de passer en revue la [documentation sur l'API REST](#) si vous souhaitez plus d'informations sur des objets ou des opérations spécifiques.

La version 11 cible le [service de recherche 2020-06-30](#). Étant donné que la version 11 est également une nouvelle

bibliothèque de client entièrement créée, l'essentiel de l'effort de développement s'est concentré sur l'équivalence avec la version 10, certaines fonctionnalités de l'API REST étant encore en attente de prise en charge.

La version 11.0 prend entièrement en charge les opérations et objets suivants :

- Création et gestion d'index
- Création et gestion de mappage de synonymes
- Tous les types de requêtes et la syntaxe (à l'exception des filtres géospatiaux)
- Objets et opérations de l'indexeur pour l'indexation des sources de données Azure, y compris les sources de données et ensembles de compétences

La version 11.1 ajoute ce qui suit :

- [FieldBuilder](#) (ajouté à la version 11.1)
- [Propriété du sérialiseur](#) (ajoutée à la version 11.1) pour prendre en charge la sérialisation personnalisée

### Fonctionnalités en attente

Les fonctionnalités suivantes de la version 10 ne sont pas encore disponibles dans la version 11. Si vous avez besoin de ces fonctionnalités, reportez la migration jusqu'à ce qu'elles soient prises en charge.

- Types géospatiaux
- [Base de connaissances](#)

## Procédure de mise à niveau

Les étapes suivantes vous permettent de commencer une migration de code en parcourant la première série de tâches requises, notamment en ce qui concerne les références de client.

1. Installez le [package Azure.Search.Documents](#) en cliquant avec le bouton droit sur les références de votre projet et en sélectionnant « Gérer les packages NuGet... » dans Visual Studio.
2. Remplacez les directives d'utilisation pour Microsoft.Azure.Search par ce qui suit :

```
using Azure;
using Azure.Search.Documents;
using Azure.Search.Documents.Indexes;
using Azure.Search.Documents.Indexes.Models;
using Azure.Search.Documents.Models;
```

3. Pour les classes qui requièrent la sérialisation JSON, remplacez `using Newtonsoft.Json` par `using System.Text.Json.Serialization`.
4. Révisez le code d'authentification du client. Dans les versions précédentes, vous deviez utiliser les propriétés sur l'objet client pour définir la clé API (par exemple, la propriété [SearchServiceClient.Credentials](#)). Dans la version actuelle, utilisez la classe [AzureKeyCredential](#) pour transmettre la clé en tant qu'informations d'identification. Ainsi, si nécessaire, vous pouvez mettre à jour la clé API, sans créer de nouveaux objets clients.

Les propriétés du client ont été rationalisées pour `Endpoint`, `ServiceName` et `IndexName` (le cas échéant). L'exemple suivant utilise la classe [URI](#) du système pour fournir le point de terminaison et la classe [Environnement](#) pour lire la valeur de clé :

```
Uri endpoint = new Uri(Environment.GetEnvironmentVariable("SEARCH_ENDPOINT"));
AzureKeyCredential credential = new AzureKeyCredential(
    Environment.GetEnvironmentVariable("SEARCH_API_KEY"));
SearchIndexClient indexClient = new SearchIndexClient(endpoint, credential);
```

5. Ajoutez de nouvelles références de client pour les objets liés à l'indexeur. Si vous utilisez des indexeurs, des sources de source ou des ensembles de compétences, modifiez les références de client en [SearchIndexerClient](#). Ce client est nouveau dans la version 11 et n'a pas d'antécédent.
6. Mettez à jour les références de client pour les requêtes et l'importation de données. Les instances de [SearchIndexClient](#) doivent être remplacées par [SearchClient](#). Pour éviter toute confusion dans les noms, veillez à intercepter toutes les instances avant de passer à l'étape suivante.
7. Mettez à jour les références de client pour les objets index, indexeur, mappage de synonymes et analyseur. Les instances de [SearchServiceClient](#) doivent être remplacées par [SearchIndexClient](#).
8. Autant que possible, mettez à jour les classes, les méthodes et les propriétés pour utiliser les API de la nouvelle bibliothèque. La section relative aux [différences d'affectation de noms](#) est un point de départ, mais vous pouvez également consulter le [journal des modifications](#).  
Si vous avez des difficultés à trouver des API équivalentes, nous vous suggérons de signaler le problème sur <https://github.com/MicrosoftDocs/azure-docs/issues> afin que nous puissions améliorer la documentation ou examiner le problème.
9. Régénérez la solution. Après avoir corrigé les erreurs de build ou les avertissements, vous pouvez apporter des modifications supplémentaires à votre application pour bénéficier des [nouvelles fonctionnalités](#).

## Changements cassants dans la version 11

Compte tenu des modifications radicales apportées aux bibliothèques et aux API, une mise à niveau vers la version 11 n'est pas anodine et constitue un changement cassant dans la mesure où votre code ne sera plus rétrocompatible avec la version 10 et les versions antérieures. Pour une analyse approfondie des différences, consultez le [journal des modifications](#) pour [Azure.Search.Documents](#).

En termes de mises à jour de version de service, lorsque les modifications de code de la version 11 sont liées à la fonctionnalité existante (et pas simplement à la refactorisation des API), vous noterez les modifications de comportement suivantes :

- L'[algorithme de classement BM25](#) remplace l'algorithme de classement précédent par une technologie plus récente. Les nouveaux services utiliseront cet algorithme automatiquement. Pour les services existants, vous devez définir des paramètres pour utiliser le nouvel algorithme.
- Les [résultats ordonnés](#) pour les valeurs Null ont été modifiés dans cette version ; les valeurs Null apparaissent en premier si le tri est `asc` et en dernier si le tri est `desc`. Si vous avez écrit du code pour gérer la manière dont les valeurs Null sont triées, vous devez revoir et éventuellement supprimer ce code s'il n'est plus nécessaire.

## Étapes suivantes

- [Package Azure.Search.Documents](#)
- [Exemples sur GitHub](#)
- [Informations de référence sur l'API Azure.Search.Document](#)

# Effectuer une mise à niveau vers la version 10 du SDK .NET Recherche cognitive Azure

04/10/2020 • 14 minutes to read • [Edit Online](#)

Si vous utilisez la version 9.0-ou une version antérieure du [SDK .NET](#), cet article vous aidera à mettre à niveau votre application pour utiliser la version 10.

La Recherche Azure est renommée Recherche cognitive Azure dans la version 10, mais les espaces de noms et les noms de package restent inchangés. Les versions antérieures du SDK (9.0 et précédentes) continuent d'utiliser l'ancien nom. Pour plus d'informations sur l'utilisation du SDK et pour découvrir des exemples, consultez [Guide pratique pour utiliser la Recherche cognitive Azure à partir d'une application .NET](#).

La version 10 ajoute plusieurs fonctionnalités et correctifs de bogues, ce qui la place au même niveau fonctionnel que la version de l'API REST [2019-05-06](#). Dans les cas où une modification rompt le code existant, nous vous guiderons à travers les [étapes requises pour résoudre le problème](#).

## NOTE

Si vous utilisez la version 8.0-preview ou une version antérieure, vous devez tout d'abord effectuer une mise à niveau vers la version 9, puis vers la version 10. Consultez [Mise à niveau vers la version 9 du SDK .NET Recherche Azure](#) pour obtenir des conseils sur la migration.

Votre instance de service de recherche prend en charge plusieurs versions de l'API REST, y compris la plus récente. Vous pouvez continuer à utiliser une version lorsqu'elle n'est pas la plus récente, mais nous vous recommandons de migrer votre code pour utiliser la dernière version. Lorsque vous utilisez l'API REST, vous devez spécifier la version de l'API dans chaque requête via le paramètre « `api-version` » (Version de l'API). Lorsque vous utilisez le Kit de développement logiciel (SDK) .NET, la version du Kit de développement logiciel (SDK) que vous utilisez détermine la version correspondante de l'API REST. Si vous utilisez un Kit de développement logiciel (SDK) plus ancien, vous pouvez continuer à exécuter ce code sans changement, même si le service est mis à niveau pour prendre en charge une version plus récente de l'API.

## Nouveautés de la version 10

Version 10 du SDK .NET Recherche cognitive cible l'API REST ([2019-05-06](#)) avec les mises à jour suivantes :

- Présentation de deux nouvelles compétences : [compétence Conditionnelle](#) et [compétence Traduction de texte](#).
- Les entrées de [compétence du modélisateur](#) ont été restructurées pour s'adapter à la consolidation des contextes imbriqués. Pour plus d'informations, consultez cet [exemple de définition JSON](#).
- Ajout de deux nouvelles [fonctions de mappage de champs](#) :
  - [urlEncode](#)
  - [urlDecode](#)
- À certaines occasions, les erreurs et avertissements qui s'affichent dans [l'état d'exécution de l'indexeur](#) peuvent contenir des détails supplémentaires qui facilitent le débogage. [IndexerExecutionResult](#) a été mis à jour pour refléter ce comportement.
- Les compétences individuelles définies dans un [ensemble de compétences](#) peuvent éventuellement être identifiées en spécifiant une propriété [name](#).
- [ServiceLimits](#) affiche les limites pour les [types complexes](#) et [IndexerExecutionInfo](#) affiche les limites/quotas pertinent(e)s de l'indexeur.

# Procédure de mise à niveau

1. Mettez à jour vos références NuGet `Microsoft.Azure.Search`, soit en utilisant la console NuGet Package, soit en effectuant un clic droit sur les références de votre projet et en sélectionnant « Gérer les packages NuGet » dans Visual Studio.
2. Une fois que NuGet a téléchargé les nouveaux packages et leurs dépendances, recompilez votre projet.
3. Si votre build échoue, vous devrez résoudre chaque erreur de build. Consultez [Changements cassants dans la version 10](#) pour en savoir plus sur la façon de résoudre chaque erreur de build potentielle.
4. Une fois que vous avez résolu les erreurs de build ou les avertissements sur la génération, vous pouvez apporter des changements à votre application pour exploiter les nouvelles fonctionnalités, si vous le souhaitez. Les nouvelles fonctionnalités du Kit SDK sont détaillées dans la section [Nouveautés de la version 10](#).

## Changements cassants dans la version 10

Il y a plusieurs changements cassants dans la version 10 qui sont susceptibles de nécessiter des modifications de code en plus de la régénération de votre application.

### NOTE

La liste des modifications ci-dessous n'est pas exhaustive. Certaines modifications ne généreront probablement pas d'erreur de build mais un arrêt technique, car elles rompent la compatibilité binaire avec des assemblies qui dépendent de versions antérieures d'assemblies du SDK .NET Azure Cognitive Search. Les changements significatifs qui appartiennent à cette catégorie sont également répertoriés avec des recommandations. Régénérez votre application lors de la mise à niveau vers la version 10 pour éviter tout problème de compatibilité binaire.

### Définition de compétence API web personnalisée

La définition de la [compétence API web personnalisée](#) n'a pas été correctement spécifiée dans la version 9 et les versions antérieures.

Le modèle de `WebApiSkill` spécifiait `HttpHeaders` en tant que propriété d'objet qui *contient* un dictionnaire. La création d'un ensemble de compétences avec un `WebApiSkill` construit de cette manière entraînait une exception, car l'API REST considérait que la demande était mal formée. Ce problème a été corrigé, en faisant de `HttpHeaders` une propriété de dictionnaire de niveau supérieur sur le modèle `WebApiSkill` proprement dit, ce qui est considéré comme une demande valide de l'API REST.

Par exemple, si vous avez précédemment tenté d'instancier un `WebApiSkill` comme suit :

```
var webApiSkill = new WebApiSkill(
    inputs,
    outputs,
    uri: "https://contoso.example.org")
{
    HttpHeaders = new WebApiHttpHeaders()
    {
        Headers = new Dictionary<string, string>()
        {
            ["header"] = "value"
        }
    }
};
```

Remplacez-le par ce qui suit, afin d'éviter l'erreur de validation de l'API REST :

```
var webApiSkill = new WebApiSkill(
    inputs,
    outputs,
    uri: "https://contoso.example.org")
{
    HttpHeaders = new Dictionary<string, string>()
    {
        ["header"] = "value"
    }
};
```

## La compétence du modélisateur permet la consolidation des contextes imbriqués

La compétence du modélisateur peut désormais autoriser la consolidation des entrées à partir de contextes imbriqués. Pour activer ce changement, nous avons modifié `InputFieldMappingEntry` de sorte qu'il puisse être instancié en spécifiant simplement une propriété `Source`, ou les propriétés `SourceContext` et `Inputs`.

Vous n'aurez probablement pas besoin d'apporter des modifications au code. Toutefois, notez qu'une seule de ces deux combinaisons est autorisée. La procédure est la suivante :

- La création d'un `InputFieldMappingEntry` où seul `Source` est initialisé est valide.
- La création d'un `InputFieldMappingEntry` où seuls `SourceContext` et `Inputs` sont initialisés est valide.
- Toutes les autres combinaisons qui impliquent ces trois propriétés ne sont pas valides.

Si vous décidez de commencer à utiliser cette nouvelle fonctionnalité, assurez-vous d'abord que tous vos clients sont mis à jour pour utiliser la version 10, avant de déployer ce changement. Dans le cas contraire, il est possible qu'une mise à jour par un client (à l'aide d'une version antérieure du SDK) vers la compétence du modélisateur puisse entraîner des erreurs de validation.

### NOTE

Même si le modèle `InputFieldMappingEntry` sous-jacent a été modifié pour permettre la consolidation à partir de contextes imbriqués, son utilisation n'est valide que dans la définition d'une compétence de modélisateur. L'utilisation de cette fonctionnalité dans d'autres compétences, bien que valide au moment de la compilation, génère une erreur de validation au moment de l'exécution.

## Les compétences peuvent être identifiées par un nom

Chaque compétence au sein d'un jeu de compétences a maintenant une nouvelle propriété `Name`, qui peut être initialisée dans votre code pour aider à identifier la compétence. Cette option est facultative. Quand elle n'est pas spécifiée (valeur par défaut, si aucun changement de code explicite n'a été apporté), un nom par défaut lui est attribué à l'aide de l'index de base 1 de la compétence dans le jeu de compétences, avec en préfixe le caractère « # ». Par exemple, dans la définition du jeu de compétences suivante (la plupart des initialisations ont été ignorées par souci de concision) :

```
var skillset = new Skillset()
{
    Skills = new List<Skill>()
    {
        new SentimentSkill(),
        new WebApiSkill(),
        new ShaperSkill(),
        ...
    }
}
```

`SentimentSkill` reçoit le nom #1, `WebApiSkill` reçoit le nom #2, `ShaperSkill` reçoit le nom #3, etc.

Si vous choisissez d'identifier les compétences avec un nom personnalisé, veillez d'abord à mettre à jour toutes les instances de vos clients vers la version 10 du SDK. Dans le cas contraire, il est possible qu'un client qui utilise une version antérieure du SDK puisse `null` la propriété `Name` d'une compétence, ce qui amène le client à revenir au schéma de nommage par défaut.

## Détails sur les erreurs et les avertissements

Les modèles `ModelError` et `ItemWarning` qui encapsulent les détails des erreurs et des avertissements (respectivement) qui se produisent pendant l'exécution d'un indexeur ont été modifiés de façon à inclure trois nouvelles propriétés avec l'objectif de faciliter le débogage de l'indexeur. Ces propriétés sont :

- `Name` : Nom de la source à l'origine de l'erreur. Par exemple, cela peut faire référence à une compétence particulière dans le jeu de compétences joint.
- `Details` : Détails supplémentaires sur l'erreur ou l'avertissement.
- `DocumentationLink` : Lien vers un guide de dépannage pour l'erreur ou l'avertissement spécifique.

### NOTE

Nous avons commencé à structurer nos erreurs et avertissements pour inclure ces détails utiles chaque fois que cela est possible. Nous nous efforçons de faire en sorte que, pour l'ensemble des erreurs et des avertissements, ces détails soient présents, mais il s'agit d'un travail en cours et ces détails supplémentaires ne sont pas toujours remplis.

## Étapes suivantes

- Les modifications apportées à la compétence du modélisateur ont l'impact potentiel le plus important sur le code nouveau ou existant. Pour continuer, veillez à consulter à nouveau cet exemple illustrant la structure d'entrée : [Exemple de définition JSON de compétence du modélisateur](#)
- Consultez la [vue d'ensemble de l'enrichissement par IA](#).
- N'hésitez pas à nous faire part de vos commentaires sur le kit de développement logiciel. Si vous rencontrez des problèmes, n'hésitez pas à nous demander de l'aide sur [Stack Overflow](#). Si vous trouvez un bogue, vous pouvez signaler un problème dans le [Référentiel GitHub du Kit de développement logiciel \(SDK\) Azure .NET](#). Veillez à faire précéder le titre de votre problème du préfixe « [Recherche cognitive Azure] ».

# Effectuer une mise à niveau vers la version 9 du SDK .NET Recherche Azure

04/10/2020 • 16 minutes to read • [Edit Online](#)

Si vous utilisez la version 7.0-preview ou une version antérieure du [SDK .NET Recherche Azure](#), cet article vous aidera à mettre à niveau votre application pour utiliser la version 9.

## NOTE

Si vous souhaitez utiliser la version 8.0-preview pour tester les fonctionnalités qui ne sont pas encore à la disposition générale, vous pouvez également suivre les instructions de cet article pour passer à la version 8.0-preview à partir de versions antérieures.

Pour avoir un aperçu général du kit de développement logiciel et avoir des exemples, voir [Comment utiliser Azure Search à partir d'une application .NET](#).

La version 9 du SDK .NET Recherche Azure contient de nombreuses modifications par rapport aux versions antérieures. Certaines de ces modifications sont des changements cassants, mais elles ne devraient que des changement relativement mineurs sur votre code. Consultez la page [Procédure de mise à niveau](#) pour obtenir des instructions sur la façon de modifier le code à utiliser avec la nouvelle version du kit de développement logiciel.

## NOTE

Si vous utilisez la version 4.0-preview ou une version antérieure, vous devez tout d'abord effectuer une mise à niveau vers la version 5, puis vers la version 9. Consultez [Mise à niveau vers la version 5 du SDK .NET Recherche Azure](#) pour obtenir des conseils sur la migration.

Votre instance de service Recherche Azure prend en charge plusieurs versions de l'API REST, y compris la plus récente. Vous pouvez continuer à utiliser une version lorsqu'elle n'est pas la plus récente, mais nous vous recommandons de migrer votre code pour utiliser la dernière version. Lorsque vous utilisez l'API REST, vous devez spécifier la version de l'API dans chaque requête via le paramètre « api-version » (Version de l'API). Lorsque vous utilisez le Kit de développement logiciel (SDK) .NET, la version du Kit de développement logiciel (SDK) que vous utilisez détermine la version correspondante de l'API REST. Si vous utilisez un Kit de développement logiciel (SDK) plus ancien, vous pouvez continuer à exécuter ce code sans changement, même si le service est mis à niveau pour prendre en charge une version plus récente de l'API.

## Nouveautés de la version 9

La version 9 d'Azure Search .NET SDK est basée sur la version du 06/05/2019 de l'API REST Azure Search, avec les fonctions suivantes :

- L'[enrichissement par IA](#) est la capacité à extraire du texte à partir d'images, d'objets blob et d'autres sources de données non structurées pour enrichir le contenu et ainsi faciliter les recherches dans un index de Recherche Azure.
- La prise en charge de [types complexes](#) vous permet de modéliser presque n'importe quelle structure JSON imbriquée dans un index Recherche Azure.
- La fonctionnalité [Autocomplétion](#) fournit une alternative à l'API [Suggestion](#) pour implémenter le comportement de recherche au cours de la frappe. L'autocomplétion « termine » le mot ou l'expression que l'utilisateur est en train de taper.

- Le mode d'analyse [JsonLines](#), inclus dans l'indexation des objets Blob Azure, crée un document de recherche par entité JSON, séparé par un saut de ligne.

## Nouvelles fonctionnalités d'évaluation dans la version 8.0-preview

La version 8.0-preview du SDK .NET Recherche Azure cible l'API version 2017-11-11-Preview. Cette version inclut toutes les fonctionnalités de la version 9, plus :

- [Clés de chiffrement gérées par le client](#) pour le chiffrement au repos côté service est une nouvelle fonctionnalité d'évaluation. Outre le chiffrement au repos intégré géré par Microsoft, vous pouvez appliquer une couche supplémentaire de chiffrement lorsque vous êtes l'unique propriétaire des clés.

## Procédure de mise à niveau

Tout d'abord, mettez à jour vos références NuGet [Microsoft.Azure.Search](#), soit en utilisant la console NuGet Package, soit en effectuant un clic droit sur les références de votre projet et en sélectionnant « Gérer les packages NuGet » dans Visual Studio.

Une fois que NuGet a téléchargé les nouveaux packages et leurs dépendances, recompilez votre projet. En fonction de la structure de votre code, la recompilation peut réussir. Si c'est le cas, vous êtes prêt !

Si votre build échoue, vous devrez résoudre chaque erreur de build. Voir [Changements cassants dans la version 9](#) pour en savoir plus sur la façon de résoudre chaque erreur de build potentielle.

D'autres avertissements de compilation sont susceptibles de s'afficher ; ils sont liés à des propriétés ou des méthodes obsolètes. Ces avertissements incluent des instructions sur les fonctionnalités à utiliser à la place de la fonctionnalité déconseillée. Par exemple, si votre application utilise la propriété [DataSourceType.DocumentDb](#), vous devriez obtenir un avertissement indiquant « Ce membre est déconseillé. Utilisez CosmosDb à la place ».

Une fois que vous avez résolu les erreurs de build ou les avertissements sur la génération, vous pouvez apporter des changements à votre application pour exploiter les nouvelles fonctionnalités, si vous le souhaitez. Les nouvelles fonctionnalités du Kit SDK sont détaillées dans la section [Nouveautés de la version 9](#).

## Changements cassants dans la version 9

Il y a plusieurs changements cassants dans la version 9 qui sont susceptibles de nécessiter des modifications de code en plus de la régénération de votre application.

### NOTE

La liste des modifications ci-dessous n'est pas exhaustive. Certains changements ne provoqueront probablement pas d'erreurs de build, mais sont techniquement cassants dans la mesure où ils rompent la compatibilité binaire avec des assemblies qui dépendent de versions antérieures d'assemblies SDK .NET Recherche Azure. Ces modifications ne sont pas répertoriées ci-dessous. Régénérez votre application lors de la mise à niveau vers la version 9 pour éviter tout problème de compatibilité binaire.

### Propriétés immuables

Les propriétés publiques de plusieurs classes de modèle sont désormais immuables. Si vous avez besoin créer des instances personnalisées de ces classes à des fins de test, vous pouvez utiliser les nouveaux constructeurs paramétrés :

- [AutocompleteItem](#)
- [DocumentSearchResult](#)
- [DocumentSuggestResult](#)
- [FacetResult](#)

- `SearchResult`
- `SuggestResult`

## Changements dans les champs

La classe `Field` a été modifiée maintenant qu'elle peut également représenter des champs complexes.

Les propriétés `bool` suivantes sont désormais Nullable :

- `IsFilterable`
- `IsFacetable`
- `IsSearchable`
- `IsSortable`
- `IsRetrievable`
- `IsKey`

Cela vient du fait que ces propriétés doivent maintenant être `null` dans le cas des champs complexes. Si vous avez un code qui lit ces propriétés, il doit être prêt à gérer `null`. Notez que toutes les autres propriétés de `Field` ont toujours été et continuent à être Nullable, et certaines d'entre elles seront également `null` dans le cas de champs complexes, notamment dans les cas suivants :

- `Analyzer`
- `SearchAnalyzer`
- `IndexAnalyzer`
- `SynonymMaps`

Le constructeur sans paramètre de `Field` est devenu `internal`. Dès lors, chaque `Field` requiert un nom explicite et un type de données au moment de la construction.

## Types de lot et résultats simplifiés

Dans la version 7.0-preview et versions antérieures, les différentes classes qui encapsulent des groupes de documents ont été structurées dans des hiérarchies de classes parallèles :

- `DocumentSearchResult` et `DocumentSearchResult<T>` héritées de `DocumentSearchResultBase`
- `DocumentSuggestResult` et `DocumentSuggestResult<T>` héritées de `DocumentSuggestResultBase`
- `IndexAction` et `IndexAction<T>` héritées de `IndexActionBase`
- `IndexBatch` et `IndexBatch<T>` héritées de `IndexBatchBase`
- `SearchResult` et `SearchResult<T>` héritées de `SearchResultBase`
- `SuggestResult` et `SuggestResult<T>` héritées de `SuggestResultBase`

Les types dérivés sans paramètre de type générique étaient destinés à être utilisés dans des scénarios de « type dynamique » et supposaient l'utilisation du type `Document`.

À compter de la version 8.0-preview, les classes de base et les classes dérivées non génériques ont toutes été supprimées. Pour les scénarios de type dynamique, vous pouvez utiliser `IndexBatch<Document>`, `DocumentSearchResult<Document>`, et ainsi de suite.

## ExtensibleEnum supprimé

La classe de base `ExtensibleEnum` a été supprimée. Toutes les classes qui en sont dérivées sont désormais de type struct, comme `AnalyzerName`, `DataType` et `DataSourceType`, par exemple. Leurs méthodes `Create` ont également été supprimées. Vous pouvez simplement supprimer les appels à `Create` étant donné que ces types sont convertibles implicitement à partir de chaînes. Si cela entraîne des erreurs du compilateur, vous pouvez appeler explicitement l'opérateur de conversion par le biais de cast pour lever l'ambiguité des types. Par exemple, vous pouvez modifier le code comme suit :

```

var index = new Index()
{
    Fields = new[]
    {
        new Field("id", DataType.String) { IsKey = true },
        new Field("message", AnalyzerName.Create("my_email_analyzer")) { IsSearchable = true }
    },
    ...
}

```

par ceci :

```

var index = new Index()
{
    Fields = new[]
    {
        new Field("id", DataType.String) { IsKey = true },
        new Field("message", (AnalyzerName)"my_email_analyzer") { IsSearchable = true }
    },
    ...
}

```

Les propriétés qui détenaient des valeurs facultatives de ces types sont maintenant explicitement de type Nullable afin qu'elles continuent à être facultatives.

### FacetResults et HitHighlights supprimés

Les classes `FacetResults` et `HitHighlights` ont été supprimées. Les résultats de facette ont désormais le type `IDictionary<string, IList<FacetResult>>` et les mises en surbrillance des correspondances le type `IDictionary<string, IList<string>>`. Un moyen rapide pour résoudre les erreurs de build introduites par cette modification consiste à ajouter les alias `using` en haut de chaque fichier qui utilise les types supprimés. Par exemple :

```

using FacetResults = System.Collections.Generic.IDictionary<string,
System.Collections.Generic.IList<Models.FacetResult>>;
using HitHighlights = System.Collections.Generic.IDictionary<string,
System.Collections.Generic.IList<string>>;

```

### Changer pour SynonymMap

Le constructeur `SynonymMap` n'a plus de paramètre `enum` pour `SynonymMapFormat`. Cette énumération avait une seule valeur et était donc redondante. Si vous voyez des erreurs de build suite à ce changement, supprimez simplement les références au paramètre `SynonymMapFormat`.

### Diverses modifications de modèles de classe

La propriété `AutocompleteMode` de `AutocompleteParameters` n'est plus Nullable. Si vous avez du code qui attribue cette propriété à `null`, vous pouvez simplement la supprimer et la propriété sera automatiquement initialisée à la valeur par défaut.

L'ordre des paramètres pour le constructeur `IndexAction`, maintenant que ce constructeur est généré automatiquement. Au lieu d'utiliser le constructeur, nous vous recommandons d'utiliser les méthodes de fabrique `IndexAction.Upload`, `IndexAction.Merge` et ainsi de suite.

### Fonctionnalités préliminaires supprimées

Si vous mettez à niveau à partir de la version 8.0-preview vers la version 9, n'oubliez pas que le chiffrement avec des clés gérées par le client a été supprimé dans la mesure où cette fonctionnalité est toujours en préversion.

Plus précisément, les propriétés `EncryptionKey` de `Index` et `SynonymMap` ont été supprimées.

Si votre application dépend fortement de cette fonctionnalité, vous ne pourrez pas effectuer une mise à niveau vers la version 9 du SDK .NET Recherche Azure. Vous pouvez continuer à utiliser la version 8.0-preview. Mais n'oubliez pas que nous déconseillons l'utilisation de Kits de développement logiciel en version préliminaire dans les applications de production. Les fonctionnalités préliminaires servent uniquement à des fins d'évaluation et peuvent changer.

#### NOTE

Si vous avez créé des index chiffrés ou des cartes de synonymes à l'aide de la version 8.0-preview du SDK, vous pourrez toujours les utiliser et modifier leurs définitions à l'aide de la version 9 du SDK sans nuire à leur état de chiffrement. La version 9 du kit SDK n'envoie pas la propriété `encryptionKey` de l'API REST et par conséquent l'API REST ne changera pas l'état de chiffrement de la ressource.

### Changement de comportement dans l'extraction de données

Si vous utilisez les API de « type dynamique » `Search`, `Suggest` ou `Get` qui retournent des instances de type `Document`, n'oubliez pas qu'elles désérialisent maintenant les tableaux JSON vides sur `object[]` au lieu de `string[]`.

## Conclusion

Si vous souhaitez plus d'informations sur l'utilisation du Kit de développement logiciel (SDK) Azure Search .NET, consultez les articles [Procédures .NET](#).

N'hésitez pas à nous faire part de vos commentaires sur le kit de développement logiciel. Si vous rencontrez des problèmes, n'hésitez pas à nous demander de l'aide sur [Stack Overflow](#). Si vous trouvez un bogue, vous pouvez signaler un problème dans le [Référentiel GitHub du Kit de développement logiciel \(SDK\) Azure .NET](#). N'oubliez pas de faire précéder le titre de votre problème du préfixe « [Recherche Azure] ».

Merci d'utiliser Azure Search !

# Effectuer une mise à niveau vers la version 5 du SDK .NET Recherche Azure

04/10/2020 • 12 minutes to read • [Edit Online](#)

Si vous utilisez la version 4.0-preview ou une version antérieure du [SDK .NET](#), cet article vous aidera à mettre à niveau votre application pour utiliser la version 5.

Pour avoir un aperçu général du kit de développement logiciel et avoir des exemples, voir [Comment utiliser Azure Search à partir d'une application .NET](#).

La version 5 du SDK .NET Recherche Azure contient des modifications par rapport aux versions antérieures. Elles sont pour la plupart mineures, et donc, la modification de votre code devrait ne nécessiter que peu d'effort.

Consultez la page [Procédure de mise à niveau](#) pour obtenir des instructions sur la façon de modifier le code à utiliser avec la nouvelle version du kit de développement logiciel.

## NOTE

Si vous utilisez la version 2.0-preview ou une version antérieure, vous devez tout d'abord effectuer une mise à niveau vers la version 3, puis vers la version 5. Consultez [Mise à niveau vers la version du Kit de développement logiciel \(SDK\) .NET version 3 d'Azure Search](#) pour obtenir des conseils sur la migration.

Votre instance de service Recherche Azure prend en charge plusieurs versions de l'API REST, y compris la plus récente. Vous pouvez continuer à utiliser une version lorsqu'elle n'est pas la plus récente, mais nous vous recommandons de migrer votre code pour utiliser la dernière version. Lorsque vous utilisez l'API REST, vous devez spécifier la version de l'API dans chaque requête via le paramètre « api-version » (Version de l'API). Lorsque vous utilisez le Kit de développement logiciel (SDK) .NET, la version du Kit de développement logiciel (SDK) que vous utilisez détermine la version correspondante de l'API REST. Si vous utilisez un Kit de développement logiciel (SDK) plus ancien, vous pouvez continuer à exécuter ce code sans changement, même si le service est mis à niveau pour prendre en charge une version plus récente de l'API.

## Nouveautés de la version 5

La version 5 du SDK .NET Recherche Azure cible la dernière version en disponibilité générale de l'API REST Recherche Azure, spécifiquement 2017-11-11. Cela permet d'utiliser de nouvelles fonctionnalités de Recherche Azure à partir d'une application .NET, notamment les suivantes :

- [Synonymes](#).
- Vous pouvez à présent accéder par programmation aux avertissements dans l'historique de l'exécution d'un indexeur (voir la propriété `Warning` d'`IndexerExecutionResult` dans les [informations de référence sur .NET](#) pour plus d'informations).
- Prise en charge de .NET Core 2.
- La nouvelle structure des packages vous permet de n'utiliser que les parties du SDK dont vous avez besoin (voir [Modifications notables dans la version 5](#) pour plus d'informations).

## Procédure de mise à niveau

Tout d'abord, mettez à jour vos références NuGet `Microsoft.Azure.Search`, soit en utilisant la console NuGet Package, soit en effectuant un clic droit sur les références de votre projet et en sélectionnant « Gérer les packages NuGet » dans Visual Studio.

Une fois que NuGet a téléchargé les nouveaux packages et leurs dépendances, recompilez votre projet. En fonction

de la structure de votre code, la recompilation peut réussir. Si c'est le cas, vous êtes prêt !

Si la compilation échoue, vous devriez voir une erreur de ce type :

```
The name 'SuggesterSearchMode' does not exist in the current context
```

L'étape suivante consiste à corriger cette erreur de build. Pour plus d'informations sur la cause de l'erreur et les possibilités de correction, consultez [Modifications notables dans la version 5](#).

Notez qu'en raison des modifications dans l'empaquetage du SDK .NET Recherche Azure, vous devez regénérer votre application pour pouvoir utiliser la version 5. Ces modifications sont détaillées dans [Modifications notables dans la version 5](#).

D'autres avertissements de compilation sont susceptibles de s'afficher ; ils sont liés à des propriétés ou des méthodes obsolètes. Ces avertissements incluent des instructions sur les fonctionnalités à utiliser à la place de la fonctionnalité déconseillée. Par exemple, si votre application utilise la méthode

`IndexingParametersExtensions.DoNotFailOnUnsupportedContentType`, vous devez obtenir un avertissement indiquant que le comportement concerné est désormais activé par défaut et qu'il n'est plus nécessaire d'appeler cette méthode.

Une fois que vous avez résolu les erreurs de build ou les avertissements sur la génération, vous pouvez apporter des changements à votre application pour exploiter les nouvelles fonctionnalités, si vous le souhaitez. Les nouvelles fonctionnalités du Kit SDK sont détaillées dans la section [Nouveautés de la version 5](#).

## Modifications notables dans la version 5

### Nouvelle structure des packages

La modification notable la plus importante dans la version 5 est que l'assembly `Microsoft.Azure.Search` et son contenu ont été divisés en quatre assemblies distincts qui sont désormais distribués sous la forme de quatre packages NuGet distincts :

- `Microsoft.Azure.Search` : Il s'agit d'un méta-package qui inclut tous les autres packages Recherche Azure en tant que dépendances. Si vous effectuez une mise à niveau à partir d'une version antérieure du SDK, la mise à niveau de ce package et sa régénération devraient suffire pour commencer à utiliser la nouvelle version.
- `Microsoft.Azure.Search.Data` : Utilisez ce package si vous développez une application .NET à l'aide de Recherche Azure et que vous devez uniquement interroger ou mettre à jour des documents dans vos index. Si vous devez également créer ou mettre à jour des index, des cartes de synonymes ou d'autres ressources de niveau service, utilisez le package `Microsoft.Azure.Search` à la place.
- `Microsoft.Azure.Search.Service` : Utilisez ce package si vous développez un processus d'automatisation en .NET pour gérer les index Recherche Azure, cartes de synonymes, indexeurs, sources de données ou autres ressources de niveau service. Si vous devez uniquement interroger ou mettre à jour des documents dans vos index, utilisez le package `Microsoft.Azure.Search.Data` à la place. Si vous avez besoin de toutes les fonctionnalités de Recherche Azure, utilisez le package `Microsoft.Azure.Search` à la place.
- `Microsoft.Azure.Search.Common` : Types courants requis par les bibliothèques .NET de Recherche Azure. Vous ne devriez pas avoir besoin d'utiliser ce package directement dans votre application ; il est uniquement destiné à être utilisé en tant que dépendance.

Cette modification marque techniquement une rupture, car plusieurs types ont été déplacés entre les assemblies. C'est pourquoi il est indispensable de regénérer votre application pour effectuer une mise à niveau vers la version 5 du SDK.

Il n'y a qu'un petit nombre d'autres modifications dans la version 5 qui soient susceptibles de nécessiter des modifications de code en plus de la régénération de votre application.

## Changement apporté à Suggesters

Le constructeur `Suggerer` n'a plus de paramètre `enum` pour `SuggererSearchMode`. Cette énumération avait une seule valeur et était donc redondante. Si vous voyez des erreurs de build suite à ce changement, supprimez simplement les références au paramètre `SuggererSearchMode`.

## Membres obsolètes supprimés

Vous pouvez voir des erreurs de build liées à des méthodes ou propriétés qui ont été marquées comme obsolètes dans les versions antérieures, puis supprimées de la version 5. Si vous rencontrez ces erreurs, voici comment les résoudre :

- Si vous utilisez la méthode `IndexingParametersExtensions.IndexStorageMetadataOnly`, utilisez `SetBlobExtractionMode(BlobExtractionMode.StorageMetadata)` à la place.
- Si vous utilisez la méthode `IndexingParametersExtensions.SkipContent`, utilisez `SetBlobExtractionMode(BlobExtractionMode.AllMetadata)` à la place.

## Fonctionnalités préliminaires supprimées

Si vous effectuez une mise à niveau de la version 4.0-preview vers la version 5, n'oubliez pas que la prise en charge de l'analyse CSV et des tableaux JSON des indexeurs d'objets blob a été supprimée, car ces fonctionnalités sont toujours en préversion. En particulier, les méthodes suivantes de la classe `IndexingParametersExtensions` ont été supprimées :

- `ParseJsonArrays`
- `ParseDelimitedTextFiles`

Si votre application dépend fortement de ces fonctionnalités, vous ne pourrez pas effectuer une mise à niveau vers la version 5 du SDK .NET Recherche Azure. Vous pouvez continuer à utiliser la version 4.0-preview. Mais n'oubliez pas que **nous déconseillons l'utilisation de Kits de développement logiciel en version préliminaire dans les applications de production**. Les fonctionnalités préliminaires servent uniquement à des fins d'évaluation et peuvent changer.

## Conclusion

Si vous souhaitez plus d'informations sur l'utilisation du Kit de développement logiciel (SDK) Azure Search .NET, consultez les articles [Procédures .NET](#).

N'hésitez pas à nous faire part de vos commentaires sur le kit de développement logiciel. Si vous rencontrez des problèmes, n'hésitez pas à nous demander de l'aide sur [Stack Overflow](#). Si vous trouvez un bogue, vous pouvez signaler un problème dans le [Référentiel GitHub du Kit de développement logiciel \(SDK\) Azure .NET](#). N'oubliez pas de faire précéder le titre de votre problème du préfixe « [Recherche Azure] ».

Merci d'utiliser Azure Search !

# Effectuer une mise à niveau vers la version 3 du SDK .NET Recherche Azure

04/10/2020 • 11 minutes to read • [Edit Online](#)

Si vous utilisez la version préliminaire 2.0 ou une version antérieure du [Kit de développement logiciel \(SDK\) .NET Azure Search](#), cet article vous aidera à mettre à niveau votre application pour utiliser la version 3.

Pour avoir un aperçu général du kit de développement logiciel et avoir des exemples, voir [Comment utiliser Azure Search à partir d'une application .NET](#).

La version 3 du SDK .NET Azure Search contient des modifications par rapport aux versions antérieures. Elles sont pour la plupart mineures, et donc, la modification de votre code devrait ne nécessiter que peu d'effort. Consultez la page [Procédure de mise à niveau](#) pour obtenir des instructions sur la façon de modifier le code à utiliser avec la nouvelle version du kit de développement logiciel.

## NOTE

Si vous utilisez la version 1.0.2-preview ou une version antérieure, vous devez tout d'abord mettre à niveau vers la version 1.1, puis vers la version 3. Consultez [Mise à niveau vers la version du Kit de développement logiciel \(SDK\) .NET version 1.1 d'Azure Search](#) pour plus d'instructions.

Votre instance de service Recherche Azure prend en charge plusieurs versions de l'API REST, y compris la plus récente. Vous pouvez continuer à utiliser une version lorsqu'elle n'est pas la plus récente, mais nous vous recommandons de migrer votre code pour utiliser la dernière version. Lorsque vous utilisez l'API REST, vous devez spécifier la version de l'API dans chaque requête via le paramètre « api-version » (Version de l'API). Lorsque vous utilisez le Kit de développement logiciel (SDK) .NET, la version du Kit de développement logiciel (SDK) que vous utilisez détermine la version correspondante de l'API REST. Si vous utilisez un Kit de développement logiciel (SDK) plus ancien, vous pouvez continuer à exécuter ce code sans changement, même si le service est mis à niveau pour prendre en charge une version plus récente de l'API.

## Nouveautés de la version 3

La version 3 du SDK .NET Azure Search cible la dernière version disponible de l'API REST Azure Search, spécifiquement 2016-09-01. Cela permet d'utiliser de nombreuses nouvelles fonctionnalités d'Azure Search à partir d'une application .NET, notamment les suivantes :

- [Analyseurs personnalisés](#)
- Prise en charge des indexeurs [Stockage Blob Azure](#) et [Stockage Table Azure](#)
- Personnalisation des indexeurs avec les [mappages de champs](#)
- Prise en charge des ETag pour permettre la mise à jour simultanée sécurisée des définitions d'index, des indexeurs et des sources de données
- Prise en charge de la création déclarative des définitions de champ index en décorant votre classe de modèle et en utilisant la nouvelle classe `FieldBuilder`.
- Prise en charge de .NET Core et .NET Portable Profile 111

## Procédure de mise à niveau

Tout d'abord, mettez à jour vos références NuGet `Microsoft.Azure.Search`, soit en utilisant la console NuGet Package, soit en effectuant un clic droit sur les références de votre projet et en sélectionnant « Gérer les packages NuGet » dans Visual Studio.

Une fois que NuGet a téléchargé les nouveaux packages et leurs dépendances, recompilez votre projet. En fonction de la structure de votre code, la recompilation peut réussir. Si c'est le cas, vous êtes prêt !

Si la compilation échoue, vous devriez voir une erreur de ce type :

```
Program.cs(31,45,31,86): error CS0266: Cannot implicitly convert type  
'Microsoft.Azure.Search.ISearchIndexClient' to 'Microsoft.Azure.Search.SearchIndexClient'. An explicit  
conversion exists (are you missing a cast?)
```

L'étape suivante consiste à corriger cette erreur de build. Consultez [Dernières modifications dans la version 3](#) pour plus d'informations sur la cause de l'erreur et les possibilités de correction.

D'autres avertissements de compilation sont susceptibles de s'afficher ; ils sont liés à des propriétés ou des méthodes obsolètes. Ces avertissements incluent des instructions sur les fonctionnalités à utiliser à la place de la fonctionnalité déconseillée. Par exemple, si votre application utilise la propriété

`IndexingParameters.Base64EncodeKeys`, vous devriez obtenir un avertissement indiquant

```
"This property is obsolete. Please create a field mapping using 'FieldMapping.Base64Encode' instead."
```

Une fois que vous avez résolu les erreurs de build, vous pouvez apporter des modifications à votre application pour exploiter les nouvelles fonctionnalités si vous le souhaitez. Les nouvelles fonctionnalités du SDK sont détaillées dans [Nouveautés de la version 3](#).

## Dernières modifications dans la version 3

Il n'y a qu'un petit nombre de modifications dans la version 3 qui soient susceptibles de nécessiter des modifications de code en plus de la recompilation de votre application.

### Type de retour `Indexes.GetClient`

La méthode `Indexes.GetClient` a un nouveau type de retour. Auparavant, elle renvoyait `SearchIndexClient`, mais elle a été remplacée par `ISearchIndexClient` dans la version 2.0-preview et cette modification a été répercutée dans la version 3. L'objectif est de prendre en charge les clients qui souhaitent simuler la méthode `GetClient` pour les tests unitaires en retournant une implémentation fictive de `ISearchIndexClient`.

#### Exemple

Si votre code ressemble à ce qui suit :

```
SearchIndexClient indexClient = serviceClient.Indexes.GetClient("hotels");
```

vous pouvez le modifier pour résoudre les éventuelles erreurs de build :

```
ISearchIndexClient indexClient = serviceClient.Indexes.GetClient("hotels");
```

### Les éléments `AnalyzerName` et `DataType`, entre autres, ne sont plus implicitement convertibles en chaînes

Il existe de nombreux types dans le SDK .NET Azure Search qui dérivent de `ExtensibleEnum`. Précédemment, ces types étaient tous implicitement convertibles en type `string`. Toutefois, un bogue a été découvert dans l'implémentation `Object.Equals` de ces classes et la résolution de ce bogue a nécessité la désactivation de cette conversion implicite. Une conversion explicite vers `string` reste autorisée.

#### Exemple

Si votre code ressemble à ce qui suit :

```

var customTokenizerName = TokenizerName.Create("my_tokenizer");
var customTokenFilterName = TokenFilterName.Create("my_tokenfilter");
var customCharFilterName = CharFilterName.Create("my_charfilter");

var index = new Index();
index.Analyzers = new Analyzer[]
{
    new CustomAnalyzer(
        "my_analyzer",
        customTokenizerName,
        new[] { customTokenFilterName },
        new[] { customCharFilterName }),
};

```

vous pouvez le modifier pour résoudre les éventuelles erreurs de build :

```

const string CustomTokenizerName = "my_tokenizer";
const string CustomTokenFilterName = "my_tokenfilter";
const string CustomCharFilterName = "my_charfilter";

var index = new Index();
index.Analyzers = new Analyzer[]
{
    new CustomAnalyzer(
        "my_analyzer",
        CustomTokenizerName,
        new TokenFilterName[] { CustomTokenFilterName },
        new CharFilterName[] { CustomCharFilterName })
};

```

## Membres obsolètes supprimés

Vous pouvez voir des erreurs de build liées à des méthodes ou propriétés qui ont été marquées comme obsolètes dans la version 2.0-preview puis supprimées de la version 3. Si vous rencontrez ces erreurs, voici comment les résoudre :

- Si vous utilisez ce constructeur : `ScoringParameter(string name, string value)`, remplacez-le par `ScoringParameter(string name, IEnumerable<string> values)`
- Si vous utilisez la propriété `ScoringParameter.Value`, remplacez-la par la propriété `ScoringParameter.Values` ou la méthode `ToString`.
- Si vous utilisez la propriété `SearchRequestOptions.RequestId`, remplacez-la par la propriété `ClientRequestId`.

## Fonctionnalités préliminaires supprimées

Si vous effectuez une mise à niveau de la version 2.0-preview vers la version 3, n'oubliez pas que la prise en charge de l'analyse JSON et CSV des indexeurs d'objets blob a été supprimée car ces fonctionnalités sont toujours en version préliminaire. En particulier, les méthodes suivantes de la classe `IndexingParametersExtensions` ont été supprimées :

- `ParseJson`
- `ParseJsonArrays`
- `ParseDelimitedTextFiles`

Si votre application dépend fortement de ces fonctionnalités, vous ne pourrez pas mettre à niveau vers la version 3 du SDK .NET Azure Search. Vous pouvez continuer à utiliser la version 2.0-preview. Mais n'oubliez pas que **nous déconseillons l'utilisation de Kits de développement logiciel en version préliminaire dans les applications de production**. Les fonctionnalités préliminaires servent uniquement à des fins d'évaluation et peuvent changer.

## Conclusion

Si vous souhaitez plus d'informations sur l'utilisation du Kit de développement logiciel (SDK) Azure Search .NET, consultez les articles [Procédures .NET](#).

N'hésitez pas à nous faire part de vos commentaires sur le kit de développement logiciel. Si vous rencontrez des problèmes, n'hésitez pas à nous demander de l'aide sur [Stack Overflow](#). Si vous trouvez un bogue, vous pouvez signaler un problème dans le [Référentiel GitHub du Kit de développement logiciel \(SDK\) Azure .NET](#). N'oubliez pas de faire précéder le titre de votre problème du préfixe « [Recherche Azure] ».

Merci d'utiliser Azure Search !

# Effectuer une mise à niveau vers la version 1.1 du SDK .NET Recherche Azure

04/10/2020 • 21 minutes to read • [Edit Online](#)

Si vous utilisez la version préliminaire 1.0.2 ou une version antérieure du [Kit de développement logiciel \(SDK\) .NET Azure Search](#), cet article vous aidera à mettre à niveau votre application pour utiliser la version 1.1.

Pour avoir un aperçu général du kit de développement logiciel et avoir des exemples, voir [Comment utiliser Azure Search à partir d'une application .NET](#).

## NOTE

Une fois que vous mettez à niveau vers la version 1.1, ou si vous utilisez déjà une version entre 1.1 et 2.0 (préversion), vous devez mettre à niveau vers la version 3. Consultez [Mise à niveau vers la version du Kit de développement logiciel \(SDK\) .NET version 3 d'Azure Search](#) pour obtenir des conseils sur la migration.

Tout d'abord, mettez à jour vos références NuGet `Microsoft.Azure.Search`, soit en utilisant la console NuGet Package, soit en effectuant un clic droit sur les références de votre projet et en sélectionnant « Gérer les packages NuGet » dans Visual Studio.

Une fois que NuGet a téléchargé les nouveaux packages et leurs dépendances, recompilez votre projet.

Si vous utilisez précédemment la version préliminaire 1.0.0-preview, 1.0.1-preview ou 1.0.2-preview, la build doit réussir et vous pouvez donc vous lancer !

Si vous utilisez précédemment la version préliminaire 0.13.0-preview ou une version antérieure, vous obtiendrez sans doute les erreurs de build suivantes :

```
Program.cs(137,56,137,62): error CS0117: 'Microsoft.Azure.Search.Models.IndexBatch' does not contain a definition for 'Create'  
Program.cs(137,99,137,105): error CS0117: 'Microsoft.Azure.Search.Models.IndexAction' does not contain a definition for 'Create'  
Program.cs(146,41,146,54): error CS1061: 'Microsoft.Azure.Search.IndexBatchException' does not contain a definition for 'IndexResponse' and no extension method 'IndexResponse' accepting a first argument of type 'Microsoft.Azure.Search.IndexBatchException' could be found (are you missing a using directive or an assembly reference?)  
Program.cs(163,13,163,42): error CS0246: The type or namespace name 'DocumentSearchResponse' could not be found (are you missing a using directive or an assembly reference?)
```

L'étape suivante consiste à corriger les erreurs de build une par une. La plupart d'entre elles nécessitent la modification de certains noms de classe et de méthode qui ont été renommés dans le kit de développement logiciel. [Liste des dernières modifications dans la version 1.1](#) répertorie ces changements de nom.

Si vous utilisez des classes personnalisées pour modéliser vos documents et que ces classes ont des propriétés de types primitifs ne pouvant avoir pour valeur null (par exemple, `int` ou `bool` en C#), il existe un correctif de bogue que vous devez connaître dans la version 1.1 du Kit de développement logiciel (SDK). Consultez [Résolution des bogues dans la version 1.1](#) pour plus de détails.

Enfin, une fois que vous avez résolu les erreurs de build, vous pouvez apporter des modifications à votre application pour exploiter les nouvelles fonctionnalités si vous le souhaitez.

# Liste des dernières modifications dans la version 1.1

La liste qui suit est classée selon la probabilité que la modification affecte votre code d'application.

## Modifications IndexBatch et IndexAction

`IndexBatch.Create` a été renommé `IndexBatch.New` et ne comporte plus d'argument `params`. Vous pouvez utiliser `IndexBatch.New` pour les lots qui combinent différents types d'actions (fusions, suppressions, etc.). En outre, voici les nouvelles méthodes statiques pour la création de lots comportant tous les mêmes actions : `Delete`, `Merge`, `MergeOrUpload` et `Upload`.

`IndexAction` ne contient plus de constructeurs publics et ses propriétés sont immuables. Vous devez utiliser les nouvelles méthodes statiques pour la création d'actions à des fins différentes : `Delete`, `Merge`, `MergeOrUpload` et `Upload`. `IndexAction.Create` a été supprimé. Si vous avez utilisé la surcharge qui accepte uniquement un document, veillez à utiliser `Upload` à la place.

### Exemple

Si votre code ressemble à ce qui suit :

```
var batch = IndexBatch.Create(documents.Select(doc => IndexAction.Create(doc)));
indexClient.Documents.Index(batch);
```

vous pouvez le modifier pour résoudre les éventuelles erreurs de build :

```
var batch = IndexBatch.New(documents.Select(doc => IndexAction.Upload(doc)));
indexClient.Documents.Index(batch);
```

Si vous le souhaitez, vous pouvez encore le simplifier en le ramenant à ce qui suit :

```
var batch = IndexBatch.Upload(documents);
indexClient.Documents.Index(batch);
```

## Modifications IndexBatchException

La propriété `IndexBatchException.IndexResponse` a été renommée `IndexingResults`, et son type est désormais `IList<IndexingResult>`.

### Exemple

Si votre code ressemble à ce qui suit :

```
catch (IndexBatchException e)
{
    Console.WriteLine(
        "Failed to index some of the documents: {0}",
        String.Join(", ", e.IndexResponse.Results.Where(r => !r.Succeeded).Select(r => r.Key)));
}
```

vous pouvez le modifier pour résoudre les éventuelles erreurs de build :

```
catch (IndexBatchException e)
{
    Console.WriteLine(
        "Failed to index some of the documents: {0}",
        String.Join(", ", e.IndexingResults.Where(r => !r.Succeeded).Select(r => r.Key)));
}
```

## Modifications des méthodes d'opération

Chaque opération du kit de développement logiciel .NET Azure Search est exposée en tant qu'ensemble de chargements de méthode pour les appelants synchrones et asynchrones. Les signatures et la factorisation de ces surcharges de méthode ont changé dans la version 1.1.

Par exemple, l'opération « Obtenir des statistiques d'Index » dans les versions antérieures du kit de développement expose ces signatures :

Dans `IIndexOperations` :

```
// Asynchronous operation with all parameters
Task<IndexGetStatisticsResponse> GetStatisticsAsync(
    string indexName,
    CancellationToken cancellationToken);
```

Dans `IndexOperationsExtensions` :

```
// Asynchronous operation with only required parameters
public static Task<IndexGetStatisticsResponse> GetStatisticsAsync(
    this IIndexOperations operations,
    string indexName);

// Synchronous operation with only required parameters
public static IndexGetStatisticsResponse GetStatistics(
    this IIndexOperations operations,
    string indexName);
```

Les signatures de méthode pour la même opération en version 1.1 ressemblent à ce qui suit :

Dans `IIndexesOperations` :

```
// Asynchronous operation with lower-level HTTP features exposed
Task<AzureOperationResponse<IndexGetStatisticsResult>> GetStatisticsWithHttpMessagesAsync(
    string indexName,
    SearchRequestOptions searchRequestOptions = default(SearchRequestOptions),
    Dictionary<string, List<string>> customHeaders = null,
    CancellationToken cancellationToken = default(CancellationToken));
```

Dans `IndexesOperationsExtensions` :

```
// Simplified asynchronous operation
public static Task<IndexGetStatisticsResult> GetStatisticsAsync(
    this IIndexesOperations operations,
    string indexName,
    SearchRequestOptions searchRequestOptions = default(SearchRequestOptions),
    CancellationToken cancellationToken = default(CancellationToken));

// Simplified synchronous operation
public static IndexGetStatisticsResult GetStatistics(
    this IIndexesOperations operations,
    string indexName,
    SearchRequestOptions searchRequestOptions = default(SearchRequestOptions));
```

À partir de la version 1.1, le Kit de développement logiciel (SDK) .NET Azure Search organise les méthodes d'opération différemment :

- Les paramètres facultatifs sont désormais modélisés en tant que paramètres par défaut plutôt que les surcharges de méthode supplémentaires. Cela réduit le nombre de surcharges de méthode, parfois

considérablement.

- Les méthodes d'extension masquent maintenant un grand nombre des détails superflus de HTTP de la part de l'appelant. Par exemple, les versions antérieures du SDK ont renvoyé un objet de réponse avec un code d'état HTTP que vous n'avez pas besoin de contrôler, car les méthodes de fonctionnement lèvent `CloudException` pour un code d'état qui signale une erreur. Les nouvelles méthodes d'extension ne retournent que des objets de modèle, ce qui vous évite de les désencapsuler dans votre code.
- À l'inverse, les principales interfaces exposent maintenant les méthodes qui vous offrent davantage de contrôle au niveau HTTP si vous en avez besoin. Vous pouvez maintenant transférer des en-têtes HTTP personnalisés à inclure dans les demandes et le nouveau type de retour `AzureOperationResponse<T>` vous donne un accès direct à `HttpRequestMessage` et à `HttpResponseMessage` pour l'opération. `AzureOperationResponse` est défini dans l'espace de noms `Microsoft.Rest.Azure` et remplace `Hyak.Common.OperationResponse`.

## Modifications ScoringParameters

Une nouvelle classe nommée `ScoringParameter` a été ajoutée à la dernière version du Kit de développement logiciel (SDK) pour faciliter la fourniture de paramètres de profils de score dans une requête de recherche. Précédemment, la propriété `ScoringProfiles` de la classe `SearchParameters` était de type `IList<string>`. À présent, elle est de type `IList<ScoringParameter>`.

### Exemple

Si votre code ressemble à ce qui suit :

```
var sp = new SearchParameters();
sp.ScoringProfile = "jobsScoringFeatured";      // Use a scoring profile
sp.ScoringParameters = new[] { "featuredParam-featured", "mapCenterParam-" + lon + "," + lat };
```

vous pouvez le modifier pour résoudre les éventuelles erreurs de build :

```
var sp = new SearchParameters();
sp.ScoringProfile = "jobsScoringFeatured";      // Use a scoring profile
sp.ScoringParameters =
    new[]
    {
        new ScoringParameter("featuredParam", new[] { "featured" }),
        new ScoringParameter("mapCenterParam", GeographyPoint.Create(lat, lon))
    };
```

## Modifications de modèles de classe

En raison des modifications de signature décrites dans [Modifications des méthodes d'opération](#), de nombreuses classes de l'espace de noms `Microsoft.Azure.Search.Models` ont été renommées ou supprimées. Par exemple :

- `IndexDefinitionResponse` a été remplacé par `AzureOperationResponse<Index>`
- `DocumentSearchResponse` a été renommé en `DocumentSearchResult`
- `IndexResult` a été renommé en `IndexingResult`
- `Documents.Count()` renvoie désormais un élément `long` avec le nombre de documents et non un élément `DocumentCountResponse`
- `IndexGetStatisticsResponse` a été renommé en `IndexGetStatisticsResult`
- `IndexListResponse` a été renommé en `IndexListResult`

Pour résumer, les classes dérivées de `OperationResponse` qui servaient uniquement à encapsuler un objet de modèle ont été supprimées. Les classes restantes ont vu leur suffixe passer de `Response` à `Result`.

### Exemple

Si votre code ressemble à ce qui suit :

```

IndexerGetStatusResponse statusResponse = null;

try
{
    statusResponse = _searchClient.Indexers.GetStatus(indexer.Name);
}
catch (Exception ex)
{
    Console.WriteLine("Error polling for indexer status: {0}", ex.Message);
    return;
}

IndexerExecutionResult lastResult = statusResponse.ExecutionInfo.LastResult;

```

vous pouvez le modifier pour résoudre les éventuelles erreurs de build :

```

IndexerExecutionInfo status = null;

try
{
    status = _searchClient.Indexers.GetStatus(indexer.Name);
}
catch (Exception ex)
{
    Console.WriteLine("Error polling for indexer status: {0}", ex.Message);
    return;
}

IndexerExecutionResult lastResult = status.LastResult;

```

#### **Les classes de réponse et IEnumarable**

Une modification supplémentaire pouvant affecter votre code est que les classes de réponse contenant des collections ne sont plus mises en œuvre `IEnumerable<T>`. Au lieu de cela, vous pourrez directement accéder à la propriété de collection. Par exemple, si votre code ressemble à ceci :

```

DocumentSearchResponse<Hotel> response = indexClient.Documents.Search<Hotel>(searchText, sp);
foreach (SearchResult<Hotel> result in response)
{
    Console.WriteLine(result.Document);
}

```

vous pouvez le modifier pour résoudre les éventuelles erreurs de build :

```

DocumentSearchResult<Hotel> response = indexClient.Documents.Search<Hotel>(searchText, sp);
foreach (SearchResult<Hotel> result in response.Results)
{
    Console.WriteLine(result.Document);
}

```

#### **Cas particulier pour les applications web**

Si vous disposez d'une application web qui sérialise `DocumentSearchResponse` directement pour envoyer des résultats de recherche au navigateur, vous devez modifier votre code sinon les résultats ne seront pas sérialisés correctement. Par exemple, si votre code ressemble à ceci :

```

public ActionResult Search(string q = "")
{
    // If blank search, assume they want to search everything
    if (string.IsNullOrWhiteSpace(q))
        q = "*";

    return new JsonResult
    {
        JsonRequestBehavior = JsonRequestBehavior.AllowGet,
        Data = _featuresSearch.Search(q)
    };
}

```

Vous pouvez le modifier en faisant en sorte que la propriété `.Results` de la réponse de la recherche corrige le rendu des résultats de recherche :

```

public ActionResult Search(string q = "")
{
    // If blank search, assume they want to search everything
    if (string.IsNullOrWhiteSpace(q))
        q = "*";

    return new JsonResult
    {
        JsonRequestBehavior = JsonRequestBehavior.AllowGet,
        Data = _featuresSearch.Search(q).Results
    };
}

```

Vous devrez rechercher ces cas dans votre code vous-même. Le compilateur ne vous avertira pas parce que `JsonResult.Data` est de type `object`.

### Modifications CloudException

La classe `CloudException` a été déplacée de l'espace de noms `Hyak.Common` à l'espace de noms `Microsoft.Rest.Azure`. En outre, sa propriété `Error` a été renommée en `Body`.

### Modifications de SearchServiceClient et SearchIndexClient

Le type de la propriété `Credentials` est passé de `SearchCredentials` à sa classe de base, `ServiceClientCredentials`. Si vous avez besoin d'accéder à l'élément `SearchCredentials` d'un élément `SearchIndexClient` ou `SearchServiceClient`, veuillez utiliser la nouvelle propriété `SearchCredentials`.

Dans les versions antérieures du Kit de développement logiciel (SDK), `SearchServiceClient` et `SearchIndexClient` avaient des constructeurs incluant un paramètre `HttpClient`. Ils ont été remplacés par des constructeurs qui incluent un élément `HttpClientHandler` et un tableau d'objets `DelegatingHandler`. Cela facilite l'installation de gestionnaires personnalisés pour pré-traiter des demandes HTTP si nécessaire.

Enfin, les constructeurs incluant un élément `Uri` et `SearchCredentials` ont été modifiés. Par exemple, si vous avez un code qui ressemble à ceci qui suit :

```

var client =
    new SearchServiceClient(
        new SearchCredentials("abc123"),
        new Uri("http://myservice.search.windows.net"));

```

vous pouvez le modifier pour résoudre les éventuelles erreurs de build :

```
var client =
    new SearchServiceClient(
        new Uri("http://myservice.search.windows.net"),
        new SearchCredentials("abc123"));
```

Notez également que le type du paramètre d'informations d'identification a été modifié en `ServiceClientCredentials`. Il est peu probable que cela affecte votre code, car l'élément `SearchCredentials` est dérivé de `ServiceClientCredentials`.

## Transfert d'un ID de requête

Dans les versions antérieures du Kit de développement logiciel (SDK), vous pouviez définir un ID de demande sur `SearchServiceClient` ou `SearchIndexClient`, et il était inclus dans chaque demande adressée à l'API REST. Cela peut s'avérer utile pour résoudre les problèmes de votre service de recherche si vous devez contacter le support technique. Cependant, il peut être plus utile de définir un ID de requête unique pour chaque opération plutôt que d'utiliser le même ID pour toutes les informations. Pour cette raison, les méthodes `SetClientRequestId` de `SearchServiceClient` et `SearchIndexClient` ont été supprimées. Vous pouvez en revanche transférer un ID de demande à chaque méthode d'opération avec le paramètre facultatif `SearchRequestOptions`.

### NOTE

Dans une prochaine version du Kit de développement logiciel, nous allons ajouter un nouveau mécanisme permettant de définir globalement un ID de demande sur les objets clients compatibles avec l'approche utilisée par d'autres kits de développement logiciel Azure.

## Exemple

Si vous avez un code qui ressemble à ce qui suit :

```
client.SetClientRequestId(Guid.NewGuid());
...
long count = client.Documents.Count();
```

vous pouvez le modifier pour résoudre les éventuelles erreurs de build :

```
long count = client.Documents.Count(new SearchRequestOptions(requestId: Guid.NewGuid()));
```

## Modifications de nom d'interface

Les noms d'interface du groupe d'opération ont tous changé pour être cohérents avec les noms de propriété correspondants :

- Le type de `ISearchServiceClient.Indexes` a été renommé de `IIndexOperations` en `IIndexesOperations`.
- Le type de `ISearchServiceClient.Indexers` a été renommé de `IIndexerOperations` en `IIndexersOperations`.
- Le type de `ISearchServiceClient.DataSources` a été renommé de `IDataSourceOperations` en `IDataSourcesOperations`.
- Le type de `ISearchIndexClient.Documents` a été renommé de `IDocumentOperations` en `IDocumentsOperations`.

Cette modification n'affectera probablement pas votre code, à moins que vous créeiez des versions fictives de ces interfaces à des fins de test.

## Résolution des bogues dans la version 1.1

Les versions antérieures du kit de développement logiciel .NET Azure Search relatif à la sérialisation de classes de modèle personnalisé présentaient un bogue. Le bogue peut se produire si vous avez créé une classe de modèle

personnalisé avec une propriété de type de valeur ne pouvant être définie sur null.

## Opérations à reproduire

Créez une classe de modèle personnalisé avec une propriété de type avec valeur ne pouvant être définie sur null.

Par exemple, ajoutez une propriété `UnitCount` publique de type `int` au lieu de `int?`.

Si vous indexez un document avec une valeur par défaut de ce type (par exemple, 0 pour `int`), le champ sera null dans Azure Search. Si par la suite vous recherchez ce document, l'appel `Search` lancera une exception `JsonSerializationException` signalant qu'il est impossible de convertir `null` en `int`.

Les filtres peuvent également ne pas fonctionner comme prévu car c'est la valeur null qui est inscrite dans l'index, en non la valeur attendue.

## Corriger des détails

Nous avons résolu ce problème dans la version 1.1 du Kit de développement logiciel (SDK). Maintenant, si vous avez une classe de modèle comme suit :

```
public class Model
{
    public string Key { get; set; }

    public int IntValue { get; set; }
}
```

et si vous définissez `IntValue` sur 0, cette valeur est correctement serialisée en tant que 0 sur le câble et stockée en tant que 0 dans l'index. Le retour fonctionne également comme prévu.

Cette approche présente un problème à ne pas ignorer : si vous utilisez un type de modèle avec une propriété ne pouvant être définie sur null, vous devez garantir qu'aucun document dans votre index ne contient de valeur null pour le champ correspondant. Ni le kit de développement logiciel ni l'API REST Azure Search ne vous aideront à appliquer cette recommandation.

Il ne s'agit pas d'une préoccupation hypothétique : imaginez un scénario dans lequel vous ajoutez un nouveau champ à un index existant qui est de type `Edm.Int32`. Après la mise à jour de la définition d'index, ce nouveau champ prendra la valeur null pour tous les documents (car tous les types peuvent avoir la valeur null dans Azure Search). Si vous utilisez ensuite une classe de modèle avec une propriété `int` ne pouvant être définie sur null pour ce champ, vous obtiendrez l'exception `JsonSerializationException` ci-dessous lorsque vous tenterez de récupérer des documents :

```
Error converting value {null} to type 'System.Int32'. Path 'IntValue'.
```

Pour cette raison, nous vous recommandons d'utiliser des types pour lesquels la valeur null est autorisée en tant que meilleure pratique.

Pour plus d'informations sur ce bogue et le correctif, consultez [ce problème sur GitHub](#).

# Mise à niveau des versions du Kit de développement logiciel (SDK) .NET Management

04/10/2020 • 7 minutes to read • [Edit Online](#)

Cet article explique comment migrer vers des versions ultérieures du SDK .NET Management de Recherche Azure, qui sert à provisionner ou à déprovisionner les services de recherche, à ajuster la capacité et à gérer les clés API.

Les kits SDK de gestion ciblent une version spécifique de l'API REST de gestion. Pour plus d'informations sur les concepts et les opérations, consultez [Gestion des recherches \(REST\)](#).

## Versions

VERSION DU SDK	VERSION DE L'API REST CORRESPONDANTE	AJOUT DE FONCTIONNALITÉS OU CHANGEMENT DE COMPORTEMENT
3.0	api-version=2020-30-20	Ajout de la sécurité de point de terminaison (pare-feu IP et intégration avec <a href="#">Azure Private Link</a> )
2.0	api-version=2019-10-01	Améliorations de la convivialité. Rupture de la modification de la <a href="#">Liste des clés de requête</a> (GET n'existe plus).
1.0	api-version=2015-08-19	Première version

## Mise à niveau

1. Mettez à jour vos références NuGet `Microsoft.Azure.Management.Search`, soit en utilisant la console NuGet Package, soit en effectuant un clic droit sur les références de votre projet et en sélectionnant « Gérer les packages NuGet » dans Visual Studio.
2. Une fois que NuGet a téléchargé les nouveaux packages et leurs dépendances, recompilez votre projet. En fonction de la structure de votre code, la recopilation peut réussir, auquel cas vous avez terminé.
3. Si votre build échoue, cela est peut-être dû au fait que vous avez implémenté certaines des interfaces du Kit SDK (par exemple, à des fins de tests unitaires) qui ont été modifiées. Pour résoudre ce problème, vous devez implémenter de nouvelles méthodes telles que `BeginCreateOrUpdateWithHttpMessagesAsync`.
4. Une fois les erreurs de build résolues, vous pouvez apporter des modifications à votre application pour bénéficier des nouvelles fonctionnalités.

## Mettre à niveau vers la version 3.0

La version 3.0 ajoute la protection de point de terminaison privée en limitant l'accès à des plages d'adresses IP, et en intégrant éventuellement Azure Private Link pour les services de recherche qui ne doivent pas être visibles sur l'Internet public.

### Nouvelles API

API	CATEGORY	DÉTAILS
<a href="#">NetworkRuleSet</a>	Pare-feu IP	Limitez l'accès à un point de terminaison de service à une liste d'adresses IP autorisées. Pour plus d'informations sur les concepts et pour savoir comment procéder dans le portail, consultez <a href="#">Configurer le pare-feu IP</a> .
<a href="#">Ressource de liaison privée partagée</a>	Private Link	Créez une ressource de liaison privée partagée qui sera utilisée par un service de recherche.
<a href="#">Connexions de point de terminaison privé</a>	Private Link	Établissez et gérez les connexions à un service de recherche par le biais d'un point de terminaison privé. Pour plus d'informations sur les concepts et pour savoir comment procéder dans le portail, consultez <a href="#">Créer un point de terminaison privé</a> .
<a href="#">Ressource de liaison privée</a>	Private Link	Pour un service de recherche qui a une connexion de point de terminaison privé, obtenez la liste de tous les services utilisés sur le même réseau virtuel. Si votre solution de recherche comprend des indexeurs qui extraient des données à partir de sources Azure (Stockage Azure, Cosmos DB, Azure SQL), ou qui utilisent Cognitive Services ou Key Vault, toutes ces ressources doivent avoir des points de terminaison sur le réseau virtuel, et cette API doit retourner une liste.
<a href="#">PublicNetworkAccess</a>	Private Link	Il s'agit d'une propriété sur les requêtes de création ou de mise à jour de service. Quand elle est désactivée, la liaison privée est la seule modalité d'accès.

### Changements cassants

Vous ne pouvez plus utiliser GET sur une requête [Lister les clés de requête](#). Dans les versions précédentes, vous pouviez utiliser GET ou POST. Dans cette version, et dans toutes les versions ultérieures, seul POST est pris en charge.

## Mettre à niveau vers la version 2.0

La version 2 du SDK .NET Management de Recherche Azure est une mise à niveau mineure. La modification de votre code ne devrait donc nécessiter que peu d'efforts. Les modifications apportées au SDK sont strictement des modifications côté client destinées à améliorer la convivialité du SDK. Elles incluent notamment les suivantes :

- `Services.CreateOrUpdate` et ses versions asynchrones interrogeront désormais automatiquement le provisionnement `SearchService` et ne retourneront aucune donnée avant la fin du provisionnement du service. Cela vous évite d'avoir à écrire vous-même ce code d'interrogation.
- Si vous souhaitez continuer à interroger manuellement le provisionnement du service, vous pouvez utiliser la nouvelle méthode `Services.BeginCreateOrUpdate` ou l'une de ses versions asynchrones.

- De nouvelles méthodes `Services.Update` et leurs versions asynchrones ont été ajoutées au Kit SDK. Ces méthodes utilisent HTTP PATCH pour prendre en charge la mise à jour incrémentielle d'un service. Par exemple, vous pouvez désormais mettre à l'échelle un service en appliquant ces méthodes à une instance `SearchService` contenant uniquement les propriétés `partitionCount` et `replicaCount` souhaitées. L'ancienne manière d'appeler `Services.Get`, qui modifiait l'instance `SearchService` renournée et lui appliquait `Services.CreateOrUpdate`, est toujours prise en charge, mais elle n'est plus nécessaire.

## Étapes suivantes

Si vous rencontrez des problèmes, le meilleur forum pour publier des questions est [Stack Overflow](#). Si vous trouvez un bogue, vous pouvez signaler un problème dans le [Référentiel GitHub du Kit de développement logiciel \(SDK\) Azure .NET](#). N'oubliez pas d'étiqueter le titre de votre problème avec « [Search] ».

# Effectuer une mise à niveau vers la dernière API REST dans Recherche cognitive Azure

04/10/2020 • 14 minutes to read • [Edit Online](#)

Si vous utilisez une version antérieure de l'[API REST du service Recherche](#), cet article vous aidera à mettre à niveau votre application pour utiliser la toute dernière version de l'API en disponibilité générale, **2020-06-30**.

La version 2020-06-30 inclut une nouvelle fonctionnalité importante ([magasin de connaissances](#)) et introduit plusieurs modifications de comportement mineures. Par conséquent, cette version est essentiellement à compatibilité descendante. Les modifications du code doivent donc être minimales si vous effectuez une mise à niveau à partir de la version précédente (2019-05-06).

## NOTE

Un service de recherche prend en charge un éventail de versions de l'API REST, dont des versions plus récentes. Vous pouvez continuer à utiliser ces versions d'API, mais nous vous recommandons de migrer votre code vers la dernière version. Vous pourrez ainsi accéder aux nouvelles fonctionnalités. Au fil du temps, les versions les plus anciennes de l'API REST seront déconseillées et ne seront [plus prises en charge](#).

## Mise à niveau

Lors de la mise à niveau vers une nouvelle version, vous n'aurez probablement pas à modifier beaucoup votre code, en dehors du numéro de version. Les seules situations dans lesquelles vous pouvez avoir à modifier votre code sont les suivantes :

- Lorsque votre code échoue, car des propriétés non reconnues sont renvoyées dans une réponse de l'API. Par défaut, votre application doit ignorer les propriétés qu'elle ne comprend pas.
- Votre code conserve des demandes d'API et tente de les renvoyer à la nouvelle version de l'API. Par exemple, cela peut se produire si votre application conserve les jetons de continuation renvoyés par l'API Recherche (pour plus d'informations, recherchez `@search.nextPageParameters` dans les [références sur l'API Recherche](#)).
- Votre code fait référence à une version d'API antérieure à la version 2019-05-06 et soumise à un ou plusieurs changements cassants de cette version. La section [Mise à niveau vers la version 2019-05-06](#) fournit plus de détails.

Si vous êtes concerné par l'une de ces situations, vous aurez peut-être à modifier votre code en conséquence. Dans le cas contraire, aucune modification n'est nécessaire, sauf si vous souhaitez commencer à utiliser les fonctionnalités ajoutées dans la nouvelle version.

## Mise à niveau vers la version 2020-06-30

La version 2020-06-30 est la nouvelle version de l'API REST en disponibilité générale. Il n'y a pas de changements cassants, mais il existe quelques différences de comportement.

Les fonctionnalités désormais en disponibilité générale dans cette version de l'API sont notamment :

- La [base de connaissances](#), un stockage persistant de contenu enrichi créé via des ensembles de compétences, pour l'analyse et le traitement en aval par le biais d'autres applications. Avec cette fonctionnalité, un pipeline d'enrichissement par IA piloté par un indexeur peut remplir une base de connaissances en plus d'un index de

recherche. Si vous avez utilisé la version préliminaire de cette fonctionnalité, elle équivaut à la version en disponibilité générale. La seule modification de code requise est celle de la version de l'API.

Voici certains des changements de comportement :

- L'[algorithme de classement BM25](#) remplace l'algorithme de classement précédent par une technologie plus récente. Les nouveaux services utiliseront cet algorithme automatiquement. Pour les services existants, vous devez définir des paramètres pour utiliser le nouvel algorithme.
- Les résultats ordonnés pour les valeurs null ont été modifiés dans cette version ; les valeurs null apparaissent en premier si le tri est `asc` et en dernier si le tri est `desc`. Si vous avez écrit du code pour gérer le mode de tri des valeurs null, tenez compte de cette modification.

## Mise à niveau vers la version 2019-05-06

La version 2019-05-06 est la version précédente de l'API REST en disponibilité générale. Les fonctionnalités désormais en disponibilité générale dans cette version de l'API sont notamment :

- [L'autocomplétion](#) est une fonctionnalité prédictive qui complète une entrée de terme partiellement saisie.
- Les [types complexes](#) assurent la prise en charge native des données d'objet structurées dans un index de recherche.
- Les [modes d'analyse JsonLines](#), inclus dans l'indexation des objets Blob Azure, créent un document de recherche par entité JSON, séparé par un saut de ligne.
- L'[enrichissement par IA](#) assure une indexation qui tire parti des moteurs d'enrichissement IA de Cognitive Services.

### Changements cassants

Le code existant écrit sur des versions d'API antérieures s'arrête sur `api-version=2019-05-06` et version ultérieure si le code contient la fonctionnalité suivante :

**Indexeur pour Azure Cosmos DB : la source de données est désormais "type": "cosmosdb".**

Si vous utilisez un [indexeur Cosmos DB](#), vous devez remplacer `"type": "documentdb"` par `"type": "cosmosdb"`.

**Les erreurs de résultat d'exécution de l'indexeur n'ont plus d'état.**

La structure d'erreur pour l'exécution de l'indexeur comportait précédemment un élément `status`. Cet élément a été supprimé, car il ne fournissait pas d'informations utiles.

**L'API de source de données de l'indexeur ne renvoie plus de chaînes de connexion.**

À partir des versions d'API 2019-05-06 et 2019-05-06-Preview, l'API de source de données ne renvoie plus de chaînes de connexion dans la réponse d'une opération REST, quelle qu'elle soit. Dans les versions d'API précédentes, pour les sources de données créées à l'aide de POST, la Recherche cognitive Azure retournait 201, suivi de la réponse OData, qui contenait la chaîne de connexion en texte brut.

**La compétence cognitive Reconnaissance d'entité nommée n'est plus proposée.**

Si vous avez appelé la compétence [Reconnaissance d'entité nommée](#) dans votre code, l'appel échoue. La fonctionnalité de remplacement est [Reconnaissance d'entité](#). Vous devriez pouvoir remplacer la référence de compétence sans modification supplémentaire. La signature de l'API est la même pour les deux versions.

### Mise à niveau des types complexes

La version d'API 2019-05-06 contient désormais une prise en charge formelle des types complexes. Si votre code implémentait des suggestions précédentes pour l'équivalence de type complexe dans 2017-11-11-Preview ou 2016-09-01-Preview, il existe de nouvelles limites et des limites modifiées à partir de la version 2019-05-06 que vous devez connaître :

- Les limites relatives à la profondeur des sous-champs et au nombre de collections complexes par index ont été réduites. Si vous avez créé des index qui dépassent ces limites à l'aide de versions d'API en préversion, toute tentative de mise à jour ou de recréation des index à l'aide de la version d'API 2019-05-06 échouera.

Si tel est votre cas, vous devrez reconcevoir votre schéma de sorte qu'il respecte les nouvelles limites, puis reconstruire votre index.

- Il existe une nouvelle limite à partir de la version d'API 2019-05-06 concernant le nombre d'éléments de collections complexes par document. Si vous avez créé des index avec des documents qui dépassent ces limites à l'aide de versions d'API en préversion, toute tentative de réindexation de ces données à l'aide de la version d'API 2019-05-06 échouera. Si tel est votre cas, vous devrez réduire le nombre d'éléments de collections complexes par document avant de réindexer les données.

Pour plus d'informations, consultez [Limites de service de la Recherche cognitive Azure](#).

#### Comment mettre à niveau une ancienne structure de type complexe

Si votre code utilise des types complexes avec l'une des anciennes versions d'API en préversion, vous utilisez peut-être un format de définition d'index qui se présente ainsi :

```
{
  "name": "hotels",
  "fields": [
    { "name": "HotelId", "type": "Edm.String", "key": true, "filterable": true },
    { "name": "HotelName", "type": "Edm.String", "searchable": true, "filterable": false, "sortable": true,
      "facetable": false },
    { "name": "Description", "type": "Edm.String", "searchable": true, "filterable": false, "sortable": false,
      "facetable": false, "analyzer": "en.microsoft" },
    { "name": "Description_fr", "type": "Edm.String", "searchable": true, "filterable": false, "sortable": false,
      "facetable": false, "analyzer": "fr.microsoft" },
    { "name": "Category", "type": "Edm.String", "searchable": true, "filterable": true, "sortable": true,
      "facetable": true },
    { "name": "Tags", "type": "Collection(Edm.String)", "searchable": true, "filterable": true, "sortable": false,
      "facetable": true, "analyzer": "tagsAnalyzer" },
    { "name": "ParkingIncluded", "type": "Edm.Boolean", "filterable": true, "sortable": true, "facetable": true },
    { "name": "LastRenovationDate", "type": "Edm.DateTimeOffset", "filterable": true, "sortable": true,
      "facetable": true },
    { "name": "Rating", "type": "Edm.Double", "filterable": true, "sortable": true, "facetable": true },
    { "name": "Address", "type": "Edm.ComplexType" },
    { "name": "Address/StreetAddress", "type": "Edm.String", "filterable": false, "sortable": false,
      "facetable": false, "searchable": true },
    { "name": "Address/City", "type": "Edm.String", "searchable": true, "filterable": true, "sortable": true,
      "facetable": true },
    { "name": "Address/StateProvince", "type": "Edm.String", "searchable": true, "filterable": true,
      "sortable": true, "facetable": true },
    { "name": "Address/PostalCode", "type": "Edm.String", "searchable": true, "filterable": true, "sortable": true,
      "facetable": true },
    { "name": "Address/Country", "type": "Edm.String", "searchable": true, "filterable": true, "sortable": true,
      "facetable": true },
    { "name": "Location", "type": "Edm.GeographyPoint", "filterable": true, "sortable": true },
    { "name": "Rooms", "type": "Collection(Edm.ComplexType)" },
    { "name": "Rooms/Description", "type": "Edm.String", "searchable": true, "filterable": false, "sortable": false,
      "facetable": false, "analyzer": "en.lucene" },
    { "name": "Rooms/Description_fr", "type": "Edm.String", "searchable": true, "filterable": false,
      "sortable": false, "facetable": false, "analyzer": "fr.lucene" },
    { "name": "Rooms/Type", "type": "Edm.String", "searchable": true },
    { "name": "Rooms/BaseRate", "type": "Edm.Double", "filterable": true, "facetable": true },
    { "name": "Rooms/BedOptions", "type": "Edm.String", "searchable": true },
    { "name": "Rooms/SleepsCount", "type": "Edm.Int32", "filterable": true, "facetable": true },
    { "name": "Rooms/SmokingAllowed", "type": "Edm.Boolean", "filterable": true, "facetable": true },
    { "name": "Rooms/Tags", "type": "Collection(Edm.String)", "searchable": true, "filterable": true,
      "facetable": true, "analyzer": "tagsAnalyzer" }
  ]
}
```

Un format plus récent de type arborescence pour la définition de champs d'index a été introduit dans la version d'API 2017-11-11-Preview. Avec le nouveau format, chaque champ complexe a une collection de champs dont les

sous-champs sont définis. Dans la version d'API 2019-05-06, ce nouveau format est utilisé exclusivement. Toute tentative de création ou de mise à jour d'un index à l'aide de l'ancien format échouera. Si vous avez créé des index à l'aide de l'ancien format, vous devez utiliser la version d'API 2017-11-11-Preview pour les mettre à jour vers le nouveau format, de sorte qu'ils puissent être gérés avec la version d'API 2019-05-06.

Vous pouvez mettre à jour des index « plats » vers le nouveau format en procédant comme suit avec la version d'API 2017-11-11-Preview :

1. Effectuez une requête GET pour récupérer votre index. S'il est déjà au nouveau format, vous avez terminé.
2. Traduisez l'index à partir du format « plat » vers le nouveau format. Vous devrez écrire du code à cette fin.  
En effet, aucun exemple de code n'est disponible au moment de la rédaction de ce document.
3. Effectuez une requête PUT pour mettre à jour l'index vers le nouveau format. Veillez à ne modifier aucune autre information de l'index, telle que la possibilité de filtrer ou d'effectuer des recherches dans les champs.  
En effet, cela n'est pas autorisé par l'API de mise à jour d'index.

#### NOTE

Il n'est pas possible de gérer les index créés avec l'ancien format « plat » à partir du Portail Azure. Mettez à niveau vos index à partir de la représentation « plate » vers la représentation de type « arborescence » dès que possible.

## Étapes suivantes

Consultez la documentation de référence relative à l'API REST du service Recherche. Si vous rencontrez des problèmes, sollicitez notre aide sur [Stack Overflow](#) ou [contactez le support](#).

[Référence de l'API REST du service Recherche](#)

# Configurer des clés gérées par le client pour le chiffrement des données dans le service Recherche cognitive Azure

04/10/2020 • 24 minutes to read • [Edit Online](#)

Par défaut, le service Recherche cognitive Azure chiffre automatiquement le contenu indexé au repos avec des [clés gérées par le service](#). Si vous avez besoin de davantage de protection, vous pouvez compléter le chiffrement par défaut avec une couche de chiffrement supplémentaire à l'aide de clés que vous créez et gérez dans Azure Key Vault. Cet article vous guide tout au long des étapes de configuration du chiffrement à l'aide de clés gérées par le client (CMK).

Le chiffrement CMK dépend d'[Azure Key Vault](#). Vous pouvez créer vos propres clés de chiffrement et les stocker dans un coffre de clés, ou utiliser les API d'Azure Key Vault pour générer des clés de chiffrement. Azure Key Vault vous permet également d'auditer l'utilisation des clés si vous [activez la journalisation](#).

Le chiffrement avec des clés gérées par le client est appliqué à des index individuels ou à des mappages de synonymes lors de la création de ces objets, et n'est pas spécifié au niveau du service de recherche lui-même. Seuls de nouveaux objets peuvent être chiffrés. Vous ne pouvez pas chiffrer un contenu existant.

Toutes les clés ne doivent pas nécessairement se trouver dans le même coffre de clés. Un service de recherche unique peut héberger plusieurs index chiffrés ou mappages de synonymes chiffrés avec leurs propres clés de chiffrement gérées par le client et stockées dans différents coffres de clés. Vous pouvez également avoir des index et des cartes de synonymes dans le même service qui ne sont pas chiffrés à l'aide de clés gérées par le client.

## Double chiffrement

Pour les services créés après le 1er août 2020 et dans certaines régions, l'étendue du chiffrement CMK inclut des disques temporaires, ce qui permet un [double chiffrement complet](#) actuellement disponible dans les régions suivantes :

- USA Ouest 2
- USA Est
- États-Unis - partie centrale méridionale
- Gouvernement américain - Virginie
- Gouvernement des États-Unis – Arizona

Si vous utilisez une autre région ou un service créé avant le 1er août, votre chiffrement CMK est limité uniquement au disque de données, à l'exclusion des disques temporaires que le service utilise.

## Prérequis

Les services suivants sont utilisés dans cet exemple.

- [Créez un service Recherche cognitive Azure](#) ou [recherchez un service existant](#).
- [Créez une ressource Azure Key Vault](#) ou recherchez un coffre existant dans le même abonnement que le service Recherche Cognitive Azure. Cette fonctionnalité a la même exigence en matière d'abonnement.
- [Azure PowerShell](#) ou [Azure CLI](#) est utilisé pour les tâches de configuration.

- Vous pouvez utiliser [Postman](#), [Azure PowerShell](#) et la [préversion du Kit de développement logiciel \(SDK .NET\)](#) pour appeler l'API REST qui crée des index et des mappages de synonymes incluant un paramètre de clé de chiffrement. Il n'existe actuellement pas de prise en charge du portail pour l'ajout de clé aux index ou aux mappages de synonymes.

#### NOTE

En raison de la nature du chiffrement avec des clés gérées par le client, le service Recherche cognitive Azure ne pourra pas récupérer vos données en cas de suppression de votre clé Azure Key Vault. Pour éviter la perte de données résultant de suppressions accidentelles d'Azure Key Vault, vous devez activer la suppression réversible et la protection contre le vidage sur le coffre de clés. La suppression réversible étant activée par défaut, vous ne rencontrerez des problèmes que si vous l'avez désactivée intentionnellement. La par défaut n'est pas activée par défaut, mais elle est requise pour le chiffrement CMK du service Recherche cognitive Azure. Pour plus d'informations, consultez les vues d'ensemble de la [suppression réversible](#) et de la [protection contre le vidage](#).

## 1 - Activer la récupération de clé

La [suppression réversible](#) et la [protection contre le vidage](#) doivent être activées pour le coffre de clés. Vous pouvez définir ces fonctionnalités à l'aide du portail ou des commandes PowerShell ou Azure CLI suivantes.

#### Utilisation de PowerShell

1. Exécutez `Connect-AzAccount` pour configurer vos informations d'identification Azure.
2. Exécutez la commande suivante pour vous connecter à votre coffre de clés, en remplaçant `<vault_name>` par un nom valide :

```
$resource = Get-AzResource -ResourceId (Get-AzKeyVault -VaultName "<vault_name>").ResourceId
```

3. Azure Key Vault est créé avec la suppression réversible activée. Si elle est désactivée sur votre coffre, exécutez la commande suivante :

```
$resource.Properties | Add-Member -MemberType NoteProperty -Name "enableSoftDelete" -Value 'true'
```

4. Activer la protection contre le vidage :

```
$resource.Properties | Add-Member -MemberType NoteProperty -Name "enablePurgeProtection" -Value 'true'
```

5. Enregistrez vos mises à jour :

```
Set-AzResource -resourceid $resource.ResourceId -Properties $resource.Properties
```

#### Utilisation de l'interface de ligne de commande Azure

```
az keyvault update -n <vault_name> -g <resource_group> --enable-soft-delete --enable-purge-protection
```

## 2 - Créer une clé

Si vous utilisez une clé existante pour chiffrer le contenu de Recherche cognitive Azure, ignorez cette étape.

1. Connectez-vous au portail Azure et ouvrez la page de présentation de votre coffre de clés.
2. Sélectionnez le paramètre Clés dans le volet de navigation de gauche, puis cliquez sur Générer/Importer.
3. Dans le volet Créer une clé, dans la liste Options, choisissez la méthode que vous voulez utiliser pour créer une clé. Vous pouvez Générer une nouvelle clé, Charger une clé existante ou utiliser Restaurer la sauvegarde pour sélectionner une sauvegarde d'une clé.
4. Entrez un nom pour votre clé et sélectionnez éventuellement d'autres propriétés de clé.
5. Cliquez sur le bouton Créer pour démarrer le déploiement.

Notez l'identificateur de la clé : il se compose de l'**URI de la valeur de la clé**, du **nom de la clé** et de la **version de la clé**. Vous en aurez besoin pour définir un index chiffré dans Recherche cognitive Azure.

The screenshot shows the 'Create a key' blade in the Azure portal. At the top, it displays the key identifier: 208d83db2625450cba22f5bd3ec38027, with a note below it: 'Key Version'. Below this are 'Save' and 'Discard' buttons. The main area is titled 'Properties' and shows the following details:

- Key Type:** RSA
- Created:** 22 h ago
- Updated:** 22 h ago

Under 'Key Identifier', the URL <https://asvault01.vault.local.azurestack.external/keys/ASKey01/208d83db2625450cba22f5bd3ec38027> is displayed, with a copy icon next to it. This URL is highlighted with a red box.

In the 'Settings' section, there are checkboxes for 'Set activation date.' and 'Set expiration date.', both of which are unchecked. Below these is a 'Enabled' switch, which is set to 'Yes' (highlighted in blue).

The 'Tags' section shows '0 tags' with a 'View' button.

Under 'Permitted operations', the following checkboxes are checked:

Operation	Status
Encrypt	<input checked="" type="checkbox"/>
Decrypt	<input checked="" type="checkbox"/>
Sign	<input checked="" type="checkbox"/>
Verify	<input checked="" type="checkbox"/>
Wrap Key	<input checked="" type="checkbox"/>
Unwrap Key	<input checked="" type="checkbox"/>

### 3 - Créez une identité de service

L'attribution d'une identité à votre service de recherche vous permet d'accorder des droits d'accès Key Vault à votre service de recherche. Votre service de recherche utilisera son identité pour s'authentifier auprès d'Azure Key Vault.

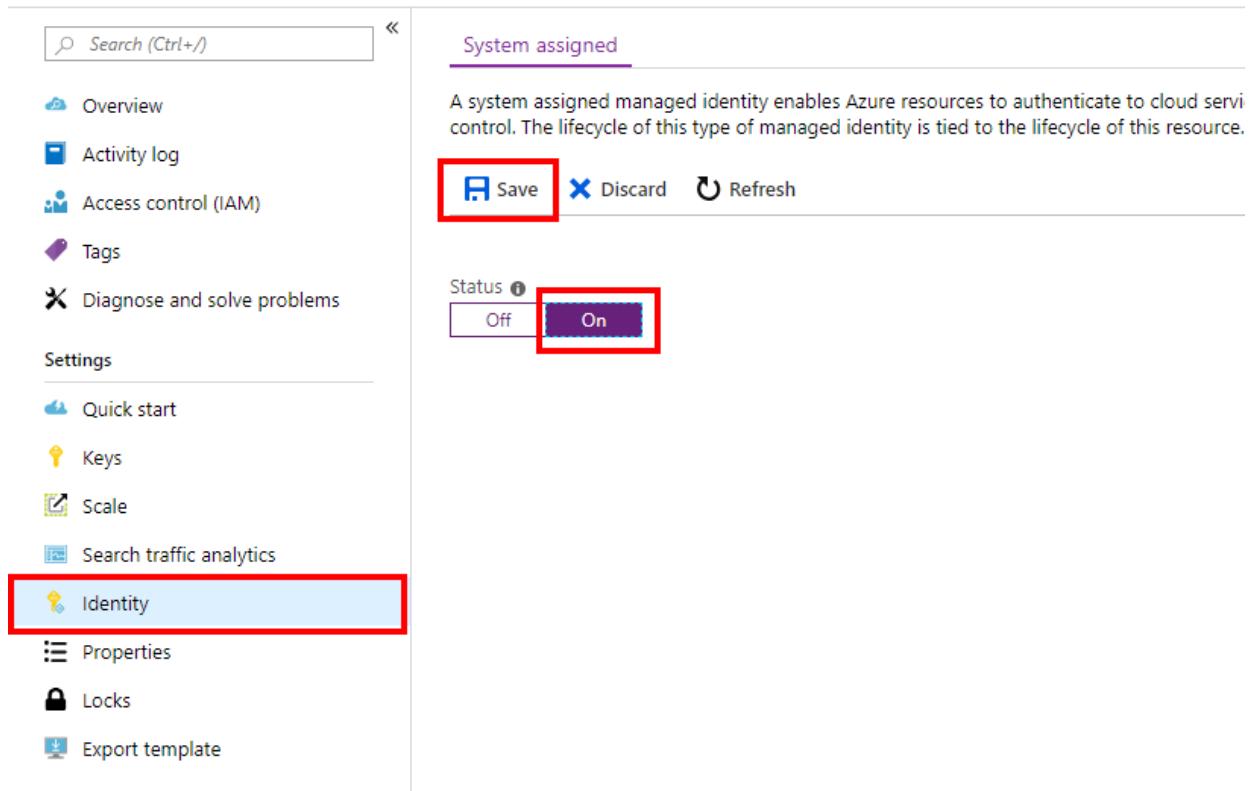
Recherche cognitive Azure prend en charge deux façons d'attribuer une identité : une identité managée ou une application Azure Active Directory managée en externe.

Si possible, utilisez une identité managée. C'est le moyen le plus simple d'attribuer une identité à votre service de recherche et le mieux adapté à la plupart des scénarios. Si vous utilisez plusieurs clés pour des index et des mappages de synonymes, ou si votre solution se trouve dans une architecture distribuée qui exclut

l'authentification basée sur l'identité, utilisez l'[approche Azure Active Directory managée en externe](#) avancée, décrite à la fin de cet article.

En général, une identité managée permet à votre service de recherche de s'authentifier auprès d'Azure Key Vault, sans stocker les informations d'identification dans le code. Le cycle de vie de ce type d'identité managée est lié au cycle de vie de votre service de recherche, qui ne peut avoir qu'une seule identité managée. [En savoir plus sur les identités managées.](#)

1. [Connectez-vous au portail Azure](#) et ouvrez la page de présentation de votre service de recherche.
2. Cliquez sur **Identité** dans le volet de navigation de gauche, définissez son statut sur **On** (Activé), puis cliquez sur **Enregistrer**.



The screenshot shows the Azure portal interface for managing a search service. On the left, there's a sidebar with various navigation options like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Quick start, Keys, Scale, Search traffic analytics, and Identity. The Identity option is highlighted with a red box. The main content area has a header 'System assigned' with a sub-header explaining it enables Azure resources to authenticate to cloud service control. It includes Save, Discard, and Refresh buttons. Below that is a 'Status' section with Off and On buttons, where 'On' is also highlighted with a red box. The overall theme is light blue and white.

## 4 - Accorder des autorisations d'accès à la clé

Pour activer votre service de recherche afin d'utiliser votre clé Key Vault, vous devrez accorder à votre service de recherche certaines autorisations d'accès.

Les autorisations d'accès peuvent être révoquées à tout moment. Une fois révoqué, tout index ou mappage de synonymes d'un service de recherche qui utilise ce coffre de clés sera inutilisable. Une restauration ultérieure des autorisations d'accès au coffre de clés restaurera l'accès à l'index ou au mappage de synonymes. Pour plus d'informations, consultez [Accès sécurisé à un coffre de clés](#).

1. [Connectez-vous au portail Azure](#) et ouvrez la page de présentation de votre coffre de clés.
2. Sélectionnez le paramètre **Stratégies d'accès** dans le volet de navigation à gauche, puis cliquez sur **+ Ajouter nouveau**.

DemoKeyVault - Access policies

Search (Ctrl+ /)

Save Discard Refresh

Click to show advanced access policies

Add new ...

- Overview
- Activity log
- Access control (IAM)
- Tags
- Diagnose and solve problems
- Settings
  - Keys
  - Secrets
  - Certificates
  - Access policies** (highlighted with a red box)
  - Firewalls and virtual networks
  - Properties
  - Locks
  - Export template

3. Cliquez sur **Sélectionner le principal** puis choisissez votre service Recherche cognitive Azure. Vous pouvez le rechercher par nom ou par l'ID d'objet affiché après avoir activé l'identité managée.

Home > DemoKeyVault - Access policies > Add access policy

Add access policy

Configure from template (optional)

Key permissions: 0 selected

Secret permissions: 0 selected

Certificate permissions: 0 selected

\* Select principal  
None selected

Authorized application: None selected

OK

Principal

Select a principal

+ Invite

Select my-search-service

my-search-service

Selected: None

Select

4. Cliquez sur **Autorisations de clé**, puis sélectionnez *Obtenir*, *Ne pas inclure la clé* et *Inclure la clé*. Vous pouvez utiliser le modèle *Azure Data Lake Storage* ou *Azure Storage* pour sélectionner rapidement les autorisations requises.

Recherche cognitive Azure doit obtenir les **autorisations d'accès** suivantes :

- *Obtenir*: permet à votre service de recherche de récupérer les parties publiques de votre clé dans un coffre de clés
- *Inclure la clé*: permet à votre service de recherche d'utiliser votre clé pour protéger la clé de chiffrement interne
- *Ne pas inclure la clé*: permet à votre service de recherche d'utiliser votre clé pour désencapsuler la clé

de chiffrement interne

#### Add access policy

Add a new access policy

Configure from template (optional)

Azure Data Lake Storage or Azure Storage

\* Select principal  
my-search-service

Key permissions

3 selected

Select all

##### Key Management Operations

Get

List

Update

Create

Import

Delete

Recover

Backup

Restore

##### Cryptographic Operations

Decrypt

Encrypt

Unwrap Key

Wrap Key

Verify

Sign

##### Privileged Key Operations

Purge

OK

5. Dans **Autorisations du secret**, sélectionnez *Get*.

6. Pour **Autorisations de certificat**, sélectionnez *Get*.

7. Cliquez sur **OK** puis sur **Enregistrer** pour enregistrer les changements de stratégie d'accès.

#### IMPORTANT

Le contenu chiffré dans Recherche cognitive Azure est configuré pour utiliser une clé Azure Key Vault spécifique avec une **version** spécifique. Si vous modifiez la clé ou la version, l'index ou le mappage de synonymes doit être mis à jour pour utiliser la nouvelle clé\version **avant** de supprimer la clé\version précédente. Si vous ne le faites pas, l'index ou le mappage de synonymes deviendra inutilisable et vous ne pourrez pas déchiffrer le contenu une fois que l'accès aux clés sera perdu.

## 5 - Chiffrer le contenu

Pour ajouter une clé gérée par le client sur un index ou un mappage de synonymes, vous devez utiliser l'[API REST Recherche](#) ou un Kit de développement logiciel (SDK). Le portail n'expose pas les mappages de synonymes ou les propriétés de chiffrement. Lorsque vous utilisez une API valide, les index et les mappages de synonymes prennent en charge une propriété **encryptionKey** de niveau supérieur.

En utilisant l'**URI du coffre de clés**, le **nom de la clé** et la **version de la clé** de votre clé de coffre de clés, nous pouvons créer une définition **encryptionKey** comme suit :

```
{  
  "encryptionKey": {  
    "keyVaultUri": "https://demokeyvault.vault.azure.net",  
    "keyVaultKeyName": "myEncryptionKey",  
    "keyVaultKeyVersion": "eaab6a663d59439ebb95ce2fe7d5f660"  
  }  
}
```

#### NOTE

Aucune de ces informations sur le coffre de clés n'est considérée comme secrète et peut être facilement récupérée en accédant à la page de la clé Azure Key Vault appropriée dans le portail Azure.

Si vous utilisez une application AAD pour l'authentification Key Vault au lieu d'utiliser une identité managée, ajoutez à votre clé de chiffrement les **informations d'identification d'accès** de l'application AAD :

```
{  
  "encryptionKey": {  
    "keyVaultUri": "https://demokeyvault.vault.azure.net",  
    "keyVaultKeyName": "myEncryptionKey",  
    "keyVaultKeyVersion": "eaab6a663d59439ebb95ce2fe7d5f660",  
    "accessCredentials": {  
      "applicationId": "00000000-0000-0000-0000-000000000000",  
      "applicationSecret": "myApplicationSecret"  
    }  
  }  
}
```

## Exemple : Chiffrement d'index

Les détails de la création d'un nouvel index via l'API REST se trouvent dans la section [Créer un index \(API REST du service Recherche cognitive Azure\)](#), où la seule différence ici consiste à spécifier les détails de la clé de chiffrement dans la définition de l'index :

```
{
  "name": "hotels",
  "fields": [
    {"name": "HotelId", "type": "Edm.String", "key": true, "filterable": true},
    {"name": "HotelName", "type": "Edm.String", "searchable": true, "filterable": false, "sortable": true,
     "facetable": false},
    {"name": "Description", "type": "Edm.String", "searchable": true, "filterable": false, "sortable": false,
     "facetable": false, "analyzer": "en.lucene"},
    {"name": "Description_fr", "type": "Edm.String", "searchable": true, "filterable": false, "sortable": false,
     "facetable": false, "analyzer": "fr.lucene"},
    {"name": "Category", "type": "Edm.String", "searchable": true, "filterable": true, "sortable": true,
     "facetable": true},
    {"name": "Tags", "type": "Collection(Edm.String)", "searchable": true, "filterable": true, "sortable": false,
     "facetable": true},
    {"name": "ParkingIncluded", "type": "Edm.Boolean", "filterable": true, "sortable": true, "facetable": true},
    {"name": "LastRenovationDate", "type": "Edm.DateTimeOffset", "filterable": true, "sortable": true,
     "facetable": true},
    {"name": "Rating", "type": "Edm.Double", "filterable": true, "sortable": true, "facetable": true},
    {"name": "Location", "type": "Edm.GeographyPoint", "filterable": true, "sortable": true},
  ],
  "encryptionKey": {
    "keyVaultUri": "https://demokeyvault.vault.azure.net",
    "keyVaultKeyName": "myEncryptionKey",
    "keyVaultKeyVersion": "eaab6a663d59439ebb95ce2fe7d5f660"
  }
}
```

Vous pouvez maintenant envoyer la demande de création d'un index, puis commencer à utiliser l'index normalement.

## Exemple : Chiffrement de mappage de synonymes

Les détails de la création d'un nouveau mappage de synonymes via l'API REST se trouvent dans la section [Créer un mappage de synonymes \(API REST du service Recherche cognitive Azure\)](#), où la seule différence ici consiste à spécifier les détails de la clé de chiffrement dans la définition du mappage de synonymes :

```
{
  "name" : "synonymmap1",
  "format" : "solr",
  "synonyms" : "United States, United States of America, USA\n
Washington, Wash. => WA",
  "encryptionKey": {
    "keyVaultUri": "https://demokeyvault.vault.azure.net",
    "keyVaultKeyName": "myEncryptionKey",
    "keyVaultKeyVersion": "eaab6a663d59439ebb95ce2fe7d5f660"
  }
}
```

Vous pouvez maintenant envoyer la demande de création d'un mappage de synonymes, puis commencer à utiliser l'index normalement.

#### **IMPORTANT**

Même si la propriété **encryptionKey** peut être ajoutée aux index ou mappages de synonymes Recherche cognitive Azure existants, elle peut être mise à jour en fournissant des valeurs différentes pour chacun des trois informations du coffre de clés (par exemple, en mettant à jour la version de la clé). Lorsque vous passez à une nouvelle clé Key Vault ou à une nouvelle version de clé, tout index ou mappage de synonymes Recherche cognitive Azure qui utilise la clé doit d'abord être mis à jour pour utiliser la nouvelle clé **avant** de supprimer la clé\version précédente. Si vous ne le faites pas, l'index ou le mappage de synonymes deviendra inutilisable et il ne pourra pas déchiffrer le contenu une fois que l'accès aux clés sera perdu.

Une restauration ultérieure des autorisations d'accès au coffre de clés restaurera l'accès au contenu.

## Avancé : Utiliser une application Azure Active Directory managée en externe

Lorsqu'une identité managée n'est pas possible, vous pouvez créer une application Azure Active Directory avec un principal de sécurité pour votre service Recherche cognitive Azure. Plus précisément, une identité managée n'est pas viable dans ces conditions :

- Vous ne pouvez pas accorder directement à votre service de recherche les droits d'accès au coffre de clés (par exemple, si le service de recherche se trouve dans un autre locataire Active Directory qu'Azure Key Vault).
- Un seul service de recherche est nécessaire pour héberger plusieurs index/mappages de synonymes chiffrés, chacun utilisant une clé différente d'un coffre de clés différent, où chaque coffre de clés doit utiliser **une identité différente** pour l'authentification. Si l'utilisation d'une identité différente pour gérer différents coffres de clés n'est pas une exigence, vous pouvez utiliser l'option d'identité managée ci-dessus.

Pour s'adapter à de telles topologies, Recherche cognitive Azure prend en charge la recherche à l'aide d'applications Azure Active Directory (AAD) pour l'authentification entre votre service de recherche et Key Vault. Pour créer une application AAD dans le portail :

1. [Créez une application Azure Active Directory](#).
2. [Obtenez l'ID de l'application et la clé d'authentification](#) car ces informations seront nécessaires pour créer un index chiffré. Vous devrez également fournir l'**ID d'application** et la **clé d'authentification**.

#### **IMPORTANT**

Lorsque vous décidez d'utiliser une application AAD d'authentification au lieu d'une identité managée, considérez le fait que Recherche cognitive Azure n'est pas autorisé à gérer votre application AAD en votre nom, et c'est à vous de gérer votre application AAD, par exemple la rotation périodique de la clé d'authentification de l'application. Lors du changement d'une application AAD ou de sa clé d'authentification, tout index ou mappage de synonymes Recherche cognitive Azure qui utilise cette application doit d'abord être mis à jour pour utiliser le nouvel ID d'application **avant** de supprimer l'application précédente ou sa clé d'autorisation, et avant de révoquer votre accès Key Vault. Si vous ne le faites pas, l'index ou le mappage de synonymes deviendra inutilisable et il ne pourra pas déchiffrer le contenu une fois que l'accès aux clés sera perdu.

## Utiliser des colonnes chiffrées

Avec le chiffrement CMK, vous remarquerez une latence pour l'indexation et les requêtes en raison du travail de chiffrement/déchiffrement supplémentaire. Le service Recherche cognitive Azure ne journalise pas l'activité de chiffrement, mais vous pouvez surveiller l'accès aux clés par le biais de la journalisation du coffre de clés. Nous vous recommandons d'[activer la journalisation](#) dans le cadre de la configuration du coffre de clés.

Une rotation de clés est supposée se produire au fil du temps. Chaque fois que vous voulez opérer une rotation de clés, il est important de suivre cette séquence :

1. [Déterminez la clé utilisée par un index ou un mappage de synonyme.](#)
2. [Créez une clé dans le coffre de clés](#), mais gardez la clé d'origine disponible.
3. [Mettez à jour les propriétés encryptionKey](#) sur un index ou un mappage de synonymes pour utiliser les nouvelles valeurs. Seuls des objets créés à l'origine avec cette propriété peuvent être mis à jour pour utiliser une autre valeur.
4. Désactivez ou supprimez la clé précédente dans le coffre de clés. Surveillez l'accès à la clé pour vérifier que la nouvelle clé est utilisée.

Pour des raisons de performances, le service de recherche met en cache la clé pendant plusieurs heures. Si vous désactivez ou supprimez la clé sans en fournir de nouvelle, les requêtes continuent de fonctionner sur une base temporaire jusqu'à ce que le cache expire. Toutefois, une fois que le service de recherche ne peut pas déchiffrer le contenu, vous obtenez le message suivant : « Accès interdit. La clé de requête utilisée a peut-être été révoquée. Réessayez. »

## Étapes suivantes

Si vous n'êtes pas familiarisé avec l'architecture de sécurité Azure, passez en revue la [documentation sur la sécurité Azure](#), et en particulier cet article :

[Chiffrement des données Azure au repos](#)

# Obtenir des informations de clé gérée par le client à partir d'index et de synonym maps (cartes de synonymes)

04/10/2020 • 3 minutes to read • [Edit Online](#)

Dans Azure Cognitive Search, les clés de chiffrement gérées par le client sont créées, stockées et gérées dans Azure Key Vault. Si vous avez besoin de déterminer si un objet est chiffré ou si le nom ou la version de la clé a été utilisée, utilisez l'API REST ou un kit de développement logiciel (SDK) pour récupérer la propriété `encryptionKey` à partir d'un index ou d'une définition de synonym map.

Nous vous recommandons d'[activer la journalisation](#) sur Key Vault afin de pouvoir surveiller l'utilisation de clé.

## Obtient la clé de l'API administrateur

Pour obtenir des définitions d'objet à partir d'un service de recherche, vous devez vous authentifier avec des droits d'administrateur. Le moyen le plus simple d'accéder à la clé d'API d'administration se fait via le portail.

1. Connectez-vous au [portail Azure](#) et ouvrez la page de présentation de votre service de recherche.
2. Sur le côté gauche, cliquez sur Clés et copiez une API d'administration. Une clé d'administrateur est requise pour l'index et la récupération de carte des synonymes.

Pour les étapes restantes, basculez vers PowerShell et l'API REST. Le portail n'affiche pas les cartes de synonymes, ni les propriétés de clé de chiffrement des index.

## Utilisez PowerShell Core

Exécutez les commandes suivantes pour configurer les variables et récupérer les définitions d'objet.

```
<# Connect to Azure #>
$Connect-AzAccount

<# Provide the admin API key used for search service authentication #>
$headers = @{
    'api-key' = '<YOUR-ADMIN-API-KEY>'
    'Content-Type' = 'application/json'
    'Accept' = 'application/json' }

<# List all existing synonym maps #>
$url = 'https://<YOUR-SEARCH-SERVICE>.search.windows.net/synonyms?api-version=2020-06-30&$select=name'
Invoke-RestMethod -Uri $url -Headers $headers | ConvertTo-Json

<# List all existing indexes #>
$url = 'https://<YOUR-SEARCH-SERVICE>.search.windows.net/indexes?api-version=2020-06-30&$select=name'
Invoke-RestMethod -Uri $url -Headers $headers | ConvertTo-Json

<# Return a specific synonym map definition. The encryptionKey property is at the end #>
$url = 'https://<YOUR-SEARCH-SERVICE>.search.windows.net/synonyms/<YOUR-SYNONYM-MAP-NAME>?api-version=2020-06-30'
Invoke-RestMethod -Uri $url -Headers $headers | ConvertTo-Json

<# Return a specific index definition. The encryptionKey property is at the end #>
$url = 'https://<YOUR-SEARCH-SERVICE>.search.windows.net/indexes/<YOUR-INDEX-NAME>?api-version=2020-06-30'
Invoke-RestMethod -Uri $url -Headers $headers | ConvertTo-Json
```

## Étapes suivantes

Maintenant que vous connaissez la clé de chiffrement et la version utilisées, vous pouvez gérer la clé dans Azure Key Vault ou vérifier d'autres paramètres de configuration.

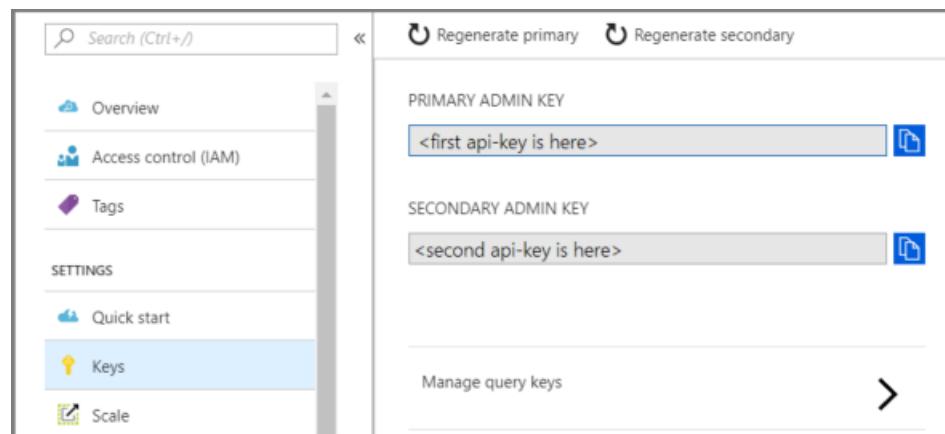
- [Démarrage rapide : Définir et récupérer un secret depuis Azure Key Vault à l'aide de PowerShell](#)
- [Configurer des clés gérées par le client pour le chiffrement des données dans le service Recherche cognitive Azure](#)

# Créer et gérer des clés API pour un service Recherche cognitive Azure

04/10/2020 • 11 minutes to read • [Edit Online](#)

Toutes les demandes adressées à un service de recherche ont besoin d'une clé API en lecture seule générée spécialement pour votre service. Cette clé API constitue le seul mécanisme d'authentification de l'accès au point de terminaison de votre service de recherche et doit être incluse dans chaque demande. Dans les [solutions REST](#), la clé API est généralement spécifiée dans un en-tête de demande. Dans les [solutions .NET](#), une clé est souvent spécifiée sous forme de paramètre de configuration, puis transmise en tant que [Credentials](#) (clé d'administration) ou de [SearchCredentials](#) (clé de requête) sur [SearchServiceClient](#).

Lors du provisionnement du service, les clés sont créées avec votre service de recherche. Vous pouvez afficher et obtenir des valeurs de clés dans le [portail Azure](#).



## Qu'est-ce qu'une clé API ?

Une clé API est une chaîne composée de nombres et de lettres générée de manière aléatoire. Par le biais des [autorisations basées sur le rôle](#), vous pouvez supprimer ou lire les clés, mais vous ne pouvez pas remplacer une clé avec un mot de passe défini par l'utilisateur ou utiliser Active Directory en tant que méthode d'authentification principale pour accéder aux opérations de recherche.

Deux types de clés sont utilisés pour accéder à votre service de recherche : administration (lecture-écriture) et requête (lecture seule).

CLÉ	DESCRIPTION	LIMITES
-----	-------------	---------

CLÉ	DESCRIPTION	LIMITES
Admin	<p>Accorde des droits d'accès complets à toutes les opérations, avec notamment la possibilité de gérer le service ou de créer et supprimer des index, des indexeurs et des sources de données.</p> <p>Deux clés d'administration, appelées clés <i>principale</i> et <i>secondaire</i> dans le portail, sont générées quand le service est créé et peuvent être régénérées individuellement à la demande. La possession de deux clés permet de substituer une clé quand l'autre est utilisée pour un accès continu au service.</p> <p>Les clés d'administration sont spécifiées uniquement dans les en-têtes de requête HTTP. Vous ne pouvez pas insérer de clé API d'administration dans une URL.</p>	2 max. par service
Requête	<p>Accorde un accès en lecture seule aux index et aux documents. Ces clés sont généralement distribuées aux applications clientes qui émettent des demandes de recherche.</p> <p>Les clés de requête sont créées à la demande. Vous pouvez les créer manuellement dans le portail ou par programme via l'<a href="#">API REST de gestion</a>.</p> <p>Les clés de requête peuvent être spécifiées dans un en-tête de requête HTTP pour les opérations de recherche, de suggestion ou de consultation. Vous pouvez également transmettre une clé de requête en tant que paramètre pour une URL. Selon la façon dont votre application cliente formule la demande, il peut être plus facile de transmettre la clé en tant que paramètre de requête :</p> <pre data-bbox="620 1578 1044 1695">GET /indexes/hotels/docs? search=&amp;\$orderby=lastRenovationDate desc&amp;api-version=2020-06-30&amp;api-key= [query key]</pre>	50 par service

Visuellement, il n'existe aucune distinction entre une clé d'administration et une clé de requête. Les deux clés sont des chaînes composées de 32 caractères alphanumériques générés de façon aléatoire. Si vous n'êtes pas sûr du type de clé spécifié dans votre application, vous pouvez [vérifier les valeurs de clé dans le portail](#) ou utiliser l'[API REST](#) pour retourner la valeur et le type de clé.

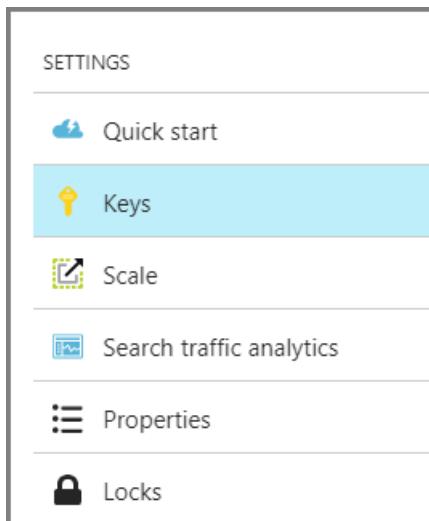
#### NOTE

L'insertion de données sensibles comme une `api-key` dans l'URI de requête est considérée comme une pratique peu sécurisée. C'est pourquoi Recherche cognitive Azure accepte uniquement une clé de requête sous forme de `api-key` dans la chaîne de requête, et il est conseillé de procéder autrement, sauf si le contenu de l'index doit être accessible au public. En règle générale, nous vous recommandons de transmettre votre `api-key` en tant qu'en-tête de demande.

## Rechercher des clés existantes

Vous pouvez obtenir les clés d'accès dans le portail ou via l'[API REST de gestion](#). Pour plus d'informations, consultez la page [Gérer les clés API d'administration et de requête](#).

1. Connectez-vous au [portail Azure](#).
2. Répertoriez les [services de recherche](#) pour votre abonnement.
3. Sélectionnez le service, puis sur la page de présentation, cliquez sur **Paramètres >Clés** pour afficher les clés d'administration et de requête.

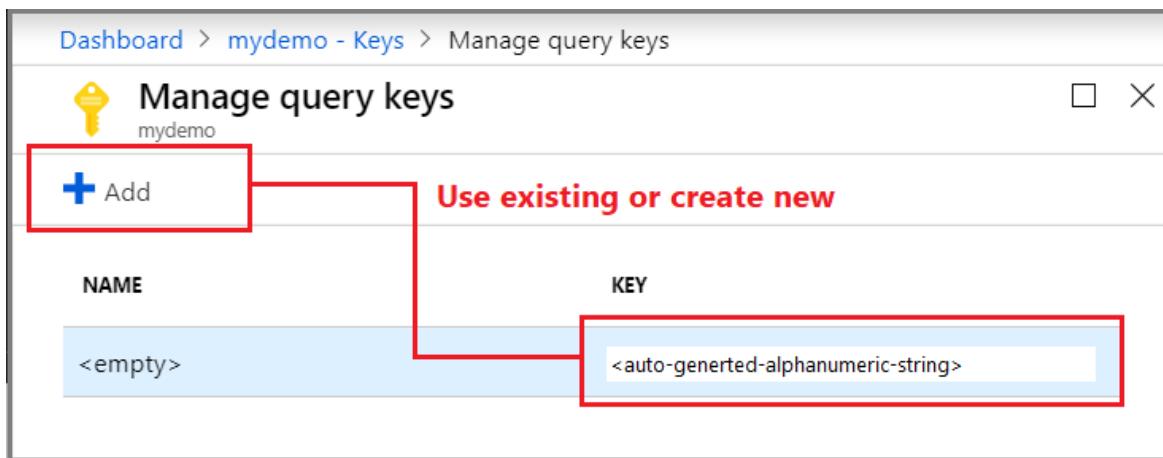


## Créer des clés de requête

Les clés de requête sont utilisées pour l'accès en lecture seule aux documents au sein d'un index pour les opérations ciblant une collection de documents. Les requêtes de recherche, de filtrage et de suggestion sont toutes des opérations qui utilisent une clé de requête. Toute opération en lecture seule qui renvoie des données système ou des définitions d'objet, comme une définition d'index ou un statut d'indexation, nécessite une clé d'administration.

Il est essentiel de restreindre l'accès et les opérations dans les applications clientes afin de protéger les ressources de recherche de votre service. Utilisez toujours une clé de requête plutôt qu'une clé d'administration pour toutes les requêtes provenant d'une application cliente.

1. Connectez-vous au [portail Azure](#).
2. Répertoriez les [services de recherche](#) pour votre abonnement.
3. Sélectionnez le service, puis sur la page de présentation, cliquez sur **Paramètres >Clés**.
4. Cliquez sur **Gérer les clés de requête**.
5. Utilisez la clé de requête déjà générée pour votre service, ou créez jusqu'à 50 nouvelles clés de requête. La clé de requête par défaut n'est pas nommée, mais des clés de requête supplémentaires peuvent être nommées pour faciliter la gestion.



#### NOTE

Vous trouverez un exemple de code illustrant l'utilisation de la clé de requête dans [Interroger un index Recherche cognitive Azure en C#](#).

## Régénération des clés d'administration

Deux clés d'administration sont créées pour chaque service. Vous pouvez ainsi remplacer la clé primaire par la clé secondaire pour assurer la continuité de vos activités.

1. Dans la page **Paramètres >Clés**, copiez la clé secondaire.
2. Pour toutes les applications, mettez à jour les paramètres de la clé API afin d'utiliser la clé secondaire.
3. Régénérez la clé principale.
4. Mettez à jour toutes les applications pour qu'elles utilisent la nouvelle clé principale.

Si, par inadvertance, vous régénérez les deux clés en même temps, toutes les requêtes de client utilisant ces clés échoueront (HTTP 403 Refusé). Toutefois, le contenu ne sera pas supprimé et vous ne subirez pas de verrouillage permanent.

Vous pouvez toujours accéder au service via le portail ou la couche de gestion ([REST API](#), [PowerShell](#) ou Azure Resource Manager). Les fonctions de gestion reposent sur un ID d'abonnement et non sur une clé API de service. Elles restent donc disponibles même si vos clés API ne le sont pas.

Après avoir créé de nouvelles clés via le portail ou la couche de gestion, l'accès à votre contenu (index, indexeurs, sources de données, cartes de synonymes) est restauré dès que vous disposez des nouvelles clés et que vous les fournissez sur les requêtes.

## Sécuriser les clés API

La sécurité des clés est assurée en limitant l'accès via le portail ou des interfaces Resource Manager (PowerShell ou interface de ligne de commande). Comme indiqué, les administrateurs des abonnements peuvent afficher et régénérer toutes les clés API. Par précaution, passez en revue les affectations de rôle pour comprendre qui a accès aux clés Admin.

- Dans le tableau de bord de service, cliquez sur **Contrôle d'accès (IAM)** , puis sur l'onglet **Attributions de rôles** pour afficher les affectations de rôle pour votre service.

Les membres des rôles suivants peuvent afficher et régénérer les clés : Propriétaire, Collaborateur, [Collaborateurs Search Service](#)

#### **NOTE**

Pour un accès en fonction de l'identité sur les résultats de recherche, vous pouvez créer des filtres de sécurité pour ajuster les résultats par identité, en supprimant les documents auxquels le demandeur ne doit pas avoir accès. Pour plus d'informations, consultez [Filtres de sécurité](#) et [Sécuriser avec Active Directory](#).

## Voir aussi

- [Contrôle d'accès en fonction du rôle dans Recherche cognitive Azure](#)
- [Gestion à l'aide de PowerShell](#)
- [Article sur les performances et l'optimisation](#)

# Configurer le pare-feu IP pour Recherche cognitive Azure

04/10/2020 • 5 minutes to read • [Edit Online](#)

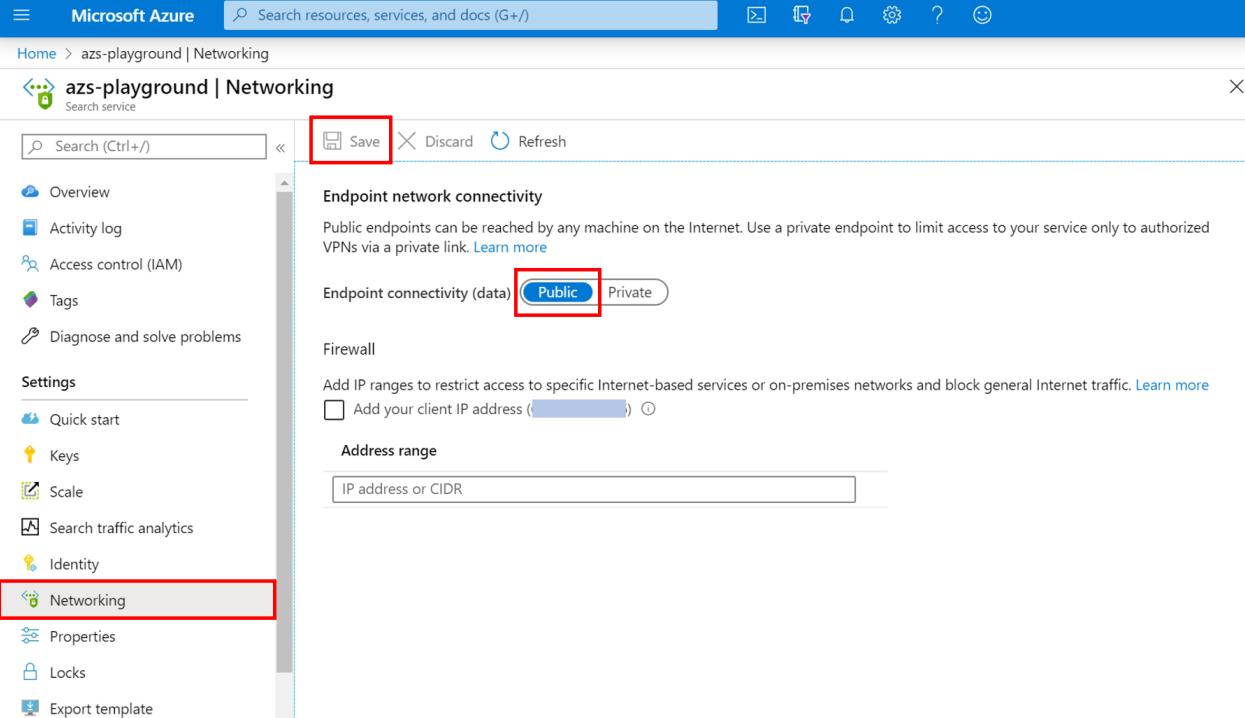
Recherche cognitive Azure prend en charge les règles IP pour la prise en charge du pare-feu entrant. Ce modèle fournit une couche supplémentaire de sécurité pour votre service de recherche, similaire aux règles IP que vous trouverez dans un groupe de sécurité de réseau virtuel Azure. Avec ces règles IP, vous pouvez configurer votre service de recherche pour qu'il soit accessible uniquement à partir d'un ensemble d'ordinateurs et/ou de services cloud approuvés. L'accès aux données stockées dans votre service de recherche à partir de ces ensembles d'ordinateurs et de services approuvés nécessite toujours que l'appelant présente un jeton d'autorisation valide.

## IMPORTANT

Les règles IP sur votre service Recherche cognitive Azure peuvent être configurées à l'aide du Portail Azure ou de l'[API REST de gestion version 2020-03-13](#).

## Configurer un pare-feu IP à l'aide du Portail Azure

Pour définir la stratégie de contrôle d'accès IP dans le Portail Azure, accédez à la page de votre service Recherche cognitive Azure et sélectionnez **Mise en réseau** dans le menu de navigation. La connectivité réseau des points de terminaison doit être **publique**. Si votre connectivité est définie sur **Privée**, vous pouvez accéder à votre service de recherche uniquement via un point de terminaison privé.



The screenshot shows the Microsoft Azure portal interface. At the top, there's a blue header bar with the Microsoft Azure logo, a search bar, and various navigation icons. Below the header, the URL 'Home > azs-playground | Networking' is visible. The main content area has a white background. On the left, a vertical sidebar lists several service management options: Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Settings (with sub-options like Quick start, Keys, Scale, Search traffic analytics, Identity), and Networking. The 'Networking' option is highlighted with a red box. The main content area starts with a 'Search (Ctrl+/' input field and a 'Save' button. Below that, the 'Endpoint network connectivity' section contains the following text: 'Public endpoints can be reached by any machine on the Internet. Use a private endpoint to limit access to your service only to authorized VPNs via a private link.' A 'Learn more' link is provided. Underneath this text are two buttons: 'Public' (which is highlighted with a red box) and 'Private'. Further down, there's a 'Firewall' section with a checkbox for 'Add IP ranges to restrict access to specific Internet-based services or on-premises networks and block general Internet traffic.' The checkbox is followed by a placeholder '( )' and a 'Learn more' link. Finally, there's an 'Address range' section with an input field labeled 'IP address or CIDR'.

Le Portail Azure permet de spécifier des adresses IP et des plages d'adresses IP au format CIDR. Un exemple de notation CIDR est 8.8.8.0/24, qui représente les adresses IP comprises entre 8.8.8.0 et 8.8.8.255.

## NOTE

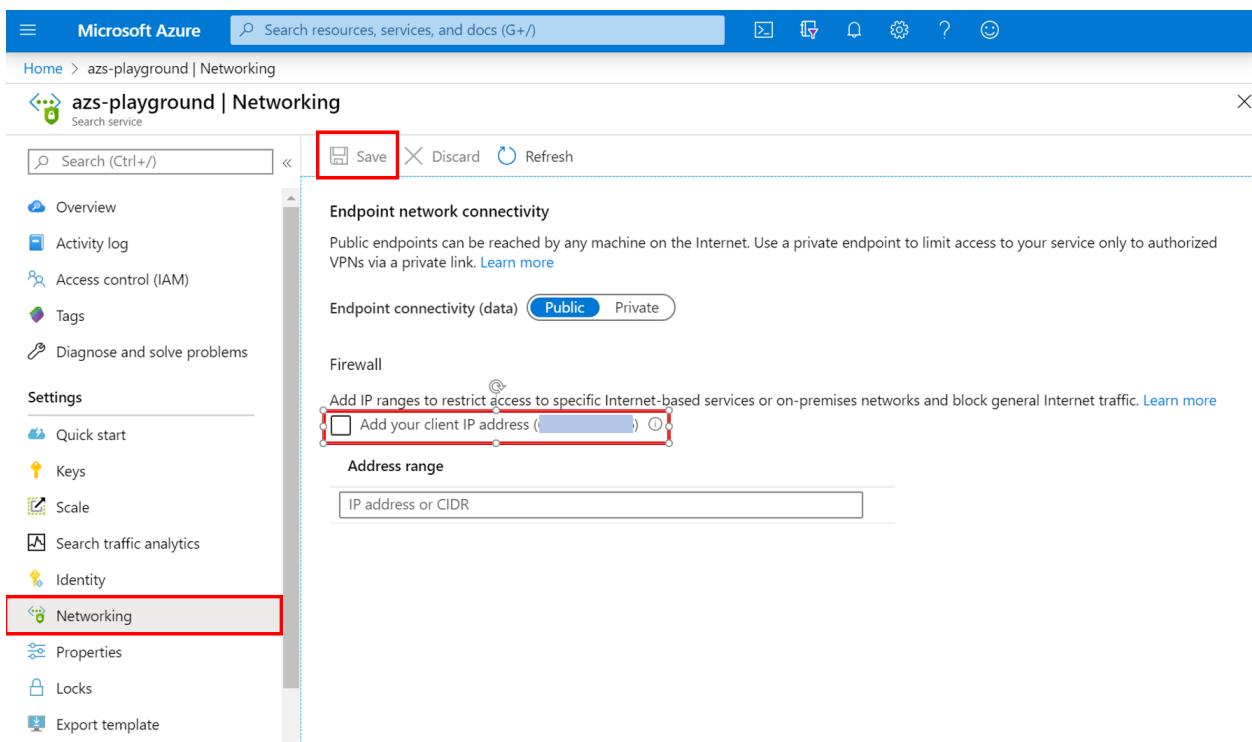
Une fois que vous avez activé la stratégie de contrôle d'accès IP pour votre service Recherche cognitive Azure, toutes les demandes adressées au plan de données à partir d'ordinateurs ne figurant pas dans la liste des plages d'adresses IP autorisées sont rejetées. Lorsque des règles IP sont configurées, certaines fonctionnalités du Portail Azure sont désactivées. Vous pouvez voir et gérer les informations au niveau du service, mais l'accès du portail aux données d'index et aux divers composants de ce service, comme les définitions d'index, d'indexeur et d'ensemble de compétences, est limité pour des raisons de sécurité.

## Demandes à partir de votre adresse IP actuelle

Pour simplifier le développement, le portail Azure vous aide à identifier et à ajouter l'adresse IP de votre ordinateur client à la liste autorisée. Les applications qui s'exécutent sur votre ordinateur peuvent ensuite accéder à votre service Recherche cognitive Azure.

Le portail détecte automatiquement l'adresse IP de votre client. Il peut s'agir de l'adresse IP du client de votre ordinateur ou de la passerelle réseau. N'oubliez pas de supprimer cette adresse IP avant de mettre vos charges de travail en production.

Pour ajouter votre adresse IP actuelle à la liste des adresses IP, cochez **Ajouter l'adresse IP de votre client**. Ensuite, sélectionnez **Enregistrer**.



The screenshot shows the Microsoft Azure portal interface. The top navigation bar includes the Microsoft Azure logo, a search bar, and various navigation icons. Below the navigation bar, the URL 'Home > azs-playground | Networking' is displayed. The main content area shows the 'azs-playground | Networking' page. On the left, a sidebar lists several service settings: Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Settings (with sub-options Quick start, Keys, Scale, Search traffic analytics, Identity), and Networking (which is currently selected and highlighted with a red box). The main content area displays 'Endpoint network connectivity' settings. It includes sections for 'Endpoint connectivity (data)' (set to 'Public'), 'Firewall', and a 'Address range' input field. A specific section for 'Firewall' contains a checkbox labeled 'Add your client IP address' with an associated input field. Both the checkbox and the input field are highlighted with a red box. Above this highlighted area, the 'Save' button is also highlighted with a red box. The overall interface is clean and modern, typical of the Microsoft Azure web platform.

## Résoudre les problèmes de stratégie de contrôle d'accès IP

Vous pouvez résoudre les problèmes de stratégie de contrôle d'accès IP en utilisant les options suivantes :

### Portail Azure

L'activation d'une stratégie de contrôle d'accès IP pour votre service Recherche cognitive Azure bloque toutes les demandes des ordinateurs en dehors de la liste des plages d'adresses IP autorisées, y compris le Portail Azure. Vous pouvez voir et gérer les informations au niveau du service, mais l'accès du portail aux données d'index et aux divers composants de ce service, comme les définitions d'index, d'indexeur et d'ensemble de compétences, est limité pour des raisons de sécurité.

### Kits SDK

Quand vous accédez au service Recherche cognitive Azure à l'aide du Kit de développement logiciel (SDK) à

partir d'ordinateurs ne figurant pas dans la liste autorisée, une réponse **403 interdit** générique est retournée sans aucun détail supplémentaire. Vérifiez la liste des adresses IP autorisées pour votre compte et assurez-vous que la configuration appropriée a été mise à jour pour votre service de recherche.

## Étapes suivantes

Pour plus d'informations sur l'accès à votre service de recherche via Azure Private Link, consultez l'article suivant :

- [Créer un point de terminaison privé pour une connexion sécurisée à Recherche cognitive Azure](#)

# Créer un point de terminaison privé pour une connexion sécurisée à Recherche cognitive Azure

04/10/2020 • 15 minutes to read • [Edit Online](#)

Dans cet article, vous allez utiliser le Portail Microsoft Azure pour créer une instance de service Recherche cognitive Azure qui n'est pas accessible via Internet. Ensuite, vous allez configurer une machine virtuelle Azure dans le même réseau virtuel et l'utiliser pour accéder au service Search via un point de terminaison privé.

Les points de terminaison privés sont fournis par [Azure Private Link](#), en tant que service distinct. Pour plus d'informations sur les coûts, consultez la page [Tarification](#).

## IMPORTANT

La prise en charge des points de terminaison privés pour Recherche cognitive Azure peut être configurée à l'aide du Portail Azure ou de l'[API REST de gestion version 2020-03-13](#). Lorsque le point de terminaison de service est privé, certaines fonctionnalités du portail sont désactivées. Vous pouvez voir et gérer les informations au niveau du service, mais l'accès du portail aux données d'index et aux divers composants de ce service, comme les définitions d'index, d'indexeur et d'ensemble de compétences, est limité pour des raisons de sécurité.

## Pourquoi utiliser un point de terminaison privé pour sécuriser l'accès ?

Les [points de terminaison privés](#) de Recherche cognitive Azure permettent à un client d'un réseau virtuel d'accéder en toute sécurité aux données d'un index de recherche grâce à une [liaison privée](#). Ils utilisent une adresse IP de l'[espace d'adressage du réseau virtuel](#) pour votre service Search. Le trafic entre le client et le service Search traverse le réseau virtuel et une liaison privée sur le réseau principal de Microsoft, ce qui élimine l'exposition sur l'Internet public. Pour obtenir la liste des autres services PaaS qui prennent en charge la liaison privée, consultez la [section disponibilité](#) dans la documentation du produit.

Les points de terminaison privés de votre service Search vous permettent de :

- Bloquer toutes les connexions sur le point de terminaison public de votre service Search.
- Renforcer la sécurité du réseau virtuel en assurant le blocage de l'exfiltration des données à partir du réseau virtuel.
- Vous connecter en toute sécurité à votre service Search à partir de réseaux locaux qui se connectent au réseau virtuel à l'aide de [VPN](#) ou [d'ExpressRoutes](#) avec le peering privé.

## Créer un réseau virtuel

Dans cette section, vous allez créer un réseau virtuel et un sous-réseau pour héberger la machine virtuelle qui est utilisée pour accéder au point de terminaison privé de votre service Search.

1. Dans l'onglet Accueil du portail Azure, sélectionnez **Créer une ressource > Mise en réseau > Réseau virtuel**.
2. Dans **Créer un réseau virtuel**, entrez ou sélectionnez ces informations :

PARAMÈTRE	VALEUR
Abonnement	Sélectionnez votre abonnement

PARAMÈTRE	VALEUR
Resource group	Sélectionnez <b>Créer nouveau</b> , entrez <i>myResourceGroup</i> et cliquez sur <b>OK</b>
Nom	Entrez <i>MyVirtualNetwork</i>
Région	Sélectionnez la région de votre choix

3. Conservez les valeurs par défaut pour les autres paramètres. Cliquez sur **Vérifier + créer**, puis sur **Créer**.

## Créer un service Search avec un point de terminaison privé

Dans cette section, vous allez créer un service Recherche cognitive Azure avec un point de terminaison privé.

1. En haut à gauche de l'écran du portail Azure, sélectionnez **Créer une ressource > Web > Recherche cognitive Azure**.
2. Dans **Nouveau service Search – Concepts de base**, entrez ou sélectionnez les informations suivantes :

PARAMÈTRE	VALEUR
<b>DÉTAILS DU PROJET</b>	
Abonnement	Sélectionnez votre abonnement.
Resource group	Sélectionnez <b>myResourceGroup</b> . Vous avez créé cela dans la section précédente.
<b>DÉTAILS DE L'INSTANCE</b>	
URL	Entrez un nom unique.
Emplacement	Sélectionnez la région de votre choix.
Niveau tarifaire	Sélectionnez <b>Changer le niveau tarifaire</b> et choisissez le niveau de service souhaité. (Pas de prise en charge pour le niveau <b>Gratuit</b> . Il doit s'agir du niveau <b>De base</b> ou supérieur.)

3. Sélectionnez **Suivant : Mise à l'échelle**.
4. Conservez les valeurs par défaut, puis sélectionnez **Suivant : Mise en réseau**.
5. Dans **Nouveau service Search – Mise en réseau**, sélectionnez **Privé** pour la **Connectivité du point de terminaison (données)**.
6. Dans **Nouveau service Search – Mise en réseau**, sélectionnez **+ Ajouter sous Point de terminaison privé**.
7. Dans **Créer un point de terminaison privé**, entrez ou sélectionnez les informations suivantes :

PARAMÈTRE	VALEUR
Abonnement	Sélectionnez votre abonnement.
Resource group	Sélectionnez <b>myResourceGroup</b> . Vous avez créé cela dans la section précédente.
Emplacement	Sélectionnez <b>USA Ouest</b> .
Nom	Entrez <i>myPrivateEndpoint</i> .
Sous-ressource cible	Conservez le <b>searchService</b> par défaut.
<b>MISE EN RÉSEAU</b>	
Réseau virtuel	Sélectionnez <i>MyVirtualNetwork</i> dans le groupe de ressources <i>myResourceGroup</i> .
Subnet	Sélectionnez <i>mySubnet</i> .
<b>INTÉGRATION À DNS PRIVÉ</b>	
Intégrer à une zone DNS privée	Conservez la valeur par défaut <b>Oui</b> .
Zone DNS privée	Conservez la valeur par défaut ** privatelink.search.windows.net**.

8. Sélectionnez OK.
9. Sélectionnez **Revoir + créer**. Vous êtes redirigé vers la page **Vérifier + créer** où Azure valide votre configuration.
10. Lorsque le message **Validation passed** (Validation réussie) apparaît, sélectionnez **Créer**.
11. Une fois la configuration de votre nouveau service terminée, accédez à la ressource que vous venez de créer.
12. Sélectionnez **Clés** dans le menu de contenu de gauche.
13. Copiez la clé **d'administration principale** pour plus tard, lors de la connexion au service.

## Création d'une machine virtuelle

1. En haut à gauche de l'écran du portail Azure, sélectionnez **Créer une ressource > Calcul > Machine virtuelle**.
2. Dans **Créer une machine virtuelle - Notions de base**, entrez ou sélectionnez ces informations :

PARAMÈTRE	VALEUR
<b>DÉTAILS DU PROJET</b>	
Abonnement	Sélectionnez votre abonnement.

PARAMÈTRE	VALEUR
Resource group	Sélectionnez <b>myResourceGroup</b> . Vous avez créé cela dans la section précédente.
<b>DÉTAILS DE L'INSTANCE</b>	
Nom de la machine virtuelle	Entrez <i>myVm</i> .
Région	Sélectionnez <b>USA Ouest</b> ou la région que vous utilisez.
Options de disponibilité	Conservez la valeur par défaut <b>Aucune redondance d'infrastructure nécessaire</b> .
Image	Sélectionnez <b>Windows Server 2019 Datacenter</b> .
Taille	Conservez la valeur par défaut <b>Standard DS1 v2</b> .
<b>COMPTE ADMINISTRATEUR</b>	
Nom d'utilisateur	Entrez un nom d'utilisateur de votre choix.
Mot de passe	Entrez un mot de passe de votre choix. Le mot de passe doit contenir au moins 12 caractères et satisfaire aux <a href="#">exigences de complexité définies</a> .
Confirmer le mot de passe	Retapez le mot de passe.
<b>RÈGLES DES PORTS D'ENTRÉE</b>	
Aucun port d'entrée public	Conservez la valeur par défaut <b>Autoriser les ports sélectionnés</b> .
Sélectionner des ports d'entrée	Conservez la valeur par défaut <b>RDP (3389)</b> .
<b>ÉCONOMISEZ DE L'ARGENT</b>	
Vous disposez déjà d'une licence Windows ?	Conservez la valeur par défaut <b>Non</b> .

3. Sélectionnez **Suivant : Disques**.

4. Dans **Créer une machine virtuelle - Disks**, conservez les valeurs par défaut et sélectionnez **Suivant : Mise en réseau**.

5. Dans **Créer une machine virtuelle - Mise en réseau**, sélectionnez ces informations :

PARAMÈTRE	VALEUR
Réseau virtuel	Conservez la valeur par défaut, <b>MyVirtualNetwork</b> .
Espace d'adressage	Conservez la valeur par défaut, <b>10.1.0.0/24</b> .

PARAMÈTRE	VALEUR
Subnet	Conservez la valeur par défaut, <b>mySubnet</b> (10.1.0.0/24).
Adresse IP publique	Conservez la valeur par défaut ( <b>new</b> ) <b>myVm-ip</b> .
Aucun port d'entrée public	Sélectionnez <b>Autoriser les ports sélectionnés</b> .
Selectionner des ports d'entrée	Sélectionnez <b>HTTP et RDP</b> .

6. Sélectionnez **Revoir + créer**. Vous êtes redirigé vers la page **Vérifier + créer** où Azure valide votre configuration.
7. Lorsque le message **Validation passed** (Validation réussie) apparaît, sélectionnez **Créer**.

## Connexion à la machine virtuelle

Procédez au téléchargement, puis connectez-vous à la machine virtuelle *myVm* comme suit :

1. Dans la barre de recherche du portail, entrez *myVm*.
2. Sélectionnez le bouton **Connexion**. Après avoir sélectionné le bouton **Connecter**, **Se connecter à la machine virtuelle** s'ouvre.
3. Sélectionnez **Télécharger le fichier RDP**. Azure crée un fichier de protocole RDP (Remote Desktop Protocol) (.rdp) et le télécharge sur votre ordinateur.
4. Ouvrez le fichier .rdp\* téléchargé.
  - a. Si vous y êtes invité, sélectionnez **Connexion**.
  - b. Entrez le nom d'utilisateur et le mot de passe spécifiés lors de la création de la machine virtuelle.

### NOTE

Vous devrez peut-être sélectionner **Plus de choix > Utiliser un autre compte**, pour spécifier les informations d'identification que vous avez entrées lorsque vous avez créé la machine virtuelle.

5. Sélectionnez **OK**.
6. Un avertissement de certificat peut s'afficher pendant le processus de connexion. Si vous recevez un avertissement de certificat, sélectionnez **Oui** ou **Continuer**.
7. Une fois que le bureau de la machine virtuelle s'affiche, réduisez-le pour revenir à votre poste de travail local.

## Tester les connexions

Dans cette section, vous vérifiez l'accès au réseau privé du service Search et vous connecter en privé à l'aide du point de terminaison privé.

Lorsque le point de terminaison de service de recherche est privé, certaines fonctionnalités du portail sont désactivées. Vous pouvez voir et gérer les paramètres du niveau de service, mais l'accès du portail aux données d'index et aux divers autres composants de ce service, comme les définitions d'index, d'indexeur et d'ensemble de compétences, est limité pour des raisons de sécurité.

1. Dans le Bureau à distance de *myVM*, ouvrez PowerShell.
2. Entrez « nslookup [nom du service Search].search.windows.net »

Vous recevez un message similaire à celui ci :

```
Server: UnKnown
Address: 168.63.129.16
Non-authoritative answer:
Name: [search service name].privatelink.search.windows.net
Address: 10.0.0.5
Aliases: [search service name].search.windows.net
```

3. À partir de la machine virtuelle, connectez-vous au service Search et créez un index. Vous pouvez suivre ce [démarrage rapide](#) pour créer un index de recherche dans votre service dans Postman à l'aide de l'API REST. La configuration des demandes à partir de Postman nécessite le point de terminaison du service Search ([https://\[nom du service Search\].search.windows.net](https://[nom du service Search].search.windows.net)) et la clé API de l'administrateur que vous avez copiée à l'étape précédente.
4. En effectuant le démarrage rapide à partir de la machine virtuelle, vous confirmez que le service est pleinement opérationnel.
5. Fermez la connexion Bureau à distance à *myVM*.
6. Pour vérifier que votre service n'est pas accessible sur un point de terminaison public, ouvrez Postman sur votre station de travail locale et essayez d'effectuer les premières tâches du démarrage rapide. Si vous recevez un message d'erreur indiquant que le serveur distant n'existe pas, vous avez correctement configuré un point de terminaison privé pour votre service Search.

## Nettoyer les ressources

Lorsque vous avez fini d'utiliser le point de terminaison privé, le service Search et la machine virtuelle, supprimez le groupe de ressources et toutes les ressources qu'il contient :

1. Entrez *myResourceGroup* dans la zone **Rechercher** en haut du portail, puis sélectionnez *myResourceGroup* dans les résultats de la recherche.
2. Sélectionnez **Supprimer le groupe de ressources**.
3. Entrez *myResourceGroup* pour TAPEZ LE NOM DU GROUPE DE RESSOURCES, puis sélectionnez **Supprimer**.

## Étapes suivantes

Grâce à cet article, vous avez créé une machine virtuelle sur un réseau virtuel et un service Search avec un point de terminaison privé. Vous vous êtes connecté à la machine virtuelle à partir d'Internet et avez communiqué en toute sécurité avec le service Search à l'aide de la liaison privée. Pour en savoir plus sur le point de terminaison privé, consultez [Qu'est-ce qu'Azure Private Endpoint ?](#)

# Filtres de sécurité pour le filtrage des résultats dans Recherche cognitive Azure

04/10/2020 • 7 minutes to read • [Edit Online](#)

Vous pouvez appliquer des filtres de sécurité pour filtrer les résultats de recherche dans Recherche cognitive Azure en fonction de l'identité de l'utilisateur. Cette expérience de recherche compare généralement l'identité de la personne qui lance la recherche à un champ contenant les principaux qui disposent d'autorisations d'accès au document. Quand une correspondance est trouvée, l'utilisateur ou le principal (comme un groupe ou un rôle) a accès à ce document.

Pour mettre en place le filtrage de sécurité, une méthode consiste à utiliser une disjonction complexe d'expressions d'égalité. Par exemple : `Id eq 'id1' or Id eq 'id2'`, etc. Cette approche est sujette aux erreurs et difficile à gérer. De plus, si la liste contient des centaines voire des milliers de valeurs, elle ralentit le temps de réponse de plusieurs secondes.

Une approche plus simple et plus rapide consiste à utiliser la fonction `search.in`. En utilisant `search.in(Id, 'id1, id2, ...')` à la place d'une expression d'égalité, vous pouvez obtenir des temps de réponse inférieurs à une seconde.

Cet article explique les étapes à suivre pour mettre en place le filtrage de sécurité :

- Créer un champ qui contient les identificateurs de principal
- Envoyer (push) ou mettre à jour les documents existants avec les identificateurs de principal appropriés
- Émettre une demande de recherche avec `search.in filter`

## NOTE

Le processus de récupération des identificateurs de principal n'est pas abordé dans ce document. Obtenez-le auprès de votre fournisseur de services d'identité.

## Prérequis

Cet article part du principe que vous disposez d'un [abonnement Azure](#), du service Recherche cognitive Azure et d'un [index](#).

## Créer le champ de sécurité

Vos documents doivent inclure un champ qui spécifie les groupes disposant d'autorisations d'accès. Ces informations constituent les critères de filtre par rapport auxquels les documents sont sélectionnés ou non dans le jeu de résultats retourné à l'émetteur. Imaginons que nous disposons d'un index de fichiers sécurisés et qu'un ensemble différent d'utilisateurs a accès à chaque fichier.

1. Ajoutez le champ `group_ids` (vous pouvez choisir n'importe quel nom ici) comme `Collection(Edm.String)`. Vérifiez que le champ a un attribut `filterable` défini avec la valeur `true` pour que les résultats de la recherche soient filtrés en fonction de l'accès dont dispose l'utilisateur. Par exemple, si vous définissez le champ `group_ids` avec la valeur `["group_id1, group_id2"]` pour le document ayant comme `file_name` « secured\_file\_b », seuls les utilisateurs qui appartiennent à l'ID de groupe « group\_id1 » ou « group\_id2 » ont accès en lecture au fichier.

Vérifiez que l'attribut `retrievable` du champ a la valeur `false` pour qu'il ne soit pas retourné dans le cadre de la requête de recherche.

2. Pour les besoins de cet exemple, ajoutez également les champs `file_id` et `file_name`.

```
{  
    "name": "securedfiles",  
    "fields": [  
        {"name": "file_id", "type": "Edm.String", "key": true, "searchable": false, "sortable": false, "facetable": false},  
        {"name": "file_name", "type": "Edm.String"},  
        {"name": "group_ids", "type": "Collection(Edm.String)", "filterable": true, "retrievable": false}  
    ]  
}
```

## Envoi (push) des données à votre index à l'aide de l'API REST

Émettez une requête HTTP POST au point de terminaison de l'URL de votre index. Le corps de la requête HTTP est un objet JSON contenant les documents à ajouter :

```
POST https://[search service].search.windows.net/indexes/securedfiles/docs/index?api-version=2020-06-30  
Content-Type: application/json  
api-key: [admin key]
```

Dans le corps de la requête, spécifiez le contenu de vos documents :

```
{  
    "value": [  
        {  
            "@search.action": "upload",  
            "file_id": "1",  
            "file_name": "secured_file_a",  
            "group_ids": ["group_id1"]  
        },  
        {  
            "@search.action": "upload",  
            "file_id": "2",  
            "file_name": "secured_file_b",  
            "group_ids": ["group_id1", "group_id2"]  
        },  
        {  
            "@search.action": "upload",  
            "file_id": "3",  
            "file_name": "secured_file_c",  
            "group_ids": ["group_id5", "group_id6"]  
        }  
    ]  
}
```

Si vous avez besoin de mettre à jour un document existant avec la liste des groupes, vous pouvez utiliser l'action `merge` ou `mergeOrUpload` :

```
{
  "value": [
    {
      "@search.action": "mergeOrUpload",
      "file_id": "3",
      "group_ids": ["group_id7", "group_id8", "group_id9"]
    }
  ]
}
```

Pour obtenir des informations détaillées sur l'ajout ou la mise à jour de documents, lisez [Modifier des documents](#).

## Appliquer le filtre de sécurité

Pour filtrer des documents en fonction de l'accès de `group_ids`, vous devez émettre une requête de recherche avec un filtre `group_ids/any(g:search.in(g, 'group_id1, group_id2,...'))`, où « `group_id1, group_id2,...` » sont les groupes auxquels l'émetteur de la requête de recherche appartient. Ce filtre correspond à tous les documents dont le champ `group_ids` contient l'un des identificateurs donnés. Pour obtenir des informations détaillées sur la recherche de documents à l'aide de Recherche cognitive Azure, lisez [Recherche de documents](#). Notez que cet exemple montre comment lancer une recherche dans des documents à l'aide d'une requête POST.

Émettez la requête HTTP POST :

```
POST https://[service name].search.windows.net/indexes/securedfiles/docs/search?api-version=2020-06-30
Content-Type: application/json
api-key: [admin or query key]
```

Spécifiez le filtre dans le corps de la requête :

```
{
  "filter": "group_ids/any(g:search.in(g, 'group_id1, group_id2'))"
}
```

Vous devez obtenir les documents où `group_ids` contient « `group_id1` » ou « `group_id2` ». En d'autres termes, vous obtenez les documents auxquels l'émetteur de la requête a accès en lecture.

```
{
  [
    {
      "@search.score": 1.0,
      "file_id": "1",
      "file_name": "secured_file_a",
    },
    {
      "@search.score": 1.0,
      "file_id": "2",
      "file_name": "secured_file_b"
    }
  ]
}
```

## Conclusion

Vous venez de voir comment filtrer des résultats en fonction de l'identité de l'utilisateur et de la fonction `search.in()` de Recherche cognitive Azure. Vous pouvez utiliser cette fonction pour passer les identificateurs de

principal de l'utilisateur demandeur et les mettre en correspondance avec les identificateurs de principal associés à chaque document cible. Quand une requête de recherche est traitée, la fonction `search.in` exclut les résultats de la recherche inaccessibles en lecture aux principaux de l'utilisateur. Les identificateurs de principal peuvent représenter des groupes de sécurité, des rôles ou même la propre identité de l'utilisateur.

## Voir aussi

- [Contrôle d'accès à Active Directory basé sur l'identité à l'aide des filtres Recherche cognitive Azure](#)
- [Filtres dans la Recherche cognitive Azure](#)
- [Sécurité des données et contrôle d'accès dans les opérations de Recherche cognitive Azure](#)

# Utilisation de filtres de sécurité pour filtrer les résultats de Recherche cognitive Azure à l'aide d'identités Active Directory

04/10/2020 • 11 minutes to read • [Edit Online](#)

Cet article explique comment utiliser les identités de sécurité Azure Active Directory (AAD) avec des filtres dans la Recherche cognitive Azure pour tronquer les résultats de recherche en fonction de l'appartenance à des groupes d'utilisateurs.

Cet article décrit les tâches suivantes :

- Créer des utilisateurs et des groupes AAD
- Associer l'utilisateur au groupe que vous avez créé
- Mettre en cache les nouveaux groupes
- Indexer les documents avec les groupes associés
- Émettre une demande de recherche avec un filtre d'identificateurs de groupe

## NOTE

Dans cet article, les exemples d'extraits de code sont écrits en C#. L'intégralité du code source est disponible [sur GitHub](#).

## Prérequis

Votre index dans Recherche cognitive Azure doit avoir un [champ de sécurité](#) pour stocker la liste des identités de groupe disposant d'un accès en lecture pour le document. Ce cas d'usage implique une correspondance exacte entre un élément sécurisable (par exemple l'application d'un établissement scolaire) et un champ de sécurité spécifiant qui a accès à cet élément (personnel en charge des admissions).

Vous devez disposer des autorisations d'administrateur AAD requises dans cette procédure pour créer des utilisateurs, des groupes et des associations dans AAD.

Votre application doit également être inscrite auprès d'AAD, comme décrit dans la procédure suivante.

### Inscrire votre application auprès d'AAD

Cette étape intègre votre application avec AAD pour pouvoir accepter les connexions des comptes d'utilisateur et de groupe. Si vous n'êtes pas administrateur AAD dans votre organisation, vous devrez peut-être [créer un nouveau locataire](#) pour effectuer les étapes suivantes.

1. Accédez à [Portail d'inscription des applications](#) > Application convergée > **Ajouter une application**.
2. Entrez un nom pour votre application, puis cliquez sur **Créer**.
3. Sélectionnez votre application nouvellement inscrite sur la page Mes Applications.
4. Sur la page d'inscription des applications > **Plateformes** > **Ajouter une plateforme** > API Web.
5. Toujours sur la page d'inscription des applications, accédez à > **Autorisations pour Microsoft Graph** > **Ajouter**.
6. Sous Sélectionner les autorisations, ajoutez les autorisations déléguées suivantes, puis cliquez sur **OK** :

- Directory.ReadWrite.All
- Group.ReadWrite.All
- User.ReadWrite.All

Microsoft Graph fournit une API qui permet l'accès par programme à AAD via une API REST. L'exemple de code pour cette procédure utilise les autorisations pour appeler l'API Microsoft Graph pour la création de groupes, d'utilisateurs et d'associations. Les API sont également utilisées pour mettre en cache les identificateurs de groupe afin d'accélérer le filtrage.

## Créer des utilisateurs et des groupes

Si vous ajoutez une fonctionnalité de recherche à une application établie, il est possible que vous disposiez déjà d'identificateurs d'utilisateur et de groupe dans AAD. Dans ce cas, vous pouvez ignorer les trois étapes suivantes.

Toutefois, si vous n'avez pas d'utilisateurs existants, vous pouvez utiliser les API Microsoft Graph pour créer des entités de sécurité. Les extraits de code suivants montrent comment générer des identificateurs qui deviennent des valeurs de données pour le champ de sécurité dans votre index Recherche cognitive Azure. Dans l'exemple de notre application d'admission scolaire, il s'agirait des identificateurs de sécurité pour le personnel en charge des admissions.

La gestion des groupes et des utilisateurs peut s'avérer très fluide, en particulier dans les grandes organisations. Le code qui génère les identités d'utilisateur et de groupe doit s'exécuter assez souvent pour tenir compte des modifications apportées aux groupes de l'organisation. De même, votre index Recherche cognitive Azure requiert une planification de mise à jour similaire pour refléter l'état actuel des utilisateurs et des ressources autorisés.

### Étape 1 : Créer un groupe AAD

```
// Instantiate graph client
GraphServiceClient graph = new GraphServiceClient(new DelegateAuthenticationProvider(...));
Group group = new Group()
{
    DisplayName = "My First Prog Group",
    SecurityEnabled = true,
    MailEnabled = false,
    MailNickname = "group1"
};
Group newGroup = await graph.Groups.Request().AddAsync(group);
```

### Étape 2 : Créer un utilisateur AAD

```
User user = new User()
{
    GivenName = "First User",
    Surname = "User1",
    MailNickname = "User1",
    DisplayName = "First User",
    UserPrincipalName = "User1@FirstUser.com",
    PasswordProfile = new PasswordProfile() { Password = "*****" },
    AccountEnabled = true
};
User newUser = await graph.Users.Request().AddAsync(user);
```

### Étape 3 : Associer les utilisateurs et les groupes

```
await graph.Groups[newGroup.Id].Members.References.Request().AddAsync(newUser);
```

### Étape 4 : Mettre en cache les identificateurs de groupe

Si vous le souhaitez, pour réduire la latence du réseau, vous pouvez mettre en cache les associations utilisateurs-groupes. Ainsi, lorsqu'une demande de recherche est émise, les groupes sont renvoyés à partir du cache, ce qui évite un aller-retour dans AAD. Vous pouvez utiliser l'[API de Batch AAD](#) pour envoyer une requête Http unique avec plusieurs utilisateurs et générer le cache.

Microsoft Graph est conçu pour gérer un volume élevé de demandes. Si un trop grand nombre de demandes sont émises, Microsoft Graph génère une erreur avec le code d'état HTTP 429. Pour plus d'informations, consultez le document [Limitation dans Microsoft Graph](#).

## Indexer les documents avec leurs groupes autorisés

Les opérations de demande dans Recherche cognitive Azure sont exécutées via un index Recherche cognitive Azure. Dans cette étape, une opération d'indexation importe les données interrogables sous forme d'index, y compris les identificateurs utilisés comme filtres de sécurité.

Le service Recherche cognitive Azure n'authentifie pas les identités des utilisateurs et ne fournit aucune logique pour définir le contenu qu'un utilisateur est autorisé à consulter. Le cas d'usage pour le filtrage de sécurité implique que vous fournissiez l'association entre un document sensible et l'identificateur de groupe ayant accès à ce document importé sans modification dans un index de recherche.

Dans notre exemple hypothétique, le corps de la demande PUT sur un index Recherche cognitive Azure inclurait une épreuve ou une transcription d'un candidat, ainsi que l'identificateur de groupe ayant l'autorisation d'afficher ce contenu.

Dans l'exemple générique utilisé dans l'exemple de code pour cette procédure, l'action d'indexation peut se présenter comme suit :

```
var actions = new IndexAction<SecuredFiles>[]
{
    IndexAction.Upload(
        new SecuredFiles()
    {
        FileId = "1",
        Name = "secured_file_a",
        GroupIds = new[] { groups[0] }
    }),
    ...
};

var batch = IndexBatch.New(actions);

_indexClient.Documents.Index(batch);
```

## Émettre une demande de recherche

Pour des raisons de filtrage de sécurité, les valeurs du champ de sécurité dans l'index sont des valeurs statiques utilisées pour inclure ou exclure des documents dans les résultats de la recherche. Par exemple, si l'identificateur de groupe pour les admissions est « A11B22C33D44-E55F66G77-H88I99JKK », tous les documents dans un index Recherche cognitive Azure ayant cet identificateur dans le champ de sécurité sont inclus (ou exclus) dans les résultats de la recherche renvoyés au demandeur.

Pour filtrer les documents renvoyés dans les résultats de la recherche en fonction des groupes de l'utilisateur qui émet la demande, procédez comme suit.

### Étape 1 : Récupérer les identificateurs de groupe de l'utilisateur

Si les groupes de l'utilisateur n'ont pas encore été mis en cache, ou si le cache a expiré, exécutez la demande [groupes](#).

```

private static void RefreshCacheIfRequired(string user)
{
    if (!groupsCache.ContainsKey(user))
    {
        var groups = GetGroupIdsForUser(user).Result;
        _groupsCache[user] = groups;
    }
}

private static async Task<List<string>> GetGroupIdsForUser(string userPrincipalName)
{
    List<string> groups = new List<string>();
    var allUserGroupsRequest = graph.Users[userPrincipalName].GetMemberGroups(true).Request();

    while (allUserGroupsRequest != null)
    {
        IDirectoryObjectGetMemberGroupsRequestBuilder allUserGroups = await allUserGroupsRequest.PostAsync();
        groups = allUserGroups.ToList();
        allUserGroupsRequest = allUserGroups.NextPageRequest;
    }
    return groups;
}

```

## Étape 2 : Composer la requête de recherche

En supposant que vous connaissez les groupes de l'utilisateur, vous pouvez émettre la demande de recherche avec les valeurs de filtre appropriées.

```

string filter = String.Format("groupIds/any(p:search.in(p, '{0}'))", string.Join(", ", groups.Select(g => g.ToString())));
SearchParameters parameters = new SearchParameters()
{
    Filter = filter,
    Select = new[] { "application essays" }
};

DocumentSearchResult<SecuredFiles> results = _indexClient.Documents.Search<SecuredFiles>("*", parameters);

```

## Étape 3 : Gérer les résultats

La réponse inclut une liste répertoriant uniquement les documents que l'utilisateur est autorisé à afficher. Selon la façon dont vous construisez la page de résultats, vous pouvez inclure des repères visuels pour matérialiser le jeu de résultats obtenu.

## Conclusion

Dans cette procédure pas à pas, vous avez découvert comment utiliser les connexions AAD pour filtrer des documents dans les résultats Recherche cognitive Azure et ignorer les documents ne correspondant pas au filtre défini dans la demande.

## Voir aussi

- [Contrôle d'accès basé sur l'identité à l'aide des filtres Recherche cognitive Azure](#)
- [Filtres dans la Recherche cognitive Azure](#)
- [Sécurité des données et contrôle d'accès dans les opérations de Recherche cognitive Azure](#)

# Configuration de règles de pare-feu IP pour permettre l'accès de l'indexeur

04/10/2020 • 4 minutes to read • [Edit Online](#)

Les règles de pare-feu IP sur des ressources Azure, telles que des comptes de stockage, des comptes Cosmos DB et des serveurs Azure SQL, autorisent uniquement le trafic provenant de plages d'adresses IP spécifiques pour l'accès aux données.

Cet article décrit comment configurer les règles IP via le portail Azure, pour un compte de stockage, afin que les indexeurs de Recherche cognitive Azure puissent accéder aux données en toute sécurité. Bien qu'il soit spécifique du stockage, ce guide peut être traduit directement en d'autres ressources Azure qui proposent également des règles de pare-feu IP pour sécuriser l'accès aux données.

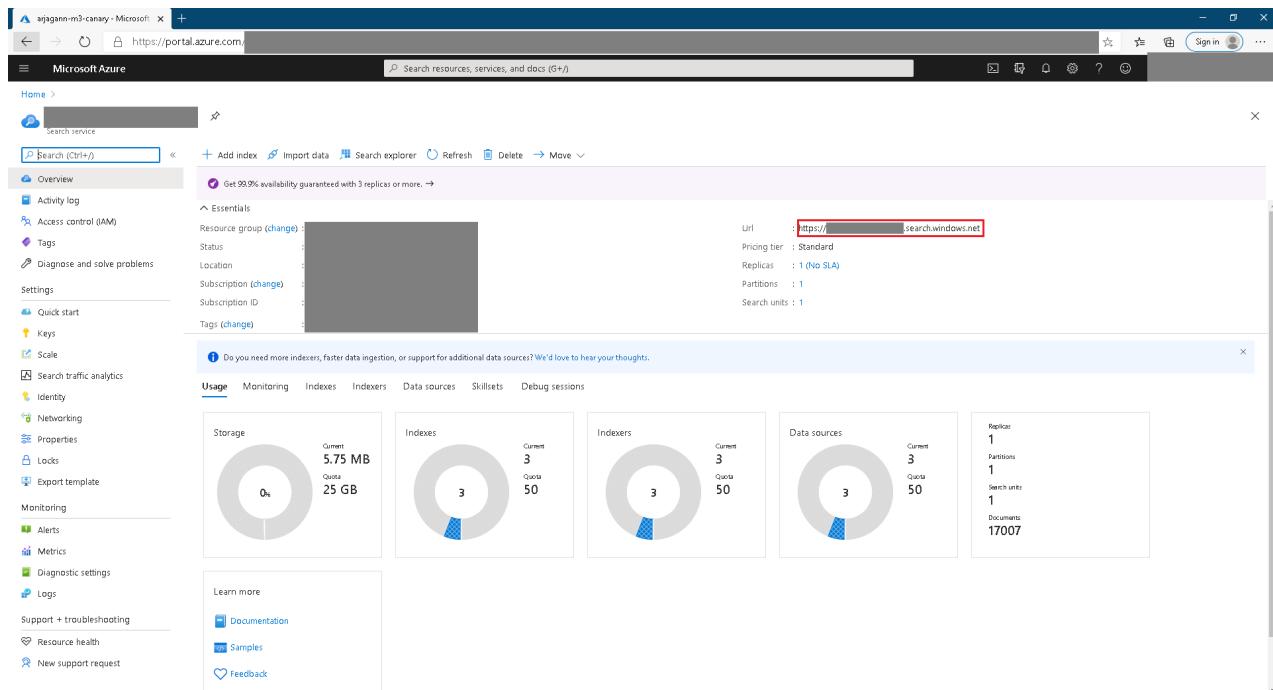
## NOTE

Les règles de pare-feu IP pour un compte de stockage ne sont effectives que si le compte de stockage et le service de recherche se trouvent dans des régions différentes. Si votre installation ne le permet pas, nous vous recommandons d'utiliser l'[option d'exception de service approuvé](#).

## Obtenir l'adresse IP du service de recherche

Obtenez le nom de domaine complet (FQDN) de votre service de recherche. Il ressemble à ceci :

`<search-service-name>.search.windows.net`. Vous pouvez déterminer le FQDN en recherchant votre service de recherche sur le portail Azure.



The screenshot shows the Microsoft Azure portal interface with the URL `https://portal.azure.com/` in the address bar. The main content area displays the configuration for a "Search service". On the left, there's a sidebar with various navigation options like "Overview", "Activity log", "Access control (IAM)", "Tags", "Diagnose and solve problems", "Settings", "Identity", "Networking", "Properties", "Logs", "Monitoring", "Metrics", "Diagnostic settings", "Logs", "Support + troubleshooting", "Resource health", and "New support request". The "Overview" tab is selected. In the center, there's a summary card with the URL `https://<service-name>.search.windows.net`, Status (Green), Pricing tier (Standard), Replicas (1), Partitions (1), and Search units (1). Below this, there are four circular dashboards: "Storage" (Current 5.75 MB, Quota 25 GB), "Indexes" (Current 3, Quota 50), "Indexers" (Current 3, Quota 50), and "Data sources" (Current 3, Quota 50). To the right of these dashboards, there's a table with the following data:

Replicas	Partitions	Search units	Documents
1	1	1	17007

At the bottom of the central area, there are links for "Learn more", "Documentation", "Samples", and "Feedback".

Vous pouvez obtenir l'adresse IP du service de recherche en exécutant une commande `nslookup` (ou `ping`) du FQDN. Il s'agit de l'une des adresses IP à ajouter aux règles de pare-feu.

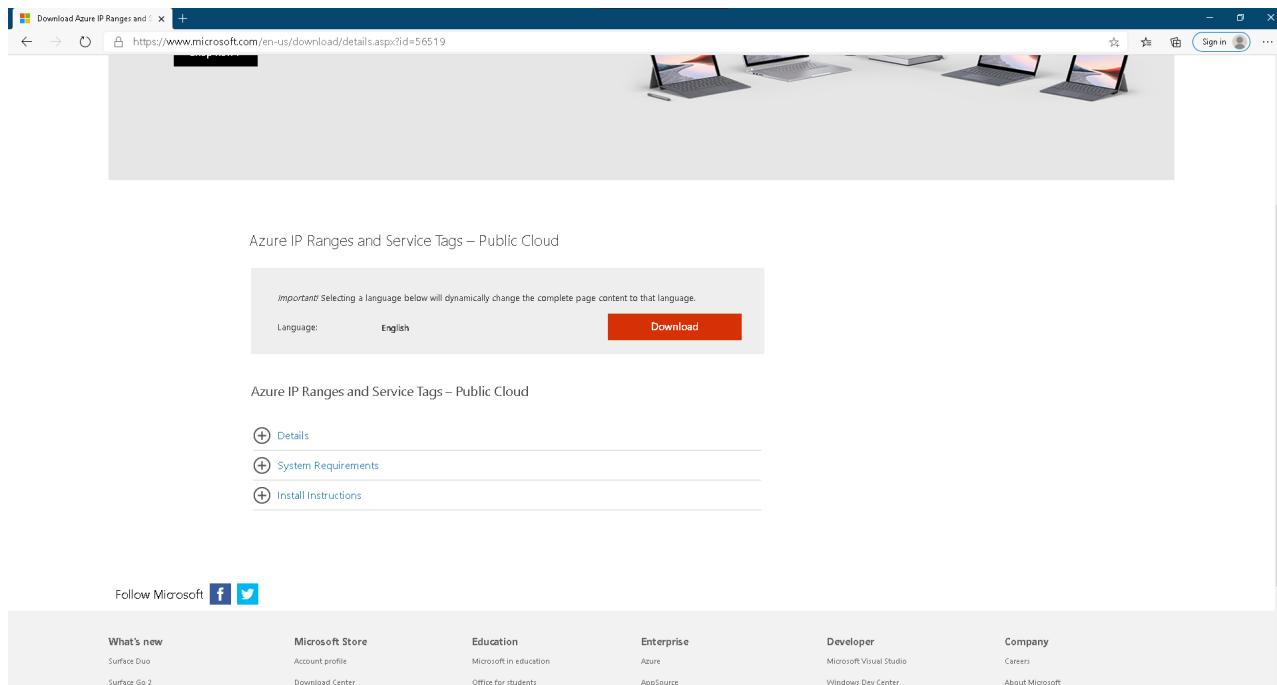
```
nslookup contoso.search.windows.net
Server: server.example.org
Address: 10.50.10.50

Non-authoritative answer:
Name: <name>
Address: 150.0.0.1
Aliases: contoso.search.windows.net
```

## Obtenir les plages d'adresses IP pour la balise de service « AzureCognitiveSearch »

Vous pouvez obtenir les plages d'adresses IP pour la balise de service `AzureCognitiveSearch` via l'[API de découverte \(préversion\)](#) ou le [fichier JSON téléchargeable](#).

Pour cette procédure pas à pas, en supposant que le service de recherche est le cloud public Azure, le [fichier JSON public Azure](#) doit être téléchargé.



Azure IP Ranges and Service Tags – Public Cloud

Important: Selecting a language below will dynamically change the complete page content to that language.

Language: English [Download](#)

Azure IP Ranges and Service Tags – Public Cloud

(+) Details  
(+) System Requirements  
(+) Install Instructions

Follow Microsoft [Facebook](#) [Twitter](#)

What's new  
Surface Duo  
Surface Go 2

Microsoft Store  
Account profile  
Download Center

Education  
Microsoft in education  
Office for students

Enterprise  
Azure  
AppSource

Developer  
Microsoft Visual Studio  
Windows Dev Center

Company  
Careers  
About Microsoft

Dans le fichier JSON, en supposant que le service de recherche se trouve dans la région USA Centre-Ouest, les adresses IP de l'environnement d'exécution de l'indexeur mutualisé suivantes sont répertoriées.

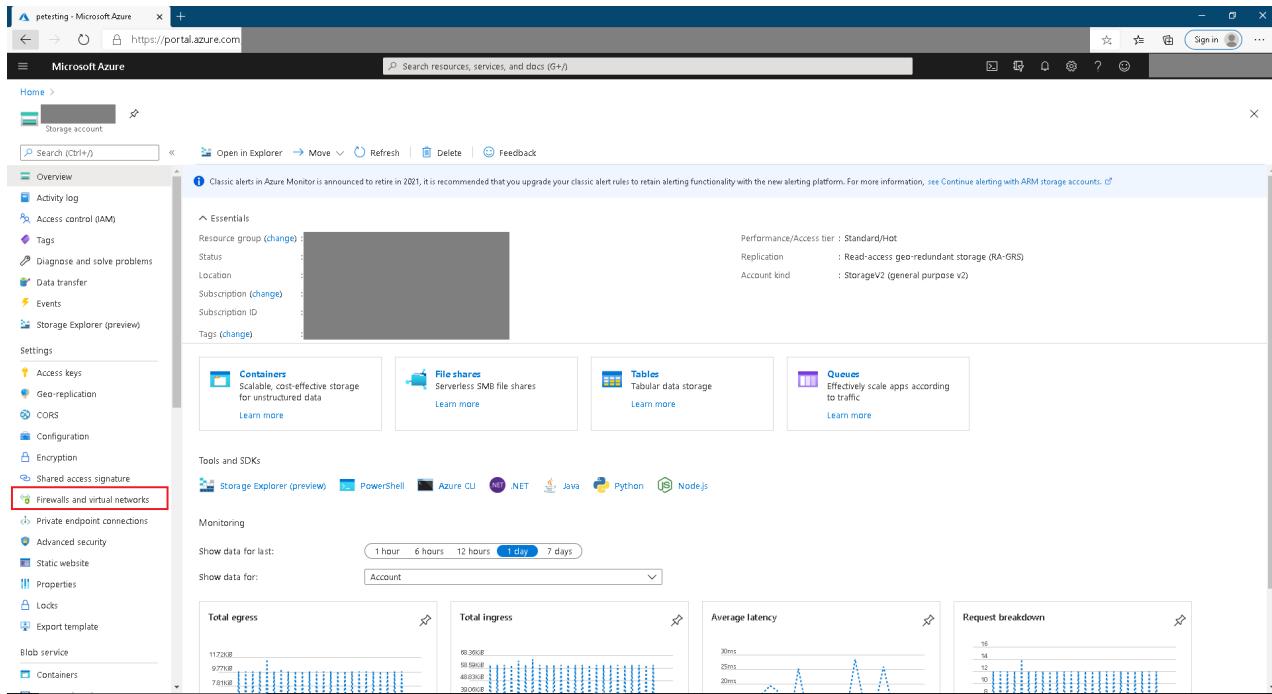
```
{
  "name": "AzureCognitiveSearch.WestCentralUS",
  "id": "AzureCognitiveSearch.WestCentralUS",
  "properties": {
    "changeNumber": 1,
    "region": "westcentralus",
    "platform": "Azure",
    "systemService": "AzureCognitiveSearch",
    "addressPrefixes": [
      "52.150.139.0/26",
      "52.253.133.74/32"
    ]
  }
}
```

Pour les adresses IP/32, supprimez la chaîne "/32" (52.253.133.74/32 -> 52.253.133.74). Vous pouvez utiliser les

autres textuellement.

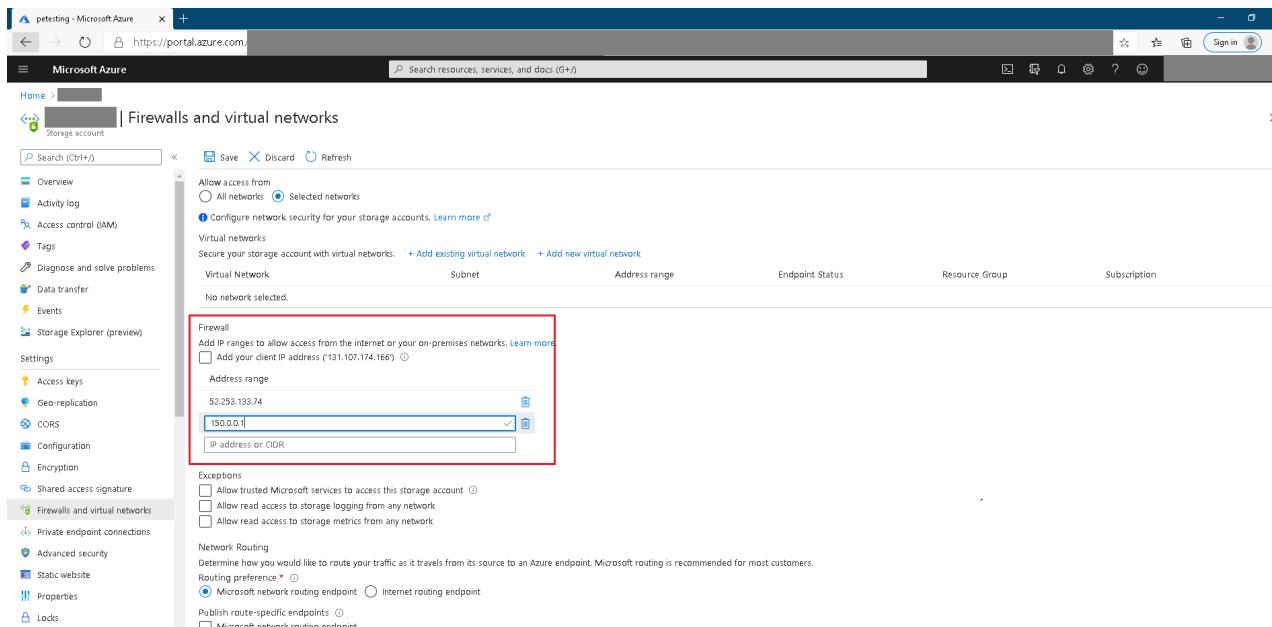
## Ajouter les plages d'adresses IP aux règles de pare-feu IP

Le moyen le plus simple d'ajouter des plages d'adresses IP à la règle de pare-feu d'un compte de stockage consiste à utiliser le portail Azure. Sur le portail, recherchez le compte de stockage sur et accédez à l'onglet «**Pare-feu et réseaux virtuels**».



The screenshot shows the Microsoft Azure portal interface for a storage account. The left sidebar lists various storage management options like Overview, Activity log, and Data transfer. The main content area displays general account information such as Resource group, Status, Location, Subscription, and Tags. Below this, there are four service tiles: Containers, File shares, Tables, and Queues. A section titled 'Tools and SDKs' includes links for Storage Explorer (preview), PowerShell, Azure CLI, .NET, Java, Python, and Node.js. The 'Monitoring' section features four charts: Total egress, Total ingress, Average latency, and Request breakdown. On the far left of the main content area, the 'Firewalls and virtual networks' link is highlighted with a red box.

Ajoutez les trois adresses IP obtenues précédemment (1 pour l'adresse IP du service de recherche, 2 pour la balise de service `AzureCognitiveSearch`) dans la plage d'adresses, puis cliquez sur « Enregistrer ».



The screenshot shows the 'Firewalls and virtual networks' configuration page for the same storage account. The left sidebar remains the same. The main area has a header with 'Save', 'Discard', and 'Refresh' buttons. It shows settings for 'Allow access from' (selected 'Selected networks') and 'Configure network security for your storage accounts'. Below this, there's a table for 'virtual networks' with columns for Virtual Network, Subnet, Address range, Endpoint Status, Resource Group, and Subscription. A note says 'No network selected.' A red box highlights the 'Address range' input field, which contains '52.253.133.74' and '150.0.0.1' separated by a comma. Below the address range, there's a 'Exceptions' section with checkboxes for allowing Microsoft services, read access to storage logging, and read access to storage metrics. At the bottom, there's a 'Network Routing' section with a note about routing traffic from source to endpoint, a 'Routing preference' dropdown set to 'Microsoft network routing endpoint', and a 'Publish route-specific endpoints' section.

La mise à jour des règles de pare-feu prend de 5 à 10 minutes après lesquelles les indexeurs peuvent accéder aux données du compte de stockage.

## Étapes suivantes

Maintenant que vous savez comment obtenir les deux ensembles d'adresses IP pour autoriser l'accès aux index, utilisez les liens suivants pour mettre à jour les règles de pare-feu IP pour certaines sources de données courantes.

- Configurer des pare-feu pour le service Stockage Azure
- Configurer un pare-feu IP pour CosmosDB
- Configurer un pare-feu IP pour Azure SQL Server

# Accéder à des ressources sécurisées par le biais de points de terminaison privés

04/10/2020 • 16 minutes to read • [Edit Online](#)

Les ressources Azure (telles que les comptes de stockage utilisés comme sources de données) peuvent être configurées pour n'être accessibles qu'à partir d'une liste spécifique de réseaux virtuels. Elles peuvent également être configurées de manière à interdire tout accès de type « réseau public ». Les clients peuvent demander au service Recherche cognitive Azure d'établir une [connexion par point de terminaison privé](#) (sortant) afin d'accéder en toute sécurité aux données de ces sources de données via des indexeurs.

## API de gestion des ressources de liaison privée partagées

Les points de terminaison privés créés par le service Recherche cognitive Azure à la demande du client pour accéder aux ressources « sécurisées » sont appelés *ressources de liaison privée partagées*. Le client « partage » l'accès à une ressource (telle qu'un compte de stockage) qui a été intégrée au [service Azure Private Link](#).

Le service Recherche cognitive Azure offre, via l'API de gestion des recherches, la possibilité de [Créer ou mettre à jour des ressources de liaison privée partagées](#). Vous utiliserez cette API avec d'autres API de gestion des *ressources de liaison privée partagées* pour configurer l'accès à une ressource sécurisée à partir d'un indexeur Recherche cognitive Azure.

Les connexions par point de terminaison privé à certaines ressources ne peuvent être établies qu'à l'aide de la préversion de l'API de gestion des recherches ([2020-08-01-Preview](#)), signalée par la mention « préversion » dans le tableau ci-dessous. Les ressources non accompagnées de la mention « préversion » peuvent être créées à la fois via l'API de la préversion et via l'API de la disponibilité générale ([2020-08-01](#))

Vous trouverez ci-dessous la liste des ressources Azure vers lesquelles des points de terminaison privés sortants peuvent être créés à partir du service Recherche cognitive Azure. Le `groupId` figurant dans le tableau ci-dessous doit être utilisé tel quel dans l'API (en respectant la casse) pour créer une ressource de liaison privée partagée.

RESSOURCE AZURE	ID DE GROUPE
Stockage Azure - Blob (ou) ADLS Gen 2	<code>blob</code>
Stockage Azure - Tables	<code>table</code>
Azure Cosmos DB - API SQL	<code>sql</code>
Azure SQL Database	<code>sqlServer</code>
Azure Database pour MySQL (préversion)	<code>mysqlServer</code>
Azure Key Vault	<code>vault</code>
Azure Functions (préversion)	<code>sites</code>

La liste des ressources Azure pour lesquelles les connexions par point de terminaison privé sortant sont prises en charge peut également être interrogée via l'[API Répertorier les ressources prises en charge](#).

Dans le cadre de ce guide, une combinaison d'[ARMClient](#) et de [Postman](#) est utilisée pour illustrer les appels d'API REST.

#### NOTE

Dans l'exemple de ce guide, le service de recherche s'appelle **contoso-search** et se trouve dans le groupe de ressources **contoso** d'un abonnement dont l'ID est **00000000-0000-0000 -0000-000000000000**. L'ID de ressource de ce service de recherche est

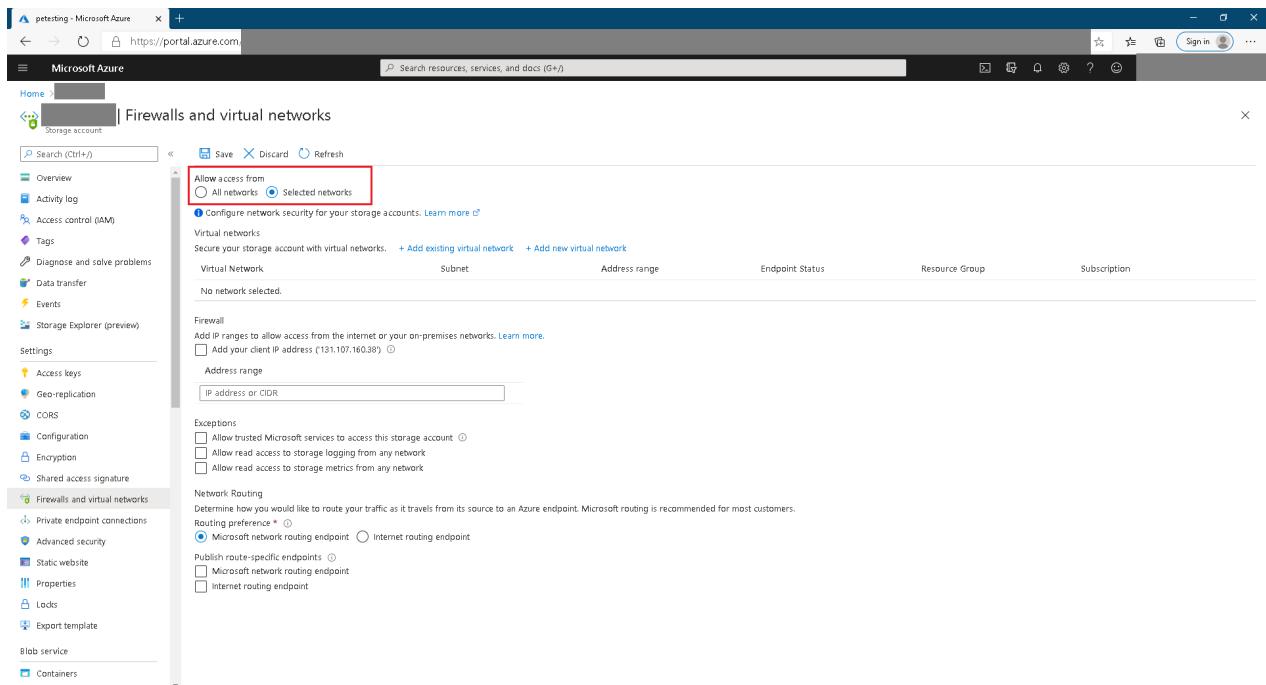
```
/subscriptions/00000000-0000-0000-0000-  
000000000000/resourceGroups/contoso/providers/Microsoft.Search/searchServices/contoso-search
```

Ce guide explique comment le service **contoso-search** peut être configuré pour que ses indexeurs aient accès aux données à partir du compte de stockage sécurisé

```
/subscriptions/00000000-0000-0000-0000-  
000000000000/resourceGroups/contoso/providers/Microsoft.Storage/storageAccounts/contoso-storage
```

## Sécuriser votre compte de stockage

Configurez le compte de stockage afin de [n'autoriser l'accès qu'à partir de sous-réseaux spécifiques](#). Via le portail Azure, si vous cochez cette option sans renseigner le reste, cela signifie qu'aucun trafic provenant d'un réseau virtuel n'est autorisé.



#### NOTE

L'[approche Service Microsoft approuvé](#) peut être utilisée pour contourner les restrictions liées aux réseaux virtuels ou aux adresses IP sur ce type de compte de stockage, et peut permettre au service de recherche d'accéder aux données du compte de stockage comme décrit dans le [guide pratique](#). Toutefois, avec cette approche, la communication entre le service Recherche cognitive Azure et le compte de stockage s'effectue via l'adresse IP publique du compte de stockage, sur le réseau principal sécurisé de Microsoft.

## Étape 1 : Créer une ressource de liaison privée partagée avec le compte de stockage

Procédez à l'appel d'API suivant pour demander au service Recherche cognitive Azure d'établir une connexion par point de terminaison privé sortant au compte de stockage

```
armclient PUT https://management.azure.com/subscriptions/00000000-0000-0000-0000-000000000000/resourceGroups/contoso/providers/Microsoft.Search/searchServices/contoso-search/sharedPrivateLinkResources/blob-pe?api-version=2020-08-01 create-pe.json
```

Le contenu du fichier `create-pe.json` (qui représente le corps de la requête adressée à l'API) est le suivant :

```
{  
    "name": "blob-pe",  
    "properties": {  
        "privateLinkResourceId": "/subscriptions/00000000-0000-0000-0000-000000000000/resourceGroups/contoso/providers/Microsoft.Storage/storageAccounts/contoso-storage",  
        "groupId": "blob",  
        "requestMessage": "please approve"  
    }  
}
```

Une réponse `202 Accepted` est renvoyée en cas de succès ; le processus de création d'un point de terminaison privé sortant est une opération longue (asynchrone). Il implique le déploiement des ressources suivantes :

1. Allocation d'un point de terminaison privé avec une adresse IP privée présentant l'état `"Pending"`. L'adresse IP privée est obtenue à partir de l'espace d'adressage alloué au réseau virtuel de l'environnement d'exécution de l'indexeur privé spécifique au service de recherche. Une fois le point de terminaison privé approuvé, toute communication entre le service Recherche cognitive Azure et le compte de stockage provient de l'adresse IP privée et d'un canal de liaison privé sécurisé.
2. Une zone DNS privée pour le type de ressource, sur la base du `groupId`. Toute recherche DNS sur la ressource privée utilisera ainsi l'adresse IP associée au point de terminaison privé.

Veuillez à spécifier le `groupId` qui convient au type de ressource pour lequel vous créez le point de terminaison privé. Toute incompatibilité générera un message d'échec.

Comme toutes les opérations Azure asynchrones, l'appel de `PUT` renvoie une valeur d'en-tête `Azure-AsyncOperation` semblable à la suivante :

```
"Azure-AsyncOperation": "https://management.azure.com/subscriptions/00000000-0000-0000-0000-000000000000/resourceGroups/contoso/providers/Microsoft.Search/searchServices/contoso-search/sharedPrivateLinkResources/blob-pe/operationStatuses/08586060559526078782?api-version=2020-08-01"
```

Cet URI peut être régulièrement interrogé pour obtenir l'état de l'opération. Nous vous recommandons d'attendre que l'état de l'opération liée à la ressource de liaison privée partagée ait atteint un état terminal (c'est-à-dire `succeeded`) avant de continuer.

```
armclient GET https://management.azure.com/subscriptions/00000000-0000-0000-0000-000000000000/resourceGroups/contoso/providers/Microsoft.Search/searchServices/contoso-search/sharedPrivateLinkResources/blob-pe/operationStatuses/08586060559526078782?api-version=2020-08-01"
```

```
{  
    "status": "running" | "succeeded" | "failed"  
}
```

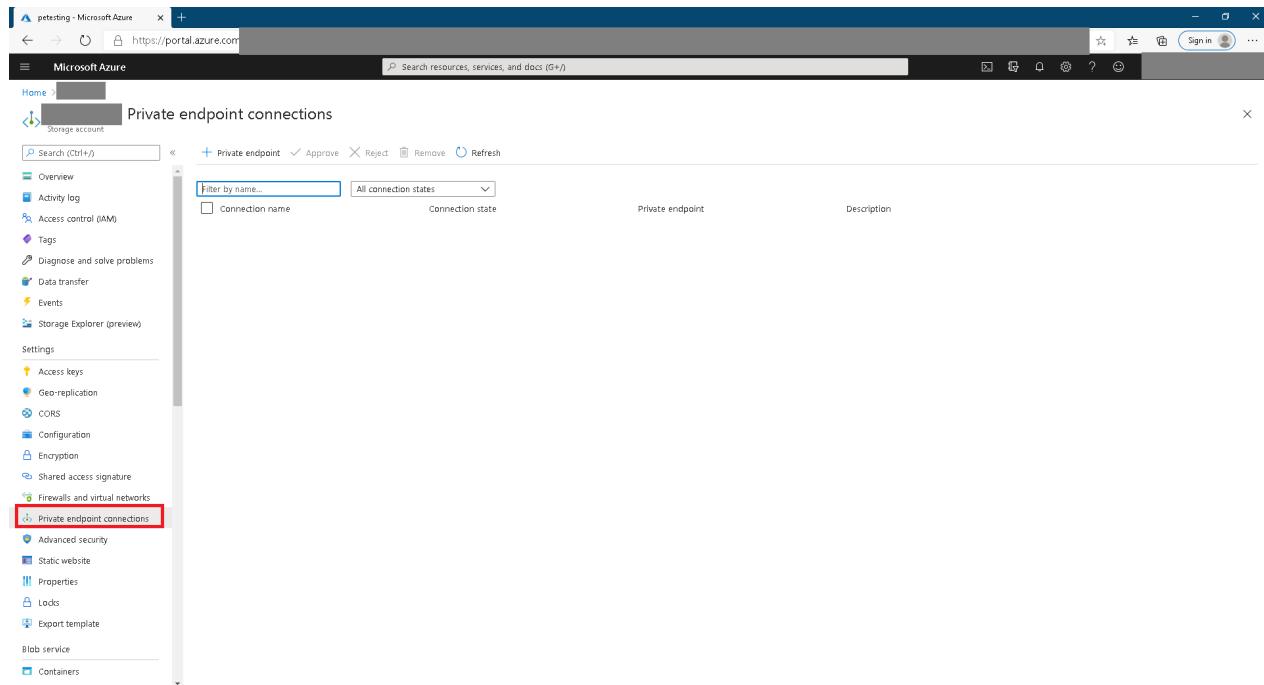
## Étape 2a : Approuver la connexion par point de terminaison privé pour le compte de stockage

## NOTE

Dans cette section, nous utiliserons le portail Azure pour suivre le processus d'approbation d'un point de terminaison privé vers le stockage. L'[API REST](#) disponible via le fournisseur de ressources de stockage peut aussi être utilisée.

D'autres fournisseurs, tels que CosmosDB et Azure SQL Server, proposent également des API de fournisseur de ressources similaires pour gérer les connexions par point de terminaison privé.

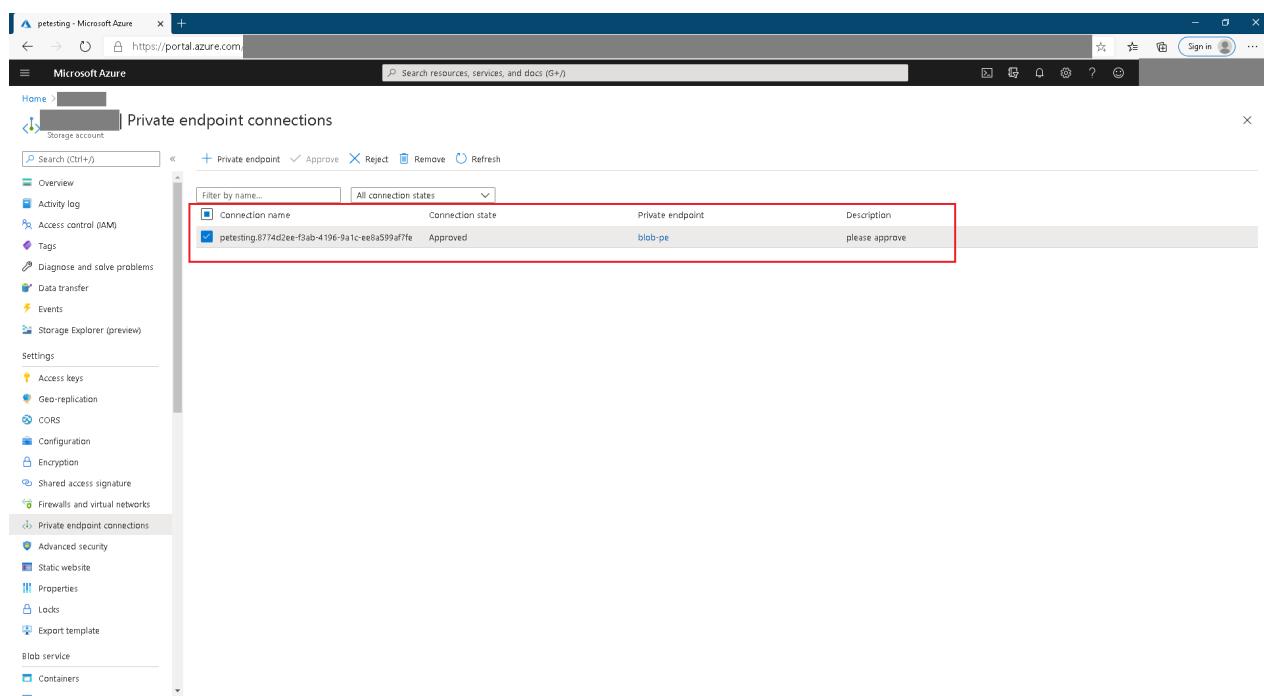
Accédez à l'onglet « **Connexions par point de terminaison privé** » du compte de stockage sur le portail Azure. Une demande de connexion par point de terminaison privé doit y figurer, avec le message de demande de l'appel d'API précédent (une fois l'opération asynchrone réussie).



The screenshot shows the Azure Storage account settings page. The left sidebar lists various storage-related options like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Data transfer, Events, Storage Explorer (preview), Settings (Access keys, Geo-replication, CORS, Configuration, Encryption, Shared access signature, Firewalls and virtual networks), Advanced security, Static website, Properties, Locks, and Export template. The 'Private endpoint connections' option is highlighted with a red box. The main content area shows a table for managing private endpoint connections:

Connection name	Connection state	Private endpoint	Description
petesting.8774d2ee-f3ab-4196-9a1c-eef8a599af7fe	Approved	blob-pe	please approve

Sélectionnez le point de terminaison privé qui a été créé par le service Recherche cognitive Azure (utilisez la colonne « Point de terminaison privé » pour identifier la connexion par point de terminaison privé grâce au nom spécifié dans l'API précédente) et choisissez « Approver », avec un message approprié (le message n'est pas important). Assurez-vous que la connexion par point de terminaison privé se présente comme suit (1 à 2 minutes peuvent être nécessaires avant que l'état soit mis à jour sur le portail).



The screenshot shows the same Azure Storage account settings page as before, but now the connection has been approved. The table in the main content area now shows the connection in the 'Approved' state:

Connection name	Connection state	Private endpoint	Description
petesting.8774d2ee-f3ab-4196-9a1c-eef8a599af7fe	Approved	blob-pe	please approve

Dès l'approbation de la demande de connexion par point de terminaison privé, le trafic *peut* transiter par le point de terminaison privé. Une fois le point de terminaison privé approuvé, le service Recherche cognitive Azure crée les mappages de zone DNS nécessaires dans la zone DNS créée pour lui.

## Étape 2b : Interroger l'état de la ressource de liaison privée partagée

Pour vérifier que la ressource de liaison privée partagée a bien été mise à jour à l'issue de l'approbation, accédez à son état à l'aide de l'[API GET](#).

```
armclient GET https://management.azure.com/subscriptions/00000000-0000-0000-0000-000000000000/resourceGroups/contoso/providers/Microsoft.Search/searchServices/contoso-search/sharedPrivateLinkResources/blob-pe?api-version=2020-08-01
```

Si la propriété `properties.provisioningState` de la ressource est `Succeeded` tandis que sa propriété `properties.status` est `Approved`, cela signifie que la ressource de liaison privée partagée est fonctionnelle et que les indexeurs peuvent être configurés pour communiquer via le point de terminaison privé.

```
{
    "name": "blob-pe",
    "properties": {
        "privateLinkResourceId": "/subscriptions/00000000-0000-0000-0000-000000000000/resourceGroups/contoso/providers/Microsoft.Storage/storageAccounts/contoso-storage",
        "groupId": "blob",
        "requestMessage": "please approve",
        "status": "Approved",
        "resourceRegion": null,
        "provisioningState": "Succeeded"
    }
}
```

## Étape 3 : Configurer l'indexeur pour qu'il s'exécute dans l'environnement privé

### NOTE

Cette opération peut être effectuée avant même que la connexion par point de terminaison privé ne soit approuvée. Tant que la connexion par point de terminaison privé n'est pas approuvée, tout indexeur qui tente de communiquer avec une ressource sécurisée (comme le compte de stockage) est en état d'échec temporaire. Il est alors impossible de créer de nouveaux indexeurs. Dès que la connexion par point de terminaison privé est approuvée, les indexeurs ont accès au compte de stockage privé.

1. [Créez une source de données](#) qui pointe vers le compte de stockage sécurisé et un conteneur approprié au sein du compte de stockage. L'exemple suivant illustre une demande effectuée via Postman.

The screenshot shows the Postman interface with a PUT request to `https://contoso-search.search.windows.net/datasources/blob/api-version=2020-06-30`. The request body is a JSON object:

```

1
2   "name": "blob",
3   "type": "azureblob",
4   "credentials": {
5     "connectionString": "some connection string"
6   },
7   "container": { "name": "container", "query": null },
8   "dataChangeDetectionPolicy": null,
9   "dataDeletionDetectionPolicy": null
10
11

```

2. De même, [créez un index](#) et, si vous le souhaitez, [créez un ensemble de compétences](#) à l'aide de l'API REST.

3. [Créez un indexeur](#) qui pointe vers la source de données, l'index et l'ensemble de compétences créés ci-dessus. En outre, forcez l'exécution de l'indexeur dans l'environnement d'exécution privé en définissant la propriété de configuration de l'indexeur `executionEnvironment` sur `"Private"`.

The screenshot shows the Postman interface with a PUT request to `https://contoso-search.search.windows.net/indexers/indexer?api-version=2020-06-30`. The request body is a JSON object:

```

1
2   "name": "indexer",
3   "description": null,
4   "dataSourceName": "blob",
5   "targetIndexName": "index",
6   "disabled": null,
7   "schedule": null,
8   "parameters": {
9     "configuration": {
10       "executionEnvironment": "private"
11     }
12   },
13   "#fieldMappings": []
14

```

Une fois créé, l'indexeur indexe le contenu du compte de stockage via la connexion par point de terminaison privé. L'état de l'indexeur peut être surveillé via l'[API d'état de l'indexeur](#).

#### NOTE

Si vous disposez déjà d'indexeurs, vous pouvez simplement les mettre à jour via l'[API PUT](#) pour définir `executionEnvironment` sur `"Private"`.

## Résolution des problèmes

- Si la création d'un indexeur échoue avec un message d'erreur de type « Les informations d'identification de la source de données ne sont pas valides », cela signifie que la connexion par point de terminaison privé n'a pas été *approuvée* ou qu'elle n'est pas fonctionnelle. Utilisez l'[API GET](#) pour accéder à l'état de la ressource de

liaison privée partagée. Si elle a été *approuvée*, vérifiez la propriété `properties.provisioningState` de la ressource. Si elle est définie sur `Incomplete`, cela signifie que certaines des dépendances sous-jacentes de la ressource n'ont pas pu être approvisionnées ; réexécutez alors la requête `PUT` pour « recréer » la ressource de liaison privée partagée, ce qui devrait résoudre le problème. Une nouvelle approbation peut être nécessaire ; vérifiez à nouveau l'état de la ressource.

- Si l'indexeur est créé sans que sa propriété `executionEnvironment` ne soit définie, la création peut aboutir, mais l'historique des exécutions de cet indexeur indiquera que celles-ci ont échoué. Vous devez [mettre à jour l'indexeur](#) pour spécifier l'environnement d'exécution.
- Si l'indexeur est créé sans que sa propriété `executionEnvironment` ne soit définie et qu'il s'exécute correctement, cela signifie que le service Recherche cognitive Azure a décidé que son environnement d'exécution est l'environnement « privé » spécifique du service de recherche. Celui-ci peut toutefois varier en fonction de différents facteurs (ressources consommées par l'indexeur, charge du service de recherche, etc.) et il peut échouer ultérieurement. Nous vous recommandons vivement de définir `executionEnvironment` sur `"Private"` pour vous assurer que l'environnement d'exécution n'échouera pas.
- Les [quotas et limites](#) déterminent le nombre de ressources de liaison privée partagée qui peuvent être créées et qui dépendent de la référence SKU du service de recherche.

## Étapes suivantes

En savoir plus sur les points de terminaison privés :

- [Présentation des points de terminaison privés](#)
- [Configurations DNS nécessaires pour les points de terminaison privés](#)

# Administration des services pour Recherche cognitive Azure sur le portail Azure

04/10/2020 • 19 minutes to read • [Edit Online](#)

Recherche cognitive Azure est un service de recherche entièrement géré, basé sur le cloud, utilisé pour la création d'une expérience de recherche riche dans des applications personnalisées. Cet article aborde les tâches d'administration des services que vous pouvez effectuer dans le [portail Azure](#) pour un service de recherche que vous avez déjà provisionné. L'administration des services est légère de par sa conception et se limite aux tâches suivantes :

- Vérifiez le stockage à l'aide du lien **Utilisation** en milieu de page.
- Vérifiez les volumes et la latence des requêtes à l'aide du lien **Surveillance** en milieu de page et vérifiez si les requêtes ont été limitées.
- Gérez l'accès à l'aide de la page **Clés** sur la gauche.
- Ajustez la capacité à l'aide de la page **Mettre à l'échelle** sur la gauche.

Les mêmes tâches effectuées dans le portail peuvent également être gérées par programmation par le biais des [API de gestion](#) et du [module AZ. Search PowerShell](#). Les tâches d'administration sont entièrement représentées sur le portail et les interfaces de programmation. Aucune tâche d'administration spécifique n'est disponible que dans une seule modalité.

Recherche cognitive Azure tire parti d'autres services Azure pour effectuer une surveillance et une gestion plus poussées. Les seules données stockées avec un service de recherche sont le contenu (index, indexeur et définitions de sources de données et autres objets). Les mesures exposées sur les pages du portail sont tirées de journaux internes sur un cycle de 30 jours continu. Pour la rétention des journaux et des événements supplémentaires contrôlés par l'utilisateur, vous devez disposer d'[Azure Monitor](#).

## Propriétés du service fixe

Plusieurs aspects d'un service de recherche sont déterminés lorsque le service est approvisionné et ne peuvent pas être modifiés ultérieurement :

- Nom du service (vous ne pouvez pas renommer un service)
- Emplacement du service (vous ne pouvez pas déplacer actuellement un service intact vers une autre région)
- Nombre maximal de réplicas et de partitions (déterminé par le niveau, De base ou Standard)

Si vous avez démarré avec le niveau De base et le nombre maximal de partitions associé, et qu'il vous faut plus de partitions, vous devez [créer un service](#) à un niveau supérieur et recréer votre contenu sur le nouveau service.

## Droits d'administrateur

L'approvisionnement ou le retrait du service lui-même peut être effectué par un administrateur d'abonnement Azure ou un coadministrateur.

En ce qui concerne l'accès au point de terminaison, toute personne ayant accès à l'URL du service et une clé API a accès au contenu. Pour plus d'informations sur les clés, consultez la page [Gestion des clés API](#).

- L'accès en lecture seule au service est un droit de requête, généralement accordé à une application cliente en lui fournissant l'URL et une clé API de requête.
- L'accès en lecture-écriture permet d'ajouter, de supprimer ou de modifier des objets serveur, notamment des

clés API, des index, des indexeurs, des sources de données et des planifications. L'accès en lecture-écriture est accordé par la fourniture de l'URL, d'une clé API administrateur.

Les droits d'accès à l'appareil d'approvisionnement de services sont accordés par le biais d'attributions de rôles. Le [contrôle d'accès en fonction du rôle Azure \(Azure RBAC\)](#) est un système d'autorisation basé sur [Azure Resource Manager](#) pour l'approvisionnement de ressources Azure.

Dans le cadre de Recherche cognitive Azure, les [attributions de rôles Azure](#) déterminent les personnes qui peut effectuer les tâches, qu'elles utilisent le [portail](#), [PowerShell](#) ou les [API REST de gestion](#) :

- Créer ou supprimer un service
- Mettre à l'échelle le service
- Supprimer ou régénérer les clés API
- Activer la journalisation des diagnostics (créer des services)
- Activer Traffic Analytics (créer des services)

#### TIP

En utilisant des mécanismes à l'échelle d'Azure, vous pouvez verrouiller un abonnement ou une ressource pour empêcher la suppression accidentelle ou non autorisée de votre service de recherche par les utilisateurs disposant de droits d'administration. Pour plus d'informations, consultez [Verrouiller les ressources pour en empêcher la suppression](#).

## Journalisation et informations système

Au niveau de base et supérieur, Microsoft surveille tous les services Recherche cognitive Azure pour vérifier la disponibilité de 99,9 % par contrat de niveau de service (SLA). Si le service est lent ou si le débit des demandes tombe en dessous des seuils de contrat SLA, les équipes de support passent en revue les fichiers journaux à leur disposition et résolvent le problème.

Recherche cognitive Azure tire parti d'[Azure Monitor](#) pour collecter et stocker les activités d'indexation et de requête. Un service de recherche en lui-même stocke uniquement son contenu (index, définitions d'indexeurs, définitions de sources de données, définitions d'ensembles de compétences, cartes de synonymes). Les informations de mise en cache et journalisées sont stockées hors service, souvent dans un compte de stockage Azure. Pour plus d'informations sur la journalisation des charges de travail d'indexation et de requête, consultez [Collecter et analyser les données de journal](#).

En matière d'informations générales sur votre service, en utilisant seulement les fonctionnalités intégrées à Recherche cognitive Azure lui-même, vous permet d'obtenir des informations des manières suivantes :

- À l'aide du service depuis la page [Présentation](#), via les notifications, les propriétés et les messages d'état.
- À l'aide de [PowerShell](#) ou de l'[API REST de gestion](#) pour [obtenir les propriétés du service](#). Aucune nouvelle information ou opération n'est fournie au niveau de la couche de programmation. Des interfaces vous permettent d'écrire des scripts.

## surveiller l'utilisation des ressources ;

Sur le tableau de bord, l'analyse des ressources se limite aux informations affichées dans le tableau de bord des services et à quelques mesures que vous pouvez obtenir en interrogeant le service. Dans la section Utilisation du tableau de bord des services, vous pouvez déterminer rapidement si les niveaux des ressources de partition sont adaptés à votre application. Vous pouvez provisionner des ressources externes, telles que la supervision Azure, si vous souhaitez capturer et conserver les événements enregistrés. Pour plus d'informations, consultez [Supervision de Recherche cognitive Azure](#).

L'API REST du service de recherche vous permet d'obtenir le nombre de documents et d'index par

programmation :

- [Obtention de statistiques d'index](#)
- [Nombre de documents](#)

## Récupération d'urgence et pannes de service

Bien que nous puissions récupérer vos données, Recherche cognitive Azure ne fournit pas de basculement instantané du service en cas de panne au niveau du centre de données ou du cluster. Si un cluster tombe en panne dans le centre de données, l'équipe d'exploitation le détecte et tente de restaurer le service. Vous rencontrerez un temps d'arrêt pendant la restauration de service, mais vous pouvez demander des crédits de service afin de compenser une indisponibilité du service conformément au [Contrat de niveau de service \(SLA\)](#).

Si le service ne doit pas être interrompu même en cas de défaillances catastrophiques qui échappent au contrôle de Microsoft, vous pouvez [approvisionner un service supplémentaire](#) dans une autre région et mettre en œuvre une stratégie de géoréplication pour assurer une redondance complète des index sur tous les services.

Les clients qui utilisent des [indexeurs](#) pour remplir et actualiser les index peuvent gérer la récupération d'urgence par le biais d'indexeurs propres à la région qui exploitent la même source de données. Deux services situés dans des régions différentes, chacun exécutant un indexeur, peuvent indexer la même source de données pour bénéficier de la géoredondance. Si vous indexez à partir de sources de données qui sont aussi géoredondantes, sachez que les indexeurs de Recherche cognitive Azure ne peuvent assurer qu'une indexation incrémentielle (en fusionnant les mises à jour de documents nouveaux, modifiés ou supprimés) à partir de répliques principaux. À l'occasion d'un basculement, veillez à refaire pointer l'indexeur vers le nouveau réplica principal.

Si vous n'utilisez pas d'indexeurs, vous devez utiliser le code de votre application pour effectuer une transmission de type push des objets et des données vers différents services de recherche en parallèle. Pour plus d'informations, consultez [Performance and optimization in Azure Search](#)(Performances et optimisation dans Recherche cognitive Azure).

## Sauvegarde et restauration

Recherche cognitive Azure n'est pas une solution de stockage de données principal, c'est pourquoi nous ne fournissons pas de mécanisme formel de sauvegarde et de restauration en libre-service. Toutefois, vous pouvez utiliser l'exemple de code **index-backup-restore** dans cet [exemple de dépôt .NET Recherche cognitive Azure](#) pour sauvegarder la définition et l'instantané d'un index dans une série de fichiers JSON, puis utiliser ces fichiers pour restaurer l'index, si nécessaire. Cet outil peut également déplacer des index entre les niveaux de service.

Sinon, le code de votre application utilisé pour créer et remplir un index est l'option de restauration de facto si vous supprimez un index par erreur. Pour reconstruire un index, vous devez le supprimer (s'il existe), recréer l'index dans le service et le recharger en récupérant les données à partir de votre banque de données principale.

## Augmentation ou réduction d'échelle

Au départ, chaque service de recherche comporte, au minimum, un réplica et une partition. Si vous êtes inscrit à un [niveau qui prend en charge plus de capacités](#), cliquez sur **Mettre à l'échelle** dans le volet de navigation gauche pour ajuster l'utilisation des ressources.

Lorsque vous ajoutez des capacités à travers l'une des ressources, le service les utilise automatiquement. Aucune autre action n'est nécessaire de votre part. Cependant, un léger décalage est à prévoir avant que l'effet de la nouvelle ressource soit perceptible. L'approvisionnement des ressources supplémentaires demande au moins 15 minutes.

### Ajout de répliques

Pour augmenter le nombre de requêtes par seconde (RPS) ou parvenir à une haute disponibilité, il convient

d'ajouter des réplicas. Chaque réplica comporte une copie d'un index. L'ajout d'un réplica se traduit donc par un ou plusieurs index supplémentaires disponibles pour satisfaire les demandes de requête. Au moins 3 réplicas sont nécessaires pour la haute disponibilité (pour plus d'informations, consultez [Régler la capacité](#)).

Un service de recherche qui comporte davantage de réplicas peut équilibrer la charge des demandes de requête sur un plus grand nombre d'index. Pour un nombre de requêtes donné, le débit sera plus élevé si davantage de copies de l'index sont disponibles pour leur traitement. Si vous constatez une latence des requêtes, la mise en ligne des répliques supplémentaires aura, à n'en pas douter, un effet positif sur les performances.

Bien que l'ajout de réplicas à votre service entraîne une augmentation du débit des requêtes, celui-ci n'est pas exactement multiplié par deux ou par trois. Toutes les applications de recherche sont, en effet, soumises à des facteurs externes susceptibles d'affecter les performances des requêtes. La complexité des requêtes et la latence réseau, notamment, sont deux facteurs qui contribuent à la fluctuation des temps de réponse des requêtes.

### Ajout de partitions

Il est plus courant d'ajouter des réplicas, mais lorsque le stockage est restreint, vous pouvez ajouter des partitions pour obtenir plus de capacité. La possibilité d'ajouter ou non des partitions dépend du niveau auquel vous avez approvisionné le service. Le niveau De base est verrouillé sur une partition. Les niveaux Standard et supérieurs prennent en charge des partitions supplémentaires.

Les partitions sont ajoutées par diviseurs de 12 (à savoir, 1, 2, 3, 4, 6 ou 12). Il s'agit d'un artefact de partitionnement. Un index est créé dans 12 fragments (ou shards) qui peuvent tous être stockés dans 1 partition ou répartis équitablement dans 2, 3, 4, 6 ou 12 partitions (un fragment par partition).

### Suppression de réplicas

Après une période de traitement de requêtes intensive, vous pouvez utiliser le curseur pour réduire le nombre de réplicas une fois la charge de requêtes de recherche revenue à la normale (à la fin d'une période de soldes, par exemple). Rien de plus ! La réduction du nombre de réplicas entraîne l'abandon des machines virtuelles dans le centre de données. Désormais, vos opérations de requête et d'ingestion de données s'exécuteront sur un nombre moins élevé de machines virtuelles. La configuration minimale nécessite un réplica.

### Suppression de partitions

Contrairement à la suppression de réplicas, qui n'exige aucune opération supplémentaire de votre part, utiliser plus de volume de stockage que la capacité disponible après réduction peut entraîner une surcharge de travail. Par exemple, si votre solution utilise trois partitions, le fait de passer à une ou deux partitions générera une erreur si le nouvel espace de stockage est inférieur à celui nécessaire pour héberger votre index. Comme vous pouvez vous y attendre, deux solutions s'offrent alors à vous : soit supprimer des index ou documents au sein d'un index associé afin de libérer de l'espace, soit conserver la configuration actuelle.

Aucune méthode de détection ne vous permet d'identifier les fragments d'index qui sont stockés sur des partitions spécifiques. Chaque partition fournit environ 25 Go de stockage. Vous devrez donc réduire l'espace de stockage à une taille pouvant être prise en charge par le nombre de partitions dont vous disposez. Si vous souhaitez revenir à une seule partition, celle-ci devra contenir les 12 fragments.

Pour faciliter la planification, vous pouvez vérifier le stockage (voir la page [Obtenir des statistiques d'index](#)) afin de connaître l'espace réellement utilisé.

## Étapes suivantes

- Automatiser avec [PowerShell](#)
- Réviser le [niveau de performance et les techniques d'optimisation](#)
- Réviser les [fonctionnalité de sécurité](#) pour protéger le contenu et les opérations
- Activer la [journalisation des diagnostics](#) pour surveiller les charges de travail de requête et d'indexation

# Gérer votre service Recherche cognitive Azure avec PowerShell

04/10/2020 • 13 minutes to read • [Edit Online](#)

Vous pouvez exécuter des scripts et des applets de commande PowerShell sur Windows, Linux ou dans [Azure Cloud Shell](#) pour créer et configurer Recherche cognitive Azure. Le module **Az.Search** étend [Azure PowerShell](#) avec une parité complète pour les [API REST de gestion de la recherche](#) et la capacité à réaliser les tâches suivantes :

- [Lister les services de recherche dans un abonnement](#)
- [Retourner les informations sur les services](#)
- [Créer ou supprimer un service](#)
- [Régénérer des clés API d'administration](#)
- [Créer ou supprimer des clés API de requête](#)
- [Effectuer un scale-up ou un scale-down avec les répliques et les partitions](#)

Parfois, des questions sont posées sur des tâches qui ne figurent *pas* dans la liste ci-dessus. Vous ne pouvez pas utiliser le module **Az.Search** ou l'API REST de gestion pour changer un nom de serveur, une région ou un niveau. Des ressources dédiées sont allouées lorsqu'un service est créé. Ainsi, toute modification du matériel sous-jacent (emplacement ou type de noeud) nécessite un nouveau service. De même, il n'existe pas d'API ou d'outils pour transférer du contenu, tel qu'un index, d'un service à un autre.

Au sein d'un service, la création et la gestion de contenu s'effectuent par le biais de l'[API REST du service de recherche](#) ou du [SDK .NET](#). Bien qu'il n'existe aucune commande PowerShell dédiée au contenu, vous pouvez écrire un script PowerShell qui appelle des API REST ou .NET afin de créer et de charger des index.

## Vérifier les versions et charger des modules

Les exemples de cet article sont interactifs et requièrent des autorisations élevées. Azure PowerShell (module Az) doit être installé. Pour plus d'informations, consultez l'article [Installation et configuration d'Azure PowerShell](#).

### Vérification de la version PowerShell (5.1 ou ultérieure)

La version locale de PowerShell doit correspondre à la version 5.1 ou ultérieure, sur n'importe quel système d'exploitation pris en charge.

```
$PSVersionTable.PSVersion
```

### Charger Azure PowerShell

Si vous ne savez pas si Az est installé, exécutez la commande suivante pour le vérifier.

```
Get-InstalledModule -Name Az
```

Certains systèmes ne chargent pas automatiquement les modules. Si la commande précédente génère une erreur, essayez de charger le module, et si cela échoue, consultez à nouveau les instructions d'installation pour vous assurer que vous n'avez oublié aucune étape.

```
Import-Module -Name Az
```

## Se connecter à Azure avec un jeton de connexion de navigateur

Vous pouvez utiliser vos informations d'identification de connexion au portail pour vous connecter à un abonnement dans PowerShell. Vous pouvez également [vous authentifier de manière non interactive avec un principal de service](#).

```
Connect-AzAccount
```

Si vous disposez de plusieurs abonnements Azure, définissez votre abonnement Azure à utiliser. Pour afficher une liste de vos abonnements en cours, exécutez la commande suivante.

```
Get-AzSubscription | sort SubscriptionName | Select SubscriptionName
```

Pour spécifier l'abonnement, exécutez la commande suivante. Dans l'exemple suivant, le nom de l'abonnement est `ContosoSubscription`.

```
Select-AzSubscription -SubscriptionName ContosoSubscription
```

## Répertorier les services dans un abonnement

Les commandes suivantes proviennent d'[Az.Resources](#) et renvoient des informations sur les ressources et services existants déjà approvisionnés dans votre abonnement. Si vous ne savez pas combien de services de recherche sont déjà créés, ces commandes vous en informent et vous évitent d'accéder au portail.

La première commande renvoie tous les services de recherche.

```
Get-AzResource -ResourceType Microsoft.Search/searchServices | ft
```

Dans la liste des services, elle permet d'obtenir des informations sur une ressource spécifique.

```
Get-AzResource -ResourceName <service-name>
```

Les résultats se présentent comme suit.

```
Name          : my-demo-searchapp
ResourceGroupName : demo-westus
ResourceType    : Microsoft.Search/searchServices
Location       : westus
ResourceId     : /subscriptions/<alpha-numeric-subscription-ID>/resourceGroups/demo-westus/providers/Microsoft.Search/searchServices/my-demo-searchapp
```

## Importer Az.Search

Les commandes provenant d'[Az.Search](#) ne sont pas disponibles tant que vous n'avez pas chargé le module.

```
Install-Module -Name Az.Search
```

## Répertorier toutes les commandes Az.Search

En guise de vérification, renvoyez une liste de commandes fournies dans le module.

```
Get-Command -Module Az.Search
```

Les résultats se présentent comme suit.

CommandType	Name	Version	Source
Cmdlet	Get-AzSearchAdminKeyPair	0.7.1	Az.Search
Cmdlet	Get-AzSearchQueryKey	0.7.1	Az.Search
Cmdlet	Get-AzSearchService	0.7.1	Az.Search
Cmdlet	New-AzSearchAdminKey	0.7.1	Az.Search
Cmdlet	New-AzSearchQueryKey	0.7.1	Az.Search
Cmdlet	New-AzSearchService	0.7.1	Az.Search
Cmdlet	Remove-AzSearchQueryKey	0.7.1	Az.Search
Cmdlet	Remove-AzSearchService	0.7.1	Az.Search
Cmdlet	Set-AzSearchService	0.7.1	Az.Search

## Obtenir des informations sur le service de recherche

Après avoir importé Az.Search et déterminé le groupe de ressources contenant votre service de recherche, exécutez [Get-AzSearchService](#) pour renvoyer la définition de service, notamment le nom, la région, le niveau, ainsi que le nombre de répliques et de partitions.

```
Get-AzSearchService -ResourceGroupName <resource-group-name>
```

Les résultats se présentent comme suit.

```
Name      : my-demo-searchapp
ResourceGroupName : demo-westus
ResourceType   : Microsoft.Search/searchServices
Location     : West US
Sku          : Standard
ReplicaCount  : 1
PartitionCount : 1
HostingMode   : Default
ResourceId    : /subscriptions/<alphanumeric-subscription-ID>/resourceGroups/demo-westus/providers/Microsoft.Search/searchServices/my-demo-searchapp
```

## Créer ou supprimer un service

[New-AzSearchService](#) permet de [créer un service de recherche](#).

```
New-AzSearchService -ResourceGroupName "demo-westus" -Name "my-demo-searchapp" -Sku "Standard" -Location
"West US" -PartitionCount 3 -ReplicaCount 3
```

Les résultats se présentent comme suit.

```
ResourceGroupName : demo-westus
Name              : my-demo-searchapp
Id                : /subscriptions/<alphanumeric-subscription-ID>/demo-
westus/providers/Microsoft.Search/searchServices/my-demo-searchapp
Location         : West US
Sku               : Standard
ReplicaCount     : 3
PartitionCount   : 3
HostingMode      : Default
Tags
```

## Régénération des clés d'administration

**New- AzSearchAdminKey** permet de substituer les [clés API](#) d'administration. Deux clés d'administration sont créées avec chaque service à des fins d'accès authentifié. Ces clés sont nécessaires pour chaque requête. Les deux clés d'administration fonctionnent de la même manière, et accordent au service de recherche un accès total en écriture, avec possibilité de récupérer des informations ou de créer et de supprimer des objets. Deux clés sont disponibles pour vous permettre d'en utiliser une lorsque vous remplacez l'autre.

Vous ne pouvez régénérer qu'une clé à la fois, spécifiée en tant que clé `primary` ou `secondary`. Pour ne pas interrompre le service, pensez à mettre à jour la totalité du code client de manière à utiliser une clé secondaire lors de la substitution de la clé primaire. Évitez tout changement de clés lorsque des opérations sont en cours.

En effet, si vous régénérez les clés sans mettre à jour le code client, les requêtes utilisant l'ancienne clé n'aboutiront pas. La régénération de toutes les nouvelles clés ne bloque indéfiniment votre service, et vous en conservez l'accès via le portail. Une fois les clés primaire et secondaire régénérées, vous pouvez mettre à jour le code client pour utiliser les nouvelles clés, après quoi les opérations reprennent.

Les valeurs correspondant aux clés API sont générées par le service. Vous ne pouvez pas fournir de clé personnalisée à des fins d'utilisation par Recherche cognitive Azure. De même, il n'existe aucun nom défini par l'utilisateur pour les clés API d'administration. Les références à la clé correspondent à des chaînes fixes, `primary` ou `secondary`.

```
New-AzSearchAdminKey -ResourceGroupName <resource-group-name> -ServiceName <search-service-name> -KeyKind Primary
```

Les résultats se présentent comme suit. Les deux clés sont renvoyées, même si vous n'en modifiez qu'une.

Primary	Secondary
-----	-----
<alphanumeric-guid>	<alphanumeric-guid>

## Créer ou supprimer des clés de requête

**New- AzSearchQueryKey** permet de créer des [clés API](#) de requête à des fins d'accès en écriture seule à partir d'applications clientes vers un index Recherche cognitive Azure. Les clés de requête sont utilisées pour s'authentifier auprès d'un index spécifique et récupérer les résultats de la recherche. Les clés de requête n'accordent pas d'accès en lecture seule à d'autres éléments du service, comme un index, une source de données ou un indexeur.

Vous ne pouvez pas fournir de clé à des fins d'utilisation par Recherche cognitive Azure. Les clés API sont générées par le service.

```
New-AzSearchQueryKey -ResourceGroupName <resource-group-name> -ServiceName <search-service-name> -Name <query-key-name>
```

## Mettre à l'échelle des réplicas et des partitions

[Set-AzSearchService](#) permet d'[augmenter ou de diminuer les réplicas et les partitions](#) pour réajuster les ressources facturables de votre service. Augmenter les réplicas ou les partitions se répercute sur votre facture, qui présente des frais fixes et variables. En cas de besoin temporaire de plus de puissance de traitement, vous pouvez augmenter les réplicas et les partitions pour gérer la charge de travail. La zone de surveillance de la page Vue d'ensemble du portail présente des vignettes sur la latence de requête, les requêtes par seconde et les limitations, indiquant si la capacité actuelle est suffisante.

L'ajout ou la suppression de ressources peut prendre un certain temps. Des ajustements de la capacité interviennent en arrière-plan, ce qui permet la poursuite des charges de travail existantes. Une capacité supplémentaire est utilisée pour les demandes entrantes, dès qu'elle est prête, sans configuration supplémentaire.

La suppression de la capacité peut entraîner des perturbations. Il est recommandé d'arrêter tous les travaux d'indexation avant de réduire la capacité pour éviter la suppression de requêtes. Si ce n'est pas possible, essayez de réduire la capacité de façon incrémentielle, un seul réplica et une seule partition à la fois, jusqu'à ce que vos nouveaux niveaux cibles soient atteints.

Il n'est pas possible d'arrêter la commande une fois celle-ci en cours d'exécution. Vous devez attendre qu'elle prenne fin avant d'examiner les résultats.

```
Set-AzSearchService -ResourceGroupName <resource-group-name> -Name <search-service-name> -PartitionCount 6 -ReplicaCount 6
```

Les résultats se présentent comme suit.

```
ResourceGroupName : demo-westus
Name            : my-demo-searchapp
Location        : West US
Sku             : Standard
ReplicaCount    : 6
PartitionCount  : 6
HostingMode     : Default
Id              : /subscriptions/65a1016d-0f67-45d2-b838-b8f373d6d52e/resourceGroups/demo-westus/providers/Microsoft.Search/searchServices/my-demo-searchapp
```

## Étapes suivantes

Génération d'un [index](#), [interrogation d'un index](#) à l'aide du portail, des API REST ou du kit de développement logiciel (SDK) .NET.

- [Créer un index Recherche cognitive Azure dans le portail Azure](#)
- [Configuration d'un indexeur pour charger des données provenant d'autres services](#)
- [Interrogation d'un index de Recherche cognitive Azure à l'aide de l'Explorateur de recherche dans le Portail Azure](#)
- [Utilisation de la Recherche cognitive Azure dans .NET](#)

# Déplacer votre service Recherche cognitive Azure vers une autre région Azure

04/10/2020 • 4 minutes to read • [Edit Online](#)

Parfois, les clients demandent à déplacer un service de recherche vers une autre région. Actuellement, il n'existe aucun mécanisme ou outil intégré pour faciliter cette tâche, mais cet article peut vous aider à comprendre les étapes manuelles permettant d'obtenir le même résultat.

## NOTE

Dans le portail Azure, tous les services ont une commande **Exporter le modèle**. Dans le cas de la Recherche cognitive Azure, cette commande produit une définition de base d'un service (nom, localisation, niveau, réplica et nombre de partitions), mais ne reconnaît pas le contenu de votre service, pas plus que les clés, les rôles ou les journaux. Bien que la commande existe, nous vous déconseillons de l'utiliser pour déplacer un service de recherche.

## Aide pour déplacer un service

1. Identifiez les dépendances et les services associés pour comprendre l'impact complet du déplacement d'un service, au cas où vous devriez déplacer plus que simplement la Recherche cognitive Azure.

Le stockage Azure est utilisé pour la journalisation, la création d'une base de connaissances et est une source de données externe couramment utilisée pour l'enrichissement et l'indexation par IA. Cognitive Services est une dépendance dans l'enrichissement par IA. Cognitive Services et votre service de recherche doivent se trouver dans la même région si vous utilisez l'enrichissement par IA.

2. Créez un inventaire de tous les objets sur le service afin de savoir ce qu'il faut déplacer : les index, les cartes de synonymes, les indexeurs, les sources de données, les ensembles de compétences. Si vous avez activé la journalisation, créez et archivez les rapports dont vous pouvez avoir besoin pour un enregistrement historique.
3. Vérifiez les tarifs et la disponibilité dans la nouvelle région pour garantir la disponibilité de la Recherche cognitive Azure ainsi que tous les services associés dans la même région. La majorité des fonctionnalités sont disponibles dans toutes les régions, mais certaines fonctionnalités d'évaluation ont une disponibilité limitée.
4. Créez un service dans la nouvelle région et republiez à partir du code source les index, les cartes de synonymes, les indexeurs, les sources de données et les ensembles de compétences. N'oubliez pas que les noms de service doivent être uniques afin que vous ne puissiez pas réutiliser le nom existant. Vérifiez chaque ensemble de compétences pour voir si les connexions à Cognitive Services sont toujours valides dans le cadre de la spécification de la même région. En outre, si vous créez des bases de connaissances, vérifiez les chaînes de connexion pour le stockage Azure si vous utilisez un autre service.
5. Rechargez les index et les bases de connaissances, le cas échéant. Vous allez utiliser du code d'application pour envoyer (push) des données JSON à un index ou réexécuter des indexeurs pour tirer (pull) des documents de sources externes.
6. Activez la journalisation et, si vous les utilisez, recréez les rôles de sécurité.
7. Mettez à jour les applications clientes et les suites de tests pour utiliser le nouveau nom de service et les clés API, puis testez toutes les applications.

8. Supprimez l'ancien service une fois que le nouveau service est entièrement testé et opérationnel.

## Étapes suivantes

Les liens suivants peuvent vous aider à trouver plus d'informations lorsque vous effectuez les étapes décrites ci-dessus.

- [Tarifs et régions Recherche cognitive Azure](#)
- [Choisir un niveau](#)
- [Créer un service de recherche](#)
- [Charger les documents de recherche](#)
- [Activer la journalisation](#)

# Définir des rôles Azure pour l'accès d'administration à Recherche cognitive Azure

04/10/2020 • 4 minutes to read • [Edit Online](#)

Azure offre un [modèle d'autorisation par rôle global](#) pour tous les services gérés via le portail ou les API Resource Manager. Les rôles Propriétaire, Contributeur et Lecteur définissent le niveau *d'administration des services* pour les utilisateurs, les groupes et les principaux de sécurité Active Directory assignés à chaque rôle.

## NOTE

Il n'existe aucun contrôle d'accès en fonction du rôle (RBAC) pour sécuriser des parties d'un index ou d'un sous-ensemble de documents. Pour un accès en fonction de l'identité sur les résultats de recherche, vous pouvez créer des filtres de sécurité pour ajuster les résultats par identité, en supprimant les documents auxquels le demandeur ne doit pas avoir accès. Pour plus d'informations, consultez [Filtres de sécurité](#) et [Sécuriser avec Active Directory](#).

## Tâches de gestion par rôle

Pour Recherche cognitive Azure, les rôles sont associés à des niveaux d'autorisation qui prennent en charge les tâches de gestion suivantes :

ROLE	TÂCHE
Propriétaire	<p>Création ou suppression du service ou de tout objet sur le service, y compris les clés API, les index, les indexeurs, les sources de données d'indexeur et les planifications de l'indexeur.</p> <p>Afficher l'état du service, notamment des compteurs et la taille du stockage.</p> <p>Ajout ou suppression d'appartenance à un rôle (seul un Propriétaire peut gérer l'appartenance à un rôle).</p> <p>Les administrateurs d'abonnement et de service appartiennent automatiquement au rôle Propriétaire.</p>
Contributeur	Même niveau d'accès que le Propriétaire, à l'exception de la gestion des rôles Azure. Par exemple, un Contributeur peut créer ou supprimer des objets ou afficher et régénérer des <a href="#">clés API</a> , mais il ne peut pas modifier l'appartenance aux rôles.
Rôle intégré du Contributeur Search Service	Équivalent au rôle de contributeur.
Lecteur	Affichez les bases et les métriques du service. Les membres de ce rôle ne peuvent pas afficher l'index, l'indexeur, la source de données ni les informations clés.

Les rôles n'accordent pas de droits d'accès au point de terminaison de service. Les opérations du service Search telles que la gestion ou le remplissage d'index, tout comme les requêtes de données de recherche, sont contrôlées via des clés api, et non par des rôles. Pour plus d'informations, consultez [Gérer des clés API](#).

## Tableau des autorisations

Le tableau suivant récapitule les opérations autorisées dans Recherche cognitive Azure, en indiquant la clé qui déverrouille l'accès à une opération particulière.

OPÉRATION	AUTORISATIONS
Créer un service	Détenteur de l'abonnement Azure
Mettre à l'échelle un service	Clé d'administration, Propriétaire ou Collaborateur RBAC sur la ressource
Supprimer un service	Clé d'administration, Propriétaire ou Collaborateur RBAC sur la ressource
Créer, modifier et supprimer des objets du service : Index et composants (y compris les définitions de l'analyseur, les profils de scoring et les options CORS), indexeurs, sources de données, synonymes et générateurs de suggestions	Clé d'administration, Propriétaire ou Collaborateur RBAC sur la ressource
Interroger un index	Clé d'administration ou clé de requête (RBAC non applicable)
Interroger des informations système, telles que l'obtention de statistiques, de comptes et de listes d'objets	Clé d'administration, RBAC sur la ressource (Propriétaire, Collaborateur ou Lecteur)
Gérer les clés d'administration	Clé d'administration, Propriétaire ou Collaborateur RBAC sur la ressource
Gérer les clés de requête	Clé d'administration, Propriétaire ou Collaborateur RBAC sur la ressource

## Voir aussi

- [Gestion à l'aide de PowerShell](#)
- [Performances et optimisation dans Recherche cognitive Azure](#)
- [Découvrir le contrôle d'accès en fonction du rôle dans le portail Azure.](#)

# Superviser les opérations et l'activité de Recherche cognitive Azure

04/10/2020 • 11 minutes to read • [Edit Online](#)

Cet article consiste en une vue d'ensemble des concepts et des outils de monitoring pour la Recherche cognitive Azure. Pour une surveillance complète, vous pouvez utiliser conjointement des fonctionnalités intégrées et des services complémentaires comme Azure Monitor.

Cela vous permettra d'effectuer le suivi des éléments suivants :

- Service : intégrité/disponibilité et modifications apportées à la configuration du service.
- Stockage : à la fois utilisé et disponible, avec le volume de chaque type de contenu par rapport au quota autorisé pour le niveau de service.
- Activité de requête : volume, latence et requêtes limitées ou annulées. [Azure Monitor](#) est nécessaire pour les demandes de requêtes journalisées.
- Activité d'indexation : [journalisation des diagnostics](#) requise avec Azure Monitor.

Comme un service de recherche ne prend pas en charge l'authentification par utilisateur, aucune information d'identité ne sera trouvée dans les journaux.

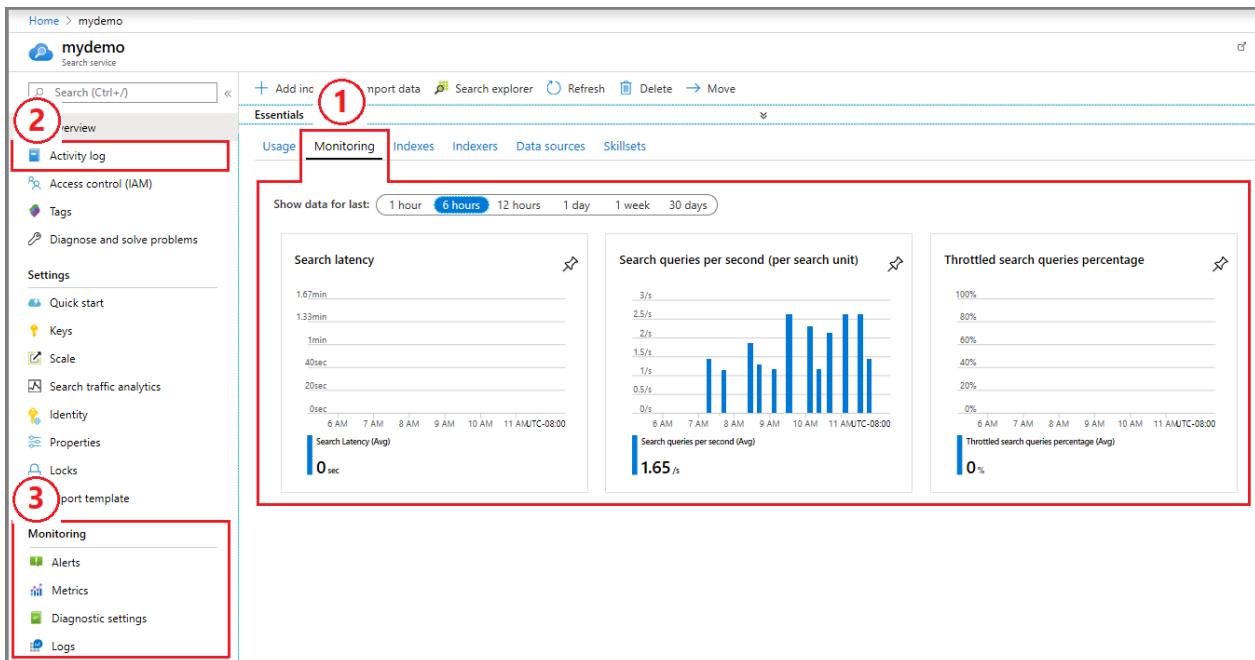
## Monitoring intégré

Le monitoring intégré fait référence aux activités journalisées par un service de recherche. À l'exception des diagnostics, aucune configuration n'est requise pour ce niveau de surveillance.

La Recherche cognitive Azure conserve les données internes suivant un calendrier de 30 jours continus pour le reporting sur l'intégrité du service et les métriques de requête, que vous trouverez sur le portail ou par le biais de ces [API REST](#).

La capture d'écran suivante vous aide à trouver les informations de monitoring sur le portail. Les données sont disponibles dès que vous commencez à utiliser le service. Les pages du portail sont actualisées à des intervalles de quelques minutes.

- L'onglet **Monitoring**, sur la page Vue d'ensemble principale, présente le volume de requêtes et la latence, et indique si le service est sous pression.
- Le **Journal d'activité**, dans le volet de navigation gauche, est connecté à Azure Resource Manager. Le journal d'activité signale les actions effectuées par Resource Manager : la disponibilité et l'état du service, les modifications de la capacité (réplicas et partitions) et les activités liées à la clé API.
- Les paramètres **Monitoring**, plus bas, proposent des alertes configurables, des métriques et des journaux de diagnostic. Créez-les lorsque vous en avez besoin. Une fois les données collectées et stockées, vous pouvez interroger ou visualiser les informations pour obtenir des insights.



## NOTE

Dans la mesure où les pages du portail sont actualisées à quelques minutes d'intervalle, les volumes rapportés sont approximatifs et destinés à donner une idée générale de l'efficacité avec laquelle le système traite les demandes. Les métriques réelles, telles que les requêtes par seconde, peuvent être supérieures ou inférieures au nombre indiqué dans la page. Si la précision fait partie des exigences, envisagez d'utiliser des API.

## API utiles pour le monitoring

Vous pouvez utiliser les API suivantes pour récupérer les mêmes informations que dans les onglets Monitoring et Utilisation du portail.

- [GET Service Statistics](#)
- [GET Index Statistics](#)
- [GET Document Counts](#)
- [GET Indexer Status](#)

## Journaux d'activité et intégrité du service

La page [Journal d'activité](#) du portail collecte des informations auprès d'Azure Resource Manager et signale les changements de l'intégrité du service. Vous pouvez superviser les conditions critiques, d'erreur et d'avertissement relatives à l'intégrité du service dans le journal d'activité.

Parmi les entrées courantes figurent les références à des clés API, à savoir des notifications d'information génériques comme *Obtenir une clé d'administration* et *Obtenir des clés de requête*. Ces activités indiquent les demandes effectuées à l'aide de la clé d'administration (créer ou supprimer des objets) ou d'une clé de requête, mais n'affichent pas la demande proprement dite. Pour plus d'informations de ce fragment, vous devez configurer la journalisation des diagnostics.

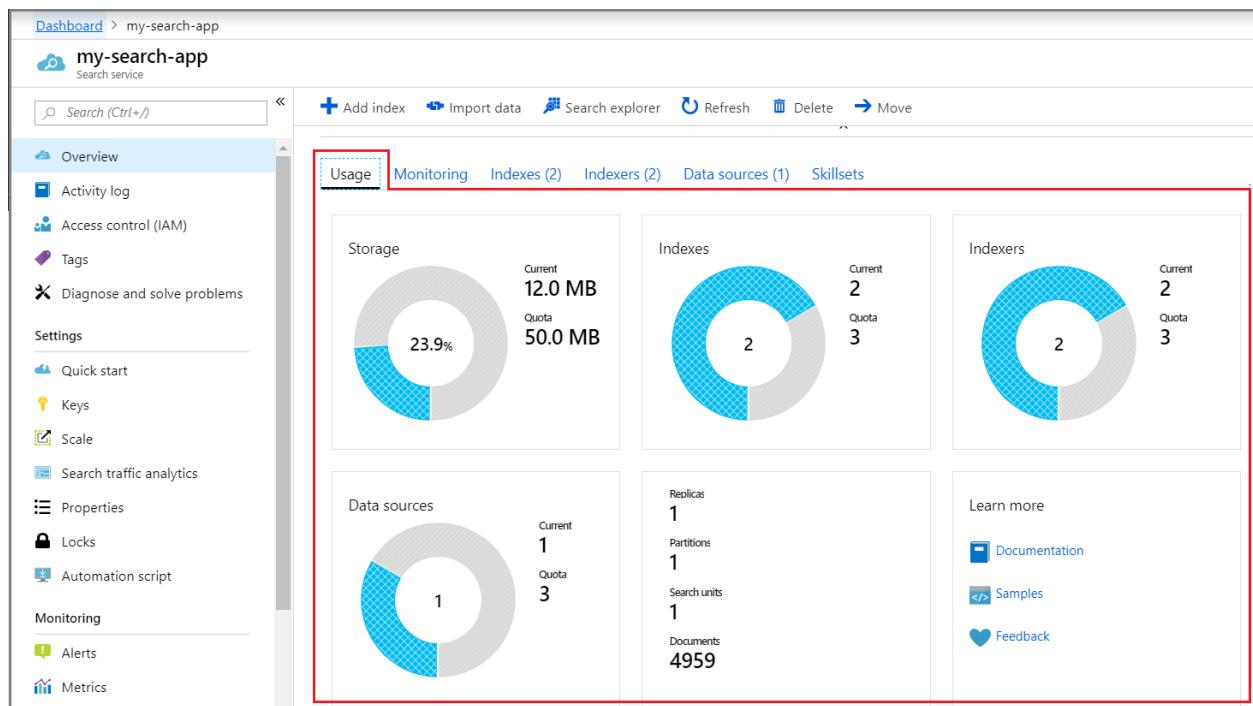
Vous pouvez accéder au [journal d'activité](#) à partir du volet de navigation de gauche, à partir de l'option Notifications dans la barre de commandes située en haut de la fenêtre ou à partir de la page [Diagnostiquer et résoudre les problèmes](#).

## Monitoring du stockage dans l'onglet Utilisation

Pour la supervision visuelle dans le portail, l'onglet **Utilisation** affiche la disponibilité des ressources par rapport aux [limites](#) actuelles imposées par le niveau de service. Si vous finalisez des décisions concernant le [niveau à utiliser pour les charges de travail de production](#) ou la nécessité d'[ajuster le nombre de partitions et de réplicas actifs](#), ces métriques peuvent vous aider dans vos décisions en vous montrant rapidement les ressources

consommées et l'efficacité de la configuration actuelle pour gérer la charge existante.

L'illustration suivante concerne le service gratuit, qui est limité à 3 objets de chaque type et à 50 Mo de stockage. Un service De base ou Standard a des limites plus élevées. De plus, si vous augmentez le nombre de partitions, le stockage maximal augmente proportionnellement.



#### NOTE

Les alertes liées au stockage ne sont pas disponibles pour l'instant ; la consommation de stockage n'est pas agrégée ni consignée dans la table **AzureMetrics** dans Azure Monitor. Pour obtenir des alertes de stockage, vous devez [créer une solution personnalisée](#) qui émet des notifications liées aux ressources et dont le code contrôle la taille de stockage et gère la réponse.

## Module complémentaire de surveillance avec Azure Monitor

De nombreux services, notamment la Recherche cognitive Azure, proposent une intégration avec [Azure Monitor](#) pour offrir des alertes et des métriques supplémentaires, et permettre la journalisation des données de diagnostic.

[Activez la journalisation des diagnostics](#) pour un service de recherche si vous souhaitez contrôler la collecte et le stockage des données. Les événements journalisés capturés par Azure Monitor sont stockés dans la table **AzureDiagnostics** et se composent de données opérationnelles liées aux requêtes et à l'indexation.

Azure Monitor propose plusieurs options de stockage ; votre choix détermine la façon dont vous pouvez consommer les données :

- Choisissez le Stockage Blob Azure si vous souhaitez [visualiser les données de journal](#) dans un rapport Power BI.
- Choisissez Log Analytics si vous souhaitez explorer les données au moyen de requêtes Kusto.

Azure Monitor possède sa propre structure de facturation ; les journaux de diagnostic mentionnés dans cette section présentent un coût. Pour plus d'informations, consultez [Utilisation et coûts estimés dans Azure Monitor](#).

## Superviser l'accès des utilisateurs

Étant donné que les index de recherche sont un composant d'une plus grande application cliente, il n'existe

aucune méthodologie intégrée pour contrôler ni superviser l'accès par utilisateur à un index. Les demandes sont supposées provenir d'une application cliente, que ce soit pour des demandes d'administration ou de requête. Les opérations de lecture-écriture d'administration incluent la création, la mise à jour et la suppression d'objets dans l'ensemble du service. Les opérations en lecture seule sont des requêtes sur la collection de documents, délimitées à un seul index.

Par conséquent, ce que vous verrez dans les journaux d'activité sont des références à des appels avec des clés d'administration ou des clés de requête. La clé appropriée est incluse dans les demandes provenant du code client. Le service n'est pas équipé pour gérer les jetons d'identité ni l'emprunt d'identité.

Quand il existe des exigences commerciales pour l'autorisation par utilisateur, la recommandation est l'intégration à Azure Active Directory. Vous pouvez utiliser \$filter et des identités utilisateur pour [filtrer les résultats de recherche](#) des documents qu'un utilisateur ne doit pas voir.

Il n'existe aucun moyen de consigner ces informations séparément de la chaîne de requête qui inclut le paramètre \$filter. Consultez [Superviser les requêtes](#) pour plus d'informations sur la création de rapports sur les chaînes de requête.

## Étapes suivantes

Une bonne maîtrise d'Azure Monitor est essentielle pour la surveillance de tout service Azure, dont des ressources comme Recherche cognitive Azure. Si vous n'êtes pas familiarisé avec Azure Monitor, prenez le temps de consulter les articles relatifs aux ressources. En plus des tutoriels, l'article suivant est un bon point de départ.

[Supervision de ressources Azure avec Azure Monitor](#)

# Collecter et analyser des données de journal pour Recherche cognitive Azure

04/10/2020 • 15 minutes to read • [Edit Online](#)

Les journaux de diagnostic ou opérationnels fournissent des informations sur les opérations détaillées de Recherche cognitive Azure et sont utiles pour surveiller les processus de charge de travail et de service. En interne, des informations système existent sur le back-end pendant une courte période, suffisante pour l'investigation et l'analyse si vous émettez un ticket de support. Toutefois, si vous souhaitez utiliser la direction automatique sur les données opérationnelles, vous devez configurer un paramètre de diagnostic pour spécifier l'emplacement où les informations de journalisation sont collectées.

La journalisation des diagnostics est activée via l'intégration à [Azure Monitor](#).

Quand vous configurez la journalisation des diagnostics, vous êtes invité à spécifier un mécanisme de stockage. Le tableau suivant énumère les options de collecte et de persistance des données.

RESSOURCE	UTILISÉ POUR
<a href="#">Envoyer à l'espace de travail Log Analytics</a>	Les événements et les mesures sont envoyés à un espace de travail Log Analytics, lequel peut être interrogé dans le portail pour retourner des informations détaillées. Pour une introduction sur le sujet, consultez <a href="#">Prise en main des journaux d'activité Azure Monitor</a> .
<a href="#">Archiver avec le stockage Blob</a>	Les événements et les mesures sont archivés dans un conteneur de blobs et stockés dans des fichiers JSON. Les journaux d'activité peuvent être très granulaires (par heure/minute), ce qui est utile pour la recherche d'un incident spécifique, mais pas pour une investigation ouverte. Utilisez un éditeur JSON pour afficher un fichier journal brut ou Power BI pour agréger et visualiser les données du journal.
<a href="#">Diffuser vers un Event Hub</a>	Les événements et les mesures sont diffusés vers un service Azure Event Hubs. Choisissez cette option comme autre service de collecte de données pour les journaux d'activité très volumineux.

## Prérequis

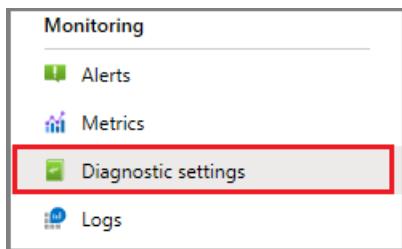
Créez des ressources à l'avance pour pouvoir en sélectionner une ou plusieurs lors de la configuration de la journalisation des diagnostics.

- [Créer un espace de travail Log Analytics](#)
- [Créez un compte de stockage](#)
- [Créer un hub d'événements](#)

## Activer la collecte des données

Les paramètres de diagnostic spécifient la façon dont les événements et les mesures journalisés sont collectés.

1. Sous **Supervision**, sélectionnez **Paramètres de diagnostic**.



2. Sélectionnez + Ajouter un paramètre de diagnostic.

3. Cochez **Log Analytics**, sélectionnez votre espace de travail, puis sélectionnez **OperationLogs** et **AllMetrics**.

A screenshot of the 'Diagnostics settings' configuration page. At the top, there are three buttons: 'Save', 'Discard', and 'Delete'. Below that is a 'Name' field containing 'demo-diagnostics' with a green checkmark. There are two unchecked options: 'Archive to a storage account' and 'Stream to an event hub'. A large red box highlights the 'Send to Log Analytics' section. Inside this section, the 'Subscription' dropdown is set to 'demo-subscription', and the 'Log Analytics workspace' dropdown is set to 'demo-westus2 (westus2)'. Below these, under 'log', 'OperationLogs' is checked. Under 'metric', 'AllMetrics' is checked. Both of these sections are also enclosed in a red box.

4. Enregistrez le paramètre.

5. Une fois la journalisation activée, utilisez votre service de recherche pour commencer à générer des journaux d'activité et des mesures. Cela prendra du temps avant que les mesures et les événements journalisés ne deviennent disponibles.

Pour Log Analytics, il faudra plusieurs minutes avant que les données ne soient disponibles, après quoi vous pourrez exécuter des requêtes Kusto pour retourner des données. Pour plus d'informations, consultez [Surveiller les demandes de requête](#).

Pour le Stockage Blob, une heure est nécessaire pour que les conteneurs s'y affichent. Il y a un seul objet blob par heure et par conteneur. Les conteneurs sont créés uniquement quand il existe une activité à journaliser ou à mesurer. Quand les données sont copiées dans un compte de stockage, les données sont au format JSON et sont placées dans deux conteneurs :

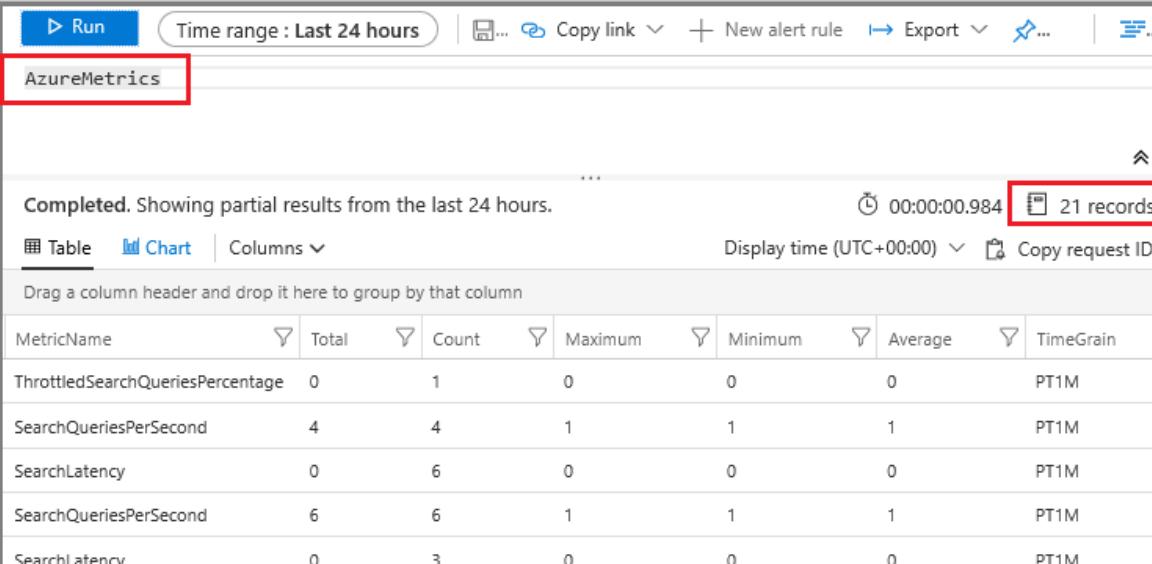
- journaux d'activité-Insights-operationlogs : pour les journaux d'activité de trafic de recherche

- mesures-Insights-pt1m : pour les mesures

## Informations sur le journal des requêtes

Deux tables contiennent des journaux d'activité et des mesures pour Recherche cognitive Azure : **AzureDiagnostics** et **AzureMetrics**.

- Sous **Supervision**, sélectionnez **Journaux d'activité**.
- Entrez **AzureMetrics** dans la fenêtre de requête. Exécutez cette requête simple pour vous familiariser avec les données collectées dans cette table. Faites défiler la table pour afficher les mesures et les valeurs. Notez le nombre d'enregistrements indiqué en haut et, si votre service a collecté des mesures pendant un certain temps, vous souhaiterez peut-être ajuster l'intervalle de temps pour obtenir un jeu de données gérable.



The screenshot shows the Azure Metrics query results table. The table has the following columns: MetricName, Total, Count, Maximum, Minimum, Average, and TimeGrain. The data rows are:

MetricName	Total	Count	Maximum	Minimum	Average	TimeGrain
ThrottledSearchQueriesPercentage	0	1	0	0	0	PT1M
SearchQueriesPerSecond	4	4	1	1	1	PT1M
SearchLatency	0	6	0	0	0	PT1M
SearchQueriesPerSecond	6	6	1	1	1	PT1M
SearchLatency	0	3	0	0	0	PT1M

At the top of the table, it says "Completed. Showing partial results from the last 24 hours." and "Display time (UTC+00:00)". There are also "Copy link" and "Copy request ID" buttons. The "Time range" is set to "Last 24 hours". The "AzureMetrics" table name is highlighted with a red box at the top left. The "21 records" count is highlighted with a red box at the top right.

- Entrez la requête suivante pour retourner un jeu de résultats tabulaire.

```
AzureMetrics
| project MetricName, Total, Count, Maximum, Minimum, Average
```

- Répétez les étapes précédentes, en commençant par **AzureDiagnostics** pour retourner toutes les colonnes à titre d'information, suivies d'une requête plus sélective qui extrait des informations plus intéressantes.

```
AzureDiagnostics
| project OperationName, resultSignature_d, DurationMs, Query_s, Documents_d, IndexName_s
| where OperationName == "Query.Search"
```

AzureDiagnostics  
| project OperationName, resultSignature\_d, DurationMs, Query\_s, Documents\_d, IndexName\_s  
| where OperationName == "Query.Search"

Completed. Showing partial results from the last 24 hours.

00:00:01.386 23 records

Table Chart Columns

Drag a column header and drop it here to group by that column

Operation...	resu...	Du...	Query_s		Doc...	IndexName_s
> Query.Search	200	8	?api-version=2019-05-06&search=seattle%20hotels		4	hotels-quicks...
> Query.Search	200	107	?api-version=2019-05-06&search=seattle%20hotels&%24count=true		4	hotels-quicks...
> Query.Search	200	68	?api-version=2019-05-06&search=washington&%24count=true		0	hotels-quicks...
> Query.Search	200	148	?api-version=2019-05-06&search=*&%24count=true&%24select=Hot...		4	hotels-quicks...
> Query.Search	200	636	?api-version=2019-05-06&search=restaurant&%24count=true&%24se...		1	hotels-quicks...
> Query.Search	200	128	?api-version=2019-05-06&search=*		14	demoindex
> Query.Search	200	138	?api-version=2017-11-11&search=*&%24count=true		14	demoindex
> Query.Search	200	118	?api-version=2019-05-06&search=*		19	hotel-reviews

## Exemples de requêtes Kusto

Si vous avez activé la journalisation des diagnostics, vous pouvez interroger **AzureDiagnostics** pour obtenir la liste des opérations exécutées sur votre service et à quel moment. Vous pouvez également mettre en corrélation l'activité pour examiner les changements de performances.

### Exemple : Lister les opérations

Retournez une liste d'opérations et un nombre de chacune d'elles.

```
AzureDiagnostics  
| summarize count() by OperationName
```

### Exemple : Mettre en corrélation les opérations

Associez une demande de requête à des opérations d'indexation et restituez les points de données sur un graphique de temps pour voir si les opérations coïncident.

```
AzureDiagnostics  
| summarize OperationName, Count=count()  
| where OperationName in ('Query.Search', 'Indexing.Index')  
| summarize Count=count(), AvgLatency=avg(DurationMs) by bin(TimeGenerated, 1h), OperationName  
| render timechart
```

## Opérations journalisées

Les événements journalisés capturés par Azure Monitor incluent ceux qui sont liés à l'indexation et aux requêtes. La table **AzureDiagnostics** dans Log Analytics collecte les données opérationnelles relatives aux requêtes et à l'indexation.

NOM OPÉRATION	DESCRIPTION
ServiceStats	Cette opération est un appel de routine à l'API <a href="#">GET Service Statistics</a> , appelée directement ou implicitement pour remplir une page de présentation du portail lors de son chargement ou de son actualisation.

NOM OPÉRATION	DESCRIPTION
Query.Search	Demandes de requêtes par rapport à un index. Consultez <a href="#">Superviser les requêtes</a> pour plus d'informations sur les requêtes journalisées.
Indexing.Index	Cette opération est un appel à <a href="#">Ajout, mise à jour ou suppression de documents</a> .
indexes.Prototype	Il s'agit d'un index créé par l'Assistant Importation de données.
Indexers.Create	Créez un indexeur de manière explicite ou implicite à l'aide de l'Assistant Importation de données.
Indexers.Get	Retourne le nom d'un indexeur chaque fois que l'indexeur est exécuté.
Indexers.Status	Retourne l'état d'un indexeur chaque fois que l'indexeur est exécuté.
DataSources.Get	Retourne le nom de la source de données chaque fois qu'un indexeur est exécuté.
Indexes.Get	Retourne le nom d'un index chaque fois qu'un indexeur est exécuté.

## Schéma du journal

Si vous créez des rapports personnalisés, les structures de données qui contiennent des données de journal Recherche cognitive Azure sont conformes au schéma ci-dessous. Pour le Stockage Blob, chaque blob a un objet racine appelé **records** contenant un tableau d'objets de journal. Chaque objet blob contient des enregistrements de toutes les opérations qui ont eu lieu au cours de la même heure.

Le tableau suivant est une liste partielle des champs communs à la journalisation des ressources.

NOM	TYPÉ	EXEMPLE	NOTES
timeGenerated	DATETIME	"2018-12-07T00:00:43.6872559Z"	Horodatage de l'opération
resourceId	string	«/SUBSCRIPTIONS/11111111-1111-1111-1111-111111111111/RESOURCEGROUPS/DEFAULTR/PROVIDERS/MICROSOFT.SEARCH/SEARCHSERVICES/SEARCHSERVICE»	Votre ID de ressource
operationName	string	« Query.Search »	Nom de l'opération
operationVersion	string	"30/06/2020"	Version d'API utilisée
catégorie	string	« OperationLogs »	constant

NOM	TYPE	EXEMPLE	NOTES
resultType	string	« Success »	Valeurs possibles : Réussite ou Échec
resultSignature	int	200	Code de résultat HTTP
durationMS	int	50	Durée de l'opération en millisecondes
properties	object	consultez le tableau suivant	Objet contenant des données propres à l'opération

## Schéma de propriétés

Les propriétés ci-dessous sont spécifiques à Recherche cognitive Azure.

NOM	TYPE	EXEMPLE	NOTES
Description_s	string	« GET /indexes('content')/docs »	Point de terminaison de l'opération
Documents_d	int	42	Nombre de documents traités
IndexName_s	string	"test-index"	Nom de l'index associé à l'opération
Query_s	string	"?search=AzureSearch&\$count=true&api-version=2020-06-30"	Paramètres de requête

## Schéma de mesures

Les mesures sont capturées pour les demandes de requête et mesurées à des intervalles d'une minute. Chaque mesure expose des valeurs minimales, maximales et moyennes par minute. Pour plus d'informations, consultez [Surveiller les demandes de requête](#).

NOM	TYPE	EXEMPLE	NOTES
resourceId	string	«/SUBSCRIPTIONS/11111111-1111-1111-1111-111111111111/RESOURCEGROUPS/DEFAULTRPROVIDERS/MICROSOFT.SEARCH/SEARCHSERVICES/SEARCHSERVICE»	votre ID de ressource
metricName	string	« Latency »	Nom de la mesure
time	DATETIME	"2018-12-07T00:00:43.6872559Z"	Horodatage de l'opération

NOM	TYPÉ	EXEMPLE	NOTES
average	int	64	Valeur moyenne des échantillons bruts dans l'intervalle de temps de la mesure, en secondes ou en pourcentage, en fonction de la mesure.
minimum	int	37	Valeur minimale des échantillons bruts dans l'intervalle de temps de la mesure, en secondes.
maximum	int	78	Valeur maximale des échantillons bruts dans l'intervalle de temps de la mesure, en secondes.
total	int	258	Valeur totale des échantillons bruts dans l'intervalle de temps de la mesure, en secondes.
count	int	4	Nombre de mesures émises à partir d'un nœud dans le journal pendant l'intervalle d'une minute.
timegrain	string	« PT1M »	Fragment de temps de la mesure au format ISO 8601.

Les requêtes sont souvent exécutées en quelques millisecondes, par conséquent seules les requêtes mesurées en secondes, QPS par exemple, apparaissent dans les métriques.

Dans le cas de la mesure **Search Queries Per Second**, la valeur minimale correspondra à la valeur la plus faible des requêtes de recherche par seconde qui a été enregistrée pendant cette minute. Il en va de même pour la valeur maximale. La moyenne représentera l'agrégat de ces valeurs pour toute la minute. Par exemple, sur une minute, le schéma peut être le suivant : une seconde de charge élevée, qui représente votre valeur SearchQueriesPerSecond maximale, puis 58 secondes de charge moyenne, et enfin une seconde avec une seule requête, qui représente la valeur minimale.

Pour **Pourcentage de requêtes de recherche limitées**, les valeurs minimales, maximales, moyennes et totales seront identiques : il s'agit du pourcentage de requêtes de recherche qui ont été limitées, en fonction du nombre total de requêtes de recherche pendant une minute.

## Afficher les fichiers journaux bruts

Le Stockage Blob est utilisé pour l'archivage des fichiers journaux. Vous pouvez utiliser n'importe quel éditeur JSON pour voir le fichier journal. Si vous n'en avez pas, nous vous recommandons [Visual Studio Code](#).

1. Dans le portail Azure, ouvrez votre compte de stockage.
2. Dans le volet de navigation de gauche, cliquez sur **Objets blob.insights-logs-operationlogs et insights-metrics-pt1m** doivent s'afficher. Ces conteneurs sont créés par Recherche cognitive Azure quand les données de journal sont exportées vers le stockage Blob.

3. Faites défiler l'arborescence des dossiers vers le bas jusqu'à atteindre le fichier .json. Utilisez le menu contextuel pour télécharger le fichier.

Une fois le fichier téléchargé, ouvrez-le dans un éditeur JSON pour en voir le contenu.

## Étapes suivantes

Si ce n'est déjà fait, passez en revue les principes de base de la surveillance des services de recherche pour en savoir plus sur les différentes capacités de surveillance disponibles.

[Surveiller les opérations et l'activité dans Recherche cognitive Azure](#)

# Visualiser les journaux et les métriques Recherche cognitive Azure avec Power BI

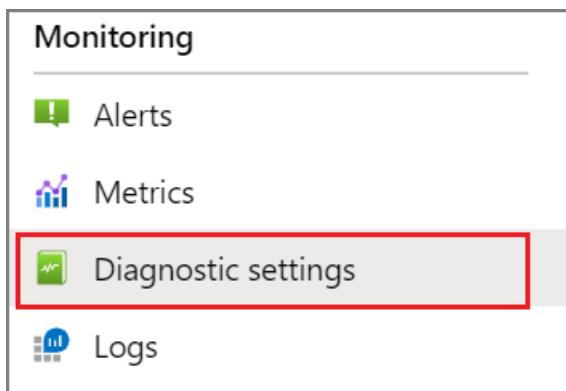
04/10/2020 • 7 minutes to read • [Edit Online](#)

Recherche cognitive Azure vous permet de stocker les journaux des opérations et les métriques de service relatifs à votre service de recherche dans un compte de stockage Azure. Cette page fournit des instructions sur la façon dont vous pouvez visualiser ces informations par le biais d'une application modèle Power BI. L'application fournit des informations détaillées sur votre service de recherche, notamment des informations sur les métriques Recherche, Indexation, Opérations et Service.

L'application modèle Power BI **Recherche cognitive Azure : Analyser les journaux et les métriques** est accessible dans le [marketplace d'applications Power BI](#).

## Prise en main de l'application

1. Activez la journalisation des métriques et des ressources pour votre service de recherche :
  - a. Créez ou identifiez un [compte Stockage Azure](#) existant dans lequel vous pouvez archiver les journaux.
  - b. Accédez à votre service Recherche cognitive Azure dans le Portail Azure.
  - c. Dans la section Supervision de la colonne gauche, sélectionnez **Paramètres de diagnostic**.



- ...
  - d. Sélectionnez + Ajouter un paramètre de diagnostic.
  - e. Activez la case **Archiver dans un compte de stockage**, fournissez les informations de votre compte de stockage, puis cochez les cases **OperationLogs** et **AllMetrics**.

Name	demo-diagnostics-settings
<input checked="" type="checkbox"/> Archive to a storage account	
Storage account >	
logsandmetricsstorage	
<input type="checkbox"/> Stream to an event hub	
<input type="checkbox"/> Send to Log Analytics	
log	
<input checked="" type="checkbox"/> OperationLogs	Retention (days) <small>①</small>
<input type="checkbox"/> AllMetrics	
metric	
<input checked="" type="checkbox"/> AllMetrics	Retention (days) <small>①</small>

f. Sélectionnez **Enregistrer**.

2. Une fois la journalisation activée, utilisez votre service de recherche pour commencer à générer des journaux d'activité et des mesures. Cela prend jusqu'à une heure avant que les conteneurs n'apparaissent dans le Stockage Blob avec ces journaux. Vous verrez un conteneur **insights-logs-operationlogs** pour les journaux de trafic de recherche et un conteneur **insights-metrics-pt1m** pour les métriques.
3. Recherchez l'application Recherche cognitive Azure Power BI dans le [marketplace d'applications Power BI](#) et installez-la dans un nouvel espace de travail ou un espace de travail existant. L'application s'appelle **Recherche cognitive Azure : Analyser les journaux et les métriques**.
4. Après avoir installé l'application, sélectionnez-la à partir de votre liste d'applications dans Power BI.



5. Sélectionnez **Se connecter** pour connecter vos données.

## Get started with your new app

Explore your app with sample data, go to the workspace to customize as needed and share with your organization, or connect your data to get up and running.

### Connect your data

Connect to a data source to view your new app with your own data.



[Connect](#)

### Explore with sample data

Open your new app to start exploring with sample data.



[Explore app](#)

### Customize and share

Your app comes with a workspace, so you can customize and share it, just like an app you built yourself.



[Edit workspace](#)

[Don't show this again](#)

6. Entrez le nom du compte de stockage qui contient vos journaux et métriques. Par défaut, l'application examine les 10 derniers jours de données, mais cette valeur peut être modifiée avec le paramètre **Jours**.

### Connect to Azure Cognitive Search

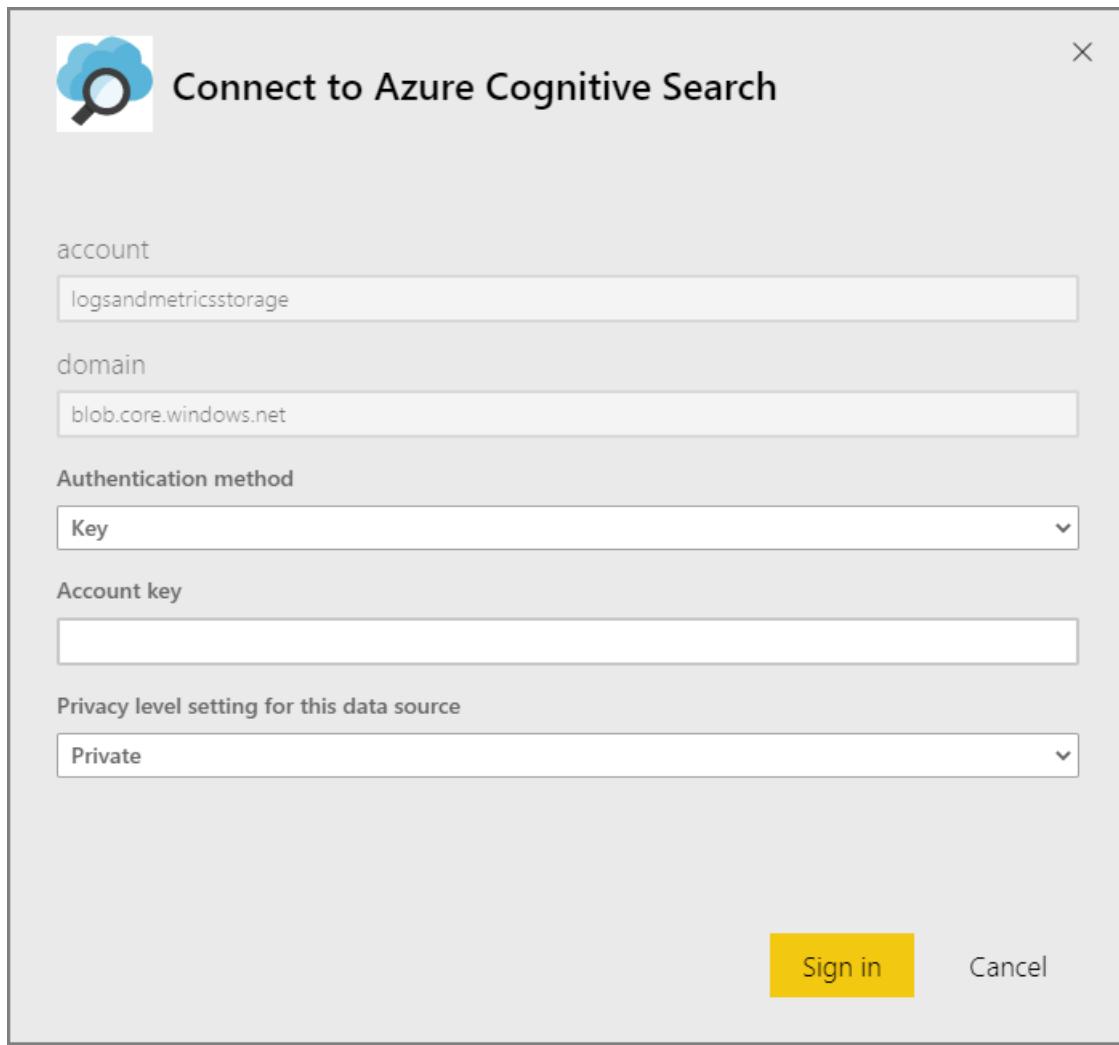
Before connecting to your data, you must update the required parameters (\*).

**StorageAccount**  
The Azure Storage account name you use for Azure Cognitive Search Logs and Metrics. More information on how to enable logging can be found here: <https://docs.microsoft.com/azure/search/search-monitor-usage>

**Days**  
Number of days of data to query.

[Next](#) [Cancel](#)

7. Sélectionnez **Clé** comme méthode d'authentification et indiquez votre clé de compte de stockage. Sélectionnez **Privé** comme niveau de confidentialité. Cliquez sur **Se connecter** pour commencer le processus de chargement.



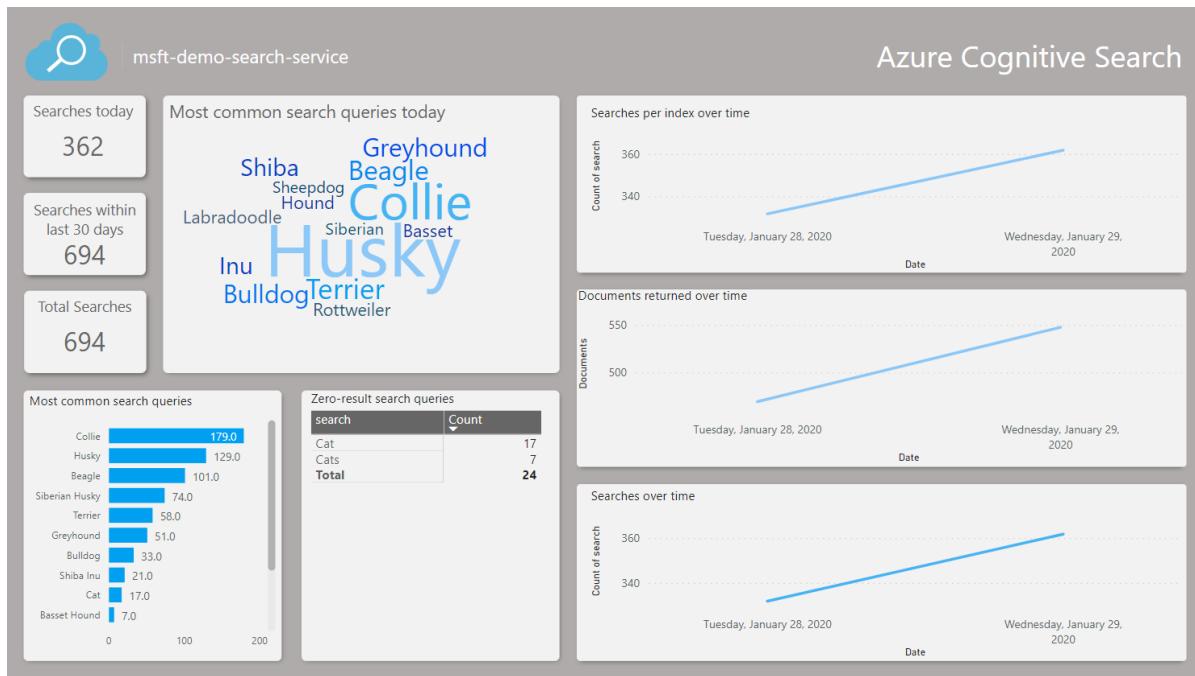
8. Attendez que les données soient actualisées. Cela peut prendre un certain temps en fonction de la quantité de données dont vous disposez. Vous pouvez voir si les données sont toujours en cours d'actualisation d'après l'indicateur ci-dessous.

Name	Type	Sensitivity (preview)	Owner	Refreshed	Endorsement	Include in app
Azure Cognitive Search Dataset	Dataset	—	Azure Cognitive Sear...	2/7/20, 4:01:01 PM	—	
Azure Cognitive Search Report	Report	—	Azure Cognitive Sear...	2/7/20, 4:01:01 PM	—	Yes

9. Une fois l'actualisation des données terminée, sélectionnez **Rapport Recherche cognitive Azure** pour afficher le rapport.

Name	Type	Sensitivity (preview)	Owner	Refreshed	Endorsement	Include in app
Azure Cognitive Search Dataset	Dataset	—	Azure Cognitive Sear...	2/7/20, 4:01:01 PM	—	
Azure Cognitive Search Report	Report	—	Azure Cognitive Sear...	2/7/20, 4:01:01 PM	—	Yes

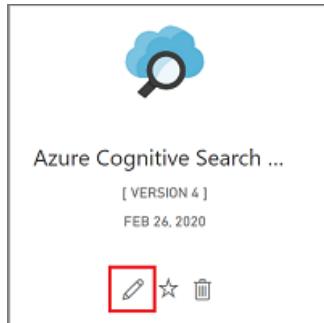
10. Veillez à actualiser la page avant d'avoir ouvert le rapport afin qu'il s'ouvre avec vos données.



## Comment modifier les paramètres de l'application

Si vous souhaitez visualiser des données à partir d'un autre compte de stockage ou modifier le nombre de jours de données à interroger, suivez les étapes ci-dessous pour modifier les paramètres **Jours** et **Compte de stockage**.

1. Accédez à vos applications Power BI, recherchez votre application Recherche cognitive Azure et sélectionnez le bouton **Modifier l'application** pour afficher l'espace de travail.



2. Sélectionnez **Paramètres** parmi les options du jeu de données.

Name	Type	Sensitivity (preview)
 Azure Cognitive Search Dataset	Dataset	—
 Azure Cognitive Search Report		

A context menu is open for the "Azure Cognitive Search Report" item. The menu items are:

- Analyze in Excel
- Create report
- Delete
- Get quick insights
- Refresh now
- Rename
- Schedule refresh
- Settings** (highlighted with a red box)
- Download .pbix
- Manage permissions
- View related

3. Dans l'onglet Jeux de données, modifiez les valeurs des paramètres et sélectionnez **Appliquer**. En cas de problème avec la connexion, mettez à jour les informations d'identification de la source de données sur la même page.
4. Revenez à l'espace de travail et sélectionnez **Actualiser** parmi les options du jeu de données.

Name	Type	Sensitivity (preview)
 Azure Cognitive Search Dataset	Dataset	—
 Azure Cognitive Search Report		

A context menu is open for the "Azure Cognitive Search Report" item. The menu items are:

- Analyze in Excel
- Create report
- Delete
- Get quick insights
- Refresh now** (highlighted with a red box)
- Rename
- Schedule refresh
- Settings
- Download .pbix
- Manage permissions
- View related

5. Ouvrez le rapport pour afficher les données mises à jour. Vous devrez peut-être actualiser également le rapport pour afficher les données les plus récentes.

# Dépannage

Si vous constatez que vous ne pouvez pas voir vos données, suivez ces étapes de résolution de problèmes :

1. Ouvrez le rapport et actualisez la page pour vous assurer de disposer des données les plus récentes. Une option du rapport permet d'actualiser les données. Sélectionnez-la pour récupérer les données les plus récentes.
2. Vérifiez que le nom du compte de stockage et la clé d'accès que vous avez fournis sont corrects. Le nom du compte de stockage doit correspondre au compte configuré avec les journaux de votre service de recherche.
3. Vérifiez que votre compte de stockage contient les conteneurs **insights-logs-operationlogs** et **insights-metrics-pt1m** et que chaque conteneur contient des données. Les journaux et les métriques se trouvent dans deux ou trois couches de dossiers.
4. Vérifiez si le jeu de données est toujours en cours d'actualisation. L'indicateur d'état d'actualisation est indiqué à l'étape 8 ci-dessus. S'il est toujours en cours d'actualisation, patientez jusqu'à la fin de l'actualisation pour ouvrir et actualiser le rapport.

## Étapes suivantes

[En savoir plus sur Recherche cognitive Azure](#)

[Qu'est-ce que Power BI ?](#)

[Concepts de base pour les concepteurs du service Power BI](#)

# Surveiller les demandes de requête dans Recherche cognitive Azure

04/10/2020 • 17 minutes to read • [Edit Online](#)

Cet article explique comment mesurer les performances et le volume des requêtes à l'aide des métriques et de la journalisation des ressources. Il explique également comment collecter les termes d'entrée utilisés dans les requêtes - informations nécessaires à l'évaluation de l'utilité et de l'efficacité de votre corpus de recherche.

Les données historiques qui alimentent les métriques sont conservées pendant 30 jours. Pour une conservation plus longue, ou pour générer des rapports sur les données opérationnelles et les chaînes de requêtes, activez un [paramètre de diagnostic](#) spécifiant l'option de stockage relative aux métriques et événements consignés persistants.

Pour optimiser l'intégrité de la mesure des données, appliquez notamment les conditions suivantes :

- Utilisez un service facturable (service créé au niveau De base ou Standard). Le service gratuit est partagé par différents abonnés, ce qui introduit une certaine volatilité à mesure que les charges se déplacent.
- Si possible, utilisez un réplica et une partition uniques afin de créer un environnement autonome et isolé. Si vous utilisez différents réplicas, la moyenne des métriques de requête est calculée sur différents nœuds, ce qui peut nuire à la précision des résultats. De même, si vous utilisez différentes partitions, les données sont éparpillées, avec le risque que certaines partitions présentent des données différentes si une indexation est également en cours. Lors du paramétrage des performances des requêtes, un nœud et une partition uniques offrent un environnement plus stable pour les tests.

## TIP

Grâce à un code côté client supplémentaire et à Application Insights, vous pouvez également capturer des données de clics pour bénéficier d'insights plus approfondis sur ce qui suscite l'intérêt des utilisateurs de votre application. Pour plus d'informations, consultez [Analytique du trafic des recherches](#).

## Volume de requêtes (RPS)

Le volume est mesuré en **requêtes de recherche par seconde** (RPS), une métrique intégrée qui peut être exprimée en tant que moyenne, nombre, minimum ou maximum de requêtes exécutées sur une période d'une minute. Pour les métriques, des intervalles d'une minute (TimeGrain = "PT1M") sont fixés dans le système.

Les requêtes sont souvent exécutées en quelques millisecondes, par conséquent seules les requêtes mesurées en secondes apparaissent dans les métriques.

TYPE D'AGRÉGATION	DESCRIPTION
Average	Sur une période d'une minute, nombre moyen de secondes nécessaires à l'exécution de la requête.
Count	Nombre de métriques émises dans le journal pendant l'intervalle d'une minute.
Maximale	Nombre maximum de requêtes de recherche par seconde enregistrées en une minute.

TYPE D'AGRÉGATION	DESCRIPTION
Minimum	Nombre minimum de requêtes de recherche par seconde enregistrées en une minute.
SUM	Somme de toutes les requêtes exécutées pendant l'intervalle d'une minute.

Par exemple, sur une minute, le schéma peut être le suivant : une seconde de charge élevée, qui représente votre valeur SearchQueriesPerSecond maximale, puis 58 secondes de charge moyenne, et enfin une seconde avec une seule requête, qui représente la valeur minimale.

Autre exemple : si un nœud émet 100 métriques, sachant que la valeur de chaque métrique est égale à 40, alors "Count" = 100, "Sum" = 4000, "Average" = 40 et "Max" = 40.

## Performances des requêtes

À l'échelle du service, les performances des requêtes sont mesurées en termes de latence de recherche (durée d'exécution d'une requête) et de requêtes limitées qui ont été abandonnées en raison d'un conflit de ressources.

### Latence de recherche

TYPE D'AGRÉGATION	LATENCE
Average	Durée moyenne de la requête en millisecondes.
Count	Nombre de métriques émises dans le journal pendant l'intervalle d'une minute.
Maximale	Requête la plus longue de l'échantillon.
Minimum	Requête la plus courte de l'échantillon.
Total	Durée d'exécution totale de toutes les requêtes de l'échantillon, exécutées dans l'intervalle (une minute).

Penchons-nous sur l'exemple suivant de métriques de **latence de recherche** : 86 requêtes ont été échantillonnées, avec une durée moyenne de 23,26 millisecondes. Un minimum de 0 indique que certaines requêtes ont été abandonnées. La requête la plus longue s'est exécutée en 1 000 millisecondes. La durée d'exécution totale a été de 2 secondes.



### Requêtes limitées

Les requêtes limitées font référence aux requêtes qui sont abandonnées au lieu d'être traitées. Dans la plupart des cas, la limitation fait partie intégrante de l'exécution du service. Cela ne traduit pas nécessairement un problème.

La limitation intervient lorsque le nombre de demandes en cours de traitement dépasse les ressources disponibles. Vous pouvez constater une augmentation du nombre de demandes limitées lorsqu'un réplica est retiré de la rotation ou pendant l'indexation. Les demandes de requête et d'indexation sont gérées par le même ensemble de ressources.

Le service détermine s'il faut abandonner les demandes en fonction de la consommation des ressources. Le

pourcentage de ressources consommées sur la mémoire, le processeur et les E/S disque est calculé en moyenne sur une période donnée. Si ce pourcentage dépasse un certain seuil, toutes les demandes adressées à l'index sont limitées jusqu'à ce que le volume de demandes baisse.

Selon votre client, une demande limitée peut être signalée comme suit :

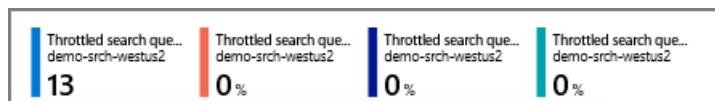
- Un service renvoie une erreur « Trop grand nombre de requêtes. Veuillez réessayer plus tard. »
- Un code d'erreur 503 est renvoyé pour indiquer que le service est indisponible.
- Si vous utilisez le portail (par exemple, l'Explorateur de recherche), la requête est abandonnée en mode silencieux et vous devez à nouveau cliquer sur Rechercher.

Pour confirmer les requêtes limitées, utilisez la métrique **Requêtes de recherche limitées**. Vous pouvez explorer les métriques sur le portail ou créer une métrique d'alerte comme décrit dans cet article. Pour les requêtes qui ont été abandonnées au cours de l'intervalle d'échantillonnage, utilisez *Total* afin d'obtenir le pourcentage de requêtes qui n'ont pas été exécutées.

TYPE D'AGRÉGATION	LIMITATION
Average	Pourcentage de requêtes abandonnées pendant l'intervalle.
Count	Nombre de métriques émises dans le journal pendant l'intervalle d'une minute.
Maximale	Pourcentage de requêtes abandonnées pendant l'intervalle.
Minimum	Pourcentage de requêtes abandonnées pendant l'intervalle.
Total	Pourcentage de requêtes abandonnées pendant l'intervalle.

Pour **Pourcentage de requêtes de recherche limitées**, les valeurs minimales, maximales, moyennes et totales seront identiques : il s'agit du pourcentage de requêtes de recherche qui ont été limitées, en fonction du nombre total de requêtes de recherche pendant une minute.

Dans la capture d'écran suivante, la première valeur correspond au nombre de métriques envoyées au journal. Les agrégations supplémentaires, qui apparaissent en haut ou en pointant sur la métrique, incluent la moyenne, le maximum et le total. Dans cet exemple, aucune demande n'a été abandonnée.



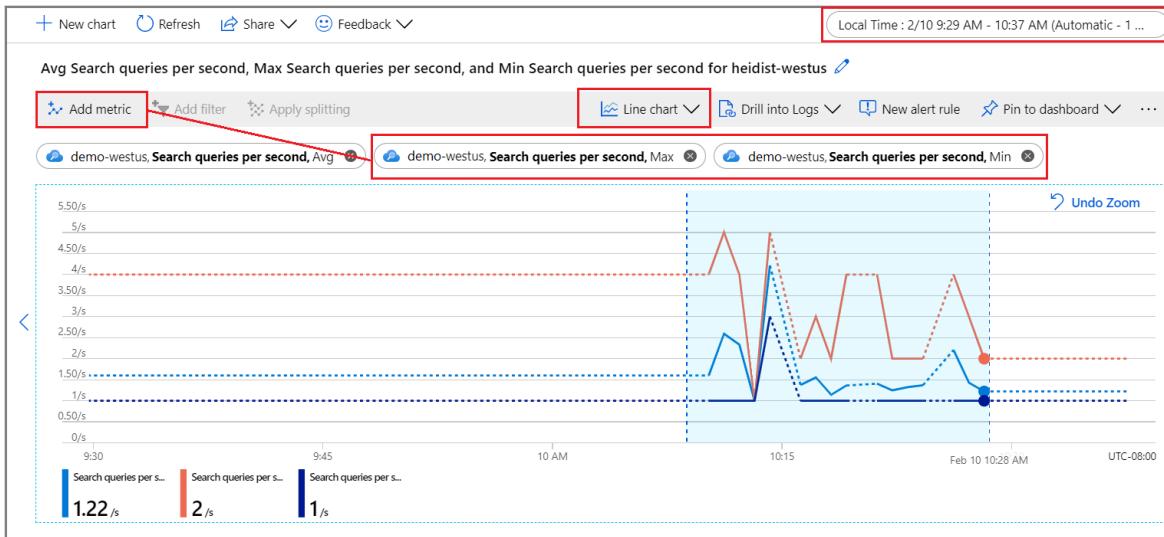
## Explorer les métriques sur le portail

Pour un aperçu rapide des valeurs actuelles, l'onglet **Surveillance** de la page de présentation du service affiche trois mesures (**Latence de recherche**, **Requêtes de recherche par seconde (par unité de recherche)**, **Pourcentage de requêtes de recherche limitées**) sur des intervalles fixes mesurés en heures, jours et semaines, avec la possibilité de modifier le type d'agrégation.

Pour une exploration plus approfondie, ouvrez Metrics Explorer à partir du menu **Surveillance**. Vous pourrez ainsi ajouter des données, effectuer un zoom sur celles-ci et les visualiser afin d'explorer les tendances ou les anomalies. Pour en savoir plus sur Metrics Explorer, suivez ce [tutoriel consacré à la création d'un graphique de métriques](#).

1. Dans la section Surveillance, sélectionnez **Métriques** pour ouvrir Metrics Explorer en veillant à ce que l'étendue soit définie en fonction de votre service de recherche.

2. Sous Métriques, choisissez-en une dans la liste déroulante et passez en revue la liste des agrégations disponibles pour sélectionner le type de votre choix. L'agrégation définit la manière dont les valeurs collectées seront échantillonnées sur chaque intervalle de temps.



3. Dans le coin supérieur droit, définissez l'intervalle de temps.
4. Choisissez une visualisation. La visualisation par défaut est un graphique en courbes.
5. Ajoutez des agrégations supplémentaires en choisissant **Ajouter des métriques** et en sélectionnant différentes agrégations.
6. Zoomez sur une zone d'intérêt du graphique en courbes. Placez le pointeur de la souris au début de la zone, cliquez et maintenez le bouton gauche de la souris enfoncé, faites glisser de l'autre côté de la zone, puis relâchez le bouton. Cet intervalle de temps sera agrandi dans le graphique.

## Identifier les chaînes utilisées dans les requêtes

Quand vous activez la journalisation des ressources, le système capture les demandes de requête dans la table **AzureDiagnostics**. Comme prérequis, vous devez avoir déjà activé la [journalisation des ressources](#), en spécifiant un espace de travail d'analytique des journaux ou une autre option de stockage.

1. Dans la section Surveillance, sélectionnez **Journaux** afin d'ouvrir une fenêtre de requête vide dans Log Analytics.
2. Exécutez l'expression suivante pour rechercher des opérations Query.Search. Vous obtiendrez un jeu de résultats, sous forme de tableau, avec le nom de l'opération, la chaîne de requête, l'index interrogé et le nombre de documents trouvés. Les deux dernières instructions excluent les chaînes de requêtes correspondant à une recherche vide ou non spécifiée, sur un exemple d'index, ce qui réduit le bruit dans vos résultats.

```
AzureDiagnostics
| project OperationName, Query_s, IndexName_s, Documents_d
| where OperationName == "Query.Search"
| where Query_s != "?api-version=2020-06-30&search=*"
| where IndexName_s != "realestate-us-sample-index"
```

3. Vous pouvez également définir un filtre de colonne sur *Query\_s* pour effectuer une recherche sur une syntaxe ou une chaîne spécifique. Par exemple, vous pouvez appliquer le filtre suivant : *est égal à ?api-version=2020-06-30&search=\*&%24filter=HostName* ).

Cette technique fonctionne pour une investigation ad hoc, mais la création d'un rapport vous permettra de regrouper les chaînes de requêtes et de les présenter dans un format plus propice à l'analyse.

## Identifier les requêtes longues

Ajoutez une colonne de durée pour obtenir les valeurs de toutes les requêtes, et pas seulement celles récupérées en tant que métriques. Le tri de ces données vous indique quelles requêtes prennent le plus de temps.

1. Dans la section Surveillance, sélectionnez **Journaux** pour rechercher des informations de journal.
2. Exécutez la requête suivante pour renvoyer des requêtes, triées par durée en millisecondes. Les requêtes longues figurent en haut.

```
AzureDiagnostics
| project OperationName, resultSignature_d, DurationMs, Query_s, Documents_d, IndexName_s
| where OperationName == "Query.Search"
| sort by DurationMs
```

DurationMs	Query_s	IndexName_s
636	?api-version=2019-05-06&search=restaurant&%24count=true&%24select=HotelId...	1 hotels-quickstart
357	?api-version=2019-05-06&search=*%24count=true	14 demoindex
181	?api-version=2019-05-06&search=*	14 demoindex
154	?api-version=2019-05-06-Preview&search=*%24count=true	14 demoindex
148	?api-version=2019-05-06&search=*%24count=true&%24select=HotelId%2C%	4 hotels-quickstart
139	?api-version=2019-05-06&search=*	19 hotel-reviews-idx

## Créer une alerte de métrique

Une alerte de métrique fixe le seuil à partir duquel vous recevez une notification ou déclenchez une action corrective définie à l'avance.

Dans un service de recherche, il est fréquent de créer une alerte de métrique pour la latence de recherche et les requêtes limitées. Si vous savez quand les requêtes sont abandonnées, vous pouvez rechercher des solutions pour réduire la charge ou augmenter la capacité. Par exemple, si les requêtes limitées augmentent pendant l'indexation, vous pouvez reporter celle-ci jusqu'à ce que l'activité de requête diminue.

Lorsque vous repousserez les limites d'une configuration réplica-partition particulière, il convient également de définir des alertes pour les seuils de volumes de requêtes (RPS).

1. Dans la section Surveillance, sélectionnez **Alertes**, puis cliquez sur **+ Nouvelle règle d'alerte**. Veillez à ce que votre service de recherche soit sélectionné comme ressource.
2. Sous Condition, cliquez sur **Ajouter**.
3. Configurez la logique du signal. Pour le type de signal, choisissez **métriques**, puis sélectionnez le signal.
4. Après avoir sélectionné le signal, vous pouvez utiliser un graphique afin de visualiser les données historiques et prendre une décision éclairée sur le mode de configuration des conditions.
5. Faites ensuite défiler jusqu'à Logique d'alerte. Pour la preuve de concept, vous pouvez spécifier une valeur artificiellement faible à des fins de test.

**Alert logic**

**Threshold** ⓘ

Static Dynamic

**Operator** ⓘ Greater than

**Aggregation type** \* ⓘ Average

**Threshold value** \* ⓘ 0 count/second

**Condition preview**

Whenever the search queries per second is greaterthan 0 count/second

**Evaluated based on**

**Aggregation granularity (Period)** \* ⓘ 5 minutes

**Frequency of evaluation** ⓘ Every 1 Minute

6. Ensuite, spécifiez ou créez un groupe d'actions. Il s'agit de la réponse à appeler lorsque le seuil est atteint. Il peut s'agir d'une notification Push ou d'une réponse automatisée.
7. Enfin, spécifiez les détails de l'alerte. Nommez et décrivez l'alerte, attribuez une valeur de gravité et spécifiez si la règle doit être activée ou désactivée.

**ALERT DETAILS**

**Alert rule name** \* ⓘ Search latency exceeds 10 seconds

**Description**

Specify alert description here...

**Severity** \* ⓘ Sev 3

**Enable rule upon creation**

Yes No

Si vous avez spécifié une notification par e-mail, vous recevrez un e-mail de « Microsoft Azure » dont l'objet sera : « Azure : Gravité activée : 3 <your rule name> ».

## Étapes suivantes

Si ce n'est déjà fait, passez en revue les principes de base de la surveillance des services de recherche pour en savoir plus sur les différentes capacités de surveillance disponibles.

[Surveiller les opérations et l'activité dans Recherche cognitive Azure](#)

# Collecter les données de télémétrie pour l'analyse du trafic de recherche

04/10/2020 • 16 minutes to read • [Edit Online](#)

L'analyse du trafic de recherche est un modèle pour la collecte des données de télémétrie concernant les interactions de l'utilisateur avec votre application Recherche cognitive Azure, comme les événements de clic initiés par l'utilisateur et les saisies au clavier. À l'aide de ces informations, vous pouvez déterminer l'efficacité de votre solution de recherche, en vous intéressant notamment aux termes de recherche populaires, aux taux de clics et aux entrées de requête qui ne produisent aucun résultat.

Ce modèle dépend d'[Application Insights](#) (fonctionnalité d'[Azure Monitor](#)) pour collecter les données utilisateur. Vous devrez également ajouter l'instrumentation à votre code client, comme le décrit cet article. Enfin, vous aurez besoin d'un mécanisme de création de rapports pour analyser les données. Nous vous recommandons Power BI, mais vous pouvez utiliser le tableau de bord d'application n'importe quel autre outil qui se connecte à Application Insights.

## NOTE

Le modèle décrit dans cet article est destiné aux scénarios avancés et aux données parcours générées par le code que vous ajoutez à votre client. En revanche, les journaux de service sont faciles à configurer, fournissent une variété de métriques et peuvent être gérés dans le portail sans code requis. L'activation de la journalisation est recommandée pour tous les scénarios. Pour plus d'informations, consultez [Collecter et analyser les données de journal](#).

## Identifier les données de recherche pertinentes

Pour obtenir des mesures utiles pour l'analyse du trafic de recherche, il est nécessaire d'enregistrer certains signaux auprès des utilisateurs de votre application de recherche. Ces signaux indiquent le contenu qui intéresse les utilisateurs et qu'ils estiment pertinent. Pour l'analyse du trafic de recherche, il s'agit des éléments suivants :

- Événements de recherche générés par l'utilisateur : Ce signal se concentre uniquement sur les requêtes de recherche lancées par un utilisateur. Les requêtes de recherche utilisées pour remplir des facettes, du contenu supplémentaire ou des informations internes ne sont pas importantes ; elles ont également tendance à biaiser vos résultats.
- Événements de clic générés par l'utilisateur : Sur une page de résultats de recherche, un événement de clic signifie généralement qu'un document est un résultat pertinent pour une requête de recherche spécifique.

En liant les événements de recherche et de clic avec un ID de corrélation, vous aurez une meilleure compréhension de la façon dont la fonctionnalité de recherche de votre application fonctionne.

## Ajouter la fonctionnalité Analytique du trafic des recherches

Dans la page du [portail](#) de votre service Recherche cognitive Azure, la page Analytique du trafic des recherches contient un aide-mémoire pour suivre ce modèle de télémétrie. À partir de cette page, vous pouvez sélectionner ou créer une ressource Application Insights, obtenir la clé d'instrumentation, copier des extraits de code que vous pouvez adapter à votre solution et télécharger un rapport Power BI qui est créé sur le schéma reflété dans le modèle.

The screenshot shows the Azure portal interface for configuring Application Insights. On the left, a sidebar lists various monitoring and support options. The main area is titled 'Search traffic analytics' and contains the following steps:

- 1. Select the Application Insights resource you will use**: A section where users can choose to 'Use existing' or 'Create new'. It includes fields for 'Resource name', 'Subscription ID', 'Instrumentation Key', and 'Resource group'.
- 2. Copy these snippets into your application code to add search telemetry**: A code editor showing snippets for 'Telemetry client' in C# and JavaScript. The C# snippet includes:
 

```
1 request.setRequestHeader("x-ms-azs-return-searchid", "true");
2 request.setRequestHeader("Access-Control-Expose-Headers", "x-ms-azs-searchid");
3 var searchId = request.getResponseHeader('x-ms-azs-searchid');
```
- 3. Monitor your search metrics with Power BI**: A section with a 'Download PowerBI report' button and a 'Get PowerBI desktop' link.

## 1 - Configurer Application Insights

Sélectionnez une ressource Application Insights ou [en créer une](#) si vous n'en n'avez pas. Si vous utilisez la page Analyse du trafic de recherche, vous pouvez copier la clé d'instrumentation dont votre application a besoin pour se connecter à Application Insights.

Une fois que vous disposez d'une ressource Application Insights, vous pouvez suivre les [instructions relatives aux plateformes et langages pris en charge](#) pour inscrire votre application. L'inscription consiste simplement à ajouter la clé d'instrumentation d'Application Insights à votre code, ce qui configure l'association. Vous pouvez trouver la clé dans le portail ou à partir de la page Analyse du trafic de recherche lorsque vous sélectionnez une ressource existante.

Un raccourci qui fonctionne pour certains types de projets Visual Studio est reflété dans les étapes suivantes. Il crée une ressource et inscrit votre application en quelques clics.

1. Pour Visual Studio et le développement ASP.NET, ouvrez votre solution et sélectionnez **Projet > Ajouter Application Insights Telemetry**.
2. Cliquez sur **Prise en main**.
3. Inscrivez votre application en fournissant un compte Microsoft, un abonnement Azure et une ressource Application Insights (une nouvelle ressource par défaut). Cliquez sur **S'inscrire**.

À ce stade, votre application est configurée pour l'analyse des applications, ce qui signifie que tous les chargements de page sont suivis avec les métriques par défaut. Pour plus d'informations sur les étapes précédentes, consultez [Activer la télémétrie Application Insights côté serveur](#).

## 2 - Ajouter la fonctionnalité d'instrumentation

Cette étape consiste à instrumenter votre propre application de recherche, à l'aide de la ressource Application Insights que vous avez créée à l'étape précédente. Il y a quatre étapes pour ce processus, en commençant par la création d'un client de télémétrie.

## Étape 1 : Créer un client de télémétrie

Créez un objet qui envoie des événements à Application Insights. Vous pouvez ajouter l'instrumentation à votre code d'application côté serveur ou au code côté client s'exécutant dans un navigateur, exprimé ici en variantes C# et JavaScript (pour les autres langages, consultez la liste complète des [plateformes et frameworks pris en charge](#)).

Choisissez l'approche qui vous donne la profondeur d'informations souhaitée.

La télémétrie côté serveur capture les métriques au niveau de la couche application, par exemple dans les applications qui s'exécutent en tant que service web dans le cloud, ou en tant qu'application locale sur un réseau d'entreprise. La télémétrie côté serveur capture les événements de recherche et de clic, la position d'un document dans les résultats et les informations sur les requêtes, mais votre collecte de données est limitée aux informations disponibles au niveau de cette couche.

Sur le client, vous pouvez avoir du code supplémentaire qui manipule les entrées de requête, ajoute la navigation ou inclut le contexte (par exemple, les requêtes lancées à partir d'une page d'accueil et celles lancées depuis une page de produit). Si cela décrit votre solution, vous pouvez opter pour l'instrumentation côté client afin que vos données de télémétrie reflètent ces détails supplémentaires. La façon dont ces détails supplémentaires sont collectés dépasse le cadre de ce modèle, mais vous pouvez consulter [Application Insights pour les pages web](#) pour plus d'informations.

### Utiliser C#

Pour C#, **InstrumentationKey** se trouve dans la configuration de votre application, par exemple appsettings.json si vous avez un projet ASP.NET. Reportez-vous aux instructions d'enregistrement si vous avez des doutes sur l'emplacement de la clé.

```
private static TelemetryClient _telemetryClient;

// Add a constructor that accepts a telemetry client:
public HomeController(TelemetryClient telemetry)
{
    _telemetryClient = telemetry;
}
```

### Utiliser JavaScript

```
<script type="text/javascript">var appInsights=window.appInsights||function(config){function r(config)
{t[config]=function(){var i=arguments;t.queue.push(function(){t[config].apply(t,i)})}}var t=
{config:config},u=document,e=window,o="script",s=u.createElement(o),i,f;s.src=config.url||"/az416426.vo.msecnd.net/scripts/a/ai.0.js";u.getElementsByTagName(o)[0].parentNode.appendChild(s);try{t.cookie=u.cookie}catch(h){}
}for(t.queue=[],i=
["Event","Exception","Metric","PageView","Trace","Dependency"],i.length;)r("track"+i.pop());return
r("setAuthenticatedUserContext"),r("clearAuthenticatedUserContext"),config.disableExceptionTracking||
(i="onerror",r("_"+i),f=e[i],e[i]=function(config,r,u,e,o){var s=f&&f(config,r,u,e,o);return s!==!0&&t["_"+i]
(config,r,u,e,o),s}),t}
({
instrumentationKey: "<YOUR INSTRUMENTATION KEY>"
});
window.appInsights=appInsights;
</script>
```

## Étape 2 : Demander un ID de recherche pour la corrélation

Pour mettre en corrélation les requêtes de recherche avec les clics, il est nécessaire de disposer d'un ID de corrélation qui lie ces deux événements distincts. La Recherche cognitive Azure vous fournit un ID de recherche avec un en-tête HTTP.

Le fait de disposer de l'ID de recherche permet de corrérer les métriques émises par la Recherche cognitive Azure pour la requête elle-même avec les métriques personnalisées que vous consignez dans Application Insights.

## Utiliser C#

```
// This sample uses the .NET SDK https://www.nuget.org/packages/Microsoft.Azure.Search

var client = new SearchIndexClient(<SearchServiceName>, <IndexName>, new SearchCredentials(<QueryKey>)

// Use HTTP headers so that you can get the search ID from the response
var headers = new Dictionary<string, List<string>>() { { "x-ms-azs-return-searchid", new List<string>() { "true" } } };
var response = await client.Documents.SearchWithHttpMessagesAsync(searchText: searchText, searchParameters: parameters, customHeaders: headers);
string searchId = string.Empty;
if (response.Response.Headers.TryGetValue("x-ms-azs-searchid", out IEnumerable<string> headerValues))
{
    searchId = headerValues.FirstOrDefault();
}
```

## Utiliser JavaScript (appel à des API REST)

```
request.setRequestHeader("x-ms-azs-return-searchid", "true");
request.setRequestHeader("Access-Control-Expose-Headers", "x-ms-azs-searchid");
var searchId = request.getResponseHeader('x-ms-azs-searchid');
```

### Étape 3 : Consigner les événements de recherche

Chaque fois qu'une requête de recherche est émise par un utilisateur, vous devez la consigner en tant qu'événement de recherche en respectant le schéma suivant sur un événement personnalisé Application Insights. N'oubliez pas de consigner uniquement les requêtes de recherche générées par l'utilisateur.

- **SearchServiceName** : (string) nom du service de recherche
- **SearchId** : (guid) identificateur unique de la requête de recherche (fourni dans la réponse de recherche)
- **IndexName** : (string) index du service de recherche à interroger
- **QueryTerms** : (string) termes de recherche entrés par l'utilisateur
- **ResultCount** : (int) nombre de documents qui ont été retournés (dans la réponse de recherche)
- **ScoringProfile** : (string) nom du profil de scoring utilisé, le cas échéant

#### NOTE

Demandez le nombre de requêtes générées par l'utilisateur en ajoutant \$count=true à votre requête de recherche. Pour en savoir plus, consultez la section relative à la [recherche de documents \(REST\)](#).

## Utiliser C#

```
var properties = new Dictionary<string, string>
{
    {"SearchServiceName", <service name>},
    {"SearchId", <search Id>},
    {"IndexName", <index name>},
    {"QueryTerms", <search terms>},
    {"ResultCount", <results count>},
    {"ScoringProfile", <scoring profile used>}
};
_telemetryClient.TrackEvent("Search", properties);
```

## Utiliser JavaScript

```

appInsights.trackEvent("Search", {
    SearchServiceName: <service name>,
    SearchId: <search id>,
    IndexName: <index name>,
    QueryTerms: <search terms>,
    ResultCount: <results count>,
    ScoringProfile: <scoring profile used>
});

```

#### Étape 4 : Consigner les événements de clic

Chaque fois qu'un utilisateur clique sur un document, vous obtenez un signal qui doit être consigné afin d'analyser la recherche. Utilisez les événements personnalisés d'Application Insights pour consigner ces événements avec le schéma suivant :

- **ServiceName** : (string) nom du service de recherche
- **SearchId** : (guid) identificateur unique de la requête de recherche associée
- **DocId** : (string) identificateur du document
- **Position** : (int) rang du document dans la page des résultats de la recherche

#### NOTE

La position fait référence à la commande cardinale dans votre application. Vous êtes libre de définir ce nombre, tant qu'il reste toujours le même, pour faciliter la comparaison.

#### Utiliser C#

```

var properties = new Dictionary <string, string>
{
    {"SearchServiceName", <service name>},
    {"SearchId", <search id>},
    {"ClickedDocId", <clicked document id>},
    {"Rank", <clicked document position>}
};
 telemetryClient.TrackEvent("Click", properties);

```

#### Utiliser JavaScript

```

appInsights.trackEvent("Click", {
    SearchServiceName: <service name>,
    SearchId: <search id>,
    ClickedDocId: <clicked document id>,
    Rank: <clicked document position>
});

```

## 3 - Analyser dans Power BI

Après avoir instrumenté votre application et vérifié qu'elle est correctement connectée à Application Insights, vous pouvez télécharger un modèle de rapport prédéfini pour analyser les données dans Power BI Desktop. Le rapport contient des graphiques et des tableaux prédéfinis utiles pour analyser les données supplémentaires capturées pour l'analytique du trafic des recherches.

1. Dans le volet de navigation de gauche du tableau de bord de la Recherche cognitive Azure, sous **Paramètres**, cliquez sur **Analytique du trafic des recherches**.
2. Dans la page **Analytique du trafic des recherches**, à l'étape 3, cliquez sur **Obtenir Power BI Desktop**

pour installer Power BI.

### 3. Monitor your search metrics with Power BI

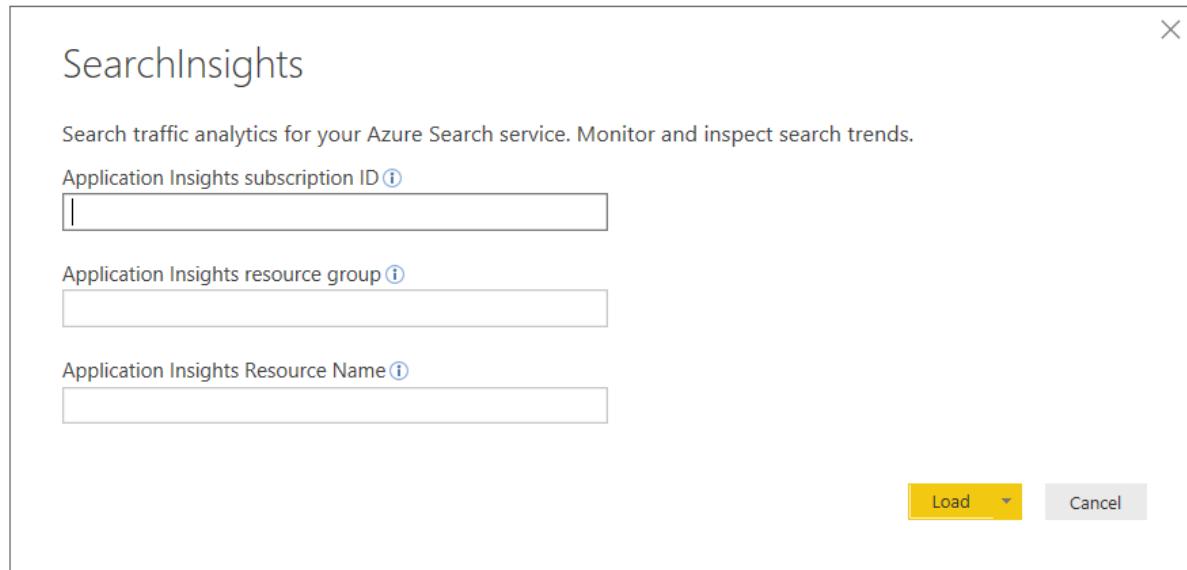
On download, use the Application Insights resource information on this page to connect your report to analytical data captured for your application.

[Download PowerBI report](#)

[Get PowerBI desktop](#)

3. Dans la même page, cliquez sur **Télécharger un rapport Power BI**.

4. Le rapport s'ouvre dans Power BI Desktop et vous invite à vous connecter à Application Insights et à fournir des informations d'identification. Des informations de connexion sont disponibles dans les pages du portail Azure relatives à votre ressource Application Insights. Pour les informations d'identification, indiquez les mêmes nom d'utilisateur et mot de passe que vous utilisez pour vous connecter au portail.



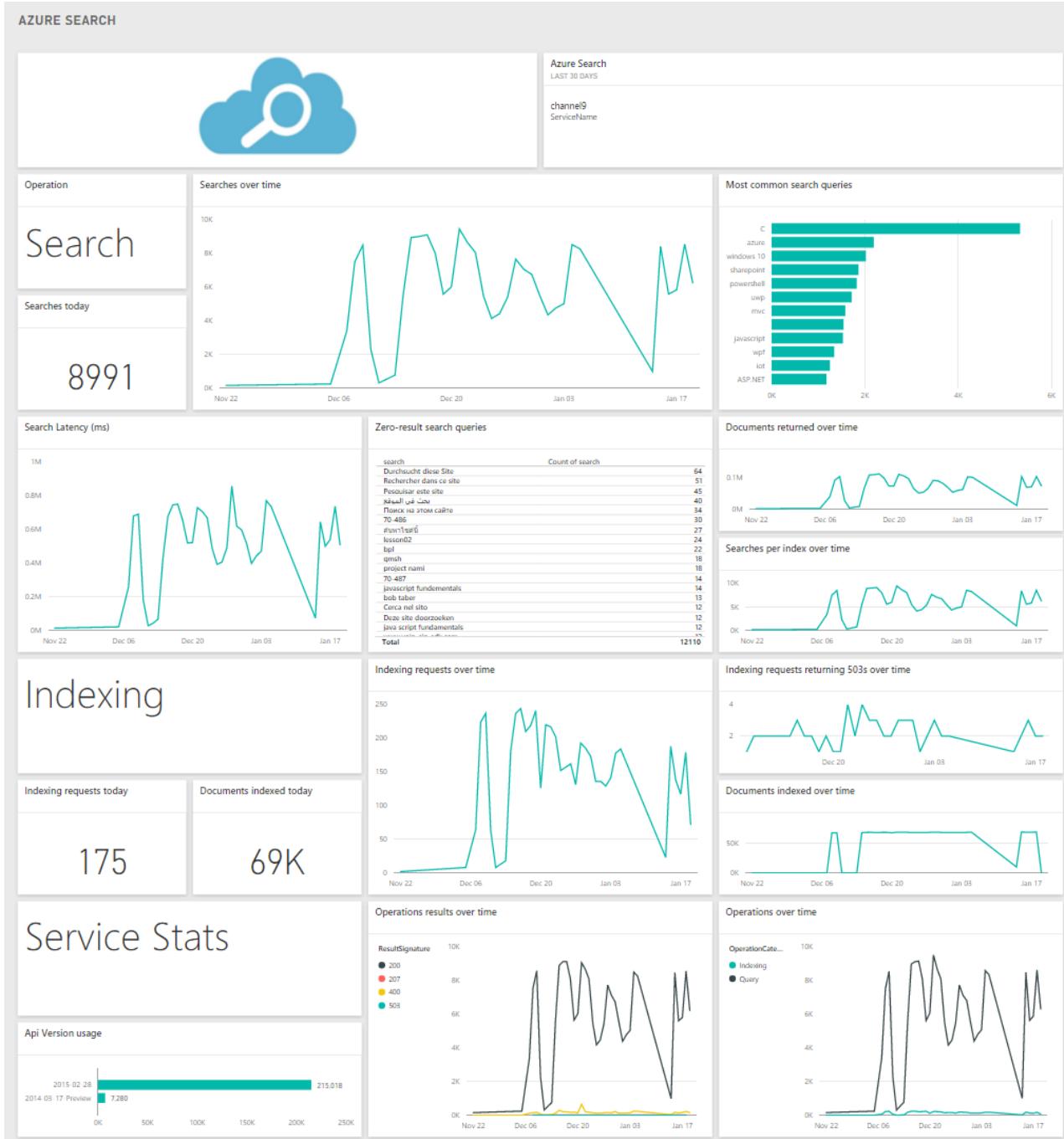
5. Cliquez sur **Charger**.

Ce rapport contient des graphiques et des tableaux qui vous aident à prendre des décisions plus éclairées pour améliorer les performances et la pertinence de vos recherches.

Les métriques incluaient les éléments suivants :

- Volume de recherche et paires terme-document les plus populaires : termes qui amènent l'utilisateur à cliquer sur le même document, classés par clics.
- Recherches sans clic : termes renvoyant aux principales requêtes qui n'enregistrent aucun clic

La capture d'écran suivante montre à quoi peut ressembler un rapport intégré si vous avez utilisé tous les éléments du schéma.



## Étapes suivantes

Instrumentez votre application de recherche pour obtenir des données puissantes et détaillées sur votre service de recherche.

Des informations supplémentaires sur [Application Insights](#) sont disponibles. Vous pouvez également visiter la [page des tarifs](#) pour en savoir plus sur les différents niveaux de service.

En savoir plus sur la création de rapports exceptionnels. Pour plus d'informations, consultez [Bien démarrer avec Power BI Desktop](#).

# Résoudre les problèmes courants des indexeurs dans la Recherche cognitive Azure

04/10/2020 • 9 minutes to read • [Edit Online](#)

Les indexeurs peuvent rencontrer un certain nombre de problèmes lors de l'indexation des données dans la Recherche cognitive Azure. Voici les principales catégories de défaillances :

- [Connexion à une source de données ou à d'autres ressources](#)
- [Traitement de documents](#)
- [Ingestion des documents dans un index](#)

## Erreurs de connexion

### NOTE

Les indexeurs ont une prise en charge limitée pour accéder aux sources de données et à d'autres ressources sécurisées par les mécanismes de sécurité réseau Azure. Actuellement, les indexeurs peuvent accéder uniquement aux sources de données par le biais de mécanismes de restriction de plage d'adresses IP ou de règles NSG correspondantes, le cas échéant. Vous trouverez ci-dessous des détails sur l'accès à chaque source de données prise en charge.

Vous trouverez l'adresse IP de votre service Search en effectuant un test ping de son nom de domaine complet (p. ex., `<your-search-service-name>.search.windows.net`).

Vous pouvez trouver la plage d'adresses IP de `AzureCognitiveSearch` [l'étiquette de service](#) en utilisant des [fichiers JSON téléchargeables](#) ou via [l'API de détection d'étiquettes de service](#). La plage d'adresses IP est mise à jour chaque semaine.

## Configurer les règles de pare-feu

Stockage Azure, CosmosDB et Azure SQL fournissent un pare-feu configurable. Il n'y a pas de message d'erreur spécifique lorsqu'il est activé. En règle générale, les erreurs de pare-feu sont génériques et se présentent comme

`The remote server returned an error: (403) Forbidden` OU

`Credentials provided in the connection string are invalid or have expired`.

Vous pouvez utiliser deux options pour autoriser les indexeurs à accéder à ces ressources dans une telle instance :

- Désactiver le pare-feu en choisissant d'autoriser l'accès de **Tous les réseaux** (si possible).
- Vous pouvez également autoriser l'accès à l'adresse IP de votre service Search et à la plage d'adresses IP de `AzureCognitiveSearch` [l'étiquette de service](#) dans les règles de pare-feu de votre ressource (restriction de plage d'adresses IP).

Pour plus d'informations sur la configuration des restrictions de plage d'adresses IP pour chaque type de source de données, consultez les liens suivants :

- [Stockage Azure](#)
- [Cosmos DB](#)
- [Azure SQL](#)

**Limitation** : Comme indiqué dans la documentation ci-dessus pour le stockage Azure, les restrictions de plage d'adresses IP ne fonctionnent que si votre service Search et votre compte de stockage se trouvent dans des régions différentes.

Azure Functions (qui peut être utilisé comme une [compétence API web personnalisée](#)) prend également en charge les [restrictions d'adresse IP](#). La liste d'adresses IP à configurer correspond à l'adresse IP de votre service Search et à la plage d'adresses IP de `AzureCognitiveSearch` l'étiquette de service.

Pour plus d'informations sur l'accès aux données dans SQL Server sur une machine virtuelle Azure, cliquez [ici](#)

### Configurer les règles du groupe de sécurité réseau (NSG)

Lors de l'accès aux données d'une instance gérée SQL, ou lorsqu'une machine virtuelle Azure est utilisée comme URI de service pour une [compétence API web personnalisée](#), les clients n'ont pas à se préoccuper des adresses IP spécifiques.

Dans ce cas, la machine virtuelle Azure ou l'instance gérée SQL peut être configurée pour résider dans un réseau virtuel. Ensuite, le groupe de sécurité réseau peut être configuré pour filtrer le type de trafic réseau pouvant circuler vers et depuis les interfaces réseau et les sous-réseaux de réseau virtuel.

`AzureCognitiveSearch` L'étiquette de service peut être utilisée directement dans les [règles de groupe de sécurité réseau](#) entrantes sans avoir besoin de rechercher sa plage d'adresses IP.

Pour plus d'informations sur l'accès aux données dans une instance managée SQL, cliquez [ici](#)

### « L'indexation » CosmosDB n'est pas activée

La Recherche cognitive Azure comporte une dépendance implicite vis-à-vis de l'indexation Cosmos DB. Si l'indexation automatique est désactivée dans Cosmos DB, la Recherche cognitive Azure renvoie un état réussi, mais ne parvient pas à indexer le contenu du conteneur. Pour savoir comment vérifier les paramètres et activer l'indexation, voir [Gérer l'indexation dans Azure Cosmos DB](#).

## Erreurs de traitement de documents

### Documents non exploitables ou non pris en charge

L'indexeur d'objets blob [précise quels formats de documents sont pris en charge explicitement](#). Il peut arriver qu'un conteneur de stockage blob contienne des documents non pris en charge ou bien problématiques. Pour ne pas avoir à arrêter votre indexeur sur ces documents, vous pouvez [modifier les options de configuration](#) :

```
PUT https://[service name].search.windows.net/indexers/[indexer name]?api-version=2020-06-30
Content-Type: application/json
api-key: [admin key]

{
    ... other parts of indexer definition
    "parameters" : { "configuration" : { "failOnUnsupportedContentType" : false, "failOnUnprocessableDocument" : false } }
}
```

### Contenu de document manquant

L'indexeur d'objets blob [recherche et extrait du texte dans les objets blob d'un conteneur](#). L'extraction de texte peut poser les problèmes suivants :

- Le document ne contient que des images numérisées. Les objets blob PDF comportant du contenu non textuel, comme des images numérisées (JPG), ne produisent pas de résultats dans un pipeline d'indexation blob standard. Si vous avez du contenu d'image comportant des éléments textuels, vous pouvez utiliser la [recherche cognitive](#) pour rechercher et extraire le texte.
- L'indexeur d'objets blob est configuré pour indexer uniquement les métadonnées. Pour extraire le contenu, il doit être configuré de façon à [extraire à la fois le contenu et les métadonnées](#) :

```
PUT https://[service name].search.windows.net/indexers/[indexer name]?api-version=2020-06-30
Content-Type: application/json
api-key: [admin key]

{
    ...
    ... other parts of indexer definition
    "parameters" : { "configuration" : { "dataToExtract" : "contentAndMetadata" } }
}
```

## Erreurs d'index

### Documents manquants

Les indexeurs recherchent des documents dans une [source de données](#). Parfois, il manque un document qui aurait dû être indexé. Ces erreurs peuvent se produire pour plusieurs raisons :

- Le document n'a pas été indexé. Consultez le portail pour une exécution réussie de l'indexeur.
- Vérifiez votre valeur de [suivi des modifications](#). Si la valeur de la limite supérieure est une date future, tous les documents dont la date est antérieure à celle-ci sont ignorés par l'indexeur. Vous pouvez comprendre l'état de suivi des modifications de votre indexeur à l'aide des champs « initialTrackingState » et « finalTrackingState » dans l'[état de l'indexeur](#).
- Le document a été mis à jour après l'exécution de l'indexeur. Si votre indexeur suit une [planification](#), il s'exécutera à nouveau et trouvera le document.
- La [requête](#) spécifiée dans la source de données exclut le document. Les indexeurs ne peuvent pas indexer de documents qui ne font pas partie de la source de données.
- Des [mappages de champs](#) ou l'[enrichissement de l'IA](#) ont modifié le document, qui n'a pas l'aspect prévu.
- Utilisez [l'API Recherche de document](#) pour trouver votre document.

# Résoudre les erreurs et les avertissements courants de l'indexeur dans la Recherche cognitive Azure

04/10/2020 • 49 minutes to read • [Edit Online](#)

Cet article fournit des informations et des solutions pour les erreurs et avertissements courants que vous pourriez rencontrer pendant l'indexation et l'enrichissement par IA dans Recherche cognitive Azure.

L'indexation s'arrête quand le nombre d'erreurs dépasse la valeur de '[maxFailedItems](#)'.

Si vous souhaitez que les indexeurs ignorent ces erreurs (et sautent les « documents en échec »), envisagez de mettre à jour `maxFailedItems` et `maxFailedItemsPerBatch` comme décrit [ici](#).

## NOTE

Chaque document en échec et la clé de document correspondante (quand elle est disponible) sont présentées comme une erreur dans l'état d'exécution de l'indexeur. Vous pouvez utiliser l'[api index](#) pour charger manuellement les documents à un moment ultérieur si vous avez défini l'indexeur de façon à tolérer les échecs.

Les informations d'erreur de cet article peuvent vous aider à résoudre les erreurs et à poursuivre l'indexation.

L'indexation des avertissements ne s'arrête pas, mais certaines conditions pourraient entraîner des résultats inattendus. Vos données et votre scénario conditionnent les mesures que vous devriez prendre.

À partir de la version d'API [2019-05-06](#), les erreurs et les avertissements de l'indexeur au niveau de l'élément sont structurés de manière à fournir une clarté accrue sur les causes et la marche à suivre. Ils contiennent les propriétés suivantes :

PROPRIÉTÉ	DESCRIPTION	EXEMPLE
key	ID du document concerné par l'erreur ou l'avertissement.	<a href="https://coromsearch.blob.core.windows.net/jfk-1k/docid-32112954.pdf">https://coromsearch.blob.core.windows.net/jfk-1k/docid-32112954.pdf</a>
name	Nom de l'opération décrivant l'emplacement où l'erreur ou l'avertissement s'est produit. Il est généré selon la structure suivante : [catégorie].[sous-catégorie].[ressourceType].[ressourceNom]	DocumentExtraction.azureblob.myBlobContainerName Enrichment.WebApiSkill.mySkillName Projection.SearchIndex.OutputFieldMapping.myOutputFieldName Projection.SearchIndex.MergeOrUpload.myIndexName Projection.KnowledgeStore.Table.myTableName
message	Description de haut niveau de l'erreur ou de l'avertissement.	Impossible d'exécuter la compétence en raison de l'échec de la requête de l'API web.
details	Tous les détails supplémentaires qui peuvent être utiles pour diagnostiquer le problème, tels que la réponse WebApi en cas d'échec de l'exécution d'une compétence personnalisée.	<pre>link-cryptonyms-list - Error processing the request record : System.ArgumentNullException: Value cannot be null. Parameter name: source at System.Linq.Enumerable.All[TSource](IEnumerable&lt;T&gt; source, Func&lt;T, bool&gt; predicate) at Microsoft.CognitiveSearch.WebApiSkills.JfkWebAp...reste de l'arborescence des appels de procédure...</pre>

PROPRIÉTÉ	DESCRIPTION	EXEMPLE
documentationLink	Lien vers la documentation appropriée contenant des informations détaillées pour déboguer et résoudre le problème. Ce lien mène souvent sur cette page, à l'une des sections ci-dessous.	<a href="https://go.microsoft.com/fwlink/?linkid=2106475">https://go.microsoft.com/fwlink/?linkid=2106475</a>

## Erreur : Impossible de lire le document

L'indexeur n'a pas pu lire le document à partir de la source de données. Cela peut se produire si :

MOTIF	DÉTAILS/EXEMPLE	RÉSOLUTION
Types de champs incohérents dans différents documents	« Le type de valeur ne correspond pas au type de colonne. Impossible de stocker '{47.6,-122.1}' dans la colonne des auteurs. Le type attendu est JArray. » « Erreur lors de la conversion du type de données nvarchar en float. » « Échec de la conversion lors de la conversion de la valeur nvarchar "12 mois" en type de données int. » « Une erreur de dépassement arithmétique s'est produite lors de la conversion de l'expression en type de données int. »	Assurez-vous que le type de chaque champ est identique dans les différents documents. Par exemple, si le champ 'startTime' du premier document est un champ DateTime et une chaîne de caractères dans le second document, cette erreur s'affichera.
erreurs provenant du service sous-jacent de la source de données	(de Cosmos DB) {"Errors": ["Request rate is large"]}	Vérifiez l'intégrité de votre instance de stockage. Vous devrez peut-être ajuster votre mise à l'échelle/partitionnement.
problèmes temporaires	Une erreur de niveau transport s'est produite lors de la réception des résultats à partir du serveur. (fournisseur : Fournisseur TCP erreur : 0 - Une connexion existante a été fermée de force par l'hôte distant	Certains problèmes de connectivité inattendus peuvent survenir. Essayez d'exécuter ultérieurement le document dans votre indexeur.

## Erreur : Impossible d'extraire le contenu ou les métadonnées de votre document

L'indexeur avec une source de données Blob n'a pas pu extraire le contenu ou les métadonnées du document (par exemple, un fichier PDF). Cela peut se produire si :

MOTIF	DÉTAILS/EXEMPLE	RÉSOLUTION
l'objet Blob dépasse la limite de taille	Le document affiche une taille de '134217728', ce qui est supérieur à la taille maximale de '150441598' pour l'extraction de document correspondant à votre niveau de service actuel.	<a href="#">Erreurs d'indexation des objets Blob</a>
L'objet Blob a un type de contenu non pris en charge	Le document contient un type de contenu 'image/png' non pris en charge	<a href="#">Erreurs d'indexation des objets Blob</a>
L'objet Blob est chiffré	Le document n'a pas pu être traité car il est peut-être chiffré ou protégé par un mot de passe.	Vous pouvez ignorer le blob grâce aux <a href="#">paramètres de blob</a> .
problèmes temporaires	« Erreur de traitement des objets blob : La demande a été abandonnée : La demande a été annulée. » « Le document a expiré pendant le traitement ».	Certains problèmes de connectivité inattendus peuvent survenir. Essayez d'exécuter ultérieurement le document dans votre indexeur.

## Erreur : Impossible d'analyser le document

L'indexeur a lu le document à partir de la source de données, mais un problème est survenu lors de la conversion du contenu du document au schéma de mappage de champs spécifié. Cela peut se produire si :

MOTIF	DÉTAILS/EXEMPLE	RÉSOLUTION
La clé du document est manquante	La clé du document ne peut pas être manquante ou vide	Vérifiez que tous les documents contiennent des clés de documents valides. La clé de document est déterminée en définissant la propriété « clé » dans le cadre de la <a href="#">définition d'index</a> . Les indexeurs émettent cette erreur quand la propriété marquée en tant que « clé » est introuvable dans un document particulier.
La clé du document n'est pas valide	La clé du document ne peut pas contenir plus de 1024 caractères	Modifiez la clé du document pour répondre aux exigences de validation.
Impossible d'appliquer le mappage de champs à un champ	Impossible d'appliquer la fonction de mappage ' <code>functionName</code> ' au champ ' <code>fieldName</code> '. Le tableau ne peut pas être null. Nom du paramètre : octets	Vérifiez soigneusement les <a href="#">mappages de champs</a> définis sur l'indexeur, puis comparez ces valeurs avec les données du champ spécifié du document en échec. Il peut être nécessaire de modifier les mappages de champs ou les données du document.
Impossible de lire la valeur du champ	Impossible de lire la valeur de la colonne ' <code>fieldName</code> ' à l'index ' <code>fieldIndex</code> '. Une erreur de niveau transport s'est produite lors de la réception des résultats à partir du serveur. (fournisseur : Fournisseur TCP, erreur : 0 - Une connexion existante a été fermée de force par l'hôte distant.)	Ces erreurs sont généralement dues à des problèmes de connectivité inattendus avec le service sous-jacent de la source de données. Essayez d'exécuter ultérieurement le document dans votre indexeur.

## Erreur : Impossible de mapper le champ de sortie « `xyz` » avec l'index de recherche en raison d'un problème de désérialisation lors de l'application de la fonction de mappage « `abc` »

Le mappage de sortie a peut-être échoué, car les données de sortie sont dans un format incorrect pour la fonction de mappage que vous utilisez. Par exemple, l'application de la fonction de mappage Base64Encode sur des données binaires génère cette erreur. Pour résoudre le problème, réexécutez l'indexeur sans spécifier la fonction de mappage ou assurez-vous que la fonction de mappage est compatible avec le type de données du champ de sortie. Pour plus d'informations, consultez [Mappage de champs de sortie](#).

## Erreur : Impossible d'exécuter une compétence

L'indexeur n'a pas pu exécuter une compétence dans l'ensemble de compétences.

MOTIF	DÉTAILS/EXEMPLE	RÉSOLUTION
Problèmes de connectivité temporaires	Une erreur temporaire s'est produite. Veuillez réessayer plus tard.	Certains problèmes de connectivité inattendus peuvent survenir. Essayez d'exécuter ultérieurement le document dans votre indexeur.
Bogue potentiel du produit	Une erreur inattendue s'est produite.	Cela indique une classe de défaillance inconnue et peut signifier qu'il existe un bogue de produit. Pour obtenir de l'aide, créez un <a href="#">ticket de support</a> .

MOTIF	DÉTAILS/EXEMPLE	RÉSOLUTION
Une compétence a rencontré une erreur lors de son exécution	(à partir de Compétence Fusion) Une ou plusieurs valeurs de décalage n'étaient pas valides. Il n'a pas été possible de les analyser. Des éléments ont été insérés à la fin du texte	Utilisez les informations contenues dans le message d'erreur pour résoudre le problème. La résolution de ce type de défaillance nécessite une action.

## Erreur : Impossible d'exécuter la compétence en raison de l'échec de la requête de l'API web

L'exécution de la compétence a échoué parce que l'appel de l'API web a échoué. En règle générale, cette classe de défaillance se produit lorsque des compétences personnalisées sont utilisées. Dans ce cas, vous devez déboguer votre code personnalisé pour résoudre le problème. Si au contraire l'échec provient d'une compétence intégrée, consultez le message d'erreur dans lequel vous trouverez peut-être de l'aide pour résoudre le problème.

Lors du débogage de ce problème, prêtez attention aux éventuels [avertissemens d'entrée de compétence](#) pour cette compétence. Le point de terminaison de votre API web peut échouer, car l'indexeur lui transmet une entrée inattendue.

## Erreur : Impossible d'exécuter la compétence parce que la réponse concernant la compétence de l'API web n'est pas valide

L'exécution de la compétence a échoué parce que l'API web a retourné une réponse non valide. En règle générale, cette classe de défaillance se produit lorsque des compétences personnalisées sont utilisées. Dans ce cas, vous devez déboguer votre code personnalisé pour résoudre le problème. Si au contraire l'échec est dû à une compétence intégrée, créez un [ticket de support](#) pour obtenir de l'aide.

## Erreur : La compétence n'a pas été exécutée dans le délai imparti

Il existe deux cas dans lesquels vous pouvez rencontrer ce message d'erreur, chacun d'entre eux devant être traité différemment. Veuillez suivre les instructions ci-dessous en fonction de la compétence qui a retourné cette erreur.

### Compétences de service cognitives intégrées

Bon nombre des compétences cognitives intégrées, notamment la détection de la langue, la reconnaissance d'entités ou la reconnaissance optique de caractères (OCR), s'appuient sur un point de terminaison Cognitive Service API. Parfois, des problèmes temporaires peuvent survenir avec ces points de terminaison, entraînant l'expiration d'une demande. Il n'existe aucun remède pour ces problèmes temporaires, si ce n'est d'attendre et de réessayer. Pour résoudre le problème, réglez votre indexeur afin de l'[exécuter selon une planification](#). L'indexation planifiée reprend là où elle s'était arrêtée. Si les problèmes temporaires ont été résolus, l'indexation et le traitement des compétences cognitives devraient reprendre à la prochaine exécution planifiée.

Si vous continuez à voir cette erreur sur le même document pour une compétence cognitive intégrée, veuillez remplir un [ticket de support](#) pour obtenir de l'aide, car cela n'est pas prévu.

### Compétences personnalisées

Si vous rencontrez une erreur d'expiration d'une compétence personnalisée que vous avez créée, vous pouvez essayer plusieurs solutions. Tout d'abord, passez en revue votre compétence personnalisée et vérifiez qu'elle ne se retrouve pas bloquée dans une boucle infinie et qu'elle renvoie un résultat constant. Une fois ce point vérifié, déterminez le délai d'exécution de votre compétence. Si vous n'avez pas explicitement défini une valeur `timeout` dans votre définition de compétence personnalisée, la valeur par défaut `timeout` est 30 secondes. Si 30 secondes ne suffisent pas à l'exécution de votre compétence, vous pouvez spécifier une valeur plus élevée `timeout` dans la définition de votre compétence personnalisée. Voici un exemple de définition de compétence personnalisée où le délai d'expiration est fixé à 90 secondes :

```
{
    "@odata.type": "#Microsoft.Skills.Custom.WebApiSkill",
    "uri": "<your custom skill uri>",
    "batchSize": 1,
    "timeout": "PT90S",
    "context": "/document",
    "inputs": [
        {
            "name": "input",
            "source": "/document/content"
        }
    ],
    "outputs": [
        {
            "name": "output",
            "targetName": "output"
        }
    ]
}
```

La valeur maximale que vous pouvez définir pour le paramètre `timeout` est de 230 secondes. Si votre compétence personnalisée ne peut pas s'exécuter de façon constante en 230 secondes, vous pouvez réduire la valeur `batchSize` de votre compétence personnalisée afin qu'elle ait moins de documents à traiter en une seule exécution. Si vous avez déjà réglé votre valeur `batchSize` sur 1, vous devrez réécrire la compétence pour pouvoir l'exécuter en moins de 230 secondes ou la diviser en plusieurs compétences personnalisées de sorte que le délai d'exécution d'une seule compétence personnalisée soit au maximum de 230 secondes. Pour plus d'informations, consultez la [documentation sur les compétences personnalisées](#).

## Erreur : Impossible de « `MergeOrUpload` » | « `Delete` » le document dans l'index de recherche

Le document a été lu et traité, mais l'indexeur n'a pas pu l'ajouter à l'index de recherche. Cela peut se produire si :

MOTIF	DÉTAILS/EXEMPLE	RÉSOLUTION
Un champ contient un terme trop grand	Votre document contient un terme qui dépasse la <a href="#">limite de 32 Ko</a>	Vous pouvez éviter cette restriction en vérifiant que le champ n'est pas configuré comme étant filtrable, à choix multiple ou triable.
Le document est trop volumineux pour être indexé	Un document est plus volumineux que la <a href="#">taille maximale de demande d'API</a>	<a href="#">Indexer les jeux de données volumineux</a>
Le document contient un trop grand nombre d'objets dans la collection	Une collection dans votre document dépasse la <a href="#">limite maximale d'éléments pour toutes collections complexes</a> « Le document avec la clé '1000052' contient '4303' objets dans des collections (tableaux JSON). '3000' objets maximum sont autorisés dans des collections sur l'ensemble du document. Supprimez des objets des collections et réessayez d'indexer le document. »	Nous vous recommandons de réduire la taille de la collection complexe dans le document en dessous de la limite et d'éviter une utilisation intensive du stockage.
Difficultés à se connecter à l'index cible (qui persiste après les nouvelles tentatives), car le service est soumis à une autre charge, par exemple l'interrogation ou l'indexation.	Échec de l'établissement d'une connexion pour mettre à jour l'index. Le service de recherche est soumis à une charge importante.	<a href="#">Effectuer le scale-up de votre service de recherche</a>
Un correctif est appliqué au service de recherche en vue de sa mise à jour ou fait l'objet d'une reconfiguration de topologie.	Échec de l'établissement d'une connexion pour mettre à jour l'index. Le service de recherche est actuellement inopérant/Le service de recherche est en cours de transition.	Configurer le service avec au moins trois répliques pour une disponibilité de 99,9 % selon la <a href="#">documentation SLA</a>

MOTIF	DÉTAILS/EXEMPLE	RÉSOLUTION
Échec dans la ressource de calcul/réseau sous-jacente (rare)	Échec de l'établissement d'une connexion pour mettre à jour l'index. Une erreur inconnue s'est produite.	Configurer les indexeurs pour une <a href="#">exécution selon une planification</a> pour récupérer d'un état d'échec.
Une requête d'indexation envoyée à l'index cible n'a pas reçu d'accusé de réception pendant la période définie en raison de problèmes réseau.	Impossible d'établir la connexion à l'index de recherche en temps opportun.	Configurer les indexeurs pour une <a href="#">exécution selon une planification</a> pour récupérer d'un état d'échec. Essayez également de réduire la <a href="#">taille du lot</a> de l'indexeur si cet état d'erreur persiste.

## Erreur : impossible d'indexer le document parce que certaines données de celui-ci n'étaient pas valides

L'indexeur a lu et traité le document mais, en raison d'une discordance de configuration des champs d'index et des données extraites et traitées par l'indexeur, il n'a pas été possible d'ajouter les données à l'index de recherche. Cela peut se produire si :

MOTIF	DÉTAILS/EXEMPLE
Le type de données du ou des champs extraits par l'indexeur est incompatible avec le modèle de données du champ d'index cible correspondant.	Le champ de données « <i>data</i> » dans le document avec la clé « 888 » a une valeur non valide de type « <i>Edm.String</i> ». Le type attendu était « <i>collection (EDM. String)</i> ».
Échec d'extraction d'une entité JSON à partir d'une valeur de chaîne.	Impossible d'analyser la valeur de type « <i>Edm.String</i> » du champ « <i>data</i> » en tant qu'objet JSON. Erreur : « Après analyse d'une valeur, un caractère inattendu a été rencontré : «. Chemin d'accès « <i>path</i> », ligne 1, position 3162. »
Échec d'extraction d'une collection d'entités JSON d'une valeur de chaîne.	Impossible d'analyser la valeur de type « <i>Edm.String</i> » du champ « <i>data</i> » sous la forme d'un tableau JSON. Erreur : « Après analyse d'une valeur, un caractère inattendu a été rencontré : «. Chemin d'accès « [0] », ligne 1, position 27. »
Un type inconnu a été découvert dans le document source.	Le type inconnu « <i>unknown</i> » ne peut pas être indexé
Une notation incompatible pour les points géographiques a été utilisée dans le document source.	Les littéraux de chaîne WKT POINT ne sont pas pris en charge. Utilisez des littéraux de points GeoJson à la place

Dans tous ces cas, consultez les [types de données pris en charge](#) et le [mappage des types de données pour les indexeurs](#) pour vous assurer de générer le schéma d'index correctement et de définir les [mappages de champs d'indexeur](#) appropriés. Le message d'erreur comprendra des détails qui peuvent aider à identifier la source de l'incompatibilité.

## Erreur : Impossible d'utiliser la stratégie de suivi des modifications intégré car la table contient une clé primaire composite

Cela s'applique aux tables SQL, et se produit généralement lorsque la clé est définie en tant que clé composite ou, lorsque la table a défini un index cluster unique (comme avec l'index SQL, mais pas l'index Azure Search). Ceci est principalement dû au fait que l'attribut de clé est transformé en une clé primaire composite dans le cas d'un [index cluster unique](#). Dans ce cas, vérifiez que votre table SQL n'a pas d'index cluster unique, ou que vous mappez le champ clé à un champ pour lequel les valeurs en double sont impossibles.

## Erreur : Impossible de traiter le document en respectant le temps d'exécution max. de l'indexeur

Cette erreur se produit lorsque l'indexeur ne peut pas terminer le traitement d'un document unique à partir de la source de données en respectant le temps d'exécution autorisé. Le [temps d'exécution maximal](#) est plus court quand des ensembles de compétences sont utilisés. Lorsque cette erreur se produit, si `maxFailedItems` est défini sur une valeur autre que 0, l'indexeur ignore le document pour les prochaines exécutions, de façon que l'indexation puisse progresser. Si vous ne pouvez pas vous permettre d'ignorer un document, ou si vous voyez cette erreur en permanence, pensez à diviser les documents en documents

plus petits pour qu'une seule exécution de l'indexeur puisse traiter une partie de ces derniers.

<a name="could-not-project-document">

## Erreur : Impossible de projeter le document

Cette erreur se produit lorsque l'indexeur tente de [projeter des données dans une base de connaissances](#) et échoue. Cet échec peut être systématique et réparable, ou il peut s'agir d'un échec temporaire au niveau du récepteur de sortie de projection. Dans ce cas, vous devrez attendre un peu avant de réessayer. Voici certains états d'échecs connus et des solutions possibles.

MOTIF	DÉTAILS/EXEMPLE	RÉSOLUTION
Impossible de mettre à jour l'objet blob de projection <code>'blobUri'</code> dans le conteneur <code>'containerName'</code>	Le conteneur spécifié n'existe pas.	L'indexeur vérifie si le conteneur spécifié a déjà été créé et le crée si nécessaire, mais il n'effectue cette vérification qu'une seule fois par exécution de l'indexeur. Cette erreur signifie que quelque chose a supprimé le conteneur après cette étape. Pour résoudre cette erreur, essayez ceci : ne modifiez pas les informations de votre compte de stockage, attendez que l'exécution de l'indexeur se termine, puis réexécutez-le.
Impossible de mettre à jour l'objet blob de projection <code>'blobUri'</code> dans le conteneur <code>'containerName'</code>	Impossible d'écrire les données dans la connexion de transport : une connexion existante a dû être fermée par l'hôte distant.	Il s'agit d'un échec temporaire au niveau du stockage Azure. Pour résoudre le problème, vous devez réexécuter l'indexeur. Si vous rencontrez cette erreur de manière systématique, créez un <a href="#">ticket de support</a> afin que nous examinions le problème plus en détail.
Impossible de mettre à jour la ligne <code>'projectionRow'</code> dans la table <code>'tableName'</code>	Le serveur est occupé.	Il s'agit d'un échec temporaire au niveau du stockage Azure. Pour résoudre le problème, vous devez réexécuter l'indexeur. Si vous rencontrez cette erreur de manière systématique, créez un <a href="#">ticket de support</a> afin que nous examinions le problème plus en détail.

## Avertissement : Entrée de compétence non valide

Il manque une entrée à la compétence, un type est incorrect ou non valide. Le message d'avertissement indique l'impact :

1. Impossible d'exécuter une compétence
2. Compétence exécutée mais susceptible d'engendrer des résultats inattendus

Les compétences cognitives nécessitaient des entrées et des entrées facultatives. Par exemple, la [compétence d'extraction d'expression clé](#) a deux entrées obligatoires, `text` et `languageCode`, et aucune entrée facultative. Les entrées de compétences personnalisées sont toutes considérées comme des entrées facultatives.

S'il manque des entrées obligatoires ou si une entrée n'est pas du bon type, la compétence est ignorée et génère un avertissement. Les compétences ignorées ne génèrent aucun résultat. Par conséquent, si d'autres compétences utilisent des résultats de la compétence ignorée, elles peuvent générer des avertissements supplémentaires.

S'il manque une entrée facultative, la compétence est quand même exécutée, mais elle risque de produire une sortie inattendue en raison de l'entrée manquante.

Dans les deux cas, cet avertissement peut être attendu en raison de la forme de vos données. Par exemple, si vous avez un document contenant des informations sur des personnes avec les champs `firstName`, `middleName` et `lastName`, certains documents n'ont peut-être pas d'entrée pour `middleName`. Si vous passez `middleName` en tant qu'entrée à une compétence dans le pipeline, alors il est attendu que cette entrée de compétence soit parfois manquante. Vous avez besoin d'évaluer vos données et votre scénario pour déterminer si une action est nécessaire suite à cet avertissement.

Si vous souhaitez fournir une valeur par défaut en cas d'entrée manquante, vous pouvez utiliser la [compétence conditionnelle](#) pour générer une valeur par défaut, puis utiliser la sortie de la [compétence conditionnelle](#) en tant qu'entrée de compétence.

```
{
  "@odata.type": "#Microsoft.Skills.Util.ConditionalSkill",
  "context": "/document",
  "inputs": [
    { "name": "condition", "source": "=/document/language == null" },
    { "name": "whenTrue", "source": "= 'en'" },
    { "name": "whenFalse", "source": "=/document/language" }
  ],
  "outputs": [ { "name": "output", "targetName": "languageWithDefault" } ]
}
```

MOTIF	DÉTAILS/EXEMPLE	RÉSOLUTION
Le type de l'entrée de compétence est incorrect	« L'entrée de compétence requise n'est pas du type attendu <code>String</code> . Nom : <code>text</code> , source : <code>/document/merged_content</code> . » « L'entrée de compétence requise n'est pas au format attendu. Nom : <code>text</code> , source : <code>/document/merged_content</code> . » « Impossible d'effectuer une itération sur le non-tableau <code>/document/normalized_images/0/imageCelebrities</code> . » « Impossible de sélectionner <code>0</code> dans le non-tableau <code>/document/normalized_images/0/imageCelebrities/0/detail/celebrities</code> . »	Certaines compétences attendent des entrées de types particuliers. Par exemple, la <a href="#">compétence Sentiment</a> attend que <code>text</code> soit une chaîne. Si l'entrée spécifie une valeur autre qu'une chaîne, la compétence ne s'exécute pas et ne génère aucune sortie. Assurez-vous que le jeu de données contient des <a href="#">valeurs d'entrée de type uniforme</a> , ou utilisez une <a href="#">compétence d'API web personnalisée</a> pour prétraiter l'entrée. Si vous répétez la compétence sur un tableau, vérifiez que le contexte et l'entrée de la compétence ont <code>*</code> aux bons emplacements. Généralement, le contexte et la source d'entrée doivent se terminer par <code>*</code> pour des tableaux.
Une entrée de compétence est manquante	« L'entrée de compétence requise est manquante. Nom : <code>text</code> , source : <code>/document/merged_content</code> . » « Valeur manquante <code>/document/normalized_images/0/imageTags</code> . » « Impossible de sélectionner <code>0</code> dans un tableau <code>/document/pages</code> de longueur <code>0</code> . »	Si tous les documents reçoivent cet avertissement, il est hautement probable que les chemins d'entrée contiennent une faute de frappe et vous devriez vérifier attentivement la casse du nom de propriété, un signe <code>*</code> manquant ou en trop dans le chemin. Vérifiez aussi que les documents de la source de données définissent les entrées nécessaires.
L'entrée du code de la langue de la compétence n'est pas valide	L'entrée de compétence <code>languageCode</code> a les codes de langue suivants <code>X,Y,Z</code> , dont au moins un n'est pas valide.	Pour plus de détails, voir <a href="#">ci-dessous</a>

## Avertissement : L'entrée de compétence 'languageCode' contient les codes de langue suivants 'X,Y,Z', dont au moins un élément n'est pas valide.

Une ou plusieurs des valeurs passées dans l'entrée optionnelle `languageCode` d'une compétence en aval n'est pas prise en charge. Cela peut se produire si vous passez la sortie de [LanguageDetectionSkill](#) à des compétences ultérieures, et que la sortie comporte plus de langues que celles prises en charge dans ces compétences en aval.

Si vous savez que votre jeu de données utilise une seule langue, vous devriez supprimer [LanguageDetectionSkill](#) et l'entrée de compétence `languageCode`, puis utiliser plutôt `defaultLanguageCode` pour ce paramètre de compétence, en supposant que la langue est prise en charge pour cette compétence.

Si vous savez que votre jeu de données contient plusieurs langues et que vous avez donc besoin de [LanguageDetectionSkill](#) et de l'entrée `languageCode`, ajoutez un paramètre [ConditionalSkill](#) pour filtrer le texte selon les langues non prises en charge avant de passer le texte à la compétence en aval. Voici un exemple de résultat avec le paramètre [EntityRecognitionSkill](#) :

```
{
    "@odata.type": "#Microsoft.Skills.Util.ConditionalSkill",
    "context": "/document",
    "inputs": [
        { "name": "condition", "source": "= $(/document/language) == 'de' || $(/document/language) == 'en' || $(/document/language) == 'es' || $(/document/language) == 'fr' || $(/document/language) == 'it'" },
        { "name": "whenTrue", "source": "/document/content" },
        { "name": "whenFalse", "source": "= null" }
    ],
    "outputs": [ { "name": "output", "targetName": "supportedByEntityRecognitionSkill" } ]
}
```

Voici quelques références pour les langues actuellement prises en charge pour chacune des compétences qui peuvent générer ce message d'erreur :

- [Langues prises en charge pour l'analyse de texte](#) (pour [KeyPhraseExtractionSkill](#), [EntityRecognitionSkill](#), [SentimentSkill](#) et [PIIDetectionSkill](#))
- [Langues prises en charge par le traducteur](#) (pour [Text TranslationSkill](#))
- [Text SplitSkill](#) - langues prises en charge : da, de, en, es, fi, fr, it, ko, pt

## Avertissement : L'entrée de la compétence a été tronquée

Les compétences cognitives comportent des limites quant à la longueur du texte qui peut être analysé en une seule fois. Si la saisie de texte pour ces compétences dépasse cette limite, nous tronquerons le texte pour respecter la limite, puis appliquerons un enrichissement à ce texte tronqué. Cela signifie que la compétence est exécutée, mais pas sur toutes vos données.

Dans l'exemple LanguageDetectionSkill ci-dessous, le champ de saisie 'text' peut déclencher cet avertissement s'il dépasse la limite de caractères. Vous trouverez les limites de saisie des compétences dans la [documentation sur les compétences](#).

```
{
    "@odata.type": "#Microsoft.Skills.Text.LanguageDetectionSkill",
    "inputs": [
        {
            "name": "text",
            "source": "/document/text"
        }
    ],
    "outputs": [...]
}
```

Si vous voulez vous assurer que tout le texte est analysé, pensez à utiliser la [compétence Split](#).

## Avertissement : La réponse de compétence de l'API web contient des avertissements

Indexer a pu exécuter une compétence dans l'ensemble de compétences, mais la réponse à la demande de l'API web indique qu'il y a eu des avertissements durant l'exécution. Examinez les avertissements pour comprendre l'impact sur vos données et déterminer si une action est requise ou non.

## Avertissement : La configuration actuelle de l'indexeur ne prend pas en charge la progression incrémentielle

Cet avertissement s'affiche uniquement pour des sources de données Cosmos DB.

Dans le cas où l'exécution de l'indexeur est interrompue par des échecs passagers ou un dépassement du délai d'exécution, une progression incrémentielle pendant une indexation veille à ce que l'indexeur puisse reprendre là où il en était lors de sa dernière exécution, afin de ne pas avoir à tout réindexer depuis le début. Ceci est particulièrement important lors de l'indexation de grandes collections.

La possibilité de reprendre un travail d'indexation inachevé dépend de la disponibilité de documents ordonnés par la colonne '\_ts'. L'indexeur utilise l'horodatage pour déterminer le prochain document à récupérer. Si la colonne '\_ts' est manquante ou si l'indexeur ne peut pas déterminer si une requête personnalisée est ordonnée par celle-ci, l'indexeur commence au début, cet

avertissement s'affiche.

Il est possible de modifier ce comportement en activant la progression incrémentielle et en supprimant l'avertissement à l'aide de la propriété de configuration `assumeOrderByHighWatermarkColumn`.

Pour plus d'informations, consultez [Progression incrémentielle et requêtes personnalisées](#).

**Avertissement : Certaines données ont été perdues pendant la projection. La ligne « X » dans la table « Y » présente la propriété de chaîne « Z » qui était trop longue.**

Le [service Stockage Table](#) impose des limites sur la taille des [propriétés d'entité](#). Les chaînes peuvent comporter 32 000 caractères ou moins. Si une ligne présentant une propriété de chaîne de plus de 32 000 caractères est en cours de projection, seuls les 32 000 premiers caractères sont conservés. Pour contourner ce problème, évitez de projeter des lignes comportant des propriétés de chaîne de plus de 32 000 caractères.

## Avertissement : Texte extrait tronqué à X caractères

Les indexeurs limitent la quantité de texte qui peut être extraite d'un document. Cette limite dépend du niveau tarifaire : 32 000 caractères pour le niveau Gratuit, 64 000 pour le niveau De base, 4 millions pour le niveau Standard, 8 millions pour le niveau Standard S2 et 16 millions pour le niveau Standard S3. Le texte qui a été tronqué ne sera pas indexé. Pour éviter cet avertissement, essayez en scindant les documents avec de grandes quantités de texte en plusieurs documents plus petits.

Pour plus d'informations, consultez [Limites des indexeurs](#).

## Avertissement : Impossible de mapper le champ de sortie « X » à l'index de recherche

Les mappages de champs de sortie qui font référence à des données inexistantes/null génèrent des avertissements pour chaque document et créent un champ d'index vide. Pour contourner ce problème, vérifiez que les chemins sources de mappage de champs de sortie sont corrects ou définissez une valeur par défaut à l'aide de la [compétence conditionnelle](#). Pour plus d'informations, consultez [Mappage de champs de sortie](#).

MOTIF	DÉTAILS/EXEMPLE	RÉSOLUTION
Impossible d'effectuer une itération sur le non-tableau	« Impossible d'effectuer une itération sur le non-tableau <code>/document/normalized_images/0/imageCelebrity[0]</code> ». /document/normalized_images/0/imageCelebrity[0]	Cette erreur se produit lorsque la sortie n'est pas un tableau. Si vous pensez que la sortie /document/normalized_images/0/imageCelebrity[0] est un tableau, recherchez les erreurs dans le chemin d'accès du champ de source de sortie indiqué. Par exemple, vous pouvez avoir un <code>*</code> manquant ou supplémentaire dans le nom du champ source. Il est également possible que l'entrée de cette qualification soit Null, ce qui se traduit par un tableau vide. Trouvez des détails similaires dans la section <a href="#">Entrée de compétence non valide</a> .
Impossible de sélectionner <code>0</code> dans le non-tableau	« Impossible de sélectionner <code>0</code> dans le non-tableau <code>/document/pages</code> ». /document/pages	Cela peut se produire si la sortie des compétences ne produit pas de tableau et que le nom du champ de source de sortie a un index de tableau ou <code>*</code> dans son chemin d'accès. Vérifiez les chemins d'accès fournis dans les noms de champs de source de sortie et la valeur de champ pour le nom de champ indiqué. Trouvez des détails similaires dans la section <a href="#">Entrée de compétence non valide</a> .

**Avertissement : La stratégie de détection des modifications de données est configurée pour utiliser la colonne clé « X ».**

Pour détecter les modifications, les [stratégies de détection des modifications de données](#) ont des exigences spécifiques pour les colonnes qu'elles utilisent. L'une de ces exigences est que la colonne est mise à jour chaque fois que l'élément source est modifié. Une autre exigence est que la nouvelle valeur de cette colonne est supérieure à la valeur précédente. Les colonnes clés ne respectent pas cette exigence, car elles ne changent pas à chaque mise à jour. Pour contourner ce problème, sélectionnez une autre colonne pour la stratégie de détection des modifications.

## Avertissement : Le texte du document semble être encodé en UTF-16, mais il manque une marque d'ordre d'octet

Les [modes d'analyse de l'indexeur](#) doivent connaître la méthode d'encodage du texte avant de pouvoir l'analyser. Les deux méthodes les plus courantes pour encoder du texte sont UTF-16 et UTF-8. UTF-8 est un encodage de longueur variable où chaque caractère est compris entre 1 et 4 octets. UTF-16 est un encodage de longueur fixe où chaque caractère a une longueur de 2 octets. UTF-16 a deux variantes différentes, « avec primauté des octets de poids fort (big-endian) » et « mode Little Endian ». L'encodage de texte est déterminé par une « marque d'ordre d'octet », une série d'octets avant le texte.

ENCODAGE	MARQUE D'ORDRE D'OCTET
UTF-16 avec primauté des octets de poids fort (big-endian)	0xFF 0xFE
UTF-16 mode Little Endian	0xFE 0xFF
UTF-8	0xEF 0xBB 0xBF

Si aucune marque d'ordre d'octet n'est présente, le texte est supposé être encodé au format UTF-8.

Pour contourner cet avertissement, déterminez l'encodage de texte pour ce blob et ajoutez la marque d'ordre d'octet appropriée.

## Avertissement : La collection « X » de Cosmos DB a une stratégie d'indexation différée. Certaines données peuvent être perdues.

Les collections dotées de stratégies d'indexation [différée](#) ne peuvent pas être interrogées de manière cohérente, ce qui entraîne une absence de données dans votre indexeur. Pour contourner cet avertissement, modifiez votre stratégie d'indexation sur Cohérent.

## Avertissement : Le document contient des mots très longs (de plus de 64 caractères). Ces mots peuvent donner lieu à des prédictions de modèle tronquées et/ou non fiables.

Cet avertissement provient du service Analyse de texte. Dans certains cas, vous pouvez sans problème ignorer cet avertissement, par exemple quand votre document contient une URL longue (qui n'est sans doute pas une phrase clé, un sentiment de conduite, etc.). Sachez que lorsqu'un mot dépasse 64 caractères, il est tronqué à 64 caractères, ce qui peut impacter les prédictions de modèle.

# Vue d'ensemble du langage OData pour \$filter, \$orderby et \$select dans la Recherche cognitive Azure

04/10/2020 • 16 minutes to read • [Edit Online](#)

La Recherche cognitive Azure prend en charge un sous-ensemble de la syntaxe des expressions OData pour les expressions `$filter`, `$orderby` et `$select`. Les expressions de filtre sont évaluées lors de l'analyse de la requête, limitant la recherche à des champs spécifiques ou ajoutant des critères de correspondance utilisés lors du parcours des index. Les expressions `order by` sont appliquées dans une étape de post-traitement sur un jeu de résultats pour trier les documents renvoyés. Les expressions `select` déterminent les champs du document inclus dans le jeu de résultats. La syntaxe de ces expressions se distingue de la syntaxe de recherche [simple](#) ou [complète](#) utilisée dans le paramètre `search` et ce, malgré la présence d'un certain chevauchement pour référencer les champs.

Cet article fournit une vue d'ensemble du langage OData utilisé dans les expressions `filter`, `order by` et `select`. Ce langage est présenté de bas en haut, en commençant par les éléments de base. La syntaxe de niveau supérieur pour chaque paramètre est décrite dans un article distinct :

- [Syntaxe \\$filter](#)
- [Syntaxe \\$orderby](#)
- [Syntaxe \\$select](#)

Les expressions OData peuvent être aussi simples que complexes, mais toutes des éléments communs. Dans la Recherche cognitive Azure, les éléments de base d'une expression OData sont les suivants :

- Les **chemins de champs**, qui font référence à des champs spécifiques de votre index.
- Les **constantes**, qui sont les valeurs littérales d'un certain type de données.

## NOTE

Dans la Recherche cognitive Azure, la terminologie employée diffère à certains égards du langage [OData standard](#). Ce que nous appelons **champ** dans la Recherche cognitive Azure est appelé **propriété** dans OData. De même, **chemin de champ** est appelé **chemin de propriété**. Un **index** contenant des **documents** dans la Recherche cognitive Azure est plus généralement désigné comme un **jeu d'entités** contenant des **entités** dans OData. La terminologie de la Recherche cognitive Azure est utilisée tout au long de cet article.

## Chemins de champs

L'extension EBNF suivante ([Extended Backus-Naur Form](#)) définit la grammaire des chemins des champs.

```
field_path ::= identifier(''identifier)*  
identifier ::= [a-zA-Z_][a-zA-Z_0-9]*
```

Un diagramme de syntaxe interactif est également disponible :

[Diagramme de syntaxe OData pour la Recherche cognitive Azure](#)

#### NOTE

Consultez [Informations de référence sur la syntaxe d'expression OData pour Recherche cognitive Azure](#) pour l'extension EBNF complète.

Un chemin de champ comprend un ou plusieurs **identificateurs** séparés par des barres obliques. Chaque identificateur correspond à une séquence de caractères qui doit commencer par une lettre ASCII ou un trait de soulignement, et contenir uniquement des lettres ASCII, des chiffres ou des traits de soulignement. Il peut s'agir de lettres majuscules ou minuscules.

Un identificateur peut faire référence à un nom de champ ou à une **variable de portée** dans le contexte d'une [expression de collection](#) (`any` ou `a11`) au sein d'un filtre. Une variable de portée s'apparente à une variable de boucle qui représente l'élément actuel de la collection. Pour les collections complexes, cette variable représente un objet, ce qui explique pourquoi vous pouvez utiliser des chemins de champs pour faire référence aux sous-champs de la variable. Cela s'apparente à la notation par points de nombreux langages de programmation.

Des exemples de chemins de champs sont présentés dans le tableau suivant :

CHEMIN DE CHAMP	DESCRIPTION
<code>HotelName</code>	Fait référence à un champ d'index de niveau supérieur
<code>Address/City</code>	Fait référence au sous-champ <code>City</code> d'un champ d'index complexe ; <code>Address</code> est de type <code>Edm.ComplexType</code> dans cet exemple
<code>Rooms/Type</code>	Fait référence au sous-champ <code>Type</code> d'un champ de collection complexe ; <code>Rooms</code> est de type <code>Collection(Edm.ComplexType)</code> dans cet exemple
<code>Stores/Address/Country</code>	Fait référence au sous-champ <code>Country</code> du sous-champ <code>Address</code> d'un champ de collection complexe ; <code>Stores</code> est de type <code>Collection(Edm.ComplexType)</code> et <code>Address</code> de type <code>Edm.ComplexType</code> dans cet exemple
<code>room/Type</code>	Fait référence au sous-champ <code>Type</code> de la variable de portée <code>room</code> , par exemple, dans l'expression filter <code>Rooms/any(room: room/Type eq 'deluxe')</code>
<code>store/Address/Country</code>	Fait référence au sous-champ <code>Country</code> du sous-champ <code>Address</code> de la variable de portée <code>store</code> , par exemple, dans l'expression filter <code>Stores/any(store: store/Address/Country eq 'Canada')</code>

La signification d'un chemin de champ diffère selon le contexte. Dans les filtres, un chemin de champ fait référence à la valeur d'une *seule instance* de champ dans le document actif. Dans d'autres contextes, tels que `$orderby`, `$select`, ou dans la [recherche par champ de la syntaxe Lucene complète](#), un chemin de champ fait référence au champ à proprement parler. Cette différence n'est pas sans conséquence sur la manière d'utiliser les chemins de champs dans les filtres.

Examinez le chemin de champ `Address/City`. Dans un filtre, il fait référence à une seule ville pour le document actif, comme « San Francisco ». En revanche, `Rooms/Type` fait référence au sous-champ

Type pour de nombreuses chambres (par exemple, « standard » pour la première chambre, « deluxe » pour la deuxième chambre, etc.). Rooms/Type ne faisant pas référence à une *instance unique* du sous-champ Type, il ne peut pas être utilisé directement dans un filtre. Au lieu de cela, pour filtrer sur un type de chambre, vous devez utiliser une [expression lambda](#) avec une variable de portée, telle que :

```
Rooms/any(room: room/Type eq 'deluxe')
```

Dans cet exemple, la variable de portée room s'affiche dans le chemin de champ room/Type. Ainsi, room/Type fait référence au type de chambre actuel dans le document actif. Il s'agit d'une instance unique du sous-champ Type, et il peut donc être utilisé directement dans le filtre.

## Utilisation des chemins de champs

Les chemins de champs sont utilisés dans de nombreux paramètres des [API REST de Recherche cognitive Azure](#). Le tableau ci-dessous répertorie tous les emplacements où ils peuvent être utilisés, ainsi que les restrictions qui s'y rapportent :

API	NOM DU PARAMÈTRE	RESTRICTIONS
Créer ou Mettre à jour l'index	suggesters/sourceFields	None
Créer ou Mettre à jour l'index	scoringProfiles/text/weights	Peut uniquement faire référence à des champs <b>pouvant faire l'objet d'une recherche</b>
Créer ou Mettre à jour l'index	scoringProfiles/functions/fieldName	Peuvent uniquement faire référence à des champs <b>filtrables</b>
action	search quand queryType est full	Peut uniquement faire référence à des champs <b>pouvant faire l'objet d'une recherche</b>
action	facet	Peut uniquement faire référence à des champs <b>à choix multiples</b>
action	highlight	Peut uniquement faire référence à des champs <b>pouvant faire l'objet d'une recherche</b>
action	searchFields	Peut uniquement faire référence à des champs <b>pouvant faire l'objet d'une recherche</b>
Suggest et Autocomplete	searchFields	Peuvent uniquement faire référence à des champs appartenant à un <b>suggesteur</b>
Search, Suggest et Autocomplete	\$filter	Peuvent uniquement faire référence à des champs <b>filtrables</b>
Search et Suggest	\$orderby	Peuvent uniquement faire référence à des champs <b>triabiles</b>
Search, Suggest et Lookup	\$select	Peuvent uniquement faire référence à des champs <b>récupérables</b>

# Constantes

Dans OData, les constantes correspondent aux valeurs littérales d'un type [Entity Data Model](#) (EDM) donné. Pour obtenir la liste des types pris en charge dans la Recherche cognitive Azure, consultez [Types de données pris en charge](#). Les constantes des types de collection ne sont pas prises en charge.

Le tableau suivant présente des exemples de constantes pour chaque type de données pris en charge par la Recherche cognitive Azure :

TYPE DE DONNÉES	EXEMPLES DE CONSTANTES
Edm.Boolean	true , false
Edm.DateTimeOffset	2019-05-06T12:30:05.451Z
Edm.Double	3.14159 , -1.2e7 , NaN , INF , -INF
Edm.GeographyPoint	geography'POINT(-122.131577 47.678581)'
Edm.GeographyPolygon	geography'POLYGON((-122.031577 47.578581, -122.031577 47.678581, -122.131577 47.678581, -122.031577 47.578581))'
Edm.Int32	123 , -456
Edm.Int64	283032927235
Edm.String	'hello'

## Échappement des caractères spéciaux dans les constantes de chaîne

Les constantes de chaîne dans OData sont délimitées par des guillemets simples. Si vous devez construire une requête avec une constante de chaîne susceptible de contenir des guillemets simples, vous pouvez placer les guillemets incorporés dans une séquence d'échappement en les doublant.

Par exemple, une expression avec une apostrophe non mise en forme comme « voiture d'Alice » est représentée dans OData en tant que constante de chaîne `'Alice''s car'`.

### IMPORTANT

Quand vous construisez des filtres programmatiquement, pensez à échapper les constantes de chaîne qui proviennent d'une entrée utilisateur. Vous pouvez ainsi limiter les risques d'[attaques par injection](#), en particulier lors de l'utilisation de filtres pour implémenter un [filtrage de sécurité](#).

## Syntaxe des constantes

L'extension EBNF suivante ([Extended Backus-Naur Form](#)) définit la grammaire de la plupart des constantes présentées dans le tableau ci-dessus. La grammaire correspondant aux types géospatiaux est disponible dans [Fonctions géospatiales OData dans la Recherche cognitive Azure](#).

```

constant ::=

    string_literal
    | date_time_offset_literal
    | integer_literal
    | float_literal
    | boolean_literal
    | 'null'

string_literal ::= """([^\"]|\"")*"""

date_time_offset_literal ::= date_part'T' time_part time_zone

date_part ::= year'-'month'-'day

time_part ::= hour':'minute(':'second('.fractional_seconds)?))

zero_to_fifty_nine ::= [0-5]digit

digit ::= [0-9]

year ::= digit digit digit digit

month ::= '0'[1-9] | '1'[0-2]

day ::= '0'[1-9] | [1-2]digit | '3'[0-1]

hour ::= [0-1]digit | '2'[0-3]

minute ::= zero_to_fifty_nine

second ::= zero_to_fifty_nine

fractional_seconds ::= integer_literal

time_zone ::= 'Z' | sign hour':'minute

sign ::= '+' | '-'

/* In practice integer literals are limited in length to the precision of
the corresponding EDM data type. */
integer_literal ::= digit+

float_literal ::=

    sign? whole_part fractional_part? exponent?
    | 'NaN'
    | '-INF'
    | 'INF'

whole_part ::= integer_literal

fractional_part ::= '.'integer_literal

exponent ::= 'e' sign? integer_literal

boolean_literal ::= 'true' | 'false'

```

Un diagramme de syntaxe interactif est également disponible :

[Diagramme de syntaxe OData pour la Recherche cognitive Azure](#)

#### NOTE

Consultez [Informations de référence sur la syntaxe d'expression OData pour Recherche cognitive Azure](#) pour l'extension EBNF complète.

# Création d'expressions à partir de chemins de champs et de constantes

Les chemins de champs et les constantes constituent les éléments de base d'une expression OData, mais elles n'en restent pas moins des expressions complètes. Dans la Recherche cognitive Azure, le paramètre `$select` n'est en fait rien d'autre qu'une liste de chemins de champs séparés par des virgules, et `$orderby` n'est pas plus compliqué que `$select`. Si votre index contient un champ de type `Edm.Boolean`, vous pouvez même écrire un filtre qui n'est rien d'autre que le chemin de ce champ. Les constantes `true` et `false` sont également des filtres valides.

Cela étant, la plupart du temps vous aurez besoin d'expressions plus complexes faisant référence à plusieurs champs et constantes. Ces expressions sont générées de différentes manières selon le paramètre.

L'extension EBNF suivante ([Extended Backus-Naur Form](#)) définit la grammaire des paramètres `$filter`, `$orderby` et `$select`. Ils sont conçus à partir d'expressions plus simples faisant référence à des chemins de champs et constantes :

```
filter_expression ::= boolean_expression  
  
order_by_expression ::= order_by_clause(' ', ' order_by_clause)*  
  
select_expression ::= '*' | field_path(' ', ' field_path)*
```

Un diagramme de syntaxe interactif est également disponible :

[Diagramme de syntaxe OData pour la Recherche cognitive Azure](#)

## NOTE

Consultez [Informations de référence sur la syntaxe d'expression OData pour Recherche cognitive Azure](#) pour l'extension EBNF complète.

Les paramètres `$orderby` et `$select` correspondent à des listes d'expressions plus simples séparées par des virgules. Le paramètre `$filter` est une expression conditionnelle composée de plus simples sous-expressions. Ces sous-expressions sont combinées à l'aide d'opérateurs logiques tels que `and`, `or` et `not`, d'opérateurs de comparaison tels que `eq`, `lt`, `gt`, et ainsi de suite, et d'opérateurs de collection tels que `any` et `all`.

Les paramètres `$filter`, `$orderby` et `$select` sont abordés plus en détail dans les articles suivants :

- [Syntaxe OData \\$filter dans la Recherche cognitive Azure](#)
- [Syntaxe OData \\$orderby dans la Recherche cognitive Azure](#)
- [Syntaxe OData \\$select dans la Recherche cognitive Azure](#)

## Voir aussi

- [Navigation par facettes dans la Recherche cognitive Azure](#)
- [Filtres dans la Recherche cognitive Azure](#)
- [Rechercher des documents \(API REST de la recherche cognitive Azure\)](#)
- [Syntaxe de requête Lucene](#)
- [Syntaxe de requête simple dans la Recherche cognitive Azure](#)

# Syntaxe OData \$filter dans Recherche cognitive Azure

04/10/2020 • 13 minutes to read • [Edit Online](#)

Recherche cognitive Azure utilise des [expressions de filtre OData](#) pour appliquer des critères supplémentaires à une requête de recherche en plus des termes de recherche en texte intégral. Cet article décrit la syntaxe des filtres en détail. Pour obtenir des informations plus générales sur ce que sont les filtres et comment les utiliser pour des scénarios de requête spécifiques, consultez [Filtres dans Recherche cognitive Azure](#).

## Syntaxe

Un filtre dans le langage OData est une expression booléenne, qui à son tour peut prendre un de différents types d'expression, comme indiqué par l'EBNF ([Extended Backus-Naur Form](#)) suivant :

```
boolean_expression ::=  
    collection_filter_expression  
    | logical_expression  
    | comparison_expression  
    | boolean_literal  
    | boolean_function_call  
    | '(' boolean_expression ')'  
    | variable  
  
/* This can be a range variable in the case of a lambda, or a field path. */  
variable ::= identifier | field_path
```

Un diagramme de syntaxe interactif est également disponible :

[Diagramme de syntaxe OData pour la Recherche cognitive Azure](#)

### NOTE

Consultez [Informations de référence sur la syntaxe d'expression OData pour Recherche cognitive Azure](#) pour l'extension EBNF complète.

Les types d'expressions booléennes incluent les suivants :

- Expressions de filtre de collection utilisant `any` ou `all`. Celles-ci appliquent des critères de filtre aux champs de collection. Pour plus d'informations, consultez [Opérateurs de collection OData dans Recherche cognitive Azure](#).
- Expressions logiques qui combinent d'autres expressions booléennes à l'aide des opérateurs `and`, `or` et `not`. Pour plus d'informations, consultez [Opérateurs logiques OData dans Recherche cognitive Azure](#).
- Expressions de comparaison, qui comparent des champs ou variables de plage à des valeurs constantes à l'aide des opérateurs `eq`, `ne`, `gt`, `lt`, `ge` et `le`. Pour plus d'informations, consultez [Opérateurs de comparaison OData dans Recherche cognitive Azure](#). Les expressions de comparaison sont également utilisées pour comparer des distances entre des coordonnées géospatiales à l'aide de la fonction `geo.distance`. Pour plus d'informations, consultez [Fonctions géospatiales OData dans Recherche cognitive Azure](#).
- Littéraux booléens `true` et `false`. Ces constantes peuvent parfois être utiles lors de la génération de filtre par programmation, mais ne sont pas systématiquement utilisées dans la pratique.

- Appels de fonctions booléennes, notamment :
  - `geo.intersects`, qui teste si un point donné se trouve dans un polygone donné. Pour plus d'informations, consultez [Fonctions géospatiales OData dans Recherche cognitive Azure](#).
  - `search.in`, qui compare un champ ou une variable de plage avec chaque valeur d'une liste de valeurs. Pour plus d'informations, consultez [Fonction OData `search.in` dans Recherche cognitive Azure](#).
  - `search.ismatch` et `search.ismatchscoring`, qui exécutent des opérations de recherche en texte intégral dans un contexte de filtre. Pour plus d'informations, consultez [Fonctions de recherche en texte intégral OData dans Recherche cognitive Azure](#).
- Chemins d'accès de champ ou variables de plage de type `Edm.Boolean`. Par exemple, si votre index comporte un champ booléen appelé `.IsEnabled` et que vous souhaitez retourner tous les documents où ce champ est `true`, votre expression de filtre peut être simplement le nom `.IsEnabled`.
- Expressions booléennes entre parenthèses. L'utilisation de parenthèses peut vous aider à déterminer explicitement l'ordre des opérations dans un filtre. Pour plus d'informations sur la priorité par défaut des opérateurs OData, consultez la section suivante.

### Précédence des opérateurs dans les filtres

Si vous écrivez une expression de filtre sans parenthèses autour de ses sous-expressions, Recherche cognitive Azure l'évalue en fonction d'un ensemble de règles de priorité d'opérateur. Ces règles sont basées sur les opérateurs utilisés pour combiner des sous-expressions. Le tableau suivant répertorie les groupes d'opérateurs dans l'ordre de la priorité la plus élevée à la plus faible :

GROUPE	OPÉRATEUR(S)
Opérateurs logiques	<code>not</code>
Opérateurs de comparaison	<code>eq</code> , <code>ne</code> , <code>gt</code> , <code>lt</code> , <code>ge</code> , <code>le</code>
Opérateurs logiques	<code>and</code>
Opérateurs logiques	<code>or</code>

Un opérateur qui est plus élevé dans le tableau ci-dessus sera « lié plus étroitement » à ses opérandes que les autres opérateurs. Par exemple, `and` a une priorité plus élevée que `or`, et les opérateurs de comparaison ont une priorité plus élevée que ces deux-ci. Par conséquent, les deux expressions suivantes sont équivalentes :

```
Rating gt 0 and Rating lt 3 or Rating gt 7 and Rating lt 10
((Rating gt 0) and (Rating lt 3)) or ((Rating gt 7) and (Rating lt 10))
```

L'opérateur `not` a la priorité la plus élevée de tous, supérieure encore aux opérateurs de comparaison. C'est pourquoi, si vous essayez d'écrire un filtre tel que le suivant :

```
not Rating gt 5
```

Vous obtenez ce message d'erreur :

```
Invalid expression: A unary operator with an incompatible type was detected. Found operand type 'Edm.Int32' for operator kind 'Not'.
```

Cette erreur se produit car l'opérateur est associé au champ `Rating` uniquement, qui est de type `Edm.Int32`, et non à l'expression de comparaison entière. La solution consiste à placer l'opérande de `not` entre parenthèses :

```
not (Rating gt 5)
```

## Limitations de taille des filtres

Il existe des limites de taille et de complexité des expressions de filtre que vous pouvez envoyer à Recherche cognitive Azure. Les limites sont en gros basées sur le nombre de clauses de votre expression de filtre. Une directive appropriée est que, si vous avez des centaines de clauses, vous risquez de dépasser la limite. Nous vous recommandons de concevoir votre application de telle sorte qu'elle ne génère pas de filtres de taille illimitée.

### TIP

L'utilisation de la fonction `search.in` au lieu de disjonctions longues de comparaisons d'égalité peut permettre d'éviter la limite de clause de filtre, dans la mesure où un appel de fonction est comptabilisé comme une seule clause.

## Exemples

Rechercher tous les hôtels avec au moins une chambre, un tarif de base inférieur à 200 \$ et une note d'évaluation égale ou supérieure à 4 :

```
$filter=Rooms/any(room: room/BaseRate lt 200.0) and Rating ge 4
```

Rechercher tous les hôtels autres que « Sea View Motel » qui ont été rénovés depuis 2010 :

```
$filter=HotelName ne 'Sea View Motel' and LastRenovationDate ge 2010-01-01T00:00:00Z
```

Rechercher tous les hôtels qui ont été rénovés en 2010 ou ultérieurement. Le littéral DateHeure inclut des informations de fuseau horaire pour l'heure standard du Pacifique :

```
$filter=LastRenovationDate ge 2010-01-01T00:00:00-08:00
```

Rechercher tous les hôtels avec un parking inclus et où toutes les chambres sont non-fumeur :

```
$filter=ParkingIncluded and Rooms/all(room: not room/SmokingAllowed)
```

- OR -

```
$filter=ParkingIncluded eq true and Rooms/all(room: room/SmokingAllowed eq false)
```

Rechercher tous les hôtels de catégorie « Luxury » ou incluant un parking et ayant une évaluation de 5 :

```
$filter=(Category eq 'Luxury' or ParkingIncluded eq true) and Rating eq 5
```

Rechercher tous les hôtels avec la balise « wifi » dans au moins une chambre (où chaque chambre a des balises stockées dans un champ `Collection(Edm.String)`) :

```
$filter=Rooms/any(room: room/Tags/any(tag: tag eq 'wifi'))
```

Rechercher tous les hôtels avec n'importe quelles chambres :

```
$filter=Rooms/any()
```

Rechercher tous les hôtels qui n'ont pas de chambres :

```
$filter=not Rooms/any()
```

Rechercher tous les hôtels dans les 10 kilomètres d'un point de référence donnée (où `Location` est un champ de type `Edm.GeographyPoint`) :

```
$filter=geo.distance(Location, geography'POINT(-122.131577 47.678581)') le 10
```

Rechercher tous les hôtels dans une fenêtre d'affichage décrite sous forme de polygone (où `Location` est un champ de type `Edm.GeographyPoint`). Le polygone doit être fermé, ce qui signifie que le premier point et le dernier doivent être le même. En outre, [les points doivent être répertoriés dans le sens inverse des aiguilles d'une montre](#).

```
$filter=geo.intersects(Location, geography'POLYGON((-122.031577 47.578581, -122.031577 47.678581, -122.131577 47.678581, -122.031577 47.578581))')
```

Rechercher tous les hôtels où le champ « Description » est nul. Le champ sera nul s'il n'a jamais été défini ou s'il a été défini explicitement sur la valeur Null :

```
$filter=Description eq null
```

Recherchez tous les hôtels avec un nom égal à « Sea View motel » ou « Budget hotel ». Ces expressions contiennent des espaces, et l'espace est un séparateur par défaut. Vous pouvez spécifier un autre séparateur entre guillemets simples comme troisième paramètre de chaîne :

```
$filter=search.in(HotelName, 'Sea View motel,Budget hotel', ',')
```

Recherchez tous les hôtels avec un nom égal à « Sea View motel » ou « Budget hotel » séparés par « | » :

```
$filter=search.in(HotelName, 'Sea View motel|Budget hotel', '|')
```

Recherchez tous les hôtels où toutes les chambres comportent la balise « wifi » ou « baignoire » :

```
$filter=Rooms/any(room: room/Tags/any(tag: search.in(tag, 'wifi, tub')))
```

Recherchez une correspondance entre des expressions tirées d'une collection, telles que « heated towel racks » ou « hairdryer included » dans les balises.

```
$filter=Rooms/any(room: room/Tags/any(tag: search.in(tag, 'heated towel racks,hairdryer included', ',')))
```

Rechercher les documents avec le mot « waterfront ». Cette requête de filtre est identique à une [demande de recherche](#) avec `search=waterfront`.

```
$filter=search.ismatchscoring('waterfront')
```

Rechercher les documents avec le mot « hostel » et une évaluation supérieure ou égale à 4, ou les documents avec le mot « motel » et une évaluation égale à 5. Cette requête n'a pas pu être exprimée sans la fonction `search.ismatchscoring` dans la mesure où elle combine la recherche en texte intégral et des opérations de filtre à l'aide de `or`.

```
$filter=search.ismatchscoring('hostel') and rating ge 4 or search.ismatchscoring('motel') and rating eq  
5
```

Rechercher les documents sans le mot « luxury ».

```
$filter=not search.ismatch('luxury')
```

Rechercher les documents avec la phrase « ocean view » ou une évaluation égale à 5. La requête `search.ismatchscoring` sera exécutée seulement sur les champs `HotelName` et `Description`. Les documents qui correspondent uniquement à la deuxième clause de la disjonction sont également renvoyés (hôtels avec une `Rating` égale à 5). Ces documents sont renvoyés avec un score égal à zéro pour montrer clairement qu'ils ne correspondent à aucune des parties avec score de l'expression.

```
$filter=search.ismatchscoring('"ocean view"', 'Description,HotelName') or Rating eq 5
```

Recherchez les hôtels où les termes « hôtel » et « aéroport » sont séparés par plus de cinq mots dans la description, et où toutes les chambres sont non-fumeur. Cette requête utilise le [langage de requête Lucene complet](#).

```
$filter=search.ismatch('"hotel airport"~5', 'Description', 'full', 'any') and not Rooms/any(room:  
room/SmokingAllowed)
```

## Étapes suivantes

- [Filtres dans la Recherche cognitive Azure](#)
- [Vue d'ensemble du langage d'expression OData pour Recherche cognitive Azure](#)
- [Informations de référence sur la syntaxe d'expression OData pour Recherche cognitive Azure](#)
- [Rechercher des documents \(API REST de la recherche cognitive Azure\)](#)

# Syntaxe OData \$orderby dans Recherche cognitive Azure

04/10/2020 • 4 minutes to read • [Edit Online](#)

Vous pouvez utiliser le [paramètre OData \\$orderby](#) pour appliquer un ordre de tri personnalisé pour les résultats de recherche dans Recherche cognitive Azure. Cet article décrit la syntaxe de `$orderby` en détail. Pour plus d'informations sur l'utilisation de `$orderby` lors de la présentation des résultats de recherche, consultez [Guide pratique pour utiliser les résultats de recherche dans Recherche cognitive Azure](#).

## Syntaxe

Le paramètre `$orderby` accepte une liste séparée par des virgules de 32 [clauses orderby](#). La syntaxe d'une clause orderby est décrite par l'extension EBNF suivante ([Extended Backus-Naur Form](#)) :

```
order_by_clause ::= (field_path | sortable_function) ('asc' | 'desc')?  
sortable_function ::= geo_distance_call | 'search.score()'
```

Un diagramme de syntaxe interactif est également disponible :

[Diagramme de syntaxe OData pour la Recherche cognitive Azure](#)

### NOTE

Consultez [Informations de référence sur la syntaxe d'expression OData pour Recherche cognitive Azure](#) pour l'extension EBNF complète.

Chaque clause possède des critères de tri, éventuellement suivis d'un ordre de tri (`asc` pour l'ordre croissant ou `desc` pour l'ordre décroissant). Si vous ne spécifiez pas d'ordre, la valeur par défaut est croissant. Si le champ contient des valeurs Null, celles-ci s'affichent en premier si le tri est `asc` et en dernier si le tri est `desc`.

Le critère de tri peut être le trajet d'un champ `sortable`, ou un appel à pour les fonctions `geo.distance` ou `search.score`.

Si plusieurs documents ont les mêmes critères de tri et que la fonction `search.score` n'est pas utilisée (par exemple si vous triez sur un champ numérique `Rating` et que trois documents ont une évaluation de 4), les regroupements sont effectués par score des documents en ordre décroissant. Quand les scores des documents sont identiques (par exemple quand aucune requête de recherche en texte intégral n'est spécifiée dans la demande), l'ordre relatif des documents regroupés est indéterminé.

Vous pouvez spécifier plusieurs critères de tri. L'ordre des expressions détermine l'ordre de tri final. Par exemple, pour trier par ordre décroissant sur le score, suivi de l'évaluation, la syntaxe est

```
$orderby=search.score() desc,Rating desc .
```

La syntaxe pour `geo.distance` dans `$orderby` est la même que dans `$filter`. Lors de l'utilisation de `geo.distance` dans `$orderby`, le champ auquel elle s'applique doit être de type `Edm.GeographyPoint` et être aussi `sortable`.

La syntaxe pour `search.score` dans `$orderby` est `search.score()`. La fonction `search.score` ne prend aucun paramètre.

## Exemples

Tirer les hôtels par ordre croissant sur le tarif de base :

```
$orderby=BaseRate asc
```

Trier les hôtels par ordre décroissant sur l'évaluation, puis par ordre croissant sur le tarif de base (rappelez-vous que l'ordre croissant est la valeur par défaut) :

```
$orderby=Rating desc,BaseRate
```

Trier les hôtels par ordre décroissant sur l'évaluation, puis par ordre croissant sur la distance à partir des coordonnées spécifiées :

```
$orderby=Rating desc,geo.distance(Location, geography'POINT(-122.131577 47.678581)') asc
```

Trier les hôtels par ordre décroissant sur search.score et l'évaluation, puis par ordre croissant sur la distance à partir des coordonnées spécifiées. Entre deux d'hôtels avec des scores ou une évaluation identique, le plus proche est listé en premier :

```
$orderby=search.score() desc,Rating desc,geo.distance(Location, geography'POINT(-122.131577 47.678581)') asc
```

## Étapes suivantes

- [Guide pratique pour utiliser les résultats de recherche dans Recherche cognitive Azure](#)
- [Vue d'ensemble du langage d'expression OData pour Recherche cognitive Azure](#)
- [Informations de référence sur la syntaxe d'expression OData pour la Recherche cognitive Azure](#)
- [Rechercher des documents \(API REST de la recherche cognitive Azure\)](#)

# Syntaxe OData \$select dans la Recherche cognitive Azure

04/10/2020 • 3 minutes to read • [Edit Online](#)

Vous pouvez utiliser le paramètre **\$select** OData pour choisir les champs à inclure dans les résultats de recherche à partir de la Recherche cognitive Azure. Cet article décrit la syntaxe de **\$select** en détail. Pour plus d'informations sur l'utilisation de **\$select** lors de la présentation des résultats de recherche, consultez [Guide pratique pour utiliser les résultats de la recherche dans la Recherche cognitive Azure](#).

## Syntaxe

Le paramètre **\$select** détermine quels champs de chaque document sont renvoyés dans le jeu de résultats de la requête. L'extension EBNF suivante ([Extended Backus-Naur Form](#)) définit la grammaire du paramètre **\$select** :

```
select_expression ::= '*' | field_path(' ',' field_path)*  
field_path ::= identifier('/'identifier)*
```

Un diagramme de syntaxe interactif est également disponible :

[Diagramme de syntaxe OData pour la Recherche cognitive Azure](#)

### NOTE

Consultez [Informations de référence sur la syntaxe d'expression OData pour Recherche cognitive Azure](#) pour l'extension EBNF complète.

Le paramètre **\$select** se présente sous deux formes :

1. Une seule étoile (\*), indiquant que tous les champs récupérables doivent être renvoyés, ou
2. Une liste de chemins d'accès aux champs séparés par des virgules identifiant les champs à renvoyer.

Lorsque vous utilisez la deuxième forme, vous pouvez uniquement spécifier les champs récupérables dans la liste.

Si vous répertoriez un champ complexe sans spécifier explicitement ses sous-champs, tous les sous-champs récupérables figureront dans le jeu de résultats de la requête. Par exemple, supposons que votre index comporte un champ `Address` avec les sous-champs `Street`, `City`, et `Country` qui sont tous récupérables. Si vous spécifiez `Address` dans `$select`, les résultats de la requête incluront les trois sous-champs.

## Exemples

Incluez champs `HotelId`, `HotelName`, et `Rating` de niveau supérieur dans les résultats, ainsi que le sous-champ `City` de `Address` :

```
$select=HotelId, HotelName, Rating, Address/City
```

Le résultat peut ressembler à cet exemple :

```
{  
    "HotelId": "1",  
    "HotelName": "Secret Point Motel",  
    "Rating": 4,  
    "Address": {  
        "City": "New York"  
    }  
}
```

Incluez le champ `HotelName` de niveau supérieur dans les résultats, ainsi que tous les sous-champs de `Address` et les sous-champs `Type` et `BaseRate` de chaque objet de la collection `Rooms` :

```
$select=HotelName, Address, Rooms/Type, Rooms/BaseRate
```

Le résultat peut ressembler à cet exemple :

```
{  
    "HotelName": "Secret Point Motel",  
    "Rating": 4,  
    "Address": {  
        "StreetAddress": "677 5th Ave",  
        "City": "New York",  
        "StateProvince": "NY",  
        "Country": "USA",  
        "PostalCode": "10022"  
    },  
    "Rooms": [  
        {  
            "Type": "Budget Room",  
            "BaseRate": 9.69  
        },  
        {  
            "Type": "Budget Room",  
            "BaseRate": 8.09  
        }  
    ]  
}
```

## Étapes suivantes

- [Guide pratique pour utiliser les résultats de la recherche dans la Recherche cognitive Azure](#)
- [Vue d'ensemble du langage d'expression OData pour Recherche cognitive Azure](#)
- [Informations de référence sur la syntaxe d'expression OData pour Recherche cognitive Azure](#)
- [Rechercher des documents \(API REST de la recherche cognitive Azure\)](#)

# Opérateurs de collection OData dans la Recherche cognitive Azure - `any` et `all`

04/10/2020 • 6 minutes to read • [Edit Online](#)

Lorsque vous écrivez une [expression de filtre OData](#) à utiliser avec la Recherche cognitive Azure, il est souvent utile de filtrer sur les champs de la collection. Vous pouvez y parvenir à l'aide des opérateurs `any` et `all`.

## Syntaxe

L'extension EBNF suivante ([Extended Backus-Naur Form](#)) définit la grammaire d'une expression OData utilisant `any` ou `all`.

```
collection_filter_expression ::=  
    field_path'/all(' lambda_expression ')'  
  | field_path'/any(' lambda_expression ')'  
  | field_path'/any()'  
  
lambda_expression ::= identifier ':' boolean_expression
```

Un diagramme de syntaxe interactif est également disponible :

[Diagramme de syntaxe OData pour la Recherche cognitive Azure](#)

### NOTE

Consultez [Informations de référence sur la syntaxe d'expression OData pour Recherche cognitive Azure](#) pour l'extension EBNF complète.

Il existe trois formes d'expression qui filtrent les collections.

- Les deux premiers itèrent sur un champ de collection, en appliquant un prédictat donné sous la forme d'une expression lambda à chaque élément de la collection.
  - Une expression qui utilise `all` renvoie `true` si le prédictat a la valeur true pour chaque élément de la collection.
  - Une expression qui utilise `any` renvoie `true` si le prédictat a la valeur true pour au moins un élément de la collection.
- La troisième forme de filtre de collection utilise `any` sans expression lambda pour tester si un champ de la collection est vide. Si la collection comporte des éléments, elle renvoie `true`. Si la collection est vide, elle renvoie `false`.

Un **expression lambda** d'un filtre de collection est comme le corps d'une boucle dans un langage de programmation. Elle définit une variable, appelée **variable de portée**, qui contient l'élément actuel de la collection pendant l'itération. Elle définit également une autre expression booléenne qui est le critère de filtre à appliquer à la variable de portée pour chaque élément de la collection.

## Exemples

Correspondance des documents dont le champ `tags` contient exactement la chaîne « wifi » :

```
tags/any(t: t eq 'wifi')
```

Correspondance des documents où chaque élément du champ `ratings` se situe entre 3 et 5, y compris :

```
ratings/all(r: r ge 3 and r le 5)
```

Correspondance des documents où chaque coordonnée géographique du champ `locations` se trouve dans le polygone donné :

```
locations/any(loc: geo.intersects(loc, geography'POLYGON((-122.031577 47.578581, -122.031577 47.678581, -122.131577 47.678581, -122.031577 47.578581))'))
```

Correspondance des documents dans lesquels le champ `rooms` est vide :

```
not rooms/any()
```

Correspondance des documents où pour toutes les salles, le champ `rooms/amenities` contient « tv » et `rooms/baseRate` est inférieur à 100 :

```
rooms/all(room: room/amenities/any(a: a eq 'tv') and room/baseRate lt 100.0)
```

## Limites

Toutes les fonctionnalités d'expressions de filtre ne sont pas disponibles dans le corps d'une expression lambda. Les restrictions diffèrent selon le type de données du champ de la collection que vous souhaitez filtrer. Le tableau suivant récapitule les restrictions :

TYPE DE DONNÉES	FONCTIONNALITÉS AUTORISÉES DANS LES EXPRESSIONS LAMBDA AVEC ANY	FONCTIONNALITÉS AUTORISÉES DANS LES EXPRESSIONS LAMBDA AVEC ALL
<code>Collection(Edm.ComplexType)</code>	Tout sauf <code>search.ismatch</code> et <code>search.ismatchscoring</code>	Identique
<code>Collection(Edm.String)</code>	Comparaisons avec <code>eq</code> ou <code>search.in</code> Combinaison de sous-expressions avec <code>or</code>	Comparaisons avec <code>ne</code> ou <code>not search.in()</code> Combinaison de sous-expressions avec <code>and</code>
<code>Collection(Edm.Boolean)</code>	Comparaisons avec <code>eq</code> ou <code>ne</code>	Identique
<code>Collection(Edm.GeographyPoint)</code>	Utilisation de <code>geo.distance</code> avec <code>lt</code> ou <code>le</code> <code>geo.intersects</code> Combinaison de sous-expressions avec <code>or</code>	Utilisation de <code>geo.distance</code> avec <code>gt</code> ou <code>ge</code> <code>not geo.intersects(...)</code> Combinaison de sous-expressions avec <code>and</code>

TYPE DE DONNÉES	FONCTIONNALITÉS AUTORISÉES DANS LES EXPRESSIONS LAMBDA AVEC ANY	FONCTIONNALITÉS AUTORISÉES DANS LES EXPRESSIONS LAMBDA AVEC ALL
Collection(Edm.DateTimeOffset), Collection(Edm.Double), Collection(Edm.Int32), Collection(Edm.Int64)	<p>Comparaisons avec <code>eq</code>, <code>ne</code>, <code>lt</code>, <code>gt</code>, <code>le</code> ou <code>ge</code></p> <p>Combinaison de comparaisons avec d'autres sous-expressions à l'aide de <code>or</code></p> <p>Combinaison de comparaisons, sauf <code>ne</code>, avec d'autres sous-expressions à l'aide de <code>and</code></p> <p>Expressions utilisant des combinaisons de <code>and</code> et <code>or</code> en <a href="#">forme normale disjonctive (FND)</a></p>	<p>Comparaisons avec <code>eq</code>, <code>ne</code>, <code>lt</code>, <code>gt</code>, <code>le</code> ou <code>ge</code></p> <p>Combinaison de comparaisons avec d'autres sous-expressions à l'aide de <code>and</code></p> <p>Combinaison de comparaisons, sauf <code>eq</code>, avec d'autres sous-expressions à l'aide de <code>or</code></p> <p>Expressions utilisant des combinaisons de <code>and</code> et <code>or</code> en <a href="#">forme normale conjonctive (FNC)</a></p>

Pour plus d'informations sur ces restrictions et pour obtenir des exemples, consultez [Résolution des problèmes de filtres de collection dans la Recherche cognitive Azure](#). Pour en savoir plus sur la raison de l'existence de ces restrictions, consultez [Présentation des filtres de collection dans la Recherche cognitive Azure](#).

## Étapes suivantes

- [Filtres dans la Recherche cognitive Azure](#)
- [Vue d'ensemble du langage d'expression OData pour Recherche cognitive Azure](#)
- [Informations de référence sur la syntaxe d'expression OData pour Recherche cognitive Azure](#)
- [Rechercher des documents \(API REST de la recherche cognitive Azure\)](#)

# Opérateurs de comparaison OData dans Recherche cognitive Azure : `eq`, `ne`, `gt`, `lt`, `ge`, `le`

04/10/2020 • 10 minutes to read • [Edit Online](#)

L'opération la plus basique dans une [expression de filtre OData](#) dans Recherche cognitive Azure consiste à comparer un champ à une valeur donnée. Deux types de comparaison sont possibles : la comparaison d'égalité et la comparaison de plages. Vous pouvez utiliser les opérateurs suivants pour comparer un champ à une valeur constante :

Opérateurs d'égalité :

- `eq` : teste si un champ est **égal** à une valeur constante
- `ne` : teste si un champ n'est **pas égal** à une valeur constante

Opérateurs de plage :

- `gt` : teste si un champ est **supérieur** à une valeur constante
- `lt` : teste si un champ est **inférieur** à une valeur constante
- `ge` : teste si un champ est **supérieur ou égal** à une valeur constante
- `le` : teste si un champ est **inférieur ou égal** à une valeur constante

Vous pouvez utiliser plusieurs opérateurs de plage en association avec les [opérateurs logiques](#) afin de tester si un champ est compris dans une certaine plage de valeurs. Consultez les [exemples](#) disponibles plus loin dans cet article.

## NOTE

Si vous préférez, vous pouvez placer la valeur constante sur le côté gauche de l'opérateur et le nom du champ sur le côté droit. Pour les opérateurs de plage, la signification de la comparaison est inversée. Par exemple, si la valeur constante se trouve sur la gauche, `gt` teste si la valeur constante est supérieure à celle du champ. Vous pouvez également utiliser les opérateurs de comparaison pour comparer le résultat d'une fonction, par exemple `geo.distance`, à une valeur. Pour les fonctions booléennes telles que `search.ismatch`, vous n'êtes pas obligé de comparer le résultat à `true` ou `false`.

## Syntaxe

L'extension EBNF suivante ([Extended Backus-Naur Form](#)) définit la grammaire d'une expression OData utilisant les opérateurs de comparaison.

```
comparison_expression ::=  
    variable_or_function comparison_operator constant |  
    constant comparison_operator variable_or_function  
  
variable_or_function ::= variable | function_call  
  
comparison_operator ::= 'gt' | 'lt' | 'ge' | 'le' | 'eq' | 'ne'
```

Un diagramme de syntaxe interactif est également disponible :

[Diagramme de syntaxe OData pour la Recherche cognitive Azure](#)

#### NOTE

Consultez [Informations de référence sur la syntaxe d'expression OData pour Recherche cognitive Azure](#) pour l'extension EBNF complète.

Il existe deux formes d'expressions de comparaison. La seule différence entre eux est si la constante apparaît sur la gauche ou la droite de l'opérateur. L'expression de l'autre côté de l'opérateur doit être une **variable** ou un appel de fonction. Une variable peut être un nom de champ ou une variable de portée dans le cadre d'une [expression lambda](#).

## Types de données pour les comparaisons

Les types de données des deux côtés d'un opérateur de comparaison doivent être compatibles. Par exemple, si le côté gauche est un champ de type `Edm.DateTimeOffset`, le côté droit doit être une valeur date-heure constante. Les types de données numériques sont plus flexibles. Vous pouvez comparer les variables et les fonctions de n'importe quel type numérique avec des constantes du type numérique de votre choix, avec cependant quelques limitations, comme décrit dans le tableau suivant.

TYPE DE VARIABLE OU DE FONCTION	TYPE VALEUR CONSTANT	LIMITES
<code>Edm.Double</code>	<code>Edm.Double</code>	La comparaison est soumise à des règles particulières pour <code>NaN</code>
<code>Edm.Double</code>	<code>Edm.Int64</code>	La constante est convertie en <code>Edm.Double</code> , ce qui entraîne une perte de précision pour les valeurs de grande ampleur
<code>Edm.Double</code>	<code>Edm.Int32</code>	n/a
<code>Edm.Int64</code>	<code>Edm.Double</code>	Les comparaisons avec <code>NaN</code> , <code>-INF</code> , ou <code>INF</code> ne sont pas autorisées
<code>Edm.Int64</code>	<code>Edm.Int64</code>	n/a
<code>Edm.Int64</code>	<code>Edm.Int32</code>	La constante est convertie en <code>Edm.Int64</code> avant la comparaison
<code>Edm.Int32</code>	<code>Edm.Double</code>	Les comparaisons avec <code>NaN</code> , <code>-INF</code> , ou <code>INF</code> ne sont pas autorisées
<code>Edm.Int32</code>	<code>Edm.Int64</code>	n/a
<code>Edm.Int32</code>	<code>Edm.Int32</code>	n/a

Pour les comparaisons qui ne sont pas autorisées, notamment la comparaison d'un champ de type `Edm.Int64` à `NaN`, l'API REST Recherche cognitive Azure renvoie une erreur « HTTP 400 : demande incorrecte ».

## IMPORTANT

Même si les comparaisons de types numériques sont flexibles, nous vous recommandons vivement d'écrire des comparaisons dans des filtres afin que la valeur constante soit du même type de données que la variable ou la fonction à laquelle elle est comparée. Cela est particulièrement important lorsque des valeurs à virgule flottante et entières sont mélangées, car les conversions implicites peuvent entraîner une perte de précision.

### Cas particuliers pour `null` et `NaN`

Lorsque vous utilisez des opérateurs de comparaison, il est important de se rappeler que tous les champs qui ne sont pas des collections dans Recherche cognitive Azure peuvent potentiellement être `null`. Le tableau suivant répertorie tous les résultats possibles pour une expression de comparaison où chaque côté peut être `null` :

OPÉRATEUR	RÉSULTAT LORSQUE SEUL LE CHAMP OU LA VARIABLE EST <code>NULL</code>	RÉSULTAT LORSQUE SEULE LA CONSTANTE EST <code>NULL</code>	RÉSULTAT LORSQUE LE CHAMP/LA VARIABLE ET LA CONSTANTE SONT <code>NULL</code>
<code>gt</code>	<code>false</code>	HTTP 400 : erreur de demande incorrecte	HTTP 400 : erreur de demande incorrecte
<code>lt</code>	<code>false</code>	HTTP 400 : erreur de demande incorrecte	HTTP 400 : erreur de demande incorrecte
<code>ge</code>	<code>false</code>	HTTP 400 : erreur de demande incorrecte	HTTP 400 : erreur de demande incorrecte
<code>le</code>	<code>false</code>	HTTP 400 : erreur de demande incorrecte	HTTP 400 : erreur de demande incorrecte
<code>eq</code>	<code>false</code>	<code>false</code>	<code>true</code>
<code>ne</code>	<code>true</code>	<code>true</code>	<code>false</code>

En résumé, `null` est uniquement égal à lui-même et n'est pas inférieur ou supérieur à toute autre valeur.

Si votre index comporte des champs de type `Edm.Double` et que vous chargez des valeurs `NaN` sur ces champs, vous devez en tenir compte lors de l'écriture des filtres. Recherche cognitive Azure implémente la norme IEEE 754 pour la gestion des valeurs `NaN`. Les comparaisons avec ces valeurs produisent des résultats non évidents, comme indiqué dans le tableau suivant.

OPÉRATEUR	RÉSULTAT LORSQU'AU MOINS UN OPÉRANDE EST <code>NAN</code>
<code>gt</code>	<code>false</code>
<code>lt</code>	<code>false</code>
<code>ge</code>	<code>false</code>
<code>le</code>	<code>false</code>
<code>eq</code>	<code>false</code>
<code>ne</code>	<code>true</code>

En résumé, `NaN` n'est pas égal à une autre valeur, y compris lui-même.

## Comparaison de données géospatiales

Vous ne pouvez pas comparer directement un champ de type `Edm.GeographyPoint` avec une valeur constante, mais vous pouvez utiliser la fonction `geo.distance`. Cette fonction retourne une valeur de type `Edm.Double`, ce qui vous permet de la comparer avec une valeur numérique constante et ainsi de filtrer en fonction de la distance qui la sépare des coordonnées géospatiales constantes. Consultez les [exemples](#) ci-dessous.

## Comparaison de données de chaîne

Les chaînes peuvent être comparées dans des filtres, en cas de correspondance exacte, à l'aide des opérateurs `eq` et `ne`. Ces comparaisons sont sensibles à la casse.

## Exemples

Correspondance des documents dans lesquels le champ `Rating` est compris entre 3 et 5 (inclusif) :

```
Rating ge 3 and Rating le 5
```

Correspondance des documents dans lesquels le champ `Location` est à moins de 2 kilomètres de la latitude et la longitude données :

```
geo.distance(Location, geography'POINT(-122.031577 47.578581)') lt 2.0
```

Correspondance des documents dans lesquels le champ `LastRenovationDate` est supérieur ou égal au 1er janvier 2015, 00:00 UTC :

```
LastRenovationDate ge 2015-01-01T00:00:00.000Z
```

Correspondance des documents dans lesquels le champ `Details/Sku` n'est pas `null` :

```
Details/Sku ne null
```

Correspondance des documents pour les hôtels comprenant au moins une chambre de type « Chambre de luxe » et où la chaîne du champ `Rooms/Type` correspond exactement au filtre :

```
Rooms/any(room: room/Type eq 'Deluxe Room')
```

## Étapes suivantes

- [Filtres dans la Recherche cognitive Azure](#)
- [Vue d'ensemble du langage d'expression OData pour Recherche cognitive Azure](#)
- [Informations de référence sur la syntaxe d'expression OData pour Recherche cognitive Azure](#)
- [Rechercher des documents \(API REST de la recherche cognitive Azure\)](#)

# Fonctions de recherche en texte intégral OData dans Recherche cognitive Azure- `search.ismatch` et `search.ismatchscoring`

04/10/2020 • 7 minutes to read • [Edit Online](#)

Recherche cognitive Azure prend en charge la recherche en texte intégral dans le contexte d'[expressions de filtre OData](#) via les fonctions `search.ismatch` et `search.ismatchscoring`. Ces fonctions permettent de combiner la recherche en texte intégral avec le filtrage booléen strict qui est impossible avec l'utilisation du paramètre `search` de niveau supérieur de l'[API de recherche](#).

## NOTE

Les fonctions `search.ismatch` et `search.ismatchscoring` sont uniquement prises en charge dans les filtres de l'[API de recherche](#). Elles ne sont pas prises en charge dans les API [Suggest](#) ou [Autocomplete](#).

## Syntaxe

L'extension EBNF suivante ([Extended Backus-Naur Form](#)) définit la grammaire des fonctions `search.ismatch` et `search.ismatchscoring` :

```
search_is_match_call ::=  
    'search.ismatch'('scoring')?(' search_is_match_parameters ')  
  
search_is_match_parameters ::=  
    string_literal(',', string_literal(',', query_type ',' search_mode)?)?  
  
query_type ::= "'full'" | "'simple'"  
  
search_mode ::= "'any'" | "'all'"
```

Un diagramme de syntaxe interactif est également disponible :

[Diagramme de syntaxe OData pour la Recherche cognitive Azure](#)

## NOTE

Consultez [Informations de référence sur la syntaxe d'expression OData pour Recherche cognitive Azure](#) pour l'extension EBNF complète.

## `search.ismatch`

La fonction `search.ismatch` évalue la requête de recherche en texte intégral comme une partie d'une expression de filtre. Les documents qui correspondent à la requête de recherche sont retournés dans le jeu de résultats. Les surcharges suivantes de cette fonction sont disponibles :

- `search.ismatch(search)`
- `search.ismatch(search, searchFields)`
- `search.ismatch(search, searchFields, queryType, searchMode)`

Les paramètres sont définis dans le tableau suivant :

NOM DU PARAMÈTRE	TYPE	DESCRIPTION
<code>search</code>	<code>Edm.String</code>	La requête de recherche (dans la syntaxe de requête Lucene <a href="#">simple</a> ou <a href="#">complète</a> ).
<code>searchFields</code>	<code>Edm.String</code>	Liste séparée par des virgules de champs de recherche où effectuer des recherches ; par défaut, tous les champs avec possibilité de recherche dans l'index. Lorsque vous utilisez la <a href="#">recherche par champ</a> dans le paramètre <code>search</code> , les spécificateurs de champ de la requête Lucene remplacent tous les champs spécifiés dans ce paramètre.
<code>queryType</code>	<code>Edm.String</code>	' <code>simple</code> ' ou ' <code>full</code> ' ; par défaut ' <code>simple</code> '. Spécifie le langage de requête utilisé dans le paramètre <code>search</code> .
<code>searchMode</code>	<code>Edm.String</code>	' <code>any</code> ' ou ' <code>all</code> ' ; par défaut ' <code>any</code> '. Indique si tout ou partie des termes de recherche du paramètre <code>search</code> doit correspondre pour que le document soit considéré comme une correspondance. Lorsque vous utilisez les <a href="#">opérateurs booléens Lucene</a> dans le paramètre <code>search</code> , ils ont priorité sur ce paramètre.

Tous les paramètres ci-dessus sont équivalents aux [paramètres de la requête de recherche de l'API de recherche](#) correspondante.

La fonction `search.ismatch` renvoie une valeur de type `Edm.Boolean`, qui vous permet de la composer avec d'autres sous-expressions de filtre à l'aide des [opérateurs logiques](#) booléens.

#### NOTE

Recherche cognitive Azure ne prend pas en charge l'utilisation de `search.ismatch` ou `search.ismatchscoring` à l'intérieur des expressions lambda. Cela signifie qu'il n'est pas possible d'écrire des filtres sur les collections d'objets qui peuvent mettre en corrélation les correspondances de recherche en texte intégral avec des correspondances de filtre strict sur le même objet. Pour plus d'informations sur cette restriction et pour obtenir des exemples, consultez [Résolution des problèmes de filtres de collection dans Recherche cognitive Azure](#). Pour en savoir plus sur la raison de l'existence de cette restriction, consultez [Présentation des filtres de collection dans Recherche cognitive Azure](#).

### Search.ismatchscoring

La fonction `search.ismatchscoring`, comme la fonction `search.ismatch`, renvoie `true` pour les documents qui correspondent à la requête de recherche en texte intégral transmise comme paramètre. La différence est que le score de pertinence des documents correspondant à la requête `search.ismatchscoring` contribue au score global du document, tandis que dans le cas de `search.ismatch`, le score du document n'est pas modifié. Les surcharges suivantes de cette fonction sont disponibles avec des paramètres identiques à ceux de `search.ismatch` :

- `search.ismatchscoring(search)`

- `search.ismatchscoring(search, searchFields)`
- `search.ismatchscoring(search, searchFields, queryType, searchMode)`

Cela signifie que les deux fonctions `search.ismatch` et `search.ismatchscoring` peuvent être utilisées dans la même expression de filtre.

## Exemples

Rechercher les documents avec le mot « `waterfront` ». Cette requête de filtre est identique à une [demande de recherche](#) avec `search=waterfront`.

```
search.ismatchscoring('waterfront')
```

Rechercher les documents avec le mot « `hostel` » et une évaluation supérieure ou égale à 4, ou les documents avec le mot « `motel` » et une évaluation égale à 5. Notez que cette demande ne peut pas être exprimée sans la fonction `search.ismatchscoring`.

```
search.ismatchscoring('hostel') and Rating ge 4 or search.ismatchscoring('motel') and Rating eq 5
```

Rechercher les documents sans le mot « `luxury` ».

```
not search.ismatch('luxury')
```

Rechercher les documents avec la phrase « `ocean view` » ou une évaluation égale à 5. La requête `search.ismatchscoring` sera exécutée seulement sur les champs `HotelName` et `Rooms/Description`.

Les documents qui correspondent uniquement à la deuxième clause de la disjonction sont également renvoyés (hôtels avec une `Rating` égale à 5). L'explication est que ces documents ne correspondent à aucune des parties avec score de l'expression : ils sont renvoyés avec un score égal à zéro.

```
search.ismatchscoring('"ocean view"', 'Rooms/Description,HotelName') or Rating eq 5
```

Rechercher les documents où les termes « `hotel` » et « `airport` » sont distants de 5 mots ou moins dans la description de l'hôtel, et où fumer n'est pas autorisé dans au moins certaines pièces. Cette requête utilise le [langage de requête Lucene complet](#).

```
search.ismatch('"hotel airport"~5', 'Description', 'full', 'any') and Rooms/any(room: not room/SmokingAllowed)
```

## Étapes suivantes

- [Filtres dans la Recherche cognitive Azure](#)
- [Vue d'ensemble du langage d'expression OData pour Recherche cognitive Azure](#)
- [Informations de référence sur la syntaxe d'expression OData pour Recherche cognitive Azure](#)
- [Rechercher des documents \(API REST de la recherche cognitive Azure\)](#)

# Fonctions géospatiales OData dans Recherche cognitive Azure - `geo.distance` et `geo.intersects`

04/10/2020 • 9 minutes to read • [Edit Online](#)

Recherche cognitive Azure prend en charge les requêtes géospatiales dans les [expressions de filtre OData](#) via les fonctions `geo.distance` et `geo.intersects`. La fonction `geo.distance` renvoie la distance en kilomètres entre deux points, l'un étant un champ ou une variable de portée et l'autre une constante passée comme partie du filtre. La fonction `geo.intersects` renvoie `true` si un point donné se trouve dans un polygone donné, où le point est un champ ou une variable de portée et où le polygone est spécifié sous la forme d'une constante passée comme partie du filtre.

La fonction `geo.distance` peut également être utilisée dans le paramètre `$orderby` pour trier les résultats de recherche par distance à partir d'un point donné. La syntaxe pour `geo.distance` dans `$orderby` est la même que dans `$filter`. Lors de l'utilisation de `geo.distance` dans `$orderby`, le champ auquel elle s'applique doit être de type `Edm.GeographyPoint` et être aussi **trieable**.

## NOTE

Si vous spécifiez `geo.distance` dans le paramètre `$orderby`, le champ que vous passez à la fonction doit contenir un seul point géographique. Autrement dit, il doit être de type `Edm.GeographyPoint` et pas de type `Collection(Edm.GeographyPoint)`. Le tri sur les champs de collection n'est pas possible dans Recherche cognitive Azure.

## Syntaxe

L'extension EBNF suivante ([Extended Backus-Naur Form](#)) définit la grammaire des fonctions `geo.distance` et `geo.intersects`, ainsi que les valeurs géospatiales sur lesquelles elles fonctionnent :

```
geo_distance_call ::=  
    'geo.distance(' variable ',' geo_point ')'  
    | 'geo.distance(' geo_point ',' variable ')'  
  
geo_point ::= "geography'POINT(" lon_lat ")"  
  
lon_lat ::= float_literal ' ' float_literal  
  
geo_intersects_call ::=  
    'geo.intersects(' variable ',' geo_polygon ')'  
  
/* You need at least four points to form a polygon, where the first and  
last points are the same. */  
geo_polygon ::=  
    "geography'POLYGON((" lon_lat ',' lon_lat ',' lon_lat ',' lon_lat_list "))'"  
  
lon_lat_list ::= lon_lat(' ' lon_lat)*
```

Un diagramme de syntaxe interactif est également disponible :

[Diagramme de syntaxe OData pour la Recherche cognitive Azure](#)

## NOTE

Consultez [Informations de référence sur la syntaxe d'expression OData pour Recherche cognitive Azure](#) pour l'extension EBNF complète.

## geo.distance

La fonction `geo.distance` prend deux paramètres de type `Edm.GeographyPoint` et renvoie une valeur `Edm.Double` qui correspond à la distance entre eux en kilomètres. Ceci diffère des autres services qui prennent en charge des opérations géospatiales OData, qui renvoient généralement les distances en mètres.

L'un des paramètres de `geo.distance` doit être une constante de point géographique et l'autre un chemin d'accès à un champ (ou une variable de portée dans le cas d'un filtre effectuant une itération sur un champ de type `Collection(Edm.GeographyPoint)`). Peu importe l'ordre de ces paramètres.

La constante de point géographique est au format `geography'POINT(<longitude> <latitude>)'`, où la longitude et latitude sont des constantes numériques.

## NOTE

Quand vous utilisez `ge` dans un filtre, vous devez comparer la distance renvoyée par la fonction avec une constante en utilisant `geo.distance`, `lt`, `le` ou `gt`. Les opérateurs `eq` et `ne` ne sont pas pris en charge lors de la comparaison de distances. Voici par exemple une utilisation correcte de `geo.distance` :

```
$filter=geo.distance(location, geography'POINT(-122.131577 47.678581)') le 5 .
```

## geo.intersects

La fonction `geo.intersects` prend une variable de type `Edm.GeographyPoint` et une constante `Edm.GeographyPolygon` et renvoie un `Edm.Boolean` -- `true` si le point se trouve dans les limites du polygone, `false` dans le cas contraire.

Le polygone est une surface en deux dimensions stockée sous la forme d'une séquence de points définissant un cadre englobant (voir [les exemples](#) ci-dessous). Le polygone doit être fermé, ce qui signifie que le premier point et le dernier doivent être le même. [Les points d'un polygone doivent être dans le sens antihoraire](#).

## Requêtes géospatiales et polygones couvrant le 180{1}e{2} méridien

Pour de nombreuses bibliothèques de requêtes géospatiales, la formulation d'une requête incluant le 180{1}e{2} méridien (près de la ligne de changement de date) est hors limites ou nécessite une solution de contournement, comme diviser le polygone en deux parties, une de chaque côté du méridien.

Dans Recherche cognitive Azure, les requêtes géospatiales incluant une longitude de 180 degrés fonctionnent normalement si la forme de la requête est rectangulaire et que vos coordonnées s'alignent sur une disposition en grille le long de la longitude et de la latitude (par exemple

```
geo.intersects(location, geography'POLYGON((179 65, 179 66, -179 66, -179 65, 179 65))'). Sinon, pour les formes non rectangulaires ou non alignées, envisagez l'approche par division du polygone.
```

## Fonctions géospatiales et `null`

Comme tous les autres champs qui ne sont pas des collections dans Recherche cognitive Azure, les champs de type `Edm.GeographyPoint` peuvent contenir des valeurs `null`. Lorsque Recherche cognitive Azure évalue `geo.intersects` pour un champ qui est `null`, le résultat sera toujours `false`. Le comportement de `geo.distance` dans ce cas dépend du contexte :

- Dans les filtres, la valeur `geo.distance` d'un champ `null` donne le résultat `null`. Cela signifie que le document ne correspond pas, car `null` comparé à n'importe quelle valeur non null a pour résultat `false`.
- Lors du tri des résultats à l'aide de `$orderby`, la valeur `geo.distance` d'un champ `null` donne pour résultat

la distance maximale possible. Les documents comportant un tel champ auront un niveau de tri inférieur par rapport aux autres lorsque le sens de tri `asc` est utilisé (par défaut) et supérieur à tous les autres utilisateurs lorsque le sens est `desc`.

## Exemples

### Exemples de filtres

Rechercher tous les hôtels dans les 10 kilomètres d'un point de référence donnée (où l'emplacement est un champ de type `Edm.GeographyPoint`) :

```
geo.distance(location, geography'POINT(-122.131577 47.678581)') le 10
```

Rechercher tous les hôtels dans une fenêtre d'affichage décrite sous forme de polygone (où l'emplacement est un champ de type `Edm.GeographyPoint`). Notez que le polygone est fermé (les définitions du premier et du dernier point doivent les mêmes) et [les points doivent être listés dans le sens antihoraire](#).

```
geo.intersects(location, geography'POLYGON((-122.031577 47.578581, -122.031577 47.678581, -122.131577 47.678581, -122.031577 47.578581))')
```

### Exemples de clause Order by

Trier les hôtels par ordre décroissant sur `rating`, puis par ordre croissant sur la distance à partir des coordonnées spécifiées :

```
rating desc,geo.distance(location, geography'POINT(-122.131577 47.678581)') asc
```

Trier les hôtels par ordre décroissant sur `search.score` et `rating`, puis par ordre croissant sur la distance à partir des coordonnées spécifiées, de sorte qu'entre deux d'hôtels avec une évaluation identique, le plus proche soit listé en premier :

```
search.score() desc, rating desc, geo.distance(location, geography'POINT(-122.131577 47.678581)') asc
```

## Étapes suivantes

- [Filtres dans la Recherche cognitive Azure](#)
- [Vue d'ensemble du langage d'expression OData pour Recherche cognitive Azure](#)
- [Informations de référence sur la syntaxe d'expression OData pour Recherche cognitive Azure](#)
- [Rechercher des documents \(API REST de la recherche cognitive Azure\)](#)

# Opérateurs logiques OData dans Recherche cognitive Azure - `and`, `or`, `not`

04/10/2020 • 6 minutes to read • [Edit Online](#)

Les [expressions de filtre OData](#) dans Recherche cognitive Azure sont des expressions booléennes qui s'évaluent à `true` ou `false`. Vous pouvez écrire un filtre complexe en rédigeant une série de [filtres plus simples](#) et en les composant des opérateurs logiques de l'[algèbre booléen](#) :

- `and` : Un opérateur binaire qui s'évalue à `true` si ses sous-expressions droites et gauches s'évaluent à `true`.
- `or` : Un opérateur binaire qui s'évalue à `true` si l'une de ses sous-expressions droite ou gauche s'évaluent à `true`.
- `not` : Un opérateur unaire qui s'évalue à `true` si sa sous-expression s'évalue à `false`, et vice versa.

Ces opérateurs, ainsi que les [opérateurs de collection](#) `any` et `all`, vous permettent de construire des filtres qui peuvent exprimer des critères de recherche complexes.

## Syntaxe

L'extension EBNF suivante ([Extended Backus-Naur Form](#)) définit la grammaire d'une expression OData utilisant les opérateurs logiques.

```
logical_expression ::=  
    boolean_expression ('and' | 'or') boolean_expression  
    | 'not' boolean_expression
```

Un diagramme de syntaxe interactif est également disponible :

[Diagramme de syntaxe OData pour la Recherche cognitive Azure](#)

### NOTE

Consultez [Informations de référence sur la syntaxe d'expression OData pour Recherche cognitive Azure](#) pour l'extension EBNF complète.

Il s'agit de deux formes d'expressions logiques : binaire (`and` / `or`), où il y a deux sous-expressions, et unaire (`not`), où il n'y en a qu'une. Les sous-expressions peuvent être des expressions booléennes de toutes sortes :

- Champs ou variables de plage de type `Edm.Boolean`
- Fonctions qui retournent des valeurs de type `Edm.Boolean`, telles que `geo.intersects` OU `search.ismatch`
- [Expressions de comparaison](#), telles que `rating gt 4`
- [Expressions de collection](#), telles que `Rooms/any(room: room/Type eq 'Deluxe Room')`
- Littéraux booléens `true` OU `false`.
- Autres expressions logiques construites à l'aide de `and`, `or` et `not`.

## IMPORTANT

Il y a certaines situations où tous les types de sous-expressions peuvent être utilisés avec `and` / `or`, particulièrement dans des expressions lambda. Consultez [Opérateurs de collection OData dans Recherche cognitive Azure](#) pour en savoir plus.

## Opérateurs logiques et `null`

La plupart des expressions booléennes, telles que les fonctions et les comparaisons, ne peuvent pas produire de valeurs `null`, et les opérateurs logiques ne peuvent pas être appliqués au littéral `null` directement (par exemple, `x and null` n'est pas autorisé). Toutefois, les champs booléens peuvent être `null`, vous devez donc être conscient du comportement des opérateurs `and`, `or` et `not` en présence de `null`. Ceci est résumé dans le tableau suivant, où `b` est un champ de type `Edm.Boolean` :

EXPRESSION	RÉSULTAT LORSQUE <code>b</code> EST <code>NULL</code>
<code>b</code>	<code>false</code>
<code>not b</code>	<code>true</code>
<code>b eq true</code>	<code>false</code>
<code>b eq false</code>	<code>false</code>
<code>b eq null</code>	<code>true</code>
<code>b ne true</code>	<code>true</code>
<code>b ne false</code>	<code>true</code>
<code>b ne null</code>	<code>false</code>
<code>b and true</code>	<code>false</code>
<code>b and false</code>	<code>false</code>
<code>b or true</code>	<code>true</code>
<code>b or false</code>	<code>false</code>

Lorsqu'un champ booléen `b` apparaît tout seul dans une expression de filtre, il se comporte comme s'il avait été écrit `b eq true`, donc si `b` est `null`, l'expression s'évalue à `false`. De même, `not b` se comporte comme `not (b eq true)`, donc elle s'évalue à `true`. Ainsi, les champs `null` se comportent de la même façon que `false`. Ceci est cohérent avec la manière dont ils se comportent lorsqu'ils sont combinés à d'autres expressions à l'aide de `and` et `or`, comme illustré dans le tableau ci-dessus. Malgré ça, une comparaison directe à `false` (`b eq false`) s'évalue toujours à `false`. En d'autres termes, `null` n'est pas égal à `false`, bien qu'il se comporte comme tel dans les expressions booléennes.

## Exemples

Correspondance des documents dans lesquels le champ `rating` est compris entre 3 et 5 (inclusif) :

```
rating ge 3 and rating le 5
```

Correspondance des documents dans lesquels tous les éléments du champ `ratings` sont inférieurs à 3 ou supérieurs à 5 :

```
ratings/all(r: r lt 3 or r gt 5)
```

Correspondance des documents dans lesquels le champ `location` est compris dans le polygone donné, et que le document ne contient pas le terme « public ».

```
geo.intersects(location, geography'POLYGON((-122.031577 47.578581, -122.031577 47.678581, -122.131577  
47.678581, -122.031577 47.578581))') and not search.ismatch('public')
```

Correspondance des documents pour les hôtels à Vancouver, au Canada, qui comprennent une chambre de luxe avec un prix de base inférieur à 160 :

```
Address/City eq 'Vancouver' and Address/Country eq 'Canada' and Rooms/any(room: room/Type eq 'Deluxe  
Room' and room/BaseRate lt 160)
```

## Étapes suivantes

- [Filtres dans la Recherche cognitive Azure](#)
- [Vue d'ensemble du langage d'expression OData pour Recherche cognitive Azure](#)
- [Informations de référence sur la syntaxe d'expression OData pour Recherche cognitive Azure](#)
- [Rechercher des documents \(API REST de la recherche cognitive Azure\)](#)

# Fonction `search.in` OData dans Recherche cognitive Azure

04/10/2020 • 7 minutes to read • [Edit Online](#)

Un scénario courant dans [Expressions de filtre OData](#) consiste à vérifier si un champ unique dans chaque document est égal à l'une des nombreuses valeurs possibles. Par exemple, voici comment certaines applications implémentent le [filtrage de sécurité](#) : en comparant un champ contenant un ou plusieurs ID de principal avec une liste d'ID de principal représentant l'utilisateur qui émet la requête. Une façon d'écrire une requête telle que celle-ci consiste à utiliser les opérateurs `eq` et `or` :

```
group_ids/any(g: g eq '123' or g eq '456' or g eq '789')
```

Toutefois, il existe une manière plus courte de l'écrire, à l'aide de la fonction `search.in` :

```
group_ids/any(g: search.in(g, '123, 456, 789'))
```

## IMPORTANT

En plus d'être plus courte et plus facile à lire, l'utilisation de `search.in` offre également des [avantages en termes de performances](#) et permet d'éviter certaines [limites de taille de filtres](#) lorsqu'il existe des centaines voire des milliers de valeurs à inclure dans le filtre. Pour cette raison, nous recommandons fortement d'utiliser `search.in` au lieu d'une disjonction plus complexe d'expressions d'égalité.

## NOTE

La version 4.01 du protocole OData standard a récemment introduit l'[opérateur `in`](#), qui a un comportement similaire à la fonction `search.in` dans Recherche cognitive Azure. Toutefois, Recherche cognitive Azure ne prend pas en charge cet opérateur, vous devez donc utiliser la fonction `search.in` à la place.

## Syntaxe

L'extension EBNF suivante ([Extended Backus-Naur Form](#)) définit la grammaire de la fonction `search.in` :

```
search_in_call ::=  
  'search.in(' variable ',' string_literal(',') string_literal)? ')'
```

Un diagramme de syntaxe interactif est également disponible :

[Diagramme de syntaxe OData pour la Recherche cognitive Azure](#)

## NOTE

Consultez [Informations de référence sur la syntaxe d'expression OData pour Recherche cognitive Azure](#) pour l'extension EBNF complète.

La fonction `search.in` teste si un champ de type chaîne donné ou une variable de portée est égal à une des valeurs d'une liste donnée. L'égalité entre la variable et chaque valeur de la liste est déterminée de manière à respecter la casse, de la même façon que pour l'opérateur `eq`. Ainsi, une expression comme `search.in(myfield, 'a, b, c')` équivaut à `myfield eq 'a' or myfield eq 'b' or myfield eq 'c'`, sauf que `search.in` aboutira à de bien meilleures performances.

Il existe deux surcharges de la fonction `search.in` :

- `search.in(variable, valueList)`
- `search.in(variable, valueList, delimiters)`

Les paramètres sont définis dans le tableau suivant :

NOM DU PARAMÈTRE	TYPÉ	DESCRIPTION
<code>variable</code>	<code>Edm.String</code>	Une référence du champ de type chaîne (ou une variable de portée sur un champ de collection de chaînes dans le cas où <code>search.in</code> est utilisé à l'intérieur d'une expression <code>any</code> ou <code>all</code> ).
<code>valueList</code>	<code>Edm.String</code>	Une chaîne contenant une liste délimitée de valeurs à mettre en correspondance le paramètre <code>variable</code> . Si le paramètre <code>delimiters</code> n'est pas spécifié, les séparateurs par défaut sont l'espace et la virgule.
<code>delimiters</code>	<code>Edm.String</code>	Une chaîne dans laquelle chaque caractère est traité comme un séparateur lors de l'analyse du paramètre <code>valueList</code> . La valeur par défaut de ce paramètre est <code>' , '</code> , ce qui signifie que toutes les valeurs avec des espaces et/ou des virgules entre elles seront séparées. Si vous devez utiliser des séparateurs autres que des espaces et des virgules dans le cas où vos valeurs incluent ces caractères, vous pouvez spécifier d'autres séparateurs, tels que <code>'   '</code> dans ce paramètre.

### Performances de `search.in`

Si vous utilisez `search.in`, vous pouvez vous attendre à des temps de réponse inférieurs à la seconde quand le deuxième paramètre contient une liste de plusieurs centaines ou plusieurs milliers de valeurs. Il n'existe pas de limite explicite quant au nombre d'éléments que vous pouvez passer à `search.in`, même si vous êtes néanmoins limité par la taille maximale de la demande. Cependant, la latence augmente en même temps que le nombre de valeurs.

## Exemples

Recherchez tous les hôtels avec un nom égal à « Sea View motel » ou « Budget hotel ». Les expressions contiennent des espaces, qui sont des séparateurs par défaut. Vous pouvez spécifier un autre séparateur entre guillemets simples comme troisième paramètre de chaîne :

```
search.in(HotelName, 'Sea View motel,Budget hotel', ',')
```

Recherchez tous les hôtels avec un nom égal à « Sea View motel » ou « Budget hotel » séparés par « | » :

```
search.in(HotelName, 'Sea View motel|Budget hotel', '|')
```

Recherchez tous les hôtels avec des chambres comportant les balises « wifi » ou « baignoire » :

```
Rooms/any(room: room/Tags/any(tag: search.in(tag, 'wifi, tub')))
```

Recherchez une correspondance entre des expressions tirées d'une collection, telles que « heated towel racks » ou « hairdryer included » dans les balises.

```
Rooms/any(room: room/Tags/any(tag: search.in(tag, 'heated towel racks,hairdryer included', ',')))
```

Recherchez tous les hôtels sans les balises « motel » ou « cabin » :

```
Tags/all(tag: not search.in(tag, 'motel, cabin'))
```

## Étapes suivantes

- [Filtres dans la Recherche cognitive Azure](#)
- [Vue d'ensemble du langage d'expression OData pour Recherche cognitive Azure](#)
- [Informations de référence sur la syntaxe d'expression OData pour Recherche cognitive Azure](#)
- [Rechercher des documents \(API REST de la recherche cognitive Azure\)](#)

# Fonction `search.score` OData dans Recherche cognitive Azure

04/10/2020 • 2 minutes to read • [Edit Online](#)

Lorsque vous envoyez une requête à Recherche cognitive Azure sans le paramètre `$orderby`, les résultats renvoyés sont triés par score décroissant de pertinence. Même quand vous utilisez `$orderby`, le score de pertinence servira à rompre les liens par défaut. Toutefois, il est parfois utile d'utiliser le score de pertinence comme critère de tri initial, et d'utiliser d'autres critères comme critères décisifs. La fonction `search.score` vous permet de le faire.

## Syntaxe

La syntaxe pour `search.score` dans `$orderby` est `search.score()`. La fonction `search.score` ne prend aucun paramètre. Elle peut être utilisée avec le spécificateur d'ordre de tri `asc` ou `desc`, tout comme les autres clauses dans le paramètre `$orderby`. Il peut apparaître n'importe où dans la liste des critères de tri.

## Exemple

Trier les hôtels par ordre décroissant sur `search.score` et `rating`, puis par ordre croissant sur la distance à partir des coordonnées spécifiées, de sorte qu'entre deux d'hôtels avec une évaluation identique, le plus proche soit listé en premier :

```
search.score() desc, rating desc, geo.distance(location, geography'POINT(-122.131577 47.678581)') asc
```

## Étapes suivantes

- [Vue d'ensemble du langage d'expression OData pour Recherche cognitive Azure](#)
- [Informations de référence sur la syntaxe d'expression OData pour Recherche cognitive Azure](#)
- [Rechercher des documents \(API REST Recherche cognitive Azure\)](#)

# Informations de référence sur la syntaxe d'expression OData pour la Recherche cognitive Azure

04/10/2020 • 5 minutes to read • [Edit Online](#)

La Recherche cognitive Azure utilise les [expressions OData](#) comme paramètres dans l'ensemble de l'API. En règle générale, les expressions OData sont utilisées pour les paramètres `$orderby` et `$filter`. Ces expressions peuvent être complexes et contenir plusieurs clauses, fonctions et opérateurs. Toutefois, les expressions OData telles que les chemins de propriété sont utilisées dans de nombreuses parties de l'API REST de Recherche cognitive Azure, y compris quand elles sont simples. Par exemple, les expressions de chemin d'accès sont utilisées pour faire référence à des sous-champs de champs complexes dans l'ensemble de l'API. C'est notamment le cas lors du référencement de sous-champs dans un [suggesteur](#), une [fonction de scoring](#), le paramètre `$select` ainsi que dans les [recherches par champ dans les requêtes Lucene](#).

Cet article décrit toutes ces formes d'expressions OData qui utilisent une grammaire formelle. Il existe également un [diagramme interactif](#) permettant de faciliter l'exploration visuelle de la grammaire.

## Grammaire formelle

Nous pouvons décrire le sous-ensemble du langage OData pris en charge par la Recherche cognitive Azure à l'aide d'une grammaire EBNF ([Extended Backus-Naur Form](#), ou Forme de Backus-Naur étendue). Les règles sont triées dans l'ordre « décroissant », en commençant par les expressions plus complexes qui sont ensuite décomposées en expressions plus primitives. Les règles de grammaire qui correspondent aux paramètres spécifiques de l'API REST de Recherche cognitive Azure sont situées dans la partie supérieure :

- `$filter` : défini par la règle `filter_expression`.
- `$orderby` : défini par la règle `order_by_expression`.
- `$select` : défini par la règle `select_expression`.
- Chemins d'accès des champs : définis par la règle `field_path`. Les chemins d'accès des champs sont utilisés dans l'ensemble de l'API. Ils peuvent faire référence aux champs de niveau supérieur d'un index ou à des sous-champs présentant un ou plusieurs [champs complexes](#) comme ancêtres.

Après l'EBNF, vous trouverez un [diagramme de syntaxe](#) dans lequel vous pouvez naviguer afin d'explorer la grammaire et les relations entre ses règles de manière interactive.

```
/* Top-level rules */

filter_expression ::= boolean_expression

order_by_expression ::= order_by_clause(',') order_by_clause)*

select_expression ::= '*' | field_path(',') field_path)*

field_path ::= identifier('/'identifier)*

/* Shared base rules */

identifier ::= [a-zA-Z_][a-zA-Z_0-9]*
```

```

/* Rules for $orderby */

order_by_clause ::= (field_path | sortable_function) ('asc' | 'desc')?

sortable_function ::= geo_distance_call | 'search.score()'

/* Rules for $filter */

boolean_expression ::=
    collection_filter_expression
  | logical_expression
  | comparison_expression
  | boolean_literal
  | boolean_function_call
  | '(' boolean_expression ')'
  | variable

/* This can be a range variable in the case of a lambda, or a field path. */
variable ::= identifier | field_path

collection_filter_expression ::=
    field_path'/all(' lambda_expression ')
  | field_path'/any(' lambda_expression ')
  | field_path'/any()

lambda_expression ::= identifier ':' boolean_expression

logical_expression ::=
    boolean_expression ('and' | 'or') boolean_expression
  | 'not' boolean_expression

comparison_expression ::=
    variable_or_function comparison_operator constant |
    constant comparison_operator variable_or_function

variable_or_function ::= variable | function_call

comparison_operator ::= 'gt' | 'lt' | 'ge' | 'le' | 'eq' | 'ne'

/* Rules for constants and literals */

constant ::=
    string_literal
  | date_time_offset_literal
  | integer_literal
  | float_literal
  | boolean_literal
  | 'null'

string_literal ::= """([^\"] | "")*"""

date_time_offset_literal ::= date_part'T' time_part time_zone

date_part ::= year'-'month'-'day

time_part ::= hour':'minute(':'second('.fractional_seconds)?)

zero_to_fifty_nine ::= [0-5]digit

digit ::= [0-9]

year ::= digit digit digit digit

month ::= '0'[1-9] | '1'[0-2]

```

```

day ::= '0'[1-9] | [1-2]digit | '3'[0-1]

hour ::= [0-1]digit | '2'[0-3]

minute ::= zero_to_fifty_nine

second ::= zero_to_fifty_nine

fractional_seconds ::= integer_literal

time_zone ::= 'Z' | sign hour':minute

sign ::= '+' | '-'

/* In practice integer literals are limited in length to the precision of
the corresponding EDM data type. */
integer_literal ::= digit+

float_literal ::=
    sign? whole_part fractional_part? exponent?
    | 'NaN'
    | '-INF'
    | 'INF'

whole_part ::= integer_literal

fractional_part ::= '.'integer_literal

exponent ::= 'e' sign? integer_literal

boolean_literal ::= 'true' | 'false'

/* Rules for functions */

function_call ::=
    geo_distance_call |
    boolean_function_call

geo_distance_call ::=
    'geo.distance(' variable ',' geo_point ')'
    | 'geo.distance(' geo_point ',' variable ')'

geo_point ::= "geography'POINT(" lon_lat ")"

lon_lat ::= float_literal ' ' float_literal

boolean_function_call ::=
    geo_intersects_call |
    search_in_call |
    search_is_match_call

geo_intersects_call ::=
    'geo.intersects(' variable ',' geo_polygon ')'

/* You need at least four points to form a polygon, where the first and
last points are the same. */
geo_polygon ::= "geography'POLYGON((" lon_lat ',' lon_lat ',' lon_lat ',' lon_lat_list "))"

lon_lat_list ::= lon_lat(',') lon_lat)*

search_in_call ::=
    'search.in(' variable ',' string_literal(',') string_literal)? ')'

/* Note that it is illegal to call search.ismatch or search.ismatchscoring
from inside a lambda expression. */
search_is_match_call ::=
    'search.ismatch(' scoring')?' '(' search is match parameters ')'

```

```
search_is_match_parameters ::=  
    string_literal(',' string_literal(',' query_type ',' search_mode)?)?  
  
query_type ::= "'full'" | "'simple'"  
  
search_mode ::= "'any'" | "'all'"
```

## Diagramme de syntaxe

Pour explorer visuellement la grammaire du langage OData pris en charge par la Recherche cognitive Azure, vous pouvez vous aider du diagramme de syntaxe interactif :

[Diagramme de syntaxe OData pour la Recherche cognitive Azure](#)

## Voir aussi

- [Filtres dans la Recherche cognitive Azure](#)
- [Rechercher des documents \(API REST de la recherche cognitive Azure\)](#)
- [Syntaxe de requête Lucene](#)
- [Syntaxe de requête simple dans la Recherche cognitive Azure](#)

# Compétences cognitives intégrées pour le traitement de texte et d'images pendant l'indexation (Recherche cognitive Azure)

04/10/2020 • 6 minutes to read • [Edit Online](#)

Dans cet article, vous découvrirez les compétences cognitives fournies avec Recherche cognitive Azure et que vous pouvez inclure dans un ensemble de compétences afin d'extraire le contenu et la structure. Une *compétence cognitive* est un module ou une opération qui transforme du contenu d'une certaine façon. Souvent, il s'agit d'un composant qui extrait des données ou déduit une structure, renforçant ainsi notre compréhension des données d'entrée. Presque toujours, la sortie est basée sur du texte. Un *ensemble de compétences* est la collection des compétences qui définissent le pipeline d'enrichissement.

## NOTE

Si vous élargissez le champ en augmentant la fréquence des traitements, en ajoutant des documents supplémentaires ou en ajoutant plusieurs algorithmes d'IA, vous devez [attacher une ressource Cognitive Services facturable](#). Des frais s'appliquent durant l'appel des API dans Cognitive Services ainsi que pour l'extraction d'images dans le cadre de la phase de craquage de document de la Recherche cognitive Azure. L'extraction de texte à partir des documents est gratuite.

L'exécution des compétences intégrées est facturée au prix actuel du [paiement à l'utilisation de Cognitive Services](#). Les prix appliqués pour l'extraction d'images sont présentés sur la [page de tarification du service Recherche cognitive Azure](#).

La fonctionnalité [d'enrichissement incrémentiel \(préversion\)](#) permet de fournir un cache qui augmente l'efficacité de l'indexeur en exécutant uniquement les compétences cognitives nécessaires en cas de modification ultérieure de l'ensemble de compétences, ce qui représente un gain de temps et d'argent.

## Compétences prédéfinies

Diverses compétences sont flexibles du point de vue de ce qu'elles consomment ou produisent. En règle générale, la plupart des compétences sont basées sur des modèles préformés, ce qui signifie que vous ne pouvez pas effectuer l'apprentissage du modèle à l'aide de vos propres données d'apprentissage. Le tableau suivant énumère et décrit les compétences fournies par Microsoft.

COMPÉTENCE	DESCRIPTION
<a href="#">Microsoft.Skills.Text.CustomEntityLookupSkill</a>	Recherche du texte dans une liste personnalisée définie par l'utilisateur de mots et d'expressions.
<a href="#">Microsoft.Skills.Text.KeyPhraseSkill</a>	Cette compétence utilise un modèle préformé pour détecter des phrases importantes en fonction de la position du terme, des règles linguistiques, de la proximité d'autres termes et du caractère inhabituel du terme dans la source de données.

COMPÉTENCE	DESCRIPTION
<a href="#">Microsoft.Skills.Text.LanguageDetectionSkill</a>	Cette compétence utilise un modèle préformé pour détecter la langue utilisée (un ID de langue par document). Quand plusieurs langues sont utilisées dans les mêmes segments de texte, la sortie est le LCID de la langue utilisée principalement.
<a href="#">Microsoft.Skills.Text.MergeSkill</a>	Consolide en un champ unique du texte issu d'une collection de champs.
<a href="#">Microsoft.Skills.Text.EntityRecognitionSkill</a>	Cette compétence utilise un modèle préentraîné pour établir des entités pour un ensemble fixe de catégories : personnes, emplacement, organisation, e-mails, URL, champs d'horodatage.
<a href="#">Microsoft.Skills.Text.PIIDetectionSkill</a>	Cette compétence utilise un modèle préformé pour extraire des informations d'identification personnelle d'un texte donné. Elle offre également différentes options pour masquer les entités d'informations d'identification personnelle dans le texte.
<a href="#">Microsoft.Skills.Text.SentimentSkill</a>	Cette compétence utilise un modèle préformé pour évaluer un sentiment positif ou négatif enregistrement par enregistrement. Le score est compris entre 0 et 1. Les scores neutres se produisent dans les cas Null où aucun sentiment n'est détectable et quand le texte est considéré comme neutre.
<a href="#">Microsoft.Skills.Text.SplitSkill</a>	Fractionne le texte en pages afin que vous puissiez enrichir ou augmenter le contenu de façon incrémentielle.
<a href="#">Microsoft.Skills.Text.TranslationSkill</a>	Cette qualification utilise un modèle préformé pour traduire le texte d'entrée en plusieurs langues pour les cas d'usage de normalisation ou de localisation.
<a href="#">Microsoft.Skills.Vision.ImageAnalysisSkill</a>	Cette compétence utilise un algorithme de détection d'image pour identifier le contenu d'une image et générer un texte descriptif.
<a href="#">Microsoft.Skills.Vision.OcrSkill</a>	Reconnaissance optique de caractères.
<a href="#">Microsoft.Skills.Util.ConditionalSkill</a>	Permet le filtrage, l'attribution d'une valeur par défaut et la fusion de données selon une condition.
<a href="#">Microsoft.Skills.Util.DocumentExtractionSkill</a>	Extrait le contenu d'un fichier dans le pipeline d'enrichissement.
<a href="#">Microsoft.Skills.Util.ShaperSkill</a>	Mappe la sortie sur un type complexe (type de données de plusieurs parties, qui peut être utilisé pour un nom complet, une adresse sur plusieurs lignes ou une combinaison d'un nom et d'un identificateur personnel.)
<a href="#">Microsoft.Skills.Custom.WebApiSkill</a>	Permet l'extensibilité du pipeline d'enrichissement de l'intelligence artificielle en passant un appel HTTP dans une API web personnalisée

COMPÉTENCE	DESCRIPTION
<a href="#">Microsoft.Skills.Custom.AmlSkill</a>	Permet l'extensibilité d'un pipeline d'enrichissement d'intelligence artificielle avec un modèle Azure Machine Learning

Pour obtenir des conseils sur la création d'une [compétence personnalisée](#), consultez [Guide pratique pour définir une interface personnalisée](#) et [Exemple : Création d'une compétence personnalisée pour l'enrichissement de l'IA](#).

## Voir aussi

- [Guide pratique pour définir un ensemble de compétences](#)
- [Définition d'une interface de compétences personnalisée](#)
- [Tutoriel : Indexation enrichie avec l'IA](#)

# Compétence cognitive conditionnelle

04/10/2020 • 7 minutes to read • [Edit Online](#)

La compétence **conditionnelle** permet des scénarios de Recherche cognitive Azure qui nécessitent une opération booléenne pour déterminer les données à affecter à une sortie. Ces scénarios incluent le filtrage, l'attribution d'une valeur par défaut et la fusion de données selon une condition.

Le pseudo-code suivant illustre ce que réalise la compétence conditionnelle :

```
if (condition)
    { output = whenTrue }
else
    { output = whenFalse }
```

## NOTE

Cette compétence n'est pas liée à une API Azure Cognitive Services et son utilisation ne vous est pas facturée. Toutefois, vous devez toujours [joindre une ressource Cognitive Services](#) pour remplacer l'option de ressource « Gratuit » qui vous limite à un petit nombre d'enrichissements quotidiens.

## @odata.type

Microsoft.Skills.Util.ConditionalSkill

## Champs évalués

Cette compétence est spéciale, car ses entrées sont des champs évalués.

Les éléments suivants sont des valeurs valides d'une expression :

- Chemins d'accès d'annotation (les chemins d'accès dans les expressions doivent être délimités par "\$(" et ")").

Exemples :

```
"= $(/document)"
 "= $(/document/content)"
```

- Littéraux (chaînes, nombres, true, false, null)

Exemples :

```
"= 'this is a string'"    // string (note the single quotation marks)
 "= 34"                  // number
 "= true"                // Boolean
 "= null"                // null value
```

- Expressions qui utilisent des opérateurs de comparaison (==, !=, >=, >, <=, <)

Exemples :

```

"= $(/document/language) == 'en'"
"= $(/document/sentiment) >= 0.5"

```

- Expressions qui utilisent des opérateurs booléens (`&&`, `||`, `!`, `^`)

Exemples :

```

"= $(/document/language) == 'en' && $(/document/sentiment) > 0.5"
"= !true"

```

- Expressions qui utilisent des opérateurs numériques (`+`, `-`, `*`, `/`, `%`)

Exemples :

```

"= $(/document/sentiment) + 0.5"           // addition
"= $(/document/totalValue) * 1.10"          // multiplication
"= $(/document/lengthInMeters) / 0.3049"     // division

```

Étant donné que la compétence conditionnelle prend en charge l'évaluation, vous pouvez l'utiliser dans les scénarios de transformation mineure. Pour obtenir un exemple, consultez la section [Définition de compétence 4](#).

## Entrées de la compétence

Les entrées respectent la casse.

ENTRÉE	DESCRIPTION
condition	<p>Cette entrée est un <a href="#">champ évalué</a> qui représente la condition à évaluer. Cette condition doit être évaluée selon une valeur booléenne (<code>true</code> ou <code>false</code>).</p> <p>Exemples :</p> <pre> "= true" "= \$(/document/language) == 'fr'" "= \$(/document/pages/*/language) == \$(/document/expectedLanguage)" </pre>
whenTrue	<p>Cette entrée est un <a href="#">champ évalué</a> qui représente la valeur à renvoyer si la condition est évaluée sur <code>true</code>. Les chaînes constantes doivent être renvoyées entre guillemets simples (' et ').</p> <p>Exemples de valeurs :</p> <pre> "= 'contract'" "= \$(/document/contractType)" "= \$(/document/entities/*)" </pre>
whenFalse	<p>Cette entrée est un <a href="#">champ évalué</a> qui représente la valeur à renvoyer si la condition est évaluée sur <code>false</code>.</p> <p>Exemples de valeurs :</p> <pre> "= 'contract'" "= \$(/document/contractType)" "= \$(/document/entities/*)" </pre>

## Sorties de la compétence

Il existe une seule sortie appelée simplement "output." Elle renvoie la valeur `whenFalse` si la condition est false ou `whenTrue` si la condition est true.

## Exemples

### Exemple de définition de compétence 1: Filtrer des documents pour renvoyer uniquement les documents français

La sortie suivante renvoie un tableau de phrases ("document/frenchSentences") si la langue du document est le français. Si la langue n'est pas le français, la valeur est définie sur *null*.

```
{  
    "@odata.type": "#Microsoft.Skills.Util.ConditionalSkill",  
    "context": "/document",  
    "inputs": [  
        { "name": "condition", "source": "= $(/document/language) == 'fr'" },  
        { "name": "whenTrue", "source": "/document/sentences" },  
        { "name": "whenFalse", "source": "= null" }  
    ],  
    "outputs": [ { "name": "output", "targetName": "frenchSentences" } ]  
}
```

Si "/document/frenchSentences" est utilisé comme *contexte* d'une autre compétence, cette compétence s'exécute uniquement si "/document/frenchSentences" n'est pas défini sur *null*.

### Exemple de définition de compétence 2 : Définir une valeur par défaut pour une valeur qui n'existe pas

La sortie suivante crée une annotation ("document/languageWithDefault") qui est définie sur la langue du document ou sur "es" si la langue n'est pas définie.

```
{  
    "@odata.type": "#Microsoft.Skills.Util.ConditionalSkill",  
    "context": "/document",  
    "inputs": [  
        { "name": "condition", "source": "= $(/document/language) == null" },  
        { "name": "whenTrue", "source": "= 'es'" },  
        { "name": "whenFalse", "source": "= $(/document/language)" }  
    ],  
    "outputs": [ { "name": "output", "targetName": "languageWithDefault" } ]  
}
```

### Exemple de définition de compétence 3 : Fusionner les valeurs de deux champs dans un même champ

Dans cet exemple, certaines phrases ont une propriété *frenchSentiment*. Chaque fois que la propriété *frenchSentiment* est null, nous souhaitons utiliser la valeur *englishSentiment*. Nous attribuons la sortie à un membre qui est appelé *sentiment* ("document/sentiment/\*/sentiment").

```
{  
    "@odata.type": "#Microsoft.Skills.Util.ConditionalSkill",  
    "context": "/document/sentences/*",  
    "inputs": [  
        { "name": "condition", "source": "= $(/document/sentences/*/frenchSentiment) == null" },  
        { "name": "whenTrue", "source": "/document/sentences/*/englishSentiment" },  
        { "name": "whenFalse", "source": "/document/sentences/*/frenchSentiment" }  
    ],  
    "outputs": [ { "name": "output", "targetName": "sentiment" } ]  
}
```

## Exemples de transformation

### Exemple de définition de compétence 4 : Transformation des données sur un seul champ

Dans cet exemple, nous recevons un *sentiment* compris entre 0 et 1. Nous voulons le transformer pour qu'il soit compris entre -1 et 1. Nous pouvons utiliser la compétence conditionnelle pour effectuer cette transformation

mineure.

Dans cet exemple, nous n'utilisons pas l'aspect conditionnel de la compétence, car la condition est toujours *true*.

```
{  
    "@odata.type": "#Microsoft.Skills.Util.ConditionalSkill",  
    "context": "/document/sentences/*",  
    "inputs": [  
        { "name": "condition", "source": "= true" },  
        { "name": "whenTrue", "source": "= ${/document/sentences/*/sentiment} * 2 - 1" },  
        { "name": "whenFalse", "source": "= 0" }  
    ],  
    "outputs": [ { "name": "output", "targetName": "normalizedSentiment" } ]  
}
```

## Considérations spéciales

Certains paramètres sont évalués. Vous devez donc suivre le modèle documenté avec une attention particulière. Les expressions doivent commencer par un signe égal. Un chemin d'accès doit être délimité par "\$(" et ")". Veillez à placer les chaînes entre guillemets simples. Ceci aide l'évaluateur à faire la distinction entre les chaînes et les opérateurs et chemins d'accès réels. En outre, assurez-vous d'ajouter un espace blanc autour des opérateurs (par exemple, un "\*" dans un chemin d'accès signifie autre chose qu'une multiplication).

## Étapes suivantes

- [Compétences prédéfinies](#)
- [Guide pratique pour définir un ensemble de compétences](#)

# Compétence cognitive Recherche d'entité personnalisée (préversion)

04/10/2020 • 17 minutes to read • [Edit Online](#)

## IMPORTANT

Cette compétence est actuellement en préversion publique. Les fonctionnalités en préversion sont fournies sans contrat de niveau de service et ne sont pas recommandées pour les charges de travail de production. Pour plus d'informations, consultez [Conditions d'Utilisation Supplémentaires relatives aux Évaluations Microsoft Azure](#). Il n'y a actuellement pas de prise en charge du portail ou du SDK .NET.

La compétence **Recherche d'entité personnalisée** recherche du texte dans une liste de mots et d'expressions personnalisée et définie par l'utilisateur. À l'aide de cette liste, elle étiquète tous les documents contenant des entités correspondantes. La compétence prend également en charge un degré de correspondance approximative qui peut être appliquée pour rechercher des correspondances similaires sans être rigoureusement exactes.

Cette compétence n'est pas liée à une API Cognitive Services et peut être utilisée gratuitement pendant toute la durée de la préversion. Vous devez néanmoins [joindre une ressource Cognitive Services](#) pour passer outre la limite d'enrichissement quotidienne. La limite quotidienne s'applique à l'accès gratuit à Cognitive Services via Recherche cognitive Azure.

## @odata.type

Microsoft.Skills.Text.CustomEntityLookupSkill

## Limites de données

- La taille maximale prise en charge pour les enregistrements d'entrée est de 256 Mo. Si vous devez subdiviser vos données avant de les envoyer à la compétence de recherche d'entité personnalisée, utilisez la [compétence Fractionnement de texte](#).
- La taille maximale prise en charge pour la table de définition des entités est de 10 Mo si elle est fournie à l'aide du paramètre *entitiesDefinitionUri*.
- Si les entités sont définies au format inline, à l'aide du paramètre *inlineEntitiesDefinition*, la taille maximale prise en charge est de 10 Ko.

## Paramètres de la compétence

Les paramètres respectent la casse.

NOM DU PARAMÈTRE	DESCRIPTION
<code>entitiesDefinitionUri</code>	Chemin d'accès à un fichier JSON ou CSV contenant tout le texte cible à comparer. Cette définition d'entité est lue au début de l'exécution d'un indexeur ; les mises à jour de ce fichier à mi-parcours ne seront pas réalisées avant l'exécution suivante. Cette configuration doit être accessible via HTTPS. Reportez-vous à la section Format de <a href="#">définition d'entité personnalisée</a> ci-dessous pour en savoir plus sur le schéma CSV ou JSON attendu.

NOM DU PARAMÈTRE	DESCRIPTION
<code>inlineEntitiesDefinition</code>	Définitions d'entités JSON au format inline. Ce paramètre remplace le paramètre entitiesDefinitionUri s'il est présent. Un maximum de 10 Ko de configuration peut être fourni au format inline. Reportez-vous à la section <a href="#">Définition d'entité personnalisée</a> ci-dessous pour en savoir plus sur le schéma JSON attendu.
<code>defaultLanguageCode</code>	(Facultatif) Code langue du texte d'entrée utilisé pour « tokeniser » et délimiter le texte. Langues prises en charge : <code>da, de, en, es, fi, fr, it, ko, pt</code> . La langue par défaut est l'anglais ( <code>en</code> ). Si vous utilisez un format codelangue-codepays, seule la partie codelangue du format est utilisée.

## Entrées de la compétence

NOM D'ENTRÉE	DESCRIPTION
<code>text</code>	Texte à analyser.
<code>languageCode</code>	facultatif. La valeur par défaut est <code>"en"</code> .

## Sorties de la compétence

NOM DE SORTIE	DESCRIPTION

NOM DE SORTIE	DESCRIPTION
entities	<p>Groupe d'objets contenant des informations sur les correspondances trouvées et les métadonnées associées. Chacune des entités identifiées peut contenir les champs suivants :</p> <ul style="list-style-type: none"> <li>• <i>nom</i> : entité de niveau supérieur identifiée. L'entité représente la forme « normalisée ».</li> <li>• <i>ID</i> : identificateur unique de l'entité, tel que défini par l'utilisateur dans le « Format de définition d'entité personnalisée ».</li> <li>• <i>description</i> : description de l'entité telle que définie par l'utilisateur dans le « Format de définition d'entité personnalisée ».</li> <li>• <i>type</i> : type de l'entité tel que défini par l'utilisateur dans le « Format de définition d'entité personnalisée ».</li> <li>• <i>subtype</i> : sous-type de l'entité, tel que défini par l'utilisateur dans le « Format de définition d'entité personnalisée ».</li> <li>• <i>matches</i> : collection qui décrit chacune des correspondances de cette entité sur le texte source. Chaque correspondance comportera les membres suivants : <ul style="list-style-type: none"> <li>• <i>text</i> : correspondance en texte brut issue du document source.</li> <li>• <i>offset</i> : emplacement où la correspondance a été trouvée dans le texte.</li> <li>• <i>length</i> : longueur du texte de la correspondance.</li> <li>• <i>matchDistance</i> : nombre de caractères séparant cette correspondance du nom ou de l'alias de l'entité d'origine.</li> </ul> </li> </ul>

## Format de définition d'entité personnalisée

La liste des entités personnalisées peut être fournie à la compétence Recherche d'entité personnalisée de trois façons différentes. Vous pouvez fournir la liste dans un fichier .CSV, un fichier JSON ou une définition inline dans le cadre de la définition de la compétence.

Si le fichier de définition est un fichier .CSV ou .JSON, le chemin d'accès à celui-ci doit être fourni dans le paramètre *entitiesDefinitionUri*. Dans ce cas, le fichier est téléchargé une fois au début de chaque exécution de l'indexeur. Le fichier doit être accessible aussi longtemps que l'indexeur est censé s'exécuter. En outre, le fichier doit être encodé en UTF-8.

Si la définition est fournie au format inline, elle doit être fournie au format inline comme contenu du paramètre de compétence *inlineEntitiesDefinition*.

### Format CSV

Vous pouvez fournir la définition des entités personnalisées à rechercher dans un fichier CSV (Comma-Separated Value) en indiquant le chemin d'accès à celui-ci et en le définissant dans le paramètre de compétence *entitiesDefinitionUri*. Le chemin d'accès doit correspondre à un emplacement https. La taille du fichier de définition ne doit pas dépasser 10 Mo.

Le format CSV est simple. Chaque ligne représente une entité unique, comme illustré ci-dessous :

```
Bill Gates, BillG, William H. Gates
Microsoft, MSFT
Satya Nadella
```

Trois entités peuvent ici être renvoyées en tant qu'entités trouvées (Bill Gates, Satya Nadella, Microsoft), mais elles seront identifiées si l'un des termes de la ligne (alias) correspond dans le texte. Par exemple, si la chaîne « William H. Gates » est trouvée dans un document, une correspondance de l'entité « Bill Gates » sera renvoyée.

## Format JSON

Vous pouvez également fournir la définition des entités personnalisées à rechercher dans un fichier JSON. Le format JSON vous offre un peu plus de flexibilité, car il vous permet de définir des règles de correspondance « par terme ». Par exemple, vous pouvez spécifier la distance de correspondance approximative (distance de Damerau-Levenshtein) pour chaque terme, ou indiquer si la correspondance doit respecter la casse ou non.

Comme avec les fichiers CSV, vous devez fournir le chemin d'accès au fichier JSON et le définir dans le paramètre de compétence *entitiesDefinitionUri*. Le chemin d'accès doit correspondre à un emplacement https. La taille du fichier de définition ne doit pas dépasser 10 Mo.

La définition de liste d'entités personnalisées JSON la plus élémentaire peut être une liste d'entités à comparer :

```
[  
  {  
    "name" : "Bill Gates"  
  },  
  {  
    "name" : "Microsoft"  
  },  
  {  
    "name" : "Satya Nadella"  
  }  
]
```

Un exemple plus complexe de définition JSON peut éventuellement fournir l'ID, la description, le type et le sous-type de chaque entité - ainsi que d'autres *alias*. Si une correspondance est trouvée avec un terme d'alias, l'entité est également renvoyée :

```
[
  {
    "name" : "Bill Gates",
    "description" : "Microsoft founder." ,
    "aliases" : [
      { "text" : "William H. Gates", "caseSensitive" : false },
      { "text" : "BillG", "caseSensitive" : true }
    ]
  },
  {
    "name" : "Xbox One",
    "type": "Hardware",
    "subtype" : "Gaming Device",
    "id" : "4e36bf9d-5550-4396-8647-8e43d7564a76",
    "description" : "The Xbox One product"
  },
  {
    "name" : "LinkedIn" ,
    "description" : "The LinkedIn company",
    "id" : "differentIdentifyingScheme123",
    "fuzzyEditDistance" : 0
  },
  {
    "name" : "Microsoft" ,
    "description" : "Microsoft Corporation",
    "id" : "differentIdentifyingScheme987",
    "defaultCaseSensitive" : false,
    "defaultFuzzyEditDistance" : 1,
    "aliases" : [
      { "text" : "MSFT", "caseSensitive" : true }
    ]
  }
]
```

Les tableaux ci-dessous décrivent plus en détail les différents paramètres de configuration que vous pouvez définir lors de la définition des entités à comparer :

NOM DU CHAMP	DESCRIPTION
<code>name</code>	descripteur d'entité de niveau supérieur. Dans la sortie de la compétence, les correspondances seront regroupées en fonction de ce nom. Il s'agit de la forme « normalisée » du texte trouvé.
<code>description</code>	(Facultatif) Ce champ peut être utilisé pour transmettre des métadonnées personnalisées au sujet du(des) texte(s) comparés. La valeur de ce champ apparaîtra avec chaque correspondance de son entité dans la sortie de la compétence.
<code>type</code>	(Facultatif) Ce champ peut être utilisé pour transmettre des métadonnées personnalisées au sujet du(des) texte(s) comparés. La valeur de ce champ apparaîtra avec chaque correspondance de son entité dans la sortie de la compétence.
<code>subtype</code>	(Facultatif) Ce champ peut être utilisé pour transmettre des métadonnées personnalisées au sujet du(des) texte(s) comparés. La valeur de ce champ apparaîtra avec chaque correspondance de son entité dans la sortie de la compétence.

NOM DU CHAMP	DESCRIPTION
<code>id</code>	(Facultatif) Ce champ peut être utilisé pour transmettre des métadonnées personnalisées au sujet du(des) texte(s) comparés. La valeur de ce champ apparaîtra avec chaque correspondance de son entité dans la sortie de la compétence.
<code>caseSensitive</code>	(Facultatif) La valeur par défaut est « False ». Valeur booléenne indiquant si les comparaisons avec le nom de l'entité doivent respecter la casse des caractères. Exemples de correspondances qui ne respectent pas la casse de « Microsoft » : microsoft, microSoft, MICROSOFT
<code>fuzzyEditDistance</code>	(Facultatif) La valeur par défaut est 0. La valeur maximale est 5. Indique le nombre acceptable de caractères divergents qui constituerait encore une correspondance avec le nom de l'entité. La plus petite approximation possible d'une correspondance donnée est renvoyée. Par exemple, si la distance de modification est définie sur 3, « Windows 10 » correspondra encore à « Windows », « Windows10 » et « windows 7 ». Lorsque le respect de la casse est défini sur « False », les différences de casse ne comptent PAS dans le cadre de la tolérance aux approximations, mais sinon elles comptent.
<code>defaultCaseSensitive</code>	(Facultatif) Modifie la valeur de respect de la casse par défaut pour cette entité. Elle peut être utilisée pour modifier les valeurs caseSensitive par défaut de tous les alias.
<code>defaultFuzzyEditDistance</code>	(Facultatif) Modifie la valeur par défaut de la distance de modification approximative pour cette entité. Elle peut être utilisée pour modifier les valeurs fuzzyEditDistance par défaut de tous les alias.
<code>aliases</code>	(Facultatif) Groupe d'objets complexes qui peut être utilisé pour spécifier d'autres orthographes ou des synonymes au nom de l'entité racine.
PROPRIÉTÉS DES ALIAS	DESCRIPTION
<code>text</code>	Autre orthographe ou représentation d'un nom d'entité cible.
<code>caseSensitive</code>	(Facultatif) Agit de la même manière que le paramètre « caseSensitive » de l'entité racine ci-dessus, mais ne s'applique qu'à cet alias.
<code>fuzzyEditDistance</code>	(Facultatif) Agit de la même manière que le paramètre « fuzzyEditDistance » de l'entité racine ci-dessus, mais ne s'applique qu'à cet alias.

### Format inline

Dans certains cas, il est plus facile de fournir directement dans la définition de la compétence la liste des entités personnalisées à comparer au format inline. Vous pouvez alors utiliser un format JSON semblable à celui décrit ci-dessus, mais il est « inlined » (inclus) dans la définition de la compétence. Seules les configurations dont la taille est inférieure à 10 Ko (taille sérialisée) peuvent être définies au format inline.

## Exemple de définition

Voici un exemple de définition de compétence utilisant un format inline :

```
{  
    "@odata.type": "#Microsoft.Skills.Text.CustomEntityLookupSkill",  
    "context": "/document",  
    "inlineEntitiesDefinition":  
    [  
        {  
            "name" : "Bill Gates",  
            "description" : "Microsoft founder.",  
            "aliases" : [  
                { "text" : "William H. Gates", "caseSensitive" : false },  
                { "text" : "BillG", "caseSensitive" : true }  
            ]  
        },  
        {  
            "name" : "Xbox One",  
            "type": "Hardware",  
            "subtype" : "Gaming Device",  
            "id" : "4e36bf9d-5550-4396-8647-8e43d7564a76",  
            "description" : "The Xbox One product"  
        }  
    ],  
    "inputs": [  
        {  
            "name": "text",  
            "source": "/document/content"  
        }  
    ],  
    "outputs": [  
        {  
            "name": "entities",  
            "targetName": "matchedEntities"  
        }  
    ]  
}
```

Par ailleurs, si vous décidez de fournir un pointeur vers le fichier de définition des entités, un exemple de définition de compétence utilisant le format `entitiesDefinitionUri` est présenté ci-dessous :

```
{  
    "@odata.type": "#Microsoft.Skills.Text.CustomEntityLookupSkill",  
    "context": "/document",  
    "entitiesDefinitionUri": "https://myblobhost.net/keyWordsConfig.csv",  
    "inputs": [  
        {  
            "name": "text",  
            "source": "/document/content"  
        }  
    ],  
    "outputs": [  
        {  
            "name": "entities",  
            "targetName": "matchedEntities"  
        }  
    ]  
}
```

## Exemple d'entrée

```
{
  "values": [
    {
      "recordId": "1",
      "data":
        {
          "text": "The company, Microsoft, was founded by Bill Gates. Microsoft's gaming console is called Xbox",
          "languageCode": "en"
        }
    }
  ]
}
```

## Exemple de sortie

```
{
  "values" :
  [
    {
      "recordId": "1",
      "data" : {
        "entities": [
          {
            "name" : "Microsoft",
            "description" : "This document refers to Microsoft the company",
            "id" : "differentIdentifyingScheme987",
            "matches" : [
              {
                "text" : "microsoft",
                "offset" : 13,
                "length" : 9,
                "matchDistance" : 0
              },
              {
                "text" : "Microsoft",
                "offset" : 49,
                "length" : 9,
                "matchDistance" : 0
              }
            ]
          },
          {
            "name" : "Bill Gates",
            "description" : "William Henry Gates III, founder of Microsoft.",
            "matches" : [
              {
                "text" : "Bill Gates",
                "offset" : 37,
                "length" : 10,
                "matchDistance" : 0
              }
            ]
          }
        ]
      }
    }
  ]
}
```

## Erreurs et avertissements

**Avertissement : la capacité maximale a été atteinte pour les correspondances, et toutes les correspondances en double ont été ignorées.**

Cet avertissement s'affiche si le nombre de correspondances détectées est supérieur au nombre maximal autorisé.

Dans ce cas, les correspondances en double ne seront pas incluses. Si cela ne vous convient pas, veuillez soumettre un [ticket de support](#) afin que nous puissions vous aider dans votre cas d'utilisation spécifique.

## Voir aussi

- [Compétences prédéfinies](#)
- [Guide pratique pour définir un ensemble de compétences](#)
- [Compétence de reconnaissance d'entités \(pour rechercher des entités bien connues\)](#)

# Compétence cognitive Extraction de document

04/10/2020 • 8 minutes to read • [Edit Online](#)

## IMPORTANT

Cette compétence est actuellement en préversion publique. Les fonctionnalités en préversion sont fournies sans contrat de niveau de service et ne sont pas recommandées pour les charges de travail de production. Pour plus d'informations, consultez [Conditions d'Utilisation Supplémentaires relatives aux Évaluations Microsoft Azure](#). Il n'y a actuellement pas de prise en charge du portail ou du SDK .NET.

La compétence **Extraction de document** extrait le contenu d'un fichier dans le pipeline d'enrichissement. Cela vous permet de tirer parti de l'étape d'extraction de document qui se produit normalement avant l'exécution des compétences avec des fichiers qui peuvent être générés par d'autres compétences.

## NOTE

Si vous élargissez le champ en augmentant la fréquence des traitements, en ajoutant des documents supplémentaires ou en ajoutant plusieurs algorithmes d'IA, vous devez [attacher une ressource Cognitive Services facturable](#). Des frais sont applicables durant l'appel des API dans Cognitive Services ainsi que pour l'extraction d'images durant la phase d'extraction du contenu des documents dans l'indexation. L'extraction de texte à partir des documents est gratuite.

L'exécution des compétences intégrées est facturée au prix actuel du [paiement à l'utilisation de Cognitive Services](#). Les prix appliqués pour l'extraction d'images sont présentés sur la [page de tarification](#).

## @odata.type

Microsoft.Skills.Util.DocumentExtractionSkill

## Paramètres de la compétence

Les paramètres respectent la casse.

ENTRÉES	VALEURS AUTORISÉES	DESCRIPTION
<code>parsingMode</code>	<code>default</code> <code>text</code> <code>json</code>	Définissez ce paramètre sur <code>default</code> pour l'extraction de documents à partir de fichiers qui ne sont pas en mode texte pur ou JSON. Définissez ce paramètre sur <code>text</code> pour améliorer les performances des fichiers en texte brut. Définissez ce paramètre sur <code>json</code> pour extraire le contenu structuré des fichiers JSON. Si <code>parsingMode</code> n'est pas défini explicitement, il sera défini sur <code>default</code> .

ENTRÉES	VALEURS AUTORISÉES	DESCRIPTION
<code>dataToExtract</code>	<code>contentAndMetadata</code> <code>allMetadata</code>	Définissez ce paramètre sur <code>contentAndMetadata</code> pour extraire toutes les métadonnées et le contenu textuel de chaque fichier. Définissez ce paramètre sur <code>allMetadata</code> pour extraire uniquement les <b>métadonnées spécifiques au type de contenu</b> (par exemple, seules les métadonnées propres aux fichiers .png). Si <code>dataToExtract</code> n'est pas défini explicitement, il sera défini sur <code>contentAndMetadata</code> .
<code>configuration</code>	Voir ci-dessous.	Dictionnaire de paramètres facultatifs qui ajustent le mode d'exécution de l'extraction de document. Reportez-vous au tableau ci-dessous pour consulter une description des propriétés de configuration prises en charge.
PARAMÈTRE DE CONFIGURATION	VALEURS AUTORISÉES	DESCRIPTION

PARAMÈTRE DE CONFIGURATION	VALEURS AUTORISÉES	DESCRIPTION
<code>imageAction</code>	<code>none</code> <code>generateNormalizedImages</code> <code>generateNormalizedImagePerPage</code>	<p>Définissez ce paramètre sur <code>none</code> pour ignorer les images incorporées ou les fichiers image du jeu de données. Il s'agit de la valeur par défaut.</p> <p>Pour l'<a href="#">analyse d'image à l'aide de compétences cognitives</a>, définissez ce paramètre sur <code>generateNormalizedImages</code> pour que la compétence crée un tableau d'images normalisées dans le cadre du craquage de documents. Cette action nécessite de définir le paramètre <code>parsingMode</code> sur <code>default</code> et le paramètre <code>dataToExtract</code> sur <code>contentAndMetadata</code>. Une image normalisée fait référence à un traitement supplémentaire qui entraîne une sortie d'image uniforme, dimensionnée et pivotée pour promouvoir un rendu cohérent lorsque vous incluez des images dans les résultats d'une recherche visuelle (par exemple, des photographies de même taille dans un contrôle de graphique comme illustré dans la <a href="#">démonstration JFK</a>). Ces informations sont générées pour chaque image lorsque vous utilisez cette option.</p> <p>Si vous définissez le paramètre sur <code>generateNormalizedImagePerPage</code>, les fichiers PDF sont traités différemment. Au lieu d'extraire les images incorporées, chaque page est rendue sous la forme d'une image et normalisée en conséquence. Les autres types de fichiers sont traités de la même façon que si <code>generateNormalizedImages</code> était défini.</p>
<code>normalizedImageMaxWidth</code>	Entier compris entre 50 et 10000	Largeur maximale (en pixels) des images normalisées générées. La valeur par défaut est 2000.
<code>normalizedImageMaxHeight</code>	Entier compris entre 50 et 10000	Hauteur maximale (en pixels) des images normalisées générées. La valeur par défaut est 2000.

#### NOTE

La valeur par défaut de 2000 pixels pour la hauteur et la largeur maximales des images normalisées est basée sur les tailles maximales prises en charge par la [compétence de reconnaissance optique de caractères](#) et la [compétence d'analyse d'image](#). La [compétence de reconnaissance optique de caractères](#) (OCR) prend en charge une largeur et une hauteur maximales de 4200 pour les langues autres que l'anglais et de 10000 pour l'anglais. Si vous augmentez les limites maximales, le traitement des images plus volumineuses peut échouer en fonction de la définition de vos compétences et de la langue des documents.

## Entrées de la compétence

NOM D'ENTRÉE	DESCRIPTION
file_data	Fichier à partir duquel le contenu doit être extrait.

L'entrée « file\_data » doit être un objet défini comme suit :

```
{  
  "$type": "file",  
  "data": "BASE64 encoded string of the file"  
}
```

Cet objet de référence de fichier peut être généré de 3 manières :

- En définissant le paramètre `allowSkillsetToReadFileDialog` de votre définition d'indexeur sur la valeur « `true` ». Ceci permet de créer un chemin `/document/file_data` qui est un objet représentant les données du fichier d'origine téléchargées à partir de votre source de données d'objets blob. Ce paramètre s'applique uniquement aux données dans le stockage d'objets BLOB.
- En définissant le paramètre `imageAction` de votre définition d'indexeur sur autre valeur que `none`. Ceci crée un tableau d'images qui suivent la convention nécessaire pour les entrées de cette compétence, si elles sont passées individuellement (c'est-à-dire `/document/normalized_images/*` ).
- Avoir une compétence personnalisée renvoie un objet JSON défini exactement comme indiqué ci-dessus. Le paramètre `$type` doit être défini sur `file` et le paramètre `data` doit être données du tableau d'octets encodé en base 64 du contenu du fichier.

## Sorties de la compétence

NOM DE SORTIE	DESCRIPTION
content	Contenu textuel du document.
normalized_images	Si la propriété <code>imageAction</code> est définie sur une autre valeur que <code>none</code> , le nouveau champ <code>normalized_images</code> contiendra un tableau d'images. Si vous souhaitez en savoir plus sur le format de sortie de chaque image, veuillez consulter la <a href="#">documentation relative à l'extraction d'images</a> .

## Exemple de définition

```
{
  "@odata.type": "#Microsoft.Skills.Util.DocumentExtractionSkill",
  "parsingMode": "default",
  "dataToExtract": "contentAndMetadata",
  "configuration": {
    "imageAction": "generateNormalizedImages",
    "normalizedImageMaxWidth": 2000,
    "normalizedImageMaxHeight": 2000
  },
  "context": "/document",
  "inputs": [
    {
      "name": "file_data",
      "source": "/document/file_data"
    }
  ],
  "outputs": [
    {
      "name": "content",
      "targetName": "content"
    },
    {
      "name": "normalized_images",
      "targetName": "normalized_images"
    }
  ]
}
```

## Exemple d'entrée

```
{
  "values": [
    {
      "recordId": "1",
      "data": {
        "file_data": {
          "$type": "file",
          "data": "aGVsbG8="
        }
      }
    }
  ]
}
```

## Exemple de sortie

```
{
  "values": [
    {
      "recordId": "1",
      "data": {
        "content": "hello",
        "normalized_images": []
      }
    }
  ]
}
```

## Voir aussi

- Compétences prédéfinies
- Guide pratique pour définir un ensemble de compétences
- Comment traiter et extraire des informations d'images dans des scénarios de recherche cognitive

# Compétence cognitive Reconnaissance d'entités

04/10/2020 • 8 minutes to read • [Edit Online](#)

La compétence **Reconnaissance d'entités** extrait les entités de différents types du texte. Cette compétence utilise les modèles Machine Learning fournis par [Analyse de texte](#) dans Cognitive Services.

## NOTE

Si vous élargissez le champ en augmentant la fréquence des traitements, en ajoutant des documents supplémentaires ou en ajoutant plusieurs algorithmes d'IA, vous devez [attacher une ressource Cognitive Services facturable](#). Des frais s'appliquent durant l'appel des API dans Cognitive Services ainsi que pour l'extraction d'images dans le cadre de la phase de craquage de document de la Recherche cognitive Azure. L'extraction de texte à partir des documents est gratuite.

L'exécution des compétences intégrées est facturée au prix actuel du [paiement à l'utilisation de Cognitive Services](#). Les prix appliqués pour l'extraction d'images sont présentés sur la [page de tarification du service Recherche cognitive Azure](#).

## @odata.type

Microsoft.Skills.Text.EntityRecognitionSkill

## Limites de données

La taille maximale d'un enregistrement doit être de 50 000 caractères telle que mesurée par `String.Length`. Si vous devez subdiviser vos données avant de les envoyer à l'extracteur de phrases clés, envisagez d'utiliser la [compétence Fractionnement de texte](#).

## Paramètres de la compétence

Les paramètres respectent la casse et sont tous facultatifs.

NOM DU PARAMÈTRE	DESCRIPTION
<code>categories</code>	Tableau des catégories à extraire. Types de catégorie possibles : "Person", "Location", "Organization", "Quantity", "Datetime", "URL", "Email". Si aucune catégorie n'est précisée, tous les types sont retournés.
<code>defaultLanguageCode</code>	Code de langue du texte d'entrée. Langues prises en charge : <code>ar, cs, da, de, en, es, fi, fr, hu, it, ja, ko, nl, no, pl, pt-BR, pt-PT, ru, sv, tr, zh-hans</code> . Les catégories d'entités ne sont pas toutes prises en charge pour toutes les langues. Consultez la remarque ci-dessous.
<code>minimumPrecision</code>	Valeur comprise entre 0 et 1. Si le score de confiance (dans la sortie <code>namedEntities</code> ) est inférieur à cette valeur, l'entité n'est pas retournée. La valeur par défaut est 0.

NOM DU PARAMÈTRE	DESCRIPTION
<code>includeTypelessEntities</code>	Affectez la valeur <code>true</code> si vous souhaitez reconnaître les entités connues qui ne correspondent pas aux catégories actuelles. Les entités reconnues sont retournées dans le champ de sortie complexe <code>entities</code> . Par exemple, « Windows 10 » est une entité bien connue (un produit), mais étant donné que la catégorie « Produits » n'est pas prise en charge, cette entité est incluse dans le champ de sortie des entités. La valeur par défaut est <code>false</code> .

## Entrées de la compétence

NOM D'ENTRÉE	DESCRIPTION
<code>languageCode</code>	facultatif. La valeur par défaut est <code>"en"</code> .
<code>text</code>	Texte à analyser.

## Sorties de la compétence

### NOTE

toutes les catégories d'entités ne sont pas prises en charge pour toutes les langues. Les types de catégories d'entités `"Person"`, `"Location"` et `"Organization"` sont pris en charge pour la liste complète des langues ci-dessus. Seules les langues `de`, `en`, `es`, `fr` et `zh-hans` prennent en charge l'extraction des types `"Quantity"`, `"Datetime"`, `"URL"` et `"Email"`. Pour plus d'informations, consultez [Langues et régions prises en charge par l'API Analyse de texte](#).

NOM DE SORTIE	DESCRIPTION
<code>persons</code>	Tableau de chaînes représentant chacune le nom d'une personne.
<code>locations</code>	Tableau de chaînes représentant chacune un lieu.
<code>organizations</code>	Tableau de chaînes représentant chacune une organisation.
<code>quantities</code>	Tableau de chaînes représentant chacune une quantité.
<code>dateTimes</code>	Tableau de chaînes représentant chacune une valeur DateTime (telle qu'elle apparaît dans le texte).
<code>urls</code>	Tableau de chaînes représentant chacune une URL.
<code>emails</code>	Tableau de chaînes représentant chacune un e-mail.

NOM DE SORTIE	DESCRIPTION
namedEntities	<p>Tableau de types complexes contenant les champs suivants :</p> <ul style="list-style-type: none"> <li>• catégorie</li> <li>• la valeur (le nom réel de l'entité) ;</li> <li>• le décalage (l'emplacement où elle a été trouvée dans le texte) ;</li> <li>• confiance (une valeur plus élevée signifie qu'il s'agit d'une entité réelle)</li> </ul>
entities	<p>Tableau de types complexes contenant des informations détaillées sur les entités extraites du texte, avec les champs suivants</p> <ul style="list-style-type: none"> <li>• name (nom réel de l'entité ; il représente une forme « normalisée »)</li> <li>• wikipediaId</li> <li>• wikipediaLanguage</li> <li>• wikipediaUrl (lien vers la page Wikipedia relative à l'entité)</li> <li>• bingId</li> <li>• type (catégorie de l'entité reconnue)</li> <li>• subType (disponible uniquement pour certaines catégories ; il offre une vue plus précise du type d'entité)</li> <li>• matches (collection complexe contenant) <ul style="list-style-type: none"> <li>◦ text (texte brut pour l'entité)</li> <li>◦ offset (emplacement où cela a été trouvé)</li> <li>◦ length (longueur du texte brut d'entité)</li> </ul> </li> </ul>

## Exemple de définition

```
{
  "@odata.type": "#Microsoft.Skills.Text.EntityRecognitionSkill",
  "categories": [ "Person", "Email" ],
  "defaultLanguageCode": "en",
  "includeTypelessEntities": true,
  "minimumPrecision": 0.5,
  "inputs": [
    {
      "name": "text",
      "source": "/document/content"
    }
  ],
  "outputs": [
    {
      "name": "persons",
      "targetName": "people"
    },
    {
      "name": "emails",
      "targetName": "contact"
    },
    {
      "name": "entities"
    }
  ]
}
```

## Exemple d'entrée

```
{  
    "values": [  
        {  
            "recordId": "1",  
            "data":  
                {  
                    "text": "Contoso corporation was founded by John Smith. They can be reached at  
contact@contoso.com",  
                    "languageCode": "en"  
                }  
        }  
    ]  
}
```

## Exemple de sortie

```
{  
    "values": [  
        {  
            "recordId": "1",  
            "data" :  
                {  
                    "persons": [ "John Smith"],  
                    "emails": ["contact@contoso.com"],  
                    "namedEntities":  
                        [  
                            {  
                                "category":"Person",  
                                "value": "John Smith",  
                                "offset": 35,  
                                "confidence": 0.98  
                            }  
                        ],  
                    "entities":  
                        [  
                            {  
                                "name": "John Smith",  
                                "wikipediaId": null,  
                                "wikipediaLanguage": null,  
                                "wikipediaUrl": null,  
                                "bingId": null,  
                                "type": "Person",  
                                "subType": null,  
                                "matches": [ {  
                                    "text": "John Smith",  
                                    "offset": 35,  
                                    "length": 10  
                                }]  
                            },  
                            {  
                                "name": "contact@contoso.com",  
                                "wikipediaId": null,  
                                "wikipediaLanguage": null,  
                                "wikipediaUrl": null,  
                                "bingId": null,  
                                "type": "Email",  
                                "subType": null,  
                                "matches": [  
                                    {  
                                        "text": "contact@contoso.com",  
                                        "offset": 70,  
                                        "length": 19  
                                    }]  
                            }  
                        ]  
                    }  
    ]  
}
```

```
        ],
    },
    {
        "name": "Contoso",
        "wikipediaId": "Contoso",
        "wikipediaLanguage": "en",
        "wikipediaUrl": "https://en.wikipedia.org/wiki/Contoso",
        "bingId": "349f014e-7a37-e619-0374-787ebb288113",
        "type": null,
        "subType": null,
        "matches": [
            {
                "text": "Contoso",
                "offset": 0,
                "length": 7
            }
        ]
    }
]
```

Notez que les décalages renvoyés pour les entités dans la sortie de cette qualification le sont directement à partir de l'[API Analyse de texte](#), ce qui signifie que si vous les utilisez pour indexer dans la chaîne d'origine, vous devez utiliser la classe [StringInfo](#) dans .NET afin d'extraire le bon contenu. [Pour plus d'informations, cliquez ici.](#)

## Cas d'erreur

Si le code de langue du document n'est pas pris en charge, une erreur est retournée et aucune entité n'est extraite.

## Voir aussi

- [Compétences prédéfinies](#)
- [Guide pratique pour définir un ensemble de compétences](#)

# Compétence cognitive Analyse d'image

04/10/2020 • 9 minutes to read • [Edit Online](#)

La compétence **Analyse d'image** extrait un ensemble complet de caractéristiques visuelles basées sur le contenu d'une image. Par exemple, vous pouvez générer une légende à partir d'une image, générer des balises ou identifier des célébrités et des paysages. Cette compétence utilise les modèles d'apprentissage automatique fournis par [Vision par ordinateur](#) dans Cognitive Services.

## NOTE

Les petits volumes (moins de 20 transactions) peuvent être exécutés gratuitement dans la Recherche cognitive Azure, mais l'exécution de charges de travail plus volumineuses nécessite l'[attachement d'une ressource facturable Cognitive Services](#). Des frais s'appliquent durant l'appel des API dans Cognitive Services ainsi que pour l'extraction d'images dans le cadre de la phase de craquage de document de la Recherche cognitive Azure. L'extraction de texte à partir des documents est gratuite.

L'exécution des compétences intégrées est facturée au prix actuel du [paiement à l'utilisation de Cognitive Services](#). Les prix appliqués pour l'extraction d'images sont présentés sur la [page de tarification du service Recherche cognitive Azure](#).

## @odata.type

Microsoft.Skills.Vision.ImageAnalysisSkill

## Paramètres de la compétence

Les paramètres respectent la casse.

NOM DU PARAMÈTRE	DESCRIPTION
<code>defaultLanguageCode</code>	Chaîne indiquant la langue à retourner. Le service retourne les résultats de la reconnaissance dans une langue donnée. Si ce paramètre n'est pas spécifié, la valeur par défaut est « en ».  Les langues prises en charge sont les suivantes : <i>en</i> : anglais (par défaut) <i>es</i> : espagnol <i>ja</i> : japonais <i>pt</i> : portugais <i>zh</i> : chinois simplifié

NOM DU PARAMÈTRE	DESCRIPTION
<code>visualFeatures</code>	<p>Tableau de chaînes qui indique les types de caractéristiques visuelles à retourner. Les types de caractéristiques visuelles valides sont les suivants :</p> <ul style="list-style-type: none"> <li>• <i>adult</i> : détecte si l'image est de nature pornographique (nudité ou acte sexuel) ou si elle est sordide (violence extrême ou sang). Le contenu sexuellement suggestif (ou contenu osé) est également détecté.</li> <li>• <i>brands</i> : détecte les différentes marques sur une image, y compris leur emplacement approximatif. La caractéristique visuelle <i>brands</i> n'est disponible qu'en anglais.</li> <li>• <i>categories</i> : catégorise le contenu de l'image en fonction d'une taxonomie définie dans la <a href="#">documentation Vision par ordinateur</a> de Cognitive Services.</li> <li>• <i>description</i> : décrit le contenu de l'image avec une phrase complète dans les langues prises en charge.</li> <li>• <i>faces</i> : détecte si des visages sont présents. Si tel est le cas, génère des coordonnées, ainsi que des paramètres d'âge et de sexe.</li> <li>• <i>objects</i> : détecte les différents objets sur une image, y compris leur emplacement approximatif. La caractéristique visuelle <i>objects</i> n'est disponible qu'en anglais.</li> <li>• <i>tags</i> : balise l'image avec une liste détaillée de mots liés au contenu de l'image.</li> </ul> <p>Les noms des caractéristiques visuelles respectent la casse. Notez que les caractéristiques visuelles <i>color</i> et <i>imageType</i> sont dépréciées, mais ces fonctionnalités sont toujours accessibles via une <a href="#">compétence personnalisée</a>.</p>
<code>details</code>	<p>Tableau de chaînes indiquant les détails spécifiques à un domaine à retourner. Les types de caractéristiques visuelles valides sont les suivants :</p> <ul style="list-style-type: none"> <li>• <i>celebrities</i> : identifie les célébrités éventuellement détectées dans l'image.</li> <li>• <i>landmarks</i> : identifie les paysages éventuellement détectés dans l'image.</li> </ul>

## Entrées de la compétence

NOM D'ENTRÉE	DESCRIPTION
<code>image</code>	Type complexe. Ne fonctionne actuellement qu'avec le champ « /documents/normalized_images », généré par l'indexeur d'objets Blob Azure lorsque <code>imageAction</code> est défini sur une valeur supérieure à <code>none</code> . Pour plus d'informations, consultez <a href="#">l'exemple</a> .

## Exemple de définition de qualification

```
{
    "description": "Extract image analysis.",
    "@odata.type": "#Microsoft.Skills.Vision.ImageAnalysisSkill",
    "context": "/document/normalized_images/*",
    "defaultLanguageCode": "en",
    "visualFeatures": [
        "tags",
        "categories",
        "description",
        "faces",
        "brands"
    ],
    "inputs": [
        {
            "name": "image",
            "source": "/document/normalized_images/*"
        }
    ],
    "outputs": [
        {
            "name": "categories"
        },
        {
            "name": "tags"
        },
        {
            "name": "description"
        },
        {
            "name": "faces"
        },
        {
            "name": "brands"
        }
    ]
}
```

#### **Exemple d'index (uniquement pour les champs categories, description, faces et tags)**

```
{
    "fields": [
        {
            "name": "id",
            "type": "Edm.String",
            "key": true,
            "searchable": true,
            "filterable": false,
            "facetable": false,
            "sortable": true
        },
        {
            "name": "blob_uri",
            "type": "Edm.String",
            "searchable": true,
            "filterable": false,
            "facetable": false,
            "sortable": true
        },
        {
            "name": "content",
            "type": "Edm.String",
            "sortable": false,
            "searchable": true,
            "filterable": false,
            "facetable": false
        }
    ]
}
```

```
{  
    "name": "categories",  
    "type": "Collection(Edm.ComplexType)",  
    "fields": [  
        {  
            "name": "name",  
            "type": "Edm.String",  
            "searchable": true,  
            "filterable": false,  
            "facetable": false  
        },  
        {  
            "name": "score",  
            "type": "Edm.Double",  
            "searchable": false,  
            "filterable": false,  
            "facetable": false  
        },  
        {  
            "name": "detail",  
            "type": "Edm.ComplexType",  
            "fields": [  
                {  
                    "name": "celebrities",  
                    "type": "Collection(Edm.ComplexType)",  
                    "fields": [  
                        {  
                            "name": "name",  
                            "type": "Edm.String",  
                            "searchable": true,  
                            "filterable": false,  
                            "facetable": false  
                        },  
                        {  
                            "name": "faceBoundingBox",  
                            "type": "Collection(Edm.ComplexType)",  
                            "fields": [  
                                {  
                                    "name": "x",  
                                    "type": "Edm.Int32",  
                                    "searchable": false,  
                                    "filterable": false,  
                                    "facetable": false  
                                },  
                                {  
                                    "name": "y",  
                                    "type": "Edm.Int32",  
                                    "searchable": false,  
                                    "filterable": false,  
                                    "facetable": false  
                                }  
                            ]  
                        },  
                        {  
                            "name": "confidence",  
                            "type": "Edm.Double",  
                            "searchable": false,  
                            "filterable": false,  
                            "facetable": false  
                        }  
                    ]  
                },  
                {  
                    "name": "landmarks",  
                    "type": "Collection(Edm.ComplexType)",  
                    "fields": [  
                        {  
                            "name": "name",  
                            "type": "Edm.String",  
                            "searchable": true,  
                            "filterable": false,  
                            "facetable": false  
                        }  
                    ]  
                }  
            ]  
        }  
    ]  
}
```

```
        "searchable": true,
        "filterable": false,
        "facetable": false
    },
    {
        "name": "confidence",
        "type": "Edm.Double",
        "searchable": false,
        "filterable": false,
        "facetable": false
    }
]
}
]
},
{
    "name": "description",
    "type": "Collection(Edm.ComplexType)",
    "fields": [
        {
            "name": "tags",
            "type": "Collection(Edm.String)",
            "searchable": true,
            "filterable": false,
            "facetable": false
        },
        {
            "name": "captions",
            "type": "Collection(Edm.ComplexType)",
            "fields": [
                {
                    "name": "text",
                    "type": "Edm.String",
                    "searchable": true,
                    "filterable": false,
                    "facetable": false
                },
                {
                    "name": "confidence",
                    "type": "Edm.Double",
                    "searchable": false,
                    "filterable": false,
                    "facetable": false
                }
            ]
        }
    ]
},
{
    "name": "faces",
    "type": "Collection(Edm.ComplexType)",
    "fields": [
        {
            "name": "age",
            "type": "Edm.Int32",
            "searchable": false,
            "filterable": false,
            "facetable": false
        },
        {
            "name": "gender",
            "type": "Edm.String",
            "searchable": false,
            "filterable": false,
            "facetable": false
        }
    ]
}
```

```
        "name": "faceBoundingBox",
        "type": "Collection(Edm.ComplexType)",
        "fields": [
            {
                "name": "x",
                "type": "Edm.Int32",
                "searchable": false,
                "filterable": false,
                "facetable": false
            },
            {
                "name": "y",
                "type": "Edm.Int32",
                "searchable": false,
                "filterable": false,
                "facetable": false
            }
        ]
    }
},
{
    "name": "tags",
    "type": "Collection(Edm.ComplexType)",
    "fields": [
        {
            "name": "name",
            "type": "Edm.String",
            "searchable": true,
            "filterable": false,
            "facetable": false
        },
        {
            "name": "confidence",
            "type": "Edm.Double",
            "searchable": false,
            "filterable": false,
            "facetable": false
        }
    ]
}
]
```

**Exemple de mappage de champs de sortie (pour l'index ci-dessus)**

```

"outputFieldMappings": [
    {
        "sourceFieldName": "/document/normalized_images/*/categories/*",
        "targetFieldName": "categories"
    },
    {
        "sourceFieldName": "/document/normalized_images/*/tags/*",
        "targetFieldName": "tags"
    },
    {
        "sourceFieldName": "/document/normalized_images/*/description",
        "targetFieldName": "description"
    },
    {
        "sourceFieldName": "/document/normalized_images/*/faces/*",
        "targetFieldName": "faces"
    },
    {
        "sourceFieldName": "/document/normalized_images/*/brands/*/name",
        "targetFieldName": "brands"
    }
]

```

### Variation sur les mappages de champs de sortie (propriétés imbriquées)

Vous pouvez définir des mappages de champs de sortie sur des propriétés de niveau inférieur, comme simplement les points de repère ou les célébrités. Dans ce cas, assurez-vous que votre schéma d'index a un champ destiné à contenir spécifiquement les points de repère.

```

"outputFieldMappings": [
    {
        "sourceFieldName": "/document/normalized_images/*/categories/detail/celebrities/*",
        "targetFieldName": "celebrities"
    }
]

```

## Exemple d'entrée

```

{
    "values": [
        {
            "recordId": "1",
            "data": {
                "image": {
                    "data": "BASE64 ENCODED STRING OF A JPEG IMAGE",
                    "width": 500,
                    "height": 300,
                    "originalWidth": 5000,
                    "originalHeight": 3000,
                    "rotationFromOriginal": 90,
                    "contentOffset": 500,
                    "pageNumber": 2
                }
            }
        }
    ]
}

```

## Exemple de sortie

```
{
    "values": [

```

```
{
  "recordId": "1",
  "data": {
    "categories": [
      {
        "name": "abstract_",
        "score": 0.00390625
      },
      {
        "name": "people_",
        "score": 0.83984375,
        "detail": {
          "celebrities": [
            {
              "name": "Satya Nadella",
              "faceBoundingBox": [
                {
                  "x": 273,
                  "y": 309
                },
                {
                  "x": 395,
                  "y": 309
                },
                {
                  "x": 395,
                  "y": 431
                },
                {
                  "x": 273,
                  "y": 431
                }
              ],
              "confidence": 0.999028444
            }
          ],
          "landmarks": [
            {
              "name": "Forbidden City",
              "confidence": 0.9978346
            }
          ]
        }
      },
      "adult": {
        "isAdultContent": false,
        "isRacyContent": false,
        "isGoryContent": false,
        "adultScore": 0.0934349000453949,
        "racyScore": 0.068613491952419281,
        "goreScore": 0.08928389008070282
      },
      "tags": [
        {
          "name": "person",
          "confidence": 0.98979085683822632
        },
        {
          "name": "man",
          "confidence": 0.94493889808654785
        },
        {
          "name": "outdoor",
          "confidence": 0.938492476940155
        },
        {
          "name": "window",
          "confidence": 0.89513939619064331
        }
      ]
    }
  }
}
```

```
        }
    ],
    "description": {
        "tags": [
            "person",
            "man",
            "outdoor",
            "window",
            "glasses"
        ],
        "captions": [
            {
                "text": "Satya Nadella sitting on a bench",
                "confidence": 0.48293603002174407
            }
        ]
    },
    "requestId": "0dbec5ad-a3d3-4f7e-96b4-dfd57efe967d",
    "metadata": {
        "width": 1500,
        "height": 1000,
        "format": "Jpeg"
    },
    "faces": [
        {
            "age": 44,
            "gender": "Male",
            "faceBoundingBox": [
                {
                    "x": 1601,
                    "y": 395
                },
                {
                    "x": 1653,
                    "y": 395
                },
                {
                    "x": 1653,
                    "y": 447
                },
                {
                    "x": 1601,
                    "y": 447
                }
            ]
        }
    ],
    "objects": [
        {
            "rectangle": {
                "x": 25,
                "y": 43,
                "w": 172,
                "h": 140
            },
            "object": "person",
            "confidence": 0.931
        }
    ],
    "brands": [
        {
            "name": "Microsoft",
            "confidence": 0.903,
            "rectangle": {
                "x": 20,
                "y": 97,
                "w": 62,
                "h": 52
            }
        }
    ]
}
```

```

        }
    ]
}
]
}
}
```

## Cas d'erreur

Dans les cas d'erreur suivants, aucun élément n'est extrait.

CODE D'ERREUR	DESCRIPTION
NotSupportedException	La langue fournie n'est pas prise en charge.
InvalidImageUrl	L'URL de l'image est incorrecte ou inaccessible.
InvalidImageFormat	Les données d'entrée ne sont pas une image valide.
InvalidImageSize	L'image d'entrée est trop grande.
UnsupportedVisualFeature	Le type de caractéristique spécifié n'est pas valide.
UnsupportedImage	Image non prise en charge, par exemple, pornographie enfantine.
InvalidDetails	Modèle spécifique à un domaine non pris en charge.

Si vous obtenez une erreur similaire à

"One or more skills are invalid. Details: Error in skill #<num>: Outputs are not supported by skill: Landmarks"

, vérifiez le chemin. Les célébrités et les points de repère sont des propriétés sous `detail`.

```

"categories": [
  {
    "name": "building_",
    "score": 0.97265625,
    "detail": {
      "landmarks": [
        {
          "name": "Forbidden City",
          "confidence": 0.92013400793075562
        }
      ]
    }
  }
]
```

## Voir aussi

- [Compétences prédéfinies](#)
- [Guide pratique pour définir un ensemble de compétences](#)
- [Créer un indexeur \(REST\)](#)

# Compétence cognitive Extraction de phrases clés

04/10/2020 • 5 minutes to read • [Edit Online](#)

La compétence **Extraction de phrases clés** évalue un texte non structuré puis, pour chaque enregistrement, retourne une liste de phrases clés. Cette compétence utilise les modèles Machine Learning fournis par [Analyse de texte](#) dans Cognitive Services.

Cette fonctionnalité est utile si vous avez besoin d'identifier rapidement les principaux points de discours dans l'enregistrement. Par exemple, si nous considérons le texte d'entrée « la nourriture était délicieuse et le personnel était merveilleux », le service retourne « nourriture » et « personnel merveilleux ».

## NOTE

Si vous élargissez le champ en augmentant la fréquence des traitements, en ajoutant des documents supplémentaires ou en ajoutant plusieurs algorithmes d'IA, vous devez [attacher une ressource Cognitive Services facturable](#). Des frais s'appliquent durant l'appel des API dans Cognitive Services ainsi que pour l'extraction d'images dans le cadre de la phase de craquage de document de la Recherche cognitive Azure. L'extraction de texte à partir des documents est gratuite.

L'exécution des compétences intégrées est facturée au prix actuel du [paiement à l'utilisation de Cognitive Services](#). Les prix appliqués pour l'extraction d'images sont présentés sur la [page de tarification du service Recherche cognitive Azure](#).

## @odata.type

Microsoft.Skills.Text.KeyPhraseExtractionSkill

## Limites de données

La taille maximale d'un enregistrement doit être de 50 000 caractères telle que mesurée par `String.Length`. Si vous devez subdiviser vos données avant de les envoyer à l'extracteur de phrases clés, envisagez d'utiliser la [compétence Fractionnement de texte](#).

## Paramètres de la compétence

Les paramètres respectent la casse.

ENTRÉES	DESCRIPTION
<code>defaultLanguageCode</code>	(Facultatif) Code de langue à appliquer aux documents qui ne spécifient pas explicitement une langue. Si le code de langue par défaut n'est pas spécifié, l'anglais (en) est utilisé comme code de langue par défaut. Voir la <a href="#">Liste complète des langues prises en charge</a> .
<code>maxKeyPhraseCount</code>	(Facultatif) Nombre maximal de phrases clés à produire.

## Entrées de la compétence

ENTRÉE	DESCRIPTION
<code>text</code>	Texte à analyser.

ENTRÉE	DESCRIPTION
languageCode	Chaîne indiquant la langue des enregistrements. Si ce paramètre n'est pas spécifié, le code de langue par défaut est utilisé pour l'analyse des enregistrements. Voir la <a href="#">Liste complète des langues prises en charge</a> .

## Sorties de la compétence

OUTPUT	DESCRIPTION
keyPhrases	Liste des expressions clés extraites du texte d'entrée. Les expressions clés sont retournées par ordre d'importance.

## Exemple de définition

Prenons l'exemple d'un enregistrement SQL qui contient les champs suivants :

```
{
    "content": "Glaciers are huge rivers of ice that ooze their way over land, powered by gravity and their own sheer weight. They accumulate ice from snowfall and lose it through melting. As global temperatures have risen, many of the world's glaciers have already started to shrink and retreat. Continued warming could see many iconic landscapes - from the Canadian Rockies to the Mount Everest region of the Himalayas - lose almost all their glaciers by the end of the century.",
    "language": "en"
}
```

Votre définition de compétence peut se présenter comme suit :

```
{
    "@odata.type": "#Microsoft.Skills.Text.KeyPhraseExtractionSkill",
    "inputs": [
        {
            "name": "text",
            "source": "/document/content"
        },
        {
            "name": "languageCode",
            "source": "/document/language"
        }
    ],
    "outputs": [
        {
            "name": "keyPhrases",
            "targetName": "myKeyPhrases"
        }
    ]
}
```

## Exemple de sortie

Dans l'exemple ci-dessus, la sortie de votre compétence sera écrite dans un nouveau nœud dans l'arborescence enrichie nommé « document/myKeyPhrases », car il s'agit du `targetName` que nous avons spécifié. Si vous ne spécifiez pas de `targetName`, le nom du nœud sera « document/keyPhrases ».

**document/myKeyPhrases**

```
[  
    "world's glaciers",  
    "huge rivers of ice",  
    "Canadian Rockies",  
    "iconic landscapes",  
    "Mount Everest region",  
    "Continued warming"  
]
```

Vous pouvez utiliser « document/myKeyPhrases » comme entrée dans d'autres compétences, ou comme source d'un [mappage de champs de sortie](#).

## Erreurs et avertissements

Si vous indiquez un code de langue non pris en charge, une erreur est générée et les phrases clés ne sont pas extraites. Si votre texte est vide, un avertissement est généré. Si votre texte comprend plus de 50 000 caractères, seuls les 50 000 premiers caractères sont analysés et un avertissement est émis.

## Voir aussi

- [Compétences prédéfinies](#)
- [Guide pratique pour définir un ensemble de compétences](#)
- [Guide pratique pour définir des mappages de champs de sortie](#)

# Compétence cognitive Détection de la langue

04/10/2020 • 4 minutes to read • [Edit Online](#)

La compétence **Détection de langue** détecte la langue du texte d'entrée et renvoie un code de langue unique pour chaque document soumis dans la requête. Le code de langue est associé à un score indiquant la puissance de l'analyse. Cette compétence utilise les modèles Machine Learning fournis par [Analyse de texte](#) dans Cognitive Services.

Cette fonctionnalité est particulièrement utile lorsqu'il est nécessaire d'indiquer la langue du texte en entrée dans d'autres compétences (par exemple, la [compétence Analyse des sentiments](#) ou la [compétence Fractionnement de texte](#)).

La détection de la langue s'appuie sur les bibliothèques de traitement en langage naturel de Bing, qui dépassent le nombre de [langues et régions prises en charge](#) répertoriées pour le service Analyse de texte. La liste exacte des langues n'est pas publiée, mais inclut toutes les langues courantes, ainsi que les variantes, dialectes et certaines langues régionales et culturelles. Si vous avez du contenu exprimé dans une langue moins fréquemment utilisée, vous pouvez [essayer l'API Détection de langue](#) pour voir si elle retourne un code. La réponse pour les langues qui ne peuvent pas être détectées est `unknown`.

## NOTE

Si vous élargissez le champ en augmentant la fréquence des traitements, en ajoutant des documents supplémentaires ou en ajoutant plusieurs algorithmes d'IA, vous devez [attacher une ressource Cognitive Services facturable](#). Des frais s'appliquent durant l'appel des API dans Cognitive Services ainsi que pour l'extraction d'images dans le cadre de la phase de craquage de document de la Recherche cognitive Azure. L'extraction de texte à partir des documents est gratuite.

L'exécution des compétences intégrées est facturée au prix actuel du [paiement à l'utilisation de Cognitive Services](#). Les prix appliqués pour l'extraction d'images sont présentés sur la [page de tarification du service Recherche cognitive Azure](#).

## @odata.type

Microsoft.Skills.Text.LanguageDetectionSkill

## Limites de données

La taille maximale d'un enregistrement doit être de 50 000 caractères telle que mesurée par `String.Length`. Si vous avez besoin de découper vos données avant de les envoyer à la compétence de détection de langage, vous pouvez utiliser la [compétence Fractionnement de texte](#).

## Entrées de la compétence

Les paramètres respectent la casse.

ENTRÉES	DESCRIPTION
<code>text</code>	Texte à analyser.

## Sorties de la compétence

NOM DE SORTIE	DESCRIPTION
languageCode	Code ISO 6391 de la langue identifiée. Exemple : « en ».
languageName	Nom de la langue. Exemple : « anglais ».
score	Valeur comprise entre 0 et 1 correspondant à la probabilité que la langue soit correctement identifiée. Le score peut être inférieur à 1 si la phrase comporte plusieurs langues.

## Exemple de définition

```
{
  "@odata.type": "#Microsoft.Skills.Text.LanguageDetectionSkill",
  "inputs": [
    {
      "name": "text",
      "source": "/document/text"
    }
  ],
  "outputs": [
    {
      "name": "languageCode",
      "targetName": "myLanguageCode"
    },
    {
      "name": "languageName",
      "targetName": "myLanguageName"
    },
    {
      "name": "score",
      "targetName": "myLanguageScore"
    }
  ]
}
```

## Exemple d'entrée

```
{
  "values": [
    {
      "recordId": "1",
      "data": {
        "text": "Glaciers are huge rivers of ice that ooze their way over land, powered by gravity and their own sheer weight."
      }
    },
    {
      "recordId": "2",
      "data": {
        "text": "Estamos muy felices de estar con ustedes."
      }
    }
  ]
}
```

## Exemple de sortie

```
{  
  "values": [  
    {  
      "recordId": "1",  
      "data":  
        {  
          "languageCode": "en",  
          "languageName": "English",  
          "score": 1,  
        }  
    },  
    {  
      "recordId": "2",  
      "data":  
        {  
          "languageCode": "es",  
          "languageName": "Spanish",  
          "score": 1,  
        }  
    }  
  ]  
}
```

## Cas d'erreur

Si le texte est exprimé dans une langue non prise en charge, une erreur est générée et aucun identificateur de langue n'est retourné.

## Voir aussi

- [Compétences prédéfinies](#)
- [Guide pratique pour définir un ensemble de compétences](#)

# Compétence cognitive OCR

04/10/2020 • 7 minutes to read • [Edit Online](#)

La compétence de **reconnaissance optique de caractères** (OCR) reconnaît le texte imprimé et manuscrit dans des fichiers image. Cette compétence utilise les modèles Machine Learning fournis par l'API [Vision par ordinateur v3.0](#) dans Cognitive Services. La compétence de **reconnaissance optique de caractères** (OCR) est mappée à la fonctionnalité suivante :

- Pour l'anglais, l'espagnol, l'allemand, le français, l'italien, le portugais et le néerlandais, la nouvelle API « [Lecture](#) » est utilisée.
- Pour toutes les autres langues, l'API « [OCR](#) » est utilisée.

La compétence **OCR** extrait le texte de fichiers d'image. Les formats de fichiers pris en charge sont les suivants :

- .JPEG
- .JPG
- .PNG
- .BMP
- .GIF
- .TIFF

## NOTE

Si vous élargissez le champ en augmentant la fréquence des traitements, en ajoutant des documents supplémentaires ou en ajoutant plusieurs algorithmes d'IA, vous devez [attacher une ressource Cognitive Services facturable](#). Des frais s'appliquent durant l'appel des API dans Cognitive Services ainsi que pour l'extraction d'images dans le cadre de la phase de craquage de document de la Recherche cognitive Azure. L'extraction de texte à partir des documents est gratuite.

L'exécution des compétences intégrées est facturée au prix actuel du [paiement à l'utilisation de Cognitive Services](#). Les prix appliqués pour l'extraction d'images sont présentés sur la [page de tarification du service Recherche cognitive Azure](#).

## Paramètres de la compétence

Les paramètres respectent la casse.

NOM DU PARAMÈTRE	DESCRIPTION
<code>detectOrientation</code>	Active la détection automatique de l'orientation de l'image. Valeurs valides : true, false.

NOM DU PARAMÈTRE	DESCRIPTION
<code>defaultLanguageCode</code>	<p>Code de langue du texte d'entrée. Les langages pris en charge incluent :</p> <ul style="list-style-type: none"> <li>zh-Hans (chinois simplifié)</li> <li>zh-Hant (chinois traditionnel)</li> <li>cs (tchèque)</li> <li>da (danois)</li> <li>nl (néerlandais)</li> <li>en (anglais)</li> <li>fi (finnois)</li> <li>fr (français)</li> <li>de (allemand)</li> <li>el (grec)</li> <li>hu (hongrois)</li> <li>it (italien)</li> <li>ja (japonais)</li> <li>ko (coréen)</li> <li>nb (norvégien)</li> <li>pl (polonais)</li> <li>pt (portugais)</li> <li>ru (russe)</li> <li>es (espagnol)</li> <li>sv (suédois)</li> <li>tr (turc)</li> <li>ar (arabe)</li> <li>ro (roumain)</li> <li>sr-Cyrl (serbe cyrillique)</li> <li>sr-Latn (serbe latin)</li> <li>sk (slovaque)</li> <li>unk (inconnu)</li> </ul> <p>Si le code langue n'est pas spécifié ou est Null, la langue est définie automatiquement sur l'anglais. Si la langue est explicitement définie sur « unk », la langue sera détectée automatiquement.</p>
<code>lineEnding</code>	<p>La valeur à utiliser entre chaque ligne détectée. Valeurs possibles : "Space", "CarriageReturn", "LineFeed". La valeur par défaut est "Space"</p>

Auparavant, un paramètre appelé "textExtractionAlgorithm" servait à indiquer si la compétence devait extraire du texte "printed" ou "handwritten". Ce paramètre est obsolète et inutile car le dernier algorithme de l'API Read est capable d'extraire les deux types de texte à la fois. Si votre définition de compétence inclut déjà ce paramètre, inutile de le supprimer, mais il ne sera plus utilisé, et les deux types de texte seront dorénavant extraits, quel que soit leur paramétrage.

## Entrées de la compétence

NOM D'ENTRÉE	DESCRIPTION
<code>image</code>	<p>Type complexe. Ne fonctionne actuellement qu'avec le champ « /documents/normalized_images », généré par l'indexeur d'objets Blob Azure lorsque <code>imageAction</code> est défini sur une valeur supérieure à <code>none</code>. Pour plus d'informations, consultez <a href="#">l'exemple</a>.</p>

## Sorties de la compétence

NOM DE SORTIE	DESCRIPTION
text	Texte brut extrait de l'image.
layoutText	Type complexe qui décrit le texte extrait ainsi que l'emplacement où le texte a été trouvé.

## Exemple de définition

```
{
  "skills": [
    {
      "description": "Extracts text (plain and structured) from image.",
      "@odata.type": "#Microsoft.Skills.Vision.OcrSkill",
      "context": "/document/normalized_images/*",
      "defaultLanguageCode": null,
      "detectOrientation": true,
      "inputs": [
        {
          "name": "image",
          "source": "/document/normalized_images/*"
        }
      ],
      "outputs": [
        {
          "name": "text",
          "targetName": "myText"
        },
        {
          "name": "layoutText",
          "targetName": "myLayoutText"
        }
      ]
    }
  ]
}
```

## Exemple de sortie text et layoutText

```
{
  "text": "Hello World. -John",
  "layoutText":
  {
    "language" : "en",
    "text" : "Hello World. -John",
    "lines" : [
      {
        "boundingBox":
        [ {"x":10, "y":10}, {"x":50, "y":10}, {"x":50, "y":30},{ "x":10, "y":30}],
        "text":"Hello World."
      },
      {
        "boundingBox": [ {"x":110, "y":10}, {"x":150, "y":10}, {"x":150, "y":30},{ "x":110, "y":30}],
        "text":"-John"
      }
    ],
    "words": [
      {
        "boundingBox": [ {"x":110, "y":10}, {"x":150, "y":10}, {"x":150, "y":30},{ "x":110, "y":30}],
        "text":"Hello"
      },
      {
        "boundingBox": [ {"x":110, "y":10}, {"x":150, "y":10}, {"x":150, "y":30},{ "x":110, "y":30}],
        "text": "World."
      },
      {
        "boundingBox": [ {"x":110, "y":10}, {"x":150, "y":10}, {"x":150, "y":30},{ "x":110, "y":30}],
        "text": "-John"
      }
    ]
  }
}
```

## Exemple : Fusion du texte extrait d'images incorporées avec le contenu du document.

La fusion de texte permet notamment de fusionner la représentation textuelle d'images (texte issu d'une compétence OCR ou légende d'une image) dans le champ de contenu d'un document.

L'exemple d'ensemble de compétences suivant crée un champ *merged\_text*. Ce champ comprend le contenu textuel de votre document et le texte obtenu par reconnaissance optique de caractères de chacune des images incorporées dans ce document.

### Syntaxe du corps de la demande

```
{
  "description": "Extract text from images and merge with content text to produce merged_text",
  "skills": [
    [
      {
        "description": "Extract text (plain and structured) from image.",
        "@odata.type": "#Microsoft.Skills.Vision.OcrSkill",
        "context": "/document/normalized_images/*",
        "defaultLanguageCode": "en",
        "detectOrientation": true,
        "inputs": [
          {
            "name": "image",
            "source": "/document/normalized_images/*"
          }
        ],
        "outputs": [
          {
            "name": "text"
          }
        ]
      },
      {
        "@odata.type": "#Microsoft.Skills.Text.MergeSkill",
        "description": "Create merged_text, which includes all the textual representation of each image inserted at the right location in the content field.",
        "context": "/document",
        "insertPreTag": " ",
        "insertPostTag": " ",
        "inputs": [
          {
            "name": "text",
            "source": "/document/content"
          },
          {
            "name": "itemsToInsert",
            "source": "/document/normalized_images/*/text"
          },
          {
            "name": "offsets",
            "source": "/document/normalized_images/*/contentOffset"
          }
        ],
        "outputs": [
          {
            "name": "mergedText",
            "targetName": "merged_text"
          }
        ]
      }
    ]
  }
}
```

L'exemple d'ensemble de compétences ci-dessus suppose l'existence d'un champ d'images normalisées. Pour générer ce champ, définissez la configuration *imageAction* dans la définition de l'indexeur sur *generateNormalizedImages* comme indiqué ci-dessous :

```
{  
  //...rest of your indexer definition goes here ...  
  "parameters": {  
    "configuration": {  
      "dataToExtract": "contentAndMetadata",  
      "imageAction": "generateNormalizedImages"  
    }  
  }  
}
```

## Voir aussi

- [Compétences prédéfinies](#)
- [Compétence Fusion de texte](#)
- [Guide pratique pour définir un ensemble de compétences](#)
- [Créer un indexeur \(REST\)](#)

# Compétence cognitive Détection PII

04/10/2020 • 7 minutes to read • [Edit Online](#)

## IMPORTANT

Cette compétence est actuellement en préversion publique. Les fonctionnalités en préversion sont fournies sans contrat de niveau de service et ne sont pas recommandées pour les charges de travail de production. Pour plus d'informations, consultez [Conditions d'Utilisation Supplémentaires relatives aux Évaluations Microsoft Azure](#). Il n'y a actuellement pas de prise en charge du portail ou du SDK .NET.

La compétence **Détection PII** extrait les informations d'identification personnelle d'un texte d'entrée et vous donne la possibilité de les y masquer de différentes façons. Cette compétence utilise les modèles Machine Learning fournis par [Analyse de texte](#) dans Cognitive Services.

## NOTE

Si vous élargissez le champ en augmentant la fréquence des traitements, en ajoutant des documents supplémentaires ou en ajoutant plusieurs algorithmes d'IA, vous devez [attacher une ressource Cognitive Services facturable](#). Des frais s'appliquent durant l'appel des API dans Cognitive Services ainsi que pour l'extraction d'images dans le cadre de la phase de craquage de document de la Recherche cognitive Azure. L'extraction de texte à partir des documents est gratuite.

L'exécution des compétences intégrées est facturée au prix actuel du [paiement à l'utilisation de Cognitive Services](#). Les prix appliqués pour l'extraction d'images sont présentés sur la [page de tarification du service Recherche cognitive Azure](#).

## @odata.type

Microsoft.Skills.Text.PIIDetectionSkill

## Limites de données

La taille maximale d'un enregistrement doit être de 50 000 caractères telle que mesurée par `String.Length`. Si vous devez subdiviser vos données avant de les envoyer à la compétence, utilisez la [compétence Fractionnement de texte](#).

## Paramètres de la compétence

Les paramètres respectent la casse et sont tous facultatifs.

NOM DU PARAMÈTRE	DESCRIPTION
<code>defaultLanguageCode</code>	Code de langue du texte d'entrée. Pour le moment, seul <code>en</code> est pris en charge.
<code>minimumPrecision</code>	Valeur comprise entre 0.0 et 1.0. Si le score de confiance (dans la sortie <code>piiEntities</code> ) est inférieur à la valeur <code>minimumPrecision</code> définie, l'entité n'est pas retournée ou masquée. L'option par défaut est 0.0.

NOM DU PARAMÈTRE	DESCRIPTION
<code>maskingMode</code>	<p>Paramètre qui fournit différentes façons de masquer les informations d'identification personnelle détectées dans le texte d'entrée. Les options suivantes sont prises en charge :</p> <ul style="list-style-type: none"> <li>• <code>none</code> (par défaut) : Cela signifie qu'aucun masquage n'est effectué et que la sortie <code>maskedText</code> n'est pas renvoyée.</li> <li>• <code>redact</code> : Cette option supprime les entités détectées du texte d'entrée sans les remplacer par quoi que ce soit. Notez que dans ce cas, le décalage dans la sortie <code>piiEntities</code> est lié au texte d'origine, et non au texte masqué.</li> <li>• <code>replace</code> : Cette option remplace les entités détectées par le caractère spécifié dans le paramètre <code>maskingCharacter</code>. Le caractère est répété sur la longueur de l'entité détectée afin que les décalages correspondent correctement à la fois au texte d'entrée et au <code>maskedText</code> de sortie.</li> </ul>
<code>maskingCharacter</code>	<p>Caractère utilisé pour masquer le texte si le paramètre <code>maskingMode</code> a la valeur <code>replace</code>. Les options suivantes sont prises en charge : <code>*</code> (par défaut), <code>#</code>, <code>x</code>. Ce paramètre peut uniquement être <code>null</code> si <code>maskingMode</code> n'est pas défini sur <code>replace</code>.</p>

## Entrées de la compétence

NOM D'ENTRÉE	DESCRIPTION
<code>languageCode</code>	facultatif. La valeur par défaut est <code>en</code> .
<code>text</code>	Texte à analyser.

## Sorties de la compétence

NOM DE SORTIE	DESCRIPTION
<code>piiEntities</code>	<p>Tableau de types complexes contenant les champs suivants :</p> <ul style="list-style-type: none"> <li>• <code>text</code> (informations d'identification personnelle réelles telles qu'elles ont été extraites)</li> <li>• <code>type</code></li> <li>• <code>subtype</code></li> <li>• <code>score</code> (une valeur plus élevée signifie qu'il s'agit probablement d'une entité réelle)</li> <li>• <code>offset</code> (décalage dans le texte d'entrée)</li> <li>• <code>length</code></li> </ul> <p><a href="#">Vous trouverez des types et sous-types possibles ici.</a></p>

NOM DE SORTIE	DESCRIPTION
maskedText	Si <code>maskingMode</code> est défini sur une valeur autre que <code>none</code> , cette sortie correspond à la chaîne résultante du masquage effectué sur le texte d'entrée tel que décrit par le <code>maskingMode</code> sélectionné. Si <code>maskingMode</code> est défini sur <code>none</code> , cette sortie n'est pas présente.

## Exemple de définition

```
{
  "@odata.type": "#Microsoft.Skills.Text.PIIDetectionSkill",
  "defaultLanguageCode": "en",
  "minimumPrecision": 0.5,
  "maskingMode": "replace",
  "maskingCharacter": "*",
  "inputs": [
    {
      "name": "text",
      "source": "/document/content"
    }
  ],
  "outputs": [
    {
      "name": "piiEntities"
    },
    {
      "name": "maskedText"
    }
  ]
}
```

## Exemple d'entrée

```
{
  "values": [
    {
      "recordId": "1",
      "data": {
        "text": "Microsoft employee with ssn 859-98-0987 is using our awesome API's."
      }
    }
  ]
}
```

## Exemple de sortie

```
{  
    "values": [  
        {  
            "recordId": "1",  
            "data" :  
            {  
                "piiEntities": [  
                    {  
                        "text": "859-98-0987",  
                        "type": "U.S. Social Security Number (SSN)",  
                        "subtype": "",  
                        "offset": 28,  
                        "length": 11,  
                        "score": 0.65  
                    }  
                ],  
                "maskedText": "Microsoft employee with ssn ***** is using our awesome API's."  
            }  
        ]  
    ]  
}
```

Notez que les décalages renvoyés pour les entités dans la sortie de cette qualification le sont directement à partir de l'[API Analyse de texte](#), ce qui signifie que si vous les utilisez pour indexer dans la chaîne d'origine, vous devez utiliser la classe [StringInfo](#) dans .NET afin d'extraire le bon contenu. [Pour plus d'informations, cliquez ici.](#)

## Cas d'erreurs et d'avertissemnts

Si le code de langue du document n'est pas pris en charge, un avertissement est retourné et aucune entité n'est extraite. Si votre texte est vide, un avertissement est généré. Si votre texte comprend plus de 50 000 caractères, seuls les 50 000 premiers caractères sont analysés et un avertissement est émis.

Si la compétence retourne un avertissement, le `maskedText` de sortie peut être vide. Dans ce cas, contrairement au fonctionnement habituel, cette sortie ne peut pas faire office d'entrée dans les compétences ultérieures. Gardez cela à l'esprit quand vous écrivez la définition de l'ensemble de compétences.

## Voir aussi

- [Compétences prédéfinies](#)
- [Guide pratique pour définir un ensemble de compétences](#)

# Compétence cognitive Sentiment

04/10/2020 • 3 minutes to read • [Edit Online](#)

La compétence **Sentiment** évalue du texte non structuré sur un continuum positif-négatif et, pour chaque enregistrement, retourne un score numérique compris entre 0 et 1. Un score proche de 1 indique un sentiment positif, et un score proche de 0 un sentiment négatif. Cette compétence utilise les modèles Machine Learning fournis par [Analyse de texte](#) dans Cognitive Services.

## NOTE

Si vous élargissez le champ en augmentant la fréquence des traitements, en ajoutant des documents supplémentaires ou en ajoutant plusieurs algorithmes d'IA, vous devez [attacher une ressource Cognitive Services facturable](#). Des frais s'appliquent durant l'appel des API dans Cognitive Services ainsi que pour l'extraction d'images dans le cadre de la phase de craquage de document de la Recherche cognitive Azure. L'extraction de texte à partir des documents est gratuite.

L'exécution des compétences intégrées est facturée au prix actuel du [paiement à l'utilisation de Cognitive Services](#). Les prix appliqués pour l'extraction d'images sont présentés sur la [page de tarification du service Recherche cognitive Azure](#).

## @odata.type

Microsoft.Skills.Text.SentimentSkill

## Limites de données

La taille maximale d'un enregistrement est de 5 000 caractères selon [String.Length](#). Si vous avez besoin de découper vos données avant de les envoyer à l'Analyseur des sentiments, utilisez la [compétence Fractionnement du texte](#).

## Paramètres de la compétence

Les paramètres respectent la casse.

NOM DU PARAMÈTRE	DESCRIPTION
<code>defaultLanguageCode</code>	(Facultatif) Code de langue à appliquer aux documents qui ne spécifient pas explicitement la langue. Voir la <a href="#">Liste complète des langues prises en charge</a> .

## Entrées de la compétence

NOM D'ENTRÉE	DESCRIPTION
<code>text</code>	Texte à analyser.
<code>languageCode</code>	(Facultatif) Chaîne indiquant la langue des enregistrements. Si ce paramètre n'est pas spécifié, la valeur par défaut est « en ». Voir la <a href="#">Liste complète des langues prises en charge</a> .

## Sorties de la compétence

NOM DE SORTIE	DESCRIPTION
score	Valeur comprise entre 0 et 1 qui représente le sentiment du texte analysé. Les valeurs proches de 0 indiquent un sentiment négatif, les valeurs proches de 0,5 un sentiment neutre et les valeurs proches de 1 un sentiment positif.

## Exemple de définition

```
{  
    "@odata.type": "#Microsoft.Skills.Text.SentimentSkill",  
    "inputs": [  
        {  
            "name": "text",  
            "source": "/document/content"  
        },  
        {  
            "name": "languageCode",  
            "source": "/document/languagecode"  
        }  
    ],  
    "outputs": [  
        {  
            "name": "score",  
            "targetName": "mySentiment"  
        }  
    ]  
}
```

## Exemple d'entrée

```
{  
    "values": [  
        {  
            "recordId": "1",  
            "data": {  
                "text": "I had a terrible time at the hotel. The staff was rude and the food was awful.",  
                "languageCode": "en"  
            }  
        }  
    ]  
}
```

## Exemple de sortie

```
{  
    "values": [  
        {  
            "recordId": "1",  
            "data": {  
                "score": 0.01  
            }  
        }  
    ]  
}
```

## Notes

Les scores de sentiment vides ne sont pas retournés pour ces enregistrements.

## Cas d'erreur

Si la langue n'est pas prise en charge, une erreur est générée et aucun score de sentiment n'est retourné.

## Voir aussi

- [Compétences prédéfinies](#)
- [Guide pratique pour définir un ensemble de compétences](#)

# Compétence cognitive Modélisation

04/10/2020 • 8 minutes to read • [Edit Online](#)

La compétence **Modélisation** regroupe plusieurs entrées dans un [type complexe](#) qui peut être référencé plus tard dans le pipeline d'enrichissement. La compétence **Modélisation** vous permet essentiellement de créer une structure, de définir le nom des membres de cette structure et d'assigner des valeurs à chaque membre. Parmi les exemples de champs regroupés utiles dans les scénarios de recherche, citons la combinaison d'un nom et d'un prénom dans une seule structure, d'une ville et d'un état dans une seule structure, ou du nom et d'une date de naissance dans une seule structure pour établir une identité unique.

De plus, la compétence **Modélisation** illustrée dans le [scénario 3](#) ajoute une propriété *sourceContext* en option dans l'entrée. Les propriétés *source* et *sourceContext* s'excluent mutuellement. Si l'entrée est dans le contexte de la compétence, utilisez simplement *source*. Si l'entrée est dans un contexte *different* de celui de la compétence, utilisez *sourceContext*. La propriété *sourceContext* implique que vous définissiez une entrée imbriquée avec l'élément spécifique traité comme source.

Le nom de sortie est toujours « *output* ». En interne, le pipeline peut mapper un autre nom, comme « *analyzedText* » comme indiqué dans les exemples ci-dessous, mais la compétence **Modélisation** elle-même retourne « *output* » dans la réponse. Cet aspect peut être important si vous effectuez un débogage de documents enrichis et notez la différence de nommage, ou si vous générez une compétence personnalisée et que vous structurez la réponse vous-même.

## NOTE

La compétence **Modélisation** n'est pas liée à une API Cognitive Services et son utilisation ne vous est pas facturée. Toutefois, vous devez toujours [attacher une ressource Cognitive Services](#) pour remplacer l'option de ressource **Gratuit** qui vous limite à un petit nombre d'enrichissements quotidiens par jour.

## @odata.type

Microsoft.Skills.Util.ShaperSkill

## Scénario 1 : types complexes

Considérez un scénario dans lequel vous souhaitez créer une structure appelée *analyzedText* dotée de deux membres : *text* et *sentiment*. Dans un index, un champ en plusieurs parties pouvant faire l'objet d'une recherche est appelé un [type complexe](#) et il est souvent créé lorsque les données sources présentent une structure complexe correspondante qui mappe à ce champ.

Toutefois, une autre approche de la création de types complexes consiste à utiliser la compétence **Modélisation**. Lorsque vous incluez cette compétence dans un ensemble de compétences, les opérations en mémoire pendant le traitement de l'ensemble de compétences peuvent sortir des formes de données avec des structures imbriquées, qui peuvent alors être mappées à un type complexe dans votre index.

L'exemple de définition de compétence suivant fournit les noms de membre comme entrée.

```
{
    "@odata.type": "#Microsoft.Skills.Util.ShaperSkill",
    "context": "/document/content/phrases/*",
    "inputs": [
        {
            "name": "text",
            "source": "/document/content/phrases/*"
        },
        {
            "name": "sentiment",
            "source": "/document/content/phrases/*/sentiment"
        }
    ],
    "outputs": [
        {
            "name": "output",
            "targetName": "analyzedText"
        }
    ]
}
```

## Exemple d'index

Un ensemble de compétences est appelé par un indexeur, et un indexeur nécessite un index. Une représentation de champ complexe dans votre index peut se présenter comme dans l'exemple suivant.

```
"name": "my-index",
"fields": [
    { "name": "myId", "type": "Edm.String", "key": true, "filterable": true },
    { "name": "analyzedText", "type": "Edm.ComplexType",
        "fields": [
            { "name": "text",
                "type": "Edm.String",
                "filterable": false,
                "sortable": false,
                "facetable": false,
                "searchable": true },
            {
                "name": "sentiment",
                "type": "Edm.Double",
                "searchable": true,
                "filterable": true,
                "sortable": true,
                "facetable": true
            }
        ]
    }
],
```

## Entrée de la compétence

Un document JSON entrant fournissant des données d'entrée exploitables pour cette compétence

Modélisation pourrait ressembler à ceci :

```
{
    "values": [
        {
            "recordId": "1",
            "data": {
                "text": "this movie is awesome",
                "sentiment": 0.9
            }
        }
    ]
}
```

## Sortie de la compétence

La compétence **Modélisation** génère un nouvel élément appelé *analyzedText* avec les éléments combinés *text* et *sentiment*. Cette sortie est conforme au schéma d'index. Elle est importée et indexée dans un index Recherche cognitive Azure.

```
{  
  "values": [  
    {  
      "recordId": "1",  
      "data":  
        {  
          "analyzedText":  
            {  
              "text": "this movie is awesome" ,  
              "sentiment": 0.9  
            }  
        }  
    }  
  ]  
}
```

## Scénario 2 : regroupement des entrées

Dans un autre exemple, imaginez qu'à différents stades du traitement du pipeline, vous avez extrait le titre d'un livre et des titres de chapitre sur des pages différentes du livre. Vous pouvez maintenant créer une structure unique composée de ces différentes entrées.

La définition de la compétence **Modélisation** pour ce scénario peut se présenter comme dans l'exemple suivant :

```
{  
  "@odata.type": "#Microsoft.Skills.Util.ShaperSkill",  
  "context": "/document",  
  "inputs": [  
    {  
      "name": "title",  
      "source": "/document/content/title"  
    },  
    {  
      "name": "chapterTitles",  
      "source": "/document/content/pages/*/chapterTitles/*/title"  
    }  
,  
  ],  
  "outputs": [  
    {  
      "name": "output",  
      "targetName": "titlesAndChapters"  
    }  
  ]  
}
```

## Sortie de la compétence

Dans ce cas, la compétence **Modélisation** aplatis tous les titres de chapitre en un tableau unique.

```
{
  "values": [
    {
      "recordId": "1",
      "data": {
        "titlesAndChapters": {
          "title": "How to be happy",
          "chapterTitles": [
            "Start young",
            "Laugh often",
            "Eat, sleep and exercise"
          ]
        }
      }
    }
  ]
}
```

## Scénario 3 : regroupement des entrées à partir de contextes imbriqués

Imaginons la situation suivante : vous avez le titre, les chapitres et le contenu d'un livre et vous avez été exécuté une reconnaissance d'entité, ainsi que des expressions clés sur le contenu. À présent, vous devez agréger les résultats à partir des différentes compétences dans une seule forme avec le nom de chapitre, les entités et les expressions clés.

La définition de la compétence **Modélisation** pour ce scénario peut se présenter comme dans l'exemple suivant :

```
{
  "@odata.type": "#Microsoft.Skills.Util.ShaperSkill",
  "context": "/document",
  "inputs": [
    {
      "name": "title",
      "source": "/document/content/title"
    },
    {
      "name": "chapterTitles",
      "sourceContext": "/document/content/pages/*/chapterTitles/*",
      "inputs": [
        {
          "name": "title",
          "source": "/document/content/pages/*/chapterTitles/*/title"
        },
        {
          "name": "number",
          "source": "/document/content/pages/*/chapterTitles/*/number"
        }
      ]
    }
  ],
  "outputs": [
    {
      "name": "output",
      "targetName": "titlesAndChapters"
    }
  ]
}
```

## Sortie de la compétence

Dans ce cas, le **modélisateur** crée un type complexe. Cette structure existe en mémoire. Si vous souhaitez l'enregistrer dans une [base de connaissances](#), vous devez créer une projection dans votre ensemble de compétences, définissant les caractéristiques de stockage.

```
{  
    "values": [  
        {  
            "recordId": "1",  
            "data": {  
                "titlesAndChapters": {  
                    "title": "How to be happy",  
                    "chapterTitles": [  
                        { "title": "Start young", "number": 1},  
                        { "title": "Laugh often", "number": 2},  
                        { "title": "Eat, sleep and exercise", "number": 3}  
                    ]  
                }  
            }  
        }  
    ]  
}
```

## Voir aussi

- [Compétences prédéfinies](#)
- [Guide pratique pour définir un ensemble de compétences](#)
- [How to use complex types](#) (Comment utiliser les types complexes)
- [Base de connaissances](#)
- [Créer une base de connaissances avec REST](#)

# Compétence cognitive Fusion de texte

04/10/2020 • 3 minutes to read • [Edit Online](#)

La compétence **Fusion de texte** consolide en un champ unique du texte issu d'une collection de champs.

## NOTE

Cette compétence n'est pas liée à une API Cognitive Services et son utilisation ne vous est pas facturée. Toutefois, vous devez toujours [attacher une ressource Cognitive Services](#) pour remplacer l'option de ressource **Gratuit** qui vous limite à un petit nombre d'enrichissements quotidiens par jour.

## @odata.type

Microsoft.Skills.Text.MergeSkill

## Paramètres de la compétence

Les paramètres respectent la casse.

NOM DU PARAMÈTRE	DESCRIPTION
<code>insertPreTag</code>	Chaîne à inclure avant chaque insertion. La valeur par défaut est <code>" "</code> . Pour omettre l'espace, choisissez la valeur <code>""</code> .
<code>insertPostTag</code>	Chaîne à inclure après chaque insertion. La valeur par défaut est <code>" "</code> . Pour omettre l'espace, choisissez la valeur <code>""</code> .

## Exemple d'entrée

Voici un exemple de document JSON fournissant des données d'entrée exploitables pour cette compétence :

```
{  
  "values": [  
    {  
      "recordId": "1",  
      "data": {  
        "text": "The brown fox jumps over the dog",  
        "itemsToInsert": ["quick", "lazy"],  
        "offsets": [3, 28]  
      }  
    }  
  ]  
}
```

## Exemple de sortie

Cet exemple montre la sortie de l'entrée précédente, à supposer que `insertPreTag` ait la valeur `" "` et `insertPostTag` la valeur `""`.

```
{  
  "values": [  
    {  
      "recordId": "1",  
      "data":  
        {  
          "mergedText": "The quick brown fox jumps over the lazy dog"  
        }  
    }  
  ]  
}
```

## Exemple étendu de définition de compétences

La fusion de texte permet notamment de fusionner la représentation textuelle d'images (texte issu d'une compétence OCR ou légende d'une image) dans le champ de contenu d'un document.

L'exemple de compétences suivant utilise la reconnaissance optique des caractères pour extraire du texte à partir d'images incorporées dans le document. Ensuite, il crée un champ *merged\_text* qui contiendra le texte avant et après reconnaissance de chaque image. Vous trouverez plus d'informations sur la reconnaissance optique des caractères [ici](#).

```
{
  "description": "Extract text from images and merge with content text to produce merged_text",
  "skills": [
    [
      {
        "description": "Extract text (plain and structured) from image.",
        "@odata.type": "#Microsoft.Skills.Vision.OcrSkill",
        "context": "/document/normalized_images/*",
        "defaultLanguageCode": "en",
        "detectOrientation": true,
        "inputs": [
          {
            "name": "image",
            "source": "/document/normalized_images/*"
          }
        ],
        "outputs": [
          {
            "name": "text"
          }
        ]
      },
      {
        "@odata.type": "#Microsoft.Skills.Text.MergeSkill",
        "description": "Create merged_text, which includes all the textual representation of each image inserted at the right location in the content field.",
        "context": "/document",
        "insertPreTag": " ",
        "insertPostTag": " ",
        "inputs": [
          {
            "name": "text",
            "source": "/document/content"
          },
          {
            "name": "itemsToInsert",
            "source": "/document/normalized_images/*/text"
          },
          {
            "name": "offsets",
            "source": "/document/normalized_images/*/contentOffset"
          }
        ],
        "outputs": [
          {
            "name": "mergedText",
            "targetName": "merged_text"
          }
        ]
      }
    ]
  }
}
```

L'exemple ci-dessus suppose l'existence d'un champ normalized-images. Pour obtenir ce champ, définissez la configuration *imageAction* dans la définition de votre indexeur sur *generateNormalizedImages* comme ci-dessous :

```
{  
    //...rest of your indexer definition goes here ...  
    "parameters":{  
        "configuration":{  
            "dataToExtract":"contentAndMetadata",  
            "imageAction":"generateNormalizedImages"  
        }  
    }  
}
```

## Voir aussi

- [Compétences prédéfinies](#)
- [Guide pratique pour définir un ensemble de compétences](#)
- [Créer un indexeur \(REST\)](#)

# Compétence cognitive Fractionnement de texte

04/10/2020 • 4 minutes to read • [Edit Online](#)

La compétence **Fractionnement de texte** découpe le texte en segments. Vous pouvez choisir de décomposer le texte en phrases ou en pages d'une longueur donnée. Cette opération est particulièrement utile si d'autres compétences en aval imposent des critères de longueur maximale du texte.

## NOTE

Cette compétence n'est pas liée à une API Cognitive Services et son utilisation ne vous est pas facturée. Toutefois, vous devez toujours [attacher une ressource Cognitive Services](#) pour remplacer l'option de ressource **Gratuit** qui vous limite à un petit nombre d'enrichissements quotidiens par jour.

## @odata.type

Microsoft.Skills.Text.SplitSkill

## Paramètres de la compétence

Les paramètres respectent la casse.

NOM DU PARAMÈTRE	DESCRIPTION
<code>textSplitMode</code>	« pages » ou « sentences » (phrases)
<code>maximumPageLength</code>	Si <code>textSplitMode</code> est défini sur « pages », il s'agit de la longueur maximale de la page selon <code>String.Length</code> . La valeur minimale est 300. Si <code>textSplitMode</code> est réglé sur « pages », l'algorithme essaie de fractionner le texte en blocs d'une taille maximale de « <code>maximumPageLength</code> ». Dans ce cas, l'algorithme fera de son mieux pour arrêter la phrase sur une limite de phrase, de sorte que la taille du bloc peut être légèrement inférieure à « <code>maximumPageLength</code> ».
<code>defaultLanguageCode</code>	(Facultatif) L'un des codes de langue suivants : <code>da, de, en, es, fi, fr, it, ko, pt</code> . La langue par défaut est l'anglais (en). Quelques points à prendre en compte : <ul style="list-style-type: none"><li>• Si vous utilisez un format codelangue-codepays, seule la partie codelangue du format est utilisée.</li><li>• Si la langue ne figure pas dans la liste précédente, la compétence Fractionnement découpe le texte suivant les limites de caractères.</li><li>• Il est utile d'indiquer un code de langue pour éviter de couper un mot en deux dans les langues sans espaces blancs, par exemple, le chinois, le japonais et le coréen.</li><li>• Si vous ne connaissez pas la langue (par exemple, vous devez fractionner le texte pour l'entrée dans <a href="#">LanguageDetectionSkill</a>), la valeur par défaut de l'anglais (en) doit être suffisante.</li></ul>

## Entrées de la compétence

NOM DU PARAMÈTRE	DESCRIPTION
<code>text</code>	Texte à fractionner en sous-chaînes.
<code>languageCode</code>	(Facultatif) Code de langue du document. Si vous ne connaissez pas la langue (par exemple, vous devez fractionner le texte pour l'entrée dans <a href="#">LanguageDetectionSkill</a> ), vous pouvez retirer cette entrée.

## Sorties de la compétence

NOM DU PARAMÈTRE	DESCRIPTION
<code>textItems</code>	Tableau des sous-chaînes extraites.

## Exemple de définition

```
{
  "@odata.type": "#Microsoft.Skills.Text.SplitSkill",
  "textSplitMode" : "pages",
  "maximumPageLength": 1000,
  "defaultLanguageCode": "en",
  "inputs": [
    {
      "name": "text",
      "source": "/document/content"
    },
    {
      "name": "languageCode",
      "source": "/document/language"
    }
  ],
  "outputs": [
    {
      "name": "textItems",
      "targetName": "mypages"
    }
  ]
}
```

## Exemple d'entrée

```
{
  "values": [
    {
      "recordId": "1",
      "data": {
        "text": "This is a the loan application for Joe Romero, a Microsoft employee who was born in Chile and who then moved to Australia...",
        "languageCode": "en"
      }
    },
    {
      "recordId": "2",
      "data": {
        "text": "This is the second document, which will be broken into several pages...",  

        "languageCode": "en"
      }
    }
  ]
}
```

## Exemple de sortie

```
{
  "values": [
    {
      "recordId": "1",
      "data": {
        "textItems": [
          "This is the loan...",
          "On the second page we..."
        ]
      }
    },
    {
      "recordId": "2",
      "data": {
        "textItems": [
          "This is the second document...",
          "On the second page of the second doc..."
        ]
      }
    }
  ]
}
```

## Cas d'erreur

Si la langue n'est pas prise en charge, un avertissement est généré et le texte est fractionné suivant les limites de caractères.

## Voir aussi

- [Compétences prédéfinies](#)
- [Guide pratique pour définir un ensemble de compétences](#)

# Compétence cognitive Traduction de texte

04/10/2020 • 7 minutes to read • [Edit Online](#)

La compétence **Traduction de texte** évalue le texte et, pour chaque enregistrement, retourne le texte traduit dans la langue cible spécifiée. Cette compétence utilise l'[API de traduction de texte Translator Text v3.0](#) disponible dans Cognitive Services.

Cette fonctionnalité est utile si vous attendez à ce que vos documents ne soient pas tous dans la même langue, auquel cas vous pouvez normaliser le texte dans une seule langue avant l'indexation de la recherche en le traduisant. Elle est également utile dans les cas d'usage de localisation, où vous pouvez avoir besoin de copies du même texte dans plusieurs langues.

L'[API Traduction de texte Translator Text v3.0](#) est un service cognitif non régional, ce qui signifie que vos données ne sont pas conservées dans la même région que votre ressource Recherche cognitive Azure ou Cognitive Services associée.

## NOTE

Si vous élargissez le champ en augmentant la fréquence des traitements, en ajoutant des documents supplémentaires ou en ajoutant plusieurs algorithmes d'IA, vous devez [attacher une ressource Cognitive Services facturable](#). Des frais s'appliquent durant l'appel des API dans Cognitive Services ainsi que pour l'extraction d'images dans le cadre de la phase de craquage de document de la Recherche cognitive Azure. L'extraction de texte à partir des documents est gratuite.

L'exécution des compétences intégrées est facturée au prix actuel du [paiement à l'utilisation de Cognitive Services](#). Les prix appliqués pour l'extraction d'images sont présentés sur la [page de tarification du service Recherche cognitive Azure](#).

## @odata.type

Microsoft.Skills.Text.TranslationSkill

## Limites de données

La taille maximale d'un enregistrement doit être de 50 000 caractères telle que mesurée par `String.Length`. Si vous devez subdiviser vos données avant de les envoyer à la compétence de traduction de texte, utilisez la [compétence Fractionnement de texte](#).

## Paramètres de la compétence

Les paramètres respectent la casse.

ENTRÉES	DESCRIPTION
defaultToLanguageCode	(Obligatoire) Code de la langue dans laquelle traduire les documents, pour ceux qui ne spécifient pas explicitement la langue cible. Voir la <a href="#">Liste complète des langues prises en charge</a> .

ENTRÉES	DESCRIPTION
defaultFromLanguageCode	(Facultatif) Code de la langue à partir de laquelle traduire les documents, pour ceux qui ne spécifient pas explicitement la langue source. Si defaultFromLanguageCode n'est pas spécifié, la détection automatique de la langue fournie par l'API de traduction de texte Translator Text est utilisée pour déterminer la langue source. Voir la <a href="#">Liste complète des langues prises en charge</a> .
suggestedFrom	(Facultatif) Code de la langue à partir de laquelle traduire les documents quand ni l'entrée fromLanguageCode ni le paramètre defaultFromLanguageCode ne sont fournis, et que la détection automatique de la langue échoue. Si la langue suggestedFrom n'est pas spécifiée, l'anglais (en) est utilisé comme langue suggestedFrom. Voir la <a href="#">Liste complète des langues prises en charge</a> .

## Entrées de la compétence

NOM D'ENTRÉE	DESCRIPTION
text	Texte à traduire.
toLanguageCode	Chaîne indiquant la langue dans laquelle le texte doit être traduit. Si cette entrée n'est pas spécifiée, defaultToLanguageCode est utilisé pour traduire le texte. Voir la <a href="#">Liste complète des langues prises en charge</a> .
fromLanguageCode	Chaîne indiquant la langue actuelle du texte. Si ce paramètre n'est pas spécifié, defaultFromLanguageCode (ou la détection automatique de la langue si defaultFromLanguageCode n'est pas fourni) est utilisé pour traduire le texte. Voir la <a href="#">Liste complète des langues prises en charge</a> .

## Sorties de la compétence

NOM DE SORTIE	DESCRIPTION
translatedText	Chaîne de résultat de la traduction du texte de translatedFromLanguageCode vers translatedToLanguageCode.
translatedToLanguageCode	Chaîne indiquant le code de langue dans lequel le texte a été traduit. Utile si vous traduisez dans plusieurs langues et que vous souhaitez pouvoir effectuer le suivi de la langue de chaque texte.
translatedFromLanguageCode	Chaîne indiquant le code de langue à partir duquel le texte a été traduit. Utile si vous avez opté pour l'option de détection automatique de la langue, car cette sortie vous donnera le résultat de cette détection.

## Exemple de définition

```
{
    "@odata.type": "#Microsoft.Skills.Text.TranslationSkill",
    "defaultToLanguageCode": "fr",
    "suggestedFrom": "en",
    "context": "/document",
    "inputs": [
        {
            "name": "text",
            "source": "/document/text"
        }
    ],
    "outputs": [
        {
            "name": "translatedText",
            "targetName": "translatedText"
        },
        {
            "name": "translatedFromLanguageCode",
            "targetName": "translatedFromLanguageCode"
        },
        {
            "name": "translatedToLanguageCode",
            "targetName": "translatedToLanguageCode"
        }
    ]
}
```

## Exemple d'entrée

```
{
    "values": [
        {
            "recordId": "1",
            "data": {
                "text": "We hold these truths to be self-evident, that all men are created equal."
            }
        },
        {
            "recordId": "2",
            "data": {
                "text": "Estamos muy felices de estar con ustedes."
            }
        }
    ]
}
```

## Exemple de sortie

```
{  
  "values": [  
    {  
      "recordId": "1",  
      "data":  
        {  
          "translatedText": "Nous tenons ces vérités pour évidentes, que tous les hommes sont créés égaux.",  
          "translatedFromLanguageCode": "en",  
          "translatedToLanguageCode": "fr"  
        }  
    },  
    {  
      "recordId": "2",  
      "data":  
        {  
          "translatedText": "Nous sommes très heureux d'être avec vous.",  
          "translatedFromLanguageCode": "es",  
          "translatedToLanguageCode": "fr"  
        }  
    }  
  ]  
}
```

## Erreurs et avertissements

Si vous fournissez un code de langue non pris en charge pour la langue source ou cible, une erreur est générée et le texte n'est pas traduit. Si votre texte est vide, un avertissement est généré. Si votre texte comprend plus de 50 000 caractères, seuls les 50 000 premiers caractères sont traduits et un avertissement est émis.

## Voir aussi

- [Compétences prédéfinies](#)
- [Guide pratique pour définir un ensemble de compétences](#)

# Compétence AML dans un pipeline d'enrichissement

## Recherche cognitive Azure

04/10/2020 • 10 minutes to read • [Edit Online](#)

### IMPORTANT

Cette compétence est actuellement en préversion publique. Les fonctionnalités en préversion sont fournies sans contrat de niveau de service et ne sont pas recommandées pour les charges de travail de production. Pour plus d'informations, consultez [Conditions d'Utilisation Supplémentaires relatives aux Évaluations Microsoft Azure](#). Le SDK .NET n'est actuellement pas pris en charge.

La compétence **AML** vous permet d'étendre l'enrichissement par IA à l'aide d'un modèle [Azure Machine Learning](#) (AML) personnalisé. Une fois qu'un modèle AML est [formé et déployé](#), une compétence **AML** l'intègre dans l'enrichissement par IA.

À l'instar des compétences intégrées, une compétence **AML** a des entrées et des sorties. Les entrées sont envoyées à votre service AML déployé sous la forme d'un objet JSON, qui génère une charge utile JSON en réponse avec un code d'état de réussite. La réponse est censée avoir les sorties spécifiées par votre compétence **AML**. Toute autre réponse est considérée comme une erreur et aucun enrichissement n'est effectué.

### NOTE

L'indexeur réessaie deux fois pour certains codes d'état HTTP standard retournés par le service AML. Ces codes d'état HTTP sont les suivants :

- 503 Service Unavailable
- 429 Too Many Requests

## Prérequis

- Un [espace de travail AML](#)
- Une [cible de calcul AML Azure Kubernetes Service](#) dans cet espace de travail avec un [modèle déployé](#)
  - La [cible de calcul doit permettre l'utilisation du protocole SSL](#). Recherche cognitive Azure autorise uniquement l'accès aux points de terminaison [https](#).
  - Les certificats autosignés ne peuvent pas être utilisés.

## @odata.type

Microsoft.Skills.Custom.AmlSkill

## Paramètres de la compétence

Les paramètres respectent la casse. Les paramètres que vous choisissez d'utiliser dépendent de [l'authentification dont votre service AML a besoin, le cas échéant](#).

NOM DU PARAMÈTRE	DESCRIPTION
------------------	-------------

NOM DU PARAMÈTRE	DESCRIPTION
<code>uri</code>	(Requis en l'absence d'authentification ou pour une authentification par clé) URI de scoring du service AML auquel la charge utile <i>JSON</i> sera envoyée. Seul le schéma d'URI <b>https</b> est autorisé.
<code>key</code>	(Requis pour l'authentification par clé) Clé du service AML.
<code>resourceId</code>	(Requis pour l'authentification par jeton). ID de ressource Azure Resource Manager du service AML. Il doit être au format <code>subscriptions/{guid}/resourceGroups/{resource-group-name}/Microsoft.MachineLearningServices/workspaces/{workspace-name}/services/{service_name}</code> .
<code>region</code>	(Facultatif pour l'authentification par jeton). Région dans laquelle le service AML est déployé.
<code>timeout</code>	(Facultatif) Si spécifié, indique le délai d'expiration pour le client http qui effectue l'appel d'API. Il doit être formaté en tant que valeur « dayTimeDuration » XSD (un sous-ensemble limité d'une valeur de <a href="#">durée ISO 8601</a> ). Par exemple, <code>PT60S</code> pour 60 secondes. S'il n'est pas défini, une valeur par défaut de 30 secondes est choisie. Le délai d'expiration peut être défini sur 230 secondes maximum et 1 seconde minimum.
<code>degreeOfParallelism</code>	(Facultatif) Lorsqu'il est spécifié, indique le nombre d'appels que l'indexeur effectuera en parallèle au point de terminaison que vous avez fourni. Vous pouvez réduire cette valeur si votre point de terminaison échoue avec une charge de requête trop élevée, ou l'augmenter si votre point de terminaison est en mesure d'accepter plus de requêtes et si vous souhaitez augmenter les performances de l'indexeur. S'il n'est pas défini, une valeur par défaut de 5 secondes est utilisée. Le <code>degreeOfParallelism</code> peut avoir une valeur maximale de 10 et un minimum de 1.

## Paramètres de compétence à utiliser

Les paramètres de compétence AML requis dépendent de l'authentification utilisée par votre service AML, le cas échéant. Les services AML fournissent trois options d'authentification :

- **Authentification basée sur une clé.** Une clé statique est fournie pour authentifier les demandes de scoring des compétences AML
  - Utiliser les paramètres `uri` et `key`
- **Authentification basée sur un jeton.** Le service AML est [déployé à l'aide de l'authentification basée sur un jeton](#). L'[identité managée](#) du service Recherche cognitive Azure reçoit le rôle [Lecteur](#) dans l'espace de travail du service AML. La compétence AML utilise ensuite l'identité managée du service Recherche cognitive Azure pour s'authentifier auprès du service AML, sans qu'aucune clé statique soit nécessaire.
  - Utiliser le paramètre `resourceId`
  - Si le service Recherche cognitive Azure se trouve dans une autre région que celle de l'espace de travail AML, utilisez le paramètre `region` pour définir la région dans laquelle le service AML a été déployé.
- Aucune authentification. Aucune authentification n'est requise pour utiliser le service AML
  - Utiliser le paramètre `uri`

## Entrées de la compétence

Il n'y a pas d'entrée « prédéfinie » pour cette compétence. Si vous choisissez comme entrées un ou plusieurs champs déjà disponibles au moment de l'exécution de cette compétence, la charge utile *JSON* envoyée au service AML aura des champs différents.

## Sorties de la compétence

Il n'y a pas de sortie « prédéfinie » pour cette compétence. En fonction de la réponse envoyée par votre service AML, ajoutez des champs de sortie à récupérer dans la réponse *JSON*.

## Exemple de définition

```
{  
    "@odata.type": "#Microsoft.Skills.Custom.AmlSkill",  
    "description": "A sample model that detects the language of sentence",  
    "uri": "https://contoso.count-things.com/score",  
    "context": "/document",  
    "inputs": [  
        {  
            "name": "text",  
            "source": "/document/content"  
        }  
    ],  
    "outputs": [  
        {  
            "name": "detected_language_code"  
        }  
    ]  
}
```

## Exemple de structure JSON d'entrée

Cette structure *JSON* représente la charge utile à envoyer à votre service AML. Les champs de haut niveau de la structure correspondront aux « noms » spécifiés dans la section `inputs` de la définition de compétence. La valeur de ces champs provient de la `source` de ces champs (qui peut être un champ dans le document ou éventuellement une autre compétence)

```
{  
    "text": "Este es un contrato en Inglés"  
}
```

## Exemple de structure JSON de sortie

Le terme « output » correspond à la réponse renvoyée par votre service AML. Le service AML ne doit retourner qu'une charge utile *JSON* (vérifiée en examinant l'en-tête de réponse `Content-Type`) et doit être un objet dans lequel les champs sont des enrichissements correspondant aux « noms » dans la section `output` et dont la valeur est l'enrichissement.

```
{  
    "detected_language_code": "es"  
}
```

## Exemple de définition de mise en forme inlined

```
{
  "@odata.type": "#Microsoft.Skills.Custom.AmlSkill",
  "description": "A sample model that detects the language of sentence",
  "uri": "https://contoso.count-things.com/score",
  "context": "/document",
  "inputs": [
    {
      "name": "shapedText",
      "sourceContext": "/document",
      "inputs": [
        {
          "name": "content",
          "source": "/document/content"
        }
      ]
    }
  ],
  "outputs": [
    {
      "name": "detected_language_code"
    }
  ]
}
```

## Structure JSON d'entrée de mise en forme inlined

```
{
  "shapedText": { "content": "Este es un contrato en Inglés" }
}
```

## Structure JSON d'exemple de sortie de mise en forme inlined

```
{
  "detected_language_code": "es"
}
```

## Cas d'erreur

En plus de la non-disponibilité de votre service AML ou de l'envoi de codes d'état non réussis, les cas suivants sont considérés comme erronés :

- Si le service AML retourne un code d'état de réussite, mais que la réponse indique que ce n'est pas `application/json`, la réponse est considérée non valide et aucun enrichissement n'est effectué.
- Si le service AML retourne un JSON non valide

Quand le service AML n'est pas disponible ou retourne une erreur HTTP, une erreur conviviale avec tous les détails disponibles sur l'erreur HTTP est ajoutée à l'historique des exécutions de l'indexeur.

## Voir aussi

- [Guide pratique pour définir un ensemble de compétences](#)
- [Résolution des problèmes liés au service AML](#)

# Compétence API web personnalisée dans un pipeline d'enrichissement de Recherche cognitive Azure

04/10/2020 • 10 minutes to read • [Edit Online](#)

La compétence **API web personnalisée** vous permet d'étendre l'enrichissement par IA en appelant un point de terminaison d'API web qui fournit des opérations personnalisées. Tout comme les compétences intégrées, une compétence **API web personnalisée** a des entrées et des sorties. Selon les entrées, votre API web reçoit une charge utile JSON pendant l'exécution de l'indexeur et génère une charge utile JSON en réponse, ainsi qu'un code d'état de réussite. La réponse est censée avoir les sorties spécifiées par votre compétence personnalisée. Toute autre réponse est considérée comme une erreur et aucun enrichissement n'est effectué.

La structure des charges utiles JSON est décrite plus bas dans ce document.

## NOTE

L'indexeur réessaie deux fois pour certains codes d'état HTTP standard retournés par l'API web. Ces codes d'état HTTP sont les suivants :

- `502 Bad Gateway`
- `503 Service Unavailable`
- `429 Too Many Requests`

## @odata.type

Microsoft.Skills.Custom.WebApiSkill

## Paramètres de la compétence

Les paramètres respectent la casse.

NOM DU PARAMÈTRE	DESCRIPTION
<code>uri</code>	URI de l'API web à laquelle la charge utile <i>JSON</i> est envoyée. Seul le schéma d'URI <b>https</b> est autorisé
<code>httpMethod</code>	Méthode à utiliser pour envoyer la charge utile. Les méthodes autorisées sont <code>PUT</code> ou <code>POST</code>
<code>httpHeaders</code>	Collection de paires clé-valeur où les clés représentent les noms d'en-tête et les valeurs représentent les valeurs d'en-tête à envoyer à votre API web avec la charge utile. Les en-têtes suivants sont interdits dans cette collection : <code>Accept</code> , <code>Accept-Charset</code> , <code>Accept-Encoding</code> , <code>Content-Length</code> , <code>Content-Type</code> , <code>Cookie</code> , <code>Host</code> , <code>TE</code> , <code>Upgrade</code> , <code>Via</code>

NOM DU PARAMÈTRE	DESCRIPTION
<code>timeout</code>	(Facultatif) Si spécifié, indique le délai d'expiration pour le client http qui effectue l'appel d'API. Il doit être formaté en tant que valeur « dayTimeDuration » XSD (un sous-ensemble limité d'une valeur de <a href="#">durée ISO 8601</a> ). Par exemple, <code>PT60S</code> pour 60 secondes. S'il n'est pas défini, une valeur par défaut de 30 secondes est choisie. Le délai d'expiration peut être défini sur 230 secondes maximum et 1 seconde minimum.
<code>batchSize</code>	(Facultatif) Indique le nombre « d'enregistrements de données » (voir la structure de charge utile <i>JSON</i> ci-dessous) à envoyer par appel d'API. S'il n'est pas défini, une valeur par défaut de 1000 est choisie. Nous vous recommandons d'utiliser ce paramètre pour avoir un compromis entre le débit d'indexation et la charge sur votre API
<code>degreeOfParallelism</code>	(Facultatif) Lorsqu'il est spécifié, indique le nombre d'appels que l'indexeur effectuera en parallèle au point de terminaison que vous avez fourni. Vous pouvez réduire cette valeur si votre point de terminaison échoue avec une charge de requête trop élevée, ou l'augmenter si votre point de terminaison est en mesure d'accepter plus de requêtes et si vous souhaitez augmenter les performances de l'indexeur. S'il n'est pas défini, une valeur par défaut de 5 secondes est utilisée. Le <code>degreeOfParallelism</code> peut avoir une valeur maximale de 10 et un minimum de 1.

## Entrées de la compétence

Il n'y a pas d'entrée « prédéfinie » pour cette compétence. Si vous choisissez comme entrées un ou plusieurs champs déjà disponibles au moment de l'exécution de cette compétence, la charge utile *JSON* envoyée à l'API web a des champs différents.

## Sorties de la compétence

Il n'y a pas de sortie « prédéfinie » pour cette compétence. En fonction de la réponse envoyée par votre API web, ajoutez des champs de sortie à choisir dans la réponse *JSON*.

## Exemple de définition

```

{
    "@odata.type": "#Microsoft.Skills.Custom.WebApiSkill",
    "description": "A custom skill that can identify positions of different phrases in the source
text",
    "uri": "https://contoso.count-things.com",
    "batchSize": 4,
    "context": "/document",
    "inputs": [
        {
            "name": "text",
            "source": "/document/content"
        },
        {
            "name": "language",
            "source": "/document/languageCode"
        },
        {
            "name": "phraseList",
            "source": "/document/keyphrases"
        }
    ],
    "outputs": [
        {
            "name": "hitPositions"
        }
    ]
}

```

## Exemple de structure JSON d'entrée

Cette structure *JSON* représente la charge utile à envoyer à votre API web. Elle suit toujours ces contraintes :

- L'entité de niveau supérieur est appelée `values` et est un tableau d'objets. Le nombre de ces objets équivaut à `batchSize` au maximum
- Chaque objet dans le tableau `values` a
  - Une propriété `recordId` constituant une chaîne **unique**, utilisée pour identifier cet enregistrement.
  - Une propriété `data` constituant un objet *JSON*. Les champs de la propriété `data` correspondent aux « noms » spécifiés dans la section `inputs` de la définition de compétence. La valeur de ces champs provient de la `source` de ces champs (qui peut être un champ dans le document ou éventuellement une autre compétence)

```
{
  "values": [
    {
      "recordId": "0",
      "data": {
        "text": "Este es un contrato en Inglés",
        "language": "es",
        "phraseList": ["Este", "Inglés"]
      }
    },
    {
      "recordId": "1",
      "data": {
        "text": "Hello world",
        "language": "en",
        "phraseList": ["Hi"]
      }
    },
    {
      "recordId": "2",
      "data": {
        "text": "Hello world, Hi world",
        "language": "en",
        "phraseList": ["world"]
      }
    },
    {
      "recordId": "3",
      "data": {
        "text": "Test",
        "language": "es",
        "phraseList": []
      }
    }
  ]
}
```

## Exemple de structure JSON de sortie

« output » correspond à la réponse renvoyée par votre API web. L'API web doit retourner uniquement une charge utile *JSON* (vérifiée en examinant l'en-tête de réponse `Content-Type`) et doit satisfaire les contraintes suivantes :

- Il doit y avoir une entité de niveau supérieur appelée `values` qui doit être un tableau d'objets.
- Le nombre d'objets dans le tableau doit être le même que le nombre d'objets envoyés à l'API web.
- Chaque objet doit avoir :
  - Une propriété `recordId`
  - Une propriété `data`, qui est un objet dans lequel les champs sont des enrichissements correspondant aux « noms » dans `output` et dont la valeur est considérée comme l'enrichissement.
  - Une propriété `errors`, tableau listant toutes les erreurs rencontrées et qui est ajouté à l'historique d'exécution de l'indexeur. Cette propriété est obligatoire, mais peut avoir une valeur `null`.
  - Une propriété `warnings`, tableau listant tous les avertissements rencontrés et qui est ajouté à l'historique d'exécution de l'indexeur. Cette propriété est obligatoire, mais peut avoir une valeur `null`.
- Les objets dans le tableau `values` ne sont pas nécessairement dans le même ordre que les objets dans le

tableau `values` envoyé dans la demande à l'API web. Toutefois, comme le `recordId` est utilisé pour la corrélation, tous les enregistrements dans la réponse contenant un `recordId` absent de la demande d'origine envoyée à l'API web sont ignorés.

```
{  
    "values": [  
        {  
            "recordId": "3",  
            "data": {},  
            "errors": [  
                {  
                    "message" : "'phraseList' should not be null or empty"  
                }  
            ],  
            "warnings": null  
        },  
        {  
            "recordId": "2",  
            "data": {  
                "hitPositions": [6, 16]  
            },  
            "errors": null,  
            "warnings": null  
        },  
        {  
            "recordId": "0",  
            "data": {  
                "hitPositions": [0, 23]  
            },  
            "errors": null,  
            "warnings": null  
        },  
        {  
            "recordId": "1",  
            "data": {  
                "hitPositions": []  
            },  
            "errors": null,  
            "warnings": {  
                "message": "No occurrences of 'Hi' were found in the input text"  
            }  
        },  
    ]  
}
```

## Cas d'erreur

En plus de la non-disponibilité de votre API web ou de l'envoi de codes d'état non réussis, les cas suivants sont considérés comme erronés :

- Si l'API web retourne un code d'état de réussite, mais que la réponse indique que ce n'est pas `application/json`, la réponse est considérée comme non valide et aucun enrichissement n'est effectué.
- S'il y a des enregistrements **non valides** (dont le `recordId` n'est pas dans la demande d'origine ou qui contient des valeurs dupliquées) dans le tableau `values` de réponse, aucun enrichissement n'est effectué pour **ces** enregistrements.

Quand l'API web n'est pas disponible ou retourne une erreur HTTP, une erreur conviviale avec tous les détails disponibles sur l'erreur HTTP est ajoutée à l'historique d'exécution de l'indexeur.

## Voir aussi

- [Guide pratique pour définir un ensemble de compétences](#)
- [Ajouter une compétence personnalisée à un pipeline d'enrichissement par l'IA](#)
- [Exemple : Création d'une compétence personnalisée pour l'enrichissement par l'IA](#)

# Compétences cognitives déconseillées dans Recherche cognitive Azure

04/10/2020 • 4 minutes to read • [Edit Online](#)

Ce document décrit les compétences cognitives qui sont considérées comme déconseillées. Utilisez le guide suivant pour le contenu :

- Nom de la compétence : nom de la compétence qui sera déconseillée, il est mappé sur l'attribut @odata.type.
- Dernière version de l'API disponible : Dernière version de l'API publique Recherche cognitive Azure via laquelle des ensembles de compétences contenant la compétence déconseillée correspondante peuvent être créés/mis à jour.
- Fin de la prise en charge : dernier jour après lequel la compétence correspondante est considérée non prise en charge. Les compétences créées précédemment doivent continuer à fonctionner, mais il est recommandé aux utilisateurs de migrer hors d'une compétence déconseillée.
- Recommandations : chemin de migration vers l'avant pour utiliser une compétence prise en charge. Il est conseillé aux utilisateurs de suivre ces suggestions pour continuer à bénéficier du support technique.

## Microsoft.Skills.Text.NamedEntityRecognitionSkill

### Dernière version d'API disponible

2017-11-11-Preview

### Fin de la prise en charge

15 février 2019

### Recommandations

Utilisez [Microsoft.Skills.Text.EntityRecognitionSkill](#) à la place. Il fournit la plupart des fonctionnalités de NamedEntityRecognitionSkill à un meilleur niveau de qualité. Il détient également des informations plus riches dans ses champs de sortie complexes.

Pour migrer vers la [compétence de reconnaissance des entités](#), vous devrez apporter une ou plusieurs des modifications suivantes à votre définition de compétence. Vous pouvez mettre à jour la définition de compétence à l'aide de l'[API de mise à jour de compétences](#).

#### NOTE

actuellement, le score de confiance comme concept n'est pas pris en charge. Le paramètre `minimumPrecision` existe sur `EntityRecognitionSkill` pour une utilisation ultérieure et la compatibilité descendante.

1. *(Obligatoire)* Modifiez `@odata.type` en remplaçant `"#Microsoft.Skills.Text.NamedEntityRecognitionSkill"` par `"#Microsoft.Skills.Text.EntityRecognitionSkill"`.
2. *(Facultatif)* Si vous utilisez la sortie `entities`, utilisez la sortie de collection complexe `namedEntities` issue de `EntityRecognitionSkill` à la place. Vous pouvez utiliser `targetName` dans la définition de compétence pour le mapper à une annotation appelée `entities`.
3. *(Facultatif)* Si vous ne spécifiez pas explicitement `categories`, `EntityRecognitionSkill` peut retourner un type différent de catégories en plus de celles qui étaient prises en charge par `NamedEntityRecognitionSkill`. Si ce comportement est indésirable, veillez à définir explicitement le paramètre `categories` sur

```
["Person", "Location", "Organization"] .
```

### *Exemples de définitions de migration*

- Migration simple

#### *(Avant) Définition de compétence NamedEntityRecognition*

```
{
    "@odata.type": "#Microsoft.Skills.Text.NamedEntityRecognitionSkill",
    "categories": [ "Person" ],
    "defaultLanguageCode": "en",
    "inputs": [
        {
            "name": "text",
            "source": "/document/content"
        }
    ],
    "outputs": [
        {
            "name": "persons",
            "targetName": "people"
        }
    ]
}
```

#### *(Après) Définition de compétence EntityRecognition*

```
{
    "@odata.type": "#Microsoft.Skills.Text.EntityRecognitionSkill",
    "categories": [ "Person" ],
    "defaultLanguageCode": "en",
    "inputs": [
        {
            "name": "text",
            "source": "/document/content"
        }
    ],
    "outputs": [
        {
            "name": "persons",
            "targetName": "people"
        }
    ]
}
```

- Migration légèrement compliquée

#### *(Avant) Définition de compétence NamedEntityRecognition*

```
{
    "@odata.type": "#Microsoft.Skills.Text.NamedEntityRecognitionSkill",
    "defaultLanguageCode": "en",
    "minimumPrecision": 0.1,
    "inputs": [
        {
            "name": "text",
            "source": "/document/content"
        }
    ],
    "outputs": [
        {
            "name": "persons",
            "targetName": "people"
        },
        {
            "name": "entities"
        }
    ]
}
```

(Après) Définition de compétence EntityRecognition

```
{
    "@odata.type": "#Microsoft.Skills.Text.EntityRecognitionSkill",
    "categories": [ "Person", "Location", "Organization" ],
    "defaultLanguageCode": "en",
    "minimumPrecision": 0.1,
    "inputs": [
        {
            "name": "text",
            "source": "/document/content"
        }
    ],
    "outputs": [
        {
            "name": "persons",
            "targetName": "people"
        },
        {
            "name": "namedEntities",
            "targetName": "entities"
        }
    ]
}
```

## Voir aussi

- [Compétences prédéfinies](#)
- [Guide pratique pour définir un ensemble de compétences](#)
- [Compétence de reconnaissance des entités](#)

# Compétence cognitive Reconnaissance d'entités nommées

04/10/2020 • 4 minutes to read • [Edit Online](#)

La compétence **Reconnaissance d'entités nommées** extrait les entités nommées du texte. Sont notamment disponibles les types d'entités suivants : `person`, `location` et `organization`.

## IMPORTANT

La compétence de reconnaissance des entités nommées est désormais remplacée par [Microsoft.Skills.Text.EntityRecognitionSkill](#). La prise en charge a pris fin le 15 février 2019 et l'API a été supprimée du produit le 2 mai 2019. Suivez les recommandations de la page [Compétences de recherche cognitive déconseillées](#) pour migrer vers une compétence prise en charge.

## NOTE

Si vous élargissez le champ en augmentant la fréquence des traitements, en ajoutant des documents supplémentaires ou en ajoutant plusieurs algorithmes d'IA, vous devez [attacher une ressource Cognitive Services facturable](#). Des frais s'appliquent durant l'appel des API dans Cognitive Services ainsi que pour l'extraction d'images dans le cadre de la phase de craquage de document de la Recherche cognitive Azure. L'extraction de texte à partir des documents est gratuite.

L'exécution des compétences intégrées est facturée au prix actuel du [paiement à l'utilisation de Cognitive Services](#). Les prix appliqués pour l'extraction d'images sont présentés sur la [page de tarification du service Recherche cognitive Azure](#).

## @odata.type

`Microsoft.Skills.Text.NamedEntityRecognitionSkill`

## Limites de données

La taille maximale d'un enregistrement doit être de 50 000 caractères telle que mesurée par `String.Length`. Si vous devez subdiviser vos données avant de les envoyer à l'extracteur de phrases clés, envisagez d'utiliser la [compétence Fractionnement de texte](#).

## Paramètres de la compétence

Les paramètres respectent la casse.

NOM DU PARAMÈTRE	DESCRIPTION
categories	Tableau des catégories à extraire. Types de catégories possibles : <code>"Person"</code> , <code>"Location"</code> , <code>"Organization"</code> . Si aucune catégorie n'est précisée, tous les types sont retournés.
defaultLanguageCode	Code de langue du texte d'entrée. Langues prises en charge : <code>de</code> , <code>en</code> , <code>es</code> , <code>fr</code> , <code>it</code> .

NOM DU PARAMÈTRE	DESCRIPTION
minimumPrecision	Nombre compris entre 0 et 1. Si la précision est inférieure à cette valeur, l'entité n'est pas retournée. La valeur par défaut est 0.

## Entrées de la compétence

NOM D'ENTRÉE	DESCRIPTION
languageCode	facultatif. La valeur par défaut est "en".
text	Texte à analyser.

## Sorties de la compétence

NOM DE SORTIE	DESCRIPTION
persons	Tableau de chaînes représentant chacune le nom d'une personne.
locations	Tableau de chaînes représentant chacune un lieu.
organizations	Tableau de chaînes représentant chacune une organisation.
entities	Tableau de types complexes. Chaque type complexe contient les champs suivants : <ul style="list-style-type: none"> <li>• la catégorie ("person", "organization" ou "location") ;</li> <li>• la valeur (le nom réel de l'entité) ;</li> <li>• le décalage (l'emplacement où elle a été trouvée dans le texte) ;</li> <li>• la confiance (une valeur comprise entre 0 et 1 représentant la confiance accordée à la valeur en tant qu'entité réelle).</li> </ul>

## Exemple de définition

```
{  
    "@odata.type": "#Microsoft.Skills.Text.NamedEntityRecognitionSkill",  
    "categories": [ "Person", "Location", "Organization"],  
    "defaultLanguageCode": "en",  
    "inputs": [  
        {  
            "name": "text",  
            "source": "/document/content"  
        }  
    ],  
    "outputs": [  
        {  
            "name": "persons",  
            "targetName": "people"  
        }  
    ]  
}
```

## Exemple d'entrée

```
{  
    "values": [  
        {  
            "recordId": "1",  
            "data":  
                {  
                    "text": "This is the loan application for Joe Romero, a Microsoft employee who was born in Chile  
and who then moved to Australia... Ana Smith is provided as a reference.",  
                    "languageCode": "en"  
                }  
        }  
    ]  
}
```

## Exemple de sortie

```
{
  "values": [
    {
      "recordId": "1",
      "data" :
      {
        "persons": [ "Joe Romero", "Ana Smith"],
        "locations": [ "Chile", "Australia"],
        "organizations": [ "Microsoft"],
        "entities":
        [
          {
            "category": "person",
            "value": "Joe Romero",
            "offset": 33,
            "confidence": 0.87
          },
          {
            "category": "person",
            "value": "Ana Smith",
            "offset": 124,
            "confidence": 0.87
          },
          {
            "category": "location",
            "value": "Chile",
            "offset": 88,
            "confidence": 0.99
          },
          {
            "category": "location",
            "value": "Australia",
            "offset": 112,
            "confidence": 0.99
          },
          {
            "category": "organization",
            "value": "Microsoft",
            "offset": 54,
            "confidence": 0.99
          }
        ]
      }
    }
  ]
}
```

## Cas d'erreur

Si le code de langue du document n'est pas pris en charge, une erreur est retournée et aucune entité n'est extraite.

## Voir aussi

- [Compétences prédéfinies](#)
- [Guide pratique pour définir un ensemble de compétences](#)
- [Compétence de reconnaissance des entités](#)

# Définitions intégrées Azure Policy pour Recherche cognitive Azure

04/10/2020 • 2 minutes to read • [Edit Online](#)

Cette page est un index des définitions de stratégie intégrées d'[Azure Policy](#) pour Recherche cognitive Azure. Pour obtenir des éléments intégrés supplémentaires d'Azure Policy pour d'autres services, consultez [Définitions intégrées d'Azure Policy](#).

Le nom de chaque définition de stratégie intégrée est un lien vers la définition de la stratégie dans le portail Azure. Utilisez le lien figurant dans la colonne **Version** pour voir la source dans le [dépôt GitHub Azure Policy](#).

## Recherche cognitive Azure

NOM (PORTAIL AZURE)	DESCRIPTION	EFFET(S)	VERSION (GITHUB)
<a href="#">Déployer les paramètres de diagnostic des services de recherche sur Event Hub</a>	Déploie les paramètres de diagnostic des services de recherche à envoyer en streaming à un hub d'événements régional quand un service de recherche nouveau ou mis à jour n'a pas ces paramètres de diagnostic.	DeployIfNotExists, Désactivé	2.0.0
<a href="#">Déployer les paramètres de diagnostic des services de recherche sur l'espace de travail Log Analytics</a>	Déploie les paramètres de diagnostic des services de recherche à envoyer en streaming à un espace de travail Log Analytics régional quand un service de recherche nouveau ou mis à jour n'a pas ces paramètres de diagnostic.	DeployIfNotExists, Désactivé	1.0.0
<a href="#">Les journaux de diagnostic dans les services Search doivent être activés</a>	Auditer l'activation des journaux de diagnostic Permet de recréer les pistes d'activité à utiliser à des fins d'investigation en cas d'incident de sécurité ou de compromission du réseau.	AuditIfNotExists, Désactivé	3.0.0

## Étapes suivantes

- Consultez les définitions intégrées dans le [dépôt Azure Policy de GitHub](#).
- Consultez la [Structure de définition Azure Policy](#).
- Consultez la page [Compréhension des effets de Policy](#).

# Ressources de documentation pour l'enrichissement de l'IA dans la recherche cognitive Azure

04/10/2020 • 3 minutes to read • [Edit Online](#)

L'enrichissement de l'IA est un module complémentaire associé à l'indexation basée sur un indexeur qui trouve des informations latentes dans des sources non textuelles et le texte indifférencié, et les transforme en contenu avec possibilité de recherche en texte intégral dans la recherche cognitive Azure.

Pour le traitement intégré, les modèles AI préentraînés dans Cognitive Services sont appelés en interne pour effectuer les analyses. Vous pouvez également intégrer des modèles personnalisés à l'aide d'Azure Machine Learning, Azure Functions ou d'autres approches.

Vous trouverez ci-dessous une liste consolidée de la documentation sur l'enrichissement par IA.

## Concepts

- [Enrichissements par IA](#)
- [Ensemble de compétences](#)
- [Sessions de débogage](#)
- [Bases de connaissances](#)
- [Projections](#)
- [Enrichissement incrémentiel \(réutilisation d'un document enrichi mis en cache\)](#)

## Procédures pratiques

- [Démarrage rapide : Créer un ensemble de compétences cognitives dans le Portail Azure](#)
- [Tutoriel : Indexation enrichie avec l'IA](#)
- [Tutoriel : Diagnostiquer, réparer et valider les changements apportés à votre ensemble de compétences avec des sessions de débogage](#)

## Bases de connaissances

- [Démarrage rapide : Créer une base de connaissances dans le portail Azure](#)
- [Créer une base de connaissances à l'aide de REST et Postman](#)
- [Visualiser une base de connaissances avec l'Explorateur Stockage](#)
- [Connecter une base de connaissances à Power BI](#)
- [Exemples de projection \(comment mettre en forme et exporter des enrichissements\)](#)

## Compétences personnalisées (avancé)

- [Guide pratique pour définir une interface de compétences personnalisées](#)
- [Exemple : Créer une qualification personnalisée à l'aide d'Azure Functions \(et des API Recherche d'entités Bing\)](#)
- [Exemple : Créer une compétence personnalisée avec Python](#)
- [Exemple : Créer une compétence personnalisée à l'aide de Form Recognizer](#)
- [Exemple : Créer une compétence personnalisée avec Azure Machine Learning](#)

## Guides pratiques

- Attacher une ressource Cognitive Services
- [How to create a skillset in an enrichment pipeline](#) (Créer un ensemble de compétences dans un pipeline d'enrichissement)
- référencer des annotations dans un ensemble de compétences
- Mapper des champs sur un index
- Traiter et extraire des informations à partir d'images
- Configurer la mise en cache pour l'enrichissement incrémentiel
- Guide pratique pour régénérer un index de recherche cognitive Azure
- Conseils de conception
- Erreurs et avertissements courants

## Informations de référence sur les compétences

- Compétences prédéfinies
  - [Microsoft.Skills.Text.KeyPhraseExtractionSkill](#)
  - [Microsoft.Skills.Text.LanguageDetectionSkill](#)
  - [Microsoft.Skills.Text.EntityRecognitionSkill](#)
  - [Microsoft.Skills.Text.MergeSkill](#)
  - [Microsoft.Skills.Text.PIIDetectionSkill](#)
  - [Microsoft.Skills.Text.SplitSkill](#)
  - [Microsoft.Skills.Text.SentimentSkill](#)
  - [Microsoft.Skills.Text.TranslationSkill](#)
  - [Microsoft.Skills.Vision.ImageAnalysisSkill](#)
  - [Microsoft.Skills.Vision.OcrSkill](#)
  - [Microsoft.Skills.Util.ConditionalSkill](#)
  - [Microsoft.Skills.Util.DocumentExtractionSkill](#)
  - [Microsoft.Skills.Util.ShaperSkill](#)
- Compétences personnalisées
  - [Microsoft.Skills.Custom.AmlSkill](#)
  - [Microsoft.Skills.Custom.WebApiSkill](#)
- Compétences dépréciées
  - [Microsoft.Skills.Text.NamedEntityRecognitionSkill](#)

## API

- REST API
  - [Créer un ensemble de compétences \(api-version=2020-06-30\)](#)
  - [Créer un indexeur \(api-version=2020-06-30\)](#)

## Voir aussi

- [API REST Recherche cognitive Azure](#)
- [Indexeurs dans Recherche cognitive Azure](#)
- [Qu'est-ce que la recherche cognitive Azure ?](#)

# Recherche cognitive Azure – Questions fréquentes (FAQ)

04/10/2020 • 13 minutes to read • [Edit Online](#)

Trouvez la réponse aux questions les plus fréquemment posées sur les concepts, le code et les scénarios liés à la Recherche cognitive Azure.

## Plateforme

### **En quoi la Recherche cognitive Azure est-elle différente de la recherche en texte intégral de mon système de gestion de base de données (SGBD) ?**

La Recherche cognitive Azure comprend les fonctionnalités suivantes : prise en charge de plusieurs sources de données, [analyse linguistique de nombreuses langues](#), [analyse personnalisée des entrées de données intéressantes et inhabituelles](#), contrôles de classement des recherches par le biais de [profils de score](#), tampon clavier, mise en surbrillance des résultats et navigation par facettes. Elle comprend également d'autres fonctionnalités, telles que les synonymes et une syntaxe de requête riche, toutefois ces fonctionnalités ne lui sont pas propres.

### **Puis-je suspendre le service Recherche cognitive Azure et arrêter la facturation ?**

Vous ne pouvez pas suspendre le service. Lorsque le service est créé, les ressources de calcul et de stockage sont allouées pour votre utilisation exclusive. Il n'est pas possible de libérer des ressources à la demande.

## Opérations d'indexation

### **Déplacer, sauvegarder et restaurer des index ou des instantanés d'index ?**

Pendant la phase de développement, vous souhaiterez peut-être déplacer votre index entre les services de recherche. Par exemple, vous pouvez utiliser un niveau tarifaire De base ou Gratuit pour développer votre index, puis le déplacer vers le niveau Standard ou vers un niveau supérieur pour une utilisation en production.

Vous pouvez aussi sauvegarder un instantané d'index dans des fichiers qui peuvent être utilisés pour le restaurer ultérieurement.

Vous pouvez effectuer toutes ces opérations avec l'exemple de code [index-backup-restore](#) dans cet [exemple de dépôt .NET Recherche cognitive Azure](#).

Vous pouvez également [obtenir une définition d'index](#) à tout moment à l'aide de l'API REST Recherche cognitive Azure.

Il n'existe aucune fonctionnalité intégrée d'extraction d'index, de capture instantanée ou de restauration de sauvegarde dans le portail Azure. Toutefois, nous envisageons d'ajouter les fonctionnalités de sauvegarde et de restauration dans une version future. Si vous souhaitez soutenir cette fonctionnalité, votez sur [UserVoice](#).

### **Puis-je restaurer mon index ou mon service une fois qu'il est supprimé ?**

Non, si vous supprimez un index ou un service Recherche cognitive Azure, il ne peut pas être récupéré. Quand vous supprimez un service Recherche cognitive Azure, tous les index dans le service sont supprimés définitivement. Si vous supprimez un groupe de ressources Azure qui contient un ou plusieurs services Recherche cognitive Azure, tous les services sont supprimés définitivement.

La recréation des ressources telles que les index, les indexeurs, les sources de données et les compétences, nécessite de les recréer à partir du code.

Pour recréer un index, vous devez réindexer les données à partir de sources externes. Pour cette raison, nous vous

recommandons de conserver une copie principale ou une sauvegarde des données d'origine dans un autre magasin de données comme Azure SQL Database ou Cosmos DB.

Vous pouvez également utiliser l'exemple de code [index-backup-restore](#) dans cet [exemple de dépôt .NET Recherche cognitive Azure](#) pour sauvegarder la définition et l'instantané d'un index dans une série de fichiers JSON. Plus tard, vous pourrez utiliser l'outil et les fichiers pour restaurer l'index, si nécessaire.

### Puis-je effectuer une indexation à partir de réplicas SQL Database ? (s'applique aux [indexeurs Azure SQL Database](#))

Il n'existe aucune restriction quant à l'utilisation d'un réplica principal ou secondaire comme source de données lorsque vous créez un index à partir de zéro. Toutefois, pour que l'index soit actualisé avec des mises à jour incrémentielles (basées sur les enregistrements modifiés), le réplica principal est nécessaire. Cela vient du fait que SQL Database assure uniquement le suivi des modifications sur les réplicas principaux. Si vous essayez d'utiliser des réplicas secondaires pour une charge de travail d'actualisation d'index, il n'est pas garanti que vous obteniez toutes les données.

## Opérations de recherche

### Puis-je effectuer une recherche sur plusieurs index ?

Non, cette opération n'est pas prise en charge. La recherche est toujours limitée à un seul index.

### Puis-je limiter l'accès à l'index de recherche en fonction de l'identité de l'utilisateur ?

Vous pouvez implémenter des [filtres de sécurité](#) avec le filtre `search.in()`. Le filtre s'adapte parfaitement aux [services de gestion d'identité comme Azure Active Directory\(AAD\)](#) afin de réduire les résultats de recherche en fonction de l'appartenance à un groupe d'utilisateurs défini.

### Pourquoi n'ai-je aucun résultat pour des termes dont je sais qu'ils sont valides ?

Dans la plupart des cas, l'utilisateur ne sait pas que chaque type de requête prend en charge des comportements de recherche et des niveaux d'analyse linguistique différents. La recherche en texte intégral, qui représente la principale charge de travail, comprend une phase d'analyse linguistique qui ne conserve que la racine des termes entrés. Cet aspect de l'analyse de requête permet d'élargir le nombre de correspondances possibles, car le terme peut ainsi correspondre à un plus grand nombre de variantes.

Toutefois, les requêtes de caractère générique, les requêtes partielles et les requêtes d'expression régulière ne sont pas analysées comme les requêtes de terme ou d'expression normales et peuvent entraîner un rappel médiocre si la requête ne correspond pas à la forme analysée du mot dans l'index de recherche. Pour plus d'informations sur l'analyse des requêtes et l'analyse lexicale, consultez [l'architecture de requêtes](#).

### Mes recherches de caractère générique sont lentes.

La plupart des requêtes de recherche de caractère générique, comme les requêtes de préfixe, partielle et d'expression régulière, sont réécrites en interne avec des termes correspondants dans l'index de recherche. Ce traitement supplémentaire d'analyse de l'index de recherche augmente la latence. Par ailleurs, les requêtes de recherche large, comme `a*`, qui sont susceptibles d'être réécrites avec de nombreux termes, peuvent être très lentes. Pour des recherches performantes avec caractère générique, définissez un [analyseur personnalisé](#).

### Pourquoi le score du classement de recherche est-il systématiquement égal à 1.0 pour tous les résultats ?

Par défaut, les résultats de la recherche sont notés en fonction des [propriétés statistiques des termes correspondants](#), et sont classés du score le plus haut vers le score le plus bas. Cependant, certains types de requête (caractère générique, préfixe, expression régulière) contribuent toujours à un score constant dans le score général du document. Ce comportement est normal. La Recherche cognitive Azure impose un score constant pour permettre aux correspondances trouvées par le biais de l'extension de requête d'être incluses dans les résultats, sans affecter le classement.

Par exemple, supposons l'entrée « tour\* » dans une recherche par caractères génériques, qui retourne les résultats « tours », « tourettes » et « tourmaline ». Étant donné la nature de ces résultats, il est impossible de déduire

raisonnablement quels termes sont plus utiles que d'autres. Pour cette raison, nous ignorons les fréquences de terme lors de la notation des résultats dans les requêtes avec caractère générique, préfixe et expression régulière. Les résultats de recherche basés sur une entrée partielle reçoivent un score constant afin d'éviter que des résultats inattendus ne soient retournés.

## Opérations d'ensemble de compétences

### **Y a-t-il des conseils ou des astuces pour réduire les frais liés aux services cognitifs lors de l'ingestion ?**

Nous comprenons bien que vous ne souhaitez pas exécuter des compétences intégrées ou des compétences personnalisées plus que ce qui est absolument nécessaire, en particulier si vous avez des millions de documents à traiter. Dans ce souci, nous avons ajouté des fonctionnalités d'« enrichissement incrémentiel » à l'exécution d'ensemble de compétences. En résumé, vous pouvez fournir un emplacement de cache (une chaîne de connexion de stockage blob) qui sera utilisé pour stocker la sortie des étapes d'enrichissement « intermédiaires ». Cela permet de faire en sorte que le pipeline d'enrichissement soit intelligent et n'applique que les enrichissements nécessaires lorsque vous modifiez votre ensemble de compétences. Cela permet également de réduire le temps d'indexation puisque le pipeline est plus efficace.

En savoir plus sur l'[enrichissement incrémentiel](#)

## Modèles de conception

### **Quelle est la meilleure méthode pour implémenter une recherche localisée ?**

La plupart des clients choisissent des champs dédiés plutôt qu'une collection lorsque plusieurs paramètres régionaux (langues) doivent être pris en charge dans un même index. Les champs spécifiques aux paramètres régionaux permettent d'attribuer un analyseur adapté. Par exemple, vous pouvez affecter l'analyseur de français Microsoft à un champ contenant des chaînes en langue française. Cela simplifie également le filtrage. Si vous savez qu'une requête est exécutée sur une page fr-fr, vous pouvez limiter les résultats de la recherche à ce champ. Sinon, vous pouvez aussi créer un [profil de score](#) afin de donner au champ un poids plus relatif. Le service Recherche cognitive Azure prend en charge plus de [50 analyseurs de langue](#) sélectionnables.

## Étapes suivantes

Votre question concerne-t-elle une fonctionnalité manquante ? Demandez cette fonctionnalité sur le [site web UserVoice](#).

## Voir aussi

[StackOverflow : Recherche cognitive Azure](#)

[Fonctionnement de la recherche en texte intégral dans la Recherche cognitive Azure](#)

[Qu'est-ce que la recherche cognitive Azure ?](#)