

Présentation d'Azure pour les développeurs

Découvrez comment créer des applications à l'aide d'Azure.

Vue d'ensemble

VUE D'ENSEMBLE

[Azure for developers](#)

[Services Azure clés pour les développeurs](#)

[Héberger des applications sur Azure](#)

[applications Connecter aux services Azure](#)

[Créer des ressources Azure](#)

[Concepts clés pour la création d'applications Azure](#)

[Facturation Azure](#)

Principaux scénarios de développement

CONCEPT

[Développer des applications qui utilisent des services Azure AI](#)

[Connexions sans mot de passe pour les services Azure](#)

Centres de développement

VUE D'ENSEMBLE

[.NET sur Azure](#)

[Java sur Azure](#)

[JavaScript sur Azure](#)

[Python sur Azure](#)

[Accéder à Azure](#)

Vue d'ensemble d'Azure pour les développeurs

Article • 23/10/2023

Azure est une plateforme cloud conçue pour simplifier le processus de création d'applications modernes. Que vous choisissez d'héberger vos applications entièrement dans Azure ou d'étendre vos applications locales avec des services Azure, Azure vous aide à créer des applications évolutives, fiables et gérables.

support Azure les langages de programmation les plus populaires utilisés aujourd'hui, notamment Python, JavaScript, Java, .NET et Go. Avec une bibliothèque de SDK complète et une prise en charge étendue dans les outils que vous utilisez déjà comme VS Code, Visual Studio, IntelliJ et Eclipse, Azure est conçu pour tirer parti des compétences dont vous disposez déjà et vous rendre productif immédiatement.

Scénarios de développement d'applications sur Azure

Vous pouvez incorporer Azure dans votre application de différentes manières en fonction de vos besoins.

[https://www.microsoft.com/fr-fr/videoplayer/embed/RE50LmJ?postJsIMsg=true&autoCaptions=fr-fr ↗](https://www.microsoft.com/fr-fr/videoplayer/embed/RE50LmJ?postJsIMsg=true&autoCaptions=fr-fr)

- **Hébergement d'applications sur Azure** - Azure peut héberger l'ensemble de votre pile d'applications, des applications web et des API aux bases de données en passant par les services de stockage. Azure prend en charge divers modèles d'hébergement, des services entièrement managés aux conteneurs, en passant par les machines virtuelles. Lorsque vous utilisez des services Azure entièrement managés, vos applications peuvent tirer parti de la scalabilité, de la haute disponibilité et de la sécurité intégrées à Azure.
- **Consommation de services cloud à partir d'applications locales existantes** : les applications locales existantes peuvent incorporer des services Azure pour étendre leurs fonctionnalités. Par exemple, une application peut utiliser Stockage Blob Azure pour stocker des fichiers dans le cloud, Azure Key Vault pour stocker en toute sécurité les secrets d'application ou [Recherche cognitive Azure](#) pour ajouter une fonctionnalité de recherche en texte intégral. Ces services sont entièrement gérés par Azure et peuvent être facilement ajoutés à vos applications existantes

sans modifier votre architecture d'application ou votre modèle de déploiement actuel.

- **Architectures basées sur des conteneurs** : Azure fournit un large éventail de services basés sur des conteneurs pour prendre en charge votre parcours de modernisation des applications. Que vous ayez besoin d'un registre privé pour vos images conteneur, conteneurisez une application existante pour faciliter le déploiement, déployer des applications basées sur des microservices ou gérer des conteneurs à grande échelle, Azure propose des solutions qui prennent en charge vos besoins.
- **Architectures serverless modernes** : Azure Functions simplifie la génération de solutions pour gérer les flux de travail pilotés par les événements, qu'il s'agisse de répondre aux requêtes HTTP, de gérer les chargements de fichiers dans le stockage Blob ou de traiter les événements dans une file d'attente. Vous écrivez uniquement le code nécessaire pour gérer votre événement sans vous soucier des serveurs ou du code de l'infrastructure. En outre, vous pouvez tirer parti de plus de 250 connecteurs vers d'autres services Azure et tiers pour résoudre vos problèmes d'intégration les plus difficiles.

Services Azure clés pour les développeurs

Article • 22/11/2023

Cet article présente certains des services Azure clés utilisés le plus fréquemment en tant que développeur. Pour obtenir la liste complète de tous les services Azure, consultez la [page](#) du hub de documentation Azure.

Hébergement et calcul des applications

Service	Description
 Azure App Service	Hébergez les applications et API web .NET, Java, Node.js et Python dans un service Azure entièrement managé. Vous devez uniquement déployer votre code sur Azure. Azure s'occupe de toutes les gestions de l'infrastructure, telles que la haute disponibilité, l'équilibrage de charge et la mise à l'échelle automatique.
 Azure Static Web Apps	Hébergez des applications web statiques créées à l'aide de frameworks tels que Gatsby, Hugo ou VuePress, ou des applications web modernes créées à l'aide d'Angular, React, Svelte ou Vue. Les applications web statiques créent et déplient automatiquement en fonction des modifications de code et de l'intégration de l'API des fonctionnalités à Azure Functions.
 Azure Functions	Plateforme de calcul serverless permettant de créer de petits segments de code discrets pouvant être déclenchés à partir de différents événements. Les applications courantes incluent la création d'API serverless ou l'orchestration d'architectures de lecteur d'événements.
 Azure Container Instances	Exécutez des conteneurs Docker à la demande dans un environnement Azure serverless géré. Azure Container Instances est une solution adaptée à tous les scénarios qui peut fonctionner dans des conteneurs isolés, sans orchestration.
 Azure Kubernetes Services	Déployez rapidement un cluster Kubernetes prêt pour la production sur le cloud et déchargez la surcharge opérationnelle sur Azure. Azure gère les tâches critiques, comme le monitoring de l'intégrité et la maintenance. Vous devez uniquement gérer et gérer les nœuds de l'agent.
 Azure Spring Apps	Héberger des applications de microservice Spring Boot dans Azure, aucune modification du code n'est requise. Azure Spring Apps fournit la supervision, la gestion de la configuration, la découverte de services, l'intégration CI/CD et bien plus encore.

Service	Description
 Machines virtuelles Azure	Hébergez votre application à l'aide de machines virtuelles dans Azure quand vous avez besoin d'un meilleur contrôle sur votre environnement informatique. Les machines virtuelles Azure offrent un environnement informatique flexible et évolutif pour les machines virtuelles Linux et Windows.

Azure AI services

Les services Azure AI vous aident à créer des applications intelligentes avec des API et des modèles prédéfinis et personnalisables. Les exemples d'applications incluent le traitement en langage naturel des conversations, la recherche, l'analyse, la traduction, le message, la vision et la prise de décision.

Service	Description
 Azure OpenAI	Utilisez des modèles de langage puissants, notamment la série de modèles GPT-3, Codex et Embeddings pour la génération de contenu, la synthèse, la recherche sémantique et le langage naturel pour la traduction de code.
 Azure AI Speech	Transcrivez la parole audible en texte lisible, pouvant faire l'objet d'une recherche ou convertissez du texte en paroles réalistes pour des interfaces plus naturelles.
 Azure AI Language	Utilisez le traitement en langage naturel (NLP) pour identifier les expressions clés et effectuer une analyse des sentiments à partir du texte.
 Azure AI Traducteur	Traduisez plus de 100 langues et dialectes.
 Azure AI Vision	Analysez le contenu des images et vidéos.
 Azure AI Search	Récupération d'informations à grande échelle pour les applications de recherche traditionnelles et conversationnelles, avec sécurité et options pour l'enrichissement par IA et la vectorisation.
 Azure AI Document Intelligence	Service d'extraction de documents qui comprend vos formulaires vous permettant d'extraire rapidement du texte et de la structure à partir de documents.

Données

Service	Description
 Azure SQL	Famille de produits de moteur de base de données SQL Server dans le cloud.
 Azure SQL Database	Version entièrement managée et basée sur le cloud de SQL Server.
 Azure Cosmos DB	Une base de données NoSQL entièrement managée basée sur le cloud. Azure Cosmos DB propose plusieurs API, notamment les API compatibles MongoDB , Cassandra et Gremlin .
 Base de données Azure pour PostgreSQL	Un service de base de données PostgreSQL entièrement managé basé sur le cloud basé sur PostgreSQL Community Edition.
 Azure Database pour MySQL	Un service de base de données MySQL entièrement managé et basé sur le cloud basé sur MySQL dans MySQL Community Edition.
 Azure Database for MariaDB	Un service de base de données MariaDB entièrement managé et basé sur le cloud basé sur l'édition de la communauté MariaDB.
 Cache Azure pour Redis	Un cache de données sécurisé et un répartiteur de messagerie qui fournit un débit élevé et un accès à faible latence aux données pour les applications.

Stockage

Stockage Azure produits offrent des services de stockage de données cloud sécurisés et évolutifs et hybrides. Les offres incluent des services pour les solutions de stockage hybride et les services pour transférer, partager et sauvegarder des données.

Service	Description
 Stockage Blob Azure	Stockage Blob Azure permet à vos applications de stocker et de récupérer des fichiers dans le cloud. Stockage Azure est hautement évolutif pour stocker de grandes quantités de données de manière redondante pour garantir la haute disponibilité.
 Azure Data Lake Storage	Azure Data Lake Storage est conçu pour prendre en charge l'analytique big data en fournissant un stockage évolutif et rentable pour les données structurées, semi-structurées ou non structurées.

Messagerie

Il s'agit de certains des services les plus populaires qui gèrent l'envoi, la réception et le routage des messages depuis et vers des applications.

Service	Description
 Azure Service Bus	Un répartiteur de messages d'entreprise entièrement géré prenant en charge les intégrations de point à point et de publication-abonnement. Il est idéal pour créer des applications découplées, un nivelingement de charge basé sur la file d'attente ou faciliter la communication entre les microservices.
 Azure Event Hubs	Azure Event Hubs est un service géré qui peut ingérer et traiter des flux de données massifs à partir de sites web, d'applications ou d'appareils.
 Stockage File d'attente Azure	File d'attente simple et fiable qui peut gérer des charges de travail volumineuses.

Identité et sécurité

Service	Description
 Microsoft Entra ID	Gérez les identités utilisateur et contrôlez l'accès à vos applications, données et ressources.
 Azure Key Vault	Stockez et accédez aux secrets d'application tels que les chaîne de connexion et les clés API dans un coffre chiffré avec un accès restreint pour vous assurer que vos secrets et votre application ne sont pas compromis.
 Configuration de l'application	Un service rapide et évolutif pour gérer de manière centralisée les paramètres d'application et les indicateurs de fonctionnalités.

Gestion

Service	Description
 Azure Monitor	Solution de supervision complète pour la collecte, l'analyse et la réponse aux données de surveillance à partir de vos environnements cloud et locaux.
 Application Insights	Cette fonctionnalité d'Azure Monitor fournit la gestion des performances des applications (APM) pour améliorer les performances, la fiabilité et la qualité de vos applications web actives.

Hébergement d'applications sur Azure

Article • 09/02/2024

Azure propose différentes façons d'héberger votre application en fonction de vos besoins. Cet article suggère des services correspondant aux exigences. Ce n'est pas proscriptif. Vous pouvez combiner et mettre en correspondance des services pour répondre à vos besoins. La plupart des environnements de production utilisent une combinaison de services pour répondre à leurs besoins métier et organisationnels.

[https://www.microsoft.com/fr-fr/videoplayer/embed/RE50vLy?
postJsIMsg=true&autoCaptions=fr-fr ↗](https://www.microsoft.com/fr-fr/videoplayer/embed/RE50vLy?postJsIMsg=true&autoCaptions=fr-fr)

Simplicité et contrôle

Les services d'hébergement Azure sont fournis avec deux considérations :

- **Simplicité et contrôle**
 - Les plateformes d'hébergement simples nécessitent moins de configuration et de gestion, mais offrent moins de contrôle sur l'infrastructure sous-jacente.
 - Les plateformes d'hébergement plus complexes nécessitent davantage de configuration et de gestion, mais fournissent davantage de contrôle sur l'infrastructure sous-jacente.
- **Cloud natif et natif Azure**
 - Le cloud natif peut être considéré comme portable dans le cloud à l'aide de charges de travail open source telles que des conteneurs et des technologies open source telles que Dapr. Les applications que vous créez peuvent être déployées sur n'importe quel fournisseur de cloud.
 - Azure native est spécifique à Azure avec un investissement dans des outils et technologies spécifiques à Azure pour gérer cette infrastructure. Bien que ces services incluent des charges de travail de conteneur, ils incluent également des outils de code first, de faible code et d'infrastructure spécifiques à Azure, en mettant l'accent sur la connexion et l'intégration entre les services Azure.

Hébergement simplifié

Les solutions d'hébergement simplifiées sont entièrement gérées par Azure. Vous êtes responsable des fonctionnalités telles que le code et la configuration de l'environnement. Azure gère le runtime et l'infrastructure sous-jacents, notamment les mises à jour et les correctifs. L'hébergement simplifié est l'approche native Azure.

- [Logic Apps](#) : Créez et exécutez des flux de travail automatisés avec peu à aucun code.
- [Power Automate](#) : utilisez quand vous devez automatiser les processus métier et les flux de travail.
- [Azure Static Web Apps](#) : Déployez des applications web statiques générées telles que Blazor et React.
- [Azure Functions Apps](#) : code serverless ou hébergement de conteneur.

Hébergement équilibré

Les solutions d'hébergement équilibrées équilibrent le besoin de simplicité et le besoin de contrôle. Vous êtes responsable des fonctionnalités telles que le code et la configuration de l'environnement. Azure gère le runtime et l'infrastructure sous-jacents, notamment les mises à jour et les correctifs. Vous pouvez également apporter votre propre conteneur au service. L'hébergement équilibré est natif azure et cloud natif.

- [Azure App Service](#) : hébergement web en service complet, y compris les runtimes de langage, les conteneurs et les charges de travail d'automatisation.
- [Azure Container Apps](#) : hébergement de conteneur serverless.
- [Azure Spring Apps](#) : Migrer des applications Spring Boot vers le cloud Azure.

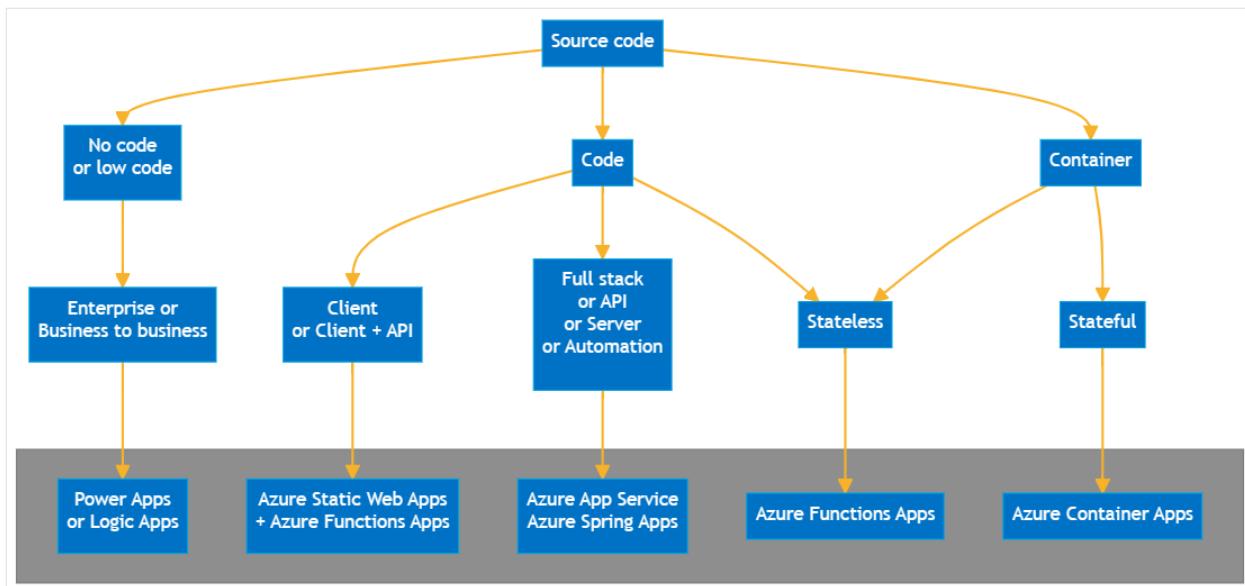
Hébergement contrôlé

Les solutions d'hébergement contrôlées vous donnent un contrôle total sur l'infrastructure sous-jacente. Vous êtes responsable des mises à jour et des correctifs, ainsi que de votre code, de vos ressources et de votre configuration d'environnement. L'hébergement contrôlé est l'approche native cloud.

- [Azure Machines Virtuelles](#) : Contrôle total de la machine virtuelle.
- [Azure Kubernetes Service](#) : Contrôle total du cluster Kubernetes.

Hébergement de code source

Pour les développeurs qui souhaitent démarrer un nouveau développement, utilisez le graphique suivant pour trouver la solution d'hébergement suggérée.



Aucun code ou code faible

Les solutions d'hébergement à faible code font partie de son approche Azure-Cloud.

- [Logic Apps](#) : utilisez un concepteur visuel avec des opérations prédéfinies pour développer un flux de travail pour vos scénarios d'entreprise et d'entreprise.
- [Power Automate](#) tel que [les applications Power](#) : utilisez quand vous devez automatiser les processus métier et les flux de travail au sein de l'organisation Microsoft 365.

Code et conteneur

Les solutions d'hébergement à faible code sont conçues pour vous permettre d'apporter vos fonctionnalités de code sans avoir à gérer l'infrastructure d'application.

- [Azure Static Web Apps](#) : déployez des applications web statiques générées.
- [Azure Functions](#) : déployez des fonctions de code dans des langages pris en charge sans avoir à gérer l'infrastructure d'application.

Les solutions d'hébergement d'abord du code sont conçues pour héberger du code. Vous pouvez déployer votre code directement sur la solution d'hébergement.

- [Azure App Service](#) : hébergement web en service complet.
- [Azure Spring Apps](#) : applications Spring Boot.

Les solutions d'hébergement de premier conteneur sont conçues pour héberger des conteneurs. Le service fournit des options et des fonctionnalités de configuration spécifiques au conteneur. Vous êtes responsable du calcul utilisé à l'intérieur du conteneur. Les services qui hébergent les conteneurs passent d'un contrôle managé à

une responsabilité totale afin que vous preniez uniquement la quantité de gestion des conteneurs souhaitée.

L'hébergement d'orchestration centré sur Kubernetes comprend les éléments suivants :

 Agrandir le tableau

Service	Focus	Utilisation
Azure Kubernetes Service	Cloud natif	Utiliser pour les clusters Kubernetes avec une approche déclarative à l'aide de fichiers de configuration et d'artefacts externes.
Azure Service Fabric	Azure-native	Utilisez une approche impérative pour déployer des microservices sur des clusters de machines. Il fournit un modèle de programmation qui permet aux développeurs d'écrire du code qui décrit l'état souhaité du système, et le runtime Service Fabric s'occupe de faire correspondre le système à cet état.

L'hébergement de conteneur préconfiguré signifie que les options d'orchestration sont préconfigurées pour vous. Votre capacité à communiquer entre des conteneurs ou des clusters de conteneurs peut nécessiter un service supplémentaire tel que [Dapr](#).

 Agrandir le tableau

Service	Utilisation
Azure App Service	Hébergement web en service complet
Azure Spring Apps	Applications Spring Boot
Azure Container Apps	hébergement de conteneur serverless
Azure Container Instances	hébergement simple et unique de conteneurs

Azure fournit un registre de conteneurs pour stocker et gérer vos images conteneur, ou vous pouvez utiliser un registre de conteneurs tiers.

 Agrandir le tableau

Service	Utilisation
Azure Container Registry	Utilisez quand vous générez et hébergez vos propres images conteneur, qui peuvent être déclenchées avec des validations de code source et des mises à jour d'images de base.

Sans serveur

Les solutions d'hébergement **serverless** sont conçues pour exécuter du code sans état, qui inclut un niveau tarifaire basé sur la consommation qui est mis à l'échelle à zéro lorsqu'il n'est pas utilisé.

[Agrandir le tableau](#)

Service	Utilisation
Azure Container Apps	Hébergement de conteneur.
Azure Functions	Hébergement de code ou de conteneur.

Microservices

Les solutions d'hébergement de **microservices** sont conçues pour exécuter de petits services indépendants qui fonctionnent ensemble pour créer une application plus grande. Les microservices sont généralement déployés en tant que conteneurs.

[Agrandir le tableau](#)

Service	Utilisation
Azure Container Apps	Utiliser pour les microservices conteneurisés serverless.
Azure Functions	Utiliser pour le code serverless ou les microservices conteneurisés.

Périphérie cloud

Cloud Edge est un terme qui indique si le service cloud est situé pour bénéficier à l'utilisateur (client) ou à l'application (serveur).

Calcul du client

Le **calcul client** est le calcul qui s'exécute sur le client loin du cloud Azure. Le calcul client est généralement utilisé pour le rendu côté client et le traitement côté client, comme les applications mobiles ou basées sur un navigateur.

[Agrandir le tableau](#)

Service	Utilisation
Azure Static Web Apps	Utiliser pour les applications web statiques qui utilisent le rendu côté client, tel que React, Angular, Svelte, Vue et Blazor.

Disponibilité du client

[\[+\] Agrandir le tableau](#)

Service	Utilisation
Azure Front Door	Utilisez toutes les applications accessibles sur Internet pour fournir un réseau global mis en cache et sécurisé à vos ressources statiques et dynamiques, notamment la protection DDoS, le chiffrement TLS de bout en bout, les pare-feu d'applications et le filtrage géographique.

Calcul du serveur

Les **ressources de calcul** du serveur sont des fichiers traités par le serveur avant d'être servis au client. Les ressources dynamiques sont développées à l'aide du calcul du serveur principal, éventuellement intégrée à d'autres services Azure.

[\[+\] Agrandir le tableau](#)

Service	Utilisation
Azure App Service	Utilisez ce service pour l'hébergement web classique. Cela prend en charge un large ensemble de points de terminaison d'API de fonctionnalités, d'applications de pile complète et de tâches en arrière-plan. Ce service est fourni avec de nombreux runtimes de langage de programmation, ainsi que la possibilité de fournir votre propre pile, langage ou charge de travail à partir d'un conteneur.
Azure Functions	Utilisez ce service pour fournir votre propre code dans les langues prises en charge pour les points de terminaison HTTP ou les déclencheurs basés sur des événements à partir des services Azure.
Azure Spring Apps	Permet de déployer des applications Spring Boot sans modification de code.
Azure Container Apps	Permet d'héberger des microservices managés et des applications conteneurisées sur une plateforme serverless.
Azure Container Instances	Utilisez-le pour des scénarios de conteneur simples qui n'ont pas besoin d'orchestration de conteneur.

Service	Utilisation
Azure Kubernetes Service	Utilisez ce service quand vous avez besoin d'un cluster Kubernetes. Le plan de contrôle permettant de gérer le cluster est créé et fourni pour vous sans frais supplémentaires.

Gestion des points de terminaison de serveur

La gestion des points de terminaison de serveur est la possibilité de gérer votre point de terminaison de serveur et son calcul via une passerelle. Cette passerelle fournit des fonctionnalités telles que le contrôle de version, la mise en cache, la transformation, les stratégies d'API et la surveillance.

 Agrandir le tableau

Service	Utilisation
Gestion des API Azure	Utilisez ce service lorsque vous productisez vos API REST, OpenAPI et GraphQL avec une passerelle d'API, y compris les quotas et les limites de débit, l'authentification et l'autorisation, la transformation et les réponses mises en cache.
Application Gateway Azure	Utiliser pour l' équilibrage de charge régional (couche OSI 7). Il peut être utilisé pour router le trafic en fonction du chemin d'URL ou des en-têtes d'hôte, et prend en charge les fonctionnalités de déchargement SSL, d'affinité de session basée sur les cookies et de pare-feu d'applications web (WAF).
Azure Front Door	Utiliser pour l' équilibrage de charge global (couche OSI 7) pour fournir un réseau global mis en cache et sécurisé à vos ressources statiques et dynamiques, notamment la protection DDoS, le chiffrement TLS de bout en bout, les pare-feu d'applications et le filtrage géographique.
Azure Traffic Manager	Utilisez cette option pour distribuer le trafic par DNS (couche 7) à vos applications publiques dans les régions Azure globales. Traffic Manager utilise le système DNS pour diriger les requêtes des clients vers le point de terminaison de service approprié, en fonction de la méthode de routage du trafic. Il prend en charge différentes méthodes de routage du trafic, telles que la priorité, les performances et le routage géographique. Il est idéal pour gérer le trafic entre plusieurs régions ou centres de données.

Calcul automatisé

Le calcul automatisé est automatisé par un événement tel qu'une planification chronométrée ou un autre service Azure et est généralement utilisé pour le traitement en arrière-plan, le traitement par lots ou les processus de longue durée.

 Agrandir le tableau

Service	Utilisation
Power Automate	Utilisez quand vous devez automatiser les processus métier et les flux de travail.
Azure Functions	Utilisez quand vous devez exécuter du code en fonction d'une planification chronologique ou en réponse à des événements dans d'autres services Azure.
Container Services (Azure Container Instances , Azure Kubernetes Service , Azure Container Apps)	Utiliser pour les charges de travail automatisables standard
Azure Batch	Utilisez quand vous avez besoin d'une automatisation hautes performances.

Cloud hybride

Le cloud hybride est un environnement informatique qui connecte les services de cloud privé locaux d'une entreprise et le cloud public tiers en une infrastructure unique et flexible pour l'exécution des applications et charges de travail de l'organisation.

 Agrandir le tableau

Service	Utilisation
Azure Arc	Utilisez quand vous devez gérer l'ensemble de votre environnement, à la fois les ressources cloud et locales, notamment la sécurité, la gouvernance, l'inventaire et la gestion.

Si vous n'avez pas besoin de gérer votre propre infrastructure, vous pouvez utiliser Azure Stack HCI pour exécuter des machines virtuelles locales.

Calcul haute performance

Le calcul haute performance (HPC) est l'utilisation d'un traitement parallèle pour exécuter des programmes d'applications avancés de manière efficace, fiable et rapide. Le terme s'applique en particulier aux systèmes qui fonctionnent au-dessus d'un teraflop ou 10^{12} opérations à virgule flottante par seconde.

 Agrandir le tableau

Service	Utilisation
Azure Batch	Azure Batch crée et gère un pool de nœuds de calcul (machines virtuelles), installe les applications que vous souhaitez exécuter, et planifie les travaux à exécuter sur les nœuds. Les développeurs peuvent utiliser Azure Batch en tant que service de plateforme pour générer des applications SaaS ou des applications clientes où l'exécution à grande échelle est requise.
Azure BareMetal Instances	Utilisez quand vous devez exécuter dans un environnement nonvirtualisé avec un accès au niveau racine au système d'exploitation, au stockage et au réseau.
Espace de travail Azure Quantum	Utilisez quand vous devez développer et expérimenter des algorithmes quantiques.
Microsoft Genomics	Utiliser pour le traitement génomique conforme à la norme HIPAA certifié ISO.

En savoir plus sur [le calcul hautes performances sur Azure](#).

Calcul basé sur les événements

Le calcul basé sur les événements est le **calcul** déclenché par un événement tel qu'une planification chronologique ou un autre service Azure. Le calcul basé sur les événements est généralement utilisé pour le traitement en arrière-plan, le traitement par lots ou les processus de longue durée.

 Agrandir le tableau

Service	Utilisation
Power Virtual Agents	Utilisez quand vous devez créer des chatbots avec une interface sans code.
Azure Functions	Utilisez quand vous devez exécuter du code en fonction d'une planification chronologique ou en réponse à des événements dans d'autres services Azure.
Messagerie Azure Service Bus	Utilisez quand vous devez dissocier les applications et les services.

Calcul CI/CD

Le calcul CI/CD est le **calcul** utilisé pour générer et déployer votre application.

Service	Description
Azure DevOps	Utilisez Azure DevOps pour une intégration étroite avec le cloud Azure, notamment l'authentification et l'autorisation pour les agents hébergés, qui créent et déploient votre application.
Actions GitHub	Utilisez GitHub Actions pour générer et déployer vos applications de référentiel GitHub. Utilisez Azure CLI pour accéder en toute sécurité à Azure au sein de l'action.
Machines virtuelles Azure	Si vous utilisez un autre système CI/CD, vous pouvez utiliser Azure Machines Virtuelles pour héberger votre système CI/CD.

Ressources relatives à Java

- [Options d'hébergement Java](#)
- [Migration Java vers Azure](#)

Ressources supplémentaires

- [Centre d'architecture Azure : Choisir un service de calcul Azure](#)

Développer des applications qui utilisent des services Azure AI

Article • 07/02/2024

Cet article fournit de la documentation, des exemples et d'autres ressources pour apprendre à développer des applications qui utilisent Azure OpenAI Service et d'autres services Azure AI.

Modèles de référence Azure AI

Les modèles de référence Azure AI vous fournissent des implémentations de référence bien gérées et faciles à déployer. Ils font office de point de départ de haute qualité pour vos applications intelligentes. Les solutions de bout en bout fournissent des applications de référence populaires et complètes. Les blocs de construction sont des exemples à plus petite échelle qui se concentrent sur des scénarios et des tâches spécifiques.

Solutions de bout en bout

[+] Agrandir le tableau

Lien	Description
Bien démarrer avec l'exemple de conversation d'entreprise .NET basé sur RAG	Article qui vous guide tout au long du déploiement et de l'utilisation de l' exemple d'application de conversation d'entreprise pour .NET . Cet exemple est une solution complète de bout en bout qui illustre le modèle RAG (Retrieval-Augmented Generation) exécuté dans Azure, utilisant Recherche Azure AI pour la récupération et les modèles de grand langage Azure OpenAI afin d'alimenter les expériences de type ChatGPT et Q&A.

- [Vidéo de démonstration](#)

Blocs de construction

[+] Agrandir le tableau

Lien	Description
Créer une application de conversation avec	Application Python Quart simple qui diffuse en continu des réponses de ChatGPT à un front-end HTML/JS à l'aide de lignes JSON sur un

Lien	Description
Azure OpenAI (Python) ↗	readableStream. (Le code Python est fourni en tant que référence et peut être adapté à .NET.)
Créer un LangChain avec Azure OpenAI (Python) ↗	Exemple qui montre comment prendre une invite humaine en tant qu'entrée HTTP Get ou Post, calcule les achèvements à l'aide de chaînes d'entrées et de modèles humains. Il s'agit d'un point de départ qui peut être utilisé pour des chaînes plus sophistiquées. (Le code Python est fourni en tant que référence et peut être adapté à .NET.)
Créer un plug-in ChatGPT avec Azure Container Apps (Python) ↗	Exemple qui permet de créer un plug-in ChatGPT à l'aide de GitHub Codespaces, VS Code et Azure. L'exemple inclut des modèles pour déployer le plug-in sur Azure Container Apps à l'aide d'Azure Developer CLI. (Le code Python est fourni en tant que référence et peut être adapté à .NET.)
Galerie de modèles .NET Azure AI ↗	Pour obtenir la liste complète des modèles Azure AI, visitez notre galerie. Tous les modèles d'application de notre galerie peuvent être lancés et déployés à l'aide d'une seule commande : <code>azd up</code> .
Équilibrage de charge intelligent avec Azure Container Apps ↗	Cette solution est créée à l'aide de l'infrastructure de proxy inverse YARP C# ↗ hautes performances de Microsoft. Toutefois, vous n'avez pas besoin de comprendre C# pour l'utiliser. Vous pouvez simplement générer l'image Docker fournie. Il s'agit d'une solution alternative à l'équilibré de charge intelligent OpenAI Gestion des API ↗ , avec la même logique.
Équilibrage de charge intelligent avec Gestion des API Azure ↗	La solution d'entreprise montre comment créer une stratégie Gestion des API Azure pour exposer en toute transparence un point de terminaison unique à vos applications tout en conservant une logique efficace pour consommer au moins deux serveurs principaux OpenAI ou d'API en fonction de la disponibilité et de la priorité.

Azure OpenAI

Solutions de bout en bout

[\[+\] Agrandir le tableau](#)

Lien	Description
Bien démarrer avec l'exemple de conversation d'entreprise .NET basé sur RAG	Article qui vous guide tout au long du déploiement et de l'utilisation de l'exemple d'application de conversation d'entreprise pour .NET ↗ . Cet exemple est une solution complète de bout en bout qui illustre le modèle RAG (Retrieval-Augmented Generation) exécuté dans Azure, utilisant Recherche Azure AI pour la récupération et les modèles de

Lien	Description
	grand langage Azure OpenAI afin d'alimenter les expériences de type ChatGPT et Q&A.

Blocs de construction

[\[+\] Agrandir le tableau](#)

Lien	Description
Recherche de similarité vectorielle avec Azure Cache pour Redis Entreprise (Python) ↗	Article qui vous guide tout au long de l'utilisation d'Azure Cache pour Redis en tant que magasin de vecteurs back-end pour les scénarios RAG. (Le code Python est fourni en tant que référence et peut être adapté à .NET.)
Solutions OpenAI avec vos propres données à l'aide de PostgreSQL (Python) ↗	Article qui explique comment le serveur flexible Azure Database pour PostgreSQL et Azure Cosmos DB pour PostgreSQL prennent en charge l'extension pgvector, ainsi qu'une vue d'ensemble, des scénarios, etc. (Le code Python est fourni en tant que référence et peut être adapté à .NET.)

Kits de développement logiciel (SDK) et autres exemples/conseils

[\[+\] Agrandir le tableau](#)

Lien	Description
Kit de développement logiciel (SDK) Azure OpenAI pour .NET ↗	La version source GitHub de la bibliothèque cliente Azure OpenAI pour .NET est une adaptation des API REST d'OpenAI qui fournit une interface idiomatique et une intégration enrichie avec le reste de l'écosystème SDK Azure. Elle peut se connecter aux ressources Azure OpenAI ou au point de terminaison d'inférence non-Azure OpenAI, ce qui en fait un excellent choix pour le développement non-Azure OpenAI.
Versions du SDK Azure OpenAI ↗	Liens vers tous les packages de bibliothèque du SDK Azure OpenAI, y compris des liens pour .NET, Java, JavaScript et Go.
Package NuGet Azure.AI.OpenAI ↗	Version NuGet de la bibliothèque cliente Azure OpenAI pour .NET.
Commencer à utiliser GPT-35-Turbo et GPT-4	Article qui vous guide tout au long de la création d'un exemple de saisie semi-automatique de conversation.

Lien	Description
Saisies semi-automatiques ↗	Collection de 10 exemples qui montrent comment utiliser la bibliothèque cliente Azure OpenAI pour .NET pour mener des conversations, diffuser des réponses en continu, utiliser vos propres données, transcrire/traduire de l'audio, générer des images, etc.
Diffusion en continu de saisies semi-automatiques de conversation ↗	Lien profond vers les exemples illustrant la diffusion en continu de saisies semi-automatiques.
OpenAI avec le contrôle d'accès en fonction du rôle Microsoft Entra ID	Aperçu de l'authentification à l'aide de Microsoft Entra ID.
OpenAI avec les identités managées	Article contenant des scénarios de sécurité plus complexes qui nécessitent un contrôle d'accès en fonction du rôle Azure (Azure RBAC). Ce document explique comment s'authentifier auprès de votre ressource OpenAI à l'aide de Microsoft Entra ID.
Plus d'exemples ↗	Collection d'exemples OpenAI écrits en .NET.
Autres conseils	Page hub de la documentation Azure OpenAI Service.

Autres Azure AI services

Solutions de bout en bout

[Agrandir le tableau](#)

Lien	Description
Sous-titrage et transcription dans un centre d'appels ↗	Référentiel contenant des exemples pour les sous-titres et les transcriptions dans un scénario de centre d'appels.
Utiliser Form Recognizer pour automatiser un processus basé sur papier à l'aide de l'atelier Nouvel inscription de patient avec Form Recognizer ↗ (code ↗)	Procédure pas à pas complète d'un scénario Azure AI Document Intelligence dans un format d'atelier.

Blocs de construction

[Agrandir le tableau](#)

Lien	Description
Utiliser Speech pour converser avec OpenAI	Article expliquant comment utiliser Azure AI Speech pour converser avec Azure OpenAI Service. Le texte reconnu par le service Speech est envoyé à Azure OpenAI. Le service Speech synthétise ensuite la réponse textuelle d'Azure OpenAI.
Traduire des documents depuis et dans plus de 100 langues différentes ↗	Article expliquant comment traduire des fichiers locaux ou des fichiers réseau dans de nombreux formats différents, dans plus de 100 langues différentes. Les formats pris en charge incluent HTML, PDF, tous les formats de document Office, Markdown, MHTML, Outlook, MSG, XLIFF, CSV, TSV et le texte brut.

SDK et autres exemples/conseils

 [Agrandir le tableau](#)

Lien	Description
Intégrer Speech à vos applications avec des exemples de SDK Speech ↗	Référentiel d'exemples pour le SDK Speech Azure Cognitive Services. Liens vers des exemples pour la reconnaissance vocale, la traduction, la synthèse vocale, etc.
Kit de développement logiciel (SDK) Azure AI Document Intelligence	Azure AI Document Intelligence (anciennement Form Recognizer) est un service cloud qui utilise le Machine Learning pour analyser du texte et des données structurées à partir de documents. Le Kit de développement logiciel (SDK) Intelligence documentaire est un ensemble de bibliothèques et d'outils qui vous permettent d'intégrer facilement les modèles et fonctionnalités d'Intelligence documentaire dans vos applications.
Extraire des données structurées à partir de formulaires, de reçus, de factures et de carte à l'aide de Form Recognizer dans .NET ↗	Référentiel d'exemples pour la bibliothèque cliente Azure.AI.FormRecognizer.
Extraire, classer et comprendre du texte dans des documents à l'aide d'Analyse de texte dans .NET ↗	Bibliothèque cliente pour Analyse de texte. Elle fait partie du service Azure AI Language , qui fournit des fonctionnalités de traitement du langage naturel (NLP) pour comprendre et analyser du texte.
Traduction de documents dans .NET ↗	Article de démarrage rapide qui explique comment utiliser la fonction Traduction de documents pour traduire un document source dans une langue cible tout en conservant la structure et la mise en forme du texte.

Lien	Description
Réponses aux questions dans .NET ↗	Article de démarrage rapide qui permet d'obtenir une réponse (et un indice de confiance) à partir d'un corps de texte que vous envoyez avec votre question.
Compréhension du langage courant dans .NET ↗	Bibliothèque cliente pour la compréhension du langage courant (CLU), un service d'IA conversationnelle basé sur le cloud, qui peut extraire des intentions et des entités dans des conversations et fait office d'orchestrateur pour sélectionner le meilleur candidat pour analyser les conversations afin d'obtenir la meilleure réponse à partir d'applications telles que Qna, Luis et Conversation App.
Analyser des images	Exemples de code et documents de configuration pour le Kit de développement logiciel (SDK) Microsoft Azure AI Vision

Connecter votre application aux services Azure

Article • 23/10/2023

Azure offre un large éventail de services que les applications peuvent tirer parti de qu'elles soient hébergées dans Azure ou localement. Par exemple, vous pouvez :

- Utilisez Stockage Blob Azure pour stocker et récupérer des fichiers dans le cloud.
- Ajoutez une fonctionnalité de recherche en texte intégral à votre application à l'aide de Recherche cognitive Azure.
- Utilisez Azure Service Bus pour gérer la messagerie entre différents composants d'une architecture de microservices.
- Utilisez Analyse de texte pour identifier et réactez les données sensibles dans un document.

Les services Azure offrent l'avantage qu'ils sont entièrement gérés par Azure.

Accès aux services Azure à partir du code d'application

Il existe deux façons d'accéder au service Azure à partir de votre code d'application.

- **Kit de développement logiciel (SDK) Azure** : disponible pour .NET, Java, JavaScript, Python et Go.
- **API REST Azure** - Disponible à partir de toutes les langues.

Si possible, il est recommandé d'utiliser le Kit de développement logiciel (SDK) Azure pour accéder aux services Azure à partir du code d'application. Les avantages de l'utilisation du Kit de développement logiciel (SDK) Azure sont les suivants :

- **L'accès aux services Azure ressemble à l'utilisation de n'importe quelle autre bibliothèque.** Vous importez le package du Kit de développement logiciel (SDK) approprié dans votre application, créez un objet client, puis appelez des méthodes sur l'objet client pour communiquer avec votre ressource Azure.
- **Simplifie le processus d'authentification de votre application auprès d'Azure.** Lors de la création d'un objet client sdk, vous incluez les informations d'identification appropriées et le Kit de développement logiciel (SDK) s'occupe de l'authentification de vos appels à Azure
- **Modèle de programmation simplifié.** En interne, le Kit de développement logiciel (SDK) Azure appelle l'API REST Azure. Toutefois, le Kit de développement logiciel

(SDK) Azure a intégré la gestion des erreurs, la logique de nouvelle tentative et la pagination des résultats, ce qui simplifie la programmation par rapport au Kit de développement logiciel (SDK) plus simple que d'appeler directement l'API REST.

Kit de développement logiciel (SDK) Azure

Le Kit de développement logiciel (SDK) Azure permet l'accès par programmation aux services Azure à partir des applications .NET, Java, JavaScript, Python et Go. Les applications installent les packages nécessaires à partir de leur gestionnaire de package respectif, puis appellent des méthodes pour accéder par programmation aux ressources Azure.

<https://www.microsoft.com/fr-fr/videoplayer/embed/RE50C7t?postJsMsg=true&autoCaptions=fr-fr>

Vous trouverez plus d'informations sur le Kit de développement logiciel (SDK) Azure pour chaque langue dans le centre de développement de chaque langage.

Langue	Vue d'ensemble	Liste des packages
	.NET Vue d'ensemble du Kit de développement logiciel (SDK) Azure pour .NET	Liste des packages Azure SDK pour .NET
	Java Vue d'ensemble du Kit de développement logiciel (SDK) Azure pour Java	Liste des packages Azure SDK pour Java
	JavaScript Vue d'ensemble du Kit de développement logiciel (SDK) Azure pour JavaScript	Liste des packages Du Kit de développement logiciel (SDK) Azure pour JavaScript
	Python Vue d'ensemble du Kit de développement logiciel (SDK) Azure pour Python	Liste des packages Azure SDK pour Python
	Go Vue d'ensemble du Kit de développement logiciel (SDK) Azure pour Go	Liste des packages du Kit de développement logiciel (SDK) Azure pour Go

API REST Azure

Les langages de programmation non pris en charge par le Kit de développement logiciel (SDK) Azure peuvent utiliser l'API REST Azure. Des détails sur l'appel de l'API REST Azure

et la liste complète des opérations sont disponibles dans la vue d'ensemble [de l'API REST Azure](#).

[Vue d'ensemble de l'API REST Azure](#)

Comment faire créer et gérer des ressources dans Azure ?

Article • 23/10/2023

Azure fournit un large éventail d'outils pour créer et gérer les ressources Azure utilisées par votre application.

<https://www.microsoft.com/fr-fr/videoplayer/embed/RE50C5I?postJsIMsg=true&autoCaptions=fr-fr>

Différents outils sont conçus pour prendre en charge différents cas d'usage, et la plupart des développeurs Azure utilisent une combinaison de différents outils en fonction du travail qu'ils doivent effectuer. Par exemple, vous pouvez :

- Utilisez un outil gui comme le Portail Azure ou l'extension Azure Tools pour VS Code lors du prototypage de ressources Azure pour une nouvelle application. Les outils gui vous guident tout au long du processus de création de nouveaux services et vous permettent de passer en revue et de sélectionner les options d'un service à l'aide de menus déroulants et d'autres éléments graphiques.
- Écrivez un script à l'aide d'Azure CLI ou d'Azure PowerShell pour automatiser une tâche courante. Par exemple, vous pouvez créer un script qui crée un environnement de développement de base pour une nouvelle application web composée d'azure App Service, d'une base de données et d'un stockage d'objets blob. L'écriture d'un script garantit que les ressources sont créées de la même façon à chaque fois et qu'elles sont plus rapides à exécuter que de cliquer sur une interface utilisateur.
- Utilisez les outils IaC (Infrastructure as Code) pour déployer et gérer de manière déclarative des ressources Azure. Les outils tels que Terraform, Ansible ou Bicep vous permettent de codifier les ressources Azure nécessaires pour une solution dans la syntaxe déclarative, en garantissant le déploiement cohérent des ressources Azure entre les environnements et en empêchant la dérive environnementale.

Portail Azure

Le [Portail Azure](#) est une interface web conçue pour la gestion des ressources Azure. Fonctionnalités Portail Azure :

- Interface utilisateur facile à utiliser pour créer et gérer des ressources Azure

- Possibilité de créer des tableaux de bord configurables
- Accès aux paramètres d'abonnement et aux informations de facturation

The screenshot shows the Microsoft Azure portal interface. On the left is a dark sidebar with various navigation links such as 'Create a resource', 'Home', 'Dashboard', 'All services', 'FAVORITES', 'All resources', 'Resource groups', 'App Services', 'SQL databases', 'Azure Cosmos DB', 'Virtual machines', 'Load balancers', 'Storage accounts', 'Virtual networks', 'Azure Active Directory', 'Monitor', 'Advisor', 'Microsoft Defender for Cloud', 'Help + support', and 'Cost Management + Billing'. The main content area has a title 'Azure services' with icons for 'Create a resource', 'App Services', 'Cost Management ...', 'Subscriptions', 'App registrations', 'Enterprise applications', 'Virtual machines', 'Azure Active Directory', 'Azure Cosmos DB', and 'All services'. Below this is a section titled 'Recent resources' with a table:

Name	Type	Last Viewed
Visual Studio Enterprise Subscription	Subscription	3 days ago
Pay-As-You-Go	Subscription	3 days ago
vm-azure-developer	Virtual machine	a week ago
msdocs-expressjs-mongodb-123	App Service	a week ago
msdocs-expressjs-mongodb-tutorial	Resource group	4 weeks ago
vm-dotnet-windows10	Virtual machine	2 months ago
rg-developer	Resource group	2 months ago
vm-dotnet-windows10-ip	Public IP address	2 months ago
stblobstoragedemo123	Storage account	3 months ago
sample-demo-db	Azure Cosmos DB API for MongoDB account	4 months ago
contoso-pizza-dcb	Service Bus Namespace	5 months ago
storageblobdemodcb	Storage account	5 months ago

At the bottom of the recent resources section is a 'See all' link and a magnifying glass icon. Below the table is a 'Navigate' button.

Pack d'extension Vs Code Azure Tools

Les développeurs utilisant [Visual Studio Code](#) peuvent gérer des ressources Azure directement à partir de VS Code à l'aide du [pack](#) d'extension Azure Tools pour VS Code. L'utilisation du pack d'extension Azure Tools peut :

- Créez, gérez et déployez du code sur des sites web à l'aide d'Azure App Service.
- Créer, parcourir et interroger des bases de données Azure
- Créer, déboguer et déployer Azure Functions directement à partir de VS Code
- Déployer des applications conteneurisées à partir de VS Code

[Télécharger le pack d'extension Azure Tools](#)

The screenshot shows the sidebar of the Azure DevOps Services interface. On the left, there's a vertical list of icons: a clipboard, a magnifying glass, a gear with a '1' notification, a play button, a grid, and a refresh/circular arrow. To the right of these icons, the sidebar is organized into sections:

- AZURE**: A dropdown menu with options like 'New Project', 'My Work', 'My Boards', 'My Pipelines', and 'My Repos'. It also includes 'Azure Resources' (Remote) which is expanded to show 'VS Code Demos' (also expanded) containing 'App Services', 'Azure Cosmos DB', 'Function App', 'PostgreSQL servers (Flexible)', 'PostgreSQL servers (Standard)', 'Static Web Apps', 'Storage accounts', and 'Virtual machines'. There are also 'WORKSPACE' (Local) and 'HELP AND FEEDBACK' sections.
- RESOURCES**: Shows 'Remote' resources: 'VS Code Demos' (expanded), 'App Services', 'Azure Cosmos DB', 'Function App', 'PostgreSQL servers (Flexible)', 'PostgreSQL servers (Standard)', 'Static Web Apps', 'Storage accounts', and 'Virtual machines'.
- WORKSPACE**: Shows 'Local' workspace resources: 'Attached Database Accounts' and 'Attached Storage Accounts'.
- HELP AND FEEDBACK**: Includes links for 'Get Started', 'Review Issues', and 'Report Issue'.

Outils en ligne de commande

Les outils en ligne de commande offrent les avantages de l'efficacité, de la répétabilité et de la possibilité de générer des scripts de tâches périodiques. Azure fournit deux outils en ligne de commande différents parmi lesquels choisir. Azure CLI et Azure

PowerShell sont fonctionnellement équivalents. Vous devez uniquement sélectionner et utiliser l'outil qui correspond le mieux à votre flux de travail individuel.

Azure CLI

Azure [CLI](#) est un outil en ligne de commande multiplateforme qui s'exécute sur Windows, Linux et macOS. Interface de ligne de commande Azure :

- Propose une syntaxe concise et efficace pour la gestion des ressources Azure.
- Génère des résultats au format JSON (par défaut). Les résultats peuvent également être mis en forme en tant que YAML, une table ASCII ou des valeurs séparées par des onglets sans clé.
- Permet d'interroger et de mettre en forme la sortie via l'utilisation de [requêtes ↗](#) JMESPath.

Les commandes Azure CLI sont facilement incorporées dans des langages de script populaires comme [Bash](#), ce qui vous permet de générer des scripts de tâches courantes.

```
Azure CLI

LOCATION='eastus'
RESOURCE_GROUP_NAME='msdocs-expressjs-mongodb-tutorial'

WEB_APP_NAME='msdocs-expressjs-mongodb-123'
APP_SERVICE_PLAN_NAME='msdocs-expressjs-mongodb-plan-123'
RUNTIME='NODE|14-lts'

# Create a resource group
az group create \
    --location $LOCATION \
    --name $RESOURCE_GROUP_NAME

# Create an app service plan
az appservice plan create \
    --name $APP_SERVICE_PLAN_NAME \
    --resource-group $RESOURCE_GROUP_NAME \
    --sku B1 \
    --is-linux

# Create the web app in the app service
az webapp create \
    --name $WEB_APP_NAME \
    --runtime $RUNTIME \
    --plan $APP_SERVICE_PLAN_NAME \
    --resource-group $RESOURCE_GROUP_NAME
```

Azure PowerShell

Azure PowerShell est un ensemble d'apports de commande permettant de gérer les ressources Azure directement à partir de PowerShell. Azure PowerShell est installé en tant que module PowerShell et fonctionne avec PowerShell 7.0.6 LTS et PowerShell 7.1.3 ou version ultérieure sur toutes les plateformes, notamment Windows, macOS et Linux. Il est également compatible avec Windows PowerShell 5.1.

Azure PowerShell est étroitement intégré au langage PowerShell. Les commandes suivent un format verbe-nom et les données sont retournées en tant qu'objets PowerShell. Si vous connaissez déjà les scripts PowerShell, Azure PowerShell est un choix naturel.

```
Azure PowerShell

$location = 'eastus'
$resourceGroupName = 'msdocs-blob-storage-demo-azps'
$storageAccountName = 'stblobstoragedemo999'

# Create a resource group
New-AzResourceGroup
    -Location $location
    -Name $resourceGroupName

# Create the storage account
New-AzStorageAccount
    -Name $storageAccountName
    -ResourceGroupName $resourceGroupName
    -Location $location
    -SkuName Standard_LRS
```

Pour plus d'informations sur le choix entre Azure CLI et Azure PowerShell, consultez l'article [Choisir l'outil](#) en ligne de commande approprié.

Outils d'infrastructure en tant que code

L'[infrastructure en tant que code](#) est le processus de gestion et d'approvisionnement des ressources via des fichiers de configuration déclaratifs. L'infrastructure en tant qu'outils de code utilise une spécification d'état final déclaratif pour garantir qu'un ensemble de ressources est créé et configuré de la même façon à chaque fois. En outre, la plupart des outils de code surveillent les ressources pour s'assurer qu'elles restent configurées dans l'état souhaité.

Pour les déploiements d'infrastructure automatisés, répétés et fiables, support Azure une variété d'outils d'infrastructure en tant que code.

Bicep

Bicep est un langage spécifique à un domaine (DSL) qui utilise la syntaxe déclarative pour déployer des ressources Azure. Il fournit une syntaxe concise, une cohérence des types fiable et une prise en charge de la réutilisation du code.

Bicep

```
param location string = resourceGroup().location
param storageAccountName string =
'toYLaunch${uniqueString(resourceGroup().id)}'

resource storageAccount 'Microsoft.Storage/storageAccounts@2021-06-01' = {
  name: storageAccountName
  location: location
  sku: {
    name: 'Standard_LRS'
  }
  kind: 'StorageV2'
  properties: {
    accessTier: 'Hot'
  }
}
```

Terraform

Hashicorp Terraform est un outil open source pour l'approvisionnement et la gestion d'infrastructure cloud. Il codifie l'infrastructure dans des fichiers de configuration qui décrivent la topologie des ressources cloud. L'interface CLI Terraform fournit un mécanisme simple pour déployer et versionner des fichiers de configuration sur Azure.

Terraform

```
provider "azurerm" {
  features {}
}

resource "azurerm_resource_group" "main" {
  name      = "${var.prefix}-resources"
  location = var.location
}

resource "azurerm_app_service_plan" "main" {
  name          = "${var.prefix}-asp"
  location      = azurerm_resource_group.main.location
  resource_group_name = azurerm_resource_group.main.name
  kind          = "Linux"
  reserved     = true
}
```

```

sku {
  tier = "Standard"
  size = "S1"
}

resource "azurerm_app_service" "main" {
  name          = "${var.prefix}-appservice"
  location      = azurerm_resource_group.main.location
  resource_group_name = azurerm_resource_group.main.name
  app_service_plan_id = azurerm_app_service_plan.main.id

  site_config {
    linux_fx_version = "NODE|10.14"
  }
}

```

Ansible

Ansible est un produit open source qui automatise l'approvisionnement du cloud, la gestion de la configuration et le déploiement des applications. Il vous permet d'approvisionner les machines virtuelles, les conteneurs et le réseau, ainsi que des infrastructures cloud complètes. De plus, Ansible vous permet d'automatiser le déploiement et la configuration de ressources dans votre environnement.

yml

```

- hosts: localhost
  connection: local
  vars:
    resource_group: myResourceGroup
    webapp_name: myfirstWebApp
    plan_name: myAppServicePlan
    location: eastus
  tasks:
    - name: Create a resource group
      azure_rm_resourcegroup:
        name: "{{ resource_group }}"
        location: "{{ location }}"

    - name: Create App Service on Linux with Java Runtime
      azure_rm_webapp:
        resource_group: "{{ resource_group }}"
        name: "{{ webapp_name }}"
        plan:
          resource_group: "{{ resource_group }}"
          name: "{{ plan_name }}"
          is_linux: true
          sku: S1
          number_of_workers: 1

```

```
frameworks:  
  - name: "java"  
    version: "8"  
    settings:  
      java_container: tomcat  
      java_container_version: 8.5
```

API REST et SDK Azure

Les ressources Azure peuvent également être créées par programmation à partir du code. Cela vous permet d'écrire des applications qui provisionnent dynamiquement des ressources Azure en réponse aux demandes des utilisateurs. Le Kit de développement logiciel (SDK) Azure fournit des packages de gestion des ressources dans .NET, Go, Java, JavaScript et Python qui permettent aux ressources Azure d'être créées et gérées directement dans le code. Vous pouvez également gérer les ressources Azure via des requêtes HTTP vers un point de terminaison RESTful.

[Utilisation du Kit de développement logiciel \(SDK\) Azure pour .NET](#)

[Utilisation du Kit de développement logiciel \(SDK\) Azure pour Go](#)

[Utilisation du Kit de développement logiciel \(SDK\) Azure pour Java](#)

[Utilisation du Kit de développement logiciel \(SDK\) Azure pour JavaScript](#)

[Utilisation du Kit de développement logiciel \(SDK\) Azure pour Python](#)

[Utilisation des API REST Azure](#)

Concepts clés pour la création d'applications Azure

Article • 23/10/2023

Avant de vous rendre trop loin dans la conception de votre application pour s'exécuter sur Azure, il est probable que vous devrez effectuer une petite planification à l'avance. À mesure que vous commencez, il existe quelques concepts Azure de base que vous devez comprendre pour prendre les meilleures décisions pour votre scénario. Éléments à prendre en compte :

Régions Azure

Une région est constituée d'un ensemble de centres de données déployés dans un périmètre avec une latence définie et connectés via un réseau régional dédié à faible latence. Azure vous offre la possibilité de déployer des applications dans lesquelles vous devez, y compris entre plusieurs régions, fournir une résilience inter-régions si nécessaire.

En règle générale, vous souhaitez que toutes les ressources d'une solution se trouver dans la même région réduisent la latence entre les différents composants de votre application. Cela signifie que si votre solution se compose d'Azure App Service, d'une base de données et d'un stockage Blob Azure, toutes ces ressources doivent être créées dans la même région Azure.

Tous les services Azure ne sont pas disponibles dans chaque région. La [page Produits disponibles par région](#) peut vous aider à trouver une région dans laquelle les services Azure nécessaires par votre application sont disponibles.

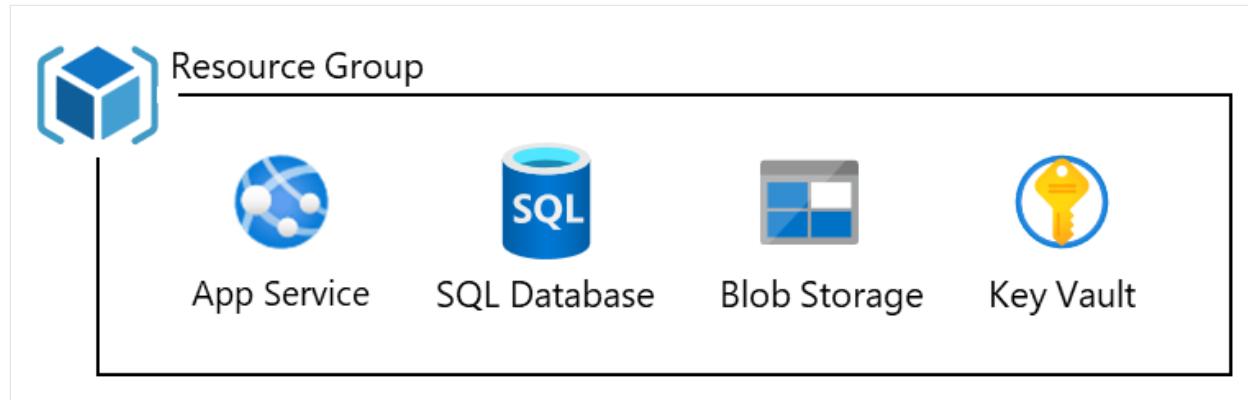
<https://www.microsoft.com/fr-fr/videoplayer/embed/RE50C5F?postJsIMsg=true&autoCaptions=fr-fr>

Groupe de ressources Azure

Un groupe de ressources dans Azure est un conteneur logique pour regrouper les ressources Azure. Chaque ressource Azure doit appartenir à un seul et un seul groupe de ressources.

Les groupes de ressources sont souvent utilisés pour regrouper toutes les ressources Azure nécessaires à une solution dans Azure. Par exemple, supposons que vous avez déployé une application web sur Azure App Service qui utilise une base de données

SQL, Stockage Azure et Azure Key Vault. Il est courant de placer toutes les ressources Azure nécessaires pour cette solution dans un seul groupe de ressources.

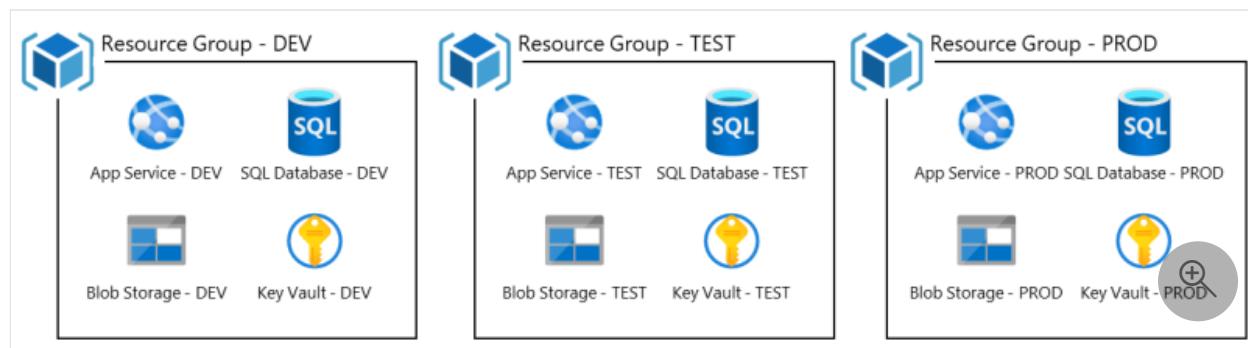


Cela facilite l'exécution des ressources nécessaires à l'exécution de l'application et à quelles ressources sont liées les unes aux autres. Par conséquent, la première étape de création de ressources pour une application dans Azure consiste généralement à créer le groupe de ressources qui servira de conteneur pour les ressources de l'application.

[https://www.microsoft.com/fr-fr/videoplayer/embed/RE50C5E?
postJslMsg=true&autoCaptions=fr-fr ↴](https://www.microsoft.com/fr-fr/videoplayer/embed/RE50C5E?postJslMsg=true&autoCaptions=fr-fr)

Environnements

Si vous avez développé localement, vous êtes familiarisé avec la promotion de votre code via des environnements de développement, de test et de production. Dans Azure, pour créer des environnements distincts, vous devez créer un ensemble distinct de ressources Azure pour chaque environnement dont vous avez besoin.



Étant donné qu'il est important que chaque environnement soit une copie exacte, il est recommandé [de générer un script de création de ressources](#) nécessaires à un environnement ou d'utiliser [des outils IaC](#) (Infrastructure as Code) pour spécifier de manière déclarative la configuration de chaque environnement. Cela permet de s'assurer que le processus de création d'environnement est reproductible et vous permet également de faire tourner de nouveaux environnements à la demande, par exemple pour les tests de performances ou de sécurité de votre application.

<https://www.microsoft.com/fr-fr/videoplayer/embed/RE50C5M?postJs||Msg=true&autoCaptions=fr-fr>

DevOps Support

Qu'il s'agisse de publier vos applications sur Azure avec l'intégration continue ou l'approvisionnement de ressources pour un nouvel environnement, Azure s'intègre à la plupart des outils DevOps populaires. Vous pouvez travailler avec les outils dont vous disposez déjà et optimiser votre expérience avec la prise en charge d'outils, notamment :

- [Actions GitHub](#)
- [Azure DevOps](#)
- [Déploiement de Pieuvre](#)
- [Jenkins](#)
- [Terraform](#)
- [Ansible](#)
- [Chef](#)

Comment suis-je facturé ?

Article • 23/10/2023

Lorsque vous créez des applications qui utilisent Azure, vous devez comprendre les facteurs qui influencent le coût des solutions que vous créez. Vous devez également comprendre comment vous pouvez estimer le coût d'une solution, la façon dont vous êtes facturé et comment vous pouvez surveiller les coûts engagés dans vos abonnements Azure.

Qu'est-ce qu'un compte Azure ?

Votre compte Azure vous permet de vous connecter à Azure. Vous disposez peut-être d'un compte Azure par le biais de l'organisation pour laquelle vous travaillez ou de l'école que vous participez. Vous pouvez également créer un compte Azure individuel pour une utilisation personnelle liée à votre compte Microsoft. Si vous souhaitez en savoir plus sur Azure et l'expérimenter, vous pouvez [créer gratuitement un compte ↗](#) Azure.

[Créez un compte Azure gratuit](#)

Si vous utilisez un compte Azure à partir de votre espace de travail ou de votre établissement scolaire, les administrateurs Azure de votre organisation ont probablement affecté différents groupes et rôles à votre compte qui régissent ce que vous pouvez et ne pouvez pas faire dans Azure. Si vous ne pouvez pas créer un certain type de ressource, case activée avec votre administrateur Azure sur les autorisations attribuées à votre compte.

Qu'est-ce qu'un abonnement Azure ?

La facturation des ressources Azure est effectuée par abonnement. Un abonnement Azure définit donc un ensemble de ressources Azure qui seront facturées ensemble.

Les organisations créent souvent plusieurs abonnements Azure à des fins de facturation et de gestion. Par exemple, une organisation peut choisir de créer un abonnement pour chaque service de l'organisation afin que chaque service paie ses propres ressources Azure. *Lors de la création de ressources Azure, il est important de prêter attention à l'abonnement dans lequel vous créez les ressources, car le propriétaire de cet abonnement paiera ces ressources.*

Si vous disposez d'un compte Azure individuel lié à votre compte Microsoft, il est également possible d'avoir plusieurs abonnements. Par exemple, un utilisateur peut avoir à la fois un abonnement Visual Studio Enterprise qui fournit des crédits Azure mensuels et un abonnement de paiement à l'utilisation qui facture à son crédit carte. Dans ce scénario, vous souhaitez à nouveau être sûr et choisir l'abonnement approprié lors de la création de ressources Azure afin d'éviter une facture inattendue pour les services Azure.

<https://www.microsoft.com/fr-fr/videoplayer/embed/RE50ydl?postJslMsg=true&autoCaptions=fr-fr>

Quels facteurs influencent le coût d'un service sur Azure ?

Plusieurs facteurs peuvent influencer le coût d'un service donné dans Azure.

- **Puissance de calcul** : la puissance de calcul fait référence à la quantité d'UC et de mémoire affectée à une ressource. Plus la puissance de calcul allouée à une ressource est élevée, plus le coût sera élevé. De nombreux services Azure incluent la possibilité d'effectuer une mise à l'échelle élastique, ce qui vous permet d'accélérer la puissance de calcul lorsque la demande est élevée, mais que vous économisez de l'argent lorsque la demande est faible.
- **Stockage montant** : la plupart des services de stockage sont facturés en fonction de la quantité de données que vous souhaitez stocker.
- **Stockage matériel** : certains services de stockage fournissent des options sur le type de matériel sur lequel vos données seront stockées. Selon le type de données que vous stockez, vous pouvez souhaiter une option de stockage à long terme plus lente avec des vitesses de lecture et d'écriture plus lentes, ou vous pouvez être prêt à payer pour une faible latence de lecture et d'écritures pour des opérations transactionnelles hautement transactionnelles.
- **Bandé passante** : la plupart des services facturent séparément l'entrée et la sortie. L'entrée est la quantité de bande passante requise pour gérer les requêtes entrantes. La sortie est la quantité de bande passante requise pour gérer les données sortantes qui répondent à ces demandes.
- **Par utilisation** : certaines factures de services basées sur le nombre de fois où le service est utilisé ou le nombre de requêtes gérées ou le nombre d'entités (telles que les comptes d'utilisateur Microsoft Entra) qui ont été configurées.
- **Par service** - Certains services facturent simplement des frais mensuels consécutifs.
- **Région** : parfois, les services ont des prix différents en fonction de la région (centre de données) où il est hébergé.

Calcul des coûts Azure

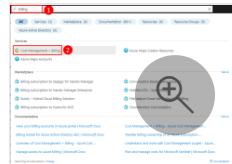
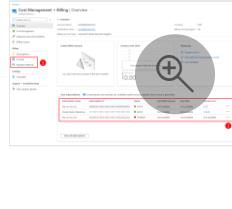
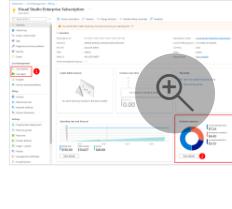
La plupart des solutions Azure impliquent plusieurs services Azure, ce qui rend difficile de déterminer le coût d'une solution en amont. Pour cette raison, Azure fournit la [calculatrice](#) de prix Azure pour vous aider à estimer le coût d'une solution.

Calcul des coûts Azure

Où puis-je trouver nos dépenses actuelles dans Azure ?

Le Portail Azure fournit une présentation visuelle et facile à parcourir de tous les services utilisés par votre organisation pendant un mois particulier. Vous pouvez afficher par service, par groupe de ressources, et ainsi de suite.

Pour accéder aux informations de facturation dans le Portail Azure, [connectez-vous au Portail Azure](#) et suivez ces étapes.

Instructions	Capture d'écran
Pour afficher les informations de facturation de votre compte Azure :	
<ol style="list-style-type: none">1. Dans la zone de recherche située en haut de la page, tapez <i>Billing*</i>.2. Sélectionnez l'élément <i>Cost Management + Billing</i> dans la boîte de dialogue.	
Vous accédez à la page Vue d'ensemble de Cost Management + Facturation. Sur cette page, vous pouvez :	
<ol style="list-style-type: none">1. Utilisez le menu de gauche pour passer en revue <i>les factures et les modes</i> de paiement de vos abonnements.2. Affichez la liste de vos abonnements et leurs frais actuels. La sélection d'un abonnement dans la table vous permet d'obtenir des informations détaillées sur les coûts relatifs à cet abonnement.	
La page de détails de chaque abonnement vous permet de :	
<ol style="list-style-type: none">1. Effectuez l'analyse <i>des coûts et configurez les alertes</i> de coût sur l'abonnement.2. Afficher les coûts détaillés par ressource dans l'abonnement.	

Vous pouvez également accéder directement à la page vue d'ensemble de Cost Management + Facturation.

Azure Cost Management dans le portail Azure

Les informations sur les coûts sont également accessibles par programme pour créer une vue personnalisée et facilement accessible dans votre dépense cloud pour la gestion via l'API facturation.

- Bibliothèques de facturation Azure pour .NET
- Bibliothèques de facturation Azure pour Python
- Bibliothèque de client de facturation Azure Resource Manager pour Java - Version 1.0.0-beta.1
- Tous les autres langages de programmation - API RESTful
- Présentation des API Azure Consumption

Quels outils sont disponibles pour surveiller et analyser mes dépenses cloud ?

Deux services sont disponibles pour configurer et gérer vos coûts cloud.

- La première est **les alertes** de coût qui vous permettent de définir des seuils de dépense et de recevoir des notifications en tant que votre facture proche de ces seuils.
- La deuxième est **Azure Cost Management** qui vous aide à planifier et à contrôler vos coûts, à fournir une analyse des coûts, des budgets, des recommandations et vous permet d'exporter des données de gestion des coûts pour l'analyse dans Excel ou votre propre rapport personnalisé.

En savoir plus sur les alertes de coût et **Azure Cost Management** :

- Utiliser les alertes de coût pour superviser l'utilisation et les dépenses
- Présentation d'Azure Cost Management + Facturation
- Guide pratique pour optimiser votre investissement dans le cloud avec Azure Cost Management

Azure pour les développeurs .NET

Apprenez à utiliser le SDK Azure pour .NET. Parcourez des informations de référence sur les API, des exemples de code, des tutoriels, des guides de démarrage rapide, des articles conceptuels, et plus encore. Sachez que .NET ❤️ Azure.



VUE D'ENSEMBLE [Introduction à Azure et .NET](#)



DÉMARRAGE RAPIDE [Créer une application web ASP.NET Core dans Azure](#)



DÉMARRAGE RAPIDE [Créer une fonction serverless](#)



DIDACTICIEL [ASP.NET Core et Docker](#)



DÉPLOYER [Déployer une application .NET avec Azure DevOps](#)



DIDACTICIEL [Authentification de bout en bout dans App Service](#)



ENTRAÎNEMENT [Sécuriser les API personnalisées avec l'identité Microsoft](#)



BIEN DÉMARRER [Kit SDK Azure pour .NET](#)

Contenu proposé

Découvrez comment développer des applications .NET en tirant parti de divers services Azure.

Créer des applications web

- [Vue d'ensemble d'App Service](#)
- [Vue d'ensemble des fonctions Azure](#)

Créer des applications natives cloud

- [Exécuter et déboguer un microservice dans Kubernetes](#)

Créer des applications mobiles

- [Utiliser des services REST à partir d'Azure dans les applications Xamarin](#)

- | | | |
|--|---|--|
|  Héberger une application web avec Azure App Service |  Créer et déployer un microservice ASP.NET Core natif cloud |  Créer une application Xamarin.Forms avec le SDK .NET et l'API Azure Cosmos DB pour MongoDB |
|  Développer, tester et déployer une fonction Azure avec Visual Studio |  Déployer et déboguer plusieurs conteneurs dans AKS |  Bibliothèque cliente Stockage Blob Azure avec Xamarin.Forms |
|  Publier et gérer vos API avec Gestion des API Azure |  Configuration dynamique et indicateurs de fonctionnalités avec Azure App Config |  Envoyer des notifications Push à des applications Xamarin.Forms avec ASP.NET Core et Azure Notification Hubs |
|  Identité managée avec ASP.NET et Azure SQL Database |  Déployer une application .NET Core sur Azure Container Registry |  Ajouter une authentification et gérer les identités des utilisateurs dans vos applications mobiles |
|  API web avec CORS dans Azure App Service | | |

Travailler avec les données et le stockage

-  Choisir l'option de stockage de données appropriée
-  Utiliser .NET pour interroger une base de données Azure SQL ou Azure SQL Managed Instance
-  Utiliser .NET pour interroger Azure PostgreSQL
-  Utiliser le modèle de référentiel avec Azure Cosmos DB ↗
-  Se connecter à et interroger une base de données Azure Database pour PostgreSQL
-  Conserver et récupérer des données relationnelles avec Entity Framework Core
-  Générer une application .NET Core avec Azure Cosmos DB dans Visual Studio Code

Authentification et sécurité

-  Présentation de la plateforme d'identités Microsoft (Azure AD)
-  Sécuriser votre application en utilisant OpenID Connect et Azure AD
-  Sécuriser les API personnalisées avec l'identité Microsoft
-  Sécuriser une application web ASP.NET Core avec le framework ASP.NET Identity
-  Ajouter la connexion à Microsoft à une application web ASP.NET
-  Authentification de bout en bout dans App Service
-  Utiliser Azure Key Vault avec ASP.NET Core
-  Intégrer Azure AD B2C avec une API web

Messagerie sur Azure

-  Stockage des messages avec des files d'attente Azure
-  Messagerie entre applications avec Azure Service Bus
-  Streaming de Big Data avec Event Hubs
-  Création d'applications basées sur des événements avec Event Grid
-  Utiliser Stockage File d'attente Azure
-  Utiliser des files d'attente Azure Service Bus
-  Ingérer des données en temps réel par le biais d'Azure Event Hubs

 Stocker des données d'application avec

Diagnostics et surveillance

-  Démarrage rapide d'Azure Monitor Application Insights
-  Capturer et afficher les temps de chargement des pages dans votre application web Azure
-  Résoudre les problèmes liés à ASP.NET Core sur Azure App Service et IIS
-  Capturer les données de télémétrie Application Insights avec .NET Core ILogger
-  Application Insights pour les applications Service Worker
-  Dépanner une application dans Azure App Service à l'aide de Visual Studio

 Bibliothèque cliente Azure Identity pour .NET

Migration

-  Choisir une option d'hébergement Azure
-  Migrer une application ou service web .NET vers Azure App Service
-  Migrer une application ASP.NET vers une machine virtuelle Azure
-  Déployer une base de données SQL Server vers Azure

 Acheminer des événements personnalisés

Kit SDK Azure pour .NET

-  Paquets
-  Authentification pour les applications
-  Journalisation
-  Exemple d'application de SDK
-  Liste de contrôle des outils
-  Informations de référence sur l'API

Ressources de la communauté .NET et Azure

.NET

- [Documentation .NET](#)
- [Documentation d'ASP.NET](#)

Webcasts et émissions

- [Azure Friday ↗](#)
- [Présentation Cloud Native Sur .NET](#)
- [Présentations de la communauté .NET ↗](#)
- [Sur .NET Live ↗](#)

Open source

- [Kit SDK Azure pour .NET ↗](#)
- [Microsoft Identity Web ↗](#)
- [Plateforme .NET ↗](#)

Aimeriez-vous contribuer à la documentation de .NET ? Pour plus d'informations, consultez notre [guide du contributeur](#).

Documentation Azure pour les développeurs Java

Commencez à développer des applications pour le cloud avec ces tutoriels et outils pour les développeurs Java.

Bien démarrer avec Java sur Azure

-  Code, déploiement et mise à l'échelle de Java de votre façon
-  Code à l'aide des outils Java que vous connaissez et adorez
-  Déployer avec confiance et facilité
-  Mettre à l'échelle avec la sécurité, la surveillance et l'automatisation de bout en bout

[En savoir plus >](#)

Ressources de formation Java

-  Vue d'ensemble
-  Exemples Java sur Azure
-  Obtenir de l'aide sur Java auprès de Microsoft

[En savoir plus >](#)

Démarrages rapides Azure pour Java

-  Déployer une application web Java SE sur Linux
-  Créer une fonction serverless
-  Déployer des microservices Spring Cloud

[En savoir plus >](#)

Outils, IDE et JDK pris en charge

-  Prise en charge de Java
-  Installation de Java JDK
-  Images de Java Docker pour Azure

[En savoir plus >](#)

Migrer vers Azure

-  Spring vers Azure Spring Apps
-  Tomcat vers Azure App Service
-  WebLogic vers Machines virtuelles Azure

[En savoir plus >](#)

Kit de développement logiciel (SDK) Azure pour Java

-  Bibliothèques, pilotes et modules Spring
-  Développement Azure avec Java
-  Présentation du SDK Azure pour Java

[En savoir plus >](#)

Azure App Service

Azure Spring Apps

-  [Créer une application Java](#)
 -  [Configurer Java](#)
 -  [Déployer une application Spring avec MySQL](#)
- [Voir la documentation App Service >](#)

-  [Qu'est-ce qu'Azure Spring Apps ?](#)
 -  [Lancer votre première application](#)
 -  [Niveau Enterprise](#)
- [En savoir plus >](#)

Spring sur l'intégration d'Azure

-  [Qu'est-ce que Spring Cloud Azure ?](#)
-  [Spring Data pour Azure Cosmos DB](#)
-  [Déployer une application Spring Boot](#)

[En savoir plus >](#)

Mise en conteneur

-  [Vue d'ensemble](#)
-  [Établir une ligne de base](#)
-  [Conteneuriser pour Kubernetes](#)

[En savoir plus >](#)

Azure Functions

-  [Créer une fonction Azure](#)
-  [Créer une fonction Spring Cloud](#)
-  [Guide du développeur](#)

[Voir la documentation Azure Functions >](#)

Supervision des applications Java

-  [Prise en main d'Application Insights](#)
-  [Prise en main d'ELK](#)
-  [Superviser les applications Spring](#)

[Voir la documentation Azure Monitor >](#)

Sécurisation des applications Java

-  [Activer l'authentification des utilisateurs finaux](#)
-  [Microsoft Authentication Library](#)
-  [Gérer les secrets d'application](#)

[Voir la documentation Active Directory >](#)

Java EE, Jakarta EE et MicroProfile

-  [Oracle WebLogic Server sur des machines virtuelles Azure](#)
-  [Déployer une application Java EE sur AKS](#)
-  [Déployer une application MicroProfile sur Azure App Service](#)

[Consulter la documentation de Java EE >](#)

outils



[Azure Toolkit for IntelliJ](#)



[Visual Studio Code](#)



Azure Toolkit for Eclipse

Maven

[Maven ↗](#)



Eclipse MicroProfile



[Gradle ↗](#)



Azure CLI



Jenkins sur Azure

Java et OpenJDK sont des marques ou des marques déposées d'Oracle et/ou de ses affiliés.

Azure pour les développeurs JavaScript et Node.js

Explorez la puissance de JavaScript sur Azure avec des démarrages rapides, des guides pratiques, des exemples de code et plus encore.

You are a new Azure user ?

BIEN DÉMARRER

[What is Azure ?](#)

[Basic principles of Azure](#)

[Install Node.js](#)

[Configure the development environment](#)

[Authenticate with the Azure SDK](#)

[Authenticate users of your web application](#)

[Storage for developers](#)

[Blog on the Azure SDK kit ↗](#)

Deploy and host applications

DIDACTICIEL

[Select a hosting service](#)

[Client : JamStack loads the image in Storage](#)

[Client : JamStack + authentication](#)

[Migrate to serverless](#)

[Serverless : Main entry](#)

[Server : Deploy Express.js](#)

[Server from a virtual machine : Express.js with NGINX](#)

Tutorials on client library SDK

DIDACTICIEL

[Application web statique : Analyser l'image avec Vision par ordinateur](#)

[Application web statique : charger le fichier dans Stockage Blob](#)

[Express.js : Ajouter la journalisation des Recommandations d'application](#)

Web + Données

DÉMARRAGE RAPIDE

[Stockage sur Azure](#)

[Serverless de pile complète avec Mongoose](#)

[API serverless + base de données](#)

[Express.js + MongoDB \(formation\)](#)

[Express.js + MongoDB \(docs\)](#)

IA/ML

DÉMARRAGE RAPIDE

[Ajouter la recherche à un site web](#)

[Détection de la langue](#)

[Extraction de phrases clés](#)

[Reconnaissance vocale](#)

[Synthèse vocale](#)

[Analyse d'image](#)

Utiliser des bibliothèques clientes Azure (SDK)

BIEN DÉMARRER

[Utiliser les kits SDK Azure pour JS/TS](#)

[Dernière version du SDK ↗](#)

[Documentation de référence du Kit de développement logiciel \(SDK\)](#)

[Code source du Kit de développement logiciel \(SDK\) pour JS ↗](#)

[Exemples de navigateur](#)

Guides du développeur

 BIEN DÉMARRER

[Guide de développement Stockage Azure](#)

Développement de bout en bout

 BIEN DÉMARRER

[Contoso Solution](#)

[Outils de développeur](#)

[Exemple ↗](#)

Outils de développeur

 BIEN DÉMARRER

[Visual Studio Code \(IDE\) ↗](#)

[Azure CLI](#)

[Azure Developer CLI](#)

[Interface CLI d'Azure Static Web Apps ↗](#)

[Interface CLI des outils de base d'Azure Functions ↗](#)

[Terminal Windows ↗](#)

[Sous-système Windows pour Linux \(WSL\)](#)

Azure pour les développeurs Python

Déployez votre code Python sur Azure pour les applications web, les applications serverless, les conteneurs et les modèles Machine Learning. Tirez parti des bibliothèques Azure (SDK) pour Python afin d'accéder programmatiquement à la gamme complète des services Azure, notamment le stockage, les bases de données, les fonctionnalités IA préconfigurées et bien plus encore.

Bibliothèques Azure (SDK)

BIEN DÉMARRER

[Bien démarrer](#)

[Configurer votre environnement de développement local](#)

[Découvrir les bibliothèques Azure](#)

[Découvrir les modèles d'utilisation des bibliothèques](#)

[S'authentifier avec les Services Azure](#)

les applications web

DIDACTICIEL

[Créer et déployer rapidement de nouvelles applications Django / Flask / FastAPI](#)

[Déployer une application web Django ou Flask](#)

[Déployer une application web avec PostgreSQL](#)

[Déployer une application web avec une identité managée](#)

[Déployer à l'aide de GitHub Actions](#)

Intelligence artificielle

DÉMARRAGE RAPIDE

[Développer à l'aide des services Azure AI](#)

conteneurs



DIDACTICIEL

[Vue d'ensemble des conteneurs Python](#)

[Déployer sur App Service](#)

[Déployer sur Container Apps](#)

[Déployer un cluster Kubernetes](#)

Données et stockage



DÉMARRAGE RAPIDE

[Bases de données SQL](#)

[Tables, objets blob, fichiers, NoSQL](#)

[Big Data et Analytics](#)

Machine Learning



GUIDE PRATIQUE

[Créer une expérience ML](#)

[Entraîner un modèle de prédiction](#)

[Créer des pipelines ML](#)

[Utiliser des services IA prêts à l'emploi \(visage, reconnaissance vocale, texte, image, etc.\)](#)

[ETL serverless cloud](#)

Fonctions serverless



GUIDE PRATIQUE

[Déployer avec Visual Studio Code](#)

[Déployer en utilisant la ligne de commande](#)

[Se connecter au stockage avec Visual Studio Code](#)

[Se connecter au stockage avec la ligne de commande](#)

Outils de développeur



BIEN DÉMARRER

[Visual Studio Code \(IDE\)](#) ↗

[Azure CLI](#)

[Sous-système Windows pour Linux \(WSL\)](#)

[Visual Studio \(pour développement en Python/C++\)](#)

Azure pour les développeurs Go

Découvrez comment utiliser le Kit de développement logiciel (SDK) Azure pour Go, parcourir les références d'API, les exemples de code, les didacticiels, les guides de démarrage rapide, les articles conceptuels, etc.

Bien démarrer

VUE D'ENSEMBLE

[Faire vos premiers pas avec Go](#)

[Qu'est-ce qu'Azure SDK pour Go ?](#)

TÉLÉCHARGER

[Installer le kit de développement logiciel Microsoft Azure SDK pour Go ↗](#)

Données

DÉMARRAGE RAPIDE

[Utiliser Stockage Blob avec Go](#)

[Se connecter à Azure Database pour PostgreSQL](#)

[Se connecter à Azure Database pour MySQL](#)

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

Machines Virtuelles

DÉMARRAGE RAPIDE

[Authentifier avec une identité managée](#)

Sans serveur

DÉMARRAGE RAPIDE

Conteneurs

DÉMARRAGE RAPIDE

[Créer et conteneuriser une application Go ↗](#)

[Azure Container Apps](#)

Open source

RÉFÉRENCE

[Dapr \(Runtime d'application distribuée\) ↗](#)

[KEDA \(Kubernetes Event Driven Autoscaler\) ↗](#)

[Module complémentaire HTTP KEDA ↗](#)

Connexions sans mot de passe pour les services Azure

Article • 23/10/2023

ⓘ Notes

Les connexions sans mot de passe sont une fonctionnalité sans langage spécifié qui couvre plusieurs services Azure. Bien que la documentation actuelle se concentre sur quelques langues et services, nous sommes actuellement en train de produire de la documentation supplémentaire pour d'autres langues et services.

Cet article décrit les problèmes de sécurité liés aux mots de passe et introduit des connexions sans mot de passe pour les services Azure.

Problèmes de sécurité liés aux mots de passe et aux secrets

Les mots de passe et les clés secrètes doivent être utilisés avec précaution, et les développeurs ne doivent jamais les placer dans un emplacement non sécurisé. De nombreuses applications se connectent à la base de données principale, au cache, à la messagerie et aux services d'événements à l'aide de noms d'utilisateur, de mots de passe et de clés d'accès. Si elles sont exposées, ces informations d'identification peuvent être utilisées pour obtenir un accès non autorisé à des informations sensibles telles qu'un catalogue de ventes que vous avez créé pour une campagne à venir ou des données client qui doivent être privées.

L'incorporation de mots de passe dans une application elle-même présente un risque de sécurité énorme pour de nombreuses raisons, notamment la découverte via un référentiel de code. De nombreux développeurs externalisent ces mots de passe à l'aide de variables d'environnement afin que les applications puissent les charger à partir de différents environnements. Toutefois, cela déplace uniquement le risque du code lui-même vers un environnement d'exécution. Toute personne qui accède à l'environnement peut voler des mots de passe, ce qui augmente à son tour le risque d'exfiltration des données.

L'exemple de code suivant montre comment se connecter au stockage Azure à l'aide d'une clé de compte de stockage. De nombreux développeurs gravitent vers cette solution, car il se sent familier des options avec lesquelles ils ont travaillé dans le passé,

même s'il n'est pas une solution idéale. Si votre application utilise actuellement des clés d'accès, envisagez de migrer vers des connexions sans mot de passe.

C#

```
// Connection using secret access keys
BlobServiceClient blobServiceClient = new(
    new Uri("https://<storage-account-name>.blob.core.windows.net"),
    new StorageSharedKeyCredential("<storage-account-name>", "<your-access-key>"));
```

Les développeurs doivent être attentifs à ne jamais exposer ces types de clés ou de secrets dans un emplacement non sécurisé. De nombreuses entreprises ont des exigences de sécurité strictes pour se connecter aux services Azure sans exposer de mots de passe aux développeurs, aux opérateurs ou à toute autre personne. Ils utilisent souvent un coffre pour stocker et charger des mots de passe dans des applications, et réduisent davantage le risque en ajoutant des exigences et procédures de rotation de mot de passe. Cette approche, à son tour, augmente la complexité opérationnelle et, parfois, entraîne des pannes de connexion d'application.

Connexions sans mot de passe et Confiance Zéro

Vous pouvez désormais utiliser des connexions sans mot de passe dans vos applications pour vous connecter aux services Basés sur Azure sans avoir besoin de faire pivoter les mots de passe. Dans certains cas, tout ce dont vous avez besoin est la configuration : aucun nouveau code n'est requis. Confiance Zéro utilise le principe « jamais confiance, toujours vérifier et sans informations d'identification ». Cela signifie la sécurisation de toutes les communications en faisant confiance aux ordinateurs ou aux utilisateurs uniquement après avoir vérifié l'identité et avant de leur accorder l'accès aux services principaux.

L'option d'authentification recommandée pour les connexions sécurisées et sans mot de passe consiste à utiliser des identités managées et un contrôle d'accès en fonction du rôle Azure (RBAC) en combinaison. Avec cette approche, vous n'avez pas besoin de suivre et de gérer manuellement de nombreux secrets différents pour les identités managées, car ces tâches sont gérées en interne en toute sécurité par Azure.

Vous pouvez configurer des connexions sans mot de passe aux services Azure à l'aide du service Connecter or ou vous pouvez les configurer manuellement. Service Connector active les identités managées dans les services d'hébergement d'applications comme Azure Spring Apps, Azure App Service et Azure Container Apps. Le Connecter or de

service configure également les services principaux avec des connexions sans mot de passe à l'aide d'identités managées et d'Azure RBAC, et hydrate les applications avec les informations de connexion nécessaires.

Si vous inspectez l'environnement en cours d'exécution d'une application configurée pour les connexions sans mot de passe, vous pouvez voir la chaîne de connexion complète. Le chaîne de connexion porte, par exemple, une adresse de serveur de base de données, un nom de base de données et une instruction pour déléguer l'authentification à un plug-in d'authentification Azure, mais elle ne contient aucun mot de passe ni secrets.

La vidéo suivante illustre les connexions sans mot de passe des applications aux services Azure, à l'aide d'applications Java comme exemple. Une couverture similaire pour d'autres langues est à venir.

<https://www.youtube-nocookie.com/embed/X6nR3AjlwJw> ↗

Présentation de DefaultAzureCredential

Les connexions sans mot de passe aux services Azure via l'ID Microsoft Entra et le contrôle d'accès en fonction du rôle (RBAC) peuvent être implémentées à `DefaultAzureCredential` partir des bibliothèques clientes Azure Identity.

ⓘ Important

Certains langages doivent être implémentés `DefaultAzureCredential` explicitement dans leur code, tandis que d'autres utilisent `DefaultAzureCredential` en interne via des plug-ins ou des pilotes sous-jacents.

`DefaultAzureCredential` prend en charge plusieurs méthodes d'authentification et détermine automatiquement celle qui doit être utilisée au moment de l'exécution. Cette approche permet à votre application d'utiliser différentes méthodes d'authentification dans différents environnements (développement local et production) sans implémenter de code spécifique à l'environnement.

L'ordre et les emplacements dans lesquels `DefaultAzureCredential` les recherches d'informations d'identification varient entre les langues :

- .NET
- C++ ↗
- Go ↗

- Java
- JavaScript
- Python

Par exemple, lors de l'utilisation locale avec .NET, `DefaultAzureCredential` s'authentifie généralement à l'aide du compte utilisé par le développeur pour se connecter à Visual Studio, Azure CLI, ou Azure PowerShell. Lorsque l'application est déployée sur Azure, `DefaultAzureCredential` découvrira et utilisera automatiquement [l'identité managée](#) du service d'hébergement associé, tel que Azure App Service. Aucune modification du code n'est requise pour cette transition.

ⓘ Notes

Une identité managée fournit une identité de sécurité pour représenter une application ou un service. L'identité est gérée par la plateforme Azure et ne vous oblige pas à provisionner ou à faire pivoter les secrets. Vous pouvez en savoir plus sur les identités managées dans la documentation [vue d'ensemble](#).

L'exemple de code suivant montre comment se connecter à Service Bus à l'aide de connexions sans mot de passe. D'autres documents expliquent comment migrer vers cette configuration pour un service spécifique plus en détail. Une application .NET peut passer une instance du `DefaultAzureCredential` constructeur d'une classe cliente de service. `DefaultAzureCredential` détecte automatiquement les informations d'identification disponibles dans cet environnement.

C#

```
ServiceBusClient serviceBusClient = new(
    new Uri("https://<your-service-bus-namespace>.blob.core.windows.net"),
    new DefaultAzureCredential());
```

Voir aussi

Pour obtenir une explication plus détaillée des connexions sans mot de passe, consultez le guide [du développeur Configurer les connexions sans mot de passe entre plusieurs applications et services Azure](#).

Configurer des connexions sans mot de passe entre plusieurs applications et services Azure

Article • 25/03/2023

Les applications nécessitent souvent des connexions sécurisées entre plusieurs services Azure simultanément. Par exemple, une instance Azure App Service d'entreprise peut se connecter à plusieurs comptes de stockage différents, une instance de base de données Azure SQL, un service bus, etc.

[Les identités managées](#) sont l'option d'authentification recommandée pour les connexions sécurisées sans mot de passe entre les ressources Azure. Les développeurs n'ont pas à suivre et gérer manuellement de nombreux secrets différents pour les identités managées, car la plupart de ces tâches sont gérées en interne par Azure. Ce didacticiel explique comment gérer les connexions entre plusieurs services à l'aide d'identités managées et de la bibliothèque cliente Azure Identity.

Comparer les types d'identités managées

Azure fournit les types d'identités managées suivantes :

- Les **identités managées affectées par le système** sont directement liées à une seule ressource Azure. Lorsque vous activez une identité managée affectée par le système sur un service, Azure crée une identité liée et gère les tâches administratives pour cette identité en interne. Quand la ressource Azure est supprimée, l'identité l'est aussi.
- Les **identités managées affectées par l'utilisateur** sont des identités indépendantes créées par un administrateur et peuvent être associées à une ou plusieurs ressources Azure. Le cycle de vie de l'identité est indépendant de ces ressources.

Vous pouvez en savoir plus sur les meilleures pratiques et quand utiliser des identités affectées par le système et des identités affectées par l'utilisateur dans les [recommandations de meilleures pratiques des identités](#).

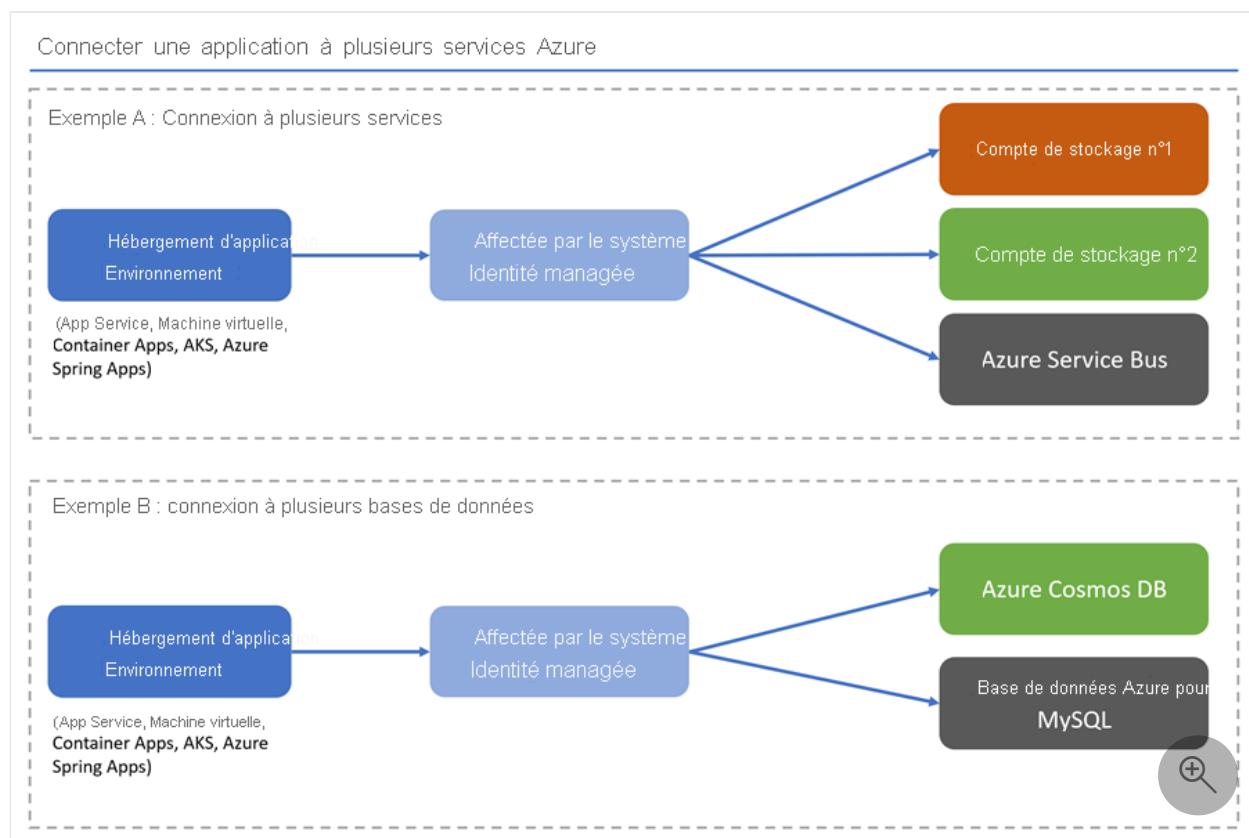
Explorer DefaultAzureCredential

Les identités managées sont généralement implémentées dans votre code d'application via une classe appelée `DefaultAzureCredential` à partir de la `Azure.Identity` bibliothèque cliente. `DefaultAzureCredential` prend en charge plusieurs méthodes d'authentification et détermine automatiquement celle qui doit être utilisée au moment de l'exécution. Vous pouvez en savoir plus sur cette approche dans la [vue d'ensemble defaultAzureCredential](#).

Connecter une application hébergée Azure à plusieurs services Azure

Vous avez été chargé de connecter une application existante à plusieurs services et bases de données Azure à l'aide de connexions sans mot de passe. L'application est une API web ASP.NET Core hébergée sur Azure App Service, bien que les étapes ci-dessous s'appliquent également à d'autres environnements d'hébergement Azure, tels qu'Azure Spring Apps, Machines Virtuelles, Container Apps et AKS.

Ce didacticiel s'applique aux architectures suivantes, bien qu'il puisse être adapté à de nombreux autres scénarios ainsi qu'à des modifications de configuration minimales.



Les étapes suivantes montrent comment configurer une application pour utiliser une identité managée affectée par le système et votre compte de développement local pour se connecter à plusieurs services Azure.

Créer une identité managée affectée par le système

1. Dans le Portail Azure, accédez à l'application hébergée que vous souhaitez vous connecter à d'autres services.
2. Sur la page vue d'ensemble du service, sélectionnez **Identité**.
3. Basculez le paramètre d'**Etat** sur **Activé** pour activer une identité managée affectée par le système pour le service.

The screenshot shows the Azure portal interface for managing an App Service named 'msdocs-web-app-123'. The left sidebar lists various service management options like 'Vue d'ensemble', 'Journal d'activité', and 'Identité'. The 'Identité' option is highlighted with a red box. The main content area displays the 'Attribuée par le système' (Assigned by system) tab, which is also highlighted with a red box. A note states: 'Une identité managée affectée par le système est limitée à une par ressource et liée au cycle de vie de celle-ci. Vous pouvez accorder des autorisations à cette identité.' Below this, there's a 'Statut' (Status) switch set to 'On', and an 'ID de l'objet (principal)' (Object ID) field containing '65634e65-f3eb-4fe7-aff3-fd31d035084b'. At the bottom, a note says: 'Cette ressource est inscrite auprès d'Azure Active Directory. L'identité managée peut être configurée pour autoriser l'accès à d'autres services.' There are also 'Enregistrer' (Save), 'Abandonner' (Cancel), and 'Actualiser' (Update) buttons.

Attribuer des rôles à l'identité managée pour chaque service connecté

1. Accédez à la page vue d'ensemble du compte de stockage auquel vous souhaitez accorder l'accès à votre identité.
2. Sélectionnez **Contrôle d'accès (IAM)** à partir de la navigation du compte de stockage.
3. Choisissez **+ Ajouter**, puis **Ajouter une attribution de rôle**.

4. Dans la zone de recherche de **Rôle**, recherchez *le Contributeur de données Blob de stockage*, qui accorde des autorisations pour effectuer des opérations de lecture et d'écriture sur les données d'objets blob. Vous pouvez attribuer le rôle approprié pour votre cas d'usage. Sélectionnez *le contributeur de données blob de stockage* dans la liste, puis choisissez **Suivant**.
5. Sur l'écran **Ajouter une attribution de rôle**, pour l'option **Attribuer l'accès à l'option**, sélectionnez **Identité managée**. Choisissez ensuite +**Sélectionner des membres**.
6. Dans le menu volant, recherchez l'identité managée que vous avez créée en entrant le nom de votre service app. Sélectionnez l'identité affectée par le système, puis choisissez **Sélectionner** pour fermer le menu volant.

7. Sélectionnez **Suivant** plusieurs fois jusqu'à ce que vous puissiez sélectionner **Vérifier + attribuer** pour terminer l'attribution de rôle.

8. Répétez ce processus pour les autres services auxquels vous souhaitez vous connecter.

Considérations relatives au développement local

Vous pouvez également activer l'accès aux ressources Azure pour le développement local en affectant des rôles à un compte d'utilisateur de la même façon que vous avez attribué des rôles à votre identité managée.

1. Après avoir attribué le rôle **Contributeur de données Blob de stockage** à votre identité managée, sous **Attribuer l'accès à**, cette fois sélectionnez **Utilisateur, groupe ou principal de service**. Choisissez + **Sélectionner des membres** pour ouvrir à nouveau le menu volant.
2. Recherchez le *compte user@domain* ou le groupe de sécurité Microsoft Entra auquel vous souhaitez accorder l'accès par adresse e-mail ou par nom, puis sélectionnez-le. Il doit s'agir du même compte que celui que vous utilisez pour vous connecter à vos outils de développement local, tels que Visual Studio ou Azure CLI.

ⓘ Notes

Vous pouvez également attribuer ces rôles à un groupe de sécurité Microsoft Entra si vous travaillez sur une équipe avec plusieurs développeurs. Vous pouvez ensuite placer n'importe quel développeur dans ce groupe qui a besoin d'accéder au développement de l'application localement.

Implémenter le code de l'application

C#

À l'intérieur de votre projet, ajoutez une référence au package NuGet `Azure.Identity`. Cette bibliothèque contient toutes les entités nécessaires pour implémenter `DefaultAzureCredential`. Vous pouvez également ajouter toutes les autres bibliothèques Azure qui sont pertinentes pour votre application. Pour cet exemple, les packages `Azure.Storage.Blobs` et `Azure.KeyVault.Keys` sont ajoutés pour se connecter au stockage Blob et Key Vault.

CLI .NET

```
dotnet add package Azure.Identity  
dotnet add package Azure.Storage.Blobs  
dotnet add package Azure.KeyVault.Keys
```

En haut de votre fichier `Program.cs`, ajoutez les instructions d'utilisation suivantes :

C#

```
using Azure.Identity;  
using Azure.Storage.Blobs;  
using Azure.Security.KeyVault.Keys;
```

Dans le fichier `Program.cs` de votre code de projet, créez des instances des services nécessaires auxquels votre application se connecte. Les exemples suivants se connectent au stockage Blob et au service bus à l'aide des classes SDK correspondantes.

C#

```
var blobServiceClient = new BlobServiceClient(  
    new Uri("https://<your-storage-account>.blob.core.windows.net"),  
    new DefaultAzureCredential(credOptions));  
  
var serviceBusClient = new ServiceBusClient("<your-namespace>", new  
DefaultAzureCredential());  
var sender = serviceBusClient.CreateSender("producttracking");
```

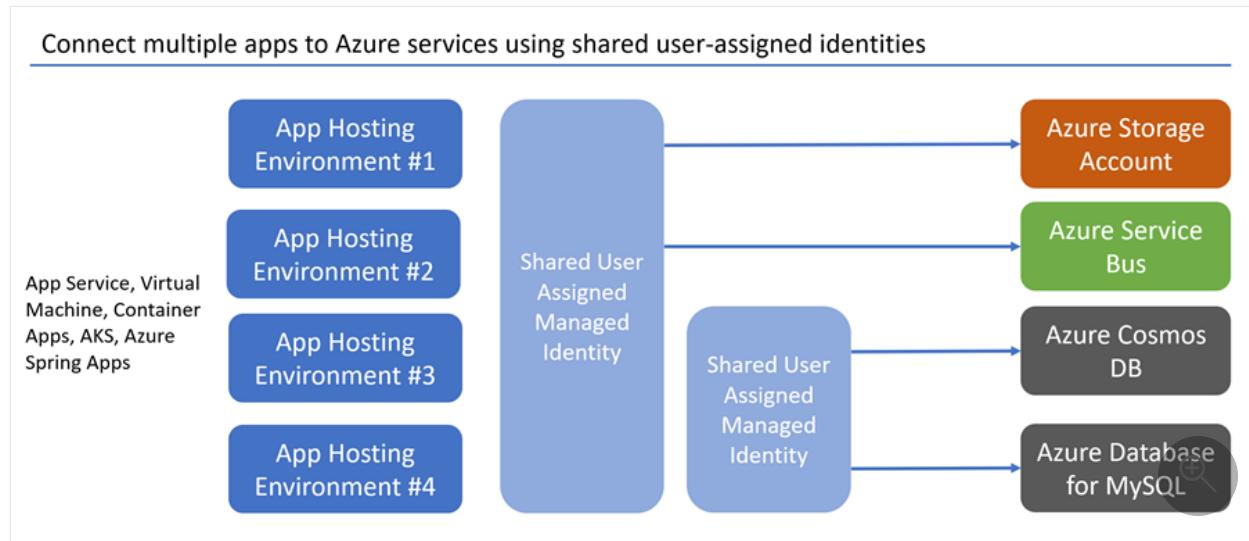
Lorsque ce code d'application s'exécute localement, `DefaultAzureCredential` recherchera une chaîne d'informations d'identification pour les premières informations d'identification disponibles. Si la valeur `Managed_Identity_Client_ID` est null localement, elle utilise automatiquement les informations d'identification de votre interface de ligne de commande Azure locale ou de la connexion Visual Studio. Vous pouvez en savoir plus sur ce processus dans la [vue d'ensemble de la bibliothèque Azure Identity](#).

Lorsque l'application est déployée sur Azure, `DefaultAzureCredential` récupère automatiquement la variable `Managed_Identity_Client_ID` à partir de l'environnement App Service. Cette valeur devient disponible lorsqu'une identité managée est associée à votre application.

Ce processus global garantit que votre application peut s'exécuter en toute sécurité localement et dans Azure sans avoir besoin de modifications de code.

Connectez plusieurs applications à l'aide de plusieurs identités managées

Bien que les applications de l'exemple précédent partagent toutes les mêmes exigences d'accès au service, les environnements réels sont souvent plus nuancés. Envisagez un scénario où plusieurs applications se connectent aux mêmes comptes de stockage, mais deux des applications accèdent également à différents services ou bases de données.



Pour configurer cette configuration dans votre code, assurez-vous que votre application inscrit des services distincts pour se connecter à chaque compte de stockage ou base de données. Veillez à extraire les ID du client d'identité managée corrects pour chaque service lors de la configuration de `DefaultAzureCredential`. L'exemple de code suivant configure les connexions de service suivantes :

- Deux connexions à des comptes de stockage distincts à l'aide d'une identité managée affectée par l'utilisateur partagé
- Connexion aux services Azure Cosmos DB et Azure SQL à l'aide d'une deuxième identité managée affectée par l'utilisateur partagé

C#

```
C#  
  
// Get the first user-assigned managed identity ID to connect to shared storage  
var clientIDstorage =  
Environment.GetEnvironmentVariable("Managed_Identity_Client_ID_Storage")  
;  
  
// First blob storage client that using a managed identity  
BlobServiceClient blobServiceClient = new BlobServiceClient(  
    new Uri("https://<receipt-storage-account>.blob.core.windows.net"),
```

```

        new DefaultAzureCredential()
    {
        ManagedIdentityClientId = clientIDstorage
    });

    // Second blob storage client that using a managed identity
    BlobServiceClient blobServiceClient2 = new BlobServiceClient(
        new Uri("https://<contract-storage-account>.blob.core.windows.net"),
        new DefaultAzureCredential()
    {
        ManagedIdentityClientId = clientIDstorage
    });

    // Get the second user-assigned managed identity ID to connect to shared
    // databases
    var clientIDdatabases =
    Environment.GetEnvironmentVariable("Managed_Identity_Client_ID_Databases"
    ");

    // Create an Azure Cosmos DB client
    CosmosClient client = new CosmosClient(
        accountEndpoint:
    Environment.GetEnvironmentVariable("COSMOS_ENDPOINT",
    EnvironmentVariableTarget.Process),
        new DefaultAzureCredential()
    {
        ManagedIdentityClientId = clientIDdatabases
    });

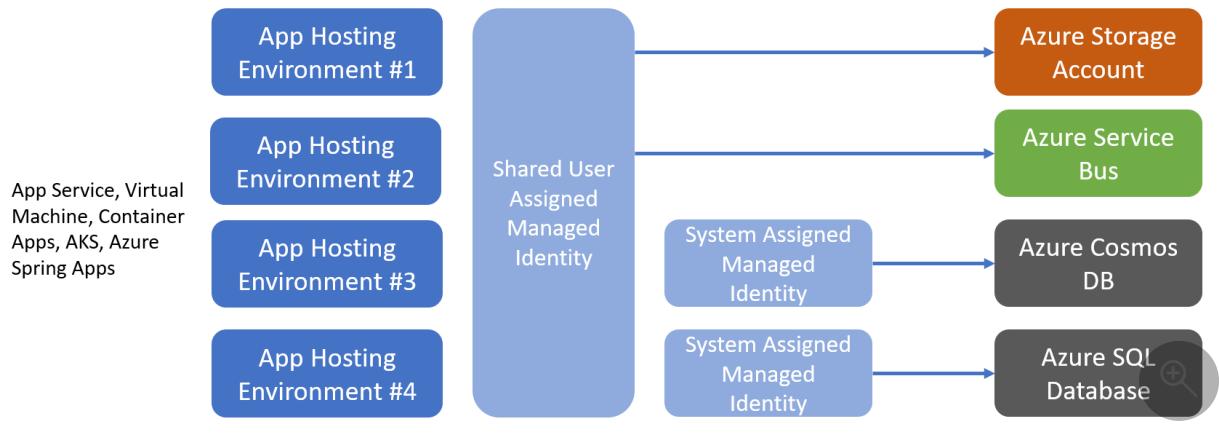
    // Open a connection to Azure SQL using a managed identity
    string ConnectionString1 = @"Server=<azure-sql-
    hostname>.database.windows.net; User Id=ObjectIdOfManagedIdentity;
    Authentication=Active Directory Default; Database=<database-name>";

    using (SqlConnection conn = new SqlConnection(ConnectionString1))
    {
        conn.Open();
    }

```

Vous pouvez également associer une identité managée affectée par l'utilisateur ainsi qu'une identité managée affectée par le système à une ressource simultanément. Cela peut être utile dans les scénarios où toutes les applications nécessitent l'accès aux mêmes services partagés, mais l'une des applications a également une dépendance très spécifique sur un service supplémentaire. L'utilisation d'une identité affectée par le système garantit également que l'identité liée à cette application spécifique est supprimée lorsque l'application est supprimée, ce qui peut vous aider à nettoyer votre environnement.

Connect multiple apps to Azure services using user-assigned and system-assigned identities



Ces types de scénarios sont explorés plus en détail dans les [recommandations de meilleures pratiques des identités](#).

Étapes suivantes

Dans ce didacticiel, vous avez appris à migrer une application vers des connexions sans mot de passe. Vous pouvez lire les ressources suivantes pour explorer les concepts abordés dans cet article plus en détails :

- [Autoriser l'accès aux objets blob à l'aide Microsoft Entra ID](#)
- Pour en savoir plus sur .NET Core, consultez [Prise en main de .NET en 10 minutes](#).

Recommandations relatives aux meilleures pratiques liées aux identités managées

Article • 28/10/2023

Les identités managées pour les ressources Azure sont une fonctionnalité de Microsoft Entra ID. Les [services Azure prenant en charge les identités managées pour ressources Azure](#) sont soumis à leur propre chronologie. Assurez-vous de passer en revue l'état [Disponibilité](#) des identités gérées pour votre ressource et les [problèmes connus](#) avant de commencer.

Choix d'identités managées affectées par le système ou par l'utilisateur

Les identités managées affectées par l'utilisateur sont plus efficaces dans un plus grand nombre de scénarios que les identités managées affectées par le système. Consultez le tableau ci-dessous pour découvrir quelques scénarios et les recommandations relatives aux identités affectées par l'utilisateur ou par le système.

Les identités affectées par l'utilisateur peuvent être utilisées par plusieurs ressources, et leurs cycles de vie sont dissociés des cycles de vie des ressources auxquelles elles sont associées. [Découvrez quelles ressources prennent en charge les identités managées](#).

Ce cycle de vie vous permet de séparer vos responsabilités en matière de création de ressources et d'administration des identités. Les identités affectées par l'utilisateur et leurs attributions de rôle peuvent être configurées avant les ressources qui en ont besoin. Les utilisateurs qui créent les ressources ont uniquement besoin de l'accès pour attribuer une identité affectée par l'utilisateur, sans avoir à créer de nouvelles identités ou attributions de rôle.

Comme les identités affectées par le système sont créées et supprimées en même temps que la ressource, les attributions de rôle ne peuvent pas être créées à l'avance. Cette séquence peut provoquer des échecs lors du déploiement d'une infrastructure si l'utilisateur qui crée la ressource n'a pas également accès à la création d'attributions de rôle.

Si votre infrastructure nécessite que plusieurs ressources aient accès aux mêmes ressources, une seule identité affectée par l'utilisateur peut leur être attribuée. La

surcharge relative à l'administration sera moins importante, car il y a moins d'identités et d'attributions de rôle distinctes à gérer.

Si vous avez besoin que chaque ressource ait sa propre identité ou si vous avez des ressources qui nécessitent un ensemble unique d'autorisations et que vous souhaitez que l'identité soit supprimée en même temps que la ressource, vous devez utiliser une identité affectée par le système.

 Agrandir le tableau

Scénario	Recommandation	Notes
Création rapide de ressources (par exemple, calcul éphémère) avec des identités managées	Identité attribuée par l'utilisateur	<p>Si vous tentez de créer plusieurs identités managées dans un court laps de temps (par exemple, en déployant plusieurs machines virtuelles avec leur propre identité affectée par le système), vous risquez de dépasser la limite de débit pour les créations d'objets Microsoft Entra, et la demande échouera avec une erreur HTTP 429.</p> <p>Si des ressources sont créées ou supprimées rapidement, vous risquez également de dépasser la limite du nombre de ressources dans Microsoft Entra ID si vous utilisez des identités affectées par le système. Bien qu'une identité affectée par le système qui a été supprimée ne soit plus accessible par aucune ressource, elle sera comptabilisée dans votre limite jusqu'à ce qu'elle soit supprimée définitivement après 30 jours.</p> <p>Le déploiement de ressources associées à une seule identité affectée par l'utilisateur nécessite la création d'un seul principal de service dans Microsoft Entra ID, ce qui permet d'éviter la limite du débit. L'utilisation d'une seule identité créée à l'avance réduira également le risque de retards de réPLICATION qui pourraient se produire si plusieurs ressources sont créées, chacune avec sa propre identité.</p>
Ressources/applications répliquées	Identité attribuée par l'utilisateur	<p>En savoir plus sur les limites du service d'abonnement Azure.</p> <p>Les ressources qui effectuent la même tâche, par exemple, des serveurs web dupliqués ou</p>

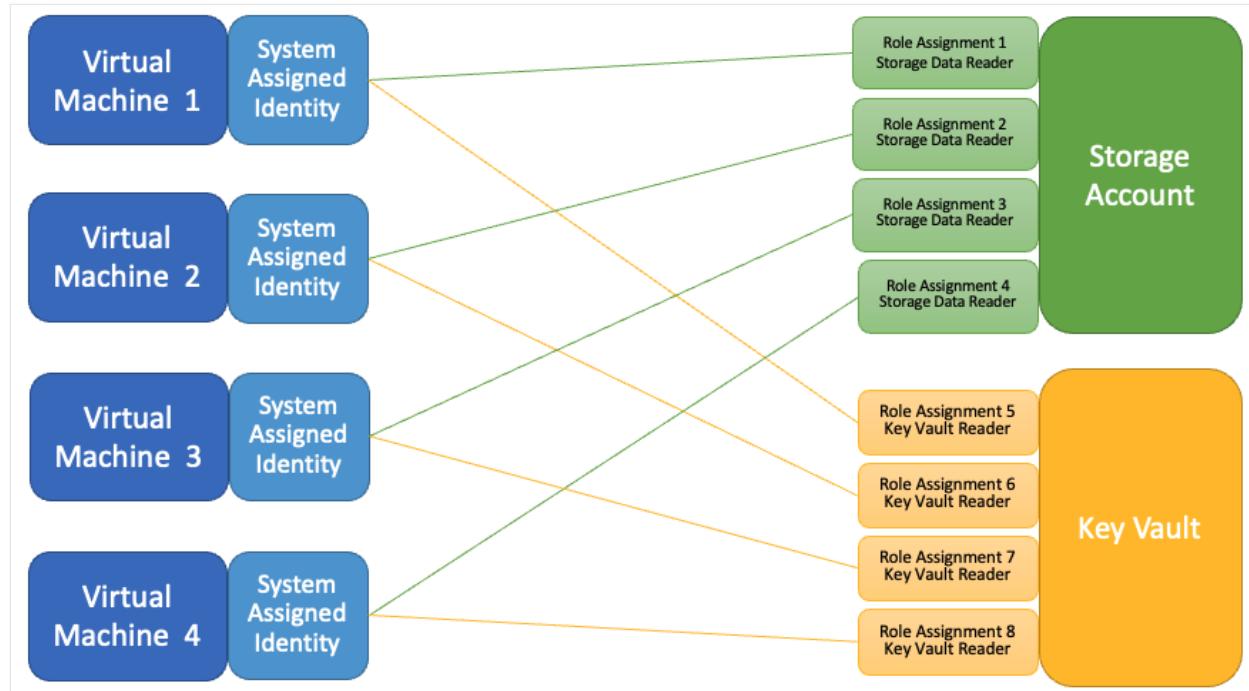
Scénario	Recommandation	Notes
		<p>des fonctionnalités identiques qui s'exécutent dans un service d'application et dans une application sur une machine virtuelle, requièrent généralement les mêmes autorisations.</p> <p>En utilisant la même identité affectée par l'utilisateur, moins d'attributions de rôle sont nécessaires, ce qui réduit la charge de gestion. Les ressources n'ont pas besoin d'être du même type.</p>
Conformité	Identité attribuée par l'utilisateur	<p>Si votre organisation exige que toute création d'identité passe par un processus d'approbation, l'utilisation d'une seule identité affectée par l'utilisateur sur plusieurs ressources nécessitera moins d'approbations que les identités affectées par le système, qui sont créées à mesure que de nouvelles ressources sont créées.</p>
Accès requis avant le déploiement d'une ressource	Identité attribuée par l'utilisateur	<p>Des ressources peuvent nécessiter un accès à certaines ressources Azure dans le cadre de leur déploiement.</p> <p>Dans ce cas, il se peut qu'une identité affectée par le système ne soit pas créée à temps ; il convient donc d'utiliser une identité affectée par l'utilisateur préexistante.</p>
Journalisation d'audit	Identité attribuée par le système	<p>Si vous devez enregistrer quelle ressource spécifique a effectué une action, plutôt que quelle identité, utilisez une identité affectée par le système.</p>
Gestion du cycle de vie des autorisations	Identité attribuée par le système	<p>Si vous souhaitez que les autorisations d'une ressource soient supprimées en même temps que cette ressource, utilisez une identité affectée par le système.</p>

Utilisation des identités affectées par l'utilisateur pour réduire l'administration

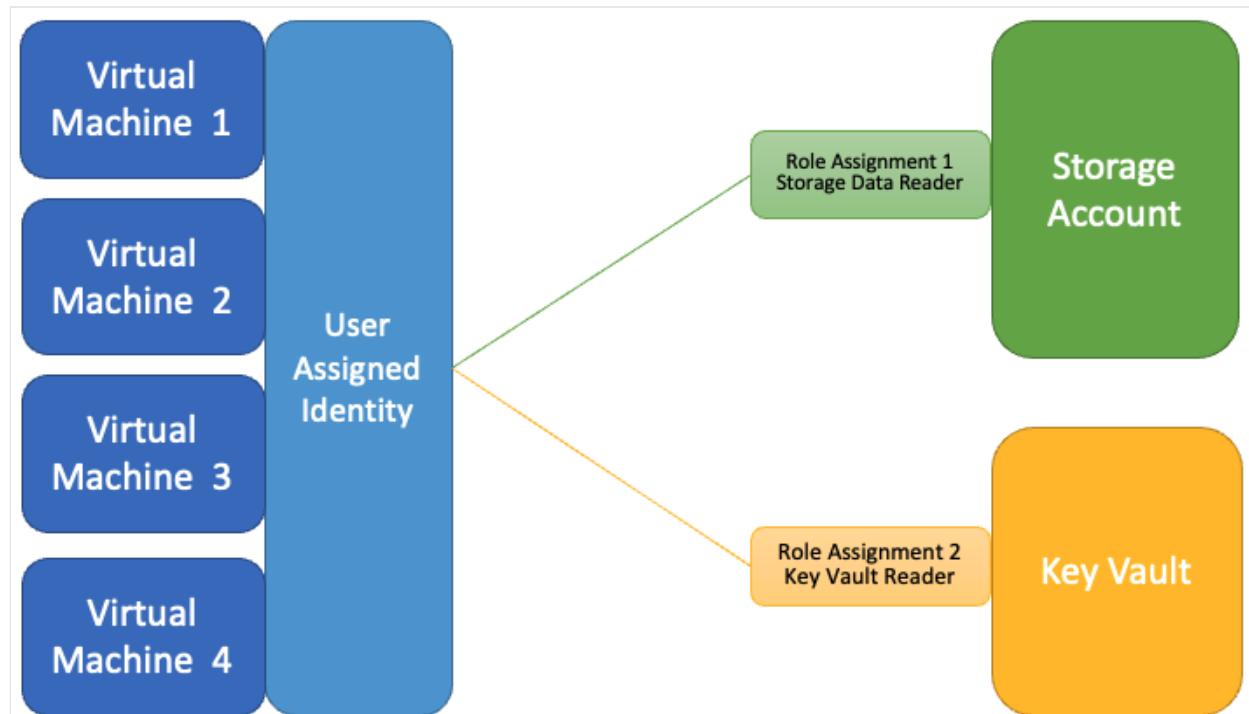
Les diagrammes illustrent la différence entre les identités affectées par le système et celles affectées par l'utilisateur lorsqu'elles sont utilisées pour permettre à plusieurs

machines virtuelles d'accéder à deux comptes de stockage.

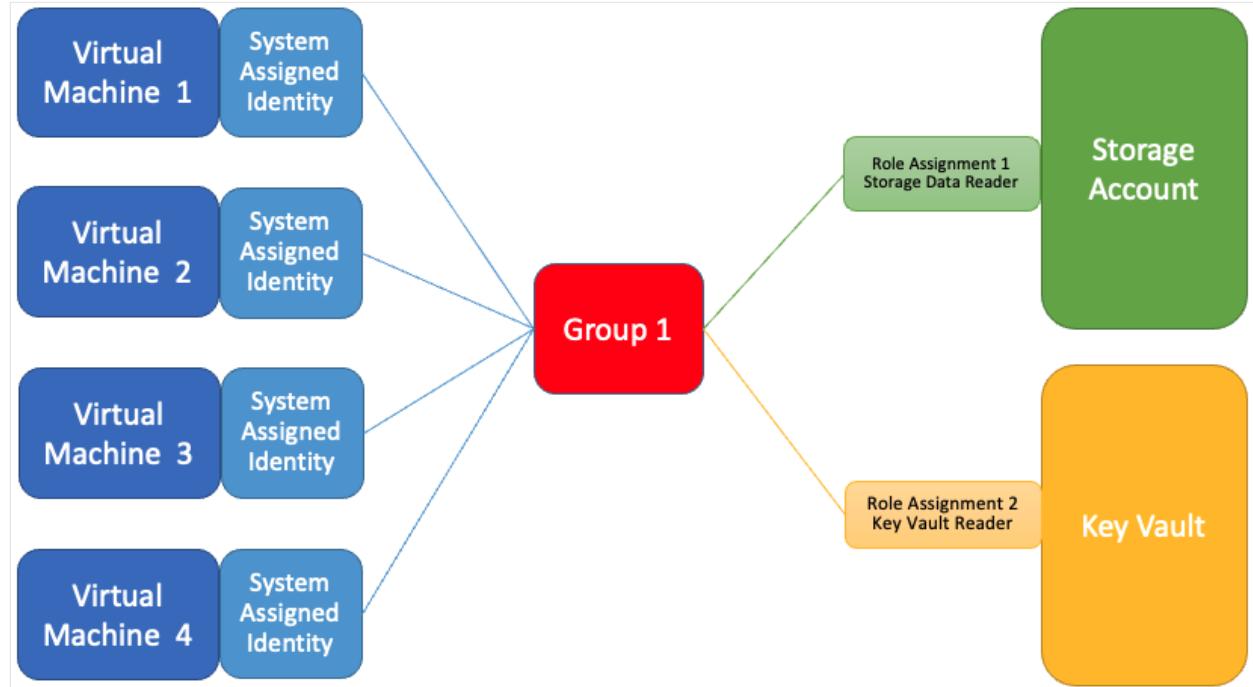
Le diagramme montre quatre machines virtuelles avec des identités affectées par le système. Chaque machine virtuelle a les mêmes attributions de rôle qui lui permettent d'accéder à deux comptes de stockage.



Lorsqu'une identité affectée par l'utilisateur est associée aux quatre machines virtuelles, seules deux attributions de rôle sont requises, contre huit avec des identités affectées par le système. Si l'identité des machines virtuelles requiert davantage d'attributions de rôle, elles seront accordées à toutes les ressources associées à cette identité.



Des groupes de sécurité peuvent également être utilisés pour réduire le nombre d'attributions de rôle requises. Ce diagramme illustre quatre machines virtuelles avec des identités affectées par le système, qui ont été ajoutées à un groupe de sécurité, avec les attributions de rôle ajoutées au groupe au lieu des identités affectées par le système. Bien que le résultat soit similaire, cette configuration n'offre pas les mêmes capacités de modèle Resource Manager que les identités affectées par l'utilisateur.

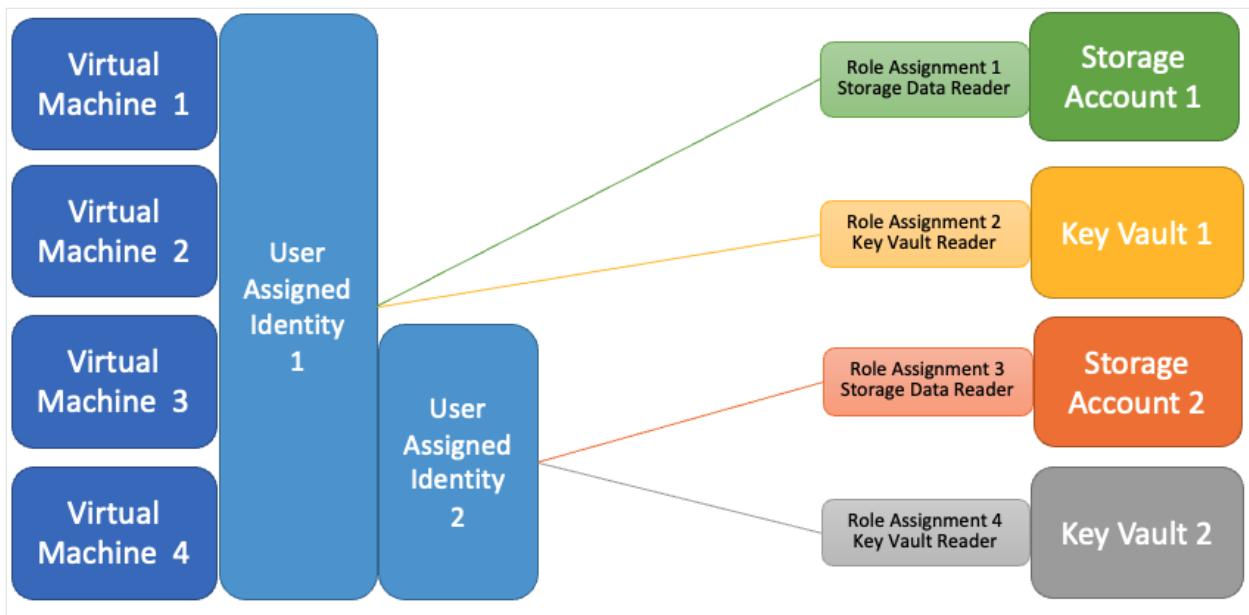


Plusieurs identités managées

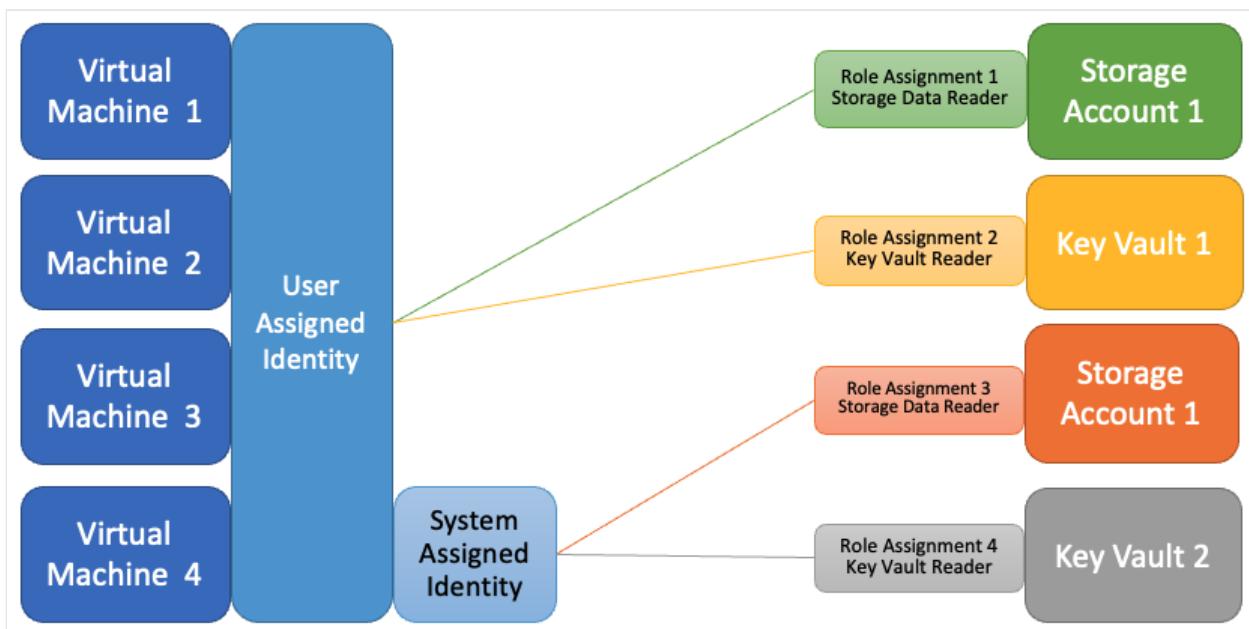
Les ressources qui prennent en charge les identités managées peuvent avoir une identité affectée par le système et une ou plusieurs identités affectées par l'utilisateur.

Ce modèle offre la flexibilité nécessaire pour utiliser une identité affectée par l'utilisateur partagée et appliquer des autorisations granulaires si nécessaire.

Dans l'exemple ci-dessous, « Machine virtuelle 3 » et « Machine virtuelle 4 » peuvent accéder à la fois aux comptes de stockage et aux coffres de clés, selon l'identité affectée par l'utilisateur qu'elles utilisent lors de l'authentification.



Dans l'exemple ci-dessous, « Machine virtuelle 4 » possède à la fois une identité affectée par l'utilisateur et une autre affectée par le système, ce qui lui donne accès aux comptes de stockage et aux coffres de clés, selon l'identité qu'elle utilise lors de l'authentification. Les attributions de rôle pour l'identité affectée par le système sont spécifiques à cette machine virtuelle.



limites

Consultez les limites des [identités managées](#) et des [rôles personnalisés et attributions de rôle](#).

Suivre le principe du moindre privilège lors de l'octroi de l'accès

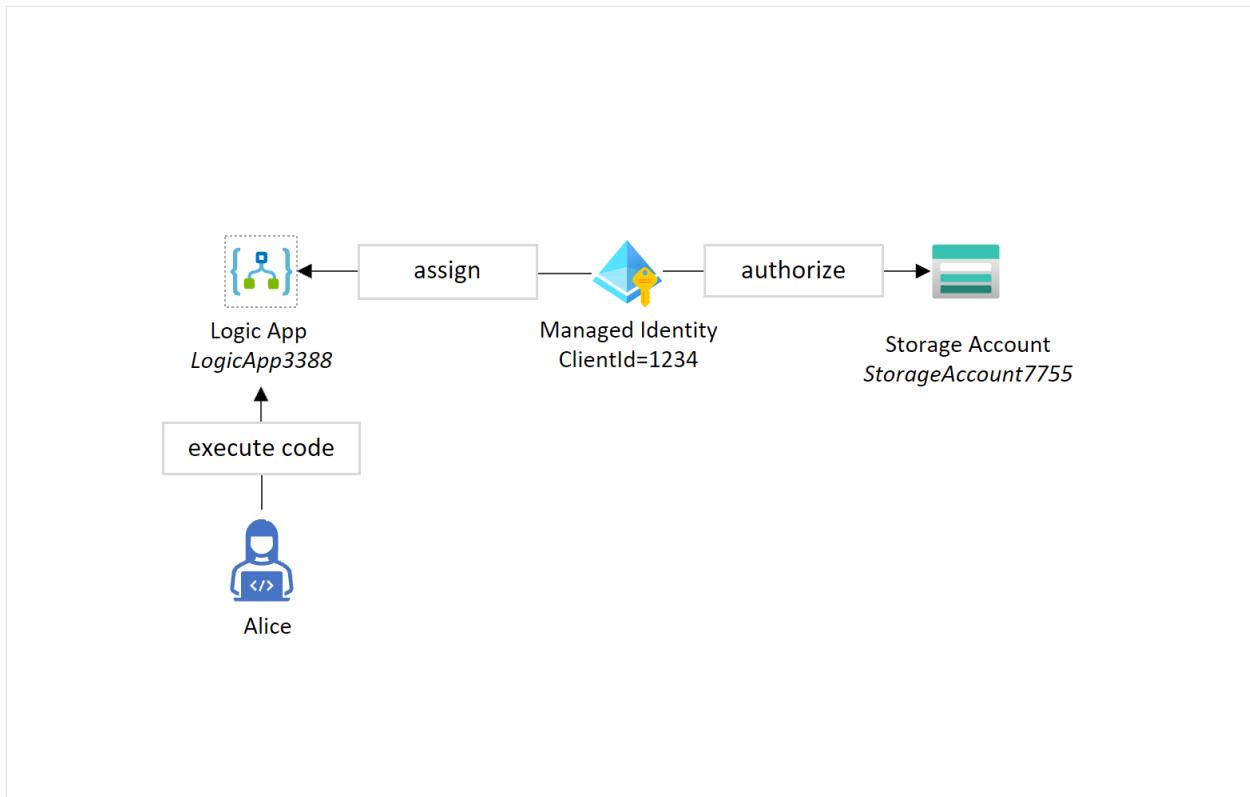
Lorsque vous accordez une identité, y compris une identité managée, aux services, accordez toujours les autorisations minimales nécessaires pour effectuer les actions souhaitées. Par exemple, si une identité managée est utilisée pour lire des données à partir d'un compte de stockage, il n'est pas nécessaire d'autoriser les autorisations d'accès à écrire des données dans le compte de stockage. En accordant des autorisations supplémentaires, par exemple, en faisant de l'identité managée un contributeur sur un abonnement Azure lorsqu'il n'est pas nécessaire, augmente le rayon de sécurité associé à l'identité. L'un doit toujours réduire le rayon de sécurité pour compromettre l'identité.

Tenir compte de l'effet de l'attribution d'identités managées à des ressources Azure et/ou de l'octroi d'autorisations d'attribution à un utilisateur

Il est important de noter que lorsqu'une ressource Azure, telle qu'une application logique Azure, une fonction Azure ou une machine virtuelle, etc. reçoit une identité managée, toutes les autorisations accordées à l'identité managée sont désormais disponibles pour la ressource Azure. Ceci est important, car si un utilisateur a accès à l'installation ou à l'exécution de code sur cette ressource, il a accès à toutes les identités attribuées/associées à la ressource Azure. L'objectif d'une identité managée est de permettre au code s'exécutant sur un accès aux ressources Azure d'accéder à d'autres ressources, sans que les développeurs aient besoin de gérer ou de placer directement les informations d'identification dans le code pour obtenir cet accès.

Par exemple, si une identité managée (ClientId = 1234) a reçu un accès en lecture/écriture à **StorageAccount7755** et qu'elle a été attribuée à **LogicApp3388**, Alice, qui n'a pas d'accès direct au compte de stockage, mais qui a l'autorisation d'exécuter du code dans **LogicApp3388**, peut également lire/écrire des données vers/à partir de **StorageAccount7755** en exécutant le code utilisant l'identité managée.

De même, si Alice dispose des autorisations nécessaires pour attribuer elle-même l'identité managée, elle peut l'affecter à une autre ressource Azure et avoir accès à toutes les autorisations disponibles pour l'identité managée.



En général, lorsqu'on accorde à un utilisateur un accès administratif à une ressource qui peut exécuter du code (comme une Logic App) et qui a une identité gérée, il faut se demander si le rôle attribué à l'utilisateur peut installer ou exécuter du code sur la ressource, et dans l'affirmative, n'attribuer ce rôle que si l'utilisateur en a vraiment besoin.

Maintenance

Les identités affectées par le système sont automatiquement supprimées lorsque la ressource est supprimée, tandis que le cycle de vie d'une identité affectée par l'utilisateur est indépendant des ressources auxquelles elle est associée.

Vous devez supprimer manuellement une identité affectée par l'utilisateur lorsqu'elle n'est plus nécessaire, même si aucune ressource ne lui est associée.

Les attributions de rôle ne sont pas supprimées automatiquement lorsque les identités managées affectées par le système ou par l'utilisateur sont supprimées. Ces attributions de rôle doivent être supprimées manuellement afin de ne pas dépasser la limite d'attributions de rôle par abonnement.

Les attributions de rôle associées aux identités managées supprimées apparaissent avec le message « Identité introuvable » lorsqu'elles sont affichées dans le portail. [En savoir plus.](#)

Storage Blob Data Reader

	Identity not found. ⓘ Unable to find identity.	Unknown
--	---	---------

Les attributions de rôles qui ne sont plus associées à un utilisateur ou à un principal de service s'affichent avec une valeur `ObjectType` de `Unknown`. Pour les supprimer, vous pouvez diriger plusieurs commandes Azure PowerShell ensemble pour obtenir d'abord toutes les attributions de rôles, filtrer uniquement celles avec une valeur `ObjectType` de `Unknown`, puis supprimer ensuite ces attributions de rôles d'Azure.

Azure PowerShell

```
Get-AzRoleAssignment | Where-Object {$__.ObjectType -eq "Unknown"} | Remove-AzRoleAssignment
```

Limitation de l'utilisation des identités managées pour l'autorisation

L'utilisation de **groupes** Microsoft Entra ID pour accorder l'accès aux services est un excellent moyen de simplifier le processus d'autorisation. L'idée est simple : accordez des autorisations à un groupe et ajoutez des identités au groupe afin qu'elles héritent des mêmes autorisations. Il s'agit d'un modèle bien établi provenant de différents systèmes locaux et qui fonctionne bien lorsque les identités représentent des utilisateurs. Une autre option pour contrôler l'autorisation dans Microsoft Entra ID consiste à utiliser des **Rôles d'application**, ce qui vous permet de déclarer des **rôles** spécifiques à une application (plutôt que des groupes, qui sont un concept global dans l'annuaire). Vous pouvez ensuite [attribuer des rôles d'application à des identités managées](#) (ainsi qu'à des utilisateurs ou des groupes).

Dans les deux cas, pour les identités non-humaines comme les applications et les identités managées Microsoft Entra, le mécanisme exact de présentation de ces informations d'autorisation à l'application n'est pas parfaitement adapté aujourd'hui. L'implémentation actuelle avec Microsoft Entra ID et le contrôle d'accès en fonction du rôle (Azure RBAC) utilise des jetons d'accès émis par Microsoft Entra ID pour l'authentification de chaque identité. Si l'identité est ajoutée à un groupe ou à un rôle, elle est exprimée sous la forme de revendications dans le jeton d'accès émis par Microsoft Entra ID. Azure RBAC utilise ces revendications pour évaluer plus en détail les règles d'autorisation afin d'autoriser ou de refuser l'accès.

Étant donné que les groupes et les rôles de l'identité sont des revendications dans le jeton d'accès, les modifications d'autorisation ne prennent pas effet tant que le jeton n'est pas actualisé. Pour un utilisateur humain, ce n'est généralement pas un problème, parce qu'un utilisateur peut acquérir un nouveau jeton d'accès en se déconnectant et se reconnectant à nouveau (ou en patientant jusqu'à ce que le jeton expire, soit au bout d'une heure par défaut). En revanche, les jetons d'identité managée sont mis en cache par l'infrastructure Azure sous-jacente à des fins de performances et de résilience : les services back-end pour les identités managées conservent un cache par URL de ressource pendant 24 heures environ. Cela signifie que les modifications de l'appartenance à un rôle ou un groupe d'une identité managée peuvent prendre plusieurs heures avant de prendre effet. Aujourd'hui, il n'est pas possible de forcer l'actualisation du jeton d'une identité managée avant son expiration. Si vous modifiez l'appartenance à un rôle ou un groupe d'une identité managée pour ajouter ou supprimer des autorisations, il peut être nécessaire de patienter plusieurs heures avant que la ressource Azure utilisant l'identité dispose de l'accès approprié.

Si ce délai n'est pas acceptable pour vos exigences, considérez les alternatives à l'utilisation de groupes ou de rôles dans le jeton. Pour que les modifications apportées aux autorisations des identités managées prennent effet rapidement, nous vous recommandons de regrouper les ressources Azure à l'aide d'une [identité managée affectée par l'utilisateur](#) avec des autorisations appliquées directement à l'identité, au lieu d'ajouter ou de supprimer des identités managées dans un groupe Microsoft Entra disposant d'autorisations. Une identité managée attribuée par l'utilisateur peut être utilisée comme un groupe, car elle peut être attribuée à une ou plusieurs ressources Azure pour l'utiliser. L'opération d'assignation peut être contrôlée à l'aide du [Contributeur d'identité managée](#) et [Rôle opérateur d'identité managée](#).

Migrer une application .NET pour utiliser des connexions sans mot de passe avec Azure SQL Database

Article • 05/05/2023

S'applique à  [Azure SQL Database](#)

Les requêtes des applications adressées à Azure SQL Database doivent être authentifiées. Bien qu'il existe plusieurs options pour l'authentification auprès d'Azure SQL Database, vous devez donner la priorité aux connexions sans mot de passe dans vos applications quand cela est possible. Les méthodes d'authentification traditionnelles qui utilisent des mots de passe ou des clés secrètes créent des risques et des complications de sécurité. Visitez le hub de [connexions sans mot de passe pour Azure Services](#) pour découvrir l'avantage des connexions sans mot de passe. Le tutoriel suivant explique comment migrer une application existante pour se connecter à Azure SQL Database afin d'utiliser des connexions sans mot de passe au lieu d'une solution avec nom d'utilisateur et mot de passe.

Configurer Azure SQL Database

Les connexions sans mot de passe utilisent l'authentification Microsoft Entra pour se connecter aux services Azure, y compris la base de données Azure SQL.

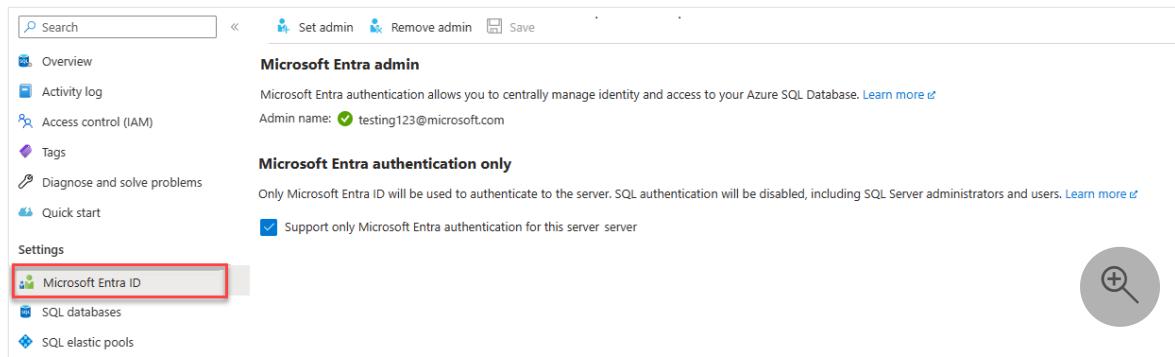
L'authentification Microsoft Entra facilite la gestion des identités dans une position centrale et simplifie la gestion des autorisations. Découvrez comment configurer l'authentification Microsoft Entra pour votre base de données Azure SQL :

- [Vue d'ensemble de l'authentification Microsoft Entra](#)
- [Configurer l'authentification Microsoft Entra](#)

Pour le présent guide de migration, assurez-vous qu'un administrateur Microsoft Entra est affecté à votre base de données Azure SQL.

1. Accédez à la page **Microsoft Entra** de votre serveur logique.
2. Sélectionnez **Définir l'administrateur** pour ouvrir le menu à contrôle suspendu de **Microsoft Entra ID**.
3. Dans le menu à contrôle suspendu de **Microsoft Entra ID**, recherchez l'utilisateur auquel vous souhaitez attribuer le rôle d'administrateur.

4. Sélectionnez l'utilisateur et choisissez Sélectionner.



The screenshot shows the Microsoft Entra admin center interface. At the top, there's a search bar and navigation links for Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Quick start, Set admin, Remove admin, and Save. Below these, the 'Microsoft Entra admin' section is shown with the sub-section 'Microsoft Entra authentication only'. It states that only Microsoft Entra ID will be used for authentication, and SQL authentication will be disabled. A checkbox labeled 'Support only Microsoft Entra authentication for this server' is checked. In the 'Settings' section, there are three options: 'Microsoft Entra ID' (which is highlighted with a red box), 'SQL databases', and 'SQL elastic pools'. A magnifying glass icon is in the bottom right corner.

Configurer votre environnement de développement local

Les connexions sans mot de passe peuvent être configurées de telle manière qu'elles fonctionnent à la fois pour les environnements locaux et hébergés par Azure. Dans cette section, vous allez appliquer des configurations afin de permettre aux utilisateurs individuels de s'authentifier auprès d'Azure SQL Database pour le développement local.

Se connecter à Azure

Pour le développement local, assurez-vous que vous êtes connecté avec le même compte Azure AD que celui que vous souhaitez utiliser pour accéder à Azure SQL Database. Vous pouvez vous authentifier au moyen d'outils de développement populaires, comme Azure CLI ou Azure PowerShell. Les outils de développement avec lesquels vous pouvez vous authentifier dépendent de la langue.

Azure CLI

Connectez-vous à Azure via Azure CLI à l'aide de la commande suivante :

```
Azure CLI
```

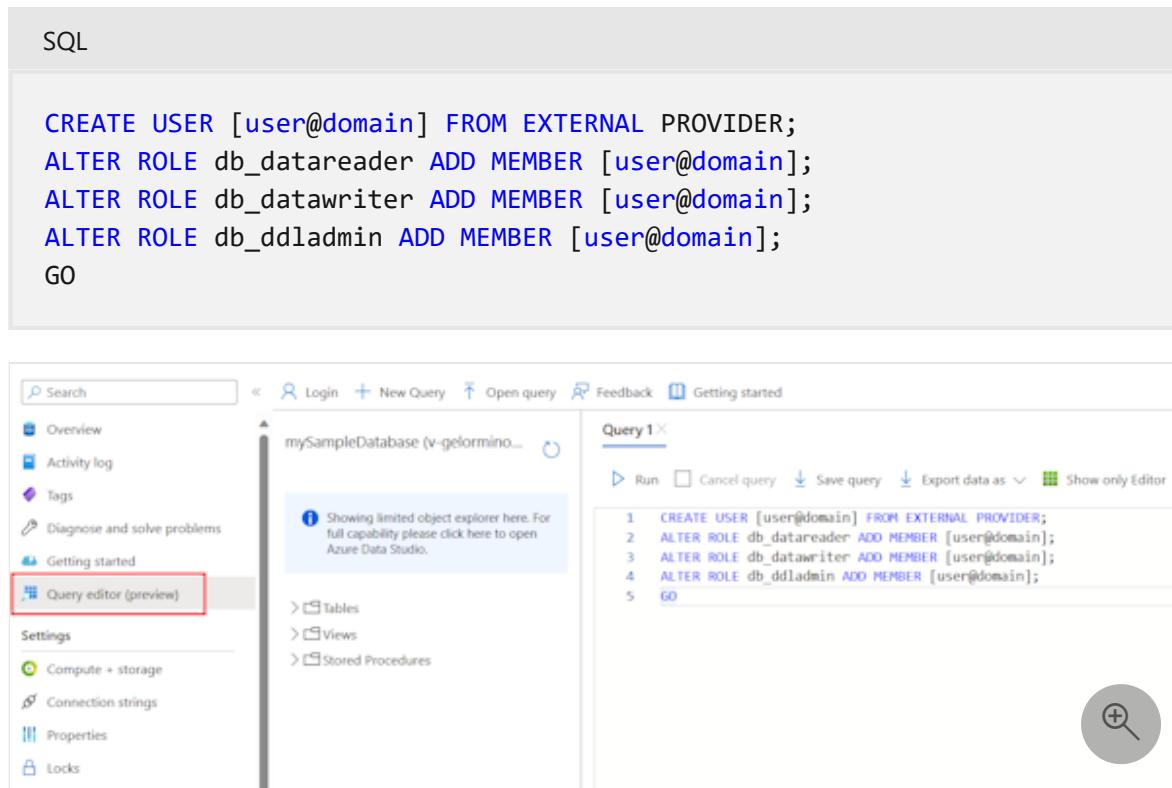
```
az login
```

Créer un utilisateur de base de données et attribuer des rôles

Créez un utilisateur dans Azure SQL Database. L'utilisateur doit correspondre au compte Azure que vous avez utilisé pour vous connecter localement via des outils de

développement comme Visual Studio ou IntelliJ.

1. Dans le [portail Azure](#), accédez à votre base de données SQL et sélectionnez **Éditeur de requête (préversion)**.
2. Sélectionnez **Continuer en tant que <your-username>** sur le côté droit de l'écran pour vous connecter à la base de données à l'aide de votre compte.
3. Dans la vue de l'éditeur de requête, exécutez les commandes T-SQL suivantes :



```
SQL

CREATE USER [user@domain] FROM EXTERNAL PROVIDER;
ALTER ROLE db_datareader ADD MEMBER [user@domain];
ALTER ROLE db_datawriter ADD MEMBER [user@domain];
ALTER ROLE db_ddladmin ADD MEMBER [user@domain];
GO
```

The screenshot shows the Azure portal's Query editor (preview) interface. On the left, there's a sidebar with various options like Overview, Activity log, Tags, and a highlighted 'Query editor (preview)'. The main area has a search bar and navigation links for 'mySampleDatabase (v-gelormino...)' (including Tables, Views, and Stored Procedures), 'Getting started', and a 'Feedback' button. The right side is the 'Query 1' editor window containing the provided T-SQL script. A large circular button with a plus sign is visible in the bottom right corner of the editor area.

L'exécution de ces commandes attribue le rôle [Contributeur de base de données SQL](#) au compte spécifié. Ce rôle permet à l'identité de lire, d'écrire et de modifier les données et le schéma de votre base de données. Pour plus d'informations sur les rôles attribués, consultez [Rôles de base de données fixes](#).

Mettre à jour la configuration de la connexion locale

Le code d'application existant qui se connecte à Azure SQL Database à l'aide de la bibliothèque `Microsoft.Data.SqlClient` ou d'Entity Framework Core continue de fonctionner avec les connexions sans mot de passe. Toutefois, vous devez mettre à jour votre chaîne de connexion de base de données pour utiliser le format sans mot de passe. Le code suivant fonctionne par exemple à la fois avec l'authentification SQL et les connexions sans mot de passe :

C#

```
string connectionString =
app.Configuration.GetConnectionString("AZURE_SQL_CONNECTIONSTRING")!;

using var conn = new SqlConnection(connectionString);
conn.Open();

var command = new SqlCommand("SELECT * FROM Persons", conn);
using SqlDataReader reader = command.ExecuteReader();
```

Pour mettre à jour la chaîne de connexion référencée (`AZURE_SQL_CONNECTIONSTRING`) et utiliser le format de chaîne de connexion sans mot de passe :

1. Recherchez votre chaîne de connexion. Pour le développement local avec des applications .NET, la chaîne est généralement stockée à l'un des emplacements suivants :
 - Fichier de configuration `appsettings.json` pour votre projet.
 - Fichier de configuration `launchsettings.json` pour les projets Visual Studio.
 - Variables d'environnement de conteneur et système local.
2. Remplacez la valeur de chaîne de connexion par le format sans mot de passe suivant. Mettez à jour les espaces réservés `<database-server-name>` et `<database-name>` en les remplaçant par vos valeurs :

JSON

```
Server=tcp:<database-server-name>.database.windows.net,1433;Initial
Catalog=<database-name>;
Encrypt=True;TrustServerCertificate=False;Connection
Timeout=30;Authentication="Active Directory Default";
```

Tester l'application

Exécutez votre application localement et vérifiez que les connexions à Azure SQL Database fonctionnent comme prévu. N'oubliez pas qu'il peut falloir plusieurs minutes pour que les modifications des utilisateurs et des rôles se propagent dans l'environnement Azure. Votre application est désormais configurée pour s'exécuter localement sans que les développeurs ne doivent gérer les secrets dans l'application elle-même.

Configurer l'environnement d'hébergement Azure

Une fois que votre application est configurée pour utiliser des connexions sans mot de passe localement, le même code peut s'authentifier auprès d'Azure SQL Database après son déploiement sur Azure. Les sections suivantes expliquent comment configurer une application déployée pour se connecter à Azure SQL Database à l'aide d'une [identité managée](#). Les identités managées fournissent une identité managée automatiquement dans Microsoft Entra ID ([anciennement Azure Active Directory](#)) que les applications peuvent utiliser lors de la connexion à des ressources qui prennent en charge l'authentification Microsoft Entra. En savoir plus sur les identités managées :

- [Vue d'ensemble des connexions sans mot de passe](#)
- [Meilleures pratiques d'identité managée](#)

Créer l'identité managée

Créez une identité managée affectée par l'utilisateur à l'aide du portail Azure ou de l'interface Azure CLI. Votre application utilise l'identité pour s'authentifier auprès d'autres services.

Azure portal

1. À partir du Portail Azure, recherchez *Identités managées*. Sélectionnez le résultat **Identités managées**.
2. Sélectionnez + **Créer** en haut de la page de présentation des **Identités managées**.
3. Sous l'onglet **Informations de base**, entrez les valeurs suivantes :
 - **Abonnement** : sélectionnez l'option souhaitée.
 - **Groupe de ressources** : sélectionnez votre groupe de ressources souhaité.
 - **Région** : sélectionnez une région proche de votre emplacement.
 - **Nom** : entrez un nom reconnaissable pour votre identité, par exemple *MigrationIdentity*.
4. Au bas de la page, sélectionnez **Examiner et créer**.
5. Une fois les vérifications de validation terminées, sélectionnez **Créer**. Azure crée une identité affectée par l'utilisateur.

Une fois la ressource créée, sélectionnez **Accéder à la ressource** pour afficher les détails de l'identité managée.

Create User Assigned Managed Identity

Basics Tags Review + create

Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * ⓘ

Test Subscription

Resource group * ⓘ

passwordlesstesting

[Create new](#)

Instance details

Region * ⓘ

South Central US

Name * ⓘ

MigrationIdentity



[Review + create](#)

[< Previous](#)

[Next : Tags >](#)

Associer l'identité managée à votre application web

Configurez votre application web pour utiliser l'identité managée affectée par l'utilisateur créée.

Azure portal

Procédez comme suit dans le portail Azure pour associer une identité managée affectée par l'utilisateur à votre application. Ces mêmes étapes s'appliquent aux services Azure suivants :

- Azure Spring Apps
- Azure Container Apps
- Machines virtuelles Azure
- Azure Kubernetes Service
- Accédez à la page de présentation de votre application web front-end.

1. Sélectionnez Identité dans la barre de navigation de gauche.

2. Dans la page **Identité**, basculez vers l'onglet **Utilisateur affecté**.
3. Sélectionnez **+ Ajouter** pour ouvrir le menu volant **Ajouter une identité managée affectée par l'utilisateur**.
4. Sélectionnez l'abonnement utilisé précédemment pour créer l'identité.
5. Recherchez **MigrationIdentity** par nom et sélectionnez-le dans les résultats de la recherche.
6. Sélectionnez **Ajouter** pour associer l'identité à votre application.

The screenshot shows the Azure portal interface for managing user-assigned managed identities. The left sidebar has 'Identity' selected. The main area shows a list of identities under 'User assigned'. A red box highlights the '+ Add' button. To the right, a 'Selected identities' section shows 'passwordlessuser' selected, with a 'Remove' link. A search bar and an 'Add' button are also visible.

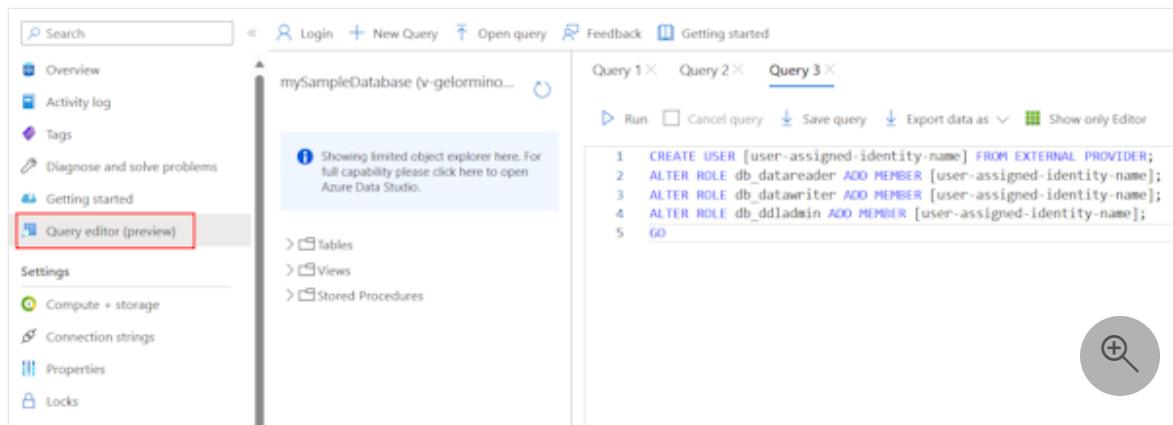
Créer un utilisateur de base de données pour l'identité et attribuer des rôles

Créez un utilisateur de base de données SQL mappé à l'identité managée affectée par l'utilisateur. Attribuez également les rôles SQL nécessaires à l'utilisateur pour autoriser votre application à lire, écrire et modifier les données et le schéma de votre base de données.

1. Dans le Portail Azure, accédez à votre base de données SQL et sélectionnez **Éditeur de requête (préversion)**.
2. Sélectionnez **Continuer en tant que <username>** sur le côté droit de l'écran pour vous connecter à la base de données à l'aide de votre compte.
3. Dans la vue de l'éditeur de requête, exécutez les commandes T-SQL suivantes :

SQL

```
CREATE USER [user-assigned-identity-name] FROM EXTERNAL PROVIDER;
ALTER ROLE db_datareader ADD MEMBER [user-assigned-identity-name];
ALTER ROLE db_datawriter ADD MEMBER [user-assigned-identity-name];
ALTER ROLE db_ddladmin ADD MEMBER [user-assigned-identity-name];
GO
```



L'exécution de ces commandes attribue le rôle [Contributeur de base de données SQL](#) à l'identité managée affectée par l'utilisateur. Ce rôle permet à l'identité de lire, d'écrire et de modifier les données et le schéma de votre base de données.

ⓘ Important

Faites preuve de précaution quand vous attribuez des rôles d'utilisateur de base de données dans des environnements de production d'entreprise. Dans ces scénarios, l'application ne doit pas effectuer toutes les opérations en utilisant une même identité avec priviléges élevés. Essayez d'implémenter le principe des priviléges minimum en configurant plusieurs identités disposant d'autorisations spécifiques pour des tâches spécifiques.

Vous pouvez en apprendre plus sur la configuration des rôles de base de données et la sécurité avec les ressources suivantes :

- [Tutoriel : Sécuriser une base de données dans Azure SQL Database](#)
- [Autoriser l'accès base de données à SQL Database](#)

Mettre à jour la chaîne de connexion

Mettez à jour la configuration de votre application Azure pour utiliser le format de chaîne de connexion sans mot de passe. Les chaînes de connexion sont généralement stockées en tant que variables d'environnement dans votre environnement

d'hébergement d'applications. Les instructions suivantes se concentrent sur App Service, mais d'autres services d'hébergement Azure fournissent des configurations similaires.

1. Accédez à la page de configuration de votre instance App Service et recherchez la chaîne de connexion Azure SQL Database.
2. Sélectionnez l'icône de modification et mettez à jour la valeur de la chaîne de connexion pour qu'elle corresponde au format suivant. Modifiez les espaces réservés <database-server-name> et <database-name> en les remplaçant par les valeurs de votre propre service.

JSON

```
Server=tcp:<database-server-name>.database.windows.net,1433;Initial  
Catalog=<database-name>;  
Encrypt=True;TrustServerCertificate=False;Connection  
Timeout=30;Authentication="Active Directory Default";
```

3. Enregistrez vos modifications et redémarrez l'application si le redémarrage n'est pas automatique.

Tester l'application

Testez votre application pour vous assurer de son bon fonctionnement. La propagation de l'ensemble des modifications dans votre environnement Azure peut prendre quelques minutes.

Étapes suivantes

Dans ce didacticiel, vous avez appris à migrer une application vers des connexions sans mot de passe.

Vous pouvez lire les ressources suivantes pour explorer les concepts abordés dans cet article plus en détails :

- [Vue d'ensemble des connexions sans mot de passe](#)
- [Meilleures pratiques d'identité managée](#)
- [Tutoriel : Sécuriser une base de données dans Azure SQL Database](#)
- [Autoriser l'accès base de données à SQL Database](#)

Migrer une application Java pour utiliser des connexions sans mot de passe avec Azure SQL Database

Article • 23/10/2023

Cet article explique comment migrer des méthodes d'authentification traditionnelles vers des connexions sans mot de passe plus sécurisées avec Azure SQL Database.

Les requêtes des applications adressées à Azure SQL Database doivent être authentifiées. Azure SQL Database fournit plusieurs façons différentes pour les applications de se connecter en toute sécurité. L'une des façons d'utiliser des mots de passe. Toutefois, vous devez hiérarchiser les connexions sans mot de passe dans vos applications lorsque cela est possible.

Comparer les options d'authentification

Lorsque l'application s'authentifie auprès d'Azure SQL Database, elle fournit une paire de nom d'utilisateur et de mot de passe pour se connecter à la base de données. Selon l'emplacement de stockage des identités, il existe deux types d'authentification : l'authentification Microsoft Entra et l'authentification Azure SQL Database.

Authentification Microsoft Entra

L'authentification Microsoft Entra est un mécanisme de connexion à Azure SQL Database à l'aide d'identités définies dans l'ID Microsoft Entra. Avec l'authentification Microsoft Entra, vous pouvez gérer les identités des utilisateurs de base de données et d'autres services Microsoft dans un emplacement centralisé, ce qui simplifie la gestion des autorisations.

L'utilisation de l'ID Microsoft Entra pour l'authentification offre les avantages suivants :

- Authentification des utilisateurs dans les services Azure de manière uniforme.
- Gestion des stratégies de mot de passe et de la rotation des mots de passe à un seul endroit.
- Plusieurs formes d'authentification prises en charge par l'ID Microsoft Entra, qui peuvent éliminer la nécessité de stocker les mots de passe.
- Les clients peuvent gérer les autorisations de base de données à l'aide de groupes externes (Microsoft Entra ID).

- L'authentification Microsoft Entra utilise les utilisateurs de base de données Azure SQL pour authentifier les identités au niveau de la base de données.
- Prise en charge de l'authentification basée sur des jetons pour les applications qui se connectent à Azure SQL Database.

Authentification Azure SQL Database

Vous pouvez créer des comptes dans Azure SQL Database. Si vous choisissez d'utiliser des mots de passe comme informations d'identification pour les comptes, ces informations d'identification sont stockées dans la table `sys.database_principals`. Étant donné que ces mots de passe sont stockés dans Azure SQL Database, vous devez gérer la rotation des mots de passe par vous-même.

Bien qu'il soit possible de se connecter à Azure SQL Database avec des mots de passe, vous devez les utiliser avec précaution. Vous devez être vigilant pour ne jamais exposer les mots de passe dans un emplacement non sécurisé. Toute personne qui accède aux mots de passe est en mesure de s'authentifier. Par exemple, il existe un risque qu'un utilisateur malveillant puisse accéder à l'application si un chaîne de connexion est accidentellement case activée dans le contrôle de code source, envoyé via un e-mail non sécurisé, collé dans une conversation incorrecte ou consulté par une personne qui ne doit pas avoir d'autorisation. Au lieu de cela, envisagez de mettre à jour votre application pour utiliser des connexions sans mot de passe.

Présentation des connexions sans mot de passe

Avec une connexion sans mot de passe, vous pouvez vous connecter aux services Azure sans stocker d'informations d'identification dans le code de l'application, ses fichiers de configuration ou dans les variables d'environnement.

De nombreux services Azure prennent en charge les connexions sans mot de passe, par exemple via Azure Managed Identity. Ces techniques fournissent des fonctionnalités de sécurité robustes que vous pouvez implémenter à l'aide [de DefaultAzureCredential](#) à partir des bibliothèques clientes Azure Identity. Dans ce tutoriel, vous allez apprendre à mettre à jour une application existante à utiliser `DefaultAzureCredential` au lieu d'alternatives telles que des chaîne de connexion.

`DefaultAzureCredential` prend en charge plusieurs méthodes d'authentification et détermine automatiquement celle qui doit être utilisée au moment de l'exécution. Cette approche permet à votre application d'utiliser différentes méthodes d'authentification dans différents environnements (développement local et production) sans implémenter de code spécifique à l'environnement.

L'ordre et les emplacements dans lesquels `DefaultAzureCredential` les recherches d'informations d'identification sont disponibles dans la vue d'ensemble [de la bibliothèque d'identités Azure](#). Par exemple, lorsque vous travaillez localement, `DefaultAzureCredential` s'authentifie généralement à l'aide du compte utilisé par le développeur pour se connecter à Visual Studio. Lorsque l'application est déployée sur Azure, `DefaultAzureCredential` bascule automatiquement pour utiliser une [identité managée](#). Aucune modification du code n'est requise pour cette transition.

Pour vous assurer que les connexions sont sans mot de passe, vous devez prendre en compte le développement local et l'environnement de production. Si une chaîne de connexion est requise à l'un ou l'autre endroit, l'application n'est pas sans mot de passe.

Dans votre environnement de développement local, vous pouvez vous authentifier auprès d'Azure CLI, d'Azure PowerShell, de Visual Studio ou de plug-ins Azure pour Visual Studio Code ou IntelliJ. Dans ce cas, vous pouvez utiliser ces informations d'identification dans votre application au lieu de configurer des propriétés.

Lorsque vous déployez des applications dans un environnement d'hébergement Azure, tel qu'une machine virtuelle, vous pouvez affecter une identité managée dans cet environnement. Ensuite, vous n'avez pas besoin de fournir des informations d'identification pour vous connecter aux services Azure.

ⓘ Notes

Une identité managée fournit une identité de sécurité pour représenter une application ou un service. Managée par la plateforme Azure, l'identité ne nécessite pas que vous approvisionniez ou permutiez de secrets. Vous pouvez en savoir plus sur les identités managées dans la documentation [vue d'ensemble](#).

ⓘ Notes

Étant donné que le pilote JDBC pour Azure SQL Database ne prend pas encore en charge les connexions sans mot de passe à partir d'environnements locaux, cet article se concentre uniquement sur les applications déployées dans des environnements d'hébergement Azure et sur la façon de les migrer pour utiliser des connexions sans mot de passe.

Migrer une application existante pour utiliser des connexions sans mot de passe

Les étapes suivantes expliquent comment migrer une application existante pour utiliser des connexions sans mot de passe au lieu d'une solution basée sur un mot de passe.

0) Préparer l'environnement de travail

Tout d'abord, utilisez la commande suivante pour configurer certaines variables d'environnement.

Bash

```
export AZ_RESOURCE_GROUP=<YOUR_RESOURCE_GROUP>
export AZ_DATABASE_SERVER_NAME=<YOUR_DATABASE_SERVER_NAME>
export AZ_DATABASE_NAME=demo
export CURRENT_USERNAME=$(az ad signed-in-user show --query
userPrincipalName --output tsv)
export CURRENT_USER_OBJECTID=$(az ad signed-in-user show --query id --output
tsv)
```

Remplacez les espaces réservés par les valeurs suivantes, qui sont utilisées dans cet article :

- <YOUR_RESOURCE_GROUP> : nom du groupe de ressources dans lequel se trouvent vos ressources.
- <YOUR_DATABASE_SERVER_NAME> : Nom de votre serveur Azure SQL Database. Il doit être unique dans tout Azure.

1) Configurer Azure SQL Database

1.1) Activer l'authentification basée sur l'ID Microsoft Entra

Pour utiliser l'accès à l'ID Microsoft Entra avec Azure SQL Database, vous devez d'abord définir l'utilisateur administrateur Microsoft Entra. Seul un utilisateur Microsoft Entra Administration peut créer/activer des utilisateurs pour l'authentification basée sur l'ID Microsoft Entra.

Si vous utilisez Azure CLI, exécutez la commande suivante pour vous assurer de disposer d'une autorisation suffisante :

Bash

```
az login --scope https://graph.microsoft.com/.default
```

Exécutez ensuite la commande suivante pour définir l'administrateur Microsoft Entra :

```
az sql server ad-admin create \
--resource-group $AZ_RESOURCE_GROUP \
--server $AZ_DATABASE_SERVER_NAME \
--display-name $CURRENT_USERNAME \
--object-id $CURRENT_USER_OBJECTID
```

Cette commande définit l'administrateur Microsoft Entra sur l'utilisateur connecté actuel.

ⓘ Notes

Vous ne pouvez créer qu'un seul administrateur Microsoft Entra par serveur Azure SQL Database. La sélection d'un autre remplacera l'administrateur Microsoft Entra existant configuré pour le serveur.

2) Migrer le code de l'application pour utiliser des connexions sans mot de passe

Ensuite, procédez comme suit pour mettre à jour votre code pour utiliser des connexions sans mot de passe. Bien que conceptuellement similaire, chaque langage utilise des détails d'implémentation différents.

Java

1. Dans votre projet, ajoutez la référence suivante au `azure-identity` package.

Cette bibliothèque contient toutes les entités nécessaires pour implémenter des connexions sans mot de passe.

XML

```
<dependency>
    <groupId>com.azure</groupId>
    <artifactId>azure-identity</artifactId>
    <version>1.5.4</version>
</dependency>
```

2. Activez l'authentification d'identité managée Microsoft Entra dans JDBC URL. Identifier les emplacements de votre code qui créent actuellement une `java.sql.Connection` connexion à Azure SQL Database. Mettez à jour votre code pour le faire correspondre à l'exemple suivant :

Java

```
String url =  
    "jdbc:sqlserver://$AZ_DATABASE_SERVER_NAME.database.windows.net:143  
3;databaseName=$AZ_DATABASE_NAME;authentication=ActiveDirectoryMSI;  
"  
Connection con = DriverManager.getConnection(url);
```

3. Remplacez les deux `$AZ_DATABASE_SERVER_NAME` variables et une `$AZ_DATABASE_NAME` variable par les valeurs que vous avez configurées au début de cet article.
4. Supprimez l'URL `user` JDBC et `password` la supprimez.

3) Configurer l'environnement d'hébergement Azure

Une fois que votre application est configurée pour utiliser des connexions sans mot de passe, le même code peut s'authentifier auprès des services Azure après son déploiement sur Azure. Par exemple, une application déployée sur une instance Azure App Service qui a une identité managée affectée peut se connecter à Stockage Azure.

Dans cette section, vous allez exécuter deux étapes pour permettre à votre application de s'exécuter dans un environnement d'hébergement Azure de manière sans mot de passe :

- Attribuez l'identité managée pour votre environnement d'hébergement Azure.
- Attribuez des rôles à l'identité managée.

ⓘ Notes

Azure fournit également un **Connecter de service**, qui peut vous aider à connecter votre service d'hébergement avec SQL Server. Avec service Connecter or pour configurer votre environnement d'hébergement, vous pouvez omettre l'étape d'attribution de rôles à votre identité managée, car Le Connecter or de service le fera pour vous. La section suivante explique comment configurer votre environnement d'hébergement Azure de deux façons : l'une via service Connecter or et l'autre en configurant directement chaque environnement d'hébergement.

ⓘ Important

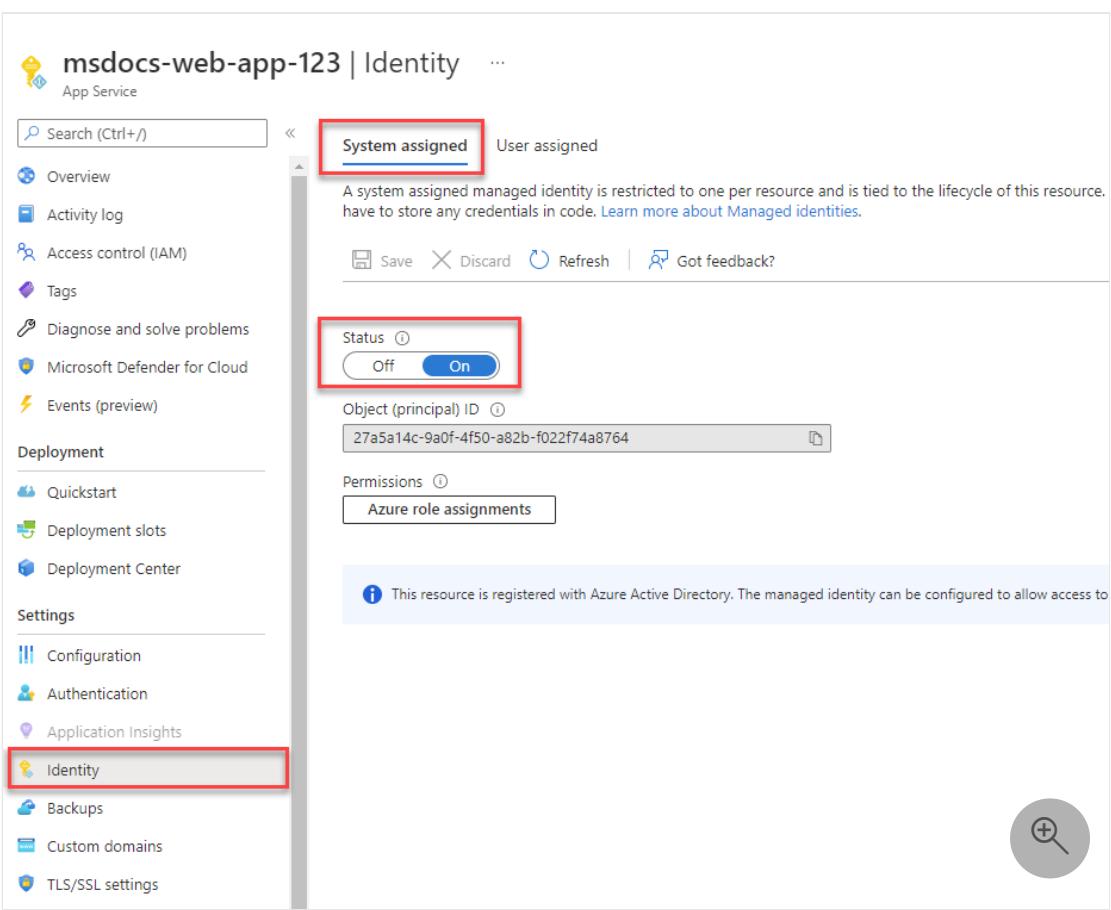
Les commandes du Connecter or de service nécessitent Azure CLI 2.41.0 ou version ultérieure.

Attribuer l'identité managée à l'aide du Portail Azure

Les étapes suivantes vous montrent comment attribuer une identité managée affectée par le système pour différents services d'hébergement web. L'identité managée peut se connecter en toute sécurité à d'autres services Azure à l'aide des configurations d'application que vous avez configurées précédemment.

App Service

1. Dans la page de vue d'ensemble principale de votre instance Azure App Service, sélectionnez **Identité** dans le volet de navigation.
2. Sous l'onglet Affecté par le système, veillez à définir le champ **État** sur activé. Une identité affectée par le système est gérée par Azure en interne et gère les tâches d'administration pour vous. Les détails et ID de l'identité ne sont jamais exposés dans votre code.



Vous pouvez également affecter une identité managée sur un environnement d'hébergement Azure à l'aide d'Azure CLI.

App Service

Vous pouvez affecter une identité managée à une instance Azure App Service avec la [commande az webapp identity assign](#), comme illustré dans l'exemple suivant :

Azure CLI

```
export AZ_MI_OBJECT_ID=$(az webapp identity assign \
    --resource-group $AZ_RESOURCE_GROUP \
    --name <service-instance-name> \
    --query principalId \
    --output tsv)
```

Attribuer des rôles à l'identité managée

Ensuite, accordez des autorisations à l'identité managée que vous avez créée pour accéder à votre base de données SQL.

Connecteur de service

Si vous avez connecté vos services à l'aide du service Connecter or, les commandes de l'étape précédente ont déjà attribué le rôle, afin de pouvoir ignorer cette étape.

Test de l'application

Après avoir apporté ces modifications de code, vous pouvez générer et redéployer l'application. Ensuite, accédez à votre application hébergée dans le navigateur. Votre application doit être en mesure de se connecter à la base de données Azure SQL avec succès. N'oubliez pas qu'il peut falloir plusieurs minutes pour que les attributions de rôle se propagent dans l'environnement Azure. Votre application est désormais configurée pour s'exécuter localement et dans un environnement de production sans que les développeurs n'aient à gérer les secrets dans l'application elle-même.

Étapes suivantes

Dans ce didacticiel, vous avez appris à migrer une application vers des connexions sans mot de passe.

Vous pouvez lire les ressources suivantes pour explorer les concepts abordés dans cet article plus en détails :

- [Autoriser l'accès aux données d'objet blob avec des identités managées pour les ressources Azure](#).
- [Autoriser l'accès aux objets blob à l'aide Microsoft Entra ID](#)

Migrer une application Node.js pour utiliser des connexions sans mot de passe avec Azure SQL Database

Article • 29/12/2023

S'applique à  [Azure SQL Database](#)

Les requêtes des applications adressées à Azure SQL Database doivent être authentifiées. Bien qu'il existe plusieurs options pour l'authentification auprès d'Azure SQL Database, vous devez donner la priorité aux connexions sans mot de passe dans vos applications quand cela est possible. Les méthodes d'authentification traditionnelles qui utilisent des mots de passe ou des clés secrètes créent des risques et des complications de sécurité. Visitez le hub de [connexions sans mot de passe pour Azure Services](#) pour découvrir l'avantage des connexions sans mot de passe.

Le tutoriel suivant explique comment migrer une application Node.js existante pour se connecter à Azure SQL Database afin d'utiliser des connexions sans mot de passe au lieu d'une solution avec nom d'utilisateur et mot de passe.

Configurer Azure SQL Database

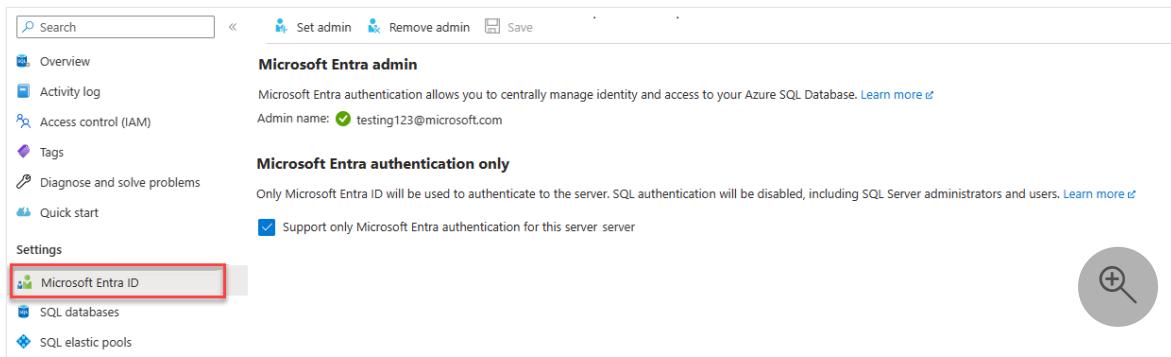
Les connexions sans mot de passe utilisent l'authentification Microsoft Entra pour se connecter aux services Azure, y compris la base de données Azure SQL. L'authentification Microsoft Entra facilite la gestion des identités dans une position centrale et simplifie la gestion des autorisations. Découvrez comment configurer l'authentification Microsoft Entra pour votre base de données Azure SQL :

- [Vue d'ensemble de l'authentification Microsoft Entra](#)
- [Configurer l'authentification Microsoft Entra](#)

Pour le présent guide de migration, assurez-vous qu'un administrateur Microsoft Entra est affecté à votre base de données Azure SQL.

1. Accédez à la page **Microsoft Entra** de votre serveur logique.
2. Sélectionnez **Définir l'administrateur** pour ouvrir le menu à contrôle suspendu de **Microsoft Entra ID**.
3. Dans le menu à contrôle suspendu de **Microsoft Entra ID**, recherchez l'utilisateur auquel vous souhaitez attribuer le rôle d'administrateur.

4. Sélectionnez l'utilisateur et choisissez Sélectionner.



The screenshot shows the Microsoft Entra admin center interface. At the top, there's a search bar and navigation links for Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Quick start, Set admin, Remove admin, and Save. Below these, the 'Microsoft Entra admin' section is shown with the sub-section 'Microsoft Entra authentication only'. It states that only Microsoft Entra ID will be used for authentication, and SQL authentication will be disabled. A checkbox labeled 'Support only Microsoft Entra authentication for this server' is checked. In the 'Settings' sidebar, the 'Microsoft Entra ID' option is highlighted with a red box. Other options listed are SQL databases and SQL elastic pools. A magnifying glass icon is in the bottom right corner.

Configurer votre environnement de développement local

Les connexions sans mot de passe peuvent être configurées pour fonctionner dans les environnements locaux et hébergés par Azure. Dans cette section, vous appliquez des configurations afin de permettre aux utilisateurs individuels de s'authentifier auprès d'Azure SQL Database pour le développement local.

Se connecter à Azure

Pour le développement local, assurez-vous que vous êtes connecté avec le même compte Azure AD que celui que vous souhaitez utiliser pour accéder à Azure SQL Database. Vous pouvez vous authentifier au moyen d'outils de développement populaires, comme Azure CLI ou Azure PowerShell. Les outils de développement avec lesquels vous pouvez vous authentifier dépendent de la langue.

Azure CLI

Connectez-vous à Azure via Azure CLI à l'aide de la commande suivante :

```
Azure CLI
```

```
az login
```

Créer un utilisateur de base de données et attribuer des rôles

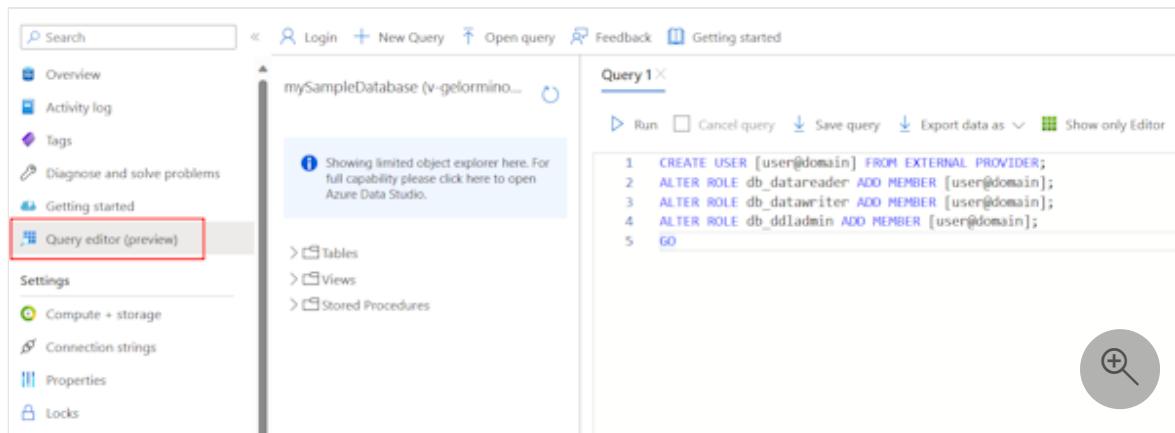
Créez un utilisateur dans Azure SQL Database. L'utilisateur doit correspondre au compte Azure que vous avez utilisé pour vous connecter localement dans la section [Se connecter à Azure](#).

connecter à Azure.

1. Dans le [portail Azure](#), accédez à votre base de données SQL et sélectionnez **Éditeur de requête (préversion)**.
2. Sélectionnez **Continuer en tant que <your-username>** sur le côté droit de l'écran pour vous connecter à la base de données à l'aide de votre compte.
3. Dans la vue de l'éditeur de requête, exécutez les commandes T-SQL suivantes :

```
SQL

CREATE USER [user@domain] FROM EXTERNAL PROVIDER;
ALTER ROLE db_datareader ADD MEMBER [user@domain];
ALTER ROLE db_datawriter ADD MEMBER [user@domain];
ALTER ROLE db_ddladmin ADD MEMBER [user@domain];
GO
```



L'exécution de ces commandes attribue le rôle [Contributeur de base de données SQL](#) au compte spécifié. Ce rôle permet à l'identité de lire, d'écrire et de modifier les données et le schéma de votre base de données. Pour plus d'informations sur les rôles attribués, consultez [Rôles de base de données fixes](#).

Mettre à jour la configuration de la connexion locale

1. Créez des paramètres d'environnement pour votre application.

```
ini

AZURE_SQL_SERVER=<YOURSERVERNAME>.database.windows.net
AZURE_SQL_DATABASE=<YOURDATABASENAME>
AZURE_SQL_PORT=1433
```

2. Le code de l'application existante qui se connecte à Azure SQL Database en utilisant le [pilote SQL Node.js - tedious](#) continue de fonctionner avec des connexions sans mot de passe moyennant des modifications mineures. Pour utiliser une identité managée affectée par l'utilisateur, passez les propriétés `authentication.type` et `options.clientId`.

Node.js

```
import sql from 'mssql';

// Environment settings - no user or password
const server = process.env.AZURE_SQL_SERVER;
const database = process.env.AZURE_SQL_DATABASE;
const port = parseInt(process.env.AZURE_SQL_PORT);
const clientId = process.env.AZURE_CLIENT_ID;

// Passwordless configuration
const config = {
    server,
    port,
    database,
    authentication: {
        type: 'azure-active-directory-default',
    },
    options: {
        encrypt: true,
        clientId: process.env.AZURE_CLIENT_ID // <----- user-assigned
managed identity
    }
};

// Existing application code
export default class Database {
    config = {};
    poolconnection = null;
    connected = false;

    constructor(config) {
        this.config = config;
        console.log(`Database: config: ${JSON.stringify(config)}`);
    }

    async connect() {
        try {
            console.log(`Database connecting...${this.connected}`);
            if (this.connected === false) {
                this.poolconnection = await sql.connect(this.config);
                this.connected = true;
                console.log('Database connection successful');
            } else {
                console.log('Database already connected');
            }
        }
    }
}
```

```

        } catch (error) {
            console.error(`Error connecting to database:
${JSON.stringify(error)} `);
        }
    }

    async disconnect() {
        try {
            this.poolconnection.close();
            console.log('Database connection closed');
        } catch (error) {
            console.error(`Error closing database connection:
${error} `);
        }
    }

    async executeQuery(query) {
        await this.connect();
        const request = this.poolconnection.request();
        const result = await request.query(query);

        return result.rowsAffected[0];
    }
}

const database = new Database(config);
const result = await database.executeQuery(`select * from mytable where
id = 10`);
```

La variable d'environnement `AZURE_CLIENT_ID` est créée plus loin dans ce tutoriel.

Tester l'application

Exécutez votre application localement et vérifiez que les connexions à Azure SQL Database fonctionnent comme prévu. N'oubliez pas qu'il peut falloir plusieurs minutes pour que les modifications des utilisateurs et des rôles se propagent dans l'environnement Azure. Votre application est désormais configurée pour s'exécuter localement sans que les développeurs ne doivent gérer les secrets dans l'application elle-même.

Configurer l'environnement d'hébergement Azure

Une fois que votre application est configurée pour utiliser des connexions sans mot de passe localement, le même code peut s'authentifier auprès d'Azure SQL Database après son déploiement sur Azure. Les sections suivantes expliquent comment configurer une

application déployée pour se connecter à Azure SQL Database à l'aide d'une [identité managée](#). Les identités managées fournissent une identité managée automatiquement dans Microsoft Entra ID ([anciennement Azure Active Directory](#)) que les applications peuvent utiliser lors de la connexion à des ressources qui prennent en charge l'authentification Microsoft Entra. En savoir plus sur les identités managées :

- [Vue d'ensemble des connexions sans mot de passe](#)
- [Meilleures pratiques d'identité managée](#)
- [Identités managées dans Microsoft Entra pour Azure SQL](#)

Créer l'identité managée

Créez une identité managée affectée par l'utilisateur à l'aide du portail Azure ou de l'interface Azure CLI. Votre application utilise l'identité pour s'authentifier auprès d'autres services.

Azure portal

1. À partir du Portail Azure, recherchez *Identités managées*. Sélectionnez le résultat **Identités managées**.
2. Sélectionnez + **Créer** en haut de la page de présentation des **Identités managées**.
3. Sous l'onglet **Informations de base**, entrez les valeurs suivantes :
 - **Abonnement** : sélectionnez l'option souhaitée.
 - **Groupe de ressources** : sélectionnez votre groupe de ressources souhaité.
 - **Région** : sélectionnez une région proche de votre emplacement.
 - **Nom** : entrez un nom reconnaissable pour votre identité, par exemple *MigrationIdentity*.
4. Au bas de la page, sélectionnez **Examiner et créer**.
5. Une fois les vérifications de validation terminées, sélectionnez **Créer**. Azure crée une identité affectée par l'utilisateur.

Une fois la ressource créée, sélectionnez **Accéder à la ressource** pour afficher les détails de l'identité managée.

Create User Assigned Managed Identity

Basics Tags Review + create

Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * ⓘ

Test Subscription

Resource group * ⓘ

passwordlesstesting

[Create new](#)

Instance details

Region * ⓘ

South Central US

Name * ⓘ

MigrationIdentity



[Review + create](#)

[< Previous](#)

[Next : Tags >](#)

Associer l'identité managée à votre application web

Configurez votre application web pour utiliser l'identité managée affectée par l'utilisateur créée.

Azure portal

Procédez comme suit dans le portail Azure pour associer une identité managée affectée par l'utilisateur à votre application. Ces mêmes étapes s'appliquent aux services Azure suivants :

- Azure Spring Apps
- Azure Container Apps
- Machines virtuelles Azure
- Azure Kubernetes Service
- Accédez à la page de présentation de votre application web front-end.

1. Sélectionnez Identité dans la barre de navigation de gauche.

2. Dans la page **Identité**, basculez vers l'onglet **Utilisateur affecté**.
3. Sélectionnez **+ Ajouter** pour ouvrir le menu volant **Ajouter une identité managée affectée par l'utilisateur**.
4. Sélectionnez l'abonnement utilisé précédemment pour créer l'identité.
5. Recherchez **MigrationIdentity** par nom et sélectionnez-le dans les résultats de la recherche.
6. Sélectionnez **Ajouter** pour associer l'identité à votre application.

The screenshot shows the Azure portal interface for managing user-assigned managed identities. The left sidebar has 'Identity' selected. The main area shows a list of identities under 'User assigned'. A red box highlights the '+ Add' button. To the right, there's a 'Selected identities' section with a single identity named 'passwordlessuser' checked. A search bar and an 'Add' button are also visible.

Créer un utilisateur de base de données pour l'identité et attribuer des rôles

Créez un utilisateur de base de données SQL mappé à l'identité managée affectée par l'utilisateur. Attribuez également les rôles SQL nécessaires à l'utilisateur pour autoriser votre application à lire, écrire et modifier les données et le schéma de votre base de données.

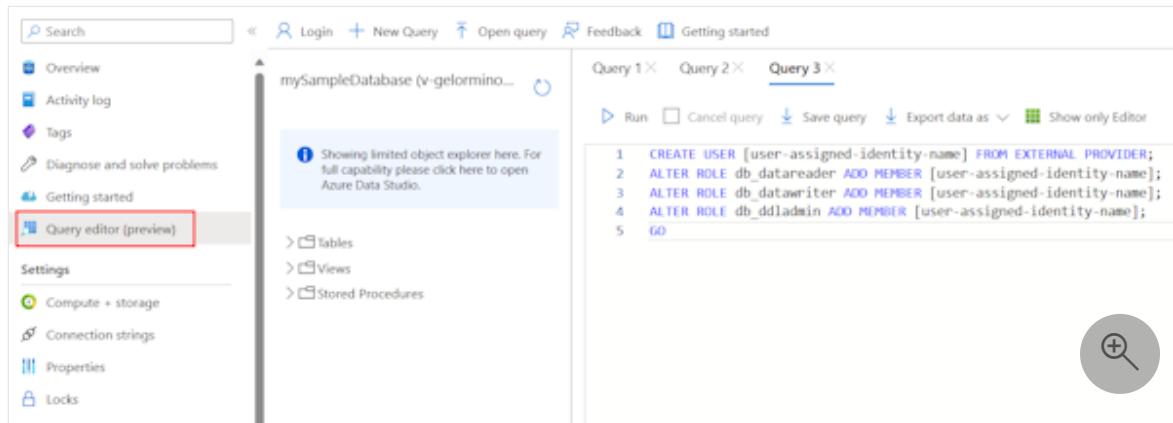
1. Dans le Portail Azure, accédez à votre base de données SQL et sélectionnez **Éditeur de requête (préversion)**.
2. Sélectionnez **Continuer en tant que <username>** sur le côté droit de l'écran pour vous connecter à la base de données à l'aide de votre compte.
3. Dans la vue de l'éditeur de requête, exécutez les commandes T-SQL suivantes :

SQL

```

CREATE USER [user-assigned-identity-name] FROM EXTERNAL PROVIDER;
ALTER ROLE db_datareader ADD MEMBER [user-assigned-identity-name];
ALTER ROLE db_datawriter ADD MEMBER [user-assigned-identity-name];
ALTER ROLE db_ddladmin ADD MEMBER [user-assigned-identity-name];
GO

```



L'exécution de ces commandes attribue le rôle [Contributeur de base de données SQL](#) à l'identité managée affectée par l'utilisateur. Ce rôle permet à l'identité de lire, d'écrire et de modifier les données et le schéma de votre base de données.

ⓘ Important

Faites preuve de précaution quand vous attribuez des rôles d'utilisateur de base de données dans des environnements de production d'entreprise. Dans ces scénarios, l'application ne doit pas effectuer toutes les opérations en utilisant une même identité avec priviléges élevés. Essayez d'implémenter le principe des priviléges minimum en configurant plusieurs identités disposant d'autorisations spécifiques pour des tâches spécifiques.

Vous pouvez en apprendre plus sur la configuration des rôles de base de données et la sécurité avec les ressources suivantes :

- [Tutoriel : Sécuriser une base de données dans Azure SQL Database](#)
- [Autoriser l'accès base de données à SQL Database](#)

Créer un paramètre d'application pour l'ID client de l'identité managée

Pour utiliser l'identité managée [affectée par l'utilisateur](#), créez une variable d'environnement `AZURE_CLIENT_ID` et définissez-la sur l'ID client de l'identité managée. Vous pouvez définir cette variable dans la section **Configuration** de votre application

dans le portail Azure. Vous trouverez l'ID client dans la section **Vue d'ensemble** de la ressource d'identité managée dans le portail Azure.

Enregistrez vos modifications et redémarrez l'application si elle ne redémarre pas automatiquement.

Si vous devez utiliser une identité managée **affectée par le système**, omettez la propriété `options.clientId`. Vous devez quand même passer la propriété `authentication.type`.

```
Node.js

const config = {
  server,
  port,
  database,
  authentication: {
    type: 'azure-active-directory-default'
  },
  options: {
    encrypt: true
  }
};
```

Tester l'application

Testez votre application pour vous assurer de son bon fonctionnement. La propagation de l'ensemble des modifications dans votre environnement Azure peut prendre quelques minutes.

Étapes suivantes

Dans ce didacticiel, vous avez appris à migrer une application vers des connexions sans mot de passe.

Vous pouvez lire les ressources suivantes pour explorer les concepts abordés dans cet article plus en détails :

- [Vue d'ensemble des connexions sans mot de passe](#)
- [Meilleures pratiques d'identité managée](#)
- [Tutoriel : Sécuriser une base de données dans Azure SQL Database](#)
- [Autoriser l'accès base de données à SQL Database](#)

Migrer une application Python pour utiliser des connexions sans mot de passe avec Azure SQL Database

Article • 11/10/2023

S'applique à  [Azure SQL Database](#)

Les requêtes des applications adressées à Azure SQL Database doivent être authentifiées. Bien qu'il existe plusieurs options pour l'authentification auprès d'Azure SQL Database, vous devez donner la priorité aux connexions sans mot de passe dans vos applications quand cela est possible. Les méthodes d'authentification traditionnelles qui utilisent des mots de passe ou des clés secrètes créent des risques et des complications de sécurité. Visitez le hub de [connexions sans mot de passe pour Azure Services](#) pour découvrir l'avantage des connexions sans mot de passe. Le tutoriel suivant explique comment migrer une application Python existante pour se connecter à Azure SQL Database afin d'utiliser des connexions sans mot de passe au lieu d'une solution avec nom d'utilisateur et mot de passe.

Configurer Azure SQL Database

Les connexions sans mot de passe utilisent l'authentification Microsoft Entra pour se connecter aux services Azure, y compris la base de données Azure SQL.

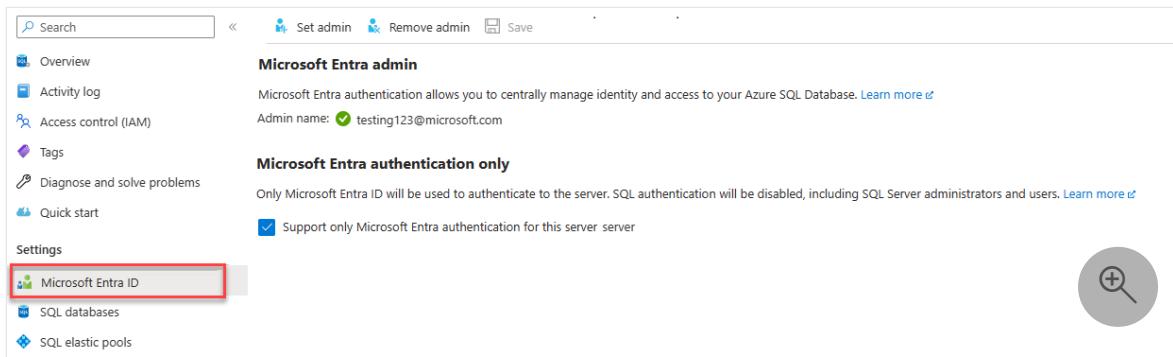
L'authentification Microsoft Entra facilite la gestion des identités dans une position centrale et simplifie la gestion des autorisations. Découvrez comment configurer l'authentification Microsoft Entra pour votre base de données Azure SQL :

- [Vue d'ensemble de l'authentification Microsoft Entra](#)
- [Configurer l'authentification Microsoft Entra](#)

Pour le présent guide de migration, assurez-vous qu'un administrateur Microsoft Entra est affecté à votre base de données Azure SQL.

1. Accédez à la page **Microsoft Entra** de votre serveur logique.
2. Sélectionnez **Définir l'administrateur** pour ouvrir le menu à contrôle suspendu de **Microsoft Entra ID**.
3. Dans le menu à contrôle suspendu de **Microsoft Entra ID**, recherchez l'utilisateur auquel vous souhaitez attribuer le rôle d'administrateur.

4. Sélectionnez l'utilisateur et choisissez Sélectionner.



The screenshot shows the Microsoft Entra admin center interface. At the top, there's a search bar and navigation links for Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Quick start, Set admin, Remove admin, and Save. Below these, the 'Microsoft Entra admin' section is shown with the sub-section 'Microsoft Entra authentication only'. It states that only Microsoft Entra ID will be used for authentication, and SQL authentication will be disabled. A checkbox labeled 'Support only Microsoft Entra authentication for this server' is checked. In the 'Settings' sidebar, the 'Microsoft Entra ID' option is highlighted with a red box. Other options like SQL databases and SQL elastic pools are also listed.

Configurer votre environnement de développement local

Les connexions sans mot de passe peuvent être configurées de telle manière qu'elles fonctionnent à la fois pour les environnements locaux et hébergés par Azure. Dans cette section, vous appliquez des configurations afin de permettre aux utilisateurs individuels de s'authentifier auprès d'Azure SQL Database pour le développement local.

Se connecter à Azure

Pour le développement local, assurez-vous que vous êtes connecté avec le même compte Azure AD que celui que vous souhaitez utiliser pour accéder à Azure SQL Database. Vous pouvez vous authentifier au moyen d'outils de développement populaires, comme Azure CLI ou Azure PowerShell. Les outils de développement avec lesquels vous pouvez vous authentifier dépendent de la langue.

Azure CLI

Connectez-vous à Azure via Azure CLI à l'aide de la commande suivante :

```
Azure CLI
az login
```

Créer un utilisateur de base de données et attribuer des rôles

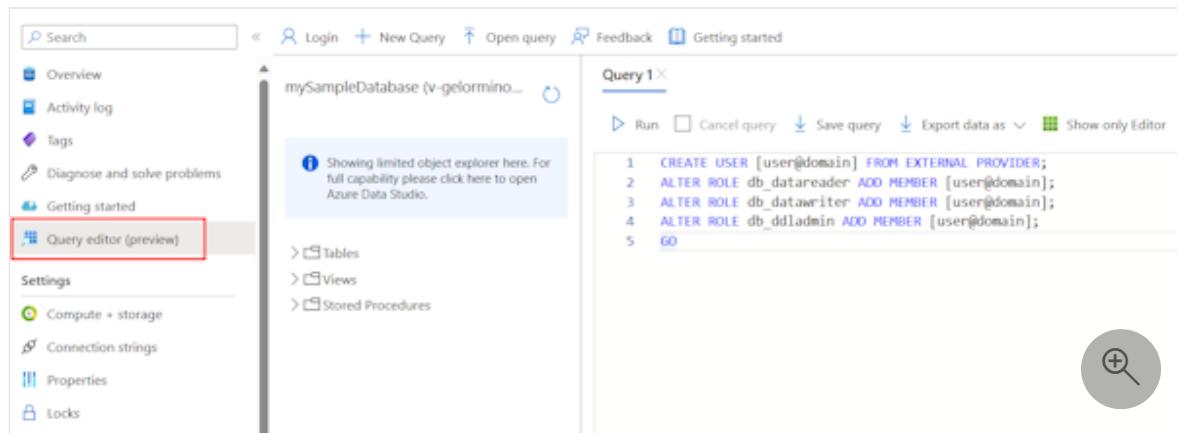
Créez un utilisateur dans Azure SQL Database. L'utilisateur doit correspondre au compte Azure que vous avez utilisé pour vous connecter localement dans la section [Se connecter à Azure](#).

connecter à Azure.

1. Dans le [portail Azure](#), accédez à votre base de données SQL et sélectionnez **Éditeur de requête (préversion)**.
2. Sélectionnez **Continuer en tant que <your-username>** sur le côté droit de l'écran pour vous connecter à la base de données à l'aide de votre compte.
3. Dans la vue de l'éditeur de requête, exécutez les commandes T-SQL suivantes :

```
SQL

CREATE USER [user@domain] FROM EXTERNAL PROVIDER;
ALTER ROLE db_datareader ADD MEMBER [user@domain];
ALTER ROLE db_datawriter ADD MEMBER [user@domain];
ALTER ROLE db_ddladmin ADD MEMBER [user@domain];
GO
```



L'exécution de ces commandes attribue le rôle [Contributeur de base de données SQL](#) au compte spécifié. Ce rôle permet à l'identité de lire, d'écrire et de modifier les données et le schéma de votre base de données. Pour plus d'informations sur les rôles attribués, consultez [Rôles de base de données fixes](#).

Mettre à jour la configuration de la connexion locale

Le code de l'application existante qui se connecte à Azure SQL Database en utilisant le pilote [SQL Python - pyodbc](#) continue de fonctionner avec des connexions sans mot de passe moyennant des modifications mineures. Par exemple, le code suivant fonctionne à la fois avec des connexions via l'authentification SQL et avec des connexions sans mot de passe lors d'une exécution locale et quand il est déployé sur Azure App Service.

```
Python

import os
import pyodbc, struct
```

```

from azure.identity import DefaultAzureCredential

connection_string = os.environ["AZURE_SQL_CONNECTIONSTRING"]

def get_all():
    with get_conn() as conn:
        cursor = conn.cursor()
        cursor.execute("SELECT * FROM Persons")
        # Do something with the data
    return

def get_conn():
    credential =
    DefaultAzureCredential(exclude_interactive_browser_credential=False)
    token_bytes =
    credential.get_token("https://database.windows.net/.default").token.encode("UTF-16-LE")
    token_struct = struct.pack(f'<I{len(token_bytes)}s', len(token_bytes),
    token_bytes)
    SQL_COPT_SS_ACCESS_TOKEN = 1256 # This connection option is defined by
    microsoft in msodbcsql.h
    conn = pyodbc.connect(connection_string, attrs_before=
    {SQL_COPT_SS_ACCESS_TOKEN: token_struct})
    return conn

```

💡 Conseil

Dans cet exemple de code, la variable d'environnement App Service `WEBSITE_HOSTNAME` est utilisée pour déterminer l'environnement dans lequel le code s'exécute. Pour d'autres scénarios de déploiement, vous pouvez utiliser d'autres variables d'environnement pour déterminer l'environnement.

Pour mettre à jour la chaîne de connexion référencée (`AZURE_SQL_CONNECTIONSTRING`) pour le développement local, utilisez le format de chaîne de connexion sans mot de passe :

```

Driver={ODBC Driver 18 for SQL Server};Server=tcp:<database-server-
name>.database.windows.net,1433;Database=<database-
name>;Encrypt=yes;TrustServerCertificate=no;Connection Timeout=30

```

Tester l'application

Exécutez votre application localement et vérifiez que les connexions à Azure SQL Database fonctionnent comme prévu. N'oubliez pas qu'il peut falloir plusieurs minutes

pour que les modifications des utilisateurs et des rôles se propagent dans l'environnement Azure. Votre application est désormais configurée pour s'exécuter localement sans que les développeurs ne doivent gérer les secrets dans l'application elle-même.

Configurer l'environnement d'hébergement Azure

Une fois que votre application est configurée pour utiliser des connexions sans mot de passe localement, le même code peut s'authentifier auprès d'Azure SQL Database après son déploiement sur Azure. Les sections suivantes expliquent comment configurer une application déployée pour se connecter à Azure SQL Database à l'aide d'une [identité managée](#). Les identités managées fournissent une identité managée automatiquement dans Microsoft Entra ID ([anciennement Azure Active Directory](#)) que les applications peuvent utiliser lors de la connexion à des ressources qui prennent en charge l'authentification Microsoft Entra. En savoir plus sur les identités managées :

- [Vue d'ensemble des connexions sans mot de passe](#)
- [Meilleures pratiques d'identité managée](#)

Créer l'identité managée

Créez une identité managée affectée par l'utilisateur à l'aide du portail Azure ou de l'interface Azure CLI. Votre application utilise l'identité pour s'authentifier auprès d'autres services.

Azure portal

1. À partir du Portail Azure, recherchez *Identités managées*. Sélectionnez le résultat **Identités managées**.
2. Sélectionnez + **Créer** en haut de la page de présentation des **Identités managées**.
3. Sous l'onglet **Informations de base**, entrez les valeurs suivantes :
 - **Abonnement** : sélectionnez l'option souhaitée.
 - **Groupe de ressources** : sélectionnez votre groupe de ressources souhaité.
 - **Région** : sélectionnez une région proche de votre emplacement.
 - **Nom** : entrez un nom reconnaissable pour votre identité, par exemple *MigrationIdentity*.

4. Au bas de la page, sélectionnez **Examiner et créer**.
5. Une fois les vérifications de validation terminées, sélectionnez **Créer**. Azure crée une identité affectée par l'utilisateur.

Une fois la ressource créée, sélectionnez **Accéder à la ressource** pour afficher les détails de l'identité managée.

The screenshot shows the 'Create User Assigned Managed Identity' wizard. The 'Basics' tab is selected. In the 'Project details' section, the subscription is set to 'Test Subscription' and the resource group is set to 'passwordlesstesting'. In the 'Instance details' section, the region is set to 'South Central US' and the name is set to 'MigrationIdentity'. At the bottom, there are buttons for 'Review + create', '< Previous', 'Next : Tags >', and a search icon.

Associer l'identité managée à votre application web

Configurez votre application web pour utiliser l'identité managée affectée par l'utilisateur créée.

Azure portal

Procédez comme suit dans le portail Azure pour associer une identité managée affectée par l'utilisateur à votre application. Ces mêmes étapes s'appliquent aux services Azure suivants :

- Azure Spring Apps
- Azure Container Apps
- Machines virtuelles Azure
- Azure Kubernetes Service
- Accédez à la page de présentation de votre application web front-end.

1. Sélectionnez **Identité** dans la barre de navigation de gauche.
2. Dans la page **Identité**, basculez vers l'onglet **Utilisateur affecté**.
3. Sélectionnez **+ Ajouter** pour ouvrir le menu volant **Ajouter une identité managée affectée par l'utilisateur**.
4. Sélectionnez l'abonnement utilisé précédemment pour créer l'identité.
5. Recherchez **MigrationIdentity** par nom et sélectionnez-le dans les résultats de la recherche.
6. Sélectionnez **Ajouter** pour associer l'identité à votre application.

The screenshot shows the Azure portal interface for managing user-assigned managed identities. On the left, there's a sidebar with various navigation options like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Microsoft Defender for Cloud, Events (preview), Deployment slots, Deployment Center, Configuration, Authentication, Application Insights, and Identity. The 'Identity' option is highlighted with a red box. The main content area has tabs for 'System assigned' and 'User assigned', with 'User assigned' selected and highlighted with a red box. Below the tabs, there's a search bar, a 'Subscription' dropdown set to 'TestSubscription', and a list of user-assigned identities. One identity, 'passwordlessuser', is selected and highlighted with a red box. The 'Selected identities' section on the right shows the same identity listed with its resource group and subscription details. There are buttons for 'Add user assigned managed identity' and 'Add' at the bottom.

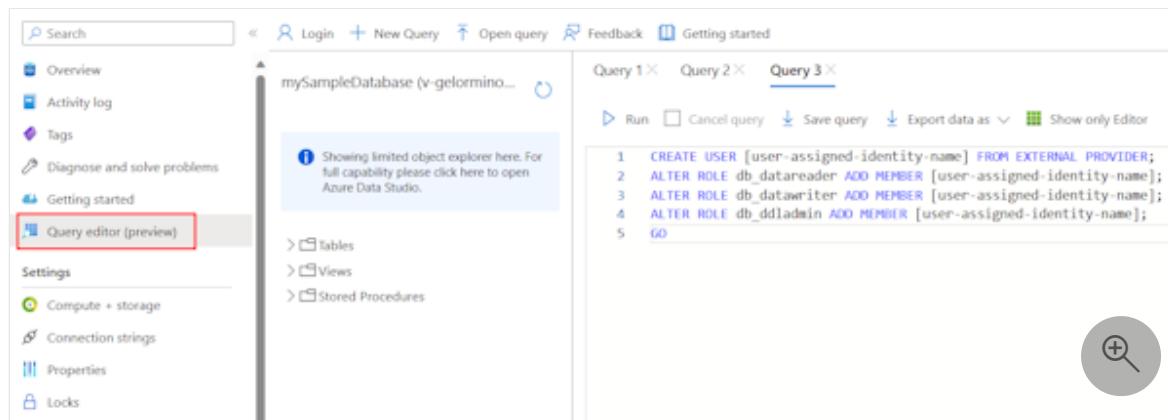
Créer un utilisateur de base de données pour l'identité et attribuer des rôles

Créez un utilisateur de base de données SQL mappé à l'identité managée affectée par l'utilisateur. Attribuez également les rôles SQL nécessaires à l'utilisateur pour autoriser votre application à lire, écrire et modifier les données et le schéma de votre base de données.

1. Dans le Portail Azure, accédez à votre base de données SQL et sélectionnez **Éditeur de requête (préversion)**.
2. Sélectionnez **Continuer en tant que <username>** sur le côté droit de l'écran pour vous connecter à la base de données à l'aide de votre compte.
3. Dans la vue de l'éditeur de requête, exécutez les commandes T-SQL suivantes :

SQL

```
CREATE USER [user-assigned-identity-name] FROM EXTERNAL PROVIDER;
ALTER ROLE db_datareader ADD MEMBER [user-assigned-identity-name];
ALTER ROLE db_datawriter ADD MEMBER [user-assigned-identity-name];
ALTER ROLE db_ddladmin ADD MEMBER [user-assigned-identity-name];
GO
```



L'exécution de ces commandes attribue le rôle **Contributeur de base de données SQL** à l'identité managée affectée par l'utilisateur. Ce rôle permet à l'identité de lire, d'écrire et de modifier les données et le schéma de votre base de données.

ⓘ Important

Faites preuve de précaution quand vous attribuez des rôles d'utilisateur de base de données dans des environnements de production d'entreprise. Dans ces scénarios, l'application ne doit pas effectuer toutes les opérations en utilisant une même identité avec priviléges élevés. Essayez d'implémenter le principe des priviléges minimum en configurant plusieurs identités disposant d'autorisations spécifiques pour des tâches spécifiques.

Vous pouvez en apprendre plus sur la configuration des rôles de base de données et la sécurité avec les ressources suivantes :

- [Tutorial : Sécuriser une base de données dans Azure SQL Database](#)

- Autoriser l'accès base de données à SQL Database

Mettre à jour la chaîne de connexion

Mettez à jour la configuration de votre application Azure pour utiliser le format de chaîne de connexion sans mot de passe. Le format doit être le même que celui utilisé dans votre environnement local.

Les chaînes de connexion peuvent être stockées en tant que variables d'environnement dans votre environnement d'hébergement d'applications. Les instructions suivantes se concentrent sur App Service, mais d'autres services d'hébergement Azure fournissent des configurations similaires.

```
Driver={ODBC Driver 18 for SQL Server};Server=tcp:<database-server-name>.database.windows.net,1433;Database=<database-name>;Encrypt=yes;TrustServerCertificate=no;Connection Timeout=30
```

<database-server-name> est le nom de votre serveur de base de données Azure SQL et <database-name> est le nom de votre base de données Azure SQL.

Créer un paramètre d'application pour l'ID client de l'identité managée

Pour utiliser l'identité managée affectée par l'utilisateur, créez une variable d'environnement AZURE_CLIENT_ID et définissez-la sur l'ID client de l'identité managée. Vous pouvez définir cette variable dans la section **Configuration** de votre application dans le portail Azure. Vous trouverez l'ID client dans la section **Vue d'ensemble** de la ressource d'identité managée dans le portail Azure.

Enregistrez vos modifications et redémarrez l'application si elle ne redémarre pas automatiquement.

ⓘ Notes

L'exemple de code de connexion présenté dans ce guide de migration utilise la classe **DefaultAzureCredential** lors du déploiement. Plus précisément, il utilise **DefaultAzureCredential** sans passer au constructeur l'ID client de l'identité managée affectée par l'utilisateur. Dans ce cas, la solution de repli consiste à vérifier la variable d'environnement AZURE_CLIENT_ID. Si la variable d'environnement

AZURE_CLIENT_ID n'existe pas, une identité managée affectée par le système est utilisée si elle est configurée.

Si vous passez l'ID client de l'identité managée dans le constructeur DefaultAzureCredential, le code de connexion peut toujours être utilisé localement et déployé, car le processus d'authentification revient à l'authentification interactive dans le scénario local. Pour plus d'informations, consultez [Bibliothèque cliente d'identité Azure pour Python](#).

Tester l'application

Testez votre application pour vous assurer de son bon fonctionnement. La propagation de l'ensemble des modifications dans votre environnement Azure peut prendre quelques minutes.

Étapes suivantes

Dans ce didacticiel, vous avez appris à migrer une application vers des connexions sans mot de passe.

Vous pouvez lire les ressources suivantes pour explorer les concepts abordés dans cet article plus en détails :

- [Vue d'ensemble des connexions sans mot de passe](#)
- [Meilleures pratiques d'identité managée](#)
- [Tutoriel : Sécuriser une base de données dans Azure SQL Database](#)
- [Autoriser l'accès base de données à SQL Database](#)

Migrer une application pour utiliser des connexions sans mot de passe avec Azure Cosmos DB for NoSQL

Article • 06/06/2023

Les demandes d'application adressées à Azure Cosmos DB for NoSQL doivent être authentifiées. Bien qu'il existe plusieurs options pour l'authentification auprès d'Azure Cosmos DB, vous devez donner la priorité aux connexions sans mot de passe dans vos applications quand cela est possible. Les méthodes d'authentification traditionnelles qui utilisent des chaînes de connexion avec des mots de passe ou des clés secrètes créent des risques et des complications de sécurité. Visitez le hub de [connexions sans mot de passe pour Azure Services](#) pour découvrir l'avantage des connexions sans mot de passe.

Le tutoriel suivant explique comment migrer une application existante pour se connecter à Azure Cosmos DB for NoSQL à l'aide de connexions sans mot de passe au lieu d'une solution basée sur des clés.

Configurer des rôles et des utilisateurs pour l'authentification de développement local

Lors du développement local avec l'authentification sans mot de passe, assurez-vous que le compte d'utilisateur qui se connecte à Cosmos DB se voie attribuer un rôle avec les autorisations appropriées pour effectuer des opérations de données. Actuellement, Azure Cosmos DB for NoSQL n'inclut pas de rôles intégrés pour les opérations de données, mais vous pouvez créer les vôtres à l'aide d'Azure CLI ou de PowerShell.

Les rôles se composent d'une collection d'autorisations ou d'actions qu'un utilisateur est autorisé à effectuer, telles que la lecture, l'écriture et la suppression. Vous pouvez en savoir plus sur [la configuration du contrôle d'accès en fonction du rôle \(RBAC\)](#) dans la documentation sur la configuration de la sécurité Cosmos DB.

Créer le rôle personnalisé

1. Attribuez un rôle à l'aide de la commande `az role definition create`. Transmettez le nom et le groupe de ressources du compte Cosmos DB, suivis d'un corps de JSON qui définit le rôle personnalisé. L'exemple suivant crée un rôle nommé `PasswordlessReadWrite` avec des autorisations pour lire et écrire des éléments dans

des conteneurs Cosmos DB. Le rôle est également étendu au niveau du compte à l'aide de `/`.

Azure CLI

```
az cosmosdb sql role definition create \
--account-name <cosmosdb-account-name> \
--resource-group <resource-group-name> \
--body '{
  "RoleName": "PasswordlessReadWrite",
  "Type": "CustomRole",
  "AssignableScopes": ["/"],
  "Permissions": [
    {
      "DataActions": [
        "Microsoft.DocumentDB/databaseAccounts/readMetadata",
        "Microsoft.DocumentDB/databaseAccounts/sqlDatabases/containers/items/*"
      ],
      "Actions": [
        "Microsoft.DocumentDB/databaseAccounts/sqlDatabases/containers/*"
      ]
    }
  ]
}'
```

2. Une fois la commande terminée, copiez la valeur d'ID du champ `name` et collez-la quelque part pour une utilisation ultérieure.
3. Attribuez le rôle que vous avez créé au compte d'utilisateur ou au principal de service qui se connectera à Cosmos DB. Pendant le développement local, il s'agit généralement de votre propre compte connecté à un outil de développement tel que Visual Studio ou Azure CLI. Récupérez les détails de votre compte à l'aide de la commande `az ad user`.

Azure CLI

```
az ad user show --id "<your-email-address>"
```

4. Copiez la valeur de la propriété `id` dans les résultats et collez-la quelque part pour une utilisation ultérieure.
5. Attribuez le rôle personnalisé que vous avez créé pour votre compte d'utilisateur à l'aide de la commande `az cosmosdb sql role assignment create` et des identifiants copiés précédemment.

Azure CLI

```
az cosmosdb sql role assignment create \
    --account-name <cosmosdb-account-name> \
    --resource-group <resource-group-name> \
    --scope "/" \
    --principal-id <your-user-id> \
    --role-definition-id <your-custom-role-id>
```

Se connecter localement à Azure

Pour le développement local, vérifiez que vous êtes authentifié avec le même compte Microsoft Entra auquel vous avez attribué le rôle. Vous pouvez vous authentifier au moyen d'outils de développement populaires, comme Azure CLI ou Azure PowerShell. Les outils de développement avec lesquels vous pouvez vous authentifier dépendent de la langue.

Azure CLI

Connectez-vous à Azure via Azure CLI à l'aide de la commande suivante :

Azure CLI

```
az login
```

Migrer le code de l'application pour utiliser des connexions sans mot de passe

.NET

1. Pour utiliser `DefaultAzureCredential` dans une application .NET, installez le `Azure.Identity` package :

CLI .NET

```
dotnet add package Azure.Identity
```

2. Ajoutez le code suivant en haut de votre fichier :

C#

```
using Azure.Identity;
```

3. Identifiez les emplacements de votre code qui créent un objet `CosmosClient` pour la connexion à Azure Cosmos DB. Mettez à jour votre code pour le faire correspondre à l'exemple suivant.

C#

```
DefaultAzureCredential credential = new();

using CosmosClient client = new(
    accountEndpoint:
    Environment.GetEnvironmentVariable("COSMOS_ENDPOINT"),
    tokenCredential: credential
);
```

Exécutez l'application localement.

Après avoir apporté ces modifications de code, exécutez votre application localement. La nouvelle configuration doit récupérer vos informations d'identification locales, telles qu'Azure CLI, Visual Studio ou IntelliJ. Les rôles que vous avez attribués à votre utilisateur de développement local dans Azure permettent à votre application de se connecter au service Azure localement.

Configurer l'environnement d'hébergement Azure

Une fois que votre application est configurée pour utiliser des connexions sans mot de passe et s'exécute localement, le même code peut s'authentifier auprès des services Azure après son déploiement sur Azure. Les sections suivantes expliquent comment configurer une application déployée pour se connecter à Azure Cosmos DB à l'aide d'une identité managée.

Créer l'identité managée

Vous pouvez créer une identité managée affectée par l'utilisateur à l'aide du Portail Azure ou d'Azure CLI. Votre application utilise l'identité pour s'authentifier auprès d'autres services.

1. À partir du Portail Azure, recherchez *Identités managées*. Sélectionnez le résultat **Identités managées**.
2. Sélectionnez + **Créer** en haut de la page de présentation des **Identités managées**.
3. Sous l'onglet **Informations de base**, entrez les valeurs suivantes :
 - **Abonnement** : sélectionnez l'option souhaitée.
 - **Groupe de ressources** : sélectionnez votre groupe de ressources souhaité.
 - **Région** : sélectionnez une région proche de votre emplacement.
 - **Nom** : entrez un nom reconnaissable pour votre identité, par exemple *MigrationIdentity*.
4. Au bas de la page, sélectionnez **Examiner et créer**.
5. Une fois les vérifications de validation terminées, sélectionnez **Créer**. Azure crée une identité affectée par l'utilisateur.

Une fois la ressource créée, sélectionnez **Accéder à la ressource** pour afficher les détails de l'identité managée.

Create User Assigned Managed Identity

Basics Tags Review + create

Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * ⓘ Test Subscription

Resource group * ⓘ passwordlesstesting
Create new

Instance details

Region * ⓘ South Central US

Name * ⓘ MigrationIdentity

Review + create < Previous Next : Tags > 

Associer l'identité managée à votre application web

Vous devez configurer votre application web pour utiliser l'identité managée créée. Attribuez l'identité à votre application à l'aide du Portail Azure ou d'Azure CLI.

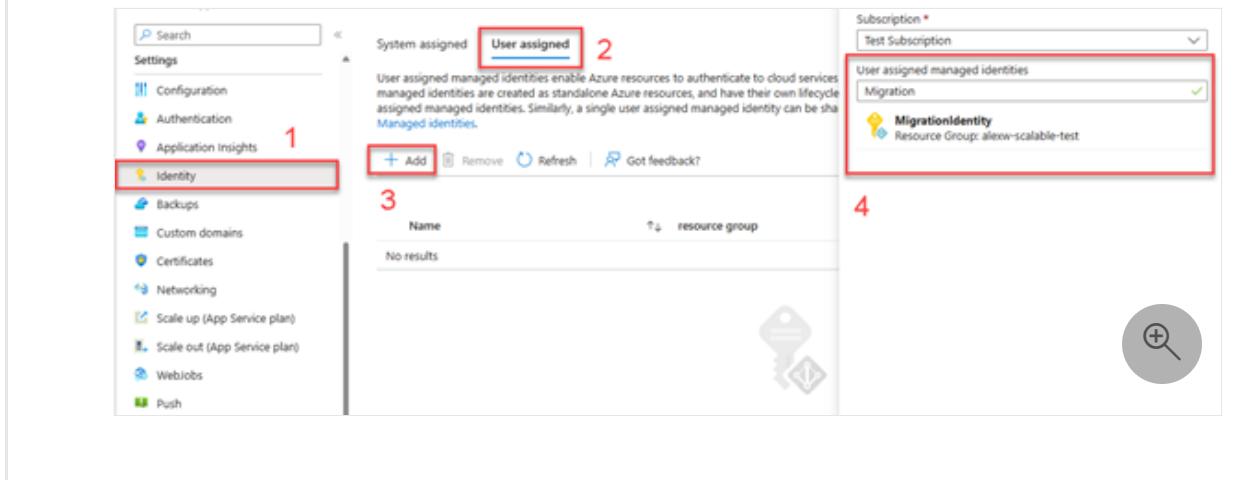
Azure portal

Effectuez les étapes suivantes dans le Portail Azure pour associer une identité à votre application. Ces mêmes étapes s'appliquent aux services Azure suivants :

- Azure Spring Apps
- Azure Container Apps
- Machines virtuelles Azure
- Azure Kubernetes Service

1. Accédez à la page de présentation de votre application web front-end.
2. Sélectionnez Identité dans la barre de navigation de gauche.
3. Dans la page Identité, basculez vers l'onglet Utilisateur affecté.

4. Sélectionnez + Ajouter pour ouvrir le menu volant Ajouter une identité managée affectée par l'utilisateur.
5. Sélectionnez l'abonnement utilisé précédemment pour créer l'identité.
6. Recherchez **MigrationIdentity** par nom et sélectionnez-le dans les résultats de la recherche.
7. Sélectionnez Ajouter pour associer l'identité à votre application.



Attribuer des rôles à l'identité managée

Accordez des autorisations à l'identité managée en lui affectant le rôle personnalisé que vous avez créé, comme vous l'avez fait avec votre utilisateur de développement local.

Pour affecter un rôle au niveau de la ressource à l'aide d'Azure CLI, vous devez d'abord récupérer l'ID de ressource à l'aide de la commande [az cosmosdb show](#). Vous pouvez filtrer les propriétés de sortie à l'aide du paramètre `--query`.

```
Azure CLI
az cosmosdb show \
--resource-group '<resource-group-name>' \
--name '<cosmosdb-name>' \
--query id
```

Copiez l'ID de sortie à partir de la commande précédente. Vous pouvez ensuite attribuer des rôles à l'aide de la commande [az role assignment](#) d'Azure CLI.

```
Azure CLI
az role assignment create \
--assignee "<your-managed-identity-name>" \
```

```
--role "PasswordlessReadWrite" \
--scope "<cosmosdb-resource-id>"
```

Mettre à jour le code d'application

Vous devez configurer votre code d'application pour rechercher l'identité managée spécifique créée lors du déploiement sur Azure. Dans certains scénarios, la définition explicite de l'identité managée pour l'application empêche également d'autres identités d'environnement d'être détectées et utilisées automatiquement par accident.

1. Dans la page de présentation de l'identité managée, copiez la valeur de l'ID client dans votre Presse-papiers.
2. Appliquez les modifications spécifiques au langage suivantes :

.NET

Créez un objet `DefaultAzureCredentialOptions` et transmettez-le à `DefaultAzureCredential`. Définissez la propriété `ManagedIdentityClientId` sur l'ID client.

C#

```
DefaultAzureCredential credential = new(
    new DefaultAzureCredentialOptions
    {
        ManagedIdentityClientId = managedIdentityClientId
    });

```

3. Redéployez votre code vers Azure après avoir apporté cette modification afin que les mises à jour de configuration soient appliquées.

Tester l'application

Après avoir déployé le code mis à jour, naviguez vers votre application hébergée dans le navigateur. Votre application doit être en mesure de se connecter à Cosmos DB avec succès. N'oubliez pas qu'il peut falloir plusieurs minutes pour que les attributions de rôle se propagent dans l'environnement Azure. Votre application est désormais configurée pour s'exécuter localement et dans un environnement de production sans que les développeurs n'aient à gérer les secrets dans l'application elle-même.

Étapes suivantes

Dans ce didacticiel, vous avez appris à migrer une application vers des connexions sans mot de passe.

Vous pouvez lire les ressources suivantes pour explorer les concepts abordés dans cet article plus en détails :

- [Autoriser l'accès aux objets blob à l'aide de Microsoft Entra ID](#))
- Pour plus d'informations sur .NET, consultez [Démarrer avec .NET en 10 minutes ↗](#).

Migrer une application pour utiliser des connexions sans mot de passe avec Azure Event Hubs

Article • 16/06/2023

Les demandes d'application adressées aux services Azure doivent être authentifiées à l'aide de configurations telles que des clés d'accès de compte ou de connexions sans mot de passe. Toutefois, vous devez hiérarchiser les connexions sans mot de passe dans vos applications lorsque cela est possible. Les méthodes d'authentification traditionnelles qui utilisent des mots de passe ou des clés secrètes créent des risques et des complications de sécurité. Visitez le hub de [connexions sans mot de passe pour Azure Services](#) pour découvrir l'avantage des connexions sans mot de passe.

Le tutoriel suivant explique comment migrer une application existante pour se connecter et utiliser des connexions sans mot de passe. Ces mêmes étapes de migration doivent s'appliquer, que vous utilisez des clés d'accès, des chaînes de connexion ou une autre approche basée sur des secrets.

Configurer votre environnement de développement local

Les connexions sans mot de passe peuvent être configurées pour fonctionner dans les environnements locaux et hébergés par Azure. Dans cette section, vous allez appliquer des configurations afin de permettre aux utilisateurs individuels de s'authentifier auprès d'Azure Event Hubs pour le développement local.

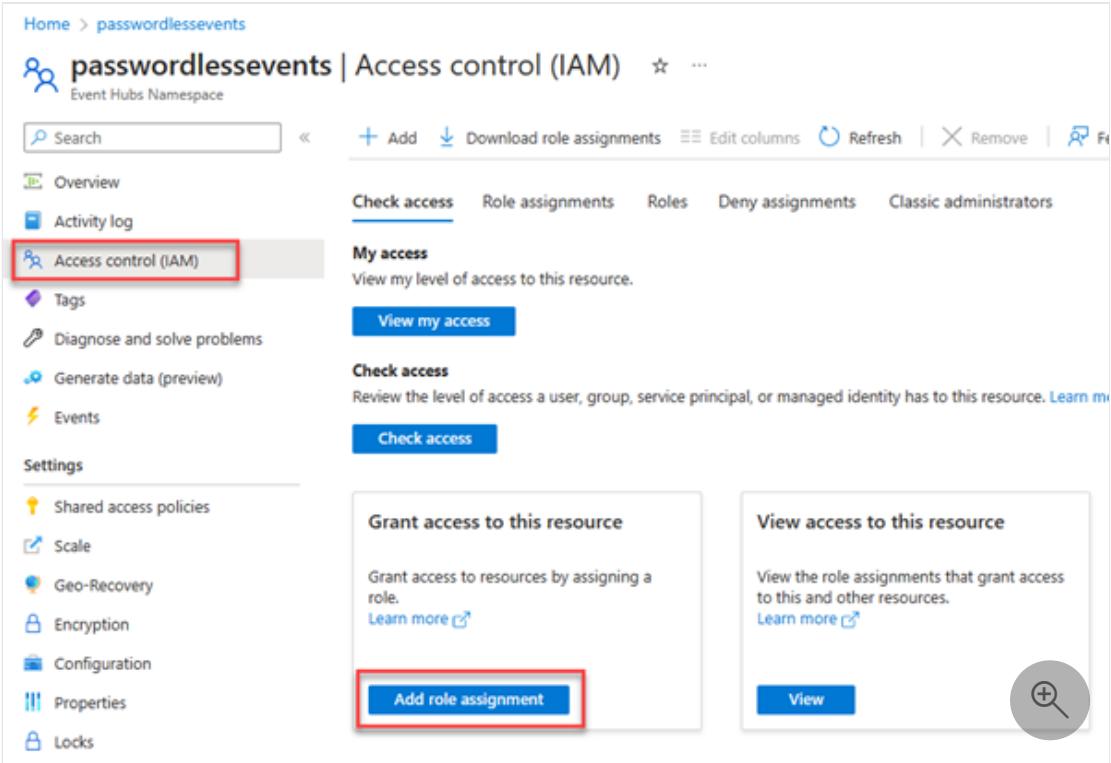
Attribuer des rôles d'utilisateur

Lors du développement localement, vous devez vérifier que le compte d'utilisateur qui accède à Azure Event Hubs dispose des autorisations appropriées. Vous aurez besoin des rôles **Récepteur de données Azure Event Hubs** et **Expéditeur de données Azure Event Hubs** pour lire et écrire des données de message. Pour vous attribuer ce rôle, vous aurez besoin du rôle **Administrateur de l'accès utilisateur** ou d'un autre rôle qui inclut l'action **Microsoft.Authorization/roleAssignments/write**. Vous pouvez attribuer des rôles RBAC Azure à un utilisateur à l'aide du Portail Azure, Azure CLI ou Azure PowerShell. Découvrez les étendues disponibles pour les attributions de rôles dans la page [vue d'ensemble de l'étendue](#).

L'exemple suivant attribue les rôles **Expéditeur de données Azure Event Hubs** et **Destinataire de données Azure Event Hubs** à votre compte d'utilisateur. Ce rôle accorde un accès en lecture et en écriture aux messages du hub d'événements.

Azure portal

1. Dans le portail Azure, recherchez votre hub local à l'aide de la barre de recherche principale ou de la navigation à gauche.
2. Dans la page Vue d'ensemble du hub d'événements, sélectionnez **Contrôle d'accès (IAM)** dans le menu de gauche.
3. Sur la page **Contrôle d'accès (IAM)**, sélectionnez l'onglet **Attributions de rôles**.
4. Sélectionnez **+ Ajouter** dans le menu supérieur, puis **Ajouter une attribution de rôle** dans le menu déroulant résultant.



5. Utilisez la zone de recherche pour filtrer les résultats sur le rôle souhaité. Pour cet exemple, recherchez *Expéditeur de données Azure Event Hubs*, sélectionnez le résultat correspondant, puis choisissez **Suivant**.
6. Sous **Attribuer l'accès à**, sélectionnez **Utilisateur, groupe ou principal de service**, puis sélectionnez **+ Sélectionner des membres**.

7. Dans la boîte de dialogue, recherchez votre nom d'utilisateur Microsoft Entra (généralement votre adresse e-mail *user@domain*), puis choisissez **Selectionner** en bas de la boîte de dialogue.
8. Sélectionnez **Vérifier + affecter** pour accéder à la page finale, puis **Vérifier + attribuer** à nouveau pour terminer le processus.
9. Répétez ces étapes pour le rôle **Destinataire de données Azure Event Hubs** afin d'autoriser le compte à envoyer et à recevoir des messages.

Important

Dans la plupart des cas, la propagation de l'attribution de rôle dans Azure prend une ou deux minutes, mais dans de rares cas, elle peut prendre jusqu'à huit minutes. Si vous recevez des erreurs d'authentification lorsque vous exécutez votre code pour la première fois, patientez quelques instants et réessayez.

Se connecter localement à Azure

Pour le développement local, vérifiez que vous êtes authentifié avec le même compte Microsoft Entra auquel vous avez attribué le rôle. Vous pouvez vous authentifier au moyen d'outils de développement populaires, comme Azure CLI ou Azure PowerShell. Les outils de développement avec lesquels vous pouvez vous authentifier dépendent de la langue.

Azure CLI

Connectez-vous à Azure via Azure CLI à l'aide de la commande suivante :

```
Azure CLI
```

```
az login
```

Mettre à jour le code de l'application pour utiliser des connexions sans mot de passe

La bibliothèque de client Azure Identity, pour chacun des écosystèmes suivants, fournit une classe `DefaultAzureCredential` qui gère l'authentification sans mot de passe auprès

d'Azure :

- .NET
- C++ ↗
- Go ↗
- Java
- Node.js
- Python

`DefaultAzureCredential` prend en charge plusieurs méthodes d'authentification. La méthode à utiliser est déterminée au moment de l'exécution. Cette approche permet à votre application d'utiliser différentes méthodes d'authentification dans différents environnements (local ou production) sans implémenter de code spécifique à l'environnement. Consultez les liens précédents pour connaître l'ordre et les emplacements dans lesquels `DefaultAzureCredential` recherche des informations d'identification.

.NET

1. Pour utiliser `DefaultAzureCredential` dans une application .NET, installez le `Azure.Identity` package :

CLI .NET

```
dotnet add package Azure.Identity
```

2. Ajoutez le code suivant en haut de votre fichier :

C#

```
using Azure.Identity;
```

3. Identifiez les emplacements de votre code qui créent un objet

`EventHubProducerClient` ou `EventProcessorClient` pour la connexion à Azure Event Hubs. Mettez à jour votre code pour le faire correspondre à l'exemple suivant :

C#

```
DefaultAzureCredential credential = new();
var eventHubNamespace =
"https://{namespace}.servicebus.windows.net";
```

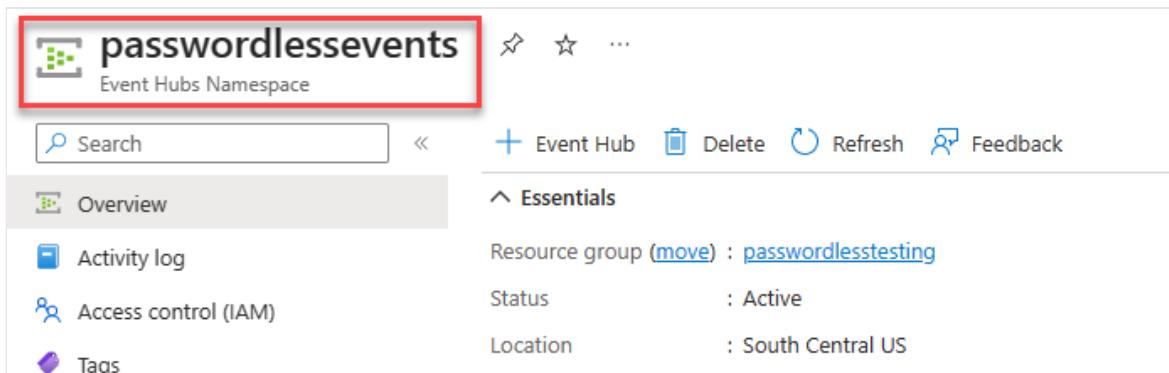
```

// Event Hubs producer
EventHubProducerClient producerClient = new(
    eventHubNamespace,
    eventHubName,
    credential);

// Event Hubs processor
EventProcessorClient processorClient = new(
    storageClient,
    EventHubConsumerClient.DefaultConsumerGroupName,
    eventHubNamespace,
    eventHubName,
    credential);

```

- Veillez à mettre à jour l'espace de noms du hub d'événements dans l'URI de vos objets `EventHubProducerClient` ou `EventProcessorClient`. Vous trouverez l'espace de noms dans la page de vue d'ensemble du portail Azure.



Exécutez l'application localement.

Après avoir apporté ces modifications de code, exécutez votre application localement. La nouvelle configuration doit récupérer vos informations d'identification locales, telles qu'Azure CLI, Visual Studio ou IntelliJ. Les rôles que vous avez attribués à votre utilisateur dans Azure permettent à votre application de se connecter au service Azure localement.

Configurer l'environnement d'hébergement Azure

Une fois que votre application est configurée pour utiliser des connexions sans mot de passe et s'exécute localement, le même code peut s'authentifier auprès des services Azure après son déploiement sur Azure. Les sections suivantes expliquent comment configurer une application déployée pour se connecter à Azure Event Hubs à l'aide

d'une [identité managée](#). Les identités managées fournissent une identité managée automatiquement dans Microsoft Entra ID que les applications peuvent utiliser lors de la connexion à des ressources qui prennent en charge l'authentification Microsoft Entra. En savoir plus sur les identités managées :

- [Vue d'ensemble des connexions sans mot de passe](#)
- [Meilleures pratiques d'identité managée](#)

Créer l'identité managée

Vous pouvez créer une identité managée affectée par l'utilisateur à l'aide du Portail Azure ou d'Azure CLI. Votre application utilise l'identité pour s'authentifier auprès d'autres services.

Azure portal

1. À partir du Portail Azure, recherchez *Identités managées*. Sélectionnez le résultat **Identités managées**.
2. Sélectionnez + **Créer** en haut de la page de présentation des **Identités managées**.
3. Sous l'onglet **Informations de base**, entrez les valeurs suivantes :
 - **Abonnement** : sélectionnez l'option souhaitée.
 - **Groupe de ressources** : sélectionnez votre groupe de ressources souhaité.
 - **Région** : sélectionnez une région proche de votre emplacement.
 - **Nom** : entrez un nom reconnaissable pour votre identité, par exemple *MigrationIdentity*.
4. Au bas de la page, sélectionnez **Examiner et créer**.
5. Une fois les vérifications de validation terminées, sélectionnez **Créer**. Azure crée une identité affectée par l'utilisateur.

Une fois la ressource créée, sélectionnez **Accéder à la ressource** pour afficher les détails de l'identité managée.

Create User Assigned Managed Identity

Basics Tags Review + create

Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * ⓘ

Test Subscription

Resource group * ⓘ

passwordlesstesting

[Create new](#)

Instance details

Region * ⓘ

South Central US

Name * ⓘ

MigrationIdentity



[Review + create](#)

[< Previous](#)

[Next : Tags >](#)

Associer l'identité managée à votre application web

Vous devez configurer votre application web pour utiliser l'identité managée créée. Attribuez l'identité à votre application à l'aide du Portail Azure ou d'Azure CLI.

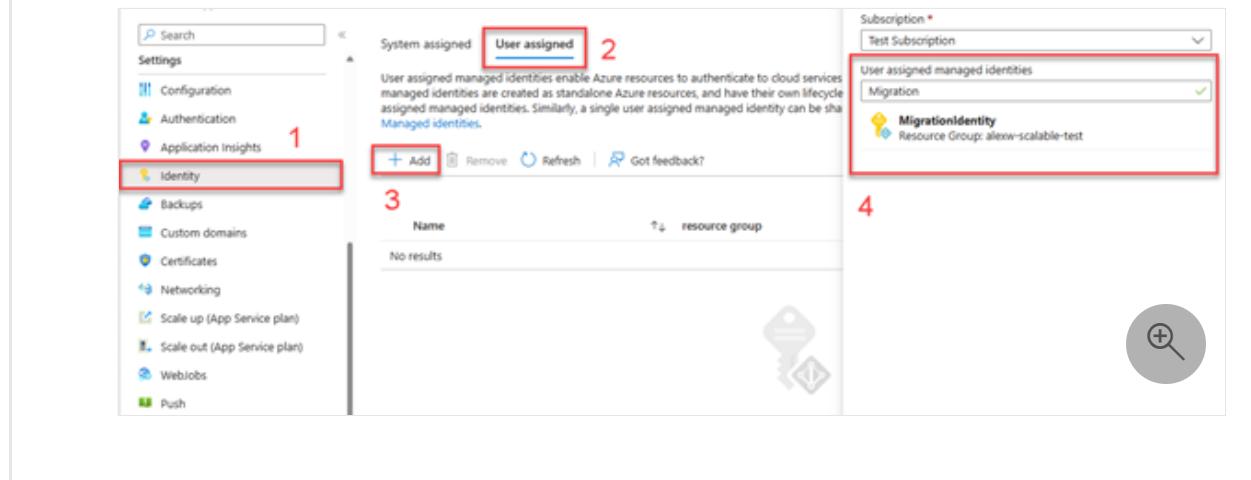
Azure portal

Effectuez les étapes suivantes dans le Portail Azure pour associer une identité à votre application. Ces mêmes étapes s'appliquent aux services Azure suivants :

- Azure Spring Apps
- Azure Container Apps
- Machines virtuelles Azure
- Azure Kubernetes Service

1. Accédez à la page de présentation de votre application web front-end.
2. Sélectionnez Identité dans la barre de navigation de gauche.
3. Dans la page Identité, basculez vers l'onglet Utilisateur affecté.

4. Sélectionnez + Ajouter pour ouvrir le menu volant Ajouter une identité managée affectée par l'utilisateur.
5. Sélectionnez l'abonnement utilisé précédemment pour créer l'identité.
6. Recherchez **MigrationIdentity** par nom et sélectionnez-le dans les résultats de la recherche.
7. Sélectionnez Ajouter pour associer l'identité à votre application.



Attribuer des rôles à l'identité managée

Ensuite, vous devez accorder des autorisations à l'identité managée que vous avez créée pour accéder à votre hub d'événements. Accordez des autorisations en affectant un rôle à l'identité managée, comme vous l'avez fait avec votre utilisateur de développement local.

Azure portal

1. Accédez à la page de vue d'ensemble du hub d'événements et sélectionnez **Access Control (IAM)** dans la navigation de gauche.
2. Choisissez **Ajouter une attribution de rôle**

3. Dans la zone de recherche de Rôle, recherchez *l'expéditeur de données Azure Event Hub*, qui est un rôle commun utilisé pour gérer les opérations de données pour les files d'attente. Vous pouvez attribuer le rôle approprié pour votre cas d'usage. Sélectionnez *Expéditeur de données Azure Event Hub* dans la liste, puis choisissez **Suivant**.
4. Sur l'écran **Ajouter une attribution de rôle**, pour l'option **Attribuer l'accès à l'option**, sélectionnez **Identité managée**. Choisissez ensuite **+ Sélectionner des membres**.
5. Dans le menu volant, recherchez l'identité managée que vous avez créée par nom, puis sélectionnez-la dans les résultats. Choisissez **Sélectionner** pour fermer le menu volant.

6. Sélectionnez **Suivant** quelques fois jusqu'à ce que vous puissiez sélectionner **Vérifier + attribuer** pour terminer l'attribution de rôle.

7. Répétez ces étapes pour le rôle Récepteur de données Azure Event Hub.

Mettre à jour le code d'application

Vous devez configurer votre code d'application pour rechercher l'identité managée spécifique créée lors du déploiement sur Azure. Dans certains scénarios, la définition explicite de l'identité managée pour l'application empêche également d'autres identités d'environnement d'être détectées et utilisées automatiquement par accident.

1. Dans la page de présentation de l'identité managée, copiez la valeur de l'ID client dans votre Presse-papiers.
2. Appliquez les modifications spécifiques au langage suivantes :

.NET

Créez un objet `DefaultAzureCredentialOptions` et transmettez-le à `DefaultAzureCredential`. Définissez la propriété `ManagedIdentityClientId` sur l'ID client.

C#

```
DefaultAzureCredential credential = new(
    new DefaultAzureCredentialOptions
    {
        ManagedIdentityClientId = managedIdentityClientId
    });

```

3. Redéployez votre code vers Azure après avoir apporté cette modification afin que les mises à jour de configuration soient appliquées.

Tester l'application

Après avoir déployé le code mis à jour, naviguez vers votre application hébergée dans le navigateur. Votre application doit être en mesure de se connecter au hub d'événements avec succès. N'oubliez pas qu'il peut falloir plusieurs minutes pour que les attributions de rôle se propagent dans l'environnement Azure. Votre application est désormais configurée pour s'exécuter localement et dans un environnement de production sans que les développeurs n'aient à gérer les secrets dans l'application elle-même.

Étapes suivantes

Dans ce didacticiel, vous avez appris à migrer une application vers des connexions sans mot de passe.

Vous pouvez lire les ressources suivantes pour explorer les concepts abordés dans cet article plus en détails :

- [Connexions sans mot de passe pour les services Azure](#)
- Pour plus d'informations sur .NET, consultez [Démarrer avec .NET en 10 minutes ↗](#).

Migrer une application pour utiliser des connexions sans mot de passe avec Azure Event Hubs pour Kafka

Article • 23/05/2023

Cet article explique comment migrer des méthodes d'authentification traditionnelles vers des connexions sans mot de passe plus sécurisées avec Azure Event Hubs pour Kafka.

Les demandes d'application adressées à Azure Event Hubs pour Kafka doivent être authentifiées. Azure Event Hubs pour Kafka fournit différentes façons pour les applications de se connecter en toute sécurité. L'une des façons d'utiliser une chaîne de connexion. Toutefois, vous devez hiérarchiser les connexions sans mot de passe dans vos applications lorsque cela est possible.

Les connexions sans mot de passe sont prises en charge depuis Spring Cloud Azure 4.3.0. Cet article est un guide de migration pour supprimer les informations d'identification des applications Kafka Spring Cloud Stream.

Comparer les options d'authentification

Lorsque l'application s'authentifie auprès d'Azure Event Hubs pour Kafka, elle fournit une entité autorisée pour connecter l'espace de noms Event Hubs. Les protocoles Apache Kafka fournissent plusieurs mécanismes d'authentification et de couche de sécurité (SASL) simples pour l'authentification. Selon les mécanismes SASL, il existe deux options d'authentification que vous pouvez utiliser pour autoriser l'accès à vos ressources sécurisées : l'authentification Microsoft Entra et l'authentification SAP (Shared Access Signature).

Authentification Microsoft Entra

L'authentification Microsoft Entra est un mécanisme de connexion à Azure Event Hubs pour Kafka à l'aide d'identités définies dans l'ID Microsoft Entra. Avec l'authentification Microsoft Entra, vous pouvez gérer les identités de principal de service et d'autres services Microsoft dans un emplacement central, ce qui simplifie la gestion des autorisations.

L'utilisation de l'ID Microsoft Entra pour l'authentification offre les avantages suivants :

- Authentification des utilisateurs sur les services Azure de manière uniforme.
- Gestion des stratégies de mot de passe et de la rotation des mots de passe à un seul endroit.
- Plusieurs formes d'authentification prises en charge par l'ID Microsoft Entra, qui peuvent éliminer la nécessité de stocker les mots de passe.
- Les clients peuvent gérer les autorisations Event Hubs à l'aide de groupes externes (Microsoft Entra ID).
- Prise en charge de l'authentification basée sur des jetons pour les applications qui se connectent à Azure Event Hubs pour Kafka.

Authentification avec SAS

Event Hubs fournit également des signatures d'accès partagé (SAP) pour l'accès délégué à Event Hubs pour les ressources Kafka.

Bien qu'il soit possible de se connecter à Azure Event Hubs pour Kafka avec SAS, il doit être utilisé avec précaution. Vous devez être vigilant pour ne jamais exposer les chaîne de connexion dans un emplacement non sécurisé. Toute personne ayant accès aux chaîne de connexion est en mesure de s'authentifier. Par exemple, il existe un risque qu'un utilisateur malveillant puisse accéder à l'application si un chaîne de connexion est accidentellement case activée dans le contrôle de code source, envoyé via un e-mail non sécurisé, collé dans une conversation incorrecte ou consulté par une personne qui ne doit pas avoir d'autorisation. Au lieu de cela, l'autorisation de l'accès à l'aide du mécanisme basé sur des jetons OAuth 2.0 offre une sécurité et une facilité d'utilisation supérieures sur SAS. Envisagez de mettre à jour votre application pour utiliser des connexions sans mot de passe.

Présentation des connexions sans mot de passe

Avec une connexion sans mot de passe, vous pouvez vous connecter aux services Azure sans stocker d'informations d'identification dans le code de l'application, ses fichiers de configuration ou dans les variables d'environnement.

De nombreux services Azure prennent en charge les connexions sans mot de passe, par exemple via Azure Managed Identity. Ces techniques fournissent des fonctionnalités de sécurité robustes que vous pouvez implémenter à l'aide [de DefaultAzureCredential](#) à partir des bibliothèques clientes Azure Identity. Dans ce tutoriel, vous allez apprendre à mettre à jour une application existante à utiliser `DefaultAzureCredential` au lieu d'alternatives telles que des chaîne de connexion.

`DefaultAzureCredential` prend en charge plusieurs méthodes d'authentification et détermine automatiquement celle qui doit être utilisée au moment de l'exécution. Cette approche permet à votre application d'utiliser différentes méthodes d'authentification dans différents environnements (développement local et production) sans implémenter de code spécifique à l'environnement.

L'ordre et les emplacements dans lesquels `DefaultAzureCredential` les recherches d'informations d'identification sont disponibles dans la vue d'ensemble [de la bibliothèque d'identités Azure](#). Par exemple, lorsque vous travaillez localement, `DefaultAzureCredential` s'authentifie généralement à l'aide du compte utilisé par le développeur pour se connecter à Visual Studio. Lorsque l'application est déployée sur Azure, `DefaultAzureCredential` bascule automatiquement pour utiliser une [identité managée](#). Aucune modification du code n'est requise pour cette transition.

Pour vous assurer que les connexions sont sans mot de passe, vous devez prendre en compte le développement local et l'environnement de production. Si une chaîne de connexion est requise à l'un ou l'autre endroit, l'application n'est pas sans mot de passe.

Dans votre environnement de développement local, vous pouvez vous authentifier auprès d'Azure CLI, d'Azure PowerShell, de Visual Studio ou de plug-ins Azure pour Visual Studio Code ou IntelliJ. Dans ce cas, vous pouvez utiliser ces informations d'identification dans votre application au lieu de configurer des propriétés.

Lorsque vous déployez des applications dans un environnement d'hébergement Azure, tel qu'une machine virtuelle, vous pouvez affecter une identité managée dans cet environnement. Ensuite, vous n'avez pas besoin de fournir des informations d'identification pour vous connecter aux services Azure.

ⓘ Notes

Une identité managée fournit une identité de sécurité pour représenter une application ou un service. Managée par la plateforme Azure, l'identité ne nécessite pas que vous approvisionniez ou permutiez de secrets. Vous pouvez en savoir plus sur les identités managées dans la documentation [vue d'ensemble](#).

Migrer une application existante pour utiliser des connexions sans mot de passe

Les étapes suivantes expliquent comment migrer une application existante pour utiliser des connexions sans mot de passe au lieu d'une solution SAP.

0) Préparer l'environnement de travail pour l'authentification de développement local

Tout d'abord, utilisez la commande suivante pour configurer certaines variables d'environnement.

Bash

```
export AZ_RESOURCE_GROUP=<YOUR_RESOURCE_GROUP>
export AZ_EVENTHUBS_NAMESPACE_NAME=<YOUR_EVENTHUBS_NAMESPACE_NAME>
export AZ_EVENTHUB_NAME=<YOUR_EVENTHUB_NAME>
```

Remplacez les espaces réservés par les valeurs suivantes, qui sont utilisées dans cet article :

- <YOUR_RESOURCE_GROUP> : nom du groupe de ressources que vous utiliserez.
- <YOUR_EVENTHUBS_NAMESPACE_NAME> : nom de l'espace de noms Azure Event Hubs que vous utiliserez.
- <YOUR_EVENTHUB_NAME> : nom du hub d'événements que vous utiliserez.

1) Accorder l'autorisation pour Azure Event Hubs

Si vous souhaitez exécuter cet exemple localement avec l'authentification Microsoft Entra, assurez-vous que votre compte d'utilisateur s'est authentifié via Azure Shared Computer Toolkit pour IntelliJ, le plug-in de compte Azure Visual Studio Code ou Azure CLI. Vérifiez également que le compte a reçu des autorisations suffisantes.

Azure portal

1. Dans le portail Azure, recherchez votre espace de noms Event Hubs à l'aide de la barre de recherche principale ou de la navigation gauche.
2. Dans la page vue d'ensemble d'Event Hubs, sélectionnez **Contrôle d'accès (IAM)** dans le menu de gauche.
3. Sur la page **Contrôle d'accès (IAM)**, sélectionnez l'onglet **Attributions de rôles**.
4. Sélectionnez **Ajouter** dans le menu supérieur, puis **Ajoutez une attribution de rôle** dans le menu déroulant résultant.

The screenshot shows the 'identityMigrationEventHubs | Access control (IAM)' blade in the Azure portal. The left sidebar has a red box around 'Access control (IAM)'. The top bar has a red box around the '+ Add' button. The main area has three sections: 'Grant access to this resource', 'View access to this resource', and 'View deny assignments'. Each section has a 'View' button and a 'Learn more' link.

5. Utilisez la zone de recherche pour filtrer les résultats sur le rôle souhaité. Pour cet exemple, recherchez *l'expéditeur* de données Azure Event Hubs et *le récepteur* de données Azure Event Hubs, puis sélectionnez le résultat correspondant, puis choisissez **Suivant**.
6. Sous **Attribuer l'accès**, sélectionnez **Utilisateur, groupe ou principal de service**, puis sélectionnez **Sélectionner des membres**.
7. Dans la boîte de dialogue, recherchez votre nom d'utilisateur Microsoft Entra (généralement votre adresse e-mail *user@domain*), puis choisissez **Sélectionner** en bas de la boîte de dialogue.
8. Sélectionnez **Vérifier + affecter** pour accéder à la page finale, puis **Vérifier + attribuer** à nouveau pour terminer le processus.

Pour plus d'informations sur l'octroi de rôles d'accès, consultez [Autoriser l'accès aux ressources Event Hubs à l'aide de l'ID Microsoft Entra](#).

2) Connectez-vous et migrez le code de l'application pour utiliser des connexions sans mot de passe

Pour le développement local, vérifiez que vous êtes authentifié avec le même compte Microsoft Entra auquel vous avez affecté le rôle sur vos Hubs d'événements. Vous pouvez vous authentifier via Azure CLI, Visual Studio, Azure PowerShell ou d'autres outils tels qu'IntelliJ.

Azure CLI

Connectez-vous à Azure via Azure CLI à l'aide de la commande suivante :

Azure CLI

```
az login
```

Ensute, procédez comme suit pour mettre à jour votre application Spring Kafka pour utiliser des connexions sans mot de passe. Bien que conceptuellement similaire, chaque infrastructure utilise des détails d'implémentation différents.

Java

1. Dans votre projet, ouvrez le *fichier pom.xml* et ajoutez la référence suivante :

XML

```
<dependency>
    <groupId>com.azure</groupId>
    <artifactId>azure-identity</artifactId>
    <version>1.6.0</version>
</dependency>
```

2. Après la migration, implémentez [AuthenticateCallbackHandler](#) et [OAuthBearerToken](#) dans votre projet pour l'authentification OAuth2, comme illustré dans l'exemple suivant.

Java

```
public class KafkaOAuth2AuthenticateCallbackHandler implements
AuthenticateCallbackHandler {

    private static final Duration ACCESS_TOKEN_REQUEST_BLOCK_TIME =
Duration.ofSeconds(30);
    private static final String TOKEN_AUDIENCE_FORMAT =
"%s://%s/.default";

    private Function<TokenCredential, Mono<OAuthBearerTokenImp>>
resolveToken;
    private final TokenCredential credential = new
DefaultAzureCredentialBuilder().build();

    @Override
    public void configure(Map<String, ?> configs, String mechanism,
List<AppConfigurationEntry> jaasConfigEntries) {
        TokenRequestContext request =
buildTokenRequestContext(configs);
        this.resolveToken = tokenCredential ->
tokenCredential.getToken(request).map(OAuthBearerTokenImp::new);
    }
}
```

```

    private TokenRequestContext buildTokenRequestContext(Map<String,
?> configs) {
    URI uri = buildEventHubsServerUri(configs);
    String tokenAudience = buildTokenAudience(uri);

    TokenRequestContext request = new TokenRequestContext();
    request.addScopes(tokenAudience);
    return request;
}

@SuppressWarnings("unchecked")
private URI buildEventHubsServerUri(Map<String, ?> configs) {
    String bootstrapServer =
    Arrays.asList(configs.get(BOOTSTRAP_SERVERS_CONFIG)).get(0).toString();
    bootstrapServer = bootstrapServer.replaceAll("\\[|\\]", "");
    URI uri = URI.create("https://" + bootstrapServer);
    return uri;
}

private String buildTokenAudience(URI uri) {
    return String.format(TOKEN_AUDIENCE_FORMAT, uri.getScheme(),
    uri.getHost());
}

@Override
public void handle(Callback[] callbacks) throws
UnsupportedCallbackException {
    for (Callback callback : callbacks) {
        if (callback instanceof OAuthBearerTokenCallback) {
            OAuthBearerTokenCallback oauthCallback =
(OAuthBearerTokenCallback) callback;
            this.resolveToken
                .apply(credential)
                .doOnNext(oauthCallback::token)
                .doOnError(throwable ->
oauthCallback.error("invalid_grant", throwable.getMessage(), null))
                .block(ACCESS_TOKEN_REQUEST_BLOCK_TIME);
        } else {
            throw new UnsupportedCallbackException(callback);
        }
    }
}

@Override
public void close() {
    // NOOP
}
}

```

```
public class OAuthBearerTokenImp implements OAuthBearerToken {
    private final AccessToken accessToken;
    private final JWTClaimsSet claims;

    public OAuthBearerTokenImp(AccessToken accessToken) {
        this.accessToken = accessToken;
        try {
            claims =
JWTParser.parse(accessToken.getToken()).getJWTClaimsSet();
        } catch (ParseException exception) {
            throw new SaslAuthenticationException("Unable to parse
the access token", exception);
        }
    }

    @Override
    public String value() {
        return accessToken.getToken();
    }

    @Override
    public Long startTimeMs() {
        return claims.getIssueTime().getTime();
    }

    @Override
    public long lifetimeMs() {
        return claims.getExpirationTime().getTime();
    }

    @Override
    public Set<String> scope() {
        // Referring to https://docs.microsoft.com/azure/active-
        // directory/develop/access-tokens#payload-claims, the scp
        // claim is a String which is presented as a space
        // separated list.
        return Optional.ofNullable(claims.getClaim("scp"))
            .map(s -> Arrays.stream((String) s)
                .split(" "))
            .collect(Collectors.toSet())
            .orElse(null);
    }

    @Override
    public String principalName() {
        return (String) claims.getClaim("upn");
    }

    public boolean isExpired() {
        return accessToken.isExpired();
    }
}
```

3. Lorsque vous créez votre producteur ou consommateur Kafka, ajoutez la configuration nécessaire pour prendre en charge le [mécanisme SASL/OAUTHBEARER](#). Les exemples suivants montrent l'apparence de votre code avant et après la migration. Dans les deux exemples, remplacez l'espace <eventhubs-namespace> réservé par le nom de votre espace de noms Event Hubs.

Avant la migration, votre code doit ressembler à l'exemple suivant :

Java

```
Properties properties = new Properties();
properties.put(CommonClientConfigs.BOOTSTRAP_SERVERS_CONFIG, "  
<eventhubs-namespace>.servicebus.windows.net:9093");
properties.put(CommonClientConfigs.SECURITY_PROTOCOL_CONFIG,
"SASL_SSL");
properties.put(ProducerConfig.KEY_SERIALIZER_CLASS_CONFIG,
StringSerializer.class.getName());
properties.put(ProducerConfig.VALUE_SERIALIZER_CLASS_CONFIG,
StringSerializer.class.getName());
properties.put(SaslConfigs.SASL_MECHANISM, "PLAIN");
properties.put(SaslConfigs.SASL_JAAS_CONFIG,  
  
String.format("org.apache.kafka.common.security.plain.PlainLoginMod  
ule required username=\"$ConnectionString\" password=\"%s\";",  
connectionString));
return new KafkaProducer<>(properties);
```

Après la migration, votre code doit ressembler à l'exemple suivant. Dans cet exemple, remplacez l'espace <path-to-your-KafkaOAuth2AuthenticateCallbackHandler> réservé par le nom de classe complet de votre implémentation `KafkaOAuth2AuthenticateCallbackHandler`.

Java

```
Properties properties = new Properties();
properties.put(CommonClientConfigs.BOOTSTRAP_SERVERS_CONFIG, "  
<eventhubs-namespace>.servicebus.windows.net:9093");
properties.put(CommonClientConfigs.SECURITY_PROTOCOL_CONFIG,
"SASL_SSL");
properties.put(SaslConfigs.SASL_MECHANISM, "OAUTHBEARER");
properties.put(SaslConfigs.SASL_JAAS_CONFIG,
"org.apache.kafka.common.security.oauthbearer.OAuthBearerLoginModul  
e required");
properties.put(SaslConfigs.SASL_LOGIN_CALLBACK_HANDLER_CLASS, "  
<path-to-your-KafkaOAuth2AuthenticateCallbackHandler>");
return new KafkaProducer<>(properties);
```

Exécutez l'application localement.

Après avoir apporté ces modifications de code, exécutez votre application localement. La nouvelle configuration doit récupérer vos informations d'identification locales, en supposant que vous êtes connecté à un outil en ligne de commande ou IDE compatible, tel que Azure CLI, Visual Studio ou IntelliJ. Les rôles que vous avez attribués à votre utilisateur de développement local dans Azure permettent à votre application de se connecter au service Azure localement.

3) Configurer l'environnement d'hébergement Azure

Une fois que votre application est configurée pour utiliser des connexions sans mot de passe et qu'elle s'exécute localement, le même code peut s'authentifier auprès des services Azure après son déploiement sur Azure. Par exemple, une application déployée sur une instance Azure Spring Apps qui a une identité managée affectée peut se connecter à Azure Event Hubs pour Kafka.

Dans cette section, vous allez exécuter deux étapes pour permettre à votre application de s'exécuter dans un environnement d'hébergement Azure de manière sans mot de passe :

- Attribuez l'identité managée pour votre environnement d'hébergement Azure.
- Attribuez des rôles à l'identité managée.

ⓘ Notes

Azure fournit également un **Connecter de service**, qui peut vous aider à connecter votre service d'hébergement avec Event Hubs. Avec service Connecter or pour configurer votre environnement d'hébergement, vous pouvez omettre l'étape d'attribution de rôles à votre identité managée, car Le Connecter or de service le fera pour vous. La section suivante explique comment configurer votre environnement d'hébergement Azure de deux façons : l'une via service Connecter or et l'autre en configurant directement chaque environnement d'hébergement.

ⓘ Important

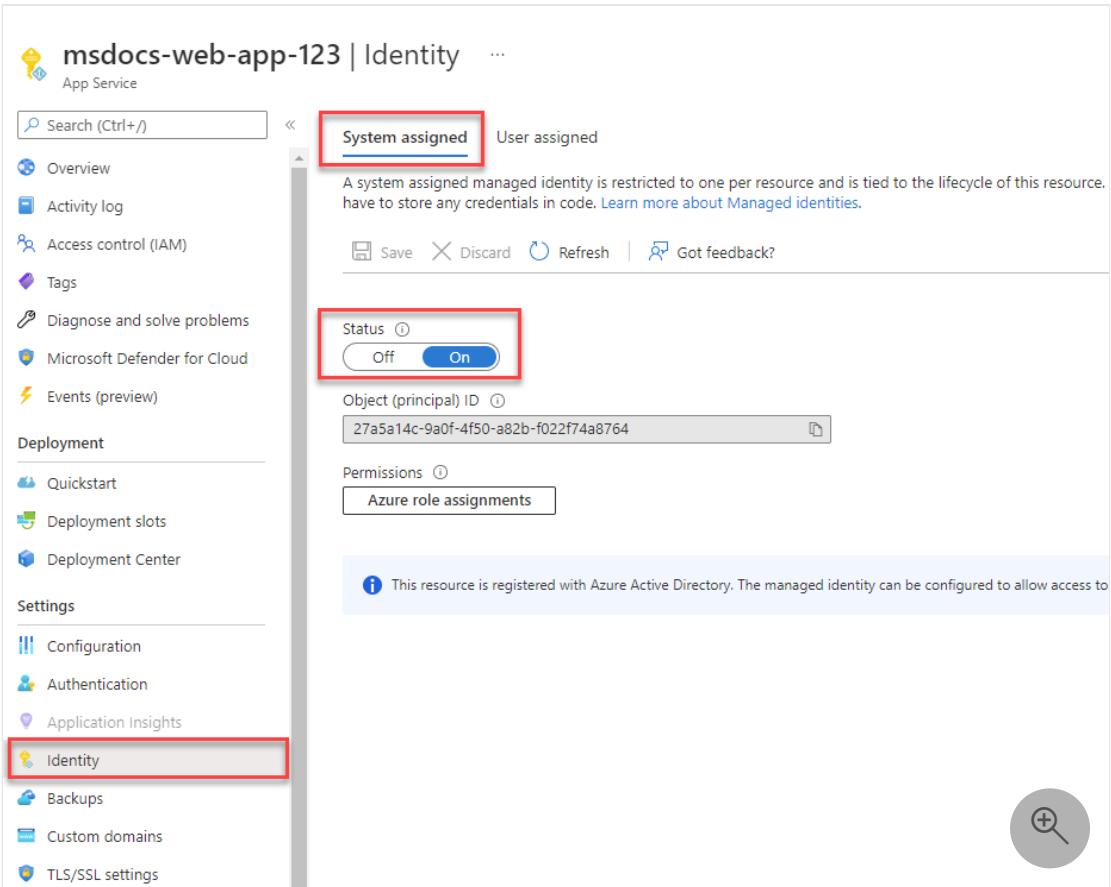
Les commandes du Connecter or de service nécessitent [Azure CLI](#) 2.41.0 ou version ultérieure.

Attribuer l'identité managée pour votre environnement d'hébergement Azure

Les étapes suivantes vous montrent comment attribuer une identité managée affectée par le système pour différents services d'hébergement web. L'identité managée peut se connecter en toute sécurité à d'autres services Azure à l'aide des configurations d'application que vous avez configurées.

App Service

1. Dans la page de vue d'ensemble principale de votre instance Azure App Service, sélectionnez **Identité** dans le volet de navigation.
2. Sous l'onglet Affecté par le système, veillez à définir le champ **État** sur activé. Une identité affectée par le système est gérée par Azure en interne et gère les tâches d'administration pour vous. Les détails et ID de l'identité ne sont jamais exposés dans votre code.



Vous pouvez également affecter une identité managée sur un environnement d'hébergement Azure à l'aide d'Azure CLI.

App Service

Vous pouvez affecter une identité managée à une instance Azure App Service avec la commande [az webapp identity assign](#), comme illustré dans l'exemple suivant.

Azure CLI

```
export AZURE_MANAGED_IDENTITY_ID=$(az webapp identity assign \
    --resource-group $AZ_RESOURCE_GROUP \
    --name <app-service-name> \
    --query principalId \
    --output tsv)
```

Attribuer des rôles à l'identité managée

Ensuite, accordez des autorisations à l'identité managée que vous avez créée pour accéder à votre espace de noms Event Hubs. Vous pouvez accorder des autorisations en affectant un rôle à l'identité managée, comme vous l'avez fait avec votre utilisateur de développement local.

Connecteur de service

Si vous avez connecté vos services à l'aide du Connecter or de service, vous n'avez pas besoin d'effectuer cette étape. Les configurations nécessaires suivantes ont été gérées pour vous :

- Si vous avez sélectionné une identité managée lors de la création de la connexion, une identité managée affectée par le système a été créée pour votre application et a affecté les *rôles Expéditeur* de données Azure Event Hubs et *Récepteur* de données Azure Event Hubs sur Event Hubs.
- Si vous avez choisi d'utiliser un chaîne de connexion, la chaîne de connexion a été ajoutée en tant que variable d'environnement d'application.

Tester l'application

Après avoir apporté ces modifications de code, accédez à votre application hébergée dans le navigateur. Votre application doit être en mesure de se connecter à Azure Event Hubs pour Kafka. N'oubliez pas qu'il peut falloir plusieurs minutes pour que les attributions de rôle se propagent dans l'environnement Azure. Votre application est

désormais configurée pour s'exécuter localement et dans un environnement de production sans que les développeurs n'aient à gérer les secrets dans l'application elle-même.

Étapes suivantes

Dans ce didacticiel, vous avez appris à migrer une application vers des connexions sans mot de passe.

Vous pouvez lire les ressources suivantes pour explorer les concepts abordés dans cet article plus en détails :

- [Autoriser l'accès aux données de blob avec des identités managées pour les ressources Azure](#)
- [Autoriser l'accès aux objets blob à l'aide Microsoft Entra ID](#)

Migrer une application pour utiliser des connexions sans mot de passe avec Azure Database pour MySQL

Article • 20/10/2023

Cet article explique comment migrer des méthodes d'authentification traditionnelles vers des connexions sans mot de passe plus sécurisées avec Azure Database pour MySQL.

Les demandes d'application à Azure Database pour MySQL doivent être authentifiées. Azure Database pour MySQL fournit plusieurs façons différentes pour les applications de se connecter en toute sécurité. L'une des façons d'utiliser des mots de passe. Toutefois, vous devez hiérarchiser les connexions sans mot de passe dans vos applications lorsque cela est possible.

Comparer les options d'authentification

Lorsque l'application s'authentifie avec Azure Database pour MySQL, elle fournit une paire nom d'utilisateur et mot de passe pour se connecter à la base de données. Selon l'emplacement de stockage des identités, il existe deux types d'authentification : l'authentification Microsoft Entra et l'authentification MySQL.

Authentification Microsoft Entra

L'authentification Microsoft Entra est un mécanisme de connexion à Azure Database pour MySQL à l'aide d'identités définies dans l'ID Microsoft Entra. Avec l'authentification Microsoft Entra, vous pouvez gérer les identités des utilisateurs de base de données et d'autres services Microsoft dans un emplacement centralisé, ce qui simplifie la gestion des autorisations.

L'utilisation de l'ID Microsoft Entra pour l'authentification offre les avantages suivants :

- Authentification des utilisateurs dans les services Azure de manière uniforme.
- Gestion des stratégies de mot de passe et de la rotation des mots de passe à un seul endroit.
- Plusieurs formes d'authentification prises en charge par l'ID Microsoft Entra, qui peuvent éliminer la nécessité de stocker les mots de passe.
- Les clients peuvent gérer les autorisations de base de données à l'aide de groupes externes (Microsoft Entra ID).

- L'authentification Microsoft Entra utilise les utilisateurs de base de données MySQL pour authentifier les identités au niveau de la base de données.
- Prise en charge de l'authentification basée sur des jetons pour les applications qui se connectent à Azure Database pour MySQL.

Authentification MySQL

Vous pouvez créer des comptes dans MySQL. Si vous choisissez d'utiliser des mots de passe comme informations d'identification pour les comptes, ces informations d'identification sont stockées dans la table `user`. Étant donné que ces mots de passe sont stockés dans MySQL, vous devez gérer la rotation des mots de passe par vous-même.

Bien qu'il soit possible de se connecter à Azure Database pour MySQL avec des mots de passe, vous devez les utiliser avec précaution. Vous devez être vigilant pour ne jamais exposer les mots de passe dans un emplacement non sécurisé. Toute personne qui accède aux mots de passe est en mesure de s'authentifier. Par exemple, il existe un risque qu'un utilisateur malveillant puisse accéder à l'application si un chaîne de connexion est accidentellement case activée dans le contrôle de code source, envoyé via un e-mail non sécurisé, collé dans une conversation incorrecte ou consulté par une personne qui ne doit pas avoir d'autorisation. Au lieu de cela, envisagez de mettre à jour votre application pour utiliser des connexions sans mot de passe.

Présentation des connexions sans mot de passe

Avec une connexion sans mot de passe, vous pouvez vous connecter aux services Azure sans stocker d'informations d'identification dans le code de l'application, ses fichiers de configuration ou dans les variables d'environnement.

De nombreux services Azure prennent en charge les connexions sans mot de passe, par exemple via Azure Managed Identity. Ces techniques fournissent des fonctionnalités de sécurité robustes que vous pouvez implémenter à l'aide [de DefaultAzureCredential](#) à partir des bibliothèques clientes Azure Identity. Dans ce tutoriel, vous allez apprendre à mettre à jour une application existante à utiliser `DefaultAzureCredential` au lieu d'alternatives telles que des chaîne de connexion.

`DefaultAzureCredential` prend en charge plusieurs méthodes d'authentification et détermine automatiquement celle qui doit être utilisée au moment de l'exécution. Cette approche permet à votre application d'utiliser différentes méthodes d'authentification dans différents environnements (développement local et production) sans implémenter de code spécifique à l'environnement.

L'ordre et les emplacements dans lesquels `DefaultAzureCredential` les recherches d'informations d'identification sont disponibles dans la vue d'ensemble [de la bibliothèque d'identités Azure](#). Par exemple, lorsque vous travaillez localement, `DefaultAzureCredential` s'authentifie généralement à l'aide du compte utilisé par le développeur pour se connecter à Visual Studio. Lorsque l'application est déployée sur Azure, `DefaultAzureCredential` bascule automatiquement pour utiliser une [identité managée](#). Aucune modification du code n'est requise pour cette transition.

Pour vous assurer que les connexions sont sans mot de passe, vous devez prendre en compte le développement local et l'environnement de production. Si une chaîne de connexion est requise à l'un ou l'autre endroit, l'application n'est pas sans mot de passe.

Dans votre environnement de développement local, vous pouvez vous authentifier auprès d'Azure CLI, d'Azure PowerShell, de Visual Studio ou de plug-ins Azure pour Visual Studio Code ou IntelliJ. Dans ce cas, vous pouvez utiliser ces informations d'identification dans votre application au lieu de configurer des propriétés.

Lorsque vous déployez des applications dans un environnement d'hébergement Azure, tel qu'une machine virtuelle, vous pouvez affecter une identité managée dans cet environnement. Ensuite, vous n'avez pas besoin de fournir des informations d'identification pour vous connecter aux services Azure.

ⓘ Notes

Une identité managée fournit une identité de sécurité pour représenter une application ou un service. Managée par la plateforme Azure, l'identité ne nécessite pas que vous approvisionniez ou permutiez de secrets. Vous pouvez en savoir plus sur les identités managées dans la documentation [vue d'ensemble](#).

Migrer une application existante pour utiliser des connexions sans mot de passe

Les étapes suivantes expliquent comment migrer une application existante pour utiliser des connexions sans mot de passe au lieu d'une solution basée sur un mot de passe.

0) Préparer l'environnement de travail

Tout d'abord, utilisez la commande suivante pour configurer certaines variables d'environnement.

Bash

```
export AZ_RESOURCE_GROUP=<YOUR_RESOURCE_GROUP>
export AZ_DATABASE_SERVER_NAME=<YOUR_DATABASE_SERVER_NAME>
export AZ_DATABASE_NAME=demo
export AZ_MYSQL_AD_NON_ADMIN_USERNAME=
<YOUR_AZURE_AD_NON_ADMIN_USER_DISPLAY_NAME>
export AZ_MYSQL_AD_MI_USERNAME=<YOUR_AZURE_AD_MI_DISPLAY_NAME>
export AZ_USER_IDENTITY_NAME=<YOUR_USER_ASSIGNED_MANAGEMED_IDENTITY_NAME>
export CURRENT_USERNAME=$(az ad signed-in-user show --query
userPrincipalName --output tsv)
export CURRENT_USER_OBJECTID=$(az ad signed-in-user show --query id --output
tsv)
```

Remplacez les espaces réservés par les valeurs suivantes, qui sont utilisées dans cet article :

- <YOUR_RESOURCE_GROUP> : nom du groupe de ressources dans lequel se trouvent vos ressources.
- <YOUR_DATABASE_SERVER_NAME> : nom de votre serveur MySQL, qui doit être unique dans Azure.
- <YOUR_AZURE_AD_NON_ADMIN_USER_DISPLAY_NAME> : nom complet de votre utilisateur non administrateur Microsoft Entra. Vérifiez que le nom est un utilisateur valide dans votre locataire Microsoft Entra.
- <YOUR_AZURE_AD_MI_DISPLAY_NAME> : nom complet de l'utilisateur Microsoft Entra pour votre identité managée. Vérifiez que le nom est un utilisateur valide dans votre locataire Microsoft Entra.
- <YOUR_USER_ASSIGNED_MANAGEMED_IDENTITY_NAME> : nom de votre serveur d'identité managée affectée par l'utilisateur, qui doit être unique dans Azure.

1) Configurer Azure Database pour MySQL

1.1) Activer l'authentification basée sur l'ID Microsoft Entra

Pour utiliser l'accès à l'ID Microsoft Entra avec Azure Database pour MySQL, vous devez d'abord définir l'utilisateur administrateur Microsoft Entra. Seul un utilisateur Microsoft Entra Administration peut créer/activer des utilisateurs pour l'authentification basée sur l'ID Microsoft Entra.

Si vous utilisez Azure CLI, exécutez la commande suivante pour vous assurer de disposer d'une autorisation suffisante :

Bash

```
az login --scope https://graph.microsoft.com/.default
```

Exécutez la commande suivante pour créer une identité utilisateur pour l'affectation :

Azure CLI

```
az identity create \
--resource-group $AZ_RESOURCE_GROUP \
--name $AZ_USER_IDENTITY_NAME
```

ⓘ Important

Après avoir créé l'identité affectée par l'utilisateur, demandez à votre *Administrateur général* ou *Administrateur de rôle privilégié* d'accorder les autorisations suivantes pour cette identité : `User.Read.All`, `GroupMember.Read.All` et `Application.Read.ALL`. Pour plus d'informations, consultez la section [Autorisations d'Authentification Active Directory](#).

Exécutez la commande suivante pour affecter l'identité au serveur MySQL pour créer l'administrateur Microsoft Entra :

Azure CLI

```
az mysql flexible-server identity assign \
--resource-group $AZ_RESOURCE_GROUP \
--server-name $AZ_DATABASE_SERVER_NAME \
--identity $AZ_USER_IDENTITY_NAME
```

Exécutez ensuite la commande suivante pour définir l'administrateur Microsoft Entra :

Azure CLI

```
az mysql flexible-server ad-admin create \
--resource-group $AZ_RESOURCE_GROUP \
--server-name $AZ_DATABASE_SERVER_NAME \
--display-name $CURRENT_USERNAME \
--object-id $CURRENT_USER_OBJECTID \
--identity $AZ_USER_IDENTITY_NAME
```

Cette commande définit l'administrateur Microsoft Entra sur l'utilisateur connecté actuel.

ⓘ Notes

Vous ne pouvez créer qu'un seul administrateur Microsoft Entra par serveur MySQL. La sélection d'un autre remplacera l'administrateur Microsoft Entra existant configuré pour le serveur.

2) Configurer Azure Database pour MySQL pour le développement local

2.1) Configurer une règle de pare-feu pour l'adresse IP locale

Les instances Azure Database pour MySQL sont sécurisées par défaut. Elles ont un pare-feu qui n'autorise aucune connexion entrante.

Youvez ignorer cette étape si vous utilisez Bash, car la commande `flexible-server create` a déjà détecté votre adresse IP locale et l'a définie sur le serveur MySQL.

Si vous vous connectez à votre serveur MySQL à partir de Sous-système Windows pour Linux (WSL) sur un ordinateur Windows, vous devez ajouter l'ID d'hôte WSL à votre pare-feu. Obtenez l'adresse IP de votre ordinateur hôte en exécutant la commande suivante dans WSL :

```
Bash
```

```
cat /etc/resolv.conf
```

Copiez l'adresse IP suivant le terme `nameserver`, puis utilisez la commande suivante pour définir une variable d'environnement pour l'adresse IP WSL :

```
Bash
```

```
export AZ_WSL_IP_ADDRESS=<the-copied-IP-address>
```

Ensuite, utilisez la commande suivante pour ouvrir le pare-feu du serveur sur votre application WSL :

```
Azure CLI
```

```
az mysql server firewall-rule create \
--resource-group $AZ_RESOURCE_GROUP \
--name $AZ_DATABASE_SERVER_NAME-database-allow-local-ip-wsl \
--server $AZ_DATABASE_SERVER_NAME \
--start-ip-address $AZ_WSL_IP_ADDRESS \
```

```
--end-ip-address $AZ_WSL_IP_ADDRESS \
--output tsv
```

2.2) Créer un utilisateur non administrateur MySQL et accorder une autorisation

Ensuite, créez un utilisateur Microsoft Entra non administrateur et accordez-lui toutes les autorisations sur la `$AZ_DATABASE_NAME` base de données. Vous pouvez modifier le nom `$AZ_DATABASE_NAME` de la base de données en fonction de vos besoins.

Créez un script SQL appelé `create_ad_user.sql` pour créer un utilisateur non administrateur. Ajoutez le contenu suivant et enregistrez-le localement :

Bash

```
export AZ_MYSQL_AD_NON_ADMIN_USERID=$(az ad signed-in-user show --query id \
--output tsv)

cat << EOF > create_ad_user.sql
SET aad_auth_validate_oids_in_tenant = OFF;
CREATE AADUSER '$AZ_MYSQL_AD_NON_ADMIN_USERNAME' IDENTIFIED BY
'$AZ_MYSQL_AD_NON_ADMIN_USERID';
GRANT ALL PRIVILEGES ON $AZ_DATABASE_NAME.* TO
'$AZ_MYSQL_AD_NON_ADMIN_USERNAME'@'%';
FLUSH privileges;
EOF
```

Utilisez ensuite la commande suivante pour exécuter le script SQL pour créer l'utilisateur non administrateur Microsoft Entra :

Bash

```
mysql -h $AZ_DATABASE_SERVER_NAME.mysql.database.azure.com --user
$CURRENT_USERNAME --enable-cleartext-plugin --password=$(az account get-
access-token --resource-type oss-rdbms --output tsv --query accessToken) <
create_ad_user.sql
```

Utilisez maintenant la commande suivante pour supprimer le fichier de script SQL temporaire :

Bash

```
rm create_ad_user.sql
```

ⓘ Notes

Vous pouvez lire des informations plus détaillées sur la création d'utilisateurs MySQL dans [Créer des utilisateurs dans Azure Database pour MySQL](#).

3) Connectez-vous et migrez le code de l'application pour utiliser des connexions sans mot de passe

Pour le développement local, vérifiez que vous êtes authentifié avec le même compte Microsoft Entra auquel vous avez affecté le rôle sur votre MySQL. Vous pouvez vous authentifier via Azure CLI, Visual Studio, Azure PowerShell ou d'autres outils tels qu'IntelliJ.

Azure CLI

Connectez-vous à Azure via Azure CLI à l'aide de la commande suivante :

Azure CLI

```
az login
```

Ensuite, procédez comme suit pour mettre à jour votre code pour utiliser des connexions sans mot de passe. Bien que conceptuellement similaire, chaque langage utilise des détails d'implémentation différents.

Java

1. Dans votre projet, ajoutez la référence suivante au `azure-identity-extensio`n package. Cette bibliothèque contient toutes les entités nécessaires pour implémenter des connexions sans mot de passe.

XML

```
<dependency>
    <groupId>com.azure</groupId>
    <artifactId>azure-identity-extensio</artifactId>
    <version>1.0.0</version>
</dependency>
```

2. Activez le plug-in d'authentification Azure MySQL dans l'URL JDBC. Identifiez les emplacements de votre code qui créent actuellement une `java.sql.Connection` connexion à Azure Database pour MySQL. Mettez à jour `url` et dans votre *fichier application.properties* pour qu'il corresponde `user` aux valeurs suivantes :

properties

```
url=jdbc:mysql://$AZ_DATABASE_SERVER_NAME.mysql.database.azure.com:  
3306/$AZ_DATABASE_NAME?  
serverTimezone=UTC&sslMode=REQUIRED&defaultAuthenticationPlugin=com  
.azure.identity.extensions.jdbc.mysql.AzureMysqlAuthenticationPlugi  
n&authenticationPlugins=com.azure.identity.extensions.jdbc.mysql.Az  
ureMysqlAuthenticationPlugin  
user=$AZ_MYSQL_AD_NON_ADMIN_USERNAME
```

① Notes

Si vous utilisez la `MysqlConnectionPoolDataSource` classe comme source de données dans votre application, veillez à supprimer `defaultAuthenticationPlugin=com.azure.identity.extensions.jdbc.mysql.`
`AzureMysqlAuthenticationPlugin` de l'URL.

properties

```
url=jdbc:mysql://$AZ_DATABASE_SERVER_NAME.mysql.database.azure.com:  
3306/$AZ_DATABASE_NAME?  
serverTimezone=UTC&sslMode=REQUIRED&authenticationPlugins=com.azure  
.identity.extensions.jdbc.mysql.AzureMysqlAuthenticationPlugin  
user=$AZ_MYSQL_AD_NON_ADMIN_USERNAME
```

3. Remplacez la variable, `$AZ_DATABASE_SERVER_NAME` une `$AZ_DATABASE_NAME` variable et une `$AZ_MYSQL_AD_NON_ADMIN_USERNAME` variable par les valeurs que vous avez configurées au début de cet article.

4. Supprimez l'URL `password` JDBC.

Exécutez l'application localement.

Après avoir apporté ces modifications de code, exécutez votre application localement. La nouvelle configuration doit récupérer vos informations d'identification locales si vous

êtes connecté à un outil en ligne de commande ou IDE compatible, tel qu’Azure CLI, Visual Studio ou IntelliJ. Les rôles que vous avez attribués à votre utilisateur de développement local dans Azure permettent à votre application de se connecter au service Azure localement.

4) Configurer l’environnement d’hébergement Azure

Une fois que votre application est configurée pour utiliser des connexions sans mot de passe et qu’elle s’exécute localement, le même code peut s’authentifier auprès des services Azure après son déploiement sur Azure. Par exemple, une application déployée sur une instance Azure App Service qui a une identité managée affectée peut se connecter à Stockage Azure.

Dans cette section, vous allez exécuter deux étapes pour permettre à votre application de s’exécuter dans un environnement d’hébergement Azure de manière sans mot de passe :

- Attribuez l’identité managée pour votre environnement d’hébergement Azure.
- Attribuez des rôles à l’identité managée.

ⓘ Notes

Azure fournit également le **Connecter or de service**, qui peut vous aider à connecter votre service d’hébergement avec PostgreSQL. Avec service Connecter or pour configurer votre environnement d’hébergement, vous pouvez omettre l’étape d’attribution de rôles à votre identité managée, car Le Connecter or de service le fera pour vous. La section suivante explique comment configurer votre environnement d’hébergement Azure de deux façons : l’une via service Connecter or et l’autre en configurant directement chaque environnement d’hébergement.

ⓘ Important

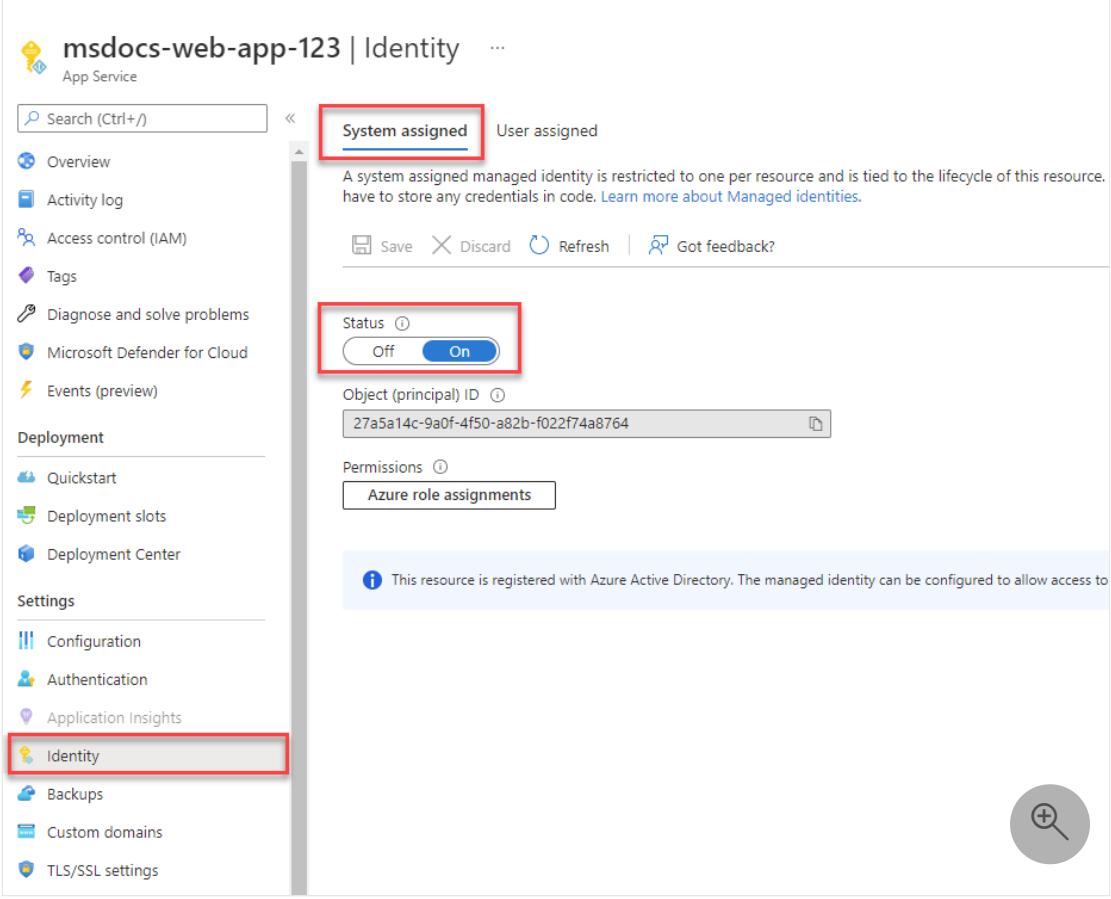
Les commandes du Connecter or de service nécessitent **Azure CLI** 2.41.0 ou version ultérieure.

Attribuer l’identité managée à l’aide du Portail Azure

Les étapes suivantes vous montrent comment attribuer une identité managée affectée par le système pour différents services d’hébergement web. L’identité managée peut se

connecter en toute sécurité à d'autres services Azure à l'aide des configurations d'application que vous avez configurées.

App Service



1. Dans la page de vue d'ensemble principale de votre instance Azure App Service, sélectionnez **Identité** dans le volet de navigation.

2. Sous l'onglet Affecté par le système, veillez à définir le champ État sur activé. Une identité affectée par le système est gérée par Azure en interne et gère les tâches d'administration pour vous. Les détails et ID de l'identité ne sont jamais exposés dans votre code.

Vous pouvez également affecter une identité managée sur un environnement d'hébergement Azure à l'aide d'Azure CLI.

App Service

Vous pouvez affecter une identité managée à une instance Azure App Service avec la commande `az webapp identity assign`, comme illustré dans l'exemple suivant :

Azure CLI

```
export AZ_MI_OBJECT_ID=$(az webapp identity assign \
    --resource-group $AZ_RESOURCE_GROUP \
    --name <service-instance-name> \
    --query principalId \
    --output tsv)
```

Attribuer des rôles à l'identité managée

Ensuite, accordez des autorisations à l'identité managée que vous avez affectée pour accéder à votre instance MySQL.

Ces étapes créent un utilisateur Microsoft Entra pour l'identité managée et lui accordent toutes les autorisations pour la base de données `$AZ_DATABASE_NAME`. Vous pouvez modifier le nom `$AZ_DATABASE_NAME` de la base de données en fonction de vos besoins.

Tout d'abord, créez un script SQL appelé *create_ad_user.sql* pour créer un utilisateur non administrateur. Ajoutez le contenu suivant et enregistrez-le localement :

Bash

```
export AZ_MYSQL_AD_MI_USERID=$(az ad sp show --id $AZ_MI_OBJECT_ID --query
appId --output tsv)

cat << EOF > create_ad_user.sql
SET aad_auth_validate_oids_in_tenant = OFF;
CREATE AADUSER '$AZ_MYSQL_AD_MI_USERNAME' IDENTIFIED BY
'$AZ_MYSQL_AD_MI_USERID';
GRANT ALL PRIVILEGES ON $AZ_DATABASE_NAME.* TO
'$AZ_MYSQL_AD_MI_USERNAME'@'%';
FLUSH privileges;
EOF
```

Utilisez ensuite la commande suivante pour exécuter le script SQL pour créer l'utilisateur non administrateur Microsoft Entra :

Bash

```
mysql -h $AZ_DATABASE_SERVER_NAME.mysql.database.azure.com --user
$CURRENT_USERNAME --enable-cleartext-plugin --password=$(az account get-
access-token --resource-type oss-rdbms --output tsv --query accessToken) <
create_ad_user.sql
```

Utilisez maintenant la commande suivante pour supprimer le fichier de script SQL temporaire :

Bash

```
rm create_ad_user.sql
```

Test de l'application

Avant de déployer l'application dans l'environnement d'hébergement, vous devez apporter une modification supplémentaire au code, car l'application va se connecter à MySQL à l'aide de l'utilisateur créé pour l'identité managée.

Java

Mettez à jour votre code pour utiliser l'utilisateur créé pour l'identité managée :

Java

```
properties.put("user", "$AZ_MYSQL_AD_MI_USERNAME");
```

Après avoir apporté ces modifications de code, vous pouvez générer et redéployer l'application. Ensuite, accédez à votre application hébergée dans le navigateur. Votre application doit être en mesure de se connecter à la base de données MySQL avec succès. N'oubliez pas qu'il peut falloir plusieurs minutes pour que les attributions de rôle se propagent dans l'environnement Azure. Votre application est désormais configurée pour s'exécuter localement et dans un environnement de production sans que les développeurs n'aient à gérer les secrets dans l'application elle-même.

Étapes suivantes

Dans ce didacticiel, vous avez appris à migrer une application vers des connexions sans mot de passe.

Vous pouvez lire les ressources suivantes pour explorer les concepts abordés dans cet article plus en détails :

- [Autoriser l'accès aux données d'objet blob avec des identités managées pour les ressources Azure](#).
- [Autoriser l'accès aux objets blob à l'aide Microsoft Entra ID](#)

Migrer une application pour utiliser des connexions sans mot de passe avec Azure Database pour PostgreSQL

Article • 20/10/2023

Cet article explique comment migrer des méthodes d'authentification traditionnelles vers des connexions sans mot de passe plus sécurisées avec Azure Database pour PostgreSQL.

Les demandes d'application à Azure Database pour PostgreSQL doivent être authentifiées. Azure Database pour PostgreSQL fournit plusieurs façons différentes pour les applications de se connecter en toute sécurité. L'une des façons d'utiliser des mots de passe. Toutefois, vous devez hiérarchiser les connexions sans mot de passe dans vos applications lorsque cela est possible.

Comparer les options d'authentification

Lorsque l'application s'authentifie avec Azure Database pour PostgreSQL, elle fournit une paire nom d'utilisateur et mot de passe pour connecter la base de données. Selon l'emplacement de stockage des identités, il existe deux types d'authentification : l'authentification Microsoft Entra et l'authentification PostgreSQL.

Authentification Microsoft Entra

L'authentification Microsoft Entra est un mécanisme de connexion à Azure Database pour PostgreSQL utilisant les identités définies dans Microsoft Entra ID. Avec l'authentification Microsoft Entra, vous pouvez gérer les identités des utilisateurs de base de données et d'autres services Microsoft dans un emplacement centralisé, ce qui simplifie la gestion des autorisations.

L'utilisation de l'ID Microsoft Entra pour l'authentification offre les avantages suivants :

- Authentification des utilisateurs dans les services Azure de manière uniforme.
- Gestion des stratégies de mot de passe et de la rotation des mots de passe à un seul endroit.
- Plusieurs formes d'authentification prises en charge par l'ID Microsoft Entra, qui peuvent éliminer la nécessité de stocker les mots de passe.
- Les clients peuvent gérer les autorisations de base de données à l'aide de groupes externes (Microsoft Entra ID).

- L'authentification Microsoft Entra utilise les utilisateurs de base de données PostgreSQL pour authentifier les identités au niveau de la base de données.
- Prise en charge de l'authentification basée sur les jetons pour les applications qui se connectent à Azure Database pour PostgreSQL.

Authentification PostgreSQL

Vous pouvez créer des comptes dans PostgreSQL. Si vous choisissez d'utiliser des mots de passe comme informations d'identification pour les comptes, ces informations d'identification sont stockées dans la table `user`. Étant donné que ces mots de passe sont stockés dans PostgreSQL, vous devez gérer la rotation des mots de passe par vous-même.

Bien qu'il soit possible de se connecter à Azure Database pour PostgreSQL avec des mots de passe, vous devez les utiliser avec précaution. Vous devez être vigilant pour ne jamais exposer les mots de passe dans un emplacement non sécurisé. Toute personne qui accède aux mots de passe est en mesure de s'authentifier. Par exemple, il existe un risque qu'un utilisateur malveillant puisse accéder à l'application si un chaîne de connexion est accidentellement case activée dans le contrôle de code source, envoyé via un e-mail non sécurisé, collé dans une conversation incorrecte ou consulté par une personne qui ne doit pas avoir d'autorisation. Au lieu de cela, envisagez de mettre à jour votre application pour utiliser des connexions sans mot de passe.

Présentation des connexions sans mot de passe

Avec une connexion sans mot de passe, vous pouvez vous connecter aux services Azure sans stocker d'informations d'identification dans le code de l'application, ses fichiers de configuration ou dans les variables d'environnement.

De nombreux services Azure prennent en charge les connexions sans mot de passe, par exemple via Azure Managed Identity. Ces techniques fournissent des fonctionnalités de sécurité robustes que vous pouvez implémenter à l'aide [de DefaultAzureCredential](#) à partir des bibliothèques clientes Azure Identity. Dans ce tutoriel, vous allez apprendre à mettre à jour une application existante à utiliser `DefaultAzureCredential` au lieu d'alternatives telles que des chaîne de connexion.

`DefaultAzureCredential` prend en charge plusieurs méthodes d'authentification et détermine automatiquement celle qui doit être utilisée au moment de l'exécution. Cette approche permet à votre application d'utiliser différentes méthodes d'authentification dans différents environnements (développement local et production) sans implémenter de code spécifique à l'environnement.

L'ordre et les emplacements dans lesquels `DefaultAzureCredential` les recherches d'informations d'identification sont disponibles dans la vue d'ensemble [de la bibliothèque d'identités Azure](#). Par exemple, lorsque vous travaillez localement, `DefaultAzureCredential` s'authentifie généralement à l'aide du compte utilisé par le développeur pour se connecter à Visual Studio. Lorsque l'application est déployée sur Azure, `DefaultAzureCredential` bascule automatiquement pour utiliser une [identité managée](#). Aucune modification du code n'est requise pour cette transition.

Pour vous assurer que les connexions sont sans mot de passe, vous devez prendre en compte le développement local et l'environnement de production. Si une chaîne de connexion est requise à l'un ou l'autre endroit, l'application n'est pas sans mot de passe.

Dans votre environnement de développement local, vous pouvez vous authentifier auprès d'Azure CLI, d'Azure PowerShell, de Visual Studio ou de plug-ins Azure pour Visual Studio Code ou IntelliJ. Dans ce cas, vous pouvez utiliser ces informations d'identification dans votre application au lieu de configurer des propriétés.

Lorsque vous déployez des applications dans un environnement d'hébergement Azure, tel qu'une machine virtuelle, vous pouvez affecter une identité managée dans cet environnement. Ensuite, vous n'avez pas besoin de fournir des informations d'identification pour vous connecter aux services Azure.

ⓘ Notes

Une identité managée fournit une identité de sécurité pour représenter une application ou un service. Managée par la plateforme Azure, l'identité ne nécessite pas que vous approvisionniez ou permutiez de secrets. Vous pouvez en savoir plus sur les identités managées dans la documentation [vue d'ensemble](#).

Migrer une application existante pour utiliser des connexions sans mot de passe

Les étapes suivantes expliquent comment migrer une application existante pour utiliser des connexions sans mot de passe au lieu d'une solution basée sur un mot de passe.

0) Préparer l'environnement de travail

Tout d'abord, utilisez la commande suivante pour configurer certaines variables d'environnement.

Bash

```
export AZ_RESOURCE_GROUP=<YOUR_RESOURCE_GROUP>
export AZ_DATABASE_SERVER_NAME=<YOUR_DATABASE_SERVER_NAME>
export AZ_DATABASE_NAME=demo
export AZ_POSTGRESQL_AD_NON_ADMIN_USERNAME=
<YOUR_AZURE_AD_NON_ADMIN_USER_DISPLAY_NAME>
export AZ_LOCAL_IP_ADDRESS=<YOUR_LOCAL_IP_ADDRESS>
export CURRENT_USERNAME=$(az ad signed-in-user show --query
userPrincipalName --output tsv)
```

Remplacez les espaces réservés par les valeurs suivantes, qui sont utilisées dans cet article :

- <YOUR_RESOURCE_GROUP> : nom du groupe de ressources dans lequel se trouvent vos ressources.
- <YOUR_DATABASE_SERVER_NAME> : nom de votre serveur PostgreSQL. Il doit être unique dans tout Azure.
- <YOUR_AZURE_AD_NON_ADMIN_USER_DISPLAY_NAME> : nom complet de votre utilisateur non administrateur Microsoft Entra. Vérifiez que le nom est un utilisateur valide dans votre locataire Microsoft Entra.
- <YOUR_LOCAL_IP_ADDRESS> : adresse IP de votre ordinateur local, à partir duquel vous exécuterez l'application Spring Boot. Un moyen pratique de le trouver est d'ouvrir whatismyip.akamai.com.

1) Configurer Azure Database pour PostgreSQL

1.1) Activer l'authentification basée sur l'ID Microsoft Entra

Pour utiliser l'accès à l'ID Microsoft Entra avec Azure Database pour PostgreSQL, vous devez d'abord définir l'utilisateur administrateur Microsoft Entra. Seul un utilisateur Microsoft Entra Administration peut créer/activer des utilisateurs pour l'authentification basée sur l'ID Microsoft Entra.

Pour configurer un administrateur Microsoft Entra après avoir créé le serveur, suivez les étapes décrites dans [Gérer les rôles Microsoft Entra dans Azure Database pour PostgreSQL - Serveur](#) flexible.

ⓘ Notes

Le serveur flexible PostgreSQL peut créer plusieurs administrateurs Microsoft Entra.

2) Configurer Azure Database pour PostgreSQL pour le développement local

2.1) Configurer une règle de pare-feu pour l'adresse IP locale

Les instances Azure Database pour PostgreSQL sont sécurisées par défaut. Elles ont un pare-feu qui n'autorise aucune connexion entrante. Pour pouvoir utiliser notre base de données, vous devez ajouter une règle de pare-feu qui permet à l'adresse IP locale d'accéder au serveur de base de données.

Comme vous avez configuré votre adresse IP locale au début de cet article, vous pouvez ouvrir le pare-feu du serveur en exécutant la commande suivante :

Azure CLI

```
az postgres flexible-server firewall-rule create \
--resource-group $AZ_RESOURCE_GROUP \
--name $AZ_DATABASE_SERVER_NAME \
--rule-name $AZ_DATABASE_SERVER_NAME-database-allow-local-ip \
--start-ip-address $AZ_LOCAL_IP_ADDRESS \
--end-ip-address $AZ_LOCAL_IP_ADDRESS \
--output tsv
```

Si vous vous connectez à votre serveur PostgreSQL à partir de Sous-système Windows pour Linux (WSL) sur un ordinateur Windows, vous devez ajouter l'ID d'hôte WSL à votre pare-feu.

Obtenez l'adresse IP de votre ordinateur hôte en exécutant la commande suivante dans WSL :

Bash

```
cat /etc/resolv.conf
```

Copiez l'adresse IP suivant le terme `nameserver`, puis utilisez la commande suivante pour définir une variable d'environnement pour l'adresse IP de WSL :

Bash

```
export AZ_WSL_IP_ADDRESS=<the-copied-IP-address>
```

Ensuite, utilisez la commande suivante pour ouvrir le pare-feu du serveur sur votre application WSL :

Azure CLI

```
az postgres flexible-server firewall-rule create \
--resource-group $AZ_RESOURCE_GROUP \
--name $AZ_DATABASE_SERVER_NAME \
--rule-name $AZ_DATABASE_SERVER_NAME-database-allow-local-ip \
--start-ip-address $AZ_WSL_IP_ADDRESS \
--end-ip-address $AZ_WSL_IP_ADDRESS \
--output tsv
```

2.2) Créer un utilisateur non administrateur PostgreSQL et accorder une autorisation

Ensuite, créez un utilisateur Microsoft Entra non administrateur et accordez-lui toutes les autorisations sur la `$AZ_DATABASE_NAME` base de données. Vous pouvez modifier le nom `$AZ_DATABASE_NAME` de la base de données en fonction de vos besoins.

Créez un script SQL appelé `create_ad_user_local.sql` pour créer un utilisateur non administrateur. Ajoutez le contenu suivant et enregistrez-le localement :

Bash

```
cat << EOF > create_ad_user_local.sql
select * from
pgaadauth_create_principal('$AZ_POSTGRESQL_AD_NON_ADMIN_USERNAME', false,
false);
EOF
```

Utilisez ensuite la commande suivante pour exécuter le script SQL pour créer l'utilisateur non administrateur Microsoft Entra :

Bash

```
psql "host=$AZ_DATABASE_SERVER_NAME.postgres.database.azure.com
user=$CURRENT_USERNAME dbname=postgres port=5432 password=$(az account get-
access-token --resource-type oss-rdbms --output tsv --query accessToken)
sslmode=require" < create_ad_user_local.sql
```

Utilisez maintenant la commande suivante pour supprimer le fichier de script SQL temporaire :

Bash

```
rm create_ad_user_local.sql
```

ⓘ Notes

Vous pouvez lire des informations plus détaillées sur la création d'utilisateurs PostgreSQL dans [Créer des utilisateurs dans Azure Database pour PostgreSQL](#).

3) Connectez-vous et migrez le code de l'application pour utiliser des connexions sans mot de passe

Pour le développement local, vérifiez que vous êtes authentifié avec le même compte Microsoft Entra auquel vous avez affecté le rôle sur votre PostgreSQL. Vous pouvez vous authentifier via Azure CLI, Visual Studio, Azure PowerShell ou d'autres outils tels qu'IntelliJ.

Azure CLI

Connectez-vous à Azure via Azure CLI à l'aide de la commande suivante :

Azure CLI

```
az login
```

Ensuite, procédez comme suit pour mettre à jour votre code pour utiliser des connexions sans mot de passe. Bien que conceptuellement similaire, chaque langage utilise des détails d'implémentation différents.

Java

1. Dans votre projet, ajoutez la référence suivante au `azure-identity-extensio`n package. Cette bibliothèque contient toutes les entités nécessaires pour implémenter des connexions sans mot de passe.

XML

```
<dependency>
    <groupId>com.azure</groupId>
    <artifactId>azure-identity-extensio</artifactId>
    <version>1.0.0</version>
</dependency>
```

2. Activez le plug-in d'authentification Azure PostgreSQL dans l'URL JDBC.

Identifiez les emplacements de votre code qui créent actuellement une `java.sql.Connection` connexion à Azure Database pour PostgreSQL. Mettez à jour `url` et dans votre *fichier application.properties* pour qu'il corresponde `user` aux valeurs suivantes :

properties

```
url=jdbc:postgresql://$AZ_DATABASE_SERVER_NAME.postgres.database.azure.com:5432/$AZ_DATABASE_NAME?  
sslmode=require&authenticationPluginClassName=com.azure.identity.extensions.jdbc.postgresql.AzurePostgresqlAuthenticationPlugin  
user=$AZ_POSTGRESQL_AD_NON_ADMIN_USERNAME
```

3. Remplacez les `$AZ_POSTGRESQL_AD_NON_ADMIN_USERNAME` deux `$AZ_DATABASE_SERVER_NAME` variables par la valeur que vous avez configurée au début de cet article.

Exécuter l'application localement.

Après avoir apporté ces modifications de code, exécutez votre application localement. La nouvelle configuration doit récupérer vos informations d'identification locales si vous êtes connecté à un outil en ligne de commande ou IDE compatible, tel qu'Azure CLI, Visual Studio ou IntelliJ. Les rôles que vous avez attribués à votre utilisateur de développement local dans Azure permettent à votre application de se connecter au service Azure localement.

4) Configurer l'environnement d'hébergement Azure

Une fois que votre application est configurée pour utiliser des connexions sans mot de passe et qu'elle s'exécute localement, le même code peut s'authentifier auprès des services Azure après son déploiement sur Azure. Par exemple, une application déployée sur une instance Azure App Service qui a une identité managée affectée peut se connecter à Stockage Azure.

Dans cette section, vous allez exécuter deux étapes pour permettre à votre application de s'exécuter dans un environnement d'hébergement Azure de manière sans mot de passe :

- Attribuez l'identité managée pour votre environnement d'hébergement Azure.
- Attribuez des rôles à l'identité managée.

ⓘ Notes

Azure fournit également le **Connecter or de service**, qui peut vous aider à connecter votre service d'hébergement avec PostgreSQL. Avec service Connecter or pour configurer votre environnement d'hébergement, vous pouvez omettre l'étape d'attribution de rôles à votre identité managée, car Le Connecter or de service le fera pour vous. La section suivante explique comment configurer votre environnement d'hébergement Azure de deux façons : l'une via service Connecter or et l'autre en configurant directement chaque environnement d'hébergement.

ⓘ Important

Les commandes du Connecter or de service nécessitent **Azure CLI** 2.41.0 ou version ultérieure.

Attribuer l'identité managée à l'aide du Portail Azure

Les étapes suivantes vous montrent comment attribuer une identité managée affectée par le système pour différents services d'hébergement web. L'identité managée peut se connecter en toute sécurité à d'autres services Azure à l'aide des configurations d'application que vous avez configurées.

App Service

1. Dans la page de vue d'ensemble principale de votre instance Azure App Service, sélectionnez **Identité** dans le volet de navigation.
2. Sous l'onglet **Affecté par le système**, veillez à définir le champ **État sur activé**. Une identité affectée par le système est gérée par Azure en interne et gère les tâches d'administration pour vous. Les détails et ID de l'identité ne sont jamais exposés dans votre code.

The screenshot shows the Azure portal interface for managing the identity of an App Service. The left sidebar lists various settings like Overview, Activity log, and Deployment. The main panel is titled 'Identity' under 'Settings'. It shows two tabs: 'System assigned' (selected) and 'User assigned'. A note states: 'A system assigned managed identity is restricted to one per resource and is tied to the lifecycle of this resource. Have to store any credentials in code. Learn more about Managed identities.' Below this are buttons for Save, Discard, Refresh, and Got feedback? A status switch is set to 'On'. The object principal ID is listed as 27a5a14c-9a0f-4f50-a82b-f022f74a8764. A permissions section shows 'Azure role assignments'. A note at the bottom says: 'This resource is registered with Azure Active Directory. The managed identity can be configured to allow access to'.

Vous pouvez également affecter une identité managée sur un environnement d'hébergement Azure à l'aide d'Azure CLI.

App Service

Vous pouvez affecter une identité managée à une instance Azure App Service avec la [commande az webapp identity assign](#), comme illustré dans l'exemple suivant :

Azure CLI

```
export AZ_MI_OBJECT_ID=$(az webapp identity assign \
    --resource-group $AZ_RESOURCE_GROUP \
    --name <service-instance-name> \
    --query principalId \
    --output tsv)
```

Attribuer des rôles à l'identité managée

Ensuite, accordez des autorisations à l'identité managée que vous avez affectée pour accéder à votre instance PostgreSQL.

Connecteur de service

Si vous avez connecté vos services à l'aide du service Connecter or, les commandes de l'étape précédente ont déjà attribué le rôle, afin de pouvoir ignorer cette étape.

Test de l'application

Avant de déployer l'application dans l'environnement d'hébergement, vous devez apporter une modification supplémentaire au code, car l'application va se connecter à PostgreSQL à l'aide de l'utilisateur créé pour l'identité managée.

Java

Mettez à jour votre code pour utiliser l'utilisateur créé pour l'identité managée :

ⓘ Notes

Si vous avez utilisé la commande Service Connecter or, ignorez cette étape.

Java

```
properties.put("user", "$AZ_POSTGRESQL_AD_MI_USERNAME");
```

Après avoir apporté ces modifications de code, vous pouvez générer et redéployer l'application. Ensuite, accédez à votre application hébergée dans le navigateur. Votre application doit être en mesure de se connecter à la base de données PostgreSQL avec succès. N'oubliez pas qu'il peut falloir plusieurs minutes pour que les attributions de rôle se propagent dans l'environnement Azure. Votre application est désormais configurée pour s'exécuter localement et dans un environnement de production sans que les développeurs n'aient à gérer les secrets dans l'application elle-même.

Étapes suivantes

Dans ce didacticiel, vous avez appris à migrer une application vers des connexions sans mot de passe.

Vous pouvez lire les ressources suivantes pour explorer les concepts abordés dans cet article plus en détails :

- Autoriser l'accès aux données d'objet blob avec des identités managées pour les ressources Azure.
- Autoriser l'accès aux objets blob à l'aide Microsoft Entra ID

Migrer une application pour utiliser des connexions sans mot de passe avec le Azure Service Bus

Article • 12/06/2023

Les demandes d'application adressées au Azure Service Bus doivent être authentifiées à l'aide de clés d'accès de compte ou de connexions sans mot de passe. Toutefois, vous devez hiérarchiser les connexions sans mot de passe dans vos applications lorsque cela est possible. Ce didacticiel explique comment migrer des méthodes d'authentification traditionnelles vers des connexions sans mot de passe plus sécurisées.

Risques de sécurité associés aux clés d'accès

L'exemple de code suivant montre comment se connecter à Azure Service Bus à l'aide d'une chaîne de connexion qui inclut une clé d'accès. Lorsque vous créez un Service Bus, Azure génère automatiquement ces clés et chaînes de connexion. De nombreux développeurs gravitent vers cette solution, car ils se sentent familiers avec les options qu'ils ont utilisées dans le passé. Si votre application utilise actuellement des chaînes de connexion, envisagez de migrer vers des connexions sans mot de passe à l'aide des étapes décrites dans ce document.

.NET

C#

```
await using ServiceBusClient client = new("<CONNECTION-STRING>");
```

Les chaînes de connexion doivent être utilisées avec précaution. Les développeurs doivent être vigilants pour ne jamais exposer les clés dans un emplacement non sécurisé. Toute personne ayant accès à la clé est en mesure de s'authentifier. Par exemple, si une clé de compte est accidentellement vérifiée dans le contrôle de code source, envoyée via un e-mail non sécurisé, collée dans une conversation incorrecte ou consultée par une personne qui ne doit pas avoir l'autorisation, il y a un risque d'accès malveillant à l'application. Au lieu de cela, envisagez de mettre à jour votre application pour utiliser des connexions sans mot de passe.

Migrer vers des connexions sans mot de passe

De nombreux services Azure prennent en charge les connexions sans mot de passe avec Microsoft Entra ID et le contrôle d'accès en fonction du rôle (RBAC). Ces techniques fournissent des fonctionnalités de sécurité robustes et peuvent être implémentées `DefaultAzureCredential` à partir des bibliothèques clientes Azure Identity.

ⓘ Important

Certains langages doivent être implémentés `DefaultAzureCredential` explicitement dans leur code, tandis que d'autres utilisent `DefaultAzureCredential` en interne via des plug-ins ou des pilotes sous-jacents.

`DefaultAzureCredential` prend en charge plusieurs méthodes d'authentification et détermine automatiquement celle qui doit être utilisée au moment de l'exécution. Cette approche permet à votre application d'utiliser différentes méthodes d'authentification dans différents environnements (développement local et production) sans implémenter de code spécifique à l'environnement.

L'ordre et les emplacements dans lesquels `DefaultAzureCredential` les recherche d'informations d'identification sont disponibles dans la [vue d'ensemble de la bibliothèque d'identités Azure](#) et varient d'une langue à l'autre. Par exemple, lors de l'utilisation locale avec .NET, `DefaultAzureCredential` s'authentifie généralement à l'aide du compte utilisé par le développeur pour se connecter à Visual Studio, Azure CLI, ou Azure PowerShell. Lorsque l'application est déployée sur Azure, `DefaultAzureCredential` découvrira et utilisera automatiquement [l'identité managée](#) du service d'hébergement associé, tel que Azure App Service. Aucune modification du code n'est requise pour cette transition.

! Notes

Une identité managée fournit une identité de sécurité pour représenter une application ou un service. Managée par la plateforme Azure, l'identité ne nécessite pas que vous approvisionniez ou permutiez de secrets. Vous pouvez en savoir plus sur les identités managées dans la documentation [vue d'ensemble](#).

L'exemple de code suivant montre comment se connecter à Service Bus à l'aide de connexions sans mot de passe. La section suivante décrit comment migrer vers cette configuration pour un service spécifique plus en détail.

Une application .NET peut passer une instance de `DefaultAzureCredential` au constructeur d'une classe cliente de service. `DefaultAzureCredential` détecte

automatiquement les informations d'identification disponibles dans cet environnement.

C#

```
ServiceBusClient serviceBusClient = new(
    new Uri($"https://{{serviceBusNamespace}}.blob.core.windows.net"),
    new DefaultAzureCredential());
```

Étapes de migration d'une application pour utiliser l'authentification sans mot de passe

Les étapes suivantes expliquent comment migrer une application existante pour utiliser des connexions sans mot de passe au lieu d'une solution basée sur des clés. Vous allez d'abord configurer un environnement de développement local, puis appliquer ces concepts à un environnement d'hébergement d'applications Azure. Ces mêmes étapes de migration doivent s'appliquer que vous utilisez directement des clés d'accès ou via des chaînes de connexion.

Configurer des rôles et des utilisateurs pour l'authentification de développement local

Lors du développement localement, assurez-vous que le compte d'utilisateur qui accède au Service Bus dispose des autorisations appropriées. Dans cet exemple, vous allez utiliser le rôle **propriétaire de données Azure Service Bus** pour envoyer et recevoir des données, bien que des rôles plus granulaires soient également disponibles. Pour vous attribuer ce rôle, vous aurez besoin du rôle **Administrateur de l'accès utilisateur** ou d'un autre rôle qui inclut l'action **Microsoft.Authorization/roleAssignments/write**. Vous pouvez attribuer des rôles RBAC Azure à un utilisateur à l'aide du Portail Azure, Azure CLI ou Azure PowerShell. Vous pouvez en savoir plus sur les étendues disponibles pour les attributions de rôles dans la page [vue d'ensemble de l'étendue](#).

Dans ce scénario, vous allez attribuer des autorisations limitées à un espace de noms Service Bus spécifique à votre compte d'utilisateur, conformément au [Principe des priviléges minimum](#). Cette pratique offre aux utilisateurs uniquement les autorisations minimales nécessaires et crée des environnements de production plus sécurisés.

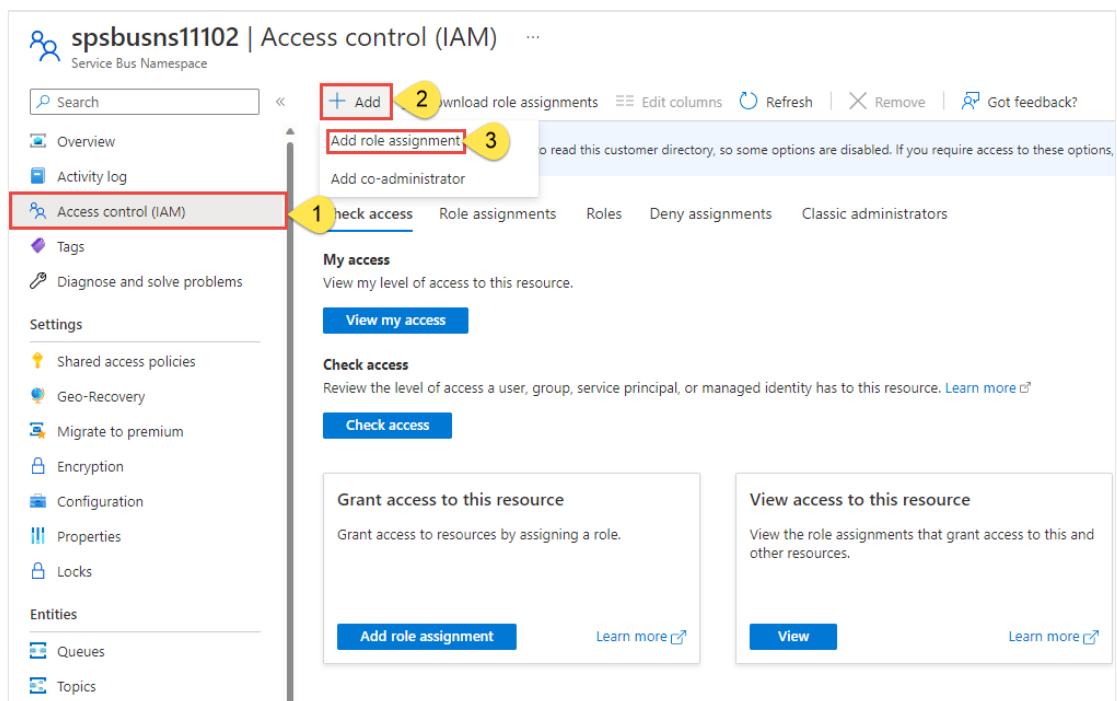
L'exemple suivant attribuera le rôle **Propriétaire de données Azure Service Bus** à votre compte d'utilisateur, ce qui vous permet d'envoyer et de recevoir des données.

 **Important**

Dans la plupart des cas, la propagation de l'attribution de rôle dans Azure peut prendre une ou deux minutes, mais dans de rares cas, cela peut prendre jusqu'à huit minutes. Si vous recevez des erreurs d'authentification lorsque vous exécutez votre code pour la première fois, patientez quelques instants et réessayez.

Azure portal

1. Dans le Portail Azure, recherchez votre espace de noms Service Bus à l'aide de la barre de recherche principale ou de la navigation gauche.
2. Dans la page de vue d'ensemble du Service Bus, sélectionnez **Contrôle d'accès (IAM)** dans le menu de gauche.
3. Sur la page **Contrôle d'accès (IAM)**, sélectionnez l'onglet **Attributions de rôles**.
4. Sélectionnez **+ Ajouter** dans le menu supérieur, puis **Ajouter une attribution de rôle** dans le menu déroulant résultant.



5. Utilisez la zone de recherche pour filtrer les résultats sur le rôle souhaité. Pour cet exemple, recherchez *Azure Service Bus Data Owner*, sélectionnez le résultat correspondant, puis choisissez **Suivant**.
6. Sous **Attribuer l'accès à**, sélectionnez **Utilisateur, groupe ou principal de service**, puis sélectionnez **+ Sélectionner des membres**.

7. Dans la boîte de dialogue, recherchez votre nom d'utilisateur Microsoft Entra (généralement votre adresse e-mail *user@domain*), puis choisissez **Selectionner** en bas de la boîte de dialogue.
8. Sélectionnez **Vérifier + affecter** pour accéder à la page finale, puis **Vérifier + attribuer** à nouveau pour terminer le processus.

Connectez-vous et migrez le code de l'application pour utiliser des connexions sans mot de passe

Pour le développement local, vérifiez que vous êtes authentifié avec le même compte Microsoft Entra auquel vous avez attribué le rôle sur l'espace de noms Service Bus. Vous pouvez vous authentifier via Azure CLI, Visual Studio, Azure PowerShell ou d'autres outils tels qu'IntelliJ.

Pour le développement local, vérifiez que vous êtes authentifié avec le même compte Microsoft Entra auquel vous avez attribué le rôle. Vous pouvez vous authentifier au moyen d'outils de développement populaires, comme Azure CLI ou Azure PowerShell. Les outils de développement avec lesquels vous pouvez vous authentifier dépendent de la langue.

Azure CLI

Connectez-vous à Azure via Azure CLI à l'aide de la commande suivante :

```
Azure CLI
```

```
az login
```

Ensuite, mettez à jour votre code pour utiliser des connexions sans mot de passe.

.NET

1. Pour utiliser `DefaultAzureCredential` dans une application .NET, installez le `Azure.Identity` package :

```
CLI .NET
```

```
dotnet add package Azure.Identity
```

2. Ajoutez le code suivant en haut de votre fichier :

C#

```
using Azure.Identity;
```

3. Identifiez le code qui crée un objet `ServiceBusClient` pour la connexion à Azure Service Bus. Mettez à jour votre code pour le faire correspondre à l'exemple suivant :

C#

```
var serviceBusNamespace =
"https://{namespace}.servicebus.windows.net";
ServiceBusClient client = new(
    serviceBusNamespace,
    new DefaultAzureCredential());
```

Exécuter l'application localement.

Après avoir apporté ces modifications de code, exécutez votre application localement. La nouvelle configuration doit récupérer vos informations d'identification locales, telles qu'Azure CLI, Visual Studio ou IntelliJ. Les rôles que vous avez attribués à votre utilisateur de développement local dans Azure permettent à votre application de se connecter au service Azure localement.

Configurer l'environnement d'hébergement Azure

Une fois que votre application est configurée pour utiliser des connexions sans mot de passe et s'exécute localement, le même code peut s'authentifier auprès des services Azure après son déploiement sur Azure. Par exemple, une application déployée sur une instance Azure App Service avec une identité managée activée peut se connecter au Azure Service Bus.

Créer une identité managée à l'aide du portail Azure

Les étapes suivantes montrent comment créer une identité managée affectée par le système pour différents services d'hébergement web. L'identité managée peut se connecter en toute sécurité à d'autres services Azure à l'aide des configurations d'application que vous avez configurées.

Certains environnements d'hébergement d'applications prennent en charge Service Connector, ce qui vous permet de connecter des services de calcul Azure à d'autres services de stockage. Service Connector configure automatiquement les paramètres réseau et les informations de connexion. Vous pouvez en savoir plus sur Service Connector et les scénarios pris en charge sur la [page de vue d'ensemble](#).

Les services de calcul suivants sont actuellement pris en charge :

- Azure App Service
- Azure Spring Cloud
- Azure Container Apps (préversion)

Pour ce guide de migration, vous allez utiliser App Service, mais les étapes sont similaires sur Azure Spring Apps et Azure Container Apps.

⚠ Notes

Azure Spring Apps prend actuellement uniquement en charge Service Connector à l'aide de chaînes de connexion.

1. Dans la page de vue d'ensemble principale de votre App Service, sélectionnez **Service Connector** dans la navigation de gauche.
2. Sélectionnez + **Créer** dans le menu supérieur et le panneau **Créer une connexion** s'ouvre. Saisissez les valeurs suivantes :
 - **Type de service** : choisissez **Service Bus**.
 - **Abonnement** : sélectionnez l'abonnement que vous souhaitez utiliser.
 - **Nom de la connexion** : entrez un nom pour votre connexion, par exemple *connector_appservice_servicebus*.
 - **Type de client**: laissez la valeur par défaut sélectionnée ou choisissez le client spécifique que vous souhaitez utiliser.
3. Sélectionnez **Suivant : authentification**.
4. Vérifiez que **l'Identité managée affectée par le système (recommandée)** est sélectionnée, puis choisissez **Suivant : Mise en réseau**.
5. Laissez les valeurs par défaut sélectionnées, puis choisissez **Suivant : Vérifier + Créer**.

5. Une fois qu'Azure a validé vos paramètres, sélectionnez **Créer**.

Le connecteur de service crée automatiquement une identité managée affectée par le système pour l'app service. Le connecteur attribue également à l'identité managée un rôle de **propriétaire de données Azure Service Bus** pour le service bus que vous avez sélectionné.

Vous pouvez également activer l'identité managée sur un environnement d'hébergement Azure à l'aide d'Azure CLI.

Connecteur de service

Vous pouvez utiliser un connecteur de services pour créer une connexion entre un environnement d'hébergement de calcul Azure et un service cible à l'aide d'Azure CLI. L'interface CLI gère automatiquement la création d'une identité managée et affecte le rôle approprié, comme expliqué dans les [instructions du portail](#).

Si vous utilisez Azure App Service, utilisez la commande `az webapp connection` :

Azure CLI

```
az webapp connection create servicebus \
--resource-group <resource-group-name> \
--name <webapp-name> \
--target-resource-group <target-resource-group-name> \
--namespace <target-service-bus-namespace> \
--system-identity
```

Si vous utilisez Azure Spring Apps, utilisez la commande `az spring connection` :

Azure CLI

```
az spring connection create servicebus \
--resource-group <resource-group-name> \
--service <service-instance-name> \
--app <app-name> \
--deployment <deployment-name> \
--target-resource-group <target-resource-group> \
--namespace <target-service-bus-namespace> \
--system-identity
```

Si vous utilisez Azure Container Apps, utilisez la commande `az containerapp connection` :

Azure CLI

```
az containerapp connection create servicebus \
    --resource-group <resource-group-name> \
    --name <webapp-name> \
    --target-resource-group <target-resource-group-name> \
    --namespace <target-service-bus-namespace> \
    --system-identity
```

Attribuer des rôles à l'identité managée

Ensuite, vous devez accorder des autorisations à l'identité managée que vous avez créée pour accéder à votre Service Bus. Vous pouvez le faire en affectant un rôle à l'identité managée, comme vous l'avez fait avec votre utilisateur de développement local.

Connecteur de service

Si vous avez connecté vos services à l'aide du connecteur de service, vous n'avez pas besoin d'effectuer cette étape. Les configurations nécessaires ont été gérées pour vous :

- Si vous avez sélectionné une identité managée lors de la création de la connexion, une identité managée affectée par le système a été créée pour votre application et le rôle de **propriétaire de données Azure Service Bus** dans le Service Bus.
- Si vous avez sélectionné la chaîne de connexion, la chaîne de connexion a été ajoutée en tant que variable d'environnement d'application.

Tester l'application

Après avoir apporté ces modifications de code, accédez à votre application hébergée dans le navigateur. Votre application doit être en mesure de se connecter au Service Bus avec succès. N'oubliez pas qu'il peut falloir plusieurs minutes pour que les attributions de rôle se propagent dans l'environnement Azure. Votre application est désormais configurée pour s'exécuter localement et dans un environnement de production sans que les développeurs n'aient à gérer les secrets dans l'application elle-même.

Étapes suivantes

Dans ce didacticiel, vous avez appris à migrer une application vers des connexions sans mot de passe.

Migrer une application pour utiliser des connexions sans mot de passe avec le service Stockage Blob Azure

Article • 15/05/2023

Les demandes d'application adressées aux services Azure doivent être authentifiées à l'aide de configurations telles que des clés d'accès de compte ou de connexions sans mot de passe. Toutefois, vous devez hiérarchiser les connexions sans mot de passe dans vos applications lorsque cela est possible. Les méthodes d'authentification traditionnelles qui utilisent des mots de passe ou des clés secrètes créent des risques et des complications de sécurité. Visitez le hub de [connexions sans mot de passe pour Azure Services](#) pour découvrir l'avantage des connexions sans mot de passe.

Le tutoriel suivant explique comment migrer une application existante pour se connecter et utiliser des connexions sans mot de passe. Ces mêmes étapes de migration doivent s'appliquer, que vous utilisez des clés d'accès, des chaînes de connexion ou une autre approche basée sur des secrets.

Configurer des rôles et des utilisateurs pour l'authentification de développement local

Lors du développement local, assurez-vous que le compte d'utilisateur qui accède aux données blob dispose des autorisations appropriées. Vous aurez besoin du **Contributeur aux données Blob de stockage** pour lire et écrire des données blob. Pour vous attribuer ce rôle, vous aurez besoin du rôle **Administrateur de l'accès utilisateur** ou d'un autre rôle qui inclut l'action **Microsoft.Authorization/roleAssignments/write**. Vous pouvez attribuer des rôles RBAC Azure à un utilisateur à l'aide du Portail Azure, Azure CLI ou Azure PowerShell. Vous pouvez en savoir plus sur les étendues disponibles pour les attributions de rôles dans la page [vue d'ensemble de l'étendue](#).

Dans ce scénario, vous allez attribuer des autorisations à votre compte d'utilisateur, étendues au compte de stockage, pour suivre le [Principe des priviléges minimum](#). Cette pratique offre aux utilisateurs uniquement les autorisations minimales nécessaires et crée des environnements de production plus sécurisés.

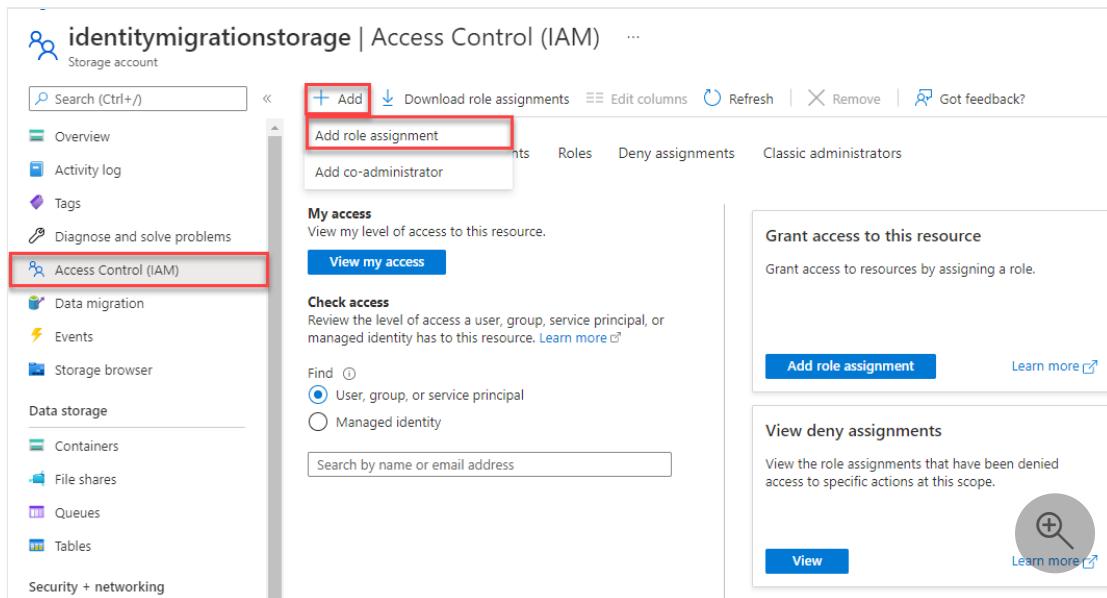
L'exemple suivant affecte le rôle **Contributeur aux données Blob du stockage** à votre compte d'utilisateur, qui fournit à la fois un accès en lecture et en écriture aux données d'objet blob dans votre compte de stockage.

ⓘ Important

Dans la plupart des cas, la propagation de l'attribution de rôle dans Azure peut prendre une ou deux minutes, mais dans de rares cas, cela peut prendre jusqu'à huit minutes. Si vous recevez des erreurs d'authentification lorsque vous exécutez votre code pour la première fois, patientez quelques instants et réessayez.

Azure portal

1. Dans le Portail Azure, recherchez votre compte de stockage à l'aide de la barre de recherche principale ou de la navigation gauche.
2. Dans la page vue d'ensemble du compte de stockage, sélectionnez **Contrôle d'accès (IAM)** dans le menu de gauche.
3. Sur la page **Contrôle d'accès (IAM)**, sélectionnez l'onglet **Attributions de rôles**.
4. Sélectionnez **+ Ajouter** dans le menu supérieur, puis **Ajouter une attribution de rôle** dans le menu déroulant résultant.



5. Utilisez la zone de recherche pour filtrer les résultats sur le rôle souhaité. Pour cet exemple, recherchez *Contributeur aux données Blob du stockage*, sélectionnez le résultat correspondant, puis choisissez **Suivant**.
6. Sous **Attribuer l'accès à**, sélectionnez **Utilisateur, groupe ou principal de service**, puis sélectionnez **+ Sélectionner des membres**.

7. Dans la boîte de dialogue, recherchez votre nom d'utilisateur Microsoft Entra (généralement votre adresse e-mail *user@domain*), puis choisissez **Selectionner** en bas de la boîte de dialogue.
8. Sélectionnez **Vérifier + affecter** pour accéder à la page finale, puis **Vérifier + attribuer** à nouveau pour terminer le processus.

Connectez-vous et migrez le code de l'application pour utiliser des connexions sans mot de passe

Pour le développement local, vérifiez que vous êtes authentifié avec le même compte Microsoft Entra auquel vous avez attribué le rôle. Vous pouvez vous authentifier au moyen d'outils de développement populaires, comme Azure CLI ou Azure PowerShell. Les outils de développement avec lesquels vous pouvez vous authentifier dépendent de la langue.

Azure CLI

Connectez-vous à Azure via Azure CLI à l'aide de la commande suivante :

```
Azure CLI
```

```
az login
```

Ensuite, mettez à jour votre code pour utiliser des connexions sans mot de passe.

.NET

1. Pour utiliser `DefaultAzureCredential` dans une application .NET, installez le `Azure.Identity` package :

```
CLI .NET
```

```
dotnet add package Azure.Identity
```

2. Ajoutez le code suivant en haut de votre fichier :

C#

```
using Azure.Identity;
```

3. Identifiez les emplacements de votre code qui créent une instance

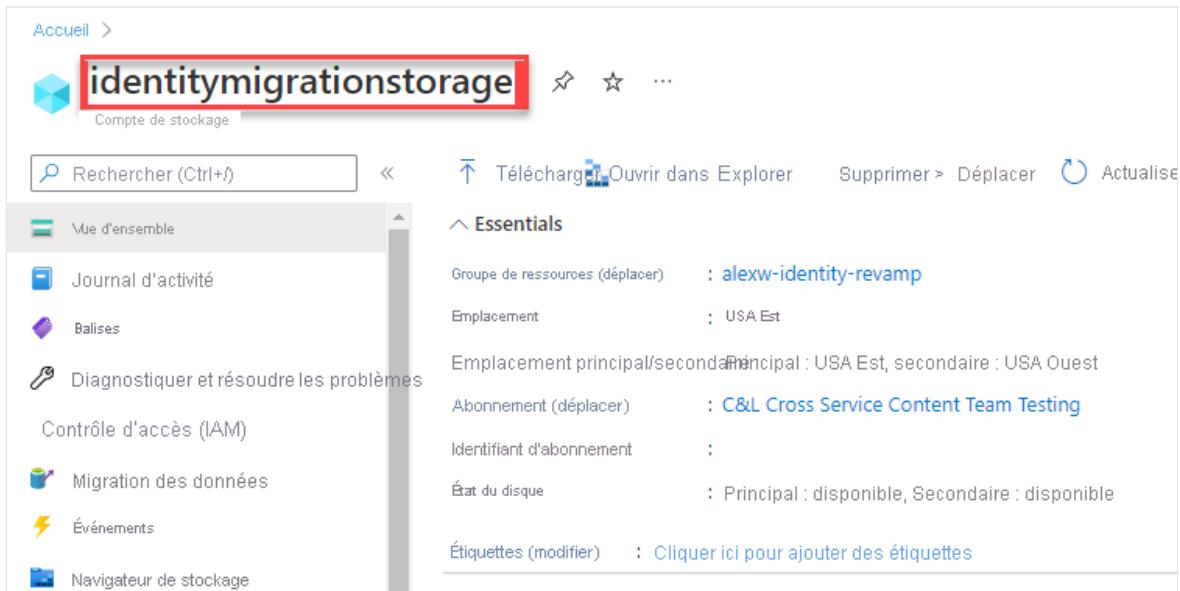
`BlobServiceClient` pour la connexion au Stockage Blob Azure. Mettez à jour votre code pour le faire correspondre à l'exemple suivant :

C#

```
DefaultAzureCredential credential = new();

BlobServiceClient blobServiceClient = new(
    new Uri($"https://{{storageAccountName}}.blob.core.windows.net"),
    credential);
```

4. Veillez à mettre à jour le nom du compte de stockage dans l'URI de votre `BlobServiceClient`. Vous trouverez le nom du compte de stockage dans la page de vue d'ensemble du portail Azure.



The screenshot shows the Azure Storage Account overview page. The account name 'identitymigrationstorage' is highlighted with a red box. The page displays various details about the storage account, including its resource group ('alexw-identity-revamp'), location ('USA Est'), primary and secondary locations ('USA Est, USA Ouest'), subscription ('C&L Cross Service Content Team Testing'), disk status ('Principal : disponible, Secondaire : disponible'), and a note to add labels. A sidebar on the left lists options like 'Vue d'ensemble', 'Journal d'activité', 'Balises', etc.

Exécutez l'application localement.

Après avoir apporté ces modifications de code, exécutez votre application localement. La nouvelle configuration doit récupérer vos informations d'identification locales, telles qu'Azure CLI, Visual Studio ou IntelliJ. Les rôles que vous avez attribués à votre utilisateur de développement local dans Azure permettent à votre application de se connecter au service Azure localement.

Configurer l'environnement d'hébergement Azure

Une fois que votre application est configurée pour utiliser des connexions sans mot de passe et s'exécute localement, le même code peut s'authentifier auprès des services Azure après son déploiement sur Azure. Les sections suivantes expliquent comment configurer une application déployée pour se connecter à Stockage Blob Azure à l'aide d'une identité managée.

Créer l'identité managée

Vous pouvez créer une identité managée affectée par l'utilisateur à l'aide du Portail Azure ou d'Azure CLI. Votre application utilise l'identité pour s'authentifier auprès d'autres services.

Azure portal

1. À partir du Portail Azure, recherchez *Identités managées*. Sélectionnez le résultat **Identités managées**.
2. Sélectionnez + **Créer** en haut de la page de présentation des **Identités managées**.
3. Sous l'onglet **Informations de base**, entrez les valeurs suivantes :
 - **Abonnement** : sélectionnez l'option souhaitée.
 - **Groupe de ressources** : sélectionnez votre groupe de ressources souhaité.
 - **Région** : sélectionnez une région proche de votre emplacement.
 - **Nom** : entrez un nom reconnaissable pour votre identité, par exemple *MigrationIdentity*.
4. Au bas de la page, sélectionnez **Examiner et créer**.
5. Une fois les vérifications de validation terminées, sélectionnez **Créer**. Azure crée une identité affectée par l'utilisateur.

Une fois la ressource créée, sélectionnez **Accéder à la ressource** pour afficher les détails de l'identité managée.

Create User Assigned Managed Identity

Basics Tags Review + create

Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * ⓘ

Test Subscription

Resource group * ⓘ

passwordlesstesting

[Create new](#)

Instance details

Region * ⓘ

South Central US

Name * ⓘ

MigrationIdentity



[Review + create](#)

[< Previous](#)

[Next : Tags >](#)

Associer l'identité managée à votre application web

Vous devez configurer votre application web pour utiliser l'identité managée créée. Attribuez l'identité à votre application à l'aide du Portail Azure ou d'Azure CLI.

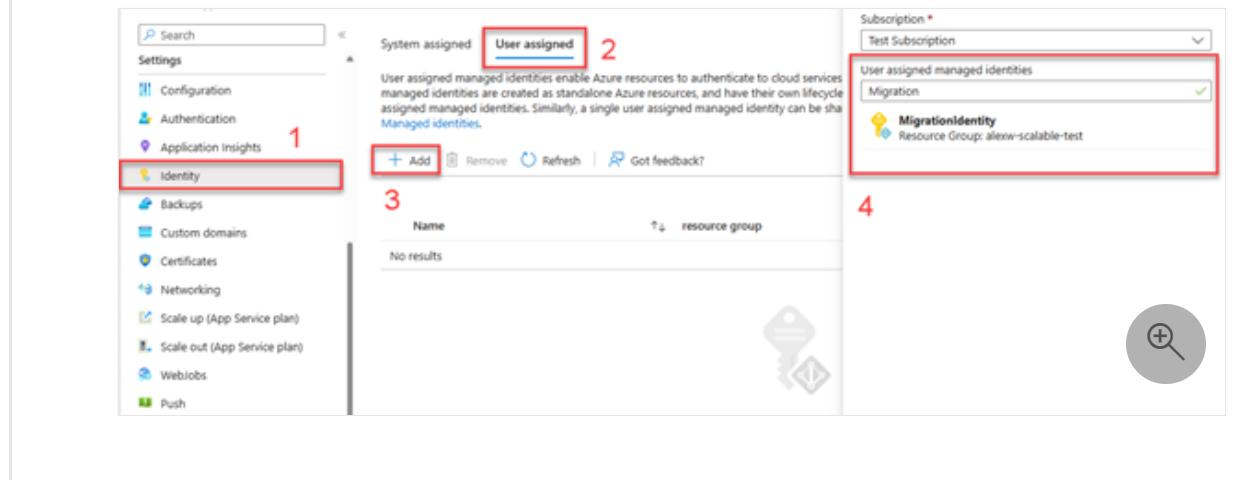
Azure portal

Effectuez les étapes suivantes dans le Portail Azure pour associer une identité à votre application. Ces mêmes étapes s'appliquent aux services Azure suivants :

- Azure Spring Apps
- Azure Container Apps
- Machines virtuelles Azure
- Azure Kubernetes Service

1. Accédez à la page de présentation de votre application web front-end.
2. Sélectionnez Identité dans la barre de navigation de gauche.
3. Dans la page Identité, basculez vers l'onglet Utilisateur affecté.

4. Sélectionnez + Ajouter pour ouvrir le menu volant Ajouter une identité managée affectée par l'utilisateur.
5. Sélectionnez l'abonnement utilisé précédemment pour créer l'identité.
6. Recherchez **MigrationIdentity** par nom et sélectionnez-le dans les résultats de la recherche.
7. Sélectionnez Ajouter pour associer l'identité à votre application.



Attribuer des rôles à l'identité managée

Ensuite, vous devez accorder des autorisations à l'identité managée que vous avez créée pour accéder à votre compte de stockage. Accordez des autorisations en affectant un rôle à l'identité managée, comme vous l'avez fait avec votre utilisateur de développement local.

Azure portal

1. Accédez à la page vue d'ensemble de votre compte de stockage et sélectionnez **Access Control (IAM)** dans la navigation de gauche.
2. Choisissez **Ajouter une attribution de rôle**

Accueil | identitymigrationstorage_1658927669422 | Vue d'ensemble | identitymigrationstorage | Contrôle d'accès (IAM) ...

Rechercher (Ctrl+I)

Vérifier l'accès Attributions de rôles Rôles Affectations de refus, Administrateurs classiques

Mon accès Afficher mon niveau d'accès à cette ressource.

Afficher mon accès

Vérifier l'accès Passez en revue le niveau d'accès d'un utilisateur, d'un groupe, d'un principal de service ou d'une identité managée à cette ressource. En savoir plus

Rechercher Utilisateur, groupe ou principal de service Identité managée

Rechercher par nom ou adresse e-mail

Ajouter une attribution de rôle En savoir plus

Afficher les affectations de refus Affichez les attributions de rôles auxquelles l'accès à des actions spécifiques a été refusé dans cette étendue. En savoir plus

Affichage

3. Dans la zone de recherche de **Rôle**, recherchez le *Contributeur de données blob de stockage*, qui est un rôle commun utilisé pour gérer les opérations de données pour les objets blob. Vous pouvez attribuer le rôle approprié pour votre cas d'usage. Sélectionnez le *contributeur de données blob de stockage* dans la liste, puis choisissez **Suivant**.
4. Sur l'écran **Ajouter une attribution de rôle**, pour l'option **Attribuer l'accès à l'option**, sélectionnez **Identité managée**. Choisissez ensuite **+ Sélectionner des membres**.
5. Dans le menu volant, recherchez l'identité managée que vous avez créée par nom, puis sélectionnez-la dans les résultats. Choisissez **Sélectionner** pour fermer le menu volant.

Accueil > identitymigrationstorage_1658927669422 | Vue d'ensemble | identitymigrationstorage | Contrôle d'accès (IAM) > Ajouter une attribution de rôle

Vous avez des commentaires ?

Membres du rôle Conditions (facultatif) Vérifier + attribuer

Rôle sélectionné

Attribuer l'accès à

Utilisateur, groupe ou principal de service Identité managée

Membres + Sélectionner des membres

Nom	ID de l'objet	Type
Aucun membre sélectionné		

Vérifier + attribuer Prédefaut Suivant 6

Sélectionner des membres

Sélectionné msdocs-web

Aucun utilisateur, groupe ou principal de service trouvé.

Membres sélectionnés : msdocs-web-app-123 Supprimer

Sélectionner Fermer

6. Sélectionnez **Suivant** quelques fois jusqu'à ce que vous puissiez sélectionner **Vérifier + attribuer** pour terminer l'attribution de rôle.

Mettre à jour le code d'application

Vous devez configurer votre code d'application pour rechercher l'identité managée spécifique créée lors du déploiement sur Azure. Dans certains scénarios, la définition explicite de l'identité managée pour l'application empêche également d'autres identités d'environnement d'être détectées et utilisées automatiquement par accident.

1. Dans la page de présentation de l'identité managée, copiez la valeur de l'ID client dans votre Presse-papiers.
2. Appliquez les modifications spécifiques au langage suivantes :

.NET

Créez un objet `DefaultAzureCredentialOptions` et transmettez-le à `DefaultAzureCredential`. Définissez la propriété `ManagedIdentityClientId` sur l'ID client.

C#

```
DefaultAzureCredential credential = new(
    new DefaultAzureCredentialOptions
    {
        ManagedIdentityClientId = managedIdentityClientId
    });

```

3. Redéployez votre code vers Azure après avoir apporté cette modification afin que les mises à jour de configuration soient appliquées.

Tester l'application

Après avoir déployé le code mis à jour, naviguez vers votre application hébergée dans le navigateur. Votre application doit être en mesure de se connecter au compte de stockage avec succès. N'oubliez pas qu'il peut falloir plusieurs minutes pour que les attributions de rôle se propagent dans l'environnement Azure. Votre application est désormais configurée pour s'exécuter localement et dans un environnement de production sans que les développeurs n'aient à gérer les secrets dans l'application elle-même.

Étapes suivantes

Dans ce didacticiel, vous avez appris à migrer une application vers des connexions sans mot de passe.

Vous pouvez lire les ressources suivantes pour explorer les concepts abordés dans cet article plus en détails :

- [Autoriser l'accès aux objets blob à l'aide Microsoft Entra ID](#)
- Pour en savoir plus sur .NET Core, consultez [Prise en main de .NET en 10 minutes](#).

Migrer une application pour utiliser des connexions sans mot de passe avec le service Stockage File d'attente Azure

Article • 15/05/2023

Les demandes d'application adressées aux services Azure doivent être authentifiées à l'aide de configurations telles que des clés d'accès de compte ou de connexions sans mot de passe. Toutefois, vous devez hiérarchiser les connexions sans mot de passe dans vos applications lorsque cela est possible. Les méthodes d'authentification traditionnelles qui utilisent des mots de passe ou des clés secrètes créent des risques et des complications de sécurité. Visitez le hub de [connexions sans mot de passe pour Azure Services](#) pour découvrir l'avantage des connexions sans mot de passe.

Le tutoriel suivant explique comment migrer une application existante pour se connecter et utiliser des connexions sans mot de passe. Ces mêmes étapes de migration doivent s'appliquer, que vous utilisez des clés d'accès, des chaînes de connexion ou une autre approche basée sur des secrets.

Configurer votre environnement de développement local

Les connexions sans mot de passe peuvent être configurées pour fonctionner dans les environnements locaux et hébergés par Azure. Dans cette section, vous allez appliquer des configurations permettant aux utilisateurs individuels de s'authentifier auprès de Stockage File d'attente Azure pour le développement local.

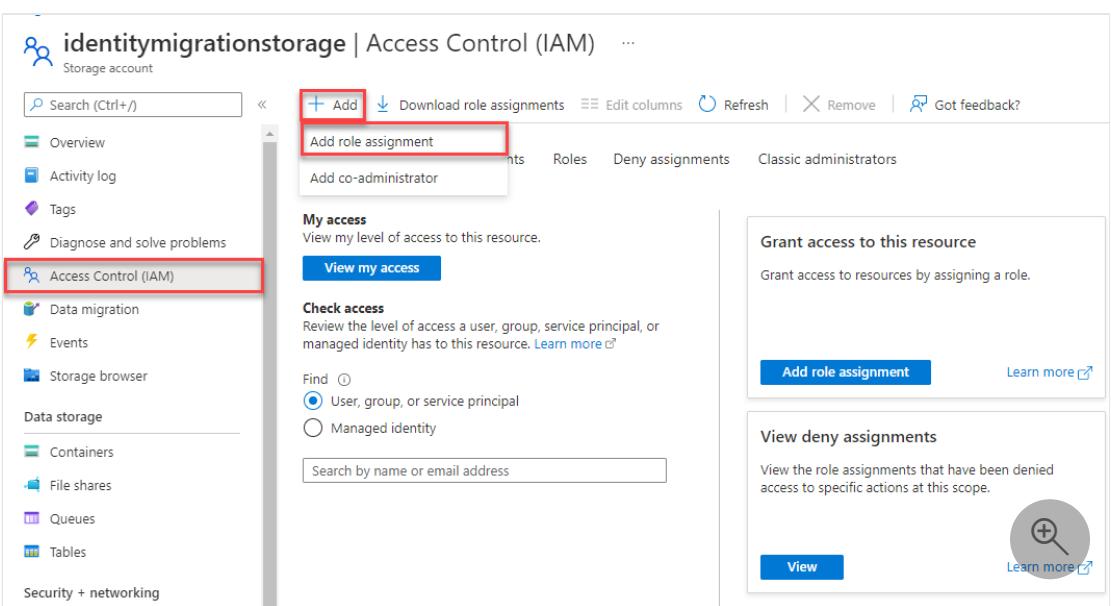
Attribuer des rôles d'utilisateur

En cas de développement local, vérifiez que le compte d'utilisateur qui accède au Stockage File d'attente Azure dispose des autorisations appropriées. Vous aurez besoin du rôle **Contributeur aux données en file d'attente du stockage** pour lire et écrire les données de la file d'attente. Pour vous attribuer ce rôle, vous aurez besoin du rôle **Administrateur de l'accès utilisateur** ou d'un autre rôle qui inclut l'action **Microsoft.Authorization/roleAssignments/write**. Vous pouvez attribuer des rôles RBAC Azure à un utilisateur à l'aide du Portail Azure, Azure CLI ou Azure PowerShell. Découvrez les étendues disponibles pour les attributions de rôles dans la page [vue d'ensemble de l'étendue](#).

L'exemple suivant attribue le rôle **Contributeur aux données en file d'attente du stockage** à votre compte d'utilisateur. Ce rôle permet d'accéder en lecture et en écriture aux données de la file d'attente dans votre compte de stockage.

Azure portal

1. Dans le Portail Azure, recherchez votre compte de stockage à l'aide de la barre de recherche principale ou de la navigation gauche.
2. Dans la page vue d'ensemble du compte de stockage, sélectionnez **Contrôle d'accès (IAM)** dans le menu de gauche.
3. Sur la page **Contrôle d'accès (IAM)**, sélectionnez l'onglet **Attributions de rôles**.
4. Sélectionnez **+ Ajouter** dans le menu supérieur, puis **Ajouter une attribution de rôle** dans le menu déroulant résultant.



5. Utilisez la zone de recherche pour filtrer les résultats sur le rôle souhaité. Pour cet exemple, recherchez *Contributeur aux données file d'attente du stockage*, sélectionnez le résultat correspondant, puis choisissez **Suivant**.
6. Sous **Attribuer l'accès à**, sélectionnez **Utilisateur, groupe ou principal de service**, puis sélectionnez **+ Sélectionner des membres**.
7. Dans la boîte de dialogue, recherchez votre nom d'utilisateur Microsoft Entra (généralement votre adresse e-mail *user@domain*), puis choisissez **Sélectionner** en bas de la boîte de dialogue.

8. Sélectionnez **Vérifier + affecter** pour accéder à la page finale, puis **Vérifier + attribuer** à nouveau pour terminer le processus.

ⓘ Important

Dans la plupart des cas, la propagation de l'attribution de rôle dans Azure prend une ou deux minutes, mais dans de rares cas, elle peut prendre jusqu'à huit minutes. Si vous recevez des erreurs d'authentification lorsque vous exécutez votre code pour la première fois, patientez quelques instants et réessayez.

Se connecter localement à Azure

Pour le développement local, vérifiez que vous êtes authentifié avec le même compte Microsoft Entra auquel vous avez attribué le rôle. Vous pouvez vous authentifier au moyen d'outils de développement populaires, comme Azure CLI ou Azure PowerShell. Les outils de développement avec lesquels vous pouvez vous authentifier dépendent de la langue.

Azure CLI

Connectez-vous à Azure via Azure CLI à l'aide de la commande suivante :

```
Azure CLI
```

```
az login
```

Mettre à jour le code de l'application pour utiliser des connexions sans mot de passe

La bibliothèque de client Azure Identity, pour chacun des écosystèmes suivants, fournit une classe `DefaultAzureCredential` qui gère l'authentification sans mot de passe auprès d'Azure :

- .NET
- Go ↗
- Java
- Node.js
- Python

`DefaultAzureCredential` prend en charge plusieurs méthodes d'authentification. La méthode à utiliser est déterminée au moment de l'exécution. Cette approche permet à votre application d'utiliser différentes méthodes d'authentification dans différents environnements (local ou production) sans implémenter de code spécifique à l'environnement. Consultez les liens précédents pour connaître l'ordre et les emplacements dans lesquels `DefaultAzureCredential` recherche des informations d'identification.

.NET

1. Pour utiliser `DefaultAzureCredential` dans une application .NET, installez le `Azure.Identity` package :

CLI .NET

```
dotnet add package Azure.Identity
```

2. Ajoutez le code suivant en haut de votre fichier :

C#

```
using Azure.Identity;
```

3. Identifiez les emplacements de votre code qui créent un objet `QueueClient` pour la connexion au Stockage File d'attente Azure. Mettez à jour votre code pour le faire correspondre à l'exemple suivant :

C#

```
DefaultAzureCredential credential = new();

QueueClient queueClient = new(
    new
    Uri($"https://{{storageAccountName}}.queue.core.windows.net/{{queueName}}"),
    new DefaultAzureCredential());
```

4. Veillez à mettre à jour le nom du compte de stockage dans l'URI de votre objet `QueueClient`. Vous trouverez le nom du compte de stockage dans la page de vue d'ensemble du portail Azure.

Accueil >

identitymigrationstorage

Compte de stockage

Rechercher (Ctrl+)

Télécharger Ouvrir dans Explorer Supprimer Déplacer Actualiser

Vue d'ensemble Journal d'activité Balises Diagnostiquer et résoudre les problèmes Contrôle d'accès (IAM) Migration des données Événements Navigateur de stockage

^ Essentials

Groupe de ressources (déplacer) : alexw-identity-revamp
Emplacement : USA Est
Emplacement principal/secondaire : Principal : USA Est, secondaire : USA Ouest
Abonnement (déplacer) : C&L Cross Service Content Team Testing
Identifiant d'abonnement :
État du disque : Principal : disponible, Secondaire : disponible
Étiquettes (modifier) : Cliquer ici pour ajouter des étiquettes

Exécutez l'application localement.

Après avoir apporté ces modifications de code, exécutez votre application localement. La nouvelle configuration doit récupérer vos informations d'identification locales, telles qu'Azure CLI, Visual Studio ou IntelliJ. Les rôles que vous avez attribués à votre utilisateur dans Azure permettent à votre application de se connecter au service Azure localement.

Configurer l'environnement d'hébergement Azure

Une fois que votre application est configurée pour utiliser des connexions sans mot de passe et s'exécute localement, le même code peut s'authentifier auprès des services Azure après son déploiement sur Azure. Les sections suivantes expliquent comment configurer une application déployée pour se connecter à Stockage File d'attente Azure à l'aide d'une [identité managée](#). Les identités managées fournissent une identité managée automatiquement dans Microsoft Entra ID que les applications peuvent utiliser lors de la connexion à des ressources qui prennent en charge l'authentification Microsoft Entra. En savoir plus sur les identités managées :

- [Vue d'ensemble des connexions sans mot de passe](#)
- [Meilleures pratiques d'identité managée](#)

Créer l'identité managée

Vous pouvez créer une identité managée affectée par l'utilisateur à l'aide du Portail Azure ou d'Azure CLI. Votre application utilise l'identité pour s'authentifier auprès

d'autres services.

Azure portal

1. À partir du Portail Azure, recherchez *Identités managées*. Sélectionnez le résultat **Identités managées**.
2. Sélectionnez + **Créer** en haut de la page de présentation des **Identités managées**.
3. Sous l'onglet **Informations de base**, entrez les valeurs suivantes :
 - **Abonnement** : sélectionnez l'option souhaitée.
 - **Groupe de ressources** : sélectionnez votre groupe de ressources souhaité.
 - **Région** : sélectionnez une région proche de votre emplacement.
 - **Nom** : entrez un nom reconnaissable pour votre identité, par exemple *MigrationIdentity*.
4. Au bas de la page, sélectionnez **Examiner et créer**.
5. Une fois les vérifications de validation terminées, sélectionnez **Créer**. Azure crée une identité affectée par l'utilisateur.

Une fois la ressource créée, sélectionnez **Accéder à la ressource** pour afficher les détails de l'identité managée.

Create User Assigned Managed Identity

Basics Tags Review + create

Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * ⓘ

Test Subscription

Resource group * ⓘ

passwordlesstesting

[Create new](#)

Instance details

Region * ⓘ

South Central US

Name * ⓘ

MigrationIdentity



[Review + create](#)

[< Previous](#)

[Next : Tags >](#)

Associer l'identité managée à votre application web

Vous devez configurer votre application web pour utiliser l'identité managée créée. Attribuez l'identité à votre application à l'aide du Portail Azure ou d'Azure CLI.

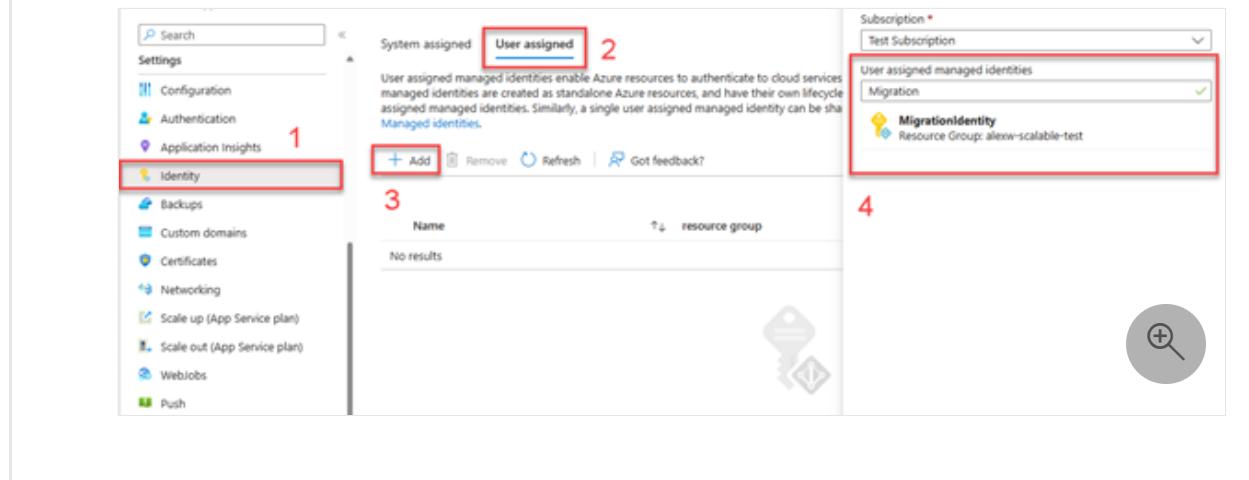
Azure portal

Effectuez les étapes suivantes dans le Portail Azure pour associer une identité à votre application. Ces mêmes étapes s'appliquent aux services Azure suivants :

- Azure Spring Apps
- Azure Container Apps
- Machines virtuelles Azure
- Azure Kubernetes Service

1. Accédez à la page de présentation de votre application web front-end.
2. Sélectionnez Identité dans la barre de navigation de gauche.
3. Dans la page Identité, basculez vers l'onglet Utilisateur affecté.

4. Sélectionnez + Ajouter pour ouvrir le menu volant Ajouter une identité managée affectée par l'utilisateur.
5. Sélectionnez l'abonnement utilisé précédemment pour créer l'identité.
6. Recherchez **MigrationIdentity** par nom et sélectionnez-le dans les résultats de la recherche.
7. Sélectionnez Ajouter pour associer l'identité à votre application.



Attribuer des rôles à l'identité managée

Ensuite, vous devez accorder des autorisations à l'identité managée que vous avez créée pour accéder à votre compte de stockage. Accordez des autorisations en affectant un rôle à l'identité managée, comme vous l'avez fait avec votre utilisateur de développement local.

Azure portal

1. Accédez à la page vue d'ensemble de votre compte de stockage et sélectionnez **Access Control (IAM)** dans la navigation de gauche.
2. Choisissez **Ajouter une attribution de rôle**

The screenshot shows the 'identitymigrationstorage | Access Control (IAM)' page in the Azure portal. The left sidebar has a red box around the 'Access Control (IAM)' item. The main area has a red box around the 'Add role assignment' button in the 'Grant access to this resource' section.

3. Dans la zone de recherche **Rôle**, recherchez *Contributeur aux données en file d'attente du stockage*, qui est un rôle commun utilisé pour gérer les opérations de données pour les files d'attente. Vous pouvez attribuer le rôle approprié pour votre cas d'usage. Sélectionnez le *Contributeur aux données en file d'attente du stockage* dans la liste et choisissez **Suivant**.
4. Sur l'écran **Ajouter une attribution de rôle**, pour l'option **Attribuer l'accès à l'option**, sélectionnez **Identité managée**. Choisissez ensuite **+ Sélectionner des membres**.
5. Dans le menu volant, recherchez l'identité managée que vous avez créée par nom, puis sélectionnez-la dans les résultats. Choisissez **Sélectionner** pour fermer le menu volant.

The screenshot shows the 'Add role assignment' wizard. Step 1 highlights the 'Managed identity' radio button. Step 2 highlights the '+ Select members' button. Step 3 highlights the search bar with 'msdocs-web' entered. Step 4 highlights the 'Selected members' list with 'msdocs-web-app-123'. Step 5 highlights the 'Select' button. Step 6 highlights the 'Next' button.

6. Sélectionnez **Suivant** quelques fois jusqu'à ce que vous puissiez sélectionner **Vérifier + attribuer** pour terminer l'attribution de rôle.

Mettre à jour le code d'application

Vous devez configurer votre code d'application pour rechercher l'identité managée spécifique créée lors du déploiement sur Azure. Dans certains scénarios, la définition explicite de l'identité managée pour l'application empêche également d'autres identités d'environnement d'être détectées et utilisées automatiquement par accident.

1. Dans la page de présentation de l'identité managée, copiez la valeur de l'ID client dans votre Presse-papiers.
2. Appliquez les modifications spécifiques au langage suivantes :

.NET

Créez un objet `DefaultAzureCredentialOptions` et transmettez-le à `DefaultAzureCredential`. Définissez la propriété `ManagedIdentityClientId` sur l'ID client.

C#

```
DefaultAzureCredential credential = new(
    new DefaultAzureCredentialOptions
    {
        ManagedIdentityClientId = managedIdentityClientId
    });

```

3. Redéployez votre code vers Azure après avoir apporté cette modification afin que les mises à jour de configuration soient appliquées.

Tester l'application

Après avoir déployé le code mis à jour, naviguez vers votre application hébergée dans le navigateur. Votre application doit être en mesure de se connecter au compte de stockage avec succès. N'oubliez pas qu'il peut falloir plusieurs minutes pour que les attributions de rôle se propagent dans l'environnement Azure. Votre application est désormais configurée pour s'exécuter localement et dans un environnement de production sans que les développeurs n'aient à gérer les secrets dans l'application elle-même.

Étapes suivantes

Dans ce didacticiel, vous avez appris à migrer une application vers des connexions sans mot de passe.

Vous pouvez lire les ressources suivantes pour explorer les concepts abordés dans cet article plus en détails :

- [Autoriser l'accès aux objets blob à l'aide Microsoft Entra ID](#)
- Pour plus d'informations sur .NET, consultez [Démarrer avec .NET en 10 minutes ↗](#).

Tutoriel : Créer une connexion sans mot de passe à un service de base de données par le biais de Service Connector

Article • 18/11/2023

Les connexions sans mot de passe utilisent des identités managées pour accéder aux services Azure. Avec cette approche, vous n'avez pas besoin de suivre et de gérer manuellement les secrets pour les identités managées. Ces tâches sont gérées de manière sécurisée en interne par Azure.

Service Connector active les identités managées dans les services d'hébergement d'applications comme Azure Spring Apps, Azure App Service et Azure Container Apps. Service Connector configure également les services de base de données, tels qu'Azure Database pour PostgreSQL, Azure Database pour MySQL et Azure SQL Database, pour accepter les identités managées.

Ce tutoriel explique comment utiliser Azure CLI pour effectuer les tâches suivantes :

- ✓ Vérifier votre environnement initial avec Azure CLI.
- ✓ Créer une connexion sans mot de passe avec Service Connector.
- ✓ Utiliser les variables d'environnement ou les configurations générées par Service Connector pour accéder à un service de base de données.

Prérequis

- [Azure CLI](#) version 2.48.1 ou ultérieure.
- Compte Azure avec un abonnement actif. [Créez gratuitement un compte Azure](#).
- Une application déployée dans [Azure App Service](#) dans une [région prise en charge par Service Connector](#).

Configurer votre environnement

Compte

Connectez-vous avec Azure CLI par le biais de `az login`. Si vous utilisez Azure Cloud Shell ou si vous êtes déjà connecté, confirmez votre compte authentifié avec `az account`

show .

Installer l'extension sans mot de passe Service Connector

Installez l'extension sans mot de passe Service Connector pour Azure CLI :

Azure CLI

```
az extension add --name serviceconnector-passwordless --upgrade
```

Créer une connexion sans mot de passe

Ensuite, nous utilisons Azure App Service comme exemple pour créer une connexion à l'aide d'une identité managée.

Si vous utilisez :

- Azure Spring Apps : utilisez plutôt `az spring connection create`. Pour plus d'exemples, consultez la section [Connecter Azure Spring Apps à la base de données Azure](#).
- Azure Container Apps : utilisez plutôt `az containerapp connection create`. Pour plus d'exemples, consultez la section [Créer et connecter une base de données PostgreSQL avec une connectivité d'identité](#).

ⓘ Notes

Si vous utilisez le portail Azure, accédez au panneau **Service Connector** d'**Azure App Service**, **Azure Spring Apps** ou **Azure Container Apps**, puis sélectionnez **Créer** pour créer une connexion. Le portail Azure compose automatiquement la commande à votre place et déclenche l'exécution de la commande sur Cloud Shell.

La commande Azure CLI suivante utilise un paramètre `--client-type`. Exécutez `az webapp connection create postgres-flexible -h` pour obtenir les types de clients pris en charge, puis choisissez celui qui correspond à votre application.

Identité managée affectée par l'utilisateur

Azure CLI

```
az webapp connection create postgres-flexible \
--resource-group $RESOURCE_GROUP \
```

```
--name $APPSCERVICE_NAME \
--target-resource-group $RESOURCE_GROUP \
--server $POSTGRESQL_HOST \
--database $DATABASE_NAME \
--user-identity client-id=XX subs-id=XX \
--client-type java
```

Cette commande Service Connector effectue les tâches suivantes en arrière-plan :

- Activez l'identité managée affectée par le système ou affectez une identité utilisateur pour l'application `$APPSCERVICE_NAME` hébergée par Azure App Service/Azure Spring Apps/Azure Container Apps.
- Définissez l'administrateur Microsoft Entra sur l'utilisateur connecté actuel.
- Ajoutez un utilisateur de base de données pour l'identité managée affectée par le système, l'identité managée affectée par l'utilisateur ou le principal de service. Accordez tous les privilèges de la base de données `$DATABASE_NAME` à cet utilisateur. Le nom d'utilisateur se trouve dans la chaîne de connexion dans la sortie de commande précédente.
- Définissez les configurations nommées `AZURE_MYSQL_CONNECTIONSTRING`, `AZURE_POSTGRESQL_CONNECTIONSTRING` ou `AZURE_SQL_CONNECTIONSTRING` sur la ressource Azure en fonction du type de base de données.
 - Pour App Service, les configurations sont définies dans le panneau **Paramètres de l'application**.
 - Pour Spring Apps, les configurations sont définies lors du lancement de l'application.
 - Pour Container Apps, les configurations sont définies sur les variables d'environnement. Vous pouvez obtenir toutes les configurations et leurs valeurs dans le panneau **Service Connector** du portail Azure.

Se connecter à une base de données avec l'authentification Microsoft Entra

Après avoir créé la connexion, vous pouvez utiliser la chaîne de connexion dans votre application pour vous connecter à la base de données avec l'authentification Microsoft Entra. Par exemple, vous pouvez utiliser les solutions suivantes pour vous connecter à la base de données avec l'authentification Microsoft Entra.

Pour .NET, il n'existe pas de plug-in ou de bibliothèque pour prendre en charge les connexions sans mot de passe. Vous pouvez obtenir un jeton d'accès pour l'identité managée ou le principal de service à l'aide d'une bibliothèque cliente telle que [Azure.Identity](#). Vous pouvez ensuite utiliser le jeton d'accès comme mot de passe pour vous connecter à la base de données. Lorsque vous utilisez le code ci-dessous, supprimez les marques de commentaire de la partie de l'extrait de code pour le type d'authentification que vous souhaitez utiliser.

C#

```
using Azure.Identity;
using Azure.Core;
using Npgsql;

// Uncomment the following lines according to the authentication type.
// For system-assigned identity.
// var sqlServerTokenProvider = new DefaultAzureCredential();

// For user-assigned identity.
// var sqlServerTokenProvider = new DefaultAzureCredential(
//     new DefaultAzureCredentialOptions
//     {
//         ManagedIdentityClientId =
// Environment.GetEnvironmentVariable("AZURE_POSTGRESQL_CLIENTID");
//     }
// );

// For service principal.
// var tenantId =
// Environment.GetEnvironmentVariable("AZURE_POSTGRESQL_TENANTID");
// var clientId =
// Environment.GetEnvironmentVariable("AZURE_POSTGRESQL_CLIENTID");
// var clientSecret =
// Environment.GetEnvironmentVariable("AZURE_POSTGRESQL_CLIENTSECRET");
// var sqlServerTokenProvider = new ClientSecretCredential(tenantId,
// clientId, clientSecret);

// Acquire the access token.
AccessToken accessToken = await sqlServerTokenProvider.GetTokenAsync(
    new TokenRequestContext(scopes: new string[]
    {
        "https://osssrdbms-aad.database.windows.net/.default"
    }));
    
// Combine the token with the connection string from the environment
variables provided by Service Connector.
string connectionString =
$"{
{Environment.GetEnvironmentVariable("AZURE_POSTGRESQL_CONNECTIONSTRING")}
};Password={accessToken.Token}";
```

```
// Establish the connection.  
using (var connection = new NpgsqlConnection(connectionString))  
{  
    Console.WriteLine("Opening connection using access token...");  
    connection.Open();  
}
```

Ensuite, si vous avez créé des tables et des séquences dans un serveur flexible PostgreSQL avant d'utiliser le connecteur de services, vous devez vous connecter en tant que propriétaire et accorder à <aad-username> l'autorisation créée par le connecteur de services. Le nom d'utilisateur de la chaîne de connexion ou de la configuration définie par le connecteur de service doit ressembler à aad_<connection name>. Si vous utilisez le Portail Azure, sélectionnez le bouton Développer à côté de la colonne Service Type et obtenez la valeur. Si vous utilisez Azure CLI, cochez configurations dans la sortie de la commande CLI.

Ensuite, exécutez la requête pour accorder l'autorisation

Azure CLI

```
az extension add --name rdbms-connect  
  
az postgres flexible-server execute -n <postgres-name> -u <owner-username> -  
p "<owner-password>" -d <database-name> --querytext "GRANT ALL PRIVILEGES ON  
ALL TABLES IN SCHEMA public TO \"<aad-username>\";GRANT ALL PRIVILEGES ON  
ALL SEQUENCES IN SCHEMA public TO \"<aad username>\";"
```

<owner-username> et <owner-password> est le propriétaire de la table existante qui peut accorder des autorisations à d'autres personnes. <aad-username> est l'utilisateur créé par Service Connector. Remplacez-les par la valeur réelle.

Validez le résultat avec la commande suivante :

Azure CLI

```
az postgres flexible-server execute -n <postgres-name> -u <owner-username> -  
p "<owner-password>" -d <database-name> --querytext "SELECT  
distinct(table_name) FROM information_schema.table_privileges WHERE  
grantee='<aad-username>' AND table_schema='public';" --output table
```

Déployer l'application sur un service d'hébergement Azure

Enfin, déployez votre application sur un service d'hébergement Azure. Ce service source peut utiliser une identité managée pour se connecter à la base de données cible sur Azure.

App Service

Par Azure App Service, vous pouvez déployer le code de l'application à l'aide de la commande `az webapp deploy`. Pour plus d'informations, consultez [Démarrage rapide : Déployer une application web ASP.NET](#).

Vous pouvez ensuite consulter le journal ou appeler l'application pour déterminer si elle peut se connecter à la base de données Azure.

Résolution des problèmes

Autorisation

Si vous rencontrez des erreurs liées aux autorisations, confirmez l'utilisateur connecté Azure CLI avec la commande `az account show`. Veillez à vous connecter avec le compte correct. Ensuite, vérifiez que vous disposez des autorisations suivantes qui peuvent être demandées pour créer une connexion sans mot de passe avec le connecteur de services.

Autorisation	Opération
<code>Microsoft.DBforPostgreSQL/flexibleServers/read</code>	Requise pour obtenir les informations du serveur de base de données
<code>Microsoft.DBforPostgreSQL/flexibleServers/write</code>	Requis pour activer l'authentification Microsoft Entra pour le serveur de base de données
<code>Microsoft.DBforPostgreSQL/flexibleServers/firewallRules/write</code>	Requise pour créer une règle de pare-feu en cas de blocage de l'adresse IP
<code>Microsoft.DBforPostgreSQL/flexibleServers/firewallRules/delete</code>	Requise pour rétablir la règle de pare-feu créée par Service Connector pour éviter un problème de sécurité
<code>Microsoft.DBforPostgreSQL/flexibleServers/administrators/read</code>	Requis pour vérifier si l'utilisateur de connexion

Autorisation	Opération
	Azure CLI est un administrateur de serveur de base de données Microsoft Entra
<code>Microsoft.DBforPostgreSQL/flexibleServers/administrators/write</code>	Requis pour ajouter l'utilisateur de connexion Azure CLI en tant qu'administrateur Microsoft Entra du serveur de base de données

Dans certains cas, les autorisations ne sont pas requises. Par exemple, si l'utilisateur authentifié par Azure CLI est déjà administrateur Active Directory sur SQL Server, vous n'avez pas besoin de l'autorisation `Microsoft.Sql/servers/administrators/write`.

Microsoft Entra ID

Si vous obtenez une erreur `ERROR: AADSTS53003: Your device is required to be managed to access this resource.`, demandez de l'aide à votre service informatique pour joindre cet appareil à Microsoft Entra ID. Pour plus d'informations, consultez [Appareils joints à Microsoft Entra](#).

Le connecteur de services doit accéder à Microsoft Entra ID pour obtenir les informations de votre compte et l'identité managée du service d'hébergement. Vous pouvez utiliser la commande suivante pour vérifier si votre appareil peut accéder Microsoft Entra ID :

Azure CLI

```
az ad signed-in-user show
```

Si vous ne vous connectez pas de manière interactive, vous pouvez également obtenir l'erreur et le message `Interactive authentication is needed`. Pour résoudre l'erreur, connectez-vous avec la commande `az login`.

Connectivité réseau

Si votre serveur de base de données est dans un réseau virtuel, assurez-vous que votre environnement qui exécute la commande Azure CLI peut accéder au serveur dans le réseau virtuel.

Étapes suivantes

Pour plus d'informations sur Service Connector et les connexions sans mot de passe, consultez les ressources suivantes :

[Documentation Service Connector](#)

[Connexions sans mot de passe pour les services Azure](#)

Intégrer Azure SQL Database avec Service Connector

Article • 06/02/2024

Cette page montre les méthodes d'authentification et les clients pris en charge, ainsi que des exemples de code que vous pouvez utiliser pour connecter des services de calcul à Azure SQL Database à l'aide du service Connector or. Vous pouvez toujours vous connecter à Azure SQL Database à l'aide d'autres méthodes. Cette page indique également les noms et les valeurs des variables d'environnement par défaut que vous obtenez lorsque vous créez la connexion du service.

Service de calcul pris en charge

Le Connecter de service peut être utilisé pour connecter les services de calcul suivants à Azure SQL Database :

- Azure App Service
- Azure Functions
- Azure Container Apps
- Azure Spring Apps

Types d'authentification et clients pris en charge

Le tableau ci-dessous montre quelles combinaisons de méthodes d'authentification et de clients sont prises en charge pour connecter votre service de calcul à Azure SQL Database à l'aide de Service Connecter or. Un « Oui » indique que la combinaison est prise en charge, tandis qu'une valeur « Non » indique qu'elle n'est pas prise en charge.

 Agrandir le tableau

Type de client	Identité managée affectée par le système	Identité managée affectée par l'utilisateur	Secret/chaîne de connexion	Principal du service
.NET	Oui	Oui	Oui	Oui
Go	Non	Non	Oui	Non
Java	Oui	Oui	Oui	Oui

Type de client	Identité managée affectée par le système	Identité managée affectée par l'utilisateur	Secret/chaîne de connexion	Principal du service
Java - Spring Boot	Oui	Oui	Oui	Oui
Node.js	Oui	Oui	Oui	Oui
PHP	Non	Non	Oui	Non
Python	Oui	Oui	Oui	Oui
Python - Django	Non	Non	Oui	Non
Ruby	Non	Non	Oui	Non
Aucun	Oui	Oui	Oui	Oui

Ce tableau indique que la méthode Secret/chaîne de connexion est prise en charge pour tous les types de clients. L'identité managée affectée par le système, l'identité managée affectée par l'utilisateur et les méthodes de principal de service sont prises en charge pour les types client .NET, Java, Java - Spring Boot, Node.js, Python et None. Ces méthodes ne sont pas prises en charge pour les types de client Go, PHP, Django et Ruby.

ⓘ Notes

L'identité managée affectée par le système, l'identité managée affectée par l'utilisateur et le principal de service sont uniquement pris en charge sur Azure CLI.

Noms de variables d'environnement par défaut ou propriétés d'application et exemple de code

Utilisez les détails de connexion ci-dessous pour connecter des services de calcul à Azure SQL Database. Pour chaque exemple ci-dessous, remplacez les textes d'espace réservé <sql-server>, <sql-database>, <sql-username> et <sql-password> par le nom de votre propre serveur, le nom de la base de données, l'ID d'utilisateur et le mot de passe. Pour plus d'informations sur les conventions d'affectation de noms, case activée l'article interne [du service Connecter or](#).

Identité managée affectée par le système

.NET

Agrandir le tableau

Nom de variable d'environnement par défaut	Description	Exemple de valeur
AZURE_SQL_CONNECTIONSTRING	Chaîne de connexion Azure SQL Database	Data Source=<sql-server>.database.windows.net,1433;Initial Catalog=<sql-database>;Authentication=ActiveDirectoryManagedIdentity

Exemple de code

Reportez-vous aux étapes et au code ci-dessous pour vous connecter à Azure SQL Database à l'aide d'une identité managée affectée par le système.

.NET

1. Installez des dépendances.

Bash

```
dotnet add package Microsoft.Data.SqlClient
```

2. Obtenez la chaîne de connexion Azure SQL Database à partir de la variable d'environnement ajoutée par le connecteur de services.

C#

```
using Microsoft.Data.SqlClient;

string connectionString =
    Environment.GetEnvironmentVariable("AZURE_SQL_CONNECTIONSTRING")!;

using var connection = new SqlConnection(connectionString);
connection.Open();
```

Pour plus d'informations, consultez [Utilisation de l'authentification d'identité managée Active Directory](#).

Pour plus d'informations, consultez [Page d'accueil pour la programmation client sur Microsoft SQL Server](#).

Identité managée affectée par l'utilisateur

.NET

[+] Agrandir le tableau

Nom de variable d'environnement par défaut	Description	Exemple de valeur
AZURE_SQL_CONNECTIONSTR ING	Chaîne de connexion Azure SQL Database	Data Source=<sql-server>.database.windows.net,1433;Initial Catalog=<sql-database>;User ID=<identity-client-ID>;Authentication=ActiveDirectoryManagedIdentity

Exemple de code

Reportez-vous aux étapes et au code ci-dessous pour vous connecter à Azure SQL Database à l'aide d'une identité managée affectée par l'utilisateur.

.NET

1. Installez des dépendances.

Bash

```
dotnet add package Microsoft.Data.SqlClient
```

2. Obtenez la chaîne de connexion Azure SQL Database à partir de la variable d'environnement ajoutée par le connecteur de services.

C#

```

using Microsoft.Data.SqlClient;

string connectionString =
    Environment.GetEnvironmentVariable("AZURE_SQL_CONNECTIONSTRING")!;

using var connection = new SqlConnection(connectionString);
connection.Open();

```

Pour plus d'informations, consultez [Utilisation de l'authentification d'identité managée Active Directory](#).

Pour plus d'informations, consultez [Page d'accueil pour la programmation client sur Microsoft SQL Server](#).

Chaîne de connexion

.NET

[Agrandir le tableau](#)

Nom de variable d'environnement par défaut	Description	Exemple de valeur
AZURE_SQL_CONNECTIONSTRIN G	Chaîne de connexion Azure SQL Database	Data Source=<sql-server>.database.windows.net,1433;Initial Catalog=<sql-database>;Password=<sql-password>

Exemple de code

Reportez-vous aux étapes et au code ci-dessous pour vous connecter à Azure SQL Database à l'aide d'un chaîne de connexion.

.NET

1. Installez des dépendances.

Bash

```
dotnet add package Microsoft.Data.SqlClient
```

2. Obtenez la chaîne de connexion Azure SQL Database à partir de la variable d'environnement ajoutée par le connecteur de services.

C#

```
using Microsoft.Data.SqlClient;

string connectionString =
    Environment.GetEnvironmentVariable("AZURE_SQL_CONNECTIONSTRING")!;

using var connection = new SqlConnection(connectionString);
connection.Open();
```

Pour plus d'informations, consultez [Page d'accueil pour la programmation client sur Microsoft SQL Server](#).

Principal du service

.NET

[+] Agrandir le tableau

Nom de variable d'environnement par défaut	Description	Exemple de valeur
AZURE_SQL_CLIENTID	Votre ID client	<client-ID>
AZURE_SQL_CLIENTSECRET	Votre clé secrète client	<client-secret>
AZURE_SQL_TENANTID	Votre ID d'abonnement	<tenant-ID>
AZURE_SQL_CONNECTIONSTRING	Chaîne de connexion Azure SQL Database	Data Source=<sql-server>.database.windows.net,1433;Initial Catalog=<sql-database>;UserID=a30eedc-e75f-4301-b1a9-56e81e0ce99c;Password=asdfgherty;Authentication=ActiveDirectoryServicePrincipal

Exemple de code

Reportez-vous aux étapes et au code ci-dessous pour vous connecter à Azure SQL Database à l'aide d'un principal de service.

.NET

1. Installez des dépendances.

Bash

```
dotnet add package Microsoft.Data.SqlClient
```

2. Obtenez la chaîne de connexion Azure SQL Database à partir de la variable d'environnement ajoutée par le connecteur de services.

C#

```
using Microsoft.Data.SqlClient;

string connectionString =
    Environment.GetEnvironmentVariable("AZURE_SQL_CONNECTIONSTRING")!;

using var connection = new SqlConnection(connectionString);
connection.Open();
```

Pour plus d'informations, consultez [Utilisation de l'authentification d'identité managée Active Directory](#).

Pour plus d'informations, consultez [Page d'accueil pour la programmation client sur Microsoft SQL Server](#).

Étapes suivantes

Suivez le tutoriel ci-dessous pour en savoir plus sur Service Connector.

[En savoir plus sur les concepts Service Connector](#)

Intégrer Azure Database pour MySQL avec Service Connector

Article • 02/02/2024

Cette page présente les méthodes d'authentification et les clients pris en charge, et présente des exemples de code que vous pouvez utiliser pour connecter Azure Database pour MySQL - Serveur flexible à d'autres services cloud à l'aide du service Connecter or. Cette page présente également les noms et les valeurs des variables d'environnement par défaut (ou configuration Spring Boot) que vous obtenez lorsque vous créez des connexions de service.

ⓘ Important

Le serveur unique Azure Database pour MySQL est en voie de mise hors service. Nous vous conseillons vivement de procéder à une mise à niveau vers Azure Database pour MySQL – Serveur flexible. Pour obtenir plus d'informations sur la migration vers Azure Database pour MySQL – Serveur flexible, consultez [Qu'en est-il du Serveur unique Azure Database pour MySQL ?](#)

Service de calcul pris en charge

Le Connecter or de service peut être utilisé pour connecter les services de calcul suivants à Azure Database pour MySQL :

- Azure App Service
- Azure Functions
- Azure Container Apps
- Azure Spring Apps

Types d'authentification et de clients pris en charge

Le tableau ci-dessous montre quelles combinaisons de méthodes d'authentification et de clients sont prises en charge pour connecter votre service de calcul à Azure Database pour MySQL à l'aide du service Connecter or. Un « Oui » indique que la combinaison est prise en charge, tandis qu'une valeur « Non » indique qu'elle n'est pas prise en charge.

Type de client	Identité managée affectée par le système	Identité managée affectée par l'utilisateur	Secret/chaîne de connexion	Principal du service
.NET	Oui	Oui	Oui	Oui
Go (pilote go-sql-driver pour MySQL)	Oui	Oui	Oui	Oui
Java (JDBC)	Oui	Oui	Oui	Oui
Java – Spring Boot (JDBC)	Oui	Oui	Oui	Oui
Node.js (mysql)	Oui	Oui	Oui	Oui
Python (mysql-connector-python)	Oui	Oui	Oui	Oui
Python-Django	Oui	Oui	Oui	Oui
PHP (MySQLi)	Oui	Oui	Oui	Oui
Ruby (mysql2)	Oui	Oui	Oui	Oui
Aucun	Oui	Oui	Oui	Oui

Ce tableau indique que toutes les combinaisons de types clients et de méthodes d'authentification dans la table sont prises en charge. Tous les types de clients peuvent utiliser l'une des méthodes d'authentification pour se connecter à Azure Database pour MySQL à l'aide du service Connecter or.

ⓘ Notes

L'identité managée affectée par le système, l'identité managée affectée par l'utilisateur et le principal de service sont uniquement pris en charge sur Azure CLI.

Noms de variables d'environnement par défaut ou propriétés d'application et exemple de code

Référez-vous aux détails de connexion et l'exemple de code dans les tableaux suivants, en fonction du type d'authentification et du type de client de votre connexion, pour connecter les services de calcul à Azure Database pour MySQL. Pour plus d'informations sur les conventions d'affectation de noms, case activée l'article interne [du service Connecter or.](#)

Identité managée affectée par le système

.NET

[Agrandir le tableau](#)

Nom de variable d'environnement par défaut	Description	Exemple de valeur
AZURE_MYSQL_CONNECTIONSTRING	Chaîne de connexion ADO.NET MySQL	Server=<MySQL-DB-name>.mysql.database.azure.com;Database=<MySQL-DB-name>;Port=3306;User Id=<MySQL-DBusername>;SSL Mode=Required;

Exemple de code

Reportez-vous aux étapes et au code ci-dessous pour vous connecter à Azure Database pour MySQL à l'aide d'une identité managée affectée par le système.

.NET

Pour .NET, il n'existe pas de plug-in ou de bibliothèque pour prendre en charge les connexions sans mot de passe. Vous pouvez obtenir un jeton d'accès pour l'identité managée ou le principal de service à l'aide d'une bibliothèque cliente telle que [Azure.Identity](#). Vous pouvez ensuite utiliser le jeton d'accès comme mot de passe pour vous connecter à la base de données. Lorsque vous utilisez le code ci-dessous, supprimez les marques de commentaire de la partie de l'extrait de code pour le type d'authentification que vous souhaitez utiliser.

C#

```
using Azure.Core;
using Azure.Identity;
using MySqlConnector;
```

```

// Uncomment the following lines according to the authentication type.
// For system-assigned managed identity.
// var credential = new DefaultAzureCredential();

// For user-assigned managed identity.
// var credential = new DefaultAzureCredential(
//     new DefaultAzureCredentialOptions
//     {
//         ManagedIdentityClientId =
// Environment.GetEnvironmentVariable("AZURE_MYSQL_CLIENTID");
//     });

// For service principal.
// var tenantId =
Environment.GetEnvironmentVariable("AZURE_MYSQL_TENANTID");
// var clientId =
Environment.GetEnvironmentVariable("AZURE_MYSQL_CLIENTID");
// var clientSecret =
Environment.GetEnvironmentVariable("AZURE_MYSQL_CLIENTSECRET");
// var credential = new ClientSecretCredential(tenantId, clientId,
clientSecret);

var tokenRequestContext = new TokenRequestContext(
    new[] { "https://osssrdbms-aad.database.windows.net/.default" });
AccessToken accessToken = await
credential.GetTokenAsync(tokenRequestContext);
// Open a connection to the MySQL server using the access token.
string connectionString =
    $""
{Environment.GetEnvironmentVariable("AZURE_MYSQL_CONNECTIONSTRING")};Pas
sword={accessToken.Token}";

using var connection = new MySqlConnection(connectionString);
Console.WriteLine("Opening connection using access token...");
await connection.OpenAsync();

// do something

```

Pour obtenir d'autres exemples de code, consultez la section [Connexion à des bases de données Azure à partir d'App Service sans secrets à l'aide d'une identité managée](#).

Identité managée affectée par l'utilisateur

.NET

 Agrandir le tableau

Nom de variable d'environnement par défaut	Description	Exemple de valeur
AZURE_MYSQL_CLIENTID	Votre ID client	<identity-client-ID>
AZURE_MYSQL_CONNECTIONSTRING	Chaîne de connexion ADO.NET MySQL	Server=<MySQL-DB-name>.mysql.database.azure.com;Database=<MySQL-DB-name>;Port=3306;User Id=<MySQL-DBusername>;SSL Mode=Required;

Exemple de code

Reportez-vous aux étapes et au code ci-dessous pour vous connecter à Azure Database pour MySQL à l'aide d'une identité managée affectée par l'utilisateur.

.NET

Pour .NET, il n'existe pas de plug-in ou de bibliothèque pour prendre en charge les connexions sans mot de passe. Vous pouvez obtenir un jeton d'accès pour l'identité managée ou le principal de service à l'aide d'une bibliothèque cliente telle que [Azure.Identity](#). Vous pouvez ensuite utiliser le jeton d'accès comme mot de passe pour vous connecter à la base de données. Lorsque vous utilisez le code ci-dessous, supprimez les marques de commentaire de la partie de l'extrait de code pour le type d'authentification que vous souhaitez utiliser.

C#

```
using Azure.Core;
using Azure.Identity;
using MySqlConnector;

// Uncomment the following lines according to the authentication type.
// For system-assigned managed identity.
// var credential = new DefaultAzureCredential();

// For user-assigned managed identity.
// var credential = new DefaultAzureCredential(
//     new DefaultAzureCredentialOptions
//     {
//         ManagedIdentityClientId =
// Environment.GetEnvironmentVariable("AZURE_MYSQL_CLIENTID");
//     });

// For service principal.
// var tenantId =
// Environment.GetEnvironmentVariable("AZURE_MYSQL_TENANTID");
```

```

// var clientId =
Environment.GetEnvironmentVariable("AZURE_MYSQL_CLIENTID");
// var clientSecret =
Environment.GetEnvironmentVariable("AZURE_MYSQL_CLIENTSECRET");
// var credential = new ClientSecretCredential(tenantId, clientId,
clientSecret);

var tokenRequestContext = new TokenRequestContext(
    new[] { "https://osssrdbms-aad.database.windows.net/.default" });
AccessToken accessToken = await
credential.GetTokenAsync(tokenRequestContext);
// Open a connection to the MySQL server using the access token.
string connectionString =
$"
{Environment.GetEnvironmentVariable("AZURE_MYSQL_CONNECTIONSTRING")};Pas
sword={accessToken.Token}";

using var connection = new MySqlConnection(connectionString);
Console.WriteLine("Opening connection using access token...");
await connection.OpenAsync();

// do something

```

Pour obtenir d'autres exemples de code, consultez la section [Connexion à des bases de données Azure à partir d'App Service sans secrets à l'aide d'une identité managée](#).

Chaîne de connexion

.NET

[Agrandir le tableau](#)

Nom de variable d'environnement par défaut	Description	Exemple de valeur
AZURE_MYSQL_CONNECTIONSTRING	Chaîne de connexion ADO.NET MySQL	Server=<MySQL-DB-name>.mysql.database.azure.com;Database=<MySQL-DB-name>;Port=3306;User Id=<MySQL-DBusername>;Password=<MySQL-DBpassword>;SSL Mode=Required

Exemple de code

Reportez-vous aux étapes et au code ci-dessous pour vous connecter à Azure Database pour MySQL à l'aide d'un chaîne de connexion.

.NET

1. Installez des dépendances. Suivez les instructions pour [installer le connecteur/NET MySQL](#)
2. Dans le code, obtenez le chaîne de connexion MySQL à partir des variables d'environnement ajoutées par le service Connecter or. Pour établir une connexion chiffrée au serveur MySQL via SSL, reportez-vous à [ces étapes](#).

C#

```
using System;
using System.Data;
using MySql.Data.MySqlClient;

string connectionString =
Environment.GetEnvironmentVariable("AZURE_MYSQL_CONNECTIONSTRING");
using (MySqlConnection connection = new
MySqlConnection(connectionString))
{
    connection.Open();
}
```

Principal du service

.NET

[\[\] Agrandir le tableau](#)

Nom de variable d'environnement par défaut	Description	Exemple de valeur
AZURE_MYSQL_CLIENTID	Votre ID client	<client-ID>
AZURE_MYSQL_CLIENTSECRET	Votre clé secrète client	<client-secret>
AZURE_MYSQL_TENANTID	Votre ID d'abonné	<tenant-ID>

Nom de variable d'environnement par défaut	Description	Exemple de valeur
AZURE_MYSQL_CONNECTIONSTRING	Chaîne de connexion ADO.NET MySQL	Server=<MySQL-DB-name>.mysql.database.azure.com;Database=<MySQL-DB-name>;Port=3306;User Id=<MySQL-DBusername>;SSL Mode=Required

Exemple de code

Reportez-vous aux étapes et au code ci-dessous pour vous connecter à Azure Database pour MySQL à l'aide d'un principal de service.

.NET

Pour .NET, il n'existe pas de plug-in ou de bibliothèque pour prendre en charge les connexions sans mot de passe. Vous pouvez obtenir un jeton d'accès pour l'identité managée ou le principal de service à l'aide d'une bibliothèque cliente telle que [Azure.Identity](#). Vous pouvez ensuite utiliser le jeton d'accès comme mot de passe pour vous connecter à la base de données. Lorsque vous utilisez le code ci-dessous, supprimez les marques de commentaire de la partie de l'extrait de code pour le type d'authentification que vous souhaitez utiliser.

C#

```
using Azure.Core;
using Azure.Identity;
using MySqlConnector;

// Uncomment the following lines according to the authentication type.
// For system-assigned managed identity.
// var credential = new DefaultAzureCredential();

// For user-assigned managed identity.
// var credential = new DefaultAzureCredential(
//     new DefaultAzureCredentialOptions
//     {
//         ManagedIdentityClientId =
// Environment.GetEnvironmentVariable("AZURE_MYSQL_CLIENTID");
//     });

// For service principal.
// var tenantId =
// Environment.GetEnvironmentVariable("AZURE_MYSQL_TENANTID");
// var clientId =
// Environment.GetEnvironmentVariable("AZURE_MYSQL_CLIENTID");
// var clientSecret =
```

```
Environment.GetEnvironmentVariable("AZURE_MYSQL_CLIENTSECRET");
// var credential = new ClientSecretCredential(tenantId, clientId,
clientSecret);

var tokenRequestContext = new TokenRequestContext(
    new[] { "https://osssrdbms-aad.database.windows.net/.default" });
AccessToken accessToken = await
credential.GetTokenAsync(tokenRequestContext);
// Open a connection to the MySQL server using the access token.
string connectionString =
$"{
{Environment.GetEnvironmentVariable("AZURE_MYSQL_CONNECTIONSTRING")};Pas
sword={accessToken.Token}";

using var connection = new MySqlConnection(connectionString);
Console.WriteLine("Opening connection using access token...");
await connection.OpenAsync();

// do something
```

Pour obtenir d'autres exemples de code, consultez la section [Connexion à des bases de données Azure à partir d'App Service sans secrets à l'aide d'une identité managée](#).

Étapes suivantes

Suivez les documentations pour en savoir plus sur le Connecter or de service.

[En savoir plus sur les concepts Service Connector](#)

Intégrer Azure Database pour PostgreSQL avec le Connecteur de services

Article • 02/02/2024

Cette page affiche les méthodes d'authentification et les clients pris en charge, et affiche des exemples de code que vous pouvez utiliser pour connecter Azure Database pour PostgreSQL à d'autres services cloud à l'aide du service Connecter or. Il se peut que vous puissiez vous connecter à Azure Database pour PostgreSQL dans d'autres langages de programmation sans utiliser le Connecteur de services. Cette page présente également les noms et les valeurs des variables d'environnement par défaut (ou configuration Spring Boot) que vous obtenez lorsque vous créez des connexions de service.

Service de calcul pris en charge

Le Connecter or de service peut être utilisé pour connecter les services de calcul suivants à Azure Database pour PostgreSQL :

- Azure App Service
- Azure Functions
- Azure App Configuration
- Azure Spring Apps

Types d'authentification et de clients pris en charge

Le tableau ci-dessous montre quelles combinaisons de méthodes d'authentification et de clients sont prises en charge pour connecter votre service de calcul à Azure Database pour PostgreSQL à l'aide du Connecter or de service. Un « Oui » indique que la combinaison est prise en charge, tandis qu'une valeur « Non » indique qu'elle n'est pas prise en charge.

[Agrandir le tableau](#)

Type de client	Identité managée affectée par le système	Identité managée affectée par l'utilisateur	Secret/chaîne de connexion	Principal du service
.NET	Oui	Oui	Oui	Oui

Type de client Go (PG)	Identité managée Oui affectée par le système	Identité managée Oui affectée par l'utilisateur	Secret/chaîne de Oui connexion	Principal Oui du service
Java (JDBC)	Oui	Oui	Oui	Oui
Java – Spring Boot (JDBC)	Oui	Oui	Oui	Oui
Node.js (pg)	Oui	Oui	Oui	Oui
PHP (natif)	Oui	Oui	Oui	Oui
Python (psycopg2)	Oui	Oui	Oui	Oui
Python-Django	Oui	Oui	Oui	Oui
Ruby (ruby-pg)	Oui	Oui	Oui	Oui
Aucun	Oui	Oui	Oui	Oui

Ce tableau indique que toutes les combinaisons de types clients et de méthodes d'authentification dans la table sont prises en charge. Tous les types de clients peuvent utiliser l'une des méthodes d'authentification pour se connecter à Azure Database pour PostgreSQL à l'aide du Connecter or de service.

ⓘ Notes

L'identité managée affectée par le système, l'identité managée affectée par l'utilisateur et le principal de service sont uniquement pris en charge sur Azure CLI.

Noms de variables d'environnement par défaut ou propriétés d'application et exemple de code

Référez les détails de connexion et l'exemple de code dans les tableaux suivants, en fonction du type d'authentification et du type de client de votre connexion, pour connecter les services de calcul à Azure Database pour PostgreSQL. Pour plus d'informations sur les conventions d'affectation de noms, case activée l'article interne [du service Connecter or](#).

Identité managée affectée par le système

.NET

[+] Agrandir le tableau

Nom de variable d'environnement par défaut	Description	Exemple de valeur
AZURE_POSTGRESQL_CONNECTIONSTRING	Chaîne de connexion .NET PostgreSQL	Server=<PostgreSQL-server-name>.postgres.database.azure.com;Database=<database-name>;Port=5432;SslMode=Require;User Id=<username>;

Exemple de code

Reportez-vous aux étapes et au code ci-dessous pour vous connecter à Azure Database pour PostgreSQL à l'aide d'une identité managée affectée par le système.

.NET

Pour .NET, il n'existe pas de plug-in ou de bibliothèque pour prendre en charge les connexions sans mot de passe. Vous pouvez obtenir un jeton d'accès pour l'identité managée ou le principal de service à l'aide d'une bibliothèque cliente telle que [Azure.Identity](#). Vous pouvez ensuite utiliser le jeton d'accès comme mot de passe pour vous connecter à la base de données. Lorsque vous utilisez le code ci-dessous, supprimez les marques de commentaire de la partie de l'extrait de code pour le type d'authentification que vous souhaitez utiliser.

C#

```
using Azure.Identity;
using Azure.Core;
using Npgsql;

// Uncomment the following lines according to the authentication type.
// For system-assigned identity.
// var sqlServerTokenProvider = new DefaultAzureCredential();

// For user-assigned identity.
// var sqlServerTokenProvider = new DefaultAzureCredential(
//     new DefaultAzureCredentialOptions
//     {
//         ManagedIdentityClientId =
// Environment.GetEnvironmentVariable("AZURE_POSTGRESQL_CLIENTID");
//     }
// );

// For service principal.
```

```

// var tenantId =
Environment.GetEnvironmentVariable("AZURE_POSTGRESQL_TENANTID");
// var clientId =
Environment.GetEnvironmentVariable("AZURE_POSTGRESQL_CLIENTID");
// var clientSecret =
Environment.GetEnvironmentVariable("AZURE_POSTGRESQL_CLIENTSECRET");
// var sqlServerTokenProvider = new ClientSecretCredential(tenantId,
clientId, clientSecret);

// Acquire the access token.
AccessToken accessToken = await sqlServerTokenProvider.GetTokenAsync(
    new TokenRequestContext(scopes: new string[]
    {
        "https://osssrdbms-aad.database.windows.net/.default"
    }));
}

// Combine the token with the connection string from the environment
variables provided by Service Connector.
string connectionString =
$"{
{Environment.GetEnvironmentVariable("AZURE_POSTGRESQL_CONNECTIONSTRING")
};Password={accessToken.Token}";

// Establish the connection.
using (var connection = new NpgsqlConnection(connectionString))
{
    Console.WriteLine("Opening connection using access token...");
    connection.Open();
}

```

Ensuite, si vous avez créé des tables et des séquences dans un serveur flexible PostgreSQL avant d'utiliser le connecteur de services, vous devez vous connecter en tant que propriétaire et accorder à <aad-username> l'autorisation créée par le connecteur de services. Le nom d'utilisateur de la chaîne de connexion ou de la configuration définie par le connecteur de service doit ressembler à <aad_<connection name>>. Si vous utilisez le Portail Azure, sélectionnez le bouton Développer à côté de la colonne **Service Type** et obtenez la valeur. Si vous utilisez Azure CLI, cochez **configurations** dans la sortie de la commande CLI.

Ensuite, exécutez la requête pour accorder l'autorisation

Azure CLI

```

az extension add --name rdbms-connect

az postgres flexible-server execute -n <postgres-name> -u <owner-username> -
p "<owner-password>" -d <database-name> --querytext "GRANT ALL PRIVILEGES ON

```

```
ALL TABLES IN SCHEMA public TO \"<aad-username>\";GRANT ALL PRIVILEGES ON  
ALL SEQUENCES IN SCHEMA public TO \"<aad username>\";"
```

<owner-username> et <owner-password> est le propriétaire de la table existante qui peut accorder des autorisations à d'autres personnes. <aad-username> est l'utilisateur créé par Service Connector. Remplacez-les par la valeur réelle.

Validez le résultat avec la commande suivante :

Azure CLI

```
az postgres flexible-server execute -n <postgres-name> -u <owner-username> -  
p "<owner-password>" -d <database-name> --querytext "SELECT  
distinct(table_name) FROM information_schema.table_privileges WHERE  
grantee='<aad-username>' AND table_schema='public';" --output table
```

Identité managée affectée par l'utilisateur

.NET

[Agrandir le tableau](#)

Nom de variable d'environnement par défaut	Description	Exemple de valeur
AZURE_POSTGRESQL_CLIENTID	Votre ID client	<identity-client-ID>
AZURE_POSTGRESQL_CONNECTIONSTRING	Chaîne de connexion .NET PostgreSQL	Server=<PostgreSQL-server-name>.postgres.database.azure.com;Database=<database-name>;Port=5432;Ssl Mode=Require;User Id=<username>;

Exemple de code

Reportez-vous aux étapes et au code ci-dessous pour vous connecter à Azure Database pour PostgreSQL à l'aide d'une identité managée affectée par l'utilisateur.

.NET

Pour .NET, il n'existe pas de plug-in ou de bibliothèque pour prendre en charge les connexions sans mot de passe. Vous pouvez obtenir un jeton d'accès pour l'identité

managée ou le principal de service à l'aide d'une bibliothèque cliente telle que [Azure.Identity](#). Vous pouvez ensuite utiliser le jeton d'accès comme mot de passe pour vous connecter à la base de données. Lorsque vous utilisez le code ci-dessous, supprimez les marques de commentaire de la partie de l'extrait de code pour le type d'authentification que vous souhaitez utiliser.

C#

```
using Azure.Identity;
using Azure.Core;
using Npgsql;

// Uncomment the following lines according to the authentication type.
// For system-assigned identity.
// var sqlServerTokenProvider = new DefaultAzureCredential();

// For user-assigned identity.
// var sqlServerTokenProvider = new DefaultAzureCredential(
//     new DefaultAzureCredentialOptions
//     {
//         ManagedIdentityClientId =
// Environment.GetEnvironmentVariable("AZURE_POSTGRESQL_CLIENTID");
//     }
// );

// For service principal.
// var tenantId =
// Environment.GetEnvironmentVariable("AZURE_POSTGRESQL_TENANTID");
// var clientId =
// Environment.GetEnvironmentVariable("AZURE_POSTGRESQL_CLIENTID");
// var clientSecret =
// Environment.GetEnvironmentVariable("AZURE_POSTGRESQL_CLIENTSECRET");
// var sqlServerTokenProvider = new ClientSecretCredential(tenantId,
// clientId, clientSecret);

// Acquire the access token.
AccessToken accessToken = await sqlServerTokenProvider.GetTokenAsync(
    new TokenRequestContext(scopes: new string[]
    {
        "https://osssrdbms-aad.database.windows.net/.default"
    }));
    
// Combine the token with the connection string from the environment
variables provided by Service Connector.
string connectionString =
$"{
{Environment.GetEnvironmentVariable("AZURE_POSTGRESQL_CONNECTIONSTRING")
};Password={accessToken.Token}";

// Establish the connection.
using (var connection = new NpgsqlConnection(connectionString))
{
    Console.WriteLine("Opening connection using access token...");
```

```
        connection.Open();  
    }
```

Ensuite, si vous avez créé des tables et des séquences dans un serveur flexible PostgreSQL avant d'utiliser le connecteur de services, vous devez vous connecter en tant que propriétaire et accorder à `<aad-username>` l'autorisation créée par le connecteur de services. Le nom d'utilisateur de la chaîne de connexion ou de la configuration définie par le connecteur de service doit ressembler à `aad_<connection name>`. Si vous utilisez le Portail Azure, sélectionnez le bouton Développer à côté de la colonne `Service Type` et obtenez la valeur. Si vous utilisez Azure CLI, cochez `configurations` dans la sortie de la commande CLI.

Ensuite, exécutez la requête pour accorder l'autorisation

```
Azure CLI  
  
az extension add --name rdbms-connect  
  
az postgres flexible-server execute -n <postgres-name> -u <owner-username> -  
p "<owner-password>" -d <database-name> --querytext "GRANT ALL PRIVILEGES ON  
ALL TABLES IN SCHEMA public TO \"<aad-username>\";GRANT ALL PRIVILEGES ON  
ALL SEQUENCES IN SCHEMA public TO \"<aad username>\";"
```

`<owner-username>` et `<owner-password>` est le propriétaire de la table existante qui peut accorder des autorisations à d'autres personnes. `<aad-username>` est l'utilisateur créé par Service Connector. Remplacez-les par la valeur réelle.

Validez le résultat avec la commande suivante :

```
Azure CLI  
  
az postgres flexible-server execute -n <postgres-name> -u <owner-username> -  
p "<owner-password>" -d <database-name> --querytext "SELECT  
distinct(table_name) FROM information_schema.table_privileges WHERE  
grantee='<aad-username>' AND table_schema='public';" --output table
```

Chaîne de connexion

.NET

[+] Agrandir le tableau

Nom de variable d'environnement par défaut	Description	Exemple de valeur
AZURE_POSTGRESQL_CONNECTIONSTRING	Chaîne de connexion .NET PostgreSQL	Server=<PostgreSQL-server-name>.postgres.database.azure.com;Database=<database-name>;Port=5432;SslMode=Require;User Id=<username>;

Exemple de code

Reportez-vous aux étapes et au code ci-dessous pour vous connecter à Azure Database pour PostgreSQL à l'aide d'un chaîne de connexion.

.NET

1. Installez des dépendances. Suivez les instructions pour [installer Npgsql](#).
2. Dans le code, obtenez le chaîne de connexion PostgreSQL à partir des variables d'environnement ajoutées par le service Connecter or. Pour définir des configurations TSL pour le serveur PostgreSQL, reportez-vous à [ces étapes](#).

C#

```
using System;
using Npgsql;

string connectionString =
Environment.GetEnvironmentVariable("AZURE_POSTGRESQL_CONNECTIONSTRING");
using (NpgsqlConnection connection = new
NpgsqlConnection(connectionString))
{
    connection.Open();
}
```

Principal du service

.NET

[+] Agrandir le tableau

Nom de variable d'environnement par défaut	Description	Exemple de valeur
AZURE_POSTGRESQL_CLIENTID	Votre ID client	<client-ID>
AZURE_POSTGRESQL_CLIENTSECRET	Votre clé secrète client	<client-secret>
AZURE_POSTGRESQL_TENANTID	Votre ID d'abonné	<tenant-ID>
AZURE_POSTGRESQL_CONNECTIONSTRING	Chaîne de connexion .NET PostgreSQL	Server=<PostgreSQL-server-name>.postgres.database.azure.com;Database=<database-name>;Port=5432;SslMode=Require;User Id=<username>;

Exemple de code

Reportez-vous aux étapes et au code ci-dessous pour vous connecter à Azure Database pour PostgreSQL à l'aide d'un principal de service.

.NET

Pour .NET, il n'existe pas de plug-in ou de bibliothèque pour prendre en charge les connexions sans mot de passe. Vous pouvez obtenir un jeton d'accès pour l'identité managée ou le principal de service à l'aide d'une bibliothèque cliente telle que [Azure.Identity](#). Vous pouvez ensuite utiliser le jeton d'accès comme mot de passe pour vous connecter à la base de données. Lorsque vous utilisez le code ci-dessous, supprimez les marques de commentaire de la partie de l'extrait de code pour le type d'authentification que vous souhaitez utiliser.

C#

```
using Azure.Identity;
using Azure.Core;
using Npgsql;

// Uncomment the following lines according to the authentication type.
// For system-assigned identity.
// var sqlServerTokenProvider = new DefaultAzureCredential();

// For user-assigned identity.
// var sqlServerTokenProvider = new DefaultAzureCredential(
//     new DefaultAzureCredentialOptions
```

```

//      {
//          ManagedIdentityClientId =
Environment.GetEnvironmentVariable("AZURE_POSTGRESQL_CLIENTID");
//      }
// );

// For service principal.
// var tenantId =
Environment.GetEnvironmentVariable("AZURE_POSTGRESQL_TENANTID");
// var clientId =
Environment.GetEnvironmentVariable("AZURE_POSTGRESQL_CLIENTID");
// var clientSecret =
Environment.GetEnvironmentVariable("AZURE_POSTGRESQL_CLIENTSECRET");
// var sqlServerTokenProvider = new ClientSecretCredential(tenantId,
clientId, clientSecret);

// Acquire the access token.
AccessToken accessToken = await sqlServerTokenProvider.GetTokenAsync(
    new TokenRequestContext(scopes: new string[]
{
    "https://osssrdbms-aad.database.windows.net/.default"
}));


// Combine the token with the connection string from the environment
variables provided by Service Connector.
string connectionString =
$"{
{Environment.GetEnvironmentVariable("AZURE_POSTGRESQL_CONNECTIONSTRING")
};Password={accessToken.Token}";

// Establish the connection.
using (var connection = new NpgsqlConnection(connectionString))
{
    Console.WriteLine("Opening connection using access token...");
    connection.Open();
}

```

Ensute, si vous avez créé des tables et des séquences dans un serveur flexible PostgreSQL avant d'utiliser le connecteur de services, vous devez vous connecter en tant que propriétaire et accorder à <aad-username> l'autorisation créée par le connecteur de services. Le nom d'utilisateur de la chaîne de connexion ou de la configuration définie par le connecteur de service doit ressembler à `aad_<connection name>`. Si vous utilisez le Portail Azure, sélectionnez le bouton Développer à côté de la colonne `Service Type` et obtenez la valeur. Si vous utilisez Azure CLI, cochez `configurations` dans la sortie de la commande CLI.

Ensute, exécutez la requête pour accorder l'autorisation

```
az extension add --name rdbms-connect

az postgres flexible-server execute -n <postgres-name> -u <owner-username> -
p "<owner-password>" -d <database-name> --querytext "GRANT ALL PRIVILEGES ON
ALL TABLES IN SCHEMA public TO \"<aad-username>\";GRANT ALL PRIVILEGES ON
ALL SEQUENCES IN SCHEMA public TO \"<aad username>\";"
```

<owner-username> et <owner-password> est le propriétaire de la table existante qui peut accorder des autorisations à d'autres personnes. <aad-username> est l'utilisateur créé par Service Connector. Remplacez-les par la valeur réelle.

Validez le résultat avec la commande suivante :

Azure CLI

```
az postgres flexible-server execute -n <postgres-name> -u <owner-username> -
p "<owner-password>" -d <database-name> --querytext "SELECT
distinct(table_name) FROM information_schema.table_privileges WHERE
grantee='<aad-username>' AND table_schema='public';" --output table
```

Étapes suivantes

Suivez les tutoriels ci-dessous pour en savoir plus sur Service Connector.

[En savoir plus sur les concepts Service Connector](#)

Configurer des connexions sans mot de passe entre plusieurs applications et services Azure

Article • 25/03/2023

Les applications nécessitent souvent des connexions sécurisées entre plusieurs services Azure simultanément. Par exemple, une instance Azure App Service d'entreprise peut se connecter à plusieurs comptes de stockage différents, une instance de base de données Azure SQL, un service bus, etc.

[Les identités managées](#) sont l'option d'authentification recommandée pour les connexions sécurisées sans mot de passe entre les ressources Azure. Les développeurs n'ont pas à suivre et gérer manuellement de nombreux secrets différents pour les identités managées, car la plupart de ces tâches sont gérées en interne par Azure. Ce didacticiel explique comment gérer les connexions entre plusieurs services à l'aide d'identités managées et de la bibliothèque cliente Azure Identity.

Comparer les types d'identités managées

Azure fournit les types d'identités managées suivantes :

- Les **identités managées affectées par le système** sont directement liées à une seule ressource Azure. Lorsque vous activez une identité managée affectée par le système sur un service, Azure crée une identité liée et gère les tâches administratives pour cette identité en interne. Quand la ressource Azure est supprimée, l'identité l'est aussi.
- Les **identités managées affectées par l'utilisateur** sont des identités indépendantes créées par un administrateur et peuvent être associées à une ou plusieurs ressources Azure. Le cycle de vie de l'identité est indépendant de ces ressources.

Vous pouvez en savoir plus sur les meilleures pratiques et quand utiliser des identités affectées par le système et des identités affectées par l'utilisateur dans les [recommandations de meilleures pratiques des identités](#).

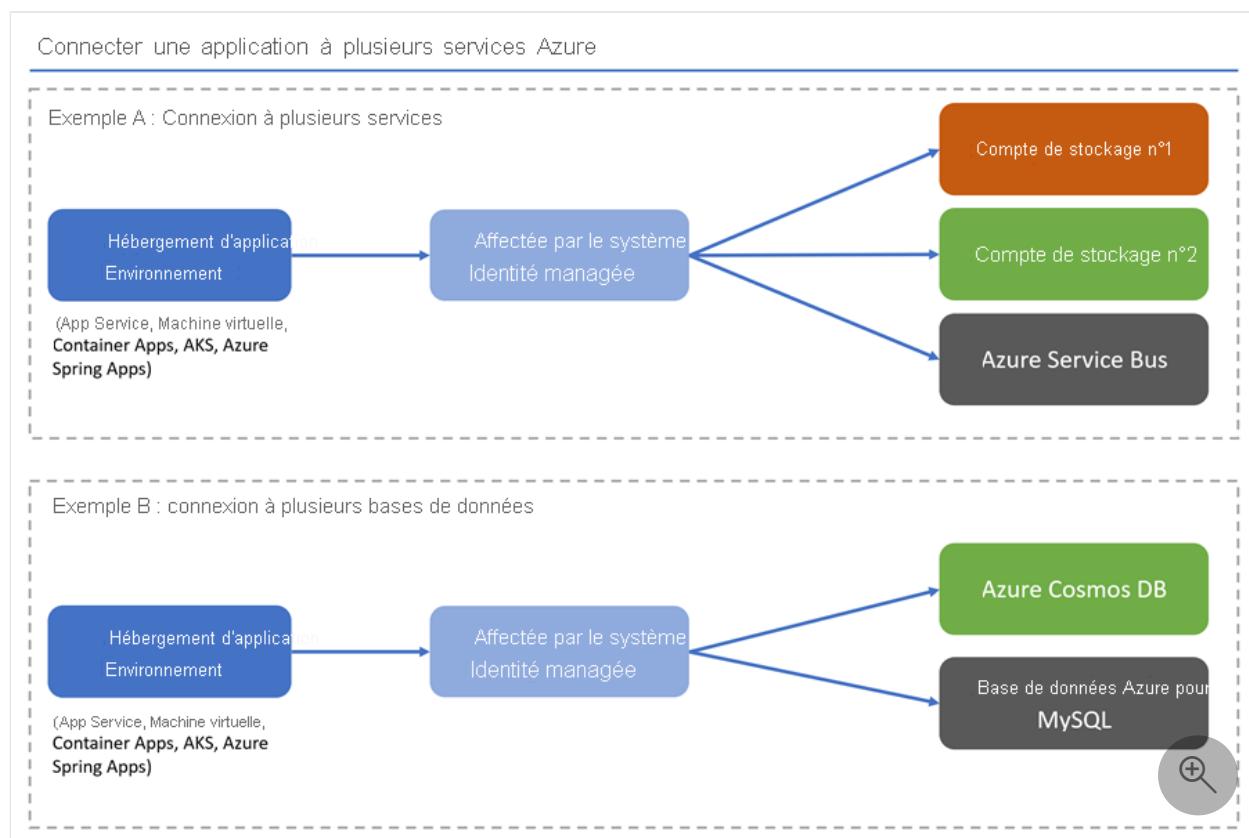
Explorer DefaultAzureCredential

Les identités managées sont généralement implémentées dans votre code d'application via une classe appelée `DefaultAzureCredential` à partir de la `Azure.Identity` bibliothèque cliente. `DefaultAzureCredential` prend en charge plusieurs méthodes d'authentification et détermine automatiquement celle qui doit être utilisée au moment de l'exécution. Vous pouvez en savoir plus sur cette approche dans la [vue d'ensemble defaultAzureCredential](#).

Connecter une application hébergée Azure à plusieurs services Azure

Vous avez été chargé de connecter une application existante à plusieurs services et bases de données Azure à l'aide de connexions sans mot de passe. L'application est une API web ASP.NET Core hébergée sur Azure App Service, bien que les étapes ci-dessous s'appliquent également à d'autres environnements d'hébergement Azure, tels qu'Azure Spring Apps, Machines Virtuelles, Container Apps et AKS.

Ce didacticiel s'applique aux architectures suivantes, bien qu'il puisse être adapté à de nombreux autres scénarios ainsi qu'à des modifications de configuration minimales.



Les étapes suivantes montrent comment configurer une application pour utiliser une identité managée affectée par le système et votre compte de développement local pour se connecter à plusieurs services Azure.

Créer une identité managée affectée par le système

1. Dans le Portail Azure, accédez à l'application hébergée que vous souhaitez vous connecter à d'autres services.
2. Sur la page vue d'ensemble du service, sélectionnez **Identité**.
3. Basculez le paramètre d'**Etat** sur **Activé** pour activer une identité managée affectée par le système pour le service.

The screenshot shows the Azure portal interface for managing an App Service named 'msdocs-web-app-123'. On the left, a navigation sidebar lists various service management options like 'Vue d'ensemble', 'Journal d'activité', and 'Identité'. The 'Identité' option is highlighted with a red box. The main content area displays the 'Attribuée par le système' (Assigned by system) tab, which is also highlighted with a red box. A note below states: 'Une identité managée affectée par le système est limitée à une par ressource et liée au cycle de vie de celle-ci. Vous pouvez accorder des autorisations à cette identité.' (A managed identity assigned by system is limited to one per resource and is linked to the lifecycle of that resource. You can grant permissions to this identity.) Below this, there's a 'Statut' (Status) switch set to 'On', and an 'ID de l'objet (principal)' (Object ID (principal)) field containing the value '65634e65-f3eb-4fe7-aff3-fd31d035084b'. At the bottom, a note indicates that the resource is registered in Azure Active Directory and can be configured to allow access to other services.

Attribuer des rôles à l'identité managée pour chaque service connecté

1. Accédez à la page vue d'ensemble du compte de stockage auquel vous souhaitez accorder l'accès à votre identité.
2. Sélectionnez **Contrôle d'accès (IAM)** à partir de la navigation du compte de stockage.
3. Choisissez **+ Ajouter**, puis **Ajouter une attribution de rôle**.

The screenshot shows the 'identitymigrationstorage | Access Control (IAM)' blade. In the top navigation bar, the 'Add role assignment' button is highlighted with a red box. On the left sidebar, the 'Access Control (IAM)' menu item is also highlighted with a red box. The main area contains sections for 'My access', 'Check access', and 'Grant access to this resource'. The 'Grant access to this resource' section includes a 'Add role assignment' button, which is also highlighted with a red box.

4. Dans la zone de recherche de Rôle, recherchez *le Contributeur de données Blob de stockage*, qui accorde des autorisations pour effectuer des opérations de lecture et d'écriture sur les données d'objets blob. Vous pouvez attribuer le rôle approprié pour votre cas d'usage. Sélectionnez le *contributeur de données blob de stockage* dans la liste, puis choisissez **Suivant**.
5. Sur l'écran **Ajouter une attribution de rôle**, pour l'option **Attribuer l'accès à l'option**, sélectionnez **Identité managée**. Choisissez ensuite +Sélectionner des membres.
6. Dans le menu volant, recherchez l'identité managée que vous avez créée en entrant le nom de votre service app. Sélectionnez l'identité affectée par le système, puis choisissez **Sélectionner** pour fermer le menu volant.

The screenshot shows the 'Ajouter une attribution de rôle' (Add role assignment) blade and a modal window titled 'Selectionner des membres' (Select members). The 'Attribuer l'accès à' dropdown is highlighted with a red box (Step 1). The 'Membres' button is highlighted with a red box (Step 2). The search input field in the modal is highlighted with a red box (Step 3). The selected member 'msdocs-web-app-123' is shown in the 'Membres sélectionnés:' list (Step 4). The 'Sélectionner' button is highlighted with a red box (Step 5). The 'Suivant' button at the bottom of the blade is highlighted with a red box (Step 6).

7. Sélectionnez **Suivant** plusieurs fois jusqu'à ce que vous puissiez sélectionner **Vérifier + attribuer** pour terminer l'attribution de rôle.

8. Répétez ce processus pour les autres services auxquels vous souhaitez vous connecter.

Considérations relatives au développement local

Vous pouvez également activer l'accès aux ressources Azure pour le développement local en affectant des rôles à un compte d'utilisateur de la même façon que vous avez attribué des rôles à votre identité managée.

1. Après avoir attribué le rôle **Contributeur de données Blob de stockage** à votre identité managée, sous **Attribuer l'accès à**, cette fois sélectionnez **Utilisateur, groupe ou principal de service**. Choisissez + **Sélectionner des membres** pour ouvrir à nouveau le menu volant.
2. Recherchez le *compte user@domain* ou le groupe de sécurité Microsoft Entra auquel vous souhaitez accorder l'accès par adresse e-mail ou par nom, puis sélectionnez-le. Il doit s'agir du même compte que celui que vous utilisez pour vous connecter à vos outils de développement local, tels que Visual Studio ou Azure CLI.

ⓘ Notes

Vous pouvez également attribuer ces rôles à un groupe de sécurité Microsoft Entra si vous travaillez sur une équipe avec plusieurs développeurs. Vous pouvez ensuite placer n'importe quel développeur dans ce groupe qui a besoin d'accéder au développement de l'application localement.

Implémenter le code de l'application

C#

À l'intérieur de votre projet, ajoutez une référence au package NuGet `Azure.Identity`. Cette bibliothèque contient toutes les entités nécessaires pour implémenter `DefaultAzureCredential`. Vous pouvez également ajouter toutes les autres bibliothèques Azure qui sont pertinentes pour votre application. Pour cet exemple, les packages `Azure.Storage.Blobs` et `Azure.KeyVault.Keys` sont ajoutés pour se connecter au stockage Blob et Key Vault.

CLI .NET

```
dotnet add package Azure.Identity  
dotnet add package Azure.Storage.Blobs  
dotnet add package Azure.KeyVault.Keys
```

En haut de votre fichier `Program.cs`, ajoutez les instructions d'utilisation suivantes :

C#

```
using Azure.Identity;  
using Azure.Storage.Blobs;  
using Azure.Security.KeyVault.Keys;
```

Dans le fichier `Program.cs` de votre code de projet, créez des instances des services nécessaires auxquels votre application se connecte. Les exemples suivants se connectent au stockage Blob et au service bus à l'aide des classes SDK correspondantes.

C#

```
var blobServiceClient = new BlobServiceClient(  
    new Uri("https://<your-storage-account>.blob.core.windows.net"),  
    new DefaultAzureCredential(credOptions));  
  
var serviceBusClient = new ServiceBusClient("<your-namespace>", new  
DefaultAzureCredential());  
var sender = serviceBusClient.CreateSender("producttracking");
```

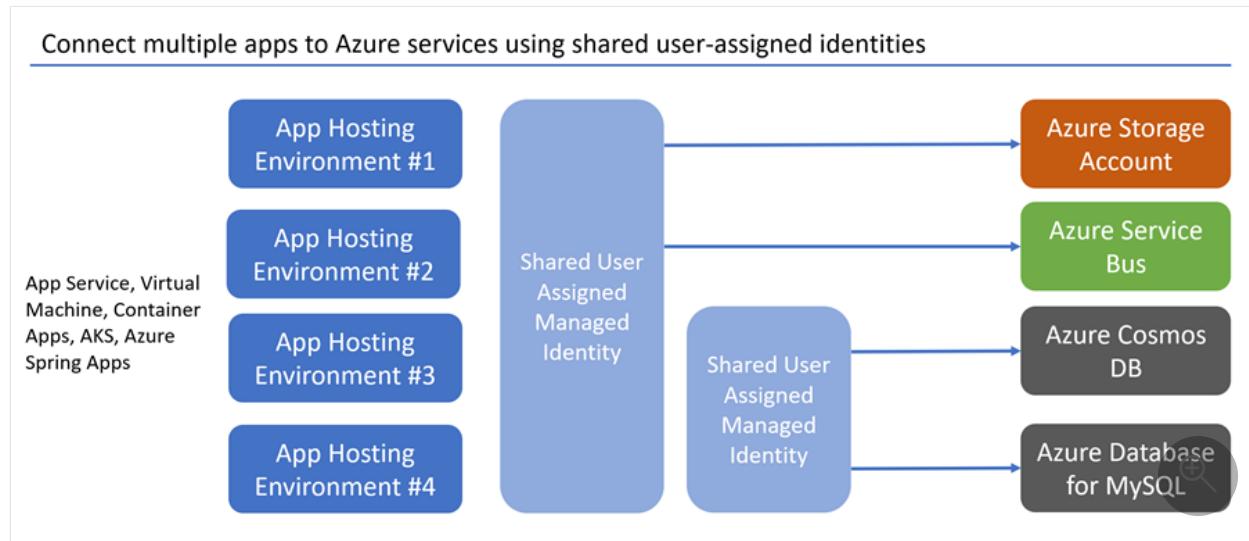
Lorsque ce code d'application s'exécute localement, `DefaultAzureCredential` recherchera une chaîne d'informations d'identification pour les premières informations d'identification disponibles. Si la valeur `Managed_Identity_Client_ID` est null localement, elle utilise automatiquement les informations d'identification de votre interface de ligne de commande Azure locale ou de la connexion Visual Studio. Vous pouvez en savoir plus sur ce processus dans la [vue d'ensemble de la bibliothèque Azure Identity](#).

Lorsque l'application est déployée sur Azure, `DefaultAzureCredential` récupère automatiquement la variable `Managed_Identity_Client_ID` à partir de l'environnement App Service. Cette valeur devient disponible lorsqu'une identité managée est associée à votre application.

Ce processus global garantit que votre application peut s'exécuter en toute sécurité localement et dans Azure sans avoir besoin de modifications de code.

Connectez plusieurs applications à l'aide de plusieurs identités managées

Bien que les applications de l'exemple précédent partagent toutes les mêmes exigences d'accès au service, les environnements réels sont souvent plus nuancés. Envisagez un scénario où plusieurs applications se connectent aux mêmes comptes de stockage, mais deux des applications accèdent également à différents services ou bases de données.



Pour configurer cette configuration dans votre code, assurez-vous que votre application inscrit des services distincts pour se connecter à chaque compte de stockage ou base de données. Veillez à extraire les ID du client d'identité managée corrects pour chaque service lors de la configuration de `DefaultAzureCredential`. L'exemple de code suivant configure les connexions de service suivantes :

- Deux connexions à des comptes de stockage distincts à l'aide d'une identité managée affectée par l'utilisateur partagé
- Connexion aux services Azure Cosmos DB et Azure SQL à l'aide d'une deuxième identité managée affectée par l'utilisateur partagé

C#

C#

```
// Get the first user-assigned managed identity ID to connect to shared storage
var clientIDstorage =
Environment.GetEnvironmentVariable("Managed_Identity_Client_ID_Storage")
;

// First blob storage client that using a managed identity
BlobServiceClient blobServiceClient = new BlobServiceClient(
new Uri("https://<receipt-storage-account>.blob.core.windows.net"),
```

```

        new DefaultAzureCredential()
    {
        ManagedIdentityClientId = clientIDstorage
    });

    // Second blob storage client that using a managed identity
    BlobServiceClient blobServiceClient2 = new BlobServiceClient(
        new Uri("https://<contract-storage-account>.blob.core.windows.net"),
        new DefaultAzureCredential()
    {
        ManagedIdentityClientId = clientIDstorage
    });

    // Get the second user-assigned managed identity ID to connect to shared
    // databases
    var clientIDdatabases =
    Environment.GetEnvironmentVariable("Managed_Identity_Client_ID_Databases"
    ");

    // Create an Azure Cosmos DB client
    CosmosClient client = new CosmosClient(
        accountEndpoint:
    Environment.GetEnvironmentVariable("COSMOS_ENDPOINT",
    EnvironmentVariableTarget.Process),
        new DefaultAzureCredential()
    {
        ManagedIdentityClientId = clientIDdatabases
    });

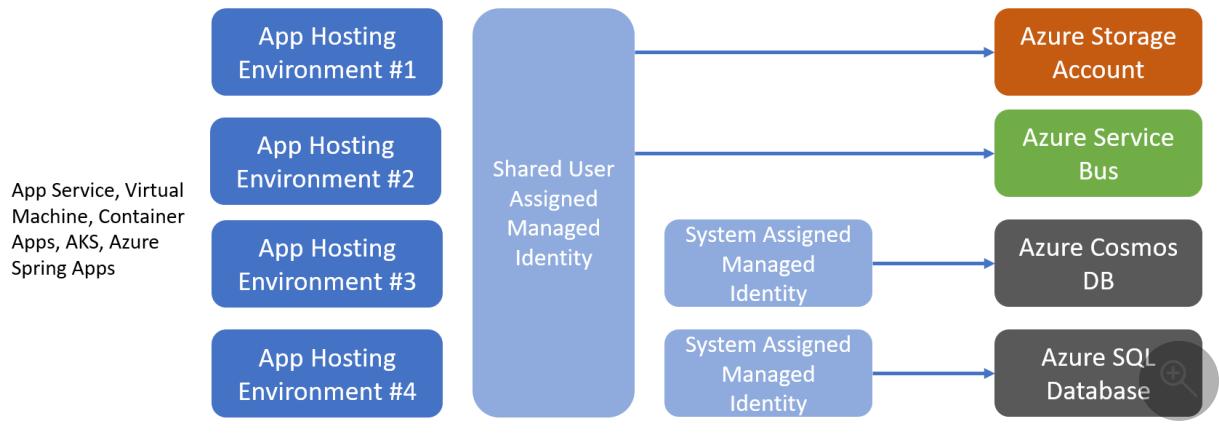
    // Open a connection to Azure SQL using a managed identity
    string ConnectionString1 = @"Server=<azure-sql-
    hostname>.database.windows.net; User Id=ObjectIdOfManagedIdentity;
    Authentication=Active Directory Default; Database=<database-name>";

    using (SqlConnection conn = new SqlConnection(ConnectionString1))
    {
        conn.Open();
    }

```

Vous pouvez également associer une identité managée affectée par l'utilisateur ainsi qu'une identité managée affectée par le système à une ressource simultanément. Cela peut être utile dans les scénarios où toutes les applications nécessitent l'accès aux mêmes services partagés, mais l'une des applications a également une dépendance très spécifique sur un service supplémentaire. L'utilisation d'une identité affectée par le système garantit également que l'identité liée à cette application spécifique est supprimée lorsque l'application est supprimée, ce qui peut vous aider à nettoyer votre environnement.

Connect multiple apps to Azure services using user-assigned and system-assigned identities



Ces types de scénarios sont explorés plus en détail dans les [recommandations de meilleures pratiques des identités](#).

Étapes suivantes

Dans ce didacticiel, vous avez appris à migrer une application vers des connexions sans mot de passe. Vous pouvez lire les ressources suivantes pour explorer les concepts abordés dans cet article plus en détails :

- [Autoriser l'accès aux objets blob à l'aide Microsoft Entra ID](#)
- Pour en savoir plus sur .NET Core, consultez [Prise en main de .NET en 10 minutes](#).

Configurer des identités managées pour les ressources Azure sur une machine virtuelle en utilisant le portail Azure

Article • 28/10/2023

Les identités managées pour les ressources Azure sont une fonctionnalité de Microsoft Entra ID. Les [services Azure prenant en charge les identités managées pour les ressources Azure](#) sont soumis à leur propre chronologie. Assurez-vous de passer en revue l'état [Disponibilité](#) des identités gérées pour votre ressource et les [problèmes connus](#) avant de commencer.

Les identités managées pour les ressources Azure fournissent aux services Azure une identité automatiquement gérée dans Microsoft Entra ID. Vous pouvez utiliser cette identité pour vous authentifier auprès de n'importe quel service prenant en charge l'authentification Microsoft Entra, sans avoir d'information d'identification dans votre code.

Dans cet article, vous allez apprendre à activer et à désactiver des identités managées affectées par le système et par l'utilisateur pour une machine virtuelle Azure en utilisant le portail Microsoft Azure.

Conditions préalables requises

- Si vous n'êtes pas familiarisé avec les identités managées pour les ressources Azure, consultez la [section Vue d'ensemble](#).
- Si vous n'avez pas encore de compte Azure, [inscrivez-vous à un essai gratuit](#) avant de continuer.

Identité managée affectée par le système

Dans cette section, vous allez découvrir comment activer et désactiver l'identité affectée par le système pour une machine virtuelle en utilisant le portail Microsoft Azure.

Activer une identité managée affectée par le système lors de la création d'une machine virtuelle

Pour activer l'identité managée affectée par le système sur une machine virtuelle durant sa création, votre compte a besoin de l'affectation de rôle [Contributeur d'identité](#)

managée. Aucune autre attribution de rôle de répertoire Microsoft Entra n'est requise.

- Sous l'onglet **Gestion** dans la section **Identité**, basculez le commutateur **Identité du service managé** sur **Activé**.

Create a virtual machine

Basics Disks Networking **Management** Guest config Tags Review + create

Configure monitoring and management options for your VM.

MONITORING

Boot diagnostics On Off

OS guest diagnostics On Off

* Diagnostics storage account

IDENTITY

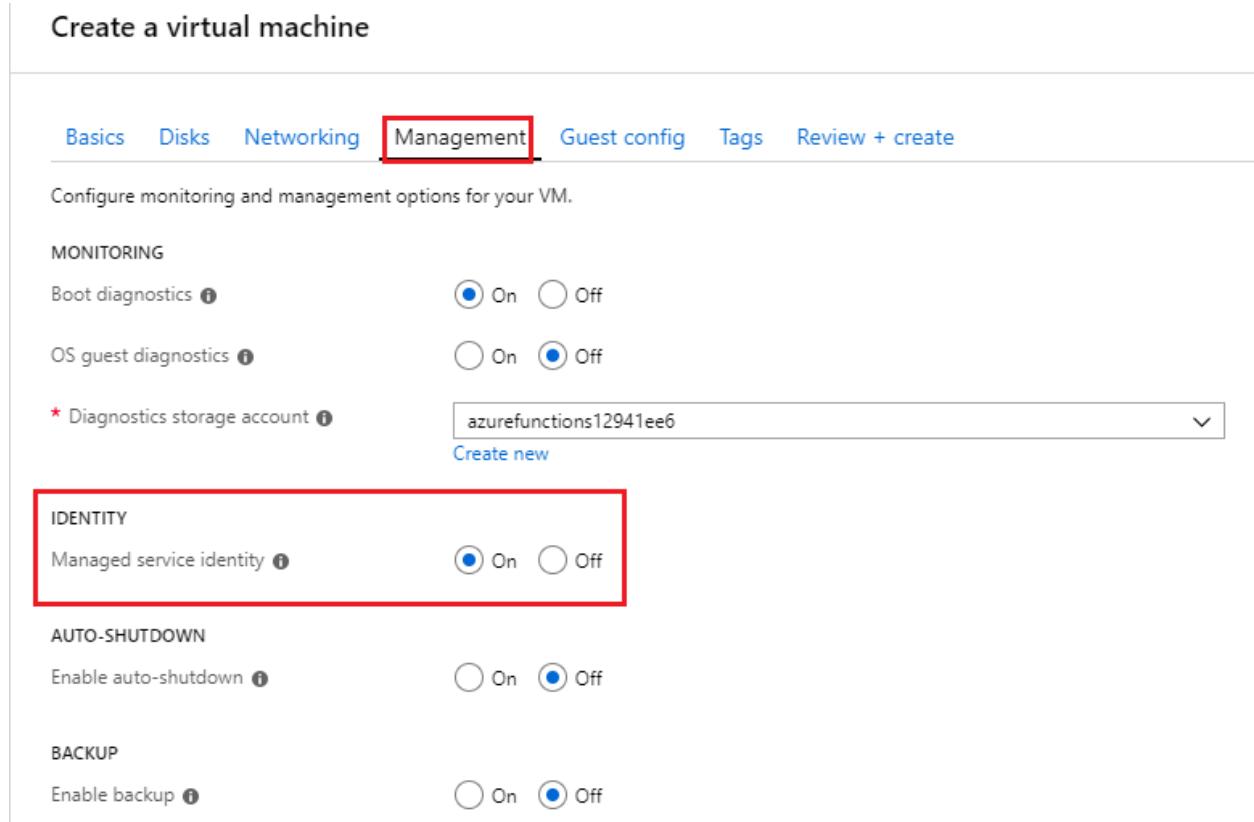
Managed service identity On Off

AUTO-SHUTDOWN

Enable auto-shutdown On Off

BACKUP

Enable backup On Off



Pour créer une machine virtuelle, consultez les Démarrages rapides suivants :

- Créer une machine virtuelle Windows avec le portail Azure
- Créer une machine virtuelle Linux avec le portail Azure

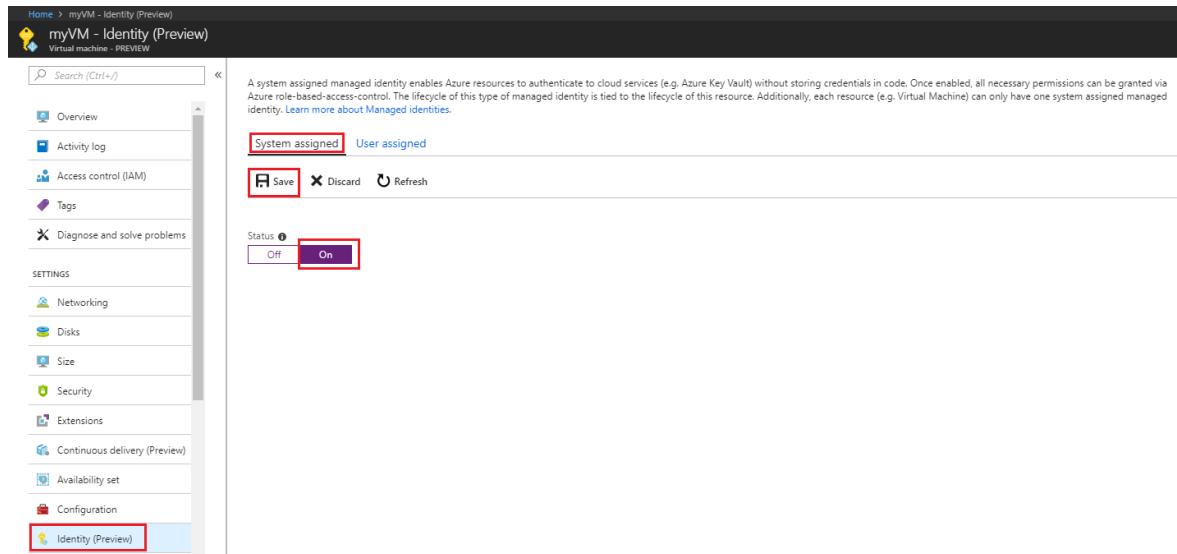
Activer une identité managée affectée par le système sur une machine virtuelle existante

💡 Conseil

Les étapes décrites dans cet article peuvent varier légèrement en fonction du portail de départ.

Pour activer l'identité managée affectée par le système sur une machine virtuelle qui en était dépourvue initialement, votre compte a besoin de l'affectation de rôle **Contributeur d'identité managée**. Aucune autre attribution de rôle de répertoire Microsoft Entra n'est requise.

1. Connectez-vous au [portail Azure](#) à l'aide d'un compte associé à l'abonnement Azure qui contient la machine virtuelle.
2. Accédez à la machine virtuelle souhaitée et sélectionnez **Identité**.
3. Sous **Attribuée par le système**, **État**, sélectionnez **Activé** puis cliquez sur **Enregistrer** :

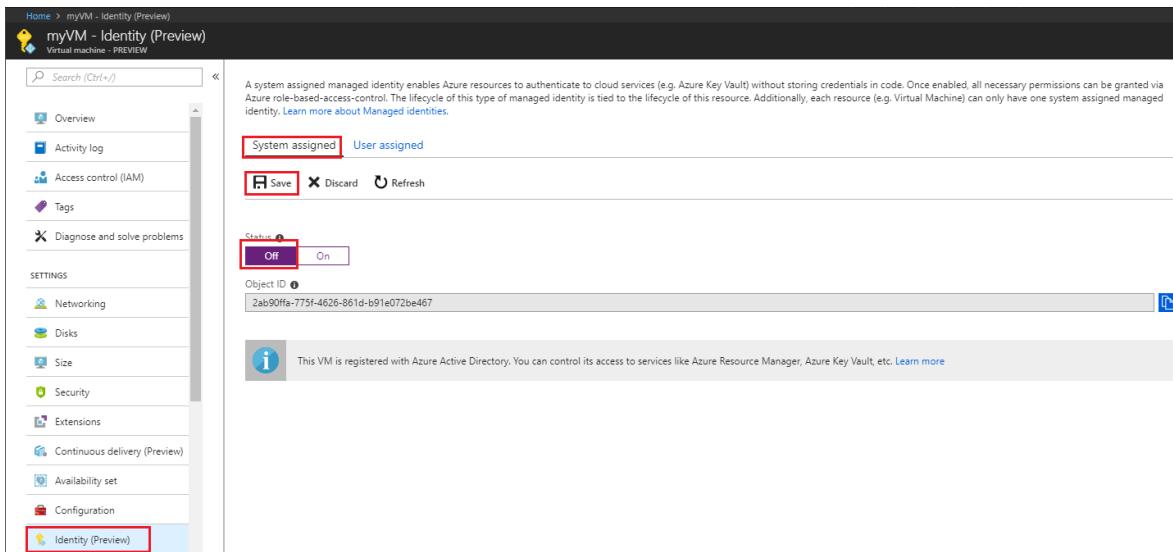


Supprimer une identité managée affectée par le système d'une machine virtuelle

Pour supprimer l'identité managée affectée par le système d'une machine virtuelle, votre compte a besoin de l'affectation de rôle **Contributeur d'identité managée**. Aucune autre attribution de rôle de répertoire Microsoft Entra n'est requise.

Si vous disposez d'une machine virtuelle qui n'a plus besoin d'identité managée affectée par le système :

1. Connectez-vous au [portail Azure](#) à l'aide d'un compte associé à l'abonnement Azure qui contient la machine virtuelle.
2. Accédez à la machine virtuelle souhaitée et sélectionnez **Identité**.
3. Sous **Attribuée par le système**, **État**, sélectionnez **Désactivé** puis cliquez sur **Enregistrer** :



Identité managée affectée par l'utilisateur

Dans cette section, vous allez découvrir comment ajouter et supprimer une identité managée affectée par l'utilisateur sur une machine virtuelle en utilisant le portail Azure.

Attribuer une identité affectée par l'utilisateur lors de la création d'une machine virtuelle

Pour affecter une identité managée affectée par l'utilisateur à une machine virtuelle, votre compte a besoin de l'affectation de rôle [Opérateur d'identité managée](#) et [Contributeur d'identité managée](#). Aucune autre attribution de rôle de répertoire Microsoft Entra n'est requise.

Actuellement, le portail Microsoft Azure ne prend pas en charge l'attribution d'une identité managée affectée par l'utilisateur lors de la création d'une machine virtuelle. Au lieu de cela, reportez-vous à l'un des articles de démarrage rapide suivants pour créer une machine virtuelle, puis passez à la section suivante pour obtenir des informations détaillées sur l'attribution d'une identité managée affectée par l'utilisateur à la machine virtuelle :

- [Créer une machine virtuelle Windows avec le portail Azure](#)
- [Créer une machine virtuelle Linux avec le portail Azure](#)

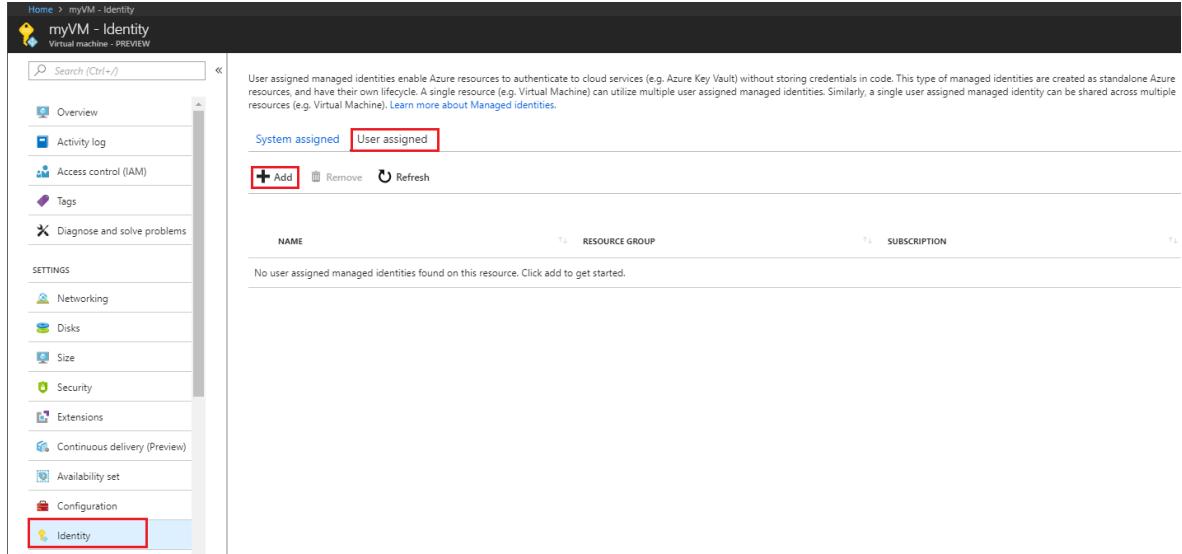
Attribuer une identité managée affectée par l'utilisateur à une machine virtuelle existante

Pour affecter une identité managée affectée par l'utilisateur à une machine virtuelle, votre compte a besoin de l'affectation de rôle [Opérateur d'identité managée](#) et

Contributeur d'identité managée. Aucune autre attribution de rôle de répertoire Microsoft Entra n'est requise.

1. Connectez-vous au [portail Azure](#) à l'aide d'un compte associé à l'abonnement Azure qui contient la machine virtuelle.

2. Accédez à la machine virtuelle souhaitée, cliquez sur **Identité, Attribuée par l'utilisateur**, puis sur **+Ajouter**.



The screenshot shows the Azure portal interface for managing identities on a virtual machine named 'myVM'. The left sidebar lists various settings like Overview, Activity log, IAM, Tags, and Identity. The 'Identity' option is highlighted with a red box. The main pane shows the 'User assigned' tab selected under 'System assigned' and 'User assigned'. It includes a 'Search (Ctrl+R)' bar, a toolbar with 'Add', 'Remove', and 'Refresh' buttons, and a table header for 'NAME', 'RESOURCE GROUP', and 'SUBSCRIPTION'. A message at the top explains user-assigned managed identities. Below the table, it says 'No user assigned managed identities found on this resource. Click add to get started.'

3. Cliquez sur l'identité managée affectée par l'utilisateur que vous souhaitez ajouter à la machine virtuelle, puis cliquez sur **Ajouter**.

Add user assigned managed identity

PREVIEW

* Subscription

▼

User assigned managed identities

 ID1	Resource Group: TestRG
 cyibarravmexid	Resource Group: cyibarratester
 devtestmsi	Resource Group: DevTest
 platApplIdentity	Resource Group: DevTest
 MKTG-UA-01	Resource Group: MARKETING-PROD
 MKTG-UA-02	Resource Group: MARKETING-PROD

Selected identities:

 ID1	TestRG	Remove
---	--------	--------

Add

Supprimer une identité managée affectée par l'utilisateur d'une machine virtuelle

Pour supprimer une identité affectée par l'utilisateur d'une machine virtuelle, votre compte a besoin de l'affectation de rôle [Contributeur d'identité managée](#). Aucune autre attribution de rôle de répertoire Microsoft Entra n'est requise.

1. Connectez-vous au [portail Azure](#) à l'aide d'un compte associé à l'abonnement Azure qui contient la machine virtuelle.

2. Accédez à la machine virtuelle souhaitée, sélectionnez **Identité**, Affectée par l'utilisateur, sélectionnez le nom de l'identité managée affectée par l'utilisateur que vous voulez supprimer, puis cliquez sur **Supprimer** (cliquez sur **Oui** dans le volet de confirmation).

The screenshot shows the Azure portal interface for managing identities of a virtual machine named 'myVM'. The left sidebar contains navigation links for Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, SETTINGS (Networking, Disks, Size, Security, Extensions, Continuous delivery (Preview), Availability set, Configuration), and Identity. The 'Identity' link is highlighted with a red box. The main content area displays the 'User assigned' identities. A table lists identities with columns for NAME, RESOURCE GROUP, and SUBSCRIPTION. One identity, 'ID1', is selected and highlighted with a red box. The 'Remove' button is also highlighted with a red box.

NAME	RESOURCE GROUP	SUBSCRIPTION
ID1	TestRG	<SUBSCRIPTION ID>

Étapes suivantes

- Utilisez le portail Azure pour accorder à une identité managée de machine virtuelle Azure l'accès à une autre ressource Azure.

Se connecter à Azure SQL Database et l'interroger à l'aide de .NET et Entity Framework Core

Article • 16/05/2023

S'applique à  [Azure SQL Database](#)

Ce démarrage rapide explique comment connecter une application à une base de données dans Azure SQL Database et effectuer des requêtes à l'aide de .NET et Entity Framework Core. Ce guide de démarrage rapide suit l'approche sans mot de passe recommandée pour se connecter à la base de données. Vous pouvez en apprendre plus sur les connexions sans mot de passe sur le [Hub des connexions sans mot de passe](#).

Prérequis

- Un [abonnement Azure](#).
- Une base de données SQL configurée avec l'authentification Microsoft Entra ID ([anciennement Azure Active Directory](#)). Vous pouvez en créer une à l'aide du [Guide de démarrage rapide Créez une base de données](#).
- [.NET 7.0](#) ou version ultérieure.
- [Visual Studio](#) ou version ultérieure avec la charge de travail de [Développement web et ASP.NET](#).
- La version la plus récente d'[Azure CLI](#).
- Dernière version des outils Entity Framework Core :
 - Les utilisateurs de Visual Studio doivent installer les [outils de console du Gestionnaire de package pour Entity Framework Core](#).
 - Les utilisateurs de l'interface CLI .NET doivent installer les [outils CLI .NET pour Entity Framework Core](#).

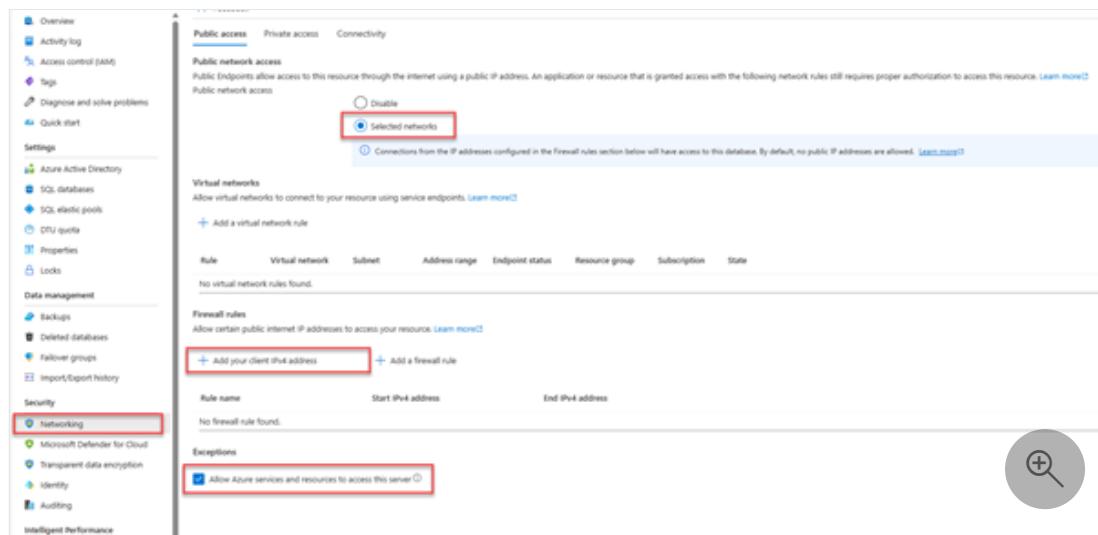
Configurer le serveur de base de données

Les connexions sécurisées et sans mot de passe à Azure SQL Database nécessitent certaines configurations de base de données. Vérifiez les paramètres suivants sur votre [serveur logique dans Azure](#) pour vous connecter correctement à Azure SQL Database dans les environnements locaux et hébergés :

1. Pour les connexions de développement local, vérifiez que votre serveur logique est configuré pour autoriser l'adresse IP de votre ordinateur local et d'autres services

Azure à se connecter :

- Accédez à la page Réseau de votre serveur.
- Activez la case d'option Réseaux sélectionnés pour voir des options de configuration supplémentaires.
- Sélectionnez Ajouter l'adresse IPv4 de votre client (xx.xx.xx.xx.xx.xx) pour ajouter une règle de pare-feu qui activera les connexions à partir de l'adresse IPv4 de votre ordinateur local. Vous pouvez également sélectionner + Ajouter une règle de pare-feu pour entrer une adresse IP spécifique de votre choix.
- Vérifiez que la case Autoriser les services et les ressources Azure à accéder à ce serveur est cochée.



⚠️ Avertissement

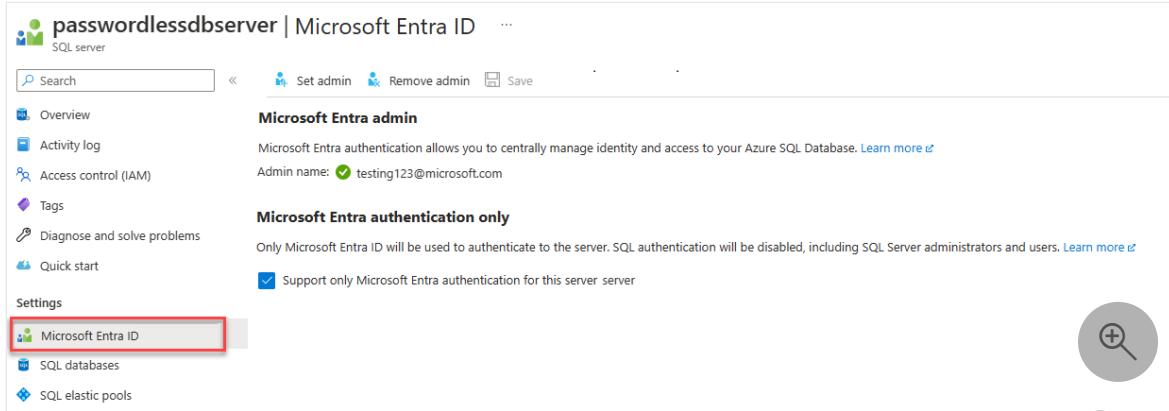
L'activation du paramètre **Autoriser les services et les ressources Azure à accéder à ce serveur** n'est pas une pratique de sécurité recommandée pour les scénarios de production. Les applications réelles doivent implémenter des approches plus sécurisées, telles que des restrictions de pare-feu ou des configurations de réseau virtuel plus strictes.

Vous pouvez en apprendre davantage sur les configurations de sécurité de base de données avec les ressources suivantes :

- Configurer des règles de pare-feu Azure SQL Database.
- Configurer un réseau virtuel avec des points de terminaison privés.

2. Le serveur doit également activer l'authentification Microsoft Entra et disposer d'un compte d'administrateur Microsoft Entra affecté. Pour les connexions de

développement local, le compte d'administrateur de Microsoft Entra doit être un compte que vous pouvez également connecter localement à Visual Studio ou à Azure CLI. Vous pouvez vérifier si l'authentification Microsoft Entra est activée sur la page **Microsoft Entra ID** de votre serveur logique.

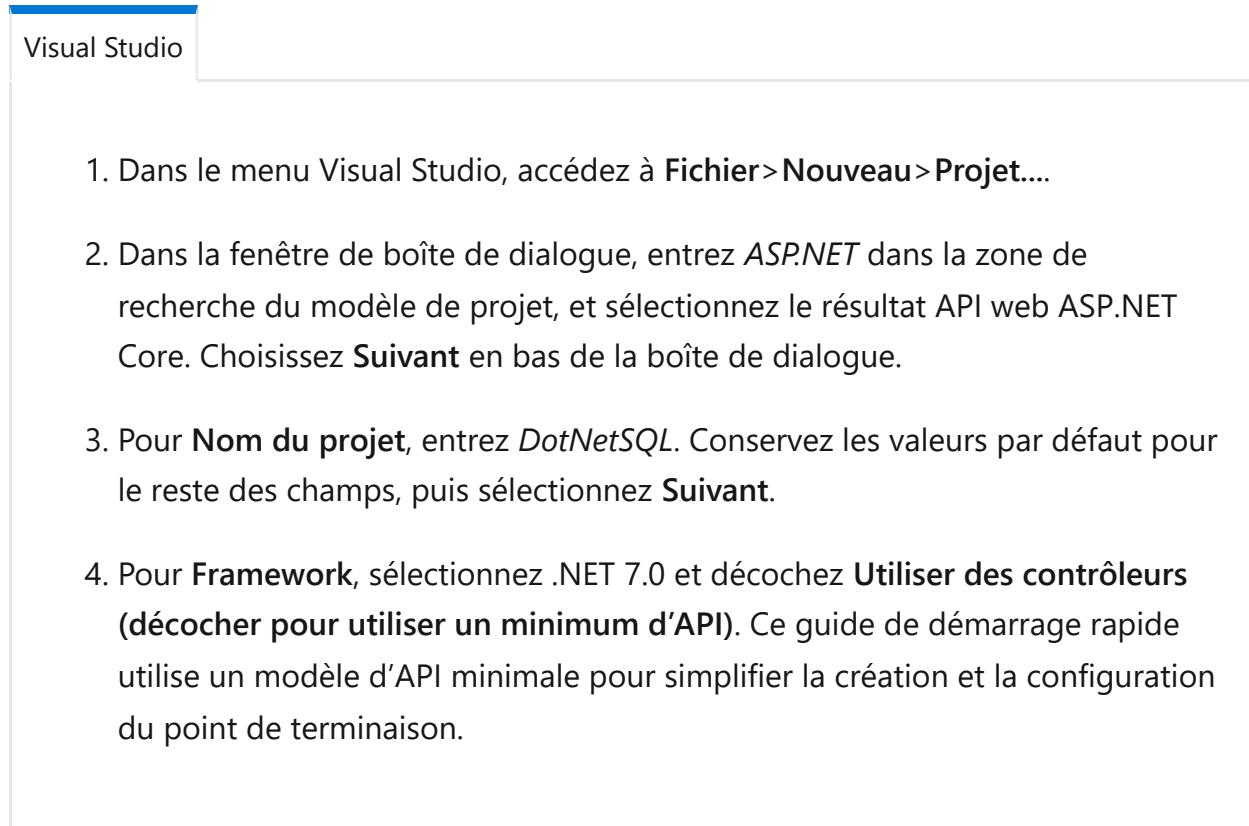


The screenshot shows the Microsoft Entra ID configuration page for a SQL server named 'passwordlessdbserver'. The left sidebar has a 'Settings' section with options like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Quick start, and Microsoft Entra ID (which is selected and highlighted with a red box). The main content area is titled 'Microsoft Entra admin' and describes Microsoft Entra authentication. It shows the 'Admin name' as 'testing123@microsoft.com' and a checked checkbox for 'Support only Microsoft Entra authentication for this server'.

3. Si vous utilisez un compte Azure personnel, assurez-vous d'avoir [Microsoft Entra installé et configuré dans la base de données Azure SQL](#) afin d'assigner votre compte en tant qu'administrateur de serveur. En revanche, si vous utilisez un compte d'entreprise, il est fort probable que Microsoft Entra ID soit déjà configuré pour vous.

Créer le projet

Les étapes à suivre de cette section permettent de créer une API web minimale .NET à l'aide de l'interface CLI .NET ou de Visual Studio 2022.



The screenshot shows the 'Create New Project' dialog in Visual Studio. The 'Visual Studio' tab is selected. Step 1 is shown: 'Dans le menu Visual Studio, accédez à Fichier > Nouveau > Projet....'.

1. Dans le menu Visual Studio, accédez à **Fichier > Nouveau > Projet....**
2. Dans la fenêtre de boîte de dialogue, entrez **ASP.NET** dans la zone de recherche du modèle de projet, et sélectionnez le résultat API web ASP.NET Core. Choisissez **Suivant** en bas de la boîte de dialogue.
3. Pour **Nom du projet**, entrez **DotNetSQL**. Conservez les valeurs par défaut pour le reste des champs, puis sélectionnez **Suivant**.
4. Pour **Framework**, sélectionnez **.NET 7.0** et décochez **Utiliser des contrôleurs (décocher pour utiliser un minimum d'API)**. Ce guide de démarrage rapide utilise un modèle d'API minimale pour simplifier la création et la configuration du point de terminaison.

5. Cliquez sur **Créer**. Le nouveau projet s'ouvre dans l'environnement Visual Studio.

Ajouter Entity Framework au projet

Pour vous connecter à Azure SQL Database à l'aide de .NET et Entity Framework Core, vous devez ajouter trois packages NuGet à votre projet à l'aide de l'une des méthodes suivantes :

Visual Studio

1. Dans la fenêtre **Explorateur de solutions**, cliquez avec le bouton droit sur le nœud **Dépendances** du projet, puis sélectionnez **Gérer les packages NuGet**.
2. Dans la fenêtre résultante, recherchez *EntityFrameworkCore*. Localisez et installez les packages suivants :
 - **Microsoft.EntityFrameworkCore** : fournit les fonctionnalités essentielles d'Entity Framework Core
 - **Microsoft.EntityFrameworkCore.SqlServer** : fournit des composants supplémentaires pour se connecter au serveur logique
 - **Microsoft.EntityFrameworkCore.Design** : prend en charge l'exécution des migrations Entity Framework

Vous pouvez également exécuter la cmdlet `Install-Package` dans la fenêtre **Console du Gestionnaire de package** :

PowerShell

```
Install-Package Microsoft.EntityFrameworkCore
Install-Package Microsoft.EntityFrameworkCore.SqlServer
Install-Package Microsoft.EntityFrameworkCore.Design
```

Ajouter le code pour se connecter à Azure SQL Database

Les bibliothèques Entity Framework Core s'appuient sur les bibliothèques `Microsoft.Data.SqlClient` et `Azure.Identity` pour implémenter des connexions sans

mot de passe à Azure SQL Database. La bibliothèque `Azure.Identity` fournit une classe appelée `DefaultAzureCredential` qui gère l'authentification sans mot de passe auprès d'Azure.

`DefaultAzureCredential` prend en charge plusieurs méthodes d'authentification et détermine laquelle utiliser au moment de l'exécution. Cette approche permet à votre application d'utiliser différentes méthodes d'authentification dans différents environnements (local ou production) sans implémenter de code spécifique à l'environnement. La [Vue d'ensemble de la bibliothèque d'identités Azure](#) explique l'ordre et les emplacements dans lesquels `DefaultAzureCredential` recherche les informations d'identification.

Effectuez les étapes suivantes pour vous connecter à Azure SQL Database à l'aide d'Entity Framework Core et de la classe sous-jacente `DefaultAzureCredential` :

1. Ajoutez une section `ConnectionStrings` au fichier `appsettings.Development.json` afin qu'il corresponde au code suivant. Veillez à mettre à jour les espaces réservés `<your database-server-name>` et `<your-database-name>`.

La chaîne de connexion sans mot de passe inclut une valeur de configuration de `Authentication=Active Directory Default`, qui permet à Entity Framework Core d'utiliser `DefaultAzureCredential` pour se connecter aux services Azure. Lorsque l'application s'exécute localement, elle s'authentifie auprès de l'utilisateur avec lequel vous êtes connecté à Visual Studio. Une fois l'application déployée sur Azure, le même code découvre et applique l'identité managée associée à l'application hébergée, que vous configurerez ultérieurement.

ⓘ Notes

Les chaînes de connexion sans mot de passe peuvent être validées en toute sécurité dans le contrôle de code source, car elles ne contiennent pas de secrets, comme des noms d'utilisateur, des mots de passe ou des clés d'accès.

JSON

```
{  
    "Logging": {  
        "LogLevel": {  
            "Default": "Information",  
            "Microsoft.AspNetCore": "Warning"  
        }  
    },  
    "ConnectionStrings": {  
        "AZURE_SQL_CONNECTIONSTRING": "Data  
        ...  
    }  
}
```

```
Source=passwordlessdbserver.database.windows.net;
    Initial Catalog=passwordlessdb; Authentication=Active
    Directory Default; Encrypt=True;
}
}
```

2. Ajoutez le code suivant au fichier `Program.cs` situé au-dessus de la ligne de code qui lit `var app = builder.Build();`. Ce code effectue les configurations suivantes :

- Récupère la chaîne de connexion de base de données sans mot de passe à partir du fichier `appsettings.Development.json` pour le développement local ou des variables d'environnement pour les scénarios de production hébergés.
- Inscrit la classe Entity Framework Core `DbContext` auprès du conteneur d'injection de dépendances .NET.

C#

```
var connection = String.Empty;
if (builder.Environment.IsDevelopment())
{
    connection =
        builder.Configuration.GetConnectionString("AZURE_SQL_CONNECTIONSTR
        ING");
}
else
{
    connection =
        Environment.GetEnvironmentVariable("AZURE_SQL_CONNECTIONSTRING");
}

builder.Services.AddDbContext<PersonDbContext>(options =>
    options.UseSqlServer(connection));
```

3. Ajoutez les points de terminaison suivants au bas du fichier `Program.cs` ci-dessus `app.Run()` pour récupérer et ajouter des entités dans la base de données à l'aide de la classe `PersonDbContext`.

C#

```
app.MapGet("/Person", (PersonDbContext context) =>
{
    return context.Person.ToList();
})
.WithName("GetPersons")
```

```
.WithOpenApi();

app.MapPost("/Person", (Person person, PersonDbContext context) =>
{
    context.Add(person);
    context.SaveChanges();
})
.WithName("CreatePerson")
.WithOpenApi();
```

Enfin, ajoutez les classes `Person` et `PersonDbContext` en bas du fichier `Program.cs`. La classe `Personne` représente un enregistrement unique dans la table de la base de données `Persons`. La classe `PersonDbContext` représente la base de données `Personne` et vous permet d'effectuer des opérations sur celle-ci via du code. Pour en savoir plus sur `DbContext`, consultez la documentation [Prise en main](#) pour Entity Framework Core.

C#

```
public class Person
{
    public int Id { get; set; }
    public string FirstName { get; set; }
    public string LastName { get; set; }
}

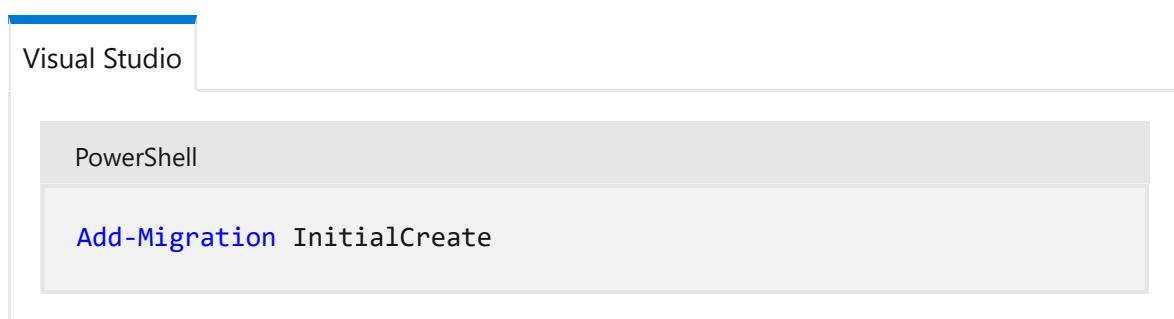
public class PersonDbContext : DbContext
{
    public PersonDbContext(DbContextOptions<PersonDbContext> options)
        : base(options)
    {
    }

    public DbSet<Person> Person { get; set; }
}
```

Exécuter les migrations pour créer la base de données

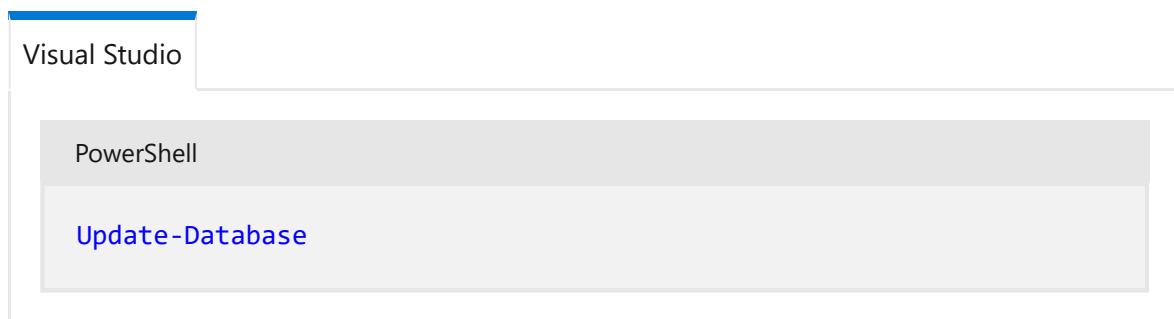
Pour mettre à jour le schéma de base de données pour qu'il corresponde à votre modèle de données à l'aide d'Entity Framework Core, vous devez utiliser une migration. Les migrations peuvent créer et mettre à jour de manière incrémentielle un schéma de base de données pour le maintenir synchronisé avec le modèle de données de votre application. Pour en savoir plus sur ce modèle, consultez la [vue d'ensemble des migrations](#).

1. Ouvrez une fenêtre du terminal vers la racine de votre projet.
2. Exécutez la commande suivante pour générer une migration initiale qui peut créer la base de données :



A screenshot of the Visual Studio interface. On the left, there's a 'Visual Studio' tab. To its right is a 'PowerShell' window. Inside the PowerShell window, the command `Add-Migration InitialCreate` is typed in blue, indicating it's a command being run.

3. Un dossier `Migrations` doit apparaître dans le répertoire de votre projet, ainsi qu'un fichier appelé `InitialCreate` avec des nombres uniques ajoutés. Exécutez la migration pour créer la base de données à l'aide de la commande suivante :



A screenshot of the Visual Studio interface, similar to the previous one. It shows the 'Visual Studio' tab on the left and a 'PowerShell' window on the right. In the PowerShell window, the command `Update-Database` is typed in blue.

Les outils Entity Framework Core créent le schéma de base de données dans Azure défini par la classe `PersonDbContext`.

Tester l'application en local

L'application est prête à être testée localement. Vérifiez que vous êtes connecté à Visual Studio ou à Azure CLI avec le même compte que l'administrateur de votre base de données.

1. Appuyez sur le bouton Exécuter en haut de Visual Studio pour lancer le projet d'API.
2. Dans la page de l'interface utilisateur Swagger, développez la méthode POST, puis sélectionnez **Essayer**.
3. Modifiez l'exemple JSON pour inclure des valeurs pour le prénom et le nom. Sélectionnez **Exécuter** pour ajouter un nouvel enregistrement à la base de données. L'API retourne une réponse de succès.

The screenshot shows the Swagger UI interface for a REST API. At the top, there is a blue header bar with a 'GET' button and a URL field containing '/Person'. Below this, there is a green header bar with a 'POST' button and the same URL. A 'Parameters' section shows 'No parameters'. In the 'Request body' section, which is marked as required, there is a dropdown set to 'application/json'. The JSON content is shown in a code editor-like area:

```
{  
    "firstName": "Bob",  
    "lastName": "Testing"  
}
```

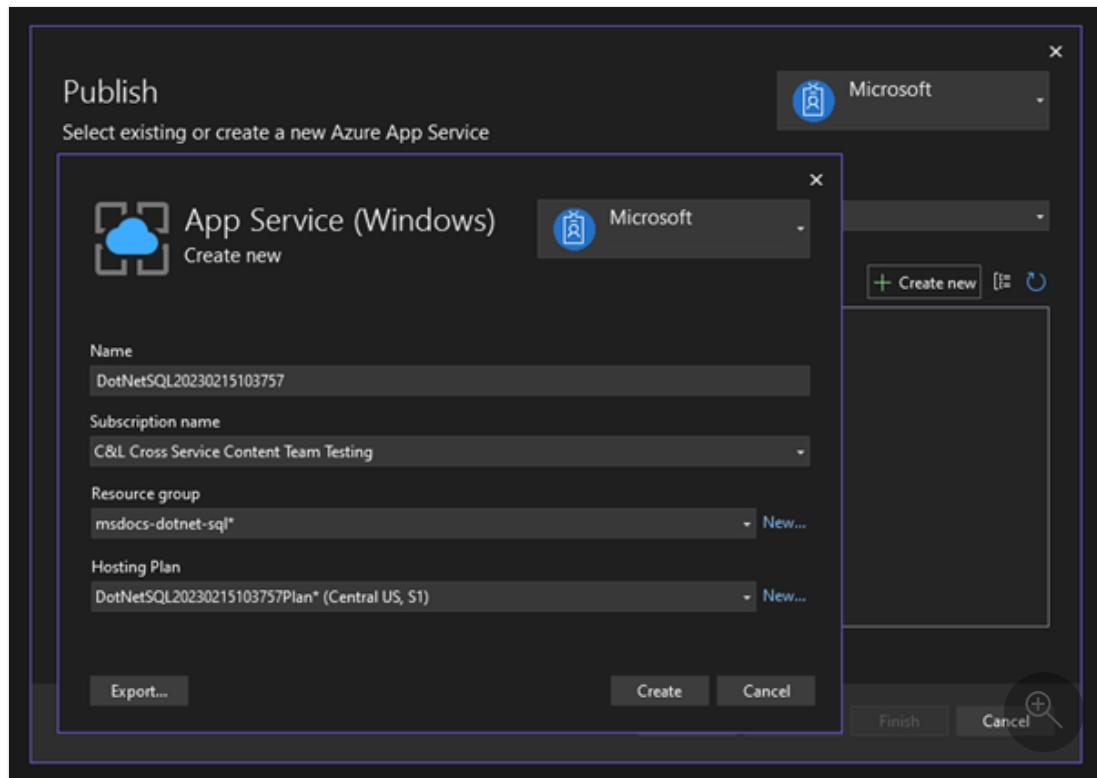
4. Développez la méthode **GET** sur la page de l'interface utilisateur Swagger, puis sélectionnez **Essayer**. Sélectionnez **Exécuter**, et la personne que vous venez de créer est renvoyée.

Déployer dans Azure App Service

L'application est prête à être déployée sur Azure. Visual Studio peut créer une instance Azure App Service et déployer votre application au cours d'un même workflow.

1. Vérifiez que l'application est arrêtée et générée correctement.
2. Dans la fenêtre **Explorateur de solutions** de Visual Studio, cliquez avec le bouton droit sur le nœud de projet de niveau supérieur, puis sélectionnez **Publier**.
3. Dans la boîte de dialogue de publication, sélectionnez **Azure** comme cible de déploiement, puis sélectionnez **Suivant**.
4. Pour la cible spécifique, sélectionnez **Azure App Service (Windows)**, puis **Suivant**.
5. Sélectionnez l'icône verte + pour créer une instance App Service sur laquelle effectuer le déploiement, et entrez les valeurs suivantes :
 - **Nom** : conservez la valeur par défaut.
 - **Nom de l'abonnement** : sélectionnez l'abonnement à déployer.

- **Groupe de ressources** : sélectionnez **Nouveau** et créez un nouveau groupe de ressources appelé *msdocs-dotnet-sql*.
- **Plan d'hébergement** : sélectionnez **Nouveau** pour ouvrir la boîte de dialogue Plan d'hébergement. Laissez les valeurs par défaut et sélectionnez **OK**.
- Sélectionnez **Créer** pour fermer la boîte de dialogue d'origine. Visual Studio crée la ressource App Service dans Azure.



6. Une fois la ressource créée, vérifiez qu'elle est sélectionnée dans la liste des services d'application, puis sélectionnez **Suivant**.
7. Dans l'étape **Gestion des API**, cochez la case **Ignorer cette étape** en bas, puis sélectionnez **Terminer**.
8. Sélectionnez **Publier** en haut à droite du résumé du profil de publication pour déployer l'application sur Azure.

Une fois le déploiement terminé, Visual Studio lance le navigateur pour afficher l'application hébergée, mais à ce stade, l'application ne fonctionne pas correctement sur Azure. Vous devez toujours configurer la connexion sécurisée entre App Service et la base de données SQL pour récupérer vos données.

Connecter App Service à Azure SQL Database

Les étapes suivantes sont requises pour connecter l'instance App Service à Azure SQL Database :

1. Créez une identité managée pour App Service. La bibliothèque `Microsoft.Data.SqlClient` incluse dans votre application découvre automatiquement l'identité managée, tout comme elle a découvert votre utilisateur Visual Studio local.
2. Créez un utilisateur de base de données SQL et associez-le à l'identité managée App Service.
3. Attribuez des rôles SQL à l'utilisateur de base de données qui octroient des autorisations de lecture, d'écriture et éventuellement d'autres autorisations.

Plusieurs outils sont disponibles pour implémenter ces étapes :

Service Connector (recommandé)

Service Connector est un outil qui simplifie les connexions authentifiées entre différents services dans Azure. Service Connector prend actuellement en charge la connexion d'App Service à une base de données SQL via Azure CLI à l'aide de la commande `az webapp connection create sql`. Cette commande unique effectue les trois étapes mentionnées ci-dessus pour vous.

Azure CLI

```
az webapp connection create sql
-g <your-resource-group>
-n <your-app-service-name>
--tg <your-database-server-resource-group>
--server <your-database-server-name>
--database <your-database-name>
--system-identity
```

Vous pouvez vérifier les modifications apportées par Service Connector sur les paramètres App Service.

1. Accédez à la page **Identité** de votre App Service. Sous l'onglet **Affecté par le système**, l'**État** doit être défini sur **Activé**. Cette valeur signifie qu'une identité managée affectée par le système a été activée pour votre application.
2. Accédez à la page **Configuration** de votre App Service. Sous l'onglet **Chaînes de connexion**, vous devriez voir une chaîne de connexion appelée **AZURE_SQL_CONNECTIONSTRING**. Sélectionnez le texte **Cliquer pour afficher la valeur** pour afficher la chaîne de connexion sans mot de passe générée. Le nom de cette chaîne de connexion s'aligne sur celui que vous avez

configuré dans votre application, de sorte qu'il est détecté automatiquement lors de l'exécution dans Azure.

ⓘ Important

Bien que cette solution offre une approche simple pour la prise en main, il ne s'agit pas d'une bonne pratique pour les environnements de production d'entreprise.

Dans ces scénarios, l'application ne doit pas effectuer toutes les opérations à l'aide d'une seule identité élevée. Vous devez essayer d'implémenter le principe de privilège minimum en configurant plusieurs identités avec des autorisations spécifiques pour des tâches spécifiques.

Vous pouvez en apprendre plus sur la configuration des rôles de base de données et la sécurité avec les ressources suivantes :

[Tutoriel : Sécuriser une base de données dans Azure SQL Database](#)

[Autoriser l'accès base de données à SQL Database](#)

Tester l'application déployée

Accédez à l'URL de l'application pour tester que la connexion à Azure SQL Database fonctionne. Vous pouvez localiser l'URL de votre application dans la page de vue d'ensemble d'App Service. Ajoutez le chemin d'accès `/person` à la fin de l'URL pour accéder au même point de terminaison que celui que vous avez testé localement.

La personne que vous avez créée localement devrait s'afficher dans le navigateur. Félicitations ! Votre application est maintenant connectée à Azure SQL Database dans les environnements locaux et hébergés.

Nettoyer les ressources

Une fois que vous avez terminé d'utiliser Azure SQL Database, supprimez la ressource pour éviter les coûts imprévus.

Azure portal

1. Dans la barre de recherche du portail Azure, recherchez *Azure SQL* et sélectionnez le résultat correspondant.

2. Recherchez et sélectionnez votre base de données dans la liste.
3. Dans la page **Vue d'ensemble** d'Azure SQL Database, sélectionnez **Supprimer**.
4. Dans la page **Voulez-vous vraiment supprimer...** qui s'ouvre, tapez le nom de la base de données pour confirmer, puis sélectionnez **Supprimer**.

Étapes suivantes

- Tutoriel : Sécuriser une base de données dans Azure SQL Database
- Autoriser l'accès base de données à SQL Database
- Une vue d'ensemble des fonctionnalités de sécurité d'Azure SQL Database
- Meilleures pratiques en matière de sécurité pour Azure SQL Database

Se connecter à la base de données et interroger Azure SQL à l'aide de .NET et de la bibliothèque Microsoft.Data.SqlClient

Article • 11/07/2023

S'applique à  [Azure SQL Database](#)

Ce guide de démarrage rapide explique comment connecter une application à une base de données dans Azure SQL Database et effectuer des requêtes à l'aide de .NET et de la bibliothèque [Microsoft.Data.SqlClient](#). Ce guide de démarrage rapide suit l'approche sans mot de passe recommandée pour se connecter à la base de données. Vous pouvez en apprendre plus sur les connexions sans mot de passe sur le [Hub des connexions sans mot de passe](#).

Prérequis

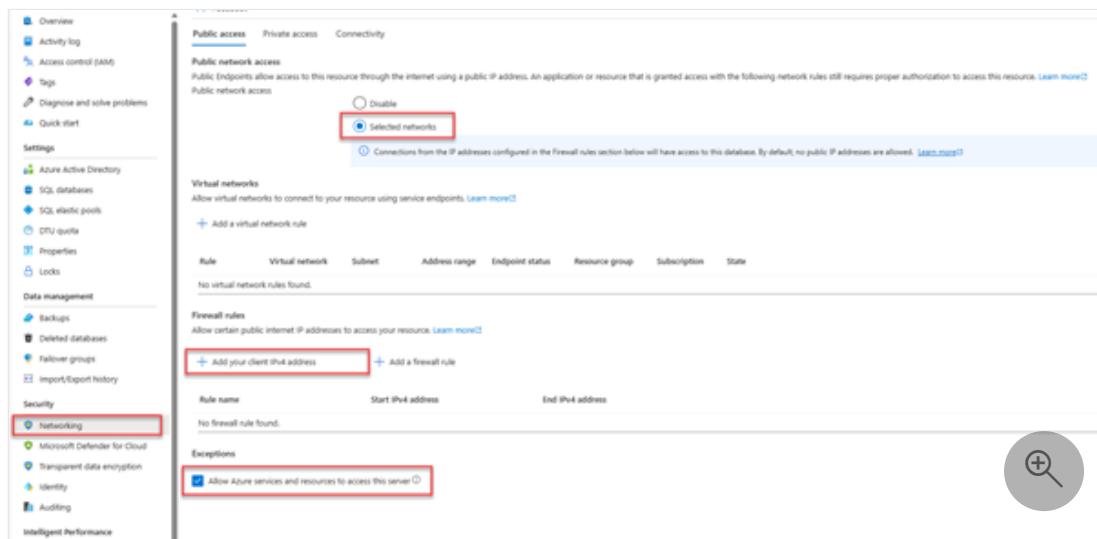
- Un [abonnement Azure](#).
- Une base de données Azure SQL configurée avec l'authentification Microsoft Entra ID ([anciennement Azure Active Directory](#)). Vous pouvez en créer une à l'aide du [Guide de démarrage rapide Créez une base de données](#).
- La version la plus récente d'[Azure CLI](#).
- [Visual Studio](#) ou version ultérieure avec la charge de travail de [Développement web et ASP.NET](#).
- [.NET 7.0](#) ou version ultérieure.

Configurer la base de données

Les connexions sécurisées et sans mot de passe à Azure SQL Database nécessitent certaines configurations de base de données. Vérifiez les paramètres suivants sur votre [serveur logique dans Azure](#) pour vous connecter correctement à Azure SQL Database dans les environnements locaux et hébergés :

1. Pour les connexions de développement local, vérifiez que votre serveur logique est configuré pour autoriser l'adresse IP de votre ordinateur local et d'autres services Azure à se connecter :
 - Accédez à la page [Réseau](#) de votre serveur.

- Activez la case d'option **Réseaux sélectionnés** pour voir des options de configuration supplémentaires.
- Sélectionnez **Ajouter l'adresse IPv4 de votre client (xx.xx.xx.xx.xx.xx)** pour ajouter une règle de pare-feu qui activera les connexions à partir de l'adresse IPv4 de votre ordinateur local. Vous pouvez également sélectionner **+ Ajouter une règle de pare-feu** pour entrer une adresse IP spécifique de votre choix.
- Vérifiez que la case **Autoriser les services et les ressources Azure à accéder à ce serveur** est cochée.



⚠ Avertissement

L'activation du paramètre **Autoriser les services et les ressources Azure à accéder à ce serveur** n'est pas une pratique de sécurité recommandée pour les scénarios de production. Les applications réelles doivent implémenter des approches plus sécurisées, telles que des restrictions de pare-feu ou des configurations de réseau virtuel plus strictes.

Vous pouvez en apprendre davantage sur les configurations de sécurité de base de données avec les ressources suivantes :

- **Configurer des règles de pare-feu Azure SQL Database.**
- **Configurer un réseau virtuel avec des points de terminaison privés.**

2. Le serveur doit également activer l'authentification Microsoft Entra et disposer d'un compte d'administrateur Microsoft Entra affecté. Pour les connexions de développement local, le compte d'administrateur de Microsoft Entra doit être un compte que vous pouvez également connecter localement à Visual Studio ou à

Azure CLI. Vous pouvez vérifier si l'authentification Microsoft Entra est activée sur la page **Microsoft Entra ID** de votre serveur logique.

The screenshot shows the Azure portal interface for managing a SQL server named 'passwordlessdbserver'. In the left sidebar, under 'Settings', the 'Microsoft Entra ID' option is highlighted with a red box. The main content area is titled 'Microsoft Entra admin' and contains the following information:

- Microsoft Entra authentication only**: A note stating that only Microsoft Entra ID will be used for authentication, and SQL authentication will be disabled.
- Admin name:** testing123@microsoft.com
- Support only Microsoft Entra authentication for this server**: A checked checkbox.

3. Si vous utilisez un compte Azure personnel, assurez-vous d'avoir [Microsoft Entra installé et configuré dans la base de données Azure SQL](#) afin d'assigner votre compte en tant qu'administrateur de serveur. En revanche, si vous utilisez un compte d'entreprise, il est fort probable que Microsoft Entra ID soit déjà configuré pour vous.

Créer le projet

Pour les étapes à suivre, créez une API web minimale .NET à l'aide de l'interface CLI .NET ou de Visual Studio 2022.

The screenshot shows the 'Visual Studio' tab selected in the 'New Project' dialog. The 'ASP.NET' template is selected in the search bar, and the 'API (Model-View-Controller)' result is highlighted. The 'Next >' button is visible at the bottom of the dialog.

1. Dans le menu de Visual Studio, accédez à **Fichier>Nouveau>Projet....**
2. Dans la fenêtre de boîte de dialogue, entrez **ASP.NET** dans la zone de recherche du modèle de projet, et sélectionnez le résultat API web ASP.NET Core. Choisissez **Suivant** en bas de la boîte de dialogue.
3. Pour **Nom du projet**, entrez *DotNetSQL*. Conservez les valeurs par défaut pour le reste des champs, puis sélectionnez **Suivant**.
4. Pour **Framework**, sélectionnez .NET 7.0 et décochez **Utiliser des contrôleurs (décocher pour utiliser un minimum d'API)**. Ce guide de démarrage rapide utilise un modèle d'API minimale pour simplifier la création et la configuration du point de terminaison.

5. Cliquez sur **Créer**. Le nouveau projet s'ouvre dans l'environnement Visual Studio.

Ajouter la bibliothèque Microsoft.Data.SqlClient

Pour vous connecter à Azure SQL Database à l'aide de .NET, installez `Microsoft.Data.SqlClient`. Ce package joue le rôle de fournisseur de données pour la connexion aux bases de données, l'exécution de commandes et la récupération des résultats.

ⓘ Notes

Veillez à installer `Microsoft.Data.SqlClient` et non `System.Data.SqlClient`.

`Microsoft.Data.SqlClient` est une version plus récente de la bibliothèque cliente SQL qui fournit des fonctionnalités supplémentaires.

Visual Studio

1. Dans la fenêtre **Explorateur de solutions**, cliquez avec le bouton droit sur le nœud **Dépendances** du projet, puis sélectionnez **Gérer les packages NuGet**.
2. Dans la fenêtre résultante, recherchez `SqlClient`. Recherchez le résultat `Microsoft.Data.SqlClient` et sélectionnez **Installer**.

Configurer la chaîne de connexion

Sans mot de passe (recommandé)

Pour un développement local avec des connexions sans mot de passe à Azure SQL Database, ajoutez la section `ConnectionStrings` ci-dessous au fichier `appsettings.json`. Remplacez les espaces réservés `<database-server-name>` et `<database-name>` par vos valeurs.

JSON

```
".ConnectionStrings": {  
    "AZURE_SQL_CONNECTIONSTRING": "Server=tcp:<database-server-name>.database.windows.net,1433;Initial Catalog=<database-name>;Encrypt=True;TrustServerCertificate=False;Connection Timeout=30;Authentication=\"Active Directory Default\";"  
}
```

La chaîne de connexion sans mot de passe définit la valeur de configuration `Authentication="Active Directory Default"`, qui donne instruction à la bibliothèque `Microsoft.Data.SqlClient` de se connecter à Azure SQL Database en utilisant une classe appelée `DefaultAzureCredential`. `DefaultAzureCredential` permet les connexions sans mot de passe aux services Azure et est fourni par la bibliothèque Azure Identity dont dépend la bibliothèque de client SQL. `DefaultAzureCredential` prend en charge plusieurs méthodes d'authentification et détermine celle qui est utilisée au moment de l'exécution pour différents environnements.

Par exemple, quand l'application s'exécute localement, `DefaultAzureCredential` s'authentifie via l'utilisateur avec lequel vous êtes connecté dans Visual Studio ou d'autres outils locaux comme Azure CLI. Une fois l'application déployée sur Azure, le même code découvre et applique l'identité managée associée à l'application hébergée, que vous configurerez ultérieurement. La [Vue d'ensemble de la bibliothèque d'identités Azure](#) explique l'ordre et les emplacements dans lesquels `DefaultAzureCredential` recherche les informations d'identification.

ⓘ Notes

Les chaînes de connexion sans mot de passe peuvent être validées sans risque dans le contrôle de code source, car elles ne contiennent pas de secrets comme des noms d'utilisateur, des mots de passe ou des clés d'accès.

Ajouter le code pour se connecter à Azure SQL Database

Remplacez le contenu du fichier `Program.cs` par le code ci-dessous, qui effectue les étapes importantes suivantes :

- Récupère la chaîne de connexion sans mot de passe dans `appsettings.json`

- Crée une table `Persons` dans la base de données au démarrage (pour les scénarios de test uniquement)
- Crée un point de terminaison HTTP GET pour récupérer tous les enregistrements stockés dans la table `Persons`
- Crée un point de terminaison HTTP POST pour ajouter de nouveaux enregistrements dans la table `Persons`

C#

```
using Microsoft.Data.SqlClient;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();

var app = builder.Build();

// For production scenarios, consider keeping Swagger configurations behind
// the environment check
// if (app.Environment.IsDevelopment())
// {
//     app.UseSwagger();
//     app.UseSwaggerUI();
// }

app.UseHttpsRedirection();

string connectionString =
app.Configuration.GetConnectionString("AZURE_SQL_CONNECTIONSTRING")!;

try
{
    // Table would be created ahead of time in production
    using var conn = new SqlConnection(connectionString);
    conn.Open();

    var command = new SqlCommand(
        "CREATE TABLE Persons (ID int NOT NULL PRIMARY KEY IDENTITY,
FirstName varchar(255), LastName varchar(255));",
        conn);
    using SqlDataReader reader = command.ExecuteReader();
}

catch (Exception e)
{
    // Table may already exist
    Console.WriteLine(e.Message);
}

app.MapGet("/Person", () => {
    var rows = new List<string>();
```

```

        using var conn = new SqlConnection(connectionString);
        conn.Open();

        var command = new SqlCommand("SELECT * FROM Persons", conn);
        using SqlDataReader reader = command.ExecuteReader();

        if (reader.HasRows)
        {
            while (reader.Read())
            {
                rows.Add($"{reader.GetInt32(0)}, {reader.GetString(1)},
{reader.GetString(2)}");
            }
        }

        return rows;
    })
    .WithName("GetPersons")
    .WithOpenApi();

app.MapPost("/Person", (Person person) => {
    using var conn = new SqlConnection(connectionString);
    conn.Open();

    var command = new SqlCommand(
        "INSERT INTO Persons (firstName, lastName) VALUES (@firstName,
@lastName)",
        conn);

    command.Parameters.Clear();
    command.Parameters.AddWithValue("@firstName", person.FirstName);
    command.Parameters.AddWithValue("@lastName", person.LastName);

    using SqlDataReader reader = command.ExecuteReader();
})
    .WithName("CreatePerson")
    .WithOpenApi();

app.Run();

```

Enfin, ajoutez la classe `Person` au bas du fichier `Program.cs`. Cette classe représente un enregistrement unique dans la table de la base de données `Persons`.

C#

```

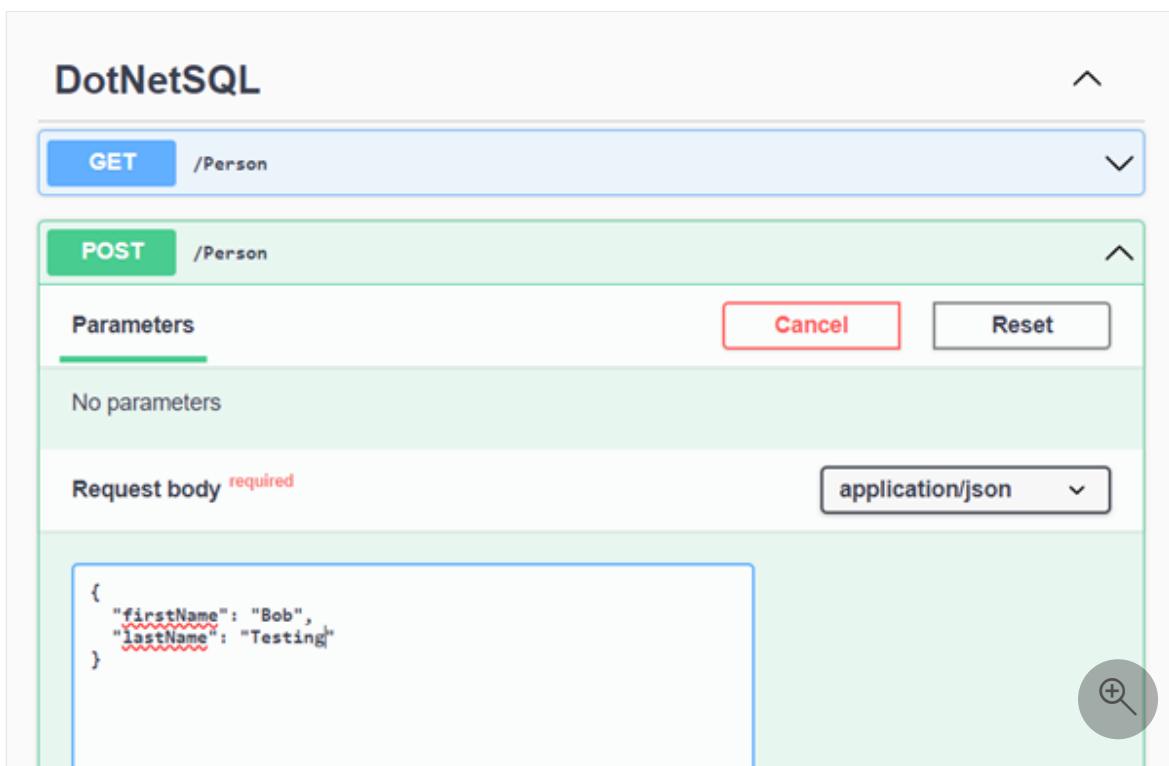
public class Person
{
    public required string FirstName { get; set; }
    public required string LastName { get; set; }
}

```

Exécuter et tester l'application localement

L'application est prête à être testée localement. Vérifiez que vous êtes connecté à Visual Studio ou à Azure CLI avec le même compte que l'administrateur de votre base de données.

1. Appuyez sur le bouton Exécuter en haut de Visual Studio pour lancer le projet d'API.
2. Dans la page de l'interface utilisateur Swagger, développez la méthode POST, puis sélectionnez **Essayer**.
3. Modifiez l'exemple JSON pour inclure des valeurs pour le prénom et le nom. Sélectionnez **Exécuter** pour ajouter un nouvel enregistrement à la base de données. L'API retourne une réponse de succès.



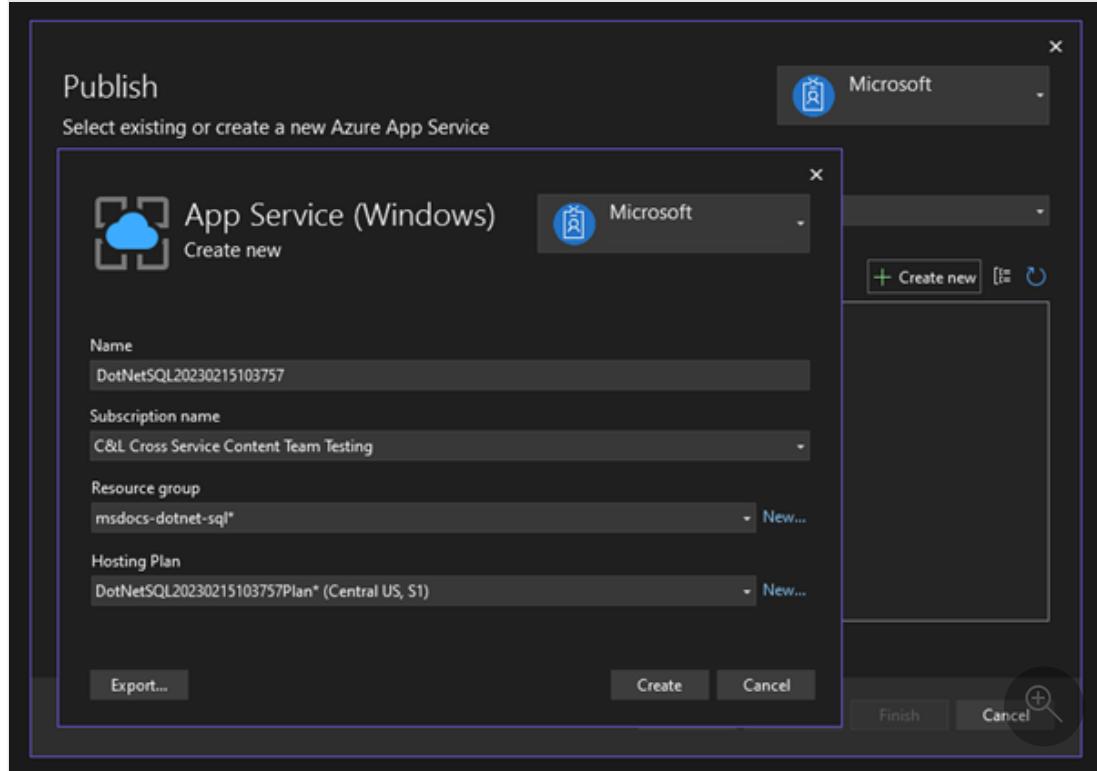
4. Développez la méthode **GET** sur la page de l'interface utilisateur Swagger, puis sélectionnez **Essayer**. Choisissez **Exécuter**, et la personne que vous venez de créer est retournée.

Déployer dans Azure App Service

L'application est prête à être déployée sur Azure. Visual Studio peut créer une instance Azure App Service et déployer votre application au cours d'un même workflow.

1. Vérifiez que l'application est arrêtée et générée correctement.

2. Dans la fenêtre **Explorateur de solutions** de Visual Studio, cliquez avec le bouton droit sur le nœud de projet de niveau supérieur, puis sélectionnez **Publier**.
3. Dans la boîte de dialogue de publication, sélectionnez **Azure** comme cible de déploiement, puis sélectionnez **Suivant**.
4. Pour la cible spécifique, sélectionnez **Azure App Service (Windows)**, puis **Suivant**.
5. Sélectionnez l'icône + pour créer une instance App Service sur laquelle effectuer le déploiement, puis entrez les valeurs suivantes :
 - **Nom** : conservez la valeur par défaut.
 - **Nom de l'abonnement** : sélectionnez l'abonnement à déployer.
 - **Groupe de ressources** : sélectionnez **Nouveau** et créez un nouveau groupe de ressources appelé *msdocs-dotnet-sql*.
 - **Plan d'hébergement** : sélectionnez **Nouveau** pour ouvrir la boîte de dialogue Plan d'hébergement. Laissez les valeurs par défaut et sélectionnez **OK**.
 - Sélectionnez **Créer** pour fermer la boîte de dialogue d'origine. Visual Studio crée la ressource App Service dans Azure.



6. Une fois la ressource créée, vérifiez qu'elle est sélectionnée dans la liste des services d'application, puis sélectionnez **Suivant**.

7. Dans l'étape **Gestion des API**, cochez la case **Ignorer cette étape** en bas, puis choisissez **Terminer**.
8. À l'étape **Terminer**, sélectionnez **Fermer** si la boîte de dialogue ne se ferme pas automatiquement.
9. Sélectionnez **Publier** en haut à droite du résumé du profil de publication pour déployer l'application sur Azure.

Une fois le déploiement terminé, Visual Studio lance le navigateur pour afficher l'application hébergée, mais à ce stade, l'application ne fonctionne pas correctement sur Azure. Vous devez toujours configurer la connexion sécurisée entre App Service et la base de données SQL pour récupérer vos données.

Connecter App Service à Azure SQL Database

Sans mot de passe (recommandé)

Les étapes suivantes sont nécessaires pour créer une connexion sans mot de passe entre l'instance App Service et Azure SQL Database :

1. Créez une identité managée pour App Service. La bibliothèque `Microsoft.Data.SqlClient` incluse dans votre application découvre automatiquement l'identité managée, tout comme elle a découvert votre utilisateur Visual Studio local.
2. Créez un utilisateur de base de données SQL et associez-le à l'identité managée App Service.
3. Attribuez des rôles SQL à l'utilisateur de base de données qui octroient des autorisations de lecture, d'écriture et éventuellement d'autres autorisations.

Plusieurs outils sont disponibles pour implémenter ces étapes :

Service Connector (recommandé)

Service Connector est un outil qui simplifie les connexions authentifiées entre différents services dans Azure. Service Connector prend actuellement en charge la connexion d'App Service à une base de données SQL via Azure CLI à l'aide de la commande `az webapp connection create sql`. Cette commande unique effectue les trois étapes mentionnées ci-dessus pour vous.

Azure CLI

```
az webapp connection create sql \
    -g <app-service-resource-group> \
    -n <app-service-name> \
    --tg <database-server-resource-group> \
    --server <database-server-name> \
    --database <database-name> \
    --system-identity
```

Vous pouvez vérifier les modifications apportées par Service Connector sur les paramètres App Service.

1. Accédez à la page **Identité** de votre App Service. Sous l'onglet **Affecté par le système**, l'**État** doit être défini sur **Activé**. Cette valeur signifie qu'une identité managée affectée par le système a été activée pour votre application.
2. Accédez à la page **Configuration** de votre App Service. Sous l'onglet **Chaînes de connexion**, vous devriez voir une chaîne de connexion appelée **AZURE_SQL_CONNECTIONSTRING**. Sélectionnez le texte **Cliquer pour afficher la valeur** pour afficher la chaîne de connexion sans mot de passe générée. Le nom de cette chaîne de connexion correspondant à celui que vous avez configuré dans votre application, il sera détecté automatiquement pendant l'exécution dans Azure.

ⓘ Important

Bien que cette solution offre une approche simple pour se lancer, il ne s'agit pas d'une bonne pratique pour les environnements de niveau production. Dans ces scénarios, l'application ne doit pas effectuer toutes les opérations en utilisant une même identité avec privilèges élevés. Vous devez essayer d'implémenter le principe de privilège minimum en configurant plusieurs identités avec des autorisations spécifiques pour des tâches spécifiques.

Vous pouvez en apprendre plus sur la configuration des rôles de base de données et la sécurité avec les ressources suivantes :

- [Tutoriel : Sécuriser une base de données dans Azure SQL Database](#)
- [Autoriser l'accès base de données à SQL Database](#)

Tester l'application déployée

1. Sélectionnez le bouton **Parcourir** en haut de la page de présentation d'App Service pour lancer l'URL racine de votre application.
2. Ajoutez le chemin `/swagger/index.html` de l'URL pour charger la même page de test Swagger que celle utilisée localement.
3. Exécutez des requêtes de test GET et POST pour vérifier que les points de terminaison fonctionnent comme prévu.

💡 Conseil

Si vous obtenez une erreur Serveur interne 500 pendant le test, ce sont peut-être vos configurations réseau de base de données qui sont en cause. Vérifiez que votre serveur logique est configuré avec les paramètres décrits dans la section [Configurer la base de données](#).

Félicitations ! Votre application est maintenant connectée à Azure SQL Database dans les environnements locaux et hébergés.

Nettoyer les ressources

Une fois que vous avez terminé d'utiliser Azure SQL Database, supprimez la ressource pour éviter les coûts imprévus.

Azure portal

1. Dans la barre de recherche du portail Azure, recherchez *Azure SQL* et sélectionnez le résultat correspondant.
2. Recherchez et sélectionnez votre base de données dans la liste.
3. Dans la page **Vue d'ensemble** d'Azure SQL Database, sélectionnez **Supprimer**.
4. Dans la page **Voulez-vous vraiment supprimer...** qui s'ouvre, tapez le nom de la base de données pour confirmer, puis sélectionnez **Supprimer**.

Démarrage rapide : Bibliothèque Azure Cosmos DB pour NoSQL pour .NET

Article • 11/01/2024

S'APPLIQUE À : NoSQL

Prise en main de la bibliothèque cliente Azure Cosmos DB pour NoSQL pour .NET afin d'interroger des données dans vos conteneurs et d'effectuer des opérations courantes sur des éléments individuels. Suivez ces étapes pour déployer une solution minimale dans votre environnement à l'aide de l'interface Azure Developer CLI.

[Documentation de référence sur l'API](#) | [Code source de la bibliothèque](#) | [Package \(NuGet\)](#) | [Azure Developer CLI](#)

Prérequis

- Compte Azure avec un abonnement actif. [Créez un compte gratuitement](#).
- [Compte GitHub](#)

Configuration

Déployez le conteneur de développement de ce projet dans votre environnement. Utilisez ensuite Azure Developer CLI (`azd`) pour créer un compte Azure Cosmos DB pour NoSQL et déployer un exemple d'application conteneurisé. L'exemple d'application utilise la bibliothèque cliente pour gérer, créer, lire et interroger des exemples de données.



Important

Les comptes GitHub incluent un droit de stockage et des heures de base sans coût. Pour plus d'informations, consultez [heures de stockage et de cœur incluses pour les comptes GitHub](#).

1. Ouvrez un terminal dans le répertoire racine du projet.
2. S'authentifier auprès d'Azure Developer CLI à l'aide de `azd auth login`. Suivez les étapes spécifiées par l'outil pour vous authentifier auprès de l'interface CLI à l'aide

de vos informations d'identification Azure préférées.

```
Azure CLI
```

```
azd auth login
```

3. Utilisez `azd init` pour initialiser le projet.

```
Azure CLI
```

```
azd init
```

4. Lors de l'initialisation, configurez un nom d'environnement unique.

💡 Conseil

Le nom de l'environnement sera également utilisé comme nom du groupe de ressources cible. Pour ce guide de démarrage rapide, envisagez d'utiliser `msdocs-cosmos-db-nosql`.

5. Déployez le compte Azure Cosmos DB pour NoSQL à l'aide de `azd up`. Les modèles Bicep déplient également un exemple d'application web.

```
Azure CLI
```

```
azd up
```

6. Pendant le processus d'approvisionnement, sélectionnez votre abonnement et l'emplacement souhaité. Attendez la fin du processus de provisionnement. Le processus peut prendre **environ cinq minutes**.

7. Une fois l'approvisionnement de vos ressources Azure terminée, une URL vers l'application web en cours d'exécution est incluse dans la sortie.

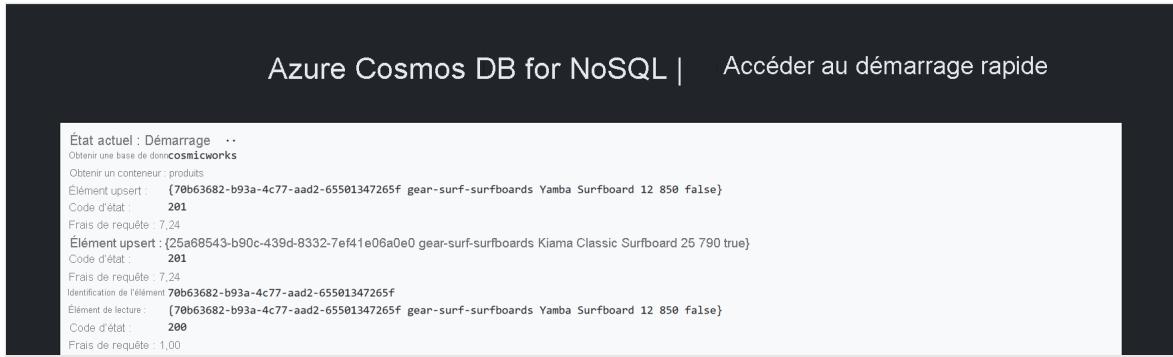
```
Output
```

```
Deploying services (azd deploy)
```

```
(✓) Done: Deploying service web
- Endpoint: <https://[container-app-sub-domain].azurecontainerapps.io>
```

```
SUCCESS: Your application was provisioned and deployed to Azure in 5
minutes 0 seconds.
```

8. Utilisez l'URL dans la console pour accéder à votre application web dans le navigateur. Observez la sortie de l'application en cours d'exécution.



Azure Cosmos DB for NoSQL | Accéder au démarrage rapide

```
État actuel : Démarrage ..  
Obtenir une base de données  
Obtenir un conteneur : produits  
Élément upsert : {70b63682-b93a-4c77-aad2-65501347265f gear-surf-surfboards Yamba Surfboard 12 850 false}  
Code d'état : 201  
Frais de requête : 7,24  
Élément upsert : {25a69543-b90c-439d-8332-7ef41e06a0e0 gear-surf-surfboards Kiama Classic Surfboard 25 790 true}  
Code d'état : 201  
Frais de requête : 7,24  
Identification de l'élément 70b63682-b93a-4c77-aad2-65501347265f  
Élément de lecture : {70b63682-b93a-4c77-aad2-65501347265f gear-surf-surfboards Yamba Surfboard 12 850 false}  
Code d'état : 200  
Frais de requête : 1,00
```

Installer la bibliothèque de client

La bibliothèque cliente est disponible via NuGet, en tant que package

`Microsoft.Azure.Cosmos`.

1. Ouvrez un terminal et accédez au dossier `/src/web`.

Bash

```
cd ./src/web
```

2. S'il n'est pas déjà installé, installez le package `Microsoft.Azure.Cosmos` à l'aide de `dotnet add package`.

Bash

```
dotnet add package Microsoft.Azure.Cosmos
```

3. Installez également le package `Azure.Identity` s'il n'est pas déjà installé.

Bash

```
dotnet add package Azure.Identity
```

4. Ouvrez et examinez le fichier `src/web/Cosmos.Samples.NoSQL.Quickstart.Web.csproj` pour vérifier que les entrées `Microsoft.Azure.Cosmos` et `Azure.Identity` existent.

Modèle objet

Nom	Description
CosmosClient	Cette classe est la classe cliente principale et est utilisée pour gérer les métadonnées ou bases de données à l'échelle du compte.
Database	Cette classe représente une base de données dans le compte.
Container	Cette classe est principalement utilisée pour effectuer des opérations de lecture, de mise à jour et de suppression sur le conteneur ou les éléments stockés dans le conteneur.
PartitionKey	Cette classe représente une clé de partition logique. Cette classe est requise pour de nombreuses opérations et requêtes courantes.

Exemples de code

- [Authentifier le client](#)
- [Obtenir une base de données](#)
- [Obtenir un conteneur](#)
- [Créer un élément](#)
- [Obtenir un élément](#)
- [Éléments de requête](#)

L'exemple de code dans le modèle utilise une base de données nommée `cosmicworks` et un conteneur nommé `products`. Le conteneur `products` contient des détails tels que le nom, la catégorie, la quantité, un identificateur unique et un indicateur de vente pour chaque produit. Le conteneur utilise la propriété `/category` comme clé de partition logique.

Authentifier le client

Les requêtes d'application vers les Services Azure doivent être autorisées. Utilisez le type `DefaultAzureCredential` comme moyen préféré d'implémenter une connexion sans mot de passe entre vos applications et Azure Cosmos DB for NoSQL. `DefaultAzureCredential` prend en charge plusieurs méthodes d'authentification et détermine quelle méthode doit être utilisée au moment de l'exécution.

ⓘ Important

Vous pouvez également autoriser directement les requêtes adressées aux services Azure à l'aide de mots de passe, de chaînes de connexion ou d'autres

informations d'identification. Toutefois, cette approche doit être utilisée avec prudence. Les développeurs doivent être vigilants pour ne jamais exposer les secrets dans un emplacement non sécurisé. Toute personne ayant accès au mot de passe ou à la clé secrète peut s'authentifier auprès du service de base de données. `DefaultAzureCredential` offre des avantages de gestion et de sécurité améliorés sur la clé de compte pour autoriser l'authentification sans mot de passe sans risque de stocker des clés.

Cet exemple crée une instance de la classe `CosmosClient` et s'authentifie à l'aide d'une instance `DefaultAzureCredential`.

C#

```
CosmosClient client = new(
    accountEndpoint:
builder.Configuration["AZURE_COSMOS_DB_NOSQL_ENDPOINT"]!,
    tokenCredential: new DefaultAzureCredential()
);
```

Obtenir une base de données

Utilisez `client.GetDatabase` pour récupérer la base de données existante nommée `cosmicworks`.

C#

```
Database database = client.GetDatabase("cosmicworks");
```

Obtenir un conteneur

Récupérez le conteneur `products` existant à l'aide de `database.GetContainer`.

C#

```
Container container = database.GetContainer("products");
```

Créer un élément

Générez un type d'enregistrement C# avec tous les membres que vous souhaitez sérialiser dans JSON. Dans cet exemple, le type a un identificateur unique et des champs pour la catégorie, le nom, la quantité, le prix et la vente.

C#

```
public record Product(  
    string id,  
    string category,  
    string name,  
    int quantity,  
    decimal price,  
    bool clearance  
);
```

Créez un élément dans le conteneur à l'aide de `container.UpsertItem`. Cette méthode « upserts » remplace efficacement l'élément s'il existe déjà.

C#

```
Product item = new(  
    id: "68719518391",  
    category: "gear-surf-surfboards",  
    name: "Yamba Surfboard",  
    quantity: 12,  
    price: 850.00m,  
    clearance: false  
);  
  
ItemResponse<Product> response = await container.UpsertItemAsync<Product>(  
    item: item,  
    partitionKey: new PartitionKey("gear-surf-surfboards")  
);
```

Lire un élément

Effectuez une opération de lecture de point à l'aide des champs d'identificateur unique (`id`) et de clé de partition. Utilisez `container.ReadItem` pour récupérer efficacement l'élément spécifique.

C#

```
ItemResponse<Product> response = await container.ReadItemAsync<Product>(   
    id: "68719518391",  
    partitionKey: new PartitionKey("gear-surf-surfboards")  
);
```

Éléments de requête

Effectuez une requête sur plusieurs éléments d'un conteneur à l'aide de `container.GetItemQueryIterator`. Recherchez tous les éléments d'une catégorie spécifiée à l'aide de cette requête paramétrable :

nosql

```
SELECT * FROM products p WHERE p.category = @category
```

C#

```
var query = new QueryDefinition(
    query: "SELECT * FROM products p WHERE p.category = @category"
)
    .WithParameter("@category", "gear-surf-surfboards");

using FeedIterator<Product> feed = container.GetItemQueryIterator<Product>(
    queryDefinition: query
);
```

Analysez les résultats paginés de la requête en boucle dans chaque page de résultats à l'aide de `feed.ReadNextAsync`. Utilisez `feed.HasMoreResults` pour déterminer s'il existe des résultats laissés au début de chaque boucle.

C#

```
List<Product> items = new();
double requestCharge = 0d;
while (feed.HasMoreResults)
{
    FeedResponse<Product> response = await feed.ReadNextAsync();
    foreach (Product item in response)
    {
        items.Add(item);
    }
    requestCharge += response.RequestCharge;
}
```

Nettoyer les ressources

Lorsque vous n'avez plus besoin de l'exemple d'application ou de ressources, supprimez le déploiement correspondant et toutes les ressources.

Azure CLI

```
azd down
```

Dans GitHub Codespaces, supprimez l'espace de code en cours d'exécution pour optimiser vos droits de stockage et de base.

Contenu connexe

- [Démarrage rapide JavaScript/Node.js](#)
- [Démarrage rapide Java](#)
- [Démarrage rapide Python](#)
- [Démarrage rapide Go](#)

Étape suivante

[Didacticiel : Développer une application console .NET](#)

Démarrage rapide : envoyer vers et recevoir des événements d'Azure Event Hubs avec .NET

Article • 15/03/2023

Dans ce guide de démarrage rapide, vous allez apprendre à envoyer des événements à un Event Hub et à recevoir ces événements à partir de l'Event Hub à l'aide de la bibliothèque .NET Azure.Messaging.EventHubs.

ⓘ Notes

Les guides de démarrage rapide vous permettent d'accélérer rapidement le service. Si vous êtes déjà familiarisé avec le service, vous pouvez consulter des exemples .NET pour Event Hubs dans notre référentiel SDK .NET sur GitHub : [Exemples Event Hubs sur GitHub](#) , [Exemples de processeurs d'événements sur GitHub](#) .

Prérequis

Si vous débutez avec Azure Event Hubs, consultez la [vue d'ensemble d'Event Hubs](#) avant de suivre ce guide de démarrage rapide.

Pour effectuer ce démarrage rapide, vous avez besoin de ce qui suit :

- **Abonnement Microsoft Azure.** Pour utiliser les services Azure, y compris Azure Event Hubs, vous avez besoin d'un abonnement. Si vous n'avez pas de compte Azure, vous pouvez vous inscrire à un [essai gratuit](#) ou utiliser les avantages de votre abonnement MSDN quand vous [créez un compte](#).
- **Microsoft Visual Studio 2022.** La bibliothèque cliente Azure Event Hubs utilise les nouvelles fonctionnalités introduites dans C# 8.0. Vous pouvez toujours utiliser la bibliothèque avec les versions précédentes du langage C#, mais la nouvelle syntaxe n'est pas disponible. Pour utiliser la syntaxe complète, nous vous recommandons d'effectuer la compilation avec la version 3.0 ou une version ultérieure du [kit SDK .NET Core](#) et la [version du langage latest](#). Si vous utilisez Visual Studio, notez que les versions antérieures à Visual Studio 2022 ne sont pas compatibles avec les outils nécessaires à la génération de projets C# 8.0. Visual Studio 2022, y compris l'édition Community gratuite, est téléchargeable [ici](#).
- **Créez un espace de noms Event Hubs et un Event Hub.** La première étape consiste à utiliser le portail Azure pour créer un espace de noms Event Hubs et un

Event Hub dans l'espace de noms. Obtenez ensuite les informations de gestion nécessaires à votre application pour communiquer avec l'Event Hub. Pour créer un espace de noms et un Event Hub, consultez [Démarrage rapide : créer un Event Hub avec le portail Azure](#).

Authentifier l'application sur Azure

Ce guide de démarrage rapide vous montre deux façons de vous connecter à Azure Event Hubs :

- Sans mot de passe (authentification Microsoft Entra)
- Connection string

La première option vous explique comment utiliser votre principal de sécurité dans Azure **Active Directory et le contrôle d'accès en fonction du rôle (RBAC)** pour vous connecter à un espace de noms Event Hubs. Vous n'avez pas à vous soucier d'avoir des chaînes de connexion codées en dur dans votre code, dans un fichier config ni dans un stockage sécurisé comme Azure Key Vault.

La deuxième option consiste à se servir d'une **chaîne de connexion** pour se connecter à un espace de noms Event Hubs. Si vous débutez avec Azure, vous trouverez peut-être l'option de chaîne de connexion plus facile à suivre. Nous vous recommandons d'utiliser l'option sans mot de passe dans les applications réelles et les environnements de production. Pour plus d'informations, consultez [Authentification et autorisation](#). Pour en savoir plus sur l'authentification sans mot de passe, reportez-vous à la [page de présentation](#).

Sans mot de passe

Attribuer des rôles à votre utilisateur Microsoft Entra

Lors du développement localement, vous devez vérifier que le compte d'utilisateur qui se connecte à Azure Event Hubs dispose des autorisations appropriées. Vous aurez besoin du rôle [Propriétaire de données Azure Event Hubs](#) pour envoyer et recevoir des messages. Pour vous attribuer ce rôle, vous aurez besoin du rôle Administrateur de l'accès utilisateur ou d'un autre rôle qui inclut l'action `Microsoft.Authorization/roleAssignments/write`. Vous pouvez attribuer des rôles RBAC Azure à un utilisateur à l'aide du Portail Azure, Azure CLI ou Azure PowerShell. Découvrez les étendues disponibles pour les attributions de rôles dans la page [vue d'ensemble de l'étendue](#).

L'exemple suivant attribue le rôle **Azure Event Hubs Data Owner** à votre compte d'utilisateur, qui fournit un accès complet aux ressources Azure Event Hubs. Dans un scénario réel, suivez le [principe des priviléges minimum](#) pour accorder aux utilisateurs uniquement les autorisations minimales nécessaires à un environnement de production plus sécurisé.

Rôles intégrés Azure pour Azure Event Hubs

Pour Azure Event Hubs, la gestion des espaces de noms et de toutes les ressources associées via le portail Azure et l'API de gestion des ressources Azure est déjà protégée à l'aide du modèle RBAC Azure. Azure fournit les rôles Azure intégrés ci-dessous pour autoriser l'accès à un espace de noms Event Hubs :

- [Propriétaire de données Azure Event Hubs](#) : permet l'accès aux données d'un espace de noms Event Hubs et de ses entités (files d'attente, rubriques, abonnements et filtres).
- [Expéditeur de données Azure Event Hubs](#) : utilisez ce rôle pour autoriser l'expéditeur à accéder à l'espace de noms Event Hubs et à ses entités.
- [Récepteur de données Azure Event Hubs](#) : utilisez ce rôle pour autoriser l'expéditeur à accéder à l'espace de noms Event Hubs et à ses entités.

Si vous souhaitez créer un rôle personnalisé, consultez [Droits requis pour les opérations Service Bus](#).

ⓘ Important

Dans la plupart des cas, la propagation de l'attribution de rôle dans Azure peut prendre une ou deux minutes. Dans de rares cas, cela peut prendre jusqu'à huit minutes. Si vous recevez des erreurs d'authentification lorsque vous exécutez votre code pour la première fois, patientez quelques instants et réessayez.

Azure portal

1. Dans le portail Azure, recherchez votre espace de noms Event Hubs à l'aide de la barre de recherche principale ou de la navigation gauche.
2. Dans la page vue d'ensemble, sélectionnez **Contrôle d'accès (IAM)** dans le menu de gauche.

3. Sur la page Contrôle d'accès (IAM), sélectionnez l'onglet Attributions de rôles.

4. Sélectionnez + Ajouter dans le menu supérieur, puis Ajouter une attribution de rôle dans le menu déroulant résultant.

The screenshot shows the 'Access control (IAM)' page for a Service Bus Namespace named 'spsbusns11102'. The left sidebar lists various options like Overview, Activity log, Tags, Diagnose and solve problems, Settings, Entities, Queues, and Topics. The 'Access control (IAM)' option is selected and highlighted with a red box and a yellow arrow labeled 1. At the top right, there's a search bar, a 'Download role assignments' button, and other navigation links. Below the search bar, there's a 'Role assignments' section with a 'Check access' button. The main area has two sections: 'Grant access to this resource' (with a 'Add role assignment' button) and 'View access to this resource' (with a 'View' button). A tooltip for the 'Add role assignment' button is visible, with a yellow arrow labeled 3 pointing to it.

5. Utilisez la zone de recherche pour filtrer les résultats sur le rôle souhaité.

Pour cet exemple, recherchez **Azure Event Hubs Data Owner** et sélectionnez le résultat correspondant. Ensuite, choisissez **Suivant**.

6. Sous **Attribuer l'accès à**, sélectionnez **Utilisateur, groupe ou principal de service**, puis sélectionnez + **Sélectionner des membres**.

7. Dans la boîte de dialogue, recherchez votre nom d'utilisateur Microsoft Entra (généralement votre adresse e-mail *user@domain*), puis choisissez **Sélectionner** en bas de la boîte de dialogue.

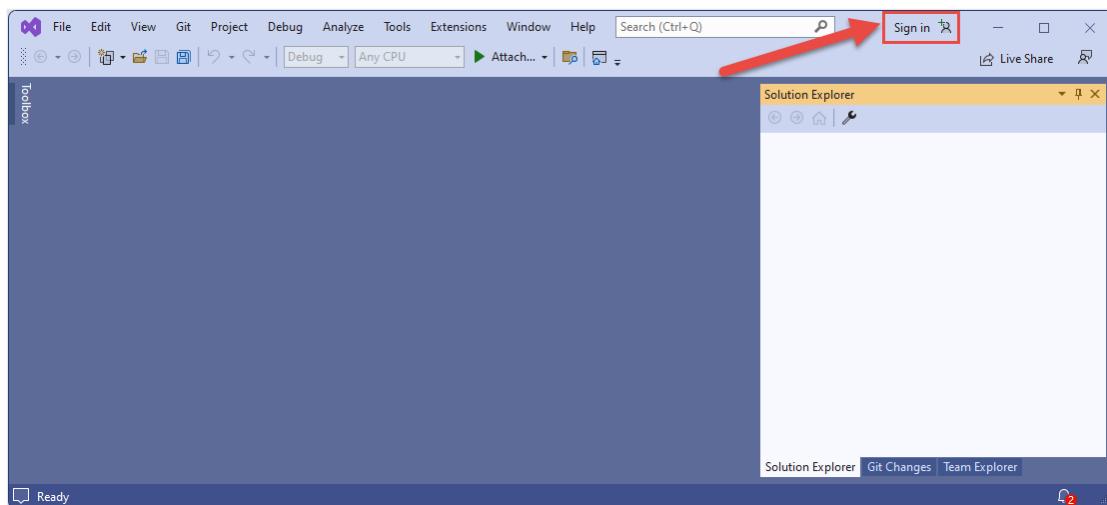
8. Sélectionnez **Vérifier + affecter** pour accéder à la page finale, puis **Vérifier + attribuer** à nouveau pour terminer le processus.

Lancer Visual Studio et vous connecter à Azure

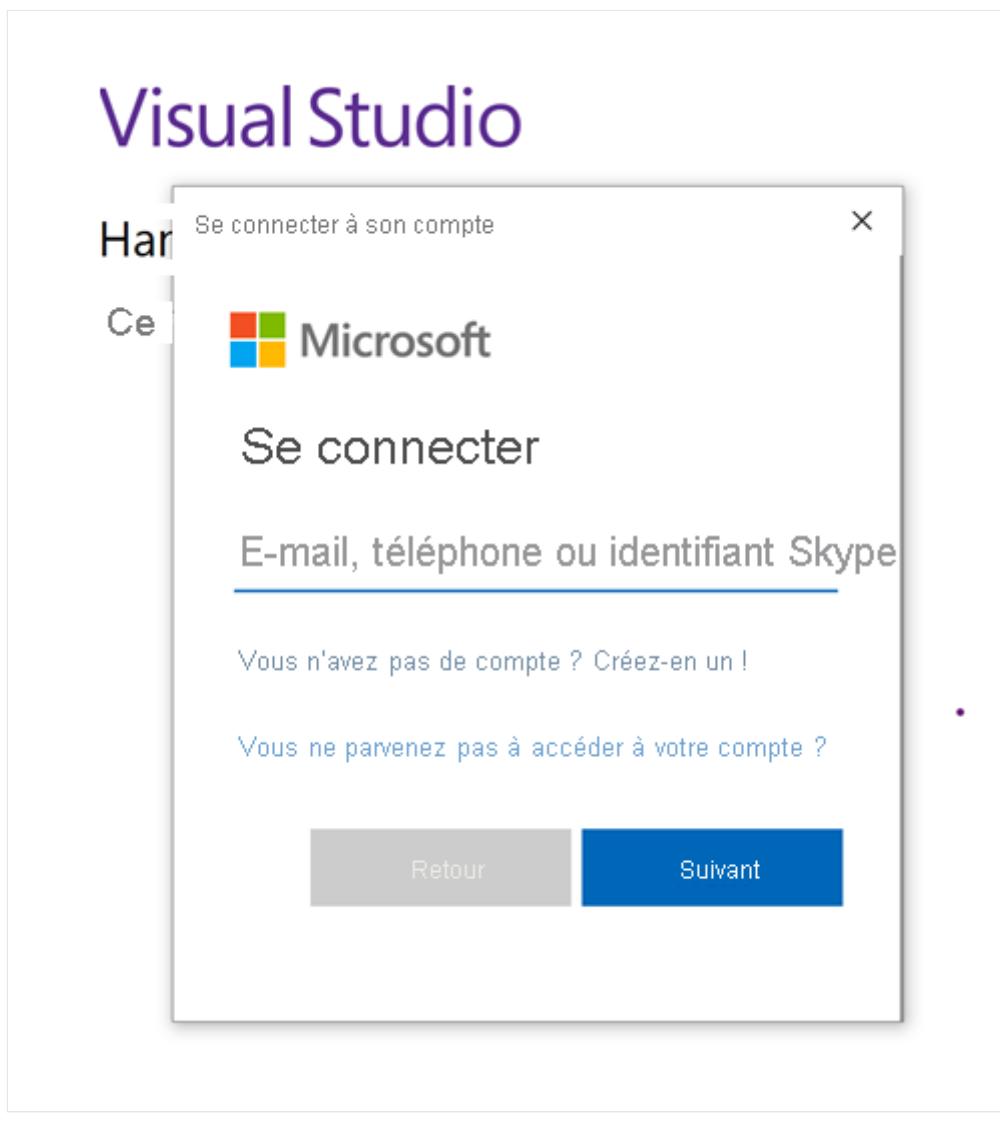
Vous pouvez autoriser l'accès à l'espace de noms Service Bus en procédant comme suit :

1. Lancez Visual Studio. Si la fenêtre **Démarrage** s'affiche, sélectionnez le lien **Continuer sans code** dans le volet de droite.

2. Sélectionnez le bouton **Se connecter** en haut à droite de Visual Studio.



3. Connectez-vous à l'aide du compte Microsoft Entra auquel vous avez attribué un rôle précédemment.

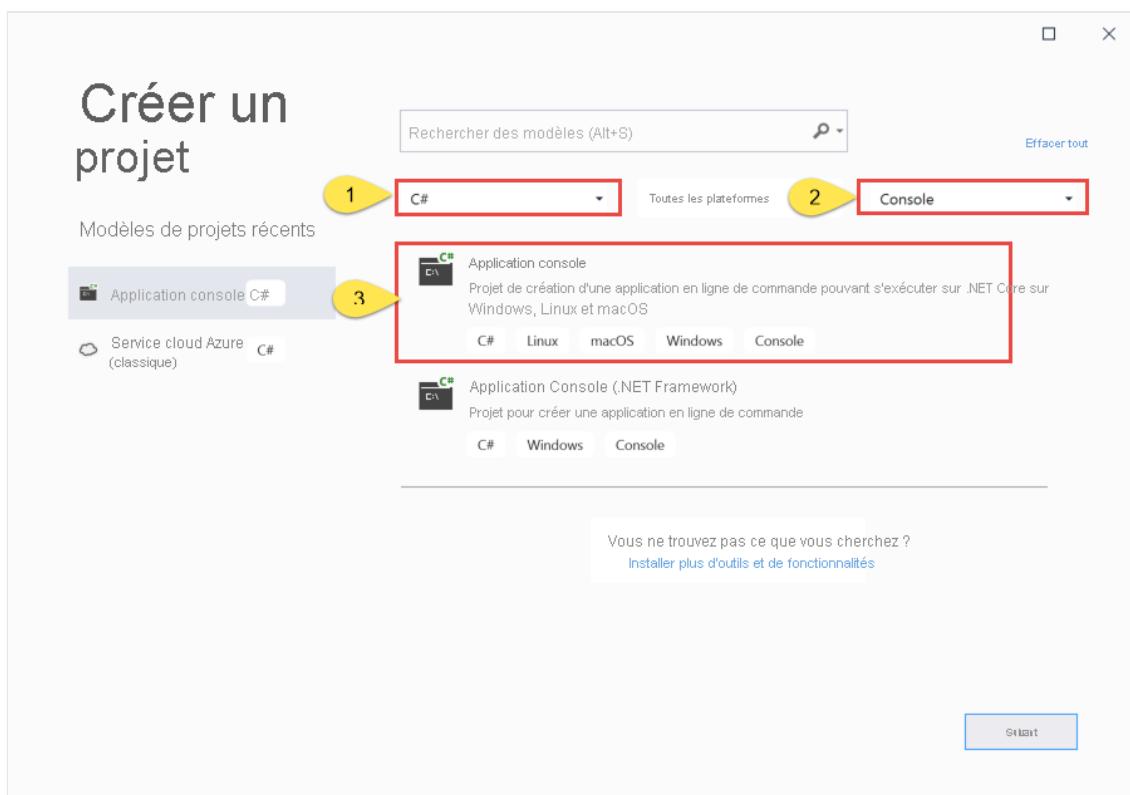


Envoyez des événements à l'Event Hub

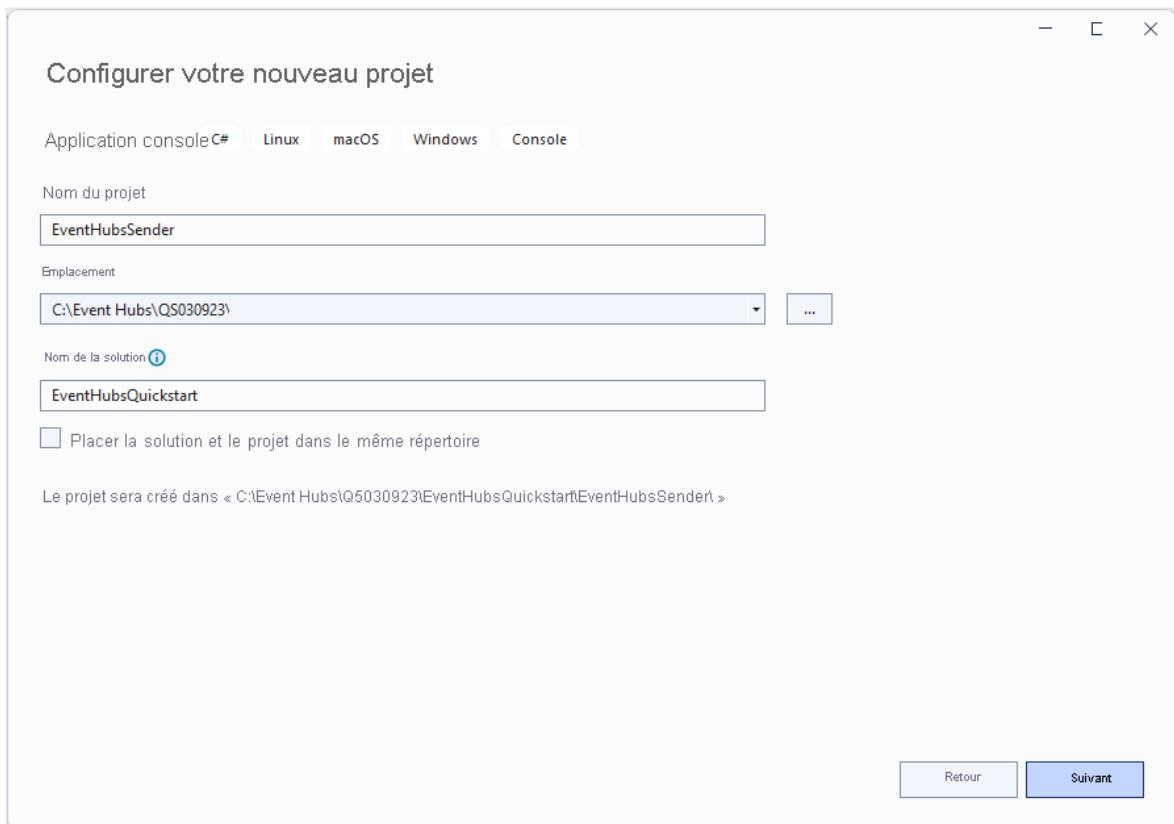
Cette section montre comment créer une application console .NET Core pour envoyer des événements à un Event Hub que vous avez créé.

Création d'une application console

1. Si Visual Studio 2022 est déjà ouvert, sélectionnez **Fichier** dans le menu, sélectionnez **Nouveau**, puis **Projet**. Sinon, lancez Visual Studio 2022 et sélectionnez **Créer un projet** si une fenêtre contextuelle s'affiche.
2. Dans la boîte de dialogue **Créer un projet**, effectuez les étapes suivantes : Si vous ne voyez pas cette boîte de dialogue, sélectionnez **Fichier** dans le menu, sélectionnez **Nouveau**, puis **Projet**.
 - a. Sélectionnez **C#** en guise de langage de programmation.
 - b. Sélectionnez **Console** comme type de l'application.
 - c. Sélectionnez **Application console** dans la liste des résultats.
 - d. Ensuite, cliquez sur **Suivant**.



3. Entrez **EventHubsSender** comme nom de projet, **EventHubsQuickStart** comme nom de solution, puis sélectionnez **Suivant**.



4. Sur la page **Informations supplémentaires**, sélectionnez **Créer**.

Ajouter les paquets NuGet au projet

Sans mot de passe (recommandé)

1. Cliquez sur **Outils > Gestionnaire de package NuGet > Console du Gestionnaire de package** à partir du menu.
2. Exécutez les commandes suivantes pour installer les packages NuGet **Azure.Messaging.EventHubs** et **Azure.Identity**. Appuyez sur **ENTRÉE** pour exécuter la seconde commande.

PowerShell

```
Install-Package Azure.Messaging.EventHubs
Install-Package Azure.Identity
```

Écriture de code pour envoyer des événements au hub d'événements

Sans mot de passe (recommandé)

1. Remplacez le code existant dans le fichier `Program.cs` par l'exemple de code suivant. Ensuite, remplacez les valeurs d'espace réservé `<EVENT_HUB_NAMESPACE>` et `<HUB_NAME>` des paramètres `EventHubProducerClient` par les noms de votre espace de noms Event Hubs et du Event Hub Par exemple :
`"spehubns0309.servicebus.windows.net"` et `"spehub"`.

Voici les étapes importantes du code :

- a. Crée un objet `EventHubProducerClient` à l'aide de l'espace de noms et du nom du hub d'événements.
- b. Appelle la méthode `CreateBatchAsync` sur l'objet `EventHubProducerClient` pour créer un objet `EventDataBatch`.
- c. Ajoutez des événements au lot à l'aide de la méthode `EventDataBatch.TryAdd`.
- d. Envoie le lot de messages à l'Event Hub à l'aide de la méthode `EventHubProducerClient.SendAsync`.

C#

```
using Azure.Identity;
using Azure.Messaging.EventHubs;
using Azure.Messaging.EventHubs.Producer;
using System.Text;

// number of events to be sent to the event hub
int numOfEvents = 3;

// The Event Hubs client types are safe to cache and use as a
// singleton for the lifetime
// of the application, which is best practice when events are being
// published or read regularly.
// TODO: Replace the <EVENT_HUB_NAMESPACE> and <HUB_NAME>
// placeholder values
EventHubProducerClient producerClient = new EventHubProducerClient(
    "<EVENT_HUB_NAMESPACE>.servicebus.windows.net",
    "<HUB_NAME>",
    new DefaultAzureCredential());

// Create a batch of events
using EventDataBatch eventBatch = await
producerClient.CreateBatchAsync();

for (int i = 1; i <= numOfEvents; i++)
{
    if (!eventBatch.TryAdd(new
EventData(Encoding.UTF8.GetBytes($"Event {i}"))))
}
```

```
        {
            // if it is too large for the batch
            throw new Exception($"Event {i} is too large for the batch
and cannot be sent.");
        }
    }

try
{
    // Use the producer client to send the batch of events to the
event hub
    await producerClient.SendAsync(eventBatch);
    Console.WriteLine($"A batch of {numOfEvents} events has been
published.");
    Console.ReadLine();
}
finally
{
    await producerClient.DisposeAsync();
}
```

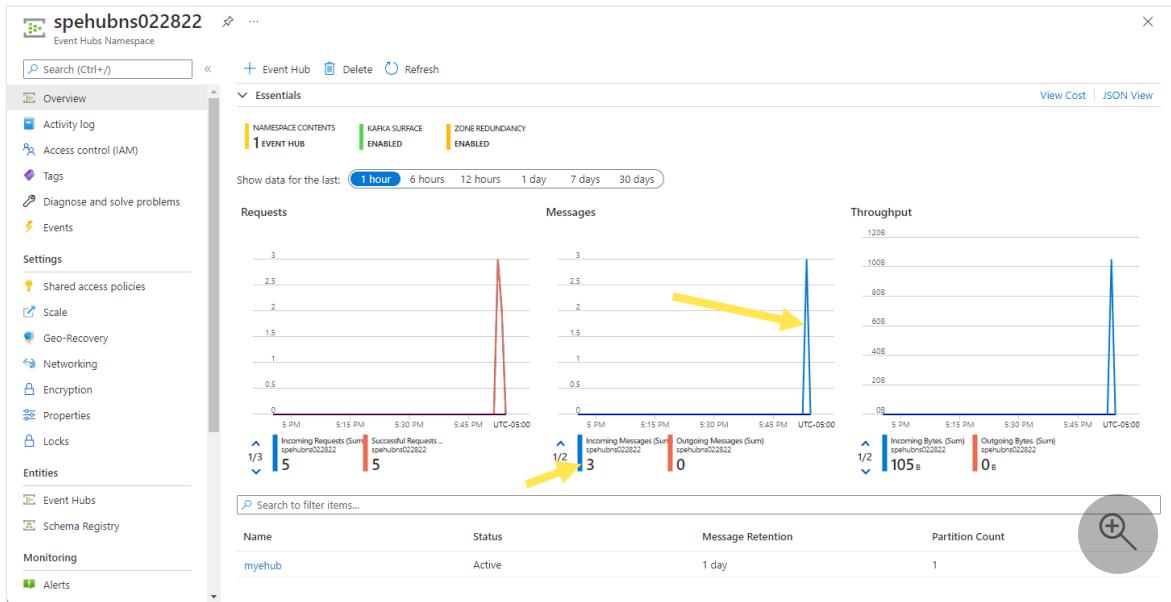
2. Générez le projet et vérifiez qu'il ne présente pas d'erreurs.

3. Exécutez le programme et attendez le message de confirmation.

C#

A batch of 3 events has been published.

4. Dans la page **Espace de noms Event Hubs** du portail Azure, vous voyez trois messages entrants dans le graphique **Messages**. Actualisez la page pour mettre à jour le graphique si nécessaire. Cela peut prendre quelques secondes pour indiquer que les messages ont été reçus.



ⓘ Notes

Pour obtenir le code source complet avec des remarques plus détaillées, consultez ce fichier sur GitHub [↗](#)

Recevez des événements de l'Event Hub

Cette section explique comment écrire une application console .NET Core qui reçoit des événements d'un hub d'événements à l'aide d'un processeur d'événements. Le processeur d'événements simplifie la réception des événements à partir des Event Hubs.

Créer un compte de stockage Azure et un conteneur d'objets blob

Dans ce guide de démarrage rapide, vous utilisez Stockage Azure comme magasin de points de contrôle. Suivez les étapes ci-dessous pour créer un compte Stockage Azure.

1. [Création d'un compte de stockage Azure](#)
2. Créez un [conteneur d'objets blob](#)
3. Authentifiez-vous auprès du conteneur d'objets blob à l'aide de l'authentification Microsoft Entra ID (sans mot de passe) ou d'une chaîne de connexion à l'espace de noms.

Suivez les recommandations ci-dessous quand vous utilisez Stockage Blob Azure comme magasin de points de contrôle :

- Utilisez un conteneur distinct pour chaque groupe de processeurs. Vous pouvez utiliser le même compte de stockage, mais utiliser un conteneur par groupe.
- N'utilisez pas le conteneur et le compte de stockage pour quoi que ce soit d'autre.
- Le compte de stockage doit se trouver dans la même région que l'application déployée. Si l'application est locale, essayez de choisir la région la plus proche possible.

Sur la page **Compte de stockage** du Portail Azure, dans la section **Service BLOB**, vérifiez que les paramètres suivants sont désactivés.

- Espace de noms hiérarchique
- Suppression réversible de blob
- Contrôle de version

Sans mot de passe (recommandé)

Lors du développement local, assurez-vous que le compte d'utilisateur qui accède aux données blob dispose des autorisations appropriées. Vous aurez besoin du **Contributeur aux données Blob de stockage** pour lire et écrire des données blob. Pour vous attribuer ce rôle, vous aurez besoin du rôle **Administrateur de l'accès utilisateur** ou d'un autre rôle qui inclut l'action **Microsoft.Authorization/roleAssignments/write**. Vous pouvez attribuer des rôles RBAC Azure à un utilisateur à l'aide du Portail Azure, Azure CLI ou Azure PowerShell. Vous pouvez en savoir plus sur les étendues disponibles pour les attributions de rôles dans la page [vue d'ensemble de l'étendue](#).

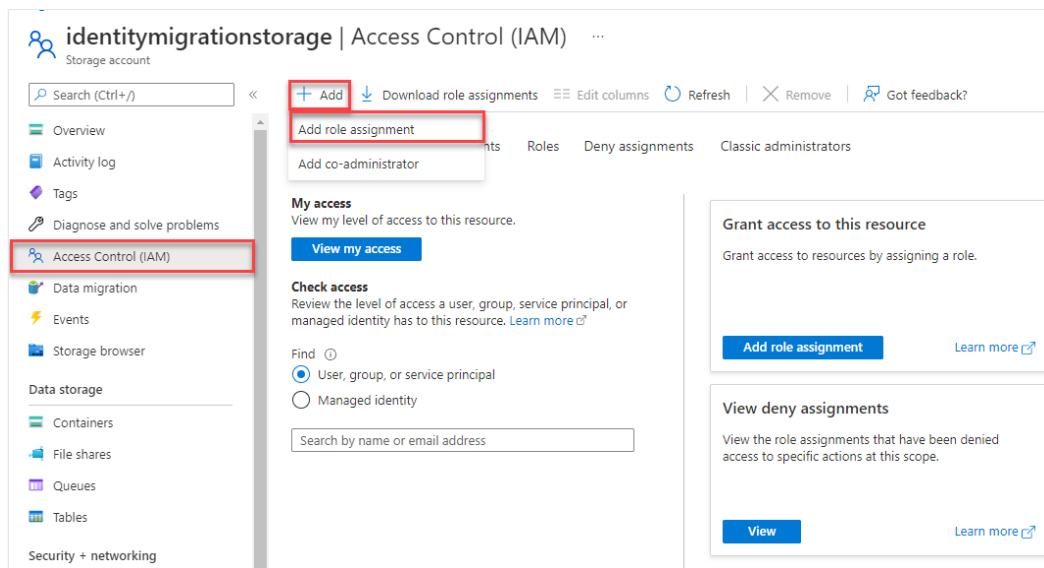
Dans ce scénario, vous allez attribuer des autorisations à votre compte d'utilisateur, étendues au compte de stockage, pour suivre le [Principe des priviléges minimum](#). Cette pratique offre aux utilisateurs uniquement les autorisations minimales nécessaires et crée des environnements de production plus sécurisés.

L'exemple suivant affecte le rôle **Contributeur aux données Blob du stockage** à votre compte d'utilisateur, qui fournit à la fois un accès en lecture et en écriture aux données d'objet blob dans votre compte de stockage.

ⓘ Important

Dans la plupart des cas, la propagation de l'attribution de rôle dans Azure peut prendre une ou deux minutes, mais dans de rares cas, cela peut prendre jusqu'à huit minutes. Si vous recevez des erreurs d'authentification lorsque vous exécutez votre code pour la première fois, patientez quelques instants et réessayez.

1. Dans le Portail Azure, recherchez votre compte de stockage à l'aide de la barre de recherche principale ou de la navigation gauche.
2. Dans la page vue d'ensemble du compte de stockage, sélectionnez **Contrôle d'accès (IAM)** dans le menu de gauche.
3. Sur la page **Contrôle d'accès (IAM)**, sélectionnez l'onglet **Attributions de rôles**.
4. Sélectionnez **+ Ajouter** dans le menu supérieur, puis **Ajouter une attribution de rôle** dans le menu déroulant résultant.



5. Utilisez la zone de recherche pour filtrer les résultats sur le rôle souhaité. Pour cet exemple, recherchez *Contributeur aux données Blob du stockage*, sélectionnez le résultat correspondant, puis choisissez **Suivant**.
6. Sous **Attribuer l'accès à**, sélectionnez **Utilisateur, groupe ou principal de service**, puis sélectionnez **+ Sélectionner des membres**.
7. Dans la boîte de dialogue, recherchez votre nom d'utilisateur Microsoft Entra (généralement votre adresse e-mail *user@domain*), puis choisissez **Sélectionner** en bas de la boîte de dialogue.
8. Sélectionnez **Vérifier + affecter** pour accéder à la page finale, puis **Vérifier + attribuer** à nouveau pour terminer le processus.

Créer un projet pour le récepteur

1. Dans la fenêtre Explorateur de solutions, cliquez avec le bouton droit sur la solution **EventHubQuickStart**, pointez sur **Ajouter**, puis sélectionnez **Nouveau projet**.
2. Sélectionnez **Application console**, puis **Suivant**.
3. Entrez **EventHubsReceiver** pour **Nom du projet**, puis sélectionnez **Créer**.
4. Dans la fenêtre **Explorateur de solutions**, cliquez avec le bouton droit sur **EventHubsReceiver**, puis sélectionnez **Définir comme projet de démarrage**.

Ajouter les paquets NuGet au projet

Sans mot de passe (recommandé)

1. Cliquez sur **Outils > Gestionnaire de package NuGet > Console du Gestionnaire de package** à partir du menu.
2. Dans la fenêtre **Console du gestionnaire de package**, vérifiez que **EventHubsReceiver** est sélectionné pour le **Projet par défaut**. Si ce n'est pas le cas, utilisez la liste déroulante pour sélectionner **EventHubsReceiver**.
3. Exécutez la commande suivante pour installer les packages NuGet **Azure.Messaging.EventHubs** et **Azure.Identity** : Appuyez sur **ENTRÉE** pour exécuter la dernière commande.

PowerShell

```
Install-Package Azure.Messaging.EventHubs
Install-Package Azure.Messaging.EventHubs.Processor
Install-Package Azure.Identity
```

Mettez à jour le code

Remplacez le contenu du fichier **Program.cs** par le code suivant :

Sans mot de passe (recommandé)

1. Remplacez le code existant dans le fichier **Program.cs** par l'exemple de code suivant. Remplacez ensuite les valeurs d'espace réservé

<STORAGE_ACCOUNT_NAME> et <BLOB_CONTAINER_NAME> pour l'URI BlobContainerClient. Remplacez également les valeurs d'espace réservé <EVENT_HUB_NAMESPACE> et <HUB_NAME> pour EventProcessorClient.

Voici les étapes importantes du code :

- a. Crée un objet [EventProcessorClient](#) à l'aide de l'espace de noms et du nom des Event Hubs. Vous devez créer un objet [BlobContainerClient](#) pour le conteneur dans le stockage Azure que vous avez créé précédemment.
- b. Spécifie des gestionnaires pour les événements [ProcessEventAsync](#) et [ProcessErrorAsync](#) de l'objet [EventProcessorClient](#).
- c. Démarre le traitement des événements en appelant [StartProcessingAsync](#) sur l'objet [EventProcessorClient](#).
- d. Arrête le traitement des événements après 30 secondes en appelant [StopProcessingAsync](#) sur l'objet [EventProcessorClient](#).

C#

```
using Azure.Identity;
using Azure.Messaging.EventHubs;
using Azure.Messaging.EventHubs.Consumer;
using Azure.Messaging.EventHubs.Processor;
using Azure.Storage.Blobs;
using System.Text;

// Create a blob container client that the event processor will use
// TODO: Replace <STORAGE_ACCOUNT_NAME> and <BLOB_CONTATINAER_NAME>
// with actual names
BlobContainerClient storageClient = new BlobContainerClient(
    new
Uri("https://<STORAGE_ACCOUNT_NAME>.blob.core.windows.net/<BLOB_CON
TAINER_NAME>"),
    new DefaultAzureCredential());

// Create an event processor client to process events in the event
hub
// TODO: Replace the <EVENT_HUBS_NAMESPACE> and <HUB_NAME>
placeholder values
var processor = new EventProcessorClient(
    storageClient,
    EventHubConsumerClient.DefaultConsumerGroupName,
    "<EVENT_HUB_NAMESPACE>.servicebus.windows.net",
    "<HUB_NAME>",
    new DefaultAzureCredential());

// Register handlers for processing events and handling errors
processor.ProcessEventAsync += ProcessEventHandler;
processor.ProcessErrorAsync += ProcessErrorHandler;

// Start the processing
await processor.StartProcessingAsync();
```

```

// Wait for 30 seconds for the events to be processed
await Task.Delay(TimeSpan.FromSeconds(30));

// Stop the processing
await processor.StopProcessingAsync();

Task ProcessEventHandler(ProcessEventArgs eventArgs)
{
    // Write the body of the event to the console window
    Console.WriteLine("\tReceived event: {0}",
        Encoding.UTF8.GetString(eventArgs.Data.Body.ToArray()));
    Console.ReadLine();
    return Task.CompletedTask;
}

Task ProcessErrorHandler(ProcessErrorEventArgs eventArgs)
{
    // Write details about the error to the console window
    Console.WriteLine($"\"\\tPartition '{eventArgs.PartitionId}': an
unhandled exception was encountered. This was not expected to
happen.\"");
    Console.WriteLine(eventArgs.Exception.Message);
    Console.ReadLine();
    return Task.CompletedTask;
}

```

2. Générez le projet et vérifiez qu'il ne présente pas d'erreurs.

ⓘ Notes

Pour obtenir le code source complet avec des remarques plus détaillées, consultez [ce fichier sur GitHub ↗](#).

3. Exécutez l'application réceptrice.
4. Vous devriez voir un message indiquant que les événements ont été reçus.

Bash

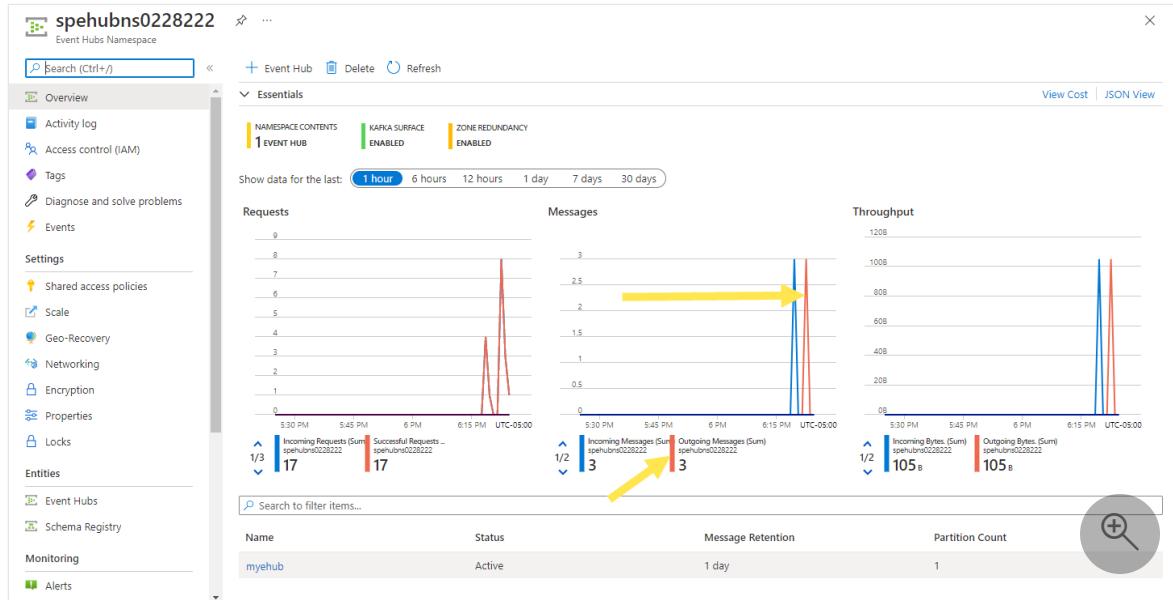
```

Received event: Event 1
Received event: Event 2
Received event: Event 3

```

Ces événements sont les trois événements que vous avez envoyés au hub d'événements en exécutant le programme émetteur.

5. Dans le portail Azure, vous pouvez vérifier qu'il existe trois messages sortants, qu'Event Hubs a envoyés à l'application de réception. Actualisez la page pour mettre à jour le graphique. Cela peut prendre quelques secondes pour indiquer que les messages ont été reçus.



Validation de schéma pour les applications basées sur le Kit de développement logiciel (SDK) Event Hubs

Vous pouvez utiliser Azure Schema Registry pour effectuer la validation de schéma lorsque vous diffusez des données avec vos applications basées sur le SDK Event Hubs. Azure Schema Registry of Event Hubs fournit un référentiel centralisé pour la gestion des schémas et vous pouvez connecter en toute transparence vos applications nouvelles ou existantes à Schema Registry.

Pour plus d'informations, consultez [Valider des schémas avec le Kit de développement logiciel \(SDK\) Event Hubs](#).

Nettoyer les ressources

Supprimez le groupe de ressources qui a l'espace de noms Event Hubs ou supprimez uniquement l'espace de noms si vous souhaitez conserver le groupe de ressources.

Exemples et référence

Ce guide de démarrage rapide fournit des instructions pas à pas pour implémenter un scénario qui consiste à envoyer un lot d'évènements à un Event Hub, puis à les recevoir. Pour plus d'exemples, sélectionnez les liens suivants.

- [Exemples Event Hubs sur GitHub](#)↗
- [Exemples de processeurs d'événement sur GitHub](#)↗
- [Exemple de contrôle d'accès en fonction du rôle Azure \(Azure RBAC\)](#)↗

Pour obtenir des informations de référence complètes sur la bibliothèque .NET, consultez notre [documentation du SDK](#).

Étapes suivantes

Consultez le tutoriel suivant :

Tutoriel : Visualiser les anomalies des données dans les événements en temps réel envoyés à Azure Event Hubs

Démarrage rapide : Bibliothèque de client de certificats Azure Key Vault pour .NET

Article • 25/03/2023

Découvrez comment démarrer avec la bibliothèque de client de certificats Azure Key Vault pour .NET. [Azure Key Vault](#) est un service cloud qui fournit un magasin de certificats sécurisé. Vous pouvez stocker des clés, des mots de passe, des certificats et d'autres secrets en toute sécurité. Vous pouvez créer et gérer des coffres de clés Azure grâce au portail Azure. Dans ce guide de démarrage rapide, vous allez découvrir comment créer, récupérer et supprimer des certificats dans un coffre de clés Azure en utilisant la bibliothèque de client .NET.

Ressources de la bibliothèque de client Key Vault :

[Documentation de référence sur l'API](#) | [Code source de la bibliothèque](#) ↗ | [Package \(NuGet\)](#) ↗

Pour plus d'informations sur Key Vault et les certificats, consultez :

- [Vue d'ensemble du coffre de clés](#)
- [Vue d'ensemble des certificats](#)

Prérequis

- Un abonnement Azure : [créez-en un gratuitement](#) ↗
- [SDK .NET 6 ou ultérieur](#) ↗
- [Azure CLI](#)
- Un coffre de clés. Vous pouvez en créer un en utilisant le [portail Azure](#), [Azure CLI](#) ou [Azure PowerShell](#)

Ce guide de démarrage rapide utilise `dotnet` et Azure CLI

Programme d'installation

Ce guide de démarrage rapide utilise la bibliothèque Azure Identity avec Azure CLI pour authentifier l'utilisateur auprès des services Azure. Les développeurs peuvent également utiliser Visual Studio ou Visual Studio Code pour authentifier leurs appels. Pour plus

d'informations, consultez [Authentifier le client avec la bibliothèque de client Azure Identity](#).

Connexion à Azure

1. Exécutez la commande `login`.

```
Azure CLI
```

```
az login
```

Si l'interface CLI peut ouvrir votre navigateur par défaut, elle le fait et charge une page de connexion Azure par la même occasion.

Sinon, ouvrez une page de navigateur à l'adresse <https://aka.ms/devicelogin> et entrez le code d'autorisation affiché dans votre terminal.

2. Dans le navigateur, connectez-vous avec les informations d'identification de votre compte.

Accorder l'accès à votre coffre de clés

Créez une stratégie d'accès pour votre coffre de clés, qui accorde des autorisations de certificat à votre compte d'utilisateur.

```
Azure CLI
```

```
az keyvault set-policy --name <your-key-vault-name> --upn user@domain.com --  
certificate-permissions delete get list create purge
```

Créer une application console .NET

1. Dans un shell de commandes, exécutez la commande suivante pour créer un projet nommé `key-vault-console-app` :

```
CLI .NET
```

```
dotnet new console --name key-vault-console-app
```

2. Accédez au répertoire `key-vault-console-app` nouvellement créé, puis exécutez la commande suivante pour générer le projet :

CLI .NET

```
dotnet build
```

La sortie de génération ne doit contenir aucun avertissement ni erreur.

Console

```
Build succeeded.  
0 Warning(s)  
0 Error(s)
```

Installer les packages

À partir de l'interpréteur de commandes, installez la bibliothèque de client de certificats Azure Key Vault pour .NET :

CLI .NET

```
dotnet add package Azure.Security.KeyVault.Certificates
```

Pour ce guide de démarrage rapide, vous devez également installer la bibliothèque de client Azure Identity :

CLI .NET

```
dotnet add package Azure.Identity
```

Définir des variables d'environnement

Cette application utilise le nom de coffre de clés en tant que variable d'environnement appelée `KEY_VAULT_NAME`.

Windows

Invite de commandes Windows

```
set KEY_VAULT_NAME=<your-key-vault-name>
```

Windows PowerShell

PowerShell

```
$Env:KEY_VAULT_NAME="your-key-vault-name"
```

macOS ou Linux

Bash

```
export KEY_VAULT_NAME=<your-key-vault-name>
```

Modèle objet

La bibliothèque de client de certificats Azure Key Vault pour .NET vous permet de gérer les certificats. La section [Exemple de code](#) montre comment créer un client et comment définir, récupérer et supprimer un certificat.

Exemples de code

Ajouter des directives

Ajoutez les directives suivantes en haut de *Program.cs* :

C#

```
using System;
using Azure.Identity;
using Azure.Security.KeyVault.Certificates;
```

Authentifier et créer un client

Les requêtes d'application vers les Services Azure doivent être autorisées. L'utilisation de la classe [DefaultAzureCredential](#) fournie par la [bibliothèque de client Azure Identity](#) est l'approche recommandée pour implémenter des connexions sans mot de passe aux services Azure dans votre code. `DefaultAzureCredential` prend en charge plusieurs méthodes d'authentification et détermine quelle méthode doit être utilisée au moment de l'exécution. Cette approche permet à votre application d'utiliser différentes méthodes d'authentification dans différents environnements (local ou production) sans implémenter de code spécifique à l'environnement.

Dans ce guide de démarrage rapide, `DefaultAzureCredential` s'authentifie auprès du coffre de clés à l'aide des informations d'identification de l'utilisateur de développement

local connecté à Azure CLI. Quand l'application est déployée sur Azure, le même code `DefaultAzureCredential` peut découvrir et utiliser automatiquement une identité managée affectée à un service d'application, une machine virtuelle ou d'autres services. Pour plus d'informations, consultez [Vue d'ensemble des identités managées](#).

Dans cet exemple, le nom de votre coffre de clés est étendu à l'URI du coffre de clés, au format `https://<your-key-vault-name>.vault.azure.net`. Pour plus d'informations sur l'authentification auprès du coffre de clés, consultez le [Guide du développeur](#).

```
C#
```

```
string keyVaultName = Environment.GetEnvironmentVariable("KEY_VAULT_NAME");
var kvUri = "https://" + keyVaultName + ".vault.azure.net";

var client = new CertificateClient(new Uri(kvUri), new
DefaultAzureCredential());
```

Enregistrer un certificat

Dans cet exemple, pour faire simple, vous pouvez utiliser un certificat auto-signé avec la stratégie d'émission par défaut. Pour cette tâche, utilisez la méthode `StartCreateCertificateAsync`. Les paramètres de la méthode acceptent un nom de certificat et la [stratégie de certificat](#).

```
C#
```

```
var operation = await client.StartCreateCertificateAsync("myCertificate",
CertificatePolicy.Default);
var certificate = await operation.WaitForCompletionAsync();
```

ⓘ Notes

Si le nom du certificat existe, le code ci-dessus crée une nouvelle version de ce certificat.

Récupérer un certificat

Vous pouvez maintenant récupérer le certificat créé précédemment à l'aide de la méthode `GetCertificateAsync`.

```
C#
```

```
var certificate = await client.GetCertificateAsync("myCertificate");
```

Supprimer un certificat

Pour finir, nous allons supprimer et purger le certificat de votre coffre de clés avec les méthodes `StartDeleteCertificateAsync` et `PurgeDeletedCertificateAsync`.

C#

```
var operation = await client.StartDeleteCertificateAsync("myCertificate");

// You only need to wait for completion if you want to purge or recover the
// certificate.
await operation.WaitForCompletionAsync();

var certificate = operation.Value;
await client.PurgeDeletedCertificateAsync("myCertificate");
```

Exemple de code

Modifiez l'application console .NET pour interagir avec le coffre de clés en effectuant les étapes suivantes :

- Remplacez le code dans `Program.cs` par le code suivant :

C#

```
using System;
using System.Threading.Tasks;
using Azure.Identity;
using Azure.Security.KeyVault.Certificates;

namespace key_vault_console_app
{
    class Program
    {
        static async Task Main(string[] args)
        {
            const string certificateName = "myCertificate";
            var keyVaultName =
                Environment.GetEnvironmentVariable("KEY_VAULT_NAME");
            var kvUri = $"https://{{keyVaultName}}.vault.azure.net";

            var client = new CertificateClient(new Uri(kvUri), new
DefaultAzureCredential());

            Console.WriteLine($"Creating a certificate in {keyVaultName}
```

```

    called '{certificateName}' ...");
        CertificateOperation operation = await
client.StartCreateCertificateAsync(certificateName,
CertificatePolicy.Default);
        await operation.WaitForCompletionAsync();
Console.WriteLine(" done.");

        Console.WriteLine($"Retrieving your certificate from
{keyVaultName}.");
        var certificate = await
client.GetCertificateAsync(certificateName);
        Console.WriteLine($"Your certificate version is
'{certificate.Value.Properties.Version}'.");

        Console.Write($"Deleting your certificate from
{keyVaultName} ...");
        DeleteCertificateOperation deleteOperation = await
client.StartDeleteCertificateAsync(certificateName);
        // You only need to wait for completion if you want to
purge or recover the certificate.
        await deleteOperation.WaitForCompletionAsync();
Console.WriteLine(" done.");

        Console.Write($"Purging your certificate from
{keyVaultName} ...");
        await client.PurgeDeletedCertificateAsync(certificateName);
Console.WriteLine(" done.");
    }
}
}

```

Tester et vérifier

Exécutez la commande suivante pour générer le projet :

CLI .NET

`dotnet build`

Une variante de la sortie suivante apparaît :

Console

```

Creating a certificate in mykeyvault called 'myCertificate' ... done.
Retrieving your certificate from mykeyvault.
Your certificate version is '8532359bcfd24e4bb2525f2d2050738a'.
Deleting your certificate from mykeyvault ... done
Purging your certificate from mykeyvault ... done

```

Étapes suivantes

Dans ce guide de démarrage rapide, vous avez créé un coffre de clés, stocké un certificat et récupéré ce certificat.

Pour plus d'informations sur Key Vault et comment l'intégrer à vos applications, consultez les articles suivants :

- Lire la [vue d'ensemble Azure Key Vault](#)
- Lire la [vue d'ensemble des certificats](#)
- Consultez un [Tutoriel sur l'accès à Key Vault depuis une application App Service](#)
- Consultez un [Tutoriel sur l'accès à Key Vault depuis une machine virtuelle](#)
- Consulter le [Guide du développeur Azure Key Vault](#)
- Passer en revue la [Vue d'ensemble de la sécurité de Key Vault](#)

Démarrage rapide : Bibliothèque de client de clés Azure Key Vault pour .NET

Article • 09/03/2023

Découvrez comment démarrer avec la bibliothèque de client de clés Azure Key Vault pour .NET. [Azure Key Vault](#) est un service cloud qui fournit un magasin de clés de chiffrement sécurisé. Vous pouvez stocker des clés de chiffrement, des mots de passe, des certificats et d'autres secrets de manière sécurisée. Vous pouvez créer et gérer des coffres de clés Azure grâce au portail Azure. Dans ce guide de démarrage rapide, vous allez découvrir comment créer, récupérer et supprimer des clés dans un coffre de clés Azure en utilisant la bibliothèque de client de clés .NET.

Ressources de la bibliothèque de client de clés Key Vault :

[Documentation de référence sur l'API](#) | [Code source de la bibliothèque](#) ↗ | [Package \(NuGet\)](#) ↗

Pour plus d'informations sur Key Vault et les clés, consultez :

- [Vue d'ensemble du coffre de clés](#)
- [Vue d'ensemble des clés](#)

Prérequis

- Un abonnement Azure : [créez-en un gratuitement](#) ↗
- [SDK .NET 6 ou ultérieur](#) ↗
- [Azure CLI](#)
- Un coffre de clés. Vous pouvez en créer un en utilisant le [portail Azure](#), [Azure CLI](#) ou [Azure PowerShell](#)

Ce guide de démarrage rapide utilise `dotnet` et Azure CLI

Programme d'installation

Ce guide de démarrage rapide utilise la bibliothèque Azure Identity avec Azure CLI pour authentifier l'utilisateur auprès des services Azure. Les développeurs peuvent également utiliser Visual Studio ou Visual Studio Code pour authentifier leurs appels. Pour plus d'informations, consultez [Authentifier le client avec la bibliothèque de client Azure Identity](#).

Connexion à Azure

1. Exécutez la commande `login`.

```
Azure CLI
```

```
az login
```

Si l'interface CLI peut ouvrir votre navigateur par défaut, elle le fait et charge une page de connexion Azure par la même occasion.

Sinon, ouvrez une page de navigateur à l'adresse <https://aka.ms/devicelogin> et entrez le code d'autorisation affiché dans votre terminal.

2. Dans le navigateur, connectez-vous avec les informations d'identification de votre compte.

Accorder l'accès à votre coffre de clés

Créez une stratégie d'accès pour votre coffre de clés, qui accorde des autorisations de clé à votre compte d'utilisateur.

```
Azure CLI
```

```
az keyvault set-policy --name <your-key-vault-name> --upn user@domain.com --key-permissions delete get list create purge
```

Créer une application console .NET

1. Dans un shell de commandes, exécutez la commande suivante pour créer un projet nommé `key-vault-console-app` :

```
CLI .NET
```

```
dotnet new console --name key-vault-console-app
```

2. Accédez au répertoire `key-vault-console-app` nouvellement créé, puis exécutez la commande suivante pour générer le projet :

```
CLI .NET
```

```
dotnet build
```

La sortie de génération ne doit contenir aucun avertissement ni erreur.

Console

```
Build succeeded.  
0 Warning(s)  
0 Error(s)
```

Installer les packages

À partir de l'interpréteur de commandes, installez la bibliothèque de client de clés Azure Key Vault pour .NET :

CLI .NET

```
dotnet add package Azure.Security.KeyVault.Keys
```

Pour ce guide de démarrage rapide, vous devez également installer la bibliothèque de client Azure Identity :

CLI .NET

```
dotnet add package Azure.Identity
```

Définir des variables d'environnement

Cette application utilise le nom de coffre de clés en tant que variable d'environnement appelée `KEY_VAULT_NAME`.

Windows

Invite de commandes Windows

```
set KEY_VAULT_NAME=<your-key-vault-name>
```

Windows PowerShell

PowerShell

```
$Env:KEY_VAULT_NAME="<your-key-vault-name>"
```

macOS ou Linux

Bash

```
export KEY_VAULT_NAME=<your-key-vault-name>
```

Modèle objet

La bibliothèque de client de clés Azure Key Vault pour .NET vous permet de gérer des clés. La section [Exemples de code](#) montre comment créer un client et définir, récupérer et supprimer une clé.

Exemples de code

Ajouter des directives

Ajoutez les directives suivantes en haut de *Program.cs* :

C#

```
using System;
using Azure.Identity;
using Azure.Security.KeyVault.Keys;
```

Authentifier et créer un client

Les requêtes d'application vers les Services Azure doivent être autorisées. L'utilisation de la classe [DefaultAzureCredential](#) fournie par la [bibliothèque de client Azure Identity](#) est l'approche recommandée pour implémenter des connexions sans mot de passe aux services Azure dans votre code. `DefaultAzureCredential` prend en charge plusieurs méthodes d'authentification et détermine quelle méthode doit être utilisée au moment de l'exécution. Cette approche permet à votre application d'utiliser différentes méthodes d'authentification dans différents environnements (local ou production) sans implémenter de code spécifique à l'environnement.

Dans ce guide de démarrage rapide, `DefaultAzureCredential` s'authentifie auprès du coffre de clés à l'aide des informations d'identification de l'utilisateur de développement local connecté à Azure CLI. Quand l'application est déployée sur Azure, le même code `DefaultAzureCredential` peut découvrir et utiliser automatiquement une identité managée affectée à un service d'application, une machine virtuelle ou d'autres services. Pour plus d'informations, consultez [Vue d'ensemble des identités managées](#).

Dans cet exemple, le nom de votre coffre de clés est étendu à l'URI du coffre de clés, au format `https://<your-key-vault-name>.vault.azure.net`. Pour plus d'informations sur l'authentification auprès du coffre de clés, consultez le [Guide du développeur](#).

C#

```
var keyVaultName = Environment.GetEnvironmentVariable("KEY_VAULT_NAME");
var kvUri = $"https://{{keyVaultName}}.vault.azure.net";

var client = new KeyClient(new Uri(kvUri), new DefaultAzureCredential());
```

Enregistrer une clé

Pour cette tâche, utilisez la méthode [CreateKeyAsync](#). Les paramètres de la méthode acceptent un nom de clé et le [type de clé](#).

C#

```
var key = await client.CreateKeyAsync("myKey", KeyType.Rsa);
```

ⓘ Notes

Si le nom de clé existe, ce code crée une nouvelle version de cette clé.

Récupérer une clé

Vous pouvez maintenant récupérer la clé créée précédemment à l'aide de la méthode [GetKeyAsync](#).

C#

```
var key = await client.GetKeyAsync("myKey");
```

Supprimer une clé

Pour finir, nous allons supprimer et purger la clé de votre coffre de clés avec les méthodes [StartDeleteKeyAsync](#) et [PurgeDeletedKeyAsync](#).

C#

```

var operation = await client.StartDeleteKeyAsync("myKey");

// You only need to wait for completion if you want to purge or recover the
key.
await operation.WaitForCompletionAsync();

var key = operation.Value;
await client.PurgeDeletedKeyAsync("myKey");

```

Exemple de code

Modifiez l'application console .NET pour interagir avec le coffre de clés en effectuant les étapes suivantes :

- Remplacez le code dans *Program.cs* par le code suivant :

C#

```

using System;
using System.Threading.Tasks;
using Azure.Identity;
using Azure.Security.KeyVault.Keys;

namespace key_vault_console_app
{
    class Program
    {
        static async Task Main(string[] args)
        {
            const string keyName = "myKey";
            var keyVaultName =
Environment.GetEnvironmentVariable("KEY_VAULT_NAME");
            var kvUri = $"https://{{keyVaultName}}.vault.azure.net";

            var client = new KeyClient(new Uri(kvUri), new
DefaultAzureCredential());

            Console.WriteLine($"Creating a key in {{keyVaultName}} called
'{keyName}' ...");
            var createdKey = await client.CreateKeyAsync(keyName,
KeyType.Rsa);
            Console.WriteLine("done.");

            Console.WriteLine($"Retrieving your key from
{{keyVaultName}}.");
            var key = await client.GetKeyAsync(keyName);
            Console.WriteLine($"Your key version is
'{key.Value.Properties.Version}'.");

            Console.WriteLine($"Deleting your key from {{keyVaultName}}

```

```
...");  
        var deleteOperation = await  
client.StartDeleteKeyAsync(keyName);  
        // You only need to wait for completion if you want to  
purge or recover the key.  
        await deleteOperation.WaitForCompletionAsync();  
Console.WriteLine("done.");  
  
Console.Write($"Purging your key from {keyVaultName} ...");  
await client.PurgeDeletedKeyAsync(keyName);  
Console.WriteLine(" done.");  
    }  
}  
}
```

Tester et vérifier

1. Exécutez la commande suivante pour générer le projet :

```
CLI .NET
```

```
dotnet build
```

2. Exécutez la commande suivante pour exécuter l'application.

```
CLI .NET
```

```
dotnet run
```

3. Quand vous y êtes invité, entrez une valeur de secret. Par exemple, mySecretPassword.

Une variante de la sortie suivante apparaît :

```
Console
```

```
Creating a key in mykeyvault called 'myKey' ... done.  
Retrieving your key from mykeyvault.  
Your key version is '8532359bcfed24e4bb2525f2d2050738a'.  
Deleting your key from jl-kv ... done  
Purging your key from <your-unique-keyvault-name> ... done.
```

Étapes suivantes

Dans ce guide de démarrage rapide, vous avez créé un coffre de clés, stocké une clé et récupéré cette clé.

Pour plus d'informations sur Key Vault et comment l'intégrer à vos applications, consultez les articles suivants :

- Lire la [vue d'ensemble Azure Key Vault](#)
- Lire la [Vue d'ensemble des clés](#)
- Consultez un [Tutoriel sur l'accès à Key Vault depuis une application App Service](#)
- Consultez un [Tutoriel sur l'accès à Key Vault depuis une machine virtuelle](#)
- Consulter le [Guide du développeur Azure Key Vault](#)
- Passer en revue la [Vue d'ensemble de la sécurité de Key Vault](#)

Démarrage rapide : Bibliothèque de client de secrets Azure Key Vault pour .NET

Article • 25/03/2023

Bien démarrer avec la bibliothèque de client de secrets Azure Key Vault pour .NET. [Azure Key Vault](#) est un service cloud qui fournit un magasin de secrets sécurisé. Vous pouvez stocker des clés, des mots de passe, des certificats et d'autres secrets en toute sécurité. Vous pouvez créer et gérer des coffres de clés Azure grâce au portail Azure. Dans ce guide de démarrage rapide, vous allez découvrir comment créer, récupérer et supprimer des secrets dans un coffre de clés Azure à l'aide de la bibliothèque de client .NET.

Ressources de la bibliothèque de client Key Vault :

[Documentation de référence sur l'API](#) | [Code source de la bibliothèque](#) ↗ | [Package \(NuGet\)](#) ↗

Pour plus d'informations sur Key Vault et les secrets, consultez :

- [Vue d'ensemble du coffre de clés](#)
- [Vue d'ensemble des secrets.](#)

Prérequis

- Un abonnement Azure : [créez-en un gratuitement](#) ↗
- [SDK .NET 6 ou ultérieur](#) ↗
- [Azure CLI](#) ou [Azure PowerShell](#)
- Un coffre de clés : vous pouvez en créer un en utilisant le [portail Azure](#), [Azure CLI](#) ou [Azure PowerShell](#)

Ce guide de démarrage rapide utilise `dotnet` et Azure CLI ou Azure PowerShell.

Programme d'installation

Azure CLI

Ce guide de démarrage rapide utilise la bibliothèque Azure Identity avec Azure CLI pour authentifier l'utilisateur auprès des services Azure. Les développeurs peuvent également utiliser Visual Studio ou Visual Studio Code pour authentifier leurs

appels. Pour plus d'informations, consultez [Authentifier le client avec la bibliothèque de client Azure Identity](#).

Connexion à Azure

1. Exécutez la commande `az login`.

```
Azure CLI
```

```
az login
```

Si l'interface CLI peut ouvrir votre navigateur par défaut, elle le fait et charge une page de connexion Azure par la même occasion.

Sinon, ouvrez une page de navigateur à l'adresse <https://aka.ms/devicelogin> et entrez le code d'autorisation affiché dans votre terminal.

2. Dans le navigateur, connectez-vous avec les informations d'identification de votre compte.

Accorder l'accès à votre coffre de clés

Créez une stratégie d'accès pour votre coffre de clés, qui accorde des autorisations de secret à votre compte d'utilisateur.

```
Azure CLI
```

```
az keyvault set-policy --name <YourKeyVaultName> --upn user@domain.com --secret-permissions delete get list set purge
```

Créer une application console .NET

1. Dans un shell de commandes, exécutez la commande suivante pour créer un projet nommé `key-vault-console-app` :

```
CLI .NET
```

```
dotnet new console --name key-vault-console-app
```

2. Accédez au répertoire `key-vault-console-app` nouvellement créé, puis exécutez la commande suivante pour générer le projet :

CLI .NET

```
dotnet build
```

La sortie de génération ne doit contenir aucun avertissement ni erreur.

Console

```
Build succeeded.  
0 Warning(s)  
0 Error(s)
```

Installer les packages

Depuis le shell de commandes, installez la bibliothèque de client de secrets Azure Key Vault pour .NET :

CLI .NET

```
dotnet add package Azure.Security.KeyVault.Secrets
```

Pour ce guide de démarrage rapide, vous devez également installer la bibliothèque de client Azure Identity :

CLI .NET

```
dotnet add package Azure.Identity
```

Définir des variables d'environnement

Cette application utilise le nom de coffre de clés en tant que variable d'environnement appelée `KEY_VAULT_NAME`.

Windows

Invite de commandes Windows

```
set KEY_VAULT_NAME=<your-key-vault-name>
```

Windows PowerShell

PowerShell

```
$Env:KEY_VAULT_NAME="your-key-vault-name"
```

macOS ou Linux

Bash

```
export KEY_VAULT_NAME=<your-key-vault-name>
```

Modèle objet

La bibliothèque de client de secrets Azure Key Vault pour .NET vous permet de gérer des secrets. La section [Exemples de code](#) montre comment créer un client, et définir, récupérer et supprimer un secret.

Exemples de code

Ajouter des directives

Ajoutez les directives suivantes en haut de *Program.cs* :

C#

```
using System;
using Azure.Identity;
using Azure.Security.KeyVault.Secrets;
```

Authentifier et créer un client

Les requêtes d'application vers les Services Azure doivent être autorisées. L'utilisation de la classe [DefaultAzureCredential](#) fournie par la [bibliothèque de client Azure Identity](#) est l'approche recommandée pour implémenter des connexions sans mot de passe aux services Azure dans votre code. `DefaultAzureCredential` prend en charge plusieurs méthodes d'authentification et détermine quelle méthode doit être utilisée au moment de l'exécution. Cette approche permet à votre application d'utiliser différentes méthodes d'authentification dans différents environnements (local ou production) sans implémenter de code spécifique à l'environnement.

Dans ce guide de démarrage rapide, `DefaultAzureCredential` s'authentifie auprès du coffre de clés à l'aide des informations d'identification de l'utilisateur de développement

local connecté à Azure CLI. Quand l'application est déployée sur Azure, le même code `DefaultAzureCredential` peut découvrir et utiliser automatiquement une identité managée affectée à un service d'application, une machine virtuelle ou d'autres services. Pour plus d'informations, consultez [Vue d'ensemble des identités managées](#).

Dans cet exemple, le nom de votre coffre de clés est étendu à l'URI du coffre de clés, au format `https://<your-key-vault-name>.vault.azure.net`. Pour plus d'informations sur l'authentification auprès du coffre de clés, consultez le [Guide du développeur](#).

```
C#
```

```
string keyVaultName = Environment.GetEnvironmentVariable("KEY_VAULT_NAME");
var kvUri = "https://" + keyVaultName + ".vault.azure.net";

var client = new SecretClient(new Uri(kvUri), new DefaultAzureCredential());
```

Enregistrer un secret

Maintenant que l'application console est authentifiée, ajoutez un secret au coffre de clés. Pour cette tâche, utilisez la méthode `SetSecretAsync`. Le premier paramètre de la méthode accepte un nom pour le secret, « mySecret » dans cet exemple.

```
C#
```

```
await client.SetSecretAsync(secretName, secretValue);
```

ⓘ Notes

Si le nom du secret existe, le code crée une nouvelle version de ce secret.

Récupérer un secret

Vous pouvez désormais récupérer la valeur définie avec la méthode `GetSecretAsync`.

```
C#
```

```
var secret = await client.GetSecretAsync(secretName);
```

Votre secret est désormais enregistré en tant que `secret.Value`.

Supprimer un secret

Pour terminer, nous allons supprimer le secret de votre coffre de clés avec les méthodes [StartDeleteSecretAsync](#) et [PurgeDeletedSecretAsync](#).

C#

```
var operation = await client.StartDeleteSecretAsync("mySecret");
// You only need to wait for completion if you want to purge or recover the
key.
await operation.WaitForCompletionAsync();

await client.PurgeDeletedSecretAsync("mySecret");
```

Exemple de code

Modifiez l'application console .NET pour interagir avec le coffre de clés en effectuant les étapes suivantes :

1. Remplacez le code dans *Program.cs* par le code suivant :

C#

```
using System;
using System.Threading.Tasks;
using Azure.Identity;
using Azure.Security.KeyVault.Secrets;

namespace key_vault_console_app
{
    class Program
    {
        static async Task Main(string[] args)
        {
            const string secretName = "mySecret";
            var keyVaultName =
                Environment.GetEnvironmentVariable("KEY_VAULT_NAME");
            var kvUri = $"https://{{keyVaultName}}.vault.azure.net";

            var client = new SecretClient(new Uri(kvUri), new
                DefaultAzureCredential());

            Console.Write("Input the value of your secret > ");
            var secretValue = Console.ReadLine();

            Console.WriteLine($"Creating a secret in {{keyVaultName}} called
'{secretName}' with the value '{secretValue}' ...");
            await client.SetSecretAsync(secretName, secretValue);
            Console.WriteLine(" done.");

            Console.WriteLine("Forgetting your secret.");
            secretValue = string.Empty;
```

```

        Console.WriteLine($"Your secret is '{secretValue}'.");
    }

    Console.WriteLine($"Retrieving your secret from {keyVaultName}.");
    var secret = await client.GetSecretAsync(secretName);
    Console.WriteLine($"Your secret is
'{secret.Value.Value}'.");

    Console.Write($"Deleting your secret from {keyVaultName}
...");
    DeleteSecretOperation operation = await
client.StartDeleteSecretAsync(secretName);
    // You only need to wait for completion if you want to
purge or recover the secret.
    await operation.WaitForCompletionAsync();
    Console.WriteLine(" done.");

    Console.Write($"Purging your secret from {keyVaultName}
...");
    await client.PurgeDeletedSecretAsync(secretName);
    Console.WriteLine(" done.");
}

}
}

```

Tester et vérifier

1. Exécutez la commande suivante pour exécuter l'application.

CLI .NET

`dotnet run`

2. Quand vous y êtes invité, entrez une valeur de secret. Par exemple, `mySecretPassword`.

Une variante de la sortie suivante apparaît :

Console

```

Input the value of your secret > mySecretPassword
Creating a secret in <your-unique-keyvault-name> called 'mySecret' with the
value 'mySecretPassword' ... done.
Forgetting your secret.
Your secret is ''.
Retrieving your secret from <your-unique-keyvault-name>.
Your secret is 'mySecretPassword'.
Deleting your secret from <your-unique-keyvault-name> ... done.
Purging your secret from <your-unique-keyvault-name> ... done.

```

Étapes suivantes

Pour plus d'informations sur Key Vault et comment l'intégrer à vos applications, consultez les articles suivants :

- Lire la [vue d'ensemble Azure Key Vault](#)
- Consultez un [Tutoriel sur l'accès à Key Vault depuis une application App Service](#)
- Consultez un [Tutoriel sur l'accès à Key Vault depuis une machine virtuelle](#)
- Consulter le [Guide du développeur Azure Key Vault](#)
- Passer en revue la [Vue d'ensemble de la sécurité de Key Vault](#)

Démarrage rapide : Envoyer et recevoir des messages à destination et en provenance d'une file d'attente Azure Service Bus (.NET)

Article • 05/12/2023

Dans ce guide de démarrage rapide, vous allez effectuer les étapes suivantes :

1. Créer un espace de noms Service Bus à l'aide du Portail Azure.
2. Créer une file d'attente Service Bus à l'aide du portail Azure.
3. Écrivez une application de console .NET pour envoyer un ensemble de messages à la file d'attente.
4. Écrivez une application de console .NET pour recevoir les messages de la file d'attente.

⚠ Notes

Ce guide de démarrage rapide fournit des instructions pas à pas pour implémenter un scénario simple qui consiste à envoyer un lot de messages à une file d'attente Service Bus, puis à les recevoir. Pour obtenir une vue d'ensemble de la bibliothèque de client .NET, consultez [bibliothèque de client Azure Service Bus pour .NET](#). Pour plus d'échantillons, consultez [Exemples relatifs à Service Bus .NET sur GitHub](#).

Prérequis

Si vous débutez avec le service, consultez [Vue d'ensemble de Service Bus](#) avant de suivre ce démarrage rapide.

- **Abonnement Azure.** Pour utiliser des services Azure, dont Azure Service Bus, vous avez besoin d'un abonnement. Si vous n'avez pas de compte Azure existant, vous pouvez demander un [essai gratuit](#).
- **Visual Studio 2022.** L'exemple d'application utilise les nouvelles fonctionnalités introduites dans C# 10. Vous pouvez toujours utiliser la bibliothèque cliente Service Bus avec les versions antérieures du langage C#, mais la syntaxe peut

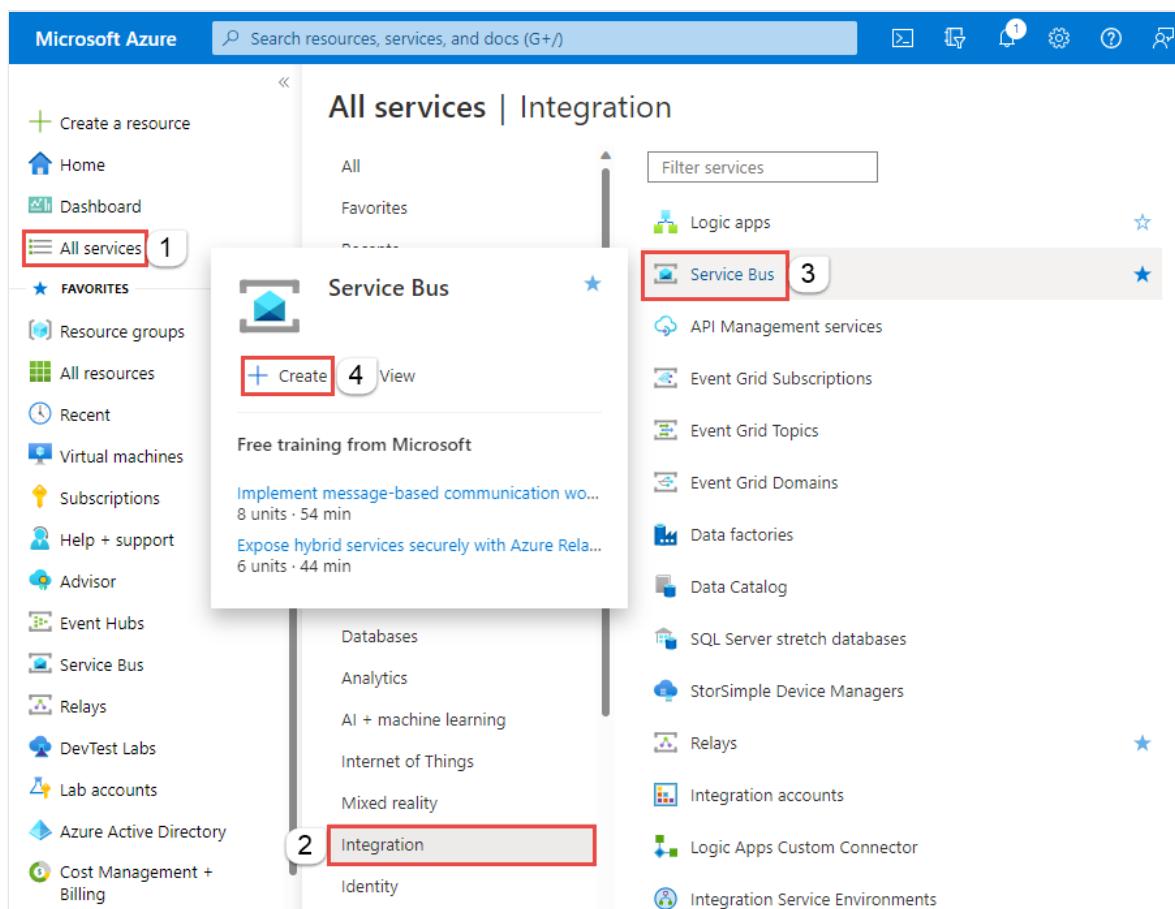
varier. Pour utiliser la dernière syntaxe, nous vous recommandons d'installer .NET 6.0 ou une version supérieure et de définir la version du langage sur `latest`. Si vous utilisez Visual Studio, les versions antérieures à Visual Studio 2022 ne sont pas compatibles avec les outils nécessaires à la génération de projets C# 10.

Créer un espace de noms dans le Portail Azure

Pour commencer à utiliser des entités de messagerie Service Bus dans Azure, vous devez d'abord créer un espace de noms avec un nom unique dans Azure. Un espace de noms fournit un conteneur d'étendue pour les ressources du Service Bus (files d'attente, thèmes, etc.) au sein de votre application.

Pour créer un espace de noms :

1. Connectez-vous au [portail Azure](#).
2. Dans le volet de navigation gauche du portail, sélectionnez **Tous les services**, puis **Intégration** dans la liste des catégories, pointez la souris sur **Service Bus**, puis sélectionnez le bouton **+** sur la vignette Service Bus.



3. Dans l'étiquette **De base** de la page **Créer un espace de noms**, suivez ces étapes :

- a. Pour l'option **Abonnement**, choisissez un abonnement Azure dans lequel créer l'espace de noms.
- b. Pour l'option **Groupe de ressources**, choisissez un groupe de ressources existant dans lequel l'espace de noms sera utilisé, ou créez-en un nouveau.
- c. Entrez un **nom pour l'espace de noms**. Le nom de l'espace de noms doit respecter les conventions de nommage suivantes :
 - Le nom doit être unique dans tout Azure. Le système vérifie immédiatement si le nom est disponible.
 - Le nom doit inclure entre 6 et 50 caractères.
 - Le nom ne peut contenir que des lettres, des chiffres et des traits d'union (« - »).
 - Le nom doit commencer par une lettre, et se terminer par une lettre ou un chiffre.
 - Le nom ne se termine ni par « -sb » ni par « -mgmt ».
- d. Pour l'option **Emplacement**, choisissez la région dans laquelle héberger votre espace de noms.

- e. Pour le **Niveau tarifaire**, sélectionnez le SKU (De base, Standard ou Premium) destiné à l'espace de noms. Pour ce guide de démarrage rapide, sélectionnez **Standard**.

 **Important**

Si vous voulez utiliser des **rubriques et des abonnements**, choisissez Standard ou Premium. Les rubriques/abonnements ne sont pas pris en charge dans le niveau tarifaire De base.

Si vous avez sélectionné le SKU **Premium**, précisez le **nombre d'unité de messagerie**. Le niveau Premium isole les ressources au niveau du processeur et de la mémoire, ce qui permet d'exécuter chaque charge de travail de manière isolée. Ce conteneur de ressources est appelé unité de messagerie. Un espace de noms Premium a au moins une unité de messagerie. Vous pouvez sélectionner 1, 2, 4, 8 ou 16 unités de messagerie pour chaque espace de noms Service Bus Premium. Pour plus d'informations, consultez [Messagerie Service Bus Premium](#).

- f. Au bas de la page, sélectionnez **Examiner et créer**.

Create namespace ...

Service Bus

Basics Advanced Networking Tags Review + create

Project Details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * Visual Studio Enterprise Subscription

Resource group * (New) spsbusrg [Create new](#)

Instance Details

Enter required settings for this namespace.

Namespace name * contosoordersns [.servicebus.windows.net](#)

Location * East US

Pricing tier * Standard [Browse the available plans and their features](#)

Review + create < Previous Next: Advanced >

g. Dans la page **Vérifier + créer**, passez en revue les paramètres, puis sélectionnez **Créer**.

4. Une fois le déploiement de la ressource réussi, sélectionnez **Accéder à la ressource** dans la page de déploiement.

contosoordersns | Overview

Deployment

Search

Delete Cancel Redeploy Download Refresh

Overview

Your deployment is complete

Deployment name: contosoordersns
Subscription: Visual Studio Enterprise Subscription
Resource group: spsbusrg

Start time: 10/20/2022, 4:45:03 PM
Correlation ID: a453ace1-bab9-4c4a-81ad-a1c5366460ea

Inputs
Outputs
Template

Deployment details
Next steps

Go to resource

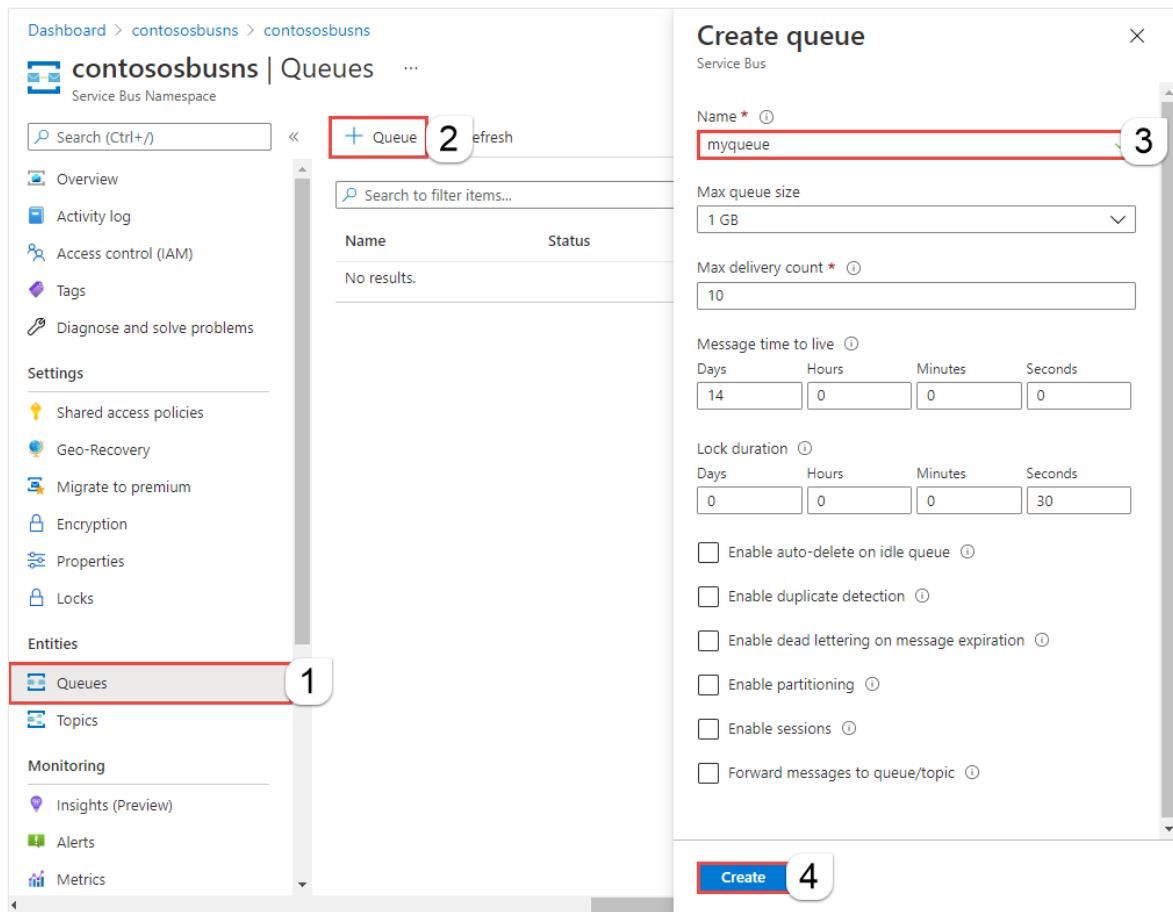
Give feedback
Tell us about your experience with deployment

5. Vous voyez la page d'accueil de votre espace de noms Service Bus.

The screenshot shows the Azure Service Bus Namespace overview page for the namespace `spsbusns1028`. The left sidebar contains navigation links for Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Settings (Shared access policies, Geo-Recovery, Migrate to premium, Encryption, Configuration, Properties, Locks), Entities (Queues, Topics), Monitoring (Insights (Preview), Alerts, Metrics, Diagnostic settings, Logs), and JSON View. The main content area displays the namespace's details: Resource group ([move](#)) `spsbusrg`, Status Active, Location East US, Subscription ([move](#)) `Visual Studio Enterprise Subscription`, Subscription ID `00000000-0000-0000-0000-000000000000`, Tags ([edit](#)), and Click here to add tags. It also shows monitoring data for Requests and Messages over the last hour, with zero activity shown. A search bar at the top left and a refresh button are also visible.

Créer une file d'attente dans le portail Azure

1. Dans la page **Espace de noms Service Bus**, sélectionnez **Files d'attente** dans le menu de navigation de gauche.
2. Dans la page **Files d'attente**, sélectionnez **+ File d'attente** dans la barre d'outils.
3. Entrez un **nom** pour la file d'attente et laissez les valeurs par défaut des autres valeurs.
4. À présent, sélectionnez **Créer**.



ⓘ Important

Si vous débutez avec Azure, vous trouverez peut-être l'option **chaîne de connexion** plus facile à suivre. Sélectionnez l'onglet **chaîne de connexion** pour afficher des instructions sur l'utilisation d'une chaîne de connexion dans ce guide de démarrage rapide. Nous vous recommandons d'utiliser l'option **sans mot de passe** dans les environnements d'applications et de production réels.

Authentifier l'application sur Azure

Ce guide de démarrage pratique vous montre deux façons de vous connecter à Azure Service Bus : **sans mot de passe** et avec une **chaîne de connexion**.

La première option vous explique comment utiliser votre principal de sécurité dans Microsoft Entra ID et le contrôle d'accès en fonction du rôle (RBAC) pour vous connecter à un espace de noms Service Bus. Vous n'avez pas à vous soucier d'avoir une chaîne de connexion codée en dur dans votre code, dans un fichier config ni dans un stockage sécurisé comme Azure Key Vault.

La deuxième option consiste à se servir d'une chaîne de connexion pour se connecter à un espace de noms Service Bus. Si vous débutez avec Azure, vous trouverez peut-être

l'option chaîne de connexion plus facile à suivre. Nous vous recommandons d'utiliser l'option sans mot de passe dans les applications réelles et les environnements de production. Pour plus d'informations, consultez [Authentification et autorisation](#). Pour en savoir plus sur l'authentification sans mot de passe, reportez-vous à la [page de présentation](#).

Sans mot de passe (recommandé)

Attribuer des rôles à votre utilisateur Microsoft Entra

Lors du développement localement, assurez-vous que le compte d'utilisateur qui se connecte à Azure Service Bus dispose des autorisations appropriées. Vous aurez besoin du rôle [Propriétaire de données Azure Service Bus](#) pour envoyer et recevoir des messages. Pour vous attribuer ce rôle, vous aurez besoin du rôle Administrateur de l'accès utilisateur ou d'un autre rôle qui inclut l'action

`Microsoft.Authorization/roleAssignments/write`. Vous pouvez attribuer des rôles RBAC Azure à un utilisateur à l'aide du Portail Azure, Azure CLI ou Azure PowerShell. Découvrez les étendues disponibles pour les attributions de rôles dans la page [vue d'ensemble de l'étendue](#).

L'exemple suivant attribue le rôle `Azure Service Bus Data Owner` à votre compte d'utilisateur, qui fournit un accès complet aux ressources Azure Service Bus. Dans un scénario réel, suivez le [principe des priviléges minimum](#) pour accorder aux utilisateurs uniquement les autorisations minimales nécessaires à un environnement de production plus sécurisé.

Rôles Azure intégrés pour Azure Service Bus

Pour Azure Service Bus, la gestion des espaces de noms et de toutes les ressources associées via le Portail Azure et l'API de gestion des ressources Azure est déjà protégée à l'aide du modèle Azure RBAC. Azure fournit les rôles Azure intégrés ci-dessous pour autoriser l'accès à un espace de noms Service Bus :

- [Propriétaire de données Azure Service Bus](#) : ce rôle permet l'accès aux données de l'espace de noms Service Bus et de ses entités (files d'attente, rubriques, abonnements et filtres). Un membre de ce rôle peut envoyer et recevoir des messages à partir de files d'attente ou de rubriques et d'abonnements.
- [Expéditeur de données Azure Service Bus](#) : utilisez ce rôle pour autoriser l'accès en envoi à l'espace de noms Service Bus et à ses entités.

- [Récepteur de données Azure Service Bus](#) : utilisez ce rôle pour autoriser l'accès en réception à l'espace de noms Service Bus et à ses entités.

Si vous souhaitez créer un rôle personnalisé, consultez [Droits requis pour les opérations Service Bus](#).

Ajouter un utilisateur Microsoft Entra au rôle Propriétaire Azure Service Bus

Ajoutez votre nom d'utilisateur Microsoft Entra au rôle **Propriétaire de données Azure Service Bus** au niveau de l'espace de noms Service Bus. Il permet à une application exécutée dans le contexte de votre compte d'utilisateur d'envoyer des messages à une file d'attente ou à une rubrique et d'en recevoir auprès d'une file d'attente ou de l'abonnement d'une rubrique.

Important

Dans la plupart des cas, la propagation de l'attribution de rôle dans Azure peut prendre une ou deux minutes. Dans de rares cas, cela peut prendre jusqu'à **huit minutes**. Si vous recevez des erreurs d'authentification lorsque vous exécutez votre code pour la première fois, patientez quelques instants et réessayez.

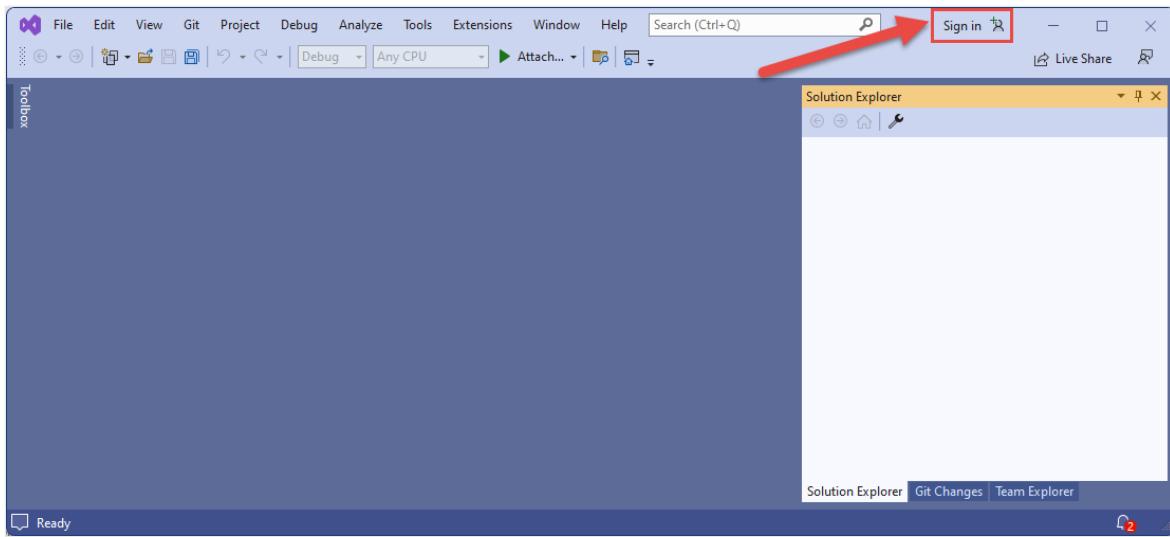
1. Si la page Espace de noms Service Bus n'est pas ouverte sur le Portail Azure, recherchez votre espace de noms Service Bus à l'aide de la barre de recherche principale ou du volet de navigation de gauche.
2. Dans la page vue d'ensemble, sélectionnez **Contrôle d'accès (IAM)** dans le menu de gauche.
3. Sur la page **Contrôle d'accès (IAM)**, sélectionnez l'onglet **Attributions de rôles**.
4. Sélectionnez **+ Ajouter** dans le menu supérieur, puis **Ajouter une attribution de rôle** dans le menu déroulant résultant.

5. Utilisez la zone de recherche pour filtrer les résultats sur le rôle souhaité. Pour cet exemple, recherchez **Azure Service Bus Data Owner** et sélectionnez le résultat correspondant. Ensuite, choisissez **Suivant**.
6. Sous **Attribuer l'accès à**, sélectionnez **Utilisateur, groupe ou principal de service**, puis sélectionnez + **Sélectionner des membres**.
7. Dans la boîte de dialogue, recherchez votre nom d'utilisateur Microsoft Entra (généralement votre adresse e-mail *user@domain*), puis choisissez **Sélectionner** en bas de la boîte de dialogue.
8. Sélectionnez **Vérifier + affecter** pour accéder à la page finale, puis **Vérifier + attribuer** à nouveau pour terminer le processus.

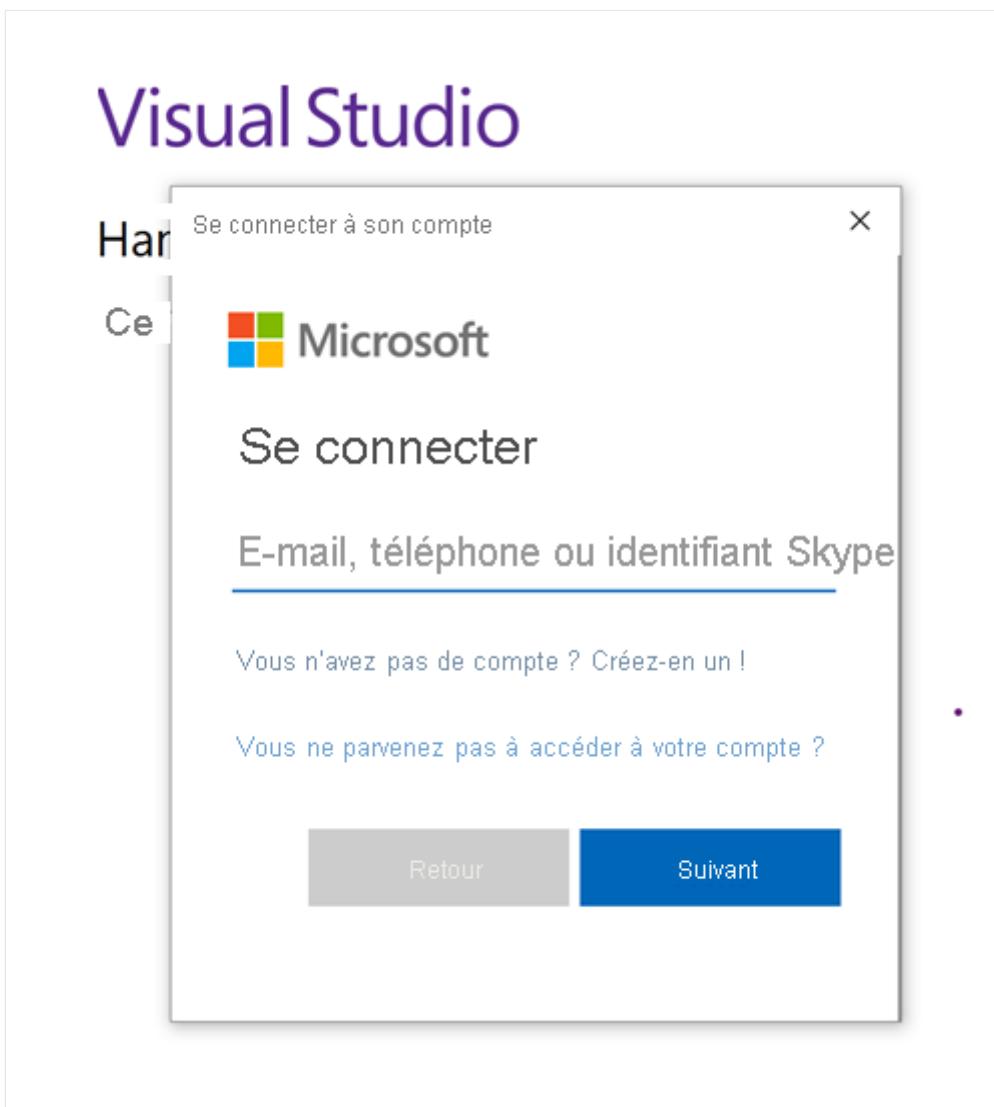
Lancer Visual Studio et vous connecter à Azure

Vous pouvez autoriser l'accès à l'espace de noms Service Bus en procédant comme suit :

1. Lancez Visual Studio. Si la fenêtre **Démarrage** s'affiche, sélectionnez le lien **Continuer sans code** dans le volet de droite.
2. Sélectionnez le bouton **Se connecter** en haut à droite de Visual Studio.



3. Connectez-vous à l'aide du compte Microsoft Entra auquel vous avez attribué un rôle précédemment.



Envoyer des messages à la file d'attente

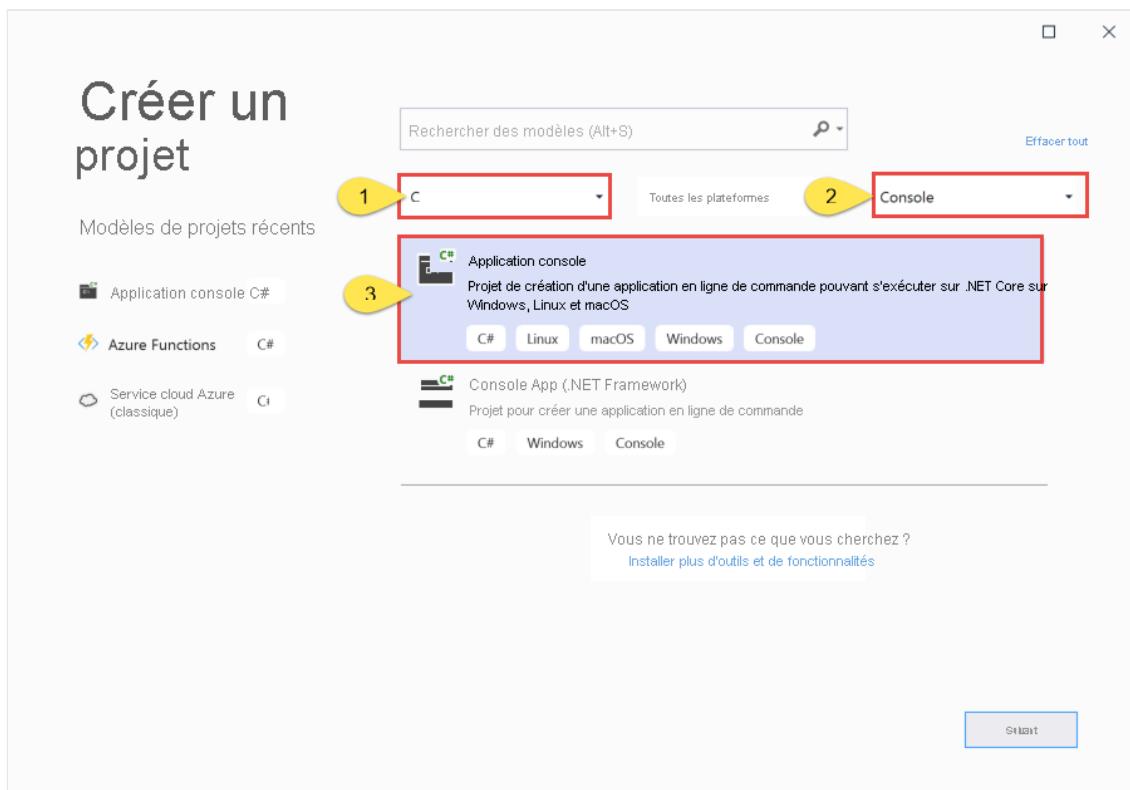
Cette section montre comment créer une application console .NET pour envoyer des messages à une file d'attente Service Bus.

ⓘ Notes

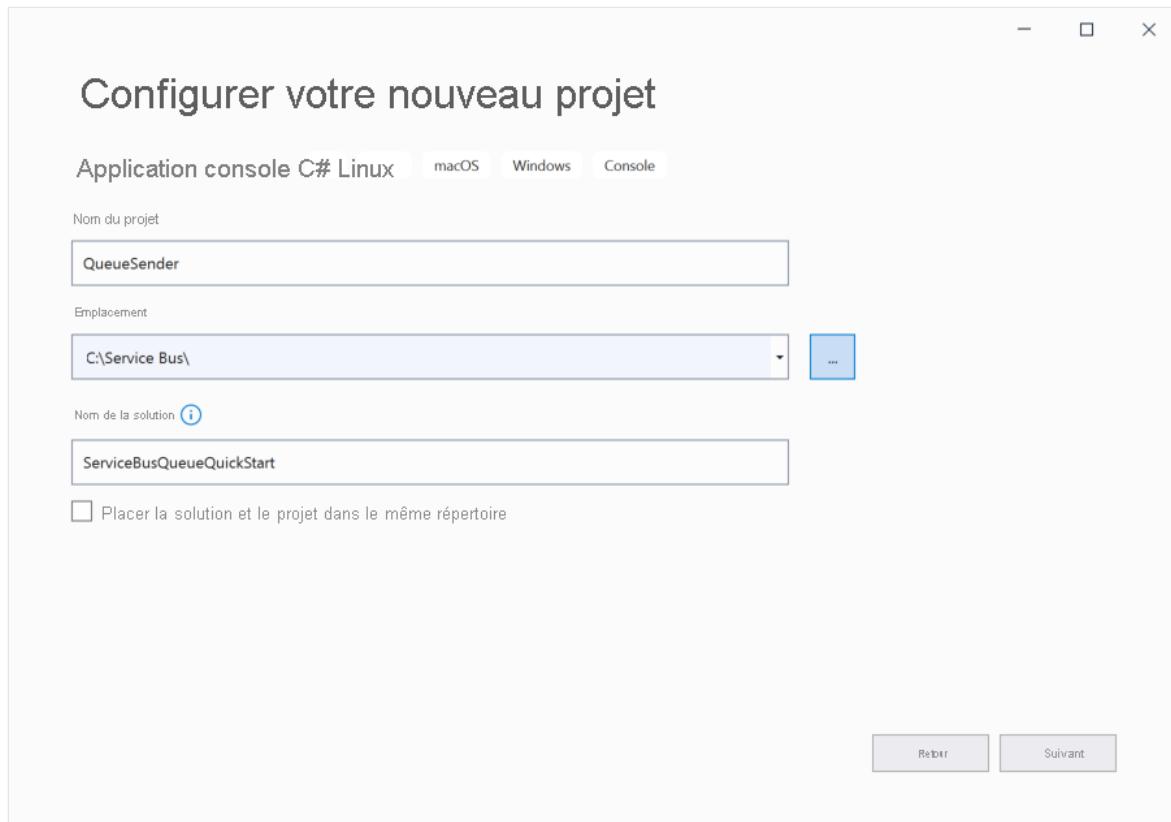
Ce guide de démarrage rapide fournit des instructions pas à pas pour implémenter un scénario simple qui consiste à envoyer un lot de messages à une file d'attente Service Bus, puis à les recevoir. Pour plus d'exemples sur d'autres scénarios et des scénarios avancés, voir [Exemples .NET Service Bus sur GitHub](#).

Création d'une application console

1. Dans Visual Studio, sélectionnez le menu **Fichier - > Nouveau - > Projet**.
2. Dans la boîte de dialogue **Créer un projet**, effectuez les étapes suivantes : Si vous ne voyez pas cette boîte de dialogue, sélectionnez **Fichier** dans le menu, sélectionnez **Nouveau**, puis **Projet**.
 - a. Sélectionnez **C#** en guise de langage de programmation.
 - b. Sélectionnez **Console** comme type de l'application.
 - c. Sélectionnez **Application console** dans la liste des résultats.
 - d. Ensuite, sélectionnez **Suivant**.



3. Entrez **QueueSender** pour le nom du projet, **ServiceBusQueueQuickStart** pour le nom de la solution, puis sélectionnez **Suivant**.



4. Dans la page **Informations supplémentaires**, sélectionnez **Créer** pour créer la solution et le projet.

Ajouter les paquets NuGet au projet

Sans mot de passe

1. Cliquez sur **Outils > Gestionnaire de package NuGet > Console du Gestionnaire de package** à partir du menu.
2. Exécutez la commande suivante pour installer le package NuGet **Azure.Messaging.ServiceBus**.

```
PowerShell
Install-Package Azure.Messaging.ServiceBus
```
3. Exécutez la commande suivante pour installer le package NuGet **Azure.Identity**.

```
PowerShell
```

```
Install-Package Azure.Identity
```

Ajouter du code pour envoyer des messages à la file d'attente

1. Remplacez le contenu de `Program.cs` par le code suivant. Les étapes importantes sont décrites dans la section suivante, avec des informations supplémentaires dans les commentaires de code.

Sans mot de passe

- Crée un objet `ServiceBusClient` à l'aide de l'objet `DefaultAzureCredential`.
`DefaultAzureCredential` détecte et utilise automatiquement les informations d'identification de votre connexion Visual Studio pour vous authentifier auprès d'Azure Service Bus.
- Utilise la méthode `CreateSender` sur l'objet `ServiceBusClient` pour créer un objet `ServiceBusSender` pour la file d'attente Service Bus spécifique.
- Crée un objet `ServiceBusMessageBatch` à l'aide de la méthode `ServiceBusSender.CreateMessageBatchAsync`.
- Ajoutez des messages au lot à l'aide de `ServiceBusMessageBatch.TryAddMessage`.
- Envoie le lot de messages à file d'attente Service Bus à l'aide de la méthode `ServiceBusSender.SendMessagesAsync`.

ⓘ Important

Mettez à jour les valeurs d'espace réservé (`<NAMESPACE-NAME>` et `<QUEUE-NAME>`) dans l'extrait de code avec les noms de votre espace de noms et de votre file d'attente Service Bus.

C#

```
using Azure.Messaging.ServiceBus;
using Azure.Identity;

// name of your Service Bus queue
// the client that owns the connection and can be used to create
// senders and receivers
ServiceBusClient client;
```

```
// the sender used to publish messages to the queue
ServiceBusSender sender;

// number of messages to be sent to the queue
const int numMessages = 3;

// The Service Bus client types are safe to cache and use as a
// singleton for the lifetime
// of the application, which is best practice when messages are
// being published or read
// regularly.
//
// Set the transport type to AmqpWebSockets so that the
ServiceBusClient uses the port 443.
// If you use the default AmqpTcp, ensure that ports 5671 and 5672
are open.
var clientOptions = new ServiceBusClientOptions
{
    TransportType = ServiceBusTransportType.AmqpWebSockets
};
//TODO: Replace the "<NAMESPACE-NAME>" and "<QUEUE-NAME>"
placeholders.
client = new ServiceBusClient(
    "<NAMESPACE-NAME>.servicebus.windows.net",
    new DefaultAzureCredential(),
    clientOptions);
sender = client.CreateSender("<QUEUE-NAME>");

// create a batch
using ServiceBusMessageBatch messageBatch = await
sender.CreateMessageBatchAsync();

for (int i = 1; i <= numMessages; i++)
{
    // try adding a message to the batch
    if (!messageBatch.TryAddMessage(new ServiceBusMessage($"Message
{i}")))
    {
        // if it is too large for the batch
        throw new Exception($"The message {i} is too large to fit
in the batch.");
    }
}

try
{
    // Use the producer client to send the batch of messages to the
    Service Bus queue
    await sender.SendMessagesAsync(messageBatch);
    Console.WriteLine($"A batch of {numMessages} messages has
been published to the queue.");
}
finally
{
```

```
// Calling DisposeAsync on client types is required to ensure  
// that network  
// resources and other unmanaged objects are properly cleaned  
up.  
    await sender.DisposeAsync();  
    await client.DisposeAsync();  
}  
  
Console.WriteLine("Press any key to end the application");  
Console.ReadKey();
```

2. Générez le projet et vérifiez qu'il ne présente pas d'erreurs.
3. Exécutez le programme et attendez le message de confirmation.

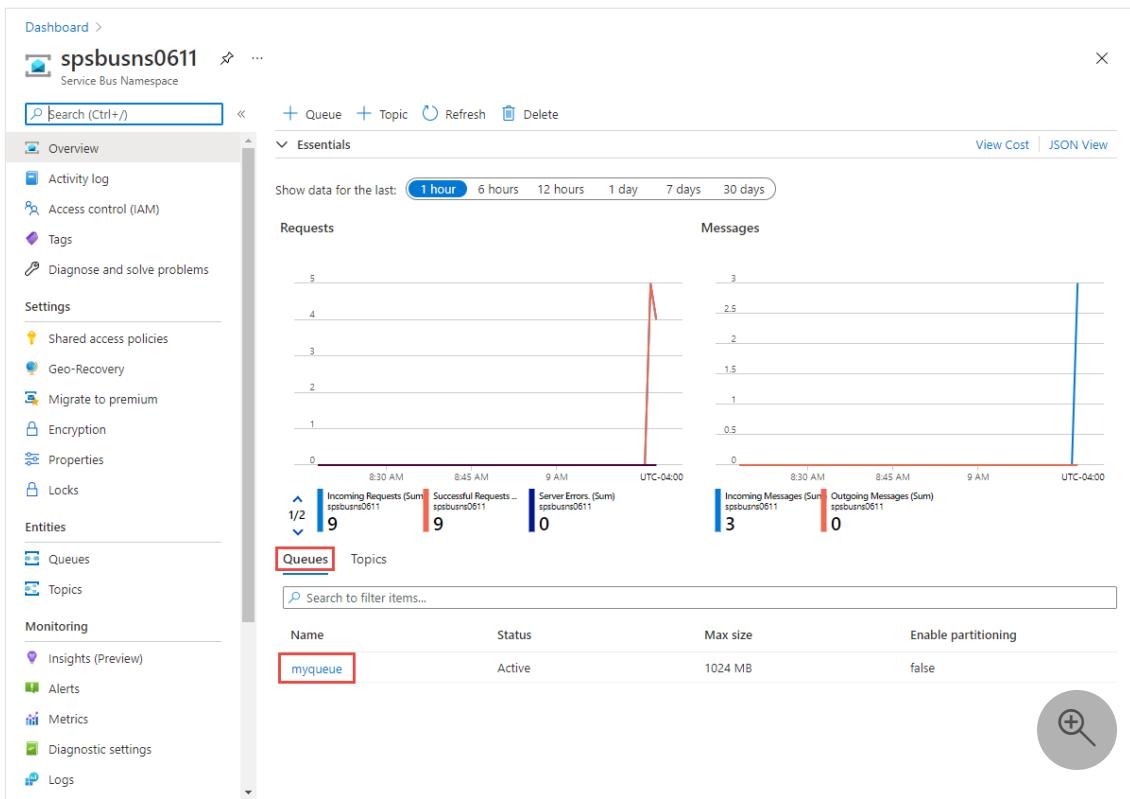
Bash

```
A batch of 3 messages has been published to the queue
```

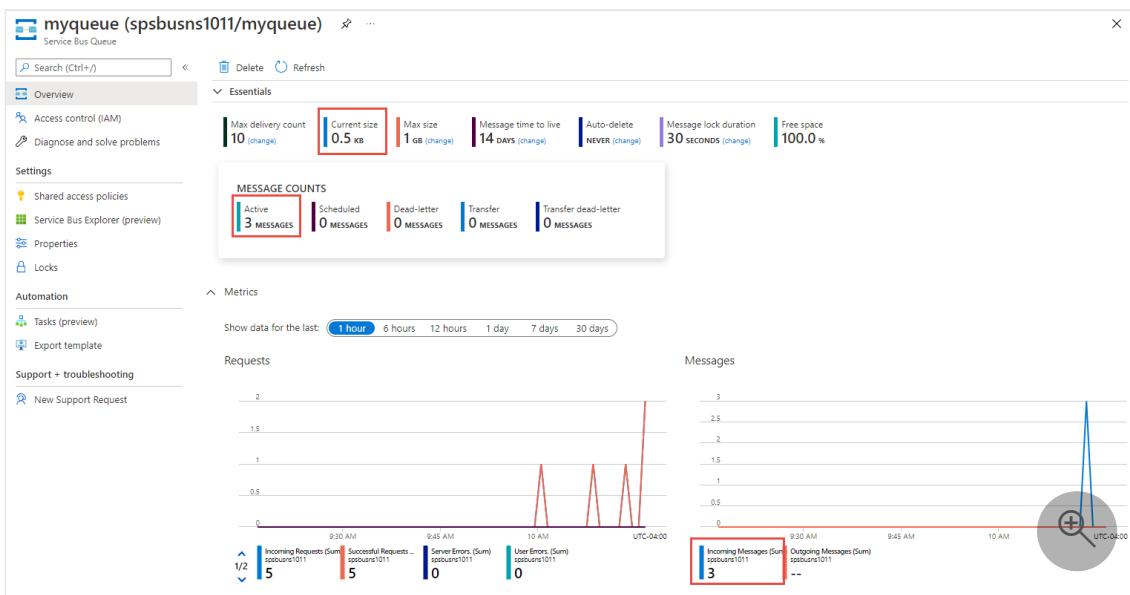
Important

Dans la plupart des cas, la propagation de l'attribution de rôle dans Azure peut prendre une ou deux minutes. Dans de rares cas, il peut prendre jusqu'à **huit minutes**. Si vous recevez des erreurs d'authentification lorsque vous exécutez votre code pour la première fois, patientez quelques instants et réessayez.

4. Dans le portail Azure, procédez comme suit :
 - a. Accédez à votre espace de noms Service Bus.
 - b. Dans la page **Vue d'ensemble**, sélectionnez la file d'attente dans le volet central inférieur.



c. Remarquez les valeurs figurant dans la section **Bases**.



Remarquez les valeurs suivantes :

- La valeur **Nombre de messages actifs** pour la file d'attente est à présent égale à 3. Chaque fois que vous exécutez cette application d'expéditeur sans récupérer les messages, cette valeur augmente de 3.
- La **taille actuelle** de la file d'attente augmente chaque fois que l'application y ajoute des messages.
- Dans le graphique **Messages** de la section inférieure **Métriques**, vous pouvez voir qu'il y a trois messages entrants pour la file d'attente.

Recevoir les messages de la file d'attente

Dans cette section, vous allez créer une application console .NET qui reçoit des messages de la file d'attente.

ⓘ Notes

Ce guide de démarrage rapide fournit des instructions pas à pas pour implémenter un scénario qui consiste à envoyer un lot de messages à une file d'attente Service Bus, puis à les recevoir. Pour plus d'exemples sur d'autres scénarios et des scénarios avancés, voir [Exemples .NET Service Bus sur GitHub](#).

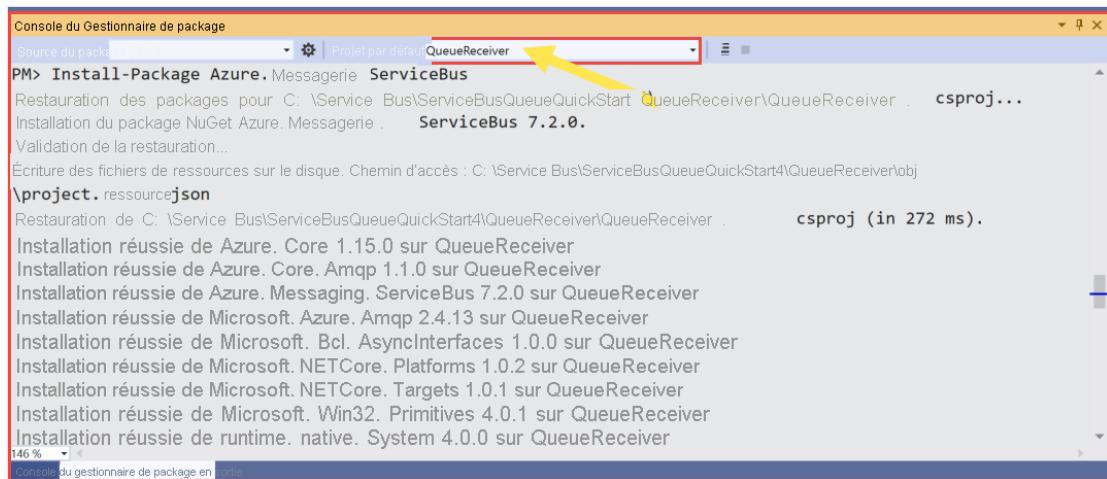
Créer un projet pour le récepteur

1. Dans la fenêtre Explorateur de solutions, cliquez avec le bouton droit sur la solution `ServiceBusQueueQuickStart`, pointez sur **Ajouter**, puis sélectionnez **Nouveau projet**.
2. Sélectionnez **Application console**, puis **Suivant**.
3. Entrez **QueueReceiver** pour **Nom du projet**, puis sélectionnez **Créer**.
4. Dans la fenêtre **Explorateur de solutions**, cliquez avec le bouton droit sur `QueueReceiver`, puis sélectionnez **Définir comme projet de démarrage**.

Ajouter les paquets NuGet au projet

Sans mot de passe

1. Cliquez sur **Outils > Gestionnaire de package NuGet > Console du Gestionnaire de package** à partir du menu.
2. Sélectionnez **QueueReceiver** pour **Projet par défaut**.



```
Console du Gestionnaire de package
Source du pack : Projet par défaut QueueReceiver
PM> Install-Package Azure.Messaging.ServiceBus
Restoration des packages pour C:\Service Bus\ServiceBusQueueQuickStart\QueueReceiver\QueueReceiver.csproj...
Installation du package NuGet Azure.Messaging.ServiceBus 7.2.0.
Validation de la restauration...
Écriture des fichiers de ressources sur le disque. Chemin d'accès : C:\Service Bus\ServiceBusQueueQuickStart\4\QueueReceiver\obj
\project.json
Restoration de C:\Service Bus\ServiceBusQueueQuickStart\4\QueueReceiver\QueueReceiver.csproj (in 272 ms).
Installation réussie de Azure.Core 1.15.0 sur QueueReceiver
Installation réussie de Azure.Core.Amqp 1.1.0 sur QueueReceiver
Installation réussie de Azure.Messaging.ServiceBus 7.2.0 sur QueueReceiver
Installation réussie de Microsoft.Azure.Amqp 2.4.13 sur QueueReceiver
Installation réussie de Microsoft.Bcl.AsyncInterfaces 1.0.0 sur QueueReceiver
Installation réussie de Microsoft.NETCore.Platforms 1.0.2 sur QueueReceiver
Installation réussie de Microsoft.NETCore.Targets 1.0.1 sur QueueReceiver
Installation réussie de Microsoft.Win32.Primitives 4.0.1 sur QueueReceiver
Installation réussie de runtime.native.System 4.0.0 sur QueueReceiver
146 %
```

3. Exécutez la commande suivante pour installer le package NuGet **Azure.Messaging.ServiceBus**.

```
PowerShell
Install-Package Azure.Messaging.ServiceBus
```

4. Exécutez la commande suivante pour installer le package NuGet **Azure.Identity**.

```
PowerShell
Install-Package Azure.Identity
```

Ajouter le code pour recevoir des messages de la file d'attente

Dans cette section, vous allez ajouter du code pour récupérer les messages de la file d'attente.

1. Dans la classe `Program`, ajoutez le code suivant :

```
Sans mot de passe
```

```
C#
using System.Threading.Tasks;
using Azure.Identity;
using Azure.Messaging.ServiceBus;

// the client that owns the connection and can be used to create
```

```
senders and receivers
ServiceBusClient client;

// the processor that reads and processes messages from the queue
ServiceBusProcessor processor;
```

2. Ajoutez les méthodes suivantes à la fin de la classe `Program`.

C#

```
// handle received messages
async Task MessageHandler(ProcessMessageEventArgs args)
{
    string body = args.Message.Body.ToString();
    Console.WriteLine($"Received: {body}");

    // complete the message. message is deleted from the queue.
    await args.CompleteMessageAsync(args.Message);
}

// handle any errors when receiving messages
Task ErrorHandler(ProcessErrorEventArgs args)
{
    Console.WriteLine(args.Exception.ToString());
    return Task.CompletedTask;
}
```

3. Ajoutez le code suivant à la fin de la classe `Program`. Les étapes importantes sont décrites dans la section suivante, avec des informations supplémentaires dans les commentaires de code.

Sans mot de passe

- Crée un objet `ServiceBusClient` à l'aide de l'objet `DefaultAzureCredential`.
`DefaultAzureCredential` découvre et utilise automatiquement les informations d'identification de votre connexion Visual Studio pour vous authentifier auprès d'Azure Service Bus.
- Appelle la méthode `CreateProcessor` sur l'objet `ServiceBusClient` pour créer un objet `ServiceBusProcessor` pour la file d'attente Service Bus spécifiée.
- Spécifie des gestionnaires pour les événements `ProcessMessageAsync` et `ProcessErrorAsync` de l'objet `ServiceBusProcessor`.
- Démarrer le traitement des messages en appelant `StartProcessingAsync` sur l'objet `ServiceBusProcessor`.

- Lorsque l'utilisateur appuie sur une clé pour terminer le traitement, il appelle `StopProcessingAsync` sur l' objet `ServiceBusProcessor`.

ⓘ Important

Mettez à jour les valeurs d'espace réservé (<NAMESPACE-NAME> et <QUEUE-NAME>) dans l'extrait de code avec les noms de votre espace de noms et de votre file d'attente Service Bus.

C#

```
// The Service Bus client types are safe to cache and use as a
// singleton for the lifetime
// of the application, which is best practice when messages are
// being published or read
// regularly.
//
// Set the transport type to AmqpWebSockets so that the
// ServiceBusClient uses port 443.
// If you use the default AmqpTcp, make sure that ports 5671 and
// 5672 are open.

// TODO: Replace the <NAMESPACE-NAME> placeholder
var clientOptions = new ServiceBusClientOptions()
{
    TransportType = ServiceBusTransportType.AmqpWebSockets
};
client = new ServiceBusClient(
    "<NAMESPACE-NAME>.servicebus.windows.net",
    new DefaultAzureCredential(),
    clientOptions);

// create a processor that we can use to process the messages
// TODO: Replace the <QUEUE-NAME> placeholder
processor = client.CreateProcessor("<QUEUE-NAME>", new
ServiceBusProcessorOptions());

try
{
    // add handler to process messages
    processor.ProcessMessageAsync += MessageHandler;

    // add handler to process any errors
    processor.ProcessErrorAsync += ErrorHandler;

    // start processing
    await processor.StartProcessingAsync();

    Console.WriteLine("Wait for a minute and then press any key to
end the processing");
}
```

```

Console.ReadKey();

// stop processing
Console.WriteLine("\nStopping the receiver...");
await processor.StopProcessingAsync();
Console.WriteLine("Stopped receiving messages");
}

finally
{
    // Calling DisposeAsync on client types is required to ensure
    // that network
    // resources and other unmanaged objects are properly cleaned
    up.
    await processor.DisposeAsync();
    await client.DisposeAsync();
}

```

4. La classe `Program` terminée doit correspondre au code suivant :

Sans mot de passe

C#

```

using System.Threading.Tasks;
using Azure.Messaging.ServiceBus;
using Azure.Identity;

// the client that owns the connection and can be used to create
// senders and receivers
ServiceBusClient client;

// the processor that reads and processes messages from the queue
ServiceBusProcessor processor;

// The Service Bus client types are safe to cache and use as a
// singleton for the lifetime
// of the application, which is best practice when messages are
// being published or read
// regularly.
//
// Set the transport type to AmqpWebSockets so that the
// ServiceBusClient uses port 443.
// If you use the default AmqpTcp, make sure that ports 5671 and
// 5672 are open.

// TODO: Replace the <NAMESPACE-NAME> and <QUEUE-NAME> placeholders
var clientOptions = new ServiceBusClientOptions()
{
    TransportType = ServiceBusTransportType.AmqpWebSockets
};
client = new ServiceBusClient("<NAMESPACE-"

```

```

NAME>.servicebus.windows.net",
    new DefaultAzureCredential(), clientOptions);

// create a processor that we can use to process the messages
// TODO: Replace the <QUEUE-NAME> placeholder
processor = client.CreateProcessor("<QUEUE-NAME>", new
ServiceBusProcessorOptions());

try
{
    // add handler to process messages
    processor.ProcessMessageAsync += MessageHandler;

    // add handler to process any errors
    processor.ProcessErrorAsync += ErrorHandler;

    // start processing
    await processor.StartProcessingAsync();

    Console.WriteLine("Wait for a minute and then press any key to
end the processing");
    Console.ReadKey();

    // stop processing
    Console.WriteLine("\nStopping the receiver...");
    await processor.StopProcessingAsync();
    Console.WriteLine("Stopped receiving messages");
}

finally
{
    // Calling DisposeAsync on client types is required to ensure
that network
    // resources and other unmanaged objects are properly cleaned
up.
    await processor.DisposeAsync();
    await client.DisposeAsync();
}

// handle received messages
async Task MessageHandler(ProcessMessageEventArgs args)
{
    string body = args.Message.Body.ToString();
    Console.WriteLine($"Received: {body}");

    // complete the message. message is deleted from the queue.
    await args.CompleteMessageAsync(args.Message);
}

// handle any errors when receiving messages
Task ErrorHandler(ProcessErrorEventArgs args)
{
    Console.WriteLine(args.Exception.ToString());
    return Task.CompletedTask;
}

```

- Générez le projet et vérifiez qu'il ne présente pas d'erreurs.
- Exécutez l'application réceptrice. Vous devriez voir les messages reçus. Appuyez sur n'importe quelle touche pour arrêter le récepteur et l'application.

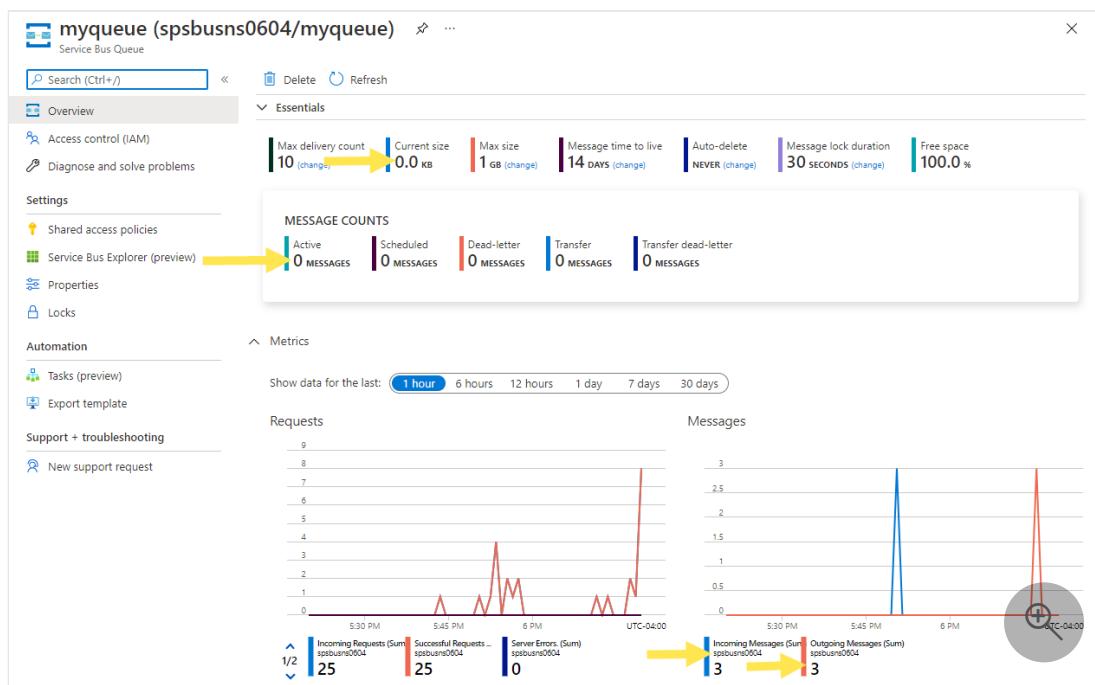
```
Console

Wait for a minute and then press any key to end the processing
Received: Message 1
Received: Message 2
Received: Message 3

Stopping the receiver...
Stopped receiving messages
```

- Vérifiez à nouveau le portail. Attendez quelques minutes et actualisez la page si vous ne voyez pas **0** pour **Nombre de messages actifs**.

- Les valeurs **Nombre de messages actifs** et **Taille actuelle** sont à présent égales à 0.
- Dans le graphique **Messages** de la section **Métriques** en bas, vous pouvez voir trois messages entrants et trois messages sortants pour la file d'attente.



Nettoyer les ressources

Accédez à votre espace de noms Service Bus dans le Portail Azure, puis sélectionnez Supprimer sur le Portail Azure pour supprimer l'espace de noms et la file d'attente qu'il

contient.

Voir aussi

Consultez la documentation et les exemples suivants :

- [Bibliothèque de client Azure Service Bus pour .NET – Fichier Lisez-moi ↗](#)
- [Exemples sur GitHub ↗](#)
- [Informations de référence sur l'API .NET](#)
- [Faire abstraction des aspects de l'infrastructure avec des frameworks de niveau supérieur comme NServiceBus](#)

Étapes suivantes

[Prise en main des rubriques et abonnements Azure Service Bus \(.NET\)](#)

Prise en main des rubriques et abonnements Azure Service Bus (.NET)

Article • 07/12/2023

Ce guide de démarrage rapide montre comment envoyer des messages à une rubrique Service Bus et recevoir des messages d'un abonnement à cette rubrique en utilisant la bibliothèque .NET [Azure.Messaging.ServiceBus](#).

Dans ce guide de démarrage rapide, vous allez effectuer les étapes suivantes :

1. Créer un espace de noms Service Bus à l'aide du Portail Azure.
2. Créer une rubrique Service Bus à l'aide du Portail Azure.
3. Créer un abonnement Service Bus vers cette rubrique à l'aide du Portail Azure.
4. Écrivez une application de console .NET pour envoyer un ensemble de messages à la rubrique.
5. Écrivez une application de console .NET pour recevoir ces messages de l'abonnement.

ⓘ Notes

Ce guide de démarrage rapide fournit des instructions pas à pas pour un scénario simple qui consiste à envoyer un lot de messages à une rubrique Service Bus et à recevoir ces messages à partir d'un abonnement de la rubrique. Pour plus d'exemples sur d'autres scénarios et des scénarios avancés, voir [Exemples .NET Service Bus sur GitHub](#).

- Ce guide de démarrage rapide vous montre deux façons de vous connecter à Azure Service Bus : avec une **chaîne de connexion** et **sans mot de passe**. La première option décrit comment utiliser une chaîne de connexion pour vous connecter à un espace de noms Service Bus. La seconde option vous explique comment utiliser votre principal de sécurité dans Microsoft Entra ID et le contrôle d'accès en fonction du rôle (RBAC) pour vous connecter à un espace de noms Service Bus. Vous n'avez pas à vous soucier d'avoir une chaîne de connexion codée en dur dans votre code, dans un fichier config ou dans un stockage sécurisé comme Azure Key Vault. Si vous débutez dans Azure, vous trouverez peut-être l'option de chaîne de connexion plus facile à suivre. Nous vous recommandons d'utiliser l'option sans mot de passe dans les

applications réelles et les environnements de production. Pour plus d'informations, consultez [Authentification et autorisation](#).

Prérequis

Si vous débutez avec le service, consultez [Vue d'ensemble de Service Bus](#) avant de suivre ce démarrage rapide.

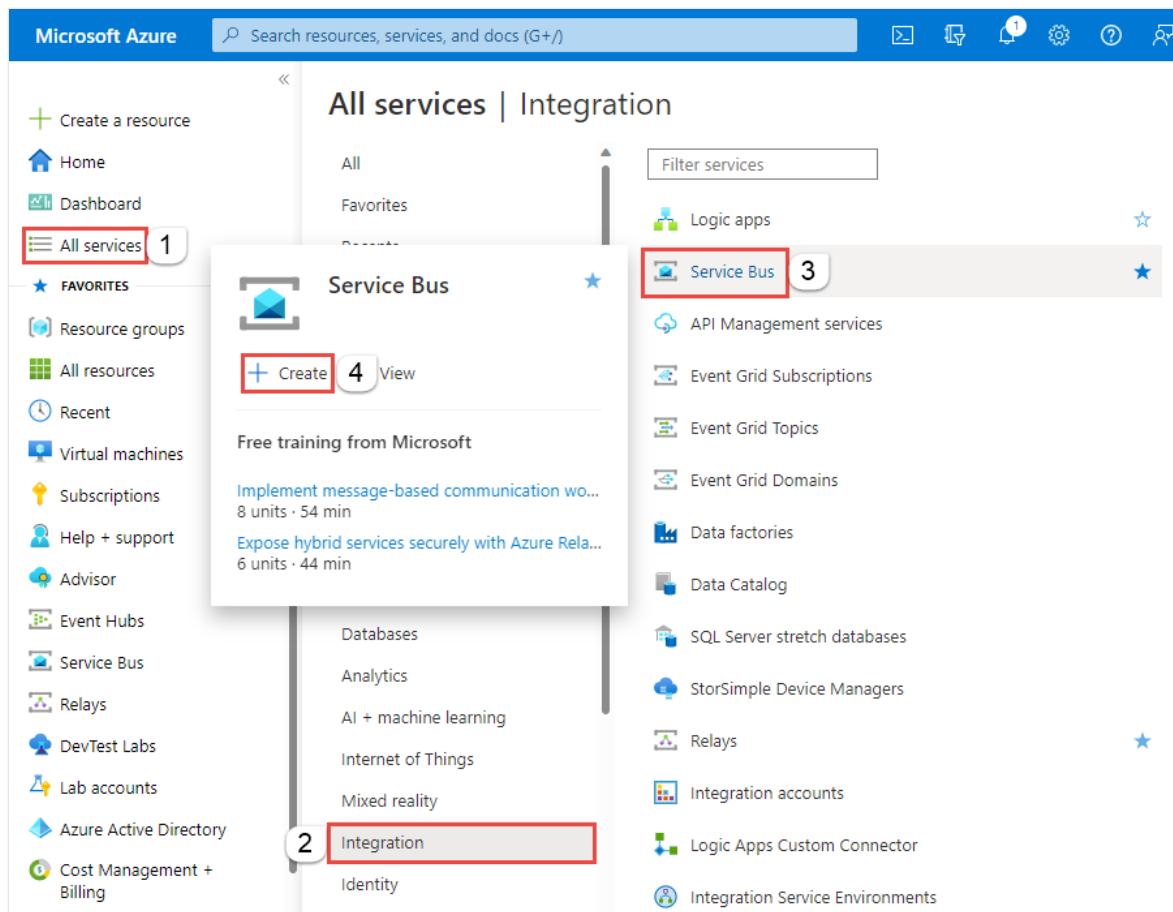
- **Abonnement Azure.** Pour utiliser des services Azure, dont Azure Service Bus, vous avez besoin d'un abonnement. Si vous n'avez pas de compte Azure existant, vous pouvez demander un [essai gratuit](#).
- **Visual Studio 2022.** L'exemple d'application utilise les nouvelles fonctionnalités introduites dans C# 10. Vous pouvez toujours utiliser la bibliothèque cliente Service Bus avec les versions antérieures du langage C#, mais la syntaxe peut varier. Pour utiliser la dernière syntaxe, nous vous recommandons d'installer .NET 6.0 ou une version supérieure et de définir la version du langage sur `latest`. Si vous utilisez Visual Studio, les versions antérieures à Visual Studio 2022 ne sont pas compatibles avec les outils nécessaires à la génération de projets C# 10.

Créer un espace de noms dans le Portail Azure

Pour commencer à utiliser des entités de messagerie Service Bus dans Azure, vous devez d'abord créer un espace de noms avec un nom unique dans Azure. Un espace de noms fournit un conteneur d'étendue pour les ressources du Service Bus (files d'attente, thèmes, etc.) au sein de votre application.

Pour créer un espace de noms :

1. Connectez-vous au [portail Azure](#).
2. Dans le volet de navigation gauche du portail, sélectionnez **Tous les services**, puis **Intégration** dans la liste des catégories, pointez la souris sur **Service Bus**, puis sélectionnez le bouton + sur la vignette Service Bus.



3. Dans l'étiquette **De base** de la page **Créer un espace de noms**, suivez ces étapes :

- Pour l'option **Abonnement**, choisissez un abonnement Azure dans lequel créer l'espace de noms.
- Pour l'option **Groupe de ressources**, choisissez un groupe de ressources existant dans lequel l'espace de noms sera utilisé, ou créez-en un nouveau.
- Entrez un **nom pour l'espace de noms**. Le nom de l'espace de noms doit respecter les conventions de nommage suivantes :
 - Le nom doit être unique dans tout Azure. Le système vérifie immédiatement si le nom est disponible.
 - Le nom doit inclure entre 6 et 50 caractères.
 - Le nom ne peut contenir que des lettres, des chiffres et des traits d'union (« - »).
 - Le nom doit commencer par une lettre, et se terminer par une lettre ou un chiffre.
 - Le nom ne se termine ni par « -sb » ni par « -mgmt ».
- Pour l'option **Emplacement**, choisissez la région dans laquelle héberger votre espace de noms.

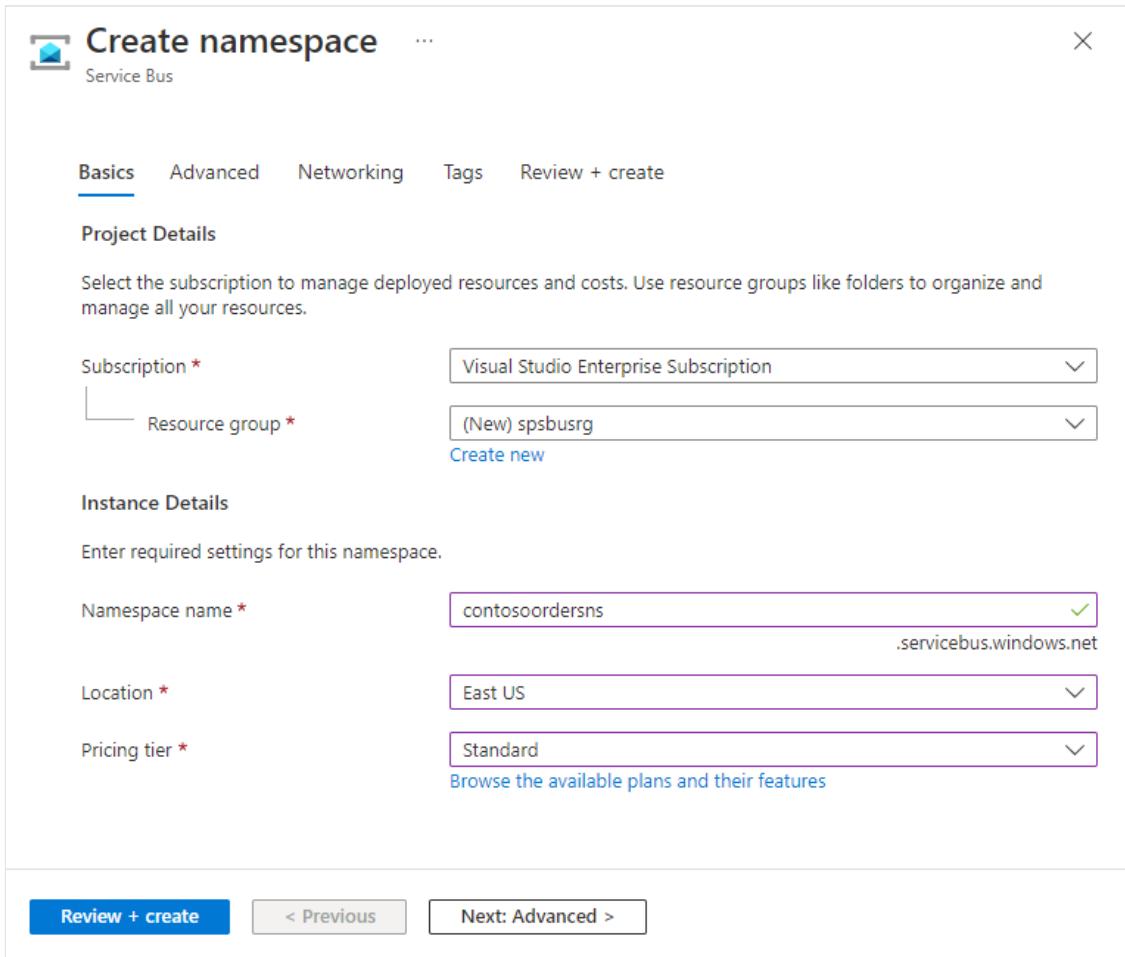
- e. Pour le **Niveau tarifaire**, sélectionnez le SKU (De base, Standard ou Premium) destiné à l'espace de noms. Pour ce guide de démarrage rapide, sélectionnez **Standard**.

 **Important**

Si vous voulez utiliser des **rubriques et des abonnements**, choisissez Standard ou Premium. Les rubriques/abonnements ne sont pas pris en charge dans le niveau tarifaire De base.

Si vous avez sélectionné le SKU **Premium**, précisez le nombre d'**unité de messagerie**. Le niveau Premium isole les ressources au niveau du processeur et de la mémoire, ce qui permet d'exécuter chaque charge de travail de manière isolée. Ce conteneur de ressources est appelé unité de messagerie. Un espace de noms Premium a au moins une unité de messagerie. Vous pouvez sélectionner 1, 2, 4, 8 ou 16 unités de messagerie pour chaque espace de noms Service Bus Premium. Pour plus d'informations, consultez [Messagerie Service Bus Premium](#).

- f. Au bas de la page, sélectionnez **Examiner et créer**.



The screenshot shows the 'Create namespace' wizard in the Azure portal. The title bar says 'Create namespace' and 'Service Bus'. The tabs at the top are 'Basics', 'Advanced', 'Networking', 'Tags', and 'Review + create'. The 'Basics' tab is selected. The 'Project Details' section contains fields for 'Subscription' (set to 'Visual Studio Enterprise Subscription') and 'Resource group' (set to '(New) spsbusrg'). Below that is the 'Instance Details' section, which includes 'Namespace name' (set to 'contosoordersns'), 'Location' (set to 'East US'), and 'Pricing tier' (set to 'Standard'). At the bottom of the screen are three buttons: 'Review + create' (highlighted in blue), '< Previous', and 'Next: Advanced >'.

g. Dans la page **Vérifier + créer**, passez en revue les paramètres, puis sélectionnez **Créer**.

4. Une fois le déploiement de la ressource réussi, sélectionnez **Accéder à la ressource** dans la page de déploiement.

The screenshot shows the 'Deployment' overview for a resource named 'contosoordersns'. A green checkmark indicates that the deployment is complete. The deployment details include the name, subscription, resource group, start time (10/20/2022, 4:45:03 PM), and correlation ID. There are sections for 'Deployment details' and 'Next steps', with a prominent red-bordered 'Go to resource' button. Below the main content, there are links to 'Give feedback' and 'Tell us about your experience with deployment'.

5. Vous voyez la page d'accueil de votre espace de noms Service Bus.

The screenshot shows the 'Overview' page for a Service Bus namespace named 'spsbusns1028'. The left sidebar lists various navigation options like 'Activity log', 'Access control (IAM)', 'Tags', etc. The main area displays 'Essentials' information such as resource group ('spsbusrg'), status ('Active'), location ('East US'), and subscription ('Visual Studio Enterprise Subscription'). It also shows 'Tags (edit)' and a timeline for the last hour with metrics for 'Requests' and 'Messages'. The 'Requests' chart shows 0 incoming and 0 successful requests, and 0 server errors. The 'Messages' chart shows 0 incoming and 0 outgoing messages.

Créer une rubrique à l'aide du Portail Azure

1. Dans la page **Espace de noms Service Bus**, sélectionnez **Rubriques** dans le menu de gauche.
2. Sélectionnez **+ Rubrique** dans la barre d'outils.

3. Entrez un **nom** pour la rubrique. Conservez les valeurs par défaut des autres options.

4. Cliquez sur **Créer**.

The screenshot shows the Azure portal interface for creating a new topic in a Service Bus namespace. On the left, there's a sidebar with 'Topics' selected under 'Entities'. In the main area, a 'Create topic' dialog is open. Step 1 highlights the 'Topics' link in the sidebar. Step 2 highlights the '+ Topic' button. Step 3 highlights the 'Name' field containing 'mytopic'. Step 4 highlights the 'Create' button at the bottom right of the dialog.

Créer un abonnement à la rubrique

1. Sélectionnez la **rubrique** que vous avez créée dans la section précédente.

The screenshot shows the Azure portal interface for managing a Service Bus topic. Under 'Entities', 'Topics' is selected. A table lists the topic 'mytopic'. Step 1 highlights the 'Topics' link in the sidebar. Step 2 highlights the '+ Rubrique' button in the top bar. Step 3 highlights the 'mytopic' row in the table. Step 4 highlights the 'Rubriques' button at the top right of the table.

2. Dans la page **Rubrique Service Bus**, dans la barre d'outils, sélectionnez + **Abonnement**.

The screenshot shows the Azure portal interface for managing Service Bus subscriptions under the 'mytopic' topic. The top navigation bar includes the tenant name 'mytopic (spsbusns1128/mytopic)', 'Abonnements', and a search bar. Below the navigation is a sidebar with links like 'Vue d'ensemble', 'Contrôle d'accès (IAM)', 'Diagnostiquer et résoudre les problèmes', 'Explorateur Service Bus', 'Paramètres', 'Stratégies d'accès partagé', 'Propriétés', and 'Verrous'. The main content area is titled 'Entités' and shows a table for 'Abonnements' with columns 'Nom', 'État', 'Nombre de messages', 'Messages actifs', and 'Messages de lettres mortes'. A search bar at the top of the table allows filtering by name. A summary bar on the right indicates '0' subscriptions. At the bottom of the table, there's a note 'Aucun résultat.' and a horizontal scrollbar. On the far left, a vertical sidebar lists 'Automatisation' options: 'CLI / PS', 'Tâches (préversion)', and 'Exporter le modèle'.

3. Dans la page **Créer un abonnement**, procédez comme suit :

- a. Entrez **S1** pour le **nom** de l'abonnement.
- b. Entrez **3** pour **Nombre maximal de remises**.
- c. Ensuite, sélectionnez **Créer** pour créer l'abonnement.

Create subscription

Service Bus

Name * ⓘ

S1



Max delivery count * ⓘ

10

Auto-delete after idle for ⓘ

Days

Hours

Minutes

Seconds

14

0

0

0

Never auto-delete

Forward messages to queue/topic ⓘ

MESSAGE SESSIONS

Service bus sessions allow ordered handling of unbounded sequences of related messages. With sessions enabled a subscription can guarantee first-in-first-out delivery of messages. [Learn more.](#)

Enable sessions

MESSAGE TIME TO LIVE AND DEAD-LETTERING

Message time to live (default) ⓘ

Days

Hours

Minutes

Seconds

14

0

0

0

Enable dead lettering on message expiration

Move messages that cause filter evaluation exceptions to the dead-letter subqueue

MESSAGE LOCK DURATION

Lock duration ⓘ

Days

Hours

Minutes

Seconds

0

0

1

0

Create

Authentifier l'application sur Azure

Ce guide de démarrage pratique vous montre deux façons de vous connecter à Azure Service Bus : **sans mot de passe** et avec une **chaîne de connexion**.

La première option vous explique comment utiliser votre principal de sécurité dans Microsoft Entra ID et le contrôle d'accès en fonction du rôle (RBAC) pour vous connecter à un espace de noms Service Bus. Vous n'avez pas à vous soucier d'avoir une chaîne de

connexion codée en dur dans votre code, dans un fichier config ni dans un stockage sécurisé comme Azure Key Vault.

La deuxième option consiste à se servir d'une chaîne de connexion pour se connecter à un espace de noms Service Bus. Si vous débutez avec Azure, vous trouverez peut-être l'option chaîne de connexion plus facile à suivre. Nous vous recommandons d'utiliser l'option sans mot de passe dans les applications réelles et les environnements de production. Pour plus d'informations, consultez [Authentification et autorisation](#). Pour en savoir plus sur l'authentification sans mot de passe, reportez-vous à la [page de présentation](#).

Sans mot de passe (recommandé)

Attribuer des rôles à votre utilisateur Microsoft Entra

Lors du développement localement, assurez-vous que le compte d'utilisateur qui se connecte à Azure Service Bus dispose des autorisations appropriées. Vous aurez besoin du rôle [Propriétaire de données Azure Service Bus](#) pour envoyer et recevoir des messages. Pour vous attribuer ce rôle, vous aurez besoin du rôle Administrateur de l'accès utilisateur ou d'un autre rôle qui inclut l'action

`Microsoft.Authorization/roleAssignments/write`. Vous pouvez attribuer des rôles RBAC Azure à un utilisateur à l'aide du Portail Azure, Azure CLI ou Azure PowerShell. Découvrez les étendues disponibles pour les attributions de rôles dans la page [vue d'ensemble de l'étendue](#).

L'exemple suivant attribue le rôle `Azure Service Bus Data Owner` à votre compte d'utilisateur, qui fournit un accès complet aux ressources Azure Service Bus. Dans un scénario réel, suivez le [principe des privilèges](#) minimum pour accorder aux utilisateurs uniquement les autorisations minimales nécessaires à un environnement de production plus sécurisé.

Rôles Azure intégrés pour Azure Service Bus

Pour Azure Service Bus, la gestion des espaces de noms et de toutes les ressources associées via le Portail Azure et l'API de gestion des ressources Azure est déjà protégée à l'aide du modèle Azure RBAC. Azure fournit les rôles Azure intégrés ci-dessous pour autoriser l'accès à un espace de noms Service Bus :

- [Propriétaire de données Azure Service Bus](#) : ce rôle permet l'accès aux données de l'espace de noms Service Bus et de ses entités (files d'attente,

rubriques, abonnements et filtres). Un membre de ce rôle peut envoyer et recevoir des messages à partir de files d'attente ou de rubriques et d'abonnements.

- [Expéditeur de données Azure Service Bus](#) : utilisez ce rôle pour autoriser l'accès en envoi à l'espace de noms Service Bus et à ses entités.
- [Récepteur de données Azure Service Bus](#) : utilisez ce rôle pour autoriser l'accès en réception à l'espace de noms Service Bus et à ses entités.

Si vous souhaitez créer un rôle personnalisé, consultez [Droits requis pour les opérations Service Bus](#).

Ajouter un utilisateur Microsoft Entra au rôle Propriétaire Azure Service Bus

Ajoutez votre nom d'utilisateur Microsoft Entra au rôle **Propriétaire de données Azure Service Bus** au niveau de l'espace de noms Service Bus. Il permet à une application exécutée dans le contexte de votre compte d'utilisateur d'envoyer des messages à une file d'attente ou à une rubrique et d'en recevoir auprès d'une file d'attente ou de l'abonnement d'une rubrique.

Important

Dans la plupart des cas, la propagation de l'attribution de rôle dans Azure peut prendre une ou deux minutes. Dans de rares cas, cela peut prendre jusqu'à **huit minutes**. Si vous recevez des erreurs d'authentification lorsque vous exécutez votre code pour la première fois, patientez quelques instants et réessayez.

1. Si la page Espace de noms Service Bus n'est pas ouverte sur le Portail Azure, recherchez votre espace de noms Service Bus à l'aide de la barre de recherche principale ou du volet de navigation de gauche.
2. Dans la page vue d'ensemble, sélectionnez **Contrôle d'accès (IAM)** dans le menu de gauche.
3. Sur la page **Contrôle d'accès (IAM)**, sélectionnez l'onglet **Attributions de rôles**.
4. Sélectionnez **+ Ajouter** dans le menu supérieur, puis **Ajouter une attribution de rôle** dans le menu déroulant résultant.

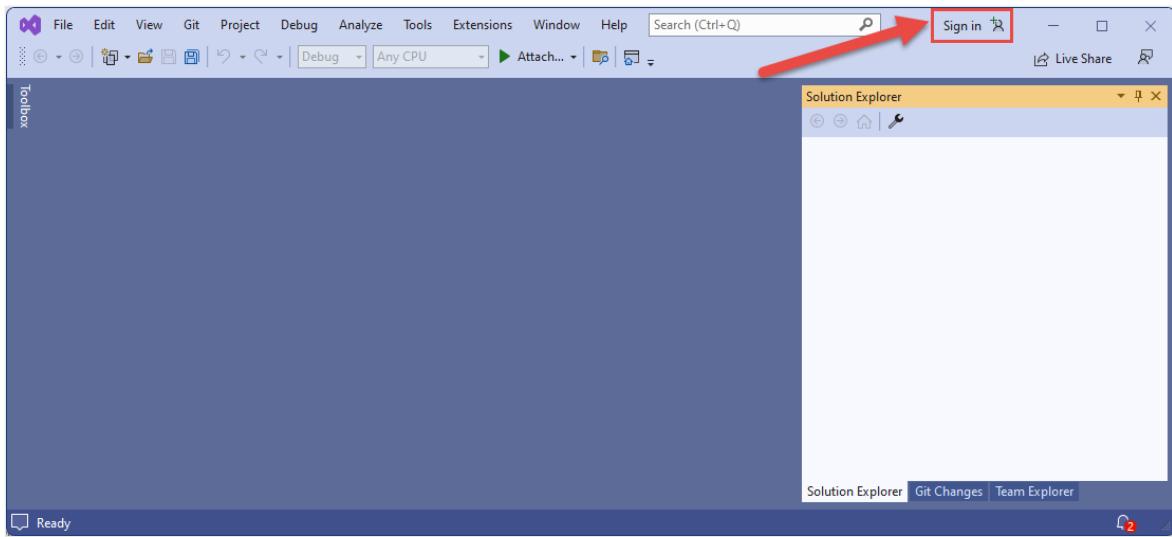
The screenshot shows the 'Access control (IAM)' section of the Azure Service Bus Namespace. The left sidebar has 'Access control (IAM)' selected. The main area has three numbered callouts: 1 points to the 'Access control (IAM)' menu item; 2 points to the 'Add role assignment' button; 3 points to the 'Add role assignment' tooltip.

5. Utilisez la zone de recherche pour filtrer les résultats sur le rôle souhaité. Pour cet exemple, recherchez **Azure Service Bus Data Owner** et sélectionnez le résultat correspondant. Ensuite, choisissez **Suivant**.
6. Sous **Attribuer l'accès à**, sélectionnez **Utilisateur, groupe ou principal de service**, puis sélectionnez + **Sélectionner des membres**.
7. Dans la boîte de dialogue, recherchez votre nom d'utilisateur Microsoft Entra (généralement votre adresse e-mail *user@domain*), puis choisissez **Sélectionner** en bas de la boîte de dialogue.
8. Sélectionnez **Vérifier + affecter** pour accéder à la page finale, puis **Vérifier + attribuer** à nouveau pour terminer le processus.

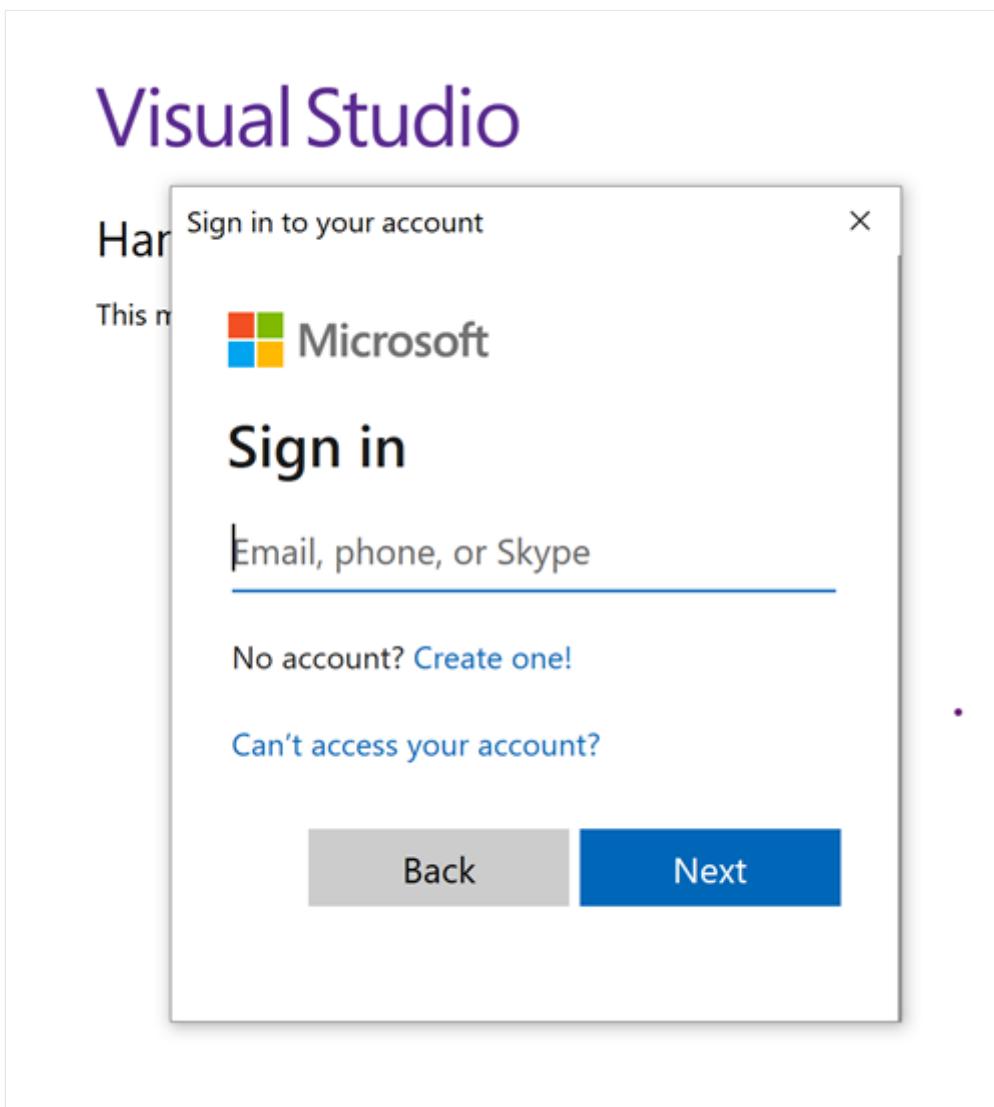
Lancer Visual Studio et vous connecter à Azure

Vous pouvez autoriser l'accès à l'espace de noms Service Bus en procédant comme suit :

1. Lancez Visual Studio. Si la fenêtre **Démarrage** s'affiche, sélectionnez le lien **Continuer sans code** dans le volet de droite.
2. Sélectionnez le bouton **Se connecter** en haut à droite de Visual Studio.



3. Connectez-vous à l'aide du compte Microsoft Entra auquel vous avez attribué un rôle précédemment.



Envoyez des messages à la rubrique

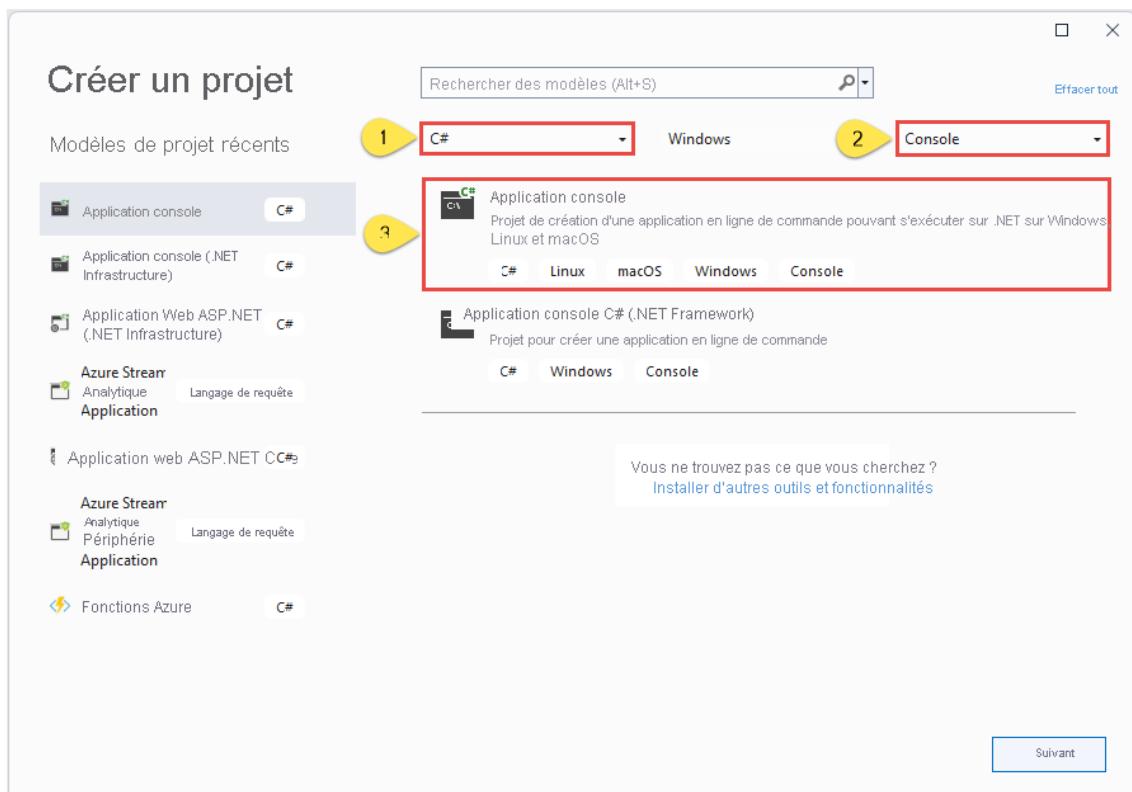
Cette section montre comment créer une application console .NET pour envoyer des messages à une rubrique Service Bus.

① Notes

Ce guide de démarrage rapide fournit des instructions pas à pas pour un scénario simple qui consiste à envoyer un lot de messages à une rubrique Service Bus et à recevoir ces messages à partir d'un abonnement de la rubrique. Pour plus d'exemples sur d'autres scénarios et des scénarios avancés, voir [Exemples .NET Service Bus sur GitHub](#).

Création d'une application console

1. Dans Visual Studio, sélectionnez le menu **Fichier - > Nouveau - > Projet**.
2. Dans la boîte de dialogue **Créer un projet**, effectuez les étapes suivantes : Si vous ne voyez pas cette boîte de dialogue, sélectionnez **Fichier** dans le menu, sélectionnez **Nouveau**, puis **Projet**.
 - a. Sélectionnez **C#** en guise de langage de programmation.
 - b. Sélectionnez **Console** comme type de l'application.
 - c. Sélectionnez **Application console** dans la liste des résultats.
 - d. Ensuite, sélectionnez **Suivant**.



3. Entrez **TopicSender** comme nom de projet, **ServiceBusTopicQuickStart** comme nom de solution, puis sélectionnez **Suivant**.
4. Dans la page **Informations supplémentaires**, sélectionnez **Créer** pour créer la solution et le projet.

Ajouter les paquets NuGet au projet

Sans mot de passe

1. Cliquez sur **Outils > Gestionnaire de package NuGet > Console du Gestionnaire de package** à partir du menu.
2. Exécutez la commande suivante pour installer le package NuGet **Azure.Messaging.ServiceBus**.

PowerShell

```
Install-Package Azure.Messaging.ServiceBus
```

3. Exécutez la commande suivante pour installer le package NuGet **Azure.Identity**.

PowerShell

```
Install-Package Azure.Identity
```

Ajouter du code pour envoyer des messages à la rubrique

1. Remplacez le contenu de Program.cs par le code suivant. Les étapes importantes sont décrites dans cette section, et vous trouverez des informations supplémentaires dans les commentaires de code.

Sans mot de passe

- a. Crée un objet **ServiceBusClient** à l'aide de l'objet **DefaultAzureCredential**.

DefaultAzureCredential détecte et utilise automatiquement les informations d'identification de votre connexion Visual Studio pour vous authentifier auprès d'Azure Service Bus.

- b. Utilise la méthode `CreateSender` sur l'objet `ServiceBusClient` pour créer un objet `ServiceBusSender` pour la rubrique Service Bus spécifique.
- c. Crée un objet `ServiceBusMessageBatch` à l'aide de `ServiceBusSender.CreateMessageBatchAsync`.
- d. Ajoutez des messages au lot à l'aide de `ServiceBusMessageBatch.TryAddMessage`.
- e. Envoie le lot de messages à la rubrique Service Bus avec la méthode `ServiceBusSender.SendMessagesAsync`.

ⓘ Important

Remplacez les valeurs d'espace réservé (`<NAMESPACE-NAME>` et `<TOPIC-NAME>`) dans l'extrait de code par les noms de votre espace de noms et de votre rubrique Service Bus.

C#

```
using System.Threading.Tasks;
using Azure.Messaging.ServiceBus;
using Azure.Identity;

// the client that owns the connection and can be used to create
// senders and receivers
ServiceBusClient client;

// the sender used to publish messages to the topic
ServiceBusSender sender;

// number of messages to be sent to the topic
const int numMessages = 3;

// The Service Bus client types are safe to cache and use as a
// singleton for the lifetime
// of the application, which is best practice when messages are
// being published or read
// regularly.

//TODO: Replace the "<NAMESPACE-NAME>" and "<TOPIC-NAME>"
//placeholders.
client = new ServiceBusClient(
    "<NAMESPACE-NAME>.servicebus.windows.net",
    new DefaultAzureCredential());
sender = client.CreateSender("<TOPIC-NAME>");

// create a batch
using ServiceBusMessageBatch messageBatch = await
    sender.CreateMessageBatchAsync();
```

```

for (int i = 1; i <= numOfMessages; i++)
{
    // try adding a message to the batch
    if (!messageBatch.TryAddMessage(new ServiceBusMessage($"Message
{i}")))
    {
        // if it is too large for the batch
        throw new Exception($"The message {i} is too large to fit
in the batch.");
    }
}

try
{
    // Use the producer client to send the batch of messages to the
    Service Bus topic
    await sender.SendMessagesAsync(messageBatch);
    Console.WriteLine($"A batch of {numOfMessages} messages has
been published to the topic.");
}
finally
{
    // Calling DisposeAsync on client types is required to ensure
    that network
    // resources and other unmanaged objects are properly cleaned
    up.
    await sender.DisposeAsync();
    await client.DisposeAsync();
}

Console.WriteLine("Press any key to end the application");
Console.ReadKey();

```

2. Générez le projet et vérifiez qu'il ne présente pas d'erreurs.

3. Exécutez le programme et attendez le message de confirmation.

Bash

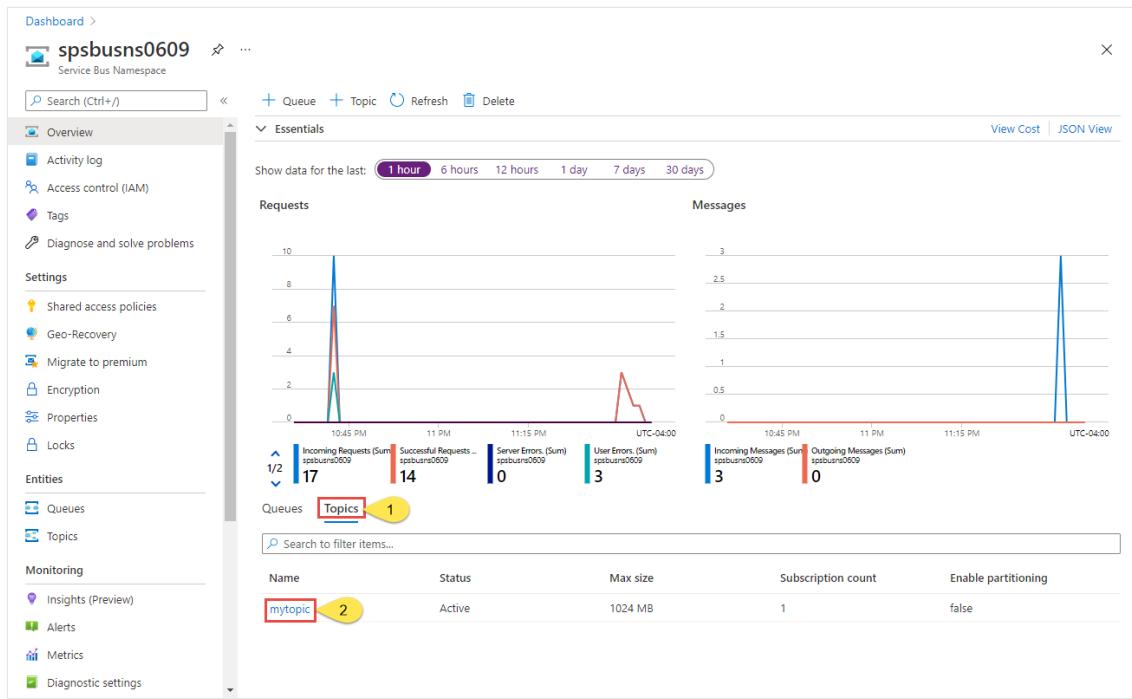
A batch of 3 messages has been published to the topic

Important

Dans la plupart des cas, la propagation de l'attribution de rôle dans Azure peut prendre une ou deux minutes. Dans de rares cas, il peut prendre jusqu'à **huit minutes**. Si vous recevez des erreurs d'authentification lorsque vous exécutez votre code pour la première fois, patientez quelques instants et réessayez.

4. Dans le portail Azure, procédez comme suit :

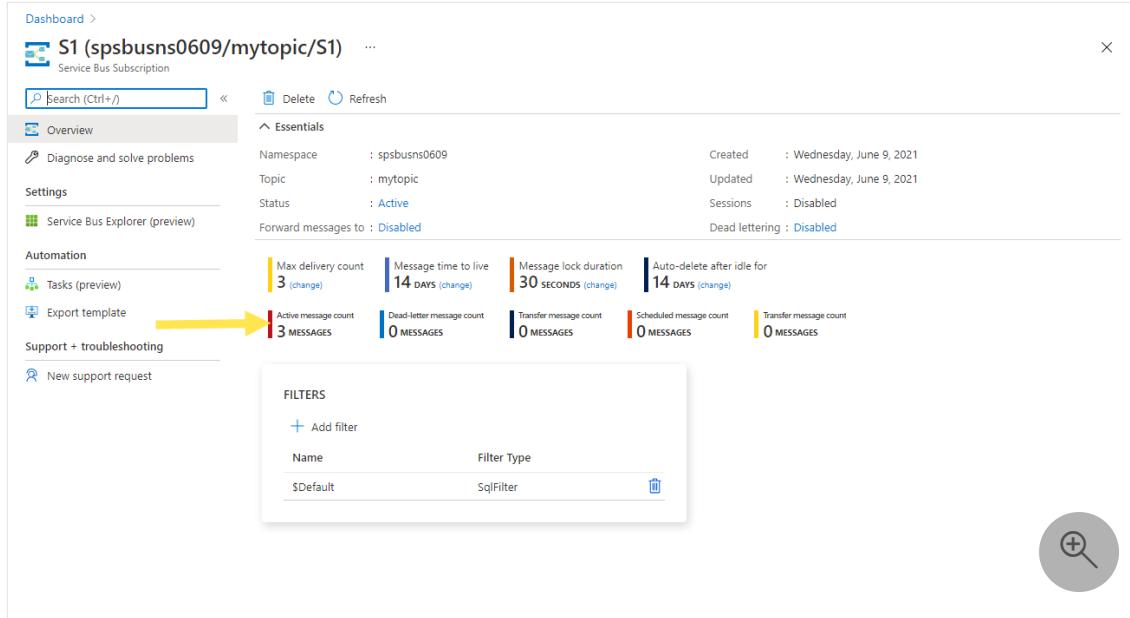
- a. Accédez à votre espace de noms Service Bus.
- b. Dans la page **Vue d'ensemble**, dans le volet central inférieur, basculez vers l'onglet **Rubriques**, puis sélectionnez la rubrique Service Bus. Dans l'exemple suivant, il s'agit de `mytopic`.



- c. Dans la page **Rubrique Service Bus**, dans le graphique **Messages** de la section inférieure **Métriques**, vous pouvez voir trois messages entrants pour la rubrique. Si vous ne voyez pas cette valeur, attendez quelques minutes, puis actualisez la page pour voir le graphique mis à jour.



d. Sélectionnez l'abonnement dans le volet inférieur. Dans l'exemple suivant, il s'agit de S1. Dans la page **Abonnement Service Bus**, vous voyez que **Nombre de messages actifs** a pour valeur 3. L'abonnement a reçu les trois messages que vous avez envoyés à la rubrique, mais aucun destinataire ne les a encore sélectionnés.



The screenshot shows the Azure Service Bus Subscription 'S1' dashboard. On the left, there's a sidebar with 'Overview', 'Diagnose and solve problems', 'Settings', 'Service Bus Explorer (preview)', 'Automation' (Tasks (preview) and Export template), 'Support + troubleshooting' (New support request), and a search bar. The main area has tabs for 'Essentials' and 'Messages'. Under 'Essentials', it shows Namespace: spsbusns0609, Topic: mytopic, Status: Active, and Forward messages to: Disabled. It also shows Created: Wednesday, June 9, 2021, Updated: Wednesday, June 9, 2021, Sessions: Disabled, and Dead lettering: Disabled. Below this, there are fields for Max delivery count (3), Message time to live (14 DAYS), Message lock duration (30 SECONDS), Auto-delete after idle for (14 DAYS), Active message count (3 MESSAGES), Dead-letter message count (0 MESSAGES), Transfer message count (0 MESSAGES), Scheduled message count (0 MESSAGES), and Transfer message count (0 MESSAGES). A yellow arrow points to the 'Active message count' field. At the bottom, there's a 'FILTERS' section with a 'Add filter' button and a table for filters, showing one entry: Name: \$Default, Filter Type: SqlFilter. A magnifying glass icon is in the bottom right corner.

Réception des messages d'un abonnement

Dans cette section, vous créez une application console .NET qui reçoit les messages de l'abonnement à la rubrique Service Bus.

ⓘ Notes

Ce guide de démarrage rapide fournit des instructions pas à pas pour un scénario simple qui consiste à envoyer un lot de messages à une rubrique Service Bus et à recevoir ces messages à partir d'un abonnement de la rubrique. Pour plus d'exemples sur d'autres scénarios et des scénarios avancés, voir [Exemples .NET Service Bus sur GitHub](#).

Créer un projet pour le récepteur

1. Dans la fenêtre Explorateur de solutions, cliquez avec le bouton droit sur la solution **ServiceBusTopicQuickStart**, pointez sur **Ajouter**, puis sélectionnez **Nouveau projet**.
2. Sélectionnez **Application console**, puis **Suivant**.
3. Entrez **SubscriptionReceiver** comme **Nom du projet**, puis sélectionnez **Suivant**.

4. Sur la page **Informations supplémentaires**, sélectionnez **Créer**.
5. Dans la fenêtre **Explorateur de solutions**, cliquez avec le bouton droit sur **SubscriptionReceiver**, puis sélectionnez **Définir comme projet de démarrage**.

Ajouter les paquets NuGet au projet

Sans mot de passe

1. Cliquez sur **Outils > Gestionnaire de package NuGet > Console du Gestionnaire de package** à partir du menu.
2. Sélectionnez **SubscriptionReceiver** dans la liste déroulante **Projet par défaut**.
3. Exécutez la commande suivante pour installer le package NuGet **Azure.Messaging.ServiceBus**.

PowerShell

```
Install-Package Azure.Messaging.ServiceBus
```

4. Exécutez la commande suivante pour installer le package NuGet **Azure.Identity**.

PowerShell

```
Install-Package Azure.Identity
```

Ajouter du code pour recevoir des messages de l'abonnement

Dans cette section, vous ajoutez du code pour récupérer les messages de l'abonnement.

1. Remplacez le contenu existant de **Program.cs** par les propriétés et méthodes suivantes :

Sans mot de passe

C#

```

using System.Threading.Tasks;
using Azure.Messaging.ServiceBus;
using Azure.Identity;

// the client that owns the connection and can be used to create
// senders and receivers
ServiceBusClient client;

// the processor that reads and processes messages from the
// subscription
ServiceBusProcessor processor;

// handle received messages
async Task MessageHandler(ProcessMessageEventArgs args)
{
    string body = args.Message.Body.ToString();
    Console.WriteLine($"Received: {body} from subscription.");

    // complete the message. messages is deleted from the
    // subscription.
    await args.CompleteMessageAsync(args.Message);
}

// handle any errors when receiving messages
Task ErrorHandler(ProcessErrorEventArgs args)
{
    Console.WriteLine(args.Exception.ToString());
    return Task.CompletedTask;
}

```

2. Ajoutez le code suivant à la fin de `Program.cs`.

Sans mot de passe

- Crée un objet `ServiceBusClient` à l'aide de l'objet `DefaultAzureCredential`.
`DefaultAzureCredential` détecte et utilise automatiquement les informations d'identification de votre connexion Visual Studio pour vous authentifier auprès d'Azure Service Bus.
- Appelle la méthode `CreateProcessor` sur l'objet `ServiceBusClient` pour créer un objet `ServiceBusProcessor` pour la rubrique Service Bus spécifiée.
- Spécifie des gestionnaires pour les événements `ProcessMessageAsync` et `ProcessErrorAsync` de l'objet `ServiceBusProcessor`.
- Démarre le traitement des messages en appelant `StartProcessingAsync` sur l' objet `ServiceBusProcessor`.

- Lorsque l'utilisateur appuie sur une clé pour terminer le traitement, il appelle `StopProcessingAsync` sur l' objet `ServiceBusProcessor`.

ⓘ Important

Remplacez les valeurs d'espace réservé (<NAMESPACE-NAME>, <TOPIC-NAME> et <SUBSCRIPTION-NAME>) dans l'extrait de code par les noms de votre espace de noms, de votre rubrique et de votre abonnement Service Bus.

Pour plus d'informations, consultez les commentaires du code.

C#

```
// The Service Bus client types are safe to cache and use as a
singleton for the lifetime
// of the application, which is best practice when messages are
being published or read
// regularly.
//
// Create the clients that we'll use for sending and processing
messages.
// TODO: Replace the <NAMESPACE-NAME> placeholder
client = new ServiceBusClient(
    "<NAMESPACE-NAME>.servicebus.windows.net",
    new DefaultAzureCredential());

// create a processor that we can use to process the messages
// TODO: Replace the <TOPIC-NAME> and <SUBSCRIPTION-NAME>
placeholders
processor = client.CreateProcessor("<TOPIC-NAME>", "<SUBSCRIPTION-
NAME>", new ServiceBusProcessorOptions());

try
{
    // add handler to process messages
    processor.ProcessMessageAsync += MessageHandler;

    // add handler to process any errors
    processor.ProcessErrorAsync += ErrorHandler;

    // start processing
    await processor.StartProcessingAsync();

    Console.WriteLine("Wait for a minute and then press any key to
end the processing");
    Console.ReadKey();

    // stop processing
    Console.WriteLine("\nStopping the receiver...");
    await processor.StopProcessingAsync();
}
```

```

        Console.WriteLine("Stopped receiving messages");
    }
    finally
    {
        // Calling DisposeAsync on client types is required to ensure
        // that network
        // resources and other unmanaged objects are properly cleaned
        // up.
        await processor.DisposeAsync();
        await client.DisposeAsync();
    }
}

```

3. Voici à quoi doit ressembler votre fichier `Program.cs` :

Sans mot de passe

C#

```

using System;
using System.Threading.Tasks;
using Azure.Messaging.ServiceBus;
using Azure.Identity;

// the client that owns the connection and can be used to create
// senders and receivers
ServiceBusClient client;

// the processor that reads and processes messages from the
// subscription
ServiceBusProcessor processor;

// handle received messages
async Task MessageHandler(ProcessMessageEventArgs args)
{
    string body = args.Message.Body.ToString();
    Console.WriteLine($"Received: {body} from subscription.");

    // complete the message. message is deleted from the
    // subscription.
    await args.CompleteMessageAsync(args.Message);
}

// handle any errors when receiving messages
Task ErrorHandler(ProcessErrorEventArgs args)
{
    Console.WriteLine(args.Exception.ToString());
    return Task.CompletedTask;
}

// The Service Bus client types are safe to cache and use as a
// singleton for the lifetime

```

```

// of the application, which is best practice when messages are
being published or read
// regularly.
//
// Create the clients that we'll use for sending and processing
messages.
// TODO: Replace the <NAMESPACE-NAME> placeholder
client = new ServiceBusClient(
    "<NAMESPACE-NAME>.servicebus.windows.net",
    new DefaultAzureCredential());

// create a processor that we can use to process the messages
// TODO: Replace the <TOPIC-NAME> and <SUBSCRIPTION-NAME>
placeholders
processor = client.CreateProcessor("<TOPIC-NAME>", "<SUBSCRIPTION-
NAME>", new ServiceBusProcessorOptions());

try
{
    // add handler to process messages
    processor.ProcessMessageAsync += MessageHandler;

    // add handler to process any errors
    processor.ProcessErrorAsync += ErrorHandler;

    // start processing
    await processor.StartProcessingAsync();

    Console.WriteLine("Wait for a minute and then press any key to
end the processing");
    Console.ReadKey();

    // stop processing
    Console.WriteLine("\nStopping the receiver...");
    await processor.StopProcessingAsync();
    Console.WriteLine("Stopped receiving messages");
}
finally
{
    // Calling DisposeAsync on client types is required to ensure
that network
    // resources and other unmanaged objects are properly cleaned
up.
    await processor.DisposeAsync();
    await client.DisposeAsync();
}

```

4. Générez le projet et vérifiez qu'il ne présente pas d'erreurs.
5. Exécutez l'application réceptrice. Vous devriez voir les messages reçus. Appuyez sur n'importe quelle touche pour arrêter le récepteur et l'application.

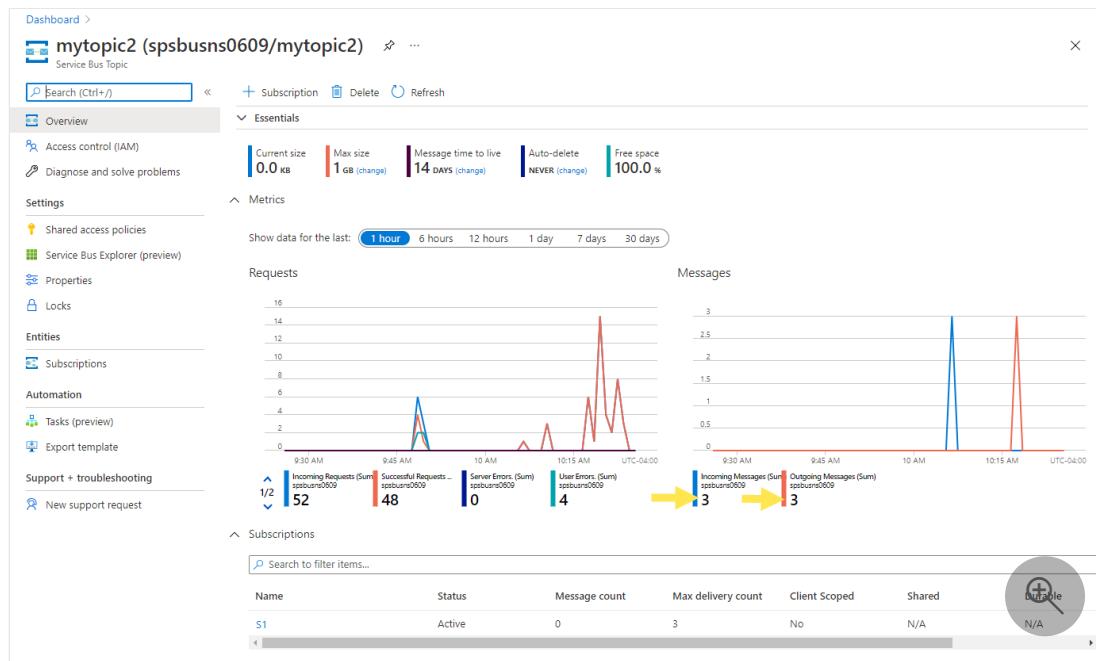
Console

```
Wait for a minute and then press any key to end the processing
Received: Message 1 from subscription: S1
Received: Message 2 from subscription: S1
Received: Message 3 from subscription: S1
```

```
Stopping the receiver...
Stopped receiving messages
```

6. Vérifiez à nouveau le portail.

- Dans la page **Rubrique Service Bus**, dans le graphique **Messages**, vous voyez trois messages entrants et trois messages sortants. Si vous ne voyez pas ces valeurs, attendez quelques minutes et actualisez la page pour voir une mise à jour du graphique.



- Dans la page **Abonnement Service Bus**, vous voyez que **Nombre de messages actifs** est égal à zéro. Cela est dû au fait qu'un destinataire a reçu des messages de cet abonnement et a terminé les messages.

Dashboard > mytopic2 (spsbusns0609/mytopic2) >

S1 (spsbusns0609/mytopic2/S1) ...

Service Bus Subscription

Search (Ctrl+ /) Delete Refresh

Overview

Diagnose and solve problems

Settings

Service Bus Explorer (preview)

Automation

Tasks (preview)

Export template

Support + troubleshooting

New support request

Namespace: spsbusns0609

Topic: mytopic2

Status: Active

Forward messages to: Disabled

Created: Thursday, June 10, 2021

Updated: Thursday, June 10, 2021

Sessions: Disabled

Dead lettering: Disabled

Max delivery count: 3 (change)

Message time to live: 14 DAYS (change)

Message lock duration: 30 SECONDS (change)

Auto-delete after idle for: 14 DAYS (change)

Active message count: 0 MESSAGES

Dead-letter message count: 0 MESSAGES

Transfer message count: 0 MESSAGES

Scheduled message count: 0 MESSAGES

Transfer message count: 0 MESSAGES

FILTERS

Add filter

Name	Filter Type
\$Default	SqlFilter

+

Étapes suivantes

Consultez la documentation et les exemples suivants :

- [Bibliothèque de client Azure Service Bus pour .NET – Fichier Lisez-moi](#) ↗
- [Échantillons .NET pour Azure Service Bus sur GitHub](#) ↗
- [Informations de référence sur l'API .NET](#)

Démarrage rapide : Bibliothèque de client Stockage Blob Azure pour .NET

Article • 12/02/2024

ⓘ Contenu assisté par l'IA. Cet article a été en partie créé à l'aide d'une IA. Un auteur a examiné et si besoin révisé le contenu. [En savoir plus](#)

ⓘ Notes

L'option **Générer à partir de zéro** vous guide pas à pas dans le processus de création d'un projet, d'installation de packages, d'écriture du code et d'exécution d'une application console de base. Cette approche est recommandée si vous souhaitez comprendre tous les détails impliqués dans la création d'une application qui se connecte au service Stockage Blob Azure. Si vous préférez automatiser les tâches de déploiement et commencer par un projet terminé, choisissez [Commencer avec un modèle](#).

Commencez à utiliser la bibliothèque cliente Stockage Blob Azure pour .NET. Stockage Blob Azure est la solution de stockage d'objets de Microsoft pour le cloud, optimisée pour stocker de grandes quantités de données non structurées.

Dans cet article, vous suivez les étapes pour installer le package et essayer l'exemple de code pour les tâches de base.

[Documentation de référence sur l'API](#) | [Code source de la bibliothèque](#) ↗ | [Package \(NuGet\)](#) ↗ | [Exemples](#)

Cette vidéo vous montre comment commencer à utiliser la bibliothèque cliente Stockage Blob Azure pour .NET.

<https://learn-video.azurefd.net/vod/player?id=cdae65e7-1892-48fe-934a-70edfbe147be&locale=fr-fr&embedUrl=%2Fazure%2Fstorage%2Fblobs%2Fstorage-quickstart-blobs-dotnet> ↗

Les étapes de la vidéo sont également décrites dans les sections suivantes.

Prérequis

- Abonnement Azure : [créez-en un gratuitement](#) ↗
- Compte de stockage Azure : [créez un compte de stockage](#)

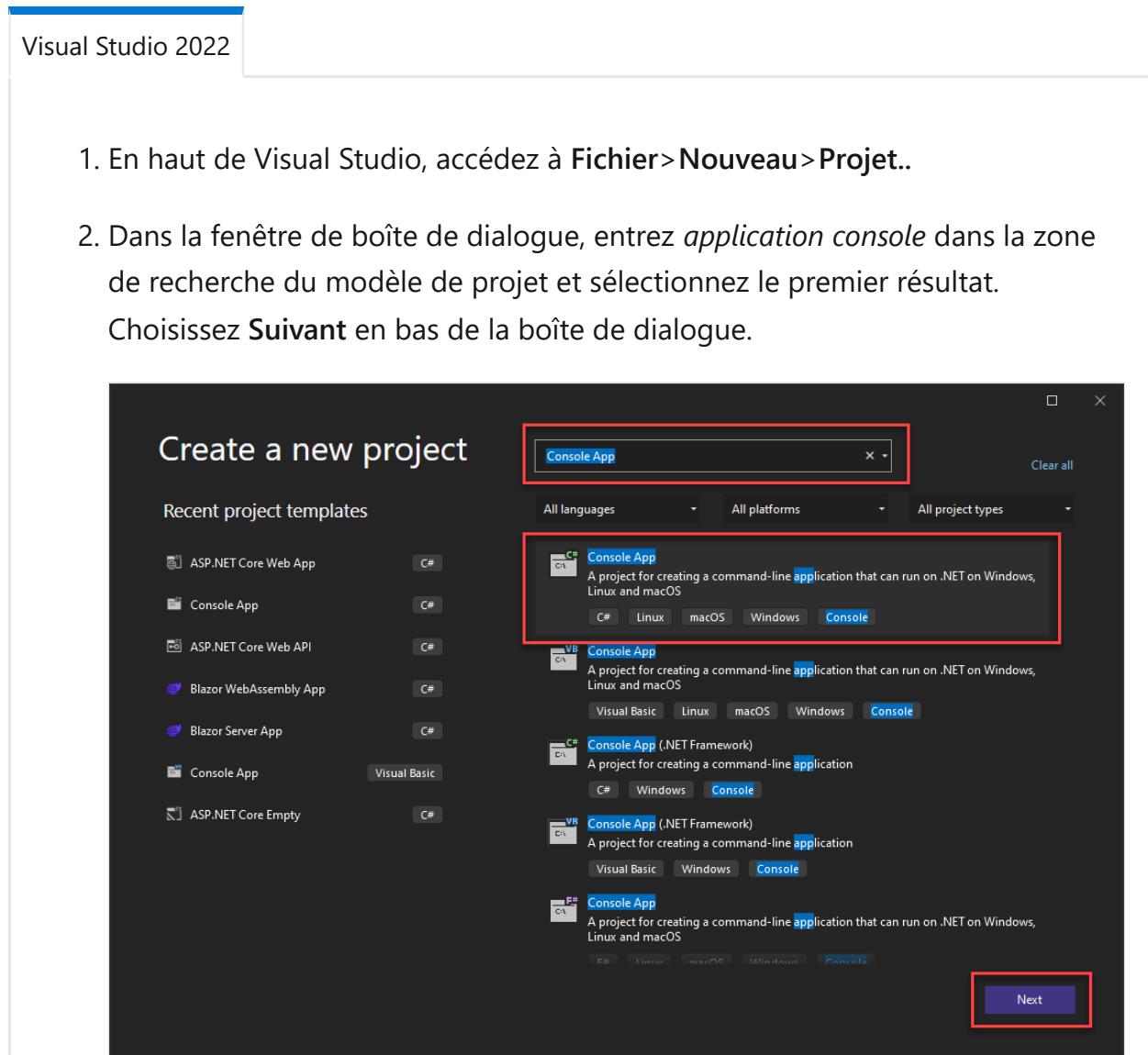
- Dernière version du [Kit de développement logiciel \(SDK\) .NET](#) pour votre système d'exploitation. Veillez à disposer du Kit de développement logiciel (SDK), et non du runtime.

Configuration

Cette section vous guide tout au long de la préparation d'un projet à utiliser avec la bibliothèque cliente Stockage Blob Azure pour .NET.

Créer le projet

Créez une application console .NET à l'aide de l'interface CLI .NET ou de Visual Studio 2022.



1. En haut de Visual Studio, accédez à **Fichier>Nouveau>Projet..**
2. Dans la fenêtre de boîte de dialogue, entrez *application console* dans la zone de recherche du modèle de projet et sélectionnez le premier résultat. Choisissez **Suivant** en bas de la boîte de dialogue.

3. Pour le **nom du projet**, entrez *BlobQuickstart*. Conservez les valeurs par défaut pour le reste des champs, puis sélectionnez **Suivant**.

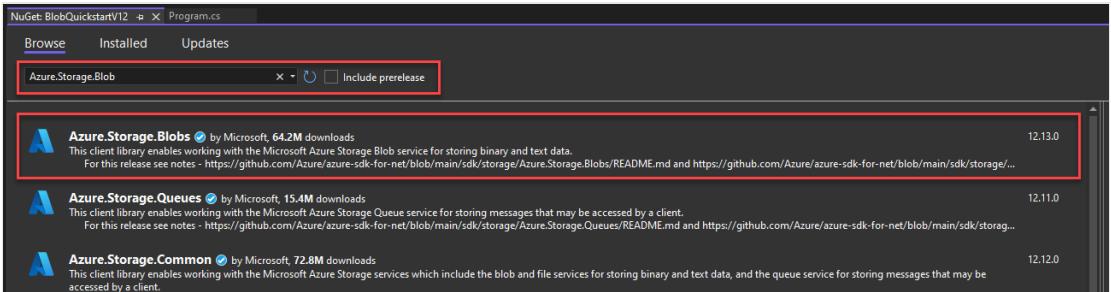
4. Pour le **Framework**, vérifiez que la dernière version installée de .NET est sélectionnée. Choisissez ensuite **Créer**. Le nouveau projet s'ouvre dans l'environnement Visual Studio.

Installer le package

Pour interagir avec Stockage Blob Azure, installez la bibliothèque cliente Stockage Blob Azure pour .NET.

Visual Studio 2022

1. Dans **Explorateur de solutions**, cliquez avec le bouton droit sur le nœud **Dépendances** de votre projet. Sélectionnez **Gérer les packages NuGet**.
2. Dans la fenêtre résultante, recherchez *Azure.Storage.Blobs*. Sélectionnez le résultat approprié, puis sélectionnez **Installer**.



Configurer le code d'application

Remplacez le code de départ dans le fichier `Program.cs` afin qu'il corresponde à l'exemple suivant, qui inclut les instructions `using` nécessaires pour cet exercice.

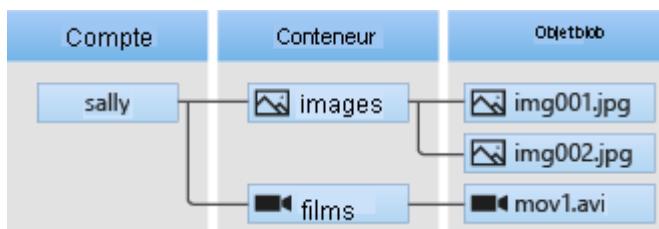
```
C#  
  
using Azure.Storage.Blobs;  
using Azure.Storage.Blobs.Models;  
using System;  
using System.IO;  
  
// See https://aka.ms/new-console-template for more information  
Console.WriteLine("Hello, World!");
```

Modèle objet

Stockage Blob Azure est optimisé pour stocker des quantités massives de données non structurées. Les données non structurées n'obéissent pas à un modèle ou une définition de données en particulier, comme des données texte ou binaires. Le stockage Blob offre trois types de ressources :

- Le compte de stockage
- Un conteneur dans le compte de stockage.
- Un blob dans le conteneur

Le diagramme suivant montre la relation entre ces ressources.



Utilisez les classes .NET suivantes pour interagir avec ces ressources :

- [BlobServiceClient](#): La classe `BlobServiceClient` vous permet de manipuler les ressources de stockage Azure et les conteneurs blob.
- [BlobContainerClient](#) : La classe `BlobContainerClient` vous permet de manipuler des conteneurs de stockage Azure et leurs blobs.
- [BlobClient](#) : La classe `BlobClient` vous permet de manipuler des blobs de stockage Azure.

Exemples de code

Les exemples d'extraits de code présentés dans les sections suivantes vous montrent comment effectuer les tâches suivantes sur les données avec la bibliothèque de client Stockage Blob Azure pour .NET :

- [S'authentifier auprès d'Azure et autoriser l'accès aux données blob](#)
- [Créer un conteneur](#)
- [Charger un objet blob dans un conteneur](#)
- [Lister les objets blob d'un conteneur](#)
- [Télécharger un objet blob](#)
- [Supprimer un conteneur](#)

Important

Vérifiez que vous avez installé les packages NuGet corrects et ajouté les instructions d'utilisation nécessaires pour que les exemples de code fonctionnent, comme décrit dans la section [configuration](#).

S'authentifier auprès d'Azure et autoriser l'accès aux données blob

Les demandes d'application vers le Stockage Blob Azure doivent être autorisées.

L'utilisation de la classe `DefaultAzureCredential` fournie par la bibliothèque de client Azure Identity est l'approche recommandée pour implémenter des connexions sans mot de passe aux services Azure dans votre code, y compris le Stockage Blob.

Vous pouvez également autoriser les demandes vers le Stockage Blob Azure à l'aide de la clé d'accès au compte. Toutefois, cette approche doit être utilisée avec prudence. Les développeurs doivent être vigilants pour ne jamais exposer la clé d'accès dans un emplacement non sécurisé. Toute personne disposant de la clé d'accès est en mesure d'autoriser les demandes sur le compte de stockage et a accès efficacement à toutes les données. `DefaultAzureCredential` offre des avantages améliorés en matière de gestion et de sécurité par rapport à la clé de compte pour autoriser l'authentification sans mot de passe. Les deux options sont illustrées dans l'exemple suivant.

Sans mot de passe (recommandé)

`DefaultAzureCredential` est une classe fournie par la bibliothèque de client Azure Identity pour .NET. Pour en savoir plus, consultez la [vue d'ensemble de DefaultAzureCredential](#). `DefaultAzureCredential` prend en charge plusieurs méthodes d'authentification et détermine quelle méthode doit être utilisée au moment de l'exécution. Cette approche permet à votre application d'utiliser différentes méthodes d'authentification dans différents environnements (local ou production) sans implémenter de code spécifique à l'environnement.

L'ordre et les emplacements dans lesquels `DefaultAzureCredential` les informations d'identification sont disponibles dans la [vue d'ensemble de la bibliothèque d'identités Azure](#).

Par exemple, votre application peut s'authentifier à l'aide de vos informations d'identification de connexion Visual Studio lors du développement local. Votre application peut ensuite utiliser une [identité managée](#) une fois qu'elle a été déployée sur Azure. Aucune modification du code n'est requise pour cette transition.

Attribuer des rôles à votre compte d'utilisateur Microsoft Entra

Lors du développement local, assurez-vous que le compte d'utilisateur qui accède aux données blob dispose des autorisations appropriées. Vous aurez besoin du **Contributeur aux données Blob de stockage** pour lire et écrire des données blob. Pour vous attribuer ce rôle, vous aurez besoin du rôle **Administrateur de l'accès utilisateur** ou d'un autre rôle qui inclut l'action **Microsoft.Authorization/roleAssignments/write**. Vous pouvez attribuer des rôles RBAC Azure à un utilisateur à l'aide du Portail Azure, Azure CLI ou Azure PowerShell. Vous pouvez en savoir plus sur les étendues disponibles pour les attributions de rôles dans la page [vue d'ensemble de l'étendue](#).

Dans ce scénario, vous allez attribuer des autorisations à votre compte d'utilisateur, étendues au compte de stockage, pour suivre le [Principe des priviléges minimum](#). Cette pratique offre aux utilisateurs uniquement les autorisations minimales nécessaires et crée des environnements de production plus sécurisés.

L'exemple suivant affecte le rôle **Contributeur aux données Blob du stockage** à votre compte d'utilisateur, qui fournit à la fois un accès en lecture et en écriture aux données d'objet blob dans votre compte de stockage.

ⓘ Important

Dans la plupart des cas, la propagation de l'attribution de rôle dans Azure peut prendre une ou deux minutes, mais dans de rares cas, cela peut prendre jusqu'à huit minutes. Si vous recevez des erreurs d'authentification lorsque vous exécutez votre code pour la première fois, patientez quelques instants et réessayez.

Azure portal

1. Dans le Portail Azure, recherchez votre compte de stockage à l'aide de la barre de recherche principale ou de la navigation gauche.
2. Dans la page vue d'ensemble du compte de stockage, sélectionnez **Contrôle d'accès (IAM)** dans le menu de gauche.
3. Sur la page **Contrôle d'accès (IAM)**, sélectionnez l'onglet **Attributions de rôles**.
4. Sélectionnez **+ Ajouter** dans le menu supérieur, puis **Ajouter une attribution de rôle** dans le menu déroulant résultant.

The screenshot shows the 'identitymigrationstorage | Access Control (IAM)' page in the Azure portal. The left sidebar has 'Access Control (IAM)' selected. At the top, there's a search bar and several navigation links: 'Add', 'Download role assignments', 'Edit columns', 'Refresh', 'Remove', and 'Got feedback?'. A red box highlights the 'Add role assignment' button. Below it, there's a list of roles: 'Storage account administrator', 'Storage account contributor', 'Storage account owner', 'Storage account reader', 'Storage account user', 'Storage blob data contributor', 'Storage blob data owner', 'Storage blob data reader', 'Storage file data contributor', 'Storage file data owner', 'Storage file data reader', 'Storage queue data contributor', 'Storage queue data owner', 'Storage queue data reader', and 'Storage table data contributor', 'Storage table data owner', 'Storage table data reader'. To the right, there are sections for 'My access', 'Check access', 'Grant access to this resource', 'View deny assignments', and a 'Learn more' link.

5. Utilisez la zone de recherche pour filtrer les résultats sur le rôle souhaité.
Pour cet exemple, recherchez *Contributeur aux données Blob du stockage*, sélectionnez le résultat correspondant, puis choisissez **Suivant**.
6. Sous **Attribuer l'accès à**, sélectionnez **Utilisateur, groupe ou principal de service**, puis sélectionnez + **Sélectionner des membres**.
7. Dans la boîte de dialogue, recherchez votre nom d'utilisateur Microsoft Entra (généralement votre adresse e-mail *utilisateur@domaine*), puis choisissez **Sélectionner** en bas de la boîte de dialogue.
8. Sélectionnez **Vérifier + affecter** pour accéder à la page finale, puis **Vérifier + attribuer** à nouveau pour terminer le processus.

Se connecter et connecter le code d'application à Azure à l'aide de DefaultAzureCredential

Vous pouvez autoriser l'accès aux données dans votre compte de stockage en procédant comme suit :

1. Pour le développement local, vérifiez que vous êtes authentifié avec le même compte Microsoft Entra auquel vous avez attribué le rôle. Vous pouvez vous authentifier au moyen d'outils de développement populaires, comme Azure CLI ou Azure PowerShell. Les outils de développement avec lesquels vous pouvez vous authentifier dépendent de la langue.

Azure CLI

Connectez-vous à Azure via Azure CLI à l'aide de la commande suivante :

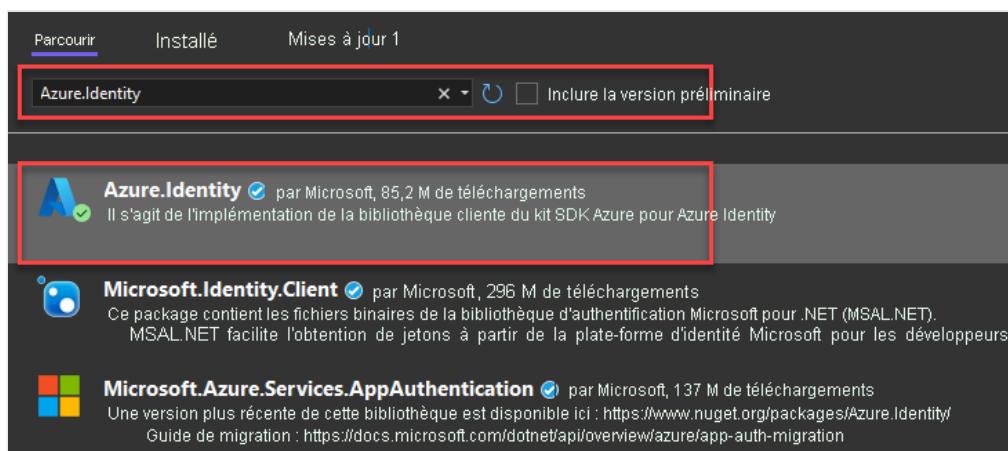
```
Azure CLI
```

```
az login
```

2. Pour utiliser `DefaultAzureCredential`, ajoutez le package **Azure.Identity** à votre application.

Visual Studio

- Dans **Explorateur de solutions**, cliquez avec le bouton droit sur le nœud **Dépendances** de votre projet. Sélectionnez **Gérer les packages NuGet**.
- Dans la fenêtre résultante, recherchez *Azure.Identity*. Sélectionnez le résultat approprié, puis sélectionnez **Installer**.



3. Mettez à jour votre code *Program.cs* pour le faire correspondre à l'exemple suivant : Lorsque le code est exécuté sur votre station de travail locale pendant le développement, il utilise les informations d'identification du développeur de l'outil hiérarchisé auquel vous êtes connecté pour vous authentifier auprès d'Azure, comme Azure CLI ou Visual Studio.

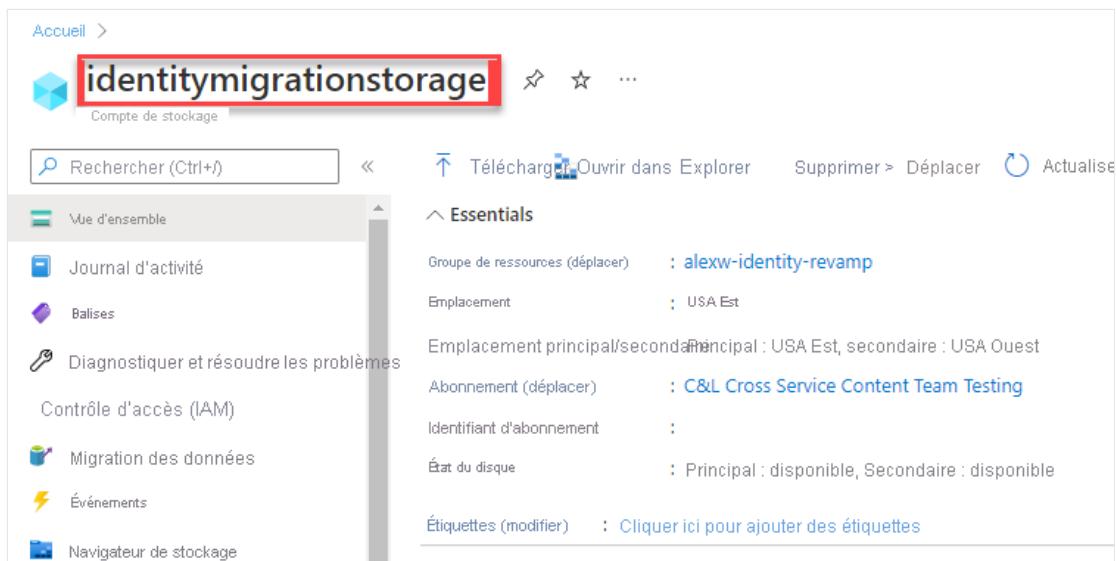
```
C#
```

```
using Azure.Storage.Blobs;
using Azure.Storage.Blobs.Models;
using System;
using System.IO;
using Azure.Identity;

// TODO: Replace <storage-account-name> with your actual storage
account name
```

```
var blobServiceClient = new BlobServiceClient(  
    new Uri("https://<storage-account-  
name>.blob.core.windows.net"),  
    new DefaultAzureCredential());
```

4. Veillez à mettre à jour le nom du compte de stockage dans l'URI de votre `BlobServiceClient`. Le nom du compte de stockage se trouve sur la page vue d'ensemble du Portail Azure.



The screenshot shows the Azure Storage Account overview page. The storage account name 'identitymigrationstorage' is highlighted with a red box. The page displays various details about the storage account, including its resource group ('alexw-identity-revamp'), location ('USA Est'), and subscription ('C&L Cross Service Content Team Testing'). It also shows disk status and label options.

⚠ Notes

Lorsqu'il est déployé sur Azure, ce même code peut être utilisé pour autoriser les demandes adressées à Stockage Azure à partir d'une application s'exécutant dans Azure. Toutefois, vous devez activer l'identité managée sur votre application dans Azure. Configurez ensuite votre compte de stockage pour autoriser cette identité managée à se connecter. Pour obtenir des instructions détaillées sur la configuration de cette connexion entre les services Azure, consultez le didacticiel [d'authentification à partir d'applications hébergées sur Azure](#).

Créez un conteneur.

Créez un nouveau conteneur dans votre compte de stockage en appelant la méthode `CreateBlobContainerAsync` sur l'objet `blobServiceClient`. Dans cet exemple, le code ajoute une valeur GUID au nom du conteneur pour s'assurer qu'il est unique.

Ajoutez le code suivant à la fin du fichier `Program.cs` :

C#

```
// TODO: Replace <storage-account-name> with your actual storage account
name
var blobServiceClient = new BlobServiceClient(
    new Uri("https://<storage-account-name>.blob.core.windows.net"),
    new DefaultAzureCredential());

//Create a unique name for the container
string containerName = "quickstartblobs" + Guid.NewGuid().ToString();

// Create the container and return a container client object
BlobContainerClient containerClient = await
blobServiceClient.CreateBlobContainerAsync(containerName);
```

Pour en savoir plus sur la création d'un conteneur et explorer d'autres exemples de code, consultez [Créer un conteneur d'objets blob avec .NET](#).

Important

Les noms de conteneurs doivent être en minuscules. Pour plus d'informations sur l'affectation de noms aux conteneurs et objets blob, consultez [Affectation de noms et références aux conteneurs, objets blob et métadonnées](#).

Charger un objet blob dans un conteneur

Charger un objet blob dans un conteneur en utilisant [UploadAsync](#). L'exemple de code crée un fichier texte dans le répertoire de *données* local à charger dans le conteneur.

Ajoutez le code suivant à la fin du fichier `Program.cs` :

C#

```
// Create a local file in the ./data/ directory for uploading and
downloading
string localPath = "data";
Directory.CreateDirectory(localPath);
string fileName = "quickstart" + Guid.NewGuid().ToString() + ".txt";
string localFilePath = Path.Combine(localPath, fileName);

// Write text to the file
await File.WriteAllTextAsync(localFilePath, "Hello, World!");

// Get a reference to a blob
BlobClient blobClient = containerClient.GetBlobClient(fileName);

Console.WriteLine("Uploading to Blob storage as blob:\n\t{0}\n",
localFilePath)
```

```
blobClient.Uri);

// Upload data from the local file, overwrite the blob if it already exists
await blobClient.UploadAsync(localFilePath, true);
```

Pour en savoir plus sur le chargement d'objets blob et explorer d'autres exemples de code, consultez [Charger un objet blob avec .NET](#).

Lister les objets blob d'un conteneur

Répertoriez les objets Blob dans le conteneur en appelant la méthode [GetBlobsAsync](#).

Ajoutez le code suivant à la fin du fichier `Program.cs` :

C#

```
Console.WriteLine("Listing blobs...");

// List all blobs in the container
await foreach (BlobItem blobItem in containerClient.GetBlobsAsync())
{
    Console.WriteLine("\t" + blobItem.Name);
}
```

Pour en savoir plus sur les listings d'objets blob et explorer d'autres exemples de code, consultez [Répertorier les objets blob avec .NET](#).

Télécharger un objet blob

Téléchargez l'objet blob que nous avons créé précédemment en appelant la méthode [DownloadToAsync](#). L'exemple de code ajoute la chaîne « DOWNLOADED » au nom de fichier afin que vous puissiez voir les deux fichiers dans votre système de fichiers local.

Ajoutez le code suivant à la fin du fichier `Program.cs` :

C#

```
// Download the blob to a local file
// Append the string "DOWNLOADED" before the .txt extension
// so you can compare the files in the data directory
string downloadFilePath = localFilePath.Replace(".txt", "DOWNLOADED.txt");

Console.WriteLine("\nDownloading blob to\n\t{0}\n", downloadFilePath);

// Download the blob's contents and save it to a file
await blobClient.DownloadToAsync(downloadFilePath);
```

Pour en savoir plus sur le téléchargement d'objets blob et explorer d'autres exemples de code, consultez [Télécharger un objet blob avec .NET](#).

Supprimer un conteneur

Le code suivant nettoie les ressources créées par l'application en supprimant le conteneur en utilisant `DeleteAsync`. L'exemple de code supprime également les fichiers locaux créés par l'application.

L'application s'interrompt pour une entrée de l'utilisateur en appelant `Console.ReadLine` avant de supprimer l'objet blob, le conteneur et les fichiers locaux. C'est l'occasion de vérifier que les ressources ont été créées correctement, avant d'être supprimées.

Ajoutez le code suivant à la fin du fichier `Program.cs` :

```
C#  
  
// Clean up  
Console.WriteLine("Press any key to begin clean up");  
Console.ReadLine();  
  
Console.WriteLine("Deleting blob container...");  
await containerClient.DeleteAsync();  
  
Console.WriteLine("Deleting the local source and downloaded files...");  
File.Delete(localFilePath);  
File.Delete(downloadFilePath);  
  
Console.WriteLine("Done");
```

Pour en savoir plus sur la suppression d'un conteneur et explorer d'autres exemples de code, consultez [Supprimer et restaurer un conteneur d'objets blob avec .NET](#).

Code terminé

Une fois ces étapes effectuées, le code de votre fichier `Program.cs` doit maintenant ressembler à ce qui suit :

Sans mot de passe (recommandé)

```
C#  
  
using Azure.Storage.Blobs;  
using Azure.Storage.Blobs.Models;  
using Azure.Identity;
```

```
// TODO: Replace <storage-account-name> with your actual storage account
// name
var blobServiceClient = new BlobServiceClient(
    new Uri("https://<storage-account-name>.blob.core.windows.net"),
    new DefaultAzureCredential());

//Create a unique name for the container
string containerName = "quickstartblobs" + Guid.NewGuid().ToString();

// Create the container and return a container client object
BlobContainerClient containerClient = await
blobServiceClient.CreateBlobContainerAsync(containerName);

// Create a local file in the ./data/ directory for uploading and
// downloading
string localPath = "data";
Directory.CreateDirectory(localPath);
string fileName = "quickstart" + Guid.NewGuid().ToString() + ".txt";
string localFilePath = Path.Combine(localPath, fileName);

// Write text to the file
await File.WriteAllTextAsync(localFilePath, "Hello, World!");

// Get a reference to a blob
BlobClient blobClient = containerClient.GetBlobClient(fileName);

Console.WriteLine("Uploading to Blob storage as blob:\n\t{0}\n",
blobClient.Uri);

// Upload data from the local file
await blobClient.UploadAsync(localFilePath, true);

Console.WriteLine("Listing blobs...");

// List all blobs in the container
await foreach (BlobItem blobItem in containerClient.GetBlobsAsync())
{
    Console.WriteLine("\t" + blobItem.Name);
}

// Download the blob to a local file
// Append the string "DOWNLOADED" before the .txt extension
// so you can compare the files in the data directory
string downloadFilePath = localFilePath.Replace(".txt",
"DOWNLOADED.txt");

Console.WriteLine("\nDownloading blob to\n\t{0}\n", downloadFilePath);

// Download the blob's contents and save it to a file
await blobClient.DownloadToAsync(downloadFilePath);

// Clean up
Console.Write("Press any key to begin clean up");
Console.ReadLine();
```

```
Console.WriteLine("Deleting blob container...");
await containerClient.DeleteAsync();

Console.WriteLine("Deleting the local source and downloaded files...");
File.Delete(localFilePath);
File.Delete(downloadFilePath);

Console.WriteLine("Done");
```

Exécuter le code

Cette application crée un fichier de test dans votre dossier local *data* et le charge sur Stockage Blob. L'exemple liste ensuite les objets blob du conteneur et télécharge le fichier avec un nouveau nom pour que vous puissiez comparer les deux fichiers.

Si vous utilisez Visual Studio, appuyez sur F5 pour générer et exécuter le code et interagir avec l'application console. Si vous utilisez l'interface CLI .NET, accédez à votre répertoire d'application, puis générez et exécutez l'application.

```
Console
dotnet build
```

```
Console
dotnet run
```

La sortie de l'application est similaire à l'exemple suivant (valeurs GUID omises pour une meilleure lisibilité) :

```
Output
Azure Blob Storage - .NET quickstart sample

Uploading to Blob storage as blob:

https://mystorageacct.blob.core.windows.net/quickstartblobsGUID/quickstartGU
ID.txt

Listing blobs...
    quickstartGUID.txt

Downloading blob to
    ./data/quickstartGUIDDOWNLOADED.txt
```

```
Press any key to begin clean up
Deleting blob container...
Deleting the local source and downloaded files...
Done
```

Avant de commencer le processus de nettoyage, vérifiez la présence des deux fichiers dans votre dossier *data*. Vous pouvez les ouvrir et constater qu'ils sont identiques.

Nettoyer les ressources

Après avoir vérifié les fichiers et terminé le test, appuyez sur la touche **Entrée** pour supprimer les fichiers de test avec le conteneur que vous avez créé dans le compte de stockage. Vous pouvez également utiliser [Azure CLI](#) pour supprimer des ressources.

Étapes suivantes

Dans ce démarrage rapide, vous avez appris à charger, télécharger et répertorier des objets blob avec .NET.

Pour afficher des exemples d'applications de stockage blob, passez à :

Exemples de bibliothèque Stockage Blob Azure pour .NET

- Pour plus d'informations, consultez [Bibliothèques de client Stockage Blob Azure pour .NET](#).
- Pour obtenir des tutoriels, des exemples, des démarrages rapides et autre documentation, consultez [Azure pour les développeurs .NET](#).
- Pour plus d'informations sur .NET, consultez [Démarrer avec .NET en 10 minutes ↗](#).

Démarrage rapide : bibliothèque cliente Stockage File d'attente Azure pour .NET

Article • 29/06/2023

Familiarisez-vous avec la bibliothèque de client Storage File d'attente Azure pour .NET. Le Stockage File d'attente Azure est un service permettant de stocker un grand nombre de messages dans le but de les récupérer et de les traiter plus tard. Suivez les étapes suivantes pour installer le package et essayer un exemple de code pour les tâches de base.

[Documentation de référence sur l'API](#) | [Code source de la bibliothèque](#) | [Package \(NuGet\)](#) | [Exemples](#)

Utilisez la bibliothèque de client Stockage File d'attente Azure pour .NET afin d'effectuer les opérations suivantes :

- Créer une file d'attente
- Ajouter des messages à une file d'attente
- Afficher un aperçu des messages d'une file d'attente
- Mettre à jour un message dans une file d'attente
- Obtention de la longueur de la file d'attente
- Recevoir les messages d'une file d'attente
- Supprimer des messages d'une file d'attente
- Suppression d'une file d'attente

Prérequis

- Abonnement Azure : [créez-en un gratuitement](#)
- Compte de stockage Azure : [créez un compte de stockage](#)
- Dernière version du [SDK .NET](#) pour votre système d'exploitation. Veillez à disposer du Kit de développement logiciel (SDK), et non du runtime.

Configuration

Cette section vous guide tout au long de la préparation d'un projet à utiliser avec la bibliothèque de client Stockage File d'attente Azure pour .NET.

Créer le projet

Créer une application .NET nommée `QueuesQuickstart`.

1. Dans une fenêtre de console (par exemple cmd, PowerShell ou Bash), utilisez la commande `dotnet new` pour créer une application console avec le nom `QueuesQuickstart`. Cette commande crée un projet C# simple « hello world » avec un seul fichier source nommé `Program.cs`.

```
Console
```

```
dotnet new console -n QueuesQuickstart
```

2. Basculez vers le répertoire `QueuesQuickstart` nouvellement créé.

```
Console
```

```
cd QueuesQuickstart
```

Installer les packages

Alors que vous êtes toujours dans le répertoire de l'application, installez le package de la bibliothèque de client Stockage File d'attente Azure pour .NET à l'aide de la commande `dotnet add package`.

```
Console
```

```
dotnet add package Azure.Storage.Queues
```

Le package de bibliothèque de client Azure Identity est également nécessaire pour les connexions sans mot de passe aux services Azure.

```
Console
```

```
dotnet add package Azure.Identity
```

Configurer le framework d'application

1. Ouvrez le projet dans l'éditeur de votre choix
2. Ouvrez le fichier `Program.cs`
3. Mettez à jour le code existant pour le faire correspondre à l'élément suivant :

```
C#
```

```
using Azure;
using Azure.Identity;
using Azure.Storage.Queues;
using Azure.Storage.Queues.Models;
using System;
using System.Threading.Tasks;

Console.WriteLine("Azure Queue Storage client library - .NET quickstart
sample");

// Quickstart code goes here
```

Authentification auprès d'Azure

Les requêtes d'application vers les Services Azure doivent être autorisées. L'utilisation de la classe `DefaultAzureCredential` fournie par la bibliothèque de client Azure Identity est l'approche recommandée pour implémenter des connexions sans mot de passe aux services Azure dans votre code.

Vous pouvez également autoriser directement les requêtes adressées aux services Azure à l'aide de mots de passe, de chaînes de connexion ou d'autres informations d'identification. Toutefois, cette approche doit être utilisée avec prudence. Les développeurs doivent être vigilants pour ne jamais exposer les secrets dans un emplacement non sécurisé. Toute personne ayant accès au mot de passe ou à la clé secrète est en mesure de s'authentifier. `DefaultAzureCredential` offre des avantages améliorés en matière de gestion et de sécurité par rapport à la clé de compte pour autoriser l'authentification sans mot de passe. Les deux options sont illustrées dans l'exemple suivant.

Sans mot de passe (recommandé)

`DefaultAzureCredential` est une classe fournie par la bibliothèque de client Azure Identity pour .NET. Pour en savoir plus sur `DefaultAzureCredential`, consultez la vue d'ensemble de [DefaultAzureCredential](#). `DefaultAzureCredential` prend en charge plusieurs méthodes d'authentification et détermine quelle méthode doit être utilisée au moment de l'exécution. Cette approche permet à votre application d'utiliser différentes méthodes d'authentification dans différents environnements (local ou production) sans implémenter de code spécifique à l'environnement.

Par exemple, votre application peut s'authentifier à l'aide de vos informations d'identification de connexion Visual Studio lors du développement local, puis

utiliser une [identité managée](#) une fois qu'elle a été déployée sur Azure. Aucune modification du code n'est requise pour cette transition.

Lors du développement localement, assurez-vous que le compte d'utilisateur qui accède aux données de file d'attente dispose des autorisations appropriées. Vous aurez besoin du **Contributeur aux données de file d'attente de stockage** pour lire et écrire des données blob. Pour vous attribuer ce rôle, vous aurez besoin du rôle **Administrateur de l'accès utilisateur** ou d'un autre rôle qui inclut l'action **Microsoft.Authorization/roleAssignments/write**. Vous pouvez attribuer des rôles RBAC Azure à un utilisateur à l'aide du Portail Azure, Azure CLI ou Azure PowerShell. Vous pouvez en savoir plus sur les étendues disponibles pour les attributions de rôles dans la page [vue d'ensemble de l'étendue](#).

Dans ce scénario, vous allez attribuer des autorisations à votre compte d'utilisateur, étendues au compte de stockage, pour suivre le [Principe des priviléges minimum](#). Cette pratique offre aux utilisateurs uniquement les autorisations minimales nécessaires et crée des environnements de production plus sécurisés.

L'exemple suivant attribue le rôle **Contributeur aux données file d'attente du stockage** à votre compte d'utilisateur, qui fournit à la fois un accès en lecture et en écriture aux données file d'attente dans votre compte de stockage.

ⓘ Important

Dans la plupart des cas, la propagation de l'attribution de rôle dans Azure peut prendre une ou deux minutes, mais dans de rares cas, cela peut prendre jusqu'à huit minutes. Si vous recevez des erreurs d'authentification lorsque vous exécutez votre code pour la première fois, patientez quelques instants et réessayez.

Azure portal

1. Dans le Portail Azure, recherchez votre compte de stockage à l'aide de la barre de recherche principale ou de la navigation gauche.
2. Dans la page vue d'ensemble du compte de stockage, sélectionnez **Contrôle d'accès (IAM)** dans le menu de gauche.
3. Sur la page **Contrôle d'accès (IAM)**, sélectionnez l'onglet **Attributions de rôles**.

4. Sélectionnez + Ajouter dans le menu supérieur, puis Ajouter une attribution de rôle dans le menu déroulant résultant.

The screenshot shows the 'identitymigrationstorage | Access Control (IAM)' blade in the Azure portal. On the left, a sidebar lists various storage-related sections like Overview, Activity log, Tags, etc., with 'Access Control (IAM)' selected and highlighted with a red box. The main area has a header with '+ Add', 'Download role assignments', 'Edit columns', 'Refresh', 'Remove', and 'Got feedback?' buttons. A dropdown menu is open over the '+ Add' button, with 'Add role assignment' highlighted and also enclosed in a red box. Below this, other options like 'Add co-administrator' are visible. To the right, there are two main sections: 'Grant access to this resource' (with a 'Add role assignment' button) and 'View deny assignments' (with a 'View' button). A search bar at the bottom of the main area says 'Search by name or email address'.

5. Utilisez la zone de recherche pour filtrer les résultats sur le rôle souhaité.

Pour cet exemple, recherchez *Contributeur aux données file d'attente du stockage*, sélectionnez le résultat correspondant, puis choisissez **Suivant**.

6. Sous **Attribuer l'accès à**, sélectionnez **Utilisateur, groupe ou principal de service**, puis sélectionnez + **Sélectionner des membres**.

7. Dans la boîte de dialogue, recherchez votre nom d'utilisateur Microsoft Entra (généralement votre adresse e-mail *user@domain*), puis choisissez **Sélectionner** en bas de la boîte de dialogue.

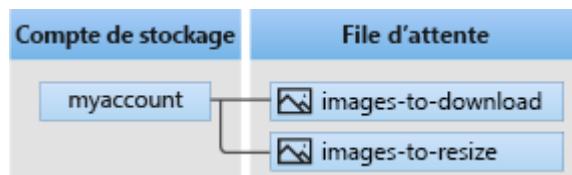
8. Sélectionnez **Vérifier + affecter** pour accéder à la page finale, puis **Vérifier + attribuer** à nouveau pour terminer le processus.

Modèle objet

Stockage File d'attente Azure est un service permettant de stocker un grand nombre de messages. La taille maximale d'un message de file d'attente est de 64 Ko. Une file d'attente peut contenir des millions de messages, dans la limite de la capacité totale d'un compte de stockage. Les files d'attente sont couramment utilisées pour créer un backlog de travail à traiter de façon asynchrone. Le Stockage File d'attente offre trois types de ressources :

- **Compte de stockage** : Tous les accès à Azure Storage passent par un compte de stockage. Pour plus d'informations sur les comptes de stockage, consultez [Vue d'ensemble des comptes de stockage](#).
- **File d'attente** : une file d'attente contient un ensemble de messages. Tous les messages doivent être dans une file d'attente. Notez que le nom de la file d'attente doit être en minuscules. Pour plus d'informations sur l'affectation de noms à des files d'attente, consultez [Affectation de noms pour les files d'attente et les métadonnées](#).
- **Message** : message dans n'importe quel format d'une taille maximale de 64 Ko. Un message peut rester dans la file d'attente pendant un maximum de 7 jours. Pour les versions du 29 juillet 2017 ou ultérieures, la durée de vie maximale peut être n'importe quel nombre positif, ou -1 indiquant que le message n'expire pas. Si ce paramètre est omis, la valeur par défaut de la durée de vie est de sept jours.

Le diagramme suivant montre la relation entre ces ressources.



Utilisez les classes .NET suivantes pour interagir avec ces ressources :

- [QueueServiceClient](#) : `QueueServiceClient` vous permet de gérer toutes les files d'attente de votre compte de stockage.
- [QueueClient](#) : la classe `QueueClient` vous permet de gérer et de manipuler une file d'attente individuelle et ses messages.
- [QueueMessage](#) : la classe `QueueMessage` représente les objets individuels retournés lors de l'appel de [ReceiveMessages](#) dans une file d'attente.

Exemples de code

Ces exemples d'extraits de code vous montrent comment effectuer les actions suivantes avec la bibliothèque de client Stockage File d'attente Azure pour .NET :

- [Autoriser l'accès et créer un objet client](#)
- [Créer une file d'attente](#)
- [Ajouter des messages à une file d'attente](#)
- [Afficher un aperçu des messages d'une file d'attente](#)
- [Mettre à jour un message dans une file d'attente](#)
- [Obtention de la longueur de la file d'attente](#)
- [Recevoir les messages d'une file d'attente](#)

- Supprimer des messages d'une file d'attente
- Supprimer une file d'attente

Sans mot de passe (recommandé)

Autoriser l'accès et créer un objet client

Pour le développement local, vérifiez que vous êtes authentifié avec le même compte Microsoft Entra auquel vous avez attribué le rôle. Vous pouvez vous authentifier au moyen d'outils de développement populaires, comme Azure CLI ou Azure PowerShell. Les outils de développement avec lesquels vous pouvez vous authentifier dépendent de la langue.

Azure CLI

Connectez-vous à Azure via Azure CLI à l'aide de la commande suivante :

Azure CLI

```
az login
```

Une fois authentifié, vous pouvez créer et autoriser un objet `QueueClient` à l'aide de `DefaultAzureCredential` pour accéder aux données de file d'attente dans le compte de stockage. `DefaultAzureCredential` découvre et utilise automatiquement le compte avec lequel vous vous êtes connecté à l'étape précédente.

Pour autoriser à l'aide de `DefaultAzureCredential`, vérifiez que vous avez ajouté le package `Azure.Identity`, comme décrit dans [Installer les packages](#). Veillez également à ajouter une directive `using` pour l'espace de noms `Azure.Identity` dans le fichier `Program.cs` :

C#

```
using Azure.Identity;
```

Ensuite, choisissez un nom pour la file d'attente et créez une instance de la classe `QueueClient`, en utilisant `DefaultAzureCredential` pour l'autorisation. Nous utilisons cet objet client pour créer et interagir avec la ressource de file d'attente dans le compte de stockage.

ⓘ Important

Les noms de file d'attente peuvent contenir uniquement des lettres minuscules, des chiffres et des traits d'union, et doivent commencer par une lettre ou un nombre. Chaque trait d'union doit être précédé et suivi d'un caractère autre qu'un tiret. Le nom doit avoir entre 3 et 63 caractères. Pour plus d'informations, consultez [Affectation de noms pour les files d'attente et les métadonnées](#).

Ajoutez le code suivant à la fin du fichier *Program.cs*. Veillez à remplacer l'espace réservé `<storage-account-name>` par la valeur :

C#

```
// Create a unique name for the queue
// TODO: Replace the <storage-account-name> placeholder
string queueName = "quickstartqueues-" + Guid.NewGuid().ToString();
string storageAccountName = "<storage-account-name>";

// Instantiate a QueueClient to create and interact with the queue
QueueClient queueClient = new QueueClient(
    new
    Uri($"https://{{storageAccountName}}.queue.core.windows.net/{{queueName}}"),
    new DefaultAzureCredential());
```

ⓘ Notes

Les messages que vous envoyez à l'aide de la classe `QueueClient` doivent être dans un format pouvant être inclus dans une requête XML avec encodage UTF-8. Si vous le souhaitez, vous pouvez définir l'option `MessageEncoding` sur `Base64` pour gérer les messages non conformes.

Créer une file d'attente

Avec l'objet `QueueClient`,appelez la méthode `CreateAsync` pour créer la file d'attente dans votre compte de stockage.

Ajoutez ce code à la fin de la méthode *Program.cs* :

C#

```
Console.WriteLine($"Creating queue: {queueName}");

// Create the queue
await queueClient.CreateAsync();
```

Ajouter des messages à une file d'attente

L'extrait de code suivant ajoute de façon asynchrone des messages à la file d'attente en appelant la méthode [SendMessageAsync](#). Il enregistre également un [SendReceipt](#) retourné à partir d'un appel de [SendMessageAsync](#). La réception est utilisée pour mettre à jour le message ultérieurement dans le programme.

Ajoutez ce code à la fin du fichier *Program.cs* :

```
C#

Console.WriteLine("\nAdding messages to the queue...");

// Send several messages to the queue
await queueClient.SendMessageAsync("First message");
await queueClient.SendMessageAsync("Second message");

// Save the receipt so we can update this message later
SendReceipt receipt = await queueClient.SendMessageAsync("Third message");
```

Afficher un aperçu des messages d'une file d'attente

Affichez un aperçu des messages de la file d'attente en appelant la méthode [PeekMessagesAsync](#). Cette méthode récupère un ou plusieurs messages du début de la file d'attente, mais ne modifie pas la visibilité du message.

Ajoutez ce code à la fin du fichier *Program.cs* :

```
C#

Console.WriteLine("\nPeek at the messages in the queue...");

// Peek at messages in the queue
PeekedMessage[] peekedMessages = await
queueClient.PeekMessagesAsync(maxMessages: 10);

foreach (PeekedMessage peekedMessage in peekedMessages)
{
    // Display the message
```

```
        Console.WriteLine($"Message: {peekedMessage.MessageText}");
    }
```

Mettre à jour un message dans une file d'attente

Mettez à jour le contenu d'un message en appelant la méthode [UpdateMessageAsync](#). Cette méthode peut changer le contenu et le délai d'expiration de la visibilité d'un message. Le contenu du message doit être une chaîne encodée en UTF-8 d'une taille maximale de 64 Ko. Avec le nouveau contenu du message, transmettez les valeurs du `SendReceipt` qui a été enregistré dans le code. Les valeurs `SendReceipt` identifient le message à mettre à jour.

C#

```
Console.WriteLine("\nUpdating the third message in the queue...");

// Update a message using the saved receipt from sending the message
await queueClient.UpdateMessageAsync(receipt.MessageId, receipt.PopReceipt,
"Third message has been updated");
```

Obtention de la longueur de la file d'attente

Vous pouvez obtenir une estimation du nombre de messages dans une file d'attente. La méthode [GetProperties](#) retourne les propriétés de file d'attente, y compris le nombre de messages. La propriété [ApproximateMessagesCount](#) contient le nombre approximatif de messages dans la file d'attente. Ce nombre n'est pas inférieur au nombre de messages réel dans la file d'attente, mais il peut être supérieur.

Ajoutez ce code à la fin du fichier *Program.cs* :

C#

```
QueueProperties properties = queueClient.GetProperties();

// Retrieve the cached approximate message count
int cachedMessagesCount = properties.ApproximateMessagesCount;

// Display number of messages
Console.WriteLine($"Number of messages in queue: {cachedMessagesCount}");
```

Réception des messages d'une file d'attente

Téléchargez les messages ajoutés en appelant la méthode [ReceiveMessagesAsync](#).

Ajoutez ce code à la fin du fichier *Program.cs* :

C#

```
Console.WriteLine("\nReceiving messages from the queue...");  
  
// Get messages from the queue  
QueueMessage[] messages = await  
queueClient.ReceiveMessagesAsync(maxMessages: 10);
```

Vous pouvez éventuellement spécifier une valeur pour `maxMessages`, qui correspond au nombre de messages à récupérer dans la file d'attente. La valeur par défaut est de 1 message et la valeur maximale est de 32 messages. Vous pouvez également spécifier une valeur pour `visibilityTimeout`, qui masque les messages aux autres opérations pendant la période d'expiration. La valeur par défaut est 30 secondes.

Supprimer des messages d'une file d'attente

Supprimez les messages de la file d'attente une fois qu'ils ont été traités. Dans ce cas, le traitement affiche simplement le message sur la console.

Avant de traiter et de supprimer les messages, l'application s'interrompt dans l'attente d'une entrée de l'utilisateur en appelant `Console.ReadLine`. Vérifiez dans votre [portail Azure](#) que les ressources ont été créées correctement avant d'être supprimées. Les messages qui ne sont pas supprimés explicitement redeviennent visibles dans la file d'attente et peuvent éventuellement être de nouveau traités.

Ajoutez ce code à la fin du fichier *Program.cs* :

C#

```
Console.WriteLine("\nPress Enter key to 'process' messages and delete them  
from the queue...");  
Console.ReadLine();  
  
// Process and delete messages from the queue  
foreach (QueueMessage message in messages)  
{  
    // "Process" the message  
    Console.WriteLine($"Message: {message.MessageText}");  
  
    // Let the service know we're finished with  
    // the message and it can be safely deleted.  
    await queueClient.DeleteMessageAsync(message.MessageId,  
    message.PopReceipt);  
}
```

Suppression d'une file d'attente

Le code suivant nettoie les ressources créées par l'application en supprimant la file d'attente avec la méthode [DeleteAsync](#).

Ajoutez ce code à la fin du fichier *Program.cs* :

C#

```
Console.WriteLine("\nPress Enter key to delete the queue...");  
Console.ReadLine();  
  
// Clean up  
Console.WriteLine($"Deleting queue: {queueClient.Name}");  
await queueClient.DeleteAsync();  
  
Console.WriteLine("Done");
```

Exécuter le code

Cette application crée trois messages et les ajoute à une file d'attente Azure. Le code liste les messages dans la file d'attente, puis les récupère et les supprime avant de supprimer la file d'attente.

Dans la fenêtre de votre console, accédez au répertoire de l'application, puis générez et exécutez l'application.

Console

```
dotnet build
```

Console

```
dotnet run
```

La sortie de l'application ressemble à l'exemple suivant :

Sortie

```
Azure Queue Storage client library - .NET quickstart sample
```

```
Creating queue: quickstartqueues-5c72da2c-30cc-4f09-b05c-a95d9da52af2
```

```
Adding messages to the queue...
```

```
Peek at the messages in the queue...
Message: First message
Message: Second message
Message: Third message

Updating the third message in the queue...

Receiving messages from the queue...

Press Enter key to 'process' messages and delete them from the queue...

Message: First message
Message: Second message
Message: Third message has been updated

Press Enter key to delete the queue...

Deleting queue: quickstartqueues-5c72da2c-30cc-4f09-b05c-a95d9da52af2
Done
```

Quand l'application s'interrompt avant de recevoir des messages, vérifiez votre compte de stockage dans le [portail Azure](#). Vérifiez que les messages se trouvent dans la file d'attente.

Appuyez sur la touche `Enter` pour recevoir et supprimer les messages. Quand vous y êtes invité, réappuyez sur la touche `Enter` pour supprimer la file d'attente et terminer la démonstration.

Étapes suivantes

Dans ce guide de démarrage rapide, vous avez appris à créer une file d'attente et à y ajouter des messages à l'aide de code .NET asynchrone. Ensuite, vous avez appris à afficher un aperçu des messages, à les récupérer et à les supprimer. Enfin, vous avez appris à supprimer une file d'attente de messages.

Pour obtenir des tutoriels, des exemples, des guides de démarrage rapide et autres documentations, visitez :

Azure pour les développeurs .NET et .NET Core

- Pour obtenir des exemples de code associés utilisant des Kits de développement logiciel (SDK) .NET version 11.x dépréciés, consultez l'article [Exemples de code utilisant .NET version 11.x](#).
- Pour plus d'informations, consultez les [bibliothèques Stockage Azure pour .NET](#).
- Pour d'autres exemples d'applications Stockage File d'attente Azure, consultez [Exemples de bibliothèques de client Stockage File d'attente Azure pour .NET](#).

- Pour en savoir plus sur .NET Core, consultez [Prise en main de .NET en 10 minutes](#).

Tutoriel : Utiliser des connexions basées sur l'identité plutôt que des secrets avec des déclencheurs et des liaisons

Article • 01/06/2023

Ce tutoriel vous montre comment configurer Azure Functions pour vous connecter à des files d'attente Azure Service Bus à l'aide d'identités managées plutôt que de secrets stockés dans les paramètres de l'application de fonction. Ce tutoriel est la suite du tutoriel [Créer une application de fonction sans secrets de stockage par défaut dans sa définition](#). Pour en savoir plus sur les connexions basées sur l'identité, consultez [Configurer une connexion basée sur l'identité](#).

Même si les procédures indiquées fonctionnent généralement pour tous les langages, ce tutoriel prend plus particulièrement en charge les fonctions de bibliothèque de classes C# sur Windows.

Ce didacticiel vous montre comment effectuer les opérations suivantes :

- ✓ Créez un espace de noms et une file d'attente Service Bus.
- ✓ Configurer votre application de fonction avec une identité managée
- ✓ Créer une attribution de rôle accordant à cette identité l'autorisation de lire à partir de la file d'attente Service Bus
- ✓ Créer et déployer une application de fonction avec un déclencheur Service Bus.
- ✓ Vérifier votre connexion basée sur l'identité à Service Bus

Configuration requise

Suivez le tutoriel précédent : [Créer une application de fonction avec des connexions basées sur l'identité](#).

Créer un bus de service et une file d'attente

1. Dans le [portail Azure](#), choisissez **Créer une ressource (+)**.
2. Dans la page **Créer une ressource**, sélectionnez **Intégration>Service Bus**.
3. Dans la page **Informations de base**, utilisez le tableau suivant pour configurer les paramètres d'espace de noms Service Bus. Utilisez les valeurs par défaut pour les options restantes.

Option	Valeur suggérée	Description
Abonnement	Votre abonnement	Abonnement sous lequel vos ressources sont créées.
Groupe de ressources	myResourceGroup	Le groupe de ressources que vous avez créé avec votre application de fonction.
Nom de l'espace de noms	Nom globalement unique	Espace de noms de votre instance à partir duquel déclencher votre fonction. Étant donné que l'espace de noms est accessible publiquement, vous devez utiliser un nom qui est globalement unique dans Azure. Le nom doit également comporter entre 6 et 50 caractères, contenir uniquement des caractères alphanumériques et des tirets, et ne pas commencer par un chiffre.

Option	Valeur suggérée	Description
Emplacement	myFunctionRegion	La région dans laquelle vous avez créé votre application de fonction.
Niveau tarifaire	De base	Niveau Service Bus de base.

4. Sélectionnez **Revoir + créer**. Une fois la validation terminée, sélectionnez **Créer**.
5. Une fois le déploiement terminé, sélectionnez **Accéder à la ressource**.
6. Dans votre espace de noms Service Bus, sélectionnez **+ File d'attente** pour ajouter une file d'attente.
7. Tapez `myinputqueue` comme nom de la nouvelle file d'attente, puis sélectionnez **Créer**.

Maintenant que vous avez une file d'attente, vous allez ajouter une attribution de rôle à l'identité managée de votre espace de noms Service Bus.

Configurer votre déclencheur Service Bus avec une identité managée

Pour utiliser des déclencheurs Service Bus avec des connexions basées sur l'identité, vous devez ajouter l'attribution du rôle **Récepteur de données Azure Service Bus** à l'identité managée dans votre application de fonction. Ce rôle est requis lors de l'utilisation d'identités managées à déclencher à partir de votre espace de noms Service Bus. Vous pouvez également ajouter votre propre compte à ce rôle, ce qui permet de se connecter à l'espace de noms Service Bus au cours du test local.

ⓘ Notes

Les exigences de rôle pour l'utilisation de connexions basées sur l'identité varient en fonction du service et de la façon dont vous vous y connectez. Les besoins varient selon les déclencheurs, les liaisons d'entrée et les liaisons de sortie. Pour plus d'informations sur des exigences de rôle spécifiques, reportez-vous à la documentation du service relative aux déclencheurs et aux liaisons.

1. Dans l'espace de noms Service Bus que vous venez de créer, sélectionnez **Contrôle d'accès (IAM)**. C'est là que vous pouvez voir et configurer qui a accès à la ressource.
2. Cliquez sur **Ajouter**, puis sélectionnez **Ajouter une attribution de rôle**.
3. Recherchez **Récepteur de données Azure Service Bus**, sélectionnez-le, puis cliquez sur **Suivant**.
4. Dans l'onglet **Membres**, sous **Attribuer l'accès à**, choisissez **Identité managée**
5. Cliquez sur **Sélectionner des membres** pour ouvrir le volet **Sélectionner des identités managées**.
6. Vérifiez que l'**Abonnement** est celui dans lequel vous avez créé les ressources précédemment.
7. Dans le sélecteur **Identité managée**, choisissez **Application de fonction** dans la catégorie **Identité managée affectée par le système**. Un nombre entre parenthèses peut être affiché en regard de l'étiquette « Application de fonction », indiquant le nombre d'applications de l'abonnement ayant des identités affectées par le système.

8. Votre application doit apparaître dans une liste sous les champs d'entrée. Si vous ne la voyez pas, vous pouvez utiliser la zone **Sélectionner** pour filtrer les résultats avec le nom de votre application.
9. Cliquez sur votre application. Elle doit descendre dans la section **Membres sélectionnés**. Cliquez sur **Sélectionner**.
10. De retour dans l'écran **Ajouter une attribution de rôle**, cliquez sur **Vérifier + attribuer**. Vérifiez la configuration, puis cliquez sur **Vérifier + attribuer**.

Vous avez accordé à votre application de fonction l'accès à l'espace de noms Service Bus à l'aide d'identités managées.

Se connecter à Service Bus dans votre application de fonction

1. Dans le portail, recherchez l'application de fonction que vous avez créée dans le [tutoriel précédent](#) ou accédez-y dans la page **Function App**.
2. Dans votre application de fonction, sélectionnez **Configuration** sous **Paramètres**.
3. Dans **Paramètres d'application**, sélectionnez **+ Nouveau paramètre d'application** pour créer le paramètre dans le tableau suivant.

Nom	Valeur	Description
ServiceBusConnection_fullyQualifiedNamespace	<SERVICE_BUS_NAMESPACE>.servicebus.windows.net	Ce paramètre connecte votre application de fonction au bus des services. Utilisez une connexion basée sur l'identité plutôt que des secrets.

4. Une fois les deux paramètres créés, sélectionnez **Enregistrer>Confirmer**.

ⓘ Notes

Lorsque vous utilisez **Azure App Configuration** ou **Key Vault** pour fournir des paramètres pour les connexions d'identité managée, les noms de paramètres doivent utiliser un séparateur de clé valide tel que `:` ou `/` à la place de `_` pour s'assurer que les noms sont résolus correctement.

Par exemple : `ServiceBusConnection:fullyQualifiedNamespace`.

Maintenant que vous avez préparé l'application de fonction à se connecter à l'espace de noms Service Bus à l'aide d'une identité managée, vous pouvez ajouter une nouvelle fonction qui utilise un déclencheur

Service Bus à votre projet local.

Ajouter une fonction déclenchée par Service Bus

1. Exécutez la commande `func init`, de la façon suivante, pour créer un projet Functions dans un dossier nommé LocalFunctionProj avec le runtime spécifié :

```
C#
```

```
func init LocalFunctionProj --dotnet
```

2. Accédez au dossier du projet :

```
Console
```

```
cd LocalFunctionProj
```

3. Dans le dossier racine du projet, exécutez les commandes suivantes :

```
command
```

```
dotnet add package Microsoft.Azure.WebJobs.Extensions.ServiceBus --version 5.2.0
```

Cela remplace la version par défaut du package d'extension Service Bus par une version qui prend en charge les identités managées.

4. Exécutez la commande suivante pour ajouter une fonction déclenchée par Service Bus au projet :

```
C#
```

```
func new --name ServiceBusTrigger --template ServiceBusQueueTrigger
```

Cela ajoute le code pour un nouveau déclencheur Service Bus et une référence au package d'extension. Vous devez ajouter un paramètre de connexion de l'espace de noms Service Bus pour ce déclencheur.

5. Ouvrez le nouveau fichier de projet ServiceBusTrigger.cs et remplacez la classe `ServiceBusTrigger` par le code suivant :

```
C#
```

```
public static class ServiceBusTrigger
{
    [FunctionName("ServiceBusTrigger")]
    public static void Run([ServiceBusTrigger("myinputqueue",
        Connection = "ServiceBusConnection")]string myQueueItem, ILogger log)
    {
        log.LogInformation($"C# ServiceBus queue trigger function processed message:
{myQueueItem}");
    }
}
```

Cet exemple de code met à jour le nom de la file d'attente de façon à utiliser `myinputqueue`, qui est le même nom que celui de la file d'attente que vous avez créée précédemment. Il définit également le nom de la connexion Service Bus sur `ServiceBusConnection`. Il s'agit de l'espace de noms Service Bus utilisé par la connexion basée sur l'identité `ServiceBusConnection_fullyQualifiedNamespace` que vous avez configuré dans le portail.

① Notes

Si vous essayez d'exécuter vos fonctions maintenant à l'aide de `func start`, une erreur s'affiche. Cela est dû au fait que vous n'avez pas de connexion basée sur l'identité définie localement. Si vous souhaitez exécuter votre fonction localement, définissez le paramètre d'application `ServiceBusConnection_fullyQualifiedNamespace` dans `local.settings.json` comme vous l'avez fait dans la section précédente. En outre, vous devez attribuer le rôle à votre identité de développeur. Pour plus d'informations, consultez la documentation sur le développement local avec des connexions basées sur l'identité.

① Notes

Lorsque vous utilisez Azure App Configuration ou Key Vault pour fournir des paramètres pour les connexions d'identité managée, les noms de paramètres doivent utiliser un séparateur de clé valide tel que `:` ou `/` à la place de `_` pour s'assurer que les noms sont résolus correctement.

Par exemple : `ServiceBusConnection:fullyQualifiedNamespace`.

Publier le projet mis à jour

1. Exécutez la commande suivante afin de générer localement les fichiers nécessaires pour le package de déploiement :

Console

```
dotnet publish --configuration Release
```

2. Accédez au sous-dossier `\bin\Release\netcoreapp3.1\publish` et créez un fichier .zip à partir de son contenu.

3. Publiez le fichier .zip. Pour cela, exécutez la commande suivante, en remplaçant les paramètres

`FUNCTION_APP_NAME`, `RESOURCE_GROUP_NAME` et `PATH_TO_ZIP`, selon le cas :

Azure CLI

```
az functionapp deploy -n FUNCTION_APP_NAME -g RESOURCE_GROUP_NAME --src-path PATH_TO_ZIP
```

Maintenant que vous avez mis à jour l'application de fonction avec le nouveau déclencheur, vous pouvez vérifier qu'elle fonctionne avec l'identité.

Valider vos changements

1. Dans le portail, recherchez **Application Insights**, puis sélectionnez **Application Insights** sous **Services**.
2. Dans **Application Insights**, naviguez jusqu'à votre instance nommée ou recherchez-la.
3. Dans votre instance, sélectionnez **Métriques en temps réel** sous **Examiner**.
4. Laissez l'onglet précédent ouvert, puis ouvrez le portail Azure dans un nouvel onglet. Dans ce nouvel onglet, accédez à votre espace de noms Service Bus, puis sélectionnez **Files d'attente** dans le panneau de gauche.
5. Sélectionnez votre file d'attente nommée `myinputqueue`.
6. Sélectionnez **Service Bus Explorer** dans le panneau de gauche.
7. Envoyez un message de test.
8. Sélectionnez votre onglet **Métriques en temps réel**, puis regardez l'exécution de la file d'attente Service Bus.

Félicitations ! Vous avez correctement configuré votre déclencheur de file d'attente Service Bus avec une identité managée.

Nettoyer les ressources

Au cours des étapes précédentes, vous avez créé des ressources Azure au sein d'un groupe de ressources. Si vous ne pensez pas avoir besoin de ces ressources à l'avenir, vous pouvez les supprimer en supprimant le groupe de ressources.

Dans le menu ou la page d'accueil du portail Azure, sélectionnez **Groupes de ressources**. Ensuite, dans la page **Groupes de ressources**, sélectionnez **myResourceGroup**.

Dans la page **myResourceGroup**, assurez-vous que les ressources répertoriées sont bien celles que vous souhaitez supprimer.

Sélectionnez **Supprimer le groupe de ressources**, tapez **myResourceGroup** dans la zone de texte pour confirmer, puis sélectionnez **Supprimer**.

Étapes suivantes

Dans ce tutoriel, vous avez créé une application de fonction avec des connexions basées sur l'identité.

Utilisez les liens suivants pour en savoir plus sur Azure Functions avec des connexions basées sur l'identité :

- [Identité managée dans Azure Functions](#)
- [Connexions basées sur l'identité dans Azure Functions](#)
- [Documentation sur les fonctions pour le développement local](#)

Tutoriel : Utiliser une identité managée pour connecter Key Vault à une application web Azure dans .NET

Article • 25/03/2023

Azure Key Vault fournit un moyen de stocker des informations d'identification et d'autres secrets avec une sécurité renforcée. Mais votre code a besoin de s'authentifier auprès de Key Vault pour les récupérer. La [vue d'ensemble des identités managées pour les ressources Azure](#) vous aide à résoudre ce problème en fournissant automatiquement aux services Azure une identité managée dans Microsoft Entra ID. Vous pouvez utiliser cette identité pour vous authentifier sur n'importe quel service prenant en charge l'authentification Microsoft Entra, y compris Key Vault, sans avoir à afficher les informations d'identification dans votre code.

Dans ce tutoriel, vous allez créer et déployer une application web Azure dans [Azure App Service](#). Vous allez utiliser une identité managée pour authentifier votre application web Azure auprès d'un coffre de clés Azure à l'aide de la [bibliothèque de client de secrets Azure Key Vault pour .NET](#) et d'[Azure CLI](#). Les mêmes principes de base s'appliquent quand vous utilisez le langage de développement de votre choix, Azure PowerShell et/ou le portail Azure.

Pour plus d'informations sur les applications web et le déploiement du service Azure App présentés dans ce tutoriel, consultez :

- [Vue d'ensemble d'App Service](#)
- [Créer une application web ASP.NET Core dans Azure App Service](#)
- [Déploiement Git local vers Azure App Service](#)

Prérequis

Pour suivre ce didacticiel, vous avez besoin des éléments suivants :

- Un abonnement Azure. [Créez-en un gratuitement](#).
- Le [SDK .NET Core 3.1 \(ou version ultérieure\)](#).
- Une installation [Git](#) version 2.28.0 ou ultérieure.
- [Azure CLI](#) ou [Azure PowerShell](#).
- [Azure Key Vault](#). Vous pouvez créer un coffre de clés à l'aide du [portail Azure](#), d'[Azure CLI](#) ou d'[Azure PowerShell](#).

- Un [secret](#) de coffre de clés. Vous pouvez créer un secret avec le [portail Azure](#), [PowerShell](#) ou [Azure CLI](#).

Si votre application web est déjà déployée dans Azure App Service, vous pouvez passer directement aux sections [Configurer l'accès de l'application web à un coffre de clés](#) et [Modifier le code de l'application web](#).

Créer une application .NET Core

Cette étape consiste à configurer le projet .NET Core local.

Dans une fenêtre de terminal sur votre machine, créez un répertoire nommé `akvwebapp` et faites-en le répertoire actuel.

Bash

```
mkdir akvwebapp  
cd akvwebapp
```

Créez une application .NET Core à l'aide de la commande [dotnet new web](#) :

Bash

```
dotnet new web
```

Exécutez l'application localement pour voir à quoi elle doit ressembler quand vous la déployez sur Azure :

Bash

```
dotnet run
```

Dans un navigateur web, accédez à l'application à l'adresse `http://localhost:5000`.

Vous voyez apparaître sur la page le message « Hello World » de l'exemple d'application.

Pour plus d'informations sur la création d'applications web pour Azure, consultez [Créer une application web ASP.NET Core dans Azure App Service](#).

Déploiement de l'application dans Azure

Dans cette étape, vous allez déployer votre application .NET Core sur Azure App Service à l'aide de Git local. Pour plus d'informations sur la création et le déploiement d'applications, consultez [Créer une application web ASP.NET Core dans Azure](#).

Configurer le déploiement Git local

Dans la fenêtre de terminal, sélectionnez **Ctrl+C** pour fermer le serveur web. Initialisez un dépôt Git pour le projet .NET Core :

Bash

```
git init --initial-branch=main  
git add .  
git commit -m "first commit"
```

Vous pouvez utiliser le protocole FTP et Git local pour déployer une application web Azure en faisant appel à un *utilisateur de déploiement*. Une fois que vous avez créé votre utilisateur de déploiement, vous pouvez l'utiliser pour tous vos déploiements Azure. Votre nom d'utilisateur et votre mot de passe de déploiement au niveau du compte sont différents de vos informations d'identification de l'abonnement Azure.

Pour configurer l'utilisateur de déploiement, exécutez la commande [az webapp deployment user set](#). Choisissez un nom d'utilisateur et un mot de passe conformes à ces instructions :

- Le nom d'utilisateur doit être unique au sein de Azure. Pour les envois (push) Git locaux, il ne peut pas contenir d'arobase (@).
- Le mot de passe doit comporter au moins huit caractères et inclure deux des trois éléments suivants : lettres, chiffres et symboles.

Azure CLI

```
az webapp deployment user set --user-name "<username>" --password "<password>"
```

La sortie JSON affiche le mot de passe comme étant `null`. Si vous obtenez une erreur `'Conflict'. Details: 409`, modifiez le nom d'utilisateur. Si vous obtenez une erreur `'Bad Request'. Details: 400`, utilisez un mot de passe plus fort.

Enregistrez votre nom d'utilisateur et votre mot de passe afin de pouvoir les utiliser pour déployer vos applications web.

Créer un groupe de ressources

Un groupe de ressources est un conteneur logique dans lequel vous déployez et gérez des ressources Azure. Créez un groupe de ressources pour y mettre à la fois votre coffre de clés et votre application web à l'aide de la commande [az group create](#) :

Azure CLI

```
az group create --name "myResourceGroup" -l "EastUS"
```

Créer un plan App Service

Créez un [plan App Service](#) avec la commande [az appservice plan create](#) d'Azure CLI. Cet exemple crée un plan App Service nommé `myAppServicePlan` dans le niveau tarifaire `FREE` :

Azure CLI

```
az appservice plan create --name myAppServicePlan --resource-group myResourceGroup --sku FREE
```

Quand le plan App Service est créé, Azure CLI affiche des informations similaires à celles que vous voyez ici :

```
{
  "adminSiteName": null,
  "appServicePlanName": "myAppServicePlan",
  "geoRegion": "West Europe",
  "hostingEnvironmentProfile": null,
  "id": "/subscriptions/0000-0000-0000-0000/resourceGroups/myResourceGroup/providers/Microsoft.Web/serverfarms/myAppServicePlan",
  "kind": "app",
  "location": "West Europe",
  "maximumNumberOfWorkers": 1,
  "name": "myAppServicePlan",
  < JSON data removed for brevity. >
  "targetWorkerSizeId": 0,
  "type": "Microsoft.Web/serverfarms",
  "workerTierName": null
}
```

Pour plus d'informations, consultez [Gérer un plan App Service dans Azure](#).

Créer une application web

Créez une [application web Azure](#) dans le plan App Service `myAppServicePlan`.

ⓘ Important

Comme un coffre de clés, une application web Azure doit porter un nom unique. Remplacez `<your-webapp-name>` par le nom de votre application web dans les exemples suivants.

Azure CLI

```
az webapp create --resource-group "myResourceGroup" --plan  
"myAppServicePlan" --name "<your-webapp-name>" --deployment-local-git
```

Quand l'application web est créée, Azure CLI présente une sortie similaire à celle que vous voyez ici :

```
Local git is configured with url of 'https://<username>@<your-webapp-  
name>.scm.azurewebsites.net/<your-webapp-name>.git'  
{  
    "availabilityState": "Normal",  
    "clientAffinityEnabled": true,  
    "clientCertEnabled": false,  
    "clientCertExclusionPaths": null,  
    "cloningInfo": null,  
    "containerSize": 0,  
    "dailyMemoryTimeQuota": 0,  
    "defaultHostName": "<your-webapp-name>.azurewebsites.net",  
    "deploymentLocalGitUrl": "https://<username>@<your-webapp-  
name>.scm.azurewebsites.net/<your-webapp-name>.git",  
    "enabled": true,  
    < JSON data removed for brevity. >  
}
```

L'URL du Git distant est indiquée dans la propriété `deploymentLocalGitUrl`, au format `https://<username>@<your-webapp-name>.scm.azurewebsites.net/<your-webapp-name>.git`. Enregistrez cette URL. Vous en aurez besoin ultérieurement.

Configurez maintenant votre application web pour un déploiement à partir de la branche `main` :

Azure CLI

```
az webapp config appsettings set -g MyResourceGroup --name "<your-webapp-name>" --settings deployment_branch=main
```

Accédez à votre nouvelle application en utilisant la commande suivante. Remplacez `<your-webapp-name>` par le nom de votre application.

Bash

```
https://<your-webapp-name>.azurewebsites.net
```

Vous allez voir la page web par défaut d'une nouvelle application web Azure.

Déployer votre application locale

De retour dans la fenêtre de terminal locale, ajoutez un dépôt distant Azure dans votre dépôt Git local. Dans la commande suivante, remplacez `<deploymentLocalGitUrl-from-create-step>` par l'URL du Git distant que vous avez enregistré dans la section [Créer une application web](#).

Bash

```
git remote add azure <deploymentLocalGitUrl-from-create-step>
```

Utilisez la commande suivante pour effectuer un envoi (push) au dépôt distant Azure afin de déployer votre application. Quand le gestionnaire d'informations d'identification de Git vous invite à entrer des informations d'identification, utilisez celles que vous avez dans la section [Configurer le déploiement Git local](#).

Bash

```
git push azure main
```

Cette commande peut prendre quelques minutes pour s'exécuter. Pendant son exécution, elle présente des informations semblables à celles que vous voyez ici :

```
Enumerating objects: 5, done.  
Counting objects: 100% (5/5), done.  
Compressing objects: 100% (3/3), done.  
Writing objects: 100% (3/3), 285 bytes | 95.00 KiB/s, done.  
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0  
remote: Deploy Async  
remote: Updating branch 'main'.  
remote: Updating submodules.
```

```
remote: Preparing deployment for commit id 'd6b54472f7'.
remote: Repository path is /home/site/repository
remote: Running oryx build...
remote: Build orchestrated by Microsoft Oryx,
https://github.com/Microsoft/Oryx
remote: You can report issues at https://github.com/Microsoft/Oryx/issues
remote:
remote: Oryx Version      : 0.2.20200114.13, Commit:
204922f30f8e8d41f5241b8c218425ef89106d1d, ReleaseTagName: 20200114.13
remote: Build Operation ID: |imoMY2y77/s=.40ca2a87_
remote: Repository Commit : d6b54472f7e8e9fd885ffafaa64522e74cf370e1
.
.
.
remote: Deployment successful.
remote: Deployment Logs : 'https://<your-webapp-
name>.scm.azurewebsites.net/newui/jsonviewer?
view_url=/api/deployments/d6b54472f7e8e9fd885ffafaa64522e74cf370e1/log'
To https://<your-webapp-name>.scm.azurewebsites.net:443/<your-webapp-
name>.git
d87e6ca..d6b5447 main -> main
```

Accédez à (ou actualisez) l'application déployée à l'aide de votre navigateur web :

Bash

```
http://<your-webapp-name>.azurewebsites.net
```

Vous verrez le message « Hello World » que vous avez vu précédemment lors de votre visite de <http://localhost:5000>.

Pour plus d'informations sur le déploiement d'applications web à l'aide de Git, consultez [Déploiement Git local vers Azure App Service](#).

Configurer l'application web pour se connecter au coffre de clés

Dans cette section, vous allez configurer l'accès web au coffre de clés et mettre à jour le code de votre application pour récupérer un secret auprès de Key Vault.

Créer et attribuer une identité gérée

Dans ce tutoriel, nous allons utiliser une [identité managée](#) pour l'authentification auprès de Key Vault. Une identité managée gère automatiquement les informations d'identification de l'application.

Dans Azure CLI, pour créer l'identité de l'application, exécutez la commande [az webapp identity assign](#) :

Azure CLI

```
az webapp identity assign --name "<your-webapp-name>" --resource-group "myResourceGroup"
```

La commande retourne cet extrait de code JSON :

JSON

```
{  
  "principalId": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",  
  "tenantId": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",  
  "type": "SystemAssigned"  
}
```

Pour permettre à votre application web d'effectuer des opérations **get** et **list** sur votre coffre de clés, passez `principalId` à la commande Azure CLI [az keyvault set-policy](#) :

Azure CLI

```
az keyvault set-policy --name "<your-keyvault-name>" --object-id "<principalId>" --secret-permissions get list
```

Vous pouvez également affecter des stratégies d'accès à l'aide du [portail Azure](#) ou de [PowerShell](#).

Modifier l'application pour accéder à votre coffre de clés

Dans ce tutoriel, vous allez utiliser la [bibliothèque de client de secrets Azure Key Vault](#) à des fins de démonstration. Vous pouvez également utiliser la [bibliothèque de client de certificats Azure Key Vault](#) ou la [bibliothèque de client de clés Azure Key Vault](#).

Installer les packages

Dans la fenêtre de terminal, installez la bibliothèque de client de secrets Azure Key Vault pour .NET et les packages de la bibliothèque de client d'identités Azure :

Console

```
dotnet add package Azure.Identity
```

```
dotnet add package Azure.Security.KeyVault.Secrets
```

Mettez à jour le code

Recherchez et ouvrez le fichier Startup.cs pour .NET 5.0 ou une version antérieure, ou Program.cs pour .NET 6.0 dans votre projet akvwebapp.

Ajoutez ces lignes à l'en-tête :

C#

```
using Azure.Identity;
using Azure.Security.KeyVault.Secrets;
using Azure.Core;
```

Ajoutez les lignes suivantes avant l'appel de `app.UseEndpoints` (.NET 5.0 ou version antérieure) ou l'appel `app.MapGet` (.NET 6.0), en mettant à jour l'URI pour refléter la valeur `vaultUri` de votre coffre de clés. Ce code utilise `DefaultAzureCredential()` pour l'authentification auprès du coffre de clés, qui utilise un jeton de l'identité managée pour s'authentifier. Pour plus d'informations sur l'authentification auprès du coffre de clés, consultez le [Guide du développeur](#). Le code utilise également un backoff exponentiel pour les nouvelles tentatives en cas de limitation du coffre de clés. Pour plus d'informations sur les limites de transaction du coffre de clés, consultez [Aide sur la limitation de requêtes Azure Key Vault](#).

C#

```
SecretClientOptions options = new SecretClientOptions()
{
    Retry =
    {
        Delay= TimeSpan.FromSeconds(2),
        MaxDelay = TimeSpan.FromSeconds(16),
        MaxRetries = 5,
        Mode = RetryMode.Exponential
    }
};

var client = new SecretClient(new Uri("https://<your-unique-key-vault-
name>.vault.azure.net/"), new DefaultAzureCredential(),options);

KeyVaultSecret secret = client.GetSecret("<mySecret>");

string secretValue = secret.Value;
```

Mettez à jour la ligne `await context.Response.WriteAsync("Hello World!");` pour qu'elle ressemble à ceci :

C#

```
await context.Response.WriteAsync(secretValue);
```

.NET 6.0

Mettez à jour la ligne `app.MapGet("/", () => "Hello World!");` pour qu'elle ressemble à ceci :

C#

```
app.MapGet("/", () => secretValue);
```

Veillez à enregistrer vos modifications avant de passer à l'étape suivante.

Redéployez votre application web

Maintenant que vous avez mis à jour votre code, vous pouvez le redéployer sur Azure à l'aide de ces commandes Git :

Bash

```
git add .
git commit -m "Updated web app to access my key vault"
git push azure main
```

Accéder à votre application web terminée

Bash

```
http://<your-webapp-name>.azurewebsites.net
```

Là où « Hello World » s'affichait, vous devez maintenant voir la valeur de votre secret.

Étapes suivantes

- Utiliser Azure Key Vault avec des applications déployées sur une machine virtuelle dans .NET

- En savoir plus sur les [identités managées pour les ressources Azure](#)
- Consulter le [Guide du développeur](#)
- Sécuriser l'accès à un coffre de clés

Guide pratique pour utiliser des identités managées pour App Service et Azure Functions

Article • 29/12/2023

Cet article vous montre comment créer une identité managée pour les applications App Service et Azure Functions et comment l'utiliser pour accéder à d'autres ressources.

ⓘ Important

Étant donné que **les identités managées ne prennent pas en charge les scénarios sur plusieurs répertoires**, elles ne se comportent pas comme prévu si la migration de votre application est effectuée sur plusieurs abonnements ou tenants. Pour recréer les identités managées après un tel déplacement, consultez [Les identités managées seront-elles recréées automatiquement si je déplace un abonnement vers un autre répertoire ?](#). Les ressources en aval doivent également disposer de stratégies d'accès mises à jour pour utiliser la nouvelle identité.

ⓘ Notes

Les identités managées ne sont pas disponibles pour les applications déployées dans [Azure Arc](#).

Une identité managée de Microsoft Entra ID permet à votre application d'accéder facilement à d'autres ressources protégées Microsoft Entra, comme Azure Key Vault. Managée par la plateforme Azure, l'identité ne nécessite pas que vous approvisionniez ou permutiez de secrets. Pour plus d'informations sur les identités managées dans Microsoft Entra ID, consultez [Identités managées pour les ressources Azure](#).

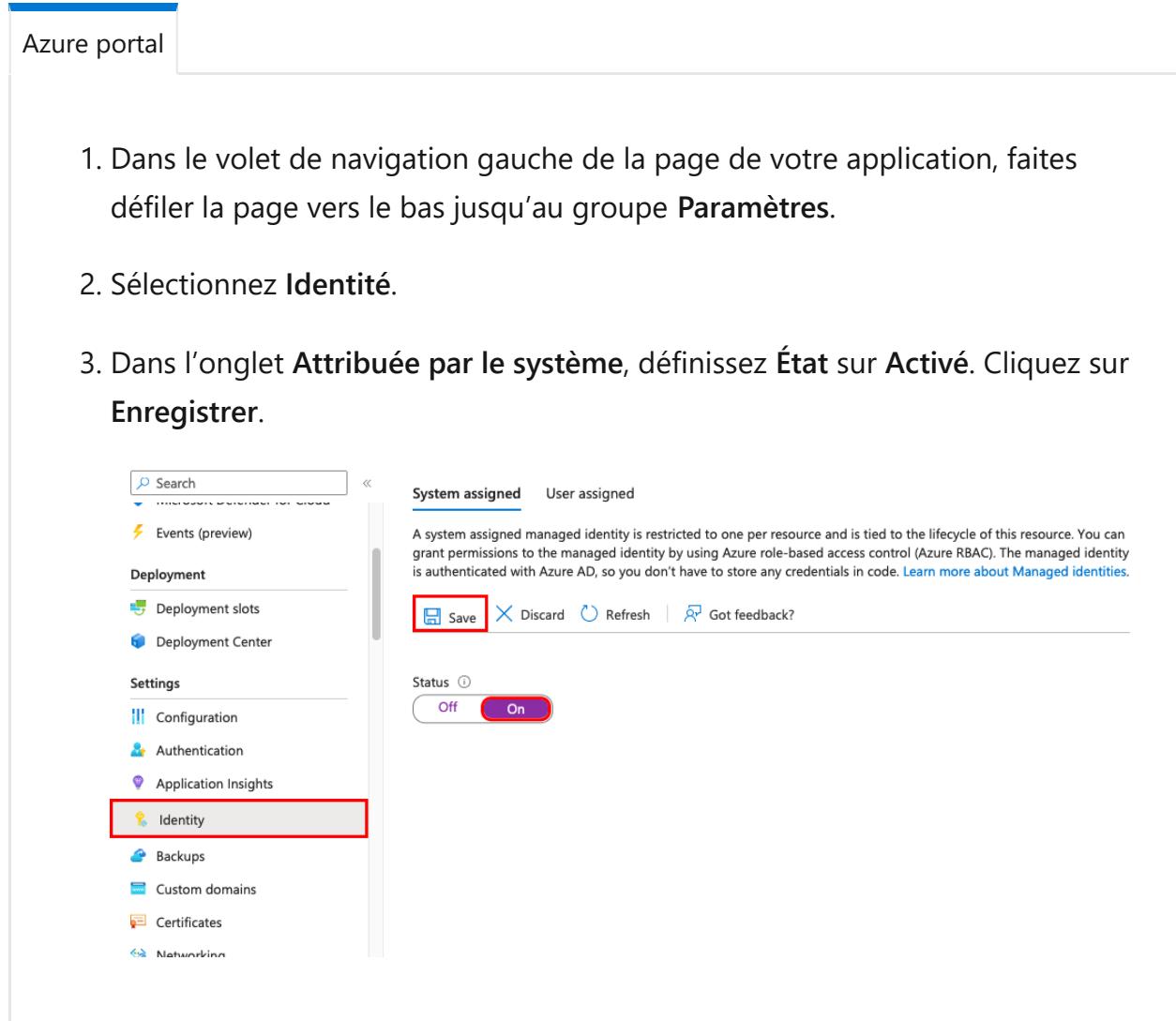
Deux types d'identité peuvent être accordés à votre application :

- Une **identité attribuée par le système** est liée à votre application et est supprimée si votre application est supprimée. Une application ne peut avoir qu'une seule identité attribuée par le système.
- Une **identité attribuée par l'utilisateur** est une ressource Azure autonome qui peut être assignée à votre application. Une application peut avoir plusieurs identités attribuées par l'utilisateur.

La configuration de l'identité managée est spécifique à l'emplacement. Pour configurer une identité managée pour un emplacement de déploiement dans le portail, accédez d'abord à l'emplacement. Pour rechercher l'identité managée pour votre application web ou votre emplacement de déploiement dans votre tenant Microsoft Entra à partir du Portail Azure, recherchez-la directement à partir de la page de **Présentation** de votre tenant. Le nom de l'emplacement est généralement semblable à `<app-name>/slots/<slot-name>`.

Ajouter une identité affectée par le système

Azure portal



1. Dans le volet de navigation gauche de la page de votre application, faites défiler la page vers le bas jusqu'au groupe **Paramètres**.

2. Sélectionnez **Identité**.

3. Dans l'onglet **Attribuée par le système**, définissez **État** sur **Activé**. Cliquez sur **Enregistrer**.

Ajouter une identité attribuée par l'utilisateur

La création d'une application avec une identité attribuée par l'utilisateur nécessite la création de l'identité, puis l'ajout de son identificateur de ressource à la configuration de votre application.

Azure portal

Tout d'abord, vous devrez créer une ressource d'identité attribuée par l'utilisateur.

1. Créez une ressource d'identité managée attribuée par l'utilisateur en suivant [ces instructions](#).
2. Dans le volet de navigation gauche de la page de votre application, faites défiler la page vers le bas jusqu'au groupe **Paramètres**.
3. Sélectionnez **Identité**.
4. Sélectionnez **Attribuée par l'utilisateur > Ajouter**.
5. Recherchez l'identité précédemment créée, sélectionnez-la, puis sélectionnez **Ajouter**.

The screenshot shows the Azure portal interface for managing identities. On the left, under the 'Identity' section of the 'my-demo-app' web app, the 'User assigned' tab is selected. A modal window titled 'Add user assigned managed i...' is open, showing a dropdown for 'Subscription' set to 'Visual Studio Enterprise Subscription'. In the 'User assigned managed identities' list, 'test' is selected. At the bottom of the modal, there is an 'Add' button.

Après la sélection de l'option **Ajouter**, l'application redémarre.

Configurer la ressource cible

Vous pouvez être amené à configurer la ressource cible pour autoriser l'accès à partir de votre application ou fonction. Par exemple, si vous [demandez un jeton](#) pour accéder à Key Vault, vous devez également ajouter une stratégie d'accès qui comprend l'identité managée de votre application ou de votre fonction. Sinon, vos appels à Key Vault seront rejetés, même si vous utilisez un jeton valide. Il en va de même pour Azure SQL Database. Pour en savoir plus sur les ressources qui prennent en charge les jetons Microsoft Entra, consultez [Services Azure prenant en charge l'authentification Microsoft Entra](#).

ⓘ Important

Les services principaux pour les identités gérées conservent un cache par URI de ressource pendant environ 24 heures. Si vous mettez à jour la stratégie d'accès d'une ressource cible particulière et que vous récupérez immédiatement un jeton pour cette ressource, vous pouvez continuer à obtenir un jeton mis en cache avec des autorisations obsolètes jusqu'à ce que ce jeton expire. Il n'existe actuellement aucun moyen de forcer l'actualisation d'un jeton.

Connexion aux services Azure dans le code de l'application

Grâce à son identité managée, une application peut obtenir des jetons pour les ressources Azure qui sont protégées par Microsoft Entra ID, comme Azure SQL Database, Azure Key Vault et Stockage Azure. Ces jetons représentent l'application qui accède à la ressource, pas un utilisateur spécifique de l'application.

App Service et Azure Functions fournissent un [point de terminaison REST](#) accessible en interne pour la récupération des jetons. Le point de terminaison REST est accessible depuis l'application avec une instruction HTTP GET standard, qui peut être implémentée avec un client HTTP générique dans chaque langage. Pour les langages .NET, JavaScript, Java et Python, la bibliothèque de client Azure Identity fournit une abstraction sur ce point de terminaison REST et simplifie l'expérience de développement. Pour se connecter à d'autres services Azure, il suffit d'ajouter un objet d'informations d'identification au client spécifique au service.

HTTP GET

La requête HTTP GET brute ressemble à l'exemple suivant :

```
HTTP  
  
GET /MSI/token?resource=https://vault.azure.net&api-version=2019-08-01  
HTTP/1.1  
Host: localhost:4141  
X-IDENTITY-HEADER: 853b9a84-5bfa-4b22-a3f3-0b9a43d9ad8a
```

Et voici un exemple de réponse :

```
HTTP  
  
HTTP/1.1 200 OK  
Content-Type: application/json
```

```
{  
    "access_token": "eyJ0eXAi...",  
    "expires_on": "1586984735",  
    "resource": "https://vault.azure.net",  
    "token_type": "Bearer",  
    "client_id": "5E29463D-71DA-4FE0-8E69-999B57DB23B0"  
}
```

Cette réponse est la même que la [réponse pour la demande de jeton d'accès de service à service de Microsoft Entra](#). Pour accéder à Key Vault, vous ajouterez ensuite la valeur de `access_token` à une connexion cliente avec le coffre.

Pour plus d'informations sur le point de terminaison REST, consultez [Référence du point de terminaison REST](#).

Supprimer une identité

Lorsque vous supprimez une identité affectée par le système, elle est supprimée de Microsoft Entra ID. Les identités affectées par le système sont également automatiquement supprimées de Microsoft Entra ID lorsque vous supprimez la ressource d'application elle-même.

Azure portal

1. Dans le volet de navigation gauche de la page de votre application, faites défiler la page vers le bas jusqu'au groupe **Paramètres**.
2. Sélectionnez **Identité**. Suivez ensuite les étapes en fonction du type d'identité :
 - **Identité affectée par le système** : Dans l'onglet **Affectée par le système**, basculez **État** sur **Désactivé**. Cliquez sur **Enregistrer**.
 - **Identité affectée par l'utilisateur** : Sélectionnez l'onglet **Affectée par l'utilisateur**, cochez la case de l'identité, puis sélectionnez **Supprimer**. Sélectionnez **Oui** pour confirmer.

ⓘ Notes

Vous pouvez également définir le paramètre d'application WEBSITE_DISABLE_MSI, qui désactive uniquement le service de jetons local. Toutefois, cela ne touche pas à

l'identité, et les outils continueront d'afficher l'identité managée comme étant activée. Par conséquent, l'utilisation de ce paramètre n'est pas recommandée.

Référence du point de terminaison REST

Une application avec une identité managée rend ce point de terminaison disponible en définissant deux variables d'environnement :

- **IDENTITY_ENDPOINT** - URL du service de jetons local.
- **IDENTITY_HEADER** - en-tête utilisé afin de limiter les attaques par falsification de requête côté serveur (SSRF). La plateforme effectue la rotation de la valeur.

IDENTITY_ENDPOINT est une URL locale à partir de laquelle votre application peut demander des jetons. Pour obtenir un jeton pour une ressource, effectuez une requête HTTP GET à destination de ce point de terminaison, en indiquant notamment les paramètres suivants :

[+] agrandir le tableau

Nom du paramètre	Dans	Description
resource	Requête	URI de ressource Microsoft Entra de la ressource pour laquelle un jeton doit être obtenu. Il peut s'agir d'un des services Azure prenant en charge l'authentification Microsoft Entra ou toute autre ressource URI.
api-version	Requête	Version de l'API de jeton à utiliser. Utiliser <code>2019-08-01</code> .
X-IDENTITY-HEADER	En-tête	Valeur de la variable d'environnement IDENTITY_HEADER . Cet en-tête est utilisé afin de limiter les attaques de falsification de requêtes côté serveur (SSRF).
client_id	Requête	(Facultatif) ID de client de l'identité affectée par l'utilisateur qui doit être utilisée. Ne peut pas être utilisée sur une demande qui inclut <code>principal_id</code> , <code>msi_res_id</code> ou <code>object_id</code> . Si tous les paramètres d'ID (<code>client_id</code> , <code>principal_id</code> , <code>object_id</code> et <code>msi_res_id</code>) sont omis, l'identité affectée par le système est utilisée.
principal_id	Requête	(Facultatif) ID de principal de service de l'identité affectée par l'utilisateur qui doit être utilisée. <code>object_id</code> est un alias qui peut être utilisé à la place. Impossible d'utiliser une demande qui inclut <code>client_id</code> , <code>msi_res_id</code> ou <code>object_id</code> . Si tous les paramètres d'ID (<code>client_id</code> , <code>principal_id</code> , <code>object_id</code> et <code>msi_res_id</code>) sont omis, l'identité affectée par le système est utilisée.

Nom du paramètre	Dans	Description
msi_res_id	Requête	(Facultatif) L'ID de ressource Azure de l'identité affectée par l'utilisateur qui doit être utilisée. Ne peut pas être utilisée sur une demande qui inclut <code>principal_id</code> , <code>client_id</code> ou <code>object_id</code> . Si tous les paramètres d'ID (<code>client_id</code> , <code>principal_id</code> , <code>object_id</code> et <code>msi_res_id</code>) sont omis, l'identité affectée par le système est utilisée.

ⓘ Important

Si vous tentez d'obtenir des jetons pour des identités affectées par l'utilisateur, vous devez inclure une des propriétés facultatives. Sinon, le service de jetons essaie d'obtenir un jeton pour une identité attribuée par le système, laquelle peut exister ou non.

Étapes suivantes

- Tutoriel : [Se connecter à SQL Database à partir d'App Service sans secrets à l'aide d'une identité managée](#)
- Accéder à Stockage Azure de manière sécurisée à l'aide d'une identité managée
- Appeler Microsoft Graph de manière sécurisée à l'aide d'une identité managée
- Se connecter en toute sécurité aux services avec des secrets Key Vault

Attribuer un rôle Azure pour l'accès aux données d'objet blob

Article • 20/10/2023

Microsoft Entra autorise les droits d'accès aux ressources sécurisées via le [contrôle d'accès en fonction du rôle Azure \(Azure RBAC\)](#). Le stockage Azure définit un ensemble de rôles intégrés Azure qui englobent les ensembles communs d'autorisations permettant d'accéder aux données blob.

Lorsqu'un rôle Azure est attribué à un principal de sécurité Microsoft Entra, Azure octroie l'accès à ces ressources pour ce principal de sécurité. Un principal de sécurité Microsoft Entra peut correspondre à un utilisateur, à un groupe, à un principal de service d'application ou à une [identité managée pour les ressources Azure](#).

Pour en savoir plus sur l'utilisation de Microsoft Entra ID pour autoriser l'accès aux données blob, consultez [Autoriser l'accès aux objets blob à l'aide de Microsoft Entra ID](#).

ⓘ Notes

Cet article explique comment attribuer un rôle Azure pour accéder aux données blob d'un compte de stockage. Pour en savoir plus sur l'attribution de rôles pour les opérations de gestion dans Stockage Azure, consultez [Utiliser le fournisseur de ressources Stockage Azure pour accéder aux ressources de gestion](#).

Affecter un rôle Azure

Vous pouvez utiliser le portail Azure, PowerShell, Azure CLI ou un modèle Azure Resource Manager pour attribuer un rôle d'accès aux données.

Azure portal

Pour accéder aux données de blob dans le portail Azure avec des informations d'identification Microsoft Entra, un utilisateur doit disposer des attributions de rôle suivantes :

- Un rôle d'accès aux données, comme **Lecteur des données Blob du stockage** ou **Contributeur aux données Blob du stockage**
- Le rôle **Lecteur** d'Azure Resource Manager, au minimum

Pour savoir comment attribuer ces rôles à un utilisateur, suivez les instructions fournies dans [Attribuer des rôles Azure à l'aide du portail Azure](#).

Le rôle **Lecteur** est un rôle d'Azure Resource Manager qui permet aux utilisateurs d'afficher les ressources de compte de stockage, mais pas de les modifier. Il ne fournit pas d'autorisations en lecture pour les données dans Stockage Azure, mais uniquement pour les ressources de gestion de compte. Le rôle **Lecteur** est nécessaire pour que les utilisateurs puissent accéder aux conteneurs d'objets blob du portail Azure.

Par exemple, si vous attribuez le rôle **Contributeur aux données Blob du stockage** à l'utilisatrice Mary au niveau d'un conteneur nommé **exemple-container**, puis Mary se voit attribuer un accès en lecture, écriture et suppression à tous les objets blob dans ce conteneur. Toutefois, si Mary souhaite afficher un objet blob dans le portail Azure, le rôle **Contributeur aux données Blob du stockage** ne fournit pas les autorisations suffisantes pour naviguer dans le portail et atteindre l'objet blob dans le but de le consulter. Les autres autorisations sont requises pour naviguer dans le portail et afficher les autres ressources qui y sont visibles.

Un utilisateur doit disposer du rôle **Lecteur** pour utiliser le portail Azure avec les informations d'identification Microsoft Entra. Cela étant, si un utilisateur dispose d'un rôle doté des autorisations

Microsoft.Storage/storageAccounts/listKeys/action, il peut utiliser le portail avec les clés de compte de stockage, par le biais de l'autorisation de clé partagée. Pour utiliser les clés de compte de stockage, l'accès à la clé partagée doit être autorisé pour le compte de stockage. Pour plus d'informations sur l'autorisation ou l'interdiction de l'accès à la clé partagée, consultez [Empêcher l'autorisation avec clé partagée pour un compte Stockage Azure](#).

Vous pouvez également attribuer un rôle Azure Resource Manager offrant des autorisations supplémentaires au-delà du rôle **Lecteur**. L'attribution d'autorisations minimales est recommandée en guise de meilleure pratique de sécurité. Pour plus d'informations, consultez [Meilleures pratiques pour Azure RBAC](#).

Notes

Avant de vous attribuer un rôle pour l'accès aux données, vous pouvez accéder aux données de votre compte de stockage via le portail Azure, car ce dernier peut également utiliser la clé de compte pour l'accès aux données. Pour plus d'informations, consultez [Choisir comment autoriser l'accès à des données blobs dans le portail Azure](#).

Gardez à l'esprit les points suivants concernant les attributions de rôles Azure dans Stockage Azure :

- Lorsque vous créez un compte Stockage Azure, aucune autorisation d'accès aux données ne vous est automatiquement attribuée via Microsoft Entra ID. Vous devez vous attribuer explicitement un rôle Azure pour le Stockage Azure. Vous pouvez l'attribuer au niveau de votre abonnement, groupe de ressources, compte de stockage ou conteneur.
- Si le compte de stockage est verrouillé à l'aide d'un verrou en lecture seule Azure Resource Manager, le verrou empêche l'attribution de rôles Azure étendus au compte de stockage ou à un conteneur.
- Si vous avez défini les autorisations nécessaires pour accéder aux données par le biais de Microsoft Entra ID et ne parvenez pas à accéder aux données, par exemple, vous obtenez une erreur « AuthorizationPermissionMismatch ». Prévoyez suffisamment de temps pour permettre la réPLICATION des modifications que vous avez apportées aux autorisations dans Microsoft Entra ID et assurez-vous de ne pas avoir d'affectations de refus bloquant votre accès. Consultez [Comprendre les affectations de refus Azure](#).

ⓘ Notes

Vous pouvez créer des rôles RBAC Azure personnalisés pour un accès granulaire aux données d'objets blob. Pour plus d'informations, consultez [Rôles Azure personnalisés](#).

Étapes suivantes

- [Qu'est-ce que le contrôle d'accès en fonction du rôle Azure \(RBAC Azure\) ?](#)
- [Meilleures pratiques pour Azure RBAC](#)

Identités managées dans Azure Container Apps

Article • 26/10/2023

Une identité managée de Microsoft Entra ID permet à votre application conteneur d'accéder à d'autres ressources protégées par Microsoft Entra. Pour plus d'informations sur les identités managées dans Microsoft Entra ID, consultez [Identités managées pour les ressources Azure](#).

Deux types d'identité peuvent être accordés à votre application de conteneur :

- Une **identité attribuée par le système** est liée à votre application de conteneur et est supprimée si votre application de conteneur est supprimée. Une application ne peut avoir qu'une seule identité attribuée par le système.
- Une **identité attribuée par l'utilisateur** est une ressource Azure autonome qui peut être affectée à une votre application de conteneur et d'autres ressources. Une application de conteneur peut avoir plusieurs identités attribuées par l'utilisateur. L'identité existe jusqu'à ce que vous la supprimiez.

Pourquoi utiliser une identité managée ?

Vous pouvez utiliser une identité managée dans une application conteneur en cours d'exécution pour vous authentifier auprès de n'importe quel [service prenant en charge l'authentification](#) Microsoft Entra.

Avec des identités managées :

- Votre application se connecte aux ressources avec l'identité managée. Vous n'avez pas besoin de gérer les informations d'identification dans votre application de conteneur.
- Vous pouvez utiliser le contrôle d'accès en fonction du rôle pour accorder des autorisations spécifiques à une identité managée.
- Les identités attribuées par le système sont automatiquement créées et gérées. Elles sont supprimées lorsque votre application de conteneur est supprimée.
- Vous pouvez ajouter et supprimer des identités attribuées par l'utilisateur et les affecter à plusieurs ressources. Elles sont indépendantes du cycle de vie de votre application de conteneur.
- Vous pouvez utiliser l'identité managée pour [vous authentifier auprès d'un Azure Container Registry privé](#) sans nom d'utilisateur et mot de passe pour extraire des conteneurs pour votre application conteneur.

- Vous pouvez utiliser l'identité managée pour créer des connexions pour les applications prenant en charge Dapr via des composants Dapr

Cas d'utilisation courants

Les identités attribuées par le système conviennent parfaitement aux charges de travail qui :

- sont contenues dans une ressource unique
- nécessitent des identités indépendantes

Les identités attribuées par l'utilisateur sont idéales pour les charges de travail qui :

- s'exécutent sur plusieurs ressources et peuvent partager une même identité
- nécessitent une pré-autorisation pour accéder à une ressource sécurisée

Limites

L'utilisation d'identités managées dans les règles de mise à l'échelle n'est pas prise en charge. Vous devez toujours inclure la chaîne de connexion ou la clé dans le `secretRef` de la règle de mise à l'échelle.

Les conteneurs Init ne peuvent pas accéder aux identités managées.

Configurer des identités managées

Vous pouvez configurer vos identités managées via :

- le portail Azure
- Interface de ligne de commande Azure
- votre modèle Azure Resource Manager (ARM)

Lorsqu'une identité managée est ajoutée, supprimée ou modifiée sur une application de conteneur en cours d'exécution, l'application ne redémarre pas automatiquement et une nouvelle révision n'est pas créée.

ⓘ Notes

Vous devez créer une nouvelle révision lors de l'ajout d'une identité managée à une application de conteneur déployée avant le 11 avril 2022.

Ajouter une identité affectée par le système

Azure portal

1. Dans le volet de navigation gauche de la page de votre application conteneur, faites défiler la page vers le bas jusqu'au groupe **Paramètres**.
2. Sélectionnez **Identité**.
3. Dans l'onglet **Attribuée par le système**, définissez **État** sur **Activé**. Cliquez sur **Enregistrer**.

The screenshot shows the Azure portal interface for a 'Container App' named 'music-store'. On the left, there's a sidebar with links like Overview, Access control (IAM), Tags, and Diagnose and solve problems. The main area is titled 'music-store | Identity' and shows two tabs: 'System assigned' (which is selected) and 'User assigned'. Below the tabs, there's a note about system-assigned identities being restricted to one per resource and tied to the lifecycle of the resource. It explains that permissions can be granted using Azure RBAC and that the managed identity is authenticated with Azure AD. There are 'Save', 'Discard', 'Refresh', and 'Got feedback?' buttons at the bottom. A red box highlights the 'Status' section, where a switch is set to 'On', indicating that a system-assigned identity is active.

Ajouter une identité attribuée par l'utilisateur

La configuration d'une application de conteneur avec une identité attribuée par l'utilisateur nécessite d'abord de créer l'identité, puis d'ajouter son identificateur de ressource à la configuration de votre application de conteneur. Vous pouvez créer des identités attribuées par l'utilisateur via le Portail Azure ou l'interface Azure CLI. Pour plus d'informations sur la création et la gestion des identités attribuées par l'utilisateur, consultez [Gérer les identités managées attribuées par l'utilisateur](#).

Azure portal

Tout d'abord, vous devrez créer une ressource d'identité attribuée par l'utilisateur.

1. Créez une ressource d'identité managée affectée par l'utilisateur en fonction des étapes décrites dans [Gérer les identités managées affectées par l'utilisateur](#).

2. Dans le volet de navigation gauche de la page de votre application conteneur, faites défiler la page vers le bas jusqu'au groupe **Paramètres**.
3. Sélectionnez **Identité**.
4. Dans l'onglet **Affecté(e) par l'utilisateur**, sélectionnez **Ajouter**.
5. Recherchez l'identité que vous avez créée précédemment et sélectionnez-la. Sélectionnez **Ajouter**.

The screenshot shows the Microsoft Azure portal interface. On the left, the 'music-store' container app blade is open, showing the 'Identity' section. The 'User assigned' tab is selected. A modal window titled 'Add user assigned managed identity...' is displayed on the right. In the modal, the 'Subscription' dropdown is set to 'Demo-Subscription'. Below it, a list of available identities is shown: 'user-identity-1' (Resource Group: my-group), 'user-identity-2' (Resource Group: my-group), and 'user-identity3' (Resource Group: mv-arouo). The identity 'music-store-user-identity' (Resource Group: my-group, Subscription: Demo-Subscription) is selected and highlighted with a red box. At the bottom of the modal is a blue 'Add' button.

Configurer une ressource cible

Pour certaines ressources, vous devez configurer les attributions de rôles pour l'identité managée de votre application afin d'en accorder l'accès. Dans le cas contraire, les appels de votre application vers des services, tels qu'Azure Key Vault et Azure SQL Database, seront rejetés même si vous utilisez un jeton valide pour cette identité. Pour plus d'informations sur le contrôle d'accès en fonction du rôle Azure (Azure RBAC), consultez [Qu'est-ce que le contrôle d'accès en fonction du rôle Azure \(Azure RBAC\) ?](#). Pour en savoir plus sur les ressources qui prennent en charge les jetons Microsoft Entra, consultez [Services Azure prenant en charge l'authentification Microsoft Entra](#).

i Important

Les services principaux pour les identités gérées conservent un cache par URI de ressource pendant environ 24 heures. Si vous mettez à jour la stratégie d'accès

d'une ressource cible particulière et que vous récupérez immédiatement un jeton pour cette ressource, vous pouvez continuer à obtenir un jeton mis en cache avec des autorisations obsolètes jusqu'à ce que ce jeton expire. Il n'existe actuellement aucun moyen de forcer l'actualisation d'un jeton.

Connexion aux services Azure dans le code de l'application

Avec des identités managées, une application peut obtenir des jetons pour accéder aux ressources Azure qui utilisent Microsoft Entra ID, comme Azure SQL Database, Azure Key Vault et Stockage Azure. Ces jetons représentent l'application qui accède à la ressource, pas un utilisateur spécifique de l'application.

Container Apps fournit un [point de terminaison REST](#) accessible en interne pour récupérer des jetons. Le point de terminaison REST est accessible depuis l'application avec une instruction HTTP GET standard, qui peut être implémentée avec un client HTTP générique dans chaque langage. Pour les langages .NET, JavaScript, Java et Python, la bibliothèque de client Azure Identity fournit une abstraction sur ce point de terminaison REST. Pour se connecter à d'autres services Azure, il suffit d'ajouter un objet d'informations d'identification au client spécifique au service.

⚠ Notes

Lors de l'utilisation de la bibliothèque de client Azure Identity, l'ID client d'identité managée affecté par l'utilisateur doit être spécifié.

.NET

⚠ Notes

Lorsque vous vous connectez à des sources de données Azure SQL avec **Entity Framework Core**, pensez à utiliser **Microsoft.Data.SqlClient**, qui fournit des chaînes de connexion spéciales pour la connectivité de l'identité managée.

Pour les applications .NET, la façon la plus simple d'utiliser une identité managée consiste à recourir à la [bibliothèque de client Azure Identity pour .NET](#). Pour plus d'informations, consultez les titres de documentation respectifs de la bibliothèque de client :

- Ajouter la bibliothèque de client Azure Identity à votre projet
- Accéder au service Azure avec une identité affectée par le système
- Accéder au service Azure avec une identité affectée par l'utilisateur

Les exemples liés utilisent `DefaultAzureCredential`. C'est utile pour la majorité des scénarios, car le même modèle fonctionne dans Azure (avec des identités managées) et sur votre ordinateur local (sans identités managées).

Afficher les identités managées

Vous pouvez afficher les identités managées attribuées par le système et attribuées par l'utilisateur à l'aide de la commande Azure CLI suivante. La sortie affiche le type d'identité managée, les ID de locataire et les ID principaux de toutes les identités managées affectées à votre application de conteneur.

Azure CLI

```
az containerapp identity show --name <APP_NAME> --resource-group  
<GROUP_NAME>
```

Supprimer une identité managée

Lorsque vous supprimez une identité affectée par le système, elle est supprimée de Microsoft Entra ID. Les identités affectées par le système sont également automatiquement supprimées de l'ID Microsoft Entra lorsque vous supprimez la ressource d'application conteneur elle-même. La suppression des identités managées affectées par l'utilisateur de votre application conteneur ne les supprime pas de l'ID Microsoft Entra.

Azure portal

1. Dans le volet de navigation gauche de la page de votre application, faites défiler la page vers le bas jusqu'au groupe **Paramètres**.
2. Sélectionnez **Identité**. Suivez ensuite les étapes en fonction du type d'identité :
 - **Identité affectée par le système** : Dans l'onglet **Affectée par le système**, basculez **État** sur **Désactivé**. Cliquez sur **Enregistrer**.
 - **Identité affectée par l'utilisateur** : Sélectionnez l'onglet **Affectée par l'utilisateur**, cochez la case de l'identité, puis sélectionnez **Supprimer**.

Sélectionnez **Oui** pour confirmer.

Étapes suivantes

Surveiller une application

Configurer le contrôle d'accès en fonction du rôle avec Microsoft Entra ID pour votre compte Azure Cosmos DB

Article • 25/10/2023

S'APPLIQUE À : NoSQL

① Notes

Cet article concerne le contrôle d'accès en fonction du rôle pour les opérations de plan de données dans Azure Cosmos DB. Si vous utilisez des opérations de plan de gestion, consultez l'article [Contrôle d'accès en fonction du rôle](#) appliqué à vos opérations de plan de gestion.

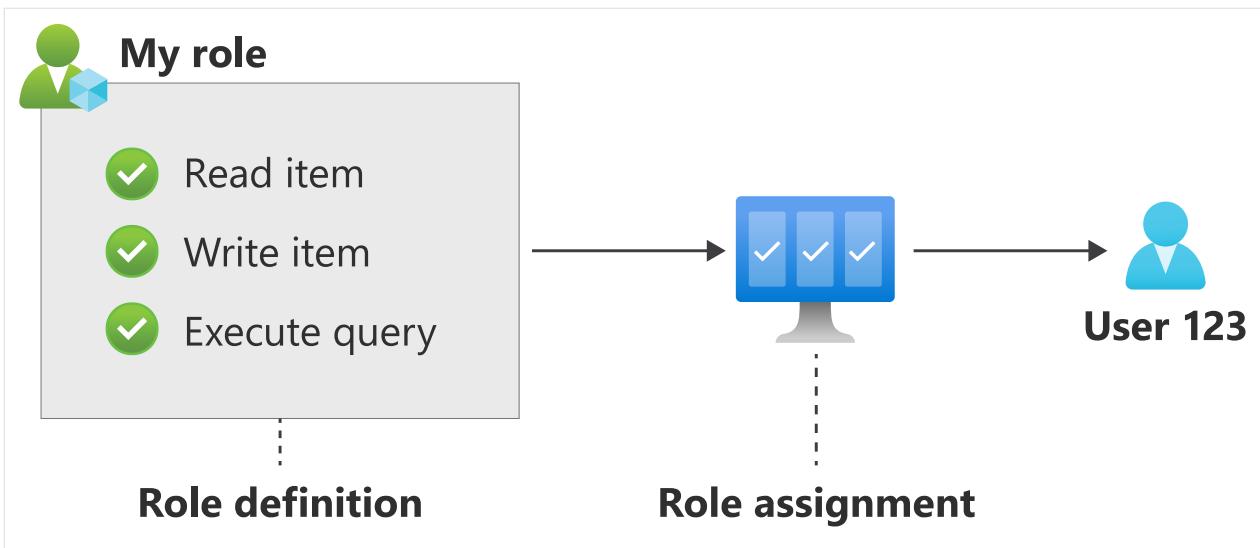
Azure Cosmos DB expose un système intégré de contrôle d'accès en fonction du rôle, qui vous permet :

- Authentifier vos requêtes de données avec une identité Microsoft Entra.
- D'autoriser vos demandes de données avec un modèle d'autorisation basé sur les rôles et précis.

Concepts

Le contrôle d'accès en fonction du rôle du plan de données Azure Cosmos DB repose sur des concepts couramment présents dans d'autres systèmes de contrôle d'accès en fonction du rôle, comme le [contrôle d'accès en fonction du rôle Azure](#) :

- Le [modèle d'autorisation](#) se compose d'un ensemble d'[actions](#) ; chacune de ces actions est mappée à une ou plusieurs opérations de base de données. Voici quelques exemples d'actions : la lecture d'un élément, l'écriture d'un élément ou l'exécution d'une requête.
- Les utilisateurs Azure Cosmos DB créent des [définitions de rôle](#) contenant une liste d'actions autorisées.
- Les définitions de rôle sont affectées à des identités Microsoft Entra spécifiques via des [attribution de rôle](#). Une attribution de rôle définit également l'étendue à laquelle s'applique la définition de rôle ; il existe actuellement trois étendues :
 - Un compte Azure Cosmos DB
 - Une base de données Azure Cosmos DB
 - Un conteneur Azure Cosmos DB.



Modèle d'autorisation

ⓘ Important

Ce modèle d'autorisation couvre uniquement les opérations de base de données qui impliquent la lecture et l'écriture de données. Il **ne couvre aucun** type d'opération de gestion sur des ressources de gestion, notamment :

- Créer/remplacer/supprimer une base de données
- Créer/remplacer/supprimer un conteneur
- Lire/modifier le débit d'un conteneur
- Créer/remplacer/supprimer/lire des procédures stockées
- Créer/remplacer/supprimer/lire des déclencheurs
- Créer/remplacer/supprimer/lire des fonctions définies par l'utilisateur

*Vous ne pouvez pas utiliser le kit de développement logiciel (SDK) de plan de données Azure Cosmos DB pour authentifier les opérations de gestion avec une identité Microsoft Entra. Au lieu de cela, vous devez utiliser un **contrôle d'accès en fonction du rôle Azure** via l'une des options suivantes :*

- **Modèles Azure Resource Manager (modèles ARM)**
- **Scripts Azure PowerShell**
- **Scripts Azure CLI**
- Bibliothèques de gestion Azure disponibles pour :
 - [.NET](#)
 - [Java](#)
 - [Python](#)

Les demandes Lire la base de données et Lire le conteneur sont considérées comme des **demandes de métadonnées**. L'accès à ces opérations peut être accordé comme indiqué

dans la section suivante.

Le tableau ci-dessous répertorie toutes les actions exposées par le modèle d'autorisation.

Nom	Opération(s) de base de données correspondante(s)
<code>Microsoft.DocumentDB/databaseAccounts/readMetadata</code>	Lire les métadonnées de compte. Pour plus d'informations, consultez Demandes de métadonnées .
<code>Microsoft.DocumentDB/databaseAccounts/sqlDatabases/containers/items/create</code>	Créer un élément.
<code>Microsoft.DocumentDB/databaseAccounts/sqlDatabases/containers/items/read</code>	Lire un élément individuel en utilisant son ID et sa clé de partition (lecture par pointage).
<code>Microsoft.DocumentDB/databaseAccounts/sqlDatabases/containers/items/replace</code>	Remplacer un élément existant.
<code>Microsoft.DocumentDB/databaseAccounts/sqlDatabases/containers/items/upsert</code>	Effectuer un « upsert » sur un élément. Cette opération crée un élément s'il n'existe pas déjà ou remplace l'élément s'il existe.
<code>Microsoft.DocumentDB/databaseAccounts/sqlDatabases/containers/items/delete</code>	Supprimer un élément.
<code>Microsoft.DocumentDB/databaseAccounts/sqlDatabases/containers/executeQuery</code>	Exécuter une requête SQL .
<code>Microsoft.DocumentDB/databaseAccounts/sqlDatabases/containers/readChangeFeed</code>	Lire dans le flux de modification du conteneur. Exécuter des requêtes SQL à l'aide des kits de développement logiciel (SDK).

Nom	Opération(s) de base de données correspondante(s)
<code>Microsoft.DocumentDB/databaseAccounts/sqlDatabases/containers/executeStoredProcedure</code>	Exécuter une procédure stockée.
<code>Microsoft.DocumentDB/databaseAccounts/sqlDatabases/containers/manageConflicts</code>	Gérer les conflits pour les comptes de régions à plusieurs écritures (c'est-à-dire lister et supprimer des éléments dans le flux en conflit).

① Notes

Lors de l'exécution de requêtes via les kits de développement logiciel (SDK), les autorisations

`Microsoft.DocumentDB/databaseAccounts/sqlDatabases/containers/executeQuery` et
`Microsoft.DocumentDB/databaseAccounts/sqlDatabases/containers/readChangeFeed` sont requises.

Les caractères génériques sont pris en charge au niveau des *conteneurs* et des *éléments* :

- `Microsoft.DocumentDB/databaseAccounts/sqlDatabases/containers/*`
- `Microsoft.DocumentDB/databaseAccounts/sqlDatabases/containers/items/*`

Requêtes de métadonnées

Les kits de développement logiciel (SDK) Azure Cosmos DB émettent des requêtes de métadonnées en lecture seule lors de l'initialisation et traitent des requêtes de données spécifiques. Ces demandes récupèrent différents détails de configuration, comme :

- La configuration globale de votre compte, qui comprend les régions Azure où le compte est disponible.
- La clé de partition de vos conteneurs ou leur stratégie d'indexation.
- La liste des partitions physiques qui constituent un conteneur et leurs adresses.

Elles ne récupèrent **pas** les données que vous avez stockées dans votre compte.

Pour garantir la meilleure transparence de notre modèle d'autorisation, ces demandes de métadonnées sont explicitement couvertes par l'action

`Microsoft.DocumentDB/databaseAccounts/readMetadata`. Cette action doit être autorisée dans toutes les situations où l'accès à votre compte Azure Cosmos DB se fait via un des SDK Azure

Cosmos DB. Elle peut être affectée (via une attribution de rôle) à n'importe quel niveau de la hiérarchie Azure Cosmos DB (c'est-à-dire le compte, la base de données ou le conteneur).

Les demandes de métadonnées réelles autorisées par l'action

`Microsoft.DocumentDB/databaseAccounts/readMetadata` dépendant de l'étendue à laquelle l'action est affectée :

Étendue	Demandes autorisées par l'action
Compte	<ul style="list-style-type: none">• Lister les bases de données sous le compte• Pour chaque base de données sous le compte, les actions autorisées au niveau de l'étendue de la base de données
Base de données	<ul style="list-style-type: none">• Lire les métadonnées de la base de données• Lister les conteneurs sous la base de données• Pour chaque conteneur sous la base de données, les actions autorisées au niveau de l'étendue du conteneur
Conteneur	<ul style="list-style-type: none">• Lire les métadonnées du conteneur• Lister les partitions physiques sous le conteneur• Résoudre l'adresse de chaque partition physique

ⓘ Important

Le débit n'est pas compris dans les métadonnées de cette action.

Définitions de rôle intégré

Azure Cosmos DB expose 2 définitions de rôles intégrées :

ⓘ Important

Les **définitions de rôle** font ici référence aux définitions de rôle spécifiques à Azure Cosmos DB. Celles-ci sont distinctes des définitions de rôle de contrôle d'accès en fonction du rôle Azure.

id	Nom	Actions incluses
00000000-0000-0000-0000-000000000001	Lecteur de données intégré	<code>Microsoft.DocumentDB/databaseAccounts/readMetadata</code> <code>Microsoft.DocumentDB/databaseAccounts/sqlDatabases/containers/items/read</code> <code>Microsoft.DocumentDB/databaseAccounts/sqlDatabases/containers/executeQuery</code> <code>Microsoft.DocumentDB/databaseAccounts/sqlDatabases/containers/readChangeFeed</code>
00000000-0000-0000-0000-000000000002	Contributeur de données intégré	<code>Microsoft.DocumentDB/databaseAccounts/readMetadata</code> <code>Microsoft.DocumentDB/databaseAccounts/sqlDatabases/containers/*</code> <code>Microsoft.DocumentDB/databaseAccounts/sqlDatabases/containers/items/*</code>
	Cosmos DB	

Créer des définitions de rôle personnalisé

Quand vous créez une définition de rôle personnalisé, vous devez fournir les éléments suivants :

- Le nom de votre compte Azure Cosmos DB.
- Le groupe de ressources contenant votre compte.
- Le type de la définition de rôle : `CustomRole`.
- Le nom de la définition de rôle.
- Une liste des `actions` que vous voulez autoriser pour le rôle.
- Une ou plusieurs étendues auxquelles la définition de rôle peut être affectée ; les étendues prises en charge sont :
 - `/` (niveau compte),
 - `/dbs/<database-name>` (niveau base de données),
 - `/dbs/<database-name>/colls/<container-name>` (niveau conteneur).

① Notes

Les opérations décrites sont disponibles dans :

- Azure PowerShell : [Az.CosmosDB version 1.2.0](#) ou ultérieure
- Azure CLI : version 2.24.0 ou ultérieure

Utilisation de Microsoft Azure PowerShell

Créez un rôle nommé `MyReadOnlyRole` qui contient seulement des actions de lecture :

PowerShell

```
$resourceGroupName = "<myResourceGroup>"  
$accountName = "<myCosmosAccount>"  
New-AzCosmosDBSqlRoleDefinition -AccountName $accountName  
    -ResourceGroupName $resourceGroupName  
    -Type CustomRole -RoleName MyReadOnlyRole  
    -DataAction @(`n        'Microsoft.DocumentDB/databaseAccounts/readMetadata',`n        'Microsoft.DocumentDB/databaseAccounts/sqlDatabases/containers/items/read',`n        'Microsoft.DocumentDB/databaseAccounts/sqlDatabases/containers/executeQuery',`n        'Microsoft.DocumentDB/databaseAccounts/sqlDatabases/containers/readChangeFeed')`n    -AssignableScope "/"
```

Créez un rôle nommé `MyReadWriteRole` qui contient toutes les actions :

PowerShell

```
New-AzCosmosDBSqlRoleDefinition -AccountName $accountName ` 
    -ResourceGroupName $resourceGroupName ` 
    -Type CustomRole -RoleName MyReadWriteRole ` 
    -DataAction @(` 
        'Microsoft.DocumentDB/databaseAccounts/readMetadata', 
        'Microsoft.DocumentDB/databaseAccounts/sqlDatabases/containers/items/*', 
        'Microsoft.DocumentDB/databaseAccounts/sqlDatabases/containers/*') ` 
    -AssignableScope "/"
```

Listez les définitions de rôle que vous avez créées pour récupérer leur ID :

PowerShell

```
Get-AzCosmosDBSqlRoleDefinition -AccountName $accountName ` 
    -ResourceGroupName $resourceGroupName
```

Sortie

```
RoleName      : MyReadWriteRole
Id           :
/subscriptions/<mySubscriptionId>/resourceGroups/<myResourceGroup>/providers/Microsoft.DocumentDB/databaseAcc
              ounts/<myCosmosAccount>/sqlRoleDefinitions/<roleDefinitionId>
Type         : CustomRole
Permissions   : {Microsoft.Azure.Management.CosmosDB.Models.Permission}
AssignableScopes :
{/subscriptions/<mySubscriptionId>/resourceGroups/<myResourceGroup>/providers/Microsoft.DocumentDB/databaseAc
              counts/<myCosmosAccount>}

RoleName      : MyReadOnlyRole
Id           :
/subscriptions/<mySubscriptionId>/resourceGroups/<myResourceGroup>/providers/Microsoft.DocumentDB/databaseAcc
              ounts/<myCosmosAccount>/sqlRoleDefinitions/<roleDefinitionId>
Type         : CustomRole
Permissions   : {Microsoft.Azure.Management.CosmosDB.Models.Permission}
AssignableScopes :
{/subscriptions/<mySubscriptionId>/resourceGroups/<myResourceGroup>/providers/Microsoft.DocumentDB/databaseAc
              counts/<myCosmosAccount>}
```

Utilisation de l'interface de ligne de commande Azure (CLI)

Créez un rôle nommé *MyReadOnlyRole* qui contient uniquement des actions de lecture dans un fichier nommé **role-definition-ro.json** :

JSON

```
{ 
  "RoleName": "MyReadOnlyRole",
```

```
"Type": "CustomRole",
"AssignableScopes": ["/"],
"Permissions": [
    "DataActions": [
        "Microsoft.DocumentDB/databaseAccounts/readMetadata",
        "Microsoft.DocumentDB/databaseAccounts/sqlDatabases/containers/items/read",
        "Microsoft.DocumentDB/databaseAccounts/sqlDatabases/containers/executeQuery",
        "Microsoft.DocumentDB/databaseAccounts/sqlDatabases/containers/readChangeFeed"
    ]
]
}
```

Azure CLI

```
resourceGroupName='<myResourceGroup>'
accountName='<myCosmosAccount>'
az cosmosdb sql role definition create --account-name $accountName --resource-group $resourceGroupName --body @role-definition-ro.json
```

Créez un rôle nommé *MyReadWriteRole* qui contient toutes les actions dans un fichier nommé **role-definition-rw.json** :

JSON

```
{
    "RoleName": "MyReadWriteRole",
    "Type": "CustomRole",
    "AssignableScopes": ["/"],
    "Permissions": [
        "DataActions": [
            "Microsoft.DocumentDB/databaseAccounts/readMetadata",
            "Microsoft.DocumentDB/databaseAccounts/sqlDatabases/containers/items/*",
            "Microsoft.DocumentDB/databaseAccounts/sqlDatabases/containers/*"
        ]
    ]
}
```

Azure CLI

```
az cosmosdb sql role definition create --account-name $accountName --resource-group $resourceGroupName --body @role-definition-rw.json
```

Listez les définitions de rôle que vous avez créées pour récupérer leur ID :

Azure CLI

```
az cosmosdb sql role definition list --account-name $accountName --resource-group
```

```
$resourceGroupName
```

JSON

```
[  
  {  
    "assignableScopes": [  
  
      "/subscriptions/<mySubscriptionId>/resourceGroups/<myResourceGroup>/providers/Microsoft.DocumentDB/databaseAccounts/<myCosmosAccount>"  
    ],  
    "id":  
      "/subscriptions/<mySubscriptionId>/resourceGroups/<myResourceGroup>/providers/Microsoft.DocumentDB/databaseAccounts/<myCosmosAccount>/sqlRoleDefinitions/<roleDefinitionId>",  
    "name": "<roleDefinitionId>",  
    "permissions": [  
      {  
        "dataActions": [  
          "Microsoft.DocumentDB/databaseAccounts/readMetadata",  
          "Microsoft.DocumentDB/databaseAccounts/sqlDatabases/containers/items/*",  
          "Microsoft.DocumentDB/databaseAccounts/sqlDatabases/containers/*"  
        ],  
        "notDataActions": []  
      }  
    ],  
    "resourceGroup": "<myResourceGroup>",  
    "roleName": "MyReadWriteRole",  
    "sqlRoleDefinitionGetResultsType": "CustomRole",  
    "type": "Microsoft.DocumentDB/databaseAccounts/sqlRoleDefinitions"  
  },  
  {  
    "assignableScopes": [  
  
      "/subscriptions/<mySubscriptionId>/resourceGroups/<myResourceGroup>/providers/Microsoft.DocumentDB/databaseAccounts/<myCosmosAccount>"  
    ],  
    "id":  
      "/subscriptions/<mySubscriptionId>/resourceGroups/<myResourceGroup>/providers/Microsoft.DocumentDB/databaseAccounts/<myCosmosAccount>/sqlRoleDefinitions/<roleDefinitionId>",  
    "name": "<roleDefinitionId>",  
    "permissions": [  
      {  
        "dataActions": [  
          "Microsoft.DocumentDB/databaseAccounts/readMetadata",  
  
          "Microsoft.DocumentDB/databaseAccounts/sqlDatabases/containers/items/read",  
  
          "Microsoft.DocumentDB/databaseAccounts/sqlDatabases/containers/executeQuery",  
  
          "Microsoft.DocumentDB/databaseAccounts/sqlDatabases/containers/readChangeFeed"  
        ],  
        "notDataActions": []  
      }  
    ],  
    "resourceGroup": "<myResourceGroup>",  
  }
```

```
        "roleName": "MyReadOnlyRole",
        "sqlRoleDefinitionGetResultsType": "CustomRole",
        "type": "Microsoft.DocumentDB/databaseAccounts/sqlRoleDefinitions"
    }
]
```

Utilisation de modèles Azure Resource Manager

Pour obtenir une référence et des exemples d'utilisation de modèles Azure Resource Manager afin de créer des définitions de rôle, consultez [Microsoft.DocumentDB databaseAccounts/sqlRoleDefinitions](#).

Créer des attributions de rôles

Vous pouvez associer des définitions de rôle intégré ou personnalisé à vos identités Microsoft Entra. Lors de la création d'une attribution de rôle, vous devez fournir :

- Le nom de votre compte Azure Cosmos DB.
- Le groupe de ressources contenant votre compte.
- L'ID de la définition de rôle à attribuer.
- L'ID principal de l'identité à laquelle la définition de rôle doit être affectée.
- L'étendue de l'attribution de rôle ; les étendues prises en charge sont :
 - / (niveau compte)
 - /dbs/<database-name> (niveau base de données)
 - /dbs/<database-name>/colls/<container-name> (niveau conteneur)

La portée doit correspondre ou être une sous-étendue de l'une des étendues assignables de la définition de rôle.

ⓘ Notes

Si vous voulez créer une attribution de rôle pour un principal de service, veillez à utiliser son ID d'objet tel qu'il figure dans la section **Applications d'entreprise** du volet du portail Microsoft Entra ID.

ⓘ Notes

Les opérations décrites sont disponibles dans :

- Azure PowerShell : [Az.CosmosDB version 1.2.0](#) ou ultérieure

- Azure CLI : version 2.24.0 ou ultérieure

Utilisation de Microsoft Azure PowerShell

Affecter un rôle à une identité :

PowerShell

```
$resourceGroupName = "<myResourceGroup>"  
$accountName = "<myCosmosAccount>"  
$readOnlyRoleDefinitionId = "<roleDefinitionId>" # as fetched above  
# For Service Principals make sure to use the Object ID as found in the Enterprise  
applications section of the Azure Active Directory portal blade.  
$principalId = "<aadPrincipalId>"  
New-AzCosmosDBSqlRoleAssignment -AccountName $accountName `  
    -ResourceGroupName $resourceGroupName `  
    -RoleDefinitionId $readOnlyRoleDefinitionId `  
    -Scope "/" `  
    -PrincipalId $principalId
```

Utilisation de l'interface de ligne de commande Azure (CLI)

Affecter un rôle à une identité :

Azure CLI

```
resourceGroupName='<myResourceGroup>'  
accountName='<myCosmosAccount>'  
readOnlyRoleDefinitionId='<roleDefinitionId>' # as fetched above  
# For Service Principals make sure to use the Object ID as found in the Enterprise  
applications section of the Azure Active Directory portal blade.  
principalId='<aadPrincipalId>'  
az cosmosdb sql role assignment create --account-name $accountName --resource-  
group $resourceGroupName --scope "/" --principal-id $principalId --role-  
definition-id $readOnlyRoleDefinitionId
```

Utilisation des modèles Bicep/Azure Resource Manager

Pour une affectation intégrée à l'aide d'un modèle Bicep :

```
resource sqlRoleAssignment  
'Microsoft.DocumentDB/databaseAccounts/sqlRoleAssignments@2023-04-15' = {  
  name: guid(<roleDefinitionId>, <aadPrincipalId>, <databaseAccountResourceId>)  
  parent: databaseAccount  
  properties:{  
    principalId: <aadPrincipalId>  
    roleDefinitionId:
```

```
'/${subscription().id}/resourceGroups/<databaseAccountResourceGroup>/providers/Microsoft.DocumentDB/databaseAccounts/<myCosmosAccount>/sqlRoleDefinitions/<roleDefinitionId>'  
    scope: <databaseAccountResourceId>  
}  
}
```

Pour obtenir une référence et des exemples d'utilisation de modèles Azure Resource Manager afin de créer des attributions de rôle, consultez [Microsoft.DocumentDB databaseAccounts/sqlRoleAssignments](#).

Initialiser le kit de développement logiciel (SDK) avec Microsoft Entra ID

Pour utiliser le contrôle d'accès en fonction du rôle Azure Cosmos DB dans votre application, vous devez mettre à jour la façon dont vous initialisez le kit de développement logiciel (SDK) Azure Cosmos DB. Au lieu de passer la clé principale de votre compte, vous devez passer une instance d'une classe `TokenCredential`. Cette instance fournit le SDK Azure Cosmos DB avec le contexte nécessaire pour récupérer un jeton Microsoft Entra au nom de l'identité que vous souhaitez utiliser.

La façon dont vous créez une instance de `TokenCredential` dépasse le cadre de cet article. Il existe de nombreuses façons de créer une telle instance selon le type d'identité Microsoft Entra que vous voulez utiliser (principal d'utilisateur, principal de service, groupe, etc.). Plus important encore, votre instance de `TokenCredential` doit se résoudre en l'identité (ID de principal) à laquelle vous avez affecté vos rôles. Vous trouverez des exemples de création d'une classe `TokenCredential` :

- [Dans .NET](#)
- [Dans Java](#)
- [En JavaScript](#)
- [En Python](#)

Les exemples suivant utilisent un principal de service avec une instance `ClientSecretCredential`.

Dans .NET

Le contrôle d'accès en fonction du rôle Azure Cosmos DB est actuellement pris en charge dans le [kit de développement logiciel \(SDK\) .NET V3](#).

C#

```
TokenCredential servicePrincipal = new ClientSecretCredential(  
    "<azure-ad-tenant-id>",  
    "<client-application-id>",
```

```
"<client-application-secret>");  
CosmosClient client = new CosmosClient("<account-endpoint>", servicePrincipal);
```

En Java

Le contrôle d'accès en fonction du rôle Azure Cosmos DB est actuellement pris en charge dans le [kit de développement logiciel \(SDK\) .Java V4](#).

Java

```
TokenCredential ServicePrincipal = new ClientSecretCredentialBuilder()  
    .authorityHost("https://login.microsoftonline.com")  
    .tenantId("<azure-ad-tenant-id>")  
    .clientId("<client-application-id>")  
    .clientSecret("<client-application-secret>")  
    .build();  
CosmosAsyncClient Client = new CosmosClientBuilder()  
    .endpoint("<account-endpoint>")  
    .credential(ServicePrincipal)  
    .build();
```

En JavaScript

Le contrôle d'accès en fonction du rôle Azure Cosmos DB est actuellement pris en charge dans le [kit de développement logiciel \(SDK\) .JavaScript V3](#).

JavaScript

```
const servicePrincipal = new ClientSecretCredential(  
    "<azure-ad-tenant-id>",  
    "<client-application-id>",  
    "<client-application-secret>");  
const client = new CosmosClient({  
    endpoint: "<account-endpoint>",  
    aadCredentials: servicePrincipal  
});
```

En Python

Le contrôle d'accès en fonction du rôle Azure Cosmos DB est pris en charge dans les [versions](#) du [kit de développement logiciel \(SDK\) Python 4.3.0b4](#) et ultérieures.

Python

```
aad_credentials = ClientSecretCredential(  
    tenant_id="<azure-ad-tenant-id>",  
    client_id="<client-application-id>",


```

```
client_secret=<client-application-secret>)
client = CosmosClient("⟨account-endpoint⟩", aad_credentials)
```

Authentifier des demandes sur l'API REST

Lors de la construction de l'[en-tête d'autorisation de l'API REST](#), définissez le paramètre **type** sur **Microsoft Entra** et la signature de hachage (**sig**) sur le **jeton OAuth**, comme indiqué dans l'exemple suivant :

```
type=aad&ver=1.0&sig=<token-from-oauth>
```

Utiliser l'explorateur de données

① Notes

L'explorateur de données exposé dans le portail Azure ne prend pas encore en charge le contrôle d'accès en fonction du rôle Azure Cosmos DB. Pour utiliser votre identité Microsoft Entra lors de l'exploration de vos données, vous devez utiliser l'[Explorateur Azure Cosmos DB](#) à la place.

Lorsque vous accédez à [Explorateur Azure Cosmos DB](#) avec le paramètre de requête `?feature.enableAadDataPlane=true` spécifique et vous connectez, la logique suivante est utilisée pour accéder à vos données :

1. Une demande d'extraction de la clé primaire du compte est tentée au nom de l'identité connectée. Si cette demande aboutit, la clé primaire est utilisée pour accéder aux données du compte.
2. Si l'identité connectée n'est pas autorisée à extraire la clé primaire du compte, cette identité est utilisée directement pour authentifier l'accès aux données. Dans ce mode, l'identité doit être [attribuée avec les définitions de rôle appropriées](#) pour garantir l'accès aux données.

Auditer les demandes de données

Les [journaux de diagnostic](#) sont complétés avec les informations d'identité et d'autorisation de chaque opération de données lors de l'utilisation du contrôle d'accès en fonction du rôle Azure Cosmos DB. Ce complément vous permet d'effectuer un audit détaillé et de récupérer l'identité Microsoft Entra utilisée pour chaque requête de données envoyée à votre compte Azure Cosmos DB.

Ces informations supplémentaires appartiennent à la catégorie de journalisation **DataPlaneRequests** et se composent de deux colonnes supplémentaires :

- `aadPrincipalId_g` montre l'ID du principal de l'identité Microsoft Entra qui a été utilisée pour authentifier la requête.
- `aadAppliedRoleAssignmentId_g` montre l'[attribution de rôle](#) qui a été respectée lors de l'autorisation de la demande.

Application du contrôle d'accès en fonction du rôle comme seule méthode d'authentification

Dans les situations où vous souhaitez forcer des clients à se connecter à Azure Cosmos DB exclusivement par un contrôle d'accès en fonction du rôle, vous pouvez désactiver les clés primaires/secondaires du compte. Si vous procédez ainsi, toute requête entrante utilisant une clé primaire/secondaire ou un jeton de ressource est activement rejetée.

Utiliser les modèles Azure Resource Manager

Lors de la création ou de la mise à jour de votre compte Azure Cosmos DB à l'aide de modèles Azure Resource Manager, définissez la propriété `disableLocalAuth` avec la valeur `true` :

JSON

```
"resources": [
  {
    "type": "Microsoft.DocumentDB/databaseAccounts",
    "properties": {
      "disableLocalAuth": true,
      // ...
    },
    // ...
  },
  // ...
]
```

Limites

- Vous pouvez créer jusqu'à 100 définitions de rôles et 2 000 attributions de rôles par compte Azure Cosmos DB.
- Vous pouvez attribuer des définitions de rôle aux identités Microsoft Entra appartenant au même tenant Microsoft Entra que votre compte Azure Cosmos DB.
- La résolution de groupe Microsoft Entra n'est actuellement pas prise en charge pour les identités qui appartiennent à plus de 200 groupes.
- Le jeton Microsoft Entra est actuellement transmis en tant qu'en-tête avec chaque requête individuelle envoyée au service Azure Cosmos DB, ce qui augmente la taille globale de la charge utile.

Forum aux questions

Cette section comprend des questions fréquemment posées sur le contrôle d'accès en fonction du rôle et sur Azure Cosmos DB.

Quelles API Azure Cosmos DB prennent en charge le contrôle d'accès en fonction du rôle ?

L'API pour NoSQL est prise en charge. La prise en charge de l'API pour MongoDB est en préversion.

Est-il possible de gérer les définitions de rôle et les attributions de rôles à partir du portail Azure ?

La prise en charge de la gestion du rôle par le portail Azure n'est pas encore disponible.

Quels kits de développement logiciel (SDK) de l'API NoSQL Azure Cosmos DB prennent en charge le contrôle d'accès en fonction du rôle ?

Les SDK [.NET V3](#), [Java V4](#), [JavaScript V3](#) et [Python V4.3+](#) sont actuellement pris en charge.

Le jeton Microsoft Entra est-il actualisé automatiquement par les SDK Azure Cosmos DB quand il expire ?

Oui.

Est-il possible de désactiver l'utilisation des clés primaires/secondaires du compte lors de l'utilisation du contrôle d'accès en fonction du rôle ?

Oui, consultez [Application du contrôle d'accès en fonction du rôle comme seule méthode d'authentification](#).

Étapes suivantes

- Obtenez une vue d'ensemble de l'[accès sécurisé aux données dans Cosmos DB](#).
- En savoir plus sur le [contrôle d'accès en fonction du rôle pour la gestion Azure Cosmos DB](#).

Comment utiliser une identité managée avec Azure Container Instances

Article • 09/03/2023

Utilisez les [identités managées pour les ressources Azure](#) pour exécuter du code dans Azure Container Instances interagissant avec d'autres services Azure, et ce sans avoir à gérer les codes secrets ou les informations d'identification dans le code. Cette fonctionnalité fournit un déploiement Azure Container Instances avec une identité managée automatiquement dans Microsoft Entra ID.

Dans cet article, vous en apprendrez davantage sur les identités managées dans Azure Container Instances, et vous découvrirez comment :

- ✓ Activer une identité attribuée par l'utilisateur ou par le système dans un groupe de conteneurs
- ✓ Autoriser l'identité à accéder à un coffre de clés Azure
- ✓ Utiliser l'identité managée pour accéder à un coffre de clés à partir d'un conteneur en cours d'exécution

Adaptez les exemples pour activer et utiliser des identités dans Azure Container Instances pour accéder aux autres services Azure. Ces exemples sont interactifs. Toutefois, en pratique, vos images de conteneur exécuteraient du code pour accéder aux services Azure.

Pourquoi utiliser une identité managée ?

Utilisez une identité managée dans un conteneur en cours d'exécution pour vous authentifier sur n'importe quel [service prenant en charge l'authentification Microsoft Entra](#), sans avoir à gérer les informations d'identification dans le code de votre conteneur. Pour les services qui ne prennent pas en charge l'authentification AD, vous pouvez stocker des codes secrets dans un coffre de clés Azure et utiliser l'identité managée pour accéder à celui-ci afin de récupérer les informations d'identification. Pour en savoir plus sur l'utilisation des identités managées, consultez la section [Que sont les identités managées pour les ressources Azure ?](#)

Activer une identité managée

Lorsque vous créez un groupe de conteneurs, activez une ou plusieurs identités managées en définissant une propriété [ContainerGroupIdentity](#). Vous pouvez également

activer ou mettre à jour des identités managées après l'exécution d'un groupe de conteneurs. Les deux actions entraînent le redémarrage du groupe de conteneurs. Pour définir les identités sur un nouveau groupe de conteneurs ou sur un groupe existant, utilisez Azure CLI, un modèle Resource Manager, un fichier YAML ou un autre outil Azure.

Azure Container Instances prend en charge les deux types d'identités Azure managées : attribuées par l'utilisateur et attribuées par le système. Sur un groupe de conteneurs, vous pouvez activer une identité attribuée par le système, une ou plusieurs identités attribuées par l'utilisateur, ou les deux types d'identités. Si vous n'êtes pas familiarisé avec les identités managées pour les ressources Azure, consultez la [présentation](#).

Utiliser une identité managée

Pour utiliser une identité managée, l'identité doit être autorisée à accéder à une ou plusieurs ressources de service Azure (par exemple, une application web, un coffre de clés ou un compte de stockage) dans l'abonnement. L'utilisation d'une identité managée dans un conteneur en cours d'exécution est semblable à l'utilisation d'une identité dans une machine virtuelle Azure. Consultez l'aide relative aux machines virtuelles pour en savoir plus sur l'utilisation d'un [jeton](#), d'[Azure PowerShell](#) ou [Azure CLI](#), ou des [kits de développement logiciel Azure](#).

Prérequis

- Utilisez l'environnement Bash dans [Azure Cloud Shell](#). Pour plus d'informations, consultez [Démarrage rapide pour Bash dans Azure Cloud Shell](#).

 [Launch Cloud Shell](#) 

- Si vous préférez exécuter les commandes de référence de l'interface de ligne de commande localement, [installez](#) l'interface Azure CLI. Si vous exécutez sur Windows ou macOS, envisagez d'exécuter Azure CLI dans un conteneur Docker. Pour plus d'informations, consultez [Guide pratique pour exécuter Azure CLI dans un conteneur Docker](#).
 - Si vous utilisez une installation locale, connectez-vous à Azure CLI à l'aide de la commande `az login`. Pour finir le processus d'authentification, suivez les étapes affichées dans votre terminal. Pour connaître les autres options de connexion, consultez [Se connecter avec Azure CLI](#).
 - Lorsque vous y êtes invité, installez l'extension Azure CLI lors de la première utilisation. Pour plus d'informations sur les extensions, consultez [Utiliser des](#)

extensions avec Azure CLI.

- Exécutez `az version` pour rechercher la version et les bibliothèques dépendantes installées. Pour effectuer une mise à niveau vers la dernière version, exécutez `az upgrade`.
- Cet article demande la version 2.0.49 ou ultérieure d'Azure CLI. Si vous utilisez Azure Cloud Shell, la version la plus récente est déjà installée.

Créer un coffre de clés Azure

Les exemples présentés dans cet article utilisent une identité managée dans Azure Container Instances pour accéder à un code secret du coffre de clés Azure.

Commencez par créer un groupe de ressources nommé *myResourceGroup* à l'emplacement *eastus* à l'aide de la commande `az group create` suivante :

Azure CLI

```
az group create --name myResourceGroup --location eastus
```

Utilisez la commande `az keyvault create` pour créer un coffre de clés. Veillez à spécifier un nom de coffre de clés unique.

Azure CLI

```
az keyvault create \
--name mykeyvault \
--resource-group myResourceGroup \
--location eastus
```

Stockez un exemple de secret dans le coffre de clés à l'aide de la commande `az keyvault secret set` :

Azure CLI

```
az keyvault secret set \
--name SampleSecret \
--value "Hello Container Instances" \
--description ACIsecret --vault-name mykeyvault
```

Utilisez les exemples suivants pour accéder au coffre de clés à l'aide d'une identité managée attribuée par l'utilisateur ou par le système dans Azure Container Instances.

Exemple 1 : Utiliser une identité attribuée par l'utilisateur pour accéder au coffre de clés Azure

Créer une identité

Créez d'abord une identité dans votre abonnement à l'aide de la commande [az identity create](#). Vous pouvez utiliser le groupe de ressources utilisé pour créer le coffre de clés, ou en utiliser un autre.

Azure CLI

```
az identity create \
--resource-group myResourceGroup \
--name myACIId
```

Pour utiliser l'identité dans les étapes suivantes, utilisez la commande [az identity show](#) pour stocker l'ID de principal de service et l'ID de ressource de l'identité dans des variables.

Azure CLI

```
# Get service principal ID of the user-assigned identity
SP_ID=$(az identity show \
--resource-group myResourceGroup \
--name myACIId \
--query principalId --output tsv)

# Get resource ID of the user-assigned identity
RESOURCE_ID=$(az identity show \
--resource-group myResourceGroup \
--name myACIId \
--query id --output tsv)
```

Autoriser l'identité attribuée par l'utilisateur à accéder au coffre de clés

Exécutez la commande [az keyvault set-policy](#) suivante pour définir une stratégie d'accès sur le coffre de clés. L'exemple suivant permet à l'identité affectée par l'utilisateur d'obtenir des secrets du coffre de clés :

Azure CLI

```
az keyvault set-policy \
--name mykeyvault \
--resource-group myResourceGroup \
--object-id $SP_ID \
--secret-permissions get
```

Activer une identité attribuée par l'utilisateur dans un groupe de conteneurs

Exécutez la commande [az container create](#) suivante pour créer une instance de conteneur basée sur une image `azure-cli` de Microsoft. Cet exemple fournit un groupe contenant un seul conteneur que vous pouvez utiliser de manière interactive pour accéder à d'autres services Azure. Dans cette section, seul le système d'exploitation de base est utilisé. Pour obtenir un exemple d'utilisation d'Azure CLI dans le conteneur, consultez [Activer l'identité attribuée par le système sur un groupe de conteneurs](#).

Le paramètre `--assign-identity` passe votre identité managée attribuée par l'utilisateur au groupe. Cette commande longue laisse le conteneur s'exécuter. Cet exemple utilise le groupe de ressources utilisé pour créer le coffre de clés, mais vous pouvez en spécifier un autre.

Azure CLI

```
az container create \
--resource-group myResourceGroup \
--name mycontainer \
--image mcr.microsoft.com/azure-cli \
--assign-identity $RESOURCE_ID \
--command-line "tail -f /dev/null"
```

Après quelques secondes, vous devez recevoir une réponse d'Azure CLI indiquant que le déploiement est terminé. Vérifiez son état à l'aide de la commande [az container show](#).

Azure CLI

```
az container show \
--resource-group myResourceGroup \
--name mycontainer
```

La section `identity` de la sortie ressemble à ce qui suit, elle montre la définition de l'identité dans le groupe de conteneurs. Le `principalID` sous `userAssignedIdentities` est le principal de service de l'identité que vous avez créée dans Microsoft Entra ID :

Sortie

```
[...]
"identity": {
    "principalId": "null",
    "tenantId": "xxxxxxxx-f292-4e60-9122-xxxxxxxxxxxx",
    "type": "UserAssigned",
    "userAssignedIdentities": {
        "/subscriptions/xxxxxxxx-0903-4b79-a55a-
xxxxxxxxxxxx/resourcegroups/danlep1018/providers/Microsoft.ManagedIdentity/u
serAssignedIdentities/myACIId": {
            "clientId": "xxxxxxxx-5523-45fc-9f49-xxxxxxxxxxxx",
            "principalId": "xxxxxxxx-f25b-4895-b828-xxxxxxxxxxxx"
        }
    }
},
[...]
```

Utiliser une identité attribuée par l'utilisateur pour obtenir un secret du coffre de clés

Vous pouvez désormais utiliser l'identité managée dans l'instance de conteneur en cours d'exécution pour accéder au coffre de clés. Commencez par lancer un shell bash dans le conteneur :

Azure CLI

```
az container exec \
--resource-group myResourceGroup \
--name mycontainer \
--exec-command "/bin/bash"
```

Exécutez les commandes suivantes dans le shell bash du conteneur. Pour obtenir un jeton d'accès permettant d'utiliser Microsoft Entra ID pour s'authentifier auprès du coffre de clés, exécutez la commande suivante :

Bash

```
client_id="xxxxxxxx-5523-45fc-9f49-xxxxxxxxxxxx"
curl "http://169.254.169.254/metadata/identity/oauth2/token?api-
version=2018-02-
01&resource=https%3A%2F%2Fvault.azure.net&client_id=$client_id" -H
Metadata:true -s
```

Sortie :

Bash

```
{"access_token": "xxxxxxxxxxxxxxxxxxxxxxxxxxxxx1QiLCJhbGciOiJSUzI1NiIsIng1dCI6Imk2bEdrM0ZaenhSY1ViMkMzbkVRN3N5SEpsWSIsImtpZCI6Imk2bEdrM0ZaenhSY1ViMkMzbkVRN3N5SEpsWSJ9.....xxxxxxxxxxxxxxxxxx", "refresh_token": "", "expires_in": "28799", "expires_on": "1539927532", "not_before": "1539898432", "resource": "https://vault.azure.net/", "token_type": "Bearer"}
```

Pour stocker le jeton d'accès dans une variable qui pourra être utilisée dans les prochaines commandes pour l'authentification, exécutez la commande suivante :

Bash

```
TOKEN=$(curl 'http://169.254.169.254/metadata/identity/oauth2/token?api-version=2018-02-01&resource=https%3A%2F%2Fvault.azure.net' -H Metadata:true | jq -r '.access_token')
```

Utilisez maintenant le jeton d'accès pour vous authentifier auprès du coffre de clés et lire un secret. Veillez à remplacer le nom de votre coffre de clés dans l'URL ([https://mykeyvault.vault.azure.net/...](https://mykeyvault.vault.azure.net/)):

Bash

```
curl https://mykeyvault.vault.azure.net/secrets/SampleSecret/?api-version=7.4 -H "Authorization: Bearer $TOKEN"
```

La réponse ressemble à ce qui suit, elle affiche le code secret. Dans votre code, il vous faudrait analyser cette sortie pour obtenir le code secret. Utilisez ensuite le code secret dans une opération ultérieure pour accéder à une autre ressource Azure.

Bash

```
{"value": "Hello Container Instances", "contentType": "ACIsecret", "id": "https://mykeyvault.vault.azure.net/secrets/SampleSecret/xxxxxxxxxxxxxxxxxxxx", "attributes": {"enabled": :true, "created": 1539965967, "updated": 1539965967, "recoveryLevel": "Purgeable"}, "tags": {"file-encoding": "utf-8"}}
```

Exemple 2 : Utiliser une identité attribuée par le système pour accéder à un coffre de clés Azure

Activer une identité attribuée par le système dans un groupe de conteneurs

Exécutez la commande `az container create` suivante pour créer une instance de conteneur basée sur une image `azure-cli` de Microsoft. Cet exemple fournit un groupe contenant un seul conteneur que vous pouvez utiliser de manière interactive pour accéder à d'autres services Azure.

Le paramètre `--assign-identity`, sans valeur supplémentaire, active une identité managée attribuée par le système dans le groupe. L'identité est limitée au groupe de ressources du groupe de conteneurs. Cette commande longue laisse le conteneur s'exécuter. Cet exemple utilise le même groupe de ressources que celui utilisé pour créer le coffre de clés, qui se trouve dans l'étendue de l'identité.

Azure CLI

```
# Get the resource ID of the resource group
RG_ID=$(az group show --name myResourceGroup --query id --output tsv)

# Create container group with system-managed identity
az container create \
--resource-group myResourceGroup \
--name mycontainer \
--image mcr.microsoft.com/azure-cli \
--assign-identity --scope $RG_ID \
--command-line "tail -f /dev/null"
```

Après quelques secondes, vous devez recevoir une réponse d'Azure CLI indiquant que le déploiement est terminé. Vérifiez son état à l'aide de la commande `az container show`.

Azure CLI

```
az container show \
--resource-group myResourceGroup \
--name mycontainer
```

La section `identity` de la sortie ressemble à ce qui suit. Elle montre qu'une identité attribuée par le système a été créée dans Microsoft Entra ID :

Sortie

```
[...]
"identity": {
    "principalId": "xxxxxxxx-528d-7083-b74c-xxxxxxxxxxxx",
    "tenantId": "xxxxxxxx-f292-4e60-9122-xxxxxxxxxxxx",
    "type": "SystemAssigned",
```

```
        "userAssignedIdentities": null  
    },  
    [...]
```

Définissez une variable sur la valeur `principalId` (l'ID du principal de service) de l'identité, pour une utilisation ultérieure.

Azure CLI

```
SP_ID=$(az container show \  
    --resource-group myResourceGroup \  
    --name mycontainer \  
    --query identity.principalId --out tsv)
```

Autoriser le groupe de conteneurs à accéder au coffre de clés

Exécutez la commande `az keyvault set-policy` suivante pour définir une stratégie d'accès sur le coffre de clés. L'exemple suivant permet à l'identité managée attribuée par le système d'obtenir des secrets du coffre de clés :

Azure CLI

```
az keyvault set-policy \  
    --name mykeyvault \  
    --resource-group myResourceGroup \  
    --object-id $SP_ID \  
    --secret-permissions get
```

Utiliser une identité de groupe de conteneurs pour obtenir un secret du coffre de clés

Maintenant, vous pouvez utiliser l'identité managée pour accéder au coffre de clés dans l'instance de conteneur en cours d'exécution. Commencez par lancer un shell bash dans le conteneur :

Azure CLI

```
az container exec \  
    --resource-group myResourceGroup \  
    --name mycontainer \  
    --exec-command "/bin/bash"
```

Exécutez les commandes suivantes dans le shell bash du conteneur. Connectez-vous d'abord à l'interface Azure CLI à l'aide de l'identité managée :

```
Azure CLI
```

```
az login --identity
```

À partir du conteneur en cours d'exécution, récupérez le secret du coffre de clés :

```
Azure CLI
```

```
az keyvault secret show \
--name SampleSecret \
--vault-name mykeyvault --query value
```

La valeur du secret est récupérée :

```
Sortie
```

```
"Hello Container Instances"
```

Activer une identité managée à l'aide d'un modèle Resource Manager

Pour activer une identité managée dans un groupe de conteneurs à l'aide d'un [modèle Resource Manager](#), définissez la propriété `identity` de l'objet

`Microsoft.ContainerInstance/containerGroups` avec un objet `ContainerGroupIdentity`. Dans les extraits de code suivants, la propriété `identity` est configurée pour différents scénarios. Consultez les [références sur les modèles Resource Manager](#). Spécifiez une version `apiVersion` minimale de `2018-10-01`.

Identité attribuée par l'utilisateur

Une identité attribuée par l'utilisateur est un ID de ressource qui se présente sous cette forme :

```
"/subscriptions/{subscriptionId}/resourceGroups/{resourceGroupName}/providers/Microsoft.ManagedIdentity/userAssignedIdentities/{identityName}"
```

Vous pouvez activer une ou plusieurs identités attribuées par l'utilisateur.

JSON

```
"identity": {  
    "type": "UserAssigned",  
    "userAssignedIdentities": {  
        "myResourceID1": {}  
    }  
}
```

Identité attribuée par le système

JSON

```
"identity": {  
    "type": "SystemAssigned"  
}
```

Identités attribuées par l'utilisateur et par le système

Sur un groupe de conteneurs, vous pouvez activer à la fois une identité attribuée par le système, et une ou plusieurs identités attribuées par l'utilisateur.

JSON

```
"identity": {  
    "type": "System Assigned, UserAssigned",  
    "userAssignedIdentities": {  
        "myResourceID1": {}  
    }  
}  
...  
}
```

Activer une identité managée à l'aide du fichier YAML

Pour activer une identité managée dans un groupe de conteneurs déployé à l'aide d'un [fichier YAML](#), incluez le code YAML suivant. Spécifiez une version `apiVersion` minimale de `2018-10-01`.

Identité attribuée par l'utilisateur

Une identité attribuée par l'utilisateur est un ID de ressource qui se présente sous la forme

```
'/subscriptions/{subscriptionId}/resourceGroups/{resourceGroupName}/providers/Microsoft.ManagedIdentity/userAssignedIdentities/{identityName}'
```

Vous pouvez activer une ou plusieurs identités attribuées par l'utilisateur.

YAML

```
identity:
  type: UserAssigned
  userAssignedIdentities:
    - 'myResourceId1':{}}
```

Identité attribuée par le système

YAML

```
identity:
  type: SystemAssigned
```

Identités attribuées par l'utilisateur et par le système

Sur un groupe de conteneurs, vous pouvez activer à la fois une identité attribuée par le système, et une ou plusieurs identités attribuées par l'utilisateur.

yml

```
identity:
  type: SystemAssigned, UserAssigned
  userAssignedIdentities:
    - 'myResourceId1':{}}
```

Étapes suivantes

Dans cet article, vous en avez appris davantage sur les identités managées dans Azure Container Instances, et vous avez découvert comment :

- ✓ Activer une identité attribuée par l'utilisateur ou par le système dans un groupe de conteneurs
- ✓ Autoriser l'identité à accéder à un coffre de clés Azure
- ✓ Utiliser l'identité managée pour accéder à un coffre de clés à partir d'un conteneur en cours d'exécution
 - En savoir plus sur les [identités managées pour les ressources Azure](#).
 - Consultez un [exemple de Kit de développement logiciel \(SDK\) Azure Go](#) dans lequel une identité managée est utilisée pour accéder à un coffre de clés Key Vault à partir d'Azure Container Instances.

Guide du développeur Azure Service Bus JMS 2.0

Article • 04/05/2023

Ce guide contient des informations détaillées pour vous aider à bien communiquer avec Azure Service Bus à l'aide de l'API Java Message Service (JMS) 2.0.

En tant que développeur Java, si vous débutez avec Azure Service Bus, nous vous recommandons de consulter les articles ci-dessous.

Prise en main	Concepts
<ul style="list-style-type: none">Présentation d'Azure Service BusFiles d'attente, rubriques et abonnements	<ul style="list-style-type: none">Niveau Premium d'Azure Service Bus

Modèle de programmation JMS (Java Message Service)

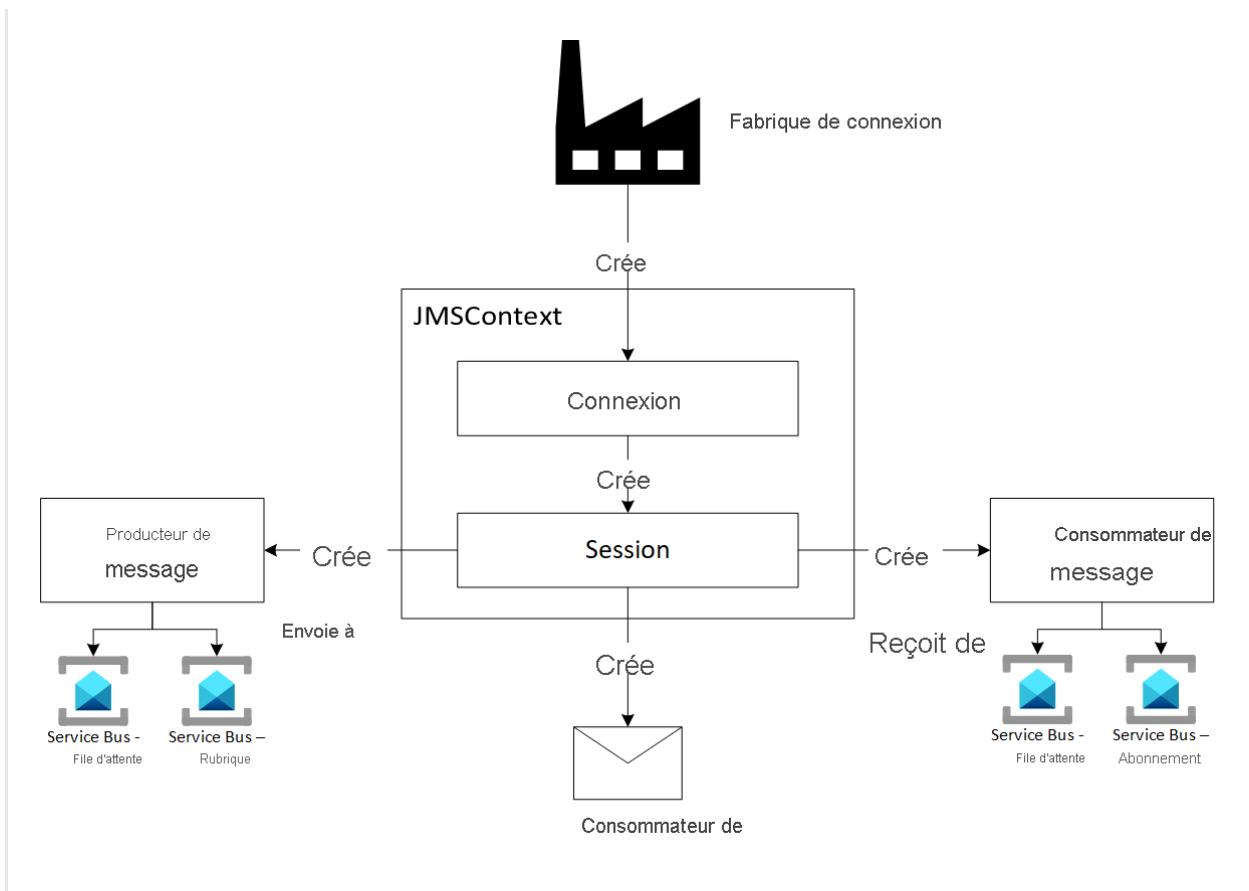
Le modèle de programmation de l'API Java Message Service est illustré ci-dessous :

ⓘ Notes

Le niveau Premium d'Azure Service Bus prend en charge JMS 1.1 et JMS 2.0.

Azure Service Bus niveau Standard prend en charge des fonctionnalités limitées de JMS 1.1. Pour plus d'informations, consultez cette [documentation](#).

Modèle de programmation JMS 2.0



JMS : blocs de construction

Les blocs de construction ci-dessous sont disponibles pour communiquer avec l'application JMS.

ⓘ Notes

Le guide ci-dessous est basé sur le [Tutoriel Oracle Java EE 6 pour Java Message Service \(JMS\)](#)

Nous vous recommandons de vous référer à ce tutoriel pour mieux comprendre le service JMS (Java Message Service).

Fabrique de connexion

L'objet de fabrique de connexion est utilisé par le client pour se connecter au fournisseur JMS. La fabrique de connexion encapsule un ensemble de paramètres de configuration de la connexion définis par l'administrateur.

Chaque fabrique de connexion est une instance de l'interface `ConnectionFactory`, `QueueConnectionFactory` OU `TopicConnectionFactory`.

Pour simplifier la connexion avec Azure Service Bus, ces interfaces sont implémentées respectivement via `ServiceBusJmsConnectionFactory`, `ServiceBusJmsQueueConnectionFactory` et `ServiceBusJmsTopicConnectionFactory`.

ⓘ Important

Les applications Java utilisant l'API JMS 2.0 peuvent se connecter à Azure Service Bus au moyen de la chaîne de connexion ou d'un `TokenCredential` pour tirer parti de l'authentification adossée à Microsoft Entra. Lorsque vous utilisez l'authentification adossée à Microsoft Entra, assurez-vous d'**attribuer des rôles et des autorisations** à l'identité au besoin.

Identité managée affectée par le système

Créez une [identité managée affectée par le système](#) dans Azure et utilisez-la pour créer un `TokenCredential`.

Java

```
TokenCredential tokenCredential = new  
DefaultAzureCredentialBuilder().build();
```

La fabrique de connexion peut ensuite être instanciée avec les paramètres ci-dessous.

- Informations d'identification du jeton : Cela représente une information d'identification capable de fournir un jeton OAuth.
- Host : le nom d'hôte de l'espace de noms Azure Service Bus niveau Premium.
- Jeu de propriétés `ServiceBusJmsConnectionFactorySettings` comprenant
 - `connectionIdleTimeoutMS` : délai d'expiration de connexion inactive en millisecondes.
 - `traceFrames` : indicateur booléen permettant de collecter des frames de trace de protocole AMQP pour le débogage.
 - *Autres paramètres de configuration*

La fabrique peut être créée comme indiqué ci-après. Les informations d'identification du jeton et l'hôte sont des paramètres obligatoires, tandis que les autres propriétés sont facultatives.

Java

```
String host = "<YourNamespaceName>.servicebus.windows.net";
ConnectionFactory factory = new
ServiceBusJmsConnectionFactory(tokenCredential, host, null);
```

Destination JMS

Une destination est l'objet qu'un client utilise pour spécifier la cible des messages qu'il génère et la source des messages qu'il consomme.

Les destinations sont mappées vers les entités dans les files d'attente (dans les scénarios point à point) et les rubriques (dans les scénarios publication-abonnement) d'Azure Service Bus.

Connexions

Une connexion encapsule une connexion virtuelle avec un fournisseur JMS. Avec Azure Service Bus, cela représente une connexion avec état entre l'application et Azure Service Bus au moyen de AMQP.

Une connexion est créée à partir de la fabrique de connexion comme indiqué ci-dessous.

Java

```
Connection connection = factory.createConnection();
```

Sessions

Une session est un contexte à thread unique pour la production et la consommation de messages. La session peut être utilisée pour créer des messages, des producteurs et des consommateurs de messages, mais elle fournit également un contexte transactionnel pour permettre le regroupement d'envois et les réceptions dans une unité de travail atomique.

Une session peut être créée à partir de l'objet de connexion comme indiqué ci-dessous.

Java

```
Session session = connection.createSession(false,
Session.CLIENT_ACKNOWLEDGE);
```

ⓘ Notes

L'API JMS ne prend pas en charge la réception de messages à partir de files d'attente ou de rubriques Service Bus ayant des sessions de messagerie activées.

Modes de session

Une session peut être créée avec l'un des modes ci-dessous.

Modes de session	Comportement
Session.AUTO_ACKNOWLEDGE	Cette session accuse automatiquement réception d'un message par un client quand la session a été retournée avec succès à partir d'un appel pour recevoir ou quand l'écouteur de message que la session a appelé pour traiter le message est retourné avec succès.
Session.CLIENT_ACKNOWLEDGE	Le client accuse réception d'un message consommé en appelant la méthode d'accusé de réception du message.
Session.DUPS_OK_ACKNOWLEDGE	Ce mode d'accusé de réception demande à la session d'accuser réception de la remise des messages en différé.
Session.SESSION_TRANSACTED	Cette valeur peut être passée en tant qu'argument à la méthode createSession(int sessionMode) sur l'objet de connexion pour spécifier que la session doit utiliser une transaction locale.

Quand le mode de session n'est pas spécifié, `Session.AUTO_ACKNOWLEDGE` est la valeur sélectionnée par défaut.

JMSContext

ⓘ Notes

JMSContext est défini dans le cadre de la spécification JMS 2.0.

JMSContext combine les fonctionnalités fournies par l'objet de connexion et de session. Il peut être créé à partir de l'objet de fabrique de connexion.

Java

```
JMSContext context = connectionFactory.createContext();
```

Modes JMSContext

Tout comme l'objet **Session**, JMSContext peut être créé avec les mêmes modes d'accusation de réception que ceux indiqués dans les [Modes de session](#).

Java

```
JMSContext context =  
connectionFactory.createContext(JMSContext.AUTO_ACKNOWLEDGE);
```

Quand le mode n'est pas spécifié, **JMSContext.AUTO_ACKNOWLEDGE** est la valeur sélectionnée par défaut.

Producteurs de messages JMS

Un producteur de messages est un objet créé à l'aide de JMSContext ou d'une session et utilisé pour envoyer des messages à une destination.

Il peut être créé en tant qu'objet autonome comme indiqué ci-après :

Java

```
JMSProducer producer = context.createProducer();
```

ou créé au moment de l'exécution quand un message doit être envoyé.

Java

```
context.createProducer().send(destination, message);
```

Consommateurs de messages JMS

Un consommateur de messages est un objet créé par JMSContext ou une session et utilisé pour recevoir des messages envoyés à une destination. Il peut être créé comme indiqué ci-dessous :

Java

```
JMSConsumer consumer = context.createConsumer(dest);
```

Réceptions synchrones via la méthode receive()

Le consommateur de messages fournit un moyen synchrone de recevoir des messages à partir de la destination via la méthode `receive()`.

Si aucun argument/délai d'expiration n'est spécifié ou si un délai d'attente de « 0 » est spécifié, le consommateur se bloque indéfiniment sauf si le message arrive ou si la connexion est interrompue (selon ce qui se produit en premier).

Java

```
Message m = consumer.receive();
Message m = consumer.receive(0);
```

Quand un argument positif différent de zéro est fourni, le consommateur se bloque jusqu'à ce que le retardateur expire.

Java

```
Message m = consumer.receive(1000); // time out after one second.
```

Réceptions asynchrones avec les écouteurs de messages JMS

Un écouteur de messages est un objet utilisé pour la gestion asynchrone des messages sur une destination. Il implémente l'interface `MessageListener` qui contient la méthode `onMessage` dans laquelle la logique métier spécifique doit se trouver.

Un objet écouteur de message doit être instancié et inscrit auprès d'un consommateur de messages spécifique à l'aide de la méthode `setMessageListener`.

Java

```
Listener myListener = new Listener();
consumer.setMessageListener(myListener);
```

Utilisation à partir de rubriques

Les [consommateurs de messages JMS](#) sont créés par rapport à une [destination](#) qui peut être une file d'attente ou une rubrique.

Les consommateurs sur les files d'attente sont simplement des objets côté client qui se trouvent dans le contexte de la session (et de la connexion) entre l'application cliente et Azure Service Bus.

En revanche, les consommateurs sur les rubriques ont 2 parties :

- Un **objet côté client** qui réside dans le contexte de la session (ou JMSContext) et
- Un **abonnement** qui est une entité sur Azure Service Bus.

Les abonnements sont documentés [ici](#) et peuvent être l'un des suivants :

- Abonnements durables partagés
- Abonnements non durables partagés
- Abonnements durables non partagés
- Abonnements non durables non partagés

Explorateurs de files d'attente JMS

L'API JMS fournit un objet `QueueBrowser` qui permet à l'application de parcourir les messages de la file d'attente et d'afficher les valeurs d'en-tête pour chaque message.

Un explorateur de file d'attente peut être créé à l'aide de JMSContext comme ci-dessous.

Java

```
QueueBrowser browser = context.createBrowser(queue);
```

ⓘ Notes

L'API JMS ne fournit pas d'API permettant de parcourir une rubrique.

Cela est dû au fait que la rubrique elle-même ne stocke pas les messages. Dès que le message est envoyé à la rubrique, il est transféré aux abonnements appropriés.

Sélecteurs de messages JMS

Les sélecteurs de messages peuvent être utilisés par les applications réceptrices pour filtrer les messages reçus. Avec les sélecteurs de message, l'application de réception décharge la tâche de filtrage des messages au fournisseur JMS (dans ce cas, Azure Service Bus) au lieu de s'en charger elle-même.

Les sélecteurs peuvent être utilisés lors de la création de l'un des consommateurs ci-dessous :

- Abonnement durable partagé

- Abonnement durable non partagé
- Abonnement non durable partagé
- Abonnement non durable non partagé
- Explorateur de files d'attente

Mappage entre une disposition AMQP et une opération Service Bus

Voici comment une disposition AMQP se traduit par une opération Service Bus :

Output

```
ACCEPTED = 1; -> Complete()
REJECTED = 2; -> DeadLetter()
RELEASED = 3; (just unlock the message in service bus, will then get
redelivered)
MODIFIED_FAILED = 4; -> Abandon() which increases delivery count
MODIFIED_FAILED_UNDELIVERABLE = 5; -> Defer()
```

Résumé

Ce guide du développeur vous a présenté les différentes connexions possibles entre Azure Service Bus et les applications clientes Java utilisant Java Message Service (JMS).

Étapes suivantes

Pour obtenir plus d'informations sur Azure Service Bus et de détails sur les entités Java Message Service (JMS), suivez les liens ci-dessous :

- [Service Bus - Files d'attente, rubriques et abonnements](#)
- [Service Bus - entités Java Message Service](#)
- [Prise en charge d'AMQP 1.0 dans Service Bus](#)
- [Guide du développeur sur l'utilisation de Service Bus avec AMQP 1.0](#)
- [Prise en main des files d'attente Service Bus](#)
- [API Java Message Service \(doc Oracle externe\) ↗](#)
- [Découvrez comment migrer d'ActiveMQ vers Service Bus](#)

Tutoriel : Déployer une application Spring sur Azure Spring Apps avec une connexion sans mot de passe à une base de données Azure

Article • 23/10/2023

Cet article explique comment utiliser des connexions sans mot de passe aux bases de données Azure dans les applications Spring Boot déployées sur Azure Spring Apps.

Dans ce tutoriel, vous allez effectuer les tâches suivantes à l'aide du portail Azure ou d'Azure CLI. Les deux méthodes sont expliquées dans les procédures suivantes.

- ✓ Provisionnez une instance d'Azure Spring Apps.
- ✓ Créez et déployez des applications sur Azure Spring Apps.
- ✓ Exécutez des applications connectées à des bases de données Azure à l'aide d'une identité managée.

ⓘ Notes

Ce didacticiel ne fonctionne pas pour R2DBC.

Prérequis

- Un abonnement Azure. Si vous n'en avez pas encore, créez un [compte gratuit](#) avant de commencer.
- [Azure CLI](#) 2.45.0 ou version ultérieure requise.
- L'extension Azure Spring Apps. Vous pouvez installer l'extension à l'aide de la commande : `az extension add --name spring`.
- [Kit de développement Java \(JDK\)](#), version 8, 11 ou 17.
- Un client [Git](#).
- [cURL](#) ou un utilitaire HTTP similaire pour tester la fonctionnalité.
- Client de ligne de commande MySQL si vous choisissez d'exécuter Azure Database pour MySQL. Vous pouvez vous connecter à votre serveur avec Azure Cloud Shell à l'aide d'un outil client populaire, l'outil [en ligne de commande mysql.exe](#). Vous pouvez aussi utiliser la ligne de commande de `mysql` dans votre environnement local.
- [ODBC Driver 18 pour SQL Server](#) si vous choisissez d'exécuter Azure SQL Database.

Préparer l'environnement de travail

Tout d'abord, configurez des variables d'environnement à l'aide des commandes suivantes :

Bash

```
export AZ_RESOURCE_GROUP=passwordless-tutorial-rg
export AZ_DATABASE_SERVER_NAME=<YOUR_DATABASE_SERVER_NAME>
export AZ_DATABASE_NAME=demodb
export AZ_LOCATION=<YOUR_AZURE_REGION>
export AZ_SPRING_APPS_SERVICE_NAME=<YOUR_AZURE_SPRING_APPS_SERVICE_NAME>
export AZ_SPRING_APPS_APP_NAME=hellospring
export AZ_DB_ADMIN_USERNAME=<YOUR_DB_ADMIN_USERNAME>
export AZ_DB_ADMIN_PASSWORD=<YOUR_DB_ADMIN_PASSWORD>
export AZ_USER_IDENTITY_NAME=<YOUR_USER_ASSIGNED_MANAGEMED_IDENTITY_NAME>
```

Remplacez les espaces réservés par les valeurs suivantes, qui sont utilisées dans cet article :

- <YOUR_DATABASE_SERVER_NAME> : nom de votre serveur Azure Database, qui doit être unique sur Azure.
- <YOUR_AZURE_REGION> : région Azure que vous souhaitez utiliser. Vous pouvez utiliser `eastus` par défaut, mais nous vous recommandons de configurer une région plus proche de l'endroit où vous vivez. Vous pouvez voir la liste complète des régions disponibles à l'aide `az account list-locations` de .
- <YOUR_AZURE_SPRING_APPS_SERVICE_NAME> : nom de votre instance Azure Spring Apps. Le nom doit comporter entre 4 et 32 caractères, et contenir uniquement des lettres minuscules, des chiffres et des traits d'union. Le premier caractère du nom du service doit être une lettre, et le dernier doit être une lettre ou un chiffre.
- <AZ_DB_ADMIN_USERNAME> : nom d'utilisateur administrateur de votre serveur de base de données Azure.
- <AZ_DB_ADMIN_PASSWORD> : mot de passe administrateur de votre serveur de base de données Azure.
- <YOUR_USER_ASSIGNED_MANAGEMED_IDENTITY_NAME> : Le nom du serveur d'identité managée affectée par l'utilisateur, qui doit être unique dans tout Azure.

Provisionner une instance d'Azure Spring Apps

Procédez comme suit pour provisionner une instance d'Azure Spring Apps.

1. Mettez à jour Azure CLI avec l'extension Azure Spring Apps à l'aide de la commande suivante :

Azure CLI

```
az extension update --name spring
```

2. Connectez-vous à Azure CLI et choisissez votre abonnement actif à l'aide des commandes suivantes :

Azure CLI

```
az login  
az account list --output table  
az account set --subscription <name-or-ID-of-subscription>
```

3. Utilisez les commandes suivantes pour créer un groupe de ressources pour contenir votre service Azure Spring Apps et une instance du service Azure Spring Apps :

Azure CLI

```
az group create \  
  --name $AZ_RESOURCE_GROUP \  
  --location $AZ_LOCATION  
az spring create \  
  --resource-group $AZ_RESOURCE_GROUP \  
  --name $AZ_SPRING_APPS_SERVICE_NAME
```

Créer une instance de base de données Azure

Procédez comme suit pour provisionner une instance Azure Database.

Azure SQL Database

1. Créez un serveur Azure SQL Database à l'aide de la commande suivante :

Azure CLI

```
az sql server create \  
  --location $AZ_LOCATION \  
  --resource-group $AZ_RESOURCE_GROUP \  
  --name $AZ_DATABASE_SERVER_NAME \  
  --admin-user $AZ_DB_ADMIN_USERNAME \  
  --admin-password $AZ_DB_ADMIN_PASSWORD
```

2. Le serveur SQL est vide. Créez donc une base de données à l'aide de la commande suivante :

Azure CLI

```
az sql db create \
--resource-group $AZ_RESOURCE_GROUP \
--server $AZ_DATABASE_SERVER_NAME \
--name $AZ_DATABASE_NAME
```

Créer une application avec un point de terminaison public affecté

Utilisez la commande suivante pour créer l'application.

Azure CLI

```
az spring app create \
--resource-group $AZ_RESOURCE_GROUP \
--service $AZ_SPRING_APPS_SERVICE_NAME \
--name $AZ_SPRING_APPS_APP_NAME \
--runtime-version=Java_17
--assign-endpoint true
```

Connecter Azure Spring Apps à la base de données Azure

Tout d'abord, installez l'extension [sans mot de passe du service Connecter or](#) pour Azure CLI :

Azure CLI

```
az extension add --name serviceconnector-passwordless --upgrade
```

Azure SQL Database

ⓘ Notes

Assurez-vous qu'Azure CLI utilise Python 64 bits, Python 32 bits présente un problème de compatibilité avec le pyodbc [de dépendance ↗](#) de la commande.

Les informations Python d'Azure CLI peuvent être obtenues avec la commande `az --version`. S'il s'affiche [MSC v.1929 32 bit (Intel)], cela signifie qu'il utilise Python 32 bits. La solution consiste à installer Python 64 bits et à installer Azure CLI à partir de [PyPI](#).

Utilisez la commande suivante pour créer une connexion sans mot de passe à la base de données.

Azure CLI

```
az spring connection create sql \
--resource-group $AZ_RESOURCE_GROUP \
--service $AZ_SPRING_APPS_SERVICE_NAME \
--app $AZ_SPRING_APPS_APP_NAME \
--target-resource-group $AZ_RESOURCE_GROUP \
--server $AZ_DATABASE_SERVER_NAME \
--database $AZ_DATABASE_NAME \
--system-identity
```

Cette commande de connecteur de services effectue les tâches suivantes en arrière-plan :

- Activez l'identité managée affectée par le système pour l'application `$AZ_SPRING_APPS_APP_NAME` hébergée par Azure Spring Apps.
- Définissez l'administrateur Microsoft Entra sur l'utilisateur de connexion actuel.
- Ajoutez un utilisateur de base de données nommé `$AZ_SPRING_APPS_SERVICE_NAME/apps/$AZ_SPRING_APPS_APP_NAME` pour l'identité managée créée à l'étape 1 et accordez tous les priviléges de la base de données `$AZ_DATABASE_NAME` à cet utilisateur.
- Ajoutez une configuration à l'application `$AZ_SPRING_APPS_APP_NAME`:
`spring.datasource.url`.

ⓘ Notes

Si le message d'erreur `The subscription is not registered to use Microsoft.ServiceLinker` apparaît, exécutez la commande `az provider register --namespace Microsoft.ServiceLinker` pour enregistrer le

fournisseur de ressources du connecteur de services, puis exécutez à nouveau la commande de connexion.

Génération et déploiement de l'application

Les étapes suivantes décrivent comment télécharger, configurer, générer et déployer l'exemple d'application.

1. Utilisez la commande suivante pour cloner l'exemple de référentiel de code :

```
Azure SQL Database

Bash

git clone https://github.com/Azure-Samples/quickstart-spring-data-jdbc-sql-server-passwordless-sample
```

2. Ajoutez la dépendance suivante à votre *fichier pom.xml* :

```
Azure SQL Database

XML

<dependency>
    <groupId>com.azure</groupId>
    <artifactId>azure-identity</artifactId>
    <version>1.5.4</version>
</dependency>
```

Il n'existe actuellement aucun démarrage Azure Spring Cloud pour Azure SQL Database, mais la `azure-identity` dépendance est requise.

3. Utilisez la commande suivante pour mettre à jour le *fichier application.properties* :

```
Azure SQL Database

Bash

cat << EOF > passwordless-
sample/src/main/resources/application.properties
```

```
logging.level.org.springframework.jdbc.core=DEBUG  
spring.sql.init.mode=always
```

```
EOF
```

4. Utilisez les commandes suivantes pour générer le projet à l'aide de Maven :

Bash

```
cd passwordless-sample  
.mvnw clean package -DskipTests
```

5. Utilisez la commande suivante pour déployer le *fichier target/demo-0.0.1-SNAPSHOTjar* pour l'application :

Azure CLI

```
az spring app deploy \  
--name $AZ_SPRING_APPS_APP_NAME \  
--service $AZ_SPRING_APPS_SERVICE_NAME \  
--resource-group $AZ_RESOURCE_GROUP \  
--artifact-path target/demo-0.0.1-SNAPSHOT.jar
```

6. Interrogez l'état de l'application après le déploiement à l'aide de la commande suivante :

Azure CLI

```
az spring app list \  
--service $AZ_SPRING_APPS_SERVICE_NAME \  
--resource-group $AZ_RESOURCE_GROUP \  
--output table
```

Vous devez voir la sortie similaire à l'exemple suivant.

Name	Location	ResourceGroup	Production Deployment	
Public Url			Provisioning	
Status	CPU	Memory	Running Instance	Registered Instance
Persistent Storage				
<app name>	eastus	<resource group>	default	

Test de l'application

Pour tester l'application, vous pouvez utiliser cURL. Tout d'abord, créez un élément « todo » dans la base de données à l'aide de la commande suivante :

Bash

```
curl --header "Content-Type: application/json" \
--request POST \
--data '{"description":"configuration","details":"congratulations, you
have set up JDBC correctly!","done": "true"}' \
https://${AZ_SPRING_APPS_SERVICE_NAME}-
hellospring.azuremicroservices.io
```

Cette commande retourne l'élément créé, comme illustré dans l'exemple suivant :

JSON

```
{"id":1,"description":"configuration","details":"congratulations, you have
set up JDBC correctly!","done":true}
```

Ensuite, récupérez les données à l'aide de la requête cURL suivante :

Bash

```
curl https://${AZ_SPRING_APPS_SERVICE_NAME}-
hellospring.azuremicroservices.io
```

Cette commande retourne la liste des éléments « todo », y compris l'élément que vous avez créé, comme illustré dans l'exemple suivant :

JSON

```
[{"id":1,"description":"configuration","details":"congratulations, you have
set up JDBC correctly!","done":true}]
```

Nettoyer les ressources

Pour propre toutes les ressources utilisées pendant ce didacticiel, supprimez le groupe de ressources à l'aide de la commande suivante :

Azure CLI

```
az group delete \
--name $AZ_RESOURCE_GROUP \
--yes
```

Étapes suivantes

- Documentation Spring Cloud Azure

Utiliser Spring Data JDBC avec Azure Database pour MySQL

Article • 23/10/2023

Ce tutoriel montre comment stocker des données dans [Azure Database pour MySQL](#) base de données à l'aide [de Spring Data JDBC](#).

[JDBC](#) est l'API Java standard pour se connecter à des bases de données relationnelles traditionnelles.

Dans ce tutoriel, nous incluons deux méthodes d'authentification : l'authentification Microsoft Entra et l'authentification MySQL. L'onglet **Sans mot de passe** affiche l'authentification Microsoft Entra et l'onglet **Mot de passe** l'authentification MySQL.

L'authentification Microsoft Entra est un mécanisme de connexion à Azure Database pour MySQL à l'aide des identités définies dans Microsoft Entra ID. Avec l'authentification Microsoft Entra, vous pouvez gérer les identités des utilisateurs de base de données et d'autres services Microsoft dans un emplacement centralisé, ce qui simplifie la gestion des autorisations.

L'authentification MySQL utilise des comptes stockés dans MySQL. Si vous choisissez d'utiliser des mots de passe comme informations d'identification pour les comptes, ces informations d'identification sont stockées dans la table `user`. Étant donné que ces mots de passe sont stockés dans MySQL, vous devez gérer la rotation des mots de passe par vous-même.

Prérequis

- Un abonnement Azure - [En créer un gratuitement](#)
- [Kit de développement Java \(JDK\)](#), version 8 ou ultérieure.
- [Apache Maven](#).
- [Azure CLI](#).
- [Client de ligne de](#) commande MySQL.
- Si vous n'avez pas d'application Spring Boot, créez un projet Maven avec [Spring Initializr](#). Veillez à sélectionner **Maven Project** et, sous **Dépendances**, ajoutez les dépendances **Spring Web**, **Spring Data JDBC** et **MySQL Driver**, puis sélectionnez Java version 8 ou ultérieure.

- Si vous n'en avez pas, créez une instance de serveur flexible Azure Database pour MySQL nommée `mysqlflexibletest`. Pour obtenir des instructions, consultez [Démarrage rapide : Utilisez le Portail Azure pour créer un serveur flexible Azure Database pour MySQL](#). Ensuite, créez une base de données nommée `demo`. Pour obtenir des instructions, consultez [Créer et gérer des bases de données pour Azure Database pour MySQL serveur flexible](#).

Voir l'exemple d'application

Dans ce tutoriel, vous allez coder un exemple d'application. Si vous souhaitez aller plus vite, cette application est déjà codée et disponible sur <https://github.com/Azure-Samples/quickstart-spring-data-jdbc-mysql>.

Configurer une règle de pare-feu pour votre serveur MySQL

Les instances Azure Database pour MySQL sont sécurisées par défaut. Elles ont un pare-feu qui n'autorise aucune connexion entrante.

Pour pouvoir utiliser votre base de données, ouvrez le pare-feu du serveur pour autoriser l'adresse IP locale à accéder au serveur de base de données. Pour plus d'informations, consultez [Gérer les règles de pare-feu pour Azure Database pour MySQL - Serveur flexible à l'aide du Portail Azure](#).

Si vous vous connectez à votre serveur MySQL à partir de Sous-système Windows pour Linux (WSL) sur un ordinateur Windows, vous devez ajouter l'adresse IP de l'hôte WSL à votre pare-feu.

Créer un utilisateur non-administrateur MySQL et accorder des autorisations

Cette étape crée un utilisateur non administrateur et lui accorde toutes les autorisations sur la `demo` base de données.

Sans mot de passe (recommandé)

You can use the following method to create a non-administrative user who uses a passwordless connection.

Service Connector (recommandé)

1. Utilisez la commande suivante pour installer l'extension [sans mot de passe du service Connecter or](#) pour Azure CLI :

Azure CLI

```
az extension add --name serviceconnector-passwordless --upgrade
```

2. Utilisez la commande suivante pour créer l'utilisateur non administrateur Microsoft Entra :

Azure CLI

```
az connection create mysql-flexible \
--resource-group <your_resource_group_name> \
--connection mysql_conn \
--target-resource-group <your_resource_group_name> \
--server mysqlflexibletest \
--database demo \
--user-account mysql-identity-
id=/subscriptions/<your_subscription_id>/resourcegroups/<your_r
esource_group_name>/providers/Microsoft.ManagedIdentity/userAss
ignedIdentities/<your_user_assigned_managed_identity_name> \
--query authInfo.userName \
--output tsv
```

Une fois la commande terminée, notez le nom d'utilisateur dans la sortie de la console.

Stocker des données à partir de Azure Database pour MySQL

Maintenant que vous disposez d'une instance de serveur flexible Azure Database pour MySQL, vous pouvez stocker des données à l'aide de Spring Cloud Azure.

Pour installer le module Spring Cloud Azure Starter JDBC MySQL, ajoutez les dépendances suivantes à votre *fichier pom.xml* :

- The Spring Cloud Azure Bill of Materials (BOM) :

XML

```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>com.azure.spring</groupId>
      <artifactId>spring-cloud-azure-dependencies</artifactId>
      <version>4.13.0</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>
```

ⓘ Notes

Si vous utilisez Spring Boot 3.x, veillez à définir la `spring-cloud-azure-dependencies` version `5.7.0` sur . Pour plus d'informations sur la `spring-cloud-azure-dependencies` version, consultez [La version de Spring Cloud Azure à utiliser ↗](#).

- Artefact JDBC MySQL Spring Cloud Azure Starter :

XML

```
<dependency>
  <groupId>com.azure.spring</groupId>
  <artifactId>spring-cloud-azure-starter-jdbc-mysql</artifactId>
</dependency>
```

ⓘ Notes

Les connexions sans mot de passe ont été prises en charge depuis la version `4.5.0`.

Configurer Spring Boot pour qu'il utilise Azure Database pour MySQL

Pour stocker des données à partir de Azure Database pour MySQL à l'aide de Spring Data JDBC, procédez comme suit pour configurer l'application :

1. Configurez Azure Database pour MySQL informations d'identification en ajoutant les propriétés suivantes à votre *fichier de configuration application.properties*.

Sans mot de passe (recommandé)

properties

```
logging.level.org.springframework.jdbc.core=DEBUG  
  
spring.datasource.url=jdbc:mysql://mysqlflexibletest.mysql.database.azure.com:3306/demo?serverTimezone=UTC  
spring.datasource.username=<your_mysql_ad_non_admin_username>  
spring.datasource.azure.passwordless-enabled=true  
  
spring.sql.init.mode=always
```

⚠ Avertissement

La propriété `spring.sql.init.mode=always` de configuration signifie que Spring Boot génère automatiquement un schéma de base de données à l'aide du *fichier schema.sql* que vous allez créer ensuite, chaque fois que le serveur est démarré. Cette fonctionnalité est idéale pour les tests, mais n'oubliez pas qu'elle supprime vos données à chaque redémarrage. Vous ne devez donc pas l'utiliser en production.

`?serverTimezone=UTC` a été ajouté à la propriété de configuration `spring.datasource.url` pour indiquer au pilote JDBC d'utiliser le format de date UTC (Coordinated Universal Time) lors de la connexion à la base de données. Sans ce paramètre, votre serveur Java n'utilise pas le même format de date que la base de données, ce qui entraînerait une erreur.

- Créez le *fichier de configuration* `src/main/resources/schema.sql` pour configurer le schéma de base de données, puis ajoutez le contenu suivant.

SQL

```
DROP TABLE IF EXISTS todo;  
CREATE TABLE todo (id SERIAL PRIMARY KEY, description VARCHAR(255),  
details VARCHAR(4096), done BOOLEAN);
```

- Créez une `Todo` classe Java. Cette classe est un modèle de domaine mappé sur la `todo` table qui sera créée automatiquement par Spring Boot. Le code suivant ignore les méthodes et `setters` les `getters` méthodes.

Java

```

import org.springframework.data.annotation.Id;

public class Todo {

    public Todo() {
    }

    public Todo(String description, String details, boolean done) {
        this.description = description;
        this.details = details;
        this.done = done;
    }

    @Id
    private Long id;

    private String description;

    private String details;

    private boolean done;
}

```

4. Modifiez le fichier de classe de démarrage pour afficher le contenu suivant.

Java

```

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.boot.context.event.ApplicationReadyEvent;
import org.springframework.context.ApplicationListener;
import org.springframework.context.annotation.Bean;
import org.springframework.data.repository.CrudRepository;

import java.util.stream.Stream;

@SpringBootApplication
public class DemoApplication {

    public static void main(String[] args) {
        SpringApplication.run(DemoApplication.class, args);
    }

    @Bean
    ApplicationListener<ApplicationReadyEvent>
    basicsApplicationListener(TodoRepository repository) {
        return event->repository
            .saveAll(Stream.of("A", "B", "C").map(name->new
Todo("configuration", "congratulations, you have set up correctly!",
true)).toList())
            .forEach(System.out::println);
    }
}

```

```
    }

}

interface TodoRepository extends CrudRepository<Todo, Long> {
```

Conseil

Dans ce tutoriel, il n'existe aucune opération d'authentification dans les configurations ou le code. Toutefois, la connexion aux services Azure nécessite une authentification. Pour effectuer l'authentification, vous devez utiliser Identité Azure. Spring Cloud Azure utilise `DefaultAzureCredential`, que la bibliothèque d'identités Azure fournit pour vous aider à obtenir des informations d'identification sans aucune modification du code.

`DefaultAzureCredential` prend en charge plusieurs méthodes d'authentification et détermine quelle méthode doit être utilisée au moment de l'exécution. Cette approche permet à votre application d'utiliser différentes méthodes d'authentification dans différents environnements (tels que les environnements locaux et de production) sans implémenter de code spécifique à l'environnement. Pour plus d'informations, consultez [DefaultAzureCredential](#).

Pour terminer l'authentification dans les environnements de développement locaux, vous pouvez utiliser Azure CLI, Visual Studio Code, PowerShell ou d'autres méthodes. Pour plus d'informations, consultez [l'authentification Azure dans les environnements](#) de développement Java. Pour terminer l'authentification dans les environnements d'hébergement Azure, nous vous recommandons d'utiliser l'identité managée affectée par l'utilisateur. Pour plus d'informations, consultez [Que sont les identités managées pour les ressources Azure ?](#)

5. Lancez l'application. L'application stocke les données dans la base de données. Vous verrez des journaux similaires à l'exemple suivant :

shell

```
2023-02-01 10:22:36.701 DEBUG 7948 --- [main]
o.s.jdbc.core.JdbcTemplate : Executing prepared SQL statement [INSERT
INTO todo (description, details, done) VALUES (?, ?, ?)]
com.example.demo.Todo@4bdb04c8
```

Déployer sur Azure Spring Apps

Maintenant que vous disposez de l'application Spring Boot en cours d'exécution localement, il est temps de le déplacer en production. [Azure Spring Apps](#) facilite le déploiement d'applications Spring Boot sur Azure sans aucune modification de code. Le service gère l'infrastructure des applications Spring, ce qui permet aux développeurs de se concentrer sur leur code. Azure Spring Apps assure la gestion du cycle de vie en utilisant des outils complets, tels que la supervision et les diagnostics, la gestion des configurations, la découverte de services, l'intégration CI/CD, les déploiements bleus-verts, etc. Pour déployer votre application sur Azure Spring Apps, consultez [Déployer votre première application sur Azure Spring Apps](#).

Étapes suivantes

[Exemples Azure pour les développeurs](#)

[Spring Cloud Azure MySQL](#)

Utiliser Spring Data JDBC avec Azure Database pour PostgreSQL

Article • 23/10/2023

Ce tutoriel montre comment stocker des données dans une [base de données Azure Database pour PostgreSQL](#) à l'aide de [Spring Data JDBC](#).

[JDBC](#) est l'API Java standard pour se connecter à des bases de données relationnelles traditionnelles.

Dans ce tutoriel, nous incluons deux méthodes d'authentification : l'authentification Microsoft Entra et l'authentification PostgreSQL. L'onglet **Sans mot de passe** affiche l'authentification Microsoft Entra et l'onglet **Mot de passe** affiche l'authentification PostgreSQL.

L'authentification Microsoft Entra est un mécanisme de connexion à Azure Database pour PostgreSQL utilisant les identités définies dans Microsoft Entra ID. Avec l'authentification Microsoft Entra, vous pouvez gérer les identités des utilisateurs de base de données et d'autres services Microsoft dans un emplacement centralisé, ce qui simplifie la gestion des autorisations.

L'authentification PostgreSQL utilise des comptes stockés dans PostgreSQL. Si vous choisissez d'utiliser des mots de passe comme informations d'identification pour les comptes, ces informations d'identification sont stockées dans la table `user`. Étant donné que ces mots de passe sont stockés dans PostgreSQL, vous devez gérer la rotation des mots de passe par vous-même.

Prérequis

- Un abonnement Azure - [En créer un gratuitement](#)
- Kit de développement Java (JDK), version 8 ou ultérieure.
- [Apache Maven](#).
- [Azure CLI](#).
- [Client](#) de ligne de commande PostgreSQL.
- Si vous n'avez pas d'application Spring Boot, créez un projet Maven avec [Spring Initializr](#). Veillez à sélectionner **Maven Project** et, sous **Dépendances**, ajoutez les

dépendances Spring Web, Spring Data JDBC et PostgreSQL Driver, puis sélectionnez Java version 8 ou ultérieure.

- Si vous n'en avez pas, créez une instance de serveur flexible Azure Database pour PostgreSQL nommée `postgresqlflexibletest` et une base de données nommée `demo`. Pour obtenir des instructions, consultez [Démarrage rapide : Créer un serveur flexible Azure Database pour PostgreSQL dans le Portail Azure](#).

Voir l'exemple d'application

Dans ce tutoriel, vous allez coder un exemple d'application. Si vous souhaitez aller plus vite, cette application est déjà codée et disponible sur <https://github.com/Azure-Samples/quickstart-spring-data-jdbc-postgresql>.

Configurer une règle de pare-feu pour votre serveur PostgreSQL

Les instances Azure Database pour PostgreSQL sont sécurisées par défaut. Elles ont un pare-feu qui n'autorise aucune connexion entrante.

Pour pouvoir utiliser votre base de données, ouvrez le pare-feu du serveur pour autoriser l'adresse IP locale à accéder au serveur de base de données. Pour plus d'informations, consultez [les règles de pare-feu dans Azure Database pour PostgreSQL - Serveur flexible](#).

Si vous vous connectez à votre serveur PostgreSQL à partir de Sous-système Windows pour Linux (WSL) sur un ordinateur Windows, vous devez ajouter l'ID d'hôte WSL à votre pare-feu.

Créer un utilisateur non administrateur PostgreSQL et accorder des autorisations

Ensuite, créez un utilisateur non-administrateur et accordez toutes les autorisations à la base de données.

Sans mot de passe (recommandé)

Vous pouvez utiliser la méthode suivante pour créer un utilisateur non administrateur qui utilise une connexion sans mot de passe.

Service Connector (recommandé)

1. Utilisez la commande suivante pour installer l'extension [sans mot de passe du service Connecter or](#) pour Azure CLI :

Azure CLI

```
az extension add --name serviceconnector-passwordless --upgrade
```

2. Utilisez la commande suivante pour créer l'utilisateur non administrateur Microsoft Entra :

Azure CLI

```
az connection create postgres-flexible \
--resource-group <your_resource_group_name> \
--connection postgres_conn \
--target-resource-group <your_resource_group_name> \
--server postgresqlflexibletest \
--database demo \
--user-account \
--query authInfo.userName \
--output tsv
```

Une fois la commande terminée, notez le nom d'utilisateur dans la sortie de la console.

Stocker des données à partir de Azure Database pour PostgreSQL

Maintenant que vous disposez d'une instance de serveur flexible Azure Database pour PostgreSQL, vous pouvez stocker des données à l'aide de Spring Cloud Azure.

Pour installer le module PostgreSQL JDBC Jdbc Spring Cloud Azure Starter, ajoutez les dépendances suivantes à votre *fichier pom.xml* :

- The Spring Cloud Azure Bill of Materials (BOM) :

XML

```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>com.azure.spring</groupId>
      <artifactId>spring-cloud-azure-dependencies</artifactId>
      <version>4.13.0</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>
```

ⓘ Notes

Si vous utilisez Spring Boot 3.x, veillez à définir la `spring-cloud-azure-dependencies` version `5.7.0` sur . Pour plus d'informations sur la `spring-cloud-azure-dependencies` version, consultez [La version de Spring Cloud Azure à utiliser ↗](#).

- L'artefact Jdbc PostgreSQL JDBC Spring Cloud Azure Starter :

XML

```
<dependency>
  <groupId>com.azure.spring</groupId>
  <artifactId>spring-cloud-azure-starter-jdbc-postgresql</artifactId>
</dependency>
```

ⓘ Notes

Les connexions sans mot de passe ont été prises en charge depuis la version `4.5.0`.

Configurer Spring Boot pour qu'il utilise Azure Database pour PostgreSQL

Pour stocker des données à partir de Azure Database pour PostgreSQL à l'aide de Spring Data JDBC, procédez comme suit pour configurer l'application :

1. Configurez Azure Database pour PostgreSQL informations d'identification en ajoutant les propriétés suivantes à votre *fichier de configuration application.properties*.

Sans mot de passe (recommandé)

properties

```
logging.level.org.springframework.jdbc.core=DEBUG  
  
spring.datasource.url=jdbc:postgresql://postgresqlflexibletest.post  
gres.database.azure.com:5432/demo?sslmode=require  
spring.datasource.username=<your_postgresql_ad_non_admin_username>  
spring.datasource.azure.passwordless-enabled=true  
  
spring.sql.init.mode=always
```

⚠ Avertissement

La propriété `spring.sql.init.mode=always` de configuration signifie que Spring Boot génère automatiquement un schéma de base de données à l'aide du *fichier schema.sql* que vous allez créer ensuite, chaque fois que le serveur est démarré. Cette fonctionnalité est idéale pour les tests, mais n'oubliez pas qu'elle supprime vos données à chaque redémarrage. Vous ne devez donc pas l'utiliser en production.

- Créez le *fichier de configuration* `src/main/resources/schema.sql` pour configurer le schéma de base de données, puis ajoutez le contenu suivant.

SQL

```
DROP TABLE IF EXISTS todo;  
CREATE TABLE todo (id SERIAL PRIMARY KEY, description VARCHAR(255),  
details VARCHAR(4096), done BOOLEAN);
```

- Créez une `Todo` classe Java. Cette classe est un modèle de domaine mappé sur la `todo` table qui sera créée automatiquement par Spring Boot. Le code suivant ignore les méthodes et `setters` les `getters` méthodes.

Java

```
import org.springframework.data.annotation.Id;  
  
public class Todo {  
  
    public Todo() {  
    }  
}
```

```

    public Todo(String description, String details, boolean done) {
        this.description = description;
        this.details = details;
        this.done = done;
    }

    @Id
    private Long id;

    private String description;

    private String details;

    private boolean done;
}

```

4. Modifiez le fichier de classe de démarrage pour afficher le contenu suivant.

Java

```

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.boot.context.event.ApplicationReadyEvent;
import org.springframework.context.ApplicationListener;
import org.springframework.context.annotation.Bean;
import org.springframework.data.repository.CrudRepository;

import java.util.stream.Stream;

@SpringBootApplication
public class DemoApplication {

    public static void main(String[] args) {
        SpringApplication.run(DemoApplication.class, args);
    }

    @Bean
    ApplicationListener<ApplicationReadyEvent>
    basicsApplicationListener(TodoRepository repository) {
        return event->repository
            .saveAll(Stream.of("A", "B", "C").map(name->new
Todo("configuration", "congratulations, you have set up correctly!",
true)).toList())
            .forEach(System.out::println);
    }

}

interface TodoRepository extends CrudRepository<Todo, Long> {
}

```

Conseil

Dans ce tutoriel, il n'existe aucune opération d'authentification dans les configurations ou le code. Toutefois, la connexion aux services Azure nécessite une authentification. Pour effectuer l'authentification, vous devez utiliser Identité Azure. Spring Cloud Azure utilise `DefaultAzureCredential`, que la bibliothèque d'identités Azure fournit pour vous aider à obtenir des informations d'identification sans aucune modification du code.

`DefaultAzureCredential` prend en charge plusieurs méthodes d'authentification et détermine quelle méthode doit être utilisée au moment de l'exécution. Cette approche permet à votre application d'utiliser différentes méthodes d'authentification dans différents environnements (tels que les environnements locaux et de production) sans implémenter de code spécifique à l'environnement. Pour plus d'informations, consultez [DefaultAzureCredential](#).

Pour terminer l'authentification dans les environnements de développement locaux, vous pouvez utiliser Azure CLI, Visual Studio Code, PowerShell ou d'autres méthodes. Pour plus d'informations, consultez [l'authentification Azure dans les environnements](#) de développement Java. Pour terminer l'authentification dans les environnements d'hébergement Azure, nous vous recommandons d'utiliser l'identité managée affectée par l'utilisateur. Pour plus d'informations, consultez [Que sont les identités managées pour les ressources Azure ?](#)

5. Lancez l'application. L'application stocke les données dans la base de données. Vous verrez des journaux similaires à l'exemple suivant :

shell

```
2023-02-01 10:22:36.701 DEBUG 7948 --- [main]
o.s.jdbc.core.JdbcTemplate : Executing prepared SQL statement [INSERT
INTO todo (description, details, done) VALUES (?, ?, ?)]
com.example.demo.Todo@4bdb04c8
```

Déployer sur Azure Spring Apps

Maintenant que vous disposez de l'application Spring Boot en cours d'exécution localement, il est temps de le déplacer en production. [Azure Spring Apps](#) facilite le déploiement d'applications Spring Boot sur Azure sans aucune modification de code. Le

service gère l'infrastructure des applications Spring, ce qui permet aux développeurs de se concentrer sur leur code. Azure Spring Apps assure la gestion du cycle de vie en utilisant des outils complets, tels que la supervision et les diagnostics, la gestion des configurations, la découverte de services, l'intégration CI/CD, les déploiements bleus-verts, etc. Pour déployer votre application sur Azure Spring Apps, consultez [Déployer votre première application sur Azure Spring Apps](#).

Étapes suivantes

[Exemples Azure pour les développeurs](#)

[Spring Cloud Azure PostgreSQL](#)

Configurer des connexions aux bases de données sans mot de passe pour les applications Java sur des serveurs Oracle WebLogic

Article • 20/10/2023

Cet article explique comment configurer des connexions de base de données sans mot de passe pour les applications Java sur Oracle WebLogic Server avec le Portail Azure.

Dans ce guide, vous effectuez les tâches suivantes :

- ✓ Provisionnez des ressources de base de données à l'aide d'Azure CLI.
- ✓ Activez l'administrateur Microsoft Entra dans la base de données.
- ✓ Provisionnez une identité managée affectée par l'utilisateur et créez un utilisateur de base de données pour celui-ci.
- ✓ Configurez une connexion de base de données sans mot de passe dans les offres Oracle WebLogic avec la Portail Azure.
- ✓ Validez la connexion de base de données.

Les offres prennent en charge les connexions sans mot de passe pour les bases de données PostgreSQL, MySQL et Azure SQL.

Prérequis

- Si vous n'avez pas [d'abonnement Azure](#), créez un [compte gratuit](#) avant de commencer.
- Utilisez [Azure Cloud Shell](#) à l'aide de l'environnement Bash ; vérifiez que la version d'Azure CLI est 2.43.0 ou ultérieure.

 [Launch Cloud Shell](#)



- Si vous préférez, [installez Azure CLI 2.43.0 ou version ultérieure](#) pour exécuter des commandes Azure CLI.
 - Si vous utilisez une installation locale, connectez-vous à Azure CLI à l'aide de la commande [az login](#). Pour finir le processus d'authentification, suivez les étapes affichées dans votre terminal. Consultez [Se connecter avec Azure CLI](#) pour obtenir d'autres options de connexion.

- Lorsque vous y êtes invité, installez les extensions Azure CLI lors de la première utilisation. Pour plus d'informations sur les extensions, consultez [Utiliser des extensions avec Azure CLI](#).
- Exécutez `az version` pour rechercher la version et les bibliothèques dépendantes installées. Pour effectuer une mise à niveau vers la dernière version, exécutez `az upgrade`.
- Vérifiez que l'identité Azure que vous utilisez pour vous connecter et compléter cet article a le rôle [Propriétaire](#) dans l'abonnement actuel ou les rôles [Contributeur](#) et [Administrateur de l'accès utilisateur](#) dans l'abonnement actuel. Pour obtenir une vue d'ensemble des rôles Azure, consultez [Qu'est-ce que le contrôle d'accès en fonction du rôle Azure \(Azure RBAC\) ?](#) Pour plus d'informations sur les rôles spécifiques requis par l'offre de la Place de marché Oracle WebLogic, consultez [les rôles intégrés Azure](#).

Créer un groupe de ressources

Créez un groupe de ressources avec la commande `az group create`. Étant donné que les groupes de ressources doivent être uniques au sein d'un abonnement, choisissez un nom unique. Un moyen simple d'avoir des noms uniques consiste à utiliser une combinaison de vos initiales, de la date d'aujourd'hui et de certains identificateurs. Par exemple, `abc1228rg`. Cet exemple crée un groupe de ressources nommé `abc1228rg` à l'emplacement `eastus` :

Azure CLI

```
export RESOURCE_GROUP_NAME="abc1228rg"
az group create \
--name ${RESOURCE_GROUP_NAME} \
--location eastus
```

Créer un serveur de base de données et une base de données

Serveur flexible MySQL

Créez un serveur flexible avec la commande `az mysql flexible-server create`. Cet exemple crée un serveur flexible nommé `mysql120221201` avec un utilisateur administrateur et un mot de passe `Secret123456` d'utilisateur `azureuser`.

Remplacez le mot de passe par le vôtre. Pour plus d'informations, consultez [Créer un serveur flexible Azure Database pour MySQL à l'aide d'Azure CLI](#).

Azure CLI

```
export MYSQL_NAME="mysql120221201"
export MYSQL_ADMIN_USER="azureuser"
export MYSQL_ADMIN_PASSWORD="Secret123456"

az mysql flexible-server create \
    --resource-group $RESOURCE_GROUP_NAME \
    --name $MYSQL_NAME \
    --location eastus \
    --admin-user $MYSQL_ADMIN_USER \
    --admin-password $MYSQL_ADMIN_PASSWORD \
    --public-access 0.0.0.0 \
    --tier Burstable \
    --sku-name Standard_B1ms
```

Créez une base de données avec [az mysql flexible-server db create](#).

Azure CLI

```
export DATABASE_NAME="contoso"

# create mysql database
az mysql flexible-server db create \
    --resource-group $RESOURCE_GROUP_NAME \
    --server-name $MYSQL_NAME \
    --database-name $DATABASE_NAME
```

Une fois la commande terminée, la sortie doit ressembler à l'exemple suivant :

Sortie

```
Creating database with utf8 charset and utf8_general_ci collation
{
  "charset": "utf8",
  "collation": "utf8_general_ci",
  "id": "/subscriptions/contoso-
hashcode/resourceGroups/abc1228rg/providers/Microsoft.DBforMySQL/flexibl
eServers/mysql120221201/databases/contoso",
  "name": "contoso",
  "resourceGroup": "abc1228rg",
  "systemData": null,
  "type": "Microsoft.DBforMySQL/flexibleServers/databases"
}
```

Configurer un administrateur Microsoft Entra sur votre base de données

Maintenant que vous avez créé la base de données, vous devez la préparer pour prendre en charge les connexions sans mot de passe. Une connexion sans mot de passe nécessite une combinaison d'identités managées pour les ressources Azure et l'authentification Microsoft Entra. Pour obtenir une vue d'ensemble des identités managées pour les ressources Azure, consultez [Qu'est-ce que les identités managées pour les ressources Azure ?](#)

Serveur flexible MySQL

Pour plus d'informations sur l'interaction du serveur flexible MySQL avec les identités managées, consultez [Utiliser l'ID Microsoft Entra pour l'authentification avec MySQL](#).

L'exemple suivant configure l'utilisateur Azure CLI actuel en tant que compte d'administrateur Microsoft Entra. Pour activer l'authentification Azure, il est nécessaire d'attribuer une identité au serveur flexible MySQL.

Tout d'abord, créez une identité managée avec `az identity create` et affectez l'identité au serveur MySQL avec `az mysql flexible-server identity assign`.

Azure CLI

```
export MYSQL_UMI_NAME="id-mysql-aad-20221205"

# create a User Assigned Managed Identity for MySQL to be used for AAD
# authentication
az identity create \
    --resource-group $RESOURCE_GROUP_NAME \
    --name $MYSQL_UMI_NAME

## assign the identity to the MySQL server
az mysql flexible-server identity assign \
    --resource-group $RESOURCE_GROUP_NAME \
    --server-name $MYSQL_NAME \
    --identity $MYSQL_UMI_NAME
```

Ensuite, définissez l'utilisateur Azure CLI actuel en tant que compte d'administrateur Microsoft Entra avec `az mysql flexible-server ad-admin create`.

Azure CLI

```
export CURRENT_USER=$(az account show --query user.name --output tsv)
export CURRENT_USER_OBJECTID=$(az ad signed-in-user show --query id --output tsv)

az mysql flexible-server ad-admin create \
    --resource-group $RESOURCE_GROUP_NAME \
    --server-name $MYSQL_NAME \
    --object-id $CURRENT_USER_OBJECTID \
    --display-name $CURRENT_USER \
    --identity $MYSQL_UMI_NAME
```

Créer une identité managée attribuée par l'utilisateur

Ensute, dans Azure CLI, créez une identité dans votre abonnement à l'aide de la [commande az identity create](#). Vous utilisez cette identité managée pour vous connecter à votre base de données.

Azure CLI

```
az identity create \
    --resource-group ${RESOURCE_GROUP_NAME} \
    --name myManagedIdentity
```

Pour configurer l'identité dans les étapes suivantes, utilisez la [commande az identity show](#) pour stocker l'ID client de l'identité dans une variable shell.

Azure CLI

```
# Get client ID of the user-assigned identity
export CLIENT_ID=$(az identity show \
    --resource-group ${RESOURCE_GROUP_NAME} \
    --name myManagedIdentity \
    --query clientId \
    --output tsv)
```

Créer un utilisateur de base de données pour votre identité managée

À présent, connectez-vous en tant qu'utilisateur administrateur Microsoft Entra à votre base de données MySQL et créez un utilisateur MySQL pour votre identité managée.

Tout d'abord, vous devez créer une règle de pare-feu pour accéder au serveur MySQL à partir de votre client CLI. Exécutez les commandes suivantes pour obtenir votre adresse IP actuelle.

Bash

```
export MY_IP=$(curl http://whatismyip.akamai.com)
```

Si vous travaillez sur Sous-système Windows pour Linux (WSL) avec VPN activé, la commande suivante peut renvoyer une adresse IPv4 incorrecte. Une façon d'obtenir votre adresse IPv4 consiste à visiter whatismyipaddress.com. Dans tous les cas, définissez la variable `MY_IP` d'environnement comme adresse IPv4 à partir de laquelle vous souhaitez vous connecter à la base de données.

Créez une règle de pare-feu temporaire avec [az mysql flexible-server firewall-rule create](#).

Azure CLI

```
az mysql flexible-server firewall-rule create \
    --resource-group $RESOURCE_GROUP_NAME \
    --name $MYSQL_NAME \
    --rule-name AllowCurrentMachineToConnect \
    --start-ip-address ${MY_IP} \
    --end-ip-address ${MY_IP}
```

Ensuite, préparez un *fichier.sql* pour créer un utilisateur de base de données pour l'identité managée. L'exemple suivant ajoute un utilisateur avec un nom `identity-contoso` de connexion et accorde les priviléges utilisateur pour accéder à la base de données `contoso`.

Bash

```
export IDENTITY_LOGIN_NAME="identity-contoso"

cat <<EOF >createuser.sql
SET aad_auth_validate_oids_in_tenant = OFF;
DROP USER IF EXISTS '${IDENTITY_LOGIN_NAME}'@'%';
CREATE AADUSER '${IDENTITY_LOGIN_NAME}' IDENTIFIED BY '${CLIENT_ID}';
GRANT ALL PRIVILEGES ON ${DATABASE_NAME}.* TO
'${IDENTITY_LOGIN_NAME}'@'%';
```

```
FLUSH privileges;  
EOF
```

Exécutez le *fichier .sql* avec la commande [az mysql flexible-server execute](#). Vous pouvez obtenir votre jeton d'accès avec la commande [az account get-access-token](#).

Azure CLI

```
export RDBMS_ACCESS_TOKEN=$(az account get-access-token \  
    --resource-type oss-rdbms \  
    --query accessToken \  
    --output tsv)  
  
az mysql flexible-server execute \  
    --name ${MYSQL_NAME} \  
    --admin-user ${CURRENT_USER} \  
    --admin-password ${RDBMS_ACCESS_TOKEN} \  
    --file-path "createuser.sql"
```

Vous pouvez être invité à installer l'extension `rdbms-connect`, comme indiqué dans la sortie suivante. Appuyez `y` pour continuer. Si vous ne travaillez pas avec l'utilisateur `root`, vous devez entrer le mot de passe de l'utilisateur.

Sortie

```
The command requires the extension rdbms-connect. Do you want to install  
it now? The command will continue to run after the extension is  
installed. (Y/n): y  
Run 'az config set extension.use_dynamic_install=yes_without_prompt' to  
allow installing extensions without prompt.  
This extension depends on gcc, libpq-dev, python3-dev and they will be  
installed first.  
[sudo] password for user:
```

Si le *fichier .sql* s'exécute correctement, vous trouvez une sortie similaire à l'exemple suivant :

Sortie

```
Running *.sql* file 'createuser.sql'...  
Successfully executed the file.  
Closed the connection to mysql20221201
```

L'identité `myManagedIdentity` managée a désormais accès à la base de données lors de l'authentification avec le nom d'utilisateur `identity-contoso`.

Si vous ne souhaitez plus accéder au serveur à partir de cette adresse IP, vous pouvez supprimer la règle de pare-feu à l'aide de la commande suivante.

Azure CLI

```
az mysql flexible-server firewall-rule delete \
--resource-group $RESOURCE_GROUP_NAME \
--name $MYSQL_NAME \
--rule-name AllowCurrentMachineToConnect \
--yes
```

Enfin, utilisez la commande suivante pour obtenir la chaîne de connexion que vous utilisez dans la section suivante.

Azure CLI

```
export
CONNECTION_STRING="jdbc:mysql://${MYSQL_NAME}.mysql.database.azure.com:3
306/${DATABASE_NAME}?useSSL=true"
echo ${CONNECTION_STRING}
```

Configurer une connexion de base de données sans mot de passe pour Oracle WebLogic Server sur des machines virtuelles Azure

Cette section vous montre comment configurer la connexion de source de données sans mot de passe à l'aide des offres Place de marché Azure pour Oracle WebLogic Server.

Tout d'abord, commencez le processus de déploiement d'une offre. Les offres suivantes prennent en charge les connexions de base de données sans mot de passe :

- [Oracle WebLogic Server sur Azure Kubernetes Service ↗](#)
 - [Démarrage rapide](#)
- [Cluster Oracle WebLogic Server sur des machines virtuelles ↗](#)
 - [Démarrage rapide](#)
- [Oracle WebLogic Server avec Administration Server sur des machines virtuelles ↗](#)
 - [Démarrage rapide](#)
- [Cluster dynamique Oracle WebLogic Server sur des machines virtuelles ↗](#)
 - [Démarrage rapide](#)

Renseignez les informations requises dans le **volet Informations de base et d'autres volets** si vous souhaitez activer les fonctionnalités. Lorsque vous atteignez le **volet Base**

de données , renseignez la configuration sans mot de passe, comme indiqué dans les étapes suivantes.

Serveur flexible MySQL

1. Pour **Connecter à la base de données ?**, sélectionnez Oui.
2. Sous **les paramètres d'Connection**, pour **Choisir le type de base de données**, ouvrez le menu déroulant, puis sélectionnez **MySQL (avec prise en charge de la connexion sans mot de passe)** .
3. Pour **le nom JNDI**, entrez *testpasswordless* ou votre valeur attendue.
4. Pour **DataSource Connection String**, entrez la chaîne de connexion obtenue dans la dernière section.
5. Pour **le nom d'utilisateur** de la base de données, entrez le nom d'utilisateur de la base de données de votre identité managée (la valeur de `${IDENTITY_LOGIN_NAME}`). Dans cet exemple, la valeur est `identity-contoso`.
6. Sélectionnez **Utiliser la connexion** de source de données sans mot de passe.
7. Pour **l'identité** managée affectée par l'utilisateur, sélectionnez l'identité managée que vous avez créée précédemment. Dans cet exemple, son nom est `myManagedIdentity`.

La **section des paramètres** de Connecter ion doit ressembler à la capture d'écran suivante, qui utilise [oracle WebLogic Server Cluster sur des machines virtuelles](#) comme exemple.

Connection settings

Choose database type * ⓘ MySQL (with support for passwordless connection) ▼

Information: To support passwordless connection and various functionalities, the offer will upgrade the [Oracle WebLogic Server MySQL driver](#) with recent [MySQL Connector Java driver](#).

JNDI Name * ⓘ	testpasswordless ✓
DataSource Connection String * ⓘ	jdbc:mysql://mysql20221201.mysql.database.azure.com:3306/checklist... ✓
Global transactions protocol * ⓘ	OnePhaseCommit ▼
Database username * ⓘ	identity-contoso ✓
Use passwordless datasource connection ⓘ	<input checked="" type="checkbox"/> ✓

User assigned managed identity

Select a user assigned identity that has access to your database. For how to create a database user for your managed identity, see <https://aka.ms/javaee-db-identity>.

+ Add - Remove

Name	↑↓ resource group	↑↓ subscription
<input type="checkbox"/> myManagedIdentity	mydbrg20221201	+ 🔍

Vous avez maintenant terminé de configurer la connexion sans mot de passe. Vous pouvez continuer à remplir les volets suivants ou sélectionner **Vérifier + créer**, puis créer pour déployer l'offre.

Vérifier la connexion de base de données

La connexion de base de données est configurée correctement si le déploiement de l'offre se termine sans erreur.

Pour continuer à prendre [le cluster Oracle WebLogic Server sur des machines virtuelles](#) comme exemple, une fois le déploiement terminé, suivez ces étapes dans le Portail Azure pour rechercher l'URL de la console Administration.

1. Recherchez le groupe de ressources dans lequel vous avez déployé WLS.
2. Sous **Paramètres**, sélectionnez **Déploiements**.
3. Sélectionnez le déploiement avec la durée **la plus longue**. Ce déploiement doit se trouver en bas de la liste.
4. Sélectionnez **Sorties**.

5. L'URL de la console WebLogic Administration istration est la valeur de la **soutie adminConsoleUrl**.
6. Copiez la valeur de la variable `adminConsoleUrl` de soutie.
7. Collez la valeur dans la barre d'adresses de votre navigateur et appuyez sur **Entrée** pour ouvrir la page de connexion de la console WebLogic Administration istration.

Procédez comme suit pour vérifier la connexion de base de données.

1. Connectez-vous à la console WebLogic Administration istration avec le nom d'utilisateur et le mot de passe que vous avez fournis dans le **volet Informations de base**.
2. Sous la **structure** de domaine, sélectionnez **Services**, **Sources de données**, puis **testpasswordless**.
3. Sélectionnez l'onglet **Surveillance**, où l'état de la source de données est *en cours d'exécution*, comme illustré dans la capture d'écran suivante.

The screenshot shows the 'Flexible MySQL Server' configuration page. On the left, the 'Domain Structure' tree shows 'wlsd' with 'Data Sources' selected. The main panel displays 'Settings for testpasswordless' with tabs for Configuration, Targets, Monitoring (which is highlighted with a red box), Control, Security, and Notes. Under the Monitoring tab, it says 'This page displays statistics associated with this JDBC data source. Use this page to monitor the activity of the data source.' Below this is a section titled 'Deployed Instances of this Data Source (Filtered - More Columns Exist)' with a table showing one instance:

Server	Enabled	State	JDBC Driver
msp1	true	Running	com.mysql.cj.jdbc.Driver

4. Sélectionnez l'onglet **Test**, puis sélectionnez la case d'option en regard du serveur souhaité.
5. Sélectionnez **Tester la source de données**. Vous devez voir un message indiquant un test réussi, comme illustré dans la capture d'écran suivante.

Messages

✓ Test of testpasswordless on server admin was successful.

Settings for testpasswordless

Configuration Targets **Monitoring** Control Security Notes

Statistics **Testing**

Use this page to test database connections in this JDBC data source.

▶ **Customize this table**

Test Data Source (Filtered - More Columns Exist)

Test Data Source		Showing 1 to 1 of 1 Previous Next
	Server	State
<input checked="" type="radio"/>	admin	Running

Test Data Source Showing 1 to 1 of 1 Previous | Next 

Nettoyer les ressources

Si vous n'avez pas besoin de ces ressources, vous pouvez les supprimer en effectuant les commandes suivantes :

Azure CLI

```
az group delete --name ${RESOURCE_GROUP_NAME}  
az group delete --name <resource-group-name-that-deploys-the-offer>
```

Étapes suivantes

Pour en savoir plus sur l'exécution de WLS sur AKS ou sur des machines virtuelles, suivez ces liens :

[WLS sur AKS](#)

[WLS sur des machines virtuelles](#)

[Exemples de Connexions sans mot de passe pour les applications Java](#)

Tutoriel : Déployer une application Spring sur Azure Spring Apps avec une connexion sans mot de passe à une base de données Azure

Article • 23/10/2023

Cet article explique comment utiliser des connexions sans mot de passe aux bases de données Azure dans les applications Spring Boot déployées sur Azure Spring Apps.

Dans ce tutoriel, vous allez effectuer les tâches suivantes à l'aide du portail Azure ou d'Azure CLI. Les deux méthodes sont expliquées dans les procédures suivantes.

- ✓ Provisionnez une instance d'Azure Spring Apps.
- ✓ Créez et déployez des applications sur Azure Spring Apps.
- ✓ Exécutez des applications connectées à des bases de données Azure à l'aide d'une identité managée.

ⓘ Notes

Ce didacticiel ne fonctionne pas pour R2DBC.

Prérequis

- Un abonnement Azure. Si vous n'en avez pas encore, créez un [compte gratuit](#) avant de commencer.
- [Azure CLI](#) 2.45.0 ou version ultérieure requise.
- L'extension Azure Spring Apps. Vous pouvez installer l'extension à l'aide de la commande : `az extension add --name spring`.
- [Kit de développement Java \(JDK\)](#), version 8, 11 ou 17.
- Un client [Git](#).
- [cURL](#) ou un utilitaire HTTP similaire pour tester la fonctionnalité.
- Client de ligne de commande MySQL si vous choisissez d'exécuter Azure Database pour MySQL. Vous pouvez vous connecter à votre serveur avec Azure Cloud Shell à l'aide d'un outil client populaire, l'outil [en ligne de commande mysql.exe](#). Vous pouvez aussi utiliser la ligne de commande de `mysql` dans votre environnement local.
- [ODBC Driver 18 pour SQL Server](#) si vous choisissez d'exécuter Azure SQL Database.

Préparer l'environnement de travail

Tout d'abord, configurez des variables d'environnement à l'aide des commandes suivantes :

Bash

```
export AZ_RESOURCE_GROUP=passwordless-tutorial-rg
export AZ_DATABASE_SERVER_NAME=<YOUR_DATABASE_SERVER_NAME>
export AZ_DATABASE_NAME=demodb
export AZ_LOCATION=<YOUR_AZURE_REGION>
export AZ_SPRING_APPS_SERVICE_NAME=<YOUR_AZURE_SPRING_APPS_SERVICE_NAME>
export AZ_SPRING_APPS_APP_NAME=hellospring
export AZ_DB_ADMIN_USERNAME=<YOUR_DB_ADMIN_USERNAME>
export AZ_DB_ADMIN_PASSWORD=<YOUR_DB_ADMIN_PASSWORD>
export AZ_USER_IDENTITY_NAME=<YOUR_USER_ASSIGNED_MANAGEMED_IDENTITY_NAME>
```

Remplacez les espaces réservés par les valeurs suivantes, qui sont utilisées dans cet article :

- <YOUR_DATABASE_SERVER_NAME> : nom de votre serveur Azure Database, qui doit être unique sur Azure.
- <YOUR_AZURE_REGION> : région Azure que vous souhaitez utiliser. Vous pouvez utiliser `eastus` par défaut, mais nous vous recommandons de configurer une région plus proche de l'endroit où vous vivez. Vous pouvez voir la liste complète des régions disponibles à l'aide `az account list-locations` de .
- <YOUR_AZURE_SPRING_APPS_SERVICE_NAME> : nom de votre instance Azure Spring Apps. Le nom doit comporter entre 4 et 32 caractères, et contenir uniquement des lettres minuscules, des chiffres et des traits d'union. Le premier caractère du nom du service doit être une lettre, et le dernier doit être une lettre ou un chiffre.
- <AZ_DB_ADMIN_USERNAME> : nom d'utilisateur administrateur de votre serveur de base de données Azure.
- <AZ_DB_ADMIN_PASSWORD> : mot de passe administrateur de votre serveur de base de données Azure.
- <YOUR_USER_ASSIGNED_MANAGEMED_IDENTITY_NAME> : Le nom du serveur d'identité managée affectée par l'utilisateur, qui doit être unique dans tout Azure.

Provisionner une instance d'Azure Spring Apps

Procédez comme suit pour provisionner une instance d'Azure Spring Apps.

1. Mettez à jour Azure CLI avec l'extension Azure Spring Apps à l'aide de la commande suivante :

Azure CLI

```
az extension update --name spring
```

2. Connectez-vous à Azure CLI et choisissez votre abonnement actif à l'aide des commandes suivantes :

Azure CLI

```
az login  
az account list --output table  
az account set --subscription <name-or-ID-of-subscription>
```

3. Utilisez les commandes suivantes pour créer un groupe de ressources pour contenir votre service Azure Spring Apps et une instance du service Azure Spring Apps :

Azure CLI

```
az group create \  
  --name $AZ_RESOURCE_GROUP \  
  --location $AZ_LOCATION  
az spring create \  
  --resource-group $AZ_RESOURCE_GROUP \  
  --name $AZ_SPRING_APPS_SERVICE_NAME
```

Créer une instance de base de données Azure

Procédez comme suit pour provisionner une instance Azure Database.

Azure SQL Database

1. Créez un serveur Azure SQL Database à l'aide de la commande suivante :

Azure CLI

```
az sql server create \  
  --location $AZ_LOCATION \  
  --resource-group $AZ_RESOURCE_GROUP \  
  --name $AZ_DATABASE_SERVER_NAME \  
  --admin-user $AZ_DB_ADMIN_USERNAME \  
  --admin-password $AZ_DB_ADMIN_PASSWORD
```

2. Le serveur SQL est vide. Créez donc une base de données à l'aide de la commande suivante :

Azure CLI

```
az sql db create \
--resource-group $AZ_RESOURCE_GROUP \
--server $AZ_DATABASE_SERVER_NAME \
--name $AZ_DATABASE_NAME
```

Créer une application avec un point de terminaison public affecté

Utilisez la commande suivante pour créer l'application.

Azure CLI

```
az spring app create \
--resource-group $AZ_RESOURCE_GROUP \
--service $AZ_SPRING_APPS_SERVICE_NAME \
--name $AZ_SPRING_APPS_APP_NAME \
--runtime-version=Java_17
--assign-endpoint true
```

Connecter Azure Spring Apps à la base de données Azure

Tout d'abord, installez l'extension [sans mot de passe du service Connecter or](#) pour Azure CLI :

Azure CLI

```
az extension add --name serviceconnector-passwordless --upgrade
```

Azure SQL Database

ⓘ Notes

Assurez-vous qu'Azure CLI utilise Python 64 bits, Python 32 bits présente un problème de compatibilité avec le pyodbc [de dépendance ↗](#) de la commande.

Les informations Python d'Azure CLI peuvent être obtenues avec la commande `az --version`. S'il s'affiche [MSC v.1929 32 bit (Intel)], cela signifie qu'il utilise Python 32 bits. La solution consiste à installer Python 64 bits et à installer Azure CLI à partir de [PyPI](#).

Utilisez la commande suivante pour créer une connexion sans mot de passe à la base de données.

Azure CLI

```
az spring connection create sql \
--resource-group $AZ_RESOURCE_GROUP \
--service $AZ_SPRING_APPS_SERVICE_NAME \
--app $AZ_SPRING_APPS_APP_NAME \
--target-resource-group $AZ_RESOURCE_GROUP \
--server $AZ_DATABASE_SERVER_NAME \
--database $AZ_DATABASE_NAME \
--system-identity
```

Cette commande de connecteur de services effectue les tâches suivantes en arrière-plan :

- Activez l'identité managée affectée par le système pour l'application `$AZ_SPRING_APPS_APP_NAME` hébergée par Azure Spring Apps.
- Définissez l'administrateur Microsoft Entra sur l'utilisateur de connexion actuel.
- Ajoutez un utilisateur de base de données nommé `$AZ_SPRING_APPS_SERVICE_NAME/apps/$AZ_SPRING_APPS_APP_NAME` pour l'identité managée créée à l'étape 1 et accordez tous les priviléges de la base de données `$AZ_DATABASE_NAME` à cet utilisateur.
- Ajoutez une configuration à l'application `$AZ_SPRING_APPS_APP_NAME`:
`spring.datasource.url`.

ⓘ Notes

Si le message d'erreur `The subscription is not registered to use Microsoft.ServiceLinker` apparaît, exécutez la commande `az provider register --namespace Microsoft.ServiceLinker` pour enregistrer le

fournisseur de ressources du connecteur de services, puis exécutez à nouveau la commande de connexion.

Génération et déploiement de l'application

Les étapes suivantes décrivent comment télécharger, configurer, générer et déployer l'exemple d'application.

1. Utilisez la commande suivante pour cloner l'exemple de référentiel de code :

```
Azure SQL Database

Bash

git clone https://github.com/Azure-Samples/quickstart-spring-data-jdbc-sql-server-passwordless-sample
```

2. Ajoutez la dépendance suivante à votre *fichier pom.xml* :

```
Azure SQL Database

XML

<dependency>
    <groupId>com.azure</groupId>
    <artifactId>azure-identity</artifactId>
    <version>1.5.4</version>
</dependency>
```

Il n'existe actuellement aucun démarrage Azure Spring Cloud pour Azure SQL Database, mais la `azure-identity` dépendance est requise.

3. Utilisez la commande suivante pour mettre à jour le *fichier application.properties* :

```
Azure SQL Database

Bash

cat << EOF > passwordless-
sample/src/main/resources/application.properties
```

```
logging.level.org.springframework.jdbc.core=DEBUG  
spring.sql.init.mode=always
```

```
EOF
```

4. Utilisez les commandes suivantes pour générer le projet à l'aide de Maven :

Bash

```
cd passwordless-sample  
.mvnw clean package -DskipTests
```

5. Utilisez la commande suivante pour déployer le *fichier target/demo-0.0.1-SNAPSHOTjar* pour l'application :

Azure CLI

```
az spring app deploy \  
--name $AZ_SPRING_APPS_APP_NAME \  
--service $AZ_SPRING_APPS_SERVICE_NAME \  
--resource-group $AZ_RESOURCE_GROUP \  
--artifact-path target/demo-0.0.1-SNAPSHOT.jar
```

6. Interrogez l'état de l'application après le déploiement à l'aide de la commande suivante :

Azure CLI

```
az spring app list \  
--service $AZ_SPRING_APPS_SERVICE_NAME \  
--resource-group $AZ_RESOURCE_GROUP \  
--output table
```

Vous devez voir la sortie similaire à l'exemple suivant.

Name	Location	ResourceGroup	Production Deployment	
Public Url			Provisioning	
Status	CPU	Memory	Running Instance	Registered Instance
Persistent Storage				
<app name>	eastus	<resource group>	default	

Test de l'application

Pour tester l'application, vous pouvez utiliser cURL. Tout d'abord, créez un élément « todo » dans la base de données à l'aide de la commande suivante :

Bash

```
curl --header "Content-Type: application/json" \
--request POST \
--data '{"description":"configuration","details":"congratulations, you
have set up JDBC correctly!","done": "true"}' \
https://${AZ_SPRING_APPS_SERVICE_NAME}-
hellospring.azuremicroservices.io
```

Cette commande retourne l'élément créé, comme illustré dans l'exemple suivant :

JSON

```
{"id":1,"description":"configuration","details":"congratulations, you have
set up JDBC correctly!","done":true}
```

Ensuite, récupérez les données à l'aide de la requête cURL suivante :

Bash

```
curl https://${AZ_SPRING_APPS_SERVICE_NAME}-
hellospring.azuremicroservices.io
```

Cette commande retourne la liste des éléments « todo », y compris l'élément que vous avez créé, comme illustré dans l'exemple suivant :

JSON

```
[{"id":1,"description":"configuration","details":"congratulations, you have
set up JDBC correctly!","done":true}]
```

Nettoyer les ressources

Pour propre toutes les ressources utilisées pendant ce didacticiel, supprimez le groupe de ressources à l'aide de la commande suivante :

Azure CLI

```
az group delete \
--name $AZ_RESOURCE_GROUP \
--yes
```

Étapes suivantes

- Documentation Spring Cloud Azure

Utiliser Spring Data JPA avec Azure Database pour MySQL

Article • 23/10/2023

Ce tutoriel montre comment stocker des données dans [Azure Database pour MySQL](#) base de données à l'aide [de Spring Data JPA](#).

L'[API de persistance Java \(JPA\)](#) est l'API Java standard pour le mappage objet-relationnel.

Dans ce tutoriel, nous incluons deux méthodes d'authentification : l'authentification Microsoft Entra et l'authentification MySQL. L'onglet **Sans mot de passe** affiche l'authentification Microsoft Entra et l'onglet **Mot de passe** l'authentification MySQL.

L'authentification Microsoft Entra est un mécanisme de connexion à Azure Database pour MySQL à l'aide des identités définies dans Microsoft Entra ID. Avec l'authentification Microsoft Entra, vous pouvez gérer les identités des utilisateurs de base de données et d'autres services Microsoft dans un emplacement centralisé, ce qui simplifie la gestion des autorisations.

L'authentification MySQL utilise des comptes stockés dans MySQL. Si vous choisissez d'utiliser des mots de passe comme informations d'identification pour les comptes, ces informations d'identification sont stockées dans la table `user`. Étant donné que ces mots de passe sont stockés dans MySQL, vous devez gérer la rotation des mots de passe par vous-même.

Prérequis

- Un abonnement Azure - [En créer un gratuitement](#)
- [Kit de développement Java \(JDK\)](#), version 8 ou ultérieure.
- [Apache Maven](#).
- [Azure CLI](#).
- [Client de ligne de commande MySQL](#).
- Si vous n'avez pas d'application Spring Boot, créez un projet Maven avec [Spring Initializr](#). Veillez à sélectionner **Maven Project** et, sous **Dépendances**, ajoutez les dépendances **Spring Web**, **Spring Data JPA** et **MySQL Driver**, puis sélectionnez Java version 8 ou ultérieure.

- Si vous n'en avez pas, créez une instance de serveur flexible Azure Database pour MySQL nommée `mysqlflexibletest`. Pour obtenir des instructions, consultez [Démarrage rapide : Utilisez le Portail Azure pour créer un serveur flexible Azure Database pour MySQL](#). Ensuite, créez une base de données nommée `demo`. Pour obtenir des instructions, consultez [Créer et gérer des bases de données pour Azure Database pour MySQL serveur flexible](#).

ⓘ Important

Pour utiliser des connexions sans mot de passe, créez un utilisateur administrateur Microsoft Entra pour votre instance de Azure Database pour MySQL. Pour obtenir des instructions, consultez la [section Configurer microsoft Entra Administration de Configurer l'authentification Microsoft Entra pour Azure Database pour MySQL - Serveur flexible](#).

Voir l'exemple d'application

Dans ce tutoriel, vous allez coder un exemple d'application. Si vous souhaitez aller plus vite, cette application est déjà codée et disponible sur <https://github.com/Azure-Samples/quickstart-spring-data-jpa-mysql>.

Configurer une règle de pare-feu pour votre serveur MySQL

Les instances Azure Database pour MySQL sont sécurisées par défaut. Elles ont un pare-feu qui n'autorise aucune connexion entrante.

Pour pouvoir utiliser votre base de données, ouvrez le pare-feu du serveur pour autoriser l'adresse IP locale à accéder au serveur de base de données. Pour plus d'informations, consultez [Gérer les règles de pare-feu pour Azure Database pour MySQL - Serveur flexible à l'aide du Portail Azure](#).

Si vous vous connectez à votre serveur MySQL à partir de Sous-système Windows pour Linux (WSL) sur un ordinateur Windows, vous devez ajouter l'adresse IP de l'hôte WSL à votre pare-feu.

Créer un utilisateur non-administrateur MySQL et accorder des autorisations

Cette étape crée un utilisateur non administrateur et lui accorde toutes les autorisations sur la `demo` base de données.

Sans mot de passe (recommandé)

Vous pouvez utiliser la méthode suivante pour créer un utilisateur non administrateur qui utilise une connexion sans mot de passe.

Service Connector (recommandé)

1. Utilisez la commande suivante pour installer l'extension [sans mot de passe du service Connecter or](#) pour Azure CLI :

Azure CLI

```
az extension add --name serviceconnector-passwordless --upgrade
```

2. Utilisez la commande suivante pour créer l'utilisateur non administrateur Microsoft Entra :

Azure CLI

```
az connection create mysql-flexible \
    --resource-group <your_resource_group_name> \
    --connection mysql_conn \
    --target-resource-group <your_resource_group_name> \
    --server mysqlflexibletest \
    --database demo \
    --user-account mysql-identity-
    id=/subscriptions/<your_subscription_id>/resourcegroups/<your_r
    esource_group_name>/providers/Microsoft.ManagedIdentity/userAss
    ignedIdentities/<your_user_assigned_managed_identity_name> \
    --query authInfo.userName \
    --output tsv
```

Une fois la commande terminée, notez le nom d'utilisateur dans la sortie de la console.

Stocker des données à partir de Azure Database pour MySQL

Maintenant que vous disposez d'une instance de serveur flexible Azure Database pour MySQL, vous pouvez stocker des données à l'aide de Spring Cloud Azure.

Pour installer le module Spring Cloud Azure Starter JDBC MySQL, ajoutez les dépendances suivantes à votre *fichier pom.xml* :

- The Spring Cloud Azure Bill of Materials (BOM) :

XML

```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>com.azure.spring</groupId>
      <artifactId>spring-cloud-azure-dependencies</artifactId>
      <version>4.13.0</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>
```

ⓘ Notes

Si vous utilisez Spring Boot 3.x, veillez à définir la `spring-cloud-azure-dependencies` version `5.7.0` sur . Pour plus d'informations sur la `spring-cloud-azure-dependencies` version, consultez [La version de Spring Cloud Azure à utiliser ↗](#).

- Artefact JDBC MySQL Spring Cloud Azure Starter :

XML

```
<dependency>
  <groupId>com.azure.spring</groupId>
  <artifactId>spring-cloud-azure-starter-jdbc-mysql</artifactId>
</dependency>
```

ⓘ Notes

Les connexions sans mot de passe ont été prises en charge depuis la version `4.5.0`.

Configurer Spring Boot pour qu'il utilise Azure Database pour MySQL

Pour stocker des données à partir de Azure Database pour MySQL à l'aide de Spring Data JPA, procédez comme suit pour configurer l'application :

1. Configurez Azure Database pour MySQL informations d'identification en ajoutant les propriétés suivantes à votre *fichier de configuration application.properties*.

Sans mot de passe (recommandé)

properties

```
logging.level.org.hibernate.SQL=DEBUG  
  
spring.datasource.azure.passwordless-enabled=true  
spring.datasource.url=jdbc:mysql://mysqlflexibletest.mysql.database.azure.com:3306/demo?serverTimezone=UTC  
spring.datasource.username=<your_mysql_ad_non_admin_username>  
  
spring.jpa.hibernate.ddl-auto=create-drop  
spring.jpa.properties.hibernate.dialect  
=org.hibernate.dialect.MySQL8Dialect
```

⚠ Avertissement

?serverTimezone=UTC a été ajouté à la propriété de configuration spring.datasource.url pour indiquer au pilote JDBC d'utiliser le format de date UTC (Coordinated Universal Time) lors de la connexion à la base de données. Sans ce paramètre, votre serveur Java n'utilise pas le même format de date que la base de données, ce qui entraînerait une erreur.

2. Créez une Todo classe Java. Cette classe est un modèle de domaine mappé sur la todo table qui sera créée automatiquement par JPA. Le code suivant ignore les méthodes et setters les getters méthodes.

Java

```
package com.example.demo;  
  
import javax.persistence.Entity;  
import javax.persistence.GeneratedValue;  
import javax.persistence.Id;
```

```

@Entity
public class Todo {

    public Todo() {
    }

    public Todo(String description, String details, boolean done) {
        this.description = description;
        this.details = details;
        this.done = done;
    }

    @Id
    @GeneratedValue
    private Long id;

    private String description;

    private String details;

    private boolean done;
}

}

```

3. Modifiez le fichier de classe de démarrage pour afficher le contenu suivant.

Java

```

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.boot.context.event.ApplicationReadyEvent;
import org.springframework.context.ApplicationListener;
import org.springframework.context.annotation.Bean;
import org.springframework.data.jpa.repository.JpaRepository;

import java.util.stream.Collectors;
import java.util.stream.Stream;

@SpringBootApplication
public class DemoApplication {

    public static void main(String[] args) {
        SpringApplication.run(DemoApplication.class, args);
    }

    @Bean
    ApplicationListener<ApplicationReadyEvent>
    basicsApplicationListener(TodoRepository repository) {
        return event->repository
            .saveAll(Stream.of("A", "B", "C").map(name->new
Todo("configuration", "congratulations, you have set up correctly!",
true)).collect(Collectors.toList()))
            .forEach(System.out::println);
    }
}

```

```
    }

}

interface TodoRepository extends JpaRepository<Todo, Long> {
```

Conseil

Dans ce tutoriel, il n'existe aucune opération d'authentification dans les configurations ou le code. Toutefois, la connexion aux services Azure nécessite une authentification. Pour effectuer l'authentification, vous devez utiliser Identité Azure. Spring Cloud Azure utilise `DefaultAzureCredential`, que la bibliothèque d'identités Azure fournit pour vous aider à obtenir des informations d'identification sans aucune modification du code.

`DefaultAzureCredential` prend en charge plusieurs méthodes d'authentification et détermine quelle méthode doit être utilisée au moment de l'exécution. Cette approche permet à votre application d'utiliser différentes méthodes d'authentification dans différents environnements (tels que les environnements locaux et de production) sans implémenter de code spécifique à l'environnement. Pour plus d'informations, consultez [DefaultAzureCredential](#).

Pour terminer l'authentification dans les environnements de développement locaux, vous pouvez utiliser Azure CLI, Visual Studio Code, PowerShell ou d'autres méthodes. Pour plus d'informations, consultez [l'authentification Azure dans les environnements](#) de développement Java. Pour terminer l'authentification dans les environnements d'hébergement Azure, nous vous recommandons d'utiliser l'identité managée affectée par l'utilisateur. Pour plus d'informations, consultez [Que sont les identités managées pour les ressources Azure ?](#)

4. Lancez l'application. Vous verrez des journaux similaires à l'exemple suivant :

```
shell
```

```
2023-02-01 10:29:19.763 DEBUG 4392 --- [main] org.hibernate.SQL :  
insert into todo (description, details, done, id) values (?, ?, ?, ?)  
com.example.demo.Todo@1f
```

Déployer sur Azure Spring Apps

Maintenant que vous disposez de l'application Spring Boot en cours d'exécution localement, il est temps de le déplacer en production. [Azure Spring Apps](#) facilite le déploiement d'applications Spring Boot sur Azure sans aucune modification de code. Le service gère l'infrastructure des applications Spring, ce qui permet aux développeurs de se concentrer sur leur code. Azure Spring Apps assure la gestion du cycle de vie en utilisant des outils complets, tels que la supervision et les diagnostics, la gestion des configurations, la découverte de services, l'intégration CI/CD, les déploiements bleus-verts, etc. Pour déployer votre application sur Azure Spring Apps, consultez [Déployer votre première application sur Azure Spring Apps](#).

Étapes suivantes

[Exemples Azure pour les développeurs](#)

[Spring Cloud Azure MySQL](#)

Utiliser Spring Data JPA avec Azure Database pour PostgreSQL

Article • 23/10/2023

Ce tutoriel montre comment stocker des données dans [Azure Database pour PostgreSQL](#) à l'aide de [Spring Data JPA](#).

L'[API de persistance Java \(JPA\)](#) est l'API Java standard pour le mappage objet-relationnel.

Dans ce tutoriel, nous incluons deux méthodes d'authentification : l'authentification Microsoft Entra et l'authentification PostgreSQL. L'onglet **Sans mot de passe** affiche l'authentification Microsoft Entra et l'onglet **Mot de passe** affiche l'authentification PostgreSQL.

L'authentification Microsoft Entra est un mécanisme de connexion à Azure Database pour PostgreSQL utilisant les identités définies dans Microsoft Entra ID. Avec l'authentification Microsoft Entra, vous pouvez gérer les identités des utilisateurs de base de données et d'autres services Microsoft dans un emplacement centralisé, ce qui simplifie la gestion des autorisations.

L'authentification PostgreSQL utilise des comptes stockés dans PostgreSQL. Si vous choisissez d'utiliser des mots de passe comme informations d'identification pour les comptes, ces informations d'identification sont stockées dans la table `user`. Étant donné que ces mots de passe sont stockés dans PostgreSQL, vous devez gérer la rotation des mots de passe par vous-même.

Prérequis

- Un abonnement Azure - [En créer un gratuitement](#)
- Kit de développement Java (JDK), version 8 ou ultérieure.
- [Apache Maven](#).
- [Azure CLI](#).
- [Client](#) de ligne de commande PostgreSQL.
- Si vous n'avez pas d'application Spring Boot, créez un projet Maven avec [Spring Initializr](#). Veillez à sélectionner **Maven Project** et, sous **Dépendances**, ajoutez les

dépendances Spring Web, Spring Data JDBC et PostgreSQL Driver, puis sélectionnez Java version 8 ou ultérieure.

- Si vous n'en avez pas, créez une instance de serveur flexible Azure Database pour PostgreSQL nommée `postgresqlflexibletest` et une base de données nommée `demo`. Pour obtenir des instructions, consultez [Démarrage rapide : Créer un serveur flexible Azure Database pour PostgreSQL dans le Portail Azure](#).

ⓘ Important

Pour utiliser des connexions sans mot de passe, configurez l'utilisateur administrateur Microsoft Entra pour votre instance de serveur flexible Azure Database pour PostgreSQL. Pour plus d'informations, consultez [Gérer les rôles Microsoft Entra dans Azure Database pour PostgreSQL - Serveur flexible](#).

Voir l'exemple d'application

Dans ce tutoriel, vous allez coder un exemple d'application. Si vous souhaitez aller plus vite, cette application est déjà codée et disponible sur [https://github.com/Azure-Samples/quickstart-spring-data-jpa-postgresql ↗](https://github.com/Azure-Samples/quickstart-spring-data-jpa-postgresql).

Configurer une règle de pare-feu pour votre serveur PostgreSQL

Les instances Azure Database pour PostgreSQL sont sécurisées par défaut. Elles ont un pare-feu qui n'autorise aucune connexion entrante.

Pour pouvoir utiliser votre base de données, ouvrez le pare-feu du serveur pour autoriser l'adresse IP locale à accéder au serveur de base de données. Pour plus d'informations, consultez [les règles de pare-feu dans Azure Database pour PostgreSQL - Serveur flexible](#).

Si vous vous connectez à votre serveur PostgreSQL à partir de Sous-système Windows pour Linux (WSL) sur un ordinateur Windows, vous devez ajouter l'ID d'hôte WSL à votre pare-feu.

Créer un utilisateur non administrateur PostgreSQL et accorder des autorisations

Ensuite, créez un utilisateur non-administrateur et accordez toutes les autorisations à la base de données.

Sans mot de passe (recommandé)

Vous pouvez utiliser la méthode suivante pour créer un utilisateur non administrateur qui utilise une connexion sans mot de passe.

Service Connector (recommandé)

1. Utilisez la commande suivante pour installer l'extension [sans mot de passe du service Connecter or](#) pour Azure CLI :

Azure CLI

```
az extension add --name serviceconnector-passwordless --upgrade
```

2. Utilisez la commande suivante pour créer l'utilisateur non administrateur Microsoft Entra :

Azure CLI

```
az connection create postgres-flexible \
    --resource-group <your_resource_group_name> \
    --connection postgres_conn \
    --target-resource-group <your_resource_group_name> \
    --server postgresqlflexibletest \
    --database demo \
    --user-account \
    --query authInfo.userName \
    --output tsv
```

Une fois la commande terminée, notez le nom d'utilisateur dans la sortie de la console.

Stocker des données à partir de Azure Database pour PostgreSQL

Maintenant que vous disposez d'une instance de serveur flexible Azure Database pour PostgreSQL, vous pouvez stocker des données à l'aide de Spring Cloud Azure.

Pour installer le module PostgreSQL JDBC Jdbc Spring Cloud Azure Starter, ajoutez les dépendances suivantes à votre *fichier pom.xml* :

- The Spring Cloud Azure Bill of Materials (BOM) :

XML

```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>com.azure.spring</groupId>
      <artifactId>spring-cloud-azure-dependencies</artifactId>
      <version>4.13.0</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>
```

ⓘ Notes

Si vous utilisez Spring Boot 3.x, veillez à définir la `spring-cloud-azure-dependencies` version `5.7.0` sur . Pour plus d'informations sur la `spring-cloud-azure-dependencies` version, consultez [La version de Spring Cloud Azure à utiliser ↗](#).

- L'artefact Jdbc PostgreSQL JDBC Spring Cloud Azure Starter :

XML

```
<dependency>
  <groupId>com.azure.spring</groupId>
  <artifactId>spring-cloud-azure-starter-jdbc-postgresql</artifactId>
</dependency>
```

ⓘ Notes

Les connexions sans mot de passe ont été prises en charge depuis la version `4.5.0`.

Configurer Spring Boot pour qu'il utilise Azure Database pour PostgreSQL

Pour stocker des données à partir de Azure Database pour PostgreSQL à l'aide de Spring Data JPA, procédez comme suit pour configurer l'application :

1. Configurez Azure Database pour PostgreSQL informations d'identification en ajoutant les propriétés suivantes à votre *fichier de configuration application.properties*.

```
Sans mot de passe (recommandé)

properties

logging.level.org.hibernate.SQL=DEBUG

spring.datasource.url=jdbc:postgresql://postgresqlflexibletest.postgres.database.azure.com:5432/demo?sslmode=require
spring.datasource.username=<your_postgresql_ad_non_admin_username>
spring.datasource.azure.passwordless-enabled=true

spring.jpa.hibernate.ddl-auto=create-drop
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect
```

2. Créez une `Todo` classe Java. Cette classe est un modèle de domaine mappé sur la `todo` table qui sera créée automatiquement par JPA. Le code suivant ignore les méthodes et `setters` les `getters` méthodes.

```
Java

package com.example.demo;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;

@Entity
public class Todo {

    public Todo() {
    }

    public Todo(String description, String details, boolean done) {
        this.description = description;
        this.details = details;
        this.done = done;
    }

    @Id
    @GeneratedValue
```

```
    private Long id;  
  
    private String description;  
  
    private String details;  
  
    private boolean done;  
  
}
```

3. Modifiez le fichier de classe de démarrage pour afficher le contenu suivant.

Java

```
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
import org.springframework.boot.context.event.ApplicationReadyEvent;  
import org.springframework.context.ApplicationListener;  
import org.springframework.context.annotation.Bean;  
import org.springframework.data.jpa.repository.JpaRepository;  
  
import java.util.stream.Collectors;  
import java.util.stream.Stream;  
  
@SpringBootApplication  
public class DemoApplication {  
  
    public static void main(String[] args) {  
        SpringApplication.run(DemoApplication.class, args);  
    }  
  
    @Bean  
    ApplicationListener<ApplicationReadyEvent>  
    basicsApplicationListener(TodoRepository repository) {  
        return event->repository  
            .saveAll(Stream.of("A", "B", "C").map(name->new  
Todo("configuration", "congratulations, you have set up correctly!",  
true)).collect(Collectors.toList()))  
            .forEach(System.out::println);  
    }  
  
}  
  
interface TodoRepository extends JpaRepository<Todo, Long> {  
}
```

💡 Conseil

Dans ce tutoriel, il n'existe aucune opération d'authentification dans les configurations ou le code. Toutefois, la connexion aux services Azure nécessite

une authentification. Pour effectuer l'authentification, vous devez utiliser Identité Azure. Spring Cloud Azure utilise `DefaultAzureCredential`, que la bibliothèque d'identités Azure fournit pour vous aider à obtenir des informations d'identification sans aucune modification du code.

`DefaultAzureCredential` prend en charge plusieurs méthodes d'authentification et détermine quelle méthode doit être utilisée au moment de l'exécution. Cette approche permet à votre application d'utiliser différentes méthodes d'authentification dans différents environnements (tels que les environnements locaux et de production) sans implémenter de code spécifique à l'environnement. Pour plus d'informations, consultez [DefaultAzureCredential](#).

Pour terminer l'authentification dans les environnements de développement locaux, vous pouvez utiliser Azure CLI, Visual Studio Code, PowerShell ou d'autres méthodes. Pour plus d'informations, consultez [l'authentification Azure dans les environnements](#) de développement Java. Pour terminer l'authentification dans les environnements d'hébergement Azure, nous vous recommandons d'utiliser l'identité managée affectée par l'utilisateur. Pour plus d'informations, consultez [Que sont les identités managées pour les ressources Azure ?](#)

4. Lancez l'application. Vous verrez des journaux similaires à l'exemple suivant :

shell

```
2023-02-01 10:29:19.763 DEBUG 4392 --- [main] org.hibernate.SQL :  
insert into todo (description, details, done, id) values (?, ?, ?, ?)  
com.example.demo.Todo@1f
```

Déployer sur Azure Spring Apps

Maintenant que vous disposez de l'application Spring Boot en cours d'exécution localement, il est temps de le déplacer en production. [Azure Spring Apps](#) facilite le déploiement d'applications Spring Boot sur Azure sans aucune modification de code. Le service gère l'infrastructure des applications Spring, ce qui permet aux développeurs de se concentrer sur leur code. Azure Spring Apps assure la gestion du cycle de vie en utilisant des outils complets, tels que la supervision et les diagnostics, la gestion des configurations, la découverte de services, l'intégration CI/CD, les déploiements bleus-verts, etc. Pour déployer votre application sur Azure Spring Apps, consultez [Déployer votre première application sur Azure Spring Apps](#).

Étapes suivantes

[Exemples Azure pour les développeurs](#)

[Spring Cloud Azure PostgreSQL](#)

Utiliser Spring Kafka avec Azure Event Hubs pour l'API Kafka

Article • 23/10/2023

Ce tutoriel vous montre comment configurer un Spring Cloud Stream Binder basé sur Java afin d'utiliser Azure Event Hubs pour Kafka pour envoyer et recevoir des messages avec Azure Event Hubs. Pour plus d'informations, consultez [Utiliser Azure Event Hubs à partir d'applications Apache Kafka](#)

Dans ce tutoriel, nous allons inclure deux méthodes d'authentification : [l'authentification Microsoft Entra](#) et [l'authentification SAP](#) (Shared Access Signatures). L'onglet **Sans mot de passe** affiche l'authentification Microsoft Entra et l'onglet **chaîne** d'`Connecter ion` affiche l'authentification SAP.

L'authentification Microsoft Entra est un mécanisme de connexion à Azure Event Hubs pour Kafka à l'aide d'identités définies dans l'ID Microsoft Entra. Avec l'authentification Microsoft Entra, vous pouvez gérer les identités des utilisateurs de base de données et d'autres services Microsoft dans un emplacement centralisé, ce qui simplifie la gestion des autorisations.

L'authentification SAP utilise la chaîne de connexion de votre espace de noms Azure Event Hubs pour l'accès délégué à Event Hubs pour Kafka. Si vous choisissez d'utiliser des signatures d'accès partagé comme informations d'identification, vous devez gérer les chaîne de connexion par vous-même.

Prérequis

- Un abonnement Azure - [En créer un gratuitement](#)
- Kit de développement Java (JDK) version 8 ou ultérieure.
- Apache Maven [»,](#) version 3.2 ou ultérieure.
- cURL [»,](#) ou un utilitaire HTTP similaire pour tester la fonctionnalité.
- Azure Cloud Shell ou Azure CLI 2.37.0 ou version ultérieure.
- Un hub d'événements Azure. Si vous n'en avez pas, [créez un hub d'événements à l'aide de Portail Azure.](#)
- Une application Spring Boot. Si vous n'en avez pas, créez un projet Maven avec [Spring Initializr](#). Veillez à sélectionner **Maven Project** et, sous **Dépendances**,

ajoutez les dépendances Spring Web, Spring pour Apache Kafka et Cloud Stream, puis sélectionnez Java version 8 ou ultérieure.

ⓘ Important

Spring Boot version 2.5 ou ultérieure est nécessaire pour suivre les étapes décrites dans ce didacticiel.

Préparer les informations d'identification

Sans mot de passe (recommandé)

Azure Event Hubs prend en charge l'utilisation de Microsoft Entra ID pour autoriser les requêtes de ressources Event Hubs. Avec l'ID Microsoft Entra, vous pouvez utiliser le [contrôle d'accès en fonction du rôle Azure \(Azure RBAC\)](#) pour accorder des autorisations à un [principal](#) de sécurité, qui peut être un utilisateur ou un principal de service d'application.

Si vous souhaitez exécuter cet exemple localement avec l'authentification Microsoft Entra, assurez-vous que votre compte d'utilisateur s'est authentifié via Azure Shared Computer Toolkit pour IntelliJ, le plug-in de compte Azure Visual Studio Code ou Azure CLI. Vérifiez également que le compte a reçu des autorisations suffisantes.

ⓘ Notes

Lorsque vous utilisez des connexions sans mot de passe, vous devez accorder à votre compte l'accès aux ressources. Dans Azure Event Hubs, affectez le rôle et [Azure Event Hubs Data Sender](#) le [Azure Event Hubs Data Receiver](#) rôle au compte Microsoft Entra que vous utilisez actuellement. Pour plus d'informations sur l'octroi de rôles d'accès, consultez [Affecter des rôles Azure à l'aide des Portail Azure et autoriser l'accès aux ressources Event Hubs à l'aide de l'ID Microsoft Entra](#).

Envoyer et recevoir des messages d'Azure Event Hubs

Avec un hub d'événements Azure, vous pouvez envoyer et recevoir des messages à l'aide de Spring Cloud Azure.

Pour installer le module Spring Cloud Azure Starter, ajoutez les dépendances suivantes à votre *fichier pom.xml* :

- The Spring Cloud Azure Bill of Materials (BOM) :

XML

```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>com.azure.spring</groupId>
      <artifactId>spring-cloud-azure-dependencies</artifactId>
      <version>4.13.0</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>
```

ⓘ Notes

Si vous utilisez Spring Boot 3.x, veillez à définir la `spring-cloud-azure-dependencies` version `5.7.0` sur . Pour plus d'informations sur la `spring-cloud-azure-dependencies` version, consultez [La version de Spring Cloud Azure à utiliser ↗](#).

- Artefact Spring Cloud Azure Starter :

XML

```
<dependency>
  <groupId>com.azure.spring</groupId>
  <artifactId>spring-cloud-azure-starter</artifactId>
</dependency>
```

Coder l'application

Procédez comme suit pour configurer votre application pour produire et consommer des messages à l'aide d'Azure Event Hubs.

1. Configurez les informations d'identification event hub en ajoutant les propriétés suivantes à votre *fichier application.properties* .

Sans mot de passe (recommandé)

properties

```
spring.cloud.stream.kafka.binder.brokers=${AZ_EVENTHUBS_NAMESPACE_NAME}.servicebus.windows.net:9093
spring.cloud.function.definition=consume;supply
spring.cloud.stream.bindings.consume-in-0.destination=${AZ_EVENTHUB_NAME}
spring.cloud.stream.bindings.consume-in-0.group=$Default
spring.cloud.stream.bindings.supply-out-0.destination=${AZ_EVENTHUB_NAME}
```

💡 Conseil

Si vous utilisez la version `spring-cloud-azure-dependencies:4.3.0`, vous devez ajouter la propriété `spring.cloud.stream.binders.<kafka-binder-name>.environment.spring.main.sources` avec la valeur `com.azure.spring.cloud.autoconfigure.kafka.AzureKafkaSpringCloudStreamConfiguration`.

Étant donné que `4.4.0` cette propriété sera ajoutée automatiquement, il n'est donc pas nécessaire de l'ajouter manuellement.

Le tableau suivant décrit les champs de la configuration :

Champ	Description
<code>spring.cloud.stream.kafka.binder.brokers</code>	Spécifie le point de terminaison Azure Event Hubs.
<code>spring.cloud.stream.bindings.consume-in-0.destination</code>	Spécifie le hub d'événements de destination d'entrée, qui pour ce didacticiel est le hub que vous avez créé précédemment.
<code>spring.cloud.stream.bindings.consume-in-0.group</code>	Spécifie un groupe de consommateurs à partir d'Azure Event Hubs, que vous pouvez définir <code>\$Default</code> pour utiliser le groupe de consommateurs de base créé lors de la création de votre instance Azure Event Hubs.

Champ	Description
<code>spring.cloud.stream.bindings.supply-out-0.destination</code>	Spécifie le hub d'événements de destination de sortie, qui pour ce didacticiel est identique à la destination d'entrée.

ⓘ Notes

Si vous activez la création automatique de rubriques, veillez à ajouter l'élément `spring.cloud.stream.kafka.binder.replicationFactor` de configuration, avec la valeur définie sur au moins 1. Pour plus d'informations, consultez [Spring Cloud Stream Kafka Binder Reference Guide](#).

2. Modifiez le fichier de classe de démarrage pour afficher le contenu suivant.

Java

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.Bean;
import org.springframework.messaging.Message;
import org.springframework.messaging.support.GenericMessage;
import reactor.core.publisher.Flux;
import reactor.core.publisher.Sinks;
import java.util.function.Consumer;
import java.util.function.Supplier;

@SpringBootApplication
public class EventHubKafkaBinderApplication implements
CommandLineRunner {

    private static final Logger LOGGER =
LoggerFactory.getLogger(EventHubKafkaBinderApplication.class);

    private static final Sinks.Many<Message<String>> many =
Sinks.many().unicast().onBackpressureBuffer();

    public static void main(String[] args) {
        SpringApplication.run(EventHubKafkaBinderApplication.class,
args);
    }

    @Bean
    public Supplier<Flux<Message<String>>> supply() {
        return ()->many.asFlux()
    }
}
```

```

        .doOnNext(m->LOGGER.info("Manually sending
message {}", m))
        .doOnError(t->LOGGER.error("Error encountered",
t));
    }

    @Bean
    public Consumer<Message<String>> consume() {
        return message->LOGGER.info("New message received: '{}'",
message.getPayload());
    }

    @Override
    public void run(String... args) {
        many.emitNext(new GenericMessage<>("Hello World"),
Sinks.EmitFailureHandler.FAIL_FAST);
    }
}

```

Conseil

Dans ce tutoriel, il n'existe aucune opération d'authentification dans les configurations ou le code. Toutefois, la connexion aux services Azure nécessite une authentification. Pour effectuer l'authentification, vous devez utiliser Identité Azure. Spring Cloud Azure utilise `DefaultAzureCredential`, que la bibliothèque d'identités Azure fournit pour vous aider à obtenir des informations d'identification sans aucune modification du code.

`DefaultAzureCredential` prend en charge plusieurs méthodes d'authentification et détermine quelle méthode doit être utilisée au moment de l'exécution. Cette approche permet à votre application d'utiliser différentes méthodes d'authentification dans différents environnements (tels que les environnements locaux et de production) sans implémenter de code spécifique à l'environnement. Pour plus d'informations, consultez [DefaultAzureCredential](#).

Pour terminer l'authentification dans les environnements de développement locaux, vous pouvez utiliser Azure CLI, Visual Studio Code, PowerShell ou d'autres méthodes. Pour plus d'informations, consultez [l'authentification Azure dans les environnements](#) de développement Java. Pour terminer l'authentification dans les environnements d'hébergement Azure, nous vous recommandons d'utiliser l'identité managée affectée par l'utilisateur. Pour plus d'informations, consultez [Que sont les identités managées pour les ressources Azure ?](#)

3. Lancez l'application. Les messages comme l'exemple suivant seront publiés dans le journal de votre application :

Sortie

```
Kafka version: 3.0.1
Kafka commitId: 62abe01bee039651
Kafka startTimeMs: 1622616433956
New message received: 'Hello World'
```

Déployer sur Azure Spring Apps

Maintenant que vous disposez de l'application Spring Boot en cours d'exécution localement, il est temps de la déplacer en production. [Azure Spring Apps](#) facilite le déploiement d'applications Spring Boot sur Azure sans aucune modification de code. Le service gère l'infrastructure des applications Spring, ce qui permet aux développeurs de se concentrer sur leur code. Azure Spring Apps assure la gestion du cycle de vie en utilisant des outils complets, tels que la supervision et les diagnostics, la gestion des configurations, la découverte de services, l'intégration CI/CD, les déploiements bleus-verts, etc. Pour déployer votre application sur Azure Spring Apps, consultez [Déployer votre première application sur Azure Spring Apps](#).

Étapes suivantes

[Exemples Azure pour les développeurs](#)

[Spring Cloud Azure Stream Binder Event Hubs Kafka](#)

Démarrage rapide : Bibliothèque Azure Cosmos DB for NoSQL pour Java

Article • 11/01/2024

S'APPLIQUE À : NoSQL

Découvrez comment bien démarrer avec la bibliothèque de client Azure Cosmos DB for NoSQL pour Java afin d'interroger des données dans vos conteneurs et d'effectuer des opérations courantes sur des éléments spécifiques. Suivez ces étapes pour déployer une solution minimale dans votre environnement à l'aide d'Azure Developer CLI.

[Documentation de référence sur l'API](#) | [Code source de la bibliothèque](#) | [Package \(Maven\)](#) | [Azure Developer CLI](#)

Prérequis

- Compte Azure avec un abonnement actif. [Créez un compte gratuitement](#) .
- [Compte GitHub](#)

Configuration

Déployez le conteneur de développement de ce projet dans votre environnement. Utilisez ensuite Azure Developer CLI (`azd`) pour créer un compte Azure Cosmos DB for NoSQL et déployer un exemple d'application conteneurisé. L'exemple d'application utilise la bibliothèque de client pour gérer, créer, lire et interroger des exemples de données.



Important

Les comptes GitHub incluent un droit de stockage et des heures cœur sans coût supplémentaire. Pour plus d'informations, consultez [Stockage et heures cœur inclus dans les comptes GitHub](#) .

1. Ouvrez un terminal dans le répertoire racine du projet.
2. Authentifiez-vous auprès d'Azure Developer CLI à l'aide de `azd auth login`. Suivez les étapes spécifiées par l'outil pour vous authentifier auprès de l'interface CLI à

l'aide de vos informations d'identification Azure préférées.

```
Azure CLI
```

```
azd auth login
```

3. Utilisez `azd init` pour initialiser le projet.

```
Azure CLI
```

```
azd init
```

4. Lors de l'initialisation, configurez un nom d'environnement unique.

💡 Conseil

Le nom de l'environnement sera également utilisé comme nom du groupe de ressources cible. Pour ce guide de démarrage rapide, envisagez d'utiliser `msdocs-cosmos-db-nosql`.

5. Déployez le compte Azure Cosmos DB for NoSQL à l'aide de `azd up`. Les modèles Bicep déploient également un exemple d'application web.

```
Azure CLI
```

```
azd up
```

6. Pendant le processus d'approvisionnement, sélectionnez votre abonnement et l'emplacement souhaité. Attendez la fin du processus de provisionnement. Le processus peut prendre **environ cinq minutes**.

7. Une fois l'approvisionnement de vos ressources Azure terminé, une URL vers l'application web en cours d'exécution est incluse dans la sortie.

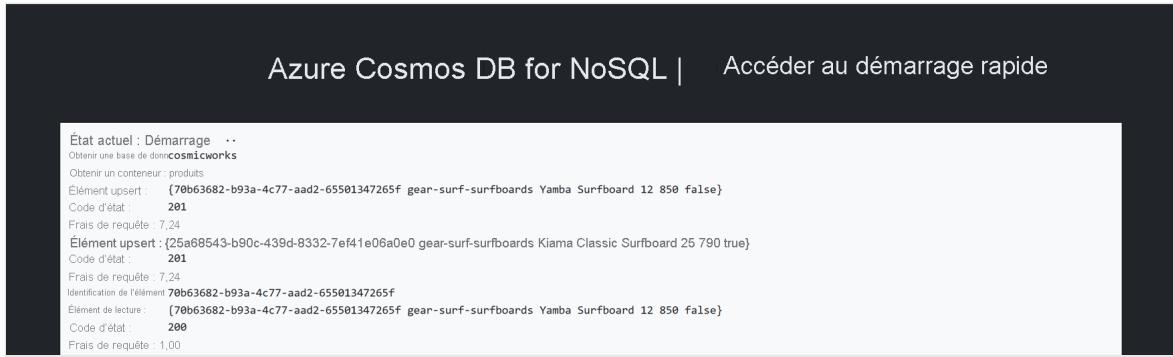
```
Output
```

```
Deploying services (azd deploy)
```

```
(✓) Done: Deploying service web
- Endpoint: <https://[container-app-sub-domain].azurecontainerapps.io>
```

```
SUCCESS: Your application was provisioned and deployed to Azure in 5
minutes 0 seconds.
```

8. Utilisez l'URL dans la console pour accéder à votre application web dans le navigateur. Observez la sortie de l'application en cours d'exécution.



Azure Cosmos DB for NoSQL | Accéder au démarrage rapide

```
État actuel : Démarrage ..  
Obtenir une base de données  
Obtenir un conteneur : produits  
Élément upsert : {70b63682-b93a-4c77-aad2-65501347265f gear-surf-surfboards Yamba Surfboard 12 850 false}  
Code d'état : 201  
Frais de requête : 7,24  
Élément upsert : {25a69543-b90c-439d-8332-7ef41e06a0e0 gear-surf-surfboards Kiama Classic Surfboard 25 790 true}  
Code d'état : 201  
Frais de requête : 7,24  
Identification de l'élément 70b63682-b93a-4c77-aad2-65501347265f  
Élément de lecture : {70b63682-b93a-4c77-aad2-65501347265f gear-surf-surfboards Yamba Surfboard 12 850 false}  
Code d'état : 200  
Frais de requête : 1,00
```

Installer la bibliothèque de client

La bibliothèque de client est disponible via Maven, sous forme de package `azure-spring-data-cosmos`.

1. Accédez au dossier `/src/web` et ouvrez le fichier `pom.xml`.

2. Si elle n'existe pas, ajoutez une entrée pour le package `azure-spring-data-cosmos`.

XML

```
<dependency>  
    <groupId>com.azure</groupId>  
    <artifactId>azure-spring-data-cosmos</artifactId>  
</dependency>
```

3. Ajoutez également une autre dépendance pour le package `azure-identity` s'il n'en existe aucune.

XML

```
<dependency>  
    <groupId>com.azure</groupId>  
    <artifactId>azure-identity</artifactId>  
</dependency>
```

Modèle objet

[+] Agrandir le tableau

Nom	Description
EnableCosmosRepositories	Ce type est un décorateur de méthode utilisé pour configurer un référentiel pour accéder à Azure Cosmos DB for NoSQL.
CosmosRepository	Cette classe est la classe cliente principale ; elle est utilisée pour gérer les données au sein d'un conteneur.
CosmosClientBuilder	Cette classe est une fabrique utilisée pour créer un client utilisé par le référentiel.
Query	Ce type est un décorateur de méthode utilisé pour spécifier la requête exécutée par le référentiel.

Exemples de code

- [Authentifier le client](#)
- [Obtenir une base de données](#)
- [Obtenir un conteneur](#)
- [Créer un élément](#)
- [Obtenir un élément](#)
- [Éléments de requête](#)

L'exemple de code du modèle utilise une base de données nommée `cosmicworks` et un conteneur nommé `products`. Le conteneur `products` contient des détails tels que le nom, la catégorie, la quantité, un identificateur unique et un indicateur de vente pour chaque produit. Le conteneur utilise la propriété `/category` comme clé de partition logique.

Authentifier le client

Les requêtes d'application vers les Services Azure doivent être autorisées. Utilisez le type `DefaultAzureCredential` comme moyen préféré d'implémenter une connexion sans mot de passe entre vos applications et Azure Cosmos DB for NoSQL.

`DefaultAzureCredential` prend en charge plusieurs méthodes d'authentification et détermine quelle méthode doit être utilisée au moment de l'exécution.

ⓘ Important

Vous pouvez également autoriser directement les requêtes adressées aux services Azure à l'aide de mots de passe, de chaînes de connexion ou d'autres informations d'identification. Toutefois, cette approche doit être utilisée avec

prudence. Les développeurs doivent être vigilants pour ne jamais exposer les secrets dans un emplacement non sécurisé. Toute personne ayant accès au mot de passe ou à la clé secrète peut s'authentifier auprès du service de base de données. `DefaultAzureCredential` offre des avantages de gestion et de sécurité améliorés par rapport à la clé de compte pour autoriser l'authentification sans mot de passe sans risque de stocker des clés.

Tout d'abord, cet exemple crée une classe qui hérite de `AbstractCosmosConfiguration` pour configurer la connexion à Azure Cosmos DB for NoSQL.

Java

```
@Configuration  
@EnableCosmosRepositories  
public class CosmosConfiguration extends AbstractCosmosConfiguration {
```

Dans la classe de configuration, cet exemple crée une nouvelle instance de la classe `CosmosClientBuilder`, et configure l'authentification à l'aide d'une instance `DefaultAzureCredential`.

Java

```
@Bean  
public CosmosClientBuilder getCosmosClientBuilder() {  
    DefaultAzureCredential azureTokenCredential = new  
DefaultAzureCredentialBuilder()  
    .build();  
  
    return new CosmosClientBuilder()  
        .endpoint(uri)  
        .credential(azureTokenCredential);  
}
```

Obtenir une base de données

Dans la classe de configuration, l'exemple implémente une méthode pour retourner le nom de la base de données existante nommée `cosmicworks`.

Java

```
@Override  
protected String getDatabaseName() {  
    return "cosmicworks";  
}
```

Obtenir un conteneur

Utilisez le décorateur de méthode `Container` afin de configurer une classe pour représenter des éléments dans un conteneur. Créez la classe pour inclure tous les membres que vous souhaitez sérialiser dans JSON. Dans cet exemple, le type a un identificateur unique et des champs pour la catégorie, le nom, la quantité, le prix et la réduction.

Java

```
@Container(containerName = "products", autoCreateContainer = false)
public class Item {
    private String id;
    private String name;
    private Integer quantity;
    private Boolean sale;

    @PartitionKey
    private String category;
```

Créer un élément

Créez un élément dans le conteneur à l'aide de `repository.save`.

Java

```
Item item = new Item(
    "70b63682-b93a-4c77-aad2-65501347265f",
    "gear-surf-surfboards",
    "Yamba Surfboard",
    12,
    false
);
Item created_item = repository.save(item);
```

Lire un élément

Effectuez une opération de lecture de point à l'aide des champs d'identificateur unique (`id`) et de clé de partition. Utilisez `repository.findById` pour récupérer efficacement l'élément spécifique.

Java

```
PartitionKey partitionKey = new PartitionKey("gear-surf-surfboards");
Optional<Item> existing_item = repository.findById("70b63682-b93a-4c77-aad2-65501347265f", partitionKey);
if (existing_item.isPresent()) {
    // Do something
}
```

Éléments de requête

Effectuez une requête sur plusieurs éléments d'un conteneur en définissant une requête dans l'interface du référentiel. Cet exemple utilise le décorateur de méthode `Query` pour définir une méthode qui exécute cette requête paramétrisable :

nosql

```
SELECT * FROM products p WHERE p.category = @category
```

Java

```
@Repository
public interface ItemRepository extends CosmosRepository<Item, String> {
    @Query("SELECT * FROM products p WHERE p.category = @category")
    List<Item> getItemsByCategory(@Param("category") String category);
}
```

Récupérez (fetch) tous les résultats de la requête à l'aide de `repository.getItemsByCategory`. Effectuez une boucle parmi les résultats de la requête.

Java

```
List<Item> items = repository.getItemsByCategory("gear-surf-surfboards");
for (Item item : items) {
    // Do something
}
```

Contenu connexe

- [Démarrage rapide .NET](#)
- [Démarrage rapide JavaScript/Node.js](#)
- [Démarrage rapide Java](#)
- [Démarrage rapide Go](#)

Étape suivante

Tutoriel : Générer une application web Java

Utiliser Java pour envoyer ou recevoir des événements d'Azure Event Hubs

Article • 12/12/2023

Ce guide de démarrage rapide montre comment recevoir des événements d'un hub d'événements et lui en envoyer en utilisant le package Java `azure-messaging-eventhubs`.

💡 Conseil

Si vous utilisez des ressources Azure Event Hubs dans une application Spring, nous vous recommandons de considérer [Spring Cloud Azure](#) comme alternative. Azure Spring Cloud est un projet open source qui fournit une intégration de Spring fluide aux services Azure. Pour en savoir plus sur Spring Cloud Azure et pour voir un exemple d'utilisation d'Event Hubs, consultez [Spring Cloud Stream avec Azure Event Hubs](#).

Prérequis

Si vous débutez avec Azure Event Hubs, consultez la [vue d'ensemble d'Event Hubs](#) avant de suivre ce guide de démarrage rapide.

Pour effectuer ce démarrage rapide, vous avez besoin de ce qui suit :

- **Abonnement Microsoft Azure.** Pour utiliser les services Azure, y compris Azure Event Hubs, vous avez besoin d'un abonnement. Si vous n'avez pas de compte Azure, vous pouvez vous inscrire à un [essai gratuit](#) ou utiliser les avantages de votre abonnement MSDN quand vous [créez un compte](#).
- Un environnement de développement Java. Ce guide de démarrage rapide utilise [Eclipse](#). Le Kit de développement Java (JDK) avec version 8 ou ultérieure est nécessaire.
- **Créez un espace de noms Event Hubs et un Event Hub.** La première étape consiste à utiliser le [portail Azure](#) pour créer un espace de noms de type Event Hubs et obtenir les informations de gestion nécessaires à votre application pour communiquer avec le concentrateur d'événements. Pour créer un espace de noms et un hub d'événements, suivez la procédure décrite dans [cet article](#). Ensuite, obtenez la **chaîne de connexion de l'espace de noms Event Hubs** en suivant les instructions à partir de l'article : [Obtenir la chaîne de connexion](#). Vous utiliserez la chaîne de connexion plus loin dans ce guide de démarrage rapide.

Envoyer des événements

Cette section montre comment créer une application Java pour envoyer des événements à un hub d'événements.

Ajouter une référence à la bibliothèque Azure Event Hubs

Créez tout d'abord un nouveau projet **Maven** pour une application de console/shell dans votre environnement de développement Java favori. Mettez le fichier `pom.xml` à jour comme suit. La bibliothèque cliente de Java pour Event Hubs est disponible dans le [Référentiel central Maven](#).

Sans mot de passe (recommandé)

XML

```
<dependency>
    <groupId>com.azure</groupId>
    <artifactId>azure-messaging-eventhubs</artifactId>
    <version>5.15.0</version>
</dependency>
<dependency>
    <groupId>com.azure</groupId>
    <artifactId>azure-identity</artifactId>
    <version>1.8.0</version>
    <scope>compile</scope>
</dependency>
```

ⓘ Notes

Procédez à une mise à jour vers la dernière version publiée sur le référentiel Maven.

Authentifier l'application sur Azure

Ce guide de démarrage rapide présente deux façons de vous connecter à Azure Event Hubs : sans mot de passe et avec une chaîne de connexion. La première option vous explique comment utiliser votre principal de sécurité dans Microsoft Entra ID et le contrôle d'accès en fonction du rôle (RBAC) pour vous connecter à un espace de noms Event Hubs. Vous n'avez pas à vous soucier d'avoir des chaînes de connexion codées en dur dans votre code, dans un fichier config ni dans un stockage sécurisé comme Azure Key Vault. La deuxième option consiste à se servir d'une chaîne de connexion pour se

connecter à un espace de noms Event Hubs. Si vous débutez avec Azure, vous trouverez peut-être l'option de chaîne de connexion plus facile à suivre. Nous vous recommandons d'utiliser l'option sans mot de passe dans les applications réelles et les environnements de production. Pour plus d'informations, consultez [Authentification et autorisation](#). Pour en savoir plus sur l'authentification sans mot de passe, reportez-vous à la [page de présentation](#).

Sans mot de passe (recommandé)

Attribuer des rôles à votre utilisateur Microsoft Entra

Lors du développement localement, vous devez vérifier que le compte d'utilisateur qui se connecte à Azure Event Hubs dispose des autorisations appropriées. Vous aurez besoin du rôle [Propriétaire de données Azure Event Hubs](#) pour envoyer et recevoir des messages. Pour vous attribuer ce rôle, vous aurez besoin du rôle Administrateur de l'accès utilisateur ou d'un autre rôle qui inclut l'action

`Microsoft.Authorization/roleAssignments/write`. Vous pouvez attribuer des rôles RBAC Azure à un utilisateur à l'aide du Portail Azure, Azure CLI ou Azure PowerShell. Découvrez les étendues disponibles pour les attributions de rôles dans la page [vue d'ensemble de l'étendue](#).

L'exemple suivant attribue le rôle `Azure Event Hubs Data Owner` à votre compte d'utilisateur, qui fournit un accès complet aux ressources Azure Event Hubs. Dans un scénario réel, suivez le [principe des privilèges minimum](#) pour accorder aux utilisateurs uniquement les autorisations minimales nécessaires à un environnement de production plus sécurisé.

Rôles intégrés Azure pour Azure Event Hubs

Pour Azure Event Hubs, la gestion des espaces de noms et de toutes les ressources associées via le portail Azure et l'API de gestion des ressources Azure est déjà protégée à l'aide du modèle RBAC Azure. Azure fournit les rôles Azure intégrés ci-dessous pour autoriser l'accès à un espace de noms Event Hubs :

- [Propriétaire de données Azure Event Hubs](#) : permet l'accès aux données d'un espace de noms Event Hubs et de ses entités (files d'attente, rubriques, abonnements et filtres).
- [Expéditeur de données Azure Event Hubs](#) : utilisez ce rôle pour autoriser l'expéditeur à accéder à l'espace de noms Event Hubs et à ses entités.

- **Récepteur de données Azure Event Hubs** : utilisez ce rôle pour autoriser l'expéditeur à accéder à l'espace de noms Event Hubs et à ses entités.

Si vous souhaitez créer un rôle personnalisé, consultez [Droits requis pour les opérations Service Bus](#).

ⓘ Important

Dans la plupart des cas, la propagation de l'attribution de rôle dans Azure peut prendre une ou deux minutes. Dans de rares cas, cela peut prendre jusqu'à huit minutes. Si vous recevez des erreurs d'authentification lorsque vous exécutez votre code pour la première fois, patientez quelques instants et réessayez.

Azure portal

1. Dans le portail Azure, recherchez votre espace de noms Event Hubs à l'aide de la barre de recherche principale ou de la navigation gauche.
2. Dans la page vue d'ensemble, sélectionnez **Contrôle d'accès (IAM)** dans le menu de gauche.
3. Sur la page **Contrôle d'accès (IAM)**, sélectionnez l'onglet **Attributions de rôles**.
4. Sélectionnez **+ Ajouter** dans le menu supérieur, puis **Ajouter une attribution de rôle** dans le menu déroulant résultant.

The screenshot shows the Azure portal interface for managing access control (IAM) in a Service Bus Namespace named 'spsbusns11102'. The left sidebar lists various management options like Overview, Activity log, and Tags. The 'Access control (IAM)' option is highlighted with a red box and a yellow circle containing the number 1. The main content area shows the 'Access control (IAM)' blade. At the top, there's a search bar, a 'Download role assignments' button, and other navigation links. A dropdown menu is open over the '+ Add' button, with the 'Add role assignment' option highlighted with a yellow circle containing the number 2. Another yellow circle containing the number 3 points to the 'Add role assignment' option itself. The right side of the blade displays sections for 'My access' (with a 'Check access' button), 'Check access' (with a 'Check access' button), 'Grant access to this resource' (with a 'Add role assignment' button), and 'View access to this resource' (with a 'View' button). A note at the top right states: 'To read this customer directory, so some options are disabled. If you require access to these options, add co-administrator'.

5. Utilisez la zone de recherche pour filtrer les résultats sur le rôle souhaité.

Pour cet exemple, recherchez Azure Event Hubs Data Owner et sélectionnez le résultat correspondant. Ensuite, choisissez Suivant.

6. Sous Attribuer l'accès à, sélectionnez Utilisateur, groupe ou principal de service, puis sélectionnez + Sélectionner des membres.

7. Dans la boîte de dialogue, recherchez votre nom d'utilisateur Microsoft Entra (généralement votre adresse e-mail *utilisateur@domaine*), puis choisissez Sélectionner en bas de la boîte de dialogue.

8. Sélectionnez Vérifier + affecter pour accéder à la page finale, puis Vérifier + attribuer à nouveau pour terminer le processus.

Écriture de code pour envoyer des messages à un hub d'événements

Sans mot de passe (recommandé)

Ajoutez une classe nommée Sender, et ajoutez-y le code suivant :

ⓘ Important

- Mettez à jour <NAMESPACE NAME> avec le nom de votre espace de noms Event Hub.
- Mettez à jour <EVENT HUB NAME> avec le nom de votre hub d'événements.

Java

```
package ehubquickstart;

import com.azure.messaging.eventhubs.*;
import java.util.Arrays;
import java.util.List;

import com.azure.identity.*;

public class SenderAAD {

    // replace <NAMESPACE NAME> with the name of your Event Hubs
    // namespace.
    // Example: private static final String namespaceName =
```

```
"contosons.servicebus.windows.net";
    private static final String namespaceName = "<NAMESPACE
NAME>.servicebus.windows.net";

    // Replace <EVENT HUB NAME> with the name of your event hub.
    // Example: private static final String eventHubName = "ordersehub";
private static final String eventHubName = "<EVENT HUB NAME>";

    public static void main(String[] args) {
        publishEvents();
    }
    /**
     * Code sample for publishing events.
     * @throws IllegalArgumentException if the EventData is bigger than
the max batch size.
     */
    public static void publishEvents() {
        // create a token using the default Azure credential
        DefaultAzureCredential credential = new
DefaultAzureCredentialBuilder()
            .authorityHost(AzureAuthorityHosts.AZURE_PUBLIC_CLOUD)
            .build();

        // create a producer client
        EventHubProducerClient producer = new EventHubClientBuilder()
            .fullyQualifiedNamespace(namespaceName)
            .eventHubName(eventHubName)
            .credential(credential)
            .buildProducerClient();

        // sample events in an array
        List<EventData> allEvents = Arrays.asList(new EventData("Foo"),
new EventData("Bar"));

        // create a batch
        EventDataBatch eventDataBatch = producer.createBatch();

        for (EventData eventData : allEvents) {
            // try to add the event from the array to the batch
            if (!eventDataBatch.tryAdd(eventData)) {
                // if the batch is full, send it and then create a new
batch
                producer.send(eventDataBatch);
                eventDataBatch = producer.createBatch();

                // Try to add that event that couldn't fit before.
                if (!eventDataBatch.tryAdd(eventData)) {
                    throw new IllegalArgumentException("Event is too
large for an empty batch. Max size: "
                        + eventDataBatch.getMaxSizeInBytes());
                }
            }
        }
        // send the last batch of remaining events
        if (eventDataBatch.getCount() > 0) {
```

```
        producer.send(eventDataBatch);
    }
    producer.close();
}
}
```

Générez le programme et assurez-vous qu'il n'y a aucune erreur. Vous exécuterez ce programme après avoir exécuté le programme récepteur.

Recevoir des événements

Le code de ce tutoriel est basé sur l'[exemple EventProcessorClient sur GitHub](#), que vous pouvez analyser pour voir la version complète de l'application en cours.

Suivez les recommandations ci-dessous quand vous utilisez Stockage Blob Azure comme magasin de points de contrôle :

- Utilisez un conteneur distinct pour chaque groupe de processeurs. Vous pouvez utiliser le même compte de stockage, mais utiliser un conteneur par groupe.
- N'utilisez pas le conteneur et le compte de stockage pour quoi que ce soit d'autre.
- Le compte de stockage doit se trouver dans la même région que l'application déployée. Si l'application est locale, essayez de choisir la région la plus proche possible.

Sur la page **Compte de stockage** du Portail Azure, dans la section **Service BLOB**, vérifiez que les paramètres suivants sont désactivés.

- Espace de noms hiérarchique
- Suppression réversible de blob
- Contrôle de version

Créer un compte Stockage Azure et un conteneur de blobs

Dans ce guide de démarrage rapide, vous utilisez Stockage Azure (plus particulièrement, Stockage Blob) comme magasin de points de contrôle. Le marquage de point de contrôle est un processus par lequel un processeur d'événements marque ou valide la position du dernier événement correctement traité dans une partition. Le marquage d'un point de contrôle s'effectue généralement au sein de la fonction qui traite les événements. Pour en savoir plus sur le marquage de point de contrôle, consultez [Processeur d'événements](#).

Suivez les étapes ci-dessous pour créer un compte Stockage Azure.

1. [Création d'un compte de stockage Azure](#)
2. Créer un [conteneur d'objets blob](#)
3. S'authentifier auprès du conteneur d'objets blob

Sans mot de passe (recommandé)

Lors du développement local, assurez-vous que le compte d'utilisateur qui accède aux données blob dispose des autorisations appropriées. Vous aurez besoin du **Contributeur aux données Blob de stockage** pour lire et écrire des données blob. Pour vous attribuer ce rôle, vous aurez besoin du rôle **Administrateur de l'accès utilisateur** ou d'un autre rôle qui inclut l'action **Microsoft.Authorization/roleAssignments/write**. Vous pouvez attribuer des rôles RBAC Azure à un utilisateur à l'aide du Portail Azure, Azure CLI ou Azure PowerShell. Vous pouvez en savoir plus sur les étendues disponibles pour les attributions de rôles dans la page [vue d'ensemble de l'étendue](#).

Dans ce scénario, vous allez attribuer des autorisations à votre compte d'utilisateur, étendues au compte de stockage, pour suivre le [Principe des priviléges minimum](#). Cette pratique offre aux utilisateurs uniquement les autorisations minimales nécessaires et crée des environnements de production plus sécurisés.

L'exemple suivant affecte le rôle **Contributeur aux données Blob du stockage** à votre compte d'utilisateur, qui fournit à la fois un accès en lecture et en écriture aux données d'objet blob dans votre compte de stockage.

Important

Dans la plupart des cas, la propagation de l'attribution de rôle dans Azure peut prendre une ou deux minutes, mais dans de rares cas, cela peut prendre jusqu'à huit minutes. Si vous recevez des erreurs d'authentification lorsque vous exécutez votre code pour la première fois, patientez quelques instants et réessayez.

Azure portal

1. Dans le Portail Azure, recherchez votre compte de stockage à l'aide de la barre de recherche principale ou de la navigation gauche.

2. Dans la page vue d'ensemble du compte de stockage, sélectionnez **Contrôle d'accès (IAM)** dans le menu de gauche.
3. Sur la page **Contrôle d'accès (IAM)**, sélectionnez l'onglet **Attributions de rôles**.
4. Sélectionnez **+ Ajouter** dans le menu supérieur, puis **Ajouter une attribution de rôle** dans le menu déroulant résultant.

The screenshot shows the 'Access Control (IAM)' blade for a storage account named 'identitymigrationstorage'. The left sidebar lists various management options like Overview, Activity log, Tags, and Data storage. The 'Access Control (IAM)' option is highlighted with a red box. The main area has a header with '+ Add', 'Download role assignments', 'Edit columns', 'Refresh', 'Remove', and 'Got feedback?'. A dropdown menu is open at '+ Add', with 'Add role assignment' highlighted and also enclosed in a red box. Below the dropdown are other options: 'Add co-administrator', 'My access' (with a 'View my access' button), 'Check access' (with a 'Find' dropdown set to 'User, group, or service principal' and a radio button selected), and a search bar. To the right, there are two sections: 'Grant access to this resource' (with a 'Add role assignment' button) and 'View deny assignments' (with a 'View' button). Both sections have 'Learn more' links.

5. Utilisez la zone de recherche pour filtrer les résultats sur le rôle souhaité. Pour cet exemple, recherchez *Contributeur aux données Blob du stockage*, sélectionnez le résultat correspondant, puis choisissez **Suivant**.
6. Sous **Attribuer l'accès à**, sélectionnez **Utilisateur, groupe ou principal de service**, puis sélectionnez **+ Sélectionner des membres**.
7. Dans la boîte de dialogue, recherchez votre nom d'utilisateur Microsoft Entra (généralement votre adresse e-mail *utilisateur@domaine*), puis choisissez **Sélectionner** en bas de la boîte de dialogue.
8. Sélectionnez **Vérifier + affecter** pour accéder à la page finale, puis **Vérifier + attribuer** à nouveau pour terminer le processus.

Ajouter des bibliothèques Event Hubs à votre projet Java

Sans mot de passe (recommandé)

Ajoutez les dépendances suivantes dans le fichier pom.xml.

- [azure-messaging-eventhubs](#)
- [azure-messaging-eventhubs-checkpointstore-blob](#)
- [azure-identity](#)

XML

```
<dependencies>
    <dependency>
        <groupId>com.azure</groupId>
        <artifactId>azure-messaging-eventhubs</artifactId>
        <version>5.15.0</version>
    </dependency>
    <dependency>
        <groupId>com.azure</groupId>
        <artifactId>azure-messaging-eventhubs-checkpointstore-
blob</artifactId>
        <version>1.16.1</version>
    </dependency>
    <dependency>
        <groupId>com.azure</groupId>
        <artifactId>azure-identity</artifactId>
        <version>1.8.0</version>
        <scope>compile</scope>
    </dependency>
</dependencies>
```

Sans mot de passe (recommandé)

1. Ajoutez les instructions `import` suivantes au début du fichier Java.

Java

```
import com.azure.messaging.eventhubs.*;
import
com.azure.messaging.eventhubs.checkpointstore.blob.BlobCheckpointSt
ore;
import com.azure.messaging.eventhubs.models.*;
import com.azure.storage.blob.*;
import java.util.function.Consumer;

import com.azure.identity.*;
```

2. Créez une classe nommée `Receiver`, puis ajoutez les variables de chaîne suivantes à la classe. Remplacez les espaces réservés par les valeurs correctes.

 **Important**

Remplacez les espaces réservés par les valeurs correctes.

- <NAMESPACE NAME> par le nom de votre espace de noms Event Hubs.
- <EVENT HUB NAME> par le nom de votre instance Event Hub dans l'espace de noms.

Java

```
private static final String namespaceName = "<NAMESPACE  
NAME>.servicebus.windows.net";  
private static final String eventHubName = "<EVENT HUB NAME>";
```

3. Ajoutez la méthode `main` suivante à la classe.

ⓘ Important

Remplacez les espaces réservés par les valeurs correctes.

- <STORAGE ACCOUNT NAME> par le nom de votre compte de stockage Azure.
- <CONTAINER NAME> par le nom du conteneur d'objets blob dans le compte de stockage.

Java

```
// create a token using the default Azure credential  
DefaultAzureCredential credential = new  
DefaultAzureCredentialBuilder()  
    .authorityHost(AzureAuthorityHosts.AZURE_PUBLIC_CLOUD)  
    .build();  
  
// Create a blob container client that you use later to build an  
// event processor client to receive and process events  
BlobContainerAsyncClient blobContainerAsyncClient = new  
BlobContainerClientBuilder()  
    .credential(credential)  
    .endpoint("https://<STORAGE ACCOUNT  
NAME>.blob.core.windows.net")  
    .containerName("<CONTAINER NAME>")  
    .buildAsyncClient();  
  
// Create an event processor client to receive and process events  
// and errors.  
EventProcessorClient eventProcessorClient = new  
EventProcessorClientBuilder()  
    .fullyQualifiedNamespace(namespaceName)
```

```

.eventHubName(eventHubName)

.consumerGroup(EventHubClientBuilder.DEFAULT_CONSUMER_GROUP_NAME)
    .processEvent(PARTITION_PROCESSOR)
    .processError(ERROR_HANDLER)
    .checkpointStore(new
        BlobCheckpointStore(blobContainerAsyncClient))
        .credential(credential)
        .buildEventProcessorClient();

System.out.println("Starting event processor");
eventProcessorClient.start();

System.out.println("Press enter to stop.");
System.in.read();

System.out.println("Stopping event processor");
eventProcessorClient.stop();
System.out.println("Event processor stopped.");

System.out.println("Exiting process");

```

4. Ajoutez les deux méthodes d'assistance (PARTITION_PROCESSOR et ERROR_HANDLER) qui traitent les événements et les erreurs à la classe Receiver.

Java

```

public static final Consumer<EventContext> PARTITION_PROCESSOR =
eventContext -> {
    PartitionContext partitionContext =
eventContext.getPartitionContext();
    EventData eventData = eventContext.getEventData();

    System.out.printf("Processing event from partition %s with sequence
number %d with body: %s%n",
        partitionContext.getPartitionId(),
        eventData.getSequenceNumber(), eventData.getBodyAsString());

    // Every 10 events received, it will update the checkpoint stored
    // in Azure Blob Storage.
    if (eventData.getSequenceNumber() % 10 == 0) {
        eventContext.updateCheckpoint();
    }
};

public static final Consumer<ErrorContext> ERROR_HANDLER = errorContext
-> {
    System.out.printf("Error occurred in partition processor for
partition %s, %s.%n",
        errorContext.getPartitionContext().getPartitionId(),

```

```
        errorContext.getThrowable());  
    };
```

5. Générez le programme et assurez-vous qu'il n'y a aucune erreur.

Exécution des applications

1. Exécutez d'abord l'application **réceptrice**.
2. Ensuite, exécutez l'application **émettrice**.
3. Dans la fenêtre de l'application **réceptrice**, vérifiez que vous voyez les événements qui ont été publiés par l'application émettrice.

```
Invite de commandes Windows  
  
Starting event processor  
Press enter to stop.  
Processing event from partition 0 with sequence number 331 with body:  
Foo  
Processing event from partition 0 with sequence number 332 with body:  
Bar
```

4. Appuyez sur **Entrée** dans la fenêtre d'application réceptrice pour arrêter l'application.

```
Invite de commandes Windows  
  
Starting event processor  
Press enter to stop.  
Processing event from partition 0 with sequence number 331 with body:  
Foo  
Processing event from partition 0 with sequence number 332 with body:  
Bar  
  
Stopping event processor  
Event processor stopped.  
Exiting process
```

Étapes suivantes

Consultez les exemples suivants sur GitHub :

- [Exemples azure-messaging-eventhubs ↗](#)
- [Exemples azure-messaging-eventhubs-checkpointstore-blob ↗](#)

Démarrage rapide : diffuser des données avec Azure Event Hubs et Apache Kafka

Article • 09/08/2023

Ce guide de démarrage rapide vous montre comment diffuser en continu des données vers et à partir d'Azure Event Hubs à l'aide du protocole Apache Kafka. Vous ne modifierez aucun code dans les exemples d'applications de producteur ou de consommateur Kafka. Il vous suffit de mettre à jour les configurations que les clients utilisent pour pointer vers un espace de noms Event Hubs, qui expose un point de terminaison Kafka. Vous ne générez pas et n'utilisez pas de cluster Kafka par vous-même. Au lieu de cela, vous utilisez l'espace de noms Event Hubs avec le point de terminaison Kafka.

ⓘ Notes

Cet exemple est disponible sur [GitHub ↗](#).

Prérequis

Pour suivre ce démarrage rapide, vérifiez que vous avez :

- Lisez l'article [Event Hubs pour Apache Kafka](#).
- Un abonnement Azure. Si vous n'en avez pas, créez un [compte gratuit ↗](#) avant de commencer.
- Créez une machine virtuelle Windows et installez les composants suivants :
 - [Java Development Kit \(JDK\) 1.7+](#).
 - [Téléchargez ↗](#) et [installez ↗](#) une archive binaire Maven.
 - [Git ↗](#)

Créer un espace de noms Azure Event Hubs

Quand vous créez un espace de noms Event Hubs, le point de terminaison Kafka pour l'espace de noms est automatiquement activé. Vous pouvez diffuser des événements à partir de vos applications qui utilisent le protocole Kafka dans Event Hubs. Suivez les instructions pas à pas de l'article [Créer un hub d'événements avec le portail Azure](#) pour

créer un espace de noms Event Hubs. Si vous utilisez un cluster dédié, consultez [Créer un Event Hub et un espace de noms](#).

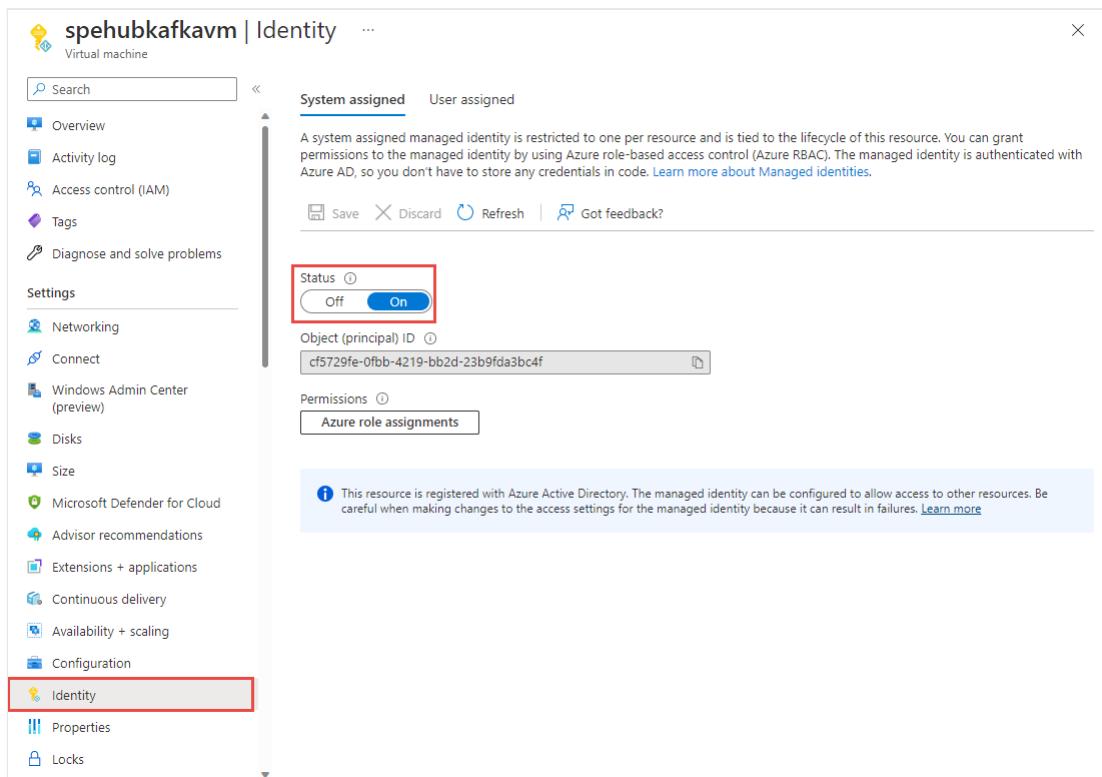
ⓘ Notes

Event Hubs pour Kafka n'est pas pris en charge dans le niveau **De base**.

Envoyer et recevoir des messages avec Kafka dans Event Hubs

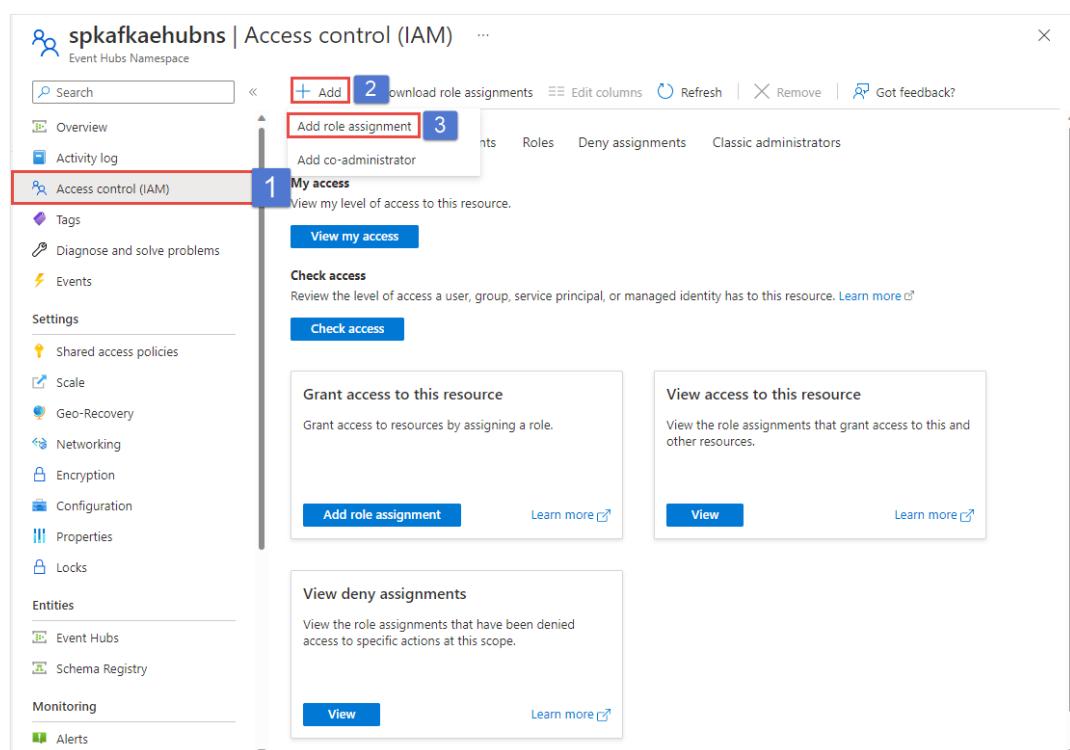
Sans mot de passe (recommandé)

1. Activez une identité managée affectée par le système pour votre machine virtuelle. Pour plus d'informations sur la configuration d'une identité managée sur une machine virtuelle, consultez [Configurer des identités managées pour les ressources Azure sur une machine virtuelle en utilisant le Portail Azure](#). Les identités managées pour les ressources Azure fournissent aux services Azure une identité automatiquement gérée dans Microsoft Entra ID. Vous pouvez utiliser cette identité pour vous authentifier auprès de n'importe quel service prenant en charge l'authentification Microsoft Entra, sans avoir d'informations d'identification dans votre code.



2. À l'aide de la page **Contrôle d'accès** de l'espace de noms Event Hubs que vous avez créé, attribuez le rôle **Propriétaire des données Azure Event Hubs** à l'identité managée de la machine virtuelle. Azure Event Hubs prend en charge l'utilisation de Microsoft Entra ID pour autoriser les requêtes de ressources Event Hubs. Microsoft Entra ID vous permet d'utiliser le contrôle d'accès en fonction du rôle (RBAC) Azure pour accorder des autorisations à un principal de sécurité, qui peut être un utilisateur ou un principal de service d'application.

- a. Dans le portail Azure, accédez à votre espace de noms Event Hubs. Accédez à « Contrôle d'accès (IAM) » dans le volet de navigation de gauche.
- b. Sélectionnez + Ajouter, puis sélectionnez `Add role assignment`.



- c. Sous l'onglet Rôle, sélectionnez **Propriétaire des données Azure Event Hubs**, puis sélectionnez le bouton **Suivant**.

Add role assignment ...

Got feedback?

Role Members Review + assign

A role definition is a collection of permissions. You can use the built-in roles or you can create your own custom roles. [Learn more](#)

Name ↑↓	Description ↑↓	Type ↑↓	Category ↑↓	Details
Owner	Grants full access to manage all resources, including th...	BuiltinRole	General	View
Contributor	Grants full access to manage all resources, but does no...	BuiltinRole	General	View
Reader	View all resources, but does not allow you to make any...	BuiltinRole	General	View
Azure Event Hubs Data Owner	Allows for full access to Azure Event Hubs resources.	BuiltinRole	Analytics	View
Azure Event Hubs Data Receiver	Allows receive access to Azure Event Hubs resources.	BuiltinRole	Analytics	View
Azure Event Hubs Data Sender	Allows send access to Azure Event Hubs resources.	BuiltinRole	Analytics	View
Log Analytics Contributor	Log Analytics Contributor can read all monitoring data ...	BuiltinRole	Analytics	View
Log Analytics Reader	Log Analytics Reader can view and search all monitorin...	BuiltinRole	Analytics	View
Managed Application Contributor Role	Allows for creating managed application resources.	BuiltinRole	Management + Gover...	View
Managed Application Operator Role	Lets you read and perform actions on Managed Applic...	BuiltinRole	Management + Gover...	View
Managed Applications Reader	Lets you read resources in a managed app and request...	BuiltinRole	Management + Gover...	View
Monitoring Contributor	Can read all monitoring data and update monitoring se...	BuiltinRole	Monitor	View
Monitoring Metrics Publisher	Enables publishing metrics against Azure resources	BuiltinRole	Monitor	View

Review + assign Previous **Next**

d. Sous l'onglet **Membres**, sélectionnez l'**identité managée** dans la section **Affecter l'accès à**.

e. Sélectionnez le lien **+Sélectionner les membres**.

f. Dans la page **Sélectionner des identités managées**, procédez comme suit :

i. Sélectionnez l'**abonnement Azure** qui a la machine virtuelle.

ii. Pour **Identité managée**, sélectionnez **Machine virtuelle**.

iii. Sélectionnez l'identité managée de votre machine virtuelle.

iv. Sélectionnez **Sélectionner** au bas de la page.

g. Sélectionnez Vérifier + attribuer.

3. Redémarrez la machine virtuelle et reconnectez-vous à la machine virtuelle pour laquelle vous avez configuré l'identité managée.

4. Clonez le [dépôt Azure Event Hubs pour Kafka](#).
5. Accédez à `azure-event-hubs-for-kafka/tutorials/oauth/java/managedidentity/consumer`.
6. Basculez vers le dossier `src/main/resources/`, puis ouvrez `consumer.config`. Remplacez `namespacename` par le nom de votre espace de noms Event Hub.

XML

```
bootstrap.servers=NAMESPACENAME.servicebus.windows.net:9093
security.protocol=SASL_SSL
sasl.mechanism=OAUTHBEARER
sasl.jaas.config=org.apache.kafka.common.security.oauthbearer.OAuth
BearerLoginModule required;
sasl.login.callback.handler.class=CustomAuthenticateCallbackHandler
;
```

ⓘ Notes

Vous trouverez [ici](#) tous les exemples OAuth applicables à Event Hubs pour Kafka.

7. Revenez au dossier **Consommateur** dans lequel se trouve le fichier pom.xml et exécutez le code consommateur et traitez les événements à partir du Event Hub à l'aide de vos clients Kafka :

Java

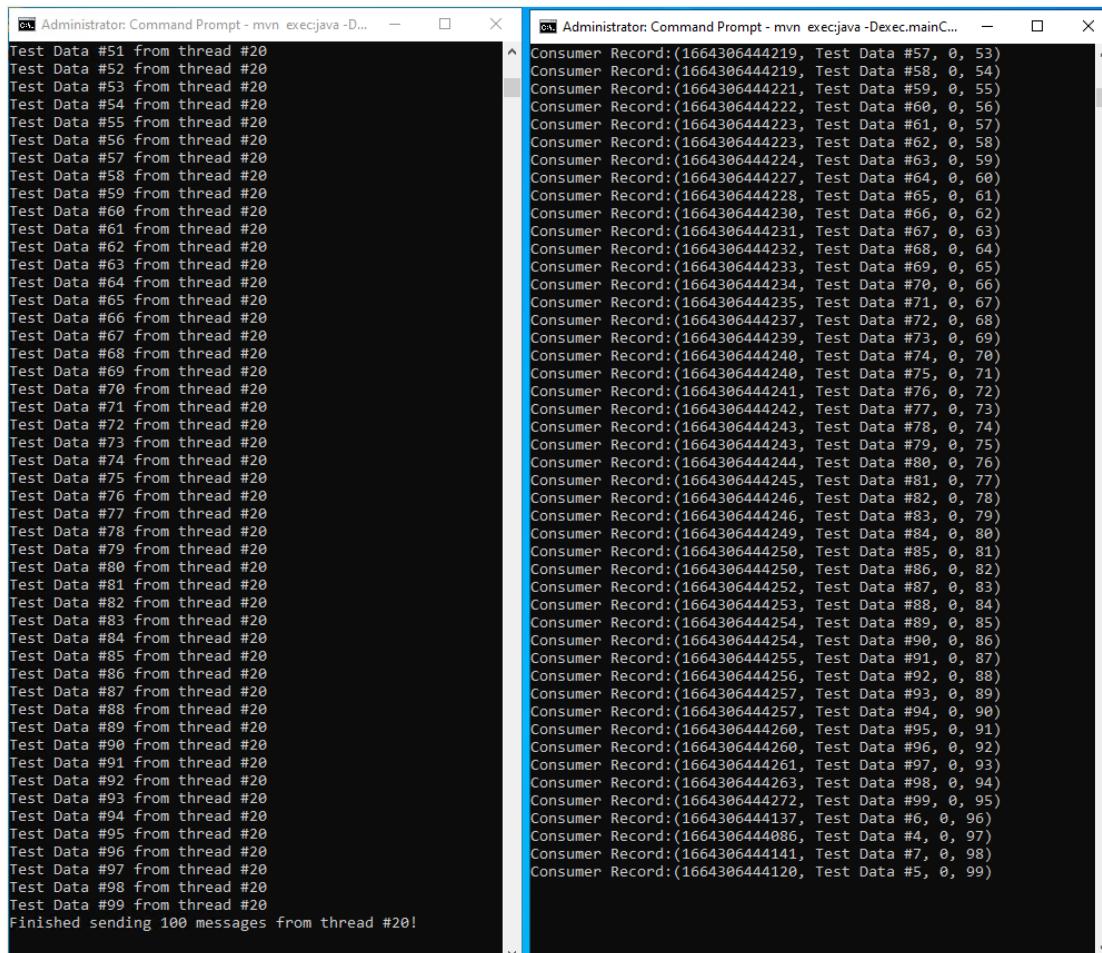
```
mvn clean package
mvn exec:java -Dexec.mainClass="TestConsumer"
```

8. Lancez une autre fenêtre d'invite de commandes et accédez à `azure-event-hubs-for-kafka/tutorials/oauth/java/managedidentity/producer`.
9. Basculez vers le dossier `src/main/resources/`, puis ouvrez `producer.config`. Remplacez `mynamespace` par le nom de votre espace de noms Event Hub.
10. Revenez au dossier **Producteur** où se trouve le fichier `pom.xml` et exécutez le code du producteur et diffusez des événements dans Event Hubs :

shell

```
mvn clean package  
mvn exec:java -Dexec.mainClass="TestProducer"
```

Vous devez voir des messages sur les événements envoyés dans la fenêtre Producteur. À présent, vérifiez la fenêtre de l'application consommateur pour afficher les messages qu'elle reçoit à partir du Event Hub.



The image shows two side-by-side Command Prompt windows. The left window, titled 'Administrator: Command Prompt - mvn exec:java -Dexec.mainC...', displays the output of the producer application. It shows 100 test messages being sent from thread #20, ranging from Test Data #51 to #100. The right window, also titled 'Administrator: Command Prompt - mvn exec:java -Dexec.mainC...', displays the output of the consumer application. It shows 100 consumer records, each consisting of a timestamp (e.g., 1664306444219) and a test data value (e.g., #57, #58, ..., #99). Both windows have standard Windows-style title bars and scrollbars.

```
Administrator: Command Prompt - mvn exec:java -Dexec.mainC...  
Test Data #51 from thread #20  
Test Data #52 from thread #20  
Test Data #53 from thread #20  
Test Data #54 from thread #20  
Test Data #55 from thread #20  
Test Data #56 from thread #20  
Test Data #57 from thread #20  
Test Data #58 from thread #20  
Test Data #59 from thread #20  
Test Data #60 from thread #20  
Test Data #61 from thread #20  
Test Data #62 from thread #20  
Test Data #63 from thread #20  
Test Data #64 from thread #20  
Test Data #65 from thread #20  
Test Data #66 from thread #20  
Test Data #67 from thread #20  
Test Data #68 from thread #20  
Test Data #69 from thread #20  
Test Data #70 from thread #20  
Test Data #71 from thread #20  
Test Data #72 from thread #20  
Test Data #73 from thread #20  
Test Data #74 from thread #20  
Test Data #75 from thread #20  
Test Data #76 from thread #20  
Test Data #77 from thread #20  
Test Data #78 from thread #20  
Test Data #79 from thread #20  
Test Data #80 from thread #20  
Test Data #81 from thread #20  
Test Data #82 from thread #20  
Test Data #83 from thread #20  
Test Data #84 from thread #20  
Test Data #85 from thread #20  
Test Data #86 from thread #20  
Test Data #87 from thread #20  
Test Data #88 from thread #20  
Test Data #89 from thread #20  
Test Data #90 from thread #20  
Test Data #91 from thread #20  
Test Data #92 from thread #20  
Test Data #93 from thread #20  
Test Data #94 from thread #20  
Test Data #95 from thread #20  
Test Data #96 from thread #20  
Test Data #97 from thread #20  
Test Data #98 from thread #20  
Test Data #99 from thread #20  
Finished sending 100 messages from thread #20!  
  
Administrator: Command Prompt - mvn exec:java -Dexec.mainC...  
Consumer Record:(1664306444219, Test Data #57, 0, 53)  
Consumer Record:(1664306444219, Test Data #58, 0, 54)  
Consumer Record:(1664306444221, Test Data #59, 0, 55)  
Consumer Record:(1664306444222, Test Data #60, 0, 56)  
Consumer Record:(1664306444223, Test Data #61, 0, 57)  
Consumer Record:(1664306444223, Test Data #62, 0, 58)  
Consumer Record:(1664306444224, Test Data #63, 0, 59)  
Consumer Record:(1664306444227, Test Data #64, 0, 60)  
Consumer Record:(1664306444228, Test Data #65, 0, 61)  
Consumer Record:(1664306444230, Test Data #66, 0, 62)  
Consumer Record:(1664306444231, Test Data #67, 0, 63)  
Consumer Record:(1664306444232, Test Data #68, 0, 64)  
Consumer Record:(1664306444233, Test Data #69, 0, 65)  
Consumer Record:(1664306444234, Test Data #70, 0, 66)  
Consumer Record:(1664306444235, Test Data #71, 0, 67)  
Consumer Record:(1664306444237, Test Data #72, 0, 68)  
Consumer Record:(1664306444239, Test Data #73, 0, 69)  
Consumer Record:(1664306444240, Test Data #74, 0, 70)  
Consumer Record:(1664306444240, Test Data #75, 0, 71)  
Consumer Record:(1664306444241, Test Data #76, 0, 72)  
Consumer Record:(1664306444242, Test Data #77, 0, 73)  
Consumer Record:(1664306444243, Test Data #78, 0, 74)  
Consumer Record:(1664306444243, Test Data #79, 0, 75)  
Consumer Record:(1664306444244, Test Data #80, 0, 76)  
Consumer Record:(1664306444245, Test Data #81, 0, 77)  
Consumer Record:(1664306444246, Test Data #82, 0, 78)  
Consumer Record:(1664306444246, Test Data #83, 0, 79)  
Consumer Record:(1664306444249, Test Data #84, 0, 80)  
Consumer Record:(1664306444250, Test Data #85, 0, 81)  
Consumer Record:(1664306444250, Test Data #86, 0, 82)  
Consumer Record:(1664306444252, Test Data #87, 0, 83)  
Consumer Record:(1664306444253, Test Data #88, 0, 84)  
Consumer Record:(1664306444254, Test Data #89, 0, 85)  
Consumer Record:(1664306444254, Test Data #90, 0, 86)  
Consumer Record:(1664306444255, Test Data #91, 0, 87)  
Consumer Record:(1664306444256, Test Data #92, 0, 88)  
Consumer Record:(1664306444257, Test Data #93, 0, 89)  
Consumer Record:(1664306444257, Test Data #94, 0, 90)  
Consumer Record:(1664306444260, Test Data #95, 0, 91)  
Consumer Record:(1664306444260, Test Data #96, 0, 92)  
Consumer Record:(1664306444261, Test Data #97, 0, 93)  
Consumer Record:(1664306444263, Test Data #98, 0, 94)  
Consumer Record:(1664306444272, Test Data #99, 0, 95)  
Consumer Record:(1664306444137, Test Data #0, 0, 96)  
Consumer Record:(1664306444086, Test Data #4, 0, 97)  
Consumer Record:(1664306444141, Test Data #7, 0, 98)  
Consumer Record:(1664306444120, Test Data #5, 0, 99)
```

Validation de schéma pour Kafka avec Schema Registry

Vous pouvez utiliser Azure Schema Registry pour effectuer la validation de schémas lorsque vous diffusez des données avec vos applications Kafka en utilisant Event Hubs. Azure Schema Registry d'Event Hubs fournit un référentiel centralisé pour la gestion des schémas et vous pouvez connecter en toute transparence vos applications Kafka nouvelles ou existantes à Schema Registry.

Si vous souhaitez obtenir plus d'informations, consultez [Valider les schémas pour les applications Apache Kafka avec Avro](#).

Étapes suivantes

Dans cet article, vous avez appris à diffuser en streaming dans Event Hubs sans changer vos clients de protocole ni exécuter vos propres clusters. Pour en savoir plus, consultez [Guide du développeur Apache Kafka pour Azure Event Hubs](#).

Démarrage rapide : Bibliothèque de client de certificats Azure Key Vault pour Java (certificats)

Article • 25/03/2023

Découvrez comment démarrer avec la bibliothèque de client de certificats Azure Key Vault pour Java. Suivez les étapes ci-dessous pour installer le package et tester un exemple de code relatif à des tâches de base.

💡 Conseil

Si vous utilisez des ressources de certificats Azure Key Vault dans une application Spring, nous vous recommandons de considérer [Azure Spring Cloud](#) comme alternative. Azure Spring Cloud est un projet open source qui fournit une intégration de Spring fluide aux services Azure. Pour en savoir plus sur Spring Cloud Azure et pour voir un exemple d'utilisation de certificats Key Vault, consultez [Activer HTTPS dans Spring Boot avec des certificats Azure Key Vault](#).

Ressources supplémentaires :

- [Code source ↗](#)
- [Documentation de référence de l'API ↗](#)
- [Documentation du produit](#)
- [Exemples ↗](#)

Prérequis

- Un abonnement Azure - [En créer un gratuitement ↗](#)
- [Kit de développement Java \(JDK\)](#), version 8 ou ultérieure
- [Apache Maven ↗](#)
- [Azure CLI](#)

Ce guide de démarrage rapide suppose que vous exécutez [Azure CLI](#) et [Apache Maven ↗](#) dans une fenêtre de terminal Linux.

Configuration

Ce guide de démarrage rapide utilise la bibliothèque Azure Identity avec Azure CLI pour authentifier l'utilisateur auprès des services Azure. Les développeurs peuvent également utiliser Visual Studio ou Visual Studio Code pour authentifier leurs appels. Pour plus d'informations, consultez [Authentifier le client avec la bibliothèque de client Azure Identity](#).

Connexion à Azure

1. Exécutez la commande `login`.

```
Azure CLI  
az login
```

Si l'interface CLI peut ouvrir votre navigateur par défaut, elle le fait et charge une page de connexion Azure par la même occasion.

Sinon, ouvrez une page de navigateur à l'adresse <https://aka.ms/devicelogin> et entrez le code d'autorisation affiché dans votre terminal.

2. Dans le navigateur, connectez-vous avec les informations d'identification de votre compte.

Créer une application console Java

Dans une fenêtre de console, utilisez la commande `mvn` pour créer une application console Java nommée `akv-certificates-java`.

```
Console  
  
mvn archetype:generate -DgroupId=com.keyvault.certificates.quickstart  
-DartifactId=akv-certificates-java  
-DarchetypeArtifactId=maven-archetype-quickstart  
-DarchetypeVersion=1.4  
-DinteractiveMode=false
```

Le résultat de la génération du projet doit ressembler à ceci :

```
Console  
  
[INFO] -----  
-----  
[INFO] Using following parameters for creating project from Archetype:  
maven-archetype-quickstart:1.4
```

```
[INFO] -----
[INFO] Parameter: groupId, Value: com.keyvault.certificates.quickstart
[INFO] Parameter: artifactId, Value: akv-certificates-java
[INFO] Parameter: version, Value: 1.0-SNAPSHOT
[INFO] Parameter: package, Value: com.keyvault.certificates.quickstart
[INFO] Parameter: packageInPathFormat, Value: com/keyvault/quickstart
[INFO] Parameter: package, Value: com.keyvault.certificates.quickstart
[INFO] Parameter: groupId, Value: com.keyvault.certificates.quickstart
[INFO] Parameter: artifactId, Value: akv-certificates-java
[INFO] Parameter: version, Value: 1.0-SNAPSHOT
[INFO] Project created from Archetype in dir: /home/user/quickstarts/akv-
certificates-java
[INFO] -----
---
[INFO] BUILD SUCCESS
[INFO] -----
---
[INFO] Total time:  38.124 s
[INFO] Finished at: 2019-11-15T13:19:06-08:00
[INFO] -----
---
```

Changez de répertoire pour le dossier `akv-certificates-java/` créé.

Console

```
cd akv-certificates-java
```

Installer le package

Ouvrez le fichier `pom.xml` dans votre éditeur de texte. Ajoutez les éléments de dépendance suivants au groupe de dépendances.

XML

```
<dependency>
    <groupId>com.azure</groupId>
    <artifactId>azure-security-keyvault-certificates</artifactId>
    <version>4.1.3</version>
</dependency>

<dependency>
    <groupId>com.azure</groupId>
    <artifactId>azure-identity</artifactId>
    <version>1.2.0</version>
</dependency>
```

Créer un groupe de ressources et un coffre de clés

Ce guide de démarrage rapide utilise un coffre de clés Azure créé au préalable. Vous pouvez créer un coffre de clés en suivant les étapes décrites dans le [guide de démarrage rapide d'Azure CLI](#), le [guide de démarrage rapide d'Azure PowerShell](#) ou le [guide de démarrage rapide du portail Azure](#).

Sinon, vous pouvez simplement exécuter les commandes Azure CLI ou Azure PowerShell ci-dessous.

ⓘ Important

Chaque coffre de clés doit avoir un nom unique. Remplacez <your-unique-keyvault-name> par le nom de votre coffre de clés dans les exemples suivants.

Azure CLI

```
Azure CLI  
  
az group create --name "myResourceGroup" -l "EastUS"  
  
az keyvault create --name "<your-unique-keyvault-name>" -g  
"myResourceGroup"
```

Accorder l'accès à votre coffre de clés

Créez une stratégie d'accès pour votre coffre de clés, qui accorde des autorisations de certificat à votre compte d'utilisateur.

Azure CLI

```
az keyvault set-policy --name <your-key-vault-name> --upn user@domain.com --  
certificate-permissions delete get list create purge
```

Définir des variables d'environnement

Cette application utilise le nom de votre coffre de clés en tant que variable d'environnement appelée `KEY_VAULT_NAME`.

Windows

Invite de commandes Windows

```
set KEY_VAULT_NAME=<your-key-vault-name>
```

Windows PowerShell

PowerShell

```
$Env:KEY_VAULT_NAME="your-key-vault-name"
```

macOS ou Linux

Invite de commandes Windows

```
export KEY_VAULT_NAME=<your-key-vault-name>
```

Modèle objet

La bibliothèque de client de certificats Azure Key Vault pour Java vous permet de gérer les certificats. La section [Exemples de code](#) montre comment créer un client et comment créer, récupérer et supprimer un certificat.

L'application console complète est [ci-dessous](#).

Exemples de code

Ajouter des directives

Ajoutez les directives suivantes en haut de votre code :

Java

```
import com.azure.core.util.polling.SyncPoller;
import com.azure.identity.DefaultAzureCredentialBuilder;

import com.azure.security.keyvault.certificates.CertificateClient;
import com.azure.security.keyvault.certificates.CertificateClientBuilder;
import com.azure.security.keyvault.certificates.models.CertificateOperation;
import com.azure.security.keyvault.certificates.models.CertificatePolicy;
import com.azure.security.keyvault.certificates.models.DeletedCertificate;
import com.azure.security.keyvault.certificates.models.KeyVaultCertificate;
import
```

```
com.azure.security.keyvault.certificates.models.KeyVaultCertificateWithPolicy;
```

Authentifier et créer un client

Les requêtes d'application vers les Services Azure doivent être autorisées. L'utilisation de [DefaultAzureCredential](#) est l'approche recommandée pour implémenter des connexions sans mot de passe aux services Azure dans votre code. `DefaultAzureCredential` prend en charge plusieurs méthodes d'authentification et détermine quelle méthode doit être utilisée au moment de l'exécution. Cette approche permet à votre application d'utiliser différentes méthodes d'authentification dans différents environnements (local ou production) sans implémenter de code spécifique à l'environnement.

Dans ce guide de démarrage rapide, `DefaultAzureCredential` s'authentifie auprès du coffre de clés à l'aide des informations d'identification de l'utilisateur de développement local connecté à Azure CLI. Quand l'application est déployée sur Azure, le même code `DefaultAzureCredential` peut découvrir et utiliser automatiquement une identité managée affectée à un service d'application, une machine virtuelle ou d'autres services. Pour plus d'informations, consultez [Vue d'ensemble des identités managées](#).

Dans cet exemple, le nom de votre coffre de clés est étendu à l'URI du coffre de clés, au format `https://<your-key-vault-name>.vault.azure.net`. Pour plus d'informations sur l'authentification auprès du coffre de clés, consultez le [Guide du développeur](#).

Java

```
String keyVaultName = System.getenv("KEY_VAULT_NAME");
String keyVaultUri = "https://" + keyVaultName + ".vault.azure.net";

CertificateClient certificateClient = new CertificateClientBuilder()
    .vaultUrl(keyVaultUri)
    .credential(new DefaultAzureCredentialBuilder().build())
    .buildClient();
```

Enregistrer un secret

Votre application étant authentifiée, vous pouvez créer un certificat dans votre coffre de clés en utilisant la méthode `certificateClient.beginCreateCertificate`. Cette opération nécessite un nom pour le certificat et une stratégie de certificat : nous avons affecté la valeur « myCertificate » à la variable `certificateName` dans cet exemple et utilisons une stratégie par défaut.

La création du certificat est une opération longue, dont vous pouvez interroger la progression ou attendre la fin.

Java

```
SyncPoller<CertificateOperation, KeyVaultCertificateWithPolicy>
certificatePoller =
    certificateClient.beginCreateCertificate(certificateName,
CertificatePolicy.getDefault());
certificatePoller.waitForCompletion();
```

Une fois la création terminée, vous pouvez obtenir le certificat par le biais de l'appel suivant :

Java

```
KeyVaultCertificate createdCertificate = certificatePoller.getFinalResult();
```

Récupérer un certificat

Vous pouvez maintenant récupérer le certificat créé précédemment à l'aide de la méthode `certificateClient.getCertificate`.

Java

```
KeyVaultCertificate retrievedCertificate =
certificateClient.getCertificate(certificateName);
```

Vous pouvez maintenant accéder aux détails du certificat récupéré à l'aide d'opérations telles que `retrievedCertificate.getName`, `retrievedCertificate.getProperties`, etc. Ainsi qu'à son contenu (`retrievedCertificate.getCer`).

Supprimer un certificat

Enfin, nous allons supprimer le certificat de votre coffre de clés avec la méthode `certificateClient.beginDeleteCertificate`, qui est également une opération longue.

Java

```
SyncPoller<DeletedCertificate, Void> deletionPoller =
certificateClient.beginDeleteCertificate(certificateName);
deletionPoller.waitForCompletion();
```

Nettoyer les ressources

Une fois que vous n'en avez plus besoin, vous pouvez supprimer votre coffre de clés et le groupe de ressources correspondant en utilisant Azure CLI ou Azure PowerShell.

Azure CLI

```
az group delete -g "myResourceGroup"
```

Azure PowerShell

```
Remove-AzResourceGroup -Name "myResourceGroup"
```

Exemple de code

Java

```
package com.keyvault.certificates.quickstart;

import com.azure.core.util.polling.SyncPoller;
import com.azure.identity.DefaultAzureCredentialBuilder;

import com.azure.security.keyvault.certificates.CertificateClient;
import com.azure.security.keyvault.certificates.CertificateClientBuilder;
import com.azure.security.keyvault.certificates.models.CertificateOperation;
import com.azure.security.keyvault.certificates.models.CertificatePolicy;
import com.azure.security.keyvault.certificates.models.DeletedCertificate;
import com.azure.security.keyvault.certificates.models.KeyVaultCertificate;
import com.azure.security.keyvault.certificates.models.KeyVaultCertificateWithPolicy;

public class App {
    public static void main(String[] args) throws InterruptedException,
IllegalArgumentException {
        String keyVaultName = System.getenv("KEY_VAULT_NAME");
        String keyVaultUri = "https://" + keyVaultName + ".vault.azure.net";

        System.out.printf("key vault name = %s and kv uri = %s \n",
keyVaultName, keyVaultUri);

        CertificateClient certificateClient = new CertificateClientBuilder()
            .vaultUrl(keyVaultUri)
            .credential(new DefaultAzureCredentialBuilder().build())
            .buildClient();

        String certificateName = "myCertificate";
```

```

        System.out.print("Creating a certificate in " + keyVaultName + " "
    called "" + certificateName + " ... ");

        SyncPoller<CertificateOperation, KeyVaultCertificateWithPolicy>
certificatePoller =
            certificateClient.beginCreateCertificate(certificateName,
CertificatePolicy.getDefault());
        certificatePoller.waitForCompletion();

        System.out.print("done.");
        System.out.println("Retrieving certificate from " + keyVaultName +
".");
    }

    KeyVaultCertificate retrievedCertificate =
certificateClient.getCertificate(certificateName);

    System.out.println("Your certificate's ID is '" +
retrievedCertificate.getId() + "'.");

    System.out.println("Deleting your certificate from " + keyVaultName
+ " ... ");

    SyncPoller<DeletedCertificate, Void> deletionPoller =
certificateClient.beginDeleteCertificate(certificateName);
    deletionPoller.waitForCompletion();

    System.out.print("done.");
}
}

```

Étapes suivantes

Dans ce guide de démarrage rapide, vous avez créé un coffre de clés et créé, récupéré, puis supprimé un certificat. Pour en savoir plus sur Key Vault et sur la manière de l'intégrer à vos applications, consultez les articles ci-dessous.

- Lire la [vue d'ensemble Azure Key Vault](#)
- Consulter le [Guide du développeur Azure Key Vault](#)
- Découvrir comment [Sécuriser l'accès à un coffre de clés](#)

Démarrage rapide : Bibliothèque de client de clés Azure Key Vault pour Java

Article • 01/06/2023

Découvrez comment démarrer avec la bibliothèque de client de clés Azure Key Vault pour Java. Suivez les étapes suivantes pour installer le package et essayer un exemple de code pour les tâches de base.

Ressources supplémentaires :

- [Code source ↗](#)
- [Documentation de référence de l'API ↗](#)
- [Documentation du produit](#)
- [Exemples ↗](#)

Prérequis

- Un abonnement Azure - [En créer un gratuitement↗](#)
- [Kit de développement Java \(JDK\)](#), version 8 ou ultérieure
- [Apache Maven ↗](#)
- [Azure CLI](#)

Ce guide de démarrage rapide suppose que vous exécutez l'interface [Azure CLI](#) et [Apache Maven ↗](#) dans une fenêtre de terminal Linux.

Configuration

Ce guide de démarrage rapide utilise la bibliothèque Azure Identity avec Azure CLI pour authentifier l'utilisateur auprès des services Azure. Les développeurs peuvent également utiliser Visual Studio ou Visual Studio Code pour authentifier leurs appels. Pour plus d'informations, consultez [Authentifier le client avec la bibliothèque de client Azure Identity](#).

Connexion à Azure

1. Exécutez la commande `login`.

Azure CLI

```
az login
```

Si l'interface CLI peut ouvrir votre navigateur par défaut, elle le fait et charge une page de connexion Azure par la même occasion.

Sinon, ouvrez une page de navigateur à l'adresse <https://aka.ms/devicelogin> et entrez le code d'autorisation affiché dans votre terminal.

2. Dans le navigateur, connectez-vous avec les informations d'identification de votre compte.

Créer une application console Java

Dans une fenêtre de console, utilisez la commande `mvn` pour créer une application console Java nommée `akv-keys-java`.

Console

```
mvn archetype:generate -DgroupId=com.keyvault.keys.quickstart
-DartifactId=akv-keys-java
-DarchetypeArtifactId=maven-archetype-quickstart
-DarchetypeVersion=1.4
-DinteractiveMode=false
```

Le résultat de la génération du projet doit ressembler à ceci :

Console

```
[INFO] -----
[INFO] Using following parameters for creating project from Archetype:
maven-archetype-quickstart:1.4
[INFO] -----
[INFO] Parameter: groupId, Value: com.keyvault.keys.quickstart
[INFO] Parameter: artifactId, Value: akv-keys-java
[INFO] Parameter: version, Value: 1.0-SNAPSHOT
[INFO] Parameter: package, Value: com.keyvault.keys.quickstart
[INFO] Parameter: packageInPathFormat, Value: com/keyvault/quickstart
[INFO] Parameter: package, Value: com.keyvault.keys.quickstart
[INFO] Parameter: groupId, Value: com.keyvault.keys.quickstart
[INFO] Parameter: artifactId, Value: akv-keys-java
[INFO] Parameter: version, Value: 1.0-SNAPSHOT
[INFO] Project created from Archetype in dir: /home/user/quickstarts/akv-
keys-java
[INFO] -----
---
```

```
[INFO] BUILD SUCCESS
[INFO] -----
---
[INFO] Total time:  38.124 s
[INFO] Finished at: 2019-11-15T13:19:06-08:00
[INFO] -----
---
```

Changez de répertoire pour le dossier `akv-keys-java/` créé.

Console

```
cd akv-keys-java
```

Installer le package

Ouvrez le fichier `pom.xml` dans votre éditeur de texte. Ajoutez les éléments de dépendance suivants au groupe de dépendances.

XML

```
<dependency>
    <groupId>com.azure</groupId>
    <artifactId>azure-security-keyvault-keys</artifactId>
    <version>4.2.3</version>
</dependency>

<dependency>
    <groupId>com.azure</groupId>
    <artifactId>azure-identity</artifactId>
    <version>1.2.0</version>
</dependency>
```

Créer un groupe de ressources et un coffre de clés

Ce guide de démarrage rapide utilise un coffre de clés Azure créé au préalable. Vous pouvez créer un coffre de clés en suivant les étapes décrites dans le [guide de démarrage rapide d'Azure CLI](#), le [guide de démarrage rapide d'Azure PowerShell](#) ou le [guide de démarrage rapide du portail Azure](#).

Sinon, vous pouvez simplement exécuter les commandes Azure CLI ou Azure PowerShell ci-dessous.

 **Important**

Chaque coffre de clés doit avoir un nom unique. Remplacez <your-unique-keyvault-name> par le nom de votre coffre de clés dans les exemples suivants.

Azure CLI

Azure CLI

```
az group create --name "myResourceGroup" -l "EastUS"  
az keyvault create --name "<your-unique-keyvault-name>" -g  
"myResourceGroup"
```

Accorder l'accès à votre coffre de clés

Créez une stratégie d'accès pour votre coffre de clés, qui accorde des autorisations de clé à votre compte d'utilisateur.

Azure CLI

```
az keyvault set-policy --name <your-key-vault-name> --upn user@domain.com --  
key-permissions delete get list create purge
```

Définir des variables d'environnement

Cette application utilise le nom de votre coffre de clés en tant que variable d'environnement appelée `KEY_VAULT_NAME`.

Windows

Invite de commandes Windows

```
set KEY_VAULT_NAME=<your-key-vault-name>
```

Windows PowerShell

PowerShell

```
$Env:KEY_VAULT_NAME="<your-key-vault-name>"
```

macOS ou Linux

Invite de commandes Windows

```
export KEY_VAULT_NAME=<your-key-vault-name>
```

Modèle objet

La bibliothèque de client de clés Azure Key Vault pour Java vous permet de gérer des clés. La section [Exemples de code](#) montre comment créer un client et créer, récupérer et supprimer une clé.

L'application console entière est fournie dans [l'exemple de code](#).

Exemples de code

Ajouter des directives

Ajoutez les directives suivantes en haut de votre code :

Java

```
import com.azure.core.util.polling.SyncPoller;
import com.azure.identity.DefaultAzureCredentialBuilder;

import com.azure.security.keyvault.keys.KeyClient;
import com.azure.security.keyvault.keys.KeyClientBuilder;
import com.azure.security.keyvault.keys.models.DeletedKey;
import com.azure.security.keyvault.keys.models.KeyType;
import com.azure.security.keyvault.keys.models.KeyVaultKey;
```

Authentifier et créer un client

Les requêtes d'application vers les Services Azure doivent être autorisées. L'utilisation de la classe [DefaultAzureCredential](#) est l'approche recommandée pour implémenter des connexions sans mot de passe aux services Azure dans votre code.

`DefaultAzureCredential` prend en charge plusieurs méthodes d'authentification et détermine quelle méthode doit être utilisée au moment de l'exécution. Cette approche permet à votre application d'utiliser différentes méthodes d'authentification dans différents environnements (local ou production) sans implémenter de code spécifique à l'environnement.

Dans ce guide de démarrage rapide, `DefaultAzureCredential` s'authentifie auprès du coffre de clés à l'aide des informations d'identification de l'utilisateur de développement local connecté à Azure CLI. Quand l'application est déployée sur Azure, le même code `DefaultAzureCredential` peut découvrir et utiliser automatiquement une identité managée affectée à un service d'application, une machine virtuelle ou d'autres services. Pour plus d'informations, consultez [Vue d'ensemble des identités managées](#).

Dans cet exemple, le nom de votre coffre de clés est étendu à l'URI du coffre de clés, au format `https://<your-key-vault-name>.vault.azure.net`. Pour plus d'informations sur l'authentification auprès du coffre de clés, consultez le [Guide du développeur](#).

Java

```
String keyVaultName = System.getenv("KEY_VAULT_NAME");
String keyVaultUri = "https://" + keyVaultName + ".vault.azure.net";

KeyClient keyClient = new KeyClientBuilder()
    .vaultUrl(keyVaultUri)
    .credential(new DefaultAzureCredentialBuilder().build())
    .buildClient();
```

Créer une clé

Votre application étant authentifiée, vous pouvez créer une clé dans votre coffre de clés en utilisant la méthode `keyClient.createKey`. Un nom pour la clé et un type de clé sont nécessaires. Nous avons attribué la valeur « myKey » à la variable `keyName` et utilisons un `KeyType` RSA dans cet exemple.

Java

```
keyClient.createKey(keyName, KeyType.RSA);
```

Vous pouvez vérifier que la clé a été définie à l'aide de la commande [az keyvault key show](#) :

Azure CLI

```
az keyvault key show --vault-name <your-unique-key-vault-name> --name myKey
```

Récupérer une clé

Vous pouvez maintenant récupérer la clé créée précédemment à l'aide de la méthode `keyClient.getKey`.

Java

```
KeyVaultKey retrievedKey = keyClient.getKey(keyName);
```

Vous pouvez maintenant accéder aux détails de la clé récupérée avec les opérations comme `retrievedKey.getProperties`, `retrievedKey.getKeyOperations`, etc.

Supprimer une clé

Enfin, nous allons supprimer la clé de votre coffre de clés avec la méthode `keyClient.beginDeleteKey`.

La suppression de clé est une opération longue, dont vous pouvez interroger la progression ou attendre la fin.

Java

```
SyncPoller<DeletedKey, Void> deletionPoller =
keyClient.beginDeleteKey(keyName);
deletionPoller.waitForCompletion();
```

Vous pouvez vérifier que la clé a été supprimée avec la commande [az keyvault key show](#) :

Azure CLI

```
az keyvault key show --vault-name <your-unique-key-vault-name> --name myKey
```

Nettoyer les ressources

Une fois que vous n'en avez plus besoin, vous pouvez supprimer votre coffre de clés et le groupe de ressources correspondant en utilisant Azure CLI ou Azure PowerShell.

Azure CLI

```
az group delete -g "myResourceGroup"
```

Azure PowerShell

```
Remove-AzResourceGroup -Name "myResourceGroup"
```

Exemple de code

Java

```
package com.keyvault.keys.quickstart;

import com.azure.core.util.polling.SyncPoller;
import com.azure.identity.DefaultAzureCredentialBuilder;

import com.azure.security.keyvault.keys.KeyClient;
import com.azure.security.keyvault.keys.KeyClientBuilder;
import com.azure.security.keyvault.keys.models.DeletedKey;
import com.azure.security.keyvault.keys.models.KeyType;
import com.azure.security.keyvault.keys.models.KeyVaultKey;

public class App {
    public static void main(String[] args) throws InterruptedException,
IllegalArgumentException {
        String keyVaultName = System.getenv("KEY_VAULT_NAME");
        String keyVaultUri = "https://" + keyVaultName + ".vault.azure.net";

        System.out.printf("key vault name = %s and key vault URI = %s \n",
keyVaultName, keyVaultUri);

        KeyClient keyClient = new KeyClientBuilder()
            .vaultUrl(keyVaultUri)
            .credential(new DefaultAzureCredentialBuilder().build())
            .buildClient();

        String keyName = "myKey";

        System.out.print("Creating a key in " + keyVaultName + " called '" +
keyName + " ... ");

        keyClient.createKey(keyName, KeyType.RSA);

        System.out.print("done.");
        System.out.println("Retrieving key from " + keyVaultName + ".");

        KeyVaultKey retrievedKey = keyClient.getKey(keyName);

        System.out.println("Your key's ID is '" + retrievedKey.getId() +
".'");
        System.out.println("Deleting your key from " + keyVaultName + " ... ");

        SyncPoller<DeletedKey, Void> deletionPoller =
keyClient.beginDeleteKey(keyName);
        deletionPoller.waitForCompletion();
```

```
        System.out.print("done.");
    }
}
```

Étapes suivantes

Dans ce guide de démarrage rapide, vous avez créé un coffre de clés et créé, récupéré, puis supprimé une clé. Pour en savoir plus sur la solution Key Vault et sur la manière de l'intégrer à vos applications, consultez ces articles.

- Lire la [vue d'ensemble Azure Key Vault](#)
- Consultez la [Vue d'ensemble de la sécurité de Key Vault](#)
- Consulter le [Guide du développeur Azure Key Vault](#)
- Découvrir comment [Sécuriser l'accès à un coffre de clés](#)

Démarrage rapide : Bibliothèque de client de secrets Azure Key Vault pour Java

Article • 25/03/2023

Bien démarrer avec la bibliothèque de client de secrets Azure Key Vault pour Java. Suivez les étapes suivantes pour installer le package et essayer un exemple de code pour les tâches de base.

💡 Conseil

Si vous utilisez des ressources de secrets Azure Key Vault dans une application Spring, nous vous recommandons de considérer **Azure Spring Cloud** comme alternative. Azure Spring Cloud est un projet open source qui fournit une intégration de Spring fluide aux services Azure. Pour en découvrir plus sur Spring Cloud Azure et pour voir un exemple d'utilisation de secrets Key Vault, consultez [Charger un secret à partir d'Azure Key Vault dans une application Spring Boot](#).

Ressources supplémentaires :

- [Code source ↗](#)
- [Documentation de référence de l'API ↗](#)
- [Documentation du produit](#)
- [Exemples ↗](#)

Prérequis

- Un abonnement Azure - [En créer un gratuitement↗](#)
- [Kit de développement Java \(JDK\)](#), version 8 ou ultérieure
- [Apache Maven ↗](#)
- [Azure CLI](#)

Ce guide de démarrage rapide suppose que vous exécutez l'interface [Azure CLI](#) et [Apache Maven ↗](#) dans une fenêtre de terminal Linux.

Configuration

Ce guide de démarrage rapide utilise la bibliothèque Azure Identity avec Azure CLI pour authentifier l'utilisateur auprès des services Azure. Les développeurs peuvent également utiliser Visual Studio ou Visual Studio Code pour authentifier leurs appels. Pour plus d'informations, consultez [Authentifier le client avec la bibliothèque de client Azure Identity](#).

Connexion à Azure

1. Exécutez la commande `login`.

```
Azure CLI  
az login
```

Si l'interface CLI peut ouvrir votre navigateur par défaut, elle le fait et charge une page de connexion Azure par la même occasion.

Sinon, ouvrez une page de navigateur à l'adresse <https://aka.ms/devicelogin> et entrez le code d'autorisation affiché dans votre terminal.

2. Dans le navigateur, connectez-vous avec les informations d'identification de votre compte.

Créer une application console Java

Dans une fenêtre de console, utilisez la commande `mvn` pour créer une application console Java nommée `akv-secrets-java`.

```
Console  
mvn archetype:generate -DgroupId=com.keyvault.secrets.quickstart  
-DartifactId=akv-secrets-java  
-DarchetypeArtifactId=maven-archetype-quickstart  
-DarchetypeVersion=1.4  
-DinteractiveMode=false
```

Le résultat de la génération du projet doit ressembler à ceci :

```
Console  
[INFO] -----  
-----  
[INFO] Using following parameters for creating project from Archetype:  
maven-archetype-quickstart:1.4
```

```
[INFO] -----
[INFO] Parameter: groupId, Value: com.keyvault.secrets.quickstart
[INFO] Parameter: artifactId, Value: akv-secrets-java
[INFO] Parameter: version, Value: 1.0-SNAPSHOT
[INFO] Parameter: package, Value: com.keyvault.secrets.quickstart
[INFO] Parameter: packageInPathFormat, Value: com/keyvault/quickstart
[INFO] Parameter: package, Value: com.keyvault.secrets.quickstart
[INFO] Parameter: groupId, Value: com.keyvault.secrets.quickstart
[INFO] Parameter: artifactId, Value: akv-secrets-java
[INFO] Parameter: version, Value: 1.0-SNAPSHOT
[INFO] Project created from Archetype in dir: /home/user/quickstarts/akv-
secrets-java
[INFO] -----
---
[INFO] BUILD SUCCESS
[INFO] -----
---
[INFO] Total time: 38.124 s
[INFO] Finished at: 2019-11-15T13:19:06-08:00
[INFO] -----
---
```

Changez de répertoire pour le dossier `akv-secrets-java/` créé.

Azure CLI

```
cd akv-secrets-java
```

Installer le package

Ouvrez le fichier `pom.xml` dans votre éditeur de texte. Ajoutez les éléments de dépendance suivants au groupe de dépendances.

XML

```
<dependency>
    <groupId>com.azure</groupId>
    <artifactId>azure-security-keyvault-secrets</artifactId>
    <version>4.2.3</version>
</dependency>

<dependency>
    <groupId>com.azure</groupId>
    <artifactId>azure-identity</artifactId>
    <version>1.2.0</version>
</dependency>
```

Créer un groupe de ressources et un coffre de clés

Ce guide de démarrage rapide utilise un coffre de clés Azure créé au préalable. Vous pouvez créer un coffre de clés en suivant les étapes décrites dans le [guide de démarrage rapide d'Azure CLI](#), le [guide de démarrage rapide d'Azure PowerShell](#) ou le [guide de démarrage rapide du portail Azure](#).

Sinon, vous pouvez simplement exécuter les commandes Azure CLI ou Azure PowerShell ci-dessous.

ⓘ Important

Chaque coffre de clés doit avoir un nom unique. Remplacez <your-unique-keyvault-name> par le nom de votre coffre de clés dans les exemples suivants.

Azure CLI

```
Azure CLI  
  
az group create --name "myResourceGroup" -l "EastUS"  
  
az keyvault create --name "<your-unique-keyvault-name>" -g  
"myResourceGroup"
```

Accorder l'accès à votre coffre de clés

Créez une stratégie d'accès pour votre coffre de clés, qui accorde des autorisations de secret à votre compte d'utilisateur.

Azure CLI

```
az keyvault set-policy --name <your-key-vault-name> --upn user@domain.com --  
secret-permissions delete get list set purge
```

Définir des variables d'environnement

Cette application utilise le nom de votre coffre de clés en tant que variable d'environnement appelée `KEY_VAULT_NAME`.

Windows

Invite de commandes Windows

```
set KEY_VAULT_NAME=<your-key-vault-name>
```

Windows PowerShell

PowerShell

```
$Env:KEY_VAULT_NAME="<your-key-vault-name>"
```

macOS ou Linux

Invite de commandes Windows

```
export KEY_VAULT_NAME=<your-key-vault-name>
```

Modèle objet

La bibliothèque de client de secrets Azure Key Vault pour Java vous permet de gérer des secrets. La section [Exemples de code](#) montre comment créer un client, et définir, récupérer et supprimer un secret.

Exemples de code

Ajouter des directives

Ajoutez les directives suivantes en haut de votre code :

Java

```
import com.azure.core.util.polling.SyncPoller;
import com.azure.identity.DefaultAzureCredentialBuilder;

import com.azure.security.keyvault.secrets.SecretClient;
import com.azure.security.keyvault.secrets.SecretClientBuilder;
import com.azure.security.keyvault.secrets.models.DeletedSecret;
import com.azure.security.keyvault.secrets.models.KeyVaultSecret;
```

Authentifier et créer un client

Les requêtes d'application vers les Services Azure doivent être autorisées. L'utilisation de la classe `DefaultAzureCredential` est l'approche recommandée pour implémenter des connexions sans mot de passe aux services Azure dans votre code.

`DefaultAzureCredential` prend en charge plusieurs méthodes d'authentification et détermine quelle méthode doit être utilisée au moment de l'exécution. Cette approche permet à votre application d'utiliser différentes méthodes d'authentification dans différents environnements (local ou production) sans implémenter de code spécifique à l'environnement.

Dans ce guide de démarrage rapide, `DefaultAzureCredential` s'authentifie auprès du coffre de clés à l'aide des informations d'identification de l'utilisateur de développement local connecté à Azure CLI. Quand l'application est déployée sur Azure, le même code `DefaultAzureCredential` peut découvrir et utiliser automatiquement une identité managée affectée à un service d'application, une machine virtuelle ou d'autres services. Pour plus d'informations, consultez [Vue d'ensemble des identités managées](#).

Dans cet exemple, le nom de votre coffre de clés est étendu à l'URI du coffre de clés, au format `https://<your-key-vault-name>.vault.azure.net`. Pour plus d'informations sur l'authentification auprès du coffre de clés, consultez le [Guide du développeur](#).

Java

```
String keyVaultName = System.getenv("KEY_VAULT_NAME");
String keyVaultUri = "https://" + keyVaultName + ".vault.azure.net";

SecretClient secretClient = new SecretClientBuilder()
    .vaultUrl(keyVaultUri)
    .credential(new DefaultAzureCredentialBuilder().build())
    .buildClient();
```

Enregistrer un secret

Maintenant que votre application est authentifiée, vous pouvez placer un secret dans votre coffre en utilisant la méthode `secretClient.setSecret`. Ceci nécessite un nom pour la clé secrète : dans cet exemple, nous avons affecté la valeur « mySecret » à la variable `secretName`.

Java

```
secretClient.setSecret(new KeyVaultSecret(secretName, secretValue));
```

Vous pouvez vérifier que le secret a été défini à l'aide de la commande `az keyvault secret show` :

Azure CLI

```
az keyvault secret show --vault-name <your-unique-key-vault-name> --name  
mySecret
```

Récupérer un secret

Vous pouvez maintenant récupérer le secret défini précédemment avec la méthode `secretClient.getSecret`.

Java

```
KeyVaultSecret retrievedSecret = secretClient.getSecret(secretName);
```

Vous pouvez à présent accéder à la valeur du secret récupéré avec `retrievedSecret.getValue()`.

Supprimer un secret

Enfin, nous allons supprimer le secret de votre coffre de clés avec la méthode `secretClient.beginDeleteSecret`.

La suppression d'un secret est une opération longue, dont vous pouvez interroger la progression ou attendre la fin.

Java

```
SyncPoller<DeletedSecret, Void> deletionPoller =  
secretClient.beginDeleteSecret(secretName);  
deletionPoller.waitForCompletion();
```

Vous pouvez vérifier que le secret a été supprimé à l'aide de la commande `az keyvault secret show` :

Azure CLI

```
az keyvault secret show --vault-name <your-unique-key-vault-name> --name  
mySecret
```

Nettoyer les ressources

Une fois que vous n'en avez plus besoin, vous pouvez supprimer votre coffre de clés et le groupe de ressources correspondant en utilisant Azure CLI ou Azure PowerShell.

Azure CLI

```
az group delete -g "myResourceGroup"
```

Azure PowerShell

```
Remove-AzResourceGroup -Name "myResourceGroup"
```

Exemple de code

Java

```
package com.keyvault.secrets.quickstart;

import java.io.Console;

import com.azure.core.util.polling.SyncPoller;
import com.azure.identity.DefaultAzureCredentialBuilder;

import com.azure.security.keyvault.secrets.SecretClient;
import com.azure.security.keyvault.secrets.SecretClientBuilder;
import com.azure.security.keyvault.secrets.models.DeletedSecret;
import com.azure.security.keyvault.secrets.models.KeyVaultSecret;

public class App {
    public static void main(String[] args) throws InterruptedException,
IllegalArgumentException {
        String keyVaultName = System.getenv("KEY_VAULT_NAME");
        String keyVaultUri = "https://" + keyVaultName + ".vault.azure.net";

        System.out.printf("key vault name = %s and key vault URI = %s \n",
keyVaultName, keyVaultUri);

        SecretClient secretClient = new SecretClientBuilder()
            .vaultUrl(keyVaultUri)
            .credential(new DefaultAzureCredentialBuilder().build())
            .buildClient();

        Console con = System.console();

        String secretName = "mySecret";

        System.out.println("Please provide the value of your secret > ");

        String secretValue = con.readLine();
```

```

        System.out.print("Creating a secret in " + keyVaultName + " called
        '" + secretName + "' with value '" + secretValue + "' ... ");

        secretClient.setSecret(new KeyVaultSecret(secretName, secretValue));

        System.out.println("done.");
        System.out.println("Forgetting your secret.");

        secretValue = "";
        System.out.println("Your secret's value is '" + secretValue + "'.");

        System.out.println("Retrieving your secret from " + keyVaultName +
        ".");

        KeyVaultSecret retrievedSecret = secretClient.getSecret(secretName);

        System.out.println("Your secret's value is '" +
retrievedSecret.getValue() + "'.");

        System.out.print("Deleting your secret from " + keyVaultName + " ...
        ");

        SyncPoller<DeletedSecret, Void> deletionPoller =
secretClient.beginDeleteSecret(secretName);
        deletionPoller.waitForCompletion();

        System.out.println("done.");
    }
}

```

Étapes suivantes

Dans ce guide de démarrage rapide, vous avez créé un coffre de clés et stocké, récupéré, puis supprimé un secret. Pour en savoir plus sur la solution Key Vault et sur la manière de l'intégrer à vos applications, consultez ces articles.

- Lire la [vue d'ensemble Azure Key Vault](#)
- Consulter le [Guide du développeur Azure Key Vault](#)
- Découvrir comment [Sécuriser l'accès à un coffre de clés](#)

Démarrage rapide : Utiliser Java et JDBC avec Azure Database pour MySQL

Article • 03/05/2023

S'APPLIQUE À :  Azure Database pour MySQL - Serveur unique

Important

Azure Database pour MySQL - Serveur unique est en voie de mise hors service. Nous vous recommandons vivement de procéder à une mise à niveau vers Azure Database pour MySQL - Serveur flexible. Pour plus d'informations sur la migration vers le Serveur flexible Azure Database pour MySQL, consultez [Qu'en est-il du Serveur unique Azure Database pour MySQL ?](#)

Cette article montre la création d'un exemple d'application qui utilise Java et [JDBC](#) pour stocker et récupérer des informations dans [Azure Database pour MySQL](#).

JDBC est l'API Java standard pour se connecter à des bases de données relationnelles classiques.

Dans cet article, nous allons inclure deux méthodes d'authentification : l'authentification Azure Active Directory (Azure AD) et l'authentification MySQL. L'onglet **Sans mot de passe** montre l'authentification Azure AD et l'onglet **Mot de passe** montre l'authentification MySQL.

L'authentification Azure AD est un mécanisme de connexion à Azure Database pour MySQL utilisant les identités définies dans Azure AD. Avec l'authentification Azure AD, vous pouvez gérer les identités des utilisateurs de base de données et d'autres services Microsoft dans un emplacement centralisé, ce qui simplifie la gestion des autorisations.

L'authentification MySQL utilise des comptes stockés dans MySQL. Si vous choisissez d'utiliser des mots de passe comme informations d'identification pour les comptes, ces informations d'identification sont stockées dans la table `user`. Comme ces mots de passe sont stockés dans MySQL, vous devez gérer vous-même la rotation des mots de passe.

Prérequis

- Un compte Azure. Si vous n'en avez pas, inscrivez-vous pour un [essai gratuit](#).

- Azure Cloud Shell ou Azure CLI. Nous vous recommandons d'utiliser Azure Cloud Shell. Vous serez ainsi connecté automatiquement et vous aurez accès à tous les outils dont vous avez besoin.
- Un Java Development Kit pris en charge, version 8 (inclus dans Azure Cloud Shell).
- L'outil de build Apache Maven.
- Client en ligne de commande MySQL. Vous pouvez vous connecter à votre serveur en utilisant l'outil en ligne de commande mysql.exe avec Azure Cloud Shell. Vous pouvez aussi utiliser la ligne de commande de mysql dans votre environnement local.

Préparer l'environnement de travail

Tout d'abord, configurez quelques variables d'environnement. Dans Azure Cloud Shell, exéutez les commandes suivantes :

Sans mot de passe (recommandé)

Bash

```
export AZ_RESOURCE_GROUP=database-workshop
export AZ_DATABASE_SERVER_NAME=<YOUR_DATABASE_SERVER_NAME>
export AZ_DATABASE_NAME=demo
export AZ_LOCATION=<YOUR_AZURE_REGION>
export AZ_MYSQL_AD_NON_ADMIN_USERNAME=demo-non-admin
export AZ_LOCAL_IP_ADDRESS=<YOUR_LOCAL_IP_ADDRESS>
export CURRENT_USERNAME=$(az ad signed-in-user show --query
userPrincipalName -o tsv)
export CURRENT_USER_OBJECTID=$(az ad signed-in-user show --query id -o
tsv)
```

Remplacez les espaces réservés par les valeurs suivantes, qui sont utilisées dans cet article :

- <YOUR_DATABASE_SERVER_NAME> : nom de votre serveur MySQL, qui doit être unique dans Azure.
- <YOUR_AZURE_REGION> : région Azure que vous allez utiliser. Vous pouvez utiliser eastus par défaut, mais nous vous recommandons de configurer une région plus proche de l'endroit où vous vivez. Vous pouvez voir la liste complète des régions disponibles en entrant az account list-locations.

- <YOUR_LOCAL_IP_ADDRESS> : adresse IP de votre ordinateur local, à partir duquel vous exécuterez votre application. Un moyen pratique de le trouver est d'ouvrir whatismyip.akamai.com.

Ensute, créez un groupe de ressources à l'aide de la commande suivante :

Azure CLI

```
az group create \
--name $AZ_RESOURCE_GROUP \
--location $AZ_LOCATION \
--output tsv
```

Créer une instance Azure Database pour MySQL

Créer un serveur MySQL et configurer un utilisateur administrateur

La première chose que vous allez créer est un serveur MySQL managé.

ⓘ Notes

Vous trouverez des informations plus détaillées sur la création de serveurs MySQL dans [Démarrage rapide : Créer un serveur Azure Database pour MySQL en utilisant le portail Azure](#).

Sans mot de passe (recommandé)

Si vous utilisez Azure CLI, exécutez la commande suivante pour vous assurer de disposer d'une autorisation suffisante :

Azure CLI

```
az login --scope https://graph.microsoft.com/.default
```

Exécutez ensuite la commande suivante pour créer le serveur :

Azure CLI

```
az mysql server create \
--resource-group $AZ_RESOURCE_GROUP \
--name $AZ_DATABASE_SERVER_NAME \
--location $AZ_LOCATION \
--sku-name B_Gen5_1 \
--storage-size 5120 \
--output tsv
```

Exécutez ensuite la commande suivante pour définir l'utilisateur administrateur Azure AD :

Azure CLI

```
az mysql server ad-admin create \
--resource-group $AZ_RESOURCE_GROUP \
--server-name $AZ_DATABASE_SERVER_NAME \
--display-name $CURRENT_USERNAME \
--object-id $CURRENT_USER_OBJECTID
```

ⓘ Important

Lorsque vous définissez l'administrateur, un nouvel utilisateur est ajouté au serveur Azure Database pour MySQL avec toutes les autorisations d'administrateur. Vous ne pouvez créer qu'un seul administrateur Azure AD par serveur MySQL. La sélection d'un autre utilisateur va remplacer l'administrateur Azure AD existant configuré pour le serveur.

Cette commande crée un petit serveur MySQL et définit l'administrateur Active Directory sur l'utilisateur connecté.

Configurer une règle de pare-feu pour votre serveur MySQL

Les instances Azure Database pour MySQL sont sécurisées par défaut. Ces instances ont un pare-feu qui n'autorise aucune connexion entrante. Pour pouvoir utiliser notre base de données, vous devez ajouter une règle de pare-feu qui permet à l'adresse IP locale d'accéder au serveur de base de données.

Comme vous avez configuré votre adresse IP locale au début de cet article, vous pouvez ouvrir le pare-feu du serveur en exécutant la commande suivante :

Azure CLI

```
az mysql server firewall-rule create \
--resource-group $AZ_RESOURCE_GROUP \
--name $AZ_DATABASE_SERVER_NAME-database-allow-local-ip \
--server $AZ_DATABASE_SERVER_NAME \
--start-ip-address $AZ_LOCAL_IP_ADDRESS \
--end-ip-address $AZ_LOCAL_IP_ADDRESS \
--output tsv
```

Si vous vous connectez à votre serveur MySQL à partir du Sous-système Windows pour Linux (WSL) sur un ordinateur Windows, vous devez ajouter l'ID d'hôte WSL à votre pare-feu.

Obtenez l'adresse IP de votre ordinateur hôte en exécutant la commande suivante dans WSL :

Bash

```
cat /etc/resolv.conf
```

Copiez l'adresse IP suivant le terme `nameserver`, puis utilisez la commande suivante pour définir une variable d'environnement pour l'adresse IP de WSL :

Bash

```
AZ_WSL_IP_ADDRESS=<the-copied-IP-address>
```

Ensuite, utilisez la commande suivante pour ouvrir le pare-feu du serveur sur votre application WSL :

Azure CLI

```
az mysql server firewall-rule create \
--resource-group $AZ_RESOURCE_GROUP \
--name $AZ_DATABASE_SERVER_NAME-database-allow-local-ip-wsl \
--server $AZ_DATABASE_SERVER_NAME \
--start-ip-address $AZ_WSL_IP_ADDRESS \
--end-ip-address $AZ_WSL_IP_ADDRESS \
--output tsv
```

Configurer une base de données MySQL

Le serveur MySQL que vous avez créé précédemment est vide. Créez une base de données avec la commande suivante.

Azure CLI

```
az mysql db create \
--resource-group $AZ_RESOURCE_GROUP \
--name $AZ_DATABASE_NAME \
--server-name $AZ_DATABASE_SERVER_NAME \
--output tsv
```

Créer un utilisateur non-administrateur MySQL et accorder des autorisations

Ensuite, créez un utilisateur non-administrateur et accordez toutes les autorisations sur la base de données.

ⓘ Notes

Vous pouvez lire des informations plus détaillées sur la création d'utilisateurs MySQL dans [Créer des utilisateurs dans Azure Database pour MySQL](#).

Sans mot de passe (recommandé)

Créez un script SQL appelé *create_ad_user.sql* pour créer un utilisateur non administrateur. Ajoutez le contenu suivant et enregistrez-le localement :

Bash

```
export AZ_MYSQL_AD_NON_ADMIN_USERID=$CURRENT_USER_OBJECTID

cat << EOF > create_ad_user.sql
SET aad_auth_validate_oids_in_tenant = OFF;

CREATE AADUSER '$AZ_MYSQL_AD_NON_ADMIN_USERNAME' IDENTIFIED BY
'$AZ_MYSQL_AD_NON_ADMIN_USERID';

GRANT ALL PRIVILEGES ON $AZ_DATABASE_NAME.* TO
'$AZ_MYSQL_AD_NON_ADMIN_USERNAME'@'%';

FLUSH privileges;

EOF
```

Utilisez ensuite la commande suivante pour exécuter le script SQL pour créer l'utilisateur non administrateur Azure AD :

Bash

```
mysql -h $AZ_DATABASE_SERVER_NAME.mysql.database.azure.com --user  
$CURRENT_USERNAME@$AZ_DATABASE_SERVER_NAME --enable-cleartext-plugin --  
password=$(az account get-access-token --resource-type oss-rdbms --  
output tsv --query accessToken) < create_ad_user.sql
```

Utilisez maintenant la commande suivante pour supprimer le fichier de script SQL temporaire :

Bash

```
rm create_ad_user.sql
```

Créer un projet Java

En utilisant votre IDE préféré, créez un projet Java en utilisant Java 8 ou ultérieur. Créez un fichier *pom.xml* dans son répertoire racine et ajoutez le contenu suivant :

Sans mot de passe (recommandé)

XML

```
<?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  
https://maven.apache.org/xsd/maven-4.0.0.xsd">  
    <modelVersion>4.0.0</modelVersion>  
    <groupId>com.example</groupId>  
    <artifactId>demo</artifactId>  
    <version>0.0.1-SNAPSHOT</version>  
    <name>demo</name>  
  
    <properties>  
        <java.version>1.8</java.version>  
        <maven.compiler.source>1.8</maven.compiler.source>  
        <maven.compiler.target>1.8</maven.compiler.target>  
    </properties>  
  
    <dependencies>  
        <dependency>  
            <groupId>mysql</groupId>  
            <artifactId>mysql-connector-java</artifactId>  
            <version>8.0.30</version>  
        </dependency>  
        <dependency>
```

```
<groupId>com.azure</groupId>
<artifactId>azure-identity-extensions</artifactId>
<version>1.0.0</version>
</dependency>
</dependencies>
</project>
```

Ce fichier est un fichier [Apache Maven](#) qui configure votre projet pour utiliser Java 8 et un pilote MySQL récent pour Java.

Préparer un fichier de configuration pour se connecter à Azure Database pour MySQL

Exécutez le script suivant dans le répertoire racine du projet pour créer un fichier *src/main/resources/database.properties* et ajoutez des détails de configuration :

Sans mot de passe (recommandé)

Bash

```
mkdir -p src/main/resources && touch
src/main/resources/database.properties

cat << EOF > src/main/resources/database.properties
url=jdbc:mysql://${AZ_DATABASE_SERVER_NAME}.mysql.database.azure.com:330
6/${AZ_DATABASE_NAME}?
sslMode=REQUIRED&serverTimezone=UTC&defaultAuthenticationPlugin=com.azure.identity.extensions.jdbc.mysql.AzureMysqlAuthenticationPlugin&authenti
cationPlugins=com.azure.identity.extensions.jdbc.mysql.AzureMysqlAuthent
icationPlugin
user=${AZ_MYSQL_AD_NON_ADMIN_USERNAME}@${AZ_DATABASE_SERVER_NAME}
EOF
```

ⓘ Notes

Si vous utilisez la classe `MysqlConnectionPoolDataSource` comme source de données dans votre application, supprimez
« `defaultAuthenticationPlugin=com.azure.identity.extensions.jdbc.mysql.AzureMysqlAuthenticationPlugin` » dans l'URL.

Bash

```
mkdir -p src/main/resources && touch
src/main/resources/database.properties
```

```
cat << EOF > src/main/resources/database.properties
url=jdbc:mysql://${AZ_DATABASE_SERVER_NAME}.mysql.database.azure.com:330
6/${AZ_DATABASE_NAME}?
sslMode=REQUIRED&serverTimezone=UTC&authenticationPlugins=com.azure.iden
tity.extensions.jdbc.mysql.AzureMysqlAuthenticationPlugin
user=${AZ_MYSQL_AD_NON_ADMIN_USERNAME}@${AZ_DATABASE_SERVER_NAME}
EOF
```

ⓘ Notes

?serverTimezone=UTC a été ajouté à la propriété de configuration url pour indiquer au pilote JDBC d'utiliser le format de date UTC (Coordinated Universal Time) lors de la connexion à la base de données. Sans cela, votre serveur Java n'utilisera pas le même format de date que la base de données, ce qui entraînerait une erreur.

Créer un fichier SQL pour générer le schéma de base de données

Ensute, vous allez utiliser un fichier *src/main/resources/schema.sql* pour créer un schéma de base de données. Créez ce fichier et ajoutez le contenu suivant :

Bash

```
touch src/main/resources/schema.sql

cat << EOF > src/main/resources/schema.sql
DROP TABLE IF EXISTS todo;
CREATE TABLE todo (id SERIAL PRIMARY KEY, description VARCHAR(255), details
VARCHAR(4096), done BOOLEAN);
EOF
```

Coder l'application

Se connecter à la base de données

Ensute, ajoutez le code Java qui utilisera JDBC pour stocker et récupérer des données à partir de votre serveur MySQL.

Créez un fichier *src/main/java/DemoApplication.java*, puis ajoutez le contenu suivant :

Java

```
package com.example.demo;

import com.mysql.cj.jdbc.AbandonedConnectionCleanupThread;

import java.sql.*;
import java.util.*;
import java.util.logging.Logger;

public class DemoApplication {

    private static final Logger log;

    static {
        System.setProperty("java.util.logging.SimpleFormatter.format", "[%4$-7s] %5$s %n");
        log =Logger.getLogger(DemoApplication.class.getName());
    }

    public static void main(String[] args) throws Exception {
        log.info("Loading application properties");
        Properties properties = new Properties();

        properties.load(DemoApplication.class.getClassLoader().getResourceAsStream("database.properties"));

        log.info("Connecting to the database");
        Connection connection =
DriverManager.getConnection(properties.getProperty("url"), properties);
        log.info("Database connection test: " + connection.getCatalog());

        log.info("Create database schema");
        Scanner scanner = new
Scanner(DemoApplication.class.getClassLoader().getResourceAsStream("schema.sql"));
        Statement statement = connection.createStatement();
        while (scanner.hasNextLine()) {
            statement.execute(scanner.nextLine());
        }

        /* Prepare to store and retrieve data from the MySQL server.
        Todo todo = new Todo(1L, "configuration", "congratulations, you have
set up JDBC correctly!", true);
        insertData(todo, connection);
        todo = readData(connection);
        todo.setDetails("congratulations, you have updated data!");
        updateData(todo, connection);
        deleteData(todo, connection);
        */

        log.info("Closing database connection");
        connection.close();
        AbandonedConnectionCleanupThread.uncheckedShutdown();
    }
}
```

```
    }  
}
```

Ce code Java utilise les fichiers *database.properties* et *schema.sql* que vous avez créés précédemment. Après vous être connecté au serveur MySQL, vous pouvez créer un schéma pour stocker vos données.

Dans ce fichier, vous pouvez voir que nous avons commenté les méthodes pour insérer, lire, mettre à jour et supprimer des données. Vous allez implémenter ces méthodes dans le reste de cet article, et vous pourrez les décommenter l'une après l'autre.

ⓘ Notes

Les informations d'identification de base de données sont stockées dans les propriétés *user* et *password* du fichier *database.properties*. Ces informations d'identification sont utilisées lors de l'exécution de

`DriverManager.getConnection(properties.getProperty("url"), properties);`, car le fichier de propriétés est passé comme argument.

ⓘ Notes

La ligne `AbandonedConnectionCleanupThread.uncheckedShutdown();` à la fin est une commande du pilote MySQL pour détruire un thread interne lors de l'arrêt de l'application. Vous pouvez sans problème ignorer cette ligne.

Vous pouvez maintenant exécuter cette classe main avec votre outil favori :

- Avec votre IDE, vous devez être en mesure de cliquer avec le bouton droit sur la classe *DemoApplication* et de l'exécuter.
- En utilisant Maven, vous pouvez exécuter l'application avec la commande suivante :
`mvn exec:java -Dexec.mainClass="com.example.demo.DemoApplication".`

L'application doit se connecter à Azure Database pour MySQL, créer un schéma de base de données, puis fermer la connexion. Vous devez voir une sortie similaire à l'exemple suivant dans les journaux de la console :

Sortie

```
[INFO    ] Loading application properties  
[INFO    ] Connecting to the database  
[INFO    ] Database connection test: demo
```

```
[INFO    ] Create database schema
[INFO    ] Closing database connection
```

Créer une classe de domaine

Créez une classe Java `Todo` parallèlement à la classe `DemoApplication`, puis ajoutez le code suivant :

Java

```
package com.example.demo;

public class Todo {

    private Long id;
    private String description;
    private String details;
    private boolean done;

    public Todo() {
    }

    public Todo(Long id, String description, String details, boolean done) {
        this.id = id;
        this.description = description;
        this.details = details;
        this.done = done;
    }

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }

    public String getDetails() {
        return details;
    }

    public void setDetails(String details) {
        this.details = details;
    }
}
```

```

    }

    public boolean isDone() {
        return done;
    }

    public void setDone(boolean done) {
        this.done = done;
    }

    @Override
    public String toString() {
        return "Todo{" +
            "id=" + id +
            ", description='" + description + '\'' +
            ", details='" + details + '\'' +
            ", done=" + done +
            '}';
    }
}

```

Cette classe est un modèle de domaine mappé sur la table `todo` que vous avez créée lors de l'exécution du script `schema.sql`.

Insertion des données dans Azure Database pour MySQL

Dans le fichier `src/main/java/DemoApplication.java`, après la méthode `main`, ajoutez la méthode suivante pour insérer des données dans la base de données :

Java

```

private static void insertData(Todo todo, Connection connection) throws
SQLException {
    log.info("Insert data");
    PreparedStatement insertStatement = connection
        .prepareStatement("INSERT INTO todo (id, description, details,
done) VALUES (?, ?, ?, ?);");

    insertStatement.setLong(1, todo.getId());
    insertStatement.setString(2, todo.getDescription());
    insertStatement.setString(3, todo.getDetails());
    insertStatement.setBoolean(4, todo.isDone());
    insertStatement.executeUpdate();
}

```

Vous pouvez maintenant supprimer les marques de commentaire des deux lignes suivantes dans la méthode `main` :

Java

```
Todo todo = new Todo(1L, "configuration", "congratulations, you have set up  
JDBC correctly!", true);  
insertData(todo, connection);
```

L'exécution de la classe main doit maintenant produire la sortie suivante :

Sortie

```
[INFO ] Loading application properties  
[INFO ] Connecting to the database  
[INFO ] Database connection test: demo  
[INFO ] Create database schema  
[INFO ] Insert data  
[INFO ] Closing database connection
```

Lecture des données depuis Azure Database pour MySQL

Ensuite, lisez les données que vous avez précédemment insérées pour vérifier le bon fonctionnement de votre code.

Dans le fichier `src/main/java/DemoApplication.java`, après la méthode `insertData`, ajoutez la méthode suivante pour lire des données de la base de données :

Java

```
private static Todo readData(Connection connection) throws SQLException {  
    log.info("Read data");  
    PreparedStatement readStatement = connection.prepareStatement("SELECT *  
FROM todo;");  
    ResultSet resultSet = readStatement.executeQuery();  
    if (!resultSet.next()) {  
        log.info("There is no data in the database!");  
        return null;  
    }  
    Todo todo = new Todo();  
    todo.setId(resultSet.getLong("id"));  
    todo.setDescription(resultSet.getString("description"));  
    todo.setDetails(resultSet.getString("details"));  
    todo.setDone(resultSet.getBoolean("done"));  
    log.info("Data read from the database: " + todo.toString());  
    return todo;  
}
```

Vous pouvez maintenant supprimer les marques de commentaire de la ligne suivante dans la méthode `main` :

Java

```
todo = readData(connection);
```

L'exécution de la classe main doit maintenant produire la sortie suivante :

Sortie

```
[INFO    ] Loading application properties
[INFO    ] Connecting to the database
[INFO    ] Database connection test: demo
[INFO    ] Create database schema
[INFO    ] Insert data
[INFO    ] Read data
[INFO    ] Data read from the database: Todo{id=1,
description='configuration', details='congratulations, you have set up JDBC
correctly!', done=true}
[INFO    ] Closing database connection
```

Mise à jour des données dans Azure Database pour MySQL

Ensuite, mettez à jour les données que vous avez insérées précédemment.

Toujours dans le fichier *src/main/java/DemoApplication.java*, après la méthode `readData`, ajoutez la méthode suivante pour mettre à jour des données dans la base de données :

Java

```
private static void updateData(Todo todo, Connection connection) throws
SQLException {
    log.info("Update data");
    PreparedStatement updateStatement = connection
        .prepareStatement("UPDATE todo SET description = ?, details = ?,
done = ? WHERE id = ?;");
    updateStatement.setString(1, todo.getDescription());
    updateStatement.setString(2, todo.getDetails());
    updateStatement.setBoolean(3, todo.isDone());
    updateStatement.setLong(4, todo.getId());
    updateStatement.executeUpdate();
    readData(connection);
}
```

Vous pouvez maintenant supprimer les marques de commentaire des deux lignes suivantes dans la méthode `main` :

Java

```
todo.setDetails("congratulations, you have updated data!");
updateData(todo, connection);
```

L'exécution de la classe main doit maintenant produire la sortie suivante :

Sortie

```
[INFO    ] Loading application properties
[INFO    ] Connecting to the database
[INFO    ] Database connection test: demo
[INFO    ] Create database schema
[INFO    ] Insert data
[INFO    ] Read data
[INFO    ] Data read from the database: Todo{id=1,
description='configuration', details='congratulations, you have set up JDBC
correctly!', done=true}
[INFO    ] Update data
[INFO    ] Read data
[INFO    ] Data read from the database: Todo{id=1,
description='configuration', details='congratulations, you have updated
data!', done=true}
[INFO    ] Closing database connection
```

Suppression des données dans Azure Database pour MySQL

Enfin, supprimez les données que vous avez insérées précédemment.

Toujours dans le fichier *src/main/java/DemoApplication.java*, après la méthode `updateData`, ajoutez la méthode suivante pour supprimer des données dans la base de données :

Java

```
private static void deleteData(Todo todo, Connection connection) throws
SQLException {
    log.info("Delete data");
    PreparedStatement deleteStatement = connection.prepareStatement("DELETE
FROM todo WHERE id = ?;");
    deleteStatement.setLong(1, todo.getId());
    deleteStatement.executeUpdate();
    readData(connection);
}
```

Vous pouvez maintenant supprimer les marques de commentaire de la ligne suivante dans la méthode `main` :

Java

```
deleteData(todo, connection);
```

L'exécution de la classe main doit maintenant produire la sortie suivante :

Sortie

```
[INFO    ] Loading application properties
[INFO    ] Connecting to the database
[INFO    ] Database connection test: demo
[INFO    ] Create database schema
[INFO    ] Insert data
[INFO    ] Read data
[INFO    ] Data read from the database: Todo{id=1,
description='configuration', details='congratulations, you have set up JDBC
correctly!', done=true}
[INFO    ] Update data
[INFO    ] Read data
[INFO    ] Data read from the database: Todo{id=1,
description='configuration', details='congratulations, you have updated
data!', done=true}
[INFO    ] Delete data
[INFO    ] Read data
[INFO    ] There is no data in the database!
[INFO    ] Closing database connection
```

Nettoyer les ressources

Félicitations ! Vous avez créé une application Java qui utilise JDBC pour stocker et récupérer des données dans Azure Database pour MySQL.

Pour nettoyer toutes les ressources utilisées dans le cadre de ce guide de démarrage rapide, supprimez le groupe de ressources à l'aide de la commande suivante :

Azure CLI

```
az group delete \
--name $AZ_RESOURCE_GROUP \
--yes
```

Étapes suivantes

Migrer une base de données MySQL vers une base de données Azure pour MySQL à l'aide des images mémoire et de la restauration

Démarrage rapide : Utiliser Java et JDBC avec Azure Database pour PostgreSQL

Article • 23/10/2023

S'APPLIQUE À :  Azure Database pour PostgreSQL – Serveur unique

Important

Azure Database pour PostgreSQL - Serveur unique est en voie de mise hors service.

Nous vous recommandons vivement de procéder à une mise à niveau vers Azure Database pour PostgreSQL - Serveur flexible. Pour plus d'informations sur la migration vers le Serveur flexible Azure Database pour PostgreSQL, consultez l'article [Qu'arrive-t-il au Serveur unique Azure Database pour PostgreSQL ?](#)

Cet article illustre la création d'un exemple d'application qui utilise Java et [JDBC](#) pour stocker et récupérer des informations dans [Azure Database pour PostgreSQL](#).

JDBC est l'API Java standard pour se connecter à des bases de données relationnelles classiques.

Dans cet article, nous allons inclure deux méthodes d'authentification : l'authentification Microsoft Entra et l'authentification PostgreSQL. L'onglet **Sans mot de passe** affiche l'authentification Microsoft Entra et l'onglet **Mot de passe** affiche l'authentification PostgreSQL.

L'authentification Microsoft Entra est un mécanisme de connexion à Azure Database pour PostgreSQL utilisant les identités définies dans Microsoft Entra ID. Avec l'authentification Microsoft Entra, vous pouvez gérer les identités des utilisateurs de base de données et d'autres services Microsoft dans un emplacement centralisé, ce qui simplifie la gestion des autorisations.

L'authentification PostgreSQL utilise des comptes stockés dans PostgreSQL. Si vous choisissez d'utiliser des mots de passe comme informations d'identification pour les comptes, ces informations d'identification sont stockées dans la table `user`. Étant donné que ces mots de passe sont stockés dans PostgreSQL, vous devez gérer vous-même la rotation des mots de passe.

Prérequis

- Un compte Azure. Si vous n'en avez pas, inscrivez-vous pour un [essai gratuit](#).

- Azure Cloud Shell ou Azure CLI 2.37.0 ou version ultérieure requise. Nous vous recommandons d'utiliser Azure Cloud Shell. Vous serez ainsi connecté automatiquement et vous aurez accès à tous les outils dont vous avez besoin.
- Un Java Development Kit pris en charge, version 8 (inclus dans Azure Cloud Shell).
- L'outil de build Apache Maven ↗.

Préparer l'environnement de travail

Tout d'abord, configurez quelques variables d'environnement. Dans Azure Cloud Shell ↗, exécutez les commandes suivantes :

Sans mot de passe (recommandé)

Bash

```
export AZ_RESOURCE_GROUP=database-workshop
export AZ_DATABASE_SERVER_NAME=<YOUR_DATABASE_SERVER_NAME>
export AZ_DATABASE_NAME=demo
export AZ_LOCATION=<YOUR_AZURE_REGION>
export AZ_POSTGRESQL_AD_NON_ADMIN_USERNAME=
<YOUR_POSTGRESQL_AD_NON_ADMIN_USERNAME>
export AZ_LOCAL_IP_ADDRESS=<YOUR_LOCAL_IP_ADDRESS>
export CURRENT_USERNAME=$(az ad signed-in-user show --query
userPrincipalName -o tsv)
export CURRENT_USER_OBJECTID=$(az ad signed-in-user show --query id -o
tsv)
```

Remplacez les espaces réservés par les valeurs suivantes, qui sont utilisées dans cet article :

- <YOUR_DATABASE_SERVER_NAME> : nom de votre serveur PostgreSQL, qui doit être unique dans Azure.
- <YOUR_AZURE_REGION> : région Azure que vous allez utiliser. Vous pouvez utiliser `eastus` par défaut, mais nous vous recommandons de configurer une région plus proche de l'endroit où vous vivez. Vous pouvez voir la liste complète des régions disponibles en entrant `az account list-locations`.
- <YOUR_POSTGRESQL_AD_NON_ADMIN_USERNAME> : le nom d'utilisateur de votre serveur de base de données PostgreSQL. Assurez-vous que le nom d'utilisateur est un utilisateur valide dans votre locataire Microsoft Entra.
- <YOUR_LOCAL_IP_ADDRESS> : adresse IP de votre ordinateur local, à partir duquel vous exécuterez l'application Spring Boot. Un moyen pratique de le trouver est d'ouvrir whatismyip.akamai.com ↗.

ⓘ Important

Lorsque vous définissez <YOUR_POSTGRESQL_AD_NON_ADMIN_USERNAME>, le nom d'utilisateur doit déjà exister dans votre locataire Microsoft Entra ou vous ne pourrez pas créer un utilisateur Microsoft Entra dans votre base de données.

Ensute, créez un groupe de ressources à l'aide de la commande suivante :

Azure CLI

```
az group create \
--name $AZ_RESOURCE_GROUP \
--location $AZ_LOCATION \
--output tsv
```

Créer une instance Azure Database pour PostgreSQL

Les sections suivantes décrivent comment créer et configurer votre instance de base de données.

Créer un serveur PostgreSQL et configurer un utilisateur administrateur

La première chose que vous allez créer est un serveur PostgreSQL managé avec un utilisateur administrateur.

ⓘ Notes

Vous trouverez des informations plus détaillées sur la création de serveurs PostgreSQL dans l'article [Création d'un serveur Azure Database pour PostgreSQL à l'aide du portail Azure](#).

Sans mot de passe (recommandé)

Si vous utilisez Azure CLI, exécutez la commande suivante pour vous assurer de disposer d'une autorisation suffisante :

Azure CLI

```
az login --scope https://graph.microsoft.com/.default
```

Exécutez ensuite la commande suivante pour créer le serveur :

Azure CLI

```
az postgres server create \
--resource-group $AZ_RESOURCE_GROUP \
--name $AZ_DATABASE_SERVER_NAME \
--location $AZ_LOCATION \
--sku-name B_Gen5_1 \
--storage-size 5120 \
--output tsv
```

Exécutez maintenant la commande suivante pour définir l'utilisateur administrateur Microsoft Entra :

Azure CLI

```
az postgres server ad-admin create \
--resource-group $AZ_RESOURCE_GROUP \
--server-name $AZ_DATABASE_SERVER_NAME \
--display-name $CURRENT_USERNAME \
--object-id $CURRENT_USER_OBJECTID
```

ⓘ Important

Lorsque vous définissez l'administrateur, un nouvel utilisateur est ajouté au serveur Azure Database pour PostgreSQL avec les autorisations d'administrateur complètes. Un seul administrateur Microsoft Entra peut être créé par serveur PostgreSQL et la sélection d'un autre administrateur remplacera l'administrateur Microsoft Entra existant configuré pour le serveur.

Cette commande crée un petit serveur PostgreSQL et définit l'administrateur Active Directory sur l'utilisateur connecté.

Configurer une règle de pare-feu pour votre serveur PostgreSQL

Les instances Azure Database pour PostgreSQL sont sécurisées par défaut. Elles ont un pare-feu qui n'autorise aucune connexion entrante. Pour pouvoir utiliser notre base de données, vous devez ajouter une règle de pare-feu qui permet à l'adresse IP locale d'accéder au serveur de base de données.

Comme vous avez configuré votre adresse IP locale au début de cet article, vous pouvez ouvrir le pare-feu du serveur en exécutant la commande suivante :

Azure CLI

```
az postgres server firewall-rule create \
--resource-group $AZ_RESOURCE_GROUP \
--name $AZ_DATABASE_SERVER_NAME-database-allow-local-ip \
--server $AZ_DATABASE_SERVER_NAME \
--start-ip-address $AZ_LOCAL_IP_ADDRESS \
--end-ip-address $AZ_LOCAL_IP_ADDRESS \
--output tsv
```

Si vous vous connectez à votre serveur PostgreSQL à partir du Sous-système Windows pour Linux (WSL) sur un ordinateur Windows, vous devez ajouter l'ID d'hôte WSL à votre pare-feu.

Obtenez l'adresse IP de votre ordinateur hôte en exécutant la commande suivante dans WSL :

Bash

```
cat /etc/resolv.conf
```

Copiez l'adresse IP suivant le terme `nameserver`, puis utilisez la commande suivante pour définir une variable d'environnement pour l'adresse IP de WSL :

Bash

```
AZ_WSL_IP_ADDRESS=<the-copied-IP-address>
```

Ensuite, utilisez la commande suivante pour ouvrir le pare-feu du serveur sur votre application WSL :

Azure CLI

```
az postgres server firewall-rule create \
--resource-group $AZ_RESOURCE_GROUP \
--name $AZ_DATABASE_SERVER_NAME-database-allow-local-ip \
--server $AZ_DATABASE_SERVER_NAME \
--start-ip-address $AZ_WSL_IP_ADDRESS \
```

```
--end-ip-address $AZ_WSL_IP_ADDRESS \
--output tsv
```

Configurer une base de données PostgreSQL

Le serveur PostgreSQL que vous avez créé précédemment est vide. Créez une base de données avec la commande suivante.

Azure CLI

```
az postgres db create \
--resource-group $AZ_RESOURCE_GROUP \
--name $AZ_DATABASE_NAME \
--server-name $AZ_DATABASE_SERVER_NAME \
--output tsv
```

Créer un utilisateur non administrateur PostgreSQL et accorder des autorisations

Ensuite, créez un utilisateur non-administrateur et accordez toutes les autorisations à la base de données.

ⓘ Notes

Vous pouvez lire des informations plus détaillées sur la création d'utilisateurs PostgreSQL dans [Créer des utilisateurs dans Azure Database pour PostgreSQL](#).

Sans mot de passe (recommandé)

Créez un script SQL appelé *create_ad_user.sql* pour créer un utilisateur non administrateur. Ajoutez le contenu suivant et enregistrez-le localement :

Bash

```
cat << EOF > create_ad_user.sql
SET aad_validate_oids_in_tenant = off;
CREATE ROLE "$AZ_POSTGRESQL_AD_NON_ADMIN_USERNAME" WITH LOGIN IN ROLE
azure_ad_user;
GRANT ALL PRIVILEGES ON DATABASE $AZ_DATABASE_NAME TO
"$AZ_POSTGRESQL_AD_NON_ADMIN_USERNAME";
EOF
```

Utilisez ensuite la commande suivante pour exécuter le script SQL pour créer l'utilisateur non administrateur Microsoft Entra :

Bash

```
psql "host=$AZ_DATABASE_SERVER_NAME.postgres.database.azure.com
user=$CURRENT_USERNAME@$AZ_DATABASE_SERVER_NAME dbname=$AZ_DATABASE_NAME
port=5432 password=$(az account get-access-token --resource-type oss-
rdbms --output tsv --query accessToken) sslmode=require" <
create_ad_user.sql
```

Utilisez maintenant la commande suivante pour supprimer le fichier de script SQL temporaire :

Bash

```
rm create_ad_user.sql
```

Créer un projet Java

À l'aide de votre IDE favori, créez un projet Java avec Java 8 ou version ultérieure, puis ajoutez un fichier *pom.xml* dans son répertoire racine avec le contenu suivant :

Sans mot de passe (recommandé)

XML

```
<?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
  https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.example</groupId>
  <artifactId>demo</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>demo</name>

  <properties>
    <java.version>1.8</java.version>
    <maven.compiler.source>1.8</maven.compiler.source>
    <maven.compiler.target>1.8</maven.compiler.target>
  </properties>

  <dependencies>
    <dependency>
      <groupId>org.postgresql</groupId>
```

```
<artifactId>postgresql</artifactId>
  <version>42.3.6</version>
</dependency>
<dependency>
  <groupId>com.azure</groupId>
  <artifactId>azure-identity-extensions</artifactId>
  <version>1.0.0</version>
</dependency>
</dependencies>
</project>
```

Ce fichier est un fichier [Apache Maven](#) qui configure votre projet pour utiliser Java 8 et un pilote PostgreSQL récent pour Java.

Préparer un fichier de configuration pour se connecter à Azure Database pour PostgreSQL

Créez un fichier `src/main/resources/application.properties`, puis ajoutez le contenu suivant :

Sans mot de passe (recommandé)

Bash

```
cat << EOF > src/main/resources/application.properties
url=jdbc:postgresql://${AZ_DATABASE_SERVER_NAME}.postgres.database.azure
.com:5432/${AZ_DATABASE_NAME}?
sslmode=require&authenticationPluginClassName=com.azure.identity.extensions.jdbc.postgresql.AzurePostgresqlAuthenticationPlugin
user=${AZ_POSTGRESQL_AD_NON_ADMIN_USERNAME}@${AZ_DATABASE_SERVER_NAME}
EOF
```

ⓘ Notes

Nous ajoutons `url` à la propriété de configuration `?sslmode=require` pour indiquer au pilote JDBC d'utiliser TLS ([protocole TLS](#)) lors de la connexion à la base de données. L'utilisation du protocole TLS avec Azure Database pour PostgreSQL est obligatoire et constitue une pratique de sécurité.

Créer un fichier SQL pour générer le schéma de base de données

Vous allez utiliser un fichier `src/main/resources/schema.sql` pour créer un schéma de base de données. Créez ce fichier et ajoutez le contenu suivant :

Bash

```
cat << EOF > src/main/resources/schema.sql
DROP TABLE IF EXISTS todo;
CREATE TABLE todo (id SERIAL PRIMARY KEY, description VARCHAR(255), details
VARCHAR(4096), done BOOLEAN);
EOF
```

Coder l'application

Se connecter à la base de données

Ensute, ajoutez le code Java qui utilisera JDBC pour stocker et récupérer des données à partir de votre serveur PostgreSQL.

Créez un fichier `src/main/java/DemoApplication.java`, puis ajoutez le contenu suivant :

Java

```
package com.example.demo;

import java.sql.*;
import java.util.*;
import java.util.logging.Logger;

public class DemoApplication {

    private static final Logger log;

    static {
        System.setProperty("java.util.logging.SimpleFormatter.format", "%4$-7s] %5$s %n");
        log =Logger.getLogger(DemoApplication.class.getName());
    }

    public static void main(String[] args) throws Exception {
        log.info("Loading application properties");
        Properties properties = new Properties();

        properties.load(DemoApplication.class.getClassLoader().getResourceAsStream("application.properties"));

        log.info("Connecting to the database");
        Connection connection =
DriverManager.getConnection(properties.getProperty("url"), properties);
        log.info("Database connection test: " + connection.getCatalog());
```

```

        log.info("Create database schema");
        Scanner scanner = new
Scanner(DemoApplication.class.getClassLoader().getResourceAsStream( "schema.s
ql"));
        Statement statement = connection.createStatement();
        while (scanner.hasNextLine()) {
            statement.execute(scanner.nextLine());
        }

        /* Prepare for data processing in the PostgreSQL server.
        Todo todo = new Todo(1L, "configuration", "congratulations, you have
set up JDBC correctly!", true);
        insertData(todo, connection);
        todo = readData(connection);
        todo.setDetails("congratulations, you have updated data!");
        updateData(todo, connection);
        deleteData(todo, connection);
        */

        log.info("Closing database connection");
        connection.close();
    }
}

```

Ce code Java utilisera les fichiers *application.properties* et *schema.sql* que vous avez créés précédemment pour établir la connexion à PostgreSQL Server et créer un schéma qui stockera vos données.

Dans ce fichier, vous pouvez voir que nous avons commenté les méthodes pour insérer, lire, mettre à jour et supprimer des données. Vous allez coder ces méthodes dans le reste de cet article, et vous serez en mesure de les décommenter l'une après l'autre.

ⓘ Notes

Les informations d'identification de base de données sont stockées dans les propriétés `user` et `password` du fichier *application.properties*. Ces informations d'identification sont utilisées lors de l'exécution de

`DriverManager.getConnection(properties.getProperty("url"), properties);`, car le fichier de propriétés est passé comme argument.

Vous pouvez maintenant exécuter cette classe main avec votre outil favori :

- Avec votre IDE, vous devez être en mesure de cliquer avec le bouton droit sur la classe *DemoApplication* et de l'exécuter.
- À l'aide de Maven, vous pouvez exécuter l'application avec la commande suivante :

```
mvn exec:java -Dexec.mainClass="com.example.demo.DemoApplication"
```

L'application doit se connecter à Azure Database pour PostgreSQL Database, créer un schéma de base de données, puis fermer la connexion, comme vous le verrez dans les journaux de la console :

```
Sortie

[INFO    ] Loading application properties
[INFO    ] Connecting to the database
[INFO    ] Database connection test: demo
[INFO    ] Create database schema
[INFO    ] Closing database connection
```

Créer une classe de domaine

Créez une classe Java `Todo` parallèlement à la classe `DemoApplication`, puis ajoutez le code suivant :

```
Java

package com.example.demo;

public class Todo {

    private Long id;
    private String description;
    private String details;
    private boolean done;

    public Todo() {
    }

    public Todo(Long id, String description, String details, boolean done) {
        this.id = id;
        this.description = description;
        this.details = details;
        this.done = done;
    }

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getDescription() {
        return description;
    }
```

```

    public void setDescription(String description) {
        this.description = description;
    }

    public String getDetails() {
        return details;
    }

    public void setDetails(String details) {
        this.details = details;
    }

    public boolean isDone() {
        return done;
    }

    public void setDone(boolean done) {
        this.done = done;
    }

    @Override
    public String toString() {
        return "Todo{" +
            "id=" + id +
            ", description='" + description + '\'' +
            ", details='" + details + '\'' +
            ", done=" + done +
            '}';
    }
}

```

Cette classe est un modèle de domaine mappé sur la table `todo` que vous avez créée lors de l'exécution du script `schema.sql`.

Insertion des données dans Azure Database pour PostgreSQL

Dans le fichier `src/main/java/DemoApplication.java`, après la méthode `main`, ajoutez la méthode suivante pour insérer des données dans la base de données :

Java

```

private static void insertData(Todo todo, Connection connection) throws
SQLException {
    log.info("Insert data");
    PreparedStatement insertStatement = connection
        .prepareStatement("INSERT INTO todo (id, description, details,
done) VALUES (?, ?, ?, ?);");

    insertStatement.setLong(1, todo.getId());

```

```
        insertStatement.setString(2, todo.getDescription());
        insertStatement.setString(3, todo.getDetails());
        insertStatement.setBoolean(4, todo.isDone());
        insertStatement.executeUpdate();
    }
```

Vous pouvez maintenant supprimer les marques de commentaire des deux lignes suivantes dans la méthode `main` :

Java

```
Todo todo = new Todo(1L, "configuration", "congratulations, you have set up
JDBC correctly!", true);
insertData(todo, connection);
```

L'exécution de la classe main doit maintenant produire la sortie suivante :

Sortie

```
[INFO    ] Loading application properties
[INFO    ] Connecting to the database
[INFO    ] Database connection test: demo
[INFO    ] Create database schema
[INFO    ] Insert data
[INFO    ] Closing database connection
```

Lecture de données de Azure Database pour PostgreSQL

Pour valider que votre code fonctionne correctement, lisez les données que vous avez précédemment insérées.

Dans le fichier `src/main/java/DemoApplication.java`, après la méthode `insertData`, ajoutez la méthode suivante pour lire des données de la base de données :

Java

```
private static Todo readData(Connection connection) throws SQLException {
    log.info("Read data");
    PreparedStatement readStatement = connection.prepareStatement("SELECT *
FROM todo;");
    ResultSet resultSet = readStatement.executeQuery();
    if (!resultSet.next()) {
        log.info("There is no data in the database!");
        return null;
    }
    Todo todo = new Todo();
    todo.setId(resultSet.getLong("id"));
```

```
        todo.setDescription(resultSet.getString("description"));
        todo.setDetails(resultSet.getString("details"));
        todo.setDone(resultSet.getBoolean("done"));
        log.info("Data read from the database: " + todo.toString());
        return todo;
    }
```

Vous pouvez maintenant supprimer les marques de commentaire de la ligne suivante dans la méthode `main` :

Java

```
todo = readData(connection);
```

L'exécution de la classe main doit maintenant produire la sortie suivante :

Sortie

```
[INFO    ] Loading application properties
[INFO    ] Connecting to the database
[INFO    ] Database connection test: demo
[INFO    ] Create database schema
[INFO    ] Insert data
[INFO    ] Read data
[INFO    ] Data read from the database: Todo{id=1,
description='configuration', details='congratulations, you have set up JDBC
correctly!', done=true}
[INFO    ] Closing database connection
```

Mise à jour des données dans Azure Database pour PostgreSQL

Ensuite, mettez à jour les données que vous avez insérées précédemment.

Toujours dans le fichier `src/main/java/DemoApplication.java`, après la méthode `readData`, ajoutez la méthode suivante pour mettre à jour des données dans la base de données :

Java

```
private static void updateData(Todo todo, Connection connection) throws
SQLException {
    log.info("Update data");
    PreparedStatement updateStatement = connection
        .prepareStatement("UPDATE todo SET description = ?, details = ?,
done = ? WHERE id = ?;");
```

```
        updateStatement.setString(1, todo.getDescription());
        updateStatement.setString(2, todo.getDetails());
        updateStatement.setBoolean(3, todo.isDone());
        updateStatement.setLong(4, todo.getId());
        updateStatement.executeUpdate();
        readData(connection);
    }
```

Vous pouvez maintenant supprimer les marques de commentaire des deux lignes suivantes dans la méthode `main` :

Java

```
todo.setDetails("congratulations, you have updated data!");
updateData(todo, connection);
```

L'exécution de la classe main doit maintenant produire la sortie suivante :

Sortie

```
[INFO    ] Loading application properties
[INFO    ] Connecting to the database
[INFO    ] Database connection test: demo
[INFO    ] Create database schema
[INFO    ] Insert data
[INFO    ] Read data
[INFO    ] Data read from the database: Todo{id=1,
description='configuration', details='congratulations, you have set up JDBC
correctly!', done=true}
[INFO    ] Update data
[INFO    ] Read data
[INFO    ] Data read from the database: Todo{id=1,
description='configuration', details='congratulations, you have updated
data!', done=true}
[INFO    ] Closing database connection
```

Suppression des données dans Azure Database pour PostgreSQL

Enfin, supprimez les données que vous avez insérées précédemment.

Toujours dans le fichier `src/main/java/DemoApplication.java`, après la méthode `updateData`, ajoutez la méthode suivante pour supprimer des données dans la base de données :

Java

```
private static void deleteData(Todo todo, Connection connection) throws
SQLException {
    log.info("Delete data");
    PreparedStatement deleteStatement = connection.prepareStatement("DELETE
FROM todo WHERE id = ?;");
    deleteStatement.setLong(1, todo.getId());
    deleteStatement.executeUpdate();
    readData(connection);
}
```

Vous pouvez maintenant supprimer les marques de commentaire de la ligne suivante dans la méthode `main` :

Java

```
deleteData(todo, connection);
```

L'exécution de la classe main doit maintenant produire la sortie suivante :

Sortie

```
[INFO    ] Loading application properties
[INFO    ] Connecting to the database
[INFO    ] Database connection test: demo
[INFO    ] Create database schema
[INFO    ] Insert data
[INFO    ] Read data
[INFO    ] Data read from the database: Todo{id=1,
description='configuration', details='congratulations, you have set up JDBC
correctly!', done=true}
[INFO    ] Update data
[INFO    ] Read data
[INFO    ] Data read from the database: Todo{id=1,
description='configuration', details='congratulations, you have updated
data!', done=true}
[INFO    ] Delete data
[INFO    ] Read data
[INFO    ] There is no data in the database!
[INFO    ] Closing database connection
```

Nettoyer les ressources

Félicitations ! Vous avez créé une application Java qui utilise JDBC pour stocker et récupérer des données dans Azure Database pour PostgreSQL.

Pour nettoyer toutes les ressources utilisées dans le cadre de ce guide de démarrage rapide, supprimez le groupe de ressources à l'aide de la commande suivante :

Azure CLI

```
az group delete \
--name $AZ_RESOURCE_GROUP \
--yes
```

Étapes suivantes

Migration de votre base de données PostgreSQL par exportation et importation

Envoyer et recevoir des messages à partir de files d'attente Azure Service Bus (Java)

Article • 21/04/2023

Dans ce guide de démarrage rapide, vous apprendrez à créer une application Java pour envoyer des messages à une file d'attente Azure Service Bus et en recevoir d'elle.

ⓘ Notes

Ce guide de démarrage rapide fournit des instructions pas à pas pour un scénario simple qui consiste à envoyer des messages à une file d'attente Service Bus et à les recevoir. Vous trouverez des exemples Java prédéfinis pour Azure Service Bus dans le dépôt du kit SDK Azure pour Java sur [GitHub](#).

💡 Conseil

Si vous utilisez des ressources Azure Service Bus dans une application Spring, nous vous recommandons de considérer [Spring Cloud Azure](#) comme alternative. Azure Spring Cloud est un projet open source qui fournit une intégration de Spring fluide aux services Azure. Pour en savoir plus sur Spring Cloud Azure et pour voir un exemple d'utilisation de Service Bus, consultez [Spring Cloud Stream avec Azure Service Bus](#).

Prérequis

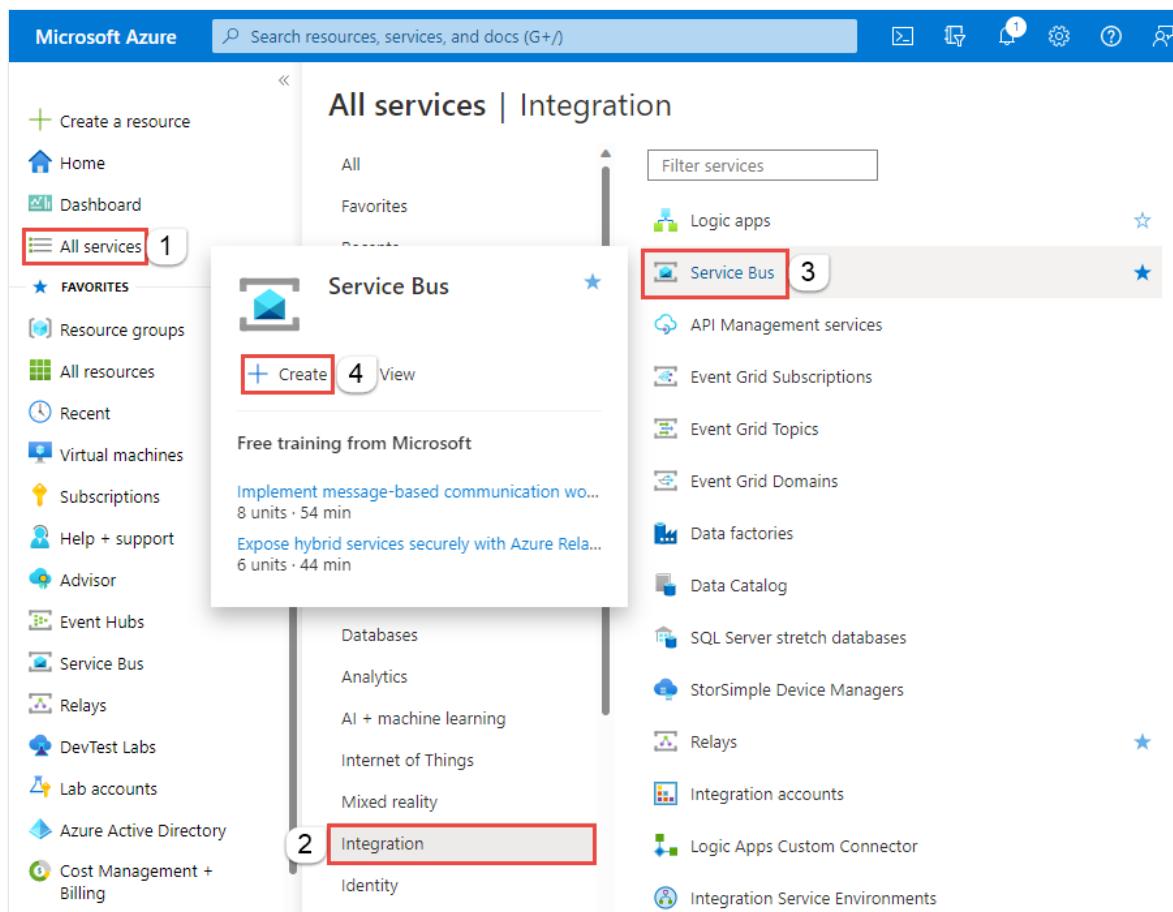
- Un abonnement Azure. Pour suivre ce tutoriel, vous avez besoin d'un compte Azure. Vous pouvez [activer les avantages de votre abonnement MSDN](#) ou [vous inscrire pour un compte gratuit](#).
- Installez le [kit de développement logiciel \(SDK\) Azure pour Java](#). Si vous utilisez Eclipse, vous pouvez installer [Azure Toolkit for Eclipse](#), qui inclut le SDK Azure pour Java. Vous pouvez ensuite ajouter les [bibliothèques Microsoft Azure pour Java](#) à votre projet. Si vous utilisez IntelliJ, consultez [Installer Azure Toolkit for IntelliJ](#).

Créer un espace de noms dans le Portail Azure

Pour commencer à utiliser des entités de messagerie Service Bus dans Azure, vous devez d'abord créer un espace de noms avec un nom unique dans Azure. Ce dernier fournit un conteneur d'étendue pour les ressources Service Bus au sein de votre application.

Pour créer un espace de noms :

1. Connectez-vous au [portail Azure](#).
2. Dans le volet de navigation gauche du portail, sélectionnez **Tous les services**, puis **Intégration** dans la liste des catégories, pointez la souris sur **Service Bus**, puis sélectionnez **Créer** dans la vignette Service Bus.



3. Dans l'étiquette **De base** de la page **Créer un espace de noms**, suivez ces étapes :
 - a. Pour l'option **Abonnement**, choisissez un abonnement Azure dans lequel créer l'espace de noms.
 - b. Pour l'option **Groupe de ressources**, choisissez un groupe de ressources existant dans lequel l'espace de noms sera utilisé, ou créez-en un nouveau.
 - c. Entrez un **nom pour l'espace de noms**. Le nom de l'espace de noms doit respecter les conventions de nommage suivantes :
 - Le nom doit être unique dans tout Azure. Le système vérifie immédiatement si le nom est disponible.

- Le nom doit inclure entre 6 et 50 caractères.
- Le nom ne peut contenir que des lettres, des chiffres et des traits d'union (« - »).
- Le nom doit commencer par une lettre, et se terminer par une lettre ou un chiffre.
- Le nom ne se termine ni par « -sb » ni par « -mgmt ».

d. Pour l'option **Emplacement**, choisissez la région dans laquelle héberger votre espace de noms.

e. Pour le **Niveau tarifaire**, sélectionnez le SKU (De base, Standard ou Premium) destiné à l'espace de noms. Pour ce guide de démarrage rapide, sélectionnez **Standard**.

 **Important**

Si vous voulez utiliser des **rubriques et des abonnements**, choisissez Standard ou Premium. Les rubriques/abonnements ne sont pas pris en charge dans le niveau tarifaire De base.

Si vous avez sélectionné le SKU **Premium**, précisez le nombre d'**unité de messagerie**. Le niveau Premium isole les ressources au niveau du processeur et de la mémoire, ce qui permet d'exécuter chaque charge de travail de manière isolée. Ce conteneur de ressources est appelé unité de messagerie. Un espace de noms Premium a au moins une unité de messagerie. Vous pouvez sélectionner 1, 2, 4, 8 ou 16 unités de messagerie pour chaque espace de noms Service Bus Premium. Pour plus d'informations, consultez [Messagerie Service Bus Premium](#).

f. Au bas de la page, sélectionnez **Examiner et créer**.

Create namespace ...

Service Bus

Basics Advanced Networking Tags Review + create

Project Details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * Visual Studio Enterprise Subscription

Resource group * (New) spsbusrg [Create new](#)

Instance Details

Enter required settings for this namespace.

Namespace name * contosoordersns [.servicebus.windows.net](#)

Location * East US

Pricing tier * Standard [Browse the available plans and their features](#)

Review + create < Previous Next: Advanced >

g. Dans la page **Vérifier + créer**, passez en revue les paramètres, puis sélectionnez **Créer**.

4. Une fois le déploiement de la ressource réussi, sélectionnez **Accéder à la ressource** dans la page de déploiement.

contosoordersns | Overview

Deployment

Search

Delete Cancel Redeploy Download Refresh

Overview

Your deployment is complete

Deployment name: contosoordersns
Subscription: Visual Studio Enterprise Subscription
Resource group: spsbusrg

Start time: 10/20/2022, 4:45:03 PM
Correlation ID: a453ace1-bab9-4c4a-81ad-a1c5366460ea

Inputs
Outputs
Template

Deployment details
Next steps

Go to resource

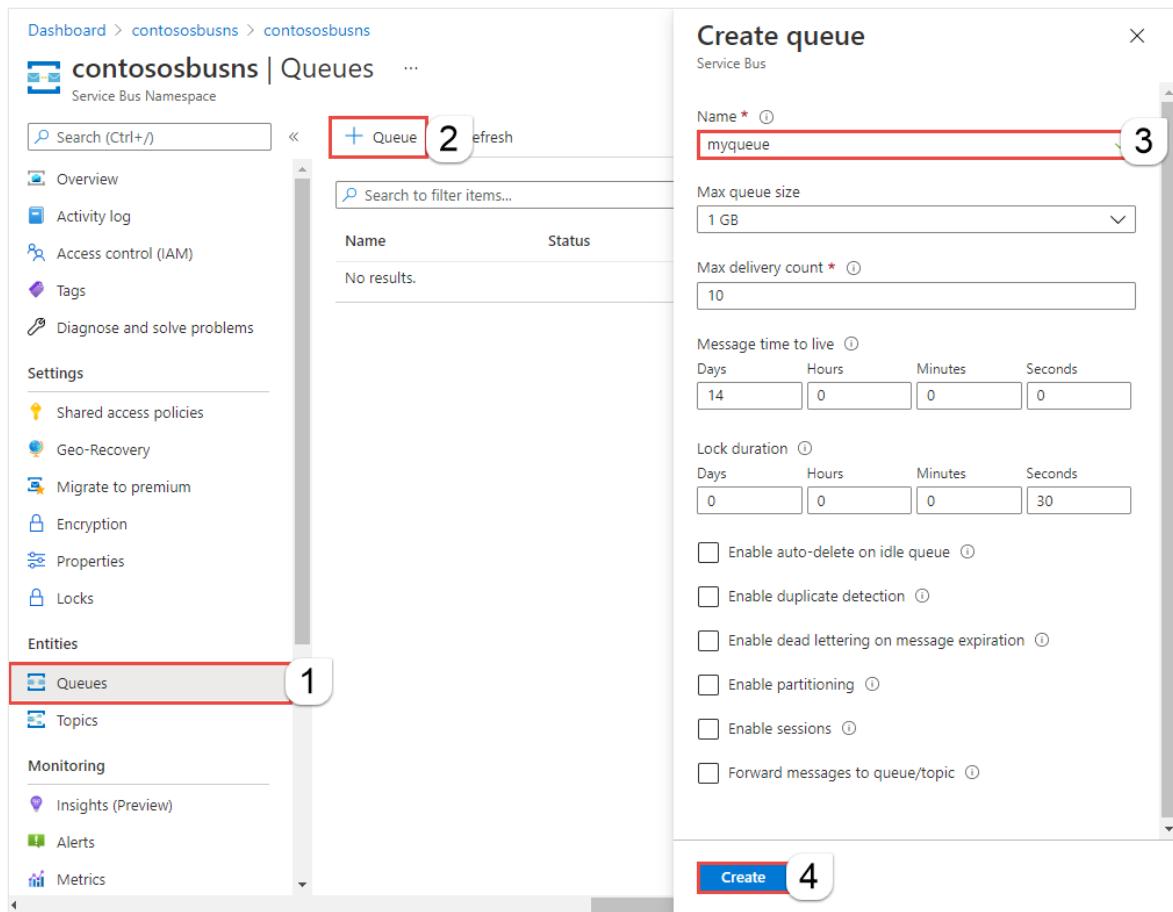
Give feedback
Tell us about your experience with deployment

5. Vous voyez la page d'accueil de votre espace de noms Service Bus.

The screenshot shows the Azure Service Bus Namespace overview page for the namespace `spsbusns1028`. The left sidebar contains navigation links for Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Settings (Shared access policies, Geo-Recovery, Migrate to premium, Encryption, Configuration, Properties, Locks), Entities (Queues, Topics), Monitoring (Insights (Preview), Alerts, Metrics, Diagnostic settings, Logs), and JSON View. The main content area displays the namespace's details: Resource group ([move](#)) `spsbusrg`, Status Active, Location East US, Subscription ([move](#)) `Visual Studio Enterprise Subscription`, Subscription ID `00000000-0000-0000-0000-000000000000`, Tags ([edit](#)), and Click here to add tags. It also shows monitoring data for Requests and Messages over the last hour, with zero activity shown.

Créer une file d'attente dans le portail Azure

1. Dans la page Espace de noms Service Bus, sélectionnez **Files d'attente** dans le menu de navigation de gauche.
2. Dans la page **Files d'attente**, sélectionnez + **File d'attente** dans la barre d'outils.
3. Entrez un **nom** pour la file d'attente et laissez les valeurs par défaut des autres valeurs.
4. À présent, sélectionnez **Créer**.



Authentifier l'application sur Azure

Ce guide de démarrage pratique vous montre deux façons de vous connecter à Azure Service Bus : **sans mot de passe** et avec une **chaîne de connexion**.

La première option vous explique comment utiliser votre principal de sécurité dans Microsoft Entra ID et le contrôle d'accès en fonction du rôle (RBAC) pour vous connecter à un espace de noms Service Bus. Vous n'avez pas à vous soucier d'avoir une chaîne de connexion codée en dur dans votre code, dans un fichier config ni dans un stockage sécurisé comme Azure Key Vault.

La deuxième option consiste à se servir d'une chaîne de connexion pour se connecter à un espace de noms Service Bus. Si vous débutez avec Azure, vous trouverez peut-être l'option chaîne de connexion plus facile à suivre. Nous vous recommandons d'utiliser l'option sans mot de passe dans les applications réelles et les environnements de production. Pour plus d'informations, consultez [Authentification et autorisation](#). Pour en savoir plus sur l'authentification sans mot de passe, reportez-vous à la [page de présentation](#).

Sans mot de passe (recommandé)

Attribuer des rôles à votre utilisateur Microsoft Entra

Lors du développement localement, assurez-vous que le compte d'utilisateur qui se connecte à Azure Service Bus dispose des autorisations appropriées. Vous aurez besoin du rôle [Propriétaire de données Azure Service Bus](#) pour envoyer et recevoir des messages. Pour vous attribuer ce rôle, vous aurez besoin du rôle Administrateur de l'accès utilisateur ou d'un autre rôle qui inclut l'action

`Microsoft.Authorization/roleAssignments/write`. Vous pouvez attribuer des rôles RBAC Azure à un utilisateur à l'aide du Portail Azure, Azure CLI ou Azure PowerShell. Découvrez les étendues disponibles pour les attributions de rôles dans la page [vue d'ensemble de l'étendue](#).

L'exemple suivant attribue le rôle `Azure Service Bus Data Owner` à votre compte d'utilisateur, qui fournit un accès complet aux ressources Azure Service Bus. Dans un scénario réel, suivez le [principe des privilèges minimum](#) pour accorder aux utilisateurs uniquement les autorisations minimales nécessaires à un environnement de production plus sécurisé.

Rôles Azure intégrés pour Azure Service Bus

Pour Azure Service Bus, la gestion des espaces de noms et de toutes les ressources associées via le Portail Azure et l'API de gestion des ressources Azure est déjà protégée à l'aide du modèle Azure RBAC. Azure fournit les rôles Azure intégrés ci-dessous pour autoriser l'accès à un espace de noms Service Bus :

- [Propriétaire de données Azure Service Bus](#) : ce rôle permet l'accès aux données de l'espace de noms Service Bus et de ses entités (files d'attente, rubriques, abonnements et filtres). Un membre de ce rôle peut envoyer et recevoir des messages à partir de files d'attente ou de rubriques et d'abonnements.
- [Expéditeur de données Azure Service Bus](#) : utilisez ce rôle pour autoriser l'accès en envoi à l'espace de noms Service Bus et à ses entités.
- [Récepteur de données Azure Service Bus](#) : utilisez ce rôle pour autoriser l'accès en réception à l'espace de noms Service Bus et à ses entités.

Si vous souhaitez créer un rôle personnalisé, consultez [Droits requis pour les opérations Service Bus](#).

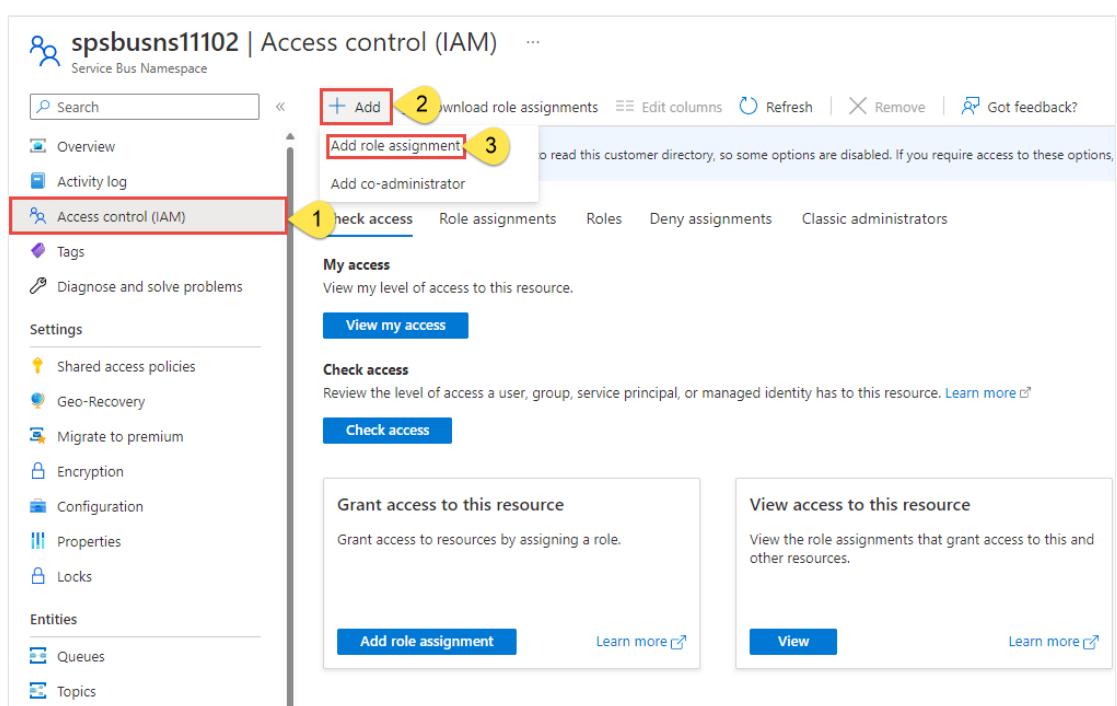
Ajouter un utilisateur Microsoft Entra au rôle Propriétaire Azure Service Bus

Ajoutez votre nom d'utilisateur Microsoft Entra au rôle **Propriétaire de données** Azure Service Bus au niveau de l'espace de noms Service Bus. Il permet à une application exécutée dans le contexte de votre compte d'utilisateur d'envoyer des messages à une file d'attente ou à une rubrique et d'en recevoir auprès d'une file d'attente ou de l'abonnement d'une rubrique.

ⓘ Important

Dans la plupart des cas, la propagation de l'attribution de rôle dans Azure peut prendre une ou deux minutes. Dans de rares cas, cela peut prendre jusqu'à **huit minutes**. Si vous recevez des erreurs d'authentification lorsque vous exécutez votre code pour la première fois, patientez quelques instants et réessayez.

1. Si la page Espace de noms Service Bus n'est pas ouverte sur le Portail Azure, recherchez votre espace de noms Service Bus à l'aide de la barre de recherche principale ou du volet de navigation de gauche.
2. Dans la page vue d'ensemble, sélectionnez **Contrôle d'accès (IAM)** dans le menu de gauche.
3. Sur la page **Contrôle d'accès (IAM)**, sélectionnez l'onglet **Attributions de rôles**.
4. Sélectionnez **+ Ajouter** dans le menu supérieur, puis **Ajouter une attribution de rôle** dans le menu déroulant résultant.



5. Utilisez la zone de recherche pour filtrer les résultats sur le rôle souhaité. Pour cet exemple, recherchez `Azure Service Bus Data Owner` et sélectionnez le résultat correspondant. Ensuite, choisissez **Suivant**.
6. Sous **Attribuer l'accès à**, sélectionnez **Utilisateur, groupe ou principal de service**, puis sélectionnez + **Sélectionner des membres**.
7. Dans la boîte de dialogue, recherchez votre nom d'utilisateur Microsoft Entra (généralement votre adresse e-mail `user@domain`), puis choisissez **Sélectionner** en bas de la boîte de dialogue.
8. Sélectionnez **Vérifier + affecter** pour accéder à la page finale, puis **Vérifier + attribuer** à nouveau pour terminer le processus.

Envoi de messages à une file d'attente

Dans cette section, vous allez apprendre à créer un projet de console Java et ajouter du code pour envoyer des messages à la file d'attente que vous avez créée.

Créer un projet de console Java

Créez un projet Java avec Eclipse ou un outil de votre choix.

Configuration de votre application pour l'utilisation de Service Bus

Ajoutez des références aux bibliothèques Azure Core et Azure Service Bus.

Si vous utilisez Eclipse et avez créé une application de console Java, convertissez votre projet Java en Maven : faites un clic droit sur le projet dans la fenêtre **Explorateur de package**, puis sélectionnez **Configurer – >Convertir en projet Maven**. Ajoutez ensuite des dépendances à ces deux bibliothèques, comme indiqué dans l'exemple suivant.

Sans mot de passe (recommandé)

Mettez à jour le fichier `pom.xml` pour ajouter des dépendances aux packages Azure Service Bus et Azure Identity.

XML

```
<dependencies>
  <dependency>
    <groupId>com.azure</groupId>
    <artifactId>azure-messaging-servicebus</artifactId>
    <version>7.13.3</version>
  </dependency>
  <dependency>
    <groupId>com.azure</groupId>
    <artifactId>azure-identity</artifactId>
    <version>1.8.0</version>
    <scope>compile</scope>
  </dependency>
</dependencies>
```

Ajouter du code pour envoyer des messages à la file d'attente

1. Ajoutez les instructions `import` suivantes au début du fichier Java.

Sans mot de passe (recommandé)

Java

```
import com.azure.messaging.servicebus.*;
import com.azure.identity.*;

import java.util.concurrent.CountDownLatch;
import java.util.concurrent.TimeUnit;
import java.util.Arrays;
import java.util.List;
```

2. Dans la classe, définissez des variables pour stocker la chaîne de connexion et le nom de la file d'attente.

Sans mot de passe (recommandé)

Java

```
static String queueName = "<QUEUE NAME>";
```

 **Important**

et remplacez <QUEUE NAME> par le nom de la file d'attente.

3. Ajoutez une méthode nommée `sendMessage` dans la classe pour envoyer un message à la file d'attente.

Sans mot de passe (recommandé)

ⓘ Important

Remplacez `NAMESPACENAME` par le nom de votre espace de noms Service Bus.

Java

```
static void sendMessage()
{
    // create a token using the default Azure credential
    DefaultAzureCredential credential = new
DefaultAzureCredentialBuilder()
    .build();

    ServiceBusSenderClient senderClient = new
ServiceBusClientBuilder()

    .fullyQualifiedNamespace("NAMESPACENAME.servicebus.windows.net")
        .credential(credential)
        .sender()
        .queueName(queueName)
        .buildClient();

    // send one message to the queue
    senderClient.sendMessage(new ServiceBusMessage("Hello,
World!"));
    System.out.println("Sent a single message to the queue: " +
queueName);
}
```

4. Ajoutez une méthode nommée `createMessages` dans la classe pour créer une liste de messages. En général, vous recevez ces messages à partir de différentes parties de votre application. Ici, nous créons une liste d'exemples de messages.

Java

```

static List<ServiceBusMessage> createMessages()
{
    // create a list of messages and return it to the caller
    ServiceBusMessage[] messages = {
        new ServiceBusMessage("First message"),
        new ServiceBusMessage("Second message"),
        new ServiceBusMessage("Third message")
    };
    return Arrays.asList(messages);
}

```

5. Ajoutez une méthode nommée `sendMessageBatch` pour envoyer des messages à la file d'attente que vous avez créée. Cette méthode crée un `ServiceBusSenderClient` pour la file d'attente, appelle la méthode `createMessages` pour obtenir la liste des messages, prépare un ou plusieurs lots, et envoie les lots à la file d'attente.

Sans mot de passe (recommandé)

i Important

Remplacez `NAMESPACENAME` par le nom de votre espace de noms Service Bus.

Java

```

static void sendMessageBatch()
{
    // create a token using the default Azure credential
    DefaultAzureCredential credential = new
DefaultAzureCredentialBuilder()
    .build();

    ServiceBusSenderClient senderClient = new
ServiceBusClientBuilder()

    .fullyQualifiedNamespace("NAMESPACENAME.servicebus.windows.net")
        .credential(credential)
        .sender()
        .queueName(queueName)
    .buildClient();

    // Creates an ServiceBusMessageBatch where the ServiceBus.
    ServiceBusMessageBatch messageBatch =
senderClient.createMessageBatch();

    // create a list of messages
    List<ServiceBusMessage> listOfMessages = createMessages();
}

```

```

    // We try to add as many messages as a batch can fit based on
    the maximum size and send to Service Bus when
    // the batch can hold no more messages. Create a new batch for
    next set of messages and repeat until all
    // messages are sent.
    for (ServiceBusMessage message : listOfMessages) {
        if (messageBatch.tryAddMessage(message)) {
            continue;
        }

        // The batch is full, so we create a new batch and send the
        batch.
        senderClient.sendMessages(messageBatch);
        System.out.println("Sent a batch of messages to the queue:
" + queueName);

        // create a new batch
        messageBatch = senderClient.createMessageBatch();

        // Add that message that we couldn't before.
        if (!messageBatch.tryAddMessage(message)) {
            System.err.printf("Message is too large for an empty
batch. Skipping. Max size: %s.", messageBatch.getMaxSizeInBytes());
        }
    }

    if (messageBatch.getCount() > 0) {
        senderClient.sendMessages(messageBatch);
        System.out.println("Sent a batch of messages to the queue:
" + queueName);
    }

    //close the client
    senderClient.close();
}

```

Réception des messages d'une file d'attente

Dans cette section, vous allez ajouter du code pour récupérer les messages de la file d'attente.

1. Ajoutez une méthode nommée `receiveMessages` pour recevoir des messages à partir de la file d'attente. Cette méthode crée un `ServiceBusProcessorClient` pour la file d'attente en spécifiant un gestionnaire pour le traitement des messages et un autre pour la gestion des erreurs. Ensuite, elle démarre le processeur, attend quelques secondes, imprime les messages reçus, puis arrête et ferme le processeur.

Sans mot de passe (recommandé)

ⓘ Important

- Remplacez `NAMESPACE` par le nom de votre espace de noms Service Bus.
- Dans le code, remplacez `QueueTest` dans `QueueTest::processMessage` par le nom de votre classe.

Java

```
// handles received messages
static void receiveMessages() throws InterruptedException
{
    CountDownLatch countdownLatch = new CountDownLatch(1);

    DefaultAzureCredential credential = new
DefaultAzureCredentialBuilder()
        .build();

    ServiceBusProcessorClient processorClient = new
ServiceBusClientBuilder()

    .fullyQualifiedNamespace("NAMESPACE.servicebus.windows.net")
        .credential(credential)
        .processor()
        .queueName(queueName)
        .processMessage(QueueTest::processMessage)
        .processError(context -> processError(context,
countdownLatch))
        .buildProcessorClient();

    System.out.println("Starting the processor");
    processorClient.start();

    TimeUnit.SECONDS.sleep(10);
    System.out.println("Stopping and closing the processor");
    processorClient.close();
}
```

2. Ajoutez la méthode `processMessage` pour traiter un message reçu de l'abonnement Service Bus.

Java

```

private static void processMessage(ServiceBusReceivedMessageContext context) {
    ServiceBusReceivedMessage message = context.getMessage();
    System.out.printf("Processing message. Session: %s, Sequence #: %s.
Contents: %s%n", message.getMessageId(),
    message.getSequenceNumber(), message.getBody());
}

```

3. Ajoutez la méthode `processError` pour gérer les messages d'erreur.

Java

```

private static void processError(ServiceBusErrorHandlerContext context,
CountDownLatch countdownLatch) {
    System.out.printf("Error when receiving messages from namespace:
'%s'. Entity: '%s'%n",
        context.getFullyQualifiedNamespace(), context.getEntityPath());

    if (!(context.getException() instanceof ServiceBusException)) {
        System.out.printf("Non-ServiceBusException occurred: %s%n",
            context.getException());
        return;
    }

    ServiceBusException exception = (ServiceBusException)
context.getException();
    ServiceBusFailureReason reason = exception.getReason();

    if (reason == ServiceBusFailureReason.MESSAGING_ENTITY_DISABLED
        || reason == ServiceBusFailureReason.MESSAGING_ENTITY_NOT_FOUND
        || reason == ServiceBusFailureReason.UNAUTHORIZED) {
        System.out.printf("An unrecoverable error occurred. Stopping
processing with reason %s: %s%n",
            reason, exception.getMessage());

        countdownLatch.countDown();
    } else if (reason == ServiceBusFailureReason.MESSAGE_LOCK_LOST) {
        System.out.printf("Message lock lost for message: %s%n",
            context.getException());
    } else if (reason == ServiceBusFailureReason.SERVICE_BUSY) {
        try {
            // Choosing an arbitrary amount of time to wait until
trying again.
            TimeUnit.SECONDS.sleep(1);
        } catch (InterruptedException e) {
            System.err.println("Unable to sleep for period of time");
        }
    } else {
        System.out.printf("Error source %s, reason %s, message: %s%n",
            context.getErrorSource(),
            reason, context.getException());
    }
}

```

```
}
```

4. Mettez à jour la méthode `main` pour appeler les méthodes `sendMessage`, `sendMessageBatch` et `receiveMessages`, et pour lever `InterruptedException`.

```
Java
```

```
public static void main(String[] args) throws InterruptedException {  
    sendMessage();  
    sendMessageBatch();  
    receiveMessages();  
}
```

Exécuter l'application

Sans mot de passe (recommandé)

1. Si vous utilisez Eclipse, faites un clic droit sur le projet, sélectionnez **Exporter**, développez **Java**, sélectionnez **Fichier JAR exécutable**, puis suivez les étapes pour créer un fichier JAR exécutable.
2. Si vous êtes connecté à l'ordinateur à l'aide d'un compte d'utilisateur différent du compte d'utilisateur ajouté au rôle **propriétaire des données Azure Service Bus**, suivez les étapes suivantes. Sinon, ignorez cette étape et passez à l'exécution du fichier JAR à l'étape suivante.
 - a. [Installez Azure CLI](#) sur votre machine.
 - b. Exécutez la commande CLI suivante pour vous connecter à Azure. Utilisez le même compte d'utilisateur que celui que vous avez ajouté au rôle **propriétaire des données Azure Service Bus**.

```
Azure CLI
```

```
az login
```

3. Exécutez le fichier JAR à l'aide de la commande suivante.

```
Java
```

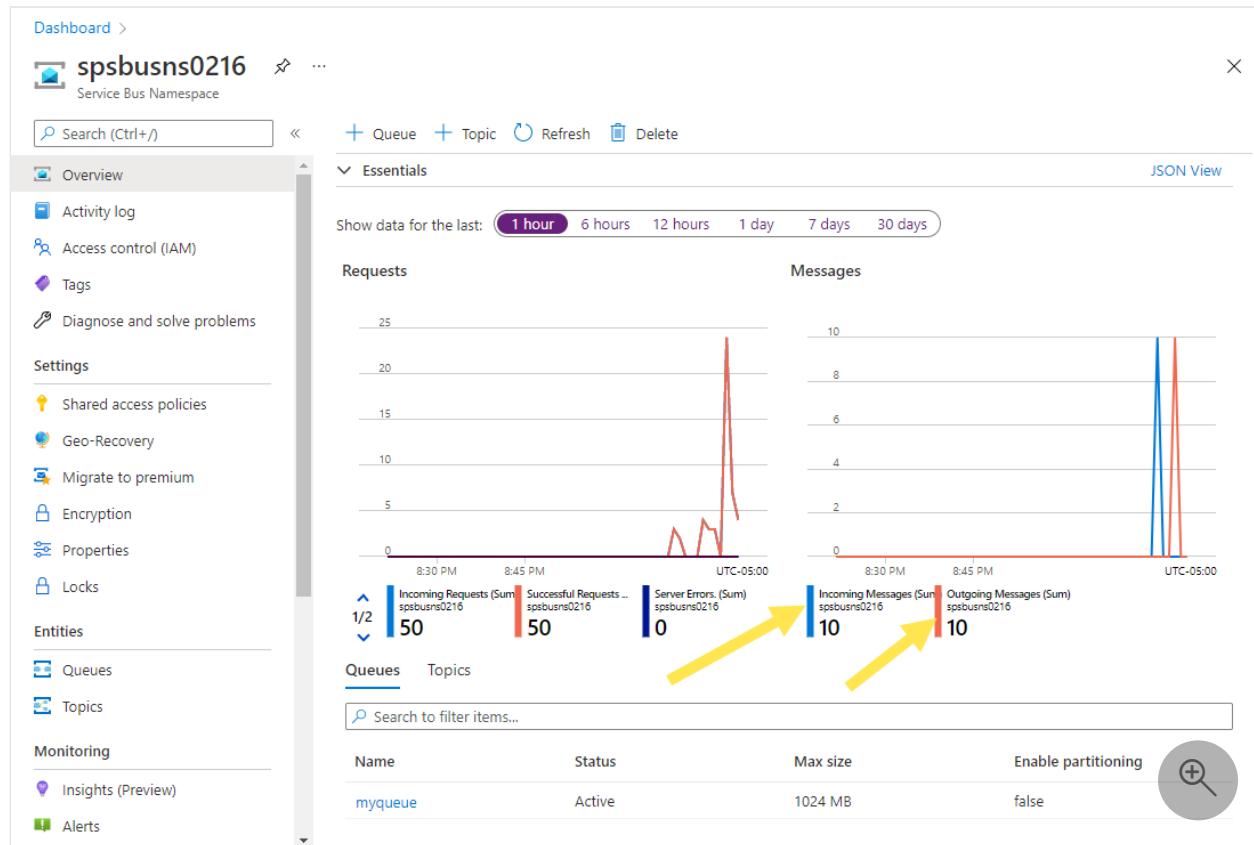
```
java -jar <JAR FILE NAME>
```

4. Vous voyez la sortie suivante dans la fenêtre de console.

```
Console

Sent a single message to the queue: myqueue
Sent a batch of messages to the queue: myqueue
Starting the processor
Processing message. Session: 88d961dd801f449e9c3e0f8a5393a527,
Sequence #: 1. Contents: Hello, World!
Processing message. Session: e90c8d9039ce403bbe1d0ec7038033a0,
Sequence #: 2. Contents: First message
Processing message. Session: 311a216a560c47d184f9831984e6ac1d,
Sequence #: 3. Contents: Second message
Processing message. Session: f9a871be07414baf9505f2c3d466c4ab,
Sequence #: 4. Contents: Third message
Stopping and closing the processor
```

Dans la page **Vue d'ensemble** de l'espace de noms Service Bus dans le portail Azure, vous pouvez voir le nombre de messages **entrants et sortants**. Vous devrez peut-être attendre environ une minute puis actualiser la page pour voir les valeurs les plus récentes.



Sélectionnez la file d'attente dans cette page **Vue d'ensemble** pour accéder à la page **File d'attente Service Bus**. Le nombre de messages **entrants et sortants** est visible dans

cette page, ainsi que d'autres informations telles que la **taille actuelle** de la file d'attente, la **taille maximale** et le **nombre de messages actifs**.

The screenshot shows the Azure Service Bus Queue 'myqueue' dashboard. At the top, it displays basic queue properties: Max delivery count (10), Current size (0.0 KB), Max size (1 GB), Message time to live (14 days), Auto-delete (NEVER), Message lock duration (30 seconds), and Free space (100.0%). Below this, the 'MESSAGE COUNTS' section shows Active (0 messages), Scheduled (0 messages), Dead-letter (0 messages), Transfer (0 messages), and Transfer dead-letter (0 messages). The 'Metrics' section has two charts: 'Requests' and 'Messages'. The 'Requests' chart shows Incoming Requests (Sum) spibusns0216 at 24, Successful Requests - spibusns0216 at 24, and Server Errors (Sum) spibusns0216 at 0. The 'Messages' chart shows Incoming Messages (Sum) spibusns0216 at 10 and Outgoing Messages (Sum) spibusns0216 at 10. Two yellow arrows point to the 'Incoming Messages' and 'Outgoing Messages' values, which are both highlighted in red. A magnifying glass icon is also present over the 'Outgoing Messages' value.

Étapes suivantes

Voir la documentation et les exemples suivants :

- [Bibliothèque de client Azure Service Bus pour Java - Lisez-moi ↗](#)
- [Exemples sur GitHub](#)
- [Informations de référence sur l'API Java ↗](#)

Envoyer des messages à une rubrique Azure Service Bus et recevoir des messages à partir d'abonnements à la rubrique (Java)

Article • 20/04/2023

Dans ce guide de démarrage rapide, vous allez écrire du code Java à l'aide du package `azure-messaging-servicebus` pour envoyer des messages à une rubrique Azure Service Bus, puis vous recevrez les messages des abonnements à cette rubrique.

ⓘ Notes

Ce guide de démarrage rapide fournit des instructions pas à pas pour un scénario simple qui consiste à envoyer un lot de messages à une rubrique Service Bus et à recevoir ces messages à partir d'un abonnement de la rubrique. Vous trouverez des exemples Java prédéfinis pour Azure Service Bus dans le dépôt du kit SDK Azure pour Java sur GitHub [GitHub](#).

💡 Conseil

Si vous utilisez des ressources Azure Service Bus dans une application Spring, nous vous recommandons de considérer **Spring Cloud Azure** comme alternative. Azure Spring Cloud est un projet open source qui fournit une intégration de Spring fluide aux services Azure. Pour en savoir plus sur Spring Cloud Azure et pour voir un exemple d'utilisation de Service Bus, consultez **Spring Cloud Stream avec Azure Service Bus**.

Prérequis

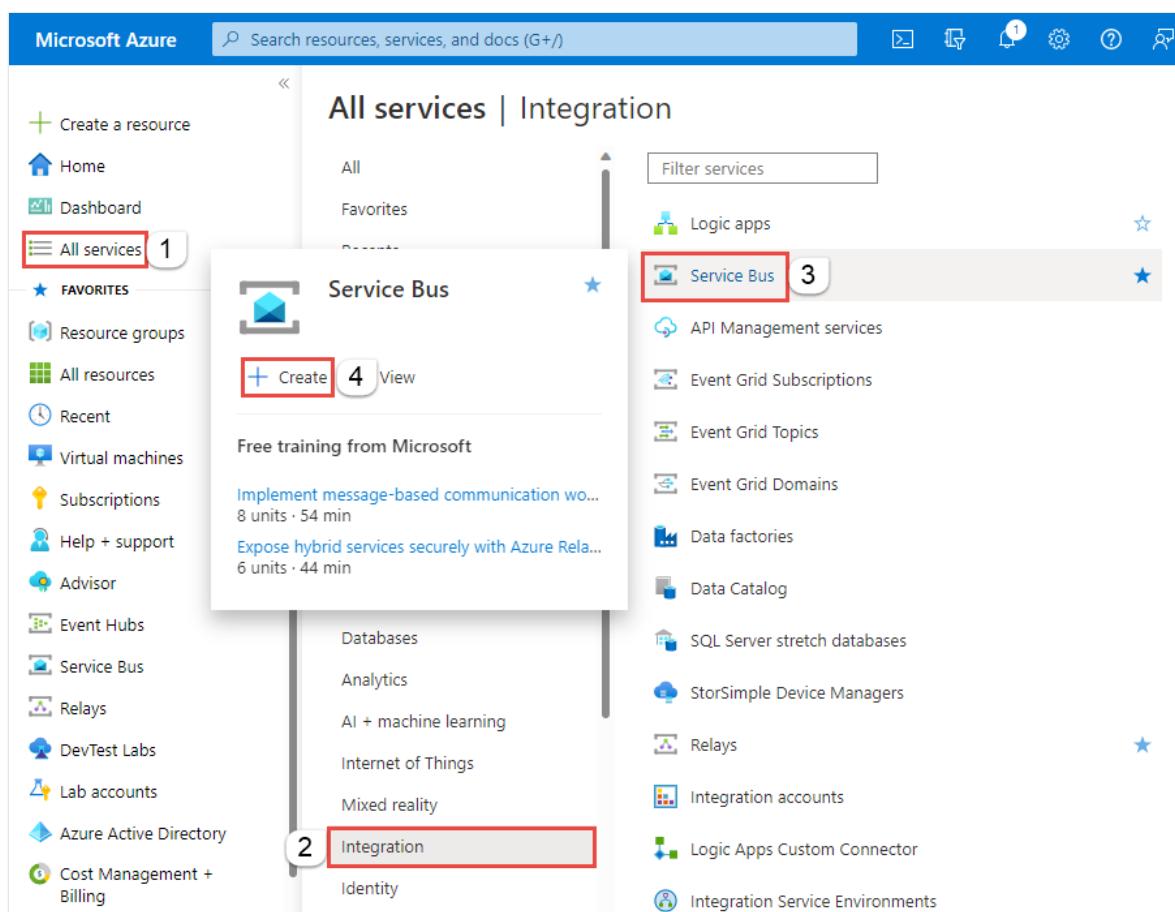
- Un abonnement Azure. Pour suivre ce tutoriel, vous avez besoin d'un compte Azure. Vous pouvez activer les [avantages de votre abonnement Visual Studio ou MSDN](#) ou vous inscrire pour créer un [compte gratuit](#).
- Installez le [kit de développement logiciel \(SDK\) Azure pour Java](#). Si vous utilisez Eclipse, vous pouvez installer [Azure Toolkit for Eclipse](#), qui inclut le SDK Azure pour Java. Vous pouvez ensuite ajouter les [bibliothèques Microsoft Azure pour Java](#) à votre projet. Si vous utilisez IntelliJ, consultez [Installer Azure Toolkit for IntelliJ](#).

Créer un espace de noms dans le Portail Azure

Pour commencer à utiliser des entités de messagerie Service Bus dans Azure, vous devez d'abord créer un espace de noms avec un nom unique dans Azure. Ce dernier fournit un conteneur d'étendue pour les ressources Service Bus au sein de votre application.

Pour créer un espace de noms :

1. Connectez-vous au [portail Azure](#).
2. Dans le volet de navigation gauche du portail, sélectionnez **Tous les services**, puis **Intégration** dans la liste des catégories, pointez la souris sur **Service Bus**, puis sélectionnez **Créer** dans la vignette Service Bus.



3. Dans l'étiquette De base de la page **Créer un espace de noms**, suivez ces étapes :
 - a. Pour l'option **Abonnement**, choisissez un abonnement Azure dans lequel créer l'espace de noms.
 - b. Pour l'option **Groupe de ressources**, choisissez un groupe de ressources existant dans lequel l'espace de noms sera utilisé, ou créez-en un nouveau.
 - c. Entrez un **nom pour l'espace de noms**. Le nom de l'espace de noms doit respecter les conventions de nommage suivantes :

- Le nom doit être unique dans tout Azure. Le système vérifie immédiatement si le nom est disponible.
 - Le nom doit inclure entre 6 et 50 caractères.
 - Le nom ne peut contenir que des lettres, des chiffres et des traits d'union (« - »).
 - Le nom doit commencer par une lettre, et se terminer par une lettre ou un chiffre.
 - Le nom ne se termine ni par « -sb » ni par « -mgmt ».
- d. Pour l'option **Emplacement**, choisissez la région dans laquelle héberger votre espace de noms.
- e. Pour le **Niveau tarifaire**, sélectionnez le SKU (De base, Standard ou Premium) destiné à l'espace de noms. Pour ce guide de démarrage rapide, sélectionnez **Standard**.

 **Important**

Si vous voulez utiliser des **rubriques et des abonnements**, choisissez Standard ou Premium. Les rubriques/abonnements ne sont pas pris en charge dans le niveau tarifaire De base.

Si vous avez sélectionné le SKU **Premium**, précisez le nombre d'**unité de messagerie**. Le niveau Premium isole les ressources au niveau du processeur et de la mémoire, ce qui permet d'exécuter chaque charge de travail de manière isolée. Ce conteneur de ressources est appelé unité de messagerie. Un espace de noms Premium a au moins une unité de messagerie. Vous pouvez sélectionner 1, 2, 4, 8 ou 16 unités de messagerie pour chaque espace de noms Service Bus Premium. Pour plus d'informations, consultez [Messagerie Service Bus Premium](#).

- f. Au bas de la page, sélectionnez **Examiner et créer**.

Create namespace ...

Service Bus

Basics Advanced Networking Tags Review + create

Project Details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * Visual Studio Enterprise Subscription

Resource group * (New) spsbusrg Create new

Instance Details

Enter required settings for this namespace.

Namespace name * contosoordersns .servicebus.windows.net

Location * East US

Pricing tier * Standard Browse the available plans and their features

Review + create < Previous Next: Advanced >

g. Dans la page Vérifier + créer, passez en revue les paramètres, puis sélectionnez **Créer**.

4. Une fois le déploiement de la ressource réussi, sélectionnez **Accéder à la ressource** dans la page de déploiement.

contosoordersns | Overview Deployment

Search < Delete Cancel Redeploy Download Refresh

Overview Deployment name: contosoordersns Start time: 10/20/2022, 4:45:03 PM

Inputs Deployment name: contosoordersns Subscription: Visual Studio Enterprise Subscription Correlation ID: a453ace1-bab9-4c4a-81ad-a1c5366460ea

Outputs Resource group: spsbusrg

Template Deployment details

Deployment details

Next steps

Go to resource

Give feedback

Tell us about your experience with deployment

5. Vous voyez la page d'accueil de votre espace de noms Service Bus.

The screenshot shows the Azure Service Bus Namespace overview page for the namespace `spsbusns1028`. The left sidebar contains navigation links for Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Settings (Shared access policies, Geo-Recovery, Migrate to premium, Encryption, Configuration, Properties, Locks), Entities (Queues, Topics), Monitoring (Insights (Preview), Alerts, Metrics, Diagnostic settings, Logs), and JSON View. The main content area displays the namespace's details under the **Essentials** section, including Resource group ([move](#)), Status (Active), Location (East US), Subscription ([move](#)), Subscription ID (00000000-0000-0000-0000-000000000000), Tags ([edit](#)), and Click here to add tags. It also shows monitoring data for Requests and Messages over the last hour, with zero activity shown. A search bar at the top left and a refresh button are also visible.

Créer une rubrique à l'aide du Portail Azure

1. Dans la page **Espace de noms Service Bus**, sélectionnez **Rubriques** dans le menu de gauche.
2. Sélectionnez **+ Rubrique** dans la barre d'outils.
3. Entrez un **nom** pour la rubrique. Conservez les valeurs par défaut des autres options.
4. Cliquez sur **Créer**.

All services > spsbusns1028 | Overview > spsbusns1028

spsbusns1028 | Topics ...

Service Bus Namespace

Search 2 + Topic Refresh

Settings

- Shared access policies
- Geo-Recovery
- Migrate to premium
- Encryption
- Configuration
- Properties
- Locks

Entities

- Queues
- Topics** 1

Monitoring

- Insights (Preview)
- Alerts
- Metrics
- Diagnostic settings

Create topic

Service Bus

Name * ① 3 mytopic ✓

Max topic size ① 1 GB

Message time to live ①

Days	Hours	Minutes	Seconds
14	0	0	0

Enable auto-delete on idle topic ①

Enable duplicate detection ①

Enable partitioning ①

4 Create

Créer un abonnement à la rubrique

1. Sélectionnez la **rubrique** que vous avez créée dans la section précédente.

All services > spsbusns1028 | Overview > spsbusns1028

spsbusns1028 | Topics ...

Service Bus Namespace

Search 2 + Topic Refresh

Settings

- Shared access policies
- Geo-Recovery
- Migrate to premium
- Encryption
- Configuration
- Properties
- Locks

Entities

- Queues
- Topics**

Monitoring

- Insights (Preview)
- Alerts
- Metrics
- Diagnostic settings

Name	Status	Max size	Subscription count	Enable partitioning
mytopic	Active	1024 MB	0	false

2. Dans la page **Rubrique Service Bus**, dans la barre d'outils, sélectionnez **+ Abonnement**.

The screenshot shows the Azure portal interface for a Service Bus Topic named 'mytopic'. The top navigation bar includes 'Subscriptions', a star icon, and three dots. On the left, there's a sidebar with links like 'Overview', 'Access control (IAM)', 'Diagnose and solve problems', 'Service Bus Explorer', 'Settings' (with 'Shared access policies', 'Properties', and 'Locks'), 'Entities' (with 'Subscriptions' highlighted by a red box), 'Automation' (with 'Tasks (preview)' and 'Export template'), and 'Support + troubleshooting' (with 'New Support Request'). The main content area has a search bar and a table with columns 'Name', 'Status', 'Message count', 'Max delivery count', and 'Client Scoped'. A message states 'No results.' Below the table is a horizontal scrollbar.

3. Dans la page **Créer un abonnement**, procédez comme suit :

- a. Entrez **S1** pour le **nom** de l'abonnement.
- b. Entrez **3** pour **Nombre maximal de remises**.
- c. Ensuite, sélectionnez **Créer** pour créer l'abonnement.

Create subscription

X

Service Bus

Name * ⓘ

S1



Max delivery count * ⓘ

3



Auto-delete after idle for ⓘ

Days

Hours

Minutes

Seconds

14

0

0

0

Never auto-delete

Forward messages to queue/topic ⓘ

MESSAGE SESSIONS

Service bus sessions allow ordered handling of unbounded sequences of related messages. With sessions enabled a subscription can guarantee first-in-first-out delivery of messages. [Learn more.](#)

Enable sessions

MESSAGE TIME TO LIVE AND DEAD-LETTERING

Message time to live (default) ⓘ

Days

Hours

Minutes

Seconds

14

0

0

0

Enable dead lettering on message expiration

Move messages that cause filter evaluation exceptions to the dead-letter subqueue

MESSAGE LOCK DURATION

Lock duration ⓘ

Days

Hours

Minutes

Seconds

0

0

0

30

Create

Authentifier l'application sur Azure

Ce guide de démarrage pratique vous montre deux façons de vous connecter à Azure Service Bus : **sans mot de passe** et avec une **chaîne de connexion**.

La première option vous explique comment utiliser votre principal de sécurité dans Microsoft Entra ID et le contrôle d'accès en fonction du rôle (RBAC) pour vous connecter à un espace de noms Service Bus. Vous n'avez pas à vous soucier d'avoir une chaîne de connexion codée en dur dans votre code, dans un fichier config ni dans un stockage sécurisé comme Azure Key Vault.

La deuxième option consiste à se servir d'une chaîne de connexion pour se connecter à un espace de noms Service Bus. Si vous débutez avec Azure, vous trouverez peut-être l'option chaîne de connexion plus facile à suivre. Nous vous recommandons d'utiliser l'option sans mot de passe dans les applications réelles et les environnements de production. Pour plus d'informations, consultez [Authentification et autorisation](#). Pour en savoir plus sur l'authentification sans mot de passe, reportez-vous à la [page de présentation](#).

Sans mot de passe (recommandé)

Attribuer des rôles à votre utilisateur Microsoft Entra

Lors du développement localement, assurez-vous que le compte d'utilisateur qui se connecte à Azure Service Bus dispose des autorisations appropriées. Vous aurez besoin du rôle [Propriétaire de données Azure Service Bus](#) pour envoyer et recevoir des messages. Pour vous attribuer ce rôle, vous aurez besoin du rôle Administrateur de l'accès utilisateur ou d'un autre rôle qui inclut l'action

`Microsoft.Authorization/roleAssignments/write`. Vous pouvez attribuer des rôles RBAC Azure à un utilisateur à l'aide du Portail Azure, Azure CLI ou Azure PowerShell. Découvrez les étendues disponibles pour les attributions de rôles dans la page [vue d'ensemble de l'étendue](#).

L'exemple suivant attribue le rôle `Azure Service Bus Data Owner` à votre compte d'utilisateur, qui fournit un accès complet aux ressources Azure Service Bus. Dans un scénario réel, suivez le [principe des privilèges minimum](#) pour accorder aux utilisateurs uniquement les autorisations minimales nécessaires à un environnement de production plus sécurisé.

Rôles Azure intégrés pour Azure Service Bus

Pour Azure Service Bus, la gestion des espaces de noms et de toutes les ressources associées via le Portail Azure et l'API de gestion des ressources Azure est déjà protégée à l'aide du modèle Azure RBAC. Azure fournit les rôles Azure intégrés ci-dessous pour autoriser l'accès à un espace de noms Service Bus :

- [Propriétaire de données Azure Service Bus](#) : ce rôle permet l'accès aux données de l'espace de noms Service Bus et de ses entités (files d'attente, rubriques, abonnements et filtres). Un membre de ce rôle peut envoyer et recevoir des messages à partir de files d'attente ou de rubriques et d'abonnements.

- [Expéditeur de données Azure Service Bus](#) : utilisez ce rôle pour autoriser l'accès en envoi à l'espace de noms Service Bus et à ses entités.
- [Récepteur de données Azure Service Bus](#) : utilisez ce rôle pour autoriser l'accès en réception à l'espace de noms Service Bus et à ses entités.

Si vous souhaitez créer un rôle personnalisé, consultez [Droits requis pour les opérations Service Bus](#).

Ajouter un utilisateur Microsoft Entra au rôle Propriétaire Azure Service Bus

Ajoutez votre nom d'utilisateur Microsoft Entra au rôle **Propriétaire de données Azure Service Bus** au niveau de l'espace de noms Service Bus. Il permet à une application exécutée dans le contexte de votre compte d'utilisateur d'envoyer des messages à une file d'attente ou à une rubrique et d'en recevoir auprès d'une file d'attente ou de l'abonnement d'une rubrique.

Important

Dans la plupart des cas, la propagation de l'attribution de rôle dans Azure peut prendre une ou deux minutes. Dans de rares cas, cela peut prendre jusqu'à **huit minutes**. Si vous recevez des erreurs d'authentification lorsque vous exécutez votre code pour la première fois, patientez quelques instants et réessayez.

1. Si la page Espace de noms Service Bus n'est pas ouverte sur le Portail Azure, recherchez votre espace de noms Service Bus à l'aide de la barre de recherche principale ou du volet de navigation de gauche.
2. Dans la page vue d'ensemble, sélectionnez **Contrôle d'accès (IAM)** dans le menu de gauche.
3. Sur la page **Contrôle d'accès (IAM)**, sélectionnez l'onglet **Attributions de rôles**.
4. Sélectionnez **+ Ajouter** dans le menu supérieur, puis **Ajouter une attribution de rôle** dans le menu déroulant résultant.

5. Utilisez la zone de recherche pour filtrer les résultats sur le rôle souhaité. Pour cet exemple, recherchez **Azure Service Bus Data Owner** et sélectionnez le résultat correspondant. Ensuite, choisissez **Suivant**.
6. Sous **Attribuer l'accès à**, sélectionnez **Utilisateur, groupe ou principal de service**, puis sélectionnez + **Sélectionner des membres**.
7. Dans la boîte de dialogue, recherchez votre nom d'utilisateur Microsoft Entra (généralement votre adresse e-mail *user@domain*), puis choisissez **Sélectionner** en bas de la boîte de dialogue.
8. Sélectionnez **Vérifier + affecter** pour accéder à la page finale, puis **Vérifier + attribuer** à nouveau pour terminer le processus.

Envoi de messages à une rubrique

Dans cette section, vous allez créer un projet de console Java et ajouter du code pour envoyer des messages à la rubrique que vous avez créée.

Créer un projet de console Java

Créez un projet Java avec Eclipse ou un outil de votre choix.

Configuration de votre application pour l'utilisation de Service Bus

Ajoutez des références aux bibliothèques Azure Core et Azure Service Bus.

Si vous utilisez Eclipse et avez créé une application de console Java, convertissez votre projet Java en Maven : faites un clic droit sur le projet dans la fenêtre **Explorateur de package**, puis sélectionnez **Configurer – >Convertir en projet Maven**. Ajoutez ensuite des dépendances à ces deux bibliothèques, comme indiqué dans l'exemple suivant.

Sans mot de passe (recommandé)

Mettez à jour le fichier `pom.xml` pour ajouter des dépendances aux packages Azure Service Bus et Azure Identity.

XML

```
<dependencies>
  <dependency>
    <groupId>com.azure</groupId>
    <artifactId>azure-messaging-servicebus</artifactId>
    <version>7.13.3</version>
  </dependency>
  <dependency>
    <groupId>com.azure</groupId>
    <artifactId>azure-identity</artifactId>
    <version>1.8.0</version>
    <scope>compile</scope>
  </dependency>
</dependencies>
```

Ajouter du code pour envoyer des messages à la rubrique

1. Ajoutez les instructions `import` suivantes au début du fichier Java.

Sans mot de passe (recommandé)

Java

```
import com.azure.messaging.servicebus.*;
import com.azure.identity.*;

import java.util.concurrent.CountDownLatch;
import java.util.concurrent.TimeUnit;
import java.util.Arrays;
import java.util.List;
```

2. Dans la classe , définissez des variables pour contenir la chaîne de connexion (non nécessaire pour le scénario sans mot de passe), le nom de la rubrique et le nom de l'abonnement.

Sans mot de passe (recommandé)

Java

```
static String topicName = "<TOPIC NAME>";  
static String subName = "<SUBSCRIPTION NAME>";
```

ⓘ Important

Remplacez <TOPIC NAME> par le nom de la rubrique et <SUBSCRIPTION NAME> par le nom de l'abonnement de la rubrique.

3. Ajoutez une méthode nommée `sendMessage` dans la classe pour envoyer un message à la rubrique.

Sans mot de passe (recommandé)

ⓘ Important

Remplacez NAMESPACENAME par le nom de votre espace de noms Service Bus.

Java

```
static void sendMessage()  
{  
    // create a token using the default Azure credential  
    DefaultAzureCredential credential = new  
DefaultAzureCredentialBuilder()  
        .build();  
  
    ServiceBusSenderClient senderClient = new  
ServiceBusClientBuilder()  
  
.fullyQualifiedNamespace("NAMESPACENAME.servicebus.windows.net")  
    .credential(credential)  
    .sender()  
    .topicName(topicName)  
    .buildClient();
```

```
// send one message to the topic
    senderClient.sendMessage(new ServiceBusMessage("Hello,
World!"));
    System.out.println("Sent a single message to the topic: " +
topicName);
}
```

4. Ajoutez une méthode nommée `createMessages` dans la classe pour créer une liste de messages. En général, vous recevez ces messages à partir de différentes parties de votre application. Ici, nous créons une liste d'exemples de messages.

Java

```
static List<ServiceBusMessage> createMessages()
{
    // create a list of messages and return it to the caller
    ServiceBusMessage[] messages = {
        new ServiceBusMessage("First message"),
        new ServiceBusMessage("Second message"),
        new ServiceBusMessage("Third message")
    };
    return Arrays.asList(messages);
}
```

5. Ajoutez une méthode nommée `sendMessageBatch` pour envoyer des messages à la rubrique que vous avez créée. Cette méthode crée un `ServiceBusSenderClient` pour la rubrique, appelle la méthode `createMessages` pour obtenir la liste des messages, prépare un ou plusieurs lots, et envoie les lots à la rubrique.

Sans mot de passe (recommandé)

ⓘ Important

Remplacez `NAMESPACE` par le nom de votre espace de noms Service Bus.

Java

```
static void sendMessageBatch()
{
    // create a token using the default Azure credential
    DefaultAzureCredential credential = new
DefaultAzureCredentialBuilder()
```

```
        .build();

        ServiceBusSenderClient senderClient = new
ServiceBusClientBuilder()

        .fullyQualifiedNamespace("NAMESPACE.servicebus.windows.net")
            .credential(credential)
            .sender()
            .topicName(topicName)
            .buildClient();

        // Creates an ServiceBusMessageBatch where the ServiceBus.
        ServiceBusMessageBatch messageBatch =
senderClient.createMessageBatch();

        // create a list of messages
        List<ServiceBusMessage> listOfMessages = createMessages();

        // We try to add as many messages as a batch can fit based on
        the maximum size and send to Service Bus when
        // the batch can hold no more messages. Create a new batch for
        next set of messages and repeat until all
        // messages are sent.
        for (ServiceBusMessage message : listOfMessages) {
            if (messageBatch.tryAddMessage(message)) {
                continue;
            }

            // The batch is full, so we create a new batch and send the
            batch.
            senderClient.sendMessages(messageBatch);
            System.out.println("Sent a batch of messages to the topic:
" + topicName);

            // create a new batch
            messageBatch = senderClient.createMessageBatch();

            // Add that message that we couldn't before.
            if (!messageBatch.tryAddMessage(message)) {
                System.err.printf("Message is too large for an empty
batch. Skipping. Max size: %s.", messageBatch.getMaxSizeInBytes());
            }
        }

        if (messageBatch.getCount() > 0) {
            senderClient.sendMessages(messageBatch);
            System.out.println("Sent a batch of messages to the topic:
" + topicName);
        }

        //close the client
        senderClient.close();
    }
```

Réception des messages d'un abonnement

Dans cette section, vous allez ajouter du code pour récupérer les messages à partir d'un abonnement à la rubrique.

1. Ajoutez une méthode nommée `receiveMessages` pour recevoir des messages à partir de l'abonnement. Cette méthode crée un `ServiceBusProcessorClient` pour l'abonnement en spécifiant un gestionnaire pour le traitement des messages et un autre pour la gestion des erreurs. Ensuite, elle démarre le processeur, attend quelques secondes, imprime les messages reçus, puis arrête et ferme le processeur.

Sans mot de passe (recommandé)

ⓘ Important

- Remplacez `NAMESPACE` par le nom de votre espace de noms Service Bus.
- Dans le code, remplacez `ServiceBusTopicTest` dans `ServiceBusTopicTest::processMessage` par le nom de votre classe.

Java

```
// handles received messages
static void receiveMessages() throws InterruptedException
{
    CountDownLatch countdownLatch = new CountDownLatch(1);

    DefaultAzureCredential credential = new
DefaultAzureCredentialBuilder()
        .build();

    // Create an instance of the processor through the
    ServiceBusClientBuilder
    ServiceBusProcessorClient processorClient = new
ServiceBusClientBuilder()

    .fullyQualifiedNamespace("NAMESPACE.servicebus.windows.net")
        .credential(credential)
        .processor()
        .topicName(topicName)
        .subscriptionName(subName)
        .processMessage(ServiceBusTopicTest::processMessage)
        .processError(context -> processError(context,
countdownLatch))
        .buildProcessorClient();
```

```

        System.out.println("Starting the processor");
        processorClient.start();

        TimeUnit.SECONDS.sleep(10);
        System.out.println("Stopping and closing the processor");
        processorClient.close();
    }
}

```

2. Ajoutez la méthode `processMessage` pour traiter un message reçu de l'abonnement Service Bus.

Java

```

private static void processMessage(ServiceBusReceivedMessageContext context) {
    ServiceBusReceivedMessage message = context.getMessage();
    System.out.printf("Processing message. Session: %s, Sequence #: %s.
Contents: %s%n", message.getMessageId(),
    message.getSequenceNumber(), message.getBody());
}

```

3. Ajoutez la méthode `processError` pour gérer les messages d'erreur.

Java

```

private static void processError(ServiceBusErrorContext context,
CountDownLatch countdownLatch) {
    System.out.printf("Error when receiving messages from namespace:
'%s'. Entity: '%s'%n",
    context.getFullyQualifiedNamespace(), context.getEntityPath());

    if (!(context.getException() instanceof ServiceBusException)) {
        System.out.printf("Non-ServiceBusException occurred: %s%n",
    context.getException());
        return;
    }

    ServiceBusException exception = (ServiceBusException)
context.getException();
    ServiceBusFailureReason reason = exception.getReason();

    if (reason == ServiceBusFailureReason.MESSAGING_ENTITY_DISABLED
    || reason == ServiceBusFailureReason.MESSAGING_ENTITY_NOT_FOUND
    || reason == ServiceBusFailureReason.UNAUTHORIZED) {
        System.out.printf("An unrecoverable error occurred. Stopping
processing with reason %s: %s%n",
        reason, exception.getMessage());

    countdownLatch.countDown();
}

```

```

    } else if (reason == ServiceBusFailureReason.MESSAGE_LOCK_LOST) {
        System.out.printf("Message lock lost for message: %s%n",
context.getException());
    } else if (reason == ServiceBusFailureReason.SERVICE_BUSY) {
        try {
            // Choosing an arbitrary amount of time to wait until
            // trying again.
            TimeUnit.SECONDS.sleep(1);
        } catch (InterruptedException e) {
            System.err.println("Unable to sleep for period of time");
        }
    } else {
        System.out.printf("Error source %s, reason %s, message: %s%n",
context.getErrorSource(),
                    reason, context.getException());
    }
}

```

4. Mettez à jour la méthode `main` pour appeler les méthodes `sendMessage`, `sendMessageBatch` et `receiveMessages`, et pour lever `InterruptedException`.

Java

```

public static void main(String[] args) throws InterruptedException {
    sendMessage();
    sendMessageBatch();
    receiveMessages();
}

```

Exécuter l'application

Exécutez le programme pour obtenir une sortie de ce type :

Sans mot de passe (recommandé)

- Si vous utilisez Eclipse, faites un clic droit sur le projet, sélectionnez **Exporter**, développez **Java**, sélectionnez **Fichier JAR exécutable**, puis suivez les étapes pour créer un fichier JAR exécutable.
- Si vous êtes connecté à l'ordinateur à l'aide d'un compte d'utilisateur différent du compte d'utilisateur ajouté au rôle **propriétaire des données Azure Service Bus**, suivez les étapes suivantes. Sinon, ignorez cette étape et passez à l'exécution du fichier JAR à l'étape suivante.
 - [Installez Azure CLI](#) sur votre machine.

b. Exécutez la commande CLI suivante pour vous connecter à Azure. Utilisez le même compte d'utilisateur que celui que vous avez ajouté au rôle **propriétaire des données Azure Service Bus**.

```
Azure CLI
```

```
az login
```

3. Exécutez le fichier JAR à l'aide de la commande suivante.

```
Java
```

```
java -jar <JAR FILE NAME>
```

4. Vous voyez la sortie suivante dans la fenêtre de console.

```
Console
```

```
Sent a single message to the topic: mytopic
Sent a batch of messages to the topic: mytopic
Starting the processor
Processing message. Session: e0102f5fbaf646988a2f4b65f7d32385,
Sequence #: 1. Contents: Hello, World!
Processing message. Session: 3e991e232ca248f2bc332caa8034bed9,
Sequence #: 2. Contents: First message
Processing message. Session: 56d3a9ea7df446f8a2944ee72cca4ea0,
Sequence #: 3. Contents: Second message
Processing message. Session: 7bd3bd3e966a40ebbc9b29b082da14bb,
Sequence #: 4. Contents: Third message
```

Dans la page **Vue d'ensemble** de l'espace de noms Service Bus dans le portail Azure, vous pouvez voir le nombre de messages **entrants** et **sortants**. Vous devrez peut-être attendre environ une minute puis actualiser la page pour voir les valeurs les plus récentes.

Dashboard > spsbusns0216

Overview Queue Topic Refresh Delete

Show data for the last: 1 hour 6 hours 12 hours 1 day 7 days 30 days

Requests

Messages

UTC-05:00

Name	Status	Max size	Enable partitioning
myqueue	Active	1024 MB	false

Basculez vers l'onglet **Rubriques** dans le volet du milieu inférieur, puis sélectionnez la rubrique pour afficher la page **Rubrique Service Bus** correspondant à votre rubrique. Dans cette page, vous devez voir quatre messages entrants et quatre messages sortants dans le graphique **Messages**.

mytopic (spsbusns1115/mytopic)

Overview Subscription Delete Refresh

Access control (IAM) Diagnose and solve problems

Settings

Shared access policies Service Bus Explorer (preview)

Properties

Locks

Entities

Subscriptions

Automation

Tasks

Export template

Support + troubleshooting

New support request

Show data for the last: 1 hour 6 hours 12 hours 1 day 7 days 30 days

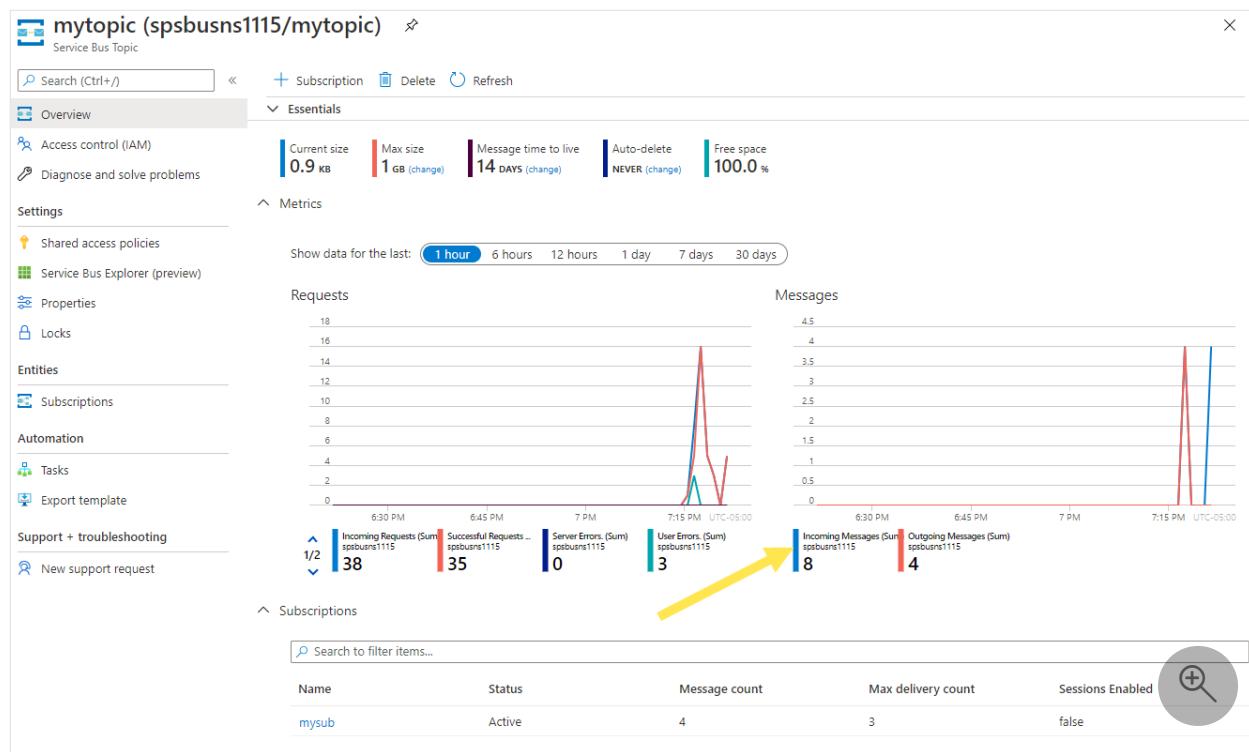
Requests

Messages

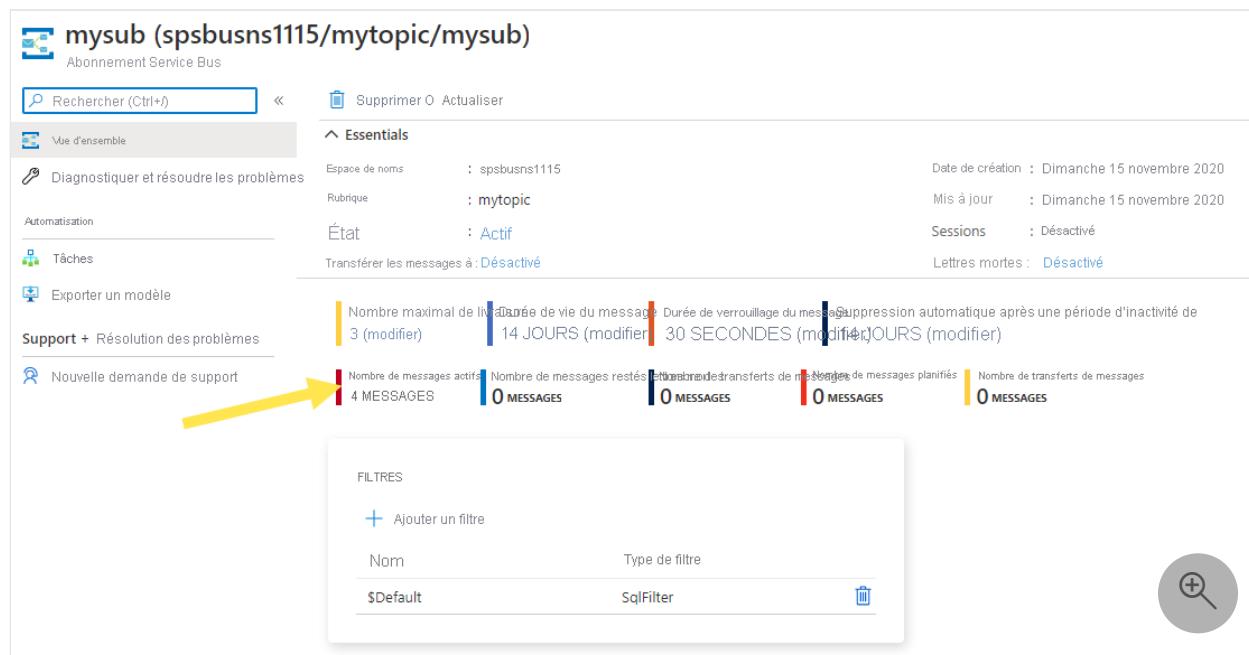
UTC-05:00

Name	Status	Message count	Max delivery count	Sessions Enabled
mysub	Active	0	3	false

Si vous commentez l'appel `receiveMessages` dans la méthode `main` et que vous réexécutez l'application, dans la page **Rubrique Service Bus** vous verrez huit messages entrants (quatre nouveaux) mais quatre messages sortants.



Dans cette page, si vous sélectionnez un abonnement, vous accédez à la page **Abonnement Service Bus**. Elle indique entre autres le nombre de messages actifs et le nombre de messages de lettres mortes. Dans cet exemple, quatre messages actifs n'ont pas encore été reçus par un récepteur.



Étapes suivantes

Voir la documentation et les exemples suivants :

- Bibliothèque de client Azure Service Bus pour Java - Lisez-moi ↗
- Exemples sur GitHub

- Informations de référence sur l'API Java ↗

Démarrage rapide : Bibliothèque de client Stockage Blob Azure pour Java

Article • 25/03/2023

Commencez à utiliser la bibliothèque de client Stockage Blob Azure pour Java pour gérer les objets blob et les conteneurs. Suivez les étapes suivantes pour installer le package et essayer un exemple de code pour les tâches de base.

💡 Conseil

Si vous utilisez des ressources de Stockage Azure dans une application Spring, nous vous recommandons de considérer [Azure Spring Cloud](#) comme alternative. Azure Spring Cloud est un projet open source qui fournit une intégration de Spring fluide aux services Azure. Pour en savoir plus sur Azure Spring Cloud et pour voir un exemple d'utilisation du Stockage Blob, consultez [Charger un fichier dans un Stockage Blob Azure](#).

[Documentation de référence sur l'API](#) | [Code source de la bibliothèque](#) | [Package \(Maven\)](#) | [Exemples](#)

Prérequis

- Compte Azure avec un abonnement actif : [créez gratuitement un compte](#).
- Compte de stockage Azure : [créez un compte de stockage](#).
- [Java Development Kit \(JDK\)](#) version 8 ou ultérieure.
- [Apache Maven](#).

Configuration

Cette section vous guide tout au long de la préparation d'un projet à utiliser avec la bibliothèque de client Stockage Blob Azure pour Java.

Créer le projet

Créez une application Java nommée *blob-quickstart*.

1. Dans une fenêtre de console (par exemple, PowerShell ou Bash), utilisez Maven pour créer une application console nommée *blob-quickstart*. Tapez la commande

mvn suivante pour créer un projet Java « Hello world! ».

PowerShell

```
mvn archetype:generate `--define interactiveMode=n` --define groupId=com.blobs.quickstart` --define artifactId=blob-quickstart` --define archetypeArtifactId=maven-archetype-quickstart` --define archetypeVersion=1.4
```

2. Le résultat de la génération du projet doit ressembler à ceci :

Console

```
[INFO] Scanning for projects...
[INFO]
[INFO] -----< org.apache.maven:standalone-pom >-----
[INFO] Building Maven Stub Project (No POM) 1
[INFO] -----[ pom ]-----
[INFO]
[INFO] >>> maven-archetype-plugin:3.1.2:generate (default-cli) >
generate-sources @ standalone-pom >>>
[INFO]
[INFO] <<< maven-archetype-plugin:3.1.2:generate (default-cli) <
generate-sources @ standalone-pom <<<
[INFO]
[INFO]
[INFO] --- maven-archetype-plugin:3.1.2:generate (default-cli) @
standalone-pom ---
[INFO] Generating project in Batch mode
[INFO] -----
[INFO] Using following parameters for creating project from Archetype:
maven-archetype-quickstart:1.4
[INFO] -----
[INFO] Parameter: groupId, Value: com.blobs.quickstart
[INFO] Parameter: artifactId, Value: blob-quickstart
[INFO] Parameter: version, Value: 1.0-SNAPSHOT
[INFO] Parameter: package, Value: com.blobs.quickstart
[INFO] Parameter: packageInPathFormat, Value: com/blobs/quickstart
[INFO] Parameter: version, Value: 1.0-SNAPSHOT
[INFO] Parameter: package, Value: com.blobs.quickstart
[INFO] Parameter: groupId, Value: com.blobs.quickstart
[INFO] Parameter: artifactId, Value: blob-quickstart
[INFO] Project created from Archetype in dir: C:\QuickStarts\blob-
```

```
quickstart
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 7.056 s
[INFO] Finished at: 2019-10-23T11:09:21-07:00
[INFO] -----
```
``
```

3. Basculez vers le dossier *blob-quickstart* nouvellement créé.

```
Console
cd blob-quickstart
```

4. Dans le répertoire *blob-quickstart*, créez un autre répertoire nommé *data*. Ce dossier est l'endroit où les fichiers de données Blob seront créés et stockés.

```
Console
mkdir data
```

## Installer les packages

Ouvrez le fichier `pom.xml` dans votre éditeur de texte.

Ajoutez **azure-sdk-bom** pour établir une dépendance sur la dernière version de la bibliothèque. Dans l'extrait de code suivant, remplacez l'espace réservé `{bom_version_to_target}` par le numéro de version. L'utilisation d'**azure-sdk-bom** vous empêche de devoir spécifier la version de chaque dépendance individuelle. Pour en savoir plus sur la nomenclature, consultez le [fichier Lisez-moi de la nomenclature du SDK Azure](#).

```
XML
<dependencyManagement>
 <dependencies>
 <dependency>
 <groupId>com.azure</groupId>
 <artifactId>azure-sdk-bom</artifactId>
 <version>{bom_version_to_target}</version>
 <type>pom</type>
```

```
<scope>import</scope>
</dependency>
</dependencies>
</dependencyManagement>
```

Ajoutez ensuite les éléments dependency suivants au groupe de dépendances. La dépendance **azure-identity** est nécessaire pour les connexions sans mot de passe aux services Azure.

XML

```
<dependency>
 <groupId>com.azure</groupId>
 <artifactId>azure-storage-blob</artifactId>
</dependency>
<dependency>
 <groupId>com.azure</groupId>
 <artifactId>azure-identity</artifactId>
</dependency>
```

## Configurer le framework d'application

À partir du répertoire du projet, procédez comme suit pour créer la structure de base de l'application :

1. Accédez au répertoire `/src/main/Java/com/blobs/QuickStart`
2. Ouvrez le fichier `App.java` dans votre éditeur.
3. Supprimez la ligne `System.out.println("Hello world!");`
4. Ajoutez les directives `import` requises

Le code doit ressembler à ce framework :

Java

```
package com.blobs.quickstart;

/**
 * Azure Blob Storage quickstart
 */
import com.azure.identity.*;
import com.azure.storage.blob.*;
import com.azure.storage.blob.models.*;
import java.io.*;

public class App
{
 public static void main(String[] args) throws IOException
 {
```

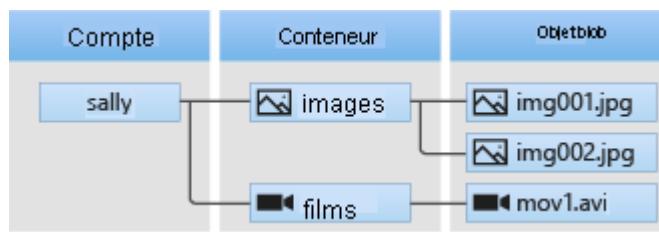
```
// Quickstart code goes here
}
}
```

## Modèle objet

Stockage Blob Azure est optimisé pour stocker des quantités massives de données non structurées. Les données non structurées n'obéissent pas à un modèle ou une définition de données en particulier, comme des données texte ou binaires. Le stockage Blob offre trois types de ressources :

- Le compte de stockage
- Un conteneur dans le compte de stockage.
- Un blob dans le conteneur

Le diagramme suivant montre la relation entre ces ressources.



Utilisez les classes Java suivantes pour interagir avec ces ressources :

- **BlobServiceClient:** La classe `BlobServiceClient` vous permet de manipuler les ressources de stockage Azure et les conteneurs blob. Le compte de stockage fournit l'espace de noms de niveau supérieur pour le service BLOB.
- La classe [fournit une API de générateur Fluent pour faciliter la configuration et l'instanciation d'objets](#) .
- **BlobContainerClient** : La classe `BlobContainerClient` vous permet de manipuler des conteneurs de stockage Azure et leurs blobs.
- La classe [vous permet de manipuler des blobs de stockage Azure](#).
- **BlobItem** : La classe `BlobItem` représente des objets blob retournés par un appel à [listBlobs](#).

## Exemples de code

Ces exemples d'extraits de code vous montrent comment effectuer les actions suivantes avec la bibliothèque de client Stockage Blob Azure pour Java :

- [S'authentifier auprès d'Azure et autoriser l'accès aux données d'objet blob](#)

- [Créer un conteneur](#)
- [Charger des objets blob sur un conteneur](#)
- [Lister les objets blob d'un conteneur](#)
- [Télécharger des objets blob](#)
- [Supprimer un conteneur](#)

### Important

Vérifiez que vous avez les dépendances appropriées dans pom.xml ainsi que les directives requises pour que les exemples de code fonctionnent, comme décrit dans la section de [configuration](#).

## S'authentifier auprès d'Azure et autoriser l'accès aux données d'objet blob

Les demandes d'application vers le Stockage Blob Azure doivent être autorisées.

L'utilisation de la classe `DefaultAzureCredential` fournie par la bibliothèque de client Azure Identity est l'approche recommandée pour implémenter des connexions sans mot de passe aux services Azure dans votre code, y compris le Stockage Blob.

Vous pouvez également autoriser les demandes vers le Stockage Blob Azure à l'aide de la clé d'accès au compte. Toutefois, cette approche doit être utilisée avec prudence. Les développeurs doivent être vigilants pour ne jamais exposer la clé d'accès dans un emplacement non sécurisé. Toute personne disposant de la clé d'accès est en mesure d'autoriser les demandes sur le compte de stockage et a accès efficacement à toutes les données. `DefaultAzureCredential` offre des avantages améliorés en matière de gestion et de sécurité par rapport à la clé de compte pour autoriser l'authentification sans mot de passe. Les deux options sont illustrées dans l'exemple suivant.

### Sans mot de passe (recommandé)

`DefaultAzureCredential` est une classe fournie par la bibliothèque de client Azure Identity pour Java. `DefaultAzureCredential` prend en charge plusieurs méthodes d'authentification et détermine quelle méthode doit être utilisée au moment de l'exécution. Cette approche permet à votre application d'utiliser différentes méthodes d'authentification dans différents environnements (local ou production) sans implémenter de code spécifique à l'environnement.

L'ordre et les emplacements dans lesquels `DefaultAzureCredential` les informations d'identification sont disponibles dans la [vue d'ensemble de la bibliothèque](#)

d'identités Azure.

Par exemple, votre application peut s'authentifier à l'aide de vos informations d'identification de connexion Visual Studio Code lors du développement local. Votre application peut ensuite utiliser une [identité managée](#) une fois qu'elle a été déployée sur Azure. Aucune modification du code n'est requise pour cette transition.

## Attribuer des rôles à votre compte d'utilisateur Azure AD

Lors du développement local, assurez-vous que le compte d'utilisateur qui accède aux données blob dispose des autorisations appropriées. Vous aurez besoin du **Contributeur aux données Blob de stockage** pour lire et écrire des données blob. Pour vous attribuer ce rôle, vous aurez besoin du rôle **Administrateur de l'accès utilisateur** ou d'un autre rôle qui inclut l'action [Microsoft.Authorization/roleAssignments/write](#). Vous pouvez attribuer des rôles RBAC Azure à un utilisateur à l'aide du Portail Azure, Azure CLI ou Azure PowerShell. Vous pouvez en savoir plus sur les étendues disponibles pour les attributions de rôles dans la page [vue d'ensemble de l'étendue](#).

Dans ce scénario, vous allez attribuer des autorisations à votre compte d'utilisateur, étendues au compte de stockage, pour suivre le [Principe des priviléges minimum](#). Cette pratique offre aux utilisateurs uniquement les autorisations minimales nécessaires et crée des environnements de production plus sécurisés.

L'exemple suivant affecte le rôle **Contributeur aux données Blob du stockage** à votre compte d'utilisateur, qui fournit à la fois un accès en lecture et en écriture aux données d'objet blob dans votre compte de stockage.

### ⓘ Important

Dans la plupart des cas, la propagation de l'attribution de rôle dans Azure peut prendre une ou deux minutes, mais dans de rares cas, cela peut prendre jusqu'à huit minutes. Si vous recevez des erreurs d'authentification lorsque vous exécutez votre code pour la première fois, patientez quelques instants et réessayez.

Azure portal

1. Dans le Portail Azure, recherchez votre compte de stockage à l'aide de la barre de recherche principale ou de la navigation gauche.
2. Dans la page vue d'ensemble du compte de stockage, sélectionnez **Contrôle d'accès (IAM)** dans le menu de gauche.
3. Sur la page **Contrôle d'accès (IAM)**, sélectionnez l'onglet **Attributions de rôles**.
4. Sélectionnez **+ Ajouter** dans le menu supérieur, puis **Ajouter une attribution de rôle** dans le menu déroulant résultant.

The screenshot shows the 'Access Control (IAM)' blade for a storage account named 'identitymigrationstorage'. The left sidebar lists various management options like Data migration, Events, Storage browser, etc. The 'Access Control (IAM)' option is selected and highlighted with a red box. At the top, there's a search bar and several buttons: '+ Add' (highlighted with a red box), 'Download role assignments', 'Edit columns', 'Refresh', 'Remove', and 'Got feedback?'. A dropdown menu is open over the '+ Add' button, showing 'Add role assignment' (also highlighted with a red box) and 'Add co-administrator'. Below this, there are sections for 'My access' (with a 'View my access' button), 'Check access' (with a 'Find' input field containing '(User, group, or service principal)' and two radio buttons for 'User, group, or service principal' and 'Managed identity'), and a search bar. To the right, there are three main cards: 'Grant access to this resource' (with a 'Add role assignment' button and 'Learn more' link), 'View deny assignments' (with a 'View' button and 'Learn more' link), and a 'View' button next to a magnifying glass icon.

5. Utilisez la zone de recherche pour filtrer les résultats sur le rôle souhaité. Pour cet exemple, recherchez *Contributeur aux données Blob du stockage*, sélectionnez le résultat correspondant, puis choisissez **Suivant**.
6. Sous **Attribuer l'accès à**, sélectionnez **Utilisateur, groupe ou principal de service**, puis sélectionnez **+ Sélectionner des membres**.
7. Dans la boîte de dialogue, recherchez votre nom d'utilisateur Azure AD (généralement votre adresse e-mail *user@domain*), puis choisissez **Sélectionner** en bas de la boîte de dialogue.
8. Sélectionnez **Vérifier + affecter** pour accéder à la page finale, puis **Vérifier + attribuer** à nouveau pour terminer le processus.

**Connectez-vous et connectez votre code d'application à Azure à l'aide de DefaultAzureCredential**

Vous pouvez autoriser l'accès aux données dans votre compte de stockage en procédant comme suit :

1. Vérifiez que vous êtes authentifié avec le même compte Azure AD auquel vous avez attribué le rôle sur votre compte de stockage. Vous pouvez vous authentifier via Azure CLI, Visual Studio Code ou Azure PowerShell.

Azure CLI

Connectez-vous à Azure via Azure CLI à l'aide de la commande suivante :

Azure CLI

```
az login
```

2. Pour utiliser `DefaultAzureCredential`, vérifiez que la dépendance `azure-identity` est ajoutée dans `pom.xml` :

XML

```
<dependency>
 <groupId>com.azure</groupId>
 <artifactId>azure-identity</artifactId>
</dependency>
```

3. Ajoutez ce code à la méthode `Main`. Lorsque le code est exécuté sur votre station de travail locale, il utilise les informations d'identification du développeur de l'outil hiérarchisé auquel vous êtes connecté pour vous authentifier auprès d'Azure, comme Azure CLI ou Visual Studio Code.

Java

```
/*
 * The default credential first checks environment variables for
 * configuration
 * If environment configuration is incomplete, it will try managed
 * identity
 */
DefaultAzureCredential defaultCredential = new
DefaultAzureCredentialBuilder().build();

// Azure SDK client builders accept the credential as a parameter
// TODO: Replace <storage-account-name> with your actual storage
account name
BlobServiceClient blobServiceClient = new
BlobServiceClientBuilder()
```

```
.endpoint("https://<storage-account-name>.blob.core.windows.net/")
.credential(defaultCredential)
.buildClient();
```

4. Veillez à mettre à jour le nom du compte de stockage dans l'URI de votre `BlobServiceClient`. Le nom du compte de stockage se trouve sur la page vue d'ensemble du Portail Azure.

A screenshot of the Azure Storage Account overview page. The account name "identitymigrationstorage" is highlighted with a red box. The "Essentials" section displays the following information:

- Resource group ([move](#)) : alexw-identity-revamp
- Location : East US
- Primary/Secondary Location : Primary: East US, Secondary: West US
- Subscription ([move](#)) : C&L Cross Service Content Team Testing
- Subscription ID :
- Disk state : Primary: Available, Secondary: Available
- Tags ([edit](#)) : Click here to add tags

### ⓘ Notes

Lorsqu'il est déployé sur Azure, ce même code peut être utilisé pour autoriser les demandes adressées à Stockage Azure à partir d'une application s'exécutant dans Azure. Toutefois, vous devez activer l'identité managée sur votre application dans Azure. Configurez ensuite votre compte de stockage pour autoriser cette identité managée à se connecter. Pour obtenir des instructions détaillées sur la configuration de cette connexion entre les services Azure, consultez le didacticiel [d'authentification à partir d'applications hébergées sur Azure](#).

## Créez un conteneur.

Choisissez un nom pour le nouveau conteneur. Le code ci-dessous ajoute une valeur UUID au nom du conteneur pour garantir son unicité.

### ⓘ Important

Les noms de conteneurs doivent être en minuscules. Pour plus d'informations sur l'affectation de noms aux conteneurs et objets blob, consultez [Affectation de noms et références aux conteneurs, objets blob et métadonnées](#).

Ensuite, créez une instance de la classe [BlobContainerClient](#), puis appelez la méthode `create` pour créer le conteneur dans votre compte de stockage.

Ajoutez ce code à la fin de la méthode `Main` :

Java

```
// Create a unique name for the container
String containerName = "quickstartblobs" + java.util.UUID.randomUUID();

// Create the container and return a container client object
BlobContainerClient blobContainerClient =
blobServiceClient.createBlobContainer(containerName);
```

Pour en savoir plus sur la création d'un conteneur et explorer d'autres exemples de code, consultez [Créer un conteneur d'objets blob avec Java](#).

## Charger des objets blob sur un conteneur

Ajoutez ce code à la fin de la méthode `Main` :

Java

```
// Create a local file in the ./data/ directory for uploading and
// downloading
String localPath = "./data/";
String fileName = "quickstart" + java.util.UUID.randomUUID() + ".txt";

// Get a reference to a blob
BlobClient blobClient = blobContainerClient.getBlobClient(fileName);

// Write text to the file
FileWriter writer = null;
try
{
 writer = new FileWriter(localPath + fileName, true);
 writer.write("Hello, World!");
 writer.close();
}
catch (IOException ex)
{
 System.out.println(ex.getMessage());
}
```

```
System.out.println("\nUploading to Blob storage as blob:\n\t" +
blobClient.getBlobUrl());

// Upload the blob
blobClient.uploadFromFile(localPath + fileName);
```

L'extrait de code effectue les étapes suivantes :

1. Crée un fichier texte dans le répertoire *data* local.
2. Obtient une référence à un objet `BlobClient` en appelant la méthode `getBlobClient` sur le conteneur évoqué dans la section [Créer un conteneur](#).
3. Charge le fichier texte local dans l'objet Blob en appelant la méthode `uploadFromFile`. Cette méthode crée l'objet blob s'il n'existe pas déjà, mais ne le remplacera pas s'il existe.

Pour en savoir plus sur le chargement d'objets blob et explorer d'autres exemples de code, consultez [Charger un objet blob avec Java](#).

## Créer la liste des objets blob d'un conteneur

Répertoriez les objets Blob dans le conteneur en appelant la méthode `list_Blobs`. Dans ce cas, un seul objet blob a été ajouté au conteneur. Il n'y a donc qu'un objet blob répertorié.

Ajoutez ce code à la fin de la méthode `Main` :

Java

```
System.out.println("\nListing blobs...");

// List the blob(s) in the container.
for (BlobItem blobItem : blobContainerClient.listBlobs()) {
 System.out.println("\t" + blobItem.getName());
}
```

Pour en savoir plus sur les listings d'objets blob et explorer d'autres exemples de code, consultez [Répertorier les objets blob avec Java](#).

## Télécharger des objets blob

Téléchargez l'objet blob créé précédemment en appelant la méthode `downloadToFile`. L'exemple de code ajoute le suffixe « DOWNLOAD » au nom de fichier afin que vous puissiez voir les deux fichiers dans votre système de fichiers local.

Ajoutez ce code à la fin de la méthode `Main` :

```
Java

// Download the blob to a local file

// Append the string "DOWNLOAD" before the .txt extension for comparison
purposes
String downloadFileName = fileName.replace(".txt", "DOWNLOAD.txt");

System.out.println("\nDownloading blob to\n\t" + localPath +
downloadFileName);

blobClient.downloadToFile(localPath + downloadFileName);
```

Pour en savoir plus sur le téléchargement d'objets blob et explorer d'autres exemples de code, consultez [Télécharger un objet blob avec Java](#).

## Supprimer un conteneur

Le code suivant nettoie les ressources créées par l'application en supprimant le conteneur tout entier à l'aide de la méthode [supprimer](#). Il supprime également les fichiers locaux créés par l'application.

L'application s'interrompt pour une entrée de l'utilisateur en appelant `System.console().readLine()` avant de supprimer l'objet blob, le conteneur et les fichiers locaux. C'est l'occasion de vérifier que les ressources ont été créées correctement, avant d'être supprimées.

Ajoutez ce code à la fin de la méthode `Main` :

```
Java

File downloadedFile = new File(localPath + downloadFileName);
File localFile = new File(localPath + fileName);

// Clean up resources
System.out.println("\nPress the Enter key to begin clean up");
System.console().readLine();

System.out.println("Deleting blob container...");
blobContainerClient.delete();

System.out.println("Deleting the local source and downloaded files...");
localFile.delete();
downloadedFile.delete();
```

```
System.out.println("Done");
```

Pour en savoir plus sur la suppression d'un conteneur et explorer d'autres exemples de code, consultez [Supprimer et restaurer un conteneur d'objets blob avec Java](#).

## Exécuter le code

Cette application crée un fichier de test dans votre dossier local et le charge sur un stockage blob. L'exemple liste ensuite les objets blob du conteneur et télécharge le fichier avec un nouveau nom pour que vous puissiez comparer les deux fichiers.

Suivez les étapes pour compiler, empaqueter et exécuter le code

1. Accédez au répertoire contenant le fichier `pom.xml`, puis compilez le projet à l'aide de la commande `mvn` suivante :

```
Console
```

```
mvn compile
```

2. Empaquez le code compilé dans son format distribuable :

```
Console
```

```
mvn package
```

3. Exécutez la commande `mvn` suivante pour exécuter l'application :

```
Console
```

```
mvn exec:java -D exec.mainClass=com.blobs.quickstart.App -D
exec.cleanupDaemonThreads=false
```

Pour simplifier l'étape d'exécution, vous pouvez ajouter `exec-maven-plugin` et `pom.xml`, et configurer comme indiqué ci-dessous :

```
XML
```

```
<plugin>
 <groupId>org.codehaus.mojo</groupId>
 <artifactId>exec-maven-plugin</artifactId>
 <version>1.4.0</version>
 <configuration>
 <mainClass>com.blobs.quickstart.App</mainClass>
 <cleanupDaemonThreads>false</cleanupDaemonThreads>
```

```
</configuration>
</plugin>
```

Avec cette configuration, vous pouvez exécuter l'application avec la commande suivante :

Console

```
mvn exec:java
```

La sortie de l'application est similaire à l'exemple suivant (valeurs UUID omises pour une meilleure lisibilité) :

Sortie

```
Azure Blob Storage - Java quickstart sample

Uploading to Blob storage as blob:

https://mystorageacct.blob.core.windows.net/quickstartblobsUUID/quickstartUU
ID.txt

Listing blobs...
 quickstartUUID.txt

Downloading blob to
 ./data/quickstartUUIDDOWNLOAD.txt

Press the Enter key to begin clean up

Deleting blob container...
Deleting the local source and downloaded files...
Done
```

Avant de commencer le processus de nettoyage, vérifiez la présence des deux fichiers dans votre dossier *data*. Vous pouvez les comparer et constater qu'ils sont identiques.

## Nettoyer les ressources

Une fois que vous avez vérifié les fichiers et terminé le test, appuyez sur la touche **Entrée** pour supprimer les fichiers de test avec le conteneur que vous avez créé dans le compte de stockage. Vous pouvez également utiliser [Azure CLI](#) pour supprimer des ressources.

## Étapes suivantes

Dans ce démarrage rapide, vous avez appris à charger, télécharger et répertorier des objets blob avec Java.

Pour afficher des exemples d'applications de stockage blob, passez à :

### Exemples de bibliothèque Stockage Blob Azure pour Java

- Pour plus d'informations, consultez [Bibliothèques de client Stockage Blob Azure pour Java](#).
- Pour obtenir des tutoriels, des exemples, des guides de démarrage rapide et autre documentation, consultez [Azure pour les développeurs Java](#).

# Démarrage rapide : bibliothèque cliente Stockage File d'attente Azure pour Java

Article • 19/07/2023

Familiarisez-vous avec la bibliothèque de client Storage File d'attente Azure pour Java. Le Stockage File d'attente Azure est un service permettant de stocker un grand nombre de messages dans le but de les récupérer et de les traiter plus tard. Suivez les étapes suivantes pour installer le package et essayer un exemple de code pour les tâches de base.

[Documentation de référence sur l'API](#) | [Code source de la bibliothèque](#) | [Package \(Maven\)](#) | [Exemples](#)

Utilisez la bibliothèque de client Stockage File d'attente Azure pour Java afin d'effectuer les opérations suivantes :

- Créer une file d'attente
- Ajouter des messages à une file d'attente
- Afficher un aperçu des messages d'une file d'attente
- Mettre à jour un message dans une file d'attente
- Obtention de la longueur de la file d'attente
- Recevoir les messages d'une file d'attente
- Supprimer des messages d'une file d'attente
- Suppression d'une file d'attente

## Prérequis

- [Kit de développement Java \(JDK\)](#), version 8 ou ultérieure
- [Apache Maven](#)
- Abonnement Azure : [créez-en un gratuitement](#)
- Compte de stockage Azure : [créez un compte de stockage](#)

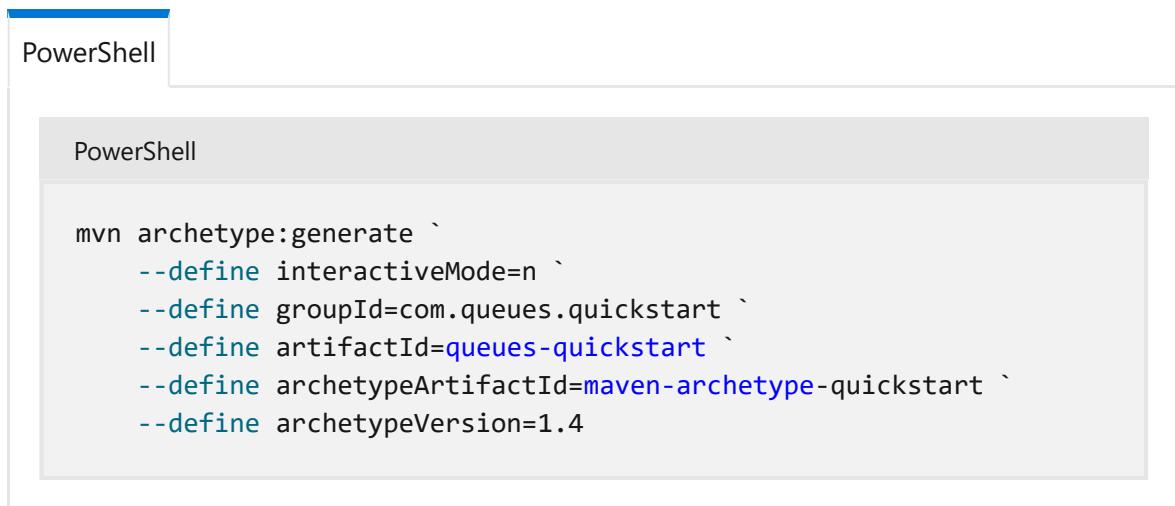
## Configuration

Cette section vous guide tout au long de la préparation d'un projet à utiliser avec la bibliothèque de client Stockage File d'attente Azure pour Java.

## Créer le projet

Créez une application Java nommée *queues-quickstart*.

1. Dans une fenêtre de console (par exemple cmd, PowerShell ou Bash), utilisez Maven pour créer une application de console nommée *queues-quickstart*. Tapez la commande `mvn` suivante pour créer un projet Java « hello world ! ».



```
PowerShell

mvn archetype:generate \
 --define interactiveMode=n \
 --define groupId=com.queues.quickstart \
 --define artifactId=queues-quickstart \
 --define archetypeArtifactId=maven-archetype-quickstart \
 --define archetypeVersion=1.4
```

2. Le résultat de la génération du projet doit ressembler à ceci :



```
Console

[INFO] Scanning for projects...
[INFO]
[INFO] -----< org.apache.maven:standalone-pom >-----
[INFO] Building Maven Stub Project (No POM) 1
[INFO] -----[pom]-----
[INFO]
[INFO] >>> maven-archetype-plugin:3.1.2:generate (default-cli) >
generate-sources @ standalone-pom >>>
[INFO]
[INFO] <<< maven-archetype-plugin:3.1.2:generate (default-cli) <
generate-sources @ standalone-pom <<<
[INFO]
[INFO]
[INFO] --- maven-archetype-plugin:3.1.2:generate (default-cli) @
standalone-pom ---
[INFO] Generating project in Batch mode
[INFO] -----<----->-----

[INFO] Using following parameters for creating project from Archetype:
maven-archetype-quickstart:1.4
[INFO] -----<----->-----

[INFO] Parameter: groupId, Value: com.queues.quickstart
[INFO] Parameter: artifactId, Value: queues-quickstart
[INFO] Parameter: version, Value: 1.0-SNAPSHOT
[INFO] Parameter: package, Value: com.queues.quickstart
[INFO] Parameter: packageInPathFormat, Value: com/queues/quickstart
```

```
[INFO] Parameter: version, Value: 1.0-SNAPSHOT
[INFO] Parameter: package, Value: com.queues.quickstart
[INFO] Parameter: groupId, Value: com.queues.quickstart
[INFO] Parameter: artifactId, Value: queues-quickstart
[INFO] Project created from Archetype in dir:
C:\quickstarts\queues\queues-quickstart
[INFO] -----

[INFO] BUILD SUCCESS
[INFO] -----

[INFO] Total time: 6.394 s
[INFO] Finished at: 2019-12-03T09:58:35-08:00
[INFO] -----

```

3. Basculez dans le répertoire *queues-quickstart* créé.

```
Console
```

```
cd queues-quickstart
```

## Installer les packages

Ouvrez le fichier `pom.xml` dans votre éditeur de texte.

Ajoutez **azure-sdk-bom** pour établir une dépendance sur la dernière version de la bibliothèque. Dans l'extrait de code suivant, remplacez l'espace réservé `{bom_version_to_target}` par le numéro de version. L'utilisation d'**azure-sdk-bom** vous empêche de devoir spécifier la version de chaque dépendance individuelle. Pour en savoir plus sur la nomenclature, consultez le [fichier Lisez-moi de la nomenclature du SDK Azure](#).

```
XML
```

```
<dependencyManagement>
 <dependencies>
 <dependency>
 <groupId>com.azure</groupId>
 <artifactId>azure-sdk-bom</artifactId>
 <version>{bom_version_to_target}</version>
 <type>pom</type>
 <scope>import</scope>
 </dependency>
 </dependencies>
</dependencyManagement>
```

Ajoutez ensuite les éléments dependency suivants au groupe de dépendances. La dépendance `azure-identity` est nécessaire pour les connexions sans mot de passe aux services Azure.

XML

```
<dependency>
 <groupId>com.azure</groupId>
 <artifactId>azure-storage-queue</artifactId>
</dependency>
<dependency>
 <groupId>com.azure</groupId>
 <artifactId>azure-identity</artifactId>
</dependency>
```

## Configurer le framework d'application

À partir du répertoire de projet :

1. Accédez au répertoire `/src/main/java/com/queues/quickstart`
2. Ouvrez le fichier `App.java` dans votre éditeur
3. Supprimez l'instruction `System.out.println("Hello, world");`
4. Ajoutez des directives `import`.

Voici le code :

Java

```
package com.queues.quickstart;

/**
 * Azure Queue Storage client library quickstart
 */
import com.azure.identity.*;
import com.azure.storage.queue.*;
import com.azure.storage.queue.models.*;
import java.io.*;

public class App
{
 public static void main(String[] args) throws IOException
 {
 // Quickstart code goes here
 }
}
```

# Authentification auprès d'Azure

Les requêtes d'application vers les Services Azure doivent être autorisées. L'utilisation de la classe `DefaultAzureCredential` fournie par la bibliothèque de client Azure Identity est l'approche recommandée pour implémenter des connexions sans mot de passe aux services Azure dans votre code.

Vous pouvez également autoriser directement les requêtes adressées aux services Azure à l'aide de mots de passe, de chaînes de connexion ou d'autres informations d'identification. Toutefois, cette approche doit être utilisée avec prudence. Les développeurs doivent être vigilants pour ne jamais exposer les secrets dans un emplacement non sécurisé. Toute personne ayant accès au mot de passe ou à la clé secrète est en mesure de s'authentifier. `DefaultAzureCredential` offre des avantages améliorés en matière de gestion et de sécurité par rapport à la clé de compte pour autoriser l'authentification sans mot de passe. Les deux options sont illustrées dans l'exemple suivant.

## Sans mot de passe (recommandé)

`DefaultAzureCredential` est une classe fournie par la bibliothèque de client Azure Identity pour Java. Pour en savoir plus sur `DefaultAzureCredential`, consultez la vue d'ensemble de [DefaultAzureCredential](#). `DefaultAzureCredential` prend en charge plusieurs méthodes d'authentification et détermine quelle méthode doit être utilisée au moment de l'exécution. Cette approche permet à votre application d'utiliser différentes méthodes d'authentification dans différents environnements (local ou production) sans implémenter de code spécifique à l'environnement.

Par exemple, votre application peut s'authentifier à l'aide de vos informations d'identification de connexion Azure CLI lors du développement local, puis utiliser une [identité managée](#) une fois qu'elle a été déployée sur Azure. Aucune modification du code n'est requise pour cette transition.

Lors du développement localement, assurez-vous que le compte d'utilisateur qui accède aux données de file d'attente dispose des autorisations appropriées. Vous aurez besoin du **Contributeur aux données de file d'attente de stockage** pour lire et écrire des données blob. Pour vous attribuer ce rôle, vous aurez besoin du rôle **Administrateur de l'accès utilisateur** ou d'un autre rôle qui inclut l'action **Microsoft.Authorization/roleAssignments/write**. Vous pouvez attribuer des rôles RBAC Azure à un utilisateur à l'aide du Portail Azure, Azure CLI ou Azure PowerShell. Vous pouvez en savoir plus sur les étendues disponibles pour les attributions de rôles dans la page [vue d'ensemble de l'étendue](#).

Dans ce scénario, vous allez attribuer des autorisations à votre compte d'utilisateur, étendues au compte de stockage, pour suivre le [Principe des priviléges minimum](#). Cette pratique offre aux utilisateurs uniquement les autorisations minimales nécessaires et crée des environnements de production plus sécurisés.

L'exemple suivant attribue le rôle **Contributeur aux données file d'attente du stockage** à votre compte d'utilisateur, qui fournit à la fois un accès en lecture et en écriture aux données file d'attente dans votre compte de stockage.

### ⓘ Important

Dans la plupart des cas, la propagation de l'attribution de rôle dans Azure peut prendre une ou deux minutes, mais dans de rares cas, cela peut prendre jusqu'à huit minutes. Si vous recevez des erreurs d'authentification lorsque vous exécutez votre code pour la première fois, patientez quelques instants et réessayez.

#### Azure portal

1. Dans le Portail Azure, recherchez votre compte de stockage à l'aide de la barre de recherche principale ou de la navigation gauche.
2. Dans la page vue d'ensemble du compte de stockage, sélectionnez **Contrôle d'accès (IAM)** dans le menu de gauche.
3. Sur la page **Contrôle d'accès (IAM)**, sélectionnez l'onglet **Attributions de rôles**.
4. Sélectionnez **+ Ajouter** dans le menu supérieur, puis **Ajouter une attribution de rôle** dans le menu déroulant résultant.

The screenshot shows the Azure Storage Access Control (IAM) interface. On the left, there's a sidebar with various options like Overview, Activity log, Tags, Diagnose and solve problems, and Access Control (IAM). The Access Control (IAM) option is highlighted with a red box. The main area has a header with 'Search (Ctrl+ /)', '+ Add', 'Download role assignments', 'Edit columns', 'Refresh', 'Remove', and 'Got feedback?'. Below the header, there's a menu with 'Add role assignment' (which is also highlighted with a red box), 'Add co-administrator', 'Roles', 'Deny assignments', and 'Classic administrators'. A section titled 'My access' allows viewing access levels. Another section, 'Check access', reviews access levels for users, groups, or service principals. There's a search bar for 'Search by name or email address'. To the right, there are two main sections: 'Grant access to this resource' (with a 'Add role assignment' button) and 'View deny assignments' (with a 'View' button).

5. Utilisez la zone de recherche pour filtrer les résultats sur le rôle souhaité. Pour cet exemple, recherchez *Contributeur aux données file d'attente du stockage*, sélectionnez le résultat correspondant, puis choisissez **Suivant**.
6. Sous **Attribuer l'accès à**, sélectionnez **Utilisateur, groupe ou principal de service**, puis sélectionnez + **Sélectionner des membres**.
7. Dans la boîte de dialogue, recherchez votre nom d'utilisateur Microsoft Entra (généralement votre adresse e-mail *user@domain*), puis choisissez **Sélectionner** en bas de la boîte de dialogue.
8. Sélectionnez **Vérifier + affecter** pour accéder à la page finale, puis **Vérifier + attribuer** à nouveau pour terminer le processus.

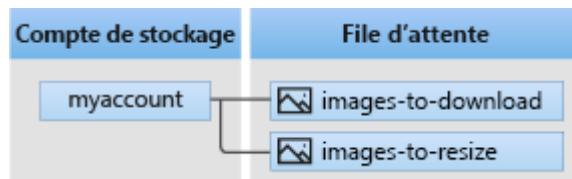
## Modèle objet

Stockage File d'attente Azure est un service permettant de stocker un grand nombre de messages. La taille maximale d'un message de file d'attente est de 64 Ko. Une file d'attente peut contenir des millions de messages, dans la limite de la capacité totale d'un compte de stockage. Les files d'attente sont couramment utilisées pour créer un backlog de travail à traiter de façon asynchrone. Le Stockage File d'attente offre trois types de ressources :

- **Compte de stockage** : Tous les accès à Azure Storage passent par un compte de stockage. Pour plus d'informations sur les comptes de stockage, consultez [Vue d'ensemble des comptes de stockage](#).

- **File d'attente** : une file d'attente contient un ensemble de messages. Tous les messages doivent être dans une file d'attente. Notez que le nom de la file d'attente doit être en minuscules. Pour plus d'informations sur l'affectation de noms à des files d'attente, consultez [Affectation de noms pour les files d'attente et les métadonnées](#).
- **Message** : message dans n'importe quel format d'une taille maximale de 64 Ko. Un message peut rester dans la file d'attente pendant un maximum de 7 jours. Pour les versions du 29 juillet 2017 ou ultérieures, la durée de vie maximale peut être n'importe quel nombre positif, ou -1 indiquant que le message n'expire pas. Si ce paramètre est omis, la valeur par défaut de la durée de vie est de sept jours.

Le diagramme suivant montre la relation entre ces ressources.



Utilisez les classes Java suivantes pour interagir avec ces ressources :

- [QueueClientBuilder](#) : la classe `QueueClientBuilder` configure et instancie un objet `QueueClient`.
- [QueueServiceClient](#) : `QueueServiceClient` vous permet de gérer toutes les files d'attente de votre compte de stockage.
- [QueueClient](#) : la classe `QueueClient` vous permet de gérer et de manipuler une file d'attente individuelle et ses messages.
- [QueueMessageItem](#) : la classe `QueueMessageItem` représente les objets individuels retournés lors de l'appel de `ReceiveMessages` dans une file d'attente.

## Exemples de code

Ces exemples d'extraits de code vous montrent comment effectuer les actions suivantes avec la bibliothèque de client Stockage File d'attente Azure pour Java :

- [Autoriser l'accès et créer un objet client](#)
- [Créer une file d'attente](#)
- [Ajouter des messages à une file d'attente](#)
- [Afficher un aperçu des messages d'une file d'attente](#)
- [Mettre à jour un message dans une file d'attente](#)
- [Obtention de la longueur de la file d'attente](#)
- [Recevoir et supprimer des messages d'une file d'attente](#)
- [Supprimer une file d'attente](#)

Sans mot de passe (recommandé)

## Autoriser l'accès et créer un objet client

Vérifiez que vous êtes authentifié avec le même compte Microsoft Entra auquel vous avez attribué le rôle. Vous pouvez vous authentifier via Azure CLI, Visual Studio Code ou Azure PowerShell.

Azure CLI

Connectez-vous à Azure via Azure CLI à l'aide de la commande suivante :

Azure CLI

```
az login
```

Une fois authentifié, vous pouvez créer et autoriser un objet `QueueClient` à l'aide de `DefaultAzureCredential` pour accéder aux données de file d'attente dans le compte de stockage. `DefaultAzureCredential` découvre et utilise automatiquement le compte avec lequel vous vous êtes connecté à l'étape précédente.

Pour autoriser l'utilisation `DefaultAzureCredential`, vérifiez que vous avez ajouté la dépendance `azure-identity` dans `pom.xml`, comme décrit dans [Installer les packages](#). Veillez également à ajouter une directive d'importation pour `com.azure.identity` dans le fichier `App.java` :

Java

```
import com.azure.identity.*;
```

Choisissez un nom pour la file d'attente et créez une instance de la classe `QueueClient`, en utilisant `DefaultAzureCredential` pour l'autorisation. Nous utilisons cet objet client pour créer et interagir avec la ressource de file d'attente dans le compte de stockage.

### ⓘ Important

Les noms de file d'attente peuvent contenir uniquement des lettres minuscules, des chiffres et des traits d'union, et doivent commencer par une

lettre ou un nombre. Chaque trait d'union doit être précédé et suivi d'un caractère autre qu'un tiret. Le nom doit avoir entre 3 et 63 caractères. Pour plus d'informations sur le nommage des files d'attente, consultez [Nommage des files d'attente et des métadonnées](#).

Ajoutez ce code à l'intérieur de la `main` méthode et veillez à remplacer la valeur de l'espace `<storage-account-name>` réservé :

Java

```
System.out.println("Azure Queue Storage client library - Java quickstart sample\n");

// Create a unique name for the queue
String queueName = "quickstartqueues-" + java.util.UUID.randomUUID();

// Instantiate a QueueClient
// We'll use this client object to create and interact with the queue
// TODO: replace <storage-account-name> with the actual name
QueueClient queueClient = new QueueClientBuilder()
 .endpoint("https://<storage-account-
name>.queue.core.windows.net/")
 .queueName(queueName)
 .credential(new DefaultAzureCredentialBuilder().build())
 .buildClient();
```

## ⓘ Notes

Les messages que vous envoyez à l'aide de la classe `QueueClient` doivent être dans un format pouvant être inclus dans une requête XML avec encodage UTF-8. Si vous le souhaitez, vous pouvez définir l'option `QueueMessageEncoding` sur `BASE64` pour gérer les messages non conformes.

## Créer une file d'attente

Avec l'objet `QueueClient`, appelez la méthode `create` pour créer la file d'attente dans votre compte de stockage.

Ajoutez ce code à la fin de la méthode `main` :

Java

```
System.out.println("Creating queue: " + queueName);
```

```
// Create the queue
queueClient.create();
```

## Ajouter des messages à une file d'attente

L'extrait de code suivant ajoute des messages à la file d'attente en appelant la méthode [sendMessage](#). Il enregistre également un [SendMessageResult](#) retourné à partir d'un appel de `sendMessage`. Le résultat est utilisé pour mettre à jour le message ultérieurement dans le programme.

Ajoutez ce code à la fin de la méthode `main` :

Java

```
System.out.println("\nAdding messages to the queue...");

// Send several messages to the queue
queueClient.sendMessage("First message");
queueClient.sendMessage("Second message");

// Save the result so we can update this message later
SendMessageResult result = queueClient.sendMessage("Third message");
```

## Afficher un aperçu des messages d'une file d'attente

Affichez un aperçu des messages de la file d'attente en appelant la méthode [peekMessages](#). Cette méthode récupère un ou plusieurs messages du début de la file d'attente, mais ne modifie pas la visibilité du message.

Ajoutez ce code à la fin de la méthode `main` :

Java

```
System.out.println("\nPeek at the messages in the queue...");

// Peek at messages in the queue
queueClient.peekMessages(10, null, null).forEach(
 peekedMessage -> System.out.println("Message: " +
 peekedMessage.getMessageText()));
```

## Mettre à jour un message dans une file d'attente

Mettez à jour le contenu d'un message en appelant la méthode [updateMessage](#). Cette méthode peut changer le contenu et le délai d'expiration de la visibilité d'un message.

Le contenu du message doit être une chaîne encodée en UTF-8 d'une taille maximale de 64 Ko. Avec le nouveau contenu du message, transmettez l'ID du message et la réception POP en utilisant le `SendMessageResult` qui a été enregistré dans le code. L'ID du message et la réception POP identifient le message à mettre à jour.

Java

```
System.out.println("\nUpdating the third message in the queue...");

// Update a message using the result that
// was saved when sending the message
queueClient.updateMessage(result.getMessageId(),
 result.getPopReceipt(),
 "Third message has been updated",
 Duration.ofSeconds(1));
```

## Obtention de la longueur de la file d'attente

Vous pouvez obtenir une estimation du nombre de messages dans une file d'attente.

La méthode `getProperties` retourne plusieurs valeurs, y compris le nombre de messages qui se trouvent dans une file d'attente. Ce nombre est approximatif étant donné que des messages peuvent être ajoutés ou supprimés après votre demande. La méthode `getApproximateMessageCount` retourne la dernière valeur récupérée par l'appel à `getProperties`, sans appeler le Stockage File d'attente.

Java

```
QueueProperties properties = queueClient.getProperties();
long messageCount = properties.getApproximateMessagesCount();

System.out.println(String.format("Queue length: %d", messageCount));
```

## Recevoir et supprimer des messages d'une file d'attente

Téléchargez les messages ajoutés en appelant la méthode `receiveMessages`. L'exemple de code supprime également les messages de la file d'attente une fois qu'ils sont reçus et traités. Dans ce cas, le traitement affiche simplement le message sur la console.

Avant de recevoir et de supprimer les messages, l'application s'interrompt dans l'attente d'une entrée de l'utilisateur en appelant `System.console().readLine()`. Vérifiez dans votre [portail Azure](#) que les ressources ont été créées correctement avant d'être

supprimées. Les messages qui ne sont pas supprimés explicitement redeviennent visibles dans la file d'attente et peuvent éventuellement être de nouveau traités.

Ajoutez ce code à la fin de la méthode `main` :

Java

```
System.out.println("\nPress Enter key to receive messages and delete them
from the queue...");
System.console().readLine();

// Get messages from the queue
queueClient.receiveMessages(10).forEach(
 // "Process" the message
 receivedMessage -> {
 System.out.println("Message: " + receivedMessage.getMessageText());

 // Let the service know we're finished with
 // the message and it can be safely deleted.
 queueClient.deleteMessage(receivedMessage.getMessageId(),
 receivedMessage.getPopReceipt());
 }
);
```

Lorsque vous appelez la méthode `receiveMessages`, vous pouvez éventuellement spécifier une valeur pour `maxMessages`, qui correspond au nombre de messages à récupérer dans la file d'attente. La valeur par défaut est de 1 message et la valeur maximale est de 32 messages. Vous pouvez également spécifier une valeur pour `visibilityTimeout`, qui masque les messages aux autres opérations pendant la période d'expiration. La valeur par défaut est 30 secondes.

## Suppression d'une file d'attente

Le code suivant nettoie les ressources créées par l'application en supprimant la file d'attente avec la méthode [Delete](#).

Ajoutez ce code à la fin de la méthode `main` :

Java

```
System.out.println("\nPress Enter key to delete the queue...");
System.console().readLine();

// Clean up
System.out.println("Deleting queue: " + queueClient.getQueueName());
queueClient.delete();
```

```
System.out.println("Done");
```

## Exécuter le code

Cette application crée trois messages et les ajoute à une file d'attente Azure. Le code liste les messages dans la file d'attente, puis les récupère et les supprime avant de supprimer la file d'attente.

Dans la fenêtre de votre console, accédez au répertoire de l'application, puis générez et exécutez l'application.

```
Console
```

```
mvn compile
```

Ensuite, générez le package.

```
Console
```

```
mvn package
```

Utilisez la commande `mvn` suivante pour exécuter l'application.

```
Console
```

```
mvn exec:java -Dexec.mainClass="com.queues.quickstart.App" -
Dexec.cleanupDaemonThreads=false
```

La sortie de l'application ressemble à l'exemple suivant :

```
Sortie
```

```
Azure Queue Storage client library - Java quickstart sample
```

```
Adding messages to the queue...
```

```
Peek at the messages in the queue...
```

```
Message: First message
```

```
Message: Second message
```

```
Message: Third message
```

```
Updating the third message in the queue...
```

```
Press Enter key to receive messages and delete them from the queue...
```

```
Message: First message
Message: Second message
Message: Third message has been updated
```

```
Press Enter key to delete the queue...
```

```
Deleting queue: quickstartqueues-fbf58f33-4d5a-41ac-ac0e-1a05d01c7003
Done
```

Quand l'application s'interrompt avant de recevoir des messages, vérifiez votre compte de stockage dans le [portail Azure](#). Vérifiez que les messages se trouvent dans la file d'attente.

Appuyez sur la touche `Enter` pour recevoir et supprimer les messages. Quand vous y êtes invité, réappuyez sur la touche `Enter` pour supprimer la file d'attente et terminer la démonstration.

## Étapes suivantes

Dans ce guide de démarrage rapide, vous avez appris à créer une file d'attente et à y ajouter des messages à l'aide de code Java. Ensuite, vous avez appris à afficher un aperçu des messages, à les récupérer et à les supprimer. Enfin, vous avez appris à supprimer une file d'attente de messages.

Pour obtenir des tutoriels, des exemples, des guides de démarrage rapide et d'autres documents, visitez :

### Azure pour les développeurs cloud Java

- Pour obtenir des exemples de code associés utilisant des Kits de développement logiciel (SDK) Java version 8 déconseillés, consultez les [Exemples de code utilisant Java version 8](#).
- Pour d'autres exemples d'applications Stockage File d'attente Azure, consultez [Exemples de bibliothèques de client Stockage File d'attente Azure pour Java](#).

# Tutoriel : Déployer une application Spring sur Azure Spring Apps avec une connexion sans mot de passe à une base de données Azure

Article • 23/10/2023

Cet article explique comment utiliser des connexions sans mot de passe aux bases de données Azure dans les applications Spring Boot déployées sur Azure Spring Apps.

Dans ce tutoriel, vous allez effectuer les tâches suivantes à l'aide du portail Azure ou d'Azure CLI. Les deux méthodes sont expliquées dans les procédures suivantes.

- ✓ Provisionnez une instance d'Azure Spring Apps.
- ✓ Créez et déployez des applications sur Azure Spring Apps.
- ✓ Exécutez des applications connectées à des bases de données Azure à l'aide d'une identité managée.

## ⓘ Notes

Ce didacticiel ne fonctionne pas pour R2DBC.

## Prérequis

- Un abonnement Azure. Si vous n'en avez pas encore, créez un [compte gratuit](#) avant de commencer.
- [Azure CLI](#) 2.45.0 ou version ultérieure requise.
- L'extension Azure Spring Apps. Vous pouvez installer l'extension à l'aide de la commande : `az extension add --name spring`.
- [Kit de développement Java \(JDK\)](#), version 8, 11 ou 17.
- Un client [Git](#).
- [cURL](#) ou un utilitaire HTTP similaire pour tester la fonctionnalité.
- Client de ligne de commande MySQL si vous choisissez d'exécuter Azure Database pour MySQL. Vous pouvez vous connecter à votre serveur avec Azure Cloud Shell à l'aide d'un outil client populaire, l'outil [en ligne de commande mysql.exe](#). Vous pouvez aussi utiliser la ligne de commande de `mysql` dans votre environnement local.
- [ODBC Driver 18 pour SQL Server](#) si vous choisissez d'exécuter Azure SQL Database.

# Préparer l'environnement de travail

Tout d'abord, configurez des variables d'environnement à l'aide des commandes suivantes :

Bash

```
export AZ_RESOURCE_GROUP=passwordless-tutorial-rg
export AZ_DATABASE_SERVER_NAME=<YOUR_DATABASE_SERVER_NAME>
export AZ_DATABASE_NAME=demodb
export AZ_LOCATION=<YOUR_AZURE_REGION>
export AZ_SPRING_APPS_SERVICE_NAME=<YOUR_AZURE_SPRING_APPS_SERVICE_NAME>
export AZ_SPRING_APPS_APP_NAME=hellospring
export AZ_DB_ADMIN_USERNAME=<YOUR_DB_ADMIN_USERNAME>
export AZ_DB_ADMIN_PASSWORD=<YOUR_DB_ADMIN_PASSWORD>
export AZ_USER_IDENTITY_NAME=<YOUR_USER_ASSIGNED_MANAGEMED_IDENTITY_NAME>
```

Remplacez les espaces réservés par les valeurs suivantes, qui sont utilisées dans cet article :

- <YOUR\_DATABASE\_SERVER\_NAME> : nom de votre serveur Azure Database, qui doit être unique sur Azure.
- <YOUR\_AZURE\_REGION> : région Azure que vous souhaitez utiliser. Vous pouvez utiliser `eastus` par défaut, mais nous vous recommandons de configurer une région plus proche de l'endroit où vous vivez. Vous pouvez voir la liste complète des régions disponibles à l'aide `az account list-locations` de .
- <YOUR\_AZURE\_SPRING\_APPS\_SERVICE\_NAME> : nom de votre instance Azure Spring Apps. Le nom doit comporter entre 4 et 32 caractères, et contenir uniquement des lettres minuscules, des chiffres et des traits d'union. Le premier caractère du nom du service doit être une lettre, et le dernier doit être une lettre ou un chiffre.
- <AZ\_DB\_ADMIN\_USERNAME> : nom d'utilisateur administrateur de votre serveur de base de données Azure.
- <AZ\_DB\_ADMIN\_PASSWORD> : mot de passe administrateur de votre serveur de base de données Azure.
- <YOUR\_USER\_ASSIGNED\_MANAGEMED\_IDENTITY\_NAME> : Le nom du serveur d'identité managée affectée par l'utilisateur, qui doit être unique dans tout Azure.

# Provisionner une instance d'Azure Spring Apps

Procédez comme suit pour provisionner une instance d'Azure Spring Apps.

1. Mettez à jour Azure CLI avec l'extension Azure Spring Apps à l'aide de la commande suivante :

Azure CLI

```
az extension update --name spring
```

2. Connectez-vous à Azure CLI et choisissez votre abonnement actif à l'aide des commandes suivantes :

Azure CLI

```
az login
az account list --output table
az account set --subscription <name-or-ID-of-subscription>
```

3. Utilisez les commandes suivantes pour créer un groupe de ressources pour contenir votre service Azure Spring Apps et une instance du service Azure Spring Apps :

Azure CLI

```
az group create \
 --name $AZ_RESOURCE_GROUP \
 --location $AZ_LOCATION
az spring create \
 --resource-group $AZ_RESOURCE_GROUP \
 --name $AZ_SPRING_APPS_SERVICE_NAME
```

## Créer une instance de base de données Azure

Procédez comme suit pour provisionner une instance Azure Database.

Azure Database pour MySQL

1. Créez un serveur Azure Database pour MySQL à l'aide de la commande suivante :

Azure CLI

```
az mysql flexible-server create \
 --resource-group $AZ_RESOURCE_GROUP \
 --name $AZ_DATABASE_SERVER_NAME \
 --location $AZ_LOCATION \
 --admin-user $AZ_DB_ADMIN_USERNAME \
 --admin-password $AZ_DB_ADMIN_PASSWORD
```

```
--admin-password $AZ_DB_ADMIN_PASSWORD \
--yes
```

### ⓘ Notes

Si vous ne fournissez pas ou ne fournissez `admin-user` pas de `admin-password` paramètres, le système génère un utilisateur administrateur par défaut ou un mot de passe administrateur aléatoire par défaut.

1. Créez une base de données à l'aide de la commande suivante :

Azure CLI

```
az mysql flexible-server db create \
--resource-group $AZ_RESOURCE_GROUP \
--database-name $AZ_DATABASE_NAME \
--server-name $AZ_DATABASE_SERVER_NAME
```

## Créer une application avec un point de terminaison public affecté

Utilisez la commande suivante pour créer l'application.

Azure CLI

```
az spring app create \
--resource-group $AZ_RESOURCE_GROUP \
--service $AZ_SPRING_APPS_SERVICE_NAME \
--name $AZ_SPRING_APPS_APP_NAME \
--runtime-version=Java_17
--assign-endpoint true
```

## Connecter Azure Spring Apps à la base de données Azure

Tout d'abord, installez l'extension [sans mot de passe du service Connecter or](#) pour Azure CLI :

Azure CLI

```
az extension add --name serviceconnector-passwordless --upgrade
```

## Azure Database pour MySQL

Utilisez ensuite la commande suivante pour créer une identité managée affectée par l'utilisateur pour l'authentification Microsoft Entra. Pour plus d'informations, consultez [Configurer l'authentification Microsoft Entra pour Azure Database pour MySQL - Serveur flexible](#).

## Azure CLI

```
export AZ_IDENTITY_RESOURCE_ID=$(az identity create \
 --name $AZ_USER_IDENTITY_NAME \
 --resource-group $AZ_RESOURCE_GROUP \
 --query id \
 --output tsv)
```

### ⓘ Important

Après avoir créé l'identité affectée par l'utilisateur, demandez à votre *Administrateur général* ou *Administrateur de rôle privilégié* d'accorder les autorisations suivantes pour cette identité : `User.Read.All`, `GroupMember.Read.All` et `Application.Read.ALL`. Pour plus d'informations, consultez la section [Autorisations d'Authentification Active Directory](#).

Ensuite, utilisez la commande suivante pour créer une connexion sans mot de passe à la base de données.

## Azure CLI

```
az spring connection create mysql-flexible \
 --resource-group $AZ_RESOURCE_GROUP \
 --service $AZ_SPRING_APPS_SERVICE_NAME \
 --app $AZ_SPRING_APPS_APP_NAME \
 --target-resource-group $AZ_RESOURCE_GROUP \
 --server $AZ_DATABASE_SERVER_NAME \
 --database $AZ_DATABASE_NAME \
 --system-identity mysql-identity-id=$AZ_IDENTITY_RESOURCE_ID
```

Cette commande de connecteur de services effectue les tâches suivantes en arrière-plan :

- Activez l'identité managée affectée par le système pour l'application `$AZ_SPRING_APPS_APP_NAME` hébergée par Azure Spring Apps.
- Définissez l'administrateur Microsoft Entra sur l'utilisateur connecté actuel.
- Ajoutez un utilisateur de base de données nommé `$AZ_SPRING_APPS_SERVICE_NAME/apps/$AZ_SPRING_APPS_APP_NAME` pour l'identité managée créée à l'étape 1 et accordez tous les privilèges de la base de données `$AZ_DATABASE_NAME` à cet utilisateur.
- Ajoutez deux configurations à l'application `$AZ_SPRING_APPS_APP_NAME` : `spring.datasource.url` et `spring.datasource.username`.

#### ⓘ Notes

Si le message d'erreur `The subscription is not registered to use Microsoft.ServiceLinker` apparaît, exécutez la commande `az provider register --namespace Microsoft.ServiceLinker` pour enregistrer le fournisseur de ressources du connecteur de services, puis exécutez à nouveau la commande de connexion.

## Génération et déploiement de l'application

Les étapes suivantes décrivent comment télécharger, configurer, générer et déployer l'exemple d'application.

1. Utilisez la commande suivante pour cloner l'exemple de référentiel de code :

Azure Database pour MySQL

Bash

```
git clone https://github.com/Azure-Samples/quickstart-spring-data-jdbc-mysql passwordless-sample
```

2. Ajoutez la dépendance suivante à votre *fichier pom.xml* :

Azure Database pour MySQL

### XML

```
<dependency>
 <groupId>com.azure.spring</groupId>
 <artifactId>spring-cloud-azure-starter-jdbc-mysql</artifactId>
</dependency>
```

Cette dépendance ajoute la prise en charge du démarrage d'Azure Spring Cloud.

### ⚠ Notes

Pour plus d'informations sur la gestion des versions de bibliothèque Spring Cloud Azure à l'aide d'une facture de documents (BOM), consultez la section **Prise en main** du guide du développeur Azure Spring Cloud.

3. Utilisez la commande suivante pour mettre à jour le *fichier application.properties* :

#### Azure Database pour MySQL

##### Bash

```
cat << EOF > passwordless-
sample/src/main/resources/application.properties

logging.level.org.springframework.jdbc.core=DEBUG
spring.datasource.azure.passwordless-enabled=true
spring.sql.init.mode=always

EOF
```

4. Utilisez les commandes suivantes pour générer le projet à l'aide de Maven :

##### Bash

```
cd passwordless-sample
./mvnw clean package -DskipTests
```

5. Utilisez la commande suivante pour déployer le *fichier target/demo-0.0.1-SNAPSHOTjar* pour l'application :

##### Azure CLI

```
az spring app deploy \
--name $AZ_SPRING_APPS_APP_NAME \
--service $AZ_SPRING_APPS_SERVICE_NAME \
--resource-group $AZ_RESOURCE_GROUP \
--artifact-path target/demo-0.0.1-SNAPSHOT.jar
```

6. Interrogez l'état de l'application après le déploiement à l'aide de la commande suivante :

Azure CLI

```
az spring app list \
--service $AZ_SPRING_APPS_SERVICE_NAME \
--resource-group $AZ_RESOURCE_GROUP \
--output table
```

Vous devez voir la sortie similaire à l'exemple suivant.

Name	Location	ResourceGroup	Production Deployment Provisioning	Deployment Registration
Public Url				
Status	CPU	Memory	Running Instance	Registered Instance
Persistent Storage				
<app name>	eastus	<resource group>	default	
Succeeded	1	2	1/1	0/1
-				

## Test de l'application

Pour tester l'application, vous pouvez utiliser cURL. Tout d'abord, créez un élément « todo » dans la base de données à l'aide de la commande suivante :

Bash

```
curl --header "Content-Type: application/json" \
--request POST \
--data '{"description":"configuration","details":"congratulations, you
have set up JDBC correctly!","done": "true"}' \
https://${AZ_SPRING_APPS_SERVICE_NAME}-
hellospring.azuremicroservices.io
```

Cette commande retourne l'élément créé, comme illustré dans l'exemple suivant :

JSON

```
{"id":1,"description":"configuration","details":"congratulations, you have
set up JDBC correctly!","done":true}
```

Ensuite, récupérez les données à l'aide de la requête cURL suivante :

Bash

```
curl https://${AZ_SPRING_APPS_SERVICE_NAME}-
hellospring.azuremicroservices.io
```

Cette commande retourne la liste des éléments « todo », y compris l'élément que vous avez créé, comme illustré dans l'exemple suivant :

JSON

```
[{"id":1,"description":"configuration","details":"congratulations, you have
set up JDBC correctly!","done":true}]
```

## Nettoyer les ressources

Pour propre toutes les ressources utilisées pendant ce didacticiel, supprimez le groupe de ressources à l'aide de la commande suivante :

Azure CLI

```
az group delete \
--name $AZ_RESOURCE_GROUP \
--yes
```

## Étapes suivantes

- Documentation Spring Cloud Azure

# Utiliser une identité managée pour connecter Azure SQL Database à une application déployée dans Azure Spring Apps

Article • 02/02/2024

## ⓘ Notes

Azure Spring Apps est le nouveau nom du service Azure Spring Cloud. Bien que le service ait un nouveau nom, vous verrez l'ancien nom à divers endroits pendant un certain temps, car nous travaillons à mettre à jour les ressources telles que les captures d'écran, les vidéos et les diagrammes.

Cet article s'applique à :  Java  C#

Cet article s'applique au : Niveau  De base/Standard  Entreprise

Cet article explique comment créer une identité managée pour une application déployée sur Azure Spring Apps et l'utiliser pour accéder à Azure SQL Database.

[Azure SQL Database](#) est le service de bases de données relationnelles, évolutif et intelligent, conçu pour le cloud. Il est toujours à jour, avec des fonctionnalités automatisées et alimentées par l'IA qui optimisent les performances et la durabilité. Les options de stockage Hyperscale et de calcul serverless mettent automatiquement à l'échelle les ressources à la demande, ce qui vous permet de vous concentrer sur la création de nouvelles applications sans vous soucier de la taille du stockage ou de la gestion des ressources.

## Prérequis

- Compte Azure avec un abonnement actif. [Créez un compte gratuitement](#).
- [Azure CLI](#) version 2.45.0 ou ultérieure.
- Suivez le [tutoriel](#) Spring Data JPA pour provisionner une base de données Azure SQL et l'utiliser localement avec une application Java.
- Suivez le didacticiel [sur l'identité managée affectée par le](#) système Azure Spring Apps pour provisionner une application dans Azure Spring Apps avec une identité managée activée.

# Se connecter à Azure SQL Database avec une identité managée

Vous pouvez connecter votre application à une base de données Azure SQL avec une identité managée en suivant des étapes manuelles ou en utilisant [service Connecter or.](#)

## Configuration manuelle

## Accorder l'autorisation à l'identité managée

Connectez-vous à votre serveur SQL et exéutez la requête SQL suivante :

SQL

```
CREATE USER [<managed-identity-name>] FROM EXTERNAL PROVIDER;
ALTER ROLE db_datareader ADD MEMBER [<managed-identity-name>];
ALTER ROLE db_datawriter ADD MEMBER [<managed-identity-name>];
ALTER ROLE db_ddladmin ADD MEMBER [<managed-identity-name>];
GO
```

La valeur de l'espace réservé <managed-identity-name> suit la règle <service-instance-name>/apps/<app-name> ; par exemple : myspringcloud/apps/sqldemo . Vous pouvez également utiliser la commande suivante pour interroger le nom de l'identité managée avec Azure CLI :

Azure CLI

```
az ad sp show --id <identity-object-ID> --query displayName
```

# Configurer votre application Java pour utiliser une identité managée

Ouvrez le fichier `src/main/resources/application.properties`, puis ajoutez `Authentication=ActiveDirectoryMSI;` à la fin de la ligne `spring.datasource.url`, comme illustré dans l'exemple suivant. Veillez à utiliser la valeur appropriée pour la variable `$AZ DATABASE NAME`.

## properties

```
spring.datasource.url=jdbc:sqlserver://$AZ_DATABASE_NAME.database.windows.net:1433;database=demo;encrypt=true;trustServerCertificate=false;hostNameInCertificate=*.database.windows.net;loginTimeout=30;errorTimeout=30
```

```
ameInCertificate=*.database.windows.net;loginTimeout=30;Authentication=A
ctiveDirectoryMSI;
```

# Générer et déployer l'application sur Azure Spring Apps

Regénérez l'application et déployez-la sur l'application Azure Spring Apps provisionnée lors de la deuxième puce de la section Prérequis. Vous disposez maintenant d'une application Spring Boot authentifiée par une identité managée qui utilise JPA pour stocker et récupérer des données à partir d'une base de données Azure SQL dans Azure Spring Apps.

## Étapes suivantes

- Comment accéder à un objet blob de stockage avec une identité managée dans Azure Spring Apps ↗
- Guide pratique pour activer une identité managée affectée par le système pour des applications dans Azure Spring Apps
- Que sont les identités managées pour les ressources Azure ?
- Authentifier Azure Spring Apps avec Key Vault dans GitHub Actions

# Connecter une instance Azure Database pour MySQL à votre application dans Azure Spring Apps

Article • 02/02/2024

## ⓘ Notes

Azure Spring Apps est le nouveau nom du service Azure Spring Cloud. Bien que le service ait un nouveau nom, vous verrez l'ancien nom à divers endroits pendant un certain temps, car nous travaillons à mettre à jour les ressources telles que les captures d'écran, les vidéos et les diagrammes.

Cet article s'applique à : Java C#

Cet article s'applique au : Niveau De base/Standard Entreprise

Avec Azure Spring Apps, vous pouvez connecter automatiquement les services Azure sélectionnés à vos applications, au lieu de devoir configurer votre application Spring Boot manuellement. Cet article explique comment connecter votre application à votre instance de Azure Database pour MySQL.

## Prérequis

- Une application déployée dans Azure Spring Apps. Pour plus d'informations, consultez [Démarrage rapide : Déployer votre première application sur Azure Spring Apps](#).
- Une instance de Serveur flexible Azure Database pour MySQL.
- [Azure CLI](#) version 2.45.0 ou ultérieure.

## Préparation du projet

Java

1. Dans le fichier *pom.xml* de votre projet, ajoutez la dépendance suivante :

XML

```
<dependency>
 <groupId>org.springframework.boot</groupId>
 <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
<dependency>
 <groupId>com.azure.spring</groupId>
 <artifactId>spring-cloud-azure-starter-jdbc-mysql</artifactId>
</dependency>
```

2. Dans le fichier *application.properties*, supprimez toutes les propriétés `spring.datasource.*`.
3. Mettez à jour l'application actuelle en exécutant `az spring app deploy` ou créez un déploiement pour ce changement en exécutant `az spring app deployment create`.

## Connecter votre application à l'instance de Azure Database pour MySQL

Connecteur de service

### ⓘ Notes

Par défaut, les Connecteurs de service sont créés au niveau de l'application. Pour remplacer les connexions, vous pouvez créer d'autres connexions dans les déploiements.

Suivez ces étapes pour configurer votre application Spring pour vous connecter à un serveur flexible Azure Database pour MySQL avec une identité managée affectée par le système.

1. Utilisez la commande suivante pour installer l'extension sans mot de passe du service Connecteur ou pour Azure CLI.

Azure CLI

```
az extension add --name serviceconnector-passwordless --upgrade
```

2. Ensuite, utilisez la commande suivante pour créer une identité managée affectée par l'utilisateur pour l'authentification Microsoft Entra. Veillez à remplacer les variables de l'exemple par des valeurs réelles. Pour plus d'informations, consultez [Configurer l'authentification Microsoft Entra pour Azure Database pour MySQL : Serveur flexible](#).

Azure CLI

```
export AZ_IDENTITY_RESOURCE_ID=$(az identity create \
--name $AZURE_USER_IDENTITY_NAME \
--resource-group $AZURE_IDENTITY_RESOURCE_GROUP \
--query id \
--output tsv)
```

3. Exécutez la `az spring connection create` commande, comme illustré dans l'exemple suivant. Veillez à remplacer les variables de l'exemple par des valeurs réelles.

Azure CLI

```
az spring connection create mysql-flexible \
--resource-group $AZURE_SPRING_APPS_RESOURCE_GROUP \
--service $AZURE_SPRING_APPS_SERVICE_INSTANCE_NAME \
--app $APP_NAME \
--target-resource-group $MYSQL_RESOURCE_GROUP \
--server $MYSQL_SERVER_NAME \
--database $DATABASE_NAME \
--system-identity mysql-identity-id=$AZ_IDENTITY_RESOURCE_ID
```

## Étapes suivantes

Dans cet article, vous avez appris à connecter une application dans Azure Spring Apps à une instance de Azure Database pour MySQL. Pour en savoir plus sur la connexion de services à une application, consultez [Connecter une base de données Azure Cosmos DB à une application dans Azure Spring Apps](#).

# Lier une instance Azure Database pour PostgreSQL à votre application dans Azure Spring Apps

Article • 07/02/2024

## ⓘ Notes

Azure Spring Apps est le nouveau nom du service Azure Spring Cloud. Bien que le service ait un nouveau nom, vous verrez l'ancien nom à divers endroits pendant un certain temps, car nous travaillons à mettre à jour les ressources telles que les captures d'écran, les vidéos et les diagrammes.

Cet article s'applique à :  Java  C#

Cet article s'applique au : Niveau  De base/Standard  Entreprise

Azure Spring Apps vous permet de lier automatiquement certains services Azure à vos applications au lieu de devoir configurer manuellement votre application Spring Boot. Cet article vous montre comment lier votre application à votre instance Azure Database pour PostgreSQL.

Dans cet article, nous incluons deux méthodes d'authentification : l'authentification Microsoft Entra et l'authentification PostgreSQL. L'onglet Sans mot de passe affiche l'authentification Microsoft Entra et l'onglet Mot de passe affiche l'authentification PostgreSQL.

L'authentification Microsoft Entra est un mécanisme de connexion à Azure Database pour PostgreSQL utilisant les identités définies dans Microsoft Entra ID. Avec l'authentification Microsoft Entra, vous pouvez gérer les identités des utilisateurs de base de données et d'autres services Microsoft dans un emplacement centralisé, ce qui simplifie la gestion des autorisations.

L'authentification PostgreSQL utilise des comptes stockés dans PostgreSQL. Si vous choisissez d'utiliser des mots de passe comme informations d'identification pour les comptes, ces informations d'identification sont stockées dans la table utilisateur. Étant donné que ces mots de passe sont stockés dans PostgreSQL, vous devez gérer la rotation des mots de passe par vous-même.

## Prérequis

- Une application déployée dans Azure Spring Apps. Pour plus d'informations, consultez [Démarrage rapide : Déployer votre première application sur Azure Spring Apps](#).
- Instance Azure Database pour PostgreSQL serveur unique.
- [Azure CLI](#) version 2.45.0 ou ultérieure.

## Préparation du projet

Java

Suivez les étapes ci-après pour préparer votre projet.

1. Dans le fichier *pom.xml* de votre projet, ajoutez la dépendance suivante :

XML

```
<dependency>
 <groupId>org.springframework.boot</groupId>
 <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
<dependency>
 <groupId>com.azure.spring</groupId>
 <artifactId>spring-cloud-azure-starter-jdbc-
postgresql</artifactId>
</dependency>
```

2. Dans le fichier *application.properties*, supprimez toutes les propriétés `spring.datasource.*`.

3. Mettez à jour l'application actuelle en exécutant `az spring app deploy` ou créez un déploiement pour ce changement en exécutant `az spring app deployment create`.

## Lier votre application à l'instance Azure Database pour PostgreSQL

### ⓘ Notes

Les Connecter de service sont créés au niveau du déploiement. Par conséquent, si un autre déploiement est créé, vous devez recréer les connexions.

Sans mot de passe

1. Installez l'extension sans mot de passe [Connecteur de services](#) pour Azure CLI :

Azure CLI

```
az extension add --name serviceconnector-passwordless --upgrade
```

2. Configurez Azure Spring Apps pour vous connecter à la base de données PostgreSQL avec une identité managée affectée par le système à l'aide de la commande `az spring connection create`.

Azure CLI

```
az spring connection create postgres \
--resource-group $AZ_SPRING_APPS_RESOURCE_GROUP \
--service $AZ_SPRING_APPS_SERVICE_INSTANCE_NAME \
--app $APP_NAME \
--deployment $DEPLOYMENT_NAME \
--target-resource-group $POSTGRES_RESOURCE_GROUP \
--server $POSTGRES_SERVER_NAME \
--database $DATABASE_NAME \
--system-identity
```

## Étapes suivantes

Dans cet article, vous avez appris à lier une application dans Azure Spring Apps à une instance Azure Database pour PostgreSQL. Pour en savoir plus sur la liaison de services à une application, consultez [Lier une base de données Azure Cosmos DB à une application dans Azure Spring Apps](#).

# Tutoriel : Se connecter à une base de données MySQL à partir de Java JBoss EAP App Service avec une connexion sans mot de passe

Article • 25/10/2023

Azure App Service offre un service d'hébergement web hautement évolutif appliquant des mises à jour correctives automatiques dans Azure. Il offre également une [identité managée](#) pour votre application, qui constitue une solution clé en main permettant de sécuriser l'accès à [Azure Database pour MySQL](#) et à d'autres services Azure. Les identités managées dans App Service sécurisent votre application en en éliminant les secrets, par exemple les informations d'identification dans les variables d'environnement. Dans ce tutoriel, vous allez apprendre à :

- ✓ Créer une base de données MySQL.
- ✓ Déployez un exemple d'application JBoss EAP pour Azure App Service à l'aide d'un package WAR.
- ✓ Configurer une application web Spring Boot pour utiliser l'authentification Microsoft Entra avec une base de données MySQL.
- ✓ Se connecter à une base de données MySQL avec une identité managée à l'aide d'un connecteur de services.

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

## Prérequis

- [Git](#)
- Java JDK
- Maven
- Azure CLI version 2.46.0 ou ultérieure.
- Extension Azure CLI `serviceconnector-passwordless` version 0.2.2 ou ultérieure.
- `jq`

## Cloner l'exemple d'application de liste de tâches et préparer le référentiel

Exécutez les commandes suivantes dans votre terminal pour cloner l'exemple de dépôt et configurer l'exemple d'environnement d'application.

Bash

```
git clone https://github.com/Azure-Samples>Passwordless-Connections-for-
Java-Apps
cd Passwordless-Connections-for-Java-Apps/JakartaEE/jboss-eap/
```

## Créer une base de données Azure pour MySQL

Suivez ces étapes pour créer un serveur Azure Database pour MySQL dans votre abonnement. L'application Spring Boot se connecte à cette base de données et stocke ses données lors de l'exécution, entraînant la persistance de l'état de l'application, quel que soit l'endroit d'où vous l'exécutez.

1. Connectez-vous à Azure CLI et définissez votre abonnement si vous en avez plusieurs connectés à vos informations d'identification.

Azure CLI

```
az login
az account set --subscription <subscription-ID>
```

2. Créez un groupe de ressources Azure, en notant son nom.

Azure CLI

```
export RESOURCE_GROUP=<resource-group-name>
export LOCATION=eastus

az group create --name $RESOURCE_GROUP --location $LOCATION
```

3. Créez un serveur Azure Database pour MySQL. Le serveur est créé avec un compte administrateur, mais il n'est pas utilisé, car nous utilisons le compte administrateur Microsoft Entra pour effectuer les tâches administratives.

Azure CLI

```
export MYSQL_ADMIN_USER=azureuser
MySQL admin access rights won't be used because Azure AD
authentication is leveraged to administer the database.
export MYSQL_ADMIN_PASSWORD=<admin-password>
export MYSQL_HOST=<mysql-host-name>
```

```
Create a MySQL server.
az mysql flexible-server create \
 --name $MYSQL_HOST \
 --resource-group $RESOURCE_GROUP \
 --location $LOCATION \
 --admin-user $MYSQL_ADMIN_USER \
 --admin-password $MYSQL_ADMIN_PASSWORD \
 --public-access 0.0.0.0 \
 --tier Burstable \
 --sku-name Standard_B1ms \
 --storage-size 32
```

4. Créez une base de données pour l'application.

Azure CLI

```
export DATABASE_NAME=checklist

az mysql flexible-server db create \
 --resource-group $RESOURCE_GROUP \
 --server-name $MYSQL_HOST \
 --database-name $DATABASE_NAME
```

## Créer une instance App Service

Créez une ressource Azure App Service sur Linux. JBoss EAP nécessite une référence SKU Premium.

Azure CLI

```
export APPSERVICE_PLAN=<app-service-plan>
export APPSERVICE_NAME=<app-service-name>
Create an App Service plan
az appservice plan create \
 --resource-group $RESOURCE_GROUP \
 --name $APPSERVICE_PLAN \
 --location $LOCATION \
 --sku P1V3 \
 --is-linux

Create an App Service resource.
az webapp create \
 --resource-group $RESOURCE_GROUP \
 --name $APPSERVICE_NAME \
 --plan $APPSERVICE_PLAN \
 --runtime "JBOSSEAP:7-java8"
```

# Connecter la base de données MySQL avec une connectivité d'identité

Ensuite, connectez la base de données en utilisant [Service Connector](#).

Installez l'extension sans mot de passe Service Connector pour Azure CLI :

Azure CLI

```
az extension add --name serviceconnector-passwordless --upgrade
```

Ensuite, utilisez la commande suivante pour créer une identité managée affectée par l'utilisateur pour l'authentification Microsoft Entra. Pour plus d'informations, consultez [Configurer l'authentification Microsoft Entra pour Azure Database pour MySQL : Serveur flexible](#).

Azure CLI

```
export USER_IDENTITY_NAME=<your-user-assigned-managed-identity-name>
export IDENTITY_RESOURCE_ID=$(az identity create \
 --name $USER_IDENTITY_NAME \
 --resource-group $RESOURCE_GROUP \
 --query id \
 --output tsv)
```

## ⓘ Important

Après avoir créé l'identité affectée par l'utilisateur, demandez à votre *Administrateur général* ou *Administrateur de rôle privilégié* d'accorder les autorisations suivantes pour cette identité : `User.Read.All`, `GroupMember.Read.All` et `Application.Read.ALL`. Pour plus d'informations, consultez la section [Autorisations d'Authentification Active Directory](#).

Ensuite, connectez votre application à une base de données MySQL avec une identité managée affectée par le système à l'aide de Service Connector. Pour effectuer cette connexion, exécutez la commande [az webapp connection create](#).

Azure CLI

```
az webapp connection create mysql-flexible \
 --resource-group $RESOURCE_GROUP \
 --name $APPSERVICE_NAME \
 --target-resource-group $RESOURCE_GROUP \
```

```
--server $MYSQL_HOST \
--database $DATABASE_NAME \
--system-identity mysql-identity-id=$IDENTITY_RESOURCE_ID \
--client-type java
```

Cette commande de connecteur de services effectue les tâches suivantes en arrière-plan :

- Activer l'identité managée affectée par le système pour l'application `$APPSCERVICE_NAME` hébergée par Azure App Service.
- Définissez l'utilisateur actuellement connecté comme administrateur Microsoft Entra.
- Ajoutez un utilisateur de base de données pour l'identité managée affectée par le système à l'étape 1 et accordez tous les privilèges de la base de données `$DATABASE_NAME` à cet utilisateur. Vous pouvez obtenir le nom d'utilisateur à partir de la chaîne de connexion dans la sortie de la commande précédente.
- Ajoutez une chaîne de connexion aux paramètres de l'application dans l'application nommée `AZURE_MYSQL_CONNECTIONSTRING`.

#### ⓘ Notes

Si le message d'erreur `The subscription is not registered to use Microsoft.ServiceLinker` apparaît, exécutez la commande `az provider register --namespace Microsoft.ServiceLinker` pour enregistrer le fournisseur de ressources du connecteur de services, puis exécutez à nouveau la commande de connexion.

## Déployer l'application

Suivez ces étapes pour préparer des données dans une base de données et déployer l'application.

### Créer un schéma de la base de données

1. Ouvrez un pare-feu pour autoriser la connexion à partir de votre adresse IP actuelle.

```
Create a temporary firewall rule to allow connections from your
current machine to the MySQL server
export MY_IP=$(curl http://whatismyip.akamai.com)
az mysql flexible-server firewall-rule create \
 --resource-group $RESOURCE_GROUP \
 --name $MYSQL_HOST \
 --rule-name AllowCurrentMachineToConnect \
 --start-ip-address ${MY_IP} \
 --end-ip-address ${MY_IP}
```

## 2. Se connecter à la base de données et créer des tables.

Azure CLI

```
export DATABASE_FQDN=${MYSQL_HOST}.mysql.database.azure.com
export CURRENT_USER=$(az account show --query user.name --output tsv)
export RDBMS_ACCESS_TOKEN=$(az account get-access-token \
 --resource-type oss-rdbms \
 --output tsv \
 --query accessToken)
mysql -h "${DATABASE_FQDN}" --user "${CURRENT_USER}" --enable-
cleartext-plugin --password="$RDBMS_ACCESS_TOKEN" < azure/init-db.sql
```

## 3. Supprimez la règle de pare-feu temporaire.

Azure CLI

```
az mysql flexible-server firewall-rule delete \
 --resource-group $RESOURCE_GROUP \
 --name $MYSQL_HOST \
 --rule-name AllowCurrentMachineToConnect
```

# Déployer l'application

## 1. Mettez à jour la chaîne de connexion dans les paramètres de l'application.

Obtenez la chaîne de connexion générée par le connecteur de services et ajoutez le plug-in d'authentification sans mot de passe. Cette chaîne de connexion est référencée dans le script de démarrage.

Azure CLI

```
export PASSWORDLESS_URL=$(\
 az webapp config appsettings list \
 --resource-group $RESOURCE_GROUP \
 --name $APPSERVICE_NAME \
 | jq -c '.[] \
```

```

| select (.name == "AZURE_MYSQL_CONNECTIONSTRING") \
| .value' \
| sed 's//g')
Create a new environment variable with the connection string
including the passwordless authentication plugin
export
PASSWORDLESS_URL=${PASSWORDLESS_URL}'&defaultAuthenticationPlugin=com.azure.identity.extensions.jdbc.mysql.AzureMysqlAuthenticationPlugin&authenticationPlugins=com.azure.identity.extensions.jdbc.mysql.AzureMysqlAuthenticationPlugin'
az webapp config appsettings set \
--resource-group $RESOURCE_GROUP \
--name $APPSERVICE_NAME \
--settings
"AZURE_MYSQL_CONNECTIONSTRING_PASSWORDLESS=${PASSWORDLESS_URL}"

```

2. L'exemple d'application contient un fichier *pom.xml* qui peut générer le fichier WAR. Exécutez la commande suivante pour générer l'application.

Bash

```
mvn clean package -DskipTests
```

3. Déployez le fichier WAR et le script de démarrage sur le service d'application.

Azure CLI

```

az webapp deploy \
--resource-group $RESOURCE_GROUP \
--name $APPSERVICE_NAME \
--src-path target/ROOT.war \
--type war
az webapp deploy \
--resource-group $RESOURCE_GROUP \
--name $APPSERVICE_NAME \
--src-path src/main/webapp/WEB-INF/createMySQLDataSource.sh \
--type startup

```

## Exemple de test d'application web

Exécutez la commande suivante pour tester l'application.

Bash

```

export WEBAPP_URL=$(az webapp show \
--resource-group $RESOURCE_GROUP \
--name $APPSERVICE_NAME \
--query defaultHostName \

```

```
--output tsv)

Create a list
curl -X POST -H "Content-Type: application/json" -d '{"name": "list1", "date": "2022-03-21T00:00:00", "description": "Sample checklist"}' https://${WEBAPP_URL}/checklist

Create few items on the list 1
curl -X POST -H "Content-Type: application/json" -d '{"description": "item 1"}' https://${WEBAPP_URL}/checklist/1/item
curl -X POST -H "Content-Type: application/json" -d '{"description": "item 2"}' https://${WEBAPP_URL}/checklist/1/item
curl -X POST -H "Content-Type: application/json" -d '{"description": "item 3"}' https://${WEBAPP_URL}/checklist/1/item

Get all lists
curl https://${WEBAPP_URL}/checklist

Get list 1
curl https://${WEBAPP_URL}/checklist/1
```

## Nettoyer les ressources

Au cours des étapes précédentes, vous avez créé des ressources Azure au sein d'un groupe de ressources. Si vous ne pensez pas avoir besoin de ces ressources à l'avenir, supprimez le groupe de ressources en exécutant la commande suivante dans Cloud Shell :

Azure CLI

```
az group delete --name myResourceGroup
```

L'exécution de cette commande peut prendre une minute.

## Étapes suivantes

Découvrez-en plus sur l'exécution des applications Java sur App Service sur Linux dans le guide du développeur.

[Guide du développeur Java dans App Service Linux](#)

# Tutoriel : Se connecter à Database pour PostgreSQL à partir d'App Service Java Tomcat sans secrets à l'aide d'une identité managée

Article • 25/10/2023

Azure App Service offre un service d'hébergement web hautement évolutif appliquant des mises à jour correctives automatiques dans Azure. Il offre également une [identité managée](#) pour votre application, qui constitue une solution clé en main permettant de sécuriser l'accès à [Azure Database pour PostgreSQL](#) et à d'autres services Azure. Les identités managées dans App Service sécurisent votre application en en éliminant les secrets, par exemple les informations d'identification dans les variables d'environnement. Dans ce tutoriel, vous allez apprendre à :

- ✓ Créer une base de données PostgreSQL.
- ✓ Déployer l'exemple d'application sur Azure App Service sur Tomcat à l'aide d'un package WAR.
- ✓ Configurer une application web Tomcat pour utiliser l'authentification Microsoft Entra avec une base de données PostgreSQL.
- ✓ Vous connecter à Database pour PostgreSQL avec une identité managée à l'aide d'un connecteur de services.

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

## Prérequis

- [Git](#)
- Java JDK
- Maven
- Azure CLI version 2.45.0 ou ultérieure.

## Cloner l'exemple d'application de liste de tâches et préparer le référentiel

Exécutez les commandes suivantes dans votre terminal pour cloner l'exemple de dépôt et configurer l'exemple d'environnement d'application.

Bash

```
git clone https://github.com/Azure-Samples/Passwordless-Connections-for-Java-Apps
cd Passwordless-Connections-for-Java-Apps/Tomcat/
```

## Créer une base de données Azure pour PostgreSQL

Suivez ces étapes pour créer un serveur Azure Database pour PostgreSQL dans votre abonnement. L'application Tomcat se connecte à cette base de données et stocke ses données lors de l'exécution, entraînant la persistance de l'état de l'application, quel que soit le lieu de l'exécution.

1. Connectez-vous à Azure CLI et définissez votre abonnement si vous en avez plusieurs connectés à vos informations d'identification.

Azure CLI

```
az login
az account set --subscription <subscription-ID>
```

2. Créez un groupe de ressources Azure, en notant son nom.

Azure CLI

```
export RESOURCE_GROUP=<resource-group-name>
export LOCATION=eastus

az group create --name $RESOURCE_GROUP --location $LOCATION
```

3. Créez un serveur Azure Database pour PostgreSQL. Le serveur est créé avec un compte administrateur, mais il n'est pas utilisé, car nous utilisons le compte administrateur Microsoft Entra pour effectuer les tâches administratives.

Serveur flexible

Azure CLI

```
export POSTGRESQL_ADMIN_USER=azureuser
PostgreSQL admin access rights won't be used because Azure AD authentication is leveraged to administer the database.
export POSTGRESQL_ADMIN_PASSWORD=<admin-password>
export POSTGRESQL_HOST=<postgresql-host-name>
```

```
Create a PostgreSQL server.
az postgres flexible-server create \
 --resource-group $RESOURCE_GROUP \
 --name $POSTGRESQL_HOST \
 --location $LOCATION \
 --admin-user $POSTGRESQL_ADMIN_USER \
 --admin-password $POSTGRESQL_ADMIN_PASSWORD \
 --public-access 0.0.0.0 \
 --sku-name Standard_D2s_v3
```

- Créez une base de données pour l'application.

Serveur flexible

Azure CLI

```
export DATABASE_NAME=checklist

az postgres flexible-server db create \
 --resource-group $RESOURCE_GROUP \
 --server-name $POSTGRESQL_HOST \
 --database-name $DATABASE_NAME
```

## Déployer l'application sur App Service

Suivez ces étapes pour générer un fichier WAR et le déployer sur Azure App Service sur Tomcat à l'aide d'un package WAR.

- L'exemple d'application contient un fichier *pom.xml* qui peut générer le fichier WAR. Exécutez la commande suivante pour générer l'application.

Bash

```
mvn clean package -f pom.xml
```

- Créez une ressource Azure App Service sur Linux à l'aide de Tomcat 9.0.

Azure CLI

```
export APPSERVICE_PLAN=<app-service-plan>
export APPSERVICE_NAME=<app-service-name>
Create an App Service plan
az appservice plan create \
 --resource-group $RESOURCE_GROUP \
 --name $APPSERVICE_NAME
```

```
--name $APPSERVICE_PLAN \
--location $LOCATION \
--sku B1 \
--is-linux

Create an App Service resource.
az webapp create \
--resource-group $RESOURCE_GROUP \
--name $APPSERVICE_NAME \
--plan $APPSERVICE_PLAN \
--runtime "TOMCAT:10.0-javal1"
```

### 3. Déployez le package WAR sur App Service.

Azure CLI

```
az webapp deploy \
--resource-group $RESOURCE_GROUP \
--name $APPSERVICE_NAME \
--src-path target/app.war \
--type war
```

## Connecter la base de données pour PostgreSQL avec une connectivité d'identité

Ensuite, connectez la base de données en utilisant [Service Connector](#).

Installez l'extension sans mot de passe Service Connector pour Azure CLI :

Azure CLI

```
az extension add --name serviceconnector-passwordless --upgrade
```

Ensuite, connectez votre application à une base de données PostgreSQL avec une identité managée affectée par le système à l'aide de Service Connector.

Serveur flexible

Pour effectuer cette connexion, exécutez la commande [az webapp connection create](#).

Azure CLI

```
az webapp connection create postgres-flexible \
--resource-group $RESOURCE_GROUP \
```

```
--name $APPSERVICE_NAME \
--target-resource-group $RESOURCE_GROUP \
--server $POSTGRESQL_HOST \
--database $DATABASE_NAME \
--system-identity \
--client-type java
```

Cette commande crée une connexion entre votre application web et votre serveur PostgreSQL, puis gère l'authentification par le biais d'une identité managée affectée par le système.

Ensuite, mettez à jour les paramètres de l'application et ajoutez un plug-in dans la chaîne de connexion

Azure CLI

```
export AZURE_POSTGRESQL_CONNECTIONSTRING=$(\
 az webapp config appsettings list \
 --resource-group $RESOURCE_GROUP \
 --name $APPSERVICE_NAME \
 | jq -c -r '.[] \
 | select (.name == "AZURE_POSTGRESQL_CONNECTIONSTRING") \
 | .value')\

az webapp config appsettings set \
 --resource-group $RESOURCE_GROUP \
 --name $APPSERVICE_NAME \
 --settings 'CATALINA_OPTS=-
DdbUrl=""${AZURE_POSTGRESQL_CONNECTIONSTRING}""
&authenticationPluginClassName=com.azure.identity.extensions.jdbc.postgresql.AzurePostgresqlAuthenticationPlugin'"
```

## Tester l'exemple d'application web

Exécutez la commande suivante pour tester l'application.

Bash

```
export WEBAPP_URL=$(az webapp show \
 --resource-group $RESOURCE_GROUP \
 --name $APPSERVICE_NAME \
 --query defaultHostName \
 --output tsv)

Create a list
curl -X POST -H "Content-Type: application/json" -d '{"name": "list1", "date": "2022-03-21T00:00:00", "description": "Sample checklist"}'
https://${WEBAPP_URL}/checklist
```

```
Create few items on the list 1
curl -X POST -H "Content-Type: application/json" -d '{"description": "item 1"}' https://${WEBAPP_URL}/checklist/1/item
curl -X POST -H "Content-Type: application/json" -d '{"description": "item 2"}' https://${WEBAPP_URL}/checklist/1/item
curl -X POST -H "Content-Type: application/json" -d '{"description": "item 3"}' https://${WEBAPP_URL}/checklist/1/item

Get all lists
curl https://${WEBAPP_URL}/checklist

Get list 1
curl https://${WEBAPP_URL}/checklist/1
```

## Nettoyer les ressources

Au cours des étapes précédentes, vous avez créé des ressources Azure au sein d'un groupe de ressources. Si vous ne pensez pas avoir besoin de ces ressources à l'avenir, supprimez le groupe de ressources en exécutant la commande suivante dans Cloud Shell :

Azure CLI

```
az group delete --name myResourceGroup
```

L'exécution de cette commande peut prendre une minute.

## Étapes suivantes

Découvrez-en plus sur l'exécution des applications Java sur App Service sur Linux dans le guide du développeur.

[Guide du développeur Java dans App Service Linux](#)

Découvrez comment sécuriser votre application avec un domaine personnalisé et un certificat.

[Sécuriser avec un domaine personnalisé et un certificat](#)

# Tutoriel : Se connecter à une base de données PostgreSQL à partir d'une application de conteneur Java Quarkus sans secrets à l'aide d'une identité managée

Article • 09/03/2023

Azure Container Apps offre une [identité managée](#) pour votre application, qui constitue une solution clé en main permettant de sécuriser l'accès à [Azure Database pour PostgreSQL](#) et à d'autres services Azure. Les identités managées dans Container Apps sécurisent votre application en éliminant les secrets de celle-ci, par exemple les informations d'identification dans les variables d'environnement.

Ce tutoriel vous guide tout au long du processus de création, de configuration, de déploiement et de mise à l'échelle des applications de conteneur Java sur Azure. À la fin de ce tutoriel, vous disposerez d'une application [Quarkus](#) stockant des données dans une base de données [PostgreSQL](#) avec une identité managée s'exécutant sur [Container Apps](#).

Contenu :

- ✓ Configurez une application Quarkus pour vous authentifier à l'aide de Microsoft Entra ID avec une base de données PostgreSQL.
- ✓ Créez un registre de conteneurs Azure et envoyez-lui (par push) une image d'application Java.
- ✓ Créez une application de conteneur dans Azure.
- ✓ Créez une base de données PostgreSQL dans Azure.
- ✓ Connectez-vous à une base de données PostgreSQL avec une identité managée à l'aide d'un connecteur de services.

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

## 1. Prérequis

- [Azure CLI](#) : version 2.45.0 ou ultérieure.
- [Git](#)
- [Java JDK](#)

- [Maven](#)
- [Docker](#)
- [GraalVM](#)

## 2. Créer un registre de conteneurs

Créez un groupe de ressources avec la commande `az group create`. Un groupe de ressources Azure est un conteneur logique dans lequel les ressources Azure sont déployées et gérées.

L'exemple suivant crée un groupe de ressources nommé `myResourceGroup` dans la région Azure USA Est.

Azure CLI

```
az group create --name myResourceGroup --location eastus
```

Créez une instance de registre de conteneurs Azure à l'aide de la commande `az acr create`. Le nom du registre doit être unique dans Azure, contenir entre 5 et 50 caractères alphanumériques. Toutes les lettres doivent être indiquées en minuscules. `mycontainerregistry007` 'exemple suivant. Mettez à jour le nom de façon à utiliser une valeur unique.

Azure CLI

```
az acr create \
--resource-group myResourceGroup \
--name mycontainerregistry007 \
--sku Basic
```

## 3. Cloner l'exemple d'application et préparer l'image de conteneur

Ce tutoriel utilise un exemple d'application de liste Fruits avec une interface utilisateur Web qui appelle une API REST Quarkus soutenue par [Azure Database for PostgreSQL](#). Le code pour l'application est disponible sur [GitHub](#). Pour en savoir plus sur l'écriture d'applications Java à l'aide de [Quarkus](#) et de [PostgreSQL](#), consultez le guide Quarkus Hibernate ORM avec Panache et le guide Quarkus Datasource.

Exécutez les commandes suivantes dans votre terminal pour cloner l'exemple de dépôt et configurer l'exemple d'environnement d'application.

```
git
```

```
git clone https://github.com/quarkusio/quarkus-quickstarts
cd quarkus-quickstarts/hibernate-orm-panache-quickstart
```

## Modifier votre projet

1. Ajoutez les dépendances nécessaires au fichier BOM de votre projet.

XML

```
<dependency>
 <groupId>com.azure</groupId>
 <artifactId>azure-identity-providers-jdbc-postgresql</artifactId>
 <version>1.0.0-beta.1</version>
</dependency>
```

2. Configurez les propriétés de l'application Quarkus.

La configuration de Quarkus se trouve dans le fichier `src/main/resources/application.properties`. Ouvrez ce fichier dans votre éditeur et observez plusieurs propriétés par défaut. Les propriétés préfixées avec `%prod` sont utilisées uniquement lorsque l'application est générée et déployée, par exemple lorsqu'elle est déployée sur Azure App Service. Lorsque l'application s'exécute localement, les propriétés `%prod` sont ignorées. De même, les propriétés `%dev` sont utilisées dans le mode Live Coding /Dev de Analysisus, et les propriétés `%test` sont utilisées pendant les tests continus.

Supprimez le contenu existant dans `application.properties` et remplacez-le par ce qui suit pour configurer la base de données pour les modes de développement, de test et de production :

Serveur flexible

properties

```
quarkus.package.type=uber-jar

quarkus.hibernate-orm.database.generation=drop-and-create
quarkus.datasource.db-kind=postgresql
quarkus.datasource.jdbc.max-size=8
quarkus.datasource.jdbc.min-size=2
quarkus.hibernate-orm.log.sql=true
quarkus.hibernate-orm.sql-load-script=import.sql
quarkus.datasource.jdbc.acquisition-timeout = 10
```

```
%dev.quarkus.datasource.username=${AZURE_CLIENT_NAME}
%dev.quarkus.datasource.jdbc.url=jdbc:postgresql://${DBHOST}.postgres.database.azure.com:5432/${DBNAME}?\
authenticationPluginClassName=com.azure.identity.providers.postgresql.AzureIdentityPostgresqlAuthenticationPlugin\
&sslmode=require\
&azure.clientId=${AZURE_CLIENT_ID}\
&azure.clientSecret=${AZURE_CLIENT_SECRET}\
&azure.tenantId=${AZURE_TENANT_ID}

%prod.quarkus.datasource.username=${AZURE_MI_NAME}
%prod.quarkus.datasource.jdbc.url=jdbc:postgresql://${DBHOST}.postgres.database.azure.com:5432/${DBNAME}?\
authenticationPluginClassName=com.azure.identity.providers.postgresql.AzureIdentityPostgresqlAuthenticationPlugin\
&sslmode=require

%dev.quarkus.class-loading.parent-first-artifacts=com.azure:azure-
core::jar,\
com.azure:azure-core-http-netty::jar,\
io.projectreactor.netty:reactor-netty-core::jar,\
io.projectreactor.netty:reactor-netty-http::jar,\
io.netty:netty-resolver-dns::jar,\
io.netty:netty-codec::jar,\
io.netty:netty-codec-http::jar,\
io.netty:netty-codec-http2::jar,\
io.netty:netty-handler::jar,\
io.netty:netty-resolver::jar,\
io.netty:netty-common::jar,\
io.netty:netty-transport::jar,\
io.netty:netty-buffer::jar,\
com.azure:azure-identity::jar,\
com.azure:azure-identity-providers-core::jar,\
com.azure:azure-identity-providers-jdbc-postgresql::jar,\
com.fasterxml.jackson.core:jackson-core::jar,\
com.fasterxml.jackson.core:jackson-annotations::jar,\
com.fasterxml.jackson.core:jackson-databind::jar,\
com.fasterxml.jackson.dataformat:jackson-dataformat-xml::jar,\
com.fasterxml.jackson.datatype:jackson-datatype-jsr310::jar,\
org.reactivestreams:reactive-streams::jar,\
io.projectreactor:reactor-core::jar,\
com.microsoft.azure:msal4j::jar,\
com.microsoft.azure:msal4j-persistence-extension::jar,\
org.codehaus.woodstox:stax2-api::jar,\
com.fasterxml.woodstox:woodstox-core::jar,\
com.nimbusds:oauth2-oidc-sdk::jar,\
com.nimbusds:content-type::jar,\
com.nimbusds:nimbus-jose-jwt::jar,\
net.minidev:json-smart::jar,\
net.minidev:accessors-smart::jar,\
io.netty:netty-transport-native-unix-common::jar
```

# Générer une image Docker et l'envoyer (push) au registre de conteneurs

## 1. Générez l'image de conteneur.

Exécutez la commande suivante pour générer l'image de l'application Quarkus.

Vous devez la marquer avec le nom complet de votre serveur de connexion au registre. Le nom du serveur de connexion est au format `<nom_registro>.azurecr.io` (obligatoirement en minuscules uniquement). Par exemple :

`monregistreconteneurs007.azurecr.io`. Remplacez le nom par votre propre nom de registre.

Bash

```
mvnw quarkus:add-extension -Dextensions="container-image-jib"
mvnw clean package -Pnative -Dquarkus.native.container-build=true -
Dquarkus.container-image.build=true -Dquarkus.container-
image.registry=mycontainerregistry007 -Dquarkus.container-
image.name=quarkus-postgres-passwordless-app -Dquarkus.container-
image.tag=v1
```

## 2. Connectez-vous au registre.

Avant d'envoyer (push) des images conteneur, vous devez vous connecter au registre. Pour ce faire, utilisez la commande `az acr login`. Spécifiez uniquement le nom de la ressource du registre au moment de la connexion avec l'interface Azure CLI. N'utilisez pas le nom complet du serveur de connexion.

Azure CLI

```
az acr login --name <registry-name>
```

Une fois l'opération terminée, la commande retourne un message `Login Succeeded`.

## 3. Envoyez (push) l'image vers le registre.

Utilisez [docker push][docker-push] pour envoyer l'image vers l'instance du registre. Remplacez `mycontainerregistry007` par le nom du serveur de connexion de votre instance de registre. Cet exemple crée le référentiel `quarkus-postgres-passwordless-app` qui contient l'image `quarkus-postgres-passwordless-app:v1`.

Bash

```
docker push mycontainerregistry007/quarkus-postgres-passwordless-app:v1
```

## 4. Créer une application de conteneur sur Azure

1. Créez une instance Container Apps en exécutant la commande suivante. Veillez à remplacer la valeur des variables d'environnement par le nom et l'emplacement réels que vous souhaitez utiliser.

Azure CLI

```
RESOURCE_GROUP="myResourceGroup"
LOCATION="eastus"
CONTAINERAPPS_ENVIRONMENT="my-environment"

az containerapp env create \
 --resource-group $RESOURCE_GROUP \
 --name $CONTAINERAPPS_ENVIRONMENT \
 --location $LOCATION
```

2. Créez une application de conteneur avec votre image d'application en exécutant la commande suivante. Remplacez les espaces réservés par vos valeurs. Pour trouver les détails du compte administrateur du registre de conteneurs, consultez [S'authentifier avec un registre de conteneurs Azure](#)

Azure CLI

```
CONTAINER_IMAGE_NAME=quarkus-postgres-passwordless-app:v1
REGISTRY_SERVER=mycontainerregistry007
REGISTRY_USERNAME=<REGISTRY_USERNAME>
REGISTRY_PASSWORD=<REGISTRY_PASSWORD>

az containerapp create \
 --resource-group $RESOURCE_GROUP \
 --name my-container-app \
 --image $CONTAINER_IMAGE_NAME \
 --environment $CONTAINERAPPS_ENVIRONMENT \
 --registry-server $REGISTRY_SERVER \
 --registry-username $REGISTRY_USERNAME \
 --registry-password $REGISTRY_PASSWORD
```

## 5. Créer et connecter une base de données PostgreSQL avec une connectivité d'identité

Ensuite, créez une base de données PostgreSQL et configurez votre application de conteneur pour vous connecter à une base de données PostgreSQL avec une identité managée affectée par le système. L'application Quarkus se connecte à cette base de données et stocke ses données lors de l'exécution, entraînant la persistance de l'état de l'application, quel que soit l'endroit où vous exécutez cette dernière.

### 1. Créez le service de la base de données.



The screenshot shows the Azure portal interface. A modal window titled "Serveur flexible" is open. Inside, the "Azure CLI" tab is selected, displaying the following command:

```
DB_SERVER_NAME='msdocs-quarkus-postgres-webapp-db'
ADMIN_USERNAME='demoadmin'
ADMIN_PASSWORD='<admin-password>'

az postgres flexible-server create \
 --resource-group $RESOURCE_GROUP \
 --name $DB_SERVER_NAME \
 --location $LOCATION \
 --admin-user $DB_USERNAME \
 --admin-password $DB_PASSWORD \
 --sku-name GP_Gen5_2
```

Les paramètres suivants sont utilisés dans la commande Azure CLI ci-dessus :

- *resource-group* → Utilisez le nom du groupe de ressources dans lequel vous avez créé l'application web, par exemple `msdocs-quarkus-postgres-webapp-rg`.
- *name* → Nom du serveur de base de données PostgreSQL. Ce nom doit être **unique dans Azure** (le point de terminaison de serveur devient `https://<name>.postgres.database.azure.com`). Les caractères autorisés sont `A-Z`, `0-9` et `-`. Une bonne approche consiste à utiliser une combinaison du nom de votre société et d'un identificateur de serveur. (`msdocs-quarkus-postgres-webapp-db`)
- *location* → Utilisez l'emplacement que vous avez utilisé pour l'application web.
- *admin-user* → Nom d'utilisateur du compte administrateur. Ce ne peut pas être `azure_superuser`, `admin`, `administrator`, `root`, `guest` ou `public`. Par exemple, `demoadmin` convient.
- *admin-password* → Mot de passe de l'utilisateur administrateur. Il doit contenir entre 8 et 128 caractères de trois des catégories suivantes : Lettres majuscules,

lettres minuscules, chiffres et caractères non alphanumériques.

### ⓘ Important

Lorsque vous créez des noms d'utilisateur ou des mots de passe, **n'utilisez pas le caractère \$**. Plus tard dans ce tutoriel, vous créerez des variables d'environnement avec ces valeurs où le caractère \$ a une signification spéciale dans le conteneur Linux utilisé pour exécuter des applications Java.

- *public-access* → `None` qui définit le serveur en mode d'accès public sans règles de pare-feu. Les règles seront créées à une étape ultérieure.
  - *sku-name* → Nom du niveau tarifaire et de la configuration de calcul, par exemple `GP_Gen5_2`. Pour plus d'informations, consultez [Niveaux tarifaires d'Azure Database pour PostgreSQL](#).
1. Créez une base de données nommée `fruits` dans le service PostgreSQL avec cette commande :

```
Serveur flexible

Azure CLI

az postgres flexible-server db create \
 --resource-group $RESOURCE_GROUP \
 --server-name $DB_SERVER_NAME \
 --database-name fruits
```

2. Installez l'extension sans mot de passe [Connecteur de services](#) pour Azure CLI :

```
Azure CLI

az extension add --name serviceconnector-passwordless --upgrade
```

3. Connectez la base de données à l'application de conteneur avec une identité managée affectée par le système à l'aide de la commande de connexion.

```
Serveur flexible

Azure CLI
```

```
az containerapp connection create postgres-flexible \
 --resource-group $RESOURCE_GROUP \
 --name my-container-app \
 --target-resource-group $RESOURCE_GROUP \
 --server $DB_SERVER_NAME \
 --database fruits \
 --managed-identity
```

## 6. Passer en revue les modifications apportées

Vous pouvez trouver l'URL (FQDN) de l'application à l'aide de la commande suivante :

Azure CLI

```
az containerapp list --resource-group $RESOURCE_GROUP
```

Lorsque la nouvelle page web affiche votre liste de fruits, votre application se connecte à la base de données à l'aide de l'identité managée. Vous devez désormais être en mesure de modifier la liste de fruits comme auparavant.

## Nettoyer les ressources

Au cours des étapes précédentes, vous avez créé des ressources Azure au sein d'un groupe de ressources. Si vous ne pensez pas avoir besoin de ces ressources à l'avenir, supprimez le groupe de ressources en exécutant la commande suivante dans Cloud Shell :

Azure CLI

```
az group delete --name myResourceGroup
```

L'exécution de cette commande peut prendre une minute.

## Étapes suivantes

Découvrez-en plus sur l'exécution des applications Java sur Azure dans le guide du développeur.

[Azure pour les développeurs Java](#)

# Se connecter à Azure SQL Database et l'interroger en utilisant Node.js et le package npm mssql

Article • 26/10/2023

S'applique à  Azure SQL Database

Ce guide de démarrage rapide explique comment connecter une application à une base de données dans Azure SQL Database et effectuer des requêtes avec Node.js et mssql. Ce guide de démarrage rapide suit l'approche sans mot de passe recommandée pour se connecter à la base de données.

## Connexions sans mot de passe pour les développeurs

Les connexions sans mot de passe offrent un mécanisme plus sécurisé pour accéder aux ressources Azure. Les étapes générales suivantes sont utilisées pour se connecter à une base de données Azure SQL à l'aide de connexions sans mot de passe dans cet article :

- Préparez votre environnement pour l'authentification sans mot de passe.
  - Pour un environnement local : votre identité personnelle est utilisée. Cette identité peut être extraite d'un IDE, d'une CLI ou d'autres outils de développement locaux.
  - Pour un environnement cloud : une [identité managée](#) est utilisée.
- Authentifiez-vous dans l'environnement à l'aide des `DefaultAzureCredential` de la bibliothèque Azure Identity pour obtenir des identifiants vérifiés.
- Utilisez les identifiants vérifiés pour créer des objets client du kit de développement logiciel (SDK) Azure pour l'accès aux ressources.

Vous pouvez en apprendre plus sur les connexions sans mot de passe sur le [Hub des connexions sans mot de passe](#).

## Prérequis

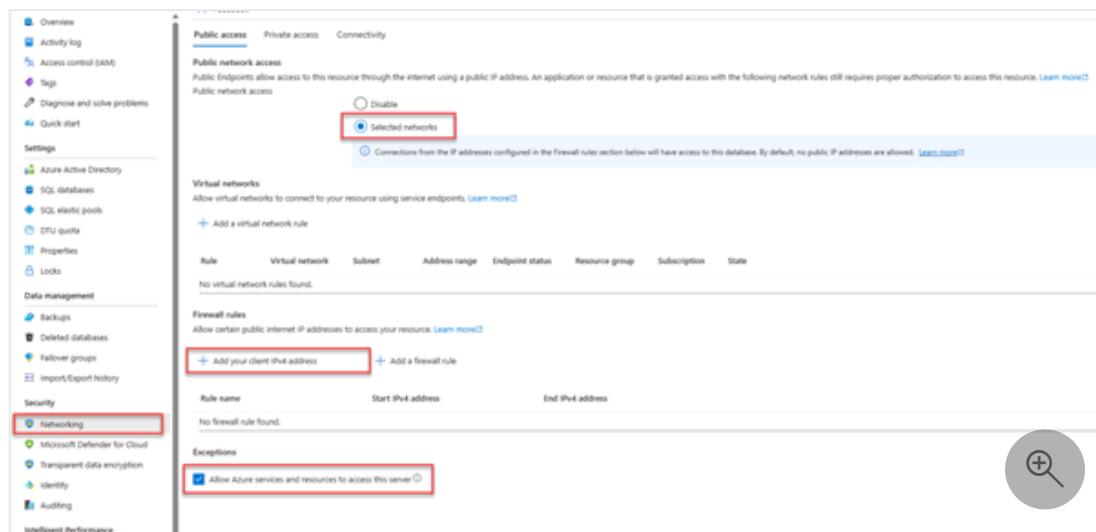
- Un [abonnement Azure](#)
- Une base de données dans Azure SQL Database configurée l'authentification avec Microsoft Entra ID ([anciennement Azure Active Directory](#)). Vous pouvez en créer une à l'aide du [Guide de démarrage rapide Créez une base de données](#).

- Interpréteur de commandes avec Bash
- [Node.js LTS ↗](#)
- [Visual Studio Code ↗](#)
- [Extension App Service pour Visual Studio Code ↗](#)
- La version la plus récente de l'interface [Azure CLI](#)

## Configurer le serveur de base de données

Les connexions sécurisées et sans mot de passe à Azure SQL Database nécessitent certaines configurations de base de données. Vérifiez les paramètres suivants sur votre [serveur logique dans Azure](#) pour vous connecter correctement à Azure SQL Database dans les environnements locaux et hébergés :

1. Pour les connexions de développement local, vérifiez que votre serveur logique est configuré pour autoriser l'adresse IP de votre ordinateur local et d'autres services Azure à se connecter :
  - Accédez à la page **Réseau** de votre serveur.
  - Activez la case d'option **Réseaux sélectionnés** pour voir des options de configuration supplémentaires.
  - Sélectionnez **Ajouter l'adresse IPv4 de votre client (xx.xx.xx.xx.xx.xx)** pour ajouter une règle de pare-feu qui activera les connexions à partir de l'adresse IPv4 de votre ordinateur local. Vous pouvez également sélectionner **+ Ajouter une règle de pare-feu** pour entrer une adresse IP spécifique de votre choix.
  - Vérifiez que la case **Autoriser les services et les ressources Azure à accéder à ce serveur** est cochée.



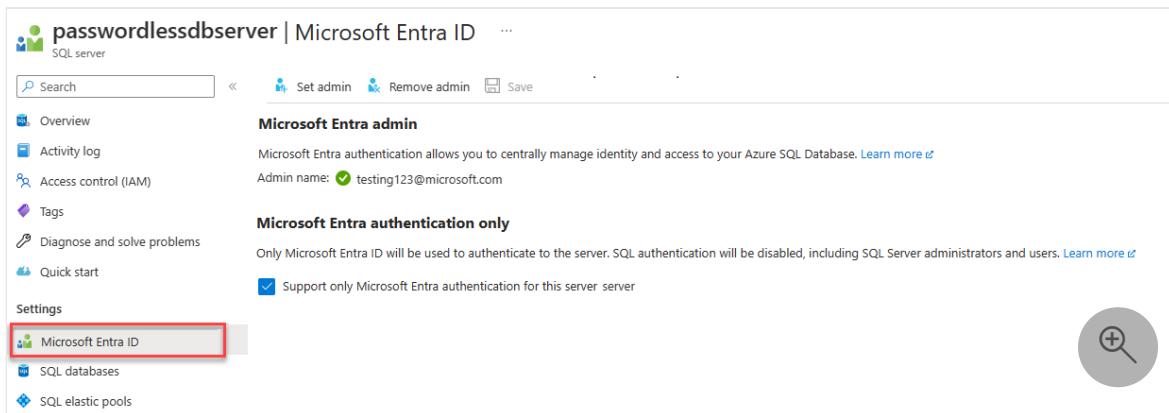
## **⚠️ Avertissement**

L'activation du paramètre **Autoriser les services et les ressources Azure à accéder à ce serveur** n'est pas une pratique de sécurité recommandée pour les scénarios de production. Les applications réelles doivent implémenter des approches plus sécurisées, telles que des restrictions de pare-feu ou des configurations de réseau virtuel plus strictes.

Vous pouvez en apprendre davantage sur les configurations de sécurité de base de données avec les ressources suivantes :

- **Configurer des règles de pare-feu Azure SQL Database.**
- **Configurer un réseau virtuel avec des points de terminaison privés.**

2. Le serveur doit également activer l'authentification Microsoft Entra et disposer d'un compte d'administrateur Microsoft Entra affecté. Pour les connexions de développement local, le compte d'administrateur de Microsoft Entra doit être un compte que vous pouvez également connecter localement à Visual Studio ou à Azure CLI. Vous pouvez vérifier si l'authentification Microsoft Entra est activée sur la page **Microsoft Entra ID** de votre serveur logique.



The screenshot shows the Microsoft Entra ID settings page for a SQL server named 'passwordlessdbserver'. The left sidebar lists various options like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Quick start, and Settings. Under Settings, the 'Microsoft Entra ID' option is selected and highlighted with a red box. The main content area is titled 'Microsoft Entra admin' and describes how Microsoft Entra authentication allows central management of identity and access to the Azure SQL Database. It shows the 'Admin name' as 'testing123@microsoft.com'. Below this, the 'Microsoft Entra authentication only' section is shown, stating that only Microsoft Entra ID will be used for authentication, and SQL authentication will be disabled. A checkbox labeled 'Support only Microsoft Entra authentication for this server' is checked. At the bottom right of the page is a search icon.

3. Si vous utilisez un compte Azure personnel, assurez-vous d'avoir **Microsoft Entra installé et configuré dans la base de données Azure SQL** afin d'assigner votre compte en tant qu'administrateur de serveur. En revanche, si vous utilisez un compte d'entreprise, il est fort probable que Microsoft Entra ID soit déjà configuré pour vous.

# Créer le projet

Les étapes de cette section créent une API REST Node.js.

1. Créez un répertoire pour le projet et accédez-y.

2. Initialisez le projet en exécutant la commande suivante dans le terminal :

```
Bash
```

```
npm init -y
```

3. Installez les packages requis utilisés dans l'exemple de code de cet article :

```
Bash
```

```
npm install mssql swagger-ui-express yamljs
```

4. Installez le package de développement utilisé dans l'exemple de code de cet article :

```
Bash
```

```
npm install --save-dev dotenv
```

5. Ouvrez le projet dans Visual Studio Code.

```
Bash
```

```
code .
```

6. Ouvrez le fichier **package.json** et ajoutez la propriété et la valeur suivantes après la propriété *name* pour configurer le projet pour les modules ESM.

```
JSON
```

```
"type": "module",
```

## Créer le code d'application Express.js

Pour créer l'application OpenAPI Express.js, vous allez créer plusieurs fichiers :

 Agrandir le tableau

Fichier	Description
.env.development	Fichier d'environnement de développement local uniquement.

Fichier	Description
index.js	Fichier d'application principal, qui démarre l'application Express.js sur le port 3000.
person.js	Fichier de l'API de routage Express.js <b>/person</b> pour gérer les opérations CRUD.
openapi.js	Route Express.js <b>/api-docs</b> pour l'interface utilisateur de l'explorateur OpenAPI. Root redirige vers cette route.
openApiSchema.yml	Fichier de schéma OpenAPI 3.0 qui définit l'API Person.
config.js	Fichier de configuration pour lire les variables d'environnement et construire l'objet de connexion mssql approprié.
database.js	Classe de base de données pour gérer les opérations CRUD Azure SQL à l'aide du package npm <b>mssql</b> .
./vscode/settings.json	Ignorer les fichiers du modèle Glob durant le déploiement.

1. Créez un fichier **index.js** et ajoutez-y le code suivant :

JavaScript

```

import express from 'express';
import { config } from './config.js';
import Database from './database.js';

// Import App routes
import person from './person.js';
import openapi from './openapi.js';

const port = process.env.PORT || 3000;

const app = express();

// Development only - don't do in production
// Run this to create the table in the database
if (process.env.NODE_ENV === 'development') {
 const database = new Database(config);
 database
 .executeQuery(
 `CREATE TABLE Person (id int NOT NULL IDENTITY, firstName
varchar(255), lastName varchar(255));`
)
 .then(() => {
 console.log('Table created');
 })
 .catch((err) => {
 // Table may already exist
 console.error(`Error creating table: ${err}`);
 });
}

```

```

}

// Connect App routes
app.use('/api-docs', openapi);
app.use('/persons', person);
app.use('*', (_, res) => {
 res.redirect('/api-docs');
});

// Start the server
app.listen(port, () => {
 console.log(`Server started on port ${port}`);
});

```

2. Créez un fichier de routage **person.js** et ajoutez-y le code suivant :

JavaScript

```

import express from 'express';
import { config } from './config.js';
import Database from './database.js';

const router = express.Router();
router.use(express.json());

// Development only - don't do in production
console.log(config);

// Create database object
const database = new Database(config);

router.get('/', async (_, res) => {
 try {
 // Return a list of persons
 const persons = await database.readAll();
 console.log(`persons: ${JSON.stringify(persons)}`);
 res.status(200).json(persons);
 } catch (err) {
 res.status(500).json({ error: err?.message });
 }
});

router.post('/' , async (req, res) => {
 try {
 // Create a person
 const person = req.body;
 console.log(`person: ${JSON.stringify(person)}`);
 const rowsAffected = await database.create(person);
 res.status(201).json({ rowsAffected });
 } catch (err) {
 res.status(500).json({ error: err?.message });
 }
});

```

```
router.get('/:id', async (req, res) => {
 try {
 // Get the person with the specified ID
 const personId = req.params.id;
 console.log(`personId: ${personId}`);
 if (personId) {
 const result = await database.read(personId);
 console.log(`persons: ${JSON.stringify(result)}`);
 res.status(200).json(result);
 } else {
 res.status(404);
 }
 } catch (err) {
 res.status(500).json({ error: err?.message });
 }
});

router.put('/:id', async (req, res) => {
 try {
 // Update the person with the specified ID
 const personId = req.params.id;
 console.log(`personId: ${personId}`);
 const person = req.body;

 if (personId && person) {
 delete person.id;
 console.log(`person: ${JSON.stringify(person)}`);
 const rowsAffected = await database.update(personId, person);
 res.status(200).json({ rowsAffected });
 } else {
 res.status(404);
 }
 } catch (err) {
 res.status(500).json({ error: err?.message });
 }
});

router.delete('/:id', async (req, res) => {
 try {
 // Delete the person with the specified ID
 const personId = req.params.id;
 console.log(`personId: ${personId}`);

 if (!personId) {
 res.status(404);
 } else {
 const rowsAffected = await database.delete(personId);
 res.status(204).json({ rowsAffected });
 }
 } catch (err) {
 res.status(500).json({ error: err?.message });
 }
});
```

```
export default router;
```

3. Créez un fichier de routage **openapi.js** et ajoutez-y le code suivant pour l'explorateur de l'interface utilisateur OpenAPI :

JavaScript

```
import express from 'express';
import { join, dirname } from 'path';
import swaggerUi from 'swagger-ui-express';
import yaml from 'yamljs';
import { fileURLToPath } from 'url';

const __dirname = dirname(fileURLToPath(import.meta.url));

const router = express.Router();
router.use(express.json());

const pathToSpec = join(__dirname, './openApiSchema.yml');
const openApiSpec = yaml.load(pathToSpec);

router.use('/', swaggerUi.serve, swaggerUi.setup(openApiSpec));

export default router;
```

4. Créez un fichier de schéma **openApiSchema.yml** et ajoutez-y le contenu YAML suivant :

yml

```
openapi: 3.0.0
info:
 version: 1.0.0
 title: Persons API
paths:
 /persons:
 get:
 summary: Get all persons
 responses:
 '200':
 description: OK
 content:
 application/json:
 schema:
 type: array
 items:
 $ref: '#/components/schemas/Person'
 post:
 summary: Create a new person
 requestBody:
```

```
 required: true
 content:
 application/json:
 schema:
 $ref: '#/components/schemas/Person'
responses:
 '201':
 description: Created
 content:
 application/json:
 schema:
 $ref: '#/components/schemas/Person'
/persons/{id}:
parameters:
- name: id
 in: path
 required: true
 schema:
 type: integer
get:
 summary: Get a person by ID
 responses:
 '200':
 description: OK
 content:
 application/json:
 schema:
 $ref: '#/components/schemas/Person'
 '404':
 description: Person not found
put:
 summary: Update a person by ID
 requestBody:
 required: true
 content:
 application/json:
 schema:
 $ref: '#/components/schemas/Person'
 responses:
 '200':
 description: OK
 content:
 application/json:
 schema:
 $ref: '#/components/schemas/Person'
 '404':
 description: Person not found
delete:
 summary: Delete a person by ID
 responses:
 '204':
 description: No Content
 '404':
 description: Person not found
components:
```

```
schemas:
 Person:
 type: object
 properties:
 id:
 type: integer
 readOnly: true
 firstName:
 type: string
 lastName:
 type: string
```

## Configurer l'objet de connexion mssql

Le package `mssql` implémente la connexion à Azure SQL Database en fournissant un paramètre de configuration pour un type d'authentification.

Sans mot de passe (recommandé)

1. Dans Visual Studio Code, créez un fichier `config.js` et ajoutez-y le code de configuration `mssql` suivant pour l'authentification auprès d'Azure SQL Database.

JavaScript

```
import * as dotenv from 'dotenv';
dotenv.config({ path: `.${process.env.NODE_ENV}`, debug: true });

const server = process.env.AZURE_SQL_SERVER;
const database = process.env.AZURE_SQL_DATABASE;
const port = parseInt(process.env.AZURE_SQL_PORT);
const type = process.env.AZURE_SQL_AUTHENTICATIONTYPE;

export const config = {
 server,
 port,
 database,
 authentication: {
 type
 },
 options: {
 encrypt: true
 }
};
```

- Créez un fichier `.env.development` pour vos variables d'environnement locales, ajoutez-y le texte suivant, et remplacez `<YOURSERVERNAME>` et `<YOURDATABASENAME>` par vos propres valeurs.

```
text
```

```
AZURE_SQL_SERVER=<YOURSERVERNAME>.database.windows.net
AZURE_SQL_DATABASE=<YOURDATABASENAME>
AZURE_SQL_PORT=1433
AZURE_SQL_AUTHENTICATIONTYPE=azure-active-directory-default
```

### ⚠ Notes

Les objets de configuration sans mot de passe peuvent être commités en toute sécurité dans le contrôle de code source, car ils ne contiennent pas de secrets, comme des noms d'utilisateur, des mots de passe ou des clés d'accès.

- Créez un dossier `.vscode` et créez un fichier `settings.json` dans le dossier.
- Ajoutez le code suivant pour ignorer les variables d'environnement et les dépendances pendant le déploiement zip.

```
JSON
```

```
{
 "appService.zipIgnorePattern": ["./.env*", "node_modules{,/**}"]
}
```

## Ajouter le code pour se connecter à Azure SQL Database

- Créez un fichier `database.js` et ajoutez-y le code suivant :

```
JavaScript
```

```
import sql from 'mssql';

export default class Database {
 config = {};
 poolconnection = null;
 connected = false;

 constructor(config) {
```

```
 this.config = config;
 console.log(`Database: config: ${JSON.stringify(config)}`);
}

async connect() {
 try {
 console.log(`Database connecting...${this.connected}`);
 if (this.connected === false) {
 this.poolconnection = await sql.connect(this.config);
 this.connected = true;
 console.log('Database connection successful');
 } else {
 console.log('Database already connected');
 }
 } catch (error) {
 console.error(`Error connecting to database:
${JSON.stringify(error)}`);
 }
}

async disconnect() {
 try {
 this.poolconnection.close();
 console.log('Database connection closed');
 } catch (error) {
 console.error(`Error closing database connection: ${error}`);
 }
}

async executeQuery(query) {
 await this.connect();
 const request = this.poolconnection.request();
 const result = await request.query(query);

 return result.rowsAffected[0];
}

async create(data) {
 await this.connect();
 const request = this.poolconnection.request();

 request.input('firstName', sql.NVarChar(255), data.firstName);
 request.input('lastName', sql.NVarChar(255), data.lastName);

 const result = await request.query(
 `INSERT INTO Person (firstName, lastName) VALUES (@firstName,
@lastName)`
);

 return result.rowsAffected[0];
}

async readAll() {
 await this.connect();
 const request = this.poolconnection.request();
```

```

 const result = await request.query(`SELECT * FROM Person`);

 return result.recordsets[0];
}

async read(id) {
 await this.connect();

 const request = this.poolconnection.request();
 const result = await request
 .input('id', sql.Int, +id)
 .query(`SELECT * FROM Person WHERE id = @id`);

 return result.recordset[0];
}

async update(id, data) {
 await this.connect();

 const request = this.poolconnection.request();

 request.input('id', sql.Int, +id);
 request.input('firstName', sql.NVarChar(255), data.firstName);
 request.input('lastName', sql.NVarChar(255), data.lastName);

 const result = await request.query(
 `UPDATE Person SET firstName=@firstName, lastName=@lastName WHERE
id = @id`
);

 return result.rowsAffected[0];
}

async delete(id) {
 await this.connect();

 const idAsNumber = Number(id);

 const request = this.poolconnection.request();
 const result = await request
 .input('id', sql.Int, idAsNumber)
 .query(`DELETE FROM Person WHERE id = @id`);

 return result.rowsAffected[0];
}
}

```

## Tester l'application en local

L'application est prête à être testée localement. Vérifiez que vous êtes connecté au cloud Azure dans Visual Studio Code avec le même compte que l'administrateur de

votre base de données.

1. Exécutez l'application avec la commande suivante. Cette application démarre sur le port 3000.

```
Bash
```

```
NODE_ENV=development node index.js
```

La table **Person** est créée dans la base de données lorsque vous exécutez cette application.

2. Dans un navigateur, accédez à l'explorateur OpenAPI à l'adresse <http://localhost:3000>.
3. Dans la page de l'interface utilisateur Swagger, développez la méthode POST, puis sélectionnez **Essayer**.
4. Modifiez l'exemple JSON pour inclure les valeurs des propriétés. La propriété ID est ignorée.

The screenshot shows the Swagger UI interface for the Persons API. At the top, it says "Persons API 1.0.0 OAS3". Below that, under the "default" section, there are two methods listed: "GET /persons Get all persons" and "POST /persons Create a new person". The "POST" method is currently selected. Under the "POST" method, there is a "Parameters" section which says "No parameters". Below that is a "Request body" section with the status "required". A dropdown menu next to "Request body" shows "application/json". The "Request body" field contains the following JSON code:

```
{
 "firstName": "Bill",
 "lastName": "Jones"
}
```

5. Sélectionnez **Exécuter** pour ajouter un nouvel enregistrement à la base de données. L'API retourne une réponse de succès.
6. Développez la méthode **GET** sur la page de l'interface utilisateur Swagger, puis sélectionnez **Essayer**. Sélectionnez **Exécuter**, et la personne que vous venez de

créer est retournée.

# Déployer dans Azure App Service

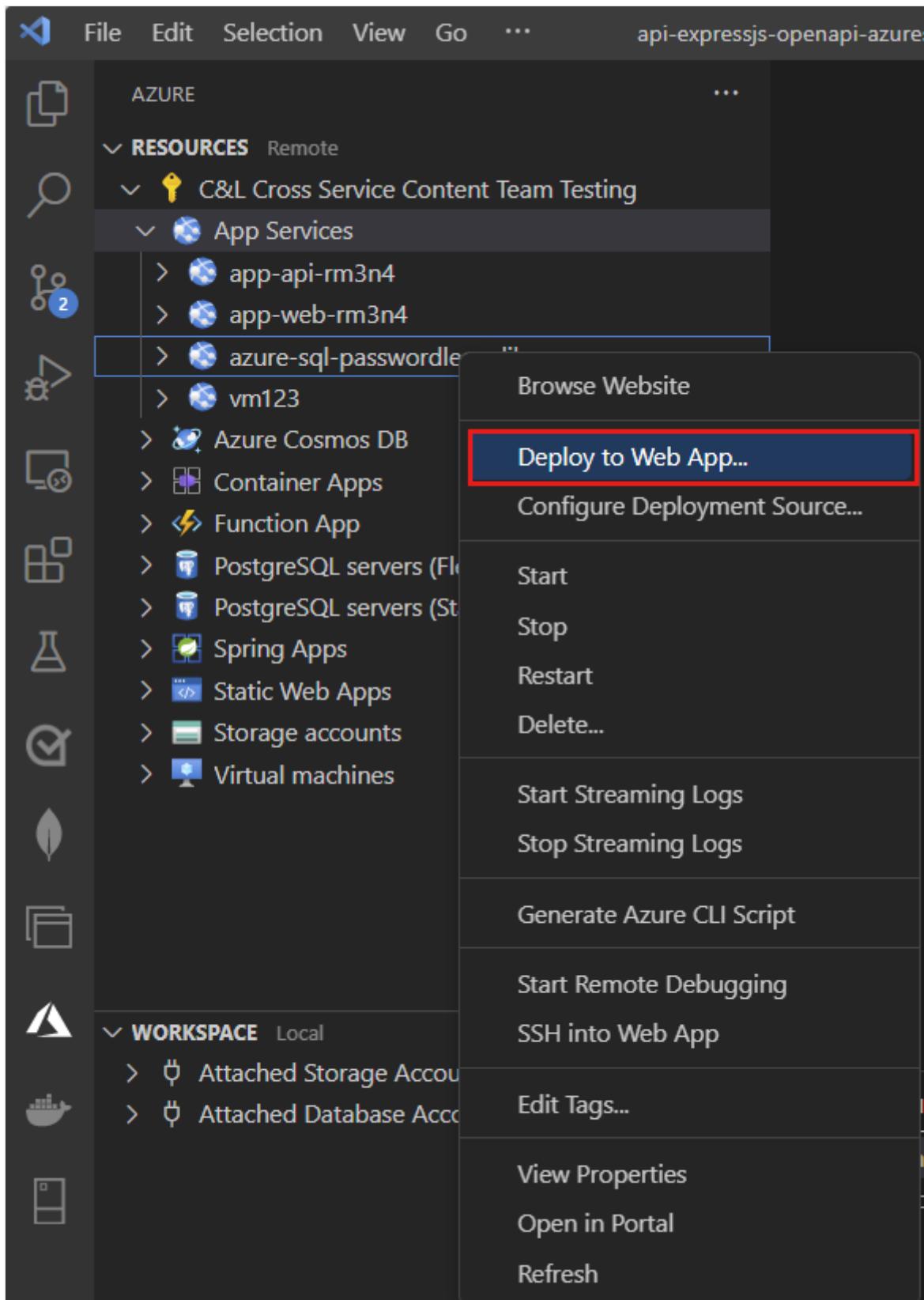
L'application est prête à être déployée sur Azure. Visual Studio Code peut créer une instance Azure App Service et déployer votre application au cours d'un même workflow.

1. Vérifiez que l'application est arrêtée.
2. Connectez-vous à Azure, si ce n'est pas déjà fait, en sélectionnant la commande **Azure : Se connecter au cloud Azure** dans la palette de commandes (**Ctrl** + **Maj** + **P**)
3. Dans la fenêtre **Azure Explorer** de Visual Studio Code, cliquez avec le bouton droit sur le nœud **App Services** et sélectionnez **Créer une application web (avancé)**.
4. Utilisez le tableau suivant pour créer App Service :

 Agrandir le tableau

Prompt	Valeur
Entrez un nom global unique pour la nouvelle application web.	Entrez une invite similaire à <code>azure-sql-passwordless</code> . Ajoutez après une chaîne unique telle que <code>123</code> .
Sélectionnez un groupe de ressources pour les nouvelles ressources.	Sélectionnez <b>+Créer un groupe de ressources</b> , puis sélectionnez le nom par défaut.
Sélectionnez une pile de runtime.	Sélectionnez une version LTS de la pile Node.js.
Sélectionnez un système d'exploitation.	Sélectionnez <b>Linux</b> .
Sélectionnez un emplacement pour les nouvelles ressources.	Sélectionnez un emplacement proche de vous.
Sélectionnez un plan App Service Linux.	Sélectionnez <b>Créer un plan App Service</b> , puis sélectionnez le nom par défaut.
Sélectionnez un niveau tarifaire.	Sélectionnez <b>Gratuit (F1)</b> .
Sélectionnez une ressource Application Insights pour votre application.	Sélectionnez <b>Ignorer pour le moment</b> .

5. Avant de continuer, attendez l'affichage de la notification indiquant que votre application a été créée.
6. Dans Azure Explorer, développez le nœud **App Services** et cliquez avec le bouton droit sur votre nouvelle application.
7. Sélectionnez **Déployer sur l'application web**.



8. Sélectionnez le dossier racine du projet JavaScript.

9. Lorsque la fenêtre contextuelle Visual Studio Code s'affiche, sélectionnez **Déployer**.

À la fin du déploiement, l'application ne fonctionne pas correctement sur Azure. Vous devez toujours configurer la connexion sécurisée entre App Service et la base de données SQL pour récupérer vos données.

## Connecter App Service à Azure SQL Database

Sans mot de passe (recommandé)

Les étapes suivantes sont requises pour connecter l'instance App Service à Azure SQL Database :

1. Créez une identité managée pour App Service.
2. Créez un utilisateur de base de données SQL et associez-le à l'identité managée App Service.
3. Attribuez des rôles SQL à l'utilisateur de base de données qui octroient des autorisations de lecture, d'écriture et éventuellement d'autres autorisations.

Plusieurs outils sont disponibles pour implémenter ces étapes :

Service Connector (recommandé)

Service Connector est un outil qui simplifie les connexions authentifiées entre différents services dans Azure. Service Connector prend actuellement en charge la connexion d'App Service à une base de données Azure SQL en utilisant la commande `az webapp connection create sql` dans Azure CLI. Cette commande unique effectue les trois étapes mentionnées ci-dessus pour vous.

### Créer l'identité managée avec Service Connector

Exécutez la commande suivante dans le Cloud Shell du portail Azure. Le Cloud Shell utilise la dernière version d'Azure CLI. Remplacez les variables dans `<>` par vos propres valeurs.

Azure CLI

```
az webapp connection create sql \
-g <app-service-resource-group> \
-n <app-service-name> \
--tg <database-server-resource-group> \
--server <database-server-name> \
```

```
--database <database-name> \
--system-identity
```

## Vérifier les paramètres de l'application App Service

Vous pouvez vérifier les modifications apportées par Service Connector sur les paramètres App Service.

1. Dans Visual Studio Code, dans Azure Explorer, cliquez avec le bouton droit sur votre application App Service et sélectionnez **Ouvrir dans le portail**.
2. Accédez à la page **Identité** de votre App Service. Sous l'onglet **Affecté par le système**, l'**État** doit être défini sur **Activé**. Cette valeur signifie qu'une identité managée affectée par le système a été activée pour votre application.
3. Accédez à la page **Configuration** de votre App Service. Sous l'onglet **Paramètres de l'application**, vous devez voir plusieurs variables d'environnement, lesquelles se trouvaient déjà dans l'objet de configuration **mssql**.
  - AZURE\_SQL\_SERVER
  - AZURE\_SQL\_DATABASE
  - AZURE\_SQL\_PORT
  - AZURE\_SQL\_AUTHENTICATIONTYPE

Ne supprimez pas et ne modifiez pas les noms de propriétés ou leurs valeurs.

## Tester l'application déployée

Accédez à l'URL de l'application pour tester que la connexion à Azure SQL Database fonctionne. Vous pouvez localiser l'URL de votre application dans la page de vue d'ensemble d'App Service.

La personne que vous avez créée localement devrait s'afficher dans le navigateur. Félicitations ! Votre application est maintenant connectée à Azure SQL Database dans les environnements locaux et hébergés.

## 💡 Conseil

Si vous obtenez une erreur Serveur interne 500 pendant le test, ce sont peut-être vos configurations réseau de base de données qui sont en cause. Vérifiez que votre serveur logique est configuré avec les paramètres décrits dans la section [Configurer la base de données](#).

## Nettoyer les ressources

Une fois que vous avez terminé d'utiliser Azure SQL Database, supprimez la ressource pour éviter les coûts imprévus.

Azure portal

1. Dans la barre de recherche du portail Azure, recherchez *Azure SQL* et sélectionnez le résultat correspondant.
2. Recherchez et sélectionnez votre base de données dans la liste.
3. Dans la page **Vue d'ensemble** d'Azure SQL Database, sélectionnez **Supprimer**.
4. Dans la page **Voulez-vous vraiment supprimer...** qui s'ouvre, tapez le nom de la base de données pour confirmer, puis sélectionnez **Supprimer**.

## Exemple de code

L'exemple de code pour cette application est disponible sur [GitHub](#).

## Étapes suivantes

- [Tutoriel : Sécuriser une base de données dans Azure SQL Database](#)
- [Autoriser l'accès base de données à SQL Database](#)
- [Une vue d'ensemble des fonctionnalités de sécurité d'Azure SQL Database](#)
- [Meilleures pratiques en matière de sécurité pour Azure SQL Database](#)

# Démarrage rapide : Bibliothèque Azure Cosmos DB for NoSQL pour .Node.js

Article • 11/01/2024

S'APPLIQUE À : NoSQL

Démarrage avec la bibliothèque de client Azure Cosmos DB for NoSQL pour .Node.js afin d'interroger des données dans vos conteneurs et d'effectuer des opérations courantes sur des éléments individuels. Suivez ces étapes pour déployer une solution minimale dans votre environnement à l'aide d'Azure Developer CLI.

[Documentation de référence sur l'API](#) | [Code source de la bibliothèque](#) ↗ | [Package \(npm\)](#) ↗ | [Azure Developer CLI](#)

## Prérequis

- Compte Azure avec un abonnement actif. [Créez un compte gratuitement](#) ↗.
- [Compte GitHub](#) ↗

## Configuration

Déployez le conteneur de développement de ce projet dans votre environnement. Utilisez ensuite Azure Developer CLI (`azd`) pour créer un compte Azure Cosmos DB for NoSQL et déployer un exemple d'application conteneurisé. L'exemple d'application utilise la bibliothèque de client pour gérer, créer, lire et interroger des exemples de données.



### Important

Les comptes GitHub incluent un droit de stockage et des heures cœur sans coût supplémentaire. Pour plus d'informations, consultez [Stockage et heures cœur inclus dans les comptes GitHub](#) ↗.

1. Ouvrez un terminal dans le répertoire racine du projet.
2. Authentifiez-vous auprès d'Azure Developer CLI à l'aide de `azd auth login`. Suivez les étapes spécifiées par l'outil pour vous authentifier auprès de l'interface CLI à

l'aide de vos informations d'identification Azure préférées.

```
Azure CLI
```

```
azd auth login
```

3. Utilisez `azd init` pour initialiser le projet.

```
Azure CLI
```

```
azd init
```

4. Lors de l'initialisation, configurez un nom d'environnement unique.

#### 💡 Conseil

Le nom de l'environnement sera également utilisé comme nom du groupe de ressources cible. Pour ce démarrage rapide, envisagez d'utiliser `msdocs-cosmos-db-nosql`.

5. Déployez le compte Azure Cosmos DB for NoSQL à l'aide de `azd up`. Les modèles Bicep déploient également un exemple d'application web.

```
Azure CLI
```

```
azd up
```

6. Pendant le processus d'approvisionnement, sélectionnez votre abonnement et l'emplacement souhaité. Attendez la fin du processus de provisionnement. Le processus peut prendre **environ cinq minutes**.

7. Une fois l'approvisionnement de vos ressources Azure terminée, une URL vers l'application web en cours d'exécution est incluse dans la sortie.

```
Output
```

```
Deploying services (azd deploy)
```

```
(✓) Done: Deploying service web
- Endpoint: <https://[container-app-sub-domain].azurecontainerapps.io>
```

```
SUCCESS: Your application was provisioned and deployed to Azure in 5
minutes 0 seconds.
```

8. Utilisez l'URL dans la console pour accéder à votre application web dans le navigateur. Observez la sortie de l'application en cours d'exécution.



Azure Cosmos DB for NoSQL | Accéder au démarrage rapide

```
État actuel : Démarrage ..
Obtenir une base de données
Obtenir un conteneur : produits
Élément upsert : {70b63682-b93a-4c77-aad2-65501347265f gear-surf-surfboards Yamba Surfboard 12 850 false}
Code d'état : 201
Frais de requête : 7,24
Élément upsert : {25a69543-b90c-439d-8332-7ef41e06a0e0 gear-surf-surfboards Kiama Classic Surfboard 25 790 true}
Code d'état : 201
Frais de requête : 7,24
Identification de l'élément 70b63682-b93a-4c77-aad2-65501347265f
Élément de lecture : {70b63682-b93a-4c77-aad2-65501347265f gear-surf-surfboards Yamba Surfboard 12 850 false}
Code d'état : 200
Frais de requête : 1,00
```

## Installer la bibliothèque de client

La bibliothèque de client est disponible via le gestionnaire de package Node, en tant que package `@azure/cosmos`.

1. Depuis un terminal, accédez au dossier `/src`.

```
Bash

cd ./src
```

2. S'il n'est pas déjà installé, installez le package `@azure/cosmos` à l'aide de `npm install`.

```
Bash

npm install --save @azure/cosmos
```

3. Installez également le package `@azure/identity` s'il n'est pas déjà installé.

```
Bash

npm install --save @azure/identity
```

4. Ouvrez et évaluez le fichier `src/package.json` pour vérifier que les entrées `azure-cosmos` et `azure-identity` existent.

## Modèle objet

Nom	Description
CosmosClient	Cette classe est la classe cliente principale utilisée pour gérer les métadonnées ou bases de données à l'échelle du compte.
Database	Cette classe représente une base de données dans le compte.
Container	Cette classe est principalement utilisée pour effectuer des opérations de lecture, de mise à jour et de suppression sur le conteneur ou les éléments stockés dans le conteneur.
PartitionKey	Cette classe représente une clé de partition logique. Cette classe est requise dans de nombreuses opérations et requêtes courantes.
SqlQuerySpec	Cette interface représente une requête SQL et tous les paramètres de requête.

## Exemples de code

- [Authentifier le client](#)
- [Obtenir une base de données](#)
- [Obtenir un conteneur](#)
- [Créer un élément](#)
- [Obtenir un élément](#)
- [Éléments de requête](#)

L'exemple de code du modèle utilise une base de données nommée `cosmicworks` et un conteneur nommé `products`. Le conteneur `products` contient des détails tels que le nom, la catégorie, la quantité, un identificateur unique et un indicateur de vente pour chaque produit. Le conteneur utilise la propriété `/category` comme clé de partition logique.

## Authentifier le client

Les requêtes d'application vers les Services Azure doivent être autorisées. Utilisez le type `DefaultAzureCredential` comme moyen préféré d'implémenter une connexion sans mot de passe entre vos applications et Azure Cosmos DB for NoSQL.  
`DefaultAzureCredential` prend en charge plusieurs méthodes d'authentification et détermine quelle méthode doit être utilisée au moment de l'exécution.

 **Important**

Vous pouvez également autoriser directement les requêtes adressées aux services Azure à l'aide de mots de passe, de chaînes de connexion ou d'autres informations d'identification. Toutefois, cette approche doit être utilisée avec prudence. Les développeurs doivent être vigilants pour ne jamais exposer les secrets dans un emplacement non sécurisé. Toute personne ayant accès au mot de passe ou à la clé secrète peut s'authentifier auprès du service de base de données. `DefaultAzureCredential` offre des avantages de gestion et de sécurité améliorés sur la clé de compte pour autoriser l'authentification sans mot de passe sans risque de stocker des clés.

Cet exemple crée une instance de type `CosmosClient` et s'authentifie à l'aide d'une instance `DefaultAzureCredential`.

JavaScript

```
const credential = new DefaultAzureCredential();

const client = new CosmosClient({
 endpoint,
 aadCredentials: credential
});
```

## Obtenir une base de données

Utilisez `client.database` pour récupérer la base de données existante nommée `cosmicworks`.

JavaScript

```
const database = client.database('cosmicworks');
```

## Obtenir un conteneur

Récupérez le conteneur `products` existant à l'aide de `database.container`.

JavaScript

```
const container = database.container('products');
```

## Créer un élément

Générez un objet avec tous les membres que vous souhaitez sérialiser dans JSON. Dans cet exemple, le type a un identificateur unique et des champs pour la catégorie, le nom, la quantité, le prix et la vente. Créez un élément dans le conteneur à l'aide de `container.items.upsert`. Cette méthode fait un « upsert » de l'élément efficacement en le remplaçant s'il existe déjà.

JavaScript

```
var item = {
 'id': '70b63682-b93a-4c77-aad2-65501347265f',
 'category': 'gear-surf-surfboards',
 'name': 'Yamba Surfboard',
 'quantity': 12,
 'price': 850.00,
 'clearance': false
};

var response = await container.items.upsert(item);
```

## Lire un élément

Effectuez une opération de lecture de point à l'aide des champs d'identificateur unique (`id`) et de clé de partition. Utilisez `container.item` pour obtenir un pointeur vers un élément et `item.read` pour récupérer efficacement l'élément spécifique.

JavaScript

```
var id = '70b63682-b93a-4c77-aad2-65501347265f';
var partitionKey = 'gear-surf-surfboards';

var response = await container.item(id, partitionKey).read();
var read_item = response.resource;
```

## Éléments de requête

Effectuez une requête sur plusieurs éléments d'un conteneur à l'aide de `container.items.query`. Recherchez tous les éléments d'une catégorie spécifiée à l'aide de cette requête paramétrable :

nosql

```
SELECT * FROM products p WHERE p.category = @category
```

Récupérez (fetch) tous les résultats de la requête à l'aide de `query.fetchAll`. Effectuez une boucle dans les résultats de la requête.

JavaScript

```
const querySpec = {
 query: 'SELECT * FROM products p WHERE p.category = @category',
 parameters: [
 {
 name: '@category',
 value: 'gear-surf-surfboards'
 }
]
};

var response = await container.items.query(querySpec).fetchAll();
for (var item of response.resources) {

}
```

## Contenu connexe

- Démarrage rapide .NET
- Démarrage rapide Java
- Démarrage rapide Python
- Démarrage rapide Go

## Étape suivante

Tutoriel : Générer une application web Node.js

# Envoyer des événements à Event Hubs ou en recevoir à l'aide de JavaScript

Article • 19/06/2023

Ce guide de démarrage rapide montre la procédure pour recevoir des événements d'un Event Hub et lui en envoyer à l'aide du package npm [@azure/event-hubs](#).

## Prérequis

Si vous débutez avec Azure Event Hubs, consultez la [vue d'ensemble d'Event Hubs](#) avant de suivre ce guide de démarrage rapide.

Pour effectuer ce démarrage rapide, vous avez besoin de ce qui suit :

- **Abonnement Microsoft Azure.** Pour utiliser les services Azure, y compris Azure Event Hubs, vous avez besoin d'un abonnement. Si vous n'avez pas de compte Azure, vous pouvez vous inscrire à un [essai gratuit](#) ou utiliser les avantages de votre abonnement MSDN quand vous [créez un compte](#).
- Node.js LTS. Téléchargez la [version LTS \(prise en charge à long terme\)](#) la plus récente.
- Visual Studio Code (recommandé) ou tout autre environnement de développement intégré (IDE).
- **Créez un espace de noms Event Hubs et un Event Hub.** La première étape consiste à utiliser le [portail Azure](#) pour créer un espace de noms de type Event Hubs et obtenir les informations de gestion nécessaires à votre application pour communiquer avec le concentrateur d'événements. Pour créer un espace de noms et un hub d'événements, suivez la procédure décrite dans [cet article](#).

## Installer le ou les packages npm pour envoyer des événements

Pour installer le [package npm \(Node Package Manager\) pour Event Hubs](#), ouvrez une invite de commandes dont le chemin contient *npm*. Remplacez le répertoire par le dossier où vous souhaitez conserver vos exemples.

Sans mot de passe (recommandé)

Exécutez ces commandes :

```
shell
```

```
npm install @azure/event-hubs
npm install @azure/identity
```

## Authentifier l'application sur Azure

Ce guide de démarrage rapide présente deux façons de vous connecter à Azure Event Hubs : sans mot de passe et avec une chaîne de connexion. La première option vous explique comment utiliser votre principal de sécurité dans Microsoft Entra ID et le contrôle d'accès en fonction du rôle (RBAC) pour vous connecter à un espace de noms Event Hubs. Vous n'avez pas à vous soucier d'avoir des chaînes de connexion codées en dur dans votre code, dans un fichier config ni dans un stockage sécurisé comme Azure Key Vault. La deuxième option consiste à se servir d'une chaîne de connexion pour se connecter à un espace de noms Event Hubs. Si vous débutez avec Azure, vous trouverez peut-être l'option de chaîne de connexion plus facile à suivre. Nous vous recommandons d'utiliser l'option sans mot de passe dans les applications réelles et les environnements de production. Pour plus d'informations, consultez [Authentification et autorisation](#). Pour en savoir plus sur l'authentification sans mot de passe, reportez-vous à la [page de présentation](#).

Sans mot de passe (recommandé)

## Attribuer des rôles à votre utilisateur Microsoft Entra

Lors du développement localement, vous devez vérifier que le compte d'utilisateur qui se connecte à Azure Event Hubs dispose des autorisations appropriées. Vous aurez besoin du rôle [Propriétaire de données Azure Event Hubs](#) pour envoyer et recevoir des messages. Pour vous attribuer ce rôle, vous aurez besoin du rôle Administrateur de l'accès utilisateur ou d'un autre rôle qui inclut l'action `Microsoft.Authorization/roleAssignments/write`. Vous pouvez attribuer des rôles RBAC Azure à un utilisateur à l'aide du Portail Azure, Azure CLI ou Azure PowerShell. Découvrez les étendues disponibles pour les attributions de rôles dans la page [vue d'ensemble de l'étendue](#).

L'exemple suivant attribue le rôle `Azure Event Hubs Data Owner` à votre compte d'utilisateur, qui fournit un accès complet aux ressources Azure Event Hubs. Dans un scénario réel, suivez le [principe des priviléges minimum](#) pour accorder aux

utilisateurs uniquement les autorisations minimales nécessaires à un environnement de production plus sécurisé.

## Rôles intégrés Azure pour Azure Event Hubs

Pour Azure Event Hubs, la gestion des espaces de noms et de toutes les ressources associées via le portail Azure et l'API de gestion des ressources Azure est déjà protégée à l'aide du modèle RBAC Azure. Azure fournit les rôles Azure intégrés ci-dessous pour autoriser l'accès à un espace de noms Event Hubs :

- [Propriétaire de données Azure Event Hubs](#) : permet l'accès aux données d'un espace de noms Event Hubs et de ses entités (files d'attente, rubriques, abonnements et filtres).
- [Expéditeur de données Azure Event Hubs](#) : utilisez ce rôle pour autoriser l'expéditeur à accéder à l'espace de noms Event Hubs et à ses entités.
- [Récepteur de données Azure Event Hubs](#) : utilisez ce rôle pour autoriser l'expéditeur à accéder à l'espace de noms Event Hubs et à ses entités.

Si vous souhaitez créer un rôle personnalisé, consultez [Droits requis pour les opérations Service Bus](#).

### ⓘ Important

Dans la plupart des cas, la propagation de l'attribution de rôle dans Azure peut prendre une ou deux minutes. Dans de rares cas, cela peut prendre jusqu'à huit minutes. Si vous recevez des erreurs d'authentification lorsque vous exécutez votre code pour la première fois, patientez quelques instants et réessayez.

Azure portal

1. Dans le portail Azure, recherchez votre espace de noms Event Hubs à l'aide de la barre de recherche principale ou de la navigation gauche.
2. Dans la page vue d'ensemble, sélectionnez **Contrôle d'accès (IAM)** dans le menu de gauche.
3. Sur la page **Contrôle d'accès (IAM)**, sélectionnez l'onglet **Attributions de rôles**.

4. Sélectionnez + Ajouter dans le menu supérieur, puis Ajouter une attribution de rôle dans le menu déroulant résultant.

The screenshot shows the 'Access control (IAM)' blade for a Service Bus Namespace named 'spsbusns11102'. The left sidebar lists various management options like Overview, Activity log, Tags, and Diagnose and solve problems. The 'Access control (IAM)' option is selected and highlighted with a red box. The main area has a header with 'Add', 'Download role assignments', 'Edit columns', 'Refresh', 'Remove', and 'Got feedback?' buttons. Below the header, there's a sub-menu with 'Add role assignment' (highlighted with a red box and yellow arrow '2'), 'Add co-administrator' (highlighted with yellow arrow '3'), 'Check access', 'Role assignments', 'Roles', 'Deny assignments', and 'Classic administrators'. A tooltip for 'Add role assignment' is displayed, stating: 'o read this customer directory, so some options are disabled. If you require access to these options, Add co-administrator'. The 'Check access' section contains a 'View my access' button. The 'Grant access to this resource' section has a 'Add role assignment' button and a 'Learn more' link. The 'View access to this resource' section has a 'View' button and a 'Learn more' link.

5. Utilisez la zone de recherche pour filtrer les résultats sur le rôle souhaité.

Pour cet exemple, recherchez **Azure Event Hubs Data Owner** et sélectionnez le résultat correspondant. Ensuite, choisissez **Suivant**.

6. Sous **Attribuer l'accès à**, sélectionnez **Utilisateur, groupe ou principal de service**, puis sélectionnez + **Sélectionner des membres**.

7. Dans la boîte de dialogue, recherchez votre nom d'utilisateur Microsoft Entra (généralement votre adresse e-mail *user@domain*), puis choisissez **Sélectionner** en bas de la boîte de dialogue.

8. Sélectionnez **Vérifier + affecter** pour accéder à la page finale, puis **Vérifier + attribuer** à nouveau pour terminer le processus.

## Envoyer des événements

Dans cette section, vous créez une application JavaScript qui envoie des événements à un hub d'événements.

1. Ouvrez votre éditeur favori, tel que [Visual Studio Code](#).

2. Créez un fichier appelé *send.js* et collez-y le code suivant :

Sans mot de passe (recommandé)

Dans le code, utilisez des valeurs réelles pour remplacer les espaces réservés suivants :

- EVENT HUBS RESOURCE NAME
- EVENT HUB NAME

JavaScript

```
const { EventHubProducerClient } = require("@azure/event-hubs");
const { DefaultAzureCredential } = require("@azure/identity");

// Event hubs
const eventHubsResourceName = "EVENT HUBS RESOURCE NAME";
const fullyQualifiedNamespace =
`${eventHubsResourceName}.servicebus.windows.net`;
const eventHubName = "EVENT HUB NAME";

// Azure Identity - passwordless authentication
const credential = new DefaultAzureCredential();

async function main() {

 // Create a producer client to send messages to the event hub.
 const producer = new
EventHubProducerClient(fullyQualifiedNamespace, eventHubName,
credential);

 // Prepare a batch of three events.
 const batch = await producer.createBatch();
 batch.tryAdd({ body: "passwordless First event" });
 batch.tryAdd({ body: "passwordless Second event" });
 batch.tryAdd({ body: "passwordless Third event" });

 // Send the batch to the event hub.
 await producer.sendBatch(batch);

 // Close the producer client.
 await producer.close();

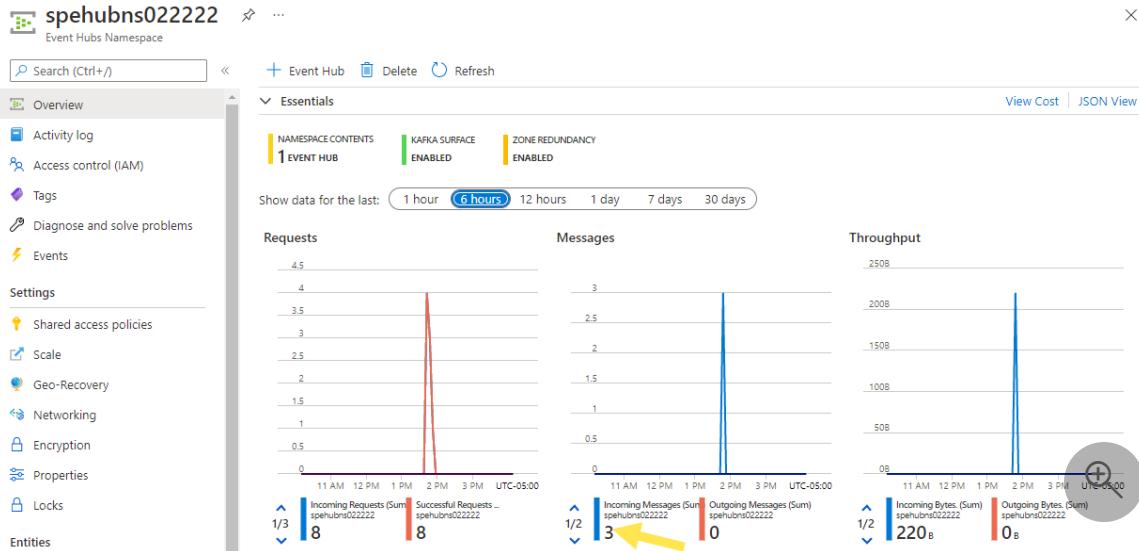
 console.log("A batch of three events have been sent to the event
hub");
}

main().catch((err) => {
 console.log("Error occurred: ", err);
});
```

3. Exécutez `node send.js` pour exécuter ce fichier. Cette commande envoie un lot de trois événements à votre hub d'événements.

4. Dans le portail Azure, vérifiez que le hub d'événements a reçu les messages.

Actualisez la page pour mettre à jour le graphique. Cela peut prendre quelques secondes pour indiquer que les messages ont été reçus.



### ⓘ Notes

Pour obtenir le code source complet, y compris des commentaires d'informations supplémentaires, accédez à la page [GitHub sendEvents.js ↗](#).

Félicitations ! Vous venez d'envoyer des événements à un hub d'événements.

## Recevoir des événements

Dans cette section, vous recevez des événements d'un hub d'événements à l'aide d'un magasin de points de contrôle de stockage Blob Azure dans une application JavaScript. Il effectue des points de contrôle de métadonnées sur les messages reçus à intervalles réguliers dans un blob Stockage Azure. Cette approche permet ultérieurement de continuer à recevoir des messages à partir du point où vous vous étiez arrêté.

Suivez les recommandations ci-dessous quand vous utilisez Stockage Blob Azure comme magasin de points de contrôle :

- Utilisez un conteneur distinct pour chaque groupe de processeurs. Vous pouvez utiliser le même compte de stockage, mais utiliser un conteneur par groupe.
- N'utilisez pas le conteneur et le compte de stockage pour quoi que ce soit d'autre.

- Le compte de stockage doit se trouver dans la même région que l'application déployée. Si l'application est locale, essayez de choisir la région la plus proche possible.

Sur la page **Compte de stockage** du Portail Azure, dans la section **Service BLOB**, vérifiez que les paramètres suivants sont désactivés.

- Espace de noms hiérarchique
- Suppression réversible de blob
- Contrôle de version

## Créer un compte de stockage Azure et un conteneur d'objets blob

Pour créer un compte de stockage Azure et un conteneur d'objets blob dans celui-ci, effectuez les actions suivantes :

1. [Créer un compte de stockage Azure](#)
2. [Créer un conteneur d'objets blob dans le compte de stockage](#)
3. S'authentifier auprès du conteneur d'objets blob

Sans mot de passe (recommandé)

Lors du développement local, assurez-vous que le compte d'utilisateur qui accède aux données blob dispose des autorisations appropriées. Vous aurez besoin du **Contributeur aux données Blob de stockage** pour lire et écrire des données blob. Pour vous attribuer ce rôle, vous aurez besoin du rôle **Administrateur de l'accès utilisateur** ou d'un autre rôle qui inclut l'action **Microsoft.Authorization/roleAssignments/write**. Vous pouvez attribuer des rôles RBAC Azure à un utilisateur à l'aide du Portail Azure, Azure CLI ou Azure PowerShell. Vous pouvez en savoir plus sur les étendues disponibles pour les attributions de rôles dans la page [vue d'ensemble de l'étendue](#).

Dans ce scénario, vous allez attribuer des autorisations à votre compte d'utilisateur, étendues au compte de stockage, pour suivre le **Principe des privilèges minimum**. Cette pratique offre aux utilisateurs uniquement les autorisations minimales nécessaires et crée des environnements de production plus sécurisés.

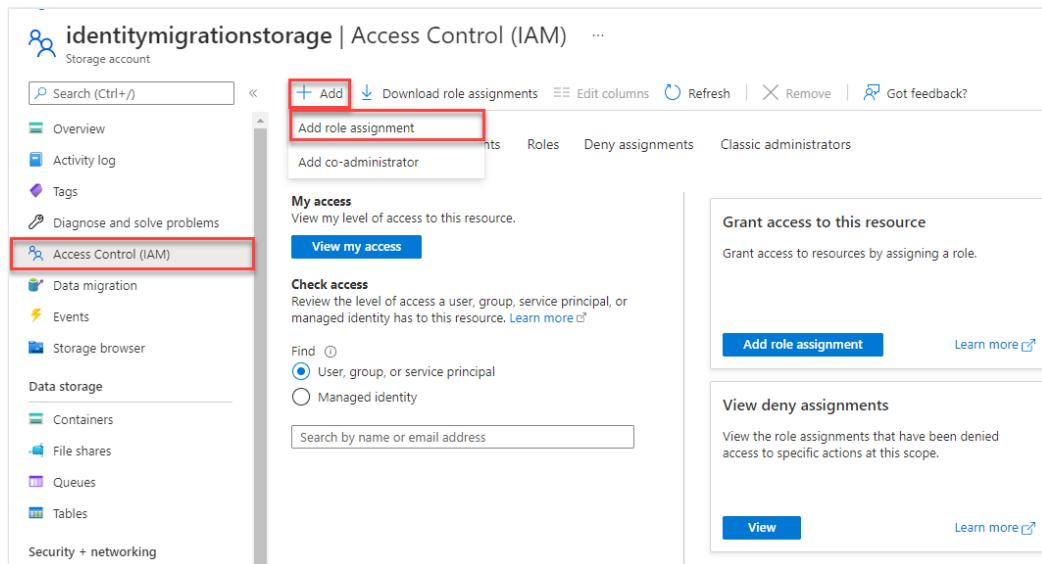
L'exemple suivant affecte le rôle **Contributeur aux données Blob du stockage** à votre compte d'utilisateur, qui fournit à la fois un accès en lecture et en écriture aux données d'objet blob dans votre compte de stockage.

## ⓘ Important

Dans la plupart des cas, la propagation de l'attribution de rôle dans Azure peut prendre une ou deux minutes, mais dans de rares cas, cela peut prendre jusqu'à huit minutes. Si vous recevez des erreurs d'authentification lorsque vous exécutez votre code pour la première fois, patientez quelques instants et réessayez.

Azure portal

1. Dans le Portail Azure, recherchez votre compte de stockage à l'aide de la barre de recherche principale ou de la navigation gauche.
2. Dans la page vue d'ensemble du compte de stockage, sélectionnez **Contrôle d'accès (IAM)** dans le menu de gauche.
3. Sur la page **Contrôle d'accès (IAM)**, sélectionnez l'onglet **Attributions de rôles**.
4. Sélectionnez **+ Ajouter** dans le menu supérieur, puis **Ajouter une attribution de rôle** dans le menu déroulant résultant.



5. Utilisez la zone de recherche pour filtrer les résultats sur le rôle souhaité. Pour cet exemple, recherchez *Contributeur aux données Blob du stockage*, sélectionnez le résultat correspondant, puis choisissez **Suivant**.
6. Sous **Attribuer l'accès à**, sélectionnez **Utilisateur, groupe ou principal de service**, puis sélectionnez **+ Sélectionner des membres**.

7. Dans la boîte de dialogue, recherchez votre nom d'utilisateur Microsoft Entra (généralement votre adresse e-mail *user@domain*), puis choisissez **Sélectionner** en bas de la boîte de dialogue.
8. Sélectionnez **Vérifier + affecter** pour accéder à la page finale, puis **Vérifier + attribuer** à nouveau pour terminer le processus.

## Installer les packages npm pour recevoir des événements

Pour le côté récepteur, vous avez besoin d'installer deux autres packages. Dans ce guide de démarrage rapide, vous utilisez le stockage Blob Azure pour conserver les points de contrôle afin que le programme ne lise pas les événements qu'il a déjà lus. Il effectue des points de contrôle de métadonnées sur les messages reçus à intervalles réguliers dans un blob. Cette approche permet ultérieurement de continuer à recevoir des messages à partir du point où vous vous étiez arrêté.

Sans mot de passe (recommandé)

Exécutez ces commandes :

shell

```
npm install @azure/storage-blob
npm install @azure/eventhubs-checkpointstore-blob
npm install @azure/identity
```

## Écrire du code pour recevoir des événements

1. Ouvrez votre éditeur favori, tel que [Visual Studio Code](#).

2. Créez un fichier appelé *receive.js* et collez-y le code suivant :

Sans mot de passe (recommandé)

Dans le code, utilisez des valeurs réelles pour remplacer les espaces réservés suivants :

- EVENT HUBS RESOURCE NAME
- EVENT HUB NAME

- STORAGE ACCOUNT NAME
- STORAGE CONTAINER NAME

JavaScript

```

const { DefaultAzureCredential } = require("@azure/identity");
const { EventHubConsumerClient, earliestEventPosition } =
require("@azure/event-hubs");
const { ContainerClient } = require("@azure/storage-blob");
const { BlobCheckpointStore } = require("@azure/eventhubs-
checkpointstore-blob");

// Event hubs
const eventHubsResourceName = "EVENT HUBS RESOURCE NAME";
const fullyQualifiedNamespace =
`${eventHubsResourceName}.servicebus.windows.net`;
const eventHubName = "EVENT HUB NAME";
const consumerGroup = "$Default"; // name of the default consumer
group

// Azure Storage
const storageAccountName = "STORAGE ACCOUNT NAME";
const storageContainerName = "STORAGE CONTAINER NAME";
const baseUrl =
`https://${storageAccountName}.blob.core.windows.net`;

// Azure Identity - passwordless authentication
const credential = new DefaultAzureCredential();

async function main() {

 // Create a blob container client and a blob checkpoint store
 // using the client.
 const containerClient = new ContainerClient(
 `${baseUrl}/${storageContainerName}`,
 credential
);
 const checkpointStore = new BlobCheckpointStore(containerClient);

 // Create a consumer client for the event hub by specifying the
 // checkpoint store.
 const consumerClient = new EventHubConsumerClient(consumerGroup,
 fullyQualifiedNamespace, eventHubName, credential,
 checkpointStore);

 // Subscribe to the events, and specify handlers for processing
 // the events and errors.
 const subscription = consumerClient.subscribe({
 processEvents: async (events, context) => {
 if (events.length === 0) {
 console.log(`No events received within wait time. Waiting
for next interval`);
 return;
 }
 }
 });
}

```

```

 }

 for (const event of events) {
 console.log(`Received event: '${event.body}' from
partition: '${context.partitionId}' and consumer group:
'${context.consumerGroup}'`);
 }
 // Update the checkpoint.
 await context.updateCheckpoint(events[events.length - 1]);
},

processError: async (err, context) => {
 console.log(`Error : ${err}`);
}
},
{ startPosition: earliestEventPosition
);

// After 30 seconds, stop processing.
await new Promise((resolve) => {
 setTimeout(async () => {
 await subscription.close();
 await consumerClient.close();
 resolve();
 }, 30000);
});
}

main().catch((err) => {
 console.log("Error occurred: ", err);
});

```

3. Ensuite, exéutez `node receive.js` dans une invite de commandes pour exécuter ce fichier. La fenêtre doit afficher les messages relatifs aux événements reçus.

```

C:\Self Study\Event Hubs\JavaScript>node receive.js
Received event: 'First event' from partition: '0' and consumer group:
'$Default'
Received event: 'Second event' from partition: '0' and consumer group:
'$Default'
Received event: 'Third event' from partition: '0' and consumer group:
'$Default'

```

### ⓘ Notes

Pour obtenir le code source complet, y compris des commentaires d'informations supplémentaires, accédez à la [page GitHub](#)

[receiveEventsUsingCheckpointStore.js ↗](#)

Félicitations ! Vous recevez maintenant des événements depuis votre hub d'événements. Le programme récepteur recevra les événements de toutes les partitions du groupe de consommateurs par défaut dans le hub d'événements.

## Étapes suivantes

Consultez ces exemples sur GitHub :

- [Exemples JavaScript ↗](#)
- [Exemples TypeScript ↗](#)

# Démarrage rapide : bibliothèque de client de certificats Azure Key Vault pour JavaScript

Article • 24/05/2023

Démarrez avec la bibliothèque de client de certificats Azure Key Vault pour JavaScript. [Azure Key Vault](#) est un service cloud qui fournit un magasin de certificats sécurisé. Vous pouvez stocker des clés, des mots de passe, des certificats et d'autres secrets en toute sécurité. Vous pouvez créer et gérer des coffres de clés Azure grâce au portail Azure. Dans ce guide de démarrage rapide, vous allez découvrir comment créer, récupérer et supprimer des certificats dans un coffre de clés Azure en utilisant la bibliothèque de client JavaScript.

Ressources de la bibliothèque de client Key Vault :

[Documentation de référence sur les API](#) | [Code source de la bibliothèque](#) ↗ | [Package \(npm\)](#) ↗

Pour plus d'informations sur Key Vault et les certificats, consultez :

- [Vue d'ensemble du coffre de clés](#)
- [Vue d'ensemble des certificats](#)

## Prérequis

- Un abonnement Azure - [En créer un gratuitement](#) ↗
- Actuellement [Node.js LTS](#) ↗ .
- [Azure CLI](#)
- Un coffre-fort de clé existant - vous pouvez en créer un à l'aide des éléments ci-après :
  - [Azure CLI](#)
  - [Azure portal](#)
  - [Azure PowerShell](#)

Ce guide de démarrage rapide suppose que vous exécutez l'interface [Azure CLI](#).

## Connexion à Azure

1. Exécutez la commande `login`.

```
Azure CLI
```

```
az login
```

Si l'interface CLI peut ouvrir votre navigateur par défaut, elle le fait et charge une page de connexion Azure par la même occasion.

Sinon, ouvrez une page de navigateur à l'adresse <https://aka.ms/devicelogin> et entrez le code d'autorisation affiché dans votre terminal.

2. Dans le navigateur, connectez-vous avec les informations d'identification de votre compte.

## Créer une application Node.js

Créez une application Node.js qui utilise votre coffre-fort de clé.

1. Dans un terminal, créez un dossier nommé `key-vault-node-app` et modifiez-le dans ce dossier :

```
terminal
```

```
mkdir key-vault-node-app && cd key-vault-node-app
```

2. Initialiser le projet Node.js :

```
terminal
```

```
npm init -y
```

## Installer des packages Key Vault

1. À l'aide du terminal, installez la bibliothèque de secrets Azure Key Vault, [@azure/keyvault-certificates](#) pour Node.js.

```
terminal
```

```
npm install @azure/keyvault-certificates
```

2. Installez la bibliothèque de client Azure Identity, [@azure/identity](#) pour vous authentifier dans Key Vault.

terminal

```
npm install @azure/identity
```

## Accorder l'accès à votre coffre de clés

Créez une stratégie d'accès au coffre de votre coffre de clés qui accorde des autorisations de clé à votre compte d'utilisateur.

Azure CLI

```
az keyvault set-policy --name <YourKeyVaultName> --upn user@domain.com --
certificate-permissions delete get list create purge update
```

## Définir des variables d'environnement

Cette application utilise le nom de coffre de clés en tant que variable d'environnement appelée `KEY_VAULT_NAME`.

Windows

Invite de commandes Windows

```
set KEY_VAULT_NAME=<your-key-vault-name>
```

## Authentifier et créer un client

Les requêtes d'application vers les Services Azure doivent être autorisées. L'utilisation de la méthode `DefaultAzureCredential` fournie par la [bibliothèque de client Azure Identity](#) est l'approche recommandée pour implémenter des connexions sans mot de passe aux services Azure dans votre code. `DefaultAzureCredential` prend en charge plusieurs méthodes d'authentification et détermine quelle méthode doit être utilisée au moment de l'exécution. Cette approche permet à votre application d'utiliser différentes méthodes d'authentification dans différents environnements (local ou production) sans implémenter de code spécifique à l'environnement.

Dans ce guide de démarrage rapide, `DefaultAzureCredential` s'authentifie auprès du coffre de clés à l'aide des informations d'identification de l'utilisateur de développement local connecté à Azure CLI. Quand l'application est déployée sur Azure, le même code

`DefaultAzureCredential` peut découvrir et utiliser automatiquement une identité managée affectée à un service d'application, une machine virtuelle ou d'autres services. Pour plus d'informations, consultez [Vue d'ensemble des identités managées](#).

Dans cet code, le nom de votre coffre de clés est utilisé pour créer l'URI du coffre de clés, au format `https://<your-key-vault-name>.vault.azure.net`. Pour plus d'informations sur l'authentification auprès du coffre de clés, consultez le [Guide du développeur](#).

## Exemple de code

Ce code utilise les [classes et méthodes de certificat Key Vault](#) suivantes :

- [Classe DefaultAzureCredential](#)
- [Classe CertificateClient](#)
  - [beginCreateCertificate](#)
  - [getCertificate](#)
  - [getCertificateVersion](#)
  - [updateCertificateProperties](#)
  - [updateCertificatePolicy](#)
  - [beginDeleteCertificate](#)
- [Interface PollerLike](#)
  - [getResult](#)

## Configurer le framework d'application

1. Créez un fichier texte et collez le code suivant dans le fichier `index.js`.

JavaScript

```
const { CertificateClient, DefaultCertificatePolicy } =
require("@azure/keyvault-certificates");
const { DefaultAzureCredential } = require("@azure/identity");

async function main() {
 // If you're using MSI, DefaultAzureCredential should "just work".
 // Otherwise, DefaultAzureCredential expects the following three
 environment variables:
 // - AZURE_TENANT_ID: The tenant ID in Azure Active Directory
 // - AZURE_CLIENT_ID: The application (client) ID registered in the
 AAD tenant
 // - AZURE_CLIENT_SECRET: The client secret for the registered
 application
 const credential = new DefaultAzureCredential();
```

```
const keyVaultName = process.env["KEY_VAULT_NAME"];
if(!keyVaultName) throw new Error("KEY_VAULT_NAME is empty");
const url = "https://" + keyVaultName + ".vault.azure.net";

const client = new CertificateClient(url, credential);

const uniqueString = new Date().getTime();
const certificateName = `cert${uniqueString}`;

// Creating a self-signed certificate
const createPoller = await client.beginCreateCertificate(
 certificateName,
 DefaultCertificatePolicy
);

const pendingCertificate = createPoller.getResult();
console.log("Certificate: ", pendingCertificate);

// To read a certificate with their policy:
let certificateWithPolicy = await
client.getCertificate(certificateName);
// Note: It will always read the latest version of the certificate.

console.log("Certificate with policy:", certificateWithPolicy);

// To read a certificate from a specific version:
const certificateFromVersion = await client.getCertificateVersion(
 certificateName,
 certificateWithPolicy.properties.version
);
// Note: It will not retrieve the certificate's policy.
console.log("Certificate from a specific version:",
certificateFromVersion);

const updatedCertificate = await
client.updateCertificateProperties(certificateName, "", {
 tags: {
 customTag: "value"
 }
});
console.log("Updated certificate:", updatedCertificate);

// Updating the certificate's policy:
await client.updateCertificatePolicy(certificateName, {
 issuerName: "Self",
 subject: "cn=MyOtherCert"
});
certificateWithPolicy = await client.getCertificate(certificateName);
console.log("updatedCertificate certificate's policy:",
certificateWithPolicy.policy);

// delete certificate
const deletePoller = await
client.beginDeleteCertificate(certificateName);
const deletedCertificate = await deletePoller.pollUntilDone();
```

```

 console.log("Recovery Id: ", deletedCertificate.recoveryId);
 console.log("Deleted Date: ", deletedCertificate.deletedOn);
 console.log("Scheduled Purge Date: ",
deletedCertificate.scheduledPurgeDate);
 }

 main().catch((error) => {
 console.error("An error occurred:", error);
 process.exit(1);
});

```

## Exécuter l'exemple d'application

1. Exéutez l'application :

terminal

```
node index.js
```

2. Les méthodes de création et d'obtenir retournent un objet JSON complet pour le certificat :

JSON

```
{
 "keyId": undefined,
 "secretId": undefined,
 "name": "YOUR-CERTIFICATE-NAME",
 "reuseKey": false,
 "keyCurveName": undefined,
 "exportable": true,
 "issuerName": 'Self',
 "certificateType": undefined,
 "certificateTransparency": undefined
},
"properties": {
 "createdOn": 2021-11-29T20:17:45.000Z,
 "updatedOn": 2021-11-29T20:17:45.000Z,
 "expiresOn": 2022-11-29T20:17:45.000Z,
 "id": "https://YOUR-KEY-VAULT-
NAME.vault.azure.net/certificates/YOUR-CERTIFICATE-NAME/YOUR-
CERTIFICATE-VERSION",
 "enabled": false,
 "notBefore": 2021-11-29T20:07:45.000Z,
 "recoveryLevel": "Recoverable+Purgeable",
 "name": "YOUR-CERTIFICATE-NAME",
 "vaultUrl": "https://YOUR-KEY-VAULT-NAME.vault.azure.net",
 "version": "YOUR-CERTIFICATE-VERSION",
 "tags": undefined,
 "x509Thumbprint": undefined,
}
```

```
 "recoverableDays": 90
 }
}
```

## Intégration à la configuration de l'application

Le kit SDK Azure fournit une méthode d'assistance, `parseKeyVaultCertificateIdentifier`, pour analyser l'ID de certificat Key Vault donné, ce qui est nécessaire si vous utilisez des références [App Configuration](#) à Key Vault. La config d'application stocke l'ID de certificat de clé Vault. Vous avez besoin de la méthode `parseKeyVaultCertificateIdentifier` pour analyser cet ID et obtenir le nom du certificat. Une fois que vous avez le nom du certificat, vous pouvez obtenir le certificat actuel à l'aide du code de ce démarrage rapide.

## Étapes suivantes

Dans ce guide de démarrage rapide, vous avez créé un coffre de clés, stocké un certificat et récupéré ce certificat. Pour en savoir plus sur la solution Key Vault et sur la manière de l'intégrer à vos applications, consultez ces articles.

- Lire la [vue d'ensemble Azure Key Vault](#)
- Lire la [vue d'ensemble des certificats](#)
- Consultez un [Tutoriel sur l'accès à Key Vault depuis une application App Service](#)
- Consultez un [Tutoriel sur l'accès à Key Vault depuis une machine virtuelle](#)
- Consulter le [Guide du développeur Azure Key Vault](#)
- Passer en revue la [Vue d'ensemble de la sécurité de Key Vault](#)

# Démarrage rapide : bibliothèque de client de clés Azure Key Vault pour JavaScript

Article • 25/03/2023

Commencez à utiliser la bibliothèque de client de clés Azure Key Vault pour JavaScript. [Azure Key Vault](#) est un service cloud qui fournit un magasin de clés de chiffrement sécurisé. Vous pouvez stocker des clés, des mots de passe, des certificats et d'autres secrets en toute sécurité. Vous pouvez créer et gérer des coffres de clés Azure grâce au portail Azure. Dans ce guide de démarrage rapide, vous allez découvrir comment créer, récupérer et supprimer des clés dans un coffre de clés Azure en utilisant la bibliothèque de client de clés JavaScript.

Ressources de la bibliothèque de client Key Vault :

[Documentation de référence sur les API](#) | [Code source de la bibliothèque](#) ↗ | [Package \(npm\)](#) ↗

Pour plus d'informations sur Key Vault et les clés, consultez :

- [Vue d'ensemble du coffre de clés](#)
- [Vue d'ensemble des clés](#)

## Prérequis

- Un abonnement Azure - [En créer un gratuitement](#) ↗
- Actuellement [Node.js LTS](#) ↗ .
- [Azure CLI](#)
- Un coffre-fort de clé existant - vous pouvez en créer un à l'aide des éléments ci-après :
  - [Azure CLI](#)
  - [Azure portal](#)
  - [Azure PowerShell](#)

Ce guide de démarrage rapide suppose que vous exécutez l'interface [Azure CLI](#).

## Connexion à Azure

1. Exécutez la commande `login`.

```
Azure CLI
```

```
az login
```

Si l'interface CLI peut ouvrir votre navigateur par défaut, elle le fait et charge une page de connexion Azure par la même occasion.

Sinon, ouvrez une page de navigateur à l'adresse <https://aka.ms/devicelogin> et entrez le code d'autorisation affiché dans votre terminal.

2. Dans le navigateur, connectez-vous avec les informations d'identification de votre compte.

## Créer une application Node.js

Créez une application Node.js qui utilise votre coffre-fort de clé.

1. Dans un terminal, créez un dossier nommé `key-vault-node-app` et modifiez-le dans ce dossier :

```
terminal
```

```
mkdir key-vault-node-app && cd key-vault-node-app
```

2. Initialiser le projet Node.js :

```
terminal
```

```
npm init -y
```

## Installer des packages Key Vault

1. À l'aide du terminal, installez la bibliothèque de client de secrets Azure Key Vault, [@azure/keyvault-keys](#) pour Node.js.

```
terminal
```

```
npm install @azure/keyvault-keys
```

2. Installez la bibliothèque de client d'identité Azure, le package [@azure/identity](#) pour vous authentifier dans un coffre de clés.

terminal

```
npm install @azure/identity
```

## Accorder l'accès à votre coffre de clés

Créez une stratégie d'accès pour votre coffre de clés, qui accorde des autorisations de clé à votre compte d'utilisateur.

Azure CLI

```
az keyvault set-policy --name <YourKeyVaultName> --upn user@domain.com --
key-permissions delete get list create update purge
```

## Définir des variables d'environnement

Cette application utilise le nom de coffre de clés en tant que variable d'environnement appelée `KEY_VAULT_NAME`.

Windows

Invite de commandes Windows

```
set KEY_VAULT_NAME=<your-key-vault-name>
```

## Authentifier et créer un client

Les requêtes d'application vers les Services Azure doivent être autorisées. L'utilisation de la méthode `DefaultAzureCredential` fournie par la [bibliothèque de client Azure Identity](#) est l'approche recommandée pour implémenter des connexions sans mot de passe aux services Azure dans votre code. `DefaultAzureCredential` prend en charge plusieurs méthodes d'authentification et détermine quelle méthode doit être utilisée au moment de l'exécution. Cette approche permet à votre application d'utiliser différentes méthodes d'authentification dans différents environnements (local ou production) sans implémenter de code spécifique à l'environnement.

Dans ce guide de démarrage rapide, `DefaultAzureCredential` s'authentifie auprès du coffre de clés à l'aide des informations d'identification de l'utilisateur de développement local connecté à Azure CLI. Quand l'application est déployée sur Azure, le même code

`DefaultAzureCredential` peut découvrir et utiliser automatiquement une identité managée affectée à un service d'application, une machine virtuelle ou d'autres services. Pour plus d'informations, consultez [Vue d'ensemble des identités managées](#).

Dans cet code, le nom de votre coffre de clés est utilisé pour créer l'URI du coffre de clés, au format `https://<your-key-vault-name>.vault.azure.net`. Pour plus d'informations sur l'authentification auprès du coffre de clés, consultez le [Guide du développeur](#).

## Exemple de code

Les exemples de code ci-dessous vous montrent comment créer un client et définir, récupérer et supprimer un secret.

Ce code utilise les [classes et méthodes de secret Key Vault](#) suivantes :

- [Classe DefaultAzureCredential](#)
- [Classe KeyClient](#)
  - [createKey](#)
  - [createEcKey](#)
  - [createRsaKey](#)
  - [getKey](#)
  - [listPropertiesOfKeys](#)
  - [updateKeyProperties](#)
  - [beginDeleteKey](#)
  - [getDeletedKey](#)
  - [purgeDeletedKey](#)

## Configurer le framework d'application

1. Créez un fichier texte et collez le code suivant dans le fichier `index.js`.

JavaScript

```
const { KeyClient } = require("@azure/keyvault-keys");
const { DefaultAzureCredential } = require("@azure/identity");

async function main() {

 // DefaultAzureCredential expects the following three environment
 // variables:
 // - AZURE_TENANT_ID: The tenant ID in Azure Active Directory
 // - AZURE_CLIENT_ID: The application (client) ID registered in the
 // AAD tenant
```

```
// - AZURE_CLIENT_SECRET: The client secret for the registered
application
const credential = new DefaultAzureCredential();

const keyVaultName = process.env["KEY_VAULT_NAME"];
if(!keyVaultName) throw new Error("KEY_VAULT_NAME is empty");
const url = "https://" + keyVaultName + ".vault.azure.net";

const client = new KeyClient(url, credential);

const uniqueString = Date.now();
const keyName = `sample-key-${uniqueString}`;
const ecKeyName = `sample-ec-key-${uniqueString}`;
const rsaKeyName = `sample-rsa-key-${uniqueString}`;

// Create key using the general method
const result = await client.createKey(keyName, "EC");
console.log("key: ", result);

// Create key using specialized key creation methods
const ecResult = await client.createEcKey(ecKeyName, { curve: "P-256" });
const rsaResult = await client.createRsaKey(rsaKeyName, { keySize: 2048 });
console.log("Elliptic curve key: ", ecResult);
console.log("RSA Key: ", rsaResult);

// Get a specific key
const key = await client.getKey(keyName);
console.log("key: ", key);

// Or list the keys we have
for await (const keyProperties of client.listPropertiesOfKeys()) {
 const key = await client.getKey(keyProperties.name);
 console.log("key: ", key);
}

// Update the key
const updatedKey = await client.updateKeyProperties(keyName,
result.properties.version, {
 enabled: false
});
console.log("updated key: ", updatedKey);

// Delete the key - the key is soft-deleted but not yet purged
const deletePoller = await client.beginDeleteKey(keyName);
await deletePoller.pollUntilDone();

const deletedKey = await client.getDeletedKey(keyName);
console.log("deleted key: ", deletedKey);

// Purge the key - the key is permanently deleted
// This operation could take some time to complete
console.time("purge a single key");
await client.purgeDeletedKey(keyName);
```

```
 console.timeEnd("purge a single key");
 }

 main().catch((error) => {
 console.error("An error occurred:", error);
 process.exit(1);
});
```

## Exécuter l'exemple d'application

1. Exéutez l'application :

```
terminal
```

```
node index.js
```

2. Les méthodes de création et d'obtenir retournent un objet JSON complet pour le certificat :

```
JSON
```

```
"key": {
 "key": {
 "kid": "https://YOUR-KEY-VAULT-NAME.vault.azure.net/keys/YOUR-KEY-NAME/YOUR-KEY-VERSION",
 "kty": "YOUR-KEY-TYPE",
 "keyOps": [ARRAY-OF-VALID-OPERATIONS],
 ... other properties based on key type
 },
 "id": "https://YOUR-KEY-VAULT-NAME.vault.azure.net/keys/YOUR-KEY-NAME/YOUR-KEY-VERSION",
 "name": "YOUR-KEY-NAME",
 "keyOperations": [ARRAY-OF-VALID-OPERATIONS],
 "keyType": "YOUR-KEY-TYPE",
 "properties": {
 "tags": undefined,
 "enabled": true,
 "notBefore": undefined,
 "expiresOn": undefined,
 "createdOn": 2021-11-29T18:29:11.000Z,
 "updatedOn": 2021-11-29T18:29:11.000Z,
 "recoverableDays": 90,
 "recoveryLevel": "Recoverable+Purgeable",
 "exportable": undefined,
 "releasePolicy": undefined,
 "vaultUrl": "https://YOUR-KEY-VAULT-NAME.vault.azure.net",
 "version": "YOUR-KEY-VERSION",
 "name": "YOUR-KEY-VAULT-NAME",
 "managed": undefined,
 "id": "https://YOUR-KEY-VAULT-NAME.vault.azure.net/keys/YOUR-KEY-
```

```
 NAME/YOUR-KEY-VERSION"
}
}
```

# Intégration à la configuration de l'application

Le SDK Azure fournit une méthode d'aide, [parseKeyVaultCertificateIdentifier](#), pour analyser l'ID du certificat Key Vault donné. Cette étape est nécessaire si vous utilisez des références [de configuration d'application](#) à Key Vault. App Config stocke l'ID de la clé du coffre-fort. Vous avez besoin de la méthode *parseKeyVaultCertificateIdentifier* pour analyser cet ID et obtenir le nom du certificat. Une fois que vous avez le nom de la clé, vous pouvez obtenir la valeur de clé actuelle à l'aide du code de ce démarrage rapide.

## Étapes suivantes

Dans ce guide de démarrage rapide, vous avez créé un coffre de clés, stocké une clé et récupéré cette clé. Pour en savoir plus sur Key Vault et sur la manière de l'intégrer à vos applications, consultez ces articles.

- Lire la [vue d'ensemble Azure Key Vault](#)
- Lire une [présentation des clés Azure Key Vault](#)
- Découvrir comment [Sécuriser l'accès à un coffre de clés](#)
- Consulter le [Guide du développeur Azure Key Vault](#)
- Passer en revue la [Vue d'ensemble de la sécurité de Key Vault](#)

# Démarrage rapide – Bibliothèque de client de secrets Azure Key Vault pour JavaScript

Article • 24/05/2023

Bien démarrer avec la bibliothèque de client de secrets Azure Key Vault pour JavaScript. [Azure Key Vault](#) est un service cloud qui fournit un magasin de secrets sécurisé. Vous pouvez stocker des clés, des mots de passe, des certificats et d'autres secrets en toute sécurité. Vous pouvez créer et gérer des coffres de clés Azure grâce au portail Azure. Dans ce guide de démarrage rapide, vous allez découvrir comment créer, récupérer et supprimer des secrets dans un coffre de clés Azure à l'aide de la bibliothèque de client JavaScript.

Ressources de la bibliothèque de client Key Vault :

[Documentation de référence sur les API](#) | [Code source de la bibliothèque](#) ↗ | [Package \(npm\)](#) ↗

Pour plus d'informations sur Key Vault et les secrets, consultez :

- [Vue d'ensemble du coffre de clés](#)
- [Vue d'ensemble des secrets](#)

## Prérequis

- Un abonnement Azure - [En créer un gratuitement](#) ↗
- Actuellement [Node.js LTS](#) ↗ .
- [Azure CLI](#)
- Un coffre-fort de clé existant - vous pouvez en créer un à l'aide des éléments ci-après :
  - [Azure CLI](#)
  - [Azure portal](#)
  - [Azure PowerShell](#)

Ce guide de démarrage rapide suppose que vous exécutez [Azure CLI](#).

## Connexion à Azure

1. Exécutez la commande `login`.

```
Azure CLI
```

```
az login
```

Si l'interface CLI peut ouvrir votre navigateur par défaut, elle le fait et charge une page de connexion Azure par la même occasion.

Sinon, ouvrez une page de navigateur à l'adresse <https://aka.ms/devicelogin> et entrez le code d'autorisation affiché dans votre terminal.

2. Dans le navigateur, connectez-vous avec les informations d'identification de votre compte.

## Créer une application Node.js

Créez une application Node.js qui utilise votre coffre-fort de clé.

1. Dans un terminal, créez un dossier nommé `key-vault-node-app` et modifiez-le dans ce dossier :

```
terminal
```

```
mkdir key-vault-node-app && cd key-vault-node-app
```

2. Initialiser le projet Node.js :

```
terminal
```

```
npm init -y
```

## Installer des packages Key Vault

1. À l'aide du terminal, installez la bibliothèque de client de secrets Azure Key Vault, [@azure/keyvault-secrets](#) pour Node.js.

```
terminal
```

```
npm install @azure/keyvault-secrets
```

2. Installez la bibliothèque de client d'identité Azure, le package [@azure/identity](#) pour vous authentifier dans un coffre de clés.

terminal

```
npm install @azure/identity
```

## Accorder l'accès à votre coffre de clés

Créez une politique d'accès pour votre coffre-fort de clés qui accorde des autorisations secrètes à votre compte d'utilisateur avec la commande [az keyvault set-policy](#).

Azure CLI

```
az keyvault set-policy --name <your-key-vault-name> --upn user@domain.com --
secret-permissions delete get list set purge update
```

## Définir des variables d'environnement

Cette application utilise le nom de coffre de clés en tant que variable d'environnement appelée `KEY_VAULT_NAME`.

Windows

Invite de commandes Windows

```
set KEY_VAULT_NAME=<your-key-vault-name>
```

## Authentifier et créer un client

Les requêtes d'application vers les Services Azure doivent être autorisées. L'utilisation de la méthode `DefaultAzureCredential` fournie par la [bibliothèque de client Azure Identity](#) est l'approche recommandée pour implémenter des connexions sans mot de passe aux services Azure dans votre code. `DefaultAzureCredential` prend en charge plusieurs méthodes d'authentification et détermine quelle méthode doit être utilisée au moment de l'exécution. Cette approche permet à votre application d'utiliser différentes méthodes d'authentification dans différents environnements (local ou production) sans implémenter de code spécifique à l'environnement.

Dans ce guide de démarrage rapide, `DefaultAzureCredential` s'authentifie auprès du coffre de clés à l'aide des informations d'identification de l'utilisateur de développement local connecté à Azure CLI. Quand l'application est déployée sur Azure, le même code

`DefaultAzureCredential` peut découvrir et utiliser automatiquement une identité managée affectée à un service d'application, une machine virtuelle ou d'autres services. Pour plus d'informations, consultez [Vue d'ensemble des identités managées](#).

Dans cet code, le nom de votre coffre de clés est utilisé pour créer l'URI du coffre de clés, au format `https://<your-key-vault-name>.vault.azure.net`. Pour plus d'informations sur l'authentification auprès du coffre de clés, consultez le [Guide du développeur](#).

## Exemple de code

Les exemples de code ci-dessous vous montrent comment créer un client et définir, récupérer et supprimer un secret.

Ce code utilise les [classes et méthodes de secret Key Vault](#) suivantes :

- `DefaultAzureCredential`
- Classe `SecretClient`
  - `setSecret`
  - `getSecret`
  - `updateSecretProperties`
  - `beginDeleteSecret`

## Configurer le framework d'application

1. Créez un fichier texte et collez le code suivant dans le fichier `index.js`.

JavaScript

```
const { SecretClient } = require("@azure/keyvault-secrets");
const { DefaultAzureCredential } = require("@azure/identity");

async function main() {
 // If you're using MSI, DefaultAzureCredential should "just work".
 // Otherwise, DefaultAzureCredential expects the following three
 environment variables:
 // - AZURE_TENANT_ID: The tenant ID in Azure Active Directory
 // - AZURE_CLIENT_ID: The application (client) ID registered in the
 AAD tenant
 // - AZURE_CLIENT_SECRET: The client secret for the registered
 application
 const credential = new DefaultAzureCredential();

 const keyVaultName = process.env["KEY_VAULT_NAME"];
 if(!keyVaultName) throw new Error("KEY_VAULT_NAME is empty");
 const url = "https://" + keyVaultName + ".vault.azure.net";
```

```

const client = new SecretClient(url, credential);

// Create a secret
// The secret can be a string of any kind. For example,
// a multiline text block such as an RSA private key with newline
// characters,
// or a stringified JSON object, like `JSON.stringify({ mySecret:
'MySecretValue'})`.
const uniqueString = new Date().getTime();
const secretName = `secret${uniqueString}`;
const result = await client.setSecret(secretName, "MySecretValue");
console.log("result: ", result);

// Read the secret we created
const secret = await client.getSecret(secretName);
console.log("secret: ", secret);

// Update the secret with different attributes
const updatedSecret = await client.updateSecretProperties(secretName,
result.properties.version, {
 enabled: false
});
console.log("updated secret: ", updatedSecret);

// Delete the secret immediately without ability to restore or purge.
await client.beginDeleteSecret(secretName);
}

main().catch((error) => {
 console.error("An error occurred:", error);
 process.exit(1);
});

```

## Exécuter l'exemple d'application

1. Exéutez l'application :

terminal

```
node index.js
```

2. Les méthodes de création et d'obtenir retournent un objet JSON complet pour le certificat :

JSON

```
{
 "value": "MySecretValue",
 "name": "secret1637692472606",
```

```

 "properties": {
 "createdOn": "2021-11-23T18:34:33.000Z",
 "updatedOn": "2021-11-23T18:34:33.000Z",
 "enabled": true,
 "recoverableDays": 90,
 "recoveryLevel": "Recoverable+Purgeable",
 "id": "https://YOUR-KEYVAULT-
NAME.vault.azure.net/secrets/secret1637692472606/YOUR-VERSION",
 "vaultUrl": "https://YOUR-KEYVAULT-NAME.vault.azure.net",
 "version": "YOUR-VERSION",
 "name": "secret1637692472606"
 }
}

```

La méthode de mise à jour renvoie les propriétés nom/valeurs de propriétés :

JSON

```

"createdOn": "2021-11-23T18:34:33.000Z",
"updatedOn": "2021-11-23T18:34:33.000Z",
"enabled": true,
"recoverableDays": 90,
"recoveryLevel": "Recoverable+Purgeable",
"id": "https://YOUR-KEYVAULT-
NAME.vault.azure.net/secrets/secret1637692472606/YOUR-VERSION",
"vaultUrl": "https://YOUR-KEYVAULT-NAME.vault.azure.net",
"version": "YOUR-VERSION",
"name": "secret1637692472606"

```

## Intégration à la configuration de l'application

Le SDK Azure fournit une méthode d'aide, [parseKeyVaultCertificateIdentifier](#), pour analyser l'ID du certificat Key Vault donné. Cette étape est nécessaire si vous utilisez des références [de configuration d'application](#) à Key Vault. App Config stocke l'ID de la clé du coffre-fort. Vous avez besoin de la méthode *parseKeyVaultCertificateIdentifier* pour analyser cet ID et obtenir le nom du certificat. Une fois que vous avez le nom de la clé, vous pouvez obtenir la valeur de clé actuelle à l'aide du code de ce démarrage rapide.

## Étapes suivantes

Dans ce guide de démarrage rapide, vous avez créé un coffre de clés, stocké un secret et récupéré ce secret. Pour en savoir plus sur Key Vault et sur la manière de l'intégrer à vos applications, consultez les articles ci-dessous.

- Lire la [vue d'ensemble Azure Key Vault](#)
- Lire une [présentation des secrets Azure Key Vault](#).

- Découvrir comment [Sécuriser l'accès à un coffre de clés](#)
- Consulter le [Guide du développeur Azure Key Vault](#)
- Passer en revue la [Vue d'ensemble de la sécurité de Key Vault](#)

# Envoyer et recevoir des messages à partir de files d'attente Azure Service Bus (JavaScript)

Article • 08/12/2023

Dans ce tutoriel, vous allez effectuer les étapes suivantes :

1. Créer un espace de noms Service Bus à l'aide du Portail Azure.
2. Créer une file d'attente Service Bus à l'aide du portail Azure.
3. Écrivez une application JavaScript qui utilise le package [@azure/service-bus](#) pour effectuer les actions suivantes :
  - a. Envoyer un ensemble de messages à la file d'attente
  - b. Recevoir ces messages de la file d'attente.

## ⓘ Notes

Ce guide de démarrage rapide fournit des instructions pas à pas pour un scénario simple qui consiste à envoyer des messages à une file d'attente Service Bus et à les recevoir. Vous trouverez des exemples JavaScript et TypeScript prédéfinis pour Azure Service Bus dans le dépôt du [SDK Azure pour JavaScript sur GitHub](#).

## Prérequis

Si vous débutez avec le service, consultez [Vue d'ensemble de Service Bus](#) avant de suivre ce démarrage rapide.

- Un abonnement Azure. Pour suivre ce tutoriel, vous avez besoin d'un compte Azure. Vous pouvez [activer les avantages de votre abonnement MSDN](#) ou vous inscrire pour un [compte gratuit](#).
- [Node.js LTS](#)

Sans mot de passe

Pour utiliser ce guide de démarrage rapide avec votre propre compte Azure, vous avez besoin des éléments suivants :

- Installez [Azure CLI](#), qui fournit l'authentification sans mot de passe à votre ordinateur de développement.

- Connectez-vous avec votre compte Azure au terminal ou à l'invite de commandes avec `az login`.
- Utilisez le même compte lorsque vous ajoutez le rôle de données approprié à votre ressource.
- Exécutez le code dans le même terminal ou la même invite de commande.
- Notez le nom de la **file d'attente** de votre espace de noms Service Bus. Vous en aurez besoin dans le code.

### ⓘ Notes

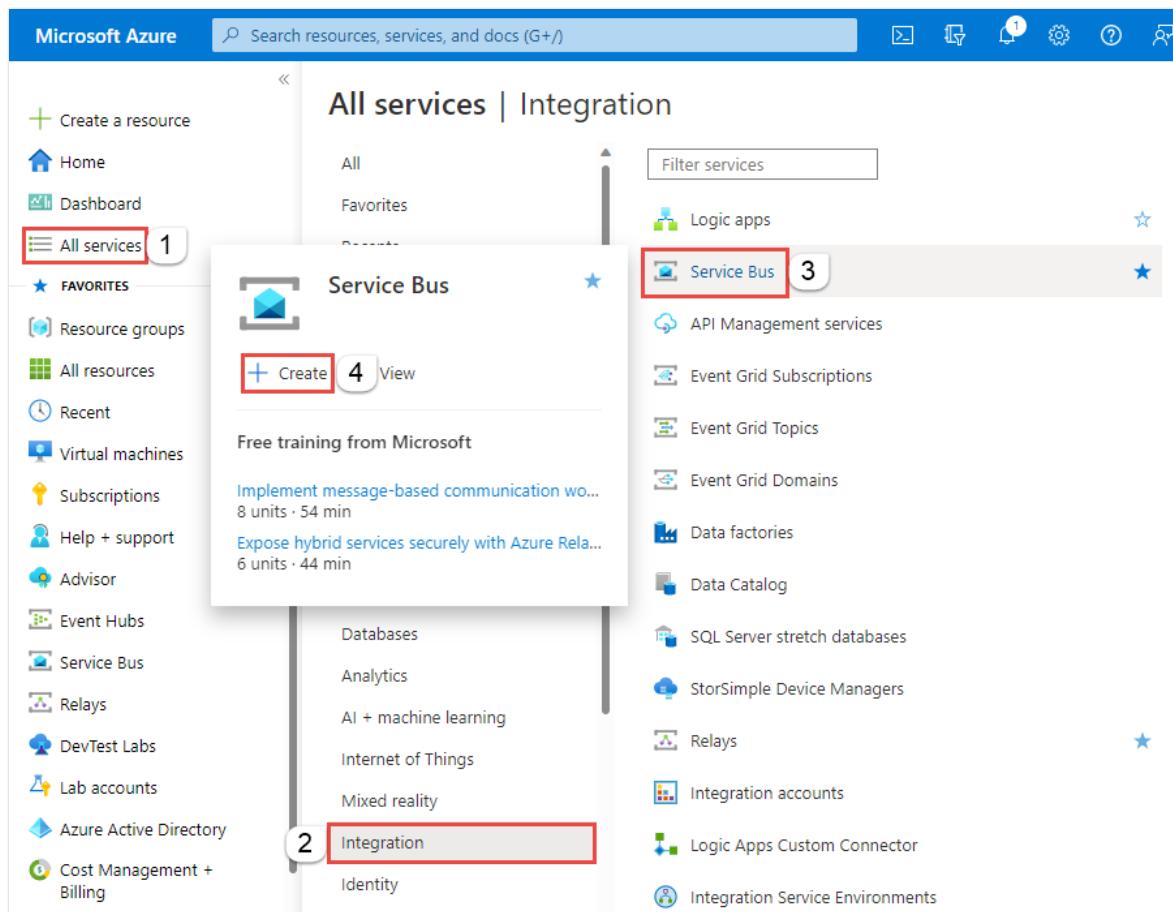
Ce didacticiel utilise des exemples que vous pouvez copier et exécuter à l'aide de [Nodejs](#). Pour obtenir des instructions sur la création d'une application Node.js, veuillez consulter les pages [Création et déploiement d'une application Node.js sur un site web Azure](#) ou [Service cloud Node.js avec Windows PowerShell](#).

## Créer un espace de noms dans le Portail Azure

Pour commencer à utiliser des entités de messagerie Service Bus dans Azure, vous devez d'abord créer un espace de noms avec un nom unique dans Azure. Un espace de noms fournit un conteneur d'étendue pour les ressources du Service Bus (files d'attente, thèmes, etc.) au sein de votre application.

Pour créer un espace de noms :

1. Connectez-vous au [portail Azure](#).
2. Dans le volet de navigation gauche du portail, sélectionnez **Tous les services**, puis **Intégration** dans la liste des catégories, pointez la souris sur **Service Bus**, puis sélectionnez le bouton + sur la vignette Service Bus.



3. Dans l'étiquette **De base** de la page **Créer un espace de noms**, suivez ces étapes :

- Pour l'option **Abonnement**, choisissez un abonnement Azure dans lequel créer l'espace de noms.
- Pour l'option **Groupe de ressources**, choisissez un groupe de ressources existant dans lequel l'espace de noms sera utilisé, ou créez-en un nouveau.
- Entrez un **nom pour l'espace de noms**. Le nom de l'espace de noms doit respecter les conventions de nommage suivantes :
  - Le nom doit être unique dans tout Azure. Le système vérifie immédiatement si le nom est disponible.
  - Le nom doit inclure entre 6 et 50 caractères.
  - Le nom ne peut contenir que des lettres, des chiffres et des traits d'union (« - »).
  - Le nom doit commencer par une lettre, et se terminer par une lettre ou un chiffre.
  - Le nom ne se termine ni par « -sb » ni par « -mgmt ».
- Pour l'option **Emplacement**, choisissez la région dans laquelle héberger votre espace de noms.

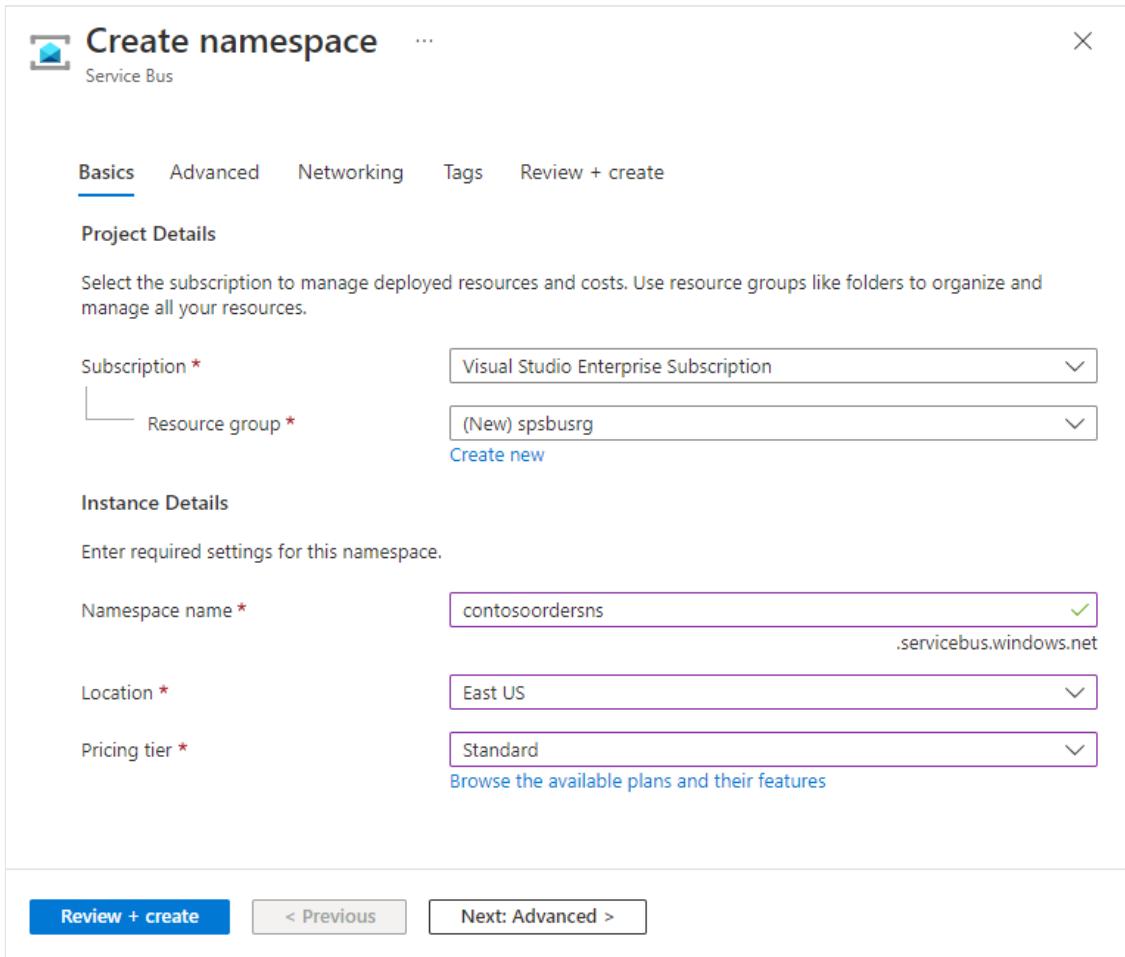
- e. Pour le **Niveau tarifaire**, sélectionnez le SKU (De base, Standard ou Premium) destiné à l'espace de noms. Pour ce guide de démarrage rapide, sélectionnez **Standard**.

 **Important**

Si vous voulez utiliser des **rubriques et des abonnements**, choisissez Standard ou Premium. Les rubriques/abonnements ne sont pas pris en charge dans le niveau tarifaire De base.

Si vous avez sélectionné le SKU **Premium**, précisez le nombre d'**unité de messagerie**. Le niveau Premium isole les ressources au niveau du processeur et de la mémoire, ce qui permet d'exécuter chaque charge de travail de manière isolée. Ce conteneur de ressources est appelé unité de messagerie. Un espace de noms Premium a au moins une unité de messagerie. Vous pouvez sélectionner 1, 2, 4, 8 ou 16 unités de messagerie pour chaque espace de noms Service Bus Premium. Pour plus d'informations, consultez [Messagerie Service Bus Premium](#).

- f. Au bas de la page, sélectionnez **Examiner et créer**.



The screenshot shows the 'Create namespace' wizard in the Azure portal. The title bar says 'Create namespace' and 'Service Bus'. The tabs at the top are 'Basics', 'Advanced', 'Networking', 'Tags', and 'Review + create'. The 'Basics' tab is selected. The 'Project Details' section contains fields for 'Subscription' (set to 'Visual Studio Enterprise Subscription') and 'Resource group' (set to '(New) spsbusrg'). Below that is the 'Instance Details' section, which includes 'Namespace name' (set to 'contosoordersns'), 'Location' (set to 'East US'), and 'Pricing tier' (set to 'Standard'). At the bottom of the screen are three buttons: 'Review + create' (highlighted in blue), '< Previous', and 'Next: Advanced >'.

g. Dans la page **Vérifier + créer**, passez en revue les paramètres, puis sélectionnez **Créer**.

4. Une fois le déploiement de la ressource réussi, sélectionnez **Accéder à la ressource** dans la page de déploiement.

The screenshot shows the 'Deployment' overview for a resource named 'contosoordersns'. A green checkmark indicates that the deployment is complete. The deployment details include the name 'contosoordersns', subscription 'Visual Studio Enterprise Subscription', and resource group 'spsbusrg'. The start time was 10/20/2022, 4:45:03 PM, and the correlation ID is a453ace1-bab9-4c4a-81ad-a1c5366460ea. There are sections for 'Deployment details' and 'Next steps', with a prominent blue 'Go to resource' button. Below the main content, there are links to 'Give feedback' and 'Tell us about your experience with deployment'.

5. Vous voyez la page d'accueil de votre espace de noms Service Bus.

The screenshot shows the 'Overview' page for a Service Bus namespace named 'spsbusns1028'. The left sidebar lists navigation options like 'Activity log', 'Access control (IAM)', 'Tags', 'Diagnose and solve problems', 'Settings' (with sub-options for Shared access policies, Geo-Recovery, Migrate to premium, Encryption, Configuration, Properties, Locks), 'Entities' (Queues, Topics), 'Monitoring' (Insights (Preview), Alerts, Metrics, Diagnostic settings, Logs), and 'Logs'. The main area displays 'Essentials' information such as Resource group (spbusrg), Status (Active), Location (East US), Subscription (Visual Studio Enterprise Subscription), and Tags. It also shows performance metrics for Requests and Messages over the last hour, with zero activity shown. A 'JSON View' link is available in the top right corner.

## Créer une file d'attente dans le portail Azure

1. Dans la page **Espace de noms Service Bus**, sélectionnez **Files d'attente** dans le menu de navigation de gauche.
2. Dans la page **Files d'attente**, sélectionnez **+ File d'attente** dans la barre d'outils.

3. Entrez un **nom** pour la file d'attente et laissez les valeurs par défaut des autres valeurs.

4. À présent, sélectionnez **Créer**.

The screenshot shows the Azure Service Bus Queues creation interface. On the left, the navigation pane is open with the 'Queues' option selected (marked with a red box and number 1). In the center, there's a search bar and a 'Queue' button (marked with a red box and number 2). A 'Create queue' dialog box is open on the right, containing fields for 'Name' (marked with a red box and number 3), 'Max queue size' (1 GB), 'Max delivery count' (10), 'Message time to live' (14 days, 0 hours, 0 minutes, 0 seconds), 'Lock duration' (0 days, 0 hours, 0 minutes, 30 seconds), and several optional checkboxes. A 'Create' button is at the bottom of the dialog (marked with a red box and number 4).

## Authentifier l'application sur Azure

Ce guide de démarrage pratique vous montre deux façons de vous connecter à Azure Service Bus : **sans mot de passe** et avec une **chaîne de connexion**.

La première option vous explique comment utiliser votre principal de sécurité dans Microsoft Entra ID et le contrôle d'accès en fonction du rôle (RBAC) pour vous connecter à un espace de noms Service Bus. Vous n'avez pas à vous soucier d'avoir une chaîne de connexion codée en dur dans votre code, dans un fichier config ni dans un stockage sécurisé comme Azure Key Vault.

La deuxième option consiste à se servir d'une chaîne de connexion pour se connecter à un espace de noms Service Bus. Si vous débutez avec Azure, vous trouverez peut-être l'option chaîne de connexion plus facile à suivre. Nous vous recommandons d'utiliser l'option sans mot de passe dans les applications réelles et les environnements de production. Pour plus d'informations, consultez [Authentification et autorisation](#). Pour en savoir plus sur l'authentification sans mot de passe, reportez-vous à la [page de présentation](#).

Sans mot de passe (recommandé)

## Attribuer des rôles à votre utilisateur Microsoft Entra

Lors du développement localement, assurez-vous que le compte d'utilisateur qui se connecte à Azure Service Bus dispose des autorisations appropriées. Vous aurez besoin du rôle [Propriétaire de données Azure Service Bus](#) pour envoyer et recevoir des messages. Pour vous attribuer ce rôle, vous aurez besoin du rôle Administrateur de l'accès utilisateur ou d'un autre rôle qui inclut l'action

`Microsoft.Authorization/roleAssignments/write`. Vous pouvez attribuer des rôles RBAC Azure à un utilisateur à l'aide du Portail Azure, Azure CLI ou Azure PowerShell. Découvrez les étendues disponibles pour les attributions de rôles dans la page [vue d'ensemble de l'étendue](#).

L'exemple suivant attribue le rôle [Azure Service Bus Data Owner](#) à votre compte d'utilisateur, qui fournit un accès complet aux ressources Azure Service Bus. Dans un scénario réel, suivez le [principe des privilèges minimum](#) pour accorder aux utilisateurs uniquement les autorisations minimales nécessaires à un environnement de production plus sécurisé.

## Rôles Azure intégrés pour Azure Service Bus

Pour Azure Service Bus, la gestion des espaces de noms et de toutes les ressources associées via le Portail Azure et l'API de gestion des ressources Azure est déjà protégée à l'aide du modèle Azure RBAC. Azure fournit les rôles Azure intégrés ci-dessous pour autoriser l'accès à un espace de noms Service Bus :

- [Propriétaire de données Azure Service Bus](#) : ce rôle permet l'accès aux données de l'espace de noms Service Bus et de ses entités (files d'attente, rubriques, abonnements et filtres). Un membre de ce rôle peut envoyer et recevoir des messages à partir de files d'attente ou de rubriques et d'abonnements.
- [Expéditeur de données Azure Service Bus](#) : utilisez ce rôle pour autoriser l'accès en envoi à l'espace de noms Service Bus et à ses entités.
- [Récepteur de données Azure Service Bus](#) : utilisez ce rôle pour autoriser l'accès en réception à l'espace de noms Service Bus et à ses entités.

Si vous souhaitez créer un rôle personnalisé, consultez [Droits requis pour les opérations Service Bus](#).

# Ajouter un utilisateur Microsoft Entra au rôle Propriétaire Azure Service Bus

Ajoutez votre nom d'utilisateur Microsoft Entra au rôle **Propriétaire de données Azure Service Bus** au niveau de l'espace de noms Service Bus. Il permet à une application exécutée dans le contexte de votre compte d'utilisateur d'envoyer des messages à une file d'attente ou à une rubrique et d'en recevoir auprès d'une file d'attente ou de l'abonnement d'une rubrique.

## Important

Dans la plupart des cas, la propagation de l'attribution de rôle dans Azure peut prendre une ou deux minutes. Dans de rares cas, cela peut prendre jusqu'à **huit minutes**. Si vous recevez des erreurs d'authentification lorsque vous exécutez votre code pour la première fois, patientez quelques instants et réessayez.

1. Si la page Espace de noms Service Bus n'est pas ouverte sur le Portail Azure, recherchez votre espace de noms Service Bus à l'aide de la barre de recherche principale ou du volet de navigation de gauche.
2. Dans la page vue d'ensemble, sélectionnez **Contrôle d'accès (IAM)** dans le menu de gauche.
3. Sur la page **Contrôle d'accès (IAM)**, sélectionnez l'onglet **Attributions de rôles**.
4. Sélectionnez **+ Ajouter** dans le menu supérieur, puis **Ajouter une attribution de rôle** dans le menu déroulant résultant.

The screenshot shows the 'Access control (IAM)' section of the Azure Service Bus Namespace. The left sidebar has 'Access control (IAM)' selected. The main area has a tooltip: 'o read this customer directory, so some options are disabled. If you require access to these options, click here.' Below it are tabs: 'Check access', 'Role assignments', 'Roles', 'Deny assignments', and 'Classic administrators'. Under 'My access', there's a 'View my access' button. Under 'Check access', there's a 'Check access' button. Two boxes are shown: 'Grant access to this resource' (with 'Add role assignment' and 'Learn more' buttons) and 'View access to this resource' (with 'View' and 'Learn more' buttons).

5. Utilisez la zone de recherche pour filtrer les résultats sur le rôle souhaité. Pour cet exemple, recherchez **Azure Service Bus Data Owner** et sélectionnez le résultat correspondant. Ensuite, choisissez **Suivant**.
6. Sous **Attribuer l'accès à**, sélectionnez **Utilisateur, groupe ou principal de service**, puis sélectionnez + **Sélectionner des membres**.
7. Dans la boîte de dialogue, recherchez votre nom d'utilisateur Microsoft Entra (généralement votre adresse e-mail *user@domain*), puis choisissez **Sélectionner** en bas de la boîte de dialogue.
8. Sélectionnez **Vérifier + affecter** pour accéder à la page finale, puis **Vérifier + attribuer** à nouveau pour terminer le processus.

## Utilisation de Node Package Manager (NPM) pour installer le package

Sans mot de passe

1. Pour installer le ou les packages npm requis pour Service Bus, ouvrez une invite de commandes dont le chemin contient `npm`, remplacez le répertoire par le dossier où vous souhaitez stocker vos exemples, puis exécutez cette commande.

## 2. Installez les packages suivants :

Bash

```
npm install @azure/service-bus @azure/identity
```

# Envoi de messages à une file d'attente

L'exemple de code suivant illustre comment envoyer un message à une file d'attente.

Sans mot de passe

Vous devez avoir ouvert une session avec `az login` dans Azure CLI pour que votre ordinateur local fournisse l'authentification sans mot de passe requise dans ce code.

1. Ouvrez votre éditeur favori, tel que [Visual Studio Code](#).
2. Créez un fichier appelé `send.js` et collez-y le code ci-dessous. Ce code envoie les noms des scientifiques à votre file d'attente sous forme de messages.

### ⓘ Important

Les informations d'identification sans mot de passe sont fournies avec [DefaultAzureCredential](#).

JavaScript

```
const { ServiceBusClient } = require("@azure/service-bus");
const { DefaultAzureCredential } = require("@azure/identity");

// Replace `<SERVICE-BUS-NAMESPACE>` with your namespace
const fullyQualifiedNamespace = "<SERVICE-BUS-
NAMESPACE>.servicebus.windows.net";

// Passwordless credential
const credential = new DefaultAzureCredential();

// name of the queue
const queueName = "<QUEUE NAME>

const messages = [
 { body: "Albert Einstein" },
 { body: "Werner Heisenberg" },
 { body: "Marie Curie" },
```

```
{ body: "Steven Hawking" },
{ body: "Isaac Newton" },
{ body: "Niels Bohr" },
{ body: "Michael Faraday" },
{ body: "Galileo Galilei" },
{ body: "Johannes Kepler" },
{ body: "Nikolaus Kopernikus" }
];

async function main() {
 // create a Service Bus client using the passwordless
 authentication to the Service Bus namespace
 const sbClient = new ServiceBusClient(fullyQualifiedNamespace,
 credential);

 // createSender() can also be used to create a sender for a
 topic.
 const sender = sbClient.createSender(queueName);

 try {
 // Tries to send all messages in a single batch.
 // Will fail if the messages cannot fit in a batch.
 // await sender.sendMessages(messages);

 // create a batch object
 let batch = await sender.createMessageBatch();
 for (let i = 0; i < messages.length; i++) {
 // for each message in the array

 // try to add the message to the batch
 if (!batch.tryAddMessage(messages[i])) {
 // if it fails to add the message to the current
batch
 // send the current batch as it is full
 await sender.sendMessages(batch);

 // then, create a new batch
 batch = await sender.createMessageBatch();

 // now, add the message failed to be added to the
previous batch to this batch
 if (!batch.tryAddMessage(messages[i])) {
 // if it still can't be added to the batch, the
message is probably too big to fit in a batch
 throw new Error("Message too big to fit in a
batch");
 }
 }
 }

 // Send the last created batch of messages to the queue
 await sender.sendMessages(batch);

 console.log(`Sent a batch of messages to the queue:
${queueName}`);
 }
}
```

```
// Close the sender
 await sender.close();
} finally {
 await sbClient.close();
}
}

// call the main function
main().catch((err) => {
 console.log("Error occurred: ", err);
 process.exit(1);
});
```

3. Remplacez <SERVICE-BUS-NAMESPACE> par le nom de votre espace de noms Service Bus.
4. et remplacez <QUEUE NAME> par le nom de la file d'attente.
5. Ensuite, exécutez la commande à partir d'une invite de commandes pour exécuter ce fichier.

Console

```
node send.js
```

6. Vous devez voir la sortie suivante.

Console

```
Sent a batch of messages to the queue: myqueue
```

## Réception des messages d'une file d'attente

Sans mot de passe

Vous devez avoir ouvert une session avec `az login` dans Azure CLI pour que votre ordinateur local fournisse l'authentification sans mot de passe requise dans ce code.

1. Ouvrez votre éditeur favori, tel que [Visual Studio Code](#)
2. Créez un fichier appelé `receive.js` et collez-y le code suivant.

JavaScript

```
const { delay, ServiceBusClient, ServiceBusMessage } =
require("@azure/service-bus");
const { DefaultAzureCredential } = require("@azure/identity");

// Replace `<SERVICE-BUS-NAMESPACE>` with your namespace
const fullyQualifiedNamespace = "<SERVICE-BUS-
NAMESPACE>.servicebus.windows.net";

// Passwordless credential
const credential = new DefaultAzureCredential();

// name of the queue
const queueName = "<QUEUE NAME>

async function main() {
 // create a Service Bus client using the passwordless
 authentication to the Service Bus namespace
 const sbClient = new ServiceBusClient(fullyQualifiedNamespace,
 credential);

 // createReceiver() can also be used to create a receiver for a
 subscription.
 const receiver = sbClient.createReceiver(queueName);

 // function to handle messages
 const myMessageHandler = async (messageReceived) => {
 console.log(`Received message: ${messageReceived.body}`);
 };

 // function to handle any errors
 const myErrorHandler = async (error) => {
 console.log(error);
 };

 // subscribe and specify the message and error handlers
 receiver.subscribe({
 processMessage: myMessageHandler,
 processError: myErrorHandler
 });

 // Waiting long enough before closing the sender to send
 messages
 await delay(20000);

 await receiver.close();
 await sbClient.close();
}

// call the main function
main().catch((err) => {
 console.log("Error occurred: ", err);
 process.exit(1);
});
```

3. Remplacez <SERVICE-BUS-NAMESPACE> par le nom de votre espace de noms Service Bus.
4. et remplacez <QUEUE NAME> par le nom de la file d'attente.
5. Ensuite, exécutez la commande à partir d'une invite de commandes pour exécuter ce fichier.

```
Console
```

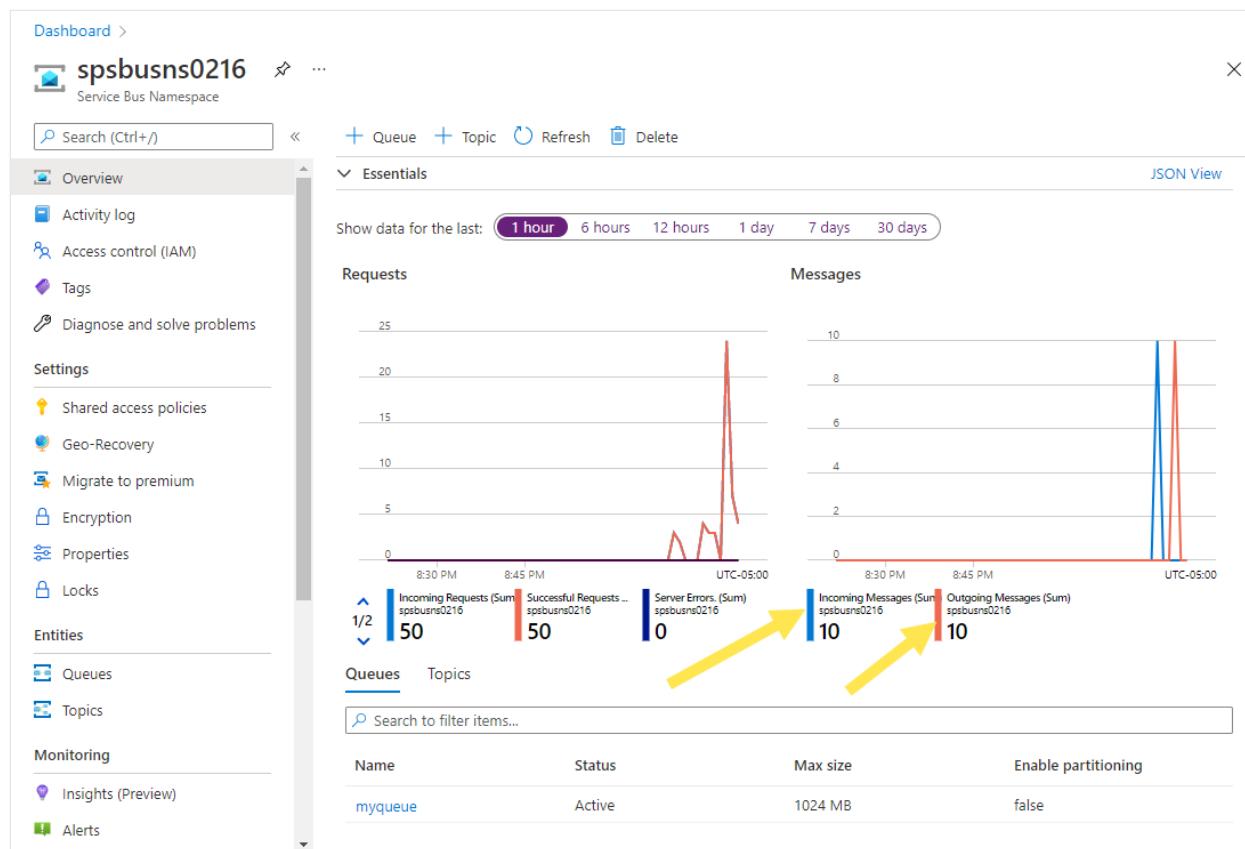
```
node receive.js
```

Vous devez voir la sortie suivante.

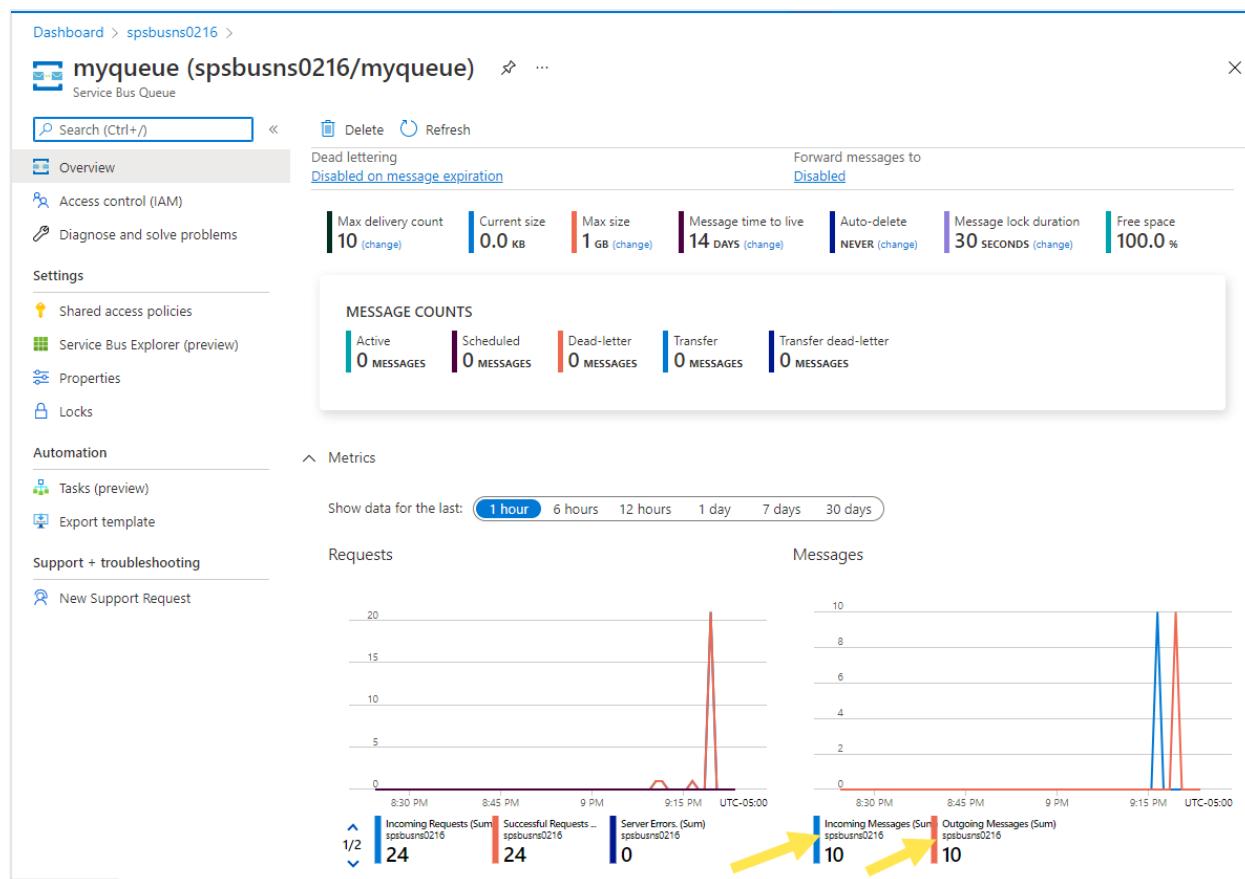
```
Console
```

```
Received message: Albert Einstein
Received message: Werner Heisenberg
Received message: Marie Curie
Received message: Steven Hawking
Received message: Isaac Newton
Received message: Niels Bohr
Received message: Michael Faraday
Received message: Galileo Galilei
Received message: Johannes Kepler
Received message: Nikolaus Kopernikus
```

Dans la page **Vue d'ensemble** de l'espace de noms Service Bus dans le portail Azure, vous pouvez voir le nombre de messages **entrants** et **sortants**. Vous devrez peut-être attendre environ une minute puis actualiser la page pour voir les valeurs les plus récentes.



Sélectionnez la file d'attente dans cette page **Vue d'ensemble** pour accéder à la page **File d'attente Service Bus**. Le nombre de messages entrants et sortants est visible dans cette page, ainsi que d'autres informations telles que la **taille actuelle** de la file d'attente, la **taille maximale** et le **nombre de messages actifs**.



# Résolution des problèmes

Si vous recevez l'une des erreurs suivantes alors que vous exécutez la version **sans mot de passe** du code JavaScript, vérifiez que vous avez ouvert une session avec la commande `az login` d'Azure CLI et que le **rôle approprié** est appliqué à votre compte d'utilisateur Azure :

- Une ou plusieurs revendications « Send » sont requises pour effectuer cette opération.
- Une ou plusieurs revendications « Receive » sont requises pour effectuer cette opération.

## Nettoyer les ressources

Accédez à votre espace de noms Service Bus dans le Portail Azure, puis sélectionnez **Supprimer** sur le Portail Azure pour supprimer l'espace de noms et la file d'attente qu'il contient.

## Étapes suivantes

Voir la documentation et les exemples suivants :

- [Bibliothèque de client Azure Service Bus pour JavaScript ↗](#)
- [Exemples JavaScript ↗](#)
- [Exemples TypeScript ↗](#)
- [Documentation de référence de l'API](#)

# Envoyer des messages à une rubrique Azure Service Bus et recevoir des messages à partir d'abonnements à la rubrique (JavaScript)

Article • 08/12/2023

Dans ce tutoriel, vous allez effectuer les étapes suivantes :

1. Créer un espace de noms Service Bus à l'aide du Portail Azure.
2. Créer une rubrique Service Bus à l'aide du Portail Azure.
3. Créer un abonnement Service Bus vers cette rubrique à l'aide du Portail Azure.
4. Écrivez une application JavaScript qui utilise le package [@azure/service-bus](#) pour effectuer les actions suivantes :
  - Envoyer un ensemble de messages à la rubrique.
  - Recevoir les messages de l'abonnement.

## ⓘ Notes

Ce guide de démarrage rapide fournit des instructions pas à pas pour un scénario simple qui consiste à envoyer un lot de messages à une rubrique Service Bus et à recevoir ces messages à partir d'un abonnement de la rubrique. Vous trouverez des exemples JavaScript et TypeScript prédéfinis pour Azure Service Bus dans le [dépôt du SDK Azure pour JavaScript sur GitHub](#).

## Prérequis

- Un abonnement Azure. Pour suivre ce tutoriel, vous avez besoin d'un compte Azure. Vous pouvez [activer les avantages de votre abonnement MSDN](#) ou [vous inscrire pour un compte gratuit](#).
- [Node.js LTS](#)
- Suivez les étapes dans [Démarrage rapide : Utiliser le portail Azure pour créer une rubrique Service Bus et des abonnements à cette rubrique](#). Vous n'utiliserez qu'un seul abonnement dans ce guide de démarrage rapide.

Sans mot de passe

Pour utiliser ce guide de démarrage rapide avec votre propre compte Azure, vous avez besoin des éléments suivants :

- Installez [Azure CLI](#), qui fournit l'authentification sans mot de passe à votre ordinateur de développement.
- Connectez-vous avec votre compte Azure au terminal ou à l'invite de commandes avec `az login`.
- Utilisez le même compte quand vous ajoutez le rôle approprié à votre ressource.
- Exécutez le code dans le même terminal ou la même invite de commande.
- Notez le nom de votre **rubrique** et **abonnement** pour votre espace de noms Service Bus. Vous en aurez besoin dans le code.

#### ⓘ Notes

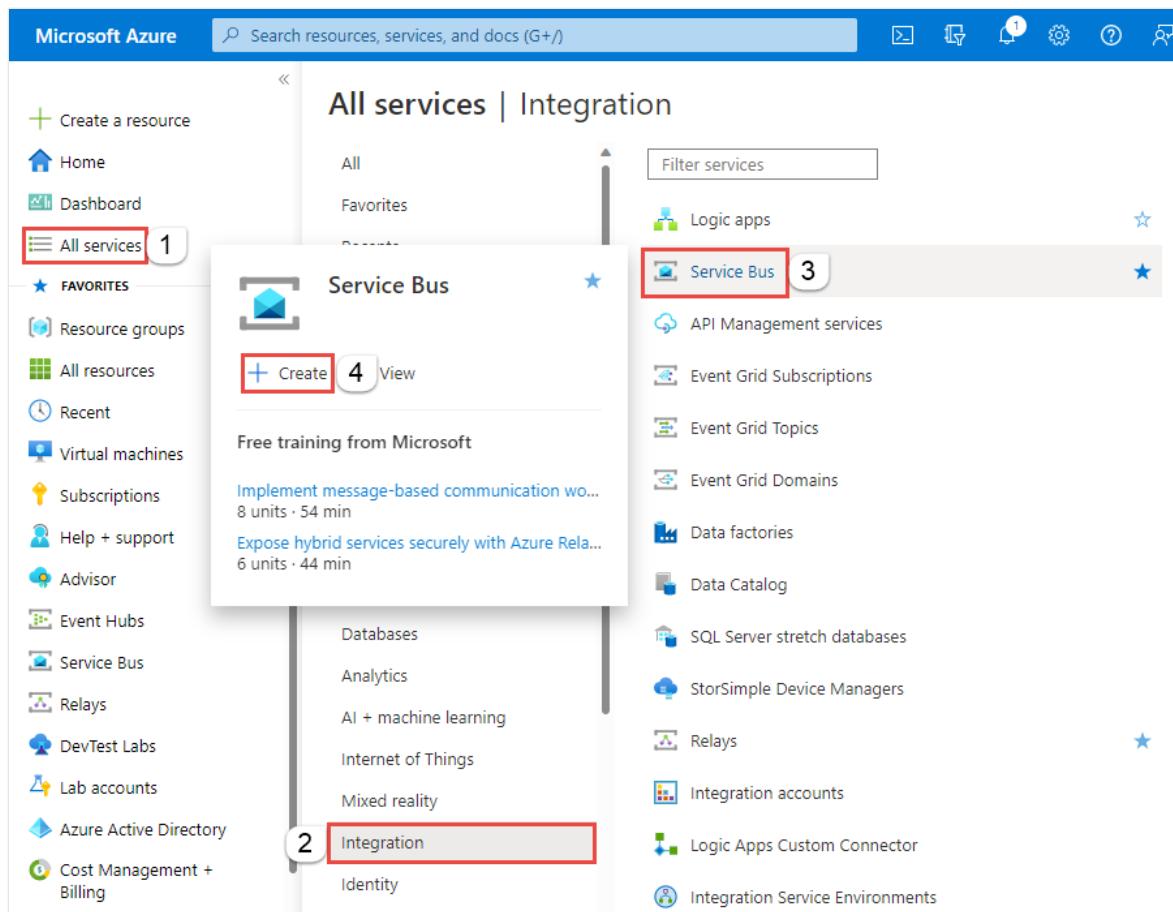
- Ce didacticiel utilise des exemples que vous pouvez copier et exécuter à l'aide de [Nodejs](#). Pour obtenir des instructions sur la création d'une application Node.js, veuillez consulter les pages [Création et déploiement d'une application Node.js sur un site web Azure](#) ou [Service cloud Node.js avec Windows PowerShell](#).

## Créer un espace de noms dans le Portail Azure

Pour commencer à utiliser des entités de messagerie Service Bus dans Azure, vous devez d'abord créer un espace de noms avec un nom unique dans Azure. Un espace de noms fournit un conteneur d'étendue pour les ressources du Service Bus (files d'attente, thèmes, etc.) au sein de votre application.

Pour créer un espace de noms :

1. Connectez-vous au [portail Azure](#).
2. Dans le volet de navigation gauche du portail, sélectionnez **Tous les services**, puis **Intégration** dans la liste des catégories, pointez la souris sur **Service Bus**, puis sélectionnez le bouton + sur la vignette Service Bus.



3. Dans l'étiquette **De base** de la page **Créer un espace de noms**, suivez ces étapes :

- Pour l'option **Abonnement**, choisissez un abonnement Azure dans lequel créer l'espace de noms.
- Pour l'option **Groupe de ressources**, choisissez un groupe de ressources existant dans lequel l'espace de noms sera utilisé, ou créez-en un nouveau.
- Entrez un **nom pour l'espace de noms**. Le nom de l'espace de noms doit respecter les conventions de nommage suivantes :
  - Le nom doit être unique dans tout Azure. Le système vérifie immédiatement si le nom est disponible.
  - Le nom doit inclure entre 6 et 50 caractères.
  - Le nom ne peut contenir que des lettres, des chiffres et des traits d'union (« - »).
  - Le nom doit commencer par une lettre, et se terminer par une lettre ou un chiffre.
  - Le nom ne se termine ni par « -sb » ni par « -mgmt ».
- Pour l'option **Emplacement**, choisissez la région dans laquelle héberger votre espace de noms.

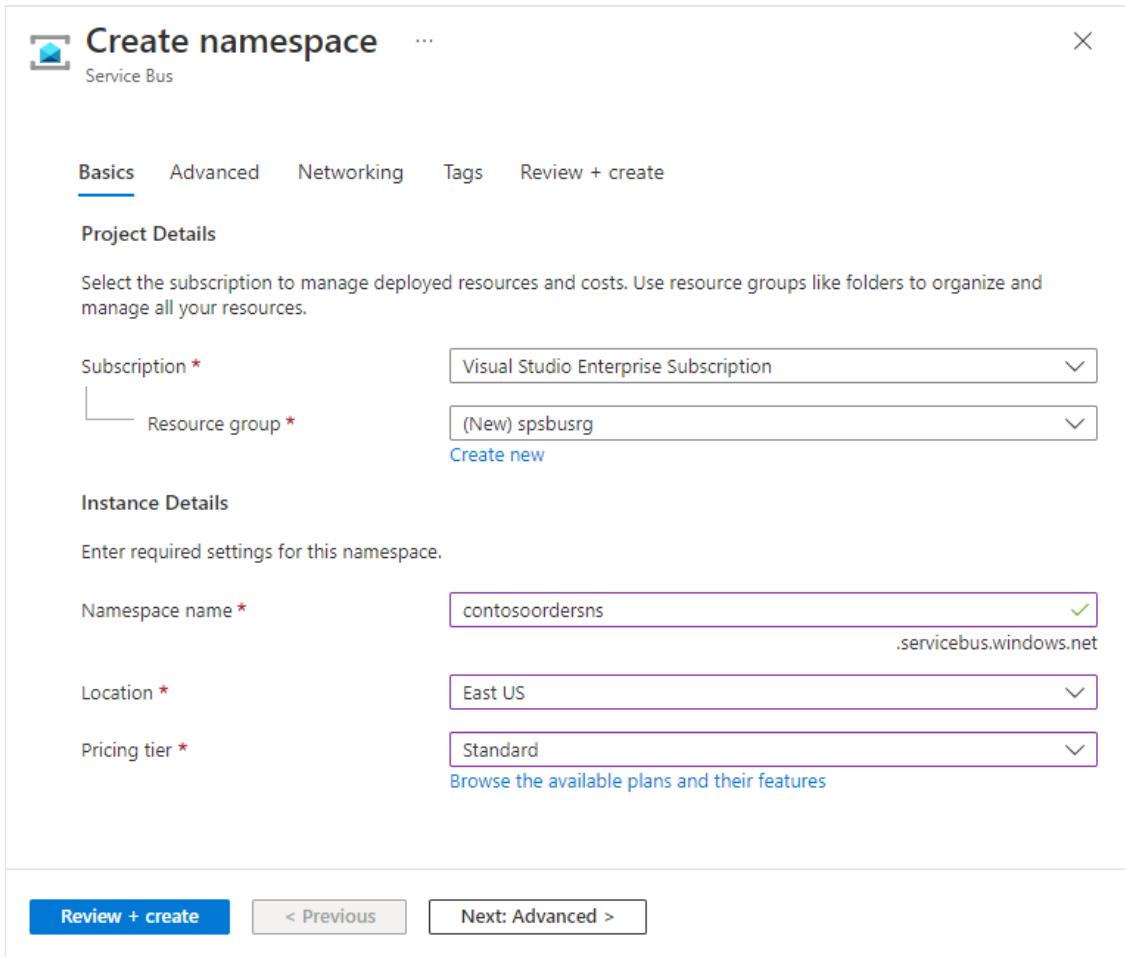
- e. Pour le **Niveau tarifaire**, sélectionnez le SKU (De base, Standard ou Premium) destiné à l'espace de noms. Pour ce guide de démarrage rapide, sélectionnez **Standard**.

 **Important**

Si vous voulez utiliser des **rubriques et des abonnements**, choisissez Standard ou Premium. Les rubriques/abonnements ne sont pas pris en charge dans le niveau tarifaire De base.

Si vous avez sélectionné le SKU **Premium**, précisez le nombre d'**unité de messagerie**. Le niveau Premium isole les ressources au niveau du processeur et de la mémoire, ce qui permet d'exécuter chaque charge de travail de manière isolée. Ce conteneur de ressources est appelé unité de messagerie. Un espace de noms Premium a au moins une unité de messagerie. Vous pouvez sélectionner 1, 2, 4, 8 ou 16 unités de messagerie pour chaque espace de noms Service Bus Premium. Pour plus d'informations, consultez [Messagerie Service Bus Premium](#).

- f. Au bas de la page, sélectionnez **Examiner et créer**.



The screenshot shows the 'Create namespace' wizard in the Azure portal. The title bar says 'Create namespace' and 'Service Bus'. The tabs at the top are 'Basics', 'Advanced', 'Networking', 'Tags', and 'Review + create'. The 'Basics' tab is selected. The 'Project Details' section contains fields for 'Subscription' (set to 'Visual Studio Enterprise Subscription') and 'Resource group' (set to '(New) spsbusrg'). Below that is the 'Instance Details' section, which includes 'Namespace name' (set to 'contosoordersns'), 'Location' (set to 'East US'), and 'Pricing tier' (set to 'Standard'). At the bottom of the screen are three buttons: 'Review + create' (highlighted in blue), '< Previous', and 'Next: Advanced >'.

g. Dans la page **Vérifier + créer**, passez en revue les paramètres, puis sélectionnez **Créer**.

4. Une fois le déploiement de la ressource réussi, sélectionnez **Accéder à la ressource** dans la page de déploiement.

The screenshot shows the 'Deployment' overview for a resource named 'contosoordersns'. A green checkmark indicates that the deployment is complete. The deployment details include the name, subscription, resource group, start time (10/20/2022, 4:45:03 PM), and correlation ID. There are sections for 'Deployment details' and 'Next steps', with a prominent red-bordered 'Go to resource' button. Below the main content, there are links to 'Give feedback' and 'Tell us about your experience with deployment'.

5. Vous voyez la page d'accueil de votre espace de noms Service Bus.

The screenshot shows the 'Overview' page for a Service Bus namespace named 'spsbusns1028'. The left sidebar lists various navigation options like 'Activity log', 'Access control (IAM)', 'Tags', etc. The main area displays 'Essentials' information such as resource group ('spsbusrg'), status ('Active'), location ('East US'), and subscription ('Visual Studio Enterprise Subscription'). It also shows metrics for 'Requests' and 'Messages' over the last hour. The 'JSON View' link is visible in the top right corner.

## Créer une rubrique à l'aide du Portail Azure

1. Dans la page **Espace de noms Service Bus**, sélectionnez **Rubriques** dans le menu de gauche.
2. Sélectionnez **+ Rubrique** dans la barre d'outils.

3. Entrez un **nom** pour la rubrique. Conservez les valeurs par défaut des autres options.

4. Cliquez sur **Créer**.

The screenshot shows the Azure portal interface for creating a new topic. On the left, the navigation pane is visible with sections like 'Settings', 'Entities' (Queues and Topics selected), and 'Monitoring'. The main area shows a list of topics with one entry: 'mytopic'. A red box highlights the '+ Topic' button at the top right of the list. A modal window titled 'Create topic' is open on the right. It contains fields for 'Name' (set to 'mytopic'), 'Max topic size' (set to '1 GB'), and 'Message time to live' (set to 'Days: 14, Hours: 0, Minutes: 0, Seconds: 0'). There are also three unchecked checkboxes for 'Enable auto-delete on idle topic', 'Enable duplicate detection', and 'Enable partitioning'. A red box highlights the 'Create' button at the bottom right of the modal.

## Créer un abonnement à la rubrique

1. Sélectionnez la **rubrique** que vous avez créée dans la section précédente.

The screenshot shows the same Azure portal interface as the previous step, but now the 'Topics' list is populated with one item: 'mytopic'. A red box highlights the 'mytopic' entry in the list. The rest of the interface is identical to the previous screenshot, showing the navigation pane and the 'Topics' creation dialog still open on the right.

2. Dans la page **Rubrique Service Bus**, dans la barre d'outils, sélectionnez + **Abonnement**.

The screenshot shows the Azure portal interface for a Service Bus Topic named 'mytopic'. The top navigation bar includes 'Subscriptions', a star icon, and three dots. On the left, there's a sidebar with links like 'Overview', 'Access control (IAM)', 'Diagnose and solve problems', 'Service Bus Explorer', 'Settings' (with 'Shared access policies', 'Properties', and 'Locks'), 'Entities' (with 'Subscriptions' highlighted by a red box), 'Automation' (with 'Tasks (preview)' and 'Export template'), and 'Support + troubleshooting' (with 'New Support Request'). The main content area has a search bar and a table with columns 'Name', 'Status', 'Message count', 'Max delivery count', and 'Client Scoped'. A message states 'No results.' Below the table is a horizontal scrollbar.

3. Dans la page **Créer un abonnement**, procédez comme suit :

- a. Entrez **S1** pour le **nom** de l'abonnement.
- b. Entrez **3** pour **Nombre maximal de remises**.
- c. Ensuite, sélectionnez **Créer** pour créer l'abonnement.

## Create subscription

X

Service Bus

Name \* ⓘ

S1



Max delivery count \* ⓘ

3



Auto-delete after idle for ⓘ

Days

Hours

Minutes

Seconds

14

0

0

0

Never auto-delete

Forward messages to queue/topic ⓘ

### MESSAGE SESSIONS

Service bus sessions allow ordered handling of unbounded sequences of related messages. With sessions enabled a subscription can guarantee first-in-first-out delivery of messages. [Learn more.](#)

Enable sessions

### MESSAGE TIME TO LIVE AND DEAD-LETTERING

Message time to live (default) ⓘ

Days

Hours

Minutes

Seconds

14

0

0

0

Enable dead lettering on message expiration

Move messages that cause filter evaluation exceptions to the dead-letter subqueue

### MESSAGE LOCK DURATION

Lock duration ⓘ

Days

Hours

Minutes

Seconds

0

0

0

30

**Create**

## Authentifier l'application sur Azure

Ce guide de démarrage pratique vous montre deux façons de vous connecter à Azure Service Bus : **sans mot de passe** et avec une **chaîne de connexion**.

La première option vous explique comment utiliser votre principal de sécurité dans Microsoft Entra ID et le contrôle d'accès en fonction du rôle (RBAC) pour vous connecter à un espace de noms Service Bus. Vous n'avez pas à vous soucier d'avoir une chaîne de connexion codée en dur dans votre code, dans un fichier config ni dans un stockage sécurisé comme Azure Key Vault.

La deuxième option consiste à se servir d'une chaîne de connexion pour se connecter à un espace de noms Service Bus. Si vous débutez avec Azure, vous trouverez peut-être l'option chaîne de connexion plus facile à suivre. Nous vous recommandons d'utiliser l'option sans mot de passe dans les applications réelles et les environnements de production. Pour plus d'informations, consultez [Authentification et autorisation](#). Pour en savoir plus sur l'authentification sans mot de passe, reportez-vous à la [page de présentation](#).

Sans mot de passe (recommandé)

## Attribuer des rôles à votre utilisateur Microsoft Entra

Lors du développement localement, assurez-vous que le compte d'utilisateur qui se connecte à Azure Service Bus dispose des autorisations appropriées. Vous aurez besoin du rôle [Propriétaire de données Azure Service Bus](#) pour envoyer et recevoir des messages. Pour vous attribuer ce rôle, vous aurez besoin du rôle Administrateur de l'accès utilisateur ou d'un autre rôle qui inclut l'action

`Microsoft.Authorization/roleAssignments/write`. Vous pouvez attribuer des rôles RBAC Azure à un utilisateur à l'aide du Portail Azure, Azure CLI ou Azure PowerShell. Découvrez les étendues disponibles pour les attributions de rôles dans la page [vue d'ensemble de l'étendue](#).

L'exemple suivant attribue le rôle `Azure Service Bus Data Owner` à votre compte d'utilisateur, qui fournit un accès complet aux ressources Azure Service Bus. Dans un scénario réel, suivez le [principe des privilèges minimum](#) pour accorder aux utilisateurs uniquement les autorisations minimales nécessaires à un environnement de production plus sécurisé.

## Rôles Azure intégrés pour Azure Service Bus

Pour Azure Service Bus, la gestion des espaces de noms et de toutes les ressources associées via le Portail Azure et l'API de gestion des ressources Azure est déjà protégée à l'aide du modèle Azure RBAC. Azure fournit les rôles Azure intégrés ci-dessous pour autoriser l'accès à un espace de noms Service Bus :

- [Propriétaire de données Azure Service Bus](#) : ce rôle permet l'accès aux données de l'espace de noms Service Bus et de ses entités (files d'attente, rubriques, abonnements et filtres). Un membre de ce rôle peut envoyer et recevoir des messages à partir de files d'attente ou de rubriques et d'abonnements.

- [Expéditeur de données Azure Service Bus](#) : utilisez ce rôle pour autoriser l'accès en envoi à l'espace de noms Service Bus et à ses entités.
- [Récepteur de données Azure Service Bus](#) : utilisez ce rôle pour autoriser l'accès en réception à l'espace de noms Service Bus et à ses entités.

Si vous souhaitez créer un rôle personnalisé, consultez [Droits requis pour les opérations Service Bus](#).

## Ajouter un utilisateur Microsoft Entra au rôle Propriétaire Azure Service Bus

Ajoutez votre nom d'utilisateur Microsoft Entra au rôle **Propriétaire de données Azure Service Bus** au niveau de l'espace de noms Service Bus. Il permet à une application exécutée dans le contexte de votre compte d'utilisateur d'envoyer des messages à une file d'attente ou à une rubrique et d'en recevoir auprès d'une file d'attente ou de l'abonnement d'une rubrique.

### Important

Dans la plupart des cas, la propagation de l'attribution de rôle dans Azure peut prendre une ou deux minutes. Dans de rares cas, cela peut prendre jusqu'à **huit minutes**. Si vous recevez des erreurs d'authentification lorsque vous exécutez votre code pour la première fois, patientez quelques instants et réessayez.

1. Si la page Espace de noms Service Bus n'est pas ouverte sur le Portail Azure, recherchez votre espace de noms Service Bus à l'aide de la barre de recherche principale ou du volet de navigation de gauche.
2. Dans la page vue d'ensemble, sélectionnez **Contrôle d'accès (IAM)** dans le menu de gauche.
3. Sur la page **Contrôle d'accès (IAM)**, sélectionnez l'onglet **Attributions de rôles**.
4. Sélectionnez **+ Ajouter** dans le menu supérieur, puis **Ajouter une attribution de rôle** dans le menu déroulant résultant.

5. Utilisez la zone de recherche pour filtrer les résultats sur le rôle souhaité. Pour cet exemple, recherchez **Azure Service Bus Data Owner** et sélectionnez le résultat correspondant. Ensuite, choisissez **Suivant**.
6. Sous **Attribuer l'accès à**, sélectionnez **Utilisateur, groupe ou principal de service**, puis sélectionnez + **Sélectionner des membres**.
7. Dans la boîte de dialogue, recherchez votre nom d'utilisateur Microsoft Entra (généralement votre adresse e-mail *user@domain*), puis choisissez **Sélectionner** en bas de la boîte de dialogue.
8. Sélectionnez **Vérifier + affecter** pour accéder à la page finale, puis **Vérifier + attribuer** à nouveau pour terminer le processus.

## Utilisation de Node Package Manager (NPM) pour installer le package

Sans mot de passe

1. Pour installer le ou les packages npm requis pour Service Bus, ouvrez une invite de commandes dont le chemin contient `npm`, remplacez le répertoire par le dossier où vous souhaitez stocker vos exemples, puis exécutez cette commande.

2. Installez les packages suivants :

Bash

```
npm install @azure/service-bus @azure/identity
```

## Envoi de messages à une rubrique

L'exemple de code suivant montre comment envoyer un lot de messages à une rubrique Service Bus. Pour plus d'informations, consultez les commentaires du code.

Sans mot de passe

Vous devez avoir ouvert une session avec `az login` dans Azure CLI pour que votre ordinateur local fournisse l'authentification sans mot de passe requise dans ce code.

1. Ouvrez votre éditeur favori, tel que [Visual Studio Code](#)
2. Créez un fichier appelé `sendtotoTopic.js` et collez-y le code ci-dessous. Ce code envoie un message vers votre rubrique.

### Important

Les informations d'identification sans mot de passe sont fournies avec [DefaultAzureCredential](#).

JavaScript

```
const { ServiceBusClient } = require("@azure/service-bus");
const { DefaultAzureCredential } = require("@azure/identity");

// Replace `<SERVICE-BUS-NAMESPACE>` with your namespace
const fullyQualifiedNamespace = "<SERVICE-BUS-
NAMESPACE>.servicebus.windows.net";

// Passwordless credential
const credential = new DefaultAzureCredential();

const topicName = "<TOPIC NAME>";

const messages = [
 { body: "Albert Einstein" },
 { body: "Werner Heisenberg" },
 { body: "Marie Curie" },
```

```
{ body: "Steven Hawking" },
{ body: "Isaac Newton" },
{ body: "Niels Bohr" },
{ body: "Michael Faraday" },
{ body: "Galileo Galilei" },
{ body: "Johannes Kepler" },
{ body: "Nikolaus Kopernikus" }
];

async function main() {
 // create a Service Bus client using the passwordless
 authentication to the Service Bus namespace
 const sbClient = new ServiceBusClient(fullyQualifiedNamespace,
 credential);

 // createSender() can also be used to create a sender for a
 queue.
 const sender = sbClient.createSender(topicName);

 try {
 // Tries to send all messages in a single batch.
 // Will fail if the messages cannot fit in a batch.
 // await sender.sendMessages(messages);

 // create a batch object
 let batch = await sender.createMessageBatch();
 for (let i = 0; i < messages.length; i++) {
 // for each message in the array

 // try to add the message to the batch
 if (!batch.tryAddMessage(messages[i])) {
 // if it fails to add the message to the current
batch
 // send the current batch as it is full
 await sender.sendMessages(batch);

 // then, create a new batch
 batch = await sender.createMessageBatch();

 // now, add the message failed to be added to the
previous batch to this batch
 if (!batch.tryAddMessage(messages[i])) {
 // if it still can't be added to the batch, the
message is probably too big to fit in a batch
 throw new Error("Message too big to fit in a
batch");
 }
 }
 }

 // Send the last created batch of messages to the topic
 await sender.sendMessages(batch);

 console.log(`Sent a batch of messages to the topic:
${topicName}`);
 }
}
```

```
// Close the sender
 await sender.close();
} finally {
 await sbClient.close();
}
}

// call the main function
main().catch((err) => {
 console.log("Error occurred: ", err);
 process.exit(1);
});
```

3. Remplacez <SERVICE BUS NAMESPACE CONNECTION STRING> par la chaîne de connexion de votre espace de noms Service Bus,
4. et remplacez <TOPIC NAME> par le nom de la rubrique.
5. Ensuite, exécutez la commande à partir d'une invite de commandes pour exécuter ce fichier.

```
Console
node sendtotoTopic.js
```

6. Vous devez voir la sortie suivante.

```
Console
Sent a batch of messages to the topic: mytopic
```

## Réception des messages d'un abonnement

Sans mot de passe

Vous devez avoir ouvert une session avec `az login` dans Azure CLI pour que votre ordinateur local fournisse l'authentification sans mot de passe requise dans ce code.

1. Ouvrez votre éditeur favori, tel que [Visual Studio Code](#)
2. Créez un fichier nommé `receivefromsubscription.js` et collez-y le code suivant.  
Pour plus d'informations, consultez les commentaires du code.

## JavaScript

```
const { delay, ServiceBusClient, ServiceBusMessage } =
require("@azure/service-bus");
const { DefaultAzureCredential } = require("@azure/identity");

// Replace `<SERVICE-BUS-NAMESPACE>` with your namespace
const fullyQualifiedNamespace = "<SERVICE-BUS-
NAMESPACE>.servicebus.windows.net";

// Passwordless credential
const credential = new DefaultAzureCredential();

const topicName = "<TOPIC NAME>";
const subscriptionName = "<SUBSCRIPTION NAME>";

async function main() {
 // create a Service Bus client using the passwordless
 authentication to the Service Bus namespace
 const sbClient = new ServiceBusClient(fullyQualifiedNamespace,
 credential);

 // createReceiver() can also be used to create a receiver for a
 queue.
 const receiver = sbClient.createReceiver(topicName,
 subscriptionName);

 // function to handle messages
 const myMessageHandler = async (messageReceived) => {
 console.log(`Received message: ${messageReceived.body}`);
 };

 // function to handle any errors
 const myErrorHandler = async (error) => {
 console.log(error);
 };

 // subscribe and specify the message and error handlers
 receiver.subscribe({
 processMessage: myMessageHandler,
 processError: myErrorHandler
 });

 // Waiting long enough before closing the sender to send
 messages
 await delay(5000);

 await receiver.close();
 await sbClient.close();
}

// call the main function
main().catch((err) => {
 console.log("Error occurred: ", err);
```

```
 process.exit(1);
});
```

3. Remplacez <SERVICE BUS NAMESPACE CONNECTION STRING> par la chaîne de connexion à l'espace de noms,
4. et remplacez <TOPIC NAME> par le nom de la rubrique.
5. Remplacez <SUBSCRIPTION NAME> par le nom de l'abonnement à la rubrique.
6. Ensuite, exécutez la commande à partir d'une invite de commandes pour exécuter ce fichier.

Console

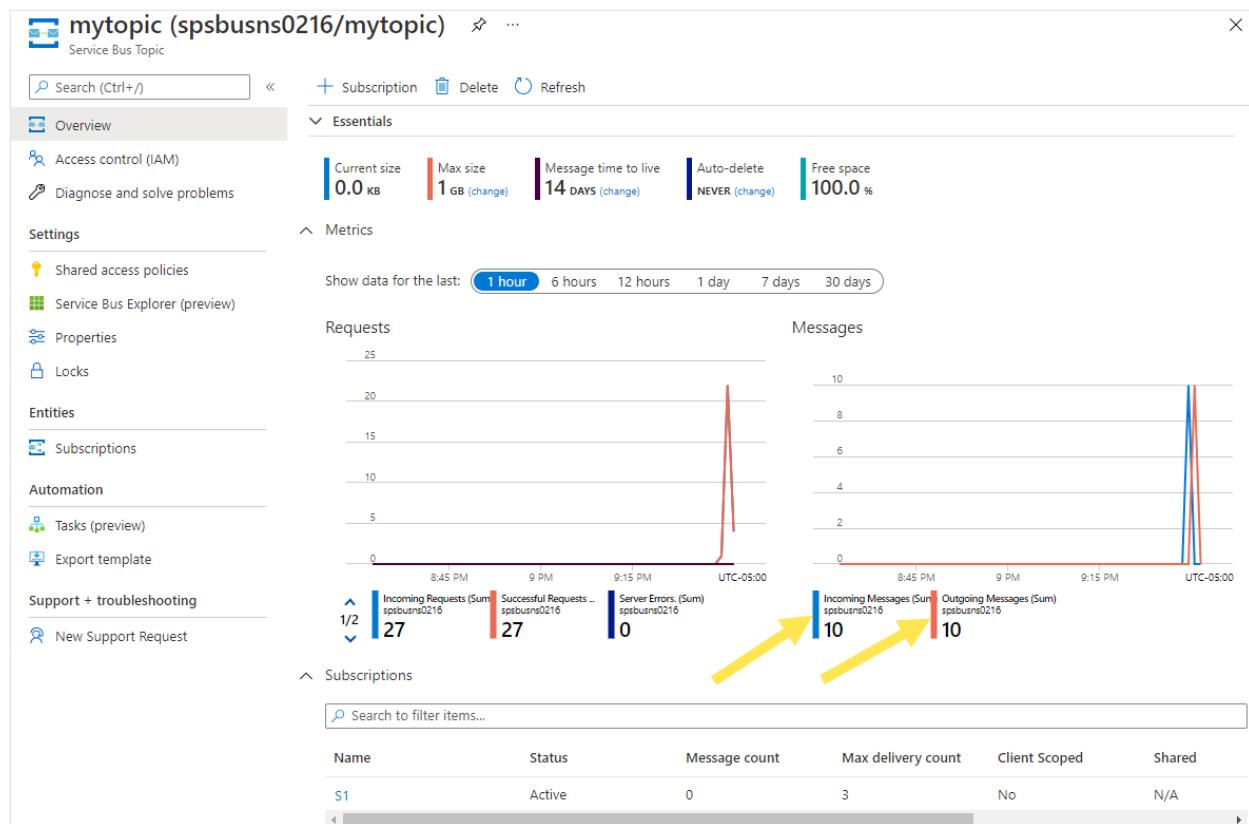
```
node receivefromsubscription.js
```

Vous devez voir la sortie suivante.

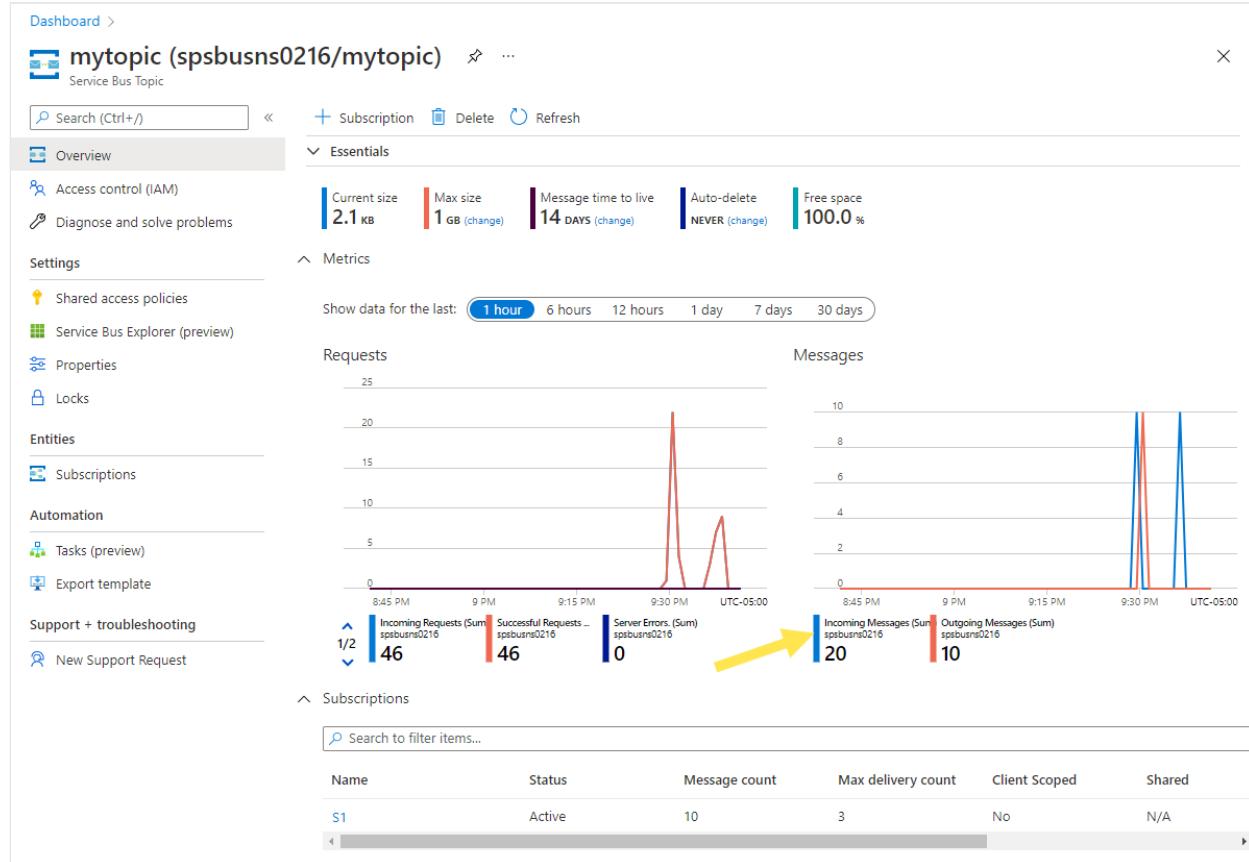
Console

```
Received message: Albert Einstein
Received message: Werner Heisenberg
Received message: Marie Curie
Received message: Steven Hawking
Received message: Isaac Newton
Received message: Niels Bohr
Received message: Michael Faraday
Received message: Galileo Galilei
Received message: Johannes Kepler
Received message: Nikolaus Kopernikus
```

Dans le portail Azure, accédez à votre espace de noms Service Bus, basculez vers **Rubriques** dans le volet du bas et sélectionnez votre rubrique pour voir la page **Rubrique Service Bus** correspondant à votre rubrique. Dans cette page, vous devez voir 10 messages entrants et 10 messages sortants dans le graphique **Messages**.



Si la fois suivante vous exécutez uniquement l'application d'envoi, dans la page Rubrique Service Bus, vous voyez 20 messages entrants (10 nouveaux), mais 10 messages sortants.



Dans cette page, si vous sélectionnez un abonnement dans le volet du bas, vous accédez à la page Abonnement Service Bus. Elle indique entre autres le nombre de

messages actifs et le nombre de messages de lettres mortes. Dans cet exemple, il y a 10 messages actifs qui n'ont pas encore été reçus par un récepteur.

The screenshot shows the Azure Service Bus Subscription settings for a subscription named 'S1'. It displays various configuration parameters and metrics. A yellow arrow points to the 'Active message count' which is currently set to 10 messages. Below the main settings, there is a 'FILTERS' section with a table:

Name	Filter Type
\$Default	SqlFilter

## Résolution des problèmes

Si une erreur survient quand vous exécutez la version **sans mot de passe** du code JavaScript concernant les revendications requises, vérifiez que vous vous êtes connecté avec la commande `az login` d'Azure CLI et que le **rôle approprié** est appliqué à votre compte d'utilisateur Azure.

## Nettoyer les ressources

Accédez à votre espace de noms Service Bus dans le Portail Azure, puis sélectionnez **Supprimer** sur le Portail Azure pour supprimer l'espace de noms et la file d'attente qu'il contient.

## Étapes suivantes

Voir la documentation et les exemples suivants :

- [Bibliothèque de client Azure Service Bus pour JavaScript ↗](#)
- [Exemples JavaScript](#)
- [Exemples TypeScript](#)
- [Documentation de référence de l'API](#)

# Démarrage rapide : Bibliothèque de client Stockage Blob Azure pour Node.js

Article • 25/03/2023

Commencez à utiliser la bibliothèque de client Stockage Blob Azure pour Node.js pour gérer les objets blob et les conteneurs. Suivez les étapes suivantes pour installer le package et essayer un exemple de code pour les tâches de base.

[Référence API](#) | [Code source de la bibliothèque](#) | [Package \(npm\)](#) | [Exemples](#)

## Prérequis

- Compte Azure avec un abonnement actif. [Créez un compte gratuitement](#).
- Un compte de stockage Azure. [Créer un compte de stockage](#).
- [Node.js LTS](#).

## Créer le projet Node.js

Créez une application JavaScript nommée *blob-quickstart*.

1. Dans une fenêtre de console (telle que cmd, PowerShell ou Bash), créez un nouveau répertoire pour le projet.

```
Console
mkdir blob-quickstart
```

2. Basculez vers le répertoire *blob-quickstart* nouvellement créé.

```
Console
cd blob-quickstart
```

3. Créez un *package.json*.

```
Console
npm init -y
```

4. Ouvrez le projet dans Visual Studio Code :

Console

```
code .
```

## Installer les packages

À partir du répertoire du projet, installez les packages suivants à l'aide de la commande `npm install`.

1. Installer le package npm Stockage Azure :

Console

```
npm install @azure/storage-blob
```

2. Installez le package npm Azure Identity pour une connexion sans mot de passe :

Console

```
npm install @azure/identity
```

3. Installer d'autres dépendances utilisées dans ce démarrage rapide :

Console

```
npm install uuid dotenv
```

## Créer des fichiers JavaScript

À partir du répertoire de projet :

1. Créez un nouveau fichier appelé `index.js`.
2. Copiez le code suivant dans le fichier. Davantage de code est ajouté au fur et à mesure que vous consultez ce démarrage rapide.

JavaScript

```
const { BlobServiceClient } = require("@azure/storage-blob");
const { v1: uuidv1 } = require("uuid");
require("dotenv").config();

async function main() {
```

```

try {
 console.log("Azure Blob storage v12 - JavaScript quickstart
sample");

 // Quick start code goes here

} catch (err) {
 console.error(`Error: ${err.message}`);
}
}

main()
.then(() => console.log("Done"))
.catch((ex) => console.log(ex.message));

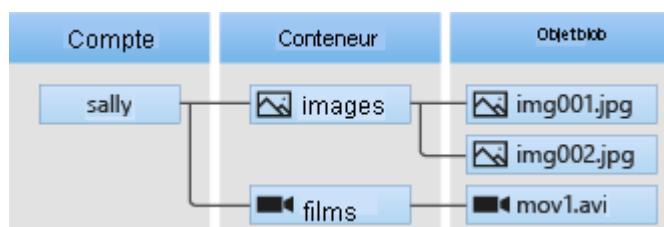
```

## Modèle objet

Le Stockage Blob Azure est optimisé pour stocker de grandes quantités de données non structurées. Les données non structurées sont des données qui n'obéissent pas à un modèle ou une définition de données en particulier, comme des données texte ou binaires. Le stockage Blob offre trois types de ressources :

- Le compte de stockage
- Un conteneur dans le compte de stockage.
- Un blob dans le conteneur

Le diagramme suivant montre la relation entre ces ressources.



Utilisez les classes JavaScript suivantes pour interagir avec ces ressources :

- **BlobServiceClient**: La classe `BlobServiceClient` vous permet de manipuler les ressources de stockage Azure et les conteneurs blob.
- **ContainerClient** : La classe `ContainerClient` vous permet de manipuler des conteneurs de stockage Azure et leurs blobs.
- **BlobClient** : La classe `BlobClient` vous permet de manipuler des blobs de stockage Azure.

# Exemples de code

Ces exemples d'extraits de code vous montrent comment effectuer les tâches suivantes avec la bibliothèque cliente Stockage Blob Azure pour JavaScript :

- [S'authentifier auprès d'Azure et autoriser l'accès aux données blob](#)
- [Créer un conteneur](#)
- [Charger des objets blob sur un conteneur](#)
- [Lister les objets blob d'un conteneur](#)
- [Télécharger des objets blob](#)
- [Supprimer un conteneur](#)

Un exemple de code est également disponible sur [GitHub](#).

## S'authentifier auprès d'Azure et autoriser l'accès aux données blob

Les demandes d'application vers le Stockage Blob Azure doivent être autorisées.

L'utilisation de la classe `DefaultAzureCredential` fournie par la bibliothèque de client Azure Identity est l'approche recommandée pour implémenter des connexions sans mot de passe aux services Azure dans votre code, y compris le Stockage Blob.

Vous pouvez également autoriser les demandes vers le Stockage Blob Azure à l'aide de la clé d'accès au compte. Toutefois, cette approche doit être utilisée avec prudence. Les développeurs doivent être vigilants pour ne jamais exposer la clé d'accès dans un emplacement non sécurisé. Toute personne disposant de la clé d'accès est en mesure d'autoriser les demandes sur le compte de stockage et a accès efficacement à toutes les données. `DefaultAzureCredential` offre des avantages améliorés en matière de gestion et de sécurité par rapport à la clé de compte pour autoriser l'authentification sans mot de passe. Les deux options sont illustrées dans l'exemple suivant.

Sans mot de passe (recommandé)

`DefaultAzureCredential` prend en charge plusieurs méthodes d'authentification et détermine quelle méthode doit être utilisée au moment de l'exécution. Cette approche permet à votre application d'utiliser différentes méthodes d'authentification dans différents environnements (local ou production) sans implémenter de code spécifique à l'environnement.

L'ordre et les emplacements dans lesquels `DefaultAzureCredential` les informations d'identification sont disponibles dans la [vue d'ensemble de la bibliothèque](#)

d'identités Azure.

Par exemple, votre application peut s'authentifier à l'aide de vos informations d'identification de connexion Azure CLI lors du développement local. Votre application peut ensuite utiliser une [identité managée](#) une fois qu'elle a été déployée sur Azure. Aucune modification du code n'est requise pour cette transition.

## Attribuer des rôles à votre compte d'utilisateur Microsoft Entra

Lors du développement local, assurez-vous que le compte d'utilisateur qui accède aux données blob dispose des autorisations appropriées. Vous aurez besoin du **Contributeur aux données Blob de stockage** pour lire et écrire des données blob. Pour vous attribuer ce rôle, vous aurez besoin du rôle **Administrateur de l'accès utilisateur** ou d'un autre rôle qui inclut l'action [Microsoft.Authorization/roleAssignments/write](#). Vous pouvez attribuer des rôles RBAC Azure à un utilisateur à l'aide du Portail Azure, Azure CLI ou Azure PowerShell. Vous pouvez en savoir plus sur les étendues disponibles pour les attributions de rôles dans la page [vue d'ensemble de l'étendue](#).

Dans ce scénario, vous allez attribuer des autorisations à votre compte d'utilisateur, étendues au compte de stockage, pour suivre le [Principe des priviléges minimum](#). Cette pratique offre aux utilisateurs uniquement les autorisations minimales nécessaires et crée des environnements de production plus sécurisés.

L'exemple suivant affecte le rôle **Contributeur aux données Blob du stockage** à votre compte d'utilisateur, qui fournit à la fois un accès en lecture et en écriture aux données d'objet blob dans votre compte de stockage.

### ⓘ Important

Dans la plupart des cas, la propagation de l'attribution de rôle dans Azure peut prendre une ou deux minutes, mais dans de rares cas, cela peut prendre jusqu'à huit minutes. Si vous recevez des erreurs d'authentification lorsque vous exécutez votre code pour la première fois, patientez quelques instants et réessayez.

Azure portal

1. Dans le Portail Azure, recherchez votre compte de stockage à l'aide de la barre de recherche principale ou de la navigation gauche.
2. Dans la page vue d'ensemble du compte de stockage, sélectionnez **Contrôle d'accès (IAM)** dans le menu de gauche.
3. Sur la page **Contrôle d'accès (IAM)**, sélectionnez l'onglet **Attributions de rôles**.
4. Sélectionnez **+ Ajouter** dans le menu supérieur, puis **Ajouter une attribution de rôle** dans le menu déroulant résultant.

The screenshot shows the 'Access Control (IAM)' blade for a storage account named 'identitymigrationstorage'. The left sidebar lists various management options like Data migration, Events, Storage browser, etc. The 'Access Control (IAM)' option is selected and highlighted with a red box. At the top, there's a search bar and several action buttons: '+ Add' (highlighted with a red box), 'Download role assignments', 'Edit columns', 'Refresh', 'Remove', and 'Got feedback?'. Below these buttons is a dropdown menu with 'Add role assignment' (also highlighted with a red box) and other options like 'Add co-administrator'. The main content area has sections for 'My access', 'Check access', and 'Grant access to this resource'. Under 'Check access', there's a 'Find' input field with radio buttons for 'User, group, or service principal' (selected) and 'Managed identity'. A search bar below it says 'Search by name or email address'. On the right side, there are two boxes: 'Grant access to this resource' with a 'Add role assignment' button and 'Learn more' link, and 'View deny assignments' with a 'View' button and 'Learn more' link.

5. Utilisez la zone de recherche pour filtrer les résultats sur le rôle souhaité. Pour cet exemple, recherchez *Contributeur aux données Blob du stockage*, sélectionnez le résultat correspondant, puis choisissez **Suivant**.
6. Sous **Attribuer l'accès à**, sélectionnez **Utilisateur, groupe ou principal de service**, puis sélectionnez **+ Sélectionner des membres**.
7. Dans la boîte de dialogue, recherchez votre nom d'utilisateur Microsoft Entra (généralement votre adresse e-mail *user@domain*), puis choisissez **Sélectionner** en bas de la boîte de dialogue.
8. Sélectionnez **Vérifier + affecter** pour accéder à la page finale, puis **Vérifier + attribuer** à nouveau pour terminer le processus.

**Se connecter et connecter le code d'application à Azure à l'aide de DefaultAzureCredential**

Vous pouvez autoriser l'accès aux données dans votre compte de stockage en procédant comme suit :

1. Vérifiez que vous êtes authentifié avec le même compte Microsoft Entra auquel vous avez attribué le rôle sur votre compte de stockage. Vous pouvez vous authentifier via Azure CLI, Visual Studio Code ou Azure PowerShell.

Azure CLI

Connectez-vous à Azure via Azure CLI à l'aide de la commande suivante :

Azure CLI

```
az login
```

2. Pour utiliser `DefaultAzureCredential`, assurez-vous que le package `@azure\identity` est installé et à ce que la classe est importée :

JavaScript

```
const { DefaultAzureCredential } = require('@azure/identity');
```

3. Ajoutez ce code dans le bloc `try`. Quand le code est exécuté sur votre station de travail locale, `DefaultAzureCredential` utilise les informations d'identification de développeur de l'outil prioritaire auquel vous êtes connecté pour l'authentification auprès d'Azure, par exemple Azure CLI ou Visual Studio Code.

JavaScript

```
const accountName = process.env.AZURE_STORAGE_ACCOUNT_NAME;
if (!accountName) throw Error('Azure Storage accountName not found');

const blobServiceClient = new BlobServiceClient(
 `https://${accountName}.blob.core.windows.net`,
 new DefaultAzureCredential()
);
```

4. Assurez-vous de mettre à jour le nom du compte de stockage, `AZURE_STORAGE_ACCOUNT_NAME`, dans le fichier `.env` ou vos variables d'environnement. Le nom du compte de stockage se trouve sur la page vue d'ensemble du Portail Azure.

identitymigrationstorage

Storage account

Search (Ctrl+ /)

Upload Open in Explorer Delete Move Refresh

Overview

Activity log

Tags

Diagnose and solve problems

Access Control (IAM)

Data migration

Events

Storage browser

Resource group (move) : alexw-identity-revamp

Location : East US

Primary/Secondary Location : Primary: East US, Secondary: West US

Subscription (move) : C&L Cross Service Content Team Testing

Subscription ID :

Disk state : Primary: Available, Secondary: Available

Tags (edit) : Click here to add tags

### ⓘ Notes

Lorsqu'il est déployé sur Azure, ce même code peut être utilisé pour autoriser les demandes adressées à Stockage Azure à partir d'une application s'exécutant dans Azure. Toutefois, vous devez activer l'identité managée sur votre application dans Azure. Configurez ensuite votre compte de stockage pour autoriser cette identité managée à se connecter. Pour obtenir des instructions détaillées sur la configuration de cette connexion entre les services Azure, consultez le didacticiel [d'authentification à partir d'applications hébergées sur Azure](#).

## Créez un conteneur.

1. Choisissez un nom pour le nouveau conteneur. Les noms de conteneurs doivent être en minuscules.

Pour plus d'informations sur l'affectation de noms aux conteneurs et objets blob, consultez [Affectation de noms et références aux conteneurs, objets blob et métadonnées](#).

2. Ajoutez ce code à la fin de la fonction `main` :

JavaScript

```
// Create a unique name for the container
const containerName = 'quickstart' + uuidv1();

console.log('\nCreating container...');
console.log('\t', containerName);
```

```
// Get a reference to a container
const containerClient =
blobServiceClient.getContainerClient(containerName);
// Create the container
const createContainerResponse = await containerClient.create();
console.log(
`Container was created
successfully.\n\trequestId:${createContainerResponse.requestId}\n\tURL:
${containerClient.url}
`);
```

Le code précédent prend un objet `BlobServiceClient` et appelle la méthode `getContainerClient` afin d'obtenir une référence à un conteneur. Enfin, le code appelle `créer` pour effectivement créer le conteneur dans votre compte de stockage.

Pour en savoir plus sur la création d'un conteneur et explorer d'autres exemples de code, consultez [Créer un conteneur d'objets blob avec JavaScript](#).

## Charger des objets blob sur un conteneur

Copiez le code suivant à la fin de la fonction `main` pour charger une chaîne de texte dans un objet blob :

JavaScript

```
// Create a unique name for the blob
const blobName = 'quickstart' + uuidv1() + '.txt';

// Get a block blob client
const blockBlobClient = containerClient.getBlockBlobClient(blobName);

// Display blob name and url
console.log(
`\nUploading to Azure storage as blob\n\tname: ${blobName}:\n\tURL:
${blockBlobClient.url}`
);

// Upload data to the blob
const data = 'Hello, World!';
const uploadBlobResponse = await blockBlobClient.upload(data, data.length);
console.log(
`Blob was uploaded successfully. requestId:
${uploadBlobResponse.requestId}`
);
```

Le code précédent obtient une référence à un objet `BlockBlobClient` en appelant la méthode `getBlockBlobClient` sur le `ContainerClient` dans la section [Créer un conteneur](#). Le code charge les données de chaîne de texte dans l'objet blob en appelant la méthode `upload_blob`.

Pour en savoir plus sur le chargement d'objets blob et explorer d'autres exemples de code, consultez [Charger un objet blob avec JavaScript](#).

## Créer la liste des objets blob d'un conteneur

Ajoutez le code suivant à la fin de la fonction `main` pour répertorier les objets blob dans le conteneur.

JavaScript

```
console.log('\nListing blobs...');

// List the blob(s) in the container.
for await (const blob of containerClient.listBlobsFlat()) {
 // Get Blob Client from name, to get the URL
 const tempBlockBlobClient = containerClient.getBlockBlobClient(blob.name);

 // Display blob name and URL
 console.log(
 ` \n\tname: ${blob.name}\n\tURL: ${tempBlockBlobClient.url}\n`
);
}
```

Le code précédent appelle la méthode `listBlobsFlat`. Dans ce cas, un seul objet blob a été ajouté au conteneur. Il n'y a donc qu'un objet blob répertorié.

Pour en savoir plus sur les listings d'objets blob et explorer d'autres exemples de code, consultez [Répertorier les objets blob avec JavaScript](#).

## Télécharger des objets blob

1. Ajoutez le code suivant à la fin de la fonction `main` pour télécharger l'objet blob créé précédemment dans le runtime de l'application.

JavaScript

```
// Get blob content from position 0 to the end
// In Node.js, get downloaded data by accessing
downloadBlockBlobResponse.readableStreamBody
// In browsers, get downloaded data by accessing
```

```
downloadBlockBlobResponse.blobBody
const downloadBlockBlobResponse = await blockBlobClient.download(0);
console.log('\nDownloaded blob content...');
console.log(
 '\t',
 await streamToText(downloadBlockBlobResponse.readableStreamBody)
);
```

Le code précédent appelle la méthode `download`.

2. Copiez le code suivant *après* la fonction `main` pour reconvertir un flux en chaîne.

JavaScript

```
// Convert stream to text
async function streamToText(readable) {
 readable.setEncoding('utf8');
 let data = '';
 for await (const chunk of readable) {
 data += chunk;
 }
 return data;
}
```

Pour en savoir plus sur le téléchargement d'objets blob et explorer d'autres exemples de code, consultez [Télécharger un objet blob avec JavaScript](#).

## Supprimer un conteneur

Ajoutez ce code à la fin de la fonction `main` pour supprimer le conteneur et tous ses objets blob :

JavaScript

```
// Delete container
console.log('\nDeleting container...');

const deleteContainerResponse = await containerClient.delete();
console.log(
 'Container was deleted successfully. requestId: ',
 deleteContainerResponse.requestId
);
```

Le code précédent nettoie les ressources créées par l'application en supprimant le conteneur tout entier à l'aide de la méthode `delete`. Si vous voulez, vous pouvez aussi supprimer les fichiers locaux.

Pour en savoir plus sur la suppression d'un conteneur et explorer d'autres exemples de code, consultez [Supprimer et restaurer un conteneur d'objets blob avec JavaScript](#).

## Exécuter le code

1. À partir d'un terminal Visual Studio Code, exécutez l'application.

```
Console
```

```
node index.js
```

2. La sortie de l'application ressemble à l'exemple suivant :

```
Sortie
```

```
Azure Blob storage - JavaScript quickstart sample

Creating container...
 quickstart4a0780c0-fb72-11e9-b7b9-b387d3c488da

Uploading to Azure Storage as blob:
 quickstart4a3128d0-fb72-11e9-b7b9-b387d3c488da.txt

Listing blobs...
 quickstart4a3128d0-fb72-11e9-b7b9-b387d3c488da.txt

Downloaded blob content...
 Hello, World!

Deleting container...
Done
```

Parcourez le code dans le débogueur et vérifiez votre [portail Azure](#) tout au long du processus. Vérifiez que le conteneur est en cours d'exécution. Vous pouvez ouvrir le blob à l'intérieur du conteneur et afficher le contenu.

## Nettoyage

1. Lorsque vous avez terminé ce démarrage rapide, supprimez le répertoire `blob-quickstart`.
2. Si vous avez terminé d'utiliser votre ressource Stockage Azure, utilisez [Azure CLI pour supprimer la ressource de stockage](#).

## Étapes suivantes

Dans ce démarrage rapide, vous avez appris à charger, télécharger et répertorier des blobs avec JavaScript.

Pour afficher des exemples d'applications de stockage blob, passez à :

### Exemples de bibliothèques de client Stockage Blob Azure pour JavaScript

- Pour plus d'informations, consultez les [bibliothèques de client Stockage Blob Azure pour JavaScript](#).
- Pour obtenir des tutoriels, des exemples, des guides de démarrage rapide et autre documentation, consultez [Azure pour les développeurs JavaScript et Node.js](#).

# Démarrage rapide : Bibliothèque de client Stockage File d'attente Azure pour JavaScript

Article • 29/06/2023

Familiarisez-vous avec la bibliothèque de client Storage File d'attente Azure pour JavaScript. Le Stockage File d'attente Azure est un service permettant de stocker un grand nombre de messages dans le but de les récupérer et de les traiter plus tard. Suivez les étapes suivantes pour installer le package et essayer un exemple de code pour les tâches de base.

[Documentation de référence sur les API](#) | [Code source de la bibliothèque](#) ↗ |  
[Package \(npm\)](#) ↗ | [Exemples](#)

Utilisez la bibliothèque de client Stockage File d'attente Azure pour JavaScript afin d'effectuer les opérations suivantes :

- Créer une file d'attente
- Ajouter des messages à une file d'attente
- Afficher un aperçu des messages d'une file d'attente
- Mettre à jour un message dans une file d'attente
- Obtention de la longueur de la file d'attente
- Recevoir les messages d'une file d'attente
- Supprimer des messages d'une file d'attente
- Suppression d'une file d'attente

## Prérequis

- Abonnement Azure : [créez-en un gratuitement](#) ↗
- Compte de stockage Azure : [créez un compte de stockage](#)
- [Node.js](#) ↗ actuel pour votre système d'exploitation.

## Configuration

Cette section vous guide tout au long de la préparation d'un projet à utiliser avec la bibliothèque de client Stockage File d'attente Azure pour JavaScript.

## Créer le projet

Créez une application Node.js nommée `queues-quickstart`.

1. Dans une fenêtre de console (telle que cmd, PowerShell ou Bash), créez un nouveau répertoire pour le projet :

```
Console
mkdir queues-quickstart
```

2. Basculez vers le répertoire `queues-quickstart` créé :

```
Console
cd queues-quickstart
```

3. Créez un `package.json` :

```
Console
npm init -y
```

4. Ouvrez le projet dans Visual Studio Code :

```
Console
code .
```

## Installer les packages

À partir du répertoire du projet, installez les packages suivants à l'aide de la commande

`npm install`.

1. Installer le package npm Stockage File d'attente Azure :

```
Console
npm install @azure/storage-queue
```

2. Installez le package npm Azure Identity pour prendre en charge les connexions sans mot de passe :

```
Console
```

```
npm install @azure/identity
```

3. Installer d'autres dépendances utilisées dans ce démarrage rapide :

Console

```
npm install uuid dotenv
```

## Configurer le framework d'application

À partir du répertoire de projet :

1. Ouvrez un nouveau fichier texte dans votre éditeur de code
2. Ajouter des appels `require` pour charger des modules Azure et Node.js
3. Créer la structure du programme, y compris la gestion des exceptions de base

Voici le code :

JavaScript

```
const { QueueClient } = require("@azure/storage-queue");
const { DefaultAzureCredential } = require('@azure/identity');
const { v1: uuidv1 } = require("uuid");

async function main() {
 console.log("Azure Queue Storage client library - JavaScript
quickstart sample");

 // Quickstart code goes here
}

main().then(() => console.log("\nDone")).catch((ex) =>
 console.log(ex.message));
```

4. Enregistrez le nouveau fichier sous `index.js` dans le répertoire `queues-quickstart`.

## Authentification auprès d'Azure

Les requêtes d'application vers les Services Azure doivent être autorisées. L'utilisation de la classe `DefaultAzureCredential` fournie par la bibliothèque de client Azure Identity est

l'approche recommandée pour implémenter des connexions sans mot de passe aux services Azure dans votre code.

Vous pouvez également autoriser directement les requêtes adressées aux services Azure à l'aide de mots de passe, de chaînes de connexion ou d'autres informations d'identification. Toutefois, cette approche doit être utilisée avec prudence. Les développeurs doivent être vigilants pour ne jamais exposer les secrets dans un emplacement non sécurisé. Toute personne ayant accès au mot de passe ou à la clé secrète est en mesure de s'authentifier. `DefaultAzureCredential` offre des avantages améliorés en matière de gestion et de sécurité par rapport à la clé de compte pour autoriser l'authentification sans mot de passe. Les deux options sont illustrées dans l'exemple suivant.

#### Sans mot de passe (recommandé)

`DefaultAzureCredential` est une classe fournie par la bibliothèque de client Azure Identity pour JavaScript. Pour en savoir plus sur `DefaultAzureCredential`, consultez la vue d'ensemble de [DefaultAzureCredential](#). `DefaultAzureCredential` prend en charge plusieurs méthodes d'authentification et détermine quelle méthode doit être utilisée au moment de l'exécution. Cette approche permet à votre application d'utiliser différentes méthodes d'authentification dans différents environnements (local ou production) sans implémenter de code spécifique à l'environnement.

Par exemple, votre application peut s'authentifier à l'aide de vos informations d'identification de connexion Azure CLI lors du développement local, puis utiliser une [identité managée](#) une fois qu'elle a été déployée sur Azure. Aucune modification du code n'est requise pour cette transition.

Lors du développement localement, assurez-vous que le compte d'utilisateur qui accède aux données de file d'attente dispose des autorisations appropriées. Vous aurez besoin du **Contributeur aux données de file d'attente de stockage** pour lire et écrire des données blob. Pour vous attribuer ce rôle, vous aurez besoin du rôle **Administrateur de l'accès utilisateur** ou d'un autre rôle qui inclut l'action **Microsoft.Authorization/roleAssignments/write**. Vous pouvez attribuer des rôles RBAC Azure à un utilisateur à l'aide du Portail Azure, Azure CLI ou Azure PowerShell. Vous pouvez en savoir plus sur les étendues disponibles pour les attributions de rôles dans la page [vue d'ensemble de l'étendue](#).

Dans ce scénario, vous allez attribuer des autorisations à votre compte d'utilisateur, étendues au compte de stockage, pour suivre le [Principe des priviléges minimum](#). Cette pratique offre aux utilisateurs uniquement les autorisations minimales nécessaires et crée des environnements de production plus sécurisés.

L'exemple suivant attribue le rôle **Contributeur aux données file d'attente du stockage** à votre compte d'utilisateur, qui fournit à la fois un accès en lecture et en écriture aux données file d'attente dans votre compte de stockage.

### ⓘ Important

Dans la plupart des cas, la propagation de l'attribution de rôle dans Azure peut prendre une ou deux minutes, mais dans de rares cas, cela peut prendre jusqu'à huit minutes. Si vous recevez des erreurs d'authentification lorsque vous exécutez votre code pour la première fois, patientez quelques instants et réessayez.

Azure portal

1. Dans le Portail Azure, recherchez votre compte de stockage à l'aide de la barre de recherche principale ou de la navigation gauche.
2. Dans la page vue d'ensemble du compte de stockage, sélectionnez **Contrôle d'accès (IAM)** dans le menu de gauche.
3. Sur la page **Contrôle d'accès (IAM)**, sélectionnez l'onglet **Attributions de rôles**.
4. Sélectionnez **+ Ajouter** dans le menu supérieur, puis **Ajouter une attribution de rôle** dans le menu déroulant résultant.

The screenshot shows the Azure Storage Account Access Control (IAM) interface. The left sidebar has 'Access Control (IAM)' selected. The main area has a red box around the 'Add role assignment' button in the top navigation bar. A dropdown menu is open, also with a red box around 'Add role assignment'. To the right, there are sections for 'Grant access to this resource' (with 'Add role assignment' and 'Learn more' buttons) and 'View deny assignments' (with 'View' and 'Learn more' buttons).

5. Utilisez la zone de recherche pour filtrer les résultats sur le rôle souhaité. Pour cet exemple, recherchez *Contributeur aux données file d'attente du*

stockage, sélectionnez le résultat correspondant, puis choisissez **Suivant**.

6. Sous **Attribuer l'accès à**, sélectionnez **Utilisateur, groupe ou principal de service**, puis sélectionnez + **Sélectionner des membres**.

7. Dans la boîte de dialogue, recherchez votre nom d'utilisateur Microsoft Entra (généralement votre adresse e-mail *utilisateur@domaine*), puis choisissez **Sélectionner** en bas de la boîte de dialogue.

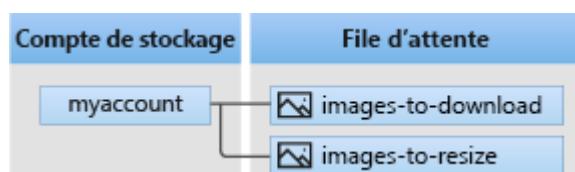
8. Sélectionnez **Vérifier + affecter** pour accéder à la page finale, puis **Vérifier + attribuer** à nouveau pour terminer le processus.

## Modèle objet

Stockage File d'attente Azure est un service permettant de stocker un grand nombre de messages. La taille maximale d'un message de file d'attente est de 64 Ko. Une file d'attente peut contenir des millions de messages, dans la limite de la capacité totale d'un compte de stockage. Les files d'attente sont couramment utilisées pour créer un backlog de travail à traiter de façon asynchrone. Le Stockage File d'attente offre trois types de ressources :

- **Compte de stockage** : Tous les accès à Azure Storage passent par un compte de stockage. Pour plus d'informations sur les comptes de stockage, consultez [Vue d'ensemble des comptes de stockage](#).
- **File d'attente** : une file d'attente contient un ensemble de messages. Tous les messages doivent être dans une file d'attente. Notez que le nom de la file d'attente doit être en minuscules. Pour plus d'informations sur l'affectation de noms à des files d'attente, consultez [Affectation de noms pour les files d'attente et les métadonnées](#).
- **Message** : message dans n'importe quel format d'une taille maximale de 64 Ko. Un message peut rester dans la file d'attente pendant un maximum de 7 jours. Pour les versions du 29 juillet 2017 ou ultérieures, la durée de vie maximale peut être n'importe quel nombre positif, ou -1 indiquant que le message n'expire pas. Si ce paramètre est omis, la valeur par défaut de la durée de vie est de sept jours.

Le diagramme suivant montre la relation entre ces ressources.



Utilisez les classes JavaScript suivantes pour interagir avec ces ressources :

- [QueueServiceClient](#) : une instance `QueueServiceClient` représente une connexion à un compte de stockage donné dans le service de File d'attente Stockage Azure. Ce client vous permet de gérer toutes les files d'attente de votre compte de stockage.
- [QueueClient](#) : une instance `QueueClient` représente une file d'attente unique dans un compte de stockage. Ce client vous permet de gérer et de manipuler une file d'attente individuelle et ses messages.

## Exemples de code

Ces exemples d'extraits de code vous montrent comment effectuer les actions suivantes avec la bibliothèque de client Stockage File d'attente Azure pour JavaScript :

- Autoriser l'accès et créer un objet client
- Créer une file d'attente
- Ajouter des messages à une file d'attente
- Afficher un aperçu des messages d'une file d'attente
- Mettre à jour un message dans une file d'attente
- Obtention de la longueur de la file d'attente
- Recevoir les messages d'une file d'attente
- Supprimer des messages d'une file d'attente
- Supprimer une file d'attente

Sans mot de passe (recommandé)

### Autoriser l'accès et créer un objet client

Vérifiez que vous êtes authentifié avec le même compte Microsoft Entra auquel vous avez attribué le rôle. Vous pouvez vous authentifier via Azure CLI, Visual Studio Code ou Azure PowerShell.

Azure CLI

Connectez-vous à Azure via Azure CLI à l'aide de la commande suivante :

Azure CLI

```
az login
```

Une fois authentifié, vous pouvez créer et autoriser un objet `QueueClient` à l'aide de `DefaultAzureCredential` pour accéder aux données de file d'attente dans le compte de stockage. `DefaultAzureCredential` découvre et utilise automatiquement le compte avec lequel vous vous êtes connecté à l'étape précédente.

Pour autoriser à l'aide de `DefaultAzureCredential`, vérifiez que vous avez ajouté le package `@azure/identité`, comme décrit dans [Installer les packages](#). Veillez également à charger le module `@azure/identité` dans le fichier `index.js` :

JavaScript

```
const { DefaultAzureCredential } = require('@azure/identity');
```

Choisissez un nom pour la file d'attente et créez une instance de la classe `QueueClient`, en utilisant `DefaultAzureCredential` pour l'autorisation. Nous utilisons cet objet client pour créer et interagir avec la ressource de file d'attente dans le compte de stockage.

### ⓘ Important

Les noms de file d'attente peuvent contenir uniquement des lettres minuscules, des chiffres et des traits d'union, et doivent commencer par une lettre ou un nombre. Chaque trait d'union doit être précédé et suivi d'un caractère autre qu'un tiret. Le nom doit avoir entre 3 et 63 caractères. Pour plus d'informations sur le nommage des files d'attente, consultez [Nommage des files d'attente et des métadonnées](#).

Ajoutez le code suivant à l'intérieur de la méthode `main` et veillez à remplacer la valeur d'espace `<storage-account-name>` réservé :

JavaScript

```
// Create a unique name for the queue
const queueName = "quickstart" + uuidv1();

// Instantiate a QueueClient which will be used to create and interact
// with a queue
// TODO: replace <storage-account-name> with the actual name
const queueClient = new QueueClient(`https://<storage-account-
name>.queue.core.windows.net/${queueName}`, new
DefaultAzureCredential());
```

## ⓘ Notes

Les messages que vous envoyez à l'aide de la classe `QueueClient` doivent être dans un format pouvant être inclus dans une requête XML avec encodage UTF-8. Pour inclure un balisage dans le message, le contenu de ce dernier doit être placé dans une séquence d'échappement XML ou encodé au format Base64.

Les messages des files d'attente sont stockés sous forme de chaînes. Si vous devez envoyer un autre type de données, vous devez le sérialiser dans une chaîne lors de l'envoi du message et désérialiser le format de chaîne lors de la lecture du message.

Pour convertir **JSON** au format chaîne et revenir à nouveau dans Node.js, utilisez les fonctions d'assistance suivantes :

JavaScript

```
function jsonToBase64(jsonObj) {
 const jsonString = JSON.stringify(jsonObj)
 return Buffer.from(jsonString).toString('base64')
}

function encodeBase64ToJson(base64String) {
 const jsonString = Buffer.from(base64String, 'base64').toString()
 return JSON.parse(jsonString)
}
```

## Créer une file d'attente

Avec l'objet `QueueClient`,appelez la méthode `create` pour créer la file d'attente dans votre compte de stockage.

Ajoutez ce code à la fin de la méthode `main` :

JavaScript

```
console.log("\nCreating queue...");
console.log("\t", queueName);

// Create the queue
const createQueueResponse = await queueClient.create();
console.log("Queue created, requestId:", createQueueResponse.requestId);
```

## Ajouter des messages à une file d'attente

L'extrait de code suivant ajoute des messages à la file d'attente en appelant la méthode `sendMessage`. Il enregistre également le `QueueSendMessageResponse` retourné par le troisième appel de `sendMessage`. Le `sendMessageResponse` retourné est utilisé pour mettre à jour le contenu du message ultérieurement dans le programme.

Ajoutez ce code à la fin de la fonction `main` :

```
JavaScript

console.log("\nAdding messages to the queue...");

// Send several messages to the queue
await queueClient.sendMessage("First message");
await queueClient.sendMessage("Second message");
const sendMessageResponse = await queueClient.sendMessage("Third message");

console.log("Messages added, requestId:", sendMessageResponse.requestId);
```

## Afficher un aperçu des messages d'une file d'attente

Affichez un aperçu des messages de la file d'attente en appelant la méthode `peekMessages`. Cette méthode récupère un ou plusieurs messages du début de la file d'attente, mais ne modifie pas la visibilité du message. Par défaut, `peekMessages` lit furtivement un seul message.

Ajoutez ce code à la fin de la fonction `main` :

```
JavaScript

console.log("\nPeek at the messages in the queue...");

// Peek at messages in the queue
const peekedMessages = await queueClient.peekMessages({ numberOfMessages : 5 });

for (i = 0; i < peekedMessages.peekedMessageItems.length; i++) {
 // Display the peeked message
 console.log("\t", peekedMessages.peekedMessageItems[i].messageText);
}
```

## Mettre à jour un message dans une file d'attente

Mettez à jour le contenu d'un message en appelant la méthode `updateMessage`. Cette méthode peut changer le contenu et le délai d'expiration de la visibilité d'un message. Le contenu du message doit être une chaîne encodée en UTF-8 d'une taille maximale de

64 Ko. Avec le nouveau contenu, transmettez `messageId` et `popReceipt` à partir de la réponse qui a été enregistrée dans le code. Les propriétés `sendMessageResponse` identifient le message à mettre à jour.

JavaScript

```
console.log("\nUpdating the third message in the queue...");

// Update a message using the response saved when calling sendMessage
earlier
updateMessageResponse = await queueClient.updateMessage(
 sendMessageResponse.messageId,
 sendMessageResponse.popReceipt,
 "Third message has been updated"
);

console.log("Message updated, requestId:", updateMessageResponse.requestId);
```

## Obtention de la longueur de la file d'attente

La méthode `getProperties` renvoie des métadonnées sur la file d'attente, notamment le nombre approximatif de messages en attente dans la file d'attente.

JavaScript

```
const properties = await queueClient.getProperties();
console.log("Approximate queue length: ",
properties.approximateMessagesCount);
```

## Réception des messages d'une file d'attente

Téléchargez les messages ajoutés en appelant la méthode `receiveMessages`. Dans le champ `numberOfMessages`, transmettez le nombre maximal de messages à recevoir pour cet appel.

Ajoutez ce code à la fin de la fonction `main` :

JavaScript

```
console.log("\nReceiving messages from the queue...");

// Get messages from the queue
const receivedMessagesResponse = await queueClient.receiveMessages({
 numberOfMessages : 5 });
```

```
console.log("Messages received, requestId:",
receivedMessagesResponse.requestId);
```

Lorsque vous appelez la méthode `receiveMessages`, vous pouvez éventuellement spécifier des valeurs dans `QueueReceiveMessageOptions` pour personnaliser la récupération des messages. Vous pouvez spécifier une valeur pour `numberOfMessages`, qui est le nombre de messages à récupérer à partir de la file d'attente. La valeur par défaut est de 1 message et la valeur maximale est de 32 messages. Vous pouvez également spécifier une valeur pour `visibilityTimeout`, qui masque les messages aux autres opérations pendant la période d'expiration. La valeur par défaut est 30 secondes.

## Supprimer des messages d'une file d'attente

Vous pouvez supprimer les messages de la file d'attente une fois qu'ils sont reçus et traités. Dans ce cas, le traitement affiche simplement le message sur la console.

Supprimez les messages en appelant la méthode `deleteMessage`. Les messages qui ne sont pas supprimés explicitement redeviennent visibles dans la file d'attente et peuvent éventuellement être de nouveau traités.

Ajoutez ce code à la fin de la fonction `main` :

JavaScript

```
// 'Process' and delete messages from the queue
for (i = 0; i < receivedMessagesResponse.receivedMessageItems.length; i++) {
 receivedMessage = receivedMessagesResponse.receivedMessageItems[i];

 // 'Process' the message
 console.log("\tProcessing:", receivedMessage.messageText);

 // Delete the message
 const deleteMessageResponse = await queueClient.deleteMessage(
 receivedMessage.messageId,
 receivedMessage.popReceipt
);
 console.log("\tMessage deleted, requestId:",
 deleteMessageResponse.requestId);
}
```

## Suppression d'une file d'attente

Le code suivant nettoie les ressources créées par l'application en supprimant la file d'attente avec la méthode `delete`.

Ajoutez ce code à la fin de la fonction `main` et enregistrez le fichier :

JavaScript

```
// Delete the queue
console.log("\nDeleting queue...");
const deleteQueueResponse = await queueClient.delete();
console.log("Queue deleted, requestId:", deleteQueueResponse.requestId);
```

## Exécuter le code

Cette application crée trois messages et les ajoute à une file d'attente Azure. Le code liste les messages dans la file d'attente, puis les récupère et les supprime avant de supprimer la file d'attente.

Dans la fenêtre de votre console, accédez au répertoire contenant le fichier `index.js`, puis utilisez la commande `node` suivante pour exécuter l'application.

Console

```
node index.js
```

La sortie de l'application ressemble à l'exemple suivant :

Sortie

```
Azure Queue Storage client library - JavaScript quickstart sample
```

```
Creating queue...
 quickstart<UUID>
Queue created, requestId: 5c0bc94c-6003-011b-7c11-b13d06000000
```

```
Adding messages to the queue...
Messages added, requestId: a0390321-8003-001e-0311-b18f2c000000
```

```
Peek at the messages in the queue...
 First message
 Second message
 Third message
```

```
Updating the third message in the queue...
Message updated, requestId: cb172c9a-5003-001c-2911-b18dd6000000
```

```
Receiving messages from the queue...
Messages received, requestId: a039036f-8003-001e-4811-b18f2c000000
 Processing: First message
 Message deleted, requestId: 4a65b82b-d003-00a7-5411-b16c22000000
 Processing: Second message
```

```
Message deleted, requestId: 4f0b2958-c003-0030-2a11-b10feb000000
Processing: Third message has been updated
Message deleted, requestId: 6c978fcb-5003-00b6-2711-b15b39000000

Deleting queue...
Queue deleted, requestId: 5c0bca05-6003-011b-1e11-b13d06000000

Done
```

Parcourez le code dans le débogueur et vérifiez votre [portail Azure](#) tout au long du processus. Vérifiez votre compte de stockage pour vous assurer que les messages de la file d'attente sont créés et supprimés.

## Étapes suivantes

Dans ce guide de démarrage rapide, vous avez appris à créer une file d'attente et à y ajouter des messages à l'aide de code JavaScript. Ensuite, vous avez appris à afficher un aperçu des messages, à les récupérer et à les supprimer. Enfin, vous avez appris à supprimer une file d'attente de messages.

Pour obtenir des tutoriels, des exemples, des guides de démarrage rapide et d'autres documents, visitez :

### Documentation Azure pour JavaScript

- Pour plus d'informations, consultez la [bibliothèque de client Stockage File d'attente Azure pour JavaScript](#).
- Pour d'autres exemples d'applications Stockage File d'attente Azure, consultez [Exemples de bibliothèques de client Stockage File d'attente Azure pour JavaScript](#).

# Se connecter à et interroger Azure SQL Database à l'aide de Python et du pilote pyodbc

Article • 29/12/2023

ⓘ Contenu assisté par l'IA. Cet article a été en partie créé à l'aide d'une IA. Un auteur a examiné et si besoin révisé le contenu. [En savoir plus](#)

S'applique à  [Azure SQL Database](#)

Ce démarrage rapide explique comment connecter une application à une base de données dans Azure SQL Database et effectuer des requêtes à l'aide de Python et du [Pilote SQL Python - pyodbc](#). Ce guide de démarrage rapide suit l'approche sans mot de passe recommandée pour se connecter à la base de données. Vous pouvez en apprendre plus sur les connexions sans mot de passe sur le [Hub des connexions sans mot de passe](#).

## Prérequis

- Un [abonnement Azure](#).
- Une base de données Azure SQL configurée avec l'authentification Microsoft Entra. Vous pouvez en créer une à l'aide du [Guide de démarrage rapide Créer une base de données](#).
- La version la plus récente d'[Azure CLI](#).
- Visual Studio Code avec l'[extension Python](#).
- Python 3.8 ou version ultérieure.

## Configurer la base de données

Les connexions sécurisées et sans mot de passe à Azure SQL Database nécessitent certaines configurations de base de données. Vérifiez les paramètres suivants sur votre [serveur logique dans Azure](#) pour vous connecter correctement à Azure SQL Database dans les environnements locaux et hébergés :

1. Pour les connexions de développement local, vérifiez que votre serveur logique est configuré pour autoriser l'adresse IP de votre ordinateur local et d'autres services Azure à se connecter :
  - Accédez à la page [Réseau](#) de votre serveur.

- Activez la case d'option **Réseaux sélectionnés** pour voir des options de configuration supplémentaires.
- Sélectionnez **Ajouter l'adresse IPv4 de votre client (xx.xx.xx.xx.xx.xx)** pour ajouter une règle de pare-feu qui activera les connexions à partir de l'adresse IPv4 de votre ordinateur local. Vous pouvez également sélectionner **+ Ajouter une règle de pare-feu** pour entrer une adresse IP spécifique de votre choix.
- Vérifiez que la case **Autoriser les services et les ressources Azure à accéder à ce serveur** est cochée.

The screenshot shows the Azure portal's Networking settings for an Azure SQL Database. The left sidebar shows various service categories like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Quick start, Settings, Azure Active Directory, SQL databases, SQL elastic pools, DTU quota, Properties, Locks, Data management, Backups, Deleted databases, Failover groups, Import/Export history, Security, and Networking. The Networking option is highlighted with a red box. The main pane shows the Public access tab with 'Selected networks' selected. Under Firewall rules, there is a button to 'Add your client IPv4 address' which is also highlighted with a red box. In the Exceptions section, a checkbox for 'Allow Azure services and resources to access this server' is checked and highlighted with a red box. A magnifying glass icon is in the bottom right corner.

### ⚠️ Avertissement

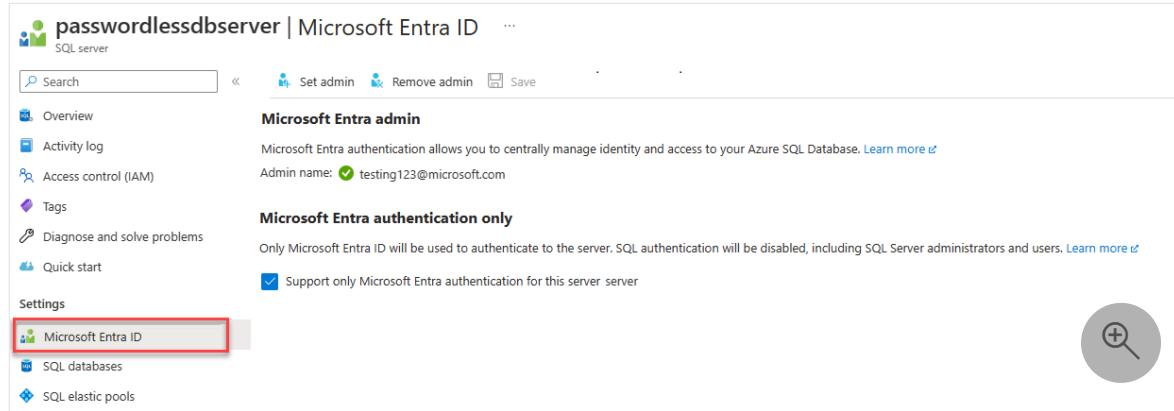
L'activation du paramètre **Autoriser les services et les ressources Azure à accéder à ce serveur** n'est pas une pratique de sécurité recommandée pour les scénarios de production. Les applications réelles doivent implémenter des approches plus sécurisées, telles que des restrictions de pare-feu ou des configurations de réseau virtuel plus strictes.

Vous pouvez en apprendre davantage sur les configurations de sécurité de base de données avec les ressources suivantes :

- **Configurer des règles de pare-feu Azure SQL Database.**
- **Configurer un réseau virtuel avec des points de terminaison privés.**

2. Le serveur doit également activer l'authentification Microsoft Entra et disposer d'un compte d'administrateur Microsoft Entra affecté. Pour les connexions de développement local, le compte d'administrateur de Microsoft Entra doit être un compte que vous pouvez également connecter localement à Visual Studio ou à

Azure CLI. Vous pouvez vérifier si l'authentification Microsoft Entra est activée sur la page **Microsoft Entra ID** de votre serveur logique.



The screenshot shows the Azure portal interface for managing a SQL server named 'passwordlessdbserver'. In the left sidebar under 'Settings', the 'Microsoft Entra ID' option is highlighted with a red box. The main content area is titled 'Microsoft Entra admin' and contains the following information:

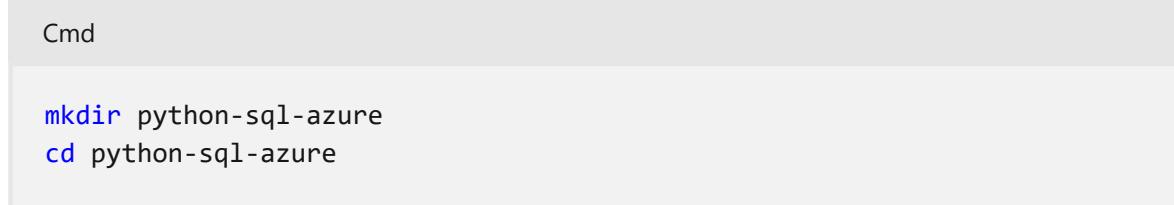
- Microsoft Entra authentication allows you to centrally manage identity and access to your Azure SQL Database. [Learn more](#)
- Admin name: **testing123@microsoft.com**
- Microsoft Entra authentication only**: A checkbox labeled 'Only Microsoft Entra will be used to authenticate to the server. SQL authentication will be disabled, including SQL Server administrators and users.' is checked.
- Support only Microsoft Entra authentication for this server

3. Si vous utilisez un compte Azure personnel, assurez-vous d'avoir [Microsoft Entra installé et configuré dans la base de données Azure SQL](#) afin d'assigner votre compte en tant qu'administrateur de serveur. En revanche, si vous utilisez un compte d'entreprise, il est fort probable que Microsoft Entra ID soit déjà configuré pour vous.

## Créer le projet

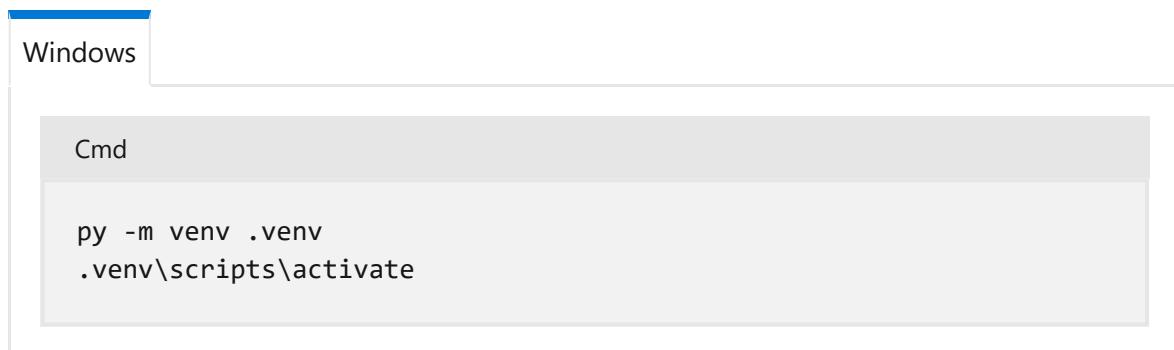
Créez un projet de rapport à l'aide de Visual Studio Code.

1. Ouvrez Visual Studio Code et créez un dossier pour votre projet et modifiez-y le répertoire.



```
Cmd
mkdir python-sql-azure
cd python-sql-azure
```

2. Créez un environnement virtuel pour l'application.



```
Windows
Cmd
py -m venv .venv
.venv\scripts\activate
```

3. Créez un fichier Python appelé `app.py`.

# Installer le pilote pyodbc

Pour vous connecter à Azure SQL Database à l'aide de Python, installez le pilote `pyodbc`. Ce package joue le rôle de fournisseur de données pour la connexion aux bases de données, l'exécution de commandes et la récupération des résultats. Dans ce démarrage rapide, vous installez également les packages `flask`, `uvicorn` et `pydantic` pour créer et exécuter une API.

Pour plus d'informations et des instructions spécifiques sur l'installation du pilote `pyodbc` sur tous les systèmes d'exploitation, consultez [Configurer l'environnement de développement pour le développement Python pyodbc](#).

1. Créez un fichier `requirements.txt` avec les lignes suivantes :

```
pyodbc
fastapi
uvicorn[standard]
pydantic
azure-identity
```

2. Installez les exigences.

```
Console
pip install -r requirements.txt
```

## Configurer la chaîne de connexion locale

Pour le développement local et la connexion à Azure SQL Database, ajoutez la variable d'environnement suivante `AZURE_SQL_CONNECTIONSTRING`. Remplacez les espaces réservés `<database-server-name>` et `<database-name>` par vos valeurs. Des exemples de variables d'environnement sont affichés pour l'interpréteur de commandes Bash.

L'authentification interactive fournit une option sans mot de passe lorsque vous exécutez localement.

### Authentification interactive

Dans Windows, l'authentification interactive Microsoft Entra peut utiliser la technologie d'authentification multifacteur Microsoft Entra pour configurer la

connexion. Dans ce mode, lorsque l'ID de connexion est fourni, une boîte de dialogue d'authentification Azure est déclenchée, permettant ainsi à l'utilisateur d'entrer le mot de passe pour établir la connexion.

Bash

```
export AZURE_SQL_CONNECTIONSTRING='Driver={ODBC Driver 18 for SQL Server};Server=tcp:<database-server-name>.database.windows.net,1433;Database=<database-name>;Encrypt=yes;TrustServerCertificate=no;Connection Timeout=30'
```

Pour plus d'informations, consultez [Utilisation de Microsoft Entra ID avec le Pilote ODBC](#). Si vous utilisez cette option, recherchez la fenêtre qui vous invite à entrer des informations d'identification.

Vous pouvez obtenir les détails de la création de votre chaîne de connexion à partir du Portail Azure :

1. Accédez à Azure SQL Server, sélectionnez la page **Bases de données SQL** pour trouver le nom de votre base de données, puis sélectionnez la base de données.
2. Dans la base de données, accédez à la page **Chaînes de connexion** pour obtenir des informations sur la chaîne de connexion. Regardez sous l'onglet **ODBC**.

#### ⓘ Notes

Si vous avez installé **Azure Arc** et l'avez associé à votre abonnement Azure, vous pouvez également utiliser l'approche d'identité managée indiquée pour l'application déployée sur App Service.

## Ajouter du code pour se connecter à Azure SQL Database

Dans le dossier du projet, créez un fichier *app.py* et ajoutez l'exemple de code. Ce code crée une API qui :

- récupère une chaîne de connexion de base Azure SQL Database à partir d'une variable d'environnement
- crée une table `Persons` dans la base de données au démarrage (pour les scénarios de test uniquement)

- définit une fonction pour récupérer tous les enregistrements `Person` de la base de données.
- définit une fonction pour récupérer un enregistrement `Person` de la base de données
- Définit une fonction pour ajouter de nouveaux enregistrements `Person` à la base de données.

Python

```

import os
import pyodbc, struct
from azure import identity

from typing import Union
from fastapi import FastAPI
from pydantic import BaseModel

class Person(BaseModel):
 first_name: str
 last_name: Union[str, None] = None

connection_string = os.environ["AZURE_SQL_CONNECTIONSTRING"]

app = FastAPI()

@app.get("/")
def root():
 print("Root of Person API")
 try:
 conn = get_conn()
 cursor = conn.cursor()

 # Table should be created ahead of time in production app.
 cursor.execute("""
 CREATE TABLE Persons (
 ID int NOT NULL PRIMARY KEY IDENTITY,
 FirstName varchar(255),
 LastName varchar(255)
);
 """)

 conn.commit()
 except Exception as e:
 # Table may already exist
 print(e)
 return "Person API"

@app.get("/all")
def get_persons():
 rows = []
 with get_conn() as conn:
 cursor = conn.cursor()

```

```

 cursor.execute("SELECT * FROM Persons")

 for row in cursor.fetchall():
 print(row.FirstName, row.LastName)
 rows.append(f"{row.ID}, {row.FirstName}, {row.LastName}")
 return rows

@app.get("/person/{person_id}")
def get_person(person_id: int):
 with get_conn() as conn:
 cursor = conn.cursor()
 cursor.execute("SELECT * FROM Persons WHERE ID = ?", person_id)

 row = cursor.fetchone()
 return f"{row.ID}, {row.FirstName}, {row.LastName}"

@app.post("/person")
def create_person(item: Person):
 with get_conn() as conn:
 cursor = conn.cursor()
 cursor.execute(f"INSERT INTO Persons (FirstName, LastName) VALUES (?, ?)", item.first_name, item.last_name)
 conn.commit()

 return item

def get_conn():
 credential =
 identity.DefaultAzureCredential(exclude_interactive_browser_credential=False)
 token_bytes =
 credential.get_token("https://database.windows.net/.default").token.encode("UTF-16-LE")
 token_struct = struct.pack(f'<I{len(token_bytes)}s', len(token_bytes),
 token_bytes)
 SQL_COPT_SS_ACCESS_TOKEN = 1256 # This connection option is defined by
 microsoft in msodbcsql.h
 conn = pyodbc.connect(connection_string, attrs_before=
 {SQL_COPT_SS_ACCESS_TOKEN: token_struct})
 return conn

```

### Avertissement

L'exemple de code montre des instructions SQL brutes, qui ne doivent pas être utilisées dans le code de production. Utilisez plutôt un package ORM (Object Relational Mapper) comme [SqlAlchemy](#) qui génère une couche d'objets plus sécurisée pour accéder à votre base de données.

## Exécuter et tester l'application localement

L'application est prête à être testée localement.

1. Ouvrez le fichier `app.py` dans Visual Studio Code.

Console

```
uvicorn app:app --reload
```

2. Dans la page de l'interface utilisateur Swagger pour l'application <http://127.0.0.1:8000/docs>, développez la méthode POST, puis sélectionnez **Essayer**.

Vous pouvez également utiliser essayer `/redoc` pour afficher une autre forme de documentation générée pour l'API.

3. Modifiez l'exemple JSON pour inclure des valeurs pour le prénom et le nom. Sélectionnez **Exécuter** pour ajouter un nouvel enregistrement à la base de données. L'API retourne une réponse de succès.
4. Développez la méthode **GET** sur la page de l'interface utilisateur Swagger, puis sélectionnez **Essayer**. Choisissez **Exécuter**, et la personne que vous venez de créer est retournée.

## Déployer dans Azure App Service

L'application est prête à être déployée sur Azure.

1. Créez un fichier `start.sh` afin que gunicorn dans Azure App Service puisse exécuter uvicorn. Le fichier `start.sh` a une ligne :

Console

```
gunicorn -w 4 -k uvicorn.workers.UvicornWorker app:app
```

2. Utilisez `az webapp up` pour déployer le code sur App Service. (Vous pouvez utiliser l'option `-dryrun` pour voir ce que fait la commande sans créer la ressource.)

Azure CLI

```
az webapp up \
--resource-group <resource-group-name> \
--name <web-app-name>
```

3. Utilisez la commande [az webapp config set](#) pour configurer App Service et utiliser le fichier *start.sh*.

```
Azure CLI
```

```
az webapp config set \
--resource-group <resource-group-name> \
--name <web-app-name> \
--startup-file start.sh
```

4. Utilisez la commande [az webapp identity assign](#) pour activer une identité managée affectée par le système pour le App Service.

```
Azure CLI
```

```
az webapp identity assign \
--resource-group <resource-group-name> \
--name <web-app-name>
```

Dans ce démarrage rapide, une identité managée affectée par le système est utilisée pour la démonstration. Une identité managée affectée par l'utilisateur est plus efficace dans un plus large éventail de scénarios. Pour plus d'informations, consultez les [suggestions relatives aux meilleures pratiques d'identités managées](#). Pour obtenir un exemple d'utilisation d'une identité managée affectée par l'utilisateur avec pyodbc, consultez [Migrer une application Python pour utiliser des connexions sans mot de passe avec Azure SQL Database](#).

## Connecter App Service à Azure SQL Database

Dans la section [Configurer la base de données](#), vous avez configuré la mise en réseau et l'authentification Microsoft Entra pour le serveur SQL Database Azure. Dans cette section, vous terminez la configuration de la base de données et configurez App Service avec une chaîne de connexion pour accéder au serveur de base de données.

Pour exécuter ces commandes, vous pouvez utiliser n'importe quel outil ou IDE qui peut se connecter à Azure SQL Database, y compris [SQL Server Management Studio \(SSMS\)](#), [Azure Data Studio](#) et Visual Studio Code avec l'extension [mssql SQL Server](#). Vous pouvez également utiliser le Portail Azure comme décrit dans [Démarrage rapide : utiliser l'éditeur de requêtes Portail Azure pour interroger Azure SQL Database](#).

1. Ajoutez un utilisateur à Azure SQL Database avec des commandes SQL pour créer un utilisateur et un rôle pour un accès sans mot de passe.

## SQL

```
CREATE USER [<web-app-name>] FROM EXTERNAL PROVIDER
ALTER ROLE db_datareader ADD MEMBER [<web-app-name>]
ALTER ROLE db_datawriter ADD MEMBER [<web-app-name>]
```

Pour plus d'informations, voir [Utilisateurs de base de données autonome - Rendre votre base de données portable](#). Pour obtenir un exemple qui montre le même principe, mais appliqué à une machine virtuelle Azure, consultez [Tutoriel : utiliser une identité managée affectée par le système d'une machine virtuelle Windows pour accéder à Azure SQL](#). Pour plus d'informations sur les rôles attribués, consultez [Rôles de base de données fixes](#).

Si vous désactivez puis activez l'identité managée affectée par le système App Service, supprimez l'utilisateur et recréez-le. Exécutez `DROP USER [<web-app-name>]` et réexécutez les commandes `CREATE` et `ALTER`. Pour afficher les utilisateurs, utilisez `SELECT * FROM sys.database_principals`.

2. Utilisez la commande `az webapp config appsettings set` pour ajouter un paramètre d'application pour la chaîne de connexion.

## Azure CLI

```
az webapp config appsettings set \
--resource-group <resource-group-name> \
--name <web-app-name> \
--settings AZURE_SQL_CONNECTIONSTRING=<connection-string>"
```

Pour l'application déployée, la chaîne de connexion doit ressembler à :

```
Driver={ODBC Driver 18 for SQL Server};Server=tcp:<database-server-name>.database.windows.net,1433;Database=<database-name>;Encrypt=yes;TrustServerCertificate=no;Connection Timeout=30
```

Remplacez `<dabaser-server-name>` et `<database-name>` par vos valeurs.

La chaîne de connexion sans mot de passe ne contient pas de nom d'utilisateur ou de mot de passe. Au lieu de cela, lorsque l'application s'exécute dans Azure, le code utilise `DefaultAzureCredential` à partir de la [bibliothèque Azure Identity](#) pour obtenir un jeton à utiliser avec `pyodbc`.

# Tester l'application déployée

Accédez à l'URL de l'application pour tester que la connexion à Azure SQL Database fonctionne. Vous pouvez localiser l'URL de votre application dans la page de vue d'ensemble d'App Service.

HTTP

`https://<web-app-name>.azurewebsites.net`

Ajoutez `/docs` à l'URL pour afficher l'interface utilisateur Swagger et tester les méthodes d'API.

Félicitations ! Votre application est maintenant connectée à Azure SQL Database dans les environnements locaux et hébergés.

## Contenu connexe

- [Migrer une application Python pour utiliser des connexions sans mot de passe avec Azure SQL Database](#) - Affiche l'identité managée affectée par l'utilisateur.
- [Connexions sans mot de passe pour les services Azure](#)
- [Recommandations relatives aux meilleures pratiques liées aux identités managées](#)

# Démarrage rapide : Bibliothèque Azure Cosmos DB for NoSQL pour Python

Article • 15/01/2024

S'APPLIQUE À : NoSQL

Bien démarrer avec la bibliothèque de client Azure Cosmos DB for NoSQL pour Python afin d'interroger des données dans vos conteneurs, et effectuer des opérations courantes sur des éléments individuels. Suivez ces étapes pour déployer une solution minimale sur votre environnement à l'aide de l'interface Azure Developer CLI.

[Documentation de référence API](#) | [Code source de la bibliothèque](#) | [Package \(PyPI\)](#) | [Azure Developer CLI](#)

## Prérequis

- Compte Azure avec un abonnement actif. [Créez un compte gratuitement](#).
- [Compte GitHub](#)

## Configuration

Déployez le conteneur de développement de ce projet sur votre environnement. Utilisez ensuite l'interface Azure Developer CLI (`azd`) pour créer un compte Azure Cosmos DB for NoSQL, et déployer un exemple d'application conteneurisé. L'exemple d'application utilise la bibliothèque de client pour gérer, créer, lire et interroger des exemples de données.

[GITHUB CODESPACES](#) [OPEN](#)

### Important

Les comptes GitHub incluent un droit d'utilisation relatif au stockage et aux heures d'utilisation d'un cœur gratuites. Pour plus d'informations, consultez [Stockage et heures cœur inclus pour les comptes GitHub](#).

1. Ouvrez un terminal dans le répertoire racine du projet.
2. Authentifiez-vous auprès de l'interface Azure Developer CLI en utilisant `azd auth login`. Suivez les étapes spécifiées par l'outil pour vous authentifier auprès de

l'interface CLI à l'aide de vos informations d'identification Azure préférées.

```
Azure CLI
```

```
azd auth login
```

3. Utilisez `azd init` pour initialiser le projet.

```
Azure CLI
```

```
azd init
```

4. Lors de l'initialisation, configurez un nom d'environnement unique.

#### 💡 Conseil

Le nom de l'environnement sera également utilisé comme nom du groupe de ressources cible. Pour ce guide de démarrage rapide, utilisez `msdocs-cosmos-db-nosql`.

5. Déployez le compte Azure Cosmos DB for NoSQL en utilisant `azd up`. Les modèles Bicep déploient également un exemple d'application web.

```
Azure CLI
```

```
azd up
```

6. Pendant le processus d'approvisionnement, sélectionnez votre abonnement et l'emplacement souhaité. Attendez la fin du processus de provisionnement. Le processus peut prendre **environ cinq minutes**.

7. Une fois le provisionnement de vos ressources Azure effectué, une URL vers l'application web en cours d'exécution est incluse dans la sortie.

```
Output
```

```
Deploying services (azd deploy)
```

```
(✓) Done: Deploying service web
- Endpoint: <https://[container-app-sub-domain].azurecontainerapps.io>
```

```
SUCCESS: Your application was provisioned and deployed to Azure in 5
minutes 0 seconds.
```

8. Utilisez l'URL dans la console pour accéder à votre application web dans le navigateur. Observez la sortie de l'application en cours d'exécution.



Azure Cosmos DB for NoSQL | Accéder au démarrage rapide

```
État actuel : Démarrage ..
Obtenir une base de données
Obtenir un conteneur : produits
Élément upsert : {70b63682-b93a-4c77-aad2-65501347265f gear-surf-surfboards Yamba Surfboard 12 850 false}
Code d'état : 201
Frais de requête : 7,24
Élément upsert : {25a698543-b90c-439d-8332-7ef41e06a0e0 gear-surf-surfboards Kiama Classic Surfboard 25 790 true}
Code d'état : 201
Frais de requête : 7,24
Identification de l'élément 70b63682-b93a-4c77-aad2-65501347265f
Élément de lecture : {70b63682-b93a-4c77-aad2-65501347265f gear-surf-surfboards Yamba Surfboard 12 850 false}
Code d'état : 200
Frais de requête : 1,00
```

## Installer la bibliothèque de client

La bibliothèque de client est disponible via le module Python Package Index (PyPI), en tant que bibliothèque `azure-cosmos`.

1. Ouvrez un terminal, puis accédez au dossier `/src`.

```
Bash

cd ./src
```

2. S'il n'est pas déjà installé, installez le package `azure-cosmos` à l'aide de `pip install`.

```
Bash

pip install azure-cosmos
```

3. Installez également le package `azure-identity`, s'il n'est pas déjà installé.

```
Bash

pip install azure-identity
```

4. Ouvrez et passez en revue le fichier `src/requirements.txt` pour vérifier que les entrées `azure-cosmos` et `azure-identity` existent.

## Modèle objet

Nom	Description
<a href="#">CosmosClient</a>	Il s'agit de la classe cliente principale utilisée pour gérer les métadonnées ou les bases de données à l'échelle du compte.
<a href="#">DatabaseProxy</a>	Cette classe représente une base de données dans le compte.
<a href="#">ContainerProxy</a>	Cette classe est principalement utilisée pour effectuer des opérations de lecture, de mise à jour et de suppression sur le conteneur ou les éléments stockés dans le conteneur.
<a href="#">PartitionKey</a>	Cette classe représente une clé de partition logique. Cette classe est nécessaire pour de nombreuses opérations et requêtes courantes.

## Exemples de code

- [Authentifier le client](#)
- [Obtenir une base de données](#)
- [Obtenir un conteneur](#)
- [Créer un élément](#)
- [Obtenir un élément](#)
- [Éléments de requête](#)

L'exemple de code du modèle utilise une base de données nommée `cosmicworks` et un conteneur nommé `products`. Le conteneur `products` contient des détails tels que le nom, la catégorie, la quantité, un identificateur unique et un indicateur de vente pour chaque produit. Le conteneur utilise la propriété `/category` en tant que clé de partition logique.

## Authentifier le client

Les requêtes d'application vers les Services Azure doivent être autorisées. Utilisez le type `DefaultAzureCredential` en tant que méthode privilégiée pour implémenter une connexion sans mot de passe entre vos applications et Azure Cosmos DB for NoSQL. `DefaultAzureCredential` prend en charge plusieurs méthodes d'authentification et détermine quelle méthode doit être utilisée au moment de l'exécution.

### ⓘ Important

Vous pouvez également autoriser directement les requêtes adressées aux services Azure à l'aide de mots de passe, de chaînes de connexion ou d'autres

informations d'identification. Toutefois, cette approche doit être utilisée avec prudence. Les développeurs doivent être vigilants pour ne jamais exposer les secrets dans un emplacement non sécurisé. Toute personne ayant accès au mot de passe ou à la clé secrète peut s'authentifier auprès du service de base de données. `DefaultAzureCredential` offre des avantages améliorés aux niveaux de la gestion et de la sécurité par rapport à la clé de compte pour permettre une authentification sans mot de passe sans les risques liés au stockage des clés.

Cet exemple crée une instance du type `CosmosClient`, et l'authentifie à l'aide d'une instance de `DefaultAzureCredential`.

Python

```
credential = DefaultAzureCredential()
client = CosmosClient(url=endpoint, credential=credential)
```

## Obtenir une base de données

Utilisez `client.get_database_client` pour récupérer la base de données existante nommée `cosmicworks`.

Python

```
database = client.get_database_client("cosmicworks")
```

## Obtenir un conteneur

Récupérez le conteneur `products` existant à l'aide de `database.get_container_client`.

Python

```
container = database.get_container_client("products")
```

## Créer un élément

Créez un objet avec tous les membres à sérialiser au format JSON. Dans cet exemple, le type a un identificateur unique et des champs pour la catégorie, le nom, la quantité, le prix et la vente. Créez un élément dans le conteneur à l'aide de `container.upsert_item`. Cette méthode fait un « upsert » de l'élément en le remplaçant de manière effective s'il existe déjà.

Python

```
new_item = {
 "id": "70b63682-b93a-4c77-aad2-65501347265f",
 "category": "gear-surf-surfboards",
 "name": "Yamba Surfboard",
 "quantity": 12,
 "sale": False,
}
created_item = container.upsert_item(new_item)
```

## Lire un élément

Effectuez une opération de lecture de point à l'aide des champs d'identificateur unique (`id`) et de clé de partition. Utilisez `container.read_item` pour récupérer efficacement l'élément spécifique.

Python

```
existing_item = container.read_item(
 item="70b63682-b93a-4c77-aad2-65501347265f",
 partition_key="gear-surf-surfboards",
)
```

## Éléments de requête

Effectuez une requête sur plusieurs éléments d'un conteneur à l'aide de `container.GetItemQueryIterator`. Recherchez tous les éléments d'une catégorie spécifique à l'aide de cette requête paramétrable :

nosql

```
SELECT * FROM products p WHERE p.category = @category
```

Python

```
queryText = "SELECT * FROM products p WHERE p.category = @category"
results = container.query_items(
 query=queryText,
 parameters=[
 dict(
 name="@category",
 value="gear-surf-surfboards",
)
],
```

```
 enable_cross_partition_query=False,
)
```

Parcourez les résultats de la requête dans une boucle.

Python

```
items = [item for item in results]
output = json.dumps(items, indent=True)
```

## Contenu connexe

- [Démarrage rapide de .NET](#)
- [Démarrage rapide de JavaScript/Node.js](#)
- [Démarrage rapide Java](#)
- [Démarrage rapide de Go](#)

## Étape suivante

[Package PyPI](#)

# Envoyer des événements à des hubs d'événements ou en recevoir de ces derniers à l'aide de Python

Article • 07/02/2024

Ce guide de démarrage rapide montre comment recevoir des événements d'un hub d'événements et lui en envoyer à l'aide du package Python [azure-eventhub](#).

## Prérequis

Si vous débutez avec Azure Event Hubs, consultez la [vue d'ensemble d'Event Hubs](#) avant de suivre ce guide de démarrage rapide.

Pour effectuer ce démarrage rapide, vous avez besoin de ce qui suit :

- **Abonnement Microsoft Azure.** Pour utiliser les services Azure, y compris Azure Event Hubs, vous avez besoin d'un abonnement. Si vous n'avez pas de compte Azure existant, inscrivez-vous pour obtenir un [essai gratuit](#).
- Python 3.8 ou version ultérieure, avec pip installé et à jour.
- Visual Studio Code (recommandé) ou tout autre environnement de développement intégré (IDE).
- **Créez un espace de noms Event Hubs et un Event Hub.** La première étape consiste à utiliser le [portail Azure](#) pour créer un espace de noms de type Event Hubs et obtenir les informations de gestion nécessaires à votre application pour communiquer avec le concentrateur d'événements. Pour créer un espace de noms et un hub d'événements, suivez la procédure décrite dans [cet article](#).

## Installer les packages pour envoyer des événements

Pour installer les packages Python pour Event Hubs, ouvrez une invite de commandes qui a Python dans son chemin. Remplacez le répertoire par le dossier dans lequel vous souhaitez conserver vos exemples.

Sans mot de passe (recommandé)

shell

```
pip install azure-eventhub
pip install azure-identity
```

```
pip install aiohttp
```

## Authentifier l'application sur Azure

Ce guide de démarrage rapide présente deux façons de vous connecter à Azure Event Hubs : sans mot de passe et avec une chaîne de connexion. La première option vous explique comment utiliser votre principal de sécurité dans Microsoft Entra ID et le contrôle d'accès en fonction du rôle (RBAC) pour vous connecter à un espace de noms Event Hubs. Vous n'avez pas à vous soucier d'avoir des chaînes de connexion codées en dur dans votre code, dans un fichier config ni dans un stockage sécurisé comme Azure Key Vault. La deuxième option consiste à se servir d'une chaîne de connexion pour se connecter à un espace de noms Event Hubs. Si vous débutez avec Azure, vous trouverez peut-être l'option de chaîne de connexion plus facile à suivre. Nous vous recommandons d'utiliser l'option sans mot de passe dans les applications réelles et les environnements de production. Pour plus d'informations, consultez [Authentification et autorisation](#). Pour en savoir plus sur l'authentification sans mot de passe, reportez-vous à la [page de présentation](#).

Sans mot de passe (recommandé)

## Attribuer des rôles à votre utilisateur Microsoft Entra

Lors du développement localement, vous devez vérifier que le compte d'utilisateur qui se connecte à Azure Event Hubs dispose des autorisations appropriées. Vous aurez besoin du rôle [Propriétaire de données Azure Event Hubs](#) pour envoyer et recevoir des messages. Pour vous attribuer ce rôle, vous aurez besoin du rôle Administrateur de l'accès utilisateur ou d'un autre rôle qui inclut l'action

`Microsoft.Authorization/roleAssignments/write`. Vous pouvez attribuer des rôles RBAC Azure à un utilisateur à l'aide du Portail Azure, Azure CLI ou Azure PowerShell. Découvrez les étendues disponibles pour les attributions de rôles dans la page [vue d'ensemble de l'étendue](#).

L'exemple suivant attribue le rôle `Azure Event Hubs Data Owner` à votre compte d'utilisateur, qui fournit un accès complet aux ressources Azure Event Hubs. Dans un scénario réel, suivez le [principe des priviléges minimum](#) pour accorder aux utilisateurs uniquement les autorisations minimales nécessaires à un environnement de production plus sécurisé.

## Rôles intégrés Azure pour Azure Event Hubs

Pour Azure Event Hubs, la gestion des espaces de noms et de toutes les ressources associées via le portail Azure et l'API de gestion des ressources Azure est déjà protégée à l'aide du modèle RBAC Azure. Azure fournit les rôles Azure intégrés ci-dessous pour autoriser l'accès à un espace de noms Event Hubs :

- [Propriétaire de données Azure Event Hubs](#) : permet l'accès aux données d'un espace de noms Event Hubs et de ses entités (files d'attente, rubriques, abonnements et filtres).
- [Expéditeur de données Azure Event Hubs](#) : utilisez ce rôle pour autoriser l'expéditeur à accéder à l'espace de noms Event Hubs et à ses entités.
- [Récepteur de données Azure Event Hubs](#) : utilisez ce rôle pour autoriser l'expéditeur à accéder à l'espace de noms Event Hubs et à ses entités.

Si vous souhaitez créer un rôle personnalisé, consultez [Droits requis pour les opérations Service Bus](#).

### Important

Dans la plupart des cas, la propagation de l'attribution de rôle dans Azure peut prendre une ou deux minutes. Dans de rares cas, cela peut prendre jusqu'à huit minutes. Si vous recevez des erreurs d'authentification lorsque vous exécutez votre code pour la première fois, patientez quelques instants et réessayez.

### Azure portal

1. Dans le portail Azure, recherchez votre espace de noms Event Hubs à l'aide de la barre de recherche principale ou de la navigation gauche.
2. Dans la page vue d'ensemble, sélectionnez **Contrôle d'accès (IAM)** dans le menu de gauche.
3. Sur la page **Contrôle d'accès (IAM)**, sélectionnez l'onglet **Attributions de rôles**.
4. Sélectionnez **+ Ajouter** dans le menu supérieur, puis **Ajouter une attribution de rôle** dans le menu déroulant résultant.

The screenshot shows the Azure Service Bus Namespace Access control (IAM) page. The left sidebar has a red box around the 'Access control (IAM)' item. A yellow arrow labeled 1 points to the 'Check access' button. A tooltip for 'Add role assignment' has a yellow arrow labeled 2 pointing to it. Another tooltip for 'Add role assignment' has a yellow arrow labeled 3 pointing to it.

5. Utilisez la zone de recherche pour filtrer les résultats sur le rôle souhaité.

Pour cet exemple, recherchez **Azure Event Hubs Data Owner** et sélectionnez le résultat correspondant. Ensuite, choisissez **Suivant**.

6. Sous **Attribuer l'accès à**, sélectionnez **Utilisateur, groupe ou principal de service**, puis sélectionnez **+ Sélectionner des membres**.

7. Dans la boîte de dialogue, recherchez votre nom d'utilisateur Microsoft Entra (généralement votre adresse e-mail *utilisateur@domaine*), puis choisissez **Sélectionner** en bas de la boîte de dialogue.

8. Sélectionnez **Vérifier + affecter** pour accéder à la page finale, puis **Vérifier + attribuer** à nouveau pour terminer le processus.

## Envoyer des événements

Dans cette section, vous allez créer un script Python pour envoyer des événements au hub d'événements que vous avez créé.

- Ouvrez votre éditeur Python favori, tel que [Visual Studio Code](#).
- Créez un script appelé *send.py*. Ce script envoie un lot d'événements au hub d'événements que vous avez créé.
- Collez le code suivant dans *send.py* :

Sans mot de passe (recommandé)

Dans le code, utilisez des valeurs réelles pour remplacer les espaces réservés suivants :

- EVENT\_HUB\_FULLY\_QUALIFIED\_NAMESPACE
- EVENT\_HUB\_NAME

Python

```
import asyncio

from azure.eventhub import EventData
from azure.eventhub.aio import EventHubProducerClient
from azure.identity.aio import DefaultAzureCredential

EVENT_HUB_FULLY_QUALIFIED_NAMESPACE =
"EVENT_HUB_FULLY_QUALIFIED_NAMESPACE"
EVENT_HUB_NAME = "EVENT_HUB_NAME"

credential = DefaultAzureCredential()

async def run():
 # Create a producer client to send messages to the event hub.
 # Specify a credential that has correct role assigned to access
 # event hubs namespace and the event hub name.
 producer = EventHubProducerClient(
 fully_qualified_namespace=EVENT_HUB_FULLY_QUALIFIED_NAMESPACE,
 eventhub_name=EVENT_HUB_NAME,
 credential=credential,
)
 async with producer:
 # Create a batch.
 event_data_batch = await producer.create_batch()

 # Add events to the batch.
 event_data_batch.add(EventData("First event "))
 event_data_batch.add(EventData("Second event"))
 event_data_batch.add(EventData("Third event"))

 # Send the batch of events to the event hub.
 await producer.send_batch(event_data_batch)

 # Close credential when no longer needed.
 await credential.close()

asyncio.run(run())
```

## ⓘ Notes

Pour obtenir des exemples d'autres options d'envoi d'événements à Event Hub de manière asynchrone à l'aide d'une chaîne de connexion, consultez la [page GitHub send\\_async.py ↗](#). Les modèles indiqués s'appliquent également à l'envoi d'événements sans mot de passe.

# Recevoir des événements

Ce guide de démarrage rapide utilise un stockage Blob Azure comme magasin de points de contrôle. Le magasin de points de contrôle est utilisé pour conserver les points de contrôle (autrement dit, les dernières positions de lecture).

Suivez les recommandations ci-dessous quand vous utilisez Stockage Blob Azure comme magasin de points de contrôle :

- Utilisez un conteneur distinct pour chaque groupe de processeurs. Vous pouvez utiliser le même compte de stockage, mais utiliser un conteneur par groupe.
- N'utilisez pas le conteneur et le compte de stockage pour quoi que ce soit d'autre.
- Le compte de stockage doit se trouver dans la même région que l'application déployée. Si l'application est locale, essayez de choisir la région la plus proche possible.

Sur la page **Compte de stockage** du Portail Azure, dans la section **Service BLOB**, vérifiez que les paramètres suivants sont désactivés.

- Espace de noms hiérarchique
- Suppression réversible de blob
- Contrôle de version

## Créer un compte de stockage Azure et un conteneur d'objets blob

Créez un compte de stockage Azure et un conteneur d'objets blob dans celui-ci en effectuant les étapes suivantes :

1. [Création d'un compte de stockage Azure](#)
2. [Créez un conteneur d'objets blob.](#)
3. S'authentifier auprès du conteneur d'objets blob.

Veillez à enregistrer la chaîne de connexion et le nom du conteneur en vue de les utiliser dans le code de réception.

#### Sans mot de passe (recommandé)

Lors du développement local, assurez-vous que le compte d'utilisateur qui accède aux données blob dispose des autorisations appropriées. Vous aurez besoin du **Contributeur aux données Blob de stockage** pour lire et écrire des données blob. Pour vous attribuer ce rôle, vous aurez besoin du rôle **Administrateur de l'accès utilisateur** ou d'un autre rôle qui inclut l'action **Microsoft.Authorization/roleAssignments/write**. Vous pouvez attribuer des rôles RBAC Azure à un utilisateur à l'aide du Portail Azure, Azure CLI ou Azure PowerShell. Vous pouvez en savoir plus sur les étendues disponibles pour les attributions de rôles dans la page [vue d'ensemble de l'étendue](#).

Dans ce scénario, vous allez attribuer des autorisations à votre compte d'utilisateur, étendues au compte de stockage, pour suivre le [Principe des priviléges minimum](#). Cette pratique offre aux utilisateurs uniquement les autorisations minimales nécessaires et crée des environnements de production plus sécurisés.

L'exemple suivant affecte le rôle **Contributeur aux données Blob du stockage** à votre compte d'utilisateur, qui fournit à la fois un accès en lecture et en écriture aux données d'objet blob dans votre compte de stockage.

#### ⓘ Important

Dans la plupart des cas, la propagation de l'attribution de rôle dans Azure peut prendre une ou deux minutes, mais dans de rares cas, cela peut prendre jusqu'à huit minutes. Si vous recevez des erreurs d'authentification lorsque vous exécutez votre code pour la première fois, patientez quelques instants et réessayez.

#### Azure portal

1. Dans le Portail Azure, recherchez votre compte de stockage à l'aide de la barre de recherche principale ou de la navigation gauche.
2. Dans la page vue d'ensemble du compte de stockage, sélectionnez **Contrôle d'accès (IAM)** dans le menu de gauche.

3. Sur la page Contrôle d'accès (IAM), sélectionnez l'onglet Attributions de rôles.

4. Sélectionnez + Ajouter dans le menu supérieur, puis Ajouter une attribution de rôle dans le menu déroulant résultant.

The screenshot shows the 'Access Control (IAM)' blade for a storage account named 'identitymigrationstorage'. The left sidebar has a red box around the 'Access Control (IAM)' item. The main area has a red box around the '+ Add' button in the top navigation bar. A dropdown menu is open, showing 'Add role assignment' as the first option. Other options include 'Download role assignments', 'Edit columns', 'Refresh', 'Remove', and 'Got feedback?'. Below the '+ Add' button, there are sections for 'My access', 'Check access' (with radio buttons for 'User, group, or service principal' and 'Managed identity'), and a search bar. To the right, there are three cards: 'Grant access to this resource', 'View deny assignments', and another 'Add role assignment' card.

5. Utilisez la zone de recherche pour filtrer les résultats sur le rôle souhaité.

Pour cet exemple, recherchez *Contributeur aux données Blob du stockage*, sélectionnez le résultat correspondant, puis choisissez **Suivant**.

6. Sous **Attribuer l'accès à**, sélectionnez **Utilisateur, groupe ou principal de service**, puis sélectionnez + **Sélectionner des membres**.

7. Dans la boîte de dialogue, recherchez votre nom d'utilisateur Microsoft Entra (généralement votre adresse e-mail *utilisateur@domaine*), puis choisissez **Sélectionner** en bas de la boîte de dialogue.

8. Sélectionnez **Vérifier + affecter** pour accéder à la page finale, puis **Vérifier + attribuer** à nouveau pour terminer le processus.

## Installer les packages pour recevoir des événements

Pour le côté récepteur, vous avez besoin d'installer un autre package, ou plus. Dans ce guide de démarrage rapide, vous utilisez le stockage Blob Azure pour conserver les points de contrôle afin que le programme ne lise pas les événements qu'il a déjà lus. Il effectue des points de contrôle de métadonnées sur les messages reçus à intervalles réguliers dans un blob. Cette approche permet ultérieurement de continuer à recevoir des messages à partir du point où vous vous étiez arrêté.

Sans mot de passe (recommandé)

shell

```
pip install azure-eventhub-checkpointstoreblob-aio
pip install azure-identity
```

## Créer un script Python pour recevoir des événements

Dans cette section, vous allez créer un script Python pour recevoir des événements de votre Event Hub :

1. Ouvrez votre éditeur Python favori, tel que [Visual Studio Code](#).
2. Créez un script appelé *recv.py*.
3. Collez le code suivant dans *recv.py* :

Sans mot de passe (recommandé)

Dans le code, utilisez des valeurs réelles pour remplacer les espaces réservés suivants :

- BLOB\_STORAGE\_ACCOUNT\_URL
- BLOB\_CONTAINER\_NAME
- EVENT\_HUB\_FULLY\_QUALIFIED\_NAMESPACE
- EVENT\_HUB\_NAME

Python

```
import asyncio

from azure.eventhub.aio import EventHubConsumerClient
from azure.eventhub.extensions.checkpointstoreblobaio import (
 BlobCheckpointStore,
)
from azure.identity.aio import DefaultAzureCredential

BLOB_STORAGE_ACCOUNT_URL = "BLOB_STORAGE_ACCOUNT_URL"
BLOB_CONTAINER_NAME = "BLOB_CONTAINER_NAME"
EVENT_HUB_FULLY_QUALIFIED_NAMESPACE =
"EVENT_HUB_FULLY_QUALIFIED_NAMESPACE"
EVENT_HUB_NAME = "EVENT_HUB_NAME"

credential = DefaultAzureCredential()
```

```

async def on_event(partition_context, event):
 # Print the event data.
 print(
 'Received the event: "{}" from the partition with ID: "{}"'.format(
 event.body_as_str(encoding="UTF-8"),
 partition_context.partition_id
)
)

 # Update the checkpoint so that the program doesn't read the
 # events that it has already read when you run it next time.
 await partition_context.update_checkpoint(event)

async def main():
 # Create an Azure blob checkpoint store to store the
 # checkpoints.
 checkpoint_store = BlobCheckpointStore(
 blob_account_url=BLOB_STORAGE_ACCOUNT_URL,
 container_name=BLOB_CONTAINER_NAME,
 credential=credential,
)

 # Create a consumer client for the event hub.
 client = EventHubConsumerClient(
 fully_qualified_namespace=EVENT_HUB_FULLY_QUALIFIED_NAMESPACE,
 eventhub_name=EVENT_HUB_NAME,
 consumer_group="$Default",
 checkpoint_store=checkpoint_store,
 credential=credential,
)

 async with client:
 # Call the receive method. Read from the beginning of the
 # partition
 # (starting_position: "-1")
 await client.receive(on_event=on_event,
 starting_position="-1")

 # Close credential when no longer needed.
 await credential.close()

if __name__ == "__main__":
 # Run the main method.
 asyncio.run(main())

```

## ⓘ Notes

Pour obtenir des exemples d'autres options permettant de recevoir des événements d'Event Hub de manière asynchrone à l'aide d'une chaîne de connexion, consultez la [page GitHub recv\\_with\\_checkpoint\\_store\\_async.py](#). Les modèles indiqués s'appliquent également à la réception d'événements sans mot de passe.

## Exécuter l'application réceptrice

Pour exécuter le script, ouvrez une invite de commandes qui a Python dans son chemin, puis exécutez cette commande :

Bash

```
python recv.py
```

## Exécuter l'application de l'expéditeur

Pour exécuter le script, ouvrez une invite de commandes qui a Python dans son chemin, puis exécutez cette commande :

Bash

```
python send.py
```

La fenêtre réceptrice doit afficher les messages qui ont été envoyés au hub d'événements.

## Dépannage

Si vous ne voyez pas d'événements dans la fenêtre du récepteur ou si le code signale une erreur, essayez les conseils de dépannage suivants :

- Si vous ne voyez pas les résultats de `recv.py`, exécutez `send.py` plusieurs fois.
- Si vous voyez des erreurs concernant « coroutine » lors de l'utilisation du code sans mot de passe (avec des informations d'identification), vérifiez que vous utilisez l'importation à partir de `azure.identity.aio`.
- Si vous voyez « Session client non fermée » avec du code sans mot de passe (avec des informations d'identification), veillez à fermer les informations d'identification

lorsque vous avez terminé. Pour plus d'informations, consultez [Informations asynchrones](#).

- Si vous voyez des erreurs d'autorisation avec `recv.py` lors de l'accès au stockage, vérifiez que vous avez suivi les étapes décrites dans [Créer un compte de stockage Azure et un conteneur d'objets blob](#) et que vous avez attribué le rôle **Contributeur aux données blob de stockage** au principal de service.
- Si vous recevez des événements avec des ID de partition différents, ce résultat est attendu. Les partitions constituent un mécanisme d'organisation des données. Elles sont liées au degré de parallélisme en aval requis lors de la consommation des applications. Le choix du nombre de partitions dans un concentrateur d'événements est directement lié au nombre de lecteurs simultanés que vous prévoyez d'avoir. Pour plus d'informations, consultez [En savoir plus sur les partitions](#).

## Étapes suivantes

Dans ce guide de démarrage rapide, vous avez envoyé et reçu des événements de manière asynchrone. Pour savoir comment envoyer et recevoir des événements de manière synchrone, accédez à la [page GitHub sync\\_samples](#).

Pour tous les exemples (synchrone comme asynchrones) sur GitHub, accédez à la [bibliothèque cliente Azure Event Hubs des exemples Python](#).

# Démarrage rapide : Bibliothèque de client de certificats Azure Key Vault pour Python

Article • 16/03/2023

Bien démarrer avec la bibliothèque de client de certificats Azure Key Vault pour Python. Suivez les étapes suivantes pour installer le package et essayer un exemple de code pour les tâches de base. En stockant des certificats à l'aide de Key Vault, vous évitez de les stocker dans votre code, ce qui renforce la sécurité de votre application.

[Documentation de référence de l'API](#) | [Code source bibliothèque](#) | [Package \(Index package Python\)](#)

## Prérequis

- Un abonnement Azure - [En créer un gratuitement](#)
- [Python 3.7+](#)
- [Azure CLI](#)

Ce démarrage rapide suppose que vous exécutez [Azure CLI](#) ou [Azure PowerShell](#) dans une fenêtre de terminal Linux.

## Configurer votre environnement local

Ce guide de démarrage rapide utilise la bibliothèque Azure Identity ainsi qu'Azure CLI ou PowerShell pour authentifier l'utilisateur auprès des services Azure. Les développeurs peuvent également utiliser Visual Studio ou Visual Studio Code pour authentifier leurs appels. Pour plus d'informations, consultez [Authentifier le client avec la bibliothèque cliente Azure Identity](#).

## Connexion à Azure

Azure CLI

1. Exécutez la commande `login`.

Azure CLI

```
az login
```

Si l'interface CLI peut ouvrir votre navigateur par défaut, elle le fait et charge une page de connexion Azure par la même occasion.

Sinon, ouvrez une page de navigateur à l'adresse <https://aka.ms/devicelogin> et entrez le code d'autorisation affiché dans votre terminal.

2. Dans le navigateur, connectez-vous avec les informations d'identification de votre compte.

## Installer les packages

1. Dans un terminal ou une invite de commandes, créez un dossier de projet approprié, puis créez et activez un environnement virtuel Python comme décrit dans [Utiliser des environnements virtuels Python](#)
2. Installez la bibliothèque d'identités Microsoft Entra :

terminal

```
pip install azure.identity
```

3. Installez la bibliothèque de client de certificats Key Vault :

terminal

```
pip install azure-keyvault-certificates
```

## Créer un groupe de ressources et un coffre de clés

Azure CLI

1. Utilisez la commande `az group create` pour créer un groupe de ressources :

Azure CLI

```
az group create --name myResourceGroup --location eastus
```

Si vous préférez, vous pouvez remplacer « eastus » par un emplacement plus proche de vous.

2. Utilisez `az keyvault create` pour créer le coffre de clés :

Azure CLI

```
az keyvault create --name <your-unique-keyvault-name> --resource-group myResourceGroup
```

Remplacez `<your-unique-keyvault-name>` par un nom unique à l'échelle d'Azure. La pratique courante consiste à utiliser son nom ou le nom de son entreprise et à ajouter des chiffres ou des identificateurs.

## Définir la variable d'environnement KEY\_VAULT\_NAME

Notre script utilise la valeur attribuée à la variable d'environnement `KEY_VAULT_NAME` comme nom du coffre de clés. Vous devez donc définir cette valeur à l'aide de la commande suivante :

Console

```
export KEY_VAULT_NAME=<your-unique-keyvault-name>
```

## Accorder l'accès à votre coffre de clés

Créer une stratégie d'accès pour votre coffre de clés qui accorde une autorisation de certificat à votre compte d'utilisateur

Azure CLI

Azure CLI

```
az keyvault set-policy --name <your-unique-keyvault-name> --upn user@domain.com --certificate-permissions delete get list create
```

## Créer l'exemple de code

La bibliothèque de client de certificats Azure Key Vault pour Python vous permet de gérer les certificats. L'exemple de code suivant montre comment créer un client et comment définir, récupérer et supprimer un certificat.

Créez un fichier nommé *kv\_certificates.py* qui contient ce code.

Python

```
import os
from azure.keyvault.certificates import CertificateClient, CertificatePolicy
from azure.identity import DefaultAzureCredential

keyVaultName = os.environ["KEY_VAULT_NAME"]
KVUri = "https://" + keyVaultName + ".vault.azure.net"

credential = DefaultAzureCredential()
client = CertificateClient(vault_url=KVUri, credential=credential)

certificateName = input("Input a name for your certificate > ")

print(f"Creating a certificate in {keyVaultName} called '{certificateName}'\n...")

policy = CertificatePolicy.get_default()
poller = client.begin_create_certificate(certificate_name=certificateName,
policy=policy)
certificate = poller.result()

print(" done.")

print(f"Retrieving your certificate from {keyVaultName}.")

retrieved_certificate = client.get_certificate(certificateName)

print(f"Certificate with name '{retrieved_certificate.name}' was found'.")
print(f"Deleting your certificate from {keyVaultName} ...")

poller = client.begin_delete_certificate(certificateName)
deleted_certificate = poller.result()

print(" done.")
```

## Exécuter le code

Vérifiez que le code de la section précédente se trouve dans un fichier nommé *kv\_certificates.py*. Exécutez ensuite le code avec la commande suivante :

terminal

```
python kv_certificates.py
```

- Si vous rencontrez des erreurs d'autorisation, vérifiez que vous avez exécuté la commande [az keyvault set-policy](#) ou [Set-AzKeyVaultAccessPolicy](#).
- Le fait de réexécuter le code avec le même nom de clé peut produire l'erreur suivante : « (Conflit) Le certificat <name> est actuellement à l'état supprimé, mais récupérable. » Utilisez un nom de clé différent.

## Détails du code

### Authentifier et créer un client

Les requêtes d'application vers les Services Azure doivent être autorisées. L'utilisation de la classe [DefaultAzureCredential](#) fournie par la [bibliothèque de client Azure Identity](#) est l'approche recommandée pour implémenter des connexions sans mot de passe aux services Azure dans votre code. `DefaultAzureCredential` prend en charge plusieurs méthodes d'authentification et détermine quelle méthode doit être utilisée au moment de l'exécution. Cette approche permet à votre application d'utiliser différentes méthodes d'authentification dans différents environnements (local ou production) sans implémenter de code spécifique à l'environnement.

Dans ce guide de démarrage rapide, `DefaultAzureCredential` s'authentifie auprès du coffre de clés à l'aide des informations d'identification de l'utilisateur de développement local connecté à Azure CLI. Quand l'application est déployée sur Azure, le même code `DefaultAzureCredential` peut découvrir et utiliser automatiquement une identité managée affectée à un service d'application, une machine virtuelle ou d'autres services. Pour plus d'informations, consultez [Vue d'ensemble des identités managées](#).

Dans l'exemple ci-dessous, le nom de votre coffre de clés est étendu à l'URL du coffre de clés, au format `https://\<your-key-vault-name>.vault.azure.net`.

Python

```
credential = DefaultAzureCredential()
client = CertificateClient(vault_url=KVUri, credential=credential)
```

## Enregistrer un certificat

Une fois que vous avez obtenu l'objet client pour le coffre de clés, vous pouvez créer un certificat en utilisant la méthode [begin\\_create\\_certificate](#) :

Python

```
policy = CertificatePolicy.get_default()
poller = client.begin_create_certificate(certificate_name=certificateName,
policy=policy)
certificate = poller.result()
```

Ici, le certificat a besoin d'une stratégie obtenue avec la méthode [CertificatePolicy.get\\_default](#).

L'appel d'une méthode `begin_create_certificate` génère un appel asynchrone à l'API REST Azure pour le coffre de clés. L'appel asynchrone retourne un objet observateur. Pour attendre le résultat de l'opération, appelez la méthode `result` de l'observateur.

Au moment de traiter la requête, Azure authentifie l'identité de l'appelant (le principal du service) à partir de l'objet d'informations d'identification que vous avez fourni au client.

## Récupérer un certificat

Pour lire un certificat à partir de Key Vault, utilisez la méthode [get\\_certificate](#) :

Python

```
retrieved_certificate = client.get_certificate(certificateName)
```

Vous pouvez aussi vérifier que le certificat a été défini à l'aide de la commande Azure CLI [az keyvault certificate show](#) ou de la cmdlet Azure PowerShell [Get-AzKeyVaultCertificate](#)

## Supprimer un certificat

Pour supprimer un certificat, utilisez la méthode [begin\\_delete\\_certificate](#) :

Python

```
poller = client.begin_delete_certificate(certificateName)
deleted_certificate = poller.result()
```

La méthode `begin_delete_certificate` est asynchrone et retourne un objet observateur. L'appel de la méthode `result` de l'observateur attend la fin de son exécution.

Vous pouvez vérifier que le certificat a été supprimé à l'aide de la commande Azure CLI [az keyvault certificate show](#) ou de la cmdlet Azure PowerShell [Get-AzKeyVaultCertificatey](#).

Une fois supprimé, un certificat reste à l'état supprimé mais récupérable pour un temps. Si vous réexécutez le code, utilisez un nom de certificat différent.

## Nettoyer les ressources

Si vous voulez aussi tenter une expérience avec des [secrets](#) et des [clés](#), vous pouvez réutiliser le coffre de clés créé dans cet article.

Sinon, quand vous en avez terminé avec les ressources créées dans cet article, utilisez la commande suivante pour supprimer le groupe de ressources et toutes les ressources qu'il contient :

Azure CLI

Azure CLI

```
az group delete --resource-group myResourceGroup
```

## Étapes suivantes

- [Vue d'ensemble Azure Key Vault](#)
- [Sécuriser l'accès à un coffre de clés](#)
- [Guide du développeur Azure Key Vault](#)
- [Vue d'ensemble de la sécurité de Key Vault](#)
- [S'authentifier auprès de Key Vault](#)

# Démarrage rapide : Bibliothèque cliente des clés Azure Key Vault pour Python

Article • 16/03/2023

Bien démarrer avec la bibliothèque de client Azure Key Vault pour Python. Suivez les étapes suivantes pour installer le package et essayer un exemple de code pour les tâches de base. En stockant des clés de chiffrement à l'aide de Key Vault, vous évitez de les stocker dans votre code, ce qui renforce la sécurité de votre application.

[Documentation de référence de l'API](#) | [Code source bibliothèqueC ↗](#) | [Package \(Index package Python\) ↗](#)

## Prérequis

- Un abonnement Azure - [En créer un gratuitement↗](#)
- [Python 3.7+](#)
- [Azure CLI](#)

Ce démarrage rapide suppose que vous exécutez [Azure CLI](#) ou [Azure PowerShell](#) dans une fenêtre de terminal Linux.

## Configurer votre environnement local

Ce guide de démarrage rapide utilise la bibliothèque Azure Identity avec Azure CLI ou Azure PowerShell pour authentifier l'utilisateur auprès des services Azure. Les développeurs peuvent également utiliser Visual Studio ou Visual Studio Code pour authentifier leurs appels. Pour plus d'informations, consultez [Authentifier le client avec la bibliothèque cliente Azure Identity](#).

## Connexion à Azure

Azure CLI

1. Exécutez la commande `login`.

Azure CLI

```
az login
```

Si l'interface CLI peut ouvrir votre navigateur par défaut, elle le fait et charge une page de connexion Azure par la même occasion.

Sinon, ouvrez une page de navigateur à l'adresse <https://aka.ms/devicelogin> et entrez le code d'autorisation affiché dans votre terminal.

2. Dans le navigateur, connectez-vous avec les informations d'identification de votre compte.

## Installer les packages

1. Dans un terminal ou une invite de commandes, créez un dossier de projet approprié, puis créez et activez un environnement virtuel Python comme décrit dans [Utiliser des environnements virtuels Python](#).
2. Installez la bibliothèque d'identités Microsoft Entra :

```
terminal
pip install azure-identity
```

3. Installez la bibliothèque de client de clés Key Vault :

```
terminal
pip install azure-keyvault-keys
```

## Créer un groupe de ressources et un coffre de clés

Azure CLI

1. Utilisez la commande `az group create` pour créer un groupe de ressources :

```
Azure CLI
az group create --name myResourceGroup --location eastus
```

Si vous préférez, vous pouvez remplacer « eastus » par un emplacement plus proche de vous.

2. Utilisez `az keyvault create` pour créer le coffre de clés :

Azure CLI

```
az keyvault create --name <your-unique-keyvault-name> --resource-group myResourceGroup
```

Remplacez `<your-unique-keyvault-name>` par un nom unique à l'échelle d'Azure. La pratique courante consiste à utiliser son nom ou le nom de son entreprise et à ajouter des chiffres ou des identificateurs.

## Définir la variable d'environnement KEY\_VAULT\_NAME

Notre script utilise la valeur attribuée à la variable d'environnement `KEY_VAULT_NAME` comme nom du coffre de clés. Vous devez donc définir cette valeur à l'aide de la commande suivante :

Console

```
export KEY_VAULT_NAME=<your-unique-keyvault-name>
```

## Accorder l'accès à votre coffre de clés

Créez une stratégie d'accès pour votre coffre de clés qui accorde une autorisation de clé à votre compte d'utilisateur.

Azure CLI

Azure CLI

```
az keyvault set-policy --name <your-unique-keyvault-name> --upn user@domain.com --key-permissions get list create delete
```

## Créer l'exemple de code

La bibliothèque de client de clés Azure Key Vault pour Python vous permet de gérer des clés de chiffrement. L'exemple de code suivant vous montre comment créer un client et définir, récupérer et supprimer une clé.

Créez un fichier nommé *kv\_keys.py* qui contient ce code.

Python

```
import os
from azure.keyvault.keys import KeyClient
from azure.identity import DefaultAzureCredential

keyVaultName = os.environ["KEY_VAULT_NAME"]
KVUri = "https://" + keyVaultName + ".vault.azure.net"

credential = DefaultAzureCredential()
client = KeyClient(vault_url=KVUri, credential=credential)

keyName = input("Input a name for your key > ")

print(f"Creating a key in {keyVaultName} called '{keyName}' ...")

rsa_key = client.create_rsa_key(keyName, size=2048)

print(" done.")

print(f"Retrieving your key from {keyVaultName}.")

retrieved_key = client.get_key(keyName)

print(f"Key with name '{retrieved_key.name}' was found.")
print(f"Deleting your key from {keyVaultName} ...")

poller = client.begin_delete_key(keyName)
deleted_key = poller.result()

print(" done.")
```

## Exécuter le code

Vérifiez que le code de la section précédente se trouve dans un fichier nommé *kv\_keys.py*. Exécutez ensuite le code avec la commande suivante :

terminal

```
python kv_keys.py
```

- Si vous rencontrez des erreurs d'autorisation, vérifiez que vous avez exécuté la commande [az keyvault set-policy](#) ou [Set-AzKeyVaultAccessPolicy](#).
- Le fait de réexécuter le code avec le même nom de clé a pour effet de générer l'erreur : « La clé (en conflit) <name> est actuellement à l'état supprimé mais récupérable. » Utilisez un autre nom de clé.

# Détails du code

## Authentifier et créer un client

Les requêtes d'application vers les Services Azure doivent être autorisées. L'utilisation de la classe `DefaultAzureCredential` fournie par la [bibliothèque de client Azure Identity](#) est l'approche recommandée pour implémenter des connexions sans mot de passe aux services Azure dans votre code. `DefaultAzureCredential` prend en charge plusieurs méthodes d'authentification et détermine quelle méthode doit être utilisée au moment de l'exécution. Cette approche permet à votre application d'utiliser différentes méthodes d'authentification dans différents environnements (local ou production) sans implémenter de code spécifique à l'environnement.

Dans ce guide de démarrage rapide, `DefaultAzureCredential` s'authentifie auprès du coffre de clés à l'aide des informations d'identification de l'utilisateur de développement local connecté à Azure CLI. Quand l'application est déployée sur Azure, le même code `DefaultAzureCredential` peut découvrir et utiliser automatiquement une identité managée affectée à un service d'application, une machine virtuelle ou d'autres services. Pour plus d'informations, consultez [Vue d'ensemble des identités managées](#).

Dans l'exemple, le nom de votre coffre de clés est développé à l'aide de la valeur de la variable `kvUri`, au format : « `https://<your-key-vault-name>.vault.azure.net` ».

Python

```
credential = DefaultAzureCredential()
client = KeyClient(vault_url=KVUri, credential=credential)
```

## Enregistrer une clé

Une fois que vous avez obtenu l'objet client pour le coffre de clés, vous pouvez stocker une clé à l'aide de la méthode `create_rsa_key` :

Python

```
rsa_key = client.create_rsa_key(keyName, size=2048)
```

Vous pouvez aussi utiliser `create_key` ou `create_ec_key`.

L'appel d'une méthode `create` génère un appel à l'API REST Azure pour le coffre de clés.

Au moment de traiter la requête, Azure authentifie l'identité de l'appelant (le principal du service) à partir de l'objet d'informations d'identification que vous avez fourni au client.

## Récupérer une clé

Pour lire une clé à partir de Key Vault, utilisez la méthode [get\\_key](#) :

Python

```
retrieved_key = client.get_key(keyName)
```

Vous pouvez aussi vérifier que la clé a été définie à l'aide de la commande Azure CLI [az keyvault key show](#) ou de la cmdlet Azure PowerShell [Get-AzKeyVaultKey](#).

## Supprimer une clé

Pour supprimer une clé, utilisez la méthode [begin\\_delete\\_key](#) :

Python

```
poller = client.begin_delete_key(keyName)
deleted_key = poller.result()
```

La méthode `begin_delete_key` est asynchrone et retourne un objet observateur. L'appel de la méthode `result` de l'observateur attend la fin de son exécution.

Vous pouvez vérifier que la clé a été supprimée à l'aide de la commande Azure CLI [az keyvault key show](#) ou de la cmdlet Azure PowerShell [Get-AzKeyVaultKey](#).

Une fois supprimée, une clé reste à l'état supprimé mais récupérable pour un temps. Si vous réexécutez le code, utilisez un nom de clé différent.

## Nettoyer les ressources

Si vous voulez aussi tenter une expérience avec des [certificats](#) et des [secrets](#), vous pouvez réutiliser le coffre de clés créé dans cet article.

Sinon, quand vous en avez terminé avec les ressources créées dans cet article, utilisez la commande suivante pour supprimer le groupe de ressources et toutes les ressources qu'il contient :

Azure CLI

Azure CLI

```
az group delete --resource-group myResourceGroup
```

## Étapes suivantes

- [Vue d'ensemble Azure Key Vault](#)
- [Sécuriser l'accès à un coffre de clés](#)
- [Guide du développeur Azure Key Vault](#)
- [Vue d'ensemble de la sécurité de Key Vault](#)
- [S'authentifier auprès de Key Vault](#)

# Démarrage rapide : Bibliothèque de client de secrets Azure Key Vault pour Python

Article • 16/03/2023

Bien démarrer avec la bibliothèque de client de secrets Azure Key Vault pour Python. Suivez les étapes suivantes pour installer le package et essayer un exemple de code pour les tâches de base. En stockant des secrets à l'aide de Key Vault, vous évitez de les stocker dans votre code, ce qui renforce la sécurité de votre application.

[Documentation de référence de l'API](#) | [Code source bibliothèque](#) | [Package \(Index package Python\)](#)

## Prérequis

- Un abonnement Azure - [En créer un gratuitement](#)
- [Python 3.7+](#).
- [Azure CLI](#) ou [Azure PowerShell](#).

Ce démarrage rapide suppose que vous exécutez [Azure CLI](#) ou [Azure PowerShell](#) dans une fenêtre de terminal Linux.

## Configurer votre environnement local

Ce guide de démarrage rapide utilise la bibliothèque Azure Identity avec Azure CLI ou Azure PowerShell pour authentifier l'utilisateur auprès des services Azure. Les développeurs peuvent également utiliser Visual Studio ou Visual Studio Code pour authentifier leurs appels. Pour plus d'informations, consultez [Authentifier le client avec la bibliothèque de client Azure Identity](#).

## Connexion à Azure

Azure CLI

1. Exécutez la commande `az login`.

Azure CLI

```
az login
```

Si l'interface CLI peut ouvrir votre navigateur par défaut, elle le fait et charge une page de connexion Azure par la même occasion.

Sinon, ouvrez une page de navigateur à l'adresse <https://aka.ms/devicelogin> et entrez le code d'autorisation affiché dans votre terminal.

2. Dans le navigateur, connectez-vous avec les informations d'identification de votre compte.

## Installer les packages

1. Dans un terminal ou une invite de commandes, créez un dossier de projet approprié, puis créez et activez un environnement virtuel Python comme décrit dans [Utiliser des environnements virtuels Python](#).
2. Installez la bibliothèque d'identités Microsoft Entra :

terminal

```
pip install azure-identity
```

3. Installez la bibliothèque de secrets Key Vault :

terminal

```
pip install azure-keyvault-secrets
```

## Créer un groupe de ressources et un coffre de clés

Azure CLI

1. Utilisez la commande `az group create` pour créer un groupe de ressources :

Azure CLI

```
az group create --name myResourceGroup --location eastus
```

Si vous préférez, vous pouvez remplacer « eastus » par un emplacement plus proche de vous.

2. Utilisez `az keyvault create` pour créer le coffre de clés :

Azure CLI

```
az keyvault create --name <your-unique-keyvault-name> --resource-group myResourceGroup
```

Remplacez `<your-unique-keyvault-name>` par un nom unique à l'échelle d'Azure. La pratique courante consiste à utiliser son nom ou le nom de son entreprise et à ajouter des chiffres ou des identificateurs.

## Définir la variable d'environnement KEY\_VAULT\_NAME

Notre script utilise la valeur attribuée à la variable d'environnement `KEY_VAULT_NAME` comme nom du coffre de clés. Vous devez donc définir cette valeur à l'aide de la commande suivante :

Console

```
export KEY_VAULT_NAME=<your-unique-keyvault-name>
```

## Accorder l'accès à votre coffre de clés

Créez une stratégie d'accès pour votre coffre de clés qui accorde une autorisation de secret à votre compte d'utilisateur.

Azure CLI

Azure CLI

```
az keyvault set-policy --name <your-unique-keyvault-name> --upn user@domain.com --secret-permissions delete get list set
```

## Créer l'exemple de code

La bibliothèque de client de secrets Azure Key Vault pour Python vous permet de gérer des secrets. L'exemple de code suivant vous montre comment créer un client et définir, récupérer et supprimer un secret.

Créez un fichier nommé `kv_secrets.py` qui contient ce code.

Python

```
import os
from azure.keyvault.secrets import SecretClient
from azure.identity import DefaultAzureCredential

keyVaultName = os.environ["KEY_VAULT_NAME"]
KVUri = f"https://{{keyVaultName}}.vault.azure.net"

credential = DefaultAzureCredential()
client = SecretClient(vault_url=KVUri, credential=credential)

secretName = input("Input a name for your secret > ")
secretValue = input("Input a value for your secret > ")

print(f"Creating a secret in {keyVaultName} called '{secretName}' with the
 value '{secretValue}' ...")

client.set_secret(secretName, secretValue)

print(" done.")

print(f"Retrieving your secret from {keyVaultName}.")
retrieved_secret = client.get_secret(secretName)

print(f"Your secret is '{retrieved_secret.value}'.")
print(f"Deleting your secret from {keyVaultName} ...")

poller = client.begin_delete_secret(secretName)
deleted_secret = poller.result()

print(" done.")
```

## Exécuter le code

Vérifiez que le code de la section précédente se trouve dans un fichier nommé `kv_secrets.py`. Exécutez ensuite le code avec la commande suivante :

terminal

```
python kv_secrets.py
```

- Si vous rencontrez des erreurs d'autorisation, vérifiez que vous avez exécuté la commande [az keyvault set-policy](#) ou [Set-AzKeyVaultAccessPolicy](#).
- Le fait de réexécuter le code avec le même nom de secret a pour effet de générer l'erreur : « Le secret (en conflit) <name> est actuellement à l'état supprimé mais récupérable. » Utilisez un autre nom secret.

## Détails du code

### Authentifier et créer un client

Les requêtes d'application vers les Services Azure doivent être autorisées. L'utilisation de la classe [DefaultAzureCredential](#) fournie par la [bibliothèque de client Azure Identity](#) est l'approche recommandée pour implémenter des connexions sans mot de passe aux services Azure dans votre code. `DefaultAzureCredential` prend en charge plusieurs méthodes d'authentification et détermine quelle méthode doit être utilisée au moment de l'exécution. Cette approche permet à votre application d'utiliser différentes méthodes d'authentification dans différents environnements (local ou production) sans implémenter de code spécifique à l'environnement.

Dans ce guide de démarrage rapide, `DefaultAzureCredential` s'authentifie auprès du coffre de clés à l'aide des informations d'identification de l'utilisateur de développement local connecté à Azure CLI. Quand l'application est déployée sur Azure, le même code `DefaultAzureCredential` peut découvrir et utiliser automatiquement une identité managée affectée à un service d'application, une machine virtuelle ou d'autres services. Pour plus d'informations, consultez [Vue d'ensemble des identités managées](#).

Dans l'exemple, le nom de votre coffre de clés est développé à l'aide de la valeur de la variable `kvuri`, au format : « `https://<your-key-vault-name>.vault.azure.net` ».

Python

```
credential = DefaultAzureCredential()
client = SecretClient(vault_url=KVUri, credential=credential)
```

### Enregistrer un secret

Une fois que vous avez obtenu l'objet client pour le coffre de clés, vous pouvez stocker un secret à l'aide de la méthode `set_secret` :

Python

```
client.set_secret(secretName, secretValue)
```

L'appel de `set_secret` génère un appel à l'API REST Azure pour le coffre de clés.

Au moment de traiter la requête, Azure authentifie l'identité de l'appelant (le principal du service) à partir de l'objet d'informations d'identification que vous avez fourni au client.

## Récupérer un secret

Pour lire un secret à partir de Key Vault, utilisez la méthode `get_secret` :

Python

```
retrieved_secret = client.get_secret(secretName)
```

La valeur du secret est contenue dans `retrieved_secret.value`.

Vous pouvez également récupérer un secret avec la commande Azure CLI [az keyvault secret show](#) ou la cmdlet Azure PowerShell [Get-AzKeyVaultSecret](#).

## Supprimer un secret

Pour supprimer un secret, utilisez la méthode `begin_delete_secret` :

Python

```
poller = client.begin_delete_secret(secretName)
deleted_secret = poller.result()
```

La méthode `begin_delete_secret` est asynchrone et retourne un objet observateur.

L'appel de la méthode `result` de l'observateur attend la fin de son exécution.

Vous pouvez vérifier que le secret a été supprimé avec la commande Azure CLI [az keyvault secret show](#) ou la cmdlet Azure PowerShell [Get-AzKeyVaultSecret](#).

Une fois supprimé, un secret reste à l'état supprimé mais récupérable pour un temps. Si vous réexécutez le code, utilisez un nom de secret différent.

## Nettoyer les ressources

Si vous voulez aussi tenter une expérience avec des [certificats](#) et des [clés](#), vous pouvez réutiliser le coffre de clés créé dans cet article.

Sinon, quand vous en avez terminé avec les ressources créées dans cet article, utilisez la commande suivante pour supprimer le groupe de ressources et toutes les ressources qu'il contient :

```
Azure CLI
az group delete --resource-group myResourceGroup
```

## Étapes suivantes

- [Vue d'ensemble Azure Key Vault](#)
- [Guide du développeur Azure Key Vault](#)
- [Vue d'ensemble de la sécurité de Key Vault](#)
- [S'authentifier auprès de Key Vault](#)

# Envoyer et recevoir des messages à partir de files d'attente Azure Service Bus (Python)

Article • 06/02/2024

Dans ce tutoriel, vous allez effectuer les étapes suivantes :

1. Créer un espace de noms Service Bus à l'aide du Portail Azure.
2. Créer une file d'attente Service Bus à l'aide du portail Azure.
3. Écrivez du code Python afin d'utiliser le package [azure-servicebus](#) pour :
  - a. Envoyer un ensemble de messages à la file d'attente
  - b. Recevoir ces messages de la file d'attente.

## ⓘ Notes

Ce guide de démarrage rapide fournit des instructions pas à pas pour un scénario simple qui consiste à envoyer des messages à une file d'attente Service Bus et à les recevoir. Vous trouverez des exemples JavaScript et TypeScript prédéfinis pour Azure Service Bus dans le dépôt du **Kit de développement logiciel (SDK) Azure pour Python** sur [GitHub](#).

## Prérequis

Si vous débutez avec le service, consultez [Vue d'ensemble de Service Bus](#) avant de suivre ce démarrage rapide.

- Un abonnement Azure. Pour suivre ce tutoriel, vous avez besoin d'un compte Azure. Vous pouvez [activer les avantages de votre abonnement MSDN](#) ou vous inscrire pour un [compte gratuit](#).
- [Python 3.8](#) ou version ultérieure

Sans mot de passe (recommandé)

Pour utiliser ce guide de démarrage rapide avec votre propre compte Azure :

- Installez [Azure CLI](#), qui fournit l'authentification sans mot de passe à votre ordinateur de développement.

- Connectez-vous avec votre compte Azure au terminal ou à l'invite de commandes avec `az login`.
- Utilisez le même compte lorsque vous ajoutez le rôle de données approprié à votre ressource.
- Exécutez le code dans le même terminal ou la même invite de commande.
- Notez le nom de la **file d'attente** de votre espace de noms Service Bus. Vous en aurez besoin dans le code.

### ⓘ Notes

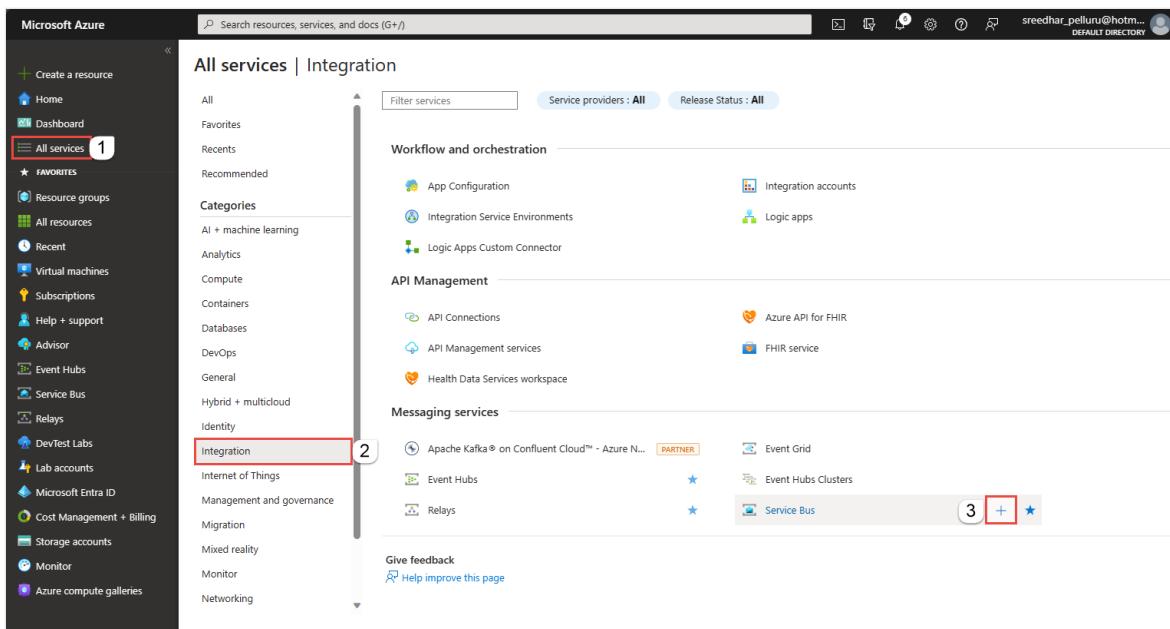
Ce tutoriel utilise des exemples que vous pouvez copier et exécuter avec Python. Pour obtenir des instructions sur la création d'une application Python, consultez [Créer et déployer une application Python sur un site web Azure](#). Pour plus d'informations sur l'installation des packages utilisés dans ce tutoriel, consultez le [Guide d'installation de Python](#).

## Créer un espace de noms dans le Portail Azure

Pour commencer à utiliser des entités de messagerie Service Bus dans Azure, vous devez d'abord créer un espace de noms avec un nom unique dans Azure. Un espace de noms fournit un conteneur d'étendue pour les ressources du Service Bus (files d'attente, thèmes, etc.) au sein de votre application.

Pour créer un espace de noms :

1. Connectez-vous au [portail Azure](#).
2. Dans le volet de navigation gauche du portail, sélectionnez **Tous les services**, puis **Intégration** dans la liste des catégories, pointez la souris sur **Service Bus**, puis sélectionnez le bouton + sur la vignette Service Bus.



3. Dans l'étiquette **De base** de la page **Créer un espace de noms**, suivez ces étapes :

- Pour l'option **Abonnement**, choisissez un abonnement Azure dans lequel créer l'espace de noms.
- Pour l'option **Groupe de ressources**, choisissez un groupe de ressources existant dans lequel l'espace de noms sera utilisé, ou créez-en un nouveau.
- Entrez un **nom pour l'espace de noms**. Le nom de l'espace de noms doit respecter les conventions de nommage suivantes :
  - Le nom doit être unique dans tout Azure. Le système vérifie immédiatement si le nom est disponible.
  - Le nom doit inclure entre 6 et 50 caractères.
  - Le nom ne peut contenir que des lettres, des chiffres et des traits d'union (« - »).
  - Le nom doit commencer par une lettre, et se terminer par une lettre ou un chiffre.
  - Le nom ne se termine ni par « -sb » ni par « -mgmt ».
- Pour l'option **Emplacement**, choisissez la région dans laquelle héberger votre espace de noms.
- Pour le **Niveau tarifaire**, sélectionnez le SKU (De base, Standard ou Premium) destiné à l'espace de noms. Pour ce guide de démarrage rapide, sélectionnez **Standard**.

### i Important

Si vous voulez utiliser des **rubriques et des abonnements**, choisissez Standard ou Premium. Les rubriques/abonnements ne sont pas pris en charge dans le niveau tarifaire De base.

Si vous avez sélectionné le SKU **Premium**, précisez le nombre d'**unité de messagerie**. Le niveau Premium isole les ressources au niveau du processeur et de la mémoire, ce qui permet d'exécuter chaque charge de travail de manière isolée. Ce conteneur de ressources est appelé unité de messagerie. Un espace de noms Premium a au moins une unité de messagerie. Vous pouvez sélectionner 1, 2, 4, 8 ou 16 unités de messagerie pour chaque espace de noms Service Bus Premium. Pour plus d'informations, consultez [Messagerie Service Bus Premium](#).

f. Au bas de la page, sélectionnez **Examiner et créer**.

The screenshot shows the 'Create namespace' wizard in the Azure portal. The title bar says 'Create namespace ...' and 'Service Bus'. The tabs at the top are 'Basics', 'Advanced', 'Networking', 'Tags', and 'Review + create'. The 'Basics' tab is selected. The 'Project Details' section has a note: 'Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.' It shows a 'Subscription' dropdown set to 'Visual Studio Enterprise Subscription' and a 'Resource group' dropdown showing '(New) spsbusrg' with a 'Create new' link. The 'Instance Details' section has fields for 'Namespace name' (contosoordersns), 'Location' (East US), and 'Pricing tier' (Standard). Below these are buttons for 'Review + create', '< Previous', and 'Next: Advanced >'.

g. Dans la page **Vérifier + créer**, passez en revue les paramètres, puis sélectionnez **Créer**.

4. Une fois le déploiement de la ressource réussi, sélectionnez **Accéder à la ressource** dans la page de déploiement.

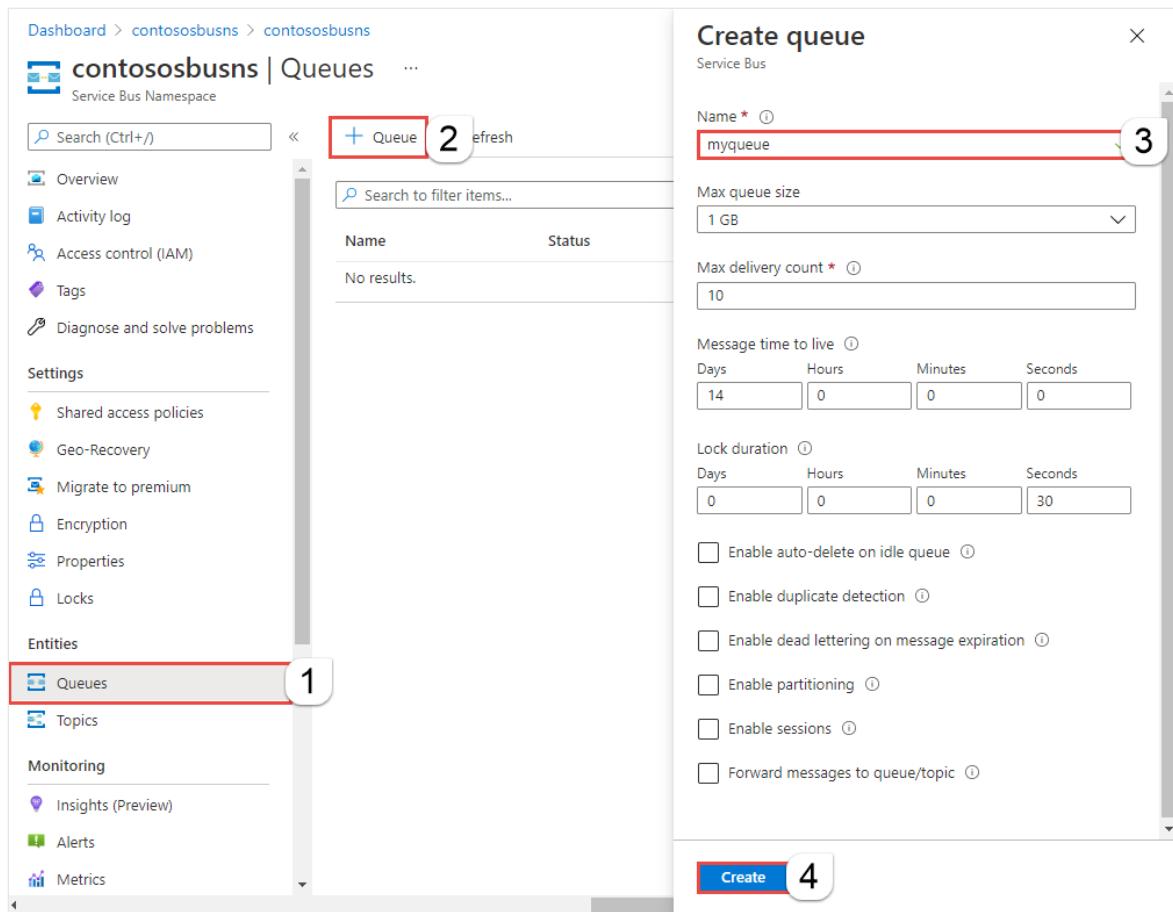
The screenshot shows the Azure Deployment Center interface. At the top, there's a search bar and several action buttons: Delete, Cancel, Redeploy, Download, and Refresh. Below the header, a sidebar on the left lists 'Overview', 'Inputs', 'Outputs', and 'Template'. The main content area displays a green checkmark icon and the message 'Your deployment is complete'. It provides deployment details: name (contosoordersns), subscription (Visual Studio Enterprise Subscription), and resource group (spsbusrg). It also shows the start time (10/20/2022, 4:45:03 PM) and correlation ID (a453ace1-bab9-4c4a-81ad-a1c5366460ea). Below this, there are sections for 'Deployment details' and 'Next steps', with a prominent blue 'Go to resource' button. At the bottom, there are links to 'Give feedback' and 'Tell us about your experience with deployment'.

5. Vous voyez la page d'accueil de votre espace de noms Service Bus.

The screenshot shows the Azure Service Bus Namespace overview page for 'spsbusns1128'. The left sidebar includes options like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Settings (Shared access policies, Geo-Recovery, Migrate to premium, Encryption, Configuration, Properties, Locks), Entities (Queues, Topics), Monitoring (Insights (Preview), Alerts), and Queues (0) / Topics (0). The main content area is titled 'Essentials' and displays various configuration details: Resource group (spsbusrg), Status (Active), Location (East US), Subscription (Visual Studio Enterprise Subscription), Subscription ID (0000000000-0000-0000-000000000000), Tags (Add tags), Created (Tuesday, November 28, 2023), Updated (Tuesday, November 28, 2023), Pricing tier (Standard), Host name (spsbusns1128.servicebus.windows.net), and Local Authentication (Enabled). Below the essentials, there are two line charts: 'Requests' and 'Messages', both showing zero activity over the last hour. A 'JSON View' link is located at the top right of the essentials section.

## Créer une file d'attente dans le portail Azure

1. Dans la page **Espace de noms Service Bus**, sélectionnez **Files d'attente** dans le menu de navigation de gauche.
2. Dans la page **Files d'attente**, sélectionnez **+ File d'attente** dans la barre d'outils.
3. Entrez un **nom** pour la file d'attente et laissez les valeurs par défaut des autres valeurs.
4. À présent, sélectionnez **Créer**.



## Authentifier l'application sur Azure

Ce guide de démarrage pratique vous montre deux façons de vous connecter à Azure Service Bus : **sans mot de passe** et avec une **chaîne de connexion**.

La première option vous explique comment utiliser votre principal de sécurité dans Microsoft Entra ID et le contrôle d'accès en fonction du rôle (RBAC) pour vous connecter à un espace de noms Service Bus. Vous n'avez pas à vous soucier d'avoir une chaîne de connexion codée en dur dans votre code, dans un fichier config ni dans un stockage sécurisé comme Azure Key Vault.

La deuxième option consiste à se servir d'une chaîne de connexion pour se connecter à un espace de noms Service Bus. Si vous débutez avec Azure, vous trouverez peut-être l'option chaîne de connexion plus facile à suivre. Nous vous recommandons d'utiliser l'option sans mot de passe dans les applications réelles et les environnements de production. Pour plus d'informations, consultez [Authentification et autorisation](#). Pour en savoir plus sur l'authentification sans mot de passe, reportez-vous à la [page de présentation](#).

Sans mot de passe (recommandé)

# Attribuer des rôles à votre utilisateur Microsoft Entra

Lors du développement localement, assurez-vous que le compte d'utilisateur qui se connecte à Azure Service Bus dispose des autorisations appropriées. Vous aurez besoin du rôle [Propriétaire de données Azure Service Bus](#) pour envoyer et recevoir des messages. Pour vous attribuer ce rôle, vous aurez besoin du rôle Administrateur de l'accès utilisateur ou d'un autre rôle qui inclut l'action

`Microsoft.Authorization/roleAssignments/write`. Vous pouvez attribuer des rôles RBAC Azure à un utilisateur à l'aide du Portail Azure, Azure CLI ou Azure PowerShell. Découvrez les étendues disponibles pour les attributions de rôles dans la page [vue d'ensemble de l'étendue](#).

L'exemple suivant attribue le rôle `Azure Service Bus Data Owner` à votre compte d'utilisateur, qui fournit un accès complet aux ressources Azure Service Bus. Dans un scénario réel, suivez le [principe des privilèges minimum](#) pour accorder aux utilisateurs uniquement les autorisations minimales nécessaires à un environnement de production plus sécurisé.

## Rôles Azure intégrés pour Azure Service Bus

Pour Azure Service Bus, la gestion des espaces de noms et de toutes les ressources associées via le Portail Azure et l'API de gestion des ressources Azure est déjà protégée à l'aide du modèle Azure RBAC. Azure fournit les rôles Azure intégrés ci-dessous pour autoriser l'accès à un espace de noms Service Bus :

- [Propriétaire de données Azure Service Bus](#) : ce rôle permet l'accès aux données de l'espace de noms Service Bus et de ses entités (files d'attente, rubriques, abonnements et filtres). Un membre de ce rôle peut envoyer et recevoir des messages à partir de files d'attente ou de rubriques et d'abonnements.
- [Expéditeur de données Azure Service Bus](#) : utilisez ce rôle pour autoriser l'accès en envoi à l'espace de noms Service Bus et à ses entités.
- [Récepteur de données Azure Service Bus](#) : utilisez ce rôle pour autoriser l'accès en réception à l'espace de noms Service Bus et à ses entités.

Si vous souhaitez créer un rôle personnalisé, consultez [Droits requis pour les opérations Service Bus](#).

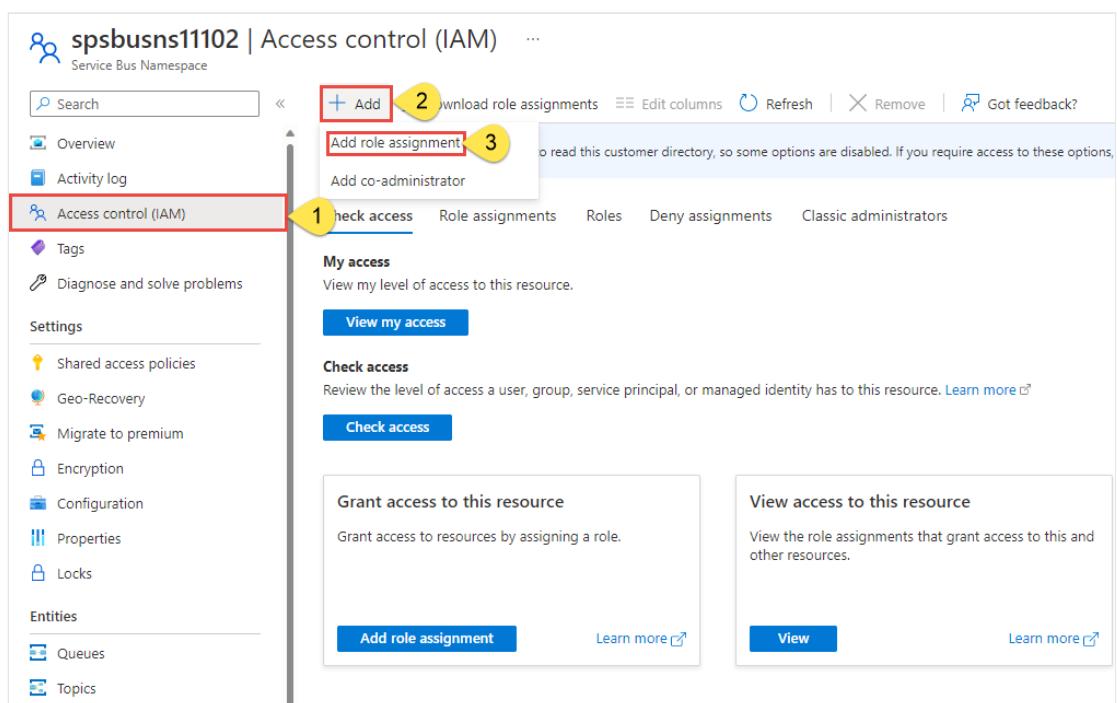
## Ajouter un utilisateur Microsoft Entra au rôle Propriétaire Azure Service Bus

Ajoutez votre nom d'utilisateur Microsoft Entra au rôle **Propriétaire de données** Azure Service Bus au niveau de l'espace de noms Service Bus. Il permet à une application exécutée dans le contexte de votre compte d'utilisateur d'envoyer des messages à une file d'attente ou à une rubrique et d'en recevoir auprès d'une file d'attente ou de l'abonnement d'une rubrique.

### ⓘ Important

Dans la plupart des cas, la propagation de l'attribution de rôle dans Azure peut prendre une ou deux minutes. Dans de rares cas, cela peut prendre jusqu'à **huit minutes**. Si vous recevez des erreurs d'authentification lorsque vous exécutez votre code pour la première fois, patientez quelques instants et réessayez.

1. Si la page Espace de noms Service Bus n'est pas ouverte sur le Portail Azure, recherchez votre espace de noms Service Bus à l'aide de la barre de recherche principale ou du volet de navigation de gauche.
2. Dans la page vue d'ensemble, sélectionnez **Contrôle d'accès (IAM)** dans le menu de gauche.
3. Sur la page **Contrôle d'accès (IAM)**, sélectionnez l'onglet **Attributions de rôles**.
4. Sélectionnez **+ Ajouter** dans le menu supérieur, puis **Ajouter une attribution de rôle** dans le menu déroulant résultant.



5. Utilisez la zone de recherche pour filtrer les résultats sur le rôle souhaité. Pour cet exemple, recherchez `Azure Service Bus Data Owner` et sélectionnez le résultat correspondant. Ensuite, choisissez **Suivant**.
6. Sous **Attribuer l'accès à**, sélectionnez **Utilisateur, groupe ou principal de service**, puis sélectionnez + **Sélectionner des membres**.
7. Dans la boîte de dialogue, recherchez votre nom d'utilisateur Microsoft Entra (généralement votre adresse e-mail `utilisateur@domaine`), puis choisissez **Sélectionner** en bas de la boîte de dialogue.
8. Sélectionnez **Vérifier + affecter** pour accéder à la page finale, puis **Vérifier + attribuer** à nouveau pour terminer le processus.

## Utiliser pip pour installer les packages

Sans mot de passe (recommandé)

1. Pour installer les packages Python requis pour ce tutoriel Service Bus, ouvrez une invite de commandes dont le chemin contient Python et remplacez le répertoire par le dossier où vous souhaitez stocker vos exemples.
2. Installez les packages suivants :

shell

```
pip install azure-servicebus
pip install azure-identity
pip install aiohttp
```

## Envoi de messages à une file d'attente

L'exemple de code suivant illustre comment envoyer un message à une file d'attente. Ouvrez votre éditeur favori, tel que [Visual Studio Code](#), créez un fichier `send.py` et ajoutez-y le code suivant.

Sans mot de passe (recommandé)

## 1. Ajoutez des instructions import.

```
Python

import asyncio
from azure.servicebus.aio import ServiceBusClient
from azure.servicebus import ServiceBusMessage
from azure.identity.aio import DefaultAzureCredential
```

## 2. Ajoutez des constantes et définissez des informations d'identification.

```
Python

FULLY_QUALIFIED_NAMESPACE = "FULLY_QUALIFIED_NAMESPACE"
QUEUE_NAME = "QUEUE_NAME"

credential = DefaultAzureCredential()
```

### ⓘ Important

- Remplacez `FULLY_QUALIFIED_NAMESPACE` par l'espace de noms complet de votre espace de noms Service Bus.
- et remplacez `QUEUE_NAME` par le nom de la file d'attente.

## 3. Ajoutez une méthode pour envoyer un message unique.

```
Python

async def send_single_message(sender):
 # Create a Service Bus message and send it to the queue
 message = ServiceBusMessage("Single Message")
 await sender.send_messages(message)
 print("Sent a single message")
```

L'expéditeur est un objet qui agit en tant que client pour la file d'attente que vous avez créée. Vous le créerez ultérieurement et l'enverrez en tant qu'argument à cette fonction.

## 4. Ajoutez une méthode pour envoyer une liste de messages.

```
Python
```

```

async def send_a_list_of_messages(sender):
 # Create a list of messages and send it to the queue
 messages = [ServiceBusMessage("Message in list") for _ in
range(5)]
 await sender.send_messages(messages)
 print("Sent a list of 5 messages")

```

5. Ajoutez une méthode pour envoyer un lot de messages.

Python

```

async def send_batch_message(sender):
 # Create a batch of messages
 async with sender:
 batch_message = await sender.create_message_batch()
 for _ in range(10):
 try:
 # Add a message to the batch

 batch_message.add_message(ServiceBusMessage("Message inside a
ServiceBusMessageBatch"))
 except ValueError:
 # ServiceBusMessageBatch object reaches max_size.
 # New ServiceBusMessageBatch object can be created
 here to send more data.
 break
 # Send the batch of messages to the queue
 await sender.send_messages(batch_message)
 print("Sent a batch of 10 messages")

```

6. Créez un client Service Bus, puis un objet expéditeur de file d'attente pour envoyer des messages.

Python

```

async def run():
 # create a Service Bus client using the credential
 async with ServiceBusClient(
 fully_qualified_namespace=FULLY_QUALIFIED_NAMESPACE,
 credential=credential,
 logging_enable=True) as servicebus_client:
 # get a Queue Sender object to send messages to the queue
 sender =
servicebus_client.get_queue_sender(queue_name=QUEUE_NAME)
 async with sender:
 # send one message
 await send_single_message(sender)
 # send a list of messages
 await send_a_list_of_messages(sender)
 # send a batch of messages

```

```
 await send_batch_message(sender)

 # Close credential when no longer needed.
 await credential.close()
```

7. Appelez la méthode `run` et affichez un message.

Python

```
asyncio.run(run())
print("Done sending messages")
print("-----")
```

## Réception des messages d'une file d'attente

L'exemple de code suivant illustre comment recevoir des messages d'une file d'attente. Le code affiché reçoit de nouveaux messages jusqu'à ce qu'il n'en reçoive plus pendant 5 (`max_wait_time`) secondes.

Ouvrez votre éditeur favori, tel que [Visual Studio Code](#), créez un fichier `recv.py` et ajoutez-y le code suivant.

Sans mot de passe (recommandé)

1. Comme dans l'exemple d'envoi, ajoutez des instructions `import`, définissez des constantes que vous devez remplacer par vos propres valeurs et définissez des informations d'identification.

Python

```
import asyncio

from azure.servicebus.aio import ServiceBusClient
from azure.identity.aio import DefaultAzureCredential

FULLY_QUALIFIED_NAMESPACE = "FULLY_QUALIFIED_NAMESPACE"
QUEUE_NAME = "QUEUE_NAME"

credential = DefaultAzureCredential()
```

2. Créez un client Service Bus, puis un objet récepteur de file d'attente pour recevoir des messages.

Python

```
async def run():
 # create a Service Bus client using the connection string
 async with ServiceBusClient(
 fully_qualified_namespace=FULLY_QUALIFIED_NAMESPACE,
 credential=credential,
 logging_enable=True) as servicebus_client:

 async with servicebus_client:
 # get the Queue Receiver object for the queue
 receiver =
 servicebus_client.get_queue_receiver(queue_name=QUEUE_NAME)
 async with receiver:
 received_msgs = await
 receiver.receive_messages(max_wait_time=5, max_message_count=20)
 for msg in received_msgs:
 print("Received: " + str(msg))
 # complete the message so that the message is
 removed from the queue
 await receiver.complete_message(msg)

 # Close credential when no longer needed.
 await credential.close()
```

3. Appelez la méthode `run`.

Python

```
asyncio.run(run())
```

## Exécuter l'application

Ouvrez une invite de commandes avec Python dans son chemin, puis exécutez le code pour envoyer et recevoir des messages de la file d'attente.

shell

```
python send.py; python recv.py
```

Vous devez normalement voir la sortie suivante :

Console

```
Sent a single message
Sent a list of 5 messages
```

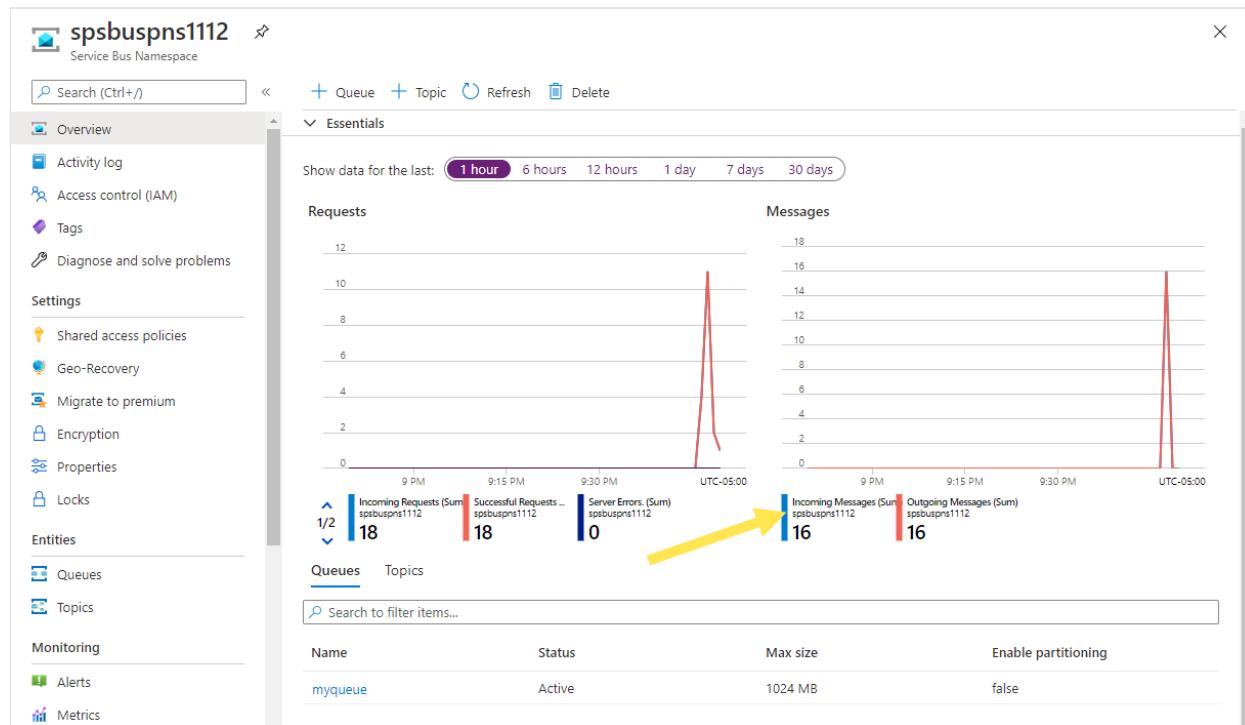
```

Sent a batch of 10 messages
Done sending messages

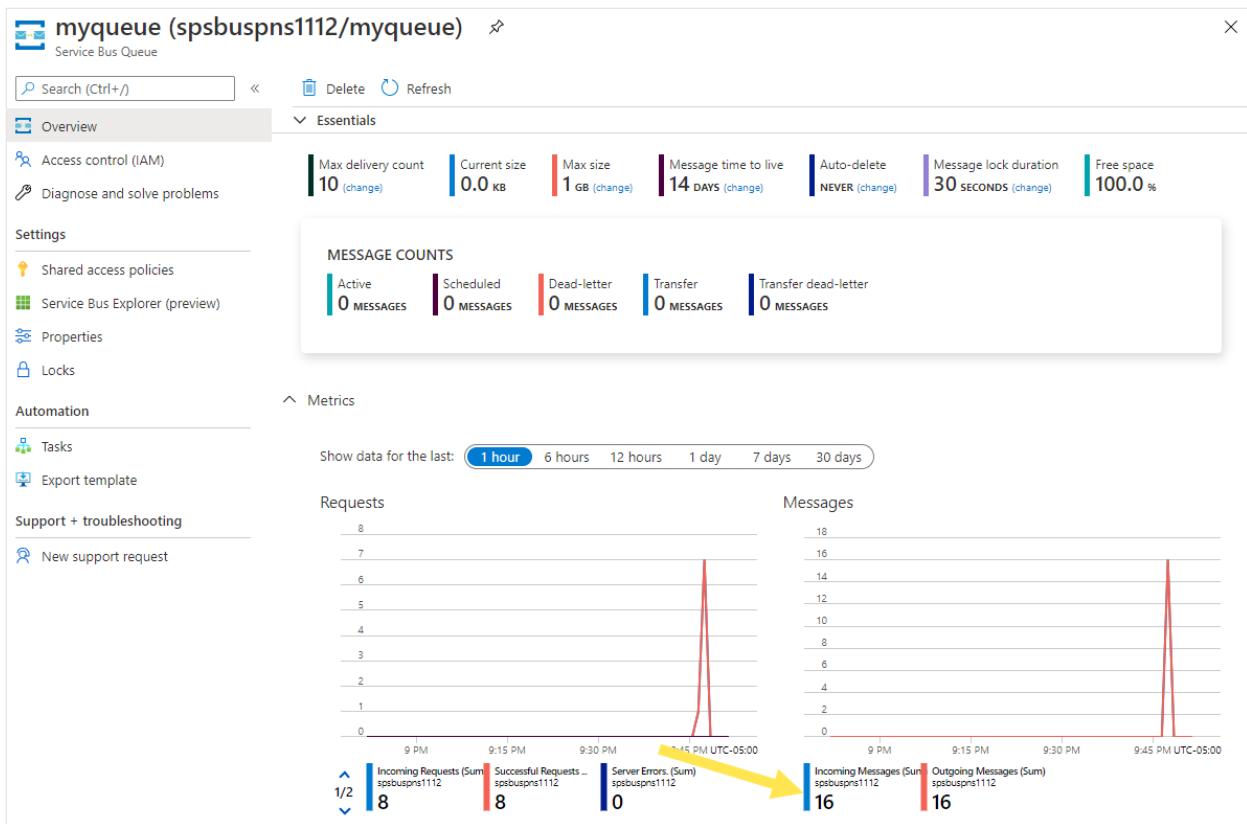
Received: Single Message
Received: Message in list
Received: Message inside a ServiceBusMessageBatch

```

Dans le portail Azure, accédez à votre espace de noms Service Bus. Dans la page **Vue d'ensemble**, vérifiez que le nombre de messages **entrants** et **sortants** est égal à 16. Si ces chiffres ne s'affichent pas, actualisez la page après quelques minutes.



Selectionnez la file d'attente dans cette page **Vue d'ensemble** pour accéder à la page **File d'attente Service Bus**. Vous pouvez également voir le nombre de messages **entrants** et **sortants** dans cette page, ainsi que d'autres informations telles que la **taille actuelle** de la file d'attente et le **nombre de messages actifs**.



## Étapes suivantes

Voir la documentation et les exemples suivants :

- [Bibliothèque de client Azure Service Bus pour Python](#)
- [Exemples](#).
  - Le dossier **sync\_samples** contient des exemples qui montrent comment interagir avec Service Bus de manière synchrone.
  - Le dossier **async\_samples** contient des exemples qui montrent comment interagir avec Service Bus de manière asynchrone. Dans ce guide de démarrage rapide, vous avez utilisé cette méthode.
- [Documentation de référence sur azure-servicebus](#)

# Envoyer des messages à une rubrique Azure Service Bus et recevoir des messages à partir d'abonnements à la rubrique (Python)

Article • 06/02/2024

Dans ce tutoriel, vous allez effectuer les étapes suivantes :

1. Créer un espace de noms Service Bus à l'aide du Portail Azure.
2. Créer une rubrique Service Bus à l'aide du Portail Azure.
3. Créer un abonnement Service Bus vers cette rubrique à l'aide du Portail Azure.
4. Écrivez une application Python pour utiliser le package [azure-servicebus](#) et effectuer les tâches suivantes :
  - Envoyer un ensemble de messages à la rubrique.
  - Recevoir les messages de l'abonnement.

## ⓘ Notes

Ce guide de démarrage rapide fournit des instructions pas à pas pour un scénario simple qui consiste à envoyer un lot de messages à une rubrique Service Bus et à recevoir ces messages à partir d'un abonnement de la rubrique. Vous trouverez des exemples Python prédéfinis pour Azure Service Bus dans le [dépôt du kit SDK Azure pour Python sur GitHub](#).

## Prérequis

- Un [abonnement Azure](#).
- [Python 3.8](#) ou version ultérieure

## ⓘ Notes

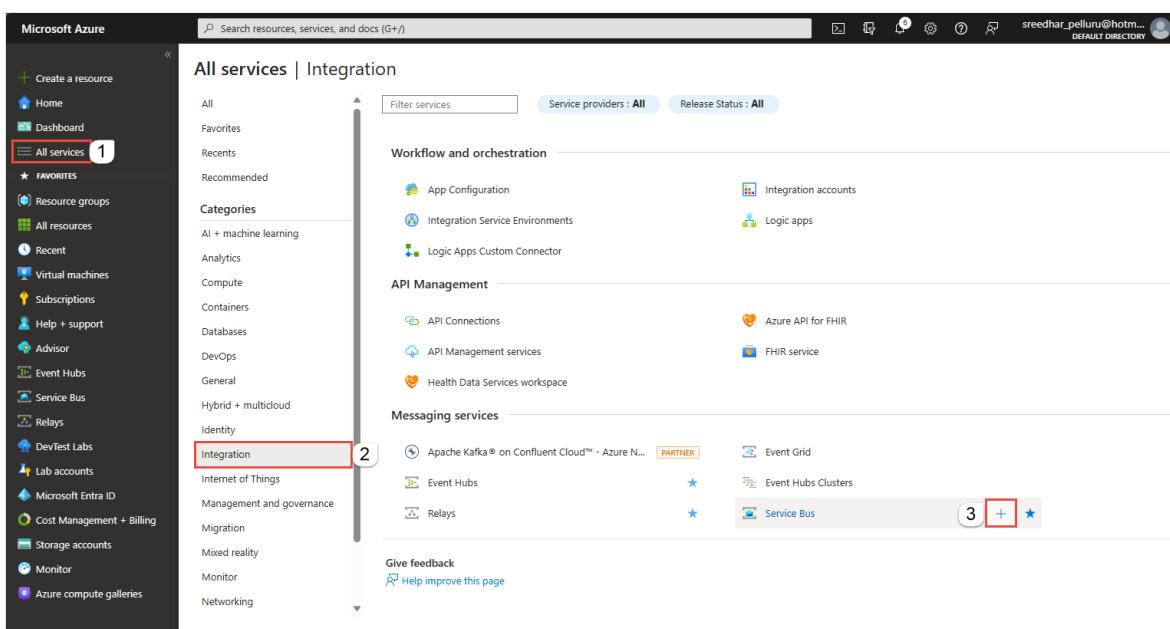
Ce tutoriel utilise des exemples que vous pouvez copier et exécuter avec Python. Pour obtenir des instructions sur la création d'une application Python, consultez [Créer et déployer une application Python sur un site web Azure](#). Pour plus d'informations sur l'installation des packages utilisés dans ce tutoriel, consultez le [Guide d'installation de Python](#).

# Créer un espace de noms dans le Portail Azure

Pour commencer à utiliser des entités de messagerie Service Bus dans Azure, vous devez d'abord créer un espace de noms avec un nom unique dans Azure. Un espace de noms fournit un conteneur d'étendue pour les ressources du Service Bus (files d'attente, thèmes, etc.) au sein de votre application.

Pour créer un espace de noms :

1. Connectez-vous au [portail Azure](#).
2. Dans le volet de navigation gauche du portail, sélectionnez **Tous les services**, puis **Intégration** dans la liste des catégories, pointez la souris sur **Service Bus**, puis sélectionnez le bouton **+** sur la vignette Service Bus.



3. Dans l'étiquette De base de la page **Créer un espace de noms**, suivez ces étapes :
  - a. Pour l'option **Abonnement**, choisissez un abonnement Azure dans lequel créer l'espace de noms.
  - b. Pour l'option **Groupe de ressources**, choisissez un groupe de ressources existant dans lequel l'espace de noms sera utilisé, ou créez-en un nouveau.
  - c. Entrez un **nom pour l'espace de noms**. Le nom de l'espace de noms doit respecter les conventions de nommage suivantes :
    - Le nom doit être unique dans tout Azure. Le système vérifie immédiatement si le nom est disponible.
    - Le nom doit inclure entre 6 et 50 caractères.

- Le nom ne peut contenir que des lettres, des chiffres et des traits d'union (« - »).
- Le nom doit commencer par une lettre, et se terminer par une lettre ou un chiffre.
- Le nom ne se termine ni par « -sb » ni par « -mgmt ».

d. Pour l'option **Emplacement**, choisissez la région dans laquelle héberger votre espace de noms.

e. Pour le **Niveau tarifaire**, sélectionnez le SKU (De base, Standard ou Premium) destiné à l'espace de noms. Pour ce guide de démarrage rapide, sélectionnez **Standard**.

 **Important**

Si vous voulez utiliser des **rubriques et des abonnements**, choisissez Standard ou Premium. Les rubriques/abonnements ne sont pas pris en charge dans le niveau tarifaire De base.

Si vous avez sélectionné le SKU **Premium**, précisez le nombre d'**unité de messagerie**. Le niveau Premium isole les ressources au niveau du processeur et de la mémoire, ce qui permet d'exécuter chaque charge de travail de manière isolée. Ce conteneur de ressources est appelé unité de messagerie. Un espace de noms Premium a au moins une unité de messagerie. Vous pouvez sélectionner 1, 2, 4, 8 ou 16 unités de messagerie pour chaque espace de noms Service Bus Premium. Pour plus d'informations, consultez [Messagerie Service Bus Premium](#).

f. Au bas de la page, sélectionnez **Examiner et créer**.

**Create namespace** ...

Service Bus

**Basics**   Advanced   Networking   Tags   Review + create

**Project Details**

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription \* Visual Studio Enterprise Subscription

Resource group \* (New) spsbusrg [Create new](#)

**Instance Details**

Enter required settings for this namespace.

Namespace name \* contosoordersns [.servicebus.windows.net](#)

Location \* East US

Pricing tier \* Standard [Browse the available plans and their features](#)

**Review + create**   < Previous   Next: Advanced >

g. Dans la page **Vérifier + créer**, passez en revue les paramètres, puis sélectionnez **Créer**.

4. Une fois le déploiement de la ressource réussi, sélectionnez **Accéder à la ressource** dans la page de déploiement.

contosoordersns | Overview

Deployment

Search

Delete Cancel Redeploy Download Refresh

Overview

Your deployment is complete

Deployment name: contosoordersns  
Subscription: Visual Studio Enterprise Subscription  
Resource group: spsbusrg

Start time: 10/20/2022, 4:45:03 PM  
Correlation ID: a453ace1-bab9-4c4a-81ad-a1c5366460ea

Deployment details

Next steps

Go to resource

Give feedback

Tell us about your experience with deployment

5. Vous voyez la page d'accueil de votre espace de noms Service Bus.

The screenshot shows the Azure Service Bus Namespace overview for the namespace spsbusns1128. The left sidebar contains navigation links for Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Settings, Entities (Queues, Topics), Monitoring (Insights (Preview), Alerts), Queues (0), and Topics (0). The main area displays the 'Essentials' section with details such as Resource group (spsbusrg), Status (Active), Location (East US), Subscription (Visual Studio Enterprise Subscription), Subscription ID (0000000000-0000-0000-0000-000000000000), and Tags (Add tags). Below this is a chart showing Requests and Messages over the last hour, with specific data points for Incoming Requests, Successful Requests, Server Errors, User Errors, Incoming Messages, and Outgoing Messages.

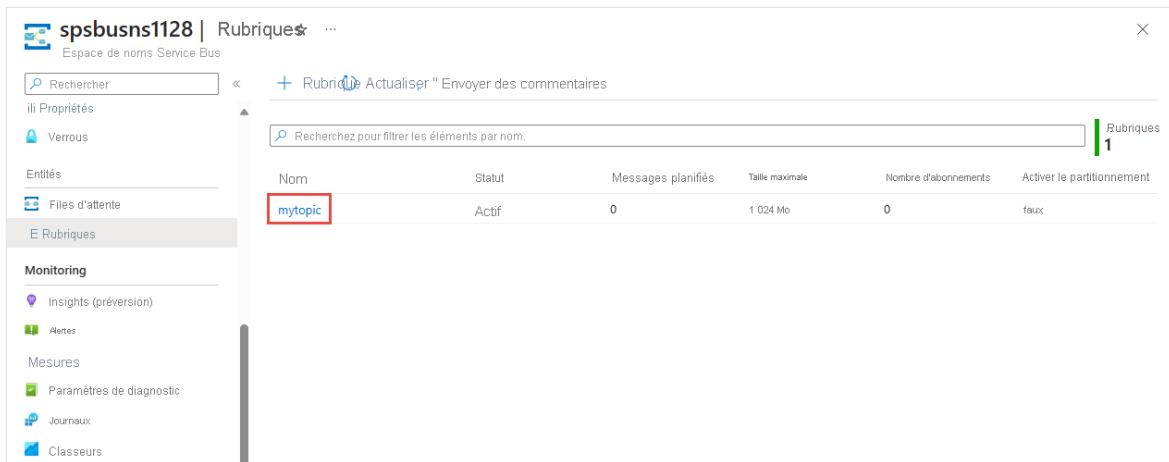
## Créer une rubrique à l'aide du Portail Azure

1. Dans la page Espace de noms Service Bus, sélectionnez Rubriques dans le menu de gauche.
2. Sélectionnez + Rubrique dans la barre d'outils.
3. Entrez un nom pour la rubrique. Conservez les valeurs par défaut des autres options.
4. Cliquez sur Créer.

The screenshot shows the Azure Service Bus Namespace Topics page for the namespace spsbusns1128. The left sidebar contains navigation links for Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Settings, Entities (Queues, Topics), Monitoring (Insights (Preview), Alerts), Queues (0), and Topics (1). The main area displays a 'Create topic' dialog. In the dialog, the 'Name' field is set to 'mytopic'. Other settings include Max topic size (1 GB), Message time to live (14 Days, 0 Hours, 0 Minutes, 0 Seconds), and several optional checkboxes: Enable auto-delete on idle topic, Enable duplicate detection, Enable partitioning, and Support ordering. At the bottom right of the dialog is a 'Create' button.

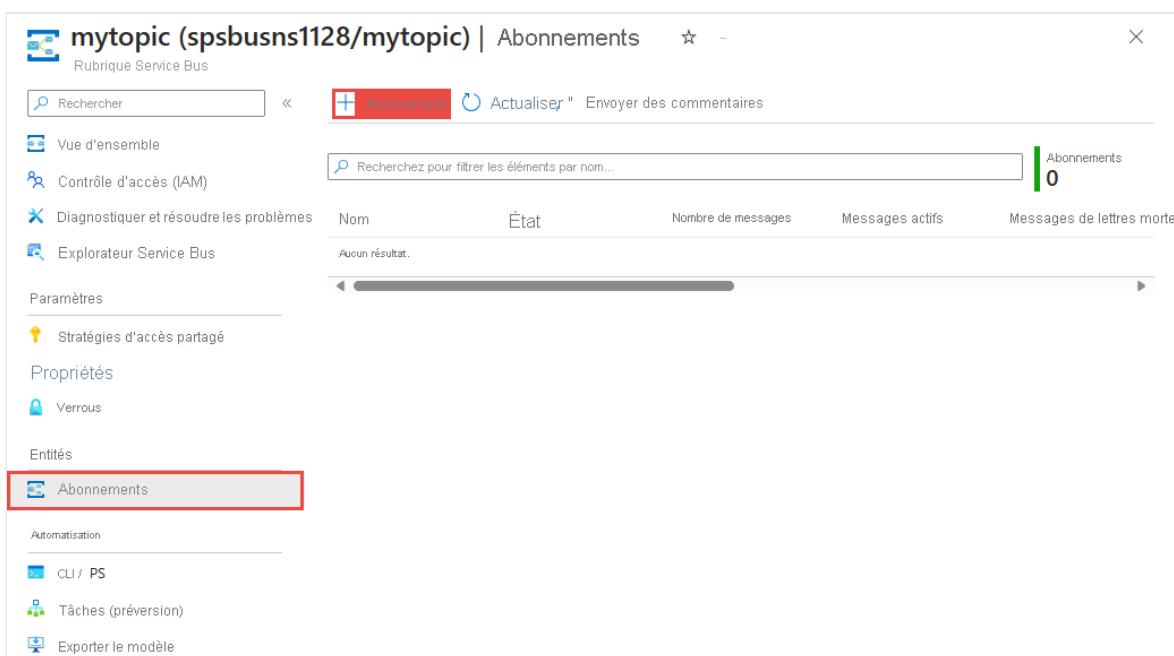
# Créer un abonnement à la rubrique

1. Sélectionnez la **rubrique** que vous avez créée dans la section précédente.



The screenshot shows the 'Rubriques' (Topics) blade in the Azure Service Bus Espace de noms (Namespace). A single topic named 'mytopic' is listed. The table columns are: Nom (Name), Statut (Status), Messages planifiés (Scheduled messages), Taille maximale (Max size), Nombre d'abonnements (Number of subscriptions), and Activer le partitionnement (Enable partitioning). The 'Nom' column for 'mytopic' contains the value 'mytopic'. The 'Statut' column shows 'Actif' (Active). The 'Messages planifiés' column shows '0'. The 'Taille maximale' column shows '1 024 Mo'. The 'Nombre d'abonnements' column shows '0'. The 'Activer le partitionnement' column shows 'faux' (false). A red box highlights the 'mytopic' entry in the list.

2. Dans la page **Rubrique Service Bus**, dans la barre d'outils, sélectionnez **+ Abonnement**.



The screenshot shows the 'Abonnements' (Subscriptions) blade for the 'mytopic' topic. The blade header is 'mytopic (spsbusns1128/mytopic) | Abonnements'. The table has columns: Nom (Name), État (Status), Nombre de messages (Number of messages), Messages actifs (Active messages), and Messages de lettres morte (Dead-letter messages). A message at the bottom states 'Aucun résultat.' (No results). A red box highlights the 'Abonnements' button in the left sidebar.

3. Dans la page **Créer un abonnement**, procédez comme suit :

- Entrez **S1** pour le **nom** de l'abonnement.
- Entrez **3** pour **Nombre maximal de remises**.
- Ensuite, sélectionnez **Créer** pour créer l'abonnement.

## Create subscription

Service Bus

Name \* ⓘ

S1



Max delivery count \* ⓘ

10

Auto-delete after idle for ⓘ

Days

Hours

Minutes

Seconds

14

0

0

0

Never auto-delete

Forward messages to queue/topic ⓘ

### MESSAGE SESSIONS

Service bus sessions allow ordered handling of unbounded sequences of related messages. With sessions enabled a subscription can guarantee first-in-first-out delivery of messages. [Learn more.](#)

Enable sessions

### MESSAGE TIME TO LIVE AND DEAD-LETTERING

Message time to live (default) ⓘ

Days

Hours

Minutes

Seconds

14

0

0

0

Enable dead lettering on message expiration

Move messages that cause filter evaluation exceptions to the dead-letter subqueue

### MESSAGE LOCK DURATION

Lock duration ⓘ

Days

Hours

Minutes

Seconds

0

0

1

0

**Create**

## Authentifier l'application sur Azure

Ce guide de démarrage pratique vous montre deux façons de vous connecter à Azure Service Bus : **sans mot de passe** et avec une **chaîne de connexion**.

La première option vous explique comment utiliser votre principal de sécurité dans Microsoft Entra ID et le contrôle d'accès en fonction du rôle (RBAC) pour vous connecter à un espace de noms Service Bus. Vous n'avez pas à vous soucier d'avoir une chaîne de

connexion codée en dur dans votre code, dans un fichier config ni dans un stockage sécurisé comme Azure Key Vault.

La deuxième option consiste à se servir d'une chaîne de connexion pour se connecter à un espace de noms Service Bus. Si vous débutez avec Azure, vous trouverez peut-être l'option chaîne de connexion plus facile à suivre. Nous vous recommandons d'utiliser l'option sans mot de passe dans les applications réelles et les environnements de production. Pour plus d'informations, consultez [Authentification et autorisation](#). Pour en savoir plus sur l'authentification sans mot de passe, reportez-vous à la [page de présentation](#).

Sans mot de passe (recommandé)

## Attribuer des rôles à votre utilisateur Microsoft Entra

Lors du développement localement, assurez-vous que le compte d'utilisateur qui se connecte à Azure Service Bus dispose des autorisations appropriées. Vous aurez besoin du rôle [Propriétaire de données Azure Service Bus](#) pour envoyer et recevoir des messages. Pour vous attribuer ce rôle, vous aurez besoin du rôle Administrateur de l'accès utilisateur ou d'un autre rôle qui inclut l'action

`Microsoft.Authorization/roleAssignments/write`. Vous pouvez attribuer des rôles RBAC Azure à un utilisateur à l'aide du Portail Azure, Azure CLI ou Azure PowerShell. Découvrez les étendues disponibles pour les attributions de rôles dans la page [vue d'ensemble de l'étendue](#).

L'exemple suivant attribue le rôle `Azure Service Bus Data Owner` à votre compte d'utilisateur, qui fournit un accès complet aux ressources Azure Service Bus. Dans un scénario réel, suivez le [principe des privilèges](#) minimum pour accorder aux utilisateurs uniquement les autorisations minimales nécessaires à un environnement de production plus sécurisé.

## Rôles Azure intégrés pour Azure Service Bus

Pour Azure Service Bus, la gestion des espaces de noms et de toutes les ressources associées via le Portail Azure et l'API de gestion des ressources Azure est déjà protégée à l'aide du modèle Azure RBAC. Azure fournit les rôles Azure intégrés ci-dessous pour autoriser l'accès à un espace de noms Service Bus :

- [Propriétaire de données Azure Service Bus](#) : ce rôle permet l'accès aux données de l'espace de noms Service Bus et de ses entités (files d'attente,

rubriques, abonnements et filtres). Un membre de ce rôle peut envoyer et recevoir des messages à partir de files d'attente ou de rubriques et d'abonnements.

- [Expéditeur de données Azure Service Bus](#) : utilisez ce rôle pour autoriser l'accès en envoi à l'espace de noms Service Bus et à ses entités.
- [Récepteur de données Azure Service Bus](#) : utilisez ce rôle pour autoriser l'accès en réception à l'espace de noms Service Bus et à ses entités.

Si vous souhaitez créer un rôle personnalisé, consultez [Droits requis pour les opérations Service Bus](#).

## Ajouter un utilisateur Microsoft Entra au rôle Propriétaire Azure Service Bus

Ajoutez votre nom d'utilisateur Microsoft Entra au rôle **Propriétaire de données Azure Service Bus** au niveau de l'espace de noms Service Bus. Il permet à une application exécutée dans le contexte de votre compte d'utilisateur d'envoyer des messages à une file d'attente ou à une rubrique et d'en recevoir auprès d'une file d'attente ou de l'abonnement d'une rubrique.

### Important

Dans la plupart des cas, la propagation de l'attribution de rôle dans Azure peut prendre une ou deux minutes. Dans de rares cas, cela peut prendre jusqu'à **huit minutes**. Si vous recevez des erreurs d'authentification lorsque vous exécutez votre code pour la première fois, patientez quelques instants et réessayez.

1. Si la page Espace de noms Service Bus n'est pas ouverte sur le Portail Azure, recherchez votre espace de noms Service Bus à l'aide de la barre de recherche principale ou du volet de navigation de gauche.
2. Dans la page vue d'ensemble, sélectionnez **Contrôle d'accès (IAM)** dans le menu de gauche.
3. Sur la page **Contrôle d'accès (IAM)**, sélectionnez l'onglet **Attributions de rôles**.
4. Sélectionnez **+ Ajouter** dans le menu supérieur, puis **Ajouter une attribution de rôle** dans le menu déroulant résultant.

The screenshot shows the 'Access control (IAM)' section of the Azure Service Bus Namespace. The left sidebar has a red box around 'Access control (IAM)'. Above the main content area, there's a toolbar with 'Add' (highlighted with a red box and yellow arrow '2'), 'Download role assignments', 'Edit columns', 'Refresh', 'Remove', and 'Got feedback?'. A tooltip '3' points to the 'Add role assignment' button. The main content includes sections for 'My access', 'Check access', 'Grant access to this resource' (with 'Add role assignment' and 'Learn more' buttons), and 'View access to this resource' (with 'View' and 'Learn more' buttons).

5. Utilisez la zone de recherche pour filtrer les résultats sur le rôle souhaité. Pour cet exemple, recherchez **Azure Service Bus Data Owner** et sélectionnez le résultat correspondant. Ensuite, choisissez **Suivant**.
6. Sous **Attribuer l'accès à**, sélectionnez **Utilisateur, groupe ou principal de service**, puis sélectionnez + **Sélectionner des membres**.
7. Dans la boîte de dialogue, recherchez votre nom d'utilisateur Microsoft Entra (généralement votre adresse e-mail *utilisateur@domaine*), puis choisissez **Sélectionner** en bas de la boîte de dialogue.
8. Sélectionnez **Vérifier + affecter** pour accéder à la page finale, puis **Vérifier + attribuer** à nouveau pour terminer le processus.

## Configuration du code

Sans mot de passe (recommandé)

Pour suivre ce guide de démarrage rapide à l'aide de l'authentification sans mot de passe et de votre propre compte Azure :

- Installez [Azure CLI](#).
- Connectez-vous avec votre compte Azure au terminal ou à l'invite de commandes avec `az login`.

- Utilisez le même compte quand vous ajouterez le rôle approprié à votre ressource plus tard dans le tutoriel.
- Exécutez le code du tutoriel dans le même terminal ou la même invite de commandes.

### ⓘ Important

Veillez à vous connecter avec `az login`. La classe `DefaultAzureCredential` du code sans mot de passe utilise les informations d'identification Azure CLI pour s'authentifier avec Microsoft Entra ID.

Pour utiliser le code sans mot de passe, vous devez spécifier un :

- espace de noms Service Bus complet, par exemple : `<spacedenoms-servicebus>.servicebus.windows.net`
- nom de rubrique
- subscription name

## Utiliser pip pour installer les packages

Sans mot de passe (recommandé)

1. Pour installer les packages Python nécessaires à ce tutoriel Service Bus, ouvrez une invite de commandes comprenant Python dans son chemin. Remplacez le répertoire par le dossier où vous souhaitez placer vos exemples.
2. Installez les packages :

shell

```
pip install azure-servicebus
pip install azure-identity
pip install aiohttp
```

## Envoi de messages à une rubrique

L'exemple de code suivant montre comment envoyer un lot de messages à une rubrique Service Bus. Pour plus d'informations, consultez les commentaires du code.

Ouvrez votre éditeur favori, tel que [Visual Studio Code](#), créez un fichier `send.py` et ajoutez-y le code suivant.

Sans mot de passe (recommandé)

1. Ajoutez les instructions `import` suivantes.

Python

```
import asyncio
from azure.servicebus.aio import ServiceBusClient
from azure.servicebus import ServiceBusMessage
from azure.identity.aio import DefaultAzureCredential
```

2. Ajoutez les constantes et définissez des informations d'identification.

Python

```
FULLY_QUALIFIED_NAMESPACE = "FULLY_QUALIFIED_NAMESPACE"
TOPIC_NAME = "TOPIC_NAME"

credential = DefaultAzureCredential()
```

### ⓘ Important

- Remplacez `FULLY_QUALIFIED_NAMESPACE` par l'espace de noms complet de votre espace de noms Service Bus.
- et remplacez `TOPIC_NAME` par le nom de la rubrique.

Dans le code précédent, vous avez utilisé la classe `DefaultAzureCredential` de la bibliothèque de client Azure Identity. Quand l'application s'exécute localement durant le développement, `DefaultAzureCredential` découvre et s'authentifie automatiquement auprès d'Azure à l'aide du compte avec lequel vous vous êtes connecté à Azure CLI. Quand l'application est déployée sur Azure, `DefaultAzureCredential` peut authentifier votre application auprès de Microsoft Entra ID via une identité managée sans aucun changement du code.

3. Ajoutez une méthode pour envoyer un message unique.

Python

```

async def send_single_message(sender):
 # Create a Service Bus message
 message = ServiceBusMessage("Single Message")
 # send the message to the topic
 await sender.send_messages(message)
 print("Sent a single message")

```

L'expéditeur est un objet qui agit en tant que client pour la rubrique que vous avez créée. Vous le créeerez ultérieurement et l'enverrez en tant qu'argument à cette fonction.

#### 4. Ajoutez une méthode pour envoyer une liste de messages.

Python

```

async def send_a_list_of_messages(sender):
 # Create a list of messages
 messages = [ServiceBusMessage("Message in list") for _ in
range(5)]
 # send the list of messages to the topic
 await sender.send_messages(messages)
 print("Sent a list of 5 messages")

```

#### 5. Ajoutez une méthode pour envoyer un lot de messages.

Python

```

async def send_batch_message(sender):
 # Create a batch of messages
 async with sender:
 batch_message = await sender.create_message_batch()
 for _ in range(10):
 try:
 # Add a message to the batch

batch_message.add_message(ServiceBusMessage("Message inside a
ServiceBusMessageBatch"))
 except ValueError:
 # ServiceBusMessageBatch object reaches max_size.
 # New ServiceBusMessageBatch object can be created
here to send more data.
 break
 # Send the batch of messages to the topic
 await sender.send_messages(batch_message)
 print("Sent a batch of 10 messages")

```

#### 6. Créez un client Service Bus, puis un objet expéditeur de rubrique pour envoyer des messages.

Python

```
async def run():
 # Create a Service Bus client using the credential.
 async with ServiceBusClient(
 fully_qualified_namespace=FULLY_QUALIFIED_NAMESPACE,
 credential=credential,
 logging_enable=True) as servicebus_client:
 # Get a Topic Sender object to send messages to the topic
 sender =
 servicebus_client.get_topic_sender(topic_name=TOPIC_NAME)
 async with sender:
 # Send one message
 await send_single_message(sender)
 # Send a list of messages
 await send_a_list_of_messages(sender)
 # Send a batch of messages
 await send_batch_message(sender)
 # Close credential when no longer needed.
 await credential.close()

asyncio.run(run())
print("Done sending messages")
print("-----")
```

## Réception des messages d'un abonnement

L'exemple de code suivant vous montre comment recevoir les messages d'un abonnement. Ce code reçoit continuellement de nouveaux messages jusqu'à ce qu'il n'en reçoive plus pendant cinq (`max_wait_time`) secondes.

Ouvrez votre éditeur favori, tel que [Visual Studio Code](#), créez un fichier `recv.py` et ajoutez-y le code suivant.

Sans mot de passe (recommandé)

1. Comme dans l'exemple d'envoi, ajoutez des instructions `import`, définissez des constantes que vous devez remplacer par vos propres valeurs et définissez des informations d'identification.

Python

```
import asyncio
from azure.servicebus.aio import ServiceBusClient
from azure.identity.aio import DefaultAzureCredential
```

```
FULLY_QUALIFIED_NAMESPACE = "FULLY_QUALIFIED_NAMESPACE"
SUBSCRIPTION_NAME = "SUBSCRIPTION_NAME"
TOPIC_NAME = "TOPIC_NAME"

credential = DefaultAzureCredential()
```

- Créez un client Service Bus, puis un objet récepteur d'abonnement pour recevoir les messages.

Python

```
async def run():
 # create a Service Bus client using the credential
 async with ServiceBusClient(
 fully_qualified_namespace=FULLY_QUALIFIED_NAMESPACE,
 credential=credential,
 logging_enable=True) as servicebus_client:

 async with servicebus_client:
 # get the Subscription Receiver object for the
 subscription
 receiver =
 servicebus_client.get_subscription_receiver(topic_name=TOPIC_NAME,
 subscription_name=SUBSCRIPTION_NAME, max_wait_time=5)
 async with receiver:
 received_msgs = await
 receiver.receive_messages(max_wait_time=5, max_message_count=20)
 for msg in received_msgs:
 print("Received: " + str(msg))
 # complete the message so that the message is
 removed from the subscription
 await receiver.complete_message(msg)
 # Close credential when no longer needed.
 await credential.close()
```

- Appelez la méthode `run`.

Python

```
asyncio.run(run())
```

## Exécuter l'application

Ouvrez une invite de commandes comprenant Python dans son chemin, puis exécutez le code pour envoyer et recevoir les messages d'un abonnement sous une rubrique.

```
shell
```

```
python send.py; python recv.py
```

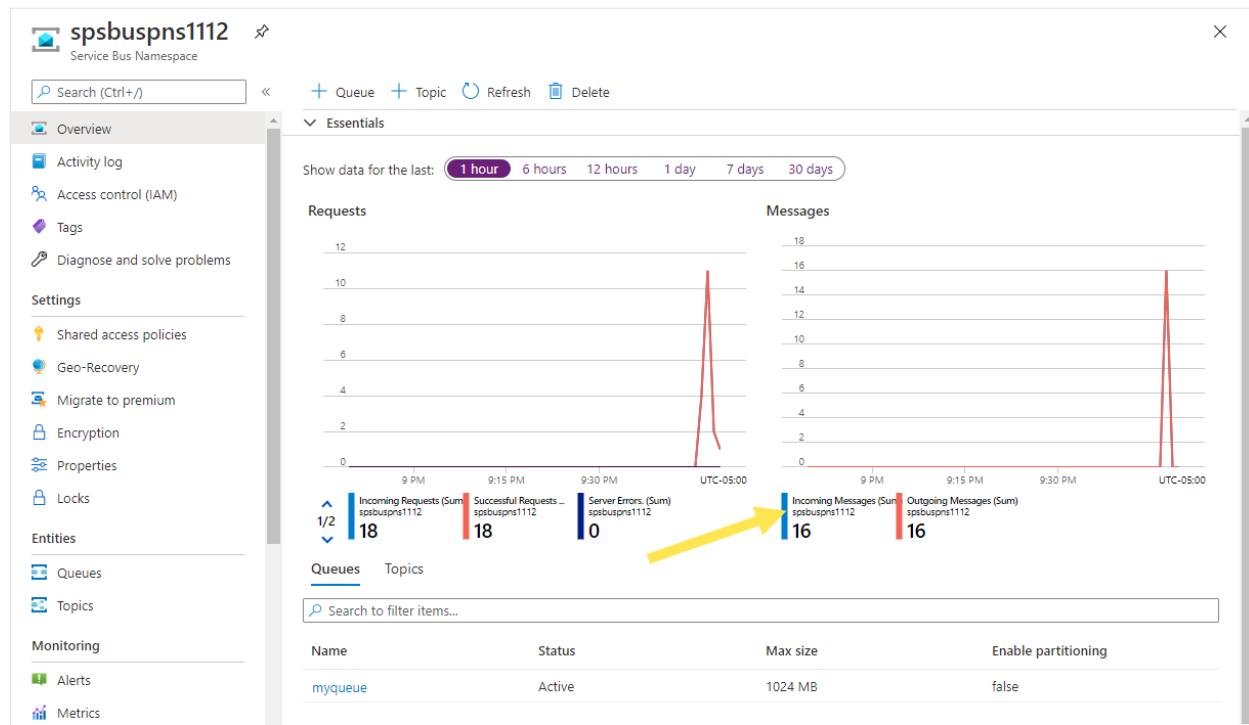
Vous devez normalement voir la sortie suivante :

Console

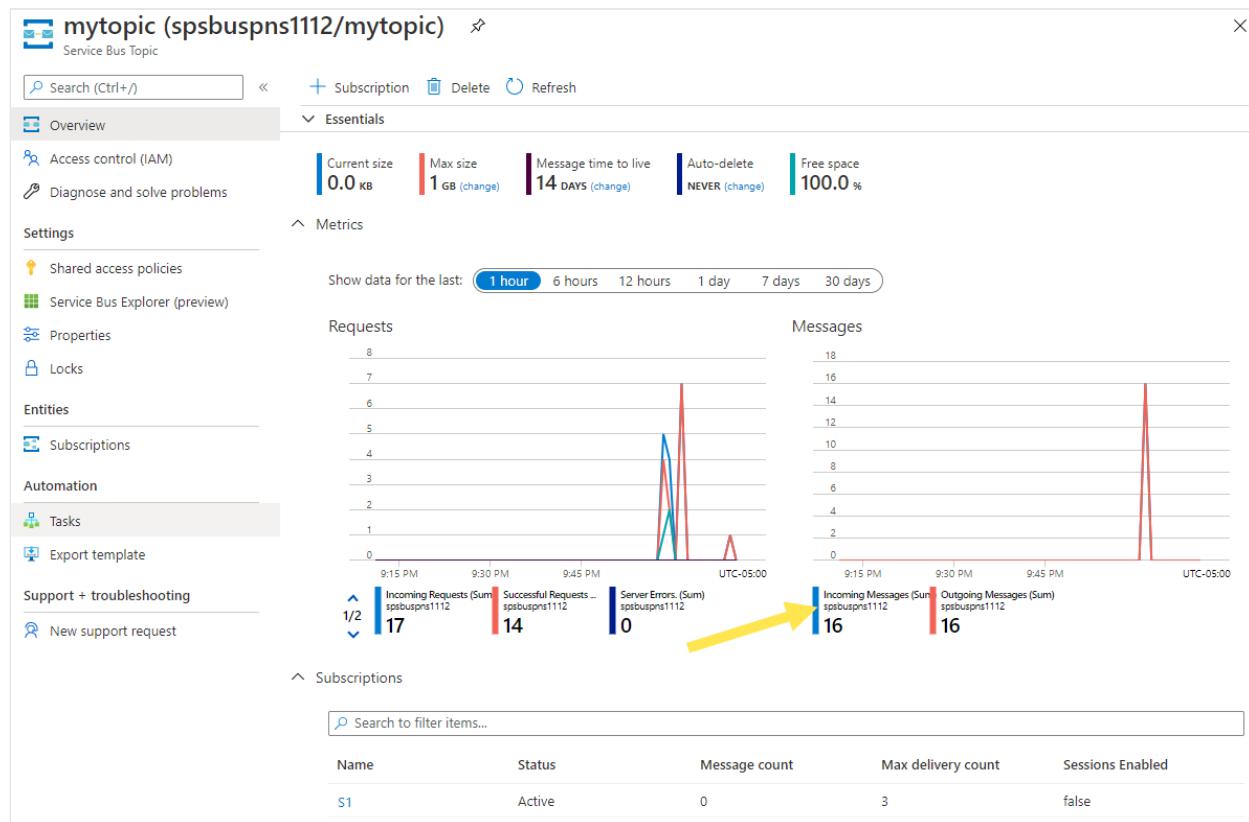
```
Sent a single message
Sent a list of 5 messages
Sent a batch of 10 messages
Done sending messages

Received: Single Message
Received: Message in list
Received: Message inside a ServiceBusMessageBatch
```

Dans le portail Azure, accédez à votre espace de noms Service Bus. Dans la page **Vue d'ensemble**, vérifiez que le nombre de messages **entrants et sortants** est égal à 16. Si ces chiffres ne s'affichent pas, actualisez la page après quelques minutes.



Sélectionnez la rubrique dans le volet inférieur pour afficher la page **Rubrique Service Bus** de votre rubrique. Dans cette page, vous devez voir trois messages entrants et trois messages sortants dans le graphique **Messages**.



Dans cette page, si vous sélectionnez un abonnement, vous accédez à la page **Abonnement Service Bus**. Elle indique entre autres le nombre de messages actifs et le nombre de messages de lettres mortes. Dans cet exemple, tous les messages ont été reçus ; le nombre de messages actifs est donc égal à zéro.

The screenshot shows the Azure Service Bus Subscription Overview page for 'S1 (spsbuspns112/mytopic/S1)'. The 'Essentials' section displays various configuration parameters and current message counts. A modal dialog titled 'FILTERS' is open, showing a single filter named '\$Default' of type 'SqlFilter'. The 'Active message count' is highlighted with a yellow arrow.

Parameter	Value
Max delivery count	3 (change)
Message time to live	14 DAYS (change)
Message lock duration	30 SECONDS (change)
Auto-delete after idle for	14 DAYS (change)
Active message count	0 MESSAGES
Dead-letter message count	0 MESSAGES
Transfer message count	0 MESSAGES
Scheduled message count	0 MESSAGES
Transfer message count	0 MESSAGES

Si vous commentez le code de réception, le nombre de messages actifs sera égal à 16.

The screenshot shows the same Azure Service Bus Subscription Overview page as before, but now the 'Active message count' has been modified to 16 messages. A yellow arrow points to this value. The rest of the interface remains identical to the first screenshot.

## Étapes suivantes

Voir la documentation et les exemples suivants :

- [Bibliothèque de client Azure Service Bus pour Python](#) ↗
- [Exemples](#) ↗.
  - Le dossier **sync\_samples** contient des exemples qui montrent comment interagir avec Service Bus de manière synchrone. Dans ce guide de démarrage rapide, vous avez utilisé cette méthode.
  - Le dossier **async\_samples** contient des exemples qui montrent comment interagir avec Service Bus de manière asynchrone.
- [Documentation de référence sur azure-servicebus](#)

# Démarrage rapide : Bibliothèque de client Stockage Blob Azure pour Python

Article • 25/03/2023

Commencez à utiliser la bibliothèque de client Stockage Blob Azure pour Python pour gérer les objets blob et les conteneurs. Suivez ces étapes pour installer le package et essayer l'exemple de code pour des tâches de base dans une application console interactive.

[Documentation de référence API](#) | [Code source de la bibliothèque](#) | [Package \(PyPI\)](#) | [Exemples](#)

## Prérequis

- Compte Azure avec un abonnement actif : [créez gratuitement un compte](#).
- Compte de stockage Azure : [créez un compte de stockage](#)
- [Python](#) 3.7+

## Configuration

Cette section vous guide tout au long de la préparation d'un projet à utiliser avec la bibliothèque de client Stockage Blob Azure pour Python.

### Créer le projet

Créez une application Python nommée *blob-quickstart*.

1. Dans une fenêtre de console (telle que PowerShell ou Bash), créez un répertoire pour le projet :

```
Console
mkdir blob-quickstart
```

2. Basculez vers le répertoire *blob-quickstart* nouvellement créé :

```
Console
cd blob-quickstart
```

## Installer les packages

À partir du répertoire du projet, installez les packages pour les bibliothèques de client Stockage Blob Azure et Azure Identity à l'aide de la commande `pip install`. Le package `azure-identity` est nécessaire pour les connexions sans mot de passe aux services Azure.

Console

```
pip install azure-storage-blob azure-identity
```

## Configurer le framework d'application

À partir du répertoire du projet, procédez comme suit pour créer la structure de base de l'application :

1. Ouvrez un nouveau fichier texte dans votre éditeur de code.
2. Ajoutez des instructions `import`, créez la structure du programme et incluez une gestion des exceptions de base comme indiqué ci-dessous.
3. Enregistrez le nouveau fichier sous `blob-quickstart.py` dans le répertoire `blob-quickstart`.

Python

```
import os, uuid
from azure.identity import DefaultAzureCredential
from azure.storage.blob import BlobServiceClient, BlobClient,
ContainerClient

try:
 print("Azure Blob Storage Python quickstart sample")

 # Quickstart code goes here

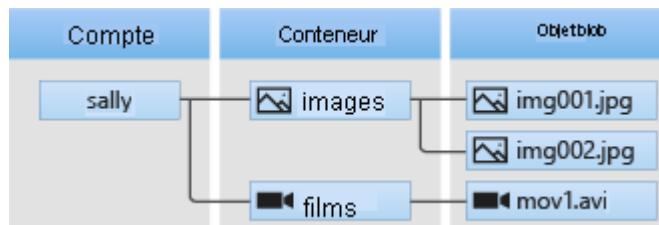
except Exception as ex:
 print('Exception:')
 print(ex)
```

## Modèle objet

Stockage Blob Azure est optimisé pour stocker des quantités massives de données non structurées. Les données non structurées sont des données qui n'obéissent pas à un modèle ou une définition de données en particulier, comme des données texte ou binaires. Le stockage Blob offre trois types de ressources :

- Le compte de stockage
- Un conteneur dans le compte de stockage.
- Un blob dans le conteneur

Le diagramme suivant montre la relation entre ces ressources :



Utilisez les classes Python suivantes pour interagir avec ces ressources :

- [BlobServiceClient](#): La classe `BlobServiceClient` vous permet de manipuler les ressources de stockage Azure et les conteneurs blob.
- [ContainerClient](#) : La classe `ContainerClient` vous permet de manipuler des conteneurs de stockage Azure et leurs blobs.
- [BlobClient](#) : La classe `BlobClient` vous permet de manipuler des blobs de stockage Azure.

## Exemples de code

Ces exemples d'extraits de code vous montrent comment effectuer les tâches suivantes avec la bibliothèque cliente Stockage Blob Azure pour Python :

- [S'authentifier auprès d'Azure et autoriser l'accès aux données blob](#)
- [Créer un conteneur](#)
- [Charger des objets blob sur un conteneur](#)
- [Lister les objets blob d'un conteneur](#)
- [Télécharger des objets blob](#)
- [Supprimer un conteneur](#)

## S'authentifier auprès d'Azure et autoriser l'accès aux données blob

Les demandes d'application vers le Stockage Blob Azure doivent être autorisées.

L'utilisation de la classe `DefaultAzureCredential` fournie par la bibliothèque de client Azure Identity est l'approche recommandée pour implémenter des connexions sans mot de passe aux services Azure dans votre code, y compris le Stockage Blob.

Vous pouvez également autoriser les demandes vers le Stockage Blob Azure à l'aide de la clé d'accès au compte. Toutefois, cette approche doit être utilisée avec prudence. Les développeurs doivent être vigilants pour ne jamais exposer la clé d'accès dans un emplacement non sécurisé. Toute personne disposant de la clé d'accès est en mesure d'autoriser les demandes sur le compte de stockage et a accès efficacement à toutes les données. `DefaultAzureCredential` offre des avantages améliorés en matière de gestion et de sécurité par rapport à la clé de compte pour autoriser l'authentification sans mot de passe. Les deux options sont illustrées dans l'exemple suivant.

#### Sans mot de passe (recommandé)

`DefaultAzureCredential` prend en charge plusieurs méthodes d'authentification et détermine quelle méthode doit être utilisée au moment de l'exécution. Cette approche permet à votre application d'utiliser différentes méthodes d'authentification dans différents environnements (local ou production) sans implémenter de code spécifique à l'environnement.

L'ordre et les emplacements dans lesquels `DefaultAzureCredential` les informations d'identification sont disponibles dans la [vue d'ensemble de la bibliothèque d'identités Azure](#).

Par exemple, votre application peut s'authentifier à l'aide de vos informations d'identification de connexion Azure CLI lors du développement local. Votre application peut ensuite utiliser une [identité managée](#) une fois qu'elle a été déployée sur Azure. Aucune modification du code n'est requise pour cette transition.

## Attribuer des rôles à votre compte d'utilisateur Microsoft Entra

Lors du développement local, assurez-vous que le compte d'utilisateur qui accède aux données blob dispose des autorisations appropriées. Vous aurez besoin du **Contributeur aux données Blob de stockage** pour lire et écrire des données blob. Pour vous attribuer ce rôle, vous aurez besoin du rôle **Administrateur de l'accès utilisateur** ou d'un autre rôle qui inclut l'action **Microsoft.Authorization/roleAssignments/write**. Vous pouvez attribuer des rôles RBAC Azure à un utilisateur à l'aide du Portail Azure, Azure CLI ou Azure PowerShell. Vous pouvez en savoir plus sur les étendues disponibles pour les attributions de rôles dans la page [vue d'ensemble de l'étendue](#).

Dans ce scénario, vous allez attribuer des autorisations à votre compte d'utilisateur, étendues au compte de stockage, pour suivre le [Principe des priviléges minimum](#).

Cette pratique offre aux utilisateurs uniquement les autorisations minimales nécessaires et crée des environnements de production plus sécurisés.

L'exemple suivant affecte le rôle **Contributeur aux données Blob du stockage** à votre compte d'utilisateur, qui fournit à la fois un accès en lecture et en écriture aux données d'objet blob dans votre compte de stockage.

### ⓘ Important

Dans la plupart des cas, la propagation de l'attribution de rôle dans Azure peut prendre une ou deux minutes, mais dans de rares cas, cela peut prendre jusqu'à huit minutes. Si vous recevez des erreurs d'authentification lorsque vous exécutez votre code pour la première fois, patientez quelques instants et réessayez.

Azure portal

1. Dans le Portail Azure, recherchez votre compte de stockage à l'aide de la barre de recherche principale ou de la navigation gauche.
2. Dans la page vue d'ensemble du compte de stockage, sélectionnez **Contrôle d'accès (IAM)** dans le menu de gauche.
3. Sur la page **Contrôle d'accès (IAM)**, sélectionnez l'onglet **Attributions de rôles**.
4. Sélectionnez **+ Ajouter** dans le menu supérieur, puis **Ajouter une attribution de rôle** dans le menu déroulant résultant.

The screenshot shows the 'Access Control (IAM)' blade for a storage account named 'identitymigrationstorage'. The left sidebar lists various account management sections. The 'Access Control (IAM)' section is highlighted with a red box. In the main content area, there's a search bar and a toolbar with buttons for 'Add', 'Download role assignments', 'Edit columns', 'Refresh', 'Remove', and 'Get feedback?'. A dropdown menu is open over the 'Add' button, with 'Add role assignment' highlighted by a red box. Below the toolbar, there are sections for 'My access', 'Check access', and a search bar. To the right, there are three cards: 'Grant access to this resource' (with a 'Add role assignment' button), 'View deny assignments' (with a 'View' button), and another 'View deny assignments' card with a magnifying glass icon and a 'Learn more' link.

5. Utilisez la zone de recherche pour filtrer les résultats sur le rôle souhaité.

Pour cet exemple, recherchez *Contributeur aux données Blob du stockage*, sélectionnez le résultat correspondant, puis choisissez **Suivant**.

6. Sous **Attribuer l'accès à**, sélectionnez **Utilisateur, groupe ou principal de service**, puis sélectionnez + **Sélectionner des membres**.

7. Dans la boîte de dialogue, recherchez votre nom d'utilisateur Microsoft Entra (généralement votre adresse e-mail *user@domain*), puis choisissez **Sélectionner** en bas de la boîte de dialogue.

8. Sélectionnez **Vérifier + affecter** pour accéder à la page finale, puis **Vérifier + attribuer** à nouveau pour terminer le processus.

## Se connecter et connecter le code d'application à Azure à l'aide de DefaultAzureCredential

Vous pouvez autoriser l'accès aux données dans votre compte de stockage en procédant comme suit :

1. Vérifiez que vous êtes authentifié avec le même compte Microsoft Entra auquel vous avez attribué le rôle sur votre compte de stockage. Vous pouvez vous authentifier via Azure CLI, Visual Studio Code ou Azure PowerShell.

Azure CLI

Connectez-vous à Azure via Azure CLI à l'aide de la commande suivante :

Azure CLI

`az login`

2. Pour utiliser `DefaultAzureCredential`, veillez à ce que le package `azure-identity` soit installé et à ce que la classe soit importée :

Python

```
from azure.identity import DefaultAzureCredential
```

3. Ajoutez ce code dans le bloc `try`. Quand le code est exécuté sur votre station de travail locale, `DefaultAzureCredential` utilise les informations

d'identification de développeur de l'outil prioritaire auquel vous êtes connecté pour l'authentification auprès d'Azure, par exemple Azure CLI ou Visual Studio Code.

```
Python

account_url = "https://<storageaccountname>.blob.core.windows.net"
default_credential = DefaultAzureCredential()

Create the BlobServiceClient object
blob_service_client = BlobServiceClient(account_url,
credential=default_credential)
```

4. Veillez à mettre à jour le nom du compte de stockage dans l'URI de votre objet `BlobServiceClient`. Le nom du compte de stockage se trouve sur la page vue d'ensemble du Portail Azure.

Home >

**identitymigrationstorage** Storage account

Search (Ctrl+ /) Upload Open in Explorer Delete Move Refresh

Overview

Activity log Tags Diagnose and solve problems Access Control (IAM) Data migration Events Storage browser

Essentials

Resource group ( <a href="#">move</a> )	: alexw-identity-revamp
Location	: East US
Primary/Secondary Location	: Primary: East US, Secondary: West US
Subscription ( <a href="#">move</a> )	: C&L Cross Service Content Team Testing
Subscription ID	:
Disk state	: Primary: Available, Secondary: Available
Tags ( <a href="#">edit</a> )	: Click here to add tags

### ⓘ Notes

Lorsqu'il est déployé sur Azure, ce même code peut être utilisé pour autoriser les demandes adressées à Stockage Azure à partir d'une application s'exécutant dans Azure. Toutefois, vous devez activer l'identité managée sur votre application dans Azure. Configurez ensuite votre compte de stockage pour autoriser cette identité managée à se connecter. Pour obtenir des instructions détaillées sur la configuration de cette connexion entre les services Azure, consultez le didacticiel d'authentification à partir d'applications hébergées sur Azure .

## Créez un conteneur.

Choisissez un nom pour le nouveau conteneur. Le code ci-dessous ajoute une valeur UUID au nom du conteneur pour garantir son unicité.

### ⓘ Important

Les noms de conteneurs doivent être en minuscules. Pour plus d'informations sur l'affectation de noms aux conteneurs et objets blob, consultez [Affectation de noms et références aux conteneurs, objets blob et métadonnées](#).

Appelez la méthode `create_container` pour créer le conteneur dans votre compte de stockage de manière effective.

Ajoutez ce code à la fin du bloc `try` :

Python

```
Create a unique name for the container
container_name = str(uuid.uuid4())

Create the container
container_client = blob_service_client.create_container(container_name)
```

Pour en savoir plus sur la création d'un conteneur et explorer d'autres exemples de code, consultez [Créer un conteneur de blobs avec Python](#).

## Charger des objets blob sur un conteneur

L'extrait de code suivant :

1. Crée un répertoire local où seront stockés les fichiers de données.
2. Crée un fichier texte dans le répertoire local.
3. Obtient une référence à un objet `BlobClient` en appelant la méthode `get_blob_client` sur le `BlobServiceClient` à partir de la section [Créer un conteneur](#).
4. Charge le fichier texte local dans l'objet Blob en appelant la méthode `upload_blob`.

Ajoutez ce code à la fin du bloc `try` :

Python

```
Create a local directory to hold blob data
local_path = "./data"
os.mkdir(local_path)
```

```

Create a file in the local data directory to upload and download
local_file_name = str(uuid.uuid4()) + ".txt"
upload_file_path = os.path.join(local_path, local_file_name)

Write text to the file
file = open(file=upload_file_path, mode='w')
file.write("Hello, World!")
file.close()

Create a blob client using the local file name as the name for the blob
blob_client = blob_service_client.get_blob_client(container=container_name,
blob=local_file_name)

print("\nUploading to Azure Storage as blob:\n\t" + local_file_name)

Upload the created file
with open(file=upload_file_path, mode="rb") as data:
 blob_client.upload_blob(data)

```

Pour en savoir plus sur le chargement de blobs et explorer d'autres exemples de code, consultez [Charger un blob avec Python](#).

## Créer la liste des objets blob d'un conteneur

Répertoriez les objets Blob dans le conteneur en appelant la méthode [list\\_blobs](#). Dans ce cas, un seul objet blob a été ajouté au conteneur. Il n'y a donc qu'un objet blob répertorié.

Ajoutez ce code à la fin du bloc `try` :

Python

```

print("\nListing blobs...")

List the blobs in the container
blob_list = container_client.list_blobs()
for blob in blob_list:
 print("\t" + blob.name)

```

Pour en savoir plus sur les listes de blobs et explorer d'autres exemples de code, consultez [Lister des blobs avec Python](#).

## Télécharger des objets blob

Téléchargez l'objet Blob créé précédemment en appelant la méthode [download\\_blob](#). L'exemple de code ajoute le suffixe « DOWNLOAD » au nom de fichier afin que vous

puissiez voir les deux fichiers dans votre système de fichiers local.

Ajoutez ce code à la fin du bloc `try` :

Python

```
Download the blob to a local file
Add 'DOWNLOAD' before the .txt extension so you can see both files in the
data directory
download_file_path = os.path.join(local_path, str.replace(local_file_name,
 '.txt', 'DOWNLOAD.txt'))
container_client = blob_service_client.get_container_client(container=
 container_name)
print("\nDownloading blob to \n\t" + download_file_path)

with open(file=download_file_path, mode="wb") as download_file:
 download_file.write(container_client.download_blob(blob.name).readall())
```

Pour en savoir plus sur le téléchargement de blobs et explorer d'autres exemples de code, consultez [Télécharger un blob avec Python](#).

## Supprimer un conteneur

Le code suivant nettoie les ressources créées par l'application en supprimant l'ensemble du conteneur avec la méthode `delete_container`. Si vous voulez, vous pouvez aussi supprimer les fichiers locaux.

L'application s'interrompt pour une entrée de l'utilisateur en appelant `input()` avant de supprimer l'objet blob, le conteneur et les fichiers locaux. Vérifiez que les ressources ont été créées correctement avant de les supprimer.

Ajoutez ce code à la fin du bloc `try` :

Python

```
Clean up
print("\nPress the Enter key to begin clean up")
input()

print("Deleting blob container...")
container_client.delete_container()

print("Deleting the local source and downloaded files...")
os.remove(upload_file_path)
os.remove(download_file_path)
os.rmdir(local_path)
```

```
print("Done")
```

Pour en savoir plus sur la suppression d'un conteneur et explorer d'autres exemples de code, consultez [Supprimer et restaurer un conteneur de blobs avec Python](#).

## Exécuter le code

Cette application crée un fichier de test dans votre dossier local et le charge dans Stockage Blob Azure. L'exemple liste ensuite le ou les objets blob du conteneur et télécharge le fichier sous un nouveau nom. Vous pouvez comparer les anciens fichiers avec les nouveaux.

Accédez au répertoire contenant le fichier *blob-quickstart.py*, puis exécutez la commande `python` suivante pour exécuter l'application :

Console

```
python blob-quickstart.py
```

La sortie de l'application est similaire à l'exemple suivant (valeurs UUID omises pour une meilleure lisibilité) :

Sortie

```
Azure Blob Storage Python quickstart sample

Uploading to Azure Storage as blob:
 quickstartUUID.txt

Listing blobs...
 quickstartUUID.txt

Downloading blob to
 ./data/quickstartUUIDDOWNLOAD.txt

Press the Enter key to begin clean up

Deleting blob container...
Deleting the local source and downloaded files...
Done
```

Avant de commencer le processus de nettoyage, vérifiez la présence des deux fichiers dans votre dossier *data*. Vous pouvez les comparer et constater qu'ils sont identiques.

# Nettoyer les ressources

Une fois que vous avez vérifié les fichiers et terminé le test, appuyez sur la touche Entrée pour supprimer les fichiers de test avec le conteneur que vous avez créé dans le compte de stockage. Vous pouvez également utiliser [Azure CLI](#) pour supprimer des ressources.

## Étapes suivantes

Dans ce démarrage rapide, vous avez appris à charger, télécharger et répertorier des objets blob avec Python.

Pour afficher des exemples d'applications de stockage blob, passez à :

### Bibliothèque Stockage Blob Azure pour Python - exemples

- Pour plus d'informations, consultez [Bibliothèques de client Stockage Blob Azure pour Python](#).
- Pour des tutoriels, des exemples, des guides de démarrage rapide et autre documentation, visitez [Azure pour les développeurs Python](#).

# Démarrage rapide : bibliothèque cliente Stockage File d'attente Azure pour Python

Article • 29/06/2023

Familiarisez-vous avec la bibliothèque de client Storage File d'attente Azure pour Python. Le Stockage File d'attente Azure est un service permettant de stocker un grand nombre de messages dans le but de les récupérer et de les traiter plus tard. Suivez les étapes suivantes pour installer le package et essayer un exemple de code pour les tâches de base.

[Documentation de référence de l'API](#) | [Code source bibliothèqueC ↗](#) | [Package \(Index package Python\) ↗](#) | [Exemples](#)

Utilisez la bibliothèque de client Stockage File d'attente Azure pour Python afin d'effectuer les opérations suivantes :

- Créer une file d'attente
- Ajouter des messages à une file d'attente
- Afficher un aperçu des messages d'une file d'attente
- Mettre à jour un message dans une file d'attente
- Obtention de la longueur de la file d'attente
- Recevoir les messages d'une file d'attente
- Supprimer des messages d'une file d'attente
- Suppression d'une file d'attente

## Prérequis

- Abonnement Azure : [créez-en un gratuitement ↗](#)
- Compte de stockage Azure : [créez un compte de stockage](#)
- [Python ↗](#) 3.7+

## Configuration

Cette section vous guide tout au long de la préparation d'un projet à utiliser avec la bibliothèque de client Stockage File d'attente Azure pour Python.

## Créer le projet

Créez une application Python nommée *queues-quickstart*.

1. Dans une fenêtre de console (telle que cmd, PowerShell ou Bash), créez un nouveau répertoire pour le projet.

```
Console
mkdir queues-quickstart
```

2. Basculez dans le répertoire *queues-quickstart* créé.

```
Console
cd queues-quickstart
```

## Installer les packages

Dans le répertoire de l'application, installez le package de la bibliothèque cliente Stockage File d'attente Azure pour Python à l'aide de la commande `pip install`. Le package **azure-identity** est nécessaire pour les connexions sans mot de passe aux services Azure.

```
Console
pip install azure-storage-queue azure-identity
```

## Configurer le framework d'application

1. Ouvrez un nouveau fichier texte dans votre éditeur de code
2. Ajoutez les instructions `import`
3. Créez la structure du programme, y compris la gestion des exceptions de base

Voici le code :

```
Python

import os, uuid
from azure.identity import DefaultAzureCredential
from azure.storage.queue import QueueServiceClient, QueueClient,
QueueMessage, BinaryBase64DecodePolicy, BinaryBase64EncodePolicy

try:
```

```
print("Azure Queue storage - Python quickstart sample")
Quickstart code goes here
except Exception as ex:
 print('Exception:')
 print(ex)
```

4. Enregistrez le nouveau fichier sous `queues-quickstart.py` dans le répertoire `queues-quickstart`.

## Authentification auprès d'Azure

Les requêtes d'application vers les Services Azure doivent être autorisées. L'utilisation de la classe `DefaultAzureCredential` fournie par la bibliothèque de client Azure Identity est l'approche recommandée pour implémenter des connexions sans mot de passe aux services Azure dans votre code.

Vous pouvez également autoriser directement les requêtes adressées aux services Azure à l'aide de mots de passe, de chaînes de connexion ou d'autres informations d'identification. Toutefois, cette approche doit être utilisée avec prudence. Les développeurs doivent être vigilants pour ne jamais exposer les secrets dans un emplacement non sécurisé. Toute personne ayant accès au mot de passe ou à la clé secrète est en mesure de s'authentifier. `DefaultAzureCredential` offre des avantages améliorés en matière de gestion et de sécurité par rapport à la clé de compte pour autoriser l'authentification sans mot de passe. Les deux options sont illustrées dans l'exemple suivant.

### Sans mot de passe (recommandé)

`DefaultAzureCredential` est une classe fournie par la bibliothèque de client Azure Identity pour Python. Pour en savoir plus sur `DefaultAzureCredential`, consultez la vue d'ensemble de [DefaultAzureCredential](#). `DefaultAzureCredential` prend en charge plusieurs méthodes d'authentification et détermine quelle méthode doit être utilisée au moment de l'exécution. Cette approche permet à votre application d'utiliser différentes méthodes d'authentification dans différents environnements (local ou production) sans implémenter de code spécifique à l'environnement.

Par exemple, votre application peut s'authentifier à l'aide de vos informations d'identification de connexion Visual Studio Code lors du développement local, puis utiliser une [identité managée](#) une fois qu'elle a été déployée sur Azure. Aucune modification du code n'est requise pour cette transition.

Lors du développement localement, assurez-vous que le compte d'utilisateur qui accède aux données de file d'attente dispose des autorisations appropriées. Vous aurez besoin du **Contributeur aux données de file d'attente de stockage** pour lire et écrire des données blob. Pour vous attribuer ce rôle, vous aurez besoin du rôle **Administrateur de l'accès utilisateur** ou d'un autre rôle qui inclut l'action **Microsoft.Authorization/roleAssignments/write**. Vous pouvez attribuer des rôles RBAC Azure à un utilisateur à l'aide du Portail Azure, Azure CLI ou Azure PowerShell. Vous pouvez en savoir plus sur les étendues disponibles pour les attributions de rôles dans la page [vue d'ensemble de l'étendue](#).

Dans ce scénario, vous allez attribuer des autorisations à votre compte d'utilisateur, étendues au compte de stockage, pour suivre le [Principe des priviléges minimum](#). Cette pratique offre aux utilisateurs uniquement les autorisations minimales nécessaires et crée des environnements de production plus sécurisés.

L'exemple suivant attribue le rôle **Contributeur aux données file d'attente du stockage** à votre compte d'utilisateur, qui fournit à la fois un accès en lecture et en écriture aux données file d'attente dans votre compte de stockage.

#### Important

Dans la plupart des cas, la propagation de l'attribution de rôle dans Azure peut prendre une ou deux minutes, mais dans de rares cas, cela peut prendre jusqu'à huit minutes. Si vous recevez des erreurs d'authentification lorsque vous exécutez votre code pour la première fois, patientez quelques instants et réessayez.

#### Azure portal

1. Dans le Portail Azure, recherchez votre compte de stockage à l'aide de la barre de recherche principale ou de la navigation gauche.
2. Dans la page vue d'ensemble du compte de stockage, sélectionnez **Contrôle d'accès (IAM)** dans le menu de gauche.
3. Sur la page **Contrôle d'accès (IAM)**, sélectionnez l'onglet **Attributions de rôles**.
4. Sélectionnez **+ Ajouter** dans le menu supérieur, puis **Ajouter une attribution de rôle** dans le menu déroulant résultant.

The screenshot shows the Azure Storage Access Control (IAM) interface. On the left, there's a sidebar with various options like Overview, Activity log, Tags, Diagnose and solve problems, and Access Control (IAM). The Access Control (IAM) option is highlighted with a red box. The main area has a search bar at the top. Below it, there are buttons for '+ Add', 'Download role assignments', 'Edit columns', 'Refresh', 'Remove', and 'Got feedback?'. A dropdown menu is open, showing 'Add role assignment' highlighted with a red box. Other options in the dropdown include 'Add co-administrator', 'Roles', 'Deny assignments', and 'Classic administrators'. To the right, there are sections for 'My access', 'Check access', 'Grant access to this resource', and 'View deny assignments'. The 'Add role assignment' button is also present here.

5. Utilisez la zone de recherche pour filtrer les résultats sur le rôle souhaité. Pour cet exemple, recherchez *Contributeur aux données file d'attente du stockage*, sélectionnez le résultat correspondant, puis choisissez **Suivant**.
6. Sous **Attribuer l'accès à**, sélectionnez **Utilisateur, groupe ou principal de service**, puis sélectionnez + **Sélectionner des membres**.
7. Dans la boîte de dialogue, recherchez votre nom d'utilisateur Microsoft Entra (généralement votre adresse e-mail *user@domain*), puis choisissez **Sélectionner** en bas de la boîte de dialogue.
8. Sélectionnez **Vérifier + affecter** pour accéder à la page finale, puis **Vérifier + attribuer** à nouveau pour terminer le processus.

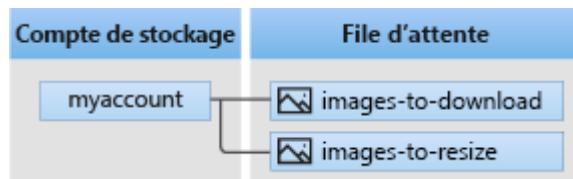
## Modèle objet

Stockage File d'attente Azure est un service permettant de stocker un grand nombre de messages. La taille maximale d'un message de file d'attente est de 64 Ko. Une file d'attente peut contenir des millions de messages, dans la limite de la capacité totale d'un compte de stockage. Les files d'attente sont couramment utilisées pour créer un backlog de travail à traiter de façon asynchrone. Le Stockage File d'attente offre trois types de ressources :

- **Compte de stockage** : Tous les accès à Azure Storage passent par un compte de stockage. Pour plus d'informations sur les comptes de stockage, consultez [Vue d'ensemble des comptes de stockage](#).

- **File d'attente** : une file d'attente contient un ensemble de messages. Tous les messages doivent être dans une file d'attente. Notez que le nom de la file d'attente doit être en minuscules. Pour plus d'informations sur l'affectation de noms à des files d'attente, consultez [Affectation de noms pour les files d'attente et les métadonnées](#).
- **Message** : message dans n'importe quel format d'une taille maximale de 64 Ko. Un message peut rester dans la file d'attente pendant un maximum de 7 jours. Pour les versions du 29 juillet 2017 ou ultérieures, la durée de vie maximale peut être n'importe quel nombre positif, ou -1 indiquant que le message n'expire pas. Si ce paramètre est omis, la valeur par défaut de la durée de vie est de sept jours.

Le diagramme suivant montre la relation entre ces ressources.



Utilisez les classes Python suivantes pour interagir avec ces ressources :

- [QueueServiceClient](#) : `QueueServiceClient` vous permet de gérer toutes les files d'attente de votre compte de stockage.
- [QueueClient](#) : la classe `QueueClient` vous permet de gérer et de manipuler une file d'attente individuelle et ses messages.
- [QueueMessage](#) : la classe `QueueMessage` représente les objets individuels retournés lors de l'appel de `receive_messages` dans une file d'attente.

## Exemples de code

Ces exemples d'extraits de code vous montrent comment effectuer les actions suivantes avec la bibliothèque de client Stockage File d'attente Azure pour Python :

- [Autoriser l'accès et créer un objet client](#)
- [Créer une file d'attente](#)
- [Ajouter des messages à une file d'attente](#)
- [Afficher un aperçu des messages d'une file d'attente](#)
- [Mettre à jour un message dans une file d'attente](#)
- [Obtention de la longueur de la file d'attente](#)
- [Recevoir les messages d'une file d'attente](#)
- [Supprimer des messages d'une file d'attente](#)
- [Supprimer une file d'attente](#)

Sans mot de passe (recommandé)

## Autoriser l'accès et créer un objet client

Vérifiez que vous êtes authentifié avec le même compte Microsoft Entra auquel vous avez attribué le rôle. Vous pouvez vous authentifier via Azure CLI, Visual Studio Code ou Azure PowerShell.

Azure CLI

Connectez-vous à Azure via Azure CLI à l'aide de la commande suivante :

Azure CLI

```
az login
```

Une fois authentifié, vous pouvez créer et autoriser un objet `QueueClient` à l'aide de `DefaultAzureCredential` pour accéder aux données de file d'attente dans le compte de stockage. `DefaultAzureCredential` découvre et utilise automatiquement le compte avec lequel vous vous êtes connecté à l'étape précédente.

Pour autoriser à l'aide de `DefaultAzureCredential`, vérifiez que vous avez ajouté le package `azure-identity`, comme décrit dans [Installer les packages](#). Veillez également à ajouter l'instruction import suivante dans le fichier `queues-quickstart.py` :

Python

```
from azure.identity import DefaultAzureCredential
```

Choisissez un nom pour la file d'attente et créez une instance de la classe `QueueClient`, en utilisant `DefaultAzureCredential` pour l'autorisation. Nous utilisons cet objet client pour créer et interagir avec la ressource de file d'attente dans le compte de stockage.

### ⓘ Important

Les noms de file d'attente peuvent contenir uniquement des lettres minuscules, des chiffres et des traits d'union, et doivent commencer par une lettre ou un nombre. Chaque trait d'union doit être précédé et suivi d'un

caractère autre qu'un tiret. Le nom doit avoir entre 3 et 63 caractères. Pour plus d'informations sur le nommage des files d'attente, consultez [Nommage des files d'attente et des métadonnées](#).

Ajoutez le code suivant à l'intérieur du bloc `try` et veillez à remplacer la valeur d'espace `<storage-account-name>` réservé :

Python

```
print("Azure Queue storage - Python quickstart sample")

Create a unique name for the queue
queue_name = "quickstartqueues-" + str(uuid.uuid4())

account_url = "https://<storageaccountname>.queue.core.windows.net"
default_credential = DefaultAzureCredential()

Create the QueueClient object
We'll use this object to create and interact with the queue
queue_client = QueueClient(account_url, queue_name=queue_name
, credential=default_credential)
```

Les messages de la file d'attente sont stockés sous forme de texte. Si vous souhaitez stocker des données binaires, configurez les fonctions d'encodage et de décodage en Base64 avant de placer un message dans la file d'attente.

Vous pouvez configurer les fonctions d'encodage et de décodage en Base64 lors de la création de l'objet client :

Python

```
Setup Base64 encoding and decoding functions
base64_queue_client = QueueClient.from_connection_string(
 conn_str=connect_str, queue_name=q_name,
 message_encode_policy =
BinaryBase64EncodePolicy(),
 message_decode_policy =
BinaryBase64DecodePolicy()
)
```

## Créer une file d'attente

Avec l'objet `QueueClient`,appelez la méthode `create_queue` pour créer la file d'attente dans votre compte de stockage.

Ajoutez ce code à la fin du bloc `try` :

```
Python

print("Creating queue: " + queue_name)

Create the queue
queue_client.create_queue()
```

## Ajouter des messages à une file d'attente

L'extrait de code suivant ajoute des messages à la file d'attente en appelant la méthode `send_message`. Il enregistre également l'élément `QueueMessage` retourné par le troisième appel de `send_message`. Le `saved_message` est utilisé pour mettre à jour le contenu du message ultérieurement dans le programme.

Ajoutez ce code à la fin du bloc `try` :

```
Python

print("\nAdding messages to the queue...")

Send several messages to the queue
queue_client.send_message(u"First message")
queue_client.send_message(u"Second message")
saved_message = queue_client.send_message(u"Third message")
```

## Afficher un aperçu des messages d'une file d'attente

Affichez un aperçu des messages de la file d'attente en appelant la méthode `peek_messages`. Cette méthode récupère un ou plusieurs messages du début de la file d'attente, mais ne modifie pas la visibilité du message.

Ajoutez ce code à la fin du bloc `try` :

```
Python

print("\nPeek at the messages in the queue...")

Peek at messages in the queue
peeked_messages = queue_client.peek_messages(max_messages=5)

for peeked_message in peeked_messages:
 # Display the message
 print("Message: " + peeked_message.content)
```

## Mettre à jour un message dans une file d'attente

Mettez à jour le contenu d'un message en appelant la méthode [update\\_message](#). Cette méthode peut changer le contenu et le délai d'expiration de la visibilité d'un message. Le contenu du message doit être une chaîne encodée en UTF-8 d'une taille maximale de 64 Ko. Avec le nouveau contenu, transmettez les valeurs du message qui a été enregistré dans le code. Les valeurs `saved_message` identifient le message à mettre à jour.

Python

```
print("\nUpdating the third message in the queue...")

Update a message using the message saved when calling send_message
earlier
queue_client.update_message(saved_message,
pop_receipt=saved_message.pop_receipt, \
content="Third message has been updated")
```

## Obtention de la longueur de la file d'attente

Vous pouvez obtenir une estimation du nombre de messages dans une file d'attente.

La méthode [get\\_queue\\_properties](#) retourne les propriétés de file d'attente, y compris `approximate_message_count`.

Python

```
properties = queue_client.get_queue_properties()
count = properties.approximate_message_count
print("Message count: " + str(count))
```

Le résultat est approximatif car des messages peuvent être ajoutés ou supprimés après que le service a répondu à votre demande.

## Réception des messages d'une file d'attente

Vous pouvez téléchargez les messages ajoutés en appelant la méthode [receive\\_messages](#).

Ajoutez ce code à la fin du bloc `try` :

Python

```
print("\nReceiving messages from the queue...")

Get messages from the queue
messages = queue_client.receive_messages(max_messages=5)
```

Lorsque vous appelez la méthode `receive_messages`, vous pouvez éventuellement spécifier une valeur pour `max_messages`, qui correspond au nombre de messages à récupérer dans la file d'attente. La valeur par défaut est de 1 message et la valeur maximale est de 32 messages. Vous pouvez également spécifier une valeur pour `visibility_timeout`, qui masque les messages aux autres opérations pendant la période d'expiration. La valeur par défaut est 30 secondes.

## Supprimer des messages d'une file d'attente

Supprimez les messages de la file d'attente une fois qu'ils sont reçus et traités. Dans ce cas, le traitement affiche simplement le message sur la console.

Avant de traiter et de supprimer les messages, l'application s'interrompt dans l'attente d'une entrée de l'utilisateur en appelant `input`. Vérifiez dans votre [portail Azure](#) que les ressources ont été créées correctement avant d'être supprimées. Les messages qui ne sont pas supprimés explicitement redeviennent visibles dans la file d'attente et peuvent éventuellement être de nouveau traités.

Ajoutez ce code à la fin du bloc `try` :

Python

```
print("\nPress Enter key to 'process' messages and delete them from the
queue...")
input()

for msg_batch in messages.by_page():
 for msg in msg_batch:
 # "Process" the message
 print(msg.content)
 # Let the service know we're finished with
 # the message and it can be safely deleted.
 queue_client.delete_message(msg)
```

## Suppression d'une file d'attente

Le code suivant nettoie les ressources créées par l'application en supprimant la file d'attente avec la méthode `delete_queue`.

Ajoutez ce code à la fin du bloc `try` et enregistrez le fichier :

Python

```
print("\nPress Enter key to delete the queue...")
input()

Clean up
print("Deleting queue...")
queue_client.delete_queue()

print("Done")
```

## Exécuter le code

Cette application crée trois messages et les ajoute à une file d'attente Azure. Le code liste les messages dans la file d'attente, puis les récupère et les supprime avant de supprimer la file d'attente.

Dans la fenêtre de votre console, accédez au répertoire contenant le fichier `queues-quickstart.py`, puis utilisez la commande `python` suivante pour exécuter l'application.

Console

```
python queues-quickstart.py
```

La sortie de l'application ressemble à l'exemple suivant :

Sortie

```
Azure Queue Storage client library - Python quickstart sample
Creating queue: quickstartqueues-<UUID>
```

```
Adding messages to the queue...
```

```
Peek at the messages in the queue...
```

```
Message: First message
```

```
Message: Second message
```

```
Message: Third message
```

```
Updating the third message in the queue...
```

```
Receiving messages from the queue...
```

```
Press Enter key to 'process' messages and delete them from the queue...
```

```
First message
```

```
Second message
```

```
Third message has been updated
```

```
Press Enter key to delete the queue...
```

```
Deleting queue...
```

```
Done
```

Quand l'application s'interrompt avant de recevoir des messages, vérifiez votre compte de stockage dans le [portail Azure](#). Vérifiez que les messages se trouvent dans la file d'attente.

Appuyez sur la touche `Enter` pour recevoir et supprimer les messages. Quand vous y êtes invité, réappuyez sur la touche `Enter` pour supprimer la file d'attente et terminer la démonstration.

## Étapes suivantes

Dans ce guide de démarrage rapide, vous avez appris à créer une file d'attente et à y ajouter des messages à l'aide de code Python. Ensuite, vous avez appris à afficher un aperçu des messages, à les récupérer et à les supprimer. Enfin, vous avez appris à supprimer une file d'attente de messages.

Pour obtenir des tutoriels, des exemples, des guides de démarrage rapide et d'autres documents, visitez :

### Azure for Python developers

(Azure pour les développeurs Python)

- Pour obtenir des exemples de code associés utilisant des Kits de développement logiciel (SDK) Python version 2 déconseillés, consultez l'article [Exemples de code utilisant Python version 2](#).
- Pour plus d'informations, consultez les [bibliothèques Stockage Azure pour Python](#).
- Pour d'autres exemples d'applications Stockage File d'attente Azure, consultez [Exemples de bibliothèques de client Stockage File d'attente Azure pour Python](#).

# Stratégie de contrôle de version pour les services Azure, les kits SDK et les outils CLI

Article • 13/12/2023

La plupart des services Azure vous permettent de contrôler et de gérer par programmation leurs ressources avec des API REST. Les services évoluent via de nouvelles versions publiées de leurs API avec différents contrats qui ajoutent de nouvelles fonctionnalités et/ou modifient leurs comportements.

Cet article décrit la stratégie que les équipes Azure service, SDK et CLI utilisent pour la gestion des versions des API REST Azure. Bien que les équipes Azure effectuent tous les efforts nécessaires pour respecter cette stratégie, les écarts peuvent parfois se produire.

## Contrôle des versions du service

Chaque version publiée d'une API est identifiée par une valeur de date au `YYYY-MM-DD` format appelé `api-version`. Les versions plus récentes ont des dates ultérieures.

Toutes les opérations d'API nécessitent que les clients spécifient une version d'API valide pour le service via le `api-version` paramètre de chaîne de requête dans l'URL. Par exemple : <https://management.azure.com/subscriptions?api-version=2020-01-01>. Les kits de développement logiciel (SDK) et les outils clients incluent automatiquement la `api-version` valeur. Pour plus d'informations, consultez la [section Des kits sdk client et des versions](#) de service plus loin dans cet article.

En règle générale, les versions de service publiées restent disponibles et prises en charge pendant de nombreuses années, même lorsque les versions plus récentes deviennent disponibles. Dans la plupart des cas, la seule fois où vous devez adopter une nouvelle version de service dans le code existant consiste à tirer parti des nouvelles fonctionnalités.

## Versions stables

La plupart des versions de service publiées sont *des versions stables*. Les versions stables sont compatibles descendantes, ce qui signifie que tout code que vous écrivez qui s'appuie sur une version d'un service peut adopter une version stable plus récente sans

nécessiter de modifications de code pour maintenir l'exactitude ou les fonctionnalités existantes.

## Versions de modification cassants

Une *version* de modification cassant d'un service n'est pas rétrocompatible. L'adoption d'une version de modification cassante dans le code client existant peut nécessiter des modifications de code pour s'assurer que le client se comporte exactement comme il l'a fait lors du ciblage de la version précédente.

Les versions de modification cassants sont rares, annoncées via la documentation et sont généralement précédées de la publication d'une version préliminaire. La publication d'une version de modification cassant peut entraîner la mise hors service éventuelle des versions stables existantes, qui reste disponible pendant au moins trois ans après les versions de modification cassants. Pour les changements cassants publiés en raison de problèmes de sécurité ou de conformité, les versions de service stables existantes peuvent rester disponibles pendant un an ou moins en fonction de la gravité du problème.

En raison de l'innovation rapide et des développements dans l'IA, les services pilotés par l'IA peuvent avoir une disponibilité minimale réduite d'un an. Chaque service publiera sa stratégie de modification cassant.

Tout service Azure dépendant d'un composant autre que Microsoft peut réduire sa stratégie de support pour qu'elle corresponde à celle de la stratégie du composant. Toute modification cassant due à cette modification est liée à la stratégie du fournisseur du composant montrant la date à laquelle le composant n'est plus pris en charge.

## Préversions

Parfois, Microsoft publie une *préversion* d'un service pour recueillir des commentaires sur les modifications proposées et les nouvelles fonctionnalités. Les versions de service en préversion sont identifiées avec le suffixe `-preview` dans leur `api-version` (par exemple, `2022-07-07-preview`).

À moins d'introduire explicitement une modification cassant de la version stable précédente, les nouvelles versions d'évaluation incluent toutes les fonctionnalités de la version stable la plus récente et ajoutent de nouvelles fonctionnalités en préversion. Toutefois, entre les versions préliminaires, un service peut interrompre l'une des fonctionnalités d'aperçu nouvellement ajoutées.

Les préversions ne sont pas destinées à une utilisation à long terme. Chaque fois qu'une nouvelle version stable ou préliminaire d'un service devient disponible, les versions en préversion existantes peuvent devenir indisponibles dès 90 jours après la disponibilité de la nouvelle version. Utilisez les versions en préversion uniquement dans les situations où vous développez activement sur de nouvelles fonctionnalités de service et que vous êtes prêt à adopter une nouvelle version sans préversion peu après sa publication. Si certaines fonctionnalités d'une préversion sont publiées dans une nouvelle version stable, les fonctionnalités restantes toujours en préversion seront généralement publiées dans une nouvelle version d'évaluation.

## Kits de développement logiciel (SDK) clients et versions de service

Les [kits de développement logiciel](#) (SDK) Azure visent à éliminer le contrôle de version du service en tant que préoccupation lors de l'écriture de code. Chaque KIT SDK est composé de bibliothèques clientes, une pour chaque service, et chaque version de bibliothèque cliente cible une seule version du service sur laquelle elle s'appuie.

Lorsque vous utilisez un SDK pour accéder à un service Azure, tirer parti des nouvelles versions et fonctionnalités nécessite généralement la mise à niveau de la version de bibliothèque cliente utilisée par l'application. De nouvelles versions stables des services sont accompagnées de nouvelles versions point des bibliothèques clientes. Pour les nouvelles versions de modification cassants, les nouvelles bibliothèques clientes sont publiées en tant que versions de version point ou versions principales. Le type de mise en production dépend de la nature du changement du service et de la façon dont la bibliothèque est en mesure de l'adapter. Seules les bibliothèques clientes bêta utilisent des versions préliminaires du service.

Les bibliothèques clientes du SDK prennent en charge la substitution manuelle de la version du service. La substitution de la version de service par défaut d'une bibliothèque cliente est un scénario avancé et peut entraîner un comportement inattendu. Si vous utilisez cette fonctionnalité, testez soigneusement votre application pour vous assurer qu'elle fonctionne comme vous le souhaitez.

## Outils en ligne de commande Azure

Comme pour les Kits de développement logiciel (SDK), les outils en ligne de commande Azure (y compris [Azure CLI](#) et [Azure PowerShell](#)) sont conçus pour permettre l'utilisation des services de gestion Azure sans tenir compte des versions. L'accès aux nouvelles fonctionnalités de service nécessite souvent une nouvelle version d'un outil. Les

nouvelles versions d'outils compatibles descendantes sont publiées mensuellement. Les versions avec des modifications cassantes sont publiées environ deux fois par an, ou si nécessaire pour résoudre les problèmes de sécurité critiques.

Les outils en ligne de commande Azure peuvent parfois exposer des fonctionnalités en préversion. Ces commandes sont marquées avec une `Preview` étiquette et affichent un avertissement indiquant une prise en charge limitée et des modifications potentielles dans les futures versions d'outils.

## Étapes suivantes

- [Spécifications de l'API REST Azure ↗](#)
- [Instructions relatives à l'API REST Microsoft ↗](#)
- [Instructions générales du Kit de développement logiciel \(SDK\) Azure ↗](#)