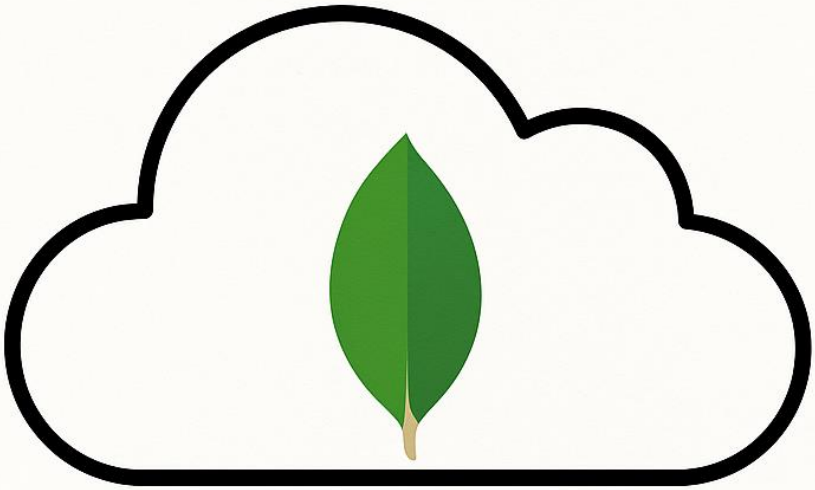
Déploiement de MongoDB sur AWS



aws

SOMMAIRE :

1. Introduction

1.1 Contexte du projet

L'entreprise souhaite moderniser son infrastructure technique afin d'améliorer la performance, la fiabilité et la gestion de ses données. Actuellement, la base de données MongoDB est hébergée de manière locale ou sur un environnement limité, ce qui pose plusieurs contraintes :

- une difficulté à absorber l'augmentation du volume de données,
- un manque de flexibilité pour adapter les ressources selon les besoins,
- un risque accru de pannes, de pertes de données ou d'indisponibilité,
- une maintenance technique lourde (mises à jour, sécurité, gestion des serveurs).

Dans ce contexte, la migration vers une plateforme cloud comme **Amazon Web Services (AWS)** représente une opportunité stratégique. AWS propose une large gamme de services permettant d'héberger, de sécuriser, de sauvegarder et de superviser efficacement des bases de données, tout en optimisant les coûts et la scalabilité.

Le projet consiste donc à **étudier les solutions AWS adaptées à MongoDB**, ainsi que les options de déploiement possibles via des technologies modernes telles que **Docker** et **Amazon ECS**.

Cette documentation a pour but de synthétiser les recherches nécessaires pour orienter l'entreprise vers la solution la mieux adaptée à ses besoins.

1.2 Objectifs de la documentation

Cette documentation vise à fournir une base de compréhension claire et complète concernant :

- **Les fondements du cloud computing**, ses avantages et ses différences avec une infrastructure traditionnelle.
- **La démarche pour créer et configurer un compte AWS**, étape essentielle avant toute migration.
- **La tarification AWS**, afin de comprendre les coûts liés au stockage, au calcul et aux transferts de données.
- **Les solutions AWS permettant d'héberger MongoDB**, notamment :
    - Amazon DocumentDB (compatible MongoDB),
    - et le déploiement de MongoDB dans un conteneur Docker via Amazon ECS.
- **Les mécanismes de sauvegarde et de surveillance** disponibles sur AWS pour garantir la sécurité et la disponibilité des données.

L'objectif final est d'aider l'entreprise à prendre une décision éclairée concernant la migration de sa base de données MongoDB vers le cloud, en présentant les options, les avantages et les limites de chaque solution.

2. Présentation du Cloud

2.1 Différence entre ordinateur, serveur et cloud

Pour comprendre l'intérêt d'une migration vers le cloud, il est essentiel de distinguer trois notions fondamentales : l'ordinateur personnel, le serveur et le cloud.

Ordinateur

Un ordinateur (PC ou laptop) est une machine destinée à un usage individuel. Il dispose de ressources limitées (CPU, RAM, stockage) et n'est pas conçu pour :

- fonctionner en continu 24/7,
- gérer un grand nombre d'utilisateurs simultanément,
- garantir une haute disponibilité ou une redondance des données.

Il n'est donc pas adapté pour héberger des applications critiques ou des bases de données professionnelles.

Serveur

Un serveur est une machine plus puissante, dédiée au traitement de requêtes venant de plusieurs utilisateurs ou applications. Il est conçu pour :

- fonctionner en permanence,
- supporter de fortes charges,
- offrir un stockage plus fiable,
- gérer des réseaux et services complexes.

Cependant, un serveur physique **nécessite une maintenance importante** : mise à jour, sécurité, remplacement du matériel, gestion du réseau, refroidissement, alimentation, etc.

Cloud

Le cloud n'est pas un type de machine, mais **un ensemble de serveurs distants** gérés par un fournisseur (AWS, Google Cloud, Azure) et accessibles via Internet.

Le cloud permet :

- d'accéder à des ressources informatiques illimitées (stockage, puissance, bases de données),
- sans acheter de matériel,
- avec une disponibilité élevée,
- en ne payant que ce qui est utilisé.

En simplifiant :
Ordinateur = usage individuel

Serveur = hébergement local avec maintenance
Cloud = serveurs distants, flexibles, évolutifs et gérés pour vous

2.2 Pourquoi passer au cloud ?

Migrer vers le cloud présente de nombreux avantages pour une entreprise qui souhaite moderniser son infrastructure.

Scalabilité

Le cloud permet d'augmenter ou de réduire automatiquement les ressources selon les besoins :

- hausse de trafic,
- croissance de la base de données,
- nouveaux utilisateurs.

Pas besoin d'acheter du matériel supplémentaire.

Réduction des coûts

- Pas d'investissement initial dans des serveurs.
- Pas de maintenance physique.
- Paiement uniquement pour l'usage réel ("pay-as-you-go").
- Possibilité de réduire encore les coûts grâce au stockage intelligent et aux instances économiques.

Sécurité renforcée

Les fournisseurs comme AWS offrent :

- chiffrement des données,
- pare-feux intégrés,
- contrôle d'accès précis (IAM),
- surveillance continue.

Les données sont mieux protégées qu'avec une infrastructure interne mal sécurisée.

Haute disponibilité

Les données et services sont répliqués dans plusieurs centres de données.
Résultat :

moins de risques d'interruption,
meilleure résilience en cas de panne.

Agilité

Le cloud permet de déployer rapidement :

- des bases de données,
- des applications,
- de nouveaux environnements.

Ce qui accélère les projets et favorise l'innovation.

Accessibilité

Les données sont accessibles :

- à distance,
- à n'importe quel moment,
- par les personnes autorisées.

Idéal pour le télétravail, les équipes distribuées ou les applications en ligne.

2.3 Qu’est-ce que le stockage dans le cloud ?

Le stockage dans le cloud désigne le fait de stocker des données sur des serveurs distants accessibles via Internet plutôt que sur un disque local ou un serveur interne.

Avec ce modèle :

- le fournisseur (ex : AWS) gère et sécurise toute l'infrastructure,
- la capacité est quasi illimitée,
- les données sont dupliquées sur plusieurs équipements pour éviter toute perte,
- l’utilisateur peut augmenter ou réduire l’espace utilisé selon ses besoins.

Pourquoi c’est utile ?

- **Coûts maîtrisés** : pas de matériel à acheter, paiement à l’usage.
- **Évolutivité** : stockage extensible automatiquement.
- **Sécurité** : données chiffrées et protégées dans des centres certifiés.
- **Accès facile** : depuis n’importe où.

Types de stockage dans le cloud

- **Stockage objet (ex : Amazon S3)** : pour images, vidéos, sauvegardes, lacs de données.
- **Stockage fichier (ex : EFS)** : pour partager un système de fichiers entre plusieurs serveurs.
- **Stockage bloc (ex : EBS)** : pour les bases de données ou applications nécessitant une faible latence.

Le stockage cloud est un pilier central des architectures modernes, car il rend les données plus flexibles, plus durables et plus faciles à gérer.

3. Création d’un compte AWS

3.1 Étapes d’inscription

La création d’un compte AWS est une étape essentielle avant de pouvoir utiliser les services cloud proposés par Amazon. Le processus est rapide et se fait en ligne.

Étape 1 : Accéder au site AWS

- Se rendre sur : **https://aws.amazon.com/**
- Cliquer sur **“Create an AWS Account”** ou **“Créer un compte AWS”**.

Étape 2 : Informations du compte

Le formulaire demande :

- une **adresse email valide**,
- un **mot de passe sécurisé**,
- un **nom de compte** (ex : nom de l’entreprise ou de l’utilisateur).

AWS utilise cet email pour valider l’ouverture du compte et gérer la sécurité.

Étape 3 : Vérification d’identité

AWS vérifie l’identité via :

- un numéro de téléphone,
- l’envoi d’un **code de validation par SMS ou appel vocal**.

Cette étape garantit que le compte appartient bien à la personne inscrite.

Étape 4 : Informations de paiement

Pour activer le compte, AWS demande une **carte bancaire**.
Elle sert à :

- valider l'identité,
- payer les services utilisés si l’on dépasse le **Free Tier**.

→ Aucun prélèvement n’est effectué tant que les services gratuits ne sont pas dépassés.

Étape 5 : Choisir une formule de support

Trois options sont proposées :

- **Basic (gratuit)**
- Developer
- Business

Pour un premier usage, l'option **Basic** est suffisante.

Une fois toutes les étapes validées, le compte est créé et prêt à être utilisé.

3.2 Accès à la console AWS

Une fois le compte créé, l'utilisateur peut se connecter à la **Console de gestion AWS** via :

🔗 https://console.aws.amazon.com/

AWS propose une interface graphique permettant de gérer tous les services cloud sans ligne de commande.

Fonctionnalités principales de la console :

- accès aux services AWS (EC2, S3, RDS, ECS, IAM, etc.),
- affichage des ressources actives,
- gestion de la facturation et de la consommation,
- configuration de la sécurité (IAM, clés d'accès),
- création de machines virtuelles, de bases de données ou de conteneurs.

Premiers réglages recommandés :

- **Configurer la zone (région AWS)** → ex : *eu-west-3 (Paris)*
- **Activer l'authentification à deux facteurs (MFA)**
- **Créer un utilisateur IAM** pour éviter d'utiliser le compte root
- **Surveiller la facturation via AWS Billing Dashboard**

La console devient ensuite le point central pour gérer l'ensemble des ressources nécessaires au déploiement de MongoDB et de tous les autres services.

4. Tarification AWS

La tarification AWS repose sur un modèle flexible permettant aux entreprises de payer uniquement les ressources qu'elles consomment. Contrairement à une infrastructure traditionnelle où l'on doit acheter et maintenir du matériel, AWS offre une facturation à l'usage ainsi que des offres gratuites destinées aux tests et aux environnements de développement.

4.1 Modèle pay-as-you-go

Le modèle **pay-as-you-go** (paiement à l'usage) signifie que l'on paie uniquement les services réellement utilisés, sans engagement à long terme ni coût initial.

Principes clés :

- **Pas d'achat de matériel** : AWS gère toute l'infrastructure physique.
- **Aucun engagement** : possibilité d'arrêter ou modifier un service à tout moment.

- **Facturation à la consommation** : calculée en fonction :
  - du temps d'exécution (EC2, ECS, Lambda),
  - de la capacité utilisée (S3, EBS, DocumentDB),
  - du volume de données transférées.

Avantages du modèle pay-as-you-go :

- **Optimisation des coûts** : pas de sur-provisionnement nécessaire.
- **Flexibilité totale** : adapter les ressources en fonction du trafic et de la charge.
- **Contrôle budgétaire** : possibilité d'activer des alertes via AWS Billing.

Ce modèle est particulièrement adapté aux applications dont la charge varie dans le temps.

4.2 Free Tier

AWS propose un **Free Tier**, une offre gratuite qui permet de tester les services sans coût pendant une période limitée.

Types de Free Tier :

1. **Free Tier 12 mois**
   Valable pendant les 12 premiers mois après la création du compte.
2. **Offres gratuites permanentes**
   Certains services offrent une capacité gratuite à vie (ex : AWS Lambda, CloudWatch logs basiques).
3. **Offres d'essai ponctuelles**
   Gratuité temporaire lors de la sortie de nouvelles fonctionnalités.

Exemples de services inclus dans le Free Tier 12 mois :

- **750 heures/mois** d'instances EC2 t2.micro ou t3.micro
  (équivalent à un serveur allumé en continu).
- **5 Go de stockage Amazon S3**.
- **750 heures/mois** de bases RDS gratuites (MySQL, PostgreSQL).
- **25 Go** de stockage DynamoDB.

Amazon DocumentDB n'est pas inclus dans le Free Tier.

Objectif du Free Tier

Le Free Tier permet :

- d'apprendre à utiliser AWS,
- de tester des environnements,
- de prototyper un projet,
- sans générer de coûts tant que les limites ne sont pas dépassées.

5. AWS et MongoDB

AWS propose plusieurs solutions pour héberger ou utiliser MongoDB dans le cloud. Cependant, toutes ne sont pas directement compatibles, ce qui nécessite de bien comprendre les options disponibles avant toute migration.

5.1 Amazon RDS : limitations

Amazon RDS (Relational Database Service) est un service managé d'AWS permettant d'héberger plusieurs bases de données relationnelles comme :

- MySQL
- PostgreSQL
- MariaDB
- Oracle
- SQL Server
- Amazon Aurora

Limitation principale : MongoDB n'est pas supporté par RDS

MongoDB n'est pas une base relationnelle mais une base **NoSQL orientée documents**, ce qui la rend incompatible avec le moteur RDS.
Il n'existe donc **aucune option native** pour héberger MongoDB directement sur Amazon RDS.

Cela signifie que pour utiliser MongoDB sur AWS, il faut se tourner vers :

- **Amazon DocumentDB**, ou
- **MongoDB dans un environnement Docker géré par ECS**.

5.2 Amazon DocumentDB (compatible MongoDB)

Amazon DocumentDB est un service **managé** conçu par AWS pour être **compatible avec les API MongoDB**.
Cela signifie que les applications peuvent utiliser les mêmes drivers et commandes qu'avec MongoDB, mais le moteur interne n'est pas exactement le même que MongoDB Community.

Avantages principaux :

- **Service managé** : AWS gère les mises à jour, la maintenance et la sécurité.
- **Haute disponibilité** via une architecture Multi-AZ.
- **Scalabilité automatique** du stockage (jusqu'à plusieurs To).
- **Sauvegardes automatiques** et point-in-time restore.
- **Sécurité avancée** : VPC, chiffrement, IAM.

Limites :

- Pas 100 % compatible avec toutes les fonctionnalités avancées de MongoDB.
- Coût plus élevé que les solutions autogérées.

Quand utiliser DocumentDB ?

- Lorsque l'on veut un service **fiable, stable et sans maintenance lourde**.
- Quand l'équipe préfère déléguer la gestion de la base au cloud.
- Pour des environnements de production nécessitant haute disponibilité.

5.3 MongoDB dans un conteneur Docker (Amazon ECS)

Une alternative consiste à installer MongoDB soi-même dans un conteneur Docker, puis à déployer ce conteneur sur **Amazon ECS (Elastic Container Service)**.
Cette solution donne plus de liberté et peut être moins coûteuse, mais demande davantage de configuration et de maintenance.

Principe général :

1. Créer une image Docker contenant MongoDB (ou utiliser l'image officielle `mongo:latest`).
2. Envoyer cette image dans un registre AWS (Amazon ECR).
3. Déployer le conteneur dans un cluster ECS, avec :
    - **Fargate** (sans serveur à gérer), ou
    - **EC2** (instances à gérer soi-même).
4. Configurer un stockage persistant :
    - EBS (volume attaché à l'instance), ou
    - EFS (système de fichiers partagé).

Avantages :

- **Contrôle total** sur la configuration MongoDB.
- Possibilité d'utiliser n'importe quelle version ou plugin MongoDB.
- **Coût potentiellement plus bas** que DocumentDB si bien optimisé.

Inconvénients :

- Nécessite de gérer :
    - les mises à jour,
    - les sauvegardes,
    - la surveillance,
    - la scalabilité,
    - la sécurité.
- Demande plus de compétences techniques (Docker, ECS, réseau).

Quand utiliser MongoDB + ECS ?

- Pour un projet nécessitant une version spécifique de MongoDB.
- Pour des environnements personnalisés (plugins, configurations avancées).
- Pour optimiser les coûts dans des environnements maîtrisés.

6. Déploiement d'une instance MongoDB sur ECS

Déployer MongoDB sur Amazon ECS consiste à exécuter la base de données dans un conteneur Docker orchestré par AWS. Cette solution offre souplesse, isolation, portabilité et permet d'utiliser des versions spécifiques de MongoDB tout en profitant des services AWS.

Le déploiement se déroule en quatre grandes étapes : créer l'image Docker, la stocker dans ECR, configurer un cluster ECS, puis mettre en place un stockage persistant pour conserver les données.

6.1 Image Docker MongoDB

La première étape consiste à créer ou utiliser une image Docker contenant MongoDB.

Option 1 : Utiliser l'image officielle MongoDB

MongoDB fournit une image préconfigurée disponible sur Docker Hub :

```
docker pull mongo:latest
```

Cette image contient :

- le serveur MongoDB,
- les outils de base,
- une configuration standard adaptée aux environnements cloud.

Option 2 : Construire une image personnalisée

Pour ajouter des configurations spécifiques (plugins, scripts d'initialisation), on peut créer un fichier `Dockerfile` :

```
FROM mongo:latest
COPY init.js /docker-entrypoint-initdb.d/
```

Puis construire l'image :

```
docker build -t mongodb-custom .
```

L'image sera ensuite poussée vers le dépôt ECR (voir étape suivante).

6.2 Dépôt ECR

Amazon ECR (Elastic Container Registry) est un service permettant d'héberger les images Docker utilisées par ECS.

Étapes principales :

1. Créer un repository ECR

Depuis la console AWS :

- Accéder à *Elastic Container Registry*
- Cliquer sur “Create Repository”
- Nommer le dépôt (ex : `mongodb`)

2. Authentifier Docker à ECR

Avec AWS CLI :

```
aws ecr get-login-password | docker login --username AWS --password-stdin
<ID>.dkr.ecr.<region>.amazonaws.com
```

3. Tagger l’image

```
docker tag mongodb-custom:latest
<ID>.dkr.ecr.<region>.amazonaws.com/mongodb:latest
```

4. Envoyer l’image vers ECR

```
docker push <ID>.dkr.ecr.<region>.amazonaws.com/mongodb:latest
```

L’image est maintenant disponible pour être utilisée dans un service ECS.

6.3 Cluster ECS (Fargate ou EC2)

Amazon ECS (Elastic Container Service) orchestre et exécute les conteneurs.
Deux modes sont possibles :

Option A : Fargate (sans serveur à gérer)

Recommandé si l’on souhaite :

- éviter la gestion d’instances EC2,
- simplifier les mises à jour,
- bénéficier d’une isolation accrue.

Avantages :

- Pas de maintenance d’OS
- Déploiement très simple
- Scalabilité automatique

Inconvénients :

- Coût un peu plus élevé que EC2 pour des workloads constants

Option B : EC2 (instances gérées par l’utilisateur)

Recommandé si l’on veut :

- optimiser les coûts,

- contrôler entièrement les machines,
- utiliser des configurations avancées d'hébergement.

Avantages :

- Prix plus bas si instances Spot ou petites tailles
- Plus grande flexibilité

Inconvénients :

- Maintenance obligatoire (patchs, OS, etc.)

Étapes dans ECS :

1. Créer un cluster ECS

- Choisir **Fargate** ou **EC2**
- Configurer le réseau (VPC, sous-réseaux, sécurité)

2. Créer une Task Definition

Une *task definition* définit :

- l'image Docker utilisée (depuis ECR),
- les ressources allouées (CPU, RAM),
- les variables d'environnement (ex : `MONGO_INITDB_ROOT_USERNAME`),
- le stockage utilisé,
- les ports ouverts (ex : 27017).

3. Créer un Service ECS

Le service assure que MongoDB reste toujours en exécution.

On configure :

- le nombre de tâches (réplicas),
- le load balancer (si nécessaire),
- la stratégie de redémarrage,
- l'autoscaling éventuel.

6.4 Stockage persistant (EBS / EFS)

MongoDB doit conserver ses données en dehors du conteneur pour éviter toute perte lors d'un redémarrage ou d'une mise à jour.

Option 1 : EBS (Elastic Block Store)

Volume attaché à une instance EC2.

Avantages :

- rapide, faible latence
- idéal pour un MongoDB hébergé sur EC2

Limites :

- lié à une seule instance
- ne convient pas pour Fargate

Option 2 : EFS (Elastic File System)

Système de fichiers partagé et scalable, compatible avec EC2 et Fargate.

Avantages :

- multi-attach (peut être monté par plusieurs tâches ECS)
- résilient, scalable automatiquement
- recommandé pour Fargate

Limites :

- latence légèrement plus élevée que EBS

Pour MongoDB en production :

- **EC2 → EBS** est souvent préféré
- **Fargate → EFS** est la solution recommandée

Le répertoire `/data/db` de MongoDB doit être monté sur le volume persistant.

7. Sauvegardes et Surveillance

La sécurité et la fiabilité d'une base de données dans le cloud reposent sur deux piliers essentiels :
✔ des mécanismes de **sauvegarde** pour éviter toute perte de données,
✔ des outils de **surveillance** pour vérifier l'état et les performances du système.

AWS fournit plusieurs services permettant d'assurer ces deux aspects, tant pour DocumentDB que pour MongoDB déployé sur ECS.

7.1 Sauvegardes automatiques

Sauvegardes avec Amazon DocumentDB

Amazon DocumentDB étant un service managé, les sauvegardes sont **automatiquement gérées par AWS**, sans action particulière de l'utilisateur.

Fonctionnalités incluses :

- **Sauvegardes continues (Point-in-Time Recovery)** → restauration à un moment précis
- **Rétention configurable** de 1 à 35 jours
- **Snapshots manuels** pour figer l'état de la base
- Stockage des sauvegardes sur **Amazon S3**, avec une durabilité de 99,999999999 %

→ Solution idéale pour réduire les risques d'erreur humaine ou de perte accidentelle.

Sauvegardes pour MongoDB sur ECS

Avec MongoDB dans un conteneur, AWS ne gère plus la sauvegarde automatiquement : c'est à l'entreprise de mettre en place une stratégie.

Options recommandées :

1. Sauvegarde via mongodump

Effectuer un export complet des collections :

```
mongodump --out /backup
```

Puis envoyer le dossier `/backup` dans Amazon S3 pour stockage durable.

2. Snapshots EBS (si EC2 + EBS)

- Sauvegarde automatique du volume attaché au conteneur.
- Rapide, incrémental et moins lourd à gérer.
- Idéal lorsque MongoDB est déployé sur une instance EC2.

3. Backup via AWS Backup

Service permettant de gérer :

- des politiques de sauvegarde,
- des calendriers,
- la rétention automatique.

Compatible avec EFS → recommandé pour MongoDB sur **Fargate + EFS**.

7.2 Monitoring avec CloudWatch

AWS CloudWatch permet de surveiller en temps réel les performances et l'état de l'infrastructure.

CloudWatch pour DocumentDB

DocumentDB fournit nativement des métriques comme :

- **CPUUtilization**
- **DatabaseConnections**
- **FreeLocalStorage**
- **Read/Write IOPS**

CloudWatch peut :

- déclencher des alertes (CPU trop élevé…),
- générer des graphiques,
- être intégré avec SNS pour envoyer des emails ou SMS.

CloudWatch pour MongoDB sur ECS

Dans ECS, la surveillance se fait à deux niveaux :

1. Monitoring de l'infrastructure ECS

CloudWatch surveille :

- utilisation CPU des tâches,
- mémoire des conteneurs,
- état des services (RUNNING, STOPPED),
- événements du cluster.

2. Monitoring applicatif de MongoDB

Il est possible d'envoyer à CloudWatch :

- les logs de MongoDB (via CloudWatch Logs),
- des métriques personnalisées (ex : taux de requêtes, latence).

→ Cela permet d'être averti rapidement en cas d'erreur, de surcharge ou de comportement anormal.

7.3 Sécurité et bonnes pratiques

La sécurité des données est un élément crucial dans tout déploiement cloud. AWS fournit un ensemble d'outils et de mécanismes permettant de protéger MongoDB.

Principales bonnes pratiques :

1. Utiliser des réseaux privés (VPC)

- Héberger MongoDB dans des sous-réseaux privés.
- Interdire l'accès public direct à la base.

2. Contrôles d'accès avec IAM

- Ne jamais utiliser le compte root pour les opérations quotidiennes.

- Créer des rôles et politiques adaptés aux services ECS et ECR.
- Limiter les permissions au strict nécessaire.

3. Chiffrement des données

- Activer le chiffrement **au repos** (EBS, EFS, DocumentDB).
- Forcer le chiffrement **en transit** (TLS) entre les applications et MongoDB.

4. Groupes de sécurité stricts

- Autoriser uniquement les ports nécessaires (27017 pour MongoDB).
- Restreindre l'accès aux seules instances ou services autorisés.

5. Rotation des identifiants

- Utiliser AWS Secrets Manager pour stocker et faire tourner les mots de passe MongoDB.
- Éviter les identifiants en clair dans les fichiers ou conteneurs.

6. Journaux et audit

- Activer CloudTrail pour suivre toutes les actions effectuées sur les ressources AWS.
- Conserver l'historique des modifications appliquées au cluster ECS ou DocumentDB.

Cette combinaison de sauvegardes, de surveillance et de bonnes pratiques de sécurité permet de maintenir un environnement MongoDB fiable, résilient et conforme aux standards professionnels.

8. Conclusion

8.1 Synthèse des solutions AWS

La migration d'une base de données MongoDB vers AWS offre de nombreuses possibilités adaptées aux besoins de flexibilité, de sécurité et de scalabilité d'une entreprise moderne.

Trois approches principales ont été étudiées :

1. Amazon RDS

- Non compatible avec MongoDB.
- Exclu d'emblée pour les projets utilisant cette base.

2. Amazon DocumentDB (compatible MongoDB)

- Solution **managée**, hautement disponible et sécurisée.
- Idéale pour les environnements de production nécessitant :
    - des sauvegardes automatiques,
    - une maintenance simplifiée,

    - une forte résilience.
- Offre une compatibilité avec les API MongoDB tout en bénéficiant d'un moteur interne optimisé pour AWS.
- Recommandée pour les entreprises privilégiant la stabilité et la simplicité.

3. MongoDB dans un conteneur Docker sur ECS

- Solution flexible, personnalisable et potentiellement moins coûteuse.
- Nécessite la gestion des mises à jour, de la sécurité, des sauvegardes et de la supervision.
- Approche adaptée aux cas d'usage nécessitant :
    - une version spécifique de MongoDB,
    - des plugins,
    - des configurations avancées.
- ECS (Fargate ou EC2) permet un déploiement moderne, scalable et automatisé.

Les sauvegardes, la surveillance et la sécurité sont assurées à travers différents services AWS comme CloudWatch, AWS Backup, IAM, EBS/EFS, garantissant ainsi un environnement fiable et durable, quelle que soit la solution choisie.

8.2 Recommandations finales

À l'issue de cette étude, voici les recommandations pour orienter l'entreprise vers la solution la plus adaptée :

1. Pour une solution simple, stable et managée : choisir Amazon DocumentDB

- Maintenance minimale
- Haute disponibilité native
- Sauvegardes automatiques
- Sécurité renforcée
- Idéal pour les équipes souhaitant se concentrer sur les applications et non l'infrastructure

Cette solution est généralement la plus adaptée aux entreprises cherchant un environnement cloud professionnel, sécurisé et prêt à l'emploi.

2. Pour une solution flexible et optimisée en coût : choisir MongoDB sur ECS

- Permet d'utiliser la version exacte de MongoDB souhaitée
- Plus économique si l'on maîtrise bien l'infrastructure
- Convient aux équipes techniques expérimentées dans Docker et AWS
- Recommandé pour les environnements personnalisés ou les projets d'innovation

Cette option nécessite cependant une gestion plus importante : sauvegardes, surveillance, mises à jour et configuration du stockage persistant.

3. Renforcer la sécurité et la fiabilité

Quel que soit le choix technologique, il est essentiel de :

- configurer les bases dans des sous-réseaux privés,
- appliquer le chiffrement des données (au repos et en transit),
- utiliser IAM de manière stricte (principe du moindre privilège),
- mettre en place une politique de sauvegarde régulière,
- surveiller en continu la base via CloudWatch.

Conclusion générale

AWS offre un écosystème complet pour exécuter MongoDB dans le cloud avec un haut niveau de performance, de sécurité et de flexibilité.
Le choix entre DocumentDB et ECS dépend principalement :

- du niveau de contrôle souhaité,
- du budget,
- des compétences techniques disponibles,
- et des contraintes de maintenance ou de conformité.

Cette documentation fournit les éléments nécessaires pour effectuer un choix éclairé et planifier une migration sereine vers le cloud AWS.