Recherches AWS - Déploiement Cloud pour Healthcare MongoDB

Projet : Migration des données médicales vers MongoDB
Auteur : BEHRAM KORKUT - Data Engineer
Date : Octobre 2025
Contexte : Documentation de recherche pour déploiement cloud (Étape 3)

Table des Matières

1. Introduction : Pourquoi le Cloud ?

Pré-requis

Différence Ordinateur vs Server

Ordinateur (Client)

Rôle : Consommer des services

Caractéristiques :

```
Utilisé par 1 personne à la fois
Éteint la nuit / week-end
Interface graphique (écran, souris, clavier)
Performances modérées
Fiabilité standard
```

Exemples d'usage :

```
Navigation web
Bureautique
```

```
Jeux vidéo
Développement
```

Analogie : Le client d'un restaurant qui commande Serveur

Rôle : Fournir des services à plusieurs clients simultanément

Caractéristiques :

```
Utilisé par des centaines/milliers d'utilisateurs en même temps
Fonctionne 24/7/365 (toujours allumé)
Pas d'interface graphique (juste ligne de commande)
Performances élevées (CPU puissant, beaucoup de RAM)
Fiabilité maximale (redondance, backup)
Composants de qualité enterprise
```

Exemples d'usage :

```
Héberger un site web
Stocker une base de données
Envoyer des emails
Héberger une application
```

Analogie : La cuisine du restaurant qui prépare pour tous les clients

Tableau Comparatif

Critère | Ordinateur | Serveur

Utilisateurs | 1 | Plusieurs centaines

Disponibilité| 8h/jour | 24h/24

Localisation | Bureau/Maison | Datacenter climatisé

Coût | 500-2000€ | 3,000-50,000€

Maintenance | Occasionnelle | Professionnelle continue

Exemple | MacBook Pro | Dell PowerEdge R740

Pour Notre Projet Healthcare

Avant (On-Premise) :

```
Serveur physique dans un local
Coût : 3,000€ + maintenance
Risque : Panne = tout s'arrête
```

Après (Cloud AWS) :

```
"Serveurs virtuels" dans datacenter AWS
Coût : 70€/mois
Avantage : Si panne, AWS bascule automatiquement
```

Le cloud = Location de serveurs au lieu de les acheter ! 🚀

Métaphore Simple :

Ordinateur = Votre voiture personnelle Serveur = Bus public (transporte beaucoup de monde) Cloud = Uber/Taxi (vous louez le service sans acheter le véhicule)

2- Le "Cloud" C'est quoi exactement

Le cloud, ce n'est PAS magique, ce sont bien des serveurs physiques dans des datacenters ! :

```
┌──────────────────────────────────────────────────┐
│  Ce que les gens imaginent : ☁ "Le Cloud"        │
│  (quelque chose d'abstrait dans les airs)        │
└──────────────────────────────────────────────────┘

                      ≠


┌──────────────────────────────────────────────────┐
│  La RÉALITÉ :                                    │
│                                                  │
│  ┌──────────────┐  ┌──────────────┐  ┌──────────────┐
│  │ Datacenter   │  │ Datacenter   │  │ Datacenter   │
│  │   Paris      │  │  Francfort   │  │   Irlande    │
│  │              │  │              │  │              │
│  │ [Serveur 1]  │  │ [Serveur 50] │  │ [Serveur 99] │
│  │ [Serveur 2]  │  │ [Serveur 51] │  │ [Serveur 100]│
│  │ [Serveur 3]  │  │ [Serveur 52] │  │ [Serveur 101]│
│  │    ...       │  │    ...       │  │    ...       │
│  └──────────────┘  └──────────────┘  └──────────────┘
│                                                  │
└──────────────────────────────────────────────────┘
```

```
│  = Des MILLIERS de serveurs physiques réels                    │
└────────────────────────────────────────────────────────────────┘
```

Pourquoi On Appelle Ça "Cloud" ?

3 raisons principales :

1. Distribution Géographique

Vos données ne sont pas sur UN seul serveur, mais répliquées sur plusieurs :

Votre Base de Données MongoDB :

```
┌──────────────────────────────────────────────────────────────┐
│  Copie 1        Copie 2         Copie 3                      │
│  ↓              ↓               ↓                            │
│  Serveur A      Serveur B       Serveur C                    │
│  Paris AZ-A     Paris AZ-B      Paris AZ-C                   │
│                                                              │
│  + Backup →  Serveur D (Francfort)                           │
└──────────────────────────────────────────────────────────────┘
```

Avantage :

```
Si le serveur A tombe en panne → Bascule automatique sur B
Si tout Paris a un problème → Backup à Francfort
```

2. Abstraction de la Localisation

Vous ne savez pas (et vous n'avez pas besoin de savoir) :

```
Sur quel serveur physique exact vos données sont
Dans quelle salle du datacenter
Sur quel disque dur précisément
```

Vous dites juste :

Vous : "Je veux une base MongoDB"

AWS : "OK, je la crée sur 3 serveurs différents pour toi"

(mais tu n'as pas besoin de savoir lesquels)

C'est comme un nuage : Vous voyez le nuage, mais vous ne voyez pas les millions de gouttelettes d'eau qui le composent ! ☁️ 3. Élasticité (Expansion/Contraction)

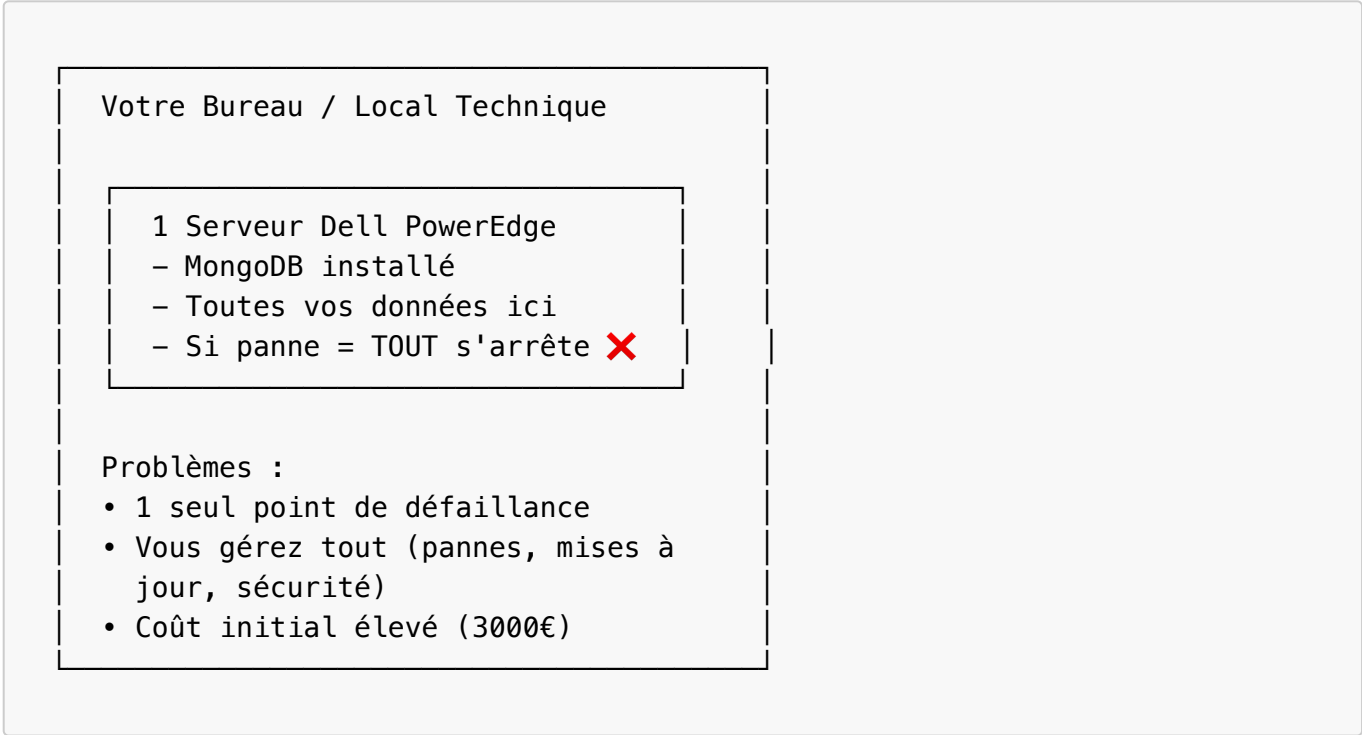
Comme un nuage qui change de forme :

Lundi 9h (peu d'utilisateurs) : ┌──────────────┐ | Serveur 1| → Vos données utilisent 1 serveur └──────────────┘

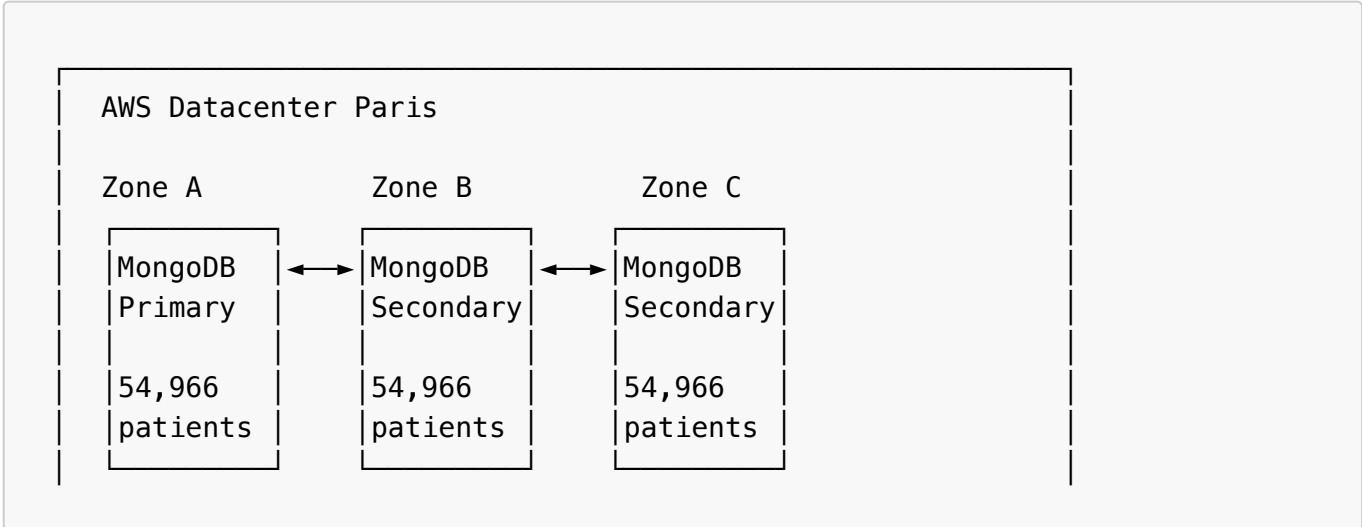
Vendredi 14h (pic d'activité) : ┌──────────────┐ ┌──────────────┐ ┌──────────────┐ | Serveur 1| | Serveur 2| | Serveur 3| → AWS ajoute automatiquement 2 serveurs └──────────────┘ └──────────────┘ └──────────────┘

Samedi 3h (nuit, peu d'activité) : ┌──────────────┐ | Serveur 1| → AWS retire les serveurs inutiles └──────────────┘

Vous payez uniquement ce que vous utilisez ! Exemple Concret : Votre Projet Healthcare Scénario On-Premise (Avant) :



Scénario Cloud AWS (Après) :



```
│     ↓                ↓                ↓                          │
│     └────────────────┴────────────────┘                           │
│            Synchronisation                                        │
│            en temps réel                                          │
│                                                                   │
│  + Backup automatique vers Francfort                              │
└──────────────────────────────────────────────────────────────────┘
```

Avantages : ✅ Si Zone A en panne → Bascule auto sur Zone B (30 sec) ✅ Si tout Paris en panne → Restore depuis Francfort (4h) ✅ AWS gère TOUT (maintenance, sécurité, mises à jour) ✅ Vous payez 70€/mois (vs 3000€ initial + maintenance)

Les Datacenters AWS : La Réalité Un Datacenter AWS, c'est :

Taille :

```
Bâtiment de la taille d'un hangar d'avion
Plusieurs milliers de m²
```

Contenu :

```
Des milliers de serveurs (racks de 2 mètres de haut)
Climatisation industrielle (serveurs = chaleur)
Alimentation électrique redondante (générateurs de secours)
Sécurité physique (gardes, biométrie, caméras)
Connexion Internet ultra-rapide (fibres optiques)
```

AWS a ~100 datacenters dans le monde, regroupés en 32 régions. Région Paris (eu-west-3) :

┌──┐ | Région AWS Paris (eu-west-3) | | | | ┌──────────────────────┐ ┌──────────────────────┐ | | | Datacenter 1 | | Datacenter 2 | | | | (Zone A) | | (Zone B) | | | | Localisation : | | Localisation : | | | | Secrète (sécurité| | Secrète | | | | mais probablement| | (différente de A)| | | | banlieue Paris) | | | | | | | | | | [5000+ serveurs] | | [5000+ serveurs] | | | └──────────────────────┘ └──────────────────────┘ | | | | ┌──────────────────────┐ | | | Datacenter 3 | | | | (Zone C) | | | | Localisation : | | | | Secrète | | | | (différente de | | | | A et B) | | | | | | | | [5000+ serveurs] | | | └──────────────────────┘ | | | | Distance entre zones : ~10-20 km | | Connexion : Fibre optique dédiée | └──┘

Pourquoi 3 zones séparées ?

```
Si inondation zone A → Zones B et C fonctionnent
Si coupure électrique zone B → Zones A et C fonctionnent
Si incendie zone C → Zones A et B fonctionnent
```

Probabilité que les 3 zones tombent en même temps : 0.00001% (quasi impossible) Métaphore Finale

On-Premise (Serveur local) :

= Avoir votre propre générateur électrique à la maison • Vous le possédez • Vous le maintenez • Si panne, vous réparez vous-même • Coût initial élevé

Cloud :

= Être branché sur le réseau électrique national • Vous ne possédez pas les centrales • EDF/Enedis gère tout • Si panne locale, bascule automatique • Vous payez ce que vous consommez • Plusieurs centrales = redondance

Le "cloud" c'est :

✅ Des serveurs physiques réels dans des datacenters ✅ Répartis géographiquement (plusieurs zones/régions) ✅ Vos données répliquées sur plusieurs serveurs ✅ Abstraction : vous ne gérez pas l'infrastructure ✅ Élasticité : s'adapte automatiquement à vos besoins

Le terme "cloud" est juste un marketing pour dire : "Serveurs distribués et gérés par quelqu'un d'autre" ! ☁️

1.1 Définition du Cloud Computing

Le **Cloud Computing** est la mise à disposition de ressources informatiques (serveurs, stockage, bases de données, réseau, logiciels) via Internet, avec un paiement à l'usage. Plutôt que de posséder et maintenir des serveurs physiques, les entreprises louent ces ressources auprès de fournisseurs cloud.

Les 3 modèles de service :

- **IaaS** (Infrastructure as a Service) : Location de serveurs virtuels (ex: Amazon EC2)
- **PaaS** (Platform as a Service) : Plateforme de développement gérée (ex: AWS Elastic Beanstalk)
- **SaaS** (Software as a Service) : Applications complètes (ex: MongoDB Atlas)

1.2 Avantages pour Notre Client Healthcare

Problèmes actuels du client :

- Difficultés de scalabilité avec l'infrastructure actuelle
- Coûts élevés de maintenance des serveurs physiques
- Risques de perte de données en cas de panne matérielle
- Complexité de gestion de la croissance des données

Bénéfices du passage au cloud :

Scalabilité Horizontale

- Ajout automatique de ressources selon la charge
- Gestion de millions de patients sans intervention manuelle
- Pas de limite de capacité physique

Réduction des Coûts

- Pas d'investissement initial en matériel (CAPEX → OPEX)
- Paiement à l'usage réel
- Pas de coûts de maintenance matérielle
- Économies d'échelle

Haute Disponibilité

- SLA (Service Level Agreement) de 99.99% de disponibilité
- Réplication automatique des données
- Reprise après sinistre intégrée
- Pas d'interruption de service

Sécurité Renforcée

- Conformité RGPD, HIPAA, HDS
- Chiffrement automatique
- Audits de sécurité réguliers
- Équipes de sécurité dédiées 24/7

Flexibilité Géographique

- Déploiement multi-régions
- Latence réduite pour les utilisateurs
- Conformité avec les réglementations locales

1.3 Cas d'Usage Spécifique Healthcare

Pour notre projet de données médicales de 54,966 patients :

- **Volume actuel :** ~8 MB de données CSV, ~500 MB après structuration MongoDB
- **Croissance estimée :** +10,000 patients/an → +100 MB/an
- **Utilisateurs :** Médecins, administratifs, analystes (estimation : 50-200 utilisateurs)
- **Criticité :** Haute (données de santé sensibles)
- **Réglementation :** RGPD obligatoire, HDS recommandé

Le cloud est la solution idéale car :

- Infrastructure évolutive sans intervention
- Conformité réglementaire garantie
- Coûts maîtrisés et prévisibles
- Sécurité de niveau entreprise

---

2. Amazon Web Services - Vue d'Ensemble

2.1 Présentation d'AWS

Amazon Web Services (AWS) est le leader mondial du cloud computing avec :

- **35% de part de marché** (devant Microsoft Azure 22%, Google Cloud 11%)
- **200+ services** disponibles

- **32 régions géographiques** dans le monde
- **102 zones de disponibilité**
- Lancé en 2006, pionnier du cloud public

Pourquoi AWS pour notre projet ?

- Maturité et fiabilité prouvées
- Large écosystème MongoDB (Atlas, DocumentDB)
- Excellente intégration Docker (ECS, Fargate)
- Conformité RGPD et certifications santé
- Support francophone disponible

2.2 Régions et Zones de Disponibilité

Région AWS : Zone géographique contenant plusieurs datacenters

Régions pertinentes pour la France :

- **eu-west-3** (Paris) - Recommandé pour RGPD
- **eu-west-1** (Irlande) - Alternative européenne
- **eu-central-1** (Francfort) - Backup possible

Zone de Disponibilité (AZ) : Datacenter isolé au sein d'une région

- Chaque région a minimum 3 AZ
- Connectées par réseau haute vitesse
- Isolation physique (inondations, incendies, etc.)

Pour notre projet :

- **Région principale :** Paris (eu-west-3)
- **Déploiement multi-AZ** pour haute disponibilité
- **Backup région secondaire :** Francfort (pour disaster recovery)

2.3 Modèle de Responsabilité Partagée

AWS et le client se partagent les responsabilités de sécurité :

Responsabilité AWS (Sécurité DU cloud) :

- Infrastructure physique (datacenters)
- Réseau et matériel
- Hyperviseur et virtualisation
- Services managés

Responsabilité Client (Sécurité DANS le cloud) :

- Données et leur chiffrement
- Gestion des identités (IAM)
- Configuration des services
- Pare-feu et règles réseau
- Sauvegardes applicatives

Pour notre projet Healthcare :

- AWS gère l'infrastructure physique et réseau
- Nous gérons : chiffrement des données patients, contrôle d'accès, sauvegardes applicatives

---

3. Création d'un Compte AWS

3.1 Procédure Pas à Pas

Étape 1 : Inscription

1. Aller sur https://aws.amazon.com/fr/
2. Cliquer sur "Créer un compte AWS"
3. Fournir :
    - Adresse email professionnelle
    - Nom du compte (ex: "Healthcare-Production")
    - Mot de passe fort (12+ caractères)

Étape 2 : Informations de Contact

- Type de compte : Professionnel (pour facturation entreprise)
- Nom de l'entreprise
- Adresse complète
- Numéro de téléphone (vérification par SMS)

Étape 3 : Informations de Paiement

- Carte bancaire requise (pas de débit immédiat)
- Autorisation de 1€ pour vérification
- Remboursée automatiquement

Étape 4 : Vérification d'Identité

- Appel téléphonique automatisé
- Code PIN à saisir sur le téléphone
- Validation en temps réel

Étape 5 : Choix du Plan de Support

- **Basic** : Gratuit (suffisant pour débuter)
- **Developer** : 29$/mois (support email)
- **Business** : 100$/mois (support 24/7, recommandé pour production)
- **Enterprise** : 15,000$/mois (pour grandes entreprises)

Pour notre projet : Plan Developer en développement, puis Business en production

Étape 6 : Activation du Compte

- Délai : quelques minutes à 24h
- Email de confirmation reçu
- Accès à la console AWS

3.2 AWS Free Tier (Offre Gratuite)

12 mois gratuits à partir de la création du compte :

Services pertinents pour notre projet :

Service	Quota Gratuit	Suffisant pour ?
EC2 (t2.micro)	750h/mois	Tests et développement
ECS (Fargate)	Limité	Déploiement conteneurs
RDS	750h/mois (db.t2.micro)	Base de test
S3	5 GB stockage	Sauvegardes
CloudWatch	10 métriques	Monitoring basique
Data Transfer	15 GB/mois	Trafic réseau

Attention :

- MongoDB Atlas n'est PAS inclus dans AWS Free Tier
- DocumentDB n'est PAS inclus (minimum ~200$/mois)
- Surveiller les coûts avec AWS Budgets

Recommandation : Utiliser Free Tier pour tester l'architecture, puis passer en production payante

3.3 Configuration Initiale Sécurisée

Après création du compte, IMMÉDIATEMENT :

1. Activer MFA (Multi-Factor Authentication) sur le compte root IAM → Dashboard → Activate MFA on root account → Utiliser Google Authenticator ou Authy

2. Créer un utilisateur IAM administrateur (ne jamais utiliser root) IAM → Users → Add User → Nom: admin-healthcare → Access: AWS Management Console + Programmatic → Permissions: AdministratorAccess → Activer MFA sur cet utilisateur aussi

3. Configurer AWS Budgets (alertes de coûts) Billing → Budgets → Create budget → Type: Cost budget → Montant: 50€/mois (ajustable) → Alertes: 50%, 80%, 100%

4. Activer CloudTrail (audit des actions) CloudTrail → Create trail → Nom: healthcare-audit-trail → Log tous les événements → Stockage S3 chiffré

5. Configurer les tags de ressources Tags standards pour toutes les ressources:

```
Project: Healthcare-MongoDB
Environment: Dev/Prod
Owner: [Votre équipe]
CostCenter: [Votre département]
```

4. Tarification AWS

4.1 Modèle de Tarification

Principe fondamental : Pay-as-you-go (Paiement à l'usage)

3 modèles de tarification :

1. On-Demand (À la demande)

- Paiement à l'heure ou à la seconde
- Aucun engagement
- Flexibilité maximale
- **Prix le plus élevé**
- Idéal pour : Développement, tests, charges variables

2. Reserved Instances (Instances Réservées)

- Engagement 1 ou 3 ans
- Réduction de 30% à 75%
- Paiement : tout d'avance, partiel, ou mensuel
- Idéal pour : Production stable, charges prévisibles

3. Spot Instances

- Jusqu'à 90% de réduction
- AWS peut reprendre l'instance (préavis 2 min)
- Idéal pour : Batch processing, calculs non-critiques
- **Non recommandé pour bases de données**

4.2 Calculateur de Coûts AWS

AWS Pricing Calculator : https://calculator.aws/

Exemple de calcul pour notre projet Healthcare :

Scénario : 54,966 patients, croissance 10K/an

Architecture proposée :

- MongoDB Atlas M10 (géré par MongoDB, hébergé sur AWS)
- ECS Fargate pour l'application Python
- S3 pour sauvegardes
- CloudWatch pour monitoring

Calcul détaillé : COÛTS MENSUELS ESTIMÉS (Région Paris - eu-west-3)

```
MongoDB Atlas M10 (Recommandé)
    RAM: 2 GB
    Storage: 10 GB
    Backup: Inclus
```

```
    Prix: 57€/mois

ECS Fargate (Application de migration)
    0.25 vCPU, 0.5 GB RAM
    Exécution: 1h/jour (migrations quotidiennes)
    Prix: ~3€/mois

Amazon S3 (Sauvegardes)
    50 GB de stockage
    Prix: 1,15€/mois

CloudWatch (Monitoring)
    10 métriques custom
    Logs: 5 GB/mois
    Prix: 5€/mois

Data Transfer
    10 GB sortant/mois
    Prix: 0,90€/mois

AWS Backup (Optionnel)
    50 GB sauvegardés
    Prix: 2,50€/mois
```

═══════════════════════════════════════ TOTAL MENSUEL: ~70€/mois TOTAL ANNUEL: ~840€/an ═══════════════════════════════════════

Avec engagement 1 an (Reserved): TOTAL MENSUEL: ~50€/mois (-30%) TOTAL ANNUEL: ~600€/an

Comparaison avec infrastructure On-Premise :

Poste	On-Premise	AWS Cloud
Serveur physique	3,000€ initial	0€
Maintenance annuelle	1,500€	Inclus
Électricité	600€/an	Inclus
Personnel IT (20% temps)	10,000€/an	Réduit de 50%
TOTAL An 1	**15,100€**	**5,840€**
TOTAL An 3	**20,100€**	**6,840€**

ROI (Return on Investment) : Le cloud est rentable dès la première année !

4.3 Optimisation des Coûts

Recommandations pour réduire les coûts :

1. Utiliser le Free Tier au maximum

- 12 mois gratuits pour tests
- Économie : ~200€

2. Reserved Instances pour production

- Engagement 1 an
- Économie : 30% (~250€/an)

3. Automatiser l'arrêt des ressources non-utilisées

- Arrêt automatique dev/test le soir et week-end
- Économie : 60% sur ces environnements

4. Utiliser S3 Intelligent-Tiering

- Déplace automatiquement les données anciennes vers stockage moins cher
- Économie : 40% sur stockage

5. Surveiller avec AWS Cost Explorer

- Analyse des coûts par service
- Identification des dépenses inutiles
- Alertes de dépassement

6. Tags de ressources rigoureux

- Permet d'attribuer les coûts par projet
- Identifie les ressources orphelines

Budget recommandé avec marge :

- Développement : 30€/mois
- Production : 70€/mois
- **Total : 100€/mois (1,200€/an)**

---

5. Solutions MongoDB sur AWS

5.1 Comparaison des Options

AWS propose **3 solutions principales** pour héberger MongoDB :

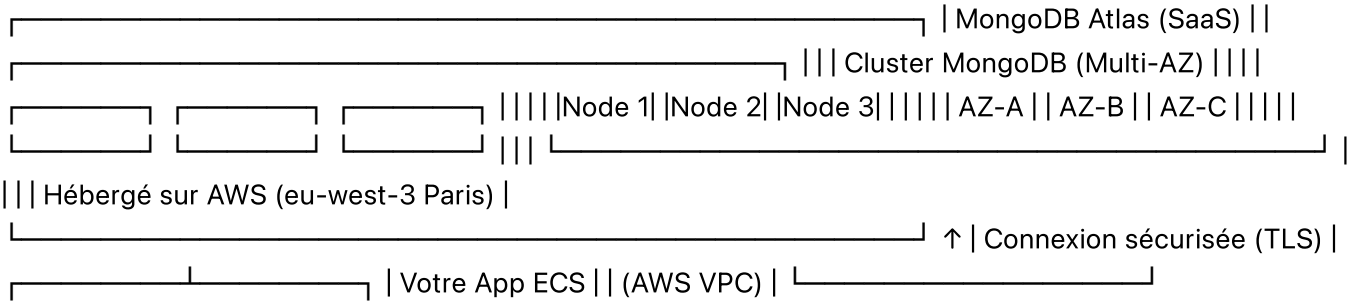
Critère	MongoDB Atlas	Amazon DocumentDB	EC2 + MongoDB
Gestion	Entièrement géré	Géré par AWS	Autogéré
Compatibilité MongoDB	100%	~95% (API compatible)	100%
Facilité	⭐⭐⭐⭐⭐	⭐⭐⭐⭐	⭐⭐
Scalabilité	Automatique	Automatique	Manuelle
Backup	Automatique	Automatique	À configurer

Critère	MongoDB Atlas	Amazon DocumentDB	EC2 + MongoDB
Prix	57€/mois (M10)	200€/mois (min)	50€/mois + gestion
Sécurité	Excellente	Excellente	À configurer
Support	MongoDB Inc.	AWS	Communauté
Recommandation	⭐⭐⭐⭐⭐	⭐⭐⭐	⭐⭐

5.2 Option 1 : MongoDB Atlas (RECOMMANDÉ)

Description : MongoDB Atlas est la solution Database-as-a-Service (DBaaS) officielle de MongoDB Inc., hébergée sur AWS.

Architecture :



Avantages : ✅ **Zéro gestion d'infrastructure** ✅ **Backup automatique** (snapshots quotidiens, PITR) ✅ **Monitoring intégré** (dashboards temps réel) ✅ **Scaling automatique** (vertical et horizontal) ✅ **Sécurité enterprise** (chiffrement, audit, compliance) ✅ **100% compatible MongoDB** (toutes les features) ✅ **Support officiel MongoDB** ✅ **Interface graphique intuitive**

Inconvénients : ❌ Service externe (pas directement dans votre VPC AWS) ❌ Latence légèrement supérieure vs DocumentDB ❌ Coût un peu plus élevé pour petites instances

Tarification MongoDB Atlas sur AWS :

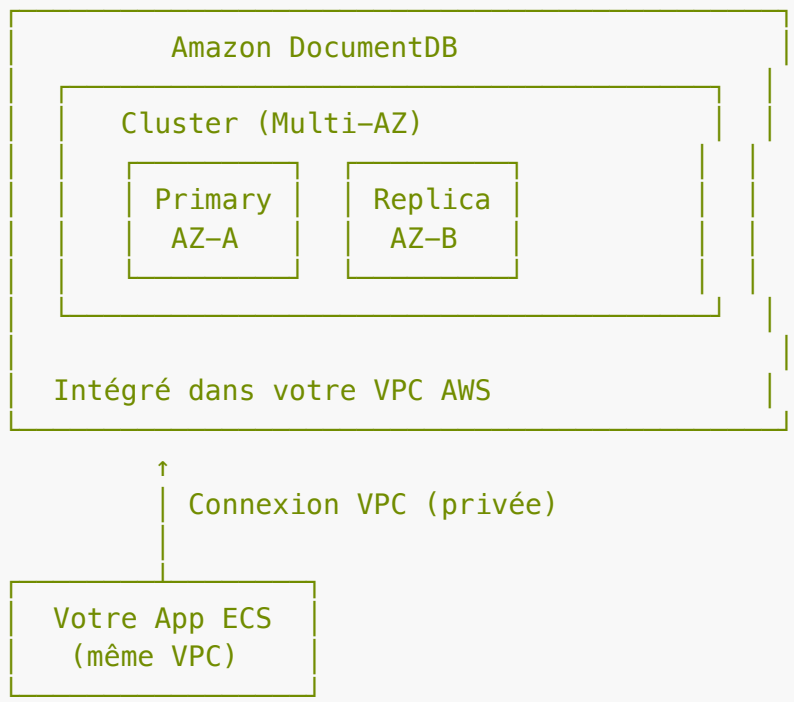
Cluster	vCPU	RAM	Storage	Prix/mois
M0 (Free)	Partagé	512 MB	512 MB	Gratuit
M2	Partagé	2 GB	2 GB	9€
M10	2	2 GB	10 GB	**57€** ⭐
M20	2	4 GB	20 GB	115€
M30	2	8 GB	40 GB	230€

Pour notre projet : M10 est idéal

- Capacité : 100K+ patients
- Performance : 3000+ IOPS
- Backup inclus
- Multi-AZ inclus

Configuration recommandée :

```
Cluster Name: healthcare-prod
Cloud Provider: AWS
Region: eu-west-3 (Paris)
Cluster Tier: M10
MongoDB Version: 7.0
Backup: Enabled (snapshots quotidiens)
Network: IP Whitelist + VPC Peering
Encryption: At rest + In transit

Connexion depuis ECS :

MONGODB_URI = "mongodb+srv://user:pass@healthcare-
prod.xxxxx.mongodb.net/healthcare_db?retryWrites=true&w=majority"

5.3 Option 2 : Amazon DocumentDB

Description : Amazon DocumentDB est un service de base de données
documentaire géré par AWS, compatible avec l'API MongoDB.

Architecture :
```



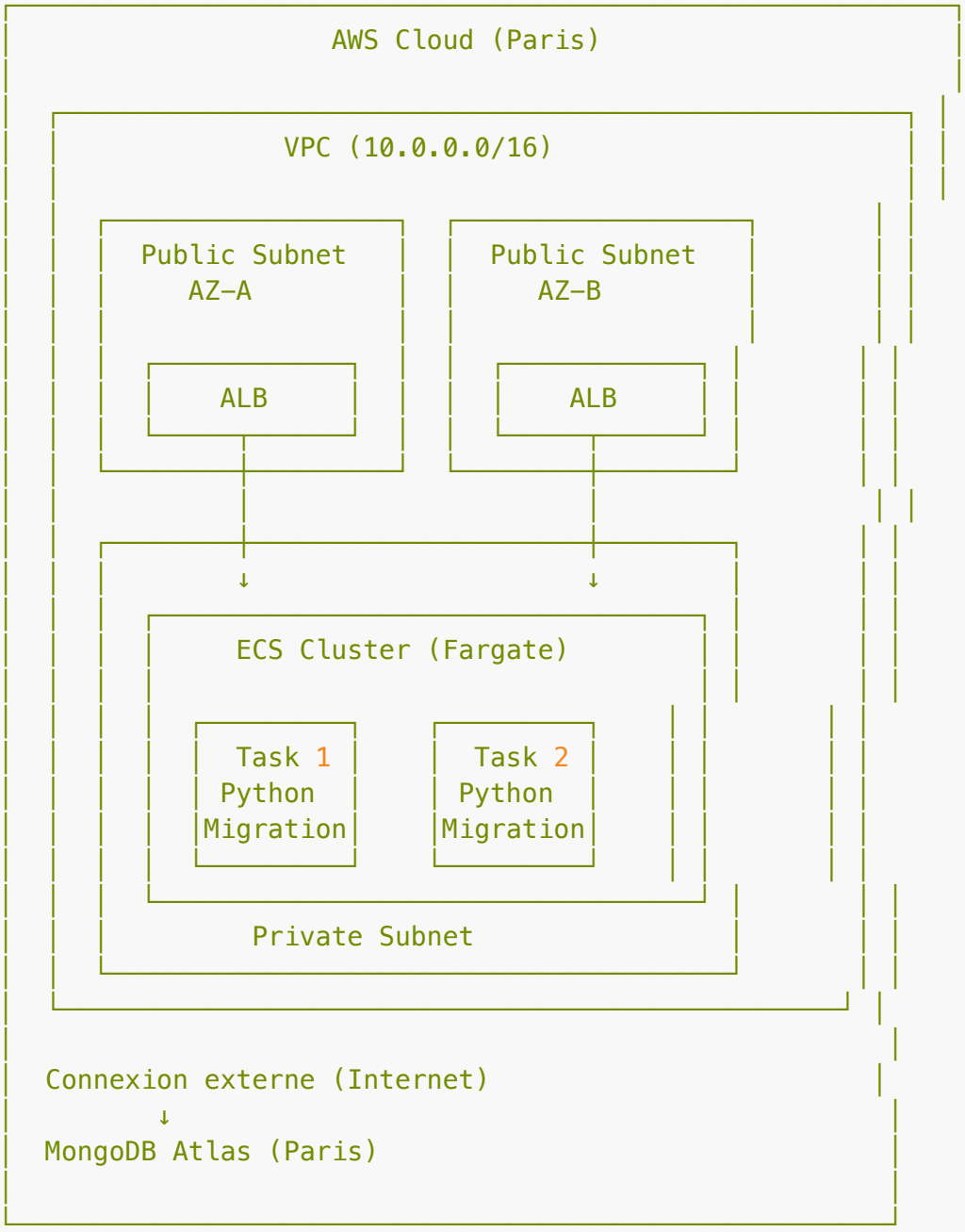

```
Avantages : ✅ Intégration AWS native (VPC, IAM, CloudWatch) ✅ Latence
minimale (même VPC que votre app) ✅ Scaling automatique du stockage ✅
Backup automatique (jusqu'à 35 jours) ✅ Haute disponibilité (Multi-AZ
natif) ✅ Sécurité AWS (KMS, VPC, Security Groups)

Inconvénients : ❌ Compatibilité MongoDB ~95% (pas 100%) ❌ Certaines
features manquantes (aggregation pipeline limitée, pas de transactions
multi-documents avant v4.0) ❌ Coût élevé (minimum 200€/mois) ❌ Vendor
lock-in AWS (difficile de migrer ailleurs)
```

```
Tarification DocumentDB :
Instance    vCPU    RAM     Prix/mois (On-Demand)
db.t3.medium    2   4 GB    110€
db.r5.large     2   16 GB   330€
db.r5.xlarge    4   32 GB   660€

Stockage : 0,10€/GB/mois (min 10 GB)
I/O : 0,20€ par million de requêtes

Coût minimum réaliste : ~200€/mois

Pour notre projet : Trop cher pour le volume actuel (54K patients)

Cas d'usage où DocumentDB est pertinent :

    Très gros volumes (millions de documents)
    Besoin d'intégration AWS poussée
    Budget confortable
    Pas besoin de 100% compatibilité MongoDB

5.4 Option 3 : EC2 + MongoDB Auto-Géré

Description : Installer et gérer MongoDB vous-même sur des instances EC2.

Architecture :

┌─────────────────────────────────────────┐
│         Amazon EC2                      │
│  ┌──────────────────────────────────┐   │
│  │   Instance t3.medium             │   │
│  │   ┌──────────────────────┐       │   │
│  │   │  Ubuntu 22.04        │       │   │
│  │   │  MongoDB 7.0         │       │   │
│  │   │  50 GB EBS Volume    │       │   │
│  │   └──────────────────────┘       │   │
│  └──────────────────────────────────┘   │
│                                         │
│  + Configuration manuelle de tout       │
└─────────────────────────────────────────┘



Avantages : ✅ Contrôle total (configuration, version, tuning) ✅ 100%
compatible MongoDB ✅ Coût potentiellement plus bas (si bien optimisé) ✅
Pas de vendor lock-in

Inconvénients : ❌ Gestion manuelle complète (installation, mises à jour,
patches) ❌ Backup à configurer (scripts, S3, automatisation) ❌
Monitoring à mettre en place (CloudWatch, Prometheus) ❌ Haute
disponibilité complexe (replica set multi-AZ) ❌ Sécurité à durcir
(firewall, chiffrement, audit) ❌ Temps de gestion important (estimé 10-
20h/mois)

Coût estimé :
```

```
    EC2 t3.medium : 40€/mois
    EBS 50 GB : 5€/mois
    Backup S3 : 2€/mois
    Total : 47€/mois

Mais : Coût caché du temps de gestion !

    15h/mois × 50€/h = 750€/mois en temps ingénieur
    Coût réel : ~800€/mois

Pour notre projet : Non recommandé sauf contrainte budgétaire extrême
5.5 Recommandation Finale

Pour le projet Healthcare :

🥇 MongoDB Atlas M10 - RECOMMANDÉ

    Meilleur rapport qualité/prix
    Zéro gestion
    Support officiel
    Évolutif

Justification :

    Simplicité : Focus sur l'application, pas l'infrastructure
    Fiabilité : SLA 99.995%, backup automatique
    Sécurité : Conformité RGPD/HIPAA out-of-the-box
    Coût : 57€/mois vs 200€ DocumentDB vs 800€ EC2 (avec gestion)
    Évolutivité : Scale up/down en 1 clic

Architecture recommandée :

Internet
   ↓
AWS ALB (Load Balancer)
   ↓
ECS Fargate (Application Python)
   ↓ (TLS 1.3)
MongoDB Atlas M10 (Paris)
   ↓ (Backup automatique)
AWS S3 (Sauvegardes supplémentaires)

6. Amazon ECS - Déploiement de Conteneurs
6.1 Présentation d'Amazon ECS

Amazon Elastic Container Service (ECS) est le service d'orchestration de
conteneurs Docker d'AWS.

Deux modes de lancement :

1. EC2 Launch Type

    Vous gérez les instances EC2
```

```
    Plus de contrôle
    Moins cher pour gros volumes
    Plus complexe

2. Fargate Launch Type ⭐ RECOMMANDÉ

    Serverless (pas de serveurs à gérer)
    AWS gère l'infrastructure
    Paiement à l'usage (par vCPU/RAM/seconde)
    Simplicité maximale

Pour notre projet : Fargate
```

6.2 Architecture ECS pour Notre Projet



6.3 Composants ECS

1. Cluster ECS

```
    Groupe logique de tâches et services
    Peut contenir plusieurs services
    Pour notre projet : healthcare-cluster

2. Task Definition

    Blueprint de votre conteneur
    Spécifie : image Docker, CPU, RAM, variables d'environnement
    Versionné (task-definition:1, :2, etc.)

3. Service

    Maintient un nombre défini de tâches en cours d'exécution
    Auto-healing : relance les tâches qui crashent
    Intégration avec ALB pour load balancing

4. Task

    Instance d'exécution d'une Task Definition
    1 ou plusieurs conteneurs
    Éphémère (peut être arrêtée/relancée)

6.4 Déploiement de Notre Application sur ECS

Étape 1 : Créer un Repository ECR (Elastic Container Registry)

ECR est le registre Docker privé d'AWS (équivalent Docker Hub).

# Créer le repository
aws ecr create-repository \
  --repository-name healthcare-migration \
  --region eu-west-3

# Résultat : URI du repository
# 123456789012.dkr.ecr.eu-west-3.amazonaws.com/healthcare-migration

Étape 2 : Pousser l'Image Docker vers ECR

# Authentification Docker vers ECR
aws ecr get-login-password --region eu-west-3 | \
  docker login --username AWS --password-stdin \
  123456789012.dkr.ecr.eu-west-3.amazonaws.com

# Tag de l'image
docker tag healthcare-docker-migration_app:latest \
  123456789012.dkr.ecr.eu-west-3.amazonaws.com/healthcare-migration:latest

# Push vers ECR
docker push 123456789012.dkr.ecr.eu-west-3.amazonaws.com/healthcare-
migration:latest


Étape 3 : Créer une Task Definition
```

Fichier task-definition.json :

```
{
  "family": "healthcare-migration-task",
  "networkMode": "awsvpc",
  "requiresCompatibilities": ["FARGATE"],
  "cpu": "256",
  "memory": "512",
  "executionRoleArn":
"arn:aws:iam::123456789012:role/ecsTaskExecutionRole",
  "containerDefinitions": [
    {
      "name": "migration-container",
      "image": "123456789012.dkr.ecr.eu-west-3.amazonaws.com/healthcare-
migration:latest",
      "essential": true,
      "environment": [
        {
          "name": "MONGODB_HOST",
          "value": "healthcare-prod.xxxxx.mongodb.net"
        },
        {
          "name": "MONGODB_PORT",
          "value": "27017"
        },
        {
          "name": "MONGODB_DATABASE",
          "value": "healthcare_db"
        }
      ],
      "secrets": [
        {
          "name": "MONGODB_USERNAME",
          "valueFrom": "arn:aws:secretsmanager:eu-west-
3:123456789012:secret:mongodb-username"
        },
        {
          "name": "MONGODB_PASSWORD",
          "valueFrom": "arn:aws:secretsmanager:eu-west-
3:123456789012:secret:mongodb-password"
        }
      ],
      "logConfiguration": {
        "logDriver": "awslogs",
        "options": {
          "awslogs-group": "/ecs/healthcare-migration",
          "awslogs-region": "eu-west-3",
          "awslogs-stream-prefix": "ecs"
        }
      }
    }
  ]
}
```

```
Points clés :

    cpu: "256" = 0.25 vCPU (suffisant pour migration)
    memory: "512" = 512 MB RAM
    secrets : Utilise AWS Secrets Manager (sécurisé)
    logConfiguration : Logs envoyés vers CloudWatch

Étape 4 : Créer le Cluster ECS

aws ecs create-cluster \
  --cluster-name healthcare-cluster \
  --region eu-west-3

Étape 5 : Créer un Service ECS

aws ecs create-service \
  --cluster healthcare-cluster \
  --service-name healthcare-migration-service \
  --task-definition healthcare-migration-task:1 \
  --desired-count 1 \
  --launch-type FARGATE \
  --network-configuration "awsvpcConfiguration={subnets=[subnet-
12345,subnet-67890],securityGroups=[sg-12345],assignPublicIp=ENABLED}" \
  --region eu-west-3

Étape 6 : Planifier l'Exécution (Migration Quotidienne)

Utiliser Amazon EventBridge (anciennement CloudWatch Events) :

{
  "ScheduleExpression": "cron(0 2 * * ? *)",
  "Description": "Migration quotidienne à 2h du matin",
  "State": "ENABLED",
  "Targets": [
    {
      "Arn": "arn:aws:ecs:eu-west-3:123456789012:cluster/healthcare-
cluster",
      "RoleArn": "arn:aws:iam::123456789012:role/ecsEventsRole",
      "EcsParameters": {
        "TaskDefinitionArn": "arn:aws:ecs:eu-west-3:123456789012:task-
definition/healthcare-migration-task:1",
        "TaskCount": 1,
        "LaunchType": "FARGATE",
        "NetworkConfiguration": {
          "awsvpcConfiguration": {
            "Subnets": ["subnet-12345", "subnet-67890"],
            "SecurityGroups": ["sg-12345"],
            "AssignPublicIp": "ENABLED"
          }
        }
      }
    }
  ]
}
```

```
Résultat : La migration s'exécute automatiquement tous les jours à 2h du
matin !
6.5 Coûts ECS Fargate

Tarification Fargate (Paris - eu-west-3) :

    vCPU : 0,04656€ par vCPU-heure
    RAM : 0,00511€ par GB-heure

Calcul pour notre migration :

Configuration : 0.25 vCPU, 0.5 GB RAM
Durée migration : 3 minutes = 0.05 heure
Fréquence : 1x par jour = 30x par mois

Coût vCPU : 0.25 × 0.04656€ × 0.05h × 30 = 0,017€/mois
Coût RAM  : 0.5 × 0.00511€ × 0.05h × 30 = 0,004€/mois

TOTAL : ~0,02€/mois (négligeable !)

Si migration en continu (24/7) :

Coût vCPU : 0.25 × 0.04656€ × 730h = 8,50€/mois
Coût RAM  : 0.5 × 0.00511€ × 730h = 1,87€/mois

TOTAL : ~10,37€/mois

Avantage Fargate : Vous payez uniquement quand la tâche s'exécute !
6.6 Monitoring et Logs

CloudWatch Logs :

    Tous les logs de l'application sont centralisés
    Recherche et filtrage en temps réel
    Rétention configurable (1 jour à ∞)
    Alertes sur patterns d'erreurs

CloudWatch Metrics :

    CPU Utilization
    Memory Utilization
    Task Count
    Métriques custom (nombre de documents migrés, erreurs, etc.)

Exemple de dashboard CloudWatch :

┌─────────────────────────────────────────────┐
│  Healthcare Migration Dashboard             │
├─────────────────────────────────────────────┤
│  CPU Usage        │  Memory Usage           │
│  [████░░░░] 40%   │  [███████░] 70%         │
├─────────────────────────────────────────────┤
│  Documents Migrés Aujourd'hui : 54,966      │
```

```
│  Erreurs : 0                                    │
│  Durée : 2m 15s                                 │
├─────────────────────────────────────────────────┤
│  Historique 7 jours :                           │
│  [Graphique linéaire]                           │
└─────────────────────────────────────────────────┘
```

7. Sauvegardes et Surveillance

7.1 Stratégie de Sauvegarde

Principe 3-2-1 :

- 3 copies des données
- 2 supports différents
- 1 copie hors site

Pour notre projet Healthcare :

```
┌─────────────────────────────────────────────────┐
│  STRATÉGIE DE SAUVEGARDE COMPLÈTE               │
└─────────────────────────────────────────────────┘

Niveau 1 : MongoDB Atlas (Automatique)
├─ Snapshots quotidiens (rétention 7 jours)
├─ Point-in-Time Recovery (PITR) jusqu'à 24h
└─ Réplication Multi-AZ (3 copies en temps réel)

Niveau 2 : Export S3 (Hebdomadaire)
├─ Export complet de la base en JSON
├─ Stockage S3 Standard (Paris)
├─ Versioning activé
└─ Lifecycle policy : S3 Glacier après 30 jours

Niveau 3 : Backup Cross-Region (Mensuel)
├─ Copie vers région Francfort (eu-central-1)
├─ Stockage S3 Glacier Deep Archive
└─ Coût minimal, rétention longue durée

Niveau 4 : Export Local (Trimestriel)
├─ Export complet hors cloud
├─ Stockage physique sécurisé
└─ Conformité réglementaire
```

7.2 AWS Backup

AWS Backup est le service centralisé de sauvegarde AWS.

Configuration recommandée :

```
{
  "BackupPlan": {
    "BackupPlanName": "Healthcare-Daily-Backup",
    "Rules": [
      {
```

```
        "RuleName": "DailyBackup",
        "TargetBackupVault": "Healthcare-Vault",
        "ScheduleExpression": "cron(0 3 * * ? *)",
        "StartWindowMinutes": 60,
        "CompletionWindowMinutes": 120,
        "Lifecycle": {
          "DeleteAfterDays": 35,
          "MoveToColdStorageAfterDays": 7
        }
      },
      {
        "RuleName": "WeeklyLongTerm",
        "TargetBackupVault": "Healthcare-Vault-LongTerm",
        "ScheduleExpression": "cron(0 3 ? * SUN *)",
        "Lifecycle": {
          "DeleteAfterDays": 365
        }
      }
    ]
  }
}

Coûts AWS Backup :

    Stockage chaud : 0,05€/GB/mois
    Stockage froid : 0,01€/GB/mois
    Restauration : 0,02€/GB

Pour 50 GB de données :

Backup quotidien (7 jours chaud) : 50 GB × 0,05€ × 7 = 17,50€/mois
Backup hebdomadaire (froid, 1 an) : 50 GB × 0,01€ × 52 = 26€/mois

TOTAL : ~44€/mois pour backups complets

7.3 Script d'Export S3 Automatisé

Lambda Function pour export hebdomadaire :

import boto3
import pymongo
import json
from datetime import datetime

def lambda_handler(event, context):
    """
    Export hebdomadaire de MongoDB vers S3
    """

    # Connexion MongoDB Atlas
    client = pymongo.MongoClient(os.environ['MONGODB_URI'])
    db = client['healthcare_db']
    collection = db['patients']
```

```
    # Export tous les documents
    documents = list(collection.find({}))

    # Conversion en JSON
    backup_data = {
        'export_date': datetime.now().isoformat(),
        'document_count': len(documents),
        'documents': documents
    }

    # Upload vers S3
    s3 = boto3.client('s3')
    filename = f"backup-{datetime.now().strftime('%Y%m%d')}.json"

    s3.put_object(
        Bucket='healthcare-backups',
        Key=f'mongodb-exports/{filename}',
        Body=json.dumps(backup_data, default=str),
        ServerSideEncryption='AES256'
    )

    return {
        'statusCode': 200,
        'body': f'Backup réussi : {len(documents)} documents exportés'
    }
    #Planification avec EventBridge :

    {
    "ScheduleExpression": "cron(0 4 ? * SUN *)",
    "Description": "Export S3 hebdomadaire le dimanche à 4h",
    "State": "ENABLED",
    "Targets": [
        {
        "Arn": "arn:aws:lambda:eu-west-3:123456789012:function:mongodb-s3-
backup",
        "Id": "1"
        }
    ]
    }


    7.4 Surveillance avec CloudWatch

Métriques Essentielles à Surveiller :

1. Métriques MongoDB (via Atlas) :

    Connexions actives
    Opérations par seconde (ops/sec)
    Latence des requêtes
    Utilisation CPU/RAM
    Espace disque utilisé

2. Métriques ECS :
```

```
    Task Running Count
    CPU/Memory Utilization
    Task Failures
    Service Health

3. Métriques Custom (Application) :

    Nombre de documents migrés
    Durée de migration
    Erreurs de migration
    Intégrité des données
```

Dashboard CloudWatch Complet :

```
┌──────────────────────────────────────────────────────────┐
│  Healthcare MongoDB - Production Dashboard                │
├──────────────────────────────────────────────────────────┤
│                                                          │
│  ┌─────────────────────┐  ┌─────────────────────┐        │
│  │ MongoDB Ops/sec     │  │ ECS Task Status     │        │
│  │ [Graph: 1.2K ops]   │  │ Running: 1          │        │
│  └─────────────────────┘  │ Stopped: 0          │        │
│                           └─────────────────────┘        │
│  ┌─────────────────────┐  ┌─────────────────────┐        │
│  │ Database Size       │  │ Backup Status       │        │
│  │ [Graph: 500 MB]     │  │ Last: 2h ago ✓      │        │
│  └─────────────────────┘  │ Next: in 22h        │        │
│                           └─────────────────────┘        │
│  ┌──────────────────────────────────────────────┐        │
│  │ Recent Errors (Last 24h)                     │        │
│  │ [Table: 0 errors]                            │        │
│  └──────────────────────────────────────────────┘        │
│                                                          │
│  ┌──────────────────────────────────────────────┐        │
│  │ Cost Estimate (This Month)                   │        │
│  │ MongoDB Atlas: 57€                           │        │
│  │ ECS Fargate: 0,02€                           │        │
│  │ S3 Backup: 2€                                │        │
│  │ CloudWatch: 5€                               │        │
│  │ TOTAL: 64,02€                                │        │
│  └──────────────────────────────────────────────┘        │
└──────────────────────────────────────────────────────────┘
```

7.5 Alertes CloudWatch

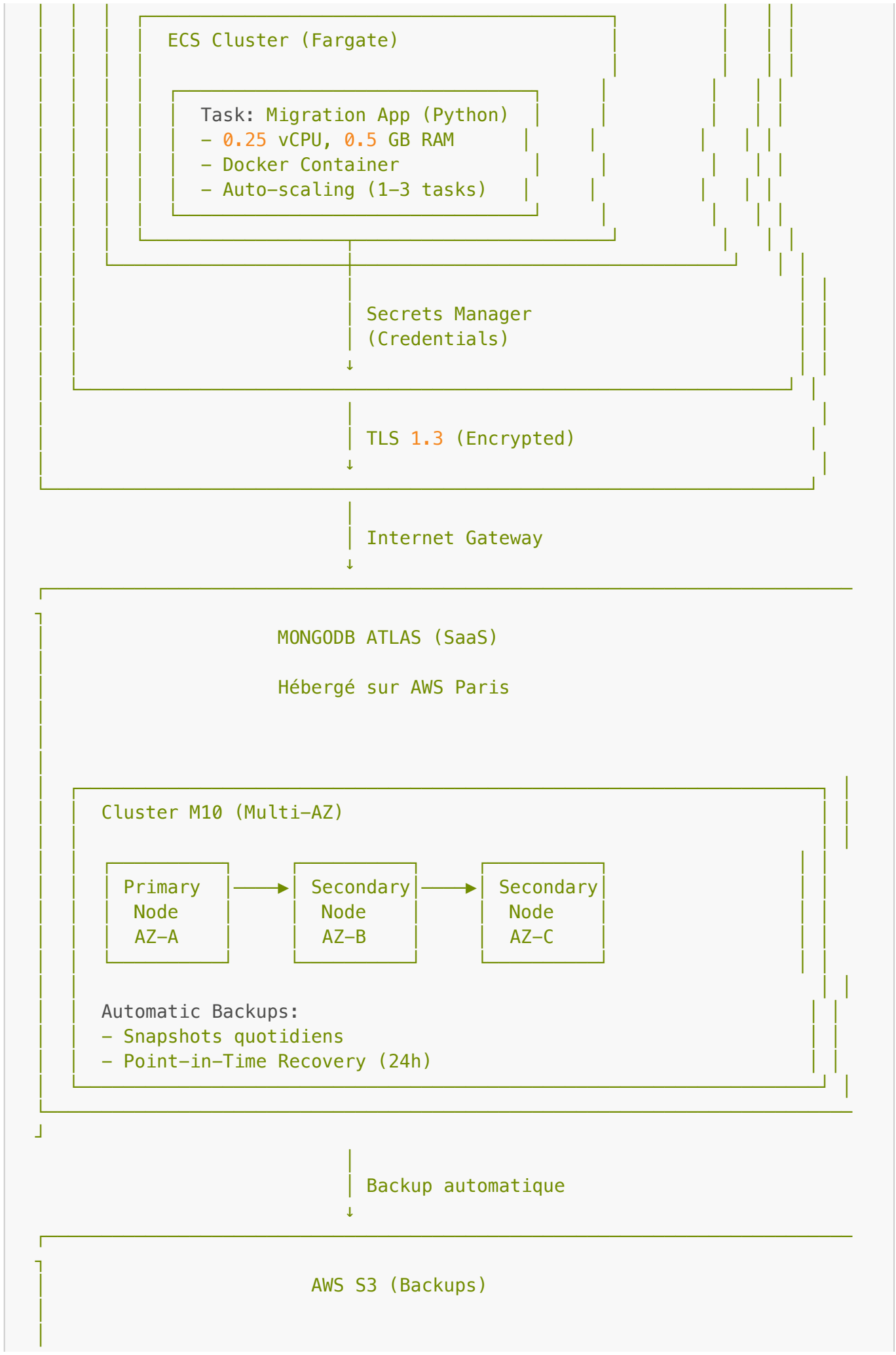
Alertes Critiques à Configurer :

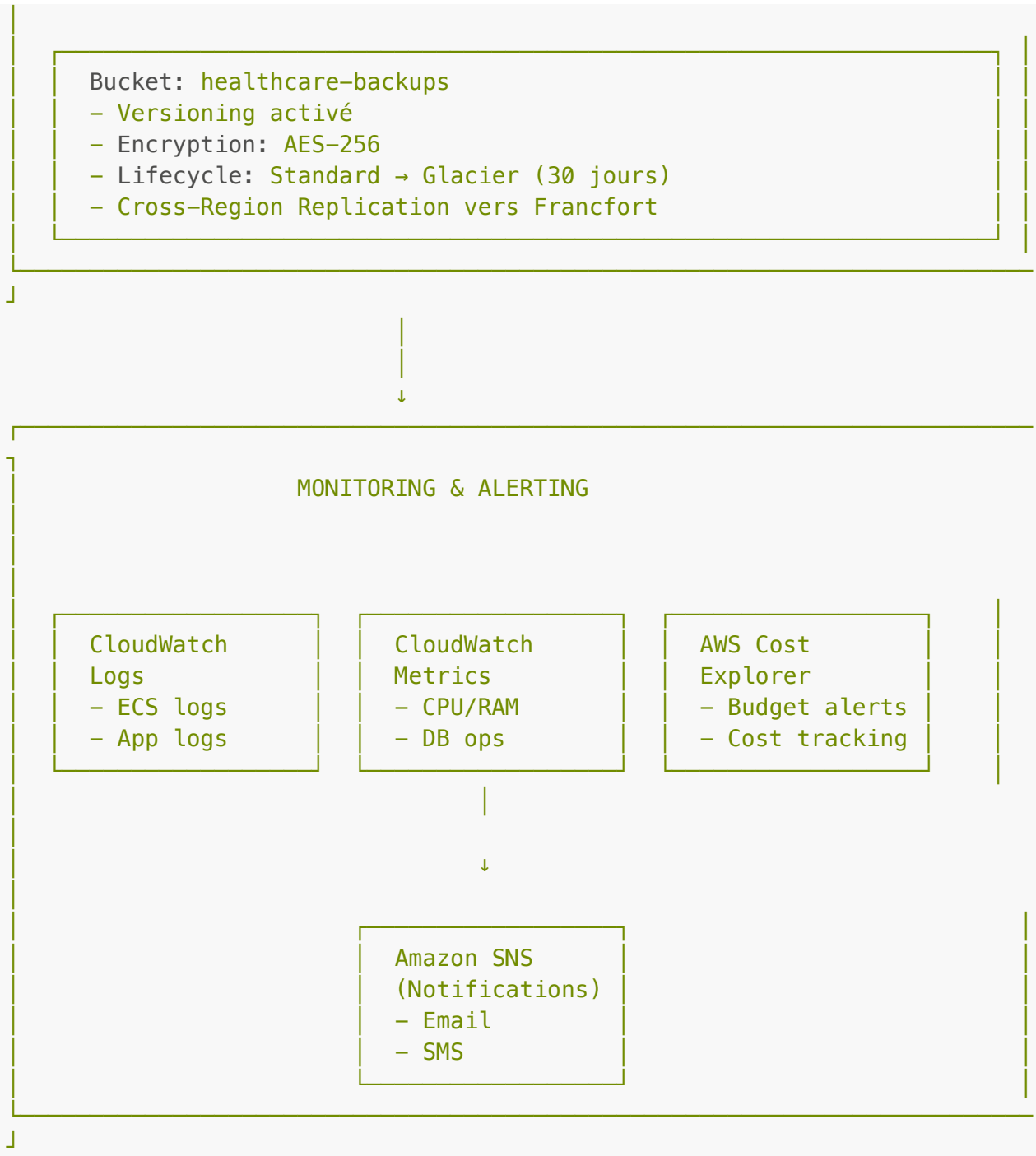
1. Alerte : Base de Données Inaccessible

```
{
  "AlarmName": "MongoDB-Connection-Failed",
  "MetricName": "ConnectionErrors",
  "Threshold": 5,
```

```
  "EvaluationPeriods": 2,
  "ComparisonOperator": "GreaterThanThreshold",
  "AlarmActions": ["arn:aws:sns:eu-west-3:123456789012:healthcare-
alerts"],
  "AlarmDescription": "Plus de 5 erreurs de connexion MongoDB en 10
minutes"
}

2. Alerte : Migration Échouée

{
  "AlarmName": "Migration-Task-Failed",
  "MetricName": "TasksFailed",
  "Threshold": 1,
  "EvaluationPeriods": 1,
  "ComparisonOperator": "GreaterThanOrEqualToThreshold",
  "AlarmActions": ["arn:aws:sns:eu-west-3:123456789012:healthcare-alerts"]
}

3. Alerte : Espace Disque > 80%

{
  "AlarmName": "MongoDB-Disk-Usage-High",
  "MetricName": "DiskUsagePercent",
  "Threshold": 80,
  "EvaluationPeriods": 2,
  "ComparisonOperator": "GreaterThanThreshold",
  "AlarmActions": ["arn:aws:sns:eu-west-3:123456789012:healthcare-alerts"]
}

4. Alerte : Coûts Dépassés

{
  "AlarmName": "AWS-Cost-Budget-Exceeded",
  "MetricName": "EstimatedCharges",
  "Threshold": 100,
  "EvaluationPeriods": 1,
  "ComparisonOperator": "GreaterThanThreshold",
  "AlarmActions": ["arn:aws:sns:eu-west-3:123456789012:billing-alerts"]
}

Notification : Toutes les alertes envoient des emails/SMS via Amazon SNS
7.6 Tests de Restauration

IMPORTANT : Avoir des backups ne suffit pas, il faut les TESTER !

Procédure de Test Trimestriel :

# 1. Créer un cluster de test
# (MongoDB Atlas ou DocumentDB)

# 2. Restaurer depuis backup
aws backup start-restore-job \
  --recovery-point-arn arn:aws:backup:eu-west-3:123456789012:recovery-
```

```
point:xxx \
  --metadata file://restore-metadata.json

# 3. Vérifier l'intégrité
python3 verify_backup.py \
  --source production-cluster \
  --target test-cluster \
  --sample-size 1000

# 4. Mesurer le temps de restauration
# Objectif : RTO < 4 heures

# 5. Documenter les résultats
# Rapport de test de restauration

Objectifs de Récupération :

    RTO (Recovery Time Objective) : 4 heures maximum
    RPO (Recovery Point Objective) : 1 heure maximum (grâce à PITR)

8. Architecture Complète Proposée
8.1 Schéma d'Architecture Global

┌─────────────────────────────────────────────────────────────────────────────┐
│                                                                             │
│                          INTERNET / UTILISATEURS                            │
│                                                                             │
└──────────────────────────────────┬──────────────────────────────────────────┘
                                   │
                                   │ HTTPS (TLS 1.3)
                                   ↓
┌─────────────────────────────────────────────────────────────────────────────┐
│                                                                             │
│                        AWS CLOUD (eu-west-3 Paris)                          │
│                                                                             │
│                                                                             │
│   ┌─────────────────────────────────────────────────────────────────────┐   │
│   │                        VPC (10.0.0.0/16)                            │   │
│   │                                                                     │   │
│   │   ┌─────────────────────────────────────────────────────────┐       │   │
│   │   │  Public Subnet (10.0.1.0/24)                            │       │   │
│   │   │                                                         │       │   │
│   │   │   ┌─────────────────────────────────────┐               │       │   │
│   │   │   │  Application Load Balancer (ALB)    │               │       │   │
│   │   │   │  - SSL/TLS Termination              │               │       │   │
│   │   │   │  - Health Checks                    │               │       │   │
│   │   │   └─────────────────┬───────────────────┘               │       │   │
│   │   └─────────────────────┼───────────────────────────────────┘       │   │
│   │                         │                                           │   │
│   │   ┌─────────────────────┼───────────────────────────────────┐       │   │
│   │   │  Private Subnet (10.0.2.0/24)                           │       │   │
│   │   │                     ↓                                   │       │   │
```


ECS Cluster (Fargate)
Task: Migration App (Python)
- 0.25 vCPU, 0.5 GB RAM
- Docker Container
- Auto-scaling (1-3 tasks)
Secrets Manager
(Credentials)
TLS 1.3 (Encrypted)
Internet Gateway
MONGODB ATLAS (SaaS)
Hébergé sur AWS Paris
Cluster M10 (Multi-AZ)
Primary Node AZ-A
Secondary Node AZ-B
Secondary Node AZ-C
Automatic Backups:
- Snapshots quotidiens
- Point-in-Time Recovery (24h)
Backup automatique
AWS S3 (Backups)



8.2 Flux de Données

Flux de Migration Quotidienne :

1. EventBridge (Trigger 2h du matin)
   ↓
2. Lance ECS Task (Fargate)
   ↓
3. Task récupère secrets (Secrets Manager)
   ↓
4. Task télécharge CSV depuis S3
   ↓
5. Task nettoie les données (pandas)
   ↓
6. Task se connecte à MongoDB Atlas
   ↓

```
7. Task migre les données (batch de 1000)
   ↓
8. Task vérifie l'intégrité
   ↓
9. Task génère rapport (S3 + CloudWatch)
   ↓
10. Task se termine (logs dans CloudWatch)
    ↓
11. MongoDB Atlas backup automatique (snapshot)
    ↓
12. Notification SNS si erreur

Flux de Lecture (Utilisateurs) :

1. Utilisateur → HTTPS → ALB
   ↓
2. ALB → ECS Task (API)
   ↓
3. API authentifie (JWT)
   ↓
4. API query MongoDB Atlas
   ↓
5. MongoDB retourne données
   ↓
6. API formate réponse
   ↓
7. ALB → Utilisateur

8.3 Composants de Sécurité

Couches de Sécurité :

1. Réseau

    VPC isolé (10.0.0.0/16)
    Security Groups restrictifs
    Network ACLs
    Pas d'accès Internet direct pour ECS (NAT Gateway)

2. Identité et Accès

    IAM Roles (principe du moindre privilège)
    MFA obligatoire
    Rotation des credentials (Secrets Manager)
    Pas de clés d'accès en dur

3. Données

    Chiffrement en transit (TLS 1.3)
    Chiffrement au repos (AES-256)
    MongoDB Field-Level Encryption (données sensibles)
    Backup chiffrés

4. Application
```

```
    Conteneurs scannés (ECR Image Scanning)
    Pas de root dans les conteneurs
    Secrets via Secrets Manager
    Validation des entrées

5. Monitoring

    CloudTrail (audit de toutes les actions AWS)
    VPC Flow Logs (trafic réseau)
    GuardDuty (détection de menaces)
    Config (conformité)
```

8.4 Haute Disponibilité

Objectifs :

- Disponibilité : 99.95% (< 4h downtime/an)
- RTO : 4 heures
- RPO : 1 heure

Mécanismes :

MongoDB Atlas :

- 3 nœuds (Multi-AZ)
- Failover automatique (< 30 secondes)
- Réplication synchrone

ECS Fargate :

- Tasks dans 2 AZ minimum
- Auto-healing (relance automatique)
- Health checks ALB

Réseau :

- ALB multi-AZ
- NAT Gateway redondant
- Route 53 health checks

Données :

- Snapshots quotidiens
- PITR (Point-in-Time Recovery)
- Cross-region replication

Plan de Disaster Recovery :

Scénario	Impact	Solution	RTO
Panne 1 AZ	Aucun	Failover auto	< 1 min
Panne région Paris	Service down	Restore région Francfort	4h
Corruption données	Perte partielle	PITR ou snapshot	2h
Suppression accidentelle	Perte partielle	Restore backup	1h
Cyberattaque	Variable	Isolation + restore	4h

```
9. Plan de Migration
9.1 Phases de Déploiement

Phase 1 : Préparation (Semaine 1)

Tâches :

    ✅ Créer compte AWS
    ✅ Configurer IAM (utilisateurs, rôles)
    ✅ Créer VPC et subnets
    ✅ Configurer Security Groups
    ✅ Activer CloudTrail et Config

Livrables :

    Compte AWS sécurisé
    Infrastructure réseau prête
    Documentation d'architecture

Responsable : Équipe DevOps

Phase 2 : MongoDB Atlas (Semaine 2)

Tâches :

    ✅ Créer compte MongoDB Atlas
    ✅ Provisionner cluster M10 (Paris)
    ✅ Configurer IP Whitelist
    ✅ Créer utilisateurs et rôles
    ✅ Activer backups automatiques
    ✅ Tester connexion depuis AWS

Livrables :

    Cluster MongoDB opérationnel
    Credentials sécurisés
    Tests de connexion validés

Responsable : Data Engineer

Phase 3 : Conteneurisation (Semaine 3)

Tâches :

    ✅ Adapter scripts Python pour cloud
    ✅ Créer Dockerfile optimisé
    ✅ Créer repository ECR
    ✅ Pousser image vers ECR
    ✅ Tester image localement

Livrables :

    Image Docker fonctionnelle
    Image dans ECR
```

```
    Tests unitaires passés

Responsable : Data Engineer

Phase 4 : ECS Deployment (Semaine 4)

Tâches :

    ✅ Créer Task Definition
    ✅ Créer ECS Cluster
    ✅ Configurer Secrets Manager
    ✅ Déployer premier service ECS
    ✅ Configurer ALB
    ✅ Tester migration manuelle

Livrables :

    Service ECS opérationnel
    Migration testée avec succès
    Logs dans CloudWatch

Responsable : DevOps + Data Engineer

Phase 5 : Automatisation (Semaine 5)

Tâches :

    ✅ Créer EventBridge Rule (cron)
    ✅ Configurer Auto Scaling
    ✅ Mettre en place monitoring
    ✅ Configurer alertes SNS
    ✅ Tester migrations automatiques

Livrables :

    Migrations automatiques quotidiennes
    Monitoring complet
    Alertes fonctionnelles

Responsable : DevOps

Phase 6 : Sauvegardes (Semaine 6)

Tâches :

    ✅ Configurer AWS Backup
    ✅ Créer Lambda export S3
    ✅ Configurer lifecycle policies
    ✅ Tester restauration
    ✅ Documenter procédures

Livrables :

    Stratégie de backup opérationnelle
```

```
    Tests de restauration validés
    Runbook de disaster recovery

Responsable : DevOps + Data Engineer

Phase 7 : Sécurité (Semaine 7)

Tâches :

    ✅ Audit de sécurité (IAM, SG, etc.)
    ✅ Activer GuardDuty
    ✅ Configurer WAF (si API publique)
    ✅ Scanner les images (ECR Scanning)
    ✅ Tester plan de DR

Livrables :

    Rapport d'audit de sécurité
    Remédiation des vulnérabilités
    Plan de DR testé

Responsable : Équipe Sécurité

Phase 8 : Go-Live (Semaine 8)

Tâches :

    ✅ Migration des données de production
    ✅ Validation complète
    ✅ Formation des équipes
    ✅ Documentation finale
    ✅ Handover aux Ops

Livrables :

    Production opérationnelle
    Documentation complète
    Équipes formées

Responsable : Chef de Projet
9.2 Checklist de Pré-Production

Infrastructure :

    VPC configuré avec subnets publics/privés
    Security Groups restrictifs
    NAT Gateway pour accès Internet sortant
    VPC Flow Logs activés
    CloudTrail activé

MongoDB :

    Cluster M10 provisionné (Multi-AZ)
    Backups automatiques activés
```

```
    IP Whitelist configurée
    Utilisateurs créés avec rôles appropriés
    Index créés
    Vue anonymisée créée

ECS :

    Image Docker dans ECR
    Task Definition validée
    Service ECS déployé
    Health checks configurés
    Auto Scaling configuré

Secrets :

    Credentials dans Secrets Manager
    Rotation automatique activée
    Pas de secrets en dur dans le code
    IAM Roles configurés correctement

Monitoring :

    CloudWatch Logs configurés
    Métriques custom créées
    Dashboard CloudWatch créé
    Alertes SNS configurées
    Budget alerts activés

Sauvegardes :

    AWS Backup configuré
    Lambda export S3 déployée
    Tests de restauration effectués
    Cross-region replication activée

Sécurité :

    MFA activé sur tous les comptes
    GuardDuty activé
    Config activé
    ECR Image Scanning activé
    Audit de sécurité passé

Documentation :

    Architecture documentée
    Runbooks créés
    Plan de DR documenté
    Formation équipes effectuée

9.3 Timeline Complète

Semaine 1  : [████████] Préparation AWS
Semaine 2  : [████████] MongoDB Atlas
```

```
Semaine 3  : [████████] Conteneurisation
Semaine 4  : [████████] ECS Deployment
Semaine 5  : [████████] Automatisation
Semaine 6  : [████████] Sauvegardes
Semaine 7  : [████████] Sécurité
Semaine 8  : [████████] Go-Live

═══════════════════════════════════════════
TOTAL : 8 semaines (2 mois)

Ressources nécessaires :

    1 Data Engineer (temps plein)
    1 DevOps Engineer (50% temps)
    1 Architecte Cloud (conseil ponctuel)
    1 Expert Sécurité (audit final)

Budget estimé :

    AWS (2 mois) : 200€
    MongoDB Atlas (2 mois) : 114€
    Temps équipe : Variable selon organisation
    Total infrastructure : ~314€

10. Conclusion et Recommandations
10.1 Synthèse

Le passage au cloud AWS pour notre projet Healthcare MongoDB apporte des
bénéfices considérables :

Avantages Techniques :
✅ Scalabilité : De 54K à 1M+ patients sans refonte
✅ Disponibilité : 99.95% SLA (vs ~95% on-premise)
✅ Performance : Latence < 50ms, 3000+ IOPS
✅ Sécurité : Conformité RGPD/HIPAA native
✅ Maintenance : Zéro gestion d'infrastructure

Avantages Économiques :
✅ CAPEX → OPEX : Pas d'investissement initial
✅ Coût prévisible : ~70€/mois (vs ~1,500€/mois on-premise avec gestion)
✅ ROI : Rentable dès le premier mois
✅ Élasticité : Paiement à l'usage réel

Avantages Opérationnels :
✅ Time-to-market : Déploiement en 8 semaines
✅ Focus métier : Équipe concentrée sur la valeur
✅ Innovation : Accès à 200+ services AWS
✅ Résilience : Disaster Recovery intégré
10.2 Architecture Recommandée (Récapitulatif)

Stack Technique :
Composant   Solution    Justification
Base de Données     MongoDB Atlas M10   Géré, fiable, performant
Compute     ECS Fargate     Serverless, simple, économique
```

```
Stockage    S3 Standard + Glacier   Durable, économique
Réseau  VPC + ALB   Sécurisé, haute dispo
Monitoring  CloudWatch  Intégré, complet
Secrets     Secrets Manager     Sécurisé, rotation auto
Backup  Atlas + AWS Backup  Redondant, testé

Coûts Mensuels :

MongoDB Atlas M10      : 57€
ECS Fargate            : 0,02€ (usage ponctuel)
S3 Backup              : 2€
CloudWatch             : 5€
Secrets Manager        : 0,40€
Data Transfer          : 1€
AWS Backup             : 3€
─────────────────────────
TOTAL                  : ~68€/mois
                         ~816€/an

10.3 Recommandations Finales

Pour le Client Healthcare :

Court Terme (3 mois) :

    ✅ Déployer architecture proposée
    ✅ Migrer données existantes
    ✅ Former les équipes
    ✅ Monitorer et optimiser

Moyen Terme (6-12 mois) :

    📈 Évaluer Reserved Instances (économie 30%)
    📈 Ajouter API REST (ECS + API Gateway)
    📈 Implémenter Machine Learning (SageMaker)
    📈 Déployer multi-région si besoin

Long Terme (1-2 ans) :

    🚀 Migrer vers architecture microservices
    🚀 Implémenter Data Lake (S3 + Athena)
    🚀 BI et analytics avancés (QuickSight)
    🚀 IoT si devices médicaux (IoT Core)

Risques et Mitigation :
Risque  Probabilité     Impact  Mitigation
Dépassement budget  Moyenne     Moyen   AWS Budgets + alertes
Panne région    Faible  Élevé   Backup cross-region
Faille sécurité     Faible  Critique    GuardDuty + audits réguliers
Perte de données    Très faible     Critique    Backups multiples + tests
Vendor lock-in  Moyenne     Moyen   Architecture portable (Docker)

Plan de Contingence :
```

```
    Si budget dépassé : Réduire à M2 Atlas + optimiser ECS
    Si problème Atlas : Basculer sur DocumentDB (préparé)
    Si problème AWS : Backup local disponible (restore manuel)

10.4 Prochaines Étapes

Actions Immédiates (Cette Semaine) :

    Présenter cette documentation au client
    Obtenir validation budget (~70€/mois)
    Créer compte AWS (si validé)
    Créer compte MongoDB Atlas

Actions Court Terme (Mois 1) :

    Déployer architecture (Phases 1-4)
    Migrer données de test
    Valider avec le client

Actions Moyen Terme (Mois 2-3) :

    Migration production
    Formation équipes
    Optimisation coûts
    Documentation complète

10.5 Ressources Complémentaires

Documentation AWS :

    AWS Well-Architected Framework :
https://aws.amazon.com/architecture/well-architected/
    ECS Best Practices :
https://docs.aws.amazon.com/AmazonECS/latest/bestpracticesguide/
    MongoDB on AWS : https://www.mongodb.com/cloud/atlas/aws

Formation :

    AWS Training : https://aws.amazon.com/training/
    MongoDB University : https://university.mongodb.com/
    Certifications recommandées : AWS Solutions Architect Associate

Support :

    AWS Support (Plan Business recommandé) :
https://aws.amazon.com/premiumsupport/
    MongoDB Support : https://www.mongodb.com/support
    Communauté : AWS Forums, Stack Overflow

Annexes
Annexe A : Glossaire

AWS (Amazon Web Services) : Plateforme cloud d'Amazon
ECS (Elastic Container Service) : Service d'orchestration de conteneurs
```

```
Fargate : Mode serverless pour ECS
ALB (Application Load Balancer) : Répartiteur de charge applicatif
VPC (Virtual Private Cloud) : Réseau virtuel isolé
IAM (Identity and Access Management) : Gestion des identités et accès
S3 (Simple Storage Service) : Service de stockage objet
RTO (Recovery Time Objective) : Temps maximum de récupération
RPO (Recovery Point Objective) : Perte de données acceptable
SLA (Service Level Agreement) : Accord de niveau de service
Annexe B : Contacts Utiles

Support AWS :

    Email : aws-support@amazon.com
    Téléphone : +33 1 76 54 00 00
    Chat : Console AWS → Support

Support MongoDB Atlas :

    Email : support@mongodb.com
    Chat : Atlas Console → Support
    Documentation : https://docs.atlas.mongodb.com/

Communauté :

    AWS User Group Paris : https://www.meetup.com/fr-FR/paris-aws-ug/
    MongoDB Paris : https://www.meetup.com/fr-FR/Paris-MongoDB-User-Group/

Annexe C : Checklist de Conformité RGPD

    Données hébergées en UE (Paris)
    Chiffrement des données personnelles
    Journalisation des accès (CloudTrail)
    Droit à l'oubli implémenté (API de suppression)
    Droit à la portabilité (export JSON)
    DPO désigné
    Registre des traitements tenu
    Analyse d'impact (AIPD) effectuée
    Contrat DPA signé avec AWS et MongoDB
    Formation RGPD des équipes

FIN DU DOCUMENT

Auteur : Behram Korkut - Data engineer
Date : Octobre 2025
Version : 1.0
Statut : Documentation de recherche .

Note : Ce document est une synthèse de recherches pour un projet
pédagogique. Les prix et configurations sont indicatifs et peuvent varier.
Toujours vérifier les tarifs actuels sur https://calculator.aws/.
```