


## ♦ 1. Qu'est-ce qu'Amazon SQS ?

Amazon SQS (Simple Queue Service) est un **service de file d'attente de messages**.

Il permet à différentes parties d'une application (ou plusieurs services AWS) de **communiquer entre elles de façon asynchrone**.

### Image mentale :

Pense à une boîte aux lettres  :

- Un service **dépose un message** dedans
- Un autre service **vient le récupérer plus tard**

## Pourquoi utiliser SQS ?

- Pour **découpler** les composants d'une application
- Pour **éviter la perte de messages**
- Pour **gérer des pics de charge** (buffer)
- Pour **rejouer ou traiter plus tard** certaines actions

## ♦ 2. Types de files SQS

- **Standard Queue** : Haute capacité, traitement "au moins une fois", l'ordre n'est pas garanti.
- **FIFO Queue (First-In-First-Out)** : Traitement **exactement une fois** et dans le même ordre.

Envoyé : CMD001 → CMD002 → CMD003 → CMD004

Consommé (ordre possible) :

→ CMD003

→ CMD001

→ CMD004

→ CMD002

→ CMD003

→ CMD001

→ CMD003 (doublon)

→ CMD004

## ♦ 3. Fonctionnement de base

- Un producteur **envoie** des messages dans une file.
- Un consommateur **lit** et **supprime** les messages après traitement.
- Les messages ont une durée de rétention configurable (jusqu'à 14 jours).
- Il est possible d'utiliser **des délais** et **des délais de visibilité**.

#### ♦ 4. Cas d'usage

- Traitement par lots
- Communication entre microservices
- Tampons pour systèmes avec débits différents

### 5. Exemples réel : Commandes e-commerce

- **Contexte** : Lorsqu'un client passe une commande, plusieurs actions doivent être réalisées : facturation, mise à jour du stock, envoi d'email de confirmation, préparation logistique.
- **Architecture** :
  - L'application envoie un message dans SQS pour chaque tâche à exécuter.
  - Chaque microservice (paiement, stock, logistique, etc.) consomme les messages correspondants.
  - Cela permet de **découpler** les traitements et d'**éviter les pannes en cascade**.

## Scénario : Gestion d'un système de notification dans un environnement de microservices

### Contexte :

Une entreprise a une application qui utilise plusieurs **microservices** pour différents domaines fonctionnels :

- **Service Utilisateurs** : Gère les informations des utilisateurs.
- **Service Notification** : Envoie des e-mails ou SMS pour les actions effectuées par l'utilisateur.
- **Service Commandes** : Gère les commandes passées par les utilisateurs.

L'objectif est de dé-coupler les services afin qu'ils puissent communiquer entre eux de manière **asynchrone** sans être directement liés.

---



## Utilisation de AWS SQS dans ce scénario :

1. **Service Commandes** crée une commande et, une fois que celle-ci est validée, il envoie un message dans une **file SQS** pour notifier le **Service Notification** que l'utilisateur doit recevoir un e-mail de confirmation.
2. Le **Service Notification**, après avoir reçu le message de la file SQS, envoie un e-mail à l'utilisateur pour confirmer la commande.
3. Les deux services peuvent fonctionner indépendamment et n'ont pas besoin de savoir quand l'autre service est en cours d'exécution, grâce à l'utilisation de SQS.



## Avantages de ce scénario avec SQS :

- **Découplage des microservices** : Le **Service Commandes** n'a pas besoin d'attendre une réponse immédiate du **Service Notification**. Cela permet à chaque service d'être autonome.
- **Scalabilité** : Si le nombre de commandes augmente, le service **Notification** peut traiter plus de messages simultanément en augmentant le nombre de workers qui lisent depuis la file SQS.
- **Fiabilité** : Si le **Service Notification** est temporairement hors ligne ou rencontre une erreur, le message reste dans SQS et sera traité plus tard, sans perte de données.

## ♦ 5. Intégrations possibles

- AWS Lambda
- Amazon EC2
- AWS SNS (pour publier un message sur plusieurs files)
- CloudWatch (monitoring)
- IAM (contrôle d'accès)



## Standard vs FIFO dans Amazon SQS

Fonctionnalité	File Standard	File FIFO
Ordre	Pas garanti (best effort)	Garanti (First-In-First-Out)
Livraison	Au moins une fois (peut y avoir des doublons)	Exactement une fois
Débit	Très élevé (des milliers/sec)	Limité : jusqu'à 300 msg/sec (avec batching)
Usage courant	Traitement parallèle, haute performance, tolère l'ordre aléatoire	Paiements, commandes, workflows (ordre strict requis)
Coût	Moins cher	Un peu plus cher
Nom	N'importe quel nom	Doit se terminer par <code>.fifo</code>

## Étape 1 : Créer une file d'attente SQS

1. Connecte-toi à la console AWS : <https://console.aws.amazon.com>
2. Dans le menu **Services**, cherche et ouvre **SQS**.
3. Clique sur "**Créer une file d'attente**".
4. Choisis :
  - Type : **Standard** (suffisant pour les tests)
  - Nom : **MaQueueTest**
5. Laisse les options par défaut (tu peux explorer si tu veux)
6. Clique sur "**Créer la file d'attente**"

☒ Ta file est créée, tu es redirigé vers la page de détails.

---

## Étape 2 : Envoyer un message

1. Dans les détails de ta file **MaQueueTest**, clique sur "**Envoyer et recevoir des messages**".
  2. Dans la zone "**Envoyer un message**" :
    - Message body : `{"message": "Hello SQS depuis la console AWS"}`
  3. Clique sur "**Envoyer le message**".
  4. Tu verras une confirmation : ☒ *Message envoyé avec succès*
- 

## Étape 3 : Recevoir (lire) un message

1. Reste sur la même page (**Envoyer et recevoir des messages**).
  2. Clique sur "**Recevoir des messages**"
  3. Tu verras un ou plusieurs messages dans la liste.
  4. Clique sur "**Afficher les détails**" pour lire le contenu.
-

## Étape 4 : Supprimer un message

1. Coche la case à gauche du message reçu.
2. Clique sur "**Supprimer**"
3. Confirme la suppression.

☒ Le message est maintenant supprimé et ne pourra plus être traité.

---

## Étape 5 : Supprimer la file (optionnel)

1. Retourne dans la liste des files (menu SQS > Files d'attente).
2. Coche ta file **MaQueueTest**.
3. Clique sur **Actions > Supprimer**.
4. Confirme.

## TP SQS + S3 via Console AWS (pas de code, tout en interface)

---

### ♦ Étape 1 : Créer une file SQS

1. Va dans le service **SQS**.
  2. Clique sur **Créer une file d'attente**
  3. Choisis **Standard**, nomme-la par exemple **FileS3Events**
  4. Clique sur **Créer la file**
- 

### ♦ Étape 2 : Créer un bucket S3

1. Va dans le service **S3**
2. Clique sur **Créer un bucket**

3. Nom du bucket : **mon-bucket-sqs-test** (ou autre nom unique globalement)
4. Laisse les options par défaut
5. Clique sur **Créer un bucket**

---

### ♦ Étape 3 : Donner à S3 le droit d'envoyer vers SQS

On va modifier la **policy de la file SQS** pour autoriser S3 à y publier des messages.

1. Retourne dans **SQS > FileS3Events**
2. Va dans l'onglet **Permissions > Access Policy**
3. Clique sur **Edit** et remplace la politique par celle-ci (en mettant le bon ARN de ton bucket S3) :

```
{
  "Version": "2012-10-17",
  "Id": "s3-event-policy",
  "Statement": [
    {
      "Sid": "AllowS3ToSendMessage",
      "Effect": "Allow",
      "Principal": {
        "Service": "s3.amazonaws.com"
      },
      "Action": "SQS:SendMessage",
      "Resource": "arn:aws:sqs:<région>:<id-compte>:FileS3Events",
      "Condition": {
        "ArnLike": {
          "aws:SourceArn": "arn:aws:s3:::mon-bucket-sqs-test"
        }
      }
    }
  ]
}
```

Remplace :

- **<région>** par ta région AWS (ex: **eu-west-3**)
- **<id-compte>** par ton AWS Account ID
- **mon-bucket-sqs-test** par le nom réel de ton bucket

### ♦ Étape 4 : Configurer S3 pour envoyer vers SQS

1. Va dans **S3 > ton bucket**


2. Clique sur **Properties**
3. Descends jusqu'à **Event notifications**
4. Clique sur **Create event notification**
  - Nom : **VersSQS**
  - Événements : coche **All object create events**
  - Destination : choisis **SQS queue**
  - Sélectionne **FileS3Events**
5. Clique sur **Save**

### ♦ **Étape 5 : Tester** 🔥

1. Va dans ton bucket S3
2. Clique sur **Upload > ajouter un fichier**
3. Uploade un fichier (image, texte, etc.)

 Attends quelques secondes...

4. Va dans **SQS > FileS3Events > Envoyer et recevoir des messages**
5. Clique sur **Recevoir des messages**

 Tu devrais voir un message avec les détails de l'événement S3 (bucket, nom du fichier, etc.)

---

### **Étape 6 : Nettoyage (optionnel)**

- Supprime la notification dans le bucket
- Supprime la file SQS
- Supprime le bucket S3 si plus nécessaire