Mise à l'échelle et équilibrage de charge TP

Objectif global du TP

Le but de ce TP est de comprendre comment AWS gère automatiquement la charge d'un site web quand il y a plus (ou moins) de visiteurs.

L'objectif ici est de construire une architecture AWS qui :

- Ne tombe pas en panne si un serveur meurt.
- Supporte beaucoup d'utilisateurs en même temps.
- Ajoute automatiquement des serveurs quand il y a trop de charge.
- En supprime quand il n'y a plus besoin (pour payer moins cher).

Pour faire ça, on utilise **3 briques clés** :

1. **AMI** → un "clone" du serveur.
2. **Load Balancer (ELB)** → un répartiteur de trafic.
3. **Auto Scaling** → un gestionnaire automatique de serveurs.

Situation de départ

- Tu as un seul serveur web (Web Server 1).
- Ce serveur fonctionne correctement.
- Mais un seul serveur = risque (surcharge ou panne).

Situation finale

- Plusieurs serveurs web identiques.
- Un équilibreur de charge devant.
- AWS ajoute/enlève des serveurs automatiquement.

TÂCHE 1 – Créer une AMI du serveur Web (Web Server 1)

Auto Scaling doit pouvoir **créer de nouveaux serveurs automatiquement en cas de pics de charge trop élevé par exemple.** Mais pour ce faire AWS a besoin de savoir à quoi doit ressembler les serveurs. On fait donc une AMI (Amazon Machine Image) de notre serveur.

Une photo complète du serveur contient :

- Système d'exploitation
- Serveur web installé
- Fichiers du site
- Configuration

Donc tous les nouveaux serveurs seront **des copies parfaites** du serveur de départ.

Pour ce faire concrètement il faut :

1. Aller dans **EC2** puis **Instances**
2. Vérifier que **Web Server 1 fonctionne bien**
    - (2/2 vérifications réussies = serveur OK)
        - System Status Check : vérifie que l'infrastructure AWS fonctionne correctement.
        - Instance Status Check : vérifie que le système d'exploitation de l'instance répond bien.
3. Puis créé une image (AMI) à partir de lui

Résultat on obtient une AMI appelée par exemple **WebServerAMI** et elle servira de base à toutes les futures instances que génèrera **Auto Scaling.**

Tâche 2 — Créer un équilibreur de charge (Load Balancer ELB)

Imaginons 100 utilisateurs qui arrivent sur notre site.

Sans Load Balancer :

- Tous tapent sur **un seul serveur**
- Il sature et crash

Avec Load Balancer :

- Les requêtes sont reçues d'un point d'entrée **unique** puis sont **réparties intelligemment** vers plusieurs serveurs.
- Chaque serveur travaille un peu
- Le site reste fluide

Il agit comme un agent de circulation et pour le mettre en place, il faut passer par plusieurs étapes.

Étape 1 : Groupe cible (Target Group)

Un groupe cible = une liste de serveurs vers lesquels le Load Balancer a le droit d'envoyer du trafic.

Le Load Balancer ne parle jamais directement aux instances pour **séparer les responsabilités et permettre l'automatisation**. Il parle au **groupe cible**, qui contient **les instances**.

Raisons principales : Compatibilité avec Auto Scaling

- Les instances **apparaissent et disparaissent** automatiquement
- Le **Target Group se met à jour tout seul**
- Le Load Balancer n'a rien à changer

Sans Target Group, le Load Balancer serait "cassé" à chaque création/suppression d'instance.

Ici, on en a un seul, car une seule application web.

Il faut créer le groupe cible :

- Type de cible : Instances EC2
- Nom : **LabGroup**
- VPC : Lab VPC (localisation des serveurs)

On ajoute aucune instance pour l'instant c'est normal parce que les serveurs n'existent pas encore. Auto Scaling les ajoutera plus tard automatiquement.

Maintenant que la “liste des serveurs” existe, on peut créer celui qui va répartir le trafic.

Étape 2 : Créer l’Application Load Balancer et son Listener

Pourquoi un Application Load Balancer ?

Parce qu’il :

- Comprend le HTTP (niveau applicatif).
- Peut répartir selon les requêtes.
- Est parfait pour une appli web.

Choix importants (et pourquoi)

Sous-réseaux publics

- Le Load Balancer doit être accessible depuis Internet Parce que les utilisateurs viennent d’Internet.
- Donc il est dans des subnets publics.

Deux zones de disponibilité

- Si une zone AWS tombe → l’autre continue = haute disponibilité

Groupe de sécurité Web

- Autorise le trafic HTTP (port 80)
- On enlève le groupe par défaut pour éviter les règles inutiles

Listener HTTP → LabGroup

- Quand quelqu’un arrive sur le port 80
- le trafic est envoyé vers LabGroup notre groupe cible

Résultat le Load Balancer est prêt, quand quelqu’un arrive sur le port 80, il envoie sa requête et si ok sera dirigé vers LabGroup.

Tâche 3 — Auto Scaling (le cerveau automatique)

Auto Scaling :

- Lance des serveurs quand il y a trop de charge.
- En supprime quand il y en a trop peu.
- Respecte des limites qui ont été définis.

Étape 1 : Modèle de lancement (Launch Template)

C'est la **fiche technique** d'un serveur il contient :

- L'AMI (le système + le site)
- Le type d'instance (t2.micro)
- Le groupe de sécurité
- La clé SSH
- Les paramètres de monitoring (CloudWatch)

Auto Scaling l'utilise **à chaque fois** qu'il crée une instance il aura besoin de toute ces informations car une AMI à elle seule **ne peut pas lancer une instance**.

Choix importants

- **AMI** : celle que qui a été créée, elle garantit que chaque serveur est identique.
- **t2.micro** : petit, pas cher, suffisant pour le TP.
- **Cloud Watch détaillé activé** permet de réagir vite à la charge.

Étape 2 : Groupe Auto Scaling

Paramètres clés (importants)

Taille du groupe

- Minimum : 2 → toujours au moins 2 serveurs
- Souhaitée : 2 → au démarrage
- Maximum : 6 → limite de sécurité

Stratégie de mise à l'échelle

- Métrique : CPU moyen
- Cible : 60 %

Même si tout explose, AWS ne dépassera pas 6 instances.

Auto Scaling enregistre automatiquement les nouvelles instances dans LabGroup.

AWS, fais en sorte que les serveurs tournent autour de 60 % de CPU. **Si ça dépasse → ajout des serveurs** et si c'est **trop bas → enlève-en**. Résultat Auto Scaling démarre 2 nouvelles si l'utilisation CPU > 60 %, les instances sont créées automatiquement.

Tâche 4 — Vérifier que tout fonctionne

Ce qu'il faut vérifier maintenant :

1. Les instances existent
   - Vérifier qu'Auto Scaling a bien créé le nombre d'instances souhaité.

2. Le Load Balancer les voit
   - S'assurer que toutes les instances sont enregistrées dans le Target Group **LabGroup**.

3. Elles sont "saines"
   - Chaque instance répond correctement aux vérifications du Target Group.

4. Le site est accessible via le Load Balancer
   - Copier l'adresse DNS de l'ALB et tester que la page s'affiche correctement.

Tâche 5 — Tester la mise à l'échelle automatique

Comment fonctionne le test de charge

1. On génère une forte charge CPU sur les instances (ex : script ou outil de test).
2. **Cloud Watch mesure le CPU moyen** de toutes les instances.
3. Si **CPU moyen > 60 %** :
   - Auto Scaling ajoute automatiquement **de nouvelles instances**.
4. Si **CPU moyen < 60 %** :
   - Auto Scaling supprime les instances **excédentaires** pour optimiser les coûts.
5. On peut **voir visuellement** le nombre d'instances augmenter et diminuer dans la console.