Projet AWS

Déploiement d'une application web sur EC2 avec Auto Scaling et Load Balancer

Introduction

Ce projet consiste à créer une infrastructure sur le cloud en utilisant plusieurs services proposés par AWS.

On commence par déployer des serveurs (instances EC2), puis on met en place un Load Balancer pour répartir le trafic.
Ensuite, on ajoute des règles de sécurité, un certificat SSL pour sécuriser les échanges, et un système de scaling automatique pour que l'infrastructure s'adapte en fonction de la charge.

Ce projet permet de comprendre comment construire une architecture cloud efficace, disponible et sécurisée.

Pour avoir 14 jours gratuit en installant un serveur en local à partir de cet URL : https://localstack.cloud/

PS : il faut installer docker en local sur votre machine (voir base de connaissances)

Job 1

Objectif: Déployer une application web sur un serveur EC2 avec une mise à l'échelle automatique pour gérer la charge.

Ressources AWS: EC2, Auto Scaling Group, Application Load Balancer, Security Groups

Étapes:

- Créer une instance EC2 avec l'application web.
- Configurer un Auto Scaling Group pour ajouter/supprimer des instances selon la charge.
- Configurer un Application Load Balancer pour distribuer le trafic.

Coût: Le niveau gratuit d'EC2 vous donne jusqu'à 750 heures par mois sur t2.micro.

Job 2

Objectif: Héberger un site web statique avec une distribution CloudFront pour une meilleure performance et sécurité.

Ressources AWS: S3, CloudFront, IAM

Étapes:

- Créer un bucket S3 pour héberger les fichiers statiques.
- Activer l'hébergement de site web sur le bucket S3.
- Configurer une distribution CloudFront pour distribuer le contenu globalement.

Coût: L'hébergement statique sur S3 et les premières 1 000 minutes CloudFront sont gratuites.

Job 3

Objectif: Déployer une base de données relationnelle avec des sauvegardes automatiques pour un stockage sécurisé des données.

Ressources AWS: Amazon RDS (MySQL ou PostgreSQL), IAM
Étapes:

- Configurer une instance RDS MySQL/PostgreSQL.
- Configurer les paramètres de sauvegarde automatique et les snapshots.

Coût: 750 heures d'utilisation mensuelle gratuites pour une instance de type db.t2.micro (RDS).

Job 4

Objectif: Créer une API RESTful sans serveur qui répond à des requêtes HTTP.

Ressources AWS: Lambda, API Gateway, CloudWatch Logs

Étapes:

- Développer une fonction Lambda en Python ou Node.js.
- Configurer une API dans API Gateway pour déclencher la fonction Lambda.
- Activer CloudWatch pour la journalisation.

Coût: 1 million de requêtes gratuites pour Lambda et 1 million d'appels API par mois pour API Gateway.

Job 5

Objectif: Mettre en place une surveillance et des alertes pour une application ou un service AWS.

Ressources AWS: CloudWatch, SNS (Simple Notification Service)

Étapes:

- Configurer des métriques CloudWatch pour surveiller les ressources (CPU, mémoire, etc.).
- Créer des alarmes pour les métriques.
- Utiliser SNS pour envoyer des notifications par email en cas de dépassement de seuil.

Coût: Le niveau gratuit inclut 10 alarmes et 1 000 notifications SNS.

Job 6

Objectif: Automatiser l'ingestion et la transformation de données en utilisant un pipeline de données.

Ressources AWS: AWS Glue, S3, IAM

Étapes :

- Charger des fichiers de données bruts dans S3.

- Créer un crawler dans AWS Glue pour détecter les schémas des données.
- Utiliser Glue pour transformer les données et les sauvegarder dans S3.

Coût: Le niveau gratuit inclut 1 million de requêtes Glue par mois.

Job 7

Objectif: Effectuer des requêtes SQL sur des données stockées dans S3 et visualiser les résultats avec QuickSight.

Ressources AWS: S3, Athena, QuickSight

Étapes:

- Importer des fichiers de données dans S3.
- Utiliser Athena pour interroger les données directement dans S3.
- Créer des tableaux de bord dans QuickSight pour visualiser les résultats.

Coût: 1 million de requêtes gratuites par mois pour Athena.

Job 8

Objectif: Déployer une application conteneurisée sans gérer l'infrastructure sous-jacente.

Ressources AWS: ECS, Fargate, ECR (Elastic Container Registry)

Étapes:

- Créer et pousser l'image Docker de l'application dans ECR.
- Configurer un cluster ECS avec Fargate.
- Déployer le conteneur sur ECS et le rendre accessible publiquement.

Coût: 750 heures par mois gratuites pour Fargate et 500 Mo gratuits pour ECR.

Job 9

Objectif: Automatiser une série de tâches de calcul sans serveur pour un flux de travail d'application.

Ressources AWS: Step Functions, Lambda, CloudWatch

Étapes:

- Créer des fonctions Lambda pour chaque étape du processus.
- Configurer un workflow dans Step Functions pour orchestrer les fonctions Lambda.
- Surveiller les exécutions avec CloudWatch Logs.

Coût: 4 000 transitions gratuites par mois avec Step Functions.

Compétences visées

- Administrer et sécuriser les infrastructures virtualisées
- Mettre en production des évolutions de l'infrastructure
- Participer à la détection et au traitement des incidents de sécurité
-

Rendu

Présentation avec diapositive et documentation sur la totalité du projet, veille, concept et architecture.

Base de connaissances

- AWS Load Balancer
- AWS Scaling Group
- AWS Scaling Policy
- Install Docker Engine on Ubuntu
- Docker Install server and client binaries on Windows