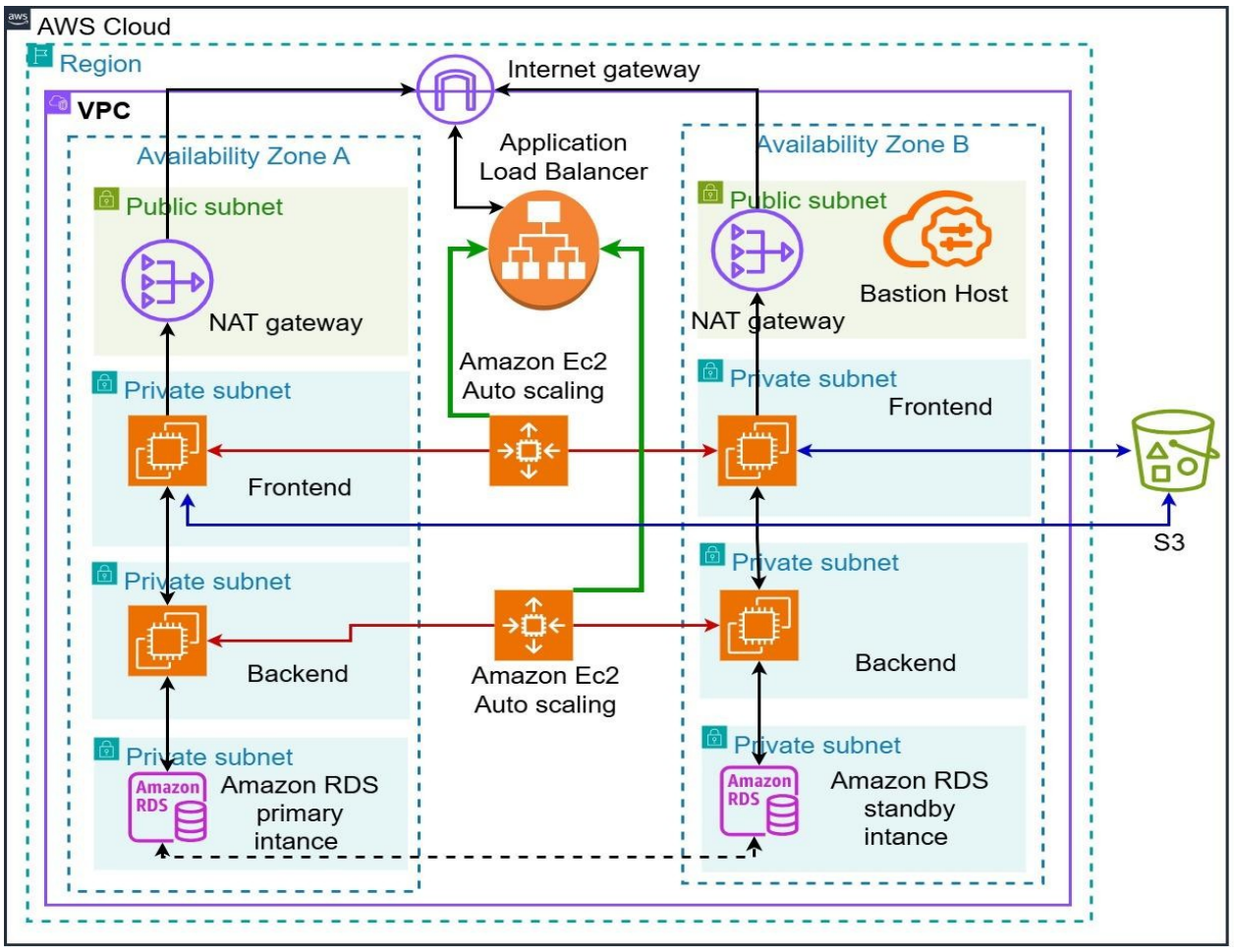
Projet d'Évaluation AWS : Déploiement d'une Application 3 Niveaux
Projet AWS- IGL4 et GLSI3

Créer une architecture cloud sécurisée et scalable sur AWS pour une application web moderne avec :

- Un frontend (React/Angular)
- Un backend (Node.js/Python/Java)
- Une base de données managée avec Amazon RDS (PostgreSQL/MySQL)



Étapes du Projet

Étape 1 : Créer un VPC et ses composants réseau

1. Créer un VPC avec une plage CIDR (ex: 10.0.0.0/16)
2. Créer 6 subnets :
    - 2 subnets publics (un dans chaque zone de disponibilité A et B)
    - 4 subnets privés : 2 pour le frontend/backend dans chaque zone
3. Créer une Internet Gateway (IGW) et l'attacher au VPC
4. Créer 2 NAT Gateways (un par AZ), placés dans les subnets publics

5. Créer les tables de routage :
    - Associer la route vers IGW aux subnets publics
    - Associer la route vers NAT Gateway aux subnets privés

Étape 2 : Créer les groupes de sécurité (Security Groups)

1. SG-LB : pour le Load Balancer (HTTP/HTTPS depuis Internet)
2. SG-FE : pour les instances frontend (accepte trafic du SG-LB)
3. SG-BE : pour le backend (accepte trafic du SG-FE, ex: port 8080)
4. SG-DB : pour la base de données (accepte trafic du SG-BE, ex: port 3306)
5. SG-Bastion : accès SSH depuis une IP fixe (celle de votre poste de travail).

Étape 3 : Déployer les ressources EC2

1. Ajoutez deux instances EC2 pour le frontend, chacune dans un subnet privé différent (un dans la zone de disponibilité A et l'autre dans la zone de disponibilité B).
2. Ajoutez également deux instances EC2 pour le backend, chacune dans un subnet privé distinct (un dans la zone de disponibilité A et l'autre dans la zone de disponibilité B).
3. Machine Bastion : Ajoutez une instance EC2 bastion dans un sous-réseau public pour se connecter aux machines dans les sous-réseaux privés.
4. Configurez un **Application Load Balancer (ALB)** pour gérer le trafic destiné au frontend, ainsi qu'un autre ALB pour le backend (ou, alternativement, utilisez un seul ALB (des **listeners**)).
5. Configurer Auto Scaling Groups pour :
    - Le frontend (dans subnets privés frontend)
    - Le backend (dans subnets privés backend)
6. Définissez des règles de scaling basées sur des métriques comme la CPU Utilization ou le nombre de requêtes.

Étape 4 : Déployer la base de données (Amazon RDS)

1. Créer une instance RDS (ex: MySQL, PostgreSQL , ou autre)
    - Dans les subnets privés
    - Multi-AZ avec primary + standby pour assurer la haute disponibilité.
    - Attacher le SG-DB

Étape 5 : Déployer un bucket S3 et CloudFront

1. Créer un bucket S3
    - Pour stocker les fichiers statiques et les assets de l'application.

2. Créer une CloudFront Distribution
    - Configurez une CloudFront Distribution pour optimiser la diffusion des fichiers stockés sur S3.

Étape 6 : Sécurité Avancée

1. Activer Amazon CloudWatch :
    - Suivi des métriques essentielles : CPU, mémoire, requêtes HTTP.
    - Agrégation des logs via CloudWatch Logs.
2. Configurer les alarmes CloudWatch :
    - Seuils critiques (ex. CPU > 70%).
    - Notifications via Amazon SNS vers :
        - Adresse email (admin@votredomaine.com),
        - Ou canal Slack.
3. Sauvegardes automatiques dans Amazon RDS
4. Activez CloudTrail pour enregistrer toutes les actions sur l'infrastructure.
5. Configurez AWS Certificate Manager (ACM) pour gérer les certificats SSL pour votre domaine.
6. Appliquez ces certificats aux load balancers pour sécuriser les connexions HTTPS.

Étape 7 : Refactorisation avec Conteneurs : Migration vers ECS

Cette étape vise à moderniser l'architecture existante basée sur EC2 en migrant vers une infrastructure conteneurisée plus flexible, évolutive et facile à gérer, à l'aide des services Amazon ECS (Elastic Container Service)

1. Reproduction de l'architecture EC2 avec des conteneurs

- Chaque instance ou service applicatif déployé sur EC2 (ex. : serveur web, base de données, API backend, etc.) est encapsulé dans un **conteneur Docker**.
- Ces conteneurs sont ensuite **déployés dans des clusters ECS** afin de reproduire l'architecture logique initiale.

2. Configuration des Task Definitions pour ECS

- Une **Task Definition** décrit la configuration d'un conteneur (ou d'un groupe de conteneurs) :
    - Image Docker à utiliser
    - Ressources allouées (CPU, mémoire)
    - Variables d'environnement
    - Ports exposés
    - Liens entre conteneurs (si plusieurs services dans la même tâche)
- Ces définitions servent de **plans de déploiement** pour exécuter les services dans le cluster ECS.

3. Mise en place du scaling automatique

- Le **scaling** consiste à ajuster automatiquement le nombre de conteneurs selon la charge du système.
- Cela s'appuie sur des **métriques CloudWatch**, telles que :
    - L'utilisation du CPU
    - La mémoire consommée
    - Le nombre de requêtes réseau ou la latence moyenne

Étape 8 : Documentation

1. Fournissez une documentation complète comprenant les schémas d'architecture, les configurations réseau, les étapes de déploiement.
2. **Critères d'Évaluation :**
    - Respect des meilleures pratiques AWS (sécurité, haute disponibilité, scalabilité).
    - Documentation et clarté des étapes.
    - Fonctionnalité et disponibilité de l'application dans les deux versions (EC2 et Conteneurs).
    - Optimisation des coûts.
    - Gestion et sécurisation des données et secrets.