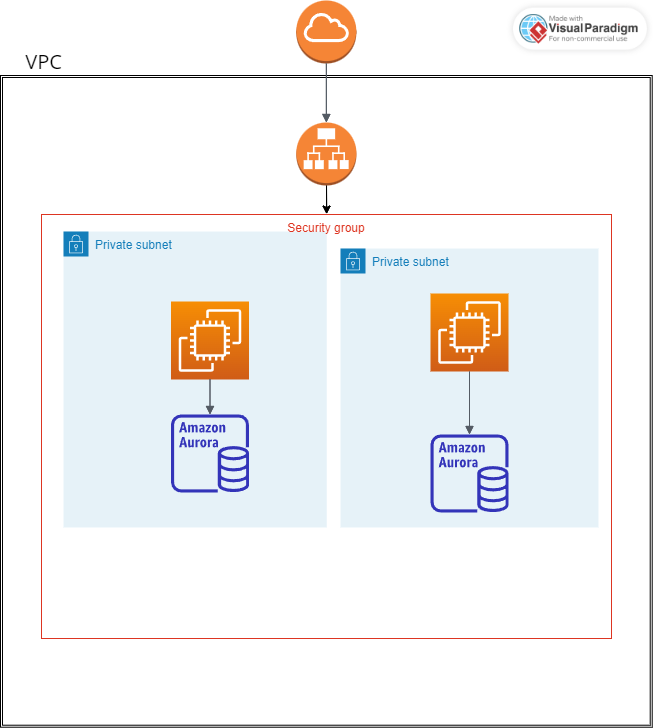
# Exercice 1



# Exercice 2

## Cas 1 : Problème de mémoire dans un pod Kubernetes

Diagnostic :

- Les pics de consommation de mémoire pourraient survenir lors de processus spécifiques ou inattendus dans l'application.

- Possible fuite de mémoire dans l'application.

- Les limites de mémoire configurées ne sont peut-être pas suffisantes pour certaines charges de travail.

Plan d'Action :

- Court Terme :

- Augmenter temporairement les limites de mémoire du pod pour éviter les redémarrages fréquents.

- Analyser les logs et les métriques de performance pour identifier les opérations menant à une haute consommation de mémoire.

- Long Terme :

- Effectuer un profilage de mémoire de l'application pour détecter et corriger les fuites de mémoire.

- Optimiser l'utilisation de la mémoire dans le code de l'application.

- Revoir et ajuster régulièrement les configurations de mémoire en fonction des changements d'usage de l'application.

## Cas 2 : Problème de connexion pour un nouveau client

Diagnostic:

- Problème de configuration spécifique au client, tel que les paramètres réseau, firewall ou DNS.

- Le client pourrait avoir des restrictions IP qui bloquent l'accès au site.

Plan d'Action :

- Court Terme :

- Demander au client de vérifier sa connexion internet, les paramètres de son navigateur, et d'essayer depuis un autre appareil ou réseau.

- Vérifier les logs du serveur pour des erreurs spécifiques liées à ce client.

- Long Terme :

- Revoir la documentation destinée aux clients pour s’assurer qu’elle couvre les problèmes de connexion courants et leurs solutions.

- Améliorer le monitoring et les alertes pour détecter rapidement ce type de problèmes.

## Cas 3 : Problèmes de fichiers statiques dans une SPA

Diagnostic:

- Configuration incorrecte de Nginx ou des chemins de fichiers statiques.

- Problème de synchronisation ou de disponibilité des fichiers entre les pods.

Plan d'Action :

- Court Terme :

- Vérifier et corriger la configuration de Nginx.

- Assurer la cohérence des fichiers statiques déployés sur tous les pods.

- Long Terme :

- Mettre en place des tests automatisés pour vérifier l'intégrité et la disponibilité des fichiers statiques après chaque déploiement.

- Utiliser un système de stockage centralisé pour les fichiers statiques si le problème persiste.

## Cas 4 : Redémarrage en boucle d'un pod après mise à jour

Diagnostic et Hypothèses :

- La mise à jour pourrait nécessiter plus de ressources au démarrage que les limites actuelles allouées.

- Problème avec la nouvelle version de l'application.

Plan d'Action :

- Court Terme :

- Examiner les logs pour comprendre pourquoi l'application ne démarre pas.

- Revenir temporairement à la version précédente si nécessaire pour maintenir la disponibilité.

- Long Terme :

- Tester les mises à jour dans un environnement de staging avant le déploiement en production.

- Adapter les limites des ressources et les dépendances de l'application en fonction des besoins de la nouvelle version.

- Améliorer les processus de CI/CD pour inclure des tests de démarrage post-déploiement.

# Exercice 3

Pour créer un module Terraform, nous allons développer une configuration qui crée un service backend HTTP accessible via un Application Load Balancer (ALB) existant et un worker qui peut communiquer avec ce service backend. Nous supposons que l'ALB et le VPC sont déjà configurés et que les DNS et sous-réseaux nécessaires sont disponibles.

Structure du module Terraform

main.tf - Définit les ressources principales.

variables.tf - Déclare les variables utilisées dans le module.

outputs.tf - Définit les sorties du module