Projet Final : CloudMemo Challenge

Contexte

L'entreprise "StudentCorp" souhaite lancer **CloudMemo**, une application de prise de notes. Pour des raisons de coûts et de sécurité, plusieurs équipes (`team-blue` et `team-green`) doivent partager le même cluster Kubernetes, mais leurs environnements doivent être **strictement isolés**.

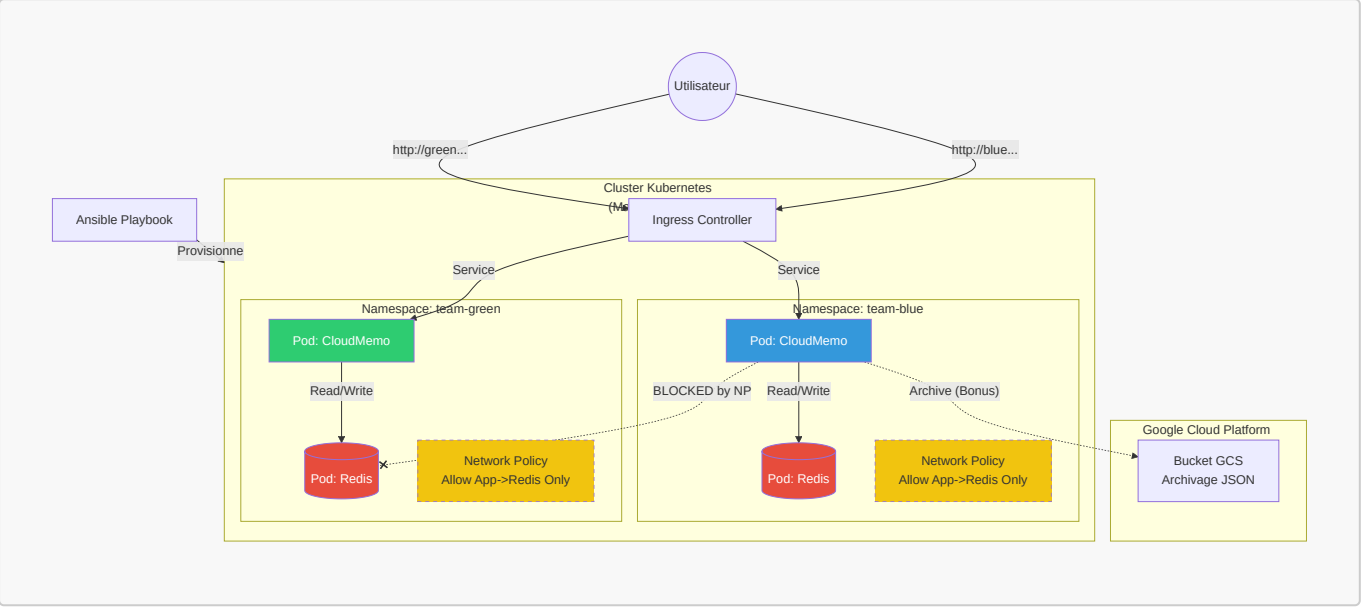
Prérequis

Avant de commencer, assurez-vous de respecter les points suivants :

- **Versionnage (Git)** : L'ensemble de votre code (Ansible, Docker, Terraform, Manifestes K8s) doit être versionné sur un dépôt Git (GitHub ou GitLab). Un historique de commits clair (Conventionnal Commits) est attendu.
- **Documentation** : Votre dépôt doit contenir un fichier `README.md` détaillant :
  - La procédure de déploiement de A à Z.
  - Les variables de configuration nécessaires.
  - Comment valider que l'installation est fonctionnelle.

Architecture Cible

- **Application** : Python/Flask (Stateless).
- **Données** : Redis (Stateful).
- **Infrastructure** : Cluster Kubernetes (Kubeadm).
- **Sécurité** : Isolation réseau (Network Policies).
- **Automation** : Ansible & Docker.



Étape 1 : Infrastructure & Automation (Ansible)

Objectif : Ne plus jamais installer un cluster à la main.

Vous disposez de 2 Machines Virtuelles (VMs) vierges (Ubuntu).

- VM 1 : `master`
- VM 2 : `worker`

Votre mission : Écrire un **Playbook Ansible** (et son inventaire) qui configure entièrement le cluster.

Le playbook doit effectuer les actions suivantes sur **tous** les nœuds :

1. Désactiver le Swap.
2. Installer les pré-requis (Modules Kernel, Sysctl).
3. Installer le Container Runtime (**Containerd**).
4. Installer les binaires Kubernetes (`kubeadm`, `kubelet`, `kubectl`).

Ensuite, spécifiquement :

1. Sur le **Master** : Initialiser le cluster (`kubeadm init`) et installer le plugin réseau (Calico).
2. Sur le **Worker** : Rejoindre le cluster (`kubeadm join`) automatiquement.

Étape 2 : Conteneurisation (Docker)

Le code source de l'application vous est fourni dans le dossier `app/`.

1. Creer le fichier Dockerfile avec le code de l'application.
2. Construisez l'image Docker localement.
3. Testez le fonctionnement avec `docker-compose up`.
4. Poussez votre image sur le Google Artifact Registry (ou Docker Hub).

Étape 3 : Déploiement Multi-Tenants (Kubernetes)

Nous allons simuler deux environnements d'équipe isolés.

1. Créez deux Namespaces : `team-blue` et `team-green`.
2. Dans **chaque** namespace, déployez la stack complète :
    - **Redis** :
        - `StatefulSet` (image `redis:alpine`).
        - `VolumeClaimTemplate` pour la persistance (`/data`).
        - `Service` Headless (ClusterIP: None).
    - **App CloudMemo** :
        - `Deployment` utilisant votre image.
        - Variable d'env `REDIS_HOST` pointant vers le service Redis local.
        - `Service` ClusterIP pour exposer l'app.
    - **Ingress** :
        - Exposez `team-blue` sur `blue.<votre-prenom>.tp.kaas-mvp-ts.fr`.
        - Exposez `team-green` sur `green.<votre-prenom>.tp.kaas-mvp-ts.fr`.

Validation : Vérifiez que vous pouvez poster des mémos sur les deux URLs indépendamment. Les données de Blue ne doivent pas apparaître sur Green.

Étape 4 : Sécurité Réseau (Network Policies)

C'est la partie critique. Par défaut, Kubernetes autorise tout le trafic entre les namespaces. **Scénario d'attaque :** Un développeur de la `team-blue` configure son application pour se connecter au Redis de la `team-green` (ex: `REDIS_HOST=redis-svc.team-green.svc.cluster.local`). Il ne doit pas pouvoir voler les données !

Mission : Mettez en place des **Network Policies** dans chaque namespace pour :

1. **Refuser tout trafic entrant** (Default Deny) par défaut.
2. Autoriser **uniquement** :
    - L'Ingress Controller à parler à l'Application (Port 5000).
    - L'Application à parler à Redis (Port 6379) **dans le même namespace uniquement**.

Validation du hack :

1. Changez la conf de l'app Blue pour viser le Redis Green.
2. Redémarrez le pod Blue.
3. L'application Blue doit planter (CrashLoopBackOff ou Timeout) car elle n'arrive pas à joindre le Redis Green.

Bonus : Cloud & Archivage (GCP & Terraform)

Pour les plus rapides, activez la fonctionnalité d'archivage vers le Cloud.

1. **Terraform** :
    - Créez un Bucket GCS unique.
    - Créez un Service Account avec droits d'écriture limités à ce bucket.
    - Récupérez la clé JSON.
2. **Kubernetes** :
    - Créez un Secret `gcp-auth` avec la clé JSON dans `team-blue`.
    - Mettez à jour le Deployment Blue pour monter le secret et définir les variables `BUCKET_NAME` et `GOOGLE_APPLICATION_CREDENTIALS`.
3. **Test** : L'équipe Blue peut maintenant archiver ses notes dans le Cloud !