TP Kubernetes - Déploiement d'un Serveur Minecraft

Date : April 23, 2025

Objectifs du TP : Mise en place d'un serveur Minecraft sous Java, sur la dernière version du jeu, avec un système de white et de black list. Assurez vous de toujours faire attention aux bonne pratiques.

Attendus en fin de TP:

- Mettre en place un cluster Kubernetes (Minikube)
- Déployer le serveur Minecraft via Helm
- Intégrer un service mesh
- Superviser avec Prometheus ou Grafana
- Automatiser le tout

Modalités de rendu :

Le TP est à rendre sous forme d'un repo **GitHub**. Il devra contenir:

- Tous les fichiers, bien structurés et organisés
- Un **README.md**, accompagnant le travail & expliquant les démarches suivies
- Lien vers le dépôt Git

Bonus : Ajout d'un logging centralisé, dashboards personnalisés ou monitoring avancé.

Étapes du TP

1. Initialisation du Cluster

Initialiser le cluster avec minikube.

2. Déploiement avec Helm

Utiliser ou créer un chart Helm pour Minecraft. Dans le fichier `values.yaml`, on configure la mémoire, la whitelist/blacklist et accepte l'EULA. Les mots de passe sensibles sont gérés via `kubectl create secret`.

Exemple de ressources :

```
resources:
  requests:
    cpu: "500m"
    memory: "1Gi"
  limits:
    cpu: "1"
    memory: "2Gi"
```

Exemple de probes :

```
livenessProbe:
  tcpSocket:
    port: 25565
readinessProbe:
  tcpSocket:
    port: 25565
```

3. Tableau de bord Prometheus et Grafana

Ajouter un exporter compatible Minecraft comme `mc-monitor`, et créer un dashboard Grafana affichant CPU, RAM, joueurs connectés, latence, ...

4. Service Mesh avec Istio ou Linkerd

Créer une Gateway et un VirtualService pour exposer le port 25565, et tester un canary deployment en redirigeant 10% du trafic vers une nouvelle version.

5. GitOps avec ArgoCD ou Flux

L'outil GitOps synchronisera un dépôt contenant le chart Helm, les manifests Kubernetes et les configurations personnalisées.

6. CI/CD avec Jenkins ou Gitlab CI

Le pipeline automatisera les déploiements via :

- Modifications dans les fichiers Helm
- Push vers le dépôt Git
- Mise à jour automatique par ArgoCD

Pour aller plus loin (si tu te fais chier, mm moi j'y comprend rien mdr)

Pour ceux qui souhaitent aller plus loin dans la réalisation de ce TP, voilci une liste de tâches. (j'espère que c'est pas trop de la merde mdr)

Personnalisation avancée du chart Helm et logging : Créer un chart Helm modulaire avec des valeurs conditionnelles, des sous-charts, et la possibilité d'activer ou désactiver certains composants comme le monitoring ou l'accès RCON. Rendre ce chart adaptable à différents types d'environnements (dev, prod...). Ajout d'une solution de logging, de la façon que tu souhaites.

Gestion avancée de la sécurité : Implémenter des *Network Policies* pour restreindre la communication entre pods et namespaces. Ajouter des contrôles d'admission comme **OPA/Gatekeeper** pour imposer des règles de déploiement strictes. Utiliser des outils comme *Trivy* pour scanner les images déployées, et configurer des secrets chiffrés avec *SealedSecrets*.

Mise en place d'un Ingress Controller + TLS : Déployer un *Ingress Controller* (comme NGINX ou Traefik) pour exposer proprement le serveur ou les dashboards. Sécuriser l'accès via HTTPS avec des certificats générés automatiquement grâce à *cert-manager*.

Système d'alertes: Configurer des alertes Prometheus basées sur des seuils critiques (CPU, RAM, indisponibilité du pod, etc.) et les envoyer sur un canal Discord, Slack ou par e-mail.

Création d'un Operator personnalisé :

Développer un petit *Kubernetes Operator* en Python pour gérer automatiquement des actions spécifiques : redémarrage programmé, sauvegarde, mise à jour conditionnelle... Cet Operator permet d'aller plus loin dans l'automatisation au sein du cluster.

Pour aller encore plus loin (si tu te fais vrm vrm chier mdr)

Serveur multi-monde : Déployer plusieurs instances Minecraft connectées via un proxy BungeeCord ou Waterfall. Cela permet de créer un cluster de serveurs Minecraft interconnectés (par exemple : survie, créatif, nether séparés), tout en orchestrant chaque monde via un chart Helm dédié.

< 3