Rapport TME6 – TME7 – TME8
Déploiement d'une Application Web Scalable

Taha Yacine GUEZZEN — M1 Réseaux
Sorbonne Université

24 avril 2025

1 Introduction

Ce rapport présente les étapes de déploiement d'une application web composée de trois services principaux :

— Une base de données Redis
— Un backend Node.js interagissant avec Redis
— Un frontend React communiquant avec le backend

L'objectif est de construire, déployer, scaler et monitorer cette infrastructure en trois étapes : TME6 (local avec Podman), TME7 (Kubernetes/Minikube), et TME8 (scalabilité, réplication, monitoring).

2 TME6 – Tests Locaux avec Podman

Objectif

Tester localement les services à l'aide de conteneurs Podman connectés à un même réseau.

Étapes réalisées

— Création d'un réseau `redisnet`
— Construction des images :
  — `node-redis-server` pour le backend
  — `redis-frontend` pour le frontend
— Lancement des conteneurs :

```
podman network create redisnet
podman run -d --name redis --network redisnet -p 6379:6379 redis
podman run -d --name node-server --network redisnet -p 8080:8080 node-redis-
podman run -d --name frontend --network redisnet -p 5400:80 redis-frontend
```

— Vérification via navigateur et inspection des logs

Problèmes rencontrés

— `host.containers.internal` inaccessible avec Podman : résolu en utilisant le réseau commun explicite.

3 TME7 – Déploiement Kubernetes sur Minikube

Objectif

Déployer l'application sur un cluster Kubernetes Minikube avec les fichiers YAML correspondants.

Étapes réalisées

— Démarrage de Minikube :

```
minikube start --driver=podman
```

— Chargement des images :

```
minikube image load node-redis-server
minikube image load redis-frontend
```

— Application des fichiers YAML :
  — `redis-deployment.yaml` et `redis-service.yaml`
  — `node-deployment.yaml` et `node-service.yaml`
  — `frontend-deployment.yaml` et `frontend-service.yaml`

— Accès via :

```
minikube service frontend-service
```

Résultat

L'interface React fonctionne, communique avec Node.js et permet l'ajout et la récupération de données Redis.

4 TME8 – Réplication, Autoscaling et Monitoring

Objectif

Rendre l'infrastructure scalable et observable via Prometheus et Grafana.

1. Réplication Redis

— Déploiement d'un master Redis
— Déploiement de pods Redis replicas avec configuration `replicaof`
— Utilisation d'un `StatefulSet` et d'un `Headless Service`

2. Autoscaling

— Utilisation d'un `HorizontalPodAutoscaler` sur `node-server`
— Scalabilité de 2 à 5 pods selon l'utilisation CPU

3. Monitoring

— Déploiement de Prometheus avec configuration `prometheus-config.yaml`
— Déploiement de Grafana
— Exposition de l'endpoint `/metrics` sur Node.js
— Dashboard personnalisé avec taux de requêtes, métriques Redis, CPU pods, etc.

5 Conclusion

Ce projet a permis de construire une infrastructure cloud native complète. Le passage de la virtualisation locale à Kubernetes a facilité la scalabilité et la gestion dynamique des composants. L'ajout de monitoring garantit la visibilité et le bon fonctionnement de l'ensemble du système.