Exécutez le script avec la commande :

bash ./deploy.sh

TME 6
BASE DE DONNEE REDIS
-docker pull redis
-docker run -d --name … redis

SERVEUR
telecharger le dossier git du TME6 pour commencer le coté serveur
git clone URL dossier server nodejs
se mettre sur le repertoir redis-note pour faire le dockerfile
creer le dockerfile
"docker build -t "nom_image que je veux donner" ."
creer le fichier docker-compose.yaml
docker compose up
pour verifier, aller sur le local host sur le navigateur !

FRONT-END
git clone URL gti react
aller sur le src-conf.js et mettre port 3000
creer le dockerfile (identique Yasmine)
docker build -t react .
docker run --name react-container -p 8080:80 react

TME 7 - Kubernetes
installer la derniere version de minikube
puis :
minikube start --driver=docker -force
(verifier avec minikube Status)
minikube image load node-redis-server
minikube image load react
minikube image load redis

nano tous les fichiers deployment + service pour nos 3 parties (front,serveur & bdd)
Puis (pour deployer) :
kubectl apply -f redis-service.yml -f redis-deployment.yml -f redis-hpa.yml
kubectl apply -f node-redis-service.yml -f node-redis-deployment.yml -f node-redis-hpa.yml
kubectl apply -f react-service.yml -f react-deployment.yml -f react-hpa.yml
kubectl apply -f prometheus-config.yml -f prometheus-deployment.yml -f
prometheus-service.yml
vérifier s'ils sont bien crées :
kubectl get service

kubectl get pods
kubectl get hpa (pour l'auto scaling)

Ensuite, pour avoir l'URL et vérifier tout ça :
minikube service node-redis-front-service --url
PS : pour bien etre certain du nom "node-redis-front-service" de notre service : minikube service list

ATTENTION :
Pour que ca fonctionne bien, il faut que le tunnel minikube soit activé (commande minikube tunnel)
Etapes pour lancer :
-sur un terminal, lancer minikube tunnel
-sur un autre terminal, commande minikube service node-redis-front-service --url

Pod = Conteneur(s) qui exécutent une application.

Service = Adresse fixe qui permet de communiquer avec les Pods.

Volume = Stockage persistant utilisé par un Pod.

---------- Partie Monitoring
Prometheus a besoin d'un fichier de configuration prometheus.yml pour savoir où récupérer les métriques.
On va stocker ce fichier dans un ConfigMap

Appliquer le ConfigMap dans Kubernetes :
kubectl apply -f prometheus-config.yaml

Après avoir créer les fichier deployment et service pour Prometheus: Déployer Prometheus :
kubectl apply -f prometheus-deployment.yaml
kubectl apply -f prometheus-service.yaml
On verifie avec kubectl get pods
Puis : minikube service prometheus --url

Pareil pour graphana :
kubectl apply -f grafana-deployment.yaml
kubectl apply -f grafana-service.yaml
On verifie avec kubectl get pods
Puis : minikube service grafana --url

Avoir activé :

minikube addons enable metrics-server

vérifier que Prometheus scrappe bien ces données :
http://localhost:9090/targets
Regarder si `nodejs-backend` est marqué **UP**