DEVOPS

TP 2

Partie 1

Sur votre ordinateur

1°_ Grâce au Dockerfile déjà présent, faite un build d'une image docker basée sur ce Dockerfile.

2°_ Faites un run de cette image pour lancer un container sur le port 4000 de votre machine et visualisez le résultat sur http://localhost:4000/.

3°_ Créez un repo github et pushez le code dessus. Visualisez les githubs actions. Modifiez ensuite le fichier index.html, committez, pushez et confirmez que les githubs actions s'activent.

Rajoutez un job qui affiche « Good By » dans hello-world.yml, commitez, pushez et confirmez que vous visualisez ce nouveau job.

4°_ Vous avez précédemment créé un image docker. Pushez là sur docker hub.

Partie 2

Sur votre VPS

1°_ Installez putty et connectez vous à votre VPS via putty.

Installez docker

Confirmez bien l'installation de docker via docker -v

2°_ faites un pull de votre image sur docker hub

Faites un run de cette image sur le port 4000 de votre serveur

Visualisez votre page web sur http://<IP_SERVEUR>:4000

3°_ Supprimez le container via ‘docker stop <ID_CONTAINER>’ puis ‘docker rm <ID_CONTAINER>’

Faites un run de cette image sur le port 80 de votre serveur

4°_ dans Gandi, grâce au compte et à l’url fourni par le professeur, configurez l’enregistrement DNS A pour pointer vers l’IP de votre serveur.

Attendez quelques minutes et confirmez que votre url amène à votre page web

Partie 3

1°_ Créez un nouveau fichier CI-CD prod.yml qui s’active sur un push d’une branche « prod ». Pour l’instant celui-ci affiche toujours un message simple « hello prod »

Créez votre branche « prod » et confirmez que le fichier prod.yml est activé sur un push

2°_ modifie prod.yml et crée un job push-image qui build une nouvelle image docker et push la sur ton espace docker-hub

3°_ rajoute à prod.yml un nouveau job qui se connecte à ton VPS, stop ton container, pull la nouvelle image docker, run un container avec la nouvelle version.

Confirme qu’à chaque push sur la branche prod, la nouvelle version de ton appli est déployée.

Parte 4 :

1°_ sur ton VPS, stop et remove ton container.

2°_ modifie la ligne 14 du fichier docker-compose.yml pour y intégrer l’adresse sur dockerhub de ton image.

3°_ Confirme que docker-compose est installé sur ton VPS via ‘docker compose version’. Si ce n’est pas le cas, installe le.

execute ‘sudo mkdir -p /opt/monitoring/prometheus /opt/monitoring/grafana /opt/monitoring/nginx ‘ sur ton VPS, cette commande crée 3 dossiers monitoring/grafana, monitoring/nginx, monitoring/prometheus.

Execute ‘cd /opt/monitoring’ pour te placer dans le dossier monitoring. Dans ce dossier crée le fichier docker-compose.yml via la commande ‘nano docker-compose.yml’ et colle ton fichier docker-compose.yml. Tu peux enregistrer ton ficher avec Ctrl + O puis entrée et quitter le fichier via Ctrl + Q.

De la même façon crée le fichier prometheus.yml dans le dossier prometheus et insére le contenu de ton fichier.

Place toi sur cd /opt/monitoring, au même niveau que docker-compose.yml et execute la commande ‘docker compose up -d’.

Tu pourras alors accéder à

- Prometheus via http://<IP_VPS>:9090
- Grafana via http://<IP_VPS>:3000 (à la première connexion ton username est « admin » et ton mdp « admin »
- cAdvisor via http://<IP_VPS>:8082

Partie 5 :

1°_ éteint tout avec la commande ‘docker compose stop’. Confirme avec ‘docker compose ps’ et en essayant d’acceder à l’url.

2°_ remplace le docker-compose.yml par celui dans le dossier « partie5 »

2°_ remplace le prometheus.yml par celui dans le dossier « partie5 »

3°_ crée le fichier nginx/nginx.conf et colle la contenu du dossier « partie5 »

4°_ execute la commande docker-compose up -d

4°_ Qu’est ce que ça a changé ?

Partie 6 :

Modifie prod.yml pour activer le déploiement automatique sur un push.

Créez des graphiques sur grafana