CYBERSUP

DOCKER

Embarquez pour un voyage dans l'univers Docker où même les conteneurs ont le sens de l'humour !

Introduction du sujet

Prêts à découvrir comment ces petites boîtes virtuelles rendent le déploiement d'applications aussi simple que de dire 'container' ?

Docker est un outil qui permet de créer des "conteneurs" isolés contenant tout ce dont une application a besoin pour fonctionner. Ces conteneurs sont portables et peuvent être utilisés dans divers domaines :

1. **Développement d'applications :** fournit un environnement de développement cohérent et portable.
2. **Déploiement d'applications :** facilite le déploiement sur différentes plateformes sans se soucier des différences d'environnement.
3. **Microservices :** Aide à la gestion et à l'évolutivité des services en les isolant dans des conteneurs distincts.
4. **Tests et CI/CD :** Utilisé pour exécuter des tests automatisés et pour implémenter des pipelines CI/CD, assurant une intégration continue et une livraison continue.

Rendu

Le projet est à rendre sur : https://github.com/prenom-nom/docker, 1 fichier Doc par Job avec les commandes, scripts utilisés. Pensez à mettre votre repository en public.

L'évaluation se fera sous forme de présentation sans support à l'équipe pédagogique.

Base de connaissances

- Documentation Docker
- Forum Docker Community
- Docker Captain's Blog
- Portainer
- Docker Volumes

Job01

Créer une VM debian en mode console 8Go / 1Go / 1 cpu, installer docker (cli seulement).

Job02

Tester l'installation de docker avec le conteneur « helloworld » et se familiariser avec les commandes.

Job03

Utilisation de « Dockerfile » pour recréer le conteneur « helloworld » depuis une image Debian minimum.

Job04

Utilisation de Dockerfile pour créer une image ssh (compte : root et mot de passe : root123) sans utiliser une image ssh existante , voir redirection des ports (utiliser un autre port que le 22) , créer et lancer le conteneur et se connecter pour vérifier l'accès SSH.

Job05

Tout informaticien étant «flemard» , il faut faire des alias pour les commandes docker en cli , à mettre dans ~/.bashrc pour manipuler les images / containers.

Job06

Se renseigner sur l'utilisation de Volumes entre 2 conteneurs , et la gestion des volumes .

Job07

A l'aide d'un fichier yml , de docker-compose faire 2 conteneurs : nginx et ftp liés entre eux. Création d'un volume commun pour accéder au dossier web .
Créer sur votre pc un fichier index.html , dans ce fichier faites afficher votre nom/prénom). Installer Filezilla sur votre PC , se connecter en FTP sur le conteneur ftp pour envoyer le fichier index.html , et regarder le résultat.

Job08

Sans utiliser une image nginx existante , utilisation de «Dockerfile» pour créer un conteneur nginx , voir redirection des ports , et se connecter .

Job09

Création et utilisation d'un "registry" local. Et ajouter une «UI» pour le gérer depuis une interface web .

Job10

Faire un script bash , pour effacer totalement docker (les images , volumes , conteneurs , et paquets correspondants) et rendre le système propre .

Et aussi un script pour automatiser l'installation de docker sur une machine Debian .

Job11

Découverte de **portainer** .

Refaire les Job 2 à 9 , en utilisant l'interface graphique de portainer . Se renseigner sur les alternatives à Portainer

Pour aller plus loin

On complique un peu le système , il va falloir faire l'équivalent de XAMPP

Avec un fichier Dockerfile et un fichier yml faire :

- un serveur nginx(ou apache) avec php .
- un serveur mariadb/mysql (avec phpmyadmin) .
- un serveur FTP .
- un volume pour stocker l'ensemble des données communes aux différents serveurs .

Tester le système .

Compétences visées

- virtualisation (OS-level virtualization)
- Administration système
- Script