UNIVERSITÉ CAEN NORMANDIE

Réseaux et stockage avec Docker

Virtualisation - TD

IUT GRAND OUEST NORMANDIE

Maxime Lambert

2023 / 2024

Modalités

- Travail en binôme en salle de machines.

- Matériel ou logiciels utilisés **:**
  - Accès au serveur Proxmox de l'université (https://proxmox-iutc3.unicaen.fr) depuis un navigateur web

  - Repartez de la VM que vous avez créée lors du TP n°3 (**vm-docker-*<<mes initiales>>***). Si vous n'avez pas encore suivi ce TP, ouvrez le et suivez les instructions suivantes :
    - 2.1 Initialisation d'une nouvelle VM
    - 2.3 Les étapes post-installation
      - **Puis redémarrer votre VM (`sudo reboot`)**

Avant-propos

- En TP, vous avez appris ou allez apprendre à utiliser la *CLI* de Docker pour instancier des conteneurs à partir d'images existantes. Vous avez également écrit vos premiers Dockerfiles afin de générer vos propres images.

- Il a été précisé en CM que:
  - Les conteneurs doivent plutôt être considérés comme des objets éphémères.

  - Les conteneurs doivent être, si possible, pensés pour ne contenir qu'un service.

- Il existe un réel besoin de persister des données. Il n'est pas envisageable de perdre le contenu d'une base de données lors de la suppression d'un conteneur... Nous verrons en quoi les volumes nous aideront à résoudre cette problématique.

- Une application est généralement constituée de plusieurs services. Etant donné que les conteneurs ne devraient en contenir qu'un, il y a nécessité de coopération entre eux. Cela est communément géré via le réseau.

Prérequis

- Nous n'avons pas besoin du proxy lors de ce TD. Ce dernier risque de vous bloquer si la variable d'environnement `no_proxy` ne contient pas l'adresse de vos sous-réseaux docker.

- Si vous avez déjà réalisé le TP n°3, sur votre machine hôte, merci d'effectuer la commande :
  - `mv ~/.docker/config.json ~/.docker/config.json_ignore`
  - De cette manière aucun proxy ne sera utilisé par vos conteneurs mais vous pourrez facilement rétablir la configuration au besoin.

1. Le réseau par défaut

- Consultez les réseaux disponibles dès l'installation de docker grâce à la commande `docker network ls`
  - Il en existe 3, indiquez leur fonctionnement et leur cas d'usage

- Instanciez deux nouveaux conteneurs détachés, mais avec la possibilité d'interagir avec leur shell:
  - `docker container run -dit --name c1 prof:5000/busybox sh`
  - `docker container run -dit --name c2 prof:5000/busybox sh`

- En inspectant les conteneurs et les réseaux, identifier le réseau par défaut.
  - Appuyez-vous sur la commande `docker inspect <conteneur ou réseau>`

- Réalisez un diagramme de réseau logique dans lequel vous ferez figurer: les 2 conteneurs, l'hôte, le sous-réseau proxmox auquel est connectée la VM, le sous-réseau par défaut docker, les passerelles.

- Depuis un conteneur, vérifiez que vous arrivez à joindre la VM "prof" avec l'ip 172.17.249.47 et rajoutez cette VM sur votre diagramme réseau.
  - Vous pouvez "attacher" le shell du conteneur à votre terminal de cette manière : `docker attach c1`
  - Quitter le shell du conteneur avec la commande: `exit`
  - Pour lister les conteneurs : `docker container ls -a`
  - Pour démarrer vos conteneurs à l'état stoppé: `docker start c1`
  - Exemple pour joindre la VM "prof": `ping -c 2 172.17.249.47`

- Depuis un conteneur, essayez de faire un ping vers l'autre conteneur depuis son adresse IP, puis son nom. Quel constat ?

2. Création d'un réseau "bridge"

- Instanciez un réseau avec le driver `bridge`. Nommez le `td3-net`
  - Exemple: `docker network create --driver bridge XXX`

- Instanciez deux nouveaux conteneurs que vous connecterez à votre nouveau réseau et nommerez `c3`, `c4`
  - Utilisez l'option `--network XXX`. Ne pas oublier les 3 options `-dit`

- Complétez votre diagramme de réseau logique dans lequel vous ferez figurer: les 2 nouveaux conteneurs et le sous-réseau `td3-net`

- Depuis le conteneur `c3`, essayez de contacter le conteneur `c4` par son ip, puis par son nom. Le résultat est-il différent de l'étape 1 ? Pourquoi ?

- Depuis le conteneur c3, essayez de contacter le conteneur c1 ou c2 par son ip, puis par son nom. Quel résultat obtenez-vous ? Pourquoi ?

Pour répondre à ces questions, aidez-vous des commandes: `docker inspect, ip a, ip route, sudo iptables -S`

3. Utilisation du réseau “host"

- Pour optimiser les performances ou par simplicité, il est possible de supprimer l’isolation au niveau réseau. C'est-à-dire, partager l’espace de noms de l’hôte avec des conteneurs.

- Instanciez un serveur web nginx en utilisant le réseau host.
  - Exemple: `docker container run -d --network host prof:5000/nginx`

- Sur la machine hôte, ouvrez votre navigateur sur l’adresse de `loopback` en utilisant le port `80`. Vous devriez voir une page d’accueil “Welcome to nginx!”

- Quels inconvénients à l’utilisation de ce driver ?

Gérer le stockage en dehors du conteneur

- Par défaut, tous les fichiers créés et modifiés dans un conteneur sont stockés sur la couche conteneur.

- Nous avons vu en CM qu'il est possible d'écrire les modifications de nos fichiers ailleurs en utilisant un montage lié, un volume ou point de montage en mémoire.

- La syntaxe la plus verbeuse mais la plus simple consiste à utiliser l'option `--mount` qui va contenir un ensemble de clés/valeurs.
    - La documentation détaillée: https://docs.docker.com/storage/volumes/#choose-the--v-or---mount-flag

4. Utilisation du montage lié (*bind mount*)

- Sur la machine hôte, créez un répertoire à cet emplacement `/home/user/td3`, et y placer quelques fichiers vierges.

- Instancier un conteneur en utilisant:
  - Un montage lié du répertoire `td3` vers le chemin `/td3` dans le conteneur.
  - L'image `prof:5000/busybox`
  - La commande `sh`

- Depuis ce conteneur, modifiez quelques fichiers du répertoire `/td3`. Revenez sur l'hôte, que constatez-vous dans le répertoire `/home/user/td3` ?

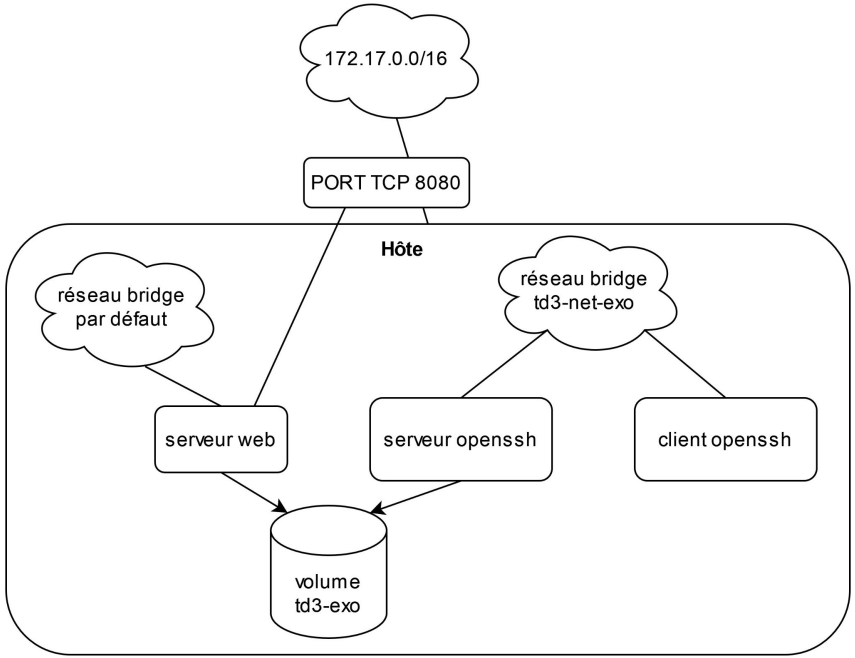
- Avez-vous des recommandations quant à l'usage du montage lié ? Que se passerait-il si vous montiez la racine du système de fichiers hôte dans un conteneur et que vous lanciez une commande `rm -rf /*` (ne le faites pas…)

5. Utilisation conjointe des volumes et réseaux 1/2

- L'objectif de cet exercice est de faire coopérer plusieurs conteneurs entre eux en utilisant les réseaux et les volumes. Il vous est demandé de reproduire l'infrastructure ci-contre.

- Vous exposerez sur le port TCP 8080 de la machine hôte, un serveur web que vous pourrez joindre depuis votre navigateur.

- Le serveur web:
  - Utilise l'image `prof:5000/nginx:latest`
  - Est connecté au réseau bridge par défaut
  - Possède le contenu de son site web dans un volume `td3-exo`
- Le serveur openssh:
  - Utilise l'image `prof:5000/ubuntu:server-ssh`
  - Est connecté au réseau bridge `td3-net-exo`
  - Permettra d'écrire le contenu du site dans le volume `td3-exo`
- Le client ssh:
  - Utilise l'image `prof:5000/ubuntu:client-ssh`
  - Est connecté au réseau bridge `td3-net-exo`
  - Sera votre point d'accès pour modifier le contenu du site.



5. Utilisation conjointe des volumes et réseaux 2/2

- Une fois votre infrastructure en place, suivez le mode opératoire suivant dans un terminal de la machine hôte:
  - `docker attach <id ou nom conteneur : client-ssh>`
  - `ssh user@<ip ou nom du conteur: server-ssh>`
    - Le mot de passe est **`pass`**
  - Passez root avec la commande sudo su
    - Utiliser le mot de passe **`pass`**
  - Placez-vous dans le répertoire de montage du volume
  - Rajoutez quelque chose au fichier `index.html`
    - exemple: `echo "Hello from td3" >> index.html`
  - Ouvrez le navigateur sur la machine hôte et constatez la modification des fichiers de votre serveur web

- Conseils et rappels:
  - Pour mapper le port TCP d'un conteneur vers la machine hôte, utiliser l'option `-p src:dest`
  - Pour construire tous vos conteneurs et exécuter le mode opératoire à la fin, utiliser les options `-dit`
  - Pour attacher `stdin`, `stdout` et `stderr` d'un conteneur actif, on utilise la commande `docker attach <id ou nom conteneur>`
  - Pour connaître la destination du volume dans le conteneur nginx, consultez la documentation de l'image sur Docker Hub

- *Remarque: l'intérêt de cet exercice est de vous faire manipuler les réseaux et les volumes. Dans la vraie vie, on évite des infrastructures aussi compliquées pour adresser un besoin aussi simple. Typiquement, l'utilisation d'un montage lié en readonly depuis la machine hôte est une solution raisonnable pour exposer des fichiers depuis notre machine hôte via un serveur web.*

Pour aller plus loin

- Il vous reste un peu de temps ?

- Vous pouvez découvrir le driver “macvlan” permettant d’attribuer une adresse mac à des conteneurs. Cela permet de faire passer le conteneur sur le réseau comme étant une machine à part entière.

- Ci-joint un lien vers un tutoriel issue de la documentation de docker: https://docs.docker.com/network/network-tutorial-macvlan/

Référence

La partie réseau de ce TD a été inspirée par les tutoriels de Docker :

Bridge network tutorial [en ligne].[Consulté le 29 janvier 2023]. Disponible à l'adresse : https://docs.docker.com/network/network-tutorial-standalone/

Merci de votre attention

Maxime Lambert

maxime.lambert@unicaen.fr