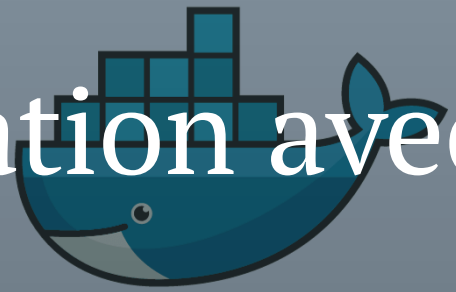
Communication avec l’extérieur



Retour sur ce qui a été vu

- Docker : API + commande pour contrôler :
  - le cycle de vie de conteneurs (docker container run/stop/create/rm/ps/pause/attach)
  - la gestion d'une bibliothèque d'images (docker image ls/pull/rm/history)
  - la création d'images (Dockerfile & docker build)
- Dockerfile :
  - Chaque instruction décrit une couche d'une image
  - FROM / RUN / COPY / ENTRYPOINT + CMD

Que nous manque-t-il ?

- Être isolé MAIS communiquer avec l'extérieur :
  - Accéder aux fichiers de l'hote depuis un conteneur (plutôt pour les développeurs)
  - Persistance des données (plutôt pour la mise en production)
  - Exposer un/des port.s de l'hôte pour accéder depuis l'extérieur à notre conteneur
  - Fournir des valeurs aux variables d'environnement du conteneurs

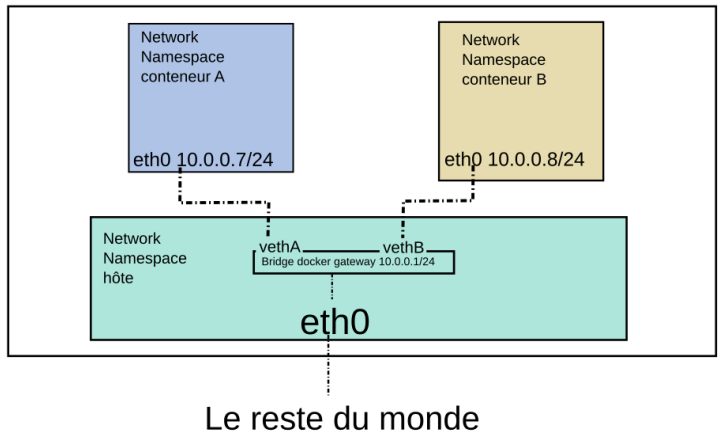
Bind mount

- Bind-mount des fichiers (pour un développeur)
  - docker run -it -v chemin_hote:chemin_conteneur ubuntu bash
  - Attention : le chemin hote ne peut pas être donné comme un chemin relatif … pourquoi ??

Volumes

- docker volume create NOMVOLUME
- docker run -v NOMVOLUME:/opt/
- Un volume peut s’appuyer sur n’importe quoi (nfs / sshfs / s3 / ceph / …)

Communication réseau



- SNAT par défaut
- DNAT avec l'option -p Port_Hote:Port_Conteneur

Exemple

- Dans le dossier demo_SNAT:
  - Dockerfile
  - src/ contient un serveur web en flask qui affiche les fichiers du répertoire spécifié dans la route
- Build : `docker build . -t demo_SNAT`
- Ceci ne sert à rien : `docker run demo_SNAT`
- SNAT de localhost:5050 vers conteneur:5000 : `docker run -p 5050:5000 demo_SNAT`
- Avec montage de fichiers : `docker run -p 5050:5000 -v $(pwd):/opt/ -demo_SNAT`
- Et pour développer : `docker run -p 5050:5000 -v $(pwd)/src/:/app/ -demo_SNAT`

Variables d’environnement

En ligne de commande :

```
docker run -e NOM=Valeur image
```

Les commandes oubliées jusqu'ici

- docker image prune
- docker container logs
- docker exec