Introduction à Docker

Jean Pommier

27 juillet 2020

Un peu d'histoire

- Serveurs *old school* : des grosses machines dédiées à une appli
- Machines virtuelles (VMs) : VMWare, virtualbox, Xen, etc
- Cloud & Conteneurs : souple, résilient, jetable

Un peu d'histoire

Serveurs *old school*

- une grosse machine, plusieurs applis se partagent le système
- un métier : administrateur système

  - Optimisation des ressources
  - Évolutivité : *scalability*
  - Durée de mise en service

Un peu d'histoire

Serveurs *old school*

Points de vue

- hébergeurs : un gaspillage de ressources et un frein au développement
- sysadmins : un bébé à couver. Pb de suivi de l'état des serveurs.
- développeurs : cycle de vie des applis très long. Pas la main sur le déploiement. Temps de déploiement typique : plusieurs mois.

Un peu d'histoire

VMs

- Émulation matérielle
- Une machine : plusieurs serveurs virtualisés

\+ Optimisation des ressources

\+ Créer une nouvelle machine est assez rapide

\+\+ Sécurité : cloisonnement des environnements

\- Un OS complet dans chaque machine

\- Volumineux

\- Attachées à une machine (déplacement lent)

Un peu d'histoire

VMs

Points de vue

- hébergeurs : Une nette amélioration. VPS => optimisation des datacenter. + de flexibilité. Débuts du Cloud
- sysadmins : Pas une grosse différence. Simplifie la sauvegarde d'un état donné de la machine. Possibilité de prendre des *snapshots*
- développeurs : pas de différence majeure. Déploiements potentiellement accélérés.

Un peu d'histoire

Conteneurs

++ Optimisation des ressources

++ Créer une nouvelle machine est très rapide

\+ Sécurité : Isolation relative des environnements

++ Noyaux commun => petites images

++ Microservices

+++ Iac : Infrastructure as Code.

\- Léger surcoût (perfs) de la couche de virtualisation

\- Stockage : toujours un peu problématique

Un peu d'histoire

Conteneurs

Points de vue

- hébergeurs : Full Cloud. Vms jetables. Les serveurs deviennent "anonymes"
- sysadmins : change tout. Pas de downtime toléré. Devient "devops".
- développeurs : déploiement simplifié, workflow, devops. Contrepartie : récupère parfois la charge allouée aux sysadmins. Augmentation de la complexité. Temps de déploiement typique : qq dizaines de minutes voire moins.

Qu'est-ce que Docker ?

- Un produit de docker.com
- Open Source
- Une fonctionnalité native Linux, packagée pour la simplicité d'utilisation
- Images
  - incrémentielles (Overlay FS) -> réutilisation ! Gain de place
  - immutables
  - définies via un fichier Dockerfile => IaC
- réseau
  - facilité de définir plusieurs réseaux
  - sécurise facilement les backends

Qu'est-ce que Docker ?

Ecosystème : autour de Docker

- Docker hub (`https://hub.docker.com/`)
- Orchestrateurs : (docker-compose), Swarm, Kubernetes, k3c
- Outils de gestion de l'infrastructure : Terraform, Rancher, Portainer
- Normalisation : OCI (*Open Container Initiative*)
- Des alternatives : containerd,

Quand utiliser Docker ?

Il fait tout... sauf le café

- Serveurs
  - production (*on-premise*, Cloud)
  - developpement (facilite la reproduction d'un env. de prod en dev)
  - tester des applis serveur (ex. geOrchestra)
- local : lancer un serveur de BD, un serveur web, en une commande
- devops
  - Compiler une app (*2-stage build*)
  - Automated testing
  - Deployment

Installer Docker

Où ça devient sérieux...

- Linux : *RTFM* (`https://www.docker.com/get-started`)
- Windows
  - Docker Desktop for Windows
  - Redémarrer. Attendre (démarrage Docker très long)
  - WSL2 (Windows 10 update 2004)
  - Redémarrer. Attendre.
  - Redémarrer. Attendre. ;o)
  - WSL2 -> **docker desktop + Linux**
- Mac OS : Docker Desktop ?

Docker pratique : les mains dans le cambouis

Restons simple : un serveur nginx

- Restons traditionnels :

```
docker run hello-world
```

- Un serveur web rapide :

```
docker run -it --rm -p 82:80 nginx
```

- ... avec du contenu :

```
docker run -it --rm -p 82:80 \
-v /home/jean/www:/usr/share/nginx/html nginx
```

Docker pratique : les mains dans le cambouis

Analysons un peu

- Téléchargement des images depuis dockerhub
- `-p` : correspondance de ports
- `-v` : monter un volume
- `--rm` : détruire le conteneur à la sortie
- `docker run --help`

Docker pratique : les mains dans le cambouis

Allons un peu plus loin

- relançons notre conteneur :

```
docker run -it --rm -p 82:80 --name nginx \
-v /home/jean/www:/usr/share/nginx/html nginx
```

- on va entrer dans le conteneur. Dans une autre console, lancer

```
docker exec -it nginx /bin/bash
```

- `docker inspect`, un outil bien utile quand la documentation fait défaut

Docker pratique : les mains dans le cambouis

Bases de données facile

- on peut lancer un serveur de BD sans effort :

```
docker run -it --rm -p 3306:3306 --name mysql mariadb:10.4
```

- Quel volume monter pour persister les données ?

```
docker image inspect --format "{{{{.Config.Volumes}}}}" \
mariadb:10.4
```

- Un serveur PostgreSQL ? Facile. Quelle version vous voulez ?

```
docker run -it --rm -p 15432:5432 --name pg \
-v /tmp/pgdata:/var/lib/postgresql/data postgres:11
```

- Vous voulez aussi un PostgreSQL version 9.3 à côté ?

```
docker run -it --rm -p 15433:5432 --name pg-9.3 \
-v /tmp/pgdata-9.3:/var/lib/postgresql/data postgres:9.3
```

Docker pratique : les mains dans le cambouis

Bases de données facile (suite)

- Vous voulez du PostGIS ?

Docker pratique : les mains dans le cambouis

Lancer un serveur carto

- On peut utiliser une image fournie par la communauté

```
docker run -it --rm -p 8085:8080 --name geoserver \
-v /tmp/geoserver_datadir:/mnt/geoserver_datadir \
-v /tmp/geoserver_geodata:/mnt/geoserver_geodata \
pigeosolutions/geoserver:2.15.3
```

- On peut interconnecter la BD PostGIS et GeoServer

```
docker run -it --rm -p 8085:8080 --name geoserver \
-v /tmp/geoserver_datadir:/mnt/geoserver_datadir \
-v /tmp/geoserver_geodata:/mnt/geoserver_geodata \
--link pgis \
pigeosolutions/geoserver:2.15.3
```

docker-compose

Les bases de l'orchestration

- publier les ports sur localhost et lier les conteneurs a ses limites
- les commandes deviennent très longues
- la solution : **docker-compose**
  - tout est écrit dans un fichier texte (yml)
  - config persistée,
  - versionnable,
  - autodocumentéé
  - IaC
  - documente facilement un moyen de tester une solution
    - wp
    - georchestra
    - lizmap

docker-compose

Les bases de l'orchestration

TODO : get a working docker-compose

```
version: '3.1'
networks:
  cqpgeom_net:

volumes:
  pgdata:

services:
  nginx-proxy:
image: nginx
ports:
- "80:80"
volumes:
- ./nginx/nginx.conf:/etc/nginx/nginx.conf
- ./nginx_html:/html
networks:
cqpgeom_net:
aliases:
- nginx
restart: unless-stopped
php:
image: pigeosolutions/php:7-fpm-alpine-pgsql
volumes:
```

Dockerfile

Construire son image

- Fichier texte
- définit le contenu de l'image
- compilation : `docker build -t jeanpommier/monimage .`
- qq commandes principales :

  FROM le point de départ. On construit sur l'existant

  RUN exécute une commande (dans la VM). Permet d'installer des librairies, dézipper une archive, bouger/supprimer des fichiers etc

  COPY copie des fichiers depuis l'ordi

  ADD idem, mais permet aussi de copier depuis une URL

  Nombreuses autres instructions, cf doc dockerfile TODO get the url