

# 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 couvrir. 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

**Conteneurs** : on allège autant que possible le concept de VM : on utilise le noyau du système hôte et on n'encapsule que le strict nécessaire pour faire tourner l'application.

Un conteneur par service. Une application sera typiquement découpée en plusieurs conteneurs Docker (BD, serveur web, etc)

# 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, LXC, rkt, etc

# 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 : <https://docs.docker.com/docker-for-windows/install/>
  - ▶ 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
```

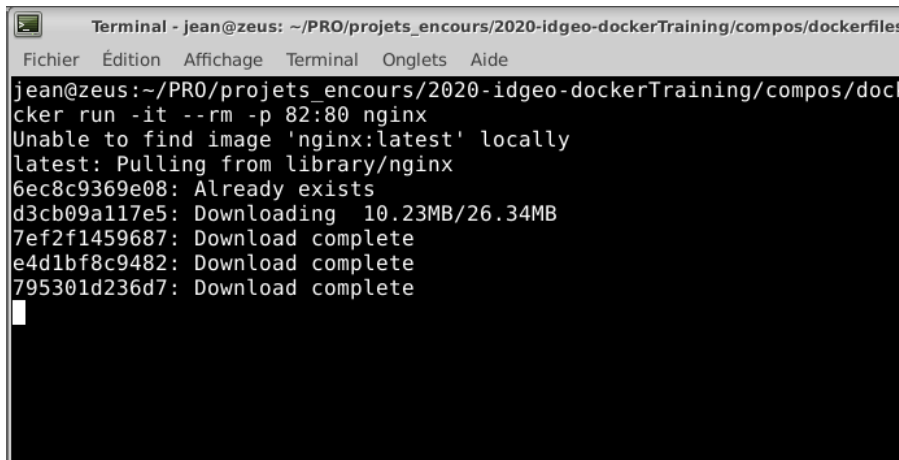
- ▶ Un serveur web rapide :

```
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



```
Terminal - jean@zeus: ~/PRO/projets_encours/2020-idgeo-dockerTraining/compos/dockerfiles
Fichier  Édition  Affichage  Terminal  Onglets  Aide

jean@zeus:~/PRO/projets_encours/2020-idgeo-dockerTraining/compos/doc
cker run -it --rm -p 82:80 nginx
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
6ec8c9369e08: Already exists
d3cb09a117e5: Downloading 10.23MB/26.34MB
7ef2f1459687: Download complete
e4d1bf8c9482: Download complete
795301d236d7: Download complete
```

# 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

## 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 PostgreSQL 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 pg \  
pigeosolutions/geoserver:2.15.3
```

# Docker pratique : les mains dans le cambouis

## Bases de données facile (suite)

Vous voulez du PostGIS ?

Il est temps de fabriquer notre propre image

# 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

<https://docs.docker.com/engine/reference/builder/>

# Dockerfile

## CMD et ENTRYPOINT

TODO : réviser les notions

**CMD** la commande lancée par le conteneur au démarrage

- ▶ format bash
- ▶ format liste. TODO : différence

**ENTRYPOINT** permet notamment de lancer des scripts d'init

Entrypoints sophistiqués : utilisation de run-parts. Exemple, PostgreSQL

# Dockerfile

## Cr  er une image PostGIS : v1

### Dockerfile

```
FROM postgres:12

RUN apt-get update && apt-get install postgis -y
# Include datastore setup scripts
COPY ./docker-entrypoint-initdb.d /docker-entrypoint-initdb.d
```

### fichier entrypoint

```
CREATE EXTENSION POSTGIS;
```

### console

```
docker build -t jeanpommier/postgis:v1 ./postgis_v1/
docker run -d -it --rm -p 5432:5432 -e POSTGRES_PASSWORD=passwd \
--name pgis jeanpommier/postgis:v1
psql -U postgres -h localhost -p 5432 -d postgres
postgres=# SELECT PostGIS_full_version();
docker stop pgis
```

# Dockerfile

## Créer une image PostGIS : v2

On suit la doc, qui pointe vers un exemple plus complet :  
<https://github.com/postgis/docker-postgis/blob/4eb614133d6aa87bfc5c952d24b7eb1f499e5c7c/12-3.0/Dockerfile>

```
FROM postgres:12

LABEL maintainer="PostGIS Project - https://postgis.net"

ENV POSTGIS_MAJOR 2.5
ENV POSTGIS_VERSION 2.5.4+dfsg-1.pgdg100+1

RUN apt-get update \
&& apt-cache showpkg postgresql-$PG_MAJOR-postgis-$POSTGIS_MAJOR \
&& apt-get install -y --no-install-recommends \
postgresql-$PG_MAJOR-postgis-$POSTGIS_MAJOR=$POSTGIS_VERSION \
postgresql-$PG_MAJOR-postgis-$POSTGIS_MAJOR-scripts=$POSTGIS_VERSION \
&& rm -rf /var/lib/apt/lists/*

RUN mkdir -p /docker-entrypoint-initdb.d
COPY ./initdb-postgis.sh /docker-entrypoint-initdb.d/10_postgis.sh
COPY ./update-postgis.sh /usr/local/bin
```



# Docker pratique : les mains dans le cambouis

## GeoServer + PostGIS

On peut interconnecter plusieurs conteneurs

```
docker run -d -it --rm -p 5432:5432 -e POSTGRES_PASSWORD=passwd \  
--name pgis jeanpommier/postgis:v1
```

On peut interconnecter la BD PostgreSQL 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**
  - ▶ assemble la compo dans un fichier texte (yml)
  - ▶ *As Code* : config persistée dans un fichier
  - ▶ permet de démarrer tous les conteneurs à la fois :  
`docker-compose up`
  - ▶ versionnable,
  - ▶ autodocumentée
  - ▶ IaC
  - ▶ documente facilement un moyen de tester une solution
    - ▶ wp
    - ▶ georchestra
    - ▶ lizmap

# docker-compose

## GeoServer + PostGIS

```
1 version: '3.1'
2 services:
3   db:
4     image: jeanpommier/postgis:v1
5     build: ./dockerfiles/postgis_v1
6     environment:
7       - POSTGRES_USER=postgres
8       - POSTGRES_PASSWORD=mysecretpassword
9     volumes:
10      - postgresql_data:/var/lib/postgresql/data
11   geoserver:
12     image: pigeosolutions/geoserver:latest
13     build: ./dockerfiles/geoserver
14     restart: always
15     ports:
16       - 8085:8080
17     volumes:
18       - geoserver_datadir:/mnt/geoserver_datadir
19       - geoserver_geodata:/mnt/geoserver_geodata
20       - geoserver_tiles:/mnt/geoserver_tiles
21
22 volumes:
23   postgresql_data:
24   geoserver_datadir:
```

# docker-compose

Inclure les infos de compilation

```
version: '3.1'
services:
  db:
    image: jeanpommier/postgis:v1
    build: ./dockerfiles/postgis_v1
    environment:
    ...

  geoserver:
    image: pigeosolutions/geoserver:latest
    build: ./dockerfiles/geoserver
    ...
```