Java EE Spring, prêt à l'emploi Docker

Guillaume Dufrêne – Lionel Seinturier – Julien Wittouck

Université de Lille

Plan

- Les principes des containers
- Les images de containers
- Les principales commandes
  - Manipuler des containers
  - Manipuler des images
- Démarrer un environnement de développement
- Construire une image applicative

Introduction

Les Containers dans le transport :

- Unité de transport intermodal (porte-containers/train/camion)
- Environnement isolé de l'extérieur
- Unité de packaging
- Normalisé
- Réutilisabilité

En informatique :

- Isolation de processus
- Ouverture au cloud (portabilité/normalisation)
- Déploiements rapides

Docker

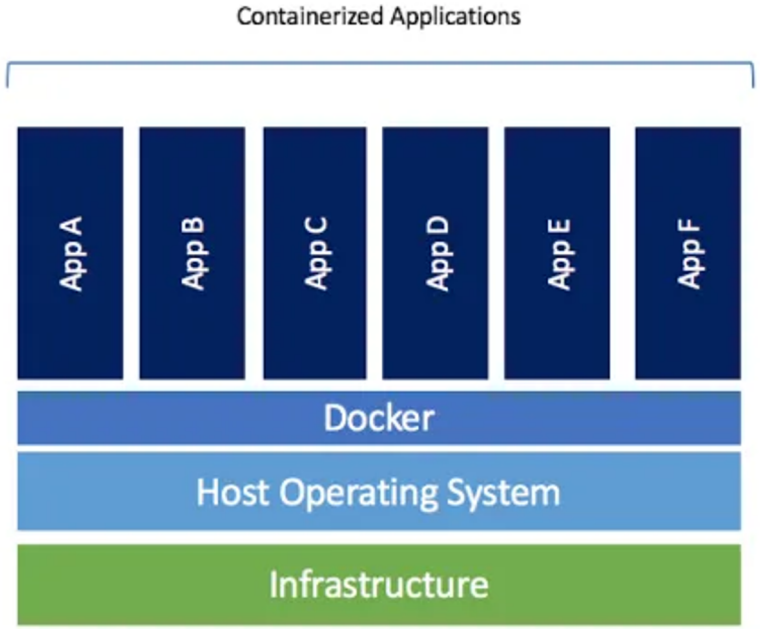
- But : Pouvoir exécuter des processus, indépendamment des autres, mais de manière plus légère que la virtualisation (VM)
- Idéal pour les environnements de type “Cloud”

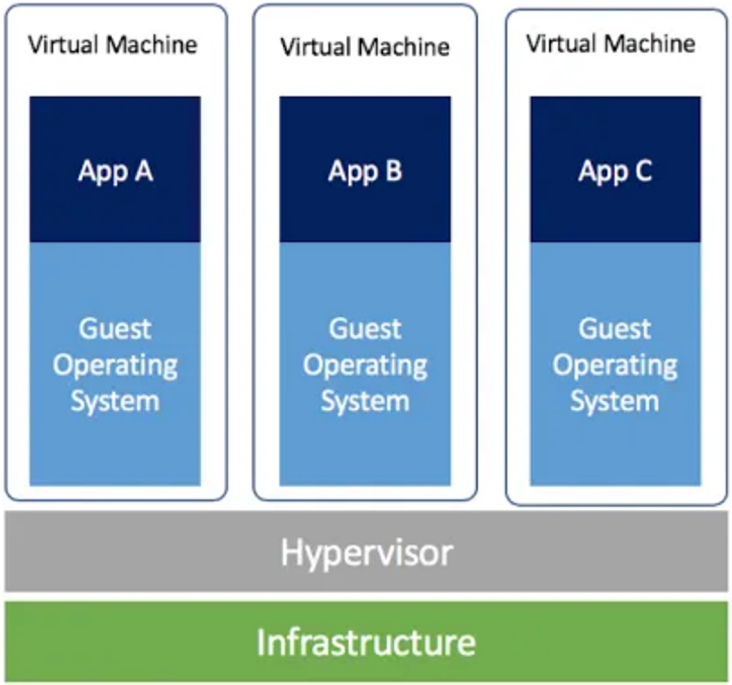
Les composants de Docker :

- un démon / serveur (appelé dockerd / docker engine)
- un “CLI” (commande docker) => permet de controller le démon docker
- un “hub” : repository de partage d’images

Container et VMs :

Pas de surcouche liée à l'hyperviseur et l'OS hôte.





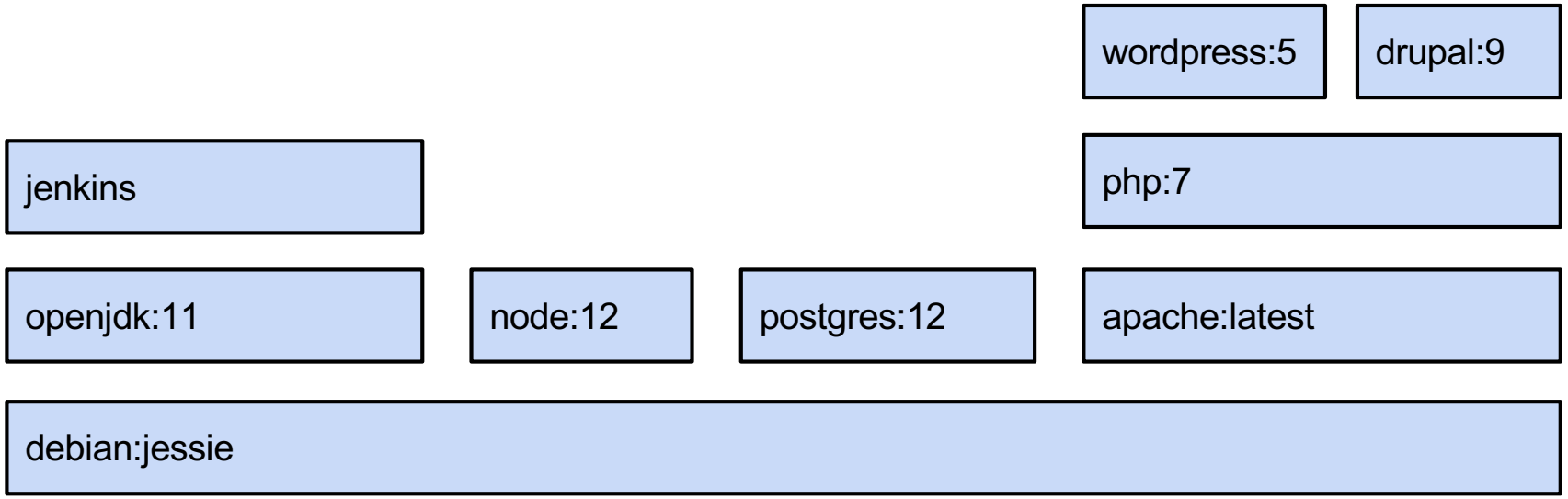
https://www.docker.com/blog/containers-replacing-virtual-machines/

Images et containers

- Les images sont l'unité de base de docker
- Les images sont découpées en 'layers'
- Une image s'appuie sur une image parente
- Un Dockerfile décrit les étapes de la construction d'une image
- Les containers sont instanciés à partir d'images
- Les images sont nommées via des tags
- Les images sont read-only
- Les containers sont read-write
- Un container exécute un process, et s'arrête avec lui

Images et containers

Images parentes et ‘layers’ réutilisables



Images applicatives

- Une image contient
  - une application / un service (un jar/un SGBD)
  - son environnement d'exécution (jre/middlewares/librairies)

Interêts

- Exécuter facilement des applications s'appuyant sur des middleware/librairies différents (ex: une application en java 8 et une application en java 11, ou plusieurs versions de postgreSQL)
- Simplifier la gestion et le déploiement des applications
- Faire cohabiter de multiples versions
- Limiter les impacts sur le système d'exploitation hôte

Les commandes principales
Manipulation de containers :

Commande	ça fait quoi?
docker create --name <CONTAINER> <IMAGE_NAME>	crée un container nommé à partir d'une image
docker start <CONTAINER> / docker stop <CONTAiNER>	démarre / arrète un container le stop est fait en envoyant un SIGTERM au processus, puis un SIGKILL si le processus ne s'est pas arrété au bout d'un certain temps
docker logs <CONTAINER>	affiche les logs d'un container (stdout+stderr)
docker exec <CONTAINER> <CMD>	exécute une commande dans un container existant

Les commandes principales
Commandes de management:

Commande	ça fait quoi?
docker images	liste les images disponibles localement
docker ps [-a]	liste les containers en l'état "RUNNING", (-a: tous les containers)
docker rm <CONTAINER>	supprime un container
docker rmi <IMAGE>	supprime une image
docker pull <IMAGE>	télécharge une image depuis un registry (hub.docker.com)
docker push <IMAGE>	envoie une image vers un registry
docker tag <IMAGE> <TAG>	crée un nouveau tag pour une image

Utilisation pour les développeurs

Construire un environnement de développement
Démarrer une base de données mongodb locale (sur le port 27117) :

```
$ docker pull mongo:4
4: Pulling from library/mongo
...
Digest: sha256:08105c3b620f4feea45c7f26fa0024d546f7cf1bebc391325937f56104545239
Status: Downloaded newer image for mongo:4
docker.io/library/mongo:4

$ docker create --name dev-db -p 27117:27017 mongo:4
f21ba20566c5cf506ed7c56e5bbffa861b0de3e804f8e42b4f61955d7cb362b6

$ docker start dev-db
dev-db
$ docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED          STATUS          PORTS                      NAMES
f21ba20566c5   mongo:4    "docker-entrypoint.s…"   45 seconds ago   Up 30 seconds   0.0.0.0:27117->27017/tcp   dev-db
```

Utilisation pour les développeurs

Construire un environnement de développement
Démarrer une base de données postgresql locale (sur le port 2345) :

```
$ docker pull postgres
Using default tag: latest
latest: Pulling from library/postgres
...
Digest: sha256:be456a40361cd836e0e1b35fc4d872e20e138f214c93138425169c4a2dfe1b0e
Status: Downloaded newer image for postgres:latest
docker.io/library/postgres:latest

$ docker create --name dev-db-pg -p 2345:5432 postgres
25af7ea7791a49317850a1e8303d222d0aa5508d1ebdec169ef94321be87af77

$ docker start dev-db-pg
dev-db-pg

$ docker ps
CONTAINER ID    IMAGE     COMMAND                  CREATED          STATUS          PORTS                      NAMES
25af7ea7791a    postgres  "docker-entrypoint.s…"   45 seconds ago   Up 30 seconds   0.0.0.0:2345->5432/tcp     dev-db-pg
f21ba20566c5    mongo:4   "docker-entrypoint.s…"   4 minutes ago    Up 4 minutes    0.0.0.0:27117->27017/tcp   dev-db
```

Construire une image applicative

Dockerfile

Commande	ça fait quoi?
FROM <IMAGE>	indique l'image parente
COPY <SRC> <DEST>	copie des fichiers du répertoire de construction dans l'image
RUN <COMMANDE>	exécute une commande dans l'image
CMD <COMMANDE>	indique quelle commande doit être exécutée au démarrage du container
EXPOSE <PORT>	marque un port à exposer
ENV <KEY=VALUE>	déclare une variable d'environnement
WORKDIR	modifie le répertoire courant à l'intérieur de l'image

Construire une image applicative

Dockerfile

```
1 FROM maven:3-jdk-11
2 # import du répertoire courant dans l'image
3 COPY . /app
4 # compilation
5 RUN mvn --batch-mode -DskipTests -f /pom.xml clean package
6 # exposition du port
7 EXPOSE 8080
8 # commande à exécuter
9 WORKDIR /opt/target
10 CMD ["java", "-jar", "mon-appli-1.0.0-SNAPSHOT.jar"]
```

Utilisation pour spring-boot

Construire une image applicative

Construction de l'image et exécution

```
1 docker build -t mon-image .
2
3 docker create -p 8080:8080 mon-container mon-image
4
5 docker start mon-container
```

Conclusion

Docker permet de :

- exécuter des applications/services facilement
  - monter rapidement un environnement complet
  - préparer un environnement de développement middleware + base de données
- isoler les applications/services
  - cohabitation des services
  - pas de gestion de middleware/librairies sur les serveurs
- livrer les applications prêtes à l'emploi
  - création d'images applicatives (application + middleware)
  - publication sur un registry partagé