Atelier de Génie Logiciel

HAI501I

Cours 5

Bilan mi parcours

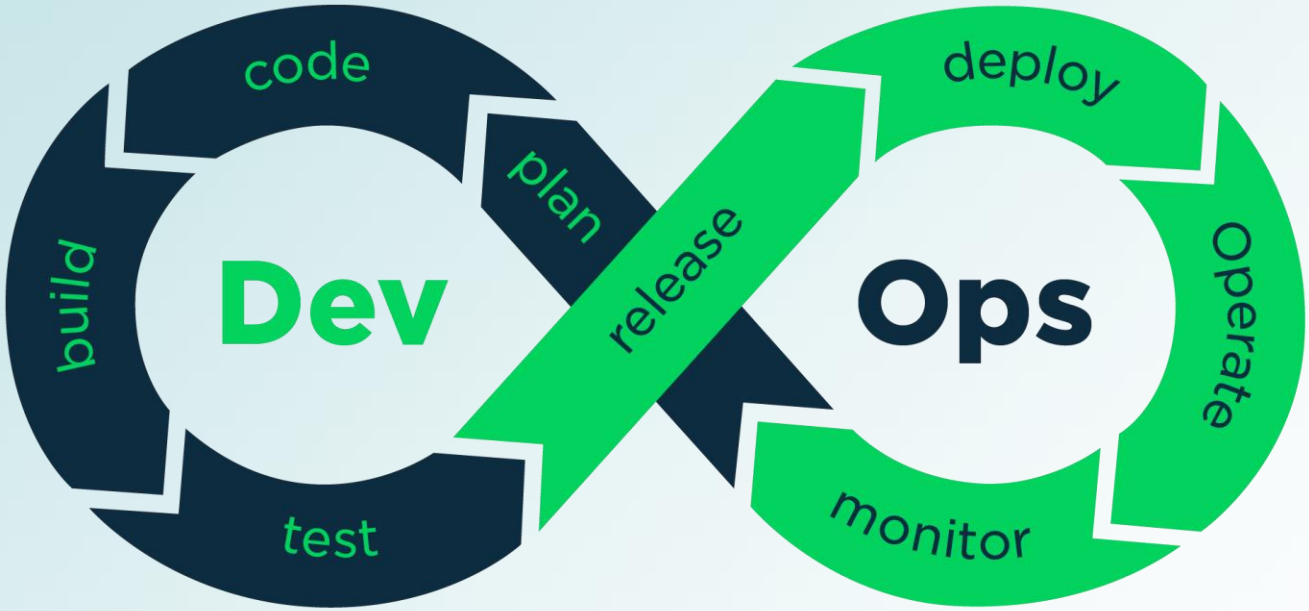
+

Docker et la conteneurisation

DevOps

LABEL 10

- CODE
- BUILD
- TEST
- RELEASE
- DEPLOY
- OPERATE
- MONITOR



GOTO 10

Bilan mi-parcours

CODE

- Code
  - Java (L3G)
  - Eclipse (IDE)
  - Git (CVS)

BUILD

- Build
  - Maven (PMS)

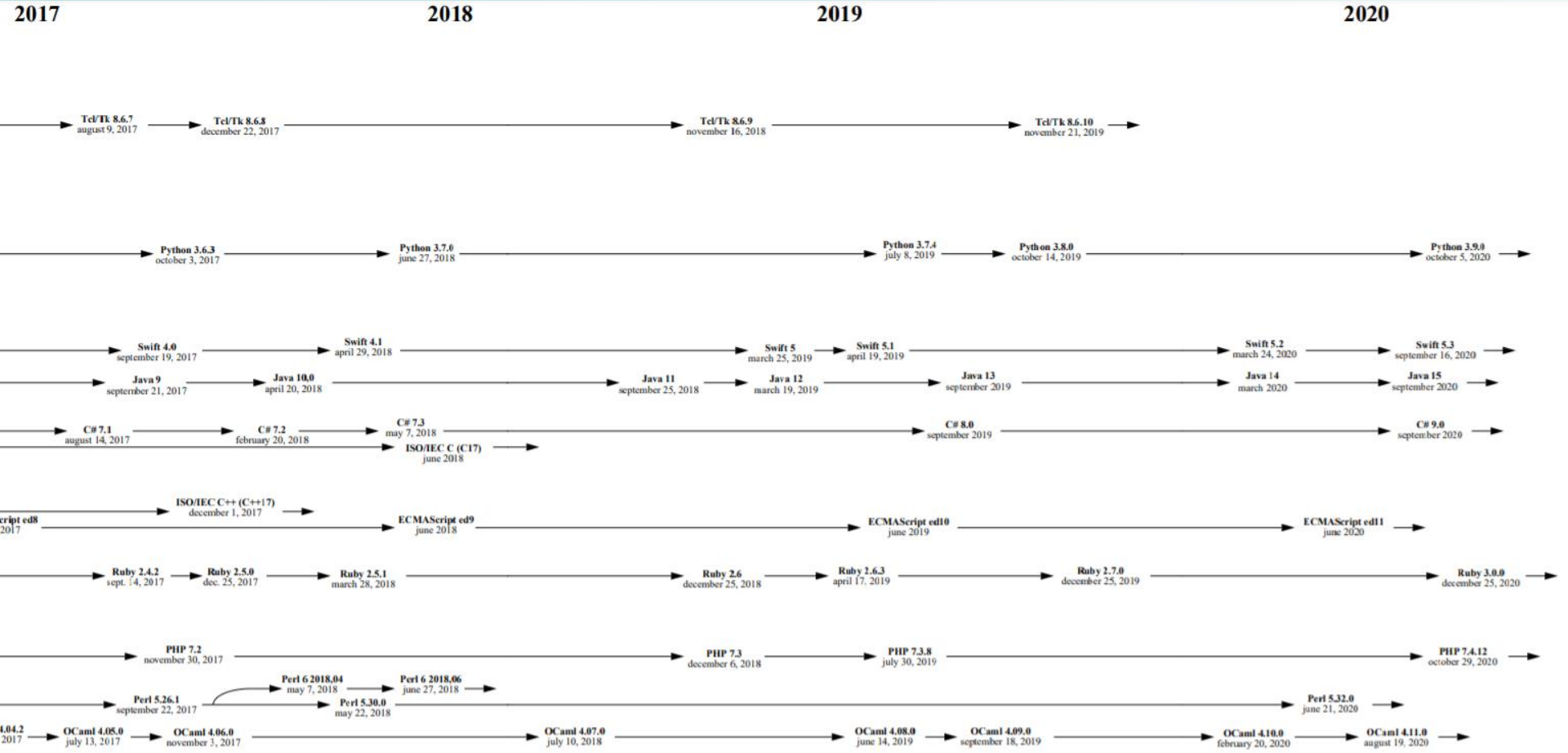
TEST

- Test
  - Junit (tests unitaires)
  - Jacoco (Couverture de code)
  - Mockito (Validation des Tests)

Cours	TD
L2 !	TD1
	TD2
C1	TD2
C2	TD3
C3	TD4
C4	TD5

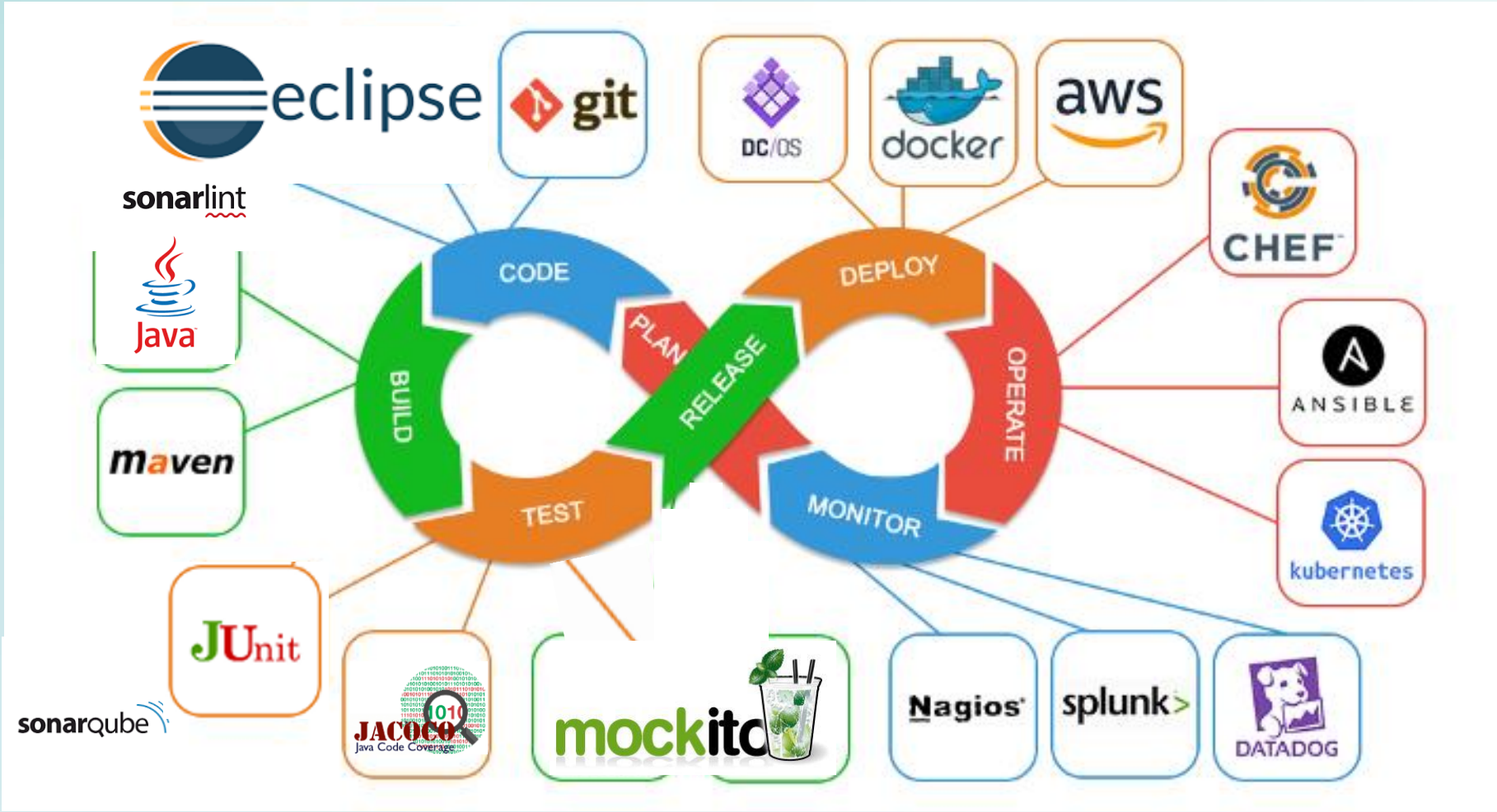
L’histoire des langages informatiques

https://www.levenez.com/lang/



https://spectrum.ieee.org/top-programming-languages/#toggle-gdpr

Prolongation de la boucle



A suivre

TEST RELEASE DEPLOY

- Qualimétrie de code
  - SonarQube (sonarlint)
- Intégration Continue
  - ~~Jenkins~~ (Gitlab CI/CD)
- Releases
  - Gitlab (GitFlow)
- Gestion des bugs
  - ~~BugZilla~~ (Gitlab issues)
- Déploiement
  - Docker (container)

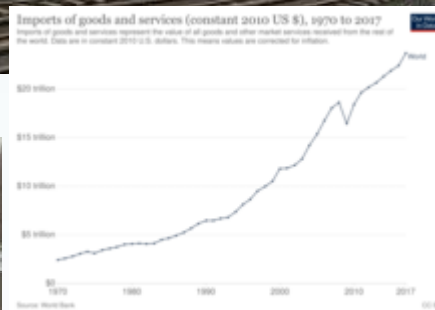
Cours	TD
C4	TD6
C6	TD8
C6	TD8
C6	TD8
C5	TD7

Conteneur IRL

Révolution Industrielle & Commerce Mondial

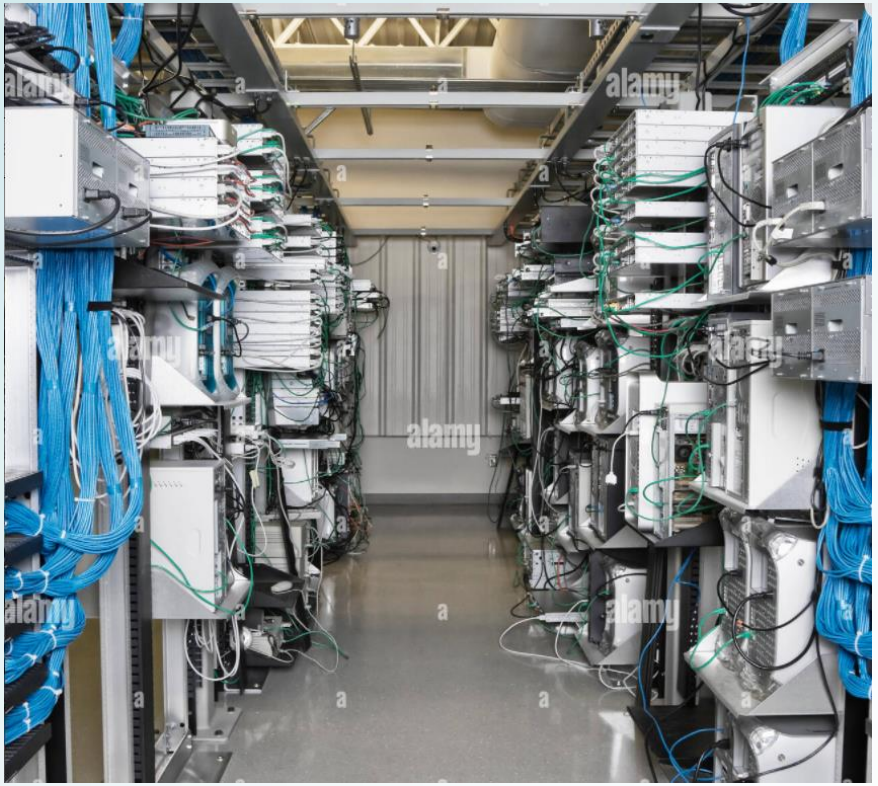
Les dimensions du container 20 pieds sont :

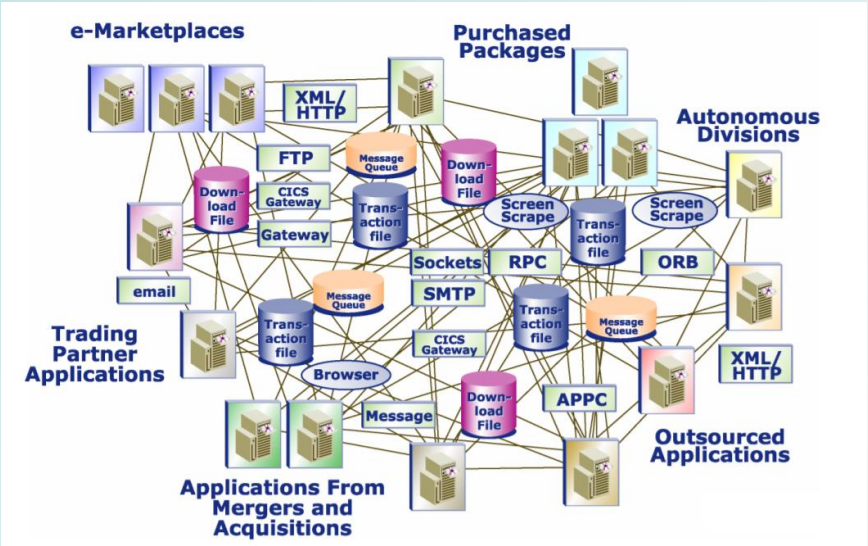
- Longueur : 6m
- Largeur : 2,43m
- Hauteur : 2,59m
- Poids maximum : 1270 kg
- Capacité de stockage : **de 30 à 33m3**



L’impasse des serveurs / machines et des OS

- Un serveur applicatif pour chaque système d’exploitation
- Un serveur applicatif pour chaque application (sécurité, dimensionnement, version des librairies tierces maitrisées)
- Un serveur backup en cas de problèmes
- Mutualisation vs QOS



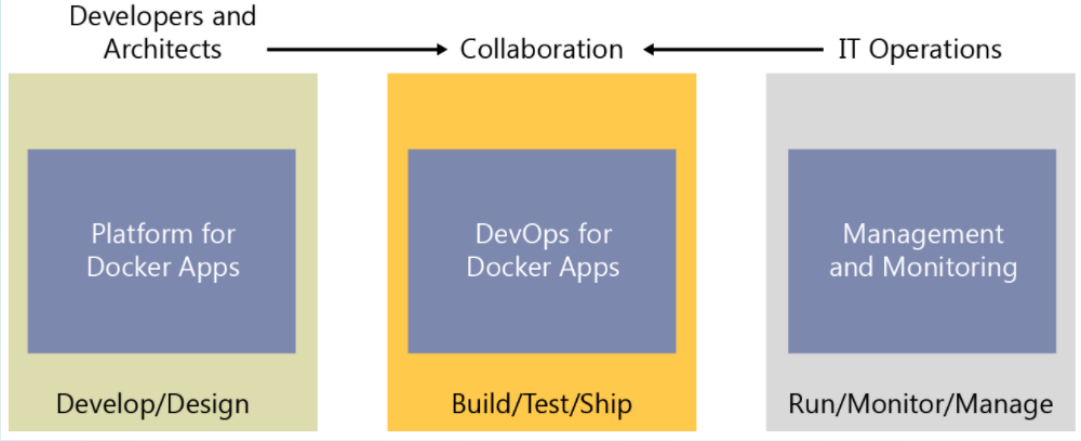


Machine Virtuelle

- Densité infrastructure
  - 1 seule grosse machine
  - n machines virtuelles
- Déploiement anticipé/facilité (architecture cible)
- Virtualisation Lourde
  - Good
    - Multi OS
    - Isolée
    - Ressource réservée
    - Consommation énergétique optimisée
  - Bad
    - Start lent
    - Ressources réservées

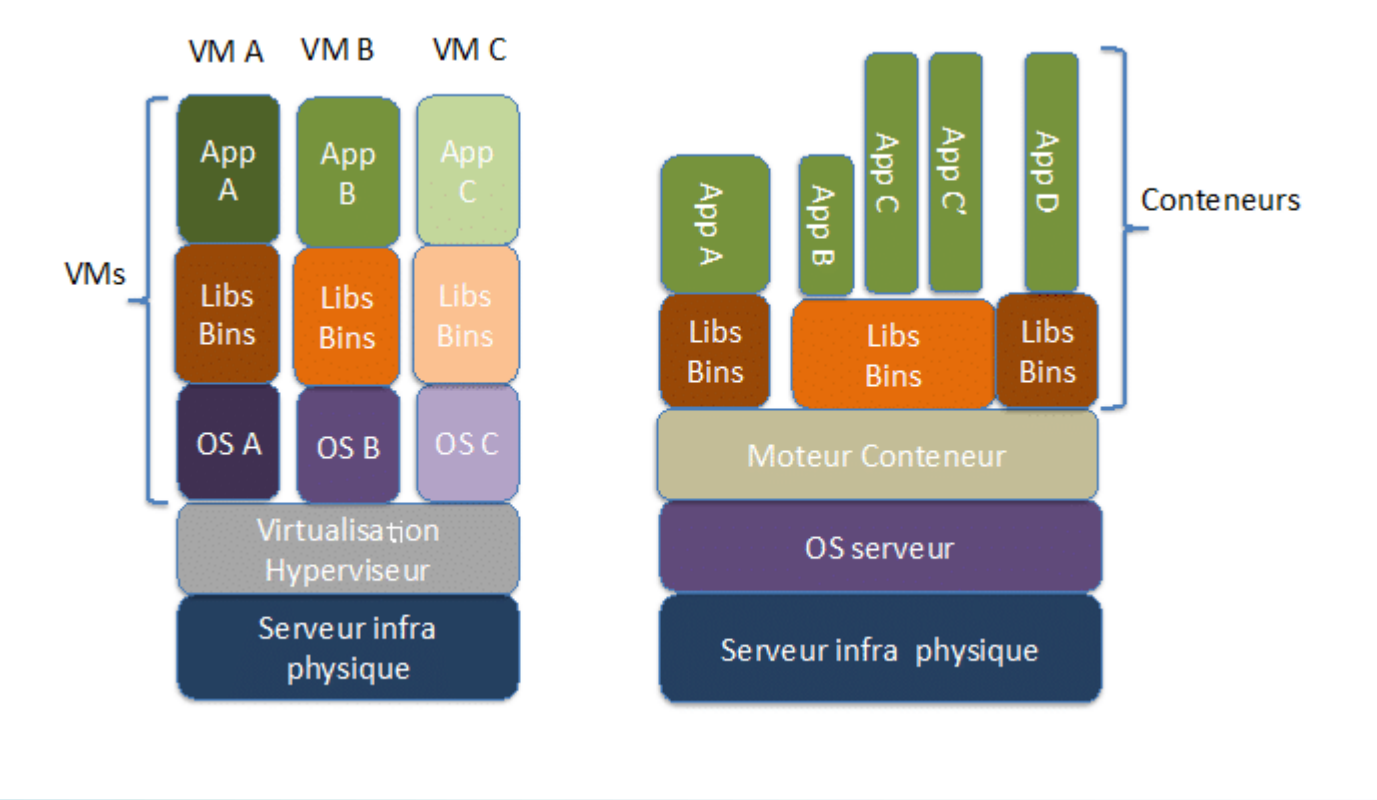
Conteneur (d'application)

- Virtualisation légère
  - non réservée
- Volatile, permet la scalabilité on-demand
- Permet à une équipe de s'abstraire de l'OS
- Au cœur de la révolution DevOps



Hyperviseur vs Conteneur

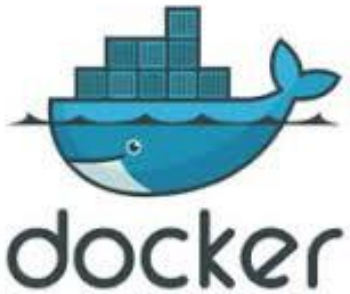
- D’un point de vue applicatif, c’est la même architecture logicielle
- Conteneur transportable : création d’image distribuable



ou les 2 !

LXD

Docker



- Platform as a Service (PaaS) v2
  - PaaS v1 (Heroku)
    simple espace de déploiement cloud de site web
- Immutabilité
  - On ne modifie pas
  - On redéploye une nouvelle image
  - Une philosophie de déploiement
- Reproductibilité
  - *ça marche sur ma machine (sic)*
  - Ça marchera chez vous !

Hello Word

```
$ docker run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/


$ docker images hello-world
REPOSITORY   TAG      IMAGE ID      SIZE
hello-world  latest   feb5d9fea6a5  13256
```

Statefull/StateLess

- StateLess / StateFull
  - State Full = se rallumer dans l'état précédent (data management)
  - State Less = toujours les mêmes actions réalisées de manière indépendantes de l'historique (traitement requête http), aucun état nécessaire
- 1 conteneur = 1 processus
  - LAMP = 3 conteneurs (Apache, MySQL, PhP)
  - docker-compose pour assembler tous les composants (*stack*) via un fichier Manifest

SETUP

- Un fichier ‘batch’ de configuration (DockerFile)

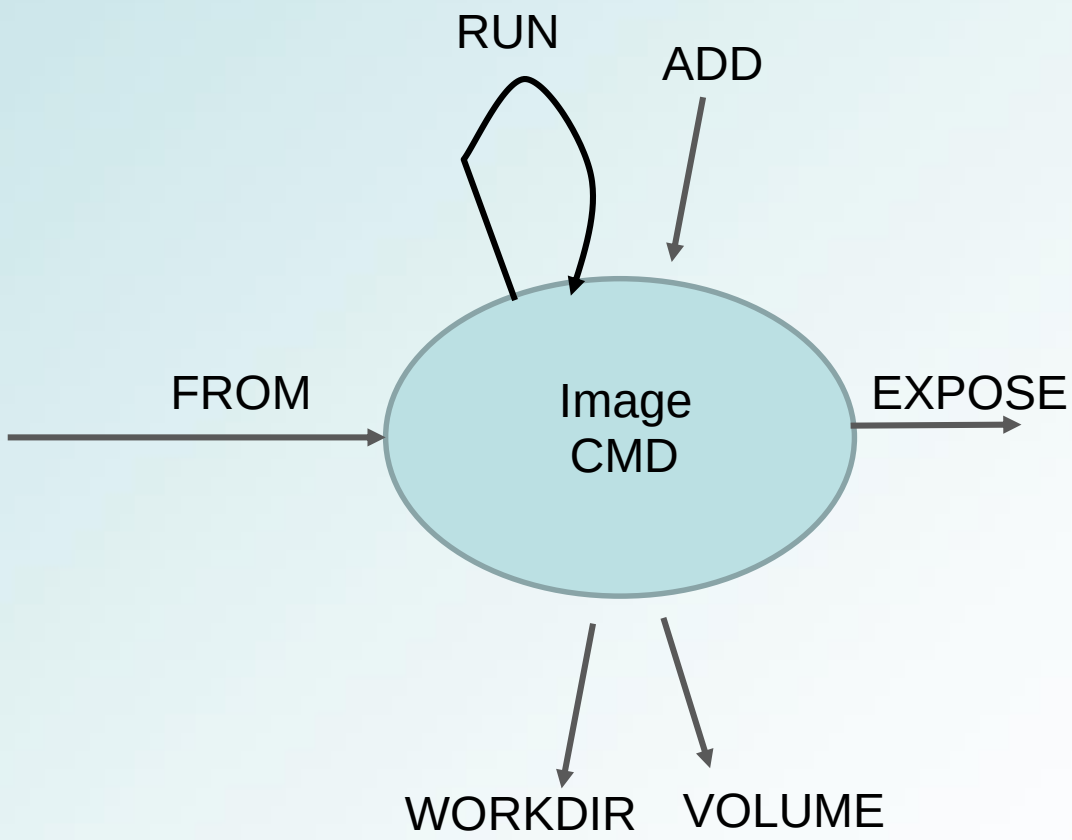
```
# syntax=docker/dockerfile:1
FROM python:3.7-alpine
WORKDIR /code
ENV FLASK_APP=app.py
ENV FLASK_RUN_HOST=0.0.0.0
RUN apk add --no-cache gcc musl-dev linux-headers
COPY requirements.txt requirements.txt
RUN pip install -r requirements.txt
EXPOSE 5000
COPY . .
CMD ["flask", "run"]
```

- Un fichier ‘d’assemblage’ (docker-compose.yml)
- Un Build

```
$ docker-compose up
```

Dockerfile

- FROM source
- RUN exécute
- ADD ajoute
- WORKDIR home
- EXPOSE port
- VOLUME disque
- CMD run initial



docker-compose.yml

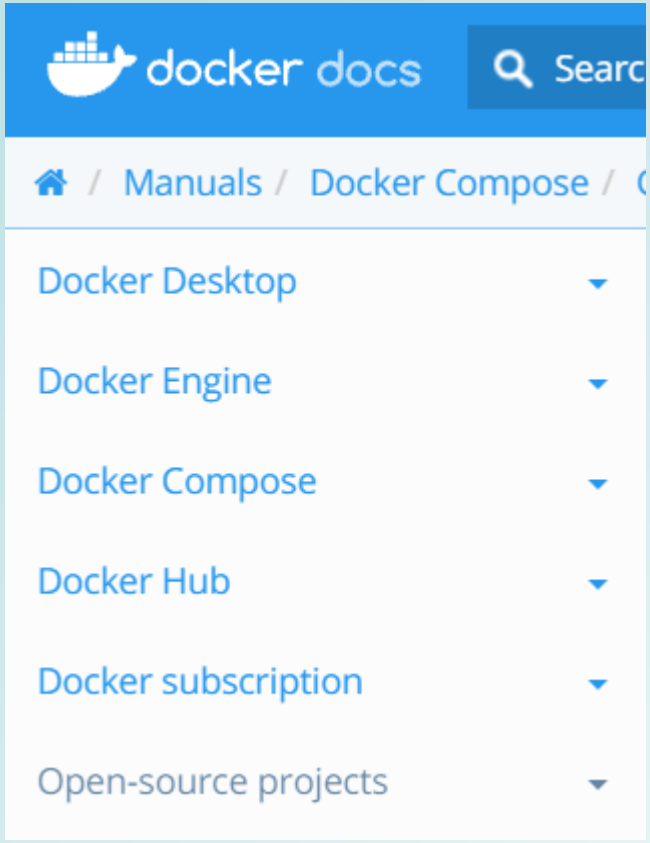
- Définir des services
- Les lier entre eux
  - origine,
  - dépendance,
  - variables d'environnement,
  - statut

```yaml
version: '3'
services:
 mysql:
   image: mysql:5.7
   volumes:
     - db_data:/var/lib/mysql
   restart: always
   environment:
     MYSQL_ROOT_PASSWORD: monPassword
     MYSQL_DATABASE: ghost
     MYSQL_USER: ghostuser
     MYSQL_PASSWORD: ocrpassword

 ghost:
   depends_on:
     - mysql
   image: mon_image_docker
   ports:
     - "8080:80"
   restart: always
   environment:
    NODE_ENV: production

volumes:
 db_data: {}
```

Une offre intégrée



La boucle est bouclée