DEVOPS

Abida Mounir

Maitre Technologue

RSI

AU. 2024-2025

Les conteneurs Plan

Objectifs

DevOps

Serveur sans Docker

Serveur avec Docker

Image Docker, Docker Hub, Docker File, Docker compose, …

Orchestration et gestion des conteneurs

Docker Swarm

Google Kubernetes

Les conteneurs

Objectifs

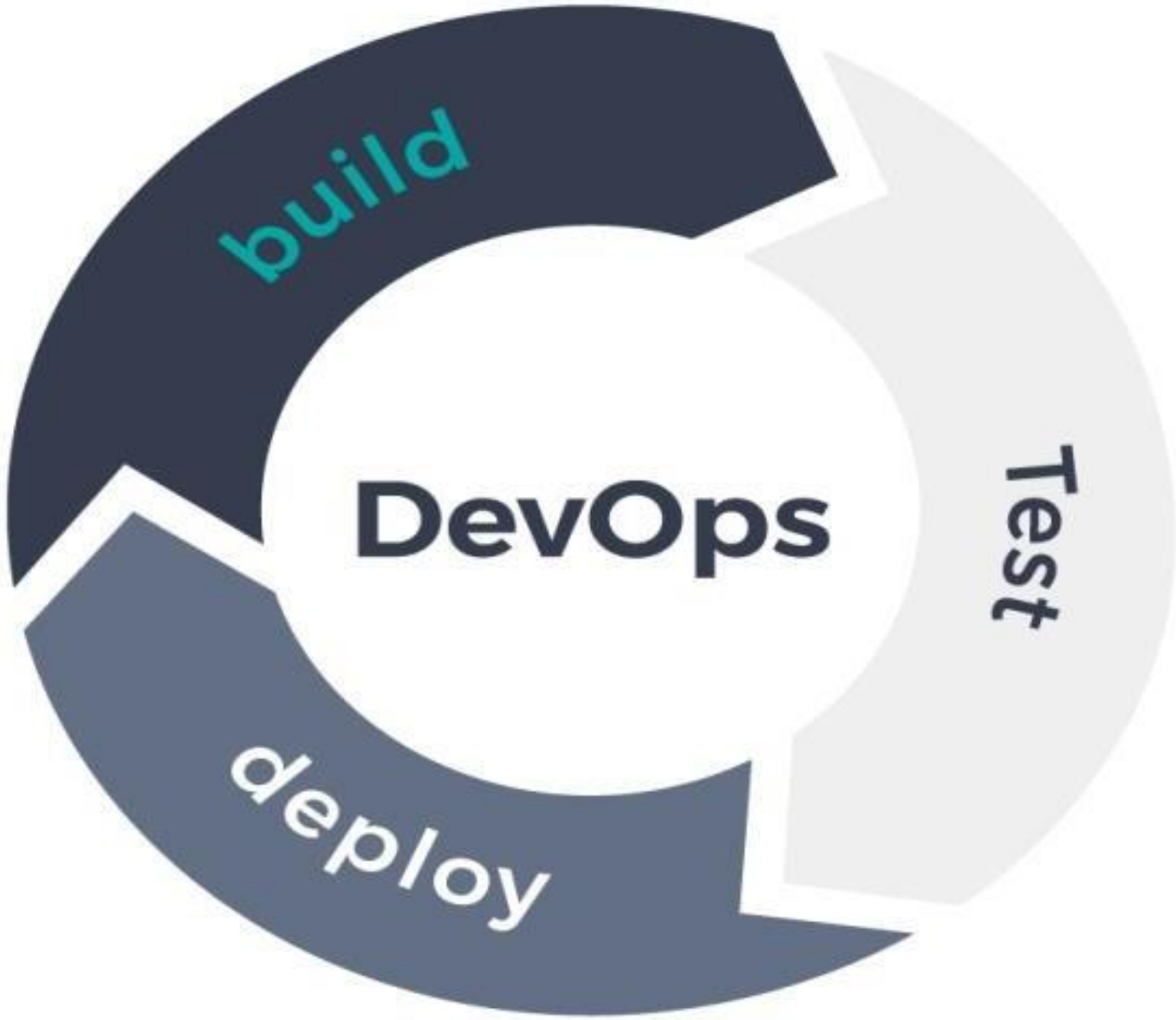
- Simplifier le déploiement des environnements avec des conteneurs portables et légers.
- Embarquer un micro service et ses dépendances dans un conteneur isolé du système d'exploitation.
- Exécuter et orchestrer des instances sur n'importe quel OS - y compris celui du développeur.
- Rolling upgrade, clustering, élasticité, et bien plus encore.

DevOps

- DevOps ?
- Problème de compatibilité :

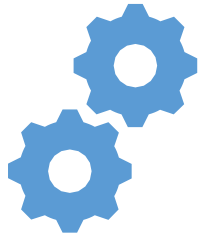
« Chez moi ça marche » !

- Le devops est un mouvement en ingénierie informatique et une pratique technique visant à l'unification du développement logiciel (dev) et de l'administration des infrastructures informatiques (ops), notamment l'administration système.



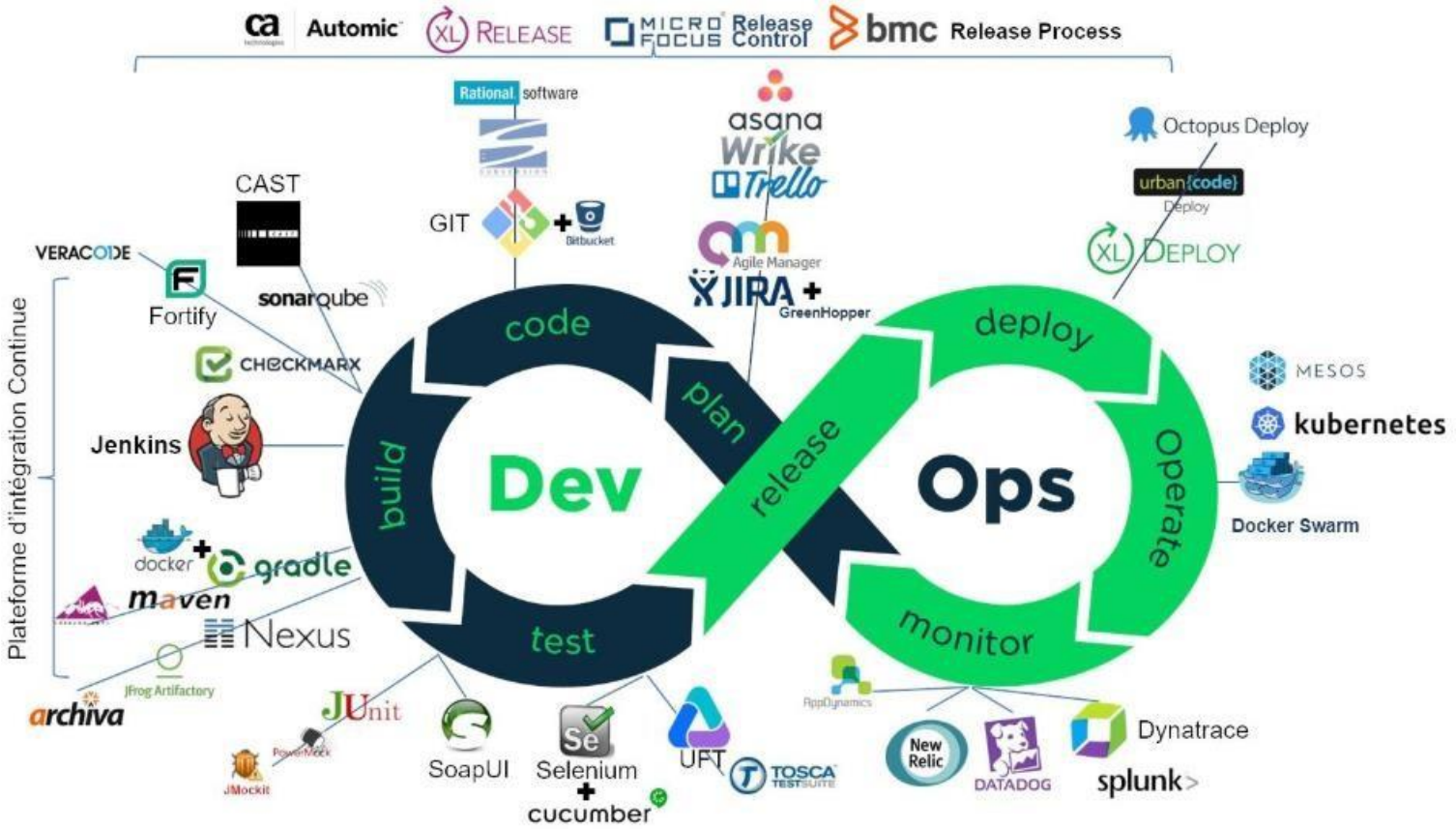
DevOps

Dans un modèle DevOps, les équipes de développement et d'opérations ne sont plus isolées. Il arrive qu'elles soient fusionnées en une seule et même équipe. Les ingénieurs qui la composent travaillent alors sur tout le cycle de vie d'une application, de la création à l'exploitation, en passant par les tests et le déploiement, et développent toute une gamme de compétences liées à différentes fonctions.



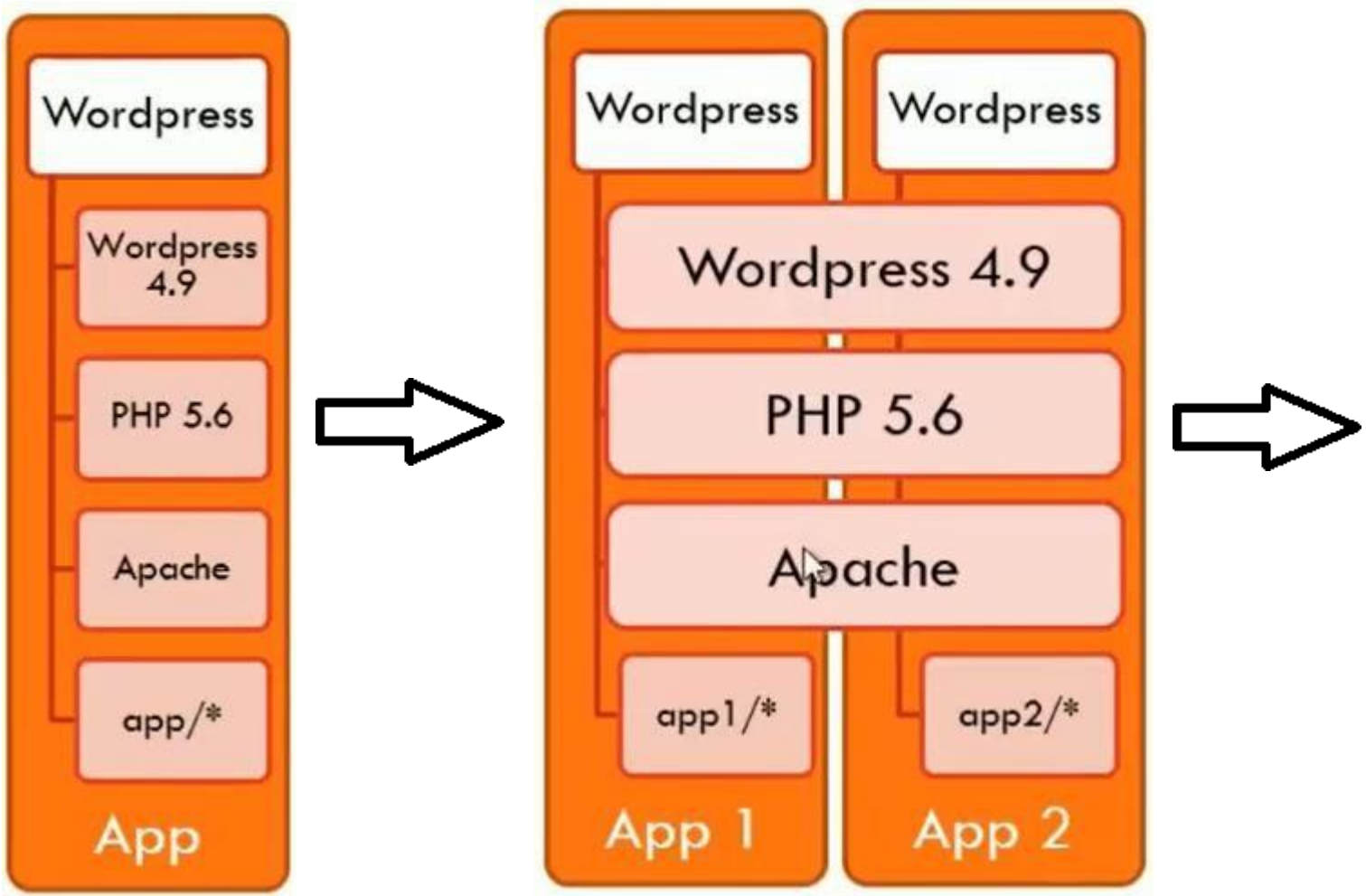
Ces équipes utilisent des pratiques pour automatiser des processus qui étaient autrefois manuels et lents. Elles exploitent une pile technologique et des outils qui les aident à faire fonctionner et à faire évoluer les applications de façon rapide et fiable. Ces outils aident également les ingénieurs à accomplir de façon autonome des tâches (par exemple, le déploiement de code ou la mise en service d'infrastructure) qui nécessiteraient normalement l'aide d'autres équipes, ce qui augmente encore davantage la productivité de l'équipe d'ingénieurs.

DevOps


Automic
RELEASE
MICRO FOCUS Release Control
bmc Release Process
Rational software
asana
Wrike
Trello
Octopus Deploy
urban{code} Deploy
DEPLOY
CAST
GIT
Bitbucket
Agile Manager
JIRA + GreenHopper
VERACODE
Fortify
sonarqube
CHECKMARX
Jenkins
Plateforme d'intégration Continue
code
plan
build
Dev
release
deploy
Ops
Operate
monitor
test
MESOS
kubernetes
Docker Swarm
docker
gradle
maven
Nexus
JFrog Artifactory
archiva
JUnit
PowerMock
JMockit
SoapUI
Selenium
cucumber
UFT
TOSCA TESTSUITE
AppDynamics
New Relic
DATADOG
Dynatrace
splunk>

Les conteneurs

Serveur sans Docker

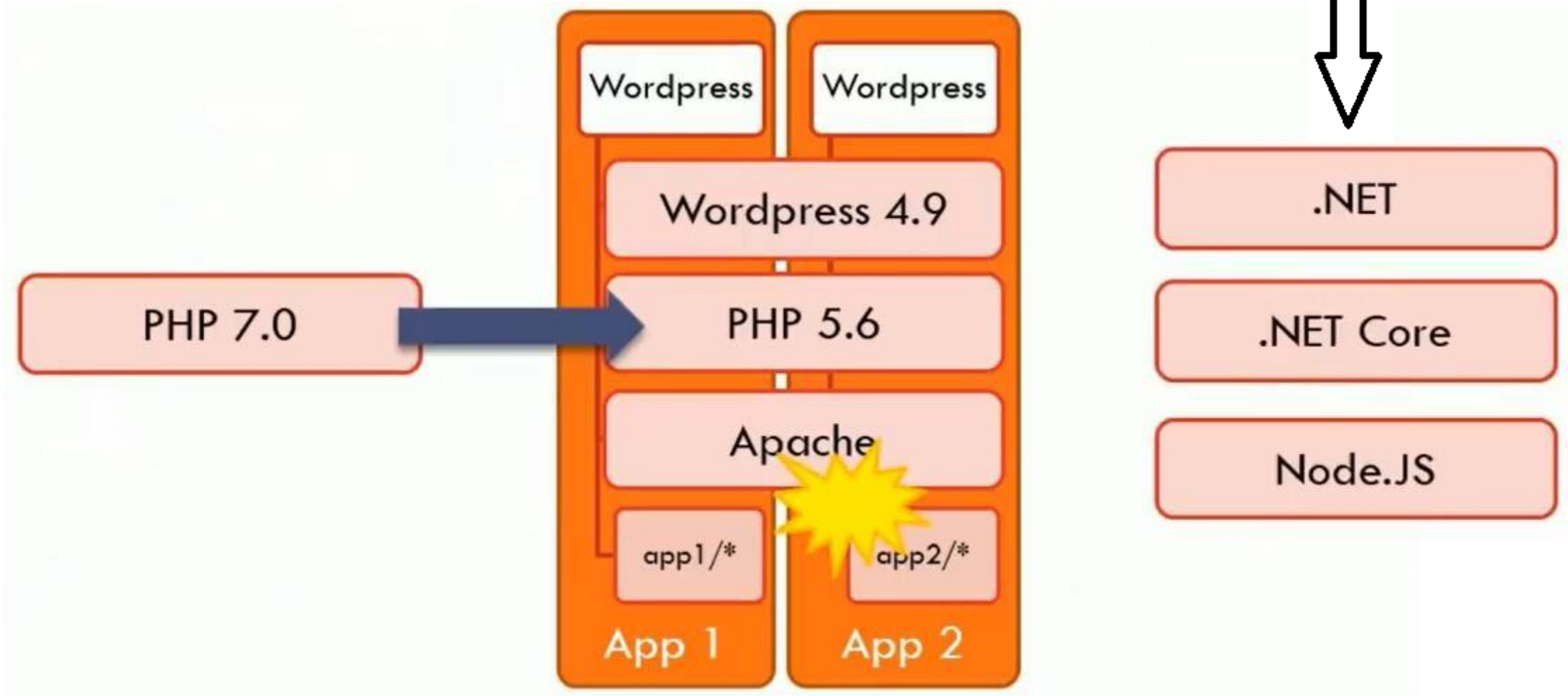


Quoi faire si une application nécessite une remontée de version pour PHP par exemple ?

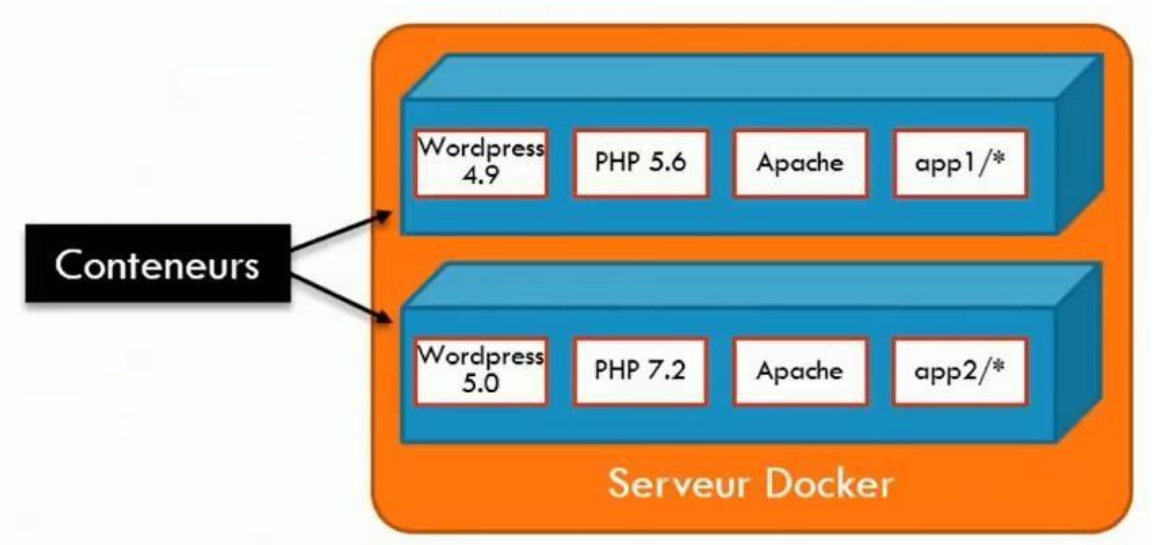
Les conteneurs

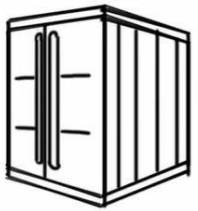
Serveur sans Docker

Et pour les autres outils et dépendances ?



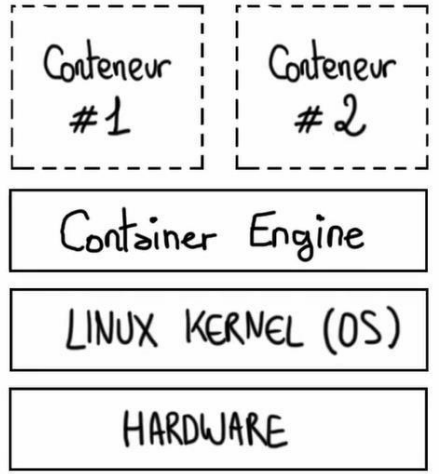
Les conteneurs
Serveur avec Docker





PRINCIPES

- Cloisonnement niveau OS
- Virtualisation de l'environnement d'éxecution



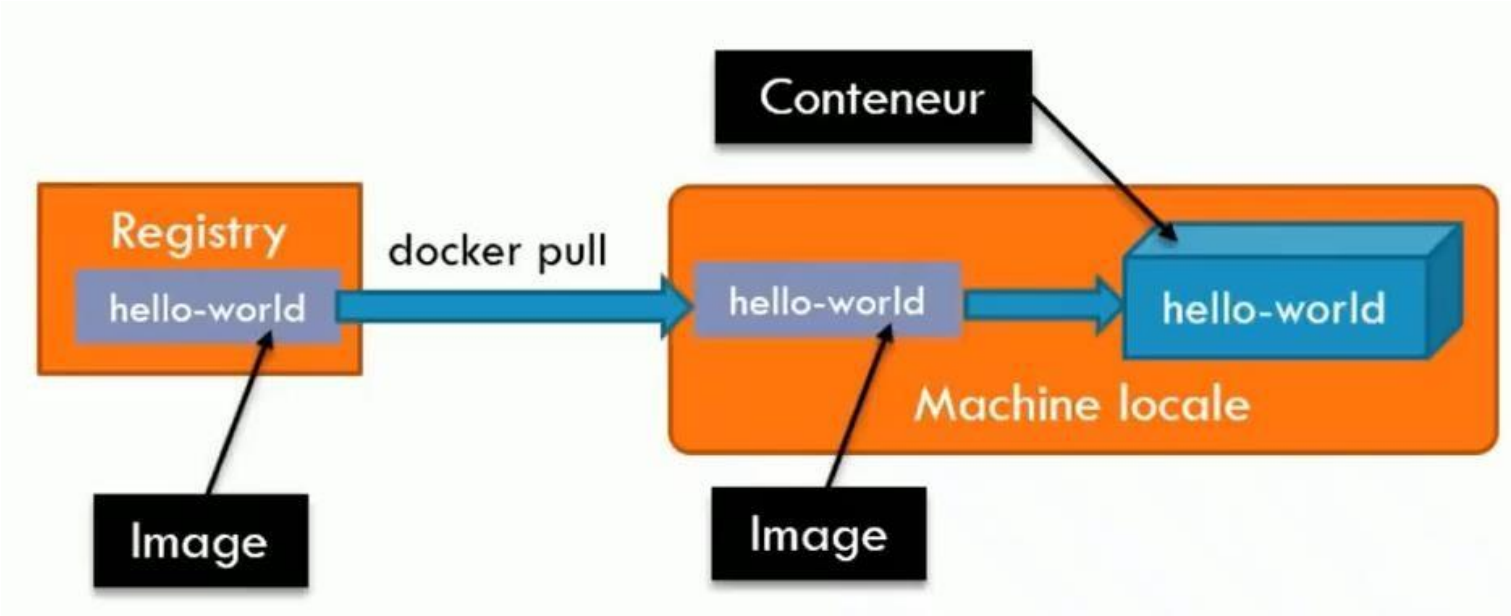
Les conteneurs
Livrer était déjà compliqué

Les conteneurs Homogénéité pour tous

Les conteneurs

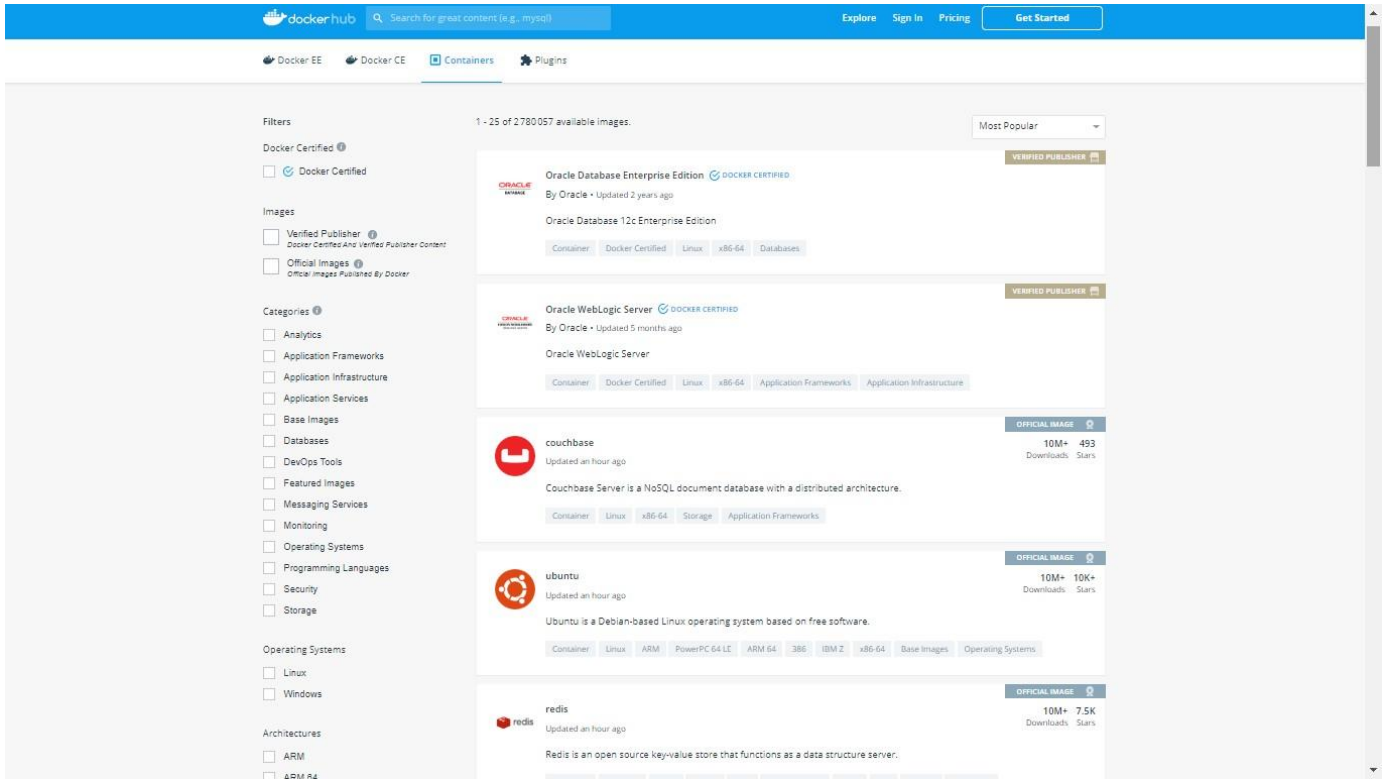
Comparaison avec les machines virtuelles

Les conteneurs Image / conteneur



Les conteneurs Docker Hub : Le répertoire des images docker.

Virtualisation et cloud

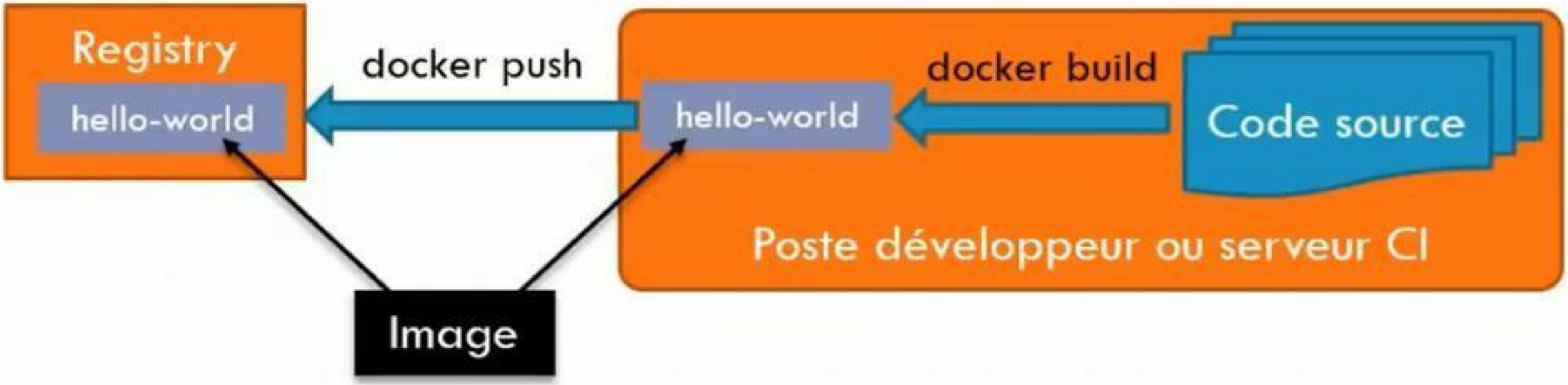


Les conteneurs
Exemple : Nginx.

```
elyes@ult-tunisie:~$ docker run --name mynginx1 -p 8080:80 -d nginx
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
8d691f585fa8: Already exists
5b07f4e08ad0: Pull complete
abc291867bca: Pull complete
Digest: sha256:922c815aa4df050d4df476e92daed4231f466acc8ee90e0e774951b0fd7195a4
Status: Downloaded newer image for nginx:latest
65b1b41f55dd54c26b828321ad4cb8afeb1b05e28484312ada01edae89bd94d2
```

```
FROM wordpress:4.9.4-php5.6-apache

COPY ./my-dir /var/www/html
```



Les conteneurs
Créer une image : fichier Dockerfile

Les conteneurs Créer une image : fichier Dockerfile

- Les Dockerfiles sont des fichiers qui permettent de construire une image Docker adaptée à vos besoins, étape par étape.

```
# Image de base
FROM debian:jessie

# Installation de curl avec apt-get
RUN apt-get update \
&& apt-get install -y curl \
&& rm -rf /var/lib/apt/lists/*

# Installation de Node.js à partir du site officiel
RUN curl -LO "https://nodejs.org/dist/v0.12.5/node-v0.12.5-linux-x64.tar.gz" \
&& tar -xzf node-v0.12.5-linux-x64.tar.gz -C /usr/local --strip-components=1 \
&& rm node-v0.12.5-linux-x64.tar.gz

# Ajout du fichier de dépendances package.json
ADD package.json /app/

# Changement du repertoire courant
WORKDIR /app

# Installation des dépendances
RUN npm install

# Ajout des sources
ADD . /app/

# On expose le port 3000
EXPOSE 3000

# On partage un dossier de log
VOLUME /app/log

# On lance le serveur quand on démarre le conteneur
```

Les conteneurs

Docker compose

- Docker Compose est un outil qui permet de décrire (dans un fichier YAML) et gérer (en ligne de commande) plusieurs conteneurs comme un ensemble de services interconnectés. Si vous travaillez sur une application Rails, vous pouvez par exemple décrire un ensemble composé de trois conteneurs :
  - un conteneur PostgreSQL
  - un conteneur Redis
  - un conteneur pour le code de votre application
- Vous pouvez alors démarrer votre ensemble de conteneurs en une seule commande **docker-compose up**. Sans Docker Compose, vous aurez dû lancer trois commandes **docker run** avec beaucoup d'arguments pour arriver au même résultat.
- Dans le fichier docker-compose.yml, chaque conteneur est décrit avec un ensemble de paramètres qui correspondent aux options disponibles lors d'un **docker run** : l'image à utiliser, les volumes à monter, les ports à ouvrir, etc. Mais vous pouvez également y décrire des éléments supplémentaires, comme la possibilité de « construire » (**docker build**) une image à la volée avant d'en lancer le conteneur.

Les conteneurs

Exemple d'une configuration Docker Compose

- Voici un exemple de fichier docker-compose.yml :

```yaml
version: 3

services:

  postgres:
    image: postgres:10
    environment:
      POSTGRES_USER: rails_user
      POSTGRES_PASSWORD: rails_password
      POSTGRES_DB: rails_db

  redis:
    image: redis:3.2-alpine

  rails:
    build: .
    depends_on:
      - postgres
      - redis
    environment:
      DATABASE_URL: 'postgres://rails_user:rails_password@postgres:5432/rails_db'
      REDIS_HOST: 'redis:6379'
    volumes:
      - .:/app

  nginx:
    image: nginx:latest
    links:
      - rails
    ports:
      - 3000:80
    volumes:
      - ./nginx.conf:/etc/nginx/conf.d/default.conf:ro
```

Dans cet exemple, quatre services sont décrits :

1. Un conteneur ***postgres*** fournissant PostgreSQL.
2. Un conteneur ***redis*** fournissant Redis.
3. Un conteneur ***rails*** fournissant mon application.
4. Un conteneur ***nginx*** fournissant un point d'entrée sur le port 3000.