Docker

Rôle & fonctionnement général

Prise en main

Prégnance dans l’IT d’aujourd’hui et demain

Rôle & fonctionnement général

Pourquoi virtualiser des applications?

Outre son code source et éventuels produits de compilation, un applicatif nécessite souvent de **nombreux “à côté”**

- Fichiers de conf présents à un endroit spécifique
- Existence de variables d’environnement
- Librairies diverses
- Dépendances systèmes

Pourquoi virtualiser des applications?

Une “*installation traditionnelle*” vient avec son lot d’emmerdements

- Longs process d’installation manuelle
- Exceptions de scripting résistantes à la variabilisation
- Conflits de librairies sur un même serveur
- L’env de DEV n’est pas le même que la PROD
- Relicats d’installations précédentes
- Problème de reproductibilité

I had a dream

Dans un monde parfait, l’admin sys devrait pouvoir déployer un applicatif sur un serveur:

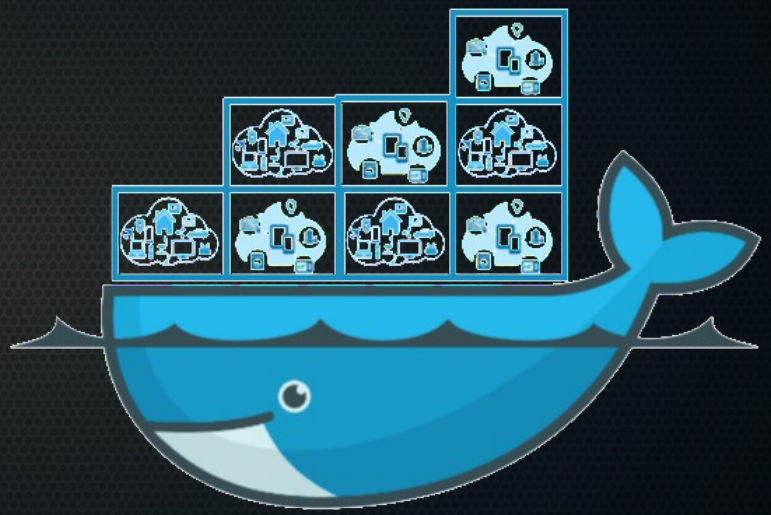
- Sans craindre de planter les autres (**encapsulation**)
- Sans avoir à tenir compte de spécificité d’OS ou de versions (**auto-suffisance**)
- Avec le moins d’opération manuelle
- De manière programmatique (**Infrastructure as Code**) voire versionnée (**GitOps**)

Déployer un applicatif sur 100 serveurs et 4 environnements différents ne devrait pas demander plus de boulot qu’un seul

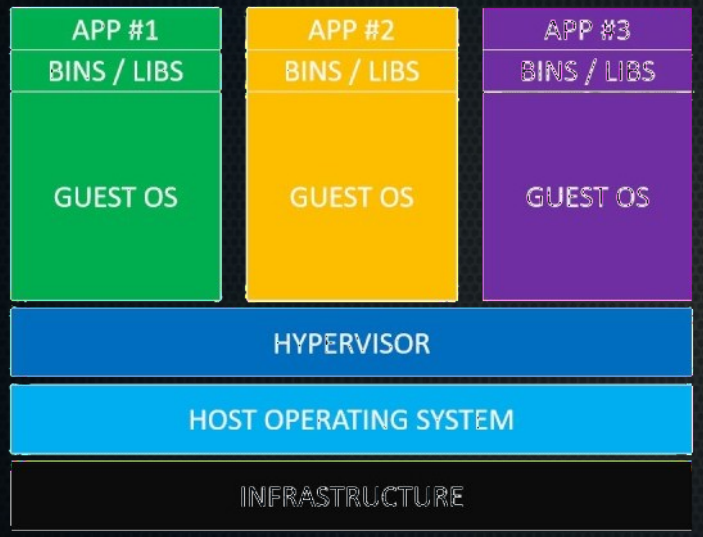
Docker: une solution de virtualisation applicative

Principe général:

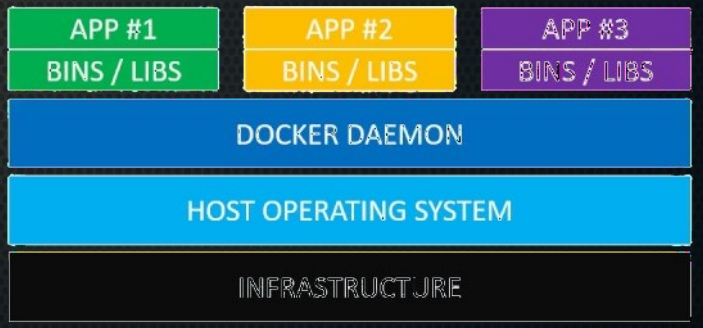
- Une **image** permet d’instancier un conteneur
- Un **conteneur** est comme une micro VM ne contenant qu’un service
- Un conteneur est :
  - Auto-suffisant
  - Encapsulé
  - Ephémère



Un conteneur n’est pas une VM

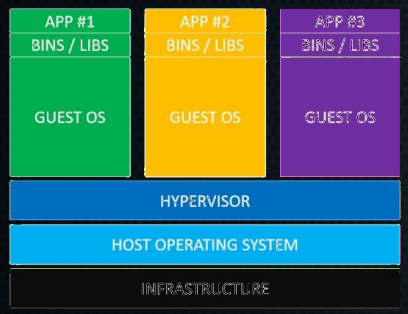


Virtual Machines

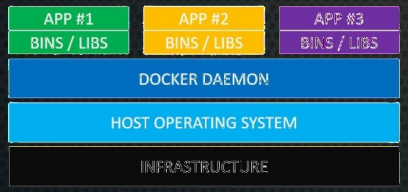


Docker Containers

Un conteneur n’est pas une VM



Virtual Machines



Docker Containers

- Images de petite taille (~ 100 Mo)
- Services léger
- Performances natives

Prise en main par l’exemple

Une stack de développement complète

Contexte fictif : vous êtes un data-scientist en mission pour un client. Il vous est demandé de produire des dashboard Jupyter d'analyse. Votre code doit être capable d'extraire les données des bases du client. Vous êtes confiné chez vous et n'avez aucun accès au SI

Stack du client :

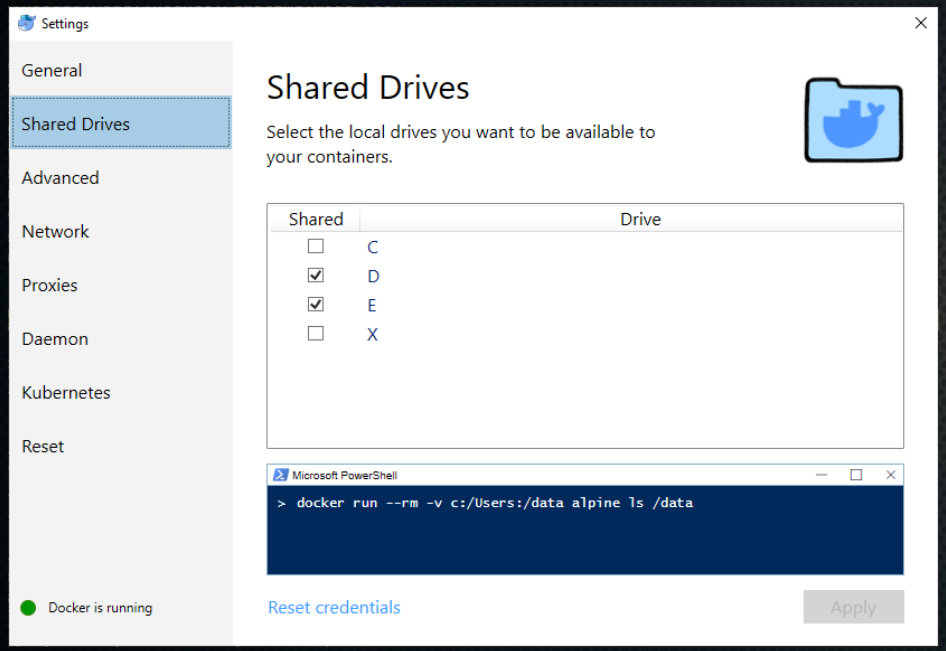
- MSSQL v 2017
- Postgresql 11.4
- Python3.8 /

Une stack de développement complète

Vous pourriez installer l’ensemble des dépendances sur une machine de développement (avec tous les emmerdements que cela implique)

Ou vous pourriez vous contenter de vous servir d’un environnement provisionné partagé par votre collègue DevOps
https://gitlab.com/d0ck3r-stacks/data-science-stack

[Win] Paramétrage des disques partagés



Outre **docker desktop**, sur Windows, vous aurez besoin de rendre vos disques accessibles pour le reste de la démo

Un tour du propriétaire

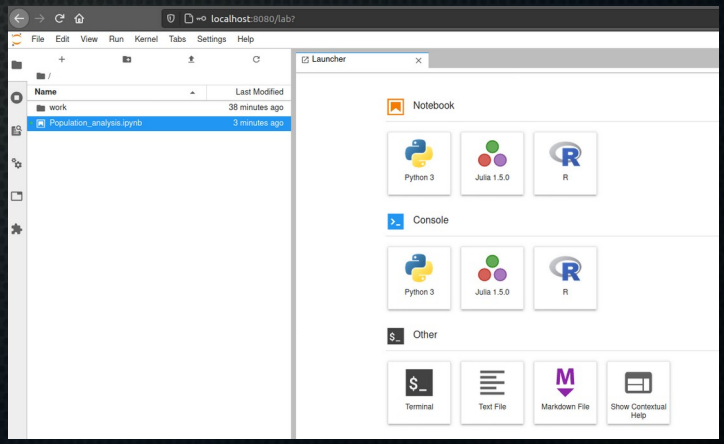
Lancer la stack

docker-compose up

Déduisez en l'endroit où vous rendre pour accéder au serveur Jupyter

Remarquez les conteneur et le mapping avec les port du host

docker container ls



Un conteneur est un environnement d’exécution distinct

Pour s’en assurer, ouvrons une session **bash** dans l’un d’eux

```
docker exec -it notebook bash
```

Où suis-je ? : pwd

Qui suis-je ? : whoami

Qu’est-ce qui tourne ? : ps

Les noms de domaines des autres conteneurs sont-ils résolus ? : curl some_postgres -v

Éphémérité et volumes

Nous avons dit précédemment qu'un conteneur est éphémère. Comment donc assurer la permanence de certaines données altérée / modifiée par l'applicatif?

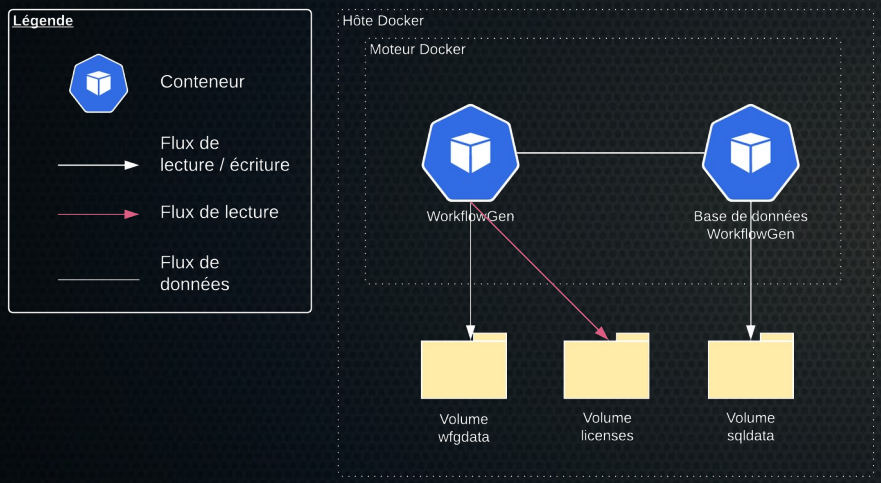
volumes:

-./notebook/notebooks:/home/jovyan/work

Mettez en évidence le mécanisme en créant

- Un notebook qui survivra à un redémarrage
- Un notebook qui sera perdu

Éphémérité et volumes

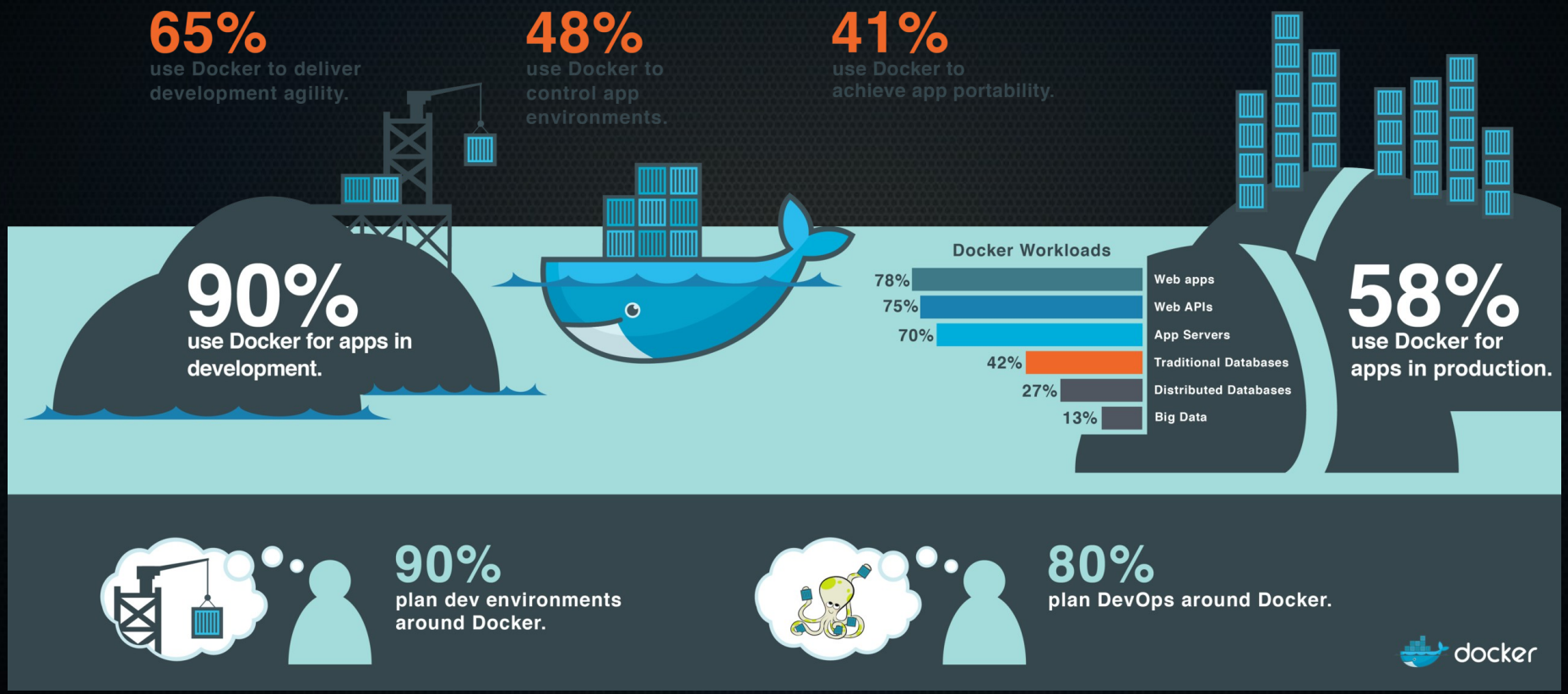


Les volumes permettent :

- D’« injecter » des fichiers de configuration
- D’assurer la permanence de ce qui doit subsister (et seulement cela)
- De partager des fichiers entre plusieurs conteneurs (e.g. certificats)

Docker en production : prégnance dans l’IT d’aujourd’hui et de demain

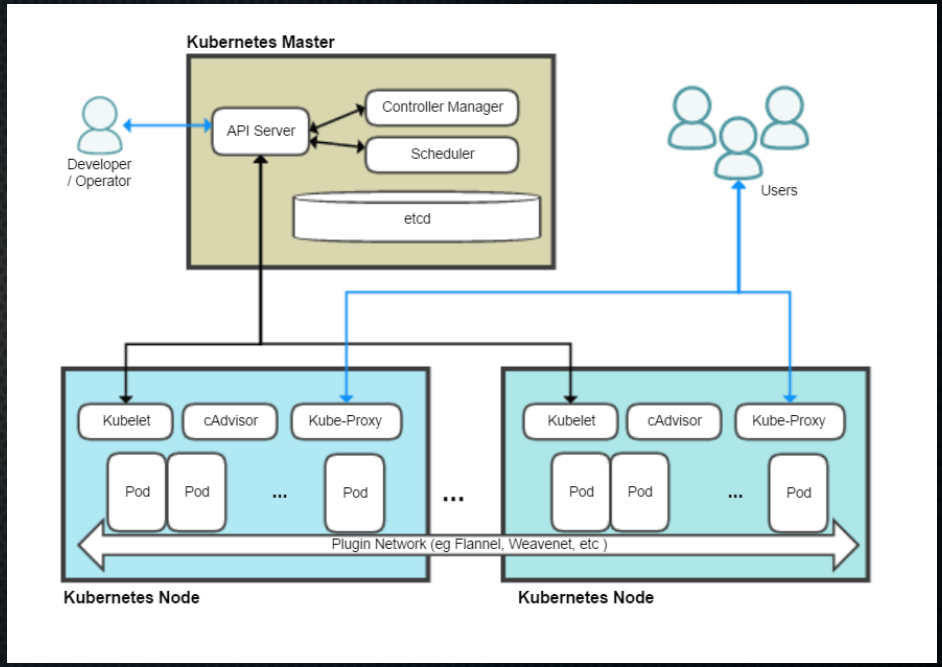
Une récente étude [source]



Etat de l’Art : Kubernetes (k8s)

Le déploiement de services dockerisés peut se faire sur un cluster de serveurs. K8s gère de manière centralisé :

- La réplication
- Failover
- Load-balancing
- Permanence
- Monitoring des services
- Montée de version

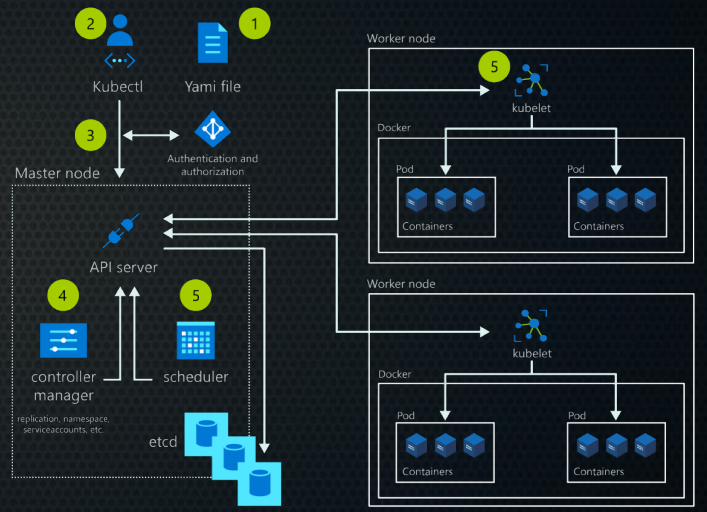


Etat de l’Art : Kubernetes (k8s)

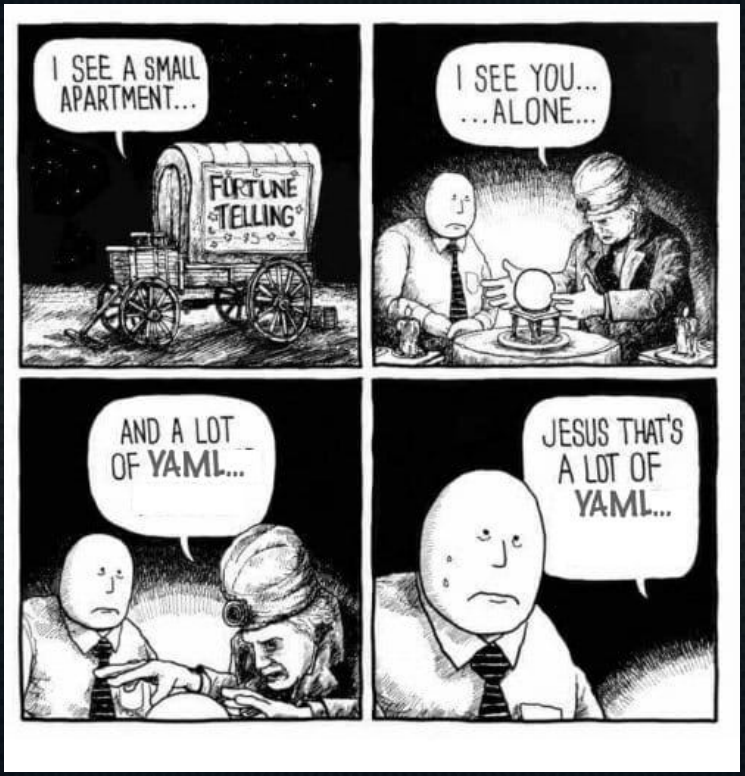
Infrastructure as Code

Une infra (ensemble d’applicatifs interconnectés) se code à l’aide d’un langage déclaratif garant de :

- sa reproductibilité
- Sa migration possible dans une autre infra



Fin


I SEE A SMALL APARTMENT...
FORTUNE TELLING
I SEE YOU...
...ALONE...
AND A LOT OF YAML...
JESUS THAT'S A LOT OF YAML...