

Docker – Présentation

Tarik NACEF

tarhack@gmail.com



Table des matières

[Docker Présentation](#)

[Docker Installation](#)

[Docker Commandes](#)

[Docker commande run](#)

[Docker – Desktop](#)

[Docker - Configuration](#)

[Docker – Construire une image](#)

[Docker – Partage d'image](#)

[Docker – Modifier une image](#)

[Docker – Persistance des données](#)

[Docker – Registry](#)

[Docker – Application multi-conteneurs](#)

[Docker – Composition de conteneurs](#)

[Docker – docker swarm](#)

[Docker – Orchestration](#)

[Docker Sauvegardes](#)



Docker Présentation

Qu'est-ce que Docker ?



Docker est une plateforme permettant aux développeurs et aux administrateurs système de développer, déployer et exécuter des applications en utilisant des **conteneurs**.

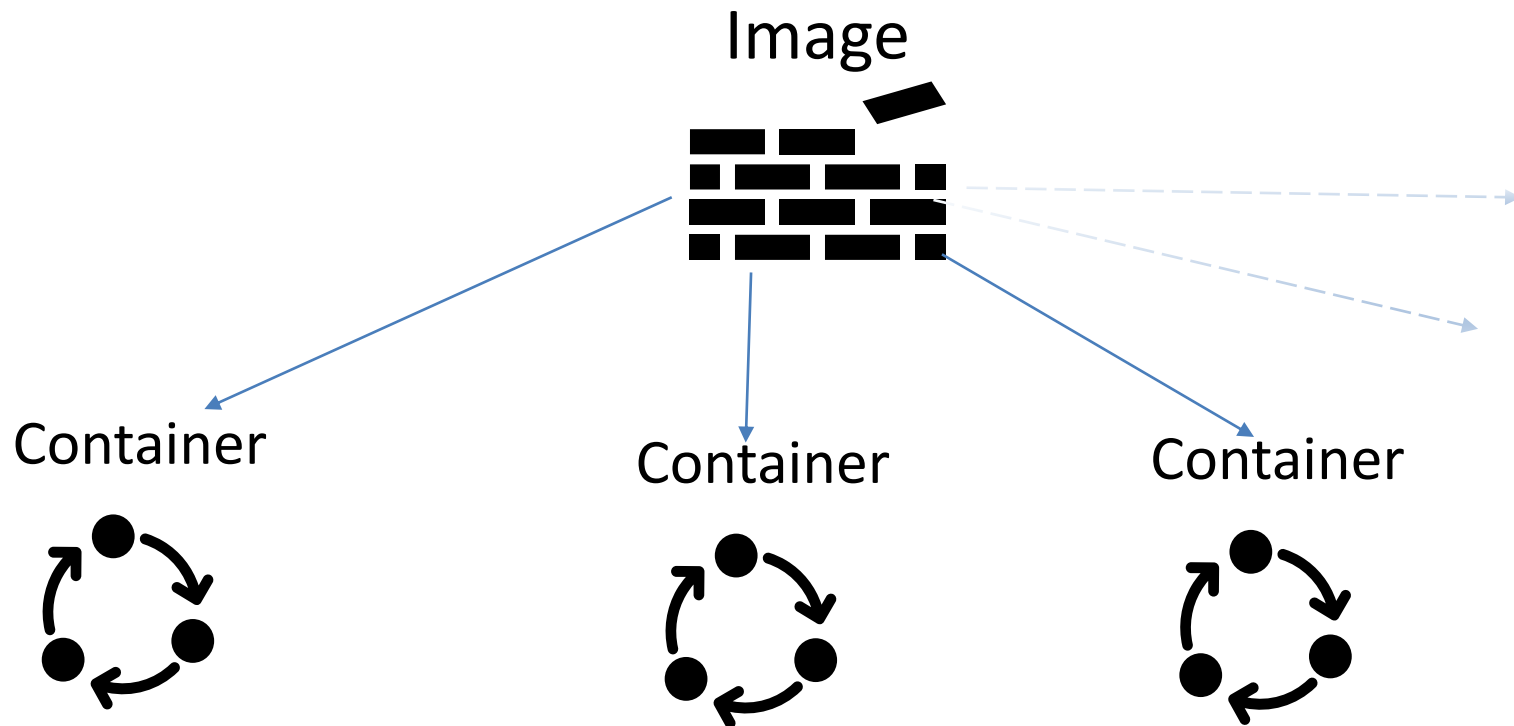
L'utilisation de conteneurs ***Linux/Windows*** pour déployer des applications s'appelle la « *containerisation* » .

« ***Containerization*** »



Docker Présentation

La conteneurisation





Docker Présentation

La conteneurisation

La conteneurisation est de plus en plus populaire car les conteneurs sont:

Flexibles : même les applications les plus complexes peuvent être conteneurisées.

Légers : les conteneurs exploitent et partagent le noyau hôte, ce qui les rend beaucoup plus efficaces en termes de ressources système que les machines virtuelles.

Portables : vous pouvez créer localement, déployer sur le cloud et exécuter n'importe où.

Faiblement couplés : les conteneurs sont hautement autonomes et encapsulés, ce qui vous permet de remplacer ou de mettre à niveau l'un sans perturber les autres.

Évolutifs : vous pouvez augmenter et distribuer automatiquement les répliques de conteneurs dans un centre de données.

Sécurisés : les conteneurs appliquent des contraintes et des isollements agressifs aux processus sans aucune configuration requise de la part de l'utilisateur.



Docker Présentation

La conteneurisation

Un conteneur s'exécute nativement sur Linux et partage le noyau de la machine hôte avec d'autres conteneurs. Il exécute un processus discret, ne prenant pas plus de mémoire que tout autre exécutable, ce qui le rend léger.

Vs VM

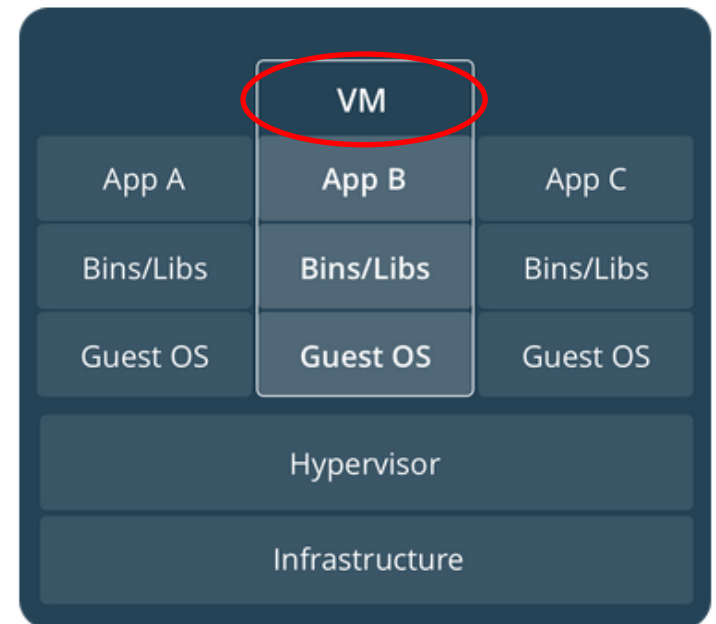
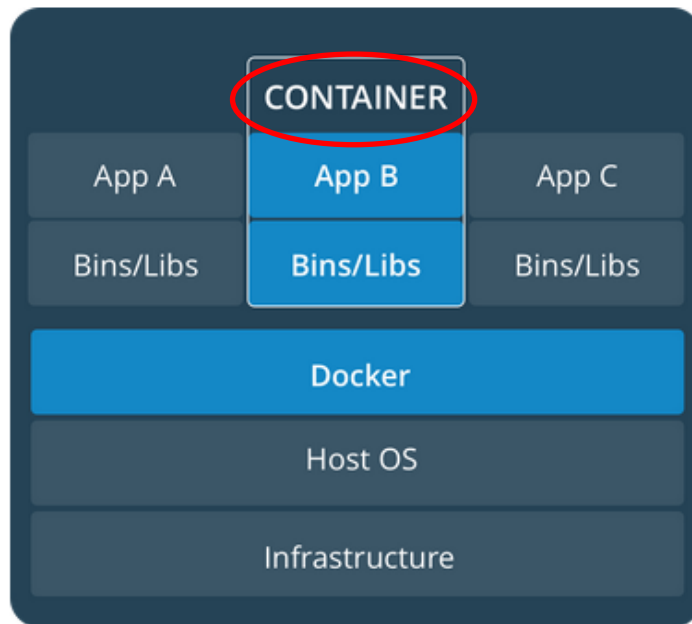
Une machine virtuelle (VM) exécute un système d'exploitation «invité» à part entière avec un accès virtuel aux ressources hôte via un hyperviseur.

Demande beaucoup de ressources(machines, hyperviseurs,...)



Docker Présentation

La conteneurisation





Docker Présentation

La conteneurisation

Un **conteneur** n'est rien d'autre qu'un processus en cours d'exécution avec quelques fonctionnalités d'encapsulation supplémentaires.

Une **image** comprend tout ce qui est nécessaire pour exécuter une application - le code ou le binaire, les environnements d'exécution, les dépendances et tout autre objet du système de fichiers requis.



Docker Installation

Installation de Docker – prérequis



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

Prérequis :

- 64 bit processor with [Second Level Address Translation \(SLAT\)](#)
- 4GB system RAM
- BIOS-level hardware virtualization support must be enabled

Attention les machines Virtuelles risquent de ne plus fonctionner !



Docker Installation

Installation de Docker – prérequis

<https://docs.docker.com/engine/install/>

installation. Find your preferred operating system below.

DESKTOP

Platform	x86_64 / amd64
Docker Desktop for Mac (macOS)	✓
Docker Desktop for Windows	✓

SERVER

Docker provides `.deb` and `.rpm` packages from the following Linux distributions and architectures:

Platform	x86_64 / amd64	ARM	ARM64 / AARCH64
CentOS	✓		✓
Debian	✓	✓	✓
Fedora	✓		✓
Raspbian		✓	✓
Ubuntu	✓	✓	✓



Docker Installation

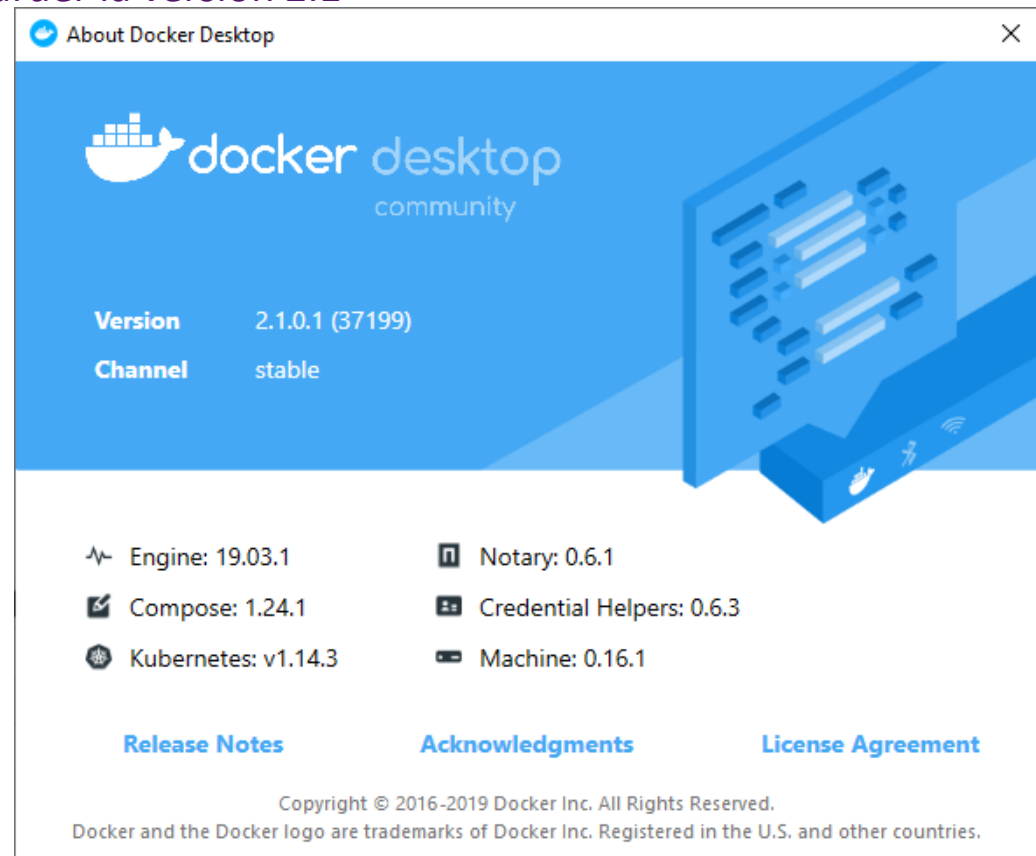
Installation de Docker – prérequis

Attention si Windows 10 < 1903 alors garder la version 2.1

Spécifications de Windows

Édition	Windows 10 Professionnel
Version	1909
Installé le	12/06/2020
Version du système d'exploitation	18363.1082
Numéro de série	R90KTSPJR9N0B661100Q
Mettre à niveau votre édition de Windows ou modifier la clé de produit (Product Key)	

[Lisez le Contrat de services Microsoft qui s'applique à nos services](#)





Docker Installation

Installation de Docker – installation

<https://docs.docker.com/get-docker/>



Docker Desktop for Mac

A native application using the macOS sandbox security model which delivers all Docker tools to your Mac.



Docker Desktop for Windows

A native Windows application which delivers all Docker tools to your Windows computer.



Docker for Linux

Install Docker on a computer which already has a Linux distribution installed.



Docker TP : Installation

TP - Installation de Docker





Docker Présentation

Installation de Docker

En ligne de commande : CMD.exe, vérification de l'installation

```
C:\Users\Developeurs>docker --version
Docker version 19.03.12, build 48a66213fe
C:\Users\Developeur>
```



Docker Commandes

Mode commande



docker

docker [management type] option

docker image ls -> liste les images du système

ou

docker images



Docker Commandes

Gestion des images

`docker images (docker image -ls)`

- liste toutes les images du système

`docker image rm [image-name|image-id]`

- remove une image

`docker run -it [image-name|image-id]`

- exécute une image en interactif, relié à un terminal



Docker Commandes

Gestion des containers

`docker container stop [container name | container Id]`

arrête un conteneur

`docker container start [container name | container Id]`

relance un conteneur

`docker container rm [container name | container Id]`

supprime un conteneur

`docker container prune`

supprime tous les conteneurs arrêtés



Docker Commandes

Docker commandes

docker ps : permet de connaitre les conteneurs en exécution

docker exec -it [container Id] command

docker exec -it gitlab-runner bash

Permet d'entrer en mode commande (bash) dans un conteneur en exécution



Docker Commandes

Docker commandes

docker ps : permet de connaitre les conteneurs en exécution



docker ps --all



docker container ls --all



Docker Commandes

Docker commandes

Exécuter une image

docker run [OPTIONS] IMAGE [COMMAND] [ARG...]

<https://docs.docker.com/engine/reference/commandline/run/>



Docker – Commandes

Donner un nom à un conteneur

`docker run --name started -d -p 80:80 docker/getting-started`

Permet

`docker stop | start [container]`

`docker stop started`

`docker start started`



Docker – Commande run



La commande docker run possède de nombreux paramètres et options : (cf. ***docker run --help***)

Options d'exposition de ressources, on dit aussi qu'on mappe des ressources :

Les ports : ***-p [port du host]:[port du container]***

Les volumes : ***-v [repertoire host]:[repertoire container]***

Variables environnement : ***-e DEBUG_LEVEL = OFF***



Docker – Commande run

Exemples :

```
docker run -d --name started -p 82:80 docker/getting-started
```

- On mappe le port **82** du **host** avec le port **80** du **container**

```
docker run -d -v D:/docker-volumes/getting-started/html:/usr/share/nginx/html
```

- On mappe le repertoire **D:/docker-volumes/getting-started/html** du Host avec le repertoire **/usr/share/nginx/html** du container

Attention au : qui est l'operateur de mapping



Docker TP - Commandes

Lancer l'exécution d'une image

(`docker run -d -p 80:80 docker/getting-started`)





Docker - Desktop



Docker Desktop

The screenshot shows the Docker Desktop interface. The top bar is blue with the Docker logo, settings, and user profile (tarhack). The main area is titled 'Images on disk' and shows a list of images. A red circle highlights the 'Dashboard' button in the top right corner.

Containers / Apps
Images

Images on disk 26 images Total size: 5.28 GB IN USE UNUSED Clean up...

LOCAL REMOTE REPOSITORIES

Search Sort by

	TAG	IMAGE ID	CREATED	SIZE
ubuntu-apache	latest	58b367cadea7	1 day ago	275.15 MB
localhost:5000/ubuntu-...	1.1	58b367cadea7	1 day ago	275.15 MB
localhost:5000/ubuntu-...	1.0	246d9cfa31f9	1 day ago	274.54 MB
localhost:5000/bulletin-...	<none>	ba1a6b9cc8c9	9 days ago	1.01 GB
localhost:5000/bulletin-...	<none>	66775c20d062	10 days ago	108.94 MB
todo-app-server-mysql	1.0	e47bc3cfa3db	12 days ago	467.77 MB
todo-app-mysql	1.0	49be3d031eee	12 days ago	220.31 MB
ubuntu-...	latest	47b...f3b77...	15 days ago	73.88 MB

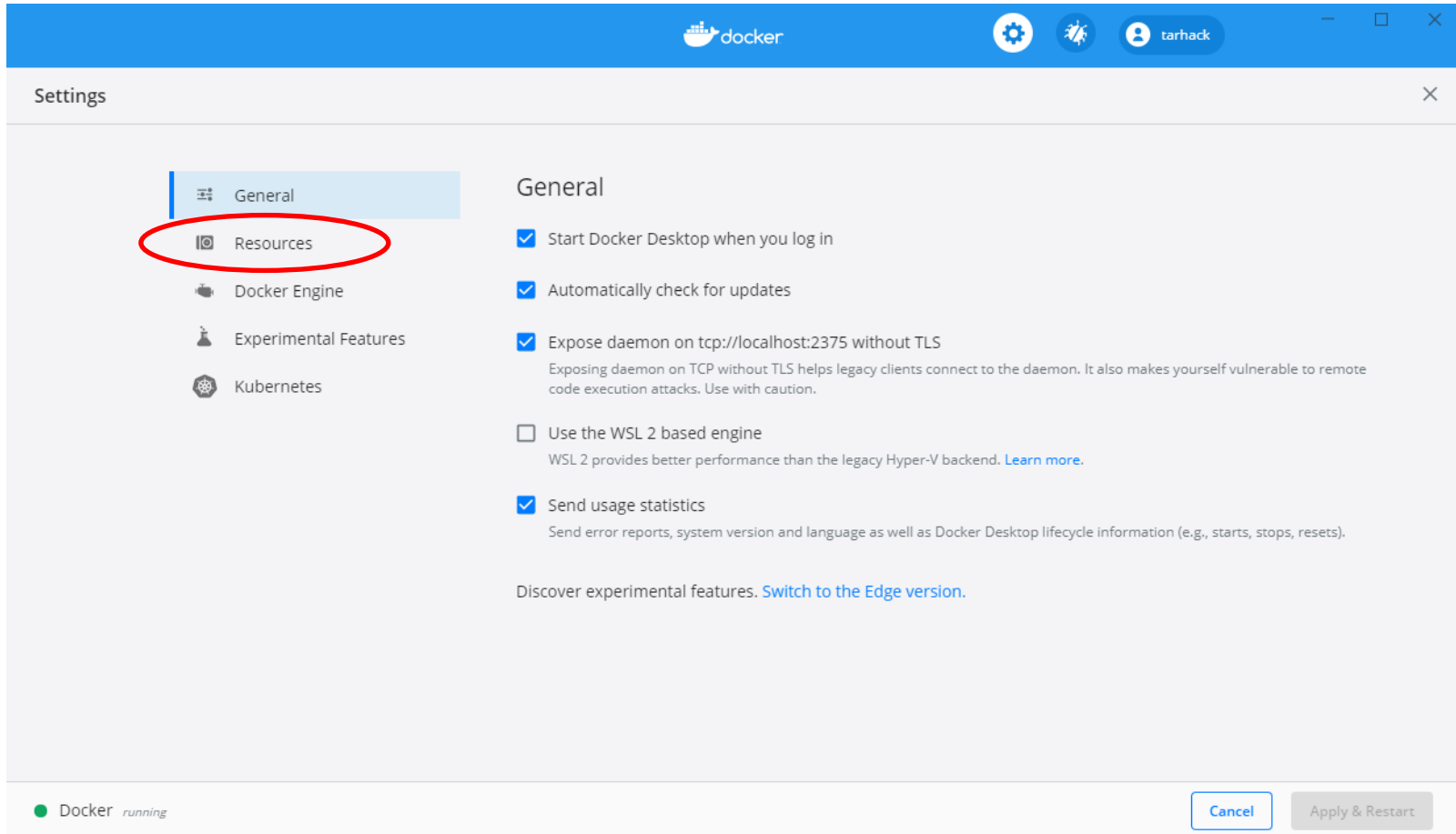
Dashboard
Settings
Check for Updates
Troubleshoot
Switch to Windows containers...
About Docker Desktop
Documentation
Quick Start Guide
Docker Hub
tarhack
Kubernetes
Restart...
Quit Docker Desktop

Docker running



Docker - Desktop

Docker Desktop





Docker – Configuration

Docker configuration - WSL



<https://docs.microsoft.com/en-us/windows/wsl/wsl-config#configure-global-options-with-wslconfig>

On paramétrer les réglages de bases concernant les ressources en utilisant le fichier ***.wslconfig***



Docker – Configuration

Docker configuration - Windows





Docker – Construire une image

Docker file



Le fichier Docker file permet d'utiliser des instructions Docker afin de construire une image.

Il sera principalement utilisé par la commande :
`docker build`



Docker – Construire une image

Docker file

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

Exemple d'instructions utilisées dans docker file

FROM

WORKDIR

COPY

CMD

...



Docker – Construire une image

Docker file

FROM : indique à Docker que l'image sera construite « A partir » de l'image indiquée en paramètre.

WORKDIR : Définit le répertoire de travail pour toutes les instructions RUN, CMD, ENTRYPOINT, COPY et ADD qui le suivent dans le Dockerfile.

COPY : Permet de copier des fichiers depuis le répertoire courant du host vers le filesystem de l'image en construction.

On pourra indiquer des droits spécifiques pour les images Linux.



Docker – Construire une image

Docker file

RUN : cette commande permet d'exécuter des commandes d'O.S de l'image en cours de construction. Elle peut prendre 2 formes dans sa syntaxe : shell ou exec forme

- RUN /bin/bash -c 'echo hello' (shell form)
- RUN ["/bin/bash", "-c", "echo hello"] (exec form)

Exemple :

...

RUN (apt-get update && apt-get install -y firefox)



Docker – Construire une image

Docker file

RUN : cette commande permet d'exécuter des commandes d'O.S de l'image en cours de construction. Elle peut prendre 2 formes dans sa syntaxe : shell ou exec forme

- RUN /bin/bash -c 'echo hello' (shell form)
- RUN ["/bin/bash", "-c", "echo hello"] (exec form)

Example :

...

RUN (apt-get update && apt-get install -y vim)



Docker – Construire une image

Docker file

CMD : permet de définir l'exécution par default d'un conteneur
Il ne peut y avoir qu'une seule commande CMD par Docker file.
Cette commande peut prendre 3 formes :

- `CMD ["executable","param1","param2"]` (exec form, this is the preferred form)
- `CMD ["param1","param2"]` (as default parameters to *ENTRYPOINT*)
- `CMD command param1 param2` (shell form)

Exemple :

`CMD ["top", "-b"]`



Docker – Construire une image

Docker file

ENTRYPOINT : Permet de configurer un conteneur qui s'exécutera en tant qu'exécutable.

La syntaxe peut prendre 2 formes :

ENTRYPOINT ["executable", "param1", "param2"] -> exec form

ENTRYPOINT command param1 param2 -> shell form

Exemple

ENTRYPOINT ["top", "-b"]



Docker – Construire une image

Build image

On construit une image depuis le docker file à l'aide de la commande : ***docker build***

docker build -f [nom_du fichier docker file] ***-t*** [nom de l'image[:tag]] .

Ou (Si le fichier s'appelle Dockerfile)

docker build -t [nom de l'image] .



Docker TP - Build

Exercice : créez votre première image

Créez une image Ubuntu hébergeant un serveur Apache





Docker – Partage d'image

Partager une image



- 1/ Créer un compte sur Docker Hub
- 2/ Créer un repository
- 3/ Tagger l'image correctement
- 4/ Déposer l'image sur le repository (push)



Docker – Partage d'image

Créer un compte Docker

A screenshot of the Docker Hub website's sign-up page. The page has a blue header with the Docker Hub logo, a search bar, and navigation links for "Explore", "Pricing", "Sign In", and "Sign Up". The main content area has a blue background with the text "Build and Ship any Application Anywhere" and a description of Docker Hub. On the right, there is a white sign-up form titled "Sign Up Today". The form includes a link for "Already have an account? Sign In", input fields for "Docker ID" (labeled "User name"), "Email" (labeled "email"), and "Password" (labeled "MdP"), a checkbox for "Send me occasional product updates and announcements.", a checkbox for "Je ne suis pas un robot" with a reCAPTCHA logo, and a blue "Sign Up" button. At the bottom of the form, it states "By creating an account, you agree to the Terms of Service, Privacy Policy, and Data Processing Terms."/>

Build and Ship any Application Anywhere

Docker Hub is the world's easiest way to create, manage, and deliver your teams' container applications.

Sign Up Today

Already have an account? [Sign In](#)

Docker ID **User name**

Email **email**

Password **MdP**

☐ Send me occasional product updates and announcements.

☐ Je ne suis pas un robot

Sign Up

By creating an account, you agree to the [Terms of Service](#), [Privacy Policy](#), and [Data Processing Terms](#).

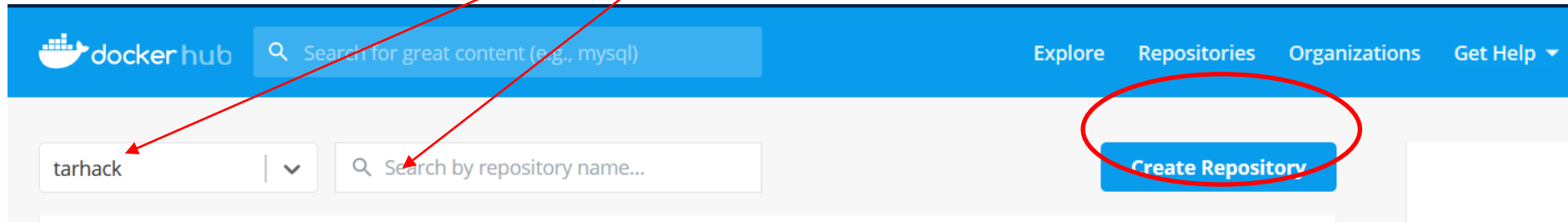


Docker – Partage d'image

Docker Hub : Créer un repository

Vous retrouvez ici le nom de votre ID Docker

Entrez le nom de votre repository





Docker – Partage d'image

Tagger l'image

On peut renommer une image à l'aide de la commande
docker tag

docker tag [image_name] [new image_name:tag]

docker tag ubuntu-apache tarhack/ubuntu-apache:1.0



Docker – Partage d'image

Push de l'image

La commande permettant d'envoyer l'image sur le repo. Docker Hub est `docker push`

```
docker push [user_name/image_name:tag]
```

Ou le ***user_name*** = le user créé sur Docker Hub

```
docker push tarhack/ubuntu-apache:1.0
```



Docker – Partage d'image

Push de l'image

Attention, AVANT le push il faudra exécuter une commande :

docker login -u [user name docker]

Et entrez le **MdP** de votre compte Docker



Docker – Modifier une image

Modifier une image



Il arrive que l'on soit obligé de modifier une image suite à une évolution et/ou à une correction.

Plusieurs cas peuvent se présenter

- Correction de l'image via le Dockerfile
- Correction à chaud dans le conteneur (* attention)



Docker – Modifier une image

Modification du conteneur

On peut modifier un conteneur en exécution et ensuite reporter ces modification sur une image

1/ d'abord on entrera dans le conteneur en mode commande

docker exec -it apache bash

2/ On pourra effectuer les modification souhaitées

3/ docker commit



Docker – Modifier une image

Modification du conteneur

On peut modifier un conteneur en exécution et ensuite reporter ces modification sur une image

1/ d'abord on entrera dans le conteneur en mode commande

docker exec -it apache bash

2/ On pourra effectuer les modification souhaitées

3/ docker commit

4/ Dans ce cas les bonnes pratiques imposent de reporter la modification dans le Dockerfile correspondant à l'image modifiée.



Docker – Modifier une image

Modifier une image

On peut modifier l'image directement dans le répertoire de développement de l'application que l'on souhaite modifier.

Exemple dans l'application Web : todo-app

Et relancer le build de l'image à partir du Dockerfile



Docker – Modifier une image

Docker – modification image

Changer le code de *todo-app* afficher une bannière





Docker – commandes images

Quelques commandes concernant les images :

Command	Description
<code>docker image build</code>	Build an image from a Dockerfile
<code>docker image history</code>	Show the history of an image
<code>docker image import</code>	Import the contents from a tarball to create a filesystem image
<code>docker image inspect</code>	Display detailed information on one or more images
<code>docker image load</code>	Load an image from a tar archive or STDIN
<code>docker image ls</code>	List images
<code>docker image prune</code>	Remove unused images
<code>docker image pull</code>	Pull an image or a repository from a registry
<code>docker image push</code>	Push an image or a repository to a registry
<code>docker image rm</code>	Remove one or more images
<code>docker image save</code>	Save one or more images to a tar archive (streamed to STDOUT by default)
<code>docker image tag</code>	Create a tag TARGET_IMAGE that refers to SOURCE_IMAGE



Docker – Persistance des données

Docker Filesystem



Chaque conteneur dispose de son propre «espace de travail» pour créer, mettre à jour ou supprimer des fichiers.

Les modifications ne seront pas visibles dans un autre conteneur, même si elles utilisent la même image.



Docker – Persistance des données

Docker Volumes

Les volumes permettent de relier des chemins de système de fichiers spécifiques du conteneur à la machine hôte.

Si un répertoire dans le conteneur est monté, les modifications dans ce répertoire sont également visibles sur la machine hôte.

Si vous montez ce même répertoire sur les redémarrages de conteneurs, vous verrez les mêmes fichiers.

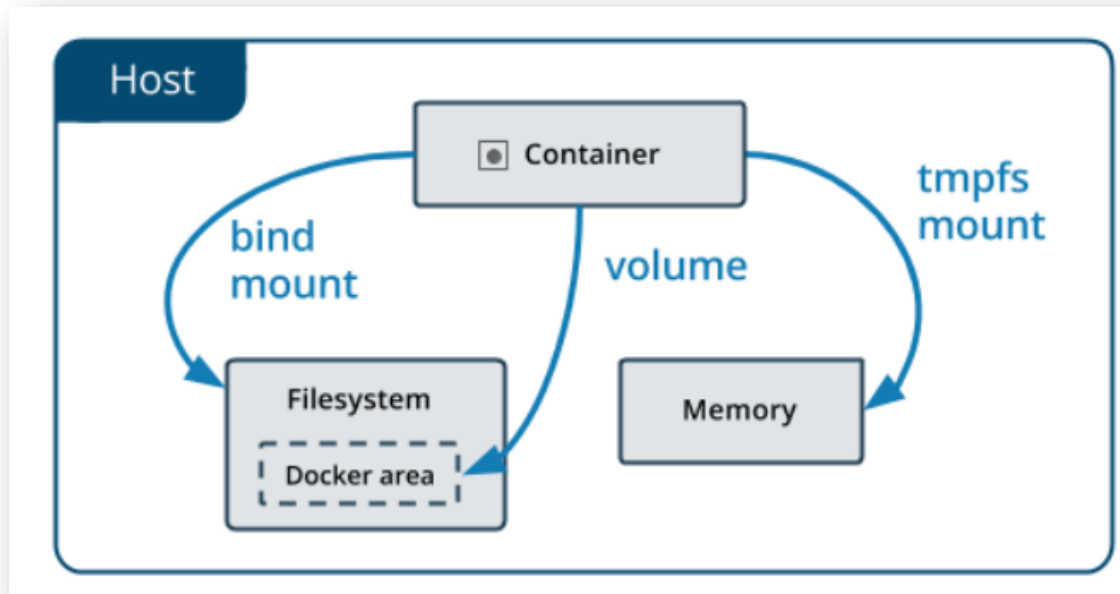


Docker – Persistance des données

Docker – Type de montages

Il existe 3 manières de monter des répertoires de data afin de persister des données.

volume, bind mount, et tmpfs mount





Docker – Persistance des données

Docker – Type de montages

Volumes : Ils sont stockés dans une partie du système de fichiers hôte qui est ***géré par Docker*** (/ var/lib/docker/volumes/ sous Linux). Les processus non-Docker ne doivent pas modifier cette partie du système de fichiers. Les volumes sont le meilleur moyen de conserver les données dans Docker.

Bind mounts : (***option -v*** de la commande docker run) Ils peuvent être stockés n'importe où sur le système hôte. Les processus non-Docker sur l'hôte Docker ou un conteneur Docker peuvent les modifier à tout moment.

Mount tmpfs : Ils sont stockés uniquement dans la mémoire du système hôte et ne sont jamais écrits sur le système de fichiers du système hôte.



Docker – Persistance des données

Docker – Volumes

https://docs.docker.com/engine/reference/commandline/volume_create/

Déclaration d'un volume :

docker volume create [nom-du-volume]

docker volume ls

liste les volumes actifs

docker volume inspect [nom-du-volume]

donne des détails sur le volume



Docker – Persistance des données

Docker - Volumes

L'utilisation d'un volume se fera au moment du run en associant un répertoire de l'application avec le nom du volume :

```
docker run -d --name todo -p3000:3000 -v todo-db : /etc/sqlite/todos todo-app
```

Nom du volume

nom du répertoire dans le conteneur



Docker – Persistance des données

Docker – bind mount

Le ***bind mount*** permet d'associer un répertoire de la machine hôte à un répertoire du conteneur

L'association se fera au moment du run de l'image :

```
docker run -d --name todo -v D:\docker-volumes\shared\todo-db : /etc/sqlite/todos  
todo-app
```




Docker – Persistance des données

Docker – tmpfs mount

Le montage ***tmpfs mount*** permet de monter des fichiers en mémoire, la mémoire du host.

Contrairement aux ***volumes*** et aux ***bind mount***, un montage tmpfs est temporaire et ne persiste que dans la mémoire hôte.

Lorsque le conteneur s'arrête, le montage tmpfs est supprimé et les fichiers qui y sont écrits ne sont pas conservés.



Docker – Persistance des données

Docker - Filesystem

Quelques manipulation sur les fichiers de conteneurs





Docker – Registry

Docker – Utilisation d'une Registry



<https://docs.docker.com/registry/configuration/>

Certaines organisations voudront contrôler étroitement ***où*** les images sont stockées et ***comment*** elle sont stockées

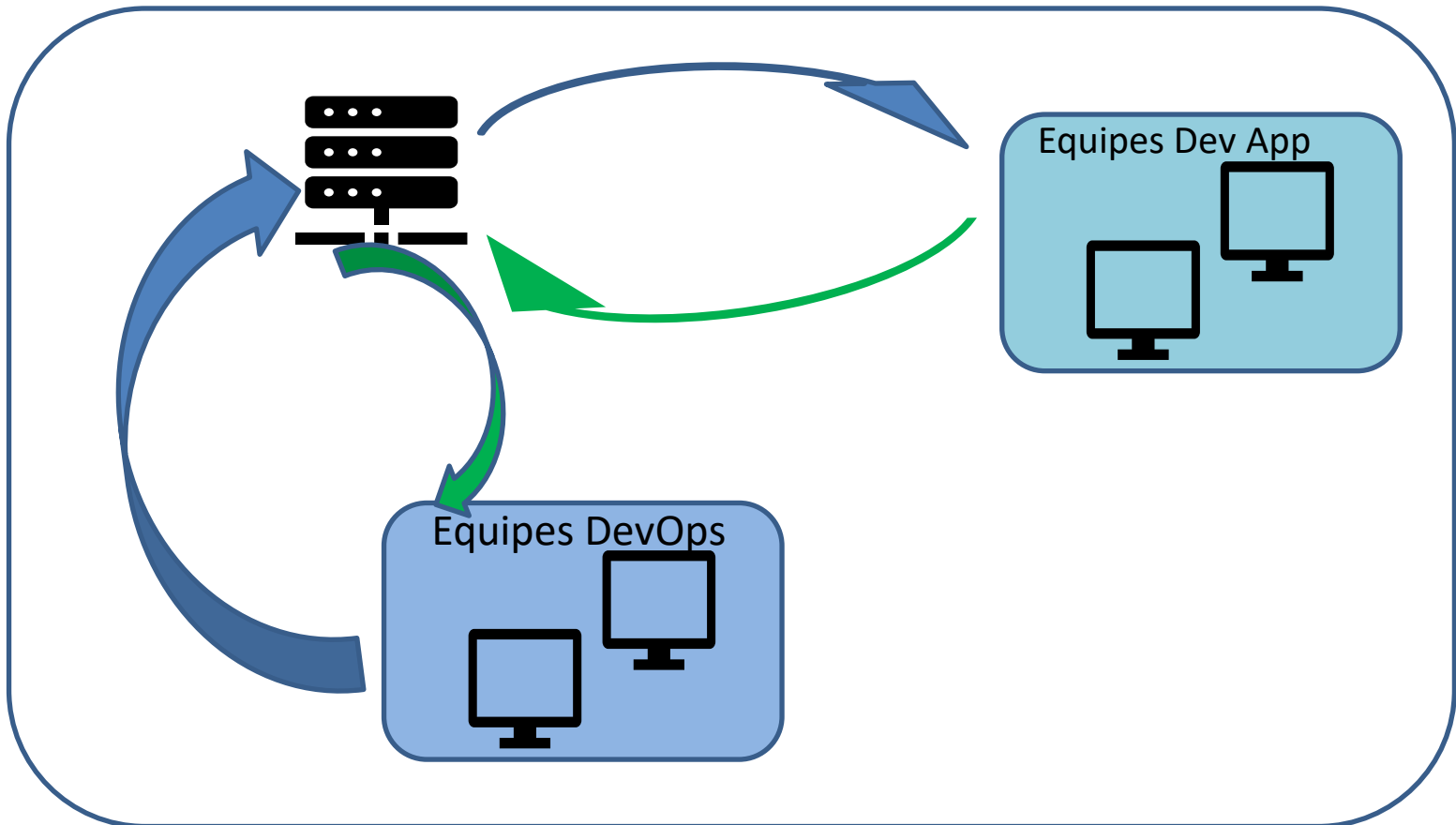
Intégrez étroitement le stockage et la distribution d'images dans un workflow de développement interne

Le contenu des images peut être confidentiel et dans ce cas on pourra se servir de la fonction ***registry*** de Docker qui permettra de créer un repository ***privé et local*** d'images.



Docker – Registry

Utilisation d'une Registry - Schéma





Docker – Registry

Installation d'une Registry

Installation de la Registry sur un serveur

```
docker run -d -p 5000:5000 --name registry registry:2
```

```
docker start registry
```



Docker – Registry

Utilisation d'une Registry

1/ On tague l'image afin qu'elle adresse la registry

docker image tag ubuntu localhost:5000/ubuntu:1.0

2/ On tague l'image afin qu'elle adresse la registry

docker image tag ubuntu localhost:5000/ubuntu:1.0

3/ dépôt de l'image sur la registry

docker push localhost:5000/ubuntu:1.0



Docker – Registry

Utilisation d'une Registry

5/ Réutilisation de l'image

docker pull localhost:5000/ubuntu:1.0

Dans le cas qui nous intéresse localhost pourrait-être un serveur de l'infrastructure, et le port peut être changé



Docker – Registry

Configuration d'une Registry

1/ Fichier de configuration YAML

pour les installations poussées (DevOps)

On peut changer la résidence des images dans la registry local en modifiant la commande

docker run ... -v [host/path:conteneur/path] ...

On peut changer le port d'écoute en modifiant la commande

docker run ...-p host.port:conteneur.port ...

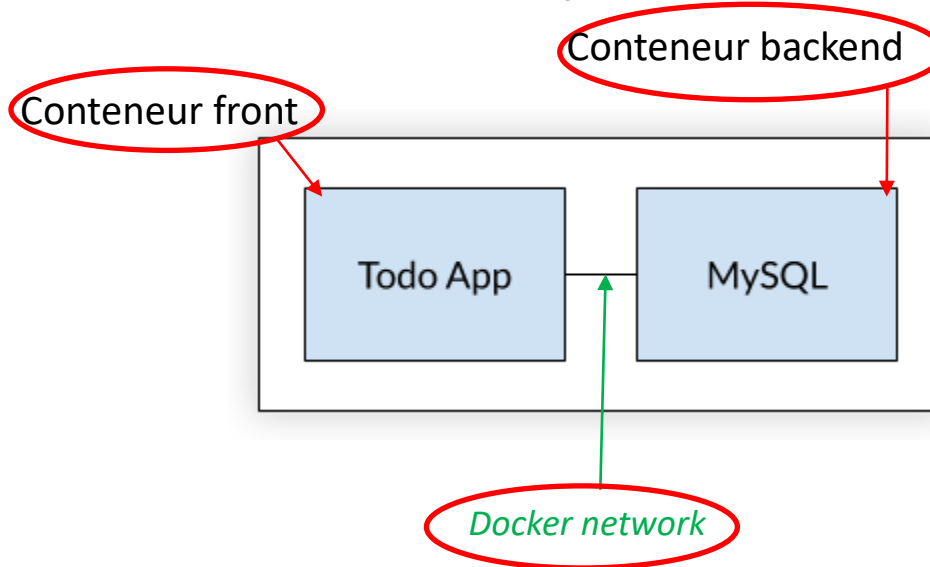


Docker – Application multi-conteneurs

Docker compose



Le découpage des applications en plusieurs services, chacun isolé dans un conteneur, communiquant par le réseau, est une architecture très utilisée de nos jours et Docker facilite sa mise en œuvre.





Docker – Application multi-conteneurs

Docker compose

Compose est un outil permettant de définir et d'exécuter des applications Docker multi-conteneurs.

Avec Compose, vous utilisez un fichier YAML pour configurer les services de votre application.

Ensuite, avec une seule commande, vous créez et démarrez tous les services à partir de votre configuration

`docker-compose up`



Docker – Application multi-conteneurs

Configuration de l'application

1/ Installer 2 conteneurs (front, backend)

On définira chacune des images à l'aide de Dockerfile

2/ Configurer le réseau pour les conteneurs

On créera un réseau afin que les conteneurs puissent parler entre eux.



Docker – Application multi-conteneurs

Multi-conteneurs - réseau

Le réseau Docker est un sous-système basé sur l'utilisation de drivers enfichables (pluggable).

Le type de driver est défini lors de la création du réseau.

Il existe de nombreux drivers prédéfinis : bridge, host, overlay, macvlan, none, network plugin.

<https://docs.docker.com/network/>

Par défaut c'est le driver **bridge** qui sera utilisé.



Docker – Application multi-conteneurs

Multi-conteneurs - réseau

bridge: le pilote réseau par défaut.

Les réseaux type "**bridge**" sont généralement utilisés lorsque vos applications s'exécutent dans des conteneurs autonomes qui doivent communiquer.

ATTENTION : Le sous-système réseau de Docker ne fonctionne pas de la même manière sous Linux et sous Windows



Docker – Application multi-conteneurs

Multi-conteneurs - réseau

Créer un réseau :

docker network create [nom-du-réseau]



Docker – Application multi-conteneurs

Multi-conteneurs – run avec *network*

Lorsque vous avez identifié le conteneur devant être accéder via le réseau, vous le lancerez en utilisant 2 options :

- ***--network*** : permet au conteneur d'identifier son réseau
- ***--network_alias*** : permet d'associer un nom logique au conteneur au sein du réseau (nom réseau)

Exemple :

```
docker run -d --name mysql --network todo-app --network-alias mysql ^  
-e MYSQL_ROOT_PASSWORD=secret ^  
-e MYSQL_DATABASE=todos^  
mysql:5.7
```



Docker – Application multi-conteneurs

Multi-conteneurs – commandes réseau

Command		Description
<code>docker network connect</code>	<input checked="" type="checkbox"/>	Connect a container to a network
<code>docker network create</code>	<input checked="" type="checkbox"/>	Create a network
<code>docker network disconnect</code>		Disconnect a container from a network
<code>docker network inspect</code>		Display detailed information on one or more networks
<code>docker network ls</code>	<input checked="" type="checkbox"/>	List networks
<code>docker network prune</code>		Remove all unused networks
<code>docker network rm</code>		Remove one or more networks



Docker – Application multi-conteneurs

Installez l'application *todo-app-mysql*





Docker – Composition de conteneurs

Docker compose



Docker Compose est un outil qui a été développé pour aider à définir et partager des applications multi-conteneurs. Avec docker compose, nous créons à l'aide d'un **fichier YAML** une configuration de services et ensuite avec une seule commande, nous pouvons démarrer ou arrêter un ensemble de services composant notre application.

Le gros avantage de **Compose** est que vous pouvez définir votre pile d'applications dans un fichier, la conserver à la racine de votre dépôt de projet et permettre facilement à quelqu'un d'autre de contribuer à votre projet.



Docker – Composition de conteneurs

Docker compose (installation)

Normalement si vous avez installé ***Docker Desktop*** ou ***Docker Community***, ***Docker Compose*** a été installé en même temps.

Vérifiez avec la commande :

docker-compose version



Docker – Composition de conteneurs

Docker compose – création de services

Dans l'application ***todo-app-mysql*** nous allons créer un fichier nommé :

docker-compose.yml

à la racine du projet

La première instruction que l'on va mettre est la version schéma (grammaire spécifique à docker-compose/YAML)

version: "3.8"



Docker – Composition de conteneurs

Docker compose – YAML

Prononcer "***Yamel***", "***YAML Ain't Markup Language***", format :

A list of tasty fruits ← Commentaire

- Apple
- Orange
- Strawberry
- Mango

-martin:

name: Martin D'vloper
job: Developer
skill: Elite

- tabitha:

name: Tabitha Bitumen
job: Developer
skills:

- lisp
- fortran
- erlang



Docker – Composition de conteneurs

Docker compose – création

Ensuite on va définir la liste des ***services*** ↔ ***containers*** que nous voulons démarrer :

services :

app:

image: node:12-alpine

command: sh -c "yarn install --production"



Docker – docker compose

Fichier : ***docker-compose.yml***

Docker compose permet d'utiliser une image afin de créer des ***services*** qui dupliquent l'instance de cette image.

Permet de gérer facilement la ***scalabilité*** (montée en puissance)

<https://docs.docker.com/compose/reference/>



Docker – docker compose

version: "3.8" ← Version de Docker compose
services : ← Déclaration des services

```
app:
  image: node:12-alpine
  command: sh -c "yarn install && yarn run dev"
  ports:
    - 3000:3000
  working_dir: /app
  volumes:
    - ./:/app
```

Front-end (node.js)

```
# Network must be created before
environment:
  MYSQL_HOST: mysql
  MYSQL_USER: root
  MYSQL_PASSWORD: secret
  MYSQL_DB: todos
```

```
mysql:
  image: mysql:5.7
  ports:
    - 3307:3306
  volumes:
    - todo-mysql-data:/var/lib/mysql
  environment:
    MYSQL_ROOT_PASSWORD: secret
    MYSQL_DATABASE: todos
```

Back-end (mysql)

```
volumes:
  todo-mysql-data:
    external: true
```

Déclaration d'un volume : persistance des données



Docker – docker compose

commandes

docker-compose up : lance tous les services de l'application

docker-compose down : arrête tous les services & remove

docker-compose stop : arrête tous les services de l'application

docker-compose start : lance tous les services de l'application

docker-compose ps : affiche les services en exécution



Docker – docker compose

Instructions

Contraintes de dépendances :

depends_on:

- "**service**"

Cette commande permet de préciser qu'un conteneur ne doit pas être lancé tant que le service "**service**" n'est pas "Up"



Docker – docker compose

Instructions

Contraintes de dépendances : wait

<https://github.com/ufoscout/docker-compose-wait/>

Cet outil permet d'attendre sur une adresse de type host:port que le conteneur soit up avant de lancer sa propre exécution.

Exemple :

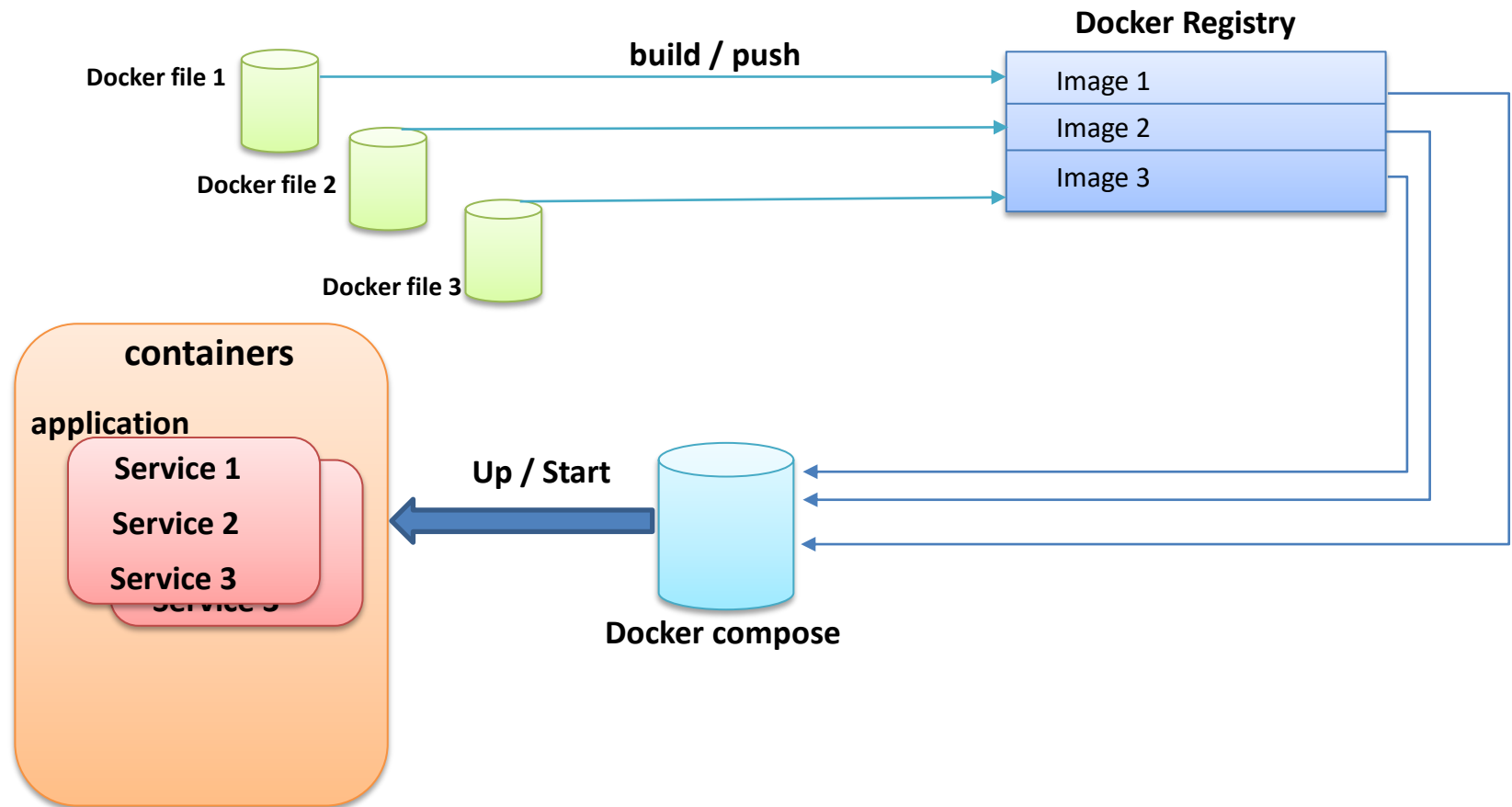
...

RUN wait && entrypoint.sh



Docker – docker compose

Récapitulatif





Docker – docker compose

Installez application "node bulletin board"





Docker – docker swarm

Docker - Orchestration



Le mode "**swarm**" (essaim) est proposé par Docker afin d'effectuer l'orchestration des containers.

Un essaim se compose de plusieurs hôtes Docker qui s'exécutent en mode essaim et agissent en tant que **manager** (gestionnaires) (pour gérer l'adhésion et la délégation) ou **workers** (travailleurs) (qui exécutent les services d'essaim).

Il existe de nombreux logiciels permettant de gérer l'orchestration de containers : Kubernetes, Docker Swarm, AWS (AWS ECS, AWS Fargate, AWS EKS) , OpenShift (RedHat), ...



Docker – Docker swarm

Docker - Orchestration

Docker propose le module "**stack**" qui permet d'aller plus loin dans la création de services et la collaboration entre les containers.

Docker stack et Docker swarm sont combinés pour faciliter l'orchestration sous Docker.



Docker – docker swarm

Docker – définition du swarm/stack

On crée un fichier YAML ***docker-stack.yml***, par défaut, dans lequel on va déclarer tous les containers utilisés.

On pourra aussi gérer des dépendances entre containers

Déclarer un réseau

Déclarer des volumes, etc...

Le ***swarm*** sera démarré à l'aide d'une command unique :

docker swarm init



Docker – Docker swarm

Docker – définition du swarm

On pourra lancer la stack après avoir initialisé le swarm avec une commande unique :

docker stack deploy -c [fichier stack yamel] [nom de la stack]



Docker – Orchestration

Docker – Orchestration

L'orchestration de conteneurs est devenue une activité à part entière, utilisant ses propres outils.

Docker swarm, Kubernetes, Ansibles, ...

L'utilisation de conteneurs en production à grande échelle demande l'automatisation de nombreuses tâches :
provisionnement, installation, réplication, monitoring, ...



Docker - Sauvegardes

Sauvegardes



Il est possible d'effectuer des sauvegardes d'images et/ou de conteneurs

```
docker commit [id-contenainer] [image:tag]
```

Enregistre toutes les modifications effectuées sur le conteneur dans une image.



Docker - Sauvegardes

Sauvegardes - distantes

Sauvegarde une image sur le docker hub

il faut avoir un compte ouvert et être connecté
avec `docker login`

```
docker push [repository/image[tag] ]
```



Docker - Sauvegardes

restauration - locale

Sauvegarde une image sur un fichier local

```
docker load --input fichier.tar
```



Docker - Sauvegardes

Sauvegardes - locale

Sauvegarde une image sur un fichier local

```
docker save --output fichier.tar image:tag
```



Docker Présentation



Questions ?