

Docker Introduction - CNAM 2017

Whoami :

- Damien DUPORTAL
 - Training Engineer @ CloudBees depuis 1 an
 - 6 ans à Atos Worldline: Production, Dev, Training, etc.
 - Intervenant ponctuel à EPSI, ENSG, CNAM
 - Docker mentor, depuis v0.3
- Contact:
 - Mail/Hangouts: damien.duportal@gmail.com
 - Twitter: [@DamienDuportal](https://twitter.com/DamienDuportal)
 - Github: [dduportal](https://github.com/dduportal)

Who are you ?

Agenda:

1. Docker: 101
2. Docker: bases
3. Docker avancé
4. Docker écosystème

Ces slides ont du contenu piqué de Docker <http://www.slideshare.net/dotCloud>

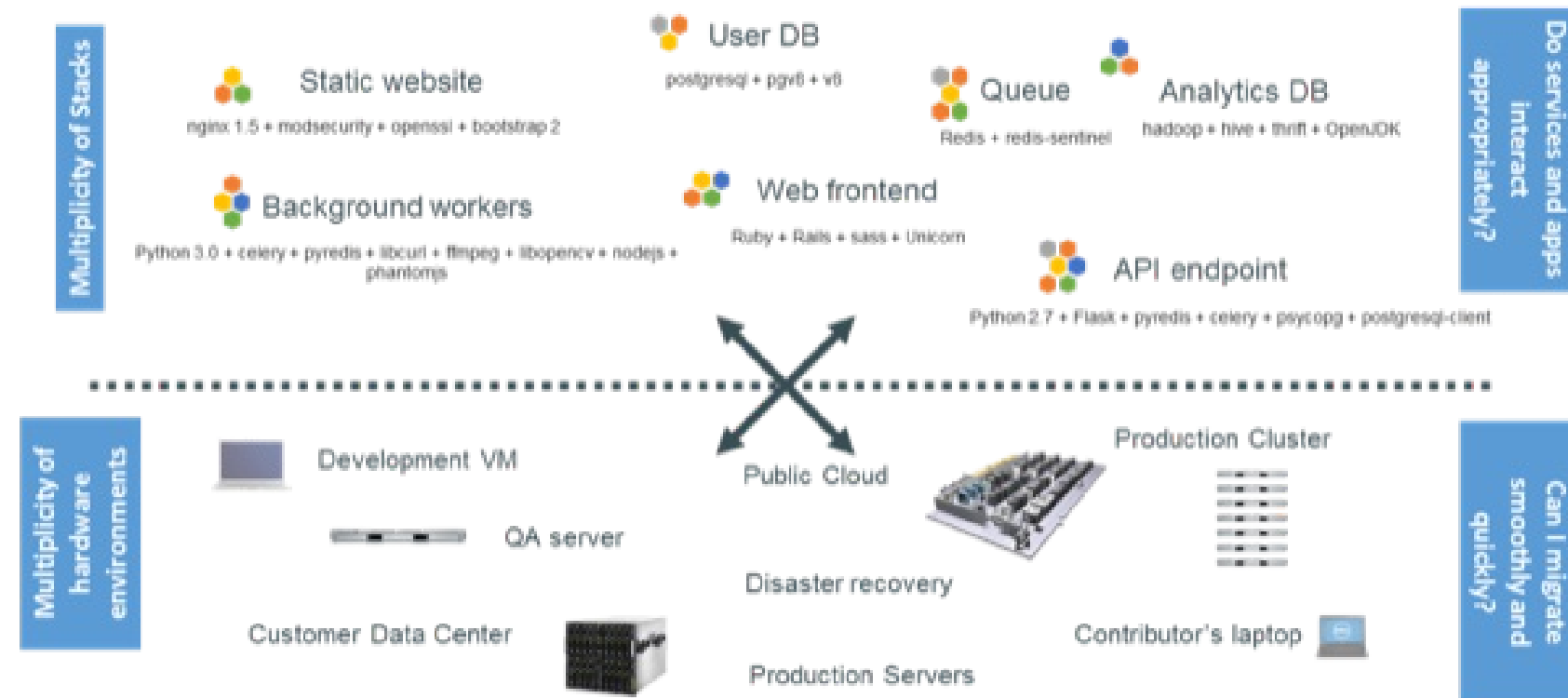
Docker: 101

Docker: 101

Pourquoi Docker ?

Pourquoi Docker ?










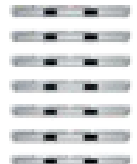



Quel est le problème que nous essayons de résoudre ?



"Matrix from Hell"

Pourquoi Docker ?

Problème de temps **exponentiel**


	Static website	?	?	?	?	?	?	?
	Web frontend	?	?	?	?	?	?	?
	Background workers	?	?	?	?	?	?	?
	User DB	?	?	?	?	?	?	?
	Analytics DB	?	?	?	?	?	?	?
	Queue	?	?	?	?	?	?	?
		Development VM	QA Server	Single Prod Server	Onsite Cluster	Public Cloud	Contributor's laptop	Customer Servers
								

Docker: 101

Déjà vu ?

Pourquoi Docker ?

L'IT n'est pas la seule industrie à résoudre des problèmes...

	?	?	?	?	?	?	?
	?	?	?	?	?	?	?
	?	?	?	?	?	?	?
	?	?	?	?	?	?	?
	?	?	?	?	?	?	?
	?	?	?	?	?	?	?
							

Docker: 101

Pourquoi Docker ?

Solution: Le container intermodal

"Separation of Concerns"



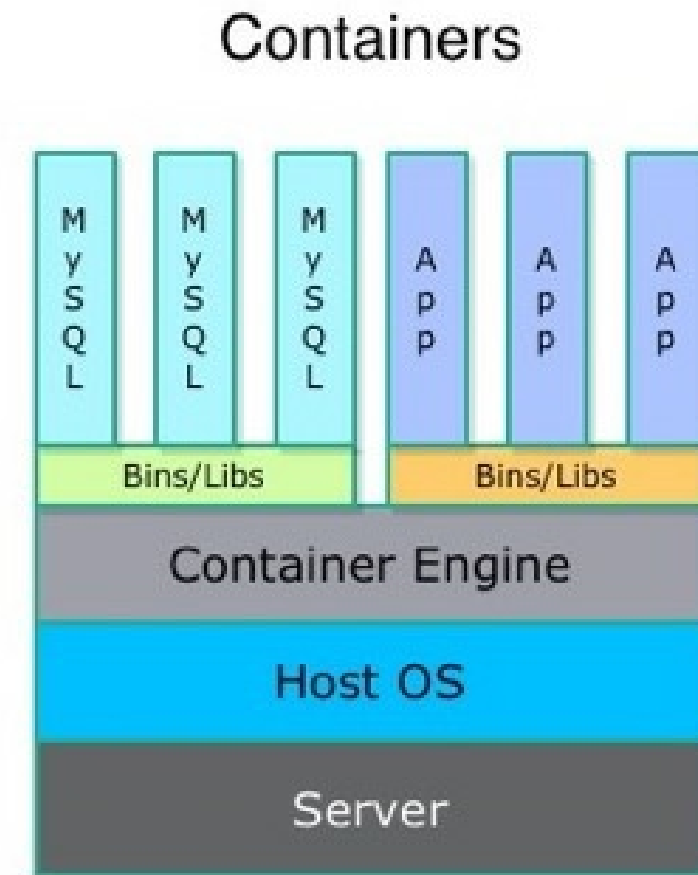
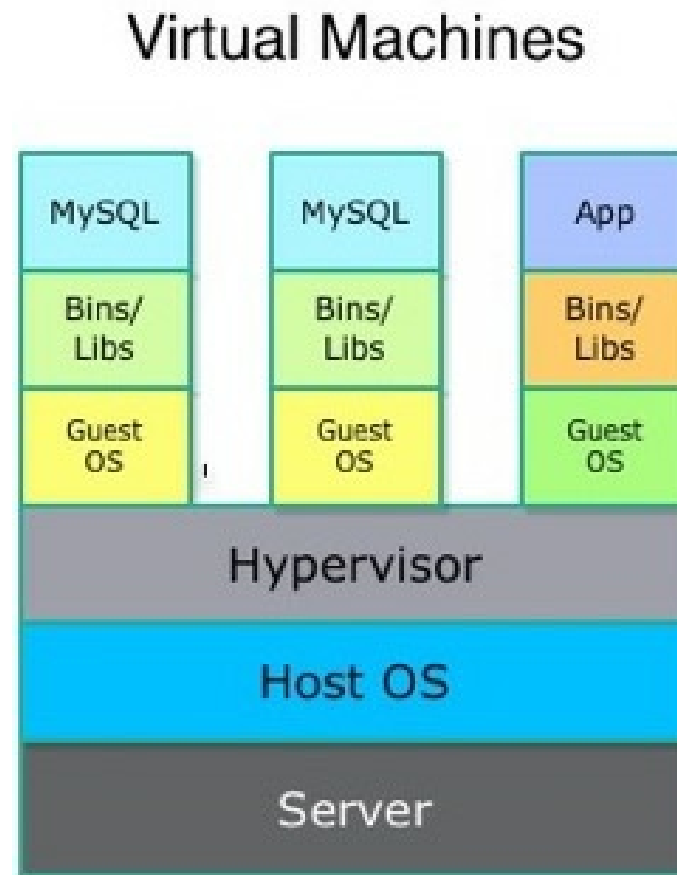
Docker: 101

Comment ça marche ?

Pourquoi Docker ?

"Virtualisation Légère"

Comment ça marche ?



Docker: 101

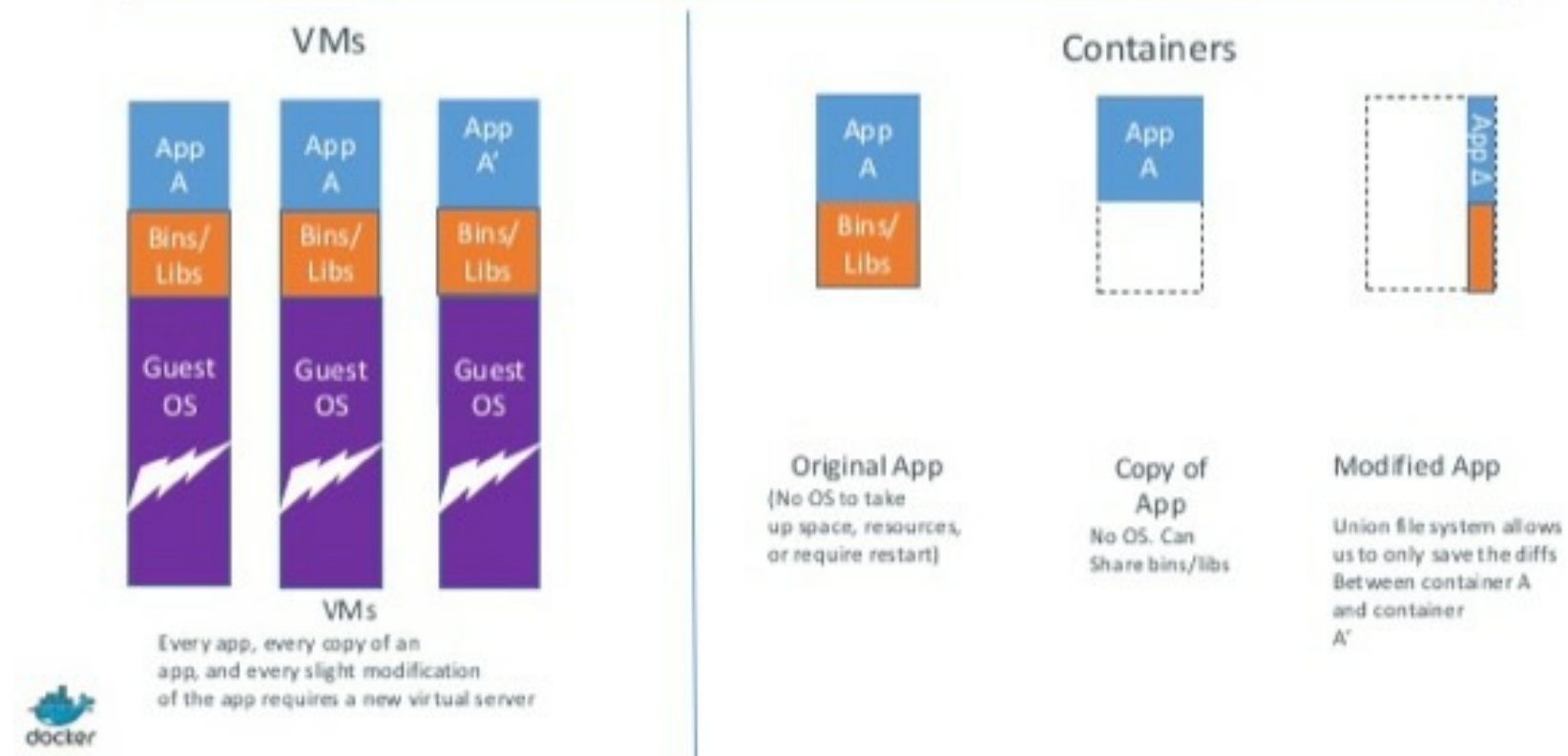
Comment ça marche ?

Pourquoi Docker ?

"Pourquoi **Léger**" ?

Comment ça marche ?

Why are Docker containers lightweight?



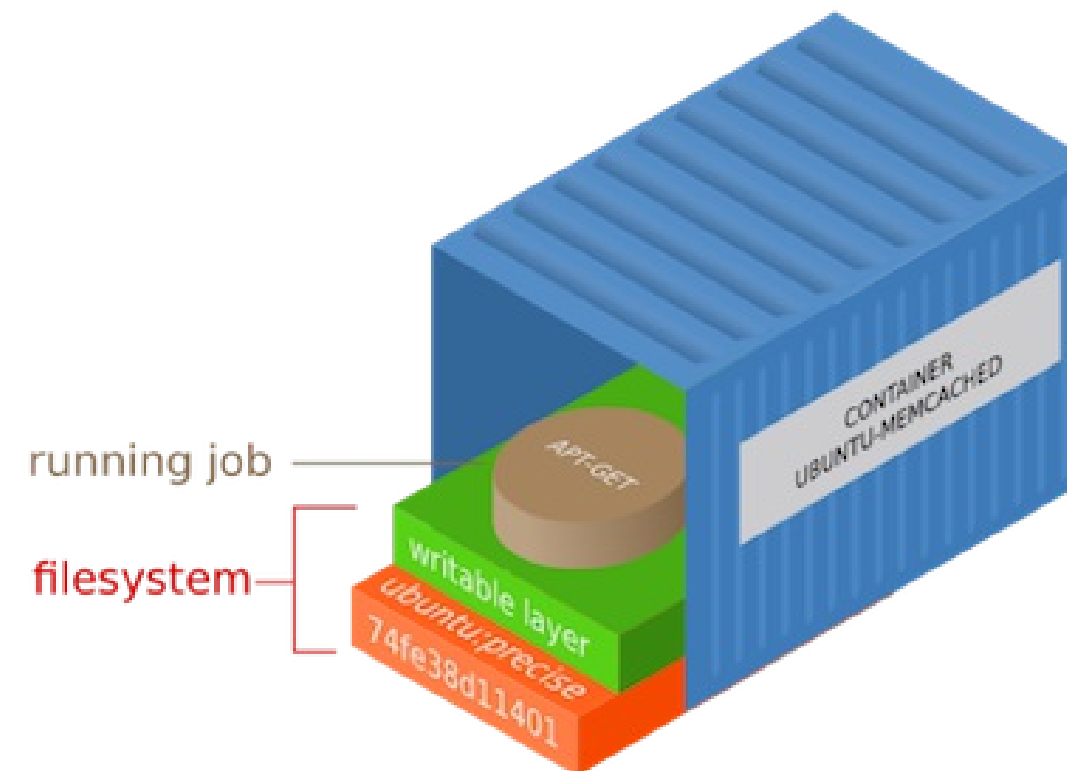
Docker: 101

Pourquoi Docker ?

Comment ça marche ?

Comment ça marche ?

- Linux Kernel requis (ou presque... Windows...)
- Linux containers: "super" chroot
 - "Namespacing": isolation (users, réseau, PIDs ...)
 - "Control Groups": gestion et contrôle (CPU, mem ...)
- Système de fichier de type "Union File System"
- Process **PID 1** et ses enfants *dans* le container



Docker: 101

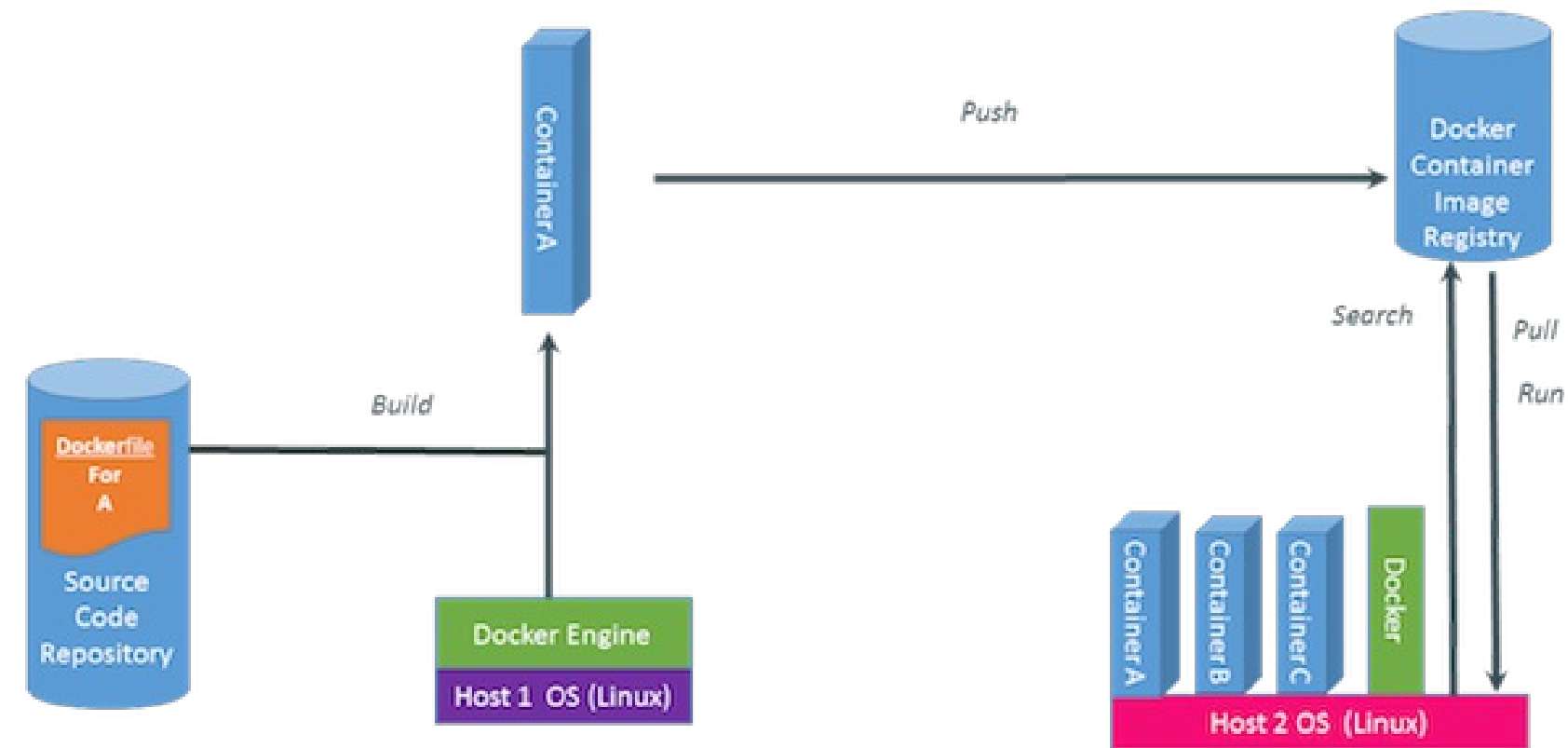
Pourquoi Docker ?

Comment ça marche ?

Docker workflow

Docker workflow

Workflow Docker basique:



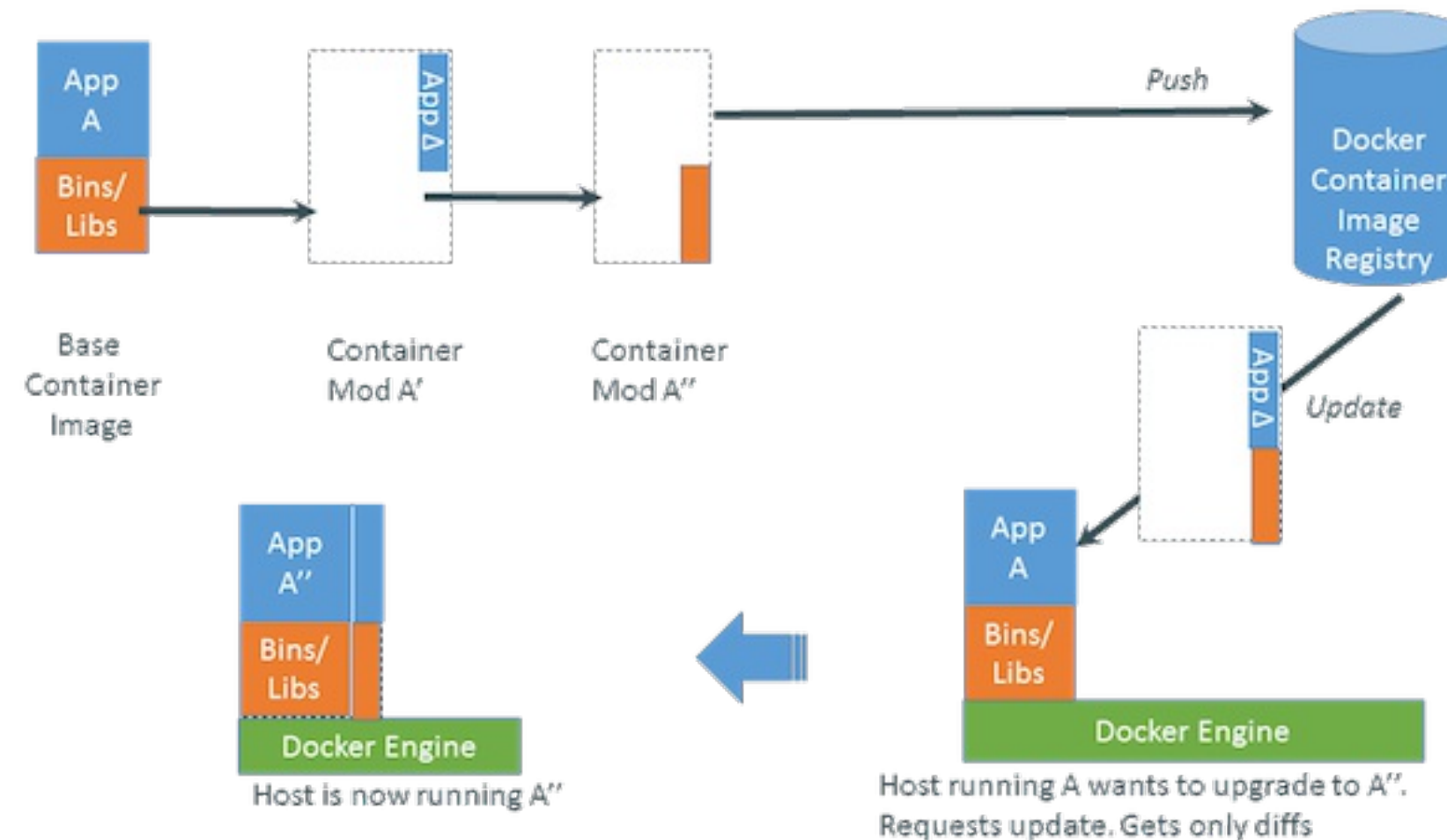
Docker: 101

Pourquoi Docker ?

Comment ça marche ?

Docker workflow

Exemple: mise à jour d'une application



Docker: 101

Pourquoi Docker ?

Comment ça marche ?

Docker workflow

Docker Inc.

Docker Inc.

- Fondé à Paris en 2008 par Solomon Hykes
- Migre à San Fransisco en 2009
- 2013: Open-source le projet Docker
- 2014: dotCloud devient Docker
- 2016: 1 milliard de levée de fond

Docker Project

- Originellement écrit en Python au sein de dotCloud
- Ré-écrit en **Golang** et *ouvert* en 2013 après une "PyCon"
- Open Source - **Apache licence**
- Disponible sur Github: <https://github.com/docker/docker>
- ~22 K commits, +1400 contributeurs

Docker: 101

Pourquoi Docker ?

Comment ça marche ?

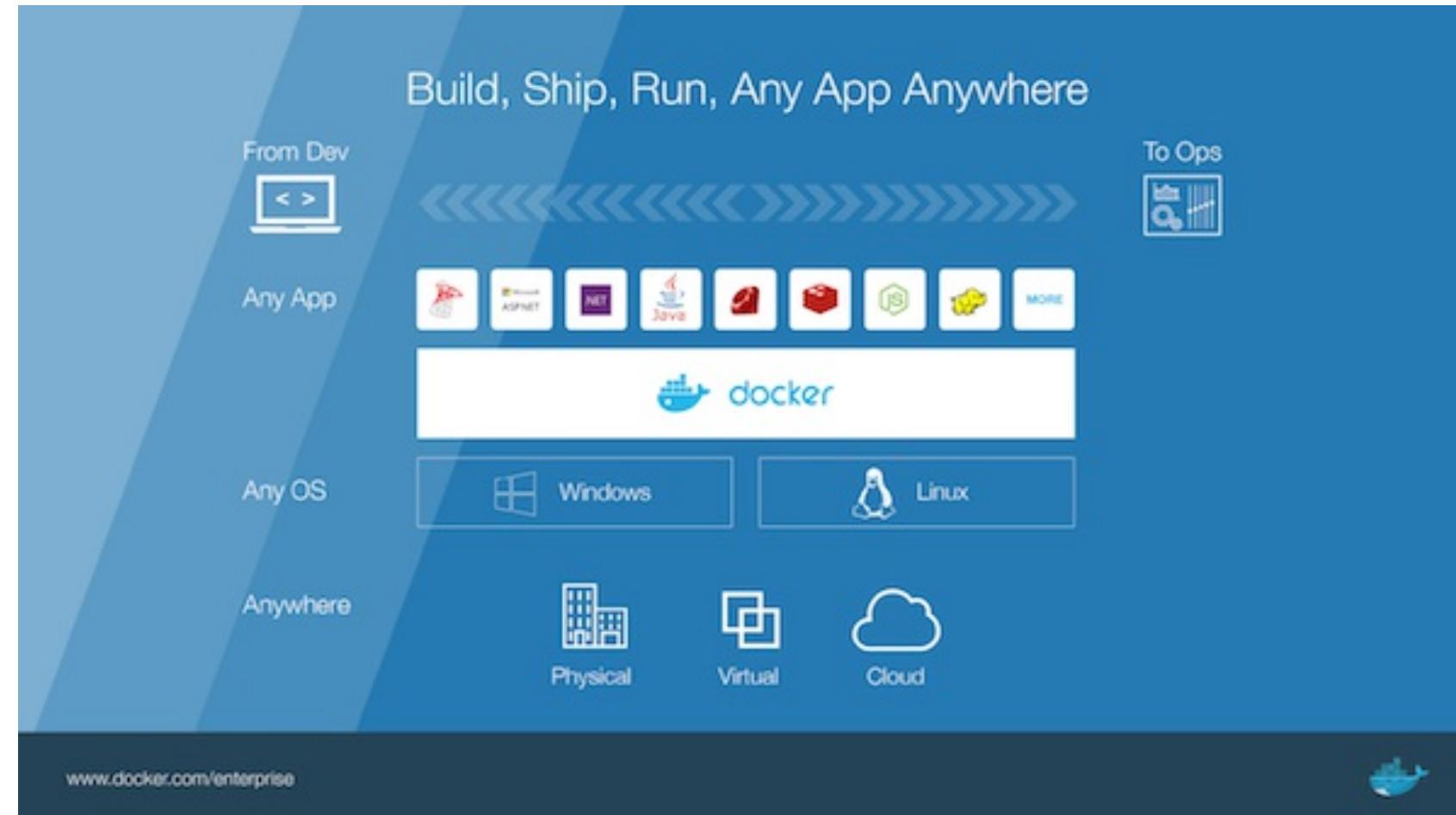
Docker workflow

Docker Inc.

Résumé

Résumé

Objectif de Docker :



Docker: 101

Container are NOT VMs

Pourquoi Docker ?

"Separation of concerns": 1 "tâche" par conteneur

Comment ça marche ?

Docker workflow

Docker Inc.

Résumé

VM



Containers



Docker: 101

Pourquoi Docker ?

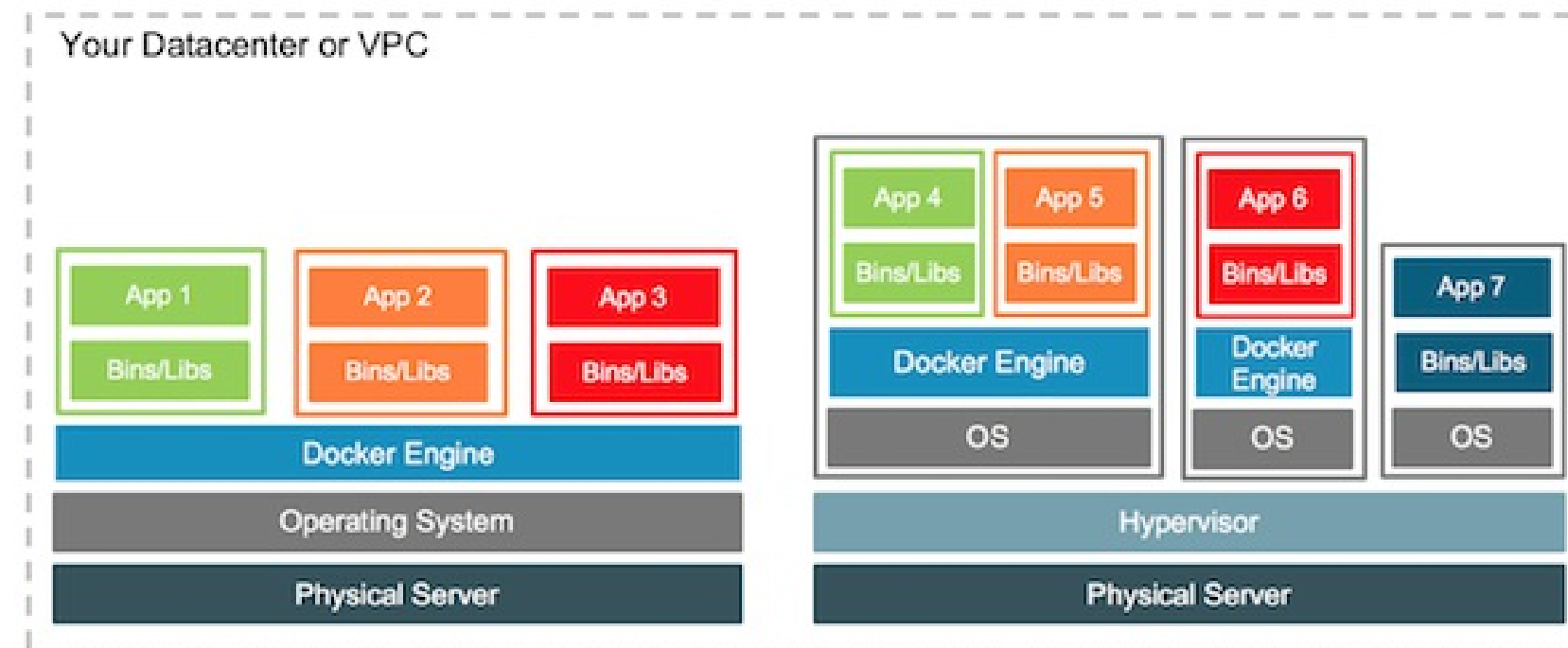
Comment ça marche ?

Docker workflow

Docker Inc.

Résumé

VM et conteneurs non exclusifs mutuellement



Docker: Bases

Un peu d'action !

Docker: 101

Docker: bases

Obtenir Docker

Obtenir Docker

- Un bac à sable est disponible sur <http://play-with-docker.com/>
 - Validez le Captcha
 - Vous avez 4 heures pour jouer avec des "instances Docker"
 - Cliquez sur le bouton "Add a new instance"
 - Une machine démarre, vous avez accès à la ligne de commande
- Page officielle d'installation: <https://www.docker.com/products/overview>
 - Linux: Utilisez votre gestionnaire de paquetage
 - MacOS and Windows: des installateurs "natifs" sont proposés
 - Vous pouvez utiliser une VM ou un service préconfiguré dans le Cloud de votre choix (Amazon, Azure, OVH, etc.)

Docker: 101

Docker: bases

Obtenir Docker

Vocabulaire

Vocabulaire

- **Docker Image** : Modèle (template) de base, représentant une application
- **Docker Container** : Unité d'exécution standard, instanciée depuis une image
- **Docker Engine** : Service responsable de créer, déployer et exécuter les conteneurs Docker sur un host physique, virtuel, distant ou local
- **Registry Service (Docker Hub or Docker Trusted Registry)** : Service de stockage et de distribution des images

Docker Images

- **Docker Image** : c'est le modèle de base
- Nommage: [REGISTRY/][NAMESPACE/]NAME[:TAG|@DIGEST]
 - Pas de Registre ? Défaut: registry.hub.docker.io
 - Pas de Namespace ? Défaut: library
 - Pas de tag ? Valeur par défaut: latest - ATTENTION !
 - Digest: signature unique basée sur le contenu

```
$ docker images
```

```
$ docker pull alpine:3.4
```

```
$ docker images # Quelles différences ?
```

```
$ docker pull alpine:latest
```

```
$ docker pull jfrog-docker-reg2.bintray.io/jfrog/artifactory-oss
```

```
$ docker images # Quelles différences ?
```

```
$ docker tag alpine:3.4 my-alpine:3.4.0-local
```

```
$ docker tag jfrog-docker-reg2.bintray.io/jfrog/artifactory-oss \  
artifactory-local
```

```
$ docker images # Quelles différences ?
```

Docker: 101

Docker: bases

Obtenir Docker

Vocabulaire

Docker Images

Docker Containers

Docker Containers 1/2

- **Docker Containers** : c'est l'unité d'exécution, instanciée depuis une image.

```
$ docker ps
$ docker run alpine:3.4 echo "Bonjour depuis un conteneur"

$ cat /etc/os-release
$ docker run -ti alpine:3.4 /bin/sh
#/ cat /etc/os-release
#/ whoami
#/ ps aux
#/ exit

$ docker run -d nginx:1.10-alpine
$ docker run -d --name webserver-1 nginx:1.10-alpine
$ docker ps

$ docker inspect nginx:1.10-alpine # IMAGE
$ docker inspect webserver-1      # Container

# Explorons d'autres options
$ docker ps -a
$ docker ps -q
```

Docker: 101

Docker Containers 2/2

Docker: bases

Obtenir Docker

Vocabulaire

Docker Images

Docker Containers

```
# Cycle de vie
$ docker run -d --name webserver-2 nginx:1.10-alpine
$ docker ps
$ docker stop webserver-2
$ docker ps
$ docker start webserver-2
$ docker ps
$ docker kill webserver-2
$ docker ps

# Accès au conteneur
$ docker run -d --name db-server postgres
$ docker ps

$ ps faux

$ docker exec db-server ps faux

$ docker exec -ti db-server sh
root@5da9f26d72c3:/# ps faux
root@5da9f26d72c3:/# exit
```


Docker: 101

Docker: bases

Obtenir Docker

Vocabulaire

Docker Images

Docker Containers

Docker Network

Docker Network

- Docker utilise du NAT par défaut:

```
$ docker run -d --name webserver-3 -p 8000:80 nginx:1.10-alpine

$ docker inspect --format '{{ .NetworkSettings.IPAddress }}' \
  webserver-3

# Access the webserver using the private network
$ curl -I http://172....:80

$ docker port webserver-3
$ docker ps

$ curl -I http://localhost:8000

# Let Docker manage this
$ docker run -d --name webserver-4 -P nginx:1.10-alpine
$ docker port webserver-3
$ curl -I http://localhost:<PORT FOUND>

# Try with your public IP
$ curl ifconfig.co # Or use Play With Docker IP
```

Docker: 101

Docker: bases

Obtenir Docker

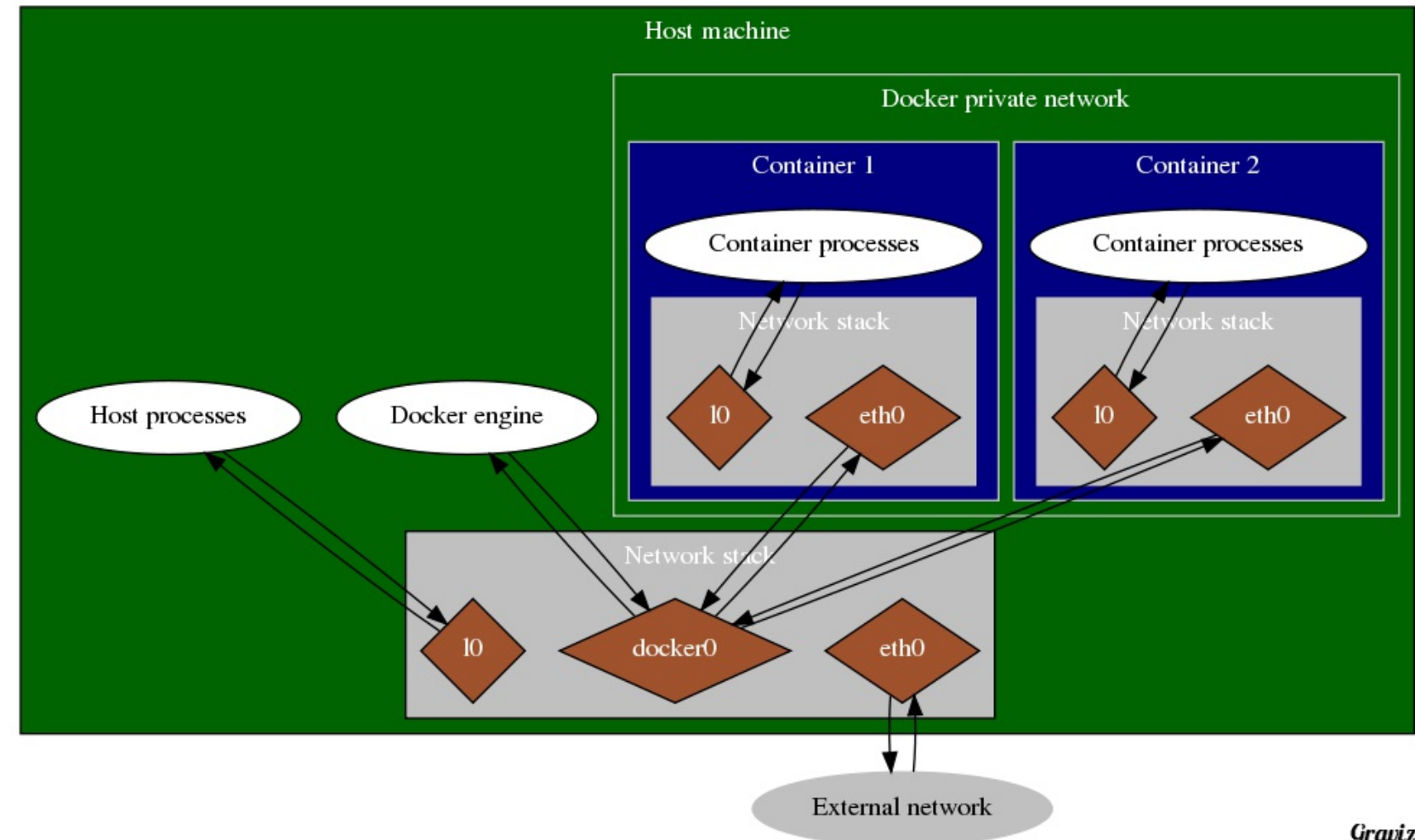
Vocabulaire

Docker Images

Docker Containers

Docker Network

Basic Network Summary



Gravizo

Docker: 101

Docker: bases

Obtenir Docker

Vocabulaire

Docker Images

Docker Containers

Docker Network

Docker Volumes

Docker Volumes

- Quand le cycle de vie de la donnée est différent de celui du conteneur
 - Application HTML/PHP/JS - Serveur Web
 - Données BDD - Serveur de BDD

```
$ docker run alpine ls -l /app
$ docker run --volume /app alpine ls -l /app

$ docker run -d -v /application --name ws-vol nginx:1.10-alpine
$ docker inspect ws-vol | grep -i -A10 Mounts

$ touch <SOURCE_DIR>/_data/toto # Sudo est peut être nécessaire
$ docker exec -ti ws-vol ls -l /application/toto

# Pour certain cas d'usages, mais ATTENTION ICI
$ pwd
$ echo "ok" > file.txt
$ ls -l
$ docker run -ti -v $(pwd):/partage alpine ls -l /partage
```

Docker: bases

Obtenir Docker

Vocabulaire

Docker Images

Docker Containers

Docker Network

Docker Volumes

Nettoyage

```
$ docker run -d --name ws-trash nginx:1.10-alpine
$ docker kill ws-trash
$ docker rm ws-trash

$ docker run -d -v /app --name ws-trash-2 nginx:1.10-alpine
$ docker kill ws-trash-2
$ docker rm -v ws-trash-2

$ docker rmi nginx:alpine

# DALECK / TERMINATOR MODE
$ docker ps -q | xargs docker kill

$ docker ps -a -q | xargs docker rm -v

$ docker images -q | xargs docker rmi -f
```

Docker avancé

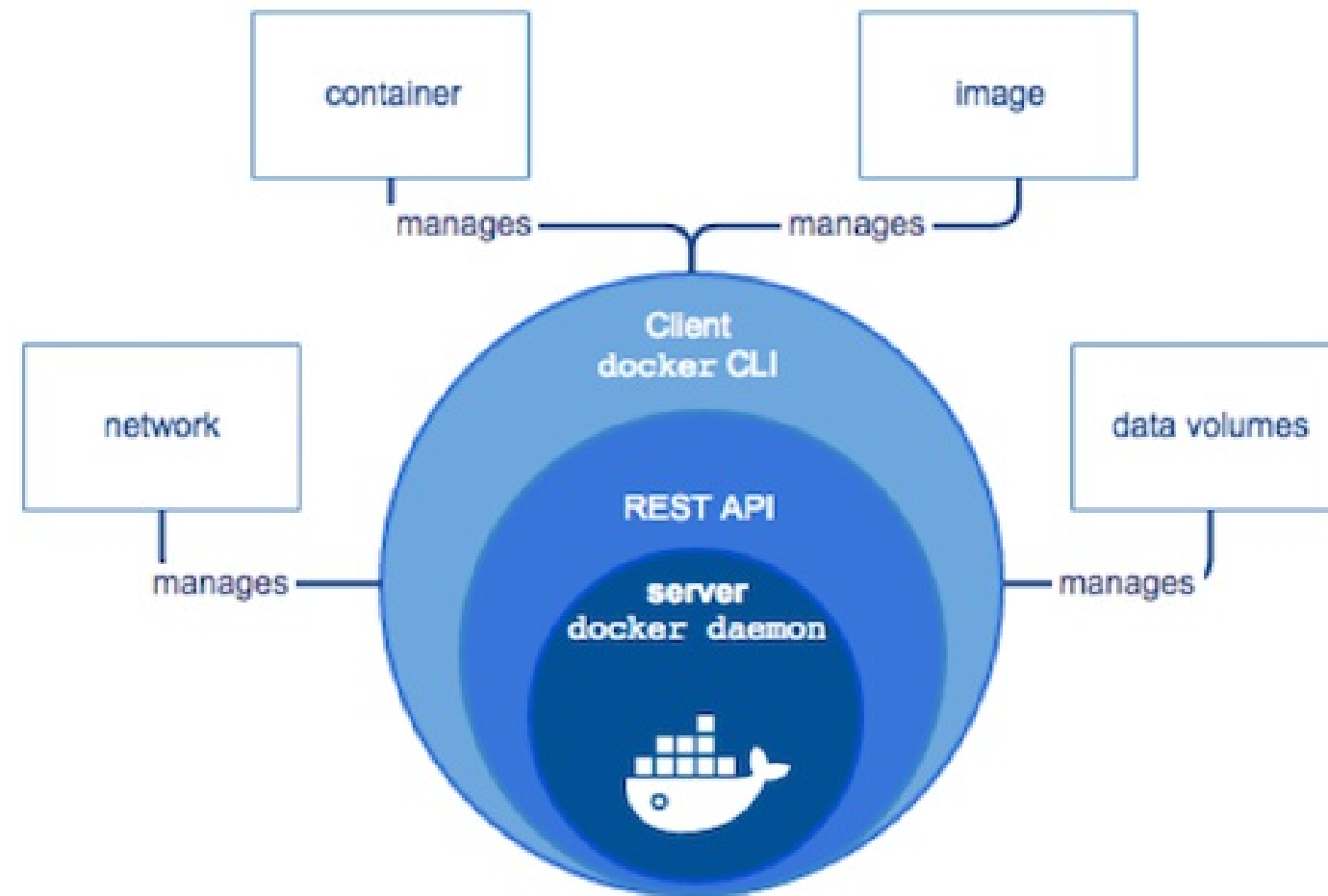
Docker: 101

Docker: bases

Docker avancé

Plateforme

Plateforme : Vue globale



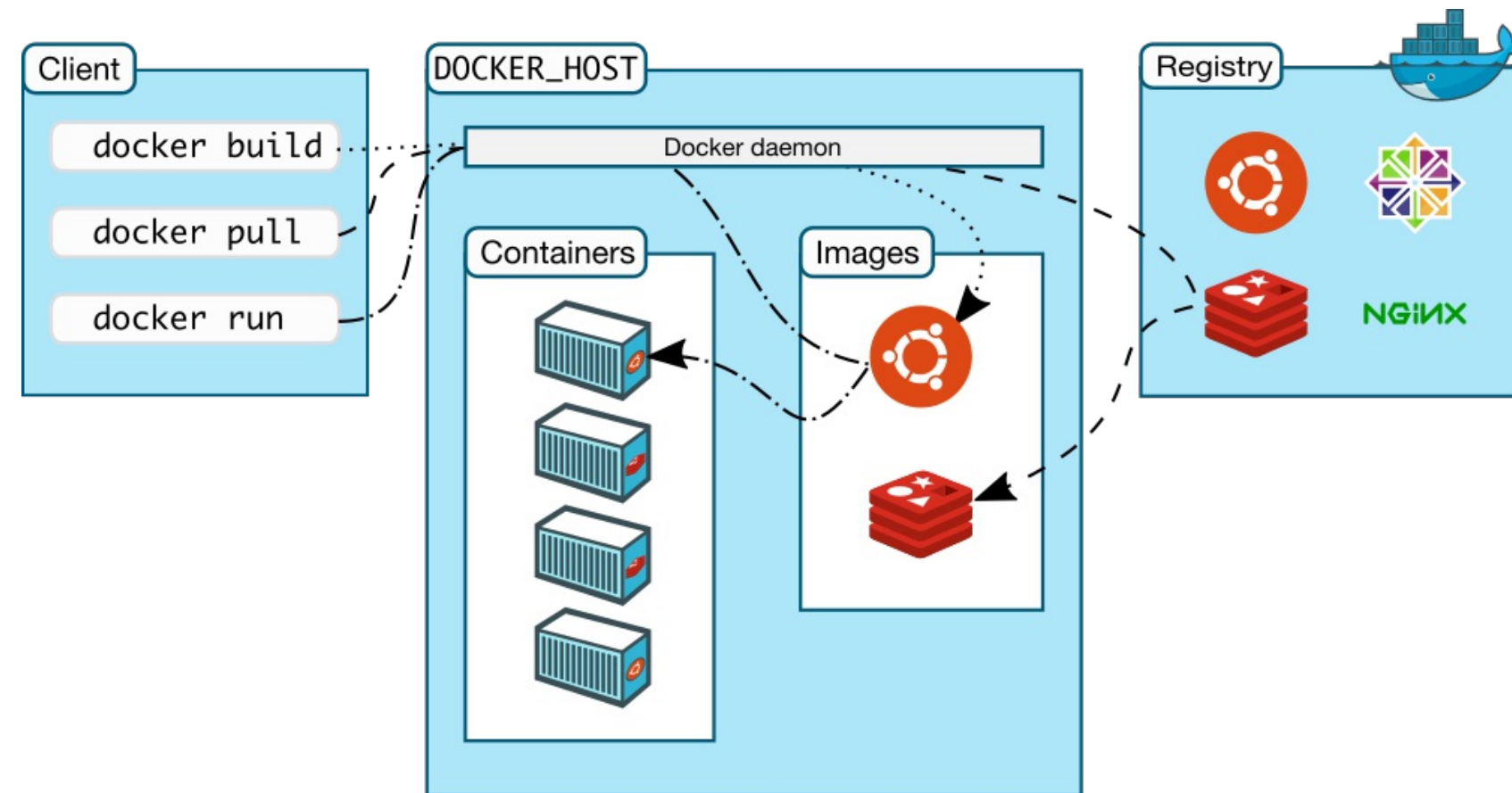
Docker: 101

Docker: bases

Docker avancé

Plateforme

Plateforme : Architecture



Docker: 101

Docker: bases

Docker avancé

Plateforme

Plateforme : Docker Hub

The screenshot shows the Docker Hub interface for the 'nginx' repository. At the top, there's a dark blue header with the Docker logo, a search bar containing 'nginx', and links for 'Explore', 'Help', 'Sign up', and 'Sign In'. Below the header, the repository is identified as the 'OFFICIAL REPOSITORY' for 'nginx', with a star icon and a note 'Last pushed: 17 days ago'. The main content area is divided into two columns. The left column has tabs for 'Repo Info' and 'Tags', with 'Tags' selected. It contains a 'Short Description' box stating 'Official build of Nginx.' and a 'Full Description' box. The 'Full Description' box contains a list of supported tags and their respective Dockerfile links, followed by two paragraphs of text providing more information about the image and its history. The right column has a 'Docker Pull Command' box with the command 'docker pull nginx' and a copy icon.

nginx

Explore Help Sign up Sign In

OFFICIAL REPOSITORY

nginx ☆

Last pushed: 17 days ago

Repo Info Tags

Short Description

Official build of Nginx.

Docker Pull Command

docker pull nginx

Full Description

Supported tags and respective Dockerfile links

- 1.11.8 , mainline , 1 , 1.11 , latest ([mainline/jessie/Dockerfile](#))
- 1.11.8-alpine , mainline-alpine , 1-alpine , 1.11-alpine , alpine ([mainline/alpine/Dockerfile](#))
- 1.10.2 , stable , 1.10 ([stable/jessie/Dockerfile](#))
- 1.10.2-alpine , stable-alpine , 1.10-alpine ([stable/alpine/Dockerfile](#))

For more information about this image and its history, please see [the relevant manifest file](#) ([library/nginx](#)). This image is updated via [pull requests to the](#) `docker-library/official-images` [GitHub repo](#).

For detailed information about the virtual/transfer sizes and individual layers of each of the above supported tags, please see [the](#) `repos/nginx/tag-details.md` [file](#) in [the](#) `docker-library/repo-info` [GitHub repo](#).

What is Nginx?

Nginx (pronounced "engine-x") is an open source reverse proxy server for HTTP, HTTPS, SMTP, POP3, and IMAP protocols, as well as a load balancer, HTTP cache, and a web server (origin server). The nginx project started with a strong focus on high concurrency, high performance and low memory usage. It is licensed under the 2-clause BSD-like license and it runs on Linux, BSD variants, Mac OS X, Solaris, AIX, HP-UX, as well as on other *nix flavors. It also has a proof of concept port for Microsoft Windows.

Docker: 101

Docker: bases

Docker avancé

Plateforme

Plateforme : Docker Cloud

The screenshot shows the Docker Cloud interface. At the top is a blue header with the Docker Cloud logo, navigation links for Reports, Docs, and Forums, and a user profile for 'borja'. Below the header is a grey navigation bar with tabs for Stacks, Services (selected), Nodes, and Repositories. The main content area is titled 'Service dashboard' and includes a 'Create service' button. A table lists the services, with columns for Name, Status, Image, and Actions. The services listed are 'db' (Running), 'wordpress' (Running), 'authorizedkeys' (Not running), 'mongo' (Redeploying), and 'web' (Running with a warning icon). Each service row includes details like the number of containers and the stack name, along with buttons for Stop, Terminate, Redeploy, and Start.

Name	Status	Image	Actions
db 1 Container • wordpress-stackable	Running	mysql:5.5	Stop Terminate Redeploy
wordpress 1 Container • wordpress-stackable	Running	tutum/wordpress-stackable:lat...	Stop Terminate Redeploy
authorizedkeys 0 Containers • authorizedkeys	Not running	dockercloud/authorizedkeys:lat...	Start Terminate Redeploy
mongo 1 Container • quickstart-go	Redeploying	mongo:latest	Redeploy
web 1 Container • quickstart-go	Running ⚠	dockercloud/quickstart-go:latest	Stop Terminate Redeploy

Docker: 101

Docker: bases

Docker avancé

Plateforme

Dockerfile

Dockerfile: Fabriquer ses propres images

- Dockerfile: recette pour fabriquer son image

```
FROM debian:jessie
MAINTAINER Damien DUPORTAL <damien.duportal@gmail.com>

RUN apt-get update && apt-get install -y nginx

VOLUME ["/tmp", "/app"]

EXPOSE 80

ENTRYPOINT ["/usr/sbin/nginx"]
CMD [ "-g", "daemon off;" ]
```

Docker: 101

Docker: bases

Docker avancé

Plateforme

Dockerfile

Dockerfile: Fabriquer ses propres images

- `docker build`: commande pour fabriquer une **image** depuis un `Dockerfile`

```
$ mkdir web && cd ./web
$ vi Dockerfile # Utilisez le Dockerfile de la slide précédente
$ docker build ./

$ cd ..
$ docker build -t web:1.0.0 ./web/

$ docker run -d -P --name my-web-1 web:1.0.0
# Affichez la page avec docker port (..) et curl (...)

$ vi ./web/Dockerfile
# Ajoutez la ligne ci-dessous
# RUN echo Hello > /var/www/html/index.nginx-debian.html

$ docker build -t web:1.1.0 ./web/
$ docker run -d -P --name my-web-2 web:1.1.0
# Affichez la nouvelle page avec docker port (..) et curl (...)
```

Docker: 101

Docker: bases

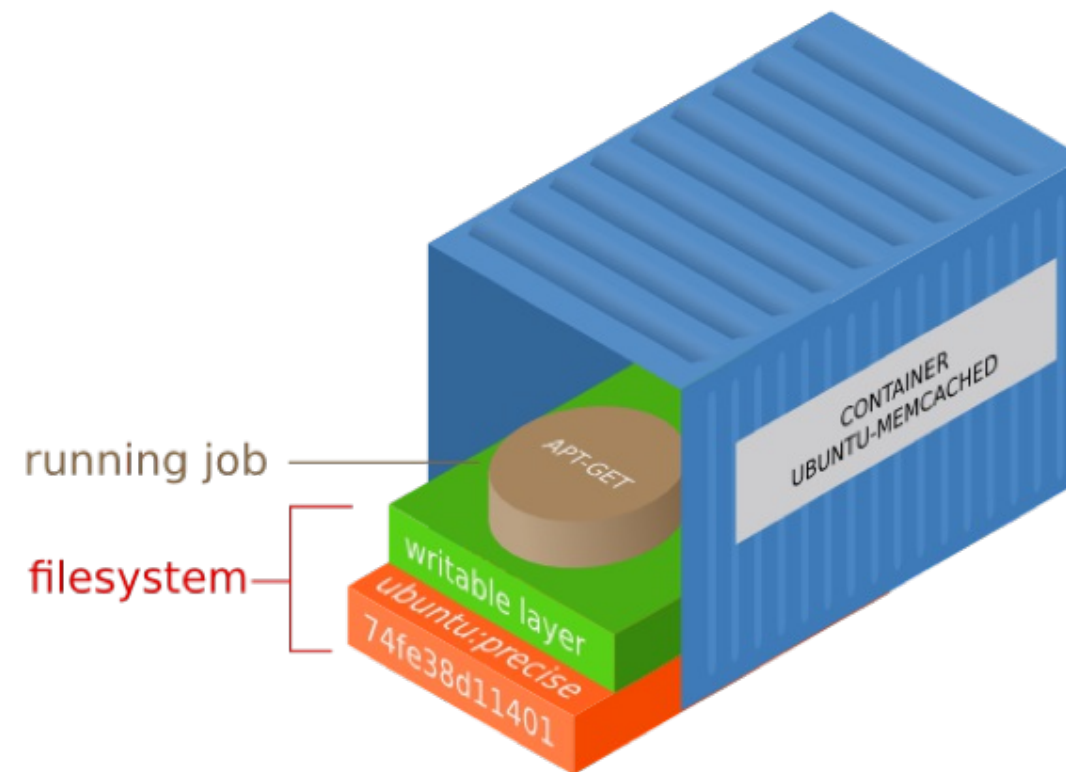
Docker avancé

Plateforme

Dockerfile

Comment ça marche ? 1/2

- Pas dans un volume ? => Union File System (AUFS/BTRFS/DeviceMapper...)
- Les images "lecture seule": les écritures se font dans une nouvelle couche ("Writable Layer")



Docker: 101

Docker: bases

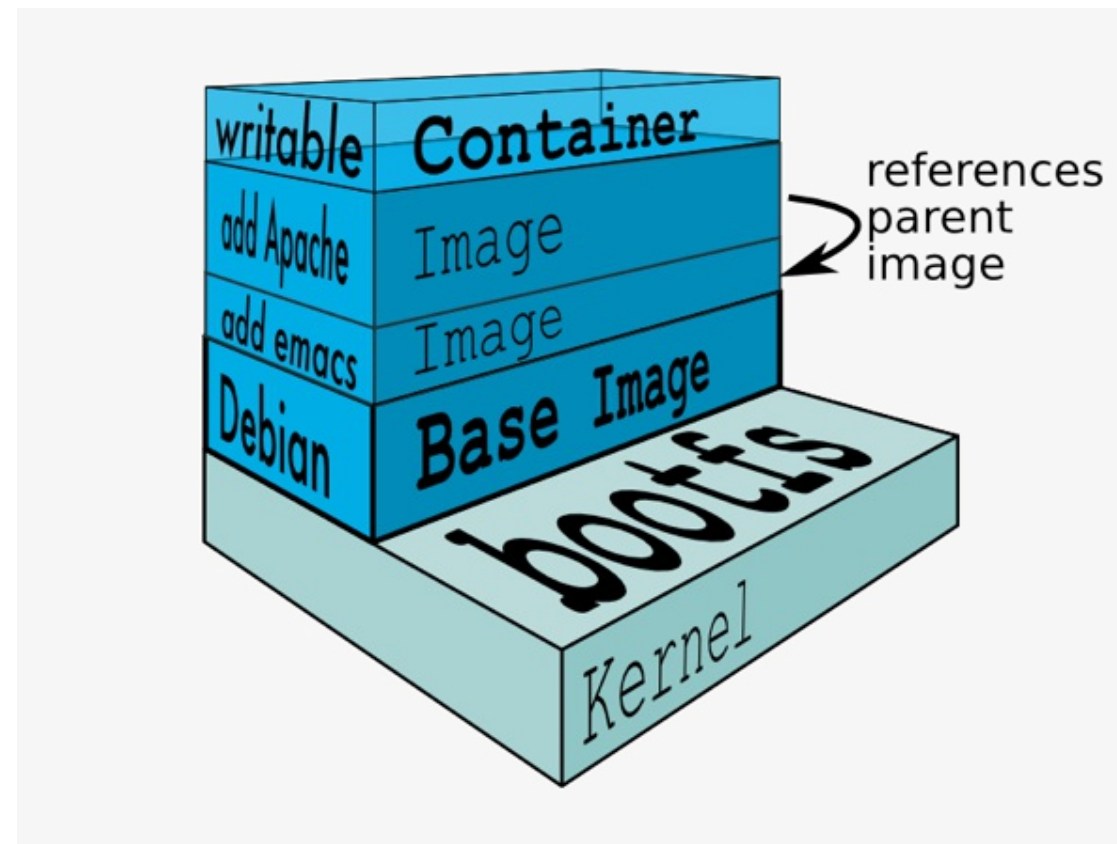
Docker avancé

Plateforme

Dockerfile

Comment ça marche ? 2/2

- On peut "committer" un "writeable layer": il devient lecture seule
- Une image est donc une collection ordonnée de "layers" en lecture seule
- Une instruction Dockerfile == 1 "layer" du Union FS, commité **automatiquement**



Docker: 101

Docker: bases

Docker avancé

Plateforme

Dockerfile

Partages

Partages: Réseau 1/2

- Comment faire dialoguer 2 conteneurs sans publier leurs ports
 - Lien Serveur d'application <-> Base de données
- Exemple avec Redis:

```
$ docker pull redis
$ docker inspect redis | grep -i -A2 expose
# Quel ports sont "exposés" ?

$ docker run -d --name redissrv redis # Pas de port publié

$ docker run -ti --link redissrv:dbserver redis env

$ docker run -ti --link redissrv:dbserver redis bash
root@CONTAINER $/ cat /etc/hosts

$ docker run -ti --link redissrv:dbserver \
redis redis-cli -h dbserver -p 6379
```

Docker: 101

Docker: bases

Docker avancé

Plateforme

Dockerfile

Partages

Partages: Réseau 2/2

- Comment gérer du réseau qui change (adresse, port, etc.) ?
 - Command `docker network`
 - Drapeau `--net` pour la commande `docker run`

```
$ docker network --help
$ docker network ls

$ docker network create db-net-1
$ docker network ls # Différences ?

$ docker run -d --net=db-net-1 --name redissrv-2 redis
# Pas de port publié

$ docker network inspect db-net-1

$ docker run -ti --rm --net=db-net-1 redis cat /etc/hosts
# Plus de ligne référençant le serveur

$ docker run -ti --rm --net=db-net-1 redis \
  redis-cli -h redissrv-2 -p 6379
# Les noms des conteneurs sont gérées dans un serveur DNS
# DNS: Dynamique !
```

Docker: 101

Docker: bases

Docker avancé

Plateforme

Dockerfile

Partages

Partages: Volumes 1/2

- Comment partager des volumes entre des conteneurs ?
 - Le conteneur initial doit déclarer un volume
 - Utiliser le drapeau `--volumes-from`

```
$ docker run -d -v /app --name pere nginx:1.10-alpine
```

```
$ docker run --rm -ti --volumes-from pere debian:jessie bash
root@CONTAINER$ echo "Hello" > /app/hello.txt
root@CONTAINER$ exit
```

```
$ docker exec -ti pere sh
root@pere$ cd /app
root@pere$ ls -l
root@pere$ cat /app/foo
root@pere$ exit
```


Docker: 101

Docker: bases

Docker avancé

Plateforme

Dockerfile

Partages

Partages: Volumes 2/2

- Tout comme pour les réseaux, docker peut gérer les volumes "à part"
 - Commande `docker volume`

```
$ docker volume --help
```

```
$ docker volume ls
```

```
$ docker volume create --name=shared-data
```

```
$ docker volume ls
```

```
$ docker run --rm -ti -v shared-data:/partage alpine sh  
#/ echo "hello again" > /partage/fichier.txt  
#/ exit
```

```
$ docker run --rm -ti -v shared-data:/DATA debian:jessie \  
cat /partage/fichier.txt
```

```
$ docker run --rm -ti -v shared-data:/DATA debian:jessie \  
cat /DATA/fichier.txt
```

Docker: 101

Docker: bases

Docker avancé

Plateforme

Dockerfile

Partages

Partages: Résumé

- Docker peut gérer les réseaux et volumes de fichiers
- AVANTAGES:
 - Convention
 - Pas de configuration à maintenir
 - Portabilité

Docker: Écosystème

Docker: 101

Docker: bases

Docker avancé

Docker:
Écosystème

Compose

Compose 1/2

- Compose est un client Docker, qui va exécuter les commandes pour vous
- Source: fichier au format YAML:

```
version: "2"

volumes:
  db-data:

networks:
  front-tier:
  back-tier:

services:
  vote:
    build: ./vote
    ports:
      - "5000:80"
    networks:
      - front-tier
      - back-tier

  redis:
    image: redis:alpine
    volumes:
      - db-data:/var/redis
    networks:
      - back-tier
```

Compose 2/2

Docker: 101

Docker: bases

Docker avancé

Docker:
Écosystème

Compose

- Commande `docker-compose`
- Gestion intelligente des cycles de vie:
 - Mise à jour d'une image sans toucher au contenu du data volume
- Scaling des conteneurs
 - Ne gère PAS le routage pour vous !
- Futur: intelligence vers le Docker Daemon avec la commande `docker service`
 - Conversion des YAML vers le Docker Daemon

Swarm 1/2

Docker: 101

Docker: bases

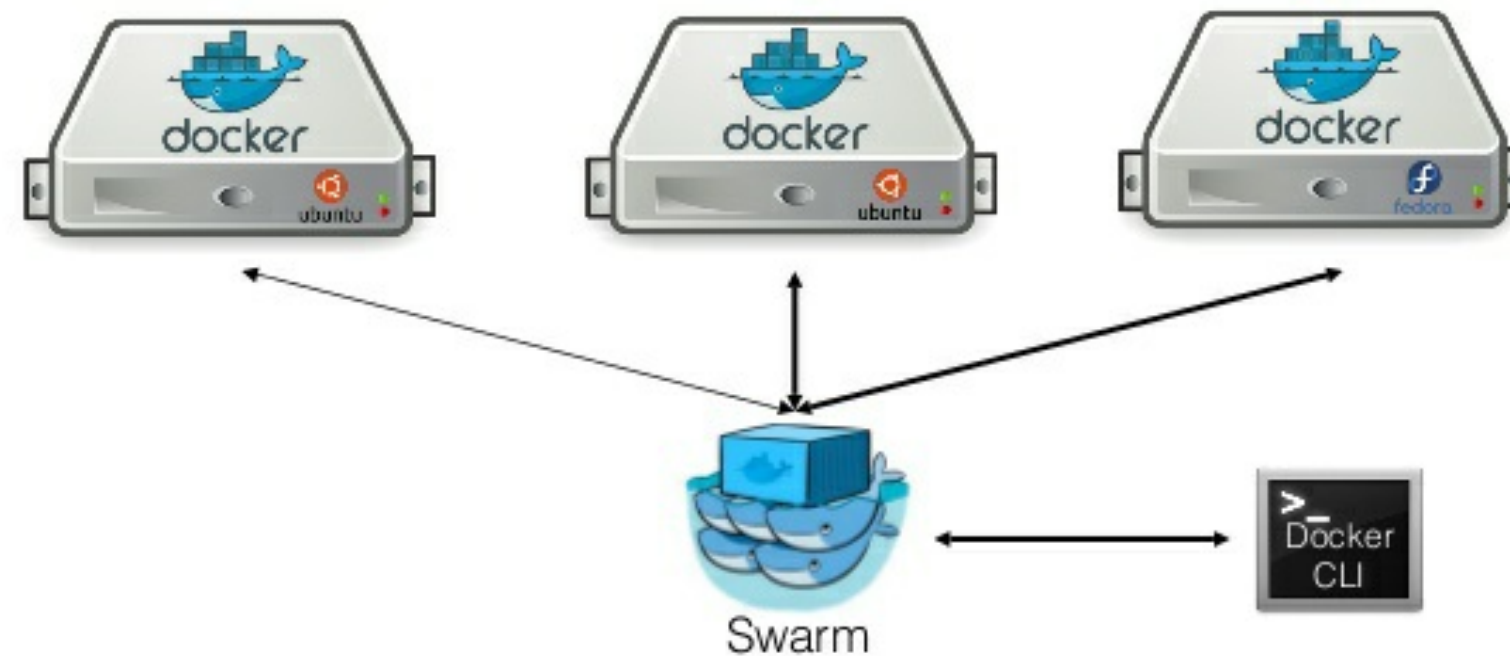
Docker avancé

Docker:
Écosystème

Compose

Swarm

With Docker Swarm



Swarm 2/2

Docker: 101

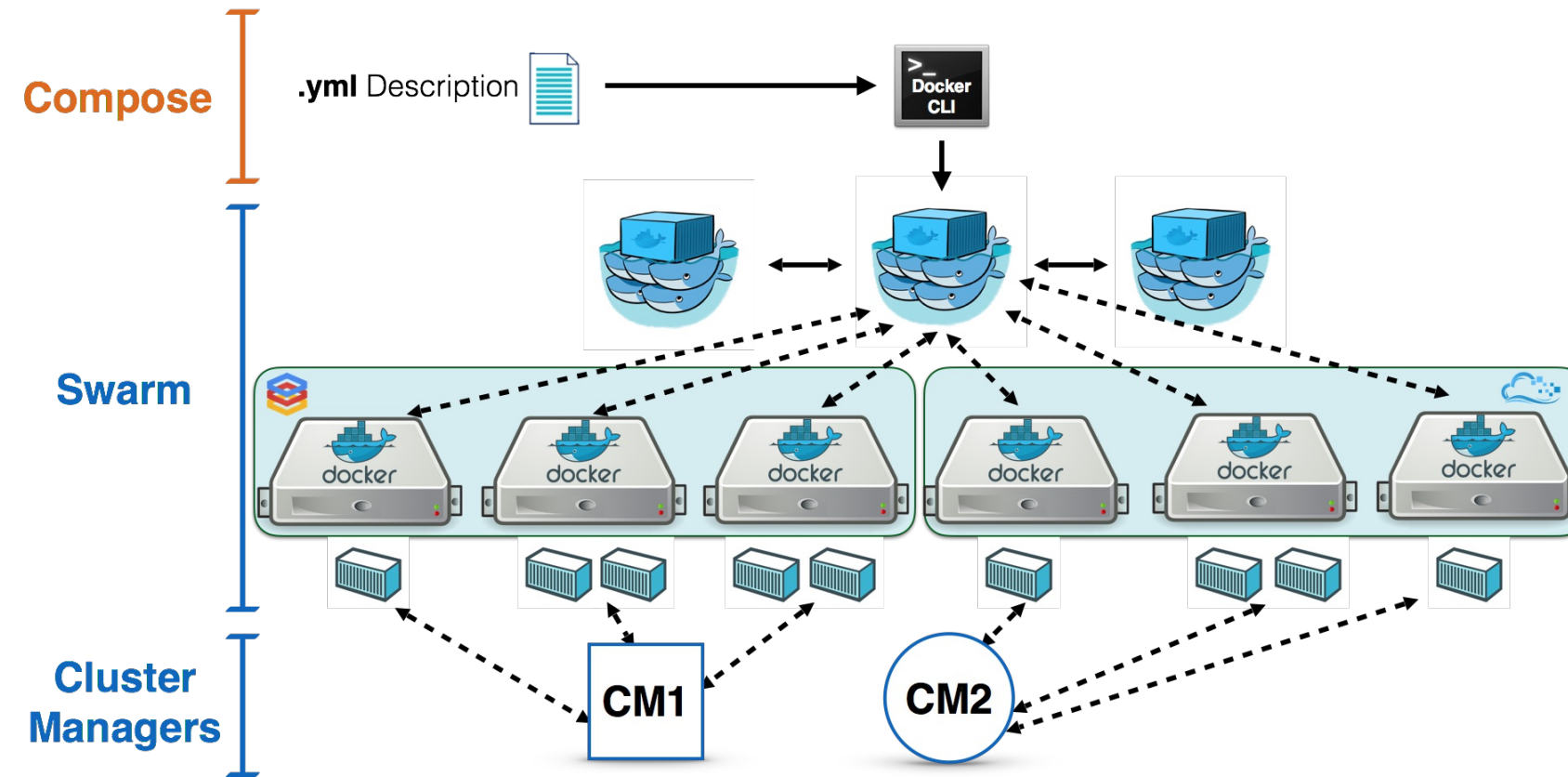
Docker: bases

Docker avancé

Docker:
Écosystème

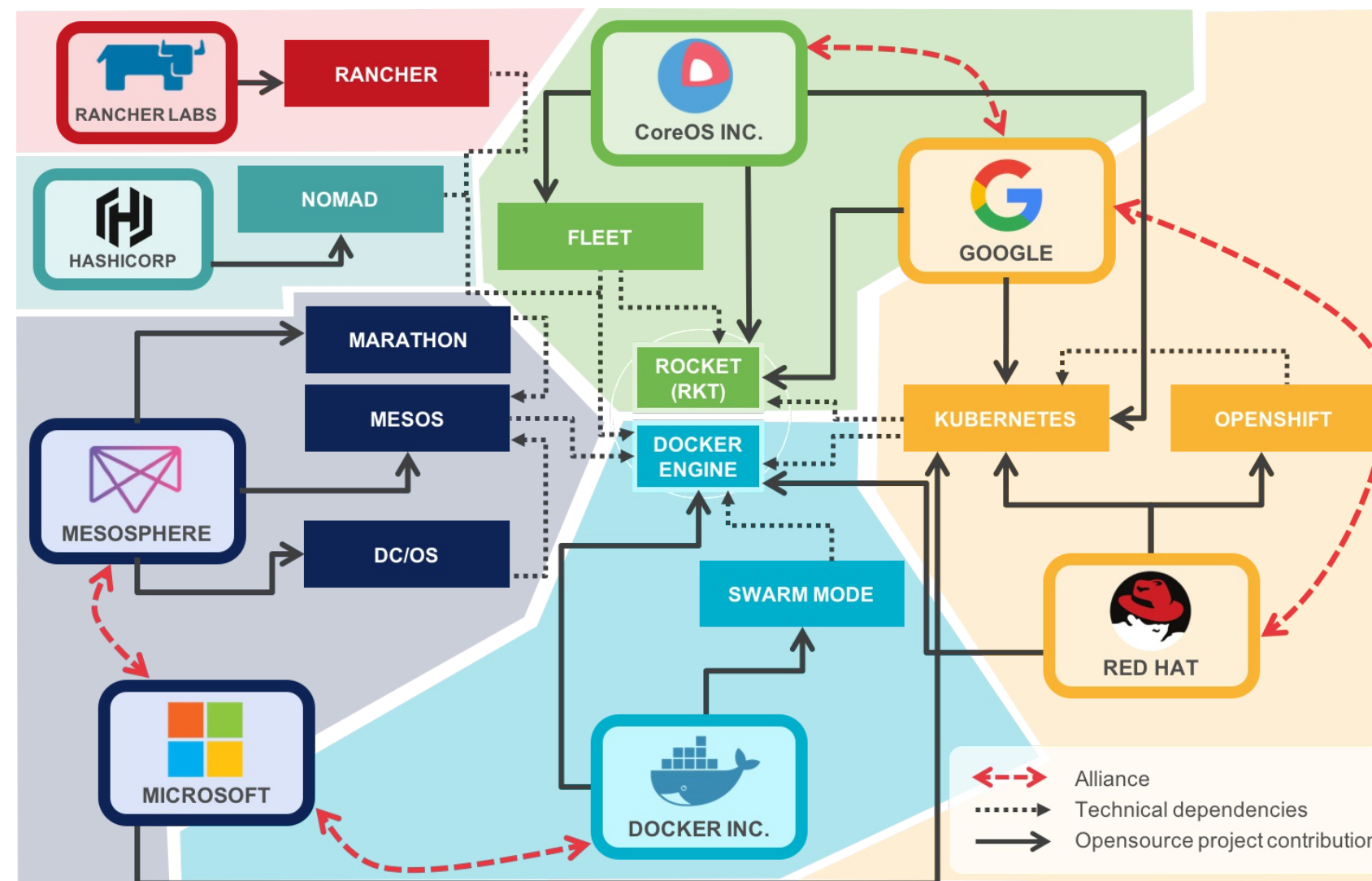
Compose

Swarm



Containerus Bellum

<http://blog.octo.com/containerus-bellum-ou-la-chronique-des-hostilites-dans-lecosysteme-docker/>



Merci !
Questions ?