DÉPLOIEMENT D'UN JUPYTERHUB À L'AIDE DE DOCKER SWARM

Aurélien Bailly-Reyre

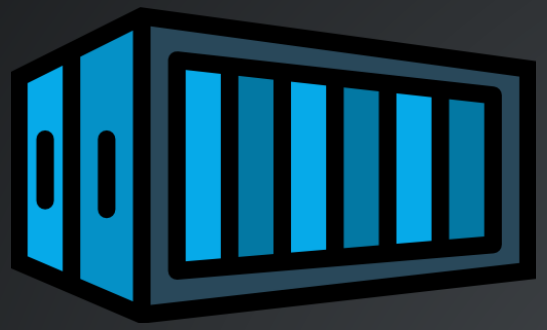
abaillyr@lpnhe.in2p3.fr

LPNHE - Sorbonne Université / IN2P3

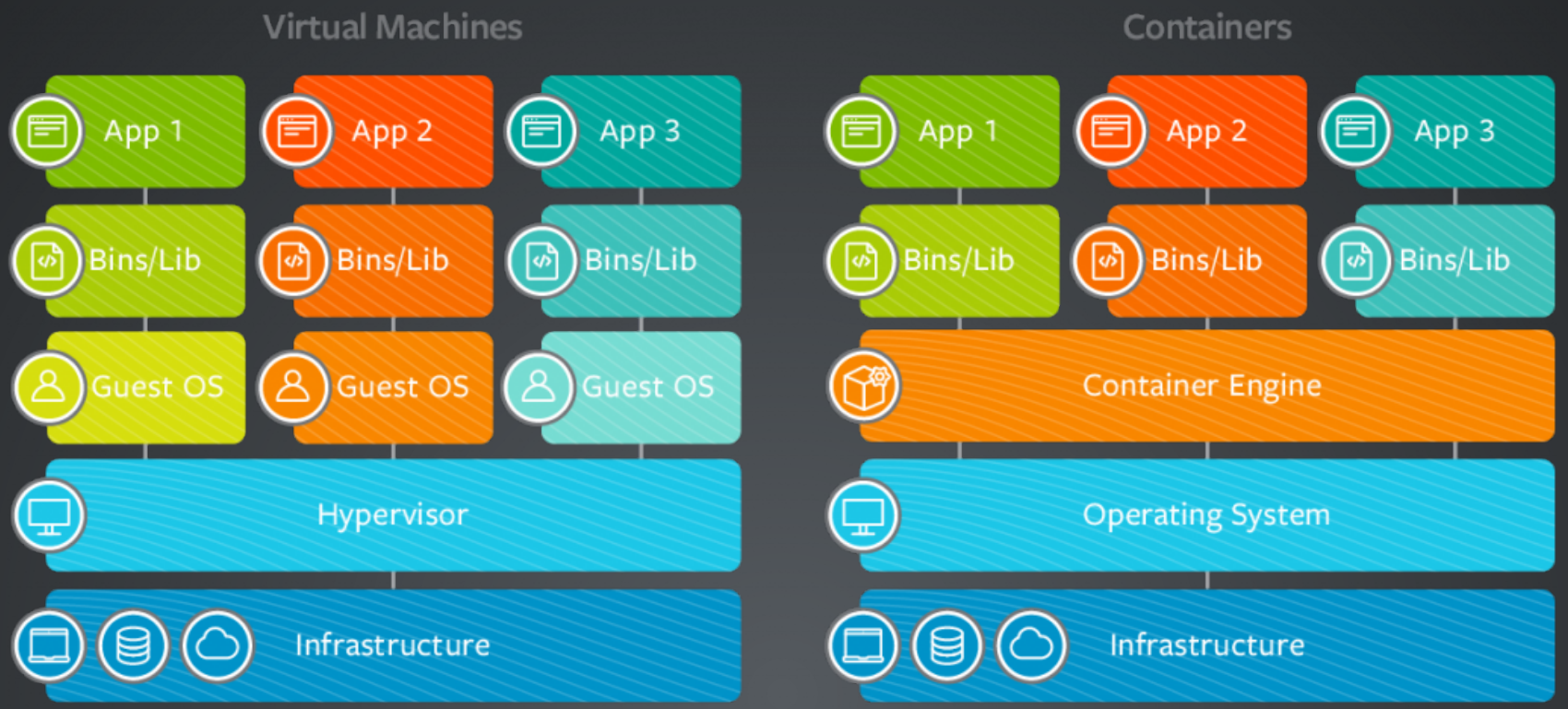
February 7, 2024

- Introduction
- Docker
- Definitions
- Docker Swarm
- Application avec Jupyterhub
- Thèmes non abordés

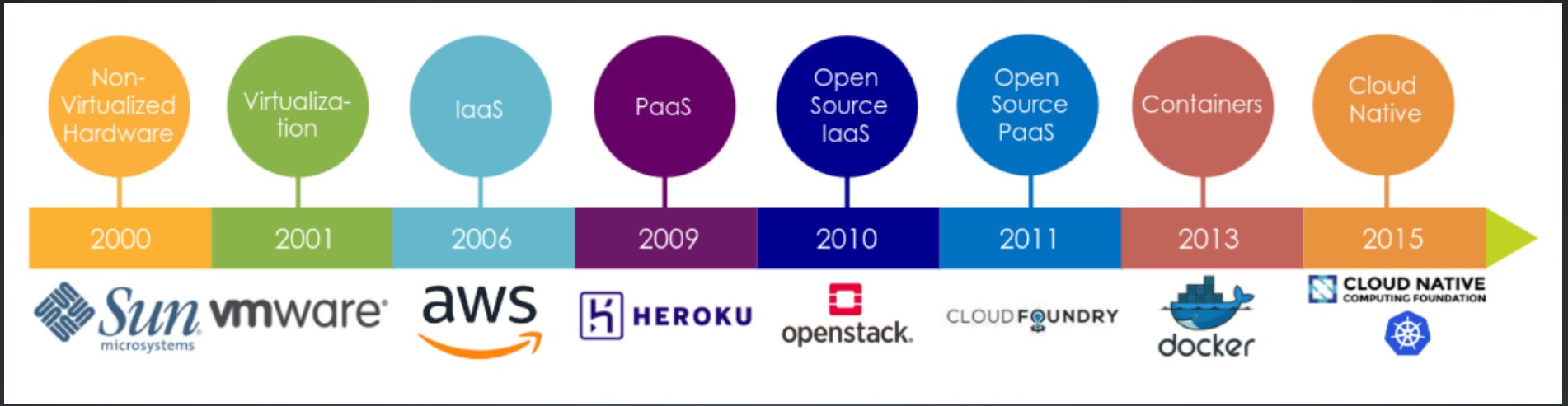
- Déploiement simple d'une plateforme `Jupyterhub` à l'aide de conteneurs `Docker` et d'un orchestrateur (`Docker Swarm`)
  - Création d'un cluster `Swarm`
  - Configuration du `Jupyterhub`
- Prérequis: connaître `Docker`



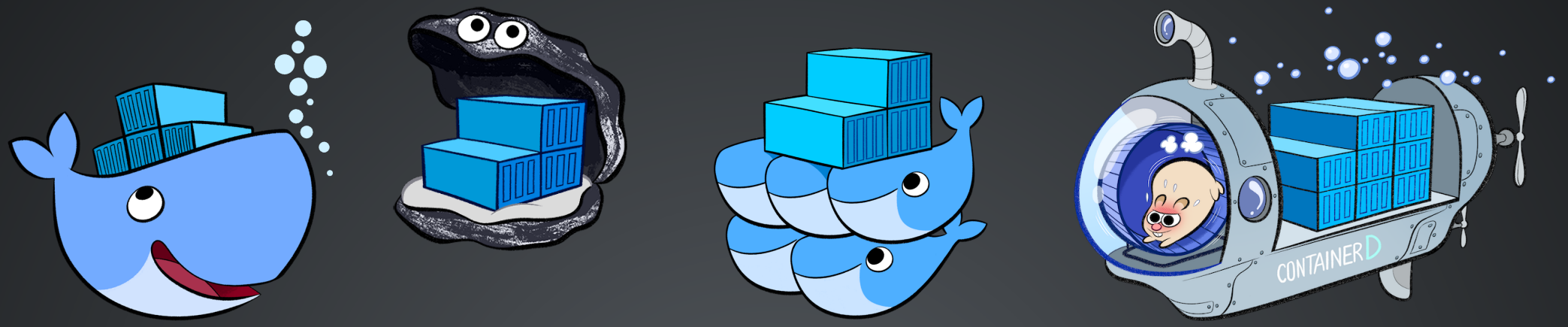
- **Conteneur**: une technologie d'isolation de processus/d'applications
- **Microservice**: un type d'architecture qui consiste à diviser une application en petits services autonomes (terminologie spécifique à l'écosystème des conteneurs)
- Même philosophie qu'une **Machine Virtuelle (VM)**: isolation, partage, réutilisation… mais plus rapide et facile!


Virtual Machines
App 1
App 2
App 3
Bins/Lib
Bins/Lib
Bins/Lib
Guest OS
Guest OS
Guest OS
Hypervisor
Infrastructure
Containers
App 1
App 2
App 3
Bins/Lib
Bins/Lib
Bins/Lib
Container Engine
Operating System
Infrastructure

- **Docker** est:
  - écrit en `Go`
  - basé sur `Union File System`
  - basé sur les `cgroups` (ressource limitation) et `namespaces` (seperated environnement)

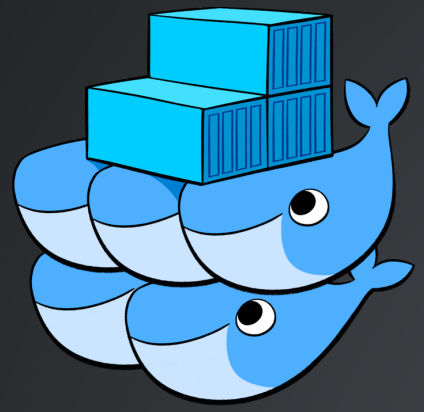


- **Docker** était au départ **dotCloud** (PaaS cloud)
  - **dotCloud**: startup créé en 2008 à Montrouge (France, 92)
  - première vesion en 2013 (`docker-0.1`)
- **Docker** est devenu un énorme projet avec plus de 3300 contributeurs



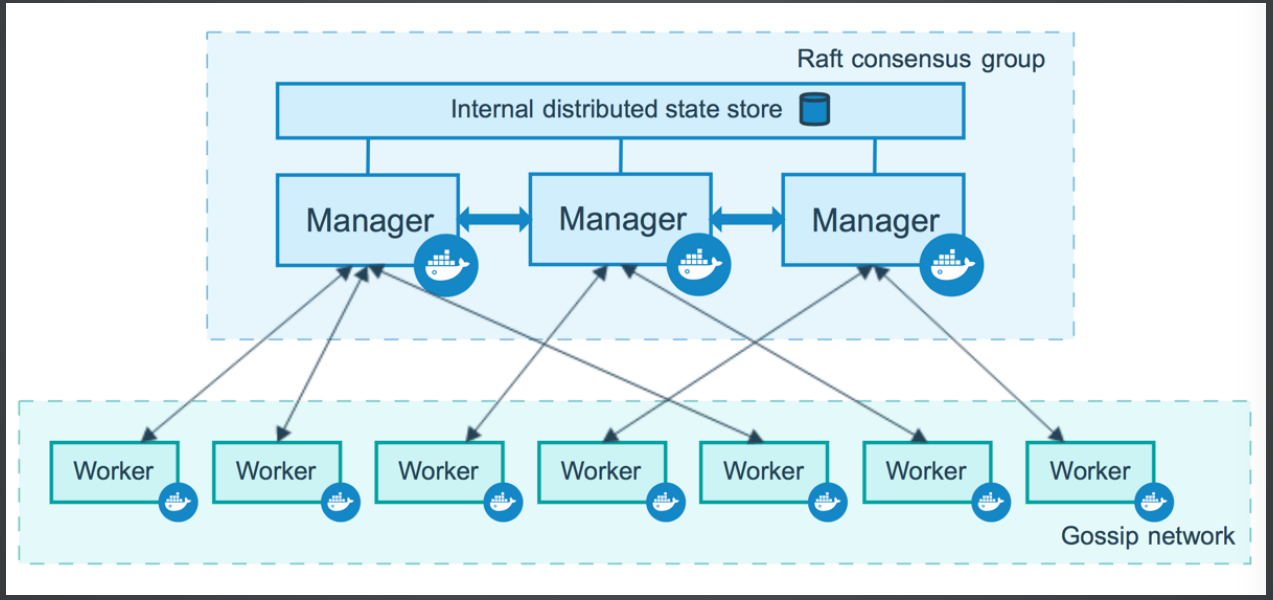
- **Docker Engine**: démon (`containerd`) et client CLI (*Command Line Interface*)
- **Docker Registry**: *registry* privée et sécurisée
- **Docker Hub**: *registry* officielle et publique
- **Docker Compose**: application pour lancer plusieurs conteneurs
- **`Docker Swarm : orchestrateur de conteneurs docker`**
- **Docker Desktop**: application pour installer docker facilement

- **Image**:
  - description statique d'un conteneur; sorte de "photographie" d'une machine
  - peut être partagée
- **Conteneur**:
  - ressemble à une machine virtuelle: peut-être démarré, arrêté, redémarré, détruit…
  - s'instancie à partie d'une image
- **Registry**: plateforme de partage d'images
- **Orchestrateur**: outil qui permet d'automatiser le déploiement, la gestion, la mise à l'échelle des conteneurs et qui assure la haute disponibilité d'une application



- `Swarm` est l'outil d'orchestration natif de Docker
- Documentation officielle: https://docs.docker.com/engine/swarm/

- Un *cluster swarm* est composé d'un ou plusieurs noeuds (serveurs) physiques ou virtuels exécutant `Docker Engine`
- Deux types de noeuds:
  - `manager`: maintient l'état du *cluster*, répartis les services sur les noeuds
  - `worker`: sert à exécuter des conteneurs



- `manager nodes`: essentiels au bon fonctionnement du cluster swarm
  - conseillé d'en avoir un nombre impair
  - trop de *manager nodes* peut réduire les performances
  - le cluster tolère une perte de $\frac{N-1}{2}$ noeuds
  - si perte du quorum: les services sur les noeuds restants sont toujours accessibles
  - par défaut, ils peuvent acceuillir des services conteneurisés commes les *workers*

Swarm Size	Majority	Fault Tolerance
1	1	0
2	2	0
3	2	**1**
4	3	1
5	3	**2**
6	4	2
7	4	**3**
8	5	3
9	5	**4**

- Installer `Docker Engine` sur tous les noeuds

Commandes Linux:

- CentOS : Official instructions

```
sudo yum install -y yum-utils device-mapper-persistent-data lvm2
sudo yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo
sudo yum install -y docker-ce docker-ce-cli containerd.io
sudo systemctl start docker
sudo usermod -aG docker $USER
```

- Ubuntu : Official instructions

```
sudo apt-get install apt-transport-https ca-certificates curl gnupg-agent softwarfe-properties-common
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
sudo apt-get install docker-ce docker-ce-cli containerd.io
sudo systemctl start docker
sudo usermod -aG docker $USER
```


- Se connecter sur un des noeuds et exécuter la commande:

```
1    docker swarm init
```

- Exécuter la commande qui s'affiche à l'écran sur d'autres noeuds pour les ajouter en tant que *worker* au cluster.
- Sinon pour ajouter à tout moment un *worker* ou un *manager* générer un `token` en exécutant sur un *manager*:

```
1    docker swarm join-token (worker|manager)
```


On peut déployer des conteneurs de deux façons:

- grâce aux `services`: distribue un seul coonteneur en un ou plusieurs exemplaires à travers le cluster

```
1    docker service create [OPTIONS] service_name
```

  Analogue à `docker run`

- grâce aux `stacks`: distribue un ensemble de coonteneurs décritsdans un fichier `Docker Compose`

```
1    docker stack deploy -c docker-compose.yml stack_name
```

  Analogue à `docker-compose`

- 4 machines virtuelles (Openstack) basée sur `Rocky 9`, avec chacune une IP publique:
  - `lpnhe-techinaire11.in2p3.fr`: serveur NFS
  - `lpnhe-techinaire12.in2p3.fr`: Noeud Swarm
  - `lpnhe-techinaire13.in2p3.fr`: Noeud Swarm
  - `lpnhe-techinaire14.in2p3.fr`: Noeud Swarm
- Une entrée DNS type A:

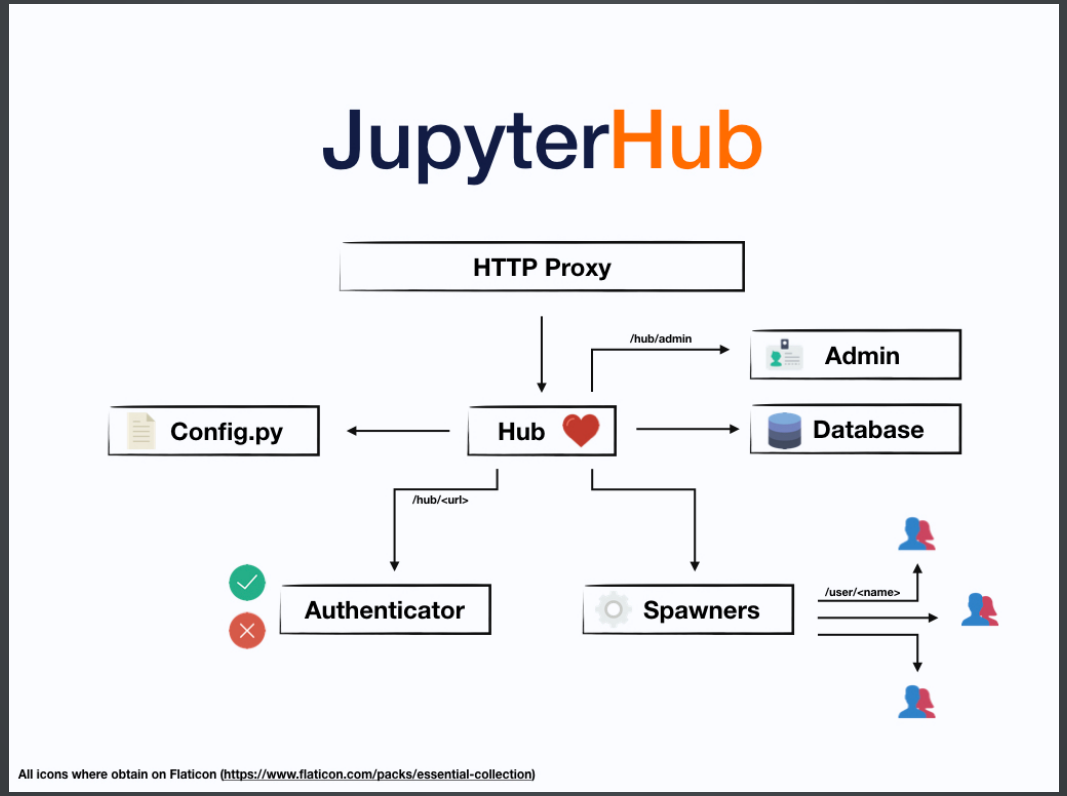
```
1   $ host lpnhe-techinaire.in2p3.fr
2   lpnhe-techinaire.in2p3.fr has address XXX.XXX.XXX.12
3   lpnhe-techinaire.in2p3.fr has address XXX.XXX.XXX.13
4   lpnhe-techinaire.in2p3.fr has address XXX.XXX.XXX.14
```

  - Chaque service a une entrée DNS automatiquement ajoutée dans le DNS interne du cluster Swarm
  - Mécanisme interne de *load balancing* qui redirige les requêtes parmi les service du cluster en se basant sur le nom DNS du sevice: `ingress`

```
1     docker swarm init       : Activer Swarm et devenir manager d'un cluster d'un seul nœud
2     docker swarm join       : Rejoindre un cluster Swarm en tant que nœud manager ou worker
3     docker service create   : Créer un service
4     docker service inspect  : Infos sur un service
5     docker service ls       : Liste des services
6     docker service rm       : Supprimer un service
7     docker service scale    : Modifier le nombre de conteneurs qui fournissent un service
8     docker service ps       : Liste et état des conteneurs qui fournissent un service
9     docker service update   : Modifier la définition d'un service
10    docker stack deploy     : Déploie une stack ou update une stack existante
11    docker stack ls         : Liste les stacks
12    docker stack ps         : Liste l'état du déploiement d'une stack
13    docker stack rm         : Supprimer une ou des stacks
14    docker stack services   : Liste les services qui composent une stack
15    docker node inspect     : Informations détaillées sur un nœud
16    docker node ls          : Liste les nœuds
17    docker node ps          : Liste les tâches en cours sur un nœud
18    docker node promote     : Transforme un nœud worker en manager
19    docker node demote      : Transforme un nœud manager en worker
```


- Documentation officielle: https://jupyterhub.readthedocs.io/
- Repository GitHub: https://github.com/jupyterhub/jupyterhub

- `Jupyterhub` peut être vu comme un “fournisseur” de *notebooks Jupyter* individuels pour chaque utilisateur qui se connecte
- Il est constitué de 4 sous-systèmes: un `hub`, d’un `proxy http`, de `notebooks` utilisateurs et d’une classe d’authentification



- `jupyterhub_config.py` fichier de configuration du Jupyter

```
1    docker run --rm -it -v $HOME/jupyterhub/:/srv/jupyterhub jupyterhub/jupyterhub bash
```

  - type de spawnwer: `SwarmSpwaner`, `DockerSpwaner`...
  - type d'authentification: `PAM` (par défaut), `ldap`, `OAuth`...
  - type de base de données: `sqlite` (par défaut), `PostgreSQL`, `MariaDB`...
  - ports utilisés; port par défaut de l'interface web: `8000`
  - type de notebooks accessibles aux utilisateurs

- Importer les fichiers de configurations type `jupyterhub_config.py` dans swarm:

```
docker config create <config_name> <file_name>
```

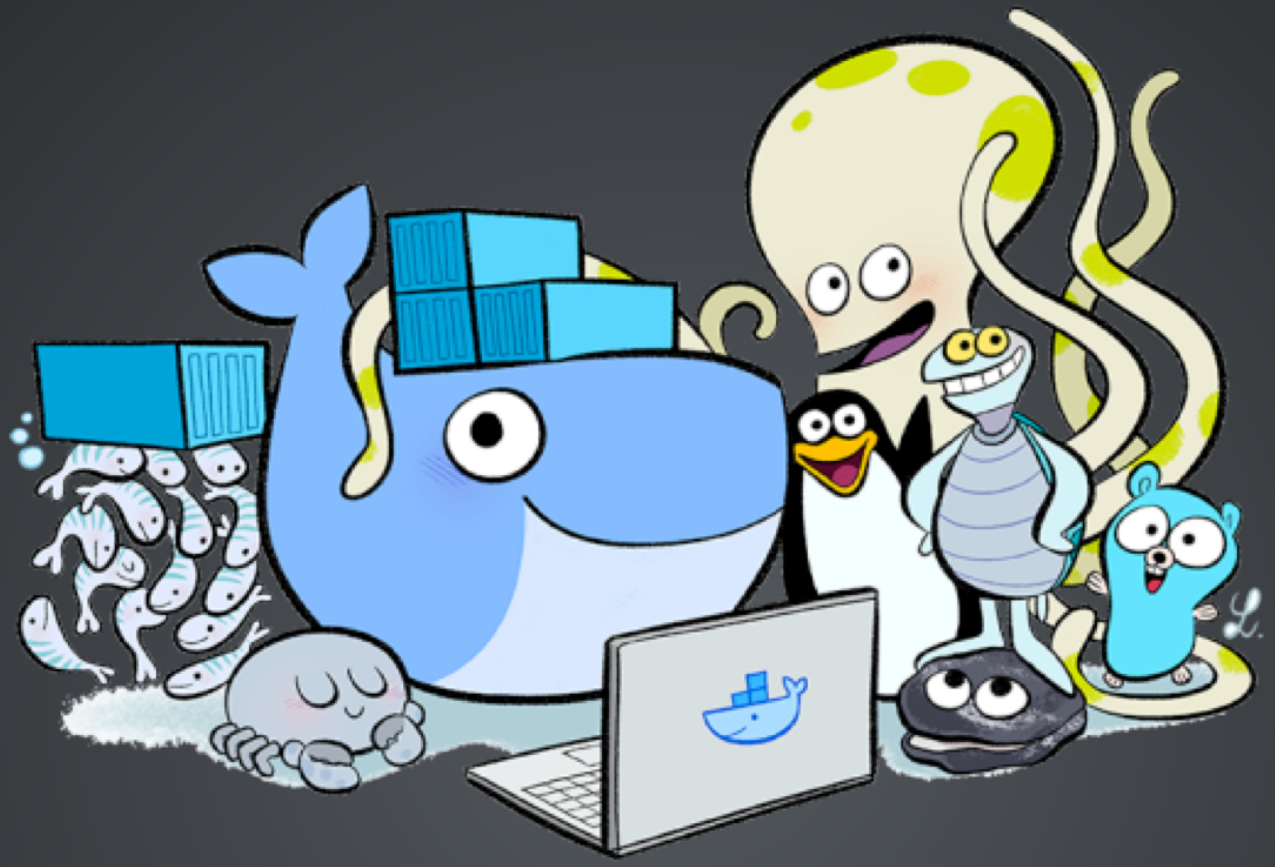
  Pour des données plus sensibles, type certificats utiliser plutôt:

```
docker secret create <secret_name> <file_name>
```

- Écrire le fichier `docker-compose.yml`:
  - le `hub` ne peut-être déployé que sur un `noeud manager` en un seul exmplaire
  - quel réseau Docker utiliser (type `overlay`)
  - montage ou non de répertoire dans les conteneurs (option `-v` de `docker run`)
  - `mapping` des ports (option `-p` de `docker run`)
- Déployer la `stack`:

```
docker stack deploy -c <docker-compose_file> <stack_name>
```


- HTTPS (SSL)
- Loadbalancer/reverse proxy externe: `traefik`, `nginx-ingress`
- Base de données non `SQLite`
- *Metrics* avec `prometheus` et `grafana`
- OS: `Fedora CoreOS`
- Personnalisation des images `notebook`



Repository git can be found here.