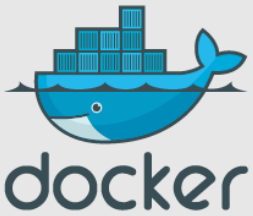


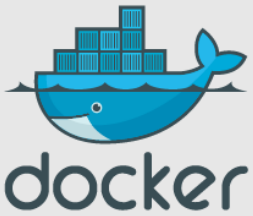
Webinaire Docker : retour d'expérience

Cécile Cavet

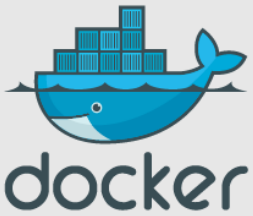
ccavet at apc.in2p3.fr

Centre François Arago (FACe), Laboratoire APC, Université Paris Diderot

16 Juin 2016



Plan



Cas d'utilisation de Docker à l'APC

Projets scientifiques :

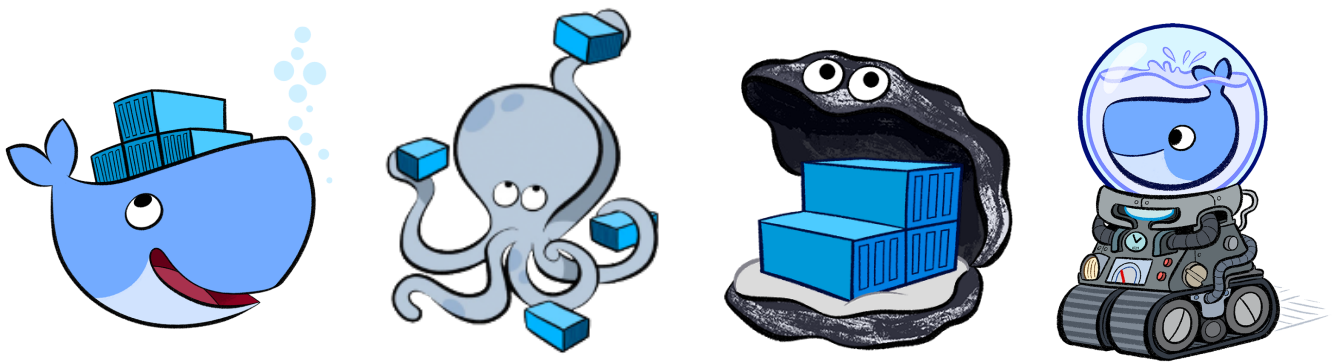
➟ LISAPathfinder (2015, succés !) / eLISA (2029) : en charge du proto-DPC (Data Processing Center)

- conteneurisation et partage des applications :
  - LPF : analyse des données de LISAPathfinder.
  - LAL : analyse des données de LIGO.
  - LISACode : simulateur d'ondes gravitationnelles.
- mise en place de services :
  - registre privé.
  - intégration continue avec Jenkins et Sonar.
- workflow : gestion de pics de charge ➟ R&T CNES/Atos.

Cas d'utilisation de Docker à l'APC

Projets scientifiques :

➙ **Besoin de la boite à outils Docker :**
Engine, Compose, Registry, Machine...



Cas d'utilisation de Docker à l'APC

Mode d'utilisation :

- sur le cloud : VM Linux + **Docker Engine**
- en local (Mac OS X) : **Docker Machine** + VM.
  - Virtualisation : VirtualBox.
  - OS : Boot2Docker.

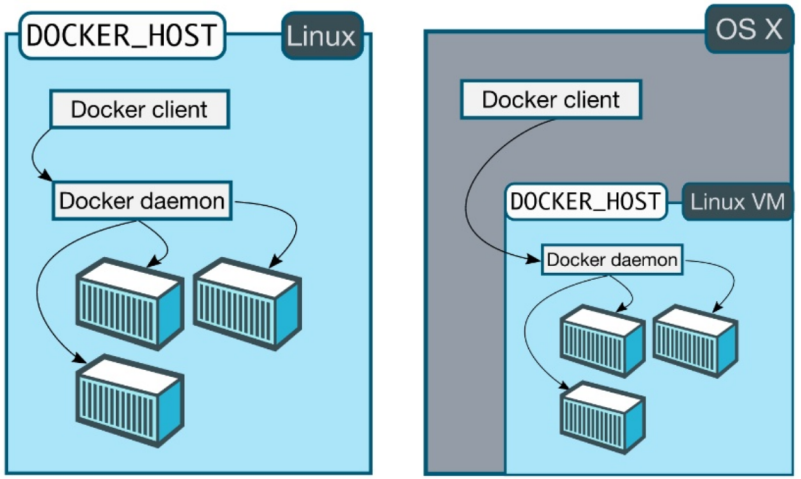


Figure: Linux vs OS X @Docker.

1ère utilisation : conteneurisation et partage des applications

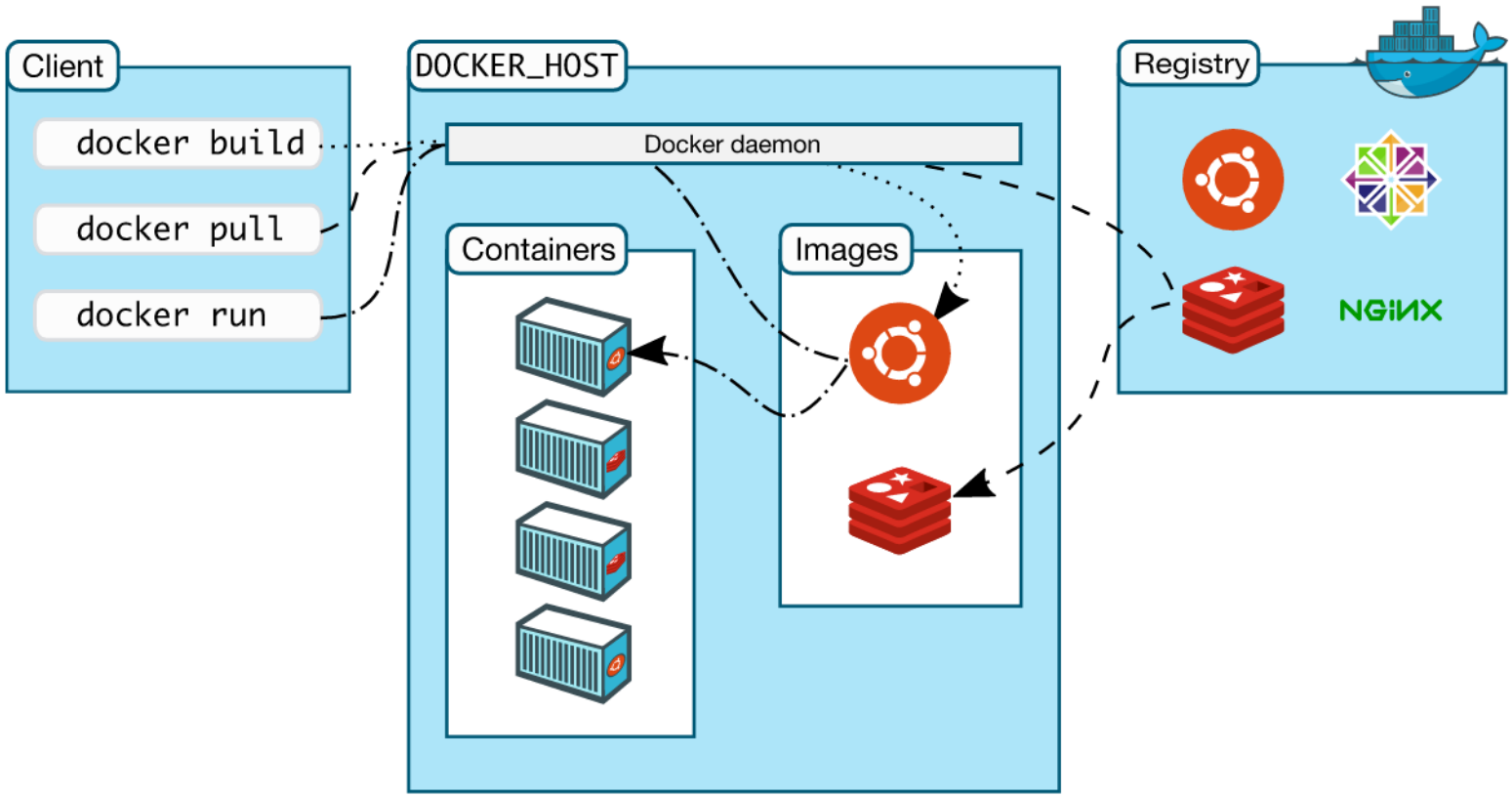
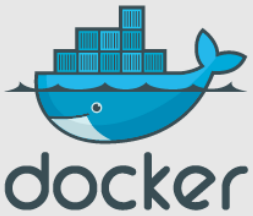


Figure: Cycle de vie @Docker.



Conteneurisation et partage des applications

Dockerfile :

- Conteneurisation : création d'images Docker.
  - Code scientifique (LISACode).
  - Librairies (hdf5...).
  - Librairies spécifiques (LAL).
- Format spécifique (proche script shell).

```
1  FROM centos:latest
2  MAINTAINER eLISA DPC ccavet@apc.in2p3.fr
3
4  RUN yum -y update
5  RUN yum install -y epel-release
6  RUN yum install -y git
7
8  RUN pip install --upgrade pip
9  RUN pip install gcovr
10
11 ENV CXX c++
12
13 # install libcmaes
14 RUN git clone https://github.com/beniz/libcmaes.git
15 RUN cd libcmaes && ./autogen.sh && ./configure --prefix=/usr/local/libcmaes && make && make install
16
17 # install swig
18 RUN wget https://sourceforge.net/projects/swig/files/swig/swig-3.0.8/swig-3.0.8.tar.gz && tar xvzf swig-3.0.8.tar.gz
19 RUN cd swig-3.0.8 && ./configure && make && make install
20
21 # install LAL
22 ENV LALDIR /workspace/Applications/LAL
23 ENV LALSUITE_SRCDIR ${LALDIR}/src
24 ENV LALSUITE_PREFIX ${LALDIR}/opt/lalsuite
25 ENV LSCSOFT_PREFIX ${LALDIR}/opt/lscsoft
```

Création d'images Docker

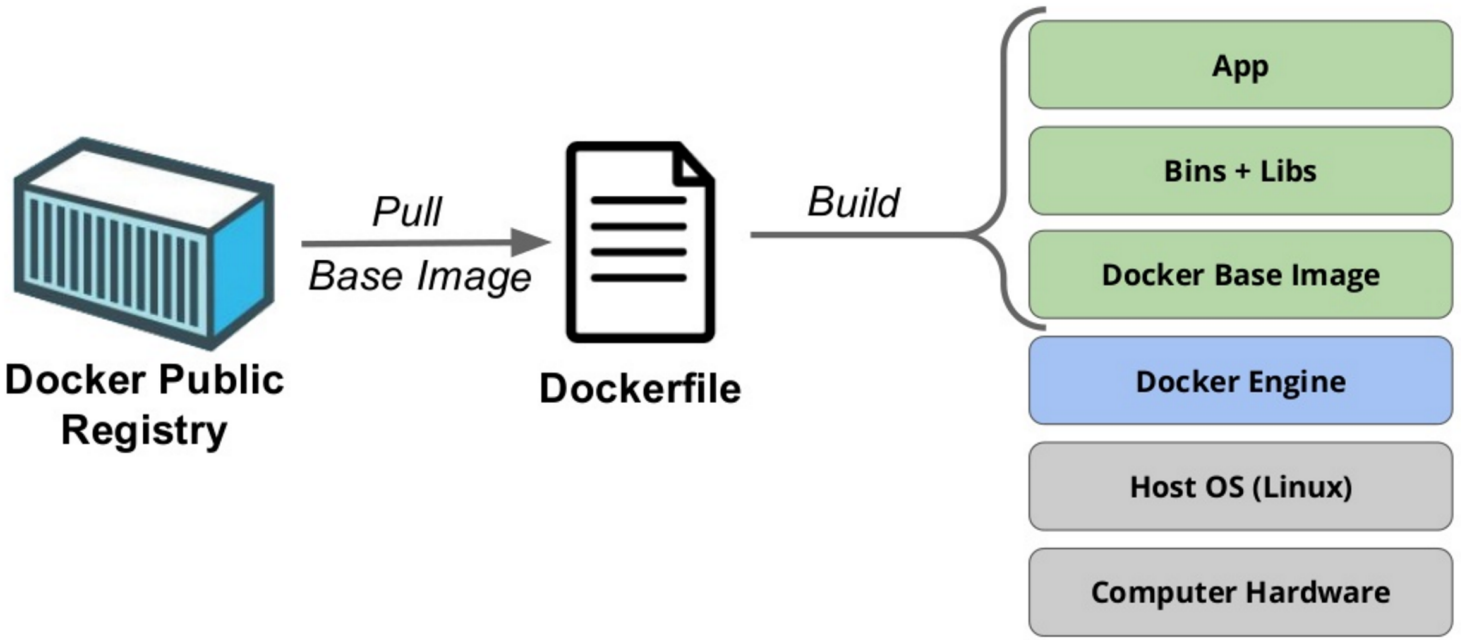
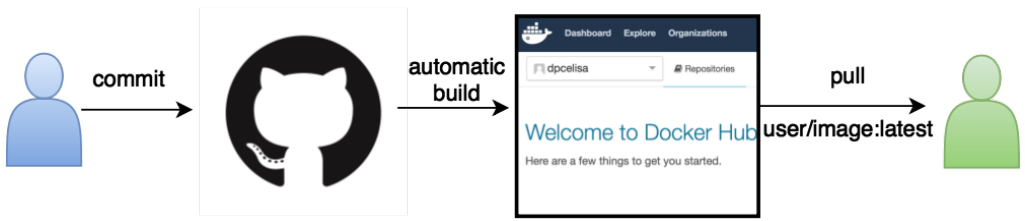


Figure: Chaîne de processus @ `http://fr.slideshare.net/s-brinkmann/locally-it-worked-virtualizing-docker`.

Conteneurisation et partage des applications

Docker Hub : Registry public

- Partage des images Docker :
  - SaaS : service hébergé sur le cloud.
- Construction (build) automatique d'images :
  - Dépôt public : GitHub ou Bitbucket pour le **Dockerfile** et le Readme.
  - Construction de l'image quand nouveau commit sur le dépôt.
  - Branches : à la demande (par ex. latest / develop).



Docker Hub

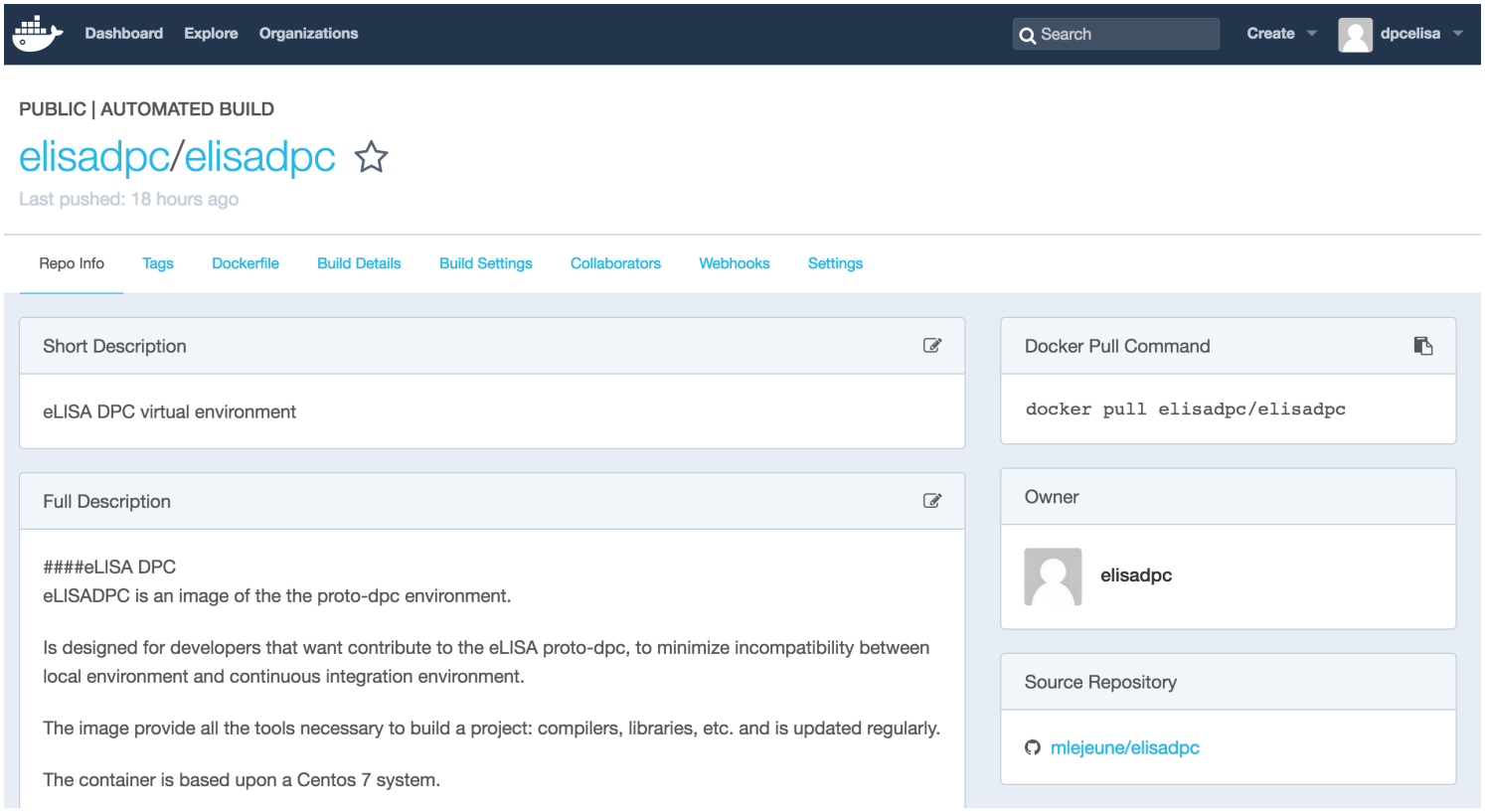
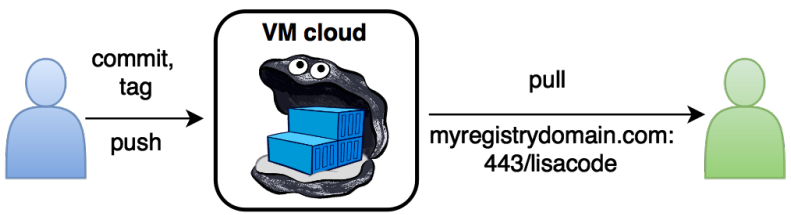


Figure: Compte public du DPC de LISA.

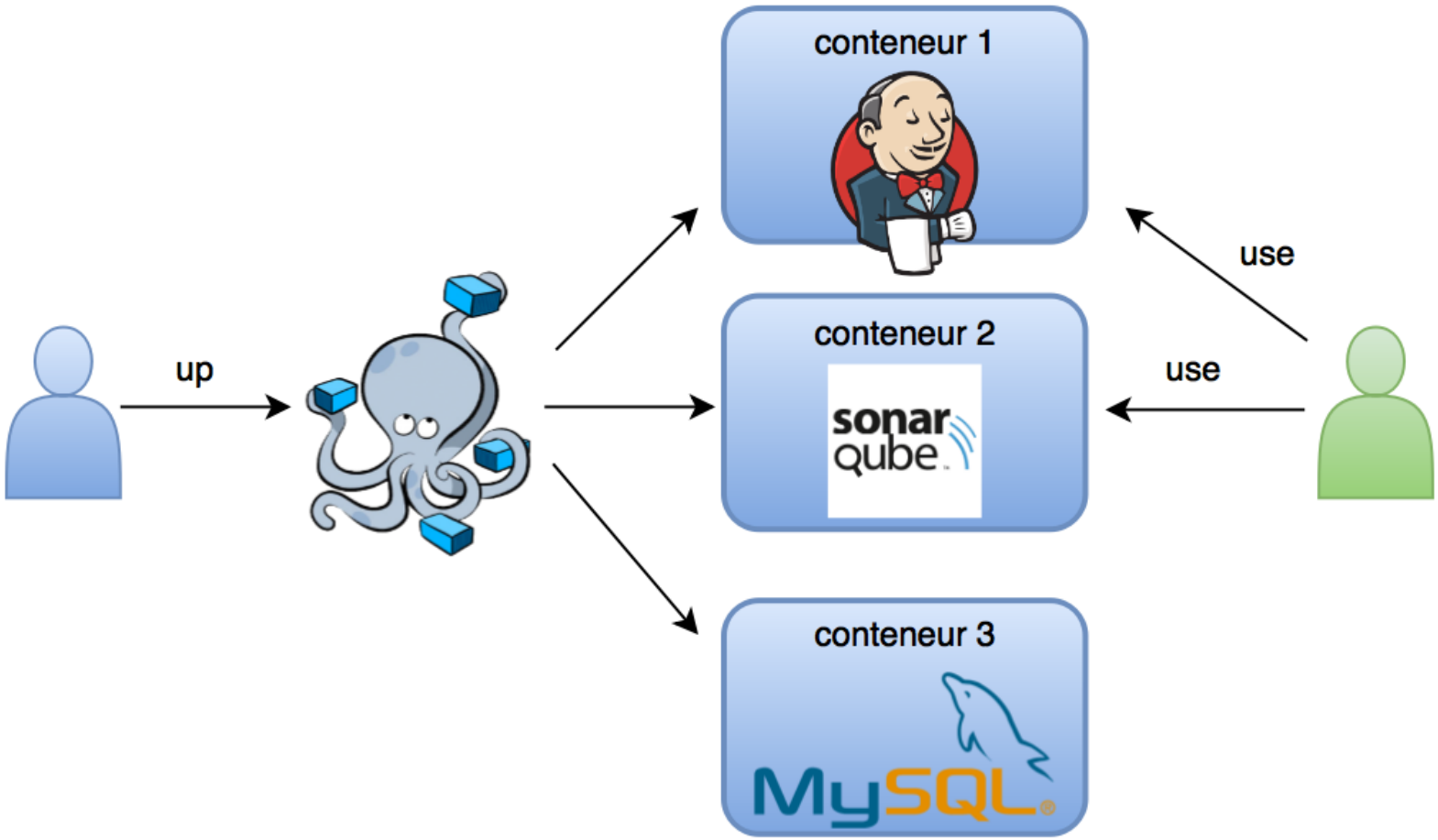
Conteneurisation et partage des applications

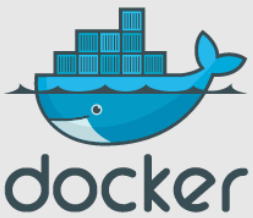
Registre privé :

- Partage des images Docker :
  - Code non public.
  - Création d'images manuelle.
- VM cloud :
  - Serveur Web simple (sans interface) avec **Docker Compose** : conteneurs ➡ Registry et Nginx.
  - Sécurité : certificat.
  - Nom de domaine : myregistrydomain.com ➡ IP.



2ème utilisation : intégration continue



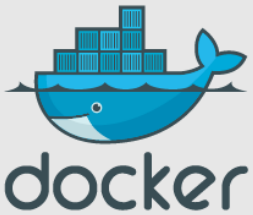


Intégration continue

Docker Compose :

- Multi-conteneurs reliés.
- Image : construction à la volée ou utilisation d'un registre (public, privé, local (Docker Host)).
- Format : YAML.

```yaml
version: '2'
services:
  sonar:
    image: sonarqube
    container_name: sonar
    network_mode: "host"
    environment:
      - SONARQUBE_JDBC_USERNAME=root
      - SONARQUBE_JDBC_PASSWORD=mypassword
      - SONARQUBE_JDBC_URL=jdbc:mysql://localhost:3306/sonar?useUnicode=true&characterEncoding=utf8&rewriteBatchedStatements=true
    ports:
      - "9000:9000"
      - "9092:9092"
      - "3306:3306"
  mysql:
    image: mysql
    container_name: mysql
    network_mode: "service:sonar"
    environment:
      - MYSQL_DATABASE=sonar
      - MYSQL_USER=sonar
      - MYSQL_PASSWORD=sonarpassword
      - MYSQL_ROOT_PASSWORD=mypassword
  myjenkins:
```



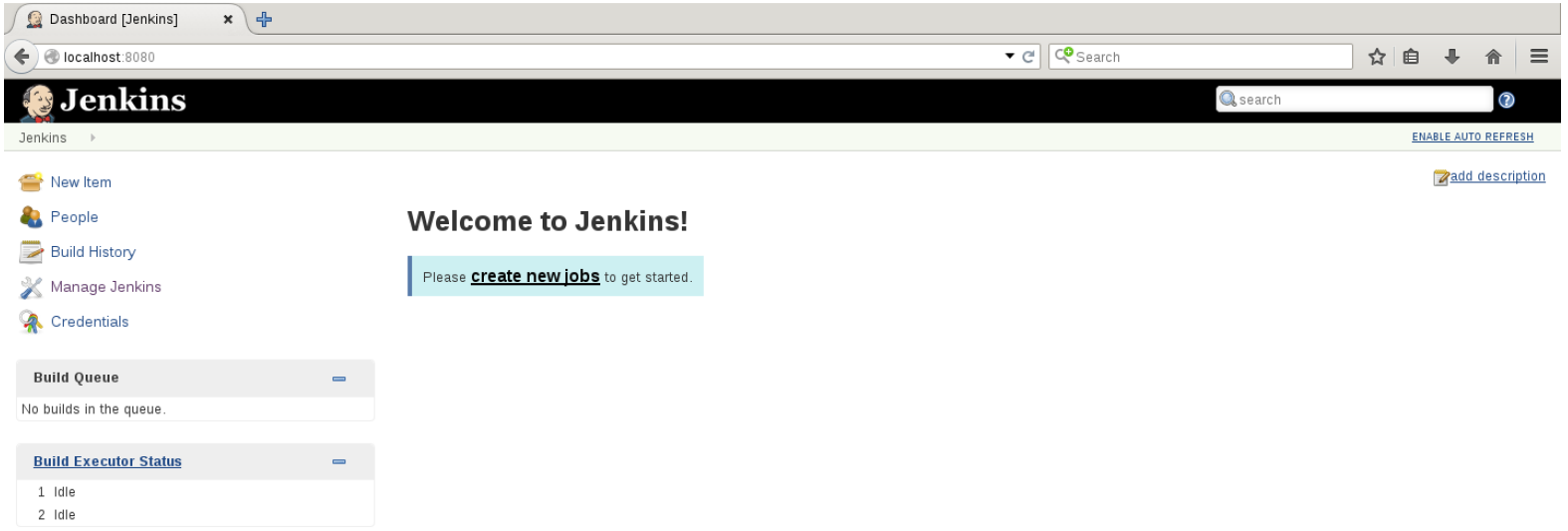
Intégration continue

Docker Compose :

- Même instructions que le CLI Docker (liens, ports, volumes...).
- Conteneurs Jenkins, Sonar et MySQL :

```
$ docker-compose up -d
$ docker-compose ps
  Name          Command          State     Ports
-----------------------------------------------
myjenkins /bin/tini -- /usr... Up 0.0.0.0:8080
mysql     entrypoint.sh mysqld Up
sonar     ./bin/run.sh         Up 0.0.0.0:9000
```

Interface Web de Jenkins



3ème utilisation : gestion de pics de charge

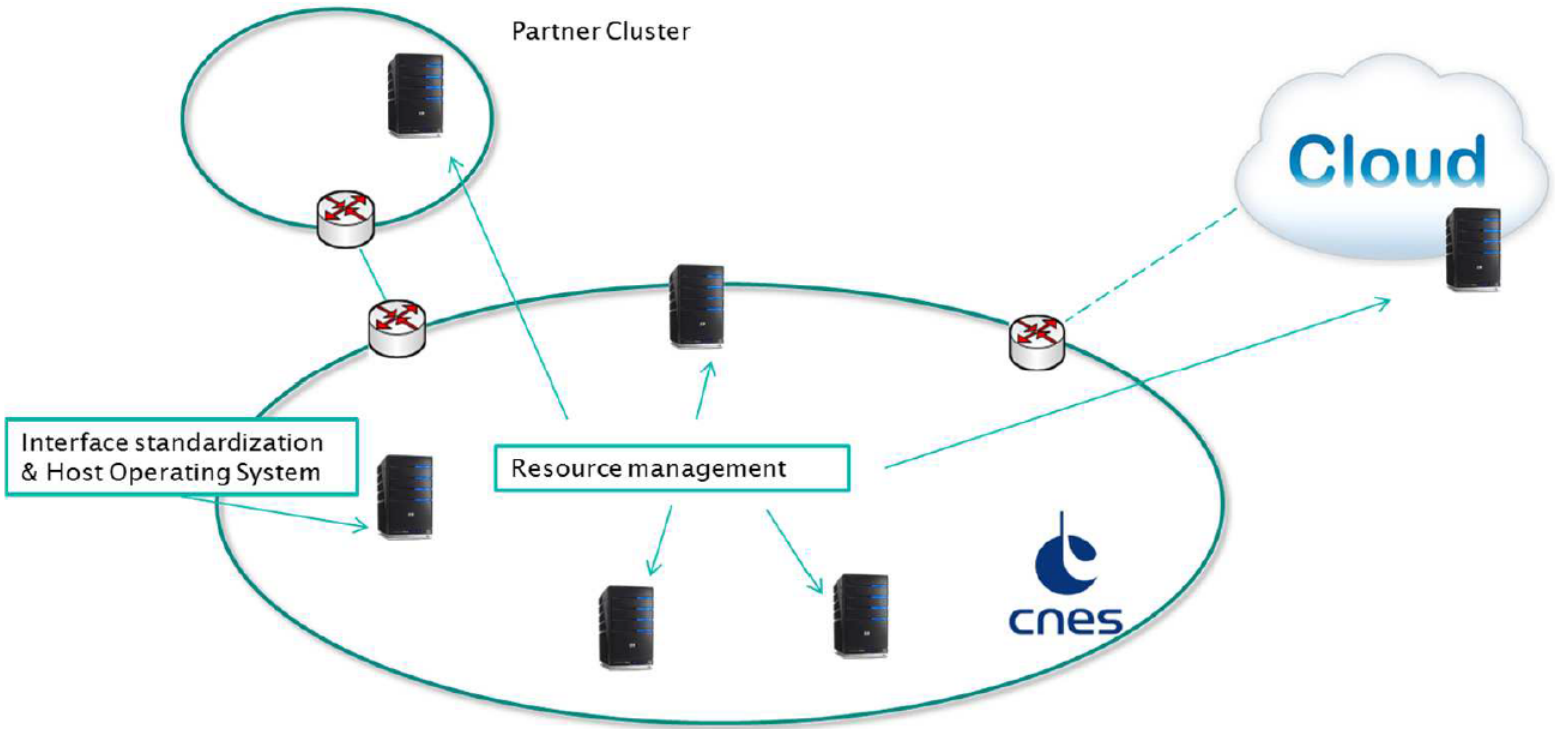
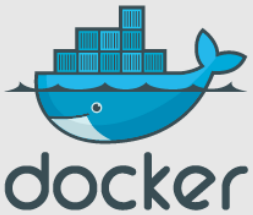


Figure: R&T APC/CNES/Atos @M. Poncet, BiD'S 16.



Gestion de pics de charge

Multi-outils :

- Conteneurisation d'une application : LPF
  - code privé (industriel).
  - algorithme MCMC (MATLAB/C++), parallèle (MPI), librairies (LAPACK...).
- Partage de l'image : registre privé sur une VM cloud.
- Exécution hybride : cluster HPC du CNES / VM du cloud Helix Nebula (Atos) + conteneurs.
- Gestion des ressources : Mesos ; des jobs Docker : Chronos.

  Ref. : Poncet et al., Enabling collaboration between space agencies using private and cloud based clusters, BiDS'16 (2016)

Workflow

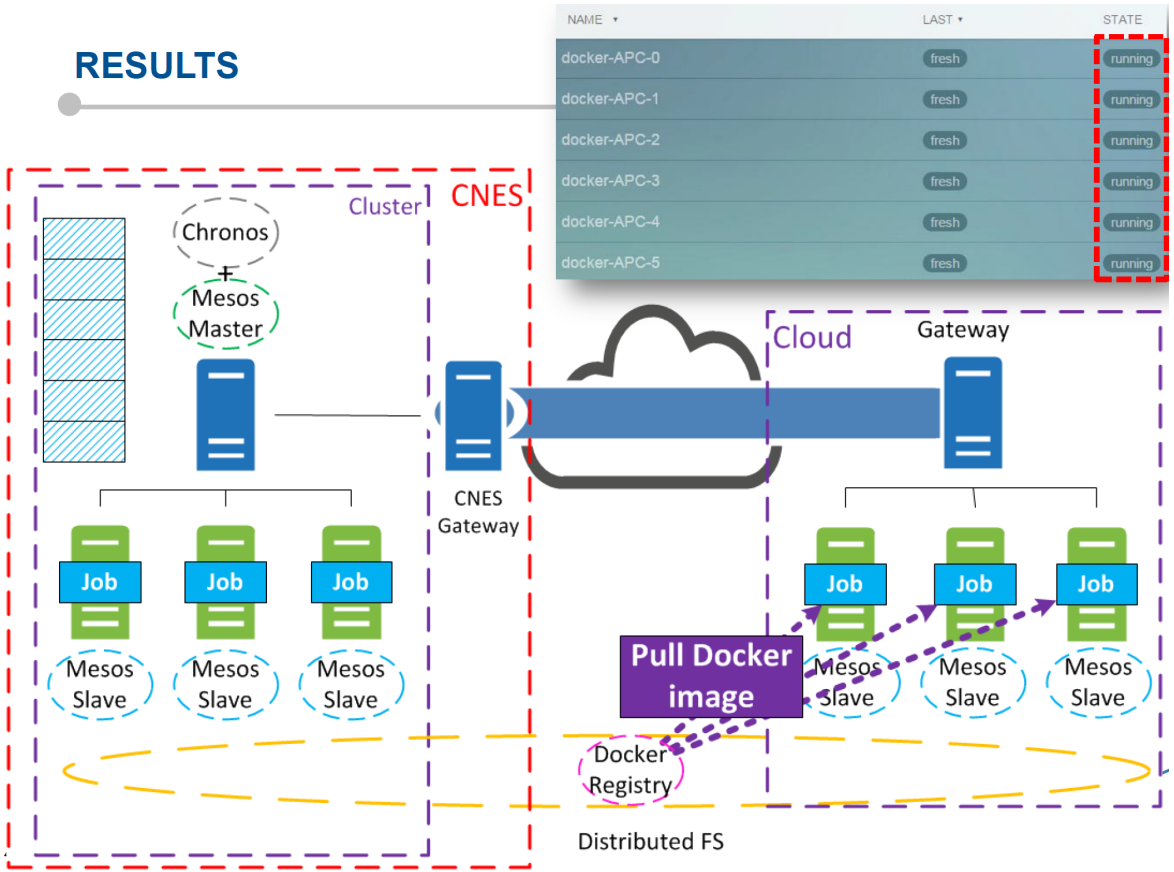


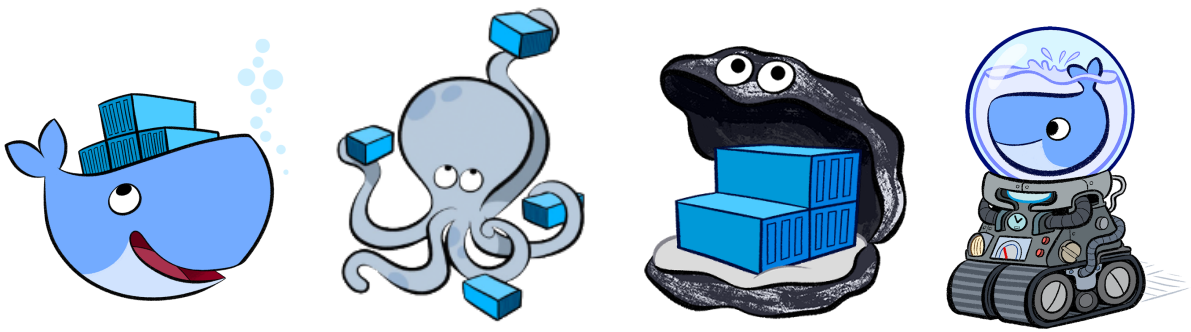
Figure: R&T APC/CNES/Atos @M. Poncet, BiD'S 16.

Conclusions

Expérience très positive :

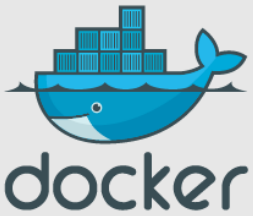
- multi-applications.
- multi-services.
- multi-workflows.

  ➙ **Docker permet de faire tout !**



- beaucoup de doc, tuto, blog, meetup...

  ➙ **Enorme communauté.**



Conclusions

Quelques problèmes :

- registre privé pas encore complet.
- pas natif sur Mac et complexité pour les utilisateurs : OS local + OS VM + conteneurs Docker.
- mais dans un futur (proche) :
  - sur le cloud : gestion directe des conteneurs Docker.
  - en local (Mac OS X) : petite VM (Unikernels).

Future :

- construction automatique des images avec Jenkins (CloudBees pluging).

Questions ?