



Contexte

Conteneurisation et partage

Dockerfile
DockerHub
Registre privé

Intégration continue

Docker Compose

Gestion de pics de charge

Workflow

Conclusion

Webinaire Docker : retour d'expérience

Cécile Cavet

ccavet at apc.in2p3.fr

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

Labex

UnivEarthS



USPC
Université Sorbonne
Paris Cité

16 Juin 2016



Plan

Contexte

Conteneurisation et partage

Dockerfile

DockerHub

Registre privé

Intégration continue

Docker Compose

Gestion de pics de charge

Workflow

Conclusion

- 1 Contexte
- 2 Conteneurisation et partage
- 3 Intégration continue
- 4 Gestion de pics de charge
- 5 Conclusion



Cas d'utilisation de Docker à l'APC

Contexte

Conteneurisation et partage

Dockerfile
DockerHub
Registre privé

Intégration continue

Docker Compose

Gestion de pics de charge

Workflow

Conclusion

Projets scientifiques :

➔ [LISAPathfinder](#) (2015, succès !) / [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

Contexte

Conteneurisation et partage

- Dockerfile
- DockerHub
- Registre privé

Intégration continue

- Docker Compose

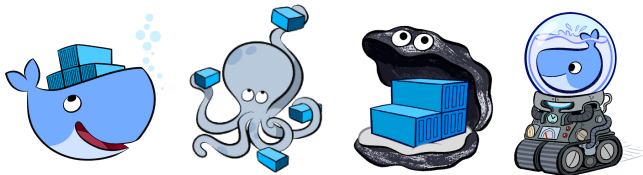
Gestion de pics de charge

- Workflow

Conclusion

Projets scientifiques :

➔ **Besoin de la boîte à outils Docker :**
Engine, Compose, Registry, Machine...





Cas d'utilisation de Docker à l'APC

Contexte

Conteneurisation et partage

Dockerfile
DockerHub
Registre privé

Intégration continue

Docker Compose

Gestion de pics de charge

Workflow

Conclusion

Mode d'utilisation :

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

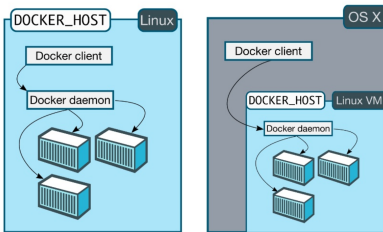


Figure: Linux vs OS X @Docker.



1ère utilisation : conteneurisation et partage des applications

Contexte

Conteneurisation
et partage

Dockerfile

DockerHub

Registre privé

Intégration
continue

Docker Compose

Gestion de pics
de charge

Workflow

Conclusion

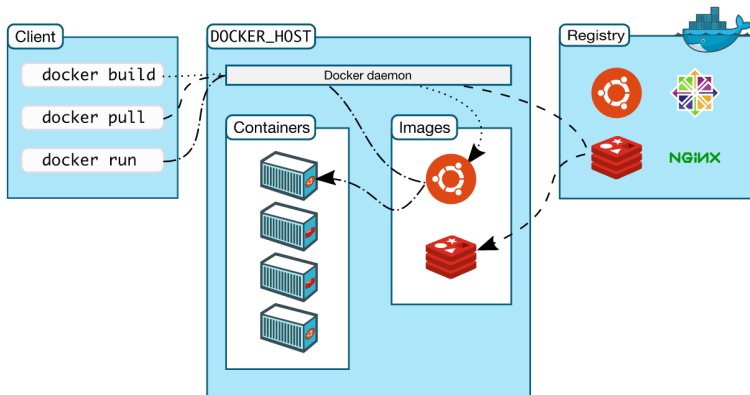


Figure: Cycle de vie @Docker.



Conteneurisation et partage des applications

Contexte

Conteneurisation
et partage

Dockerfile

DockerHub

Registre privé

Intégration
continue

Docker Compose

Gestion de pics
de charge

Workflow

Conclusion

Dockerfile :

- Conteneurisation : création d'images Docker.
 - Code scientifique (LISACode).
 - Bibliothèques (hdf5...).
 - Bibliothèques 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 libmaes
14 RUN git clone https://github.com/beniz/libmaes.git
15 RUN cd libmaes && ./autogen.sh && ./configure --prefix=/usr/local/libmaes && 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 xvfz 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

Contexte

Conteneurisation
et partage

Dockerfile

DockerHub

Registre privé

Intégration
continue

Docker Compose

Gestion de pics
de charge

Workflow

Conclusion

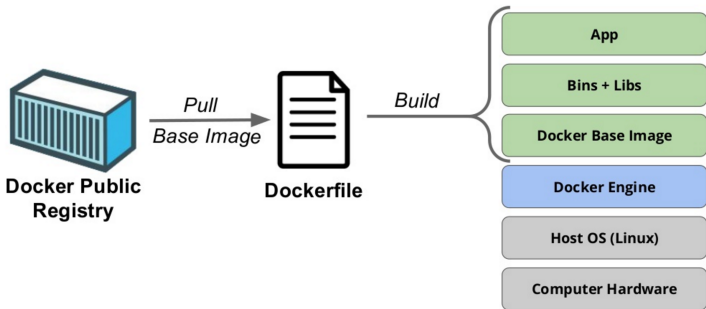


Figure: Chaîne de processus @ <http://fr.slideshare.net/s-brinkmann/locally-it-worked-virtualizing-docker>.



Conteneurisation et partage des applications

Contexte

Conteneurisation
et partage

Dockerfile

DockerHub

Registre privé

Intégration
continue

Docker Compose

Gestion de pics
de charge

Workflow

Conclusion

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

Contexte

Conteneurisation et partage

Dockerfile

DockerHub

Registre privé

Intégration continue

Docker Compose

Gestion de pics de charge

Workflow

Conclusion

The screenshot shows the Docker Hub interface for the public repository `elisadpc/elisadpc`. The page header includes navigation links (Dashboard, Explore, Organizations), a search bar, and a user profile (dpcelisa). The repository is marked as 'PUBLIC | AUTOMATED BUILD' and 'Last pushed: 18 hours ago'. The main content area is divided into two columns. The left column contains a 'Short Description' (eLISA DPC virtual environment) and a 'Full Description' (starting with '###eLISA DPC' and describing the image's purpose for development and CI/CD). The right column contains a 'Docker Pull Command' (docker pull elisadpc/elisadpc), the 'Owner' (elisadpc), and the 'Source Repository' (miejeune/elisadpc).

Dashboard Explore Organizations Search Create dpcelisa

PUBLIC | AUTOMATED BUILD
elisadpc/elisadpc ☆
Last pushed: 18 hours ago

Repo Info Tags Dockerfile Build Details Build Settings Collaborators Webhooks Settings

Short Description ✎
eLISA DPC virtual environment

Full Description ✎
###eLISA DPC
eLISADPC is an image of the the proto-dpc environment.

Is designed for developers that want contribute to the eLISA proto-dpc, to minimize incompatibility between local environment and continuous integration environment.

The image provide all the tools necessary to build a project: compilers, libraries, etc. and is updated regularly.

The container is based upon a Centos 7 system.

Docker Pull Command ✎
docker pull elisadpc/elisadpc

Owner
elisadpc

Source Repository
miejeune/elisadpc

Figure: Compte public du DPC de LISA.



Conteneurisation et partage des applications

Contexte

Conteneurisation
et partage

Dockerfile

DockerHub

Registre privé

Intégration
continue

Docker Compose

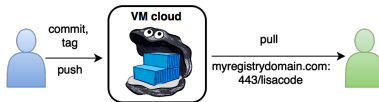
Gestion de pics
de charge

Workflow

Conclusion

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

Contexte

Conteneurisation et partage

- Dockerfile
- DockerHub
- Registre privé

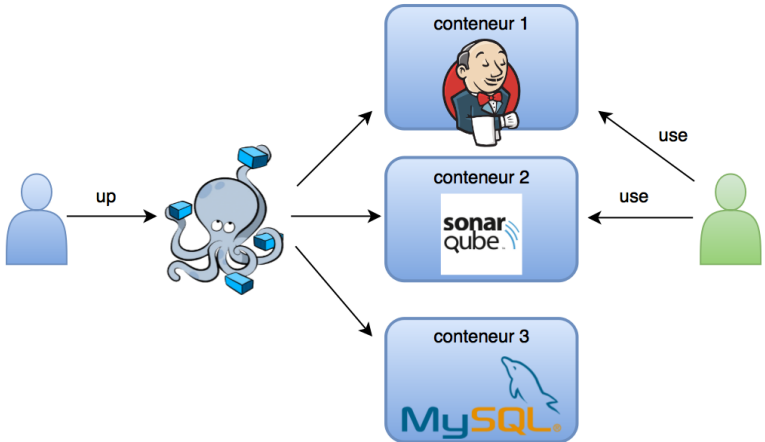
Intégration continue

- Docker Compose

Gestion de pics de charge

- Workflow

Conclusion





Intégration continue

Contexte

Conteneurisation
et partage

Dockerfile

DockerHub

Registre privé

Intégration
continue

Docker Compose

Gestion de pics
de charge

Workflow

Conclusion

Docker Compose

- Multi-conteneurs reliés.
- Image : construction à la volée ou utilisation d'un registre (public, privé, local (Docker Host)).
- Format : 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"
      - "9002:9002"
      - "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=myspassword
  myjenkins:
```



Intégration continue

Contexte

Conteneurisation
et partage

Dockerfile
DockerHub
Registre privé

Intégration
continue

Docker Compose

Gestion de pics
de charge

Workflow

Conclusion

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 mysql	Up	
sonar	./bin/run.sh	Up	0.0.0.0:9000



Interface Web de Jenkins

Contexte

Conteneurisation
et partage

Dockerfile

DockerHub

Registre privé

Intégration
continue

Docker Compose

Gestion de pics
de charge

Workflow

Conclusion

Dashboard [Jenkins]

localhost:8080

search

Jenkins

ENABLE AUTO REFRESH

[add description](#)

Welcome to Jenkins!

Please [create new jobs](#) to get started.

Build Queue

No builds in the queue.

Build Executor Status

1	Idle
2	Idle



Conclusion

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



Gestion de pics de charge

Contexte

Conteneurisation
et partage

Dockerfile

DockerHub

Registre privé

Intégration
continue

Docker Compose

Gestion de pics
de charge

Workflow

Conclusion

Multi-outils :

- Conteneurisation d'une application : LPF
 - code privé (industriel).
 - algorithme MCMC (MATLAB/C++), parallèle (MPI), bibliothèques (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

Contexte

Conteneurisation et partage

Dockerfile

DockerHub

Registre privé

Intégration continue

Docker Compose

Gestion de pics de charge

Workflow

Conclusion

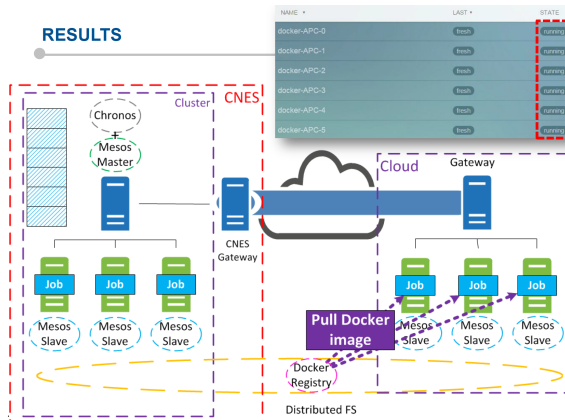


Figure: R&T APC/CNES/Atos ©M. Poncet, BiD'S 16.



Conclusions

Contexte

Conteneurisation
et partage

Dockerfile

DockerHub

Registre privé

Intégration
continue

Docker Compose

Gestion de pics
de charge

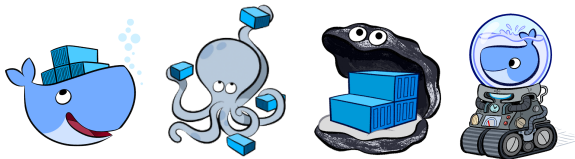
Workflow

Conclusion

Expérience très positive :

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

➔ **Docker permet de faire tout !**



- beaucoup de doc, tuto, blog, meetup...
➔ **Enorme communauté.**



Conclusions

Contexte

Conteneurisation et partage

Dockerfile
DockerHub
Registre privé

Intégration continue

Docker Compose

Gestion de pics de charge

Workflow

Conclusion

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 plugging).



Questions ?

Contexte

Conteneurisation et partage

- Dockerfile
- DockerHub
- Registre privé

Intégration continue

- Docker Compose

Gestion de pics de charge

- Workflow

Conclusion

