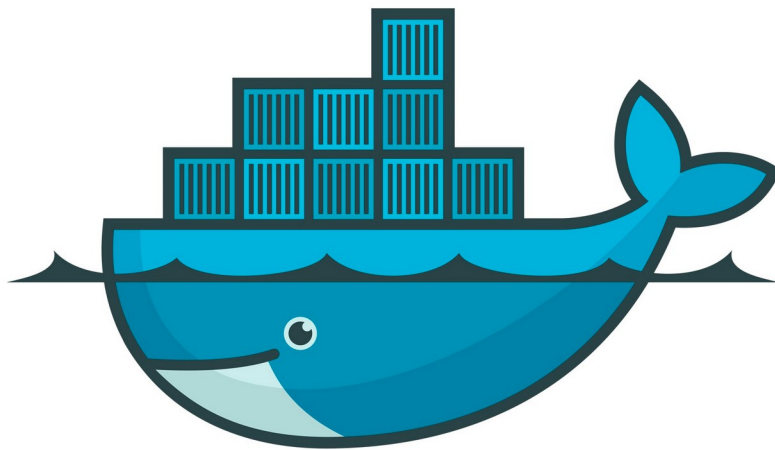





Introduction à Docker



Benjamin Loire et Jérémy Rousseau

Installation

Linux Debian Ubuntu Fedora Arch (non testé par Docker)	MacOS	Windows
		

Exemple d'installation – Ubuntu

```
sudo apt install docker.io docker-compose docker-buildx
```

VM vs Container vs Virtual Env

Virtual Machine

Container

Virtual Environment

Host Operating System

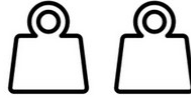
Host Hardware

VM vs Container vs Virtual Env

Virtual Machine



Container



Virtual Environment



Host Operating System

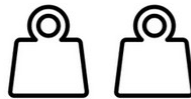
Host Hardware

VM vs Container vs Virtual Env

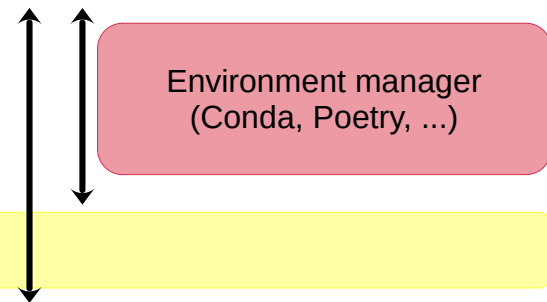
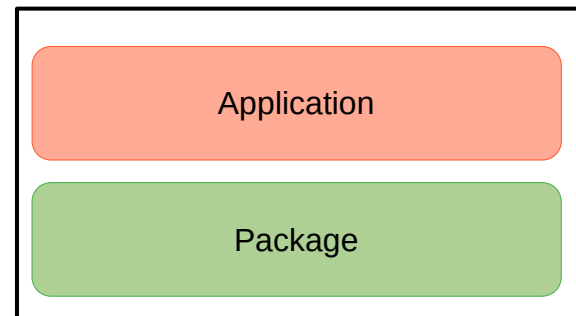
Virtual Machine



Container



Virtual Environment



Host Operating System

Host Hardware

VM vs Container vs Virtual Env

Virtual Machine

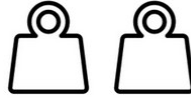


Application

Guest Operating System

Hypervisor
(VMWare, Virtualbox, ...)

Container



Virtual Environment



Application

Package

Environment manager
(Conda, Poetry, ...)

Host Operating System

Host Hardware

VM vs Container vs Virtual Env

Virtual Machine

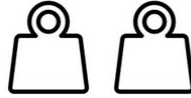


Application

Guest Operating System

Hypervisor
(VMWare, Virtualbox, ...)

Container



Application

Image-specific software

Container engine
(Docker, Singularity, ...)

Virtual Environment



Application

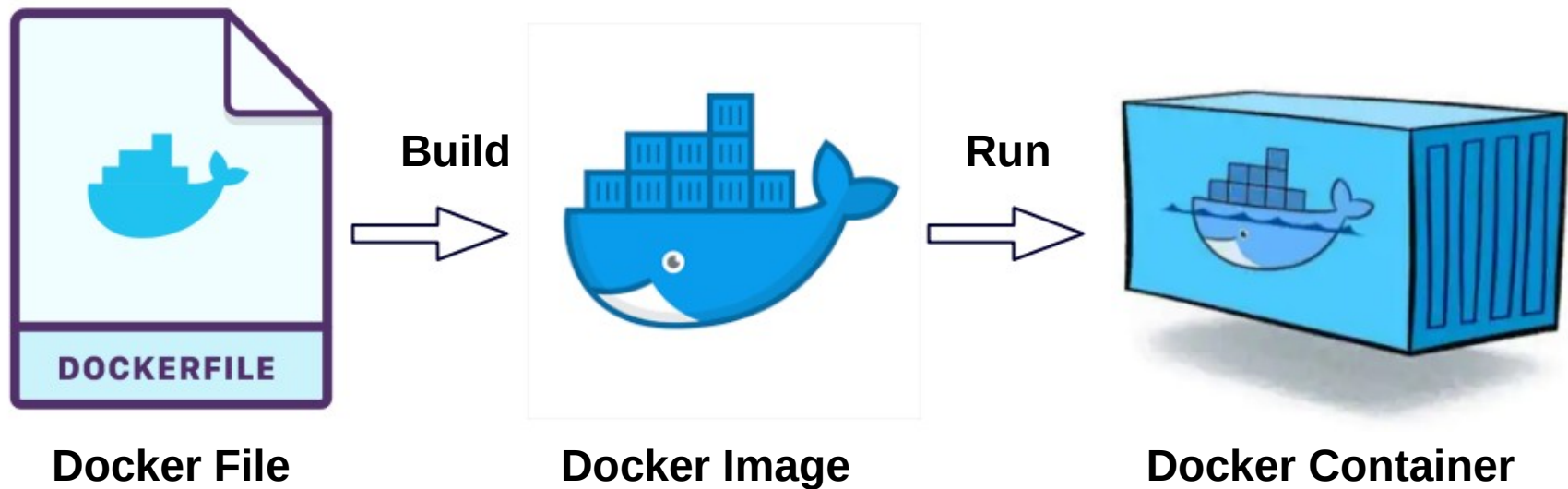
Package

Environment manager
(Conda, Poetry, ...)

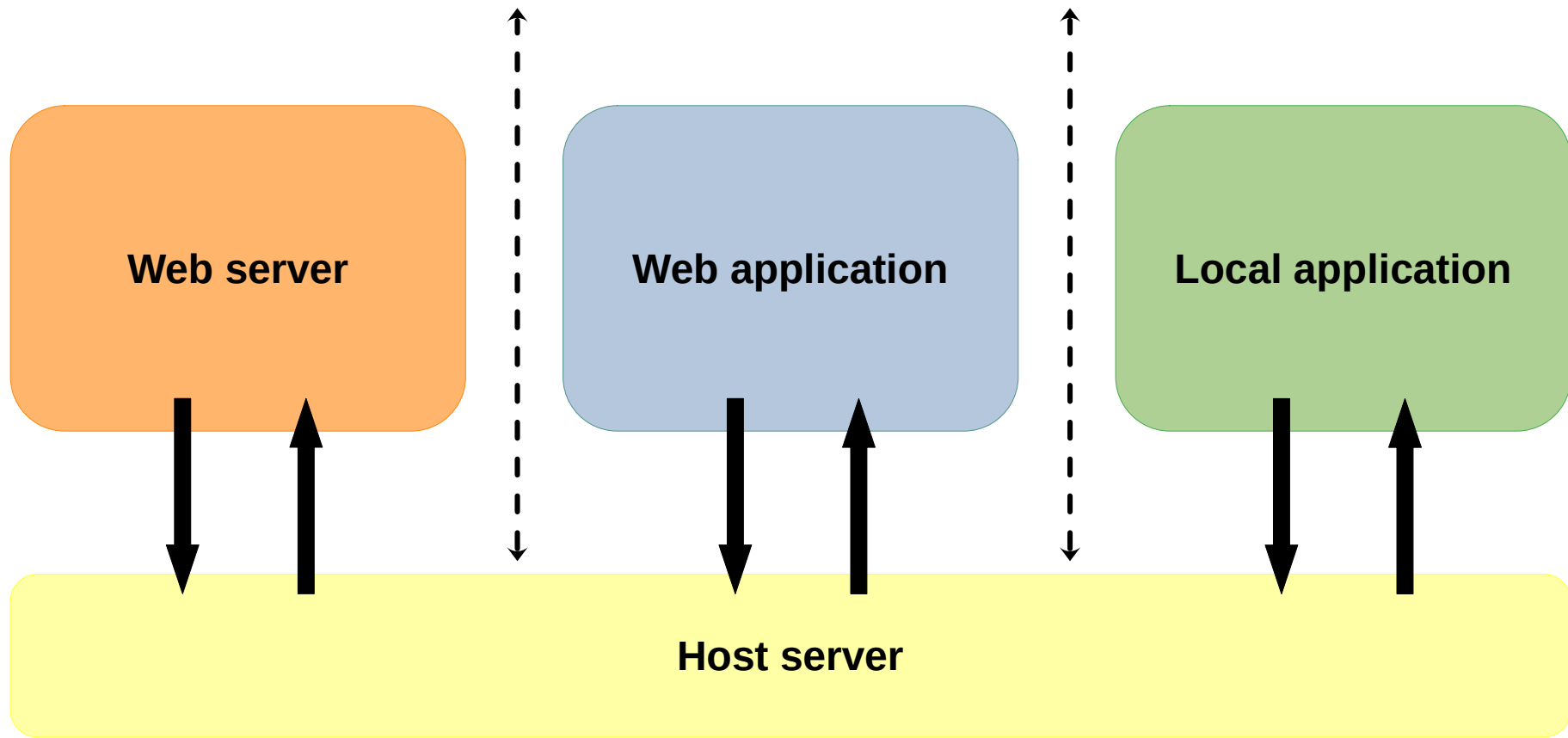
Host Operating System

Host Hardware

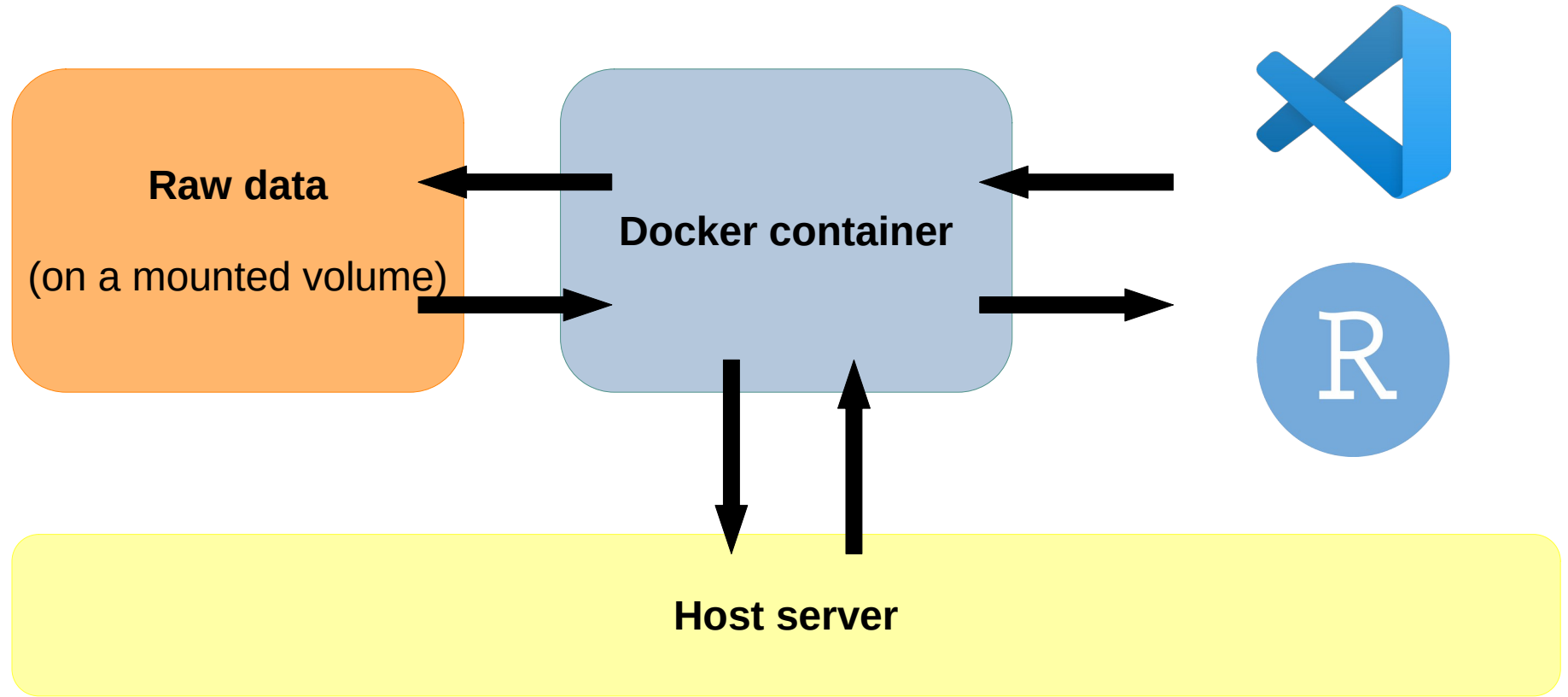
Docker, comment ça marche ?



Docker in the wild



Docker en analyse des données



Dockerfile

FROM

- Debian (base)
- Rocker (R)
- Python
- Continuumio (Anaconda2 & 3)
- ...

WORKDIR

- /home/projects

RUN

- pip install
- apt install
- git clone
- ...

CMD

- ["/bin/bash"]

Dockerfile

FROM

- Debian (base)
- Rocker (R)
- Python
- Continuumio (Anaconda2 & 3)
- ...

WORKDIR

- /home/projects

RUN

- pip install
- apt install
- git clone
- ...

CMD

- ["/bin/bash"]



Image de base du conteneur

Dockerfile

FROM

- Debian (base)
- Rocker (R)
- Python
- Continuumio (Anaconda2 & 3)
- ...

Image de base du conteneur

WORKDIR

- /home/projects

Répertoire de travail
Il faut éviter de travailler à la racine

RUN

- pip install
- apt install
- git clone
- ...

CMD

- ["/bin/bash"]

Dockerfile

FROM

- Debian (base)
- Rocker (R)
- Python
- Continuumio (Anaconda2 & 3)
- ...

Image de base du conteneur

WORKDIR

- /home/projects

Répertoire de travail
Il faut éviter de travailler à la racine

RUN

- pip install
- apt install
- git clone
- ...

Construction des différentes couches du conteneur (plusieurs "RUN" possibles)

CMD

- ["/bin/bash"]

Dockerfile

FROM

- Debian (base)
- Rocker (R)
- Python
- Continuumio (Anaconda2 & 3)
- ...

Image de base du conteneur

WORKDIR

- /home/projects

Répertoire de travail
Il faut éviter de travailler à la racine

RUN

- pip install
- apt install
- git clone
- ...

Construction des différentes couches du conteneur (plusieurs "RUN" possibles)

CMD

- ["/bin/bash"]

Première commande exécutée au lancement du conteneur

Dockerfile

FROM

- Debian (base)
- Rocker (R)
- Python
- Continuumio (Anaconda2 & 3)
- ...

WORKDIR

- /home/projects

RUN

- pip install
- apt install
- git clone
- ...

CMD

- ["/bin/bash"]

Autres commandes possibles

ENTRYPOINT	=> Spécifier l'exécutable par défaut
COPY	=> Copier des fichiers et des répertoires
MAINTAINER	=> Spécifier l'auteur d'une image
ENV	=> Définir les variables d'environnement

[Et encore plein d'autres commandes possibles]

Dockerfile - exemple

```
FROM ubuntu:20.04
```

```
WORKDIR /opt
```

```
RUN ln -snf /usr/share/zoneinfo/Europe/London /etc/localtime && \  
    echo Europe/London > /etc/timezone && \  
    apt-get update && apt-get install -y g++ automake cmake zlib1g-dev git libzstd-dev wget build-essential
```

```
RUN wget http://github.com/bbuchfink/diamond/archive/v2.1.8.tar.gz && \  
    tar xzf v2.1.8.tar.gz && \  
    cd diamond-2.1.8 && \  
    mkdir bin && \  
    cd bin && \  
    cmake .. && \  
    make -j4 && \  
    make install
```

```
WORKDIR /data
```

```
ENV PATH="$PATH:/diamond-2.1.8"
```

Dockerfile

Construire un conteneur

`docker build </path/to/dockerfile/folder> -t <container-name>:<tag>`

Exemple : `docker build Dockerfile -t diamond:2.1.8`

`docker buildx build -f </path/to/dockerfile/folder> -t <container-name>:<tag>`

Obtenir *docker buildx* (ubuntu) : *sudo apt install docker-buildx-plugin*

La construction d'un conteneur peut très longue

docker run et docker compose up

Conteneur disponible en local

```
docker run -v /path/to/your/data:/home/project <container-name>:<tag> <options>
```

Exemple 1 : docker run \
-v /path/to/fasta/folder:/data \
diamond:2.1.8 \
diamond <diamond-options>

Conteneur disponible en local

```
docker run -v /path/to/your/data:/home/project <container-name>:<tag> <options>
```

Exemple 1 : docker run \

```
-v /path/to/fasta/folder:/data \  
diamond:2.1.8 \  
diamond <diamond-options>
```

Exemple 2 : docker run \

```
-v /interproscan-5.65-97.0/data:/opt/interproscan/data \  
-v /path/to/fasta/folder:/data \  
interproscan:latest \  
interproscan.sh <interproscan-options>
```

Conteneur disponible en local

```
docker run -it -v /path/to/your/data:/home/project <container-name>:<tag>
```

Exemple 3 : docker run \
-it \
-v /path/to/fasta/folder:/data \
diamond:2.1.8

Docker run

Conteneur disponible en local

```
docker run -it -v /path/to/your/data:/home/project <container-name>:<tag>
```

Conteneur disponible sur Docker Hub

```
docker pull <user-name>/<container>:<tag>
```

```
docker run -v /path/to/your/data:/home/project <user-name>/<container>:<tag> <options>
```

```
docker run -it -v /path/to/your/data:/home/project <user-name>/<container>:<tag> <options>
```

Utilisation de docker compose

`docker compose up`

Docker compose

```
version: "2"
name: scdatapipeline_project_${USER}
services:
  pipeline:
    image: scdatapipeline:latest
    container_name: scdatapipeline_instance_${USER}
    user: ${USERID}:${PROJECT_GID}
    group_add:
      - 998 # Has to be the docker group ID
    environment:
      - SCDP_PATH_ROOT=/home
    volumes:
      - /mnt/DATA_4TB/projects/[PROJECT_NAME]:/home
      - /mnt/DATA_4TB/projects/singleCellAtlases:/mnt/DATA_4TB/projects/singleCellAtlases
      - /etc/passwd:/etc/passwd:ro
      - /etc/group:/etc/group:ro
      - /etc/shadow:/etc/shadow:ro
    working_dir: /home
    tty: true
```

name : Nom du projet Docker

service : possibilité d'avoir plusieurs conteneurs

docker-compose.yaml

Docker compose

```
version: "2"
name: scdatapipeline_project_${USER}
services:
  pipeline:
    image: scdatapipeline:latest
    container_name: scdatapipeline_instance_${USER}
    user: ${USERID}:${PROJECT_GID}
    group_add:
      - 998 # Has to be the docker group ID
    environment:
      - SCDP_PATH_ROOT=/home
    volumes:
      - /mnt/DATA_4TB/projects/[PROJECT_NAME]:/home
      - /mnt/DATA_4TB/projects/singleCellAtlases:/mnt/DATA_4TB/projects/singleCellAtlases
      - /etc/passwd:/etc/passwd:ro
      - /etc/group:/etc/group:ro
      - /etc/shadow:/etc/shadow:ro
    working_dir: /home
    tty: true
```

name : Nom du projet Docker

service : possibilité d'avoir plusieurs conteneurs

image : nom de l'image

docker-compose.yaml

Docker compose

```
version: "2"
name: scdatapipeline_project_${USER}
services:
  pipeline:
    image: scdatapipeline:latest
    container_name: scdatapipeline_instance_${USER}
    user: ${USERID}:${PROJECT_GID}
    group_add:
      - 998 # Has to be the docker group ID
    environment:
      - SCDP_PATH_ROOT=/home
    volumes:
      - /mnt/DATA_4TB/projects/[PROJECT_NAME]:/home
      - /mnt/DATA_4TB/projects/singleCellAtlases:/mnt/DATA_4TB/projects/singleCellAtlases
      - /etc/passwd:/etc/passwd:ro
      - /etc/group:/etc/group:ro
      - /etc/shadow:/etc/shadow:ro
    working_dir: /home
    tty: true
```

name : Nom du projet Docker

service : possibilité d'avoir plusieurs conteneurs

image : nom de l'image

container_name : donner un nom au conteneur

docker-compose.yaml

Docker compose

```
version: "2"
name: scdatapipeline_project_${USER}
services:
  pipeline:
    image: scdatapipeline:latest
    container_name: scdatapipeline_instance_${USER}
    user: ${USERID}:${PROJECT_GID}
    group_add:
      - 998 # Has to be the docker group ID
    environment:
      - SCDP_PATH_ROOT=/home
    volumes:
      - /mnt/DATA_4TB/projects/[PROJECT_NAME]:/home
      - /mnt/DATA_4TB/projects/singleCellAtlases:/mnt/DATA_4TB/projects/singleCellAtlases
      - /etc/passwd:/etc/passwd:ro
      - /etc/group:/etc/group:ro
      - /etc/shadow:/etc/shadow:ro
    working_dir: /home
    tty: true
```

name : Nom du projet Docker

service : possibilité d'avoir plusieurs conteneurs

image : nom de l'image

container_name : donner un nom au conteneur

environment : variable d'environnement uniquement
présente dans le conteneur

docker-compose.yaml

Docker compose

```
version: "2"
name: scdatapipeline_project_${USER}
services:
  pipeline:
    image: scdatapipeline:latest
    container_name: scdatapipeline_instance_${USER}
    user: ${USERID}:${PROJECT_GID}
    group_add:
      - 998 # Has to be the docker group ID
    environment:
      - SCDP_PATH_ROOT=/home
    volumes:
      - /mnt/DATA_4TB/projects/[PROJECT_NAME]:/home
      - /mnt/DATA_4TB/projects/singleCellAtlases:/mnt/DATA_4TB/projects/singleCellAtlases
      - /etc/passwd:/etc/passwd:ro
      - /etc/group:/etc/group:ro
      - /etc/shadow:/etc/shadow:ro
    working_dir: /home
    tty: true
```

name : Nom du projet Docker

service : possibilité d'avoir plusieurs conteneurs

image : nom de l'image

container_name : donner un nom au conteneur

environment : variable d'environnement uniquement
présente dans le conteneur

volumes : Plusieurs volumes possibles

docker-compose.yaml

Docker compose

```
version: "2"
name: scdatapipeline_project_${USER}
services:
  pipeline:
    image: scdatapipeline:latest
    container_name: scdatapipeline_instance_${USER}
    user: ${USERID}:${PROJECT_GID}
    group_add:
      - 998 # Has to be the docker group ID
    environment:
      - SCDP_PATH_ROOT=/home
    volumes:
      - /mnt/DATA_4TB/projects/[PROJECT_NAME]:/home
      - /mnt/DATA_4TB/projects/singleCellAtlases:/mnt/DATA_4TB/projects/singleCellAtlases
      - /etc/passwd:/etc/passwd:ro
      - /etc/group:/etc/group:ro
      - /etc/shadow:/etc/shadow:ro
    working_dir: /home
    tty: true
```

name : Nom du projet Docker

service : possibilité d'avoir plusieurs conteneurs

image : nom de l'image

container_name : donner un nom au conteneur

environment : variable d'environnement uniquement
présente dans le conteneur

volumes : Plusieurs volumes possibles

working_dir : répertoire de travail

docker-compose.yaml

Root, un problème ?

Root : sur les systèmes UNIX, c'est le nom conventionnel de l'utilisateur qui possède toutes les permissions sur le système. Root peut faire ce qu'il veut.

Root, un problème ?

Root : sur les systèmes UNIX, c'est le nom conventionnel de l'utilisateur qui possède toutes les permissions sur le système. Root peut faire ce qu'il veut.

Les conteneurs Docker possèdent les privilèges root

P'tit conseil :

Root, un problème ?

Root : sur les systèmes UNIX, c'est le nom conventionnel de l'utilisateur qui possède toutes les permissions sur le système. Root peut faire ce qu'il veut.

Les conteneurs Docker possèdent les privilèges root

P'tit conseil :

rm -rf * = très dangereux

(surtout à la racine du système)

Les “mauvaises” solutions - multi-utilisateurs

Dans le cas d'une utilisation multi-utilisateurs sur un cluster

Rootless Docker

Chaque utilisateur doit lancer son
daemon docker

La configuration est un enfer

Une image par utilisateur
(C'est vraiment très lourd)

Les “mauvaises” solutions - multi-utilisateurs

Dans le cas d'une utilisation multi-utilisateurs sur un cluster

Rootless Docker

Chaque utilisateur doit lancer son daemon docker

La configuration est un enfer

Créer un compte utilisateur dans l'image

Le root n'est plus disponible dans le conteneur

Demande de connaître l'ID au moment de construire l'image

Une image par utilisateur
(C'est vraiment très lourd)

La “presque” bonne solution - multi-utilisateurs

```
export USERID=$(id -u)
export PROJECT_GID=$(id -g)
```

```
user: ${USERID}:${PROJECT_GID}
group_add:
  - 998 # Has to be the docker group ID
...
volumes:
  ...
  - /etc/passwd:/etc/passwd:ro
  - /etc/group:/etc/group:ro
  - /etc/shadow:/etc/shadow:ro
  ...
```

docker-compose.yaml

```
RUN chgrp -R 998 /usr /var && chmod -R g=u /usr /var
```

Dockerfile

La “presque” bonne solution - multi-utilisateurs

```
export USERID=$(id -u)  
export PROJECT_GID=$(id -g)
```

Ligne de commande, récupérer le « user id » et le « group id »

```
user: ${USERID}:${PROJECT_GID}  
group_add:  
  - 998 # Has to be the docker group ID  
...  
volumes:  
  ...  
  - /etc/passwd:/etc/passwd:ro  
  - /etc/group:/etc/group:ro  
  - /etc/shadow:/etc/shadow:ro  
  ...
```

docker-compose.yaml

```
RUN chgrp -R 998 /usr /var && chmod -R g=u /usr /var
```

Dockerfile

La “presque” bonne solution - multi-utilisateurs

```
export USERID=$(id -u)  
export PROJECT_GID=$(id -g)
```

Ligne de commande, récupérer le « user id » et le « group id »

Permet d'identifier l'utilisateur et le groupe dans le conteneur

```
user: ${USERID}:${PROJECT_GID}  
group_add:  
  - 998 # Has to be the docker group ID  
...  
volumes:  
  ...  
  - /etc/passwd:/etc/passwd:ro  
  - /etc/group:/etc/group:ro  
  - /etc/shadow:/etc/shadow:ro  
  ...
```

Nécessaire pour que Docker les prennent en compte

docker-compose.yaml

```
RUN chgrp -R 998 /usr /var && chmod -R g=u /usr /var
```

Dockerfile

La “presque” bonne solution - multi-utilisateurs

```
export USERID=$(id -u)  
export PROJECT_GID=$(id -g)
```

Ligne de commande, récupérer le « user id » et le « group id »

Permet d'identifier l'utilisateur et le groupe dans le conteneur

Group id de Docker sur la machine

Nécessaire pour que Docker les prennent en compte


```
user: ${USERID}:${PROJECT_GID}  
group_add:  
  - 998 # Has to be the docker group ID  
...  
volumes:  
  ...  
  - /etc/passwd:/etc/passwd:ro  
  - /etc/group:/etc/group:ro  
  - /etc/shadow:/etc/shadow:ro  
  ...
```

docker-compose.yaml



```
RUN chgrp -R 998 /usr /var && chmod -R g=u /usr /var
```

Dockerfile


Visual Studio Code et Docker




Docker v1.29.0

Microsoft  microsoft.com |  31,358,390 | ★★★★★ (87)



Makes it easy to create, manage, and debug containerized applications.

[Install](#) 



★ This extension is recommended because you have Docker installed.



Dev Containers v0.338.1

Microsoft  microsoft.com |  22,703,236 | ★★★★★ (47)

Open any folder or repository inside a Docker container and take advantage of Visual Studio Code's full feature set.

[Install](#)  

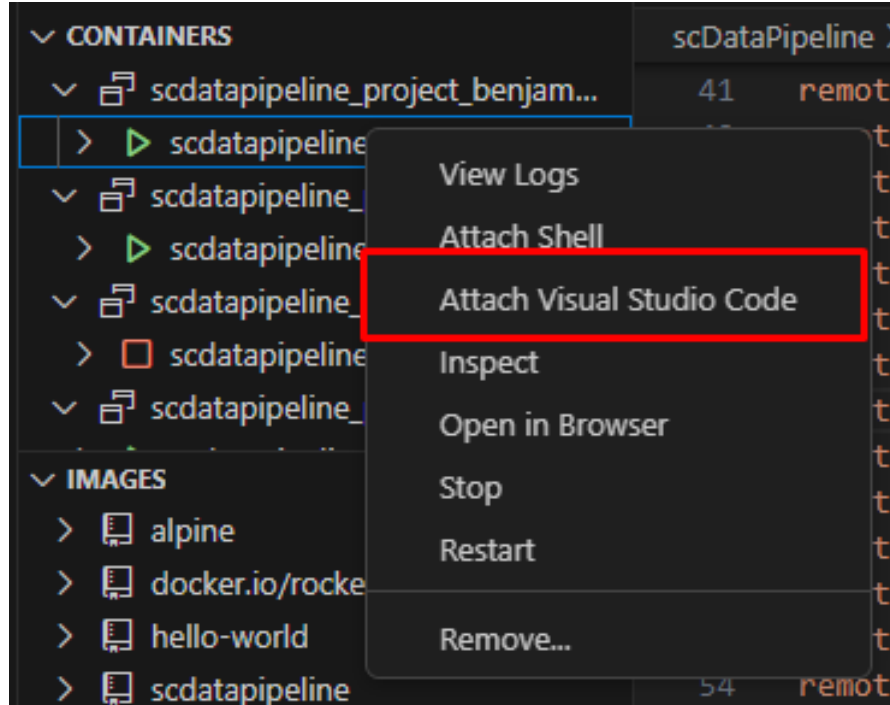
★ This extension is recommended because you have Docker installed.

Visual Studio Code et Docker

`tty: true`



docker-compose.yaml



Docker Hub

Docker Hub Container Im x

+

https://hub.docker.com

?

☆

Sign In

Sign up

dockerhub

Develop faster. Run anywhere.

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

Search Docker Hub


ctrl+K

Spotlight

CLOUD DEVELOPMENT

Build up to 39x faster with Docker Build Cloud


Introducing Docker Build Cloud: A new solution to speed up build times and improve developer productivity



AI/ML DEVELOPMENT

LLM Everywhere: Docker and Hugging Face


Set up a local development environment for Hugging Face with Docker




SOFTWARE SUPPLY CHAIN

Take action on prioritized insights

Bridge the gap between development workflows and security needs




Featured Solutions

 **Kubescape**


Secure your Kubernetes cluster and gain insight into your cluster's security...

± 10K+

 **Ambassador Telepresence**


Instantly bridge your workstation with Kubernetes clusters in the cloud. Test...

± 50K+

 grafana/grafana

The official Grafana docker container


☆ 2.9K ± 1B+


 **opensearchproject/open...**


The Official Docker Image of OpenSearch (<https://opensearch.org/>)


☆ 99 ± 50M+


Trending this week

 **homeassistant/amd64-a...**




 **paketobuildpacks/build**



 **vitess/lite**

A slimmed down version of Vitess

 **friendica**

Welcome to the free social web.

24

Docker Hub – un peu de prévention

Analysis on Docker Hub malicious images: Attacks through public container images

BY STEFANO CHIERICI - NOVEMBER 23, 2022

SHARE: [f](#) [in](#) [X](#)

CONTENT:

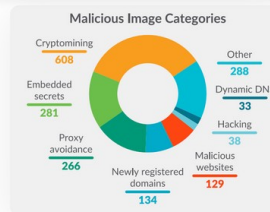
DOCKER HUB

TYPOSQUATTING, CRYPTOMINERS, AND KEYS

FINAL WORDS

HIDE -

Supply Chain attacks are not new, but this past year they received much more attention due to high profile vulnerabilities in popular dependencies. Generally, the focus has been on the dependency attack vector. This is when source code of a dependency or product is modified by a malicious actor in order to compromise anyone who uses it in their own software.



Docker Hub repositories hide over 1,650 malicious containers

By [Bill Toulas](#)

November 24, 2022

12:16 PM

0



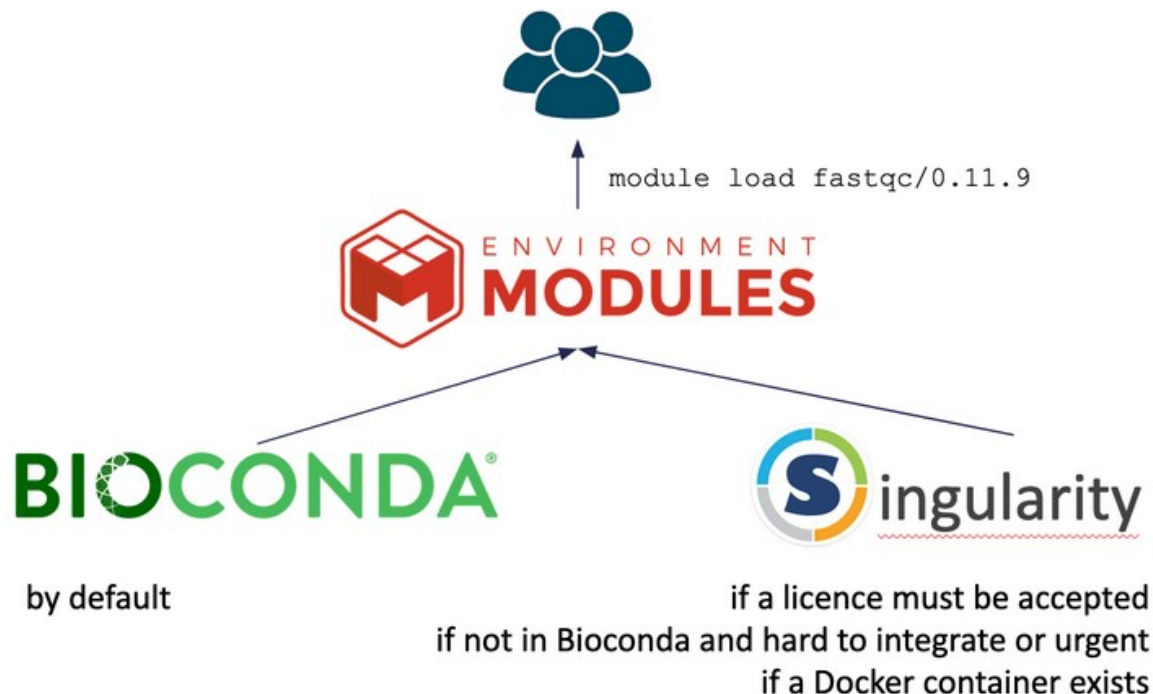
Over 1,600 publicly available Docker Hub images hide malicious behavior, including cryptocurrency miners, embedded secrets that can be used as backdoors, DNS hijackers, and website redirectors.

Docker Hub is a cloud-based container library allowing people to freely search and download Docker images or upload their creations to the public library or personal repositories.

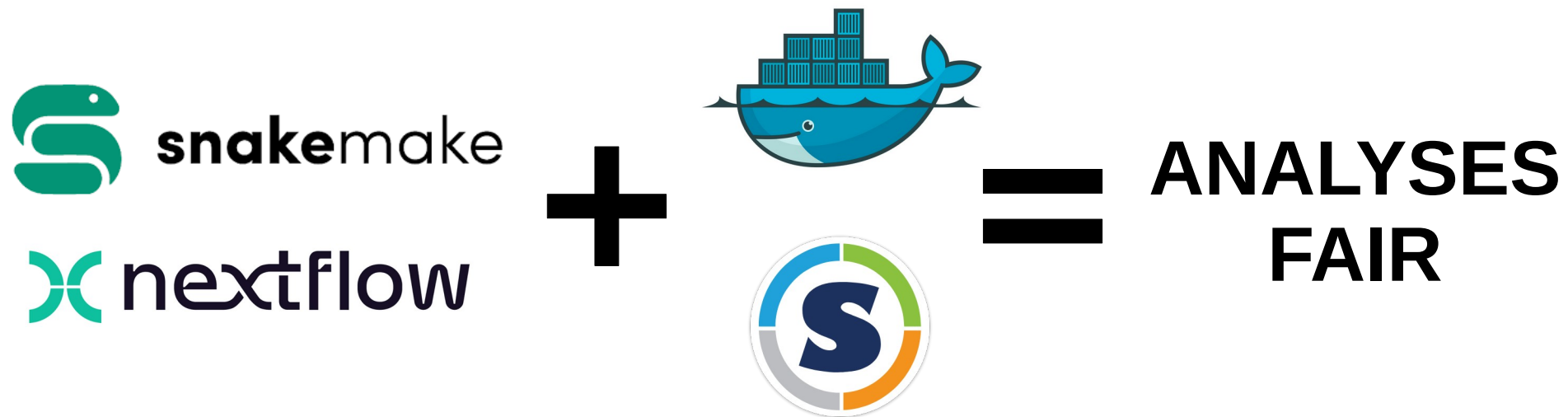
Quelques commandes utiles

<code>docker build <dockerfile></code>	Construire un conteneur à partir d'un Dockerfile
<code>docker buildx <dockerfile></code>	Construire un conteneur à partir d'un Dockerfile
<code>docker tag <old-tag> <new-tag></code>	Permet de modifier le tag d'un conteneur
<code>docker run <option> <container></code>	Permet d'exécuter un conteneur (l'option <code>-it</code> permet d'avoir un shell interactif)
<code>docker compose up</code>	Permet d'exécuter un conteneur – nécessite un fichier yaml
<code>docker ps</code>	Si rien n'est précisé : afficher uniquement les conteneurs actifs <code>-a</code> : permet de voir également les conteneurs inactif
<code>docker start <container-id></code>	Permet de démarrer un conteneur
<code>docker stop <container-id></code>	Permet d'éteindre un conteneur
<code>docker rm <container-id></code>	Permet de supprimer un conteneur (l'option <code>-f</code> permet de forcer la suppression)
<code>docker pull <user-name>/<container></code>	Permet de récupérer un conteneur sur Docker Hub
<code>docker push <user-name>/<container></code>	Permet d'envoyer un conteneur vers Docker Hub

Cas pratique de l'utilisation des conteneurs



Et après ?



FAIR : Findable, Accessible, Interoperable, Reusable

Un peu de lecture

- **Docker** <https://www.docker.com/>
- **Docker documentation** <https://docs.docker.com/>
- **Docker Hub** <https://hub.docker.com/>
- **Articles consternant les images docker malveillantes**
 - <https://sysdig.com/blog/analysis-of-supply-chain-attacks-through-public-docker-images/>
 - <https://www.bleepingcomputer.com/news/security/docker-hub-repositories-hide-over-1-650-malicious-containers/>
 - <https://www.it-connect.fr/conteneurs-docker-plus-de-1-650-images-malveillantes-identifiees-sur-le-docker-hub/>