

Environment manager from your OS to your environment

Encapsulation levels using docker, conda¹



P. Marin, M. Hiriart, P. Ruiz & N. Goué
aubi@uca.fr

Université Clermont Auvergne, AuBi, Mésocentre

15 novembre 2022

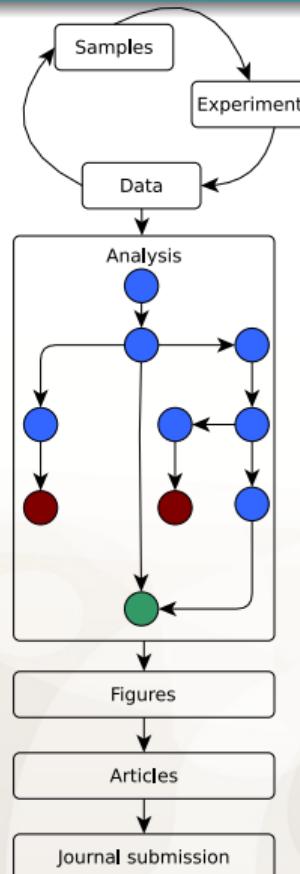


1. This work is derived from the IFB and I2BC team members

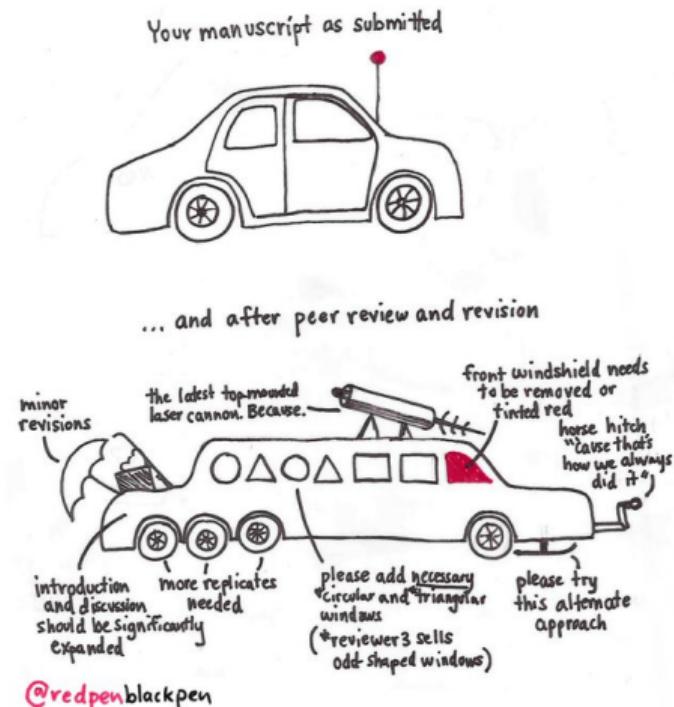
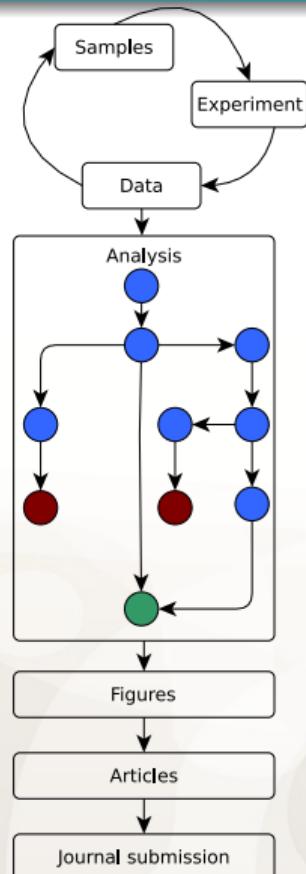
Sommaire

- 1 A common use-case
 - Retry my results
 - The use of packaging
 - Example with R
- 2 Manage your local environment
 - How conda works
- 3 Manage your hardware configuration
 - How virtual manager works
- 4 Manage your OS configuration
 - How container works
- 5 Conda ecosystem
 - a case of bioconda
- 6 A second level, the containers
 - Docker
 - Singularity

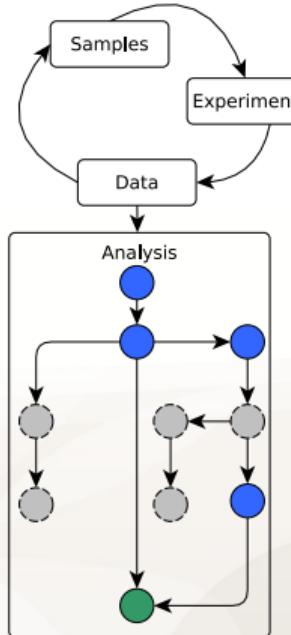
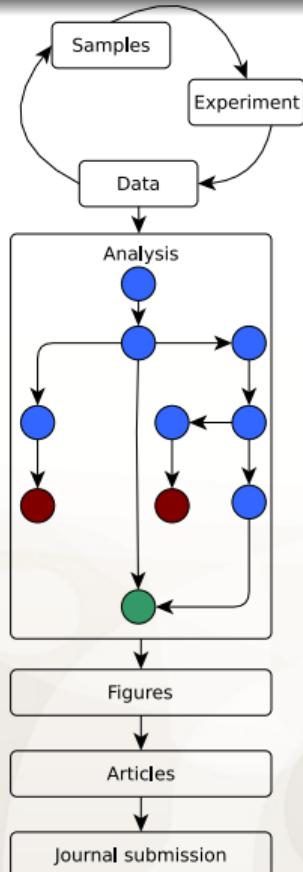
A classic use case



A classic use case



A classic use case



A classic use case

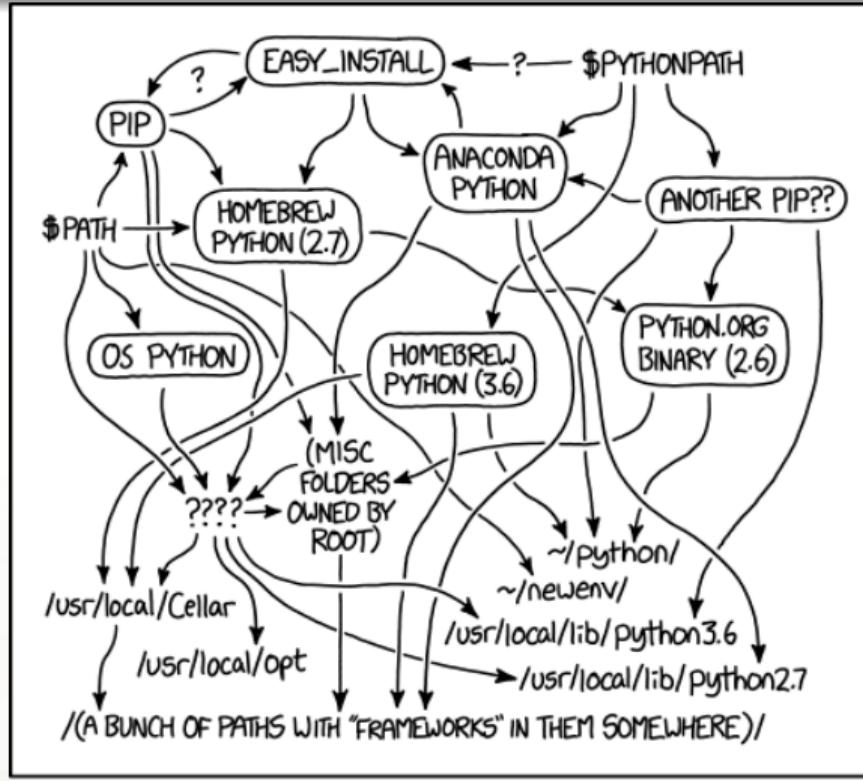
What are the changes ?

- Tool version
- Packages
- Environment variables
- OS version
- The computer
- ...

■ Tool compatibility troubles

- Python version ? 2.7, 3.8...
- Which tool version ?
- Installation without root access
- coexistence between several versions, libraries

My python env



MY PYTHON ENVIRONMENT HAS BECOME SO DEGRADED
THAT MY LAPTOP HAS BEEN DECLARED A SUPERFUND SITE.

Encapsulation levels

Encapsulation : capture the environment of applications (OS, packages, libraries) to control their execution



- Hardware virtualisation (virtual machines)

Encapsulation levels

Encapsulation : capture the environment of applications (OS, packages, libraries) to control their execution



- Hardware virtualisation (virtual machines)



- OS virtualisation (images and containers)

Encapsulation levels

Encapsulation : capture the environment of applications (OS, packages, libraries) to control their execution



- Hardware virtualisation (virtual machines)



- OS virtualisation (images and containers)
- Environment management (package manager) **CONDA**

Example of R and package installation

Classical installation

- Start with a computer and a specific OS

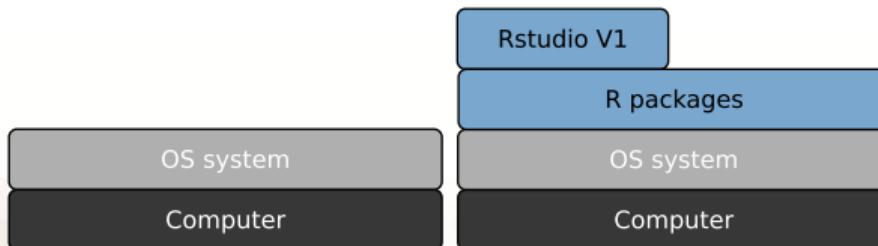
OS system

Computer

Example of R and package installation

Classical installation

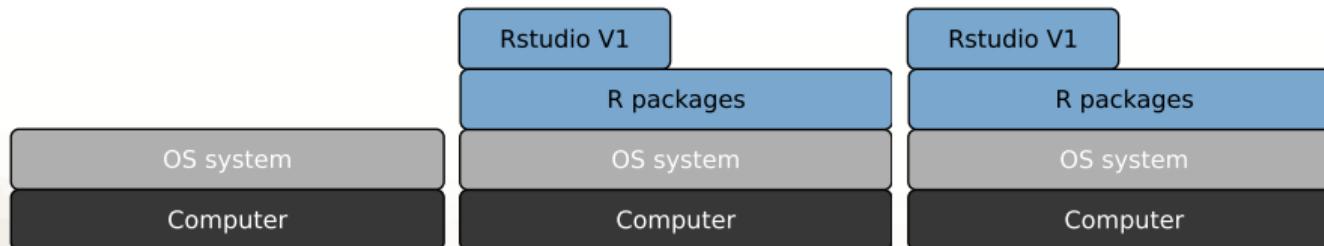
- Start with a computer and a specific OS
- Inside, we installed a new  application



Example of R and package installation

Classical installation

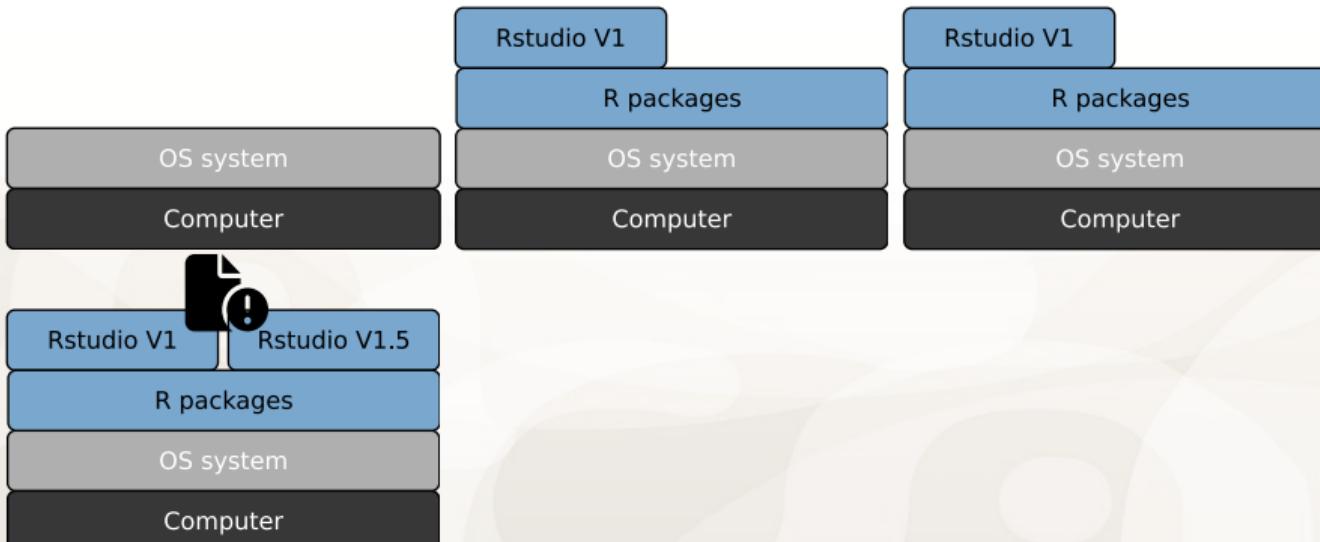
- Start with a computer and a specific OS
- Inside, we installed a new  application
-  need some dependencies



Example of R and package installation

Classical installation

- Start with a computer and a specific OS
- Inside, we installed a new  application
-  need some dependencies
- we tested the last  version -> might be conflicts



Example of R and package installation

Conda use

- The idea is to separate each application in here own environment **CONDA**

Conda env

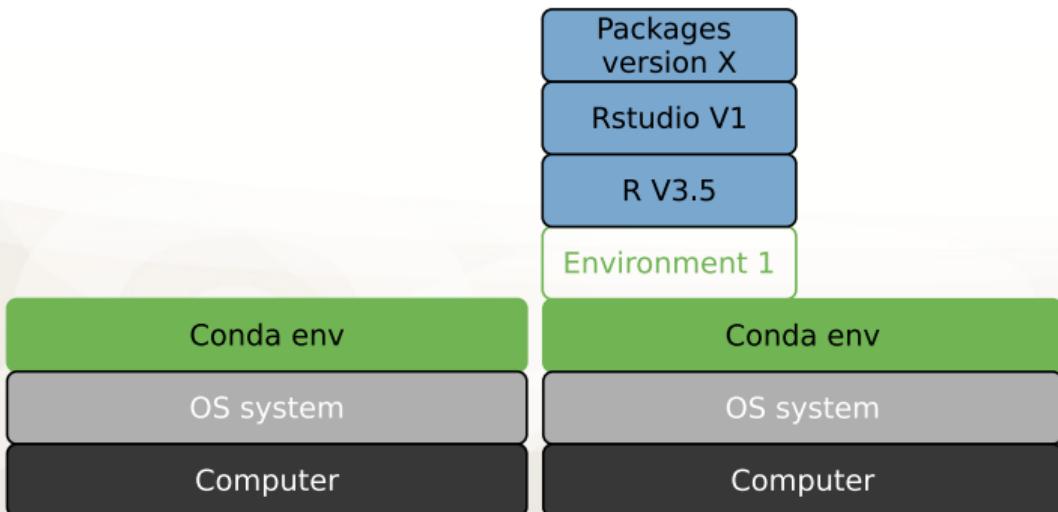
OS system

Computer

Example of R and package installation

Conda use

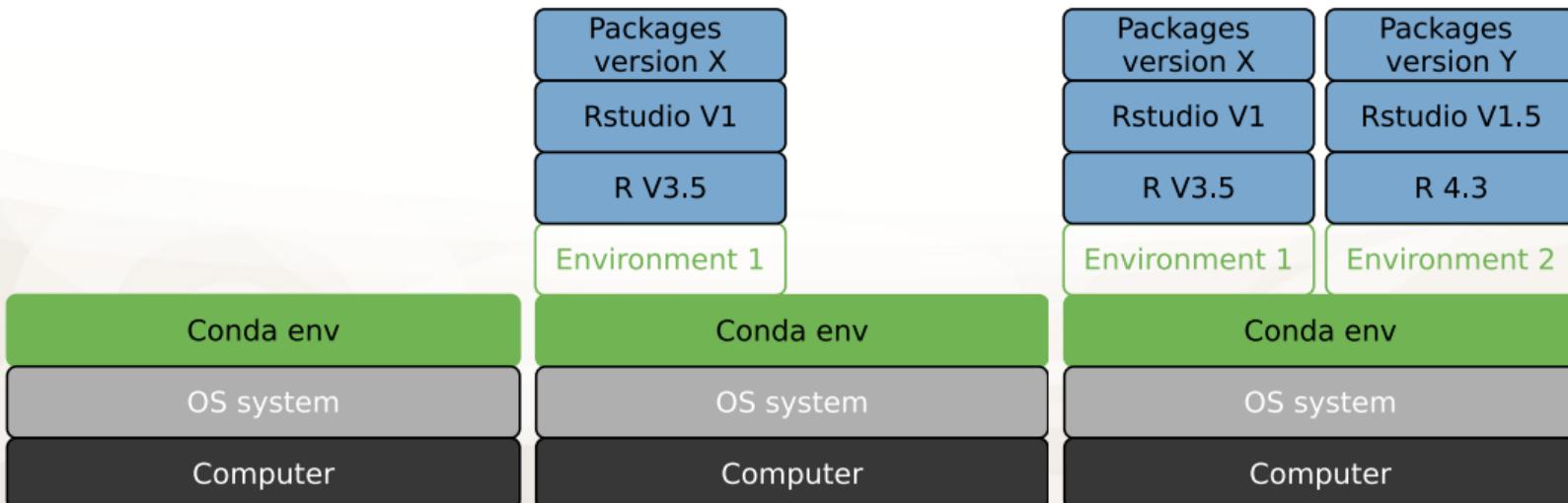
- The idea is to separate each application in here own environment **CONDA**
- A tool version, a conda environment



Example of R and package installation

Conda use

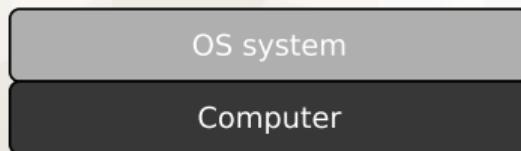
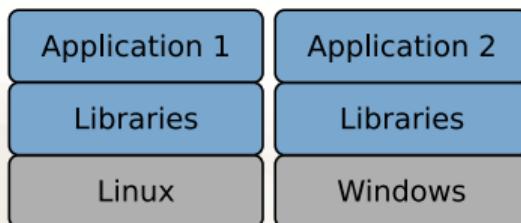
- The idea is to separate each application in here own environment **CONDA**
- A tool version, a conda environment
- Create a new environment for my new tool version, my analysis...



Example of R and package installation

hardware virtualisation

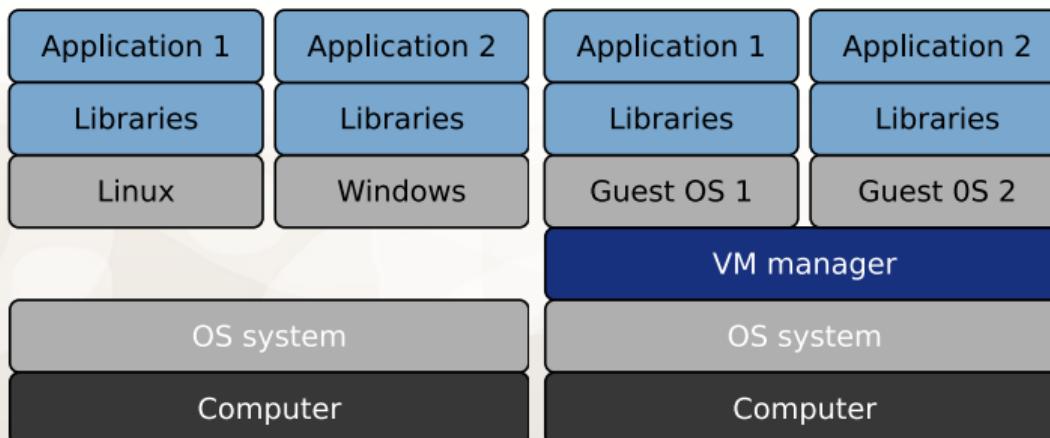
- If we want a software from a different OS ?



Example of R and package installation

hardware virtualisation

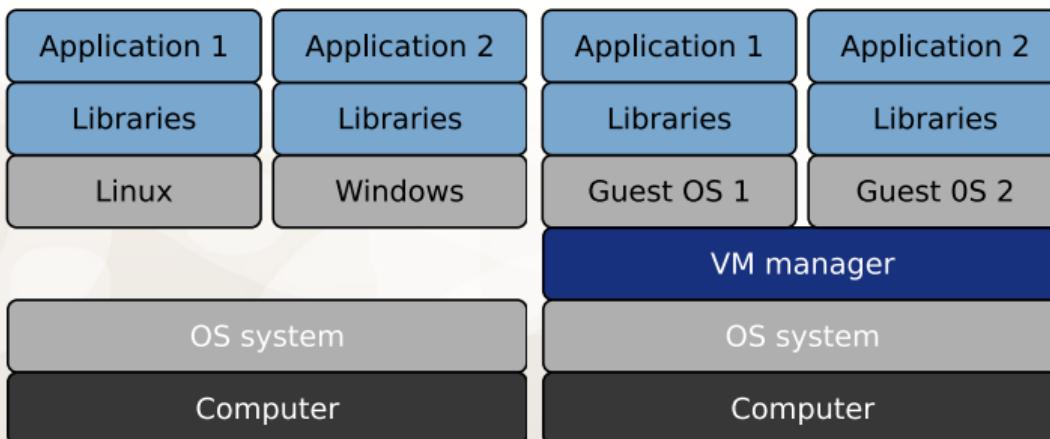
- If we want a software from a different OS ?
- Use virtual machines



Example of R and package installation

hardware virtualisation

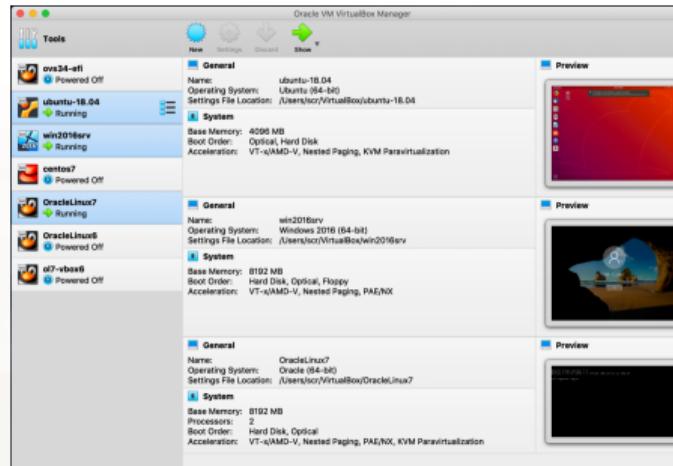
- If we want a software from a different OS ?
- Use virtual machines
- Each application get a total different and independant environment



Example of R and package installation

hardware virtualisation

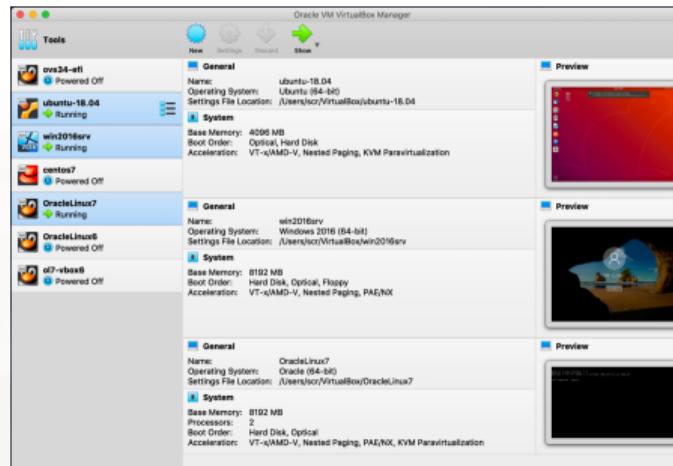
- If we want a software from a different OS ?
- Use virtual machines
- Each application get a total different and independant environment
- Virtual machine could be transfered to another computer



Example of R and package installation

hardware virtualisation

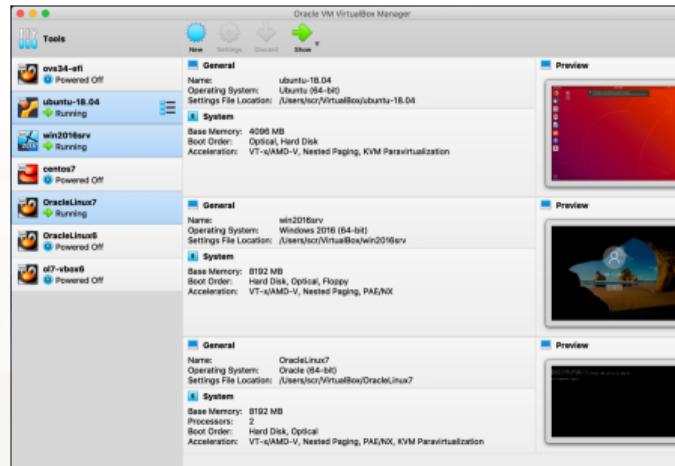
- If we want a software from a different OS ?
- Use virtual machines
- Each application get a total different and independant environment
- Virtual machine could be transfered to another computer
- Redundancy between VMs



Example of R and package installation

hardware virtualisation

- If we want a software from a different OS ?
- Use virtual machines
- Each application get a total different and independant environment
- Virtual machine could be transferred to another computer
- Redundancy between VMs
- Heavy to set up

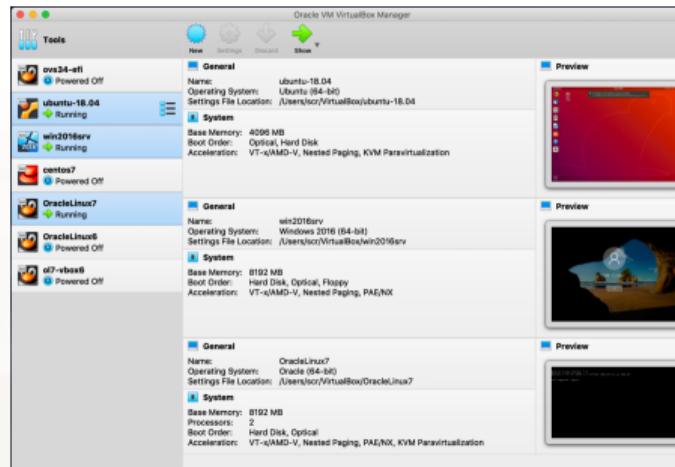


Example of R and package installation

hardware virtualisation

- If we want a software from a different OS ?
- Use virtual machines
- Each application get a total different and independant environment
- Virtual machine could be transferred to another computer

- Redundancy between VMs
- Heavy to set up
- No automation



Example of R and package installation

OS virtualisation

- "Trick" applications into believing that they are in a different OS than the host's

Example of R and package installation

OS virtualisation

- "Trick" applications into believing that they are in a different OS than the host's

- Named containers :  or 

Example of R and package installation

OS virtualisation

- "Trick" applications into believing that they are in a different OS than the host's



- Named containers : or
- Avoid redundancy

OS system

Computer

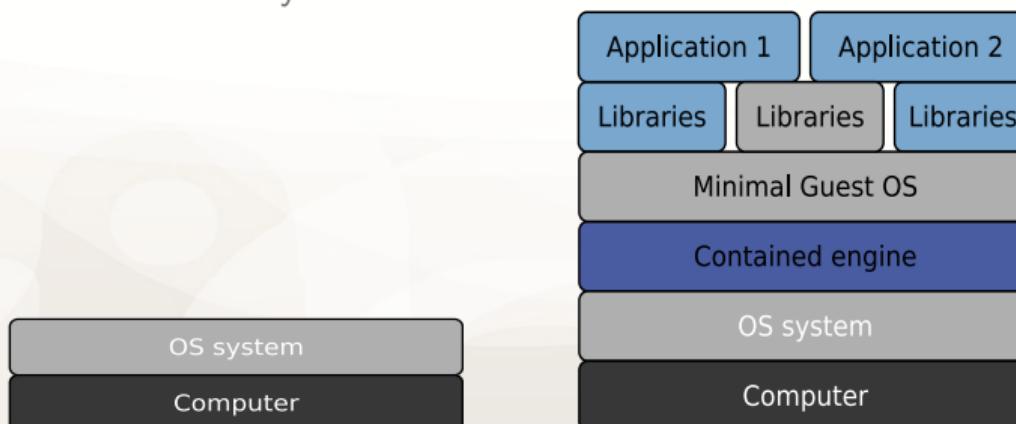
2. <https://www.docker.com/blog/scaling-docker-to-serve-millions-more-developers-network-express/>

Example of R and package installation

OS virtualisation

- "Trick" applications into believing that they are in a different OS than the host's

- Named containers :  or 
- Avoid redundancy



2. <https://www.docker.com/blog/scaling-docker-to-serve-millions-more-developers-network-express/>

Example of R and package installation

OS virtualisation

- "Trick" applications into believing that they are in a different OS than the host's



- Named containers :
- Avoid redundancy
 - Speed
 - Faster installation
 - No boot time

Example of R and package installation

OS virtualisation

- "Trick" applications into believing that they are in a different OS than the host's



- Named containers : or
- Avoid redundancy
- Speed
 - Faster installation
 - No boot time
- Lightweight
 - Minimal base OS
 - Minimal set of library and global environment
 - Easy sharing of application

Docker Hub search results for "rstudio":

- ibm/rstudio-ppc64le** (VERIFIED PUBLISHER) - Integrated development environment (IDE) for R. Linux. ppc64le. By IBM • Updated 3 years ago.
- whaleal/rstudio-base** (SPONSORED OSS) - By whaleal • Updated 4 years ago. Linux. x86_64.
- biocconductr/rstudio_yescal** (SPONSORED OSS) - By biocconductr • Updated 16 days ago. Linux. x86_64.
- truecharts/rstudio** (SPONSORED OSS) - By truecharts • Updated 11 days ago. Linux. x86_64.

Example of R and package installation

OS virtualisation

- "Trick" applications into believing that they are in a different OS than the host's



- Named containers :

- Avoid redundancy
- Speed

- Faster installation
- No boot time

- Lightweight
 - Minimal base OS
 - Minimal set of library and global environment
 - Easy sharing of application

Singularity Community Catalog

Stack	Description	Singularity* recipe	Topics	QC	Status
gist/rstudio-server-conda	Run RStudio Server in a conda environment	gist/rstudio-server-conda.singularity	rstudio, singularity, conda	106	
rickjerr/singularity-rstudio	RStudio Server In a Singularity container	rickjerr/singularity-rstudio.singularity	rstudio-server, singularity-image	40	
singularityhub/singularity-compose-examples	A simple example of running a MongoDB instance to query a database	singularityhub/singularity-compose-examples.singularity	singularity, compose, mongodb	12	
OSC/bc_osc_rstudio_server	Batch Connect - OSC RStudio Server	OSC/bc_osc_rstudio_server.singularity	batch, osc, rstudio	6	
RBioData/singularity	Singularity configurations for R and pbzR packages.	RBioData/singularity.singularity	bioconductor, r, singularity	3	

Example of R and package installation

Helping developers manage inactive Images

To help Docker economically scale its infrastructure to support free services for our growing base of users, several updates were announced. First, a new inactive image retention policy was introduced that will automatically delete images hosted in free accounts that have not been used in 6 months. In addition, Docker will also be providing tooling, in the form of a UI and APIs, that will allow users to more easily manage their images. Together, these changes will allow developers to more easily clean up their inactive images and also ensure Docker can economically scale its infrastructure.

With this new policy, starting on **November 1**, images stored in free Docker Hub repositories that have not had their manifest pushed or pulled in the last 6 months will be removed. This policy does not apply to images stored by paid [Docker Hub subscription accounts](#), [Docker Verified Publishers](#), or [Docker Official Images](#).

- **Example #1:** Molly, a free Docker Hub user, pushed a tagged image `molly/hello-world:v1` to Docker Hub on January 1, 2019. The image was never pulled since it was pushed. This tagged image will be considered inactive beginning November 1, 2020 when the new policy takes effect. The image and any tag pointing to it, will be subject to deletion on November 1, 2020.
- **Example #2:** Molly has another untagged image `molly/myapp@sha256:c0ffee` that was first pushed on January 1, 2018. This image was last pulled on August 1, 2020. This image will be considered an active image and will not be subject to deletion on November 1, 2020.

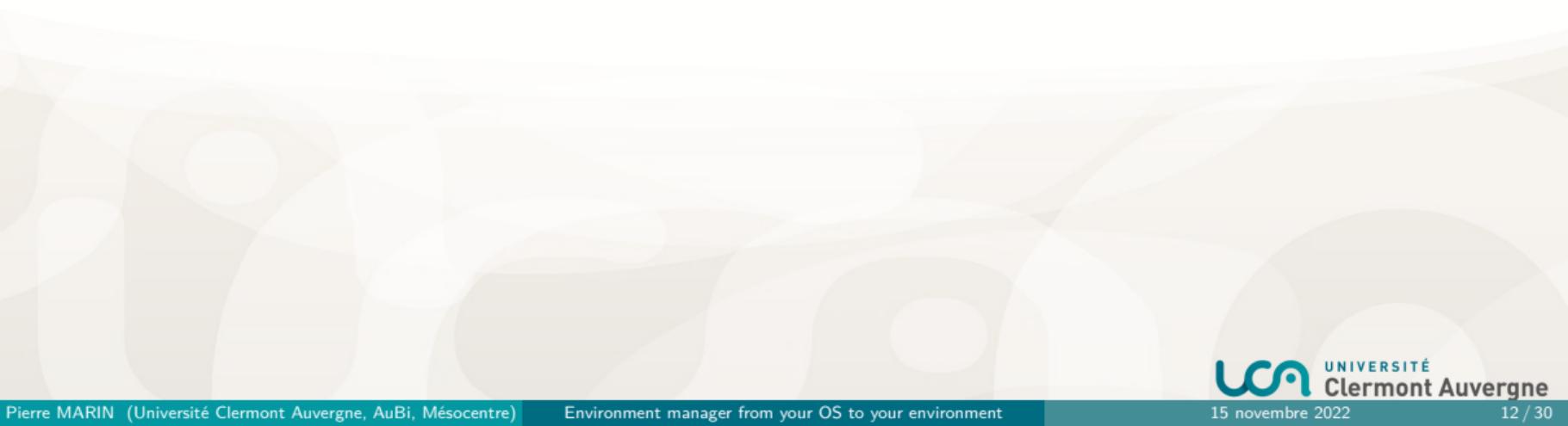
Minimizing impact to the developer community

For free accounts, Docker offers free retention of images inactive for six months. For users that need their inactive images to be retained, Docker also offers unlimited image retention as a feature of the Pro and Team plans. Visit [docker.wpsitekeep.com/pricing](#) to view the available plans.

In addition, Docker will be offering a set of tools and services to help developers easily view and manage their images, including the following product updates which will be made available on Docker Hub in the coming months:

- Image management dashboard to view and manage images across all repositories within a namespace (roadmap issue [#146](#))
- Email notifications for images that are set to expire (roadmap issue [#147](#))
- Restoration of recently deleted images (roadmap issue [#148](#))

Conda system



Conda system

■ Anaconda

- Open source distribution
- Cross platform
- Available on cluster without admin whrite
- Thousands of available tool in informatic and bioinformatic

Conda system

■ Anaconda

- Open source distribution
- Cross platform
- Available on cluster without admin whrite
- Thousands of available tool in informatic and bioinformatic

■ Miniconda

- A lightheight Anaconda version with minimal requirment
- Same advantages ad Anaconda

Conda system

■ Anaconda

- Open source distribution
- Cross platform
- Available on cluster without admin whrite
- Thousands of available tool in informatic and bioinformatic

■ Miniconda

- A lightheight Anaconda version with minimal requirment
- Same advantages ad Anaconda

■ Conda **CONDA**

- Package manager AND environment manager
- installed with Ana or Miniconda
- Python based but can also install tools from R, C++ or Julia...

Conda system

CONDA CHEATSHEET	
QUICK START	
Tip: It is recommended to create a new environment for any new project or workflow.	
verify conda install and check version	conda info
update conda in base environment	conda update -n base conda
install latest anaconda distribution (see release notes)	conda install anaconda=2022.08
create a new environment (tip: name environment descriptively)	conda create --name ENVNAME
activate environment (do this before installing packages)	conda activate ENVNAME
CHANNELS AND PACKAGES	
Tip: Package dependencies and platform specifics are automatically resolved when using conda.	
install packages from specified channel	conda install -c CHANNELNAME PKG1 PKG2
list installed packages	conda list
uninstall package	conda uninstall PKGNAME
update all packages	conda update --all
install specific version of package	conda install PKGNAME=3.1.4
install a package from specific channel	conda install CHANNELNAME::PKGNAME
install package with AND logic	conda install "PKGNAME>2.5,<3.2"
install package with OR logic	conda install "PKGNAME [version='2.5 3.2']"
list installed packages with source info	conda list --show-channel-urls
view channel sources	conda config --show-sources
add channel	conda config --add channels CHANNELNAME
set default channel for pkg fetching (targets first channel in channel sources)	conda config --set channel_priority strict
WORKING WITH CONDA ENVIRONMENTS	
Tip: List environments at the beginning of your session. Environments with an asterisk are active.	

The channels and the tools

The tools are packaged and available on several **channels**

-
3. Bioconda : sustainable and comprehensive software distribution for the life sciences *Grüning et al.*, Nature methods, 2018. DOI 10.1038/s41592-018-0046-7

The channels and the tools

The tools are packaged and available on several **channels**

- Conda-forge
- Anaconda
- R
- Bioconda -> Most of the bioinformatic tools

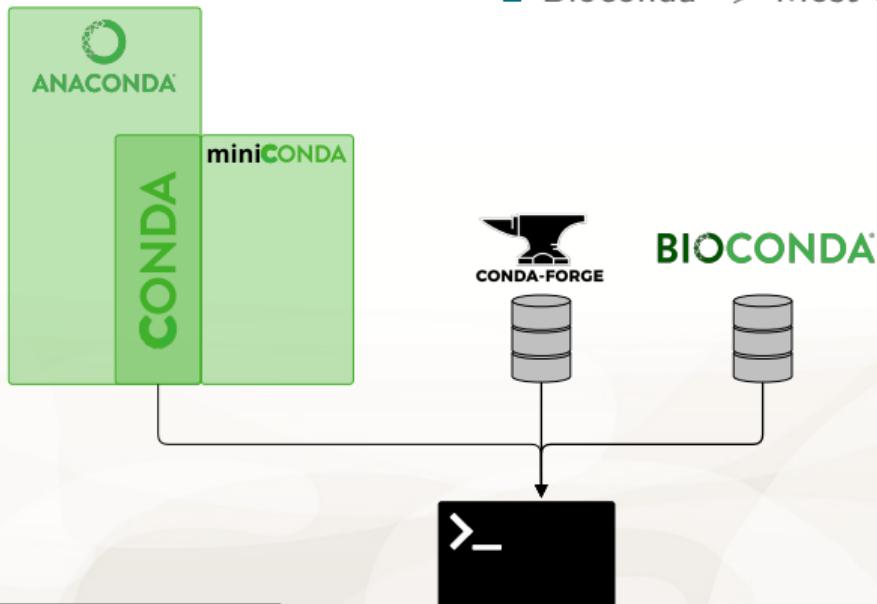
3. Bioconda : sustainable and comprehensive software distribution for the life sciences Grüning et al., Nature methods, 2018. DOI 10.1038/s41592-018-0046-7



The channels and the tools

The tools are packaged and available on several **channels**

- Conda-forge
- Anaconda
- R
- Bioconda -> Most of the bioinformatic tools



3

3. Bioconda : sustainable and comprehensive software distribution for the life sciences Grüning et al., Nature methods, 2018. DOI 10.1038/s41592-018-0046-7

Basic commands

Create environment

```
$ conda create -n [env_name]  
$ conda activate [env_name]  
$ conda deactivate [env_name]
```

Basic commands

Create environment

```
$ conda create -n [env_name]  
$ conda activate [env_name]  
$ conda deactivate [env_name]
```

list environment or packages

```
$ conda env list  
$ conda list
```

Basic commands

Create environment

```
$ conda create -n [env_name]  
$ conda activate [env_name]  
$ conda deactivate [env_name]
```

list environment or packages

```
$ conda env list  
$ conda list
```

Search tools

```
$ conda search bowtie2  
$ conda search -c bioconda bowtie2
```

Basic commands

Create environment

```
$ conda create -n [env_name]  
$ conda activate [env_name]  
$ conda deactivate [env_name]
```

list environment or packages

```
$ conda env list  
$ conda list
```

Search tools

```
$ conda search bowtie2  
$ conda search -c bioconda bowtie2
```

Install tools

```
$ conda install bowtie2  
$ conda install -c bioconda bowtie2  
$ conda install -c bioconda bowtie2=2.4.5
```

Basic commands

Create environment

```
$ conda create -n [env_name]  
$ conda activate [env_name]  
$ conda deactivate [env_name]
```

list environment or packages

```
$ conda env list  
$ conda list
```

Search tools

```
$ conda search bowtie2  
$ conda search -c bioconda bowtie2
```

Install tools

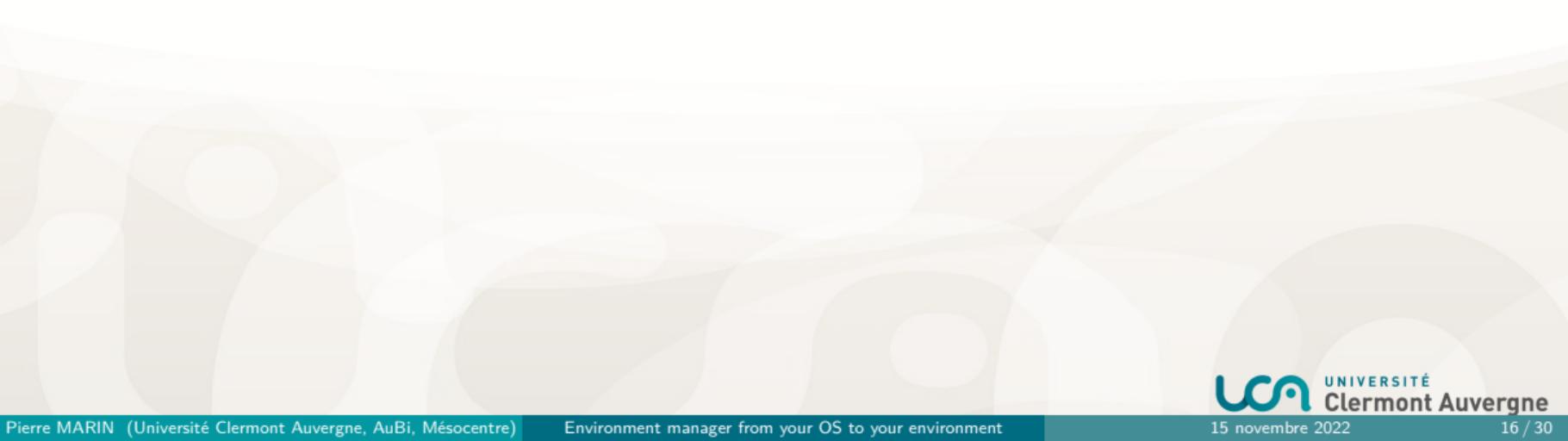
```
$ conda install bowtie2  
$ conda install -c bioconda bowtie2  
$ conda install -c bioconda bowtie2=2.4.5
```

Remove tools

```
$ conda remove bowtie2
```

Environment resolution

Conda also manage your environment to install compatible tools in the same environment



Environment resolution

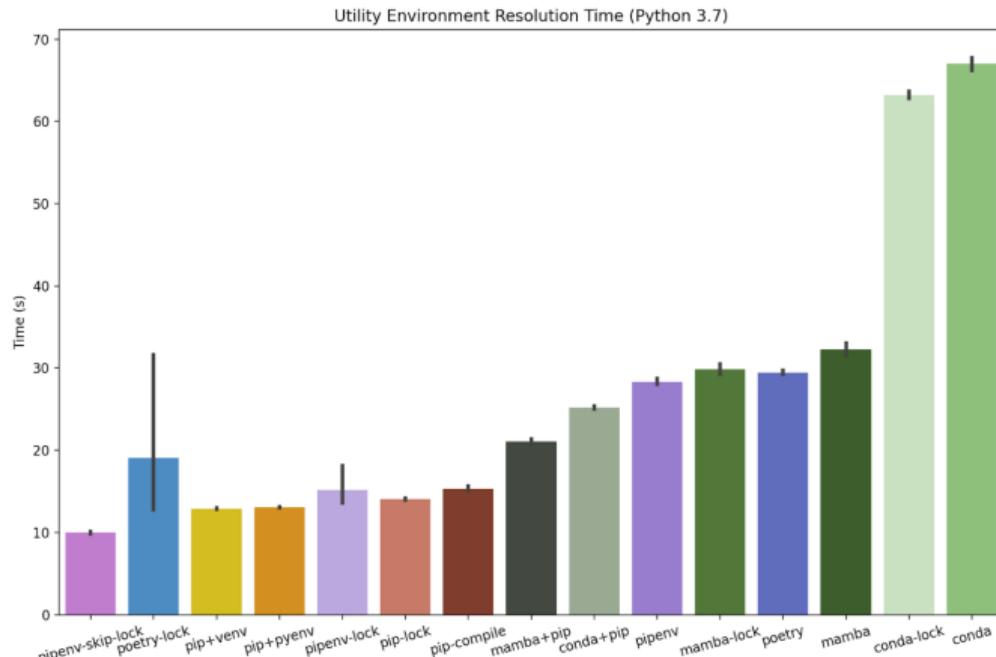
Conda also manage your environment to install compatible tools in the same environment

- time to solve the environment installation plan
- could be locked and don't install a tool

Environment resolution

Conda also manage your environment to install compatible tools in the same environment

- time to solve the environment installation plan
- could be locked and don't install a tool



Container technology is not very old

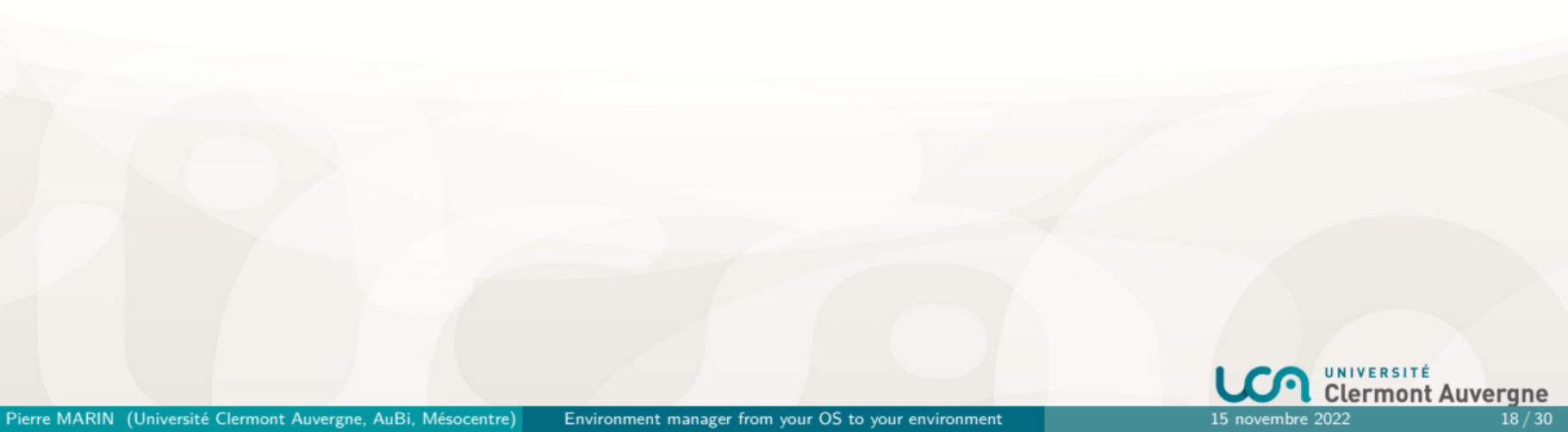
The most famous : 

Solomon Hykes was inspired by container port in the world travel



Docker is an open source project, a community and a private company

- Born in 2010



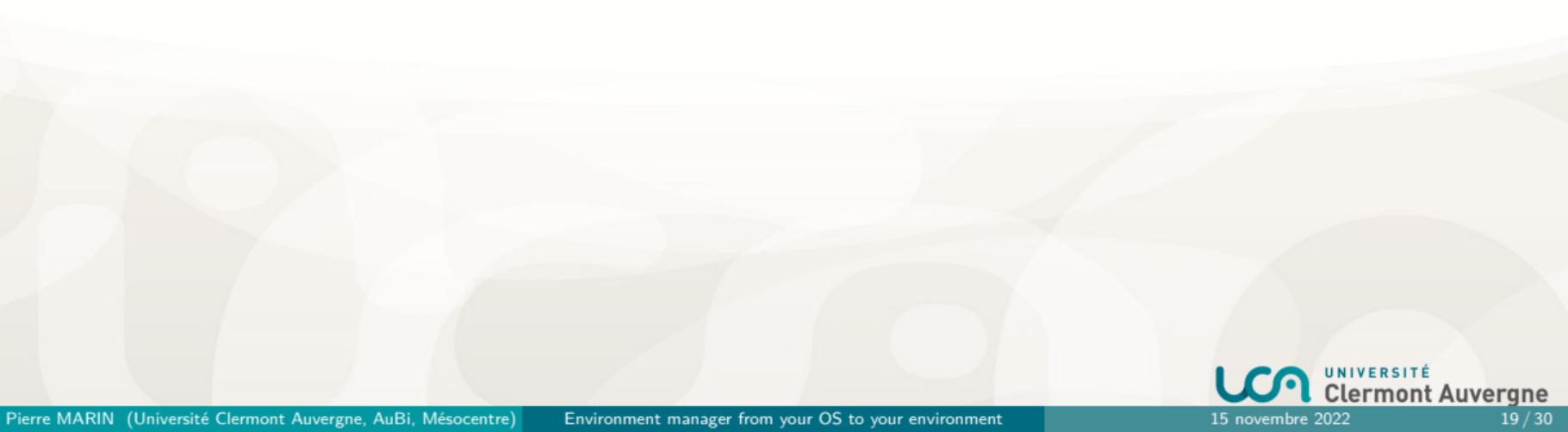
- Born in 2010
- First public release in 2013

- Born in 2010
- First public release in 2013
- V 1.0 in 2014

- Born in 2010
- First public release in 2013
- V 1.0 in 2014
- Open source and free

- Born in 2010
- First public release in 2013
- V 1.0 in 2014
- Open source and free
- Packaged to Ubuntu in 2014 (V14.04)

[<+>]Term definitions



- Docker image → "snapshot" immutable file

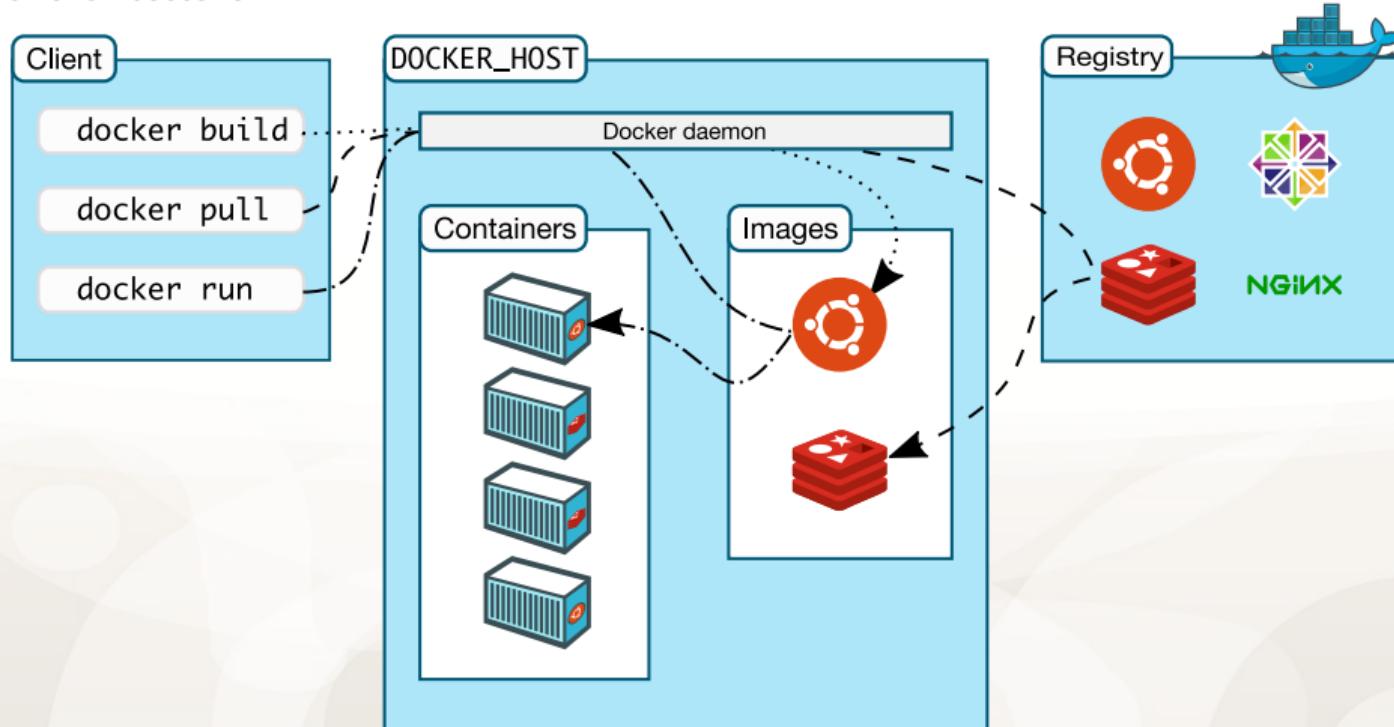
- Set of libraries, functions
- Static state
- Online Store or share
- Automatically build

- Docker container → instance of an image

- Result of the image activation
- Can be modified
- Can be turned into an image
- 1 image → multiple containers

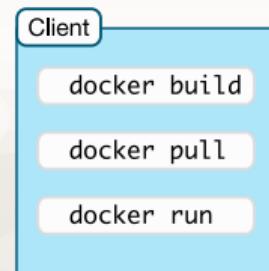
Docker architecture

client-server architecture



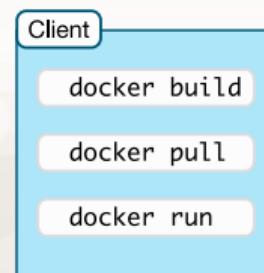
Docker client

1 The user way to interact with Docker



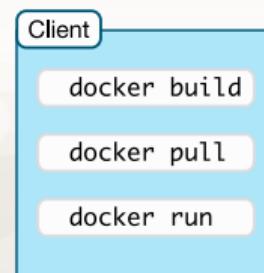
Docker client

- 1 The user way to interact with Docker
- 2 Client talk to a daemon (Docker background program)



Docker client

- 1 The user way to interact with Docker
- 2 Client talk to a daemon (Docker background program)
- 3 API based -> can communicate with several deamons

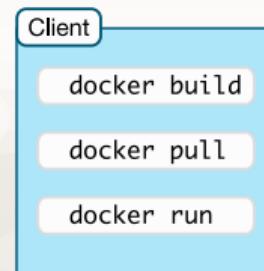


Docker client

- 1 The user way to interact with Docker
- 2 Client talk to a daemon (Docker background program)
- 3 API based -> can communicate with several deamons

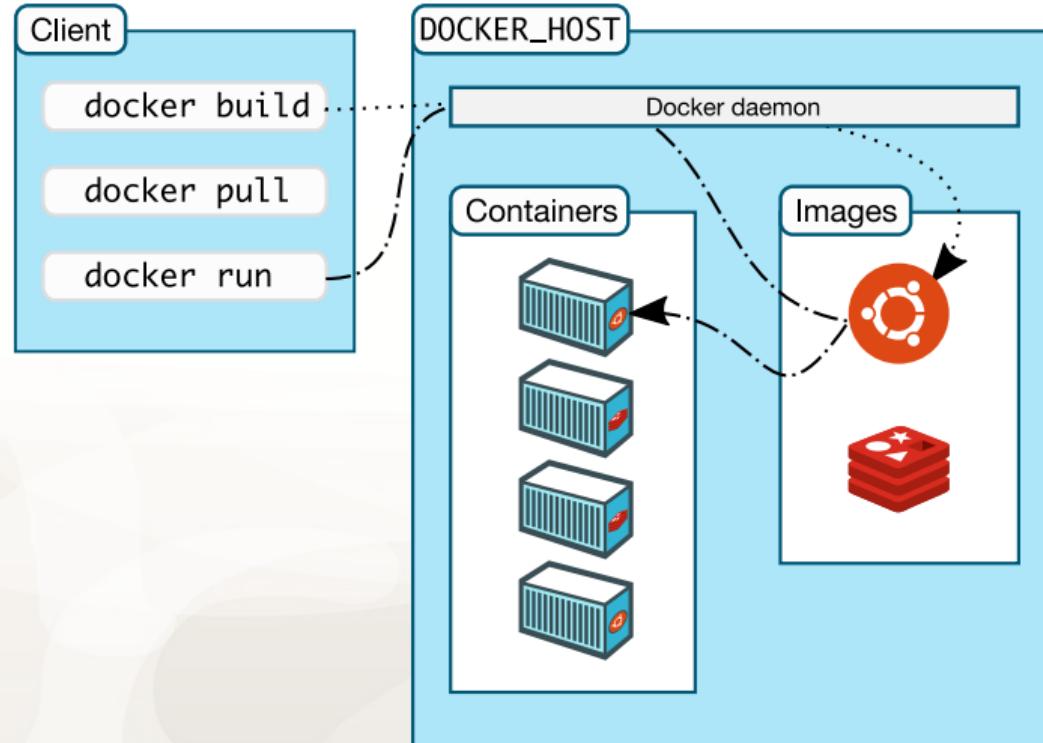
Client

```
$ docker build [path] [url]  
docker build https://github.com/docker/rootfs.git#container:docker  
$ docker pull [image_name]  
docker pull biocontainers/samtools  
$ docker run [image_name]  
docker run biocontainers/samtools
```



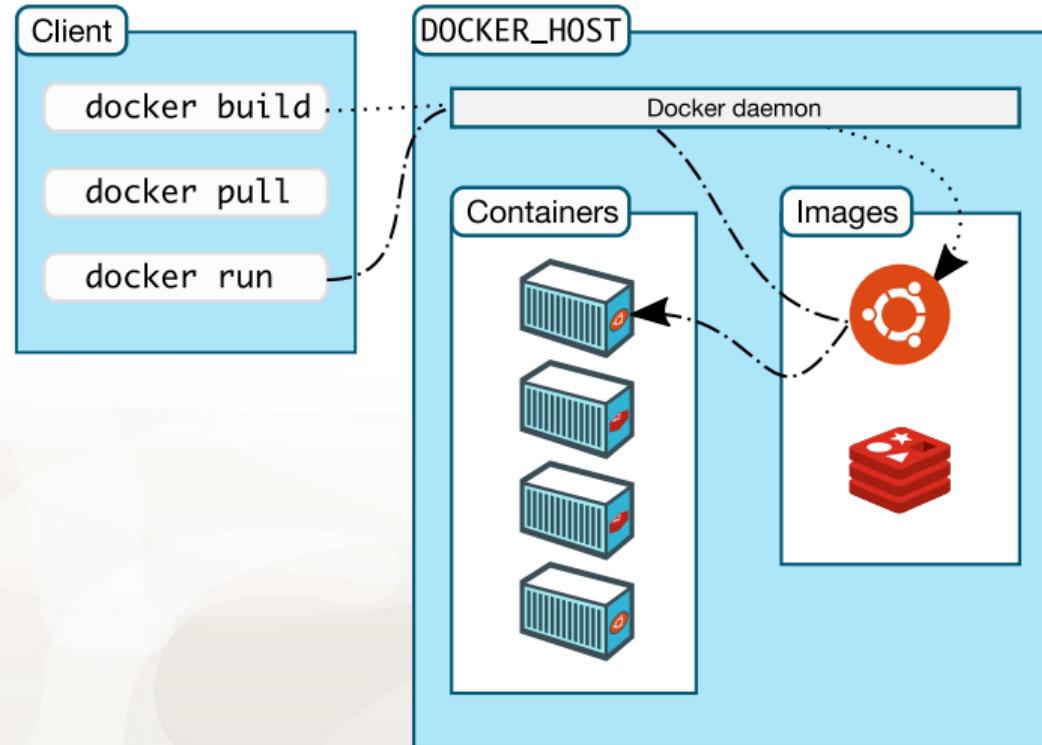
Docker daemon

- 1 listen API requests



Docker daemon

- 1 listen API requests
- 2 manage Docker images, containers...



Docker registries

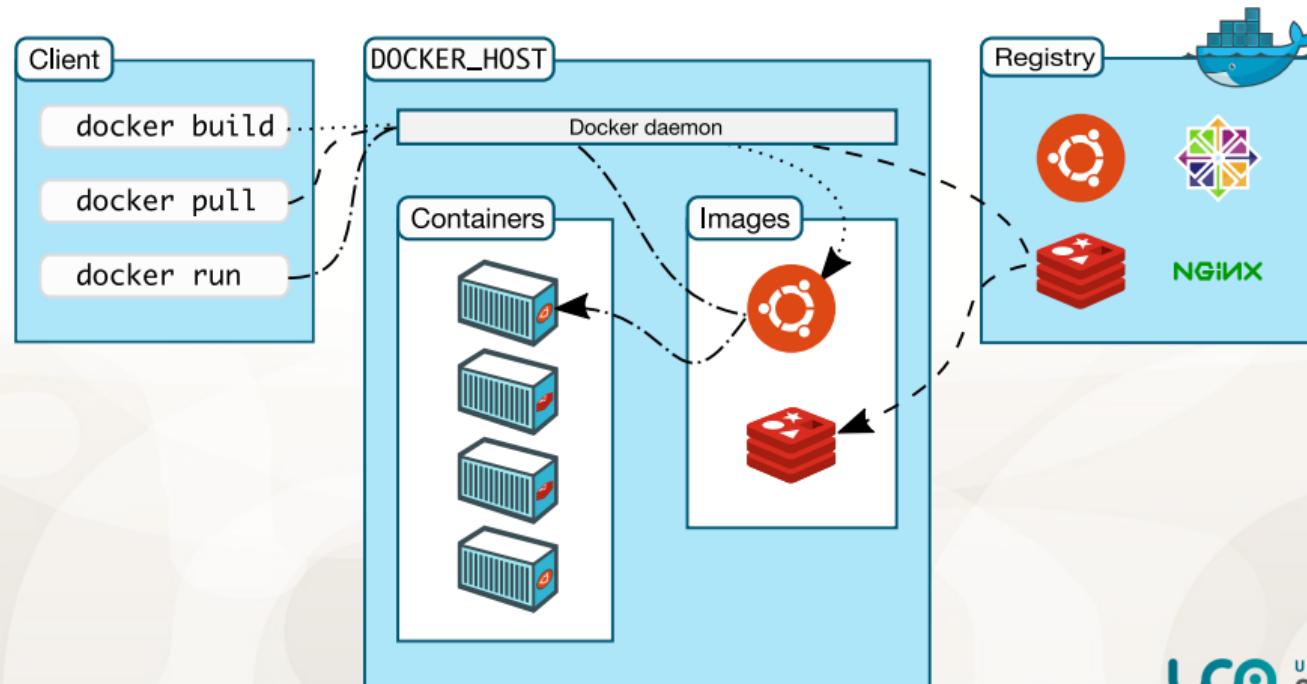
1 Store Docker images

The screenshot shows the Docker Hub search interface. The search bar at the top contains the query 'rstudio'. Below the search bar, there are navigation links for 'Explore', 'Pricing', 'Sign In', and 'Register'. On the left, there is a sidebar with 'Filters' and 'Products' sections. Under 'Products', there are checkboxes for 'Images', 'Extensions', 'Plugins', and 'Trusted Content' (which includes 'Docker Official Image', 'Verified Publisher', and 'Sponsored OSS'). Under 'Operating Systems', there are checkboxes for 'Linux', 'Windows', and 'Architectures' (ARM, ARM 64, IBM POWER, IBM Z, PowerPC 64 LE). The main content area displays search results for 'rstudio' with the following details:

- ibmcom/rstudio-ppc64le** (VERIFIED PUBLISHER)
By IBM • Updated 3 years ago
Integrated development environment (IDE) for R
Linux ppc64le
487 Downloads 4 Stars
- wholetale/rstudio-base** (SPONSORED OSS)
By wholetale • Updated 4 years ago
Linux x86-64
127 Downloads 0 Stars
- bioconductor/rstudio_yescods** (SPONSORED OSS)
By bioconductor • Updated 16 days ago
Linux x86-64
23 Downloads 0 Stars
- truecharts/rstudio** (SPONSORED OSS)
By truecharts • Updated 11 days ago
Linux x86-64
5 Downloads 0 Stars

Docker registries

- 1 Store Docker images
- 2 Docker hub is a public registry



Docker registries

- 1 Store Docker images
- 2 Docker hub is a public registry
- 3 You can run your own registry

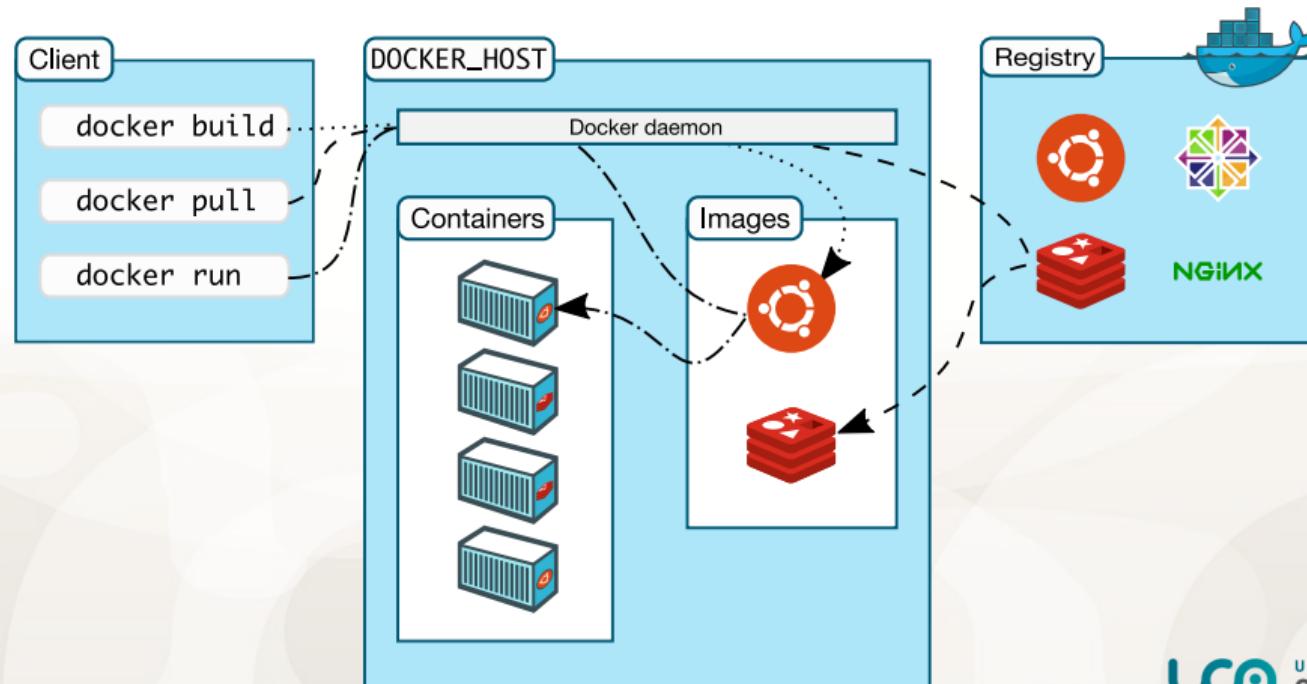


Image layers

Focus on image building

- Layers building

```
$ docker pull debian
Using default tag: latest
latest: Pulling from library/debian
fdd5d7827f33: Pull complete
a3ed95caeb02: Pull complete
Digest: sha256:e7d38b3517548a1c71e41bffe9c8ae6d6d29546ce46bf62159837aad072c90aa
Status: Downloaded newer image for debian:latest
```

Image layers

Focus on image building

- Layers building
- Several layers to one image

```
$ docker pull debian
Using default tag: latest
latest: Pulling from library/debian
fdd5d7827f33: Pull complete
a3ed95caeb02: Pull complete
Digest: sha256:e7d38b3517548a1c71e41bffe9c8ae6d6d29546ce46bf62159837aad072c90aa
Status: Downloaded newer image for debian:latest
```

Image layers

Focus on image building

- Layers building
- Several layers to one image
- Some layers shared by images when pulling

```
$ docker pull debian
Using default tag: latest
latest: Pulling from library/debian
fdd5d7827f33: Pull complete
a3ed95caeb02: Pull complete
Digest: sha256:e7d38b3517548a1c71e41bffe9c8ae6d6d29546ce46bf62159837aad072c90aa
Status: Downloaded newer image for debian:latest
```

Image layers

Focus on image building

- Layers building
- Several layers to one image
- Some layers shared by images when pulling
- Lighten the download and use of image on your computer

```
$ docker pull debian
Using default tag: latest
latest: Pulling from library/debian
fdd5d7827f33: Pull complete
a3ed95caeb02: Pull complete
Digest: sha256:e7d38b3517548a1c71e41bffe9c8ae6d6d29546ce46bf62159837aad072c90aa
Status: Downloaded newer image for debian:latest
```

Docker Cheat Sheet

Build

Build an image from the Dockerfile in the current directory and tag the image

```
docker build -t myimage:1.0 .
```

List all images that are locally stored with the Docker Engine

```
docker images 1.8
```

Delete an image from the local image store

```
docker rmi myimage:3.4
```

Share

Pull an image from a registry

```
docker pull myimage:1.0
```

Retag a local image with a new image name and tag

```
docker tag myimage:1.0 myrepo/myimage:2.0
```

Push an image to a registry

```
docker push myrepo/myimage:2.0
```

Run

Run a container from the Alpine version 3.9 image, name the running container

"web" and expose port 5000 externally, mapped to port 80 inside the container.

Mount a volume --name web -v /tmp:/tmp

Stop a running container through SIGTERM

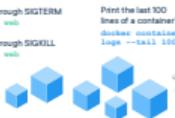
```
docker container stop web
```

Stop a running container through SIGKILL

```
docker container kill web
```

List the networks

```
docker network ls
```



Docker Management

All commands below are called as options to the base docker command. Run docker <command> --help for more information on a particular command.

app* Docker Application

assemble* Framework-aware builds (Docker Enterprise)

builder Manage builds

cluster Manage Docker clusters (Docker Enterprise)

config Manage Docker config

context Manage contexts

engine Manage the Docker Engine

image Manage images

network Manage networks

node Manage Swarm nodes

plugin Manage plugins

registry* Manage Docker registries

secret Manage Docker secrets

service Manage services

stack Manage Docker stacks

swarm Manage swarm

system Manage Docker

template* Quickly scaffold services (Docker Enterprise)

trust Manage trust on Docker images

volume Manage volumes

*Experimental in Docker Enterprise 3.0

DOCKER COMPOSE CHEAT SHEET

Properties

build

build image from dockerfile in specified directory

container1:

properties: values

container2:

properties: values

image

use specified image

image: image-name

container_name

define container name to access it later

container_name: name

types

value

key: value

array

key:

- value

- value

dictionary

master:

key: value

key: value

commands

override start command for the container

command: execute

environment

define env variables for the container

container:

environment:

KEY: VALUE

environment:

- KEY:VALUE

env_file

define a env file for the container to set and override env variables

env_file: .env

env_file:

- .env

restart

define restart rule (no, always, on-failure, unless-stopped)

restart:

- /path:/path

volumes

define container volumes to persist data

volumes:

- /path:/path

depends_on

define build, start and stop order of container

depends_on:

- container-name

networks

define all networks for the container

networks:

- network-name

ports

define ports to expose to other containers and host

ports:

- *9999:9999

expose

define ports to expose only to other containers

expose:

- *9999*

network_mode

define network driver (bridge, host, none, etc.)

network_mode: host

depends_on

define build, start and stop order of container

depends_on:

- container-name

Other

idle container

send container to idle state

> container will not stop

command: tail -f /dev/null

named volumes

create volumes that can be used in the volumes property

services:

container:

image: image-name

volumes:

- data-

volume:/path/to/dir

volumes:

data-volume:

networks

create networks that can be used in the networks property

networks:

bridge:

driver: bridge



Singularity history

- Also a container manager as Docker



Singularity history

- Also a container manager as Docker
- Open-source project

Singularity history

- Also a container manager as Docker
- Open-source project
- Release in 2015

Singularity history

- Also a container manager as Docker
- Open-source project
- Release in 2015
- More adopted in Academic institutions

Singularity history

- Also a container manager as Docker
- Open-source project
- Release in 2015
- More adopted in Academic institutions
- Fork project in 2020 with now AppTainer (linux foundation) and SingularityCE

Singularity history

- Also a container manager as Docker
- Open-source project
- Release in 2015
- More adopted in Academic institutions
- Fork project in 2020 with now AppTainer (linux foundation) and SingularityCE
- HPC compatible, no root whrite, integrate ressource managers (slurm)

Singularity history

- Also a container manager as Docker
- Open-source project
- Release in 2015
- More adopted in Academic institutions
- Fork project in 2020 with now AppTainer (linux foundation) and SingularityCE
- HPC compatible, no root whrite, integrate ressource managers (slurm)
- Could use Docker images

Singularity commands

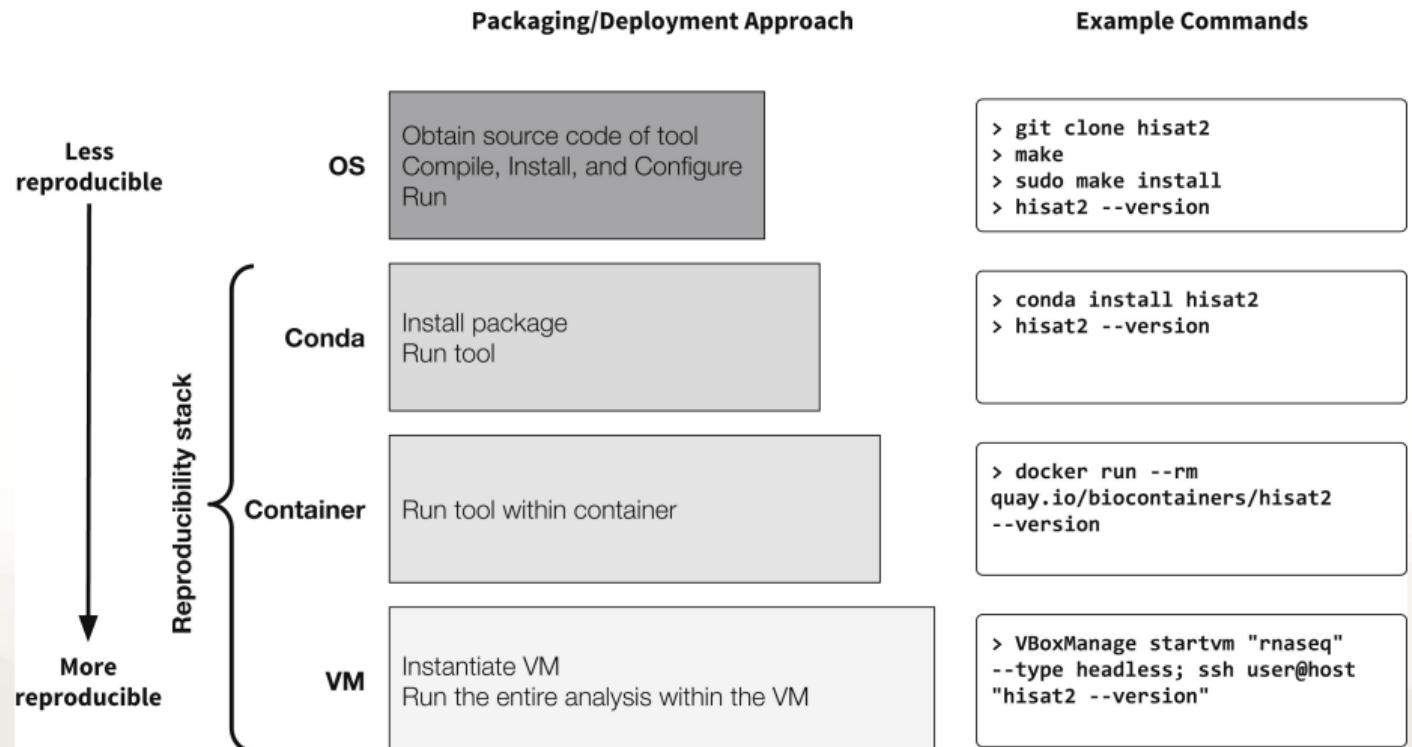
classical commands

```
$ singularity search [image_name]  
$ singularity pull [image_name]  
$ singularity run [image_name]
```

Singularity and Docker

Singularity can use Docker images

```
$ singularity pull docker://debian:latest
INFO:    Converting OCI blobs to SIF format
INFO:    Starting build...
Getting image source signatures
Copying blob f606d8928ed3 done
Copying config 0311b76201 done
Writing manifest to image destination
Storing signatures
2022/10/06 10:50:41  info unpack layer: sha256:f606d8928ed378229f2460b94b504cca239fb9
INFO:    Creating SIF file...
```



4

4. Practical Computational Reproducibility in the Life Sciences Grüning et al, Cell Systems, 2018. DOI 10.1101/j.cels.2018.03.014

Some recommandations⁵

- A package first

5. Recommendations for the packaging and containerizing of bioinformatics software Gruening, F1000 Research, 2019.
DOI 10.12688/f1000research.15140.2

Some recommandations⁵

- A package first
- One tool, one container

5. Recommendations for the packaging and containerizing of bioinformatics software Gruening, F1000 Research, 2019.
DOI 10.12688/f1000research.15140.2

Some recommandations⁵

- A package first
- One tool, one container
- Tool and container versions should be explicit

5. Recommendations for the packaging and containerizing of bioinformatics software Gruening, F1000 Research, 2019.
DOI 10.12688/f1000research.15140.2

Some recommandations⁵

- A package first
- One tool, one container
- Tool and container versions should be explicit
- Avoid using ENTRYPOINT

5. Recommendations for the packaging and containerizing of bioinformatics software Gruening, F1000 Research, 2019.
DOI 10.12688/f1000research.15140.2

Some recommandations⁵

- A package first
- One tool, one container
- Tool and container versions should be explicit
- Avoid using ENTRYPOINT
- Reduce the size of your container as much as possible

5. Recommendations for the packaging and containerizing of bioinformatics software Gruening, F1000 Research, 2019.
DOI 10.12688/f1000research.15140.2

Some recommandations⁵

- A package first
- One tool, one container
- Tool and container versions should be explicit
- Avoid using ENTRYPOINT
- Reduce the size of your container as much as possible
- Keep data outside of the container

5. Recommendations for the packaging and containerizing of bioinformatics software Gruening, F1000 Research, 2019.
DOI 10.12688/f1000research.15140.2

Some recommandations⁵

- A package first
- One tool, one container
- Tool and container versions should be explicit
- Avoid using ENTRYPOINT
- Reduce the size of your container as much as possible
- Keep data outside of the container
- Add functional testing logic

5. Recommendations for the packaging and containerizing of bioinformatics software Gruening, F1000 Research, 2019.
DOI 10.12688/f1000research.15140.2

Some recommandations⁵

- A package first
- One tool, one container
- Tool and container versions should be explicit
- Avoid using ENTRYPOINT
- Reduce the size of your container as much as possible
- Keep data outside of the container
- Add functional testing logic
- Check the license of the software

5. Recommendations for the packaging and containerizing of bioinformatics software Gruening, F1000 Research, 2019.
DOI 10.12688/f1000research.15140.2

Some recommandations⁵

- A package first
- One tool, one container
- Tool and container versions should be explicit
- Avoid using ENTRYPOINT
- Reduce the size of your container as much as possible
- Keep data outside of the container
- Add functional testing logic
- Check the license of the software
- Make your package or container discoverable

5. Recommendations for the packaging and containerizing of bioinformatics software Gruening, F1000 Research, 2019.
DOI 10.12688/f1000research.15140.2

Some recommandations⁵

- A package first
- One tool, one container
- Tool and container versions should be explicit
- Avoid using ENTRYPOINT
- Reduce the size of your container as much as possible
- Keep data outside of the container
- Add functional testing logic
- Check the license of the software
- Make your package or container discoverable
- Provide reproducible and documented builds

5. Recommendations for the packaging and containerizing of bioinformatics software Gruening, F1000 Research, 2019.
DOI 10.12688/f1000research.15140.2

Some recommandations⁵

- A package first
- One tool, one container
- Tool and container versions should be explicit
- Avoid using ENTRYPOINT
- Reduce the size of your container as much as possible
- Keep data outside of the container
- Add functional testing logic
- Check the license of the software
- Make your package or container discoverable
- Provide reproducible and documented builds
- Provide helpful usage message

5. Recommendations for the packaging and containerizing of bioinformatics software Gruening, F1000 Research, 2019.
DOI 10.12688/f1000research.15140.2