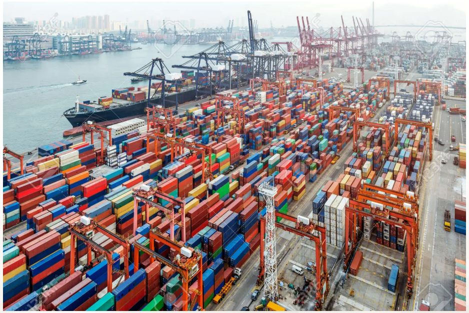
Container technology is not very old

The most famous : docker

Solomon Hykes was inspirated by container port in the world travel



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

- 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

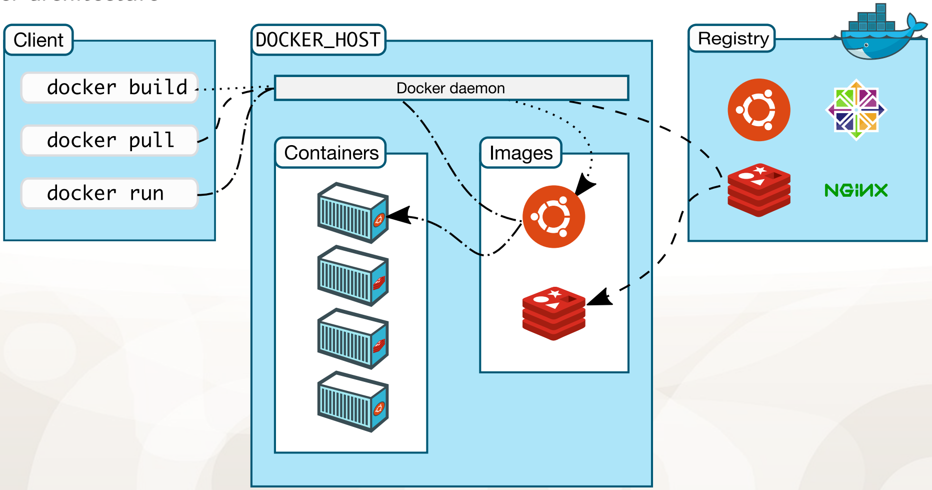
- Docker image –> "snapshot" immutable file
  - Set of libraries, functions
  - Static state
  - Online Store or share
  - Automatically build

- Docker 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 tunred into an image
  - 1 image –> multiple containers

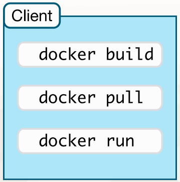
Docker architecture

client-server architecture



Docker client

1. Client to interact with Docker

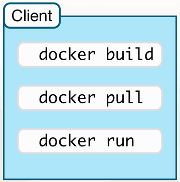


Docker client

1. Client to interact with Docker
2. Client talk to the daemons (Docker background programs)

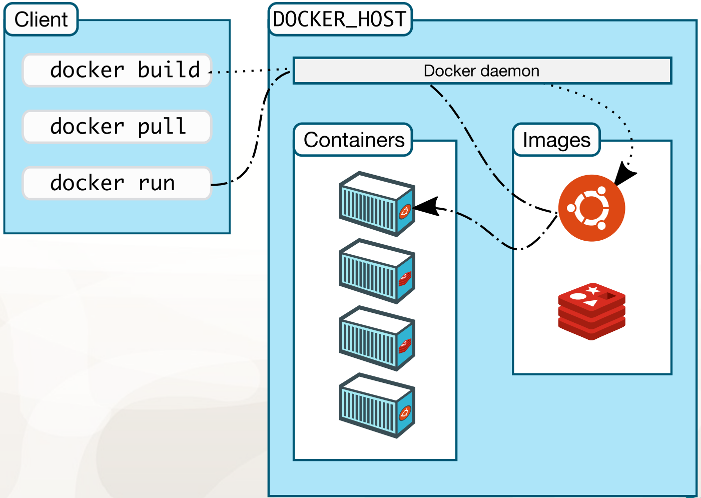
Client

```
$ 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
```



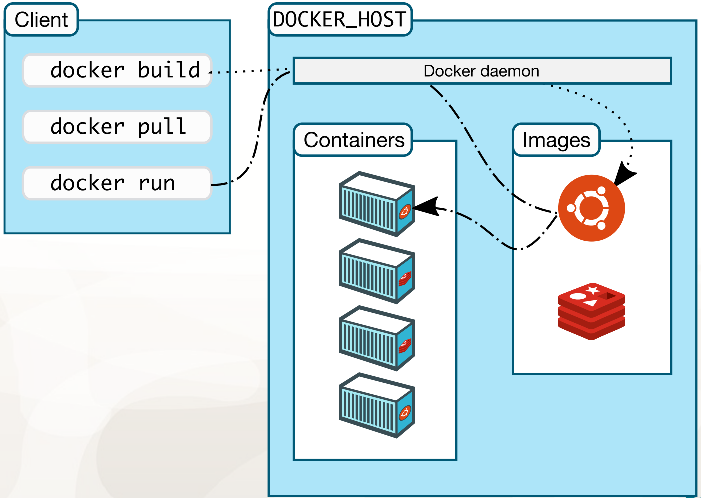
Docker daemon

1. Listen client requests



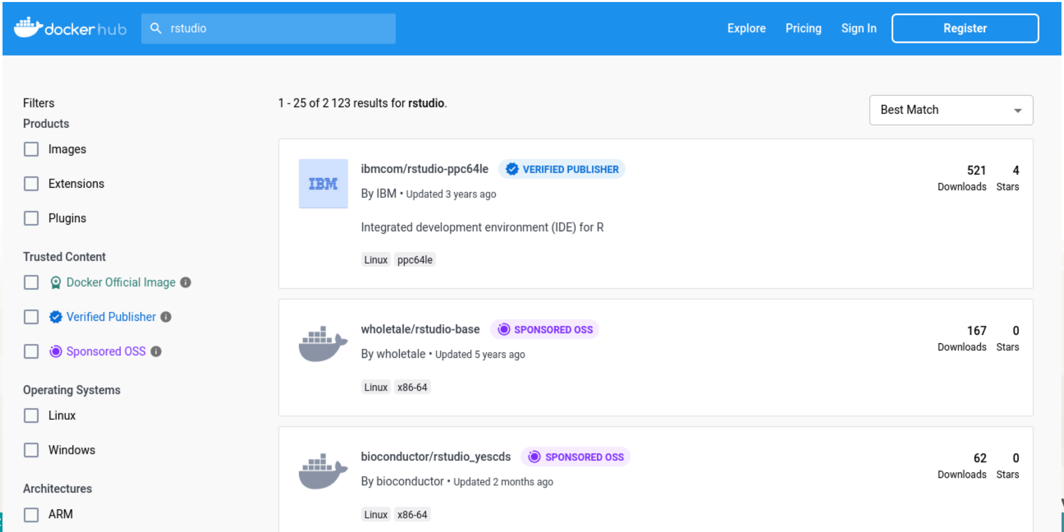
Docker daemon

1. Listen client requests
2. Manage Dockers images, containers...



Docker registries

1 Store Docker images



Filters
Products
☐ Images
☐ Extensions
☐ Plugins

Trusted Content
☐ Docker Official Image
☐ Verified Publisher
☐ Sponsored OSS

Operating Systems
☐ Linux
☐ Windows

Architectures
☐ ARM

1 - 25 of 2 123 results for **rstudio**.

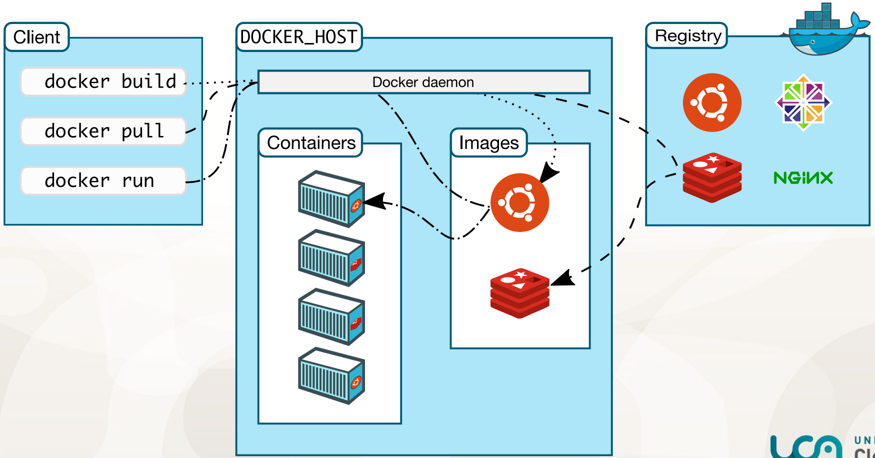
ibmcom/rstudio-ppc64le VERIFIED PUBLISHER
By IBM • Updated 3 years ago
Integrated development environment (IDE) for R
Linux ppc64le
521 Downloads 4 Stars

wholetale/rstudio-base SPONSORED OSS
By wholetale • Updated 5 years ago
Linux x86-64
167 Downloads 0 Stars

bioconductor/rstudio_yescds SPONSORED OSS
By bioconductor • Updated 2 months ago
Linux x86-64
62 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

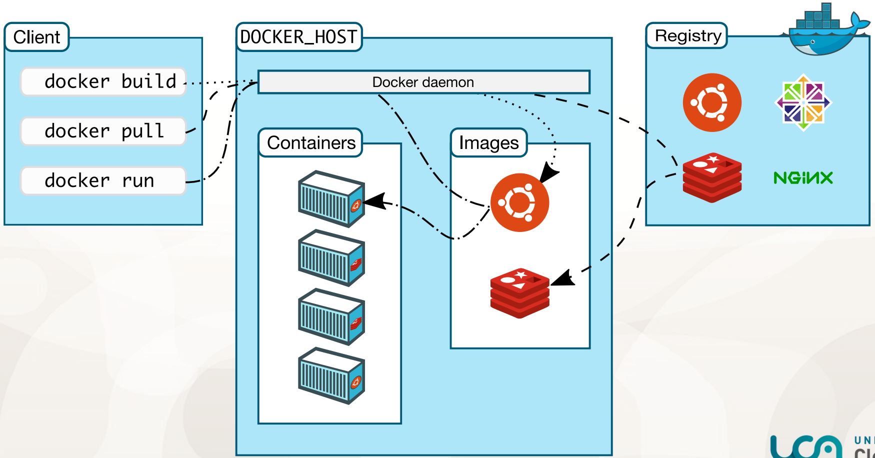


Image layers

Focus on image building

- Layers building

Image layers

Focus on image building

- Layers building
- Several layers to one image

Image layers

Focus on image building

- Layers building
- Several layers to one image
- Some layers shared by images when pulling
- Lightheight the download and use of image on you 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
```

Pull me Hello world !

- Try to pull you first image from docker hub

```
$ docker pull [path/url/docker_name]

$ docker pull hello-world
Using default tag: latest
latest: Pulling from library/hello-world
2db29710123e: Already exists
Digest: sha256:63421b18c1443a9a85139225293fae7541fb40b7832d9deff80b6a9a75ce3604
Status: Downloaded newer image for hello-world:latest
docker.io/library/hello-world:latest
```

Pull me Hello world !

- Now run the hello-world image

```
$ docker image ls
REPOSITORY              TAG       IMAGE ID       CREATED          SIZE
hello-world             latest    feb5d9fea6a5   17 months ago    13.3kB
assembly_conda          latest    5ea57e9c4563   4 hours ago      2.99GB
assembly_raw            latest    dffe598c3a14   4 hours ago      990MB
condaforge/mambaforge   latest    8562647c2abf   12 days ago      393MB
ubuntu                  bionic    b89fba62bc15   2 weeks ago      63.1MB
```

Pull me Hello world !

- Now run the hello-world image

```
$ docker run[image_name/image_tag]
$ docker run hello-world
Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.
```

Build my own image

The basic recipe of Dockerfile

- **FROM** A basic framework (image) as a linux, microsoft for ex.
- **RUN** A command to install a tool

Build my own image

Some few docker specific commands

Instruction	Description
FROM	Image parente
MAINTAINER	Auteur
ARG	Variables passées comme paramètres à la construction de l'image
ENV	Variable d'environnement
LABEL	Ajout de métadonnées
VOLUME	Crée un point de montage
RUN	Commande(s) utilisée(s) pour construire l'image
ADD	(Ajoute un fichier dans l'image *ADD vs COPY)
COPY	Ajoute un fichier dans l'image
WORKDIR	Permet de changer le chemin courant
EXPOSE	Port(s) écouté(s) par le conteneur
USER	Nom d'utilisateur ou UID à utiliser
ONBUILD	Instructions exécutées lors de la construction d'images enfants
CMD	Exécuter une commande au démarrage du conteneur

Build my own image

Dockerfile skeleton

The basic recipe of Dockerfile

- **FROM** A basic framework (image) as a linux, microsoft for ex.
- **RUN** A command to install a tool

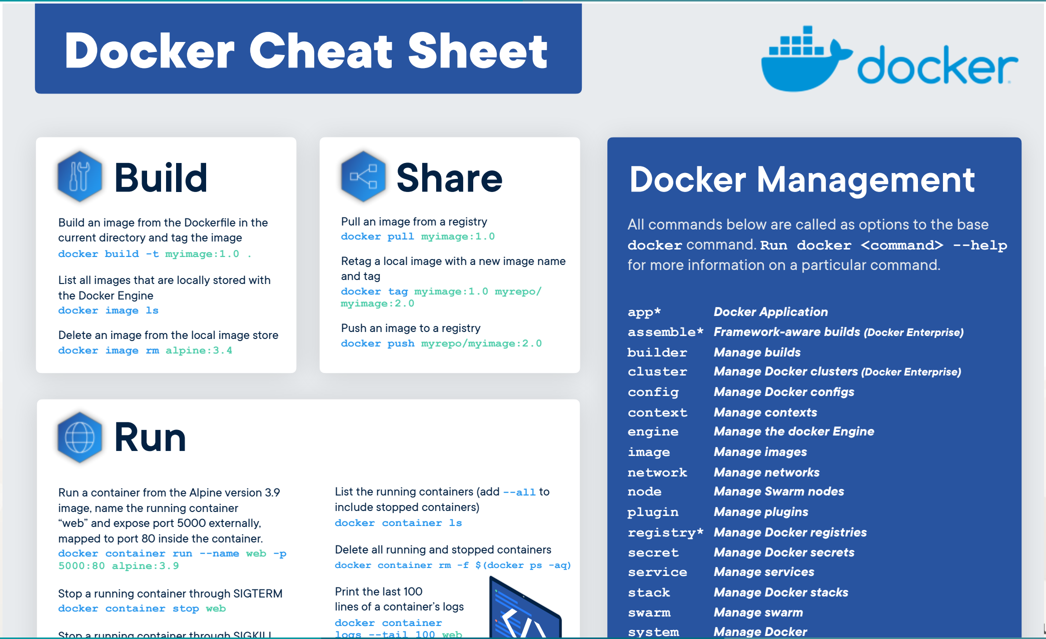
```
FROM ubuntu:bionic
ARG USER="Coco"
LABEL maintainer.email="coco@lasticot.fr"
RUN apt-get update
RUN echo "HELLO WORLD !"
```

Build command

```
$ docker build [url/path] --tag [docker_name]
special case with github url like :
[url]\#[branch_name][file_path]
$ docker build https://github.com/mesocentre-clermont-auvergne/formation_fair.git\#ma
```

Export docker image

```
$ docker save [image_name/ID] > [image_name].tar
```



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 image ls
```

Delete an image from the local image store
```
docker image rm alpine: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.
```
docker container run --name web -p
5000:80 alpine:3.9
```

Stop a running container through SIGTERM
```
docker container stop web
```

Stop a running container through SIGKILL

List the running containers (add `--all` to include stopped containers)
```
docker container ls
```

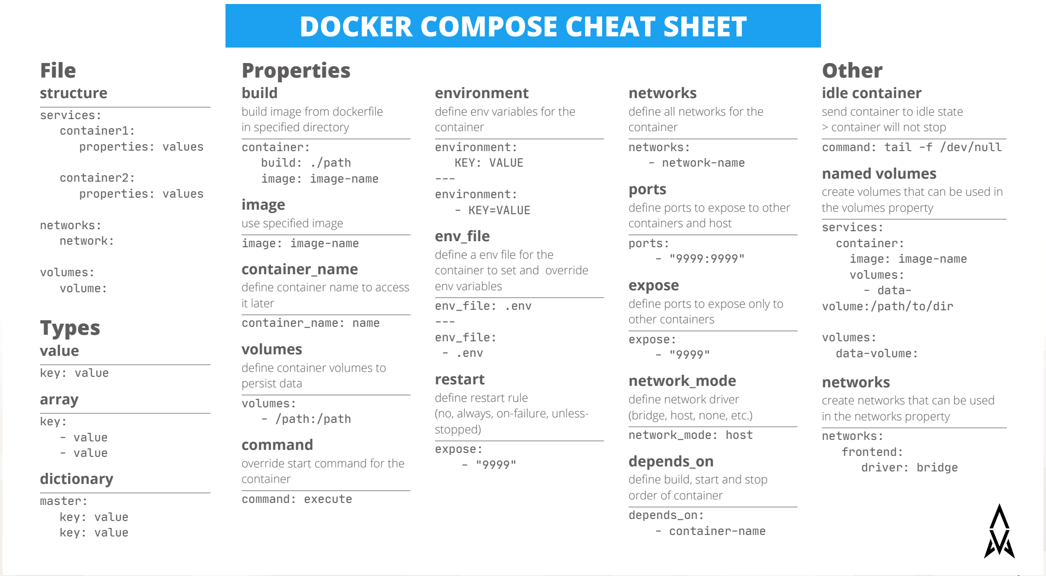
Delete all running and stopped containers
```
docker container rm -f $(docker ps -aq)
```

Print the last 100 lines of a container's logs
```
docker container
logs --tail 100 web
```

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 configs***
`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***



DOCKER COMPOSE CHEAT SHEET

File

structure

```
services:
   container1:
      properties: values

   container2:
      properties: values

networks:
   network:

volumes:
   volume:
```

Types

value

```
key: value
```

array

```
key:
   - value
   - value
```

dictionary

```
master:
   key: value
   key: value
```

Properties

build

build image from dockerfile in specified directory

```
container:
   build: ./path
   image: image-name
```

image

use specified image

```
image: image-name
```

container_name

define container name to access it later

```
container_name: name
```

volumes

define container volumes to persist data

```
volumes:
   - /path:/path
```

command

override start command for the container

```
command: execute
```

environment

define env variables for the 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)

```
expose:
    - "9999"
```

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:
   frontend:
      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
- 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
- Fork project in 2020 with now AppTainer (linux foundation) and SingularityCE
- HPC compatible, no root write, integrate ressource managers (slurm)

Singularity history

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

Singularity commands

Docker command

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

Singularity commands

Docker command

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

Singularity commands

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

Singularity and Docker

Singularity can use Docker images

- from docker hub
- from docker file

```
$ singularity build [new_image_name] docker-archive://[image_name].tar
$ singularity run [new_image_name]
$ 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...
```

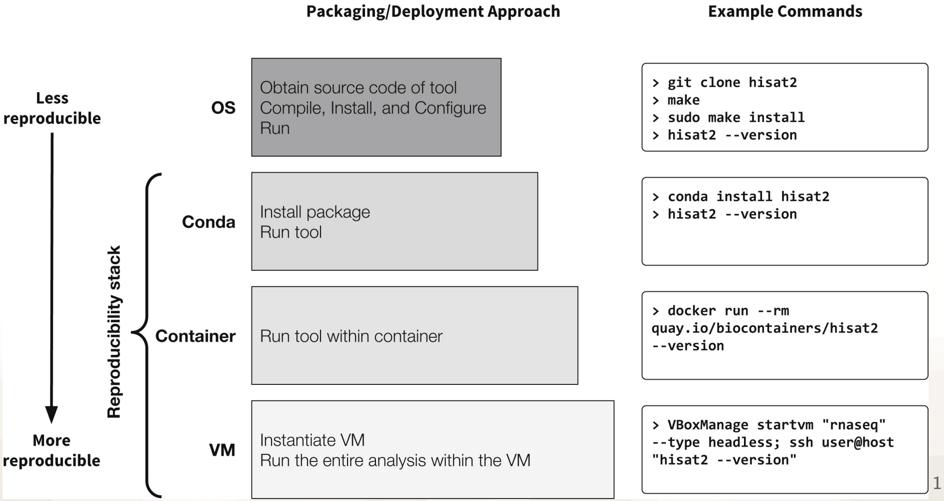
Singularity recipe

Bootstrap	base you want to use (e.g., docker, debootstrap, shub)
From	named container you want to use
%help	help text to the user
%setup	executed command on the host system outside the container
%files	allow to copy file to the containers
%labels	store metadata with your container,
%environment	environment variables sourced at runtime
%post setup	command of your container
%runscript	started when container is running
%test	command to test the image build

Singularity recipe

```
Bootstrap: docker
From: ubuntu:bionic

%help
Help me. I’m in the container.
%labels
    Maintainer "coco l’asticot"
%environment
    VADER=badguy
    LUKE=goodguy
    export VADER LUKE
%post
    echo "Hello World !"
%runscript
    echo "Rooooar!"
```


Packaging/Deployment Approach
Example Commands
Less reproducible
More reproducible
Reproducibility stack
OS
Obtain source code of tool
Compile, Install, and Configure
Run
Conda
Install package
Run tool
Container
Run tool within container
VM
Instantiate VM
Run the entire analysis within the VM

```
> git clone hisat2
> make
> sudo make install
> hisat2 --version
```

```
> conda install hisat2
> hisat2 --version
```

```
> docker run --rm
quay.io/biocontainers/hisat2
--version
```

```
> VBoxManage startvm "rnaseq"
--type headless; ssh user@host
"hisat2 --version"
```

[1]

1. Practical Computational Reproducibility in the Life Sciences Grüning et al, Cell Systems, 2018. DOI 10.1016/j.cels.2018.03.014

Some recommandations

- A package first

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

Some recommandations

- A package first
- One tool, one container

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

Some recommandations

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

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

Some recommandations

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

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

Some recommandations

- 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

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

Some recommandations

- 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

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

Some recommandations

- 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

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

Some recommandations

- 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

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

Some recommandations

- 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

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

Some recommandations

- 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

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

Some recommandations

- 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

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

Encapsulation PRACTICE

Conda, Singularity and Docker