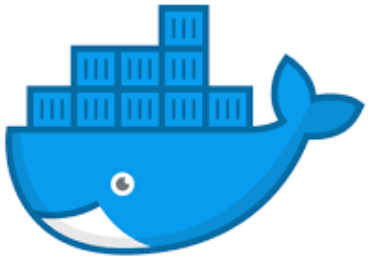
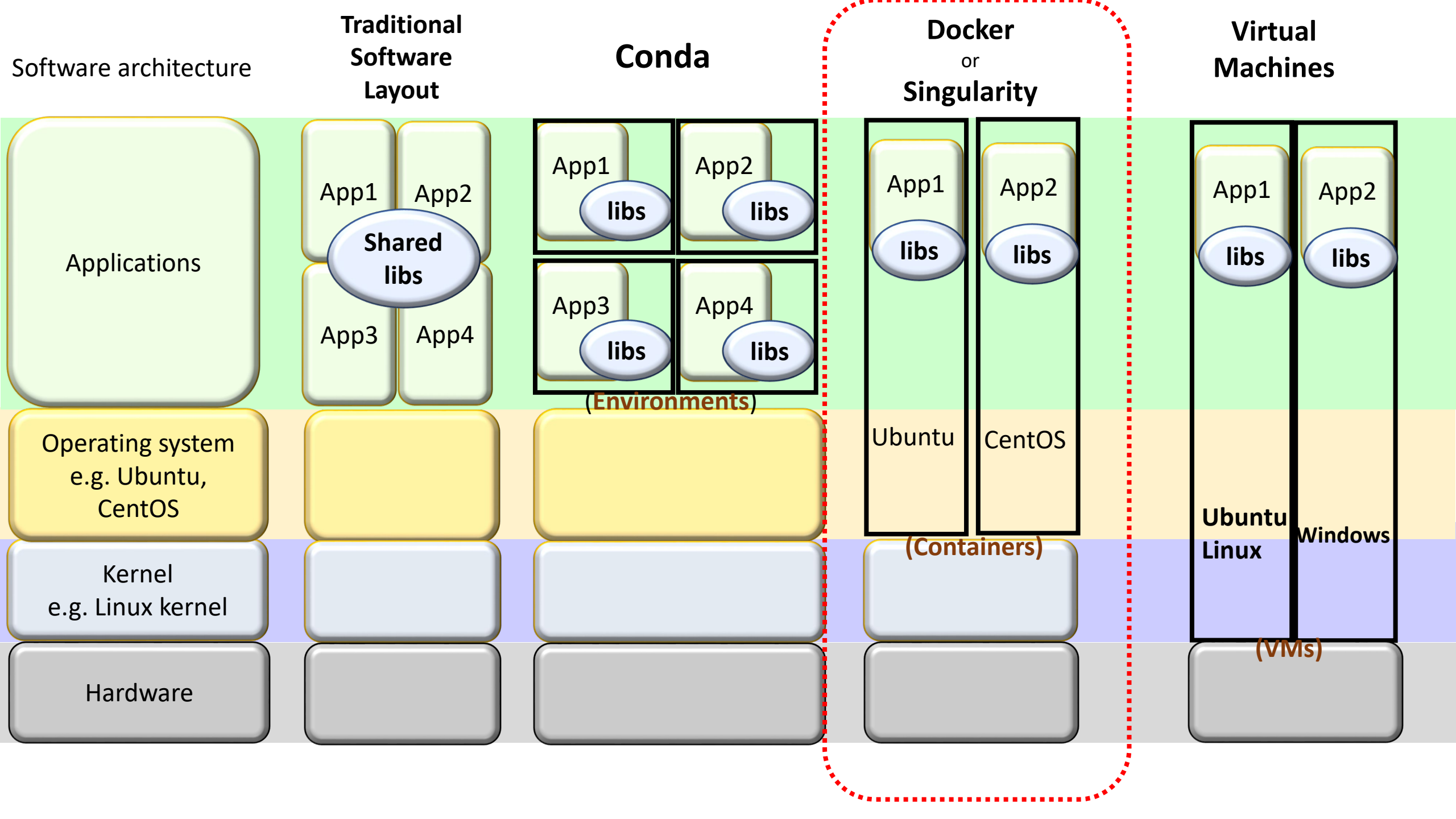
Two Linux container systems

Docker and Singularity




Software architecture
Traditional Software Layout
Conda
Docker
or
Singularity
Virtual Machines
Applications
Operating system
e.g. Ubuntu, CentOS
Kernel
e.g. Linux kernel
Hardware
App1
App2
App3
App4
Shared libs
libs
(Environments)
Ubuntu
CentOS
(Containers)
Ubuntu Linux
Windows
(VMs)

Benefits of Containers

1. Isolation ensures good reproducibility;

2. Good portability between host servers;

3. Consume very little computing power, not like the VM.

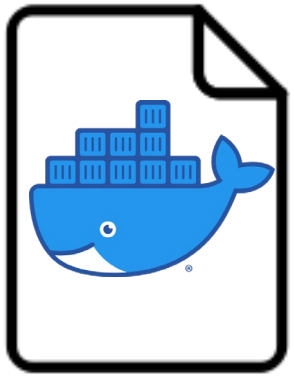
And best of all, if you mess up a container, start a new one.

Overview of Docker

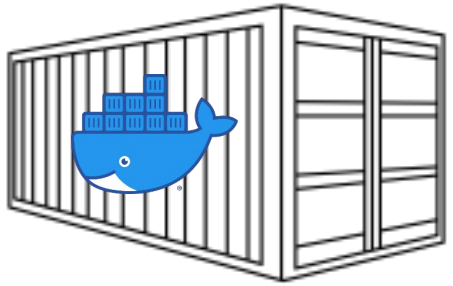
```
FROM ubuntu:18.04
RUN apt-get -y update
RUN apt-get install  r-base

…
```

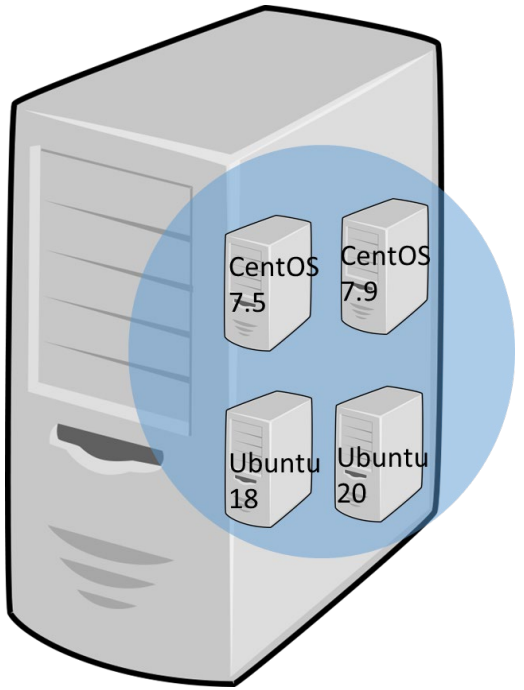
Dockerfile



Docker Image



Docker Container



Dockerfile:

a text file (script) with instructions how to build a Docker image.

- Including name of operating system, its version and where to download;
- Software/libraries, versions and where to download;
- Environment variable in the system

Docker image:

An all-inclusive software file built from the Docker file, including

- Operating system;
- software, libraries.

Docker container:

A running instance of the Docker image.

Dockerfile is not always reproducible for two reasons:

1. The developer often omits the version;
2. The software download link stops working;

Docker image is reproducible.

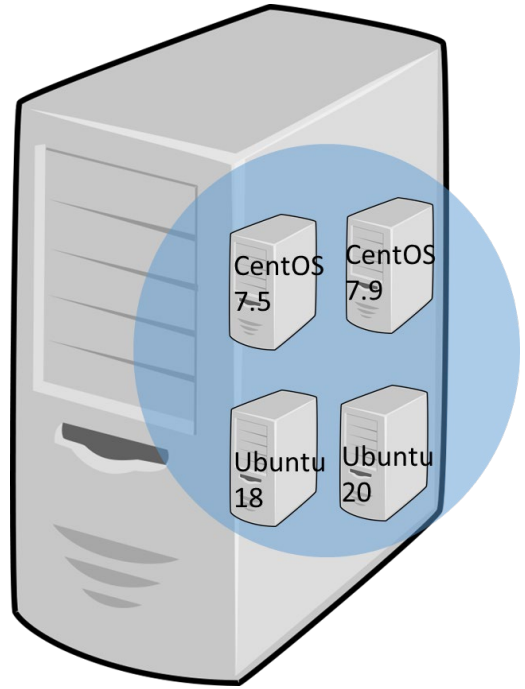
Overview of Singularity

```
BootStrap: library
From: ubuntu:18.04
%post
    apt -y update
    apt -y install r-base
```

Def file

Singularity Image

Singularity container



Singularity definition file (def file):

a text file (script) with instructions how to build a Singularity image.

- Including name of operating system, its version and where to download;
- Software/libraries, versions and where to download;
- Environment variable in the system

Singularity image:

An all-inclusive software file built from the def file, including

- Operating system;
- software, libraries.

Singularity container:

A running instance of the Singularity image.

Singularity def file is not always reproducible for two reasons:

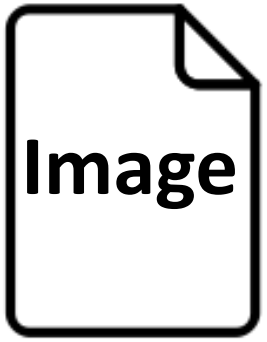
1. The developer often omits the version;
2. The software download link stops working;

Singularity image is reproducible.

Main difference: User ID

Script
.....
.....

Build image



Start container

Run software

Docker

Root Root Root

With Docker, the user can modify root directories on the host system. That is a security risk, and most HPC centers do not allow Docker.

Singularity

Fake root You You

Docker is NOT supported in most HPC systems.

On BioHPC, use **“docker1”** command for “docker”

What is “docker1”?

A script to scan the parameters before passing on to the Docker software, to ensure security of the host.

Among the features of docker1:

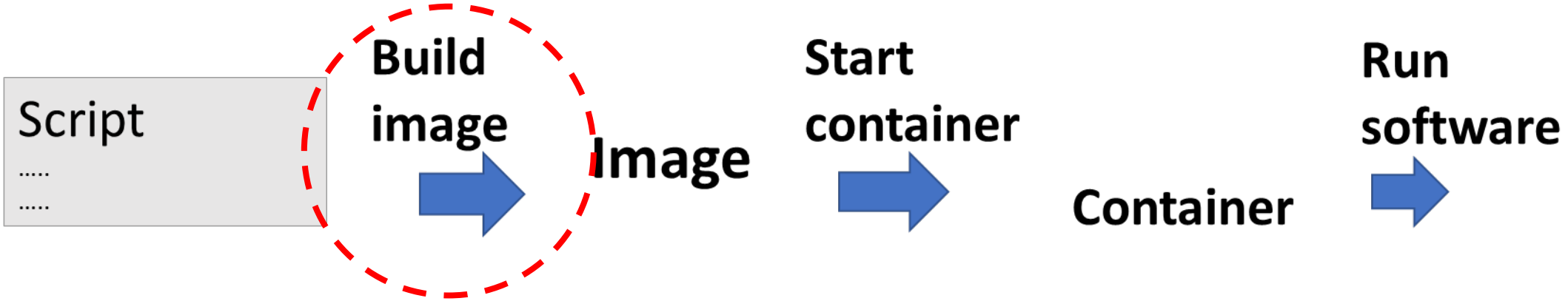
- Only directories under /workdir/$USER can be mounted in Docker container;

- /workdir/$USER is automatically mounted as /workdir in Container;

It is easy to convert a Docker image into a Singularity image.

Docker is good for setting up services in a server, e.g. a web server.

Singularity is easier to setup for computing in HPC cluster. More recent versions of Singularity can also be used for services.



Build a Singularity Image

Two formats of Singularity images

1. .sif file

Read only file. Suitable for production.

2. Sandbox

Writable directory. Suitable for development.

Different ways to build a Singularity image

- Download from a Singularity repository;
- Convert from a Docker image;
- Build from a “Singularity Definition” file;
- Develop in a “Sandbox”, then convert to a .sif file;

Commands to build singularity images

- Download a Docker image and convert to a Singularity image;

```
singularity pull myU.sif docker://ubuntu:20.04
```

- Download a Singularity image;

```
singularity pull  myU.sif  library://library/default/ubuntu
```

* The image is saved as a file “myU.sif”

- **Build from a “Sandbox”**
  - “Sandbox” is a special Singularity container, where you can install and run software as root;
  - After you finish install and test the Sandbox, you can save the sandbox as a “.sif” image file.

```
#build a sandbox
singularity build --fakeroot --sandbox myUbuntu
myUbuntu.def
#start a writable shell
singularity shell --fakeroot --writable myUbuntu
#save sandbox to an image file
singularity build --fakeroot myUbuntu.sif myUbuntu
```

- **Build from a Def file**

An example def file

```
BootStrap: library
From: ubuntu:focal

%environment

%files

%post
    apt -y update
    apt -y upgrade
    apt-get -y install software-properties-common build-essential
cmake wget nano
    add-apt-repository universe
    apt -y update
```

Recommended practice

- Build a sandbox;
- Run Linux command line within sandbox to install software;
- Document each command into a “def” file.

Once your def file is ready, you can build a image .sif file from the definition file

```
singularity build --fakeroot  test.sif test.def
```

You can build a Singularity image starting from either a Docker or a Singularity base image and modify it

From Singularity base image

```
BootStrap: library
From: ubuntu:18.04

%environment

%files

%post
   apt -y update
   apt-get -y install build-essential wget nano
```

From a Docker base image

```
BootStrap: docker
From: rocker/r-ver:4.1.1

%environment

%files

%post
      R --slave -e 'install.packages("BiocManager ") '
```

What is “fakeroot”?

Singularity build command:

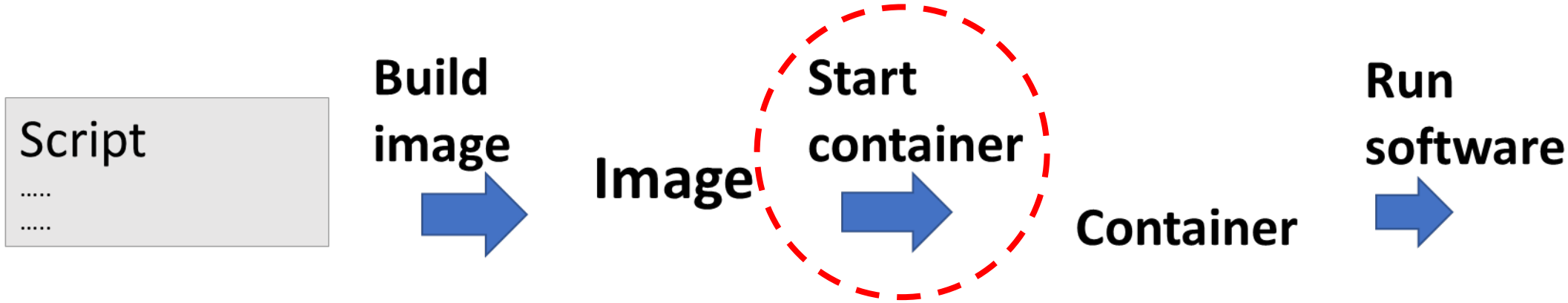
```
singularity build --fakeroot --sandbox myUbuntu myUbuntu.def
```

“singularity build” requires “fakeroot” privilege

A “fakeroot” user has almost the same administrative rights as real “root” but only inside the container. (This is different from docker, which uses real “root”)

On a BioHPC server, run this command to activate “fakeroot” privilege.

```
fakeroot
```



Start Singularity Container

Start Singularity container

1. Interactive singularity shell

#Start a Singularity shell

```
singularity shell myImage.sif
```

#Short format of the same command

```
./myImage.sif
```

#Run software installed in the Container

#exit a shell

```
exit
```

Singularity shell

- It is like a Linux shell, but interactive with operating system within the container

```
[qisun@cbsum1c2b010 qisun]$ cd /workdir/qisun

[qisun@cbsum1c2b010 qisun]$ ./ubuntu.sif
Singularity> ls /
bin   environment  lib    media  proc  sbin          sys  var
boot  etc          lib64  mnt    root  singularity   tmp  workdir
dev   home         local  opt    run   srv           usr

Singularity> whoami
qisun
```

Three directories are from the host, and the rest are from the container.

Your user ID is the same as in host system

By default, Singularity mount your home directory into container, this might not be desirable.

To disable,

```
singularity shell --no-home my_container.sif
```

* The R or Python packages installed in your home directory will be picked up by R or Python within the container. This could interfere with running software in container.

In comparison, here is the Docker container from the same image

```
[qisun@cbsum1c2b010 qisun]$ docker1 run -dit ubuntu:18.04 /bin/bash

[qisun@cbsum1c2b010 qisun]$ docker1 exec -it a2791b6e8a18 /bin/bash

root@a2791b6e8a18:/# ls /
bin   dev  home  lib32  libx32  mnt  proc  run   srv  tmp  var
boot  etc  lib   lib64  media   opt  root  sbin  sys  usr  workdir

root@a2791b6e8a18:/# whoami
root
```

There is no user in /home

/workdir is mounted by docker1

Your user ID is “root” in container

Accessing data files on the host system

Singularity

1. Two directories are mounted by default
    1. Your home directory;
    2. Current directory ($PWD);

```
./myC.sif   myInputDataFile
```

2. You can mount extra directories using “--bind” (or “-B”) parameter.

```
singularity shell --bind /workdir:/data myC.sif
```

(You can mount any directories that you have the read/write permission)

Docker (docker1)

1. “/workdir/$USER” is mounted as “/workdir” in container;

2. You can mount extra directories using “--mount” (or “-v”) parameter.

```
docker1 run -v  /workdir/$USER/data/:/data ubuntu
```

(You can only mount directory under “/workdir/$USER” or “/local/storage”)

If you create any new files in container

Singularity

The owner of the files is the your BioHPC user ID.

Docker (docker1)

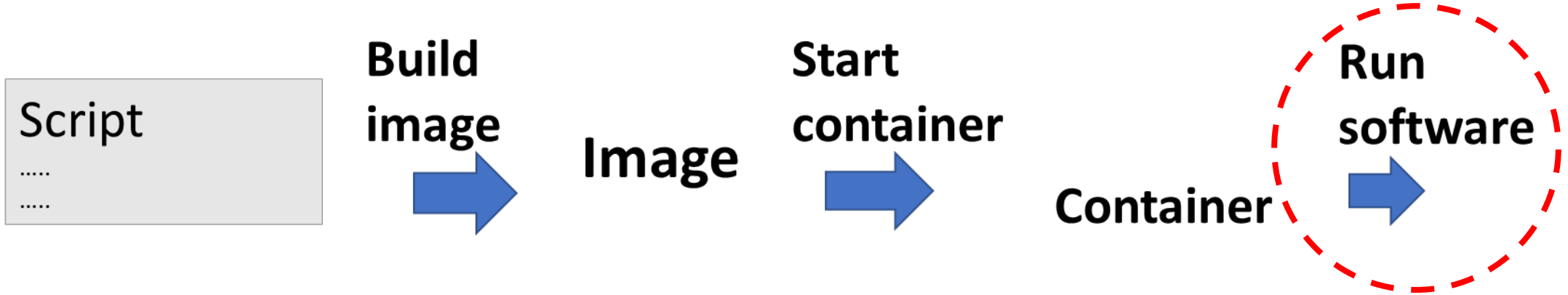
The owner of the file is **root**

To claim every file under /workdir/$USER

```
docker1 claim
```

To claim on file/directory

```
docker1 claim PATH_TO_THE_DIRECTORY
```



Run software in Singularity

Two alternative ways

- Run software through interactive shell
- Run software directly without interactive shell

1. Run software in Singularity shell

```
[qisun@cbsum1c2b010 qisun]$ ./ubuntu.sif

Singularity>bwa index mygenome.fasta
```

2. Run software in Singularity container directly without interactive shell

#Run a software (e.g. bwa)

```
singularity exec myImage.sif bwa mem genomeDB s1.fastq.gz
```

#Short format of the same command

```
./myImage.sif bwa mem genomeDB s1.fastq.gz
```

#If set the container's default software as bwa:

```
./myImage.sif mem genomeDB s1.fastq.gz
```

Singularity passes all environment variables into container.
Sometimes, this is not desirable

```
singularity shell --cleanenv my_container.sif
```

If you have environment variable PYTHONPATH set in your host machine, this might interfere with Python in container.

Usage example 1: Running Busco

Using Singularity

```
#download and build image
singularity pull busco.sif docker://ezlabgva/busco:v5.2.2_cv1

#run busco
./busco.sif busco -i myGenome.fasta -m genome -o results
```

Using Docker

```
#download image
docker1 pull ezlabgva/busco:v5.2.2_cv1

#run busco
docker1 run --rm ezlabgva/busco:v5.2.2_cv1 busco -i
/workdir/myGenome.fasta -m genome -o /workdir/results


docker1 claim /workdir/$USER/results
```

Usage example 2: Run R4.1.1

Singularity

```
#download and build image
singularity pull R4_1.sif docker://rocker/r-ver:4.1.1
# Start container and R-shell
./R4_1.sif
# Install R packages into your home directory
install.packages ("BiocManager")
```

* Without parameter “--no-home” , R can use all packages installed in your home directory.

Docker

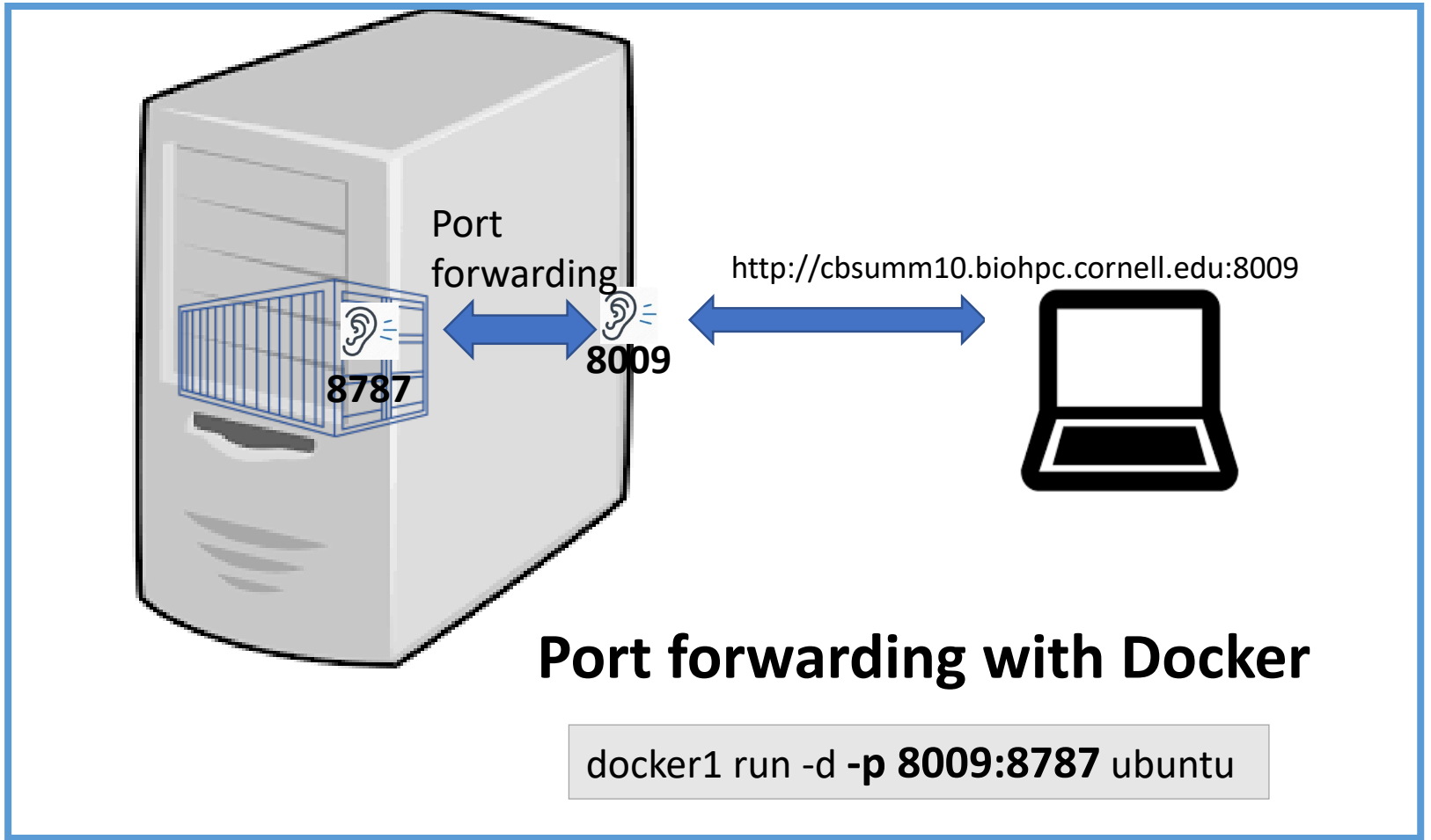
```
#download image, start container and R-shell in one step
docker1 run --rm -it rocker/r-ver:4.1.1

# Install R packages into root home directory inside container
install.packages ("BiocManager")
```

Network port

Singularity: No network isolation. No port forwarding by default.

Docker: Isolated container network. Port forwarding is required for network services.



NVIDIA GPU access

Singularity

Natively supported. But you do need the “--nv” option to enable NVIDIA GPU in container

singularity shell --nv

Docker

- NVIDIA Container Toolkit allows users to build and run GPU accelerated Docker containers.
- The “nvidia_docker” is installed on all BioHPC GPU machines, and docker1 points to nvidia_docker.

In summary

Singularity

- Isolated but with a few doors open by default
  - Home directory and current directory are mounted;
  - Host environment variables are inherited;
  - User ID inherited;
  - No network isolation;

- You can optionally close all the doors.

  For example:
  - --cleanenv: environment variable not inherited;
  - --no-home: home directory not mounted;
  - Network virtualization

Docker

- Almost fully isolated by default.

- But you can optionally open the doors.

  For example:
  - -v: mount directories;
  - -p: forwarding ports;
  - As you run as root, you can add user IDs from host;

From the Docker home page:

… The ***isolation*** and *security* allow you to run many containers simultaneously on a given host …

From the Singularity home page:

… philosophy of Singularity is ***Integration*** over *Isolation*

…