

Review of Docker commands

Images

```
docker images #list images
```

```
docker run ubuntu <cmd_in_container>
```

```
docker save -o img.tar ubuntu
```

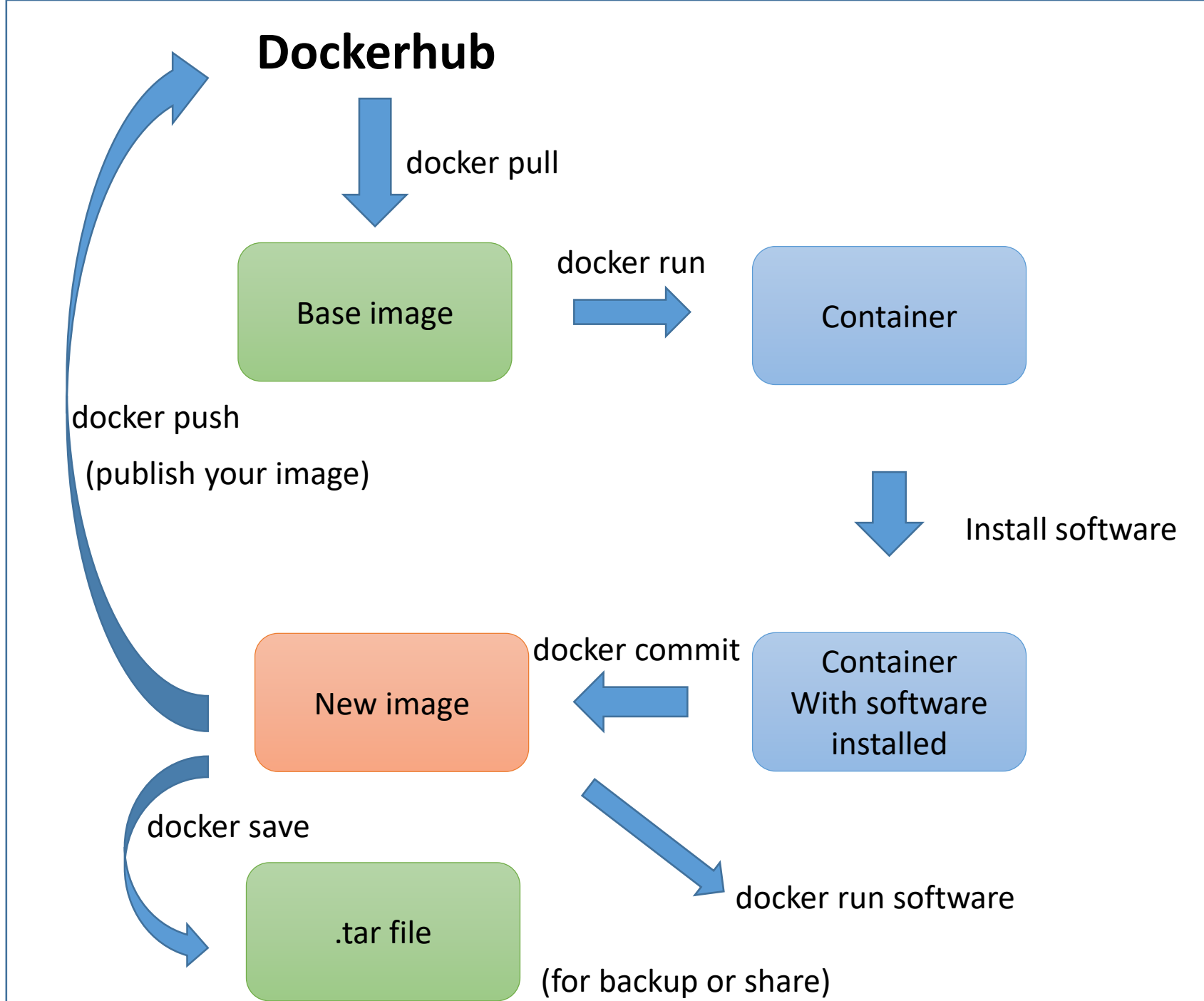
Containers

```
docker ps -a #list containers
```

```
docker exec 39sadf0a93 <cmd_in_container>
```

```
docker commit 39sadf0a93 img_name
```

Typical workflow



Review of Docker commands

`docker run --rm -v /workdir/qisun/data:/data -w /data ezlabgva/busco:v5.4.3_cv1 busco -h`

Command on host side

Image name

Command in
container

```
export BUSCO="docker run --rm -v /workdir/qisun/data:/data -w /data ezlabgva/busco:v5.4.3_cv1"
```

```
$BUSCO busco -h
```

Commonly used command options

`docker run -dit ubuntu:20.04` # -dit: start a docker container, running in background

`docker run -it --rm myimg myscript.py` # --rm: remove container after job finished; -it: in interactive mode

`docker run --rm -v /workdir/$USER/data:/data myimg myscript.py` #-v: mount a volume

`docker run --rm -v /workdir/$USER/data:/data -w /data myimg myscript.py` #-w: current dir

`docker exec -it asd9f7dasfa bash` # execute a command (e.g. "bash") in a running container, interactively

Conda in Docker

- Why?
- How?

Run BUSCO in Conda

```
source ~/miniconda3/bin/activate  
  
conda create -n busco busco  
  
conda activate busco  
  
busco
```

~/miniconda3/env/busco:

- 3.1 GB, 41442 files

Run BUSCO in Docker

```
docker run --rm ezlabgva/busco:v5.4.3_cv1 busco
```

Image size:

- 2.28 GB, single file

Run BUSCO in Singularity

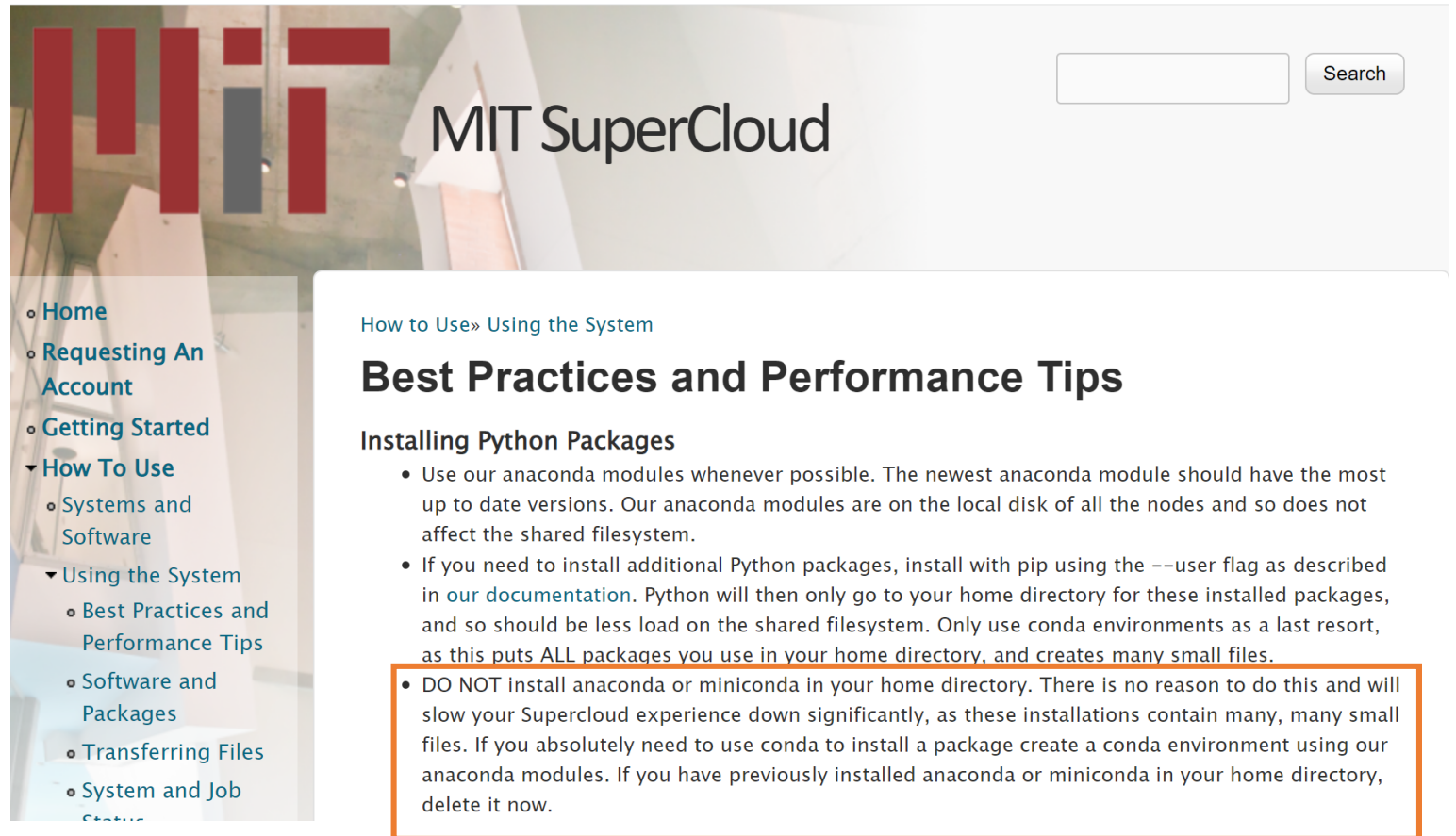
```
singularity build busco.sif docker://ezlabgva/busco:v5.4.3_cv1  
  
./busco.sif busco
```

Image size:

- 0.73 GB, single file

Why is Conda bad for the cloud?

- To many small files in Conda is a huge burden for the cloud storage;
- Not fully encapsulated;



The image is a screenshot of the MIT SuperCloud website. At the top, there is a search bar with the text 'MIT SuperCloud' and a 'Search' button. Below the search bar is a navigation menu with the following items: Home, Requesting An Account, Getting Started, How To Use (expanded), Systems and Software, Using the System (expanded), Best Practices and Performance Tips, Software and Packages, Transferring Files, and System and Job Status. The main content area is titled 'How to Use» Using the System' and 'Best Practices and Performance Tips'. Under the heading 'Installing Python Packages', there are three bullet points. The third bullet point is highlighted with an orange border and reads: 'DO NOT install anaconda or miniconda in your home directory. There is no reason to do this and will slow your Supercloud experience down significantly, as these installations contain many, many small files. If you absolutely need to use conda to install a package create a conda environment using our anaconda modules. If you have previously installed anaconda or miniconda in your home directory, delete it now.'

Solution: Put Conda in Docker/Singularity

Two base images for building Conda in Docker:

`continuumio/miniconda3`

Published by Anaconda

`mambaorg/micromamba`

Published by Mamba

Conda in Docker/Singularity

Protocol: <https://biohpc.cornell.edu/doc/CondaInContainer.html>

Using Dockerfile

create a Docker file

```
FROM continuumio/miniconda3
RUN conda install -y -c bioconda -c conda-forge \
    python=3.9.1 \
    samtools \
    bwa
RUN conda clean --all --yes
```

#build image

```
docker1 build -t samimage /workdir/qisun/
```

Interactively

```
docker run --dit continuumio/miniconda3

docker exec -it 348q0840qaf bash

conda install -y -c bioconda -c conda-forge \
    python=3.9.1 \
    samtools \
    bwa
conda clean --all yes

exit

conda commit 348q0840qaf samimage
```


#after the image is created, run command

```
docker run --rm samimage samtools
```

- **Conda directory in Docker: /opt/conda;**
- **Install in Conda “base”. No need to create environment;**
- **The “base” environment is activated automatically;**

Advantage of Conda in Docker

1. One single file, instead of thousands or even millions;
2. Fully encapsulated, portable and reproducible;

Strongly recommended !!!



Singularity (new name Apptainer)



- Another container system;
- Why do we need it?

Docker vs Singularity

Problem of Docker:

- Run as root;
- Not supported in most HPC systems;
- Docker1 on BioHPC is a restricted Docker implementation;

Advantage of Singularity:

- Run as same user ID as the host;
- Supported in almost all HPC systems;

Disadvantage of Singularity:

- A newcomer, no community supported large database like Dockerhub

Build singularity images

Singularity image file: *.sif

Convert from Docker image

Docker image in Dockerhub

```
singularity build myimg.sif docker://rocker/r-ver
```

```
ls -l myimg.sif
```

```
-rwxr-x--- 1 qisun qisun 319188992 Oct 26 20:28 myimg.sif
```



An executable file

Build singularity images

Singularity image file: *.sif

From Docker image

A local Docker image tar file

```
singularity build myimg.sif docker-archive://myimg.tar
```

Build a Singularity image without Docker

From a Singularity .def file

Create a text file myapp.def

```
BootStrap: library
From: ubuntu:focal
%environment
%files
%post
  apt -y update
  apt install -y build-essential
```

Build command:

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

Interactively with a Singularity sandbox

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

#start a shell in sandbox
singularity shell --fakeroot --writable myUbuntu

## install all needed modules in the sandbox

## convert the sandbox into a sif file
singularity build --fakeroot myUbuntu.sif myUbuntu
```

* I found it easier to build a Docker image, and then convert the Docker image into a Singularity image.

Run software in Singularity

```
singularity run img.sif cmd
```

* -C: Contain all. By default, environment and home directory are carried into the container;

```
singularity run -C img.sif cmd
```

* --bind: mount host directory into container. By default, home directory is mounted;

```
singularity run --bind /workdir:/data img.sif cmd
```

* --pwd: set default directory

```
singularity run --pwd /data img.sif cmd
```


Shortcuts in “singularity” commands

Full command : safest

```
singularity run -C \  
    --bind /workdir/$USER:/data \  
    --pwd /data \  
img.sif cmd
```

Shortest command

```
./img.sif
```

- Home directory and environment are carried from host;
- No data file bound;
- Default command and current directory setting are built into the image;

Lots of shortcuts in “singularity” commands

Full command : safest

```
singularity run -C \  
--bind $PWD \  
--pwd $PWD \  
img.sif cmd
```

- \$PWD: current directory
- If no “:”, source and destination directories are the same

Shortest command

```
./img.sif
```

- Home directory and environment are carried from host;
- No data file bound;
- Default command and current directory setting are built into the image;

By default, Singularity is not fully contained.

These items are carried into the container:

1. Home directory (R and Python modules, setting files in home directory);
2. Environment variables (e.g. PYTHONPATH, PATH)
3. Current directory (rule changed in latest Singularity)

If you run into problems, do “singularity run -C” to contain everything, and then explicitly bind the directories and set default PWD

```
singularity run -C \  
    --bind /workdir/$USER:/data \  
    --pwd /data \  
img.sif cmd
```

“singularity shell”: an interactive interface

```
singularity shell ubuntu.sif
```

```
Apptainer> pwd
```

```
/home/qisun
```

```
Apptainer> apt update
```

```
Permission denied
```

- Within container, only the mounted directories are writable;
- You cannot install extra software.
- To install software, the image needs to be converted into “Singularity sandbox” first.
- I found it easier to do all software installation in Docker first, and then to convert the Docker image to a Singularity image.

My protocol of using Docker and Singularity

1. Identify the base Docker image;
2. Build a Docker image interactively. Make sure to document every step;
3. (optional) Create a Docker file based on documented steps;
4. Convert Docker image into a Singularity image;
5. Save a copy of the Docker image .tar file;
6. Run software with Singularity image;