Singularity – Exercises

Part 1. Working with Singularity containers

1.1 Download a pre-built image for ubuntu and save into a file

```
mkdir /workdir/$USER

cd /workdir/$USER

#Pull the Ubuntu image from Singularity repository
singularity pull ubuntu.sif library://ubuntu:20.04

#Pull the same image from the Docker hub and convert to a Singularity image
singularity pull ubuntu_d.sif docker://ubuntu:20.04

#check the sizes of the two image files you just downloaded
ls -l *.sif
```

1.2 Explore the container

```
#Check the operating system of the host, you should see CentOS
head /etc/os-release

#Start the container, and enter the shell
singularity shell ./ubuntu.sif

#Now check the operating system of the container, you would see Ubuntu
head /etc/os-release

#List the root directory of the container, can you tell which are from the host?
which are from the container?
ls -l /

#List the home directory in the Container, you should see a single user which is
you.
ls -l /home

#your user id inside container
whoami

#exit the container
exit
```

Alternatively, you can start the container with a shorter command:

```
./ubuntu.sif

head /etc/os-release

exit
```

1.3 Execute a command in container

Execute the Linux command within the container without using the interactive shell

```
#The following command run "ls -l" inside the container
singularity exec ./ubuntu.sif ls -l /workdir/

#A short cut of the previous command
./ubuntu.sif ls -l /workdir/

#The following command should throw an error because /programs directory is not
mounted
singularity exec ./ubuntu.sif ls -l /programs/bwa-0.7.8/bwa

#In the following command, you mount the /programs directory inside container and
run bwa from inside container
singularity exec --bind /programs ubun_d.sif /programs/bwa-0.7.8/bwa
```

1.4 Common options

- --no-home option: This option tells Singularity not to mount the home directory. This is desirable to fully contain the environment and not use use the Python and R libraries installed in your home directory. Compare the results with and without "--no-home option".

```
 cd /workdir/$USER

 singularity exec ./ubuntu.sif ls /home/$USER

 singularity exec --no-home ./ubuntu.sif ls /home/$USER
```

- --bind option: mount a host directory in the container. In this example, the host directory /workdir/$USER will be mounted as /data directory in the container. If with no ":", e.g. "--bind /workdir/$USER", the directory will be mounted as the same path inside container.

```
singularity exec --bind /workdir/$USER:/data ubuntu.sif ls /data
```

Part 2. Build Singularity images

2.1 Convert a Docker image to a Singularity image

Pre-built docker images are available for many bioinformatics software. In this exercise you will download the docker image built for the Busco software, and convert it to singularity image. Then run the busco command:

```
singularity pull busco.sif docker://quay.io/biocontainers/busco:5.2.2--
pyhdfd78af_0

./busco.sif busco -h
```

- "./busco.sif" is a shortcut for "singularity exec ./busco.sif"

2.2 Modify an existing singularity image file

In this exercise, you will modify the image file you downloaded in the first exercise "ubuntu.sif", and install "python" in this image.

First acquire the "fakeroot" privilege by running this BioHPC command. The "fakeroot" privilege is require to build a Singularity image.

```
fakeroot
```

Convert the ubuntu.sif file into a Singularity sandbox

```
cd /workdir/$USER

singularity build --fakeroot --sandbox myubuntu ./ubuntu.sif
```

Start the sandbox as container with a writable shell and install python 3.8

```
singularity shell --fakeroot --writable myubuntu

apt update

apt upgrade

apt install python3.8

ln -s /usr/bin/python3.8 /usr/bin/python

which python

python -V

exit
```

Convert the sandbox into a Singularity image file, and test the image file

```
singularity build --fakeroot myubuntu.sif myubuntu/

singularity exec myubuntu.sif python -V
```

In practice, it is recommended to use the sandbox for testing, and write the tested steps into a def file (a text file), and then build the new image from the script file. This way, you have a record how the image was created.

Create a def file "myubuntu.def"

```
BootStrap: library
From: ubuntu:20.04

%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
    apt -u install python3.8
    ln -s /usr/bin/python3.8 /usr/bin/python
```

Now run the command

```
singularity build --fakeroot myubuntu2.sif myubuntu.def
```