FUC Documentation

Release 0.1.0

Pingjun Chen

Jul 13, 2021

Cheatsheets

The documentation for FUC is mainly organized by following four sections.

- *Cheatsheets*
- *Inspections*
- *Commands*
- *Configs*
- *About*

CHAPTER 1

docker

Docker is an open platform for developers and sysadmins to build, ship, and run distributed applications, whether on laptops, data center VMs, or the cloud.

1.1 Docker Installation

```
$ sudo apt install docker.io          # install docker
$ sudo usermod -aG docker $USER       # add user to docker group
```

1.2 Image management

Create **image** from Dockerfile

```
$ docker build -t <image_name>:[tag_name] .
```

Create **image** from a specific Dockerfile

```
$ docker build -f <dockerfile_name> -t <image_name>:[tag_name] .
```

Image management

```
$ docker save <image_name>:[tag_name] <image_name>.tar      # save image
$ docker load --input <image_name>.tar                      # load image
$ docker images                                             # list available images
$ docker rmi <image_name>                                   # remove image by name
$ docker rmi <image_id>                                     # remove image by id
$ docker images purge                                       # remove dangling images
```

1.3 Container management

Start container

```
$ docker run -it --restart always --name <container_name> <image_name>:[tag_name]
Options:
    -v <local_dir>:<docker_dir>:ro            # map local directory to docker
    --runtime=nvidia                          # expose NVIDIA GPUs
    -e NVIDIA_VISIBLE_DEVICES=5               # cuda device setting
    --shm-size 16G                            # set size of shared memory
    --rm                                      # remove container file system when␣
↪exits
```

Copy files

```
$ docker cp <container_name>:<src_dir> <local_dst_dir>  # copy files from docker to␣
↪local
$ docker cp <local_src_dir> <container_name>:<dst_dir>  # copy files from local to␣
↪docker
```

Container management

```
$ docker ps                          # list all available containers
$ docker stop <container_name>       # stop specific container
$ docker rm <container_name>         # remove specific stopped container
```

1.4 Upload image to docker hub

Login to docker hub

```
$ export DOCKER_ID_USER="user_name"    # set docker hub username
$ docker login                         # login in to docker hub
```

Tag image

```
$ docker tag <image_name>:<version> $DOCKER_ID_USER/<image_name>:<version>
```

Push to docker cloud

```
$ docker push $DOCKER_ID_USER/<image_name>:<version>
```

1.5 Convert image to singularity

Create docker image tarball

```
$ docker save <image_name>:<version> -o <image_name>.tar
```

Build singularity from image tarball

```
$ singularity build <image_name>.sif docker-archive://<image_name>.tar
```

Build singularity from DockerHub image

```
$ singularity pull <image_name>.sif docker://<user_name>/<image_name>:<version>
```

1.6 Nvidia-Docker

Install nvidia-docker 2.0

```
$ curl -s -L https://nvidia.github.io/nvidia-docker/gpgkey | sudo apt-key add -
$ distribution=$(. /etc/os-release;echo $ID$VERSION_ID)
$ curl -s -L https://nvidia.github.io/nvidia-docker/$distribution/nvidia-docker.list␣
↪| \
    sudo tee /etc/apt/sources.list.d/nvidia-docker.list
$ sudo apt-get update
$ sudo apt-get install nvidia-docker2
$ sudo pkill -SIGHUP dockerd
$ docker run --runtime nvidia --rm nvidia/cuda:9.0-base-ubuntu16.04 nvidia-smi
```

CHAPTER 2

tmux

tmux is a terminal multiplexer: it enables a number of terminals to be created, accessed, and controlled from a single screen. tmux may be detached from a screen and continue running in the background, then later reattached.

2.1 Installation

```
$ sudo apt-get install tmux
```

2.2 List available sessions

```
$ tmux ls
```

2.3 Start new session

```
$ tmux new -s <session_name>
```

2.4 Attach to session

```
$ tmux a -t <session_name>
```

2.5 Kill session

```
$ tmux kill-session -t <session_name>
```

CHAPTER 3

sftp

SSH File Transfer Protocol, a network protocol used for secure file transfer over secure shell.

3.1 Login to server

Login to user home directory of the server.

```
$ sftp <user_name>@<server_ip>
```

Login to specific directory of the server.

```
$ sftp <user_name>@<server_ip>:<remote_dir>
```

3.2 File transfer

Download file from server.

```
$ get <server_path> <local_path>
```

Upload local file to server.

```
$ get <local_path> <server_path>
```

3.3 Command

Command for local host.

```
$ !<command>
```

Command for server.

```
$ <command>
```

3.4 Exit server

Exit from server.

```
$ bye
```

CHAPTER 4

sshfs

sshfs is network filesystem client to connect to other macines, which allows server to use code and data from local machine with no need for copying.

4.1 Installation

```
$ sudo apt-get install sshfs
```

4.2 Create connection

In server machine, link server directory to local directory. Note to avoid using soft link for both local and server directory.

```
$ sshfs -o nonempty <username>@<local_ip>:local_dir server_dir
```

4.3 List sshfs mounting points

```
$ ps -elf | grep sshfs
```

4.4 Unmount the connection

```
$ sudo umount server_dir
```

CHAPTER 5

seadragon

Seadragon is a supercomputing resource of MD Anderson's High Performance Computing (HPC), which is dedicated for use by the research community.

5.1 Login Seadragon

```
$ ssh seadragon
```

5.2 Interactive Running

```
# cpu interactive
$ busb -Is -q interactive -W 1:00 -M 64 -R rusage[mem=64] -n 4 /bin/bash
# gpu interactive
$ busb -Is -q gpu-medium -gpu num=1:gmem=16 -W 3:00 -M 64 -R rusage[mem=64] -n 10 /
↪bin/bash
```

5.3 Load Modules

```
$ module load singularity/3.5.2
$ module load cuda10.1/toolkit/10.1.243
```

5.4 Start Singularity Instance

```
$ singularity run --nv --bind <local_dir>:<container_dir> <singularity_path> <program>
```

5.5 More Useful Commands

```
$ bsub < <lsf_script> # submit job via script
$ bjobs -u all | more # check all running jobs
$ bjobs -u all | grep gpu # check all gpu jobs
$ bjobs -p # show all pending jobs
$ bjobs -l xxxxxxx # check the details of one specific job
$ bkill -l xxxxxxx # kill one specific job
```

CHAPTER 6

virtualenv

6.1 Installation

```
$ pip install virtualenv
```

6.2 Create virtual environment

Create with specific python version

```
$ virtualenv -p python3 myvenv
```

6.3 Activate environment

```
$ source myvenv/bin/activate
```

6.4 Deactivate the environment

```
$ deactivate
```

CHAPTER 7

conda env

7.1 List available envs

```
$ conda info --envs
```

7.2 Create env

```
$ conda create --name <env_name> python=3.6
```

7.3 Activate env

```
$ source activate <env_name>
```

7.4 Deactivate env

```
$ source deactivate
```

7.5 Remove env

```
$ conda remove --name <env_name> --all
```

CHAPTER 8

system info

8.1 Check Ubuntu version

```
$ lsb_release -a
```

CHAPTER 9

filesystem

9.1 Get number of all files

When number of file is small, less than 10, 000

```
$ ls -ls *.<ext> | wc -l
```

When more than 10, 000 files

```
$ find -type f -name '*.<ext>' | wc -l
```

9.2 Storage usage

Check disk partition usage

```
$ df -h
```

Check the size of a directory

```
$ du -hs <directory>
```

9.3 Check size of all subdirectories

```
$ du -hs *
```

CHAPTER 10

git command

10.1 Delete branch

Delete local branch

```
$ git branch -D <branch_name>
```

Delete remote branch

```
$ git push <remote_name> --delete <branch_name>
```

10.2 Tag

Add tag to current submit

```
$ git tag -a <tag_name> -m <comment>
```

List all tags

```
$ git tag -l
```

Delete local tag

```
$ git tag -d <tag_name>
```

Delete remote tag

```
$ git push --delete origin <tag_name>
```

10.3 Remote

Set up upstream remote

```
git remote add upstream https://github.com/repo
```

Update local repo

```
git pull upstream <branch_name>
```

Push to upstream

```
git push upstream <branch_name>
```

CHAPTER 11

user management

11.1 Add user with full profile

```
$ adduser <username>
```

11.2 Add user with only name

```
$ useradd <username>
```

11.3 Add user to sudo group

```
$ usermod -aG sudo <username>
```

11.4 List all user names

```
$ compgen -u
```

Delete user

```
$ userdel -r <username>
```

CHAPTER 12

process management

12.1 Check process

View your system's resource usage and see the processes that are taking up the most system resources.

```
$ top
```

Improved top

```
$ htop
```

12.2 Kill process

By process id

```
$ kill -9 <pid>
```

By application name

```
$pkill <app>
```

By filtering conditions

```
ps -ef | grep <command> | awk '{print $<col_num>}' | xargs kill -9
```

CHAPTER 13

compression

13.1 Tar and untar

Create tar archive File

```
$ tar -cvf tar-archive-name.tar <source>
```

Untar files in current directory

```
$ tar -xvf tar-archive-name.tar
```

Untar files to a specific directory

```
$ tar -xvf tar-archive-name.tar -C <destination>
```

List content of tar archive file

```
$ tar -tvf tar-archive-name.tar
```

13.2 Compress and uncompress

Compress folder to tar.gz

```
$ tar -czvf tar-archive-name.tar.gz <source>
```

Extract a tar.gz compressed archive

```
$ tar -xzvf tar-archive-name.tar.gz
```

Compress folder to tar.bz2

```
$ tar -cjvf tar-archive-name.tar.bz2 <source>
```

Extract a tar.bz2 compressed archive

```
$ tar -xjvf tar-archive-name.tar.bz2
```

CHAPTER 14

apache

14.1 Restart

```
$ service apache2 restart
```

14.2 Check error log

```
$ tail -n 20 /var/log/apache2/error.log
```

CHAPTER 15

git config

15.1 Git configuration

User name

```
$ git config --global user.name PingjunChen
```

User email

```
$ git config --global user.email pingjunchen@ieee.org
```

Editor

```
$ git config --global core.editor vim
```

Password

```
$ git config --global credential.helper cache
```

Default branch

```
$ git config --global init.defaultBranch main
```

Add remote

```
$ git remote add origin https://github.com/user/repo.git
```

15.2 Git alias

```
git config --global alias.cm "commit -m"
git config --global alias.st status
git config --global alias.ci commit
git config --global alias.co checkout
git config --global alias.br branch
git config --global alias.last "log -1 HEAD"
```

(continues on next page)

(continued from previous page)

```
git config --global alias.aa "add --all"
git config --global alias.cb "checkout -b"
```

CHAPTER 16

ssh no password

16.1 Login to server without password

Step 1. Generate authentication keys in local machine.

```
$ ssh-keygen -t rsa
```

Step 2. Create ~/.ssh directory in server if not exist.

```
$ mkdir -p .ssh
```

Step 3. Append public key in local machine to server.

```
$ cat .ssh/id_rsa.pub | ssh <username>@<server_ip> 'cat >> .ssh/
authorized_keys'
```

Step 4. Login to server.

```
$ ssh <username>@<server_ip>
```

CHAPTER 17

.bashrc

The purpose of a .bashrc file is to provide a place where you can set up variables, functions and aliases, define your (PS1) prompt and define other settings.

17.1 Set environment variable

Set a new variable, add a new line at the end of .bashrc and add

```
export VARNAME="var_value"
```

Add new value to a existing environment variable, such as **PATH**

```
export PATH="new_value":${PATH}
```

Or define a new variable first, then use the new variable.

```
export VARNAME="var_value"
export PATH=${VARNAME}:${PATH}
```

17.2 Set alias

alias examples

```
alias apt-get='sudo apt-get'
alias ls='ls --color=auto'
```

17.3 Take effect

Make changes in .bashrc immediately effective.

```
$ source ~/.bashrc
```

CHAPTER 18

fstab

The configuration file "/etc/fstab" contains the necessary information to automate the process of mounting partitions.

18.1 Mount partitions

Add following line to the end of **fstab** file.

```
<partition> <mount_point> auto auto,user,rw 0 0
```

Following are two examples:

```
/dev/sdc1 /data/.data1 auto auto,user,rw 0 0
/dev/sdd1 /data/.data2 auto auto,user,rw 0 0
```

18.2 Join partitions

mhddfs: tool for joining several filesystems together to form a single larger one.

Install mhddfs

```
$ sudo apt-get install mhddfs
```

Create folder for merging

```
mkdir /data/main
```

Add the following line to the end of fstab

```
mhddfs#/data/.data1,/data/.data2 /data/main fuse defaults,allow_other 0 0
```

CHAPTER 19

.gitignore

https://github.com/github/gitignore A collection of useful .gitignore templates

19.1 Python .gitignore example

```
# File

# Folder

# File List
.DS_Store
*.manifest
MANIFEST
.remote-sync.json

# Folder List
__pycache__/
data/
build/
dist/
venv/
env/
.ipynb_checkpoints/

# File with specific suffix
*.so
*.py[cod]
```

CHAPTER 20

python header

20.1 Example

```
# -*- coding: utf-8 -*-

__author__ = "Pingjun Chen"
__email__  = "chenpingjun@gmx.com"
__date__   = '2018/08/28'

import os, sys
```

CHAPTER 21

about FUC

Frequently Used Commands (FUC) is a collection of most commonly used tools in routine programming under Ubuntu or Mac OS. The purpose is to gather those easy-to-forget commands. Since I always forget and need to Google around, why not organize them together?

If you find or think of any commands or tools that would be helpful for routine work, please issue or pull request.