

Linux Basics and Efficient Tools

Yujiang Bi

CC@IHEP, CAS

The 5th Computing Summer School for High Energy Physics

08/22/2024

Outline

- Linux Intro
- Linux Environment
- Text Editors
- Remote Access Tools
- Git & IHEP Code
- Container Technology

Linux Intro



Welcome to the World of Linux



Linux? GNU/Linux?

- Linux is the open-sourced kernel of GNU OS, created by Linus Torvalds in 1991, providing
 - Management for processes, memory and file system ...
 - Devices drivers, networking, security and interprocess communication ...
- GNU OS was created by Richard Stallman to be a free replacement for Unix
 - It has an official kernel called GNU Mach, the kernel of GNU Hurd
- GNU/Linux Distribution
 - A complete operation system with different components, like RHEL and Debian
 - One distribution might be quite different from another one in UE and management
- GNU/Linux is the most successful open-sourced OS

The Most Successful Open-sourced OS

- Advantages and applications of GNU/Linux
 - Open-sourced and free to everyone. Flexible and
 - Run on hardwares of various architectures and used in different areas
 - Data center, network infrastructure, mobile and embedded devices
 - Cloud computing, supercomputing, scientific computing ...
- Comparision with other OSs like Windows and macOS
 - More open! More free! More flexible!
 - Users can freely choose and customize own distributions
 - More stable and secure than other operating systems

Linux Distributions



An Old Linux Global Map



What's Your Favorite Distribution?

Popular Linux distributions:

- Debian
 - **Ubuntu**, Kubuntu, Linux Mint, Xubuntu, **Armbian** ...
- RedHat
 - CentOS, **Fedora**, **AlmaLinux**, Rocky, Oracle Linux ...
- ArchLinux
 - Manjaro, Artix, Chakra, Endeavour ...
- OpenSUSE
 - Gecko, Kamarada ...

Linux Environment

A First Look at Linux

```
bash-3.2$ ssh lxlogin
*****
* Welcome to lxlogin003.ihep.ac.cn, AlmaLinux release 9.4 (Seafoam Ocelot)
* User Manual: http://afsapply.ihep.ac.cn/cchelp/
*****
Last login: Thu Aug 15 22:37:00 2024 from 10.100.0.116
(biyj@lxlogin003)-(0)-(10:37 PM Thu Aug 15)
[~]-|> █
```

Who Am I? Where Am I?

- Username: the passport to Linux world
 - Username with uid is unique to identify who you are
 - Using correct username & password to log into linux system and entering your home.
- Group: a logical collection of users with the same characteristics
 - To have same permissions to specific dir/files, like view or edit some file or directory
 - One can belong to multiple groups, but only have one primary group
- `whoami`: display who your are
- `pwd`: display where you are

```
$ whoami  
biyj
```

```
$ pwd [-P]  
/afs/ihep.ac.cn/users/b/biyj
```


- `id`: identity yourself or other user

```
$ id
uid=12142(biyj) gid=600(u07)
groups=600(u07),340(lhaasorun),580(lhaaso),1027(lqcd),1055(qc)...
$ id lihaibo
uid=10515(lihaibo) gid=600(u07)
groups=600(u07),290(physics),580(lhaaso),470(offlinerun)...
```

- `passwd`: change your password (or other accounts). Change from Login page for IHEP cluster

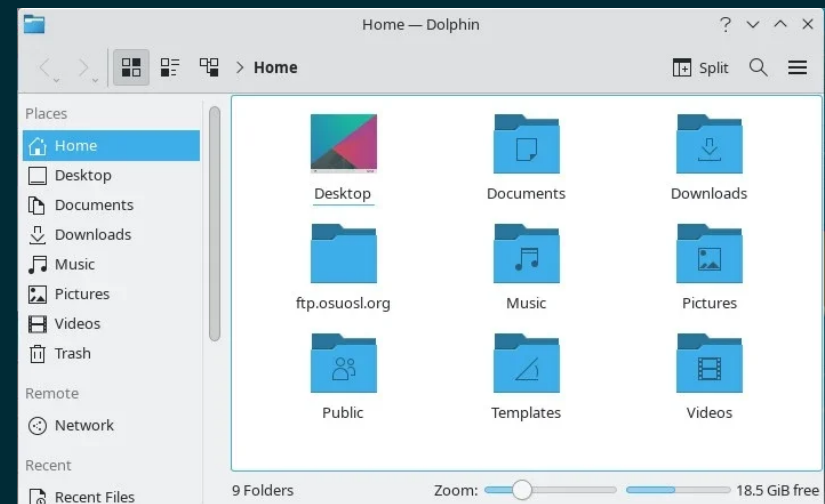
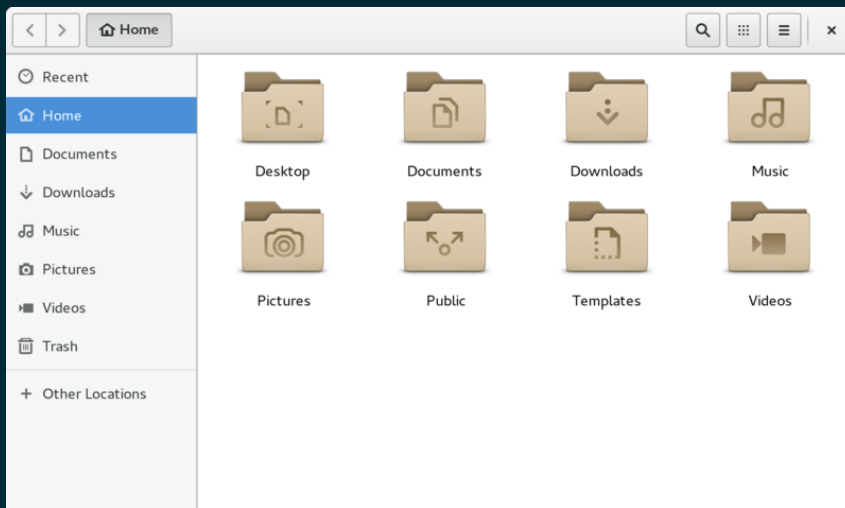
```
$ passwd
Changing password for user biyj.
Current password:
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
```

Where is my C Drive?

If you user headless environment:

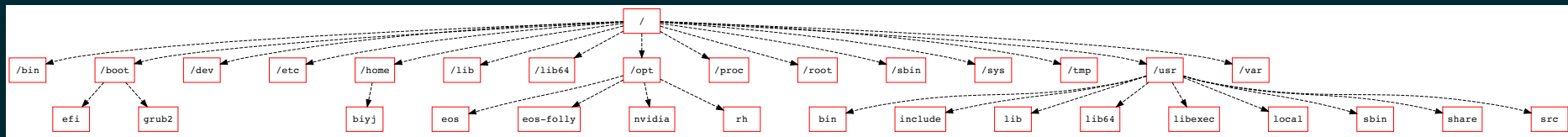
```
1 [biyj@n200 ~]$ ls
2 Desktop Documents Downloads Music Pictures Public Templates Videos
3 [biyj@n200 ~]$ ls /
4 bin dev home lib64 mnt proc run srv tmp var
5 boot etc lib media opt root sbin sys usr
```

If you use DE:



Linux Filesystem - Hierarchy

- Linux filesystems are organized in a tree structure with the root directory (/) at the top
- Modern Linux Filesystem have some typical directories, and part of the directory tree looks like:



- / : the root of whole system
- /bin and /sbin
 - Locations of system executable files
 - Normally links of /usr/bin and /usr/sbin
- /home and /root
 - Home directory of normal users and root
- /lib and /lib64
 - Locations of shared and static libraries
 - Normally links of /usr/lib and /usr/lib64
- /opt: locations of 3-party softwares
- /tmp: locations of temporary files and dirs
- /dev: location of devices like disk, cpu, and ram
- /etc: system management and config files
- /proc: virtual filesystem storing system infos
- /sys: virtual filesystem, an interface to the kernel.
- /var: storing variable files like logs
- Other storage directories:
 - /afs: afs mount point
 - /cvmfs: containing collections of various softwares
 - /eos: mount point of EOS filesystem
 - /hpcfs, /junofs, /ihepfs ...

Everything is A FILE

- In Linux everything is considered a **FILE**
 - Including hardware devices, processes, directories, regular files, sockets, links...
 - Many kinds of file types, like regular file, directory, block, links ...
 - Each type of file has a specific purpose and properties
- Aliases for directories
 - `/`: the root directory
 - `.`: the current directory
 - `..`: the parent of current directory
 - `~`: the home of current user, like `$HOME`
 - `~jack`: the home of user jack
 - `-`: the the last directory you visited

Linux Filesystem - Directory and Links

- Entering a directory

```
$ cd ~; cd $HOME; cd      # return to HOME
$ cd ..                  # go to parent dir
$ cd /path/to/dir       # enter other dir
$ cd -                   # return previous dir
```

- List a directory

```
$ ls dirname           # list files under dirname
$ ls -l dirname       # list files in detail
total 4
-rw-r--r-- 1 biyj u07 2 Aug 22 15:54 testfile
$ ls -lS dirname     # list files by size descendent
total 8
-rw-r--r-- 1 biyj u07 8 Aug 22 16:12 a.txt
-rw-r--r-- 1 biyj u07 2 Aug 22 16:18 d.txt
$ ls -ld dirname    # show info of dirname
$ ls -A dirname     # list files including hidden
.hidden testfile
```

- Two kinds of links in linux
 - Hard link and soft link (or symbolic link)
 - Using command `ln` to create links
- Hard link
 - Another name to linked file with the same inode
 - Must within the same filesystem
 - Valid even if the targeted file deleted
- Soft link
 - A special file containing the path of linked file/dir
 - Will break if targeted file/dir deleted

- Example

- Creating a directory

```
$ mkdir dir # create a blank dir under curre  
$ mkdir -p dir1/dir2 # recursively create dir
```

- Delete a directory

```
$ rmdir dir # rm an empty dir  
$ rmkdir -rf dir # rm a dir and files/dirs in
```

```
$ ln a.txt b.txt # hard link  
$ ln -sf a.txt c.txt # soft link  
$ ls -l  
total 8  
-rw-r--r-- 2 biyj u07 8 Aug 22 16:12 a.txt  
-rw-r--r-- 2 biyj u07 8 Aug 22 16:12 b.txt  
lrwxrwxrwx 1 biyj u07 5 Aug 22 16:13 c.txt
```

Linux Filesystem - File Operations

Linux Filesystem - File Operations

- cp: copy files or dirs (with -r) to another place

- Normal copy

```
$ cp a.txt b.txt # copy a file
$ cp -r a.txt c/ d/ # copy a.txt and c/ in
```

- Link mode

```
$ cp -l a.txt h.txt # hard link with -l
$ cp -s a.txt s.txt # soft link with -s
```

- Permissions

```
$ cp -p a.txt p.txt # mode,ownership,timest
$ cp --preserve=all a.txt f.txt # all attri
```

- Archive mode

```
# same as -dR --preserve=all
$ cp -a a.txt a.archive.txt
# only attributes, don't copy contents
$ cp --attributes-only a.txt c.txt
$ cp -ar src/ dst/ # archive a directory
```

- mv: move or rename files or dirs

- Normal mode

```
# move file/dir to dir
$ mv a.txt b/ c/
```

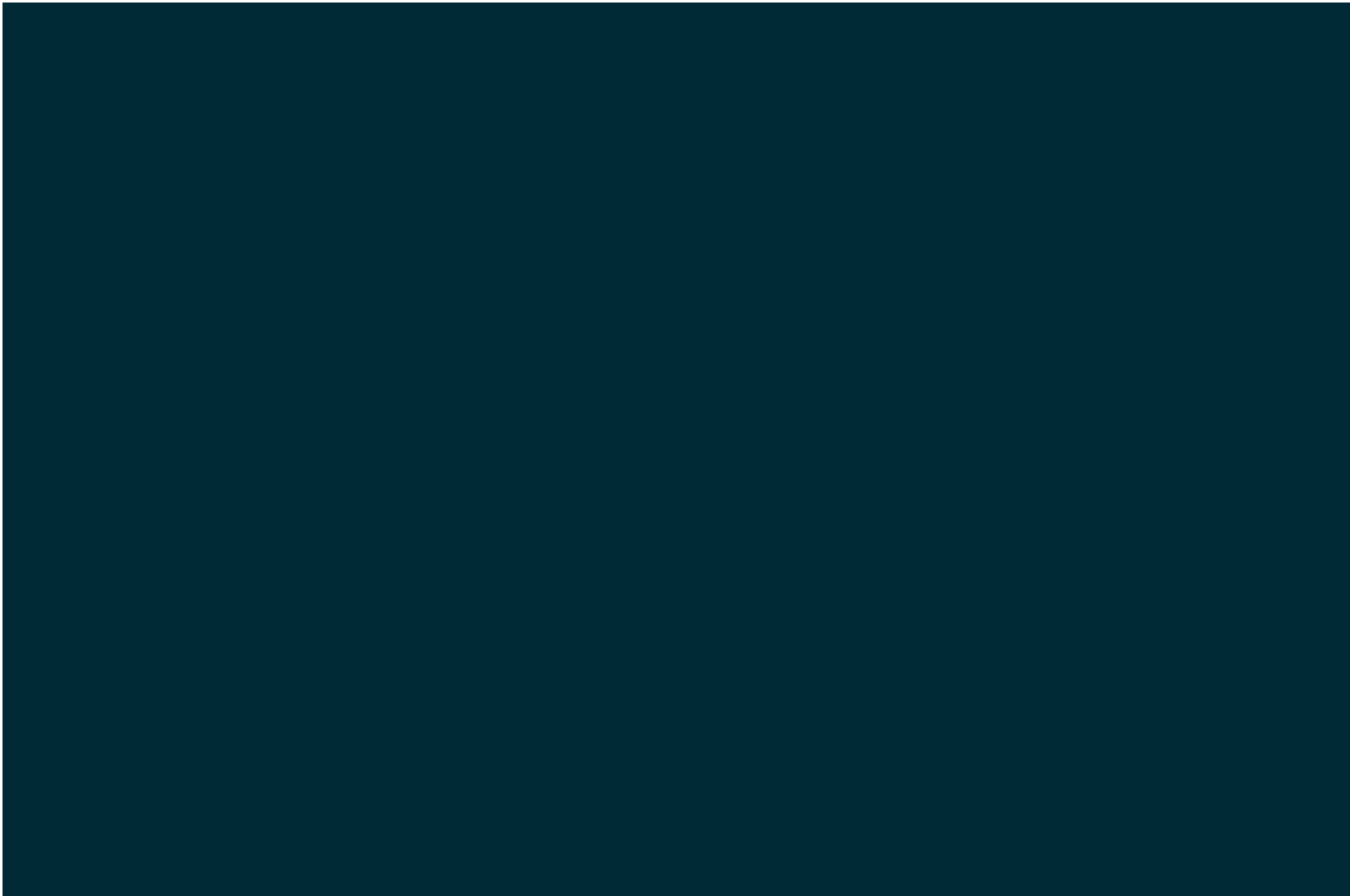
- Rename mode

```
# rename a file
$ mv a.txt b.txt
# rename dir, dir must not exist
$ mv b non-exist-dir
```

- Backup mode: create a backup if the target exist

```
$ ls
a.txt b.txt c.txt d.txt
$ mv -b a.txt b.txt; ls # simple with a end
b.txt b.txt~ c.txt d.txt
$ mv --backup=t b.txt c.txt; ls # numbered
b.txt~ c.txt c.txt.~1~ d.txt
$ mv --backup=nil c.txt d.txt; ls # follow
b.txt~ c.txt.~1~ d.txt d.txt~
```

- Backup mode: create a backup if the



Linux Filesystem - find

- Search a directory for specific files
 - Usage: `find [path] [options] [expression]`
 - Default path is current directory if omitted

- Find files but dirs starting with "test"

```
$ find . -type f -name "test*"
test.1.txt
```

- Find files created or modified within 10 days

```
$ find . -type f \( -mtime -10 -o -ctime -10 \)
ctime.within.10.txt
mtime.within.10.txt
```

- Find files created or modified 10 days ago

```
$ find . -type f \( -mtime +10 -o +ctime +10 \)
ctime.older.10.txt
```

- Search specific files and execute operations
 - Using `find` with `-exec` option
 - Using `{}` to replace files found

- Find and delete empty file

```
$ find . -type f -empty -exec rm -f {} \;
```

- Find and truncate files larger than 1G

```
$ find . -type f -size +1G -exec echo -n {}
```

- Find and delete files older than 1 year

```
$ find . -type f -mtime +365 -exec rm -f {}
$ find . -type f -mtime +365 -delete
```

- Find files owned by `biyj` and change permission

```
$ find . -type f -user biyj -exec chmod +x {}
```

Linux Filesystem - Permissions

- Type of permissions
 - Three main permission types: `read(r,4)`, `write(w,2)` and `excution(x,1)`
 - Each permission can be assigned to the owner and the group it belongs to, and to others
- Special permissions:
 - SUID
 - A file with SUID always executes as the user who owns the file, like `passwd`

```
$ ls -l /usr/bin/passwd
-rwsr-xr-x. 1 root root 32656 Apr 14 2022 /usr/bin/passwd
```

- SGID
 - For a file, it allows the file to be executed as the group that owns the file
 - For a directory, any files created in the directory have same group ownership with directory owner
- Sticky bit
 - Only the owner (and root) of a file can remove the file within that director
 - A common example of this is the `/tmp` directory:

Special Characters in Shell

- `$`: indicates the beginning of a variable, like `$PATH`
- `*`: wildcards for any character (including space)
- `|`: PIPE. Taking the output of command before `|` as the input of command after `|`
- `>`: redirecting command output to a file. If the file already exists, it will be overwritten
- `>>`: redirect command output to a file or appended to the end of the file if existing
- `<`: taking the contents of a file as input to a command
- `\`: escaping characters so that they lose their special meaning
- `;`: command separator. Separating multiple commands in one line

Process and Process Management

- What is a Process?
 - An instance of a program that is currently running on a computer
 - It is the basic unit for resource allocation and task execution in a computer
 - Each process has its own memory space, execution context, and state.
- Process Identification
 - Each process has a unique Process ID (PID) and Parent PID (PPID)
 - Using `ps` and other commands to view process infos

```
$ ps aux | grep process
```

- `kill` can be used to send signals to processes. Common signals include:
 - SIGTERM: Terminate the process.
 - SIGKILL: Forcefully terminate the process.
 - SIGSTOP: Pause the process.
- Running Processes in the Background
 - Background Execution with ``&``. Process will be killed if terminal is closed

```
$ my_command > my.log 2>&1 &
```

- Using `nohup`

```
$ nohup my_command > my.log 2>&1
```


Text Editors

Classical Editors

VIM

usable in just about any environment.

does one thing, well.



EMACS

flexible, customizable, and packed with every feature known to man.



NANO

mostly used by people who do not know what they are doing; or psychopaths.



© 2015 CURTIS LASSAM - CUBE-DRONE.COM

The Holy War of Editors



vs



Vim Cheatsheet

- Change Leader

```
let g:mapleader=";"
let mapleader=";"
```

- View a file in RO mode:

```
$ view f1 f2 ...
```

- Diff mode

```
$ vimdiff f1 f2
```

- Split mode

```
$ vim -O f1 f2 ...
$ vim -o f1 f2 ...
```

Invocation

Legend

vim - Read from STDIN **a** Key press
 vim +90 Jump to line **a** ^a, <C-a>, **Ctrl** + a
 vim -u NONE "Debug" mode **a** Hold shift to modify usage

sheet designed by Max Centor thingsfittogether.com

Registers
Normal Mode **i s a o c**
Insert Mode

d Cut line to default register

p Paste from default register

" a p Paste from

" a d d Cut line to

" E 3 y Append 3 lines to

" Register

a Append

a Append

e Append

d Save & Close

y Copy

p Paste

g f Open file under cursor

: Command Mode

Z Z Save & Close

Z Q Close

o Swap files

***** Find word under cursor

v Visual Mode

/ Search

g g Word count

o Jump back

v Visual Mode

n Next

f File

l Line

t Tag

Esc or **c** or **[** Return to Normal Mode

x Complete...

n Autocomplete

n Next **p** Previous

y Confirm **e** Cancel

Options
Command Mode
Manual

:set wrap lbr Notepad "mode"

:set ft? Check filetype

:set list et Spaces only

:set ru nu sc Inspect "mode"

:set sw=4 sts=4 Tab-to-space ratio

:set tw& Reset textwidth

:set nohls Disable search highlighting

Enter Submit command

:quit Exit Vim

:write Save file

:pwd Current directory

:ls List open files

:cd Change directory

:set Set options

f History

:edit Open file

:vsplit Vertical

:close Close split

:bn Go to file *n*

:help Learn Vim

:reg List all registers

] Drill down **t** Bubble up

:help navigation **:help :s** Search & replace

:help options **:help ^w** **w**indow commands

:help windows **:help map** Macros & keys

:help motion **:help i^w** Insert mode **w**?

:help registers **:help word-motions**

:help cmdline **:help pattern-searches**

Emacs - the God's Editor

Modern Editor

- VS Code
- Sublime
- Brackets
- PyCharm, JetBrains
- Cursor with AI
- Zed with AI

Regular Expression Tools

Sed

A stream editor used for parsing and transforming text from input stream.

- **Substitute Text**

```
$ sed -i 's/old/new/' filename # one line
$ sed -i 's/old/new/g' filename # all lines
```

- **Delete lines**

```
$ sed -i '3d' filename # del 3rd line
$ sed -i '2,10d' filename # del line 3 to 10
```

- **Insert lines**

```
# Insert before 3rd line
$ sed -i '3 i\new_line' filename
```

- **Print Specific Lines**

```
# Print only lines matching a pattern
$ sed -n '/pattern/p' filename
# Print a specific line
$ sed -n 'Np' filename
```

- **Using RE**

```
$ sed 's/[0-9]/#/g' example.txt
```

- **Multiple Commands**

```
$ sed -e "command1" -e "command2" filename
```

Grep

A powerful text search tool used to search for matching lines in a file or input stream

- Search pattern

```
$ grep 'pattern' filename
$ grep -n 'pattern' filename # with line number
$ grep -i 'pattern' filename # ignore case
$ grep -R 'pattern' dir # recursive search
```

- Search for Whole Words

```
$ grep -w 'world' filename
```

- Invert Match

```
$ grep -v 'pattern' filename
```

- Count matched lines

```
$ grep -c 'pattern' filename
```

- Show Only Matching Part

```
$ grep -o 'pattern' filename
```

- Using RE

```
$ grep -E 'foo|bar' filename
```

- Search Lines starting or ending with pattern

```
$ grep '^pattern' filename # beginning
$ grep 'pattern$' filename # ending
```

- Search multiple patterns

```
$ grep -e 'pattern1' -e 'pattern2' filename
```

- Show Context Lines

```
$ grep -C 5 'pattern' filename
$ grep -A 10 'pattern' filename
$ grep -B 20 'pattern' filename
```

- Suppress Filename in Output

```
$ grep -h 'pattern' file1 file2 ...
```

Awk

A powerful text processing tool and programming language for data extraction and reporting

- Print Specific Column like 4th col

```
$ awk '{print $4}'
```

- Print specific column of matched line

```
$ awk '/pattern/ {print $3}' filename
```

- Specifica field separator like "_"

```
$ awk -F _ '{print $0}'
```

- Built-in Variables

- NR: line number
- NF: Number of fields

```
$ awk '{print NR, NF}' filename
```

- Calculations

```
$ awk '{print $1 + $3}' filename
```

- Conditional Statements

```
$ awk '{if ($2 > 100) print $0}' filename
```

- BEGIN and END Blocks to initialize and

```
$ grep '^pattern' filename # beginning
```

- Sum a Column

```
$ awk '{sum += $3} END {print sum}' filename
```

- Print Specific Lines

```
$ awk 'NR>=3' filename
```

- Use External Variables

```
$ awk -v var=value '{print var, $0}' filename
```

- Replace a Column

```
$ awk '{$2="new_value"; print $0}' filename
```

Remote Access Tools

Remote Secure Shell Tools

- WSL
 - Windows 10/11 required, with WSLg installed by default
- XShell/XManager
 - 7 required for lxlogin.
- MobaXterm
 - Homeedition is good enough. But no IPv6 for X11
- Tabby
 - A morden terminal simulator
- Putty
- Solar Putty
- ...

Default SSH Config is not User-friendly

Log in to remote server with password:

```
1 $ ssh myuser@myhost
2 myuser@myhost's password:
3 Permission denied, please try again.
4 myuser@myhost's password:
5 Permission denied, please try again.
6 myuser@myhost's password:
7 myuser@myhost: Permission denied (publickey,gssapi-keyex,gssapi-with-mic,password)
```

Avoid password with SSH key pair

```
1 $ -f ~/.ssh/id_rsa -t rsa -b 4096
2 Generating public/private rsa key pair.
3 Enter passphrase (empty for no passphrase):
4 Enter same passphrase again:
5 Your identification has been saved in /root/.ssh/id_rsa
6 Your public key has been saved in /root/.ssh/id_rsa.pub
7 The key fingerprint is:
8 SHA256:9B+5DBkw/dYGYt06/gbz17IJBSwkgPvgwb3n1OYAvUk root@myserver
9 The key's randomart image is:
10 .....
```

Custom Your SSH

Create or modify `~/.ssh/config`

Custom Your SSH

Create or modify `~/.ssh/config`

```
# $HOME/.ssh/config
Host *
    PubkeyAcceptedKeyTypes +ssh-rsa,ssh-ed25519
    ServerAliveInterval 60
    StrictHostKeyChecking no
Host lxlogin*
    Hostname %h.ihep.ac.cn
    User biyj
    ForwardX11 yes
Host *
    IdentityFile ~/.ssh/keys/id_rsa
    AddressFamily inet
```


Custom Your SSH

Create or modify `~/.ssh/config`

```
# $HOME/.ssh/config
Host *
    PubkeyAcceptedKeyTypes +ssh-rsa,ssh-ed25519
    ServerAliveInterval 60
    StrictHostKeyChecking no
Host lxlogin*
    Hostname %h.ihep.ac.cn
    User biyj
    ForwardX11 yes
Host *
    Identityfile ~/.ssh/keys/id_rsa
    AddressFamily inet
```

And log into login servers like:

Custom Your SSH

Create or modify `~/.ssh/config`

```
# $HOME/.ssh/config
Host *
    PubkeyAcceptedKeyTypes +ssh-rsa,ssh-ed25519
    ServerAliveInterval 60
    StrictHostKeyChecking no
Host lxlogin*
    Hostname %h.ihep.ac.cn
    User biyj
    ForwardX11 yes
Host *
    Identityfile ~/.ssh/keys/id_rsa
    AddressFamily inet
```

And log into login servers like:

```
1 | $ ssh lxlogin
2 | .....
3 | Last login: Sun Aug 18 07:53:08 2024 from 10.100.0.75
4 | [biyj@lxlogin003 ~]$
```

Terminal Multiplexer

People always say:

Terminal Multiplexer

People always say:

More Windows! More Terminals! More Desktops!

Terminal Multiplexer

People always say:

More Windows! More Terminals! More Desktops!

Terminal multiplexers give you

Terminal Multiplexer

People always say:

More Windows! More Terminals! More Desktops!

Terminal multiplexers give you

- Persistent sessions that can be attached and detached
 - regardless of unstable connection or disconnection
 - long-time job can safely run in stable without interruption
- Multiple panes and windows in one terminal instead of multiple terminals
- Collaborative work by attaching an existing session as they see fit

Terminal Multiplexer

People always say:

More Windows! More Terminals! More Desktops!

Terminal multiplexers give you

- Persistent sessions that can be attached and detached
 - regardless of unstable connection or disconnection
 - long-time job can safely run in stable without interruption
- Multiple panes and windows in one terminal instead of multiple terminals
- Collaborative work by attaching an existing session as they see fit

Popular Terminal multiplexers

- Tmux
- Screen
- Byobu
- Terminator
- Konsole
- dvtm

Tmux - Basic Usage

- Create window:

<CTRL-b> c

- Split window vertically:

<C-b> %

- Split window horizontally:

<C-b> "

- Move between panes:

\$ <C-b> ↑|↓|←|→

- Switch between windows

<C-b> p # previous

<C-b> n # next

- Rename session

<C-b> \$ test

- Detach a session

<C-b> d

The screenshot displays a Tmux terminal window with several panes. The top pane shows system statistics including memory usage, network activity, and system time. The middle pane shows CPU usage for the N200 server, including temperature and load averages. The bottom-left pane shows disk usage for the root, boot, and home directories. The bottom-right pane shows a process list with columns for PID, Program, User, MemB, and Cpu%. The terminal prompt is [0] 0:bash* and the session name is n200.ihep.ac.cn.

used	free	buf	cach	recv	send	read	writ	time
823M	13G	136M	919M	1889B	8391B	0	0	122-08 18:31:36
823M	13G	136M	919M	396B	1776B	0	0	122-08 18:31:37
823M	13G	136M	919M	2400B	8438B	0	4096B	122-08 18:31:38
823M	13G	136M	919M	1760B	628B	0	0	122-08 18:31:39
823M	13G	136M	919M	756B	9918B	0	0	122-08 18:31:40
823M	13G	136M	919M	2939B	604B	0	0	122-08 18:31:41
823M	13G	136M	919M	522B	8438B	0	0	122-08 18:31:42
823M	13G	136M	919M	192B	604B	0	0	122-08 18:31:43
823M	13G	136M	919M	1840B	8422B	0	0	122-08 18:31:44
823M	13G	136M	919M	4534B	604B	0	0	122-08 18:31:45
823M	13G	136M	919M	3663B	8470B	0	0	122-08 18:31:46
823M	13G	136M	919M	1909B	604B	0	0	122-08 18:31:47
823M	13G	136M	919M	642B	8454B	0	0	122-08 18:31:48
823M	13G	136M	919M	192B	604B	0	0	122-08 18:31:49
823M	13G	136M	919M	931B	8434B	0	0	122-08 18:31:50
823M	13G	136M	919M	1992B	604B	0	0	122-08 18:31:51
823M	13G	136M	919M	2886B	8496B	0	0	122-08 18:31:52
823M	13G	136M	919M	1700B	604B	0	0	122-08 18:31:53
823M	13G	136M	919M	2572B	8446B	0	0	122-08 18:31:54
823M	13G	136M	919M	312B	604B	0	0	122-08 18:31:55

```
cpu menu preset * 18:31:55 - 2000ms + 1.0 GHz
CPU 0% 47°C 0% 51°C
C0 0% 47°C C2 0%
C1 1% 47°C C3 1%
LAV: 0.01 0.03 0.00

up 3d 09:34

Total: 15.1 GiB
U 1.09 Gi
A 14.0 Gi
C 918 MiB
F 13.4 Gi

disks io
root 195G
boot 973M
home 717G

proc filter tree < cpu lazy >
Pid: Program: User: MemB Cpu%
8037 btop: root 6.3M 0.1
8593 python3: root 25M 0.0
7926 tmux: serv root 3.9M 0.0
7883 sshd: root 6.5M 0.0
914 gnome-shel gdm 183M 0.0
789 NetworkMan root 23M 0.0
7994 bash: root 5.5M 0.0
7927 bash: root 5.4M 0.0
690 clash: root 15M 0.0
8620 kworker/2: root 0B 0.0
7849 kworker/1: root 0B 0.0
7663 kworker/2: root 0B 0.0
7879 kworker/1: root 0B 0.0

net auto zero <b enpls0 n>
download
2.10 KiB/s
upload
4.43 KiB/s

[0] 0:bash* "n200.ihep.ac.cn" 18:31 22-Aug-24
```


Tmux Cheatsheet

- Change prefix:

```
unbind ^b
set -g prefix ^a
```

- Set VI mode:

```
setw -g mode-keys vi
```

- Bind key:

```
bind k selectp -U
bind j selectp -D
bind h selectp -L
bind l selectp -R
```

- List sessions:

```
$ tmux ls
```

- Attach session:

```
$ tmux a[tt] [-t id]
```

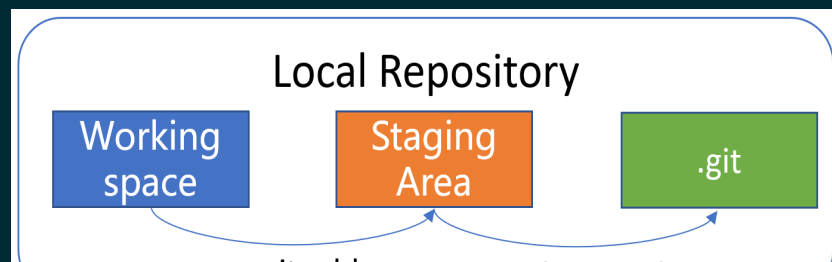
- Remember where your sessions are

[tmux session]	[tmux windows]
<p>_ new sessions</p> <pre>tmux tmux new tmux new-session tmux new -s sessionname</pre> <p>_ attach sessions</p> <pre>tmux a tmux att tmux attach tmux a -t sessionname</pre>	<p>_ remove sessions</p> <pre>tmux kill-ses tmux kill-session -t sessionname</pre> <p>_ key bindings</p> <pre>ctrl+B \$ rename session ctrl+B D detach session ctrl+B) next session ctrl+B (previous session</pre> <p>_ windows are like tabs in a browser. Windows exist in session and occupy the same of a session screen.</p> <p>_ key bindings</p> <pre>ctrl+B 0-9 select window by number ctrl+B ' select window by name ctrl+B . change window number ctrl+B , rename window ctrl+B C create window ctrl+B N move to next window ctrl+B P move to previous window ctrl+B L move to last used window ctrl+B F search windows ctrl+B & kill window ctrl+B W list windows</pre>
[tmux panes]	[tmux copy mode] EMACS MODE - DEFAULT
<p>_ panes are sections of windows that have been split into different screens - just like the panes on this cheatsheet.</p> <p>_ key bindings</p> <pre>ctrl+B % vertical split ctrl+B " horizontal split ctrl+B → move to pane to the right ctrl+B ← move to pane to the left ctrl+B ↑ move up to pane ctrl+B ↓ move down to pane ctrl+B O go to next pane ctrl+B ; go to last active pane ctrl+B } move pane right ctrl+B { move pane left ctrl+B ! convert pane to window ctrl+B X kill pane</pre>	<p>_ key bindings</p> <pre>alt+> go to bottom ctrl+B [enter copy mode ctrl+B] paste from buffer ctrl+S search n next search PageUp scroll page up PageDown scroll page down q quit</pre> <p>_ copy mode commands</p> <pre>ctrl+space start selection ctrl+W copy selection ctrl+g clear selection alt+< go to top</pre>

Git & IHEP Code

What is Git?

- A distributed code management tool
 - Like Mercurial, Bazaar and Darcs
- Git mode: Local + remote repositories
- Local repository: a complete local copy
 - Almost all operations can be performed
 - Push codes to remote common repo
- Remote repo: common copy shared with others
 - Users pull from or push to common repo



- Three areas for local repository:
 - Working area: Files you are working on
 - Staging area: files waiting for to be committed
 - .git dir: metadata + database
- Three kinds of file status:
 - File is changed
 - File is staged
 - File is committed

- Getting git

- Linux

```
$ sudo apt install -y git ### for
debian/ubuntu
$ sudo dnf install -y git ### for
fedora/alma/rhel
$ sudo yum install -y git ### for old
rhel/centos
```

- macOS: using brew or download installer

Configuring Git

- git config

```
# User info
$ git config --global user.name "John Smith"
$ git config --global user.email
"johnsmith@example.com"
# Editor
$ git config --global core.editor vim
```

- Show all git config

```
$ git config --list
```

- Show config usage

```
$ man git config
```

- Git config files

- system:/etc/gitconfig
- global:\$HOME/.gitconfig
- local:.git/config

- Useful aliases for git

```
[alias]
a = add .
v = !gitk
br = branch
ci = commit -m
cm = commit -m
co = checkout
df = diff
dump = cat-file -p
hs = log --pretty=format:"%h %ad | %s%d
[%an]" --graph --date=short
last = log -1 HEAD
pl = pull

ps = push
st = status
type = cat-file -t
sum = shortlog -sn
```

Basic Git Operations (I)

- Prepare local repository

- Using local dir

```
## Initializing one or using existing dir
$ git init demo
$ cd demo; git init
```

- Cloning existing repo

```
## Clong local repo
$ git clone /path/to/existing/repo/
## Clone remote repo
$ git clone
username@host:/path/to/my/repo
```

- Workspace operations:

- Project status

```
$ git status
On branch main...
$ git status -s      ## Status
M   README          # M   : modified a
M   lib/a.cpp        # M   : modified b
MM  lib/b.cpp        # MM  : staged but
A   lib/c.cpp        # A   : staged new
??  lib/c.o          # ??  : file not t
```

- Staging modified file

- Workspace operations

- Using .gitignore to skip unwanted files or directories

```
$ cat .gitignore
my_passwd      # Password file
*.o            # tmp files when compili
tmp/           # tmp directory
config*       # Config file or directo
dist*         # distribution dir
*.pyc         # binary python code fil
node_modules  # node module directory
```

- Commit operations

- Prepare your commit

```
$ git commit
Add README; Update RELEASE.md
.....
# On branch main
# Your branch is up-to-date with 'ori
.....
# Changes to be committed:
#   new file:   README
#   modified:   RELEASE.md
#
.....
README RELEASE.md
```

- Commit your commit

```
$ git add README
$ git status
On branch ...
Changes to be committed:...
  new file:   README
```

```
$ git commit -m "Add README; Update R
```

Basic Git Operations (II)

- Branch operations

- Let developers work unaffected in parallel
 - Default main/master to store main code
 - Others for developing features, fixing bugs ...

- Creation

```
$ git branch dev; git checkout dev
$ git checkout -b dev
```

- Merging

```
$ git checkout main; git merge dev
```

- Deletion

```
$ git branch -d dev
```

- Tag important commits with special name

- List all tags with name

```
$ git tag -l
v1.0
```

- Tag the selected commit

- Commits

- List commits gracefully

```
$ git log --pretty --graph --dat
```

- Confliction

- Confliction may occur when merging other branches.
- Special patterns indicating confliction

```
<<<<<<< HEAD: index.html
Contact: <mailto:support@gitl
=====
Please contact us at support@
>>>>>>> 1a4f: index.html
```

- Modifying the conflicted file manually

```
Please contact us at support@
```

- Re-adding the modified file

```
$ git add index.html
```

- Committing your change, and


```
$ git tag <tag-name> [commit]
```

- Delet a tag

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

push to remote repo

```
$ git commit -am 'fix conflic  
$ git push origin main
```

What is IHEP Code?

- A self-hosted community-version communal gitlab platform.
 - <https://code.ihep.ac.cn>
- Login using IHEP SSO account, email or username
 - Click to apply an IHEP SSO account

欢迎使用中科院高能所GitLab

1, IHEP SSO Account sign in/高能所统一认证帐号, 可以直接登录。

2, Others, apply for IHEP SSO Account /其他人需要申请统一认证帐号:

<https://login.ihep.ac.cn>

3, IHEP Gitlab Manual / 用户指南:

<http://code.ihep.ac.cn/codeguide.pdf>

4, Helps/帮助平台: <http://helpdesk.ihep.ac.cn> Tel./电话: 88236855

高能所计算中心负责本系统的可靠、稳定运行, 并会对托管代码及其数据进行定期备份。

您在使用过程中如果有任何问题, 请联系: helpdesk@ihep.ac.cn



IHEP SSO Account

Username

Password

Remember me

Sign in

Adding Your SSH Pubkey to Gitlab

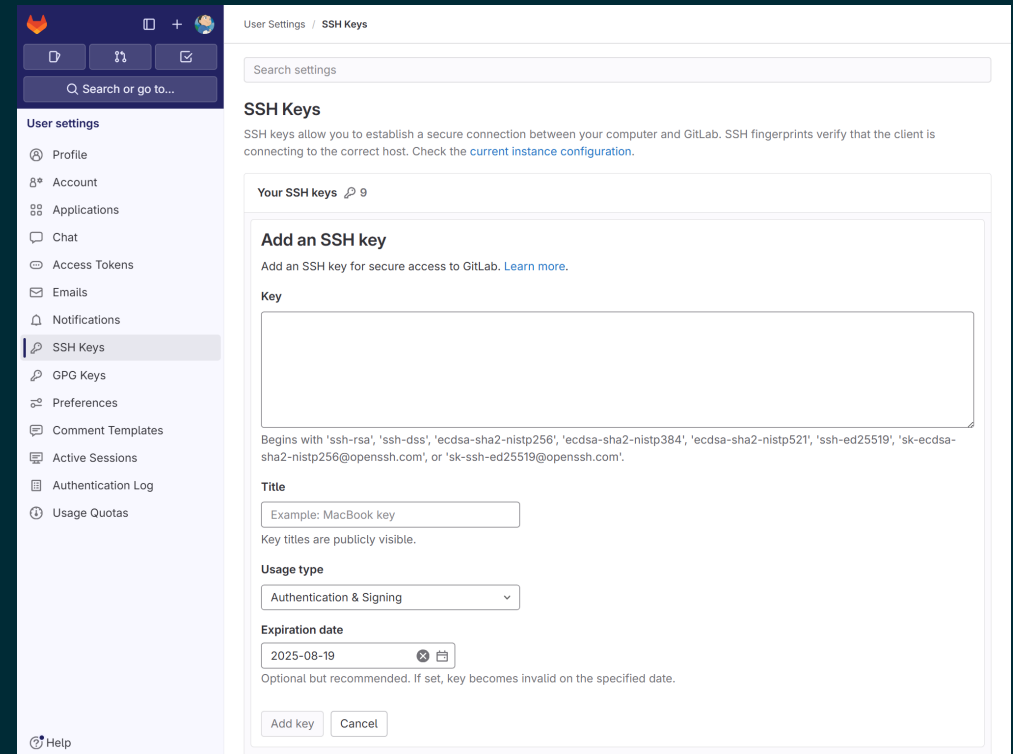
- Gitlab prefers SSH keys to push or pull codes without username/password
- User can add multiple pubkeys to gitlab
 - User logo -> Preferences -> SSH Keys -> Add new key

- Adding code.ihep.ac.cn to your ssh config

```
Host code
Hostname code.ihep.ac.cn
User git
Identityfile ~/.ssh/id_rsa
```

- Test your config:

```
$ ssh -T code Welcome to GitLab,
@biyujiang!
```



The screenshot shows the GitLab user settings interface. On the left is a sidebar with navigation options: Profile, Account, Applications, Chat, Access Tokens, Emails, Notifications, SSH Keys (selected), GPG Keys, Preferences, Comment Templates, Active Sessions, Authentication Log, and Usage Quotas. The main content area is titled 'SSH Keys' and includes a search bar, a description of SSH keys, and a section for adding a new key. The 'Add an SSH key' section contains a large text area for the key, a 'Title' field with the example 'MacBook key', a 'Usage type' dropdown menu set to 'Authentication & Signing', and an 'Expiration date' field set to '2025-08-19'. At the bottom of this section are 'Add key' and 'Cancel' buttons.

Creating A Project or Group

- Project: for single project or task
- Group: a workspace for a collection of similar tasks, complicated projects...

Your work / Projects

Projects

Explore projects [New project](#)

Yours 125 Starred 0

Filter by name Language Updated date

All Personal

Your work / Projects / New project / Create blank project

Create blank project

Create a blank project to store your files, plan your work, and collaborate on code, among other things.

Project name

My awesome project

Must start with a lowercase or uppercase letter, digit, emoji, or underscore. Can also contain dots, pluses, dashes, or spaces.

Project URL **Project slug**

https://code.hep.ac.cn/ Pick a group or namespace / my-awesome-project

Visibility Level

Private
Project access must be granted explicitly to each user. If this project is part of a group, access is granted to members of the group.

Internal
The project can be accessed by any logged in user except external users.

Public
The project can be accessed without any authentication.

Project Configuration

Initialize repository with a README
Allows you to immediately clone this project's repository. Skip this if you plan to push up an existing repository.

Enable Static Application Security Testing (SAST)
Analyze your source code for known security vulnerabilities. [Learn more.](#)

[Create project](#) [Cancel](#)

Your work / Groups / New group / Create group

Create group

Groups allow you to manage and collaborate across multiple projects. Members of a group have access to all of its projects. Groups can also be nested by creating [subgroups](#).

Group name

My awesome group

Must start with letter, digit, emoji, or underscore. Can also contain periods, dashes, spaces, and parentheses.

⚠️ Your group name must not contain a period if you intend to use SCIM integration, as it can lead to errors.

Group URL

https://code.hep.ac.cn/my-awesome-group

Visibility level

Who will be able to see this group? [View the documentation](#)

Private
The group and its projects can only be viewed by members.

Internal
The group and any internal projects can be viewed by any logged in user except external users.

Public
The group and any public projects can be viewed without any authentication.

Now, personalize your GitHub experience

We'll use this to help surface the right features and information to you.

Role

Software Developer

Who will be using this group?

My company or team Just me

What will you use this group for?

[Create group](#) [Cancel](#)

Using Gitlab to Manage Your Codes

- Clone your code:

```
$ git clone code:biyujiang/demo
Cloning into 'demo'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0),
pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.
```

- Create and enter an branch:

```
$ cd demo; git checkout -b dev
Switched to a new branch 'dev'
```

- Do someth important work:

```
$ echo 'This is a demo project' > README.md
```

- Check status:

```
$ git status
On branch dev
Changes not staged for commit:
.....
```

- Add and commit you modification:

```
$ git add .
$ git commit -m "Import notice"
[dev 01e0c49] Important notice
1 file changed, 1 insertion(+), 93 deletions(-)
)
```

- Push your work to gitlab

```
$ git push origin dev
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 16 threads
Compressing objects: 100% (1/1), done.
Writing objects: 100% (3/3), 269 bytes |
269.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-
reused 0
.....
To code:biyujiang/demo
5dbb2fb..ef455c4 dev -> dev
```

- Merge your work to main branch

```
$ git checkout main
Switched to branch 'main' Your branch is up to date
```

Container Technology

What is Container Technology?

What is Container Technology?

- A lightweight, portable and self-contained way to package and run software
 - Using linux virtualization technology to sperarte apps from host
 - Run apps in containers without modification on VMs, physical or cloud servers

What is Container Technology?

- A lightweight, portable and self-contained way to package and run software
 - Using linux virtualization technology to sperarte apps from host
 - Run apps in containers without modification on VMs, physical or cloud servers
- Properties of Container
 - Lightweight, portability, isolation, expandability, security

What is Container Technology?

- A lightweight, portable and self-contained way to package and run software
 - Using linux virtualization technology to sperarte apps from host
 - Run apps in containers without modification on VMs, physical or cloud servers
- Properties of Container
 - Lightweight, portability, isolation, expandability, security
- Types of containerization platform
 - Docker, Podman, Singularity, LXC ...

What is Container Technology?

- A lightweight, portable and self-contained way to package and run software
 - Using linux virtualization technology to sperarte apps from host
 - Run apps in containers without modification on VMs, physical or cloud servers
- Properties of Container
 - Lightweight, portability, isolation, expandability, security
- Types of containerization platform
 - Docker, Podman, Singularity, LXC ...
- Singularity over Docker
 - Singularity is specifically designed for HPC and scientific computing
 - Run with user privilege, avoiding the root privilege issues
 - Support HPC features such as MPI and GPU acceleration
 - Integrate better with the host filesystems than docker

Containers in IHEP Cluster

Containers in IHEP Cluster

- IHEP Container environment deployed on `/cvmfs/container.ihep.ac.cn/`
 - Providing `SL5/6/7`, Alma 9 and custom images to support various computing need
 - Adding `/cvmfs/container.ihep.ac.cn/bin/` to `$PATH` to use `hep_container` directly

```
$ echo 'export PATH=$PATH:/cvmfs/container.ihep.ac.cn/bin' >> ~/.bashrc
```

Containers in IHEP Cluster

- IHEP Container environment deployed on `/cvmfs/container.ihep.ac.cn/`
 - Providing `SL5/6/7`, Alma 9 and custom images to support various computing need
 - Adding `/cvmfs/container.ihep.ac.cn/bin/` to `$PATH` to use `hep_container` directly

```
$ echo 'export PATH=$PATH:/cvmfs/container.ihep.ac.cn/bin' >> ~/.bashrc
```

- User Manual
 - <http://afsapply.ihep.ac.cn/cchelp/zh/local-cluster/container/>

Containers in IHEP Cluster

- IHEP Container environment deployed on `/cvmfs/container.ihep.ac.cn/`
 - Providing `SL5/6/7`, Alma 9 and custom images to support various computing need
 - Adding `/cvmfs/container.ihep.ac.cn/bin/` to `$PATH` to use `hep_container` directly

```
$ echo 'export PATH=$PATH:/cvmfs/container.ihep.ac.cn/bin' >> ~/.bashrc
```

- User Manual
 - <http://afsapply.ihep.ac.cn/cchelp/zh/local-cluster/container/>
- Supported images: `hep_container exec <image-name>`
 - Included custom images

```
$ hep_container images
Hep_container support images:
SL5 : Scientific Linux 5
SL6 : Scientific Linux 6
SL7 : Scientific Linux 7
CentOS7 : CentOS Linux 7.9
CentOS78 : CentOS Linux 7.8
CentOS79 : CentOS Linux 7.9
Alma9 : Alma Linux 9.4
MYIMAGE : Custom Image file name
HepcMyImage : Custom Image file name
```


Using Containers in IHEP Cluster

Using Containers in IHEP Cluster

- List supported groups: `hep_container groups`
 - By specifying `-g <group>`, corresponding directories will be mounted in container.

```
$ hep_container groups Hep_container support groups:  
u07|atlas|atlasrun|comet|offline|physics|bes3|higgs|ams|cms...
```

Using Containers in IHEP Cluster

- List supported groups: `hep_container groups`
 - By specifying `-g <group>`, corresponding directories will be mounted in container.

```
$ hep_container groups Hep_container support groups:  
u07|atlas|atlasrun|comet|offline|physics|bes3|higgs|ams|cms...
```

- Using container environment: `hep_container shell <image-name>`
 - Launching a shell in a container to execute complicated commands.

```
$ hep_container shell Alma9  
Apptainer> cat /etc/redhat-release  
AlmaLinux release 9.4 (Seafoam Ocelot)
```

Using Containers in IHEP Cluster

- List supported groups: `hep_container groups`
 - By specifying `-g <group>`, corresponding directories will be mounted in container.

```
$ hep_container groups Hep_container support groups:  
u07|atlas|atlasrun|comet|offline|physics|bes3|higgs|ams|cms...
```

- Using container environment: `hep_container shell <image-name>`
 - Launching a shell in a container to execute complicated commands.

```
$ hep_container shell Alma9  
Apptainer> cat /etc/redhat-release  
AlmaLinux release 9.4 (Seafoam Ocelot)
```

- Executing simple commands: `hep_container exec <image-name>`
 - Running commands without entering container environment

```
$ hep_container exec Alma9 cat /etc/redhat-release  
AlmaLinux release 9.4 (Seafoam Ocelot)
```

Using Containers in IHEP Cluster

- List supported groups: `hep_container groups`
 - By specifying `-g <group>`, corresponding directories will be mounted in container.

```
$ hep_container groups Hep_container support groups:  
u07|atlas|atlasrun|comet|offline|physics|bes3|higgs|ams|cms...
```

- Using container environment: `hep_container shell <image-name>`
 - Launching a shell in a container to execute complicated commands.

```
$ hep_container shell Alma9  
Apptainer> cat /etc/redhat-release  
AlmaLinux release 9.4 (Seafoam Ocelot)
```

- Executing simple commands: `hep_container exec <image-name>`
 - Running commands without entering container environment

```
$ hep_container exec Alma9 cat /etc/redhat-release  
AlmaLinux release 9.4 (Seafoam Ocelot)
```

- Using custom image
 - Specifying your image with `MYIMAGE` variable

```
$ export MYIMAGE=/cvmfs/container.ihep.ac.cn/userimages/raser-1.2.sif  
$ hep_container shell MYIMAGE
```

Q & A