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

```
$ whoami
biyj
```

- `pwd`: display where you are

```
$ 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

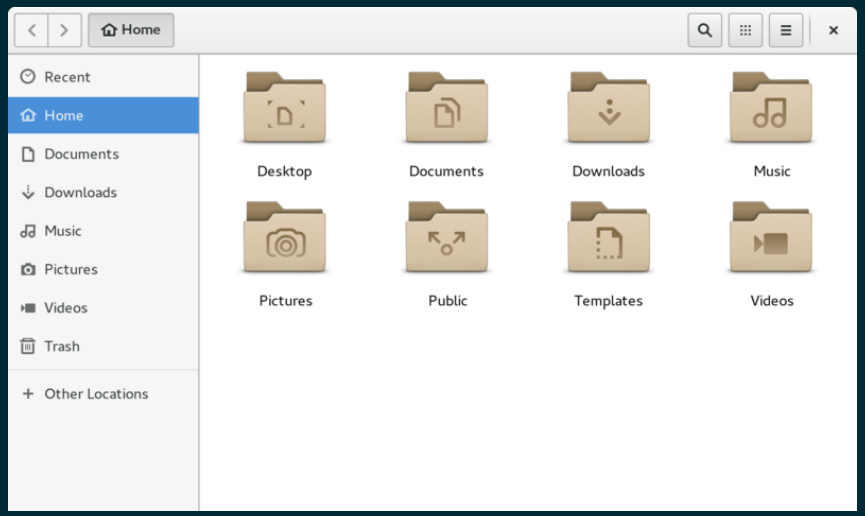
```
$ passwd
Changing password for user biyj.
Current password:
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
```

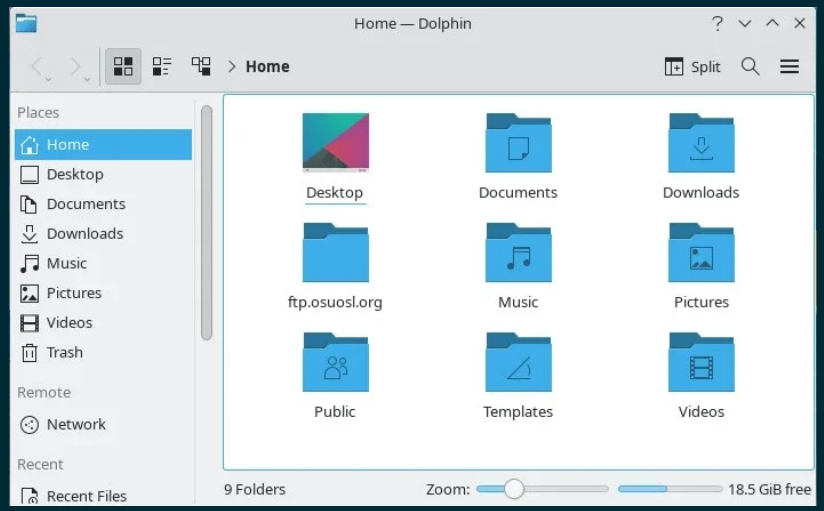
Where is my C Drive?

If you user headless environment:

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

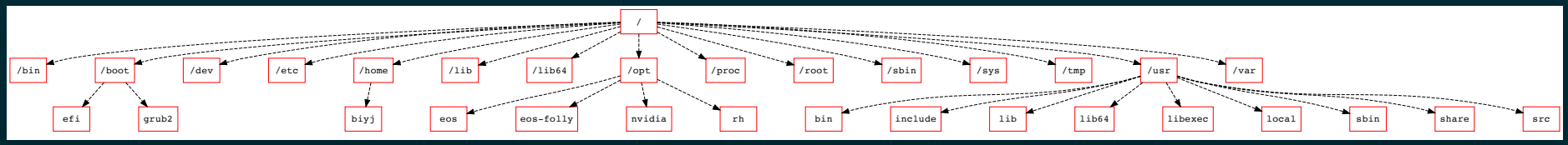
If you use DE:





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 temorary 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 prarent 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 descenc
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 hidde
.hidden  testfile
```

- Two kindes 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 ir
```

```
$ 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
```

- Backup mode: create a backup if the

- `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~
```

Linux Filesystem - find

- Search a directory for specific files
  - Usage: `find [path] [options] [expression]`
  - Default path is current directory if ommited
- 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 -1
ctime.within.10.txt
mtime.within.10.txt
```

- Find files created or modified 10 days ago

```
$ find . -type f \( -mtime +10 -o +ctime +1
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 appendeding 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 seperator. Seperating 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

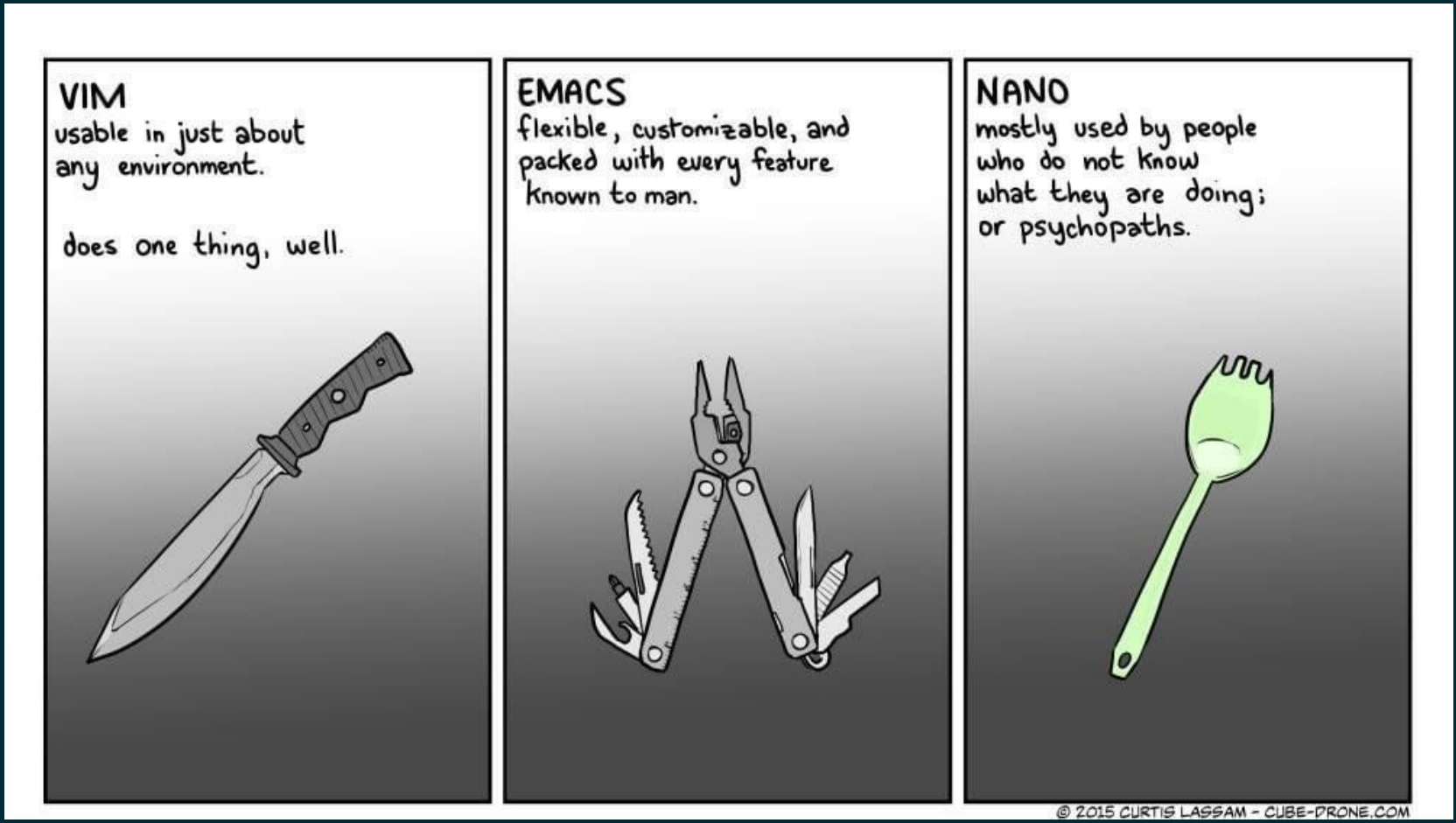
    ```
    $ my_command > my.log 2>&1 &
    ```

  - Using `nohup`

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

Text Editors

Classical Editors



The Holy War of Editors

5.4

Vim Cheatsheet

- Change Leader

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

- View a file in RO mode:

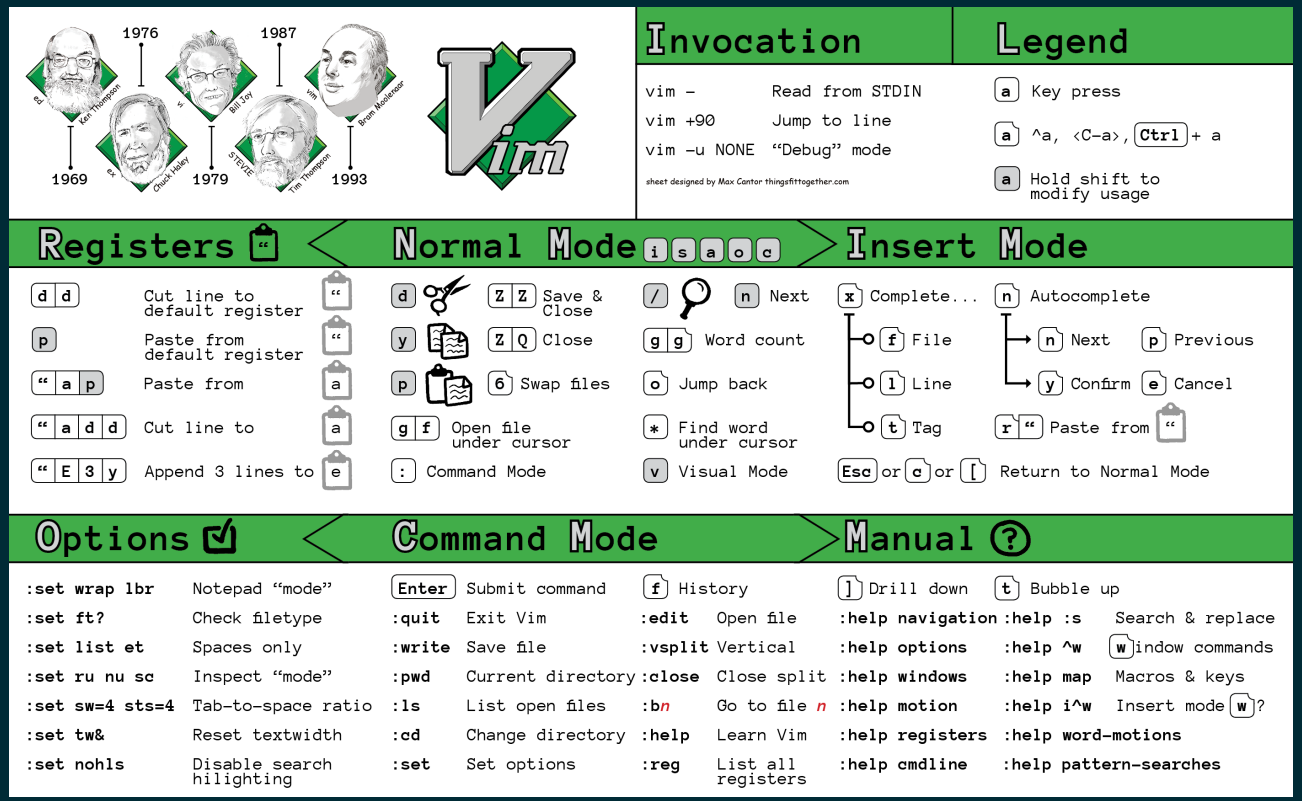
```
$ view f1 f2 ...
```

- Diff mode

```
$ vimdiff f1 f2
```

- Split mode

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



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:

```
$ ssh myuser@myhost
myuser@myhost's password:
Permission denied, please try again.
myuser@myhost's password:
Permission denied, please try again.
myuser@myhost's password:
myuser@myhost: Permission denied (publickey,gssapi-keyex,gssapi-with-mic,passwo
```

Avoid password with SSH key pair

```
$ -f ~/.ssh/id_rsa -t rsa -b 4096
Generating public/private rsa key pair.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa
Your public key has been saved in /root/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:9B+5DBkw/dYGYt06/gbzl7IJBSwkgPvgwb3n1OYAvUk root@myserver
The key's randomart image is:
......
```

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:

```
$ ssh lxlogin
......
Last login: Sun Aug 18 07:53:08 2024 from 10.100.0.75
[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 conntection 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 conntection 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 virtically:
  `<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`

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
```

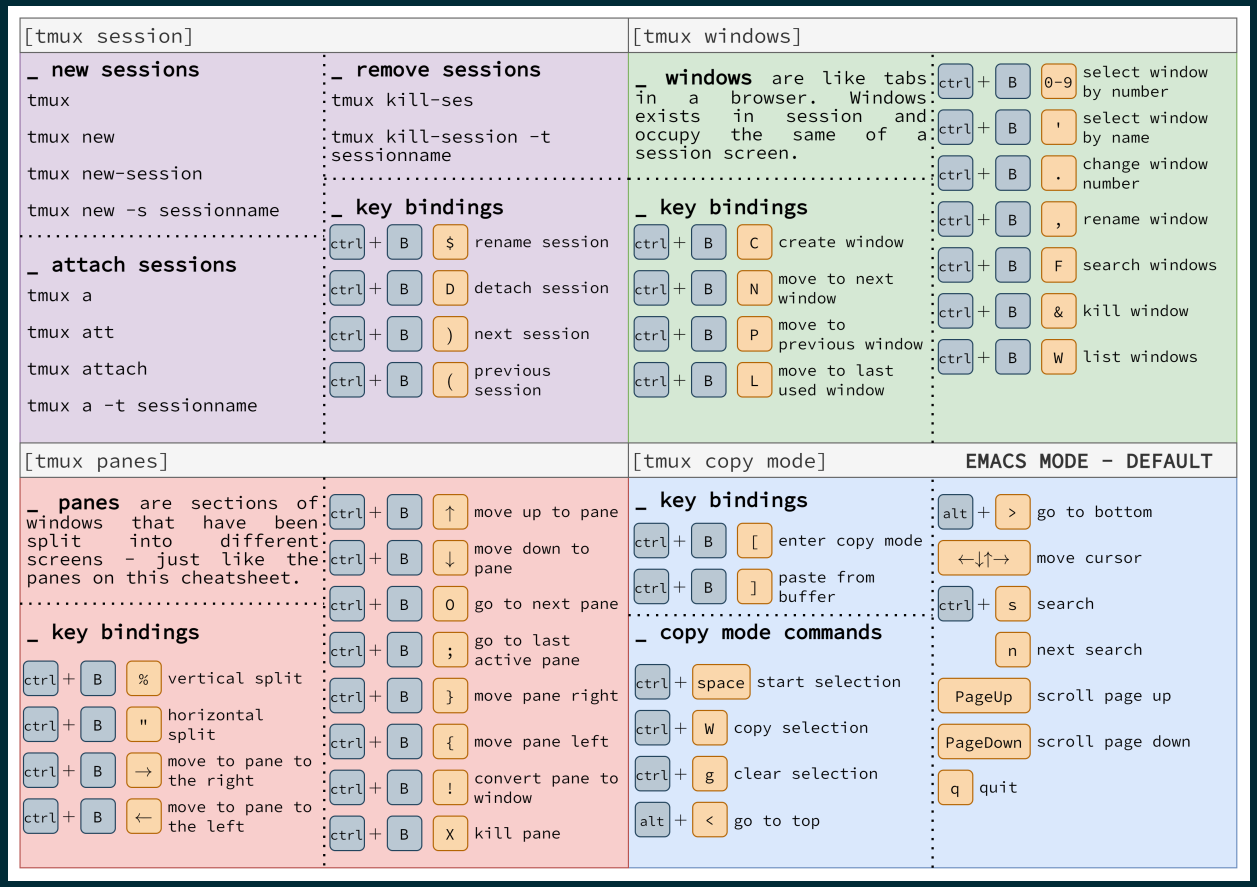
- List sessions:

```
$ tmux ls
```

- Attach session:

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

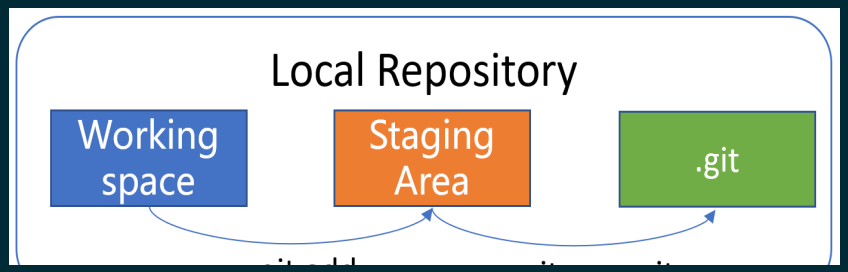
- **Remeber where your sessions are**



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 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
```

7.4

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 inidicating confliction

```
<<<<<<< HEADE: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

Adding Your SSH Pubkey to Gitlab

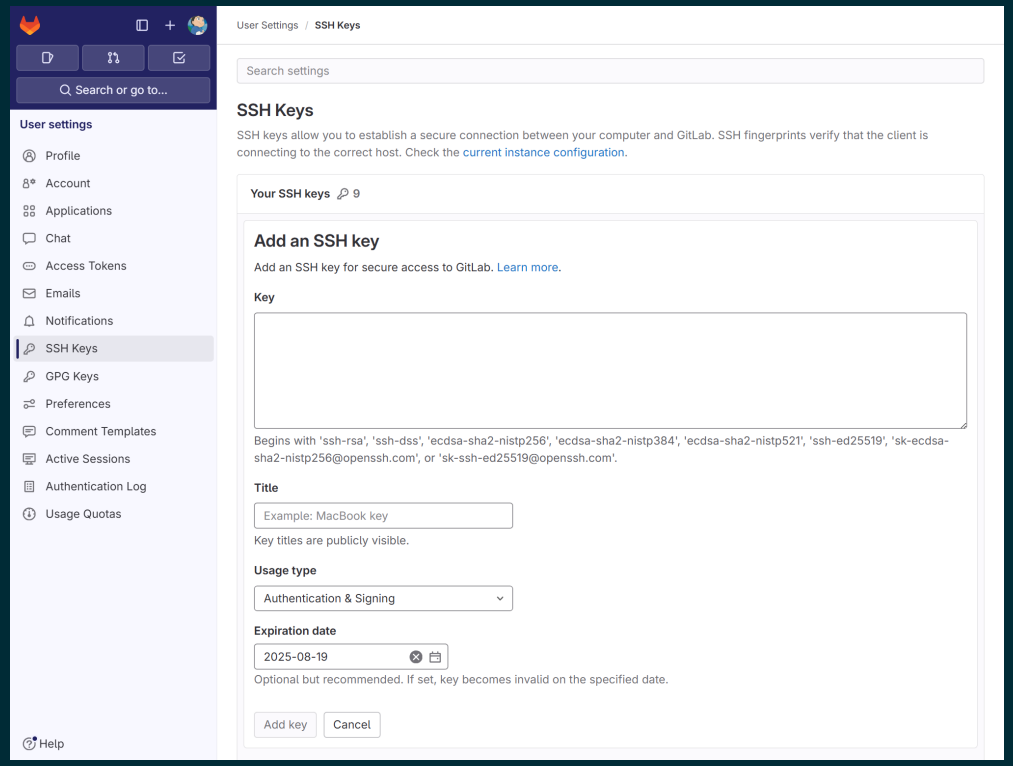
- Gitlab prefers SSH keys to push or pull codes without username/password
- User can add multiple pubkeys to gitlab
  - User logo -> Preferencs -> SSH Keys -> Add new key

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

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

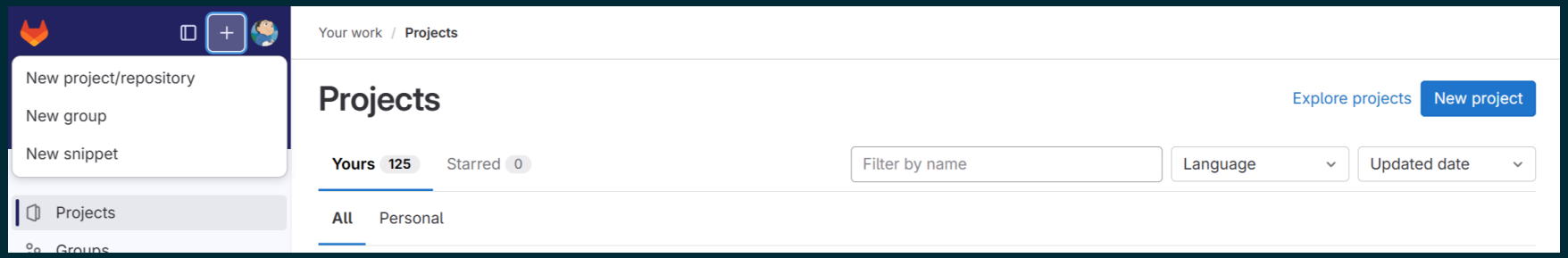
- Test your config:

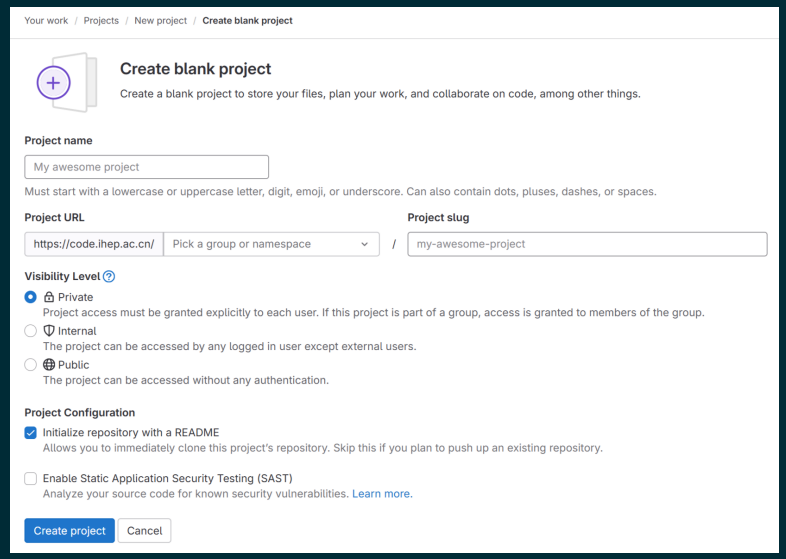
```
$ ssh -T code Welcome to GitLab,
@biyujiang!
```

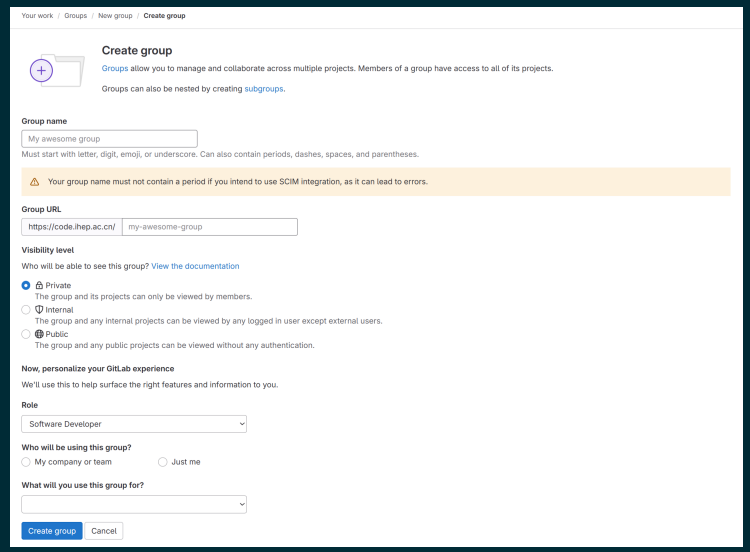


Creating A Project or Group

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







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 envrionment 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 envrionment 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 envrionment 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
  - Specifing your image with `MYIMAGE` variable

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

Q & A