ETH zürich

Scientific IT Services
High Performance Computing

Getting started with the scientific cluster
Preparation

Samuel Fux, Nadia Marounina
High Performance Computing Group
Scientific IT Services, ETH Zurich

Outline

- Basic Bash and Linux
- Accessing the cluster

Preparation > Linux terminal > Bash

On the Euler cluster we run **Linux** as operating system

To interact with the cluster, you can use the **Linux command line** which is a text interface referred to as a Linux terminal shell, console, prompts or other names.

In Linux and MacOS, a terminal or a console is provided. In Windows, you can use the third party application, e.g., MobaXterm (https://mobaxterm.mobatek.net/) which provides Local shell.

The following examples present useful basic command lines which are preceded with the dollar sign `$` and the lines below the command without the dollar sign are the screen output.

- Check your current directory

```
$ pwd
/home/sfux
```

- Create a directory

```
$ mkdir cluster-workshop
```

Preparation > Linux terminal > Bash

- Change directory

```
$ cd cluster-workshop
$ pwd
/home/jarunanp/cluster-workshop
```

cd commands	
`cd`	Change to the home directory
`cd ~`	Change to the home directory
`cd ..`	Change to the parent directory
`cd -`	Change to the previous directory
`cd dir1`	Change to `dir1` directory

Preparation > Linux terminal > Bash

- List files and folders in the current directory

```
$ ls
cluster-workshop

$ ls -l
drwxrwxr-x 3 jarunanp jarunanp 4096 Okt 26 08:26 cluster-workshop
```

ls commands	
`ls`	Display of files and folders
`ls -l`	Display Unix files types, permissions number, number of hard links, owner, group, size, last modified, date and filename
`ls -a`	Display all files, also the one starting with “.”
`ls -la`	Combine options

Preparation > Linux terminal > Bash

- Display a string to the standard output or redirect it to a file

```
$ echo hello
hello
$ echo hello > output
```

- Display the content of a file

```
$ cat output.log
hello
```

- More command to display content of a file

```
$ less file
$ more file
$ head file
$ tail file
```

Preparation > Linux terminal > Bash

- Copy a file

```
$ cp file1 file2
```

- Copy a directory

```
$ cp –r dir1 dir2
```

- Rename or move a file or a directory

```
$ mv file1 file2
```

- Delete a file

```
$ rm file1
```

- Delete a directory (it has to be empty)

```
$ rmdir dir1
```

Preparation > Linux terminal > Shortcuts which help reduce typing

- **TAB:** Type the first letter or first few letters of a filename or a directory name then press TAB to auto complete the filename or directory name

- Reuse a recently used command line by using a `!` , e.g., the command started with `python`

```
$ !python
```

- Find commands in history

```
$ history
```

- Press `Ctrl + r` then type first few letters of a command to search for previous commands

Preparation > Linux terminal > Globing

- `*` stands for **zero to more** characters. For example, there are 5 images in a folder called `Image.png`, `Image0.png`, `Image1.png`, `Image10.png` and `Image11.png`. To display all files with the extension `.png`, the command reads:

```
$ ls *.png
Image.png Image0.png  Image1.png Image10.png Image11.png
```

- `?` stands for **any single** character. For example, to display all files which start with `Image`, followed with only one number and ended with the extension `.png`

```
$ ls Image?.png
Image0.png  Image1.png
```

Preparation > Linux terminal > Basic commands

Command	Explanation
`ls`	Directory listing
`cd`	Change directory
`pwd`	Print working directory
`echo`	Print to standard output
`less, cat, more, head, tail`	Display content of a file to standard output
`cp, mv`	Copy/move a file or directory
`rm, rmdir`	Remove a file or directory
`mkdir`	Create a directory
`vi, nano, emacs`	Command line text editor
`man`	Show manual for a command in terminal
`grep`	Search for a pattern in a string or file

Preparation > Linux terminal > Basic commands

Command	Explanation
`exit`	Terminate and exit the current terminal
`sort`	Sort a file line by line
`sed`	String manipulation
`awk`	Programming language for text processing
`find`	Search for files
`du`	Show disk space used in a directory
`tar`	Create a tar archive with files/directories
`gzip`	Compress files/directories
`top`	Real time view of processes in a computer
`chmod`	Change permissions of file/directory

https://scicomp.ethz.ch/wiki/Linux_command_line

Preparation > Linux permissions

- In Linux, access to files and directories is handled via permissions
  - Read permission (r) - grants permission to read a file or directory
  - Write permission (w) - grants permission to write a file or directory
  - Execute permission (x) - grants permission to execute a file or directory
- There are 3 permission groups
  - User (u) - permissions for the user account that owns the file
  - Group (g) - permissions for the user group that owns the file
  - Other (o) - permissions for all other users except the user account and the user group

```
[sfux@eu-login-29 ~]$ ls -l gurobi.log
-rw-r--r-- 1 sfux sfux-group 800 Sep 17 10:29 gurobi.log
[sfux@eu-login-29 ~]$ ls -ld data
drwxr-xr-x 2 sfux sfux-group 4096 Jan  9  2017 data
```

https://scicomp.ethz.ch/wiki/Linux_permissions

Preparation > Linux permissions

- Another method for representing Unix permissions is an octal (base 8) notation
  - The read bit adds 4 to its total (in binary 100),
  - The write bit adds 2 to its total (in binary 010), and
  - The execute bit adds 1 to its total (in binary 001).
  - Example: `r-x` → 101 → 4+0+1=5
  - (u), (g) and (o) are then combined (755 represents `rwxr-xr-x`)
- Permissions can be changed with the `chmod` command
  - String representation:

    ```
    $ chmod ugo+rx filename
    ```

  - Number representation:

    ```
    $ chmod 750 filename
    ```

https://en.wikipedia.org/wiki/File-system_permissions#Numeric_notation

Preparation > Paths > Linux/Mac OS X vs. Windows

- There are differences how operating systems are representing file paths
- Paths on Windows: `C:\Users\Samfux\test`
  - Use backslashes
  - File and directory names are not case sensitive
  - Different drives (`C:`, `D:`, etc.)
- Paths on Linux/Mac OS X: `/cluster/home/sfux/test`
  - Use forward slashes
  - Everything is case sensitive
  - Everything is under the root file system (`/`), no drives

Preparation > Edit a text file with vim

- Vim is a command line text editor that can be started with the command `vi`. This text editor is useful to edit input files and write scripts directly on the cluster without copying forth and back files
- 2 modes (insert mode, command mode)
  - Type `i` to switch from command mode to insert mode
  - Type `esc` to switch from insert mode to command mode
- Insert mode: You can insert text into a text document
- Command mode: You can type commands after typing `:` (colon) and execute it with the enter key
  - To save a file, type `:w`
  - To close a file, type `:q`
  - To save and close a file, type `:wq`

https://www.tutorialspoint.com/unix/unix-vi-editor.htm

Preparation > Exercise (local shell)

1. Print the current working directory
2. Create a new directory called **test**
3. Change to the new directory
4. Create a text file **my_first_bash_script.sh** with vim with the following content

```
#!/bin/bash
pwd
hostname
echo "Good morning $USER"
```

5. Change the permission of the file to **755**
6. Do a directory listing of the **test** directory
7. Execute the file that you have created with the command `./my_first_bash_script.sh`
8. Use the command `grep` to search for the word **morning** in `my_first_bash_script.sh` file
9. Sort the lines in the text file alphabetically
10. Exit the shell

Preparation > Exercise solution

Tasks	Commands
Print the current working directory	`pwd`
Create a new directory called **test**	`mkdir test`
Change to the new directory	`cd test`
Create a text file **my_first_bash_script.sh** with vim with the following content	`vi my_first_bash_script.sh`
Change the permission of the file to **755**	`chmod 755 my_first_bash_script.sh`
Do a directory listing of the **test** directory	`ls test`
Execute the file that you have created with the command ./my_first_bash_script.sh	`./my_first_bash_script.sh`
Use the command grep to search for the word **morning** in my_first_bash_script.sh file	`grep "morning" my_first_bash_script.sh`
Sort the lines in the text file alphabetically	`sort my_first_bash_script.sh`
Exit the shell	`exit`

Outline

- Basic Bash and Linux
- Accessing the cluster

Access > Prerequisites

- A valid ETH account
- Local computer with an SSH client
  - Linux and macOS contain SSH client as part of the operating system
  - Windows users need to install a third party SSH client
    - MobaXterm (https://mobaxterm.mobatek.net/) is a free open source SSH client that we recommend

- An X11 server for graphical user interface (optional)
  - Linux (https://www.xorg.com)
  - macOS (https://xquartz.org)
  - Windows (included in MobaXterm)

Access > How to access the clusters

1. Start your SSH client
2. Use `ssh` command to connect to the login node of Euler

```
ssh username@euler.ethz.ch
```

3. Use your ETH credentials to login
4. First login
   - On first login a verification code is sent to your email address (*username*@ethz.ch)
   - By entering the verification code, your account is created automatically
   - New users must accept the cluster's usage rules

https://scicomp.ethz.ch/wiki/New_account_request_process_for_HPC_clusters

Access > SSH keys (Optional)

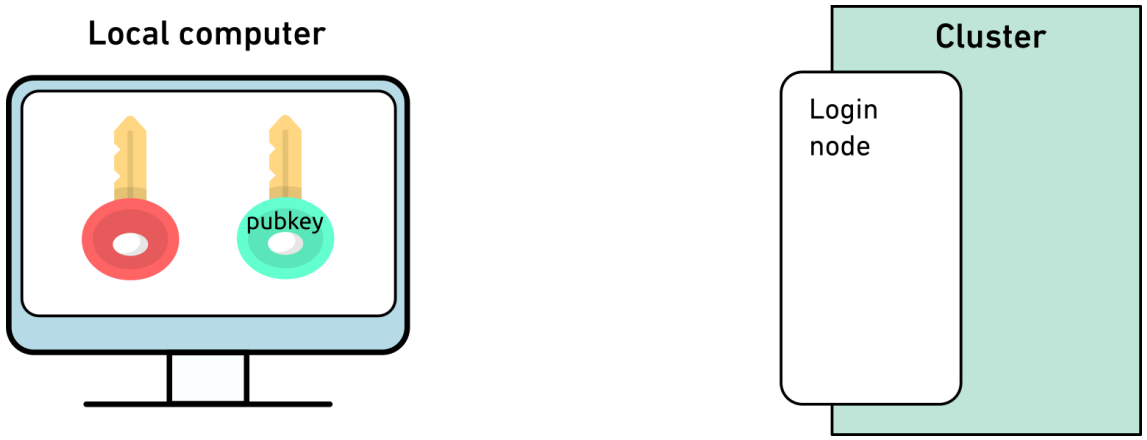
It is not compulsory to create and use SSH keys to participate the cluster workshop.

- SSH keys allows passwordless login
  - Useful for file transfers and automated tasks
  - When used properly, SSH keys are much safer than passwords

- SSH keys always come in pairs
  - A **private** key, stored on your local workstation (and nowhere else!)
  - A **public** key, stored on the computer(s) you want to connect to

- You can generate as many pairs as you like, e.g., one for each computer you intend to connect to
- Keys should be protected with a passphrase
- SSH key management tools such as `ssh-agent` and `keychain` help unlock SSH keys

Access > SSH keys > Step 1: Create your keys

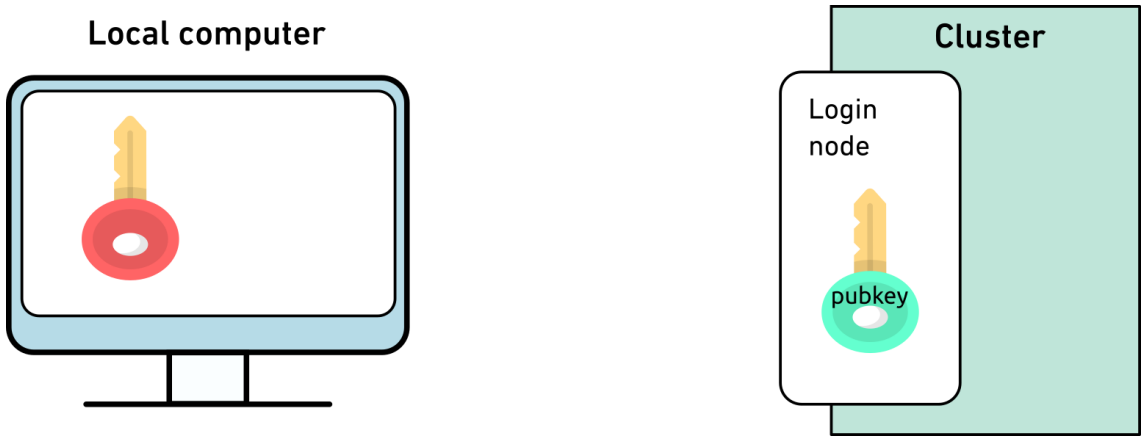


- Verify whether logging in with password works
- Generate a key pair with the ed25519 algorithm for each computer you want to connect to

```
ssh-keygen -t ed25519 -f $HOME/.ssh/id_ed25519_euler
```

- Enter a passphrase to protect your SSH keys

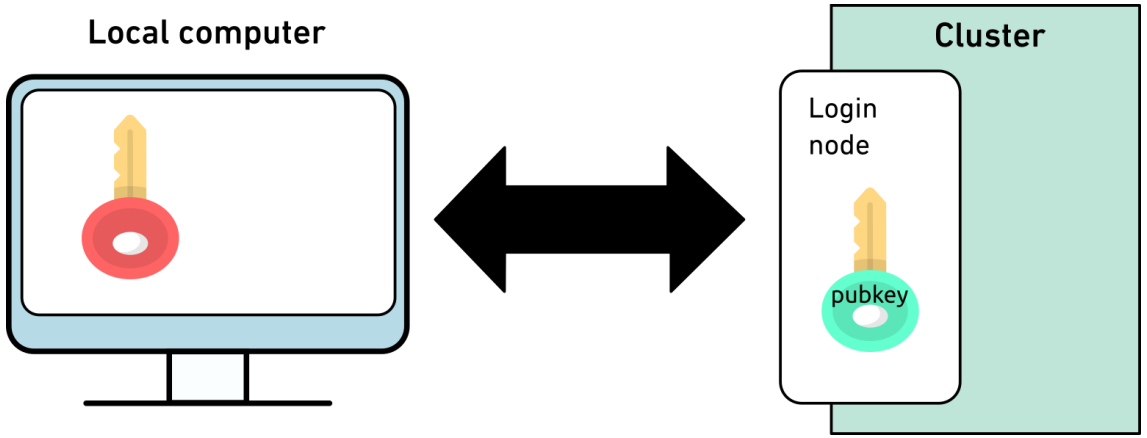
Access > SSH keys > Step 2: Copy the public key to the cluster




```
ssh-copy-id -i $HOME/.ssh/id_ed25519_euler.pub    username@euler.ethz.ch
```

https://scicomp.ethz.ch/wiki/Accessing_the_clusters#SSH_keys

Access > SSH keys > Step 3: Use keys with non-default names




```
ssh -i $HOME/.ssh/id_ed25519_euler    username@euler.ethz.ch
```

https://scicomp.ethz.ch/wiki/Accessing_the_clusters#How_to_use_keys_with_non-default_names

Access > SSH keys > Step 3: Use keys with non-default names

- SSH clients can use this option automatically by adding the option `IdentityFile` in your `$HOME/.ssh/config` file, e.g.:

```
Host euler
HostName euler.ethz.ch
User username
IdentityFile ~/.ssh/id_ed25519_euler
```

- Next time you login, you can type

```
$ ssh euler
```

https://scicomp.ethz.ch/wiki/Accessing_the_clusters#How_to_use_keys_with_non-default_names

Access > SSH Key Management > SSH Agent

As we have to enter the passphrase to unlock the keys, it takes away the convenience of passwordless login. We can use an SSH agent (`ssh-agent`) to unlock the SSH keys per terminal.

```
$ eval `ssh-agent`
Agent pid 17906

$ ssh-add -l
The agent has no identities.

$ ssh-add $HOME/.ssh/id_ed25519_euler
Enter passphrase for id_ed25519_euler:
Identity added: id_ed15519_euler (username@localcomputer)
```

ETH zürich

For any questions, please contact us

cluster-support@id.ethz.ch