**Basic Linux Commands:**

**To see the linux version:**

$ lsb\_release -a

**To clear the terminal screen:**

$ clear

**To get the present working directory:**

$ pwd

**List the contents of the directory:**

$ ls

$ ls /home

$ ls -l

$ ls -la

**Manual/help for a linux command:**

$ man ls

**Create a new directory:**

$ mkdir Testdir

$ ls -l

**Enter inside any directory:**

$ cd Testdir

**Create file: An empty file:**

$ touch file\_empty.txt

**Print the contents of a file:**

$ cat file\_empty.txt

**Print some message/print variable values within the terminal:**

$ echo "Print this message in my terminal"

$ echo $PATH

**Create file: with some content:**

$ echo "My file content for file1" > file1.txt

$ cat file1.txt

**Replace/Overwrite the contents of a file:**

$ echo "My updated content for file1" > file1.txt

$ cat file1.txt

Append to the contents of a file:

$ echo "Line 2 for file1.txt" >> file1.txt

$ cat file1.txt

**Change the file permission:**

$ chmod 777 file1.txt

[OR]

$ chmod u=rwx,g=rwx,o=rwx file1.txt

$ chmod a=r-- file1.txt

**Change the ownership:**

$ chown root file1.txt

Escalate a single command execution: sudo privilege:

$ sudo <YOUR\_COMMAND>

$ sudo chown root file1.txt

**Change the user to root user:**

$ sudo su

# exit

**Remove a file**:

$ rm file\_empty.txt

**Come out of the current directory to the parent directory**:

$ cd ..

**Remove a non-empty directory:**

$ rm -rf Testdir

**Install a package:**

$ sudo apt install tree

[OR]

$ sudo apt install -y tree

$ tree --version

**History of commands:**

$ history

$ history | tail -n 5

$ history | grep apt

**Command Line Text Editors:**

vim

nano

$ mkdir Testdir

$ cd Testdir

**vi**

$ vi file1.txt

i -> insert mode

Esc -> reset/switch the mode

Esc  :wq  Enter      -> write/save the contents of a file and quit the editor

$ cat file1.txt

**nano**

$ nano file2.txt

Ctrl+X -> y -> Enter      -> save & quit a file

$ cat file2.txt

**Tar & Extract tar files:**

**Create a tar from files:**

$ cd Testdir

$ tar -zcvf file.tar.gz file1.txt file2.txt

**See the contents of a tar file without extracting:**

$ tar -ztvf file.tar.gz

**Extracting files from a tar file:**

$ mkdir extractedfiles

$ tar -zxvf file.tar.gz -C extractedfiles

$ ls extractedfiles/

**Copy & Move files:**

$ echo "My src file" > src.txt

$ cp src.txt extractedfiles/

$ ls extractedfiles/

$ cat extractedfiles/src.txt

$ cp src.txt extractedfiles/renamed\_src.txt

$ ls extractedfiles/

$ cat extractedfiles/renamed\_src.txt

$ mv file.tar.gz extractedfiles/

$ mv file2.txt newfile2.txt

**Configure Git:**

$ git --version

SSH to lab machine:

$ ssh -p 42006 user\_name@user\_hostname

$ git config --global user.name "Your\_Name"

$ git config --global user.email "Your\_Email"

$ git config --global init.defaultBranch main

$ git config --list

**Create a git repo:**

$ mkdir mylocalrepo

$ cd mylocalrepo

$ sudo apt install -y tree

$ tree

$ tree .

$ git init

$ tree .git

$ echo "file 1 content" > file1.txt

$ git status

$ git add file1.txt

[OR]

$ git add .

$ git status

$ git status

$ git log [see the versions/commits]

**Create & clone a github repo:**

- Login to your github account: <https://github.com/>

- Create a new repo - initialize it with a readme file

> New repository

            > Repository name: remotejun24

            > Public repo

> Initialize - check - Add a README file

> Create repository

**- Clone the remote repo to local machine:**

$ git clone YOUR\_REMOTE\_REPO\_HTTPS\_URL (This command creates a local copy of the remote repository)

$ tree remotejun24

$ cd remotejun24

**Pushing a file to remote repo:**

$ echo "My local file created using local repo" > local.txt

$ git status

$ git add .

$ git commit -m "Added local.txt - my first local version"

$ git log

$ git remote

$ git remote -v

$ git remote add sshorigin YOUR\_GITHUB\_REPO\_SSH\_URL

$ git remote -v

$ git push -u sshorigin main (This command uploads your local changes to the `main` branch of the remote repository named `origin).

Authorize - ssh communication from labmachine to our Github account:

$ ssh-keygen

[Press Enter 3 times]

$ cat ~/.ssh/id\_rsa.pub

**Adding the SSH key to your Github account:**

> Login to your Github account > user profile pic > settings > SSH and GPG Keys > New SSH Key

> Add a title > Authentication key > Paste the public key of the lab machine to the key section > Add SSH key

$ git push -u sshorigin main

**Fetch & Pull:**

- Create a new file in the remote repo

Filename: remote.txt

File content: My remote file created in github

> Commit changes > Commit Changes

$ git pull origin main (This command updates your local repository with the latest changes from the `main` branch of the remote repository named `origin).

$ ls

$ cat remote.txt

$ git log

- **Create a new file in the remote repo**

Filename: fetch.txt

File content: My remote fetch file created in github

> Commit changes > Commit Changes

$ git log

$ git fetch origin main

$ ls

$ git log --all

**Forking & Pull Request:**

Source Repo Url: <https://github.com/creativeab88/sourceproject-jun24.git>

In the forked repo:

> Create a new file > filename: your\_name.txt > with some dummy content > create a commit

**Create a pull request:**

> Forked repo > Pull requests >  New Pull request > Create pull request > Add title & desc > Create pull request.

Explanation: You click the "Fork" button on the repository page to create a personal copy of the repository in your GitHub account

.

**Branching in Git:**

$ cd remotejun24

**To see the branches:**

$ git branch

$ git log

$ git log --oneline

$ git log --oneline  --decorate

**Create a new branch:**

$ git branch new\_branch

$ git branch

**Switch to a branch:**

$ git checkout new\_branch

$ git branch

**Rename a branch:**

$ git branch -M demo

$ git branch

$ echo "My first file on demo branch" > demo.txt

$ git add .

$ git commit -m "Added demo.txt on DEMO branch"

$ git log --oneline  --decorate

Push branch to the remote repo:

$ git checkout main

$ git pull origin main

$ git push sshorigin main

$ git push sshorigin --all

**Delete a branch:**

$ git branch test\_branch

$ git branch

$ git branch -d test\_branch

$ git branch

**Remotely delete a branch:**

$ git branch test\_branch

$ git branch

$ git push sshorigin --all

$ git branch -d test\_branch     [local branch deletion]

$ git branch

$ git push sshorigin --delete test\_branch

[OR]

$ git push sshorigin :test\_branch

**Merging Branch:**

$ git log --oneline --decorate

$ git log --oneline --decorate --all

$ git log --oneline --decorate --graph --all

$ git checkout <destination\_branch>

$ git checkout main

$ git merge <source\_branch>

$ git merge demo

$ git log --oneline --decorate --graph --all

**Merge Conflict:**

$ cd

$ mkdir myconflictrepo

$ cd myconflictrepo

$ git init

$ git status

$ echo "Hello, this is an examplefile. On MAIN branch" > example.txt

$ cat example.txt

$ git add .

$ git commit  -m "Added example file on main branch"

$ git checkout -b branch01

$ echo "Hello, this is an examplefile. On BRANCH01 branch" > example.txt

$ cat example.txt

$ git add .

$ git commit -m "Added example file on branch01 branch"

$ git log --oneline --all

$ git checkout main

$ cat example.txt

$ rm example.txt

$ git add .

$ git commit -m "Deleted example file from main branch"

$ git log --oneline --all

$ git merge branch01

$ git status

$ nano example.txt

$ git add .

$ git commit -m  "Resolved merge conflict commit"

$ git status

**Merge Conflict: content**

----

$ git checkout branch01

$ echo "Hello, this is an examplefile. New change on branch01" > example.txt

$ git add .

$ git commit -m "Updated the example file on branch01"

$ git log --oneline --all

$ git checkout main

$ cat example.txt

$ git merge branch01

$ echo "second line on main" >> example.txt

$ git add .

$ git commit -m "Added second line to example.txt on main branch"

$ git checkout branch01

$ echo "second line on branch01" >> example.txt

$ git add .

$ git commit -m "Added second line to example.txt on branch01 branch"

$ git checkout main

$ git merge branch01

$ git status

$ nano example.txt

**CONFLICT Resolution**

$ git add .

$ git commit -m "After my conflict resolution"

$ git log --oneline --all --graph

**Git Tags:**

$ cd remotejun24

$ git log --oneline

$ git tag v1.0-staging

$ git log --oneline

$ git tag

**Annotated Tag:**

$ git tag -a v1.0 -m "MY version 1.0"

$ git push sshorigin v1.0

$ git push sshorigin v1.0-staging

$ git tag -d v1.0-staging

$ git tag v0.1 <COMMIT\_ID>     [to tag a specific commit with id, COMMIT\_ID]

$ git push sshorigin --delete v1.0-staging

Stashing:

$ cd

$ git clone <https://github.com/abhijithvg/HelloWorld-.git>

$ cd HelloWorld-

$ ls

$ cat readme.txt

$ echo "New local work in progress content to readme.txt" >> readme.txt

$ cat readme.txt

$ echo "New local file content" > localWIP.txt

$ git pull

$ git status

$ git stash

$ git stash list

$ git status

$ cat readme.txt

$ git stash save --include-untracked

$ git stash list

$ ls

$ git status

$ git pull

$ cat readme.txt

$ git stash pop

$ git status

$ git stash list

$ git stash apply 0

$ git stash list

$ cat readme.txt

Rebase in Git:

$ cd

$ mkdir rebase

$ cd rebase

$ git init

$ echo "m1" > m1.txt

$ git add .

$ git commit -m "Added m1 to main branch"

$ git branch

$ git checkout -b feature1

$ git branch

$ echo "f1-1" > f11.txt

$ git add .

$ git commit -m "Added f11 to feature1 branch"

$ echo "f1-2" > f12.txt

$ git add .

$ git commit -m "Added f12 to feature1 branch"

$ git log --oneline --all --graph

$ git checkout main

$ echo "m2" > m2.txt

$ git add .

$ git commit -m "Added m2 to main branch"

$ git log --oneline --all --graph

$ git rebase feature1 main

$ git log --oneline --all --graph

Git Reset:

$ echo "NEW CONTENT ADDED TO m2" >> m2.txt

$ git add .

$ git commit -m "Updated m2.txt file content"

$ git log --oneline

$ git reset HEAD~1

$ git log --oneline

$ git status