Contents

1: INSTALL GIT ON LINUX 3

2: CONFIGURE GIT 3

3: CLONE A GITHUB REPO. 3

4: CREATE GIT ALIASES 3

5: CENTRALIZED GIT WORKFLOW 4

6: TRACKING FILE CHANGES 4

7: ROLLING BACK TO PREVIOUS COMMITS 5

8: CLEANING THE WORKING DIRECTORY 5

9: ADDING CHANGES TO THE LAST COMMIT 5

10: DELETING FILES IN GIT 5

11: IGNORING FILES IN GIT 6

12: RENAMING FILES IN GIT 6

13: CREATE A REPOSITORY IN GITHUB USING HTTPS 7

14: CREATE A REPOSITORY IN GITHUB USING SSH 7

15: FETCHING COMMITS MADE IN REMOTE REPO. 9

16: CREATE A FORK AND PULL REQUEST 9

17: PULLING COMMITS FROM GITHUB 10

18: MERGING FILES CHANGES IN GIT 10

19: GIT UPSTREAM 11

20: CREATE AND DELETE TAG 11

21: SWITCHING BETWEEN BRANCHES 12

22: MERGING BRANCHES IN GIT 12

23: RESOLVING MERGE CONFLICTS ON DELETE 13

24: RESOLVING MERGE CONFLICTS ON MODIFICATIONS 13

25: STASHING IN GIT 13

26: REBASING IN GIT 13

1: Install Git on Linux

- Use the following command to check the version of Git:

  $ git --version

- Installing the latest version of Git

  $ sudo apt-get update

  $ sudo apt-get install git

2: Configure Git

- Configuring the username and email id.

git config --global user.name "USERNAME"

git config --global user.email "USERMAIL"

Note: In place of *USERNAME* and *USEREMAIL* use your github account's username and email id.

- Confirming the username and email id

git config --global user.name

git config --global user.email

- Enabling credentials storage globally

git config --global credential.helper store

3: Clone a GitHub Repo.

- Open the **Terminal** tab on your lab and use the following command to clone the repository:

  git clone URL

  Note: Replace URL with the copied url from the repository

4: Create Git Aliases

- Creating aliases for common Git commands
- Open the terminal and execute the following commands to create the respective aliases:

  git config --global alias.co checkout

  git config --global alias.br branch

  git config --global alias.ci commit

  git config --global alias.st status

  git config --global alias.ad add

- Use the following command to list all the git aliases:

  git config -l | grep alias

5: Centralized Git Workflow

- Cloning a GitHub repository

  git clone <your_repository_name.git>

- Go to the repository folder using the **cd** command
- Adding a file to the cloned repository

  touch demo.txt

- Use the following command to check the status of the repository:

  git status

NOTE:If an **untracked file**, is available in the repository which is not added to Git, and Git is not able to track it.

- Use the following command to add the file to the staging area:

  git add demo.txt

- Check the repository status again

  git status

- Use the following command to commit the changes in the repository:

  git commit -m "Added a txt file"

- Use the following command to push the file to the main branch:

  git push origin main

- Also, if the command above throws an error, you can use:

  git push -u origin master

6: Tracking File Changes

- Make some changes in the file

  vi demo.txt

- Add some sample content in the file **demo.txt**, for instance:

  This is a Demo.txt file

- Save the file, and exit by pressing **esc** key and **shift+: wq**
- Use the following command to compare the file in the working directory with its last staging area:

  git diff demo.txt

Note: The **+** statement is showing the changes in the file.

- Use the **git add** command to add the file to the staging area

  git add demo.txt

- Use the **git diff** command to track any changes

  git diff demo.txt

- Use the following command to check the recent log of commits with **--oneline** flag

  git log --oneline

7: Rolling Back to Previous Commits

- Rolling back to a previous commit using the **reset** command
- In the terminal, use the **git log --oneline** command to check the log of recent commits

  Note: Ensure that you are in the repository directory that you have cloned. Make sure to type **cd <your_repository_name>** before you begin this step.
- Use the following command to rollback to a particular commit:

  git reset COMMIT_VALUE

  where **COMMIT_VALUE** is the number shown in front of each commit
- Rolling back to a previous commit using the **revert** command
- Use the following command to rollback to a particular commit:

  git revert COMMIT_VALUE

  where **COMMIT_VALUE** is the number shown in front of each commit

8: Cleaning the Working Directory

- Use the following command to perform a dry run on cleaning the untracked files:

  git clean -n
- Use the following command to force clean the untracked files:

  git clean -f
- Use the following command to clean the untracked directories:

  git clean -f -d

9: Adding Changes to the Last Commit

- Use the following command to modify the most recent commit

  git commit --amend

Editing the commit message in the text editor that shows up after executing the **amend** command.

10: Deleting Files in Git

- Use the following command to remove the file from the repository:

  rm <YourFileName>

Committing the changes

11: Ignoring Files in Git

- Creating the **.gitignore** file

Define the rules that you need to keep and save the file.

- Execute the following commands to create the **.gitignore** file

touch .gitignore
echo “README.txt”>>.gitignore

12: Renaming Files in Git

- Execute to rename the file

git mv <old file name> <new file name>

13: Create a Repository in GitHub Using HTTPS

- In the terminal execute the following commands to create and initialize a Git repository

mkdir <your_repo-name>

cd <your_repo-name>

echo "your content" >> README.md

git init

git add README.md

git commit -m "first commit"

git branch -M main

- Create a new repositoryin git & Copy the HTTPS URL from the newly created repository

git remote add origin <Your HTTPS_URL>

- Execute the **git remote -v** command to check remote repository

14: Create a Repository in GitHub Using SSH

- Generate a new SSH key

ssh-keygen -t rsa -b 4096 -C "<your_email@example.com>"

Note: Replace **<your_email@example.com>** with your GitHub email address and press **Enter** for **Enter a file in which to save the key** and **Enter passphrase**

- Use the following command to open the **id_rsa.pub** file and copy the SSH key to the clipboard

cat < ~/.ssh/id_rsa.pub

- Go to **github.com** and click on the profile photo in the upper-right corner
- Click on the **Settings** button and navigate to **SSH and GPG Keys**
- Click on the **New SSH Key** button
- Enter the Title as **mySSHKey** and **Key** (copy paste the key that is displayed in your terminal) and click on the **Add SSH key** button
- Use the following commands to create and initialize a Git repository

mkdir <your_repo-name>

cd <your_repo-name>

echo "# your content" >> README.md

git init

git add README.md

git commit -m "first commit"

git branch -M main

- Go to **github** create a new repository

	• Copy the SSH URL from the newly created repository • Use the following command to add a remote repository: **git remote add origin SSH_URL** Replace SSH_URL with the copied URL • Execute **git remote -v** command to check the remote repository

15: Fetching commits made in remote repo.

Fetching the changes from remote repository
git fetch

git rebase origin/main

16: Create a Fork and Pull Request

- Create a Fork
- Cloning your Fork
- Type the following command on your terminal:

  git clone [the copied HTTPS URL]
  Note: Replace [the copied HTTPS URL] in the above command with the URL you copied from the Github page.
- The **clone** command creates a local git repository from your remote fork on GitHub.
- Sync fork with the original repository Navigate to the directory where the fork has been cloned.
- Use the following command to see the configured remote repository for your fork:

  git remote -v
- Use **git remote add upstream** command, and paste the URL you copied

  git remote add upstream [the copied HTTPS URL]
- Verify the new upstream repository you have specified for your fork:

  git remote -v
- Use the **git push** command to upload your changes to your remote fork on GitHub:

  git push
- Creating a Pull Request
- On the GitHub page of your remote fork, click the **pull request** button
- Wait for the owner to merge or comment on your changes

17: Pulling Commits from GitHub

- Check the logs for commits history

git log --oneline

- Pull the main branch from the remote repository

git pull origin main

- Checking the logs for the latest commits

git log --oneline

18: Merging Files Changes in Git

- Create a Git repository on the local machine

git init

- Create a remote repository on GitHub , Add the file to it & Commit the changes in the GitHub remote repository
- Add the remote repository to the local repository using HTTPS url
  git remote add origin <Your HTTPS_URL>
- Verify the setup of remote repository:

git remote -v

- Pull files from remote repository, to download files from the remote repository

git pull origin main --allow-unrelated-histories

Note: Please modify the above command to use the *main* or *master* branch depending on your repository structure.

- If you Edit the file in the remote repository, Changes will not yet reflect in the local repository

cat demo.txt

- **So** Merge the changes in edited file in the local and remote repository to update the local repository file:

git fetch

git checkout origin/main -- <file_name>

- Finalize the merge by creating a new commit using the following command:

git commit -m "'Merge' demo.txt from"

19: Git Upstream

- Create a repository on the local machine and

  git init

- Create a remote repository on GitHub
- Add the remote repository to the local repository using HTTPS url

git remote add origin <Your HTTPS_URL>

- Run the following command to verify the setup of remote repository:

  git remote -v

- Add upstream to the remote repository by executing the below command:

  git remote add upstream <Your HTTPS_URL>

- Verifying if the remote repository is added correctly

  git remote -v

- Fetch upstream

  git fetch upstream

- Updating the local branch with respect to the upstream branch

  git checkout main

  git merge upstream/main --allow-unrelated-histories

Note: Please modify the above command to use the *main* or *master* branch depending on your repository structure.

20: Create and Delete Tag

- Creating a tag

  git tag <tagname>

- Listing all the tags

  git tag

- Adding a description to your tag

  git tag <tagname> -a

  Save the file with **ctrl + x**

- Deleting a tag

  git tag -d <tagname>

21: Switching Between Branches

- Create a new branch, navigate to the **repository** folder
- Execute the following command to create a new branch:

git branch <branch_name>

Step 2: Switching to the new branch

- Use the following command to switch to the newly created **branch**:

git checkout <branch_name>

- Create a file and committing the changes
- Use the following commands to add the file to the **<branch_name>** and commit the changes:

git add file>

git commit -a -m "file modified"

- Check the status of the new branch

git status

- Use the following command to check the git logs:

git log --oneline

- Check the current branch using the following command:

git branch

- Use the following command to switch back to the main branch:

git checkout master

22: Merging Branches in Git

- Use the following command to switch back to the main branch:

git checkout main

- Use the following command to merge the test_branch to the main branch:

git merge <you-branch>

23: Resolving Merge Conflicts on Delete

- Merging the branches to create a conflict
- Use the following command to checkout the master branch

  git checkout master

- Merge the **<new-branch>** branch with the master branch

  git merge <new-branch>

- If The **Auto-merging is failed** because of the conflict in the file.
- Delete the conflicted file to resolve the merge conflict
- Use the following commands to delete the conflicted file and check the status of master branch

  git rm demo.txt

  git status

24: Resolving Merge Conflicts on Modifications

- Resolving the merge conflict by modifying the conflicted file
- Modify the file to resolve the conflict
- Remove line numbers from the content to resolve the merge conflictCheck the git status on master branch

25: Stashing in Git

- Use **git stash** to stash the changes and working on another branch
- Use the following command to stash the changes on master branch

  git stash

- Switch to new branch

  git checkout -b <new-branch>

- Create a file in the new branch

  touch file.txt

- Switch back to the master branch

  git checkout master

26: Rebasing in Git

- Executing git rebase Use the following command to execute git rebasing:

  git rebase --autostash main

• Use the following command to restore the last saved state using stashing **git stash pop**	