NAME: VOMMIDAPU CHINNI **BATCH NO:** 120

MAIL.ID: vommidapuchinni@gmail.com **DATE:** 11/05/24

TRAINER: Mr. Madhukar **MODULE:** GIT

CASESTUDY - 9

write all git commands and their use cases?

1. Git config:

- Get and set configuration variables that control all facets of how Git looks and operates.
- **Set the name:** git config --global user.name "User name"
- **Set the email:** git config --global user.email himanshudubey481@gmail.com
- **Check the setting:** git config -list

2. Git init: Create a local repository ➔ git init

Use: Initializes a new Git repository in the current directory or in a specified directory

Git clone: Make a local copy of the server repository ➔ git clone <repo_URL>

Use: Clones a repository from a remote source (e.g., GitHub, GitLab) to your local machine

3. Git add:

- **Add a file** to staging (Index) area:
  git add Filename
- **Add all files** of a repo to staging (Index) area:
  git add .

Use: Adds file changes to the staging area in preparation for committing

4. Git commit: Record or snapshots the file permanently in the version history with a message.
git commit -m " Commit Message"

Use: Records changes to the repository along with a descriptive commit message

5. Git diff:

- Track the changes that have not been staged: git diff
- Track the changes that have staged but not committed: git diff –staged
- Track the changes after committing a file: git diff HEAD
- Track the changes between two commits: git diff
- Git Diff Branches: git diff < branch 2>

Use: Shows the differences between the working directory and the staging area or between commits

6. Git status:

- Display the state of the working directory and the staging area.
  git status

Use: Displays the current state of the working directory and the staging area

7. Git show: Shows objects**:** git show

Use: Display detailed information about a specific commit, including the commit message, author, timestamp, and the changes introduced by that commit

8. Git log

- Display the most recent commits and the status of the head: git log
- Display the output as one commit per line: git log -oneline
- Displays the files that have been modified: git log -stat
- Display the modified files with location: git log -p

Use: Displays a list of commits in reverse chronological order along with their details

9. Git blame:

- Display the modification on each line of a file: git blame <file name>

Use: Display line-by-line revision history of a file, showing the commit hash, author, timestamp, and the last revision where each line of the file was modified.

10. .gitignore:

- Specify intentionally untracked files that Git should ignore. Create .gitignore:
  touch .gitignore
- List the ignored files: git ls-files -i --exclude-standard

Use: Specify intentionally untracked files that Git should ignore

11. Git branch:

- Create branch: git branch <branch name> or git checkout -b <branch name>
- List Branch: git branch or git branch –list
- Delete a Branch: git branch -d <branch name>
- Delete a remote Branch: git push origin -delete <branch name>
- Rename Branch: git branch -m<branch name>

Use: Lists existing branches or creates a new branch

12. Git checkout: Switch between branches in a repository.

- Switch to a particular branch: git checkout <branch name>
- Create a new branch and switch to it: git checkout -b <branch name>
- Checkout a Remote branch: git checkout

13. Git stash: Switch branches without committing the current branch.

- Stash current work: git stash
- Saving stashes with a message: git stash save ""
- Check the stored stashes: git stash list
- Re-apply the changes that you just stashed: git stash apply
- Track the stashes and their changes: git stash show
- Re-apply the previous commits: git stash pop
- Delete a most recent stash from the queue: git stash drop
- Delete all the available stashes at once: git stash clear
- Stash work on a separate branch: git stash branch

Use: Temporarily shelves changes so you can work on something else, then come back and reapply them later

14. Git cherry pic
Apply the changes introduced by some existing commit:

- git cherry-pick <commit_id>

15. Git merge:

- Merge the branches: git merge
- Merge the specified commit to currently active branch: git merge

Use: Merges changes from one branch into another

16. Git rebase
Apply a sequence of commits from distinct branches into a final commit.
git rebase

- Continue the rebasing process: git rebase -continue
- Abort the rebasing process: git rebase –skip

17. Git interactive rebase:

- Allow various operations like edit, rewrite, reorder, and more on existing commits.
  git rebase -i

18. Git remote:

- Check the configuration of the remote server: git remote -v
- Add a remote for the repository: git remote add
- Fetch the data from the remote server: git fetch
- Remove a remote connection from the repository: git remote rm
- Rename remote server: git remote rename
- Show additional information about a particular remote: git remote show
- Change remote: git remote set-url

19. Git origin master

- Push data to the remote server: git push origin master
- Pull data from remote server: git pull origin master

20. Git push: Transfer the commits from your local repository to a remote server.

- Push data to the remote server: git push origin master
- Force push data: git push -f
- Delete a remote branch by push command: git push origin -delete <branch name>

Use: Pushes local commits to a remote repository

22. Git pull:

- Pull the data from the server: git pull origin master
- Pull a remote branch: git pull <branch name>

Use: Fetches changes from a remote repository and merges them into the current branch

23. Git fetch: Download branches and tags from one or more repositories.

- Fetch the remote repository: git fetch< repository Url>
- Fetch a specific branch: git fetch
- Fetch all the branches simultaneously: git fetch -all
- Synchronize the local repository: git fetch origin

24. Git revert:

- Undo the changes: git revert
- Revert a particular commit: git revert <commit_id>

Use: To create a new commit that undoes the changes made by a previous commit

25. Git reset
Reset the changes:

- git reset -hard <commit_id>
- git reset -soft <commit_id>
- git reset –mixed<commit_id> (default)

Use: Resets the current HEAD to a specified state, optionally modifying the index and working directory

26. Git rm:

- Remove the files from the working tree and from the index: git rm <file Name>
- Remove files from the Git But keep the files in your local repository: git rm --cached