1. **command to rollback last commit in git //git reset –hard//**
2. **difference b/w git and GitHub**

**Git**: Git is a distributed version control system (DVCS) designed for tracking changes in source code during software development. It allows multiple developers to collaborate on projects, track changes, and manage different versions of files.

GitHub provides a platform for hosting and managing Git repositories with added collaboration and project management features.

1. **diff b/w repository and branch**

a repository is the entire collection of files and their history, while a branch is a specific line of development within that history which facilitates flexible and efficient collaboration among developers by allowing them to work on different aspects of a project simultaneously while maintaining version control and history.

1. **diff b/t git pull and git push**

**git pull** fetches changes from the remote and merges them into the current branch, while **git push** sends local changes to the remote repository.

1. **git pull and git fetch**

**git pull** is a combination of **git fetch** and **git merge**. It fetches changes from the remote repository and merges them into the current branch. **git fetch** only retrieves changes from the remote repository and doesn't merge them into the current branch.

1. **git clone and git remote**

**git clone** is used to create a copy of a remote repository locally. **git remote** is used to manage connections to remote repositories.

1. **git diff and git status**

This command is used to show the differences between different states of the repository. It can be used to compare changes between the working directory and the index (staging area), changes between the working directory and the last commit, or even between two different commits

**git status** This command is used to show the current status of the repository. It displays information such as which files have been modified, which files are staged for the next commit, and which branch is currently checked out.

**1. \*\*Working Directory vs. Staging Area (Index) //git diff**

**2. \*\*Staging Area (Index) vs. Last Commit //git diff --cache or git diff --staged**

**3. \*\*Working Directory vs. Last Commit //git diff HEAD**

**4. \*\*Between Two Commits // git diff <commit1> <commit2>**

1. **git merge and git rebase**

**git merge** integrates changes from one branch into another. It creates a new commit that combines the changes from both branches.

git merge branch\_name

**git rebase** is a way to integrate changes from one branch into another by moving all the commits from one branch onto the tip of another, creating a clean, linear history.

* **Merge**: Join two branches together, creating a new point where they meet.
* **Rebase**: Move the entire history of one branch onto the tip of another, making it look like the work was done on top of the other branch from the beginning

If you encounter issues during the merge process and want to abort the merge, you can use: **git merge –abort// command to abort a conflicting merge in Git?**

1. **what is git repository and how to create it**

A Git repository is a storage location where your project's files and their revision history are stored. To create a Git repository, you can use the **git init** command within your project directory

1. **how to fix a broken commit**

To fix a broken commit, you can use **git commit --amend** to add changes to the last commit or **git reset HEAD~1** to undo the last commit and keep changes staged.

1. **what happens if the git directory gets deleted**

All commit history, branches, and other repository data will be lost unless you have a backup.

1. **what is the use of staging area or indexing in git.**

the staging area or index acts as a middle ground between your working directory (where you make changes) and your repository (where changes are permanently stored). It allows you to organize and review your changes before officially saving them to your project's history. This helps ensure that only the changes you want are included in each commit, making your project's history clean and manageable.

1. **what are different git repository hosting function**

GitHub, GitLab, Bitbucket, and Azure DevOps,

1. **what is version control system?**

Version control system (VCS) is a system that records changes to files over time, allowing you to recall specific versions later. It helps manage changes to source code, documents, and other files, enabling tracking history, and reverting to previous states.

1. **what are the advantages of git**

distributed development, easy branching and merging, decentralized workflow, built-in version control, and it provides features like tracking changes, managing conflicts, and enabling code review processes.

**The `git branch` command in Git:**

**1. \*\*List Branches\*\*: git branch**

**2. \*\*Create a New Branch: git branch <branch\_name>**

**3. \*\*Delete a Branch\*\*: git branch -d <branch\_name>**

**4. \*\*Rename a Branch\*\*: git branch -m <old\_branch\_name> <new\_branch\_name>**

**5. \*\*Switch Branches\*\*: git checkout <branch\_name> or git switch <branch\_name>**

**6. Create &Checkout a New Branch: git checkout -b <new\_branch\_name> or git switch -c <new\_branch\_name>**

1. **what is the command to set up your git username and git email?**

**git config --global user.name "Your Name"**

**git config --global user.email "your\_email@example.com"**

1. **command to add a file or multiple files to the staging area git add <file1> <file2> ...**
2. **command to commit changes with a message in git. git commit -m “my message”**
3. **see commit history in git //git log**
4. **What is the command to add remote reporting Git?**
5. **What is the command to rename files in Git? git mv <old\_name> <new\_name>**
6. **What is the command to ignore files in Git?** To ignore files in Git, you can create a **.gitignore** file and specify the files or patterns to be ignored

echo "filename.txt" >> .gitignore

1. **command to remove tracked files from the current working tree in Git? git rm <file>**
2. **What does the command Git config do?**

It can be used to configure various aspects of Git, such as user name, email, default editor, etc.

1. **What is the difference between Git revert and reset?**

Git revert adds a new commit to undo changes while keeping the history intact, whereas Git reset removes commits by moving the branch pointer backward, potentially rewriting history.

1. **Can you recover a deleted branch in Git**?

Yes, you can recover a deleted branch in Git as long as it was not pruned or garbage collected. When you delete a branch in Git, you're essentially deleting a reference to a commit, but the commits themselves still exist in the repository until they are garbage collected.

To recover a deleted branch, you can use the reflog, which is a log of all reference updates, including branch deletions.

1. **What is a conflict in Git and how to resolve it?**

A conflict in Git occurs when two branches have made changes to the same part of the same file, and Git cannot automatically reconcile those changes. To resolve a conflict, you need to manually edit the conflicted files to remove the conflict markers, stage the resolved files, and commit the changes.

**Git stash**: Allows you to temporarily store changes that are not ready to be committed, allowing you to work on other tasks or switch branches without committing incomplete work.

**Git stash apply**: Applies the most recent stash to your working directory without removing it from the stash list.

**Git stash pop**: Applies the most recent stash and removes it from the stash list.

**Git stash drop**: Removes the most recent stash from the stash list without applying it.

**Git annotate (also known as git blame)**: Shows who last modified each line of a file and when the modification was made, helping to trace the history of changes.

**Git cherry-pick**: Selects a single commit (specified by its commit hash) from one branch and applies it to another branch. This is useful for applying specific changes without merging entire branches.