**2.Git-HOL**

**1. Explain .gitignore**

The .gitignore file is used in Git to specify unwanted files or directories that Git should ignore. These files:

* Are not tracked by Git
* Will not appear in git status or be committed
* Are usually temporary, generated, or local configuration files (e.g., .log, node\_modules, \*.class, etc.)

**2. How to ignore files using .gitignore**

* Creating a file named .gitignore in the project root
* Adding patterns or filenames to it

\*.log # Ignore all .log files

log/ # Ignore entire log folder

temp.txt # Ignore specific file

build/\* # Ignore everything in build folder

**3.Git-HOL**

**1. Explain branching and merging**

**Branching** in Git allows you to create a separate line of development from the main project. It lets multiple developers work independently on features or fixes without affecting the main codebase.

**Merging** is the process of bringing the changes from one branch (e.g., a feature branch) into another (typically the master or main branch).

Benefits:

* Allows parallel development
* Isolates experimental work
* Enables easier collaboration and code review

**2. Explain creating a branch request in GitLab**

A branch request is not an official GitLab term, but it generally refers to:

* Creating a new branch from the GitLab web interface or via git commands
* This branch is then used to work on specific tasks or features

**In GitLab:**

1. Go to your repository
2. Click on Repository > Branches
3. Click New branch
4. Name your branch (e.g., GitNewBranch) and create it from master or another base

**3. Explain creating a merge request in GitLab**

A **Merge Request (MR)** in GitLab is like a "pull request" in GitHub. It's used to propose changes from one branch (e.g., GitNewBranch) to another (e.g., master) and review the code before merging.

Steps:

1. Go to your GitLab repo
2. Click **Merge Requests > New Merge Request**
3. Choose source branch (GitNewBranch) and target branch (master)
4. Add description, assign reviewers
5. Submit and wait for approval/merge

**4.Git-HOL**

**Explain how to resolve the conflict during merge.**

**Answer:**

When two branches modify the same line in a file or one branch deletes a file modified by another, a merge conflict occurs during Git merge. To resolve:

1. Identify the conflicting files using git status.
2. Open the conflicted file, and Git will highlight the conflict using:

<<<<<<< HEAD

Your changes from current branch

=======

Changes from the branch being merged

>>>>>>> branch-name

1. **Manually edit** the file to keep the correct content and remove Git conflict markers.
2. **Stage the resolved file**:

git add <filename>

1. **Commit the resolution**:

git commit -m "Resolved merge conflict"

**5.Git-HOL**

**Explain how to clean up and push back to remote Git**

**Answer:**

1. **Clean up local repo – remove untracked files and old merged branches:**

git clean -fd

git branch -d branch\_name

1. **Sync with remote – get the latest updates:**

git pull origin master

1. **Stage & commit changes:**

git add .

git commit -m "Your message"

1. **Push to remote:**

git push origin master