**Basic Git Questions**

**1)What is Git and why is it used?**

* Git is a distributed version control system used to track code changes.
* It allows multiple developers to collaborate on the same project.
* Enables tracking and reverting changes when needed.
* Provides branching and merging capabilities for parallel development.
* Improves code management and ensures version control.

2**)Explain the difference between Git and GitHub.**

* Git: A version control system used to track and manage source code changes.
* GitHub: A cloud-based platform for hosting Git repositories.
* Git is installed locally and can function without an internet connection.
* GitHub enables collaboration, code sharing, and remote repository management.
* GitHub provides additional tools like pull requests, issue tracking, and CI/CD integration.

**3)How do you install Git on your machine?**

* Windows: Download and install Git from [git-scm.com](https://git-scm.com/downloads).
* Mac: Install using Homebrew:

brew install git

* Linux: Install via package manager:

sudo apt install git

sudo yum install git

* Verify installation:

git --version

**4)How do you configure your username and email in Git?**

Set global username and email:

git config --global user.name "dileepmulagada"

git config --global user.email "dileepmulagada@puropalecreations.com"

Set username and email for a specific repository (run inside the repository folder without --global).

Verify configuration:

git config --list

**5)What is a repository in Git?**

* A repository is a storage location for a project’s files, history, and changes.
* Can be local (on a computer) or remote (hosted on GitHub, GitLab, etc.).
* Contains commits, branches, and tags to track changes and versions.

**6)How do you create a new Git repository?**

* Navigate to the project directory.
* Run the following command to initialize a Git repository:

git init

* Add files and create the first commit:

git add .

git commit -m "Initial commit"

**7)How do you clone a repository from GitHub?**

* Use the following command:

git clone <repository-url>

* To clone a repository into a specific folder:

git clone <repository-url> my-directory

**8)What is the purpose of the .gitignore file?**

* Prevents unnecessary files from being tracked by Git.
* Used to ignore files like logs, compiled binaries, and environment configurations.
* Example .gitignore file:

node\_modules/

.env

\*.log

**9)How do you check the status of your working directory in Git?**

* Use:

git status

* Displays:
* Untracked files that are not added to Git.
* Modified files that have been changed but not staged.
* Staged files that are ready for commit.

**10)How do you add files to the staging area in Git?**

* To add a specific file:

git add <filename>

* To add all files in the directory:

git add .

* Staging files prepares them for the next commit.

**Intermediate Git Questions**

**11)Explain the concept of commits in Git.**

* A commit represents a snapshot of the project's files at a specific moment.
* Each commit has a unique identifier (SHA hash) to track changes.
* Commits allow developers to save and document changes with messages.
* Multiple commits form the version history of a project.

**12)How do you create a new commit in Git?**

* Stage the changes using:

git add <filename>

* Create a commit with a descriptive message:

git commit -m "Your commit message"

**13)What is the purpose of the git log command?**

* Displays the commit history of a repository.
* Shows commit hashes, authors, timestamps, and messages.
* Helps in tracking changes and understanding the evolution of the project.

**14)How do you view the history of commits in a repository?**

* Use:

git log

* For a concise history:

git log --oneline --graph --all

**15)How do you view the changes made in a commit?**

* Use:

git show <commit-hash>

* Displays the modified files and their content differences.

**16)What is branching in Git and why is it useful?**

* Branching allows developers to work on new features without affecting the main codebase.
* Each branch is an independent line of development.
* Enables parallel development and easier collaboration.
* Helps manage different versions of a project efficiently.

**17)How do you create a new branch in Git?**

Use:

git branch <branch-name>

**18)How do you switch between branches in Git?**

Use:

git checkout <branch-name>

Or:

git switch <branch-name>

**19)What is the difference between git merge and git rebase?**

* Merge: Combines changes from one branch into another, maintaining commit history.
* Rebase: Moves or replays commits onto another branch, creating a linear commit history.
* Merge keeps all commit history, while rebase creates a cleaner, more linear history.

**20)How do you resolve merge conflicts in Git?**

* Identify conflicting files using git status.
* Open conflicting files and manually resolve differences.
* Stage resolved files using:

git add <filename>

* Complete the merge by committing the resolved changes:

git commit -m "Resolved merge conflict"

**Git Exercises**

**21)Create a new Git repository and configure your username and email.**

* Navigate to your project directory.
* Initialize a new Git repository:

git init

* Configure your username and email globally:

git config --global user.name "dileepmulagada"

git config --global user.email "dileepmulagada@puropalecreations.com"

* To verify the configuration:

git config --list

**22)Create a file, add some content to it, and commit the changes.**

* Create a new file using the command:

echo "Hello, Git!" > file.txt

* Add the file to the staging area:

git add file.txt

* Commit the file with a meaningful message:

git commit -m "Added file.txt"

**23)Create a .gitignore file and add rules to ignore specific files and directories.**

* Create a .gitignore file:

touch .gitignore

* Add rules to ignore specific files and folders:

echo "node\_modules/" > .gitignore

echo "\*.log" >> .gitignore

echo ".env" >> .gitignore

* Add and commit the .gitignore file:

git add .gitignore

git commit -m "Added .gitignore file"

**24)Clone an existing repository from GitHub and make some changes.**

* Clone the repository using:

git clone <repository-url>

* Navigate into the cloned directory:

cd <repository-name>

* Make some changes to a file and save them.
* Stage and commit the changes:

git add <modified-file>

git commit -m "Updated <modified-file>"

* Push the changes to the remote repository:

git push origin main

**25)Create a new branch, make some changes, and switch back to the main branch.**

* Create a new branch:

git branch feature-branch

* Switch to the new branch:

git checkout feature-branch

* Make some changes, then stage and commit them:

git add <modified-file>

git commit -m "Made changes in feature-branch"

* Switch back to the main branch:

git checkout main

* Merge the new branch into the main branch:

git merge feature-branch

* Delete the feature branch if no longer needed:

git branch -d feature-branch