**GIT (Global Information Tracker)**

1. **What is Git ?**

* **Git** is a **distributed version control system**
* that tracks changes to our files and allows collaboration among multiple developers working on the same project
* It helps you save, manage, and revisit any version of your project files.

1. **How Does Git Keep Track of Files?**

* Git keeps track of files by taking **snapshots** during commits.
* These snapshots are lightweight because Git stores only **deltas** for tracked files.
* Every snapshot is identified by a unique **commit hash**.

1. **How to track files with the help of Git?**

* **Git init**
* This creates a hidden. git directory in your project folder where Git stores all its tracking and versioning information.
* **git status** it will show the files that are not in track of git.

1. **What is the difference between git add and git commit?**

**git add**: Adds changes to the staging area, preparing them for a commit.

**git commit**: Saves the staged changes to the repository's history.

1. **What is a branch in Git, and why is it useful?**

* Iam developing a new feature for an application. To avoid disrupting the main branch (main), i create a branch named feature/login
* .Once the feature is complete and tested, you merge it back into the main branch.
* **git branch feature/login** # Create a new branch
* **git switch feature/login** # Switch to the branch

1. **What is the difference between git pull and git fetch?**

* **git fetch**: Downloads updates from the remote repository but does not apply them to your local branch**.( Like checking for new mail but not opening it.)**
* **git pull**: Fetches updates and merges them into your current branch**.( Like checking and opening the mail.)**

1. **How can you resolve merge conflicts?**

* Merge conflicts occur when two branches modify the same lines in a file. To resolve:
* Git will mark the conflict in the file.
* Edit the file to fix the conflict manually.
* Stage the resolved file and commit.

1. **What is .gitignore?**

**Example:**You’re packing for a trip but don’t want to take old, unnecessary items like junk files. .gitignore tells Git to skip them.

1. **What is the difference between git merge and git rebase?**

**git merge:** Like merging two rivers and keeping their separate paths visible.

**git rebase:** Like aligning everything into a single straight path.

1. **What is git stash, and how do you use it?**  
   git stash temporarily saves uncommitted changes so we can work on something else.
2. **What is git cherry-pick, and how is it used?**

**git cherry-pick** copies a specific commit from one branch to another.

1. **What is git reflog, and how can it help recover lost commits?**

git reflog shows a log of all actions performed in the repository, including branch movements and resets.

1. **How do you squash commits in Git?**

Squashing combines multiple commits into one

### Git and GitHub Detailed Documentation with Questions and Answers

---

### 1. \*\*What is Git?\*\*

\*\*Answer:\*\*

Git is a distributed version control system that allows multiple developers to track changes in files, collaborate on code, and manage different versions of a project efficiently. It works by storing snapshots of files and their history, allowing for easy rollback, collaboration, and version management.

---

### 2. \*\*What are the main features of Git?\*\*

\*\*Answer:\*\*

- \*\*Distributed Architecture\*\*: Each user has a full copy of the repository, making it resilient and fast.

- \*\*Branching and Merging\*\*: Enables parallel development through branching and merging.

- \*\*Lightweight Tags\*\*: Used to mark specific points in history (usually for releases).

- \*\*Staging Area\*\*: Allows users to prepare changes before committing them.

- \*\*High Performance\*\*: Optimized for handling large projects efficiently.

---

### 3. \*\*What is the difference between Git and GitHub?\*\*

\*\*Answer:\*\*

- \*\*Git\*\*: A version control system used to track changes locally on a machine.

- \*\*GitHub\*\*: A cloud-based platform for hosting Git repositories that adds collaboration features such as pull requests, issue tracking, and CI/CD integrations.

---

### 4. \*\*How do you check the version of Git installed?\*\*

\*\*Answer:\*\*

Run the following command in your terminal:

```bash

git --version

```

---

### 5. \*\*What is a repository in Git?\*\*

\*\*Answer:\*\*

A Git repository is a directory where Git tracks and stores all versions of your project's files, along with its history and configuration.

---

### Initial Setup

---

### 6. \*\*How do you initialize a Git repository?\*\*

\*\*Answer:\*\*

Use the command:

```bash

git init

```

This command initializes an empty Git repository by creating a `.git` folder in your project directory.

---

### 7. \*\*How do you configure a Git username and email?\*\*

\*\*Answer:\*\*

Use the following commands:

```bash

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

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

```

---

### 8. \*\*How do you clone a repository?\*\*

\*\*Answer:\*\*

Use the following command to clone a repository from GitHub or any remote server:

```bash

git clone <repository-url>

```

---

### 9. \*\*What is the .gitignore file?\*\*

\*\*Answer:\*\*

The `.gitignore` file tells Git which files or directories to ignore when committing. It helps in excluding temporary files, build outputs, or sensitive data.

---

### Staging and Committing

---

### 10. \*\*What is the staging area in Git?\*\*

\*\*Answer:\*\*

The staging area is a place where files are stored before they are committed to the Git repository. You can add files to the staging area using `git add`.

---

### 11. \*\*How do you commit changes in Git?\*\*

\*\*Answer:\*\*

Use the following command to commit changes:

```bash

git commit -m "Your commit message"

```

---

### 12. \*\*How do you check the status of a repository?\*\*

\*\*Answer:\*\*

Use the command:

```bash

git status

```

This shows which files are staged, modified, or untracked.

---

### 13. \*\*What is git diff used for?\*\*

\*\*Answer:\*\*

`git diff` shows the differences between the working directory and the staging area, or between commits.

---

### 14. \*\*How do you view commit history?\*\*

\*\*Answer:\*\*

Use the command:

```bash

git log

```

This shows the commit history in the repository.

---

### Branching and Merging

---

### 15. \*\*What is a branch in Git?\*\*

\*\*Answer:\*\*

A branch is a pointer to one of the commits in the repository. It allows developers to work on different features or fixes simultaneously without affecting the main project.

---

### 16. \*\*How do you create a new branch?\*\*

\*\*Answer:\*\*

Use the following command:

```bash

git branch <branch-name>

```

---

### 17. \*\*How do you switch branches?\*\*

\*\*Answer:\*\*

Use the command:

```bash

git checkout <branch-name>

```

Alternatively, you can use:

```bash

git switch <branch-name>

```

---

### 18. \*\*What is the difference between git merge and git rebase?\*\*

\*\*Answer:\*\*

- \*\*git merge\*\*: Combines the histories of two branches, preserving the commit history.

- \*\*git rebase\*\*: Re-applies commits on top of another branch, effectively rewriting history to make the history linear.

---

### 19. \*\*What is a merge conflict?\*\*

\*\*Answer:\*\*

A merge conflict occurs when changes made in two different branches overlap. Git cannot automatically merge them and requires the user to manually resolve the conflict before committing.

---

### 20. \*\*How do you delete a branch?\*\*

\*\*Answer:\*\*

- \*\*Local\*\*: `git branch -d <branch-name>`

- \*\*Remote\*\*: `git push origin --delete <branch-name>`

---

### Tags and References

---

### 21. \*\*What is a tag in Git?\*\*

\*\*Answer:\*\*

A tag is a reference to a specific commit in the repository, often used to mark a release point (e.g., v1.0).

---

### 22. \*\*How do you create a lightweight tag?\*\*

\*\*Answer:\*\*

Use the command:

```bash

git tag <tag-name>

```

---

### 23. \*\*What is HEAD in Git?\*\*

\*\*Answer:\*\*

`HEAD` points to the current branch reference in the repository, usually the most recent commit.

---

### 24. \*\*What is a detached HEAD?\*\*

\*\*Answer:\*\*

A detached HEAD occurs when Git points to a specific commit rather than a branch. This happens when checking out a commit directly instead of a branch.

---

### 25. \*\*What is git cherry-pick?\*\*

\*\*Answer:\*\*

`git cherry-pick` applies a specific commit from one branch to another branch, without merging the entire branch.

---

### Undoing Changes

---

### 26. \*\*How do you undo the last commit?\*\*

\*\*Answer:\*\*

- \*\*Soft reset\*\* (keeps changes staged):

```bash

git reset --soft HEAD~1

```

- \*\*Hard reset\*\* (discards changes):

```bash

git reset --hard HEAD~1

```

---

### 27. \*\*What is git stash?\*\*

\*\*Answer:\*\*

`git stash` temporarily saves changes in the working directory that are not ready for commit. You can retrieve them later using `git stash pop`.

---

### 28. \*\*How do you revert a commit?\*\*

\*\*Answer:\*\*

Use the following command to create a new commit that undoes the changes:

```bash

git revert <commit-hash>

```

---

### 29. \*\*What is git clean?\*\*

\*\*Answer:\*\*

`git clean` removes untracked files from the working directory. Use cautiously as it deletes files not tracked by Git:

```bash

git clean -f

```

---

### GitHub Basics

---

### 30. \*\*What is GitHub?\*\*

\*\*Answer:\*\*

GitHub is a platform for hosting Git repositories with additional features like collaboration tools, pull requests, issue tracking, and CI/CD pipelines.

---

### 31. \*\*How do you fork a repository on GitHub?\*\*

\*\*Answer:\*\*

Go to the repository's page on GitHub and click on the "Fork" button. This creates a personal copy of the repository under your GitHub account.

---

### 32. \*\*How do you create a pull request on GitHub?\*\*

\*\*Answer:\*\*

1. Fork the repository and make changes.

2. Go to the "Pull Requests" tab and click "New Pull Request."

3. Select the base repository and branch, then click "Create Pull Request."

---

### 33. \*\*What is GitHub Actions?\*\*

\*\*Answer:\*\*

GitHub Actions is a CI/CD platform that automates workflows like build, test, and deployment directly within GitHub repositories.

---

### 34. \*\*How do you secure your GitHub repository?\*\*

\*\*Answer:\*\*

- Enable branch protection rules.

- Use Two-Factor Authentication (2FA).

- Limit and review access permissions for collaborators.

---

### Advanced Git

---

### 35. \*\*What is git reflog?\*\*

\*\*Answer:\*\*

`git reflog` keeps track of all changes to the HEAD reference, including changes that are not part of the commit history.

---

### 36. \*\*How do you compare two commits?\*\*

\*\*Answer:\*\*

Use the command:

```bash

git diff <commit1> <commit2>

```

---

### 37. \*\*What is git blame?\*\*

\*\*Answer:\*\*

`git blame` shows which author made changes to each line in a file and when the change was made.

---

### Submodules and Hooks

---

### 38. \*\*What is a submodule in Git?\*\*

\*\*Answer:\*\*

A submodule is a repository embedded inside another repository. It is often used to manage dependencies or separate projects.

---

### 39. \*\*What are Git hooks?\*\*

\*\*Answer:\*\*

Git hooks are scripts that automatically run at certain points during Git's lifecycle (e.g., pre-commit, post-merge) to enforce rules or trigger actions like testing.

---

### 40. \*\*How do you set up a global .gitignore file?\*\*

\*\*Answer:\*\*

Create the file and configure it globally with:

```bash

touch ~/.gitignore\_global

git config --global core.excludesfile ~/.gitignore\_global

```

---

### Collaboration and Workflow

---

### 41. \*\*What is the purpose of git fetch?\*\*

\*\*Answer:\*\*

`git fetch` downloads changes from the remote repository without merging them into your local branch. It allows you to review changes before applying them.

---

### 42. \*\*How do you resolve a merge conflict?\*\*

\*\*Answer:\*\*

- Manually edit conflicting files.

- Stage the resolved files using `git add`.

- Commit the resolution with `git commit`.

---

### 43. \*\*What is the difference between SSH and HTTPS cloning in Git?\*\*

\*\*Answer:\*\*

- \*\*SSH\*\*: Secure and requires an SSH key for authentication.

- \*\*HTTPS\*\*: Requires a username and password for each interaction.

---

### 44. \*\*What is git bisect?\*\*

\*\*Answer:\*\*

`git bisect` helps identify which commit introduced a bug by performing a binary search through the commit history.

---

### 45. \*\*How do you squash commits?\*\*

\*\*Answer:\*\*

Use the following interactive rebase command:

```bash

git rebase -i <base-branch>

```

Choose "squash" to merge multiple commits into one.

---

### Advanced Branching Strategies

- \*\*Git Flow\*\*: A branching model with distinct branches for features, releases, and hotfixes.

- \*\*GitHub Flow\*\*: A simpler model with just main and feature branches, ideal for CI/CD.

- \*\*Trunk-based Development\*\*: Small, frequent merges directly into the main branch.

---

### Setting Up CI/CD Pipelines with GitHub Actions

GitHub Actions automates tasks like build, test, and deployment directly in GitHub. Features include caching, secrets management, and matrix builds.

---

### Handling Large Repositories with Git LFS

\*\*Git LFS\*\* is used to store large binary files like images and videos outside of the repository, tracking them with pointers.

```bash

git lfs install

git lfs track "\*.psd"

git add .gitattributes

```

---

### Automating Workflows Using Git Hooks

Git hooks trigger actions during Git operations. Examples include pre-commit checks or post-push notifications.

Example pre-commit hook:

```bash

#!/bin/sh

npm test

```

---

### Migrating Repositories to GitHub

1. Create a new repository on GitHub.

2. Add GitHub as a remote to your local repository:

```bash

git remote add origin https://github.com/username/repo.git

```

3. Push the repository:

```bash

git push -u origin main

```

---

### Practice Questions

1. \*\*What is the difference between git fetch and git pull?\*\*

\*\*Answer\*\*: Fetch downloads updates without merging, while pull downloads and merges updates.

2. \*\*How do you view a specific commit?\*\*

\*\*Answer\*\*: Use `git show <commit-hash>`.

3. \*\*What is the purpose of .gitignore?\*\*

\*\*Answer\*\*: To specify files and directories that Git should ignore.

4. \*\*How do you revert a commit?\*\*

\*\*Answer\*\*: Use `git revert <commit-hash>` to undo the changes.

5. \*\*How do you create a lightweight tag in Git?\*\*

\*\*Answer\*\*: Use `git tag <tag-name.