Version Control

Reading Material

PW SKILLS

Topics

- Understanding the concept
  - Version control system
  - Git
  - Github
  - SVN
  - Local Repository
  - Remote Repository
  - Git installation
  - Git setup
  - .gitignore file
  - Some important git commit
  - Staging in Git
  - Stashing in Git
  - Branching Strategy
  - Some advanced git commands
- Interview Questions
- Multiple Choice Questions

Understanding the Concept

Version Control System

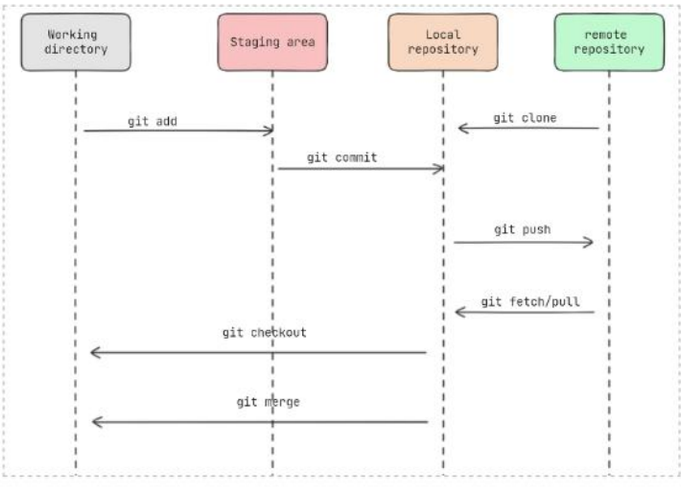
Version control is a system that records changes to a file or set of files over time. It allows you to track the history of changes, collaborate with others, and manage multiple versions of a project. Version control is commonly used in software development, but it can also be applied to any context where files need to be tracked and shared

The version control can be of

1. **Distributed Version Control System (DVCS) -** A Distributed Version Control System (DVCS) is a type of version control system that allows multiple copies of a repository to exist, each of which contains the full history of the project.
2. **Centralized Version Control System (CVCS) -** A Centralized Version Control System (CVCS) is a type of version control system where there is a central server that stores the main repository containing the entire history of the project.
3. **Local Version Control System -** A Local Version Control System is the simplest form of version control and is designed for individual use or small projects

Git

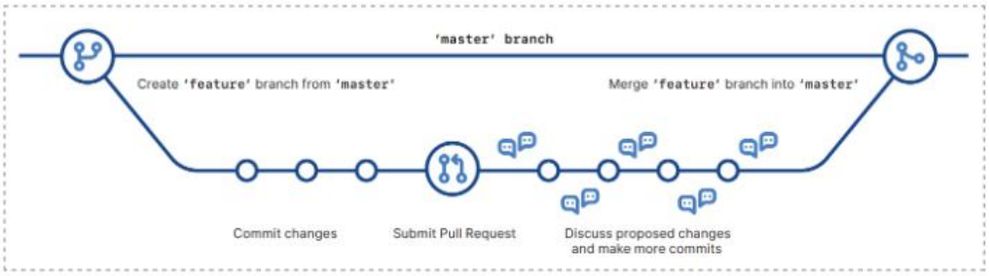
Git is a VCS, which means it helps developers manage changes to their code over time. It allows you to track modifications, collaborate with others, and maintain a history of the project.



The basic flow of Git

GitHub

GitHub is a platform where you can upload a copy of your Git repository (often shortened as repo), hosted either on github.com. More than just uploading your Git repositories it allows you to collaborate much more easily with other people on your projects. It does that by providing a centralized location to share the repository, a web-based interface to view it, and features like forking, Pull requests, Issues, Projects, and GitHub Wikis that allow you to specify, discuss, and review changes with your team more effectively.



GitHub Flow

SVN

SVN (Subversion) is a centralized version control system that allows multiple developers to collaborate on a project by storing files and their history on a central server. SVN simplifies the coding process, facilitates access management, and optimizes storage space by enabling developers to keep only the files they are working on locally and commit them to the server when ready.

In contrast, Git is a distributed version control system that utilizes multiple repositories, including a central repository and local repositories. Git's workflow involves creating local repositories that mirror the central repository, allowing developers to work on their local copies and push changes to the central repository when ready

Local Repository

A local repository in the context of software development typically refers to a Git repository that is stored on your local machine. it is used for version control features, but collaboration features like pulling and pushing code changes with teammates can only be done on a remote repository.

Remote Repository

A remote repository, also known as a remote in version control systems like Git, is a copy of your project that's stored on the internet or another network. It acts like a central location for your codebase, allowing for collaboration and version control.

Git installation

For the installation of Git on any system, certain specific instructions should be followed depending on a particular operating system-

Windows -

- Visit the official Git for Windows website i.e. **Link**
- Download the latest installer (64-bits is recommended)
- Run the installer and follow the installation wizard
- During the installation, one can choose various configuration options. The default settings are usually fine for most users.

Mac -

For installing it Mac OS Homebrew should be installed from the official Homebrew site i.e. **Link**
After installing the Homebrew run the following command below to install the git.

```
brew install git
```

Git setup

Setup or Configure your user information used across all local repositories

```
// user name
git config --global user.name “[firstname lastname]”

// user email
git config --global user.email “[valid-email]”
```

.gitignore file

The .gitignore file is a text file used in Git repositories to specify which files or directories should be ignored by Git. It is typically placed in the root directory of a project and contains patterns that match files or directories to be excluded from version control.

Example of the .gitignore file

.gitignore

```
# dependencies
/node_modules
/.pnp
.pnp.js
# testing
/coverage
# production
/build
```

Some important git commit

```
// 1. git init - initialize a new Git repository in an
existing project.
git init

// 2. git clone - make a copy of a remote Git repository on
a local machine
git clone

// 3. git status - check and display the current status of
the project
git status

// 4. git add - add new files to begin tracking
git add [file]
// 5. git commit - used to save staged changes with unique
hash
git commit -m [descriptive commit message]
```

```
// 6. git push - save all commit chanes to remote
repository
git push [alias][branch] // alias can be origin or remote

// 7. git branch - view all available branch or create
branch
git branch  // viea all branch
git branch [branch-name] // create branch with provided
branch name

// 8. git checkout - switch to another branch
git checkout [branch-name]

// 9. git log - view commit history
git log

// 10. git remote add - ad a new remote repository
git remote add [alias][remote-repo-url]

// 11. git fetch - retreive updates from remote repo
without merging
git fetch

// 12. git pull - retrieve and merge update from remote
repo
git pull

// 13. git help - display various information about git
commands
git help [any-git-command]
```

Staging in Git

Staging refers to preparing changes in your working directory to be included in the next commit. When you make changes to files in your Git repository, these changes are initially considered to be in the "unstaged" state. Staging involves selectively choosing which changes you want to include in the next commit.

It can be done with the **git add** command

```
// to add specific file
git add [file]
// to all change files
git add
```

Stashing in Git

Stashing refers to temporarily saving your work in progress, including both staged and unstaged changes, on a stack of unfinished changes that you can reapply later.
Stash can be done with the following commands -

```
// save modified and staged changes
git stash

// list all stash file changes
git stash list

// get working from the top of the stash
git stash pop

// clear stash
git stash drop
```

Branching strategy

A branching strategy in Git is a set of guidelines that define how developers create, manage, and merge branches within a project. It essentially establishes a workflow for collaborating on the codebase and keeping it organized.

Branching comments include -

```
// create branch
git branch [branch-name]

// list branch
git branch

// checkout to specific branch
git checkout [branch-name]

// merge specific branch to the current branch
git merge [specific-branch]
```

Some advanced git commands

```
1. rebase  - rebases the current branch onto another branch
git rebase [branch-name]

2. cherry pick - apply specific commits to the current
branch
git cherry-pick [commit-hash]

3. bisecting - Identifies the commit that introduced a bug
using a binary search approach.
git bisect start    // starts a bisect session
git bisect bad      // marking the current HEAD as bad
git bisect good [commit-hash] // marks as know good commit

4. reflog - list all the commits you've visited in the past
git reflog

5. blame - show commits information of each line of a
specified file
git blame [file-name]

6. tagging - create tag name for a specified commit
git tag [tag-name] [commit-hash]
git tag -1  // list all tags in your repo
```