Version Control System (VCS)

Version control systems (VCS) are essential tools for developers to manage and track changes in their codebase over time. They facilitate collaboration, accountability, code management, and disaster recovery.

Popular VCS

Git, Subversion (SVN), Mercurial (Hg), Microsoft Team Foundation Server (TFS), Bitbucket, Perforce

Distributed VCS

A distributed version control system (DVCS) mirrors the complete codebase, including its full history, on every user's local machine, allowing for collaboration without a central server. Git is the most popular DVCS.

Git

Git is a distributed VCS developed by Linus Torvalds in 2005. It allows multiple developers to work on a project simultaneously by tracking changes, managing branches, and enabling collaborative development.

Setup

1. Install Git and verify the version: $ git --version

2. Configure username and email:

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

$ git config --global user.email "Email Id"

GitHub

GitHub is a web-based hosting service for Git repositories. It facilitates version control, collaboration, hosting, social coding, project management, integration, and more.

SSH Key

SSH keys are pairs of cryptographic keys used for secure communication between a client and a server. The public key is stored on the server, while the private key is kept by the client.

SSH Key Setup

1. Generate a new SSH key: $ ssh-keygen -t ed25519 -C "your_email@example.com"

2. Ensure the ssh-agent is running: $ eval "$(ssh-agent -s)"

3. Add your SSH private key to the ssh-agent: $ ssh-add ~/.ssh/id_ed25519

4. Copy the SSH key to your clipboard: $ clip < ~/.ssh/id_ed25519.pub

5. Add the SSH key to your GitHub account.

6. Test the SSH connection: $ ssh -T git@github.com

3-Tier Architecture in Git

1. Staging Area: Prepares your code for commit.

2. Local Repository: Saves your committed changes locally.

3. GitHub: Hosts your code for collaboration and backup.

Basic Git Commands

1. Check repository status: $ git status

2. Initialize a Git repository: $ git init

3. Stage files for commit:

- Specific file: $ git add file.txt

- All files: $ git add .

4. Make a commit: $ git commit -m "Commit Message"

5. View commit history: $ git log

Uploading Files in GitHub Repositories

1. Remove a file from the staging area: $ git rm --cached <filename>

2. Remove all files from the staging area: $ git rm -r --cached .

3. Check available branches: $ git branch

4. Rename current branch: $ git branch -m new-branch-name

5. Push changes to GitHub: $ git push origin <branch_name>

Branching

Branches represent independent lines of development.

1. Create a new branch: $ git branch branch-name

2. Switch to a branch: $ git checkout branch-name

3. Create and switch to a new branch: $ git checkout -b "branchname"

.gitignore File

A .gitignore file specifies which files and directories Git should ignore and not track, preventing them from being included in version control.

Cloning

1. Clone a repository: $ git clone <repository-url>

2. Clone a specific branch of a repository: $ git clone -b branch-name <repository-url>

Collaboration

Collaboration in Git involves working with others on a shared codebase, allowing multiple developers to contribute to a project. Git provides several features to facilitate collaboration among team members.

Fetch, Pull, Merge

1. Retrieve changes from the remote repository: $ git fetch

2. Merge changes onto your local repository: $ git merge

3. Pull changes from GitHub: $ git pull

Basic Commands for Practice

1. git init

2. git status

3. git add

4. git commit

5. git push

6. git clone

7. git pull