Lecture_4

Version Control Systems

- Version control systems are software tools that help software teams manage changes to source code over time - Allow you to efficiently track the history of a collection of files and revert to the previous or more recent version of the files. - Each version captures a snapshot of files at a certain point in time

Repository

- working directory + store - It contains a collection of files that make up the project - The working directory contains a copy of the project files that is private to the developer

Repository Store

- The store contains the complete history of the project - hidden file, `.git`, is the store - The rest is the working directory

CVCS vs DVCS

- The Centralized version control system each person works on the servers repository. - More difficult to integrate - Easier to administrate and control backups, access and progress - The Distributed version control system each person has a repo that they can push to the main repository, the servers repository. - Safer for the server - Lighter direct load on the server - Offline work - Faster

Git

- A distributed version control system - Working directory Local copy of the repository where the user can make changes locally - Staging index It is the place where a commit will be prepared. We can add changes in the index. - Project History Displays the history of revisions to the project - Commit is a atomic collection of changes to a repository. - Contains all recored local modifications that lead to the new revision - Contains info such as; Commit id, Commit branch, etc - Branch represents an independent line of development - Serve as an abstraction for the edit/stage/commit process - Basically a brand new working directory, staging area, and project history - Head Pointer is the pointer to the tip (last commit) of a branch and it is the parent of the next commit

Git Commands

- `git clone ` - Copies the repository to the working directory - `git add ` - Make changes and add them to the staging area - `git commit -m ` - Allows you to commit two changes 'e' and 'f' - A new revision is created in your repository based on the state of the working directory - Should have a short descriptive message attached - `git pull ` - The command allows you to retrieve changes from a remote repository into your local repository - Finish your work and commit before pulling - `git push` - A push propagates changes from a local repository - `git merge ` - The command lets you take independent lines of development - `git log` - List of all commits made to a repository - `git revert` - revert a commit - Atomic commits are small changes

Remote

- A Remote the place where your code is stored

Merging Actions

- The process of joining two branches into one - Usually merging the last changes into your working directory - Sometimes there are conflicts between these changes, the user is usually prompted to interactively resolve them - After you merge you still need to commit your changes

Branching

- Supports quality assurance/ code quality/ and integration - Isolate bugs - ex. `git branch BugFix`