VCS

Created	@May 28, 2022 11:59 AM
Class	
Type	
Materials	
Reviewed	☐

What is VCS ?

- VCS stands for Version Control System. But what exactly it does ?
- As the name suggests one of the primary use case is to manage versions of your software that you've written. Whenever we add a new feature to our application, technically that's a new version of our application.
- So managing from the very smallest to the very biggest change in versions everything can be done by VCS.
- When I say manage, then I mean, we can add new version, we can even rollback to old versions, remove few versions as well.
- Management of versions is kinda like the primary use case for VCS. But we have other tasks also that it achieves.
- VCS provides us a sophisticated mechanism to work collaboratively. Let's say two or more developers are working on the same file.

Different version control systems are:

- GIT
- Mercurial
- SVN

In a nutshell, VCS is just going to be a software tool, that will help us to manage our software development lifecycle.

GIT

GIT is a distributed version control and source code management system. It is designed by Linus Torvalds. It helps software developers to work together and maintain a complete history of their work.

- It is free.
- It is open source.
- It works both offline and online.
- It’s widely accepted and kinda the industry standard.
- Manages backup and security.

Technically there are two things that git does

- version control management
- source code management

How to do version control management with GIT ?

The main things we need to start with for version management of a project is to initialise GIT in the project.

In whatever folder your project is, we need to initialise git there.

In order to initialise GIT we use the command

```
git init
```

Whenever we use the above command to initialise a new git project, we get a `.git` folder. It is a hidden folder, to locate it we can do `ls -a` in the terminal pr git bash.

Stages in git

In git you have multiple stages throughout the life cycle of project. We can use the command

```
git status
```

to actually get the status of the current stages of the files.

- Untracked: If any of your project is in Untracked status, then it means that GIT is currently not handling the versions of that file.

To shift our file from `untracked` stage to a stage where git starts managing it we can use the command

```
git add <filename>
```

- Staged: When we do `git add <filename>` our file gets into the new state called as staging. Here if we will make any changes to the file, git will track it. The moment we make some changes git will tell us that there are some changes that are not staged, for adding all the changes to staged state again we need to re-execute `git add <filename>`

Now let's after making some changes to the file, we added it to staging area by doing `git add <filename>` now we want to register the changes as a new version of the project.

For this we need to commit. Commit means a snapshot or I should say registration of a version. So every commit is a new version. Every commit has a message to explain the version

```
git commit -m "First commit"
```

The moment we do commit all of our files go from staging area to the state called as committed.

- Committed: In this state files can only come from staging state and when a file comes here all the changes in the file are registered as a version. Commit is also

called as version.

If we want to see all of our commits we can do

```
git log
```

The above command gives you a prompt which has a list of all the commits. If you want to exit the prompt press `q`

After adding a file to a commit, if we make more changes to the file, it will be again marked modified. So we can do more changes and after the changes we can commit the changes again. It will be added as a new commit.

If we want to add multiple files together in the staging area, we can do

```
git add .
```

If we forgot to add all the changes to the files in the last commit and we want to modify the commit we can do

```
git add <filenames>
git commit --amend
```

This will add the new modified changes to the last commit without creating a new commit. It might show you a vim/nano prompt , for if you want to make any changes to the commit message. You can exit out of the prompt using `esc` and then `:wq` (if you're using vim)