CSO: Recitation 2

Goktug Saatcioglu

09.12.2019

1 Understanding Git

Before we start using GitHub we need to understand Git first.

1.1 What is git anyways?

Git is a version control system that keeps track of changes to a given project and helps coordinate changes many people might do across multiple machines on the same (initial) set of project files (you can add more files to your project too). Almost all professional software developers must know Git or a variant of git (for example, Amazon uses an internal variant of git for their own projects) to successfully create, deploy and maintain software. Git is not GitHub. Git is the techology (version control software) we use to do version control and Github is a service that hosts git repositories which offers all the functionality git offers. So GitHub is the place that holds your git repositories and you upload to GitHub.

1.2 Why git?

1. Makes it easier to develop software in a non-linear way: Different people can work on different components of a project at their own pace. You don't need to build an application in a specific way as everyone can work on different parts. If you ever discover a bug later down in the line you can always return to a version of your software which you know was working without much hassle.

2. Distributed development: Multiple people can work on the project at the same time. For example, if we are creating a new clock app someone can work on creating a timer functionality while someone else can work on alarms at the same time. We can even work on the same file at the same time and git will try to bring together two different versions of the file into one (which most of the time works very well!).

3. Backups, backups, backups: git will save a history of previous version of your work. This history is very powerful as you can almost always undo a mistake you made in your code by using git. You can even undo mistakes you made while using git by using git! Even if you delete a file git will save a copy of it for a while. So if you decide you want that file back you can actually get it back.

1.3 Git in Detail

We can think of git as four different boxes. These boxes are called: current copy, the local repo, the stage/index, and the remote repo. Each box contains a state of the project which you are working on. In detail:

- Current copy (aka. the working tree): The current copy is the box where you are currently working on your project. It is also known as the "untracked" area of git as none of your changes done here will be tracked by git. If you make changes and do not save them using git then you will not be able to recover them if you lose them. The command `git status` will show you the status of the current copy. It'll tell you waht files are untracked which means git is not keeping track of them yet.

- Stage/index: The staging or index area is where git starts keeping track of your work or changes to existing files. You tell git which files or changes to files or removal of files you want to keep track of and git will move these files from the current copy to the staging (index) area. If you move a file to staging, then make more changes to it you must tell git that you would like to move it to staging again so that it can keep track of the latest changes. Using `git status` we see the untracked files and we can move them to stage by doing `git add #filename` where `#filename` is the file you would like to track. If you want to add all untracked files to stage do `git add .`.

- Local repo (repo means repository): Git stores a hidden directory named `.git` in the folder where you are working on a project. Never delete this or you will lose all your changes! The local repository consists of a history of checkpoints or snapshots of a program at a given point. These checkpoints capture the current state of the program at the point they were recorded in. Git refers to them as commits. So how do we take a snapshot? Well we tell git to make a commit using the `git commit` command. This command takes all the files in the staging area and turns the new state of the program into a snapshot and saves it in the local repo. Thus, a commit is basically a checkpoint that tells git what changes were made to get to the state it is in right now and what the resulting state of the project is after commiting.

- Remote repo: Well we have a local repository and we can make changes and save those changes but how can we have other see our changes? In fact, how can we have also become aware of other people's changes? After all we would like multiple people to work on the same project at the same time. This is where the remote repo comes into play. The remote repo can be considered as a global history of commits (changes) people make to a project. So when starting a project you want to either initialize a remote repo or clone one from GitHub by doing `git clone #repo_link` where `#repo_link` is the GitHub link to your repo. Once you are done committing your changes and want others to be aware of your work you should use `git push` to push your changes to the remote repo. If at any point you want to obtain changes other people might have done to the remote repo run `git pull` to pull new commits that your local repo is not aware of. Thus, the remote repo is the one central place in git where everyone's changes are tracked so that at the end of our project we will have all of the pieces in one place and ready to go.

1.4 Exercise

Go to here and follow the instructions.