**NTRODUCTION**

Hello developers,

Welcome to our first session at the Full Stack Web Developer Nanodegree.

My name is Andre and I'll be your mentor.

For today, Udacity requested me to prepare a short teaching session for you introducing a topic you will learn during the Nanodegree.

But, since it's our first time together, let me introduce myself.

When I'm not mentoring for Udacity, I am a full stack developer working with some cool stuff like Python, NodeJS, ReactJS and Blockchain creating business solutions for a really complex enterprise architecture, working with new and legacy systems hosted on Cloud and Mainframe.

well, you might be wondering how we manage all changes in our source codes.

That's why, I decide to cover one of my favorite topics in the Nanodegree. Version Control.

We will talk about version control systems, git and then I'll show you some tips about GitHub.

**AGENDA**

So today we will cover Version Control, Git and a bit about GitHub

First, I'll give you a brief explanation on what is a VCS and its importance

then we will talk about Git how it works, and I'll show you some terminologies and a quick guide to starting using it.

Finally, we will talk about GitHub, what is GitHub, and the greatest functions it has

I hope you enjoy the trip.

**VERSION CONTROL**

You might already know a bit about version control.

But if you don't, don't worry, a Version Control System is just software that helps you control (or manage) the different versions of something, like a source code.

You’ve probably created your own version control system without realizing it. Got any files like this?

MyResumeOct2015.doc

MyResumeResumeMar2016.doc

logov3.png

logov4.png

logo-old.png

That’s why we use “Save As”. We want the new file without change the old one. It’s a common problem, and solutions are usually like this:

Make a single backup copy (Document.old.txt).

If we’re clever, we add a version number or date: Document\_V1.txt, DocumentMarch2015.txt

We may even use a shared folder so other people can see and edit files without sending them over email. Hopefully they relabel the file after they save it.

So Why Do We Need A Version Control System (VCS)?

Our shared folder/naming system is fine for class projects or one-time papers. But for software development? No way, it's almost impossible to work that way and too risk.

Large, fast-changing projects with many authors need a Version Control System to track changes and avoid general chaos. A good VCS does the following:

* Backup and Restore. Files are saved as they are edited, and you can jump to any moment in time. Need that file as it was on Feb 23, 2007? No problem.
* Synchronization. Letting people share files and stay up-to-date with the latest version.
* Short-term undo. Monkeying with a file and messed it up? (That’s just like you, aren’t?). Throw away your changes and go back to the “last known good” version in the database.
* Long-term undo. Sometimes we mess up bad. Suppose you made a change a year ago, and it had a bug. Jump back to the old version, and see what change was made that day.
* Track Changes. As files are updated, you can leave messages explaining why the change happened (stored in the VCS, not the file). This makes it easy to see how a file is evolving over time, and why.
* Track Ownership. A VCS tags every change with the name of the person who made it.
* Sandboxing, or insurance against yourself. Making a big change? You can make temporary changes in an isolated area, test and work out the kinks before “checking in” your changes.
* Branching and merging. A larger sandbox. You can branch a copy of your code into a separate area and modify it in isolation (tracking changes separately). Later, you can merge your work back into the common area.

There are a number of Version Control Systems out there. Three of the most popular version control systems are:

* Git
* Subversion
* Mercurial

There are two main types of version control system models:

the centralized model - all users connect to a central, master repository

the distributed model - each user has the entire repository on their computer

So now we know that Version Control (aka Revision Control aka Source Control) let us track our files over time.

And why is it important? Well, resuming it in one sentence, I can say:

"When we mess things up, we can easily get back to a previous working version."

**GIT**

During the Nanodegree, we'll be using Git which is a distributed version control system.

Git was first developed by Linux Kernel Creator, Linus Torvalds.

Git is a distributed VSC, so there is no singular centralized code base that the code can be pulled from, as a fast and efficient system, many developers and open-source projects use Git to power their repos. Git is a superstar of the VCS.

So Let’s check a bit about (Show the git guide and the command line)

**GITHUB**

And what about GitHub, what is this? As a developer, you at least had heard of it once. We always hear about it, BTW Microsoft paid 7.5 BI $ on it, but don't worry, nothing changed on its policies.

So, what is GitHub?

You may have heard that GitHub is a code sharing and publishing service, or that it’s a social networking site for programmers. Both statements are true, but neither explain exactly why GitHub is special.

GitHub is a Git repository hosting service, but it adds some of its own features.

While Git is a command line tool, GitHub provides a Web-based graphical interface.

It also provides access control and several collaboration features, such as wikis and basic task management tools for every project.

The flagship functionality of GitHub is “forking”

Forking is the ability to copy a repository from one user’s account to another.

This enables you to take a project that you don’t have write access to and modify it under your own account.

If you make changes you’d like to share, you can send a notification called a “pull request” to the original owner.

That user can then, with a click of a button, merge the changes found in your repo with the original repo.

These three features – fork, pull request and merge – are what make GitHub so powerful.

So, before we continue let's wrap up the basics about GitHub:

GitHub is a code hosting platform for collaboration and version control.

GitHub lets you (and others) work together on projects.

Finally, GitHub has its own features that put the collaborative work in a whole new level.