Introduction to Source Control using Git and Github

What Is Source Control?

Source control refers to tracking and managing changes to code in a repository. It is a system that lets you manage changes to files and projects and is considered by many to be the backbone of professional software development due to its daily use by developers. Thus, understanding what it is, how to use it, and common terminology associated with its use is fundamental in the success of any developer. Source control has several names and is also called version control or source control management. It's also occasionally referred to as “revision control,” although the term is rarely used. Instead, you will predominantly hear it called “source control” and “version control".

These systems allow developers to track the evolution of their code, collaborate with others, and revert to previous versions of code if needed. They help to ensure that changes made by multiple developers don't conflict with each other and allow for better organization and management of code changes.

Why Do We Need Source Control Systems?

- Streamlining the development process as multiple people can work simultaneously on a single project.
- Management of code for multiple projects.
- Version control provides access to the historical versions of a project. This is insurance against computer crashes or data loss. If any mistake is made, you can easily roll back to a previous version. So it helps in recovery in case of any disaster or unforseen/contingent situation.

Types Of Source Control Systems

- Distributed Version Control: Mercurial, **Git** and Bazaar.
- Centralized version control systems: Concurrent Versions System (CVS), Subversion (or SVN) and Perforce.

What Is Git?

Git is a free, open-source distributed version control software. It was created by Linus Torvalds in 2005. This tool is a version control system that was initially developed to work with several developers on the Linux kernel. Git was designed to handle everything from small to very large projects with speed and efficiency. Git is installed and maintained on your local system (rather than in the cloud) and gives you a self-contained record of your ongoing programming versions. Real life projects generally have multiple developers working in parallel. So they need a version control system like Git to make sure that there are no code conflicts between them. Compared to other version control systems, Git is responsive, easy to use, and inexpensive (free).

There are three popular Git hosting services: **GitHub** (owned by Microsoft), GitLab (owned by GitLab) and BitBucket. Others include: Perforce, Beanstalk, Amazon AWS CodeCommit, Codebase, Microsoft Azure DevOps, SourceForge, Gerrit, etc.

When working with Git, it's important to be familiar with the term repository.

A **Repository** is a central location in which data is stored and managed. Each Git project is called a repository, or “repo” for short. A repo stores all the files and changes made to your project. It’s like a project with memory allowing you to move back and forward in time and observe the ways you have changed and modified your project. Thus, the repo stores data and changes information. We can single out two major types of Git repositories:

- Local repository - an isolated repository stored on your own computer, where you can work on the local version of your project.

- Remote repository - generally stored outside of your isolated local system, usually on a remote server. It's especially useful when working in teams - this is the place where you can share your project code, see other people's code and integrate it into your local version of the project, and also push your changes to the remote repository.

What is Github?

GitHub is a web-based hosting service for Git repositories. GitHub was launched as a company in 2008 and acquired by Microsoft in 2018. Nowadays GitHub is the largest online storage space of collaborative works that exists in the world. GitHub has become the go-to hosting platform for projects using Git.

Setup and Installation steps

1. Download Git

Windows: https://gitforwindows.org/

Linux: https://git-scm.com/download/linux

Mac: https://git-scm.com/download/mac

2. Create Your Github Account

https://github.com

3. Configure GitHub Credentials

The global git username and email address are associated with commits on all repositories on your system

Configure your local Git installation to use your GitHub credentials by entering the following:

```
>> git config --global user.name "yourGithubUsername"

>> git config --global user.email "yourGithubEmail"
```

When you create a new project and initialize it as a local github repo, by default it will be set to master as the default project branch whereas your remote repo might be set to main and this might cause issues.

So to configure the initial branch name to use main in all of your new repositories, run in terminal

```
>> git config --global init.defaultBranch main
```

4. Once your github account is ready and you've configured your github credentials. Then you can create a project (a folder containing a file) in your local computer.
5. Next, create a repository on Github where you'd want to push your local project to. It is advisable not to enable the *readMeFile* when creating your github repo.
6. Go to the directory of the local project you created earlier using your terminal and run the following commands:

*** Ensure you are in the directory of the project you want to push to your repo before you run the following.**

```
>> git init

>> git status

>> git add .

>> git commit -m "first commit to github"

>> git remote add origin remoteRepoProjectLink
```

```
>> git push -u origin main
```

If it's your first time pushing to github, a dialog might appear for you to setup your access token if you haven't or if it has expired

* To setup your personal access token

- Login to your account
- Settings
- Developer Settings
- Personal Access token
- Generate token
- enable: repo (all options), workflow, write:packages, delete:packages
- Do well to save your key somewhere
- Login using your key

Notes:

- git init: By default, any directory on our computer is not a Git repository – but we can turn it into a Git repository by executing this command. This will create an empty git repository or reinitialize an existing one.
- git status: The first thing you need to do is to check the files you have modified. To do this, you can type the following command to make a list of changes appear and know which branch that is being tracked.
- git add: The git add command adds new or changed files in your working directory to the Git staging area. It tells Git that you want to include updates to a particular file in the next commit. However, git add doesn't really affect the repository in any significant way—changes are not actually recorded until you run git commit.
- git commit: After adding files of our choice, we need to write a message to explain what we have done. This message may be useful later if we want to check the change history. Basically, this helps to record changes to the repository.

- git push: Now we can put our work on GitHub. To do that we have to ‘push’ our files to Remote. Remote is a duplicate instance of our repository that lives somewhere else on a remote server.