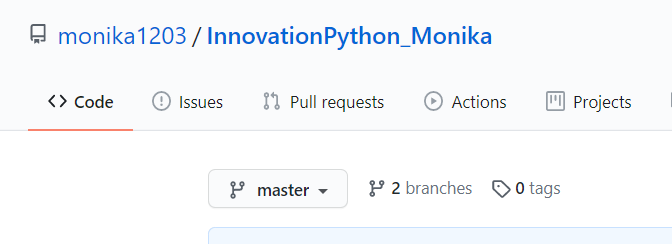
Git Theory Task ---------------------------Submitted by Monika Pawar ---------------------------------------

1. Make a repository on GitHub with the name **“InnovationPython\_yourname”**

****

1. Read about the difference between Git and GitHub?

**Git** is a version control software which should be install on local system in order to use it. Git enables developers to save snapshots of their projects over time. It’s generally best for individual use.

**GitHub** is a Web-based Git version control repository hosting service. It provides all of the distributed version control and source code management (SCM) functionalities of Git while topping it with a few of its own features.

1. Read about Git Workflow

**Git workflow is** reference to know how to use git to do work in a reliable and productive manner. There are many possible numbers of workflows with Git and these workflows are guidelines rather than concrete rule, which can implement according to the particular project team’s culture. Its best to choose workflow which enhances the effectiveness of team and not a burden that limits productivity. When working with a team on a Git managed project, it’s important to make sure the team is all in agreement on how the flow of changes will be applied. To ensure the team is on the same page, an agreed upon Git workflow should be developed or selected.

The **Centralized Workflow** is essentially a building block for other Git workflows and it uses a central repository to serve as the single point-of-entry for all changes to the project. Here, Master Branch is the default development branch and all changes are committed into this branch. Most popular Git workflows will have some sort of centralized repo that individual developers will push and pull from.

Some other common workflows are:

**Feature Branching, Gitflow Workflow, forking workflow**

Key points of Git workflow are:

* There is no one-size-fits-all Git workflow
* A workflow should be simple and enhance the productivity of your team
* Your business requirements should help shape your Git workflow

1. How many types of version control systems are there?

The version control systems are broken down into two main categories:

1. **Centralized** 🡪 In centralized source control, there is a server and a client. The server is the master repository and this is the only repo which contains all of the versions of the code and different branches of the code. So, the basic workflow involves in the centralized source control is getting the latest version of the code from a central repository that will contain other people code as well, making your own changes in the code and then committing or merging those changes into the central repository.
2. **Decentralized** (also known as distributed) 🡪 In distributed version control most of the mechanism or model applies the same as centralized. The only major difference you will find here is, instead of one single repository which is the server, here every single developer or client has their own server and they will have a copy of the entire history or version of the code and all of its branches in their local server or machine. Basically, every client or user can work locally and disconnected which is more convenient than centralized source control and that’s why it is called distributed.
3. Explain Branching concept in Git.

A **Git branch** is essentially an independent line of development. One can take advantage of branching when working on new features or bug fixes etc. because it isolates their work from that of other team members. **Branching** allows each developer to branch out from the original code base and isolate their work from others. This method allows others to easily identify what changes to expect and also makes backtracking simple. Changes in the primary branch or other branches will not affect your branch, unless you decide to pull the latest changes from those branches. Different branches can be merged into any one branch as long as they belong to the same repository.

1. Explain Forking Workflow in Git

Creating a “**fork**” is producing a personal copy of someone else's project. Forks act as a sort of bridge between the original repository and personal copy. By submitting Pull Requests, it helps make other people's projects better by offering your changes up to the original project. **Forked** **repositories** are created using the standard git clone command

The **Forking Workflow** is a bit different than other workflows. Because it gives every developer their own server-side repository. The Forking Workflows are more commonly seen in open-source projects.

The **main advantage** of the Forking Workflow is that contributions can be done without the need for everybody to push to a single central repository. Developers push to their own server-side repositories, and only the project maintainer can push to the official repository. This allows the maintainer to accept commits from any developer without giving them write access to the official codebase.