**Git Important Topics & Explanations**

1. What is Git?

Git is a Version Control System and a tool that helps to track changes in code. It is popular, free & open source, fast & scalable.

1. Features of Git.
   1. Track the changes.
   2. Collaborate
2. What is GitHub?

A website that allows developers to store and manage their code using Git.

1. Git Branching Strategies follows in the Tech Industry.

There are 4 different type of branches –

* 1. Master/Main branches – This is the default branch available in the repository.
  2. Feature branches – We can take a branch from Main/Master branch and do the development for the feature.
  3. Release branches – After every feature added to the main branches in every 1-3 months a company had introduced some Releases which can be called as versions. So, from the main branch a new branch gets created that is Release branch.
  4. Hotfix/Patchfix branches – This happens mostly after the releases when some user might have reported any issue/bug. Developer has to fix that bug, so far that we create Hotfix/Patchfix branches.

1. Give the configuration details.
   1. After installing Git, first thing that we must do is config.
   2. To config the username: **git config --global user.name “YOUR\_NAME”**
   3. To config the user email: **git config --global user.email “YOUR\_EMAIL”**
   4. Here, we give the --global, which means we are globally set the username and email address for every repository in our Local system (laptop/desktop). If you don’t want to set the username and user email address globally, and want to change the username and email for each repository in your Local system, just remove the --global, the new command will be:
      1. **git config user.name “YOUR\_NAME”**
      2. **git config user.email “YOUR\_EMAIL”**
   5. To check the configuration details:
      1. **git config --list**
      2. This will give the details of your git configuration with the user.name and user.email as well.
2. Cloning & Status.
   1. Clone – Cloning a repository on our local machine
      1. **git clone <GITHUB\_REPOSITORY\_URL>**
      2. It will create a new directory with all the codes available in the GitHub.
   2. If you don’t want to create a new directory and just want the repository files/directories.
      1. **git clone <GITHUB\_REPOSITORY\_URL> .**
   3. To check the status of the Local repository.
      1. **git status**
      2. The purpose of this command is to check if you made any changes in your code, is it untracked, or moved to staged area, or is it yet to be committed. Basically, it shows the status of the code.
      3. We have 4 types of status.
         1. Untracked – New files that git doesn’t yet tracked.
         2. Modified – Changed files.
         3. Staged – File is ready to be committed.
         4. Unmodified – Unchanged files.
3. How to initialize a Local project as a git repository?

Initialization involves designating a directory on your local system as a Git repository.

Use-case: Imagine you're working on a project on your local machine and want to publish the code to GitHub, track changes, or collaborate with other developers. The question is, how can you efficiently manage version control and teamwork? This is where Git comes into play.

Before running any Git commands, however, you must first initialize your directory as a Git repository.

* 1. **git init**
  2. This command will initialize your repository as a Git repository and add a hidden git folder in your local repository, where Git tracks everything about the changes in the code.
  3. Points to remember: When you clone any GitHub repository into your local system, you don’t need to initialize it as a Git repo.

1. Add & Commit.

Add – adds new or changed files in your working directory to the Git staging area.

* 1. **git add <FILENAME>**
     1. Stages only the mentioned filenames.
  2. **git add .**
     1. Stages new and modified, *without deleted.*
  3. **git add -A**
     1. Stages All (New/Modified/Deleted).
  4. **git add -u**
     1. Stages modified and deleted, *without new.*

Commit – It is the record of the change.

* 1. **git commit -m “COMMIT\_MESSAGE”**
  2. How to write commit messages -> Always write the commit messages that are readable and understandable by other developers.