**GIT & GITHUB**

**What is Git?**

GIT is a Version Control System (VCS); A source tool that helps in tracking and managing codes. It was designed to assist programmers to work in coordination and tracking changes in any of the files associated with the local directory.  Git also makes it easy to collaborate with others, as you can share changes and merge the changes made by different people into a single version of a file.

**What is GitHub?**

GITHUB to GIT is the same as a photograph is to a person. It provides a graphical user interface to the command line tool GIT. GitHub is designed as a repository hosting service. It also gives a wide range of features such as collaborating on the project, resolving issues, source code management, and exchanging ideas with people around the world.

**What is difference between local & remote repository?**

The local repository is a Git repository that is stored on your computer. The remote repository is a Git repository that is stored on some remote computer. The remote repository is usually used by teams as **a central repository** into which everyone pushes the changes from his local repository and from which everyone pulls changes to his local repository.

When you are finished with doing changes into **your workspace**, you can add them to **staging area** and from there you can commit the changes to your **local repository**. This can be done even when you are disconnected from the internet and **nobody else can see the changes** in your local repository.

**How to connect local to remote?**

This happens with the help of the Git Remote command when the repository is first created on local. And then the user connects it to the remote repository.

**What is Version Control? How many types of version controls we have?**

Version control is a system that tracks changes to a file or set of files over time so that you can recall specific versions later.

There are two main types of version control systems: centralized version control systems and distributed version control systems.

1. A centralized version control system (CVCS) uses a central server to store all the versions of a project's files. Developers "check out" files from the central server, make changes, and then "check in" the updated files. Examples of CVCS include Subversion and Perforce.
2. A distributed version control system (DVCS) allows developers to "clone" an entire repository, including the entire version history of the project. This means that they have a complete local copy of the repository, including all branches and past versions. Developers can work independently and then later merge their changes back into the main repository. Examples of DVCS include Git, Mercurial, and Darcs.

**What is a git branch?**

A branch is a version of the repository that diverges from the main working project. It is a feature available in most modern version control systems. A Main Git project can have more than one branch. When you want to add a new feature or fix a bug you should not do it in the main branch, so we create a new branch to summarize the changes and then merge it to the main branch using pull request.

**What is a Pull Request?**

A pull request is an event in Git where a contributor asks a maintainer of a Git repository to review code they want to merge into a project.

**Differences between git fetch and git pull?**

To understand the differences between fetch and pull, let's know the similarities between both of these commands. Both commands are used to download the data from a remote repository. But both of these commands work differently. Like when you do a git pull, it gets all the changes from the remote or central repository and makes it available to your corresponding branch in your local repository. When you do a git fetch, it fetches all the changes from the remote repository and stores it in a separate branch in your local repository. You can reflect those changes in your corresponding branches by merging.

**Differences between git fork and git clone?**

**Forking**creates your own copy of a repository in a remote location (for example, GitHub) . Your own copy means that you will be able to contribute changes to your copy of the repository without affecting the original repository.

**Cloning**makes a local copy of a repository, not your own copy. Think of it as downloading a repository onto your local hard drive. Unlike forks, clones have references to their original repositories.

If you don’t have permissions to contribute to the repository after cloning, but would like to implement changes anyway, a fork is the way to go.

**What is Merge conflict?**

Git can handle most merges on its own with automatic merging features. A conflict arises when two separate branches have made edits to the same line in a file, or when a file has been deleted in one branch but edited in the other.

**How to resolve the Merge conflict?**

Conflicts can occur when you merge or rebase branches that have diverged, and you need to manually resolve the conflicts before git can proceed with the merge/rebase. git status command shows the files that have conflicts, git diff command shows the difference between the conflicting versions and git add command is used to add the resolved files.

**What is git workflow you use in the organization?**

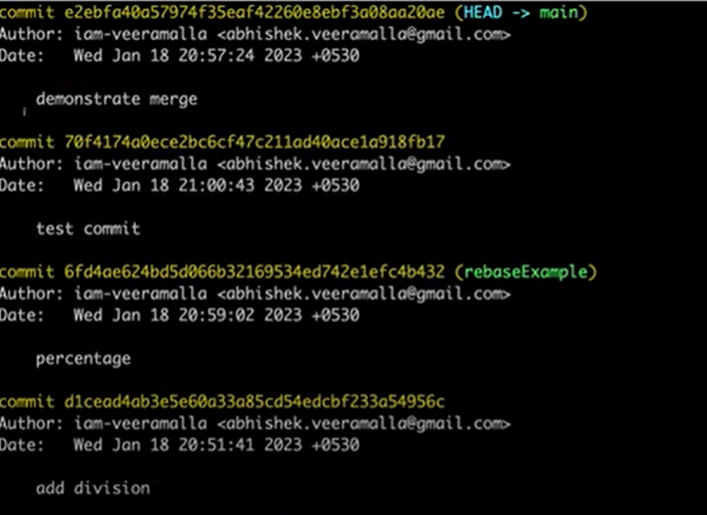
The git add && git commit -m “comments” && git push

**What is the difference between git cherry-pick, git merge and git rebase?**

The git cherry-pick is the act of picking a commit from a branch and applying it to another branch. This command is useful if there is only one or two commits, if there are many commits in one branch and to apply it to another branch becomes difficult and we cannot use cherry-pick. Instead, we can use the merge or rebase command to combine sequence of commits in a unified history or rebase them into new base commit.

**What is use of git merge and git rebase?**

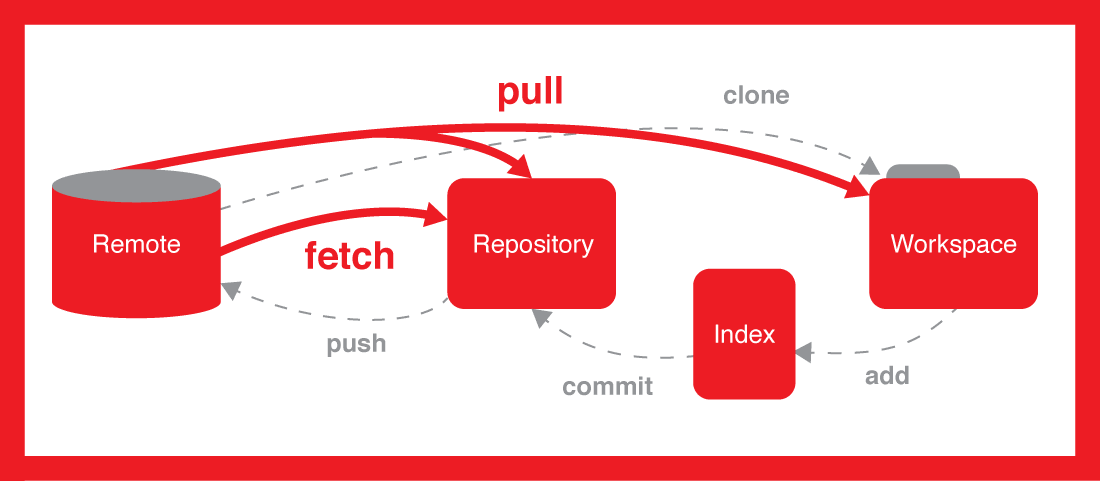
Rebasing and merging are both designed to integrate changes from one branch into another branch but in different ways.

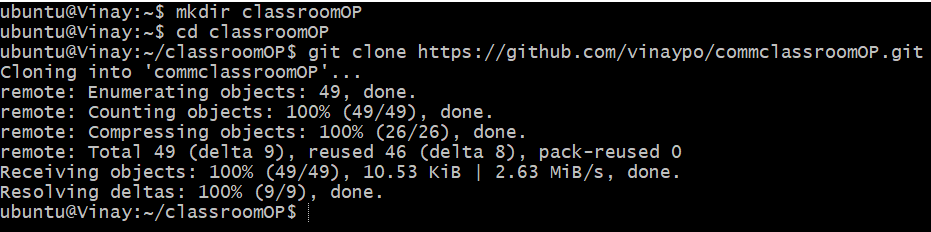
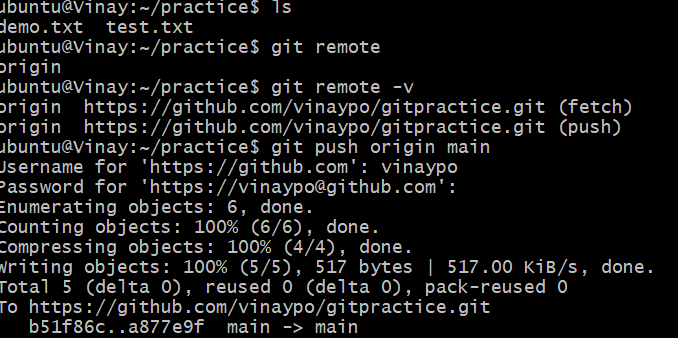
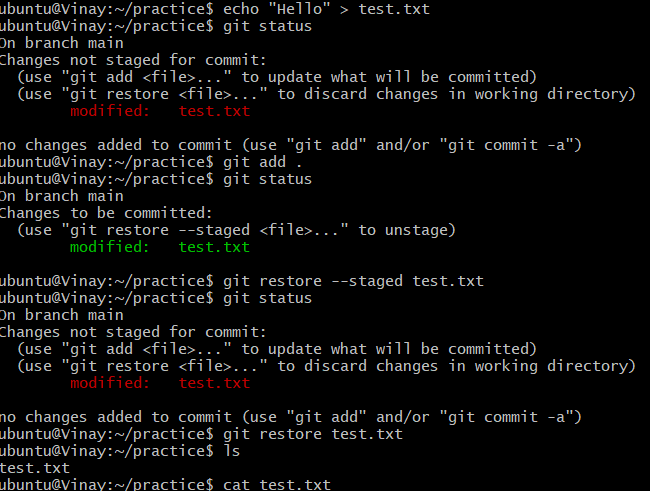


* When you do rebase a feature branch(rebaseExample) onto master, you move the base of the feature branch to master branch’s ending point.
* Merging takes the contents of the feature branch(mergeExample) and integrates it with the master branch on top.
* Merging adds a new commit to your history.

**Git Commands:**

* **git config:** The git config command is used to set git configuration values on a global or local level.
* **git init:** This command initializes an empty new git repository locally. basically a .**git** directory with subdirectories for objects.
* **git status:** This command show us the status of the repository and staging area. It allow us to see the tracked and untracked files and changes.
* **git add:** This command is used to add one or more files to staging (Index) area.
* **git commit:** It is used to record the changes in the local repository. Every commit contains the index data and the commit message.
* **git log:** This command is used to check the commit history.
* **git reset:** The git reset command is used to reset the changes in the working directory.
* **git stash:** This command can help you to (temporarily but safely) save your uncommitted locally – and allowing you to make changes, switch branches, and perform other Git operations.
* **git fork:** This command is used to copy the repository of the original project repository to freely test and debug with changes without affecting the original project after that changes contributor make a pull request to the original project owner who is the maintainer.
* **git fetch:** This command downloads all the changes from the remote repository and stores it in a separate branch in your local repository. Git fetch also updates the FETCH\_HEAD file.
* **git pull:** This command downloads all the changes from the remote or central repository and makes it available to your corresponding branch in your local repository.

****

* **git clone:** This command downloads an existing remote git repository to your local computer via using https or SSH.****
* **git squash:** In Git, the term squash is used to squash the previous commits into one. It is not a command; instead, it is a keyword**.**
* **git cherry-pick:** This command is the act of picking a commit from a branch and applying it to another branch.
* **git rebase:** Rebasing is the process of moving or combining a sequence of commits to a new base commit and integrate changes from one branch into another branch. If you use the rebase you get the linear commit history.
* **git merge:** This command will combine multiple sequences of commits into one unified history. In the most frequent use cases, git merge is used to integrate changes from one branch into another branch.
* **git diff:** This command helps to identify the differences between the states repositories or files
* **git push:** This command is used to upload local repository content to a remote repository. Pushing is how you transfer commits from your local repository to a remote repository.
* **git restore:** This **command** helps to un-stage or even discard changes in working directory. 
* **git remote:** This command lets you create, view and delete connections to other repositories from which you may wish to pull or push code.