GIT:

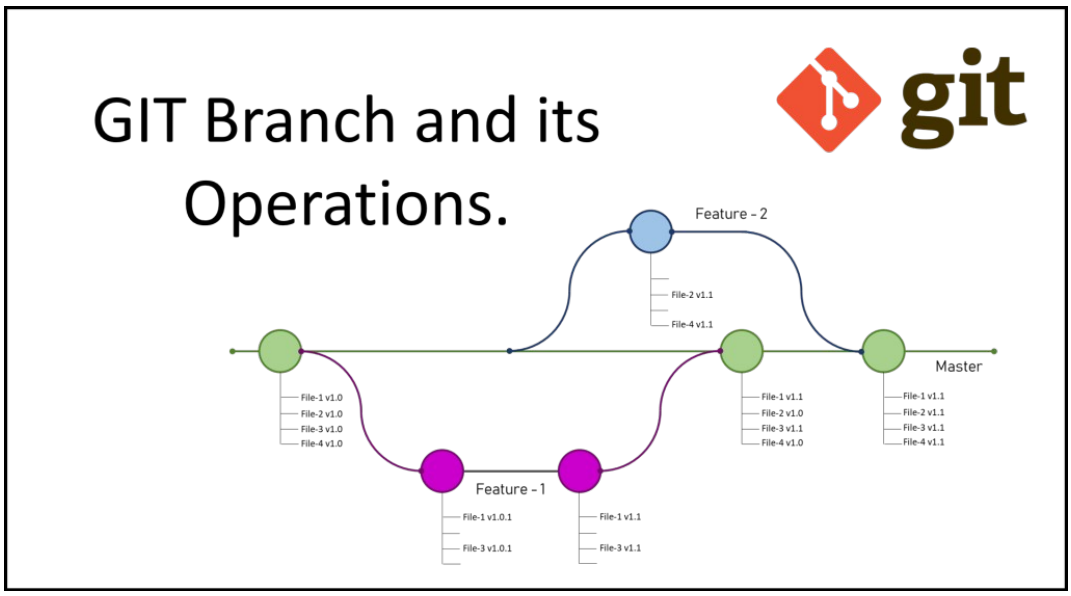
- **#git init:**
  - Initializes an empty git repository.
  {**Repository**: a repository is basically a folder all of your folder/code/file/images/movies can be stored, to store your important data}

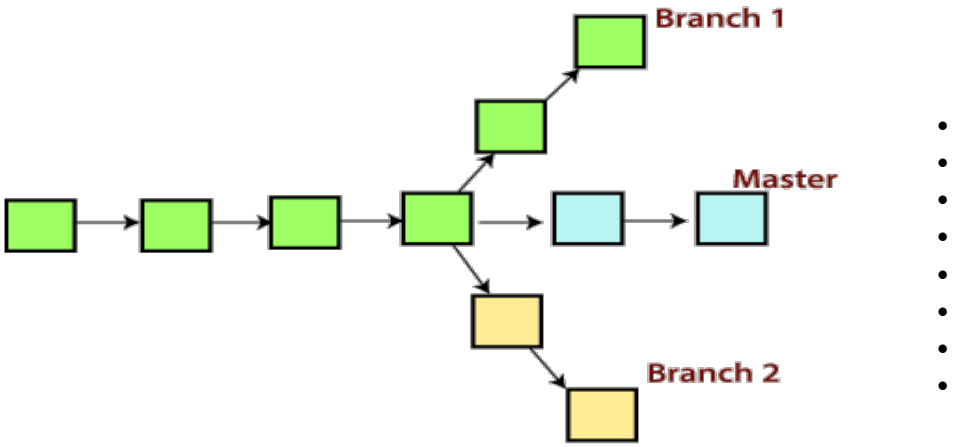
- **#git status:**
  - Its in git displays the current state of a local repository, including the working directory& staging area.
  - its gives you information about what branch are you in.
  - how many commits you have made.
  - how my track and untrack file are there

Note: you should write a git status command as you write in a folder where you done init.

- **git branch:**

- a branch is a new/separate version of the main repository.
- Gives the info / List of all the branches in git repo.
-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.





Git Branch Example:
The Nautilus developers are engaged in active development on one of the project repositories located at /usr/src/kodekloudrepos/news. During testing, several test branches were created, and now they require cleanup. Here are the requirements provided to the DevOps team: On the Storage server in Stratos DC, delete a branch named xfusioncorp_news from the /usr/src/kodekloudrepos/news Git repository.

1**.**Access the Storage Server:

•Connect to the Storage server in Stratos DC using SSH.

```
ssh your_username@storage_server_ip
```

2.Navigate to the Repository:

•Change to the directory where the repository is located.

```
cd /usr/src/kodekloudrepos/news
```

3.Check Out a Different Branch:

•Ensure you are not on the branch you want to delete. Switch to another branch, such as **master**

```
git checkout master
```

4.Delete the Branch:

•Delete the **xfusioncorp_news** branch.

```
git branch -d xfusioncorp_news
```

If the branch has not been fully merged, you might need to force delete it:

```
git branch -D xfusioncorp_news
```

5.Verify the Deletion:

•List all branches to confirm the branch has been deleted.

```
git branch
```

- **#git checkout:**

The git checkout command in Git is versatile and can be used to switch branches or restore working tree files:

- **#git checkout -b <branch Name>**

- Switching to new branch, if you want to switch to an existing branch then use git checkout branch with -b.

•Switch branches

git checkout can make the currently active branch the new HEAD branch. This can help you avoid making changes to the wrong version of your site.

•Restore files

git checkout can restore a specific file to an earlier revision while leaving the rest of the project untouched. For example, you can check out a commit to view your repository in a previous state, which can be useful for debugging.

•Update files

git checkout updates the files in the working directory to match the version stored in the specified branch. It also updates Git's HEAD, which is the pointer to the commit at the tip of a branch

e.g.

#git checkout -b dev: to checkout to a new branch from current working branch.

#git checkout -b master: to switch into master branch.

- **#git commit:**

The git commit command is one of the core primary functions of Git. Prior use of the git add command is required to select the changes that will be staged for the next commit. Then git commit is used to create a snapshot of the staged changes along a timeline of a Git projects history.

Its command is used to save your changes to the local repository

- **#git add <filename>:**

- The git add command adds a change in the working directory to the staging area. It tells Git that you want to include updates to a particular file in the next commit.
- Used to stage the Untracked file.

- **#git commit -m "your message":**

- A Git commit message serves to explain the context and intent of the changes made in a particular commit.
- Used to track the staged files after being added by git add

- **#git restore –staged <filename>:**

- Used to untrack / unstage the staged file

Git Clone:
git clone is primarily used to point to an existing repo and make a clone or copy of that repo at in a new directory, at another location

e.g.,
#git clone <github path>

Git Clone Example:

DevOps team created a new Git repository last week; however, as of now no team is using it. The Nautilus application development team recently asked for a copy of that repo on Storage server in Stratos DC. Please clone the repo as per details shared below:

The repo that needs to be cloned is /opt/official.git

Clone this git repository under /usr/src/kodekloudrepos directory. Please do not try to make any changes in repo.

Solution:

1. SSH to stsor01 ----(stsor01 is storage server)
2. cd to /usr/src/kodekloudrepos
3. Run command:
4. git clone /opt/official.git

Origin/main:
from where the branch was originally created

#git push:
The git push 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 repo. It's the counterpart to git fetch, but whereas fetching imports commits to local branches, pushing exports commits to remote branches.

e.g.,
#git push origin main

#git pull:
The git pull command is used to fetch and download content from a remote repository and immediately update the local repository to match that content. Merging remote upstream changes into your local repository is a common task in Git-based collaboration work flows.

e.g.,
#git pull origin main

- **git fork:**

- Forking is a git clone operation executed on a server copy of a projects repo. A Forking Workflow is often used in conjunction with a Git hosting service like Bitbucket. A high-level example of a Forking Workflow is: 1. You want to contribute to an open source library hosted at bitbucket.org/userA/open-project.

-Forks let you make changes to a project without affecting the original repository, also known as the "upstream" repository.