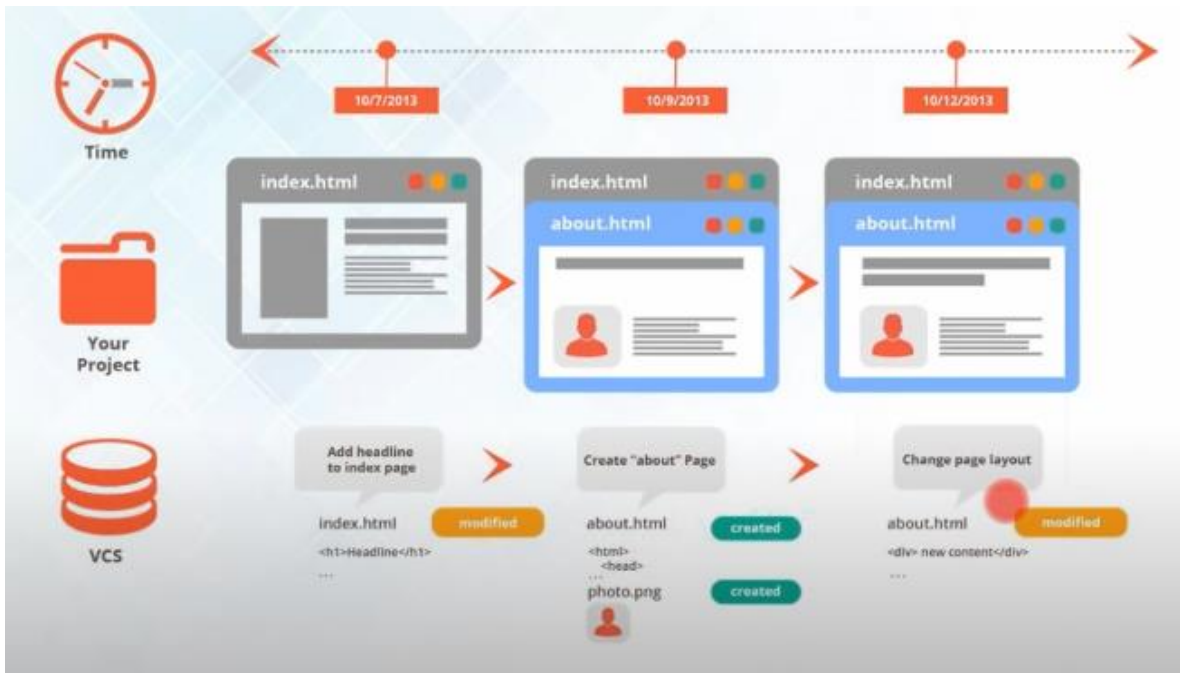


# Git & Github Notes

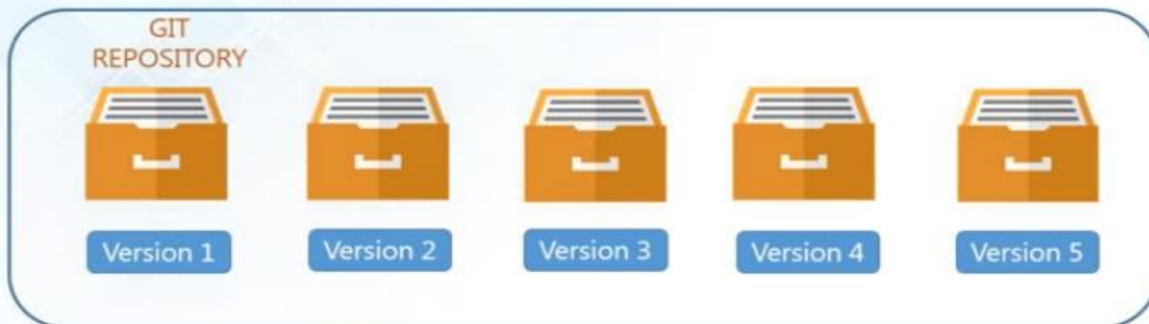
**Version Control:** - It is the management of changes to documents, computer programs, large websites, and other collections of information. These changes are usually called as “Snapshots” or “Versions”. Snapshot is actually the entire state of your project at a particular time.



## Why Version Control System (VCS)?

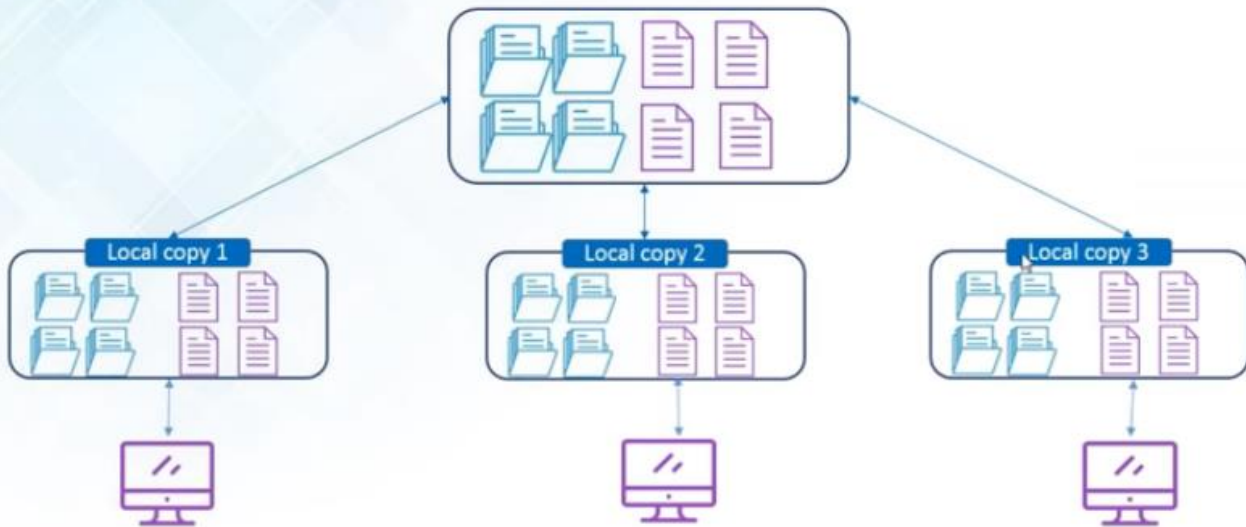
- Collaboration
- Storing versions

- Snapshots of all versions are properly documented and stored.
- Versions are also named accurately.



## - Backup

In any case if your central server crashes, a backup is always available in your local servers.



## - Analyze Project

When you change version -

- VCS provides you with proper description
- What exactly was changed
- When it was changed

And hence, you can analyze how your project evolved between versions.

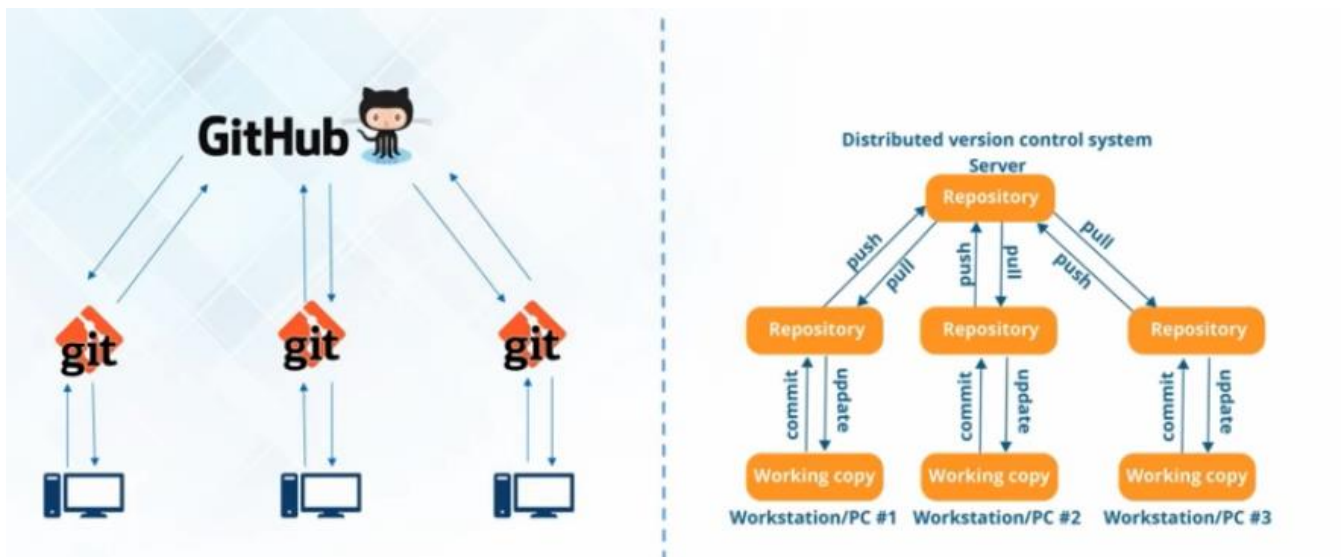


## Version Control Tools:

Git: - Distributed i.e. they provide all the developers with a local copy.

Subversion (SVN):- Centralized i.e. they don't provide all the developers with a local copy that means all collaborators/contributors are actually working directly with the central repository only. They don't maintain a local copy.

## Git & GitHub



**Repository:** - It is a data space where you store all and any kind of the files that is related to your project.

**Note:** Every developer at first make the changes in their local repo and after that they **push** those changes or transfer those changes into the central repository and also they update their local repositories with all the new files that are pushed into the central repository by an operation called **pull**.

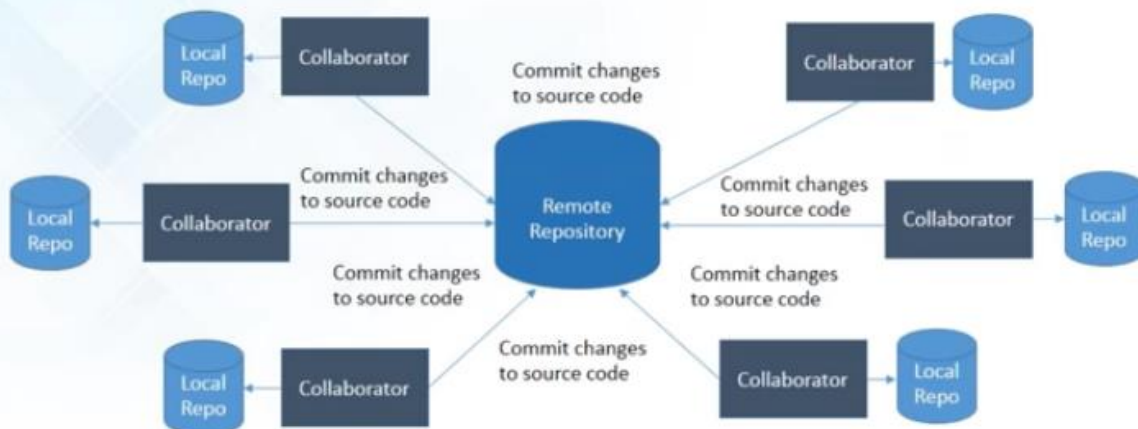
**GitHub** is a code hosting platform for version control collaboration i.e. it allows you to host your central repository in a remote server

While

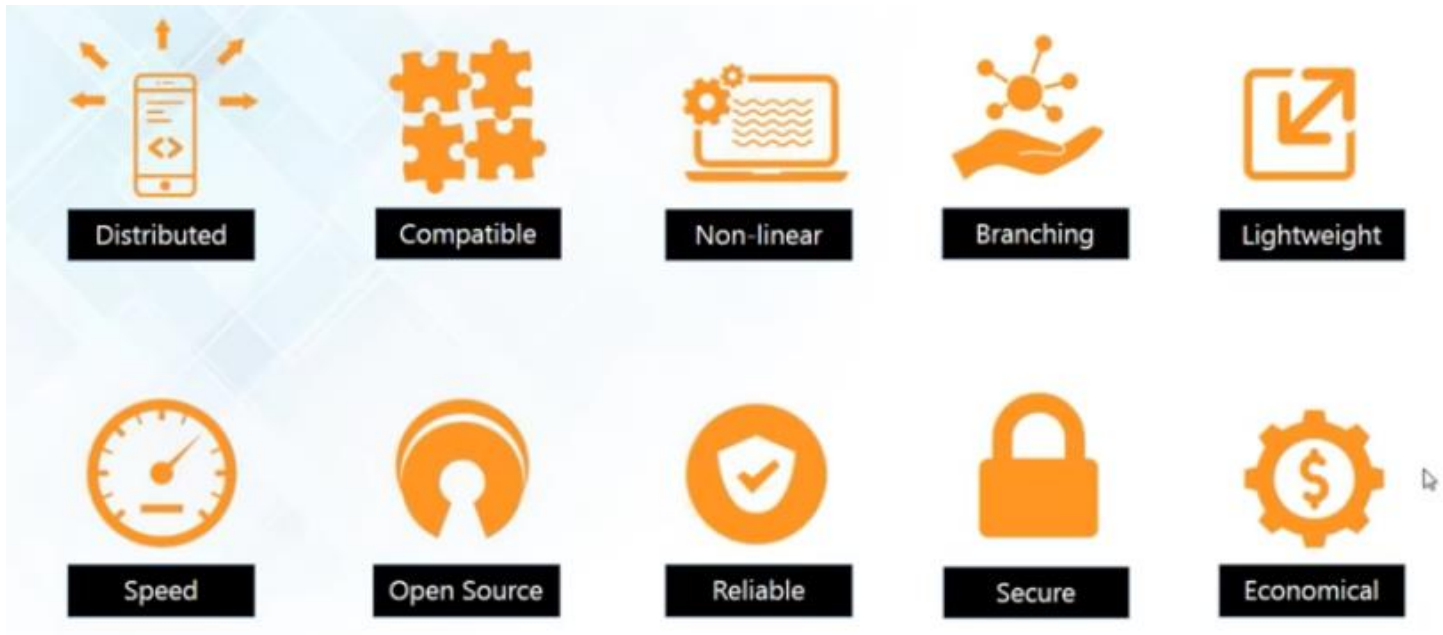
**Git** is the version control tool that allows us to create local repositories, fetch data from central repo and push all your local files into the central repo.



Git is a Distributed Version Control tool that supports distributed non-linear workflows by providing data assurance for developing quality software.



## Features of Git



### Distributed

- > Allows distributed development of code.
- > Every developer has a local copy of the entire development history and changes are copied from one repo to another.

### Compatible

- > Compatible with existing systems & protocols.
- > SVN & SVK repositories can be directly accessed using Git-SVN.

### Non-linear

- > Supports non-linear development of software i.e. git actually records the current state of your project by creating a tree graph (Directed Acyclic Graph) from the index.
- > Includes various techniques to navigate & visualize non-linear development history.
- > Git facilitates non – linear development by branching

### Branching

- > It takes only a few seconds to create & merge branches.
- > Master branch always contains production quality code.

### Lightweight

- > Uses lossless compression technique to compress data on the client's side.

### Speed

- > Fetching data from local repo is 100 times faster than remote repository.
- > GIT is one order of magnitude faster than other VCS tools by mozilla.
- > GIT is written in C

### Open Source

- > You can modify its source code according to your needs.

## Reliable

> On events of system crash, the lost data can be easily recovered from any of the local repositories of the collaborators.

## Secure

> Uses SHA1 (cryptographic algorithm) to name and identify objects.

> Every file & commit is check summed and is retrieved by its checksum at time of checkout.

## Economical

> Released under GPL's license. It is for free.

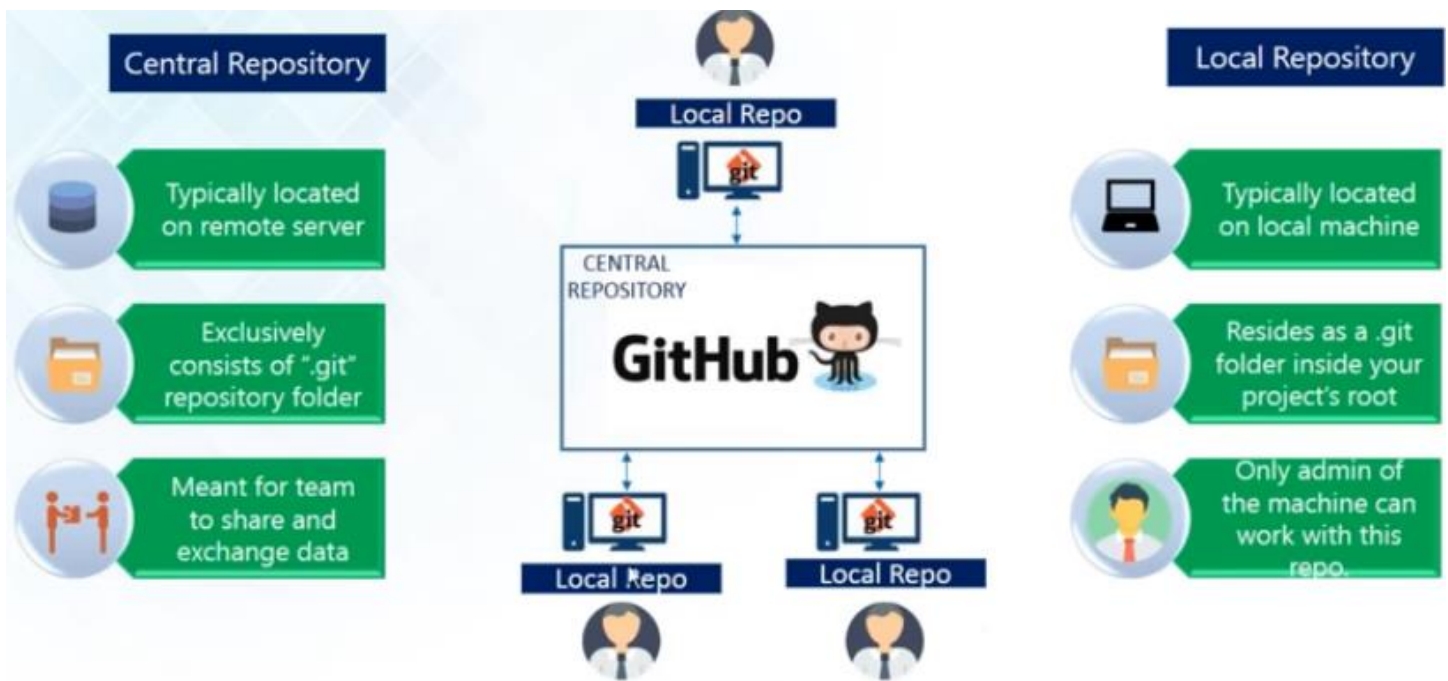
> All heavy lifting is done on client-side; hence a lot of money can be saved on costly servers.

## Repository

A directory or storage space where your projects can live. It can be local to a folder on your computer, or it can be a storage space on GitHub or another online host. You can keep code files, text files, image files, you name it, inside a repository.

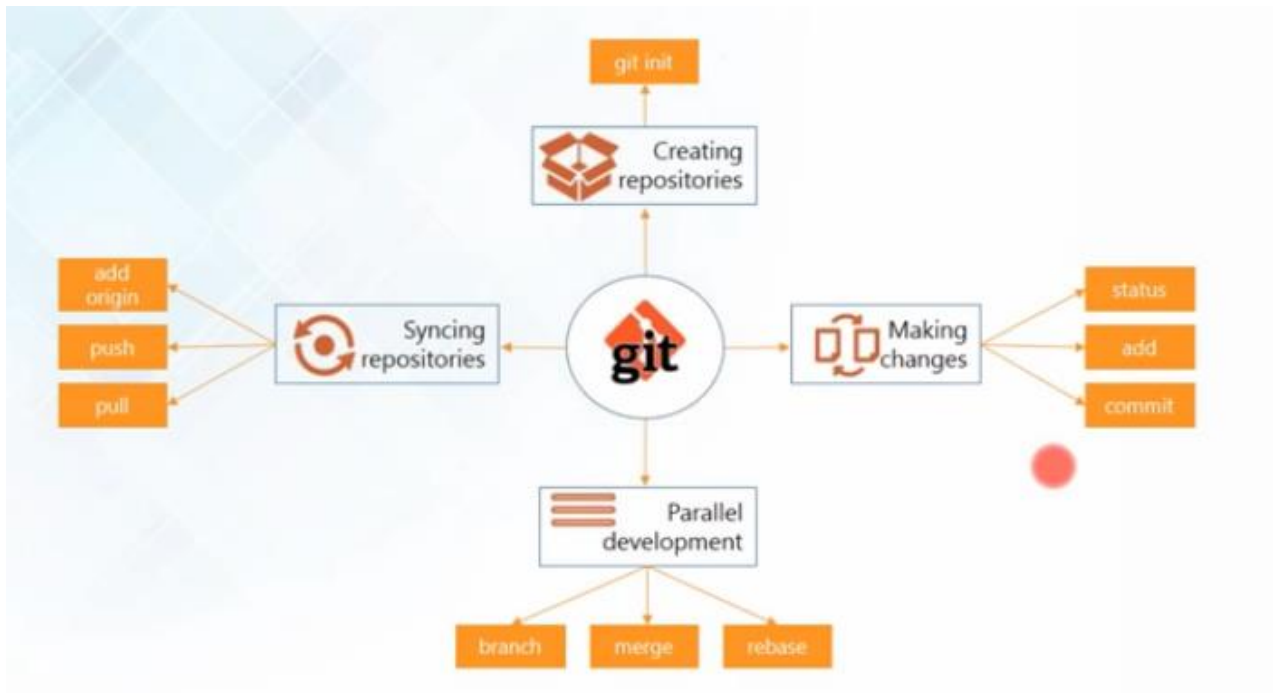
There are two types of repositories:

1. Central Repository
2. Local Repository






## Git Operations & Commands




## Create Repositories

- Create Repo
- Syncing Repos
- Making Changes
- Parallel Development
  - Branching
  - Merging
  - Rebasing
- Git Flow



# GitHub

Create your Central Repository on GitHub



# git

git init

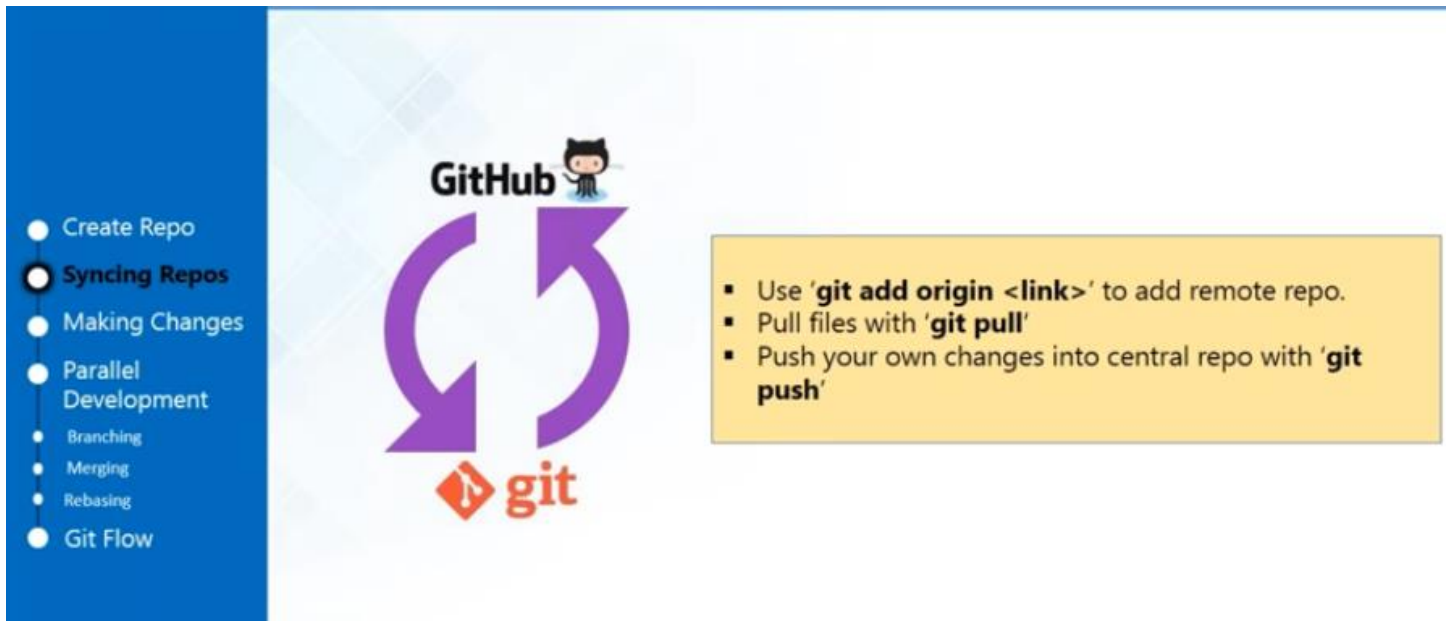
Install Git on your local machine and use "git init" to create your local repository.

git clone

Download or clone your repository from GitHub.

OR

## Syncing Repos



### Note:

1. Use **git init** only to initialize a new repository in Git repo. If you have an existing repo at Github and you want to import it to Git (as a repo) use **git clone <URL>** (it will import your whole Github repo.)
2. If you want to import existing files from Github repo (not whole repo) in a git repo
  - > **git init** (a local repo will be created)
  - > **git -v** (gives info if it has origin or not)
  - > **git remote add origin <URL>** (adds origin to your local repo)
  - > **git pull origin <branch Name>** (always check branch Name at Github. Don't assume it as master.)
  - > **git log** (check logs)
3. To uninitialized a git repo use command
  - > **rm -rf .git** (removes .git)
4. git pull = git fetch + get merge

## Making Changes



### Note:

1. To add single file or file one by one to the staging area,  
-> Use command **git add <filename>**
2. To add multiple files at once to the staging area,  
-> Use command **git add -A**
3. To commit file/files from staging area to local repo  
-> Use command **git commit -m "commit message"** or **git commit -a -m "commit message"**
4. To add and commit together  
-> Use command **git commit -am "commit message"**
5. To remove a file from staging area  
-> Use command **git rm --cached <file\_name>**



# Branching

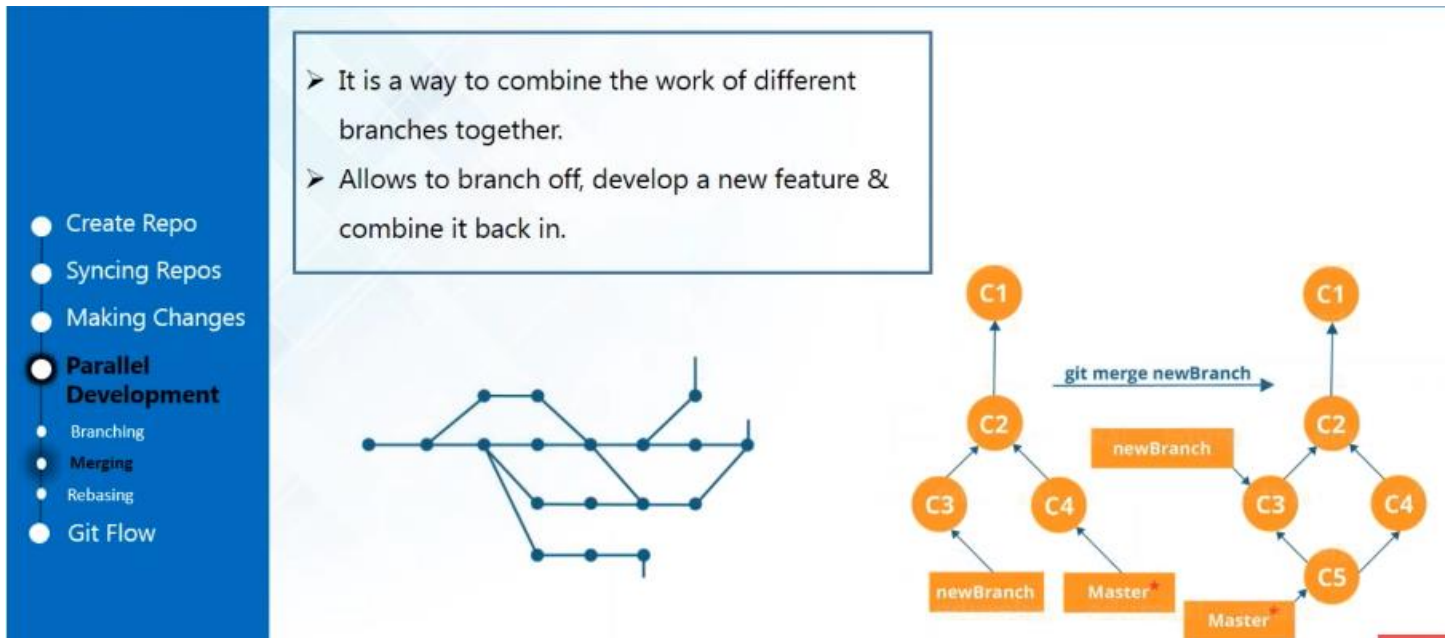
- Create Repo
- Syncing Repos
- Making Changes
- Parallel Development**
- Branching
- Merging
- Rebasing
- Git Flow

- Branches are pointers to a specific commit.
- Branches are of two types:
  - Local branches
  - Remote-tracking branches

## Note:

1. To create a new branch
  - > Use command **git branch <branch\_name>**
2. To switch from current branch to another branch
  - > Use command **git checkout <branch\_name>**
3. To delete a branch from local repo
  - > Use command **git branch -d <branch\_name>**

# Merging



Note:

1. To merge a branch to master [be in the destination branch to run this command]

-> Use command **git checkout master**

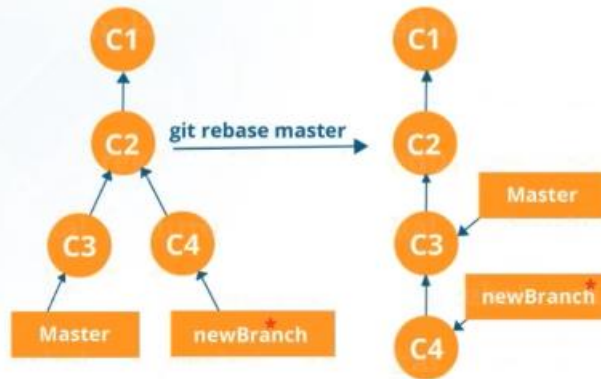
-> Use command **git merge <branch\_name>**

-> Use command **cat <file\_name>** to print content of file on the console.

## Rebasing



- This is also a way of combining the work between different branches.
- It can be used to make a linear sequence of commits.



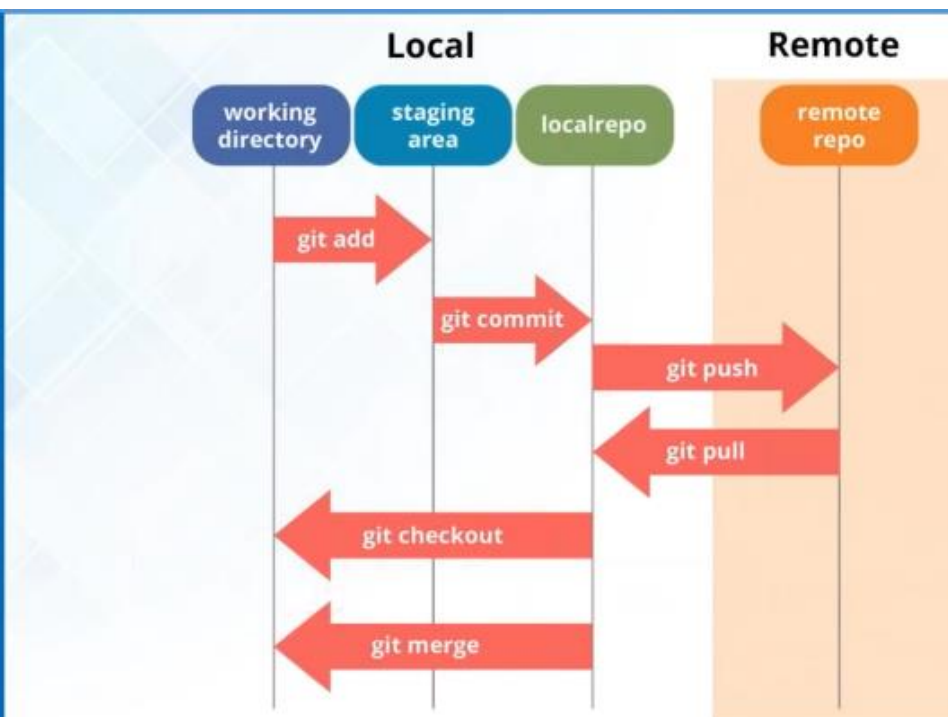
Note:

1. Rebasing is similar to merge except in rebasing instead of merging 2 branches , one branch gets added to the another branch like adding a node to the master branch.

2. To Rebase

-> Use command **git rebase <branchName>**

## Git Flow



Note:

1. To revert back the changes comit  
-> Use the command **git checkout <8digit hash of the comit to be reverted> <file\_Name>**
2. If bash goes in scroll mode  
-> **press q to exit**
3. Use command **git --help** for printing whole list of commands.

## Git Push

1. To push your changes from local repo to github  
-> Use the command **git push origin <branchName>**

Point to remember:

For proper gitFlow

1. Use command **git init** to initialize a repo in git
2. Use command **git remote -v** to list remote connections.
3. If there is no connection, use command **git remote add origin <URL (SSH or HTTP)>** to establish connection to the github repo.
4. Use command **git pull origin <branchName>** to pull any changes or readme file from github repo to git repo.
5. Now you can merge or add new changes.
6. For establishing SSH connection from your local repo to github
  - > Step 1: Generate ssh key  
-> Use Command **ssh-keygen** press Enter->Enter and your key will be generated at specified location.
  - > Step 2: Add the key in filename.pub to the github account SSH key.
  - > Step 3: Use command **ssh -T git@github.com** to establish connection.
  - > Now you can use SSH key of repo rather than HTTP.