

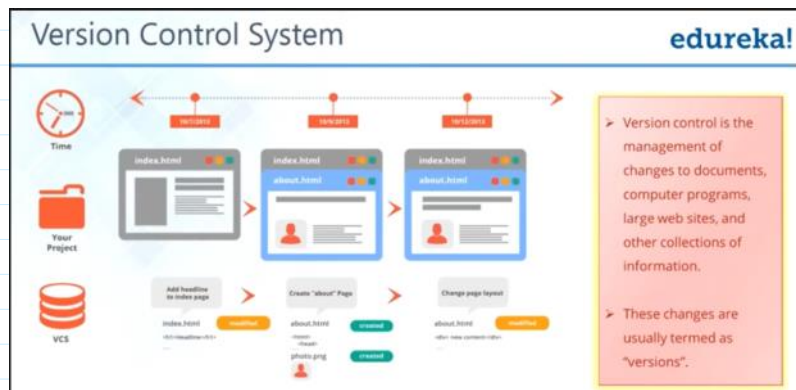
# DevOps Tutorial for Beginners | Learn DevOps in 7 Hours - Full Course | DevOps Training | Edureka

Monday, May 4, 2020 6:48 PM

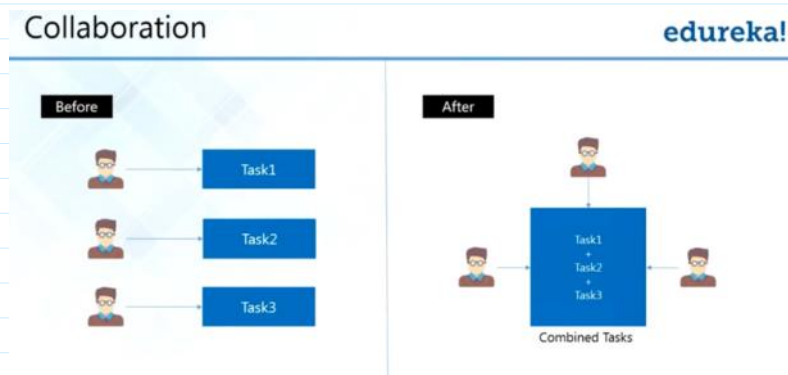
[DevOps Tutorial for Beginners | Learn DevOps in 7 Hours - Full Course | DevOps Training | Edureka](#)  
edureka!



## 2. GIT AND GITHUB



- Manages changes in the project for the entire development life
- When a change is made, the system will create a 'snapshot' of the entire project. AKA 'Different versions'



- Version Control Systems enables efficient collaborations
- Removes ambiguousness in who, what, or when changes have been made

## Storing Versions

ed

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

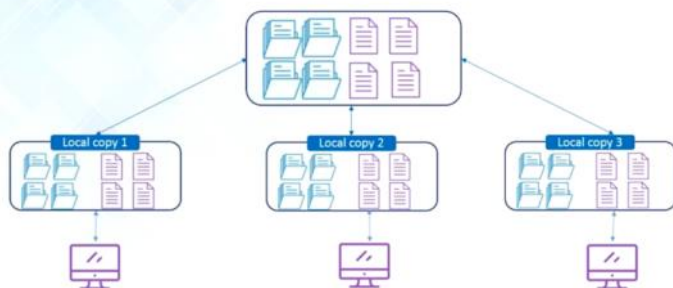


- Important to store different versions
  - Enables us to revert back when there bugs in new changes
  - Records progress, differences, and changes

## Backup

edur

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



- All developers can maintain a copy of the repository in their local machine. Perform code changes, improvements, etc in their local machine and push the code to the central server.

## Analyze

edur

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.



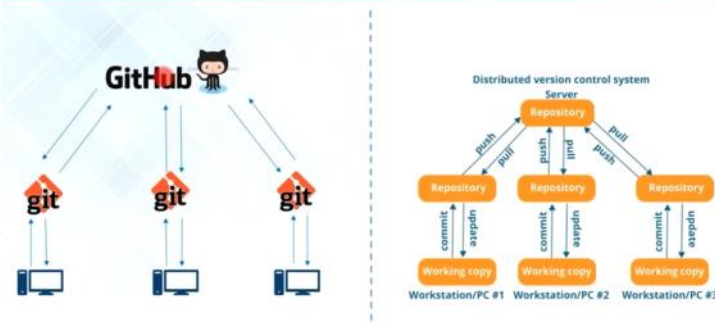
## Version Control System Tools

ed



## Git & GitHub

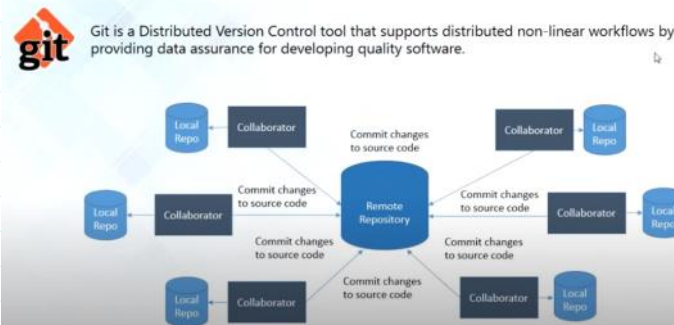
edure



- What exactly is git and github
- Diagram to the left depicts a distributed version control system
- Each developer has a copy of the central repository using the pull action, and push any changes they made to their local repository and push that code onto the central repository
- GitHub is the central repository
  - o Platform that allows to host central repository accessible through the network and the public
- Git is the tool that allows developers to create a local repository
- Distributed (has local versions of central repo) vs Centralized (has no local version of central repo) version control system.
- **Advantages:** no need for network connection everytime a change need to be pushed onto the central repo

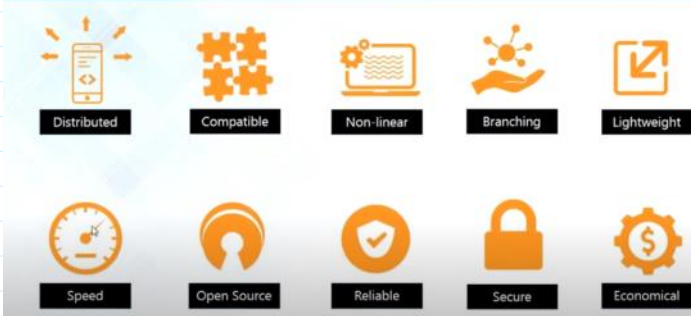
## What is Git?

ed



## Features of Git

edu



### Distributed

- Gives the power of having a local repository for each developer allowing them to make changes and test these changes locally without having to connect to the network to push and test these changes

### Compatible

- With existing systems & protocols (SVN, SVK, SSH)
- Mitigates changing things to conforms to protocols

### Non-Linear

- Tree graph model
- Includes various techniques to navigate & visualize non-linear development history

### Branching

- Git is the only one that implements the ranching model
- Takes only a few seconds to create & merge branches
- Master branch contains the production quality code

### Lightweight

- Uses lossless compression technique to compress data on the clients side, not on the server side

### Speed

- Pushing code to central repo is 100 x faster that remote repo

### Open Source

- Can modify source code according to your needs

### Reliable

- There are backups locally in case central repo server crashes

### Secure

- Uses SHA1 to name and identify objects - commits
- SHA1 cryptographic algorithm that converts commit obj to a photo digital hexadecimal code

### Economical

- It is for free
- All heavy lifting is done on the clients side

## What is a Repository?

edurek

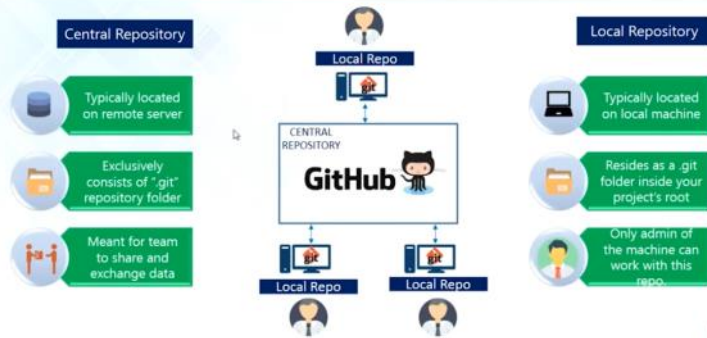
A directory or storage space where your projects can live. It can be local to a folder on your computer, or it can be a storage space on GitHub or another online host. You can keep code files, text files, image files, you name it, inside a repository.

There are two types of repositories:

1. Central Repository
2. Local Repository

## Central & Local Repository

edure




## Git Operations & Commands




## Creating Repositories

edurek

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



Create your Central Repository on GitHub



**git init**

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

**git clone** OR

Download or clone your repository from GitHub.

## Syncing Repositories

edureka!

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



- Use 'git add origin <link>' to add remote repo.
- Pull files with 'git pull'
- Push your own changes into central repo with 'git push'

## Making Changes

edureka!

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

git status



- Tells you which files are added to index and are ready to commit.

git add



- Lets you add files to your index.

git commit



- It refers to recording snapshots of the repository at a given time.
- Committed snapshots will never change unless done explicitly.

Git commit -a

Git add -A

Git log

## Parallel Development - Branching

edureka!

- 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



Git branch [name of branch] - this will create a branch from the master branch that contains all of the files originated from the master branch

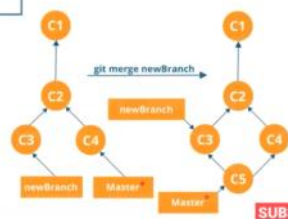
Git checkout [name of branch] - allows you to switch to the branch

## Parallel Development - Merging

edureka!

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

- It is a way to combine the work of different branches together.
- Allows to branch off, develop a new feature & combine it back in.



- Want to merge changes to the master branch

- Note that when merging, you have to be in the master branch

(in Master branch)

Git merge [name of branch]

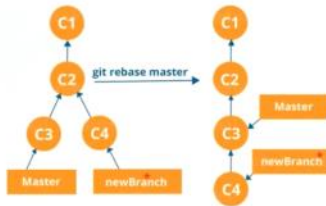
- Note that branch is still separate, user can still make changes and commit and merge again

- Note that git commit a file does not require an add iff performing the command in a branch because the file is a tracked file

## Parallel Development - Rebasing

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

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



- Adds new files to the head of the master repo

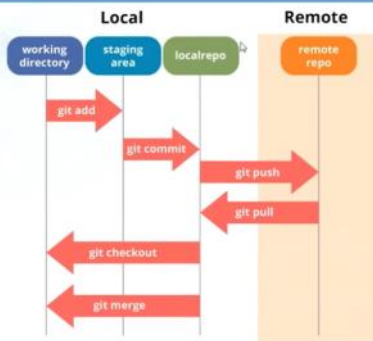
(in master branch)

Git rebase [name of branch]

- Reverting back to different version using commit hash keys

## Git Flow

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



## 3. CONTINUOUS INTEGRATION

### Process Before Continuous Integration

