**GIT IN ONE VIDEO:**

Agenda:

> Version Control - What and why ?

> Version Control Tools

> Github & Git

> Case Study: Dominion Enterprises

> Git Features

> Git Operations & Commands

>>What is Version Control ?:

> Version Control is the management of changes to documents, computer programs, large web sites, and other collections of information.

> These changes are usually termed as “Versions:.

So basically version control is created by doing any changes whether it may be creating, modifying and updating the working task. It created the number of version control on different changes done.

>> Why do we need Version Controls?

1. Basically Version controls provides the “Collaboration”.

Suppose number of employees working on the same project and they all are also working on same page and doing some modifications so when we merge the whole work and it will create lots of conflicts and we will not be able to find who have done what kind of changes in the project/ file.

So, by Version controls we can easily find out that who have done what kind of changes and it also protects the number of user to facing conflicts.

2. Version control provides the shared workspace.

Means where number pf users can make changes in the same file separately.

3. By Version controls we can visualise every work properly.

That we will be able to find who has done what kind of changes in the file.

4. Storing Versions:

> Snapshots of all versions are properly documented and stored.

> Versions are also named accurately.

5. Backup:

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

6. Analyze :

When you change version -

> VCS provides you with proper description

> What exactly was changed.

> When it was changed.

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

>> **VERSION CONTROL SYSTEM TOOLS**: There are basically 4 popular version control systems are -

1. GIT : distributed version control system

2. SUBVERSION (SVN) : centralised version control system

3. Concurrent Version System (CVS) : centralised version control system

4. Mercurial : distributed version control system

Git is a version control tool to fetch data from central server tool which allows the local repository

Github is a code hosting platform for version control systems for collaboration. We can say that GitHub is a social network for developer.

Advantages of Distributed over centralised version control system:

1. Backup

2. Speed

3. Separately work on local machines

>> **Github Case Study: Dominion Enterprises**

**Problem Statement : Each team has its own goals, projects and budgets and they also have unique needs and workflows:**

**Emp1. , emp2 , emp3 ,, emp4, emp5 having their private code**

**And they wanted to make private code “publicly” to make their work more transparent across the company.**

**So they adopted Github for this.**

**Reason for using Github as the solution:**

**> They noticed that that few of the teams were already using Github. Adapting a familiar platform has also made onboarding easier for new employees.**

**> Having all of their code in one place makes it easier for them to collaborate on projects.**

**One employee was using Jenkins and other was using Github,,**

**Github Enterprise has allowed us to store our company’s source code in a central, corporately controlled system. - Joe Fuller, CIO**

**Dominion Enterprises was using 45 projects so they choose Github.**

**Jenkins : They were also using Jenkins to facilitate continuous integration environment.**

**>>** What is Git?

**Git is a distributed version control tool that supports distributed non-linear workflows by providing data assurance for developing quality-software.**

**Local. -> Collaborator**

**Repo \. Commit changes to source code**

\ ]

Remote Repository

> Features of Git:

1. Distributed 2. Compatible. 3. Non-linear. 4. Branching 5. Lightweight. 6. Speed 7. Open Source 8. Reliable 9. Secure. 10. Economical

1. Distributed :

> Allows distributed development of code.

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

2. Compatible:

> Compatible with existing systems & protocols.

> SVN & SVK repo can be directly accessed using Git - SVN: That means we can access directly SVN repo by Git.

3. Non-linear :

> Supports non-linear development of software. : So when we work with Git then Git records the current state of the project by creating a tree-graph structure. As tree graph are non-linear structured.

4. Branching: Git is the only one which has the branching module. It provides the facility of creating, deleting branches for every new project.

> It takes only a few seconds to create & merge branches.

> Master branch always contains production quality code.

5. Lightweight :

> Use lossless compression technique to compress data on the client’s side: means as we have a large size of files on our local system when we push those files to client’s side then it compresses those files and then upload it and when we pull that code then it again decompress that code on our local machine i.e, called lossless compression.

6. Speed :

> Fetching data from local repository is 100 times faster than remote repository.

> GIT is one order of magnitude faster than other VCS tools I.e, 10 times.

7. Open Source:

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

8. Speed :

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

9. Secure :

> Uses SHA1 to name and identify objects.

> Every file & commit is checksummed and it is retrieved by its checksum at the time of checkout. : Means sometimes a commit creates a problem in the project then we can remove that commit and also revert back previous commits.

10. Reliable :

> 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.

>> **What is a 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

> Difference between Central Repository and Local Repository:

Central Repository: 1. Typically located on remote server.

2. Exclusively consists of “.git” repository folder.

3. Meant for them to share and exchange data.

Local Repository: 1. Typically located on local machines.

2. Resides as a .git folder inside your project’s root.

3. Only admin of the machine can work with this repo.

>> **Git Operations & Commands :**

**Take data from image which we I captured in phone like git init, add origin,,,**

**Working flow in GIT:**

**Create Repo -> Syncing Repos -> Making Changes -> Parallel Development in which - - Branching, Merging, Rebasing comes -> Git Flow**

**>>1. Create Repo Git:**

**1. git init : Install Git on your local machine and use “git init” to create your local repository.**

**OR**

**2. git clone: Download or clone your repository from Github.**

**>>2. Syncing Repos:**

**> Use ‘git remote add origin <link>’ to add remote repo.**

**> Pull files with ‘git pull’ : git pull is used to fetch all the files from the remote server / central repo in your local machine.**

**> Push your own changes into central repo with ‘git push’.**

**>>3.Making Changes:**

**3.1 git status: Tells you which files are added to the index and are ready to commit.**

**3.2. git add : Lets you add files to your index.**

**3.3 git commit : i) It refers to recording snapshots of the repository at a given time.**

**ii) Committed snapshots will never change unless done explicitly.**

**So when we commit the code before pushing the code on master branch that is the committed code is in our local index repository.**

**3.4 git log:** **it is used to check the all the commit history.**

**>>4. Parallel Development :**

**4.1 Branching:**

> Branches are pointers to a specific commit.

> Branches are of two types:

. Local branches

. Remote-tracking branches