VCS -- Version control system

it is like time machine to go to older version

Example:

1. In patching we will have kernel versions to boot from grub

2. Apache installed will that the version and can be installed old one with version

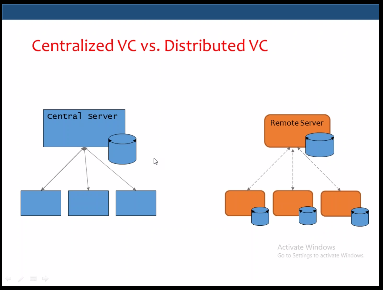
we have 2 types of vcs

1. Legacy/Centralised --> cvs,subversion

2. Distributed --> GIT,mercural

Legacy/Centralised VCS :

1. user need to upload the project code to the centralised server and all the modifications are saved only on centralised server not in local machine of user, user has only latest version if server down or crashes all the old versions is lost



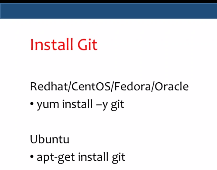
Distributed VCS:

1. From the above figure it is clear that each local machine of user have some small database that records the changes
2. Each single user has the whole repository of the project
3. We need the centralised server only when we want to share the code across
4. If server is crashed or not available we can copy the whole repository that is saved in local machine
5. GIT PULL and GIT PUSH used to send or get request from GIT
6. Everyone has the complete History
7. Everything is done offline -🡪 except Push/Pull
8. No Central Authority 🡪 except by convention
9. Changes can be shared without a server

GIT:

GIT uses Distributed vcs.

Installation:



Introduce yourself to GIT:

* git config --global user.name “Srinivas.chandupatla”
* git config --global user.email “[cnu405@gmail.com](mailto:cnu405@gmail.com)”

do the above settings only once it is applicable for all your projects

if you want to modify for only particular project for example project2

* cd project2
* git config user.name “Srinivas.chandupatla”
* git config user.email “cnu405@gmail.com”

As we skipped global in above command it is for particular project only

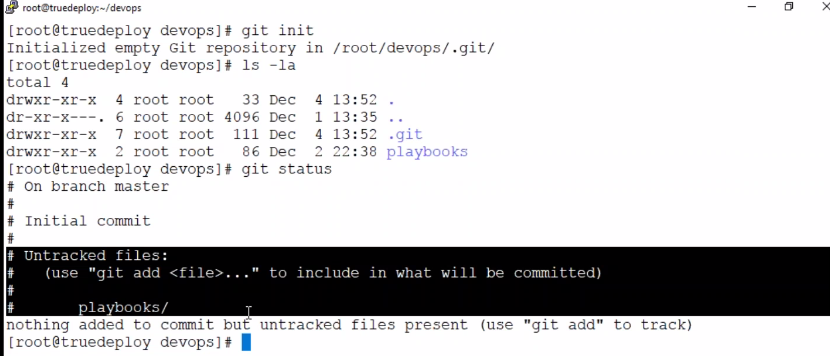
Create and Initialize repository

* mkdir /opt/project1 🡪 create project repository
* cd /opt/project1 🡪 switch to project
* git init . 🡪 Initialise the git repository

When you said git init in your project dir or when you cloned an existing project, you created a repository

* The repository is a sub-dir named .git containing various files
* The dot indicated a “hidden” dir

GIT provides the facility to track and untrack the file in the same repository



Use git add command to add the files or dir then git will start tracking the modifications

Ex:

#git add playbooks

#git status

Now we could see files are started committing but they are not saved

#git commit 🡪 will be used to save the files

If above command returns error as editor not found use below

#Export EDITOR=vi

Now file will open in that make changes at the starting only not anywhere else

#git commit

Creating first commit

:wq!

#git status 🡪 gives some information however it started saving

#git log 🡪 to check the saved files, here we could see Author as root which is not preferable so we give global or local user details

#git config --global user.name ‘srinivas.Chandupatla’

#git config --global user.email ‘cnu405@gmail.com’

Make some changes and commit to see if it is working

Ex:

We have created some file and added data as newfile

#git status

#git add newfile

#git commit –m ‘adding testing a playbook’ 🡪 instead of opening vi every time we can use this command

#git log

Now here we could see Author is changed.

#git config color.ui auto 🡪 to see commits in colors for easy identification

#git log

#git show 🡪 displays what was modified

Modify some contents in newfile

#vi newfile

Am making changes to see if git recognises it or not

:wq!

#git diff newfile 🡪 local copy to staging copy

Still file is not saved

#git status

#git add newfile

#git status

Latest version on commit area is referred as HEAD

#git diff HEAD newfile 🡪 compare file between local and commit area (if head is not provided it is from staging area)

#git commit –m ‘modified to check git’

To make simpler and use only 1 command

#git commit –a –m ‘adding other file’

#git log

#git status

Commands which we used till now

#git init

#git add

#git commit

#git log

#git show

#git status

#git diff

And user commands to add global and local

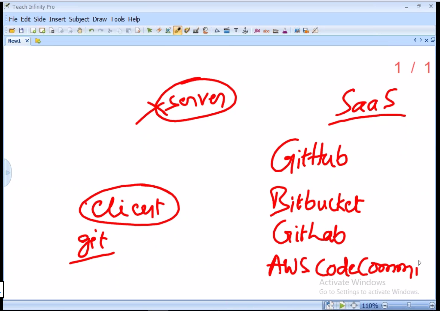
#git help –a 🡪 get list of all commands

#git help command or man git config 🡪 to use help commands

* GIT uses client and server technology
* The workstation or machine on which we install GIT is called as client
* On which we have installed GIT HUB or GIT LAB is called Server
* If we don’t want to install GIT HUB or GIT LAB it is available as SAAS (Software as a service here storage is available as software)

They are

1. GIT HUB 🡪 it is free and unlimited storage visible to the public, do not keep any sensitive date
2. Bit Bucket
3. GitLab
4. AWS Code commit



Create account in github

Create new repo then it will prompt for new repo url and commands

#git remote add origin <https://gitxxxxxxxx.git>

#git push –u origin master

At the top we can find commit to see how many commits are made and can download depending on our requirement.

To download any commit

Click at right side and select either clone or downloadzip.

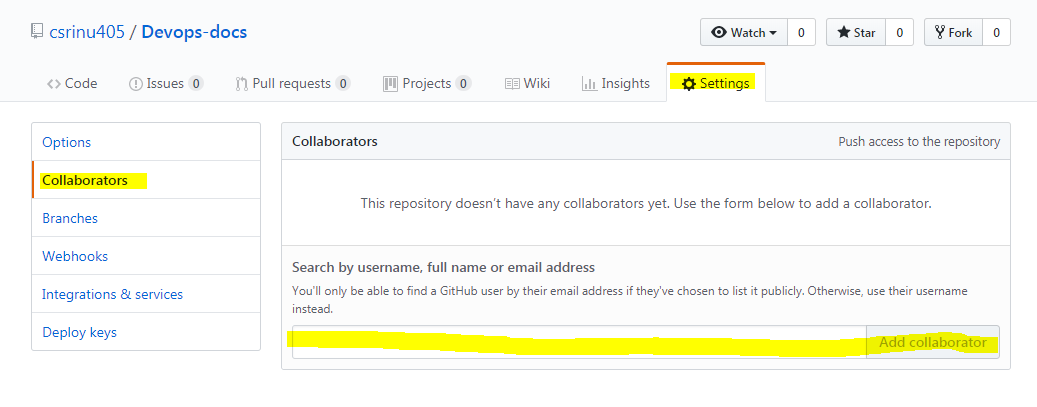
As it is public accessible use the below command to copy in any user account or scenario like you have joined a company where already members had old repo copy that to your machine and start working.

#git clone repo url (clone url)

It will not ask for any credentials as it is public accessible but to modify the contents of other repo we need permission provided by repo owner below are the steps for that

* Select settings 🡪 at left corner select Collaboration🡪 in the centre of the screen we will add user account who wants the access.

Screenshot for the same



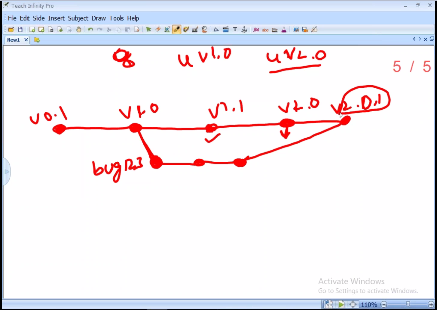
git clone is used only when we are cloning for the 1st time

git push is to upload to server

git pull is to download from server

git push and git pull alone required internet connection remaining all can be done offline

Test case for Branches a screen shot



**BRANCHES:**

Branches are used for testing or bug fixes once the code is ready we need to merge with the Main branch for release or for clients visibility

#git status 🡪 will also show on which branch we are on

#git branch feature\_login

#git branch –l 🡪 will list the branches, the one with \* is the current branch

#git checkout feature\_login 🡪 to change the branch

#git status 🡪 will show current branch

To Merge branch contents with Main branch or other branch switch to branch where we want to merge

Below is the example am merging from feature\_login branch to Master branch

Switch to master branch

#git checkout master

#git merge feature\_login

#git log

It is very important to check where we are before merging