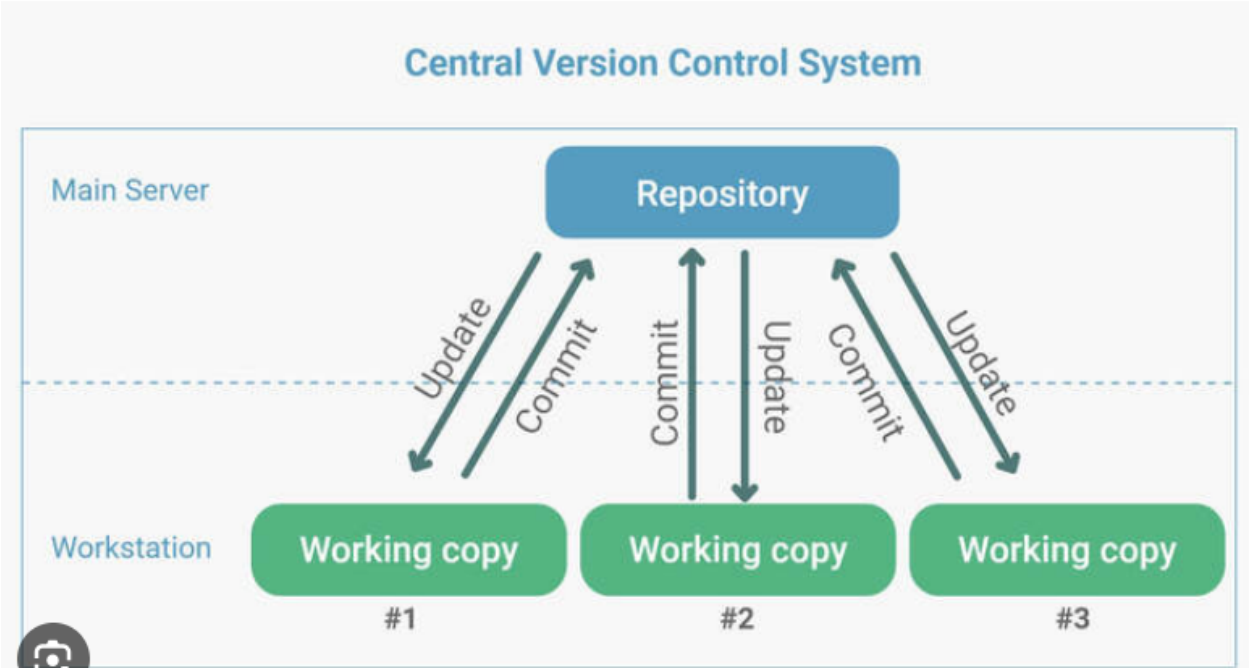
Git

Git is a Source Code Management / Software Configuration Management tool.
Version : On Linux git version 2.40.1
Version : On Windows git version 2.41.0

Type:

1. Centralised Version Control System
2. Distributed Version Control System

Centralised Version Control System (CVCS)

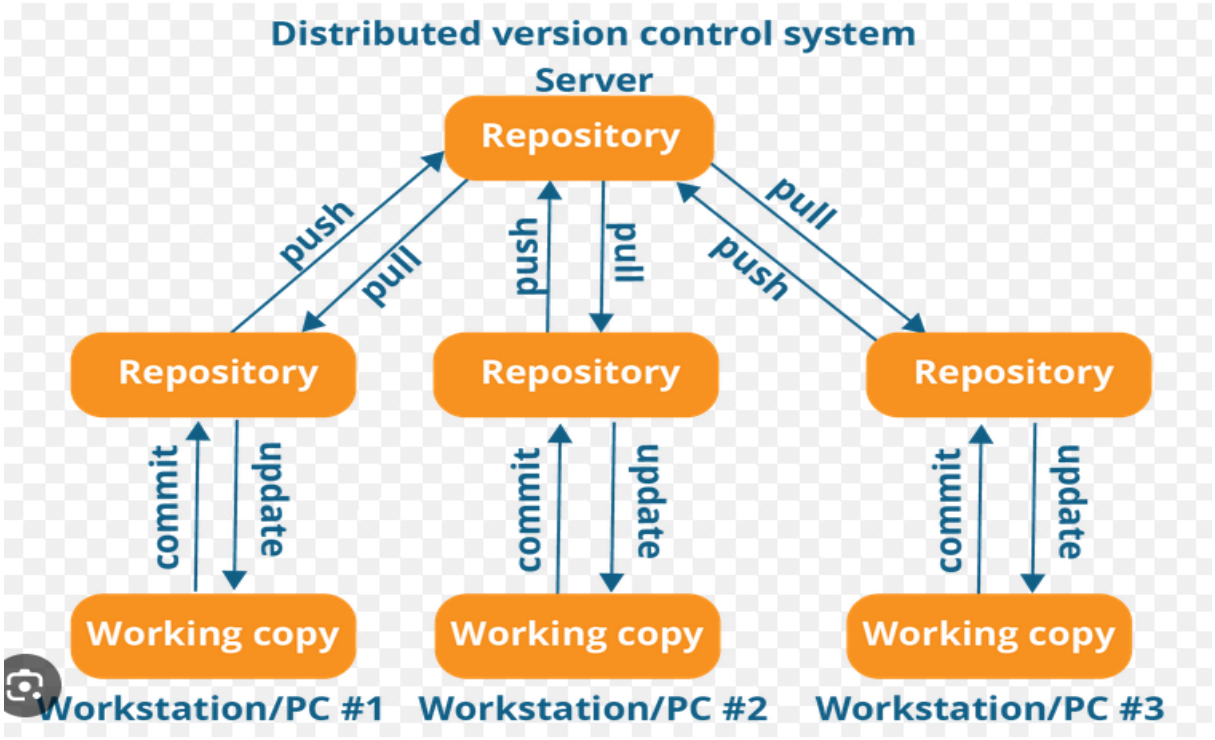


Drawback of CVCS:

1. It is not locally available meaning you always need to be connected to a network to perform any action.

2. Since everything is centralized, if the central server fails, you will lose the entire date. eg -> SVN tool

Distributed Version Control System (DVCS)



In a Distributed Version Control System, every contributor has a local copy or clone of the main repository. i.e everyonce maintains a local repository of their own which contains all the files & metadata present in the main repository.

CVCS	DVCS
In CVCS, a client needs to get a local copy of source from server, do the changes and commit those changes to source on server.	In DVCS, each client can have a local repo as well and have a complete history on it. Clients need to push the changes to the branch which will then be pushed to the server repository.
CVCS systems are easy to learn and set up.	DVCS systems are difficult for beginners. Multiple commands need to be

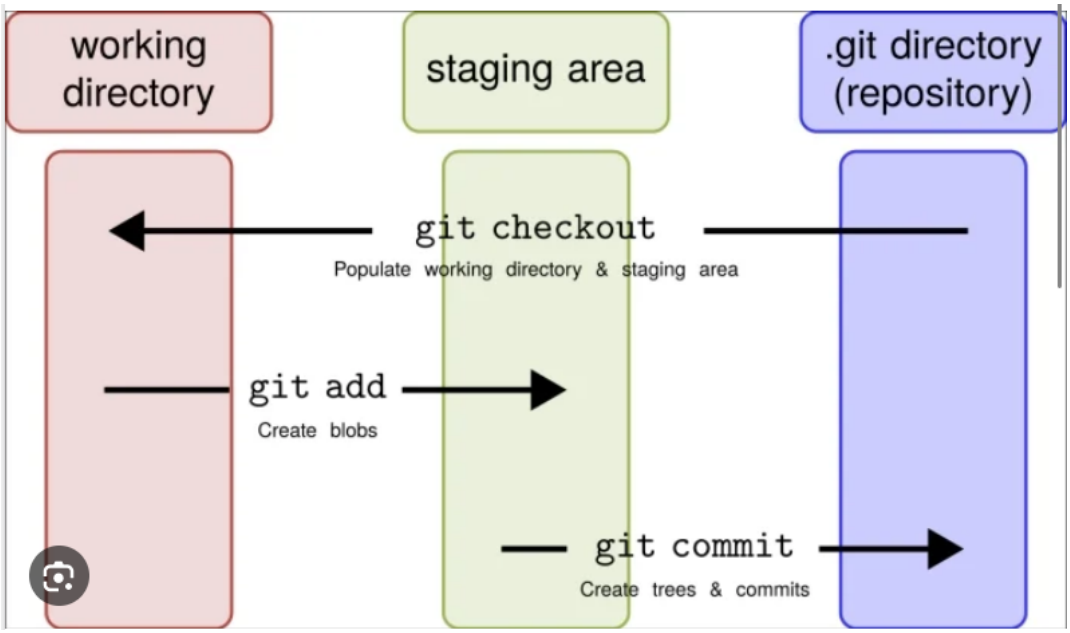
	remembered.
Working on branches is difficult in CVCS. Developers often face merge conflicts.	Working on branches is easier in DVCS. Developers face less conflit.
CVCS systems do not provide offline access.	DVCS systems are working fine in offline mode as a client copies the entire repository on their local machine.
CVCS is slower as every command needs to communicate with the server.	DVCS is faster as mostly users deal with local copy without hitting the server every time.
If the CVCS server is down, developers cannot work.	If the DVCS server is down, developers can work using their local copies.

Stages of git / Workflow

1. Working Directory
2. Staging Area
3. Local Repo

Steps to install git and create local repo:

1. Git Install
2. Create new directory (Example mumbaigit)
3. Run git init command in this directory to convert this into local repo



Git Terminology

Repository:

1. Repository is a place where you have all your code or kind of folder on the server.
2. It is a kind of folder related to one product.
3. Changes are personal to that repository.

Server:

1. It stores all repositories.
2. It contains metadata also.

Working Directory:

1. Where you see files physically and do modification.
2. At a time, you can work on a particular branch.
3. In other CVCS, Developers generally make modifications and commit their changes directly to the repository. But git does not track each and every modified file. Whenever you do commit on operations, git looks for the files present in the staging area. Only those files present in the staging area are considered for commit and not all the modified files.

Commit-ID/ Version-ID/ Version:

- ❖ Reference to identify each change.

- To identify who changed the file.

Tags: Tags assign a meaningful name with a specific version in the repository once a Tag is created for a particular save, even if you create a new commit. It will not be updated.

Snapshots: 1. Represents the same data of particular time.
2. It is always incremental i.e It stores the changes only not the entire copy.

Commit:

- Store Changes in repository. You will get one commit id.
- It has 40 alpha-numeric characters.
- It uses the SHA-1 Checksum concept.
- Even if you change one dot, the commit id will change.
- It actually helps you to track the changes.
- Commit is also named as SHA1 hash.

PUSH:

Push operation copies changes from a local repository instance to a remote or central repo. This is used to store the changes permanently into the git repository.

PULL:

Pull operation copies the changes from a remote repository to a local machine.
The pull operation is used for synchronization between two repo.

BRANCH:

Product is the same, so one repository but different task.

- Each task has one separate branch.
- Finally merge all branches.
- Useful when you want to work parallelly.
- Can create one branch on the basis of another branch.
- Changes are personal to that particular branch.
- Default branch is ‘Master’.
- File created in the workspace will be visible in any of the branch workspace until you commit. Once you commit, then that file belongs to that particular branch.

Advantages of Git

- ➔ Free and Open Source.
- ➔ Fast and small as most of the operations are performed locally, therefore it is fast.
- ➔ Security : Git uses a common Cryptographic hash function called secure hash function (SHA1) to name and identify objects within its databases.
- ➔ No need for powerful Hardware.
- ➔ Easier branching: If we create a new branch, it will copy all the codes to the new branch.

Types of Repository

- ➢ Bare Repositories (Central Repo)
  - ❖ Store and share only.
  - ❖ All central repositories are bare repo.

- ➢ Non-Bare Repositories (Local Repo)
  - ❖ Where you can modify the files.
  - ❖ All local repositories are non-bare repositories.

Lab-1: How to use git and create github account

Commands: After create EC2-Instances and login

- ➔ sudo su
- ➔ yum update -y
- ➔ yum install git -y
- ➔ git -- version
- ➔ git config --global user.name "prince.jaiswal"
- ➔ git config --global user.email "prince.jaiswal@spicemoney.com"
- ➔ git config -- list

GitHub Login- User Name : jaiswal-git

How to commit, push & pull from github

- Login into mumbai EC2 instance.
- Create one directory and go inside it.
  - mkdir mumbaigit
  - cd mumbaigit
- Run below commands
  - git init
  - touch mumbai1
    - Enter some data in mumbai1 file and save
  - git status
  - git add .
  - git commit -m "first commit from mumbai by princee.jaiswal"
  - git status
  - git log
  - git show < commit-id >
  - git remote add origin < central git url>
    Example : git remote add origin https://github.com/jaiswal-git/central-git.git

  - git push -u origin master
    (Enter github username & password / token)

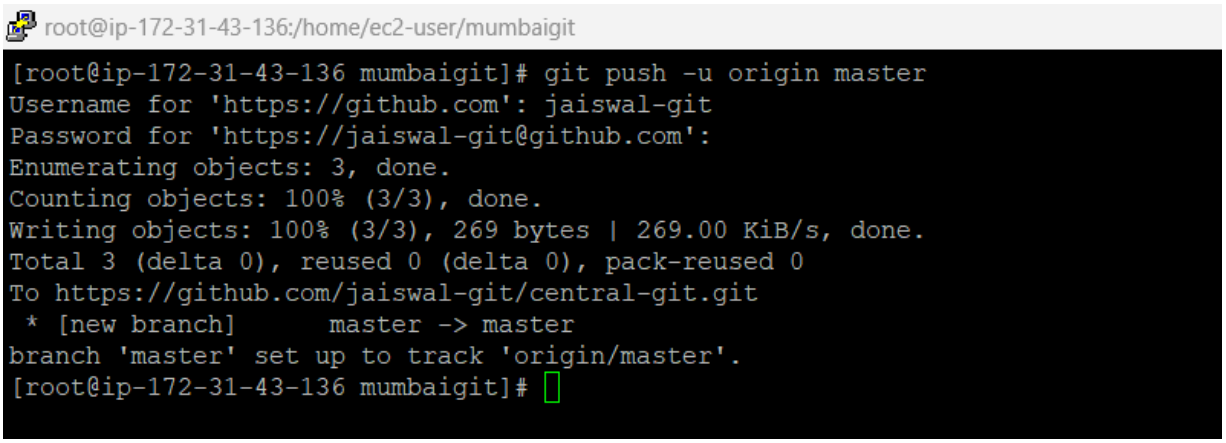
    In this step got Error: Authentication failed

    [root@ip-172-31-43-136 mumbaigit]# git push -u origin master
    Username for 'https://github.com': jaiswal-git
    Password for 'https://jaiswal-git@github.com':
    remote: Support for password authentication was removed on August 13, 2021.
    remote: Please see https://docs.github.com/en/get-started/getting-started-with-git/about-remote-repositories#cloning-with-https-urls for information on currently recommended modes of authentication.
    fatal: Authentication failed for 'https://github.com/jaiswal-git/central-git.git/'

    To resolve this issue. Use Personal Access Token, step given in below url:

https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/managing-your-personal-access-tokens

Now I'm able to push codes to github:



-> git pull origin master

To ignore some files while committing

Create one hidden file .gitignore and enter file format which you want to ignore.
For example:

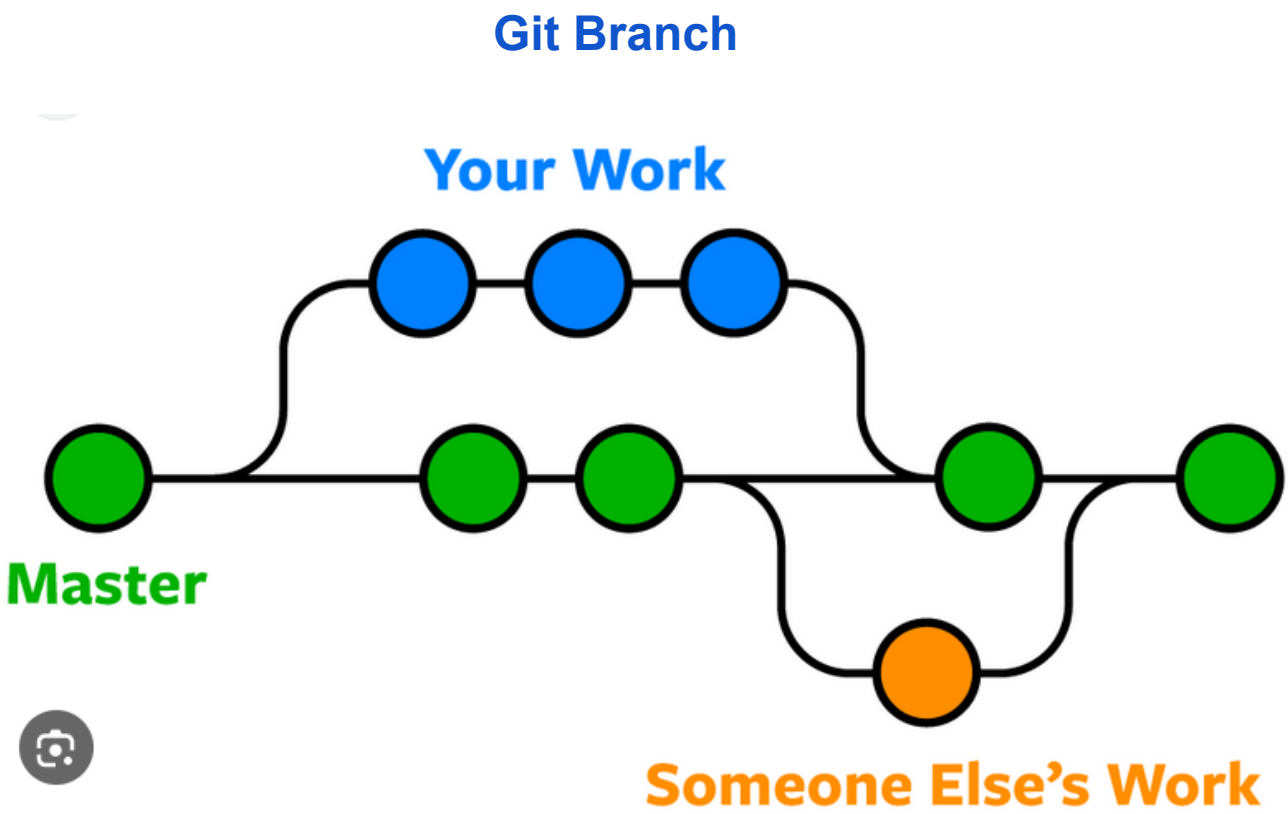
- vim .gitignore
  *.css
  *.java

- git add .gitignore
- git commit -m "Latest update in code and ignoring unwanted files from commit"
- git status

Create some text, java, css files and add them by running "git add"
For example:

- touch file1.txt file2.java file3.css
- ls
- git status
- git add
- git status

➢ git commit -m “my text files only”



The diagram above visualizes a repository with two isolated lines of development, One for your works, and one for someone else’s work by developing them as branches. It’s not only possible to work on both of them in parallel, but it also keeps the main master branch free from error.

- ☐ Each task has one separate branch.
- ☐ After done with code, Merge other branches with master.
- ☐ This concept is useful for parallel development.
- ☐ You can create any no. of branches.
- ☐ Changes are personal to that particular branch.
- ☐ Default branch is ‘Master’.
- ☐ Files created in the workspace will be visible in any of the branch workspace, until you commit. Once you commit then that file belongs to that particular branch.

- ☐ When created a new branch, data from the existing branch is copied to the new branch.

To see list of available branches
> git branch

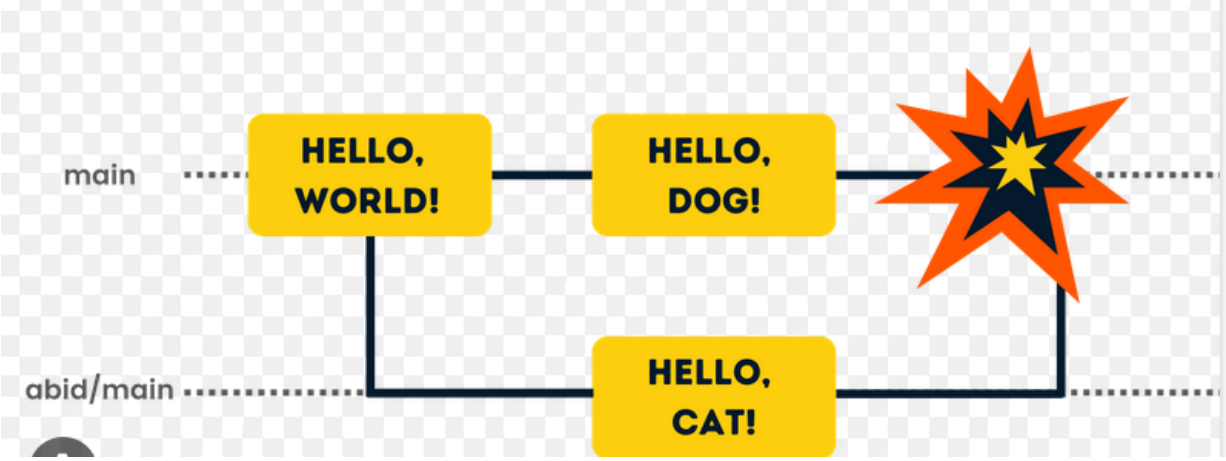
Create a new branch
> git branch <branch name>

to switch Branch
> git checkout <Name of branch you want to go>

to delete branch
> git branch -d <branch name>
> git branch -D <branch name> (this command used to forcefully delete)

Git Conflict

When the same file having different content in different branches, if you do merge, conflict occurs (Resolve conflict then add and commit).



Git Stashing

Suppose you are implementing a new feature for your product. Your code is in progress and suddenly a customer escalation comes. Because of this, you have to keep aside your new feature work for a few hours.
You cannot commit your partial code and also cannot throw away your changes. So you need some temporary storage, when you can store your partial changes and later on commit it.

To stash an item (Only applies to modified files. Not new files).

```
# To stash an item
> git stash

# To see stashed items
> git stash list

# To apply stashed item
> git stash apply stash@{0}
```

Then you can add and commit

```
# To clear the stash item
> git stash clear
```

Git Reset

Git Reset is a powerful command that is used to undo local changes to the state of a git repo.

```
# To reset staging area
> git reset < file name >
> git reset .

# To reset the changes from both staging area and working directory at a time.
> git reset --hard
```

Git Revert

The revert command helps you undo an existing commit.

- ☐ It does not delete any data in this process instead. Rather git creates a new commit with the included files reverted to their previous state. So, your version control history moves forward while the state of your file moves backward.

Commands:

- ➢ sudo su
- ➢ cd mumbaigit
- ➢ ls
- ➢ git status
- ➢ cat > newfile
  Hi, Final code for app
  ctrl + d
- ➢ git add .
- ➢ git commit -m “ write comment here”
- ➢ git log --oneline
- ➢ git revert < commit-id >

How to Remove untracked files
> git clean -n (dry run)
> git clean -f (forcefully)

Tags

Tag operation allows giving meaningful names to a specific version in the repository.

To apply tag
> git tag -a <tag name> -m <message> <commit-id>

To see the list of tags
> git tag

To see particular commit content by using tag
> git show <tag name>

To delete a tag
> git tag -d <tag name>

GitHub Clone

- ➔ Open the github website.
- ➔ Login and choose an existing repository.
- ➔ Now, go to your linux machine and run the command.
- ➔ git clone < url of the github repo>
- ➔ It creates a local repo automatically in a linux machine with the same name as in github account.

Thank you

Notes Written by: Prince Jaiswal (princejaiswal0007@gmail.com)