**GIT:**

* It is distributed system, means always there is no need of connection between server and client.
* It is faster compared to other version control systems.

**Repository:**

Central location where all files are being kept. We can create repository in 2 ways:

1. Create directory then initializing repository using ***git init*** command
2. ***Git init repo\_name***

Here we have 2 types of repositories.

1. Bare Repository – centralized , no connection to server
2. Non bare repository – distributed, easy to connect

By default repository is Non bare repository.

If we want bare repository then use ***git init repo\_name --bare***

**Staging:**

It marks files for tracking purpose.

Syn: git add file\_name

**Commit:**

It creates a snapshot of changes being made to files.

Syn: git commit –m “message”

We can also commit on particular file.

Syn: git commit –m “message” file\_name

Can we commit folder ? Yes

Create folder 🡪 create file 🡪 add the file 🡪 Commit

Note: We can’t commit empty folders.

* We can also check what commits we have made to files using ***git log*** command.
* ***Git reflog*** is used to check what are the commits we used.

How to check whether files are in staging or not :

***Git status*** gives which files are there in staging and what are files we have modified but not in staging.

* We can check file name using commit id

Syn: git show commit\_id

**Creating User:**

2 types of user’s – Local and Global

* Local user : we can see in local repository

Git config user.name “user\_name”

Git config user.email [user\_email@gmail.com](mailto:user_email@gmail.com)

User name and email are used to identify through which user we have commited.

* Global user : we can find in .gitconfig file

Git config --global user.name “user\_name”

Git config --global user.email [email@gmail.com](mailto:email@gmail.com)

* Git config –l (or) git config --list

It shows how many user’s are configured

* To commit with particular user:

Git commit --author=”user\_name” –m “message”

* We can also check configured users in **.config** file

**Gitignore :**

Is used to unmark files.

Navigate to .git/config 🡪 add ignorecase = true then create .gitignore file and then add file names which we need to ignore.

To unstage files which already in staging :

Syn: git reset HEAD file\_name -- to unstage particular file.

Git reset HEAD~1 -- to unstage all files which are in staging

**Checkout :**

We can get back files, which we have deleted mistakenly.

Syn : git checkout file\_name

We can also use checkout to delete recently modified data in a file.

**Log :**

History of commits.

Git log –n 5 🡪 prints recent 5 commits

Git log --oneline 🡪 displays commit\_id and commit\_message

Git log --pretty=oneline 🡪 displays commits with author

**Alias :**

We can give aliasing names for commands and then we can use alias names instead of command.

Syn : git config --global alias.alias\_name command-name

Ex: git config --global alias.st status

We can see aliasing names in **.gitconfig** file.

* Git branch -l to check how many branches are present.
* Asterisk(\*) mark indicates current branch.

**How to connect to server:**

To connect to server first we need public key.

* Generating public key :

Syn: ssh-keygen -t rsa -b 4096 -C “email.com”

It will generate public and private keys. It is always good practice to connect with public keys. Keys will store in below default path –

/root/.ssh/id\_rsa

We can also change that path if we want.

* **eval “$(ssh -agent -s)”** used to ensure SSH agent is available or not
* To connect to another server, first we have to copy public key of one server in .ssh/authorized keys
* **Ssh userid@ipaddress** (or) **ssh userid@hostname** to connect to particular server

To connect to git server :

* Copy public id 🡪 navigate to git profile 🡪 settings 🡪 SSh & GPG keys 🡪 new SSH key
* To authenticate that 🡪 **ssh --T git@github.com**

**Clone :**

Is used to get exact copy of repository.

Syn: git clone url (or)

Git clone path 🡪 to get files form another repository to current repository.

**Push :**

Is used to update changes to remote server.

Git push origin master

**Pull:**

Is used to get updates from particular remote server(which has connected currently).

Git pull origin

**Fetch :**

Is used to get updates from all remote servers which user connected.

Git fetch origin

**Revert :**

Is used to step back one commit.

Syn: git revert commit\_id

(or)

Git checkout commit\_id

Above syntax are used to delete commit temporarily.

To delete commits permanently 🡪 git reset commit\_id 🡪 git checkout file\_name