**Git** is an **open-source distributed version control system**. It is designed to handle minor to major projects with high speed and efficiency. It is developed to co-ordinate the work among the developers. The version control allows us to track and work together with our team members at the same workspace.

What is Git:

What is GitHub:

Difference between git and github:

A sample Work Flow:

Git: Version Control System

Track changes in files/folders

We can collaborate between Teams

|  |
| --- |
| Version Control  Version3  Version 2  Version1 |
| Version 3  Version2  Version1 |

Jithul (File)

Ankit (File)

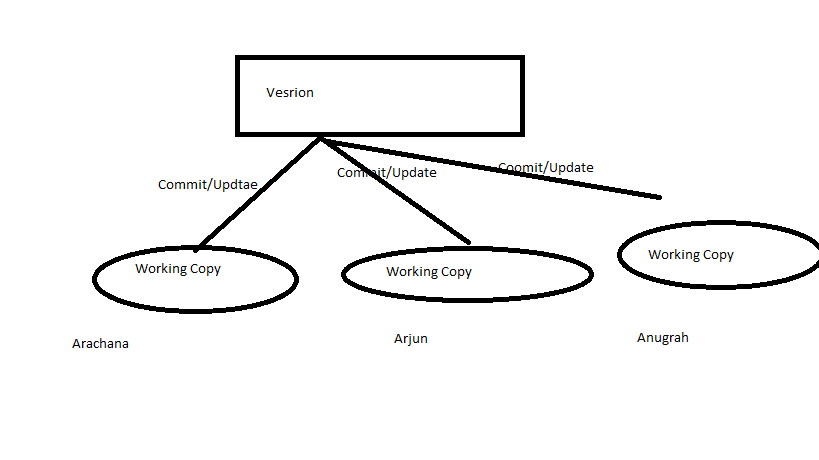
|  |
| --- |
| Checkout  File |

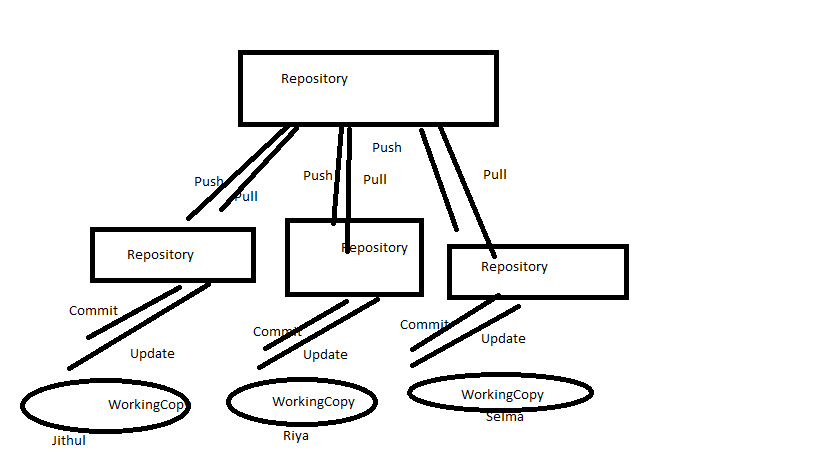
We have Two Types of Version Control Systems:

1)Centralized Version Control System

2)Distributed Version Control System

Centralized Version Control System:

2)Distributed Version Control System:



Git :Distributed Version Control System.

That is the why Git is famous and popular and it is also free and open source.

Git:https://git-scm.com/

GitHub: It is a website to upload your repositories online.

Provides Backup

Provides visual interface to your repo

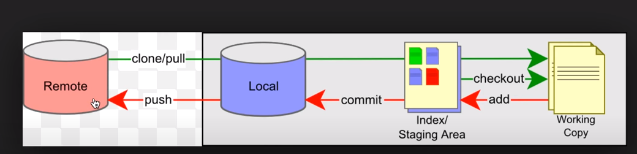
Makes collaboration easier

<https://github.com/>.

Git !=github

Git is a version Control System.

Git hub is a just website to put your repositories



This is a remote repository and this is a local repository Let us say our repository is put on a server its placed on a server which is in this remote server

We can do a clone or we can do a pull to take all the repositories present here into our local system.

So here we can get all the repositories Let us say we have all these four repositories or differenr branches of repositories now we can take or checkout any branch we will create a working copy for us and now we can do all our work on this working copy

And we can be offline during these changes So u can do all the changes and updates here,

Once we are done with our changes we can do our add command and what we will add command to is it will put all ur changes on a staging area it will still not commit it it will just put in staging area and then we have to executecommit commnd, which will make all our changes and committed to the local repository but at this point it is still not pushed it to the remote repository.

So we have to again do push command which will push our changes to the remote repositories

1)How to install git on windows

2)Add a project to git for tracking

3)git commands

4)Pushing project to remote repository

Step1:check whether git is installed or not in ur machine

Step2:download and install git

Step3:add your project to git

Step4:git commands

Git init:to intiliaze tracking files and folders

Git status

Git add

Git commit –m “any message”

Git remote add origin location of the repo

Git push –u origin

Do some changes in files

Step5:adding to remote repository:

# Git command line

There are many different ways to use Git. Git supports many command-line tools and graphical user interfaces. The Git command line is the only place where you can run all the Git commands.

The following set of commands will help you understand how to use Git via the command line.

## Basic Git Commands

Here is a list of most essential Git commands that are used daily.

* [Git Config command](https://www.javatpoint.com/git-commands#config-command)
* [Git init command](https://www.javatpoint.com/git-commands#init-command)
* [Git clone command](https://www.javatpoint.com/git-commands#clone-command)
* [Git add command](https://www.javatpoint.com/git-commands#add-command)
* [Git commit command](https://www.javatpoint.com/git-commands#commit-command)
* [Git status command](https://www.javatpoint.com/git-commands#status-command)
* [Git push Command](https://www.javatpoint.com/git-commands#push-command)
* [Git pull command](https://www.javatpoint.com/git-commands#pull-command)
* [Git Branch Command](https://www.javatpoint.com/git-commands#branch-command)
* [Git Merge Command](https://www.javatpoint.com/git-commands#merge-command)
* [Git log command](https://www.javatpoint.com/git-commands#log-command)
* [Git remote command](https://www.javatpoint.com/git-commands#remote-command)

Let's understand each command in detail.

8.1M

118

SQL CREATE TABLE

### Git config command

This command configures the user. The Git config command is the first and necessary command used on the Git command line. This command sets the author name and email address to be used with your commits. Git config is also used in other scenarios.

**Syntax**

1. $ git config --global user.name "ImDwivedi1"
2. $ git config --global user.email "Himanshudubey481@gmail.com"

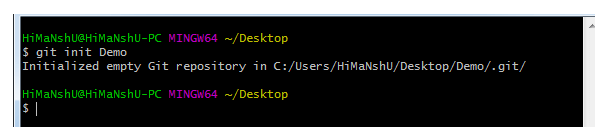
### Git Init command

This command is used to create a local repository.

**Syntax**

1. $ git init Demo

The init command will initialize an empty repository. See the below screenshot.

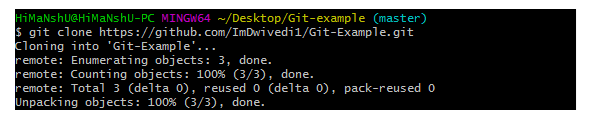


### Git clone command

This command is used to make a copy of a repository from an existing URL. If I want a local copy of my repository from GitHub, this command allows creating a local copy of that repository on your local directory from the repository URL.

**Syntax**

1. $ git clone URL



### Git add command

This command is used to add one or more files to staging (Index) area.

**Syntax**

To add one file

1. $ git add Filename

To add more than one file

1. $ git add\*

Git Commands

### Git commit command

Commit command is used in two scenarios. They are as follows.

**Git commit -m**

This command changes the head. It records or snapshots the file permanently in the version history with a message.

**Syntax**

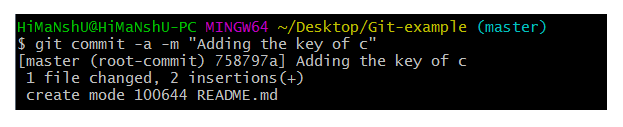
1. $ git commit -m " Commit Message"

**Git commit -a**

This command commits any files added in the repository with git add and also commits any files you've changed since then.

**Syntax**

1. $ git commit -a

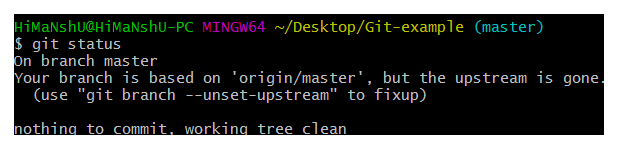


### Git status command

The status command is used to display the state of the working directory and the staging area. It allows you to see which changes have been staged, which haven't, and which files aren?t being tracked by Git. It does not show you any information about the committed project history. For this, you need to use the git log. It also lists the files that you've changed and those you still need to add or commit.

**Syntax**

1. $ git status



### Git push Command

It is used to upload local repository content to a remote repository. Pushing is an act of transfer commits from your local repository to a remote repo. It's the complement to git fetch, but whereas fetching imports commits to local branches on comparatively pushing exports commits to remote branches. Remote branches are configured by using the git remote command. Pushing is capable of overwriting changes, and caution should be taken when pushing.

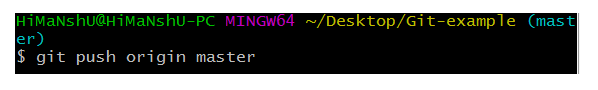
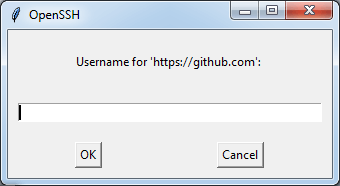
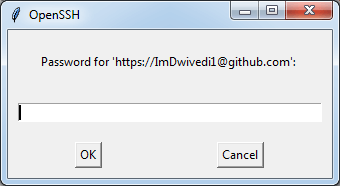
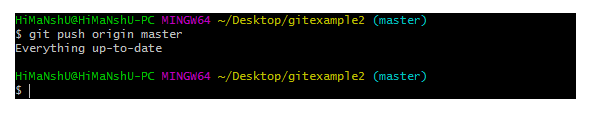
Git push command can be used as follows.

**Git push origin master**

This command sends the changes made on the master branch, to your remote repository.

**Syntax**

1. $ git push [variable name] master

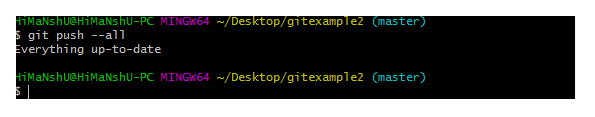
  
  
  


**Git push -all**

This command pushes all the branches to the server repository.

**Syntax**

1. $ git push --all

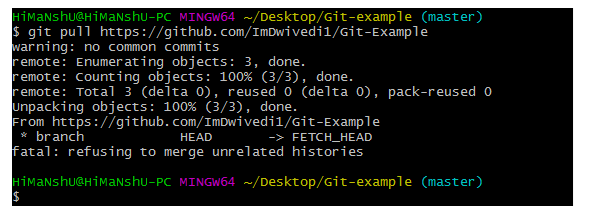


### Git pull command

Pull command is used to receive data from GitHub. It fetches and merges changes on the remote server to your working directory.

**Syntax**

1. $ git pull URL

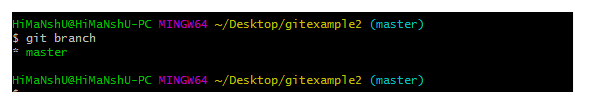


### Git Branch Command

This command lists all the branches available in the repository.

**Syntax**

1. $ git branch



### Git Merge Command

This command is used to merge the specified branch?s history into the current branch.

**Syntax**

1. $ git merge BranchName

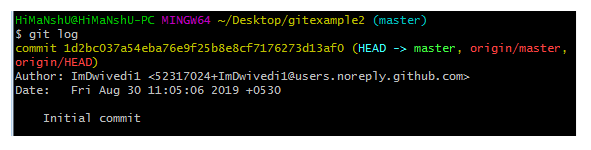


### Git log Command

This command is used to check the commit history.

**Syntax**

1. $ git log

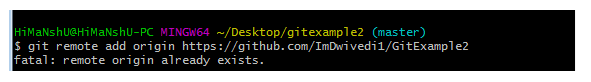


By default, if no argument passed, Git log shows the most recent commits first. We can limit the number of log entries displayed by passing a number as an option, such as -3 to show only the last three entries.

1. $ git log -3

### Git remote Command

Git Remote command is used to connect your local repository to the remote server. This command allows you to create, view, and delete connections to other repositories. These connections are more like bookmarks rather than direct links into other repositories. This command doesn't provide real-time access to repositories.



1. Git configuration

Git config

Get and set configuration variables that control all facets of how Git looks and operates.

Set the name:

$ git config --global user.name "User name"

Set the email:

$ git config --global user.email "himanshudubey481@gmail.com"

Set the default editor:

$ git config --global core.editor Vim

Check the setting:

$ git config -list

Git alias

Set up an alias for each command:

$ git config --global alias.co checkout

$ git config --global alias.br branch

$ git config --global alias.ci commit

$ git config --global alias.st status

2. Starting a project

Git init

Create a local repository:

$ git init

Git clone

Make a local copy of the server repository.

$ git clone

3. Local changes

Git add

Add a file to staging (Index) area:

$ git add Filename

Add all files of a repo to staging (Index) area:

$ git add\*

Git commit

Record or snapshots the file permanently in the version history with a message.

$ git commit -m " Commit Message"

4. Track changes

Git diff

Track the changes that have not been staged: $ git diff

Track the changes that have staged but not committed:

$ git diff --staged

Track the changes after committing a file:

$ git diff HEAD

Track the changes between two commits:

$ git diff Git Diff Branches:

$ git diff < branch 2>

Git status

Display the state of the working directory and the staging area.

$ git status

Git show Shows objects:

$ git show

5. Commit History

Git log

Display the most recent commits and the status of the head:

$ git log

Display the output as one commit per line:

$ git log -oneline

Displays the files that have been modified:

$ git log -stat

Display the modified files with location:

$ git log -p

Git blame

Display the modification on each line of a file:

$ git blame <file name>

6. Ignoring files

.gitignore

Specify intentionally untracked files that Git should ignore. Create .gitignore:

$ touch .gitignore List the ignored files:

$ git ls-files -i --exclude-standard

7. Branching

Git branch Create branch:

$ git branch List Branch:

$ git branch --list Delete a Branch:

$ git branch -d Delete a remote Branch:

$ git push origin -delete Rename Branch:

$ git branch -m

Git checkout

Switch between branches in a repository.

Switch to a particular branch:

$ git checkout

Create a new branch and switch to it:

$ git checkout -b Checkout a Remote branch:

$ git checkout

Git stash

Switch branches without committing the current branch. Stash current work:

$ git stash

Saving stashes with a message:

$ git stash save ""

Check the stored stashes:

$ git stash list

Re-apply the changes that you just stashed:

$ git stash apply

Track the stashes and their changes:

$ git stash show

Re-apply the previous commits:

$ git stash pop

Delete a most recent stash from the queue:

$ git stash drop

Delete all the available stashes at once:

$ git stash clear

Stash work on a separate branch:

$ git stash branch

Git cherry pic

Apply the changes introduced by some existing commit:

$ git cherry-pick

8. Merging

Git merge

Merge the branches:

$ git merge

Merge the specified commit to currently active branch:

$ git merge

Git rebase

Apply a sequence of commits from distinct branches into a final commit.

$ git rebase

Continue the rebasing process:

$ git rebase -continue Abort the rebasing process:

$ git rebase --skip

Git interactive rebase

Allow various operations like edit, rewrite, reorder, and more on existing commits.

$ git rebase -i

9. Remote

Git remote

Check the configuration of the remote server:

$ git remote -v

Add a remote for the repository:

$ git remote add Fetch the data from the remote server:

$ git fetch

Remove a remote connection from the repository:

$ git remote rm

Rename remote server:

$ git remote rename

Show additional information about a particular remote:

$ git remote show

Change remote:

$ git remote set-url

Git origin master

Push data to the remote server:

$ git push origin master Pull data from remote server:

$ git pull origin master

10. Pushing Updates

Git push

Transfer the commits from your local repository to a remote server. Push data to the remote server:

$ git push origin master Force push data:

$ git push -f

Delete a remote branch by push command:

$ git push origin -delete edited

11. Pulling updates

Git pull

Pull the data from the server:

$ git pull origin master

Pull a remote branch:

$ git pull

Git fetch

Download branches and tags from one or more repositories. Fetch the remote repository:

$ git fetch< repository Url> Fetch a specific branch:

$ git fetch

Fetch all the branches simultaneously:

$ git fetch -all

Synchronize the local repository:

$ git fetch origin

12. Undo changes

Git revert

Undo the changes:

$ git revert

Revert a particular commit:

$ git revert

Git reset

Reset the changes:

$ git reset -hard

$ git reset -soft:

$ git reset --mixed

13. Removing files

Git rm

Remove the files from the working tree and from the index:

$ git rm <file Name>

Remove files from the Git But keep the files in your local repository:

$ git rm --cached