

## Difference between Git and Github

Git is a distributed version-control system for tracking changes in source code during software development. It is designed for coordinating work among programmers, but it can be used to track changes in any set of files. Its goals include speed, data integrity, and support for distributed, non-linear workflows.

GitHub is a web-based hosting service for version control using Git. It is mostly used for computer code. It offers all of the distributed version control and source code management functionality of Git as well as adding its own features.

Git-hub is social network for code sharing and code collaboration.

**We can also install our own hosting Git Server.(Private Git repository)**

**Other Servers available besides GitHub are :-**

BitBucket, gitlab

## **Available Git GUI Clients**

1. GitKraken
2. SourceTree
3. Tower
4. GitHub Desktop

## Configuring a file in git Bash

$ git config

- It is the global configuration file, every git installation has a global config file, its the default configuration for everything we do with git.

Setting user.name and user.email

-These are the signatures used on every single commit on git

$ git config --list OR git config -l

- List all config settings

$ git config --global user.name "Gaurav Pawar"

$ git config --global user.email pawargaurav49@gmail.com

**Creating the project folder**

**-**Getting\_Started

-readme.md

**Initializing the git Repository**

-as soon as you start a project it’s good idea to initialize the git repository, this sets up git inside our project folder and it begins the tracking process, allowing us to commit and make changes and create branches etc.

**$ git init**

Initialized empty Git repository in C:/Users/hp/Desktop/Getting\_Started/.git/

**$ git status**

On branch master

No commits yet

Untracked files:

(use "git add <file>..." to include in what will be committed)

readme.md

nothing added to commit but untracked files present (use "git add" to track)

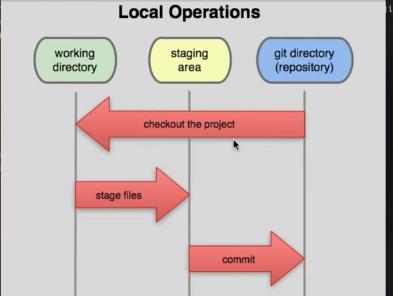
Difference between add and commit.

“add” adds a file to staging area which is temporary

and

“commit” commits a file to get repository which is permanent and have version control.

Staging area files comes under tracked files and will be committed on next git commit.



$ git add readme.md

$ git status

On branch master

No commits yet

Changes to be committed:

(use "git rm --cached <file>..." to unstage)

new file: readme.md

- readme.md is added to staging area and ready to be committed.

Creating a second file -

**$ touch secondFile.txt**

**$ git status**

On branch master

No commits yet

Changes to be committed:

(use "git rm --cached <file>..." to unstage)

new file: readme.md

Untracked files:

(use "git add <file>..." to include in what will be committed)

secondFile.txt

Adding second file to staging area

$ git add secondFile.txt

$ git status

On branch master

No commits yet

Changes to be committed:

(use "git rm --cached <file>..." to unstage)

new file: readme.md

new file: secondFile.txt

Committing staging area to git repository: -

$ git commit -m "Initial Commit - Adding readme files"

[master (root-commit) 4b052fc] Initial Commit - Adding readme files

2 files changed, 5 insertions(+)

create mode 100644 readme.md

create mode 100644 secondFile.txt

-m is used to give the message regarding the purpose of commit.

$ git status

On branch master

nothing to commit, working tree clean

Making our first commit involves us first understanding the three stages of git commits. All our work happens in the working directory (which is just a folder on our local computer). After we initialize the repository, Git will constantly watch and maintain the folder. We can get these updates from Git by running the "git status" command at any time. This command returns to us all the information we need to know what is happening in our git project right now. Files will either be untracked, in staging, or committed. Git status will tell us all of the untracked files which we can then move into our staging by using the "git add filename" command. This moves that file into the staging area, preparing to be committed. Of course we can add more files if needed into the staging area. As soon as the files we want to be in our commit are all in the staging area, then we use our "git commit -m "Message"" command to set the commit and save the snapshot of our project.

1. **Change the content of readme.md file**

$ git status

On branch master

Changes not staged for commit:

(use "git add <file>..." to update what will be committed)

(use "git checkout -- <file>..." to discard changes in working directory)

modified: readme.md

**3.** $ git add readme.md

**4.**

$ git status

On branch master

Changes to be committed:

(use "git reset HEAD <file>..." to unstage)

modified: readme.md

**5.**

$ git commit -m "Added email address to readme.md file"

[master 87a963b] Added email address to readme.md file

1 file changed, 5 insertions(+), 2 deletions(-)

**6.**

$ git status

On branch master

nothing to commit, working tree clean

## How to add multiple files at once to staging and how to see a log of our past commits.

$ git log

commit 87a963b312d10f3da35ad3815dd39639aadfdd5a (HEAD -> master)

Author: Gaurav Pawar <pawargaurav49@gmail.com>

Date: Mon May 6 04:54:46 2019 +0530

Added email address to readme.md file

commit 4b052fc8f25c65eac840a2a58332df671bb66f71

Author: Gaurav Pawar <pawargaurav49@gmail.com>

Date: Mon May 6 04:29:31 2019 +0530

Initial Commit - Adding readme files

-it shows all the history of previous commits we have made in this project so far.

### Creating a new Laravel project: -

$ laravel new getting\_started

$ cd getting\_started

$ git status

fatal: not a git repository (or any of the parent directories): .git

$ git init

Initialized empty Git repository in C:/Users/hp/Desktop/Laravel/getting\_started/.git/

$ git status

On branch master

No commits yet

Untracked files:

(use "git add <file>..." to include in what will be committed)

.editorconfig

.env.example

.gitattributes

.gitignore

.styleci.yml

app/

artisan

bootstrap/

composer.json

composer.lock

config/

database/

package.json

phpunit.xml

public/

resources/

routes/

server.php

storage/

tests/

webpack.mix.js

yarn.lock

nothing added to commit but untracked files present (use "git add" to track)

**$ git add .**

It will add all files to the staging area and all files are now being tracked.

$git add “\*.php” - This will add only all php files to the staging area.

$ git status

On branch master

No commits yet

Changes to be committed:

(use "git rm --cached <file>..." to unstage)

$ git commit -m "Initial Commit"

[master (root-commit) 174edc9] Initial Commit

88 files changed, 13366 insertions(+)

$ git status

On branch master

nothing to commit, working tree clean

## Ignoring some files to prevent them from getting into our repository.

**.gitignore  - file used to mention those files which needs to be ignored by git**

$ touch .gitignore

Add those files you want git to ignore and not stage and commit.

GitIgnore Files on GitHub [https://github.com/github/gitignore](https://www.youtube.com/redirect?redir_token=ynBjydx6PRuDKRQcMi9hqkXEEtJ8MTU1NzE4NzMyNkAxNTU3MTAwOTI2&v=0WfDe51pUU0&q=https://github.com/github/gitignore&event=video_description" \t "https://www.youtube.com/_blank) (Every framework imaginable is on here)

## Pushing our Git Project to Git Hub

### …or create a new repository on the command line

echo "# GittingStarted" >> README.md

git init

git add README.md

git commit -m "first commit"

git remote add origin https://github.com/GauravSinghPawar/GittingStarted.gitgit push -u origin master

### …or push an existing repository from the command line

git remote add origin <https://github.com/GauravSinghPawar/GittingStarted.git>

git push -u origin master

origin = name we calling our remote URL

We can have multiple remote add eg., one for production server, one for staging server, one where we sore our source code like github,

git remote add production ----

git remote add staging ---

git remote add live ---

and we can push code on which ever server we want.e

git push -u production master

git push -u staging master

git push -u live master

-u = save this as the default

Master = the branch

### clone (copy) and fork projects from github to our computer.

$ git clone https://github.com/GauravSinghPawar/GittingStarted.git

That is all it takes to copy down not only all of the code, but the entire history of commits, branches and all the same data we would have if we started the project on our own computer.