

Introduction to Git and GitHub

What is GIT?

- Git is a **free and open-source distributed version control system** designed to handle everything from small to very large projects with speed and efficiency.
- Git relies on the **basis of distributed development** of software where more than one developer may have access to the source code of a specific application and can modify changes to it that may be seen by other developers..
- Initially designed and developed by Linus Torvalds for Linux kernel development in 2005.
- Every git working directory is a **full-fledged repository** with complete history and full version tracking capabilities, independent of network access or a central server.
- Git **allows a team of people to work together**, all using the same files. And it helps **the team cope with the confusion** that tends to happen **when multiple people are editing the same files**.

Why Use Version Control Software?

- Version control software allows the user to have “versions” of a project, which show the changes that were made to the code over time, and allows the user to backtrack if necessary and undo those changes.
- This **ability alone – of being able to compare two versions or reverse changes**, makes it fairly invaluable when working on larger projects.
- In a version control system, **the changes would be saved just in time** – a patch file that could be applied to one version, in order to make it the same as the next version.
- All **versions are stored on a central server**, and individual developers checkout and upload changes back to this server.

Characteristics of Git

A. Strong support for non-linear development

- Git supports rapid branching and merging and includes specific tools for visualizing and navigating a non-linear development history.
- A major assumption in Git is that a change will be merged more often than it is written.
- **Branches** in Git **are very lightweight**.

B. Distributed development

- Git **provides** each developer a **local copy** of the entire development history, and changes are copied from one such repository to another.
- The changes can be merged in the same way as a locally developed branch very efficiently and effectively.

C. Compatibility with existing systems/protocol

- Git has a CVS server emulation, which enables the use of existing CVS clients and IDE plugins to access Git repositories.

D. Efficient handling of large projects

- Git is **very fast and scalable** compared to other version control systems.
- The **fetching power from a local repository is much faster than** is possible with a **remote server**.

E. Data Assurance

- The Git history is stored in such a way that **the ID** of a particular version **depends** upon the **complete development** history leading up to that commit.
- Once published, it is not possible to change the old versions without them being noticed.

F. Automatic Garbage Collection

- Git automatically performs **garbage collection** when enough loose objects have been created in the repository.

- Garbage collection can be called explicitly using git ***gc –prune***.

G. Periodic explicit object packing

- Git stores each newly created object as a separate file. It uses *packs* that store a large number of objects in a single file (or network byte stream) called packfile, delta-**compressed** among themselves.
- A corresponding index file is created for each pack file, **specifying** the **offset** of each object in the packfile.
- The process of **packing** can be very **expensive computationally**.
- Git allows the expensive pack operation to be deferred until later when time does not matter.
- Git **does periodic repacking automatically** but manual repacking can be done with the git gc command.

How does GIT work?

1. A Git repository is a **key-value** object store where all objects are indexed by their SHA-1 hash value.
2. All commits, files, tags, and filesystem tree nodes are different types of objects living in this repository.
3. A Git **repository is a large** hash table with **no provision** made for **hash collisions**.
4. Git specifically works by taking “**snapshots**” of files

What is GitHub about?

GitHub is an online platform where we can share our codes (or projects) online hassle-free. GitHub is place where we host our local git repository online. it basically allow you to work collaboratively within a group of peoples. It hosts all the features of Git and have its own too.

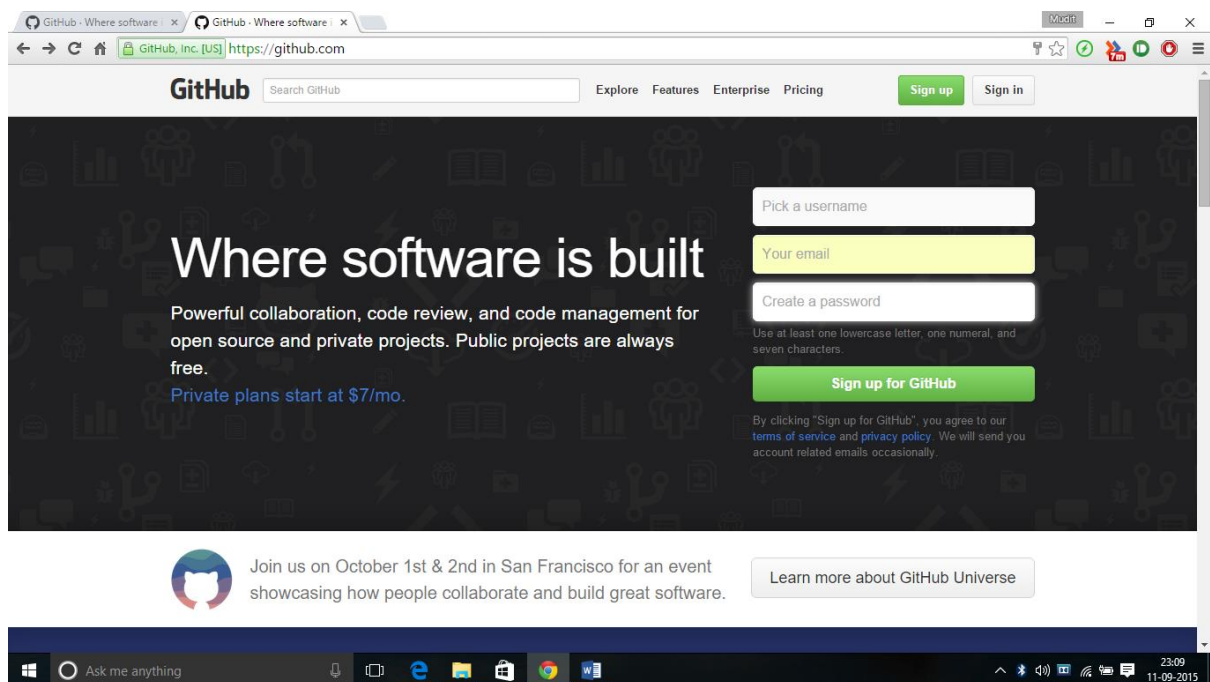
In simple words, GitHub might be considered as a social media which is made for the developers where they share their work. it might be any project regarding website development or any design of website, or some operating systems like Android, or Linux, etc.

GitHub is owned by Microsoft, provides access to public(free) and private(paid) repositories. It allows free repository to host files up to 100mb and total size for 2GB.

Let's us see how to host to a local repository to GitHub, from very beginning(creating a GitHub account).

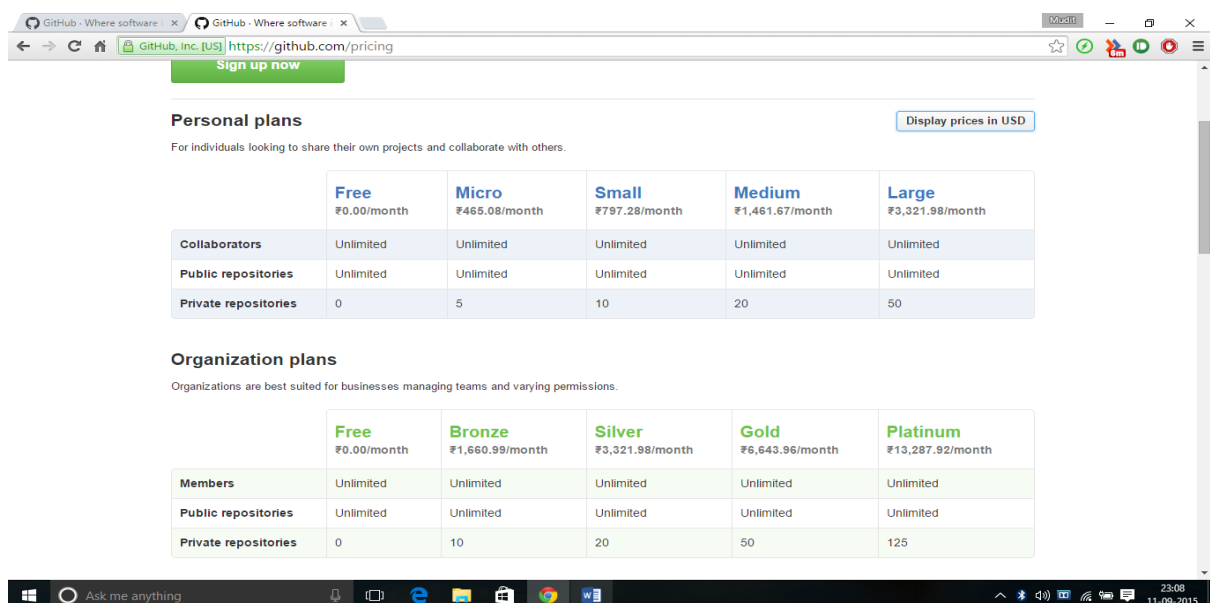
A. Creating a GitHub Account

Step 1: Go to GitHub.com and enter the required user credentials asked on the site and then click on the SignUp for GitHub button.



The screenshot shows the GitHub homepage in a web browser. The URL is https://github.com. The page features the GitHub logo, a search bar, and navigation links for Explore, Features, Enterprise, and Pricing. A prominent green 'Sign up' button is visible. The main content area has the heading 'Where software is built' and a subheading 'Powerful collaboration, code review, and code management for open source and private projects. Public projects are always free. Private plans start at \$7/mo.' To the right, there is a sign-up form with fields for 'Pick a username', 'Your email', and 'Create a password'. Below the form, there is a green 'Sign up for GitHub' button. A note below the button states: 'By clicking "Sign up for GitHub", you agree to our terms of service and privacy policy. We will send you account related emails occasionally.'

Step 2: Choose a plan that best suits you. The following plans are available as shown in below media as depicted:



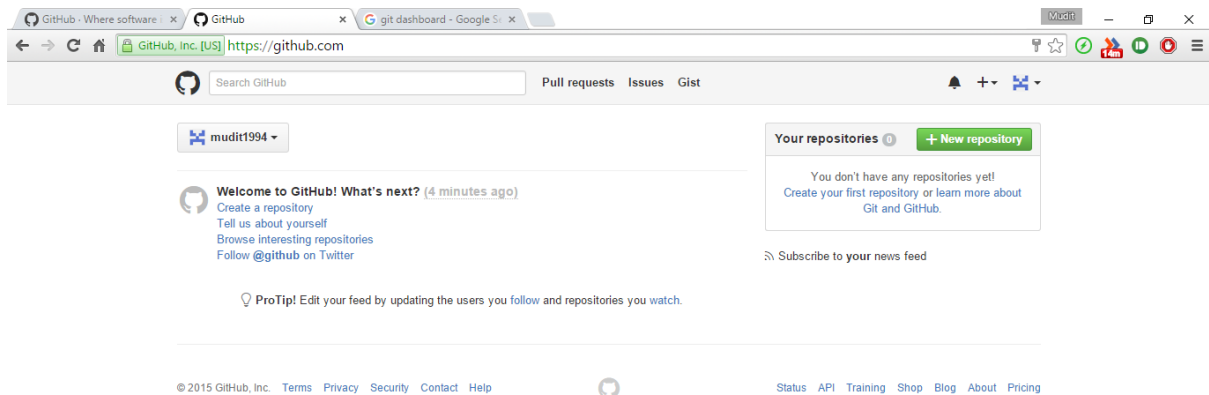
The screenshot shows the GitHub pricing page. It features a 'Sign up now' button at the top. Below it, there is a section for 'Personal plans' with a 'Display prices in USD' button. The 'Personal plans' section includes a table with five columns: Free, Micro, Small, Medium, and Large. The 'Free' plan is highlighted in green. Below the 'Personal plans' section, there is a section for 'Organization plans' with a table showing five plans: Free, Bronze, Silver, Gold, and Platinum. The 'Free' plan is highlighted in green.

	Free ₹0.00/month	Micro ₹465.08/month	Small ₹797.28/month	Medium ₹1,461.67/month	Large ₹3,321.98/month
Collaborators	Unlimited	Unlimited	Unlimited	Unlimited	Unlimited
Public repositories	Unlimited	Unlimited	Unlimited	Unlimited	Unlimited
Private repositories	0	5	10	20	50

	Free ₹0.00/month	Bronze ₹1,660.99/month	Silver ₹3,321.98/month	Gold ₹6,643.96/month	Platinum ₹13,287.92/month
Members	Unlimited	Unlimited	Unlimited	Unlimited	Unlimited
Public repositories	Unlimited	Unlimited	Unlimited	Unlimited	Unlimited
Private repositories	0	10	20	50	125

Step 3: Then Click on Finish Sign Up.

The account has been created. The user is automatically redirected to your Dashboard.



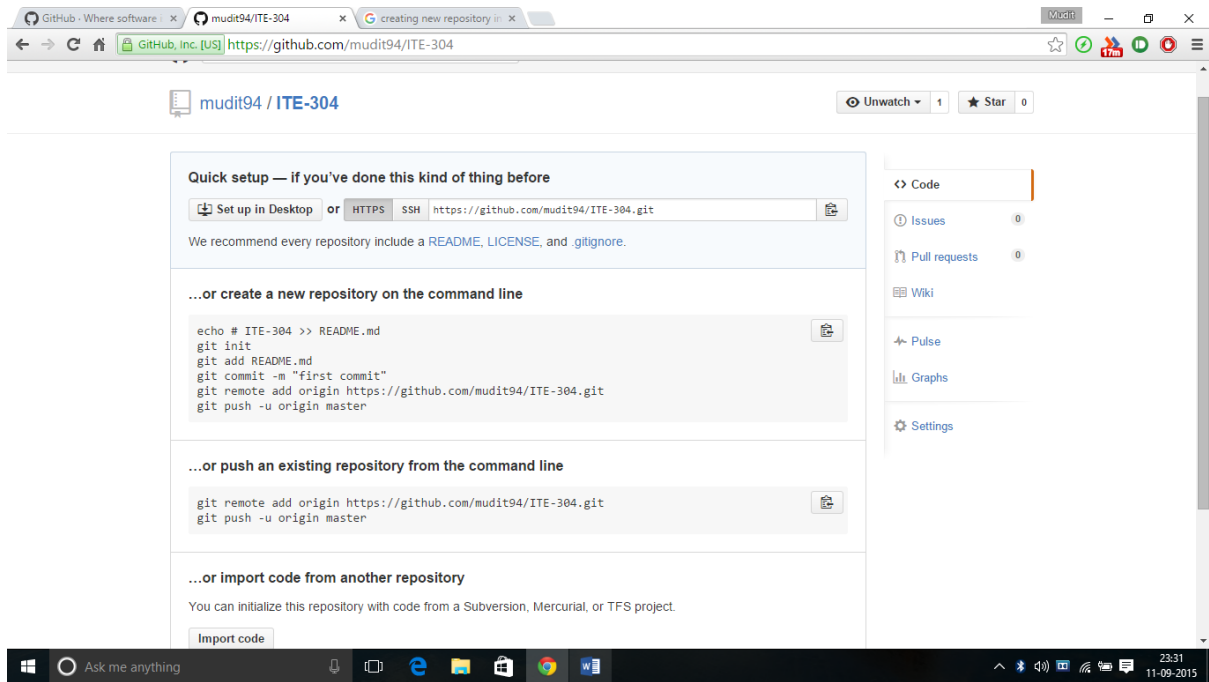
B. Creating a new Repository

- Login to your GitHub account
- On the dashboard click on the Green Button starting New repository.
- Make sure to **verify the GitHub account** by going into the mail which was provided when creating the account.
- Once verification has been done, the following screen comes

C. Start by giving a repository name, description(optional) and select the visibility and accessibility mode for the repository

D. Click on Create repository

E. The repository (in this case ITE-304 is the repository) is now created. The repository can be created looks like:



Uploading Existing Repository to GitHub

- The **system should have git installed** in it if not install git. Make sure to choose Run git from Windows Command prompt option during installation. Otherwise, open git bash in place of step 2.
- Open Terminal (for Mac users) or the command prompt (for Windows and Linux users).
- Change the current working directory to your local project
- Initialize the local directory as a git repository in different ways as described in the image.

git init

- A new **.git folder** is **created** in the directory which is by **default hidden**.
- Add the files in your new local repository. This stages them for the first commit.

git add .

#Adds the files in the local repository and stages them for commit. To unstage a file, use

```
git commit -m 'First commit'
```

Commits the tracked changes and prepares them to be pushed to a remote repository.

To remove this commit and modify the file, use

```
git reset --soft HEAD~1
```

And commit and add the file again.

At the top of the GitHub repository's Quick Setup page, click on the icon shown and copy the remote repository URL.



In the Command prompt, **add the URL for the remote repository** where your local repository will be pushed.

```
git remote add origin remote repository URL
```

Connects to the remote repository

```
git remote -v
```

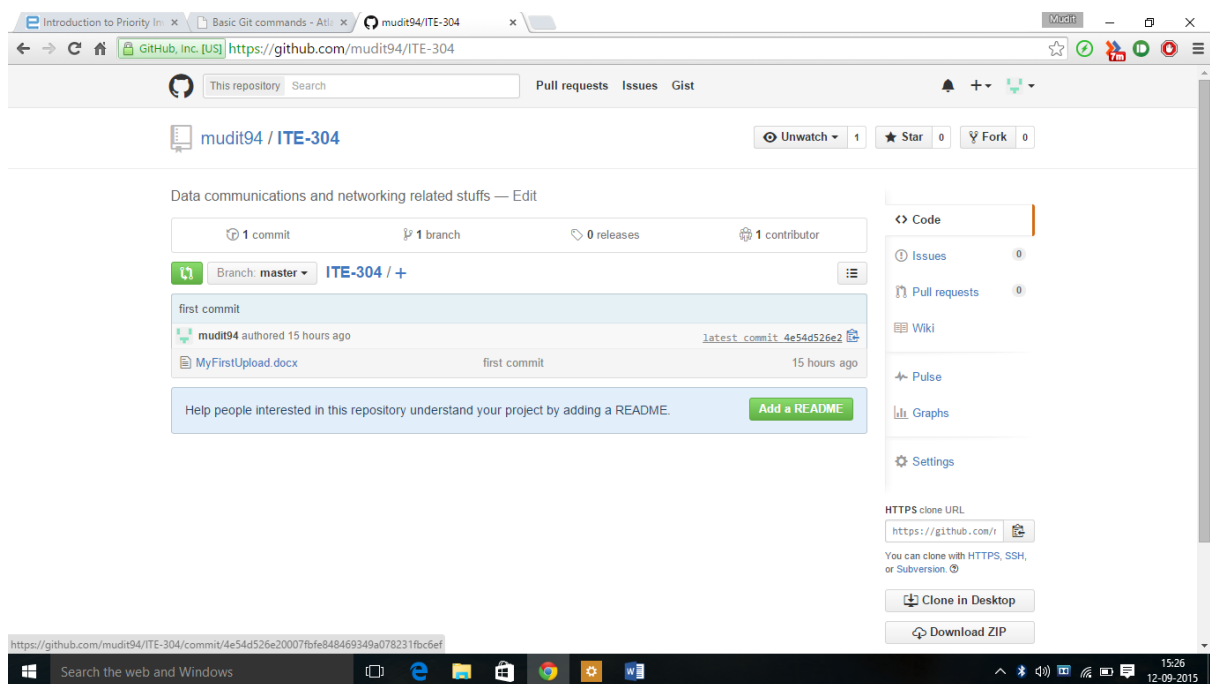
Verifies the new remote URL

Push the changes in your local repository to [GitHub](https://github.com).

`git push origin master`

Pushes the changes in your local repository up to the remote repository you specified as the origin.

And here you go...



You may download changes from remote repository to local repository using the command:

`git pull`