

# Git

Git : Now, git is nothing but is a free and open source distributed version control system. which is designed to handle everything from small to very large project with speed and efficiency

- Git is easy to learn and has tiny footprint with lightning fast performance
- Git is released under the GNU General Public License version 2.0, which is an open source license.
- Git is very fast. With Git, nearly all operations are performed locally, giving it a huge speed advantage on centralized systems that constantly have to communicate with a server somewhere.
- Git was built to work on the Linux kernel, meaning that it has had to effectively handle large repositories from day one.
- Git is written in C, reducing the overhead of runtimes associated with higher-level languages.
- Speed and performance has been a primary design goal of the Git from the start. another feature of git is that git has integrity: Now what does it mean? Everything in Git is checksummed before it is stored and is then referred to by that checksum. This means it's impossible to change the contents of any file or directory without Git knowing about it. This functionality is built into Git at the lowest levels and is integral to its philosophy. You can't lose information in transit or get file corruption without Git being able to detect it.
- The another phase is that git is only adds data : When you do actions in Git, nearly all of them only add data to the Git database. It is hard to get the system to do anything that is not undoable or to make it erase data in any way. As with any VCS, you can lose or mess up changes you haven't committed yet, but after you commit a snapshot into Git, it is very difficult to lose, especially if you regularly push your database to another repository.
- Now, moving further about the core part of git is that the three states. if you want the rest of your learning process to go smoothly.
- Git has three main states that your files can reside in: committed, modified, and staged:
  1. Committed means that the data is safely stored in your local database.
  2. Modified means that you have changed the file but have not committed it to your database yet.
  3. Staged means that you have marked a modified file in its current version to go into your next commit snapshot.
- This leads us to the three main sections of a Git project: the Git directory, the working tree, and the staging area.
- The Git directory is where Git stores the metadata and object database for your project. This is the most important part of Git, and it is what is copied when you clone a repository from another computer.

The working tree is a single checkout of one version of the project. These files are pulled out of the compressed database in the Git directory and placed on disk for you to use or modify.

The staging area is a file, generally contained in your Git directory, that stores information about what will go into your next commit.

The basic git workflows goes on like as below:

1. You modify files in your working tree.
2. You selectively stage just those changes you want to be part of your next commit, which adds only those changes to the staging area.
3. You do a commit, which takes the files as they are in the staging area and stores that snapshot permanently to your Git directory.

If any particular version of a file in the git directory, It is considered as committed. if it has been modified and was added to the staging area, it is staged. And if it was changed since it was checked out but has not been staged ,it is modified.

## Advantage:

### 1. Branching and Merging

The Git feature that really makes it stand apart from nearly every other SCM out there is its branching model.

Git allows and encourages you to have multiple local branches that can be entirely independent of each other. The creation, merging, and deletion of those lines of development takes seconds.

### 2. Small and Fast

**Git is fast.** With Git, nearly all operations are performed locally, giving it a huge speed advantage on centralized systems that constantly have to communicate with a server somewhere.

Git was built to work on the Linux kernel, meaning that it has had to effectively handle large repositories from day one. Git is written in C, reducing the overhead of runtimes associated with higher-level languages. Speed and performance has been a primary design goal of the Git from the start.

### 3. Distributed

One of the nicest features of any Distributed SCM, Git included, is that it's distributed. This means that instead of doing a "checkout" of the current tip of the source code, you do a "clone" of the entire repository.

### 4. Data Assurance

The data model that Git uses ensures the cryptographic integrity of every bit of your project. Every file and commit is checksummed and retrieved by its checksum when checked back out. It's impossible to get anything out of Git other than the **exact bits you put in**.

### 5. Free and Open Source

Git is released under the [GNU General Public License version 2.0](#), which is an [open source license](#). The Git project chose to use GPLv2 to guarantee your freedom to share and change free software---to make sure the software is free for all its users.

### 6. Staging Area

Unlike the other systems, Git has something called the "staging area" or "index". This is an intermediate area where commits can be formatted and reviewed before completing the commit.

# GitHub

## Introduction:

- 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 (SCM) functionality of Git as well as adding its own features.
- It provides access control and several collaboration features such as bug tracking, feature requests, task management, and wikis for every project.
- GitHub offers plans for enterprise, team, pro and free accounts which are commonly used to host open-source software projects.

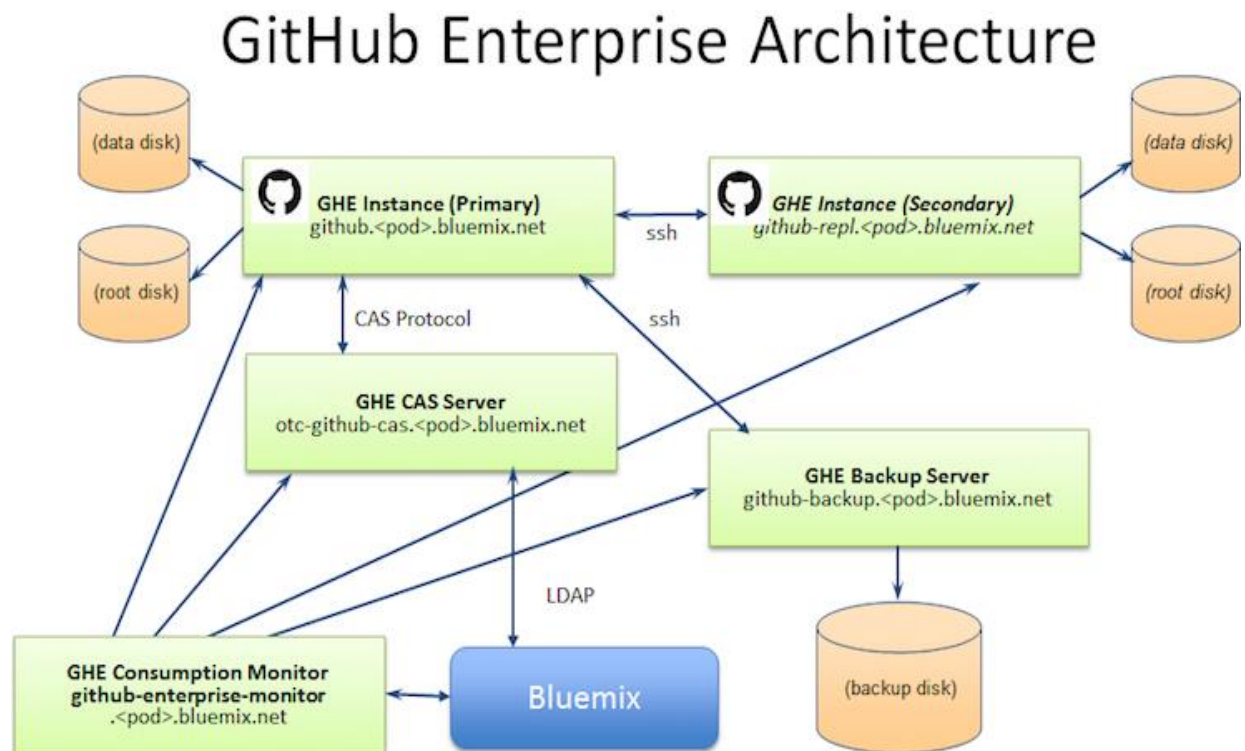
## Scope:

- Documentation, including automatically rendered README files in a variety of Markdown-like file formats
- Issue tracking (including feature requests) with labels, milestones, assignees and a search engine
- Wikis
- Pull requests with code review and comments
- Commits history
- Graphs: pulse, contributors, commits, code frequency, punch card, network, members
- Integrations Directory
- Unified and split diffs
- Email notifications
- Option to subscribe someone to notifications by @ mentioning them.
- Emojis
- GitHub Pages: small websites can be hosted from public repositories on GitHub. The URL format is `https://username.github.io`.
- Nested task-lists within files
- Visualization of geospatial data
- 3D render files that can be previewed using a new integrated STL file viewer that displays the files on a "3D canvas". The viewer is powered by WebGL and Three.js.
- Photoshop's native PSD format can be previewed and compared to previous versions of the same file.
- PDF document viewer
- Security Alerts of known Common Vulnerabilities and Exposures in different packages

## Product and feature:

- GitHub offers an on-premises version in addition to the well-known SaaS product. GitHub Enterprise supports integrated development environments and continuous integration tool integration, as well as a litany of third-party apps and services. It offers increased security and auditability than the SaaS version.
- Github Gist allows GitHub users to share pieces of code or other notes.
- GitHub Flow is a lightweight, branch-based workflow for regularly updated deployments.
- GitHub Pages are static webpages to host a project, pulling information directly from an individual's or organization's GitHub repository.
- GitHub Desktop enables users to access GitHub from Windows or Mac desktops, rather than going to GitHub's website.
- GitHub Student Developer Pack is a free offering of developer tools that is limited to students, and includes cloud resources, programming tools and support, and GitHub access.

## Architecture:



## How to Create an Account on GitHub :

1. Go to the GitHub sign up page.

The screenshot shows the GitHub sign-up process at Step 1: Create personal account. The page has a progress bar at the top with three steps: Step 1 (active), Step 2: Choose your plan, and Step 3: Tailor your experience. The main form area is titled 'Create your personal account' and contains three input fields: Username, Email Address, and Password. The Username field is empty, with a hint below it stating 'This will be your username — you can enter your organization's username next.' The Email Address field is also empty, with a hint below it stating 'You will occasionally receive account related emails. We promise not to share your email with anyone.' The Password field is empty, with a hint below it stating 'Use at least one lowercase letter, one numeral, and seven characters.' Below the password field, there is a line of text: 'By clicking on "Create an account" below, you are agreeing to the [Terms of Service](#) and the [Privacy Policy](#).' At the bottom of the form is a green button labeled 'Create an account'. To the right of the form, there is a section titled 'You'll love GitHub' with a list of benefits: 'Unlimited collaborators', 'Unlimited public repositories', 'Great communication', 'Frictionless development', and 'Open source community'.

2. Enter a username, valid email address, and password. Use at least one lowercase letter, one numeral, and seven characters.

The screenshot shows the GitHub sign-up process at Step 1: Create personal account, with the fields filled in. The Username field contains 'wikihowtest' and has a green checkmark to its right. The Email Address field contains 'wikihow@mail.com' and has a green checkmark to its right. The Password field contains '.....' and has a green checkmark to its right. The hints below the fields are the same as in the previous screenshot. The line of text below the password field is highlighted in yellow: 'By clicking on "Create an account" below, you are agreeing to the [Terms of Service](#) and the [Privacy Policy](#).' At the bottom of the form is a green button labeled 'Create an account', which is also highlighted with a yellow arrow pointing to it from the left.

3. Review carefully the GitHub Terms of Service and Privacy Policy before continuing. Upon clicking the "Create an account" button you will simultaneously be agreeing to these documents.
4. **Choose a plan.** You have two choice: Free and paid, the paid version has private repositories with \$7/month. You should try the free version then have the suitable choice.

### Choose your personal plan



☒ Unlimited public repositories for free.

☐ Unlimited private repositories for \$7/month.

Don't worry, you can cancel or upgrade at any time.




#### ☐ Help me set up an organization next

Organizations are separate from personal accounts and are best suited for businesses who need to manage permissions for many employees.

[Learn more about organizations.](#)



5. **Tailor experience.** If you have time, fill in the survey or skip it.

 <b>Completed</b> Set up a personal account	 <b>Step 2:</b> Choose your plan	 <b>Step 3:</b> Tailor your experience
---	--	--

How would you describe your level of programming experience?

☐ Totally new to programming
 ☐ Somewhat experienced
 ☐ Very experienced

What do you plan to use GitHub for? (check all that apply)


☐ Design
 ☐ Development
 ☐ Project Management  
☐ School projects
 ☐ Research
 ☐ Other (please specify)

Which is closest to how you would describe yourself?

☐ I'm a student
 ☐ I'm a hobbyist
 ☐ I'm a professional  
☐ Other (please specify)

What are you interested in?

e.g. tutorials, android, ruby, web-development, machine-learning, open-source


[skip this step](#)

[Sign in now to](#)

6. **You finished!** Your Github account created!

## Create a Repository:

A **repository** is usually used to organize a single project. Repositories can contain folders and files, images, videos, spreadsheets, and data sets – anything your project needs.

We recommend including a *README*, or a file with information about your project.




GitHub makes it easy to add one at the same time you create your new repository. *It also offers other common options such as a license file.*

## To create a new repository

1. In the upper right corner, next to your avatar or identicon, click and then select **New repository**.
2. Name your repository `hello-world`.
3. Write a short description.
4. Select **Initialize this repository with a README**.





**Owner** **Repository name**

PUBLIC   **hubot** /  


Great repository names are short and memorable. Need inspiration? How about **petulant-shame**.

**Description** (optional)

☒  **Public**  
Anyone can see this repository. You choose who can commit.

☐  **Private**  
You choose who can see and commit to this repository.

☒ **Initialize this repository with a README**  
This will allow you to `git clone` the repository immediately. Skip this step if you have already run `git init` locally.

|  

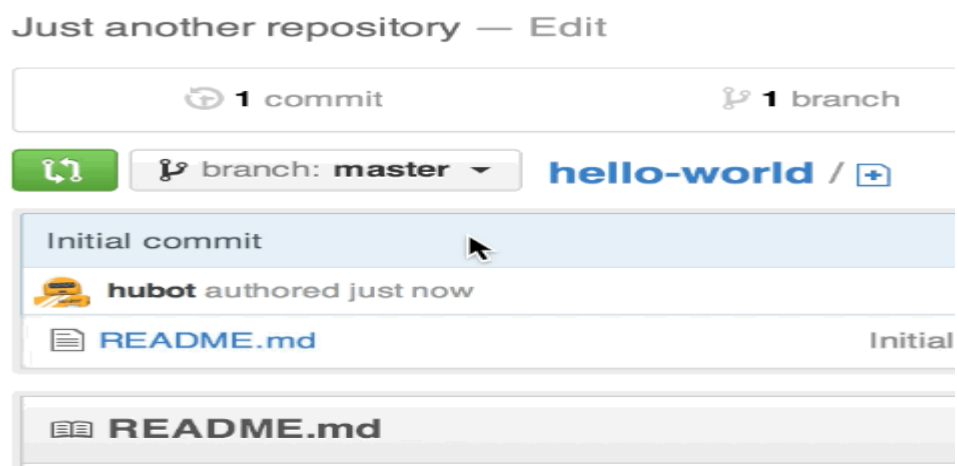
**Create repository**

## Create a Branch

**Branching** is the way to work on different versions of a repository at one

To create a new branch

1. Go to your new repository `hello-world`.
2. Click the drop down at the top of the file list that says branch: master.
3. Type a branch name, `readme-edits`, into the new branch text box.
4. Select the blue Create branch box or hit "Enter" on your keyboard.



## Make and commit changes

On GitHub, saved changes are called *commits*. Each commit has an associated *commit message*, which is a description explaining why a particular change was made. Commit messages capture the history of your changes, so other contributors can understand what you've done and why.

### Make and commit changes

1. Click the `README.md` file.
2. Click the pencil icon in the upper right corner of the file view to edit.
3. In the editor, write a bit about yourself.
4. Write a commit message that describes your changes.
5. Click Commit changes button.

The screenshot shows the GitHub web interface for the 'hubot / hello-world' repository. The 'Code' tab is selected, and the 'README.md' file is open in the editor. The file content is as follows:

```
1 # hello-world
2
3 Hi Humans!
4
5 Hubot here, I like Node.js and Coffeescript (that's what I'm made of!).
6 I've had tacos on the moon and find them far superior to Earth tacos.
7
```

Below the editor, the 'Commit changes' section is visible. The commit message is 'Finish README' and 'And mention moon tacos'. The 'Commit directly to the readme-edits branch' option is selected. The 'Commit changes' button is highlighted in green.

