# **What is GitHub?**

GitHub is a Git repository hosting service. GitHub also facilitates with many of its features, such as access control and collaboration. It provides a Web-based graphical interface.

GitHub is an American company. It hosts source code of your project in the form of different programming languages and keeps track of the various changes made by programmers.

It offers both **distributed version control and source code management (SCM)** functionality of Git. It also facilitates with some collaboration features such as bug tracking, feature requests, task management for every project.



## Features of GitHub

GitHub is a place where programmers and designers work together. They collaborate, contribute, and fix bugs together. It hosts plenty of open source projects and codes of various programming languages.

## Git



There are many words to define [git](https://www.javatpoint.com/git), but it is an open-source distributed version control system in simpler words.

There are many words to define [git](https://www.javatpoint.com/git), but it is an open-source distributed version control system in simpler words.

# **Git Version Control System**

A version control system is a software that tracks changes to a file or set of files over time so that you can recall specific versions later. It also allows you to work together with other programmers.

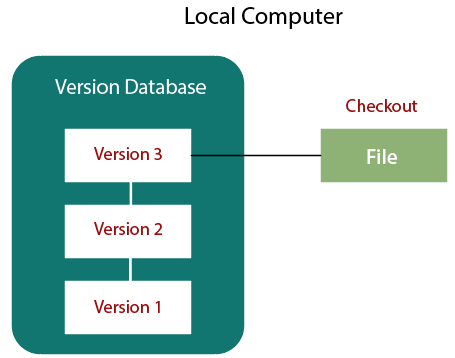
The version control system is a collection of software tools that help a team to manage changes in a source code. It uses a special kind of database to keep track of every modification to the code.

Developers can compare earlier versions of the code with an older version to fix the mistakes.

Types of Version Control System

* Localized version Control System
* Centralized version control systems
* Distributed version control systems

### **Localized Version Control Systems**



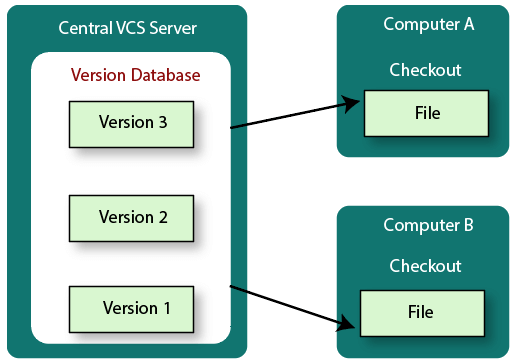
The localized version control method is a common approach because of its simplicity. But this approach leads to a higher chance of error. In this approach, you may forget which directory you're in and accidentally write to the wrong file or copy over files you don't want to.

To deal with this issue, programmers developed local VCSs that had a simple database. Such databases kept all the changes to files under revision control. A local version control system keeps local copies of the files.

The major drawback of Local VCS is that it has a single point of failure.

### **Centralized Version Control System**

The developers needed to collaborate with other developers on other systems. The localized version control system failed in this case. To deal with this problem, Centralized Version Control Systems were developed.



These systems have a single server that contains the versioned files, and some clients to check out files from a central place.

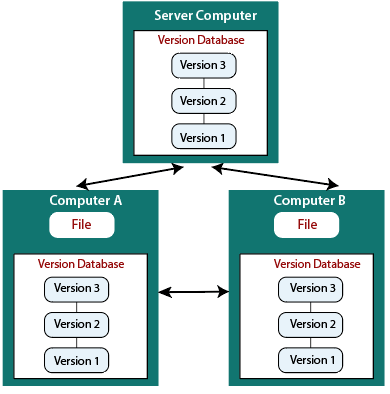
Centralized version control systems have many benefits, especially over local VCSs.

* Everyone on the system has information about the work what others are doing on the project.
* Administrators have control over other developers.
* It is easier to deal with a centralized version control system than a localized version control system.
* A local version control system facilitates with a server software component which stores and manages the different versions of the files.

### **Distributed Version Control System**

Centralized Version Control System uses a central server to store all the database and team collaboration. But due to single point failure, which means the failure of the central server, developers do not prefer it. Next, the Distributed Version Control System is developed.

In a Distributed Version Control System (such as Git, Mercurial, Bazaar or Darcs), the user has a local copy of a repository. So, the clients don't just check out the latest snapshot of the files even they can fully mirror the repository. The local repository contains all the files and metadata present in the main repository.



DVCS allows automatic management branching and merging. It speeds up of most operations except pushing and pulling. DVCS enhances the ability to work offline and does not rely on a single location for backups. If any server stops and other systems were collaborating via it, then any of the client repositories could be restored by that server. Every checkout is a full backup of all the data.

These systems do not necessarily depend on a central server to store all the versions of a project file.

## Difference between Centralized Version Control System and Distributed Version Control System

Centralized Version Control Systems are systems that use **client/server** architecture. In a centralized Version Control System, one or more client systems are directly connected to a central server. Contrarily the Distributed Version Control Systems are systems that use **peer-to-peer** architecture.

There are many benefits and drawbacks of using both the version control systems. Let's have a look at some significant differences between Centralized and Distributed version control system.

|  |  |
| --- | --- |
| **Centralized Version Control System** | **Distributed Version Control System** |
| In CVCS, The repository is placed at one place and delivers information to many clients. | In DVCS, Every user has a local copy of the repository in place of the central repository on the server-side. |
| It is based on the client-server approach. | It is based on the client-server approach. |
| It is the most straightforward system based on the concept of the central repository. | It is flexible and has emerged with the concept that everyone has their repository. |
| In CVCS, the server provides the latest code to all the clients across the globe. | In DVCS, every user can check out the snapshot of the code, and they can fully mirror the central repository. |
| CVCS is easy to administrate and has additional control over users and access by its server from one place. | DVCS is fast comparing to CVCS as you don't have to interact with the central server for every command. |
| The popular tools of CVCS are **SVN** (Subversion) and **CVS**. | The popular tools of DVCS are **Git** and **Mercurial**. |
| CVCS is easy to understand for beginners. | DVCS has some complex process for beginners. |
| If the server fails, No system can access data from another system. | if any server fails and other systems were collaborating via it, that server can restore any of the client repositories |

# **What is GitHub?**

GitHub is a Git repository hosting service. GitHub also facilitates with many of its features, such as access control and collaboration. It provides a Web-based graphical interface.

GitHub is an American company. It hosts source code of your project in the form of different programming languages and keeps track of the various changes made by programmers.

It offers both **distributed version control and source code management (SCM)** functionality of Git. It also facilitates with some collaboration features such as bug tracking, feature requests, task management for every project.



## Features of GitHub

GitHub is a place where programmers and designers work together. They collaborate, contribute, and fix bugs together. It hosts plenty of open source projects and codes of various programming languages.

Play Video[](https://campaign.adpushup.com/get-started/?utm_source=banner%26utm_campaign=growth_hack)

Some of its significant features are as follows.

* Collaboration
* Integrated issue and bug tracking
* Graphical representation of branches
* Git repositories hosting
* Project management
* Team management
* Code hosting
* Track and assign tasks
* Conversations
* Wikisc

## Benefits of GitHub

GitHub can be separated as the Git and the Hub. GitHub service includes access controls as well as collaboration features like task management, repository hosting, and team management.

The key benefits of GitHub are as follows.

* It is easy to contribute to open source projects via GitHub.
* It helps to create an excellent document.
* You can attract recruiter by showing off your work. If you have a profile on GitHub, you will have a higher chance of being recruited.
* It allows your work to get out there in front of the public.
* You can track changes in your code across versions.

# **Difference between git and gitHub**

Programming language wordings are very intuitive these days. By hearing the name of a particular language, we start imagining what all it will be.

[Java](https://www.javatpoint.com/java-tutorial) and [Javascript](https://www.javatpoint.com/javascript-tutorial) are very similar to the names ham and hamster, the logo of [python](https://www.javatpoint.com/python-tutorial) is intertwined with the image of snakes.



So, someone looking at git and github would find any apparent connection between them. Let us see git and github in detail with the differences between them.

## Git



There are many words to define [git](https://www.javatpoint.com/git), but it is an open-source distributed version control system in simpler words.

There are many words to define [git](https://www.javatpoint.com/git), but it is an open-source distributed version control system in simpler words.

Let us break each component in the definition and understand it.

* **Open-source -** A type of computer software released under a specific license. The users are given permissions to use the code, modify the code, give suggestions, clone the code to add new functionality. In other words, if the software is open-source, it is developed collaboratively in a public manner. The open-source softwares is cheaper, more flexible, and lasts longer than an authority or a company. The products in the source code include code, documents, formats for the users to understand and contribute to it. Using open-source a project can be expanded to update or revise the current features. Unix and Linux are examples of open-source softwares.
* **Control system -** The work of a control system is to track the content. In other words, git is used to storing the content to provide the services and features to the user.
* **Version Control system -** Just like an app has different updates due to bugs and additional feature addition, version changes, git also supports this feature. Many developers can add their code in parallel. So the version control system easily manages all the updates that are done previously.  
  Git provides the feature of branching in which the updated code can be done, and then it can be merged with the main branch to make it available to the users. It not only makes everything organized but keeps synchronization among the developers to avoid any mishap. Other examples of version control systems are Helix core, Microsoft TFS, etc.
* **Distributed version control system -** Here distributed version control system means if a developer contributes to open source, the code will also be available in his remote repository. The developer changes his local repository and then creates a pull request to merge his changes in the central repository. Hence, the word distributed means the code is stored in the central server and stored in every developer's remote system.

**Why is git needed?**

When a team works on real-life projects, git helps ensure no code conflicts between the developers. Furthermore, the project requirements change often. So a git manages all the versions. If needed, we can also go back to the original code. The concept of branching allows several projects to run in the same codebase.

GitHub

By the name, we can visualize that it is a Hub, projects, communities, etc. [GitHub](https://www.javatpoint.com/github) is a [Git repository](https://www.javatpoint.com/git-repository) hosting service that provides a web-based graphical interface. It is the largest community in the world. Whenever a project is open-source, that particular repository gains exposure to the public and invites several people to contribute.

The source code of several projects is available on github which developers can use in any means.

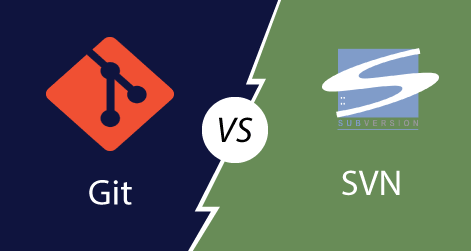
Using github, many developers can work on a single project remotely because it facilitates collaboration.

**Features of gitHub**

* Using github the project managers can collaborate, review and guide the developers regarding any changes. This makes project management easy.
* The github repositories can be made public or private. Thus allowing safety to an organization in case of a project.
* GitHub has a feature of pull requests and issues in which all the developers can stay on the same page and organize.
* All the codes and their documentation are in one place in the same repository. Hence it makes easy code hosting.
* There are some special tools that github uses to identify the vulnerabilities in the code which other softwares do not have. Hence there is safety among the developers from code start till launch.
* Github is available for mobile and desktops. The UI is so user-friendly that it becomes straightforward to get comfortable with and use it.

# **Git vs SVN**

Apache Subversion or **SVN is one of the most popular centralized version control systems**. Now, SVN's popularity is on the decrease, but there are still millions of projects stored in it. It can continue to be actively maintained by an open-source community. In SVN, you can check out a single version of the repository. It stores data in a central server. The drawback of the SVN is, it has the entire history on a local repository which limits you. You can only do commits, diffs, logs, branches, merges, file annotations, etc.



While, **Git is a popular distributed version control system**, which means that you can clone your repository. Thus you can get a complete copy of your entire history of that project. This means you can access all your commits.

**Git has more advantages than SVN**. It is much better for those developers who are not always connected to the master repository. Also, it is much faster than SVN.

To better understand the differences between Git and Subversion. Let's have a look at following significance points.

|  |  |
| --- | --- |
| **Git** | **SVN** |
| It's a distributed version control system. | It's a Centralized version control system |
| Git is an SCM (source code management). | SVN is revision control. |
| Git has a cloned repository. | SVN does not have a cloned repository. |
| The Git branches are familiar to work. The Git system helps in merging the files quickly and also assist in finding the unmerged ones. | The SVN branches are a folder which exists in the repository. Some special commands are required For merging the branches. |
| Git does not have a Global revision number. | SVN has a Global revision number. |
| Git has cryptographically hashed contents that protect the contents from repository corruption taking place due to network issues or disk failures. | SVN does not have any cryptographically hashed contents. |
| Git stored content as metadata. | SVN stores content as files. |
| Git has more content protection than SVN. | SVN's content is less secure than Git. |
| Linus Torvalds developed git for Linux kernel. | CollabNet, Inc developed SVN. |
| Git is distributed under GNU (General public license). | SVN is distributed under the open-source license. |

# **Git vs Mercurial**

Mercurial and Git both are two quite similar and most popular distributed version control systems. Their strengths and weaknesses make them ideal for different use cases. Both tools use a directed acyclic graph to store history.

**Mercurial is a distributed source control management tool.** It is free and open-source. It can handle projects of any size and offers an easy and intuitive interface.

Today, Git has more than 31 million users and is owned by Microsoft. Since the last decade, the Git has become the standard for most development projects.

Mercurial still has a handful tool of large development organizations. Some software development giants like Facebook, Mozilla, and World Wide Web Consortium are using it. But it only has approx 2 % of the VCS market share. Comparatively, Git has covered more than 80% market share.

Both version control systems, i.e., Mercurial and Git are distributed version control systems (DVCS).

To better understand the similarities and differences between Git and Mercurial, let's have a look at the following points.

|  |  |
| --- | --- |
| **Git** | **Mercurial** |
| Git is a little bit of complex than Mercurial. | Mercurial is simpler than Git. |
| No VCS are entirely secured, but Git offers many functions to enhance safety. | Mercurial may be safer for fresher. It has more security features. |
| Git has a powerful and effective branching model. Branching in Git is better than Branching in Mercurial. | Branching in Mercurial doesn't refer the same meaning as in Git. |
| Git supports the staging area, which is known as the index file. | There is no index or staging area before the commit in Mercurial. |
| The most significant benefit with Git is that it has become an industry-standard, which means more developers are familiar with it. | Mercurial's significant benefit is that it's easy to learn and use, which is useful for less-technical content contributors. |
| Git needs periodic maintenance for repositories. | It does not require any maintenance. |
| It holds Linux heritage. | It is python based. |
| Git is slightly slower than Mercurial. | It is faster than Git. |
| Git supports the unlimited number of parents. | Mercurial allows only two parents. |

# **Git Version Control System**

A version control system is a software that tracks changes to a file or set of files over time so that you can recall specific versions later. It also allows you to work together with other programmers.

The version control system is a collection of software tools that help a team to manage changes in a source code. It uses a special kind of database to keep track of every modification to the code.

Developers can compare earlier versions of the code with an older version to fix the mistakes.

## Benefits of the Version Control System

The Version Control System is very helpful and beneficial in software development; developing software without using version control is unsafe. It provides backups for uncertainty. Version control systems offer a speedy interface to developers. It also allows software teams to preserve efficiency and agility according to the team scales to include more developers.

## Benefits of the Version Control System

The Version Control System is very helpful and beneficial in software development; developing software without using version control is unsafe. It provides backups for uncertainty. Version control systems offer a speedy interface to developers. It also allows software teams to preserve efficiency and agility according to the team scales to include more developers.

Some key benefits of having a version control system are as follows.

* Complete change history of the file
* Simultaneously working
* Branching and merging
* Traceability