GitHub

GitHub is an immense platform for code hosting. It supports version controlling and collaboration.

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

In this tutorial, we will learn GitHub essentials like a repository, branches, commits, pull requests, and more. Further, we will learn how to use GitHub and will create our first project on it.

What is GitHub?

GitHub is an immense platform for code hosting. It supports version controlling and collaboration and allows developers to work together on projects. It offers both distributed version control and source code management (SCM) functionality of Git. It also facilitates collaboration features such as bug tracking, feature requests, task management for every project.

Essential components of the GitHub are:

- Repositories
- Branches
- Commits
- Pull Requests

- Git (the version control tool GitHub is built on)

Advantages 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 the 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.

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.

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

GitHub vs. Git

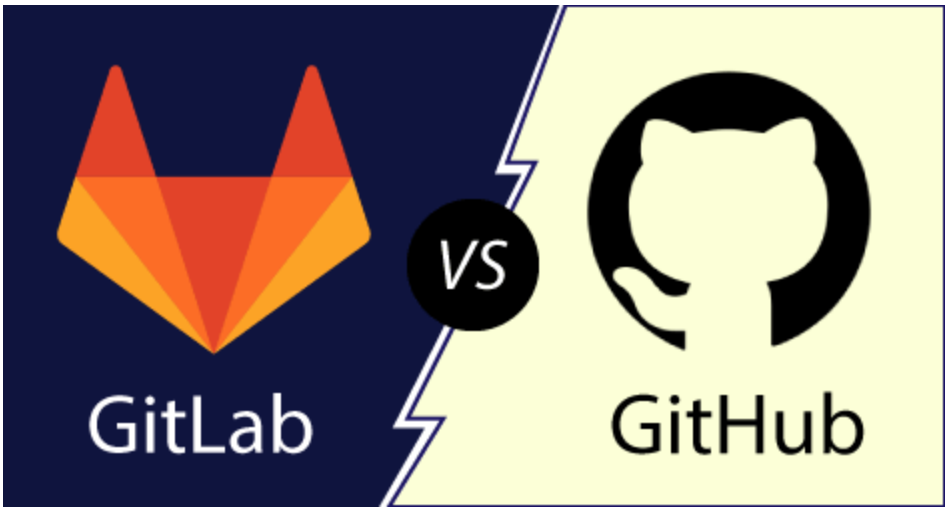
Git is an open-source distributed version control system that is available for everyone at zero cost. It is designed to handle minor to major projects with speed and efficiency. It is developed to co-ordinate the work among programmers. The version control allows you to track and work together with your team members at the same workspace.

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

GitHub	Git
It is a cloud-based tool developed around the Git tool.	It is a distributed version control tool that is used t manage the programmer's source code history.
It is an online service that is used to store code and push from the computer running Git.	Git tool is installed on our local machine for versio controlling and interacting with online Git service.
It is dedicated to centralize source code hosting.	It is dedicated to version control and code sharing.
It is managed through the web.	It is a command-line utility tool.
It provides a desktop interface called GitHub desktop GUI.	The desktop interface of Git is called Git GUI.
It has a built-in user management feature.	It does not provide any user management feature
It has a market place for tool configuration.	It has a minimal tool configuration feature.

GitLab vs. GitHub

GitLab is also a DevOps tool like GitHub. It is a **Git repository hosting service**. It provides **issue tracking, wikis,** and **CI/CD Pipeline** (Combined practices of continuous integration and delivery). It is **open-source and free** and distributed under MIT license. It is very similar to GitHub in case of functionality. It can be considered as a better choice for teamwork.

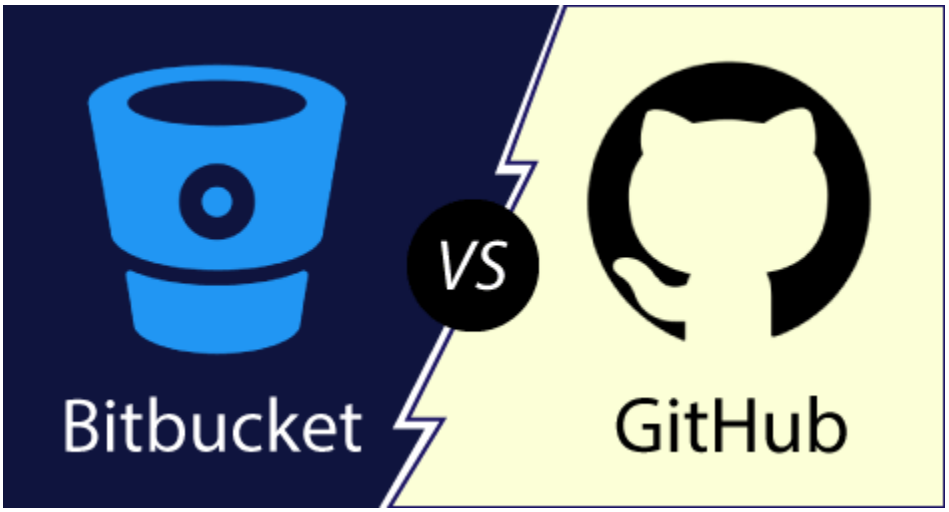


Let's see some key differences between GitLab and GitHub.

GitHub	GitLab
It was launched in 2008. It is a Git repository hosting service.	It was launched as a project in 2011 as an alternativ to the available Git repository hosting service.
It is free for public repositories and paid for private repositories.	It is free for both private and public repositories.
It has Gists (a way to share code snippets)	It does not have Gists.
We cannot attach any file to any issue.	We can attach any file to any issue.
It has a fast interface.	Comparatively, it has a slow interface.
In GitHub, we can decide the read or write access of a user to a repository.	In GitLab, we can set and modify user permissior according to their roles.
It is the largest Repository hosting service. It contains approximate 100+ million repositories.	It has lesser projects than GitHub.

Bitbucket Vs. GitHub

Bitbucket is also a **web-based version control system** owned by **Atlassian**. It contains the source code and allows us to share it among developers. It offers both free accounts and commercial accounts (paid). It provides an unlimited number of repositories for free accounts. But it has some limitations for the free accounts like a private repository can have a maximum of five users.



It is much similar to GitHub. However, no one can be considered the best. Every service has a different feel to them, and it targets different demographics no matter where you're going to get excellent service and get your work done.

Let's see the similarities and differences between Bitbucket and GitHub.

GitHub	Bitbucket
It has a user-friendly and fast interface.	It has a slick and clean interface, which provides professional view.
It is limited to the Git repository only.	It is not limited to just Git, it also supports other versio control systems like **Mercurial**, but it does n support **svn**.

It facilitates with graphs: pulse, contributors, commits, code frequency, members of it.	It assists with REST APIs to build third-party applicatio which can be used in any development language.
It is free for public repositories and paid for private repositories.	It is free for both private and public repositories. But w can have a maximum of five members for a privat repository.
GitHub comes with a lot of features and allows you to create your workflows.	Bitbucket provides a more built-in option for flexibility.
We cannot make a private repository on free accounts.	We can create an unlimited private repository for up t five users.

How to Use GitHub?

This question is prevalent for the developers who have never used GitHub. This tutorial will assist you in overcoming this question. There is nothing to worry about, the necessary steps for the using GitHub are as follows:

- Create a GitHub account
- GitHub login
- GitHub Repository
- Create a repository
- Create a file
- Create Branches

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.

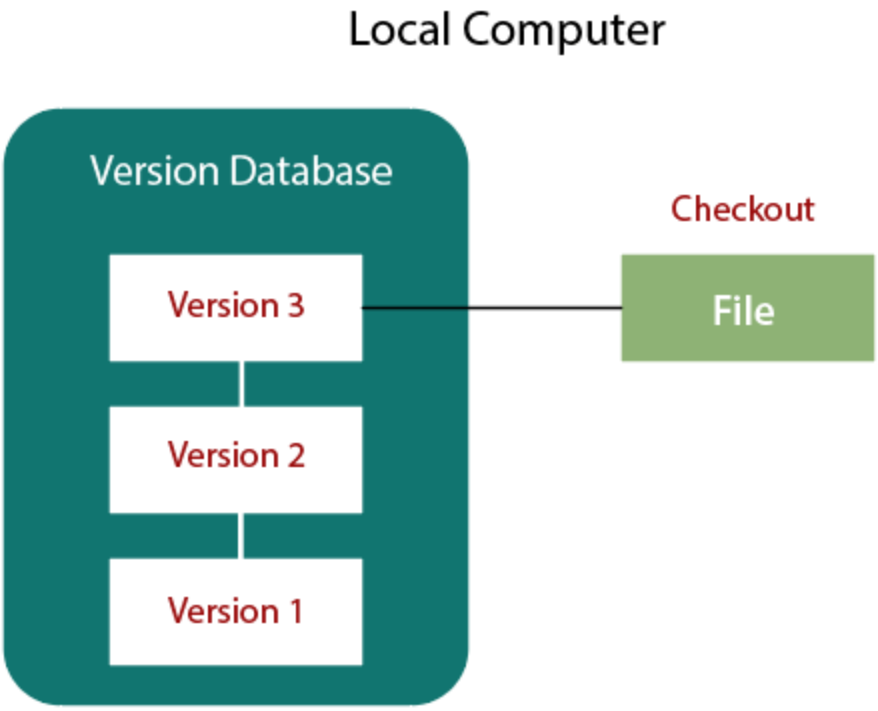
Some key benefits of having a version control system are as follows.

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

Types of Version Control System

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

Localized Version Control Systems



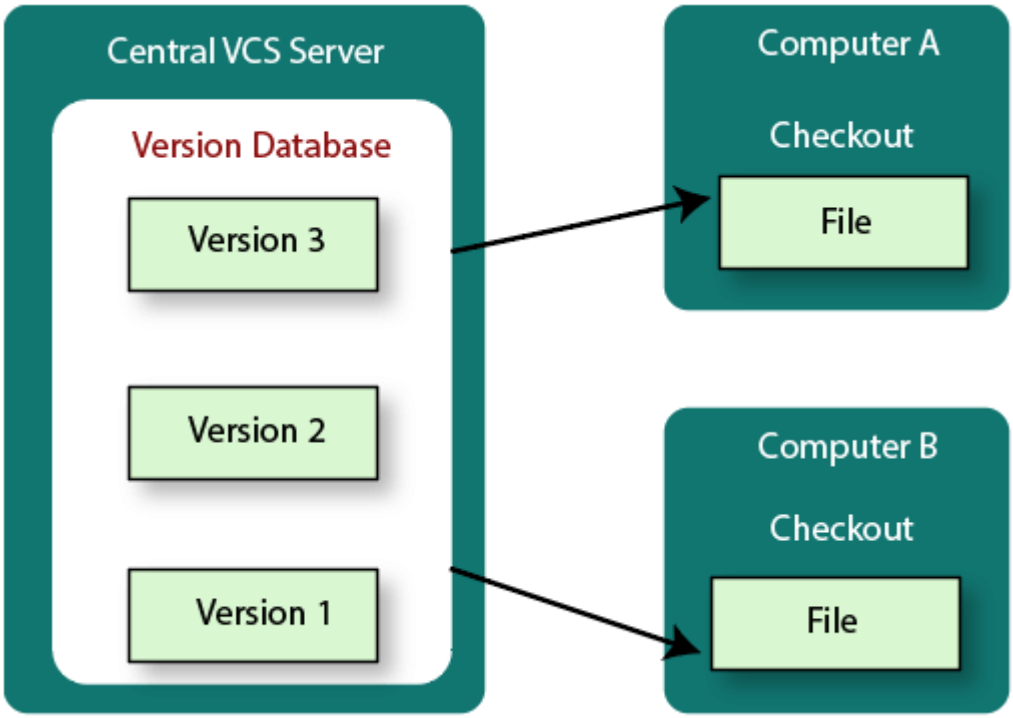
The localized version control method is a common approach because of its simplicity. But this approach leads to a higher chance of error. In this approach, you may forget which directory you're in and accidentally write to the wrong file or copy over files you don't want to.

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.

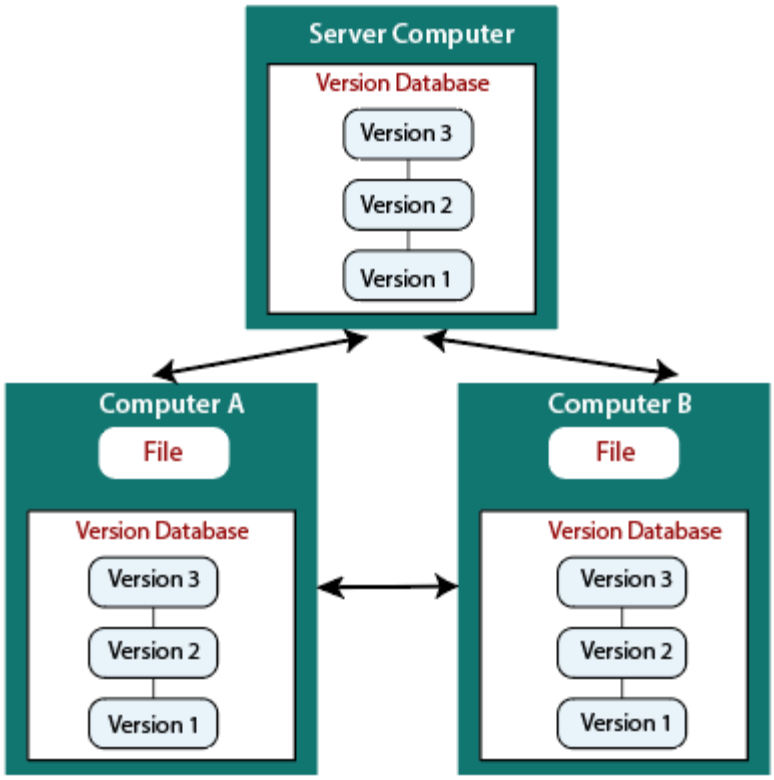
It also has the same drawback as in local version control system that it also has a single point of failure.

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 reposito in place of the central repository on the server-sid
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 th 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 the code, and they can fully mirror the centr 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 hav to interact with the central server for eve 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 we collaborating via it, that server can restore any of th client repositories