International Journal of Scientific Research in Modern Science and Technology
ISSN: 2583 -7605 (Online)
© IJSRMST | Vol. 3 | Issue 6 | June 2024
Available online at: https://ijsrmst.com/
DOI: https://doi.org/10.59828/ijsrmst.v3i6.216

An Overview of Git

Gayatri Makrand Ghodke[1], Trupti Chavan[2]

[1]Student, Department of Computer Science, Sarhad College of Arts, Commerce and Science, Pune, India
[2]Assistant Professor, Department of Computer Science, Sarhad College of Arts, Commerce and Science, Pune, India
[1]**Email:** gayatrighodke03@gmail.com | [2]**Email:** truptichavan17.tc@gmail.com

ABSTRACT

Git is a distributed version control system used in software development that tracks modifications made to source code. Git, as opposed to centralized version control systems, enables several developers to collaborate independently on a single project by letting them maintain separate local repositories that are synchronized with a remote repository. Repositories, which hold the project's files and change history, are essential to Git's operation. A commit in a repository is a snapshot of the project at that specific point in time; each commit is distinguished by a distinct hash and is accompanied by an explanation message. Branches allow developers to work independently on various features or fixes. While other branches are used for development, the stable code is usually found in the main branch, also known as main (formerly master). Rebasing and merging are two ways that changes from branches can be merged into the main branch, each with a distinct function for preserving project history. A local copy of a repository is created by cloning it, and local changes are synchronized with a remote repository through pushing and pulling. The staging area serves as a buffer where modifications are made prior to commit. To manage and review these changes, use commands like `git add`, `git commit`, `git status`, and `git log`. Submodules, cherry-pick, and Git hooks are examples of advanced features that provide more customization and control. Well-known Git hosting services, such as GitHub, GitLab, and Bitbucket, offer platforms for code review, repository management, and tool integration, which promote collaborative development. Gaining proficiency with Git greatly improves development processes by facilitating effective teamwork and code management.

Keywords: Git, GitHub, GitLab, Development, Projects, Repository, Version, System

1. Introduction

Git is currently the most widely used version control system due to its open-source nature, which makes it simple to use for a variety of tasks. Git makes it possible for multiple people to collaborate while using the same files. Additionally, it aids in the team's ability to handle the confusion that frequently arises when several people edit the same file. Git copies changes from one such repository to another and gives each developer a local copy of the complete development history. Git is a distributed version control system that can handle any size project quickly and effectively.[1] It is free and open-source. Git is based on the idea of distributed software development, to which multiple developers may have access. GitHub is essentially a Microsoft-owned for-profit business that hosts Git repositories online. It facilitates online, remote, and user-to-user sharing of git repositories. On GitHub, hosting a public repository is also free of charge. Users share their repository on the internet for a variety of purposes, such as community service, open source contribution, project sharing, and project deployment, among many others. Git is a robust version control system that is frequently used to monitor source code changes made during the software development process. Since its creation by Linus Torvalds in 2005, Git has grown to be a vital resource for developers everywhere. Gaining an understanding of Git will greatly improve your teamwork and coding productivity. With GitHub, you may access and download projects from any computer that hosts Git repositories. Git is currently the most widely used version control system due to its open-source nature, which makes it simple to use for a variety of tasks.

Git makes it possible for multiple individuals to collaborate while using the same files. Additionally, it aids in the team's ability to handle the misunderstanding that frequently arises when several people modify the same file. Git copies changes from one such repository to another and gives each developer a local copy of the complete development history.

Git is a version control system that is used by the web-based platform GitHub to assist developers in tracking and managing code changes. It enables numerous users to work together on a project from any location in the world, keep track of changes, and contribute to the code. GitHub caters to both individuals and large enterprises with its free and paid plans. Git is a potent distributed version control system that allows for smooth collaboration and extensive history tracking by monitoring and managing code changes made by several contributors. Among its fundamental functions are branching and merging, which enable concurrent development on various features or fixes, as well as a staging area for grouping changes prior to committal. Advanced features like rebasing, cherry-picking, and hooks for unique workflows are also supported by Git. These features are expanded by GitHub, a well-known platform based on Git that offers an intuitive interface for code review, repository hosting, and collaboration. It has features like issues and project boards for tracking tasks and bugs, pull requests for proposing and discussing changes, and integration with continuous integration/continuous deployment (CI/CD) tools for automating testing and deployment. GitLab is a DevOps platform that combines multiple software development tools with Git-based version control. It provides strong capabilities for project management tools, security features, container registry, continuous integration and deployment pipelines, source code management, and analytics. It supports a variety of deployment requirements and improves development workflows and teamwork. It is offered as a self-hosted and cloud-based service.[2]

2. Git Repository Organization

There are Four phases that enable a seamless workflow that ensures effective version control and collaboration throughout the development process as changes are made, reviewed, and integrated into a version-controlled system.

1. **Working directory:** This is the local directory in which you create the project and edit the code. Your active file editing directory is the working directory. This is the area you work directly with, where you make all of your file edits and new creations. Version control does not yet apply to changes made in this directory.

2. **Staging Area**: The staging area, also called the index, is where you should put your project before committing to it. Other team members use this for code reviews. The index, which is another name for the staging area, serves as a mediator between the local repository and the working directory. Here, you add new or modified files to get your changes ready for commit. You can examine and stage changes as needed before they become a part of the project's history in the staging area.

3. **Local Repository:** Before submitting changes to the project's central GitHub repository, you should first commit them to this repository. What the distributed version control system offers is this. This is in line with our directory's git folder. Git keeps track of the commits in the local repository. The changes you've staged and committed are documented in the history of the local repository. This repository's commits are each uniquely identified by a hash that contains information about the commit, such as the author, timestamp, and message. The whole history of commits for the branch you are working on is stored in the local repository.[3]

4. **Central Repository:** In a Git workflow, the "central repository" is commonly referred to as the "remote repository". Each team member has a local repository copy of this project, which is the primary project on the central server. A shared repository that acts as a focal point for cooperation is the remote repository. Usually, it is hosted on websites like Bitbucket, GitHub, and GitLab. Several contributors can synchronize their changes with one another via the remote repository. In order to update their local repository with contributions from other team members, developers push their local commits to the remote repository and pull changes from it.

3. Features of Git

1. Distributed System: Users can work on a project using distributed systems from anywhere in the world. Several distant collaborators can access a central repository within a distributed system by utilizing a version control system. These days, one of the most widely used version control systems is called Git. In the event that the central server experiences a system failure, having a central server causes a problem with data loss or disconnectivity. Git mirrors the entire repository on each snapshot of the version that the user is pulling in order to address this kind of scenario. In this scenario, users who have downloaded the most recent project snapshot may be able to retrieve the copy of the repositories in the event that the central server crashes.[4]

2. Safe: Git maintains a history of every commit made by every team member to the developer's local copy. Every time a push operation is carried out, a log file is kept track of and pushed to the central repository. Therefore, the developer can easily track and handle any issues that may arise. Git stores all of the records as objects in the hash using SHA1. These hash keys allow each object to work together with the others. A 14-digit Hex code is created from the commit object using the cryptographic algorithm SHA1. Maintaining a record of every commit made by every developer is beneficial.

3. Open-Source: Git is a distributed version control system that is free and open-source that can quickly and effectively manage any size project, from tiny to enormous. Because it allows users to alter the source code to suit their needs, it is known as open-source. In contrast to other version control systems, which charge for features like repository space, code privacy, accuracy, speed, etc., Git is entirely free software that offers these features—and does so more effectively than the competition.

Git is open-source software that makes it simple and effective for multiple people to collaborate on the same project at the same time. Git is therefore regarded as the greatest version control system on the market at the moment.

4. Lightweight: While cloning is happening, Git stores all of the data from the central repository on the local repository. Hundreds of people may be working on the same project, which could result in an enormous amount of data in the central repository. While cloning so much data into local machines could raise concerns about potential system failure, Git has already addressed this issue. Git uses lossless compression, which reduces the amount of space needed to store compressed data in the local repository. It uses the opposite method whenever this data is required, which conserves a significant amount of memory.

5. Economical: Git is freely available since it is published under the General Public's License (GPL). Git makes a copy of the central repository on the developer's local computer; as a result, all operations are carried out there before being pushed to the central repository. Pushing is only carried out once the local machine's version is flawless and prepared for pushing to the central server. Nothing experimenting is done with the central server's files. This saves a significant amount of money on pricey servers. Since all of the heavy lifting is done on the client side, large machines are not required on the server side.[5]

4. Git Hosting Services

Platforms that offer a central location for managing and storing Git repositories are known as Git hosting services. These services, which are based on the distributed version control system Git, provide a range of tools and features that make version control, collaboration, and project management easier.

1. GitHub

Utilizing Git, a distributed version control system, GitHub is a popular platform for version control and collaboration. GitHub, a 2008 startup that is currently a part of Microsoft, provides cloud-based repository hosting for both small and large development teams. It offers strong code management and sharing features, such as pull requests, code reviews, and public and private repositories, which promote teamwork and preserve code quality. GitHub's usefulness in project management, continuous integration and deployment (CI/CD), and other areas is further enhanced by its smooth integration with a wide range of third-party programs and services. Important features include an extensive issue tracking system, GitHub Pages for hosting static sites, and GitHub Actions for automating workflows. Because of its social coding feature, users can follow other developers, participate in discussions, and contribute to open-source projects, all of which help to create a thriving developer community. GitHub is a flexible tool for contemporary software development, and its enterprise offerings offer extra features and support for businesses needing more control and scalability. Git is a version control system that is used by the web-based platform GitHub to assist developers in tracking and managing code changes. It enables numerous users to work together on a project from any location in the world, keep track of changes, and contribute to the code. GitHub caters to both individuals and large organizations with its free and paid plans. With a feature-rich feature set that caters to both small and large teams, GitHub is a well-known platform for version control and collaboration. Fundamentally, GitHub offers stable repository hosting with options for both public and private repositories, making it an ideal tool for managing and storing Git-based projects. With features like pull requests and code reviews, it makes collaboration easier and enables teams to suggest, debate, and easily incorporate changes. GitHub Actions provides robust automation for continuous integration and deployment (CI/CD), making it simple for developers to write, test, and publish their code. Furthermore, GitHub has integrations with a wide range of external tools and services, which expands its functionality to cover different facets of the development lifecycle.[6]

2. GitLab

A full suite of tools for managing the software development lifecycle is integrated into the DevOps platform GitLab. GitLab was founded in 2011 and provides self-hosted and cloud-hosted versions to meet a wide range of development requirements, from small projects to complex enterprise solutions. With its powerful version control features that support both public and private repositories, advanced branching, merging, and tagging, the platform excels at managing Git repositories. Build, test, and deployment procedures are automated by GitLab's integrated CI/CD pipelines, allowing for continuous delivery with configurable pipeline configurations. Built-in tools for milestone management, issue tracking, and kanban boards streamline project management and enable effective planning and execution. With features like merge requests for code reviews, the platform improves teamwork by enabling members to jointly suggest, examine, and approve changes. With integrated scanning tools to find vulnerabilities early in the development cycle and compliance management features to adhere to regulatory standards, security is a top priority. GitLab offers tools for managing and deploying containerized applications in addition to supporting Kubernetes and containers. GitLab is highly configurable and extensible, supporting a variety of third-party integrations and custom modifications. It also includes monitoring and analytics tools for tracking application performance and CI/CD pipeline metrics. Granular user management and access control let administrators set up roles and permissions at different levels. GitLab provides a wealth of documentation and assistance, and its enterprise users can take advantage of its expert support services. Using its self-hosted configuration or its cloud-hosted version, GitLab offers a unified solution to enhance development workflows and collaboration among various teams and projects. GitLab is an effective tool for contemporary DevOps processes because it provides a full range of features intended to address the whole software development lifecycle. GitLab's strong repository management, which allows for both public and private repositories to have advanced version control features like branching, merging, and tagging, is at the heart of the software. The platform's integrated Continuous Integration/Continuous Deployment (CI/CD) pipelines streamline development workflows and guarantee effective software update delivery by automating the processes of building, testing, and deploying code. GitLab is also a master at project management; it offers capabilities like kanban boards, milestone management, and issue tracking to assist teams in efficiently organizing, monitoring, and overseeing their work.[7]

3. **Bitbucket**

Atlassian created Bitbucket, a Git repository management tool renowned for its smooth interaction with other Atlassian products like Jira and Trello. It offers reliable repository hosting for Git repositories, both public and private, and makes code management easier with features like tagging, branching, and merging. Built-in CI/CD service Bitbucket Pipelines, which automates the building, testing, and deploying of code, is a crucial part of Bitbucket. A `bitbucket-pipelines.yml` file is used to configure this automation, which streamlines workflows for continuous integration and delivery. With its support for pull requests, Bitbucket further improves teamwork by enabling members to suggest, examine, and debate code changes prior to integration. Integrating Jira provides thorough project management and issue tracking by associating pull requests and code commits with particular issues to improve project progress visibility. Furthermore, Bitbucket offers access control and fine-grained branch permissions, guaranteeing safe and controlled access to code. To further encourage teamwork and knowledge exchange, each repository may have a wiki for upholding project-related notes and documentation. Bitbucket is a useful tool for managing software development projects and encouraging productive teamwork overall thanks to its features and integrations.[8]

5. Conclusion

Git is a distributed version control system that monitors source code changes and is used in software development. It maintains independent local repositories synchronized with a remote repository, enabling numerous developers to work together on a single project. The working directory, staging area, local repository, and central repository are the four stages of the Git workflow. The staging area is where changes are added and staged before being merged into the project history, whereas the working directory is the local directory where changes are made and edited. The main project on the central server is the local repository, sometimes referred to as the remote repository.

Git is an open-source, safe version control system that lets users handle any kind of project. It keeps track of each commit that each team member has done, making issue tracking and resolution simple. Git uses SHA1 hashing to save all records as objects, and it converts the commit object into a 14-digit Hex code. Because all data from the central repository is stored locally, it is lightweight and requires less space than compressed data storage.

Git hosting services, like GitHub, GitLab, and Bitbucket, are hubs for organizing and storing Git repositories. These systems provide pull requests, code reviews, cloud-based repositories, robust code management tools, and different development requirements. Also, Bitbucket facilitates pull requests, which enhances collaboration.

6. REFERENCES

[1] Diomidis Spinellis (2012). Git. IEEE software 29 (3), 100-101.

[2] Jon Loeliger, Matthew McCullough (2012).Version Control with Git: Powerful tools and techniques for collaborative software development." O'Reilly Media, Inc."

[3] Jiaxin Zhu, Minghui Zhou, Audris Mockus (2014). Patterns of folder use and project popularity: A case study of GitHub repositories.

Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, 1-4.

[4] Johan Abildskov, Johan Abildskov (2020). Additional git features. Practical Git: Confident Git Through Practice, 139-161.

[5] Mike McQuaid (2014). Git in practice. Simon and Schuster.

[6] Valerio Cosentino, Javier Luis, Jordi Cabot (2016). Findings from GitHub: methods, datasets and limitations.

Proceedings of the 13th International Conference on Mining Software Repositories, 137-141.

[7] Prithwiraj Choudhury, Kevin Crowston, Linus Dahlander, Marco S Minervini, Sumita Raghuram (2020). GitLab: work where you want, when you want. Journal of Organization Design 9, 1-17.

[8] Sudip Chakraborty, PS Aithal (2022). A practical approach to GIT using bitbucket, GitHub and SourceTree. International Journal of Applied Engineering and Management Letters (IJAEML) 6 (2), 254-263.

Cite this Article:

Gayatri Makrand Ghodke, Trupti Chavan, **"An Overview of Git"** *International Journal of Scientific Research in Modern Science and Technology (IJSRMST), ISSN: 2583-7605 (Online), Volume 3, Issue 6, pp. 17-23 June 2024.*

Journal URL: https://ijsrmst.com/

DOI: https://doi.org/10.59828/ijsrmst.v3i6.216.