Click Here

FacebookLinkedInTwitterYouTubeFacebookLinkedInTwitterYouTube Git isn't hard to learn, and when you combine Git with GitLab, you've made it a whole lot easier to share code and manage a common Git commit history with the rest of your team. This tutorial shows you how it's done. How to learn Git and GitLab While it takes time to fully master a DevOps tool such as Git, any developer who follows this 10-step Git and GitLab tutorial will become competent with these popular DevOps and CI/CD tools very quickly. The 10 steps to follow to learn Git and GitLab fast are as follows: Install Git and create a GitLab account. Create a Git repository in GitLab. Add files and commits to your GitLab repo. Create new Git branches. Toggle between branches in GitLab. Create a GitLab merge request. Approve a GitLab merge request. Clone the remote GitLab repo locally. Use the most important Git commands locally. Push your local changes back to GitLab. Step 1: Install Git and create a GitLab account To get started with Git and GitLab, you must first install Git locally and register for a GitLab account. The Git installation on Windows is straightforward. Simply download and run the installer, accept all the defaults and Git will be installed, along with the Git Bash and Git GUI tools. To register for a GitLab account, all you need to do is create a unique username, set your credentials and validate your email address. Once that's complete, you can move onto Step 2 of this Git and GitLab tutorial. Step 2: Create a repository in GitLab When you log in to GitLab for the first time, you are asked to create a project. Choose this option to create a new, blank Git repository on the GitLab server. Teams of developers use a Git repository to perform the following tasks: Track their files and their commit histories. Create branches and issue merge requests. Share code and collaborate with others. When the form to create a project appears, simply provide a name for the project and set the GitLab repo's visibility to public. Accept all the other defaults and submit the form to create your first GitLab repository. One of the first steps in any Git and GitLab tutorial has to be the creation of a GitLab repository. Step 3: Add files and create commits online The landing page for your new repository has a button with a plus sign on it. Click this button to create a new file. When you save the file, you will be forced to provide a commit message. With each save, you add to your Git commit history. Add three new files with the following names and Git commit messages: alpha.txt First Git commit bravo.txt Second Git commit charlie.txt Third Git commit Once added, the three files and their commit messages display on your GitLab repository's homepage. You can click on them to view their content or perform further edits. GitLab allows you to add files and create Git commits online. Step 4: Create Git branches and add files Note that you are currently on a branch called main. Using the same plus button as you did in Step 3, create a new branch named dev. Add another file named devo.txt and use 'Fourth Git commit' for the message. This new file will be added to the dev branch. It will not be visible on the main branch. Step 5: Navigate between Git branches Use the drop-down box to toggle between the dev and the main branch. Notice how the devo.txt file is on the dev branch, but it is not on the main branch. Git branches allow developers to do isolated development separately and independently of the work of other developers. This encourages creativity, experimentation and chance-taking. If an experiment doesn't work, the branch can be deleted. If the creative endeavor is a success, the new branch can be merged into the main branch for others to see. The devo.txt file is only in the dev branch. A git merge must be performed for it to be visible on the main branch on GitLab. Step 6: Create a GitLab merge request To bring the dev and main branches in sync, click the blue Create merge request button at the top of the screen. Fill out the title and the description and submit the merge request form. Merges can be performed locally or through GitLab merge requests online. Step 7: Accept a merge request in GitLab When you open the GitLab navigation bar on the left, you will see a new Merge request listed. Click the Merge requests link and then Approve to approve it. If you like Devo, give it a thumbs-up as well. Leave the "'Delete source branch" option selected and click Merge. When you return to the homepage of your repository, notice that the devo.txt file is in the main branch, and the dev branch is deleted. Learning how to approve a merge request is an important step in any Git and GitLab tutorial. Step 8: Clone the GitLab repository A clone of a GitLab repository brings all the files and branches on the server down to the developer's local machine. Most Git operations happen locally, in tandem with local software development. Find the GitLab URL Click the blue Code button on the repository's GitLab page and copy the HTTPS GitLab URL. Then open Git Bash in the folder where you want to store the folders and files for this repository and issue the git clone command: git clone After the clone command completes, you can find all the files in your Git repository under a subdirectory with the same name as your GitLab project. When you clone a GitLab repo, Git makes all the files in that repository available on your local machine. Step 9: Perform standard Git operations With the GitLab repository cloned locally, you can add files, edit files, delete files and play around with various Git commands including the following: Fetch, merge and pull are three of the important porcelain Git commands with which beginners should become comfortable. Step 10: Push your changes back to GitLab When you are done with local development, you can push your updated set of local commits back to the server with a push command: git push origin The first time you push using HTTPS, make sure you are logged into GitLab on your default web browser. You must be authenticated to push changes into a GitLab repository. GitLab authentication troubleshooting On a Windows machine, GitLab will try to validate your credentials using OAuth to log you in to the server. If the git push attempt results in a 403 error, you might need to obtain a GitLab personal access token and authenticate with that. If authentication continues to be an issue, you could alternatively create SSH keys for GitLab and authenticate over secure shell. GitLab is a full-featured and powerful CI/CD platform built around the popular Git tool. Learning how to use Git with GitLab is the first step to take advantage of GitLab's many features. Cameron McKenzie has been a Java EE software engineer for 20 years. His current specialties include Agile development, DevOps and container-based technologies such as Docker, Swarm and Kubernetes. View All Videos If you are looking for a better way to organize your project files, you should consider using GitLab. The beauty of GitLab is that it allows all team members to collaborate in all phases of a project, simplifying software development for all involved. In this GitLab tutorial for beginners, you will learn how to use GitLabs main features to manage your files and increase productivity (and even learn how Gitlab time tracking works with a help of Gitlab time tracking integration!) GitLab Definition GitLab is a web-based Git repository that allows software development teams to plan, code, test, deploy, and monitor product changes in one place. Git is a versioning system that tracks changes and enables pushing/pulling updates using remote resources. GitLab offers free, open, and private repositories, along with issue tracking and wikis. It helps teams collaborate throughout the software development lifecycle, enhancing product value and user satisfaction. Essential GitLab Basics Well start with the GitLab tutorial basicsits main features, pricing, installation, and how to use it. Planning GitLab offers powerful planning tools to keep your team in sync. The platform enables planning and management through epics, milestones, and groups, allowing teams to organize and track GitLab project progress. Coding GitLab tutorial wouldnt be complete without mentioning that GitLab allows developers to code directly in the platform. It helps to create a project using CI/CD pipelines. Now, lets see How to create a project. How to Create a Project?To create a project in GitLab, do the following steps:On the above GitLab welcome, Click on Create a project.We get a four options page to create a project in GitLab, as shown in the figure.Firstly, lets create a Blank project.Enter the Project name and Enable Private Visibility level, as shown in the figure, and then click Create project.An empty repository project has been created, as shown in the figure.Now open Git Bash and run the following commands, to help to create a project.Configure your identity using git config --global user.name "Your Name" git config --global user.email "email@example.com"Make a directory to store your GitLab projects mkdir GitLabTutorial cd GitLabTutorial/Configure your identity using git config --global user.name "Your Name" git config --global user.email "email@example.com"Make a directory to store your GitLab projects mkdir GitLabTutorial cd GitLabTutorial/Now initialize a folder by using the command git init.Now lets create a file by using the command below$ touch Mindmajix.txt?Run a Notepad file by using the command below$ notepad Mindmajix.txt?You will get a Notepad file naming Mindmajix. Write a few words in the file, as shown in the figure.Check the status$ git statusAdd the file$ git add .?Commit the file$ git add .?Check the status$ git add .?Now go to GitLab, click Projects dropdown in the menu, and open Your Projects, as shown in the figure.Add remote origin$ git remote add originKeerthana_Jonnalagadda/gitlab-tutorial.?Fetch the file$ git remote -v?Push the file$ git push -u origin master 7.Open GitLab, check Your projects, and you can see the Mindmajix.txt file which is created to push into the GitLab as shown in the figure Click on the Mindmajix.txt file. 8.You can see the data in the file, as shown in the figure. Likewise, we can push our files, folder, and records into GitLab.Create from TemplateWe can also create a project using 24 in-built templates of different sources like Ruby on Rails, Spring, iOS Swift, .NET Core, and many more as shown in the figure.From the list available in-built templates, click Preview to view the template source and click Use template to start creating the template. Rest the process is the same as creating a blank project.[]Create from Importing a ProjectWe can import projects from various Git and other sources to GitLab, as shown in the figure.Click Create project to GitLab, we can connect to an external repository using a CI/CD external repo.Choose GitHub or Repo by URL to connect with an external repositoryHow to Create a Group in GitLab?The group is a collection of projects that helps to manage and organize the projects like a folder. It helps to maintain the group members access and permissions for each project in the group. To create a group, do follow the steps.Open GitLab, click Groups on the top menu bar, and then click on New group to create a group as shown in the figure.Now you get a page to create a group, as shown in the figure.Enter the group name, click Create group, and you can see the group is created as shown in the figure.Now click on the created group and click Edit group settings to edit the group as shown in the figure.Now you get a page to edit the group, as shown in the figure.Open project, click on Settings and then click General as shown in the figureA general settings page opens. Choose project avatar and select your avatar was click Save changes.You can see the project avatar is updated.We can also add, a avatar while creating the project.Add users to the groupAdding users to the group makes an organized work that benefits time to access files. Admin can restrict the access of the user and can even schedule a time for a deadline to work on the project. To add users to the group, do the following steps.Open your Group, click on Members in the groups dashboardMembers page opens, as shown in the figure.Enter GitLab member or Email address to invite into the group.We can choose the role of a particular member such as Guest, Reporter, Developer, Maintainer, and Owner.We can schedule the access expiration data that helps to organize the work in a schedule.And then Click Invite to add the member into the group.This way, we can invite members or users to the group.Changing the owner of the groupThe ownership of group members must have at least one owner. Changing the owner of a group is only possible by one owner. To change the owner of the group, do the following steps.As a current ownerGo to groups, click Members in the group dashboard.Assign different user Owner permissions.Have a new owner sign in and remove Owner permission from you(owner).As an administrator,Go to groups, click Members in the group dashboard.Assign different user Owner permissions.Now refresh the page. We can remove permissions from the original owner.Remove members from the groupIn GitLab, an owner is a person who manages the group members.To remove a member from the group, do the following steps:In the group, click Members in the group dashboard.Click the Delete button which is next to a group member to remove. A remove member window appears.Enable Also unassign this user from related issues and merge requests checkbox.Click Remove member.Project forking workflowA fork is a copy of an original repository that we make in the namespace and can apply changes later without affecting the original project. It takes a few steps to fork a project in GitLab.Create a forkTo create a fork, do the following steps:On the project page, pick a project and click on the fork icon as shown in the figureSelect the project to create a copy of the project.The project has been forked successfully.Request for merging.A request is required to merge the changes in the forked project to the original project.Click on Create merge request.Now you get a page to create a merge request, as shown in the figure.Select branches, and push into the original project and click on Compare branches and continue.A New merge request page opens. Give all the required detail to merge and click Submit merge request.A request has been sent to the person for merging the changes in the project.Removing a forkIn GitLab, we can even remove the fork relationship that helps not to send merge requests to the upstream project. Only project owners have permission to remove the fork of a project. To remove a fork, do follows the steps below:Go to projects, Click on Settings and then click General as shown in the figure.A General Settings page opens in that list, scroll down to Advanced and click Expand.An advanced settings page opens in that scroll down to Remove fork relationship and click on it.A confirmation window appears. Type the project link to proceed and click Confirm.ConclusionOn winding up, GitLab is a user-friendly and open-source tool that helps to store the file and organize them more easily. It allows developers to coordinate easily to work on the project. GitLab is easy to manage and configure the projects, and it has relied on Git, where the number of users is not restricted and can create more building applications. We have learned the basic details of GitLab, How to create a project, group, and how to fork a project. I hope this tutorial has helped you to understand GitLab. No one codes alone. Because tech teams can grow at breakneck speeds, solitary coders quickly become part of large teams of developers that need to cooperate and coordinate their efforts. Git has become the tool for source code management, and plenty of platforms have popped up to make Git accessible to dev teams. GitLab is one of them. This beginners guide covers: The GitLab Dashboard What youll find in your toolbar How to create your first project Whether youre a project manager looking to roll this out to your team, or a developer getting onto a Git platform for the first time, youre sure to get something out of this guide. Heres what your dashboard will look like the first time you open up GitLab. Nothing too crazy so far. This screen is designed for you to get an overview of your GitLab activity. Specifically, it shows you projects youve been working on, issues and merge requests assigned to you, and todos. Todos are a handy GitLab feature that let you create your own todo list. You can add todos from issues and merge requests, and mark them as done when youve completed the task. This is a huge time saver when onboarding multiple people. Think of GitLab groups a bit like a folder in Google Drive. If you dozens of documents on policies and best practices for a dev team, you wouldnt grant your team access document by document; youd stick them in a folder and grant access to the folder in one go. Explore public projects: You can access a huge library of public GitLab projects here. Maybe you want to use public projects to figure out how to organize your own, or youre looking to grab a copy of open source software and contribute to it. Learn more about GitLab: This is where you can access GitLabs documentation, geared towards helping you understand and use the platform. It covers everything from creating a project to using CI/CD pipelines. GitLabs documentation can all be edited directly in GitLab through merge requests, a bit like a wiki. So when you get to the point where you can call yourself a GitLab pro, you can give back by contributing to GitLabs documentation. Now that weve gone over what youll see when you first log in, lets go over what else you can do in GitLab. Lets start with the upper-left corner of the toolbar. Projects: Once youve created or been given access to projects, this is where youll access them. The drop-down lets you access three screens: Your Projects, Starred Projects, and Explore Projects. The Explore Projects option is how you can browse public projects. Groups: From here, you can find groups youre already a part of as well as public groups. You can request to join any group made public and contribute to their projects. More: This dropdown gives you access to dashboards that help you track a projects progression such as analytics and milestones as well as snippets, which are a bit like post-its for code. Well go over this in more detail later. Now, whats on the right side of the toolbar? From left to right: New: This drop-down menu lets you create new projects, groups or snippets. Search bar: From here, you can quickly find projects, groups, issues, and more. Issues: Issues are how work gets done in GitLab. Create an issue and assign it to yourself to keep track of what you need to get done. Assign it to another member of the team when you need something from them. You can give an issue a checklist to track progress on a more complicated task or link them to other issues to show dependencies. Merge Requests: Merge requests are created by developers when theyre ready to integrate their branch of code with the source code. Theyre virtually identical to pull requests in GitHub. It is how you can add your code changes to the master branch of a project. Todos: Todos are a handy GitLab feature that let you create your own todo list. You can add todos from issues and merge requests, and mark them as done when youve completed the task. This is a huge time saver when onboarding multiple people. Your work: This is where you can access projects youve been assigned to, have starred, or have been working on. Your todos: This is where you can access todos youve created or been assigned. Its the same screen as the todos dropdown, you can change your status, edit your profile, and personalize your settings. Now what happens when we click on that More dropdown from before? Wow. Theres a lot to go over here. The Activity section is where you can see what the people in your projects have been up to. Everything from merge events to comments can be found here. You can choose to see the activity in all the projects youre in or just starred projects. You can also filter the entries, focusing only on merge requests, for example. Setting Milestones in GitLab is a great way to track issues and merge requests. Here is where you can get a rundown of all your milestones, including the ones coming up and the ones youve already closed. GitLabs Snippets feature is essentially post-its for code. Anytime youve got a chunk of code that you want to save, you can create a snippet in GitLab to store it. Because you can make your snippets public or private, you can use them in a variety of ways. You could use snippets to store errors in code and share them with other members of the team without creating an official issue maybe to get input before you do anything with the error. Or you could just use them to keep track of what you need to work on later. The productivity analytics feature lets you keep an eye on how your dev team is doing, skipping the usual hassle of trying to get answers directly from your developers. This tool tracks merge requests and has a number of features you can use to filter by groups, projects, milestones, and more. The environments dashboard lets you get a glance at multiple projects and their environments at once. This is great for project managers and team leaders who need to supervise pipelines and spot blockers. The dashboard can display a maximum of seven projects, with three environments per project (eg. production, staging). The operations dashboard goes more in-depth on a per-project basis than the environments dashboard. This means you can see a projects number of active alerts, last commit, pipeline status, navigation bar on the left, you will see a new Merge request listed. Click the Merge requests link and then Approve to approve it. If you like Devo, give it a thumbs-up as well. Leave the "'Delete source branch" option selected and click Merge. When you return to the homepage of your repository, notice that the devo.txt file is in the main branch, and the dev branch is deleted. Learning how to approve a merge request is an important step in any Git an GitLab tutorial. Step 8: Clone the GitLab repository A clone of a GitLab repository brings all the files and branches on the server down to the developer's local machine. Most Git operations happen locally, in tandem with local software development. Find the GitLab URL Click the blue Code button on the repository's GitLab page and copy the HTTPS GitLab URL. Then open Git Bash in the folder where you want to store the folders and files for this repository and issue the git clone command: git clone After the clone command completes, you can find all the files in your Git repository under a subdirectory with the same name as your GitLab project. When you clone a GitLab repo, Git makes all the files in that repository available on your local machine. Step 9: Perform standard Git operations With the GitLab repository cloned locally, you can add files, edit files, delete files and play around with various Git commands including the following: Fetch, merge and pull are three of the important porcelain Git commands with which beginners should become comfortable. Step 10: Push your changes back to GitLab When you are done with local development, you can push your updated set of local commits back to the server with a push command: git push origin The first time you push using HTTPS, make sure you are logged into GitLab on your default web browser. You must be authenticated to push changes into a GitLab repository. GitLab authentication troubleshooting On a Windows machine, GitLab will try to validate your credentials using OAuth to log you in to the server. If the git push attempt results in a 403 error, you might need to obtain a GitLab personal access token and authenticate with that. If authentication continues to be an issue, you could alternatively create SSH keys for GitLab and authenticate over secure shell. GitLab is a full-featured and powerful CI/CD platform built around the popular Git tool. Learning how to use Git with GitLab is the first step to take advantage of GitLab's many features. Cameron McKenzie has been a Java EE software engineer for 20 years. His current specialties include Agile development, DevOps and container-based technologies such as Docker, Swarm and Kubernetes. View All Videos GitLabGitLab is an open-source, web-based DevOps lifecycle tool that provides a Git repository management system, CI/CD pipeline features, and collaboration tools, all in a single platform. Its designed to help developers and teams manage code, track issues, automate testing and deployment, and collaborate on software projects efficiently.CI / CDContinuous Integration (CI) and Continuous Delivery/Deployment (CD) are key practices in modern software development that aim to improve the efficiency and reliability of the software delivery process.CI / CD. SourceContinuous IntegrationCI is the practice of frequently integrating code changes from multiple developers into a shared repository, followed by automated testing to detect issues early.Developers (DEV) write code and commit changes frequently.The system automatically triggers an Application Test to validate code functionality.If successful, an Integration Test ensures that changes integrate well with the existing system.The goal of CI is to detect issues early and ensure the software is always in a deployable state.Continuous Integration. SourceContinuous DeliveryContinuous Delivery extends CI by ensuring that the code is always in a deployable state.After passing integration tests, the software moves to Acceptance Testing (e.g., user acceptance testing, performance testing).The release to Production is manual a human decides when to deploy.Ensures the software is always in a deployable state but requires approval for release.Continuous Delivery. SourceContinuous DeploymentContinuous Deployment takes Continuous Delivery further by automatically deploying every change that passes the automated tests directly to production, without manual approval.After passing all tests, the system automatically deploys the software to Production.No manual approval is needed.Ensures fast and frequent releases, but requires a high level of automation and confidence in testing.Continuous Deployment. SourceGitLab supports all three practices through its built-in CI/CD pipelines.GitLab CI/CD Pipelines. SourceHow GitLab Works?GitLab is built using a distributed architecture that consists of multiple components to handle code management, CI/CD execution, security scanning, and deployment efficiently.SourceGitLab Instance (or Server)This is the heart of GitLab, where your code repositories, project data, and user accounts reside. Its the central hub for everything.Self-hosted: You install and manage GitLab on your own servers, giving you full control.GitLab.com: GitLabs cloud-based SaaS offering, where they handle the infrastructure.Components:Rails application (frontend and API)Gitaly (Git repository storage)PostgreSQL database (stores metadata, issues, etc.)Redis (caching and job queues)Sidekiq (background job processing)NGINX (web server and reverse proxy)GitLab Instance and Runners. SourceGitLab JobsJobs are individual tasks defined in a .gitlab-ci.yml file within a repository. They represent steps in the CI/CD pipeline, such as building code, running tests, or deploying applications.Jobs. SourceJobs are grouped into stages (e.g., build, test, deploy). Each job specifies a script to execute, an image or environment (e.g., Docker), and conditions (e.g., run only on specific branches).GitLab RunnersThese are agents that execute the jobs defined in your CI/CD pipelines. They do the heavy lifting of building, testing, and deploying your code.Runners can be installed on various machines (physical, virtual, containers) and operating systems, giving you flexibility in how you run your pipelines.SourceShared Runners: Used across multiple projects (Cloud-based or Self-hosted).Specific Runners: Assigned to a specific project.Overall ArchitectureSingle-Server Setup:All components (web server, database, Git storage, Sidekiq, Redis) run on one machine with embedded runners.Suitable for small teams or testing but limited in scalability.High-Availability Setup:Load Balancers: Distribute web traffic across multiple GitLab application nodes.Application Nodes: Run the web server and Sidekiq, connected to a shared filesystem (e.g., NFS) for Git data.Database Cluster: Replicated PostgreSQL instances for redundancy.Redis Cluster: For caching and job queuing, also replicated.Runners: Distributed across multiple machines or cloud instances, communicating with the GitLab server via the API.The GitLab instance uses REST and WebSocket APIs to coordinate with runners, while internal services (e.g., Sidekiq, Redis) handle asynchronous tasks.Git GitHub GitLabGitLab and Git are related but serve different purposes in the software development process.GitGit is a distributed version control system (DVCS) designed to handle everything from small to very large projects with speed and efficiency. It was created by Linus Torvalds in 2005 for the development of the Linux kernel.It manages the history of code changes (commits), supports branching and merging, and allows offline work by storing a full repository locally.Developers use Git locally to commit changes, then sync with a remote repository (e.g., on GitLab or GitHub) using git push and git pull.How Git Actually Works. SourceExample: A developer runs git commit -m "Add feature X" to save changes locally, and then,git push origin mainto upload to a remote server.GitHubGitHub is a web-based platform built on top of Git, providing a hosted service for Git repositories with additional collaboration and DevOps features. It was founded in 2008 and acquired by Microsoft in 2018.It offers a centralized place to store Git repositories, collaborate with teams, and integrate with development workflows.Developers push Git repositories to GitHub, use its UI for collaboration, and configure GitHub Actions for CI/CD. Its a hosted service (GitHub.com) with paid plans for private repositories.CI/CD (via Actions) is less integrated than GitLabs native solution, and advanced DevOps features require more external tools.Git vs GitHub. SourceExample: A team pushes code to a GitHub repo, opens a pull request for review, and uses GitHub Actions to run tests automatically.GitLabGitLab is an open-source, all-in-one DevOps platform that also uses Git as its version control foundation.It provides a complete software development lifecycle solution, from version control to CI/CD, monitoring, and security, all in one platform.Developers push Git repositories to GitLab, define pipelines in .gitlab-ci.yml, and use runners to execute jobs. The platform integrates everything version control, CI/CD, and collaboration natively.It is more integrated DevOps features out of the box compared to GitHub, with flexibility for self-hosting.GitLab vs GitHub. SourceExample: A team deploys a web app by pushing code to GitLab, running automated tests and security scans in the pipeline, and deploying to production with a single click.In conclusion, GitLab offers an all-in-one DevOps platform that streamlines collaboration, automation, and deployment. We need it to enhance efficiency, ensure code quality, and accelerate delivery in an increasingly competitive landscape.At its heart lies Git, the version control system that powers its functionality, while its core is the seamless integration of CI/CD pipelines, security, and project management.As we embrace GitLab, we should keep in mind the importance of proper configuration, regular updates, and leveraging its scalability to maximize its potential for any team or project.Next, we will install GitLab, configure it, and create a project. I will add the link here once I have written it.Read MoreSources //www.pagerduty.com/resources/learn/what-is-continuous-integration/ //www.turing.com/blog/github-vs-gitlab-key-differences Git isn't hard to learn, and when you combine Git with GitLab, you've made it a whole lot easier to share code and manage a common Git commit history with the rest of your team. This tutorial shows you how it's done. How to learn Git and GitLab While it takes time to fully master a DevOps tool such as Git, any developer who follows this 10-step Git and GitLab tutorial will become competent with these popular DevOps and CI/CD tools very quickly. The 10 steps to follow to learn Git and GitLab fast are as follows: Install Git and create a GitLab account. Create a Git repository in GitLab. Add files and commits to your GitLab repo. Create new Git branches. Toggle between branches in GitLab. Create a GitLab merge request. Approve a GitLab merge request. Clone the remote GitLab repo locally. Use the most important Git commands locally. Push your local changes back to GitLab. Step 1: Install Git and create a GitLab account To get started with Git and GitLab, you must first install Git locally and register for a GitLab account. The Git installation on Windows is straightforward. Simply download and run the installer, accept all the defaults and Git will be installed, along with the Git Bash and Git GUI tools. To register for a GitLab account, all you need to do is create a unique username, set your credentials and validate your email address. Once that's complete, you can move onto Step 2 of this Git and GitLab tutorial. Step 2: Create a repository in GitLab When you log in to GitLab for the first time, you are asked to create a project. Choose this option to create a new, blank Git repository on the GitLab server. Teams of developers use a Git repository to perform the following tasks: Track their files and their commit histories. Create branches and issue merge requests. Share code and collaborate with others. When the form to create a project appears, simply provide a name for the project and set the GitLab repo's visibility to public. Accept all the other defaults and submit the form to create your first GitLab repository. One of the first steps in any Git and GitLab tutorial has to be the creation of a GitLab repository. Step 3: Add files and create commits online The landing page for your new repository has a button with a plus sign on it. Click this button to create a new file. When you save the file, you will be forced to provide a commit message. With each save, you add to your Git commit history. Add three new files with the following names and Git commit messages: alpha.txt First Git commit bravo.txt Second Git commit charlie.txt Third Git commit Once added, the three files and their commit messages display on your GitLab repository's homepage. You can click on them to view their content or perform further edits. GitLab allows you to add files and create Git commits online. Step 4: Create Git branches and add files Note that you are currently on a branch called main. Using the same plus button as you did in Step 3, create a new branch named dev. Add another file named devo.txt and use 'Fourth Git commit' for the message. This new file will be added to the dev branch. It will not be visible on the main branch. Step 5: Navigate between Git branches Use the drop-down box to toggle between the dev and the main branch. Notice how the devo.txt file is on the dev branch, but it is not on the main branch. Git branches allow developers to do isolated development separately and independently of the work of other developers. This encourages creativity, experimentation and chance-taking. If an experiment doesn't work, the branch can be deleted. If the creative endeavor is a success, the new branch can be merged into the main branch for others to see. The devo.txt file is only in the dev branch. A git merge must be performed for it to be visible on the main branch on GitLab. Step 6: Create a GitLab merge request To bring the dev and main branches in sync, click the blue Create merge request button at the top of the screen. Fill out the title and the description and submit the merge request form. Merges can be performed locally or through GitLab merge requests online. Step 7: Accept a merge request in GitLab When you open the GitLab navigation bar on the left, you will see a new Merge request listed. Click the Merge requests link and then Approve to approve it. If you like Devo, give it a thumbs-up as well. Leave the "'Delete source branch" option selected and click Merge. When you return to the homepage of your repository, notice that the devo.txt file is in the main branch, and the dev branch is deleted. Learning how to approve a merge request is an important step in any Git and GitLab tutorial. Step 8: Clone the GitLab repository A clone of a GitLab repository brings all the files and branches on the server down to the developer's local machine. Most Git operations happen locally, in tandem with local software development. Find the GitLab URL Click the blue Code button on the repository's GitLab page and copy the HTTPS GitLab URL. Then open Git Bash in the folder where you want to store the folders and files for this repository and issue the git clone command: git clone After the clone command completes, you can find all the files in your Git repository under a subdirectory with the same name as your GitLab project. When you clone a GitLab repo, Git makes all the files in that repository available on your local machine. Step 9: Perform standard Git operations With the GitLab repository cloned locally, you can add files, edit files, delete files and play around with various Git commands including the following: Fetch, merge and pull are three of the important porcelain Git commands with which beginners should become comfortable. Step 10: Push your changes back to GitLab When you are done with local development, you can push your updated set of local commits back to the server with a push command: git push origin The first time you push using HTTPS, make sure you are logged into GitLab on your default web browser. You must be authenticated to push changes into a GitLab repository. GitLab authentication troubleshooting On a Windows machine, GitLab will try to validate your credentials using OAuth to log you in to the server. If the git push attempt results in a 403 error, you might need to obtain a GitLab personal access token and authenticate with that. If authentication continues to be an issue, you could alternatively create SSH keys for GitLab and authenticate over secure shell. GitLab is a full-featured and powerful CI/CD platform built around the popular Git tool. Learning how to use Git with GitLab is the first step to take advantage of GitLab's many features. Cameron McKenzie has been a Java EE software engineer for 20 years. His current specialties include Agile development, DevOps and container-based technologies such as Docker, Swarm and Kubernetes. View All Videos If you are looking for a better way to organize your project files, you should consider using GitLab. The beauty of GitLab is that it allows all team members to collaborate in all phases of a project, simplifying software development for all involved. In this GitLab tutorial for beginners, you will learn how to use GitLabs main features to manage your files and increase productivity (and even learn how Gitlab time tracking works with a help of Gitlab time tracking integration!) GitLab Definition GitLab is a web-based Git repository that allows software development teams to plan, code, test, deploy, and monitor product changes in one place. Git is a versioning system that tracks changes and enables pushing/pulling updates using remote resources. GitLab offers free, open, and private repositories, along with issue tracking and wikis. It helps teams collaborate throughout the software development lifecycle, enhancing product value and user satisfaction. Essential GitLab Basics Well start with the GitLab tutorial basicsits main features, pricing, installation, and how to use it. Planning GitLab offers powerful planning tools to keep your team in sync. The platform enables planning and management through epics, milestones, and groups, allowing teams to organize and track GitLab project progress. Coding GitLab tutorial wouldnt be complete without mentioning that GitLab allows developers to code directly in the platform. It helps to create a project using CI/CD pipelines. Now, lets see How to create a project. How to Create a Project?To create a project in GitLab, do the following steps:On the above GitLab welcome, Click on Create a project.We get a four options page to create a project in GitLab, as shown in the figure.Firstly, lets create a Blank project.Enter the Project name and Enable Private Visibility level, as shown in the figure, and then click Create project.An empty repository project has been created, as shown in the figure.Now open Git Bash and run the following commands, to help to create a project.Configure your identity using git config --global user.name "Your Name" git config --global user.email "email@example.com"Make a directory to store your GitLab projects mkdir GitLabTutorial cd GitLabTutorial/Configure your identity using git config --global user.name "Your Name" git config --global user.email "email@example.com"Make a directory to store your GitLab projects mkdir GitLabTutorial cd GitLabTutorial/Now initialize a folder by using the command git init.Now lets create a file by using the command below$ touch Mindmajix.txt?Run a Notepad file by using the command below$ notepad Mindmajix.txt?You will get a Notepad file naming Mindmajix. Write a few words in the file, as shown in the figure.Check the status$ git statusAdd the file$ git add .?Commit the file$ git add .?Check the status$ git add .?Now go to GitLab, click Projects dropdown in the menu, and open Your Projects, as shown in the figure.Add remote origin$ git remote add originKeerthana_Jonnalagadda/gitlab-tutorial.?Fetch the file$ git remote -v?Push the file$ git push -u origin master 7.Open GitLab, check Your projects, and you can see the Mindmajix.txt file which is created to push into the GitLab as shown in the figure Click on the Mindmajix.txt file. 8.You can see the data in the file, as shown in the figure. Likewise, we can push our files, folder, and records into GitLab.Create from TemplateWe can also create a project using 24 in-built templates of different sources like Ruby on Rails, Spring, iOS Swift, .NET Core, and many more as shown in the figure.From the list available in-built templates, click Preview to view the template source and click Use template to start creating the template. Rest the process is the same as creating a blank project.[]Create from Importing a ProjectWe can import projects from various Git and other sources to GitLab, as shown in the figure.Click Create project to GitLab, we can connect to an external repository using a CI/CD external repo.Choose GitHub or Repo by URL to connect with an external repositoryHow to Create a Group in GitLab?The group is a collection of projects that helps to manage and organize the projects like a folder. It helps to maintain the group members access and permissions for each project in the group. To create a group, do follow the steps.Open GitLab, click Groups on the top menu bar, and then click on New group to create a group as shown in the figure.Now you get a page to create a group, as shown in the figure.Enter the group name, click Create group, and you can see the group is created as shown in the figure.Now click on the created group and click Edit group settings to edit the group as shown in the figure.Now you get a page to edit the group, as shown in the figure.Open project, click on Settings and then click General as shown in the figureA general settings page opens. Choose project avatar and select your avatar was click Save changes.You can see the project avatar is updated.We can also add, a avatar while creating the project.Add users to the groupAdding users to the group makes an organized work that benefits time to access files. Admin can restrict the access of the user and can even schedule a time for a deadline to work on the project. To add users to the group, do the following steps.Open your Group, click on Members in the groups dashboardMembers page opens, as shown in the figure.Enter GitLab member or Email address to invite into the group.We can choose the role of a particular member such as Guest, Reporter, Developer, Maintainer, and Owner.We can schedule the access expiration data that helps to organize the work in a schedule.And then Click Invite to add the member into the group.This way, we can invite members or users to the group.Changing the owner of the groupThe ownership of group members must have at least one owner. Changing the owner of a group is only possible by one owner. To change the owner of the group, do the following steps.As a current ownerGo to groups, click Members in the group dashboard.Assign different user Owner permissions.Have a new owner sign in and remove Owner permission from you(owner).As an administrator,Go to groups, click Members in the group dashboard.Assign different user Owner permissions.Now refresh the page. We can remove permissions from the original owner.Remove members from the groupIn GitLab, an owner is a person who manages the group members.To remove a member from the group, do the following steps:In the group, click Members in the group dashboard.Click the Delete button which is next to a group member to remove. A remove member window appears.Enable Also unassign this user from related issues and merge requests checkbox.Click Remove member.Project forking workflowA fork is a copy of an original repository that we make in the namespace and can apply changes later without affecting the original project. It takes a few steps to fork a project in GitLab.Create a forkTo create a fork, do the following steps:On the project page, pick a project and click on the fork icon as shown in the figureSelect the project to create a copy of the project.The project has been forked successfully.Request for merging.A request is required to merge the changes in the forked project to the original project.Click on Create merge request.Now you get a page to create a merge request, as shown in the figure.Select branches, and push into the original project and click on Compare branches and continue.A New merge request page opens. Give all the required detail to merge and click Submit merge request.A request has been sent to the person for merging the changes in the project.Removing a forkIn GitLab, we can even remove the fork relationship that helps not to send merge requests to the upstream project. Only project owners have permission to remove the fork of a project. To remove a fork, do follows the steps below:Go to projects, Click on Settings and then click General as shown in the figure.A General Settings page opens in that list, scroll down to Advanced and click Expand.An advanced settings page opens in that scroll down to Remove fork relationship and click on it.A confirmation window appears. Type the project link to proceed and click Confirm.ConclusionOn winding up, GitLab is a user-friendly and open-source tool that helps to store the file and organize them more easily. It allows developers to coordinate easily to work on the project. GitLab is easy to manage and configure the projects, and it has relied on Git, where the number of users is not restricted and can create more building applications. We have learned the basic details of GitLab, How to create a project, group, and how to fork a project. I hope this tutorial has helped you to understand GitLab. No one codes alone. Because tech teams can grow at breakneck speeds, solitary coders quickly become part of large teams of developers that need to cooperate and coordinate their efforts. Git has become the tool for source code management, and plenty of platforms have popped up to make Git accessible to dev teams. GitLab is one of them. This beginners guide covers: The GitLab Dashboard What youll find in your toolbar How to create your first project Whether youre a project manager looking to roll this out to your team, or a developer getting onto a Git platform for the first time, youre sure to get something out of this guide. Heres what your dashboard will look like the first time you open up GitLab. Nothing too crazy so far. This screen is designed for you to get an overview of your GitLab activity. Specifically, it shows you projects youve been working on, issues and merge requests assigned to you, and todos. Todos are a handy GitLab feature that let you create your own todo list. You can add todos from issues and merge requests, and mark them as done when youve completed the task. This is a huge time saver when onboarding multiple people. Think of GitLab groups a bit like a folder in Google Drive. If you dozens of documents on policies and best practices for a dev team, you wouldnt grant your team access document by document; youd stick them in a folder and grant access to the folder in one go. Explore public projects: You can access a huge library of public GitLab projects here. Maybe you want to use public projects to figure out how to organize your own, or youre looking to grab a copy of open source software and contribute to it. Learn more about GitLab: This is where you can access GitLabs documentation, geared towards helping you understand and use the platform. It covers everything from creating a project to using CI/CD pipelines. GitLabs documentation can all be edited directly in GitLab through merge requests, a bit like a wiki. So when you get to the point where you can call yourself a GitLab pro, you can give back by contributing to GitLabs documentation. Now that weve gone over what youll see when you first log in, lets go over what else you can do in GitLab. Lets start with the upper-left corner of the toolbar. Projects: Once youve created or been given access to projects, this is where youll access them. The drop-down lets you access three screens: Your Projects, Starred Projects, and Explore Projects. The Explore Projects option is how you can browse public projects. Groups: From here, you can find groups youre already a part of as well as public groups. You can request to join any group made public and contribute to their projects. More: This dropdown gives you access to dashboards that help you track a projects progression such as analytics and milestones as well as snippets, which are a bit like post-its for code. Well go over this in more detail later. Now, whats on the right side of the toolbar? From left to right: New: This drop-down menu lets you create new projects, groups or snippets. Search bar: From here, you can quickly find projects, groups, issues, and more. Issues: Issues are how work gets done in GitLab. Create an issue and assign it to yourself to keep track of what you need to get done. Assign it to another member of the team when you need something from them. You can give an issue a checklist to track progress on a more complicated task or link them to other issues to show dependencies. Merge Requests: Merge requests are created by developers when theyre ready to integrate their branch of code with the source code. Theyre virtually identical to pull requests in GitHub. It is how you can add your code changes to the master branch of a project. Todos: Todos are a handy GitLab feature that let you create your own todo list. You can add todos from issues and merge requests, and mark them as done when youve completed the task. This is a huge time saver when onboarding multiple people.