CircleCI vs GitLab

GitLab compared to other DevOps tools

CircleCI is a continuous integration server (available both cloud-based and self-hosted) that supports any language that builds on Linux or macOS, including C++, Javascript, .NET, PHP, Python, and Ruby. Like GitLab, CircleCI has been named a leader in CI by Forrester in 2017. CircleCI focuses on the orchestration and automation of the CI/CD pipeline and integrates to several industry standard tools for all other functionality. In contrast, GitLab offers a single integrated application providing automation of the entire DevOps lifecycle, based on tight out-of-the-box integrations with industry leading open source tools. GitLab also has the ability integrate to all other tools using CLI, API, and webhooks.

FEATURES		

Application performance monitoring

GitLab collects and displays performance metrics for deployed apps, leveraging Prometheus. Developers can determine the impact of a merge and keep an eye on their production systems, without leaving GitLab.

Learn more about monitoring deployed apps

		

Application performance alerts

GitLab allows engineers to seamlessly create service level indicator alerts and be notified of any desired events, all within the same workflow where they write their code.

Learn more about creating SLI alerts

		

GitLab server monitoring

GitLab comes out of the box enabled for Prometheus monitoring with extensive instrumentation, making it easy to ensure your GitLab deployment is responsive and healthy.

Learn more about monitoring the GitLab service

		

Cycle Analytics

GitLab provides a dashboard that lets teams measure the time it takes to go from planning to monitoring. GitLab can provide this data because it has all the tools built-in: from the idea, to the CI, to code review, to deploy to production.

Learn more about Cycle Analytics

		

Built-in Container Registry

GitLab Container Registry is a secure and private registry for Docker images. It allows for easy upload and download of images from GitLab CI. It is fully integrated with Git repository management.

Documentation on Container Registry

 

Preview your changes with Review Apps

With GitLab CI/CD you can create a new environment for each one of your branches, speeding up your development process. Spin up dynamic environments for your merge requests with the ability to preview your branch in a live environment.

Learn more about Review Apps

 

A comprehensive API

GitLab provides APIs for most features, allowing developers to create deeper integrations with the product.

Read our API Documentation

 

Built for containers and Docker

GitLab ships with its own Container Registry, Docker CI Runner, and is ready for a complete CI/CD container workflow. There is no need to install, configure, or maintain additional plugins.

 

Cloud Native

GitLab and its CI/CD is Cloud Native, purpose built for the cloud model. GitLab can be easily deployed on Kubernetes and used to deploy your application to Kubernetes with support support out of the box.

Kubernetes integration

 

Comprehensive pipeline graphs

Pipelines can be complex structures with many sequential and parallel jobs. To make it a little easier to see what is going on, you can view a graph of a single pipeline and its status.

Learn more about pipeline graphs

 

Browsable artifacts

With GitLab CI you can upload your job artifacts in GitLab itself without the need of an external service. Because of this, artifacts are also browsable through GitLab's web interface.

Learn more about using job artifacts in your project


Scheduled triggering of pipelines

You can make your pipelines run on a schedule in a cron-like environment.

Learn how to trigger pipelines on a schedule in GitLab


Code Quality

Code Quality reports, available in the merge request widget area, give you an early insight into how the change will affect the health of your code before deciding if you want to accept it.

Learn more about Code Quality reports


Multi-project pipeline graphs

With multi-project pipeline graphs you can see how upstream and downstream pipelines are linked together for projects that are linked to others via triggers as part of a more complex design, as it is for micro-services architecture.

Learn more about multi-project pipeline graphs


Protected variables

You can mark a variable as "protected" to make it available only to jobs running on protected branches, therefore only authorized users can get access to it.

Learn how to use protected variables


Environments and deployments

GitLab CI is capable of not only testing or building your projects, but also deploying them in your infrastructure, with the added benefit of giving you a way to track your deployments. Environments are like tags for your CI jobs, describing where code gets deployed.

Learn more about environments


Environments history

Environments history allows you to see what is currently being deployed on your servers, and to access a detailed view for all the past deployments. From this list you can also re-deploy the current version, or even rollback an old stable one in case something went wrong.

Learn more about history of an environment

Environment-specific variables

Limit the environment scope of a variable by defining which environments it can be available for.

Learn how to configure environment-specific variables

Group-level variables

Define variables at the group level and use them in any project in the group.

Learn how to configure variables

Customizable path for CI/CD configuration

You can define a custom path into your repository for your CI/CD configuration file.

Learn how to configure a custom CI/CD configuration file

Run CI/CD jobs on Windows

GitLab Runner supports Windows and can run jobs natively on this platform. You can automatically build, test, and deploy Windows-based projects by leveraging PowerShell or batch files.

Install GitLab Runner on Windows

Run CI/CD jobs on macOS

GitLab Runner supports macOS and can run jobs natively on this platform. You can automatically build, test, and deploy for macOS based projects by leveraging shell scripts and command line tools.

Install GitLab Runner on macOS

Run CI/CD jobs on Linux ARM

GitLab Runner supports Linux operating systems on ARM architectures and can run jobs natively on this platform. You can automatically build, test, and deploy for Linux ARM based projects by leveraging shell scripts and command line tools.

Install GitLab Runner on Linux

Run CI/CD jobs on FreeBSD

GitLab Runner supports FreeBSD and can run jobs natively on this platform. You can automatically build, test, and deploy for FreeBSD-based projects by leveraging shell scripts and command line tools.

Install GitLab Runner on FreeBSD

 

Show code coverage rate for your pipelines

GitLab is able to parse job output logs and search, via a customizable regex, any information created by tools like SimpleCov to get code coverage. Data is automatically available in the UI and also as a badge you can embedd in any HTML page or publish using GitLab Pages.

Learn how to generate and show code coverage information in GitLab

 

Manage JUnit reports created by CI jobs

Many languages use frameworks that automatically run tests on your code and create a report: one example is the JUnit format that is common to different tools. GitLab supports browsing artifacts and you can download reports, but we're still working on a proper way to integrate them directly into the product.

Read more on the issue

 

Details on duration for each command execution in GitLab CI/CD

Other CI systems show execution time for each single command run in CI jobs, not just the overall time. We're reconsidering how job output logs are managed in order to add this feature as well.

Read more on the issue

 

Auto DevOps

Auto DevOps brings DevOps best practices to your project by automatically configuring software development lifecycles by default. It automatically detects, builds, tests, deploys, and monitors applications.

Read more about Auto DevOps in the documentation

 

Protected Runners

Protected Runners allow you to protect your sensitive information, for example deployment credentials, by allowing only jobs running on protected branches to access them.

Read more on the issue

 

Easy integration of existing Kubernetes clusters

Add your existing Kubernetes cluster to your project, and easily access it from your CI/CD pipelines to host Review Apps and to deploy your application.

Read more on the issue

Easy creation of Kubernetes clusters on GKE

Create a Kubernetes cluster on GKE directly from your project, just connecting your Google Account and providing some information. The cluster can be used also by Auto DevOps to deploy your application.

Read more on the issue

Support for multiple Kubernetes clusters

Easily deploy different environments, like Staging and Production, to different Kubernetes clusters. This allows to enforce strict data separation.

Read more on the issue

Easy Deployment of Helm, Ingress, and Prometheus on Kubernetes

Install Helm Tiller, Nginx Ingress, Prometheus and GitLab Runner directly into your cluster from the GitLab Web UI with one click.

Read through the documentation on installing applications on GKE clusters

Canary Deployments

GitLab Enterprise Edition Premium can monitor your Canary Deployments when deploying your applications with Kubernetes.

Learn more about configuring Canary Deployments

Automatic Retry for Failed CI Jobs

You can specify a retry keyword in your .gitlab-ci.yml file to make GitLab CI/CD retry a job for a specific number of times before marking it as failed.

Learn more about Automatic Retry for Failed CI Jobs

Pipelines security

The ability of running CI/CD pipelines on protected branches is checked against a set of security rules that defines if you're allowed or not. It includes creating new pipelines, retrying jobs, and perform manual actions.

Learn more about pipeline security

Include external files in CI/CD pipeline definition

You can include external files in your pipeline definition file, using them as templates to reuse snippets for common jobs.

Learn more about including external files

 

Static Application Security Testing

GitLab allows easily running Static Application Security Testing (SAST) in CI/CD pipelines; checking for vulnerable source code or well known security bugs in the libraries that are included by the application. Results are then shown in the Merge Request and in the Pipeline view. This feature is available as part of [Auto DevOps](https://docs.gitlab.com/ee/topics/autodevops/#auto-sast) to provide security-by-default.

Learn more about Static Application Security Testing

 

Dependency Scanning

GitLab automatically detects well known security bugs in the libraries that are included by the application, protecting your application from vulnerabilities that affect dependencies that are used dynamically. Results are then shown in the Merge Request and in the Pipeline view. This feature is available as part of [Auto DevOps](https://docs.gitlab.com/ee/topics/autodevops/#auto-dependency-scanning) to provide security-by-default.

Learn more about Dependency Scanning

 

Container Scanning

When building a Docker image for your application, GitLab can run a security scan to ensure it does not have any known vulnerability in the environment where your code is shipped. Results are then shown in the Merge Request and in the Pipeline view. This feature is available as part of [Auto DevOps](https://docs.gitlab.com/ee/topics/autodevops/#auto-container-scanning) to provide security-by-default.

Learn more about container scanning

 

Dynamic Application Security Testing

Once your application is online, GitLab allows running Dynamic Application Security Testing (DAST) in CI/CD pipelines; your application will be scanned to ensure threats like XSS or broken authentication flaws are not affecting it. Results are then shown in the Merge Request and in the Pipeline view. This feature is available as part of [Auto DevOps](https://docs.gitlab.com/ee/topics/autodevops/#auto-sast) to provide security-by-default.

Learn more about application security for containers

 

Interactive Application Security Testing

[IAST](https://blogs.gartner.com/neil_macdonald/2012/01/30/interactive-application-security-testing/) combines elements of static and dynamic application security testing methods to improve the overall quality of the results. IAST typically uses an agent to instrument the application to monitor library calls and more. GitLab does not yet offer this feature.


Runtime Application Security Testing

RASP uses an agent to instrument the application to monitor library calls as the application is running in production. Unlike other security tools, RASP can take action to block threats in real-time, similar to a Web Application Firewall but from within the app's runtime environment rather than at the network layer. GitLab does not yet offer this feature.


Browser Performance Testing

Easily detect performance regressions for web apps, prior to merging into master. Browser Performance Testing is included in Auto DevOps, providing automatic performance analytics of the root page with zero configuration.

Learn more about Browser Performance Testing


Step folding for CI/CD logs

Collapse the job log output for each command.

Read more on the issue


View Kubernetes pod logs

Quickly and easily view the pod logs of an app deployed to Kubernetes.

Learn more about viewing Kubernetes pod logs