

# Bamboo vs GitLab

## GitLab compared to other tools

Bamboo Server is a CI/CD solution which is part of the Atlassian suite of developer tools. It is available only in a self-managed configuration and is a closed source solution. Bamboo offers build, test, and deployment automation, and has tight integrations to Atlassian BitBucket (for SCM) and Fisheye (for understanding how source code has changed), as well as integrations to over 150 other tools. In contrast, GitLab offers a git-based SCM, SCM integrations, and code change traceability out of the box in a single application.

Bamboo offers a GUI for defining build plans, but does not offer pipeline as code. Bamboo also offers deployment plans (which include the notion of environments and releases), pre-deployment visibility, and per-environment deployment permissions. GitLab also offers release tracking across environments and deep visibility into the changes in a deployment, but sets deployment permissions based on branch permissions.

Bamboo steps can be run in parallel across agents, and those agents can be auto-scaled based on need if Bamboo is configured for a feature called Elastic Bamboo. Elastic Bamboo requires the use of "remote agents", which you pay extra for. Organizations who want auto-scaling are also locked in to using Amazon Elastic Compute Cloud (EC2) and paying Amazon separately for their usage. In contrast, GitLab does not charge per remote agent (runner) and scales with a variety of cloud and container solutions.

## FEATURES



### Built-in CI/CD

GitLab has built-in Continuous Integration/Continuous Delivery, for free, no need to install it separately. Use it to build, test, and deploy your website (GitLab Pages) or webapp. The job results are displayed on merge requests for easy access.



[Learn more about CI/CD](#)

### 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](#)

### 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](#)

---

## 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](#)

---

## GitLab CI/CD Horizontal Autoscaling

GitLab CI/CD cloud native architecture can easily scale horizontally by adding new nodes if the workload increases. GitLab Runners can automatically spin up and down new containers to ensure pipelines are processed immediately and minimize costs.



[Learn more about GitLab CI/CD Horizontal Autoscaling](#)

---

## 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](#)

---

## Container debugging with an integrated web terminal

Easily debug your containers in any of your environments using the built-in GitLab Web Terminal. GitLab can open a terminal session directly from your environment if your application is deployed on Kubernetes. This is a very powerful feature where you can quickly debug issues without leaving the comfort of your web browser.



[Learn more about the web terminal](#)

---

## 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](#)

---

## Online visualization of HTML artifacts

Access your test reports, code quality and coverage information directly from your browser, with no need to download them locally.



[Learn more about using job artifacts in your project](#)

---

## 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](#)

---

## Deployment projects

A deployment project holds the software project you are deploying: releases that have been built and tested, and the environments to which releases are deployed.



[Learn about GitLab projects](#)

---

## 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](#)

---

## Per-environment permissions

Developers and QA can deploy to their own environments on demand while production stays locked down. Build engineers and ops teams spend less time servicing deploy requests, and can gate what goes into production.



[Learn about protected branches in GitLab](#)

---

## 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](#)

---

## Bad Test Quarantine

Don't let red builds become the norm. Across all tests, keep flakey or broken tests out of sight (but not out of mind), and keep the build green with one-click quarantine of tests.



[Learn how to dismiss vulnerabilities in GitLab](#)