Bamboo vs GitLab

GitLab compared to other DevOps tools

On this page

Summary

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 (see pricing). 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.

Comments/Anecdotes

- Discussion from HackerNews article about Atlassian not allowing benchmarking

  > Atlassian has always forbidden to talk about the performance of their products in their ToS and in their previous EULA. We all know why, but we donâ€™t talk about it.

- Sales heard from large networking company

  > "Terrible!, All UI Based, Cannot configure 'as code' (ala .gitlab-ci.yml)"

- Sales previous experience)

  > Previously we beat Bamboo by the reason that itâ€™s too prescripted for build only.

  This is likely no longer true as they now have deployment specific features including environments, deployment groups, and per-environment deployment permissions

- From Twitter:
  - "we have also started using GitLab (moving from our own BitBucket/Bamboo servers). The CI/CD is definitely an improvement but I'm not sold on the code review features of GitLab" https://twitter.com/carrchr/status/1003651960099176448
  - "Sweet. Unlike Atlassian's Bamboo, @GitLab CI supports "[ci skip]" out of the box. #gitlab #devtools #success"

https://twitter.com/tekkie/status/823689378371342336

- "Nice and simple GUI in @GitLab CI as well. Much easier to navigate than Bamboo for instance. #gitlab #gui #ux #success"
  https://twitter.com/tekkie/status/839054544009117696
- "Seriously, is still there a reason to use Jenkins/Hudson/TeamCity/Bamboo? I reckon @GitLab built-in CI support is sufficient for most of us!"
  https://twitter.com/AriyaHidayat/status/756919101587546112
- "A day building pipelines in VSTS for one of my teams. Get the feeling that while I find it simple not everyone finds it the same. Better than Gitlab CI (IMO) but I like Bamboo a lot tooâ€¦"
  https://twitter.com/xyglo/status/978291270161457152

- From Bamboo open Issues
  - **Issue:** If I want to use git submodules then I shouldn't have to upload and configure SSH keys on each Bamboo Agent.
    - **Key text:** "Bamboo requires separate Git authentication for submodules. This involves either using HTTPS for submodules and providing the credentials through the job's environment variables, or configuring separate SSH keys on each build agent. Using HTTPS would render local builds unusable, requiring credentials every time. Adding SSH keys to every Bamboo agent is unmaintainable. . . . This reason, among others is a large part of why we have migrated away from Bamboo. We now use Gitlab and Gitlab CI for much better Docker and git support."
    - **Link:** Bamboo Issue in Jira

Resources

- Atlassian Bamboo Website

Pricing

- Price page
- Bamboo Pricing Guide
  (includes price additions for remote agents, and academic pricing)
- Small Teams - $10/month - only 10 jobs and no remote agents
- Growing Teams - $880/month - unlimited jobs, 1 remote agent
- pricing increase in tiers by # remote agents (1, 5, 10, 25, 100, 250, 500, 1000) (see Bamboo Pricing Guide for prices)
- First purchase includes perpetual software and 1 yr maintenance. Yearly cost for maintenance is approximately 50% of initial remote agent tier cost. (e.g. 1st year @25 remote agents = $8,800, second year maintenance = $4,400)

Comparison

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	✓	✓
Runs with less memory and consumes less CPU power Uses little memory, it runs fine with 512MB. Uses little CPU power since Go is a compiled language	✗	✗

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

 

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