## Summary

Buildkite is a continuous integration tool designed to improve software developer productivity. Buildkite product consists of three main components:
- **Agent**: A small, cross-platform build runner that enables an enterprise to run its builds in their own infrastructure. Agents run on several platforms including Ubuntu, Debian, Mac, Windows, Docker, and others. Agents can only run one build job at one time.
- **Pipelines**: Are containers to define and model the build workflows and also to kick off post-build steps.
- **Deployments**: These are steps that run after a build is complete and can be either triggered automatically or manually.

As a smaller company with limited resources, Buildkite has taken a strategy of using plug-ins to extend its functionality and to integrate with other software. It is unclear how many community plug-ins have been developed. However as of 3/10/2020 there are about 19 supported plug-ins provided by Buildkite.

## Strengths

- **Parallel Testing**: One of the key value props for Buildkite is the ability to run parallel tests on builds, thus reducing test times from hours to minutes. Buildkite does this using its parallelism approach to spawn out multiple agents (e.g. several hundred parallel agents). With different tests running on different agents, this massive parallel testing is possible due to the light-weight build agent.
Jobs
- Runs on-premises and on AWS infrastructure
- Agents can be deployed on multiple platforms.
- Pipelines as code: Ability to define complex pipelines in yml including running any scripts, tool or shell commands within the pipeline.

Gaps
- Buildkite is a point solution for continuous integration and currently lacks other critical DevOps steps such as Issue Management, Source Code Management, Security Testing etc. These capabilities require integrations with other tools.
- Security - Buildkite relies on third party providers for security testing. Calling third party code to launch security tests would require special scripts and execution of APIs from within Buildkite's pipeline definition.
- Lacks native support for an Artifact Repository. File paths to various artifacts have to be noted and maintained by users.
- Secrets management is possible but users have to either write plug-ins to standard secrets management products such as Hashicorp or write custom scripts to pull secrets from a secure storage and apply them at the right step in the build process.
- Collaboration features are limited to developers annotating comments inline within log files and other points of failure.

Resources
- Buildkite Documentation
- Buildkite Videos

Feature Comparison

Free CI/CD with shared or personal Runners

<table>
<thead>
<tr>
<th>Features</th>
<th>Buildkite</th>
<th>GitLab</th>
</tr>
</thead>
<tbody>
<tr>
<td>Core</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Starter</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Premium</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Ultimate</td>
<td>✔</td>
<td>✔</td>
</tr>
</tbody>
</table>

GitLab.com has shared Runners that allow you to use GitLab CI/CD completely free up to 2000 build minutes for private projects and unlimited for public projects. Alternatively, you can set up your own Runner for faster build processing, unlimited build minutes, or special requirements.

Explore GitLab offerings

Built-in CI/CD

<table>
<thead>
<tr>
<th>Features</th>
<th>Buildkite</th>
<th>GitLab</th>
</tr>
</thead>
<tbody>
<tr>
<td>Core</td>
<td>-</td>
<td>✔</td>
</tr>
<tr>
<td>Starter</td>
<td>-</td>
<td>✔</td>
</tr>
<tr>
<td>Premium</td>
<td>-</td>
<td>✔</td>
</tr>
<tr>
<td>Ultimate</td>
<td>-</td>
<td>✔</td>
</tr>
</tbody>
</table>

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

CI/CD Horizontal Autoscaling

<table>
<thead>
<tr>
<th>Features</th>
<th>Buildkite</th>
<th>GitLab</th>
</tr>
</thead>
<tbody>
<tr>
<td>Core</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Starter</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Premium</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Ultimate</td>
<td>✔</td>
<td>✔</td>
</tr>
</tbody>
</table>

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

CI/CD Pipelines Dashboard

<table>
<thead>
<tr>
<th>Features</th>
<th>Buildkite</th>
<th>GitLab</th>
</tr>
</thead>
<tbody>
<tr>
<td>Core</td>
<td>-</td>
<td>✔</td>
</tr>
<tr>
<td>Starter</td>
<td>-</td>
<td>✔</td>
</tr>
<tr>
<td>Premium</td>
<td>-</td>
<td>✔</td>
</tr>
<tr>
<td>Ultimate</td>
<td>-</td>
<td>✔</td>
</tr>
</tbody>
</table>

Visualize the history and current status of pipelines across projects and groups all in a single dashboard that can be customized for each user.

Learn more about Cross-Project Pipelines in the Operations Dashboard

Online visualization of HTML artifacts

<table>
<thead>
<tr>
<th>Features</th>
<th>Buildkite</th>
<th>GitLab</th>
</tr>
</thead>
<tbody>
<tr>
<td>Core</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Starter</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Premium</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Ultimate</td>
<td>✔</td>
<td>✔</td>
</tr>
</tbody>
</table>
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

### 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 MR Widget

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

Learn more about Code Quality

### Code Quality Reports

Full Code Quality reports are available on the pipeline page, showing areas of the codebase that do not meet the organization’s preferred style or standards.

Learn more about Code Quality reports

### 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

### 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 can 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 embed in any HTML page or publish using GitLab Pages.

Learn how to generate and show code coverage information in GitLab

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

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

Minimal CI/CD configuration

GitLab CI/CD requires less configuration for your pipelines than other similar setups like Jenkins.

Learn more about GitLab CI/CD

Built-in and custom project templates

When creating a new project, you can choose to kickstart your project from a predefined template that already has some working example code and CI preconfigured. In addition, you can define a custom project templates by assigning a group. Child projects of this group are available as templates when creating a new project.

Read more about Project templates

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

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

Load Performance Testing

Easily detect performance regressions for software under load, prior to merging into master.

Learn more about Load Performance Testing

Step folding for CI/CD logs

Collapse the job log output for each command.

Documentation

CI/CD for external repo

Connect your projects hosted on external services (like GitHub or Bitbucket) and leverage the power of GitLab CI/CD pipelines to build, test, and deploy your applications easily.

Learn more about CI/CD for external repositories

CI/CD for GitHub

Connect your projects hosted on GitHub and leverage the power of GitLab CI/CD pipelines to build, test, and deploy your applications easily.

Learn more about CI/CD for GitHub

Bad test quarantine

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

Read more on the issue

Unit Test Report

GitLab allows you to view unit test results for a pipeline, giving you insight into the test execution for the pipeline.

Learn more about unit test reports

See unit test summaries in merge request widget
GitLab allows you to view unit test results from the merge request widget, giving you insight into quality impacts of your changes.

Learn more about unit test summaries in the merge request widget

**Interactive Web Terminals**

Interactive web terminals allow you to connect to a running or completed Kubernetes, Docker, or Shell runner job and manually run commands to better understand what's happening in the system.

Learn more about Interactive Web Terminals

**Run jobs only when there are changes to a file or path**

Jobs can be configured to run only when there are changes to a specific file or path, giving you control over execution to allow for more complex build pipelines optimized for the changes in each commit.

Learn more about Only/Except for changes

**Windows Container Executor**

With this feature you are able to use Docker containers on Windows directly, in much the same was as if they were on Linux hosts. This enables more advanced kinds of pipeline orchestration and management for users of Microsoft platforms.

Learn more about the Windows Container Executor

**Pipeline deletion**

Deleting a pipeline is possible using the API and also in the UI on the Pipeline Details page. This allows for cases where perhaps secrets have been leaked in a pipeline, many unneeded pipelines have been created, or other issues have occurred where pipelines need to be deleted.

Learn more about only/except for merge requests

**Explicit support for monorepos**

The ability to execute jobs only/except when there are changes for a given path or file support monorepos where many microservices are contained in a single repo.

Learn more about only/except CI/CD execution

**First class container building**

The ability to specify that a container should be built during a CI/CD job without needing to specify the implementation details.

Read more on the issue

**Trigger pipeline on any event in code repository**

Enables pipelines/workflows to be started based on when any defined event is executed in the code repository. For example, could run a workflow to send a welcome email on adding a new member to a repository or project.

Docs on GitLab triggerable events

**Trigger pipeline on any event in code repository app eco-system**
Enables pipelines/workflows to be started based on when any defined event is executed in the code repository or in any app extension of that repository’s eco-system. For example, when an event happens in the Slack integration, update a repo work item.

**Community powered workflows (configuration is code so are shareable)**

GitLab pipeline (workflows) are defined as yml in repos and can be shared just like actions.

**Any platform, any language, and cloud**

Can run on any OS platform, for any language, and on any cloud provider.

**No configuration, infrastructure setup, or patching necessary**

As a SaaS offering, can provide software development and delivery services without the need to set up the tool itself, infrastructure to run it, and to maintain it by patching.

**Auto suggest pipelines to start based on code language**

Through language detection, auto suggest pipeline templates to run to help users quickly get a pipeline running.

**Advanced CI/CD configuration linter**

The CI linter provide warnings and error messages when validating your `.gitlab-ci.yml` file, helping to get up and running quickly with GitLab pipelines.

Learn more about the CI YAML linter

**Comes with many pre-defined pipelines**

Offers many pre-defined pipelines that capture best practice and make it easy for a user to get started with each project for common languages, platforms, and configurations.

**Connects the diff tools & services used during the SDLC**

Can be used as a central glue to orchestrate, and connect data and outputs from your many different tools & services.

**Matrix builds**

Built-in ability to define and execute builds that automatically trigger a number of parallel jobs or pipelines based on a multitude of input variables. For example, building for 3 OS’s at once, and for 3 different versions of libraries, would automatically be done in 9 parallel jobs. At GitLab, this is implemented using dynamic child pipelines.

Learn more about child/parent pipelines

**Live streaming of logs from running pipeline**

Ability to see live job logs (while the pipeline is running).
<table>
<thead>
<tr>
<th>Feature</th>
<th>Free</th>
<th>Bronze</th>
<th>Silver</th>
<th>Gold</th>
</tr>
</thead>
<tbody>
<tr>
<td>Search across all job logs</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Enables more efficient search for errors and other</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>content of interest while troubleshooting or reviewing</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>job output</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>View raw logs in plaintext</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Ability to get the plain text of a log, no mark up, to</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>be able to share it or use it externally</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Multiple pipelines per repo</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Ability to define multiple pipelines per code repository</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>to enable either different processes to be run at</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>different times, and/or to enable monorepos where there</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>are multiple applications within one repo</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>which need to be built and handled differently per</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>application</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Read more about child/parent pipelines</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reference actions/jobs in another repo</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Ability to have pipelines/workflows reference and use</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>actions/jobs from a repo different from the one it is</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>being run from, without needing any installation</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>