# GitLab vs Codefresh

## Decision Kit

<table>
<thead>
<tr>
<th>GitLab</th>
<th>Codefresh</th>
</tr>
</thead>
<tbody>
<tr>
<td>75% (54.5/73 Requirements)</td>
<td>35% (26/73 Requirements)</td>
</tr>
</tbody>
</table>

### Summary

Codefresh is a CI/CD tool designed for containers and Kubernetes. Codefresh features a GitLab integration that allows you to use GitLab for version control and run Codefresh pipelines on your code. Codefresh has some features that make it more mature than GitLab for running pipelines across multiple projects. But it lacks the benefits of a single application.

Codefresh charges for builds per month, as well as concurrent builds. GitLab has no such limitations with the ability to elastically scale runners to handle as many concurrent builds as needed on demand and then scale down so you aren't paying to keep inactive runners up.

Codefresh only offers a self-managed option for Enterprise pricing. Free, Basic, and Pro tiers are for SaaS-only. GitLab offers self-managed and SaaS options at every price point.

### Resources

- Codefresh homepage
- Comparison page on their site
- Codefresh GitLab integration

### Comments/Ancedotes

- Codefresh makes some claims in their blog comparing themselves to GitLab that are not really accurate.
  - "GitLabCI isn't designed for micro-services since everything is tied to a single project*
    - Although we can improve our microservices support, (and plan to) this claim is not true.
  - Today, GitLab has multi project pipelines and can trigger pipelines for multi-project via API. In fact, The CI working group for CNCF chose GitLab to run their multi-project multi-cloud pipelines. Read more in the CNCF case study.
- We are missing some features identified in this post that would bring us on par with Codefresh
  - The ability to define multiple pipelines
  - Support for monorepos with the ability to run pipelines only on specific directories - UPDATE: Shipped in GitLab 11.4
  - Group level Docker registry browser
Additional Codefresh Capabilities

- Private Helm repository
- Global Helm repo configuration
- Helm Repository browser
- Helm release boards
- Helm environment board
- Zero configuration caching/artifacts using docker volumes
- Distributed Docker layer caching
- Custom docker image annotations
- Inline pipeline editor

Feature Comparison

- 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. (Codefresh will be ending their support for private docker registries as of May 1, 2020)

- 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. Review Apps support both static and dynamic URLs.

- 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.

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

- Group-level Docker registry browser
- A single UI view into images across multiple repositories.

- Helm chart repository support
- Supports storage and retrieval of Helm charts.

- 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

Scheduled triggering of pipelines

<table>
<thead>
<tr>
<th></th>
<th>Core</th>
<th>Starter</th>
<th>Premium</th>
<th>Ultimate</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Free</td>
<td>Bronze</td>
<td>Silver</td>
<td>Gold</td>
</tr>
</tbody>
</table>

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

Learn how to trigger pipelines on a schedule in GitLab

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

<table>
<thead>
<tr>
<th></th>
<th>Core</th>
<th>Starter</th>
<th>Premium</th>
<th>Ultimate</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Free</td>
<td>Bronze</td>
<td>Silver</td>
<td>Gold</td>
</tr>
</tbody>
</table>

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 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

<table>
<thead>
<tr>
<th></th>
<th>Core</th>
<th>Starter</th>
<th>Premium</th>
<th>Ultimate</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Free</td>
<td>Bronze</td>
<td>Silver</td>
<td>Gold</td>
</tr>
</tbody>
</table>

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

**GitLab Kubernetes Agent**

Manage the deployments and connection to your Kubernetes clusters in a secure and compliant way, driven by code.

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 applications for Kubernetes clusters**

Install Helm, Tiller, NGINX Ingress, Cert-Manager, Prometheus, GitLab Runner, JupyterHub, and Kreative directly into your cluster from the GitLab Web UI with one click.

Read more on the issue

**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 to provide security by default.

Learn more about Static Application Security Testing

**Secret Detection**

GitLab allows you to perform Secret Detection in CI/CD pipelines; checking for unintentionally committed secrets and credentials. Results are then shown in the Merge Request and in the Pipeline view. This feature is available as part of Auto DevOps to provide security by default.
Learn more about Secret Detection

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 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 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 that 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 to provide security-by-default.

Learn more about application security for containers

Interactive Application Security Testing

IAST 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.

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

Automated Accessibility scanning of Review Apps

Performing accessibility testing is important in order to ensure you’re serving all the users who use your products. In GitLab, you can generate Accessibility reports automatically prior to merging into master.

Learn more about Automated Accessibility scanning

Leverage virtual package registries to simplify package management workflows.

A virtual registry is a collection of local, remote and other virtual registries accessed through a single logical URL. GitLab Epic detailing the issues required to add this functionality.

Forward requests for packages not found in GitLab to npmjs.com

By default, when an NPM package is not found in the GitLab NPM Registry, the request is forwarded to npmjs.com

Check out the docs to learn more.
### Conan Repository
Conan is an open-source, decentralized, and multi-platform C/C++ Package Manager for developers to create and share native binaries. [Documentation on the Conan Repository](#)

### Maven Repository
GitLab's Maven repository makes it easier to publish and share Java libraries across an organization, and ensure dependencies are managed correctly. It is fully integrated with GitLab, including authentication and authorization. [Documentation on the Maven Repository](#)

### NPM Repository
GitLab's NPM repository makes it easier to publish and share NPM packages across an organization, and ensure dependencies are managed correctly. It is fully integrated with GitLab, including authentication and authorization. [Documentation on the NPM Registry](#)

### NuGet Repository
GitLab's NuGet Repository allows C#/.NET developers to create, publish and share packages using the NuGet client or Visual Studio. [Documentation on the NuGet Repository](#)

### PyPI Repository
Python developers can set up GitLab as a remote PyPI repository and build, publish, and share packages using the PyPI client or GitLab CI/CD. [Documentation for the PyPI Repository](#)

### RPM Repository
This planned feature will enable Linux developers to build, publish and share RPM packages alongside their source code and pipelines. [Check out the issue for additional details on implementation and timing](#)

### Debian Repository
This planned feature will enable Linux developers to build, publish and share Debian packages alongside their source code and pipelines. [Check out the issue for additional details on implementation and timing](#)

### RubyGems Repository
This planned feature will enable Ruby developers to setup GitLab as a remote RubyGems repository and to build, publish and share packages using the command line or GitLab CI/CD. This will also be a valuable feature for GitLab and help with dogfooding. [Check out the issue for additional details on implementation and timing](#)

### Go Proxy
This planned feature will enable Go developers to publish and share their packages right alongside their source code and pipelines. This will also be a valuable feature for GitLab and help with dogfooding. [Read the Go Proxy docs](#)

### Composer Repository
This planned feature will enable PHP developers to build, publish and share their packages right
alongside their source code and pipelines.

**Check out the docs**

### Use the Package Registry through REST API

<table>
<thead>
<tr>
<th>Plan</th>
<th>Free</th>
<th>Starter</th>
<th>Premium</th>
<th>Ultimate</th>
</tr>
</thead>
</table>

Enables support for automation and integration of the Gitlab Package Registry through a REST API.

**Documentation on API**

### Explicit support for monorepos

<table>
<thead>
<tr>
<th>Plan</th>
<th>Free</th>
<th>Starter</th>
<th>Premium</th>
<th>Ultimate</th>
</tr>
</thead>
</table>

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

### Global Docker registry browser

<table>
<thead>
<tr>
<th>Plan</th>
<th>Free</th>
<th>Starter</th>
<th>Premium</th>
<th>Ultimate</th>
</tr>
</thead>
</table>

A single UI view into images across multiple repositories.

Read more on the issue

### Group-level Kubernetes cluster configuration

<table>
<thead>
<tr>
<th>Plan</th>
<th>Free</th>
<th>Starter</th>
<th>Premium</th>
<th>Ultimate</th>
</tr>
</thead>
</table>

A single UI view into Kubernetes cluster configuration for groups.

Read more on the issue

### Instance-level Kubernetes cluster configuration

<table>
<thead>
<tr>
<th>Plan</th>
<th>Free</th>
<th>Starter</th>
<th>Premium</th>
<th>Ultimate</th>
</tr>
</thead>
</table>

A single UI view into Kubernetes cluster configuration for a GitLab instance.

Read more on the issue

### First class container building

<table>
<thead>
<tr>
<th>Plan</th>
<th>Free</th>
<th>Starter</th>
<th>Premium</th>
<th>Ultimate</th>
</tr>
</thead>
</table>

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

### Group-level Kubernetes clusters

<table>
<thead>
<tr>
<th>Plan</th>
<th>Free</th>
<th>Starter</th>
<th>Premium</th>
<th>Ultimate</th>
</tr>
</thead>
</table>

Create group-level Kubernetes cluster that can be used for all projects contained within the group or sub-groups.

Learn more about group-level clusters

### Instance-level Kubernetes clusters

<table>
<thead>
<tr>
<th>Plan</th>
<th>Free</th>
<th>Starter</th>
<th>Premium</th>
<th>Ultimate</th>
</tr>
</thead>
</table>

Create instance-level Kubernetes clusters that can be used for all groups and projects contained within the same GitLab instance.

Learn more about instance-level clusters

### On-demand Dynamic Application Security Testing

<table>
<thead>
<tr>
<th>Plan</th>
<th>Free</th>
<th>Starter</th>
<th>Premium</th>
<th>Ultimate</th>
</tr>
</thead>
</table>

“There’s no reason to wait for the next CI pipeline run to find out if your site is vulnerable or to reproduce a previously found vulnerability. GitLab offers scanning your running application with On-demand Dynamic Application Security Testing (DAST), independent of code changes or merge requests.”

Learn more about On-demand DAST