



> [On this page](#)

# GitLab Runner ALL TIERS

GitLab Runner is an application that works with GitLab CI/CD to run jobs in a pipeline.

## Use GitLab.com runners

If you use GitLab.com, you can run your CI/CD jobs on runners hosted by GitLab. These runners are [managed](#) by GitLab and fully integrated with GitLab.com. By default these runners are enabled for all projects. You can [disable the runners](#) if you have the Owner role for the project.

Alternatively, you can install GitLab Runner and register your own runners on GitLab.com or on your own instance.


## Use self-managed runners ALL TIERS SELF-MANAGED

To use self-managed runners, you [install](#) GitLab Runner on infrastructure that you own or manage.


## Scale a fleet of runners

When your organization scales to having a fleet of runners, you should [plan for how you will monitor and adjust performance for these runners](#).

## GitLab Runner versions

For compatibility reasons, the GitLab Runner [major.minor](#)  version should stay in sync with the GitLab major and minor version. Older runners may still work with newer GitLab versions, and vice versa. However, features may not be available or work properly if a version difference exists.

Backward compatibility is guaranteed between minor version updates. However, sometimes minor version updates of GitLab can introduce new features that require GitLab Runner to be on the same minor version.

 GitLab Runner 15.0 [introduced](#) a change to the registration API request format. It prevents the GitLab Runner from communicating with GitLab versions lower than 14.8. You must use a Runner version that is appropriate for the GitLab version, or upgrade the GitLab application.

If you host your own runners but host your repositories on GitLab.com, keep GitLab Runner [updated](#) to the latest version, as GitLab.com is [updated continuously](#).

## Runner registration

After you install the application, you [register](#) individual runners. Runners are the agents that run the CI/CD jobs that come from GitLab.

When you register a runner, you are setting up communication between your GitLab instance and the machine where GitLab Runner is installed.

Runners usually process jobs on the same machine where you installed GitLab Runner. However, you can also have a runner process jobs in a container, in a Kubernetes cluster, or in auto-scaled instances in the cloud.

## Executors

When you register a runner, you must choose an executor.

An [executor](#) determines the environment each job runs in.

For example:

- If you want your CI/CD job to run PowerShell commands, you might install GitLab Runner on a Windows server and then register a runner that uses the shell executor.
- If you want your CI/CD job to run commands in a custom Docker container, you might install GitLab Runner on a Linux server and register a runner that uses the Docker executor.

These are only a few of the possible configurations. You can install GitLab Runner on a virtual machine and have it use another virtual machine as an executor.

When you install GitLab Runner in a Docker container and choose the [Docker executor](#) to run your jobs, it's sometimes referred to as a "Docker-in-Docker" configuration.

## Who has access to runners in the GitLab UI

Before you register a runner, you should determine if everyone in GitLab should have access to it, or if you want to limit it to a specific GitLab group or project.

There are three types of runners, based on who you want to have access:

- [Shared runners](#) are for use by all projects
- [Group runners](#) are for all projects and subgroups in a group
- [Project runners](#) are for individual projects

The scope of a runner is defined during the registration. This is how the runner knows which projects it's available for.

## Tags

When you register a runner, you can add [tags](#) to it.

When a CI/CD job runs, it knows which runner to use by looking at the assigned tags. Tags are the only way to filter the list of available runners for a job.

For example, if a runner has the `ruby` tag, you would add this code to your project's `.gitlab-ci.yml` file:

```
job:
  tags:
    - ruby
```

When the job runs, it uses the runner with the `ruby` tag.

## Configuring runners

You can [configure](#) the runner by editing the `config.toml` file. This is a file that is installed during the runner installation process.

In this file you can edit settings for a specific runner, or for all runners.

You can specify settings like logging and cache. You can set concurrency, memory, CPU limits, and more.

## Monitoring runners

You can use Prometheus to [monitor](#) your runners. You can view things like the number of currently-running jobs and how much CPU your runners are using.

## Use a runner to run jobs

After a runner is configured and available for your project, your [CI/CD](#) jobs can use the runner.

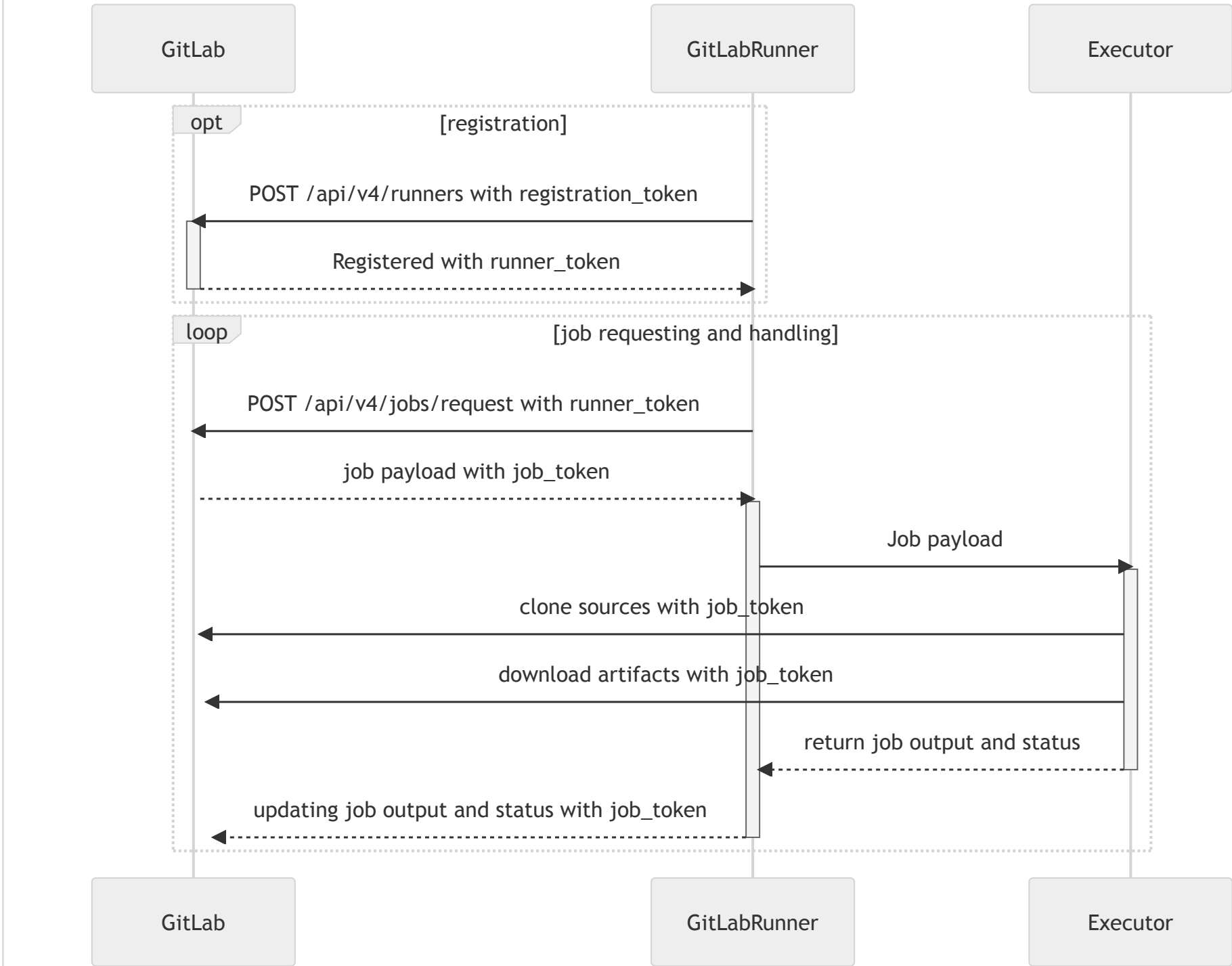
# Features

GitLab Runner has the following features.

- Run multiple jobs concurrently.
- Use multiple tokens with multiple servers (even per-project).
- Limit the number of concurrent jobs per-token.
- Jobs can be run:
  - Locally.
  - Using Docker containers.
  - Using Docker containers and executing job over SSH.
  - Using Docker containers with autoscaling on different clouds and virtualization hypervisors.
  - Connecting to a remote SSH server.
- Is written in Go and distributed as single binary without any other requirements.
- Supports Bash, PowerShell Core, and Windows PowerShell.
- Works on GNU/Linux, macOS, and Windows (pretty much anywhere you can run Docker).
- Allows customization of the job running environment.
- Automatic configuration reload without restart.
- Easy to use setup with support for Docker, Docker-SSH, Parallels, or SSH running environments.
- Enables caching of Docker containers.
- Easy installation as a service for GNU/Linux, macOS, and Windows.
- Embedded Prometheus metrics HTTP server.
- Referee workers to monitor and pass Prometheus metrics and other job-specific data to GitLab.

## Runner execution flow

This diagram shows how runners are registered and how jobs are requested and handled. It also shows which actions use [registration](#), [authentication](#), and [job tokens](#).



## Glossary

This glossary provides definitions for terms related to GitLab Runner.

- **GitLab Runner:** The application that you install that executes GitLab CI jobs on a target computing platform.
- **runner:** The agent that runs the code on the host platform and displays in the UI. If a runner is registered with the same token, the runner could represent a collection of runners and runner managers.
- **runner manager:** A type of runner that can create multiple runners for autoscaling. Specific to the type of executor used.
- **runner worker:** The process created by the runner on the host computing platform to run jobs.

## Troubleshooting

Learn how to [troubleshoot](#) common issues.

## Contributing

Contributions are welcome. See [CONTRIBUTING.md](#) and the [development documentation](#) for details.

If you're a reviewer of GitLab Runner project, take a moment to read the [Reviewing GitLab Runner](#) document.

You can also review [the release process for the GitLab Runner project](#).

## Changelog

See the [CHANGELOG](#) to view recent changes.

# License

This code is distributed under the MIT license. View the [LICENSE](#) file.

 **Help & feedback**

