

Run GitLab Runner in a container

We can run github runners locally. In this lab, we will register gitlab-runner on `gitlab.com`.

NOTE: Make sure to open new terminal and connect with your remote VM before running docker commands below:

```
ssh ubuntu@YOUR_VM_DNS_NAME.courseware.io
```

Password: Will be provided by Instructor.

```
root@982e2c2fb78d:~# ssh root@gitlab-ansible-dev.courseware.io
root@gitlab-ansible-dev.courseware.io's password:
Welcome to Ubuntu 22.10 (GNU/Linux 5.19.0-23-generic x86_64)

* Documentation:  https://help.ubuntu.com
* Management:    https://landscape.canonical.com
* Support:       https://ubuntu.com/advantage

System information as of Mon Feb 13 23:49:59 UTC 2023

System load:  0.0703125      Users logged in:      1
Usage of /:   13.0% of 154.96GB IPv4 address for docker0: 172.17.0.1
Memory usage: 15%           IPv4 address for eth0:  143.244.152.105
Swap usage:   0%            IPv4 address for eth0:  10.10.0.5
Processes:    163           IPv4 address for eth1:  10.116.0.2

94 updates can be applied immediately.
68 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

Last login: Mon Feb 13 23:48:16 2023 from 172.17.0.2
```

In this lab, you can use a configuration container to mount your custom data volume.

Create the Docker volume:

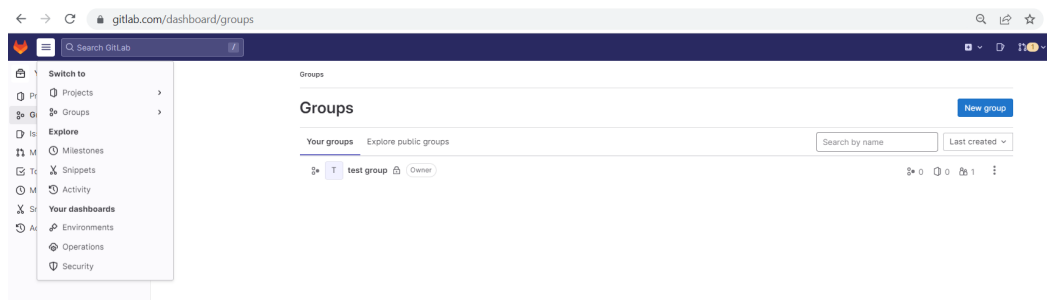
```
docker volume create gitlab-runner-config
```

Start the GitLab Runner container using the volume we just created:

```
docker run -d --name gitlab-runner --restart always \
-v /var/run/docker.sock:/var/run/docker.sock \
-v gitlab-runner-config:/etc/gitlab-runner \
gitlab/gitlab-runner:latest
```

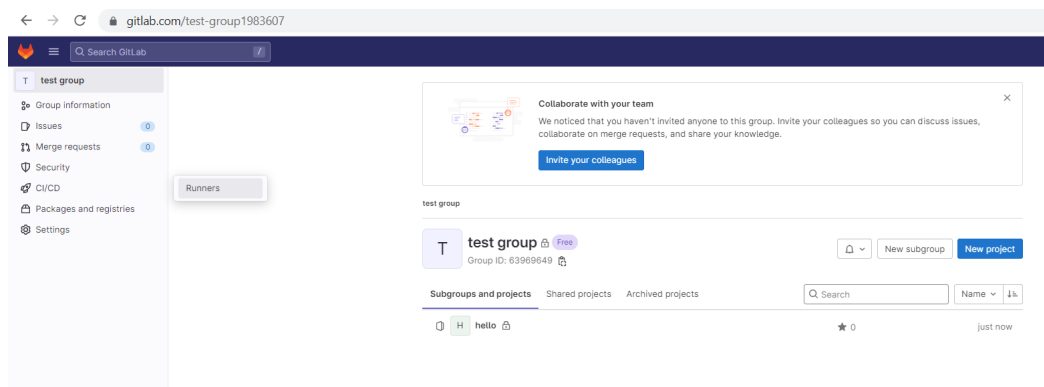
Group Runner Setup

1. On the top bar, select `Main menu` > `Groups` and find your group.

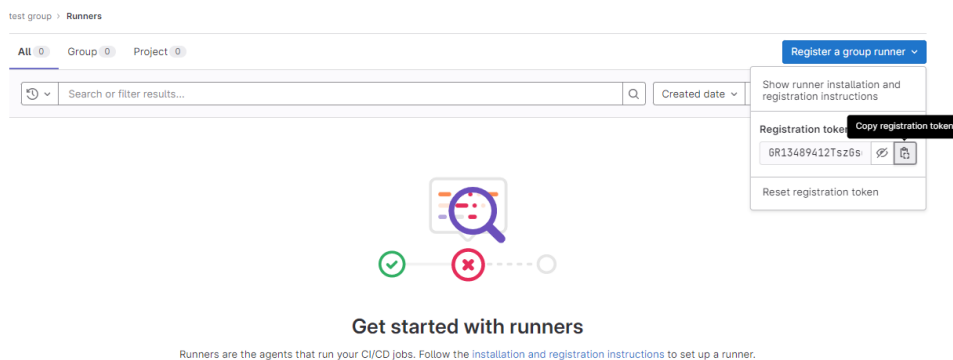


Note: If you don't have one, click `New Group` to create new one.

2. On the left sidebar, select `CI/CD` > `Runners` .



3. In the upper-right corner, select `Register a group runner` and copy **Registration token** from the next steps .



Register a Runner

1. To register a runner using a Docker container:

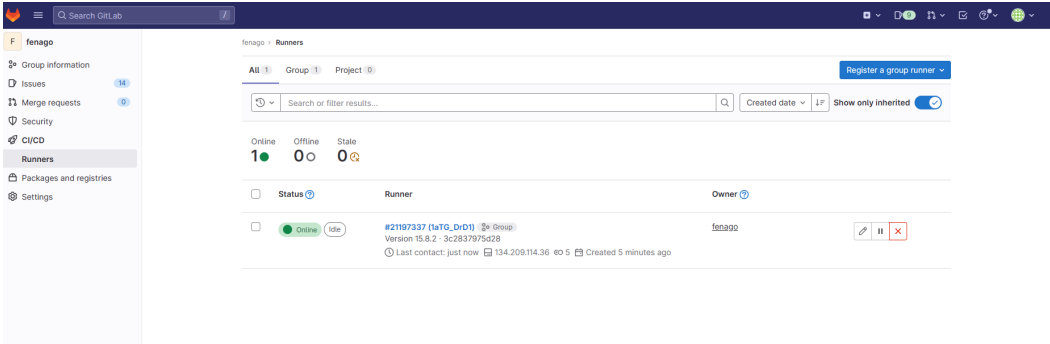
```
docker run --rm -it -v gitlab-runner-config:/etc/gitlab-runner gitlab/gitlab-runner:latest register
```

2. Enter your GitLab instance URL (also known as the gitlab-ci coordinator URL).
3. Enter the token you obtained to register the runner.
4. Enter a description for the runner. You can change this value later in the GitLab user interface.
5. Enter the tags associated with the runner, separated by commas. You can change this value later in the GitLab user interface.
6. Enter any optional maintenance note for the runner.
7. Provide the runner executor: enter **docker**.

8. If you entered docker as your executor, you are asked for the default image to be used for projects that do not define one in .gitlab-ci.yml: enter **ruby:2.7**

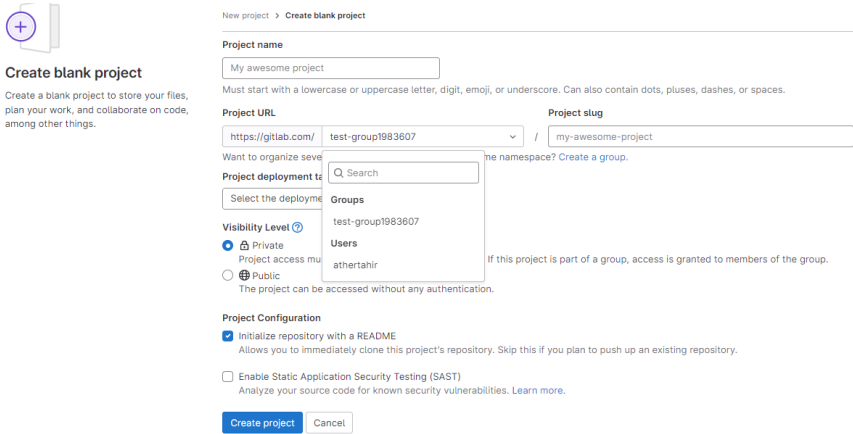
```
root@gitlabci-ansible22:~# docker run --rm -it --v gitlab-runner-config:/etc/gitlab-runner gitlab/gitlab-runner:latest register
Running platform: amd64 arch=amd64 os=linux pid=7 revision=4d1ca121 version=15.8.2
Enter the GitLab instance URL (for example, https://gitlab.com/):
https://gitlab.com/
Enter the registration token:
G13489412Gubw34C
Enter a description for the runner:
[3c28399/54d28]
Enter tags for the runner (comma-separated):
Enter optional maintenance note for the runner:
WARNING: Support for registration tokens and runner parameters in the 'register' command has been deprecated in GitLab Runner 15.6 and will be replaced with support for authentication tokens. For more information, see https://gitlab.com/gitlab-org/gitlab/-/issues/380872
Registering runner... succeeded runner=gitlabci-ansible22
Enter an executor: shell, ssh, virtualbox, docker-machine, kubernetes, docker, docker-ssh, docker-sshmachine, custom, parallels:
docker
Enter the default Docker image (for example, ruby:2.7):
ruby:2.7
runner registered successfully. Feel free to start it, but if it's running already the config should be automatically reloaded!
Configuration (with the authentication token) was saved in "/etc/gitlab-runner/config.toml"
root@gitlabci-ansible22:~#
```

9. Reload the webpage, you should see one runner available.



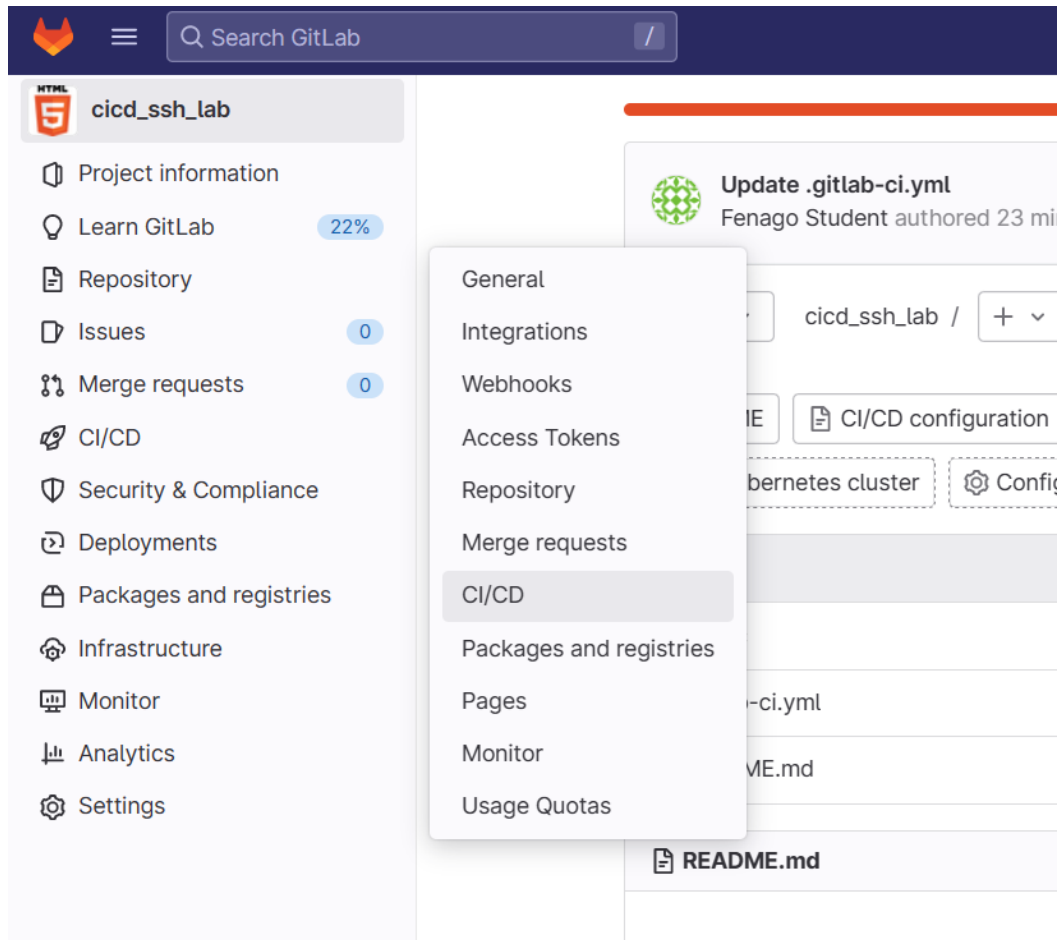
Using Gitlab Runner in CI/CD

1. **Important!** Make sure to create new project inside the group so that you can use the runner:



2. Disable shared runners for all the projects that you create. Otherwise, gitlab will fail your pipeline and ask for account verification.

3. On the left sidebar, select `Settings` > `CI/CD` .



4. Expand **Runners** and disable shared runners for this project.

Runners

Runners are processes that pick up and execute CI/CD jobs for GitLab. [What is GitLab Runner?](#)

Register as many runners as you want. You can register runners as separate users, on separate servers, and on your local machine.

How do runners pick up jobs?

Runners are either:


- **active** - Available to run jobs.
- **paused** - Not available to run jobs.

Tags control which type of jobs a runner can handle. By tagging a runner, you make sure shared runners only handle the jobs they are equipped to run. [Learn more.](#)

Project runners

These runners are assigned to this project.

Set up a project runner for a project

1. [Install GitLab Runner and ensure it's running.](#)
2. Register the runner with this URL:
`https://gitlab.com/` 

Shared runners

[These runners](#) are available to all groups and projects.

Each CI/CD job runs on a separate, isolated virtual machine.

Enable shared runners for this project



5. Scroll to **Group runners** and confirm one runner is available.


Group runners

These runners are shared across projects in this group.

Group runners can be managed with the [Runner API](#).

Disable group runners for this project

Available group runners: 1

 #21197337 (1aTG_DrD1)
3c2837975d28

Important! You will need to disable shared runner for all gitlab projects.