Lab 1: Advanced GitLab CI/CD Techniques

In this lab, we will create Gitlab account, run Gitlab runner instance locally and create and run GitLab CI/CD pipeline.

Task: Using SSH keys with GitLab private repositories

This task walks you through the steps of creating your own project, editing a file, and committing changes to a Git repository from the command line.

When you're done, you'll have a project where you can practice using Git.

What you need

Before you begin:

- Ensure you can sign in to an instance of GitLab. If your organization doesn't have GitLab, create an account on `https://gitlab.com/`.
- Create SSH keys and add them to GitLab. SSH keys are how you securely communicate between your computer and GitLab.

Create SSH key

SSH uses two keys, a public key and a private key.

- The public key can be distributed.
- The private key should be protected.

When you need to copy or upload your SSH public key, make sure you do not accidentally copy or upload your private key instead.

Let's create new ssh key using the VM. GitLab recommendation is to create SSH key type ED25519, which is more secure than RSA.

Generate an SSH key pair

If you do not have an existing SSH key pair, generate a new one:

1. Open a terminal.
2. Run `ssh-keygen -t` followed by the key type and an optional comment. This comment is included in the .pub file that's created. You may want to use an email address for the comment. To create new key run following command. Text after `-C` option is a comment and you can change it.

   ```
   ssh-keygen -t ed25519 -C "GitLab SSH key"
   ```

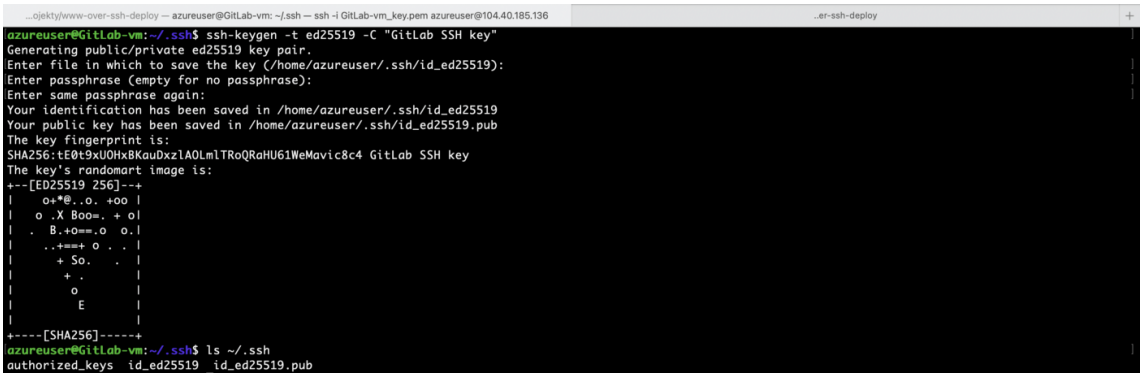
3. Press `Enter`. Output similar to the following is displayed:

   ```
   Generating public/private ed25519 key pair.
   Enter file in which to save the key (~/.ssh/id_ed25519):
   ```

4. Accept the suggested filename and directory.
5. Do not specify a passphrase:

   ```
   Enter passphrase (empty for no passphrase):
   Enter same passphrase again:
   ```

A confirmation is displayed, including information about where your files are stored.



The key will be created in default directory which for linux is `~/.ssh`. You should have two new files in `.ssh` directory:

- `id_ed25519` --- private key
- `id_ed25519.pub` --- public key

Add an SSH key to your GitLab account

To use SSH with GitLab, copy your public key to your GitLab account:

1. Copy the contents of your public key file.

```
cat ~/.ssh/id_ed25519.pub
```

2. Sign in to GitLab.
3. On the top bar, select your avatar.
4. Select **Preferences**.
5. On the left sidebar, select SSH Keys.
6. In the Key box, paste the contents of your public key. If you manually copied the key, make sure you copy the entire key, which starts with `ssh-ed25519` and may end with a comment.
7. In the **Title** box, type a description, like `Work Laptop` or `Home Workstation`.
8. Optional. Select the **Usage type** of the key. It can be used either for Authentication or Signing or both. Authentication & Signing is the default value.
9. Optional. **Update Expiration** date to modify the default expiration date.
10. Select **Add key**.

Verify that you can connect

Verify that your SSH key was added correctly.

Open a terminal and run this command, replacing gitlab.com with your GitLab instance URL:

```
ssh -T git@gitlab.com
```

If this is the first time you connect, you should verify the authenticity of the GitLab host. If you see a message like:

```
The authenticity of host 'gitlab.com (35.231.145.151)' can't be established.
ECDSA key fingerprint is SHA256:HbW3g8zUjNSksFbqTiUWPWg2Bq1x8xdGUrliXFzSnUw.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'gitlab.com' (ECDSA) to the list of known hosts.
```

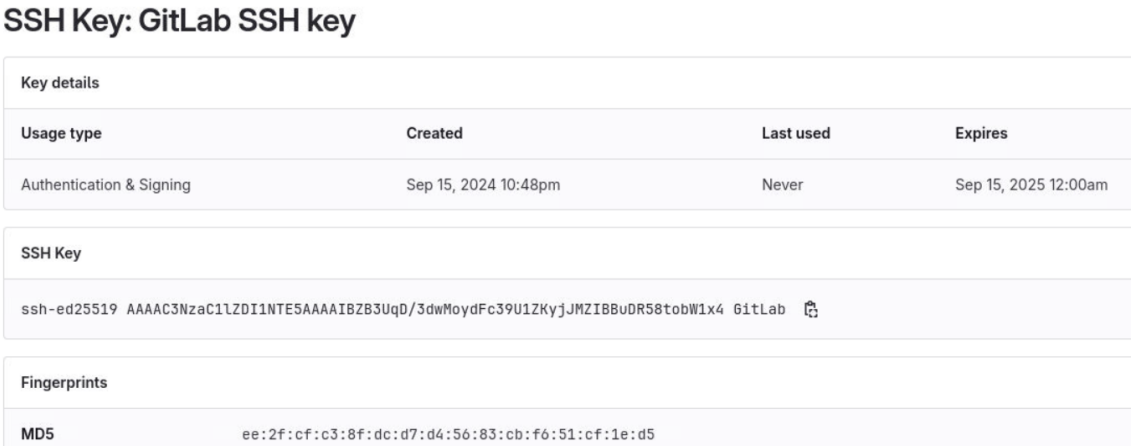
Type yes and press `Enter`.

Run the `ssh -T git@gitlab.com` command again. You should receive a Welcome to GitLab, @username! message.

View your account's SSH keys

1. Sign in to GitLab.
2. On the top bar, select your avatar.
3. Select Preferences.
4. On the left sidebar, select SSH Keys. Your existing SSH keys are listed at the bottom of the page. The information includes:

- The key's:
  - Name.
  - Public fingerprint.
  - Expiry date.
  - Permitted usage types.



Steps

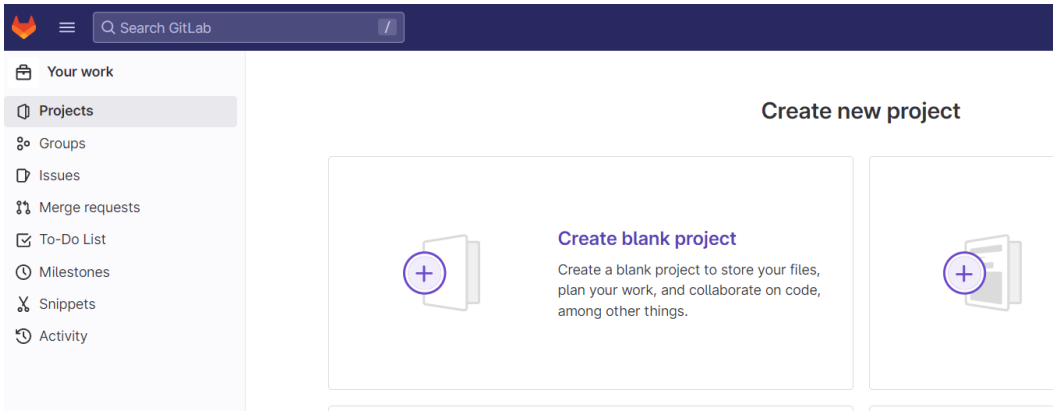
Here's an overview of what we're going to do:

1. Create a sample project.
2. Clone the repository.

Create a sample project

To start, create a sample project in GitLab.

1. In GitLab, on the top bar, select **Main menu > Projects > View all projects**.
2. On the right of the page, select **New project** > `Create blank project`.

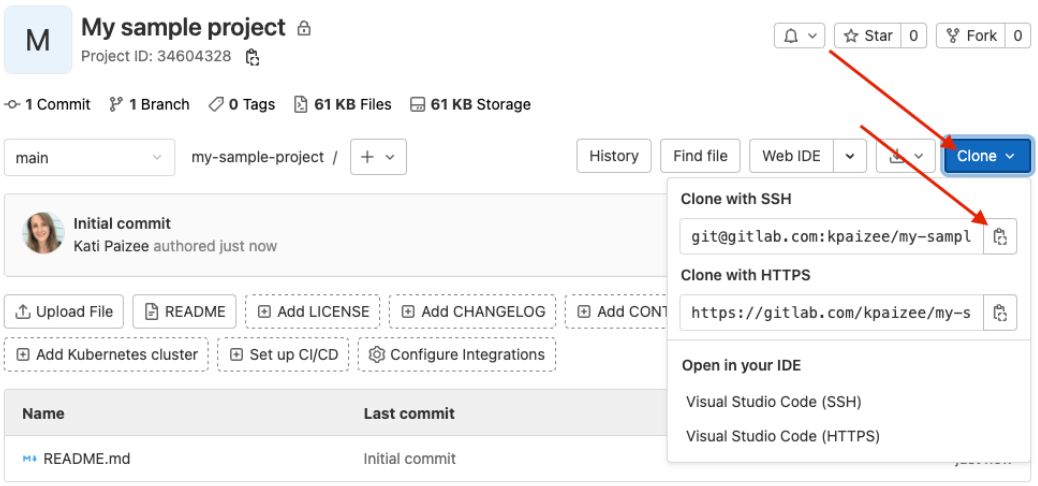


3. For **Project name**, enter `My sample project`. The project slug is generated for you. This slug is the URL you can use to access the project after it's created.
4. Ensure **Initialize repository with a README** is selected. How you complete the other fields is up to you.
5. Select **Create project**.

Clone the repository

Now you can clone the repository in your project. *Cloning* a repository means you're creating a copy on your computer, or wherever you want to store and work with the files.

1. On your project page, select **Clone**. Copy the URL for **Clone with SSH**.



2. Open a terminal on your computer and go to the directory where you want to clone the files.
3. Enter `git clone` and paste the URL:

```
git clone git@gitlab.com:YOUR_GITLAB_USERNAME/my-sample-project.git
```

4. Go to the directory:

```
cd my-sample-project
```

5. By default, you've cloned the default branch for the repository. Usually this branch is `main`. To be sure, get the name of the default branch:

   ```
   git branch
   ```

   The branch you're on is marked with an asterisk. Press `Q` on your keyboard to return to the main terminal window.

6. If the user's email and name are not yet configured in Git, follow these steps:

   - Replace the placeholders with your actual email address and name.
   - Run the following commands:

     ```
     git config --global user.email "you@example.com"
     git config --global user.name "Your Name"
     ```

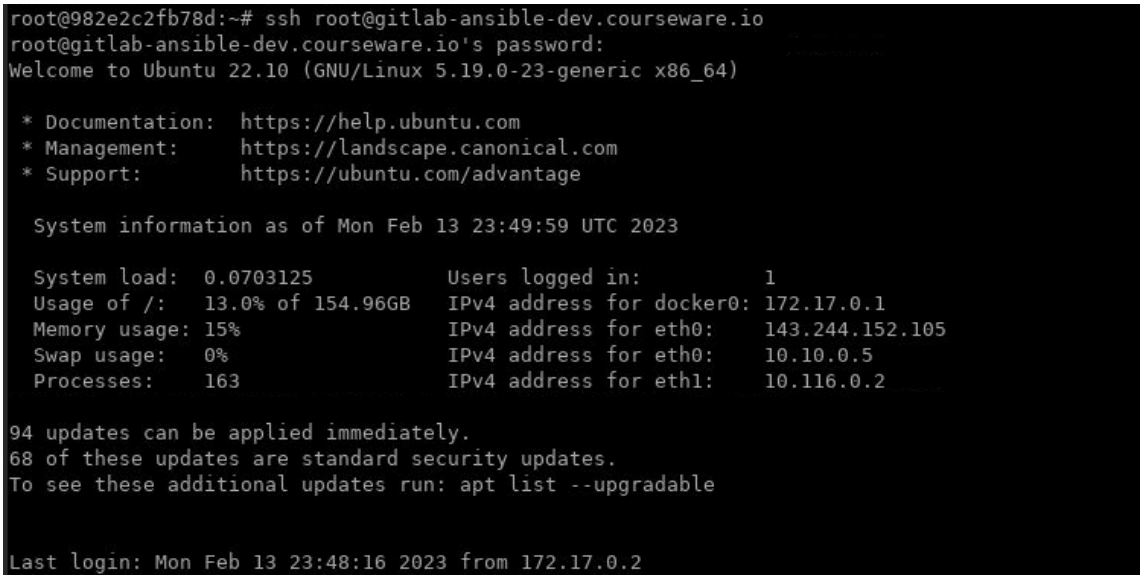
Run GitLab Runner in a container

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`

Password: Will be provided by Instructor.



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

Create the Docker volume:

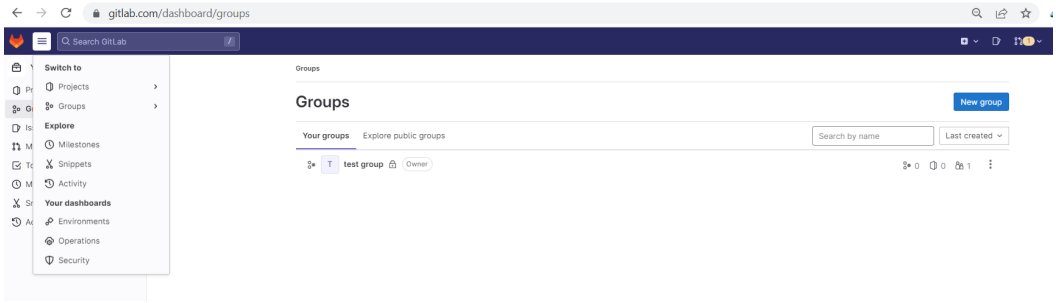
```
docker volume create gitlab-runner-config
```

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
```

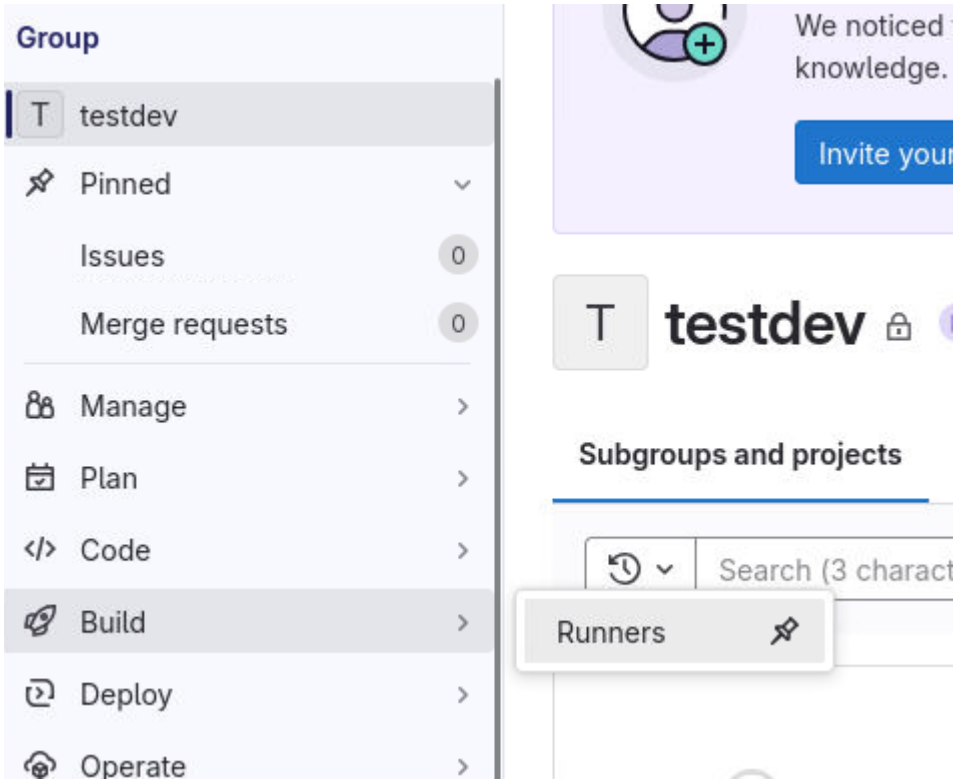
Group Runner Setup

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

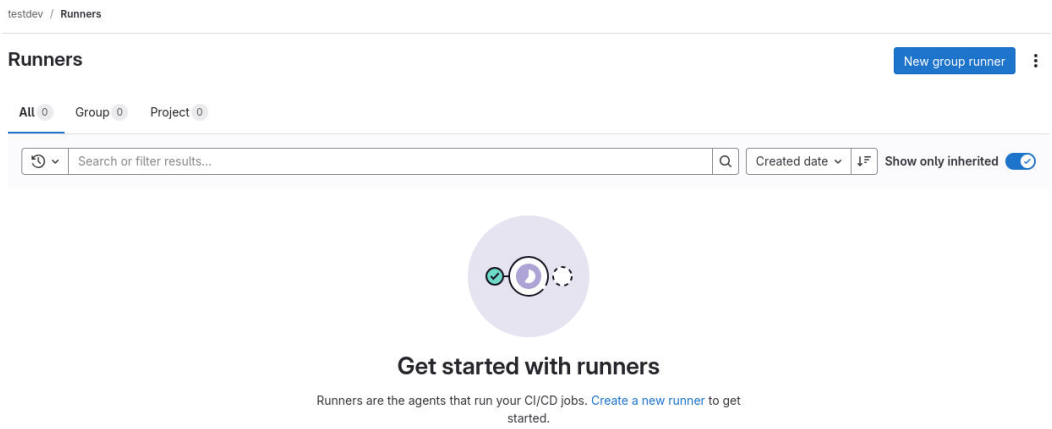


Note: If you don't have one, click `New Group` to create new one.

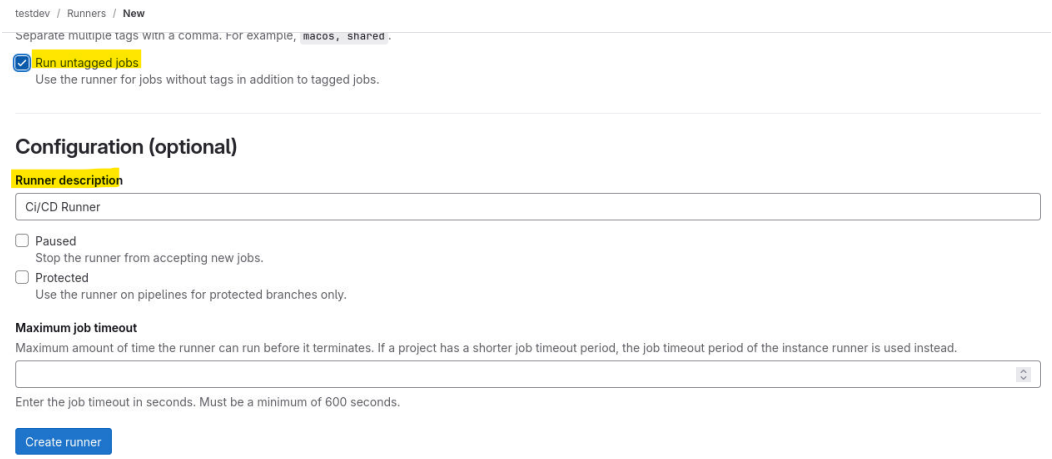
2. On the left sidebar, select `Build` > `Runners`.



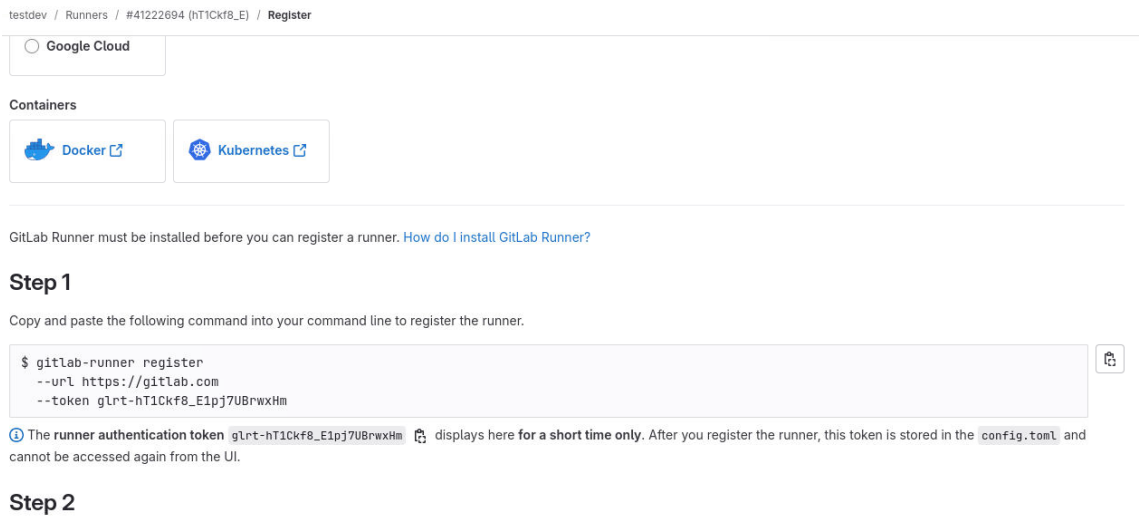
3. In the upper-right corner, select `New group runner`.



4. Check **Run untagged jobs**, add **Runner description** and create `New Runner`.



5. **Important:** Make sure to copy the token from the UI for next step (You will get different new token).

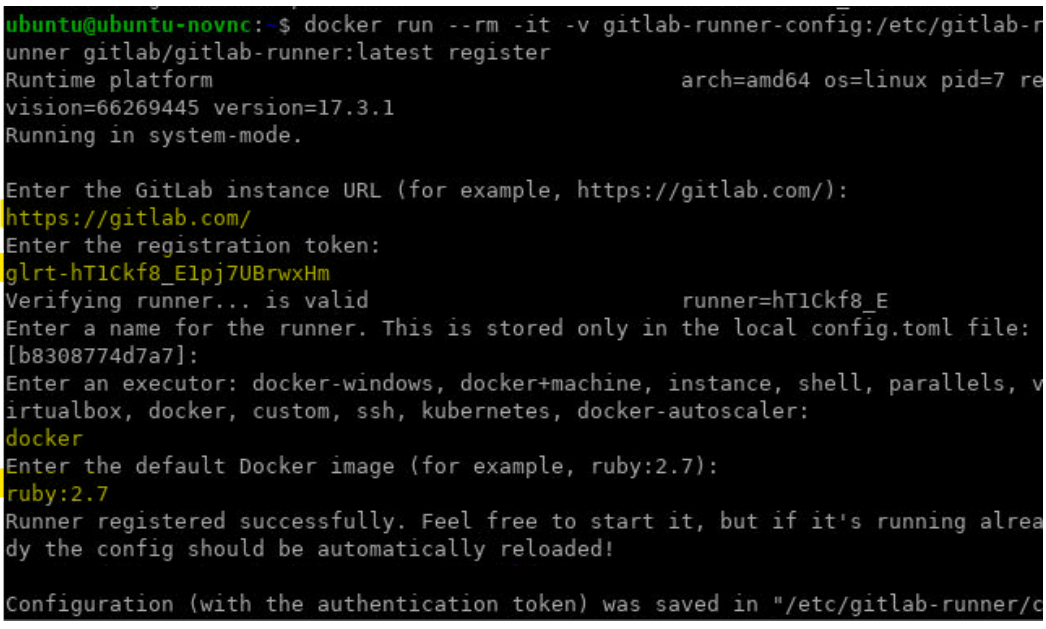


Register a Runner

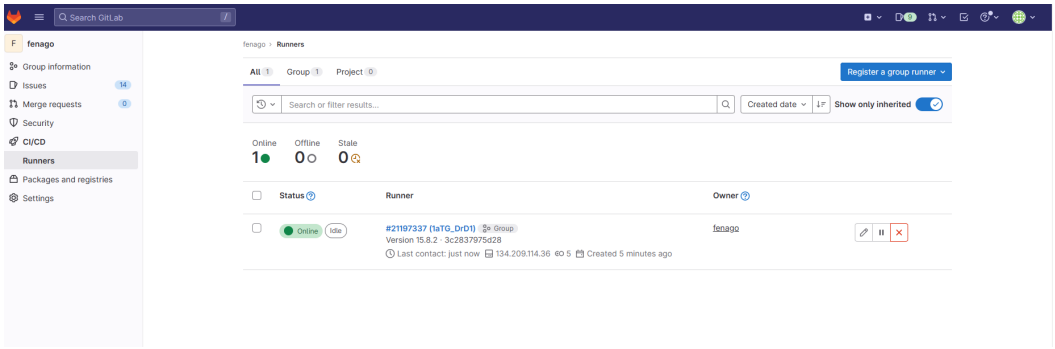
1. To register a runner using a Docker container:

```
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): `https://gitlab.com/`
3. Enter the token you obtained to register the runner.
4. Enter a name for the runner
5. Provide the runner executor: enter **docker**.
6. 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**

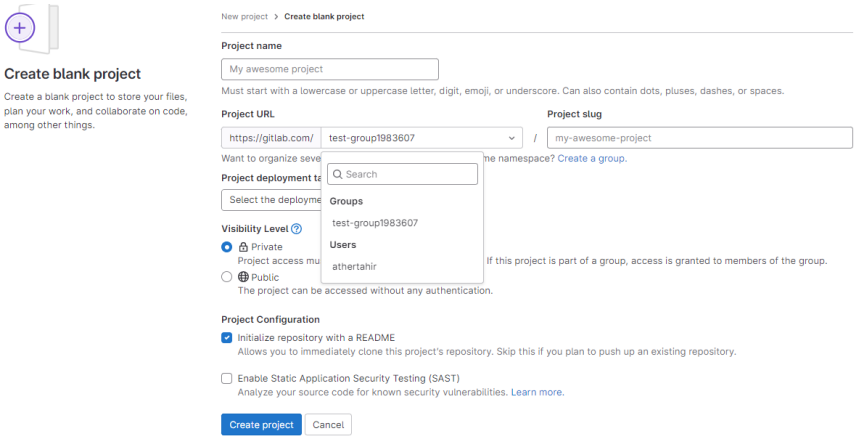


7. Open/Reload the runners webpage, you should see one runner available.



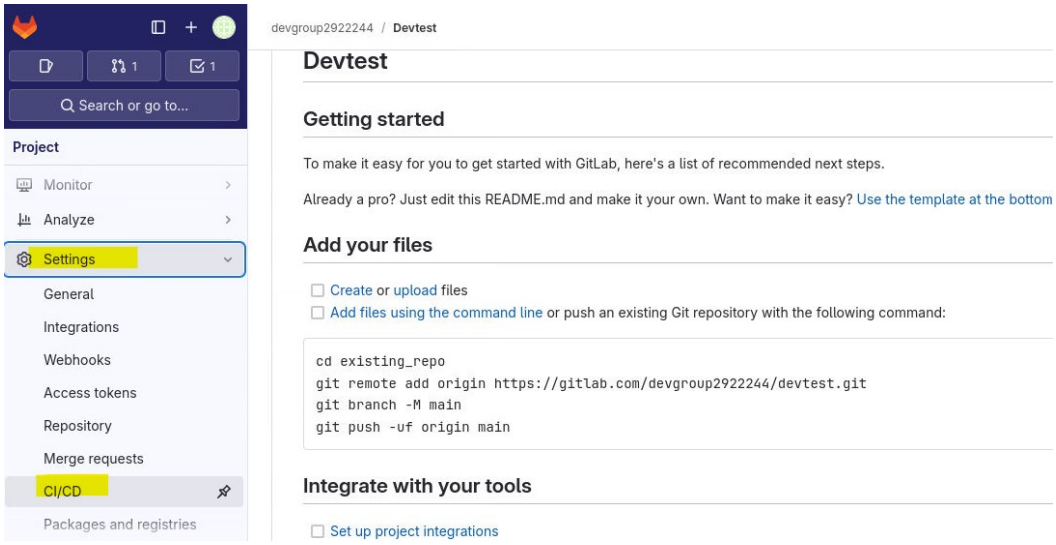
Using Gitlab Runner in CI/CD

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

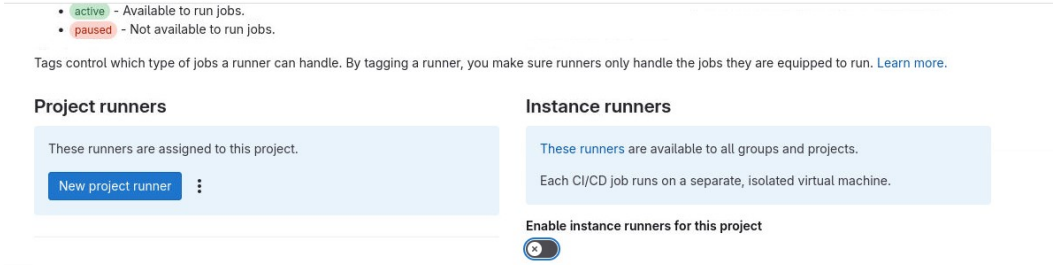


2. Disable instance runners for all the projects that you create. Otherwise, gitlab will fail your pipeline and ask for account verification.

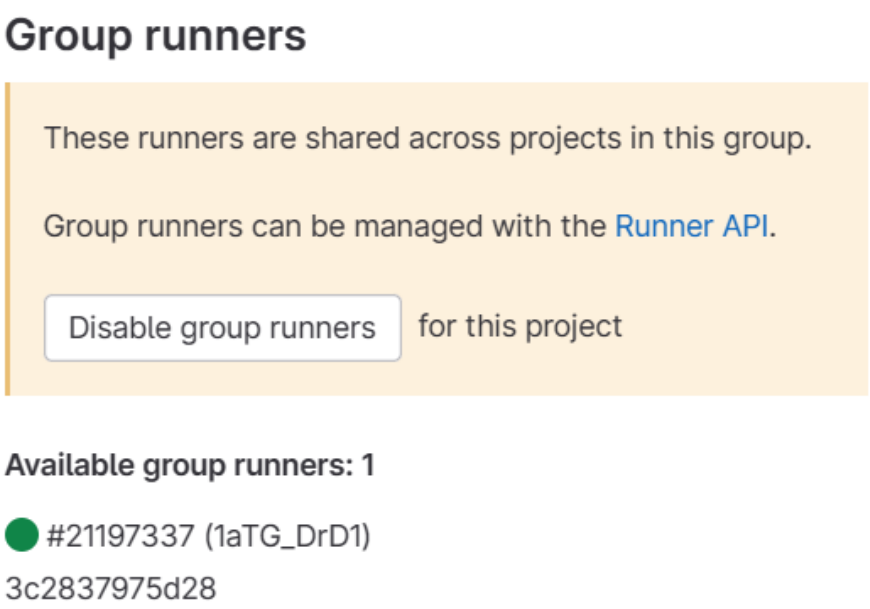
3. On the left sidebar, select `Settings` > `CI/CD`.



4. Expand `Runners` and disable instance runners for this project.



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



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

Task: Create and run GitLab CI/CD pipeline

This lab shows you how to configure and run your first CI/CD pipeline in GitLab.

Prerequisites

Before you start, make sure you have:

- A project in GitLab that you would like to use CI/CD for.
- The Maintainer or Owner role for the project.

If you don't have a project, you can create a public project for free on https://gitlab.com.

Steps

To create and run your first pipeline:

1. Ensure you have runners available to run your jobs.

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

2. Create a `.gitlab-ci.yml` file at the root of your repository. This file is where you define the CI/CD jobs.

When you commit the file to your repository, the runner runs your jobs. The job results are displayed in a pipeline.

Ensure you have runners available

In GitLab, runners are agents that run your CI/CD jobs.

To view available runners:

- Go to **Settings > CI/CD** and expand **Runners**.

As long as you have at least one runner that's active, with a green circle next to it, you have a runner available to process your jobs.

When your CI/CD jobs run, in a later step, they will run on your local machine.

Create a `.gitlab-ci.yml` file

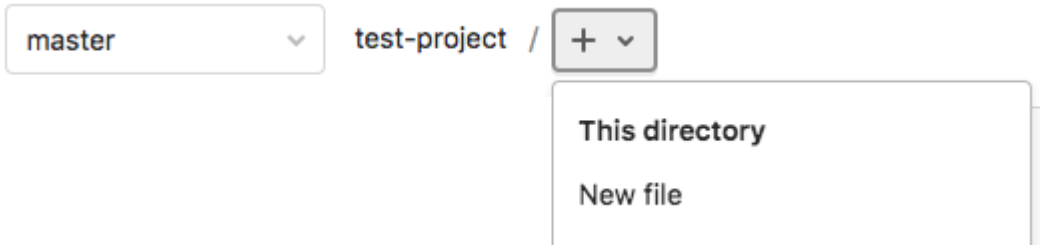
Now create a `.gitlab-ci.yml` file. It is a **YAML** file where you specify instructions for GitLab CI/CD.

In this file, you define:

- The structure and order of jobs that the runner should execute.
- The decisions the runner should make when specific conditions are encountered.

To create a `.gitlab-ci.yml` file:

1. On the left sidebar, select **Repository > Files**.
2. Above the file list, select the branch you want to commit to. If you're not sure, leave `master` or `main`. Then select the plus icon and **New file**:



3. For the **Filename**, type `.gitlab-ci.yml` and in the larger window, paste this sample code:

```
build-job:
  stage: build
  script:
    - echo "Hello, $GITLAB_USER_LOGIN!"

test-job1:
  stage: test
  script:
    - echo "This job tests something"

test-job2:
  stage: test
  script:
    - echo "This job tests something, but takes more time than test-job1."
    - echo "After the echo commands complete, it runs the sleep command for 20
seconds"
    - echo "which simulates a test that runs 20 seconds longer than test-job1"
```

```
    - sleep 20

deploy-prod:
  stage: deploy
  script:
    - echo "This job deploys something from the $CI_COMMIT_BRANCH branch."
  environment: production
```

This example shows four jobs: `build-job`, `test-job1`, `test-job2`, and `deploy-prod`. The comments listed in the `echo` commands are displayed in the UI when you view the jobs. The values for the [predefined variables] `$GITLAB_USER_LOGIN` and `$CI_COMMIT_BRANCH` are populated when the jobs run.

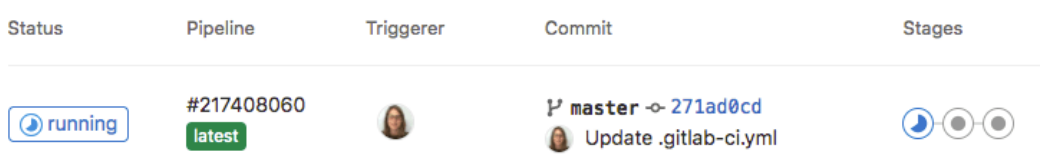
4. Select **Commit changes**.

The pipeline starts and runs the jobs you defined in the `.gitlab-ci.yml` file.

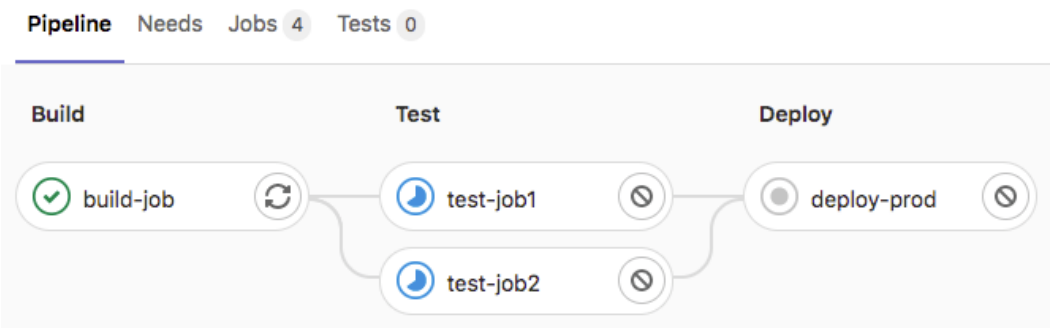
View the status of your pipeline and jobs

Now take a look at your pipeline and the jobs within.

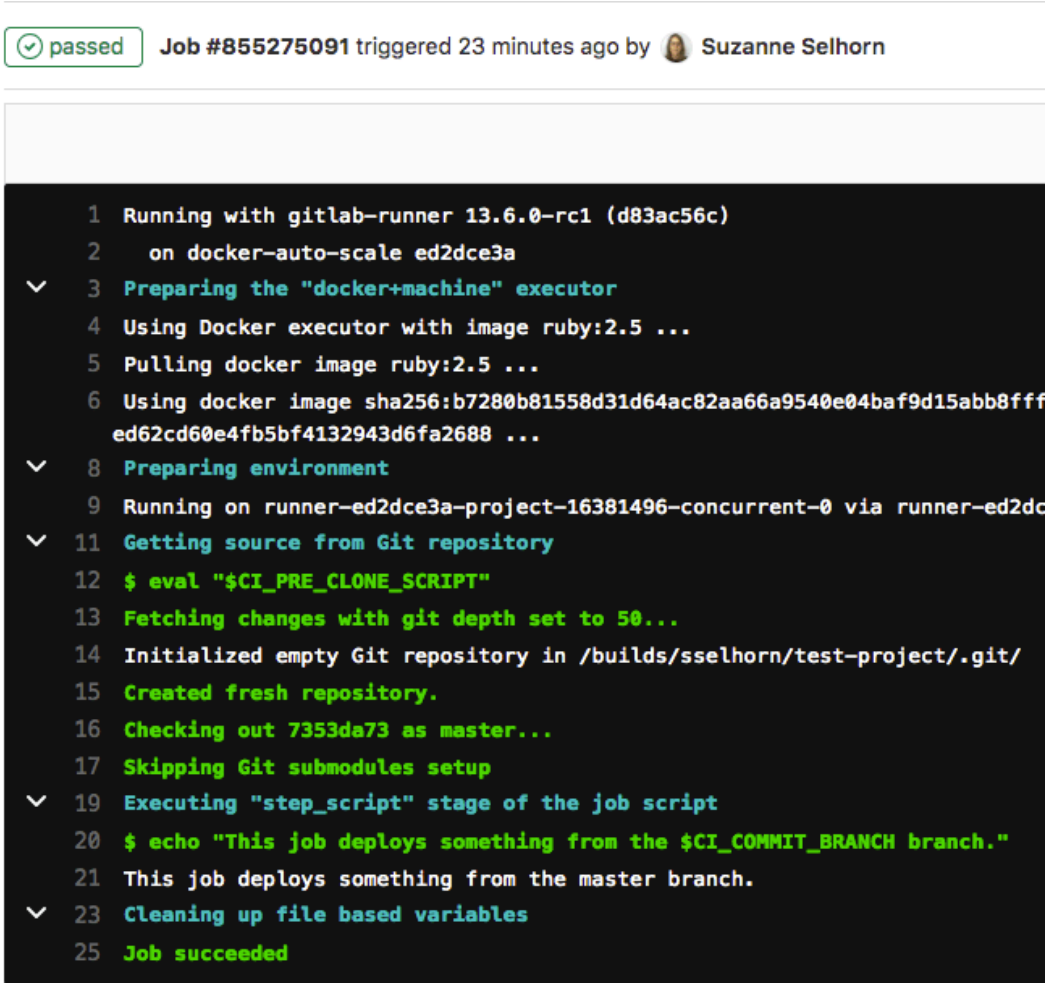
1. Go to **Build > Pipelines**. A pipeline with three stages should be displayed:



2. View a visual representation of your pipeline by selecting the pipeline ID:



3. View details of a job by selecting the job name. For example, `deploy-prod`:



You have successfully created your first CI/CD pipeline in GitLab. Congratulations!

Disabling GitLab CI/CD

GitLab CI/CD is enabled by default on all new projects. If you use an external CI/CD server like Jenkins or Drone CI, you can disable GitLab CI/CD to avoid conflicts with the commits status API.

Disable CI/CD in a project

When you disable GitLab CI/CD:

- The **CI/CD** item in the left sidebar is removed.
- The `/pipelines` and `/jobs` pages are no longer available.
- Existing jobs and pipelines are hidden, not removed.

To disable GitLab CI/CD in your project:

1. On the top bar, select **Main menu > Projects** and find your project.
2. On the left sidebar, select **Settings > General**.
3. Expand **Visibility, project features, permissions**.
4. In the **Repository** section, turn off **CI/CD**.
5. Select **Save changes**.

Enable CI/CD in a project

To enable GitLab CI/CD in your project:

1. On the top bar, select **Main menu > Projects** and find your project.
2. On the left sidebar, select **Settings > General**.
3. Expand **Visibility, project features, permissions**.
4. In the **Repository** section, turn on **CI/CD**.
5. Select **Save changes**.