

Tutorial: Create and run your first GitLab CI/CD pipeline

This tutorial 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.
If you're using GitLab.com, you can skip this step. GitLab.com provides shared runners for you.
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.

If you don't have a runner

If you don't have a runner:

1. [Install GitLab Runner](#) on your local machine.
2. [Register the runner](#) for your project. Choose the shell executor.

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

Create a .gitlab-ci.yml file

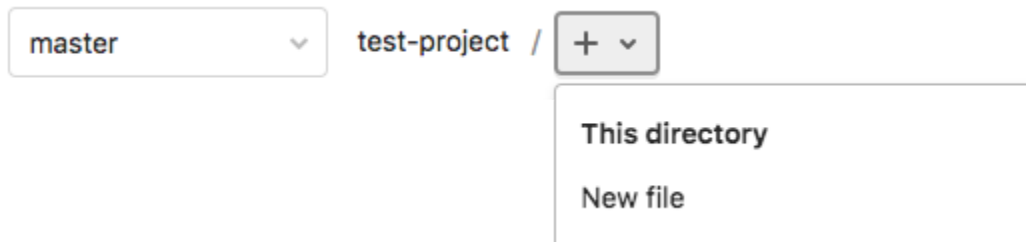
Now create a .gitlab-ci.yml file. It is a [YAML](#) file where you specify instructions for GitLab CI/CD.

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**:



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

3.





1. 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.
2. Select **Commit changes**.

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

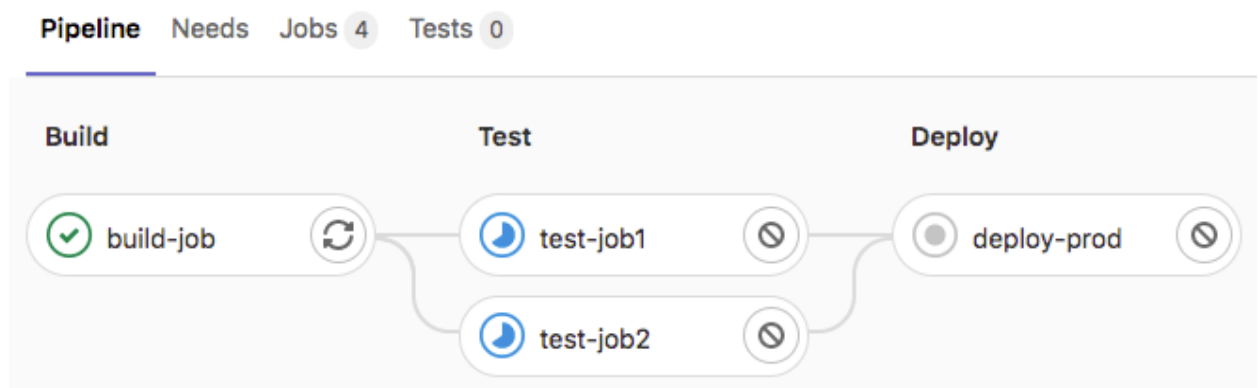
View the status of your pipeline and jobs

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


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

Status	Pipeline	Triggerer	Commit	Stages
 running	#217408060 latest		master → 271ad0cd  Update .gitlab-ci.yml	

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:

✓ passed Job #855275091 triggered 23 minutes ago by  Suzanne Selhorn

```
1 Running with gitlab-runner 13.6.0-rc1 (d83ac56c)
2   on docker-auto-scale ed2dce3a
3 ✓ Preparing the "docker+machine" executor
4   Using Docker executor with image ruby:2.5 ...
5   Pulling docker image ruby:2.5 ...
6   Using docker image sha256:b7280b81558d31d64ac82aa66a9540e04baf9d15abb8fff
   ed62cd60e4fb5bf4132943d6fa2688 ...
7 ✓ Preparing environment
8   Running on runner-ed2dce3a-project-16381496-concurrent-0 via runner-ed2dce3a
9
10 ✓ Getting source from Git repository
11   $ eval "$CI_PRE_CLONE_SCRIPT"
12   Fetching changes with git depth set to 50...
13   Initialized empty Git repository in /builds/sselhorn/test-project/.git/
14   Created fresh repository.
15   Checking out 7353da73 as master...
16   Skipping Git submodules setup
17
18 ✓ Executing "step_script" stage of the job script
19   $ echo "This job deploys something from the $CI_COMMIT_BRANCH branch."
20   This job deploys something from the master branch.
21
22 ✓ Cleaning up file based variables
23
24 Job succeeded
```

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

Now you can get started customizing your `.gitlab-ci.yml` and defining more advanced jobs.

`.gitlab-ci.yml` tips

Here are some tips to get started working with the `.gitlab-ci.yml` file.

For the complete `.gitlab-ci.yml` syntax, see [the full .gitlab-ci.yml keyword reference](#).

- Use the [pipeline editor](#) to edit your `.gitlab-ci.yml` file.
- Each job contains a script section and belongs to a stage:

- [stage](#) describes the sequential execution of jobs. If there are runners available, jobs in a single stage run in parallel.
 - Use the [needs keyword](#) to run jobs out of stage order. This creates a [Directed Acyclic Graph \(DAG\)](#).
- You can set additional configuration to customize how your jobs and stages perform:
 - Use the [rules](#) keyword to specify when to run or skip jobs. The only and except legacy keywords are still supported, but can't be used with rules in the same job.
 - Keep information across jobs and stages persistent in a pipeline with [cache](#) and [artifacts](#). These keywords are ways to store dependencies and job output, even when using ephemeral runners for each job.
 - Use the [default](#) keyword to specify additional configurations that are applied to all jobs. This keyword is often used to define [before_script](#) and [after_script](#) sections that should run on every job.