# Introduction to Livermore Computing GitLab

Neil J. O'Neill

June 14, 2023

LLNL-PRES-850102

Lawrence Livermore National Laboratory

## Livermore Computing (LC) GitLab Access & Authentication

- URLs
  - LC Collaboration Zone (CZ): https://lc.llnl.gov/gitlab
  - LC Restricted Zone (RZ): https://rzlc.llnl.gov/gitlab
  - LC SCF: https://lc.llnl.gov/gitlab

## LC GitLab Accounts

- If you have an account for any LC production machine, then you will have a corresponding account available on the GitLab instance in the same zone as that production machine.
- If you have accounts for both CZ and RZ production machines, then you will have GitLab accounts available on both the CZ and RZ GitLab instances.
- You need to login to the GitLab UI in order to activate your account. Your account needs this activation before you can use command line git commands (clone, push, pull, etc.).
- Your account will be automatically deactivated after 90 days of non-use. However, it can be reactivated simply by logging in to the GitLab UI. All your work will still be there – deactivation does not delete anything. Git commands don't count as use.

## Documentation

- LC specific docs: https://lc.llnl.gov/confluence/display/GITLAB/GitLab+CI
- General GitLab docs: https://docs.gitlab.com/
- Google search:
  - "gitlab pipeline variables"
- Issues at gitlab.com: https://gitlab.com/gitlab-org/gitlab/-/issues
  - Need to sign up for a free account to search
- Livermore Computing Compute Platforms (a.k.a. Production Machines)
  - https://hpc.llnl.gov/hardware/compute-platforms
- What computer accounts do I have?
  - "my info" portlet at MyLC
  - https://lc.llnl.gov/lorenz/mylc/mylc.cgi

## What is GitLab?

- “GitLab is a web-based platform that helps developers collaborate on large and complex projects using Git, a popular version control system.”
- GitLab Service as a Service (SaaS) at gitlab.com.  Very similar to GitHub.
- Gitlab self-hosted (what we have at LC)
  - LC has an “Ultimate” license for all its GitLab instances.

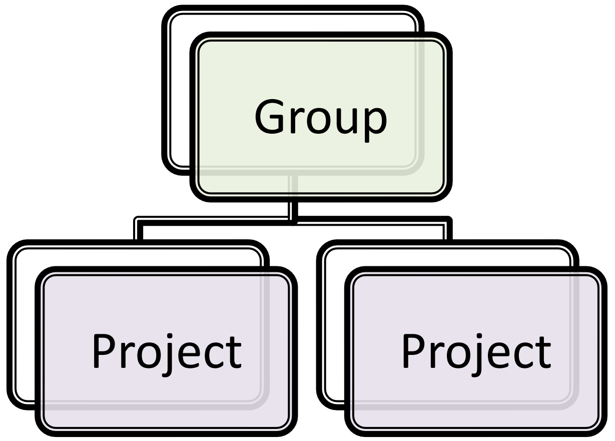
## GitLab Main Features

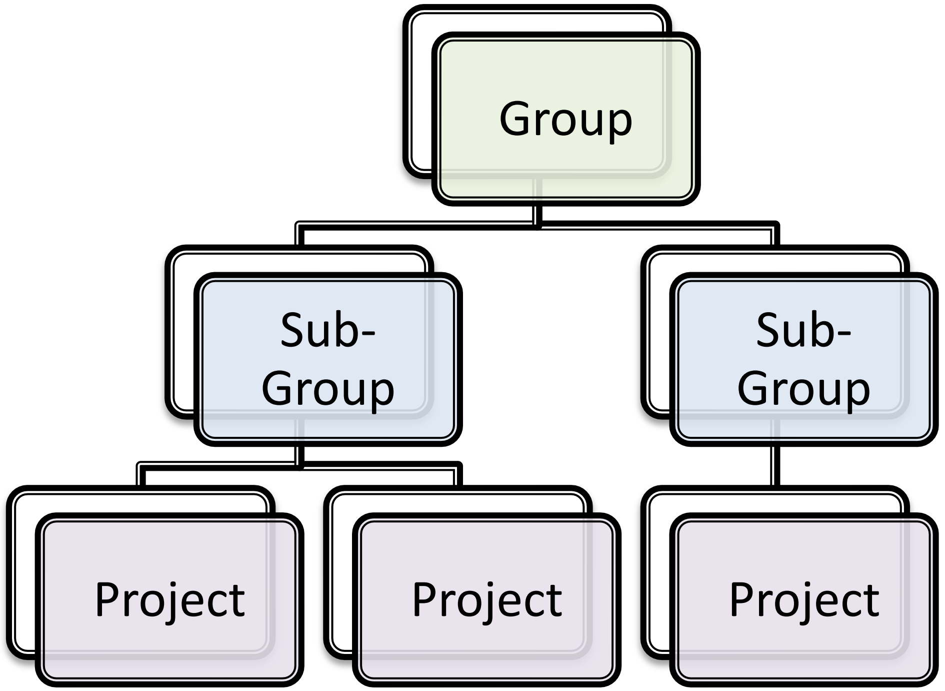
- Remote git repository
- Continuous Integration (CI) automation
- Web development environment (VS Code)
- Project organization and collaboration
  - Groups and sub-groups
  - Project membership with roles
- Code change auditing and control
  - Merge requests
  - Approver lists
  - Branch restrictions
- Issue tracking

## git

- A distributed version control system created by Linus Torvalds in 2005.
- Versions an entire repository as a whole rather than individual files or directories. Each version is called a “commit.”
- Used internally by GitLab for storing repositories.
- Installed on all LC production machines (“man git”).
- 
```
git clone ssh://git@czgitlab.llnl.gov:7999/my-awesome-group/my-awesome-project.git
```
- Primary documentation: [https://git-scm.com/doc](https://git-scm.com/doc)

## Groups and Projects

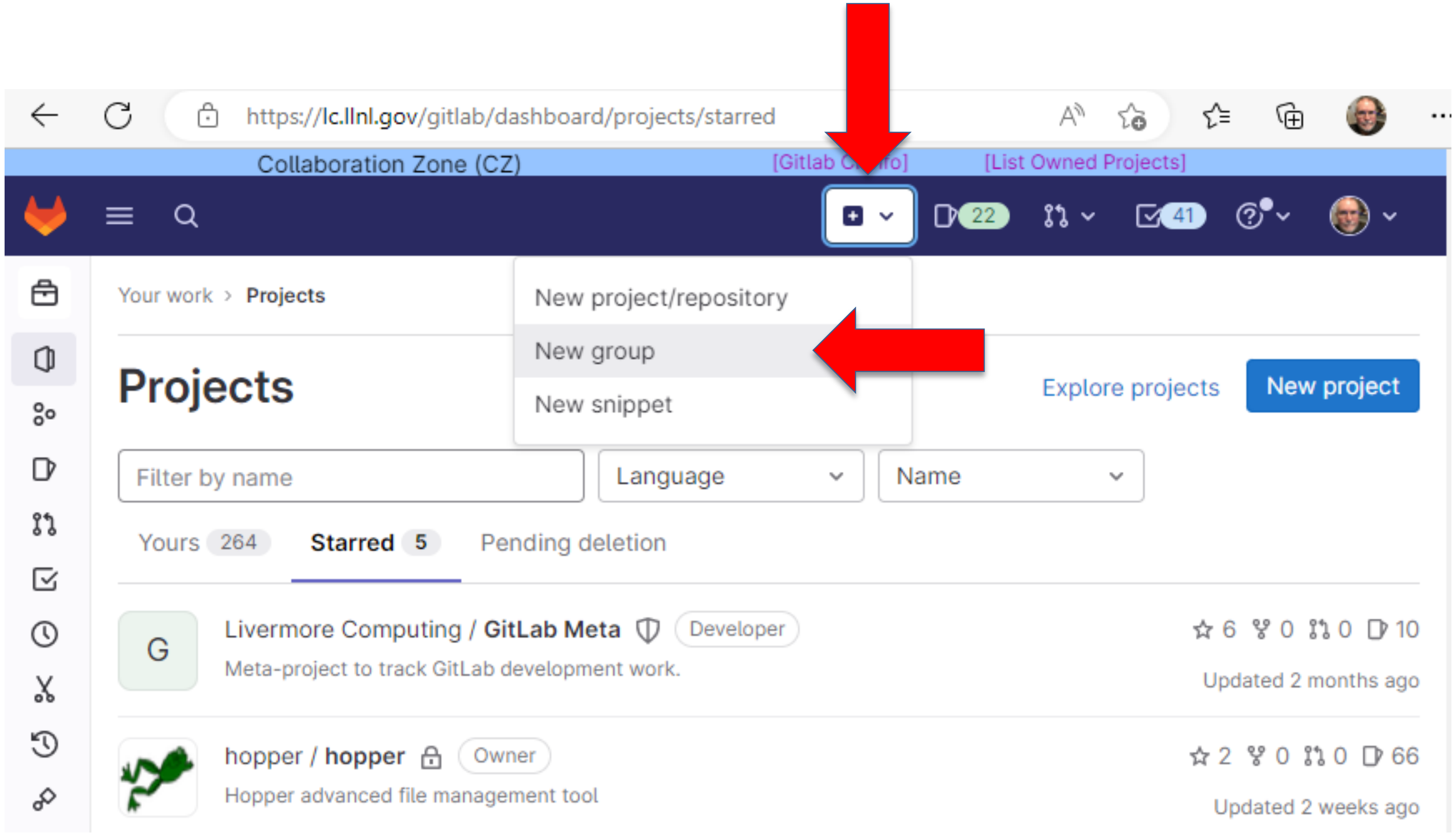




https://docs.gitlab.com/ee/user/project/
https://docs.gitlab.com/ee/user/group/

Note: each project contains only a single repository

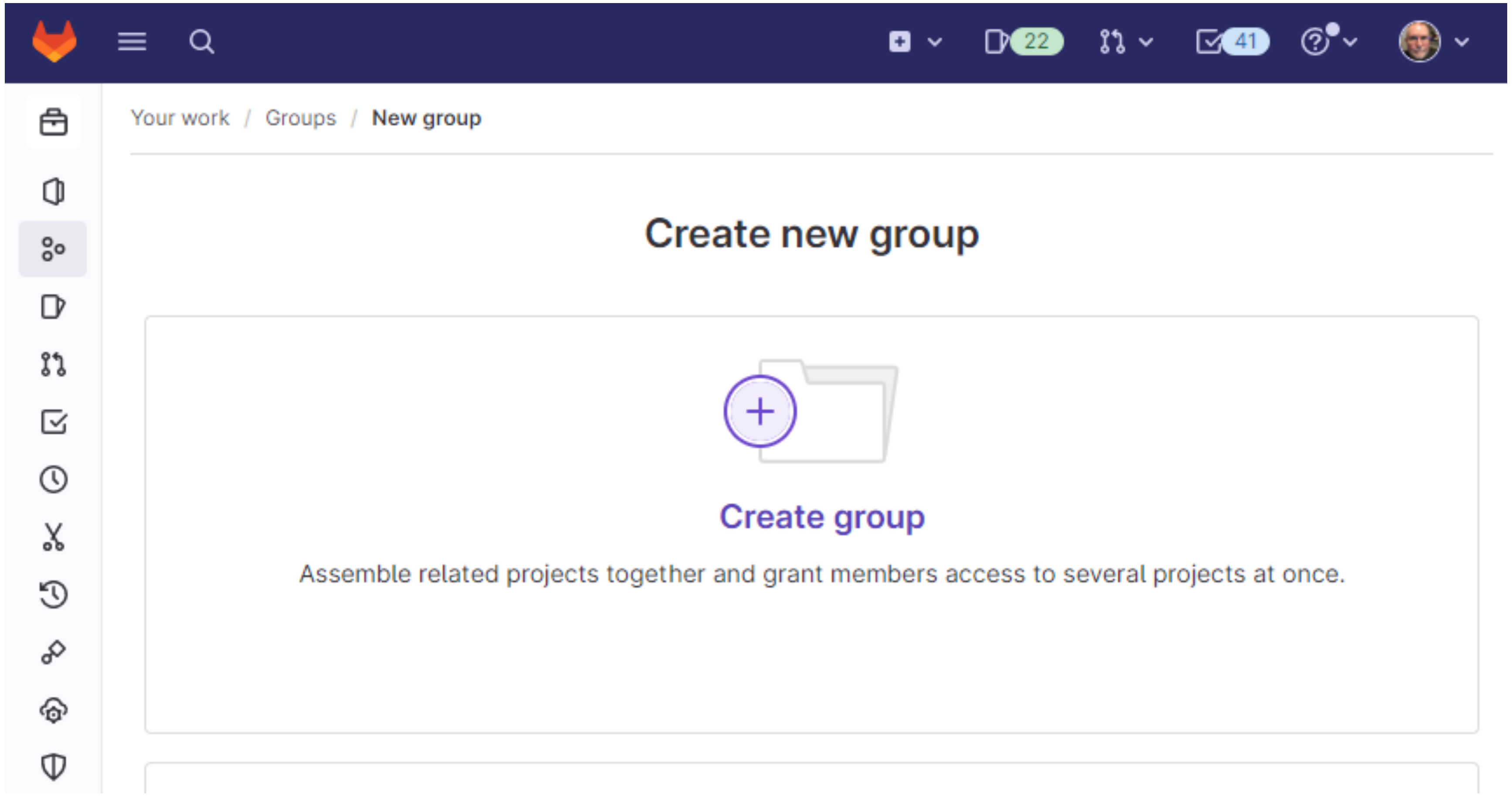
# Create Group Menu



Livermore Computing / GitLab Meta — Developer — ☆ 6, forks 0, merge requests 0, issues 10
Meta-project to track GitLab development work.
Updated 2 months ago

hopper / hopper — Owner — ☆ 2, forks 0, merge requests 0, issues 66
Hopper advanced file management tool
Updated 2 weeks ago

## Create Group

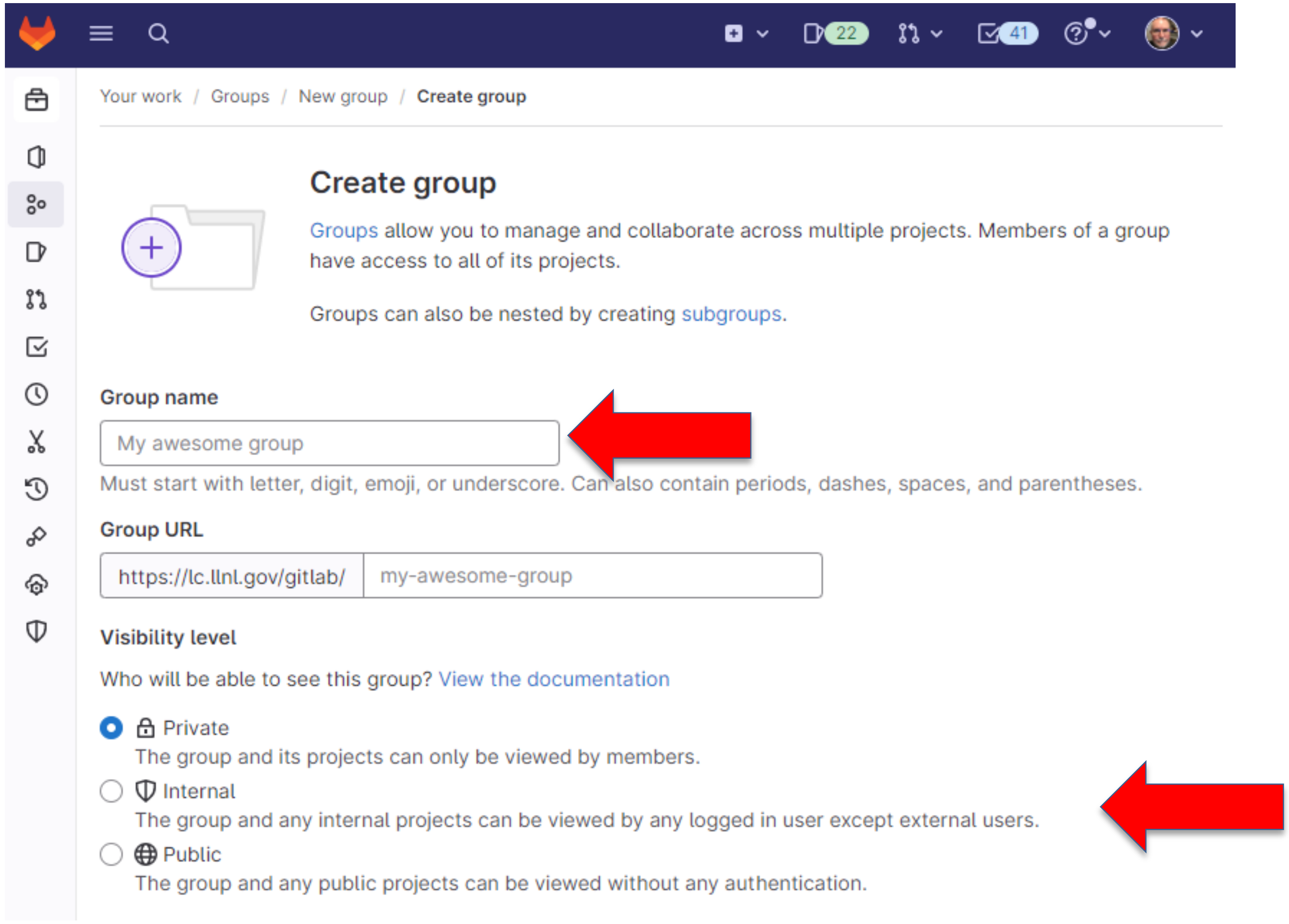


Create new group

Create group

Assemble related projects together and grant members access to several projects at once.

# Fill in Create Group form



Your work / Groups / New group / **Create group**

## Create group

Groups allow you to manage and collaborate across multiple projects. Members of a group have access to all of its projects.

Groups can also be nested by creating subgroups.

**Group name**

My awesome group

Must start with letter, digit, emoji, or underscore. Can also contain periods, dashes, spaces, and parentheses.

**Group URL**

https://lc.llnl.gov/gitlab/ my-awesome-group

**Visibility level**

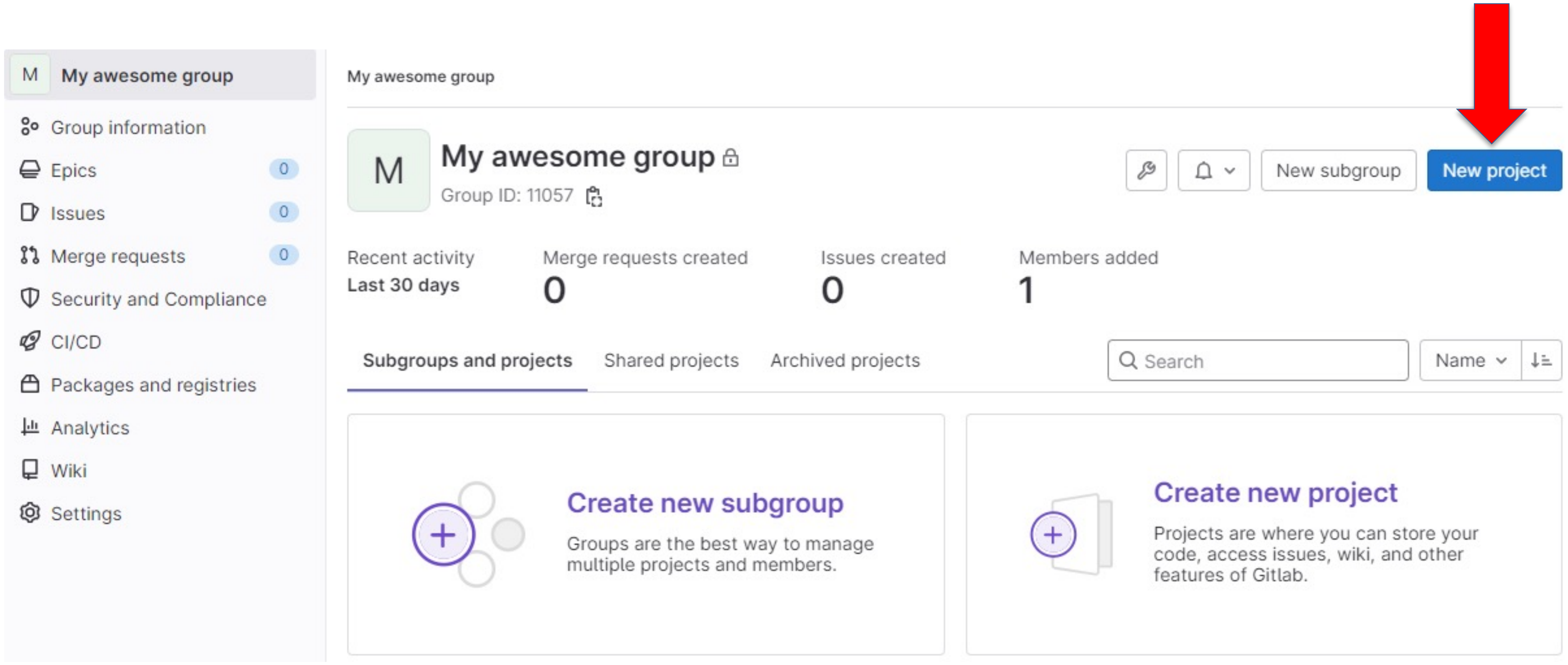
Who will be able to see this group? View the documentation

[✓ Private]
The group and its projects can only be viewed by members.

Internal
The group and any internal projects can be viewed by any logged in user except external users.

Public
The group and any public projects can be viewed without any authentication.

# Your New Group


My awesome group

My awesome group 🔒

Group ID: 11057

Recent activity
Last 30 days

| Merge requests created | Issues created | Members added |
|---|---|---|
| 0 | 0 | 1 |

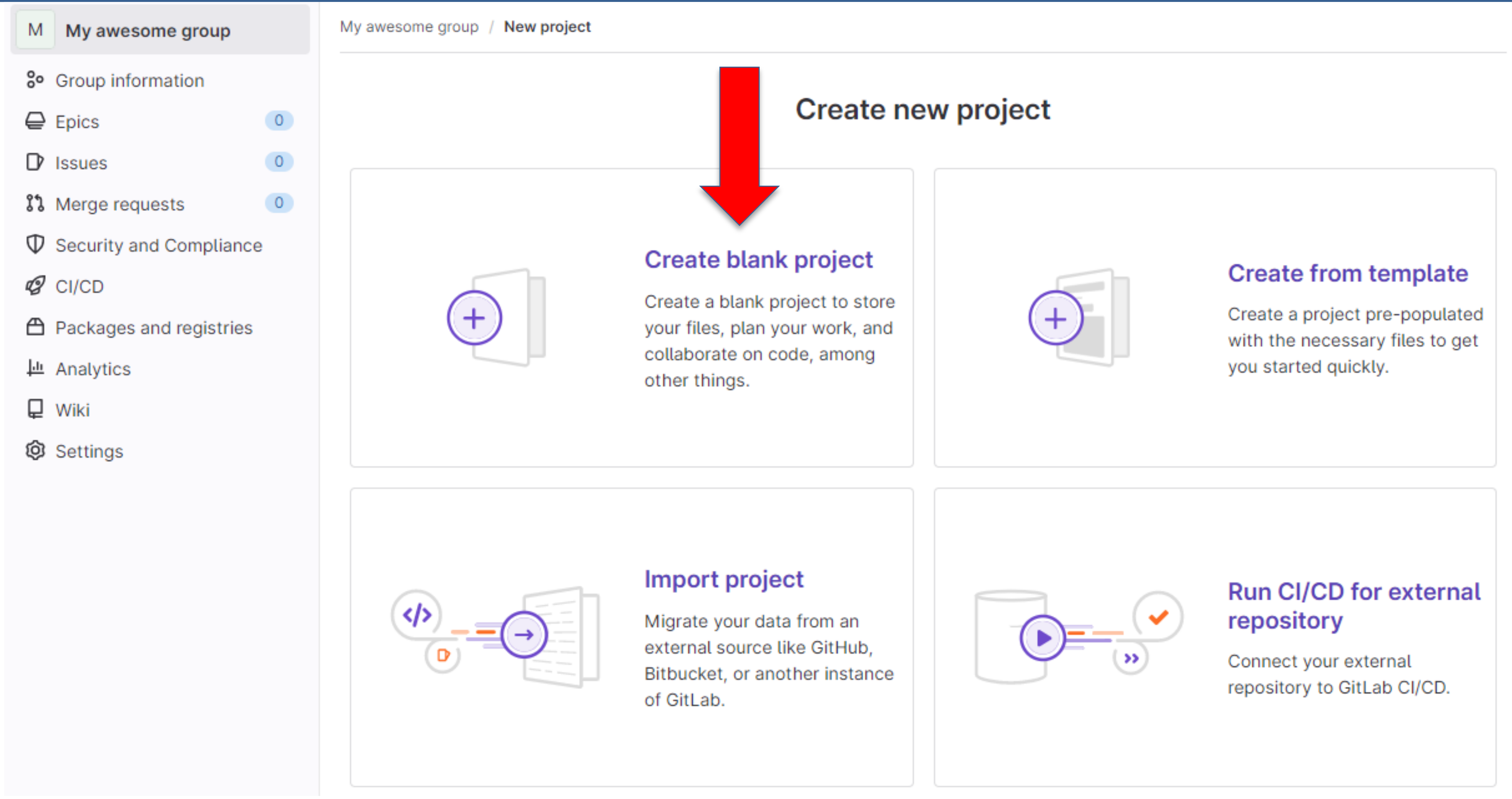
Create new subgroup

Groups are the best way to manage multiple projects and members.

Create new project

Projects are where you can store your code, access issues, wiki, and other features of Gitlab.

# Create a New Project



My awesome group / New project

M My awesome group

- Group information
- Epics 0
- Issues 0
- Merge requests 0
- Security and Compliance
- CI/CD
- Packages and registries
- Analytics
- Wiki
- Settings

Create new project

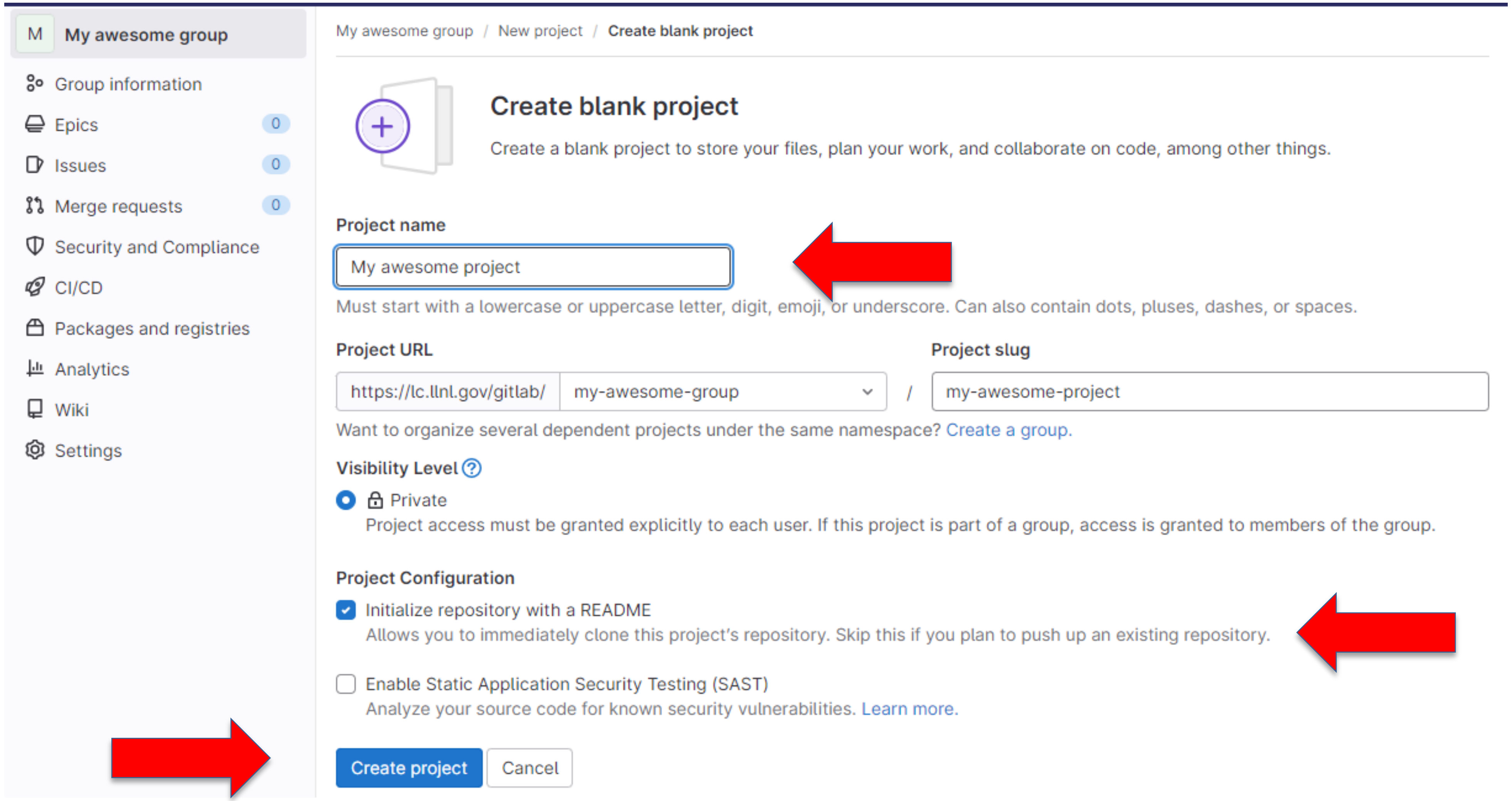
**Create blank project**
Create a blank project to store your files, plan your work, and collaborate on code, among other things.

**Create from template**
Create a project pre-populated with the necessary files to get you started quickly.

**Import project**
Migrate your data from an external source like GitHub, Bitbucket, or another instance of GitLab.

**Run CI/CD for external repository**
Connect your external repository to GitLab CI/CD.

# Fill in Create Project Form



My awesome group / New project / **Create blank project**

## Create blank project

Create a blank project to store your files, plan your work, and collaborate on code, among other things.

**Project name**

My awesome project

Must start with a lowercase or uppercase letter, digit, emoji, or underscore. Can also contain dots, pluses, dashes, or spaces.

| Project URL | | | Project slug |
|---|---|---|---|
| https://lc.llnl.gov/gitlab/ | my-awesome-group | / | my-awesome-project |

Want to organize several dependent projects under the same namespace? Create a group.

**Visibility Level**

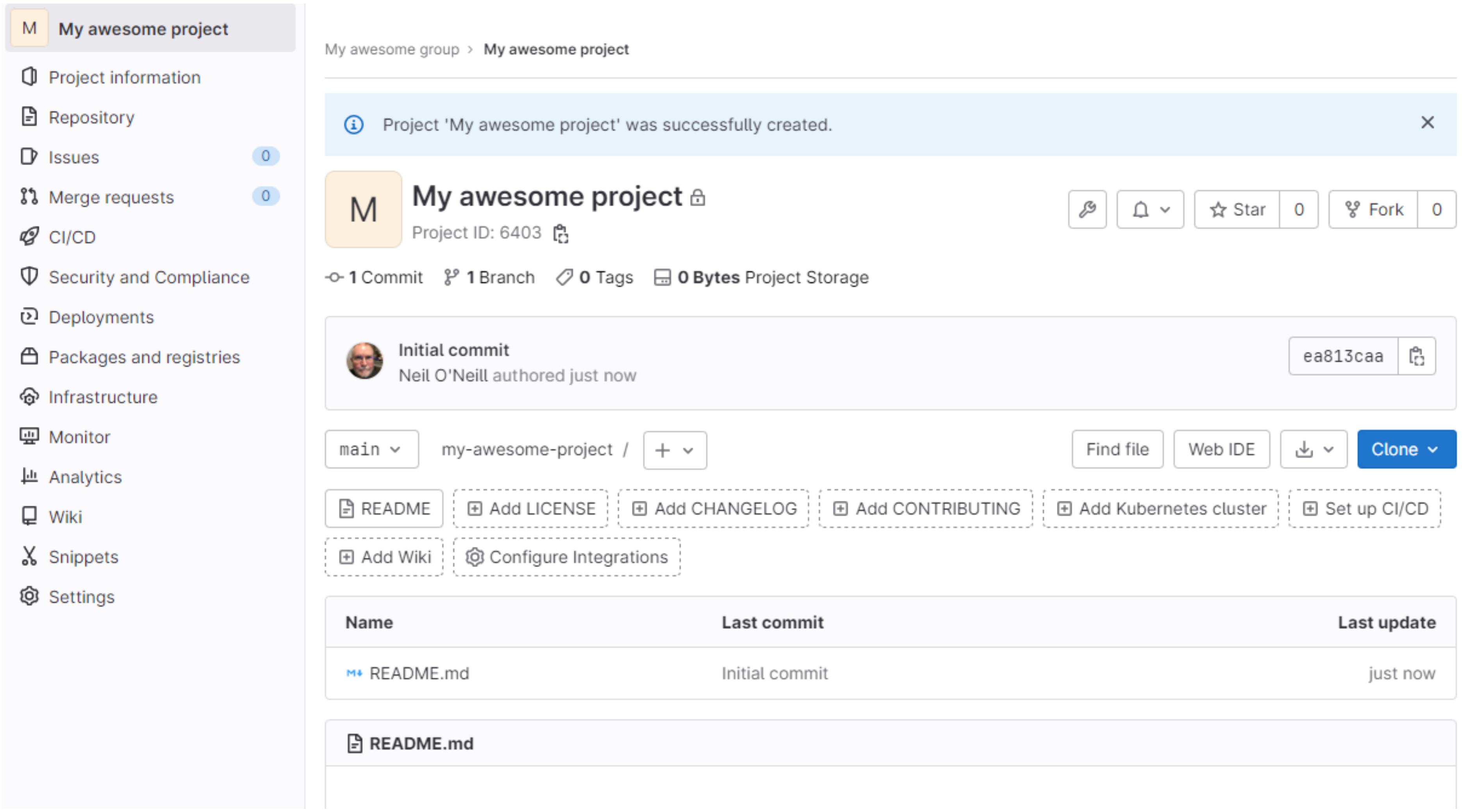
[✓ Private]
Project access must be granted explicitly to each user. If this project is part of a group, access is granted to members of the group.

**Project Configuration**

[✓] Initialize repository with a README
Allows you to immediately clone this project's repository. Skip this if you plan to push up an existing repository.

[ ] Enable Static Application Security Testing (SAST)
Analyze your source code for known security vulnerabilities. Learn more.

# Your New Project Page


My awesome group › My awesome project

Project 'My awesome project' was successfully created.

My awesome project
Project ID: 6403

1 Commit · 1 Branch · 0 Tags · 0 Bytes Project Storage

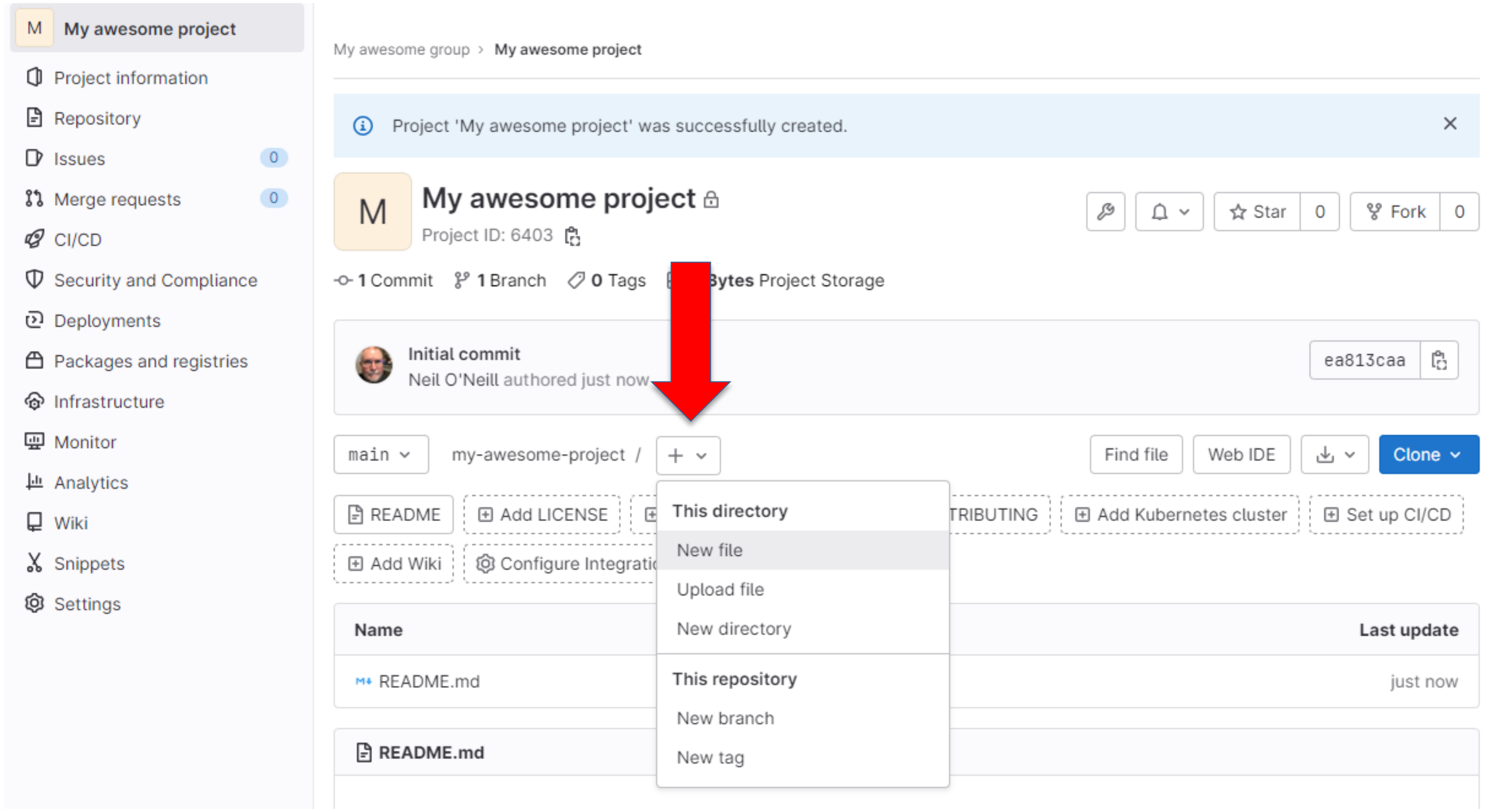
Initial commit
Neil O'Neill authored just now
ea813caa

main · my-awesome-project /

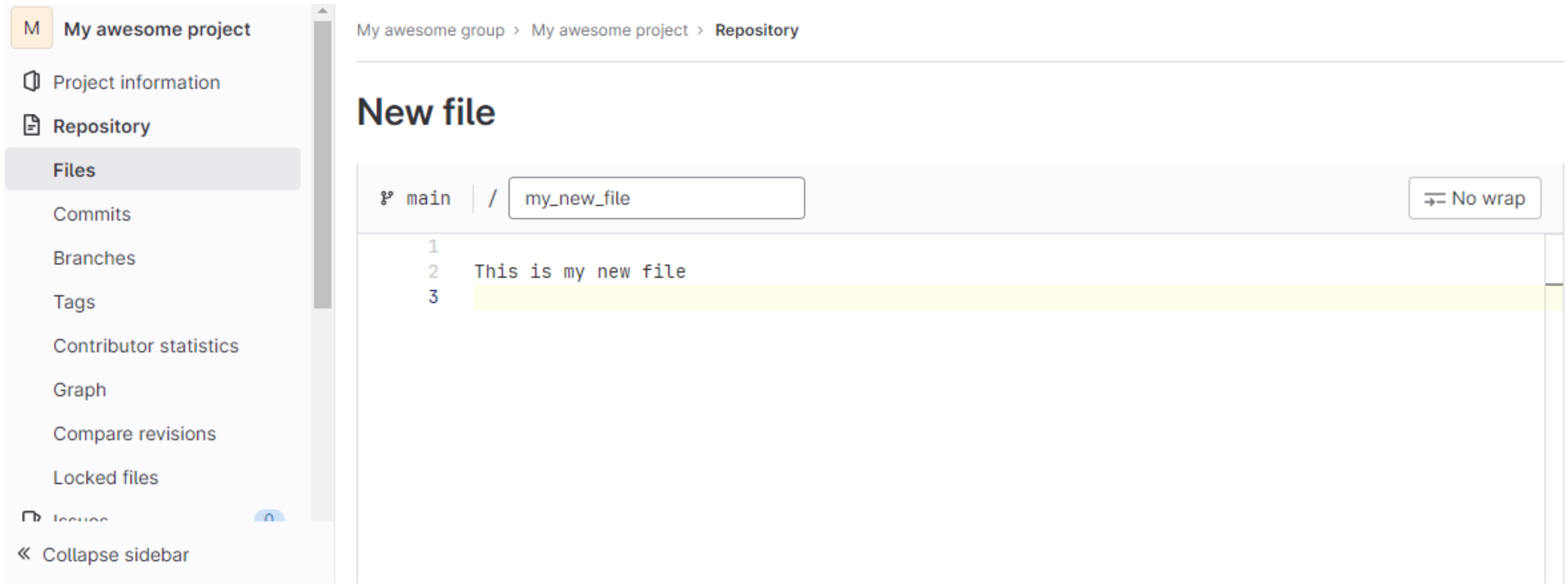
| Name | Last commit | Last update |
|---|---|---|
| README.md | Initial commit | just now |

README.md

# Create New Files and Directories from within GitLab



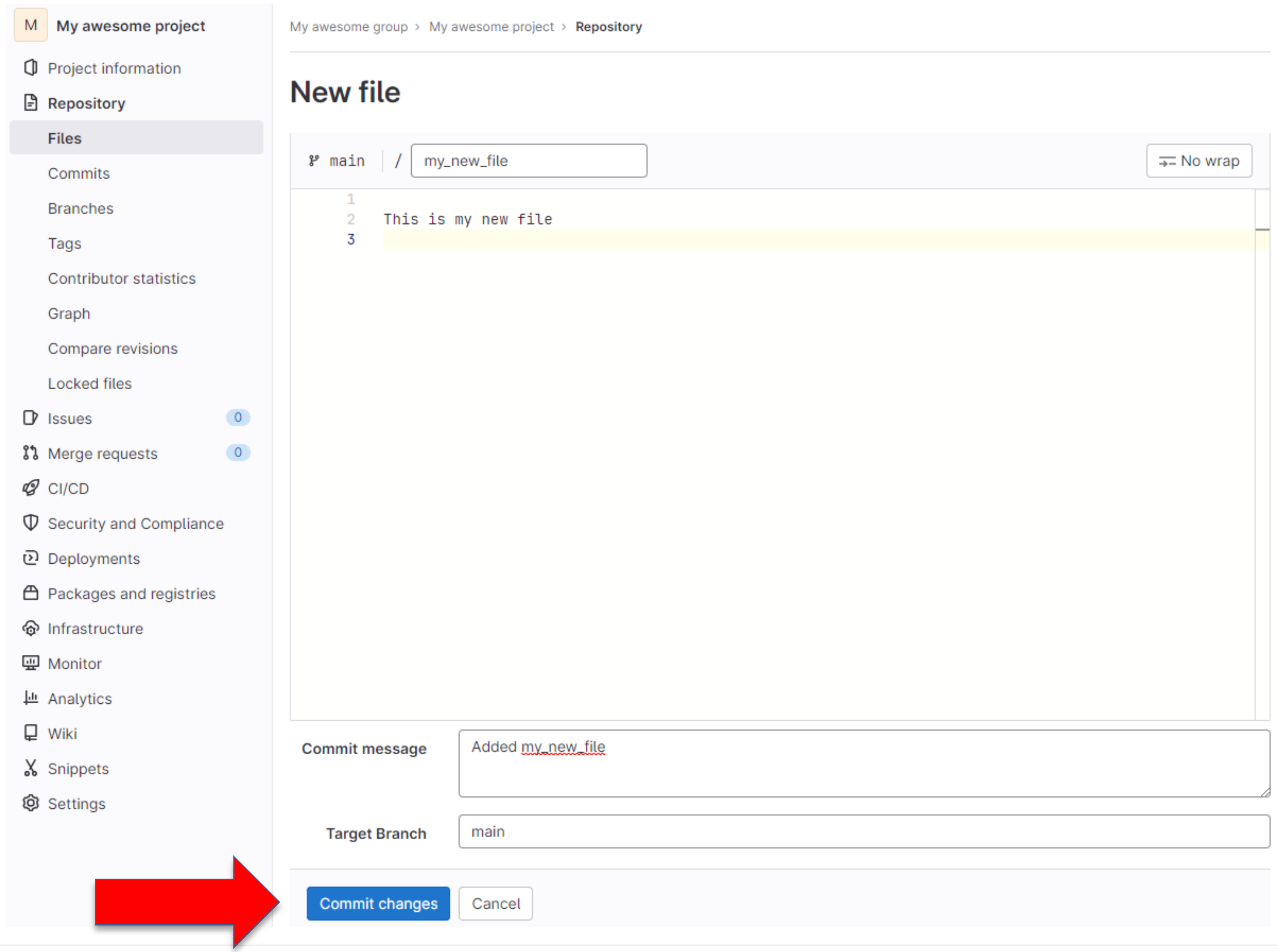
## Edit Files from within GitLab



my_new_file

```
This is my new file
```

# Commit Changes from within GitLab



New file

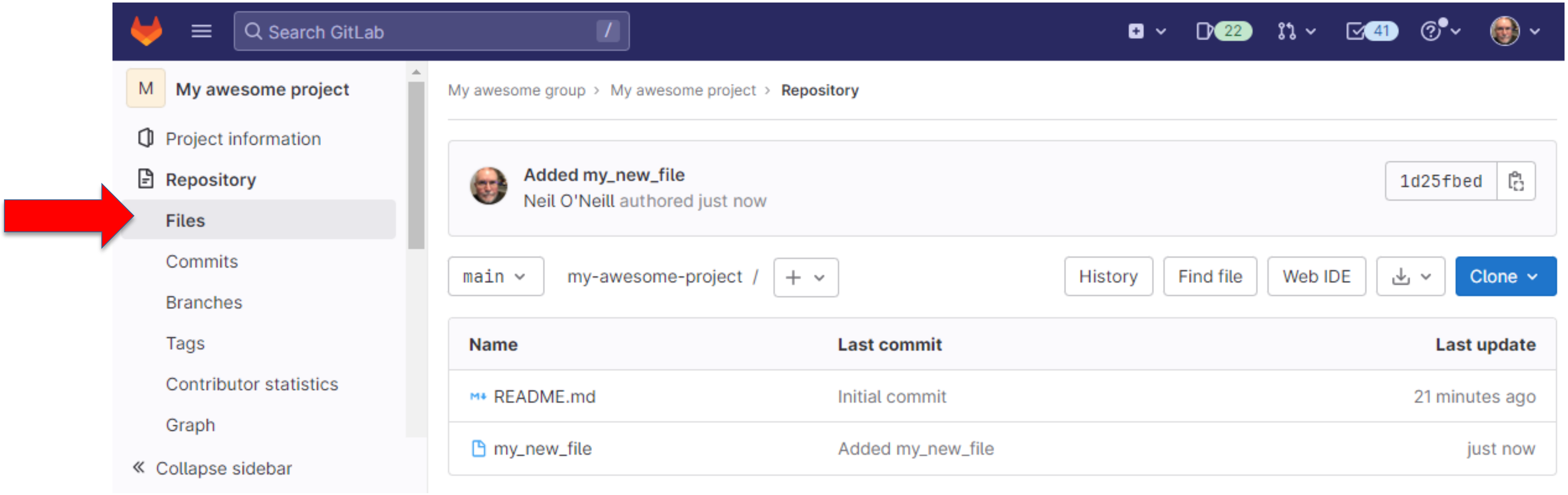
main / my_new_file

```

This is my new file

```

Commit message: Added my_new_file

Target Branch: main

# View Repository from Project Page

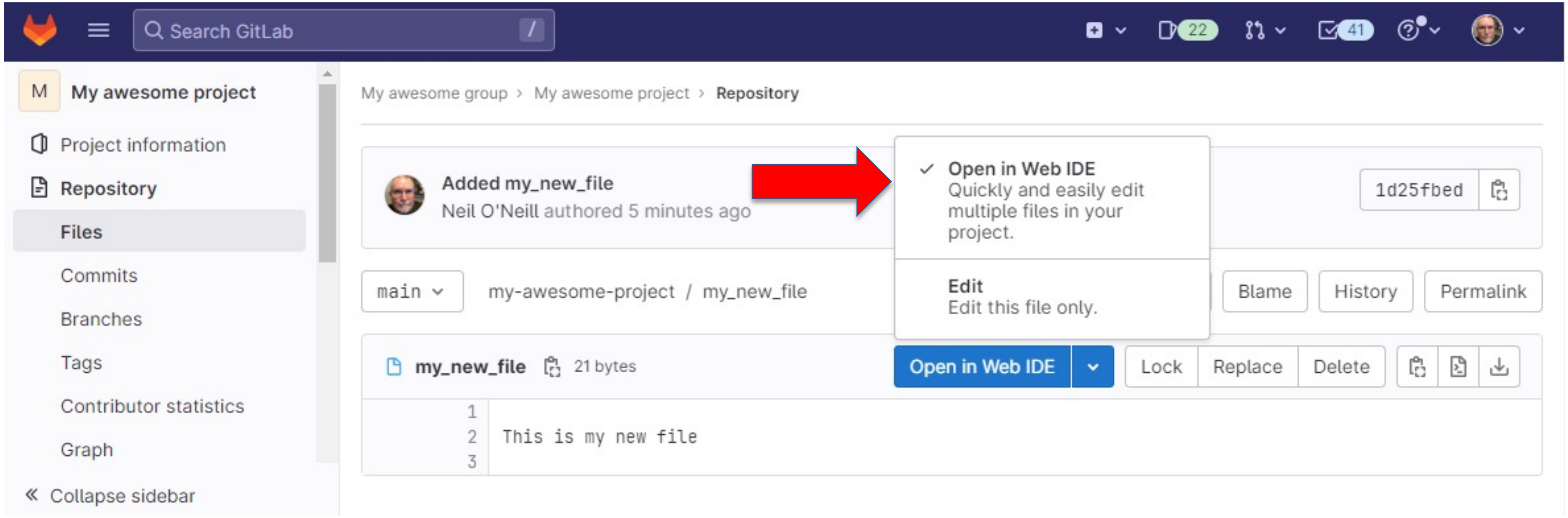


**Added my_new_file**
Neil O'Neill authored just now

1d25fbed

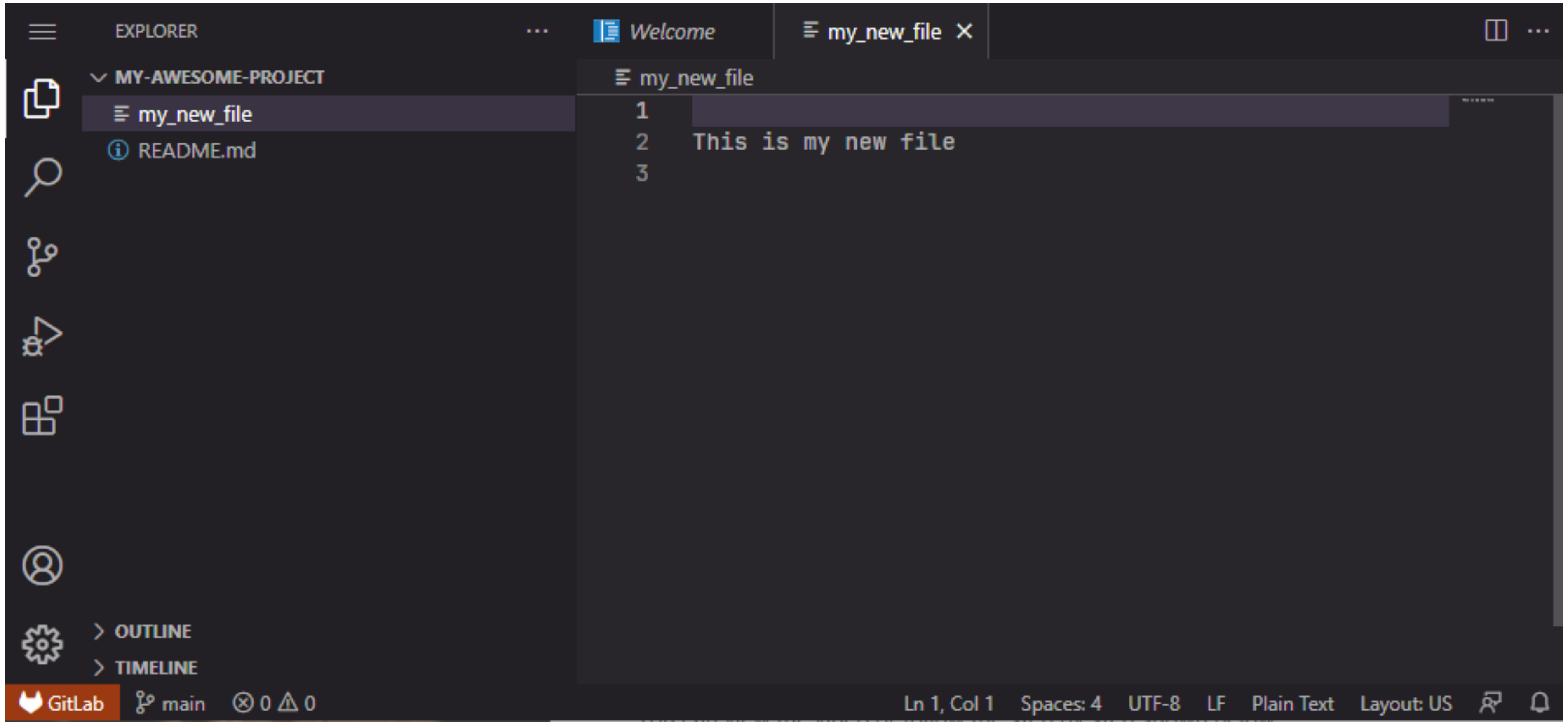
| Name | Last commit | Last update |
|---|---|---|
| README.md | Initial commit | 21 minutes ago |
| my_new_file | Added my_new_file | just now |

# VSCODE IDE is also Available from within GitLab
## (GitLab calls it "Web IDE")



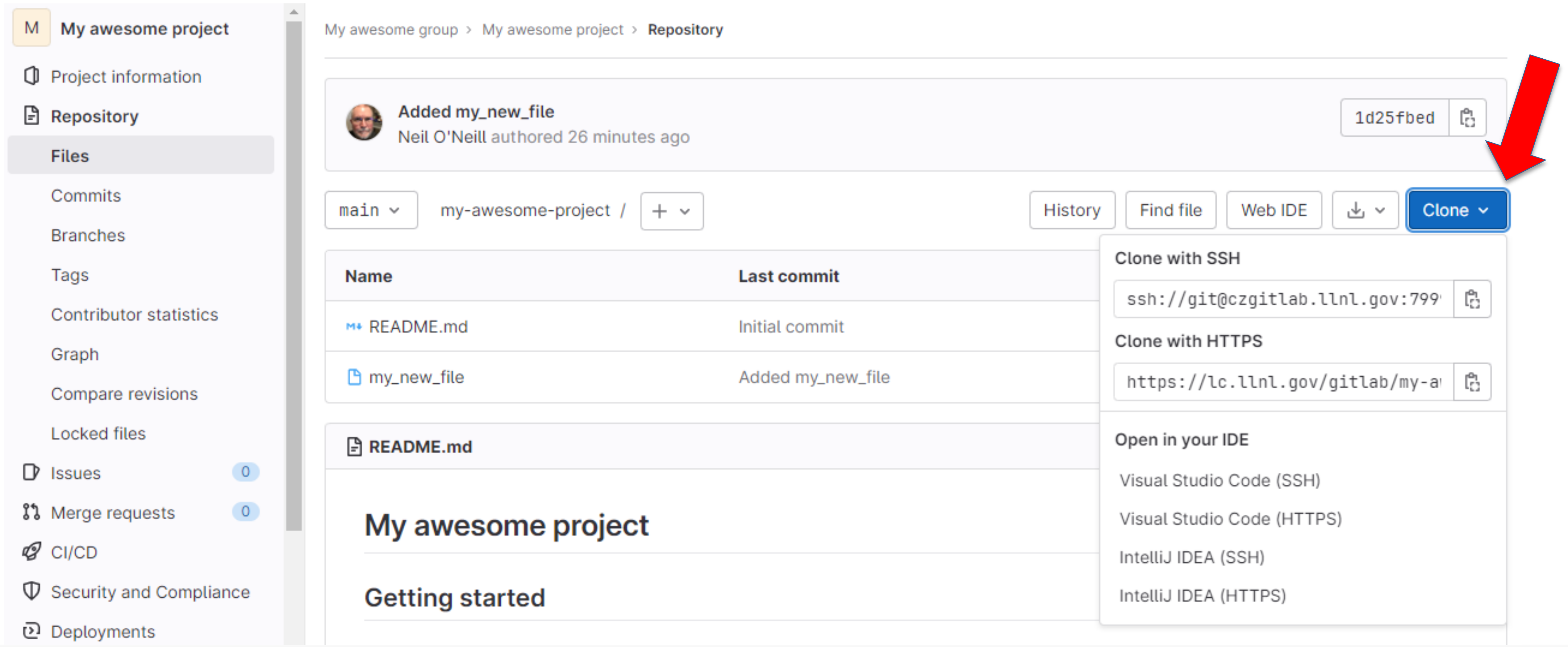
```

This is my new file

```

## Web IDE (VSCODE)



```

This is my new file

```

# Find Your Clone URLs



Added my_new_file
Neil O'Neill authored 26 minutes ago

1d25fbed

| Name | Last commit |
|---|---|
| README.md | Initial commit |
| my_new_file | Added my_new_file |

README.md

Clone with SSH

`ssh://git@czgitlab.llnl.gov:799`

Clone with HTTPS

`https://lc.llnl.gov/gitlab/my-a`

Open in your IDE

- Visual Studio Code (SSH)
- Visual Studio Code (HTTPS)
- IntelliJ IDEA (SSH)
- IntelliJ IDEA (HTTPS)

## Cloning from the Command Line: SSH & HTTP

```
>  git clone ssh://git@czgitlab.llnl.gov:7999/my-awesome-group/my-awesome-project.git
Cloning into 'my-awesome-project'...
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 6 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (6/6), done.
```

SSH

HTTP

```
> git clone https://lc.llnl.gov/gitlab/my-awesome-group/my-awesome-project.git
Cloning into 'my-awesome-project'...
Username for 'https://lc.llnl.gov’: myusername
Password for 'https://myusername@lc.llnl.gov':
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 6 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (6/6), done.
```

## Authentication for git Commands

- SSH Keys
  - Use 4096-bit RSA keys
  - Enter your keys into your GitLab account: https://lc.llnl.gov/gitlab/-/profile/keys
  - https://dev.llnl.gov/securityaccess/ssh/
  - https://dev.llnl.gov/securityaccess/ssh/cz_user/
  - https://dev.llnl.gov/securityaccess/ssh/rz_user/
  - https://lc.llnl.gov/confluence/display/GITLAB/GitLab+FAQ#GitLabFAQ-Q.HowdoIsetupSSHkeysonanLCsystem?

- Personal Access Token (PAT)
  - Create here: https://lc.llnl.gov/gitlab/-/profile/personal_access_tokens
  - When asked for “password” use PAT instead.
  - PATs generated on LC GitLab instances have a 30 day lifetime.

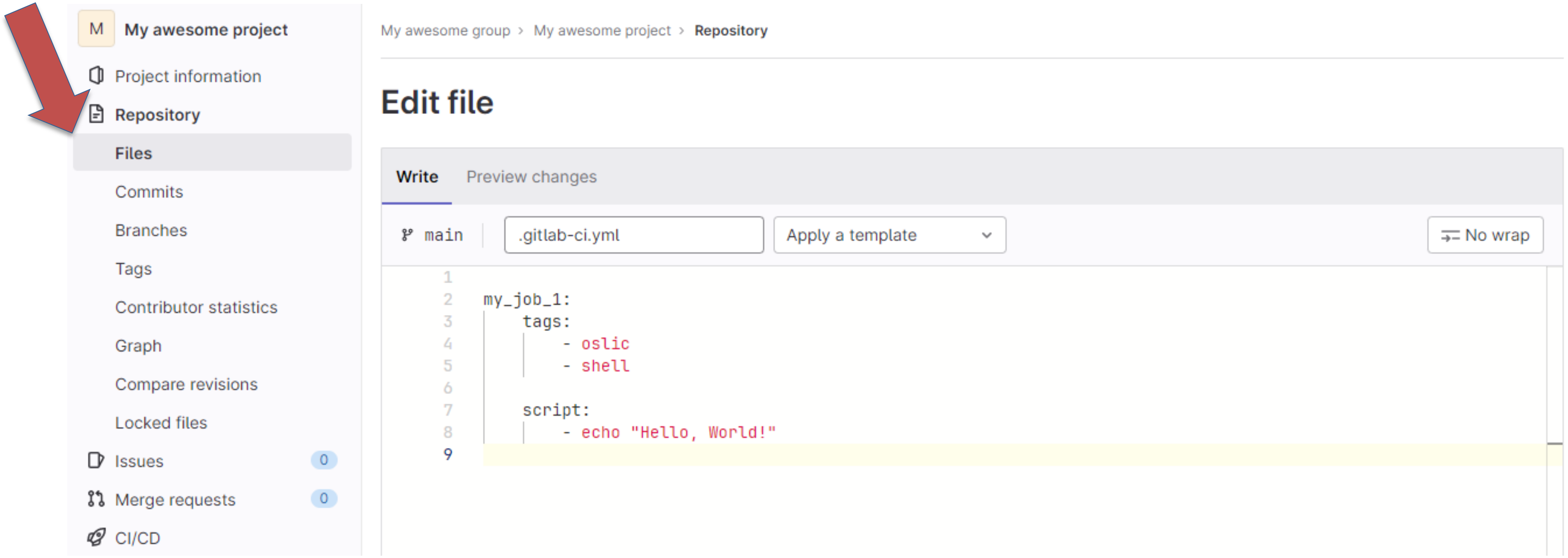
## GitLab Continuous Integration (CI)

- Makes use of software agents (systemd services) called “runners” running on the login nodes of all LC production machines.  This allows GitLab to run scripts on any production machine in the Computer Center.
- Individual scripts (literally bash scripts) that get run on runners are referred to as “jobs”.
- A collection of jobs, possibly dependent on one another and possibly running on different machines is referred to as a “pipeline”.
  - https://docs.gitlab.com/ee/ci/pipelines/
- The idea behind CI is that software projects get built and tested every time any significant change is made.

## .gitlab-ci.yml file

- GitLab uses the “configuration as code” principle, and defines piplelines using a YAML file, .gitlab-ci.yml, located in the top-level of each project repository. This file is created by the user.
  - https://docs.gitlab.com/ee/ci/yaml/
- “YAML is a human-friendly data serialization language”
  - https://yaml.org/
- The .gitlab-ci.yml file is a complete description of your CI pipleline, including what jobs are run, when they run, where they run, and what they do when they are run.
- Note that if you have a .gitlab-ci.yml file in your project then GitLab will attempt to run it whenever you make a new commit. Put “[skip-ci]” somewhere in your commit comment to prevent this.

# Simple .gitlab-ci.yml

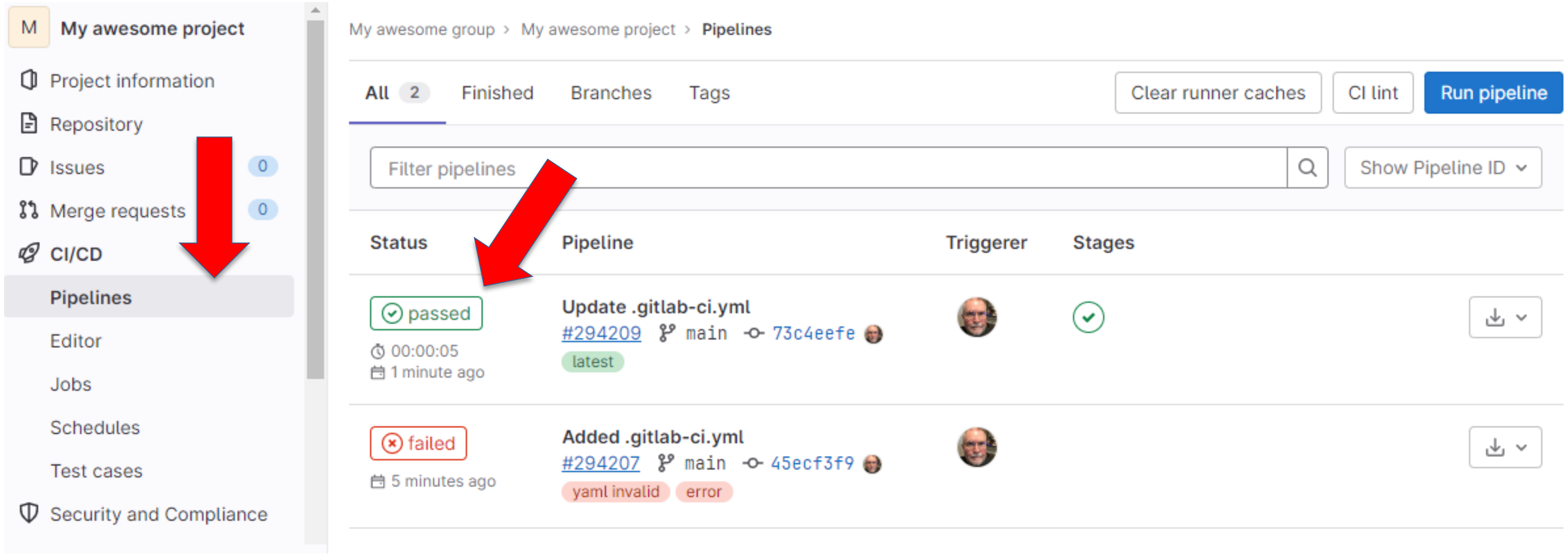


Edit file

.gitlab-ci.yml

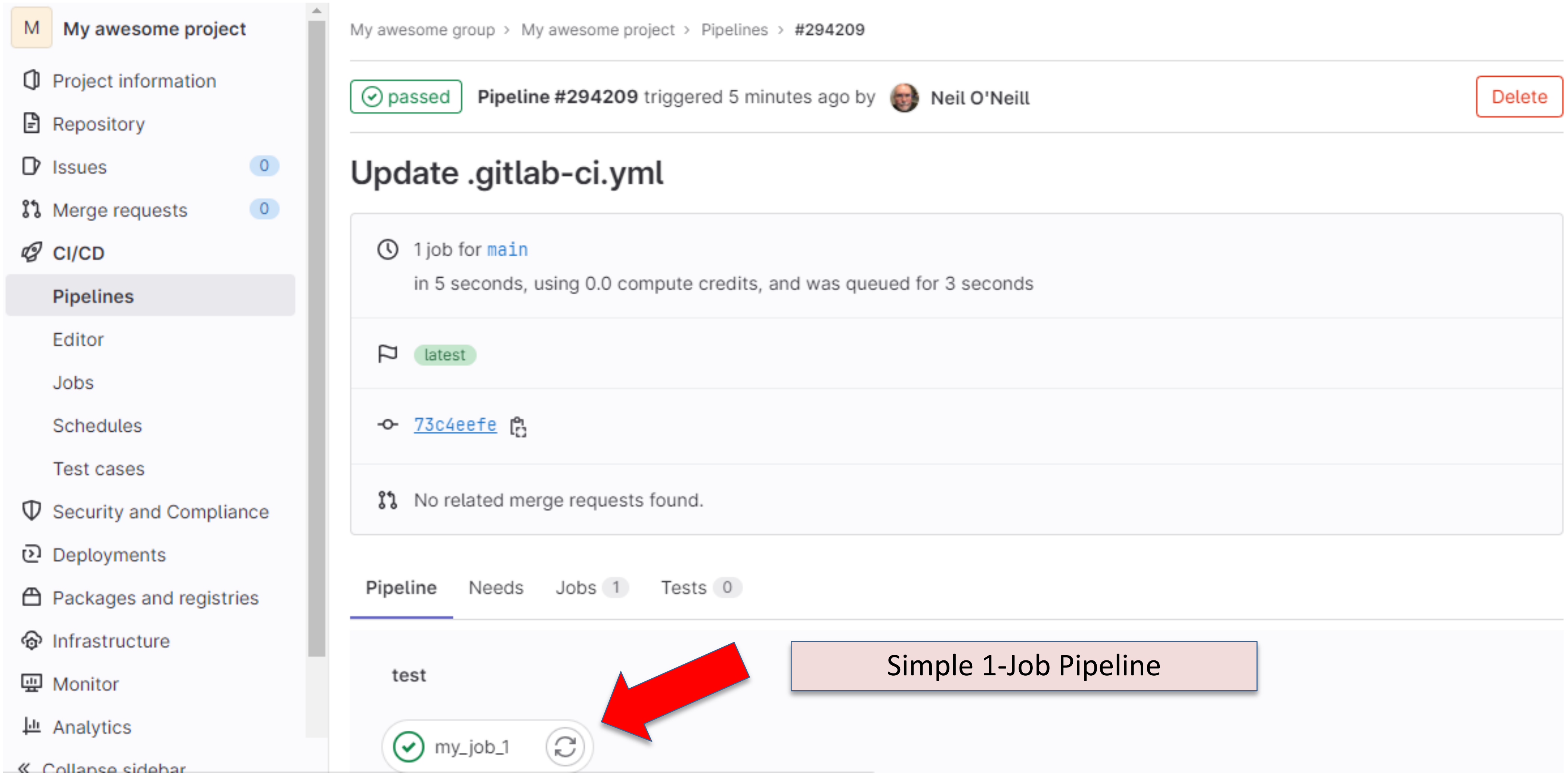
```yaml

my_job_1:
    tags:
        - oslic
        - shell

    script:
        - echo "Hello, World!"

```

# See a List of Your Pipelines



| Status | Pipeline | Triggerer | Stages |
|---|---|---|---|
| passed 00:00:05 1 minute ago | Update .gitlab-ci.yml #294209 main 73c4eefe latest | | ✓ |
| failed 5 minutes ago | Added .gitlab-ci.yml #294207 main 45ecf3f9 yaml invalid error | | |

## See an Individual Pipeline



My awesome group › My awesome project › Pipelines › #294209

passed **Pipeline #294209** triggered 5 minutes ago by **Neil O'Neill** Delete

**Update .gitlab-ci.yml**

1 job for `main`
in 5 seconds, using 0.0 compute credits, and was queued for 3 seconds

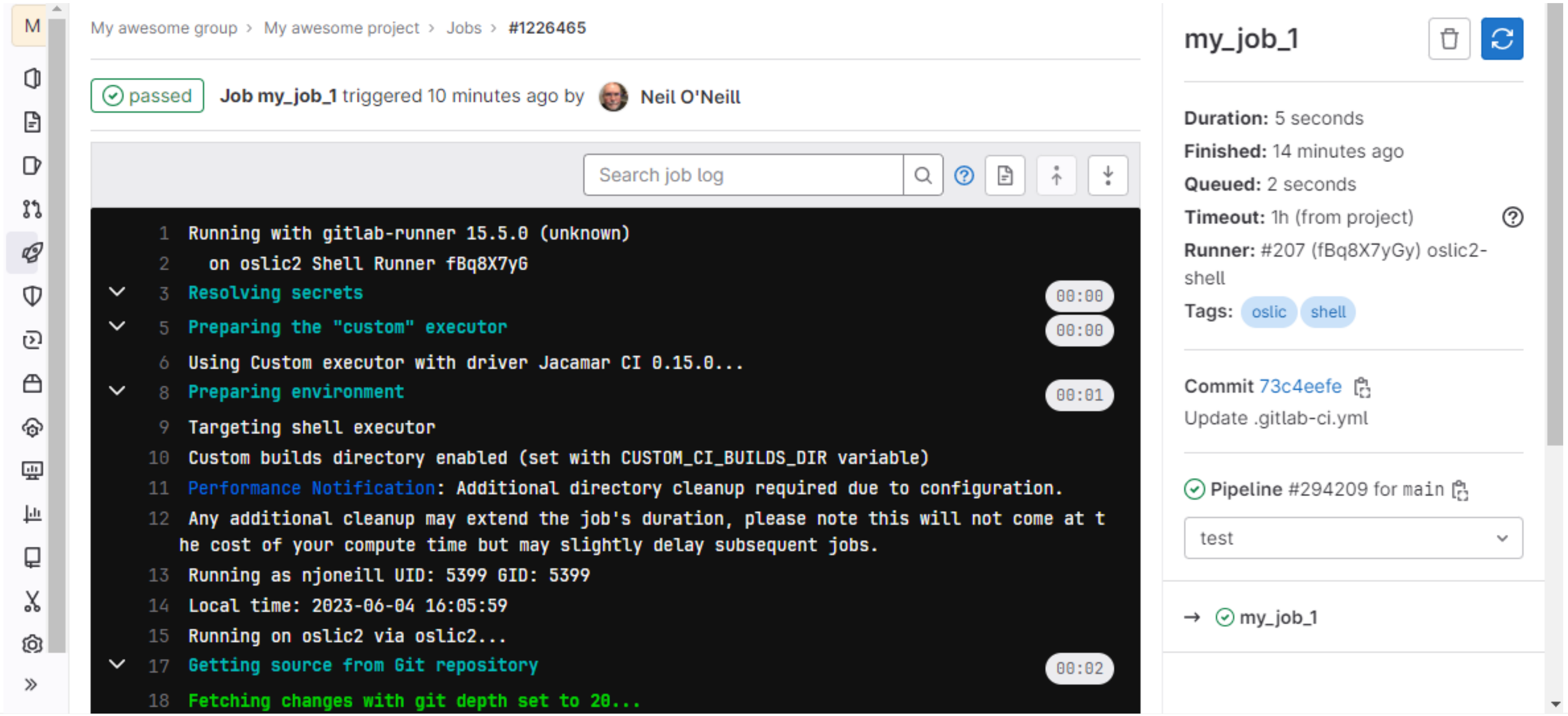
latest

`73c4eefe`

No related merge requests found.

Simple 1-Job Pipeline

# See the Log for a Particular Job



M
My awesome group › My awesome project › Jobs › #1226465
passed
Job my_job_1 triggered 10 minutes ago by Neil O'Neill
Search job log

```
Running with gitlab-runner 15.5.0 (unknown)
  on oslic2 Shell Runner fBq8X7yG
Resolving secrets                                                        00:00
Preparing the "custom" executor                                          00:00
Using Custom executor with driver Jacamar CI 0.15.0...
Preparing environment                                                    00:01
Targeting shell executor
Custom builds directory enabled (set with CUSTOM_CI_BUILDS_DIR variable)
Performance Notification: Additional directory cleanup required due to configuration.
Any additional cleanup may extend the job's duration, please note this will not come at t
he cost of your compute time but may slightly delay subsequent jobs.
Running as njoneill UID: 5399 GID: 5399
Local time: 2023-06-04 16:05:59
Running on oslic2 via oslic2...
Getting source from Git repository                                       00:02
Fetching changes with git depth set to 20...
```

my_job_1

**Duration:** 5 seconds
**Finished:** 14 minutes ago
**Queued:** 2 seconds
**Timeout:** 1h (from project)
**Runner:** #207 (fBq8X7yGy) oslic2-shell
**Tags:** oslic shell

**Commit** 73c4eefe
Update .gitlab-ci.yml

Pipeline #294209 for main

## Stages vs. Directed Acyclic Graphs

- Serial operations in pipelines can be controlled either by using “stages” or by using directed acyclic graphs (dags)

- Stages (basic pipelines)
  - Jobs declare what stage they belong to via the “stage” keyword.
  - All jobs in each stage will run before the next stage is started.
  - Default stages (.pre, build, test, deploy, .post)
  - Can create custom stages with “stages” keyword in .gitlb-ci.yml
  - See: https://docs.gitlab.com/ee/ci/yaml/#stages

- Directed Acyclic Graphs
  - Jobs use the “needs” keyword to declare which other jobs in the pipeline they depend on.
  - See: https://docs.gitlab.com/ee/ci/yaml/#needs

# Multi-stage Pipeline .gitlab-ci.yml

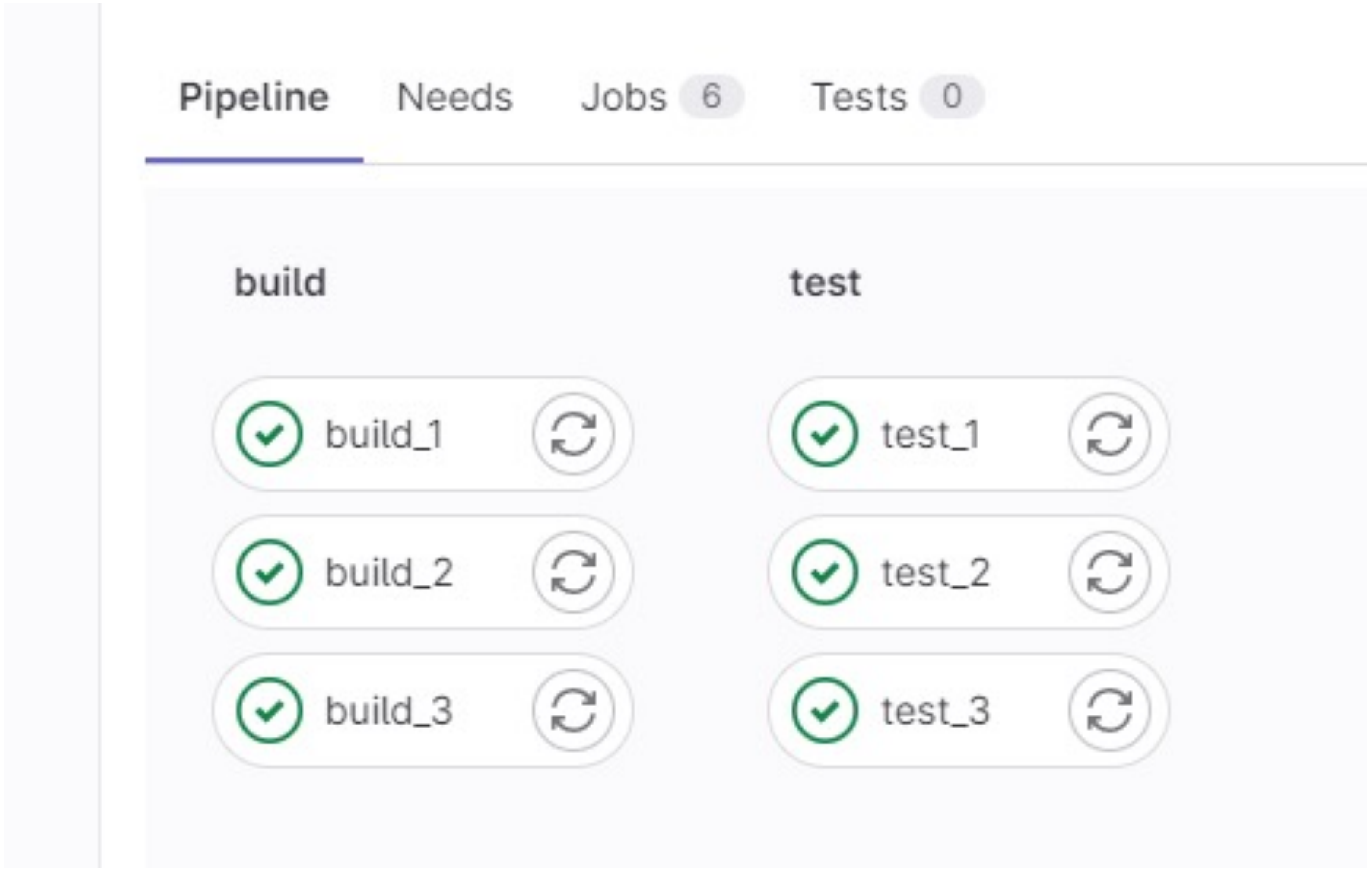
test stage

```
test_1:
   stage: test
   tags:
      - oslic
      - shell
   script:
      - echo "This is test_1 on oslic"
test_2:
   stage: test
   tags:
      - ruby
      - shell
   script:
      - echo "This is test_2 on ruby"
test_3:
   stage: test
   tags:
      - lassen
      - shell
   script:
      - echo "This is test_3 on lassen“
```

build stage

```
build_1:
   stage: build
   tags:
      - oslic
      - shell
   script:
      - echo "This is build_1 on oslic"
build_2:
   stage: build
   tags:
      - ruby
      - shell
   script:
      - echo "This is build_2 on ruby"
build_3:
   stage: build
   tags:
      - lassen
      - shell
   script:
      - echo "This is build_3 on lassen"
```

## Multi-stage Pipeline Run

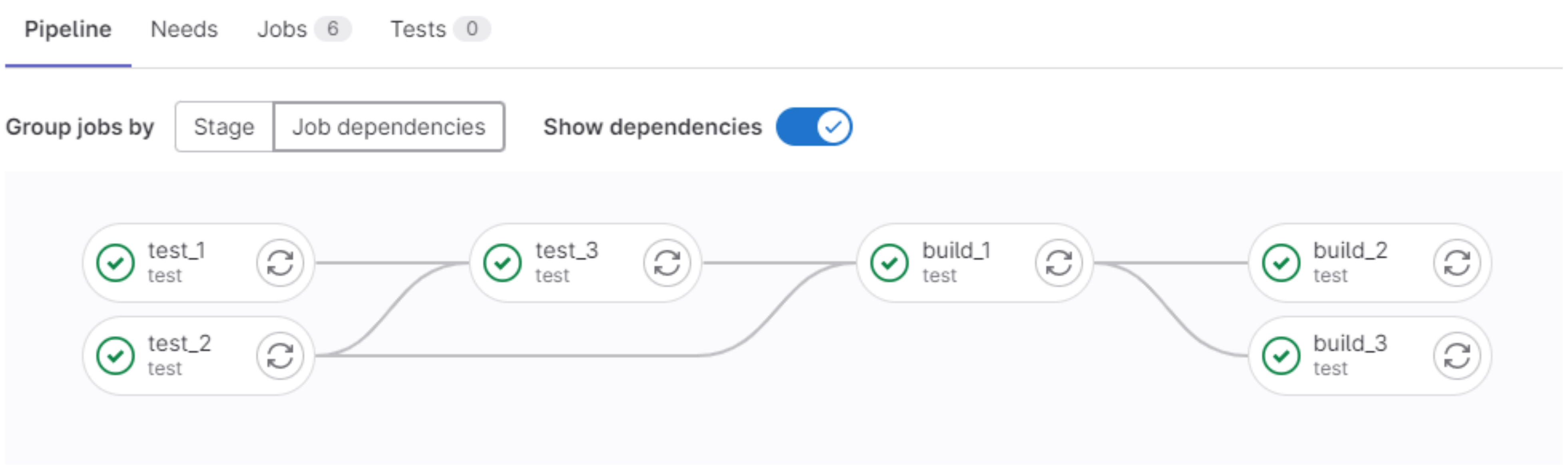


# DAG-based Pipeline .gitlab-ci.yml

```
test_1:
  tags:
    - oslic
    - shell
  script:
    - echo "This is test_1 on oslic"
test_2:
  tags:
    - ruby
    - shell
  script:
    - echo "This is test_2 on ruby"
test_3:
  needs:
    - test_1
    - test_2
  tags:
    - quartz
    - shell
  script:
    - echo "This is test_3 on quartz"
```

```
build_1:
  needs: [test_1, test_2, test_3]
  tags:
    - oslic
    - shell
  script:
    - echo "This is build_1 on oslic"
build_2:
  needs: [build_1]
  tags:
    - ruby
    - shell
  script:
    - echo "This is build_2 on ruby"
build_3:
  needs: [build_1]
  tags:
    - quartz
    - shell
  script:
    - echo "This is build_3 on quartz"
```

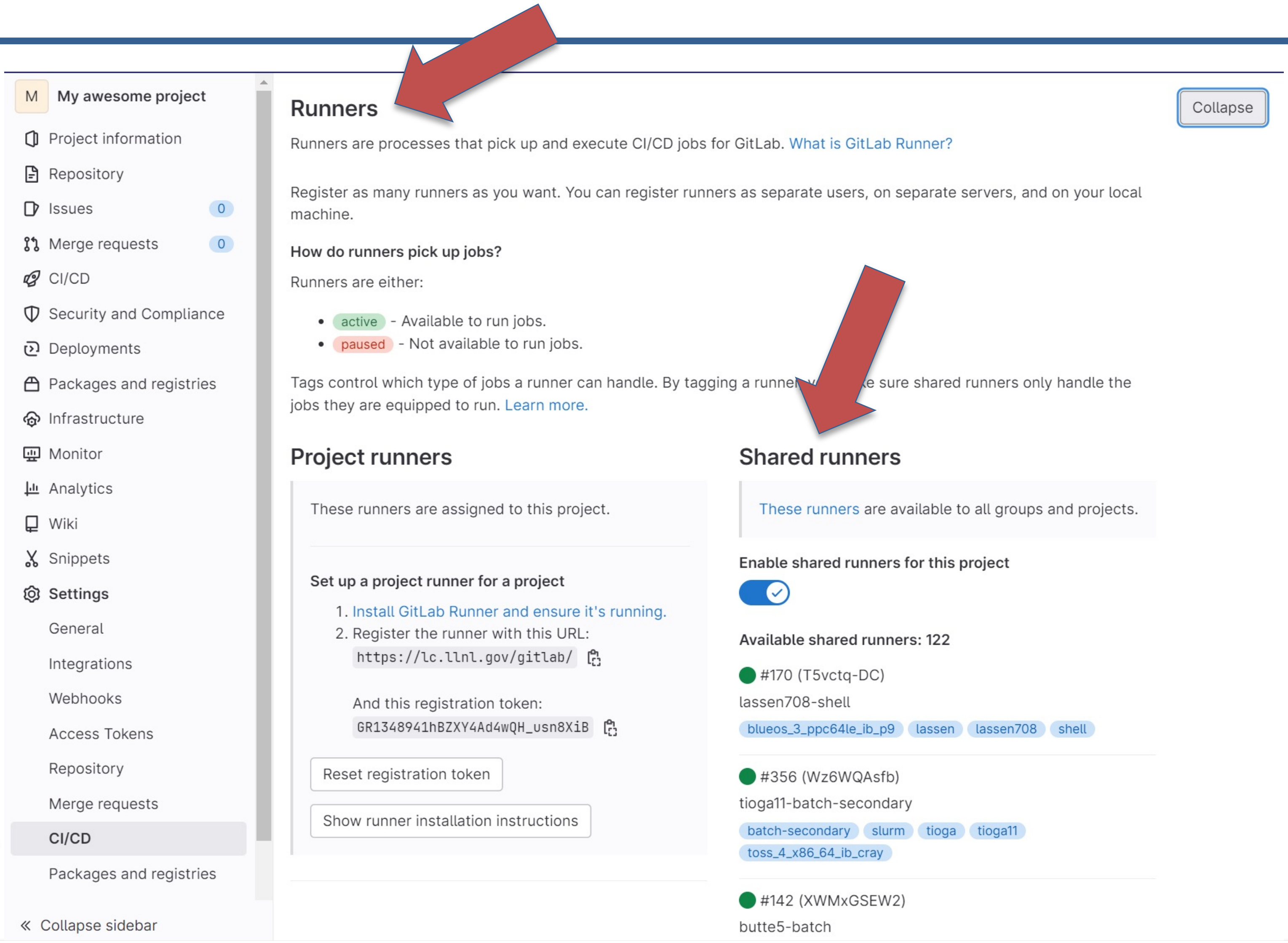
# DAG-based Pipeline Run



## LC “tags” to Choose Runner Host

- Need to use both a “machine” tag and a “runner type” tag.
  - Machine: oslic, ruby, quartz, etc.
  - Runner type: shell, batch, slurm, lsf, flux
- Need to have an account on the tagged machine or job will fail.
- Note that these “tags” have nothing to do with git commit tags.
- “batch” will get you a runner type that matches the main batch schedular used on a particular machine. For example, slurm on quartz, or LSF on lassen.
- Lists of available tags for production machines can be found here: [https://lc.llnl.gov/confluence/display/GITLAB/GitLab+CI#GitLabCI-RunnerDeploymentsandStatus](https://lc.llnl.gov/confluence/display/GITLAB/GitLab+CI#GitLabCI-RunnerDeploymentsandStatus) (but info may be out of date).
- 100% up-to-date tag information can always be found in Settings→CI/CD→Runners from a project page.

# Runner Tag and Status Information



## 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 runne[...]e 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://lc.llnl.gov/gitlab/`

   And this registration token:
   `GR1348941hBZXY4Ad4wQH_usn8XiB`

Reset registration token

Show runner installation instructions

### Shared runners

These runners are available to all groups and projects.

**Enable shared runners for this project**

✓

**Available shared runners: 122**

● #170 (T5vctq-DC)
lassen708-shell
blueos_3_ppc64le_ib_p9 lassen lassen708 shell

● #356 (Wz6WQAsfb)
tioga11-batch-secondary
batch-secondary slurm tioga tioga11 toss_4_x86_64_ib_cray

● #142 (XWMxGSEW2)
butte5-batch

## Jacamar Runner

- All LC production machines exclusively use instances of the Jacamar runner
  - https://ecp-ci.gitlab.io/docs/admin.html#jacamar-ci
  - Technically, Jacamar is an instance of a GitLab "custom executor"— https://docs.gitlab.com/runner/executors/custom.html
- Jacamar was developed as a project within the larger Exascale Computing Project (ECP) and has become the de facto standard at DOE HPC computing facilities.
- Jacamar runners have two modes of operation
  - "shell": your job script runs in a bash shell under your account on a login node of the selected cluster.
  - "batch": your job script runs in a bash shell within a batch allocation under your account. The type of node (login, compute, launch) on the cluster that your script will run on depends on the type of schedular installed on the cluster. See the table here for details: https://lc.llnl.gov/confluence/display/GITLAB/GitLab+CI#GitLabCI-runnersRunners

## Jacamar Shell Runner

- Select by using the tag “shell”.
- Runs in a bash shell under your user account.
- Uses a non-interactive shell, so environment may not be the same as you get during a normal interactive login.  This can cause things that work when run interactively “by hand” not to work when run as a GitLab CI job.

## Jacamar Batch Runner

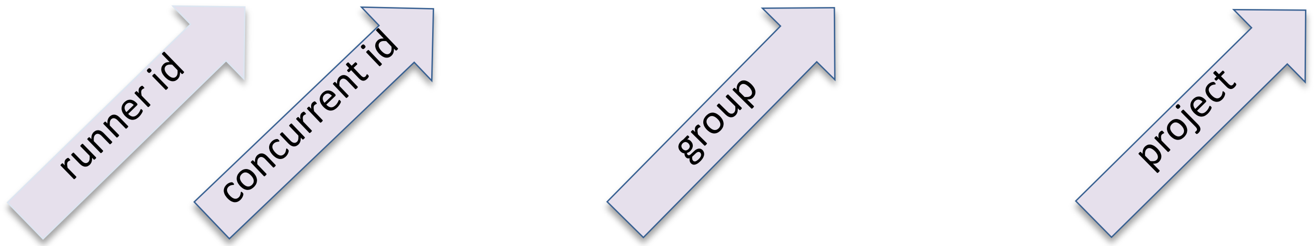
- Select by using the generic tag “batch”, or one of the specific tags “slurm”, “lsf”, or “flux”.
- Specify schedular options with special variables in .gitlab-ci.yml file.

```
variables:
   LLNL_SLURM_SCHEDULER_PARAMETERS: "--nodes=1 -p pdebug"
   LLNL_LSF_SCHEDULER_PARAMETERS: "-q pbatch -nnodes 2"
   LLNL_FLUX_SCHEDULAR_PARAMETERS: "-N2 –n1"
```

  — these variables can be specified in the global “variables” section, or in the “variables” section for any particular job.

# Jacamar Build Directories

- By default, Jacamar will create a directory at ~/.jacamar-ci to use as the top-level directory for all your GitLab CI builds.
- Depending on what your builds look like, this can cause you to exceed your home directory disk quota.  See here for ways to protect your home directory quota: [https://lc.llnl.gov/confluence/display/GITLAB/First+pipeline+with+LC+Gitlab+CI#FirstpipelinewithLCGitlabCI-Protectyourhomequota](https://lc.llnl.gov/confluence/display/GITLAB/First+pipeline+with+LC+Gitlab+CI#FirstpipelinewithLCGitlabCI-Protectyourhomequota)
- Example build directory path (created by gitlab-runner).  Reused – never deleted.
  - ~/.jacamar-ci/builds/QcvJxi8A/004/gitlab/my-awesome-group/my-awesome-project



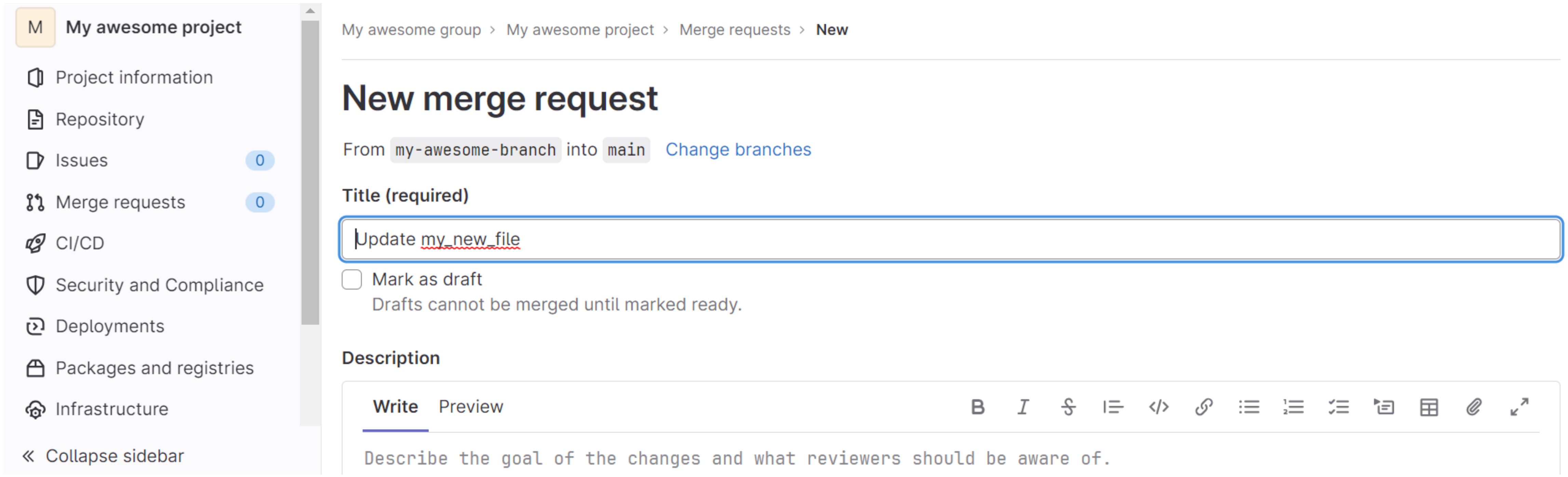
## Merge Requests

- Provides a code auditing/approval step for software projects.

- Typical work flow:
  1. Create new branch of your repo (my-awesome-branch)
  2. Make your changes to my-awesome-branch.
  3. Commit your changes.
  4. Run CI pipeline against my-awesome-branch (assume success).
  5. Create merge request requesting to merge my-awesome-branch into main branch.
  6. Merge request approval (by defined approvers) and my-awesome-branch is merged into main branch.

- https://docs.gitlab.com/ee/user/project/merge_requests/

# Merge Request (cont.)



My awesome project
- Project information
- Repository
- Issues 0
- Merge requests 0
- CI/CD
- Security and Compliance
- Deployments
- Packages and registries
- Infrastructure
- Collapse sidebar

My awesome group › My awesome project › Merge requests › **New**

## New merge request

From `my-awesome-branch` into `main` Change branches

**Title (required)**

Update my_new_file

☐ Mark as draft
Drafts cannot be merged until marked ready.

**Description**

**Write** Preview

Describe the goal of the changes and what reviewers should be aware of.

Lawrence Livermore National Laboratory