

## Creating a simple CI/CD Pipeline using Gitlab

In this tutorial, you will learn everything you need to know about to start creating a simple CI/CD pipeline and automating your application deployment using GitLab.

### Prerequisite:

If you are new to GitLab, there are some few prerequisites or some knowledge that you need to know in order to understand this tutorial completely.

- Familiar with Git
- Understand what is CI/CD pipeline

Let us get started.

### What is GitLab?

GitLab is a DevOps platform that provides you with so many features including Version Control Systems, CI/CD, security, container registry and so much more that we are going to be learning in this tutorial.

In order to use all these features or use GitLab, you need to have a GitLab account. You can create a GitLab account using your email ID or using GitHub or using Bitbucket or salesforce. I already have my GitLab account created using GitHub and I am going to use that throughout this tutorial.

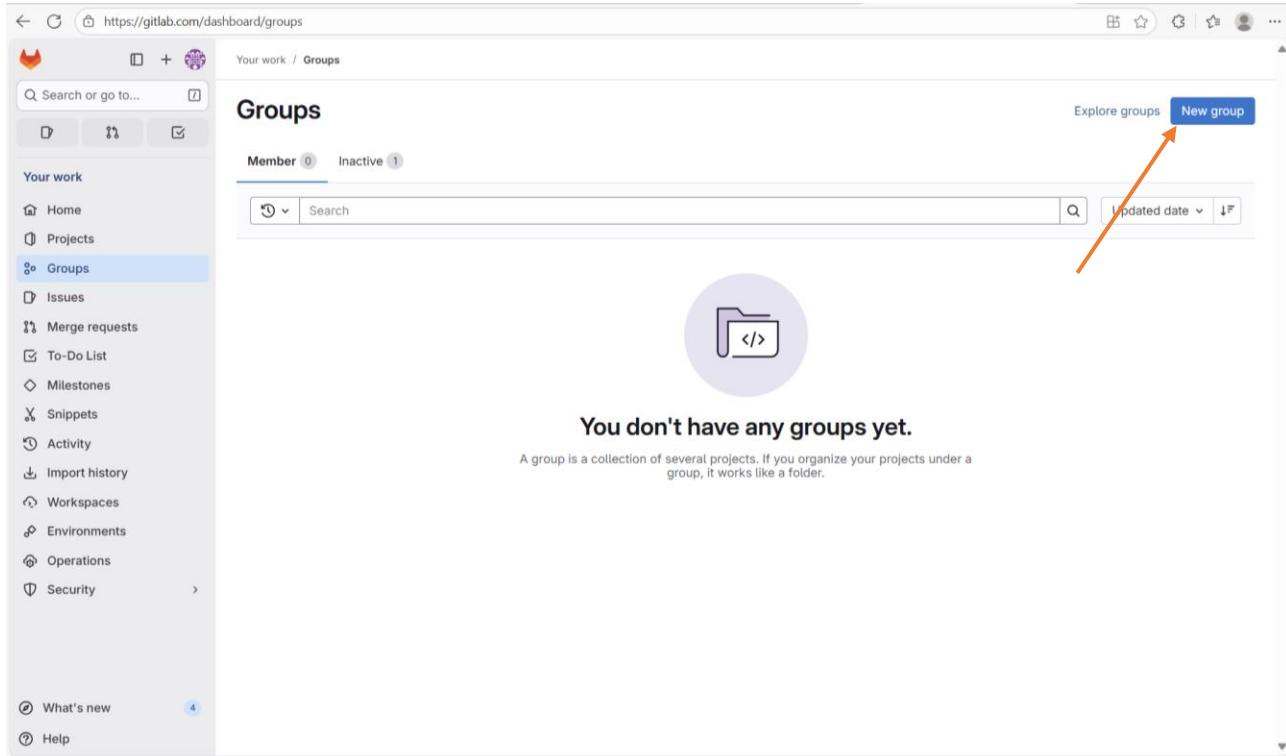
### Create Your First CI/CD Pipeline

Let us start creating our first CI/CD pipeline on GitLab. Once you have your account ID, Sign in on GitLab.

The screenshot shows the GitLab homepage with a light blue header. The top navigation bar includes links for Home, Help, Logout, and other account settings. Below the header, the main content area starts with a "Today's highlights" section featuring a purple circular icon and the greeting "Hi, Sidney". A prominent orange arrow points from the text "Groups" in the sidebar menu towards the "Groups" section in the main content area. The sidebar on the left lists several project management categories: Home, Projects, Groups (with an orange arrow pointing to it), Issues, Merge requests, To-Do List, Milestones, Snippets, Activity, Import history, Workspaces, Environments, Operations, Security, What's new (with a small '4' notification), and Help. The main content area displays a "Welcome to the new homepage" message, statistics for merge requests and issues, a "Items that need your attention" section with a green checkmark icon and the message "Good job! All your to-do items are done.", and a "Follow the latest updates" section showing a recent push to a branch. The overall interface is clean and modern, designed for easy navigation and management of multiple projects.

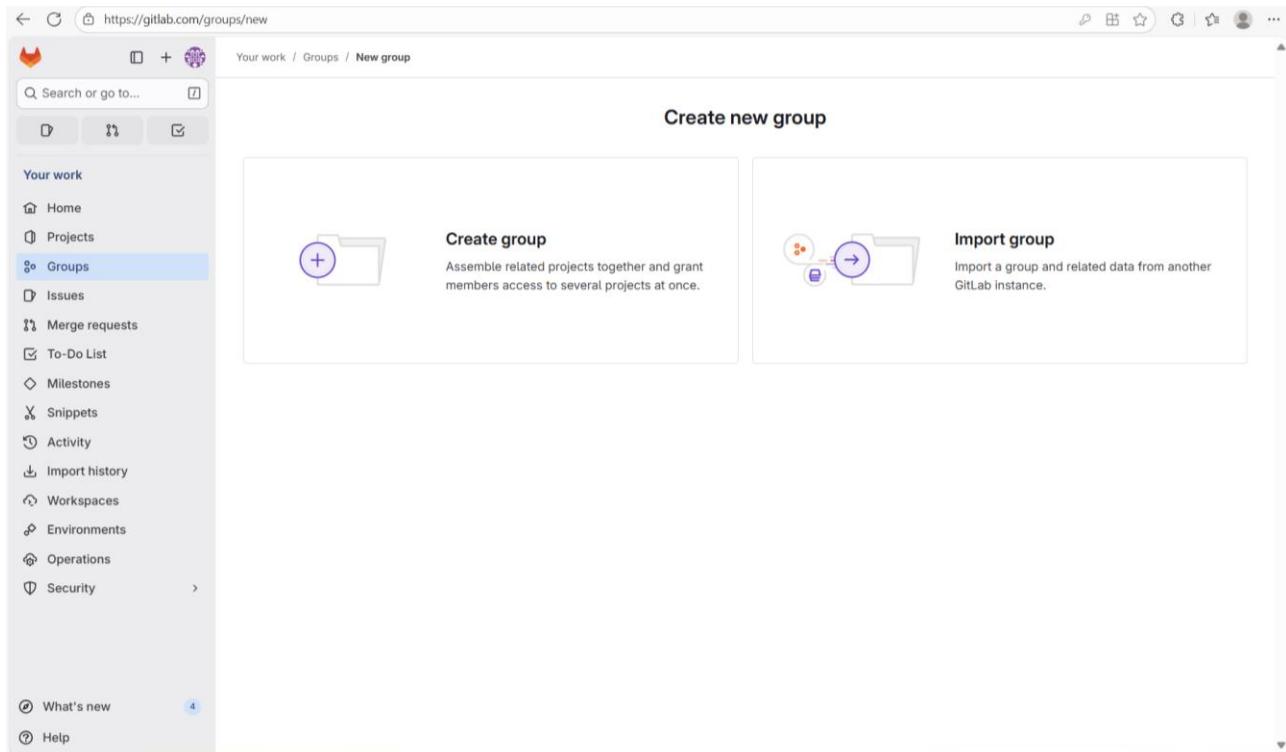
## STEP 1: Creating Group

We can create a group where we will store related projects. Click on “Groups” in the left-hand side.



The screenshot shows the GitLab dashboard with the 'Groups' section selected. A large orange arrow points to the 'New group' button in the top right corner of the main content area. The page displays a message: 'You don't have any groups yet.' with a subtext explaining what a group is.

Click on “New Group”



The screenshot shows the 'Create new group' page. It features two main options: 'Create group' and 'Import group'. The 'Create group' option is highlighted with a blue box and a larger font. The 'Import group' option is also shown with its own icon and description.

Click on “Create Group”

## Prepared by Sidney Smith

The screenshot shows the 'Create group' page on GitLab. The 'Group name' input field contains 'My group'. A red arrow points from the text 'Enter the group name, I will call it "sosoebot"' to this input field.

**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.

**You're creating a new top-level group**  
Members, projects, trials, and paid subscriptions are tied to a specific top-level group. If you are already a member of a top-level group, you can create a subgroup so your new work is part of your existing top-level group. Do you want to create a subgroup instead?  
[Learn more about subgroups](#)

**Group name**  
My group  
Must start with letter, digit, emoji, or underscore. Can also contain periods, dashes, spaces, and parentheses.  
⚠ Your group name must not contain a period if you intend to use SCIM integration, as it can lead to errors.

**Group URL**  
https://gitlab.com/ 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.

**Now, personalize your GitLab experience**  
We'll use this to help surface the right features and information to you.

Enter the group name, I will call it “**sosoebot**”

The screenshot shows the 'Create group' page on GitLab. The 'Group name' input field now contains 'sosoebot'. A red arrow points from the text 'Then on “Visibility Level” select “Public”' to the 'Public' radio button in the 'Visibility level' section.

**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.

**You're creating a new top-level group**  
Members, projects, trials, and paid subscriptions are tied to a specific top-level group. If you are already a member of a top-level group, you can create a subgroup so your new work is part of your existing top-level group. Do you want to create a subgroup instead?  
[Learn more about subgroups](#)

**Group name**  
sosoebot  
Must start with letter, digit, emoji, or underscore. Can also contain periods, dashes, spaces, and parentheses.  
⚠ Your group name must not contain a period if you intend to use SCIM integration, as it can lead to errors.

**Group URL**  
https://gitlab.com/ sosoebot

**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.

**Now, personalize your GitLab experience**  
We'll use this to help surface the right features and information to you.

Then on “**Visibility Level**” select “**Public**”

## Prepared by Sidney Smith

The screenshot shows the 'Create group' page on GitLab. The 'Groups' option is selected in the sidebar. The main form has the following fields:

- Group name:** sosoebot
- Group URL:** https://gitlab.com/sosoebot
- Visibility level:** Public (selected)
- Personalization:** A section titled 'Now, personalize your GitLab experience' with the note 'We'll use this to help surface the right features and information to you.'

Scroll down

The screenshot shows the 'Create group' page after scrolling down. The 'Visibility level' section is still visible. A red arrow points to the 'Who will be using this group?' section, which contains the following options:

- My company or team
- Just me

Below this, there is a dropdown menu for 'What will you use this group for?', a 'Invite Members (optional)' section with an email input field containing member1@company.com, and a 'Create group' button.

Then on "who will be using this group", I will select "Just me"

## Prepared by Sidney Smith

The screenshot shows the 'Create group' page on GitLab. The 'Group name' field contains 'sosoebot'. The 'Group URL' field contains 'https://gitlab.com/sosoebot'. Under 'Visibility level', the 'Public' option is selected. In the 'Now, personalize your GitLab experience' section, the 'Just me' radio button is selected. The 'Who will be using this group?' section shows 'Just me' selected. The 'What will you use this group for?' dropdown menu is open, displaying several options. An orange arrow points to the first option: 'I want to learn the basics of Git'.

On “**What will you use this group for**”, I will click on the drop down

The screenshot shows the 'Create group' page on GitLab. The 'Group name' field contains 'sosoebot'. The 'Group URL' field contains 'https://gitlab.com/sosoebot'. Under 'Visibility level', the 'Public' option is selected. In the 'Now, personalize your GitLab experience' section, the 'Just me' radio button is selected. The 'Who will be using this group?' section shows 'Just me' selected. The 'What will you use this group for?' dropdown menu is open, displaying several options. An orange arrow points to the first option: 'I want to learn the basics of Git'.

Will select “**I want to learn the basics of Git**”

Your work / Groups / New group / Create group

Group name  
sosoebot

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

⚠ Your group name must not contain a period if you intend to use SCIM integration, as it can lead to errors.

Group URL  
https://gitlab.com/ sosoebot

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.

Now, personalize your GitLab experience

We'll use this to help surface the right features and information to you.

Who will be using this group?

My company or team  Just me

What will you use this group for?

I want to learn the basics of Git

Create group Cancel

Then click on “Create Group”

sosoebot

① Group sosoebot was successfully created.

S sosoebot

New subgroup New project

Subgroups and projects Shared projects Shared groups Inactive

Search (3 character minimum)

There are no subgroups or projects in this group

Create subgroup Use groups to manage multiple projects and members.

Create project Use projects to store and access issues, wiki pages, and other GitLab features.

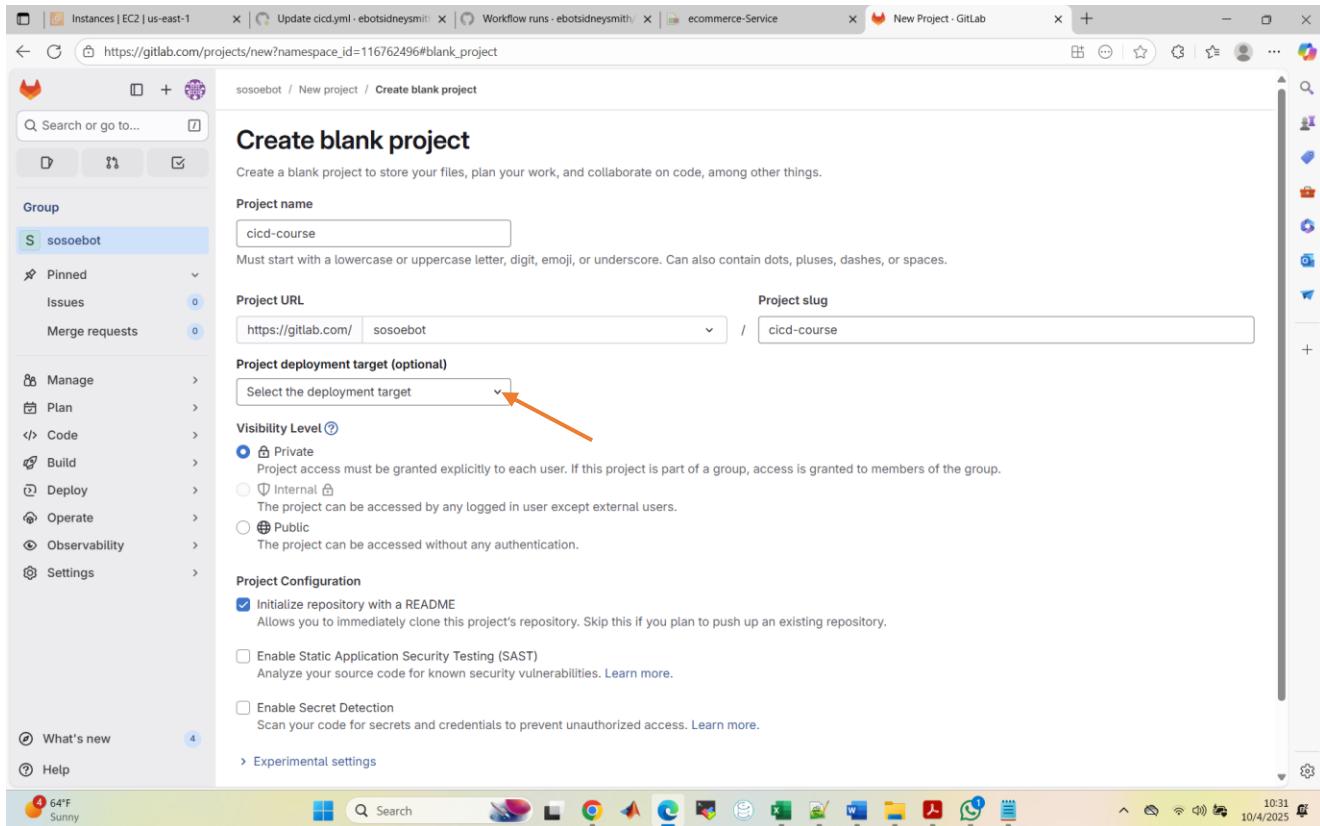
The group has been created. Let us now create a project in the group. Let us start creating our first CI/CD Pipeline. Click on “New Project”

The screenshot shows the 'Create new project' page on GitLab. On the left, there's a sidebar with a 'Group' section containing 'sosoebot' (Pinned: Issues 0, Merge requests 0), 'Manage' (Plan, Code, Build, Deploy, Operate, Observability, Settings), and links for 'What's new' (4) and 'Help'. The main area has a title 'Create new project' and four cards: 'Create blank project' (Icon: plus sign in a box), 'Create from template' (Icon: plus sign in a box), 'Import project' (Icon: code editor and document), and 'Run CI/CD for external repository' (Icon: gear with arrows). An orange arrow points to the 'Create blank project' card.

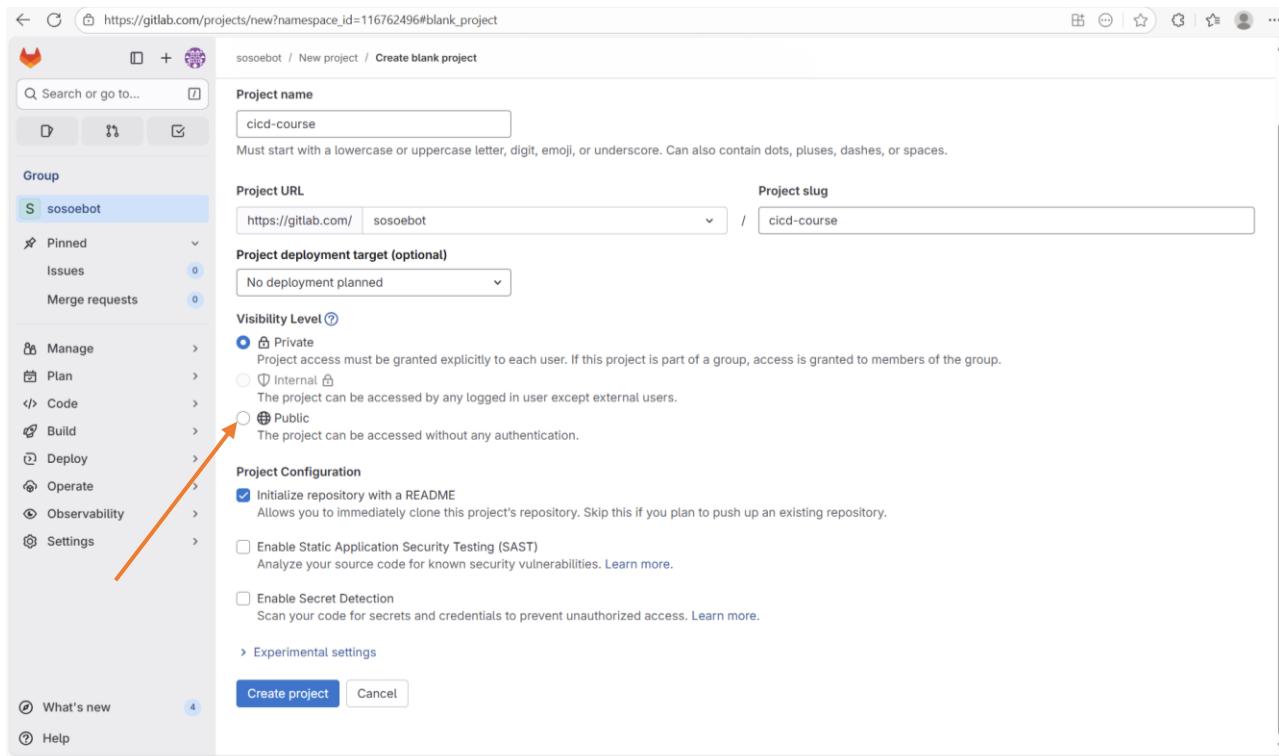
Click on “Create Blank Project”

The screenshot shows the 'Create blank project' configuration page. The left sidebar shows 'Your work' with 'Projects' selected. The main form has a title 'Create blank project' with the sub-instruction 'Create a blank project to store your files, plan your work, and collaborate on code, among other things.' It includes fields for 'Project name' (containing 'My awesome project', with a red arrow pointing to it), 'Project URL' ('https://gitlab.com/sosoebot'), 'Project slug' ('my-awesome-project'), 'Project deployment target (optional)' (dropdown menu), 'Visibility Level' (radio buttons for Private, Internal, and Public, with Private selected), and 'Project Configuration' (checkboxes for 'Initialize repository with a README' (checked) and 'Enable Static Application Security Testing (SAST)'). At the bottom are 'Create project' and 'Cancel' buttons.

Let us name the Project as “cicd-course”.

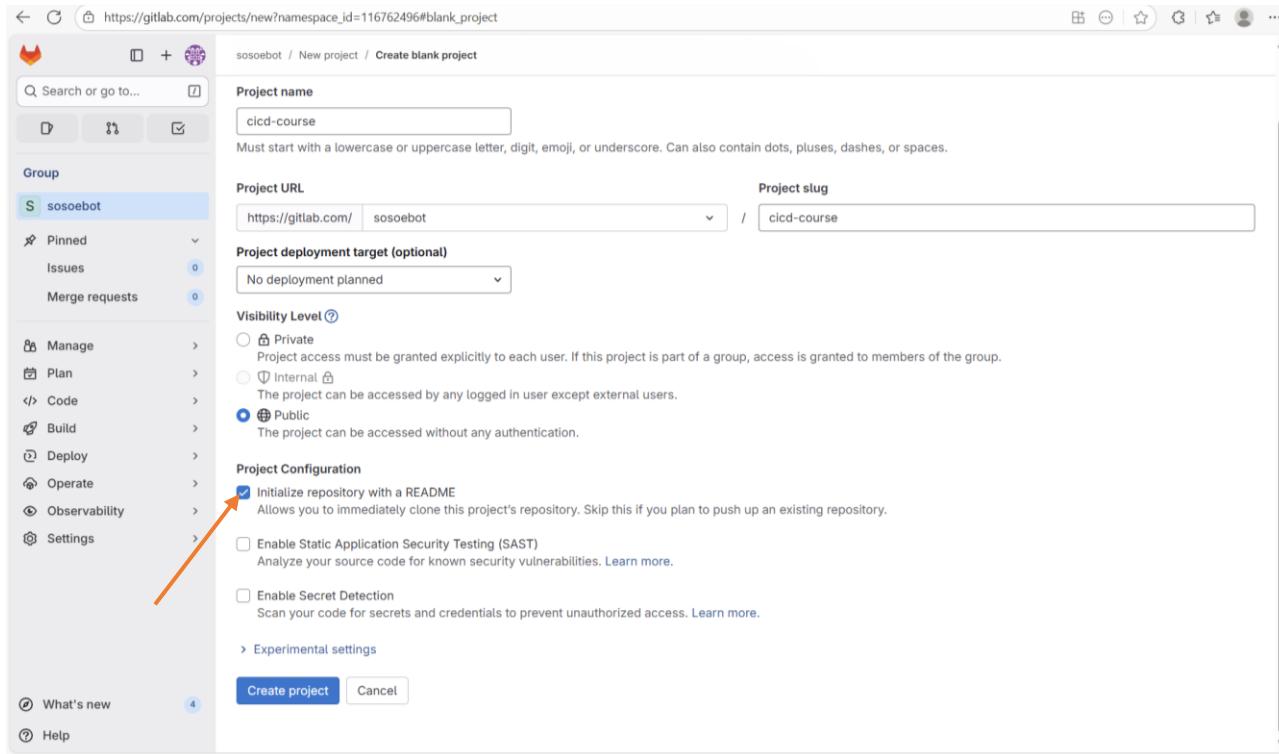


For deployment, we do not have any deployment plan. So, click on the drop down and choose “**No Deployment Planned**”.



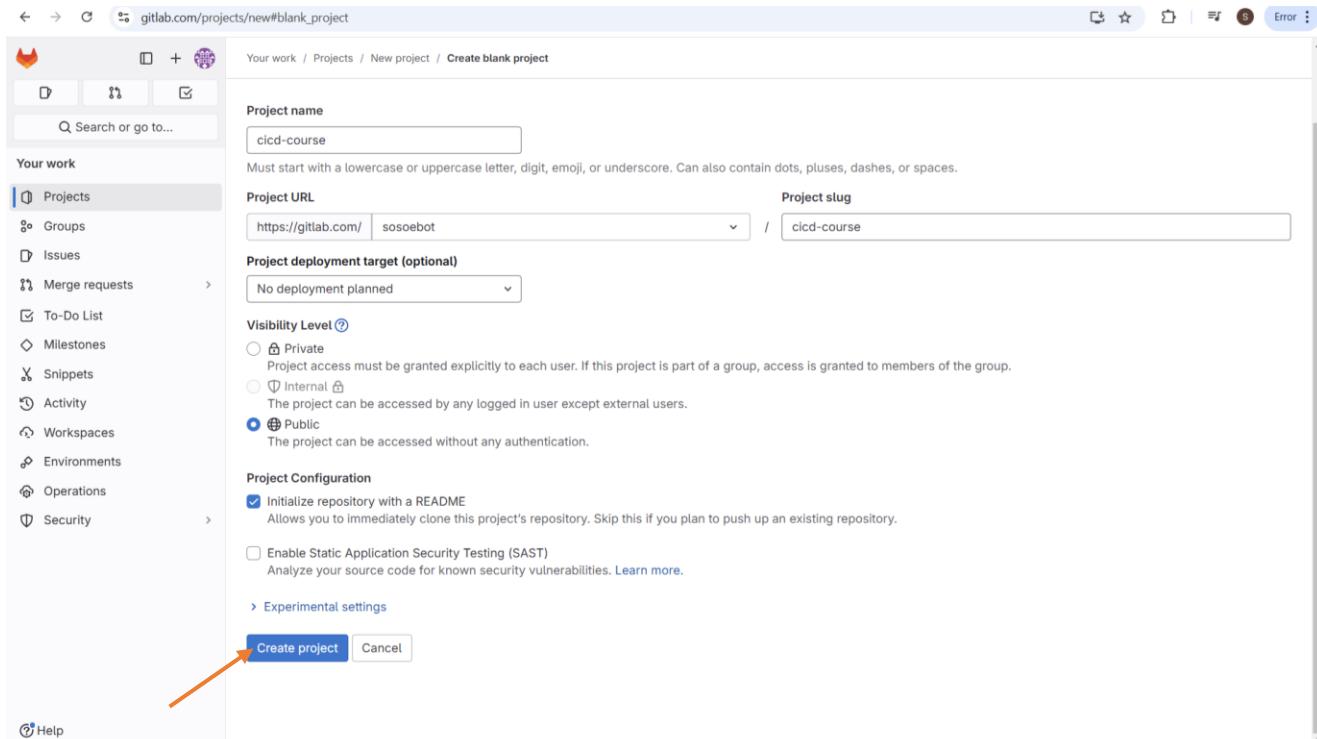
For visibility, we will make it “**public**”.

## Prepared by Sidney Smith



The screenshot shows the 'Create blank project' form on the GitLab website. The 'Project name' field contains 'cicd-course'. The 'Project URL' field shows 'https://gitlab.com/' followed by 'sosoebot'. The 'Project slug' field also contains 'cicd-course'. Under 'Visibility Level', the 'Public' option is selected. In the 'Project Configuration' section, the 'Initialize repository with a README' checkbox is checked, with a note explaining it allows immediate cloning. Other options like 'Enable Static Application Security Testing (SAST)' and 'Enable Secret Detection' are available but unchecked. A red arrow points to the 'Initialize repository with a README' checkbox. At the bottom are 'Create project' and 'Cancel' buttons.

Also, we want to Initialize repository with a **README**, so check the box "**Initialize Repository with a README**".



This screenshot shows the same 'Create blank project' form as above, but with a red arrow pointing to the 'Create project' button at the bottom. The rest of the interface is identical to the first screenshot.

Click on "**Create Project**", this will create a repository in GitLab.

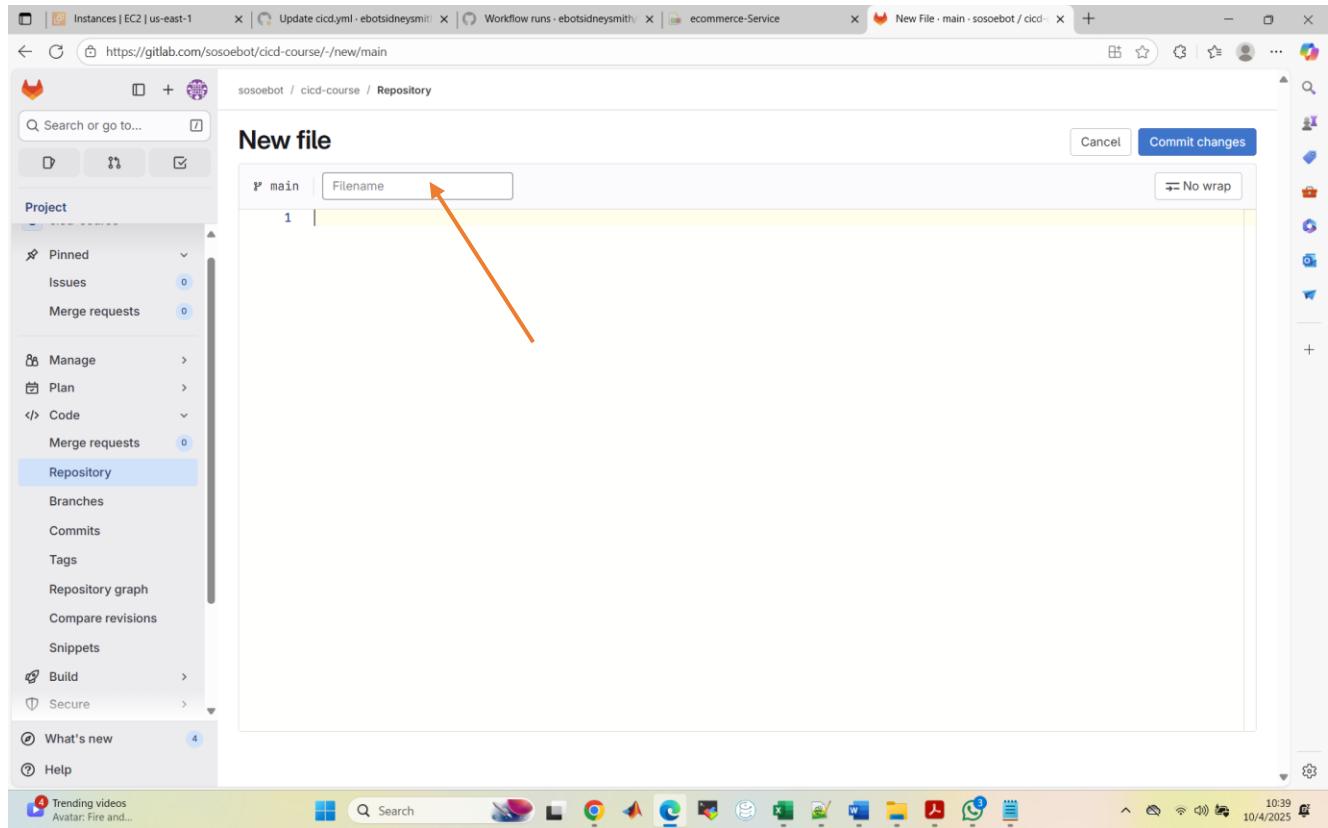
The screenshot shows a GitLab repository named 'cicd-course'. The sidebar on the left contains project management options like Pinned, Issues, Merge requests, and various security and deployment tools. The main area displays a single commit from 'Initial commit' by 'Sidney Smith Ebot' just now. The commit hash is '9d8e64c1'. Below the commit, there's a table showing the README.md file with its name, last commit, and last update. To the right, the 'Project information' section shows 1 Commit, 1 Branch, 0 Tags, and 4 KiB Project Storage. A sidebar on the right lists repository-related actions such as README, Add LICENSE, Add CHANGELOG, Add CONTRIBUTING, Enable Auto DevOps, Add Kubernetes cluster, Set up CI/CD, Add Wiki, and Configure Integrations. A red arrow points from the '+ New file' button in the top right to the 'Set up CI/CD' button in the sidebar.

The repository has been created. We are inside CI/CD repository and to set up a CI/CD pipeline for this particular repo, we need to create a file that will hold all the configurations inside it.

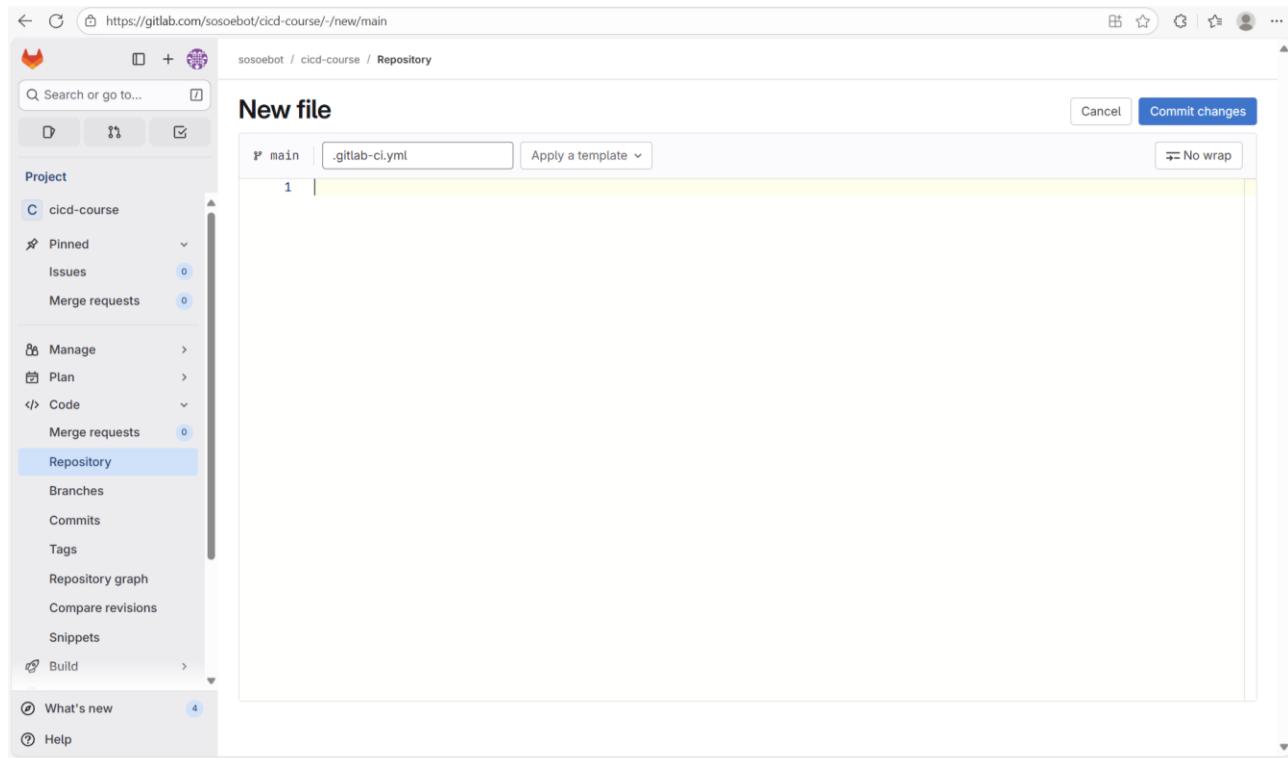
I can do that by clicking on the “set-up CI/CD” button but this will make things complex. So, instead click on the drop down on “+”

This screenshot is identical to the previous one, showing the 'cicd-course' repository on GitLab. The red arrow now points from the 'New file' option in the dropdown menu that appears when clicking the '+' button in the top right to the 'New file' option listed in the sidebar under 'Project information'.

And select “New File”



This file will hold all the CI/CD configuration. In GitLab if you want to create a pipeline which will hold all the CI/CD configuration, it should be named as **".gitlab-ci.yml"**.



This is the name you give to file that will hold the CI/CD pipeline configuration because GitLab will understand only if you use this name, unless you change the name in the settings.

You can also check the documentation on <https://docs.gitlab.com/ee/ci/>

The screenshot shows the GitLab documentation website for the Enterprise Edition (ee). The URL in the address bar is <https://docs.gitlab.com/ee/ci/>. The page title is "Get started with GitLab CI/CD". On the left, there is a sidebar with a navigation tree. Under the "Getting started" section, the "Use CI/CD to build your application" option is selected. The main content area contains text about CI/CD being a continuous method of software development, iterative code changes, and a process part of a larger workflow. It features a diagram illustrating the CI/CD pipeline stages: Plan, Create, Verify, Secure, Release, and Monitor. Below the diagram, a section titled "Step 1: Create a .gitlab-ci.yml file" explains that it starts with a ".gitlab-ci.yml" file at the root of the project, specifying stages, jobs, and scripts. A note states: "In this file, you define variables, dependencies between jobs, and specify when and how each job should be". To the right, a "On this page" sidebar lists five steps: Step 1: Create a .gitlab-ci.yml file, Step 2: Find or create runners, Step 3: Define your pipelines, Step 4: Use CI/CD variables as part of jobs, and Step 5: Use CI/CD components.

If you are new to GitLab CI/CD, start by reviewing some of the commonly used terms. The first is **gitlab-ci.yml** file. To use GitLab CI/CD, you start with “**.gitlab-ci.yml**” file at the root of your project which contains the configuration for a CI/CD pipeline and this pipeline uses yaml.

We are going to name our CI/CD file as “**.gitlab-ci.yml**”, inside this file you are going to create jobs. Job can be different types depending on what you want to do. Let us create a simple build job which is going to execute a sample script.

Going back to the documentation, again <https://docs.gitlab.com/ee/ci/jobs/>

Pipeline configuration begins with jobs. **Jobs** are the most fundamental element of a “**.gitlab-ci.yml**” file. It is used to execute a particular script.

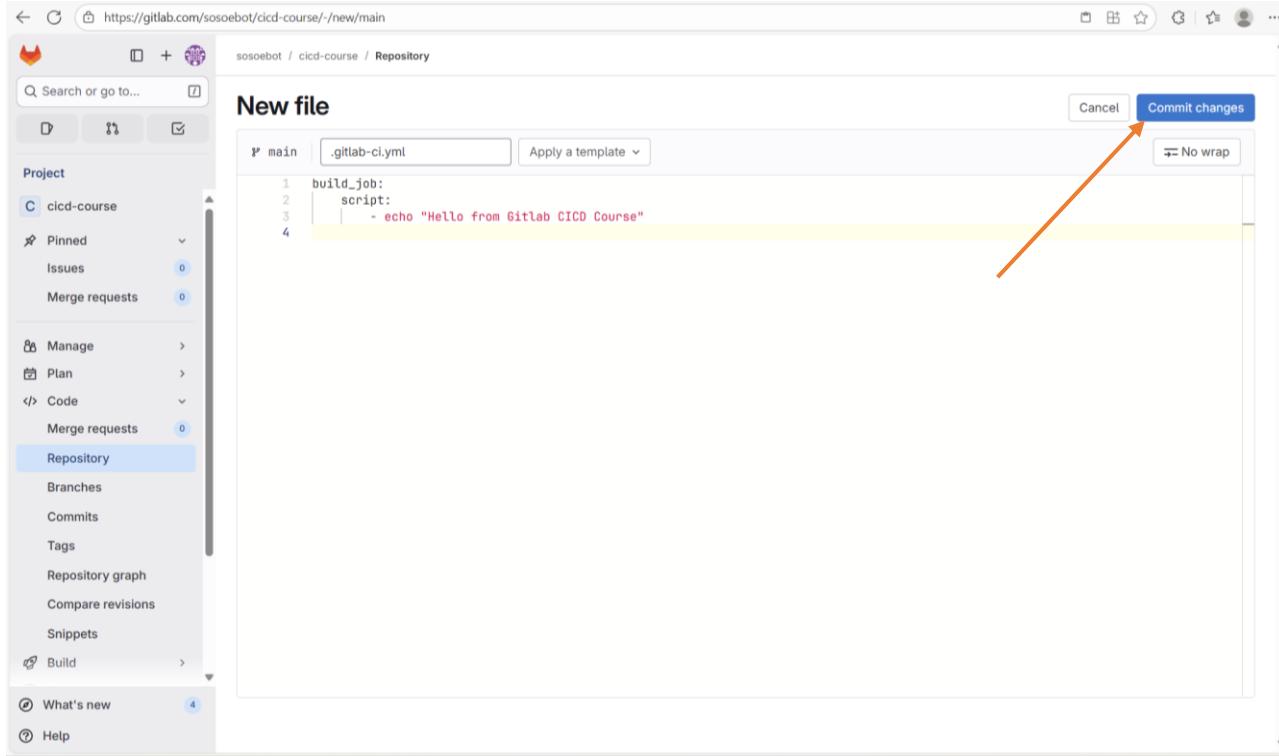
For example:

```
my-ruby-job:  
  script:  
    - bundle install  
    - bundle exec my_ruby_command
```

```
my-shell-script-job:  
  script:  
    - my_shell_script.sh
```

You can see we have two jobs in this example, namely “**my-ruby-job**” and “**my-shell-script-job**”. Job “**my-ruby-job**” is executing the script to **bundle install** and **bundle exec my\_ruby\_command**. While “**my-shell-script-job**” is executing script to the “**my\_shell\_script.sh**” file.

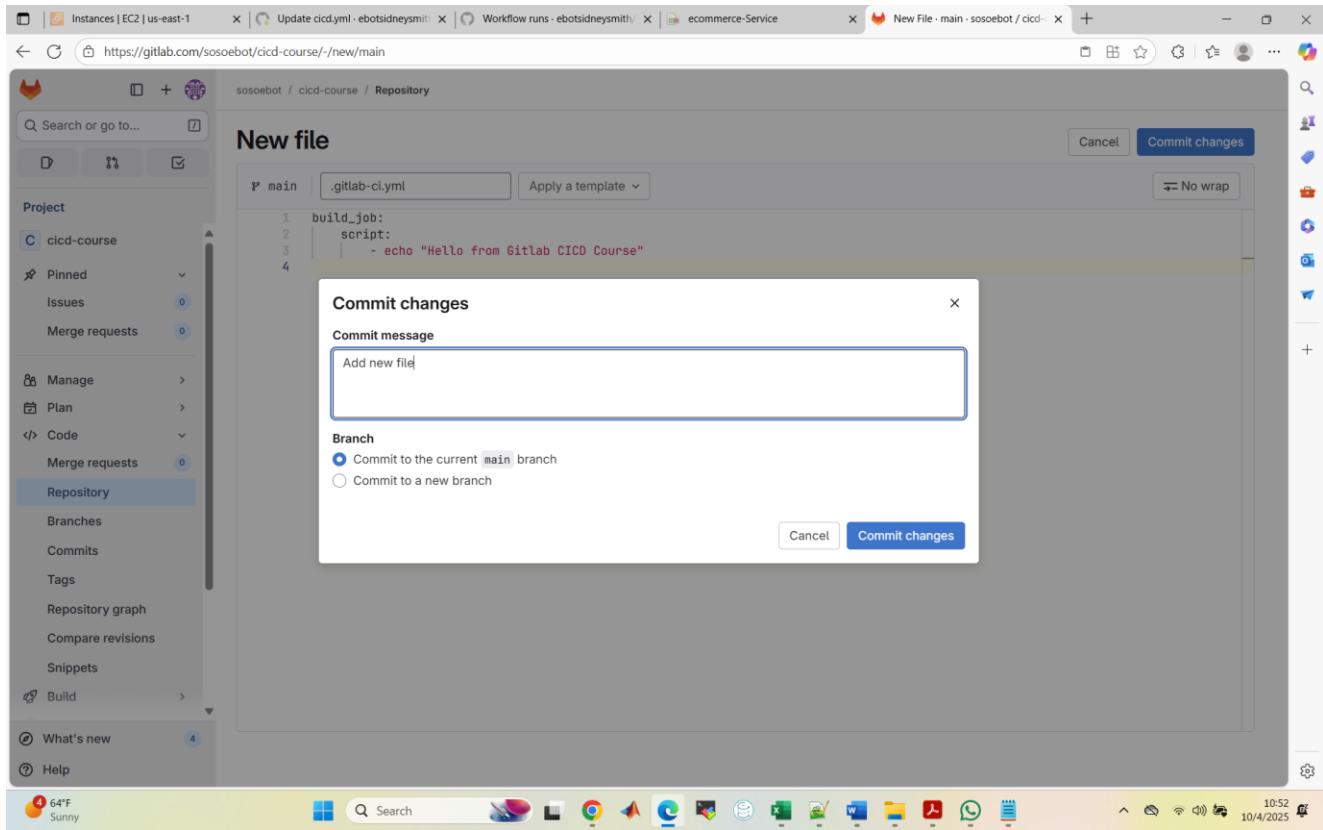
Similarly, we are going to create a simple job which will execute a script that will show us a message on our pipeline.



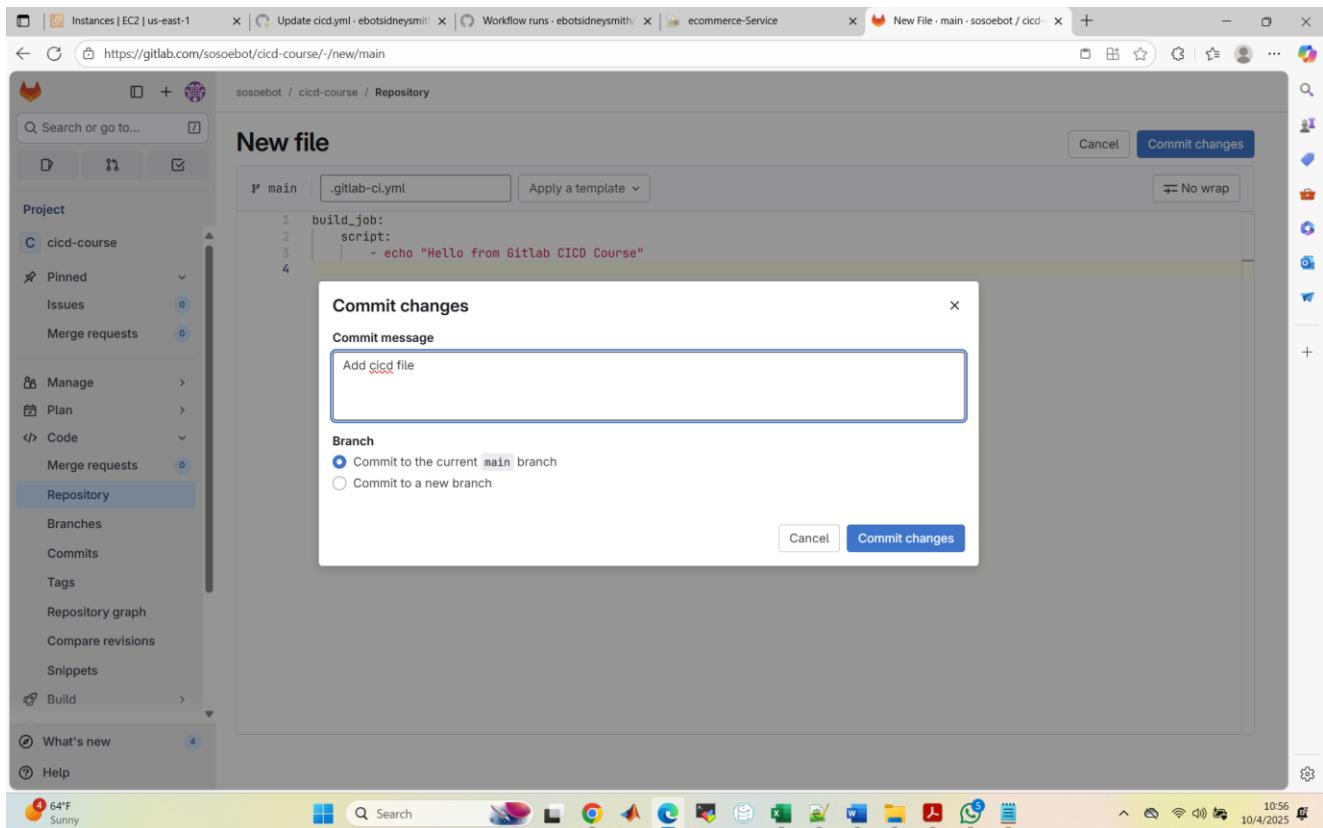
This is our CI/CD pipeline configuration now.

Now, we are going to **commit** it and let us see if our pipeline is going to succeed and show us this message of not. Click on “**Commit Changes**”

## Prepared by Sidney Smith



Add the message “Add cicd file”



Then click on “Commit Changes” again. This will be committed to my main branch.

## Prepared by Sidney Smith

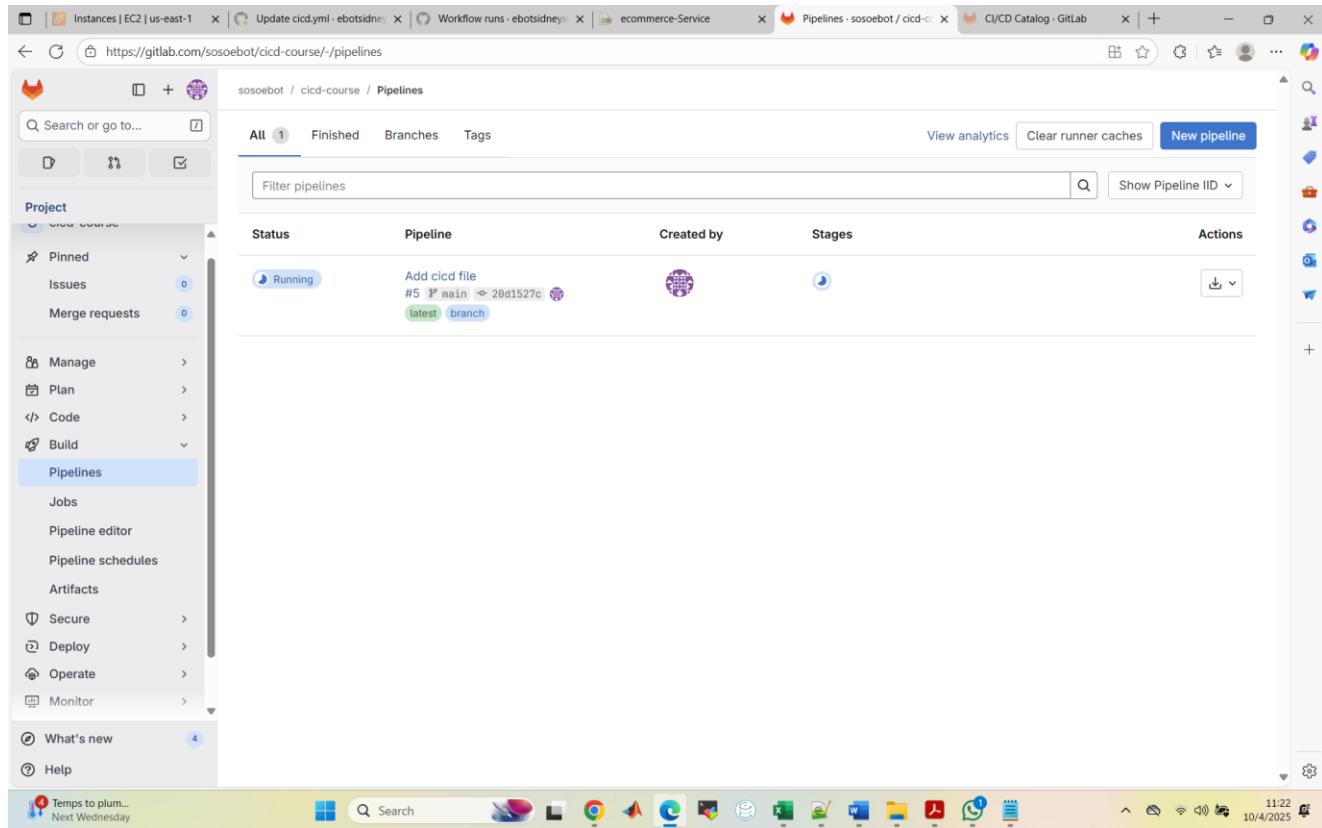
The screenshot shows a GitLab repository page for 'cicd-course'. In the top right, there's a success message: 'Your changes have been committed successfully.' Below it, the '.gitlab-ci.yml' file is shown with a validation message: '✓ This GitLab CI configuration is valid. Learn more'. A red arrow points from the text 'Once you do this; you can go **build section** and click on **pipeline**' to the 'Build' section in the sidebar. Another red arrow points to the 'Pipelines' option under the 'Build' section.

Gitlab will also provide you with a validation message if the CI/CD pipeline is valid or not, if there is syntax issues or not. Once you do this; you can go **build section** and click on **pipeline**

This screenshot is similar to the previous one but focuses on the 'Build' section of the sidebar. The 'Pipelines' option is highlighted with a red arrow. Another red arrow points to the 'Pipelines' option under the 'Build' section. The main area shows the same CI configuration and validation message as the previous screenshot.

Click on “**Pipelines**”, this will show you if the pipeline is running or not.

## Prepared by Sidney Smith



sosobot / cicd-course / Pipelines

All 1 Finished Branches Tags

Status Pipeline Created by Stages Actions

Running Add cicd file #5 P main > 20d1527c 🌐 latest branch

Filter pipelines

View analytics Clear runner caches New pipeline

Project cicd-course

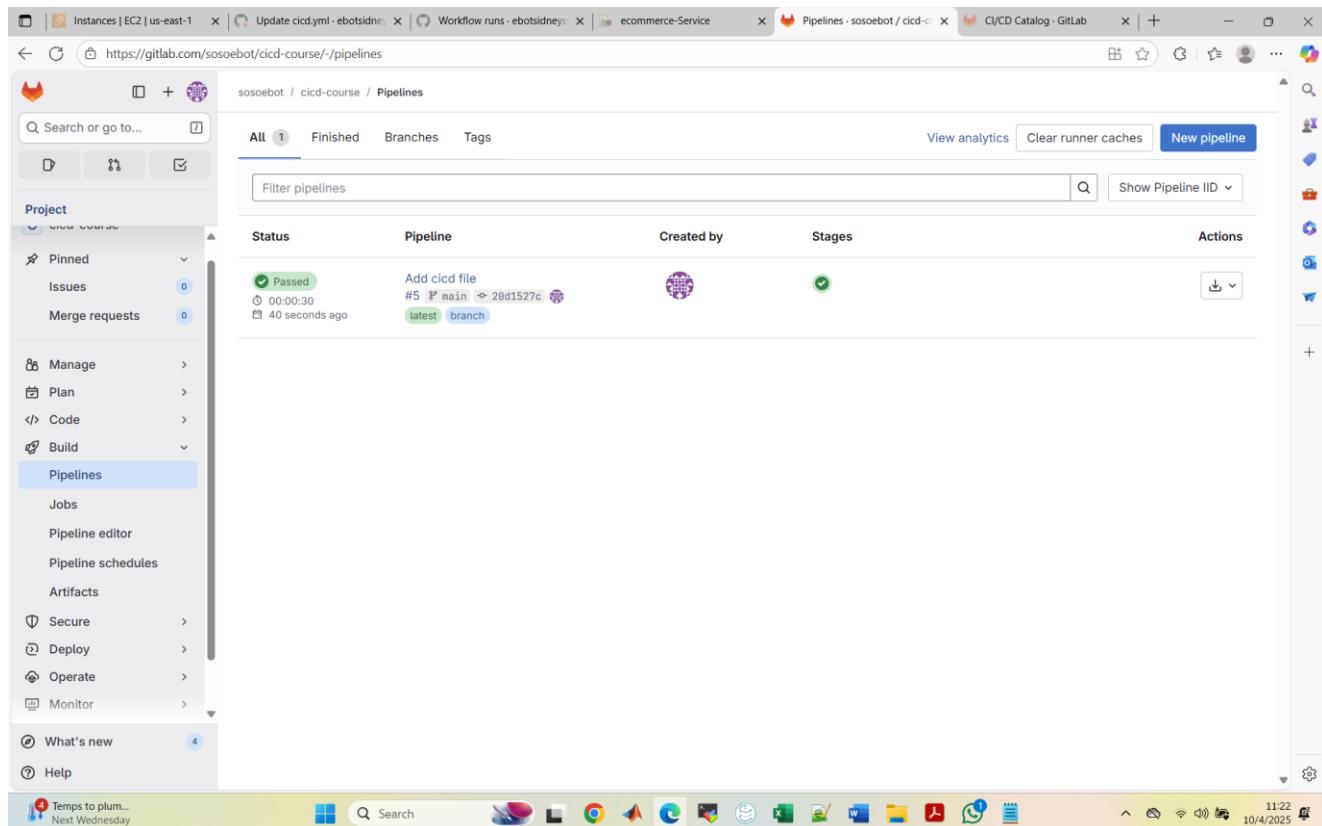
Pinned Issues Merge requests

Manage Plan Code Build Pipelines Jobs Pipeline editor Pipeline schedules Artifacts Secure Deploy Operate Monitor

What's new 4 Help

Temps to plum... Next Wednesday

You can see that the pipeline is running



sosobot / cicd-course / Pipelines

All 1 Finished Branches Tags

Status Pipeline Created by Stages Actions

Passed Add cicd file #5 P main > 20d1527c 🌐 latest branch 00:00:30 40 seconds ago

Filter pipelines

View analytics Clear runner caches New pipeline

Project cicd-course

Pinned Issues Merge requests

Manage Plan Code Build Pipelines Jobs Pipeline editor Pipeline schedules Artifacts Secure Deploy Operate Monitor

What's new 4 Help

Temps to plum... Next Wednesday

You can see that the pipeline is running and the status is “Passed”. The status can be “passed”, “Pending” or it can be “Failed”. You can also see who has initiated this pipeline.

The screenshot shows the GitLab Pipelines interface for the project 'cicd-course'. A single pipeline is listed under the 'All' tab. The pipeline details are as follows:

- Status:** Passed (green checkmark)
- Pipeline ID:** #1
- Branch:** main
- Commit SHA:** 5b9b7f46
- Created:** 30 minutes ago
- Stages:** test (green checkmark)

An orange arrow points from the 'Created by' field ('Sidney Smith Ebot') to the 'Passed' status indicator.

You can see that it is created by “**Sidney Smith Ebot**”.

The screenshot shows the same GitLab Pipelines interface. The pipeline status is now explicitly labeled as 'test: passed' (green checkmark). An orange arrow points from the 'Passed' status indicator to the 'test: passed' label.

Right now, we only have a single stage which is the “**test**” stage, by default all the jobs will run in the test stage. And you can see it says “**Passed**”

Click on the status “**Passed**”, it will bring you inside

The screenshot shows the detailed view of pipeline #1530466868. The pipeline status is 'Passed' (green checkmark). The pipeline details are:

- For:** main
- Latest Job:** build\_job (1 job, 0.5 seconds, queued for 1 seconds)

The pipeline has one stage named 'test' containing one job named 'build\_job'. An orange arrow points from the 'Jobs 1' link to the 'build\_job' entry.

You can see there is a single job, click on “**Job**”

The screenshot shows the GitLab pipeline interface for a project named 'cicd-course'. A single job, 'build\_job', is listed under the 'main' branch. The job status is 'Passed' with a duration of 0:00:30 and was run 44 minutes ago. An orange arrow points from the text 'If you click on the job, you will get more information about the job.' to the job name 'build\_job' in the list.

You can see the single job. If you click on the job, you will get more information about the job.

The screenshot shows the detailed logs for the 'build\_job' on the 'main' branch. The logs include steps like preparing the Docker executor, pulling the Docker image, and executing a step script. One specific log entry, 'Hello from CICD Course', is highlighted with an orange arrow. The right side of the screen displays summary statistics for the job, such as duration, runner details, and related jobs.

This is where you will see the logs. Let me try to explain to you here so the log has few things the first thing is it running on GitLab runner. GitLab provides you with few shared runners that you can use.

```
1 Running with gitlab-runner 17.4.0~pre.110.g27400594 (27400594)
2 on blue-5.saas-linux-small-amd64.runners-manager.gitlab.com/default -AzERasQ, system ID: s_4cb09cee29e2
3 Preparing the "docker-machine" executor
```

**Runner** is a machine which executes the scripts that you define in your job, we are going learn about **“Runners”** very soon. So, you just need to understand that runners are machines which execute your script that you define in different jobs.

Next, you can see a build has been created using Docker Image which is of **Ruby image**. So, the image used to execute this particular script is **Ruby**.

```
2 On host 0.5.6.85 CLOUD SMALL AZERASQ-PROJECTS-MANAGER.GITLAB.COM/default - AZERASQ, System ID: 0_46867662762
3 Preparing the "docker+machine" executor
4 Using Docker executor with image ruby:3.1 ...
5 Pulling docker image ruby:3.1 ...
6 Using docker image sha256:243309b48f4ab04a5de198e1ef7ec8b224aa96924f096f188dd6c616c3a71233 for ruby:3.1 with digest ruby@sha256:b
a4d592a4fc6e3f5d2a9a52a1a3bbefde53308e786de4f66ba72237b18b15676 ...
```

Next, it is preparing an environment for our job. It is getting the source from repository executing the script.

```
7 Preparing environment
8 Running on runner--azerasq-project-64291394-concurrent-0 via runner-azerasq-s-l-s-amd64-1730927391-c4de59aa...
```

Lastly, we have a message “Hello from CICD Course”

```
18 $ echo "Hello from CICD Course"
19 Hello from CICD Course
20 Cleaning up project directory and file based variables
21 Job succeeded
```

This means our job has been succeeded and our pipeline has run successfully. You can see more information about the job here.

**Duration:** 30 seconds  
**Finished:** 5 hours ago  
**Queued:** 0 seconds  
**Timeout:** 1h (from project) [?](#)  
**Runner:** #12270840 (-AzERasQ) 5-blue.saas-linux-small-amd64.runners-manager.gitlab.com/default

---

**Commit** [5b9b7f46](#)

Add cicd file

---

**Pipeline** [#1530466868](#) Passed for **main**

---

**Related jobs**

→ [build\\_job](#)

The job ran for 30 seconds, it has finished in 5 hours. There was no queue. The timeout for this particular job is 1 hour by default. The runner is a particular runner used for this particular job. The commit ID is **5b9b7f46**, which ran the pipeline. And you also see the different stages, right now it is only test.

We are also going to learn about stages very soon. Now, we have successfully created a simple CI/CD pipeline. Now, if you want to rerun this particular job, you can just click on this **Retry** option.



**Duration:** 30 seconds  
**Finished:** 5 hours ago  
**Queued:** 0 seconds  
**Timeout:** 1h (from project) [?](#)  
**Runner:** #12270840 (-AzERasQ) 5-blue.saas-linux-small-amd64.runners-manager.gitlab.com/default

**Commit** [5b9b7f46](#)

Add cicd file

**Pipeline** [#1530466868](#) Passed for main

Related jobs

→ [build\\_job](#)

If, I click on this Retry option, it will go ahead and run the job.



**Elapsed time:** 1 second  
**Queued:** 0 seconds  
**Timeout:** 1h (from project) [?](#)  
**Runner:** #12270835 (zxwgkjAP) 3-blue.saas-linux-small-amd64.runners-manager.gitlab.com/default

**Commit** [5b9b7f46](#)

Add cicd file

**Pipeline** [#1530466868](#) Running for main

Related jobs

→ [build\\_job](#)

[build\\_job](#)

You can see the job is running again. It is in the running state right now and after this is done, it will tell me if it is pass or fail. Obviously, it will be passed because we haven't made any change or there is no error. This how you create a simple CI/CD pipeline in GitLab.