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.

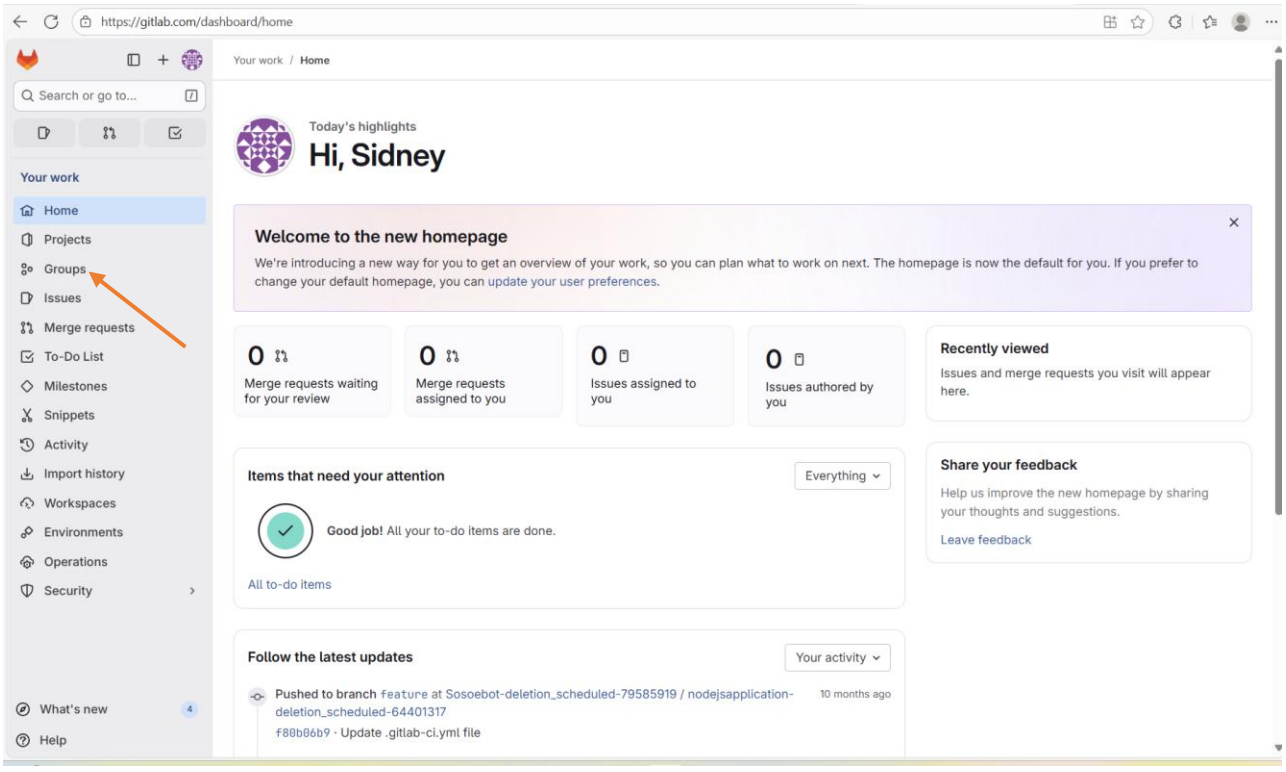
What is GitLab?

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.

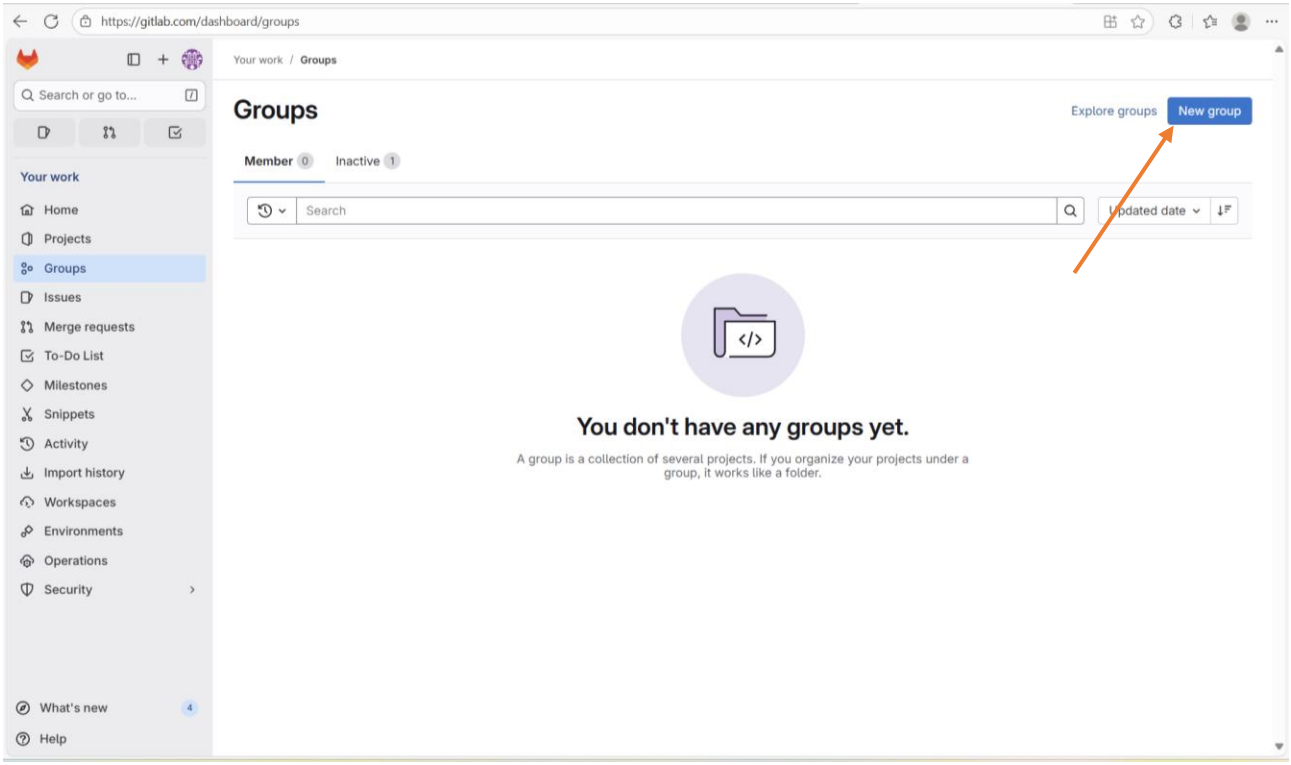
Create Your First CI/CD Pipeline

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

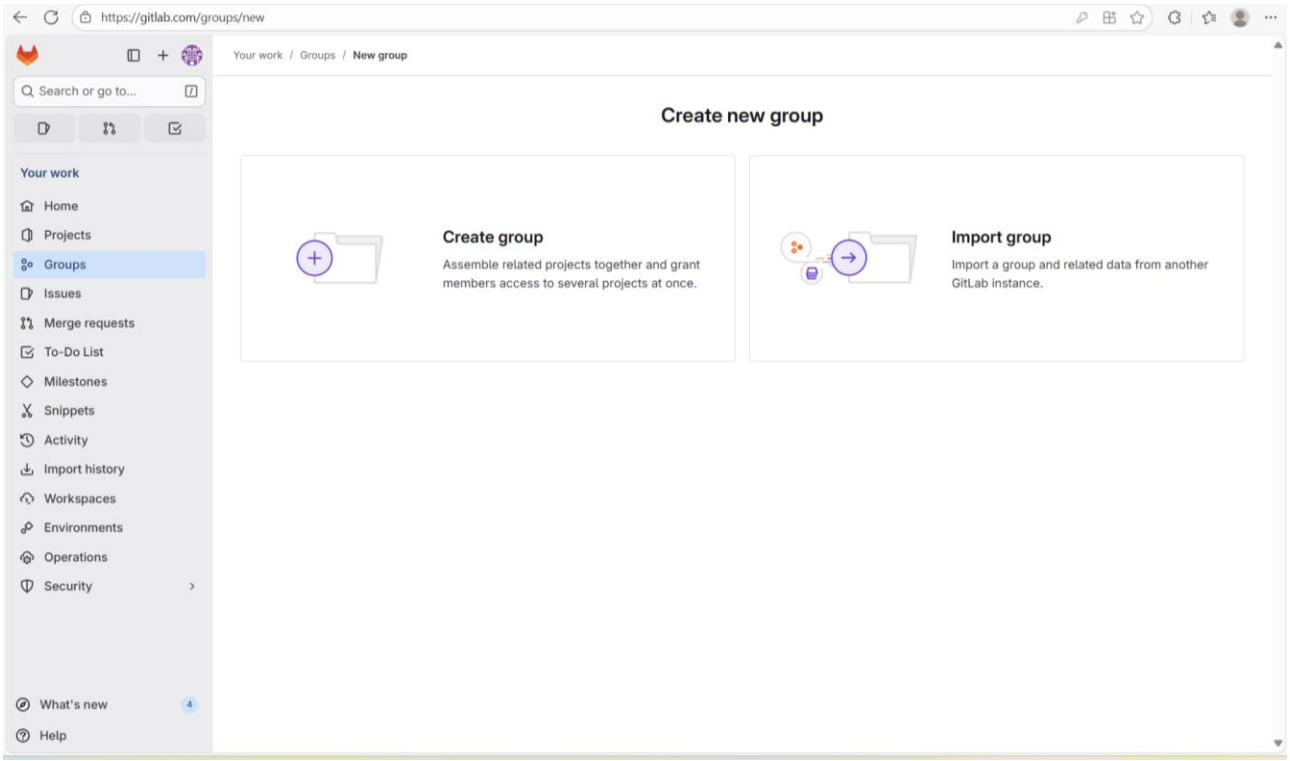


STEP 1: Creating Group

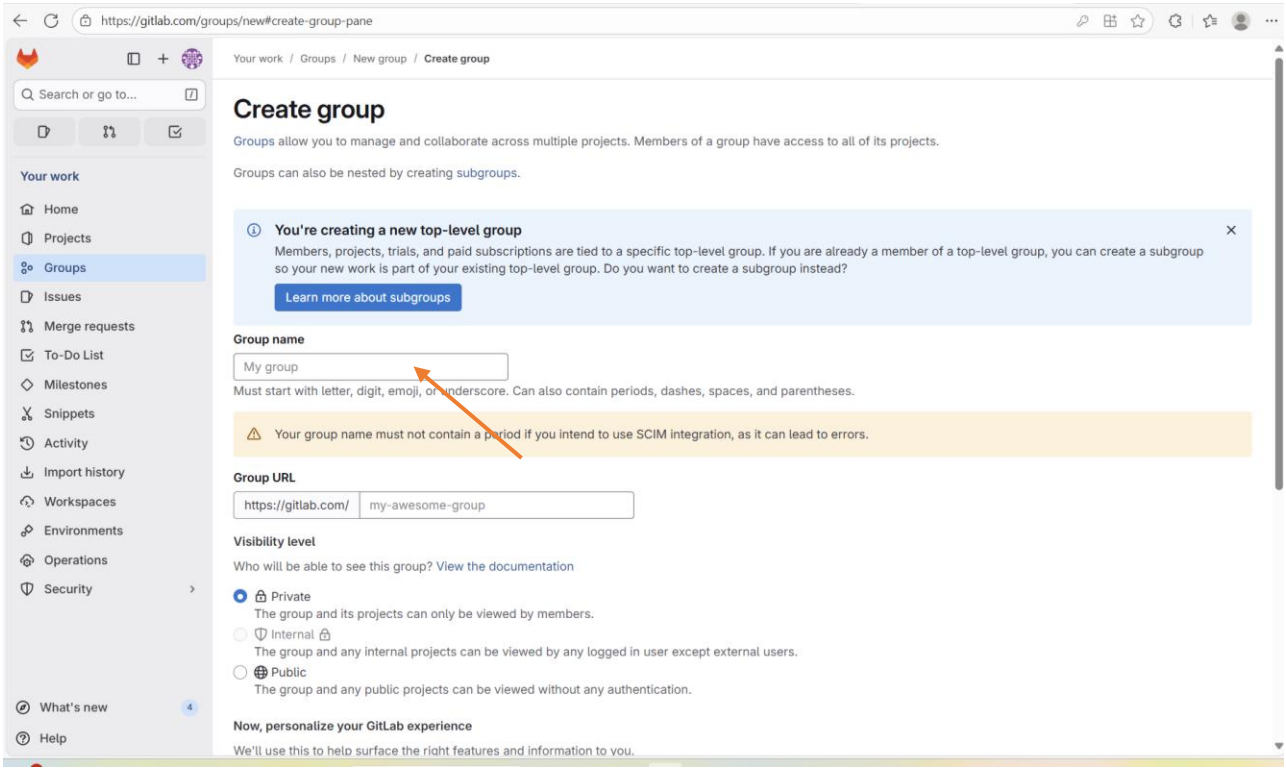
We can create a group where we will store related projects. Click on "**Groups**" in the left-hand side.



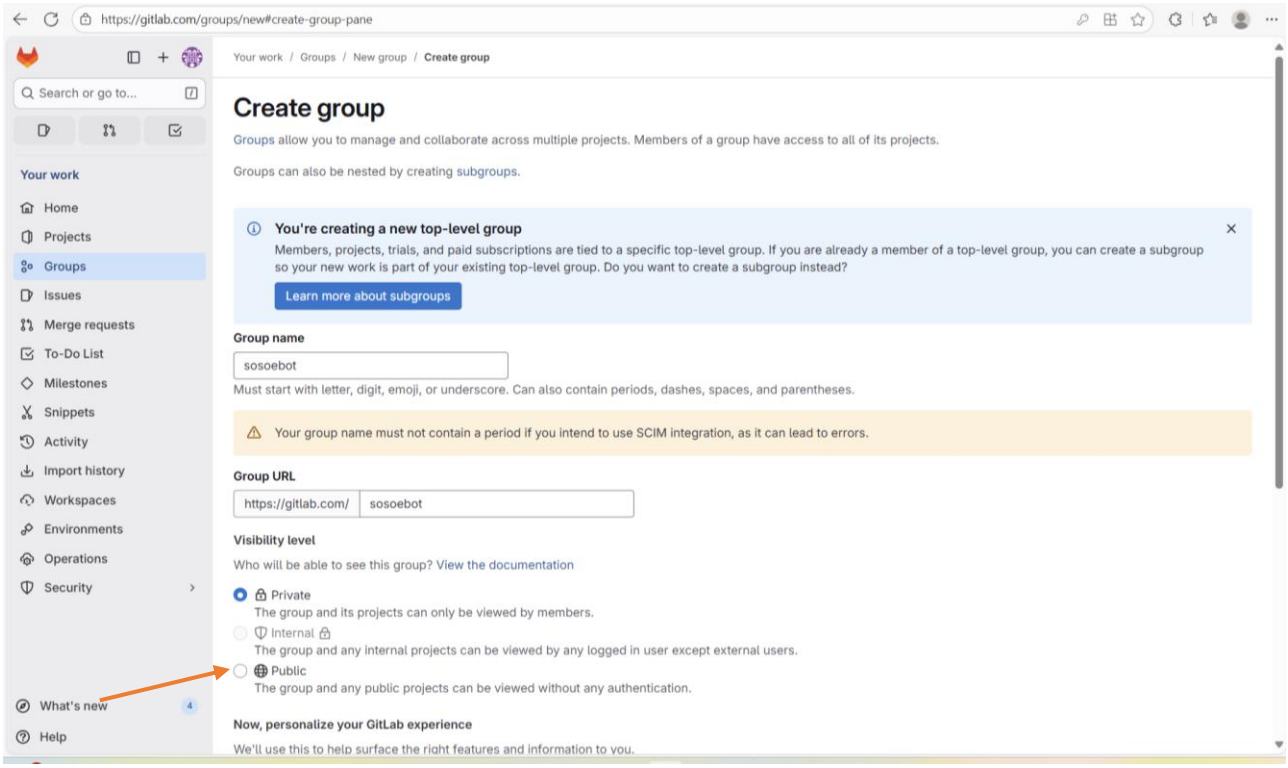
Click on "**New Group**"



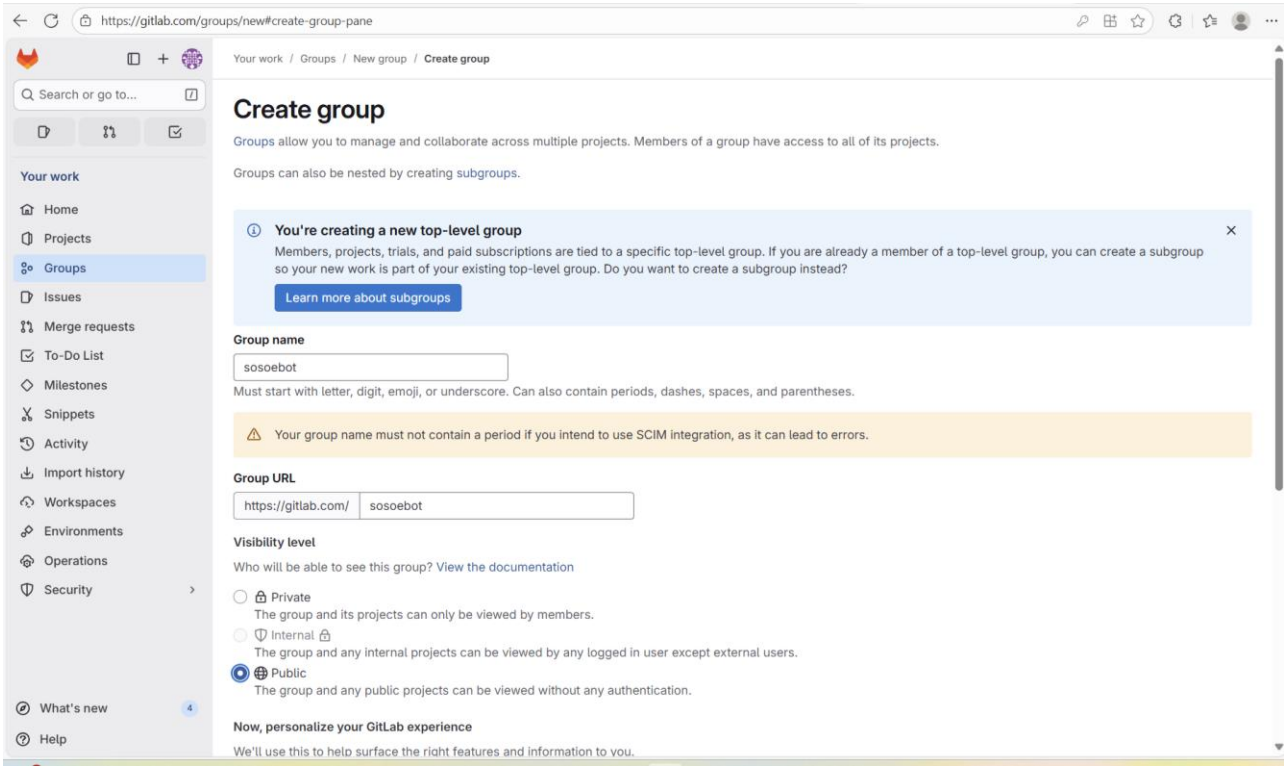
Click on "**Create Group**"



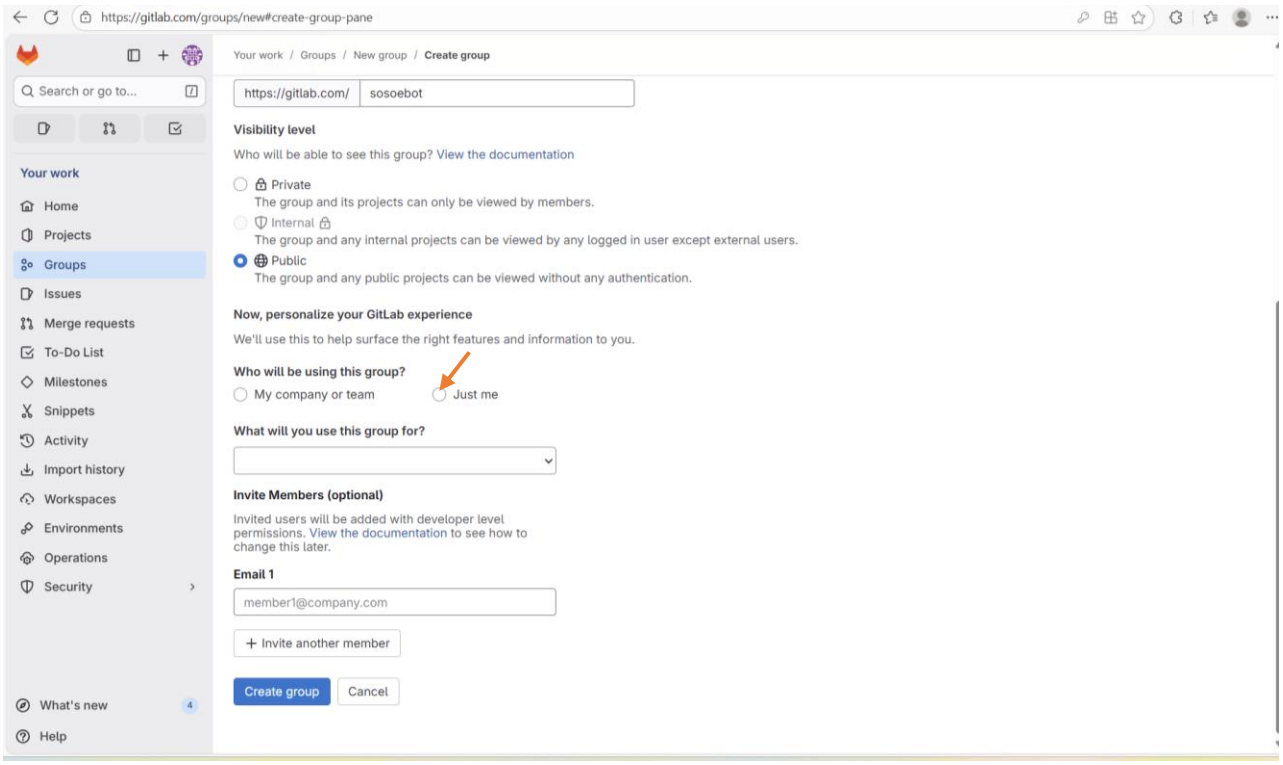
Enter the group name, I will call it "**sosoebot**"



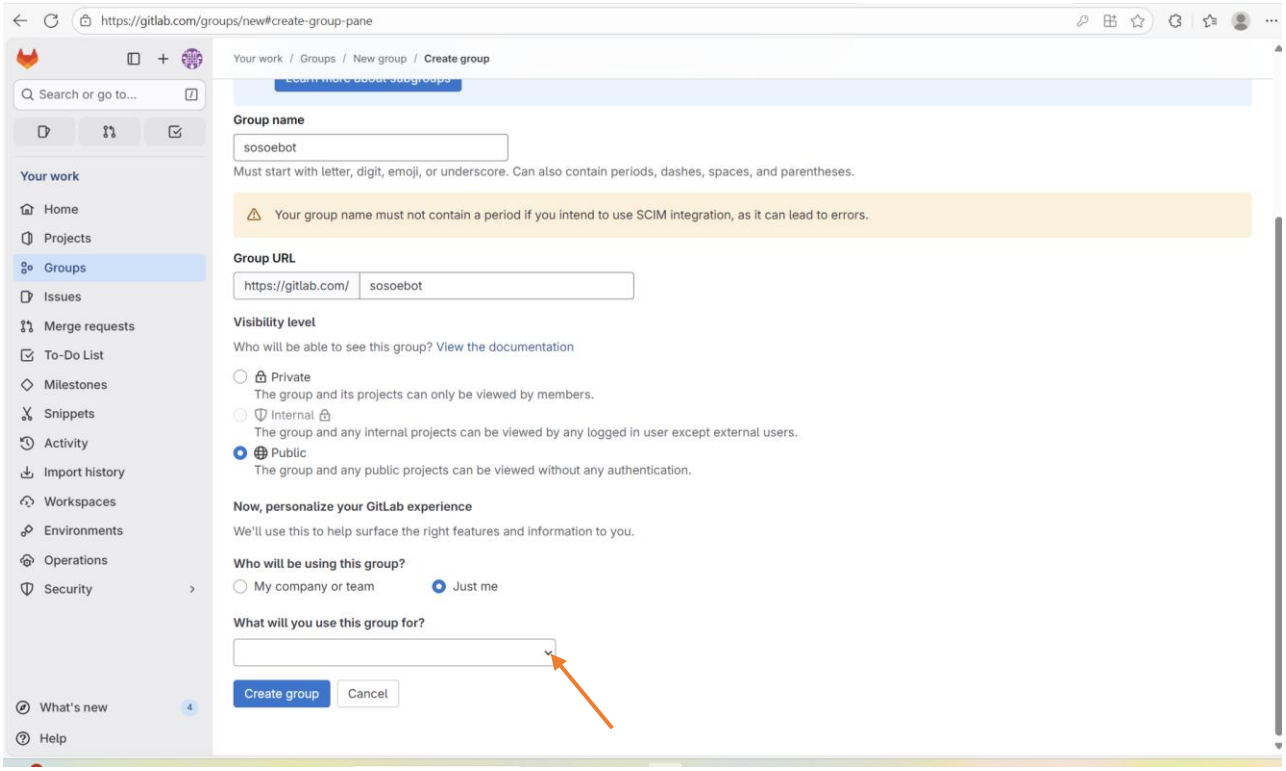
Then on "**Visibility Level**" select "**Public**"



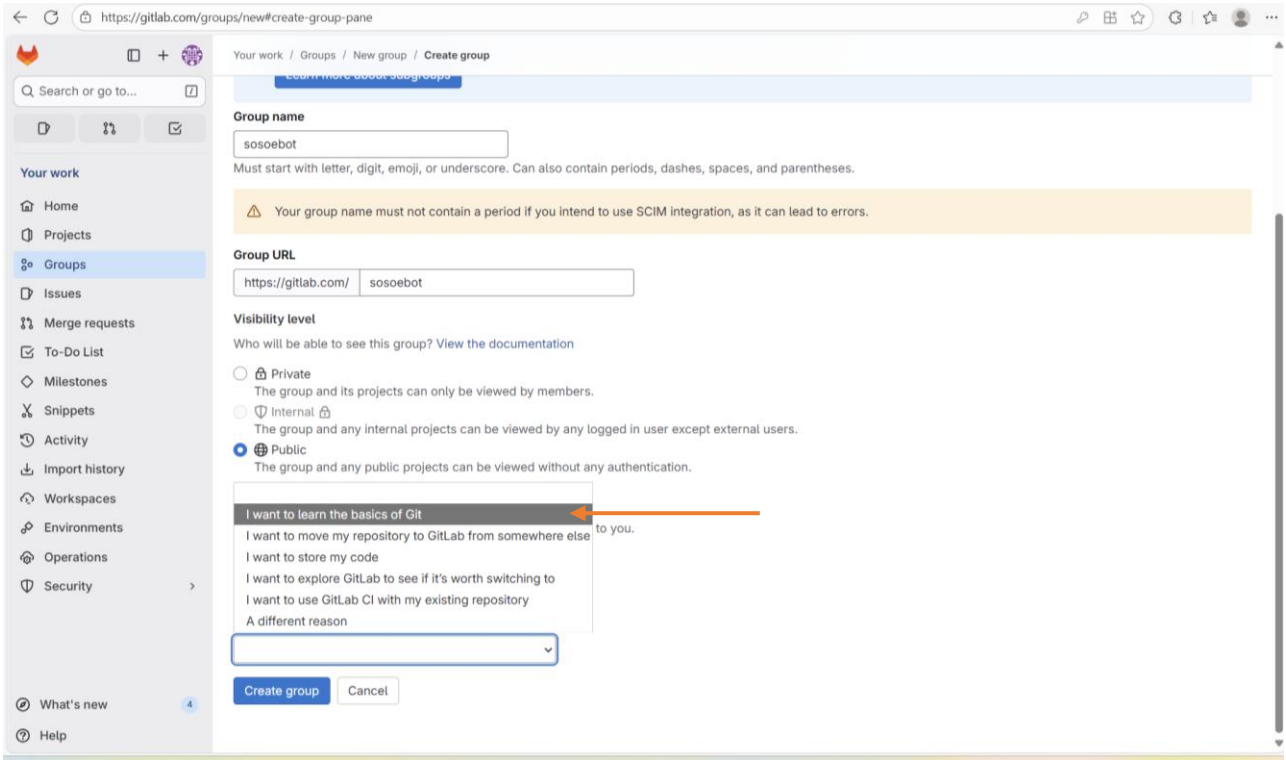
Scroll down



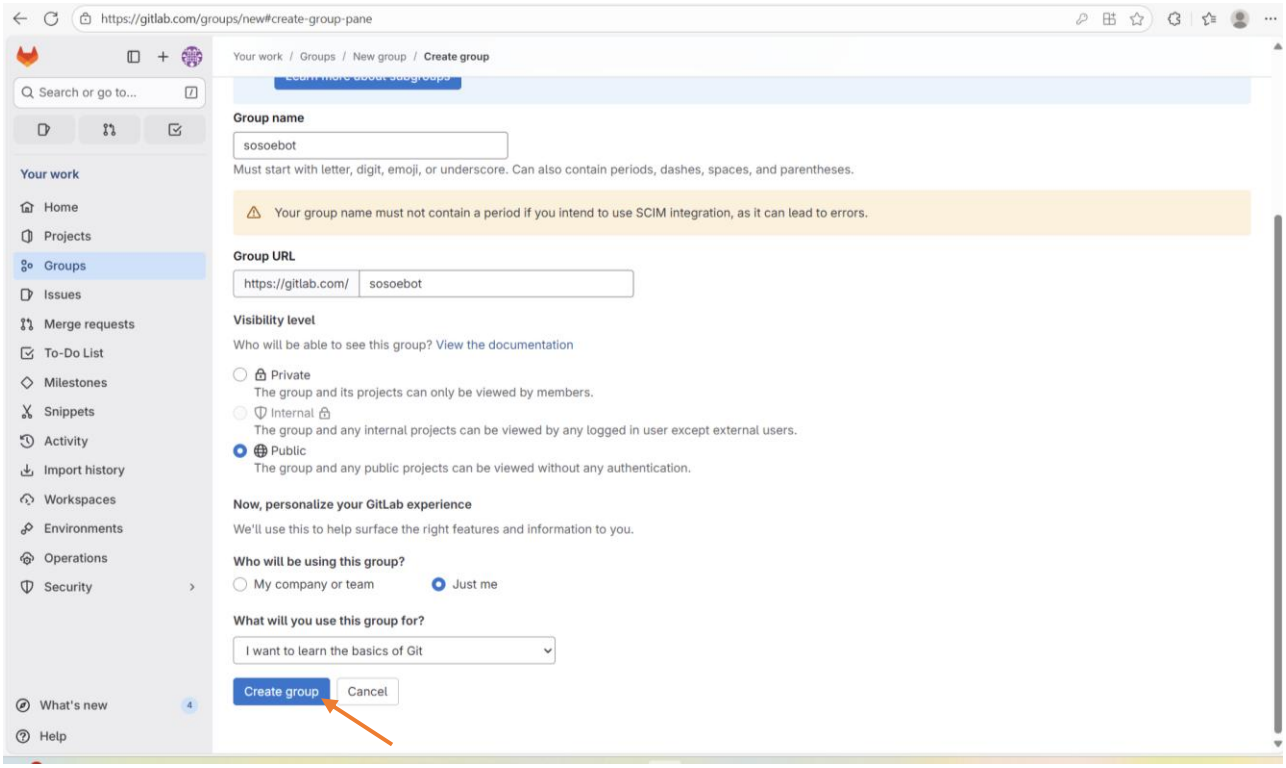
Then on "**who will be using this group**", I will select "**Just me**"



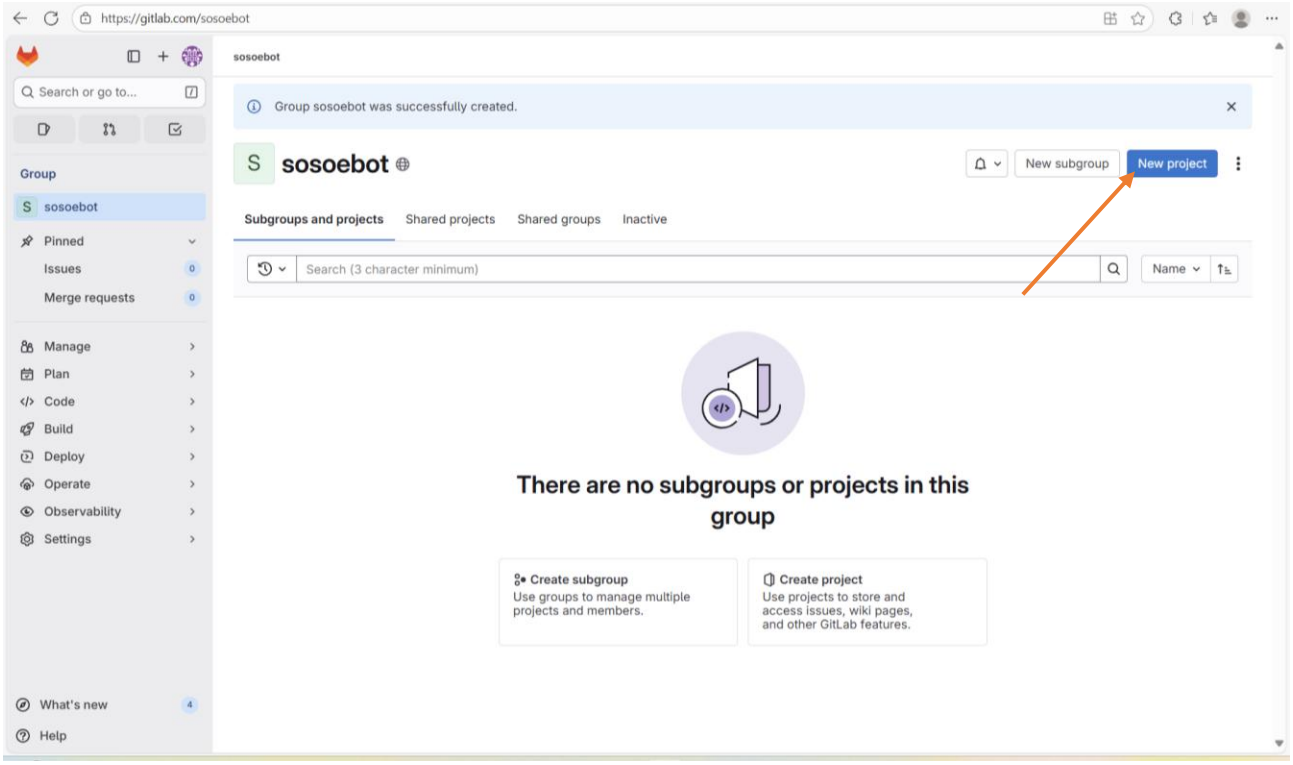
On "**What will you use this group for**", I will click on the drop down



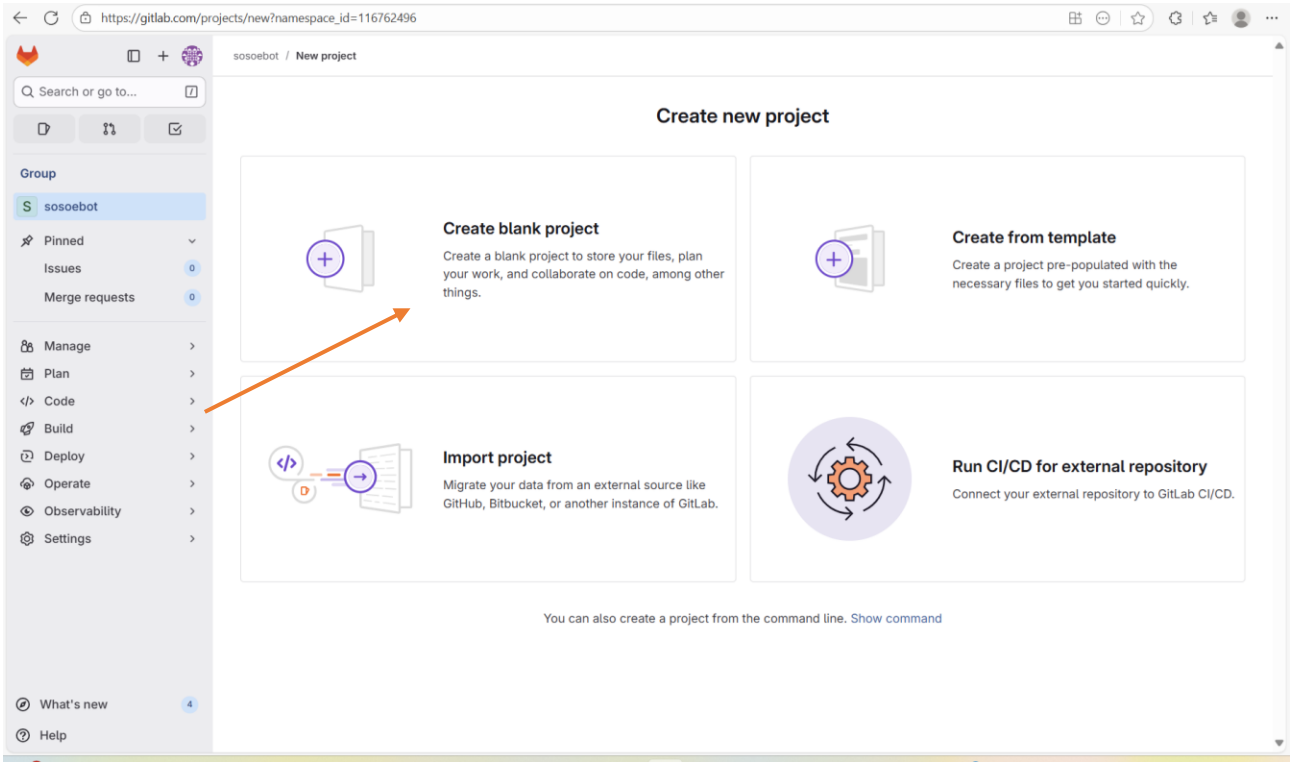
Will select "**I want to learn the basics of Git**"



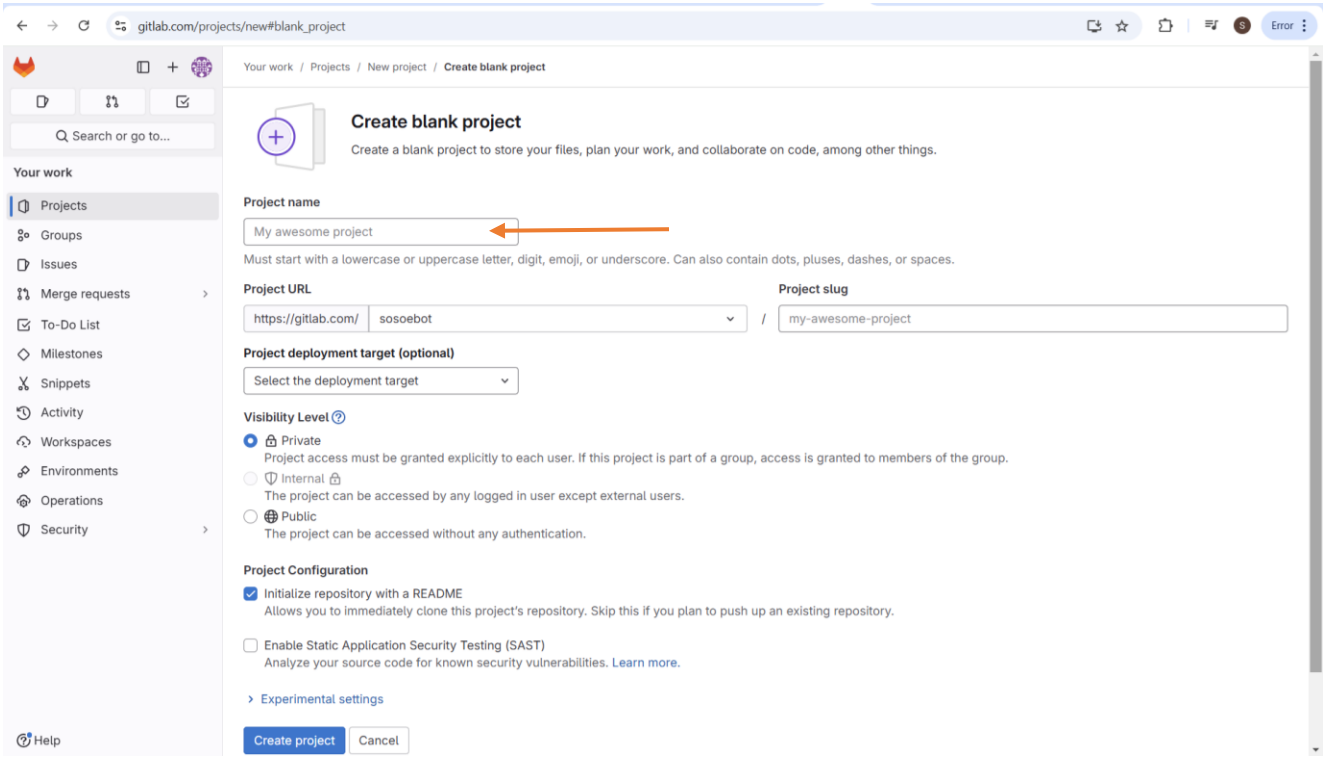
Then click on "**Create Group**"



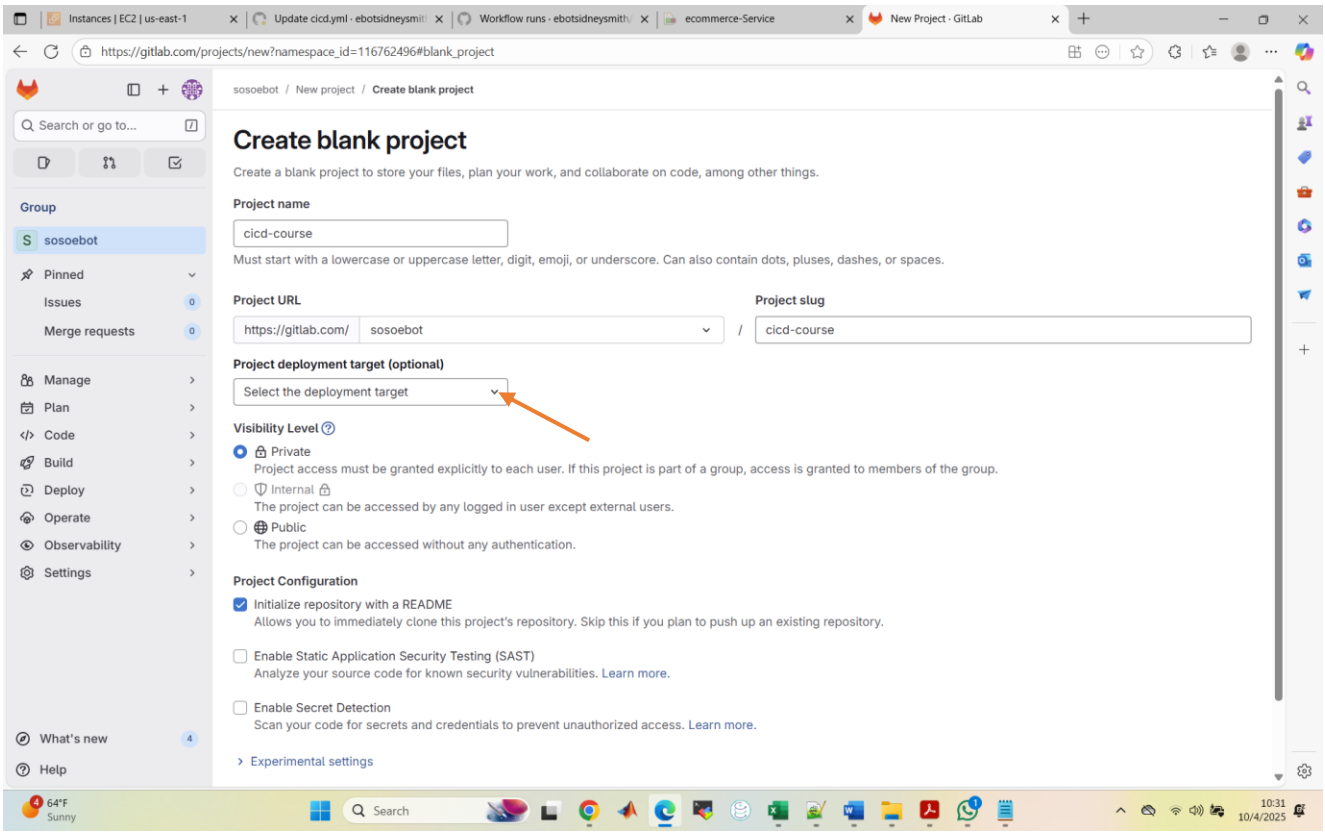
The group has been created. Let us now create a project in the group. Let us start creating our first CI/CD Pipeline. Click on "**New Project**"



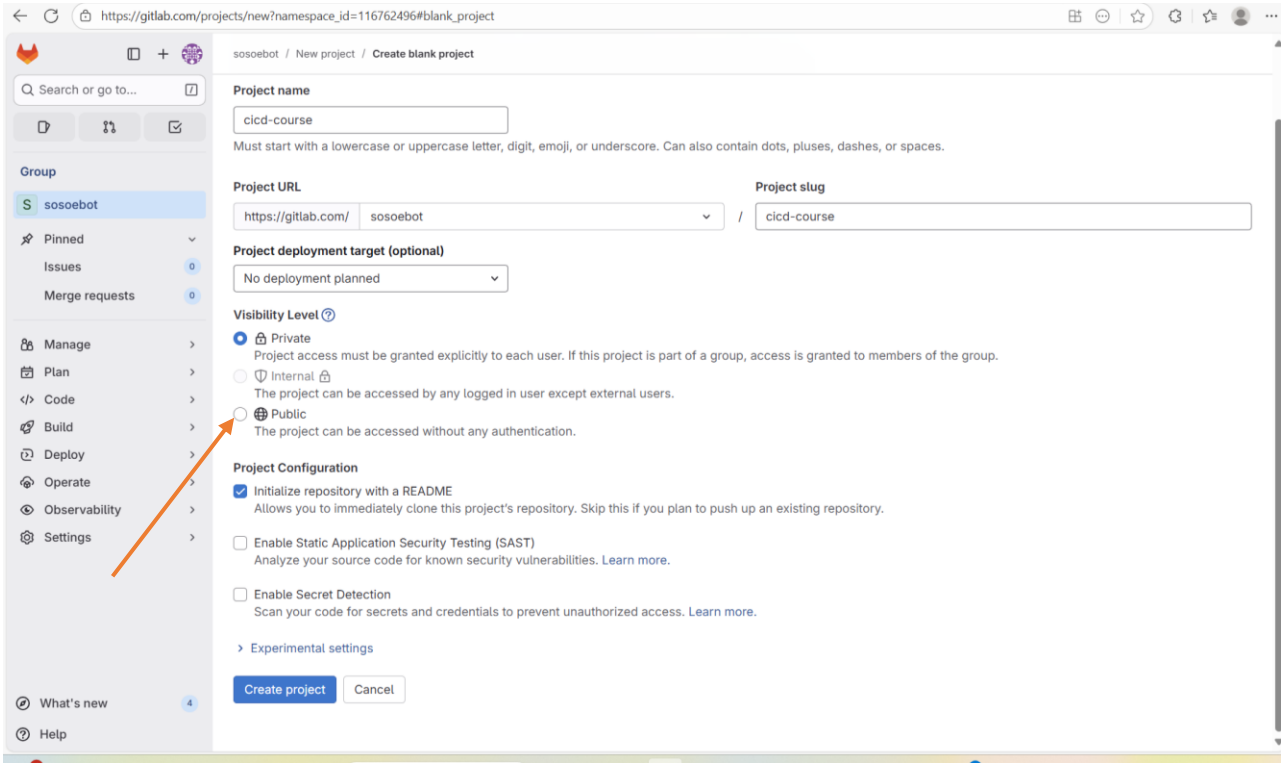
Click on "**Create Blank Project**"



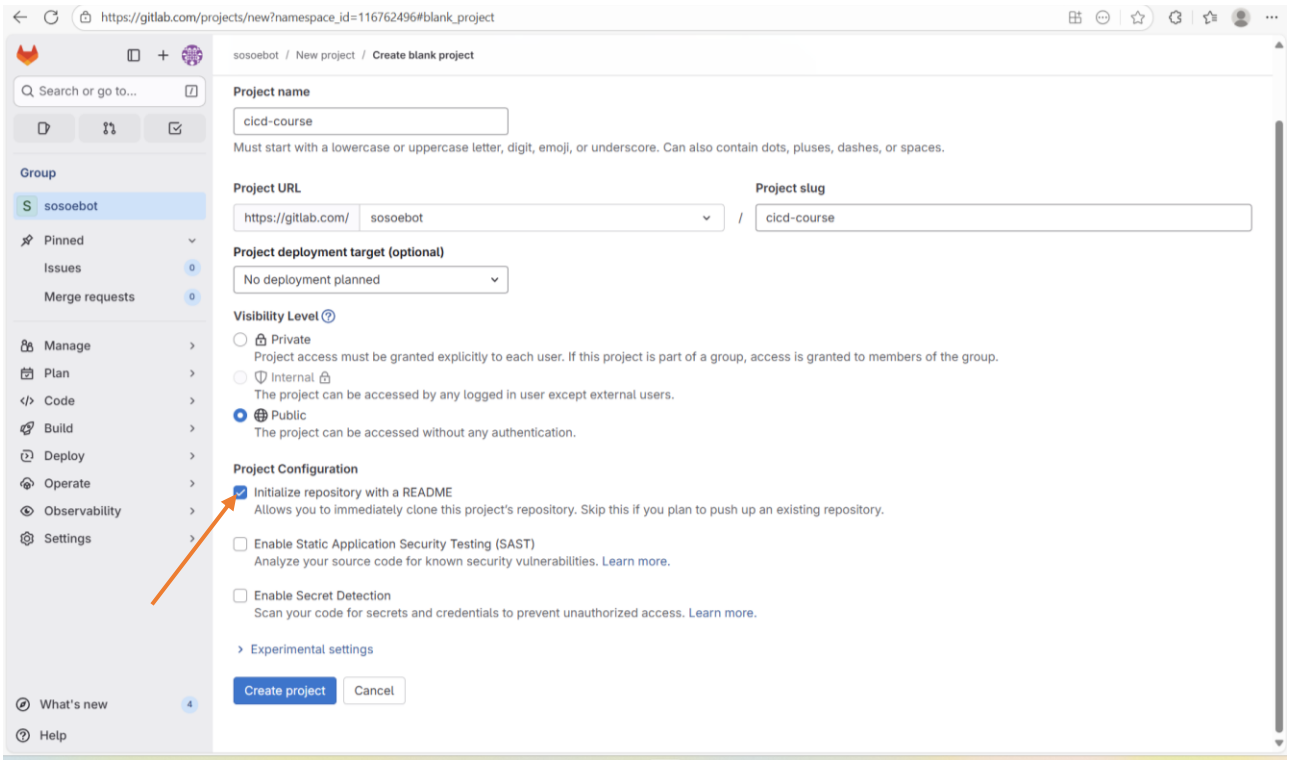
Let us name the Project as "**cicd-course**".



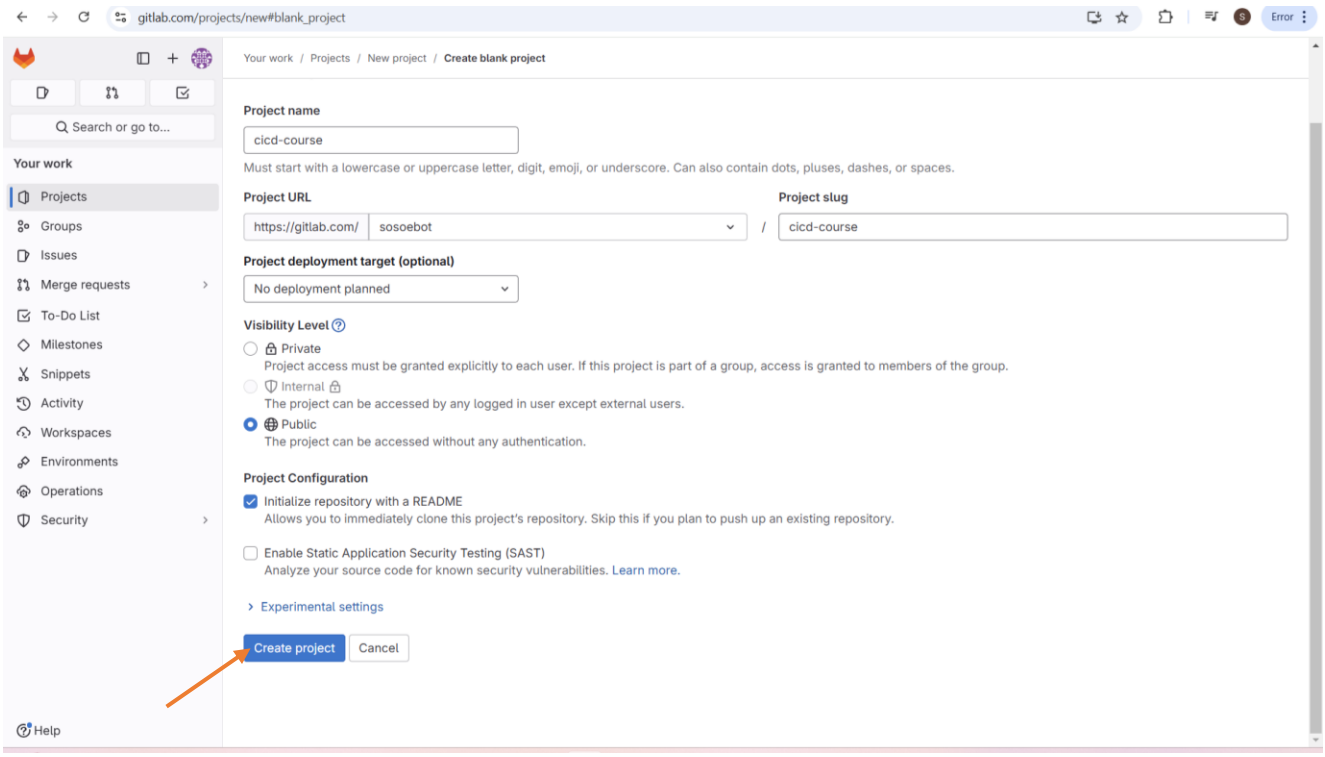
For deployment, we do not have any deployment plan. So, click on the drop down and choose "**No Deployment Planned**".



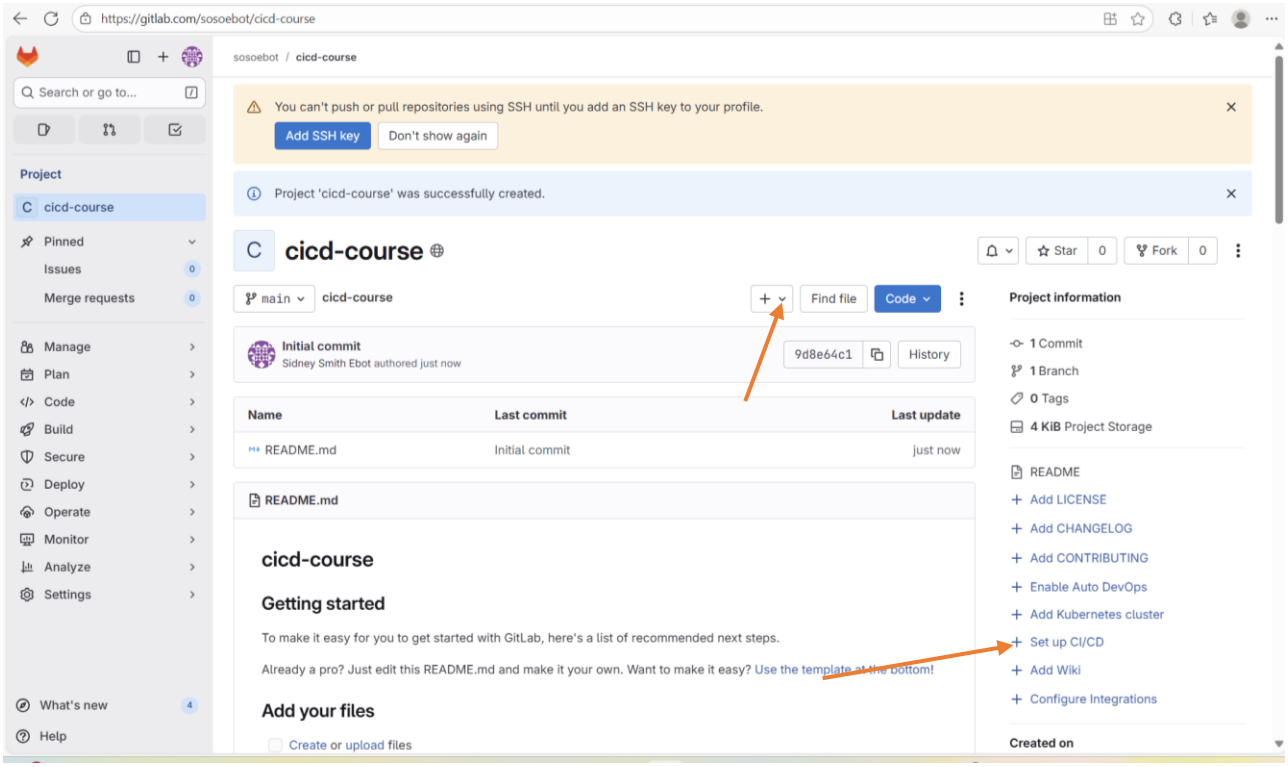
For visibility, we will make it "**public**".



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

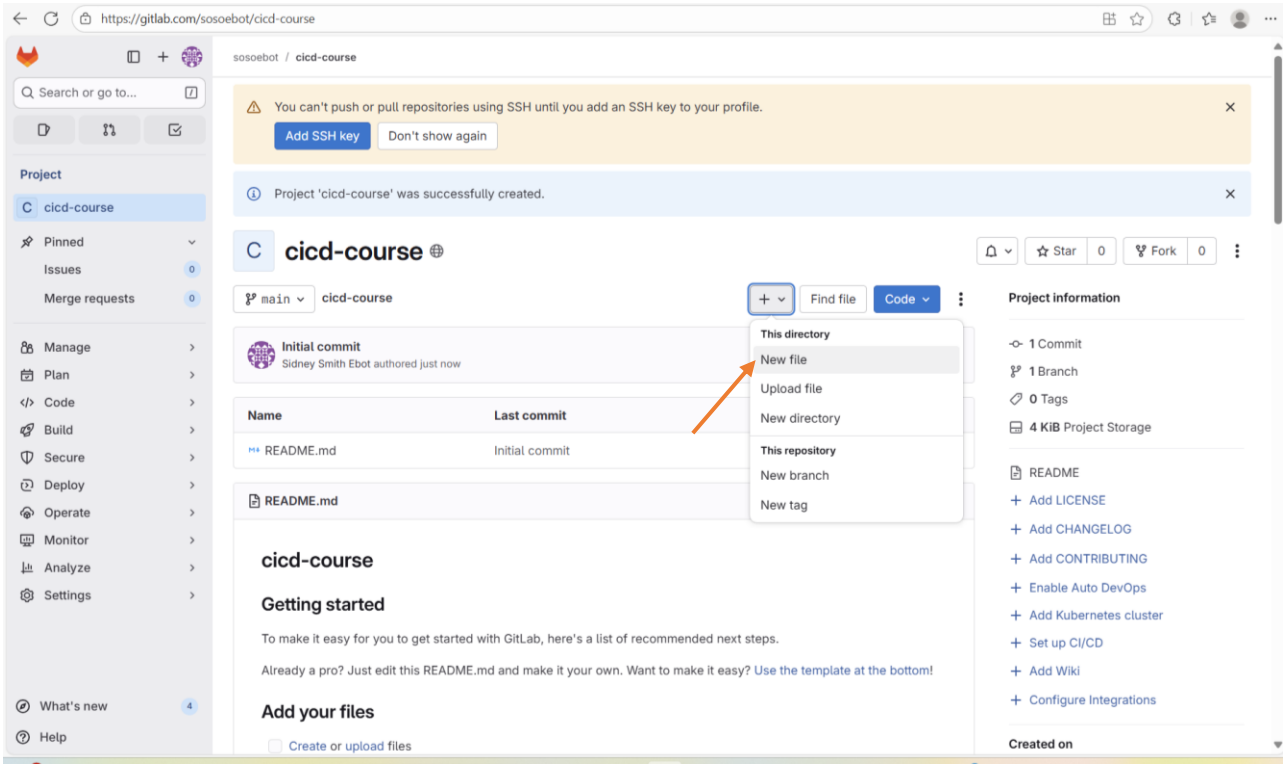


Click on "**Create Project**", this will create a repository in GitLab.

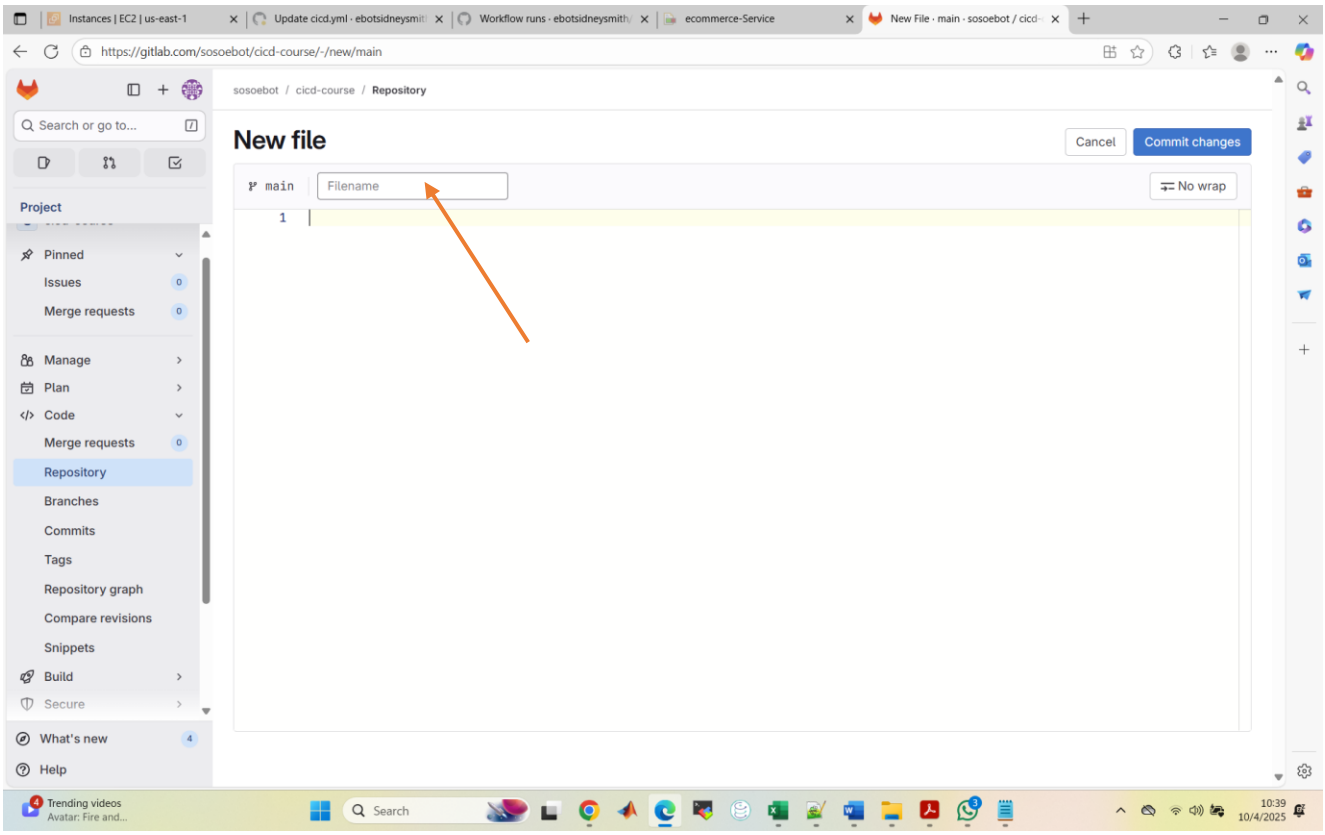


The repository has been created. We are inside CI/CD repository and to set up a CI/CD pipeline for this particular repo, we need to create a file that will hold all the configurations inside it.

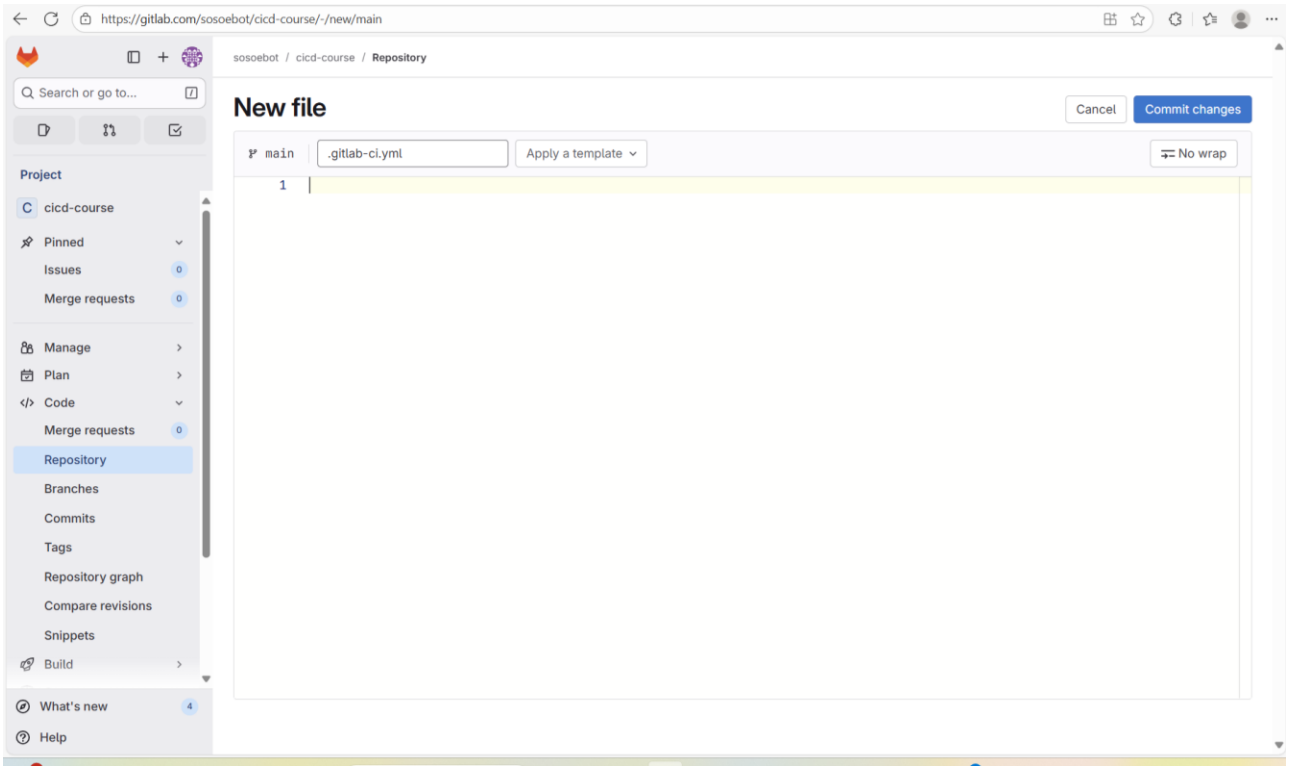
I can do that by clicking on the "**set-up CI/CD**" button but this will make things complex. So, instead click on the drop down on "**+**"



And select "**New File**"

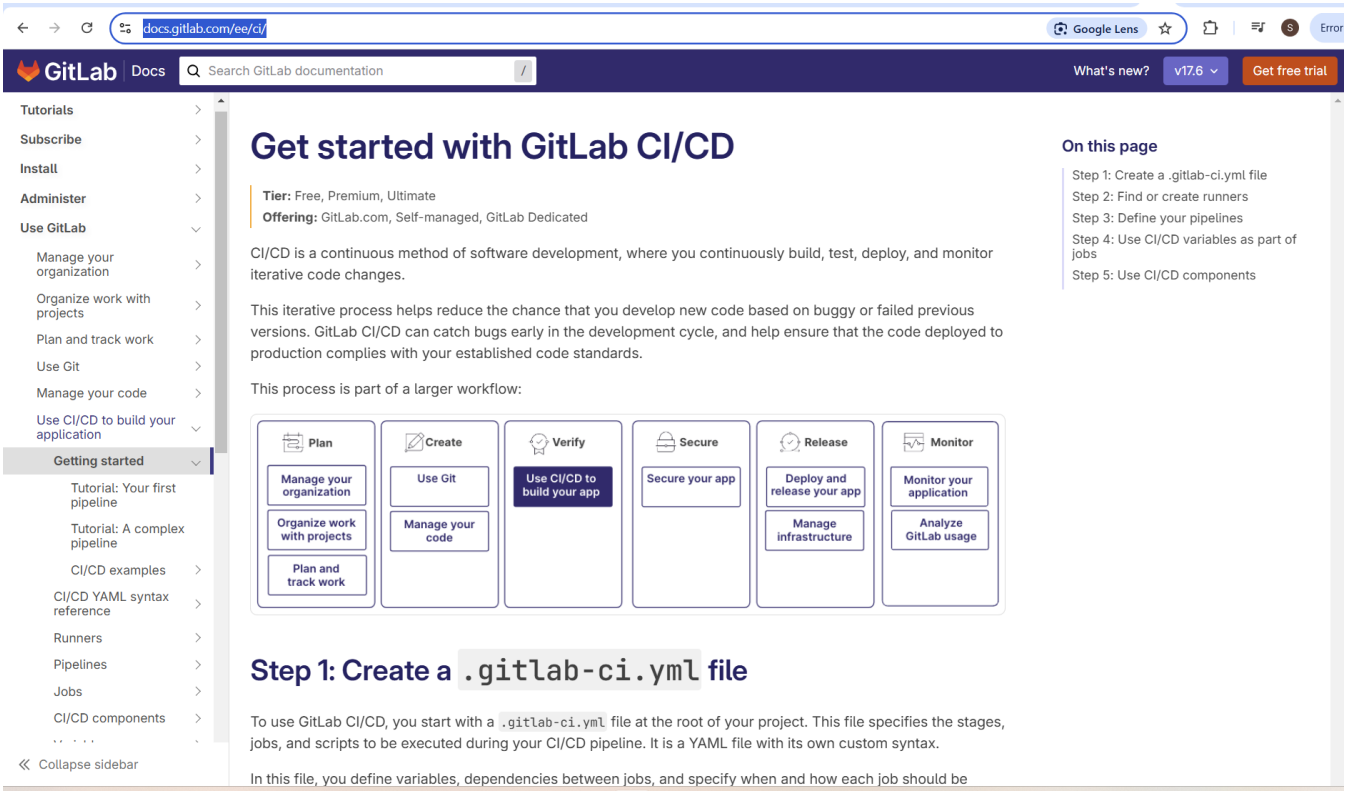


This file will hold all the CI/CD configuration. In GitLab if you want to create a pipeline which will hold all the CI/CD configuration, it should be named as "**`.gitlab-ci.yml`**".



This is the name you give to file that will hold the CI/CD pipeline configuration because GitLab will understands 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/



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.

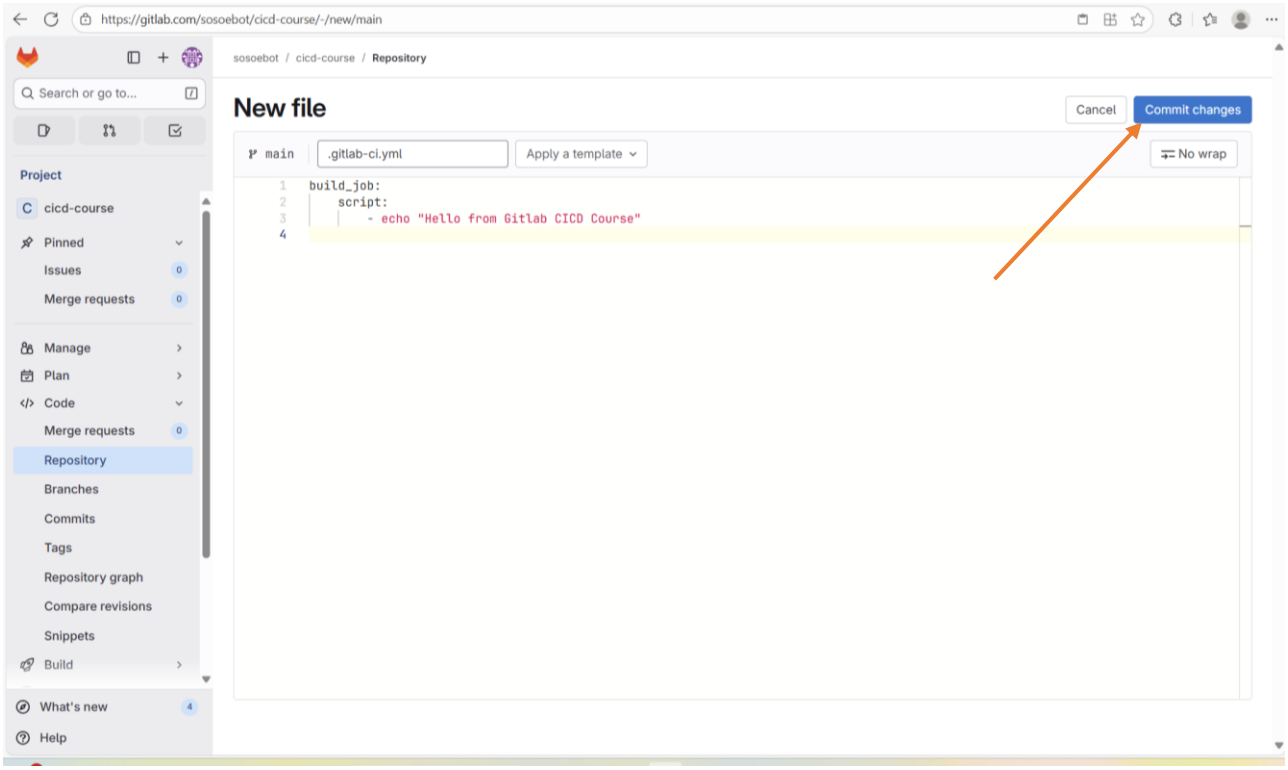
For example:

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

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

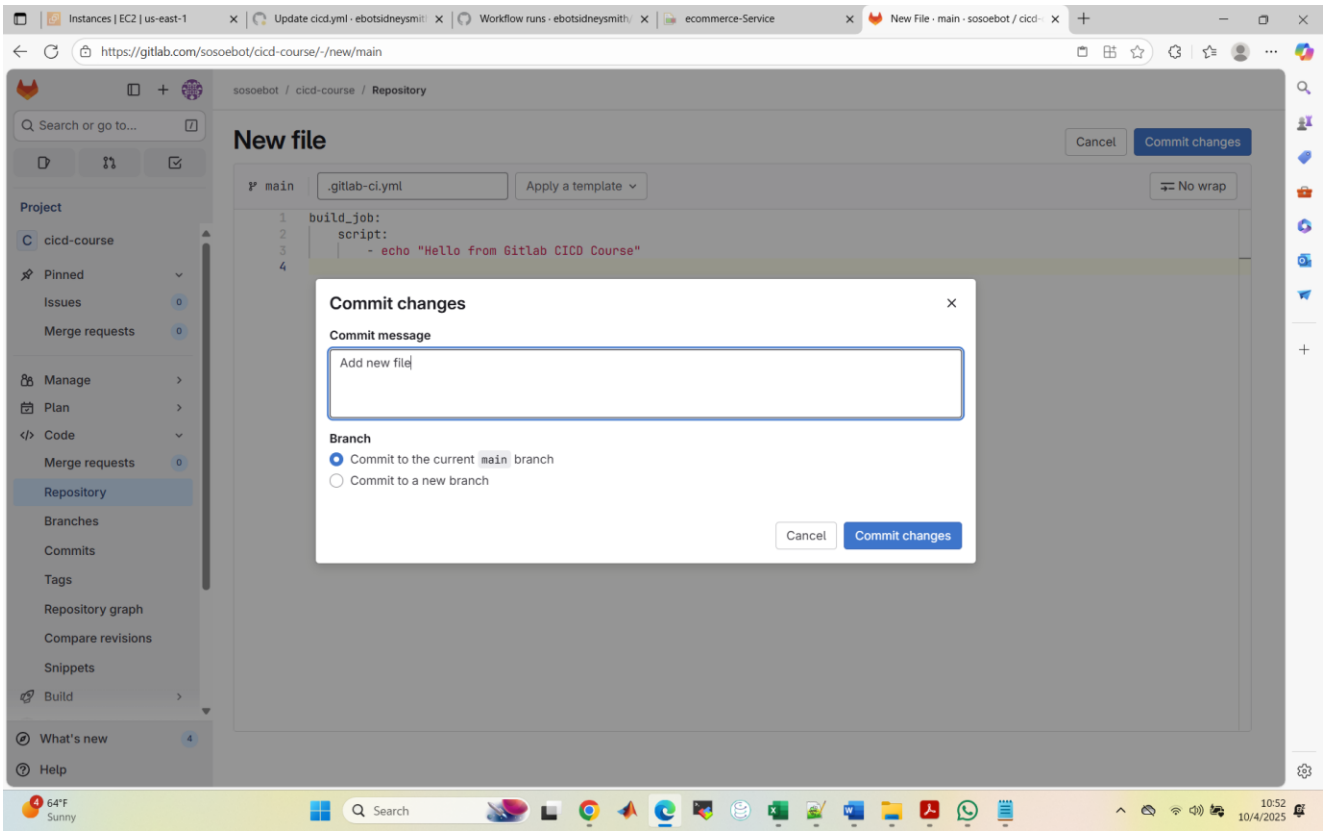
You can see we have two jobs in this example, namely "**my-ruby-job"** and "**my-shell-script-job"**. Job "**`my-ruby-job`**" is executing the script to **bundle install** and **bundle exec my_ruby_command**. While "**`my-shell-script-job`**" is executing script to the "**`my_shell_script.sh`**" file.

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

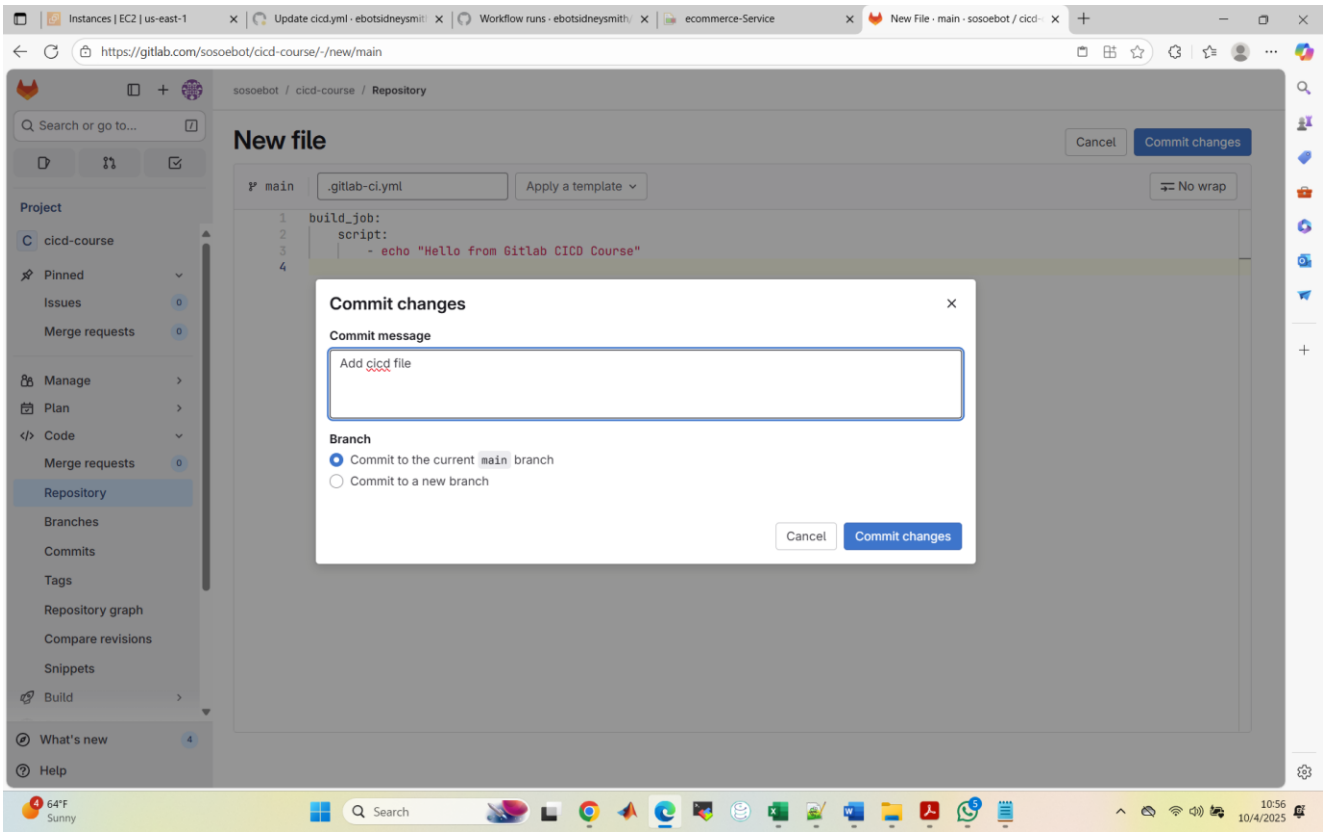


This is our CI/CD pipeline configuration now.

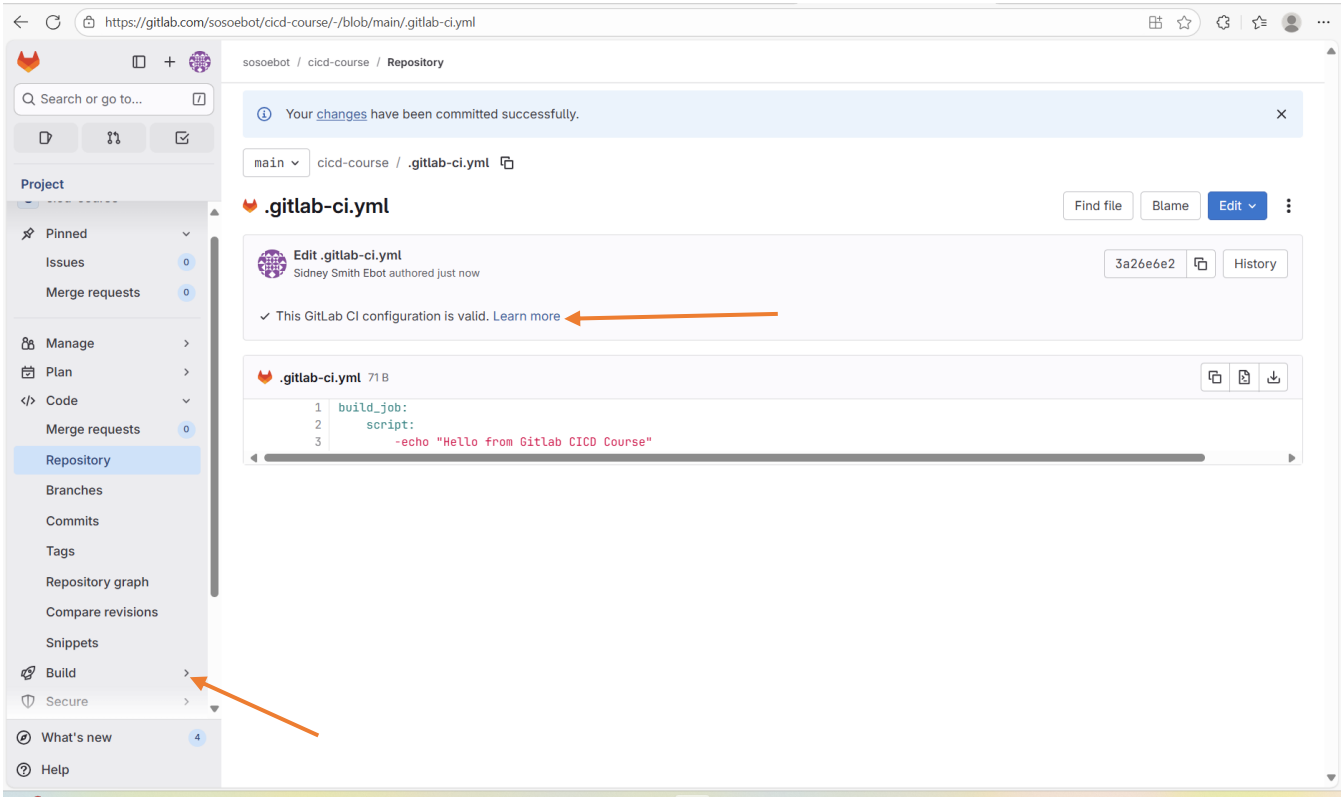
Now, we are going to **commit** it and let us see if our pipeline is going to succeed and show us this message of not. Click on "**Commit Changes**"



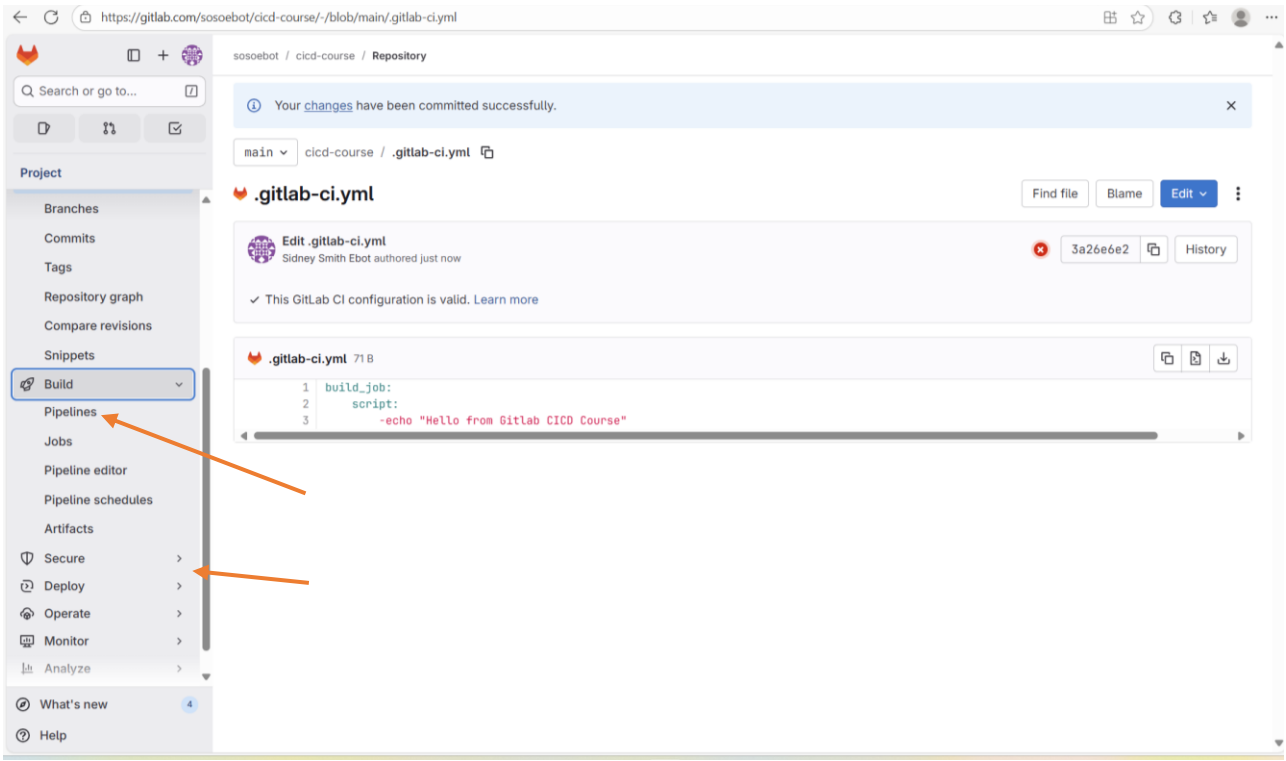
Add the message "**Add cicd file**"



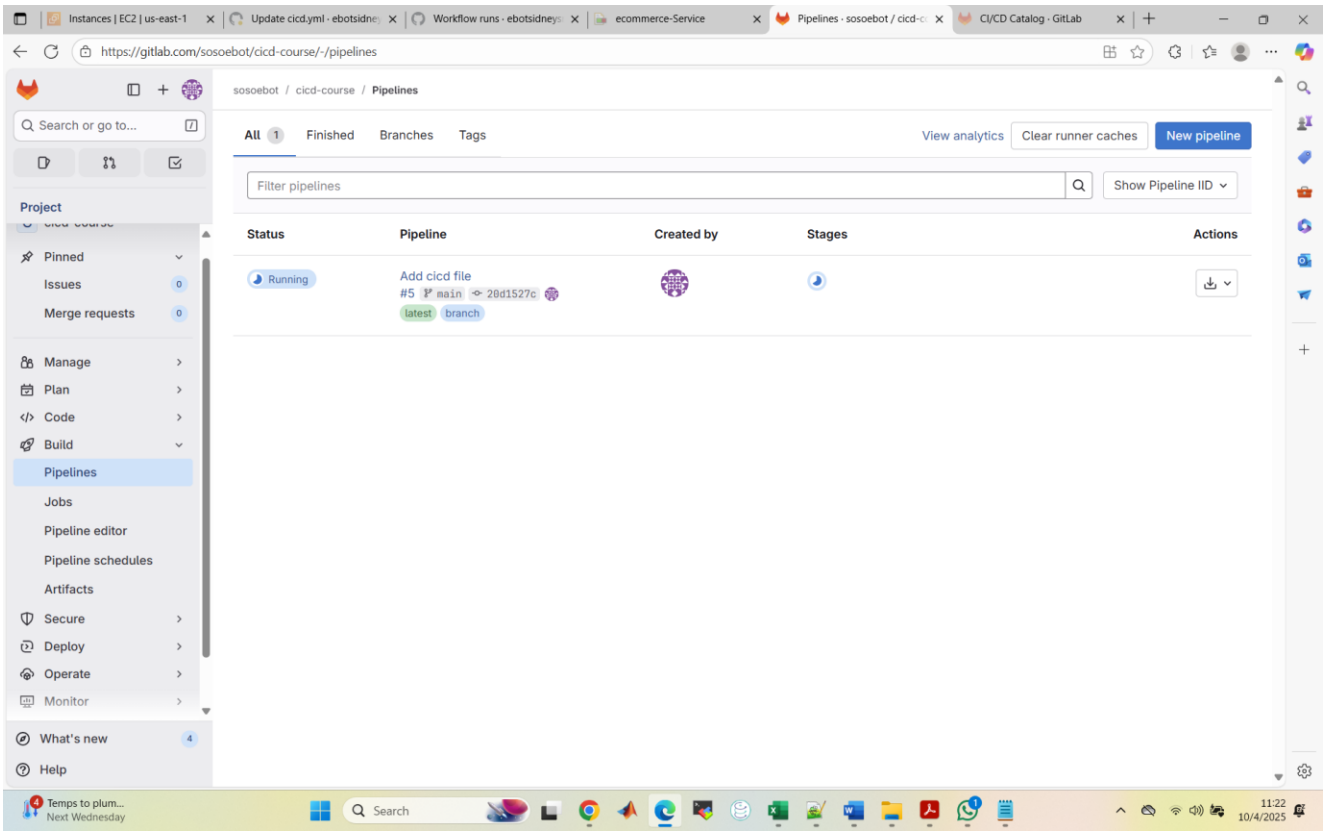
Then click on "**Commit Changes**" again. This will be committed to my main branch.



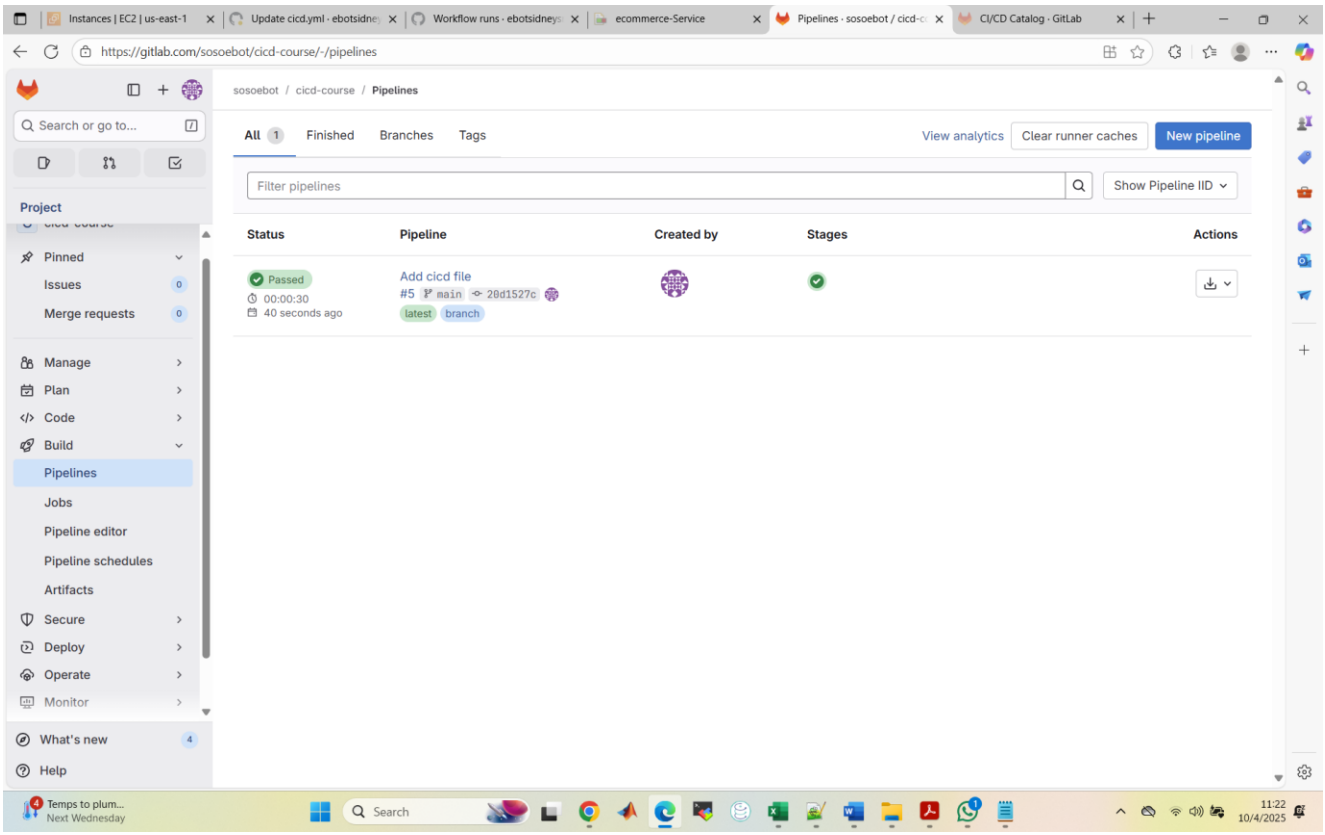
Gitlab will also provide you with a validation message if the CI/CD pipeline is valid or not, if there is syntax issues or not. Once you do this; you can go **build section** and click on **pipeline**



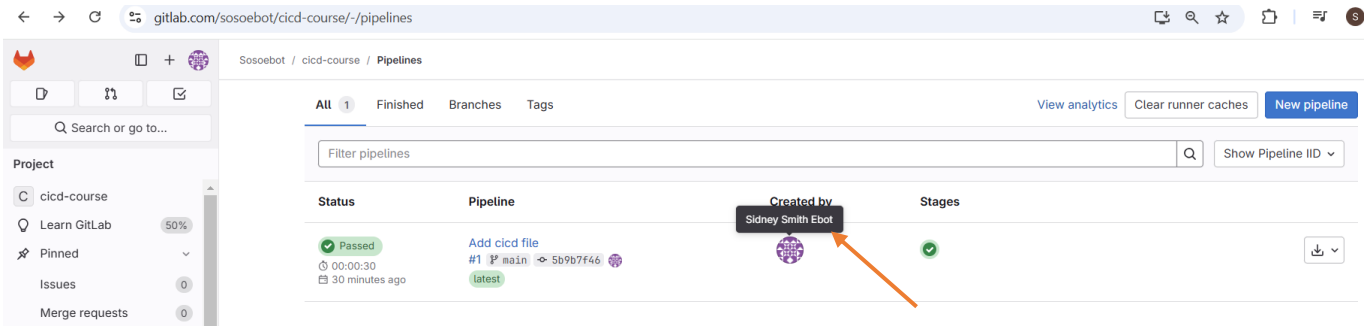
Click on "**Pipelines**", this will show you if the pipeline is running or not.



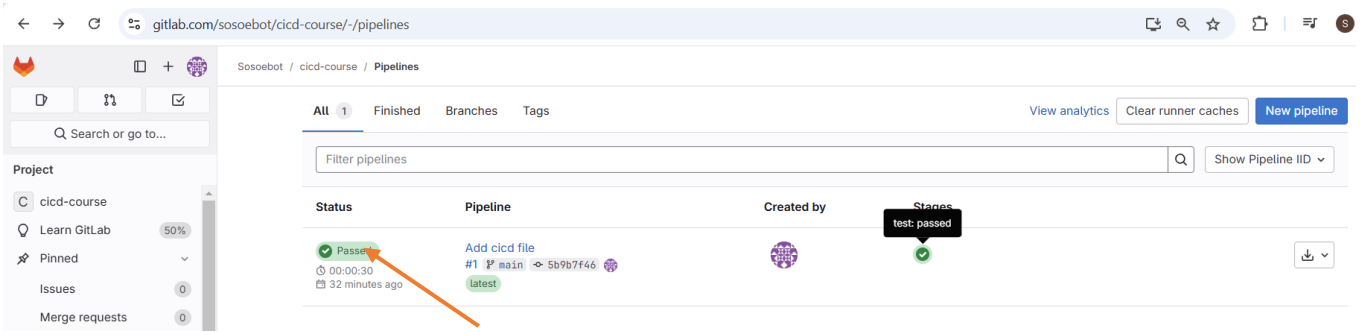
You can see that the pipeline is running



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.

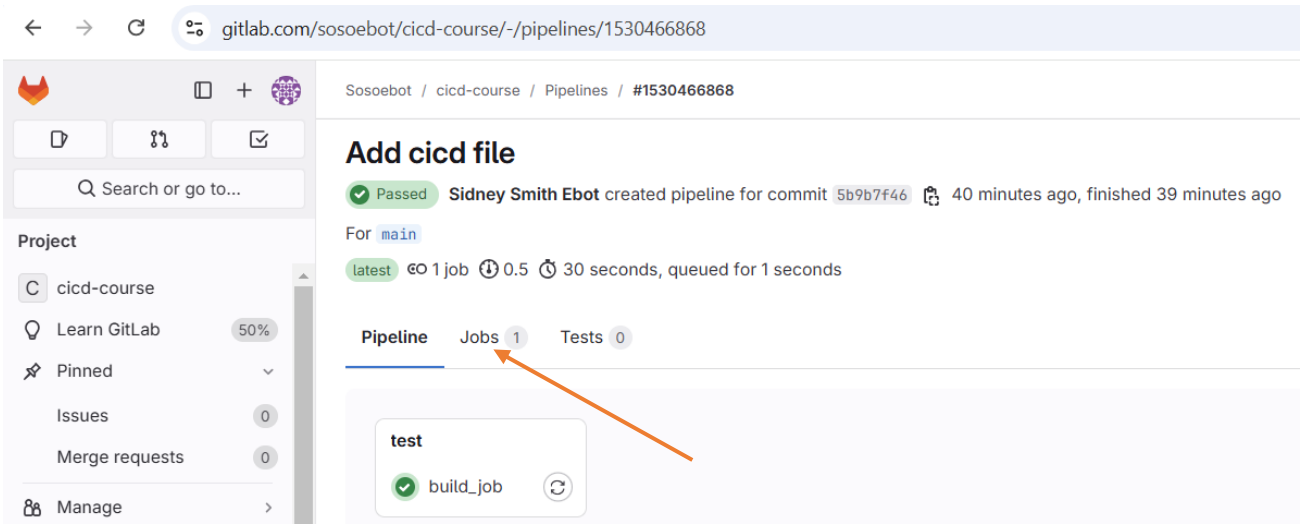


You can see that it is created by "**Sidney Smith Ebot**".

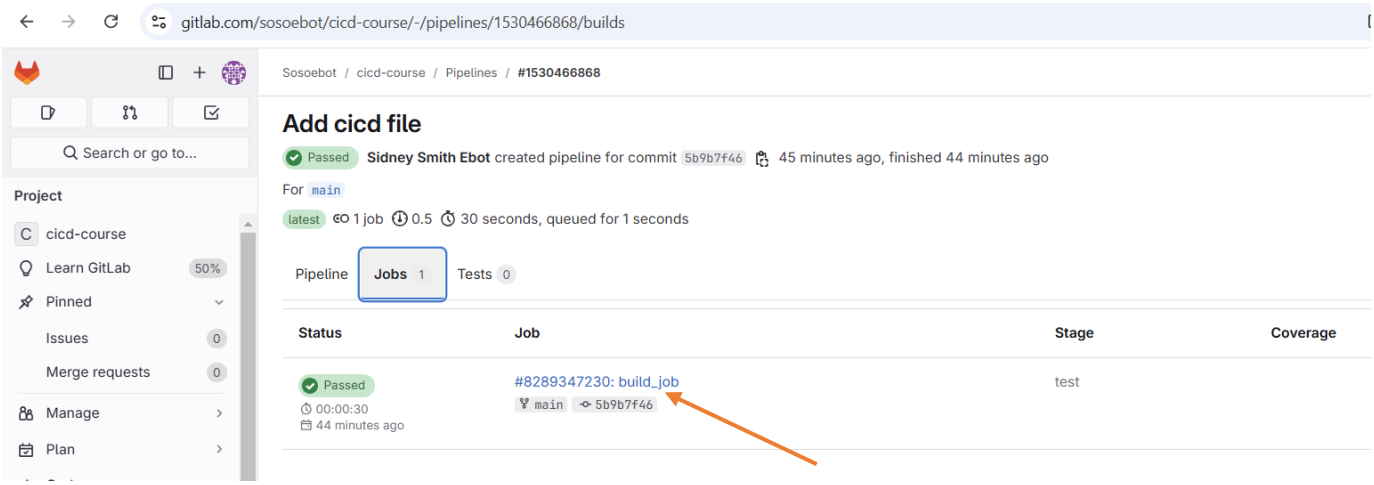


Right now, we only have a single stage which is the "**test**" stage, by default all the jobs will run in the test stage. And you can see it says "**Passed**"

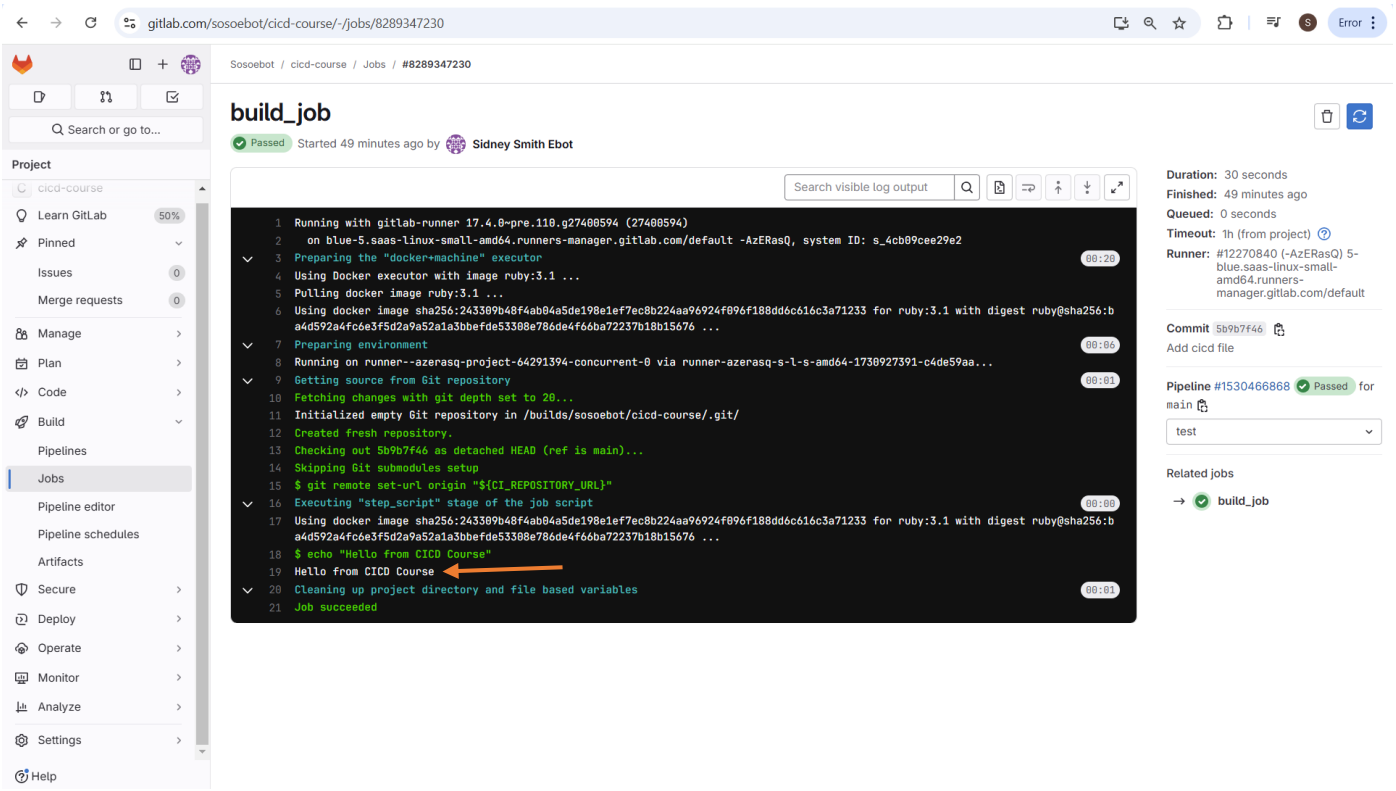
Click on the status "**Passed**", it will bring you inside



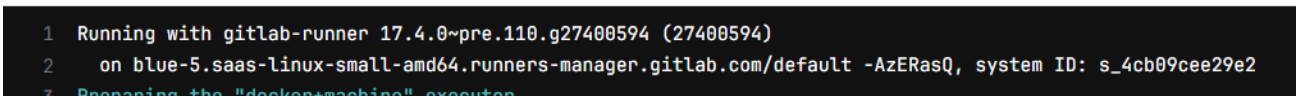
You can see there is a single job, click on "**Job**"



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



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.



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**.

```
Preparing the "docker+machine" executor                                    00:20
Using Docker executor with image ruby:3.1 ...
Pulling docker image ruby:3.1 ...
Using docker image sha256:243309b48f4ab04a5de198e1ef7ec8b224aa96924f096f188dd6c616c3a71233 for ruby:3.1 with digest ruby@sha256:b
a4d592a4fc6e3f5d2a9a52a1a3bbefde53308e786de4f66ba72237b18b15676 ...
```

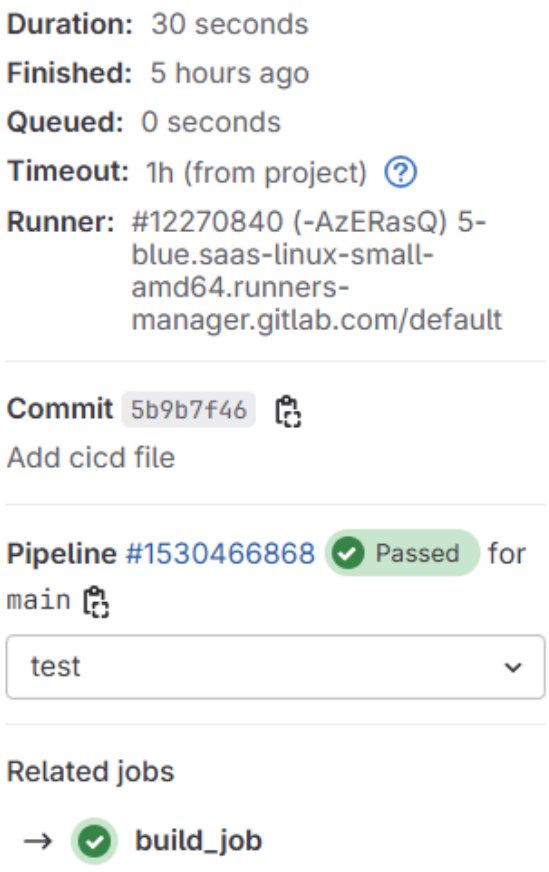
Next, it is preparing an environment for our job. It is getting the source from repository executing the script.

```
Preparing environment
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**"

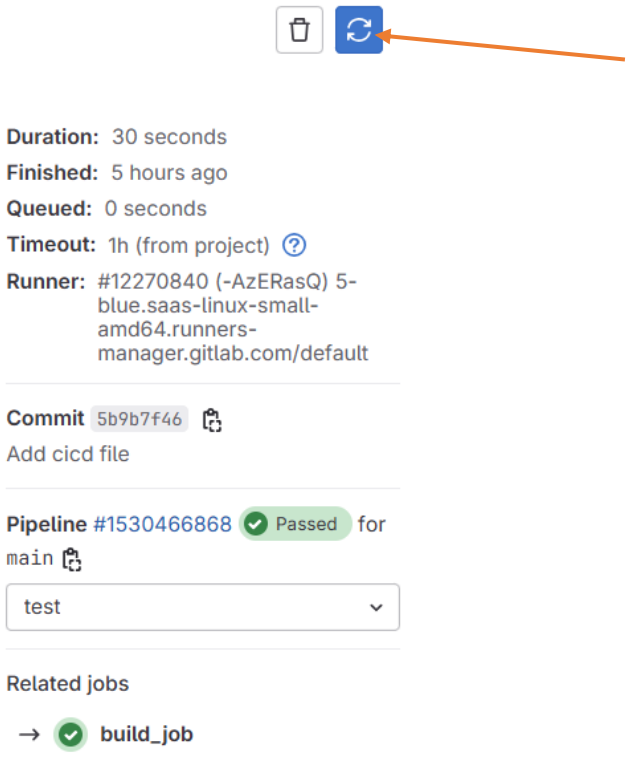
```
$ echo "Hello from CICD Course"
Hello from CICD Course
Cleaning up project directory and file based variables
Job succeeded
```

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

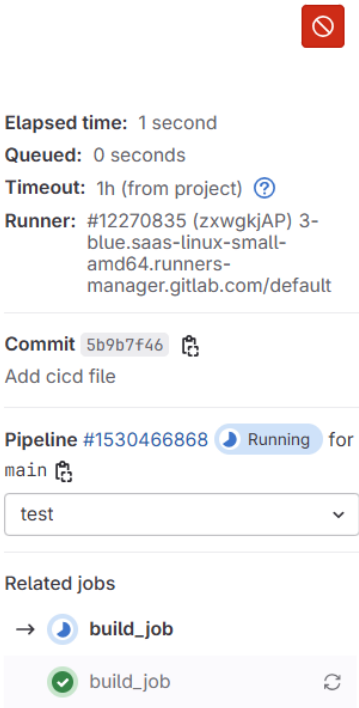


The job ran for 30 seconds, it has finished in 5 hours. There was no queue. The timeout for this particular job is 1 hour by default. The runner is a particular runner used for this particular job. The commit ID is **`5b9b7f46`**, which ran the pipeline. And you also see the different stages, right now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



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