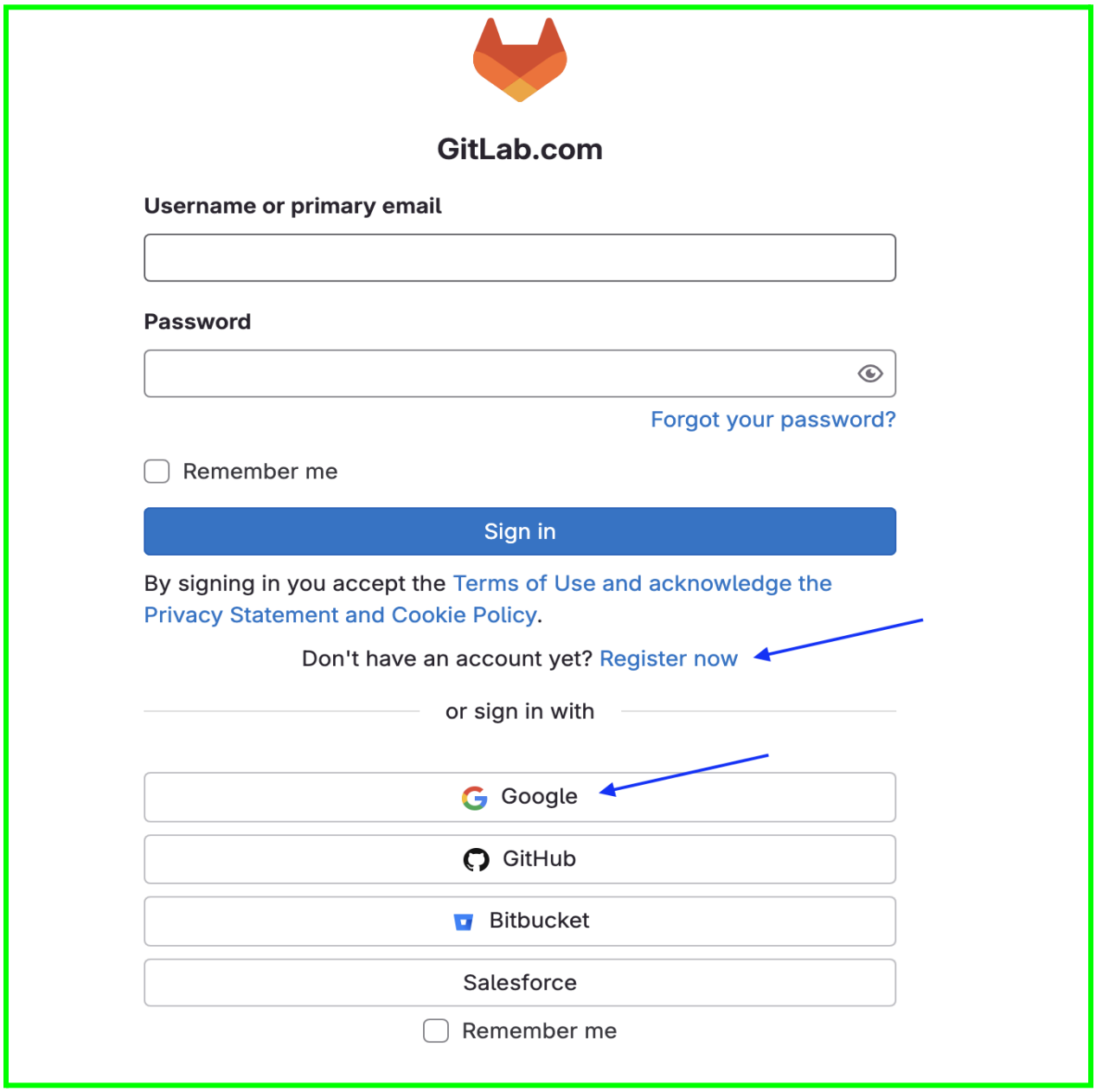
Table of Contents:

Dheeraj Yadav

Step-1 : Gitlab Account

To Start using Gitlab, first you should have a Gitlab Account, if you don't have please create by click on **"Register now"** & fill all the required details or use Google to create an account for you. Verify your email address & mobile number to register successfully.



==========

Step-2 : Create a Group & Project

Create a New Group

In GitLab, you use **groups** to manage one or more related projects at the same time.

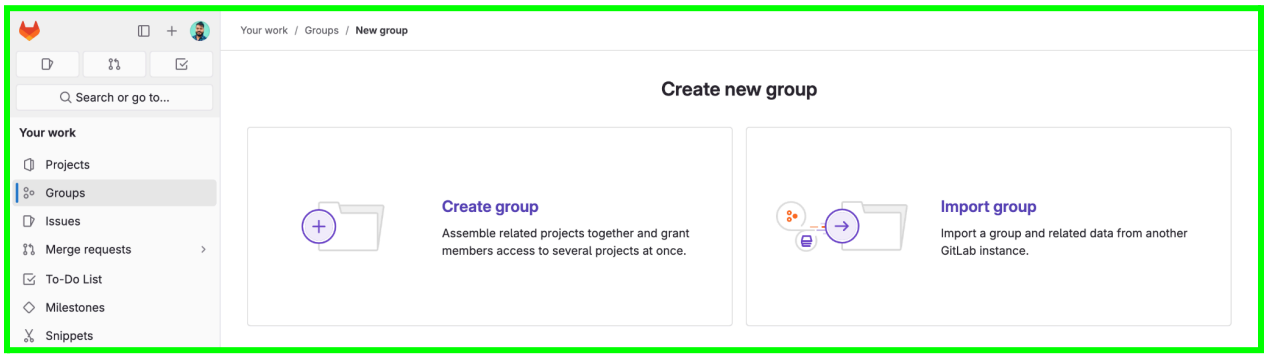
You can use groups to communicate with all group members and **manage permissions** for your projects. If someone has access to the group, they get access to **all the projects** in the group.

You can also view all of the issues and merge requests for the projects in the group, and analytics about the group's activity.

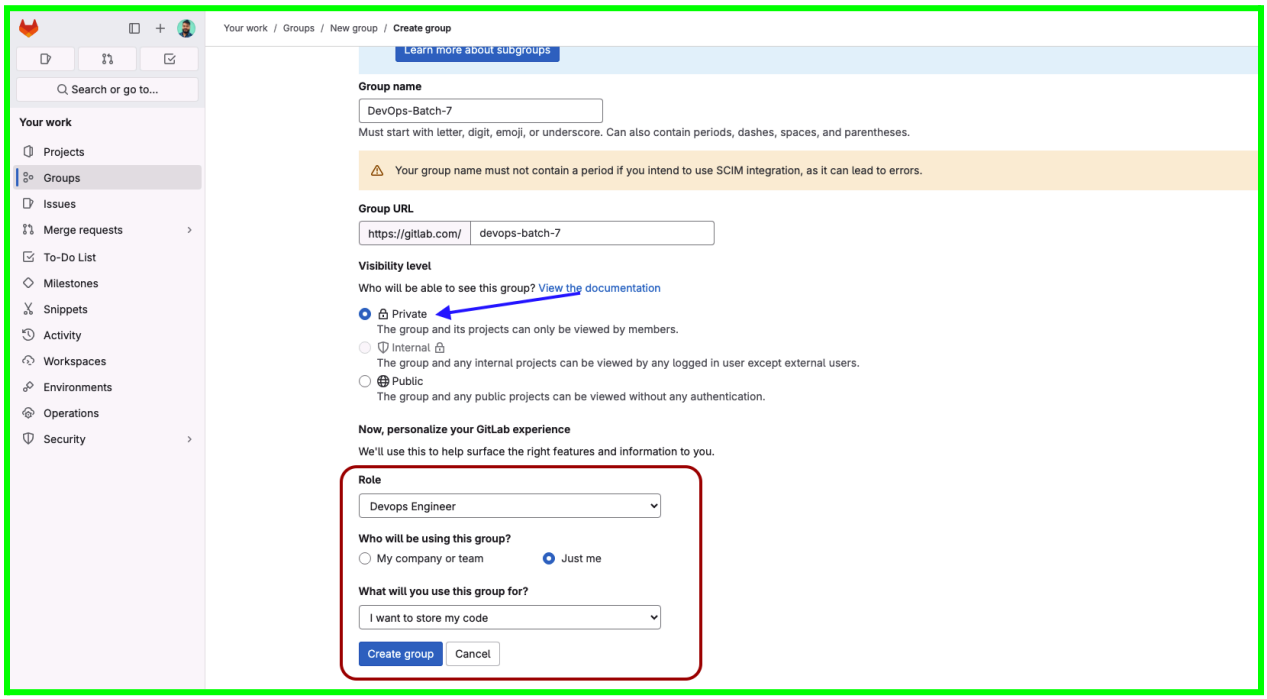
 # Let's Create a New Group:

Click on Group icon —> New Group

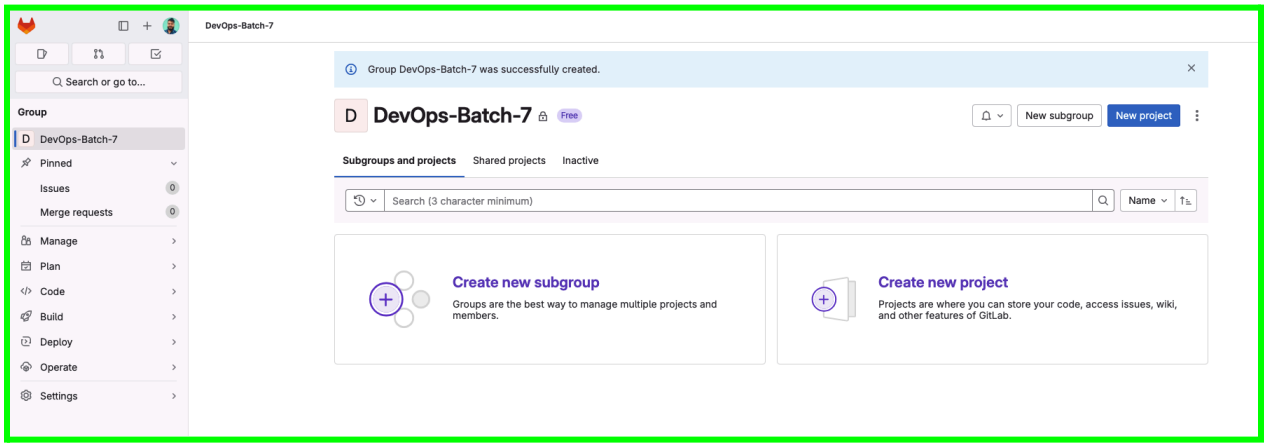
Click on **"Create Group"**



Created a New Group called **“DevOps-Batch-7”** with Private visibility, you can make it Public as well. I chose Private for this Group.



Once created, it will give you the below screen, where you can create a subgroup or New Project.

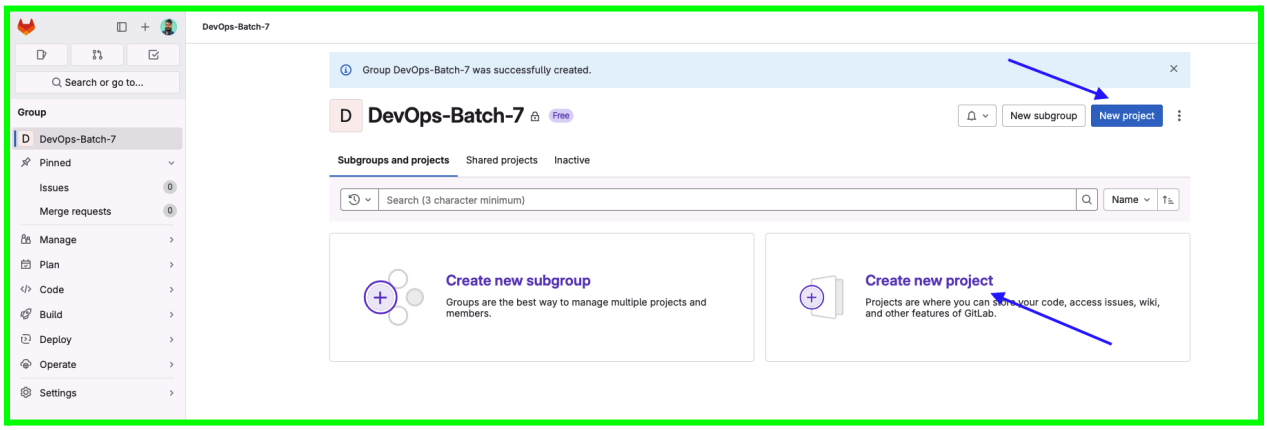


==========

Create a New Project

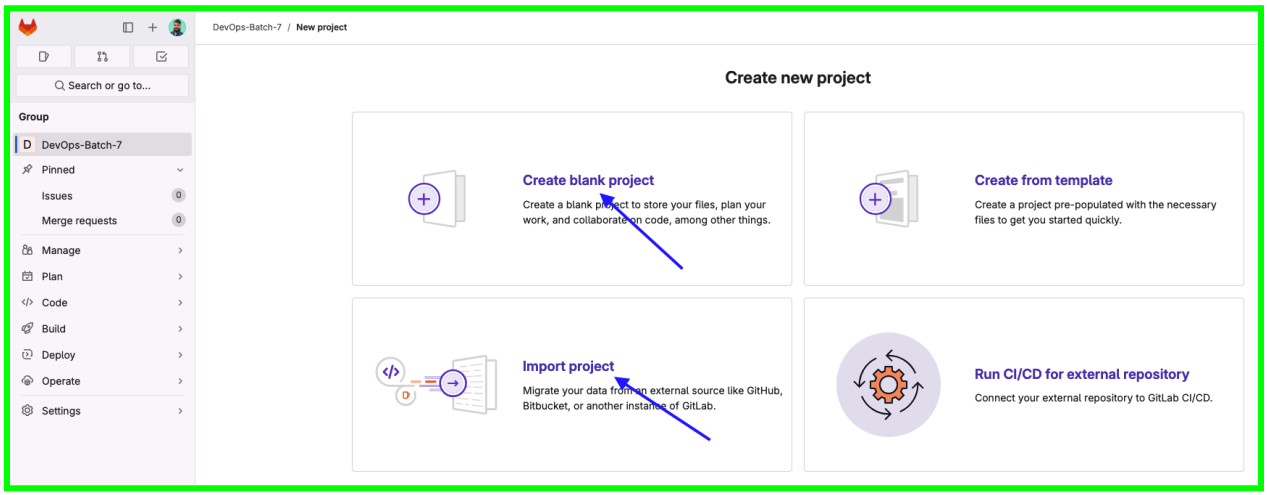
In Github we can call it a repository, in Gitlab it's Project.

Let's Create a Project → Click on **New Project**



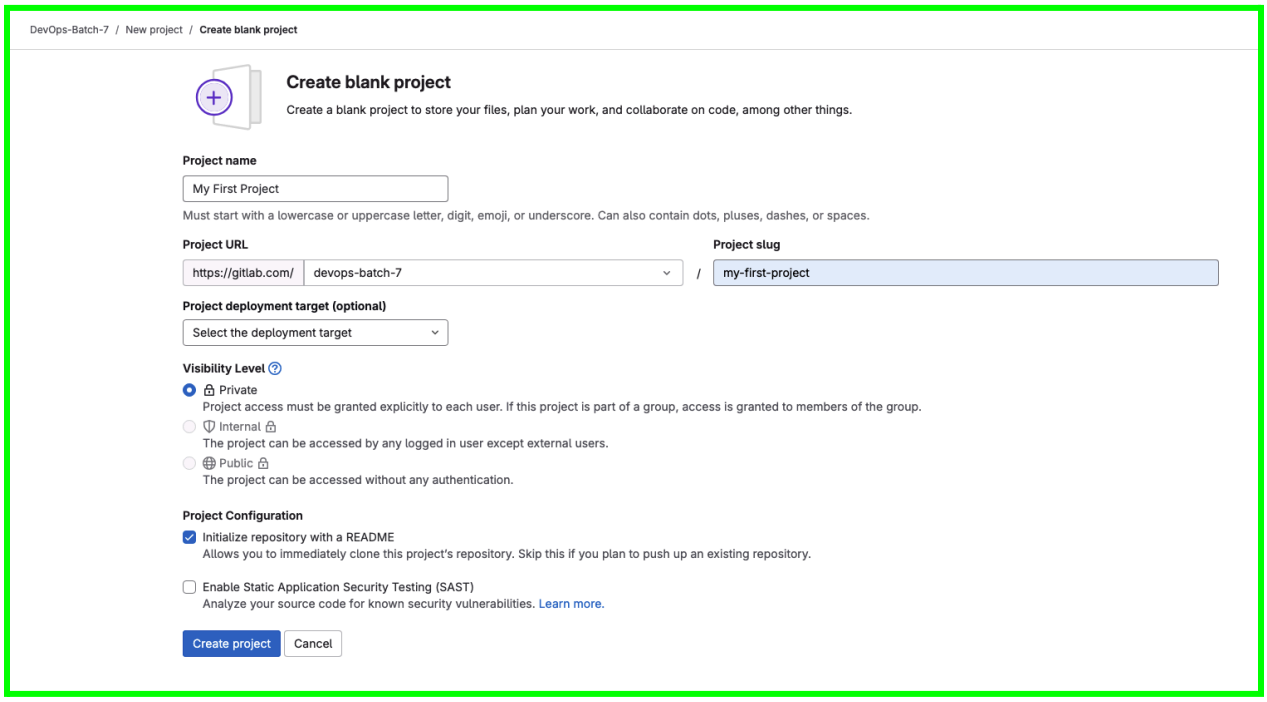
Once you click on New Project, it will give you 4 Options, as you can see below:

1. **Create a Blank Project :** If you want to create a New Project, then select this Option.
2. **Import Project :** You can import the project from github, bitbucket etc.



Dheeraj Yadav

Give a Name of your Project & Create a Project.

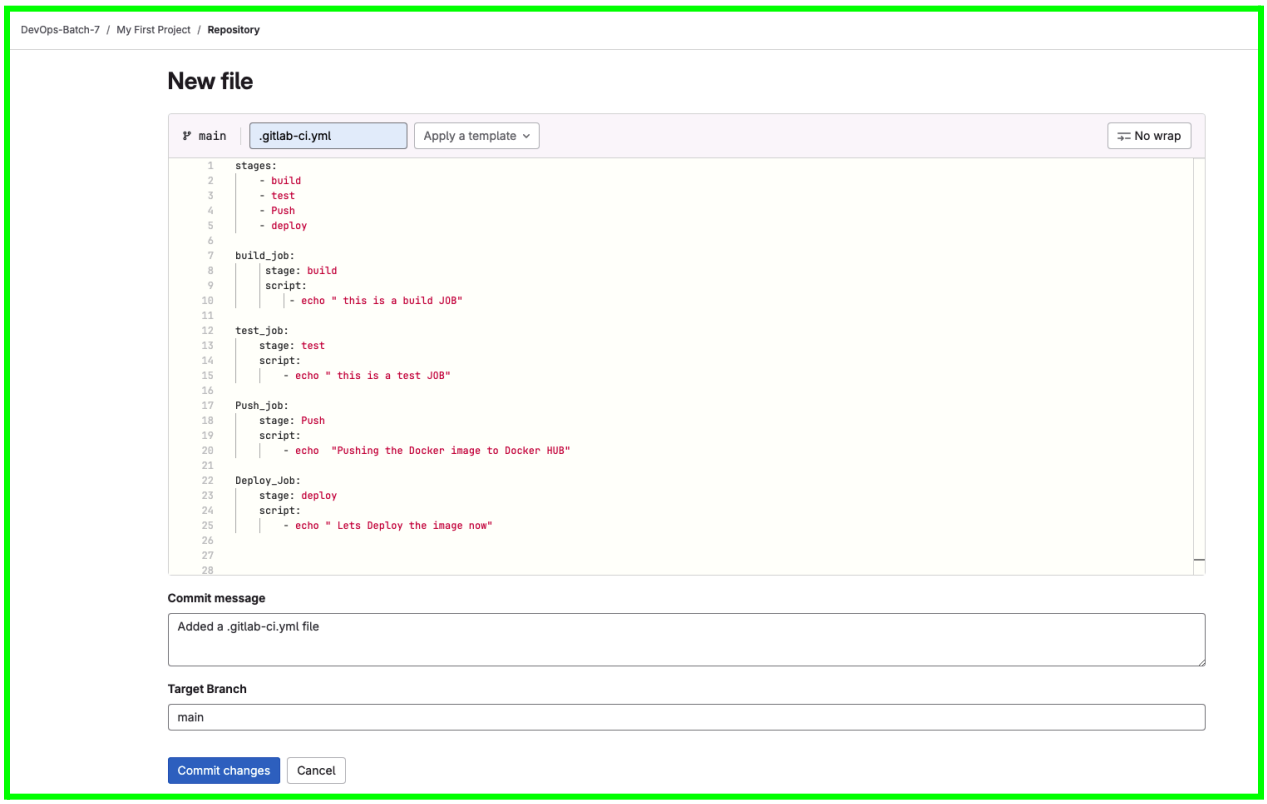


Project has been created successfully, Let's Create a **.gitlab-ci.yml** file to run a pipeline.

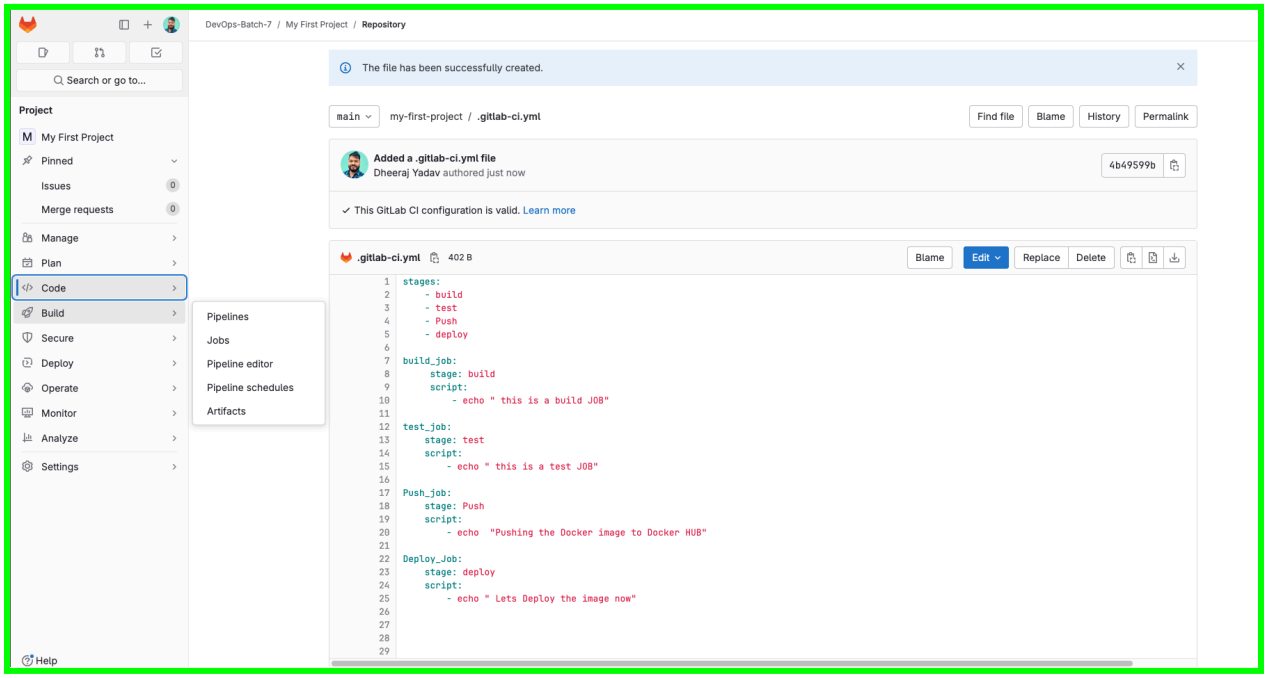
==========

Demo Pipeline

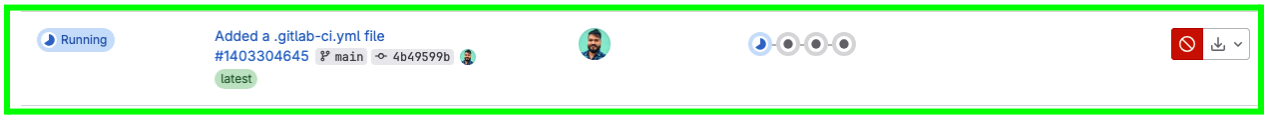
Here is the test Pipeline Config:

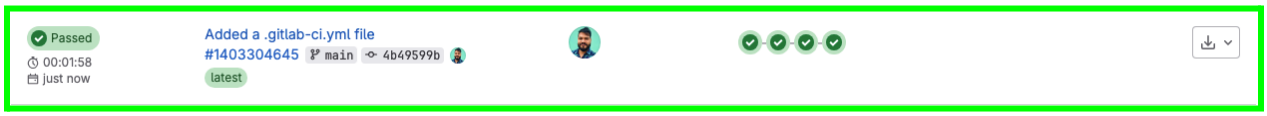


As soon as you click on Commit changes , it will trigger a **Build —> Pipelines**



Pipeline has been triggered & Completed successfully.





==========

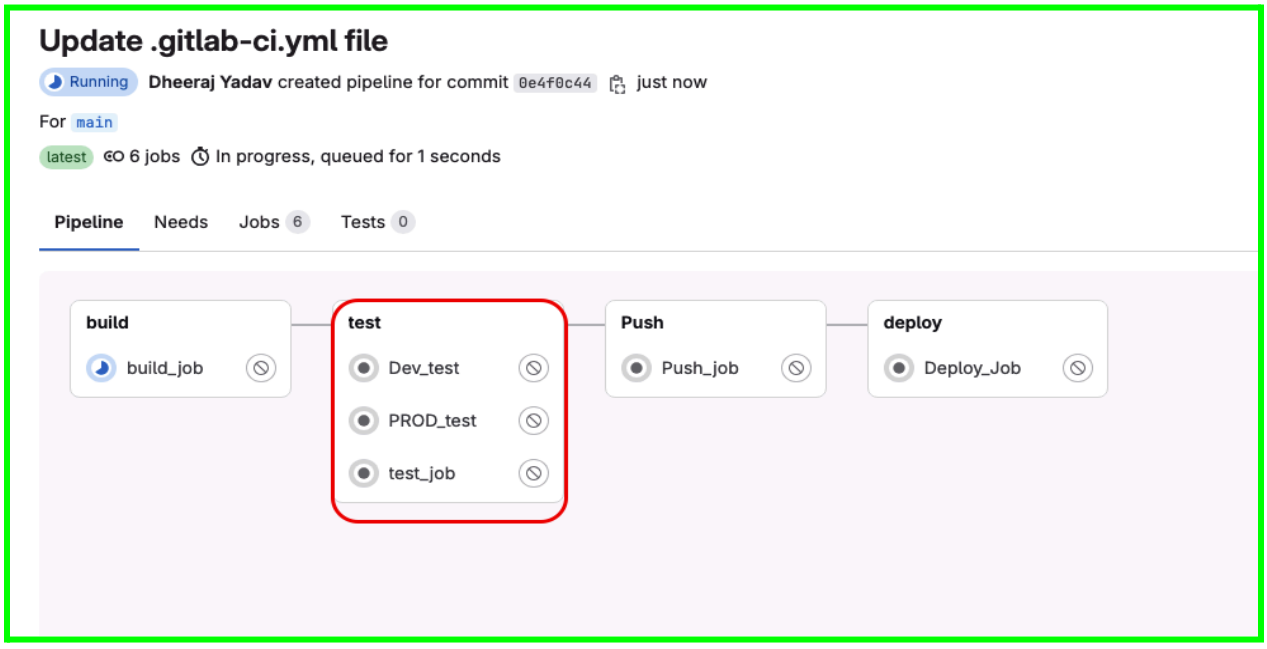
Parallel Jobs

Created a 3 test which will run parallel, as you can see below.

.gitlab-ci.yml 616 B

Blame Edit Replace Delete

```
stages:
    - build
    - test
    - Push
    - deploy

build_job:
     stage: build
     script:
         - echo " this is a build JOB for $CI_PROJECT_NAME"


test_job:
    stage: test
    script:
        - echo " this is a test JOB run for pipeline $CI_PIPELINE_NAME"


Push_job:
    stage: Push
    script:
        - echo  "Pushing the Docker image to Docker HUB"


Deploy_Job:
    stage: deploy
    script:
        - echo " Lets Deploy the image now"

Dev_test:
    stage: test
    script:
        echo "doing nore moe test"

PROD_test:
    stage: test
    script:
        echo "doing nore moe test"

```



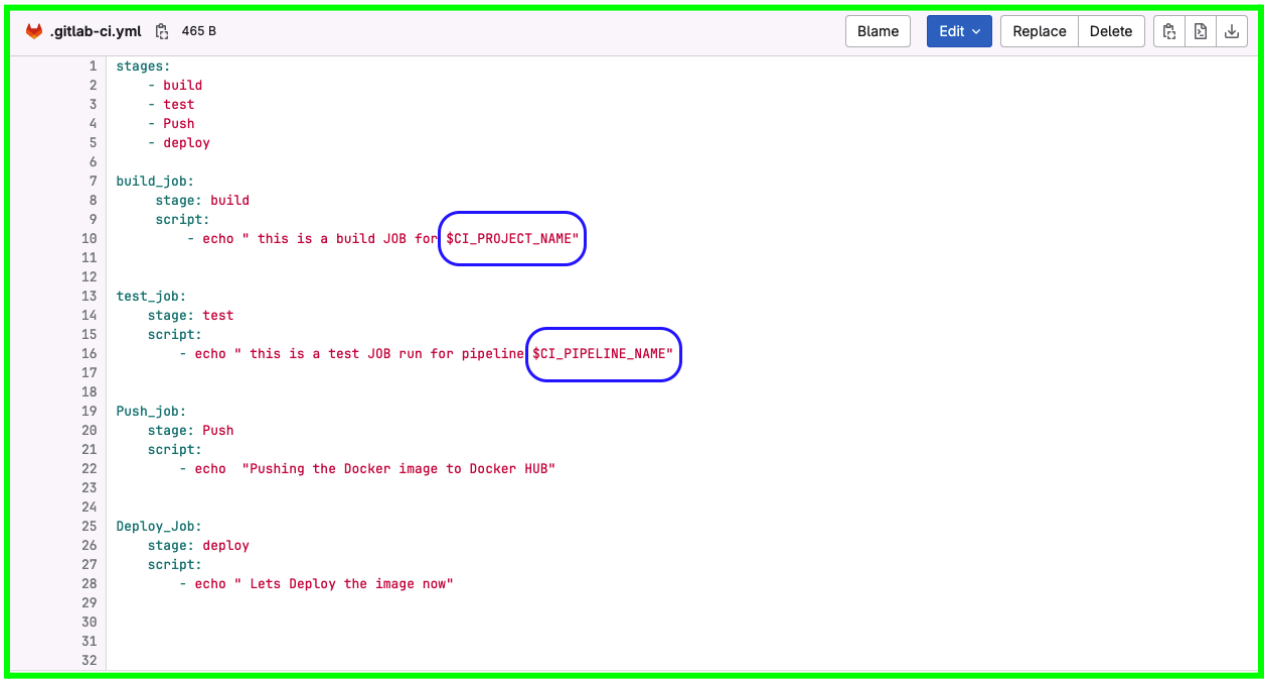
==========

Step-3: Variables Types

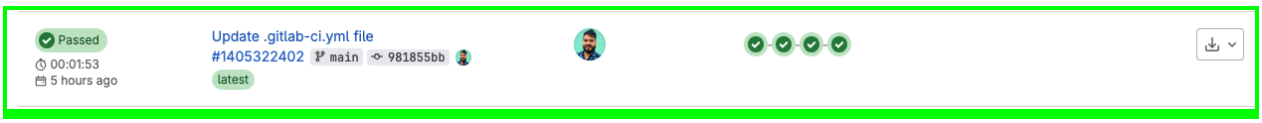
Predefined variables

Predefined variables become available at two different phases of pipeline execution. Some variables are available when GitLab creates the pipeline, and can be used to configure the pipeline or in job scripts. The other variables become available when a runner runs the job, and can only be used in job scripts.
These are defined by Gitlab, Let's add the Predefined Variables into our Pipeline.

Added the Predefined Variables into the Demo Pipelines.

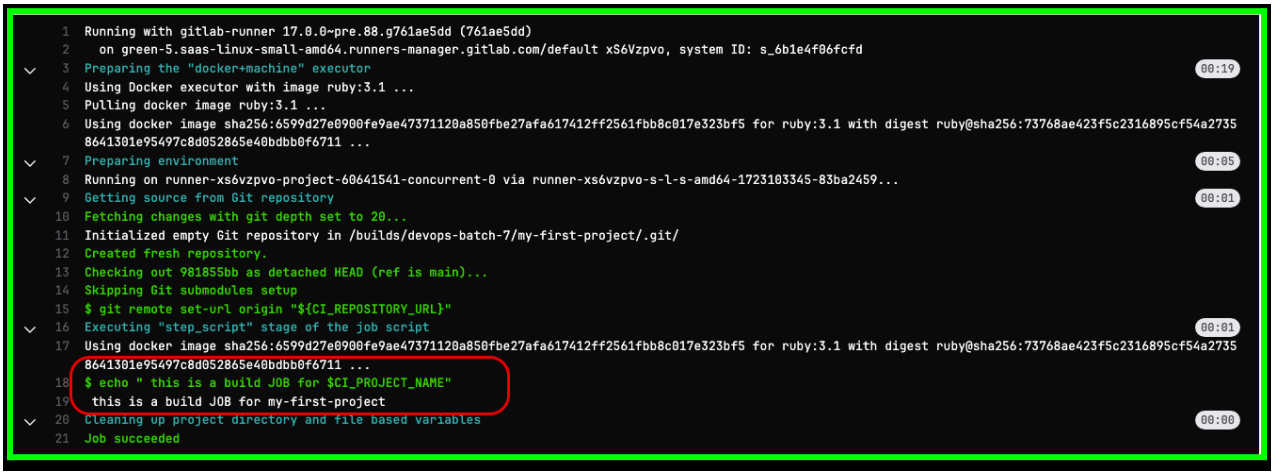


Ran the Pipeline & it's successful.



Let's check the Output, Build Output has Predefined Variable & we can see the Project Name showing in the Next Line.

This way we can use the Predefined Variable to your pipelines.



Predefined CI/CD Variables Reference

https://docs.gitlab.com/ee/ci/variables/predefined_variables.html

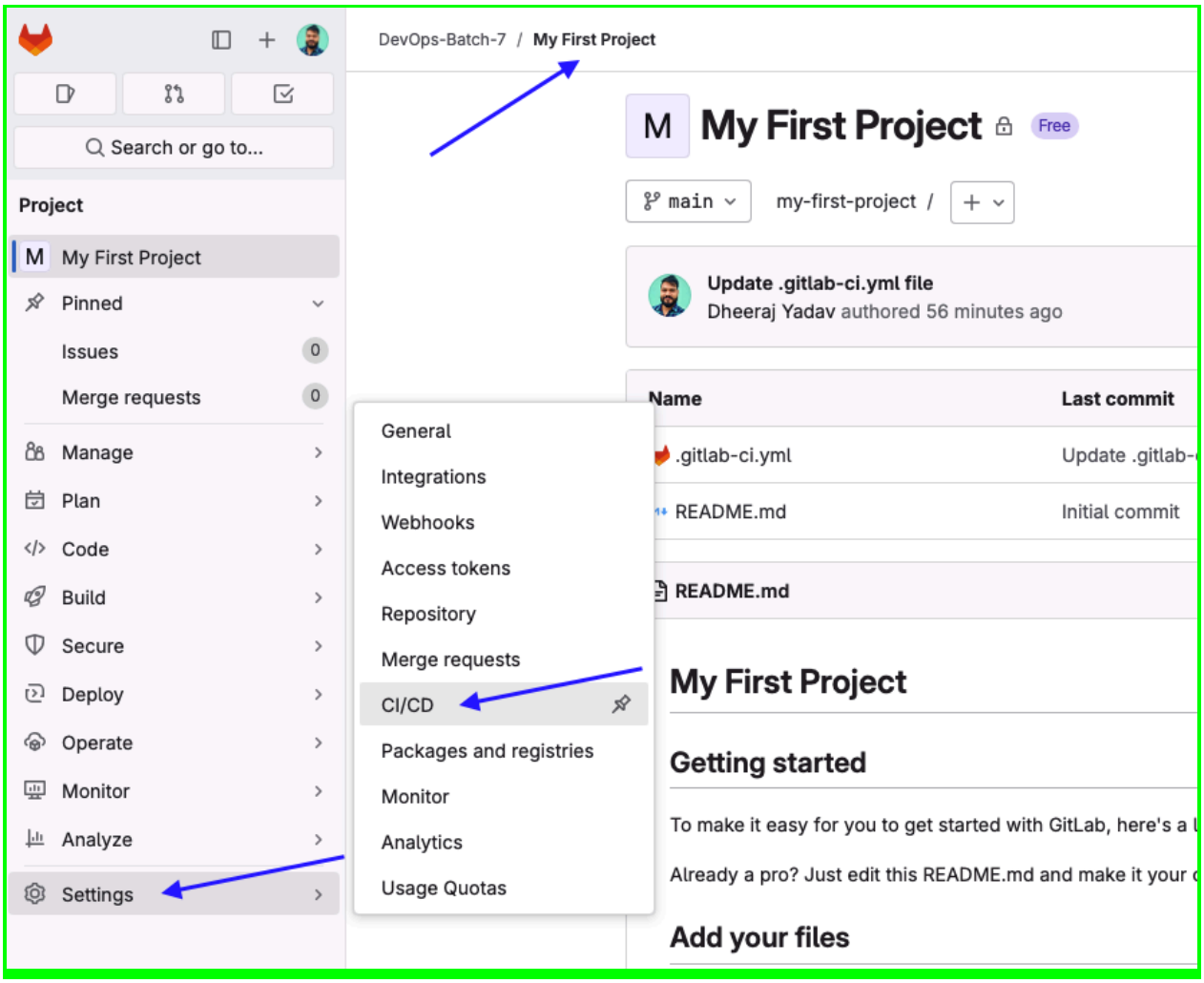
==========

User Defined Variables:

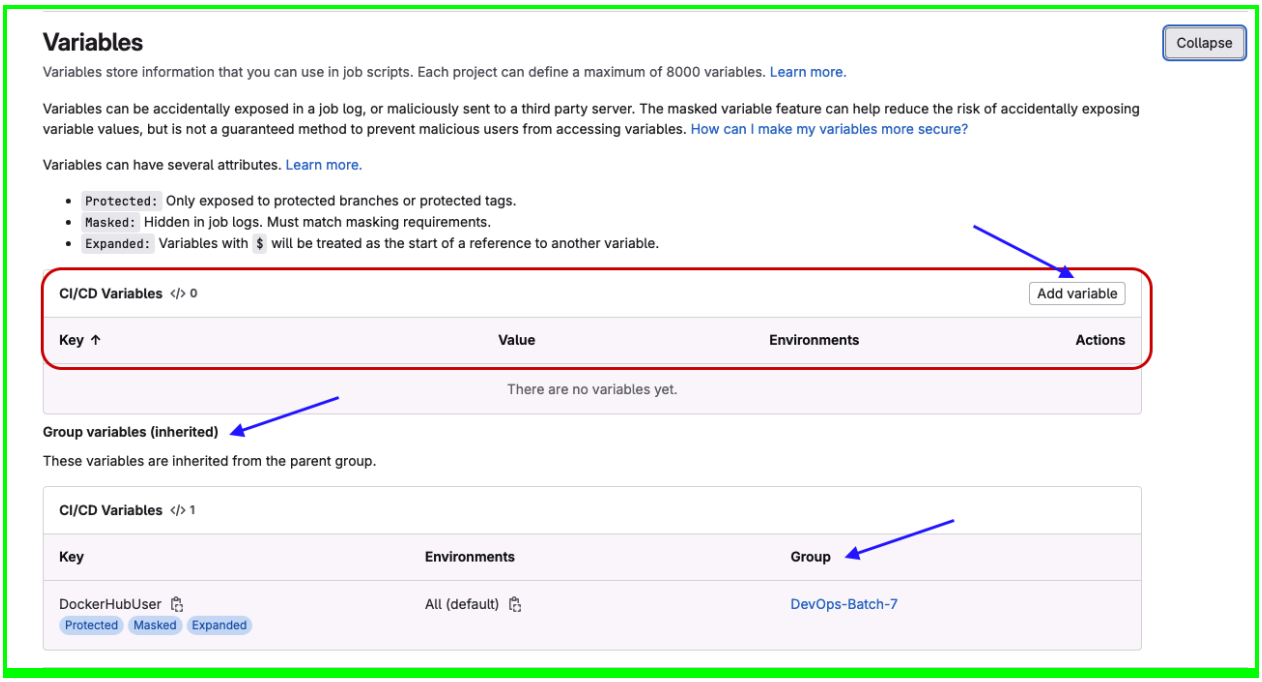
We can Define User Defined Variables either on **Project Level or Group Level.**

(a) Project Level Variable Define:

Under Project—-> Setting —-> CI/CD —> Variables —> Expand



Add the Variable by clicking on **Add Variables** & there, user Defined Variables will be defined on Project Level.

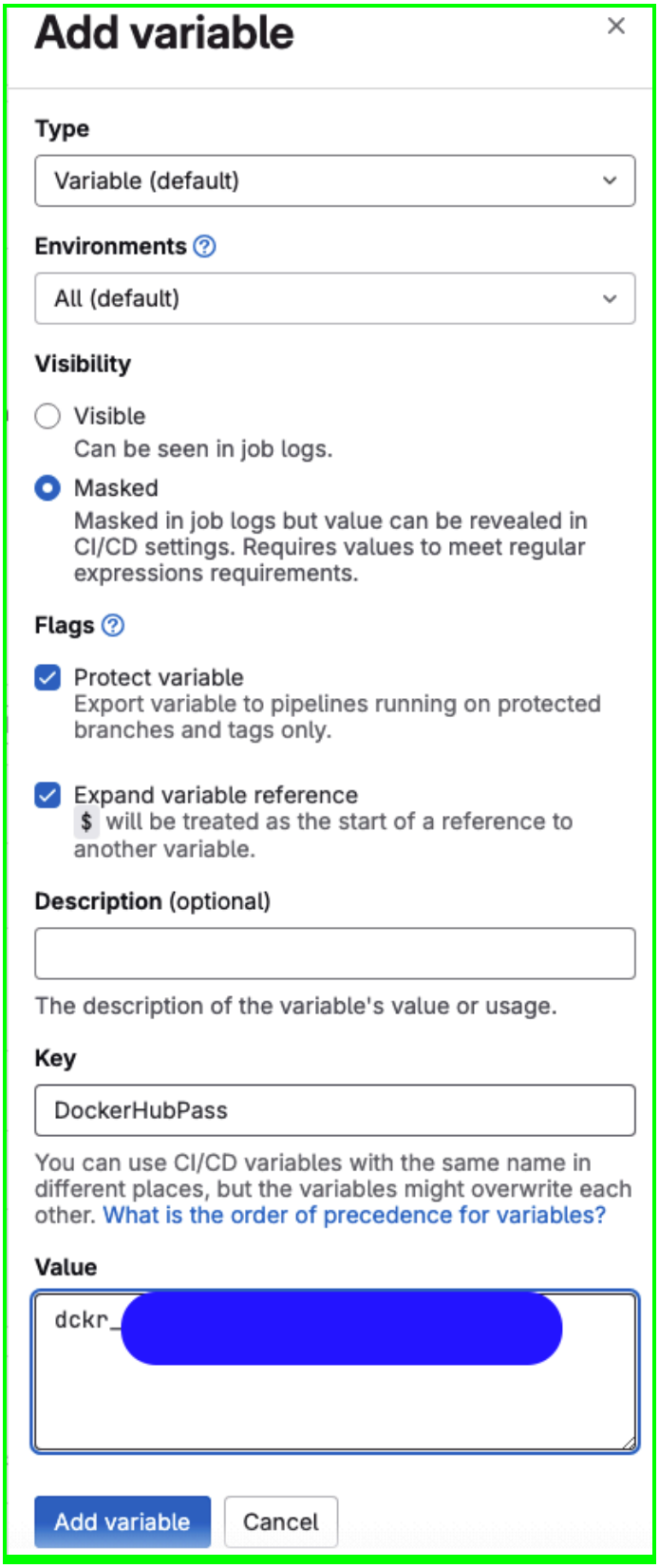


Add Variable:
We will be adding the DockerHub Username & Token to push the created docker image to DockerHub.

======
Key : Variable name “You can define
Value: Docker Hub token

Follow the below “**Generate the Personal Access Token on Docker Hub”** to generate the token.

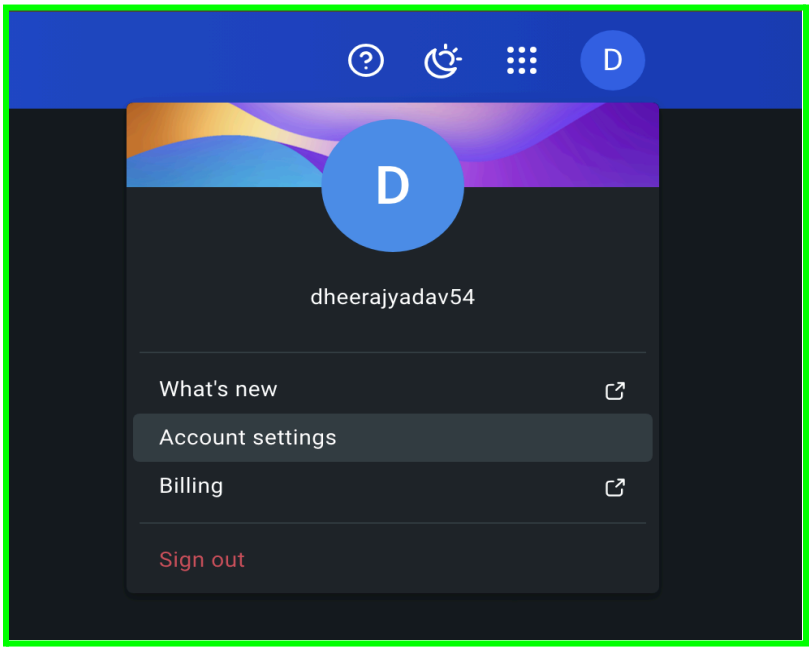
Dheeraj Yadav



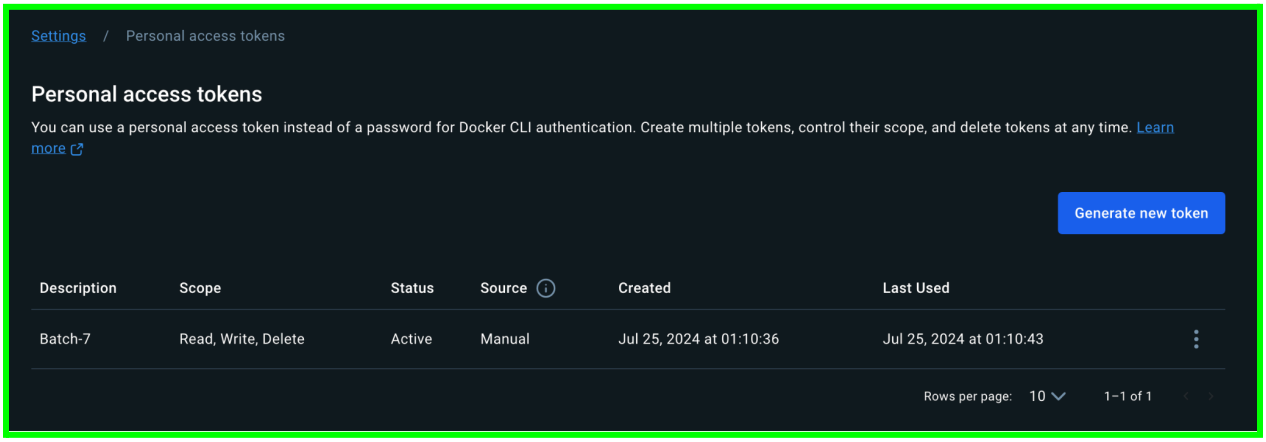
==========

Generate the Personal Access Token on Docker Hub

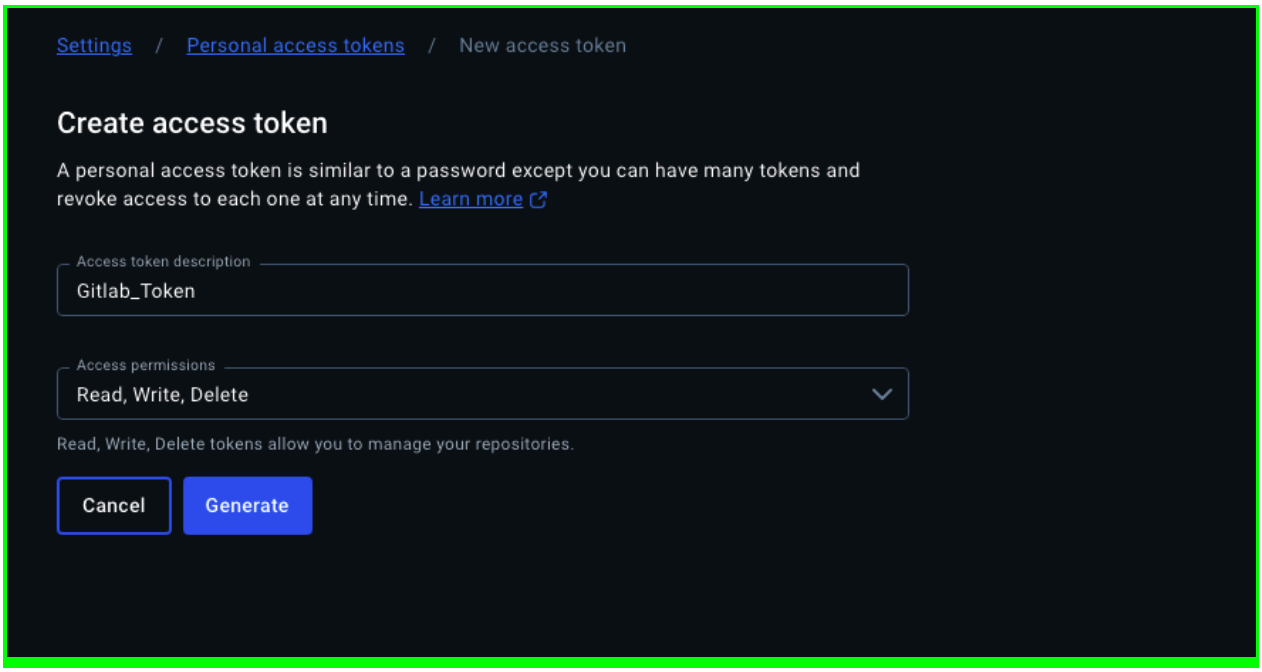
a) Docker Hub Account —-> Account Setting



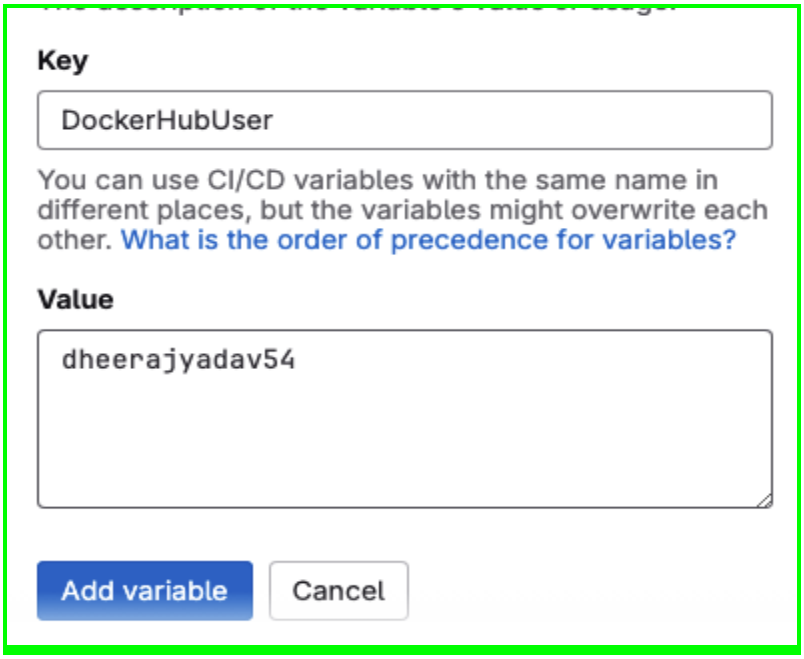
b) Personal access tokens → Generate new token



Click on Generate & Copy the token. Once you Copied the Token, go to Jenkins Credentials page & paste under Password Section.

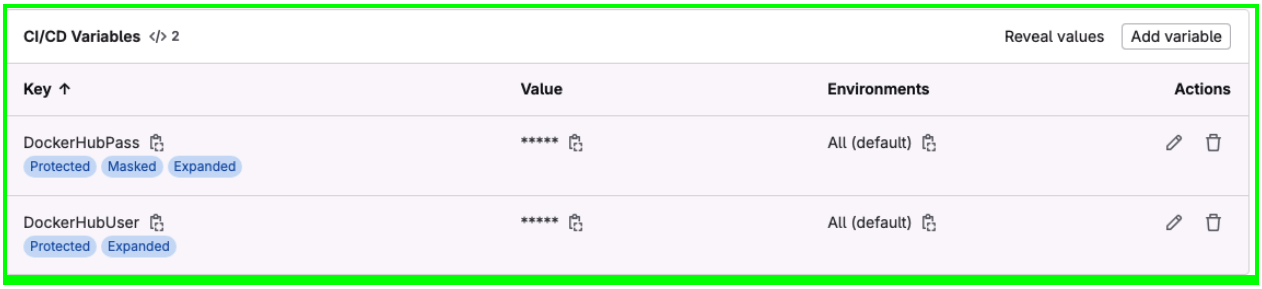


Added the DockerHub User Variable.



Added both DockerHubUser, DockerHubPass User defined Variables, Let's use these in the pipelines.

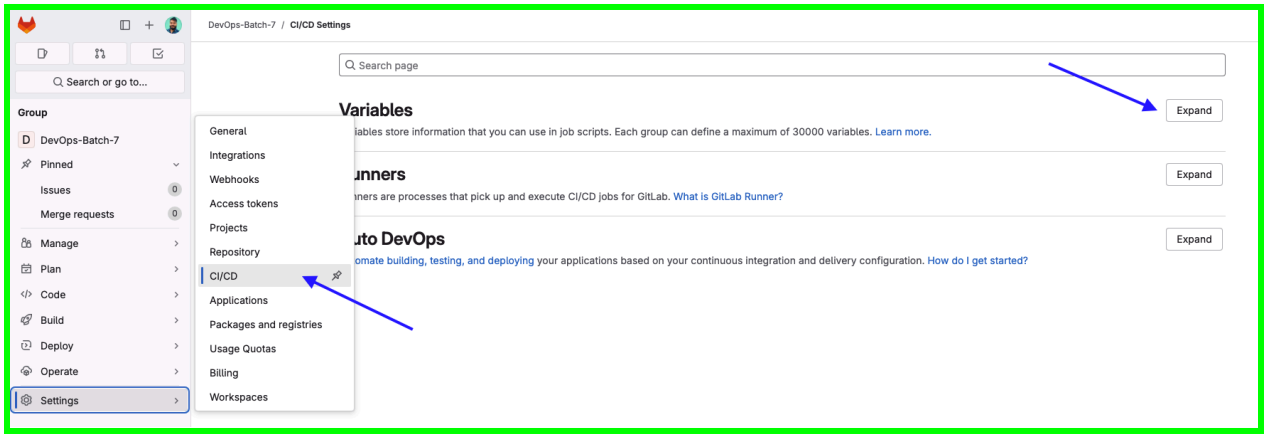
Note: These are User Defined, so can be any Variable Name.



(b) Group Level Variable Define:

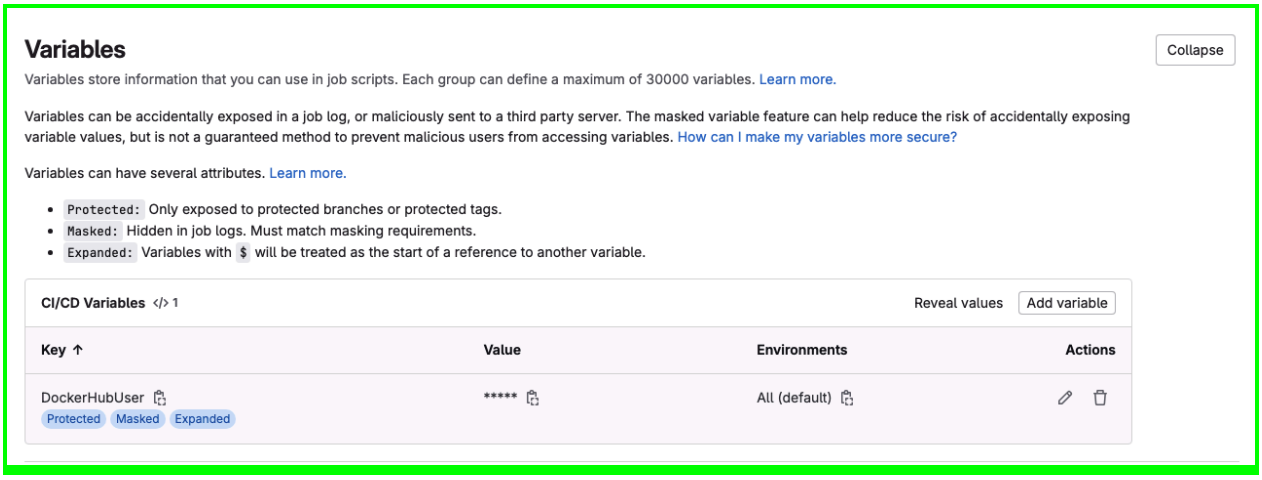
Creating the Variables, follow the same process as shown on Project Level.

Group—> Setting → CI/CD —> Variables → Expand



Dheeraj Yadav

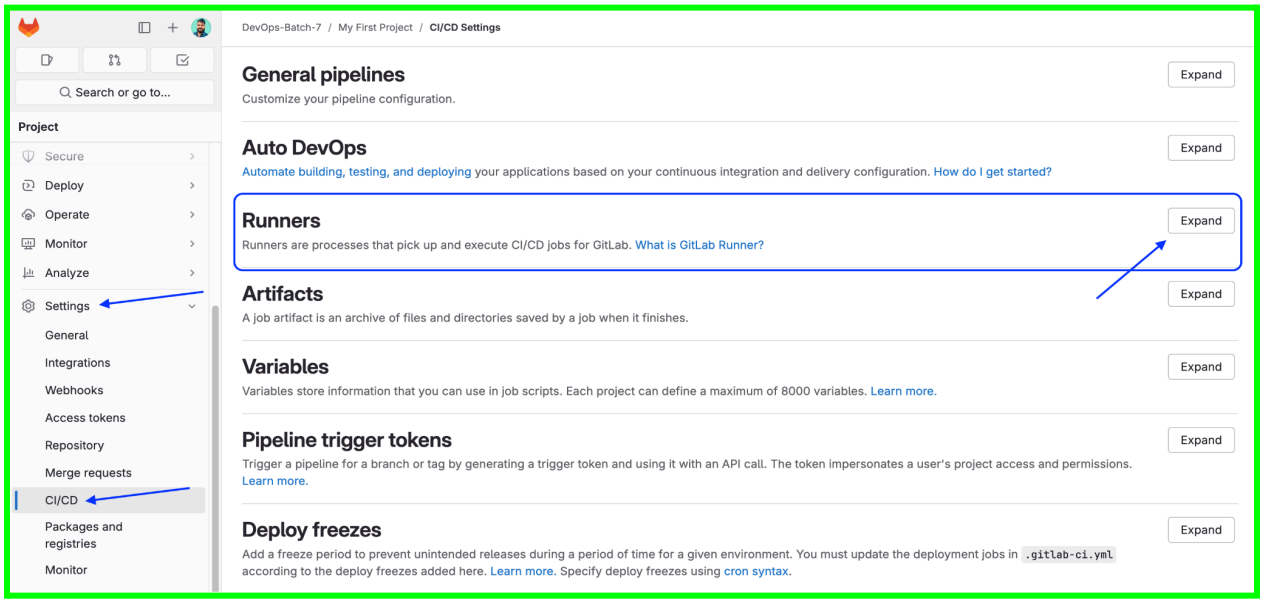
Once you add the Variables on Group Level, it will be assigned to all your Projects.



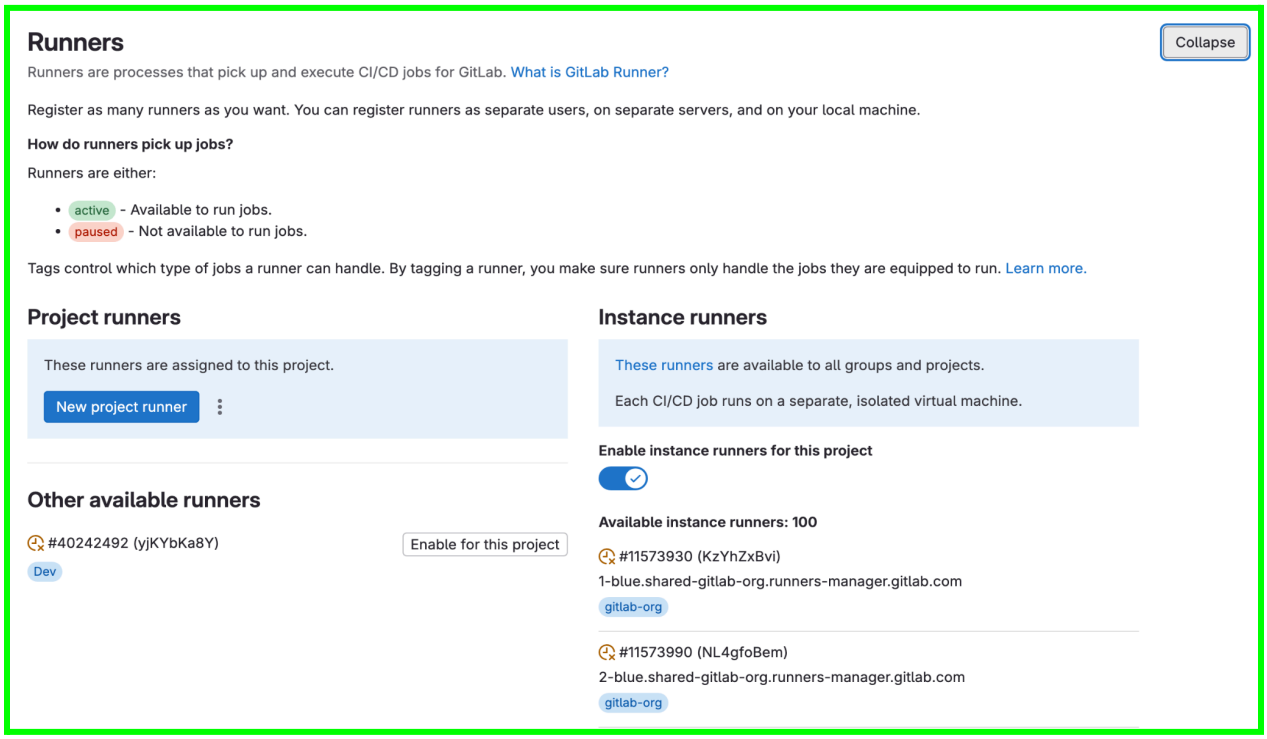
==========

Step-4 Gitlab Runners

Click on Setting—> CICD —> Runners —> Click on Expand

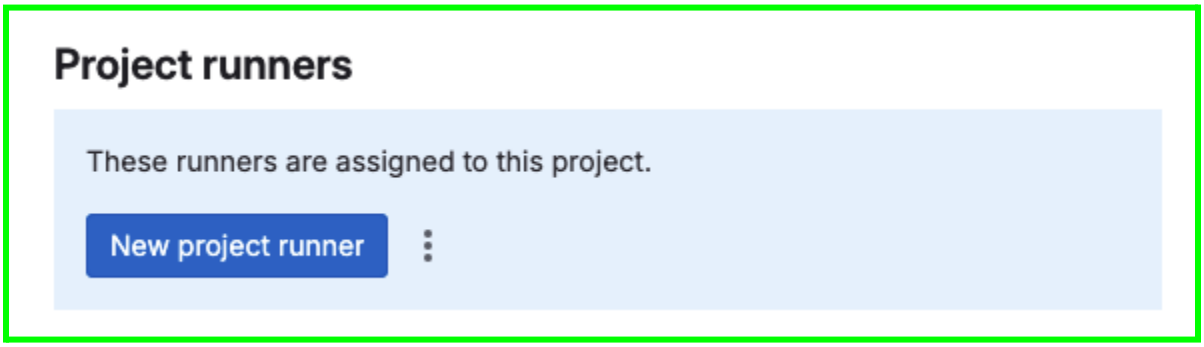


You can Create your Own runner or you can use “Instance Runners” provided by Gitlab for free.

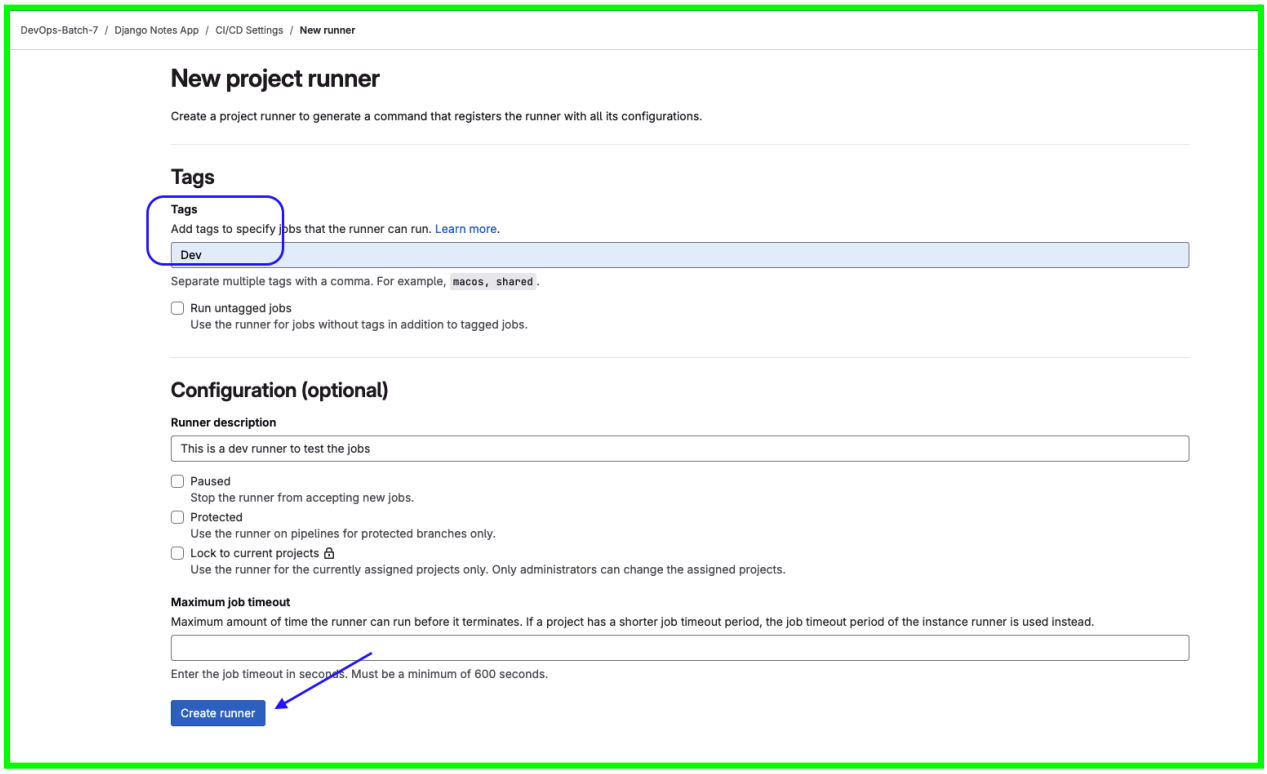


These are the shared Gitlab Provided runners.

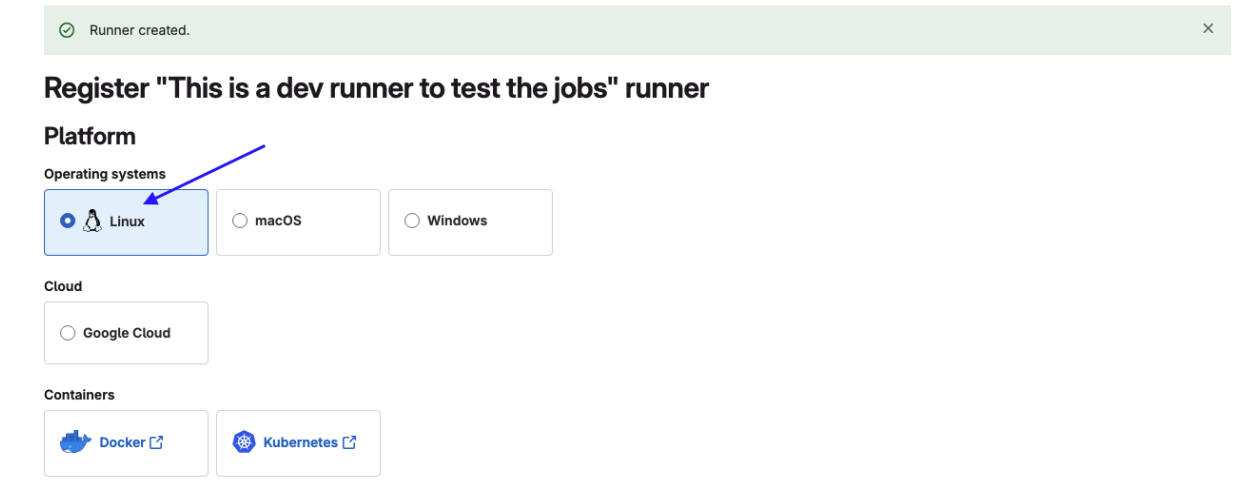
Let's add the "New Project Runner"



 # Tag : Value which we will define in the Pipeline to instruct which runner to use.

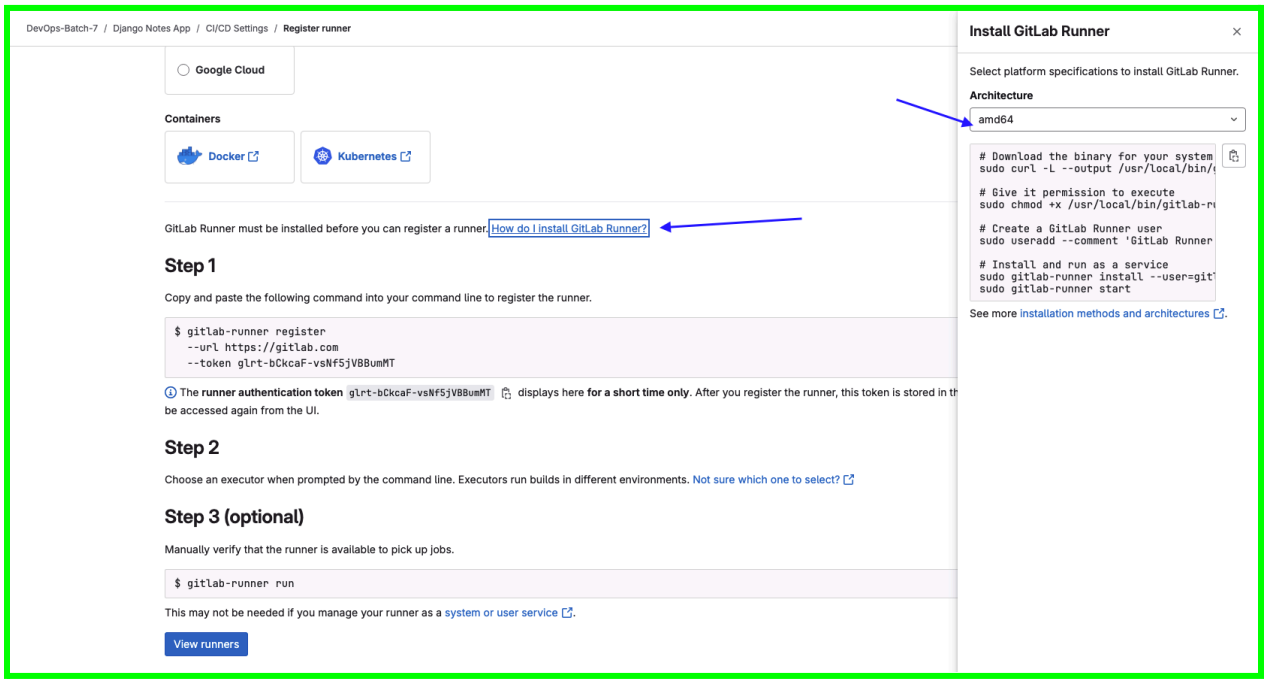


Note : We can use different platforms to run runners, but I will be using Linux Platform.



Execute the below steps on Linux EC2 Instance to register your Gitlab Runner.

Click on **“How do i install Gitlab runner”**



==========

Install the Gitlab Runner

Run the below command on an EC2 Linux instance to install the Gitlab Runner.

Download the binary for your system

sudo curl -L --output /usr/local/bin/gitlab-runner https://gitlab-runner-downloads.s3.amazonaws.com/latest/binaries/gitlab-runner-linux-amd64

Give it permission to execute

sudo chmod +x /usr/local/bin/gitlab-runner

Create a GitLab Runner user

sudo useradd --comment 'GitLab Runner' --create-home gitlab-runner --shell /bin/bash

# Install and run as a service
sudo gitlab-runner install --user=gitlab-runner --working-directory=/home/gitlab-runner
sudo gitlab-runner start

```
ubuntu@ip-172-31-35-243:~$ # Download the binary for your system
sudo curl -L --output /usr/local/bin/gitlab-runner https://gitlab-runner-downloads.s3.amazonaws.com/latest/binaries/gitlab-runner-linux-amd64

# Give it permission to execute
sudo chmod +x /usr/local/bin/gitlab-runner

# Create a GitLab Runner user
sudo useradd --comment 'GitLab Runner' --create-home gitlab-runner --shell /bin/bash

# Install and run as a service
sudo gitlab-runner install --user=gitlab-runner --working-directory=/home/gitlab-runner
sudo gitlab-runner start
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 65.5M  100 65.5M    0     0  12.8M      0  0:00:05  0:00:05 --:--:-- 15.6M
Runtime platform                                    arch=amd64 os=linux pid=10390 revision=9882d9c7 version=17.2.1
Runtime platform                                    arch=amd64 os=linux pid=10493 revision=9882d9c7 version=17.2.1
ubuntu@ip-172-31-35-243:~$
ubuntu@ip-172-31-35-243:~$
ubuntu@ip-172-31-35-243:~$
```

==========

Gitlab runner status

To Check the gitlab runner status, you have to be a root user else it will give you a below warning.

```
ubuntu@ip-172-31-35-243:~$ gitlab-runner status
Runtime platform                                    arch=amd64 os=linux pid=11044 revision=9882d9c7 version=17.2.1
FATAL: The --user is not supported for non-root users
ubuntu@ip-172-31-35-243:~$ gitlab-runner status
Runtime platform                                    arch=amd64 os=linux pid=11047 revision=9882d9c7 version=17.2.1
FATAL: The --user is not supported for non-root users
ubuntu@ip-172-31-35-243:~$
ubuntu@ip-172-31-35-243:~$
ubuntu@ip-172-31-35-243:~$ sudo su
root@ip-172-31-35-243:/home/ubuntu#
root@ip-172-31-35-243:/home/ubuntu#
root@ip-172-31-35-243:/home/ubuntu#
root@ip-172-31-35-243:/home/ubuntu# gitlab-runner status
Runtime platform                                    arch=amd64 os=linux pid=11062 revision=9882d9c7 version=17.2.1
gitlab-runner: Service is running
root@ip-172-31-35-243:/home/ubuntu#
```

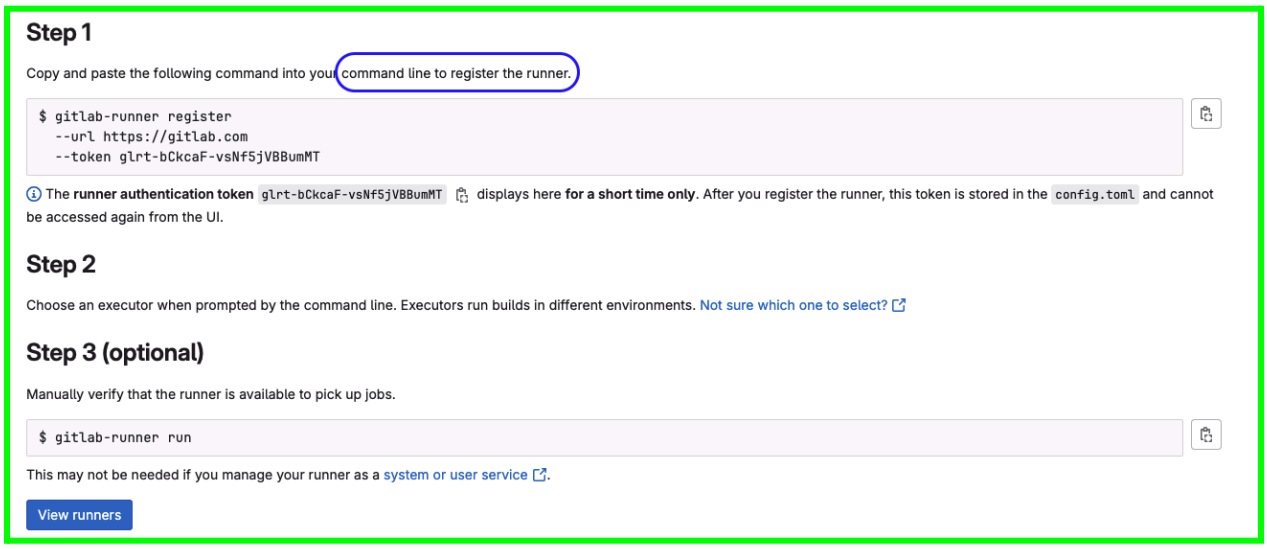
==========

Gitlab runner Service status

```
root@ip-172-31-35-243:/home/ubuntu# systemctl status gitlab-runner
● gitlab-runner.service - GitLab Runner
     Loaded: loaded (/etc/systemd/system/gitlab-runner.service; enabled; preset: enabled)
     Active: active (running) since Fri 2024-08-09 08:55:34 UTC; 25min ago
   Main PID: 10527 (gitlab-runner)
      Tasks: 6 (limit: 1130)
     Memory: 11.8M (peak: 18.1M)
        CPU: 274ms
     CGroup: /system.slice/gitlab-runner.service
             └─10527 /usr/local/bin/gitlab-runner run --working-directory /home/gitlab-runner --config /etc/gitlab-runner/config.toml --servic>

Aug 09 08:55:34 ip-172-31-35-243 systemd[1]: Started gitlab-runner.service - GitLab Runner.
Aug 09 08:55:34 ip-172-31-35-243 gitlab-runner[10527]: Runtime platform                                    arch=amd64 os=linux pid=10527 revis>
Aug 09 08:55:34 ip-172-31-35-243 gitlab-runner[10527]: Starting multi-runner from /etc/gitlab-runner/config.toml...  builds=0 max_builds=0
Aug 09 08:55:34 ip-172-31-35-243 gitlab-runner[10527]: Running in system-mode.
Aug 09 08:55:34 ip-172-31-35-243 gitlab-runner[10527]:
Aug 09 08:55:34 ip-172-31-35-243 gitlab-runner[10527]: Created missing unique system ID                   system_id=s_b5bf743c916f
Aug 09 08:55:34 ip-172-31-35-243 gitlab-runner[10527]: Configuration loaded                               builds=0 max_builds=1
Aug 09 08:55:34 ip-172-31-35-243 gitlab-runner[10527]: listen_address not defined, metrics & debug endpoints disabled  builds=0 max_builds=1
Aug 09 08:55:34 ip-172-31-35-243 gitlab-runner[10527]: [session_server].listen_address not defined, session endpoints disabled  builds=0 max_b>
Aug 09 08:55:34 ip-172-31-35-243 gitlab-runner[10527]: Initializing executor providers                    builds=0 max_builds=1
root@ip-172-31-35-243:/home/ubuntu#
root@ip-172-31-35-243:/home/ubuntu#
root@ip-172-31-35-243:/home/ubuntu#
root@ip-172-31-35-243:/home/ubuntu#
```

==========

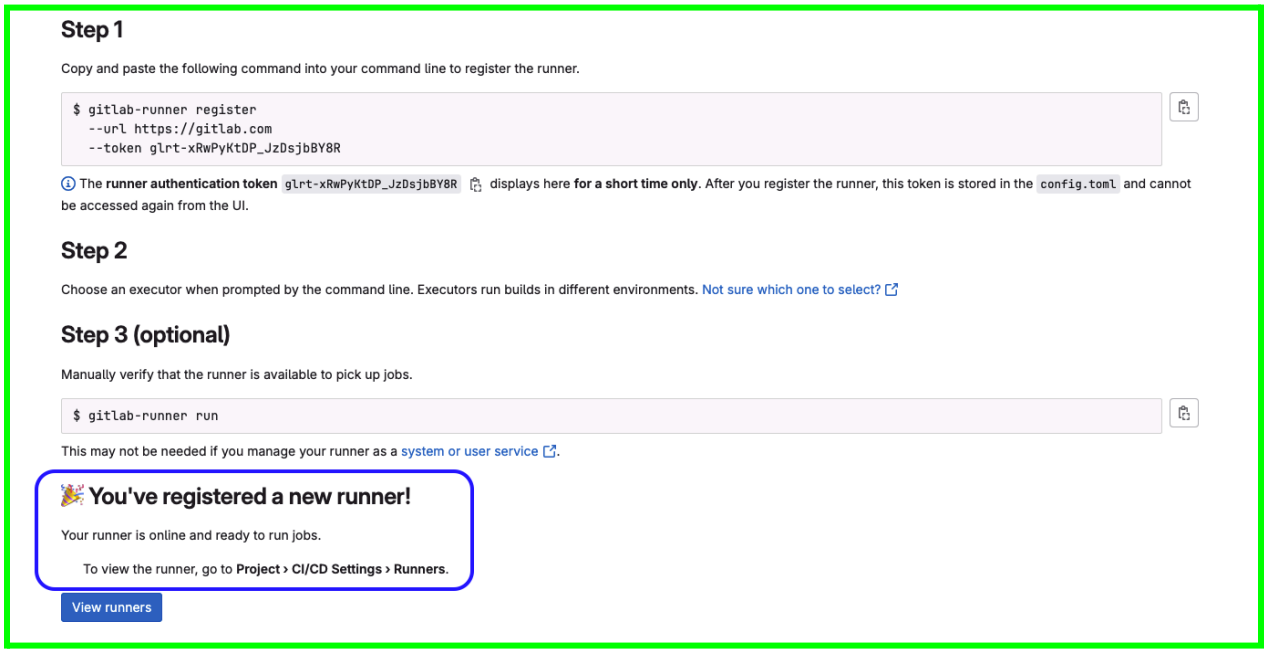
Register the Gitlab Runner



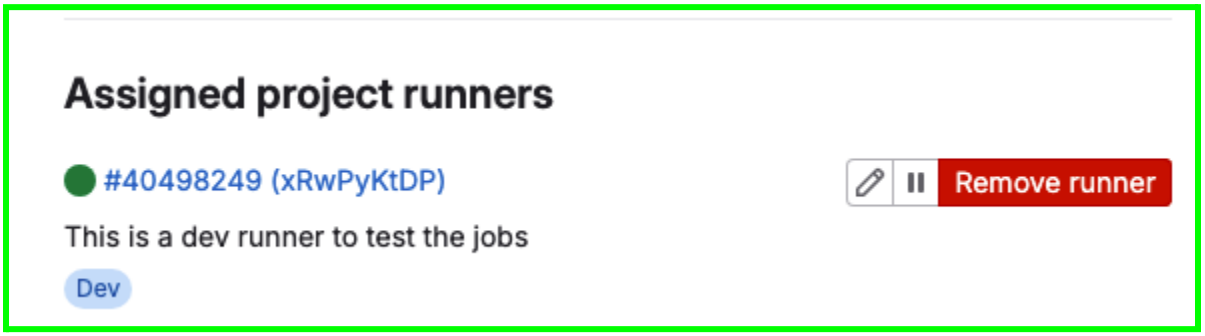
Run Step-1 Command to register the runner with your Project.

```
ubuntu@ip-172-31-35-243:~$
ubuntu@ip-172-31-35-243:~$ gitlab-runner register  --url https://gitlab.com  --token glrt-xRwPyKtDP_JzDsjbBY8R
Runtime platform                                    arch=amd64 os=linux pid=11000 revision=9882d9c7 version=17.2.1
WARNING: Running in user-mode.
WARNING: The user-mode requires you to manually start builds processing:
WARNING: $ gitlab-runner run
WARNING: Use sudo for system-mode:
WARNING: $ sudo gitlab-runner...

Enter the GitLab instance URL (for example, https://gitlab.com/):
[https://gitlab.com]:
Verifying runner... is valid                        runner=xRwPyKtDP
Enter a name for the runner. This is stored only in the local config.toml file:
[ip-172-31-35-243]: runner-dev
Enter an executor: docker-autoscaler, instance, ssh, virtualbox, kubernetes, docker, docker-windows, docker+machine, custom, shell, parallels:
shell
Runner registered successfully. Feel free to start it, but if it's running already the config should be automatically reloaded!

Configuration (with the authentication token) was saved in "/home/ubuntu/.gitlab-runner/config.toml"
ubuntu@ip-172-31-35-243:~$
```

After Step-1 Command, runner registered with the Project.



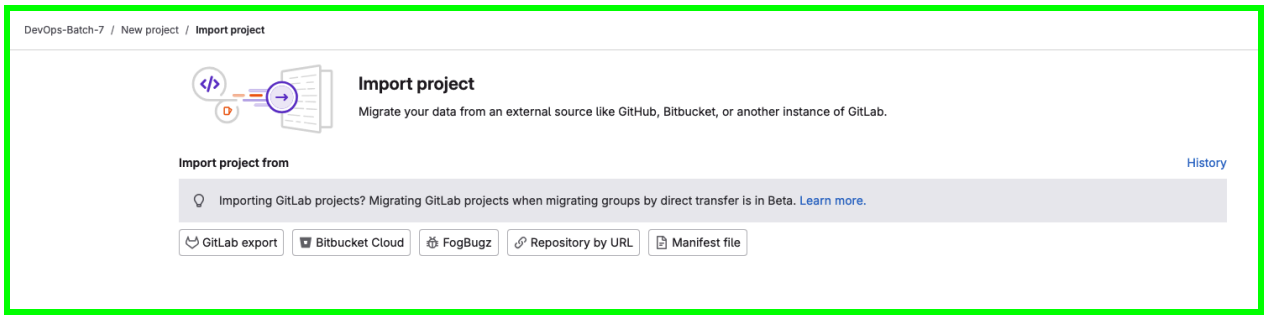
Click on View Runner, here you can see newly created runner with Tag Name : Dev



==========

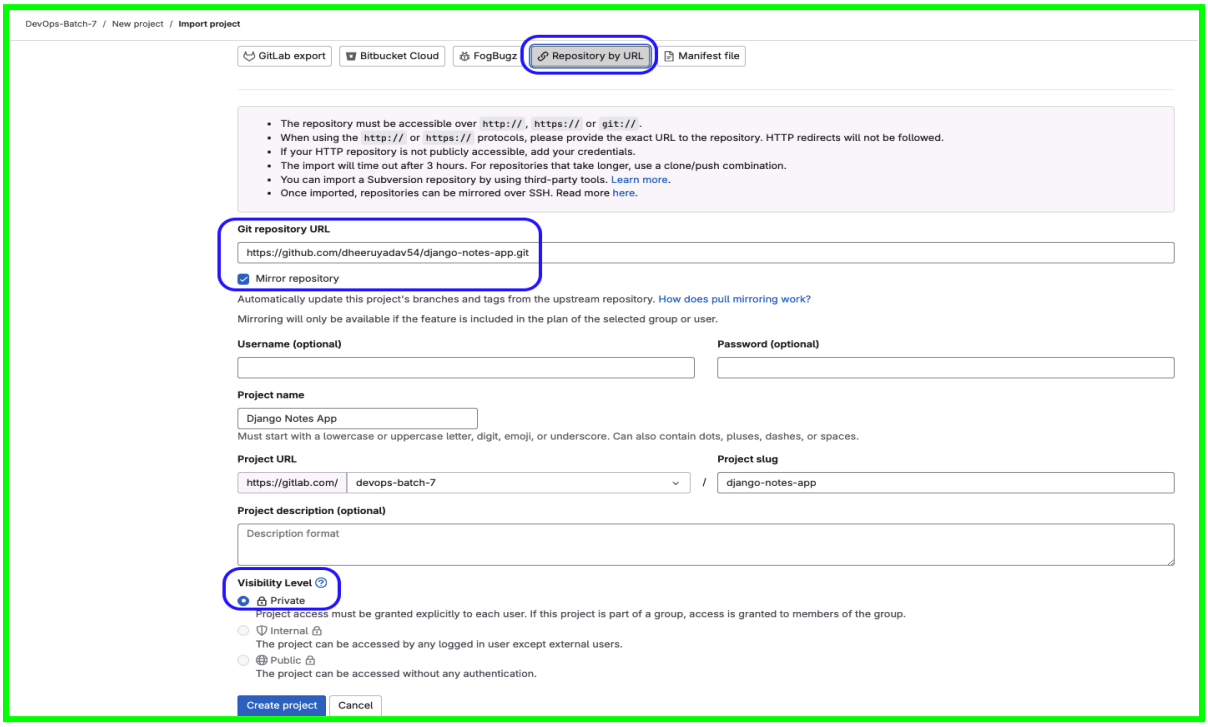
Step-5: Project Import from

Here are the Options to import your projects into Gitlab.

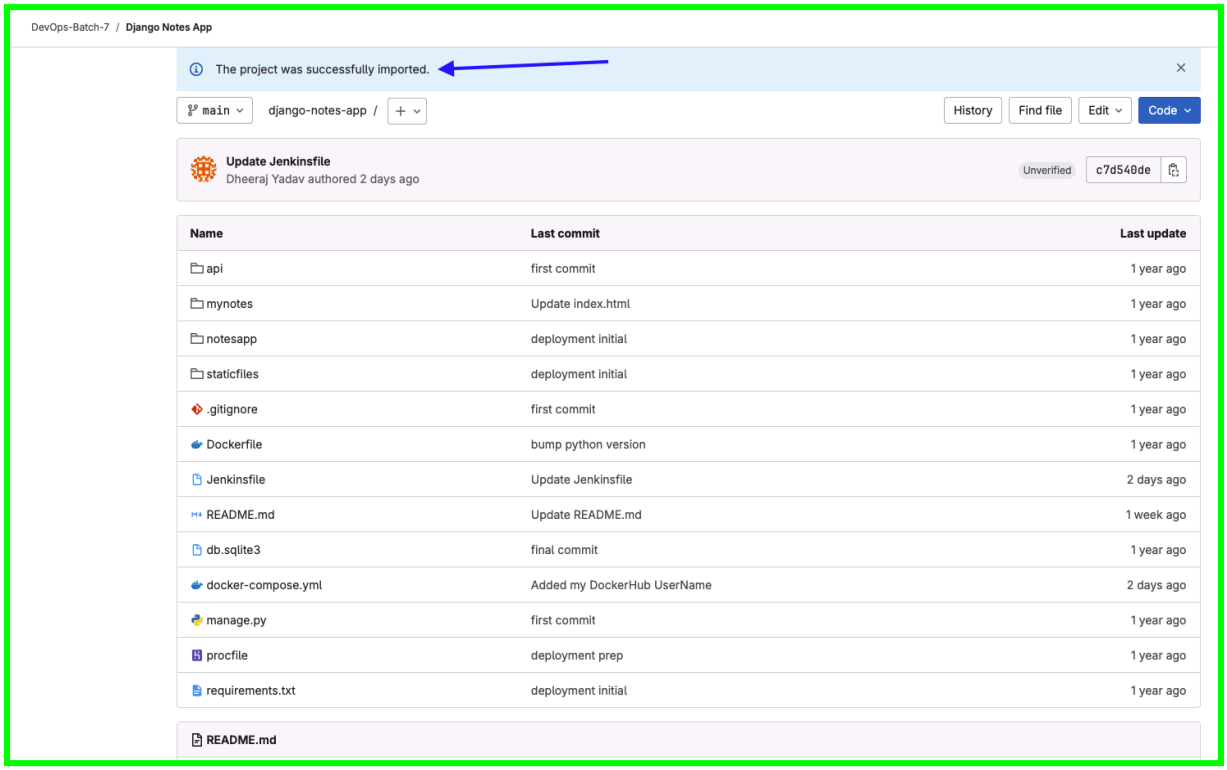


We will be using **Repository by URL** Option which is imported from GitHub.

Mirror Repository : Check this box hence no need for Manual Update, it will be Mirroring the changes from Source "**GitHub**"



Project has been imported successfully.



==========

SSH Key-Gen on EC2 Machine

$ ssh-keygen : Use this command to generate the Private & Public Key Pair on EC2 Instance.

```
ubuntu@ip-172-31-35-243:~$ ssh-keygen
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/ubuntu/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ubuntu/.ssh/id_ed25519
Your public key has been saved in /home/ubuntu/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:1OiF0TSSLrvDG+0nWjrX8D2XhvXGRIzjo0BN7dA2ies ubuntu@ip-172-31-35-243
The key's randomart image is:
+--[ED25519 256]--+
|        o+o  + . |
|        .*..+ *  |
|       .+ oo = + |
|      .o... o + o|
|       oS. . . o |
|      ... . E + .|
|     ...o+ o + * |
|      ==o + = + +|
|      +*.o   + . |
+----[SHA256]-----+
ubuntu@ip-172-31-35-243:~$
ubuntu@ip-172-31-35-243:~$
```

Copy the Public key from EC2 Instance to gitlab Account

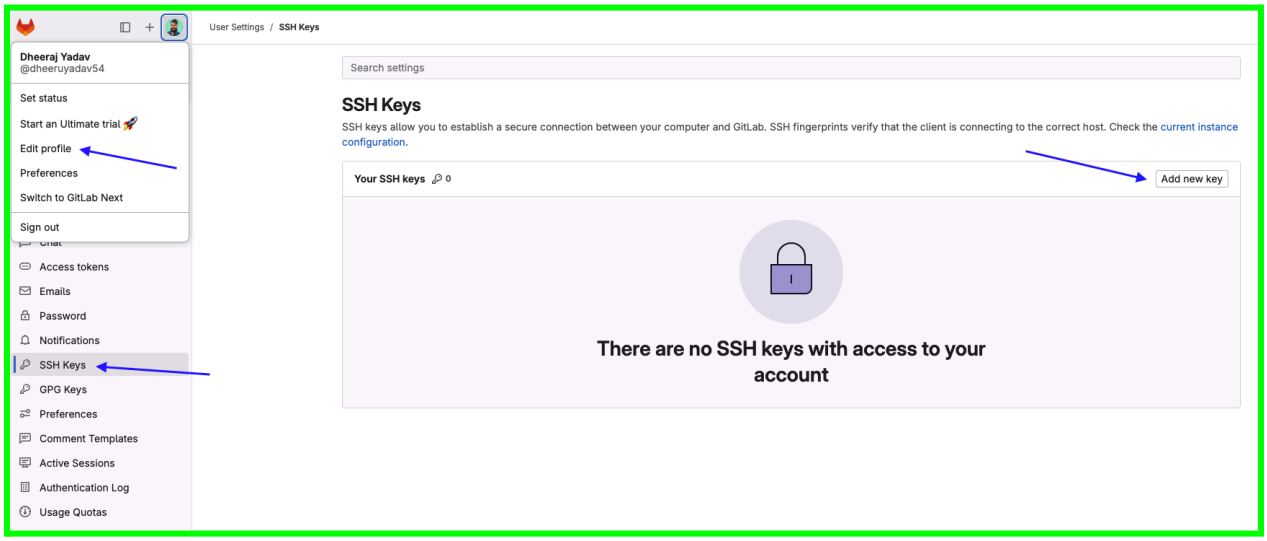
cat /home/ubuntu/.ssh/id_ed25519.pub

```
ubuntu@ip-172-31-35-243:~/.ssh$ pwd
/home/ubuntu/.ssh
ubuntu@ip-172-31-35-243:~/.ssh$ cat id_ed25519.pub
ssh-ed25519 AAAA[redacted]u@ip-172-31-35-243
ubuntu@ip-172-31-35-243:~/.ssh$
```

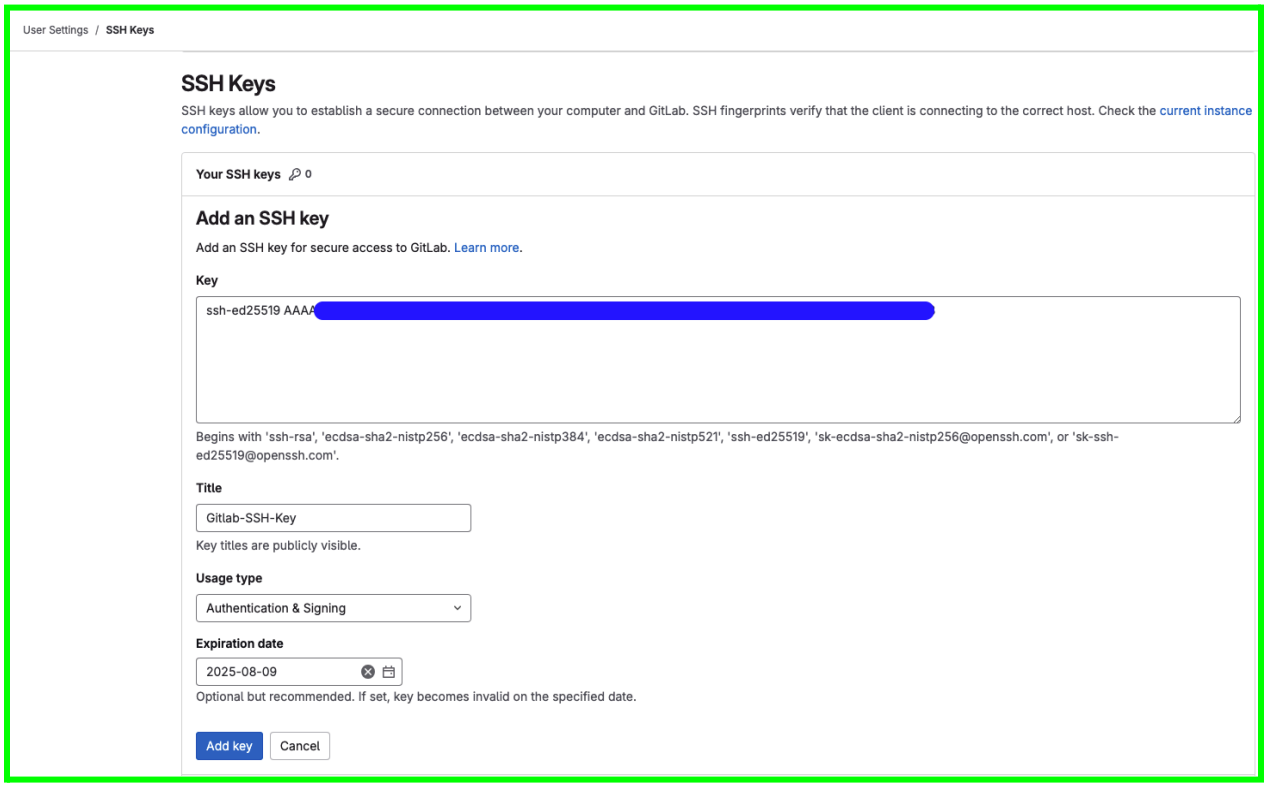
==========

Add the Public Key to Gitlab Account

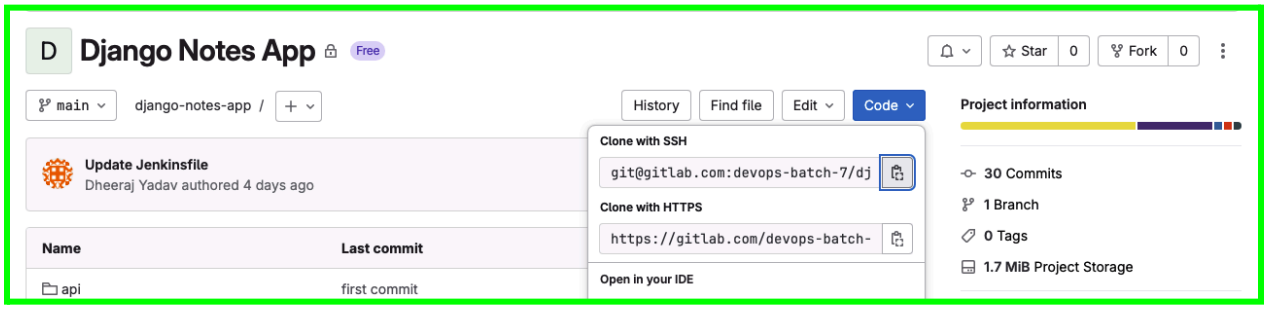
Click on your profile icon —-> Edit Profile —-> SSH Keys —-> Add new key



Paste the Copied Public key here & hit add key.



$ git clone git@gitlab.com:devops-batch-7/django-notes-app.git : Use SSH URL



After adding the Public key into gitlab, we can clone our project locally on EC2 instance.

```
ubuntu@ip-172-31-35-243:~$
ubuntu@ip-172-31-35-243:~$ git clone git@gitlab.com:devops-batch-7/django-notes-app.git
Cloning into 'django-notes-app'...
The authenticity of host 'gitlab.com (172.65.251.78)' can't be established.
ED25519 key fingerprint is SHA256:eUXGGm1YGsMAS7vkcx6JOJdOGHPem5gQp4taiCfCLB8.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'gitlab.com' (ED25519) to the list of known hosts.
remote: Enumerating objects: 379, done.
remote: Counting objects: 100% (379/379), done.
remote: Compressing objects: 100% (272/272), done.
remote: Total 379 (delta 102), reused 379 (delta 102), pack-reused 0 (from 0)
Receiving objects: 100% (379/379), 1.69 MiB | 3.90 MiB/s, done.
Resolving deltas: 100% (102/102), done.
ubuntu@ip-172-31-35-243:~$
```

Let's create a .gitlab-ci.yml file

```
ubuntu@ip-172-31-35-243:~$ ls
apps  django-notes-app
ubuntu@ip-172-31-35-243:~$
ubuntu@ip-172-31-35-243:~$ cd django-notes-app/
ubuntu@ip-172-31-35-243:~/django-notes-app$
ubuntu@ip-172-31-35-243:~/django-notes-app$
ubuntu@ip-172-31-35-243:~/django-notes-app$ ls
Dockerfile  Jenkinsfile  README.md  api  db.sqlite3  docker-compose.yml  manage.py  mynotes  notesapp  procfile  requirements.txt  staticfiles
ubuntu@ip-172-31-35-243:~/django-notes-app$
ubuntu@ip-172-31-35-243:~/django-notes-app$ vim .gitlab-ci.yml
```

Added a .gitlab-ci.yml file with the four stages with runner tag: Dev

Prerequisites

- docker & compose command should have installed

==========

Step-6 Run the Pipeline

```yaml
stages:
  - build
  - test
  - push_to_Docker-Hub
  - Deploy

build_job:
  stage: build
  script:
    - docker build -t node-app:latest .
  tags:
    - Dev

test_job:
  stage: test
  script:
    - echo " This will be my test stage"
  tags:
    - Dev

Push_job:
  stage: push_to_Docker-Hub
  script:
    - docker login -u $DockerHubUser -p $DockerHubPass
    - docker image tag node-app:latest $DockerHubUser/node-app:latest
    - docker push $DockerHubUser/node-app:latest
  tags:
    - Dev

deploy_job:
  stage: Deploy
  script:
    - docker compose up -d
  tags:
    - Dev
```

Pushed the new changes to gitlab & made a new commit.

```
ubuntu@ip-172-31-35-243:~/django-notes-app$ git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitlab-ci.yml

nothing added to commit but untracked files present (use "git add" to track)
ubuntu@ip-172-31-35-243:~/django-notes-app$ git add .gitlab-ci.yml
ubuntu@ip-172-31-35-243:~/django-notes-app$
ubuntu@ip-172-31-35-243:~/django-notes-app$
ubuntu@ip-172-31-35-243:~/django-notes-app$ git commit -m "Added a .gitlab-ci.yml file"
[main 59f0c7f] Added a .gitlab-ci.yml file
 Committer: Ubuntu <ubuntu@ip-172-31-35-243.eu-central-1.compute.internal>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly. Run the
following command and follow the instructions in your editor to edit
your configuration file:

    git config --global --edit

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author

 1 file changed, 27 insertions(+)
 create mode 100644 .gitlab-ci.yml
ubuntu@ip-172-31-35-243:~/django-notes-app$
```

Let's push the code to Origin , in case you have made the changes on gitlab directly , "git push will fail" you need to do "git pull origin –rebase"

```
ubuntu@ip-172-31-35-243:~/django-notes-app$ git push origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 510 bytes | 510.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
To gitlab.com:devops-batch-7/django-notes-app.git
   6413904..10d94e1  main -> main
ubuntu@ip-172-31-35-243:~/django-notes-app$
```

==========

Things to keep in Mind

- Your gitlab-runner User should be a Part of Docker Group

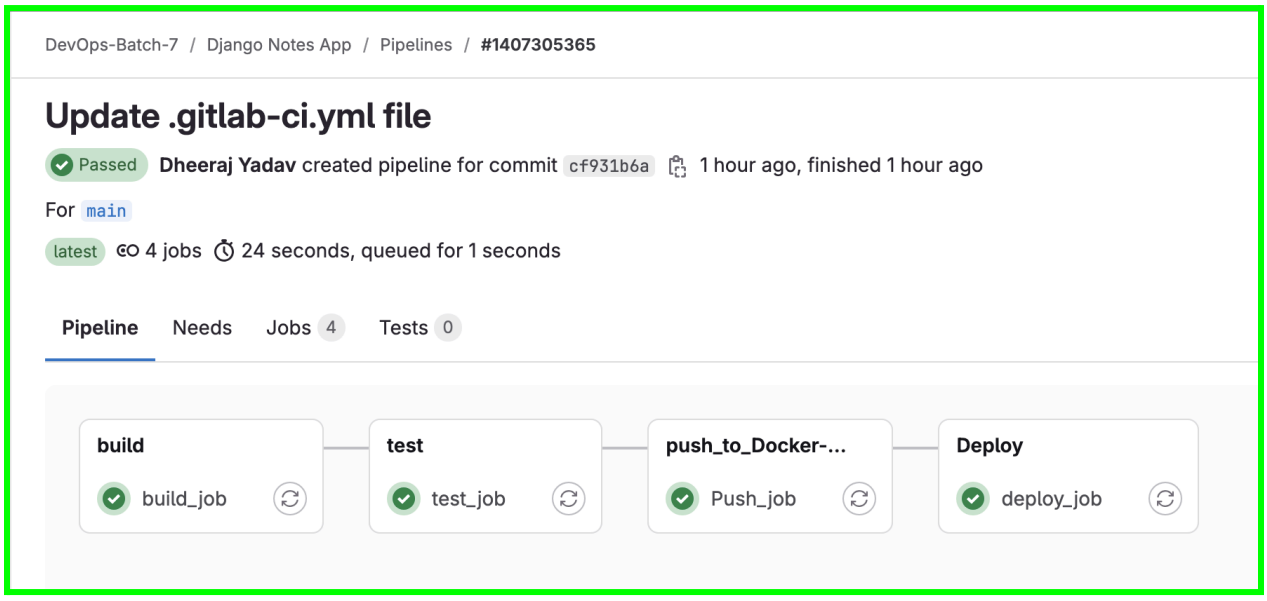
```
docker:x:113:ubuntu,gitlab-runner
```

- Runner should be running & enabled for your Project.

==========

Successful Pipeline:

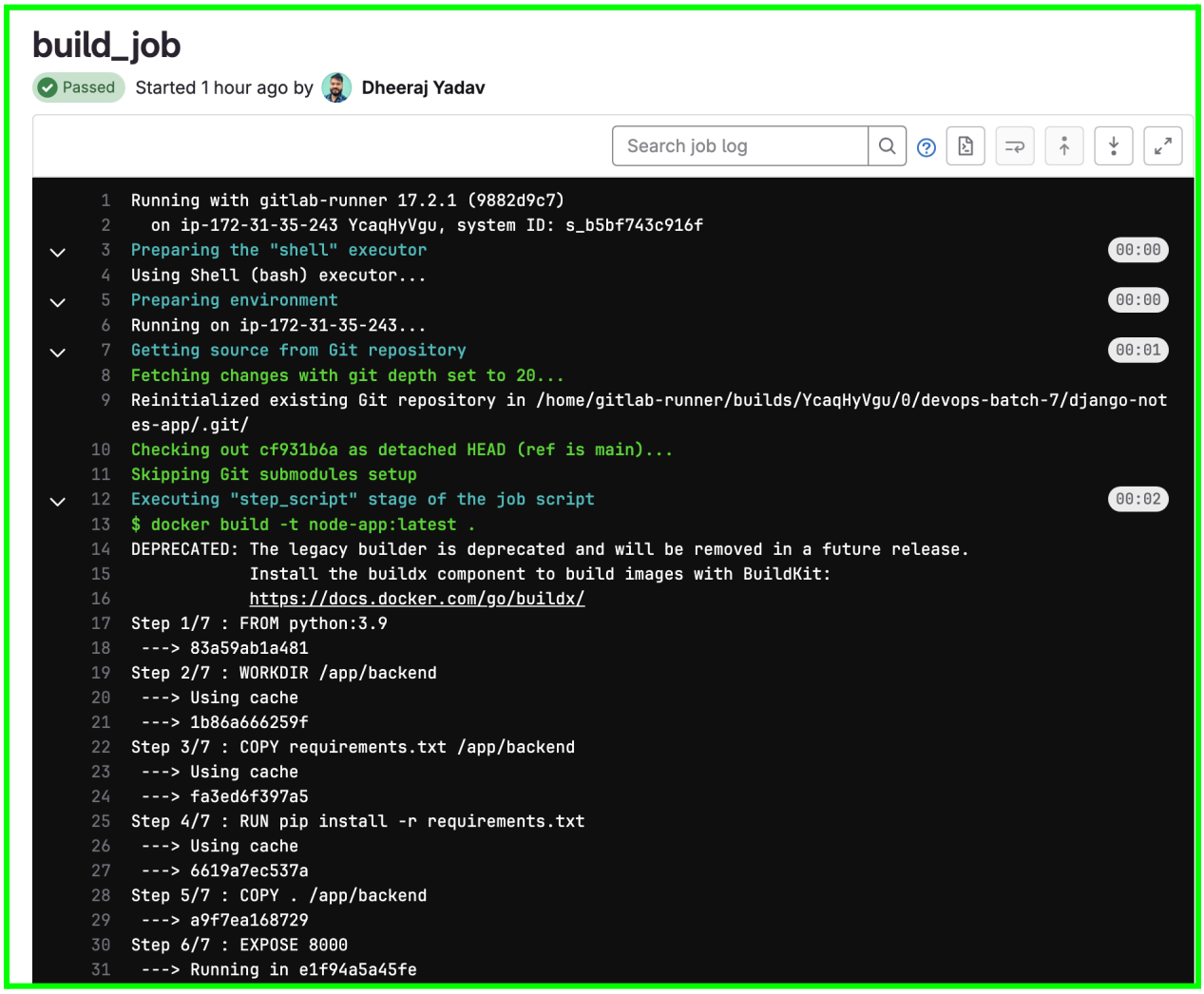
Pipeline has 4 stages & we have successfully uploaded the Docker image to DockerHub & deployed it.



==========

Build Job:

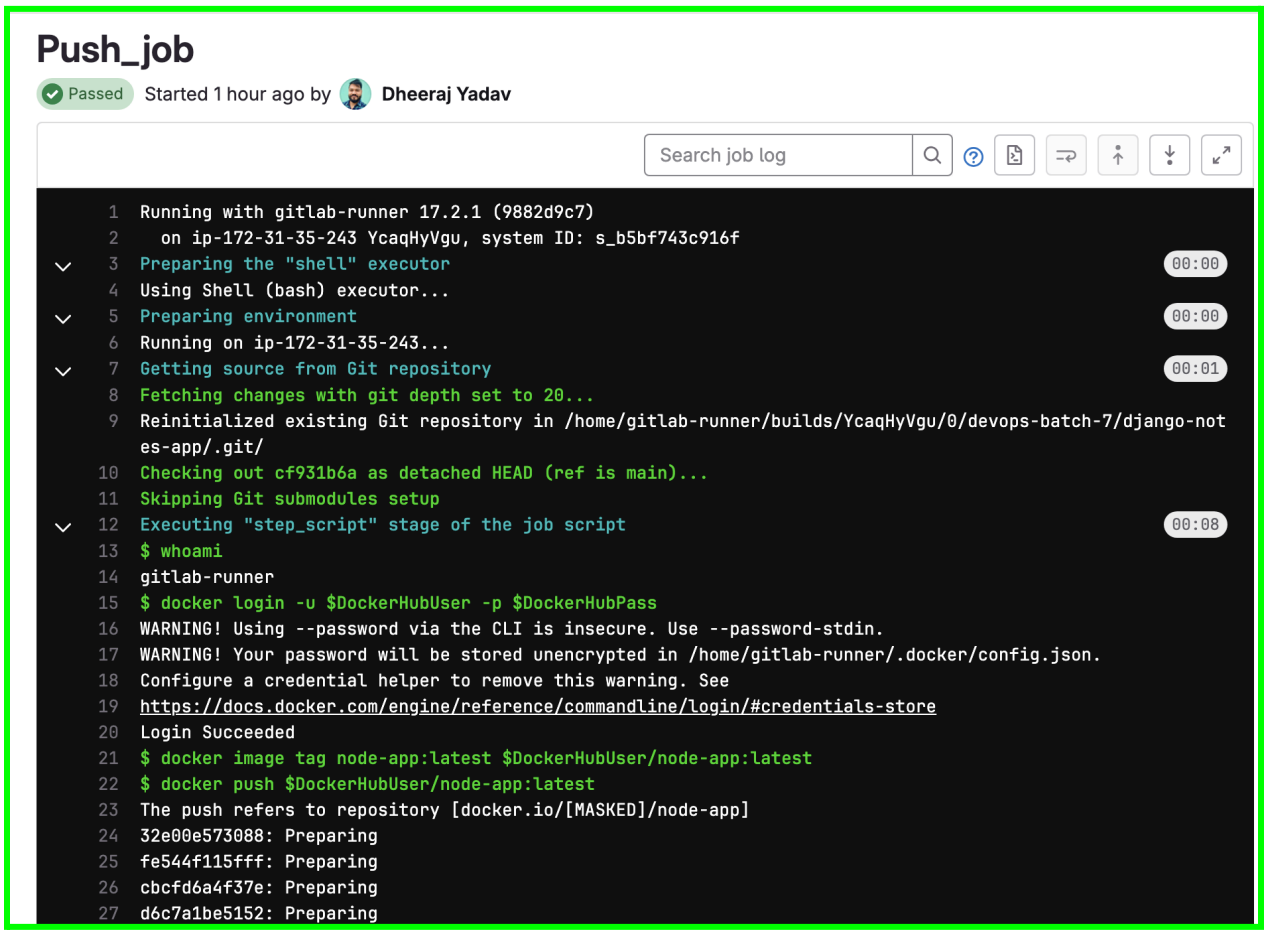
$ docker build -t node-app:latest . : Run this Docker command to build the docker image.

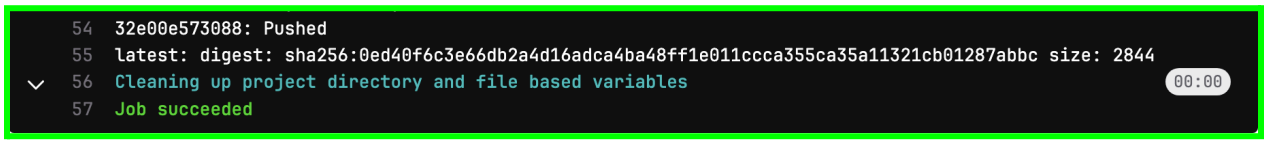


==========

Push Job:

Pushed the build image to DockerHub by using the User Defined Variables in the gitlab.

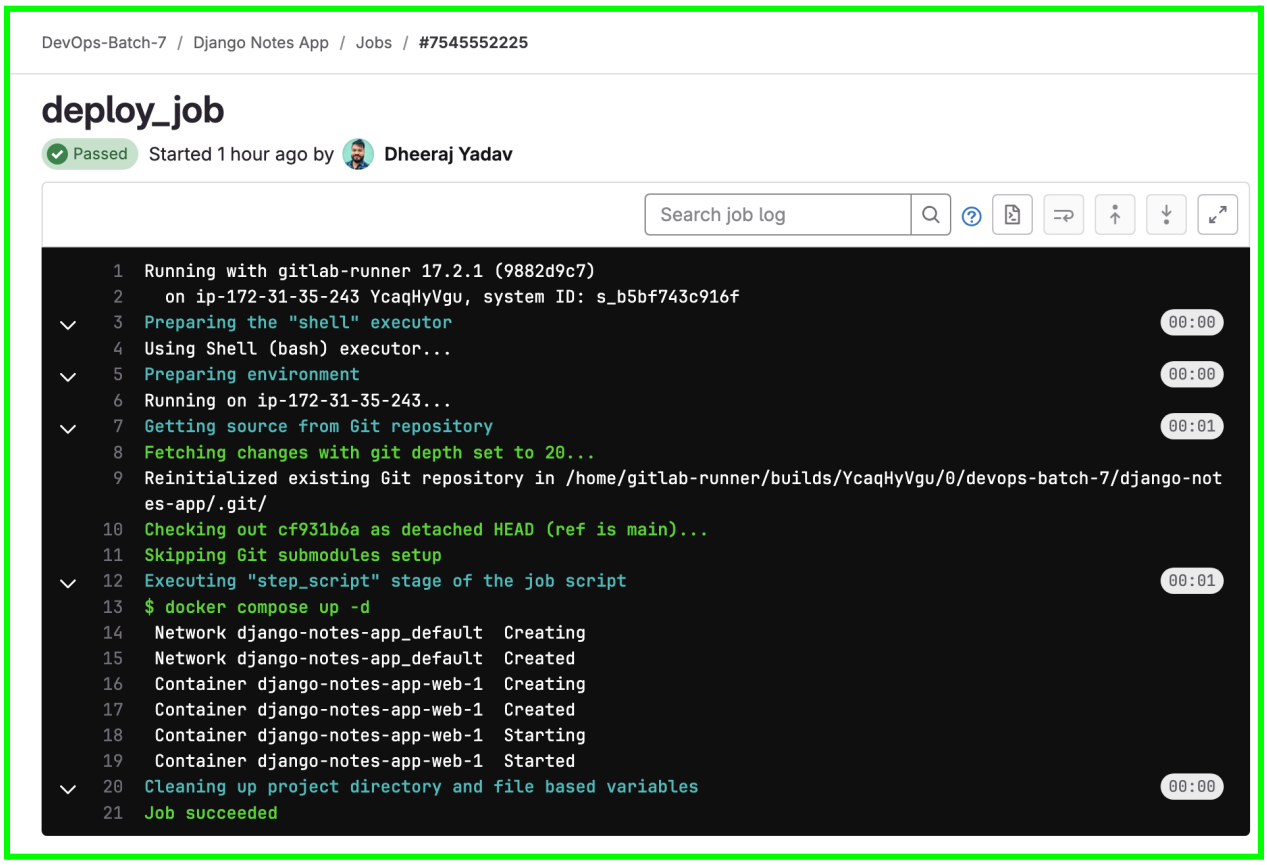


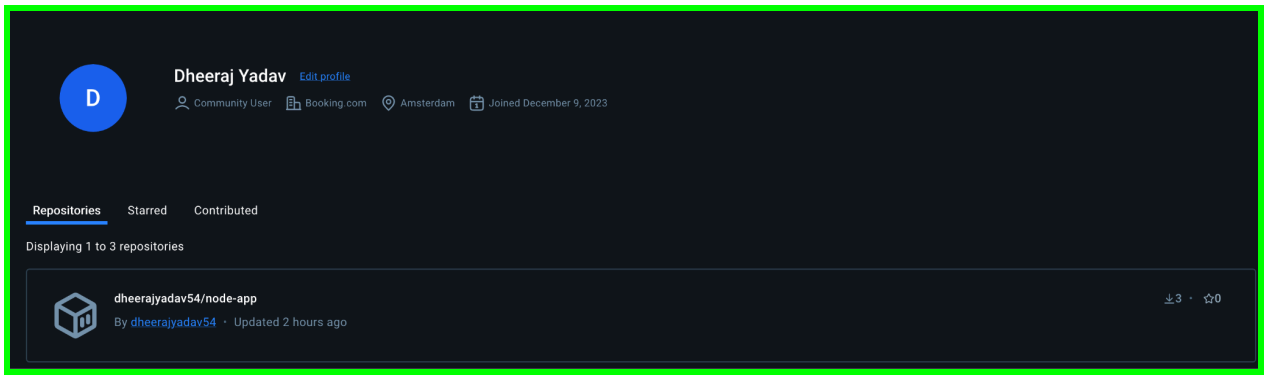


==========

Deploy Job:

Pushed the image to DockerHub.





==========

Step-7 Artifacts

We will be Storing the Artifacts for 1 week, Added the output into Artifacts Logs Folder.

Pipeline Config with Artifacts

Edit | Visualize | Validate NEW | Full configuration

CI/CD Catalog | Help

```
stages:
  - build
  - test
  - push_to_Docker-Hub
  - Deploy

build_job:
  stage: build
  script:
    - docker build -t node-app:latest .
    - mkdir logs
    - echo " We will store Docker logs into this file" >>app.logs
  artifacts:
    paths:
      - logs/
    when: on_success
    access: all
    expire_in: 1 week
  tags:
    - lab

test_job:
  stage: test
  script:
    - echo " This will be my test stage"
  tags:
    - lab
```

Pipeline was successful & we can Download the Artifacts file on the right side.