



Table of Contents:

Table of Contents:

[Step-1 : Gitlab Account](#)

[Step-2 : Create a Group & Project](#)

[Create a New Group](#)

[Create a New Project](#)

[Demo Pipeline](#)

[Parallel Jobs](#)

[Step-3: Variables Types](#)

[Predefined variables](#)

[Predefined CI/CD Variables Reference](#)

[User Defined Variables:](#)

[\(a\) Project Level Variable Define:](#)

[Generate the Personal Access Token on Docker Hub](#)

[\(b\) Group Level Variable Define:](#)

[Step-4 Gitlab Runners](#)

[Install the Gitlab Runner](#)

[Gitlab runner status](#)

[Gitlab runner Service status](#)

[Register the Gitlab Runner](#)

[Step-5: Project Import from](#)

[SSH Key-Gen on EC2 Machine](#)

[Add the Public Key to Gitlab Account](#)

[Step-6 Run the Pipeline](#)

[Things to keep in Mind](#)

[Successful Pipeline:](#)

[Build Job:](#)

[Push Job:](#)

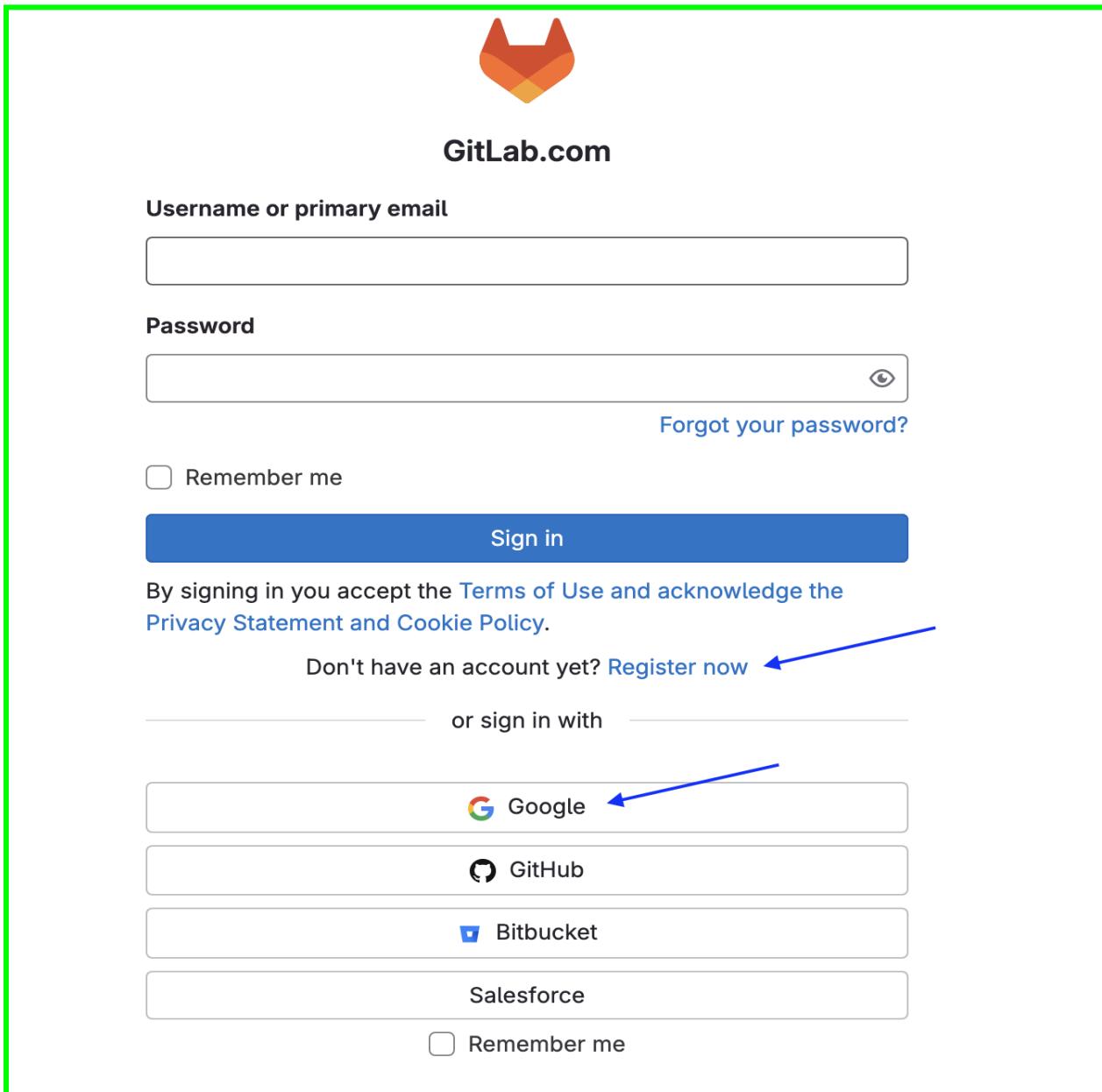
[Deploy Job:](#)

[Step-7 Artifacts](#)

[Pipeline Config with Artifacts](#)

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.



The image shows the GitLab.com login page. It features a large orange fox logo at the top center. Below it, the text "GitLab.com" is displayed. The page has two main input fields: "Username or primary email" and "Password". To the right of the password field is a blue "Sign in" button. Below these fields are links for "Forgot your password?" and "Remember me". At the bottom of the page, there is a section titled "By signing in you accept the [Terms of Use](#) and acknowledge the [Privacy Statement](#) and [Cookie Policy](#)". Below this, there is a link "Don't have an account yet? [Register now](#)". Further down, there is a "Remember me" checkbox and a "Sign in with" section containing social media logins for Google, GitHub, Bitbucket, and Salesforce. A blue arrow points from the "Register now" link to the "Sign in with" section, and another blue arrow points from the "Google" login option to the "Sign in with" section.



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

A screenshot of the GitLab interface showing the 'Groups' section. A blue arrow points to the 'Groups' link in the sidebar. Another blue arrow points to the 'New group' button in the top right corner of the main content area.

Click on “Create Group”

A screenshot of the 'Create new group' dialog. It shows two options: 'Create group' and 'Import group'. The 'Create group' option is selected, showing a description: 'Assemble related projects together and grant members access to several projects at once.' A blue arrow points to the 'Create group' button.



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

The screenshot shows the 'Create group' form on the GitLab interface. The 'Group name' field contains 'DevOps-Batch-7'. The 'Group URL' field has the value 'https://gitlab.com/devops-batch-7'. Under 'Visibility level', the 'Private' radio button is selected, indicated by a blue arrow. The 'Role' dropdown is set to 'Devops Engineer'. The 'Who will be using this group?' section shows 'Just me' selected. The 'What will you use this group for?' dropdown is set to 'I want to store my code'. At the bottom are 'Create group' and 'Cancel' buttons.

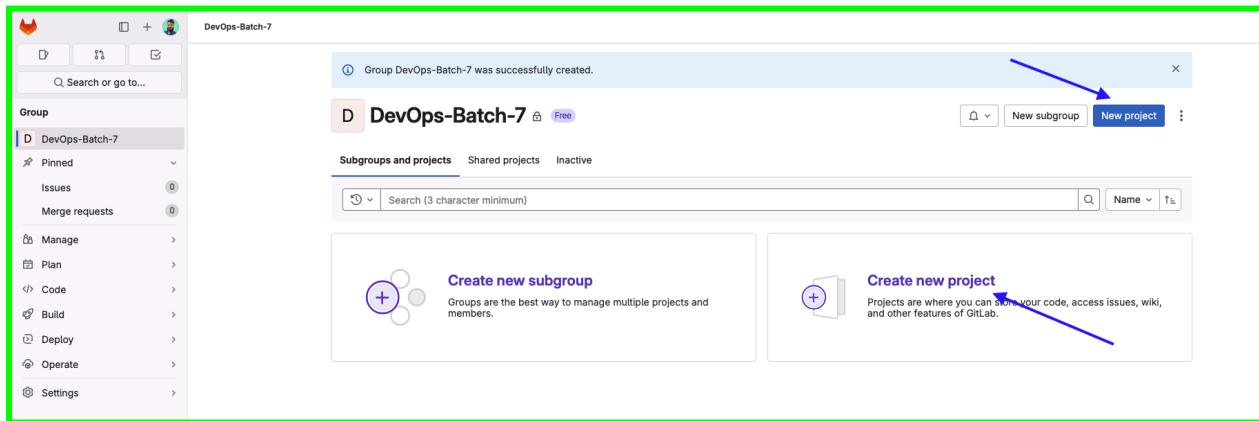
Once created, it will give you the below screen, where you can create a subgroup or New Project.

The screenshot shows the dashboard for the 'DevOps-Batch-7' group. A success message at the top states 'Group DevOps-Batch-7 was successfully created.' Below it, there's a summary of pinned issues and merge requests. The main area features two prominent buttons: 'Create new subgroup' (with a plus icon) and 'Create new project' (with a plus icon). Both buttons have descriptive text below them: 'Groups are the best way to manage multiple projects and members.' and 'Projects are where you can store your code, access issues, wiki, and other features of GitLab.'

Create a New Project

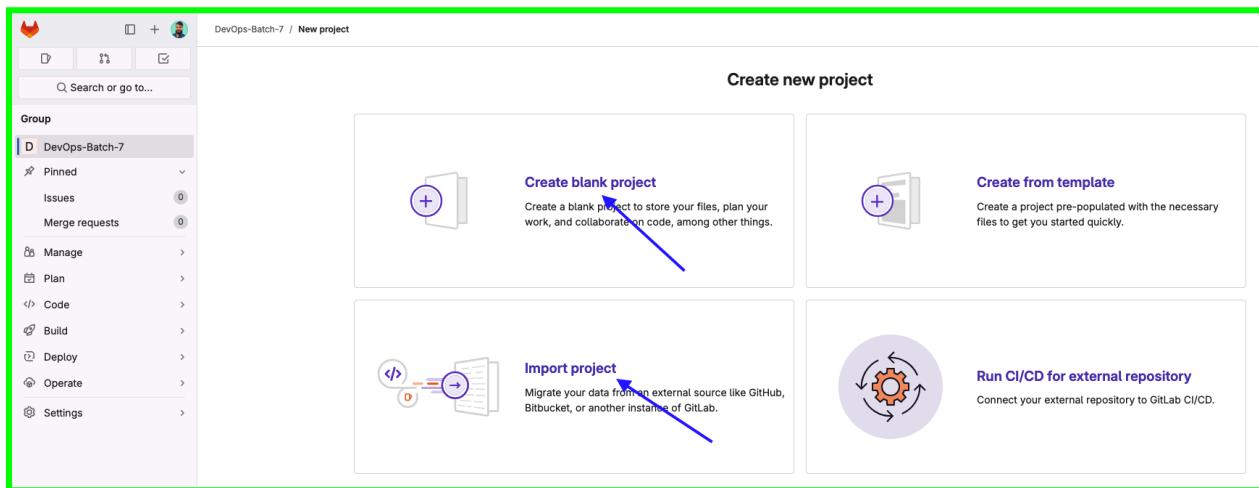
In Github we can call it a repository, in Gitlab it's Project.

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.





Give a Name of your Project & Create a Project.

The screenshot shows the 'Create blank project' page on GitLab. The URL in the address bar is 'DevOps-Batch-7 / New project / Create blank project'. The page title is 'Create blank project'. A sub-header says 'Create a blank project to store your files, plan your work, and collaborate on code, among other things.' There is a large input field for 'Project name' containing 'My First Project'. Below it, a note says 'Must start with a lowercase or uppercase letter, digit, emoji, or underscore. Can also contain dots, pluses, dashes, or spaces.' There are fields for 'Project URL' (set to 'https://gitlab.com/ devops-batch-7') and 'Project slug' (set to 'my-first-project'). A dropdown for 'Project deployment target (optional)' is set to 'Select the deployment target'. Under 'Visibility Level', the 'Private' option is selected, with a note: '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.' Other options are 'Internal' (access by logged-in users except external ones) and 'Public' (access without authentication). In the 'Project Configuration' section, the 'Initialize repository with a README' checkbox is checked, with a note: 'Allows you to immediately clone this project's repository. Skip this if you plan to push up an existing repository.' The 'Enable Static Application Security Testing (SAST)' checkbox is unchecked. At the bottom are 'Create project' and 'Cancel' buttons.

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:

DevOps-Batch-7 / My First Project / Repository

New file

main | .gitlab-ci.yml | Apply a template | No wrap

```
1 stages:
2   - build
3   - test
4   - Push
5   - deploy
6
7 build_job:
8   stage: build
9   script:
10    | - echo " this is a build JOB"
11
12 test_job:
13   stage: test
14   script:
15    | - echo " this is a test JOB"
16
17 Push_job:
18   stage: Push
19   script:
20    | - echo "Pushing the Docker image to Docker HUB"
21
22 Deploy_Job:
23   stage: deploy
24   script:
25    | - echo " Lets Deploy the image now"
26
27
28
```

Commit message
Added a .gitlab-ci.yml file

Target Branch
main

Commit changes | Cancel



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

```

stages:
  - build
  - test
  - Push
  - deploy

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

test_job:
  stage: test
  script:
    - echo " this is a test JOB"

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

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

```

Pipeline has been triggered & Completed successfully.



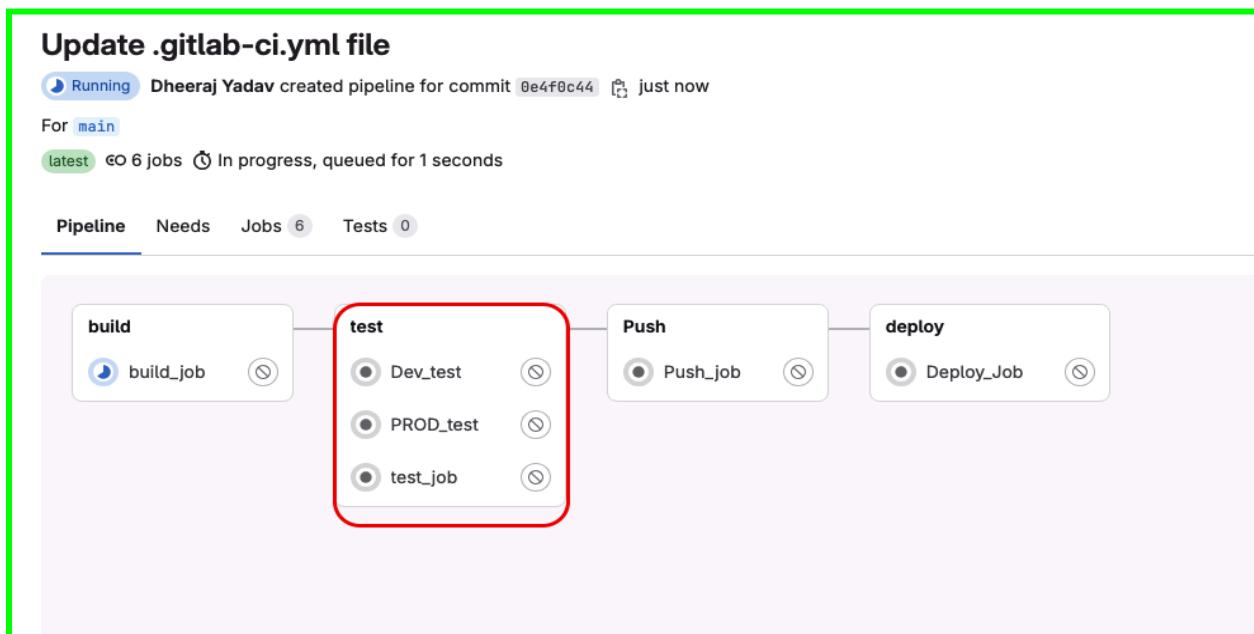
Parallel Jobs

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

.gitlab-ci.yml 616 B

Blame Edit Replace Delete

```
1 stages:
2   - build
3   - test
4   - Push
5   - deploy
6
7 build_job:
8   stage: build
9   script:
10    - echo " this is a build JOB for $CI_PROJECT_NAME"
11
12
13 test_job:
14   stage: test
15   script:
16    - echo " this is a test JOB run for pipeline $CI_PIPELINE_NAME"
17
18
19 Push_job:
20   stage: Push
21   script:
22    - echo "Pushing the Docker image to Docker HUB"
23
24
25 Deploy_Job:
26   stage: deploy
27   script:
28    - echo " Lets Deploy the image now"
29
30 Dev_test:
31   stage: test
32   script:
33    echo "doing more moe test"
34
35 PROD_test:
36   stage: test
37   script:
38    echo "doing more moe test"
39
```





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.

```
1 stages:
2   - build
3   - test
4   - Push
5   - deploy
6
7 build_job:
8   stage: build
9   script:
10    - echo " this is a build JOB for $CI_PROJECT_NAME"
11
12
13 test_job:
14   stage: test
15   script:
16    - echo " this is a test JOB run for pipeline $CI_PIPELINE_NAME"
17
18
19 Push_job:
20   stage: Push
21   script:
22    - echo "Pushing the Docker image to Docker HUB"
23
24
25 Deploy_Job:
26   stage: deploy
27   script:
28    - echo " Lets Deploy the image now"
```

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.

```
1 Running with gitlab-runner 17.8.0-pre.88.g761ae5dd (761ae5dd)
2   on green-5.saas-linux-small-amd64.runners-manager.gitlab.com/default xS6Vzpvo, system ID: s_6b1e4f06fcfd
3 
4   Preparing the "docker+machine" executor
5     Using Docker executor with image ruby:3.1 ...
6     Pulling docker image ruby:3.1 ...
7     Using docker image sha256:6599d27e0900fe9ae47371120a850fbe27afa617412ff2561fbb8c017e323bf5 for ruby:3.1 with digest ruby@sha256:73768ae423f5c2316895cf54a2735
8       8641301e95497c8d052865e40bdb0f6711 ...
9   Preparing environment
10    Running on runner-xS6Vzpvo-project-60641541-concurrent-0 via runner-xS6Vzpvo-s-l-s-amd64-1723103345-83ba2459...
11   Getting source from Git repository
12     Fetching changes with git depth set to 20...
13     Initialized empty Git repository in /builds/devops-batch-7/my-first-project/.git/
14     Created fresh repository.
15     Checking out 981855bb as detached HEAD (ref is main)...
16     Skipping Git submodules setup
17     $ git remote set-url origin "${CI_REPOSITORY_URL}"
18   Executing "step_script" stage of the job script
19     Using docker image sha256:6599d27e0900fe9ae47371120a850fbe27afa617412ff2561fbb8c017e323bf5 for ruby:3.1 with digest ruby@sha256:73768ae423f5c2316895cf54a2735
20       8641301e95497c8d052865e40bdb0f6711 ...
21     $ echo " this is a build JOB for $CI_PROJECT_NAME"
22     this is a build JOB for my-first-project
23   Cleaning up project directory and file based variables
24 
25 Job succeeded
```

Predefined CI/CD Variables Reference

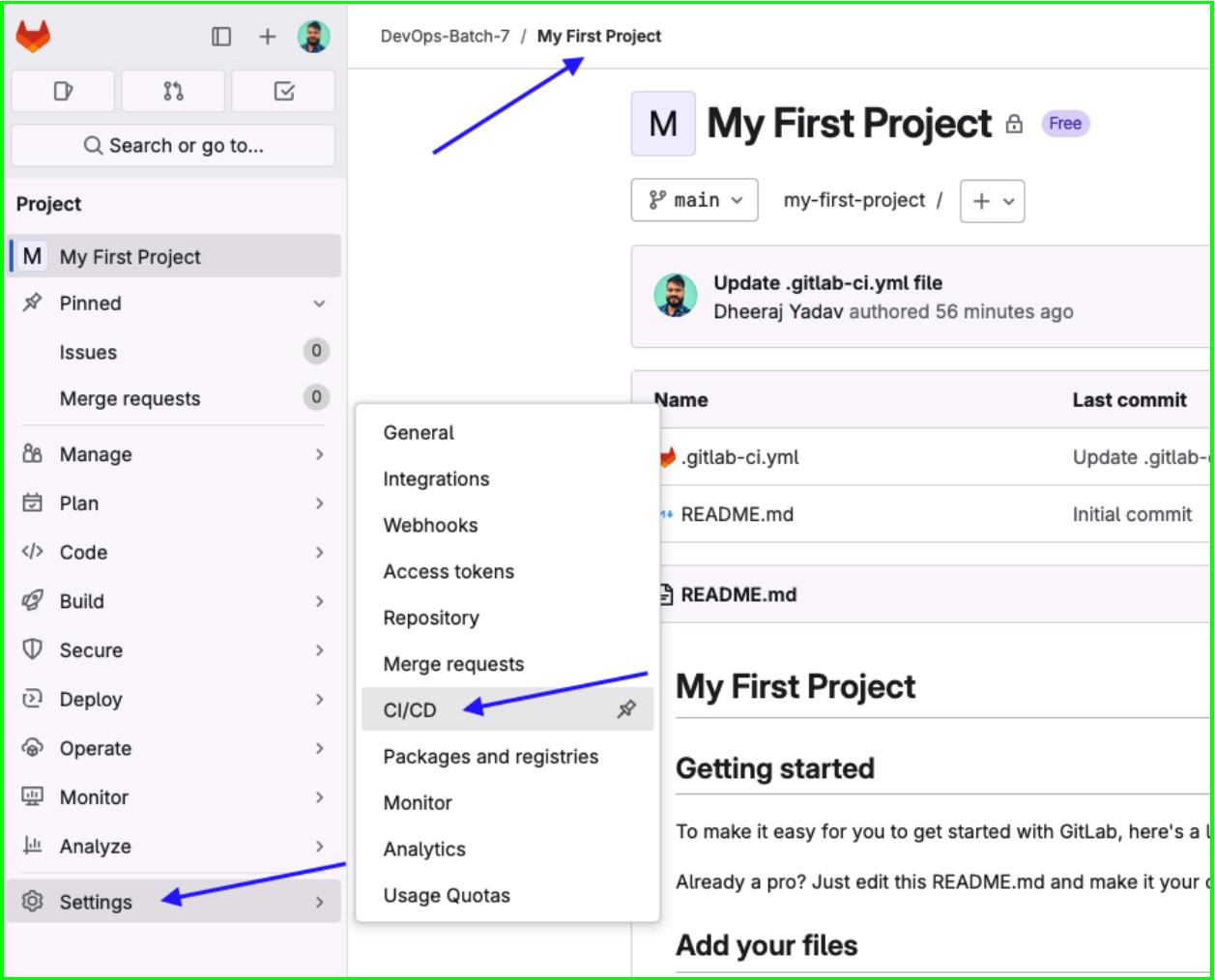
https://docs.gitlab.com/ee/ci/variables/predefined_variables.html

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



The screenshot shows the GitLab web interface for a project named "My First Project". A blue arrow points from the top-left towards the project title "My First Project". Another blue arrow points from the bottom-left towards the "Settings" menu item in the sidebar. A third blue arrow points from the right towards the "CI/CD" option in the dropdown menu under "Settings". The main content area displays a list of files in the repository, including ".gitlab-ci.yml", "README.md", and another "README.md". Below the repository list, there is a section titled "My First Project" with a "Getting started" message and a "Add your files" button.



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

Variables

Variables store information that you can use in job scripts. Each project can define a maximum of 8000 variables. [Learn more](#).

Variables can be accidentally exposed in a job log, or maliciously sent to a third party server. The masked variable feature can help reduce the risk of accidentally exposing variable values, but is not a guaranteed method to prevent malicious users from accessing variables. [How can I make my variables more secure?](#)

Variables can have several attributes. [Learn more](#).

- Protected: Only exposed to protected branches or protected tags.
- Masked: Hidden in job logs. Must match masking requirements.
- Expanded: Variables with \$ will be treated as the start of a reference to another variable.

CI/CD Variables </> 0			
Key ↑	Value	Environments	Actions
There are no variables yet.			

Group variables (inherited)

These variables are inherited from the parent group.

CI/CD Variables </> 1		
Key	Environments	Group
DockerHubUser	All (default)	DevOps-Batch-7
<small>Protected Masked Expanded</small>		

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.



Add variable

Type

Variable (default)

Environments [?](#)

All (default)

Visibility

Visible
Can be seen in job logs.

Masked
Masked in job logs but value can be revealed in CI/CD settings. Requires values to meet regular expressions requirements.

Flags [?](#)

Protect variable
Export variable to pipelines running on protected branches and tags only.

Expand variable reference
\$ will be treated as the start of a reference to another variable.

Description (optional)

The description of the variable's value or usage.

Key

DockerHubPass

You can use CI/CD variables with the same name in different places, but the variables might overwrite each other. [What is the order of precedence for variables?](#)

Value

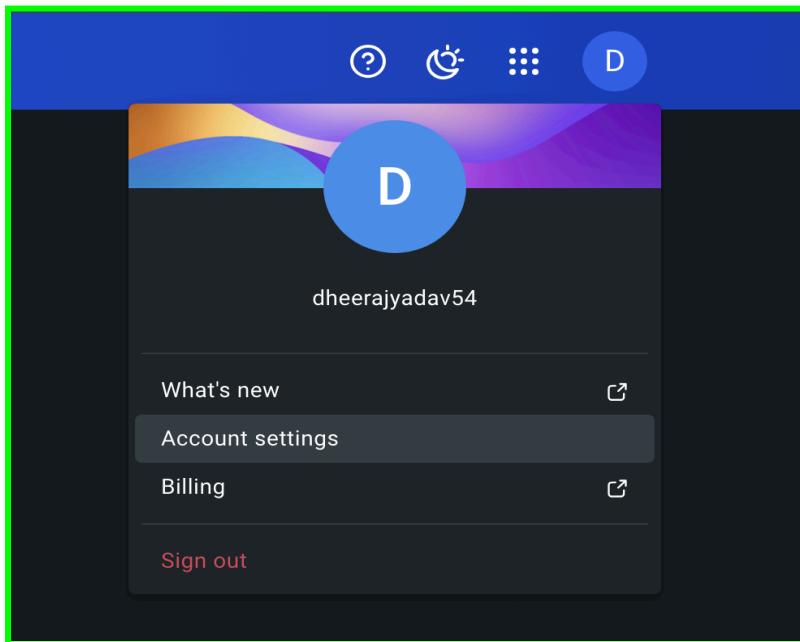
dckr_

Add variable **Cancel**



Generate the Personal Access Token on Docker Hub

- Docker Hub Account —> Account Setting

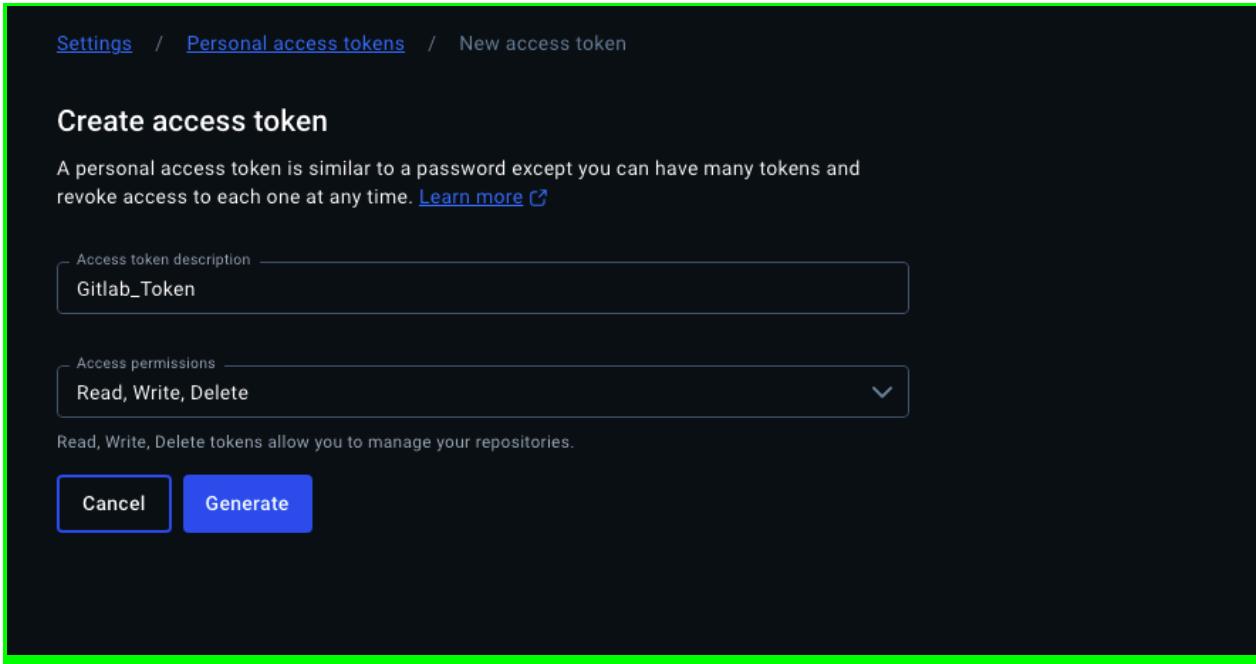


- Personal access tokens → Generate new token

A screenshot of the Docker Hub personal access tokens page. It shows a table with one token entry and a 'Generate new token' button. The token details are: Description: Batch-7, Scope: Read, Write, Delete, Status: Active, Source: Manual, Created: Jul 25, 2024 at 01:10:36, Last Used: Jul 25, 2024 at 01:10:43. The bottom of the page includes pagination controls: 'Rows per page: 10' and '1–1 of 1'.

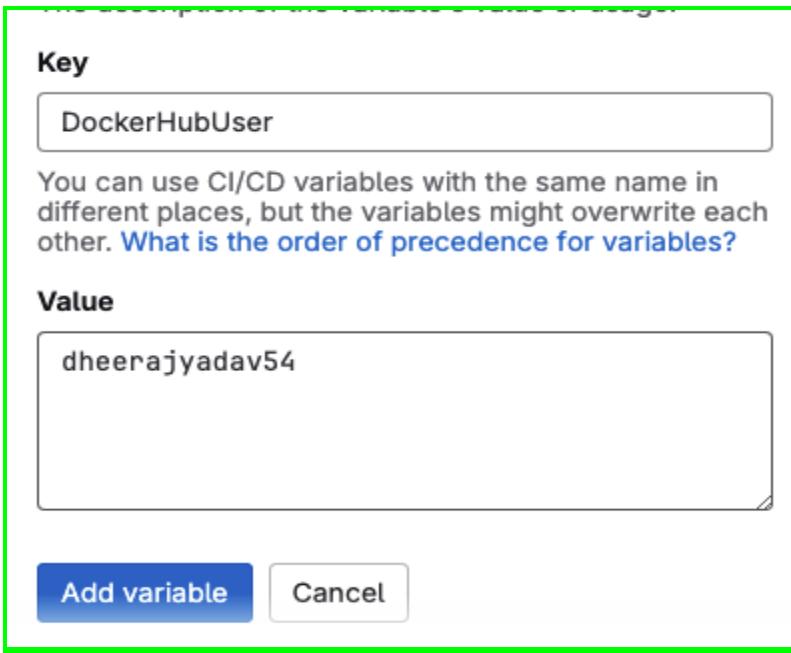


Click on Generate & Copy the token. Once you Copied the Token, go to Jenkins Credentials page & paste under Password Section.



The screenshot shows the 'Create access token' interface in GitLab. At the top, there's a breadcrumb navigation: Settings / Personal access tokens / New access token. The main title is 'Create access token'. A descriptive text states: 'A personal access token is similar to a password except you can have many tokens and revoke access to each one at any time.' Below this is a 'Learn more' link. There are two input fields: 'Access token description' containing 'Gitlab_Token' and 'Access permissions' set to 'Read, Write, Delete'. A note below says: 'Read, Write, Delete tokens allow you to manage your repositories.' At the bottom are 'Cancel' and 'Generate' buttons.

Added the DockerHub User Variable.



The screenshot shows the 'Add variable' dialog in Jenkins. It has two sections: 'Key' and 'Value'. The 'Key' section contains the value 'DockerHubUser'. The 'Value' section contains the value 'dheerajyadav54'. At the bottom are 'Add variable' and 'Cancel' buttons.



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.

CI/CD Variables </> 2		Reveal values	Add variable
Key ↑	Value	Environments	Actions
DockerHubPass	*****	All (default)	
DockerHubUser	*****	All (default)	

(b) Group Level Variable Define:

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

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

The screenshot shows the GitLab interface for managing CI/CD variables. On the left, there is a sidebar with various project management sections like Pinned, Issues, Merge requests, Plan, Build, Deploy, and Operate. Below these is a 'Settings' section. In the center, under 'CI/CD Settings', there is a 'Variables' section. This section contains a brief description: 'Variables store information that you can use in job scripts. Each group can define a maximum of 30000 variables.' It also includes sections for 'Runners' and 'Auto DevOps'. At the bottom of the 'Variables' section, there is a 'Learn more' link and an 'Expand' button. A blue arrow points from the 'Variables' section towards the 'Expand' button. Another blue arrow points from the 'CI/CD' section in the sidebar towards the 'Variables' section.



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

Variables

Variables store information that you can use in job scripts. Each group can define a maximum of 30000 variables. [Learn more](#).

Variables can be accidentally exposed in a job log, or maliciously sent to a third party server. The masked variable feature can help reduce the risk of accidentally exposing variable values, but is not a guaranteed method to prevent malicious users from accessing variables. [How can I make my variables more secure?](#)

Variables can have several attributes. [Learn more](#).

- **Protected:** Only exposed to protected branches or protected tags.
- **Masked:** Hidden in job logs. Must match masking requirements.
- **Expanded:** Variables with \$ will be treated as the start of a reference to another variable.

CI/CD Variables </> 1		Reveal values	Add variable
Key ↑	Value	Environments	Actions
DockerHubUser	*****	All (default)	
<small>Protected Masked Expanded</small>			



Step-4 Gitlab Runners

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

The screenshot shows the 'General pipelines' and 'Auto DevOps' sections collapsed. The 'Runners' section is expanded, indicated by a blue arrow pointing to its 'Expand' button. Below it, the 'Artifacts', 'Variables', 'Pipeline trigger tokens', and 'Deploy freezes' sections are also collapsed, each with its own 'Expand' button.

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

The screenshot shows the 'Runners' configuration page. The 'Runners' section is expanded, with a 'Collapse' button visible. It contains information about what runners are, how many to register, and how they pick up jobs. It also lists 'Project runners' assigned to the current project and 'Instance runners' available to all groups and projects. A toggle switch is shown for enabling instance runners for the project. Below this, a list of available instance runners is displayed, including their names and groups.

Runner Type	Description
Project runners	These runners are assigned to this project.
Instance runners	These runners are available to all groups and projects.

Available instance runners:

- #40242492 (yjKYbKa8Y) Dev
- #11573930 (KzYhZxBvi) gitlab-org
- 1-blue.shared-gitlab-org.runners-manager.gitlab.com gitlab-org
- #11573990 (NL4gfoBem) gitlab-org
- 2-blue.shared-gitlab-org.runners-manager.gitlab.com gitlab-org



These are the shared Gitlab Provided runners.

● #11574068 (Mf8beF5G3)

1-green.shared-gitlab-org.runners-manager.gitlab.com

gitlab-org

● #11574076 (8zCxmpPth)

2-green.shared-gitlab-org.runners-manager.gitlab.com

gitlab-org

● #11574084 (EuhiQzPR7)

3-green.shared-gitlab-org.runners-manager.gitlab.com

gitlab-org

● #11574096 (x5QiHUKwt)

4-green.shared-gitlab-org.runners-manager.gitlab.com

gitlab-org

Let's add the "New Project Runner"

Project runners

These runners are assigned to this project.

New project runner

⋮



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

DevOps-Batch-7 / Django Notes App / CI/CD Settings / New runner

New project runner

Create a project runner to generate a command that registers the runner with all its configurations.

Tags

Tags
Add tags to specify jobs that the runner can run. [Learn more.](#)

Dev

Separate multiple tags with a comma. For example, `macos, shared`.

Run untagged jobs
Use the runner for jobs without tags in addition to tagged jobs.

Configuration (optional)

Runner description
This is a dev runner to test the jobs

Paused
Stop the runner from accepting new jobs.

Protected
Use the runner on pipelines for protected branches only.

Lock to current projects
Use the runner for the currently assigned projects only. Only administrators can change the assigned projects.

Maximum job timeout
Maximum amount of time the runner can run before it terminates. If a project has a shorter job timeout period, the job timeout period of the instance runner is used instead.

Enter the job timeout in seconds. Must be a minimum of 600 seconds.

Create runner

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

Runner created.

Register "This is a dev runner to test the jobs" runner

Platform

Operating systems

Linux

macOS

Windows

Cloud

Google Cloud

Containers

Docker

Kubernetes



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

Click on “How do i install Gitlab runner”

The screenshot shows the GitLab interface for registering a runner. The main page has a green border around its content. On the left, there are sections for Google Cloud, Containers (Docker and Kubernetes), and a note about installing the runner before registering. Step 1 provides a command to register the runner. Step 2 asks to choose an executor. Step 3 (optional) shows a command to run the runner. A blue arrow points from the 'How do I install GitLab Runner?' link in Step 1 to a modal window titled 'Install GitLab Runner' on the right, which contains the installation script for amd64 architecture.

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   Current
         Dload  Upload Total   Spent    Left Speed
100  65.5M  100  65.5M    0     0  12.8M      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:~$
```

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:~$ 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 --service

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
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
Aug 09 08:55:34 ip-172-31-35-243 gitlab-runner[10527]: Configuration loaded
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_builds=0
Aug 09 08:55:34 ip-172-31-35-243 gitlab-runner[10527]: Initializing executor providers
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

Step 1

Copy and paste the following command into your command line to register the runner.

```
$ gitlab-runner register
--url https://gitlab.com
--token glrt-bCkcaF-vsNf5jVBBumMT
```

ⓘ The runner authentication token `glrt-bCkcaF-vsNf5jVBBumMT` displays here for a short time only. After you register the runner, this token is stored in the `config.toml` and cannot be accessed again from the UI.

Step 2

Choose an executor when prompted by the command line. Executors run builds in different environments. Not sure which one to select? ⓘ

Step 3 (optional)

Manually verify that the runner is available to pick up jobs.

```
$ gitlab-runner run
```

This may not be needed if you manage your runner as a [system or user service](#).

[View runners](#)



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

```
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]: runner=xRwPyKtDP
Verifying runner... is valid
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.

Step 1

Copy and paste the following command into your command line to register the runner.

```
$ gitlab-runner register
--url https://gitlab.com
--token glrt-xRwPyKtDP_JzDsjbBY8R
```

ⓘ The runner authentication token `glrt-xRwPyKtDP_JzDsjbBY8R` ⏺ displays here for a short time only. After you register the runner, this token is stored in the `config.toml` and cannot be accessed again from the UI.

Step 2

Choose an executor when prompted by the command line. Executors run builds in different environments. Not sure which one to select? ⏺

Step 3 (optional)

Manually verify that the runner is available to pick up jobs.

```
$ gitlab-runner run
```

This may not be needed if you manage your runner as a [system or user service](#) ⏺.

You've registered a new runner!

Your runner is online and ready to run jobs.

To view the runner, go to Project > CI/CD Settings > Runners.

[View runners](#)

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

Assigned project runners

 #40498249 (xRwPyKtDP)

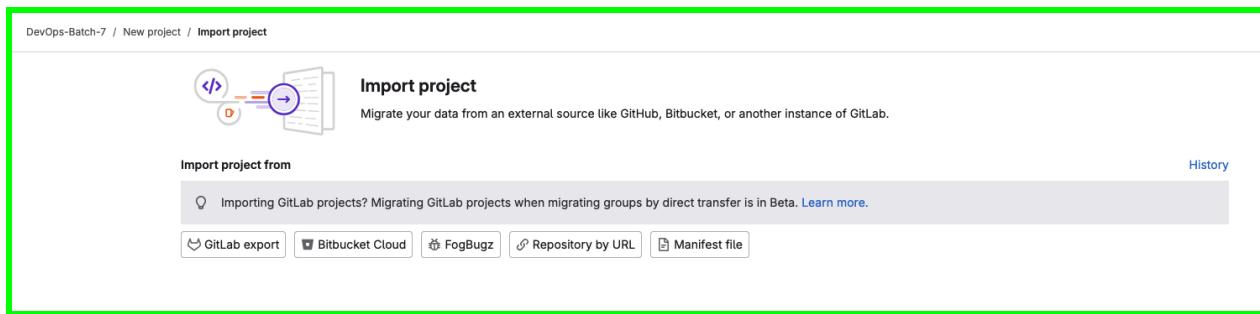
  Remove runner

This is a dev runner to test the jobs

Dev

Step-5: Project Import from

Here are the Options to import your projects into Gitlab.



The screenshot shows the 'Import project' interface in GitLab. At the top, there's a header with 'DevOps-Batch-7 / New project / Import project'. Below the header, there's a section titled 'Import project' with a sub-section 'Import project from'. Under 'Import project from', there are several options: 'GitLab export', 'Bitbucket Cloud', 'FogBugz', 'Repository by URL' (which is highlighted with a green border), and 'Manifest file'. A note at the bottom says 'Importing GitLab projects? Migrating GitLab projects when migrating groups by direct transfer is in Beta. Learn more.' and includes a 'History' link.

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**"



DevOps-Batch-7 / New project / Import project

The repository must be accessible over `http://`, `https://` or `git://`.
 When using the `http://` or `https://` protocols, please provide the exact URL to the repository. HTTP redirects will not be followed.
 If your HTTP repository is not publicly accessible, add your credentials.
 The import will time out after 3 hours. For repositories that take longer, use a clone/push combination.
 You can import a Subversion repository by using third-party tools. [Learn more](#).
 Once imported, repositories can be mirrored over SSH. Read more [here](#).

Git repository URL

 Mirror repository

Automatically update this project's branches and tags from the upstream repository. [How does pull mirroring work?](#)
 Mirroring will only be available if the feature is included in the plan of the selected group or user.

Username (optional)
Password (optional)

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

Project URL / **Project slug**

Project description (optional)
 Description format

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.
 Internal
 The project can be accessed by any logged in user except external users.
 Public
 The project can be accessed without any authentication.

Project has been imported successfully.

DevOps-Batch-7 / Django Notes App

The project was successfully imported. ←

/

Update Jenkinsfile
 Dheeraj Yadav authored 2 days ago Unverified c7d540de

Name	Last commit	Last update
api	first commit	1 year ago
mynotes	Update index.html	1 year ago
notesapp	deployment initial	1 year ago
staticfiles	deployment initial	1 year ago
.gitignore	first commit	1 year ago
Dockerfile	bump python version	1 year ago
Jenkinsfile	Update Jenkinsfile	2 days ago
README.md	Update README.md	1 week ago
db.sqlite3	final commit	1 year ago
docker-compose.yml	Added my DockerHub UserName	2 days ago
manage.py	first commit	1 year ago
procfile	deployment prep	1 year ago
requirements.txt	deployment initial	1 year ago
README.md		



=====

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:
SHA256:10iF0TSSLrvDG+0nWjrX8D2XhvXGRIZjo0BN7dA2ies ubuntu@ip-172-31-35-243
The key's randomart image is:
+--[ED25519 256]--+
|   o+o + . |
|   .*..+ * |
|   .+ oo = + |
|   .o... o + ol |
|   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] zu@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



The screenshot shows the 'User Settings / SSH Keys' page. On the left sidebar, under 'SSH Keys', there is a blue arrow pointing to the 'SSH Keys' link. In the main content area, there is a large purple circle with a lock icon, and the text 'There are no SSH keys with access to your account'. A blue arrow points to the 'Add new key' button at the top right of the list area.

Paste the Copied Public key here & hit add key.

The screenshot shows the 'Add an SSH key' form. The 'Key' field contains the copied public SSH key: `ssh-ed25519 AAAA[REDACTED]`. Below the key field, a note says: 'Begins with 'ssh-rsa', 'ecdsa-sha2-nistp256', 'ecdsa-sha2-nistp384', 'ecdsa-sha2-nistp521', 'ssh-ed25519', 'sk-ecdsa-sha2-nistp256@openssh.com', or 'sk-ssh-ed25519@openssh.com''. The 'Title' field is set to 'Gitlab-SSH-Key'. The 'Usage type' dropdown is set to 'Authentication & Signing'. The 'Expiration date' field is set to '2025-08-09'. At the bottom, there are 'Add key' and 'Cancel' buttons.



\$ 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:~$ 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:eUXGGm1YGsMAS7vkcx6J0Jd0GHPem5gQp4taiCfCLB8.
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:~$ cd 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$ 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

```
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.

The screenshot shows a successful pipeline named "Update .gitlab-ci.yml file" for commit #1407305365 on the main branch. The pipeline completed 4 jobs in 24 seconds, queued for 1 second, and finished 1 hour ago. The stages are: build (build_job), test (test_job), push_to_DockerHub (Push_job), and Deploy (deploy_job). All stages are marked as passed.

Passed 00:00:24 1 hour ago

Update .gitlab-ci.yml file
#1407305365 main cf931b6a latest

DevOps-Batch-7 / Django Notes App / Pipelines / #1407305365

Update .gitlab-ci.yml file

Passed Dheeraj Yadav created pipeline for commit cf931b6a 1 hour ago, finished 1 hour ago

For main

latest 4 jobs 24 seconds, queued for 1 seconds

Pipeline Needs Jobs 4 Tests 0

build build_job

test test_job

push_to_DockerHub Push_job

Deploy deploy_job



Build Job:

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

build_job

Passed Started 1 hour ago by Dheeraj Yadav

Search job log

```
1 Running with gitlab-runner 17.2.1 (9882d9c7)
2   on ip-172-31-35-243 YcaqHyVgu, system ID: s_b5bf743c916f
3 Preparing the "shell" executor
4 Using Shell (bash) executor...
5 Preparing environment
6 Running on ip-172-31-35-243...
7 Getting source from Git repository
8 Fetching changes with git depth set to 20...
9 Reinitialized existing Git repository in /home/gitlab-runner/builds/YcaqHyVgu/0/devops-batch-7/django-not
es-app/.git/
10 Checking out cf931b6a as detached HEAD (ref is main)...
11 Skipping Git submodules setup
12 Executing "step_script" stage of the job script
13 $ docker build -t node-app:latest .
14 DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
15     Install the buildx component to build images with BuildKit:
16     https://docs.docker.com/go/buildx/
17 Step 1/7 : FROM python:3.9
18   --> 83a59ab1a481
19 Step 2/7 : WORKDIR /app/backend
20   --> Using cache
21   --> 1b86a666259f
22 Step 3/7 : COPY requirements.txt /app/backend
23   --> Using cache
24   --> fa3ed6f397a5
25 Step 4/7 : RUN pip install -r requirements.txt
26   --> Using cache
27   --> 6619a7ec537a
28 Step 5/7 : COPY . /app/backend
29   --> a9f7ea168729
30 Step 6/7 : EXPOSE 8000
31   --> Running in e1f94a5a45fe
```

```
32 Removing intermediate container e1f94a5a45fe
33   --> 948c0d08047b
34 Step 7/7 : CMD python /app/backend/manage.py runserver 0.0.0.0:8000
35   --> Running in c2600d688ced
36 Removing intermediate container c2600d688ced
37   --> 4880563382ee
38 Successfully built 4880563382ee
39 Successfully tagged node-app:latest
40 Cleaning up project directory and file based variables
41 Job succeeded
```



Push Job:

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

Push_job

Passed Started 1 hour ago by Dheeraj Yadav

Search job log

```
1 Running with gitlab-runner 17.2.1 (9882d9c7)
2   on ip-172-31-35-243 YcaqHyVgu, system ID: s_b5bf743c916f
3 Preparing the "shell" executor
4 Using Shell (bash) executor...
5 Preparing environment
6 Running on ip-172-31-35-243...
7 Getting source from Git repository
8 Fetching changes with git depth set to 20...
9 Reinitialized existing Git repository in /home/gitlab-runner/builds/YcaqHyVgu/0/devops-batch-7/django-not
es-app/.git/
10 Checking out cf931b6a as detached HEAD (ref is main)...
11 Skipping Git submodules setup
12 Executing "step_script" stage of the job script
13 $ whoami
14 gitlab-runner
15 $ docker login -u $DockerHubUser -p $DockerHubPass
16 WARNING! Using --password via the CLI is insecure. Use --password-stdin.
17 WARNING! Your password will be stored unencrypted in /home/gitlab-runner/.docker/config.json.
18 Configure a credential helper to remove this warning. See
19 https://docs.docker.com/engine/reference/commandline/login/#credentials-store
20 Login Succeeded
21 $ docker image tag node-app:latest $DockerHubUser/node-app:latest
22 $ docker push $DockerHubUser/node-app:latest
23 The push refers to repository [docker.io/[MASKED]/node-app]
24 32e00e573088: Preparing
25 fe544f115fff: Preparing
26 cbcfda4f37e: Preparing
27 d6c7a1be5152: Preparing
```

```
54 32e00e573088: Pushed
55 latest: digest: sha256:0ed40f6c3e66db2a4d16adca4ba48ff1e011ccca355ca35a11321cb01287abbc size: 2844
56 Cleaning up project directory and file based variables
57 Job succeeded
```



=====

Deploy Job:

Pushed the image to DockerHub.

DevOps-Batch-7 / Django Notes App / Jobs / #7545552225

deploy_job

Passed Started 1 hour ago by Dheeraj Yadav

Search job log

```
1 Running with gitlab-runner 17.2.1 (9882d9c7)
2 on ip-172-31-35-243 YcaqHyVgu, system ID: s_b5bf743c916f
3 Preparing the "shell" executor
4 Using Shell (bash) executor...
5 Preparing environment
6 Running on ip-172-31-35-243...
7 Getting source from Git repository
8 Fetching changes with git depth set to 20...
9 Reinitialized existing Git repository in /home/gitlab-runner/builds/YcaqHyVgu/0/devops-batch-7/django-not
es-app/.git/
10 Checking out cf931b6a as detached HEAD (ref is main)...
11 Skipping Git submodules setup
12 Executing "step_script" stage of the job script
13 $ docker compose up -d
14 Network django-notes-app_default Creating
15 Network django-notes-app_default Created
16 Container django-notes-app-web-1 Creating
17 Container django-notes-app-web-1 Created
18 Container django-notes-app-web-1 Starting
19 Container django-notes-app-web-1 Started
20 Cleaning up project directory and file based variables
21 Job succeeded
```

Dheeraj Yadav [Edit profile](#)
 Community User Booking.com Amsterdam Joined December 9, 2023

[Repositories](#) [Starred](#) [Contributed](#)

Displaying 1 to 3 repositories

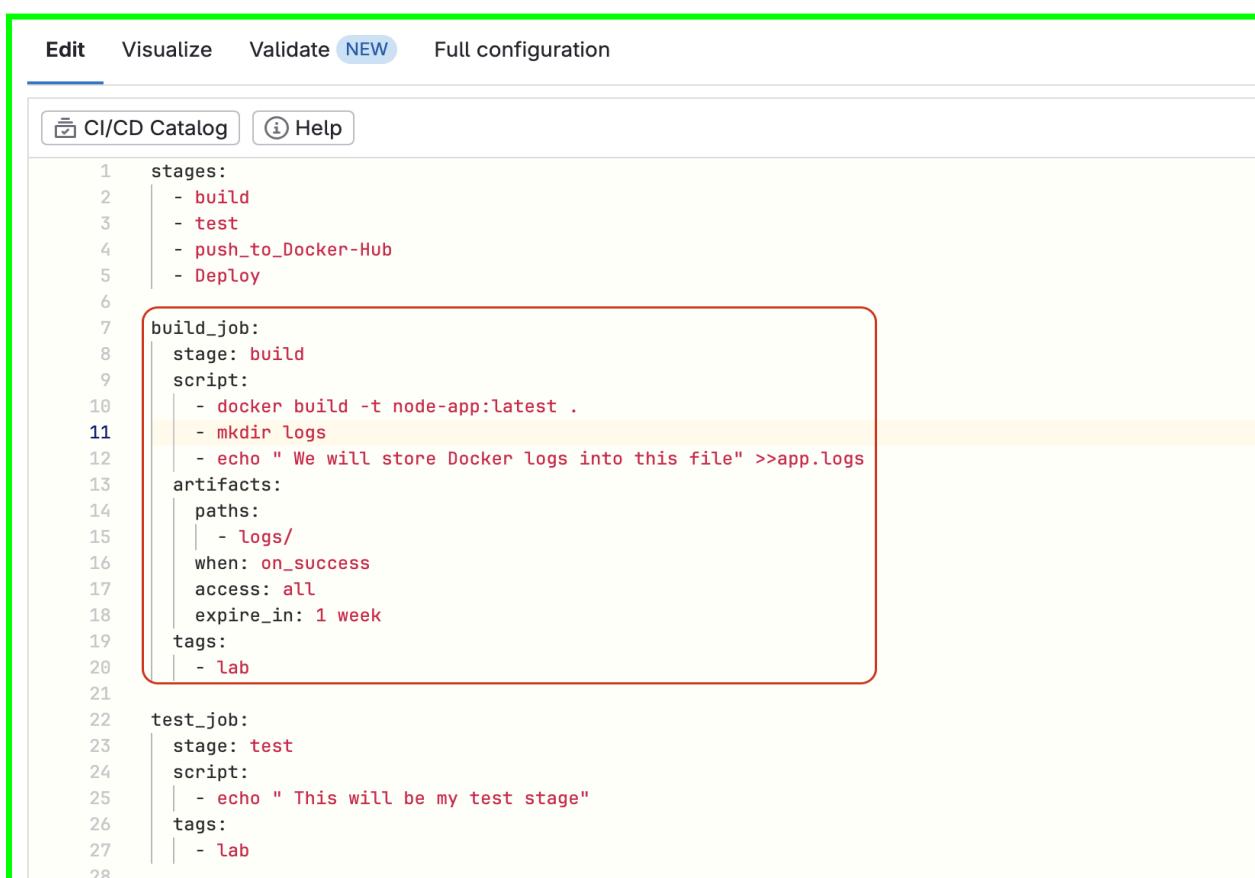
dheerajyadav54/node-app By dheerajyadav54 · Updated 2 hours ago	3 · 0
------------------------------------------------------------------------------------	--------



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

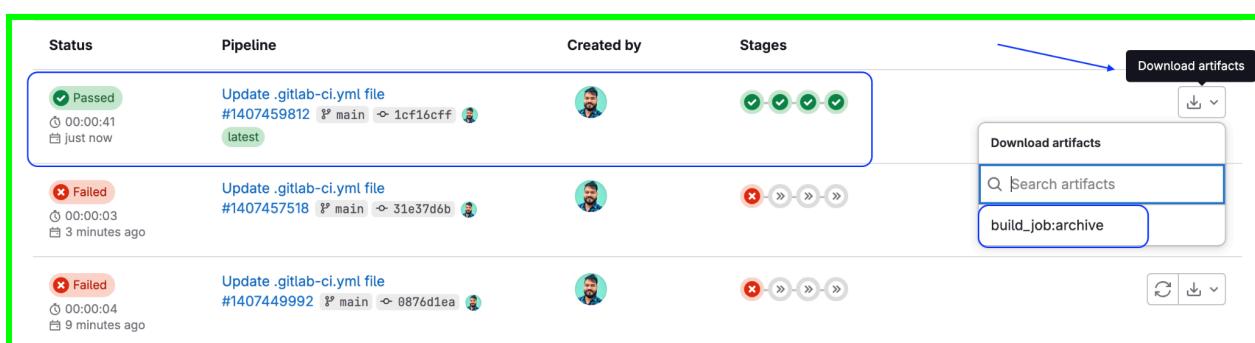
stages:
  - build
  - test
  - push_to_Docker_Hub
  - Deploy

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.



Status	Pipeline	Created by	Stages
✓ Passed ⌚ 00:00:41 ⌚ just now	Update .gitlab-ci.yml file #1407459812 ➔ main ➔ 1cf16cff 🚀 latest		✓ ✓ ✓ ✓ ✓
✗ Failed ⌚ 00:00:03 ⌚ 3 minutes ago	Update .gitlab-ci.yml file #1407457518 ➔ main ➔ 31e37d6b 🚧		✗ ➞ ➞ ➞ ➞
✗ Failed ⌚ 00:00:04 ⌚ 9 minutes ago	Update .gitlab-ci.yml file #1407449992 ➔ main ➔ 0876d1ea 🚧		✗ ➞ ➞ ➞ ➞

Download artifacts

build_job:archive