TOPIC: DEVOPS PROJECT FOR BEGINNERS: 1

PROJECT NAME: CI/CD using Gitlab

TOOLS: Linux, Shell Scripting, Gitlab, Docker & AWS

- Abhishek Kishor

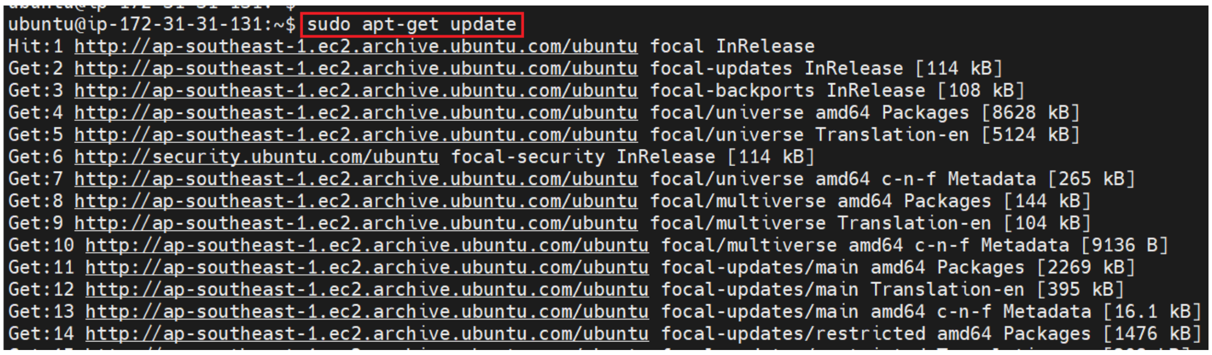
Index

Installing Gitlab Runner (Ubuntu)

- Note: Create an AWS Instance first (Linux). I have included the steps on how to create an ec2 instance in detail in this project. So, kindly go through it once.

- Update the system

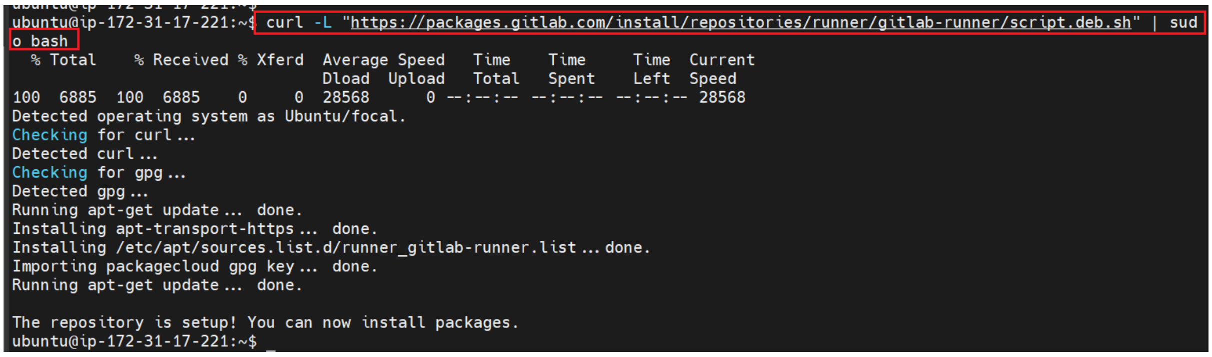
sudo apt-get update



```
ubuntu@ip-172-31-31-131:~$ sudo apt-get update
Hit:1 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu focal InRelease
Get:2 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu focal-updates InRelease [114 kB]
Get:3 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu focal-backports InRelease [108 kB]
Get:4 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu focal/universe amd64 Packages [8628 kB]
Get:5 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu focal/universe Translation-en [5124 kB]
Get:6 http://security.ubuntu.com/ubuntu focal-security InRelease [114 kB]
Get:7 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu focal/universe amd64 c-n-f Metadata [265 kB]
Get:8 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu focal/multiverse amd64 Packages [144 kB]
Get:9 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu focal/multiverse Translation-en [104 kB]
Get:10 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu focal/multiverse amd64 c-n-f Metadata [9136 B]
Get:11 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu focal-updates/main amd64 Packages [2269 kB]
Get:12 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu focal-updates/main Translation-en [395 kB]
Get:13 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu focal-updates/main amd64 c-n-f Metadata [16.1 kB]
Get:14 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu focal-updates/restricted amd64 Packages [1476 kB]
```

- Add the official GitLab repository:

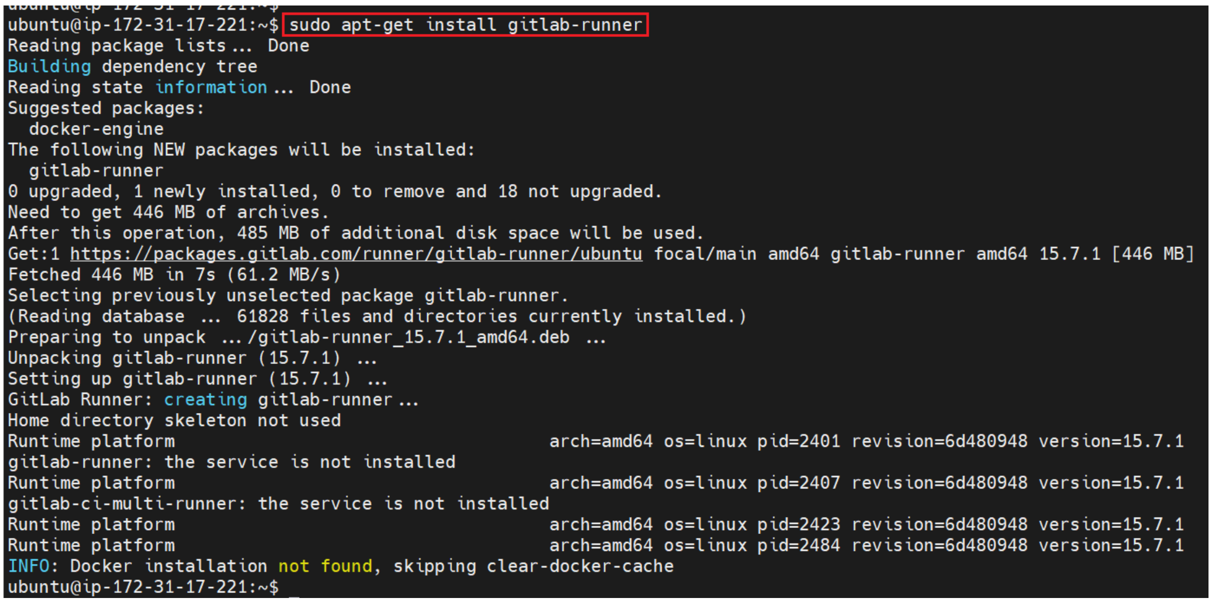
curl -L "https://packages.gitlab.com/install/repositories/runner/gitlab-runner/script.deb.sh" | sudo bash



```
ubuntu@ip-172-31-17-221:~$ curl -L "https://packages.gitlab.com/install/repositories/runner/gitlab-runner/script.deb.sh" | sud
o bash
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  6885  100  6885    0     0  28568      0 --:--:-- --:--:-- --:--:-- 28568
Detected operating system as Ubuntu/focal.
Checking for curl...
Detected curl...
Checking for gpg...
Detected gpg...
Running apt-get update... done.
Installing apt-transport-https... done.
Installing /etc/apt/sources.list.d/runner_gitlab-runner.list...done.
Importing packagecloud gpg key... done.
Running apt-get update... done.

The repository is setup! You can now install packages.
ubuntu@ip-172-31-17-221:~$ _
```

- Install the latest version of GitLab Runner

sudo apt-get install gitlab-runner



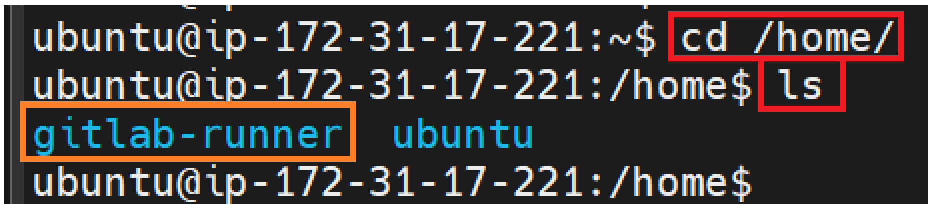
```
ubuntu@ip-172-31-17-221:~$ sudo apt-get install gitlab-runner
Reading package lists... Done
Building dependency tree
Reading state information... Done
Suggested packages:
  docker-engine
The following NEW packages will be installed:
  gitlab-runner
0 upgraded, 1 newly installed, 0 to remove and 18 not upgraded.
Need to get 446 MB of archives.
After this operation, 485 MB of additional disk space will be used.
Get:1 https://packages.gitlab.com/runner/gitlab-runner/ubuntu focal/main amd64 gitlab-runner amd64 15.7.1 [446 MB]
Fetched 446 MB in 7s (61.2 MB/s)
Selecting previously unselected package gitlab-runner.
(Reading database ... 61828 files and directories currently installed.)
Preparing to unpack .../gitlab-runner_15.7.1_amd64.deb ...
Unpacking gitlab-runner (15.7.1) ...
Setting up gitlab-runner (15.7.1) ...
GitLab Runner: creating gitlab-runner...
Home directory skeleton not used
Runtime platform                                    arch=amd64 os=linux pid=2401 revision=6d480948 version=15.7.1
gitlab-runner: the service is not installed
Runtime platform                                    arch=amd64 os=linux pid=2407 revision=6d480948 version=15.7.1
gitlab-ci-multi-runner: the service is not installed
Runtime platform                                    arch=amd64 os=linux pid=2423 revision=6d480948 version=15.7.1
Runtime platform                                    arch=amd64 os=linux pid=2484 revision=6d480948 version=15.7.1
INFO: Docker installation not found, skipping clear-docker-cache
ubuntu@ip-172-31-17-221:~$ _
```

Grant Sudo Permission To Gitlab Runner User

- A user will be created named “**gitlab-runner**”.

cd /home

ls



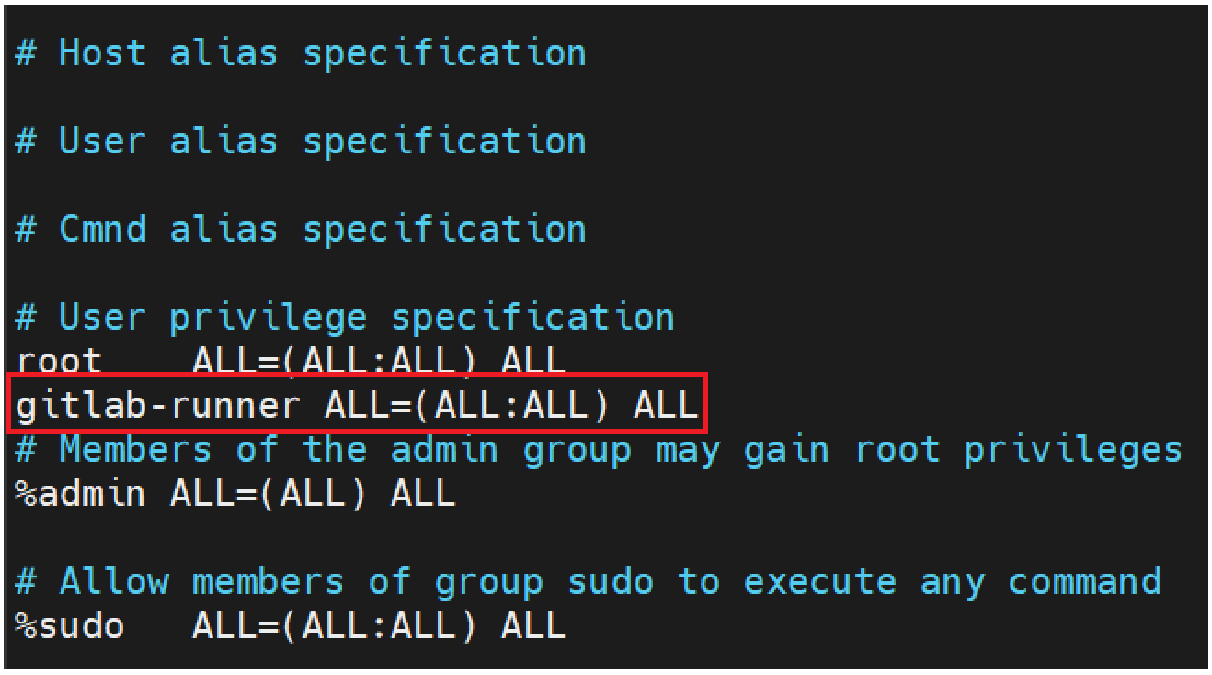
```
ubuntu@ip-172-31-17-221:~$ cd /home/
ubuntu@ip-172-31-17-221:/home$ ls
gitlab-runner  ubuntu
ubuntu@ip-172-31-17-221:/home$
```

- Grant sudo permission to gitlab-runner user
  - Open the file “**sudoers**”

sudo visudo

  - Add the user in that file in order to grant sudo permission.

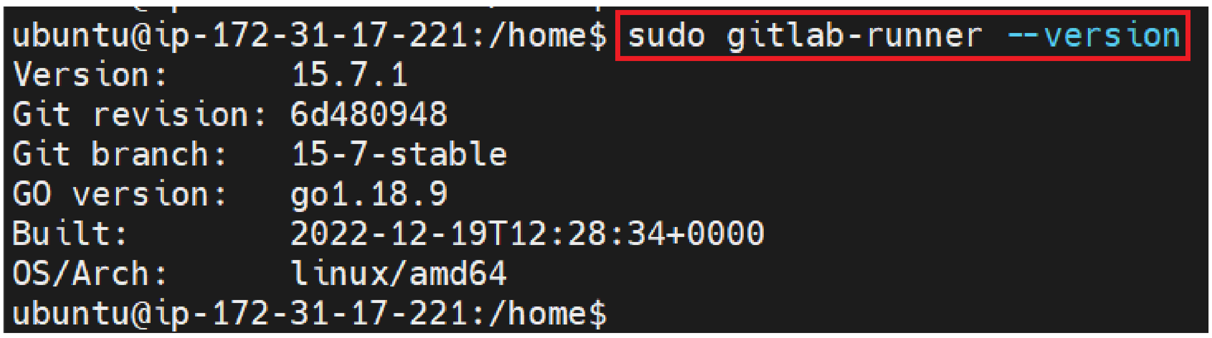
gitlab-runner ALL=(ALL:ALL) ALL



```
# Host alias specification

# User alias specification

# Cmnd alias specification

# User privilege specification
root    ALL=(ALL:ALL) ALL
gitlab-runner ALL=(ALL:ALL) ALL
# Members of the admin group may gain root privileges
%admin ALL=(ALL) ALL

# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL
```

Checking Gitlab-Runner Version

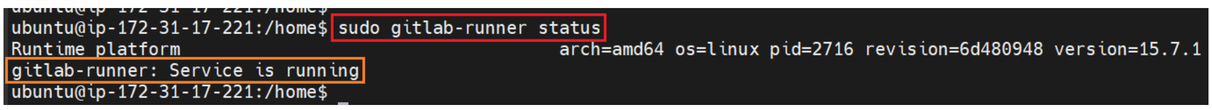
sudo gitlab-runner --version



```
ubuntu@ip-172-31-17-221:/home$ sudo gitlab-runner --version
Version:       15.7.1
Git revision:  6d480948
Git branch:    15-7-stable
GO version:    go1.18.9
Built:         2022-12-19T12:28:34+0000
OS/Arch:       linux/amd64
ubuntu@ip-172-31-17-221:/home$
```

- Checking Status of Gitlab-Runner

sudo gitlab-runner status

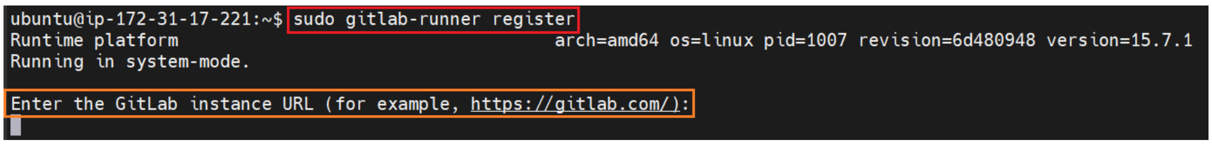


```
ubuntu@ip-172-31-17-221:/home$ sudo gitlab-runner status
Runtime platform                                    arch=amd64 os=linux pid=2716 revision=6d480948 version=15.7.1
gitlab-runner: Service is running
ubuntu@ip-172-31-17-221:/home$
```

Register A Shell Gitlab-Runner

sudo gitlab-runner register

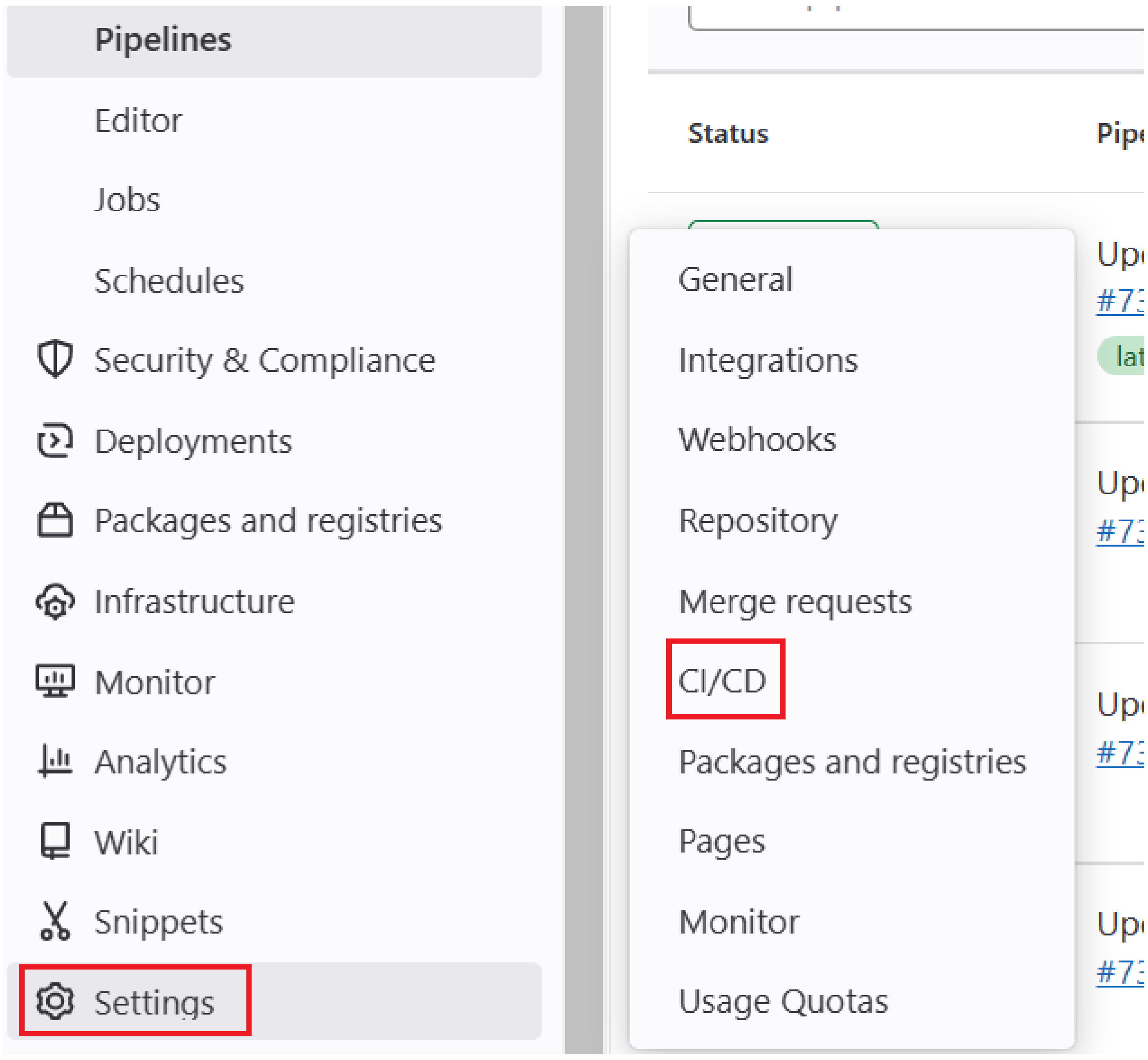
- Enter the GitLab instance URL



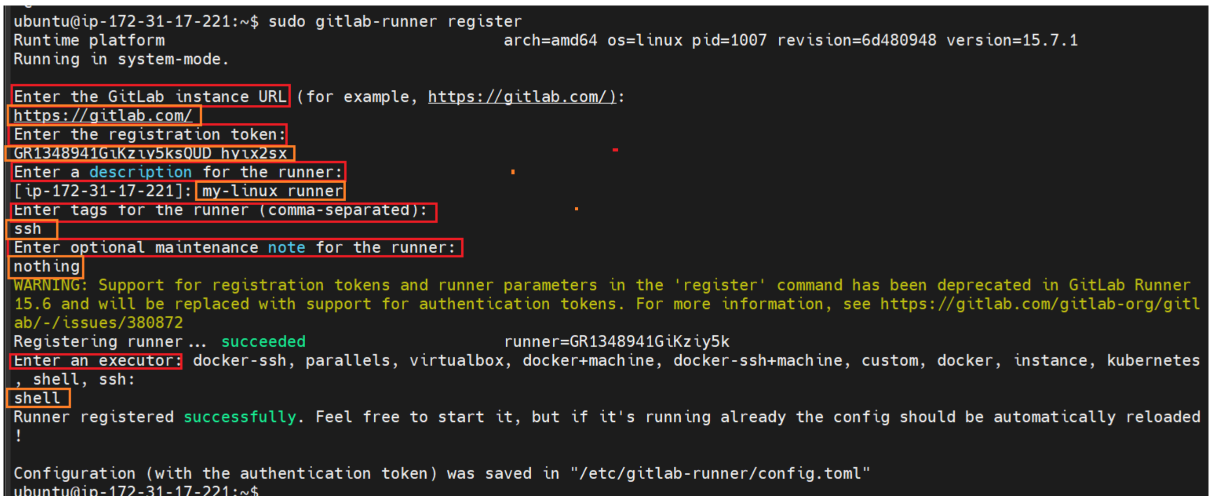
```
ubuntu@ip-172-31-17-221:~$ sudo gitlab-runner register
Runtime platform                                    arch=amd64 os=linux pid=1007 revision=6d480948 version=15.7.1
Running in system-mode.

Enter the GitLab instance URL (for example, https://gitlab.com/):
```

- Open Gitlab

Project > Setting > CI/CD

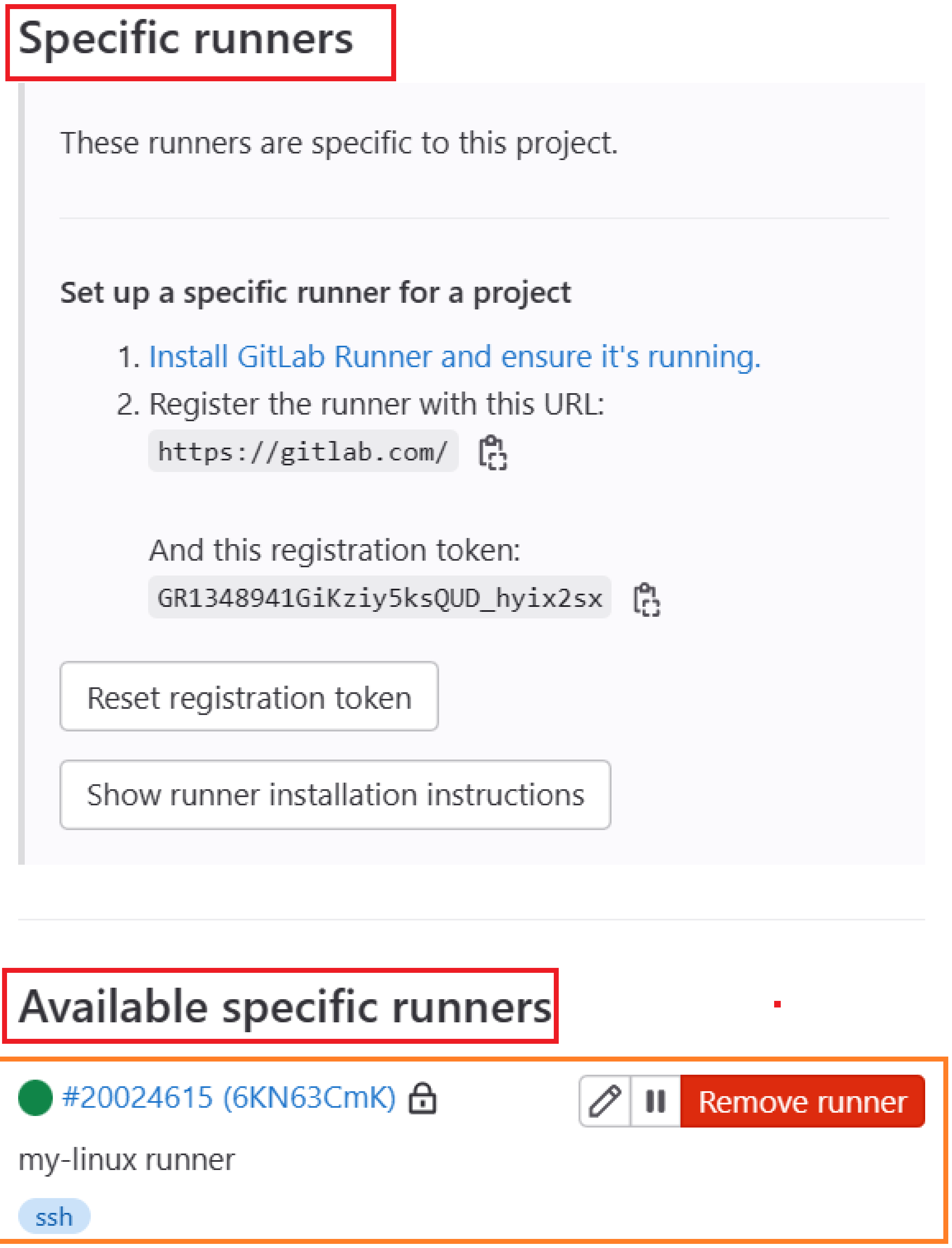


CI/CD > Runners > Expand > **Specific runners** > Copy & Paste the link > https://gitlab.com/ > Copy the Registration token > Paste it into the section > Enter the registration token > **GR1348941GiKziy5ksQUD_hyix2sx** > Enter a description > **my-linux runner** > Enter tags for the runner > **ssh** > Any Note > **Nothing** > Enter an executor > **shell**



```
ubuntu@ip-172-31-17-221:~$ sudo gitlab-runner register
Runtime platform                                    arch=amd64 os=linux pid=1007 revision=6d480948 version=15.7.1
Running in system-mode.

Enter the GitLab instance URL (for example, https://gitlab.com/):
https://gitlab.com/
Enter the registration token:
GR1348941GiKziy5ksQUD hyix2sx
Enter a description for the runner:
[ip-172-31-17-221]: my-linux runner
Enter tags for the runner (comma-separated):
ssh
Enter optional maintenance note for the runner:
nothing
WARNING: Support for registration tokens and runner parameters in the 'register' command has been deprecated in GitLab Runner
15.6 and will be replaced with support for authentication tokens. For more information, see https://gitlab.com/gitlab-org/gitl
ab/-/issues/380872
Registering runner... succeeded                     runner=GR1348941GiKziy5k
Enter an executor: docker-ssh, parallels, virtualbox, docker+machine, docker-ssh+machine, custom, docker, instance, kubernetes
, shell, ssh:
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 "/etc/gitlab-runner/config.toml"
ubuntu@ip-172-31-17-221:~$
```

- Now, reload the runner’s section of the project in gitlab and then we’ll see the gitlab-runner has been added in the sub-section “**Available specific runners**”.


Specific runners

These runners are specific to this project.

Set up a specific runner for a project

1. Install GitLab Runner and ensure it's running.
2. Register the runner with this URL:
   `https://gitlab.com/`

   And this registration token:
   `GR1348941GiKziy5ksQUD_hyix2sx`

Available specific runners

● #20024615 (6KN63CmK)

my-linux runner

ssh

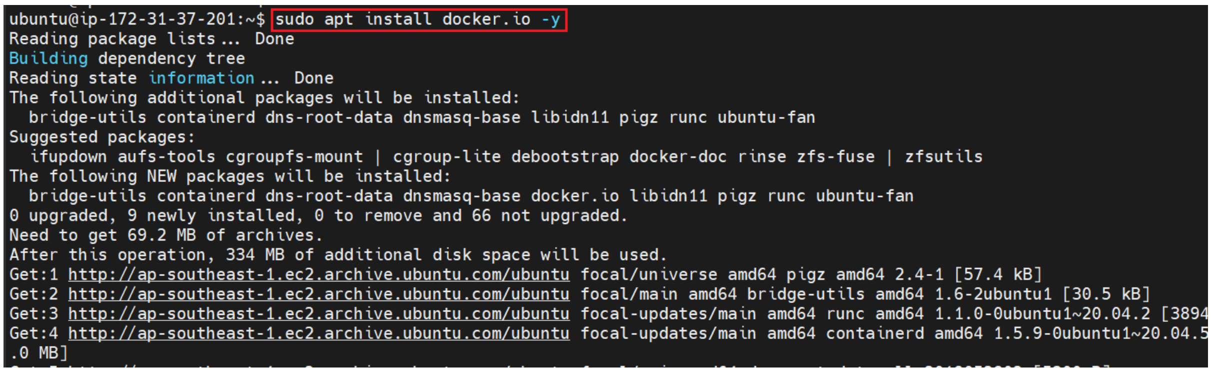
- You can change the tag afterwards.

Register New Runner (Docker) On Same Server

- In order to register a docker runner on the server, we should first install docker on the same machine.

Install Docker

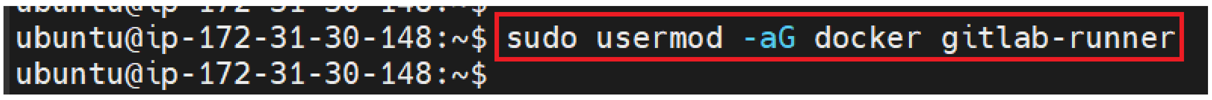
sudo apt install docker.io -y



```
ubuntu@ip-172-31-37-201:~$ sudo apt install docker.io -y
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base libidn11 pigz runc ubuntu-fan
Suggested packages:
  ifupdown aufs-tools cgroupfs-mount | cgroup-lite debootstrap docker-doc rinse zfs-fuse | zfsutils
The following NEW packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base docker.io libidn11 pigz runc ubuntu-fan
0 upgraded, 9 newly installed, 0 to remove and 66 not upgraded.
Need to get 69.2 MB of archives.
After this operation, 334 MB of additional disk space will be used.
Get:1 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu focal/universe amd64 pigz amd64 2.4-1 [57.4 kB]
Get:2 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu focal/main amd64 bridge-utils amd64 1.6-2ubuntu1 [30.5 kB]
Get:3 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu focal-updates/main amd64 runc amd64 1.1.0-0ubuntu1~20.04.2 [3894
Get:4 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu focal-updates/main amd64 containerd amd64 1.5.9-0ubuntu1~20.04.5
.0 MB]
```

Add Ubuntu User in Docker Group (Ubuntu Server)

sudo usermod -aG docker ubuntu



```
ubuntu@ip-172-31-30-148:~$ sudo usermod -aG docker gitlab-runner
ubuntu@ip-172-31-30-148:~$
```

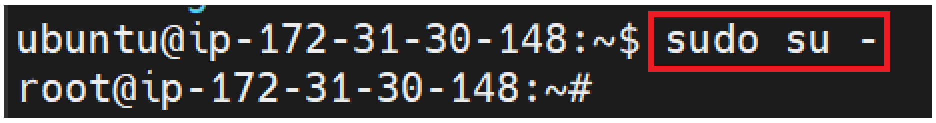
- **a**: add
- **G**: Group

Add Password For User (ubuntu, gitlab-runner)

- Now, we'll add/change (for the server whose password is not set) because in the next step while refreshing the terminal, it'll ask for the password.

- Firstly we'll move to the root user

sudo su -



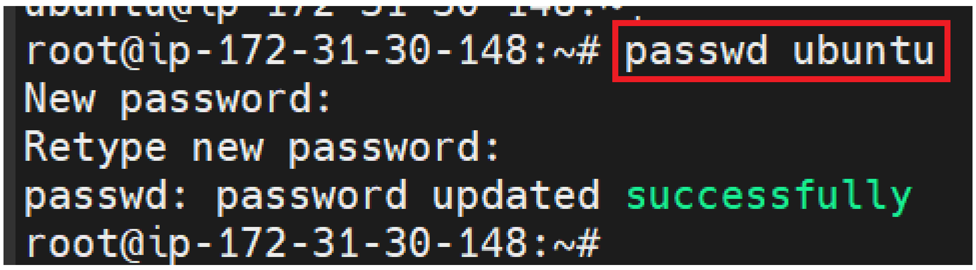
```
ubuntu@ip-172-31-30-148:~$ sudo su -
root@ip-172-31-30-148:~#
```

Now, we'll add the password using the command

passwd <username>

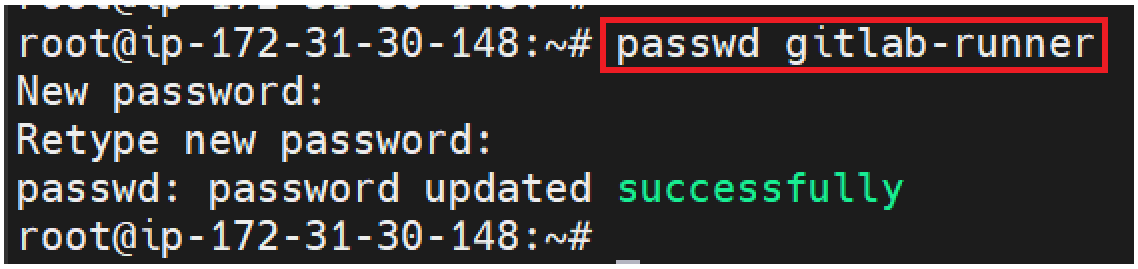
In our case, it'll be

passwd ubuntu



```
root@ip-172-31-30-148:~# passwd ubuntu
New password:
Retype new password:
passwd: password updated successfully
root@ip-172-31-30-148:~#
```

passwd gitlab-runner

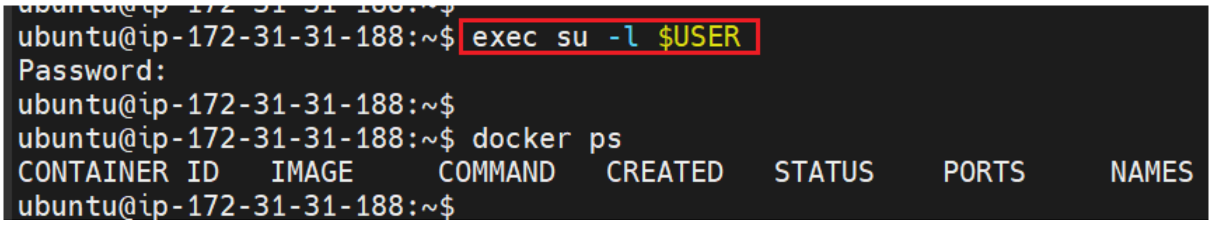


```
root@ip-172-31-30-148:~# passwd gitlab-runner
New password:
Retype new password:
passwd: password updated successfully
root@ip-172-31-30-148:~#
```

- The changes (Added Ubuntu in the Docker Group) will not reflect in the current terminal. So, we'll refresh the terminal using the command:

exec su -l $USER

- **exec**: It'll start a new process. So, our shell will get refreshed.
- **USER**: Current User "ubuntu".

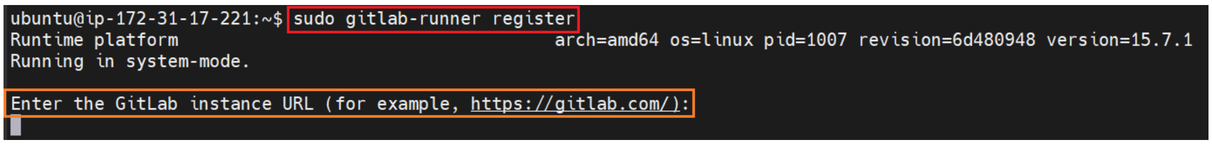


```
ubuntu@ip-172-31-31-188:~$ exec su -l $USER
Password:
ubuntu@ip-172-31-31-188:~$
ubuntu@ip-172-31-31-188:~$ docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
ubuntu@ip-172-31-31-188:~$
```

Register Docker Runner

sudo gitlab-runner register

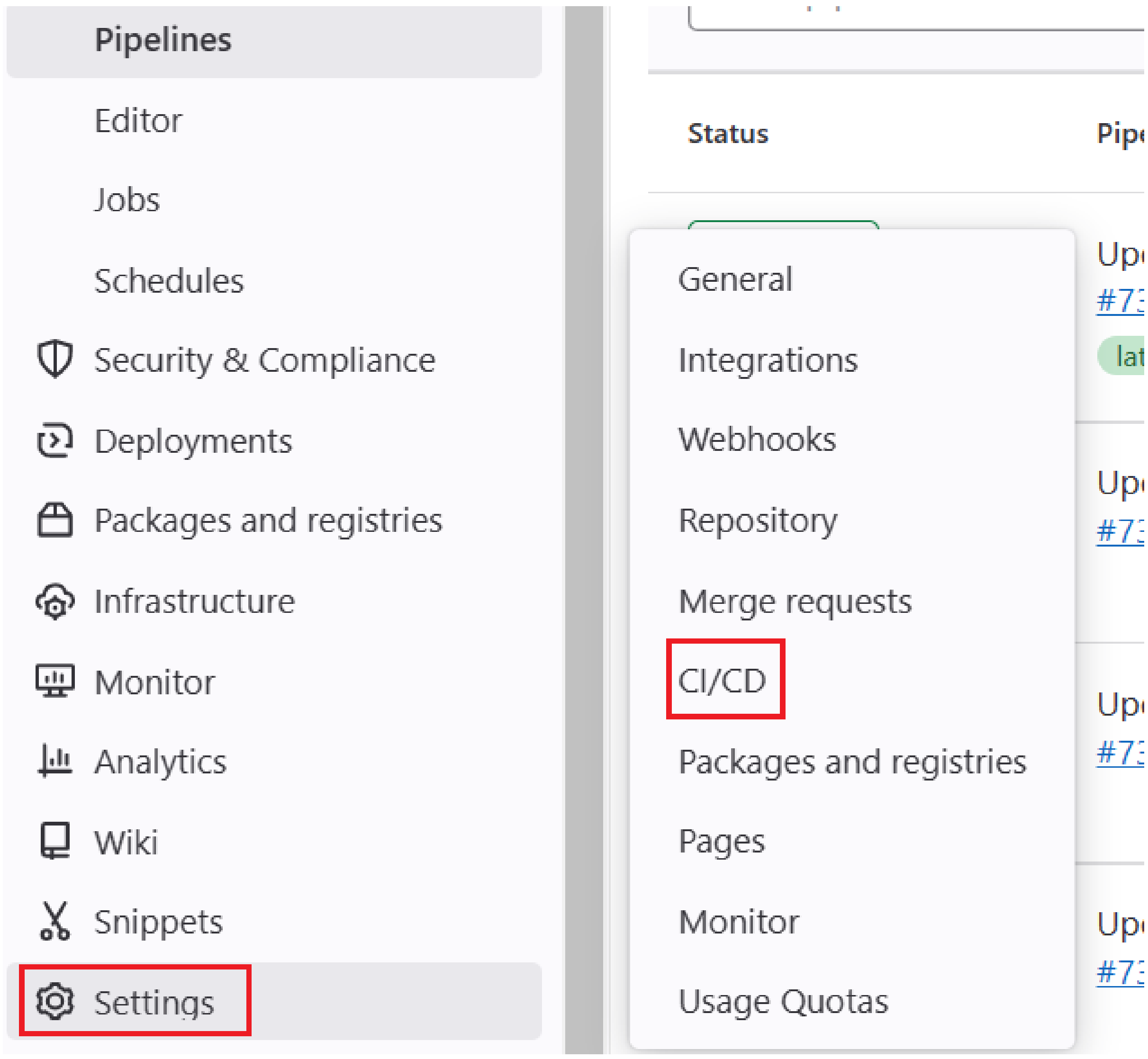
- Enter the GitLab instance URL



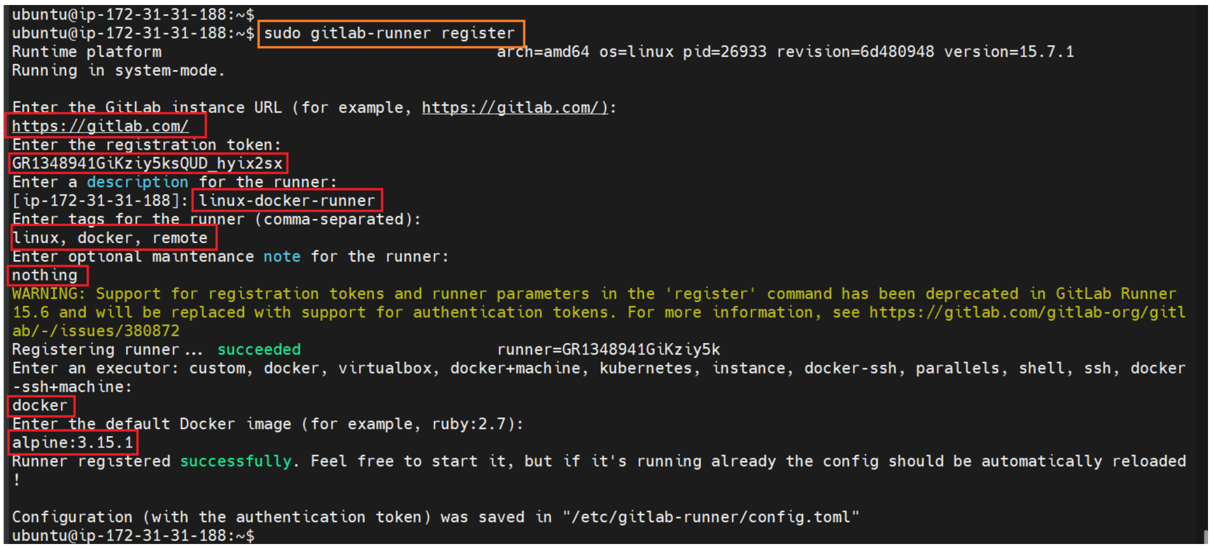
```
ubuntu@ip-172-31-17-221:~$ sudo gitlab-runner register
Runtime platform                                    arch=amd64 os=linux pid=1007 revision=6d480948 version=15.7.1
Running in system-mode.

Enter the GitLab instance URL (for example, https://gitlab.com/):
```

- Open Gitlab

Project > Setting > CI/CD

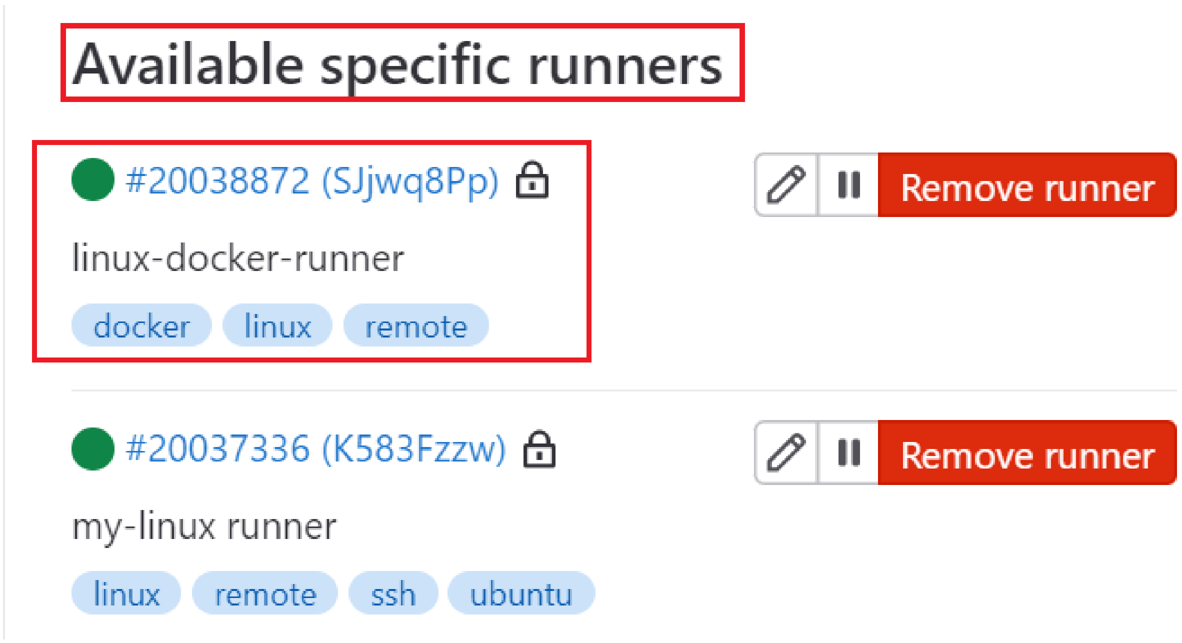


CI/CD > Runners > Expand > **Specific runners** > Copy & Paste the link > https://gitlab.com/ > Copy the Registration token > Paste it into the section > Enter the registration token > **GR1348941GiKziy5ksQUD_hyix2sx** > Enter a description > **linux-docker-runner** > Enter tags for the runner > **linux, docker, remote** > Any Note > **Nothing** > Enter an executor > **docker** > Enter the default Docker image > **alpine:3.15.1**



```
ubuntu@ip-172-31-31-188:~$
ubuntu@ip-172-31-31-188:~$ sudo gitlab-runner register
Runtime platform                                    arch=amd64 os=linux pid=26933 revision=6d480948 version=15.7.1
Running in system-mode.

Enter the GitLab instance URL (for example, https://gitlab.com/):
https://gitlab.com/
Enter the registration token:
GR1348941GiKziy5ksQUD_hyix2sx
Enter a description for the runner:
[ip-172-31-31-188]: linux-docker-runner
Enter tags for the runner (comma-separated):
linux, docker, remote
Enter optional maintenance note for the runner:
nothing
WARNING: Support for registration tokens and runner parameters in the 'register' command has been deprecated in GitLab Runner
15.6 and will be replaced with support for authentication tokens. For more information, see https://gitlab.com/gitlab-org/gitl
ab/-/issues/380872
Registering runner... succeeded                     runner=GR1348941GiKziy5k
Enter an executor: custom, docker, virtualbox, docker+machine, kubernetes, instance, docker-ssh, parallels, shell, ssh, docker
-ssh+machine:
docker
Enter the default Docker image (for example, ruby:2.7):
alpine:3.15.1
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 "/etc/gitlab-runner/config.toml"
ubuntu@ip-172-31-31-188:~$
```

- Now, reload the runner’s section of the project in gitlab and then we’ll see the gitlab-runner has been added in the sub-section “**Available specific runners**”.



Available specific runners

#20038872 (SJjwq8Pp)

linux-docker-runner

docker linux remote

#20037336 (K583Fzzw)

my-linux runner

linux remote ssh ubuntu

Unit Test

- We'll run unit tests in the self-managed runner having **docker executor**.

```
run_unit_tests:
    tags:
      - docker
      - linux
      - remote
```

- Script for executing unit test:

```
run_unit_tests:
    tags:
      - docker
      - linux
      - remote
    script:
      - npm test
```

- But before that we have to move inside the directory "app" so that we can take the reference from **package.json**.

- And we need to run the command "npm install" as we need the dependencies for the test so that they can run.

- So the two commands need to be run before running the tests.

- So, we can include it in "**before_script:**"

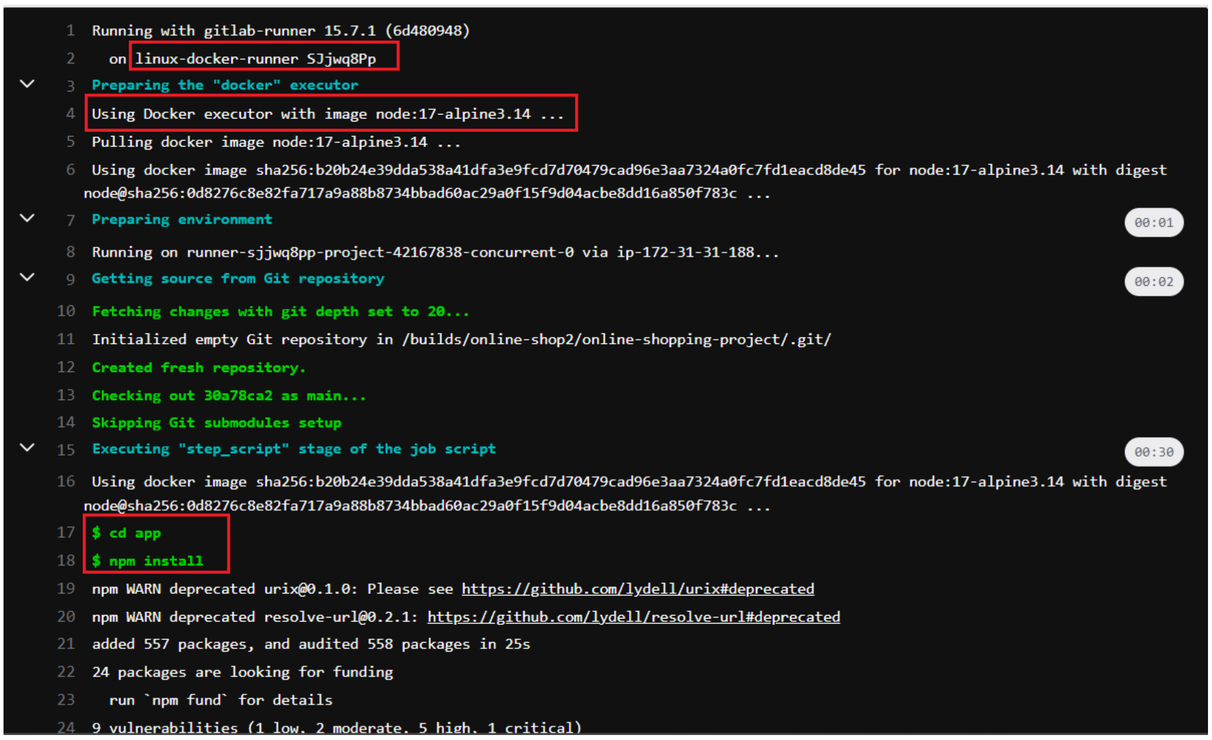
- And as we'll be running npm commands, we need those commands to be available inside our docker container.

  Therefore, we'll use an image which will be having those commands i.e **node** image.

Full Script:

```yaml
run_unit_tests:
    image: node:17-alpine3.14

    tags:
      - docker
      - linux
      - remote

    before_script:
      - cd app
      - npm install

    script:
      - npm test
```

Output:



```
Running with gitlab-runner 15.7.1 (6d480948)
  on linux-docker-runner SJjwq8Pp
Preparing the "docker" executor
Using Docker executor with image node:17-alpine3.14 ...
Pulling docker image node:17-alpine3.14 ...
Using docker image sha256:b20b24e39dda538a41dfa3e9fcd7d70479cad96e3aa7324a0fc7fd1eacd8de45 for node:17-alpine3.14 with digest
node@sha256:0d8276c8e82fa717a9a88b8734bbad60ac29a0f15f9d04acbe8dd16a850f783c ...
Preparing environment                                                                    00:01
Running on runner-sjjwq8pp-project-42167838-concurrent-0 via ip-172-31-31-188...
Getting source from Git repository                                                       00:02
Fetching changes with git depth set to 20...
Initialized empty Git repository in /builds/online-shop2/online-shopping-project/.git/
Created fresh repository.
Checking out 30a78ca2 as main...
Skipping Git submodules setup
Executing "step_script" stage of the job script                                          00:30
Using docker image sha256:b20b24e39dda538a41dfa3e9fcd7d70479cad96e3aa7324a0fc7fd1eacd8de45 for node:17-alpine3.14 with digest
node@sha256:0d8276c8e82fa717a9a88b8734bbad60ac29a0f15f9d04acbe8dd16a850f783c ...
$ cd app
$ npm install
npm WARN deprecated urix@0.1.0: Please see https://github.com/lydell/urix#deprecated
npm WARN deprecated resolve-url@0.2.1: https://github.com/lydell/resolve-url#deprecated
added 557 packages, and audited 558 packages in 25s
24 packages are looking for funding
  run `npm fund` for details
9 vulnerabilities (1 low, 2 moderate, 5 high, 1 critical)
```

Configure Tests Report

- Use the attribute named “**artifacts**”, and in that sub-attribute there is named “**reports**”.

- And we’ll be using “**junit.xml**” that jest gives us from the code inside the **app** folder.

Workflow:

- Junit report will collect the report from the xml file.

- Those collected unit test reports will be uploaded on gitlab as an **artifact**, that’s why we are using the attribute **artifacts**.

- And then gitlab will be able to visualize/display it for us in its UI.

- And we want the test to be uploaded every time, even when the test fails in order to check the failures of the tests. So, we’ll put the condition as

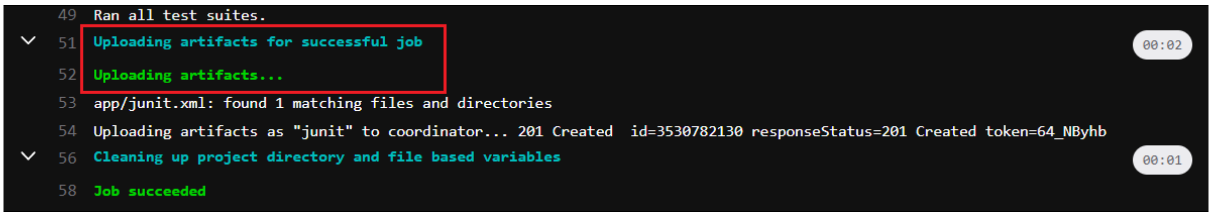
when: always

```
artifacts:
      when: always
      reports:
        junit: app/junit.xml
```

Full Script:

```yaml
run_unit_tests:
    image: node:17-alpine3.14

    tags:
      - docker
      - linux
      - remote

    before_script:
      - cd app
      - npm install

    script:
      - npm test

    artifacts:
      when: always
      reports:
        junit: app/junit.xml
```

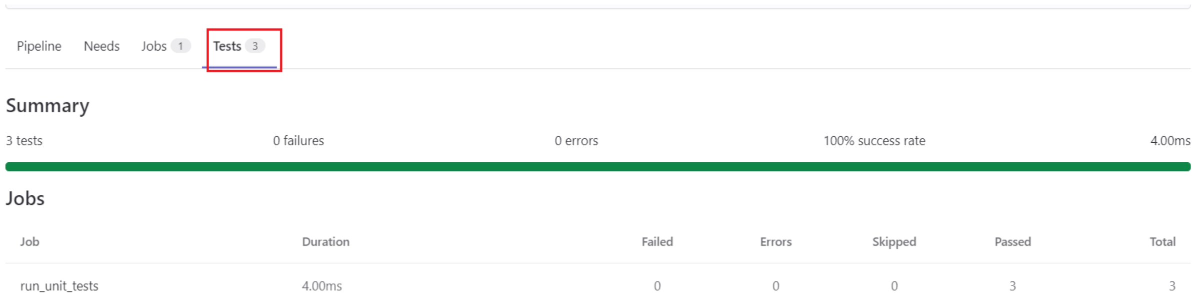
Output:



```
Ran all test suites.
Uploading artifacts for successful job                                                                         00:02
Uploading artifacts...
app/junit.xml: found 1 matching files and directories
Uploading artifacts as "junit" to coordinator... 201 Created  id=3530782130 responseStatus=201 Created token=64_NByhb
Cleaning up project directory and file based variables                                                         00:01
Job succeeded
```

Checking Test Reports

Project > CI/CD > Pipelines > **Tests**


Pipeline Needs Jobs 1 Tests 3

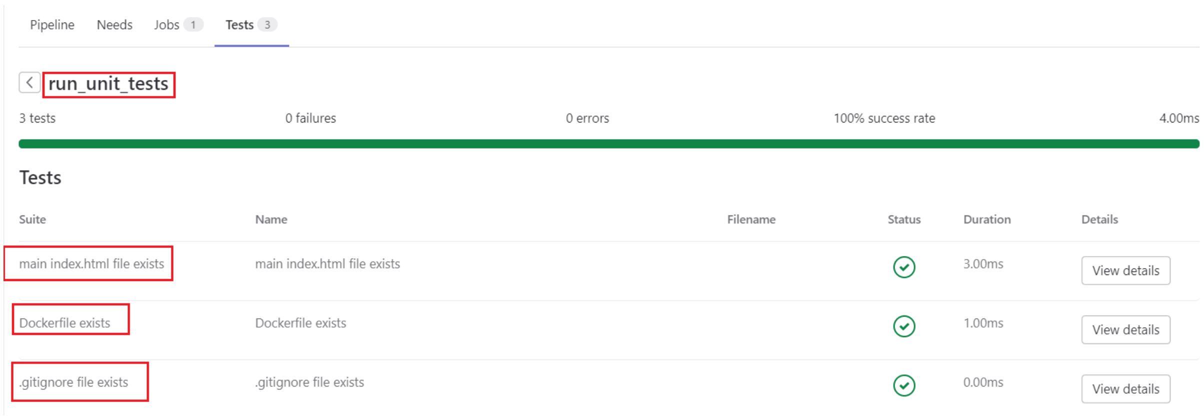
Summary

3 tests 0 failures 0 errors 100% success rate 4.00ms

Jobs

Job	Duration	Failed	Errors	Skipped	Passed	Total
run_unit_tests	4.00ms	0	0	0	3	3

- If we click on the test then it'll get expanded and show us a list of all the tests in that particular job.


Pipeline Needs Jobs 1 Tests 3

run_unit_tests

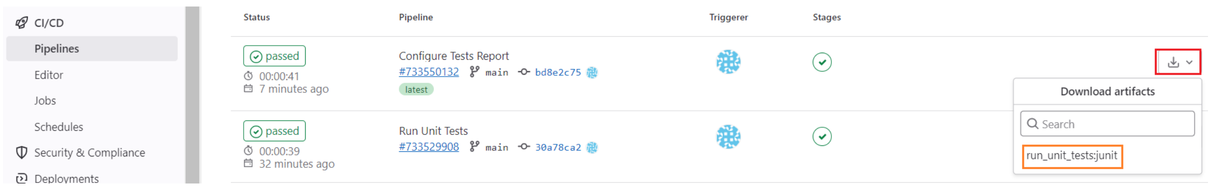
3 tests 0 failures 0 errors 100% success rate 4.00ms

Tests

Suite	Name	Filename	Status	Duration	Details
main index.html file exists	main index.html file exists		✓	3.00ms	View details
Dockerfile exists	Dockerfile exists		✓	1.00ms	View details
.gitignore file exists	.gitignore file exists		✓	0.00ms	View details

Downloading Artifacts

Project > Pipeline > Project > Download



Status	Pipeline	Triggerer	Stages
passed 00:00:41 7 minutes ago	Configure Tests Report #733550132 main bd8e2c75 latest		✓
passed 00:00:39 32 minutes ago	Run Unit Tests #733529908 main 30a78ca2		✓

Give A Path To Test Report

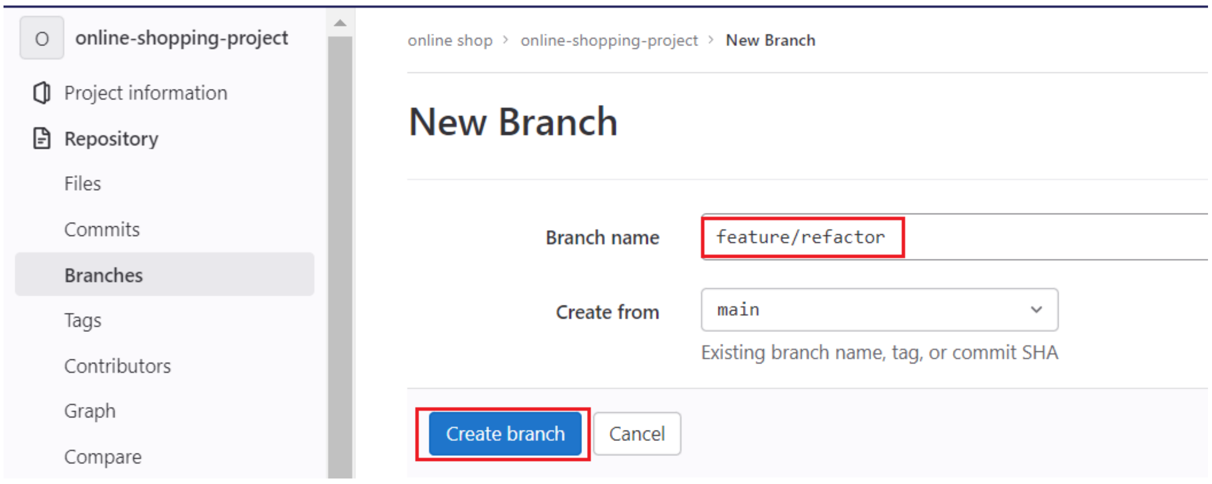
```
artifacts:
      when: always
      paths:
        - app/junit.xml
      reports:
        junit: app/junit.xml
```

Test Reports In Development Process

Create A New Branch

- Let's assume that we wanna refractor something in the project.
- So, we'll create a branch named "**feature/refactor**".

Project > Repository > Branches > **New Branch** > **feature/refactor** > Create Branch



Branch name: feature/refactor

Create from: main

Existing branch name, tag, or commit SHA

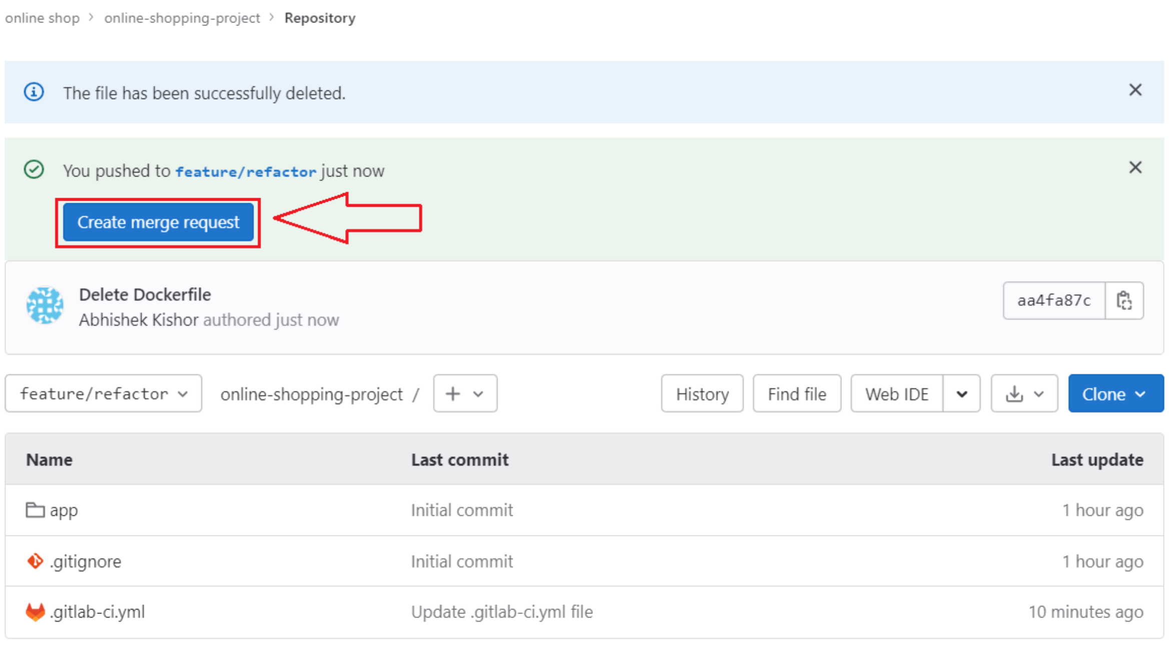
Delete Dockerfile Or (Any file)

- While refactoring we accidentally delete the dockerfile.

Create Merge Request

- Now, we wanna create a merge request.

Note: In test file we have mentioned about the presence of dockerfile.


online shop > online-shopping-project > Repository

The file has been successfully deleted.

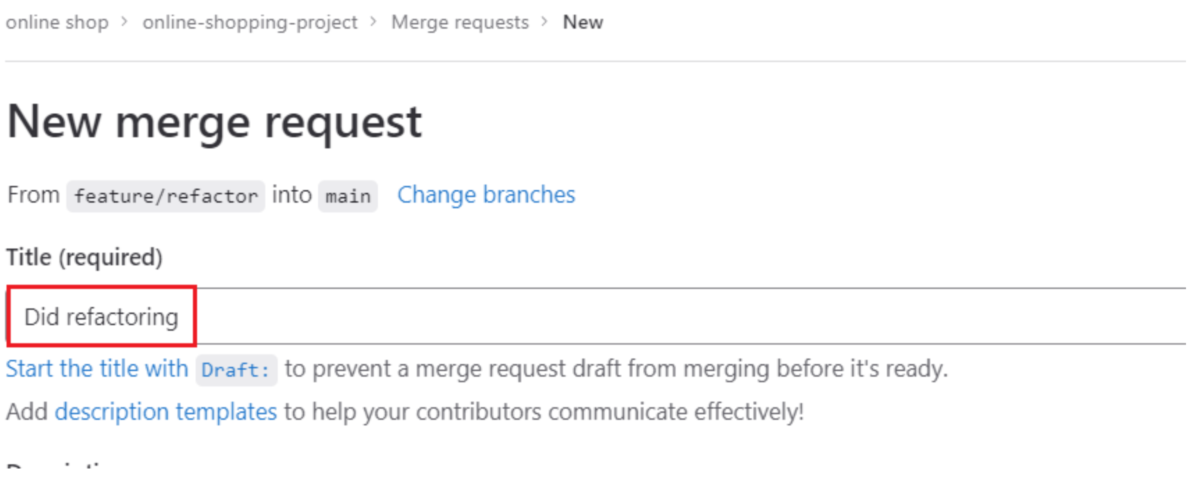
You pushed to `feature/refactor` just now

Create merge request

Delete Dockerfile
Abhishek Kishor authored just now
aa4fa87c

Name	Last commit	Last update
app	Initial commit	1 hour ago
.gitignore	Initial commit	1 hour ago
.gitlab-ci.yml	Update .gitlab-ci.yml file	10 minutes ago

New merge request > Title (required) > **Did refactoring** > Create Merge Request


online shop > online-shopping-project > Merge requests > New

New merge request

From `feature/refactor` into `main` Change branches

Title (required)

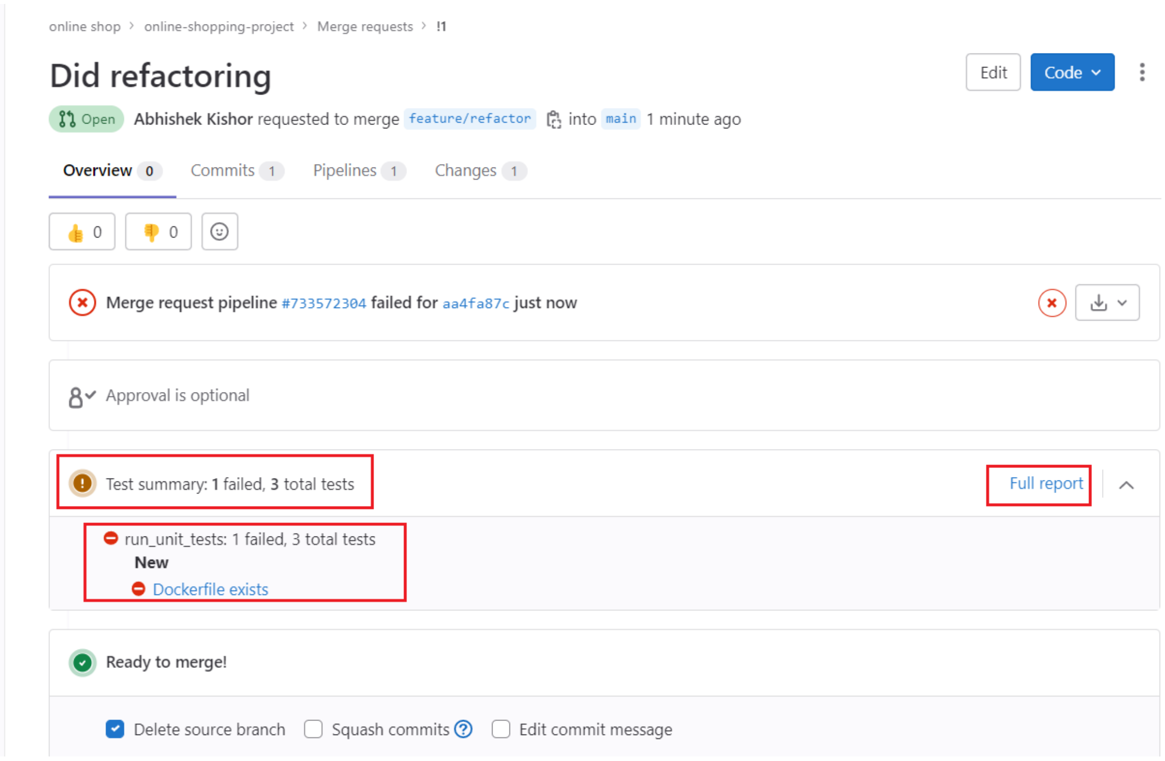
Did refactoring

Start the title with `Draft:` to prevent a merge request draft from merging before it's ready.

Add description templates to help your contributors communicate effectively!

- Now, other developers will review the merge request and see the status of it.

- As we have already written the pipeline code that if any merge request will be approved then the pipeline will get triggered.


online shop › online-shopping-project › Merge requests › !1

Did refactoring

Open Abhishek Kishor requested to merge feature/refactor into main 1 minute ago

Merge request pipeline #733572304 failed for aa4fa87c just now

Approval is optional

Test summary: 1 failed, 3 total tests

run_unit_tests: 1 failed, 3 total tests

New

Dockerfile exists

Ready to merge!

☑ Delete source branch ☐ Squash commits ☐ Edit commit message

- So, the developer can see that tests are failing coz of the merge requests.

Build Docker Image & Push To Private Repository

- Every Gitlab Project can have its own space to store its Docker images.

Packages & Registries

- Gitlab provides us Packages & Registries. And under it there are three types of registries.

  → **Package Registries**: Use Gitlab as a **Private** or **Public** Registry for a variety of supported package managers.

  - Used for general purpose, artifacts like zip files, jar, war files etc.

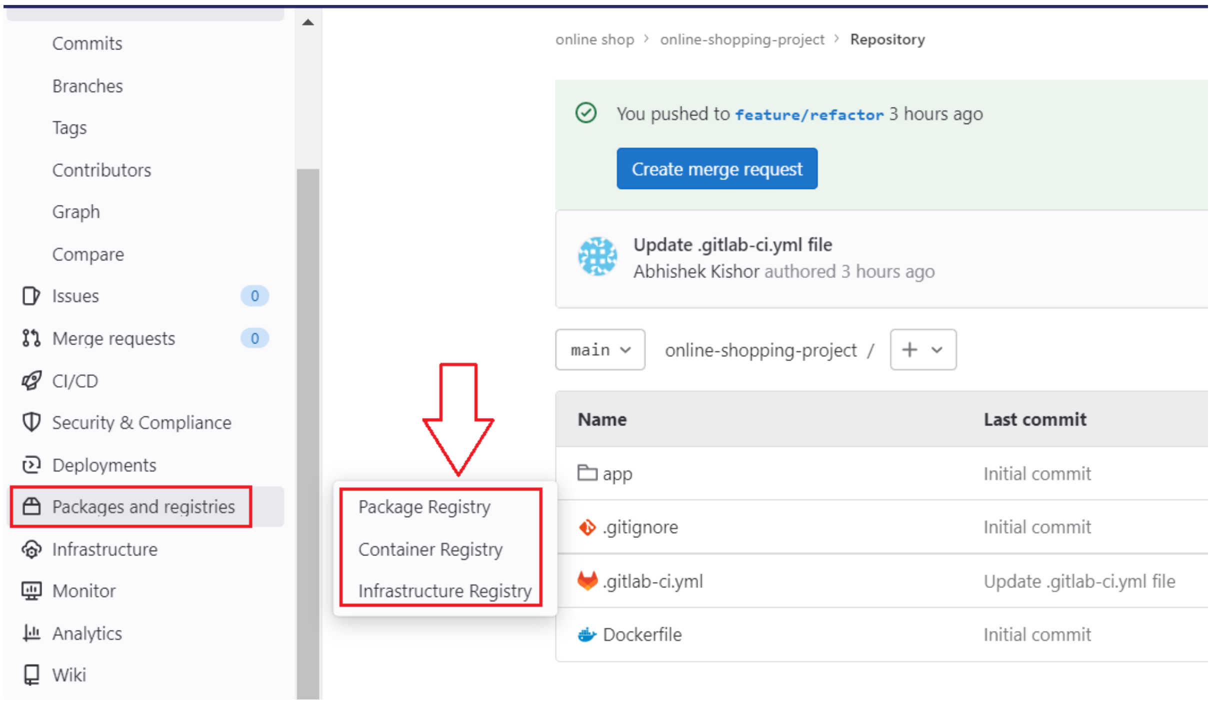
  → **Container Registries**: Registry to store Docker Private Images.

  - This is the place where we can push all the images that will be built for the application.

  - The name of the docker image for the application will be **registry.gitlab.com/username/projectname**

  → **Infrastructure Registries**: Private registry for infrastructure as code packages (Terraform).

  - We can write our own terraform modules for infrastructure provisioning which is related to the project or application, and we can host them at this place.



Name	Last commit
app	Initial commit
.gitignore	Initial commit
.gitlab-ci.yml	Update .gitlab-ci.yml file
Dockerfile	Initial commit

Build Docker Image

Project > CI/CD > **Editor**

- We'll run it on our managed gitlab-runner (Shell executor).

- As we have installed docker also on our **gitlab-runner**. So, docker commands will be available as well.

```
build_image:
  tags:
    - linux
    - remote
    - ubuntu
```

- Now, we'll write the script to build the docker image.

```
  script:
    - docker build -t
registry.gitlab.com/online-shop2/online-shopping-project:1.0
```

registry.gitlab.com/online-shop2/online-shopping-project: **Name of the Image for Gitlab CI Container Registry**

online-shop2: Username

online-shopping-project : Project Name

1.0: Version of the Image

Pushing Docker Image To Gitlab Registry

- We'll execute it on the same shell executor on our gitlab-runner.

```
push_image:
  tags:
    - linux
    - remote
    - ubuntu
```

Authenticate To Gitlab Private Registry

- Before pushing Image to a private repository, we need to authenticate using Docker Login.

- Gitlab provides temporary credentials for the container registry in our CI/CD Pipeline through **Environment Variables**.

CI_REGISTRY_USER
CI_REGISTRY_PASSWORD

- The value of these will only be valid for one job. So, even if the credentials get leaked, it can't be used again.

```
  before_script:
    - docker login -u $CI_REGISTRY_USER -p $CI_REGISTRY_PASSWORD
registry.gitlab.com
```

- Pushing the docker image that we'll build to the repository.

```
script:
    - docker push
registry.gitlab.com/online-shop2/online-shopping-project:1.0
```

Introduce Stages

- As all three jobs will be running parallelly and we don’t want that. Therefore, we’ll define them (Jobs) into **stages**.

- We’ll mention the stage at the top (after workflow).

```
workflow:
  rules:
    - if: $CI_COMMIT_BRANCH != "main" && $CI_PIPELINE_SOURCE !=
"merge_request_event"
      when: never
    - when: always

stages:
  - test
  - build
```

- Put the stages under each job so that one will trigger after completion of the previous job.

```
workflow:
  rules:
    - if: $CI_COMMIT_BRANCH != "main" && $CI_PIPELINE_SOURCE !=
"merge_request_event"
      when: never
    - when: always

stages:
  - test
  - build

run_unit_tests:
    image: node:17-alpine3.14
    stage: test
    tags:
      - docker
      - linux
      - remote

    before_script:
```

```yaml
      - cd app
      - npm install

    script:
      - npm test

    artifacts:
      when: always
      paths:
        - app/junit.xml
      reports:
        junit: app/junit.xml


build_image:
  stage: build
  tags:
    - linux
    - remote
    - ubuntu

  script:
    - docker build -t
registry.gitlab.com/online-shop2/online-shopping-project:1.0 .


push_image:
  stage: build
  tags:
    - linux
    - remote
    - ubuntu
  before_script:
    - docker login -u $CI_REGISTRY_USER -p $CI_REGISTRY_PASSWORD
registry.gitlab.com

  script:
    - docker push
registry.gitlab.com/online-shop2/online-shopping-project:1.0
```

Introduce Needs

- As the jobs “build_image” & “push_image” belong to the same stage i.e **build**, therefore they both will execute in parallel and we don’t want that.

- We’ll put the job “push_image” dependent on the build_image; So that if the image will built then only the image will be pushed.

- We’ll introduce an attribute “needs” in the job section “push_image”.

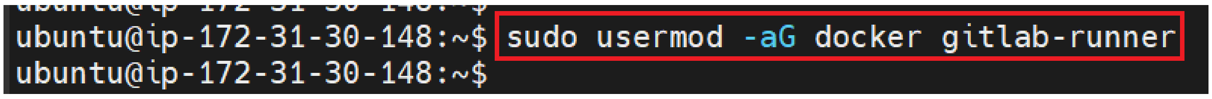
```
push_image:
  stage: build
  needs:
    - build_image
  tags:
    - linux
    - remote
    - ubuntu
  before_script:
    - docker login -u $CI_REGISTRY_USER -p $CI_REGISTRY_PASSWORD
registry.gitlab.com

  script:
    - docker push
registry.gitlab.com/online-shop2/online-shopping-project:1.0
```

- Commit the changes.

Note: We will get **permission error** if we don’t add the user “gitlab-runner” in the “docker” group.

Add gitlab-runner In docker Group

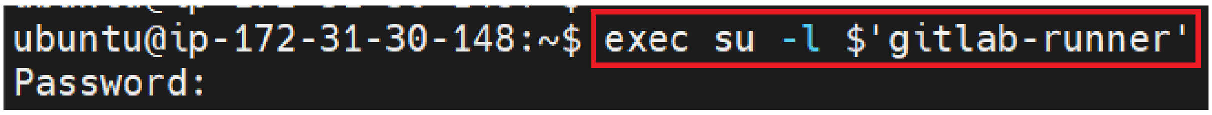
sudo usermod -aG docker gitlab-runner



```
ubuntu@ip-172-31-30-148:~$ sudo usermod -aG docker gitlab-runner
ubuntu@ip-172-31-30-148:~$
```

- The changes (Added Ubuntu in the Docker Group) will not reflect in the current terminal. So, we'll refresh the terminal using the command:

exec su -l $'gitlab-runner'

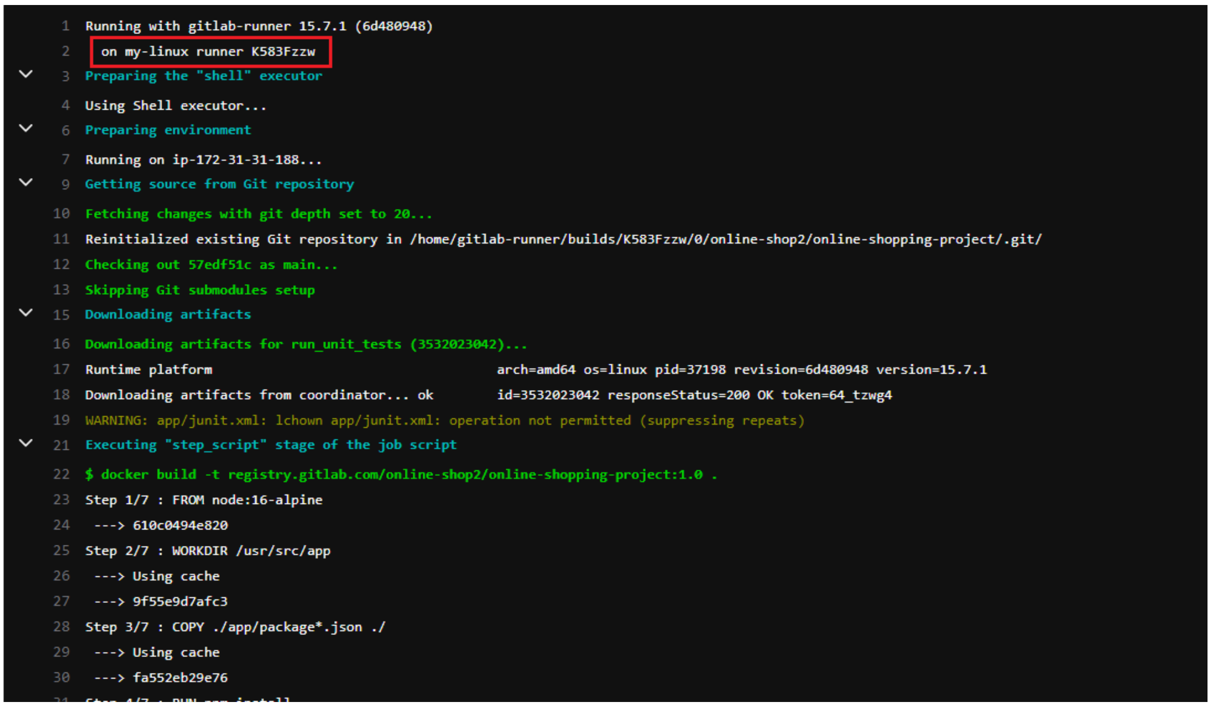


```
ubuntu@ip-172-31-30-148:~$ exec su -l $'gitlab-runner'
Password:
```

- **exec**: It'll start a new process. So, our shell will get refreshed.
- **gitlab-runner**: User name.

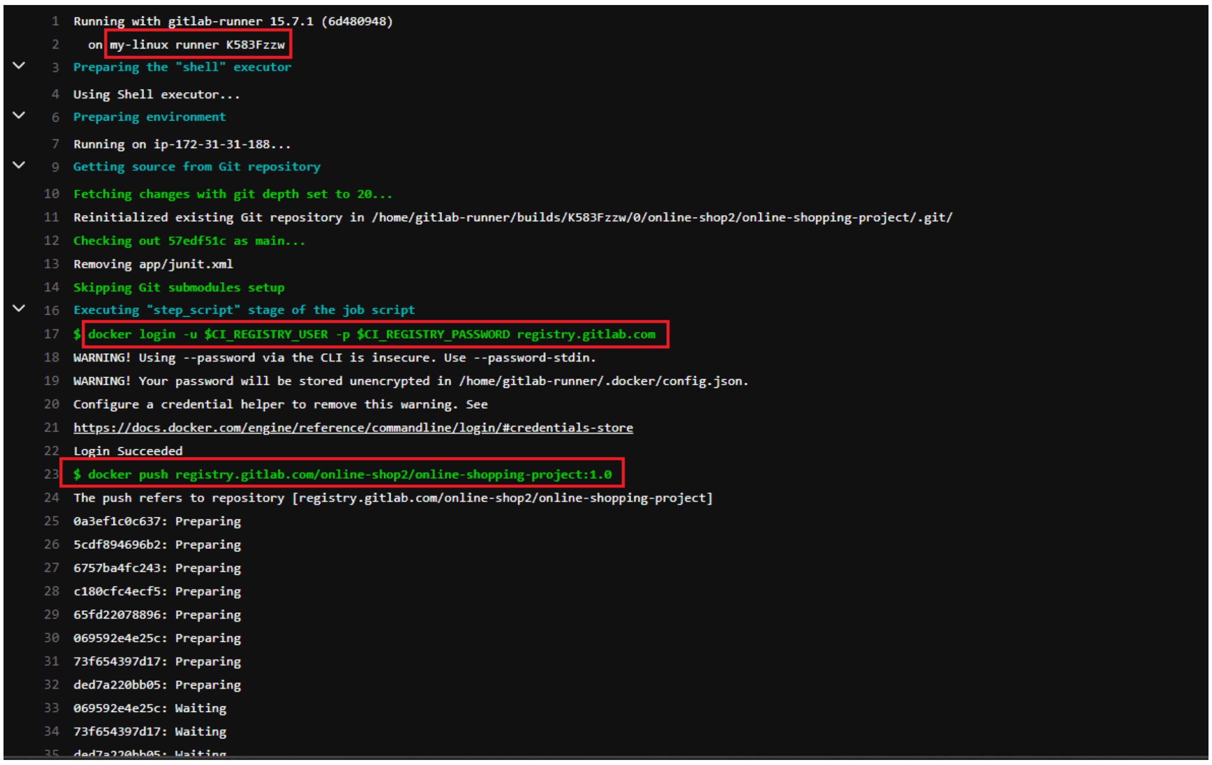
Output

Build Image



```
Running with gitlab-runner 15.7.1 (6d480948)
  on my-linux runner K583Fzzw
Preparing the "shell" executor
Using Shell executor...
Preparing environment
Running on ip-172-31-31-188...
Getting source from Git repository
Fetching changes with git depth set to 20...
Reinitialized existing Git repository in /home/gitlab-runner/builds/K583Fzzw/0/online-shop2/online-shopping-project/.git/
Checking out 57edf51c as main...
Skipping Git submodules setup
Downloading artifacts
Downloading artifacts for run_unit_tests (3532023042)...
Runtime platform                                    arch=amd64 os=linux pid=37198 revision=6d480948 version=15.7.1
Downloading artifacts from coordinator... ok        id=3532023042 responseStatus=200 OK token=64_tzwg4
WARNING: app/junit.xml: lchown app/junit.xml: operation not permitted (suppressing repeats)
Executing "step_script" stage of the job script
$ docker build -t registry.gitlab.com/online-shop2/online-shopping-project:1.0 .
Step 1/7 : FROM node:16-alpine
 ---> 610c0494e820
Step 2/7 : WORKDIR /usr/src/app
 ---> Using cache
 ---> 9f55e9d7afc3
Step 3/7 : COPY ./app/package*.json ./
 ---> Using cache
 ---> fa552eb29e76
Step 4/7 : RUN npm install
```

Push Image

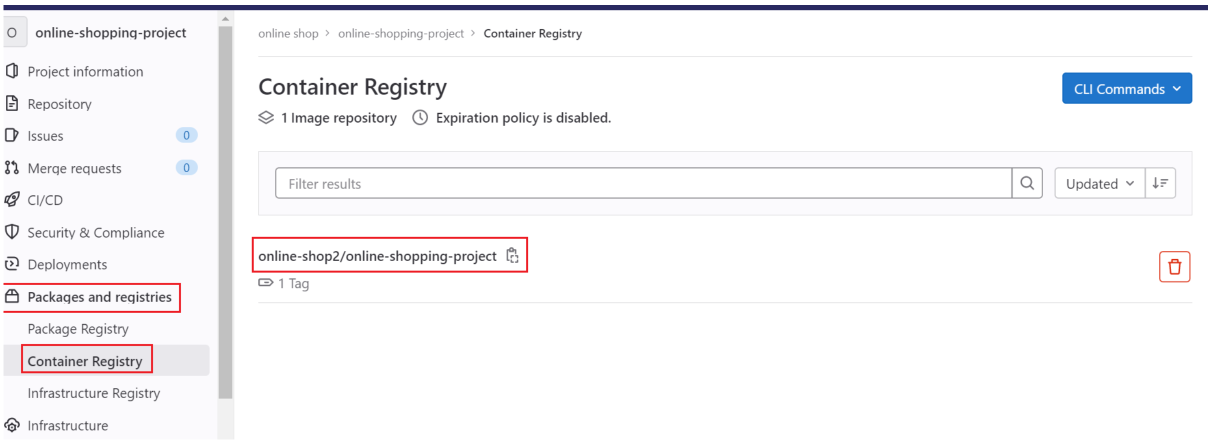


```
Running with gitlab-runner 15.7.1 (6d480948)
  on my-linux runner K583Fzzw
Preparing the "shell" executor
Using Shell executor...
Preparing environment
Running on ip-172-31-31-188...
Getting source from Git repository
Fetching changes with git depth set to 20...
Reinitialized existing Git repository in /home/gitlab-runner/builds/K583Fzzw/0/online-shop2/online-shopping-project/.git/
Checking out 57edf51c as main...
Removing app/junit.xml
Skipping Git submodules setup
Executing "step_script" stage of the job script
$ docker login -u $CI_REGISTRY_USER -p $CI_REGISTRY_PASSWORD registry.gitlab.com
WARNING! Using --password via the CLI is insecure. Use --password-stdin.
WARNING! Your password will be stored unencrypted in /home/gitlab-runner/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store
Login Succeeded
$ docker push registry.gitlab.com/online-shop2/online-shopping-project:1.0
The push refers to repository [registry.gitlab.com/online-shop2/online-shopping-project]
0a3ef1c0c637: Preparing
5cdf894696b2: Preparing
6757ba4fc243: Preparing
c180cfc4ecf5: Preparing
65fd22078896: Preparing
069592e4e25c: Preparing
73f654397d17: Preparing
ded7a220bb05: Preparing
069592e4e25c: Waiting
73f654397d17: Waiting
ded7a220bb05: Waiting
```

Checking The Image

- Now, we'll confirm if the image has been uploaded or not.

Project > **Packages and registries** > **Container Registry**



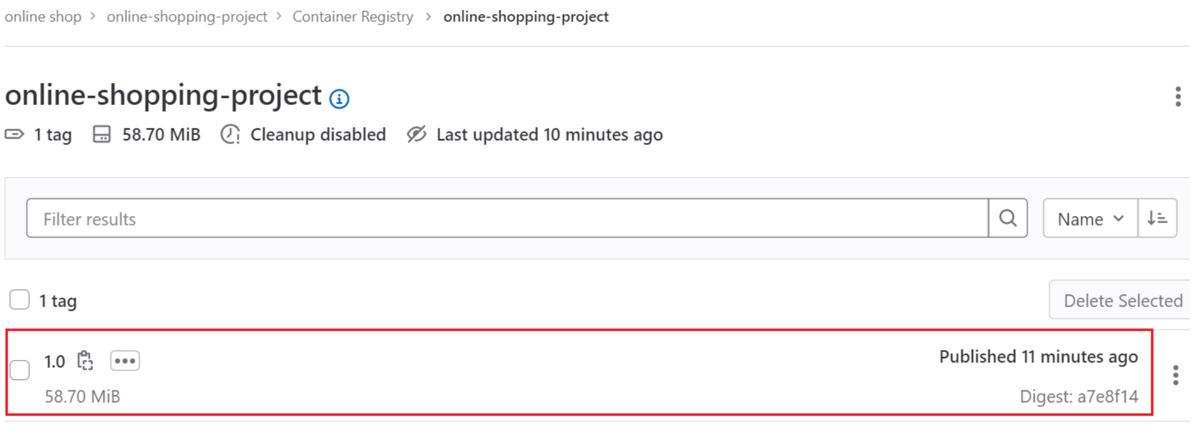
Container Registry

1 Image repository   Expiration policy is disabled.

online-shop2/online-shopping-project

1 Tag

- If we click on the image, we can see the version.



online shop > online-shopping-project > Container Registry > online-shopping-project
Filter results
Name
Delete Selected

online-shopping-project

1 tag   58.70 MiB   Cleanup disabled   Last updated 10 minutes ago

1 tag

1.0   Published 11 minutes ago

58.70 MiB   Digest: a7e8f14

Optimise Pipeline Configuration

- Issues:
    - We are using the image name at two different places in the pipeline code.
    - Image tag is hard-coded.
    - Registry name is hard-coded.
- We don't want the hardcoded values, instead we want to dynamically get the name of the image repository and reference it in the pipeline.
- **registry.gitlab.com**: General address of the Gitlab registry. So, we can reference it as a variable.

Before:

```
docker login -u $CI_REGISTRY_USER -p $CI_REGISTRY_PASSWORD
registry.gitlab.com
```

After:

```
docker login -u $CI_REGISTRY_USER -p $CI_REGISTRY_PASSWORD $CI_REGISTRY
```

- **registry.gitlab.com/online-shop2/online-shopping-project**: Image registry name.

Before:

```
script:
    - docker build -t
registry.gitlab.com/online-shop2/online-shopping-project:1.0 .
```

After:

```
script:
    - docker build -t $CI_REGISTRY_IMAGE:1.0 .
```

Before:

```
script:
    - docker push
registry.gitlab.com/online-shop2/online-shopping-project:1.0
```

After:

```
  script:
    - docker push $CI_REGISTRY_IMAGE:1.0
```

Full Code:

```
workflow:
  rules:
    - if: $CI_COMMIT_BRANCH != "main" && $CI_PIPELINE_SOURCE !=
"merge_request_event"
      when: never
    - when: always

stages:
  - test
  - build

run_unit_tests:
    image: node:17-alpine3.14
    stage: test
    tags:
      - docker
      - linux
      - remote

    before_script:
      - cd app
      - npm install

```

```yaml
    script:
      - npm test

    artifacts:
      when: always
      paths:
        - app/junit.xml
      reports:
        junit: app/junit.xml


build_image:
  stage: build
  tags:
    - linux
    - remote
    - ubuntu

  script:
    - docker build -t $CI_REGISTRY_IMAGE:1.0 .


push_image:
  stage: build
  needs:
    - build_image
  tags:
    - linux
    - remote
    - ubuntu
  before_script:
    - echo "Docker registry url is $CI_REGISTRY"
    - echo "Docker registry username is $CI_REGISTRY_USER"
    - echo "Docker image repo is $CI_REGISTRY_IMAGE"
    - docker login -u $CI_REGISTRY_USER -p $CI_REGISTRY_PASSWORD
$CI_REGISTRY
  script:
    - docker push $CI_REGISTRY_IMAGE:1.0
```

Assigning Own Name To Image

- We can assign different names to the docker image according to our needs.

- We can put the name after the registry image name followed with slash “/”.

Before:

```
  script:
    - docker build -t $CI_REGISTRY_IMAGE:1.0 .
```

After:

```
  script:
    - docker build -t $CI_REGISTRY_IMAGE/microservice/payment:1.0 .
```

Before:

```
script:
    - docker push $CI_REGISTRY_IMAGE:1.0
```

After:

```
  script:
    - docker push $CI_REGISTRY_IMAGE/microservice/payment:1.0
```

Assigning Image Name & Tag As Variable

- As, the size of the image is quite lengthy and we can have the tag as dynamic value, therefore we'll put both of them as variables (Global variable) and put them above all the jobs so that we can use those variables globally i.e in any jobs.

```
workflow:
  rules:
    - if: $CI_COMMIT_BRANCH != "main" && $CI_PIPELINE_SOURCE !=
"merge_request_event"
      when: never
    - when: always

variables:
  IMAGE_NAME: $CI_REGISTRY_IMAGE/microservice/payment
  IMAGE_TAG: "1.0"
```

Before:

```
script:
    - docker build -t $CI_REGISTRY_IMAGE/microservice/payment:1.0 .
```

After:

```
  script:
    - docker build -t $IMAGE_NAME:$IMAGE_TAG .
```

Before:

```
script:
    - docker push $CI_REGISTRY_IMAGE/microservice/payment:1.0
```

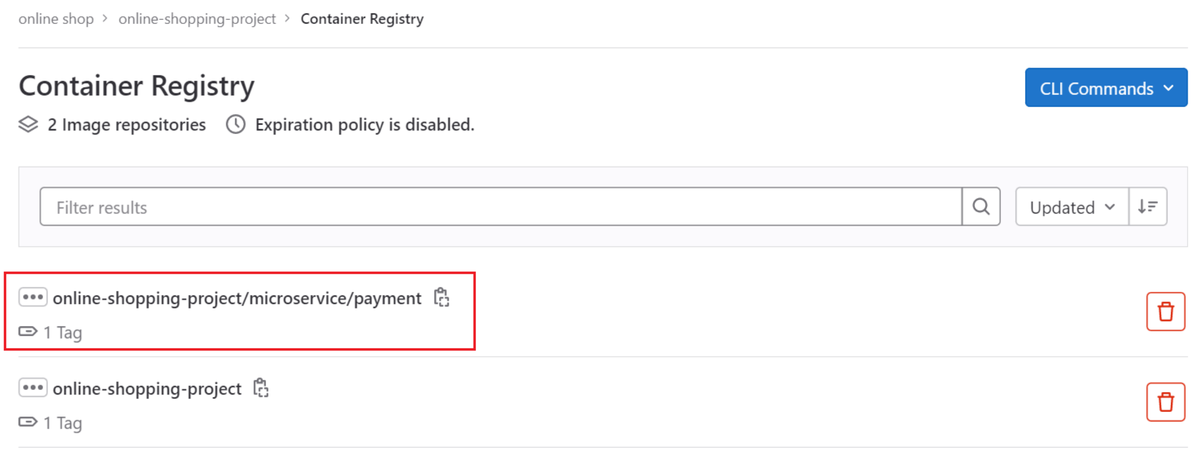
After:

```
script:
    - docker push $IMAGE_NAME:$IMAGE_TAG
```

Full Code

```
workflow:
  rules:
    - if: $CI_COMMIT_BRANCH != "main" && $CI_PIPELINE_SOURCE !=
"merge_request_event"
      when: never
    - when: always

variables:
  IMAGE_NAME: $CI_REGISTRY_IMAGE/microservice/payment
  IMAGE_TAG: "1.0"

stages:
  - test
  - build

run_unit_tests:
    image: node:17-alpine3.14
    stage: test
    tags:
      - docker
      - linux
      - remote

    before_script:
      - cd app
      - npm install

    script:
      - npm test

    artifacts:
      when: always
      paths:
        - app/junit.xml
      reports:
        junit: app/junit.xml


build_image:
  stage: build
  tags:
    - linux
    - remote
    - ubuntu

  script:
    - docker build -t $IMAGE_NAME:$IMAGE_TAG .


push_image:
  stage: build
  needs:
    - build_image
  tags:
    - linux
    - remote
    - ubuntu
  before_script:
    - echo "Docker registry url is $CI_REGISTRY"
    - echo "Docker registry username is $CI_REGISTRY_USER"
    - echo "Docker image repo is $CI_REGISTRY_IMAGE"
    - docker login -u $CI_REGISTRY_USER -p $CI_REGISTRY_PASSWORD
$CI_REGISTRY
  script:
    - docker push $IMAGE_NAME:$IMAGE_TAG
```

- Now, if we'll check the **Container Registry** again, then there will be two images as we have customised the image name.



Container Registry

2 Image repositories   Expiration policy is disabled.

online-shopping-project/microservice/payment
1 Tag

online-shopping-project
1 Tag

Pushing Image To Docker Hub

- We can't push the image to Docker Hub using the current script as we'll have to add the credentials for Docker Hub using the variable **CI_REGISTRY_USER** & **CI_REGISTRY_PASSWORD**.

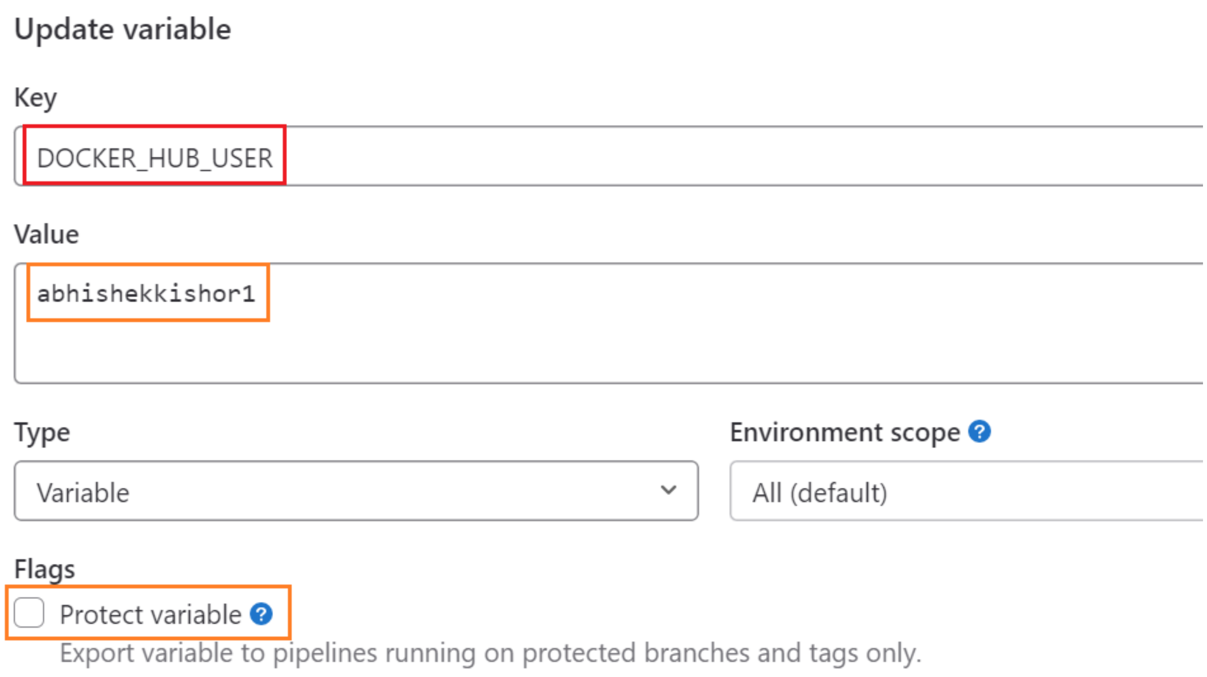
- But we can take another variable.

Assigning Variables To Docker Hub Credentials

- We'll assign the variables for the Docker Hub Credentials.

Project > Setting > CI/CD > **Variables** > Expand

Add Variable > Key > **DOCKER_HUB_USER** > Value > abhishekkishor1 > Untick Protect variable


Update variable

Key
DOCKER_HUB_USER

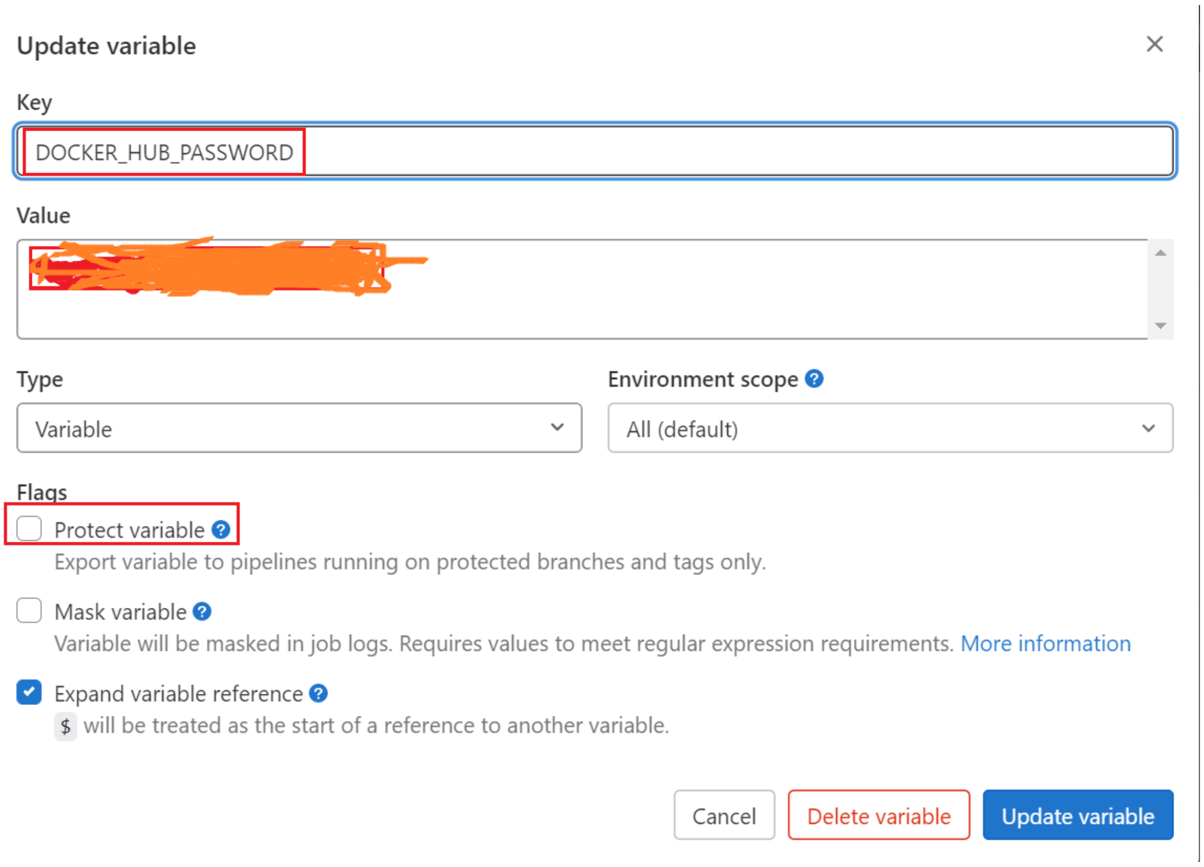
Value
abhishekkishor1

Type
Variable

Environment scope
All (default)

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

Add Variable > Key > **DOCKER_HUB_PASSWORD** > Value > Itna Bewaquf Nhi hn Bhai Ki password bta de > Untick Protect variable



Update variable

Key

DOCKER_HUB_PASSWORD

Value

[redacted]

Type

Variable

Environment scope

All (default)

Flags

☐ Protect variable

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

☐ Mask variable

Variable will be masked in job logs. Requires values to meet regular expression requirements. More information

☑ Expand variable reference

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

Changing Image Name

- We can write the script to change the name of the image for docker. As “/” etc are not allowed in dockerhub.

- We’ll assign the variable globally (at the top before any job) “**DOCKER_IMAGE_NAME**” storing the name of the docker image

```
DOCKER_IMAGE_NAME: abhishekkishor1/online-shopping-project
```

Script:

```
change_image_name:
  stage: build
  tags:
    - linux
    - remote
    - ubuntu

  script:
    - docker image tag $IMAGE_NAME:$IMAGE_TAG
$DOCKER_IMAGE_NAME:$IMAGE_TAG
```

Job: **change_image_name**

Tag: We’ll build using Shell executor Gitlab Runner

Script to change the name of the image:

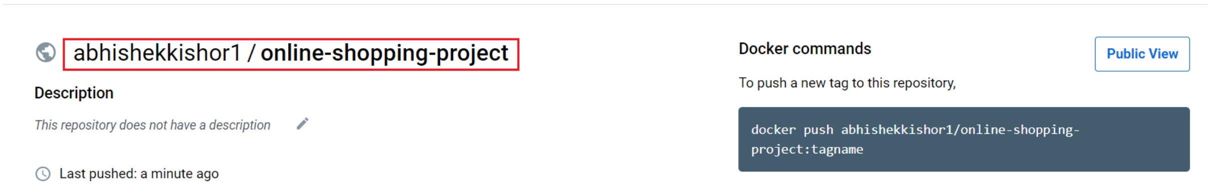
```
docker image tag $IMAGE_NAME:$IMAGE_TAG $DOCKER_IMAGE_NAME:$IMAGE_TAG
```

$IMAGE_NAME:$IMAGE_TAG → Name of the Image along with tag: 1.0 that was build in Gitlab CI Registry

$DOCKER_IMAGE_NAME:$IMAGE_TAG → Name of the Image that we wanna keep in order to push on Docker Hub. Tag will remain the same.

Pushing Image To Docker Hub

- Now, after changing the Image name we can push the image on docker hub.
- But before that create a repository on docker hub with the same name as the Changed Image.


abhishekkishor1 / online-shopping-project

Description

This repository does not have a description

Last pushed: a minute ago

Docker commands

To push a new tag to this repository,

```
docker push abhishekkishor1/online-shopping-
project:tagname
```

Script:

```
push_to_dockerhub:
  stage: build
  needs:
    # - build_docker_image
    - change_image_name
  tags:
    - linux
    - remote
    - ubuntu
  before_script:
    - docker login -u $DOCKER_HUB_USER -p $DOCKER_HUB_PASSWORD
  script:
    - docker push $DOCKER_IMAGE_NAME:$IMAGE_TAG
```

Login to Docker Hub

- We'll use the script before pushing the image to docker hub

$DOCKER_HUB_USER: Variable that stores the username of dockerhub.

$DOCKER_HUB_PASSWORD: Variable that stores the password of dockerhub.

$DOCKER_IMAGE_NAME: Variable that has been assigned to store the name of the docker Image.

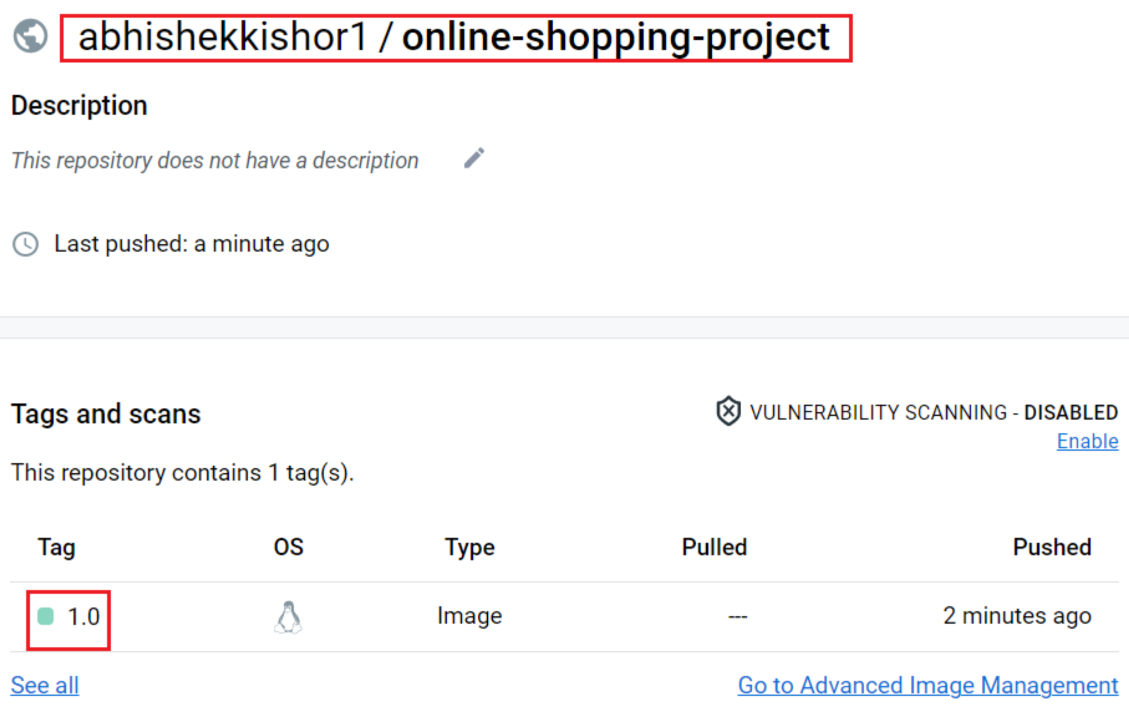
Full Script:

```
workflow:
  rules:
    - if: $CI_COMMIT_BRANCH != "main" && $CI_PIPELINE_SOURCE !=
"merge_request_event"
      when: never
    - when: always

variables:
  IMAGE_NAME: $CI_REGISTRY_IMAGE/microservice/payment
  DOCKER_IMAGE_NAME: abhishekkishor1/online-shopping-project
  IMAGE_TAG: "1.0"

stages:
  - test
  - build

run_unit_tests:
    image: node:17-alpine3.14
    stage: test
    tags:
      - docker
      - linux
      - remote

    before_script:
      - cd app
      - npm install

    script:
      - npm test

    artifacts:
      when: always
      paths:
        - app/junit.xml
      reports:
        junit: app/junit.xml
```

```yaml
build_image:
  stage: build
  tags:
    - linux
    - remote
    - ubuntu

  script:
    - docker build -t $IMAGE_NAME:$IMAGE_TAG .


push_image:
  stage: build
  needs:
    - build_image
  tags:
    - linux
    - remote
    - ubuntu
  before_script:
    - echo "Docker registry url is $CI_REGISTRY"
    - echo "Docker registry username is $CI_REGISTRY_USER"
    - echo "Docker image repo is $CI_REGISTRY_IMAGE"
    - docker login -u $CI_REGISTRY_USER -p $CI_REGISTRY_PASSWORD
$CI_REGISTRY
  script:
    - docker push $IMAGE_NAME:$IMAGE_TAG

# build_docker_image:
#   stage: build
#   tags:
#     - linux
#     - remote
#     - ubuntu

#   script:
#     - docker build -t $DOCKER_IMAGE_NAME:$IMAGE_TAG .

change_image_name:
  stage: build
  tags:
    - linux
    - remote
    - ubuntu

  script:
    - docker image tag $IMAGE_NAME:$IMAGE_TAG
$DOCKER_IMAGE_NAME:$IMAGE_TAG


push_to_dockerhub:
  stage: build
  needs:
    # - build_docker_image
    - change_image_name
  tags:
    - linux
    - remote
    - ubuntu
  before_script:
    - docker login -u $DOCKER_HUB_USER -p $DOCKER_HUB_PASSWORD
  script:
    - docker push $DOCKER_IMAGE_NAME:$IMAGE_TAG
```

Output:

Docker Hub:



abhishekkishor1 / **online-shopping-project**

Description

This repository does not have a description

Last pushed: a minute ago

Tags and scans

VULNERABILITY SCANNING - **DISABLED**
Enable

This repository contains 1 tag(s).

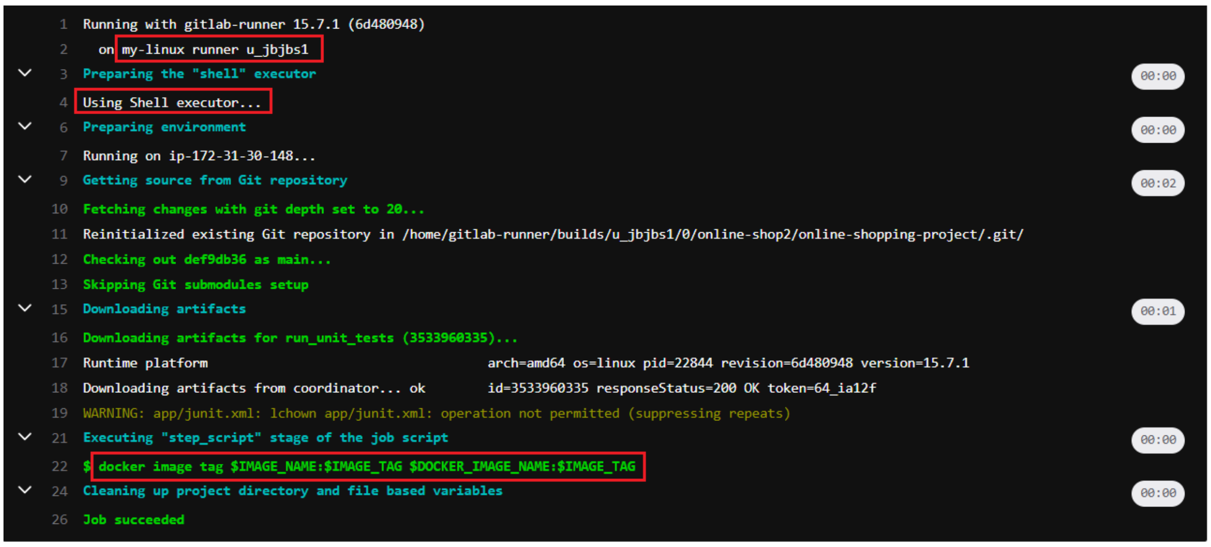
Tag	OS	Type	Pulled	Pushed
🟩 1.0		Image	---	2 minutes ago

See all

Go to Advanced Image Management

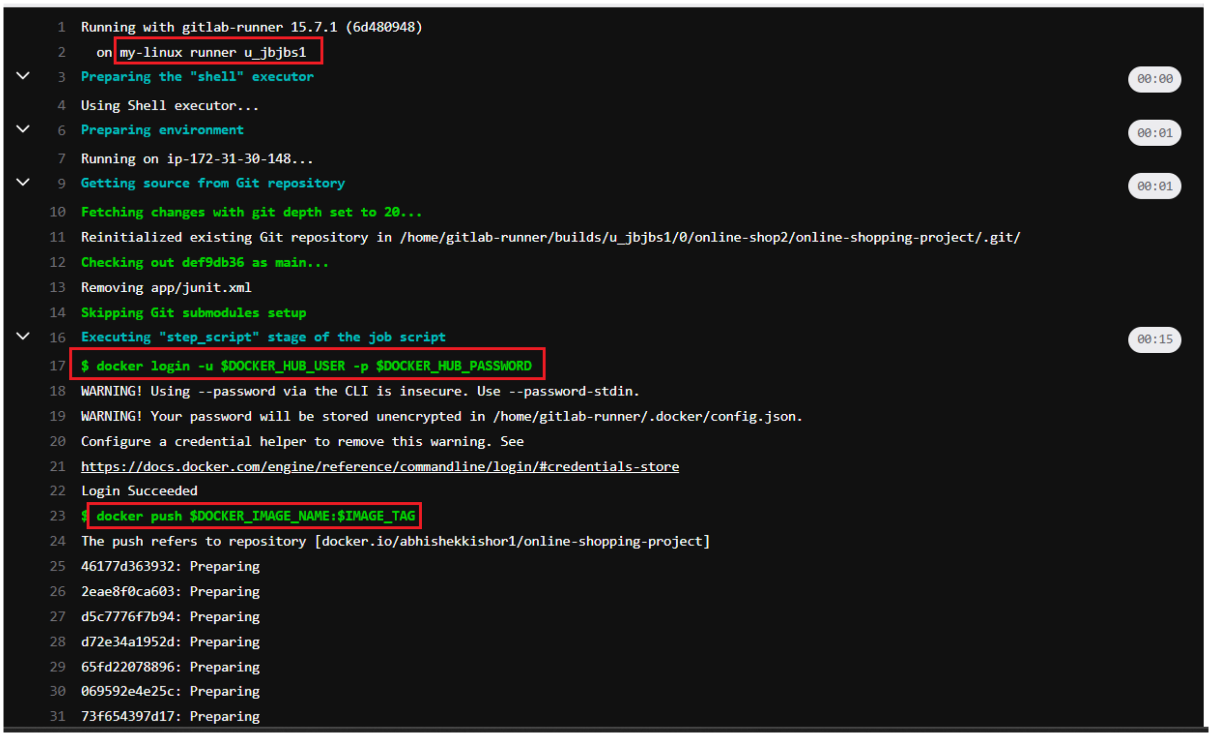
Console:

Changing Image Name



```
Running with gitlab-runner 15.7.1 (6d480948)
  on my-linux runner u_jbjbs1
Preparing the "shell" executor                                                          00:00
Using Shell executor...
Preparing environment                                                                   00:00
Running on ip-172-31-30-148...
Getting source from Git repository                                                      00:02
Fetching changes with git depth set to 20...
Reinitialized existing Git repository in /home/gitlab-runner/builds/u_jbjbs1/0/online-shop2/online-shopping-project/.git/
Checking out def9db36 as main...
Skipping Git submodules setup
Downloading artifacts                                                                   00:01
Downloading artifacts for run_unit_tests (3533960335)...
Runtime platform                                    arch=amd64 os=linux pid=22844 revision=6d480948 version=15.7.1
Downloading artifacts from coordinator... ok        id=3533960335 responseStatus=200 OK token=64_ia12f
WARNING: app/junit.xml: lchown app/junit.xml: operation not permitted (suppressing repeats)
Executing "step_script" stage of the job script                                         00:00
$ docker image tag $IMAGE_NAME:$IMAGE_TAG $DOCKER_IMAGE_NAME:$IMAGE_TAG
Cleaning up project directory and file based variables                                  00:00
Job succeeded
```

Pushing Image To Docker Hub



```
Running with gitlab-runner 15.7.1 (6d480948)
  on my-linux runner u_jbjbs1
Preparing the "shell" executor                                                          00:00
Using Shell executor...
Preparing environment                                                                   00:01
Running on ip-172-31-30-148...
Getting source from Git repository                                                      00:01
Fetching changes with git depth set to 20...
Reinitialized existing Git repository in /home/gitlab-runner/builds/u_jbjbs1/0/online-shop2/online-shopping-project/.git/
Checking out def9db36 as main...
Removing app/junit.xml
Skipping Git submodules setup
Executing "step_script" stage of the job script                                         00:15
$ docker login -u $DOCKER_HUB_USER -p $DOCKER_HUB_PASSWORD
WARNING! Using --password via the CLI is insecure. Use --password-stdin.
WARNING! Your password will be stored unencrypted in /home/gitlab-runner/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store
Login Succeeded
$ docker push $DOCKER_IMAGE_NAME:$IMAGE_TAG
The push refers to repository [docker.io/abhishekkishor1/online-shopping-project]
46177d363932: Preparing
2eae8f0ca603: Preparing
d5c7776f7b94: Preparing
d72e34a1952d: Preparing
65fd22078896: Preparing
069592e4e25c: Preparing
73f654397d17: Preparing
```

Deploy Image To Dev Server

- Remove a certain thing from the script as we are not having any microservice application. We are having just one application.

Before:

```
IMAGE_NAME: $CI_REGISTRY_IMAGE/microservice/payment
```

After:

```
IMAGE_NAME: $CI_REGISTRY_IMAGE
```

Adding A Job

- We'll now add another job for deployment on the **dev server**.

```
deploy_to_dev:
```

- In order to deploy on a dev server we need a dev server first.

Create & Configure A Dev Server

- We’ll create a dev server on AWS.

Creating a security group

- A security group acts as a firewall that controls the traffic allowed to reach one or more EC2 instances. When you launch an instance, you can assign it to one or more security groups.

- You add rules that control the traffic allowed to reach the instances in each security group. You can modify a security group’s rules any time, and the new rules take effect immediately.

- We will create a security group and add the following rules:

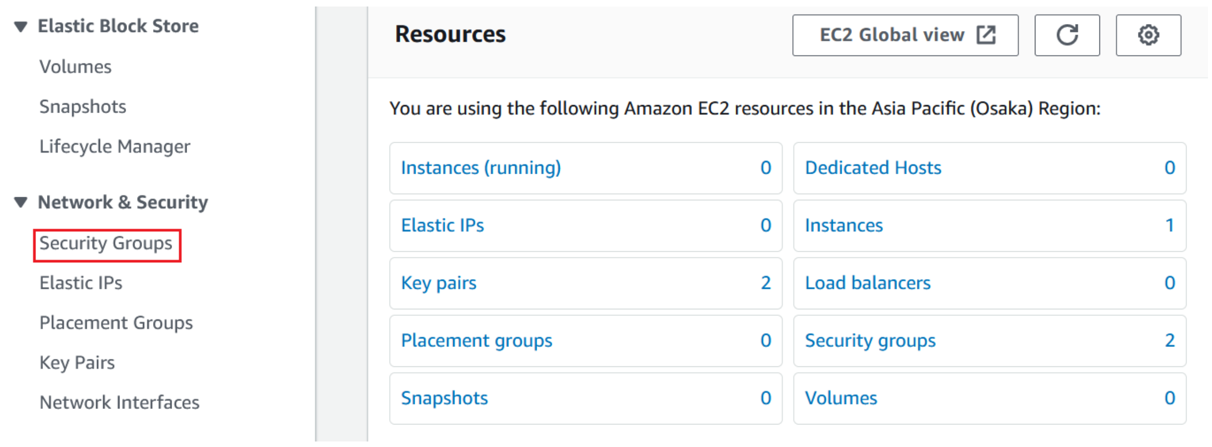
    - Allow inbound HTTP access from anywhere.

    - Allow inbound SSH traffic from anywhere.

Steps:

1. Open the Amazon EC2 console by selecting EC2 under Compute Or just search EC2 on the search bar.

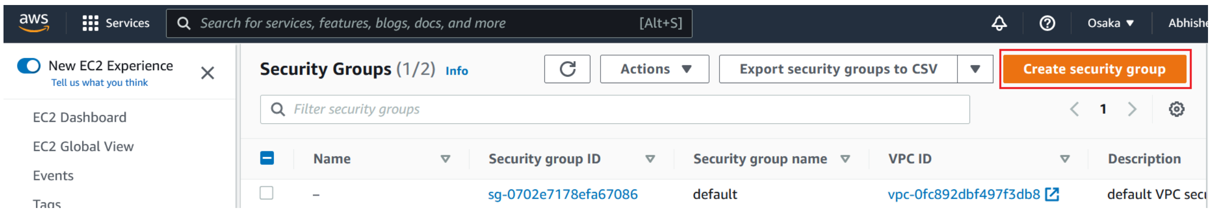
2. When we reach the Dashboard of EC2, we can see on the left-hand navigation bar, and select **Security Groups**.



You are using the following Amazon EC2 resources in the Asia Pacific (Osaka) Region:

Instances (running)	0	Dedicated Hosts	0
Elastic IPs	0	Instances	1
Key pairs	2	Load balancers	0
Placement groups	0	Security groups	2
Snapshots	0	Volumes	0

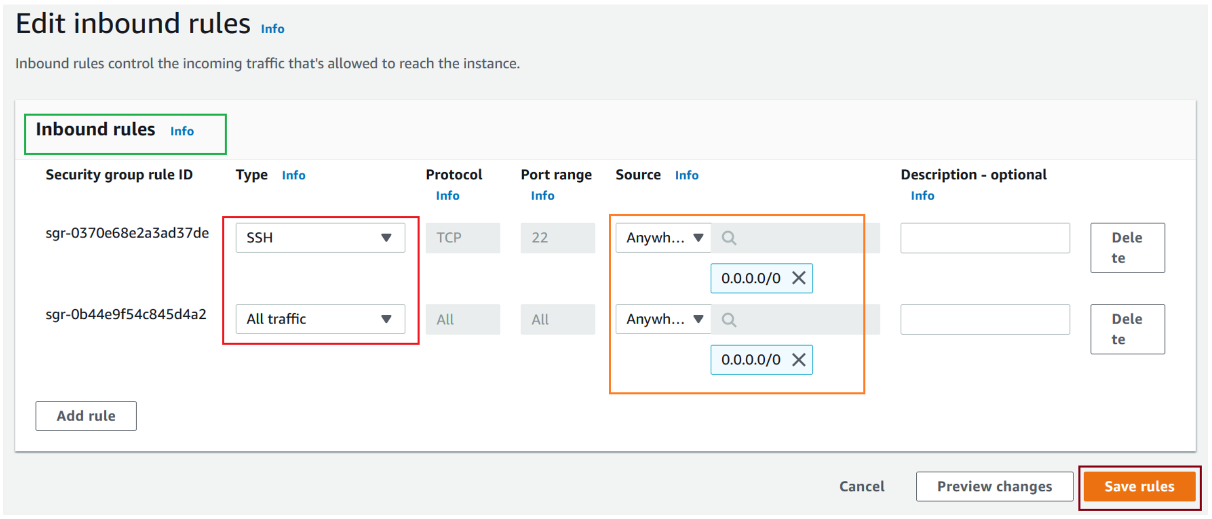
3. And then select **Create Security Group**.



	Name	Security group ID	Security group name	VPC ID	Description
☐	–	sg-0702e7178efa67086	default	vpc-0fc892dbf497f3db8	default VPC secu

Then on the **Inbound tab**, add the rules as follows:

- Select **Add Rule**, and then select **SSH** from the Type list. Under **Source**, select **Anywhere**.Select Add Rule.
- And then for the second rule select **HTTP** from the Type list. Under **Source**, select **Anywhere**.Select Add Rule. Select Create/Save Rules.

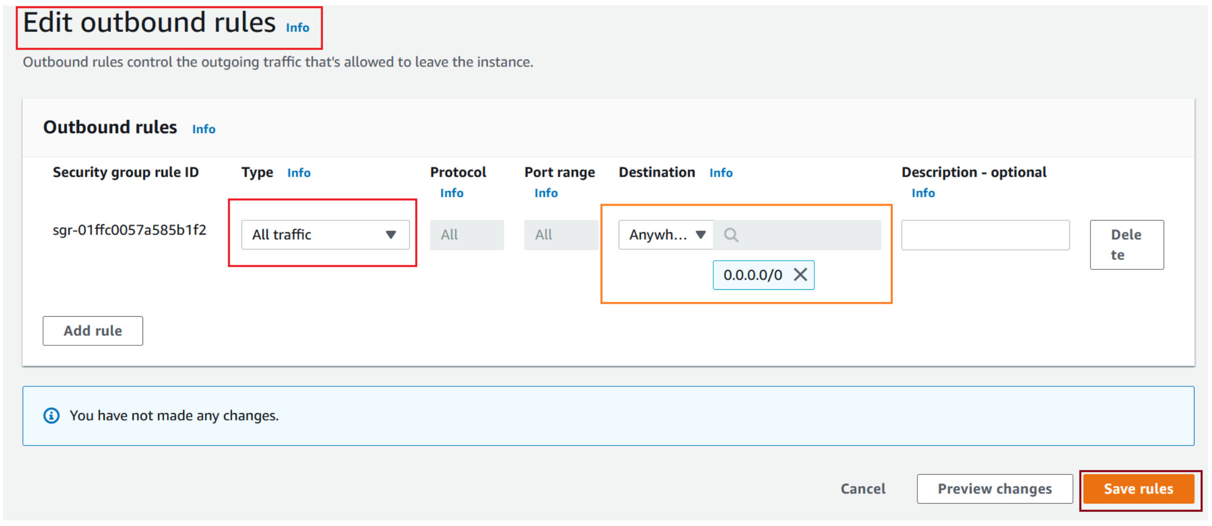


Edit inbound rules Info
Inbound rules control the incoming traffic that's allowed to reach the instance.
Inbound rules Info
Add rule
Cancel
Preview changes
Save rules

Security group rule ID	Type	Protocol	Port range	Source	Description - optional	
sgr-0370e68e2a3ad37de	SSH	TCP	22	Anywh... 0.0.0.0/0		Delete
sgr-0b44e9f54c845d4a2	All traffic	All	All	Anywh... 0.0.0.0/0		Delete

Then on the **Outbound tab**, add the rules as follows:

- Select **Add Rule**, and then select **All Traffic** from the Type list. Under **Source**, select **Anywhere**.Select Add Rule. Select Create/Save Rules.


Edit outbound rules Info

Outbound rules control the outgoing traffic that's allowed to leave the instance.

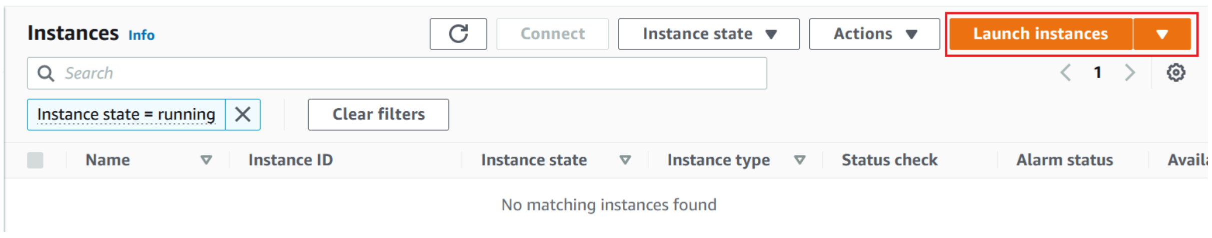
Outbound rules Info

Security group rule ID	Type	Protocol	Port range	Destination	Description - optional
sgr-01ffc0057a585b1f2	All traffic	All	All	Anywh... 0.0.0.0/0	

You have not made any changes.

Launching an Amazon EC2 instance

- Open the Amazon EC2 console by selecting **EC2** under Compute.
- From the Amazon EC2 dashboard, select **Launch Instance**.

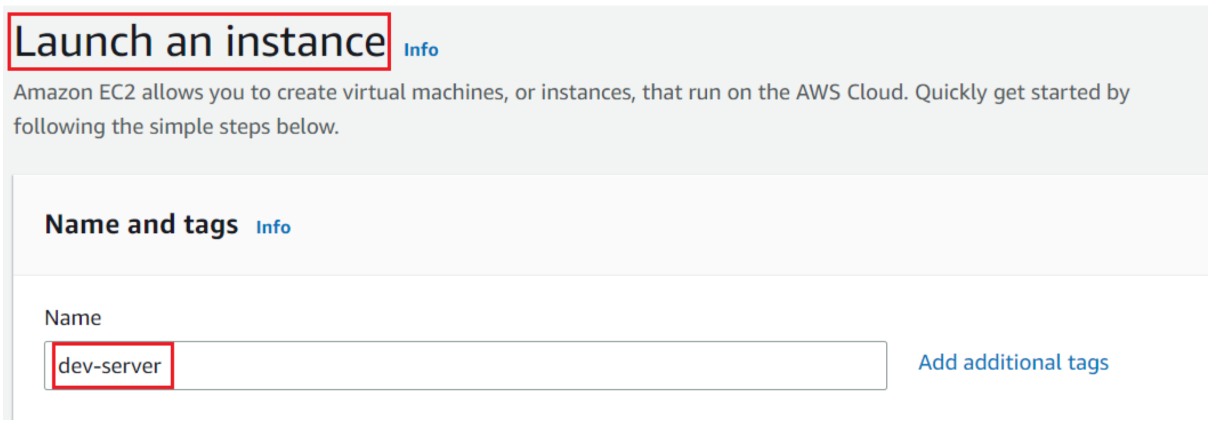

Instances Info

Instance state = running

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Avail

No matching instances found

1. Give the **Name and tags** to the Instance.



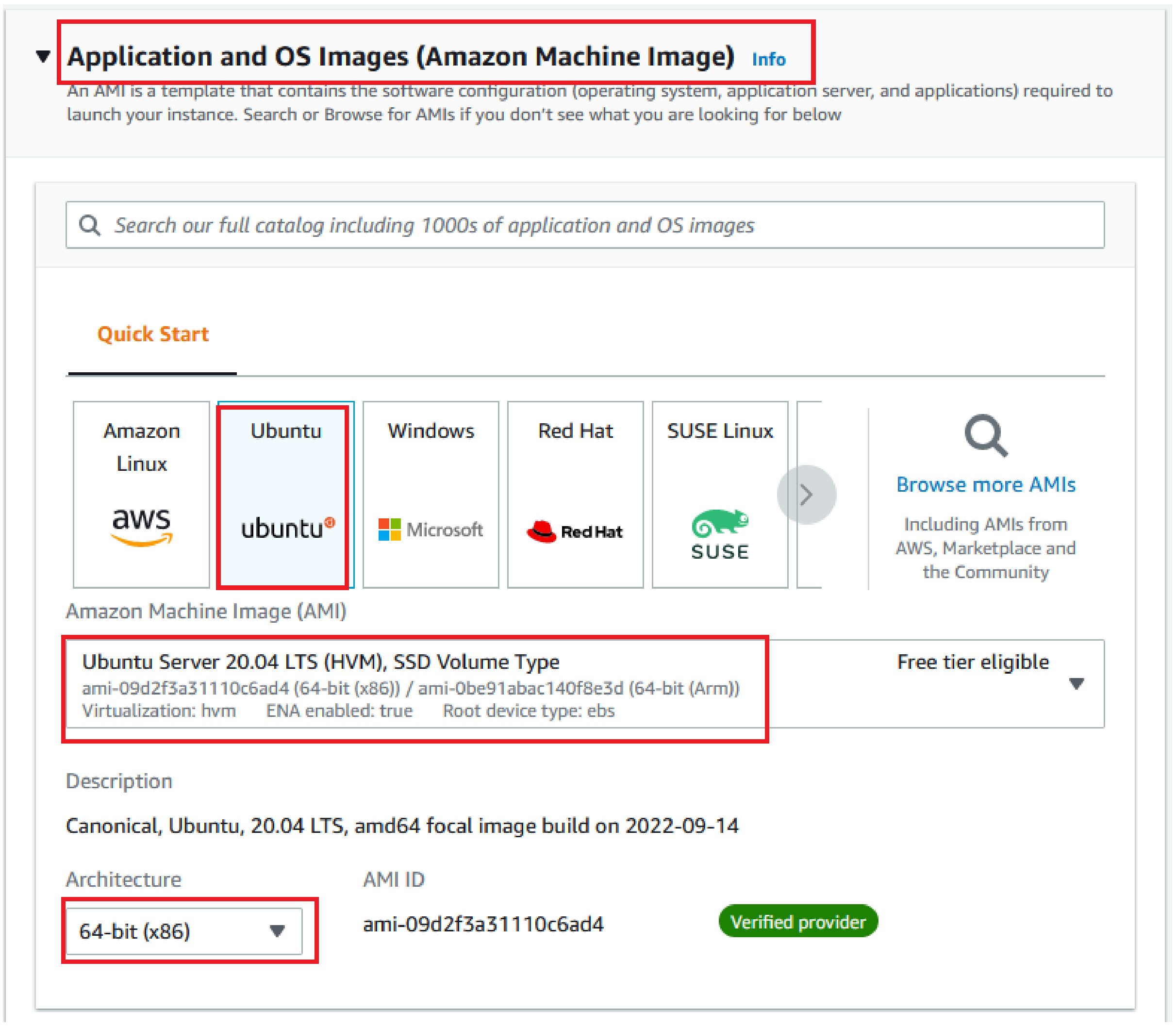
Launch an instance

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

Name and tags

Name: dev-server

1. In the **Application and OS Images (Amazon Machine Image)** section select **Ubuntu** and select **Amazon Machine Image (AMI)** as **Linux Server 20.04 LTS (HVM),SSD Volume Type**.



Application and OS Images (Amazon Machine Image)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Search our full catalog including 1000s of application and OS images

Amazon Machine Image (AMI)

Ubuntu Server 20.04 LTS (HVM), SSD Volume Type — Free tier eligible
ami-09d2f3a31110c6ad4 (64-bit (x86)) / ami-0be91abac140f8e3d (64-bit (Arm))
Virtualization: hvm ENA enabled: true Root device type: ebs

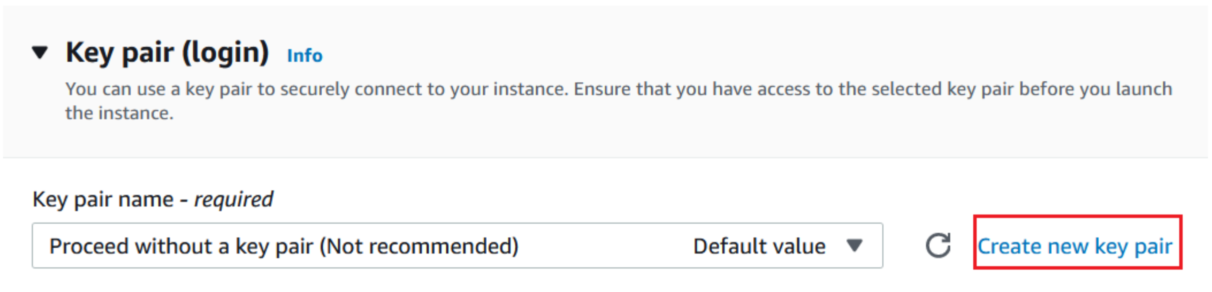
Description

Canonical, Ubuntu, 20.04 LTS, amd64 focal image build on 2022-09-14

Architecture: 64-bit (x86)

AMI ID: ami-09d2f3a31110c6ad4

2. Now, in the section **Key pair (login)**, select **Create new key pair** option.


Key pair (login) Info

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - *required*

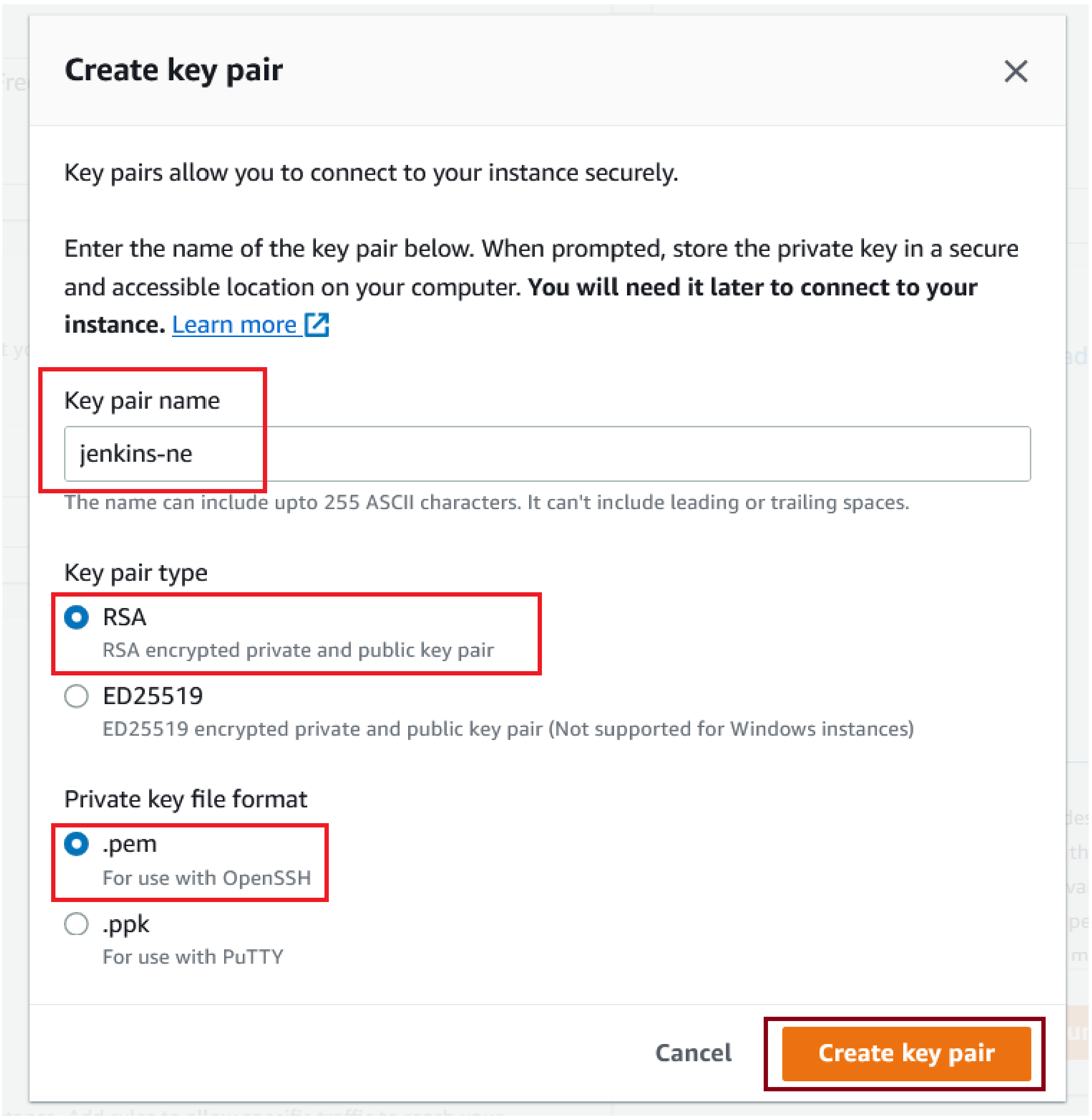
Proceed without a key pair (Not recommended) Default value

Create new key pair

- After that assign **Key pair name**, Then put the **Key pair type** as **RSA**. And then in the **Private key file format** section select **.pem** and then select **Create Key Pair**.

Note: Don't go on Name i have copy & paste this section from my Jenkins note..

Waise bhi naam me kya rakha h!!!!!!!



Create key pair

Key pairs allow you to connect to your instance securely.

Enter the name of the key pair below. When prompted, store the private key in a secure and accessible location on your computer. **You will need it later to connect to your instance.** Learn more

Key pair name

jenkins-ne

The name can include upto 255 ASCII characters. It can't include leading or trailing spaces.

Key pair type

[✓ RSA] RSA encrypted private and public key pair

ED25519 ED25519 encrypted private and public key pair (Not supported for Windows instances)

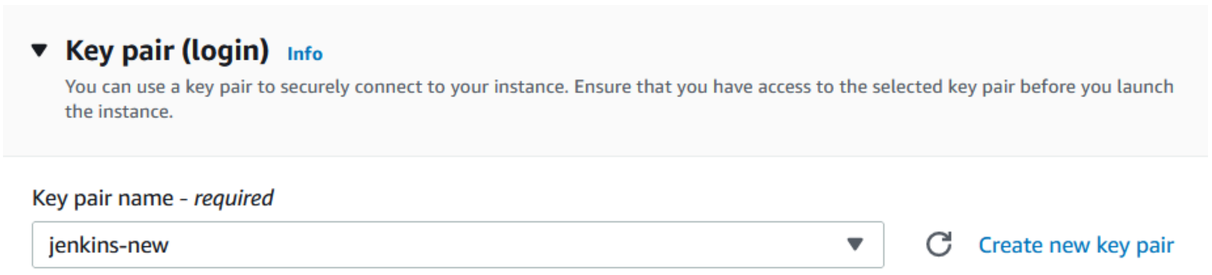
Private key file format

[✓ .pem] For use with OpenSSH

.ppk For use with PuTTY

And Download the Key.

OutPut:



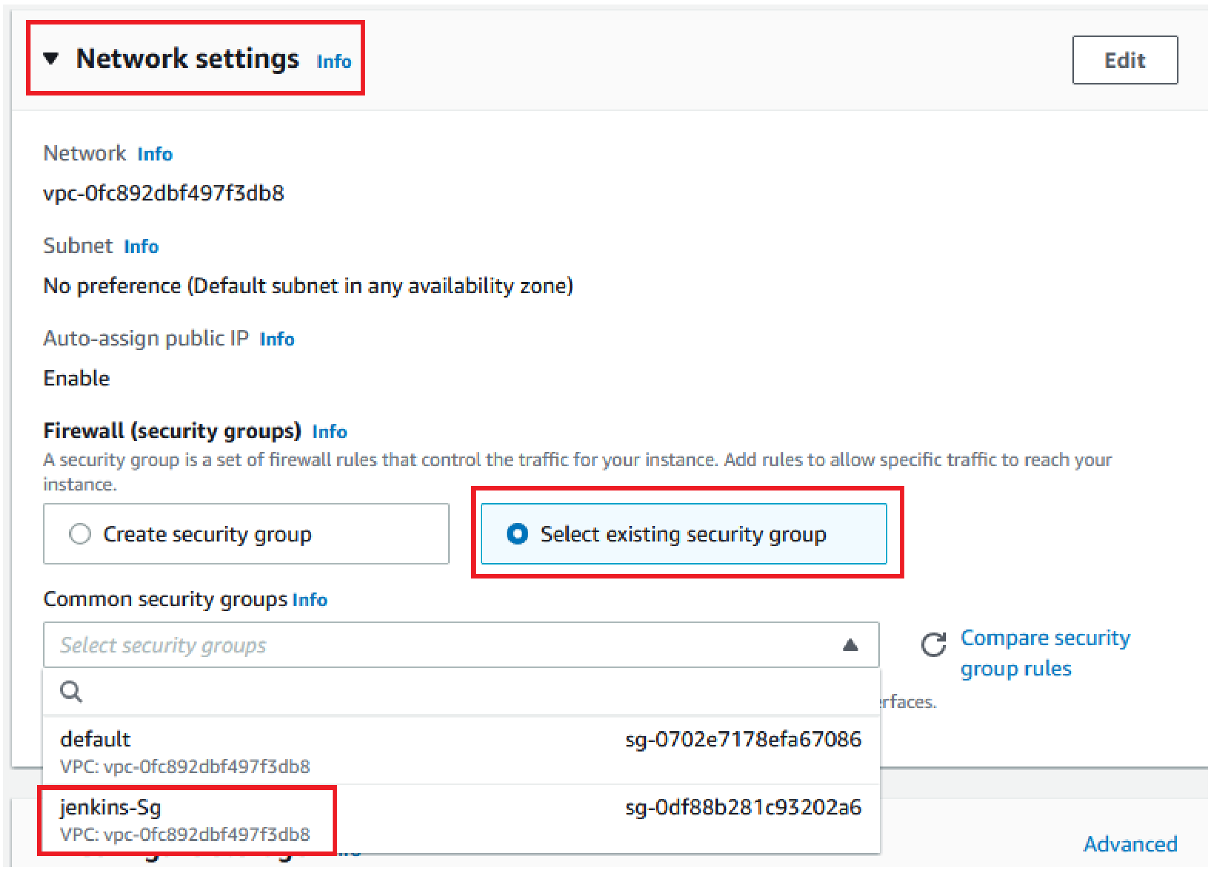
Key pair (login) Info

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - *required*

jenkins-new

3. Now, in the **Network settings** section select the option **Select existing security group** and then select the Security Group that we had created in the first step i.e **Jenkins-Sg**



Network settings Info

Network: vpc-0fc892dbf497f3db8

Subnet: No preference (Default subnet in any availability zone)

Auto-assign public IP: Enable

Firewall (security groups)

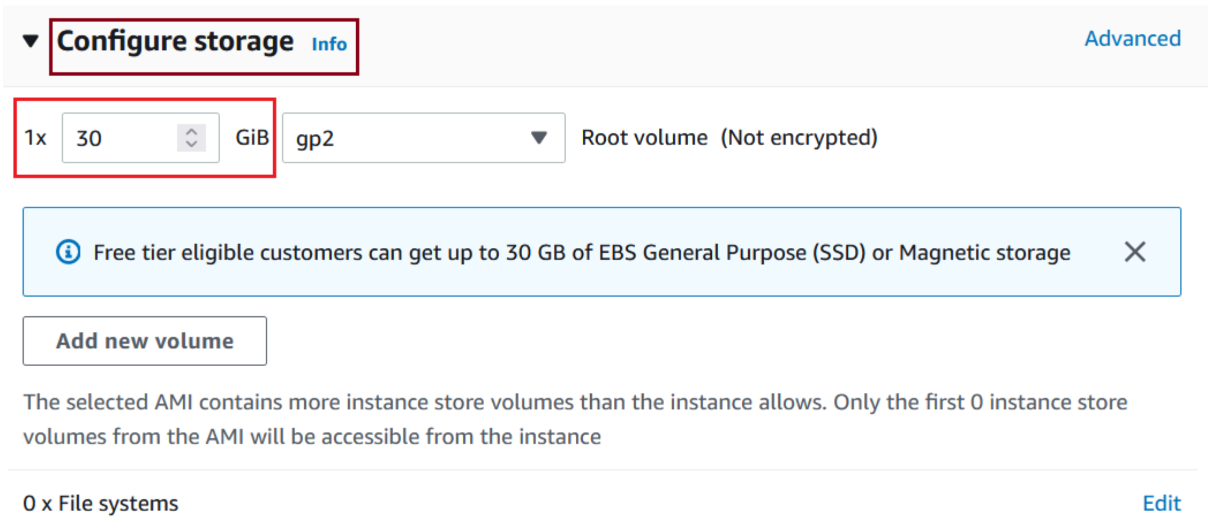
A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

Create security group [✓ Select existing security group]

Common security groups

default VPC: vpc-0fc892dbf497f3db8	sg-0702e7178efa67086
jenkins-Sg VPC: vpc-0fc892dbf497f3db8	sg-0df88b281c93202a6

4. After that in the section **Configure storage** put the **30Gb** as Storage.



Configure storage Info

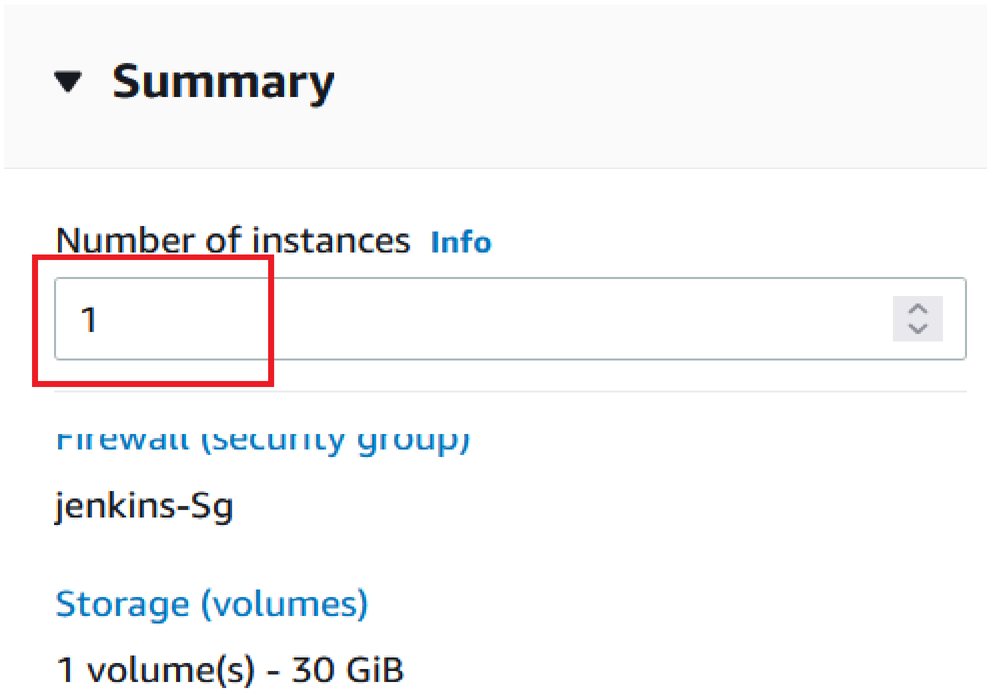
1x 30 GiB gp2 Root volume (Not encrypted)

Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage

The selected AMI contains more instance store volumes than the instance allows. Only the first 0 instance store volumes from the AMI will be accessible from the instance

0 x File systems

5. Put the No. of Instances as **1**



▼ Summary

Number of instances Info

1

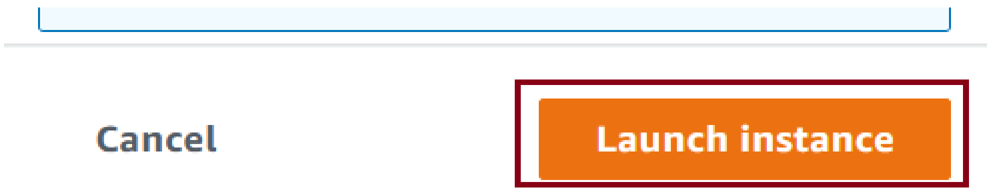
Firewall (security group)

jenkins-Sg

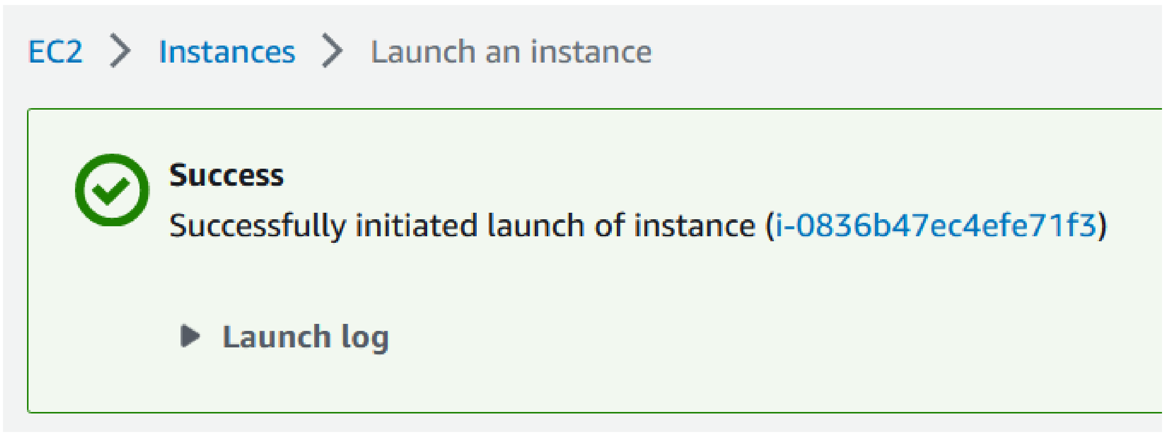
Storage (volumes)

1 volume(s) - 30 GiB

6. Now, click on **Launch Instance** in order to create an EC2 instance.



7. **Output:**



EC2 > Instances > Launch an instance

Success
Successfully initiated launch of instance (i-0836b47ec4efe71f3)

▶ Launch log

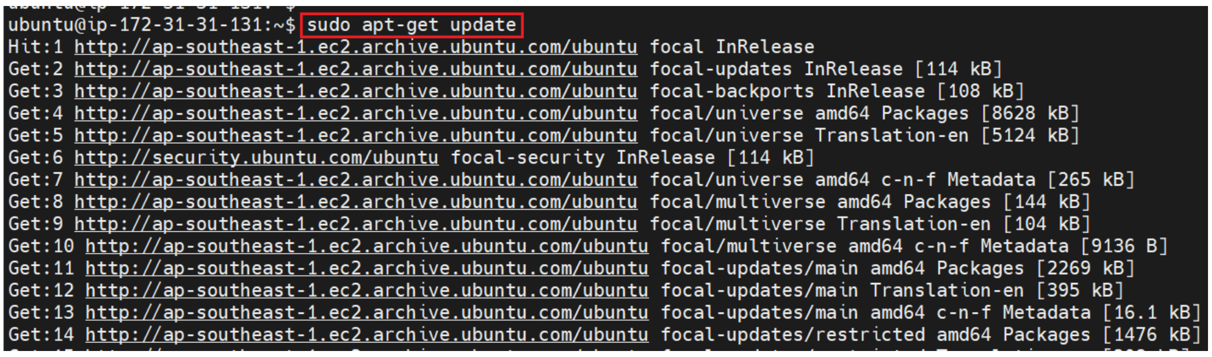
Installing & Configuring Docker On Dev Server

- Now, as we'll be deploying Image to the dev server and running it as a container, then we need Docker installed & configured on that server otherwise we'll get an error.

- Therefore, we'll install & configure docker on the dev server.

Update the system

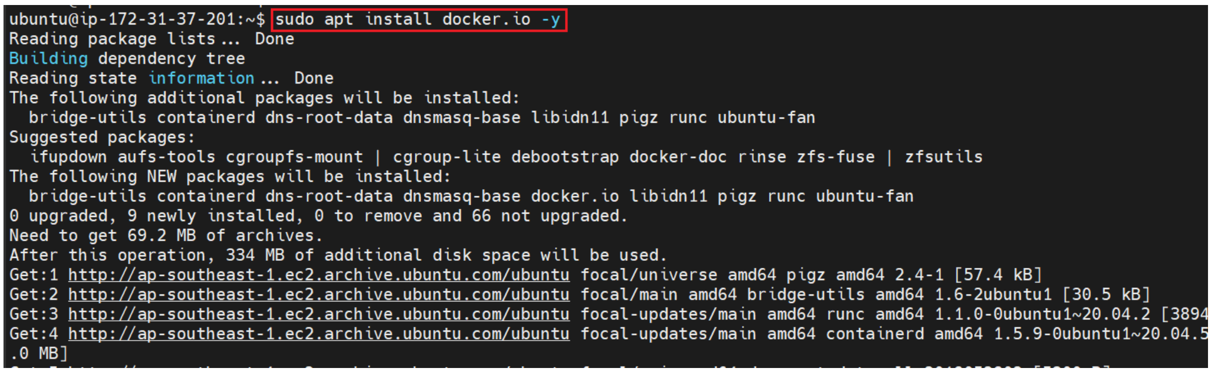
sudo apt-get update



```
ubuntu@ip-172-31-31-131:~$ sudo apt-get update
Hit:1 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu focal InRelease
Get:2 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu focal-updates InRelease [114 kB]
Get:3 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu focal-backports InRelease [108 kB]
Get:4 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu focal/universe amd64 Packages [8628 kB]
Get:5 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu focal/universe Translation-en [5124 kB]
Get:6 http://security.ubuntu.com/ubuntu focal-security InRelease [114 kB]
Get:7 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu focal/universe amd64 c-n-f Metadata [265 kB]
Get:8 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu focal/multiverse amd64 Packages [144 kB]
Get:9 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu focal/multiverse Translation-en [104 kB]
Get:10 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu focal/multiverse amd64 c-n-f Metadata [9136 B]
Get:11 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu focal-updates/main amd64 Packages [2269 kB]
Get:12 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu focal-updates/main Translation-en [395 kB]
Get:13 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu focal-updates/main amd64 c-n-f Metadata [16.1 kB]
Get:14 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu focal-updates/restricted amd64 Packages [1476 kB]
```

Install Docker

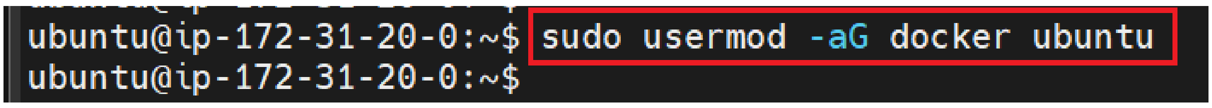
sudo apt install docker.io -y



```
ubuntu@ip-172-31-37-201:~$ sudo apt install docker.io -y
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base libidn11 pigz runc ubuntu-fan
Suggested packages:
  ifupdown aufs-tools cgroupfs-mount | cgroup-lite debootstrap docker-doc rinse zfs-fuse | zfsutils
The following NEW packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base docker.io libidn11 pigz runc ubuntu-fan
0 upgraded, 9 newly installed, 0 to remove and 66 not upgraded.
Need to get 69.2 MB of archives.
After this operation, 334 MB of additional disk space will be used.
Get:1 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu focal/universe amd64 pigz amd64 2.4-1 [57.4 kB]
Get:2 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu focal/main amd64 bridge-utils amd64 1.6-2ubuntu1 [30.5 kB]
Get:3 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu focal-updates/main amd64 runc amd64 1.1.0-0ubuntu1~20.04.2 [3894
Get:4 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu focal-updates/main amd64 containerd amd64 1.5.9-0ubuntu1~20.04.5
.0 MB]
```

Add Ubuntu User In Docker Group (Dev Server)

sudo usermod -aG docker ubuntu

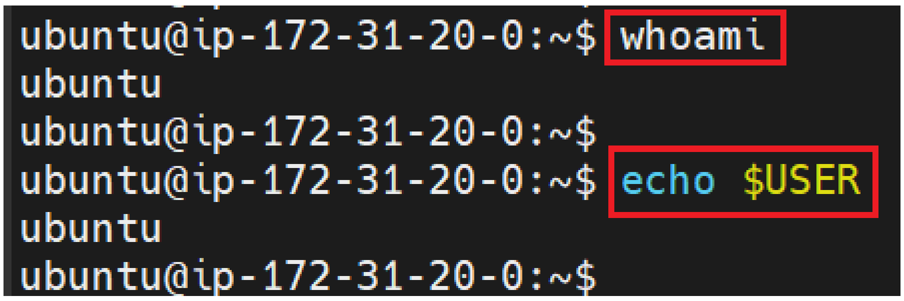


```
ubuntu@ip-172-31-20-0:~$ sudo usermod -aG docker ubuntu
ubuntu@ip-172-31-20-0:~$
```

Note: In case of Amazon Linux your default username will be different. It'll be **ec2-user**.

For Linux (Ubuntu) the default username will be **ubuntu**.

If you are not sure about your username then check it using the command **whoami** or **echo $USER**



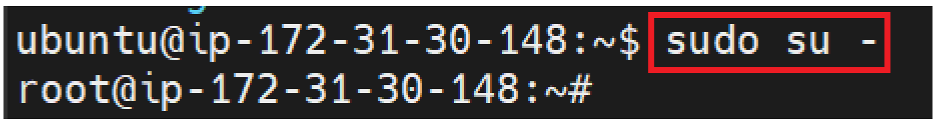
```
ubuntu@ip-172-31-20-0:~$ whoami
ubuntu
ubuntu@ip-172-31-20-0:~$
ubuntu@ip-172-31-20-0:~$ echo $USER
ubuntu
ubuntu@ip-172-31-20-0:~$
```

Add Password For User (ubuntu)

- Now, we'll add/change (for the server whose password is not set) because in the next step while refreshing the terminal, it'll ask for the password.

- Firstly we'll move to the root user

sudo su -



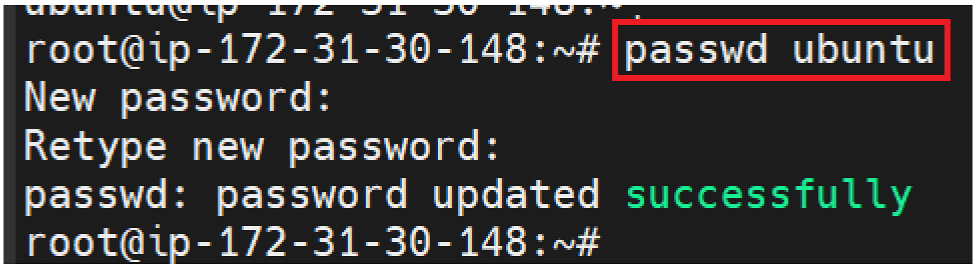
```
ubuntu@ip-172-31-30-148:~$ sudo su -
root@ip-172-31-30-148:~#
```

Now, we'll add the password using the command

passwd <username>

In our case, it'll be

passwd ubuntu

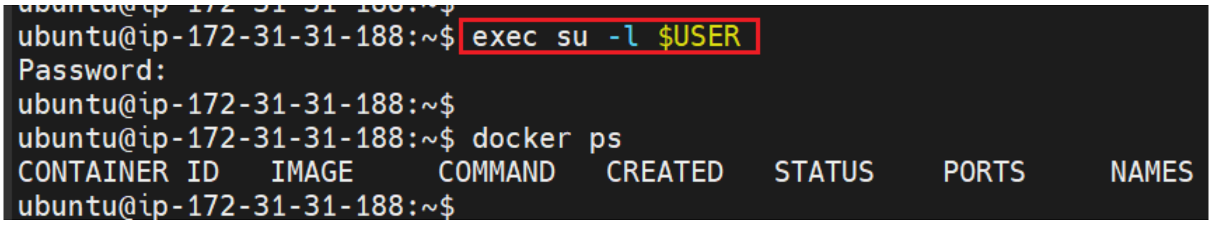


```
root@ip-172-31-30-148:~# passwd ubuntu
New password:
Retype new password:
passwd: password updated successfully
root@ip-172-31-30-148:~#
```

- The changes (Added Ubuntu in the Docker Group) will not reflect in the current terminal. So, we'll refresh the terminal using the command:

exec su -l $USER

- **exec**: It'll start a new process. So, our shell will get refreshed.
- **USER**: Current User "ubuntu".



```
ubuntu@ip-172-31-31-188:~$ exec su -l $USER
Password:
ubuntu@ip-172-31-31-188:~$
ubuntu@ip-172-31-31-188:~$ docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
ubuntu@ip-172-31-31-188:~$
```

Connect To Dev Server Using CI/CD Pipeline

- As, now the dev server is ready for the deployment of Docker Application.

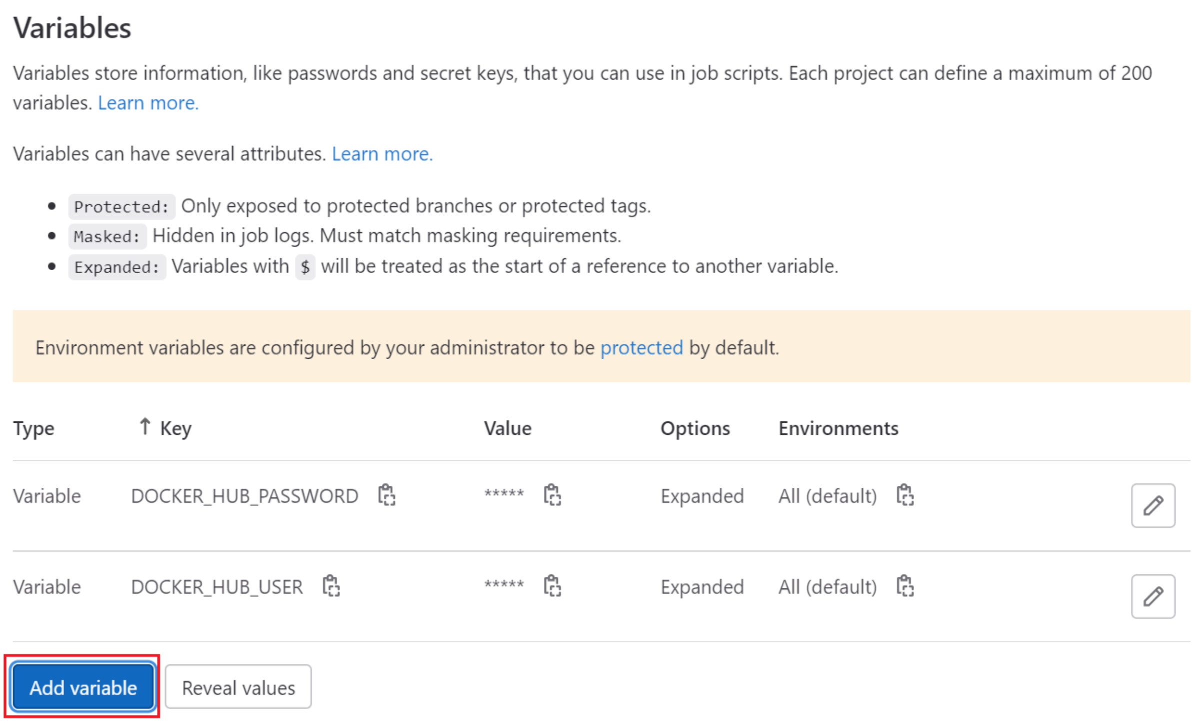
- Now, we'll be connecting to the Dev server from Gitlab using Gitlab Pipeline.

- We'll ssh from Gitlab to the dev server just the same way as we use to connect to the ec2 instance using terminal.

Storing Server's Credentials In Gitlab

- In order to ssh from Gitlab to the dev server we'll store the credentials of the dev-server in Gitlab and use those credentials (Pem Key) as a variable.

Project > Settings > CI/CD > Variables >Expand > Add Variable



Variables

Variables store information, like passwords and secret keys, that you can use in job scripts. Each project can define a maximum of 200 variables. Learn more.

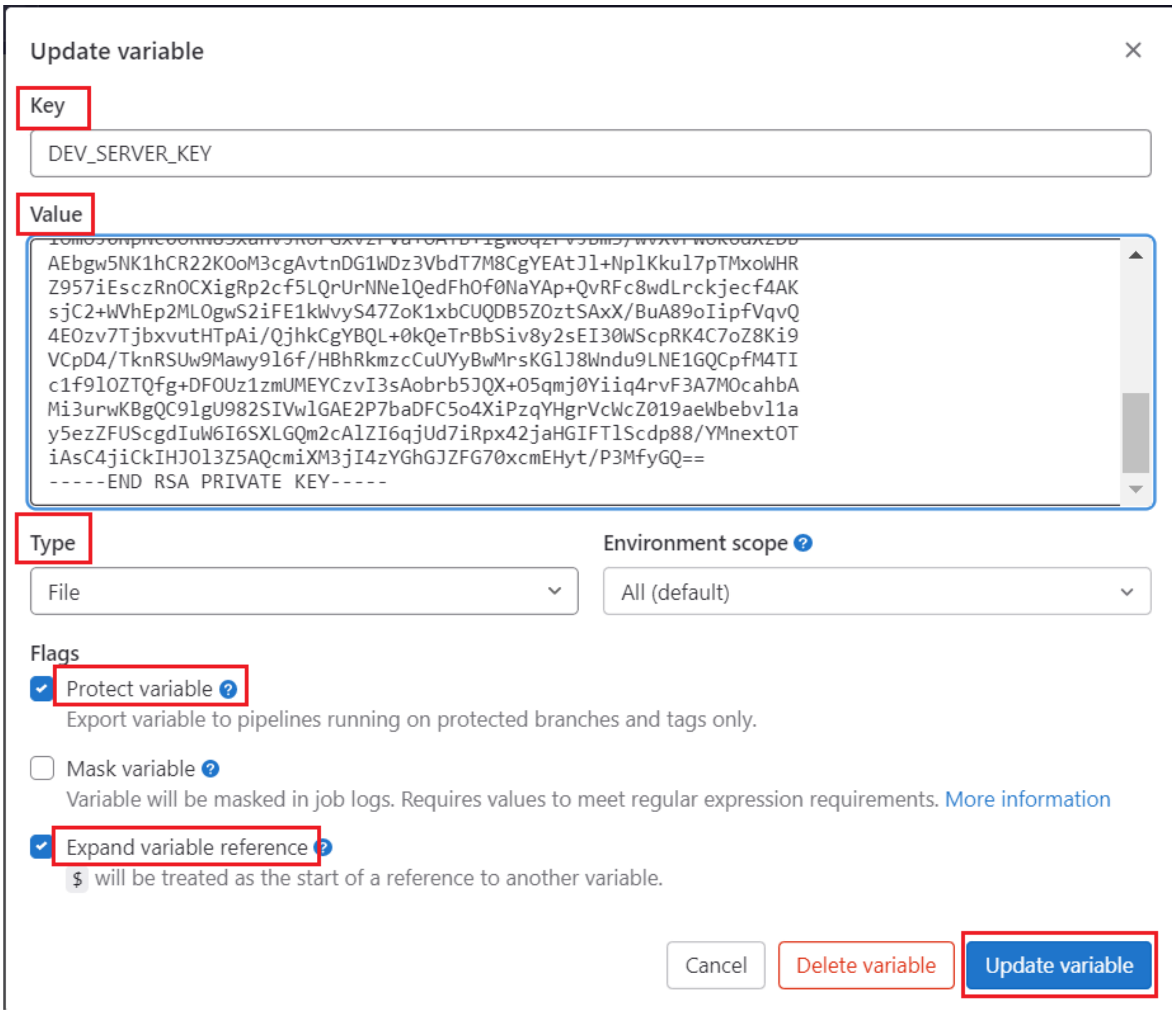
Variables can have several attributes. Learn more.

- `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.

Environment variables are configured by your administrator to be protected by default.

Type	↑ Key	Value	Options	Environments
Variable	DOCKER_HUB_PASSWORD	*****	Expanded	All (default)
Variable	DOCKER_HUB_USER	*****	Expanded	All (default)

Add variable > **Key** > DEV_SERVER_KEY > **Value** > Hashed Value inside .pem file used at the time create dev-server > **Type** > File > Check Protect variable > Add Variable


Update variable

Key

DEV_SERVER_KEY

Value

```
AEbgw5NK1hCR22KOoM3cgAvtnDG1WDz3VbdT7M8CgYEAtJl+NplKkul7pTMxoWHR
Z957iEsczRnOCXigRp2cf5LQrUrNNelQedFhOf0NaYAp+QvRFc8wdLrckjecf4AK
sjC2+WVhEp2MLOgwS2iFE1kWvyS47ZoK1xbCUQDB5ZOztSAxX/BuA89oIipfVqvQ
4EOzv7TjbxvutHTpAi/QjhkCgYBQL+0kQeTrBbSiv8y2sEI30WScpRK4C7oZ8Ki9
VCpD4/TknRSUw9Mawy9l6f/HBhRkmzcCuUYyBwMrsKGlJ8Wndu9LNE1GQCpfM4TI
c1f9lOZTQfg+DFOUz1zmUMEYCzvI3sAobrb5JQX+O5qmj0Yiiq4rvF3A7MOcahbA
Mi3urwKBgQC9lgU982SIVwlGAE2P7baDFC5o4XiPzqYHgrVcWcZ019aeWbebvl1a
y5ezZFUScgdIuW6I6SXLGQm2cAlZI6qjUd7iRpx42jaHGIFTlScdp88/YMnextOT
iAsC4jiCkIHJOl3Z5AQcmiXM3jI4zYGhGJZFG70xcmEHyt/P3MfyGQ==
-----END RSA PRIVATE KEY-----
```

Type: File

Environment scope: All (default)

Flags

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

[ ] Mask variable
Variable will be masked in job logs. Requires values to meet regular expression requirements. More information

[✓] Expand variable reference
`$` will be treated as the start of a reference to another variable.

- **Protect Variable**: It is only exposed to the protected branches. Means, the variables are only available when the pipeline runs on protected branches or protected tags, for example default "**main**" branches and not on the other feature branches.

- **Mask Variable**: It should be checked while using any secret data. Variables containing secrets should always be masked. Example private key.

  Using the Mask Variable we can avoid the risk of exposing the value of the variable. Example if some-one trying to use the

command **echo $VARIABLE_NAME** ; As using this command we can check the value stored inside that particular variable.
But if we use the Masked variable, then the output for the command **echo $VARIABLE_NAME** will be the Masked Value of the value stored inside the variable.

Note: We can’t use the option **Masked Variable** in order to mask the Private key of the EC2 Instance.

We can only mask any secret value example password or token.

Script To SSH In Dev-Server

- Name of the Job

```
deploy_to_dev:
```

- Stage

```
deploy_to_dev:
  stage: deploy
```

- Add the name of the stage in the stage section

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

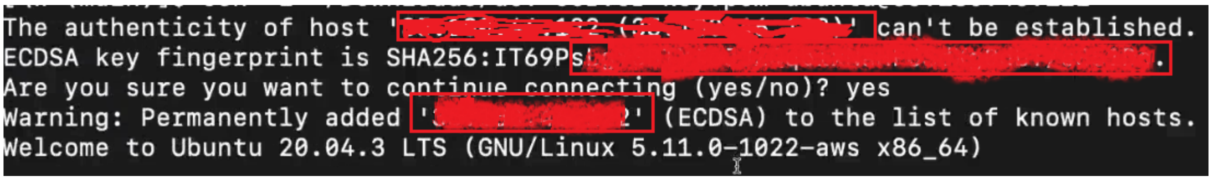
- We'll run the job using Gitlab-runner with **Shell** Executor. Therefore, we'll use tags in order to identify that executor.

```
deploy_to_dev:
  stage: deploy
  tags:
    - linux
    - remote
    - ubuntu
```

- Script to ssh dev-server.

```
  script:
    ssh -o StrictHostKeyChecking=no -i $DEV_SERVER_KEY
$DEV_SERVER_USERNAME@$DEV_SERVER_IP
```

- **-o**: Disabling the strict Host Key checking



```
The authenticity of host '[redacted]' can't be established.
ECDSA key fingerprint is SHA256:IT69Ps[redacted].
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '[redacted]' (ECDSA) to the list of known hosts.
Welcome to Ubuntu 20.04.3 LTS (GNU/Linux 5.11.0-1022-aws x86_64)
```

- Whenever we connect to the server for the first time, we get the prompt that asks whether we want to confirm the authenticity of the host on our side. And for that user have to type “yes”.

- And as the pipeline will be running in a non-interactive mode, that’s why we’ll not be able to enter any value over there. That’s why we need to disable this option.

Note: In an automated pipeline we have to disable any interactive step.

- Now, as we took the variable to refer to the username & public ip address of the dev-server; Therefore we need to assign the authentic value to these variables (globally (at the top before any job)).

```
variables:
  IMAGE_NAME: $CI_REGISTRY_IMAGE/microservice/payment
  DOCKER_IMAGE_NAME: abhishekkishor1/online-shopping-project
  IMAGE_TAG: "1.0"
  DEV_SERVER_USERNAME: ubuntu
  DEV_SERVER_IP: 54.169.115.242
```

- As the value of the key will be open, therefore we need to give the 600 permissions. With that only the owner of the file has full read and write access to it.

- So, we’ll put that thing in the **before_script** section under the “**deploy_to_dev**” job.

```
- chmod 600 $DEV_SERVER_KEY
```

- Now, we’ll assign the value

```
ssh -o StrictHostKeyChecking=no -i $DEV_SERVER_KEY
$DEV_SERVER_USERNAME@$DEV_SERVER_IP
```

To a variable “**DEV_LOGIN**”.

```
DEV_LOGIN: ssh -o StrictHostKeyChecking=no -i $DEV_SERVER_KEY
$DEV_SERVER_USERNAME@$DEV_SERVER_IP
```

Login To Docker In Dev Server

- Now, we'll login & pull the docker image from Docker Hub.
- Therefore, we'll use the following script

Login To DockerHub

```
- $DEV_LOGIN "docker login -u $DOCKER_HUB_USER -p $DOCKER_HUB_PASSWORD"
```

Note: We'll use the variable "$DEV_LOGIN" in order to execute the command in the dev-server.

Stopping Docker Container

```
- $DEV_LOGIN "docker stop $CONTAINER_NAME || true"
```

- Stop the docker container with the same name if it's running.
- **|| true** → If any container will not be running with the same name then it'll throw an error and once it throws an error then at that point only our project will exit.
  That's why we'll use the option "true" so that even if the output is not throwing any correct output then at least it'll become true or gets ignored.
  Otherwise if any container running with the same name, then it'll get stopped.

Removing Docker Container

```
- $DEV_LOGIN "docker rm $CONTAINER_NAME || true"
```

Removing Docker Images

```
$DEV_LOGIN "docker rmi $(docker images -a -q) || true"
```

- In order to reduce the load or space on the server, we can delete the images as well. As we'll be pulling a new image of the particular project every time then the previous image will not be helpful.

  It's totally on you, if you want you can include this command.

Pull Docker Image

```
- $DEV_LOGIN "docker pull $DOCKER_IMAGE_NAME:$IMAGE_TAG"
```

- We've already defined the value to the variables

$DOCKER_IMAGE_NAME: abhishekkishor1/online-shopping-project
$IMAGE_TAG: "1.0"

- Listing Docker Images in the **dev-server**

```
- $DEV_LOGIN "docker images"
```

Running Docker Image

- Now, we'll run the image that we have pulled in Dev Server.

As, the command to run a docker container is:

docker run -d -p <Docker Exposed Port no.:user's Port > --name <Container's name given by user> <Image name>

- Therefore, in our case it'll be

```
- $DEV_LOGIN "docker run -d -p 3000:3000 --name online-shoping-app
$DOCKER_IMAGE_NAME:$IMAGE_TAG"
```

-d: Detached Mode
-p: Port
online-shoping-app: Container's Name
DOCKER_IMAGE_NAME:$IMAGE_TAG: Image name (Variable) along with tag that we had pulled to dev-server.

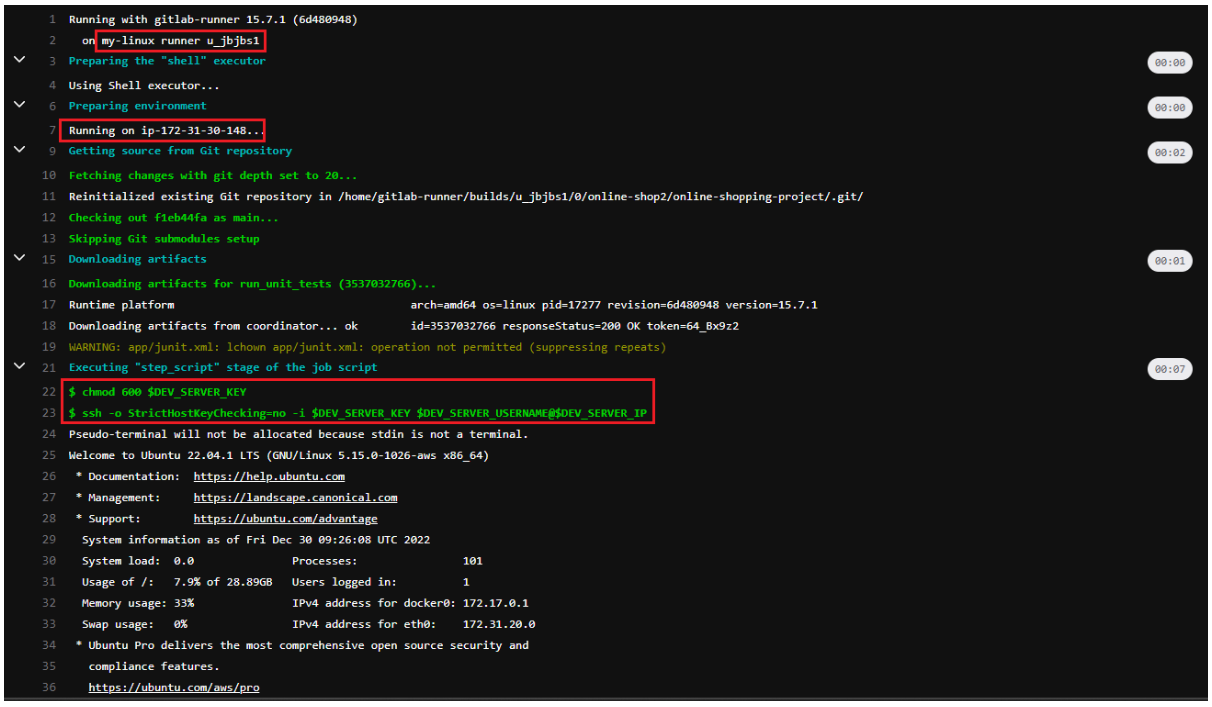
Full Script

```
workflow:
  rules:
    - if: $CI_COMMIT_BRANCH != "main" && $CI_PIPELINE_SOURCE !=
"merge_request_event"
      when: never
    - when: always

variables:
  IMAGE_NAME: $CI_REGISTRY_IMAGE/microservice/payment
  DOCKER_IMAGE_NAME: abhishekkishor1/online-shopping-project
  IMAGE_TAG: "1.0"
  DEV_SERVER_USERNAME: ubuntu
  DEV_SERVER_IP: 13.213.38.238
  DEV_LOGIN: ssh -o StrictHostKeyChecking=no -i $DEV_SERVER_KEY
$DEV_SERVER_USERNAME@$DEV_SERVER_IP
  CONTAINER_NAME: online-shoping-app

stages:
  - test
  - build
  - deploy

run_unit_tests:
    image: node:17-alpine3.14
    stage: test
    tags:
      - docker
      - linux
      - remote

    before_script:
      - cd app
      - npm install

    script:
      - npm test

    artifacts:
      when: always
      paths:
        - app/junit.xml
      reports:
        junit: app/junit.xml


build_image:
  stage: build
  tags:
    - linux
    - remote
    - ubuntu

  script:
    - docker build -t $IMAGE_NAME:$IMAGE_TAG .


push_image:
  stage: build
  needs:
    - build_image
  tags:
    - linux
    - remote
    - ubuntu
  before_script:
    - echo "Docker registry url is $CI_REGISTRY"
    - echo "Docker registry username is $CI_REGISTRY_USER"
    - echo "Docker image repo is $CI_REGISTRY_IMAGE"
    - docker login -u $CI_REGISTRY_USER -p $CI_REGISTRY_PASSWORD
$CI_REGISTRY
  script:
    - docker push $IMAGE_NAME:$IMAGE_TAG

# build_docker_image:
#   stage: build
#   tags:
#     - linux
#     - remote
#     - ubuntu

#   script:
#     - docker build -t $DOCKER_IMAGE_NAME:$IMAGE_TAG .
```

```yaml
change_image_name:
  stage: build
  tags:
    - linux
    - remote
    - ubuntu

  script:
    - docker image tag $IMAGE_NAME:$IMAGE_TAG
$DOCKER_IMAGE_NAME:$IMAGE_TAG


push_to_dockerhub:
  stage: build
  needs:
    # - build_docker_image
    - change_image_name
  tags:
    - linux
    - remote
    - ubuntu
  before_script:
    - docker login -u $DOCKER_HUB_USER -p $DOCKER_HUB_PASSWORD
  script:
    - docker push $DOCKER_IMAGE_NAME:$IMAGE_TAG

deploy_to_dev:
  stage: deploy
  tags:
    - linux
    - remote
    - ubuntu
  before_script:
    - chmod 600 $DEV_SERVER_KEY

  script:
    - ssh -o StrictHostKeyChecking=no -i $DEV_SERVER_KEY
$DEV_SERVER_USERNAME@$DEV_SERVER_IP
    - $DEV_LOGIN "docker login -u $DOCKER_HUB_USER -p
$DOCKER_HUB_PASSWORD"
    - $DEV_LOGIN "docker stop $CONTAINER_NAME || true"
    - $DEV_LOGIN "docker rm $CONTAINER_NAME || true"
    - $DEV_LOGIN "docker rmi $(docker images -a -q) || true"
    - $DEV_LOGIN "docker pull $DOCKER_IMAGE_NAME:$IMAGE_TAG"
    - $DEV_LOGIN "docker images"
    - $DEV_LOGIN "docker run -d -p 3000:3000 --name online-shoping-app
$DOCKER_IMAGE_NAME:$IMAGE_TAG"
```

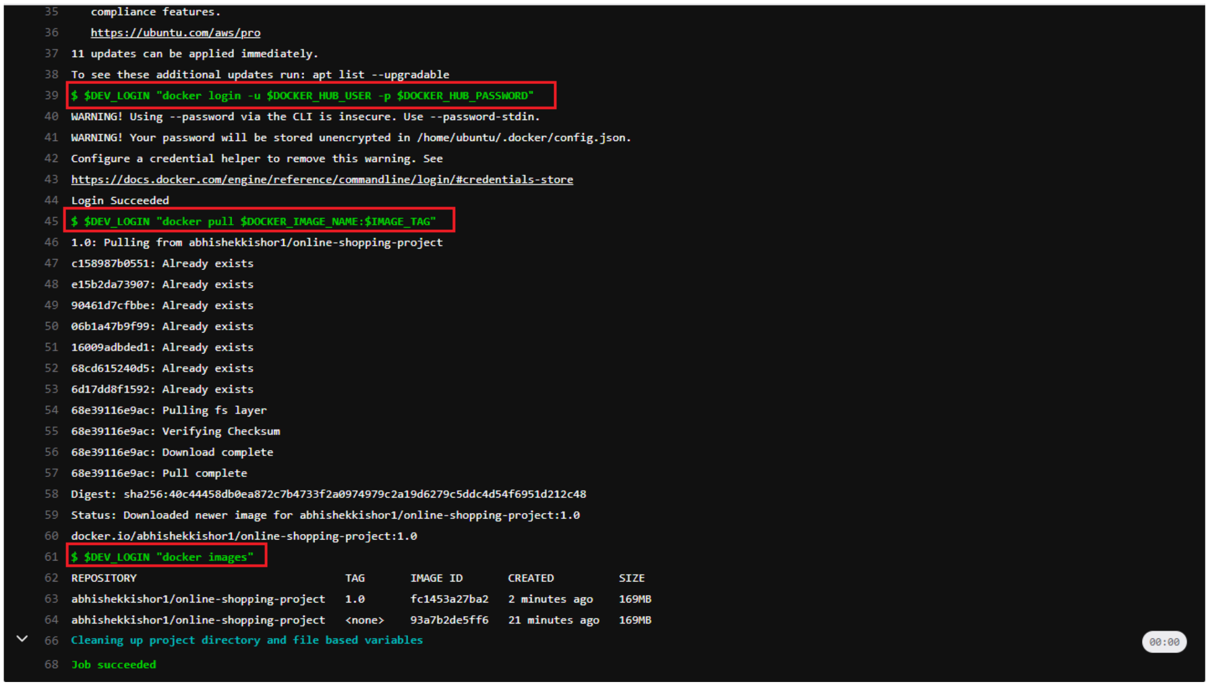
Output:

Login To Dev-Server:



```
Running with gitlab-runner 15.7.1 (6d480948)
  on my-linux runner u_jbjbs1
Preparing the "shell" executor
Using Shell executor...
Preparing environment
Running on ip-172-31-30-148...
Getting source from Git repository
Fetching changes with git depth set to 20...
Reinitialized existing Git repository in /home/gitlab-runner/builds/u_jbjbs1/0/online-shop2/online-shopping-project/.git/
Checking out f1eb44fa as main...
Skipping Git submodules setup
Downloading artifacts
Downloading artifacts for run_unit_tests (3537032766)...
Runtime platform                                    arch=amd64 os=linux pid=17277 revision=6d480948 version=15.7.1
Downloading artifacts from coordinator... ok        id=3537032766 responseStatus=200 OK token=64_Bx9z2
WARNING: app/junit.xml: lchown app/junit.xml: operation not permitted (suppressing repeats)
Executing "step_script" stage of the job script
$ chmod 600 $DEV_SERVER_KEY
$ ssh -o StrictHostKeyChecking=no -i $DEV_SERVER_KEY $DEV_SERVER_USERNAME@$DEV_SERVER_IP
Pseudo-terminal will not be allocated because stdin is not a terminal.
Welcome to Ubuntu 22.04.1 LTS (GNU/Linux 5.15.0-1026-aws x86_64)
 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage
  System information as of Fri Dec 30 09:26:08 UTC 2022
  System load:  0.0               Processes:                101
  Usage of /:   7.9% of 28.89GB   Users logged in:          1
  Memory usage: 33%               IPv4 address for docker0: 172.17.0.1
  Swap usage:   0%                IPv4 address for eth0:    172.31.20.0
 * Ubuntu Pro delivers the most comprehensive open source security and
   compliance features.
   https://ubuntu.com/aws/pro
```

Pulling Docker Image From Docker Hub:



```
   compliance features.
   https://ubuntu.com/aws/pro
11 updates can be applied immediately.
To see these additional updates run: apt list --upgradable
$ $DEV_LOGIN "docker login -u $DOCKER_HUB_USER -p $DOCKER_HUB_PASSWORD"
WARNING! Using --password via the CLI is insecure. Use --password-stdin.
WARNING! Your password will be stored unencrypted in /home/ubuntu/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store
Login Succeeded
$ $DEV_LOGIN "docker pull $DOCKER_IMAGE_NAME:$IMAGE_TAG"
1.0: Pulling from abhishekkishor1/online-shopping-project
c158987b0551: Already exists
e15b2da73907: Already exists
90461d7cfbbe: Already exists
06b1a47b9f99: Already exists
16009adbded1: Already exists
68cd615240d5: Already exists
6d17dd8f1592: Already exists
68e39116e9ac: Pulling fs layer
68e39116e9ac: Verifying Checksum
68e39116e9ac: Download complete
68e39116e9ac: Pull complete
Digest: sha256:40c44458db0ea872c7b4733f2a0974979c2a19d6279c5ddc4d54f6951d212c48
Status: Downloaded newer image for abhishekkishor1/online-shopping-project:1.0
docker.io/abhishekkishor1/online-shopping-project:1.0
$ $DEV_LOGIN "docker images"
REPOSITORY                              TAG       IMAGE ID       CREATED          SIZE
abhishekkishor1/online-shopping-project   1.0       fc1453a27ba2   2 minutes ago    169MB
abhishekkishor1/online-shopping-project   <none>    93a7b2de5ff6   21 minutes ago   169MB
Cleaning up project directory and file based variables
Job succeeded
```

Getting Your Website Live

- Now, we'll check on our local Machine if the container that we are running on the dev-server throws any output or not.

- For this, we'll copy & paste the **public ip** of the dev-server along with the port no. that we had given while running the docker container in the dev-server. In our case, it'll be Port no. **3000**

- In my case, it'll be:

http://13.213.38.238:3000/