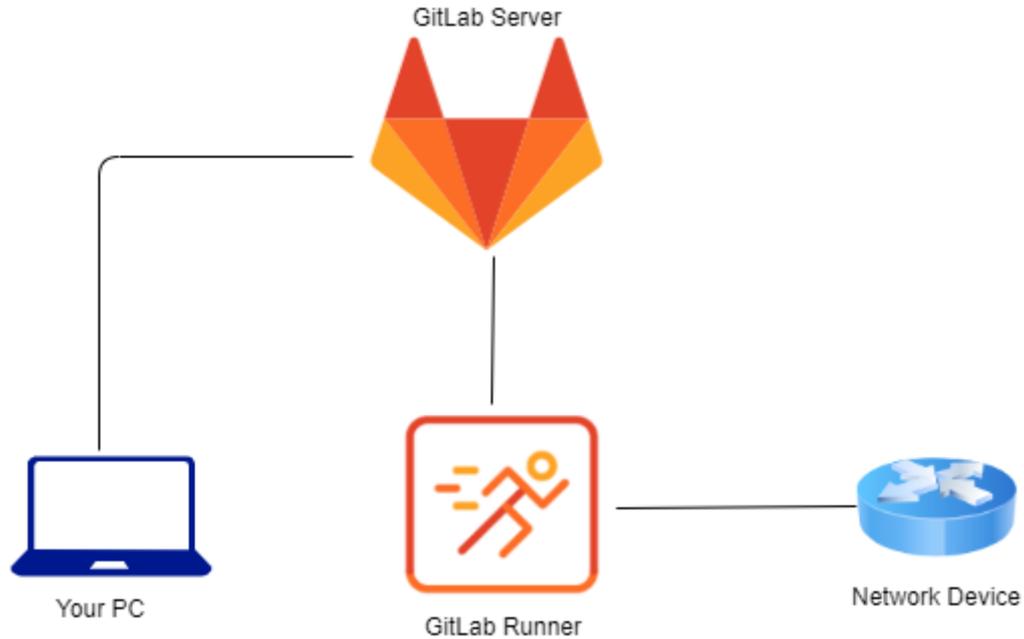


Lab - CI/CD cơ bản (Hello World)

1. Mô hình LAB



Hình 1: Mô hình LAB sử dụng để thực thi.

2. Yêu cầu

Thực hiện xây dựng luồng CI/CD cơ bản để thực hiện deploy thành công in ra “Hello World” với gitlab server và Gitlab Runner.

3. Các bước thực hiện

a) Thực hiện cài đặt Gitlab Server và Gitlab Runner theo file hướng dẫn hoặc có thể làm theo các bước sau đối với máy ảo Centos

- Các bước cài Gitlab Server

```
sudo yum -y update
```

```
sudo yum -y install curl vim policycoreutils python3-policycoreutils
```

```
sudo yum -y install postfix
```

```
sudo systemctl enable postfix && sudo systemctl start postfix
```

```
curl -s https://packages.gitlab.com/install/repositories/gitlab/gitlab-ce/script.rpm.sh | sudo bash
```

```
sudo yum install gitlab-ce
```

```
sudo vi /etc/gitlab/gitlab.rb
```

`external_url 'http://gitlab.example.com'` # Lưu ý: chỉnh sửa file thành địa chỉ card mạng để phù hợp bằng lệnh trên Centos: ifconfig

```
sudo gitlab-ctl reconfigure
```

```
sudo firewall-cmd --permanent --add-service={ssh,http,https} --permanent
```

```
sudo firewall-cmd --reload
```

Sau đó đăng nhập với username mặc định thường là root và password tại vị trí `:/etc/gitlab/initial_root_password` theo địa chỉ đã tạo card mạng ở trên.

- Các bước cài Gitlab Runner

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

```
sudo yum install gitlab-runner -y
```

- Sau khi cài đặt xong thực hiện các bước để register Gitlab Runner với Gitlab Server

b) Tạo project mới trên Gitlab Server để register Gitlab Runner

- Tạo project trên Gitlab Server

Đăng nhập vào Gitlab Server sau khi có username và password chọn “New project”:

Projects

New project

Your projects	5	Starred projects	0	Explore projects	Explore topics	Filter by name...	Name	▼
All	Personal							
C	Administrator / CICDBASIC	Maintainer	🟢	★ 0	👤 0	🔗 0	🗑️ 0	Updated 17 hours ago
H	Administrator / Hello_World	Maintainer	🟢	★ 0	👤 0	🔗 0	🗑️ 0	Updated 1 hour ago
L	Administrator / LABCICDEMO	Maintainer	🟢	★ 0	👤 0	🔗 0	🗑️ 0	Updated 1 hour ago
M	GitLab Instance / Monitoring	Owner		★ 0	👤 0	🔗 0	🗑️ 0	Updated 1 day ago <small>This project is automatically generated and helps monitor this GitLab instance. Learn more.</small>
S	Administrator / Shell_Version	Maintainer	🟢	★ 0	👤 0	🔗 0	🗑️ 0	Updated 25 minutes ago

Sau đó chọn “Create blank project”



Create blank project

Create a blank project to house your files, plan your work, and collaborate on code, among other things.



Create from template

Create a project pre-populated with the necessary files to get you started quickly.



Import project

Migrate your data from an external source like GitHub, Bitbucket, or another instance of GitLab.

Sau đó điền tên cũng như Description (nếu có) và chọn chế độ “Private, Internal, Public cần thiết và nhấn “Create project” phía cuối trang.



Create blank project

Create a blank project to house your files, plan your work, and collaborate on code, among other things.

New project › Create blank project

Project name

Project URL

Project slug

Want to house several dependent projects under the same namespace? [Create a group.](#)

Project description (optional)

Project deployment target (optional)

Visibility Level [?](#)

- Private
Project access must be granted explicitly to each user. If this project is part of a group, access will be 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 Configuration

- Initialize repository with a README
Allows you to immediately clone this project's repository. Skip this if you plan to push up an existing repository.

- Cấu hình SSH-Key

+ Trên cửa sổ Terminal của Centos nhập các lệnh sau để tạo SSH-Key:

ssh-keygen -t rsa

+ Một cửa sổ hiện ra như bên dưới bao gồm chọn nơi lưu Key cũng như mật khẩu nếu không, có thể bỏ qua.

```

root@localhost:~
File Edit View Search Terminal Help
[root@localhost ~]# ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
/root/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:Ny9J2+XRypFYfJ6zrSKyjcwpwry+2i/nmrhwhaqWr+GA root@localhost.localdomain
The key's randomart image is:
+---[RSA 3072]-----+
|
|             .
|            o .
|            o =.
|   . . .   S + . =oo|
| +o       o * + ++|
|oE. . .   + o +..|
|+..o *+..o... .|
|+oo +BxB=o... .|
+-----[SHA256]-----+
[root@localhost ~]#

```

- Tiếp theo thêm Key vào ssh-agent

```
eval "$(ssh-agent -s)"
```

```
ssh-add ~/.ssh/id_rsa
```

```
cat ~/.ssh/id_rsa.pub
```

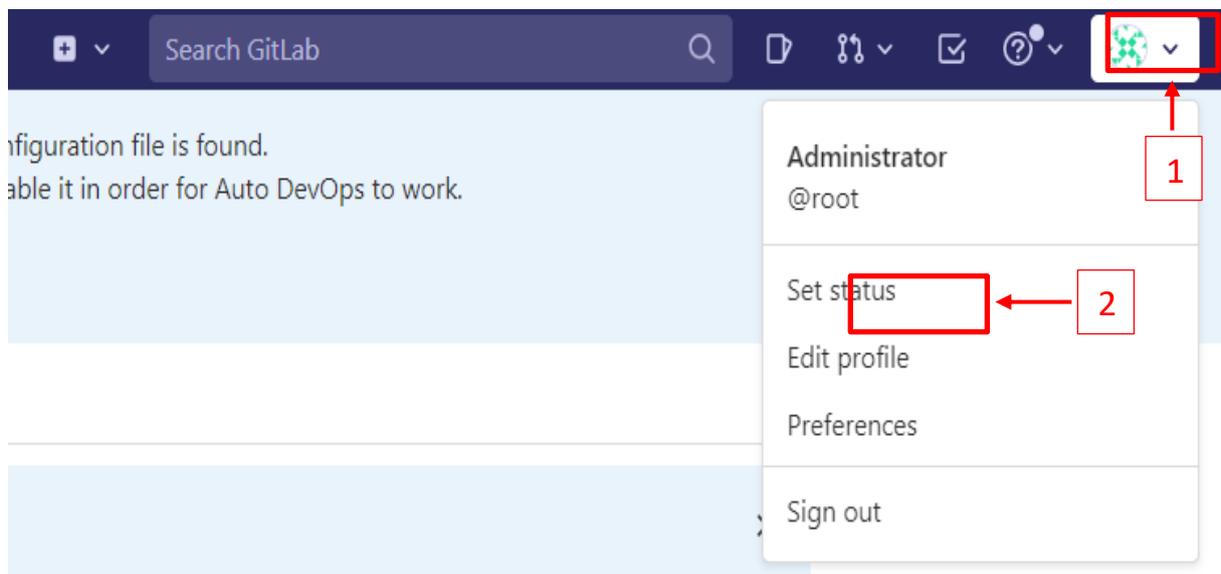
- Sau đó đoạn mã Key hiện ra và thực hiện copy:

```

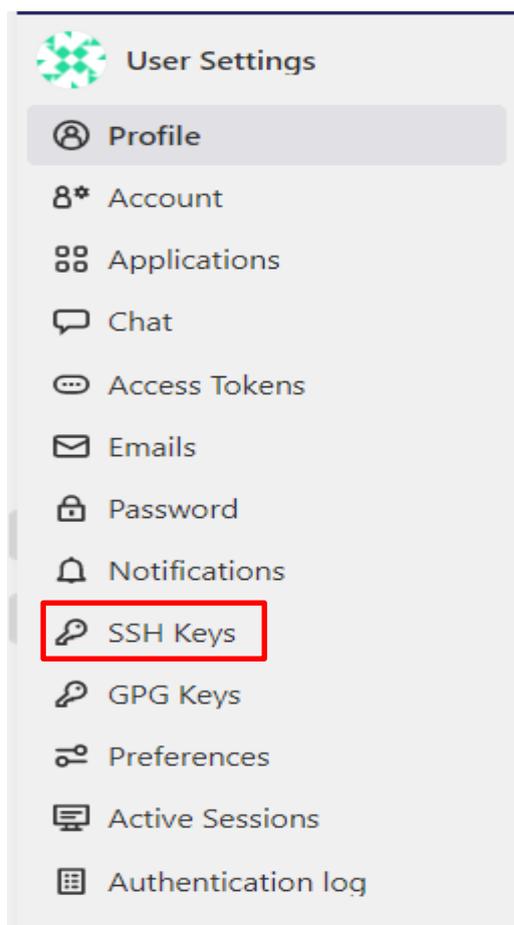
root@localhost:~
File Edit View Search Terminal Help
+-----[SHA256]-----+
[root@localhost ~]# eval "$(ssh-agent -s)"
Agent pid 39429
[root@localhost ~]# ssh-add ~/.ssh/id_rsa
Identity added: /root/.ssh/id_rsa (root@localhost.localdomain)
[root@localhost ~]# cat ~/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQGDm6CLX3Ettuq2B/Wx1RSfFE72FHbpeWH3qB9I0p2p0
uyCko6rtr2Us2LyYv0GyKIpx32qeDm47P43cvdMowrzxhS14b7IU8a3Zj+CE6GziYEWfgXE0peXKeP/
vPI22XGv+7f/3DZSBewf40o0DMTzHKWD8NPNMdY7ps6mX3ajSfTUqWuCK65oz3xvvpVC5ddk8LRJfmvYh
ylU22fCzp+sokw3CpPqgGQcKt+0rQfScUzDrR3PUXWIIKLoZBLrE0d+2yJjixpTLD5tEEoSky/hK01PL
Vw2hsEoLzMW/rIGLuA81dbyC1vbv6WnWfpxdUbKvjIsZoxmIltNd2FX+UFVuhKB/6mub6CwKEYV+dzWn
0gncYYCA3HsBp9PGZujw95Vs0Uf0q/GdHC2JcJRfeBlNXVG3xyk8MFA7YfNUoyC00pZCkJ12j5lhF0Uq
vvGSuNxYJlHkMYynPzVJn/7R6W+s5LGUzhHbzZZFYnmn0LZyZKj0ppGGgYW6ETeODNxt4m8= root@lo
calhost.localdomain
[root@localhost ~]#

```

- Tiếp theo mở gitlab Server để thêm SSH-Key. Tạo cửa sổ nhập vào biểu tượng hình bên góc phải và nhấn chọn **Edit profile**



- Sau đó nhấn chọn **SSH Keys** bên góc trái:



- Tiếp theo nhấn **Settings** và chọn vào **CI/CD** và mục bên phải ngay Runners chọn **Expand**

The screenshot shows the GitLab CI/CD settings page. On the left sidebar, the 'Settings' menu item is highlighted with a red box and labeled '1'. Below it, the 'CI/CD' sub-menu is also highlighted with a red box and labeled '2'. On the main content area, the 'Runners' section is visible, and its 'Expand' button is highlighted with a red box and labeled '3'. The page title is 'Administrator > Hello > CI/CD Settings'.

- Ta sẽ nhận được link **URL** và **Registration** để register cho Gitlab Runner:

Runners

Collapse

Runners are processes that pick up and execute CI/CD jobs for GitLab. [How do I configure runners?](#)

Register as many runners as you want. You can register runners as separate users, on separate servers, and on your local machine. Runners are either:

- **active** - Available to run jobs.
- **paused** - Not available to run jobs.

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:
`http://192.168.159.131/`

And this registration token:

`GR1348941WMr-fwr9nu2mjuJYj6aQ5`

Reset registration token

Show runner installation instructions

Shared runners

These runners are shared across this GitLab instance.

The same shared runner executes code from multiple projects, unless you configure autoscaling with [MaxBuilds](#) set to 1 (which it is on GitLab.com).

Enable shared runners for this project



This GitLab instance does not provide any shared runners yet. Instance administrators can register shared runners in the admin area.

Group runners

These runners are shared across projects in this group.

- Tiếp theo ta mở Gitlab Runner tại cửa sổ Terminal của Centos:

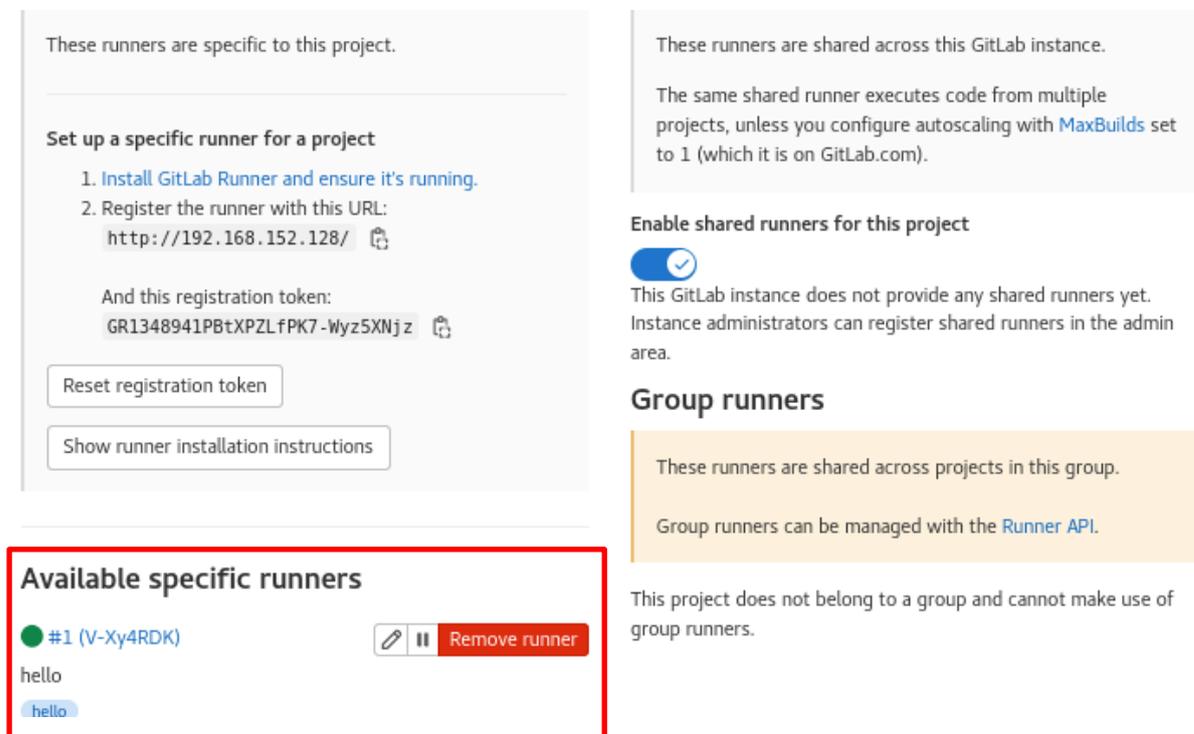
Nhập lệnh: **# sudo gitlab-runner register**

- Và nhập các URL và Token như hướng dẫn trên. Ngoài ra các thông số còn lại có thể khai báo khác nhưng ở đây Executor mình chọn là thực hiện với **docker** và có thể chọn **shell**

```
[root@localhost ~]# sudo gitlab-runner register
Runtime platform                                arch=amd64 os=linux pid=4196
6 revision=c6e7e194 version=14.8.2
Running in system-mode.

Enter the GitLab instance URL (for example, https://gitlab.com/):
http://192.168.159.131/
Enter the registration token:
GR1348941WMrfwr9nu2mjuJYj6aQS
Enter a description for the runner:
[localhost.localdomain]: hello
Enter tags for the runner (comma-separated):
hello
Enter optional maintenance note for the runner:
hello
Registering runner... succeeded                  runner=GR134894
Enter an executor: docker, parallels, docker+machine, docker-ssh+machine, virtua
lbox, kubernetes, custom, docker-ssh, shell, ssh:
docker
Enter the default Docker image (for example, ruby:2.7):
centos:8
Runner registered successfully. Feel free to start it, but if it's running alrea
dy the config should be automatically reloaded!
[root@localhost ~]#
```

- Sau đó mở gitlab server và reload lại một Gitlab Runner đã được tạo.

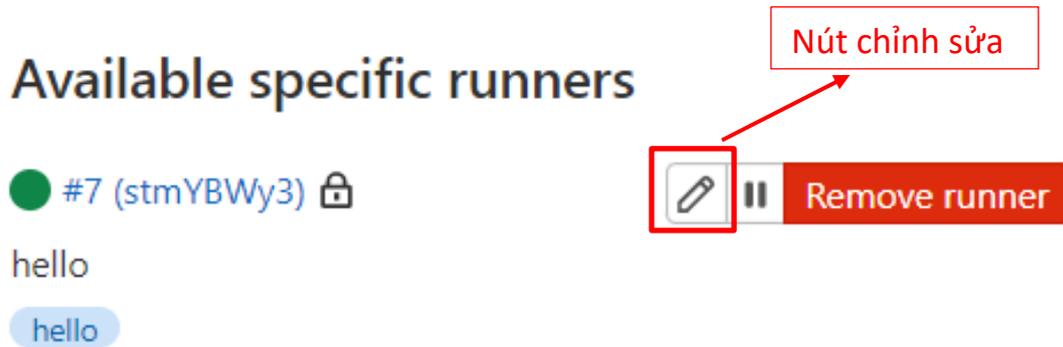


The screenshot shows the GitLab Runner configuration page. It is divided into several sections:

- These runners are specific to this project.** This section contains instructions on how to set up a specific runner for a project, including the URL and registration token.
- These runners are shared across this GitLab instance.** This section explains that shared runners execute code from multiple projects, unless autoscaling is configured.
- Enable shared runners for this project.** This section has a toggle switch that is currently turned on.
- Group runners.** This section explains that group runners are shared across projects in a group and can be managed with the Runner API.
- Available specific runners.** This section, highlighted with a red box, shows a list of runners. The first runner is named "#1 (V-Xy4RDK)" and has a description of "hello". There are buttons for "Remove runner" and "hello" next to it.

- Sau khi tạo xong sẽ xuất hiện Available specific runners như trong hình.

- Sau đó nhấn vào nút chỉnh sửa để thay đổi các thông tin cần thiết như hình bên dưới tích chọn **Indicates...** và bỏ chọn **When.....** và nhấn



Save changes

Active Paused runners don't accept new jobs

Protected This runner will only run on pipelines triggered on protected branches

Run untagged jobs Indicates whether this runner can pick jobs without tags

Lock to current projects When a runner is locked, it cannot be assigned to other projects

IP Address

Description

Maximum job timeout
Enter the number of seconds, or other human-readable input, like "1 hour". This timeout takes precedence over lower timeouts set for the project.

Tags
You can set up jobs to only use runners with specific tags. Separate tags with commas.

c) Thực hiện clone project đã tạo về trên máy Centos

- Đầu tiên nhấn vào project và nhấn vào mũi tên bên cạnh Clone sẽ hiển thị link để chúng ta clone về. Ở đây mình chọn là HTTP.

Administrator > Hello

Clone with SSH
git@192.168.159.131:root/hello.git

Clone with HTTP
http://192.168.159.131/root/hello.

Open in your IDE
Visual Studio Code (SSH)
Visual Studio Code (HTTPS)

Name	Last commit
README.md	Initial commit

- Tiếp theo, ta thực hiện clone project về bằng cách lệnh sau trên cửa sổ Terminal của Centos:

```
[root@localhost ~]# git clone http://192.168.159.131/root/hello.git
Cloning into 'hello'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 2.78 KiB | 2.78 MiB/s, done.
```

- Tiếp theo tạo file mới với tên là **.gitlab-ci.yml** trên máy Centos bằng các lệnh sau:
- Bằng cách trở đến thư mục bằng lệnh: **cd <tên_thư_mục>**
- Tiếp theo thêm file mới bằng cách: **vim .gitlab-ci.yml**

```
[root@localhost ~]# cd hello
[root@localhost hello]# vim .gitlab-ci.yml
```

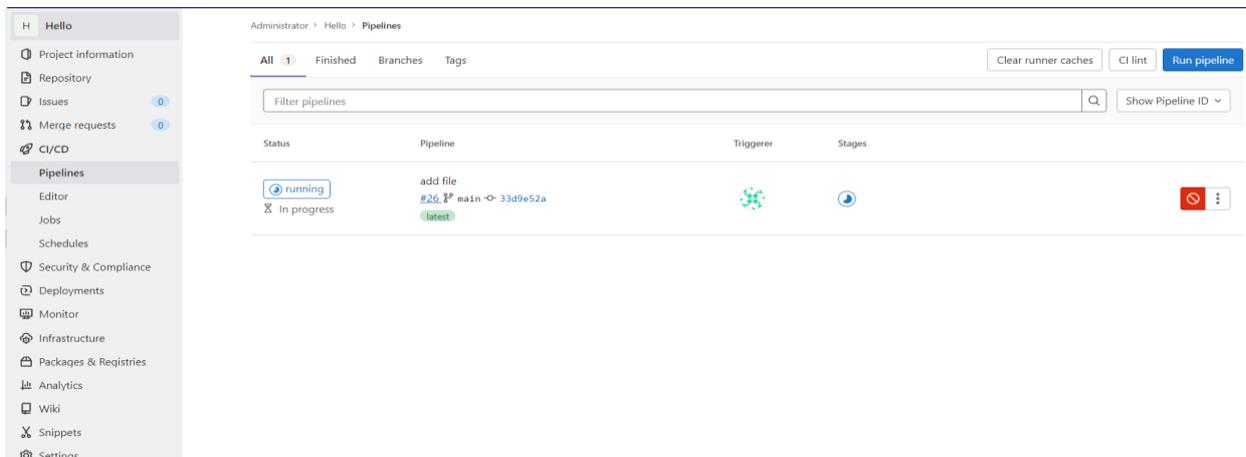
- Sau đó nhấn chữ "I" để điền câu lệnh như hình và nhấn ESC sau đó nhập **:wq** để lưu và thoát.


```
Enumerating objects: 4, done.  
Counting objects: 100% (4/4), done.  
Delta compression using up to 2 threads  
Compressing objects: 100% (2/2), done.  
Writing objects: 100% (3/3), 313 bytes | 313.00 KiB/s, done.  
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0  
To http://192.168.159.131/root/hello.git  
b8758f8..33d9e52 main -> main
```

- Sau đó kiểm tra lại file trên project đã tạo tên gitlab Server bằng cách reload lại trang:

Name	Last commit	Last update
 .gitlab-ci.yml	add file	3 minutes ago
 README.md	Initial commit	30 minutes ago

- File mới đã được tạo.
- Cuối cùng kiểm tra kết quả bằng cách nhấn vào CI/CD và chọn pipelines kết quả như hình bên dưới.



The screenshot shows the GitLab CI/CD Pipelines page for a project named 'Hello'. The left sidebar contains navigation options like Project information, Repository, Issues, Merge requests, Pipelines, Editor, Jobs, Schedules, Security & Compliance, Deployments, Monitor, Infrastructure, Packages & Registries, Analytics, Wiki, Snippets, and Settings. The main content area shows the 'Pipelines' tab with a table of pipeline runs. The table has columns for Status, Pipeline, Triggerer, and Stages. A single pipeline run is shown with a status of 'running' (indicated by a blue circle with a play icon), pipeline name 'add file', commit hash '#26 main -> 33d9e52a', and a 'latest' tag. The Triggerer column shows a green gear icon, and the Stages column shows a blue play icon. There are also buttons for 'Clear runner caches', 'CI lint', and 'Run pipeline' at the top right.

- Cuối cùng nhấn vào Jobs để xem các bước chạy và in thành công “Hello World”

passed Job hello world triggered 1 minute ago by Administrator

```
1 Running with gitlab-runner 14.8.2 (c6e7e194)
2   on hello stmYBWy3
3   ✓ Preparing the "docker" executor 00:51
4     Using Docker executor with image centos:8 ...
5     Pulling docker image centos:8 ...
6     Using docker image sha256:5d0da3dc976460b72c77d94c8a1ad043720b0416bfc16c52c45d4847e53fadb6 for centos:8 with digest centos@sha256:a27fd8080b517143cbbbab9dfb7c8571c40d67d534bbdee55bd6c473f432b177 ...
7   ✓ Preparing environment 00:04
8     Running on runner-stmyby3-project-39-concurrent-0 via localhost.localdomain...
9   ✓ Getting source from Git repository 00:03
10    Fetching changes with git depth set to 20...
11    Initialized empty Git repository in /builds/root/hello/.git/
12    Created fresh repository.
13    Checking out 33d9e52a as main...
14    Skipping Git submodules setup
15  ✓ Executing "step_script" stage of the job script 00:02
16    Using docker image sha256:5d0da3dc976460b72c77d94c8a1ad043720b0416bfc16c52c45d4847e53fadb6 for centos:8 with digest centos@sha256:a27fd8080b517143cbbbab9dfb7c8571c40d67d534bbdee55bd6c473f432b177 ...
17    $ echo Hello World
18    Hello World
19    Job succeeded
```

Chú ý: ở đâu sử dụng docker excutor để thực thi nếu bạn chưa tải nó sẽ báo lỗi bạn cũng có thể thay đổi thành shell ở bước register gitlab-runner để không gặp lỗi hoặc nếu muốn cài docker để sử dụng về sau có thể tham khảo các bước cài sau:

<https://www.liquidweb.com/kb/how-to-install-docker-on-centos-8/>

```
t none of the providers can be installed
- package docker-ce-3:20.10.7-3.el8.x86_64 requires containerd.io >= 1.4.1, bu
t none of the providers can be installed
- package docker-ce-3:20.10.8-3.el8.x86_64 requires containerd.io >= 1.4.1, bu
t none of the providers can be installed
- package docker-ce-3:20.10.9-3.el8.x86_64 requires containerd.io >= 1.4.1, bu
t none of the providers can be installed
- package docker-ce-3:20.10.10-3.el8.x86_64 requires containerd.io >= 1.4.1, b
ut none of the providers can be installed
- package docker-ce-3:20.10.11-3.el8.x86_64 requires containerd.io >= 1.4.1, b
ut none of the providers can be installed
- package docker-ce-3:20.10.12-3.el8.x86_64 requires containerd.io >= 1.4.1, b
ut none of the providers can be installed
- package docker-ce-3:20.10.13-3.el8.x86_64 requires containerd.io >= 1.4.1, b
ut none of the providers can be installed
(try to add '--allowerasing' to command line to replace conflicting packages or
'--skip-broken' to skip uninstalleable packages)
[root@localhost ~]#
```

- Nếu gặp lỗi ở bước này chỉ cần nhập lại:
dnf install docker-ce --nobest --allowerasing
- Khi đó, Gitlab sẽ bỏ qua các lỗi và tiếp tục cài đặt docker:

```
Verifying      : docker-ce-rootless-extras-20.10.13-3.el8.x86_64      6/12
Verifying      : docker-scan-plugin-0.17.0-3.el8.x86_64              7/12
Verifying      : buildah-1.22.3-2.module_el8.5.0+911+f19012f9.x86_6  8/12
Verifying      : cockpit-podman-33-1.module_el8.5.0+890+6b136101.no  9/12
Verifying      : containers-common-2:1-2.module_el8.5.0+890+6b13610 10/12
Verifying      : podman-3.3.1-9.module_el8.5.0+988+b1f0b741.x86_64 11/12
Verifying      : podman-catatonit-3.3.1-9.module_el8.5.0+988+b1f0b7 12/12

nstalled:
containerd.io-1.5.10-3.1.el8.x86_64
docker-ce-3:20.10.13-3.el8.x86_64
docker-ce-cli-1:20.10.13-3.el8.x86_64
docker-ce-rootless-extras-20.10.13-3.el8.x86_64
docker-scan-plugin-0.17.0-3.el8.x86_64
libcgroup-0.41-19.el8.x86_64
emoved:
buildah-1.22.3-2.module_el8.5.0+911+f19012f9.x86_64
cockpit-podman-33-1.module_el8.5.0+890+6b136101.noarch
containers-common-2:1-2.module_el8.5.0+890+6b136101.noarch
podman-3.3.1-9.module_el8.5.0+988+b1f0b741.x86_64
podman-catatonit-3.3.1-9.module_el8.5.0+988+b1f0b741.x86_64

omplete!
root@localhost ~]# █
```

- Sau đó chúng ta nhập: **systemctl enable --now docker** để bật docker lên.
- Tiếp tục nhập để confirm docker: **systemctl status docker**.
- Sau đó, add user là root để cấp quyền thực thi docker: **usermod -aG docker root**.