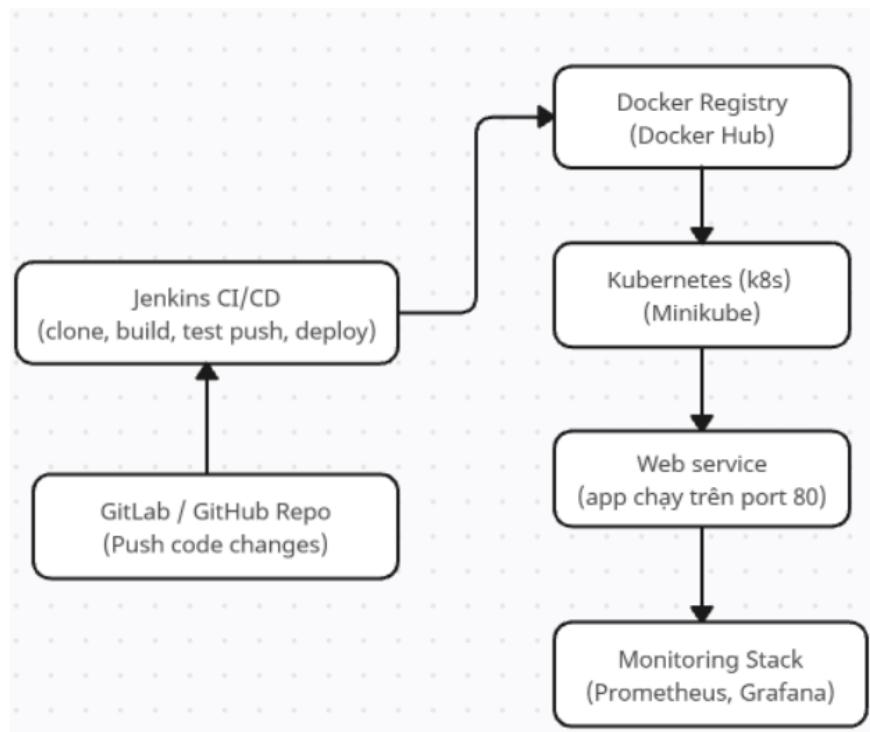


CASE STUDY #1: Thực hành quy trình DevOps minh họa việc triển khai ứng dụng Website đơn giản (“Hello, CI/CD!”) sử dụng các công cụ, công nghệ, dịch vụ: Git, GitLab/GitHub; Docker, Jenkins, K8s, Prometheus & Grafana.

Bài toán:

Một nhóm phát triển phần mềm đang xây dựng một ứng dụng web đơn giản bằng *Node.js* và *Express*. Họ muốn triển khai quy trình CI/CD hoàn chỉnh giúp tự động hóa quá trình từ *code* → *build* → *test* → *container hoá* → *triển khai lên Kubernetes* → *giám sát tình trạng hệ thống*. Các công nghệ, công cụ sử dụng bao gồm: *Jenkins*, *Docker*, *Kubernetes* và tích hợp giám sát với *Prometheus*, *Grafana*.

Sơ đồ quy trình DevOps (DevOps pipeline):



Yêu cầu:

- Triển khai ứng dụng web Node.js
- Viết Dockerfile và build/test image
- Push Docker image lên Docker Hub
- Cài đặt Jenkins, tạo pipeline có Jenkinsfile
- Deploy ứng dụng lên Kubernetes
- Tích hợp Prometheus & Grafana
- Báo cáo đầy đủ và logic

Kết quả mong đợi:

- Ứng dụng web Node.js được triển khai thành công thông qua CI/CD/DevOps pipeline.
- Mỗi lần cập nhật mã nguồn → Jenkins sẽ tự động build, deploy mới lên Kubernetes.
- Hệ thống được giám sát thời gian thực qua Prometheus & Grafana.

Gợi ý các công việc cần thực hiện:

1. *Về hạ tầng và môi trường:*
 - Cài đặt và cấu hình môi trường lab trên Ubuntu Desktop 24.04 LTS (trong máy ảo Oracle VirtualBox).
 - Cấu hình Docker, Jenkins, Minikube (Kubernetes).
 - Thiết lập các tài khoản GitLab/GitHub và Docker Hub để lưu trữ code và Docker image.
2. *Về ứng dụng:*
 - Tạo một ứng dụng Node.js (Express) đơn giản có thể chạy cổng 8080.
 - Cấu trúc mã nguồn phù hợp để container hóa (có Dockerfile, package.json, app.js).
3. *Về quy trình CI/CD với Jenkins:*
 - Tạo pipeline Jenkins sử dụng Jenkinsfile với các stage sau:
 - Clone Repository: lấy mã nguồn từ GitLab.
 - Build Docker Image: tạo Docker image từ Dockerfile.
 - Push Docker Image: đẩy image lên Docker Hub.
 - Deploy đến Kubernetes: triển khai ứng dụng lên Minikube bằng `kubectl apply`.
 - Tạo các Jenkins credentials cho:
 - GitLab (Personal Access Token)
 - Docker Hub
 - Kubeconfig
4. *Về triển khai ứng dụng trên Kubernetes:*
 - Viết file `deployment.yaml` và `service.yaml` để:
 - Tạo deployment có 2 replicas.
 - Expose service qua NodePort để truy cập qua browser.
 - Cấu hình imagePullSecrets nếu Docker Hub private.
 - Theo dõi và kiểm tra pods, services qua `kubectl` và `minikube dashboard`.
5. *Về giám sát hệ thống:*
 - Cài đặt Prometheus và Grafana trên Minikube thông qua Helm.
 - Tùy chỉnh cấu hình `values.yaml` để thêm job giám sát ứng dụng Node.js.
 - Truy cập dashboard Grafana qua port-forward để xem metrics.
6. *Về kiểm thử và đánh giá kết quả:*
 - Kiểm tra pipeline bằng cách push code mới lên GitLab → Jenkins tự động build, deploy lại ứng dụng.
 - Sử dụng `kubectl rollout restart` để triển khai lại nếu cần.
 - Kiểm tra ứng dụng qua trình duyệt và dashboard Prometheus/Grafana.

⇒ Hướng dẫn các bước triển khai chi tiết:

Môi trường thực hiện:

Oracle VirtualBox 7.0.14.

Ubuntu Desktop 24.04 LTS (Noble Numbat).

RAM: 16GB

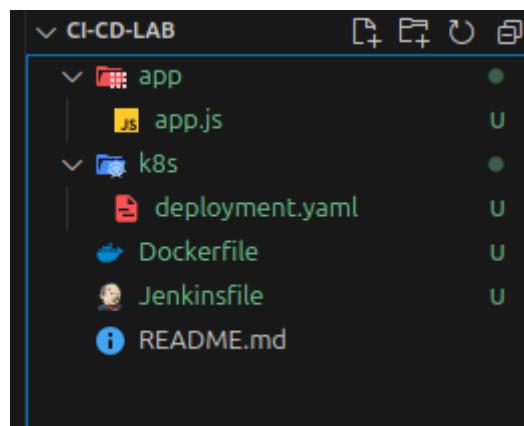
Core: 4

SSD: 20-30GB

Yêu cầu: Có tài khoản github/gitlab, dockerhub.

Bước 1: Tạo một Repository Git

- Tạo một repo trên gitlab. Sau đó clone về máy local.
- Tạo các file theo cấu trúc sau:



- Trong file app.js code một web đơn giản sử dụng nodejs và express.

```
app > js app.js > ...
1  const express = require("express");
2  const app = express();
3
4  app.get("/", (req, res) => {
5    res.send("Hello, CI/CD!");
6  );
7
8  app.listen(8080, () => {
9    console.log("App is running on http://localhost:8080");
10 });
11
```

Bước 2: Cài docker - Tạo Dockerfile

1. Cài đặt docker engine.

Gõ các gói gây xung đột:

```
for pkg in docker.io docker-doc docker-compose docker-compose-v2  
podman-docker containerd runc; do sudo apt-get remove $pkg; done
```

```
[lunox@lunox-VirtualBox:~$ for pkg in docker.io docker-doc docker-compose docker-compose-v2 podman-docker containerd runc; do sudo apt-get remove $pkg; done  
[sudo] password for lunox:  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
Package 'docker.io' is not installed, so not removed  
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
Package 'docker-doc' is not installed, so not removed  
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
```

Cài đặt docker repository:

```
# Add Docker's official GPG key:
```

```
sudo apt-get update
```

```
sudo apt-get install ca-certificates curl
```

```
sudo install -m 0755 -d /etc/apt/keyrings
```

```
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o
```

```
/etc/apt/keyrings/docker.asc
```

```
sudo chmod a+r /etc/apt/keyrings/docker.asc
```

```
# Add the repository to Apt sources:
```

```
echo \
```

```
 "deb [arch=$(dpkg --print-architecture) signed-
```

```
by=/etc/apt/keyrings/docker.asc]
```

```
https://download.docker.com/linux/ubuntu \
```

```
 $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
```

```
 sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

```
sudo apt-get update
```

```
Selecting previously unselected package curl.  
(Reading database ... 151220 files and directories currently installed.)  
Preparing to unpack .../curl_8.5.0-2ubuntu10.2_amd64.deb ...  
Unpacking curl (8.5.0-2ubuntu10.2) ...  
Setting up curl (8.5.0-2ubuntu10.2) ...  
Processing triggers for man-db (2.12.0-4build2) ...  
lunox@lunox-VirtualBox:~$ sudo install -m 0755 -d /etc/apt/keyrings  
lunox@lunox-VirtualBox:~$ sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc  
lunox@lunox-VirtualBox:~$ sudo chmod a+r /etc/apt/keyrings/docker.asc  
lunox@lunox-VirtualBox:~$ echo \  
 "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu \  
 $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \  
 sudo tee /etc/apt/sources.list.d/docker.list > /dev/null  
lunox@lunox-VirtualBox:~$ sudo apt-get update  
Get:1 https://download.docker.com/linux/ubuntu noble InRelease [48.8 kB]  
Hit:2 http://vn.archive.ubuntu.com/ubuntu noble InRelease  
Get:3 http://vn.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]  
Get:4 https://download.docker.com/linux/ubuntu/noble/stable amd64 Packages [12.4 kB]  
Hit:5 http://vn.archive.ubuntu.com/ubuntu noble-backports InRelease  
Hit:6 https://packages.microsoft.com/repos/code stable InRelease  
Hit:7 http://security.ubuntu.com/ubuntu noble-security InRelease  
Fetched 187 kB in is (180 kB/s)  
Reading package lists... Done  
lunox@lunox-VirtualBox:~$
```

Cài đặt docker engine:

```
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
```

```
lunox@lunox-VirtualBox:~$ sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
```

Test docker:

```
sudo docker run hello-world
```

```
[lunox@lunox-VirtualBox: ~]$ sudo docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
c1ec31eb5944: Pull complete
Digest: sha256:53cc4d15d839c98be39331c948609b659ed725170ad2ca8eb36951288f81b75
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/
```

Docker cài đặt thành công.

Cấu hình chạy docker không cần quyền root:

`sudo groupadd docker`

```
sudo usermod -aG docker $USER
```

newarp docker

Configure docker chạy với systemd:

```
sudo systemctl enable docker.service
```

```
sudo systemctl enable containerd.service
```

2. Tao Dockerfile.

Trong thư mục ci-cd-lab tạo Dockerfile:

```
FROM node:14
WORKDIR /usr/src/app
COPY package*.json .
RUN npm install express
COPY app/ ./app/
EXPOSE 8080
CMD ["node", "app/app.js"]
```

```
  Dockerfile ×
ci-cd-lab > Dockerfile > ...
5  WORKDIR /usr/src/app
6
7  # Sao chép tệp package.json và package-lock.json vào container
8  COPY package*.json ./
9
10 # Cài đặt các phụ thuộc cần thiết
11 RUN npm install express
12
13 # Sao chép toàn bộ mã nguồn ứng dụng vào container
14 COPY app/ ./app/
15
16 # Mở cổng 8080 để container có thể nhận kết nối
17 EXPOSE 8080
18
19 # Lệnh để khởi chạy ứng dụng Node.js khi container bắt đầu chạy
20 CMD ["node", "app/app.js"]
21
```

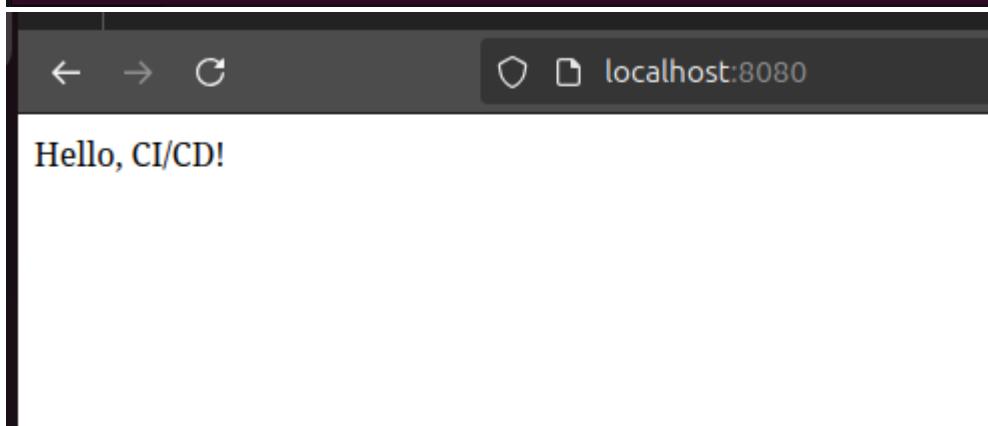
3. Build and test:

```
docker build -t ci-cd-lab-app .
```

```
[+] Building 47.5s (10/10) FINISHED
=> [Internal] Load build definition from Dockerfile
=> => transferring dockerfile: 619B
=> [Internal] load metadata for docker.io/library/node:14
=> [Internal] load .dockerignore
docker:default
  0.0s
  0.0s
  4.4s
  0.1s
```

```
docker run -p 8080:8080 ci-cd-lab-app
```

```
[lunox@lunox-VirtualBox:~/Desktop/DevOps/ci-cd-lab]$ docker run -p 8080:8080 ci-cd-lab-app
App is running on http://localhost:8080
```



Bước 3: Cài đặt và cấu hình Jenkins pipeline.

1. Cài đặt Jenkins.

<https://www.jenkins.io/doc/book/installing/linux/>

Cài đặt java cho jenkins:

```
sudo apt update  
sudo apt install fontconfig openjdk-17-jre
```

```
lunox@lunox-VirtualBox:~/Desktop/DevOps/ci-cd-lab$ sudo apt update  
[sudo] password for lunox:  
Hit:1 https://download.docker.com/linux/ubuntu noble InRelease  
Hit:2 http://vn.archive.ubuntu.com/ubuntu noble InRelease  
Get:3 http://vn.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]  
Hit:4 http://vn.archive.ubuntu.com/ubuntu noble-backports InRelease  
Hit:5 https://packages.microsoft.com/repos/code stable InRelease  
Hit:6 http://security.ubuntu.com/ubuntu noble-security InRelease  
Fetched 126 kB in 1s (101 kB/s)  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
All packages are up to date.  
lunox@lunox-VirtualBox:~/Desktop/DevOps/ci-cd-lab$ sudo apt install fontconfig openjdk-17-jre  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
fontconfig is already the newest version (2.15.0-1.1ubuntu2).  
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
```

Cài đặt jenkins:

```
sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \  
https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key  
echo "deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc]" \  
https://pkg.jenkins.io/debian-stable binary/ | sudo tee \  
/etc/apt/sources.list.d/jenkins.list > /dev/null  
sudo apt-get update  
sudo apt-get install jenkins
```

```

HTTP request sent, awaiting response... 200 OK
Length: 3175 (3.1K) [application/pgp-keys]
Saving to: '/usr/share/keyrings/jenkins-keyring.asc'

/usr/share/keyrings/jenkins-keyring.asc   100%[=====] 3.10K ---KB/s   in 0s

2024-08-17 20:12:43 (8.03 MB/s) - '/usr/share/keyrings/jenkins-keyring.asc' saved [3175/3175]

lunox@lunox-VirtualBox:~/Desktop/DevOps/ci-cd-lab$ sudo apt update
Hit:1 https://download.docker.com/linux/ubuntu noble InRelease
Hit:2 http://vn.archive.ubuntu.com/ubuntu noble InRelease
Hit:3 http://vn.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:4 http://vn.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:5 https://packages.microsoft.com/repos/code stable InRelease
Ign:6 https://pkg.jenkins.io/debian-stable binary/ InRelease
Get:7 https://pkg.jenkins.io/debian-stable binary/ Release [2,044 B]
Get:8 https://pkg.jenkins.io/debian-stable binary/ Release.gpg [833 B]
Hit:9 http://security.ubuntu.com/ubuntu noble-security InRelease
Get:10 https://pkg.jenkins.io/debian-stable binary/ Packages [27.6 kB]
Fetched 30.4 kB in 2s (17.0 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
All packages are up to date.
lunox@lunox-VirtualBox:~/Desktop/DevOps/ci-cd-lab$ sudo apt install jenkins
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  apt-tools

```

Thay đổi cổng mặc định của jenkins:

```

GNU nano 7.2                               /lib/systemd/system/jenkins.service *
# Port to listen on for HTTP requests. Set to -1 to disable.
# To be able to listen on privileged ports (port numbers less than 1024),
# add the CAP_NET_BIND_SERVICE capability to the AmbientCapabilities
# directive below.
Environment="JENKINS_PORT=8888"

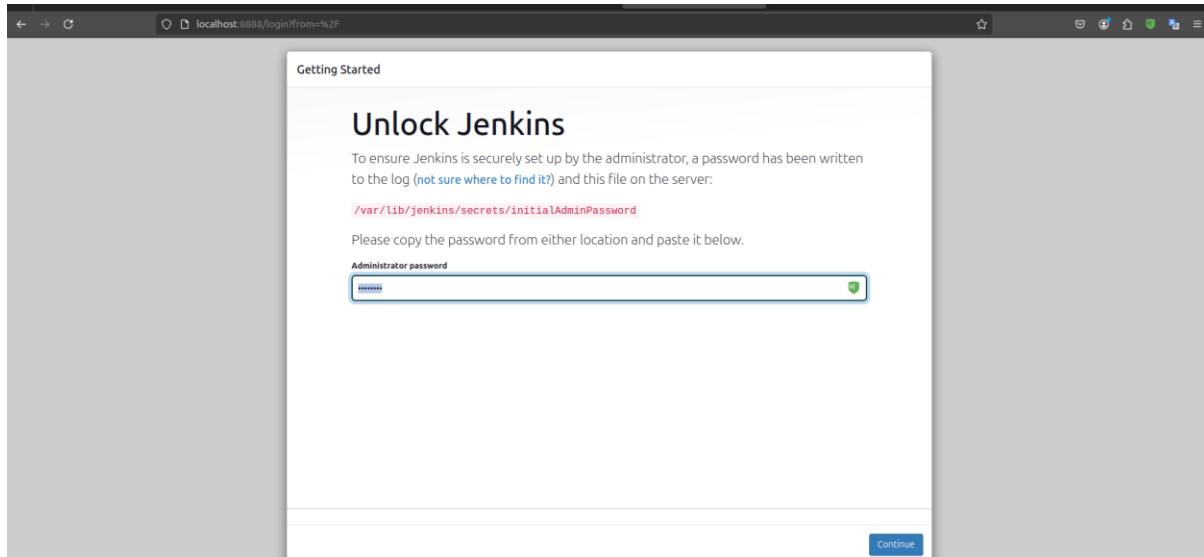
# IP address to listen on for HTTPS requests. Default is disabled.

```

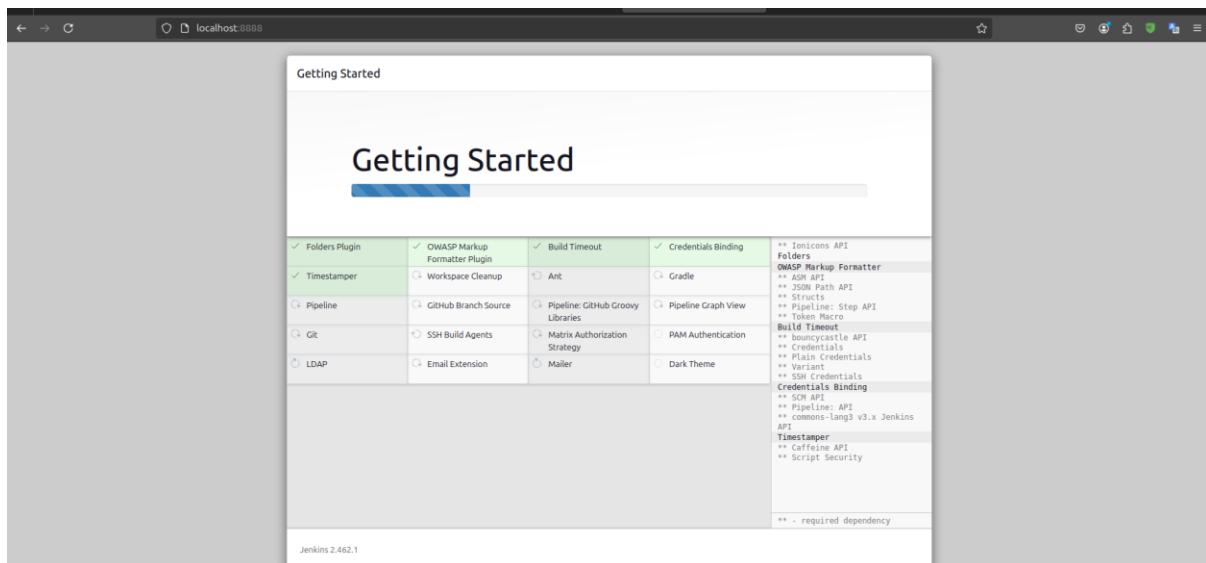
Truy cập web GUI của jenkins qua url: <http://localhost:8888>

Mật khẩu đăng nhập lần đầu lấy ở:

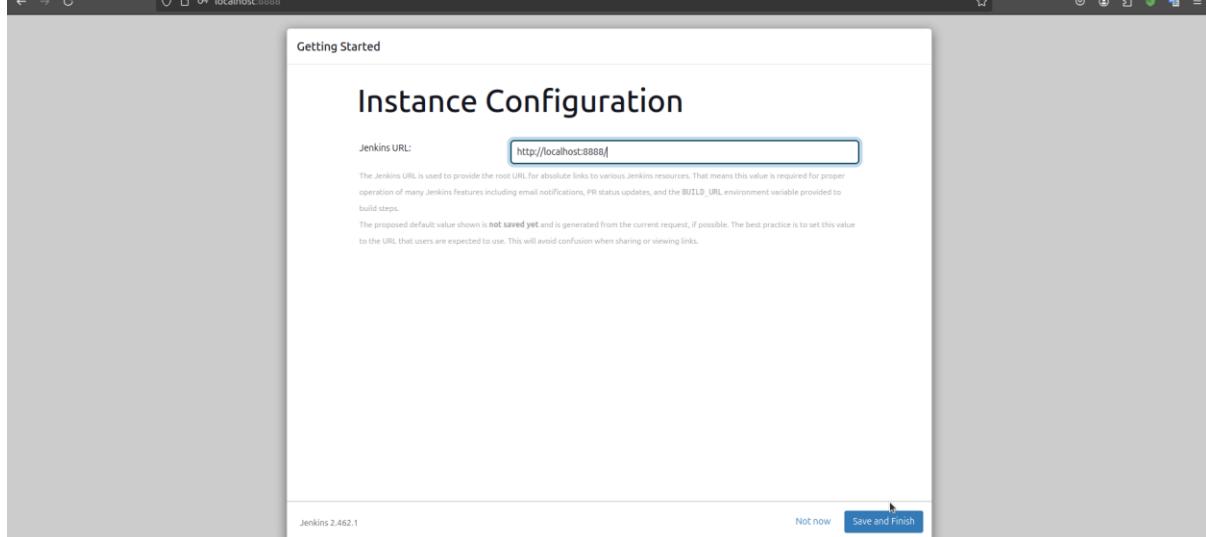
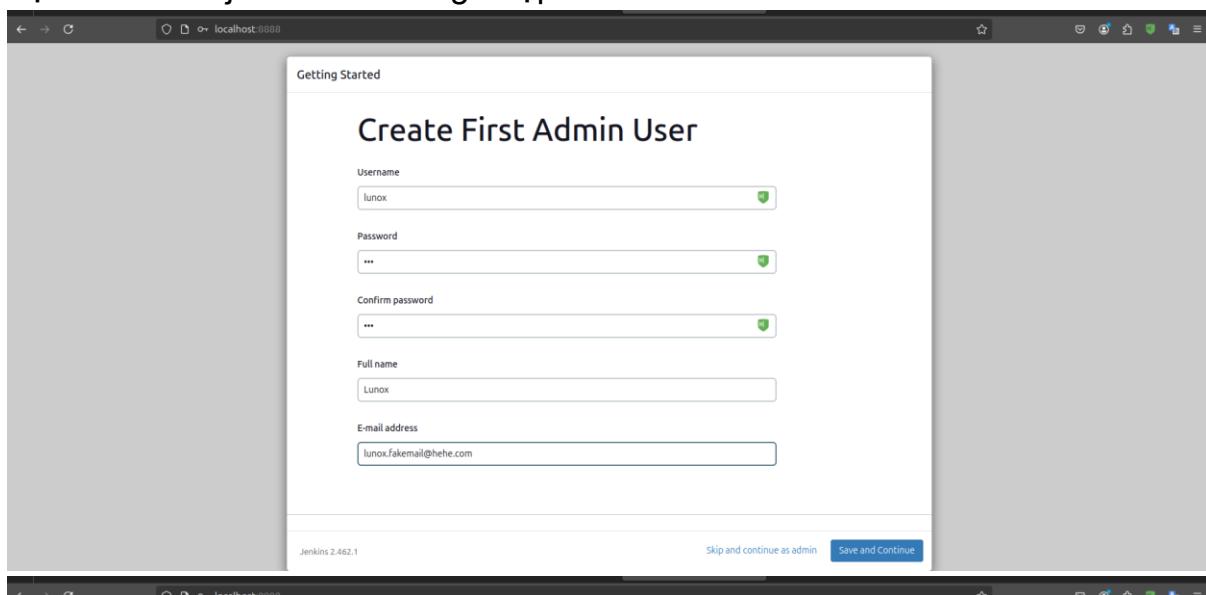
`/var/lib/jenkins/secrets/initialAdminPassword`



Chọn `install suggestion plugins` để tiếp tục



Tạo tài khoản jenkins để đăng nhập lần sau:



Welcome to Jenkins!

This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.

Start building your software project

Build Queue

No builds in the queue.

Build Executor Status

1 idle
2 idle

Create a job

+ Add description

Set up a distributed build

Set up an agent

Configure a cloud

Learn more about distributed builds

Cài đặt thành công.

Thêm các plugins cần thiết: Kubernetes, Kubernetes CLI, Docker, Git, Docker pipeline, ...

Available plugins

Install	Name	Released
✓	Kubernetes 4285.v50ed5f624918	5 days 0 hr ago
✓	Kubernetes Credentials 189.v90a_480b_d1d65	2 days 21 hr ago
✓	Docker 1.6.2	2 mo 14 days ago
✓	Docker Pipeline 580.vc0c340686b_54	2 mo 27 days ago
✓	CloudBees Docker Hub/Registry Notification 2.7.2	6 mo 27 days ago
✓	Kubernetes Credentials Provider 1.262.v2670ef7ea_0c5	5 mo 20 days ago

Plugins

Plugin	Status
E-mail Extension	Success
Mailer	Success
Theme Manager	Success
Dark Theme	Success
Loading plugin extensions	Success
Kubernetes Client API	Success
Authentication Tokens API	Success
Kubernetes Credentials	Pending
Kubernetes	Pending
Kubernetes Credentials	Pending
Cloud Statistics	Pending
Docker Commons	Pending
Apache HttpClient Components 5.x API	Pending
Docker API	Pending
Docker	Pending
Docker Pipeline	Pending
CloudBees Docker Hub/Registry Notification	Pending
Kubernetes Credentials Provider	Pending
Loading plugin extensions	Pending
Restarting Jenkins	Pending

→ Go back to the top page
(you can start using the installed plugins right away)

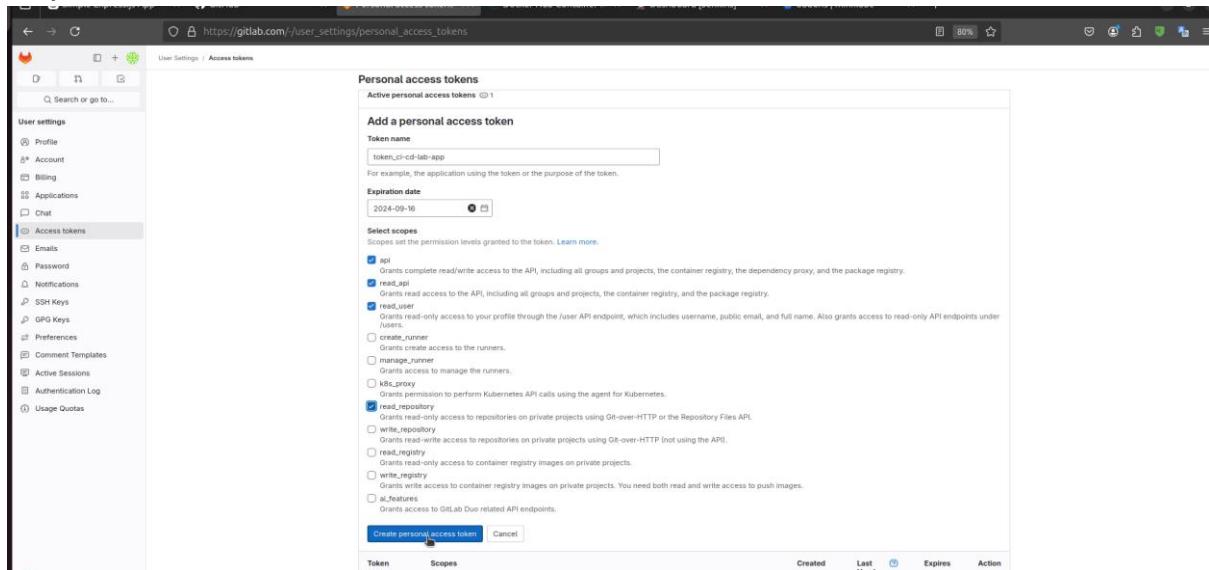
→ Restart Jenkins when installation is complete and no jobs are running

2.Thêm các credential:

a. Gitlab credential:

Truy cập: https://gitlab.com/-/user_settings/personal_access_tokens

Chọn Add new token.

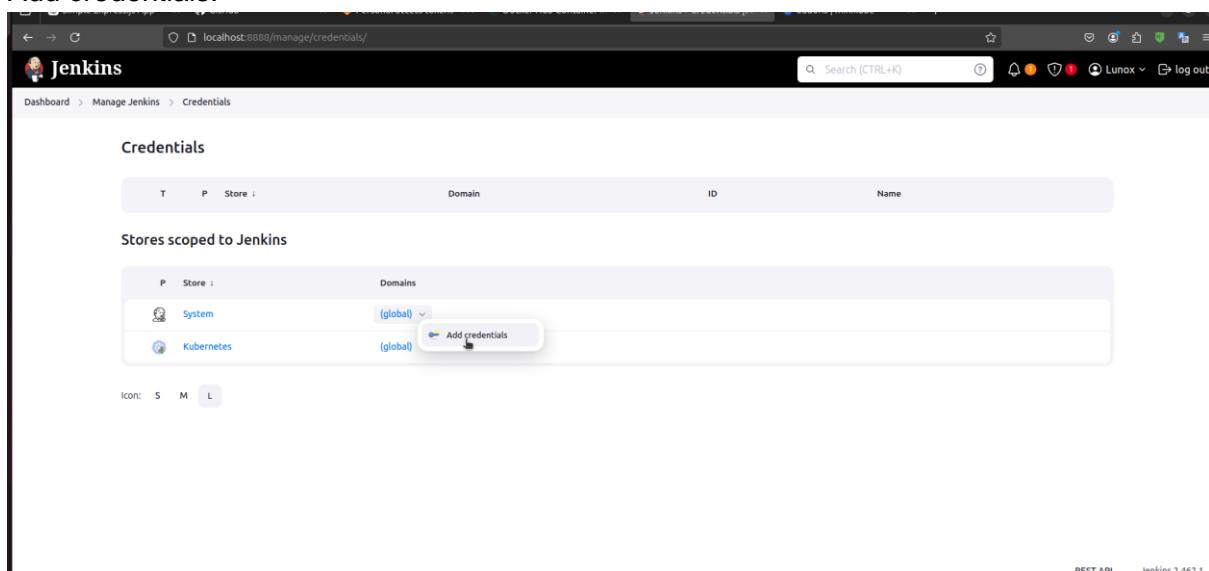


The screenshot shows the 'Personal access tokens' section of the GitLab user settings. A new token is being created with the name 'token_ci_cd_lab-app'. The expiration date is set to '2024-09-10'. Under 'Select scopes', several options are checked: 'api', 'read_repository', 'write_repository', 'read_registry', 'write_registry', and 'ai_features'. At the bottom, there is a blue 'Create personal access token!' button.

copy token và lưu lại trong một file bí mật.

Quay lại web gui jenkins. > Manage Jenkins > Credentials.

Add credentials:



The screenshot shows the Jenkins 'Credentials' management page. A new credential is being added under the 'System' store. The 'Add credentials' button is highlighted with a mouse cursor. The table below lists existing credentials: 'System' (global) and 'Kubernetes' (global).

Ở phần password điền access token vừa tạo ở gitlab.

New credentials

Kind: Username with password

Scope: Global (Jenkins, nodes, items, all child items, etc)

Username: nguyenle2k3

Treat username as secret

Password:

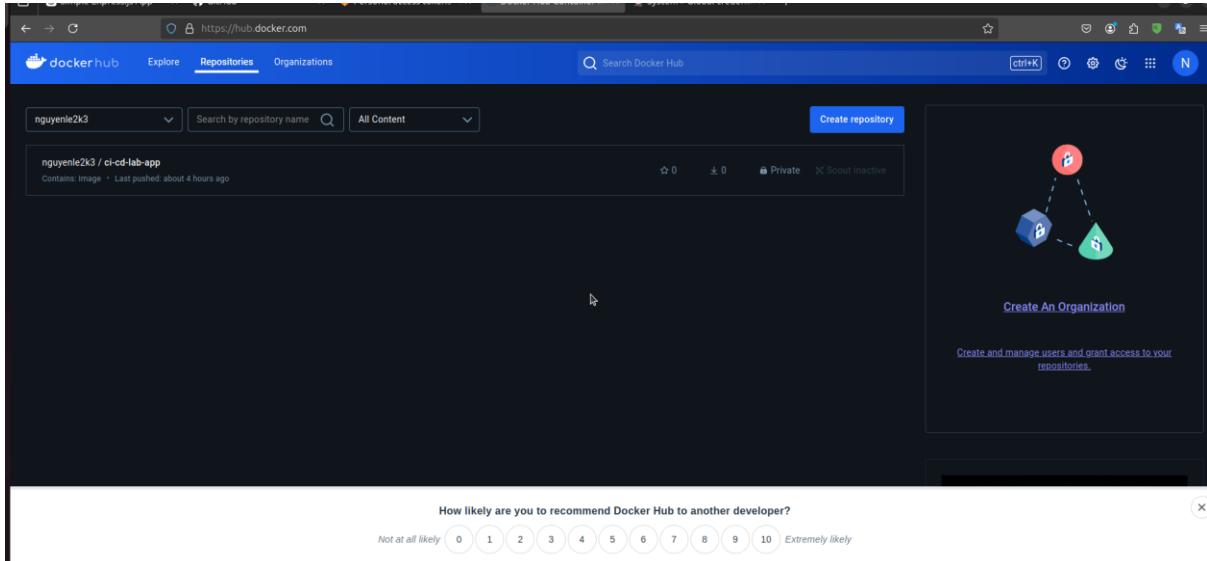
ID: gitlab_credentials

Description: Credential access gitlab

Create

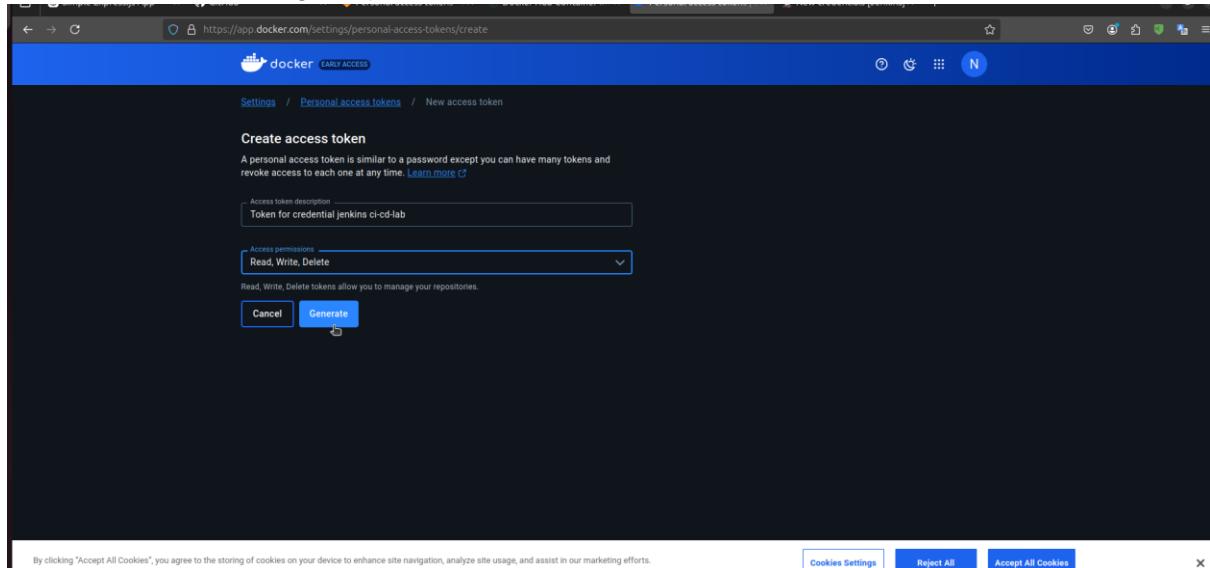
b. Docker hub credential:

Đăng nhập vào docker hub, tạo một repo mới là tên app: ci-cd-lab-app.

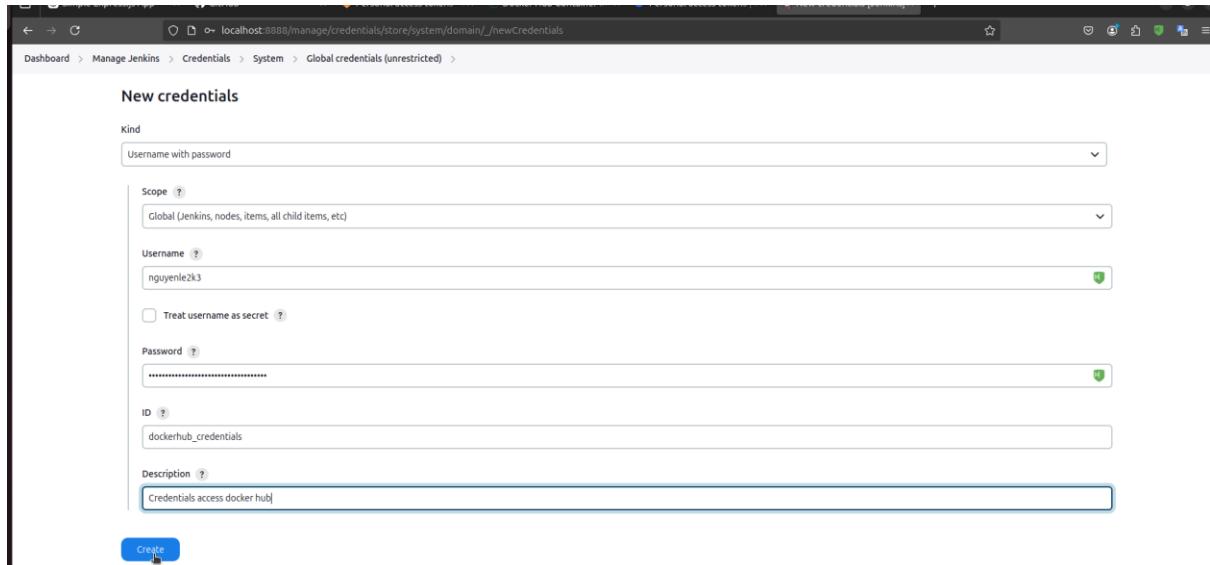


Tương tự với tạo credential gitlab, tạo một credentials cho docker hub.

Tạo access token đăng nhập



Tạo credentials:



3. Tạo Jenkinsfile.

Tạo file ci-cd-lab/Jenkinsfile:

```
pipeline {
    agent any
    environment {
        DOCKER_IMAGE = 'nguyenle2k3/ci-cd-lab-app'
        // Cần có Dockerhub để lấy registry và jenkins đẩy image lên.
        REGISTRY = 'https://index.docker.io/v1/'
        // Cần cấu hình kube credentials trước khi chạy
        KUBE_CONFIG = credentials('kube-config')
    }
    stages {
```

```
stage('Clone Repository') {
    steps {
        git credentialsId: 'gitlab_credentials', url:
'https://gitlab.com/lunox347/ci-cd-lab.git', branch: 'main'
    }
}
stage('Build Docker Image') {
    steps {
        script {
            dockerImage = docker.build("${DOCKER_IMAGE}:latest")
        }
    }
}
stage('Push Docker Image') {
    steps {
        script {
            docker.withRegistry("${REGISTRY}", 'dockerhub_credentials') {
                dockerImage.push()
            }
        }
    }
}
stage('Deploy to Kubernetes') {
    steps {
        withKubeConfig([credentialsId: 'kube-config']) {
            sh 'kubectl apply -f k8s/deployment.yaml'
        }
    }
}
post {
    always {
        cleanWs()
    }
}
}
```

```

File Edit Selection View Go Run Terminal Help
EXPLORER ci-cd-lab Jenkinsfile ...
ci-cd-lab > Jenkinsfile
nguyenle2k3, 6 hours ago | 1 author (nguyenle2k3)
1 pipeline {
2   agent any
3   environment {
4     DOCKER_IMAGE = 'nguyenle2k3/ci-cd-lab-app'
5     // Cần có Dockerhub để lấy registry và jenkins dán image lên.
6     REGISTRY = 'https://index.docker.io/v1/'
7     // Cần cài him kube credentials trước khi chạy
8     KUBE_CONFIG = credentials('kube-config')
9   }
10  stages {
11    stage('Clone Repository') {
12      steps {
13        git credentialsId: 'gitlab_credentials', url: 'https://gitlab.com/lunox347/ci-cd-lab.git', branch: 'main'
14      }
15    }
16    stage('Build Docker Image') {
17      steps {
18        script {
19          dockerImage = docker.build("${DOCKER_IMAGE}:latest")
20        }
21      }
22    }
23    stage('Push Docker Image') {
24      steps {
25        script {
26          docker.withRegistry("${REGISTRY}", 'dockerhub_credentials') {
27            dockerImage.push()
28          }
29        }
30      }
31    }
}

```

Bước 4: Cài đặt, cấu hình K8s - Minikube và tạo deployment:

1. Cài đặt minikube:

<https://minikube.sigs.k8s.io/docs/start/?arch=%2Flinux%2Fx86-64%2Fstable%2Fdebian+package>

curl -LO

https://storage.googleapis.com/minikube/releases/latest/minikube_latest_amd64.deb

sudo dpkg -i minikube_latest_amd64.deb

```

lunox@lunox-VirtualBox:~/Downloads$ curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube_latest_amd64.deb
% Total    % Received % Xferd  Average Speed   Time   Time  Current
          Dload  Upload Total Spent   Left Speed
100 27.6M  100 27.6M    0      0  9203k      0:00:03  0:00:03  --:--:-- 9204k
lunox@lunox-VirtualBox:~/Downloads$ sudo dpkg -i minikube_latest_amd64.deb
[sudo] password for lunox:
Selecting previously unselected package minikube.
(Reading database ... 151929 files and directories currently installed.)
Preparing to unpack minikube_latest_amd64.deb ...
Unpacking minikube (1.33.1-0) ...
Setting up minikube (1.33.1-0) ...
lunox@lunox-VirtualBox:~/Downloads$ 

```

Chạy cluster:

minikube start

```
lunox@lunox-VirtualBox:~/Downloads$ minikube start
minikube v1.33.1 on Ubuntu 24.04 (vbox/amd64)
Automatically selected the docker driver. Other choices: none, ssh
Using Docker driver with root privileges
Starting "minikube" primary control-plane node in "minikube" cluster
Pulling base image v0.0.44 ...
Downloading Kubernetes v1.30.0 preload ...
> preloaded-images-k8s-v18-v1...: 342.90 MiB / 342.90 MiB 100.00% 15.63 M
> gcr.io/k8s-minikube/kicbase...: 481.58 MiB / 481.58 MiB 100.00% 11.27 M
Creating docker container (CPUs=2, Memory=3900MB) ...
Preparing Kubernetes v1.30.0 on Docker 26.1.1 ...
■ Generating certificates and keys ...
■ Booting up control plane ...
■ Configuring RBAC rules ...
Configuring bridge CNI (Container Networking Interface) ...
Verifying Kubernetes components...
■ Using image gcr.io/k8s-minikube/storage-provisioner:v5
Enabled addons: storage-provisioner, default-storageclass
kubectl not found. If you need it, try: 'minikube kubectl -- get pods -A'
Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
lunox@lunox-VirtualBox:~/Downloads$
```

Cài minikubectl:

<https://kubernetes.io/docs/tasks/tools/install-kubectl-linux/>

```
curl -LO "https://dl.k8s.io/release/$(curl -L -s
https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl
```

Kiểm tra cluster:

kubectl get nodes

```
lunox@lunox-VirtualBox:~/Downloads$ kubectl get nodes
NAME      STATUS    ROLES      AGE      VERSION
minikube  Ready     control-plane  18m     v1.30.0
lunox@lunox-VirtualBox:~/Downloads$
```

2. Tạo tệp deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: ci-cd-lab-app
spec:
  replicas: 2
  selector:
  matchLabels:
```

```

app: ci-cd-lab-app
template:
metadata:
labels:
    app: ci-cd-lab-app
spec:
containers:
    - name: ci-cd-lab-app
        image: nguyenle2k3/ci-cd-lab-app:latest # Thay bằng Docker
image của bạn
ports:
    - containerPort: 8080
---
apiVersion: v1
kind: Service
metadata:
    name: ci-cd-lab-service
spec:
selector:
    app: ci-cd-lab-app
ports:
    - protocol: TCP
    port: 80
    targetPort: 8080
    type: NodePort

```

The screenshot shows a code editor interface with two files open in tabs:

- deployment.yaml**: A YAML file defining a Deployment resource. It specifies a pod template with a container named "ci-cd-lab-app" using the image "nguyenle2k3/ci-cd-lab-app:latest". The container runs on port 8080.
- service.yaml**: A YAML file defining a Service resource. It uses the same selector as the Deployment, exposing port 80 on the NodePort type.

The code editor has a dark theme and includes standard navigation and search tools.

```

File Edit Selection View Go Run Terminal Help
EXPLORER deployment.yaml ...
DEVOPS c-cd-lab
  app
  k8s
  deployment.yaml
  Dockerfile
  Jenkinsfile
  README.md
  tutorial.md
OUTLINE
TIMELINE

```

```

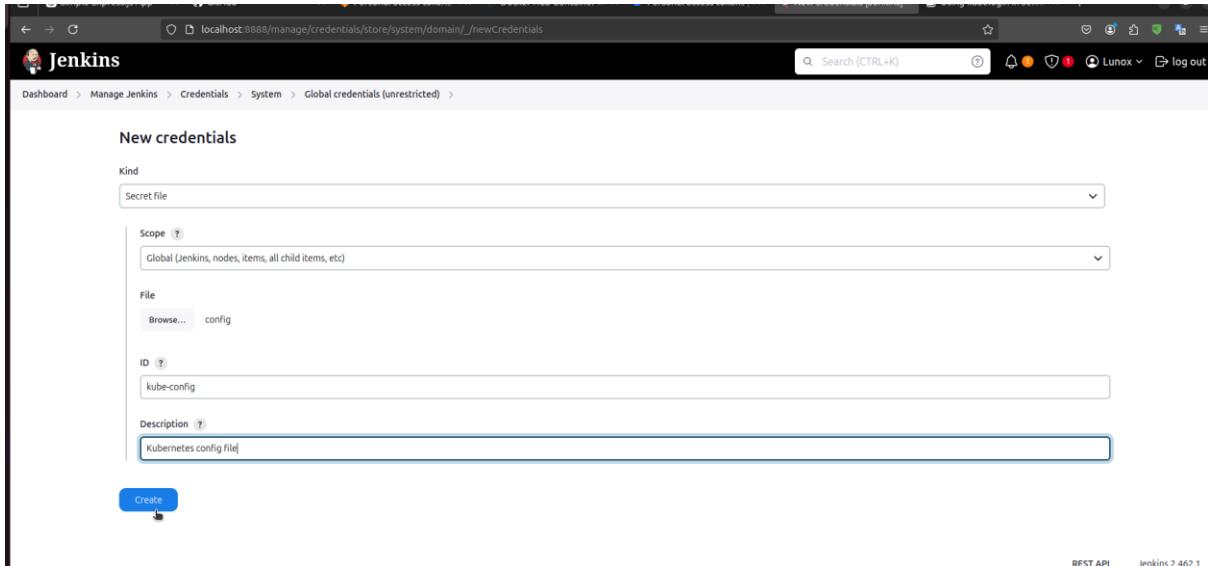
apiVersion: apps/v1
kind: Deployment
metadata:
  name: ci-cd-lab-app
spec:
  replicas: 2
  selector:
    matchLabels:
      app: ci-cd-lab-app
  template:
    metadata:
      labels:
        app: ci-cd-lab-app
    spec:
      containers:
        - name: ci-cd-lab-app
          image: nguyenle2k3/ci-cd-lab-app:latest # Thay bằng Docker image của bạn
          ports:
            - containerPort: 8080
...
apiVersion: v1
kind: Service
metadata:
  name: ci-cd-lab-service
spec:
  selector:
    app: ci-cd-lab-app
  ports:
    - protocol: TCP
      port: 80
      targetPort: 8080

```

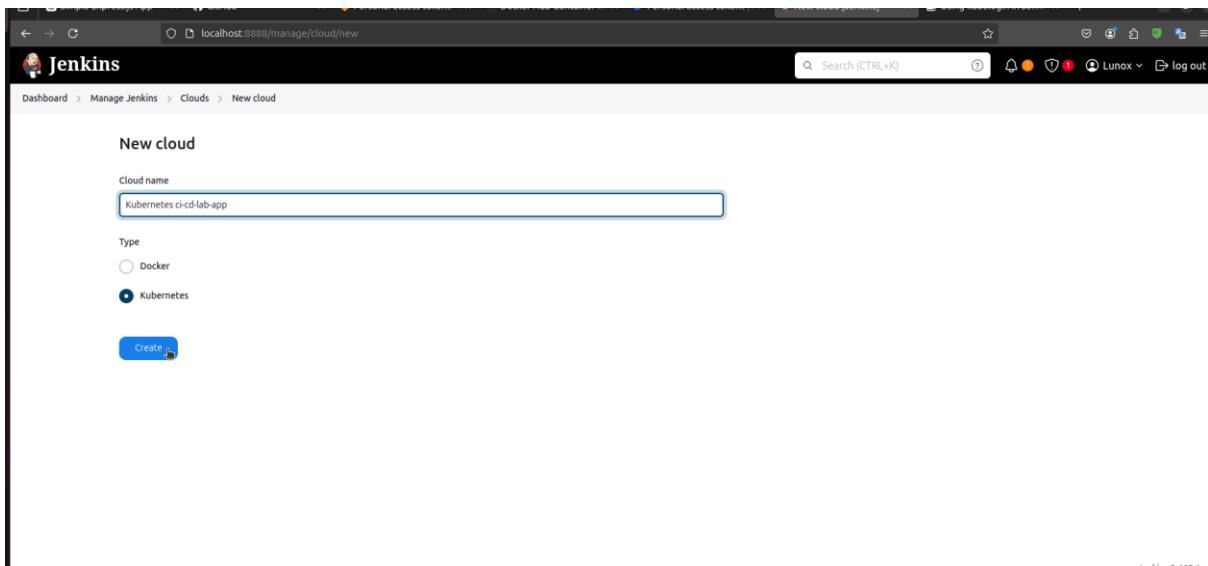
Bước 5: Tích hợp Jenkins với Kubernetes

Add quyền đọc ở /home/lunox/.kube/config và /home/lunox/minikube /home/lunox/minikube/ca.crt để jenkins đọc được kube config

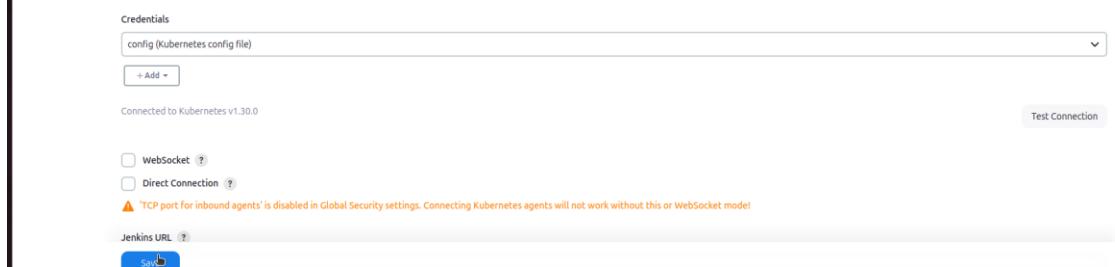
Add credentials kube-config trong jenkins:



Add cloud kubernetes trong jenkins:



add Credentials và test connection để chắc chắn credentials hoạt động đúng.



Cơ bản đã hoàn thành quá trình cấu hình. Giờ tạo một job để test.

Quay lại Dashboard > Create a job.

Nhập tên job, chọn pipeline.

Kéo xuống phần pipeline.

Definition: Pipeline script from SCM

SCM: Git

Repository URL: URL repo git lab

Credential: gitlab_credentials

Script path: Jenkinsfile

The screenshot shows the Jenkins Pipeline configuration page for a job named "ci-cd-lab-app". The left sidebar has "Configure" selected. The main area is titled "Pipeline" and shows "Definition: Pipeline script From SCM". Under "SCM", "Git" is selected. A "Repositories" section contains a single entry with "Repository URL: https://gitlab.com/lunox347/ci-cd-lab.git" and "Credentials: nguyenle2k3/******** (Credential access gitlab)". There is also an "Advanced" dropdown and a "Save" button at the bottom.

Chọn Build now để build app.

The screenshot shows the Jenkins job details page for "ci-cd-lab-app". The left sidebar lists options like Status, Changes, Build Now (which is highlighted), Configure, Delete Pipeline, Stages, Rename, and Pipeline Syntax. The main area displays the "Build History" section, which currently says "No builds". It includes links for "Atom feed for all" and "Atom feed for failures". At the bottom right, it shows "REST API" and "Jenkins 2.462.1".

Lịch sử các lượt build:

The screenshot shows a web browser window with the Jenkins logo at the top. The URL bar indicates the page is loaded from 'localhost'. The main content area displays the Jenkins pipeline interface for a project named 'ci-cd-lab-app'. The pipeline stages are listed on the left:

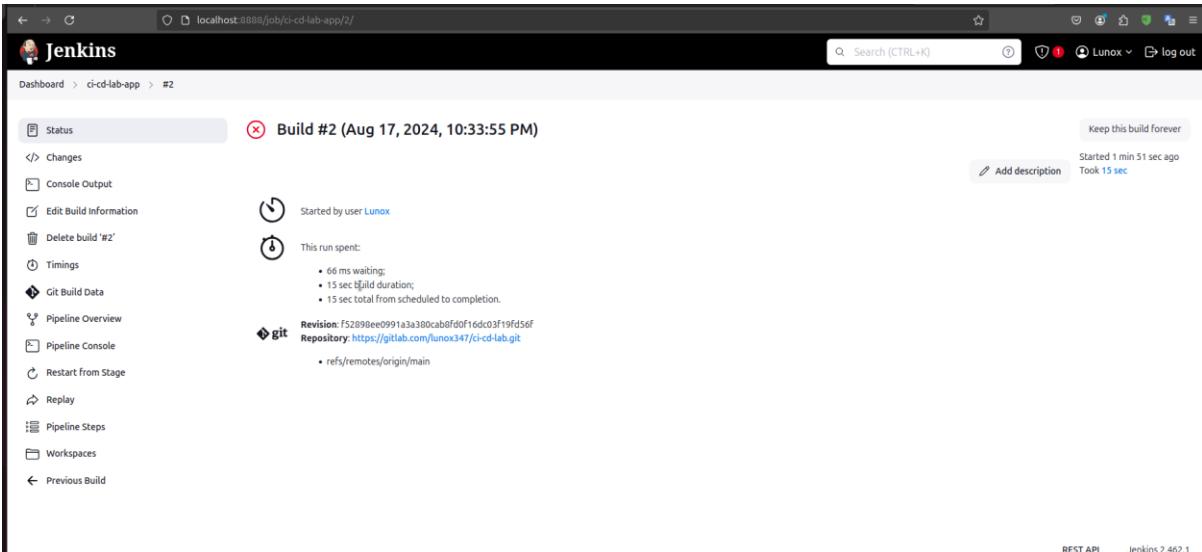
- Status
- </> Changes
- ▷ Build Now
- ⚙ Configure
- trash Delete Pipeline
- stacks Stages
- pencil Rename
- info Pipeline Syntax

Below the stages, there is a 'Build History' section with the following data:

Build #	Timestamp
#2	Aug 17, 2024, 10:33 PM
#1	Aug 17, 2024, 10:17 PM

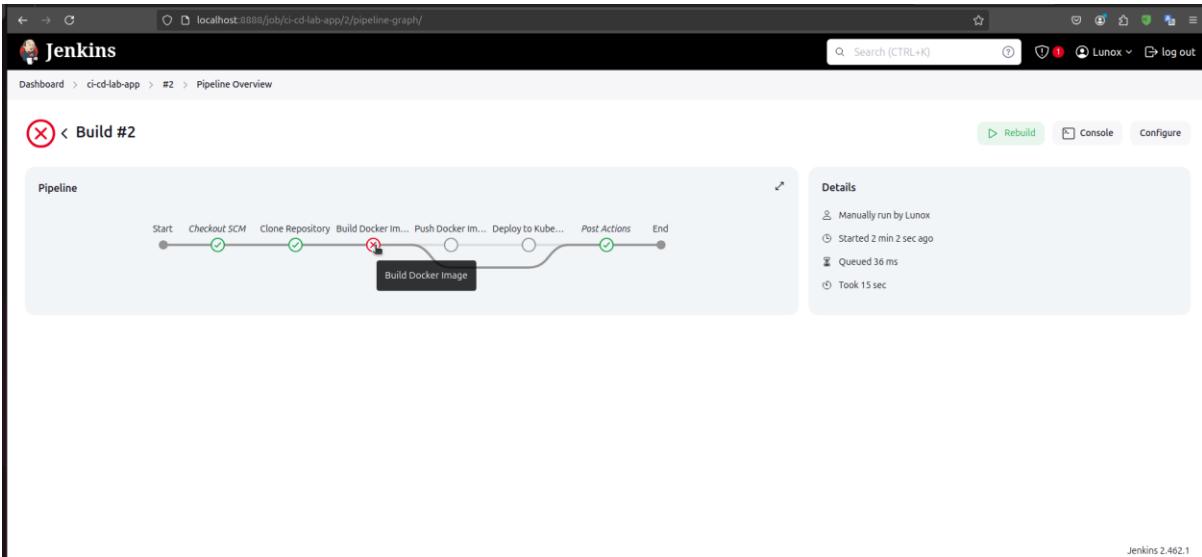
At the bottom of the history section, there are links for Atom feeds: 'Atom feed for all' and 'Atom feed for failures'.

Bấm vào #2 để xem chi tiết về lượt build



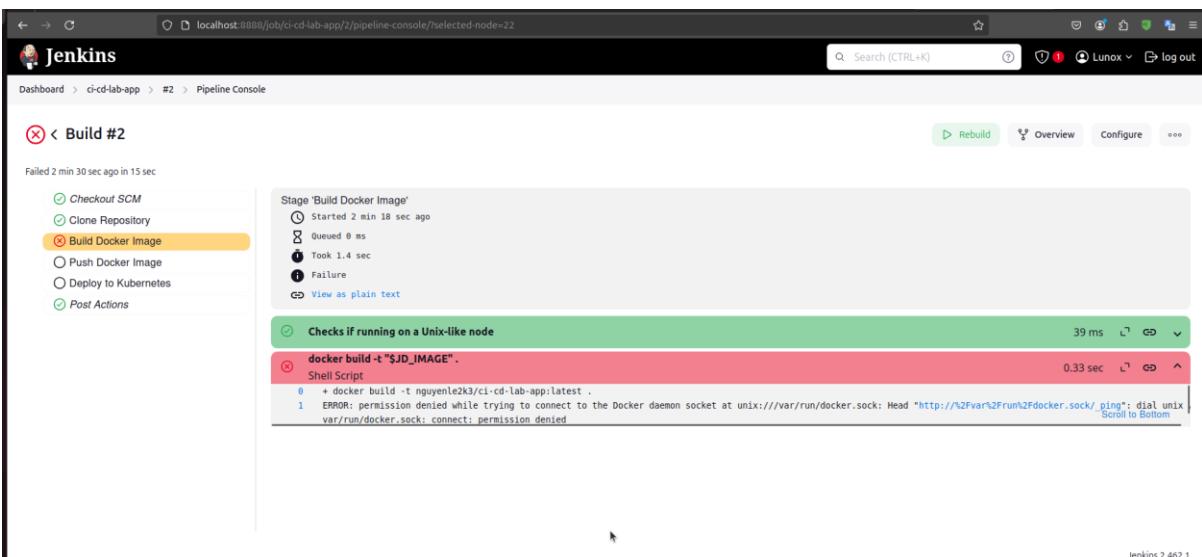
The screenshot shows the Jenkins job details for build #2. The main header says "Build #2 (Aug 17, 2024, 10:33:55 PM)". On the left, there's a sidebar with various build-related links like Status, Changes, Console Output, and Pipeline Overview. The central area displays the build configuration: "Started by user Lunox", "This run spent:", and a git commit log entry. At the bottom right, it shows "Started 1 min 51 sec ago" and "Took 15 sec".

Ta thấy build hỏng từ bước Build Docker Image



The screenshot shows the Jenkins Pipeline Overview for build #2. It displays a flowchart of the pipeline stages: Start, Checkout SCM, Clone Repository, Build Docker Image, Push Docker Image, Deploy to Kubernetes, Post Actions, and End. The "Build Docker Image" stage is highlighted with a red circle and a failure icon, indicating it failed. To the right, a "Details" panel provides information about the failed stage, including "Manually run by Lunox" and "Took 1.4 sec".

Xem chi tiết lỗi:



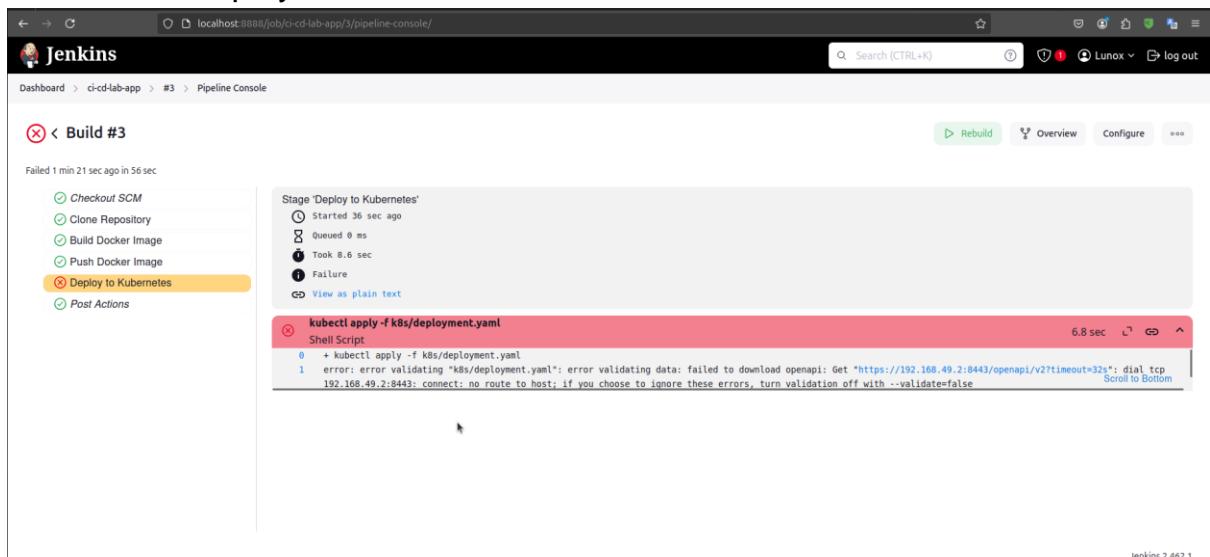
The screenshot shows the Jenkins Pipeline Console for build #2. The left sidebar lists the pipeline stages: Checkout SCM, Clone Repository, Build Docker Image (which is currently selected), Push Docker Image, Deploy to Kubernetes, and Post Actions. The main area shows the logs for the "Build Docker Image" stage. A specific error message is highlighted in red: "ERROR: permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Head http://%2Fvar%2Frun%2Fdocker.sock/ping: dial unix var/run/docker.sock: connect: permission denied".

Lỗi này do chưa add user jenkins vào group docker.

```
lunox@lunox-VirtualBox:~$ sudo usermod -aG docker jenkins
lunox@lunox-VirtualBox:~$ sudo systemctl restart jenkins.service
lunox@lunox-VirtualBox:~$ sudo systemctl restart docker.service
lunox@lunox-VirtualBox:~$ sudo systemctl restart docker.socket
lunox@lunox-VirtualBox:~$ sudo systemctl restart containerd.service
lunox@lunox-VirtualBox:~$
```

Build lại:

Lỗi ở bước Deploy:

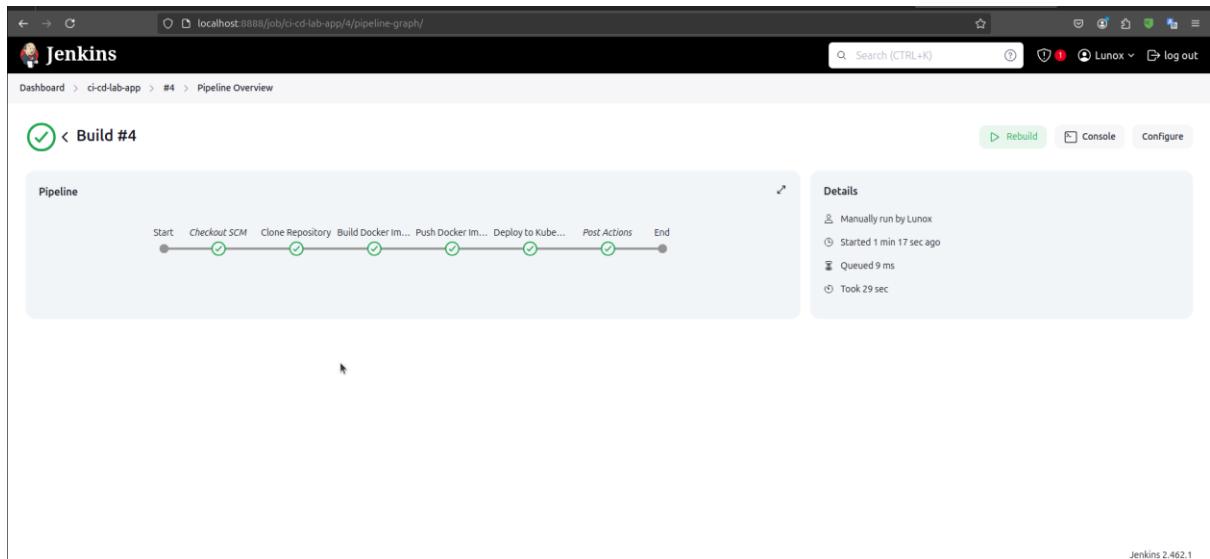


Lỗi gây ra do minikube bị tắt:

```
lunox@lunox-VirtualBox:~$ minikube status
minikube
  type: Control Plane
  host: Stopped
  kubelet: Stopped
  apiserver: Stopped
  kubeconfig: Stopped

lunox@lunox-VirtualBox:~$ minikube ip
💡 The control-plane node minikube host is not running: state=Stopped
👉 To start a cluster, run: "minikube start"
lunox@lunox-VirtualBox:~$ minikube start
😊 minikube v1.33.1 on Ubuntu 24.04 (vbox/amd64)
✨ Using the docker driver based on existing profile
👍 Starting "minikube" primary control-plane node in "minikube" cluster
_PULLING_
Pulling base image v0.0.44 ...
🔄 Restarting existing docker container for "minikube" ...
🛠️ Preparing Kubernetes v1.30.0 on Docker 26.1.1 ...
🔍 Verifying Kubernetes components...
  - Using image gcr.io/k8s-minikube/storage-provisioner:v5
🌟 Enabled addons: default-storageclass, storage-provisioner
🌟 Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
lunox@lunox-VirtualBox:~$
```

Rebuild:



Jenkins 2.462.1

Thành công.

Có thể sử dụng minikube dashboard để kiểm tra toàn diện minikube với web gui.

lunox@lunox-VirtualBox: ~ minikube dashboard

```

Enabling dashboard ...
  ▪ Using image docker.io/kubernetesui/metrics-scraper:v1.0.8
  ▪ Using image docker.io/kubernetesui/dashboard:v2.7.0
💡 Some dashboard features require the metrics-server addon. To enable all features please run:
  minikube addons enable metrics-server

💡 Verifying dashboard health ...
💡 Launching proxy ...
💡 Verifying proxy health ...
💡 Opening http://127.0.0.1:44791/api/v1/namespaces/kubernetes-dashboard/services/http:kubernetes-dashboard:/proxy/ in your default browser...
Gtk-Message: 09:32:30.388: Not loading module "atk-bridge": The functionality is provided by GTK natively. Please try to not load it.

```

localhost:44791/api/v1/namespaces/kubernetes-dashboard/services/http:kubernetes-dashboard:/proxy/#/workloads?namespace=default

Name	Images	Labels	Pods	Created
ci-cd-lab-app	nguyenie2k3/ci-cd-lab-app:latest	-	2 / 2	2 days ago

Name	Images	Labels	Node	Status	Restarts	CPU Usage (cores)	Memory Usage (bytes)	Created
ci-cd-lab-app	nguyenie2k3/ci-cd-lab-app:latest	app: ci-cd-lab-app	minikube	Running	0	0%	0B	2 days ago

Workloads

Name	Images	Labels	Pods	Created
ci-cd-lab-app	nguyene2k3/ci-cd-lab-app:latest	-	2 / 2	3 days ago

Pods

Name	Images	Labels	Node	Status	Restarts	CPU Usage (cores)	Memory Usage (bytes)	Created
ci-cd-lab-app-7c57d7dbbc-4kwhc	nguyene2k3/ci-cd-lab-app:latest	app: ci-cd-lab-app pod-template-hash: 7c57d7dbbc	minikube	Running	0	-	-	5 minutes ago
ci-cd-lab-app-7c57d7dbbc-mckbp	nguyene2k3/ci-cd-lab-app:latest	app: ci-cd-lab-app pod-template-hash: 7c57d7dbbc	minikube	Running	0	-	-	5 minutes ago

Replica Sets

Name	Images	Labels	Pods	Created
ci-cd-lab-app-7c57d7dbbc	nguyene2k3/ci-cd-lab-app:latest	app: ci-cd-lab-app pod-template-hash: 7c57d7dbbc	2 / 2	5 minutes ago
ci-cd-lab-app-6fb5479b4	nguyene2k3/ci-cd-lab-app:latest	app: ci-cd-lab-app pod-template-hash: 6fb5479b4	0 / 0	3 days ago
ci-cd-lab-app-64654489fd	nguyene2k3/ci-cd-lab-app:latest	app: ci-cd-lab-app pod-template-hash: 64654489fd	0 / 0	3 days ago

Pods

Name	Images	Labels	Node	Status	Restarts	CPU Usage (cores)	Memory Usage (bytes)	Created
ci-cd-lab-app-7c57d7dbbc-4kwhc	nguyene2k3/ci-cd-lab-app:latest	app: ci-cd-lab-app pod-template-hash: 7c57d7dbbc	minikube	Running	0	-	-	7 minutes ago
ci-cd-lab-app-7c57d7dbbc-mckbp	nguyene2k3/ci-cd-lab-app:latest	app: ci-cd-lab-app pod-template-hash: 7c57d7dbbc	minikube	Running	0	-	-	7 minutes ago

Service

Name	Labels	Type	Cluster IP	Internal Endpoints	External Endpoints	Created
ci-cd-lab-service	-	NodePort	10.101.119.189	ci-cd-lab-service:80 TCP ci-cd-lab-service:32680 TCP	-	3 days ago
kubernetes	component: apiserver provider: kubernetes	ClusterIP	10.96.0.1	kubernetes:443 TCP kubernetes:0 TCP	-	3 days ago

The image contains two screenshots of the Kubernetes Dashboard interface, both titled "default".

Screenshot 1: Config And Storage

- Config Maps:**

Name	Labels	Created
kube-root-ca.crt	-	3 days ago
- Secrets:**

Name	Labels	Type	Created
my-registry-key	-	kubernetes.io/dockerconfigjson	3 days ago
- Storage Classes:**

Name	Provisioner	Parameters	Created
standard	k8s.io/minikube-hostpath	-	3 days ago

Screenshot 2: Cluster

- Cluster Role Bindings:**

Name	Created
kubernetes-dashboard	3 minutes ago
kubeadm-node-proxier	3 days ago
minikube-rbac	3 days ago
storage-provisioner	3 days ago
kubeadm-cluster-admins	3 days ago
kubeadm-get-nodes	3 days ago
kubeadm-kublet-bootstrap	3 days ago
kubeadm-node-autoapprove-bootstrap	3 days ago
kubeadm-node-autoapprove-certificate-rotation	3 days ago
system:coredns	3 days ago
- Cluster Roles:**

Name
system:coredns

Bước 6: Kiểm tra và giám sát deployment:

Kiểm tra Deployment:

Dùng lệnh sau để xem danh sách các pods đang chạy trong cluster:
kubectl get pods

Dùng lệnh sau để xem các services đã được tạo:
kubectl get services

```

lunox@lunox-VirtualBox:~$ kubectl get pods
NAME                               READY   STATUS        RESTARTS   AGE
ci-cd-lab-app-64654489fd-hxk9h   0/1     ImagePullBackOff   0          2m46s
ci-cd-lab-app-64654489fd-rqw8q   0/1     ImagePullBackOff   0          2m46s
lunox@lunox-VirtualBox:~$ kubectl get services
NAME            TYPE      CLUSTER-IP    EXTERNAL-IP   PORT(S)    AGE
ci-cd-lab-service   NodePort    10.101.170.232 <none>       80:30311/TCP  3m2s
kubernetes       ClusterIP   10.96.0.1    <none>       443/TCP    122m
lunox@lunox-VirtualBox:~$ minikube ip
192.168.49.2
lunox@lunox-VirtualBox:~$ 

```

Truy cập ứng dụng đã deploy:

<http://192.168.49.2:32680/>

Ta thấy không truy cập được web theo URL đã tạo. Đó có thể là do cấu hình quyền truy cập của Kubernetes không đúng. Thêm các bước sau:

Tạo secret:

#Bash

```

kubectl create secret docker-registry my-registry-key \
--docker-server=https://index.docker.io/v1/ \
--docker-username=<your-dockerhub-username> \
--docker-password=<your-dockerhub-password> \
--docker-email=<your-dockerhub-email>

```

```

lunox@lunox-VirtualBox:~$ kubectl create secret docker-registry my-registry-key \
--docker-server=https://index.docker.io/v1/ \
--docker-username=nguyenle2k3 \
--docker-password=dckr_pat_████████████████████████████████████████M7Ag \
--docker-email=nguyenle████████████████████████████████████████com
secret/my-registry-key created

```

Thêm vào file deployment.yaml

imagePullSecrets: - name: my-registry-key

```

k8s > / deployment.yaml
 1  apiVersion: apps/v1
 2  kind: Deployment
 3  metadata:
 4    name: ci-cd-lab-app
 5  spec:
 6    replicas: 2
 7    selector:
 8      matchLabels:
 9        app: ci-cd-lab-app
10    template:
11      metadata:
12        labels:
13          app: ci-cd-lab-app
14      spec:
15        containers:
16          - name: ci-cd-lab-app
17            image: nguyenle2k3/ci-cd-lab-app:latest
18            ports:
19              - containerPort: 8080
20            imagePullSecrets:
21              - name: my-registry-key
22  ---
23  apiVersion: v1
24  kind: Service
25  metadata:
26    name: ci-cd-lab-service
27  spec:
28    selector:
29      app: ci-cd-lab-app
30    ports:
31      - protocol: TCP
32        port: 80
33        targetPort: 8080
34    type: NodePort

```

Rebuild and open web:

The screenshot shows the Jenkins Pipeline Overview for a job named 'ci-cd-lab-app'. The pipeline has a total of 8 stages: Start, Checkout SCM, Clone Repository, Build Docker Im..., Push Docker Im..., Deploy to Kube..., Post Actions, and End. All stages are marked with a green circle, indicating they have been successfully completed. The 'Details' panel on the right provides information about the build: it was manually run by Lunox, started 20 seconds ago, queued for 3 ms, and took 20 seconds and counting.

```

lunox@lunox-VirtualBox:~$ kubectl get svc
NAME           TYPE      CLUSTER-IP     EXTERNAL-IP   PORT(S)      AGE
ci-cd-lab-service  NodePort  10.101.119.189  <none>       80:32680/TCP  25m
kubernetes      ClusterIP  10.96.0.1    <none>       443/TCP     40m
lunox@lunox-VirtualBox:~$ minikube service --all
|-----|-----|-----|-----|-----|
| NAMESPACE | NAME      | TARGET PORT | URL          | |
|---|---|---|---|---|
| default   | ci-cd-lab-service | 80          | http://192.168.49.2:32680 |
|-----|-----|-----|-----|-----|

```



Thành công.

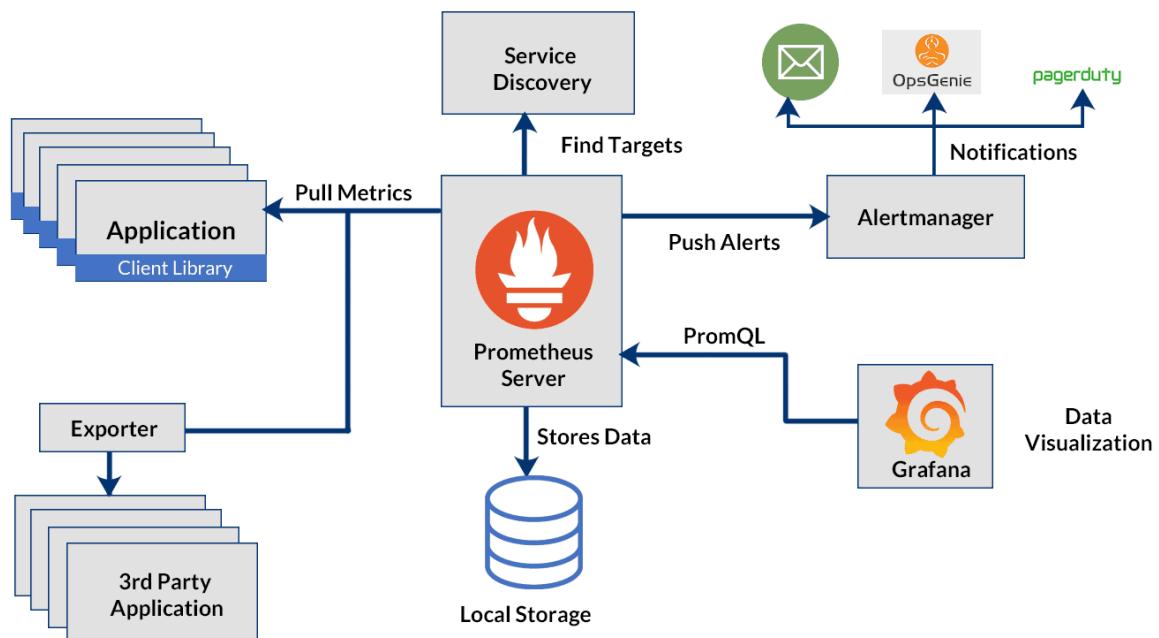
Khi có cập nhật về app. Cần sử dụng jenkins để build lại ứng dụng. Chờ khi build thành công thì sử dụng lệnh `kubectl rollout restart deployment ci-cd-lab-app` để update deployment

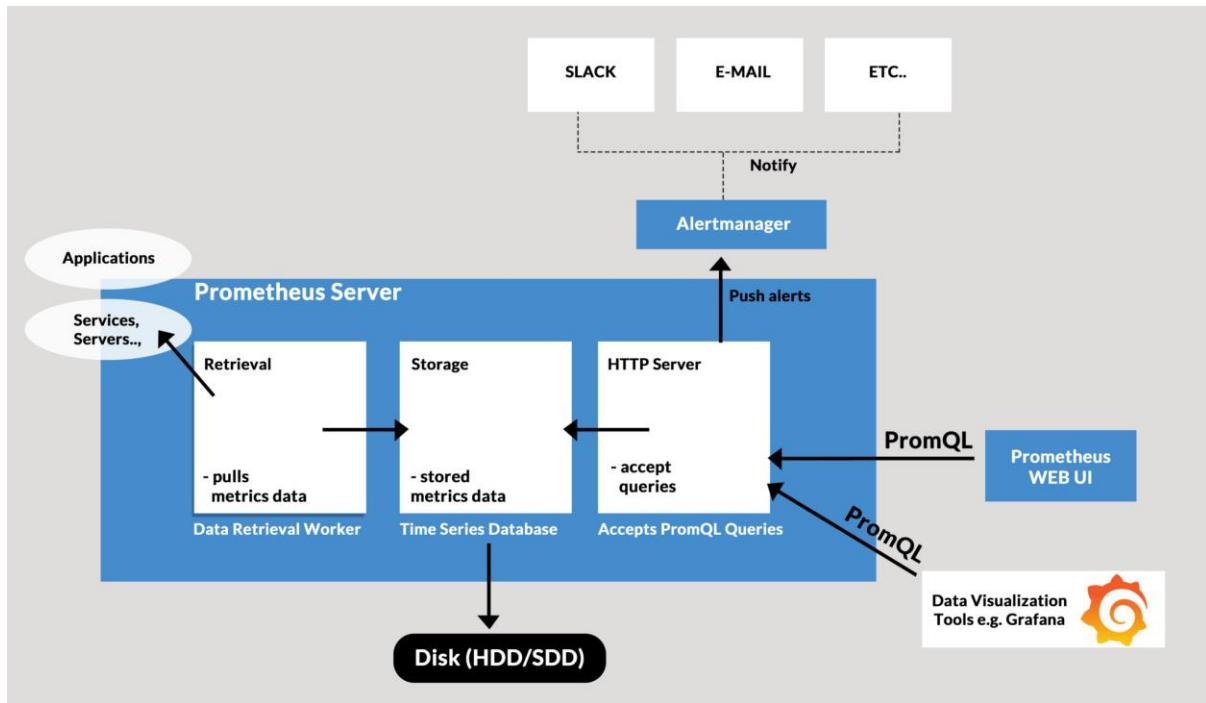
Cài đặt và sử dụng Prometheus và Grafana

<https://www.fosstechnix.com/install-prometheus-and-grafana-on-ubuntu-24-04/>

<https://www.fosstechnix.com/kubernetes-cluster-monitoring-with-prometheus-and-grafana/>

<https://k21academy.com/docker-kubernetes/prometheus-grafana-monitoring/>





Triển khai môi trường giám sát trên Kubernetes:

Cài đặt Helm từ gói apt:

```
curl https://baltocdn.com/helm/signing.asc | gpg --dearmor | sudo tee /usr/share/keyrings/helm.gpg > /dev/null
```

```
sudo apt-get install apt-transport-https --yes
```

```
echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/helm.gpg] https://baltocdn.com/helm/stable/debian/ all main" | sudo tee /etc/apt/sources.list.d/helm-stable-debian.list
```

```
sudo apt-get update
```

```
sudo apt-get install helm
```

```
lunox@lunox-VirtualBox:~$ curl https://baltocdn.com/helm/signing.asc | gpg --dearmor | sudo tee /usr/share/keyrings/helm.gpg > /dev/null
% Total    % Received % Xferd  Average Speed   Time     Time   Current
          Dload  Upload   Total  Spent   Left  Speed
100  1699  100  1699    0      0  2550      0 --:--:-- --:--:-- 2554[sudo] password for lunox:
```

```
lunox@lunox-VirtualBox:~$ sudo apt-get install apt-transport-https --yes
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
apt-transport-https
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
```

```
[Reading database ... 151930 files and directories currently installed.]  
Preparing to unpack .../apt-transport-https_2.7.14build2_all.deb ...  
Unpacking apt-transport-https (2.7.14build2) ...  
Setting up apt-transport-https (2.7.14build2) ...  
lunox@lunox-VirtualBox:~$ echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/helm.gpg] https://baltocdn.com/helm/stable/debian/ all main"  
deb [arch=amd64 signed-by=/usr/share/keyrings/helm.gpg] https://baltocdn.com/helm/stable/debian/ all main  
lunox@lunox-VirtualBox:~$ sudo apt-get update  
Hit:1 http://vn.archive.ubuntu.com/ubuntu noble InRelease  
Hit:2 http://vn.archive.ubuntu.com/ubuntu noble-updates InRelease  
Hit:3 https://download.docker.com/linux/ubuntu noble InRelease  
Hit:4 http://vn.archive.ubuntu.com/ubuntu noble-backports InRelease  
Hit:5 https://packages.microsoft.com/repos/code stable InRelease  
Ign:6 https://pkg.jenkins.io/debian-stable binary/ InRelease  
Hit:7 https://pkg.jenkins.io/debian-stable binary/ Release  
Hit:9 http://security.ubuntu.com/ubuntu noble-security InRelease  
Get:10 https://baltocdn.com/helm/stable/debian all InRelease [7,652 B]  
Get:11 https://baltocdn.com/helm/stable/debian all/main amd64 Packages [4,360 B]  
Fetched 12.0 kB in 1s (12.2 kB/s)  
Reading package lists... Done  
lunox@lunox-VirtualBox:~$ sudo apt-get install helm  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
The following NEW packages will be installed:  
  helm  
0 upgraded, 1 newly installed, 0 to remove and 1 not upgraded.  
Need to get 16.6 MB of archives.  
After this operation, 52.5 MB of additional disk space will be used.
```

Prometheus cần có một kho lưu trữ liên tục (persistent storage). Sử dụng NFS Server. Tuy nhiên với nội dung lab thì ứng dụng là stateless-không cần lưu trữ dữ liệu giữa các lần chạy nên bỏ qua phần cấu hình NFS server.

Cài đặt Prometheus:

Add repository:

```
helm repo add prometheus-community https://prometheus-community.github.io/helm-charts  
helm repo add stable https://charts.helm.sh/stable
```

```
lunox@lunox-VirtualBox:~$ helm repo add prometheus-community https://prometheus-community.github.io/helm-charts 2>/dev/null  
"prometheus-community" has been added to your repositories  
lunox@lunox-VirtualBox:~$ helm repo add stable https://charts.helm.sh/stable 2>/dev/null  
"stable" has been added to your repositories  
lunox@lunox-VirtualBox:~$ 
```

Update Helm repositories:

```
helm repo update
```

```
lunox@lunox-VirtualBox:~$ helm repo update 2>/dev/null  
Hang tight while we grab the latest from your chart repositories...  
...Successfully got an update from the "prometheus-community" chart repository  
...Successfully got an update from the "stable" chart repository  
Update Complete. *Happy Helming!*
```

Cài đặt Prometheus Kubernetes:

```
helm install prometheus prometheus-community/kube-prometheus-stack
```

```
lunox@lunox-VirtualBox: ~$ helm install prometheus prometheus-community/kube-prometheus-stack 2>/dev/null
NAME: prometheus
LAST DEPLOYED: Wed Aug 21 09:53:44 2024
NAMESPACE: default
STATUS: deployed
REVISION: 1
NOTES:
kube-prometheus-stack has been installed. Check its status by running:
  kubectl --namespace default get pods -l "release=prometheus"

Visit https://github.com/prometheus-operator/kube-prometheus for instructions on how to create & configure Alertmanager and Prometheus instances using the Operator.
lunox@lunox-VirtualBox: ~$ kubectl -n default get pods -l "release=prometheus"
NAME                           READY   STATUS    RESTARTS   AGE
prometheus-kube-prometheus-operator-5d8bbcc8f8-6sjmd   1/1     Running   0          27s
prometheus-kube-state-metrics-688d6b5b8-95z48        1/1     Running   0          27s
prometheus-prometheus-node-exporter-x5pn7            1/1     Running   0          27s
lunox@lunox-VirtualBox: ~$
```

Chuyển tiếp cổng Kubernetes Prometheus:

`kubectl port-forward deployment/prometheus-grafana 3000`

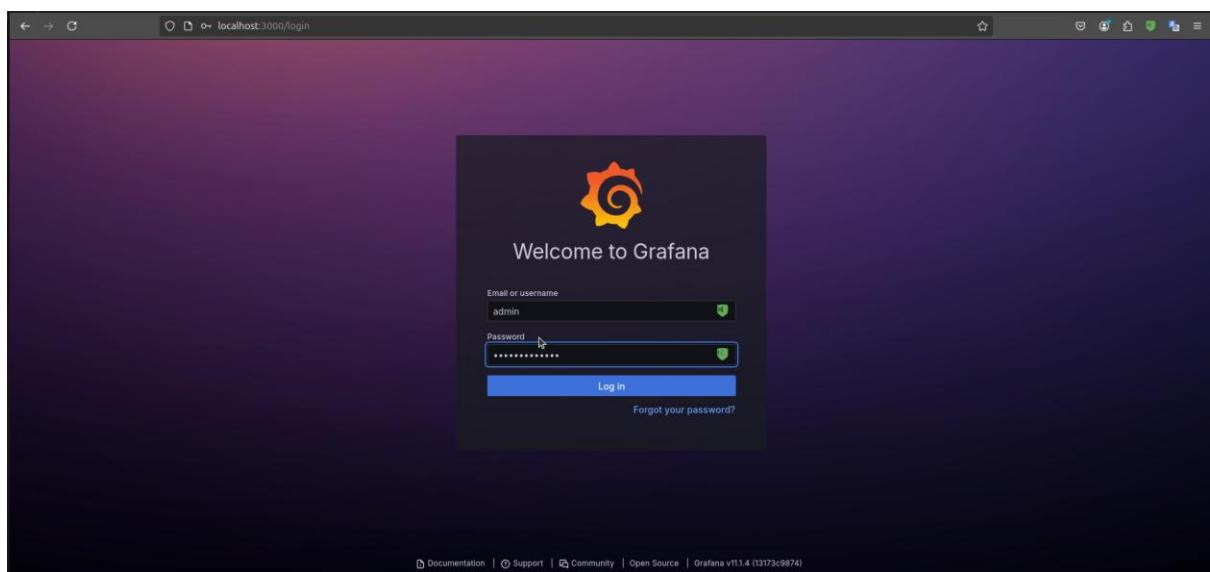
```
lunox@lunox-VirtualBox: ~$ kubectl port-forward deployment/prometheus-grafana 3000
Forwarding from 127.0.0.1:3000 -> 3000
Forwarding from [::1]:3000 -> 3000
```

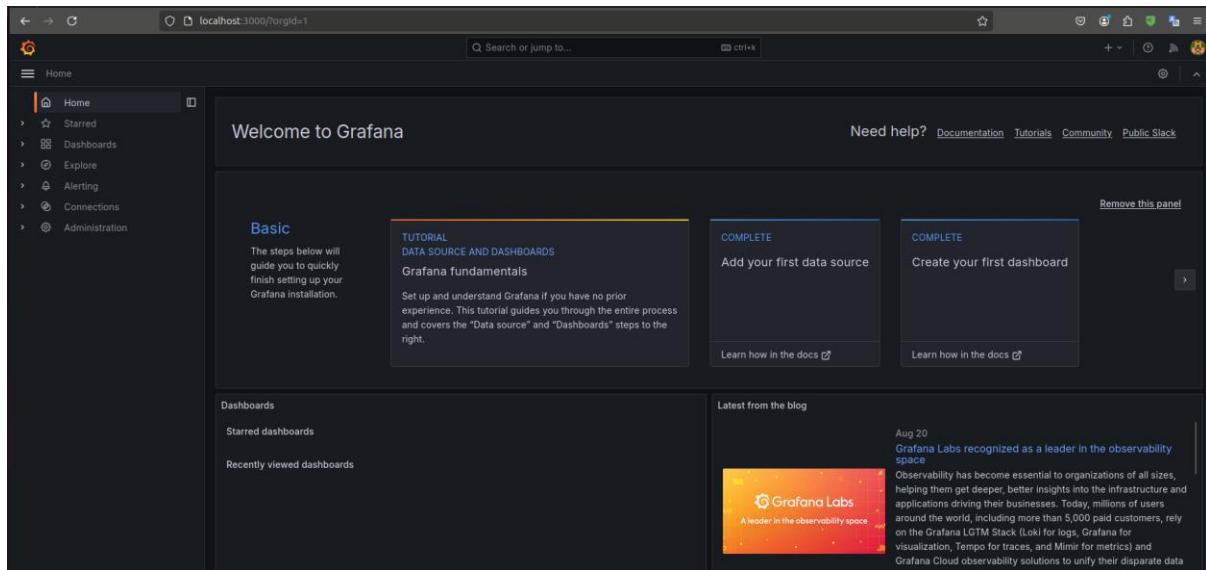
Đăng nhập web-gui grafana:

<http://localhost:3000/login>

username: admin

password: prom-operator





Cấu hình Prometheus để giám sát ứng dụng:

Thêm job vào Prometheus:

File **values.yaml** là tệp cấu hình chính khi bạn cài đặt các Helm chart. Khi cài đặt Prometheus và Grafana bằng Helm, tệp này có thể được tìm thấy trong Helm chart của ứng dụng đó.

Lấy file values.yaml mặc định của Helm chart:

```
lunox@lunox-VirtualBox:~$ ls
Desktop Documents Downloads Music Pictures Public snap Templates Videos
lunox@lunox-VirtualBox:~$ helm show values prometheus-community/kube-prometheus-stack > values.yaml 2>/dev/null
lunox@lunox-VirtualBox:~$ ls
Desktop Documents Downloads Music Pictures Public snap Templates values.yaml Videos
lunox@lunox-VirtualBox:~$
```

Chỉnh sửa values.yaml:

nano values.yaml