Môi trường thực hiện:

Oracle VirtualBox 7.0.14.

Ubuntu Desktop 24.04 LTS (Noble Numbat).

RAM: 16GB

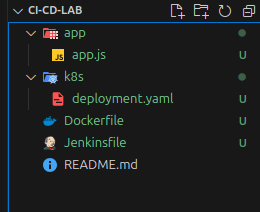
Core: 4

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.

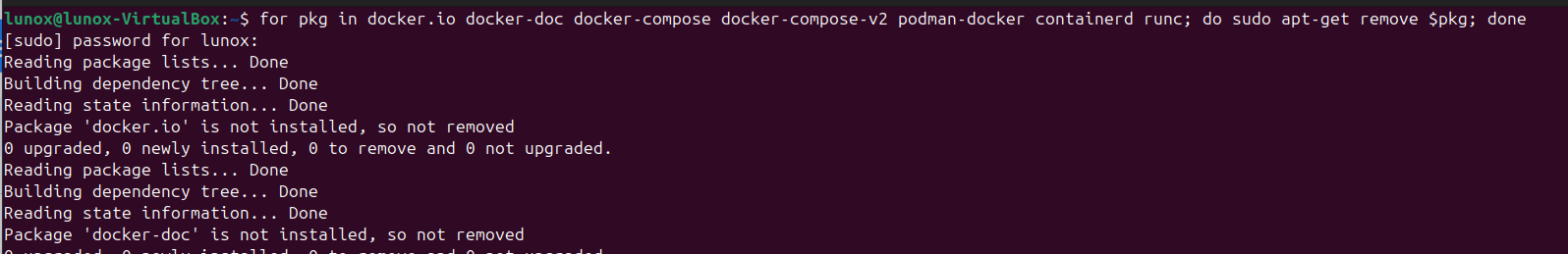


Bước 2: Cài docker - Tạo Dockerfile

# Cài đặt docker engine.

Gỡ 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



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:

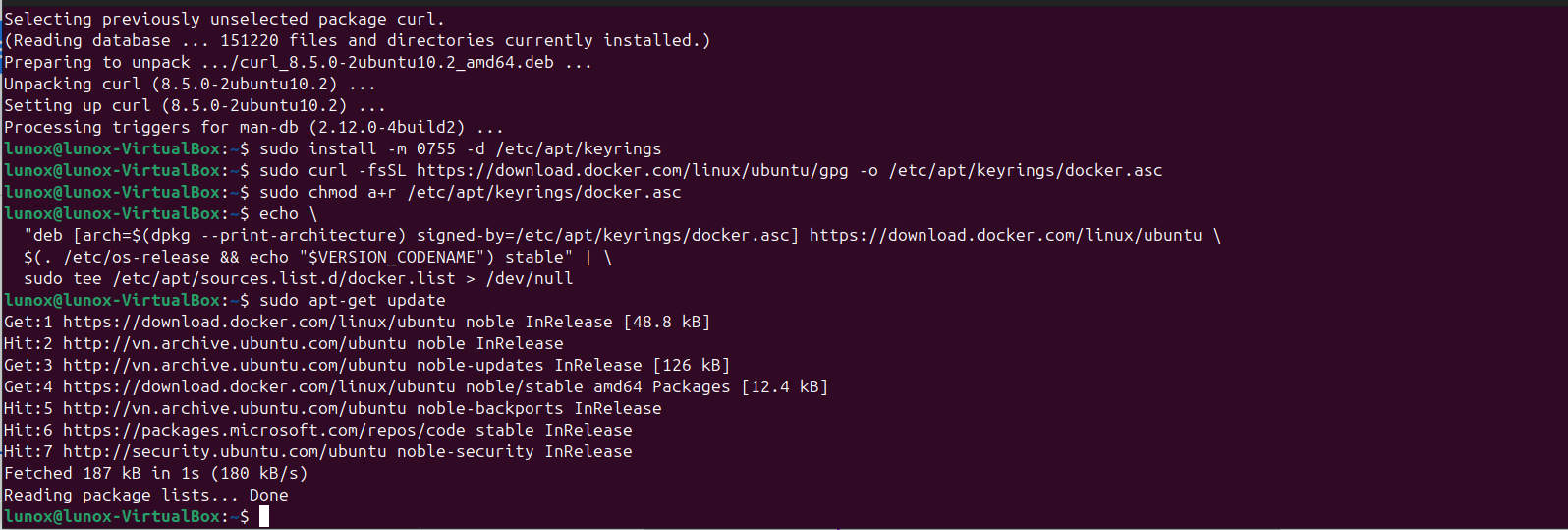
echo \

"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" | \

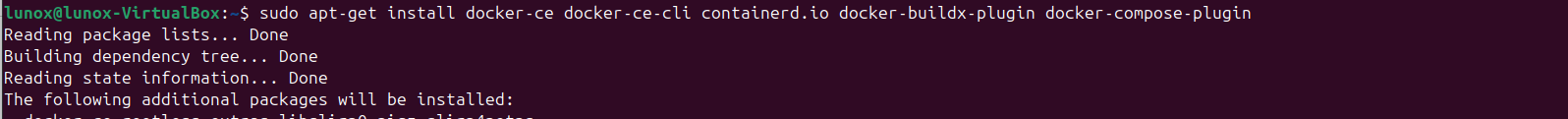
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

sudo apt-get update



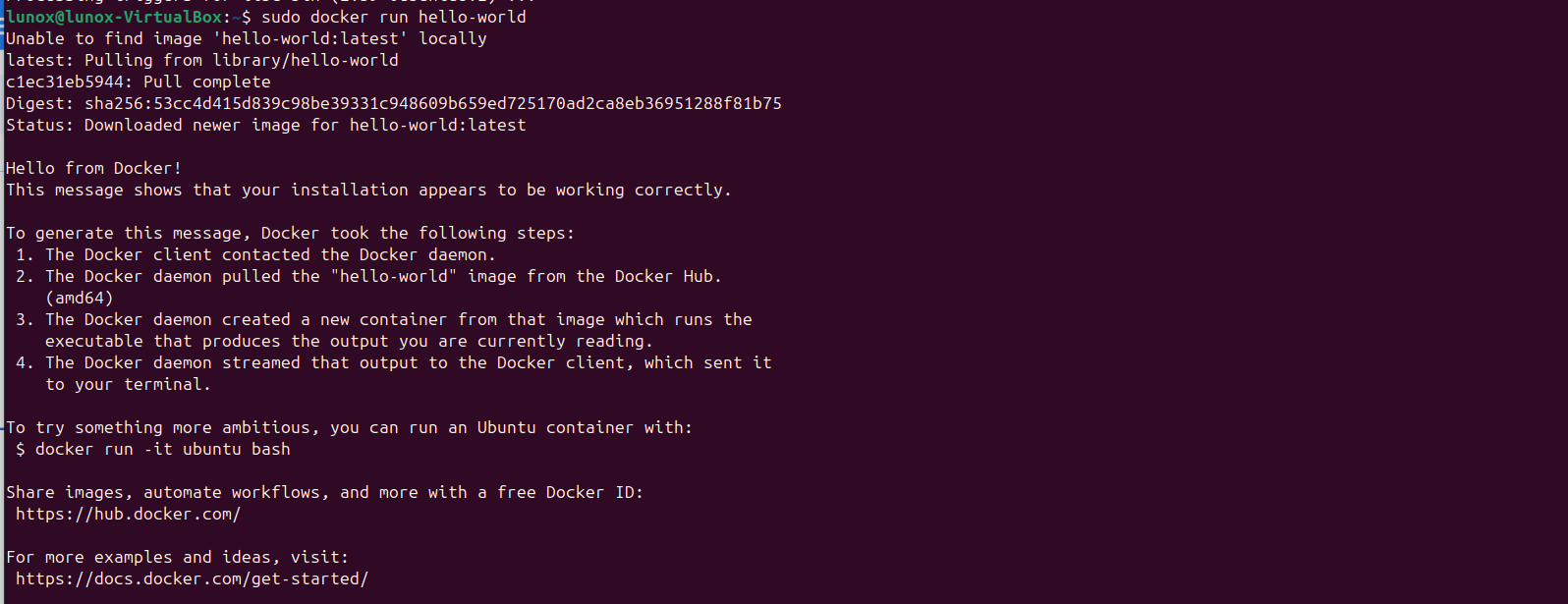
Cài đặt docker engine:

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



Test docker:

sudo docker run hello-world



Docker cài đặt thành công.

Cấu hình chạy docker không cần quyền root:

sudo groupadd docker

sudo usermod -aG docker $USER

newgrp docker

Configure docker chạy với systemd:

sudo systemctl enable docker.service

sudo systemctl enable containerd.service

# Tạo 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/

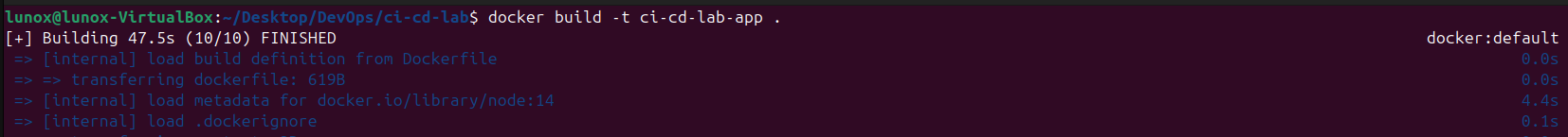
EXPOSE 8080

CMD ["node", "app/app.js"]

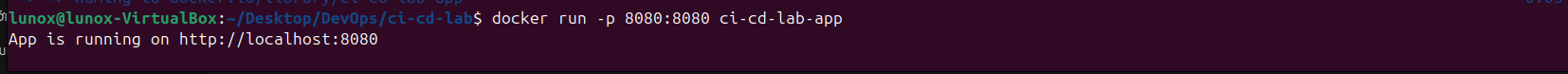


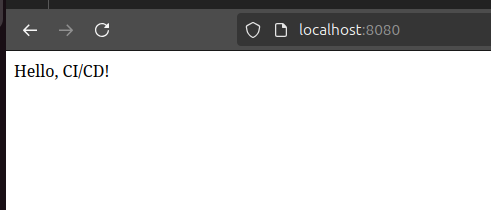
# Build and test:

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



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





Bước 3: Cài đặt và cấu hình Jenkins pipeline.

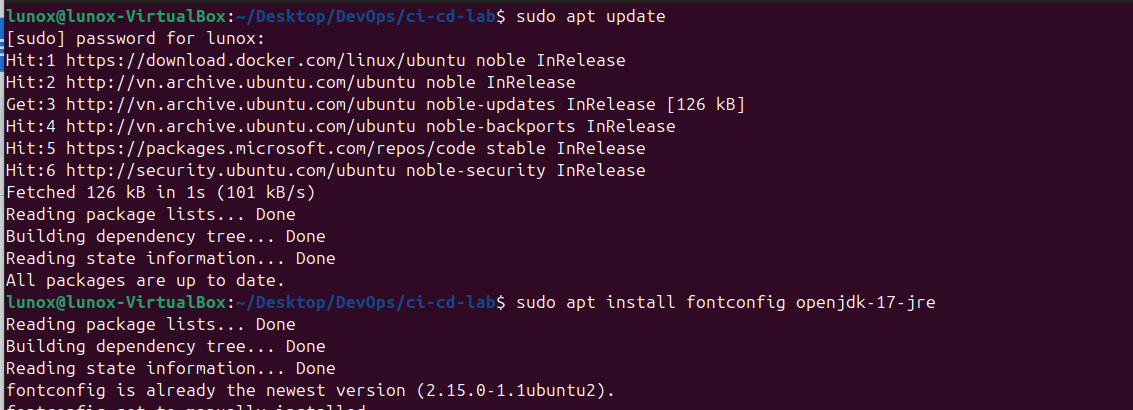
# Cài đặt Jenkins.

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

Cài đặt java cho jenkins:

sudo apt update

sudo apt install fontconfig openjdk-17-jre



Cài đặt jenkins:

sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \

https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key

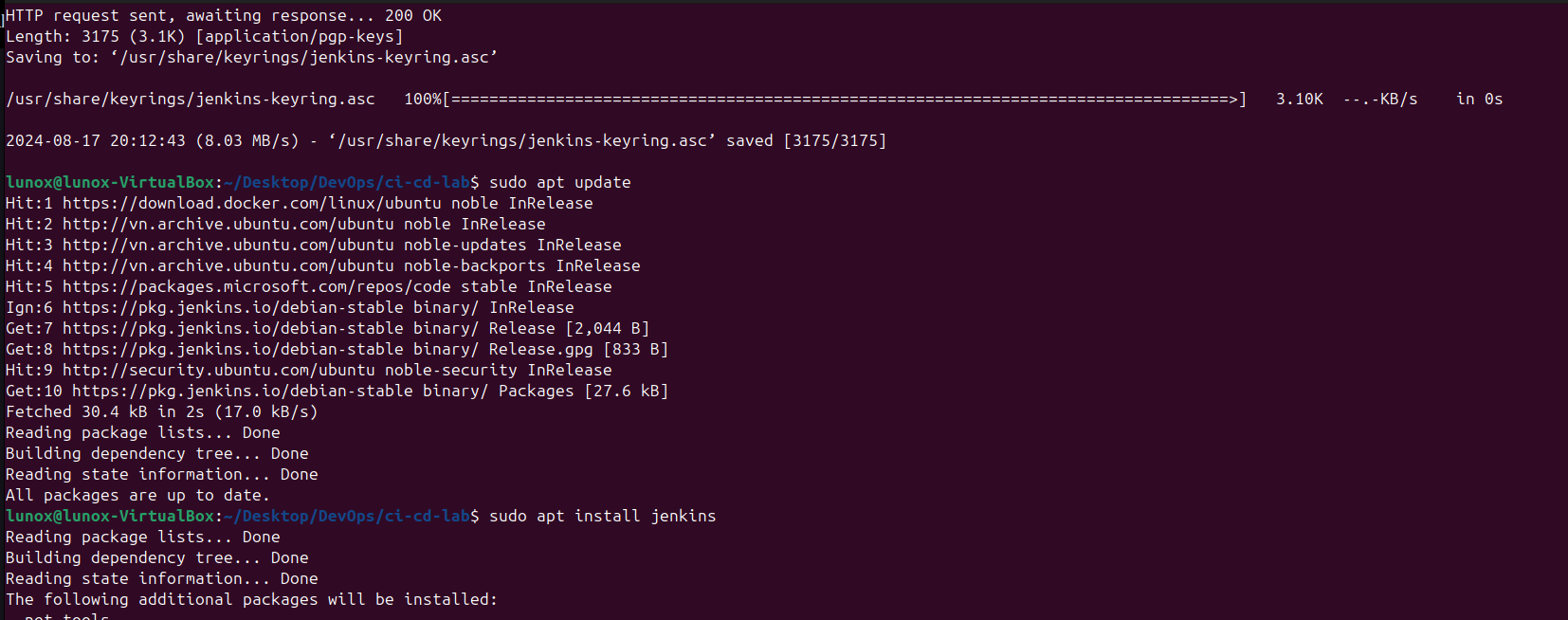
echo "deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc]" \

https://pkg.jenkins.io/debian-stable binary/ | sudo tee \

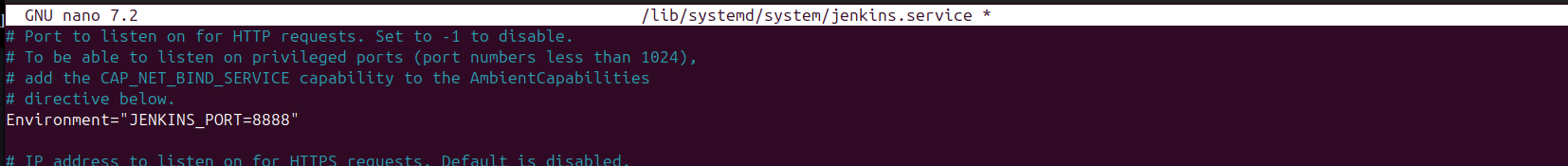
/etc/apt/sources.list.d/jenkins.list > /dev/null

sudo apt-get update

sudo apt-get install jenkins

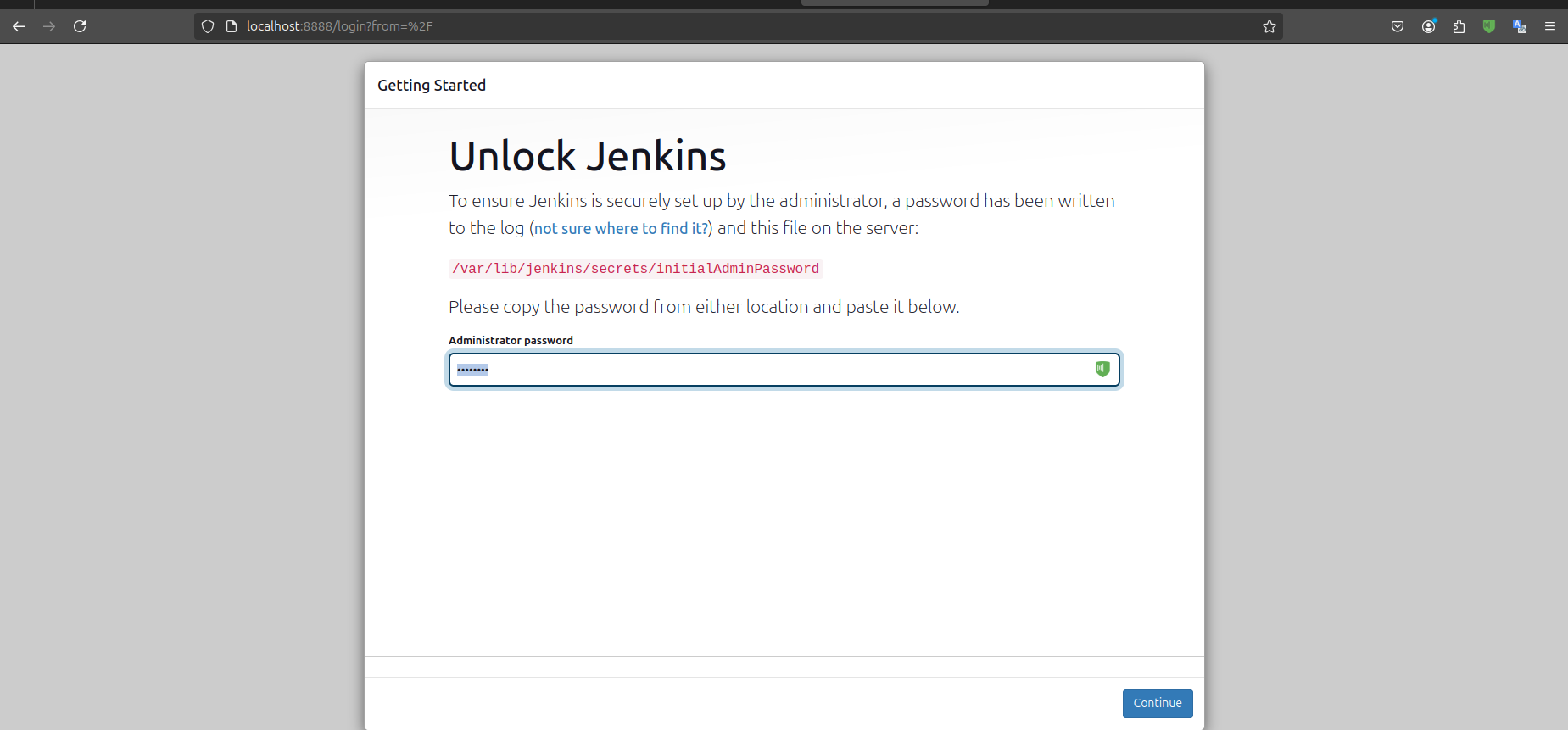


Thay đổi cổng mặc định của jenkins:

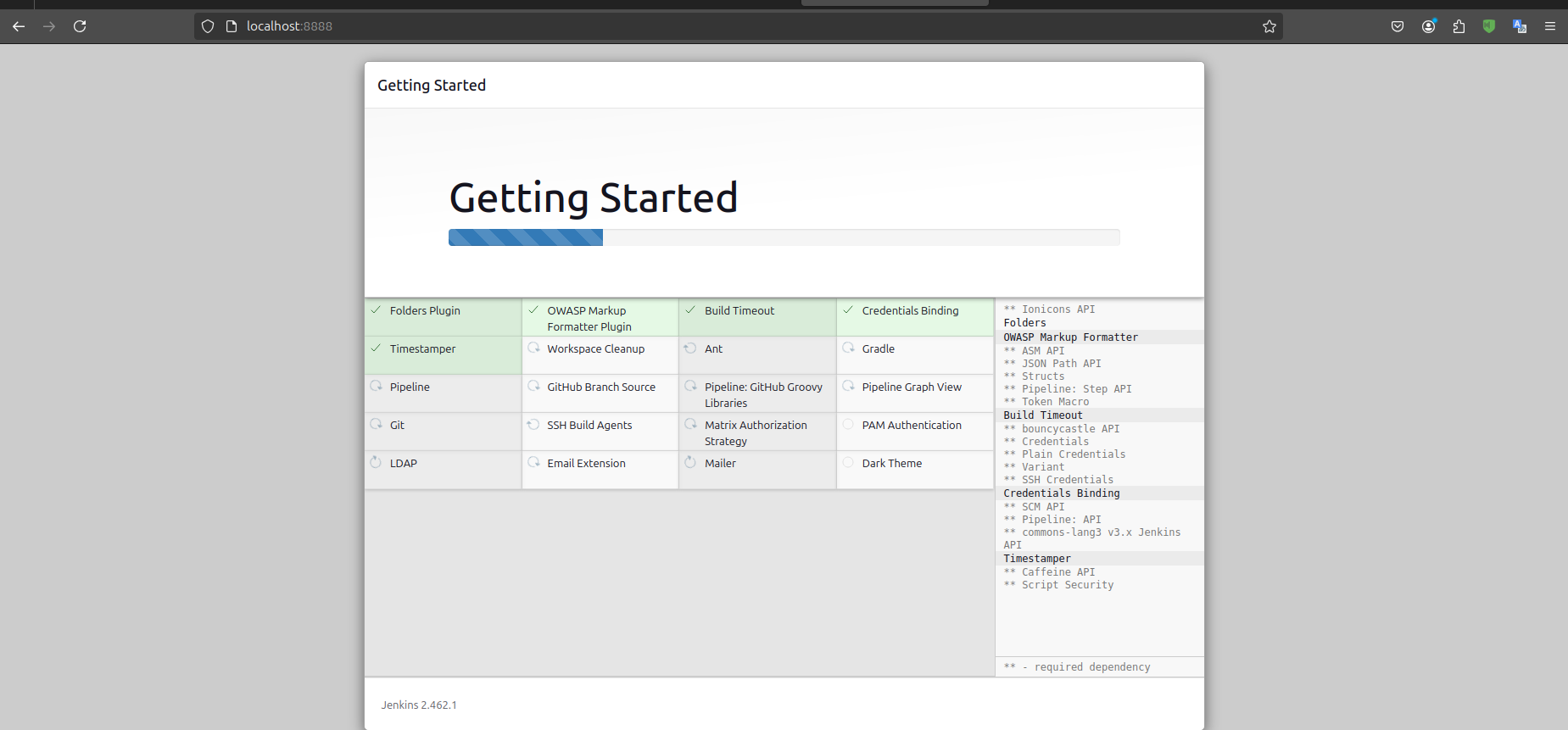


Truy cập web GUI của jenkins qua url: <http://localhost:8888>

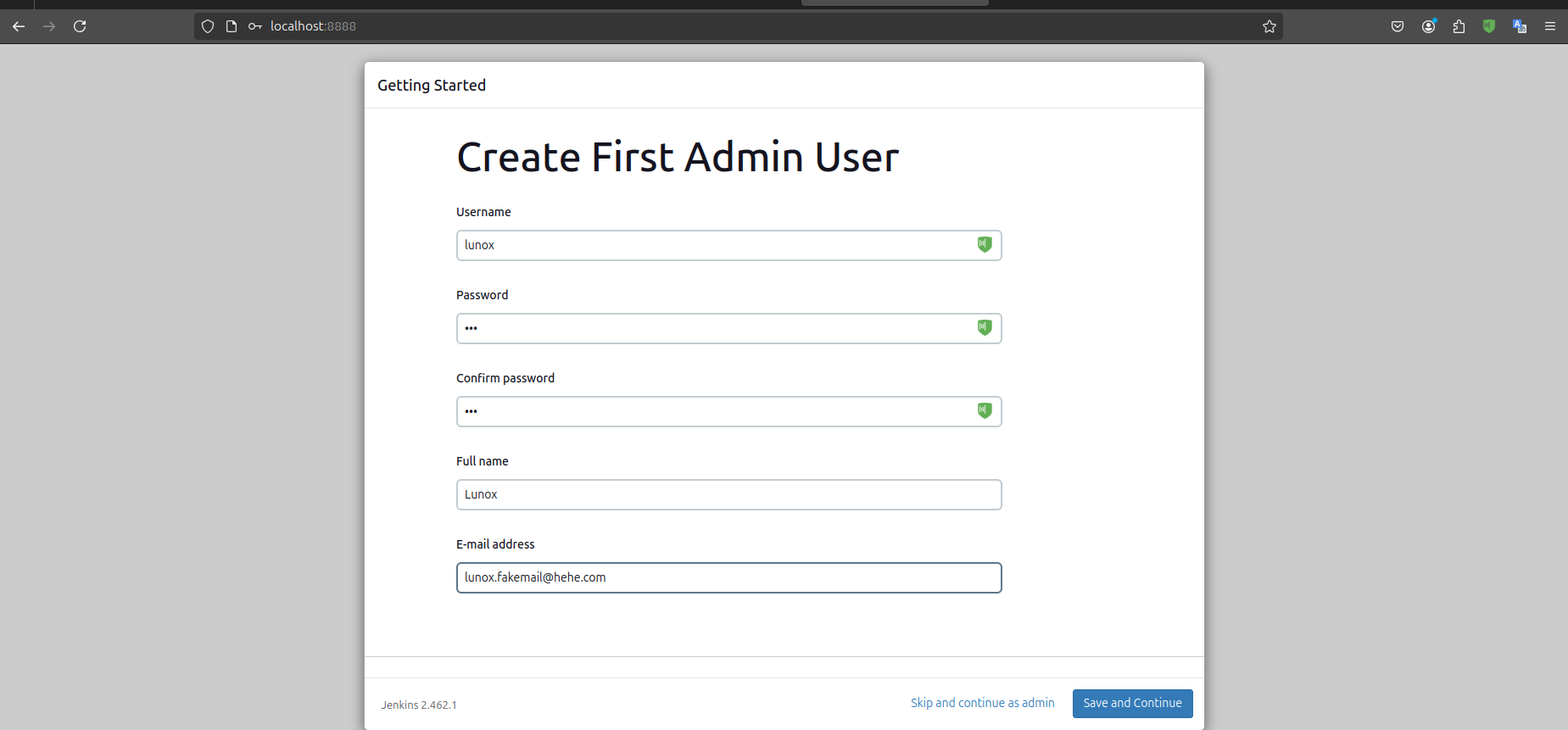
Mật khẩu đăng nhập lần đầu lấy ở: /var/lib/jenkins/secrets/initialAdminPassword

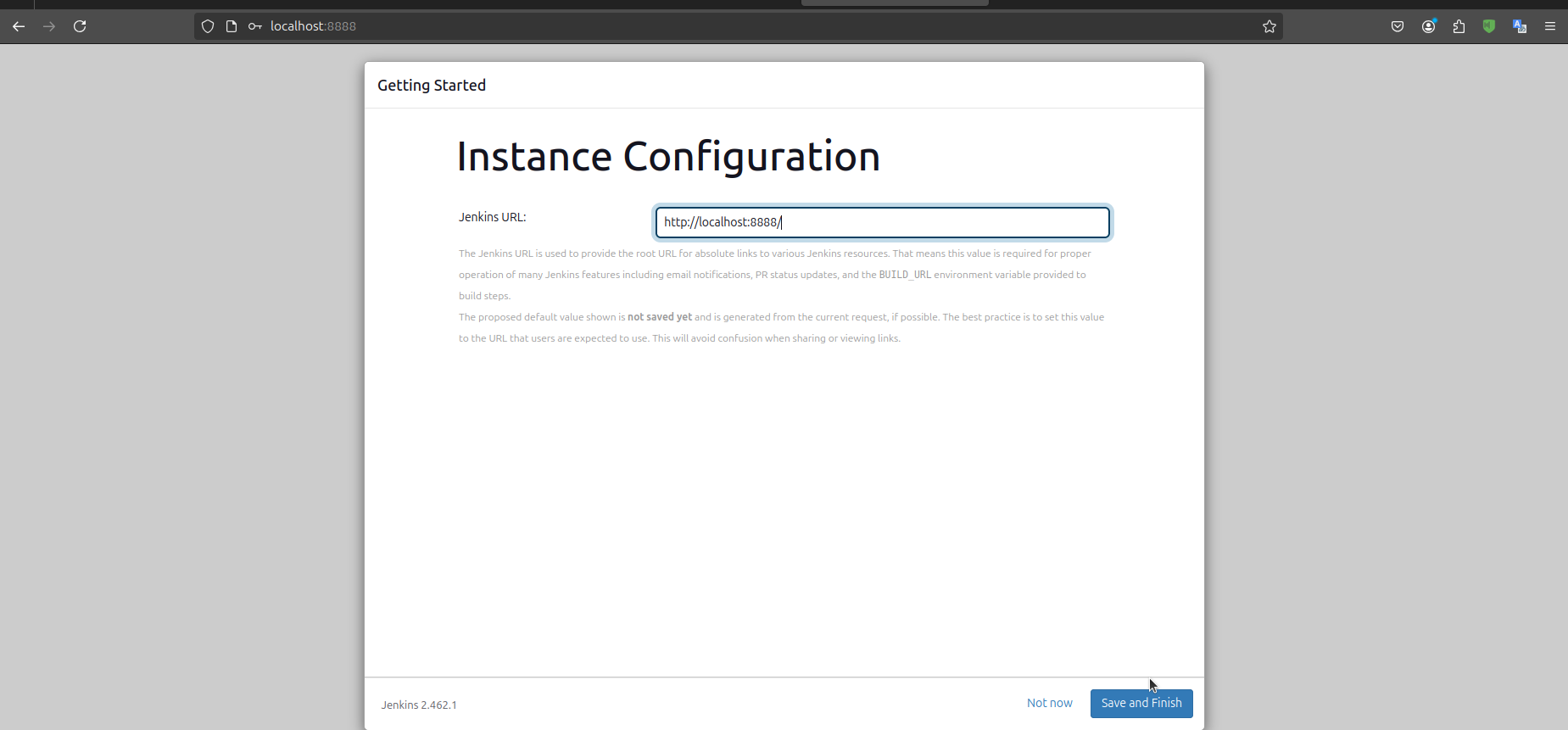


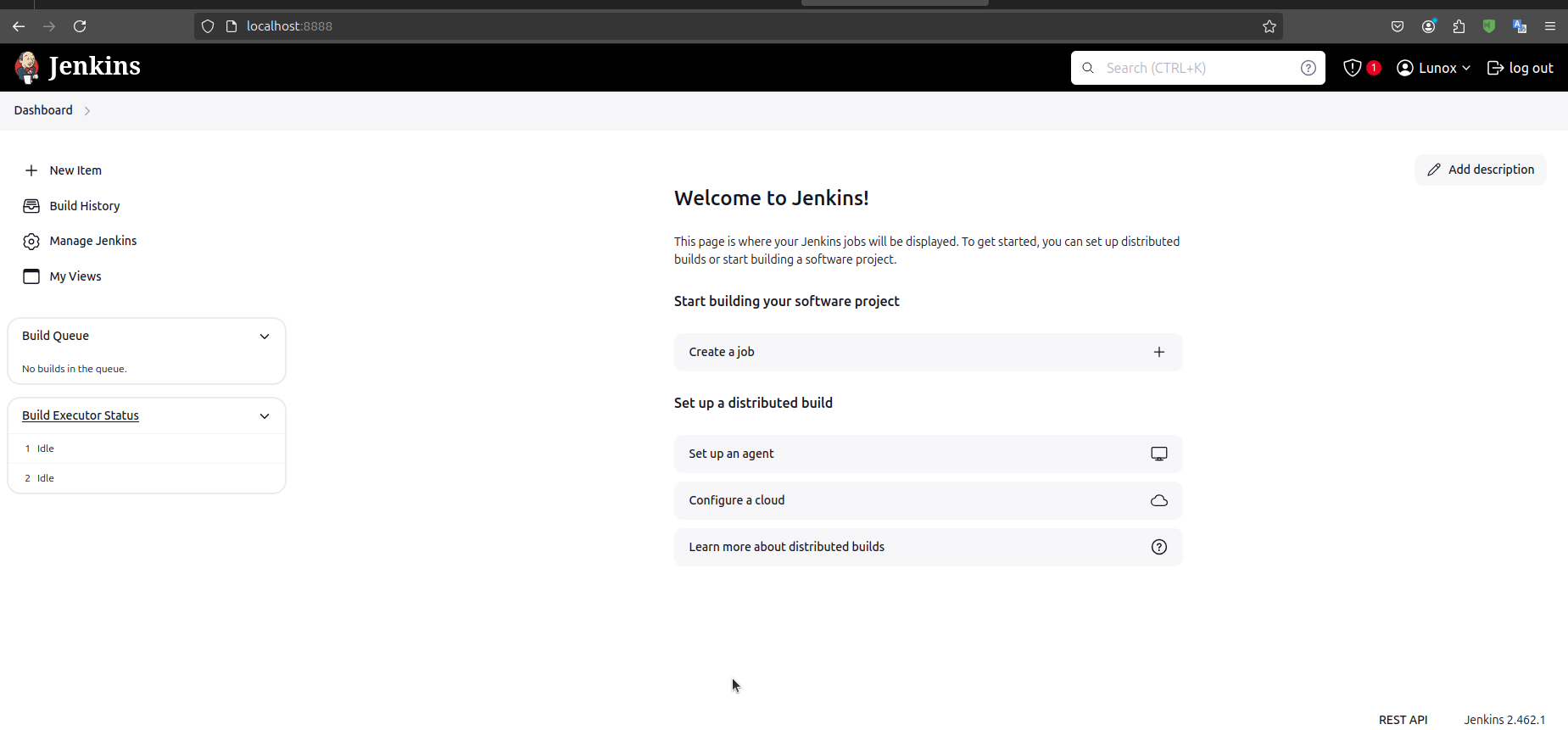
Chọn install suggestion plugins để tiếp tục



Tạo tài khoản jenkins để đăng nhập lần sau:

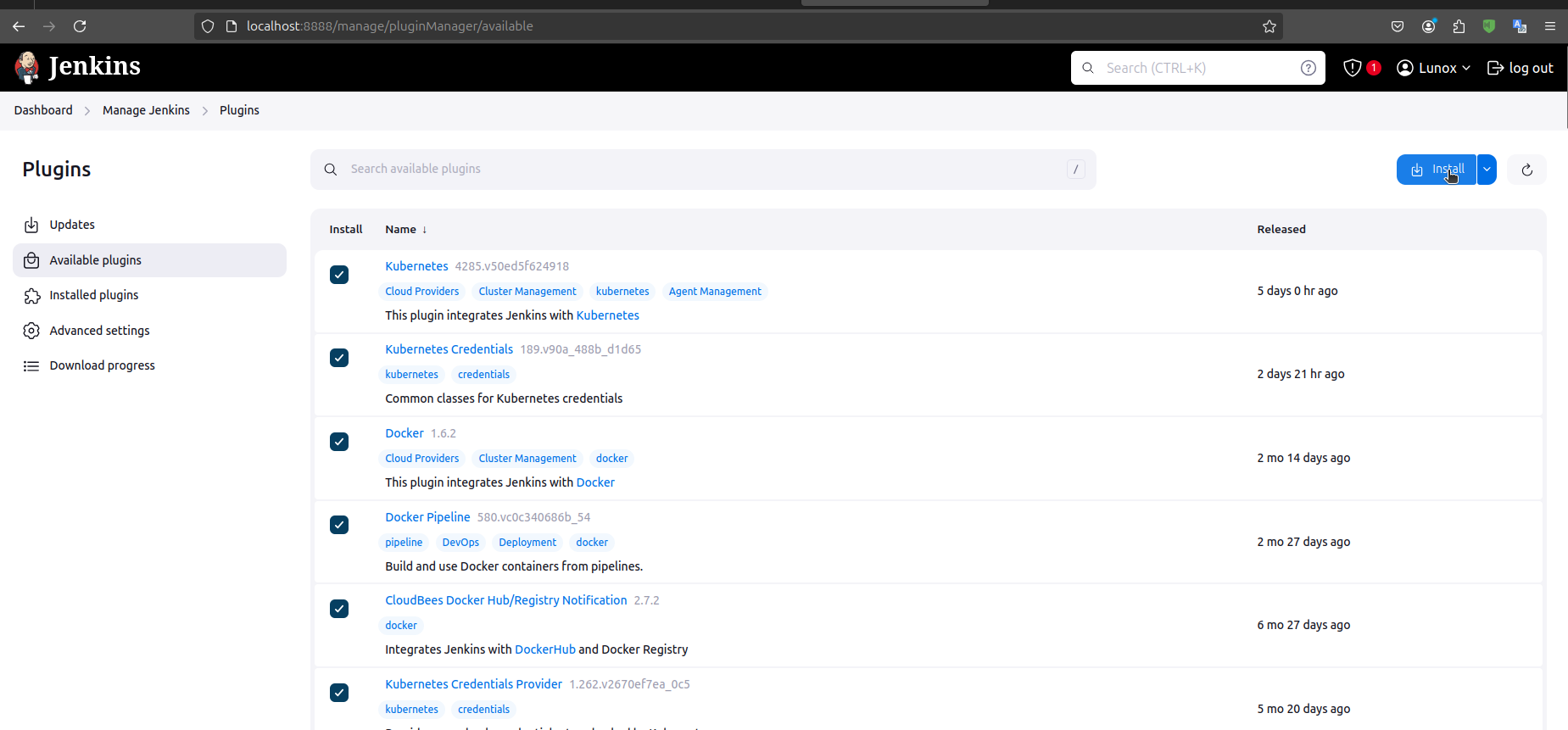


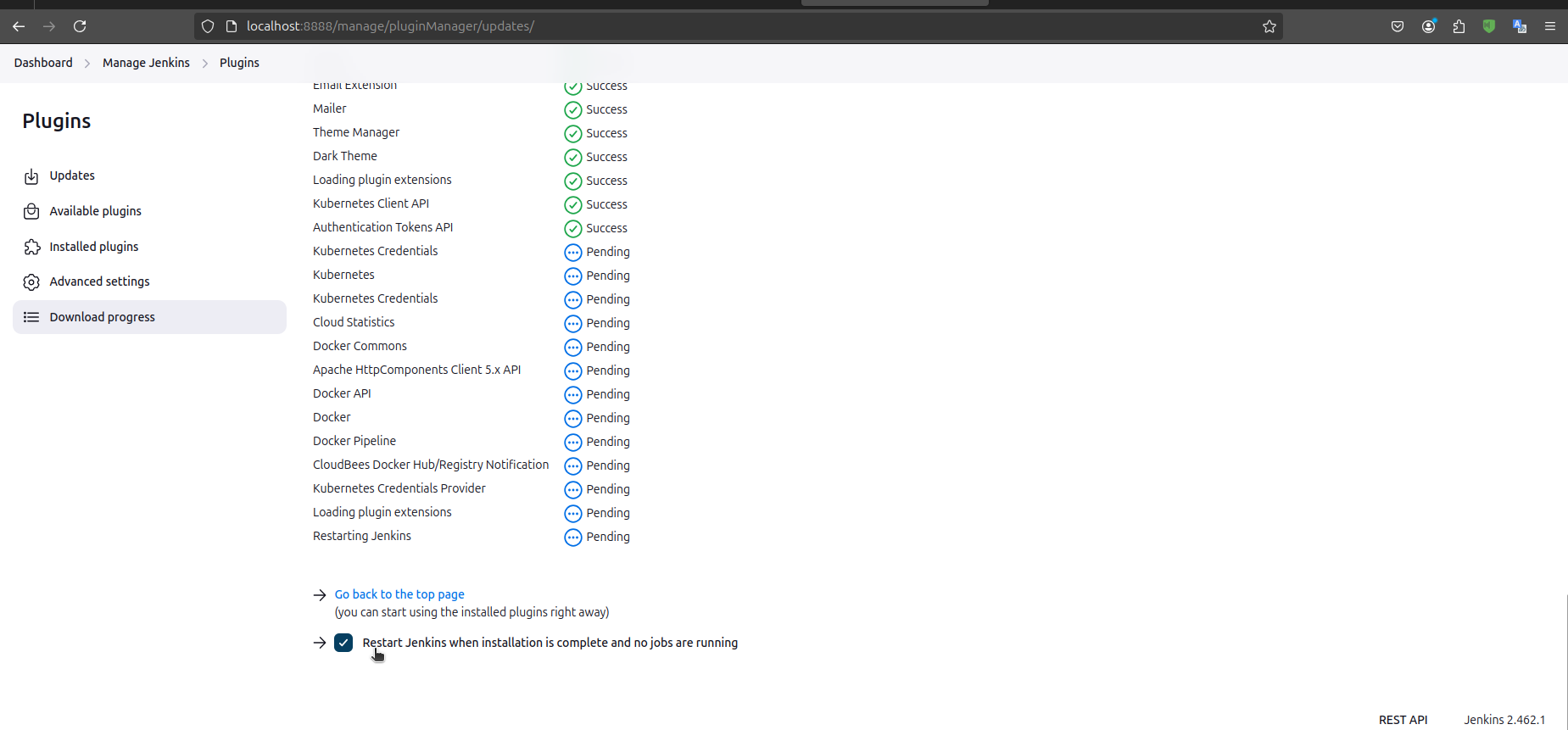




Cài đặt thành công.

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





# Thêm các credential:

## Gitlab credential:

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

Chọn Add new token.



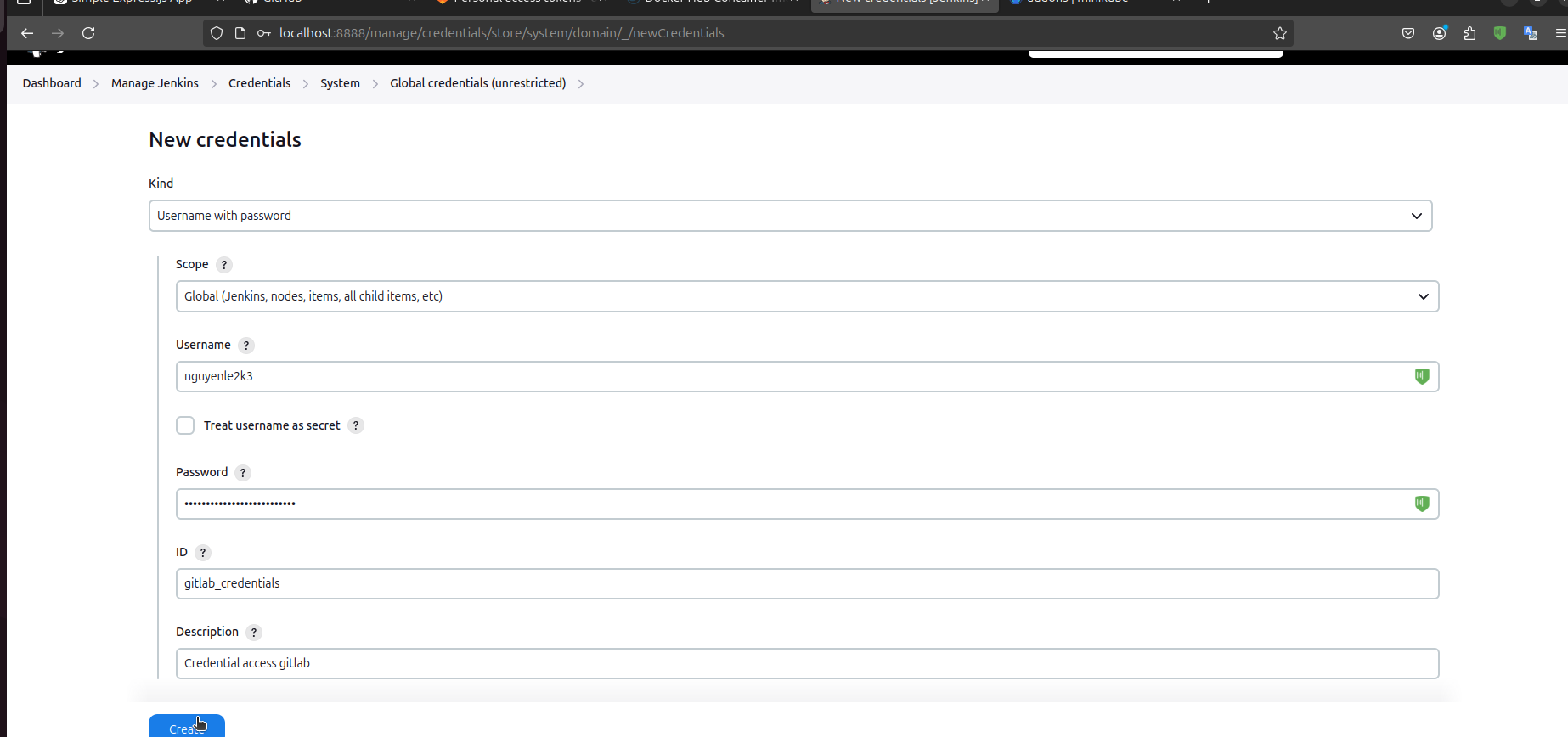
copy token và lưu lại trong một file bí mật.

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

Add credentials:

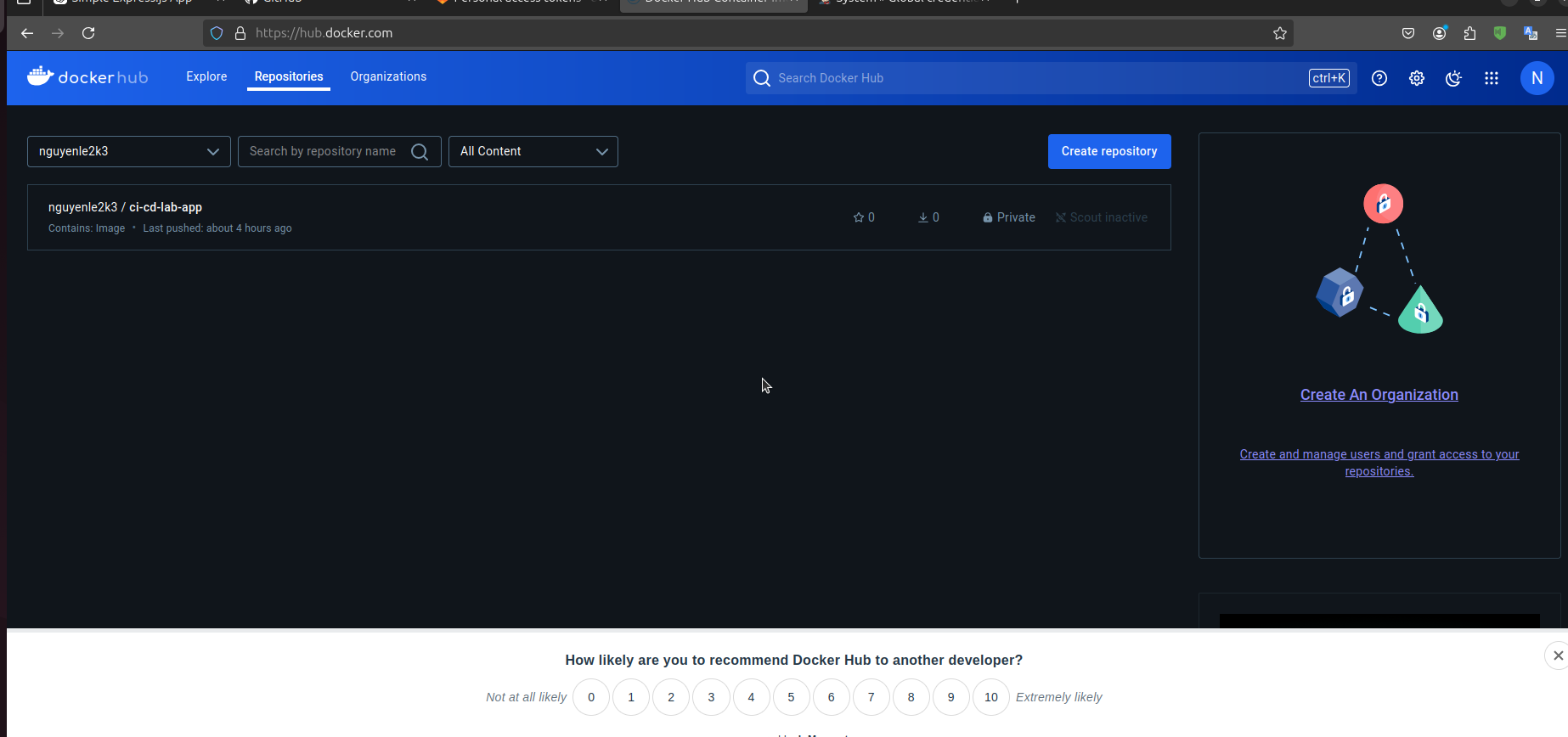


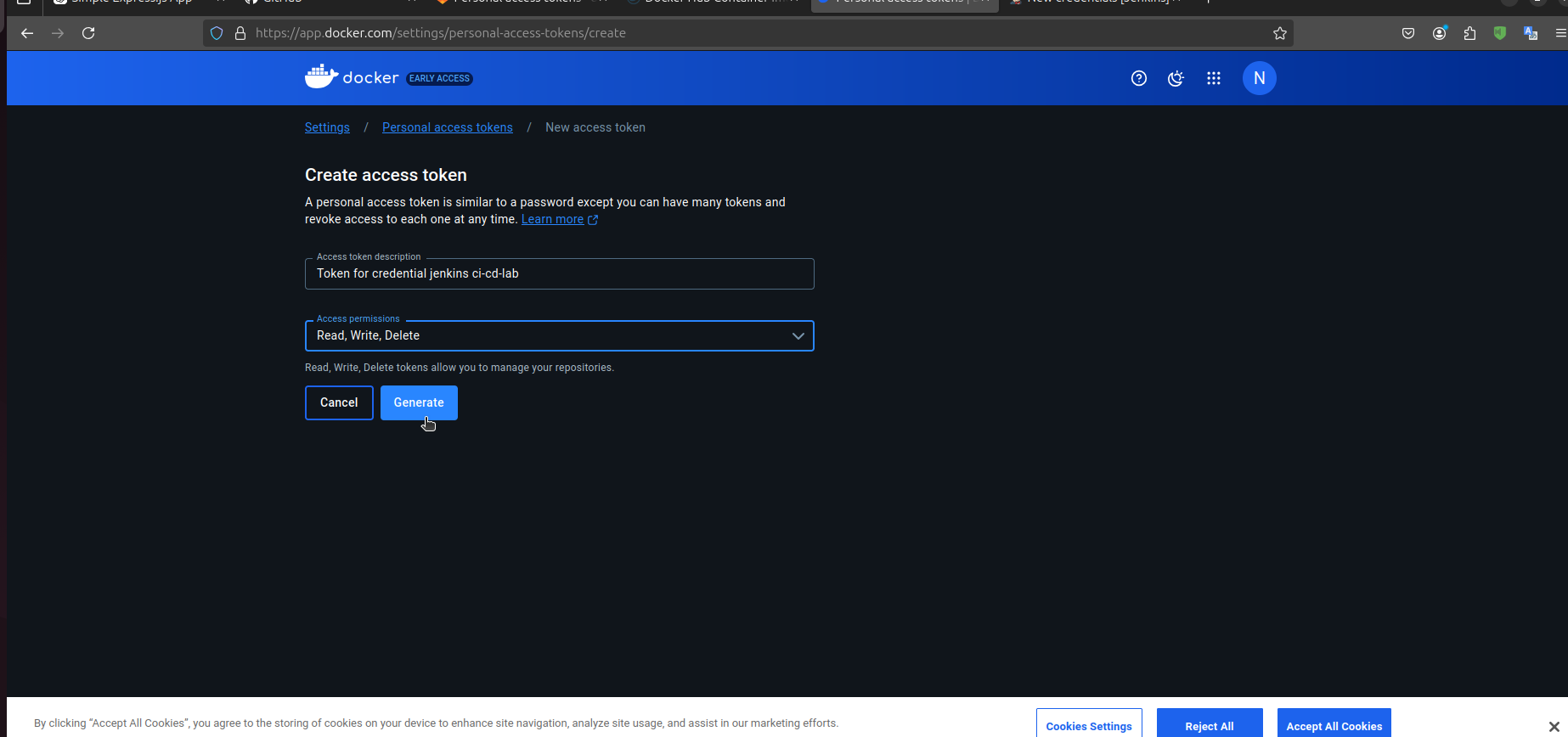
Ở phần password điền access token vừa tạo ở gitlab.



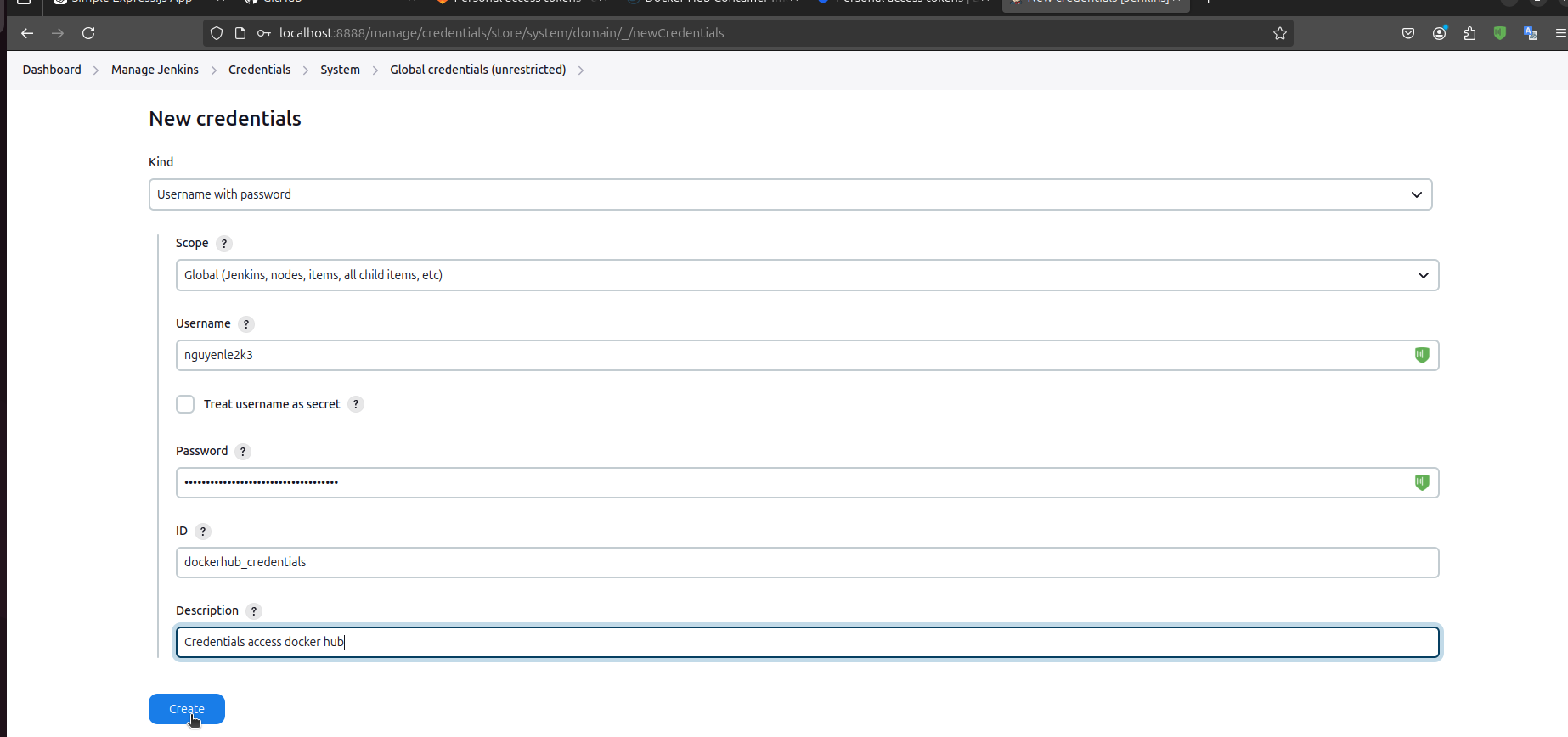
## Docker hub credential:

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:



# 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 {

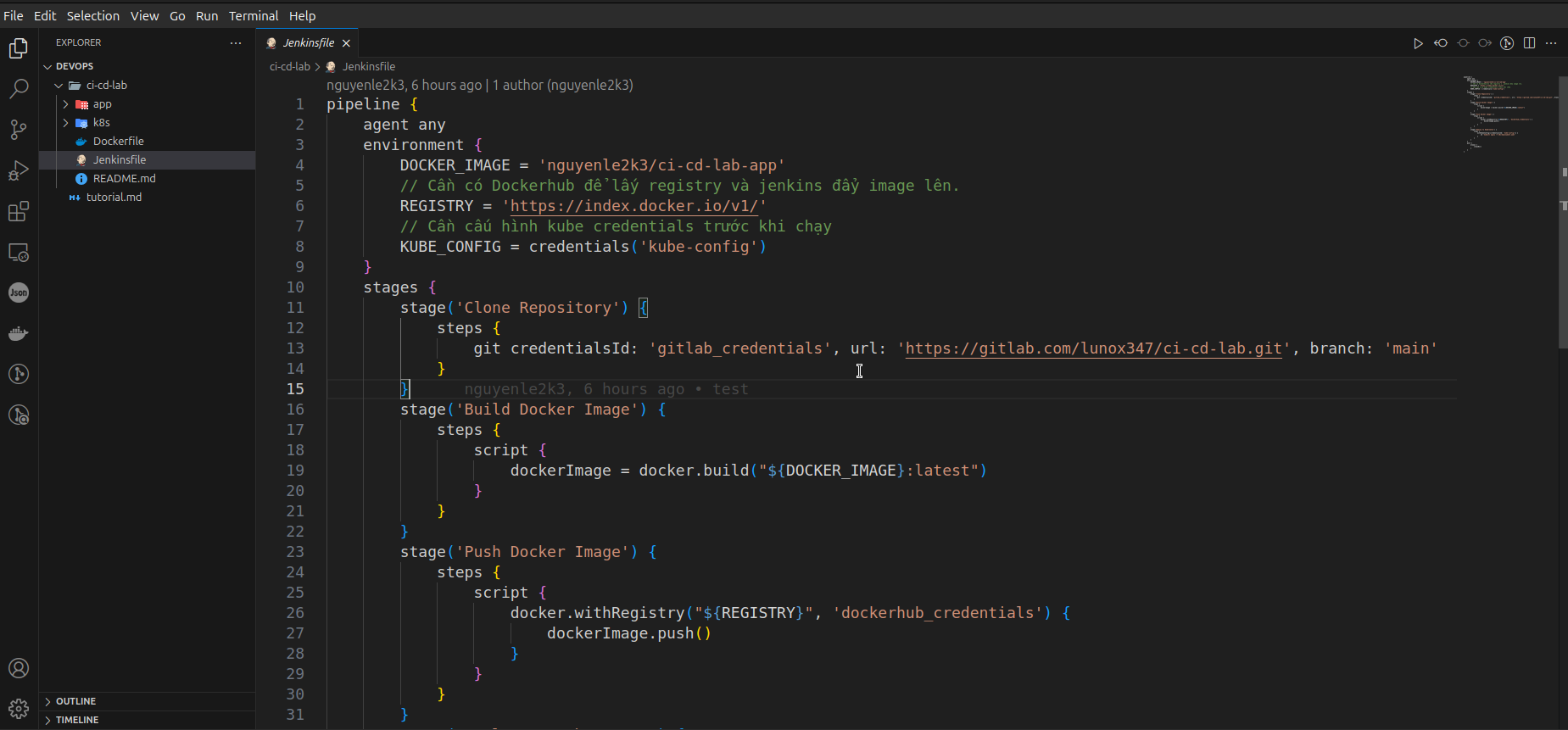
always {

cleanWs()

}

}

}



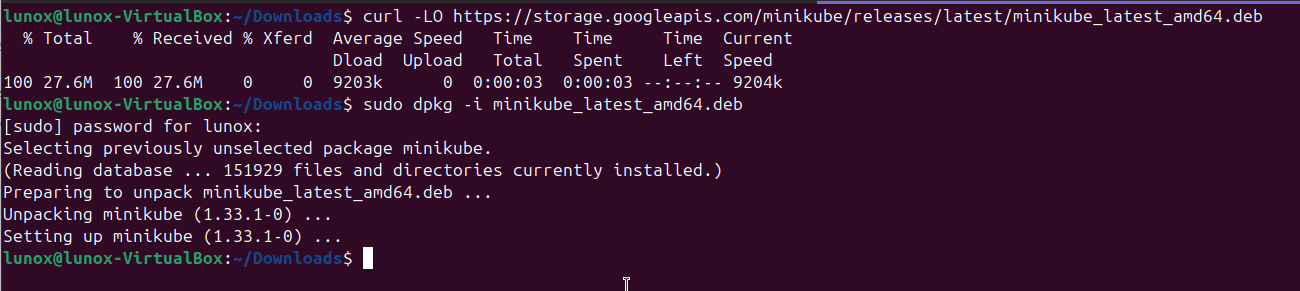
Bước 4: Cài đặt, cấu hình K8s - Minikube và tạo deployment:

# Cài đặt minikube:

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

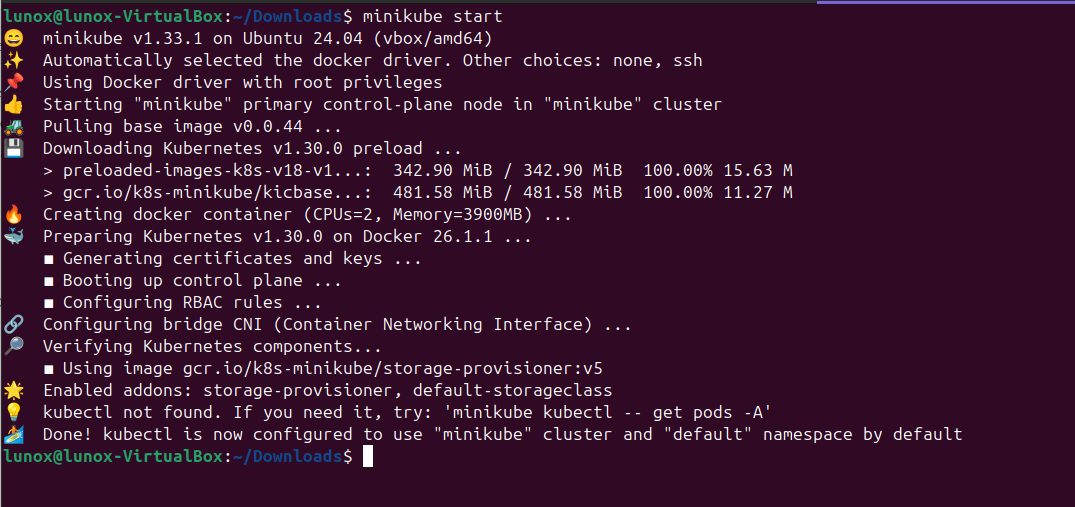
curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube\_latest\_amd64.deb

sudo dpkg -i minikube\_latest\_amd64.deb



Chạy cluster:

minikube start



Cài minikubectl:

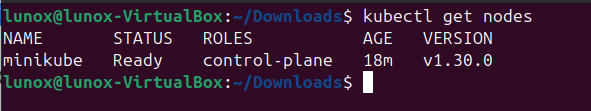
<https://kubernetes.io/docs/tasks/tools/install-kubectl-linux/>

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



# 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

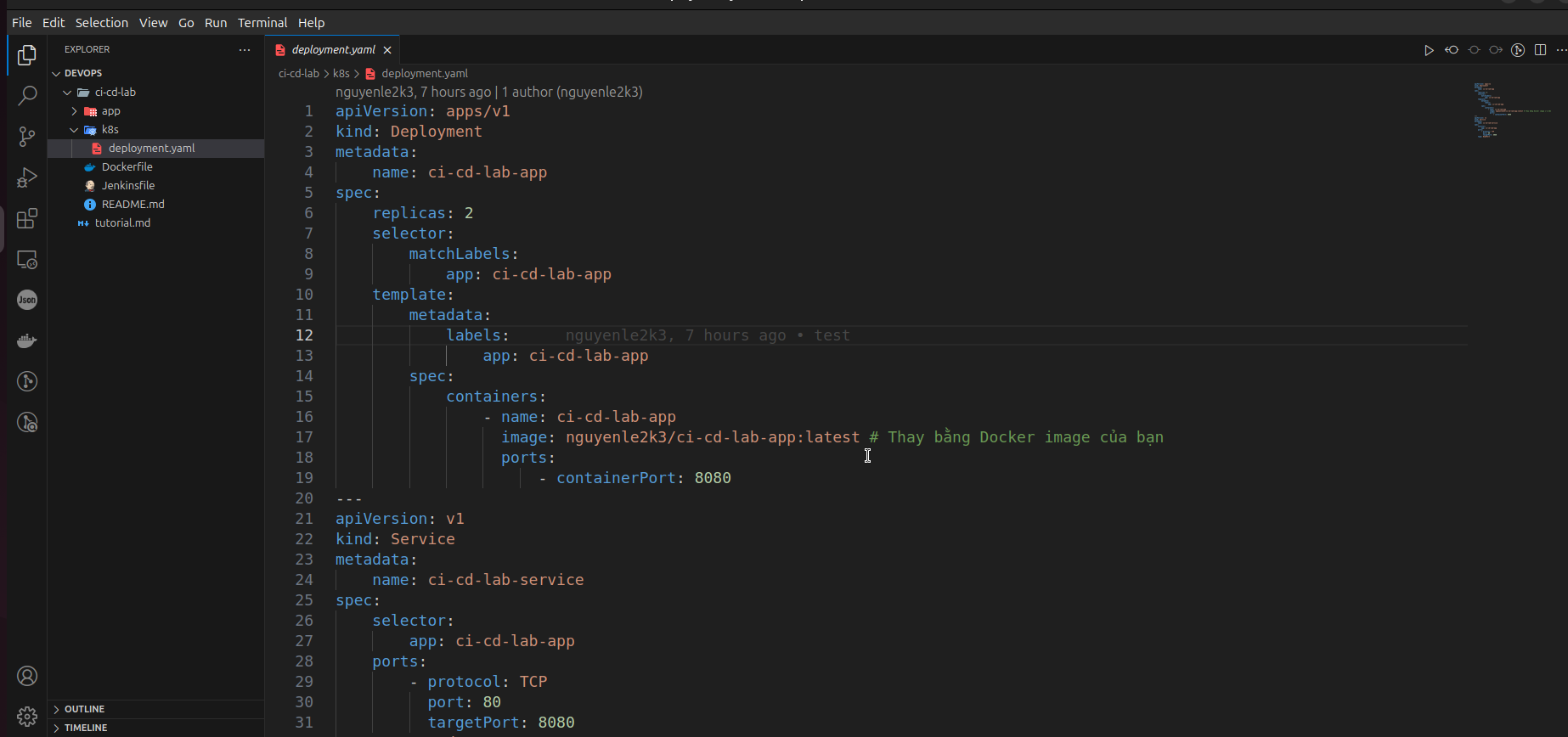
ports:

- protocol: TCP

port: 80

targetPort: 8080

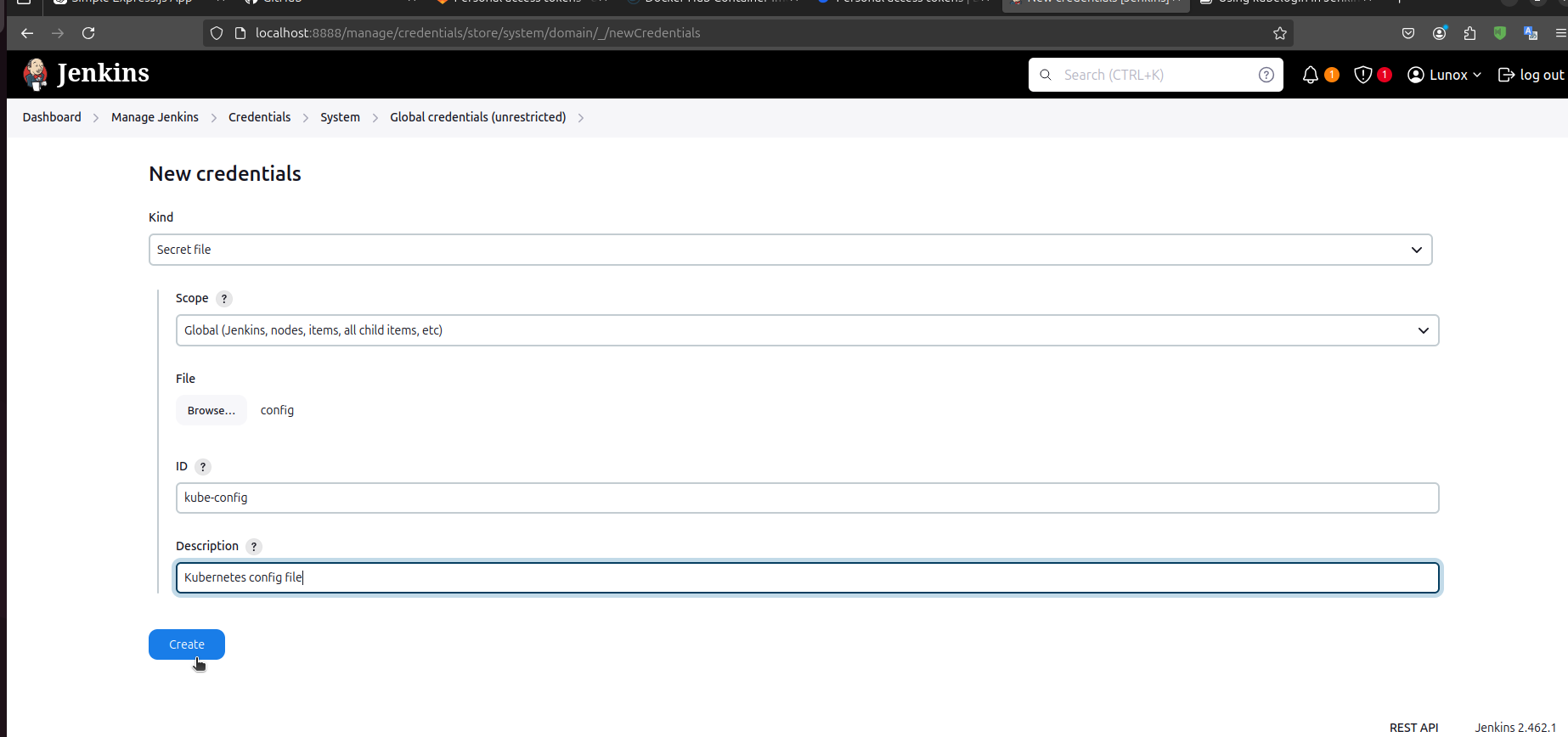
type: NodePort



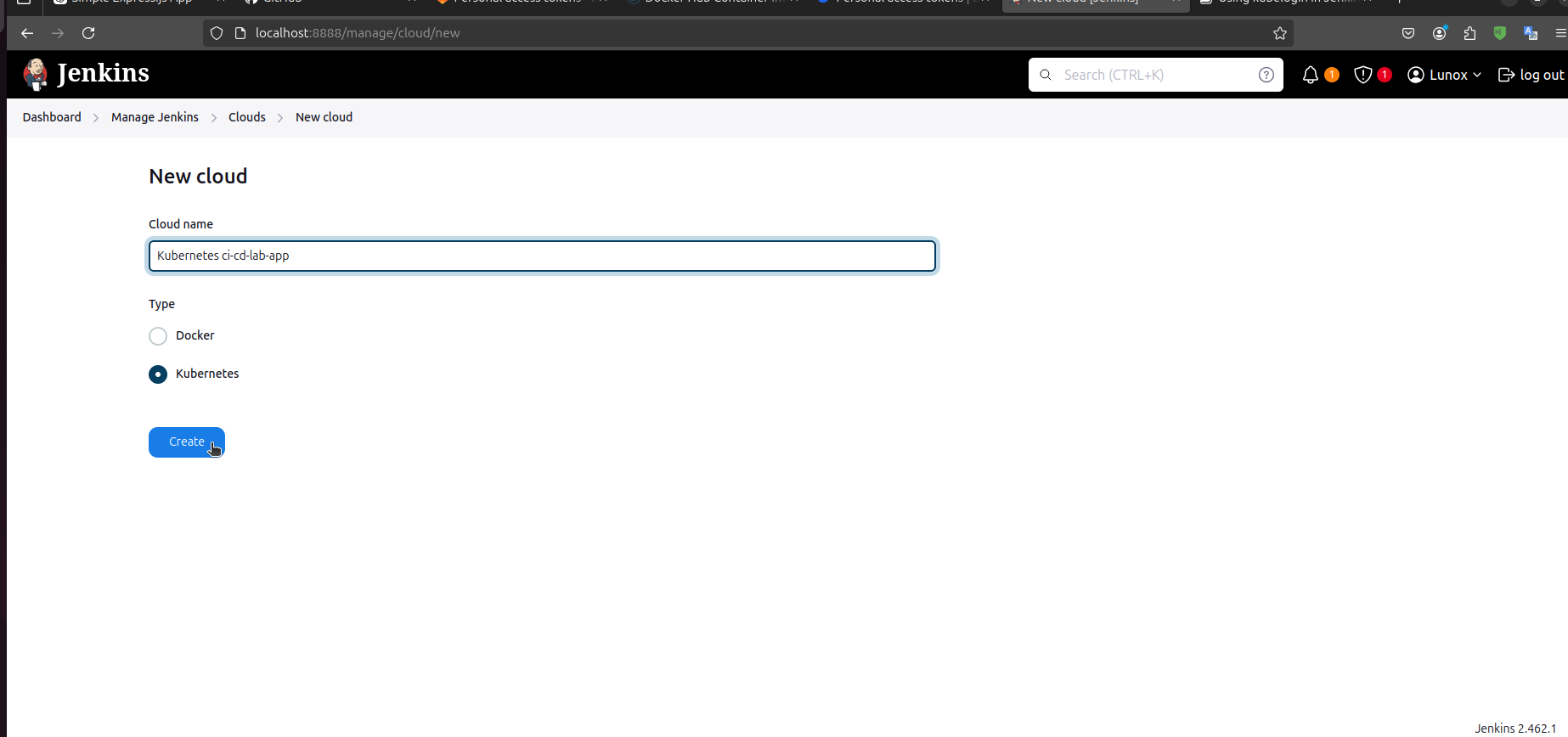
Bước 5: Tích hợp Jenkins với Kubernetes

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

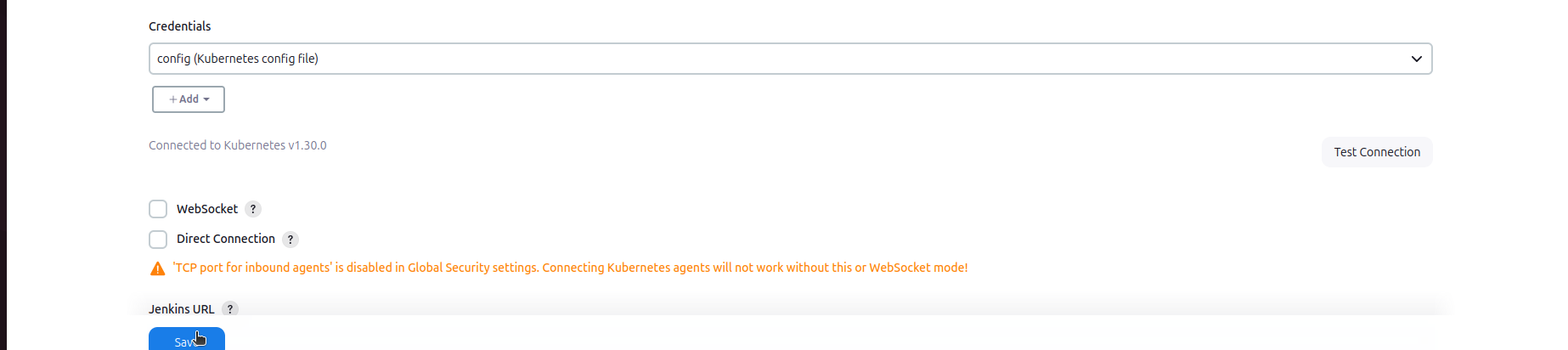
Add credentials kube-config trong jenkins:



Add cloud kubernetes trong jenkins:

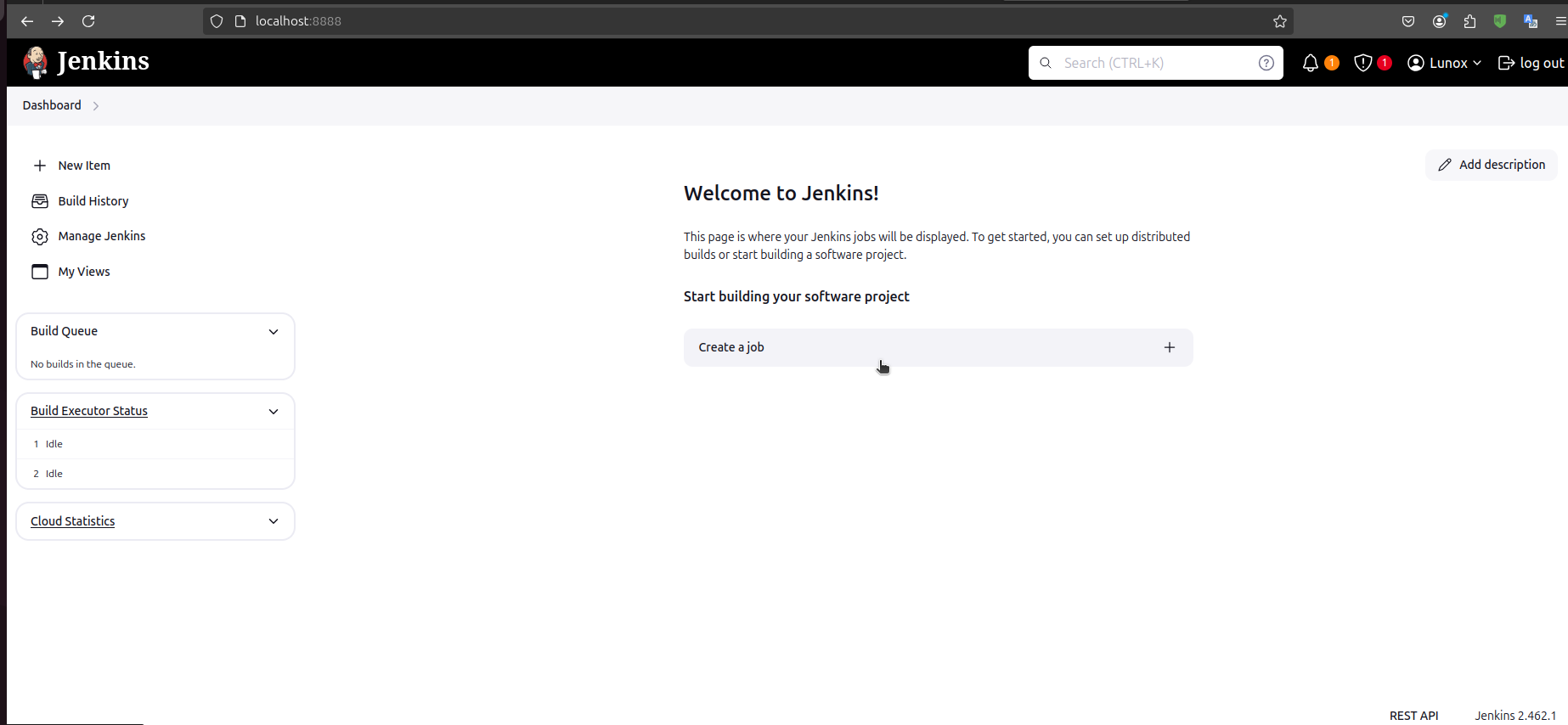


add Credentials và test connection để chắc chắn credentials hoạt động đúng.

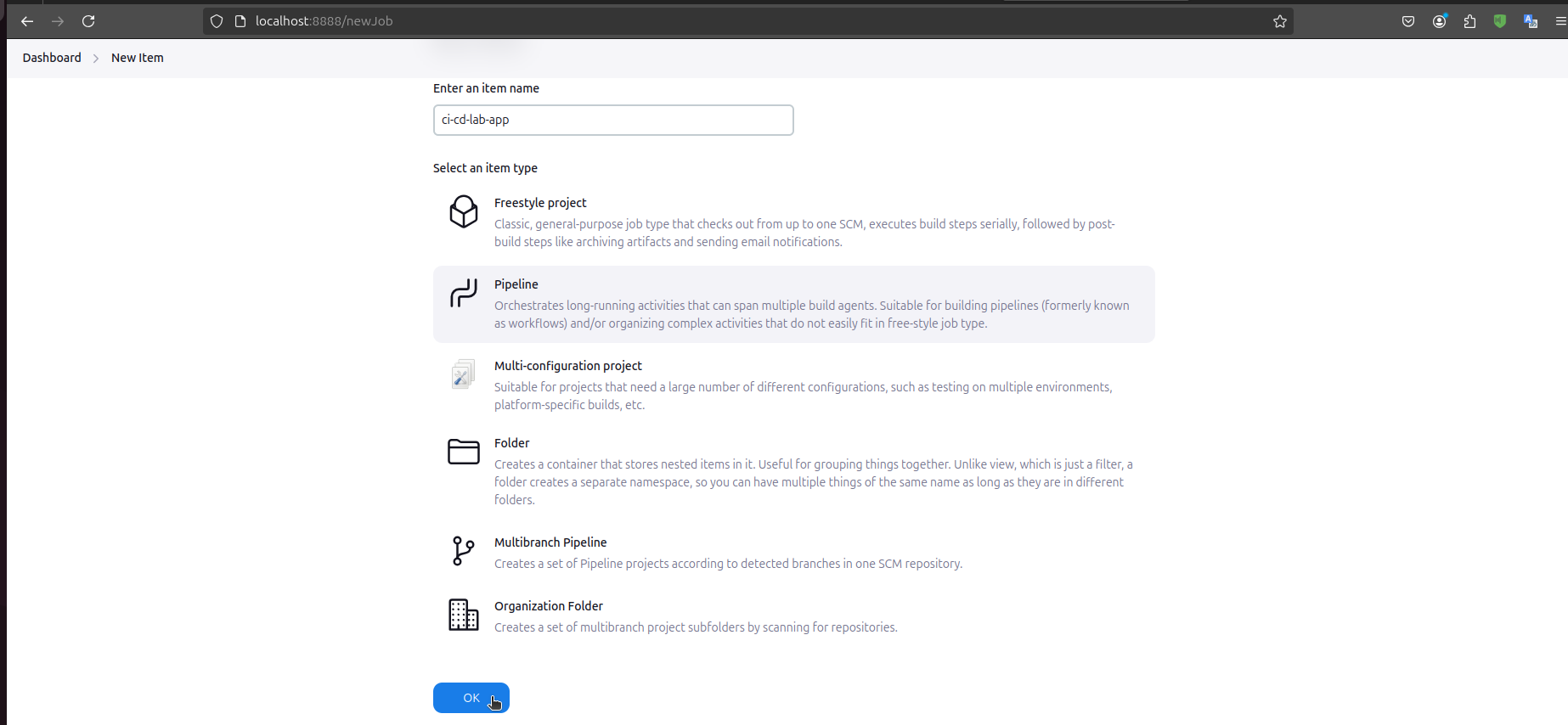


Cơ bản đã hoàn thành quá trình cấu hình. Giờ tạo một job để test.

Quay lại Dashboard > Create a job.



Nhập tên job, chọn pipeline.



Kéo xuống phần pipeline.

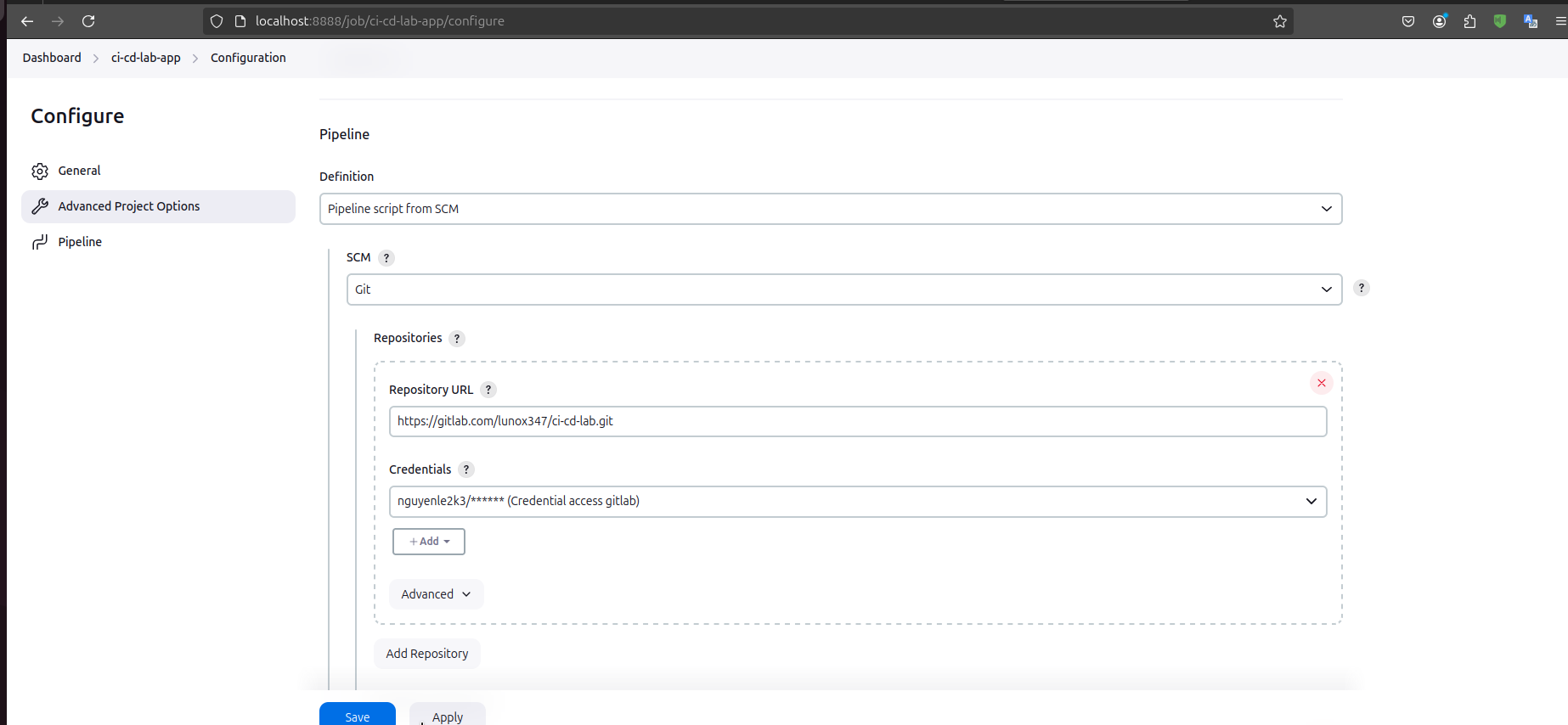
Definition: Pipeline script from SCM

SCM: Git

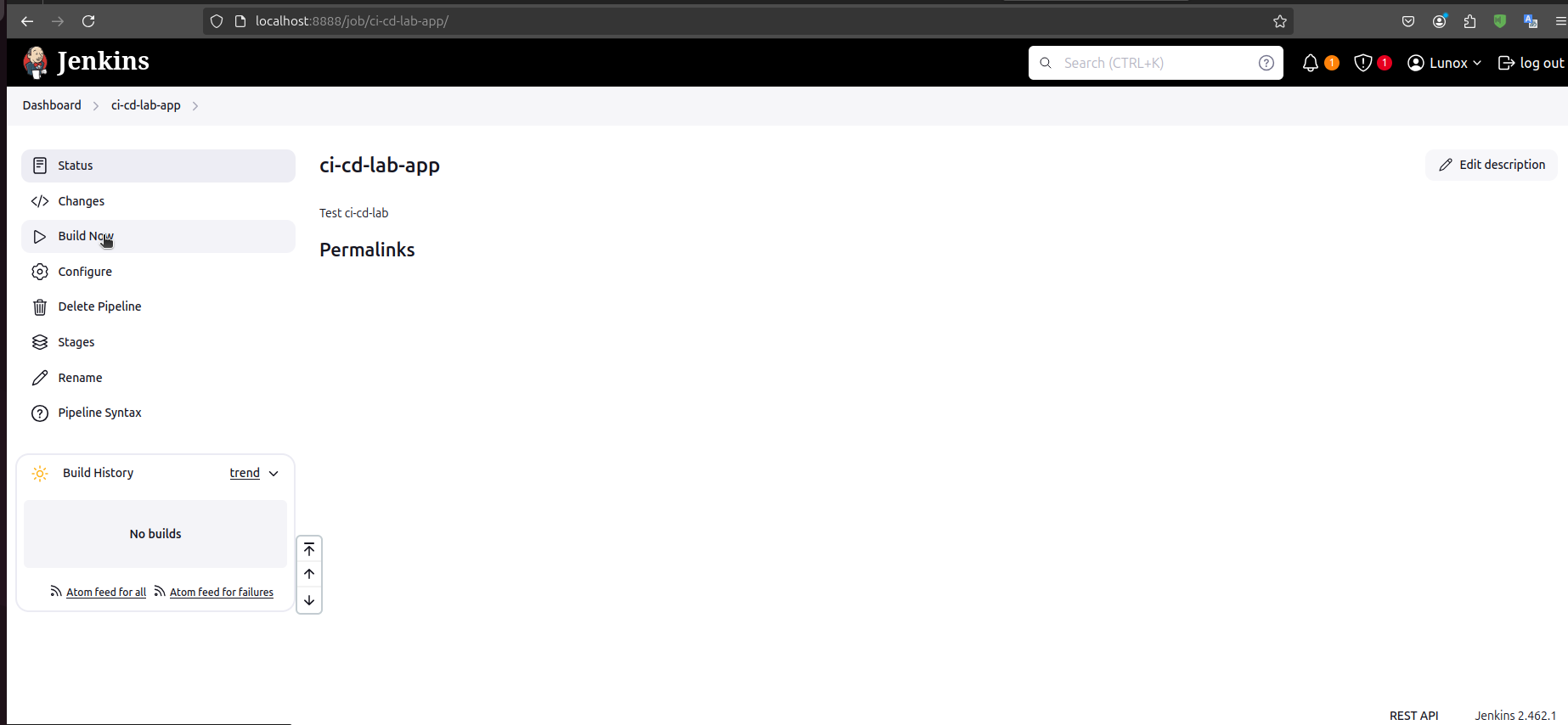
Repository URL: URL repo git lab

Credential: gitlab\_credentials

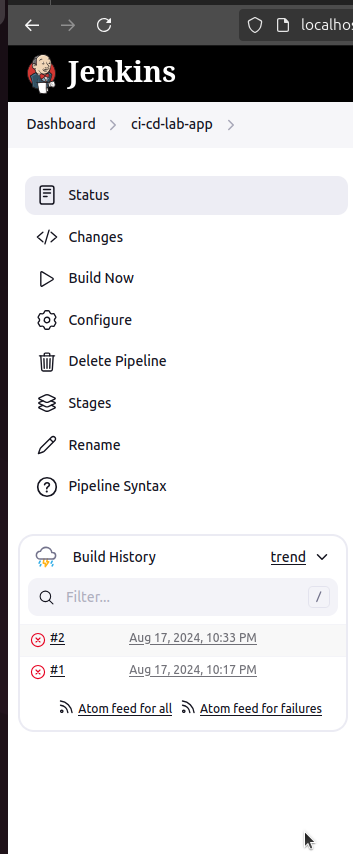
Script path: Jenkinsfile

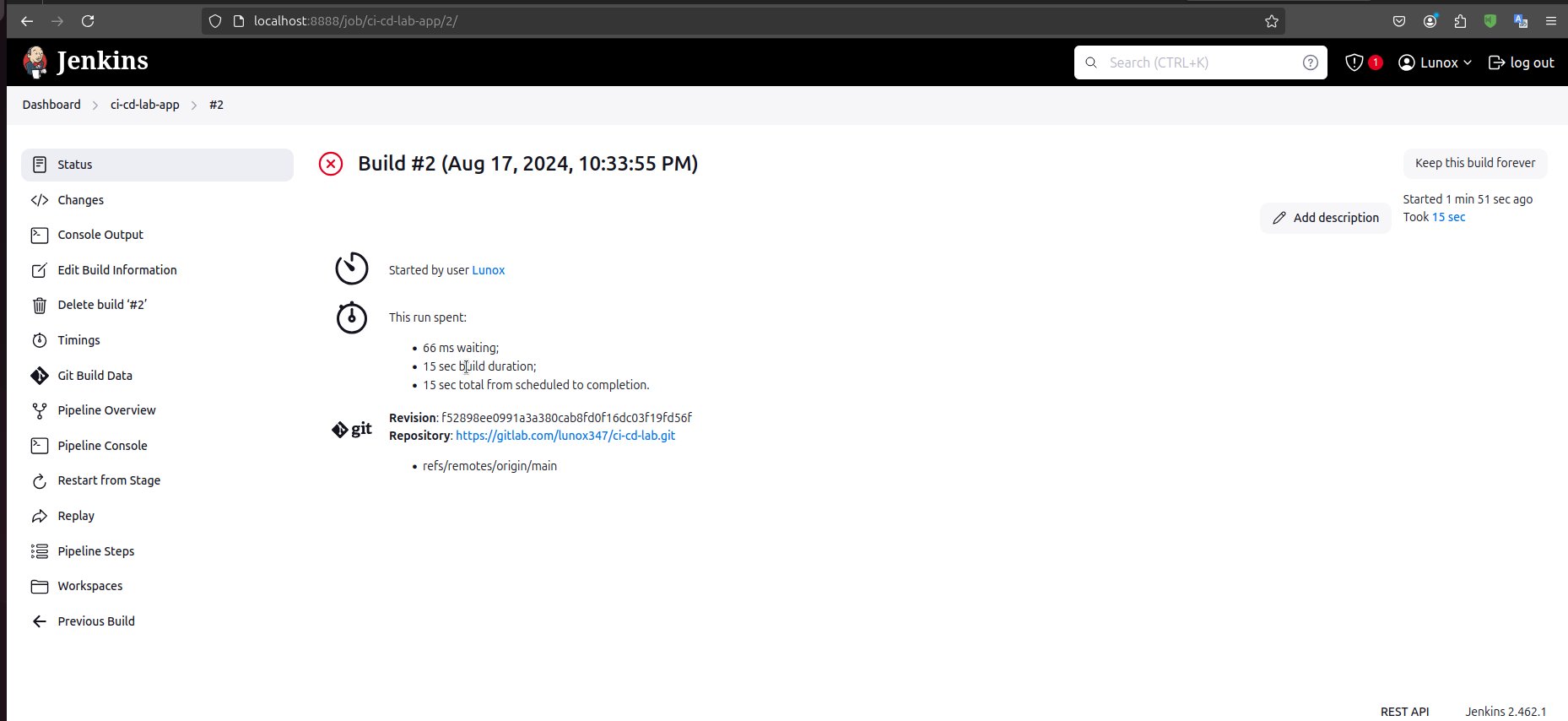


Chọn Build now để build app.

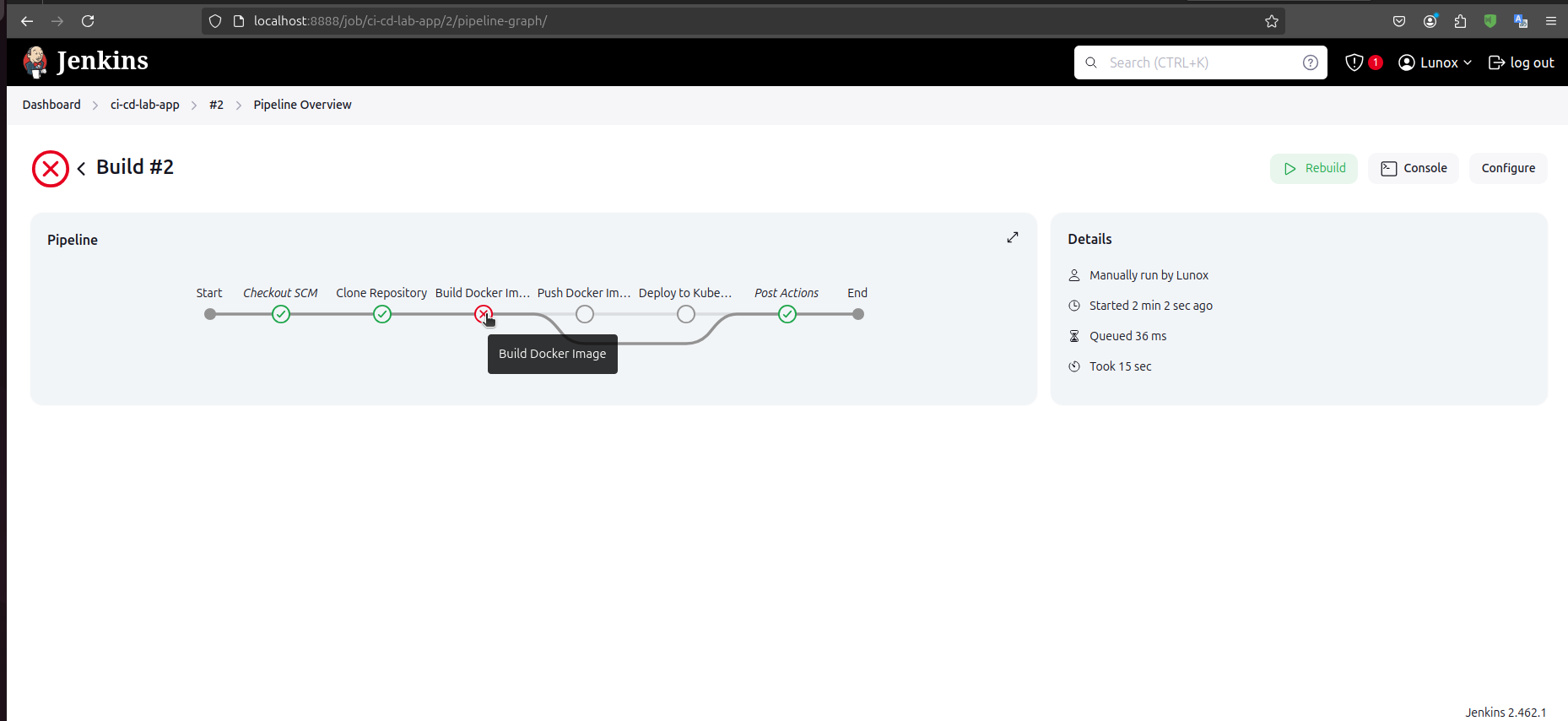


Lịch sử các lượt build:

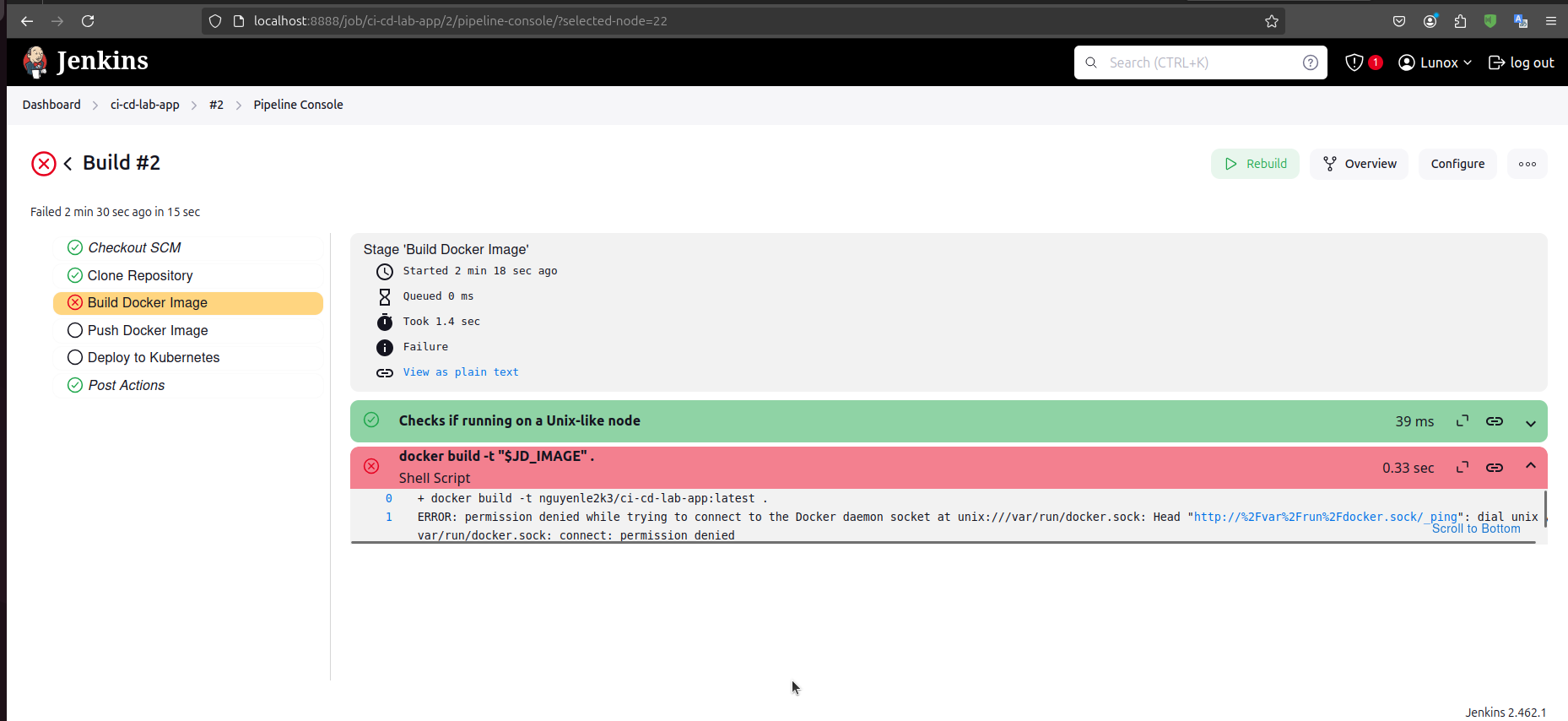


Bấm vào #2 để xem chi tiết về lượt build  


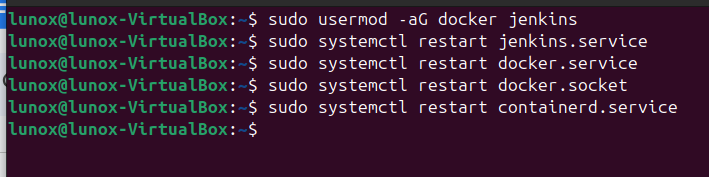
Ta thấy build hỏng từ bước Build Docker Image



Xem chi tiết lỗi:

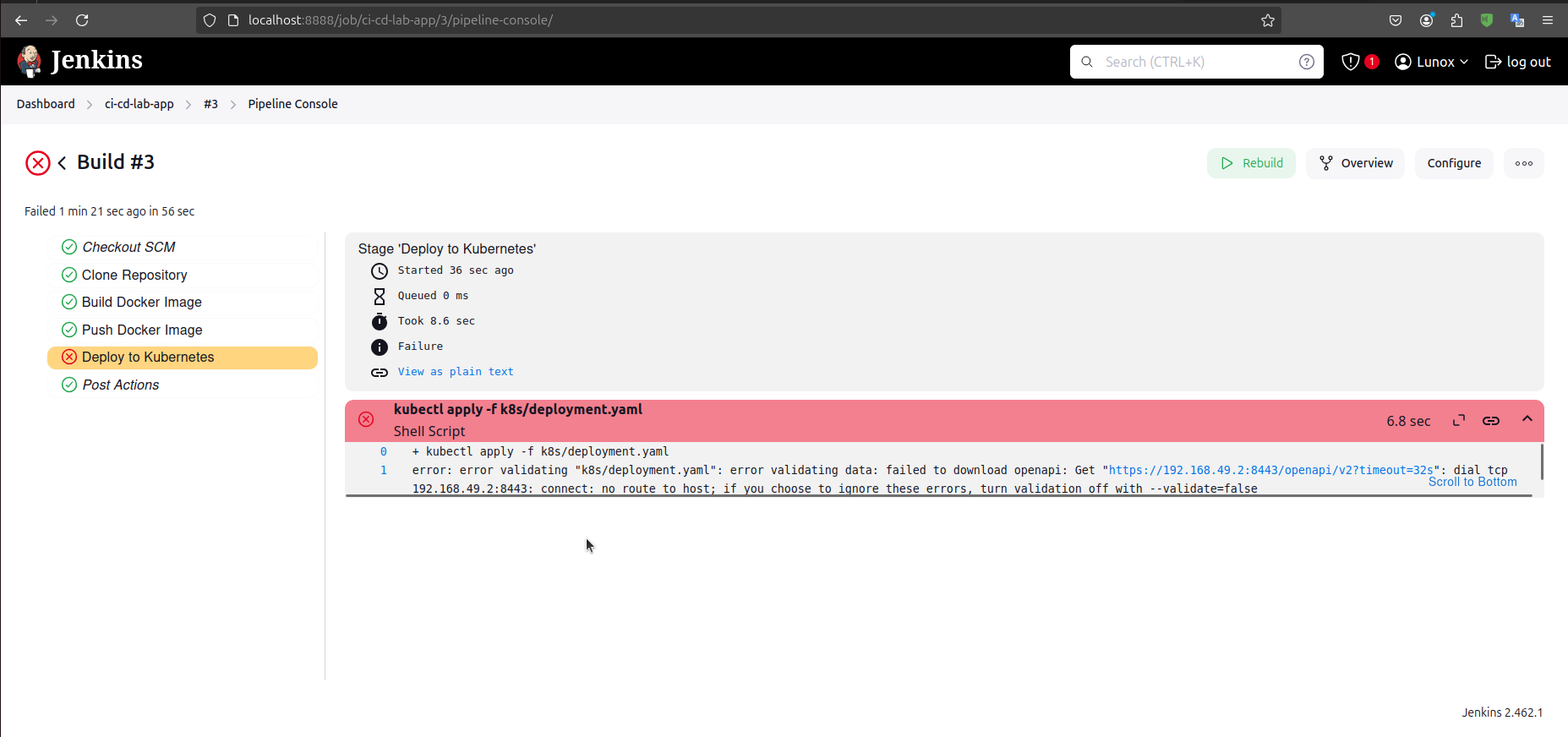


Lỗi này do chưa add user jenkins vào group docker.

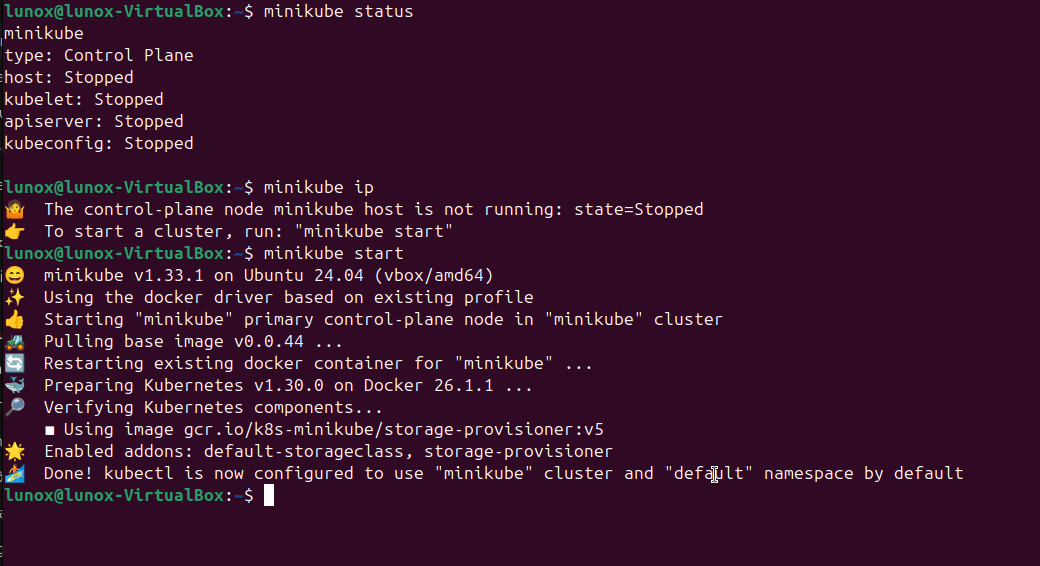


Build lại:

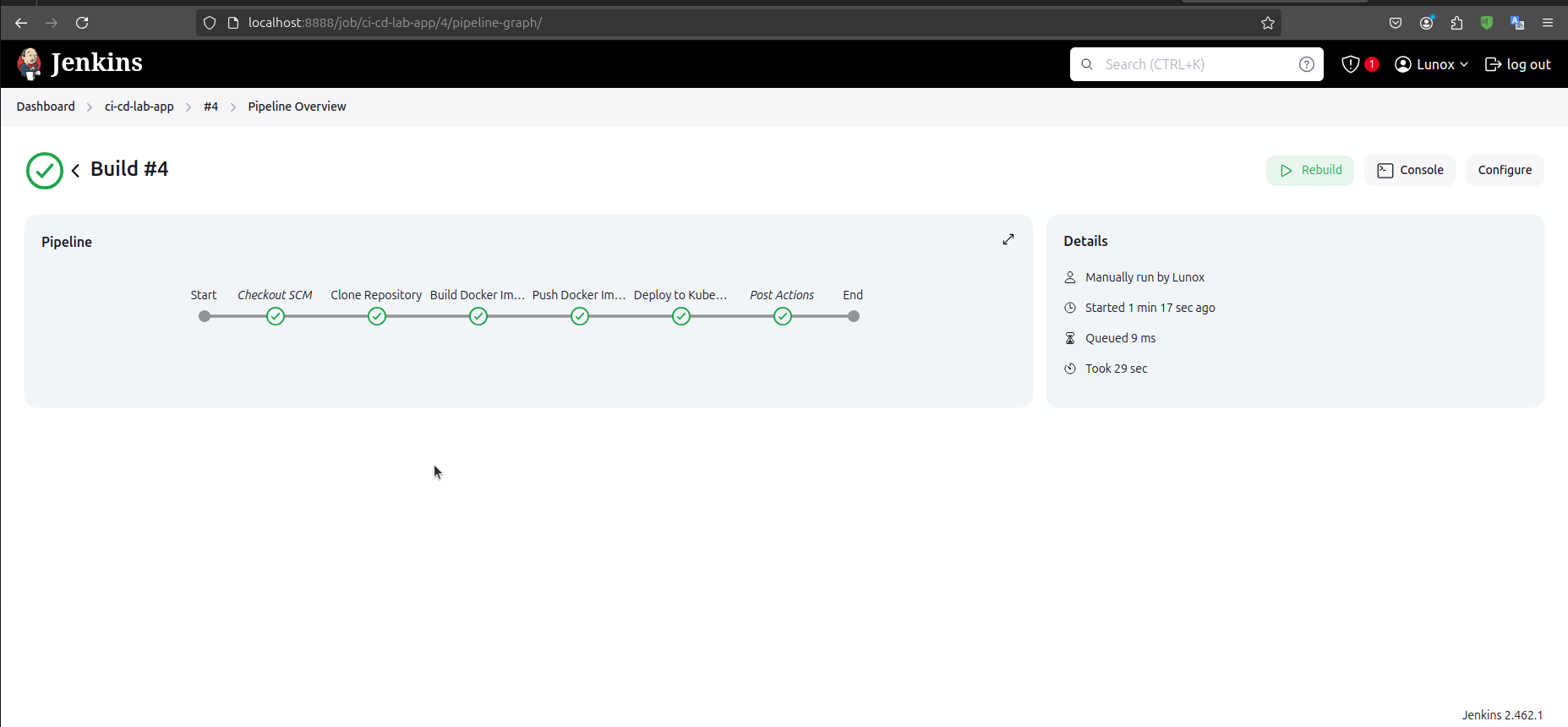
Lỗi ở bước Deploy:



Lỗi gây ra do minikube bị tắt:

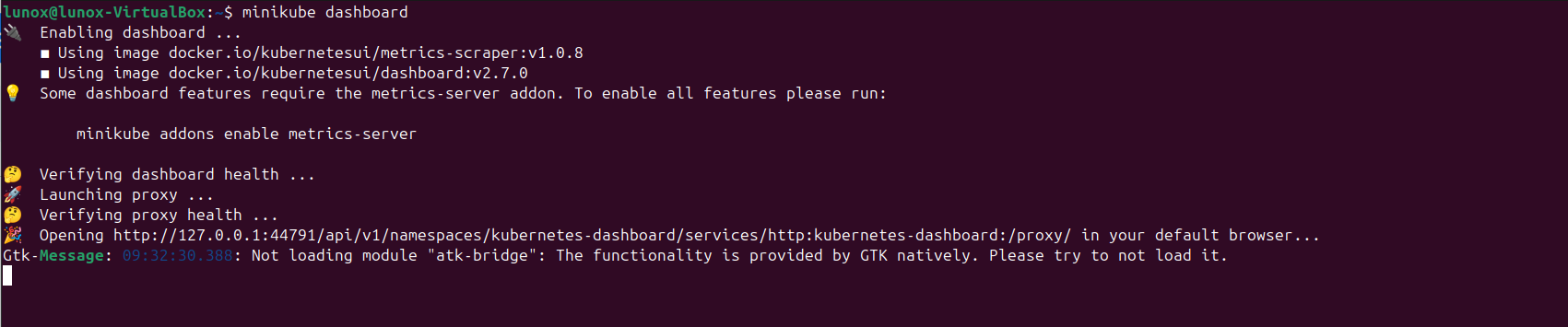


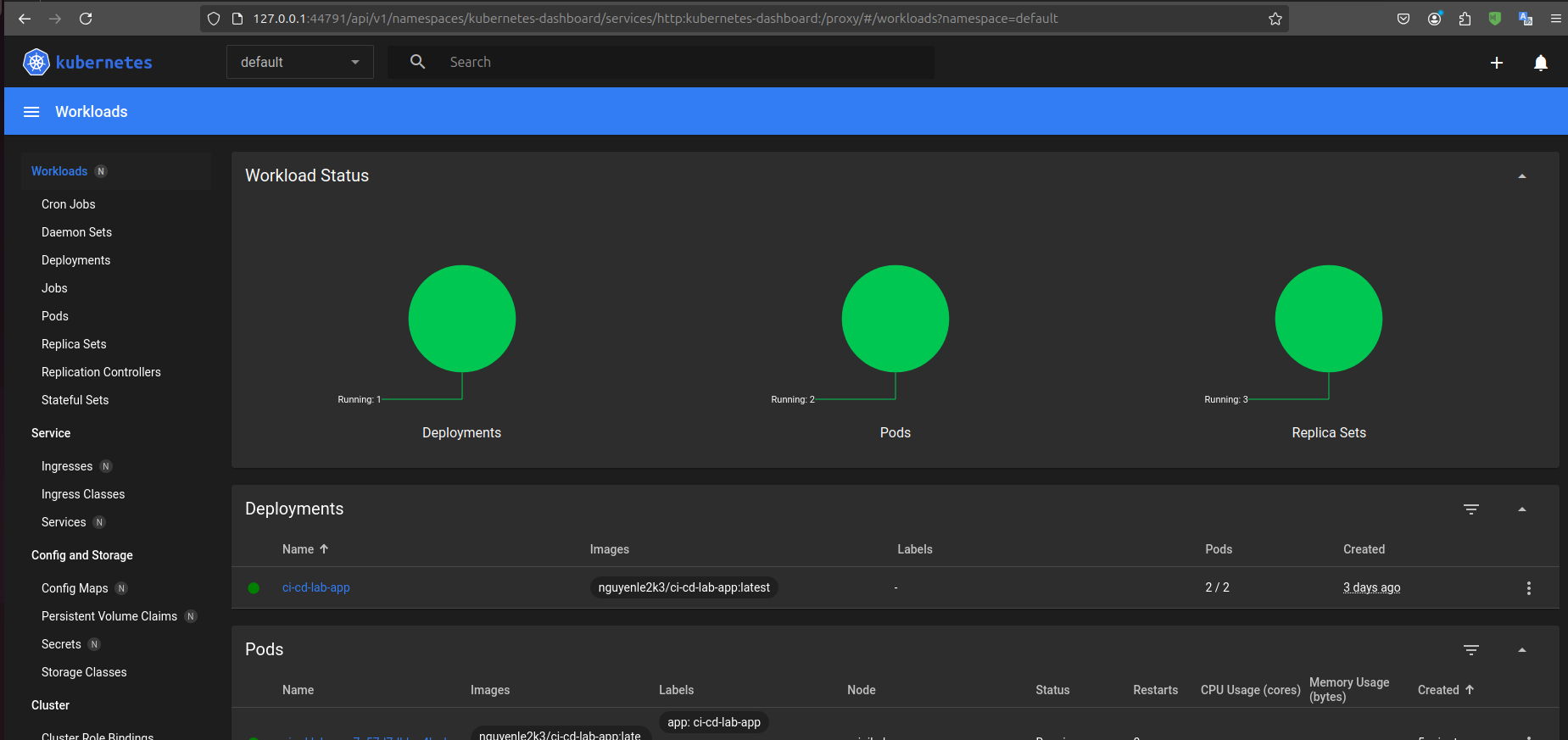
Rebuild:

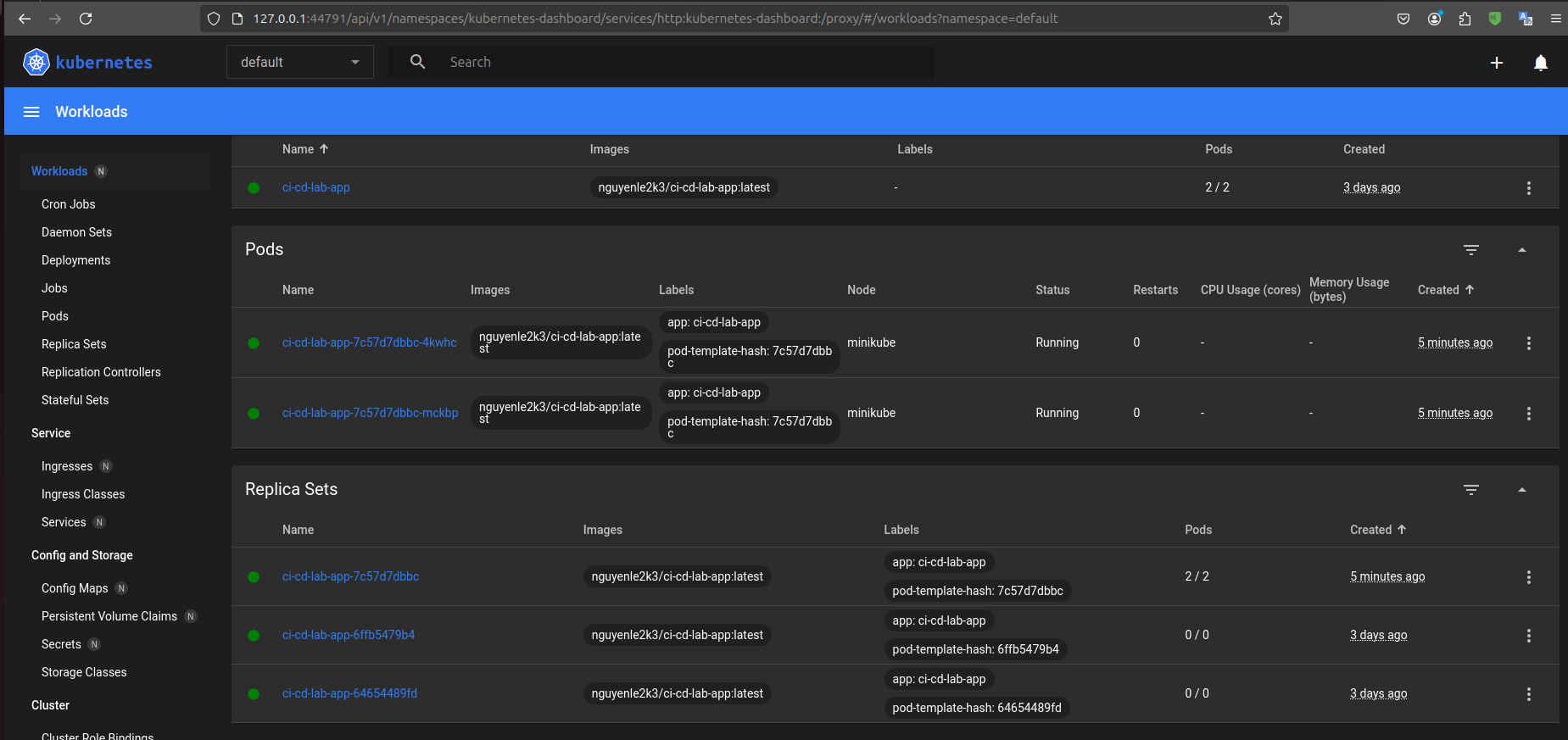


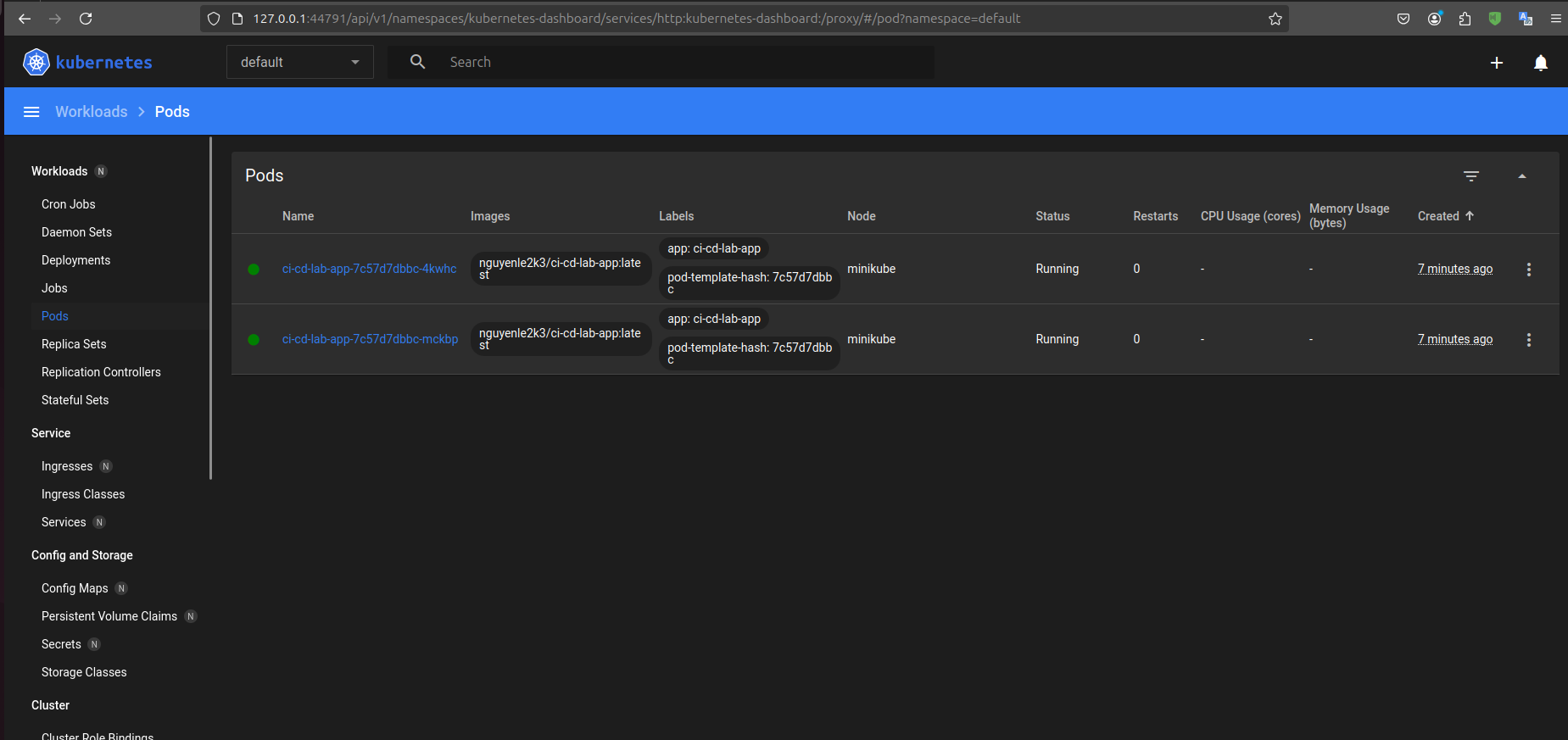
Thành công.

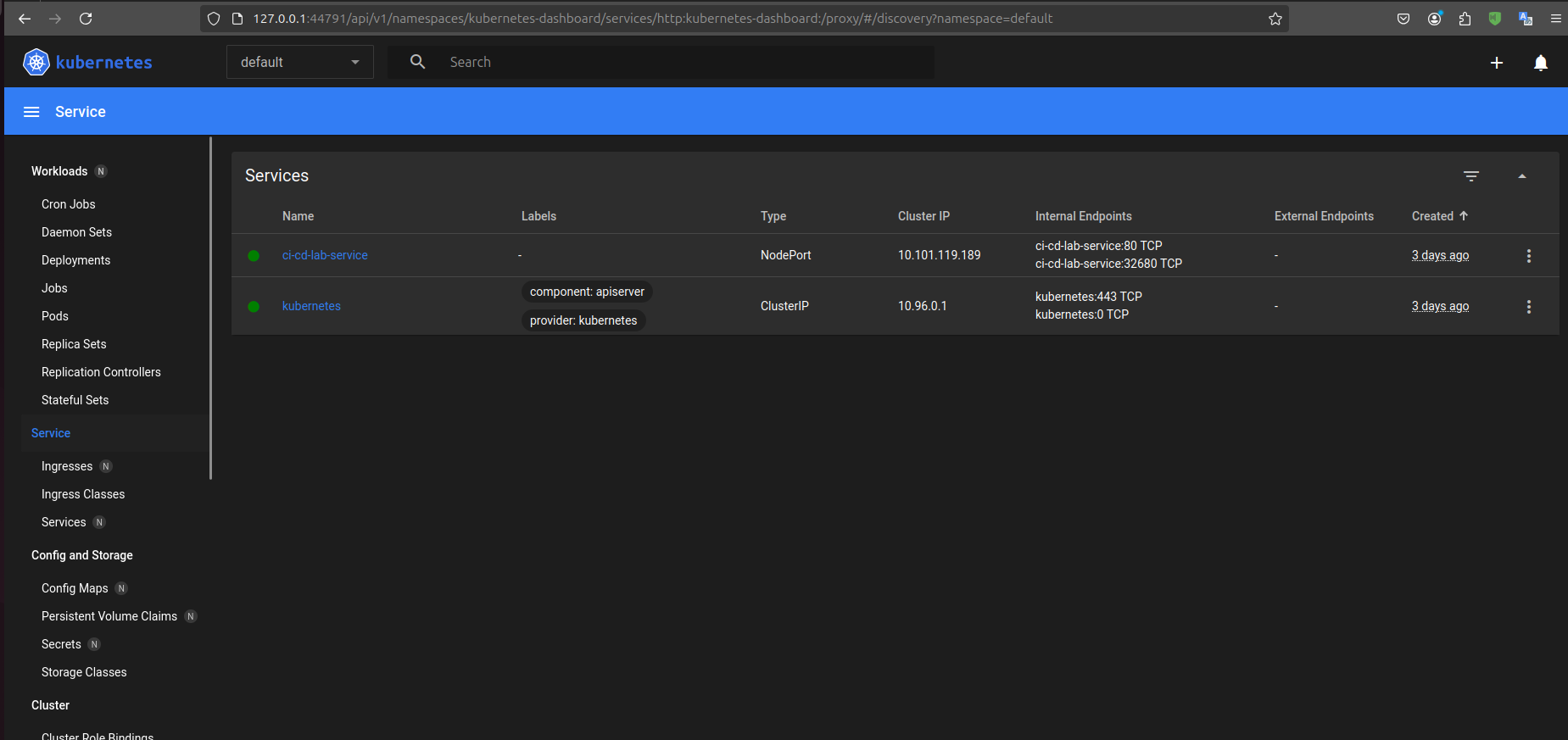
Có thể sử dụng minikube dashboard để kiểm tra toàn diện minikube với web gui.

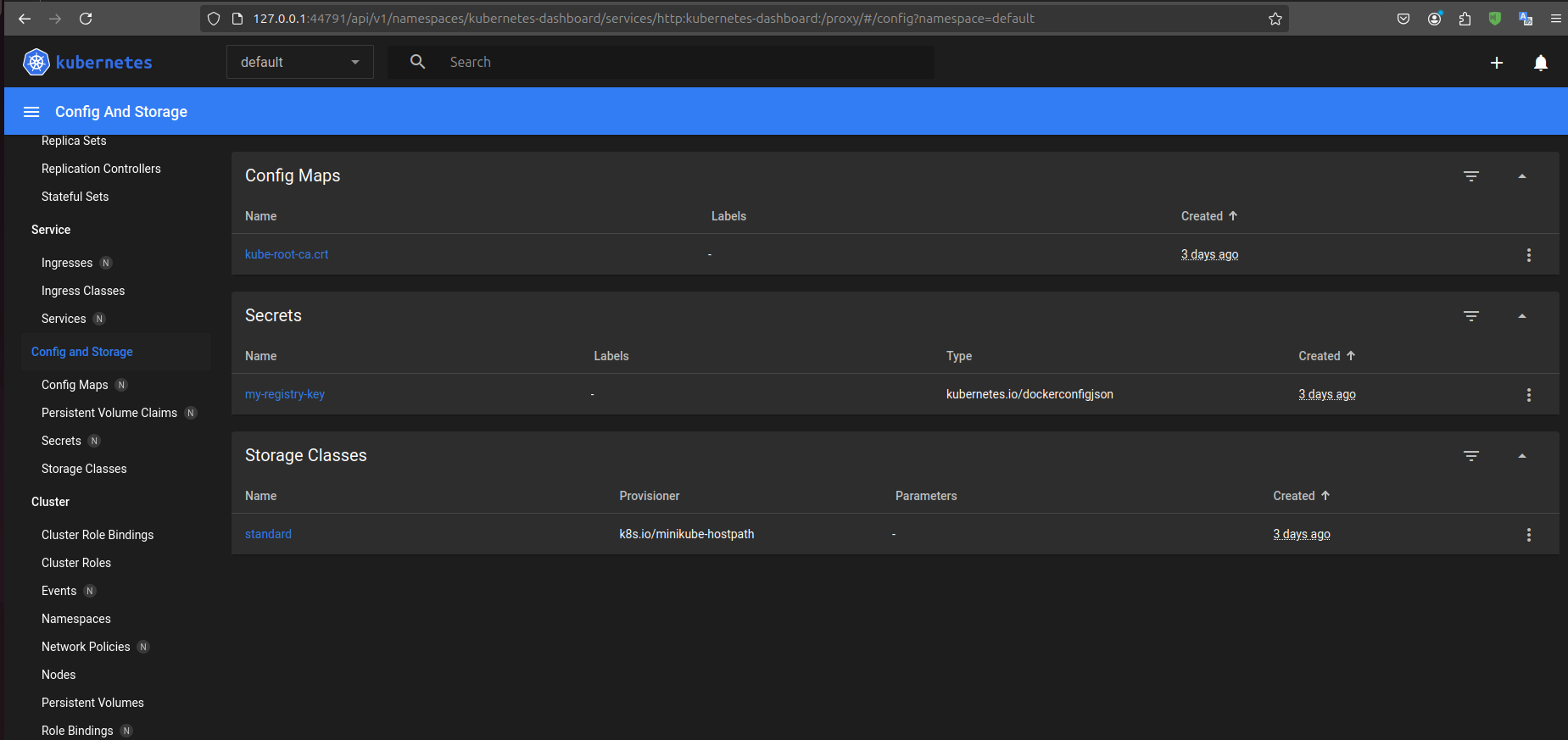


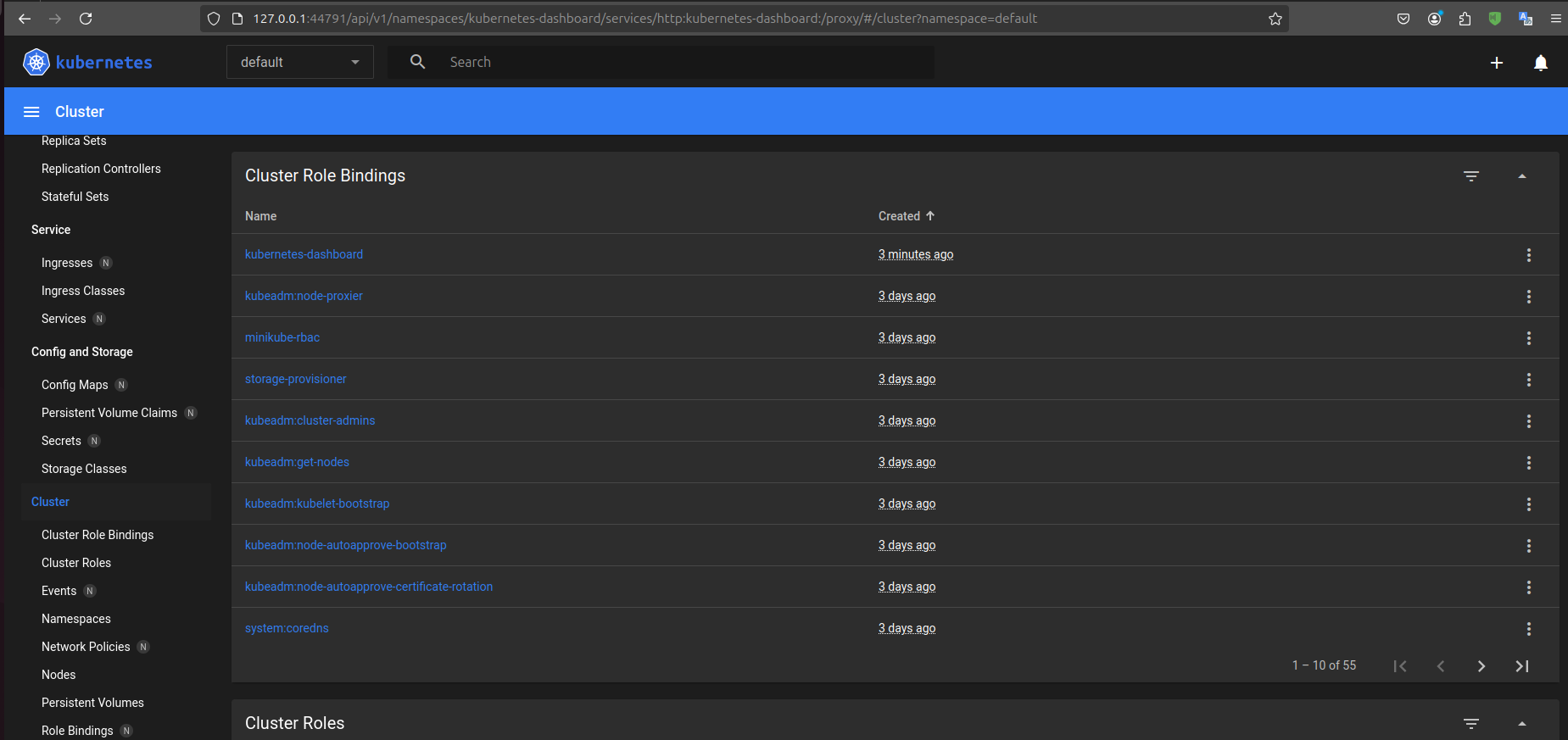












Bước 6: Kiểm tra và giám sát deployment:

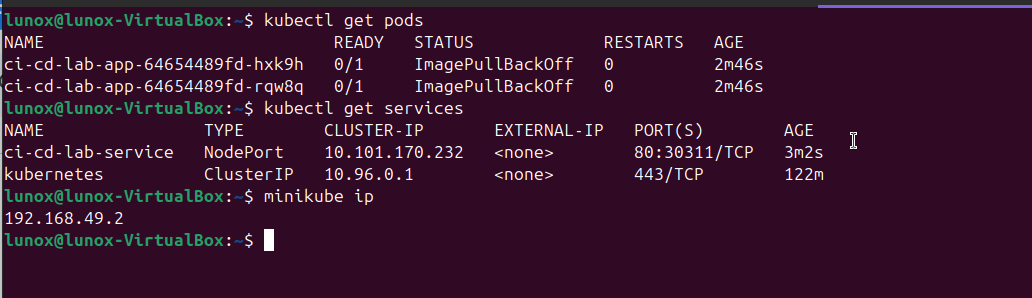
# Kiểm tra Deployment:

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



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

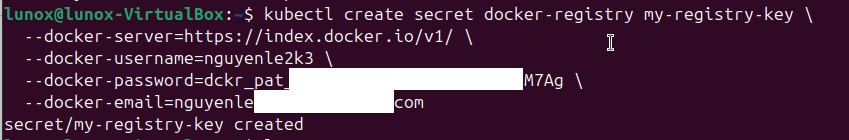
kubectl create secret docker-registry my-registry-key \

--docker-server=https://index.docker.io/v1/ \

--docker-username=<your-dockerhub-username> \

--docker-password=<your-dockerhub-password> \

--docker-email=<your-dockerhub-email>

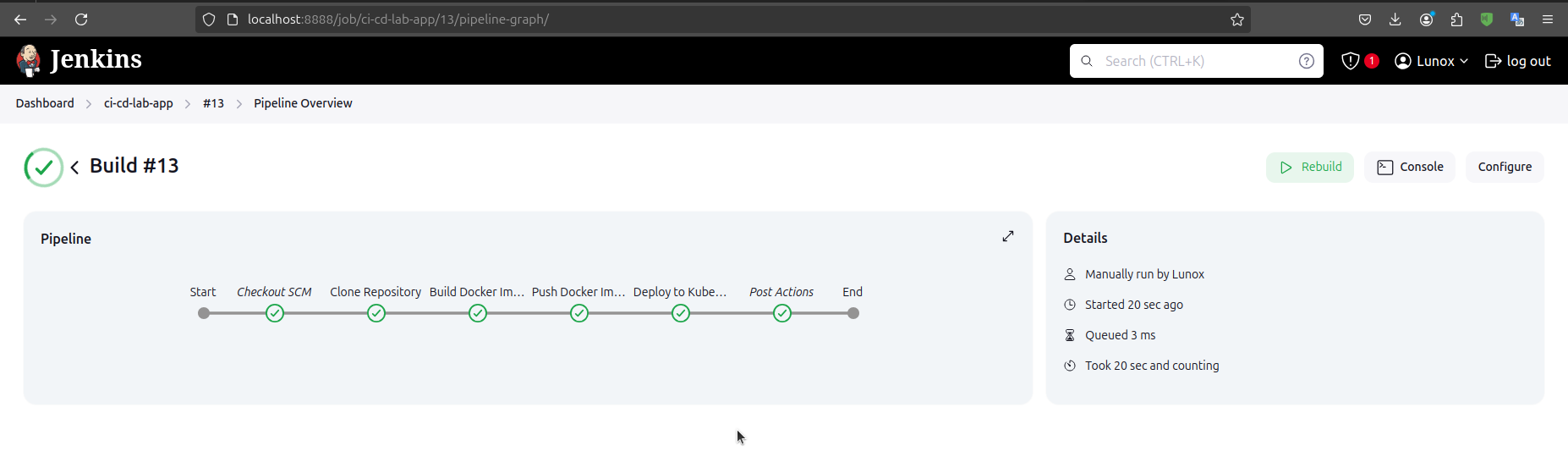


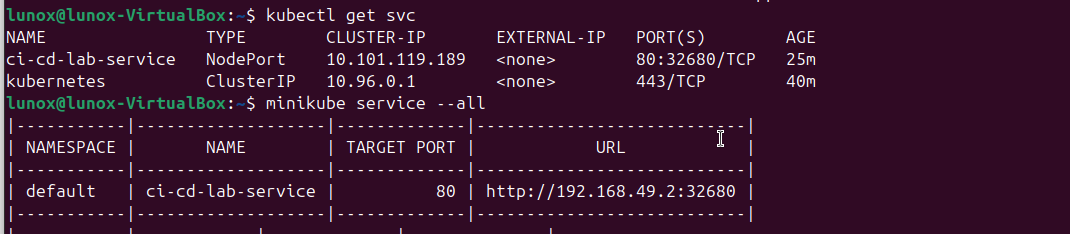
Thêm vào file deployment.yaml

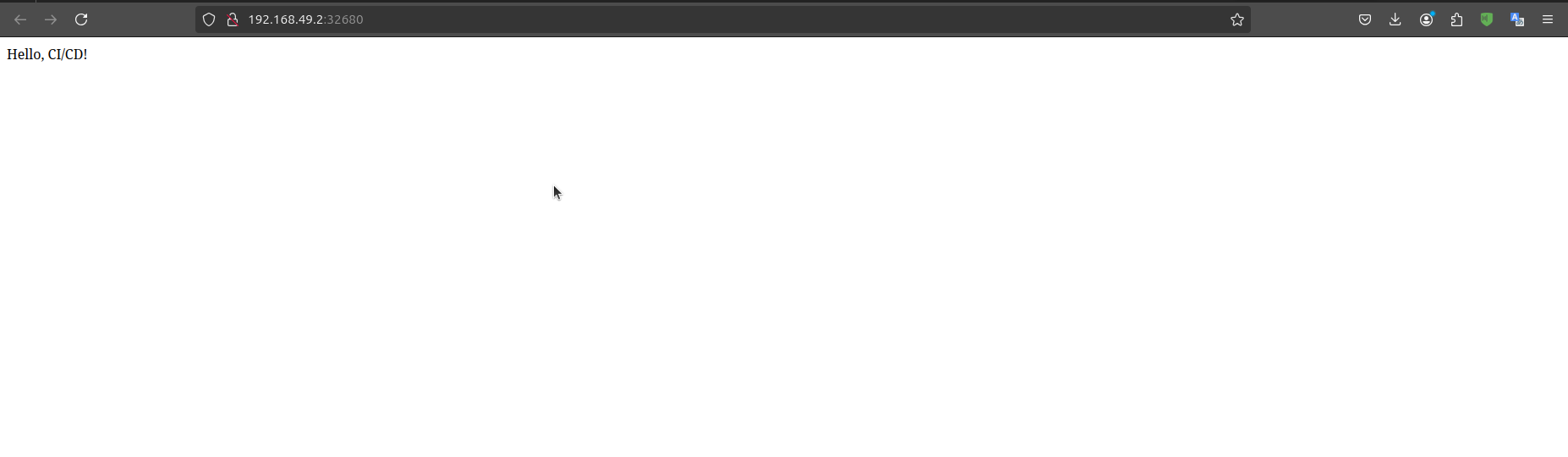
imagePullSecrets: - name: my-registry-key



Rebuild and open web:







Thành công.

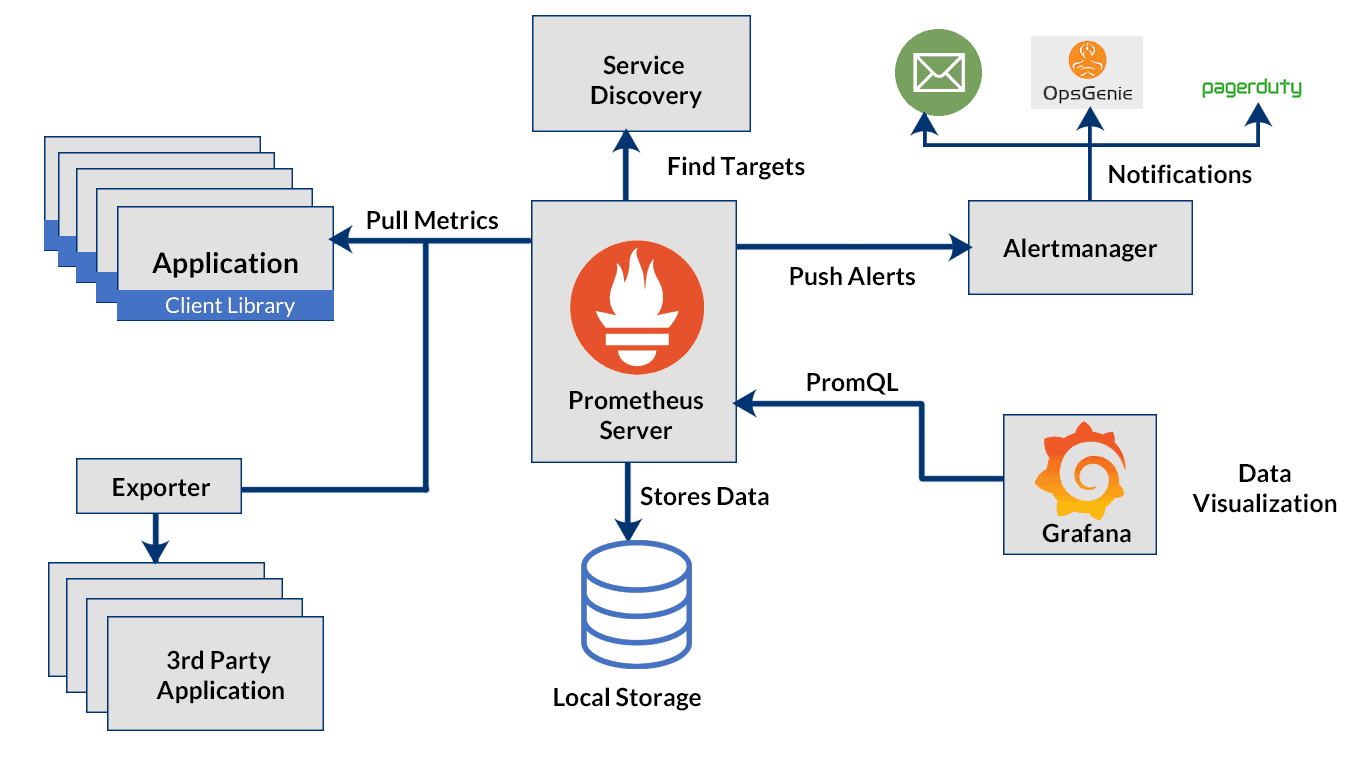
Khi có cập nhật về app. Cần sử dụng jenkins để build lại ứng dụng. Chờ khi build thành công thì sử dụng lệnh kubectl rollout restart deployment ci-cd-lab-app để update deployment

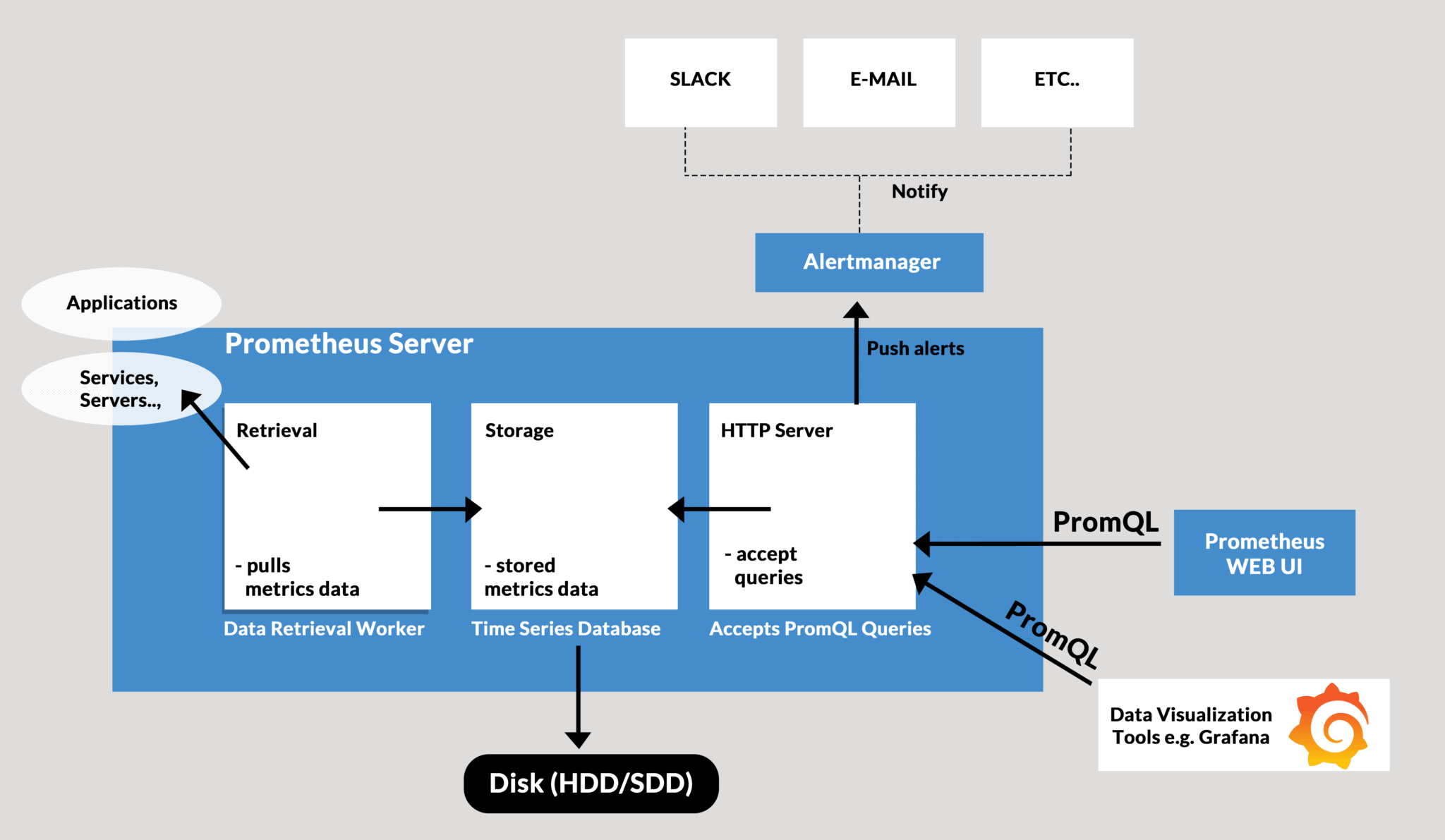
# Cài đặt và sử dụng Prometheus và Grafana

<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:

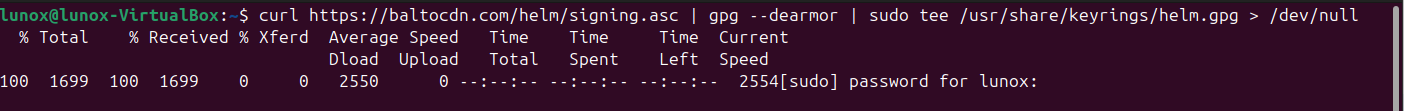
curl https://baltocdn.com/helm/signing.asc | gpg --dearmor | sudo tee /usr/share/keyrings/helm.gpg > /dev/null

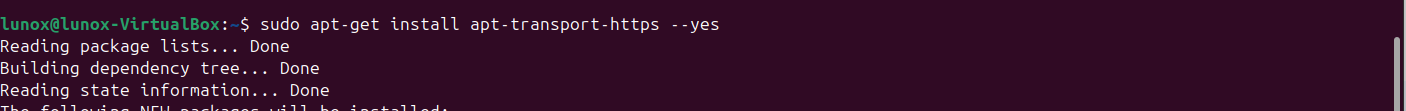
sudo apt-get install apt-transport-https --yes

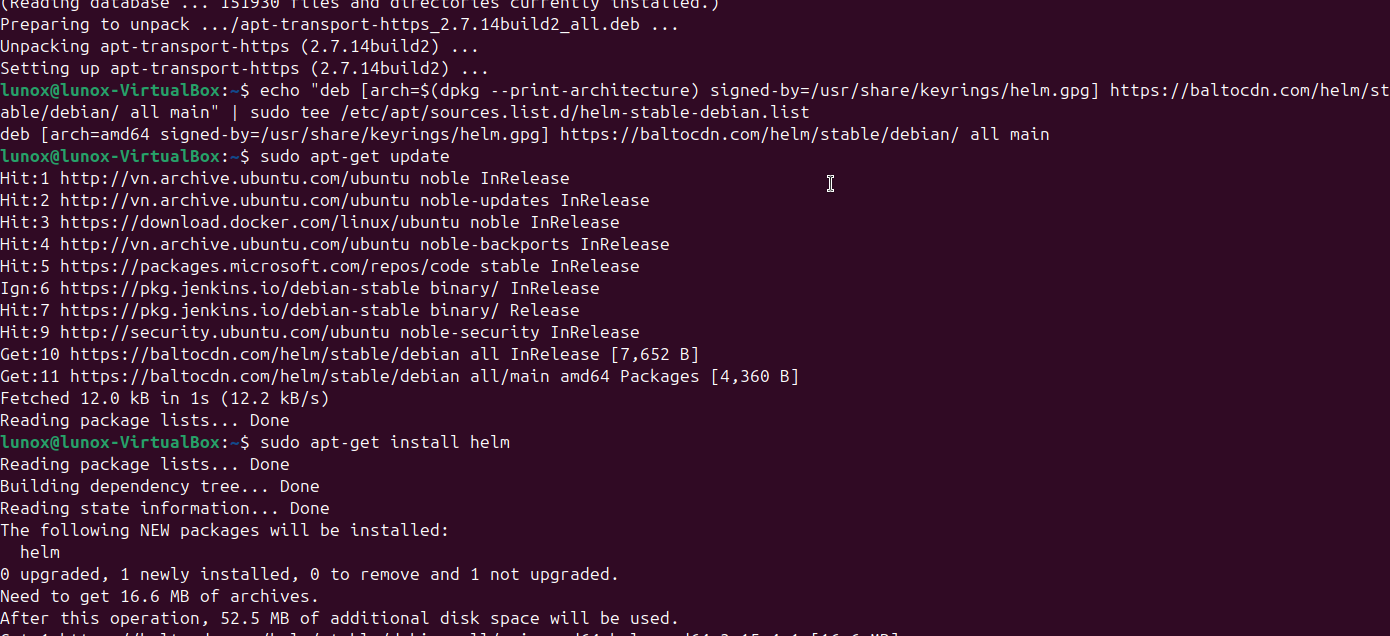
echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/helm.gpg] https://baltocdn.com/helm/stable/debian/ all main" | sudo tee /etc/apt/sources.list.d/helm-stable-debian.list

sudo apt-get update

sudo apt-get install helm







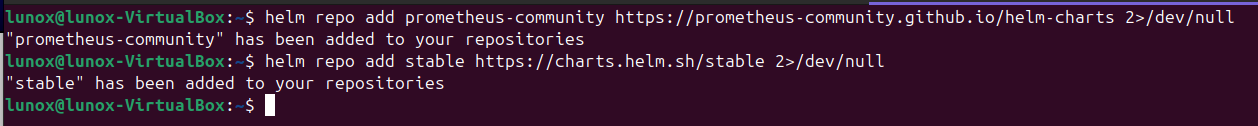
Prometheus cần có một kho lưu trữ liên tục (persistent storage). Sử dụng NFS Server. Tuy nhiên với nội dung lab thì ứng dụng là stateless-không cần lưu trữ dữ liệu giữa các lần chạy nên bỏ qua phần cấu hình NFS server.

### Cài đặt Prometheus:

Add repository:

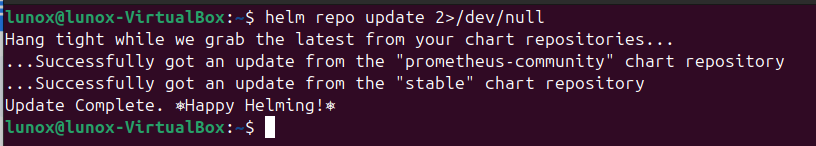
helm repo add prometheus-community https://prometheus-community.github.io/helm-charts

helm repo add stable https://charts.helm.sh/stable



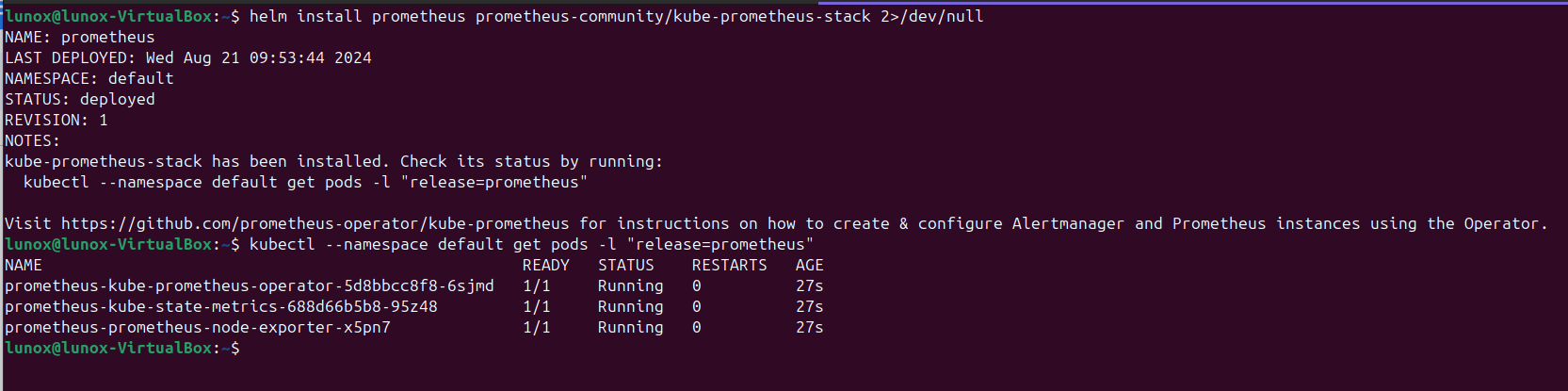
Update Helm repositories:

helm repo update



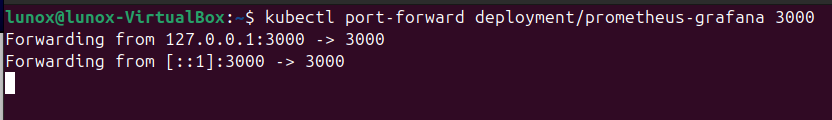
Cài đặt Prometheus Kubernetes:

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



Chuyển tiếp cổng Kubernetes Prometheus:

kubectl port-forward deployment/prometheus-grafana 3000

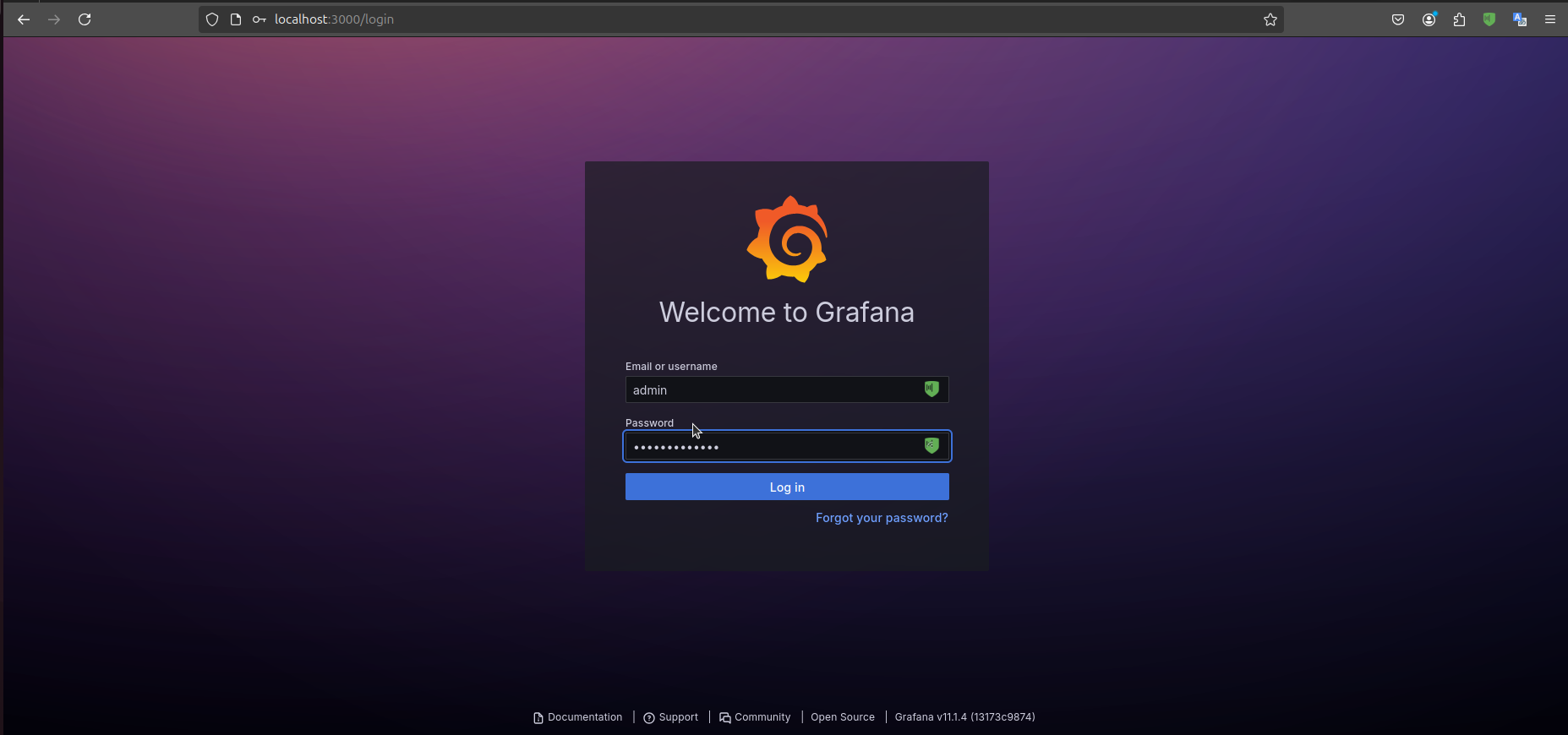


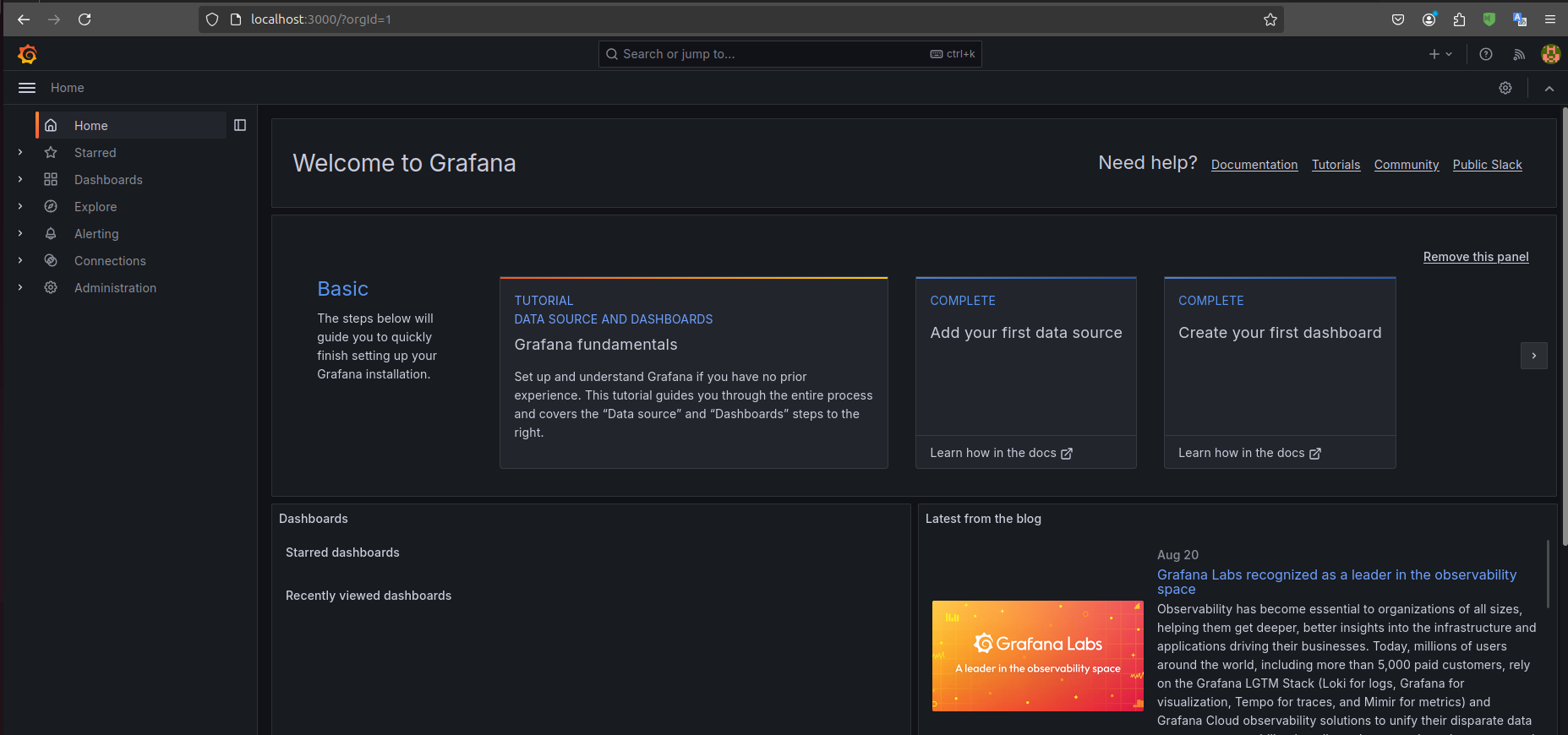
Đăng nhập web-gui grafana:

<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:



Chỉnh sửa values.yaml:

nano values.yaml