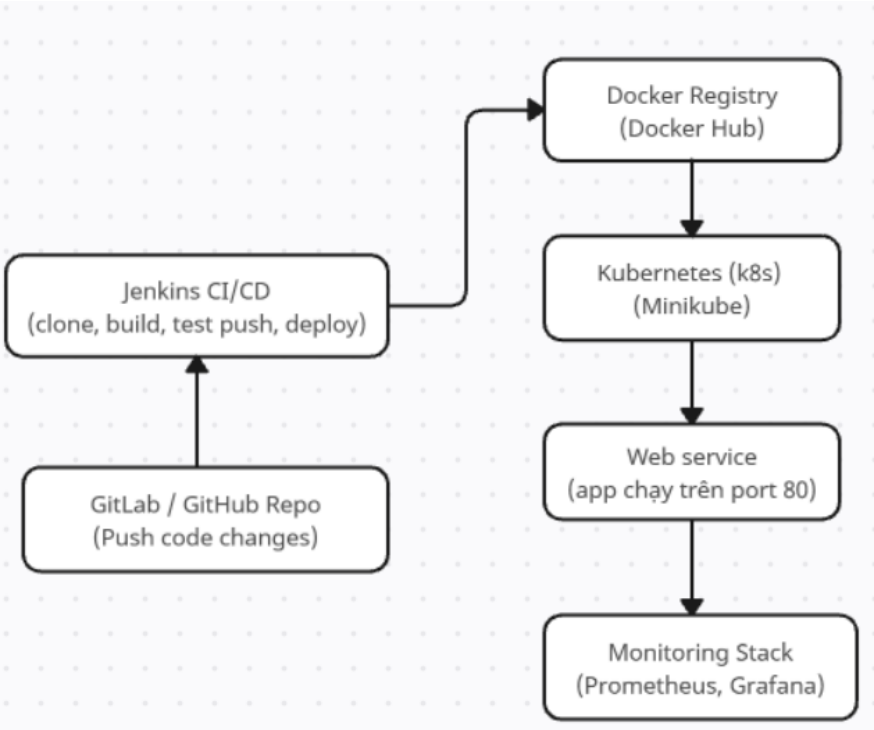
CASE STUDY #1: *Thực hành quy trình DevOps minh họa việc triển khai ứng dụng Website đơn giản ("Hello, CI/CD!") sử dụng các công cụ, công nghệ, dịch vụ: Git, GitLab/GitHub; Docker, Jenkins, K8s, Prometheus & Grafana.*

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 hoá 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 hoá (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 Mini**ku**be 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:

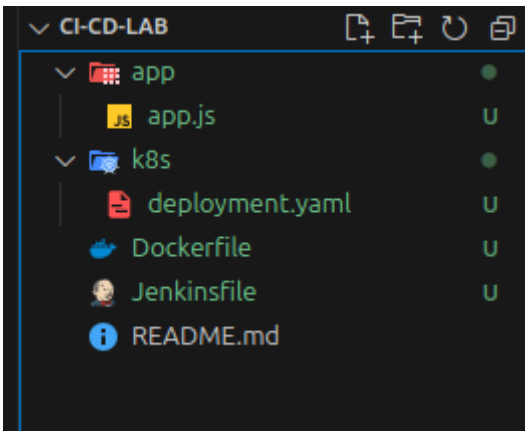
Môi trường thực hiện:

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:



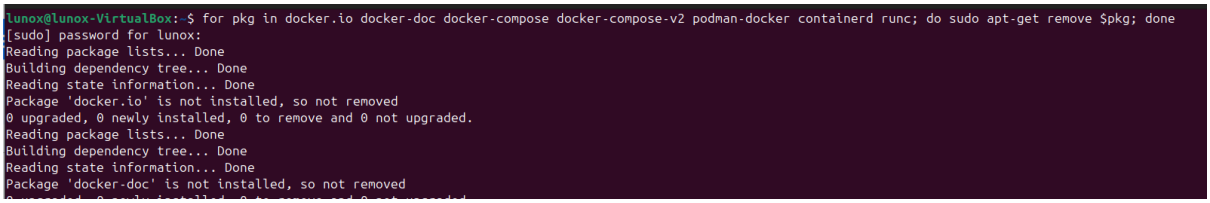
- Trong file app.js code một web đơn giản sử dụng nodejs và express.

Bước 2: Cài docker - Tạo Dockerfile

1. Cài đặt docker engine.

Gỡ các gói gây xung đột:

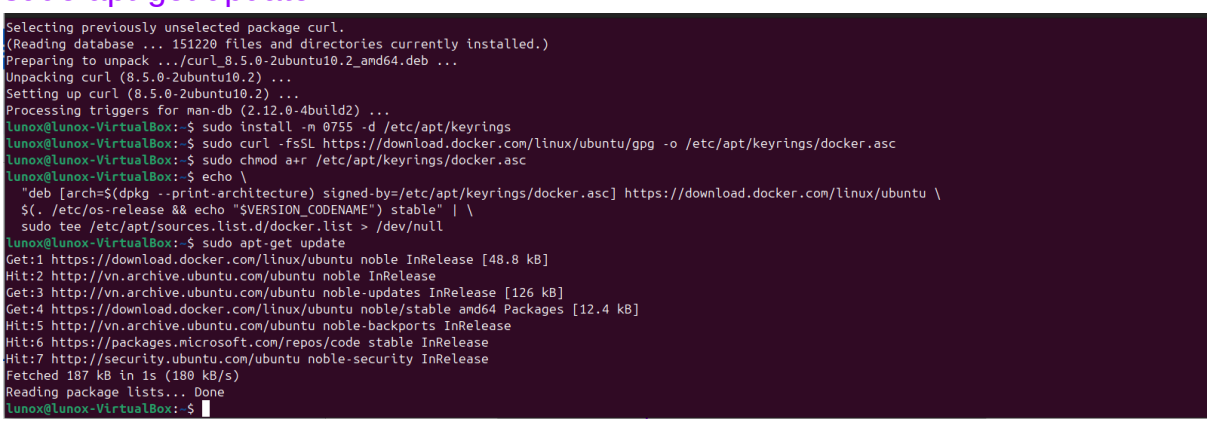
for pkg in docker.io docker-doc docker-compose docker-compose-v2 podman-docker containerd runc; do sudo apt-get remove $pkg; done



Cài đặt docker repository:

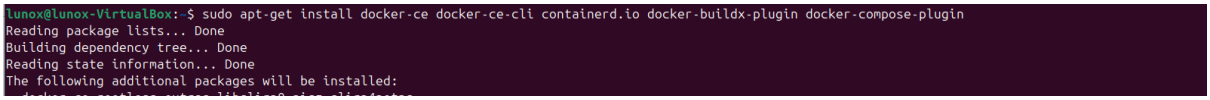
# Add Docker's official GPG key:
sudo apt-get update
sudo apt-get install ca-certificates curl
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc

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



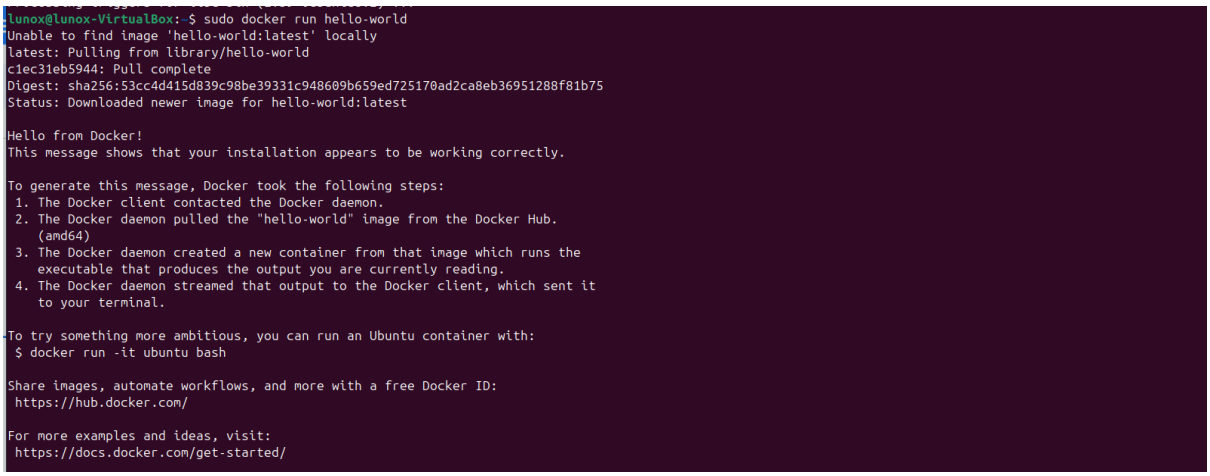
Cài đặt docker engine:

```
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
```

2. 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"]
```

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
```

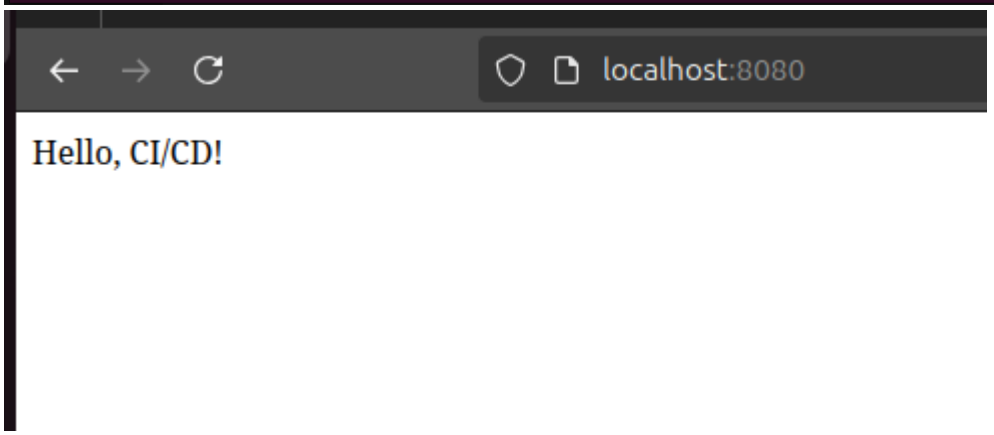
3. Build and test:

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

```
lunox@lunox-VirtualBox:~/Desktop/DevOps/ci-cd-lab$ docker build -t ci-cd-lab-app .
[+] Building 47.5s (10/10) FINISHED                                                    docker:default
 => [internal] load build definition from Dockerfile                                          0.0s
 => => transferring dockerfile: 619B                                                          0.0s
 => [internal] load metadata for docker.io/library/node:14                                    4.4s
 => [internal] load .dockerignore                                                             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
```



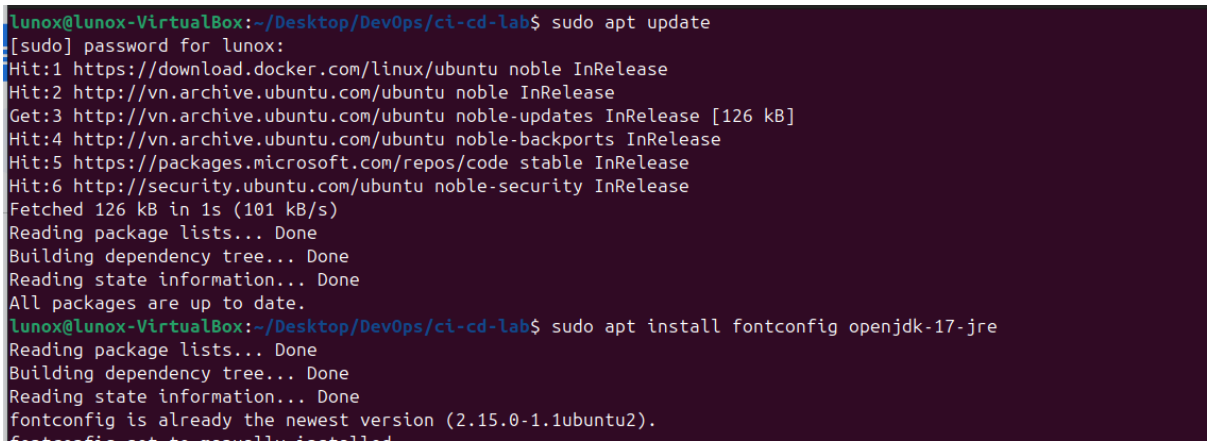
Bước 3: Cài đặt và cấu hình Jenkins pipeline.

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
```



Cài đặt jenkins:

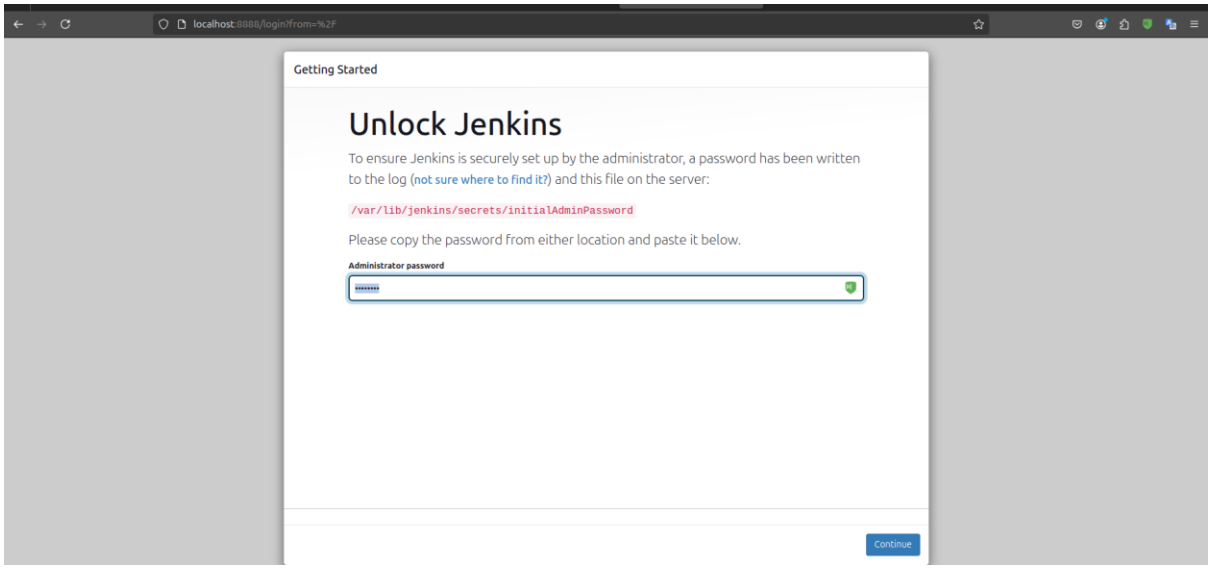
```
sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \
  https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key
echo "deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc]" \
  https://pkg.jenkins.io/debian-stable binary/ | sudo tee \
  /etc/apt/sources.list.d/jenkins.list > /dev/null
sudo apt-get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

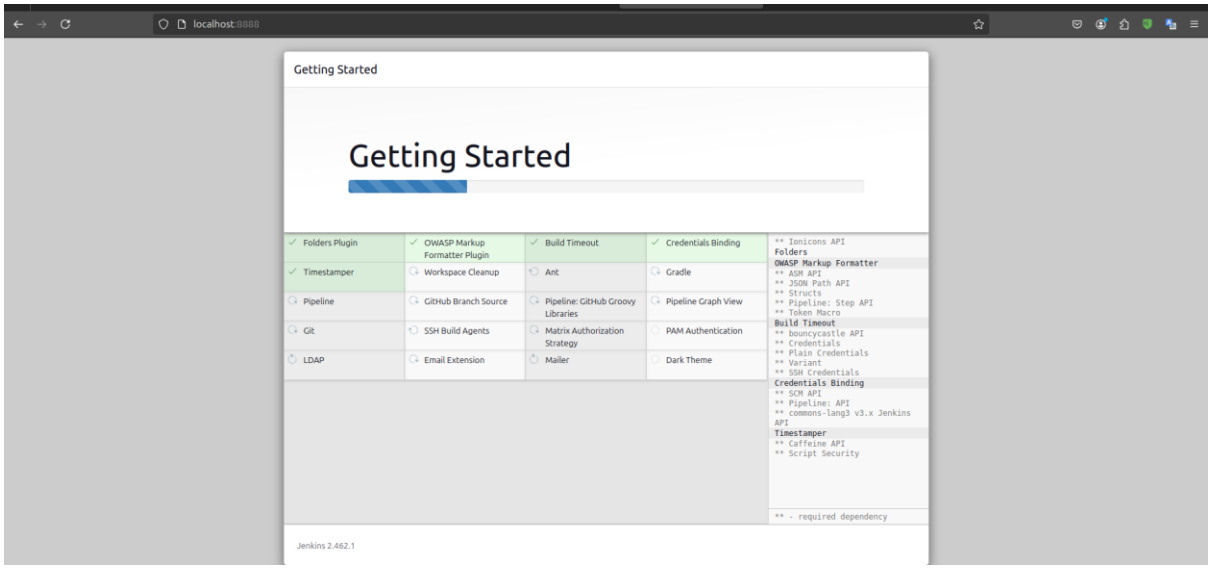
Thay đổi cổng mặc định của jenkins:

```
  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.
```

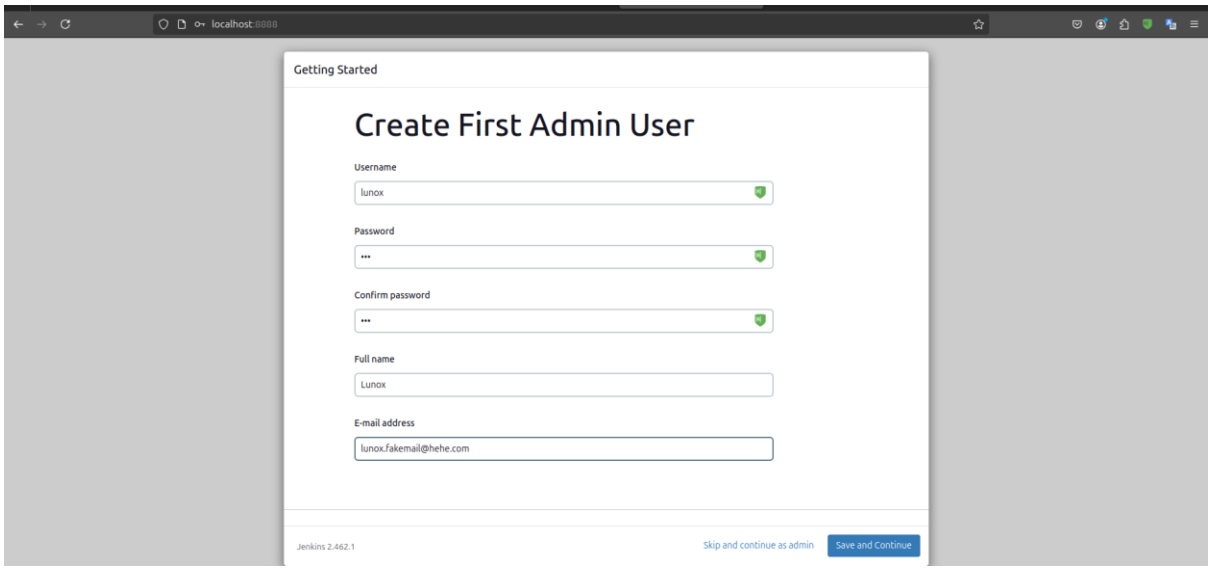
Truy cập web GUI của jenkins qua url: http://localhost:8888
Mật khẩu đăng nhập lần đầu lấy ở:
/var/lib/jenkins/secrets/initialAdminPassword

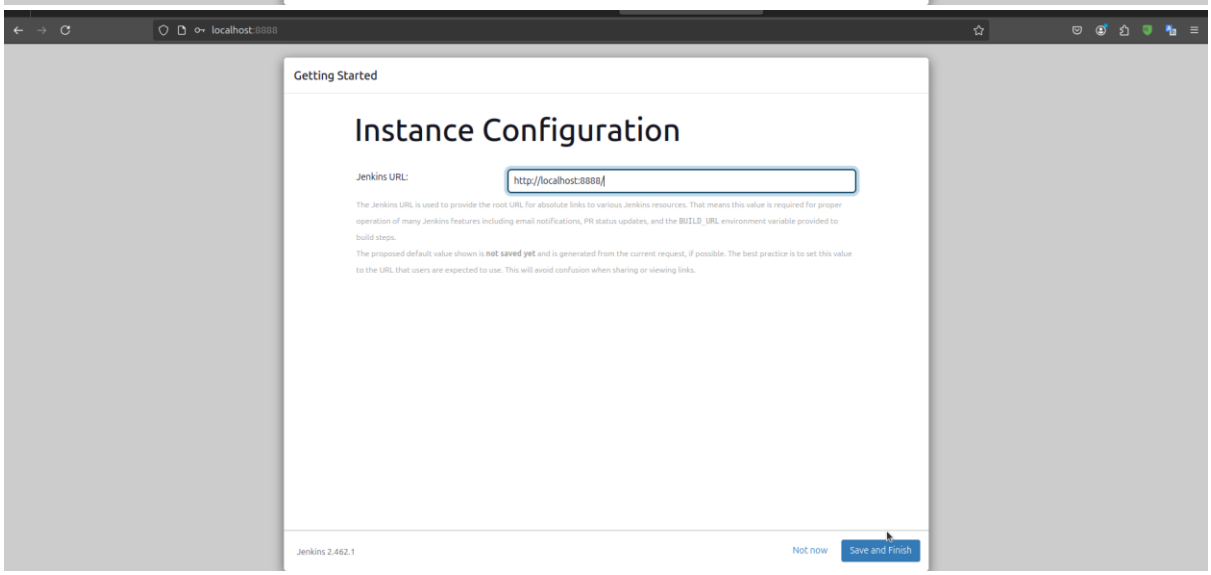


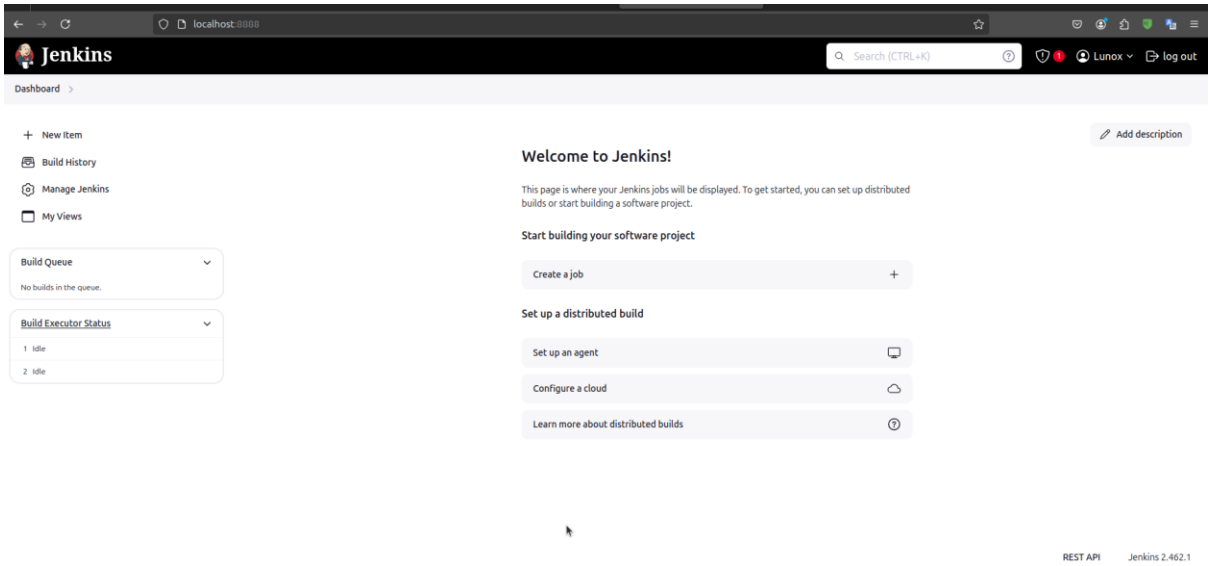
Chọn install suggestion plugins để tiếp tục



Tạo tài khoản jenkins để đăng nhập lần sau:

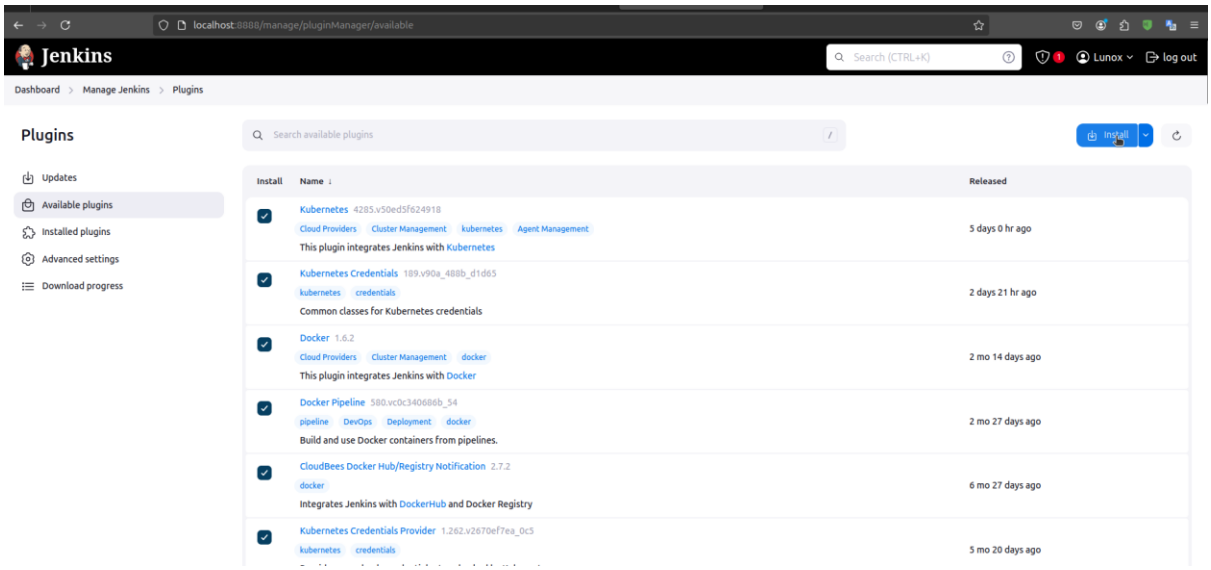


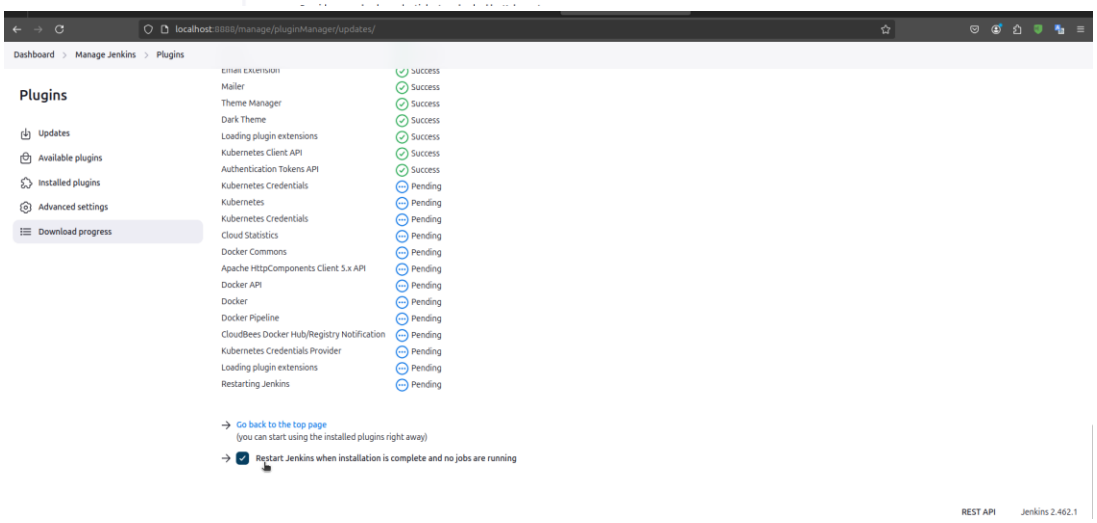




Cài đặt thành công.

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

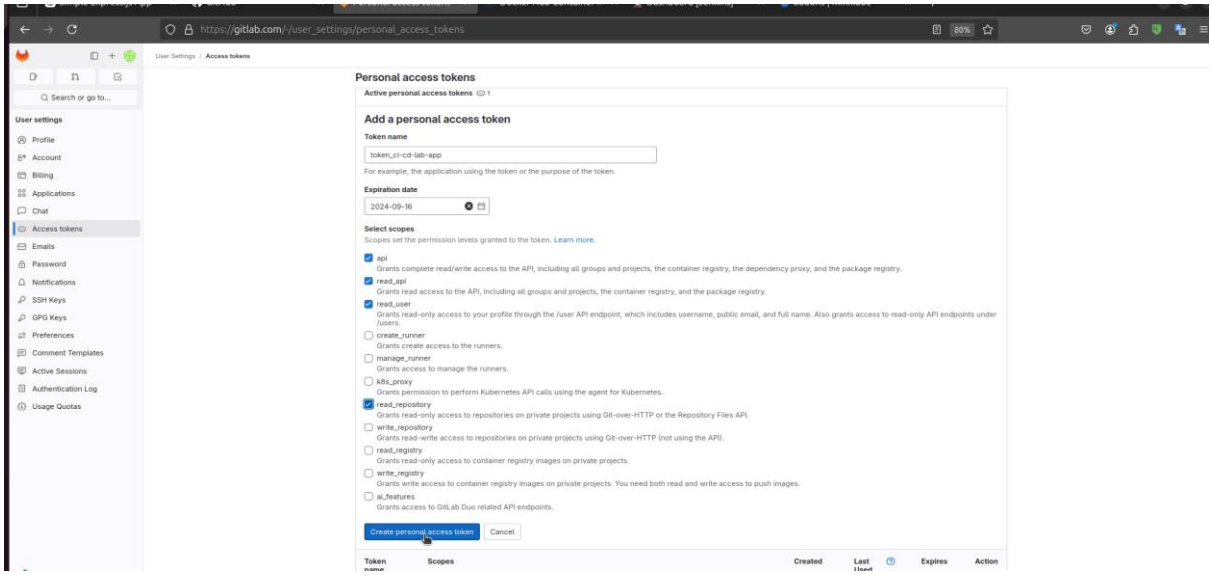




2. Thêm các credential:

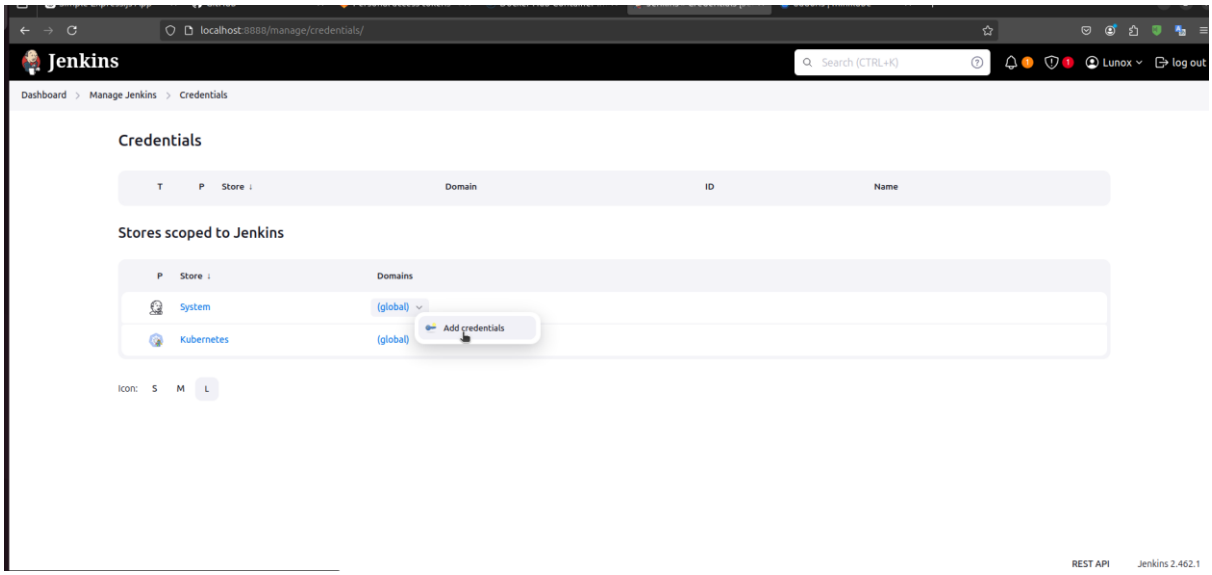
a. Gitlab credential:

Truy cập: https://gitlab.com/-/user_settings/personal_access_tokens
Chọn Add new token.

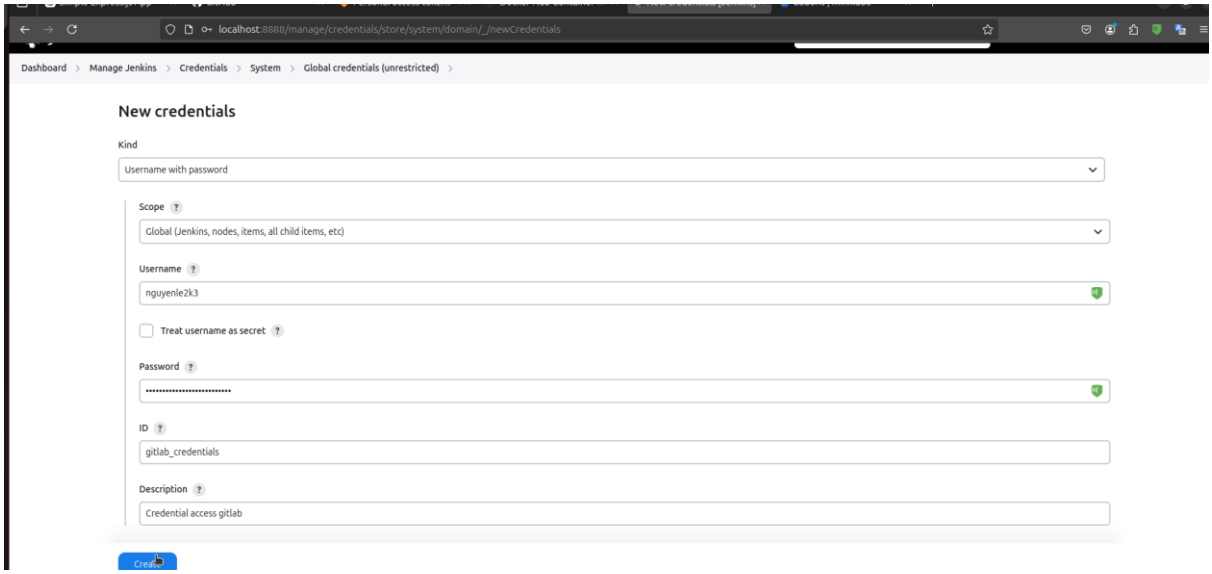


copy token và lưu lại trong một file bí mật.

Quay lại web gui jenkins. > Manage Jenkins > Credentials.
Add credentials:

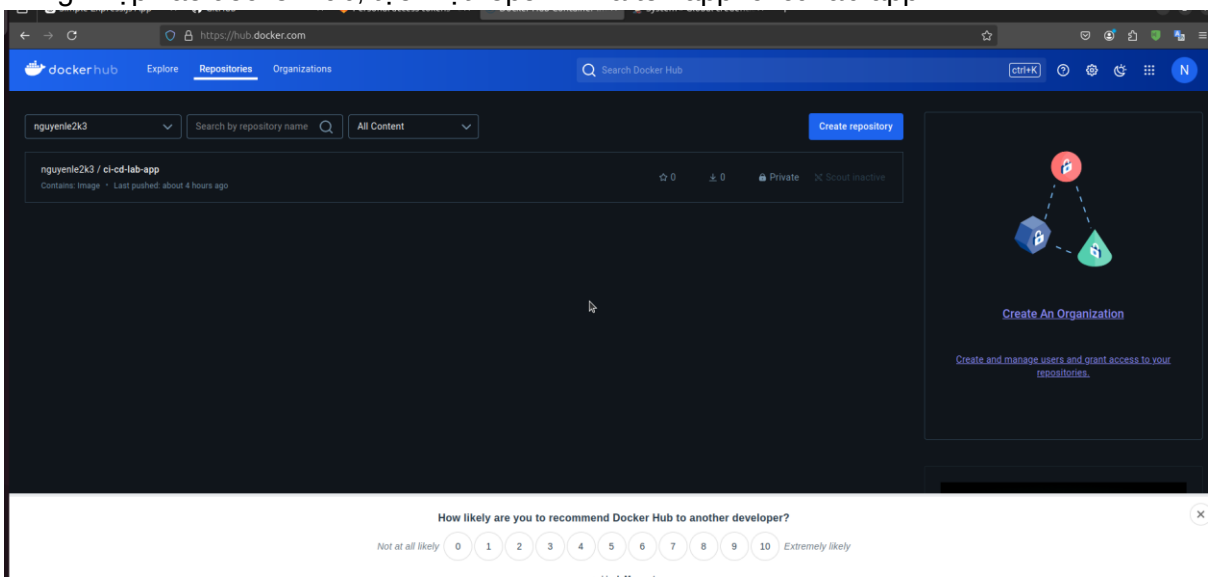


Ở phần password điền access token vừa tạo ở gitlab.



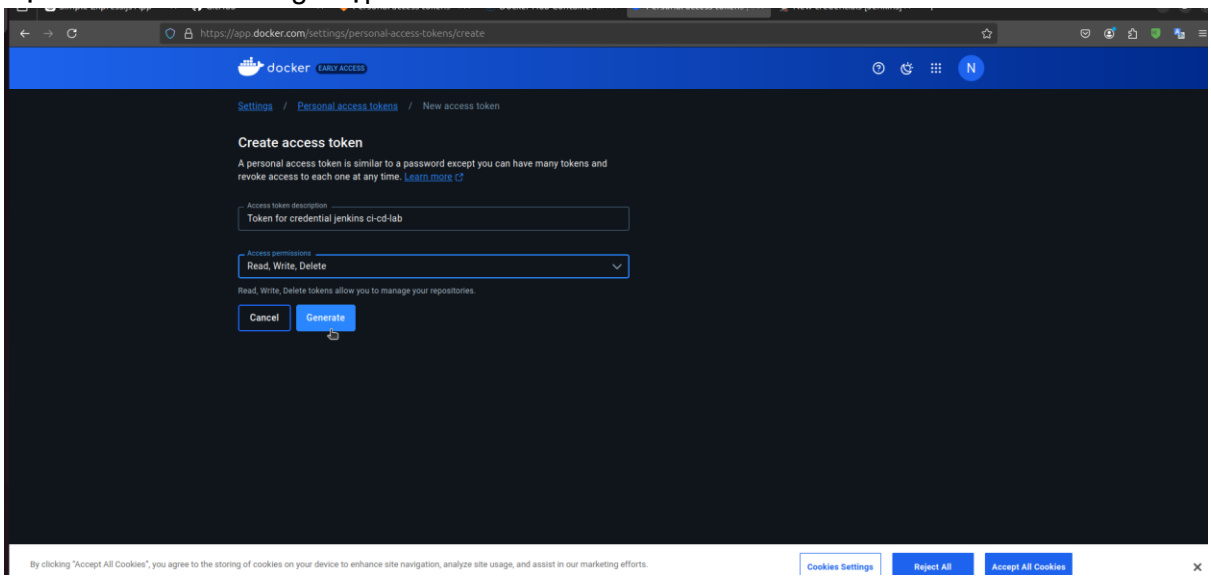
b. Docker hub credential:

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

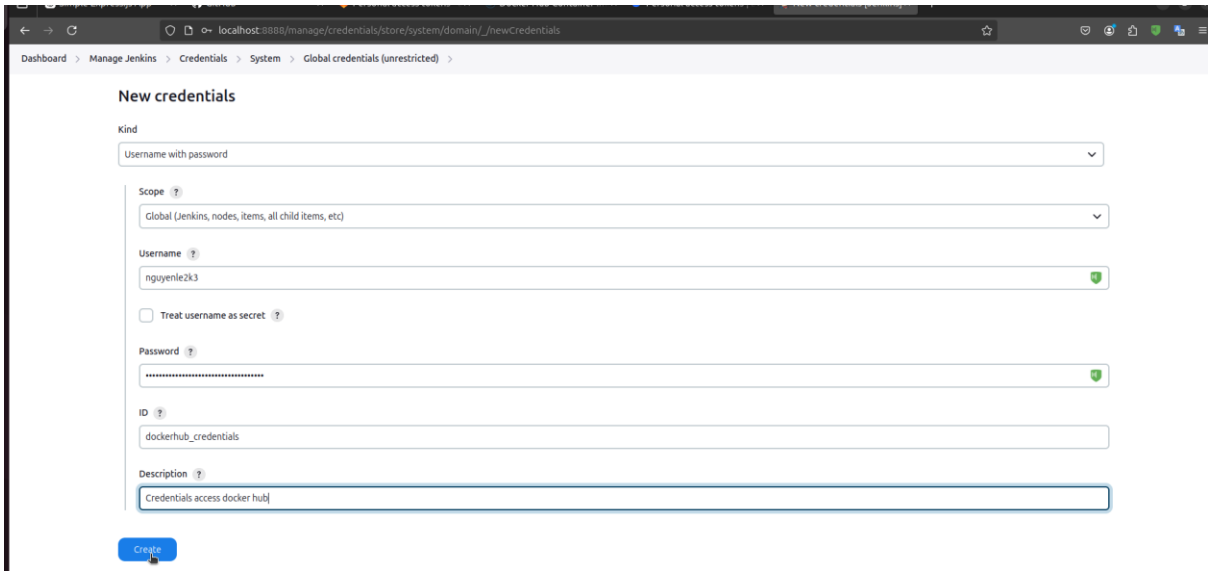


Tương tự với tạo credential gitlab, tạo một credentials cho docker hub.

Tạo access token đăng nhập



Tạo credentials:

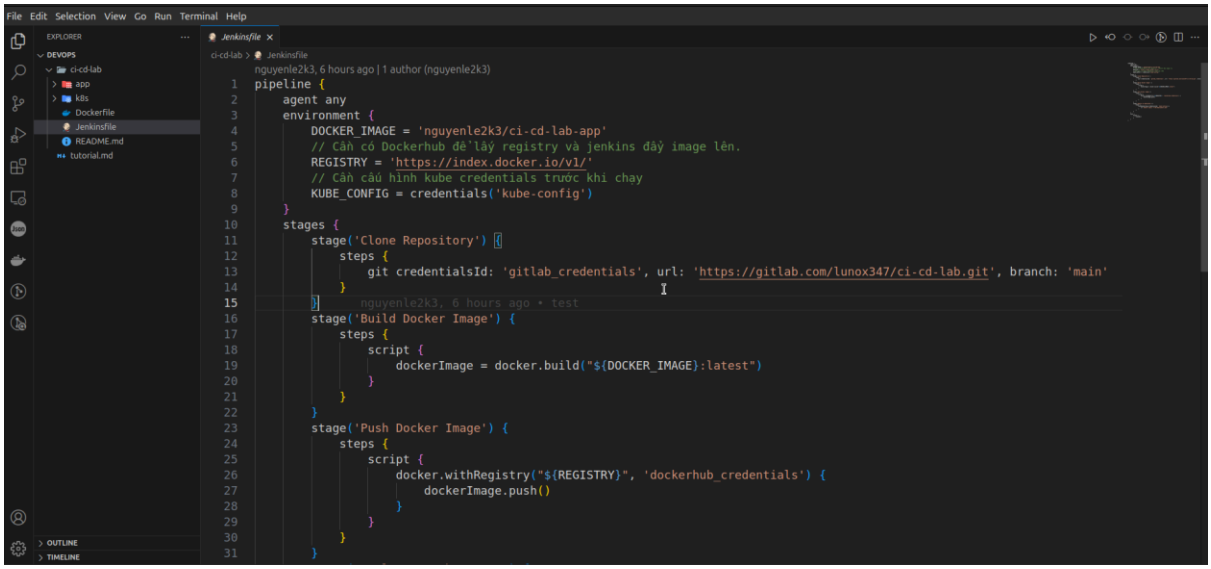


3. Tạo Jenkinsfile.

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()
        }
        }
}
```

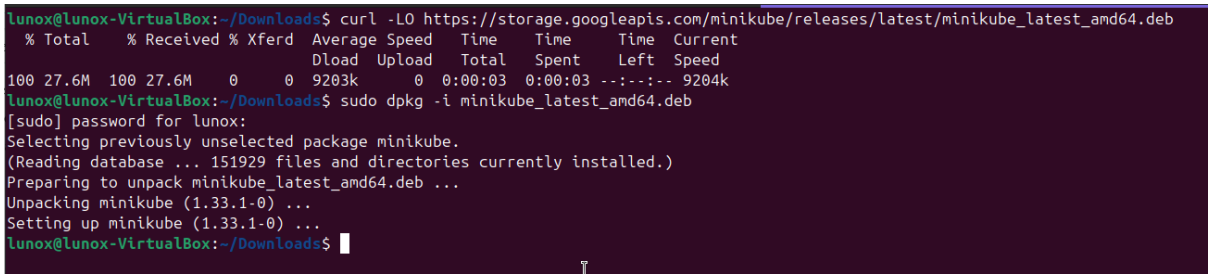


Bước 4: Cài đặt, cấu hình K8s - Minikube và tạo deployment:

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



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
```

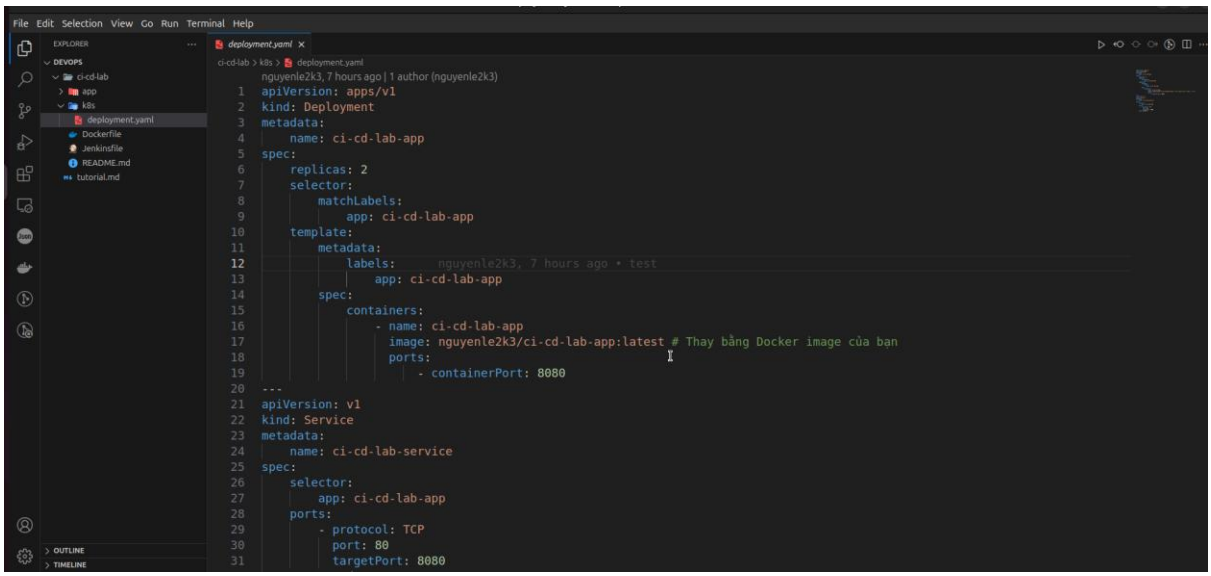
Kiểm tra cluster:
kubectl get nodes

```
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:
```

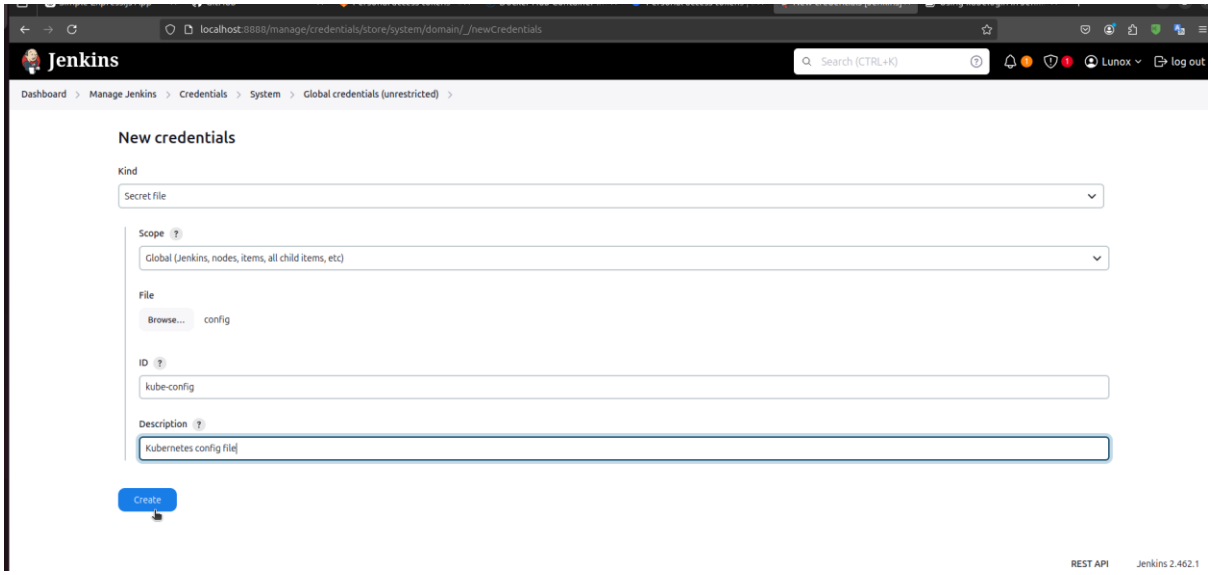
```
        app: ci-cd-lab-app
        template:
        metadata:
        labels:
                app: ci-cd-lab-app
        spec:
        containers:
                - name: ci-cd-lab-app
                image: nguyenle2k3/ci-cd-lab-app:latest # Thay bằng Docker
image của bạn
                ports:
                - 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
```



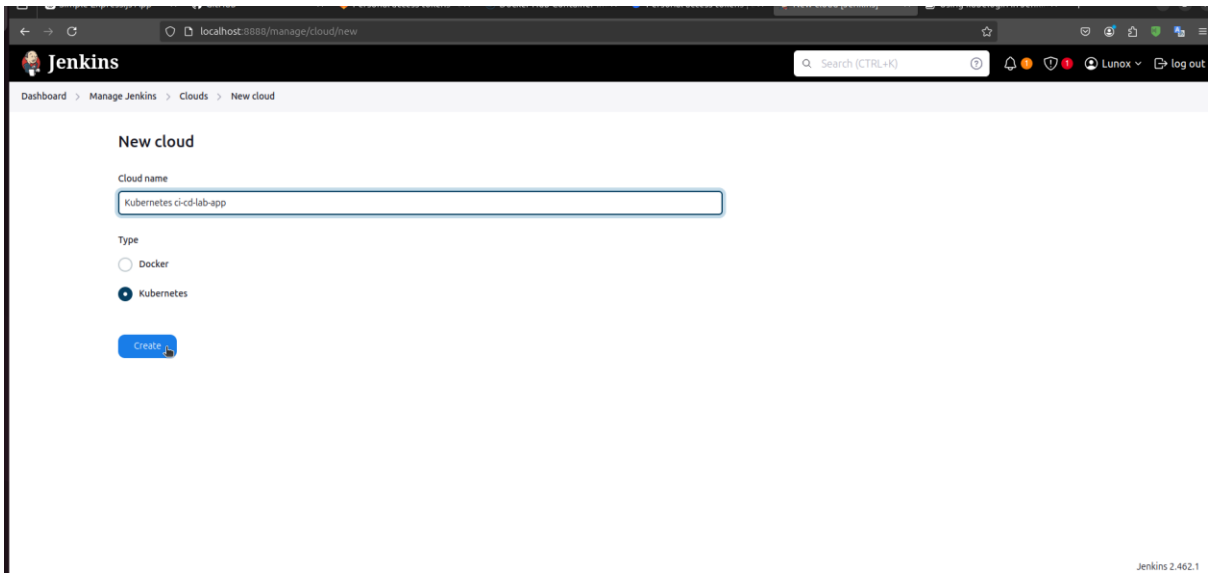
Bước 5: Tích hợp Jenkins với Kubernetes

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

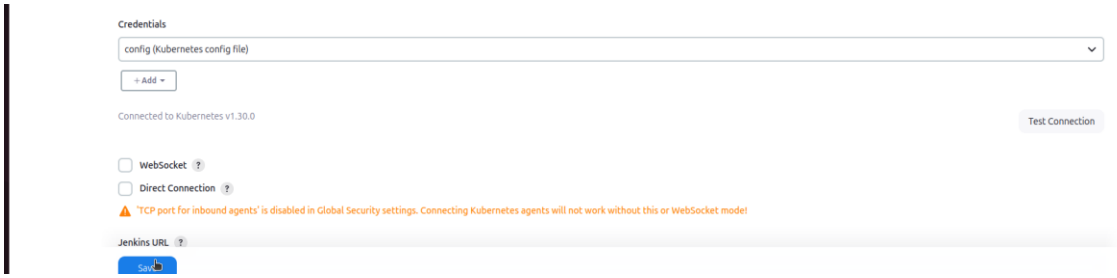
Add credentials kube-config trong jenkins:



Add cloud kubernetes trong jenkins:

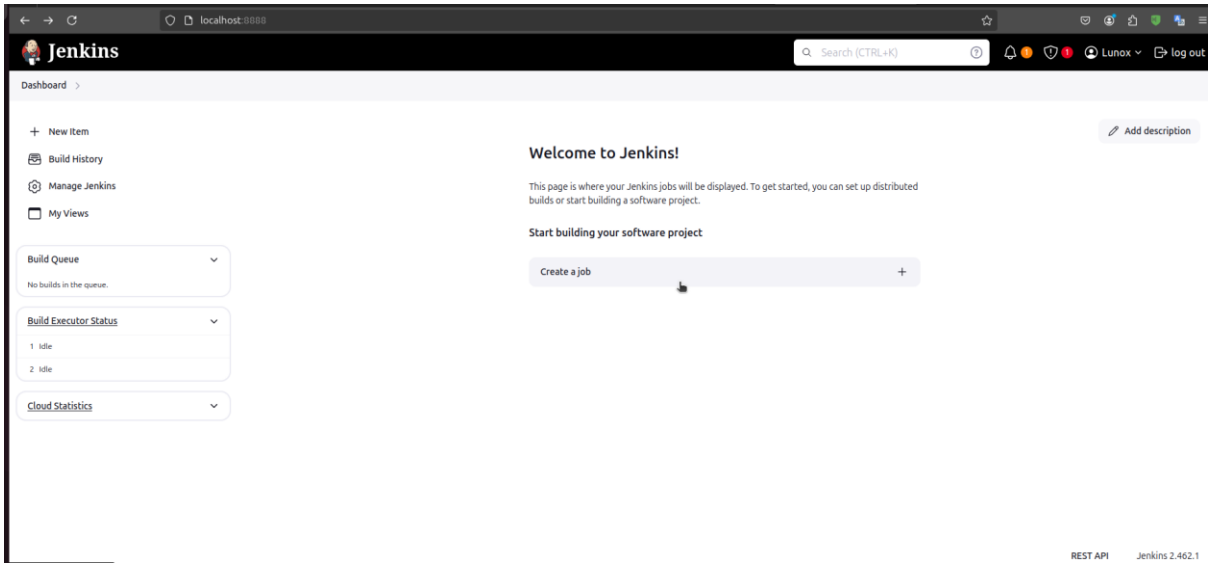


add Credentials và test connection để chắc chắn credentials hoạt động đúng.

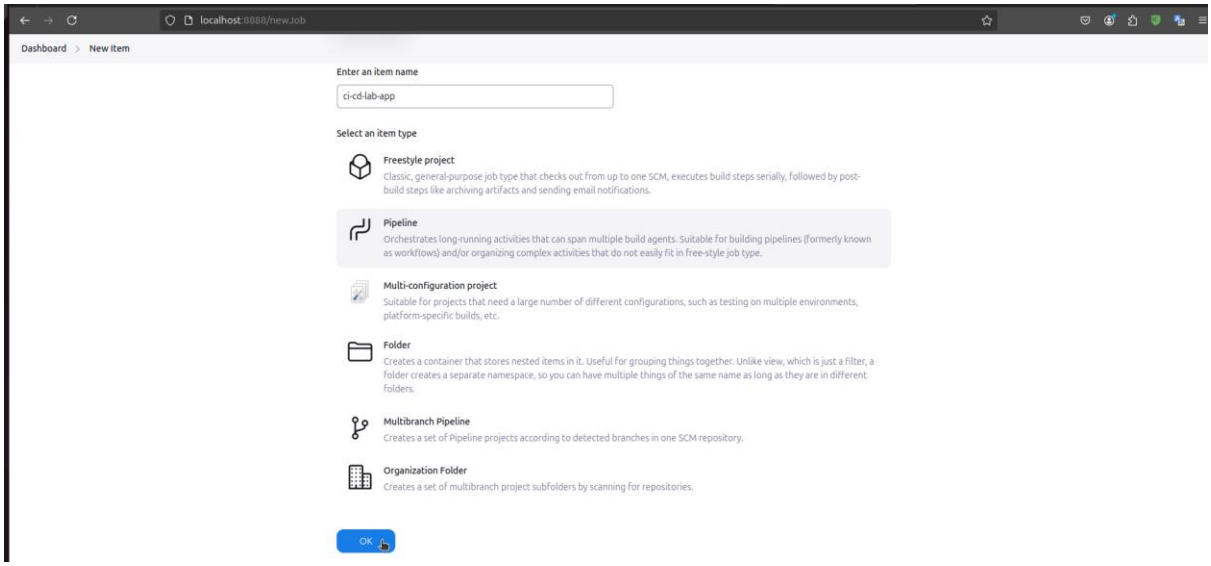


Cơ bản đã hoàn thành quá trình cấu hình. Giờ tạo một job để test.

Quay lại Dashboard > Create a job.



Nhập tên job, chọn pipeline.



Kéo xuống phần pipeline.

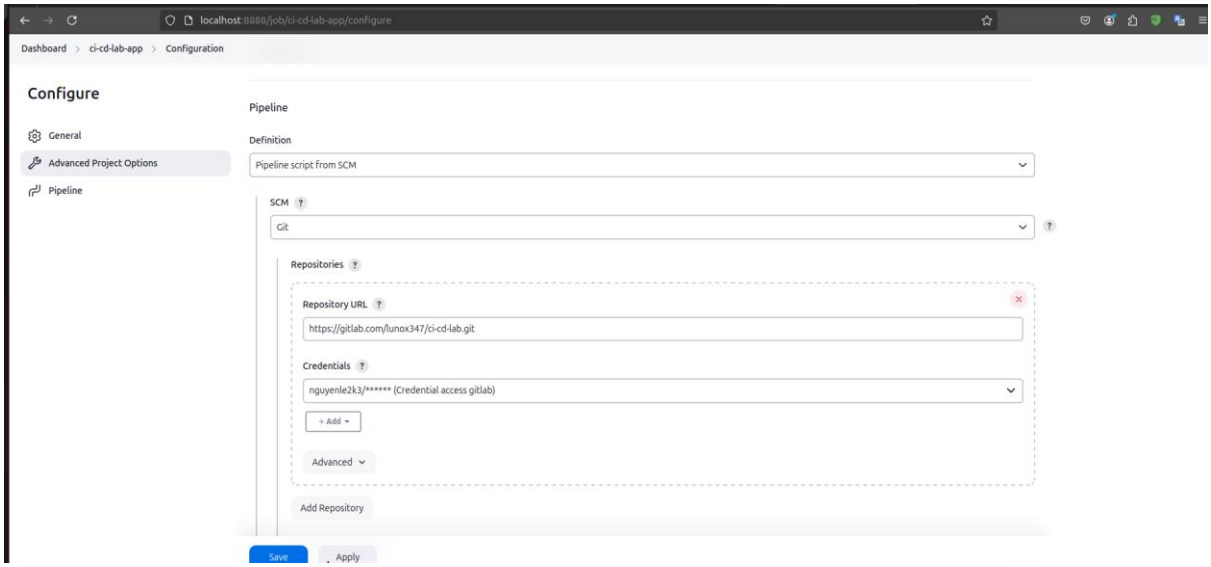
Definition: Pipeline script from SCM

SCM: Git

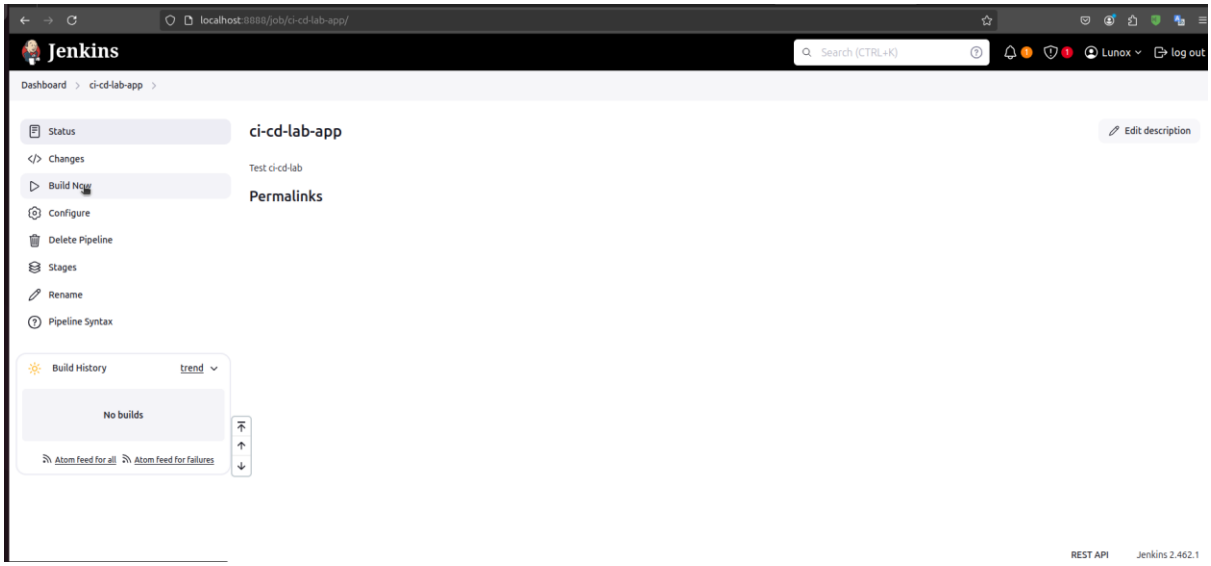
Repository URL: URL repo git lab

Credential: gitlab_credentials

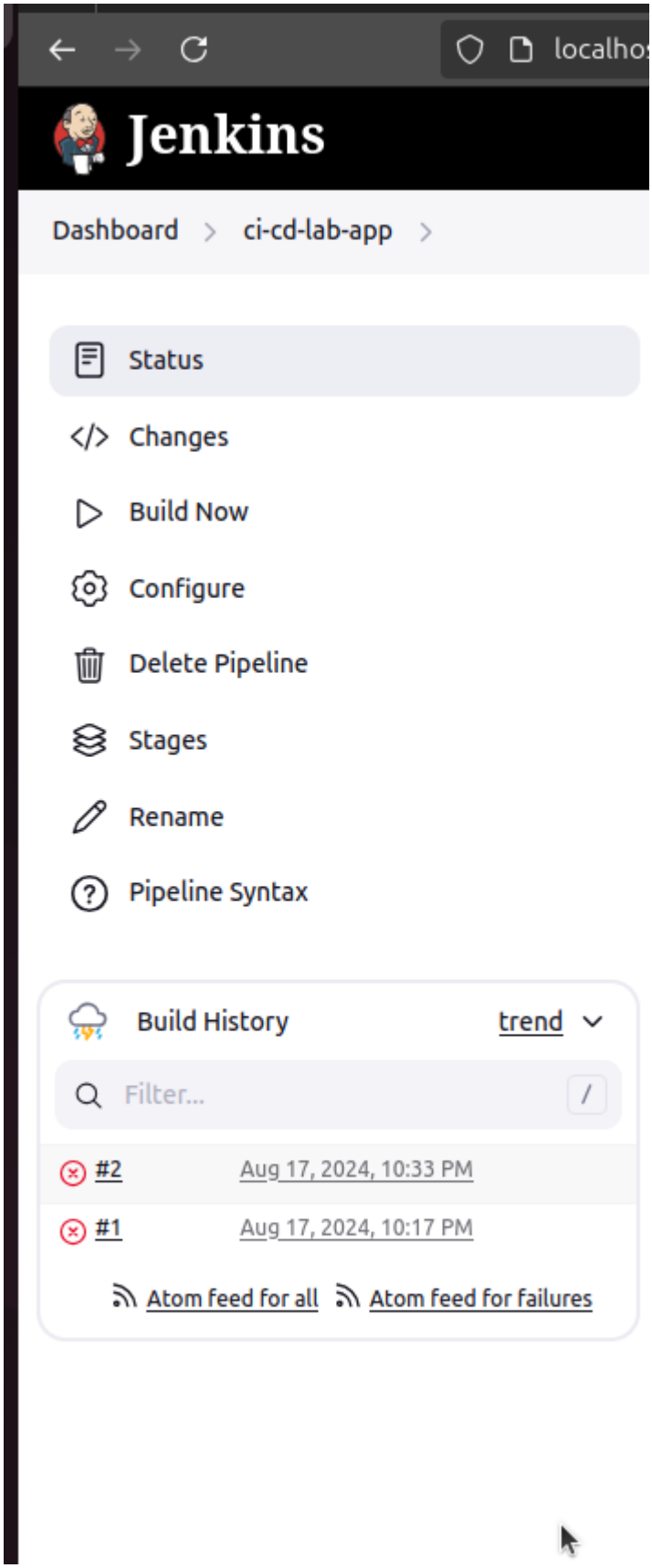
Script path: Jenkinsfile



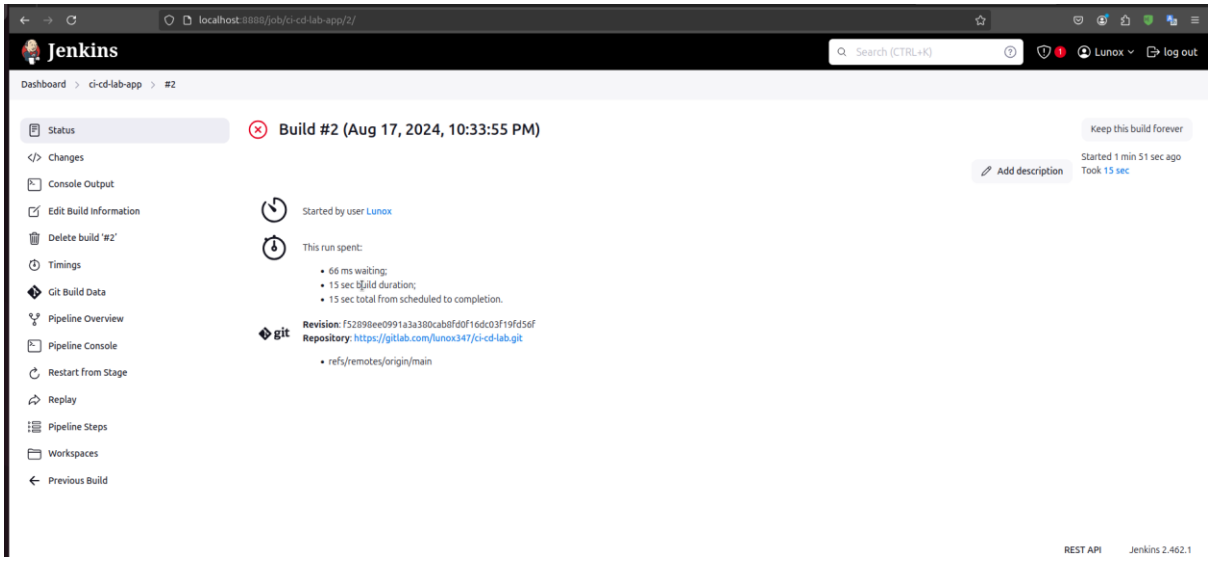
Chọn Build now để build app.



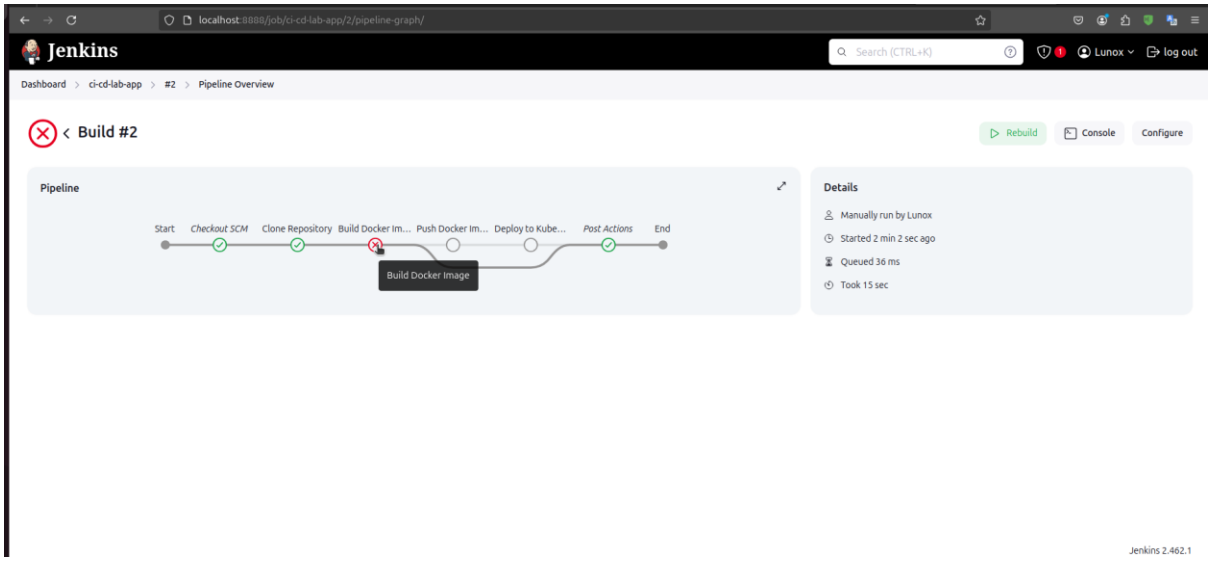
Lịch sử các lượt build:


localhos
Jenkins
Dashboard
ci-cd-lab-app
Status
Changes
Build Now
Configure
Delete Pipeline
Stages
Rename
Pipeline Syntax
Build History
trend
Filter...
/
#2
Aug 17, 2024, 10:33 PM
#1
Aug 17, 2024, 10:17 PM
Atom feed for all
Atom feed for failures

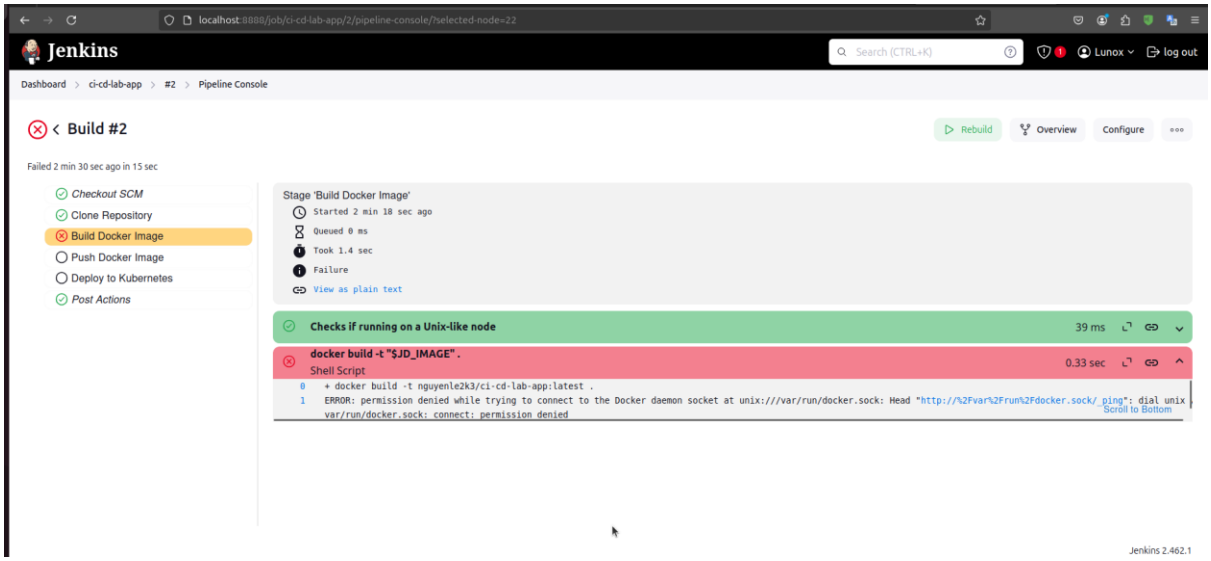
Bấm vào #2 để xem chi tiết về lượt build



Ta thấy build hỏng từ bước Build Docker Image



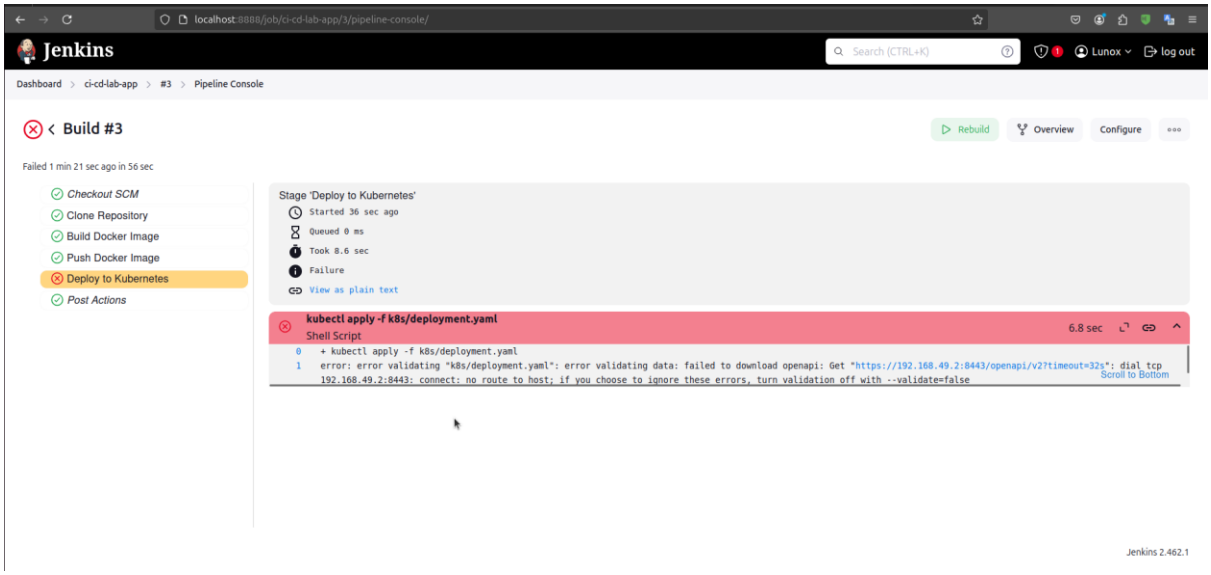
Xem chi tiết lỗi:



Lỗi này do chưa add user jenkins vào group docker.

```
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:~$
```

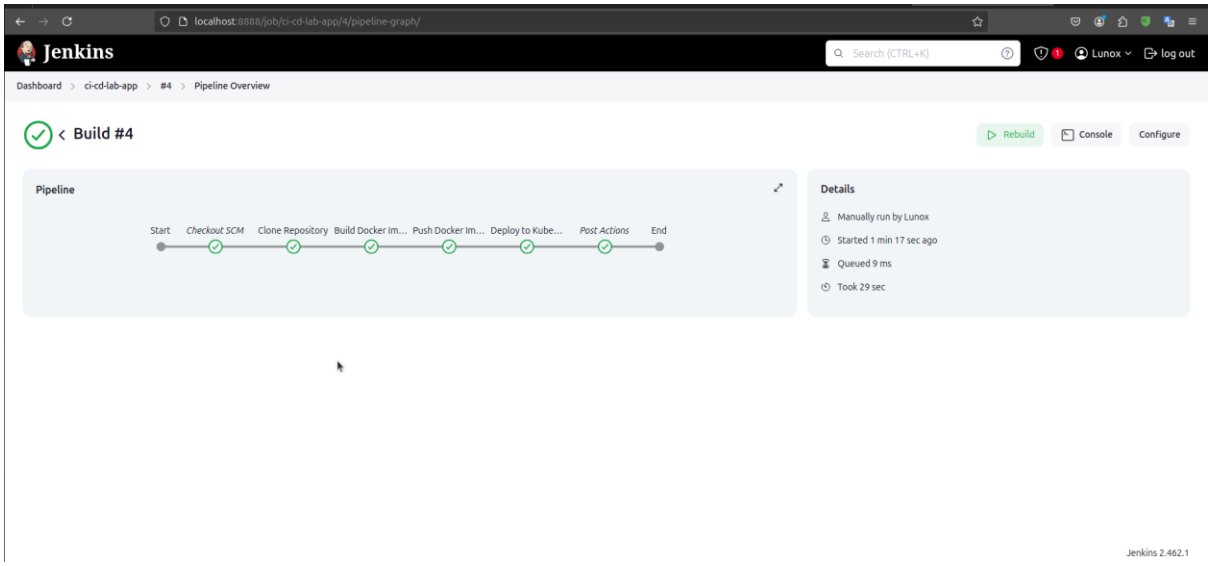
Build lại:
Lỗi ở bước Deploy:



Lỗi gây ra do minikube bị tắt:

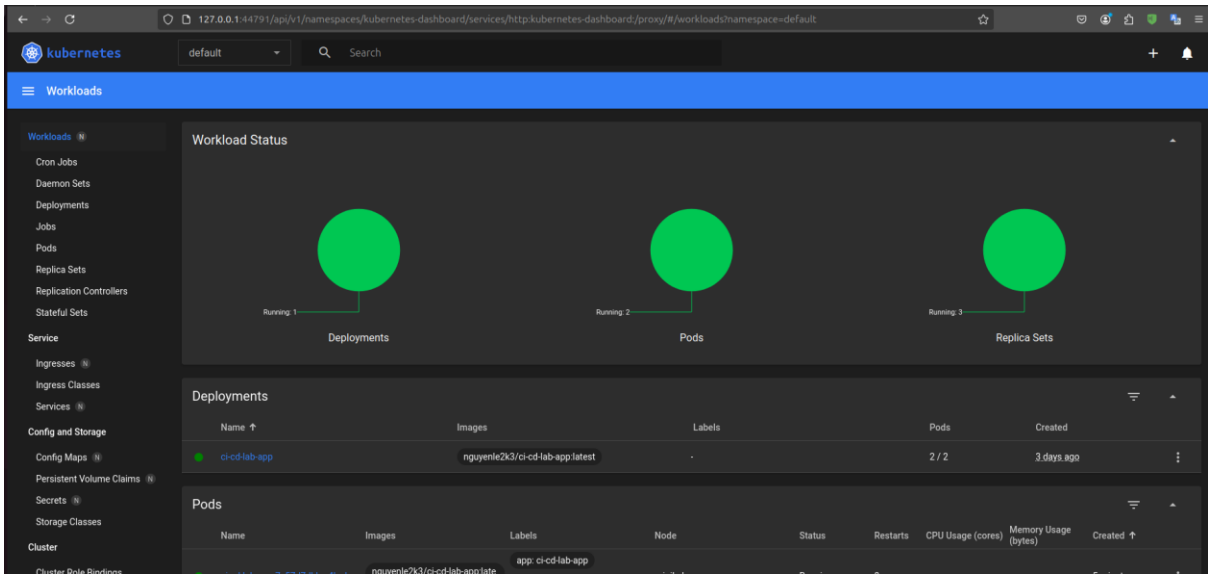
```
lunox@lunox-VirtualBox:~$ minikube status
minikube
type: Control Plane
host: Stopped
kubelet: Stopped
apiserver: Stopped
kubeconfig: Stopped

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 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:~$
```

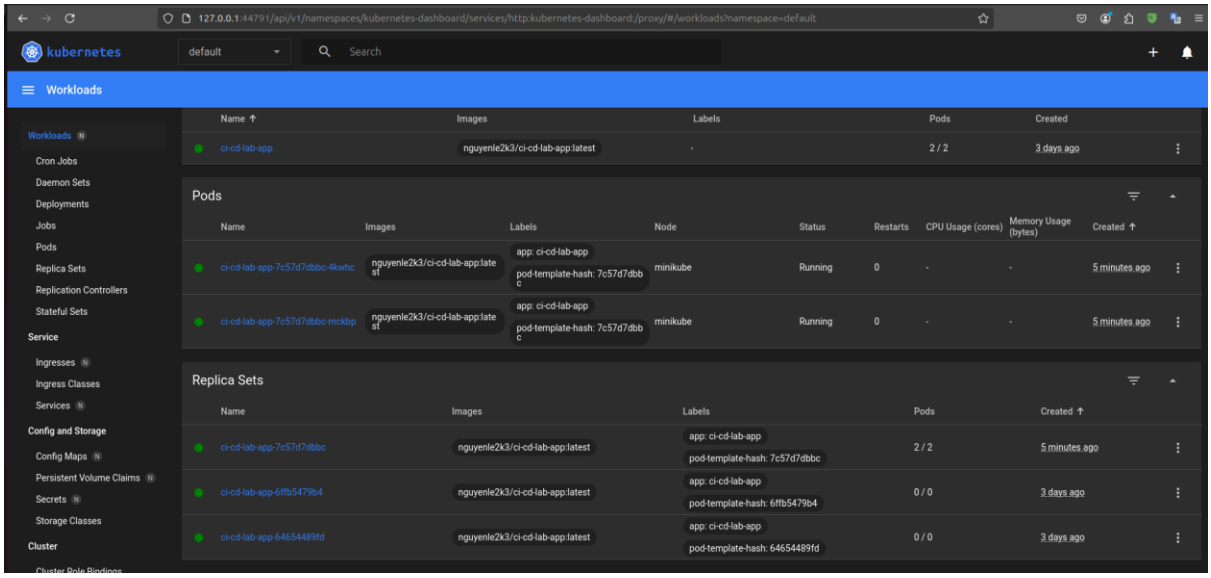
Rebuild:

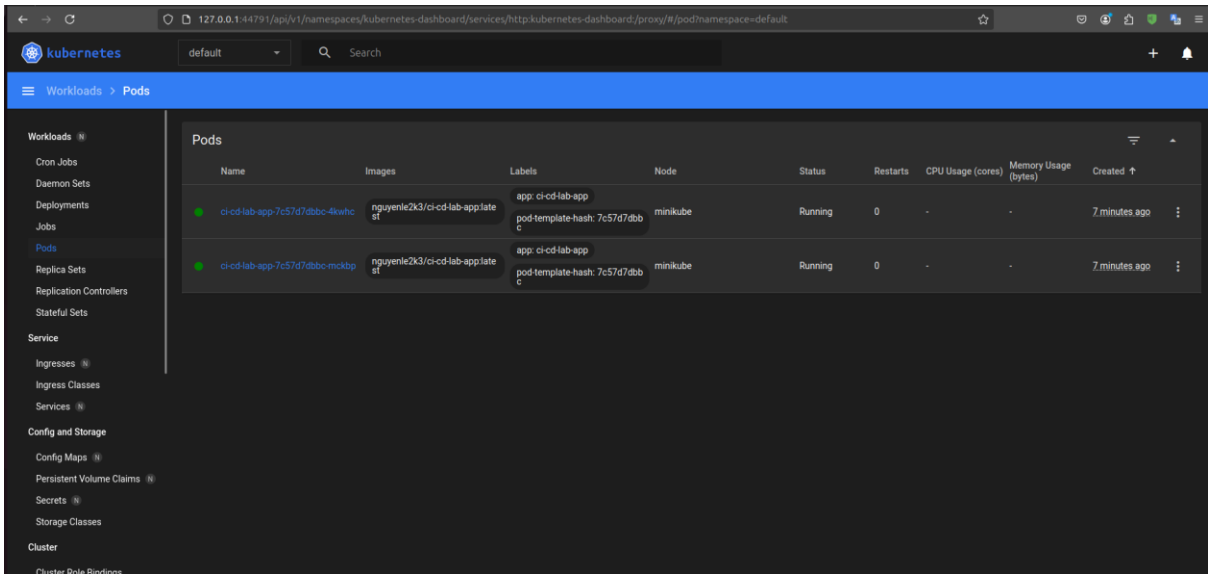


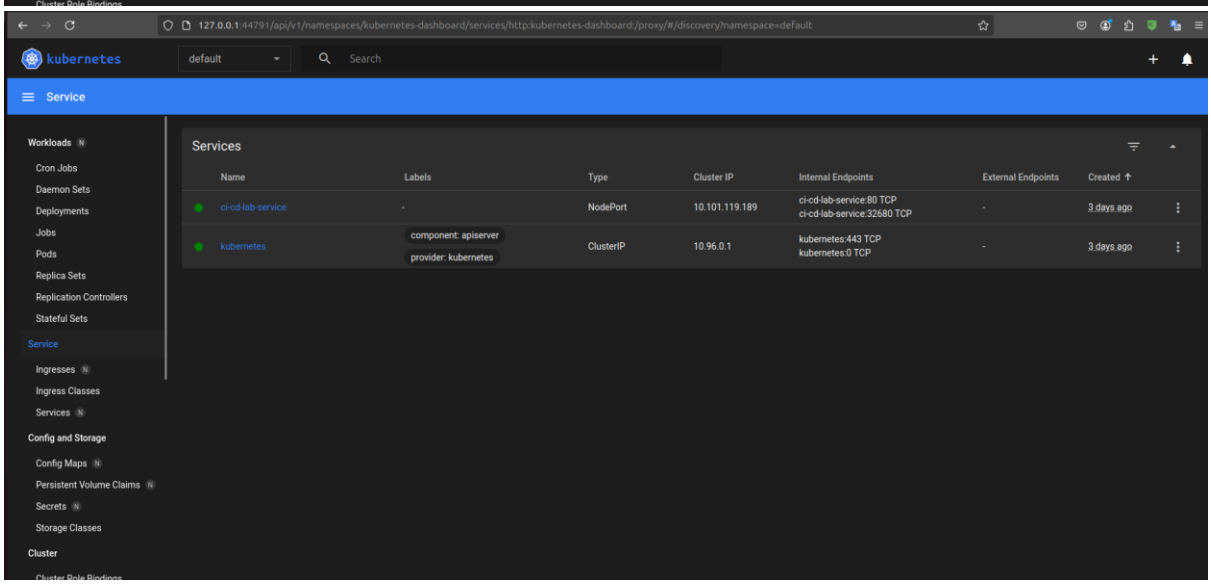
Thành công.

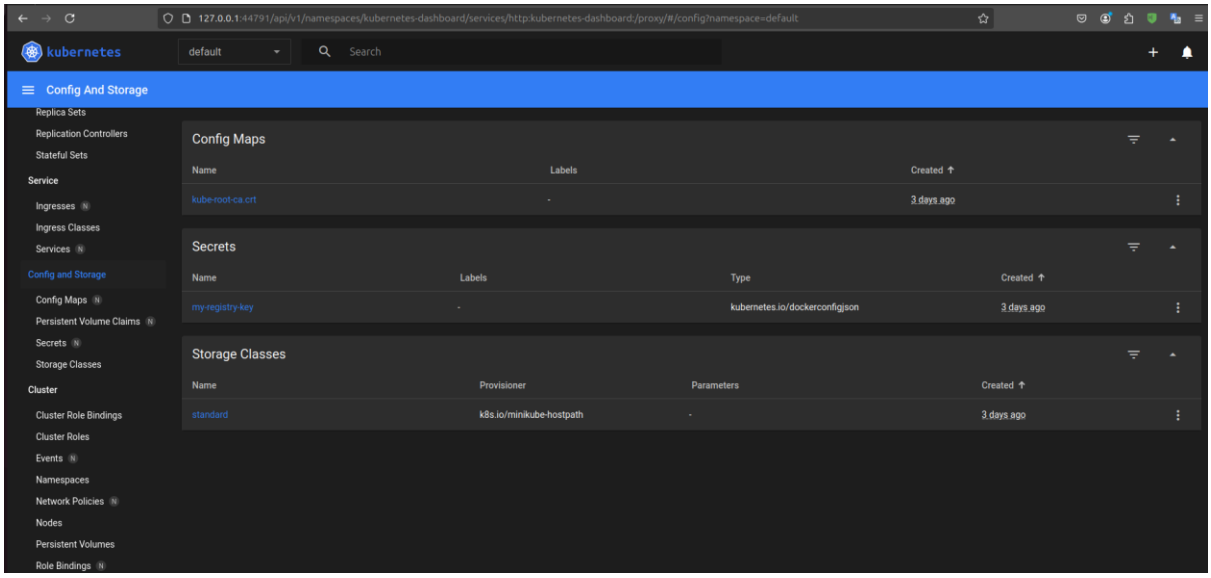
Có thể sử dụng minikube dashboard để kiểm tra toàn diện minikube với web gui.

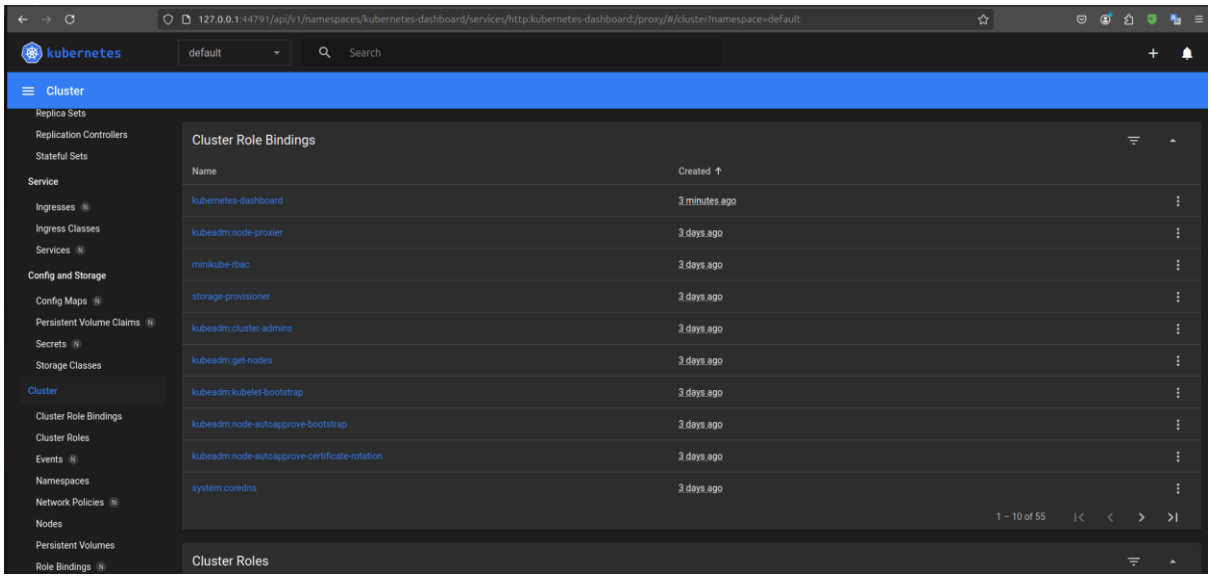



127.0.0.1:44791/api/v1/namespaces/kubernetes-dashboard/services/http:kubernetes-dashboard:/proxy/#/workloads?namespace=default
kubernetes
default
Search
Workloads
Workloads
Cron Jobs
Daemon Sets
Deployments
Jobs
Pods
Replica Sets
Replication Controllers
Stateful Sets
Service
Ingresses
Ingress Classes
Services
Config and Storage
Config Maps
Persistent Volume Claims
Secrets
Storage Classes
Cluster
Name
Images
Labels
Pods
Created
ci-cd-lab-app
nguyenle2k3/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
nguyenle2k3/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
nguyenle2k3/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
nguyenle2k3/ci-cd-lab-app:latest
app: ci-cd-lab-app
pod-template-hash: 7c57d7dbbc
2 / 2
5 minutes ago
ci-cd-lab-app-6ffb5479b4
nguyenle2k3/ci-cd-lab-app:latest
app: ci-cd-lab-app
pod-template-hash: 6ffb5479b4
0 / 0
3 days ago
ci-cd-lab-app-64654489fd
nguyenle2k3/ci-cd-lab-app:latest
app: ci-cd-lab-app
pod-template-hash: 64654489fd
0 / 0
3 days ago


127.0.0.1:44791/api/v1/namespaces/kubernetes-dashboard/services/http:kubernetes-dashboard:/proxy/#/pod?namespace=default
kubernetes
default
Search
Workloads > Pods
Workloads
Cron Jobs
Daemon Sets
Deployments
Jobs
Pods
Replica Sets
Replication Controllers
Stateful Sets
Service
Ingresses
Ingress Classes
Services
Config and Storage
Config Maps
Persistent Volume Claims
Secrets
Storage Classes
Cluster
Pods
Name
Images
Labels
Node
Status
Restarts
CPU Usage (cores)
Memory Usage (bytes)
Created
ci-cd-lab-app-7c57d7dbbc-4kwhc
nguyenle2k3/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
nguyenle2k3/ci-cd-lab-app:latest
app: ci-cd-lab-app
pod-template-hash: 7c57d7dbbc
minikube
Running
0
-
-
7 minutes ago


127.0.0.1:44791/api/v1/namespaces/kubernetes-dashboard/services/http:kubernetes-dashboard:/proxy/#/discovery?namespace=default
kubernetes
default
Search
Service
Workloads
Cron Jobs
Daemon Sets
Deployments
Jobs
Pods
Replica Sets
Replication Controllers
Stateful Sets
Service
Ingresses
Ingress Classes
Services
Config and Storage
Config Maps
Persistent Volume Claims
Secrets
Storage Classes
Cluster
Services
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





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>
```

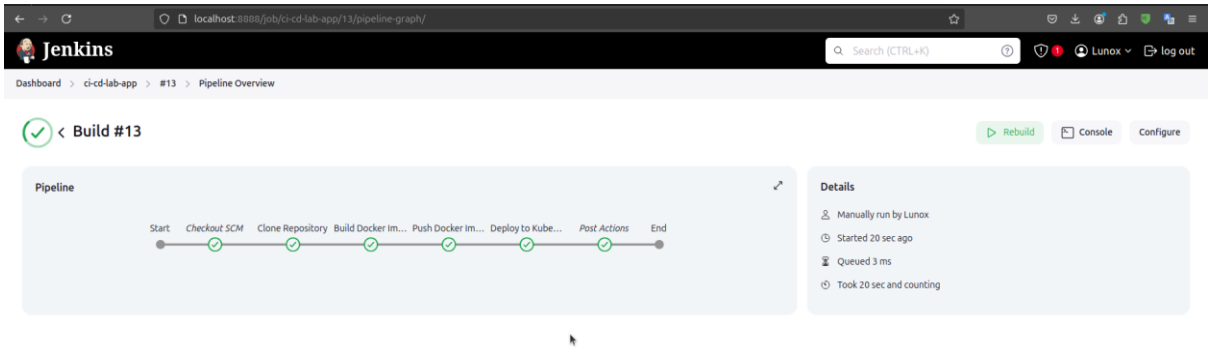
```
lunox@lunox-VirtualBox:~$ kubectl create secret docker-registry my-registry-key \
  --docker-server=https://index.docker.io/v1/ \
  --docker-username=nguyenle2k3 \
  --docker-password=dckr_pat_                    M7Ag \
  --docker-email=nguyenle                com
secret/my-registry-key created
```

Thêm vào file deployment.yaml
imagePullSecrets: - name: my-registry-key

k8s > ! deployment.yaml

```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
                  ports:
                      - containerPort: 8080
            imagePullSecrets:
                - name: my-registry-key
---
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
```

Rebuild and open web:



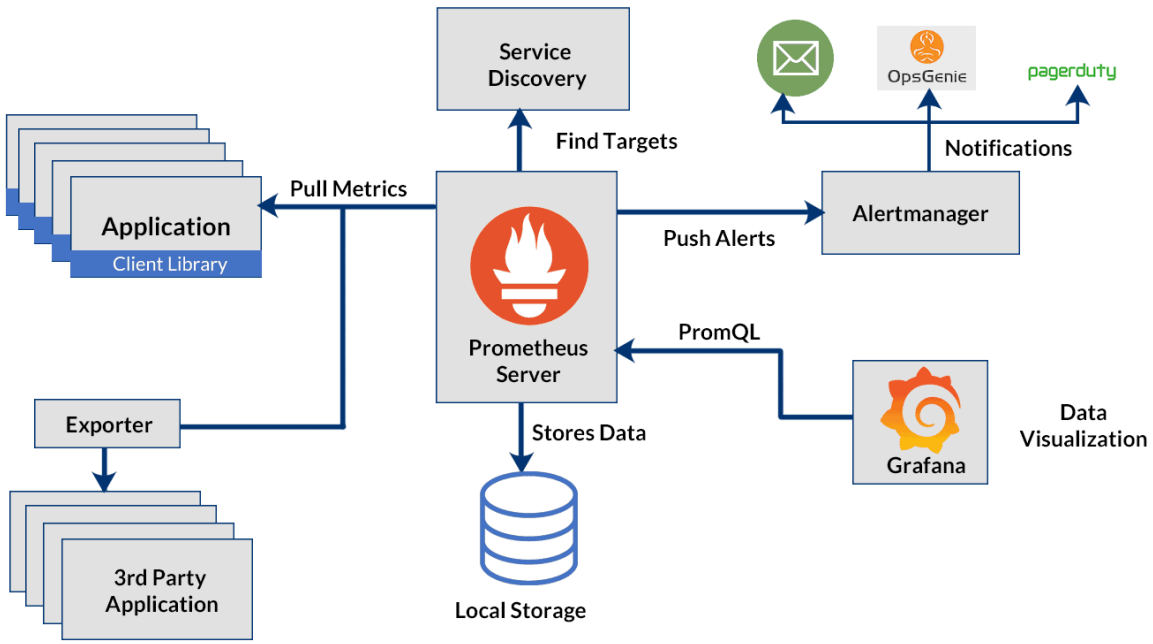

```
lunox@lunox-VirtualBox:~$ kubectl get svc
NAME                TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
ci-cd-lab-service   NodePort    10.101.119.189   <none>        80:32680/TCP   25m
kubernetes          ClusterIP   10.96.0.1        <none>        443/TCP        40m
lunox@lunox-VirtualBox:~$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

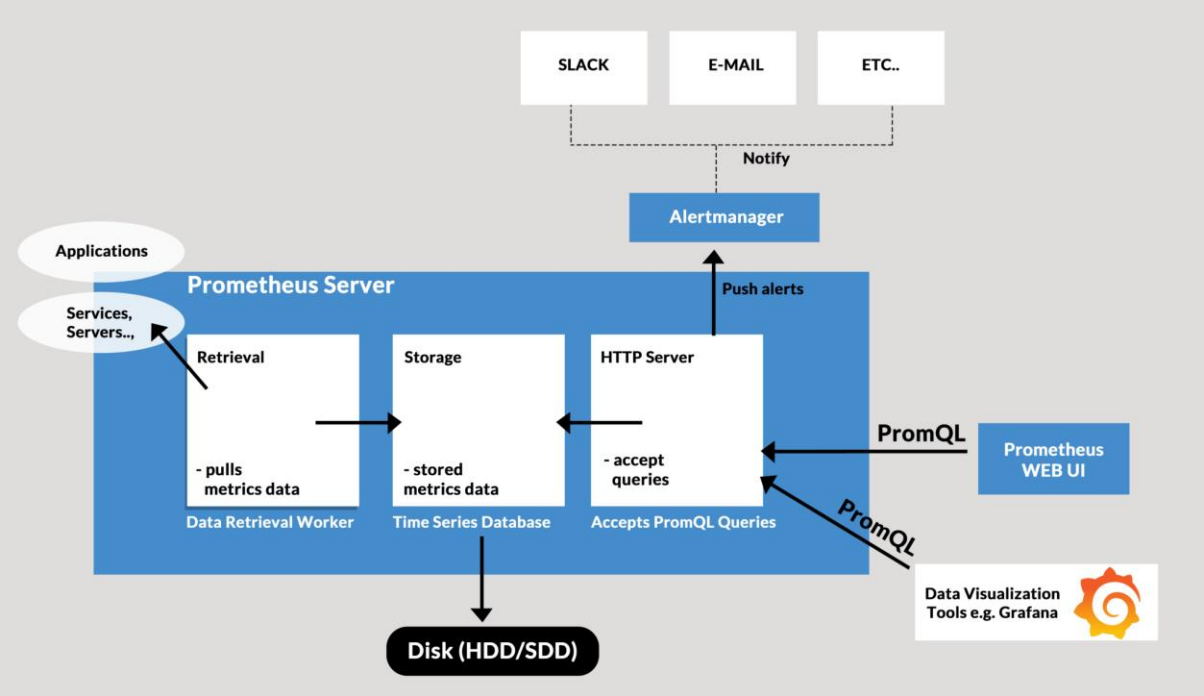
Thành công.

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     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
```

```
(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=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/helm.gpg] https://baltocdn.com/helm/st
able/debian/ all main" | sudo tee /etc/apt/sources.list.d/helm-stable-debian.list
deb [arch=amd64 signed-by=/usr/share/keyrings/helm.gpg] https://baltocdn.com/helm/stable/debian/ all main
lunox@lunox-VirtualBox:~$ 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!⎈
lunox@lunox-VirtualBox:~$
```

Cài đặt Prometheus Kubernetes:

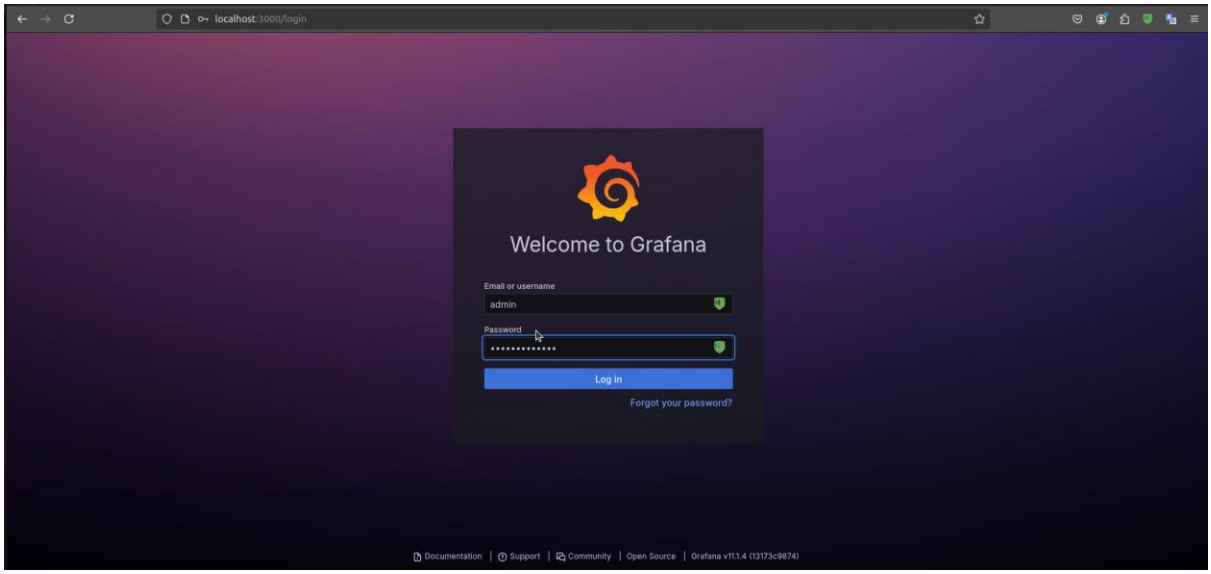
helm install prometheus prometheus-community/kube-prometheus-stack

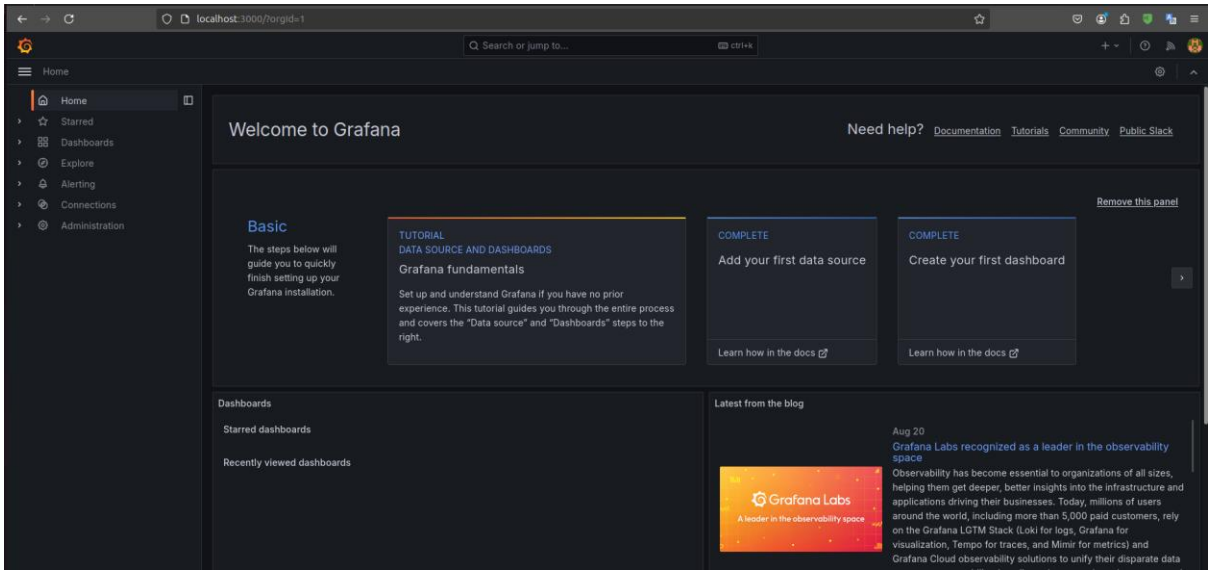
```
lunox@lunox-VirtualBox:~$ helm install prometheus prometheus-community/kube-prometheus-stack 2>/dev/null
NAME: prometheus
LAST DEPLOYED: Wed Aug 21 09:53:44 2024
NAMESPACE: default
STATUS: deployed
REVISION: 1
NOTES:
kube-prometheus-stack has been installed. Check its status by running:
  kubectl --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 --namespace default get pods -l "release=prometheus"
NAME                                                READY   STATUS    RESTARTS   AGE
prometheus-kube-prometheus-operator-5d8bbcc8f8-6sjnd   1/1     Running   0          27s
prometheus-kube-state-metrics-688d66b5b8-95z48         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