**Cài đặt Control Plane và Worker node:**

Dưới đây là hướng dẫn **đầy đủ và được cập nhật 2024**, có thể áp dụng cho Ubuntu 20.04, 22.04 (Jammy) và mới hơn:

**1. Cài đặt công cụ hỗ trợ HTTPS cho APT**

sudo apt-get update

sudo apt-get install -y apt-transport-https ca-certificates curl gpg git vim

**2. Thêm khóa GPG và tạo source list đúng cách**

sudo mkdir -p /etc/apt/keyrings

curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.29/deb/Release.key | gpg --dearmor | sudo tee /etc/apt/keyrings/kubernetes-apt-keyring.gpg > /dev/null

echo "deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.29/deb/ /" | sudo tee /etc/apt/sources.list.d/kubernetes.list > /dev/null

**3. Cập nhật và cài đặt Kubernetes:**

sudo apt-get update

sudo apt-get install -y kubelet kubeadm kubectl

sudo apt-mark hold kubelet kubeadm kubectl

**Kiểm tra sau khi cài đặt**

kubectl version --client

kubeadm version

kubelet –version

***Tiếp theo là triển khai Kubernetes cluster multi-node (1 control plane + 1 worker)***

***Bước 1: Cấu hình hệ thống (trên cả 2 VM Ubuntu)***

***Tắt swap****:*

**Bật các kernel module cần thiết:**

**Cấu hình sysctl:**

sudo swapoff -a

sudo sed -i '/ swap / s/^/#/' /etc/fstab

cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf

br\_netfilter

EOF

sudo modprobe br\_netfilter

cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf

net.bridge.bridge-nf-call-iptables = 1

net.ipv4.ip\_forward = 1

net.bridge.bridge-nf-call-ip6tables = 1

EOF

sudo sysctl --system

***Bước 2: Cài container runtime (nếu chưa có)***

***Kubernetes cần một container runtime. Dùng containerd là ổn định và phổ biến nhất:***

sudo apt-get install -y containerd

sudo mkdir -p /etc/containerd

containerd config default | sudo tee /etc/containerd/config.toml

***Chỉnh cấu hình để SystemdCgroup = true:***

sudo vim /etc/containerd/config.toml

***Tìm dòng:***

[plugins."io.containerd.grpc.v1.cri".containerd.runtimes.runc.options]

***Và chỉnh:***

SystemdCgroup = true

***Sau đó:***

sudo systemctl restart containerd

sudo systemctl enable containerd

***Bước 3: Tạo cluster trên control plane node***

***Trên VM control plane:***

sudo kubeadm init --pod-network-cidr=10.244.0.0/16

***Dùng lệnh:*** kubeadm join ... dành cho worker. hoặc có thể lấy lại token này bằng lệnh:

sudo kubeadm token create --print-join-command

***Sau đó cấu hình kubectl:***

mkdir -p $HOME/.kube

sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

sudo chown $(id -u):$(id -g) $HOME/.kube/config

***Bước 4: Cài Flannel CNI (hoặc Calico, Cilium, etc.)***

kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml

***Bước 5: Thêm worker node vào cluster***

Trên node worker, dùng lệnh kubeadm join mà bạn đã nhận từ bước 3. Ví dụ:

sudo kubeadm join 192.168.1.100:6443 --token abcdef.0123456789abcdef \

--discovery-token-ca-cert-hash sha256:xxxxxxxxxxxxxxxxxxxx

***Kiểm tra cluster (trên control plane)***

kubectl get nodes

***Mặc định, kubernetes không cho phép chạy pod trên nodo control plane, để cho phép pod chạy trên node control-plane (dành cho môi trường học tập, lab, máy ảo test)***

***Chạy lệnh sau để xóa taint:***

kubectl taint nodes <node-name> node-role.kubernetes.io/control-plane-

***Ví dụ, nếu node bạn tên là control-plane, thì:***

kubectl taint nodes control-plane node-role.kubernetes.io/control-plane-

Dấu - ở cuối lệnh là để xóa taint.

**Cài đặt Ingress Control (NodePort) cho Kubernetes**

kubectl apply -f <https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v1.10.0/deploy/static/provider/baremetal/deploy.yaml>

***Kiểm tra Ingress Control***

kubectl get svc -n ingress-nginx

Hoặc

kubectl get serice –all-namespaces

***Xác định ingressClass để cấu hình cho ingress service***

kubectl get ingressclass

***Kiểm tra cấu hình của một Ingress***

kubectl get ingress ingress-app1 -o yaml

kubectl get ingress rancher -n cattle-system -o yaml

***Kiểm tra logs Ingress***

kubectl logs -n ingress-nginx -l app.kubernetes.io/name=ingress-nginx

***Dùng forward để kiểm tra một ứng dụng trong pod với Cluster-IP:***

kubectl port-forward svc/cluster-ip-app1 18080:8080

Sau khi forword có thể kiểm tra ứng dụng thông qua localhost:18080

**Cài đặt, config và kiểm tra cert-manager trong k8s**

**Bước 1: Cài cert-manager**

kubectl apply -f <https://github.com/cert-manager/cert-manager/releases/download/v1.18.1/cert-manager.yaml>

Xem pod certificate

kubectl get pods -n cert-manager

***Giảm ndots cho pod cert-manager trước khi thực hiện bước tiếp theo (nhằm tránh việc phân giải dns sai do ndots cao):***

**Trong Kubernetes**, khi pod chạy, nó sẽ inherit ndots từ /etc/resolv.conf của node hoặc từ config CoreDNS → nhưng ta có thể **override cho từng pod** qua: **dnsConfig.options** trong Deployment

**Cụ thể — ví dụ chỉnh Deployment của cert-manager:**

apiVersion: apps/v1

kind: Deployment

metadata:

name: cert-manager

namespace: cert-manager

spec:

template:

spec:

dnsPolicy: ClusterFirst

dnsConfig:

options:

- name: ndots

value: "1"

**Với cert-manager — bạn chỉ cần làm:**

kubectl edit deployment -n cert-manager cert-manager

→ Thêm đoạn:

spec:

template:

spec:

dnsPolicy: ClusterFirst

dnsConfig:

options:

- name: ndots

value: "1"

→ Save lại → Kubernetes sẽ tự rollout lại pod cert-manager mới.

**Hoặc:**

**Patch Deployment/cert-manager để set dnsConfig.options.ndots: 1**

**Lệnh cụ thể:**

kubectl patch deployment cert-manager -n cert-manager --type='json' -p='[{"op": "add", "path": "/spec/template/spec/dnsConfig", "value": {"options": [{"name": "ndots", "value": "1"}]}}]'

**Sau đó restart lại cert-manager:**

kubectl rollout restart deployment cert-manager -n cert-manager

**Bước 2: Tạo ClusterIssuer**

*tạo file cluster-issuer.yaml:*

apiVersion: cert-manager.io/v1

kind: ClusterIssuer

metadata:

name: letsencrypt-http01

spec:

acme:

email: vovantungdt123@gmail.com

server: https://acme-v02.api.letsencrypt.org/directory

privateKeySecretRef:

name: letsencrypt-http01-key

solvers:

- http01:

ingress:

class: nginx

kubectl apply -f cluster-issuer.yaml

**Kiểm tra Issuer / ClusterIssuer**

kubectl get clusterissuer

Phải thấy:

NAME READY AGE

letsencrypt-http01 True 10m

**Bước 3: Tạo ingress với cert-manager annotation**

apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

name: nexus-ingress

namespace: nexus

annotations:

cert-manager.io/cluster-issuer: letsencrypt-http01

nginx.ingress.kubernetes.io/force-ssl-redirect: "true"

spec:

ingressClassName: nginx

tls:

- hosts:

- nexus.txuapp.com

secretName: nexus-tls

rules:

- host: nexus.txuapp.com

http:

paths:

- path: /

pathType: Prefix

backend:

service:

name: nexus

port:

number: 8081

**Bước 5: Kết quả mong muốn**

Khi bạn truy cập https://nexus.txuapp.com:

* Request sẽ qua modem NAT → Ubuntu → ingress controller → cert-manager sẽ xin cert từ Let’s Encrypt và lưu trong Secret nexus-tls.
* Ingress controller tự dùng cert đó → **HTTPS chuẩn production**

**Kiểm tra Secret nexus-tls**

kubectl get secret nexus-tls -n nexus -o yaml

→ Phải thấy:

type: kubernetes.io/tls

data:

tls.crt: <nội dung base64>

tls.key: <nội dung base64>

Nếu không có hoặc trống → là do cert-manager chưa xin được cert.

Xem chi tiết certificate đã cấp:

kubectl describe certificate -n nexus nexus-tls

**Kiểm tra Ingress**

kubectl describe ingress nexus-ingress -n nexus

**Xem logs cert-manager**

kubectl logs -n cert-manager deploy/cert-manager -f

**Cách vào pod cert-manager kiểm tra:**

kubectl run curl-test --rm -i --tty --image=alpine --restart=Never – sh

**Bên trong shell của pod, cài curl:**

apk add --no-cache curl

**Sau đó test truy cập ACME server:**

curl -v <https://acme-v02.api.letsencrypt.org/directory>

Xem DNS config trong pod:

cat /etc/resolv.conf

Kiểm tra log CoreDNS:

kubectl logs -n kube-system -l k8s-app=kube-dns

Kiểm tra trang thái certificate:

kubectl get certificate -A -o wide

kubectl describe certificate -A

kubectl get certificate -A

Kiểm tra cert-manager log

kubectl logs -n cert-manager -l app.kubernetes.io/name=cert-manager

**Xóa Certificate để cert-manager tạo lại từ đầu** — việc này an toàn:

kubectl delete certificate nexus-tls -n nexus

Rồi apply lại ingress:

kubectl apply -f ingress-nexus.yaml

Hoặc nếu bạn thích, xóa luôn cái secret cũ (nếu có):

kubectl delete secret nexus-tls -n nexus

kubectl run -n cert-manager curl-client --rm -it --image=alpine -- sh

cat /etc/resolv.conf

Vì DNS options như ndots được set từ kubelet khi pod được deploy (thường default từ kubelet hoặc từ deployment yaml).

Bạn có thể chạy pod tạm:

kubectl run -n cert-manager dns-test --rm -it --image=busybox:1.28 -- sh

Rồi bên trong:

cat /etc/resolv.conf

→ Giá trị này sẽ giống với pod cert-manager, vì cùng 1 namespace và kubelet config.

**Tạo forward từ windows host đến máy ảo:**

netsh interface portproxy add v4tov4 listenport=80 listenaddress=0.0.0.0 connectport=31338 connectaddress=192.168.98.141

**Xóa một forward:**

netsh interface portproxy delete v4tov4 listenport=443 listenaddress=0.0.0.0

**Xem các forward:**

netsh interface portproxy show all

**Cài Rancher trong k8s**

**Bước 1: Thêm repo Helm Rancher**

helm repo add rancher-latest https://releases.rancher.com/server-charts/latest

helm repo update

**Bước 2: Tạo namespace cho Rancher**

kubectl create namespace rancher

**Bước 3: Cài đặt Rancher**

helm install rancher rancher-latest/rancher \

--namespace rancher \

--create-namespace \

--set hostname=rancher.txuapp.com \

--set replicas=1 \

--set bootstrapPassword="Phan@123" \

--set ingress.tls.source=secret \

--set ingress.ingressClassName=nginx \

--set extraEnv[0].name=CATTLE\_BOOTSTRAP\_PASSWORD \

--set extraEnv[0].value="Phan@123" \

--set ingress.extraAnnotations."cert-manager\.io/cluster-issuer"="letsencrypt-http01" \

--wait \

--timeout 10m

Note:

- letsencrypt-http01: tên clusterIssuer của cert-manager

- nginx: tên class của ingress controller

- Với tham số cài đặt trên, một ingress với hostname “rancher.txuapp.com” sẽ được tạo ra và trỏ đến pod ứng dụng rancher. Ingress sẽ xin và dùng cert từ cert-manager được chỉ định.

- Lấy password mặc định (trong trường này là Phan@123 vì password được set khi cài đặt):

kubectl get secret -n rancher bootstrap-secret -o go-template='{{.data.bootstrapPassword | base64decode}}'

**Cài MinIO trong ubuntu (trên docker)**

**Bước 1: Cài đặt Docker trên Ubuntu**

***1.1. Cập nhật hệ thống:***

sudo apt update

sudo apt upgrade -y

***1.2. Cài Docker từ Docker repository chính thức:***

sudo apt install -y \

ca-certificates \

curl \

gnupg \

lsb-release

***1.3. Thêm Docker GPG key:***

sudo mkdir -p /etc/apt/keyrings

curl -fsSL https://download.docker.com/linux/ubuntu/gpg \

| sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg

***1.4. Thêm Docker repository:***

echo \

"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] \

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

$(lsb\_release -cs) stable" \

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

***1.5. Cài Docker Engine:***

sudo apt update

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

***1.6. Thêm user vào nhóm docker (để không cần sudo):***

sudo usermod -aG docker $USER

newgrp docker # chạy lại phiên làm việc hoặc logout/login

***Kiểm tra Docker hoạt động:***

docker version

docker run hello-world

**Bước 2: Tạo thư mục dữ liệu MinIO**

sudo mkdir -p /mnt/data/minio

sudo useradd -r $USER -s /sbin/nologin

sudo chown -R $USER:$USER /mnt/data/minio

sudo chmod -R 750 /mnt/data/minio

**Bước 3: Tạo service chạy minio**

sudo vim /etc/systemd/system/minio.service

***Nội dung:***

[Unit]

Description=MinIO Subnet (Full features)

After=network.target docker.service

Requires=docker.service

[Service]

User=nhuy

Group=nhuy

ExecStartPre=/usr/bin/docker pull quay.io/minio/minio:RELEASE.2024-06-11T03-13-30Z

ExecStart=/usr/bin/docker run --rm \

--name minio \

-p 9000:9000 \

-p 9001:9001 \

-e MINIO\_ROOT\_USER=admin \

-e MINIO\_ROOT\_PASSWORD=Phan@123 \

-e MINIO\_IDENTITY\_MANAGE=on \

-e MINIO\_PROMETHEUS\_AUTH\_TYPE=public \

-e MINIO\_BROWSER\_REDIRECT\_URL=https://console-minio.txuapp.com \

-v /mnt/data/minio:/data \

quay.io/minio/minio:RELEASE.2024-06-11T03-13-30Z \

server /data --console-address ":9001"

Restart=always

RestartSec=5

LimitNOFILE=65536

[Install]

WantedBy=multi-user.target

***Chạy service:***

sudo systemctl daemon-reexec

sudo systemctl daemon-reload

sudo systemctl enable --now minio

***Khởi động lại service (khi cần)***

sudo systemctl start docker

sudo systemctl restart minio

***Kiểm tra trạng thái của service:***

sudo systemctl status minio

Tạo butket “jenkins-data” cho jenkins:

Tạo access key cho phép đăng nhập và sử dụng butket “jenkins-data”:

{

"Version": "2012-10-17",

"Statement": [

{

"Action": [

"s3:GetBucketLocation",

"s3:ListBucket"

],

"Effect": "Allow",

"Resource": [

"arn:aws:s3:::jenkins-data"

]

},

{

"Action": [

"s3:PutObject",

"s3:GetObject",

"s3:DeleteObject"

],

"Effect": "Allow",

"Resource": [

"arn:aws:s3:::jenkins-data/\*"

]

}

]

}

Hoặc đơn giản:

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Action": ["s3:\*"],

"Resource": [

"arn:aws:s3:::jenkins-data",

"arn:aws:s3:::jenkins-data/\*"

]

}

]

}

Tạo secret chứa access key đăng nhập minio để gán cho pod sử dụng:

kubectl create secret generic minio-creds --from-literal=MINIO\_ACCESS\_KEY\_ID=xxxx --from-literal=MINIO\_SECRET\_ACCESS\_KEY=yyyy -n jenkins

**Bước 1: Cài đặt NFS Server trên máy Ubuntu (máy chia sẻ volume)**

**Trên máy Ubuntu có IP ví dụ là 192.168.98.150:**

sudo apt update

sudo apt install -y nfs-kernel-server

**Tạo thư mục dùng để chia sẻ**

sudo mkdir -p /mnt/data/nfs/jenkins

sudo chown -R nobody:nogroup /mnt/data/nfs/jenkins

sudo chmod 777 /mnt/data/nfs/jenkins

**Cấu hình chia sẻ trong /etc/exports**

sudo vim /etc/exports

Thêm dòng sau:

/mnt/data/nfs/jenkins 192.168.98.0/24(rw,sync,no\_subtree\_check,no\_root\_squash)

Thay 192.168.98.0/24 bằng subnet mạng Kubernetes hoặc IP của node nếu cần chính xác hơn.

**Áp dụng cấu hình:**

sudo exportfs -ra

sudo systemctl restart nfs-kernel-server

**Bước 2: Cài đặt NFS Client trên tất cả node của Kubernetes cluster**

**Trên mỗi node (cả control-plane và worker):**

sudo apt update

sudo apt install -y nfs-common

**Bước 3: Tạo PersistentVolume (PV) và PersistentVolumeClaim (PVC) trong Kubernetes**

***nfs-pv.yaml:***

apiVersion: v1

kind: PersistentVolume

metadata:

name: jenkins-nfs-pv

spec:

capacity:

storage: 10Gi

accessModes:

- ReadWriteMany

nfs:

server: 192.168.98.150

path: /mnt/data/nfs/jenkins

persistentVolumeReclaimPolicy: Retain

***nfs-pvc.yaml:***

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

name: jenkins-nfs-pvc

namespace: jenkins

spec:

accessModes:

- ReadWriteMany

resources:

requests:

storage: 10Gi

kubectl apply -f nfs-pv.yaml

kubectl apply -f nfs-pvc.yaml

**Bước 4: Gắn PVC vào Pod Jenkins**

Trong phần volumeMounts và volumes của jenkins và các sidecar như backup-to-minio, thay hostPath bằng:

volumeMounts:

- name: jenkins-home

mountPath: /var/jenkins\_home

volumes:

- name: jenkins-home

persistentVolumeClaim:

claimName: jenkins-nfs-pvc

**Kiểm tra**

Sau khi khởi động lại pod Jenkins:

kubectl get pods -n jenkins -o wide

Và kiểm tra thử ghi dữ liệu, ví dụ tạo job rồi xem /mnt/data/nfs-jenkins có dữ liệu hay chưa trên máy NFS server.

**Cài đặt Nexus trong k8s:**

**Lấy mật khẩu mặc định của nexus:**

kubectl exec -it nexus-7fd84bfdfb-z9t47 -n nexus -- cat /nexus-data/admin.password

Sau khi nhập mật khẩu mặc định, nexus yêu cầu đặt lại mật khẩu mởi cho tài khoản admin. Tài khoản này được dùng để push thư viện lên nexus, cách cấu hình:

*Thêm code sau vào file settings.xml trong thư mục* ***.m2****:*

<settings xmlns="http://maven.apache.org/SETTINGS/1.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0 https://maven.apache.org/xsd/settings-1.0.0.xsd">

<servers>

<server>

<id>nexus</id>

<username>admin</username>

<password>bungteDT@123</password>

</server>

</servers>

</settings>

***Một số lệnh kiểm tra dns:***

nslookup jenkins.jenkins.svc (trả về ip của domain)

curl -v http:// nslookup jenkins.jenkins.svc.cluster.local.:8080 (nếu không dùng full url thì cần chú ý “ndots”, “ndots” lớn hơn số dấu “.” Trong domain thì dns sẽ tự gắn thêm phần searches (ví dụ: cluster.local), gây sai domain và dns phân giải không chính xác). Để xem lại “ndots” dùng lệnh: cat /etc/resolv.conf

kiểm tra chi tiết yaml của một service, pod, … với lệnh:

kubectl get svc jenkins -n jenkins -o yaml

kubectl get svc jenkins -n jenkins -o yaml | grep noi\_dung\_tim\_kiem [-A so\_dong\_can\_xem]

kubectl get deployment jenkins -n jenkins -o yaml | grep containerPort -A 2

kubectl get svc jenkins -n jenkins -o yaml | grep selector -A 2

Xem jenkins đăng lăng nghe ở những port nào:

kubectl exec -it jenkins-bb955c969-xdm7r -n jenkins -- ss -tuln

Chạy một pod test:

kubectl run txu-test --rm -i -t --image=ubuntu:22.04 -n jenkins -- /bin/bash

Get Endpoits:

kubectl get endpoints -n jenkins jenkins -o yaml

Get Config map

kubectl -n kube-system get configmap coredns -o yaml

Kiểm tra cấu hình dns:

cat /var/lib/kubelet/config.yaml

cat /var/lib/kubelet/config.yaml | grep clusterDNS

Restart kubelet:

sudo systemctl restart kubelet

**Cài đặt và kiểm tra docker:**

***Kiểm tra người dùng trong nhóm “docker”***

getent group docker

***Nếu chưa có nhóm “docker” có thể thêm nhóm:***

sudo groupadd docker

***Thêm người dùng võ vào nhóm “docker”***

sudo usermod -aG docker vo

***Hoặc thêm người dùng hiện tại vào nhóm “docker”***

sudo usermod -aG docker $USER

***Sau khi thêm user vào nhóm “docker” có thể chạy lệnh sau và kiểm tra lại nhóm “docker”***

newgrp docker

**1. Kiểm tra CoreDNS có hoạt động đúng không**

kubectl get pods -n kube-system -l k8s-app=kube-dns -o wide

Bạn nên thấy 2 pod coredns-\* đang Running.

**2. Kiểm tra logs của CoreDNS**

kubectl logs -n kube-system -l k8s-app=kube-dns

Tìm xem có dòng lỗi nào liên quan jenkins.jenkins.svc.cluster.local hoặc các truy vấn bị từ chối.

**3. Kiểm tra lại resolv.conf của Pod curl-test**

kubectl exec -n jenkins curl-test -- cat /etc/resolv.conf

Output chuẩn nên là:

search jenkins.svc.cluster.local svc.cluster.local cluster.local

nameserver 10.96.0.10

options ndots:5

**Sau đó kiểm tra lại DNS:**

nslookup jenkins.jenkins.svc.cluster.local

curl -v <http://jenkins.jenkins.svc.cluster.local:8080>

**Tạo secret gắn cho pod kaniko sử dụng cho đăng nhập aws và push code lên ecr:**

kubectl create secret generic aws-creds-kaniko --from-literal=AWS\_ACCESS\_KEY\_ID=aws-access-key-id --from-literal=AWS\_SECRET\_ACCESS\_KEY=aws-secret-key --from-literal=AWS\_REGION=ap-southeast-1 -n jenkins

kubectl create secret generic minio-creds --from-literal=MINIO\_ACCESS\_KEY\_ID=xxxx --from-literal=MINIO\_SECRET\_ACCESS\_KEY=yyyy -n jenkins

**Tạo secret cho phép jenkins agent (kubectl) chạy deployment kéo image từ ecr về tạo container trong kubernetes:**

kubectl create secret docker-registry ecr-secret --docker-server=211125364313.dkr.ecr.ap-southeast-1.amazonaws.com --docker-username=AWS --docker-password=$(aws ecr get-login-password --region ap-southeast-1) [--docker-email=vovantungdt123@gmail.com](mailto:--docker-email=vovantungdt123@gmail.com) -n jenkins -n backend

kubectl create secret generic aws-credentials --from-file=credentials=$HOME/.aws/credentials -n jenkins

*(secret đọc file, cung cấp cho tạo aws/credentials trong pod jenkins trên kubernetes. Trong lệnh dùng cần dùng “$HOME” thay vì “~” bới vì “~” là shell nên không được hiểu trong câu lệnh kubectl, còn “$HOME” là biến môi trường nên sẽ luôn đúng cho các trường hợp)*

(Lưu ý: trước khi tạo, chạy aws config với secret được lấy từ aws với quyền phù hợp hoặc secret từ root account)

**Cài đặt Grafana, prometheus, alertmanager**

**Bước 1: Cài đặt Helm**

Nếu bạn chưa cài Helm trên máy control plane:

curl https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3 | bash

helm version

**Cài đặt StorageClass cho nfs server với tên nfs-client (dùng StorageClass này để mount dữ liệu của pod đến nfs server):**

helm repo add nfs-subdir-external-provisioner https://kubernetes-sigs.github.io/nfs-subdir-external-provisioner/

helm repo update

**Xác định thư mục mount trên NFS server**

Trên NFS server (192.168.98.150), bạn cần đảm bảo:

**Tạo thư mục dùng để chia sẻ**

sudo mkdir -p /mnt/data/nfs/nfsclinet

sudo chown -R nobody:nogroup /mnt/data/nfs/ nfsclinet

sudo chmod 777 /mnt/data/nfs/ nfsclinet

**Cấu hình chia sẻ trong /etc/exports**

sudo vim /etc/exports

Thêm dòng sau:

/mnt/data/nfs/nfsclinet 192.168.98.0/24(rw,sync,no\_subtree\_check,no\_root\_squash)

**Áp dụng cấu hình:**

sudo exportfs -ra

sudo systemctl restart nfs-kernel-server

**Kiểm tra lại trong nfs-values.yaml**

Trong file bạn dùng để deploy Helm chart:

nfs:

server: 192.168.98.150

path: /mnt/data/nfs/nfsclient

storageClass:

name: nfs-client

defaultClass: false

accessModes: [ReadWriteOnce]

reclaimPolicy: Retain

path: phải trùng **chính xác** với thư mục bạn đã cấp quyền.

**Gỡ và deploy lại NFS provisioner**

helm uninstall nfs-client -n nfs-storage

kubectl delete namespace nfs-storage

**Cài đặt StorageClass nfs-client:**

helm repo update

helm upgrade --install nfs-client \

nfs-subdir-external-provisioner/nfs-subdir-external-provisioner \

--namespace nfs-storage --create-namespace \

-f nfs-values.yaml

**Kiểm tra StorageClass**

kubectl get sc

**Có thể sử dụng Local Path Provisioner (không lưu tập trung và chủ động như nfs-client)**

kubectl apply -f https://raw.githubusercontent.com/rancher/local-path-provisioner/master/deploy/local-path-storage.yaml

storageClassName là local-path:

**Tạo namespace cho Prometheus**

kubectl create namespace monitoring

**Xóa pvc cũ để upgrade prometheus-stack (optional)**

kubectl delete pvc -n monitoring alertmanager-prometheus-stack-kube-prom-alertmanager-db-alertmanager-prometheus-stack-kube-prom-alertmanager-0

kubectl delete pvc -n monitoring prometheus-prometheus-stack-kube-prom-prometheus-db-prometheus-prometheus-stack-kube-prom-prometheus-0

**Bước 2: Thêm repo Prometheus cộng đồng**

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

helm repo update

**Bước 4: Cài Prometheus + các thành phần đi kèm**

Ta dùng chart kube-prometheus-stack — chart phổ biến nhất hiện nay, gồm:

Prometheus

Grafana  
Alertmanager  
Node Exporter  
Kube State Metrics

**Lệnh cài (với giá trị mặc định):**

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

**Hoặc cài với values file:**

helm install prometheus-stack prometheus-community/kube-prometheus-stack -n monitoring -f monitoring-values.yaml

**Hoặc upgrade nếu đã cài trước đó:**

helm upgrade --install prometheus-stack prometheus-community/kube-prometheus-stack -n monitoring -f monitoring-values.yaml

**Bước 5: Kiểm tra**

kubectl get pods -n monitoring

kubectl describe pod alertmanager-prometheus-stack-kube-prom-alertmanager-0 -n monitoring

**Bước 6: Mở Grafana để xem Dashboard**

**User/Password mặc định:**

Username: admin

Password: prom-operator

(Nếu khác thì bạn xem password từ secret)

kubectl get secret prometheus-stack-grafana -n monitoring -o jsonpath="{.data.admin-password}" | base64 -d

Hoặc: username và password có thể được đặt khi cài trong file (monitoring-values.yaml)

Khi cài đặt prometheus, grafana, alertmanager với StorageClass nfs-client, k8s sẽ tự động tạo pvc mount thư được gắn StorageClass nfs-client, và được mount vào thư mục của nfs-client đã được chỉ định khi cài đặt trên nfs-client

Linh tinh

Xem nhanh một pod đang chạy ở node nào

kubectl get pod -l app=jenkins -o wide -n jenkins

Vào lại container restore-from-minio:

kubectl exec -it <jenkins-pod-name> -n jenkins -c backup-to-minio – sh

Kiểm tra alias:

export MC\_CONFIG\_DIR=/mc-config

mc alias list

Kiểm tra butket trên minio theo đường dẫn đã set trong alias (trong container restore-from-minio, trong deployment của jenkins):

mc ls minio

Nếu vẫn báo Bucket does not exist, bạn có thể tạo bucket một lần bằng tay:

mc mb minio/jenkins-data

# Thử tạo file nhỏ

echo test > /tmp/test.txt

# Thử upload thủ công

mc cp /tmp/test.txt minio/jenkins-data/test.txt

**Thử nghiệm nhanh**

Bạn cũng có thể test bằng cách:

curl -v https://minio.txuapp.com/minio/health/ready

→ Đây là một API mặc định của MinIO → nếu trả về 200 OK là đã trỏ đúng API