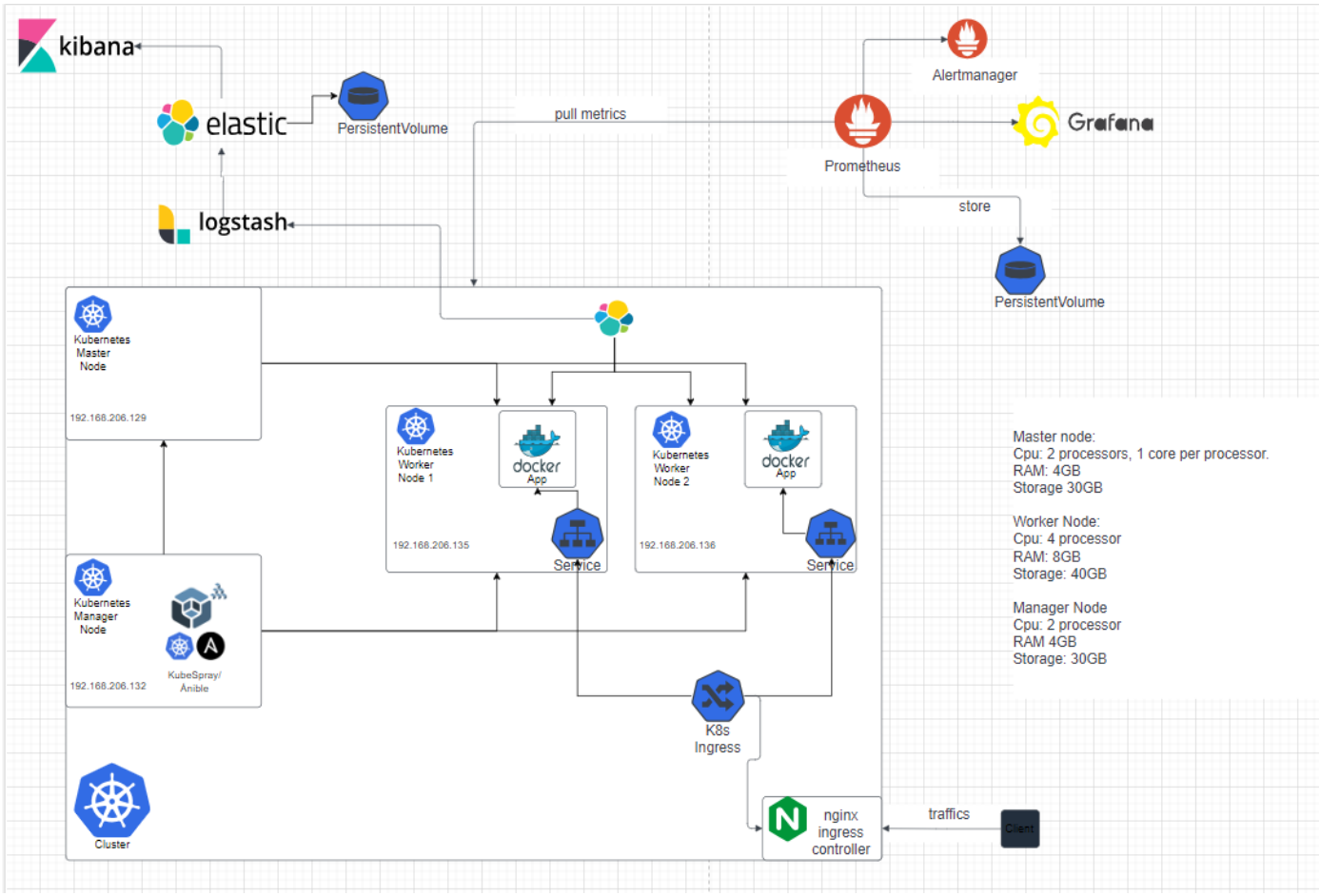
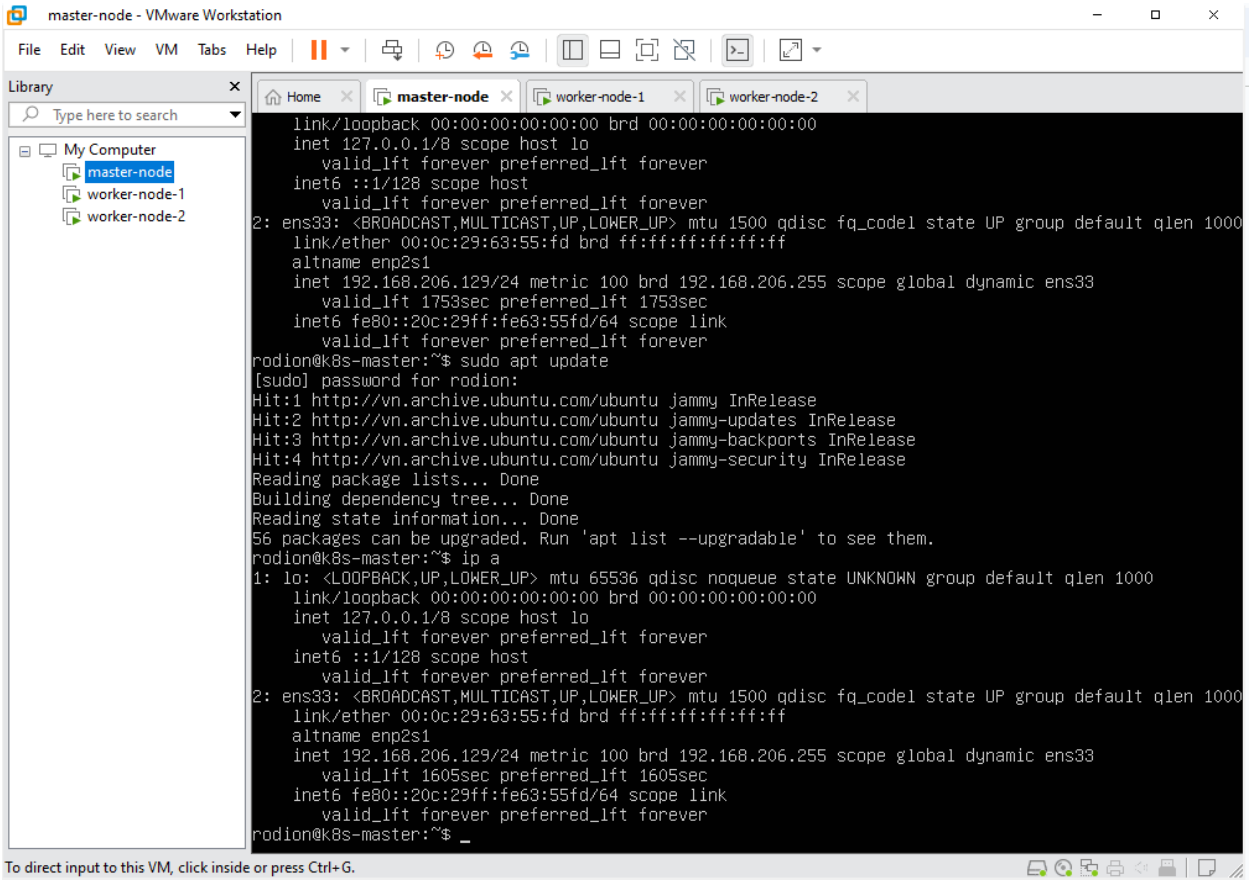
DOC

Mô hình LAB:



Khởi tạo 3 máy chủ chạy Ubuntu Server 22.04.2 trên VMware Workstation Pro



<u>Cấu hình lần lượt của 3 máy chủ:</u>

Master-Node:

HĐH: Ubuntu 22.04.2

CPU: 2 processor , 2 core (1 core / 1 processor)

RAM: 4GB

Storage: 30GB

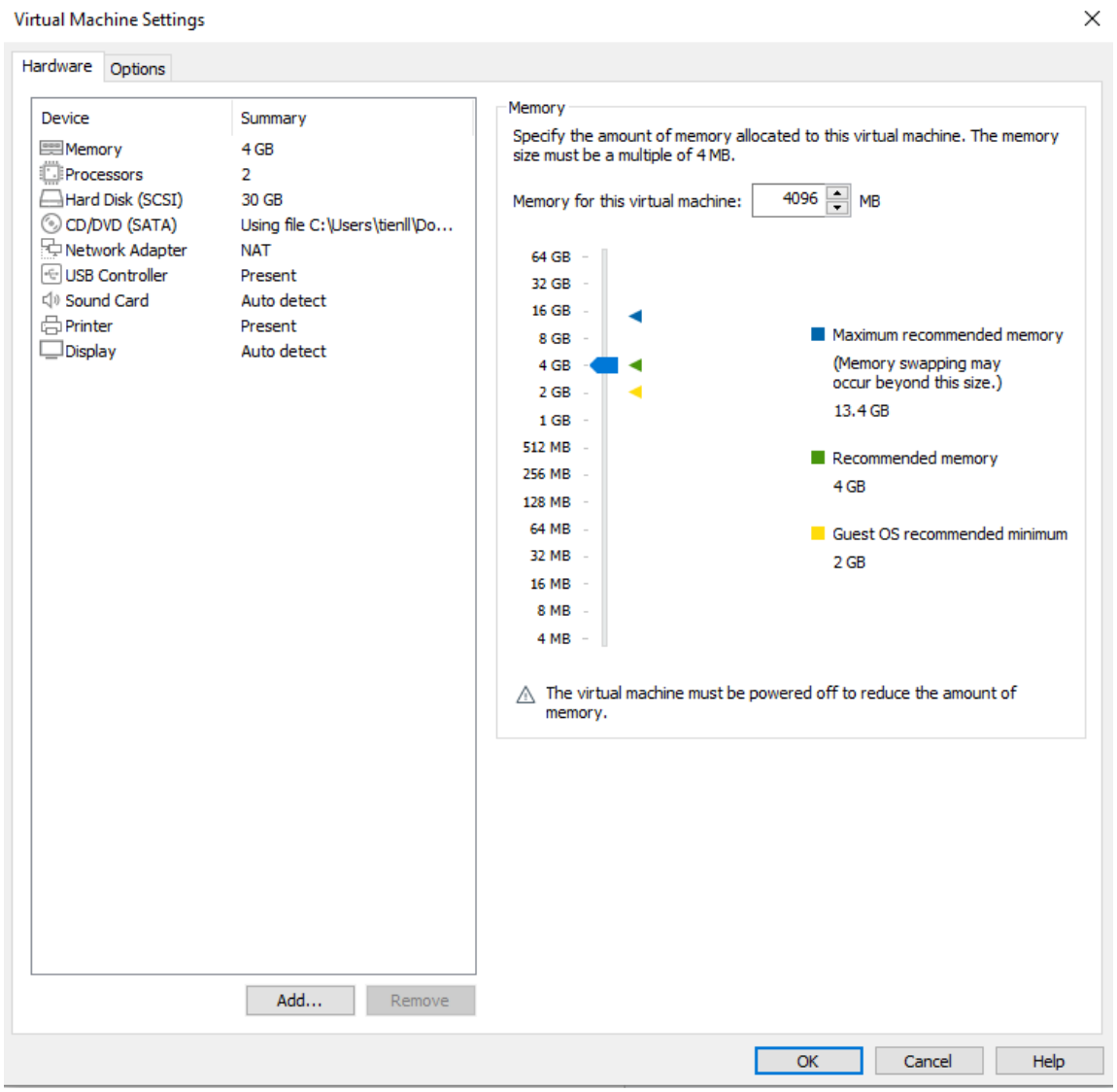
Worker-Nodes:

HĐH: Ubuntu 22.04.2

CPU: 4 processor , 4 core (1 core / 1 processor)

RAM: 8GB

Storage: 40GB



Ip Address:

Master-Node: 192.168.206.129

Worker-Node-1: 192.168.206.135

Worker-Node-2: 192.168.206.136

Sử dụng Kubespray để tạo cụm K8s:

Kubespray (trước đây được gọi là Kargo) là một công cụ mã nguồn mở được sử dụng để triển khai và quản lý các cụm Kubernetes. Nó cho phép tự động hóa quá trình cài đặt và cấu hình Kubernetes trên nhiều máy chủ vật lý hoặc ảo hóa.

Kubespray sử dụng công nghệ Ansible để định cấu hình và triển khai cụm Kubernetes. Ansible là một công cụ tự động hóa cấu hình và quản lý hệ thống, cho phép xác định trạng thái mong muốn của một hệ thống và tự động hóa quá trình đưa hệ thống đó vào trạng thái đó.

Với Kubespray, có thể tạo và quản lý cụm Kubernetes đa máy chủ, bao gồm các thành phần như Control Plane, etcd, worker nodes, networking, và storage. Kubespray hỗ trợ nhiều phiên bản Kubernetes và cung cấp các tùy chọn cấu hình linh hoạt để tùy chỉnh quá trình triển khai theo nhu cầu.

Tạo một máy chủ riêng để chạy Kubespray, máy chủ này sẽ sử dụng Ansible để đẩy cấu hình cụm K8s ra các Node trong cụm.

Cấu hình máy chủ chạy Kubespray:

HĐH: Ubuntu 22.04.2
CPU: 2 processor , 2 core (1 core / 1 processor)
RAM: 4GB
Storage: 30GB
Ip: 192.168.206.132

Cài đặt cấu hình cho Kubespray:

Cài docker:

```
sudo apt update -y

curl -fsSL https://get.docker.com/ | sh

sudo systemctl enable docker.service

sudo systemctl start docker
```

```
sudo usermod -aG docker <user>
```

Kết quả:

```
rodion@kubespray:~$ docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
rodion@kubespray:~$
```

Cấu hình file /etc/hosts để kubespray nhân diện được các host khác trong mạng thông qua hostname:

Sử dụng lệnh sudo vi /etc/hosts và thêm vào cấu hình:

192.168.206.129 k8s-master
192.168.206.135 worker1
192.168.206.136 worker2

```
# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
192.168.206.129 k8s-master
192.168.206.135 worker1
192.168.206.136 worker2
~
~
~
~
~
~
~
```

Tao thư muc cài đăt Kubespray:

```
cd~

mkdir kubernetes_installation

cd kubernetes_installation

git clone https://github.com/kubernetes-sigs/kubespray.git --branch
release-2.19
```

Lưu ý phiên bản cần cài đặt, một số phiên bản cũ không còn tương thích với các phiên bản docker và ubuntu mới. Các phiên bản cũ có

khả năng dính lỗi về các dependencies hoặc các thành phần cài đặt. Phiên bản sử dụng trong bài lab là 2.19, cài đặt trên hệ điều hành ubuntu 22.04.2

Đường dẫn của file kubespray là:
/home/rodion/kubernetes_installation/kubespray

```
rodion@kubespray:~/kubernetes_installation/kubespray$ ls
ansible.cfg          facts.yml          recover-control-plane.yml  scale.yml
ansible_version.yml  index.html         RELEASE.md                 scripts
cluster.yml          inventory          remove-node.yml            SECURITY_CONTACTS
CNAME                legacy_groups.yml  requirements-2.10.txt      setup.cfg
code-of-conduct.md   library            requirements-2.11.txt      setup.py
_config.yml          LICENSE            requirements-2.12.txt      test-infra
contrib              logo               requirements-2.9.txt       tests
CONTRIBUTING.md      Makefile           requirements-2.9.yml       upgrade-cluster.yml
Dockerfile           OWNERS             requirements.txt           Vagrantfile
docs                 OWNERS_ALIASES     reset.yml
extra_playbooks      README.md          roles
```

Điều chỉnh các file cấu hình:

Tạo một inventory mới từ bộ mẫu của kubespray:

```
cd /home/rodion/kubernetes_installation/kubespray

cp -rf inventory/sample inventory/tienll-cluster
```

Tạo file hosts.yaml: file này định nghĩa cài K8s gồm bao nhiêu node, tên và Ip của các node…

```
cd /home/rodion/kubernetes_installation/kubespray/

cd inventory/tienll-cluster

vi hosts.yaml
```

Nội dung file hosts.yaml:

```
[all]
k8s-master  ansible_host=192.168.206.129      ip=192.168.206.129
worker1  ansible_host=192.168.206.135      ip=192.168.206.135
worker2  ansible_host=192.168.206.136      ip=192.168.206.136

[kube-master]
k8s-master

[kube-node]
worker1
worker2
```

```
[etcd]
k8s-master

[k8s-cluster:children]
kube-node
kube-master

[calico-rr]

[vault]
k8s-master
worker1
worker2
```

Cài đặt K8s cluster:

Thực hiện việc cài đặt từ bên trong của một docker container:

```
docker run --rm -it --mount
type=bind,source=/home/rodion/kubernetes_installation/kubespray/inven
tory/tienll-cluster,dst=/inventory
quay.io/kubespray/kubespray:v2.19.0 bash
```

Lệnh này tạo ra một container chứa các file cấu hình được mount từ bên ngoài.

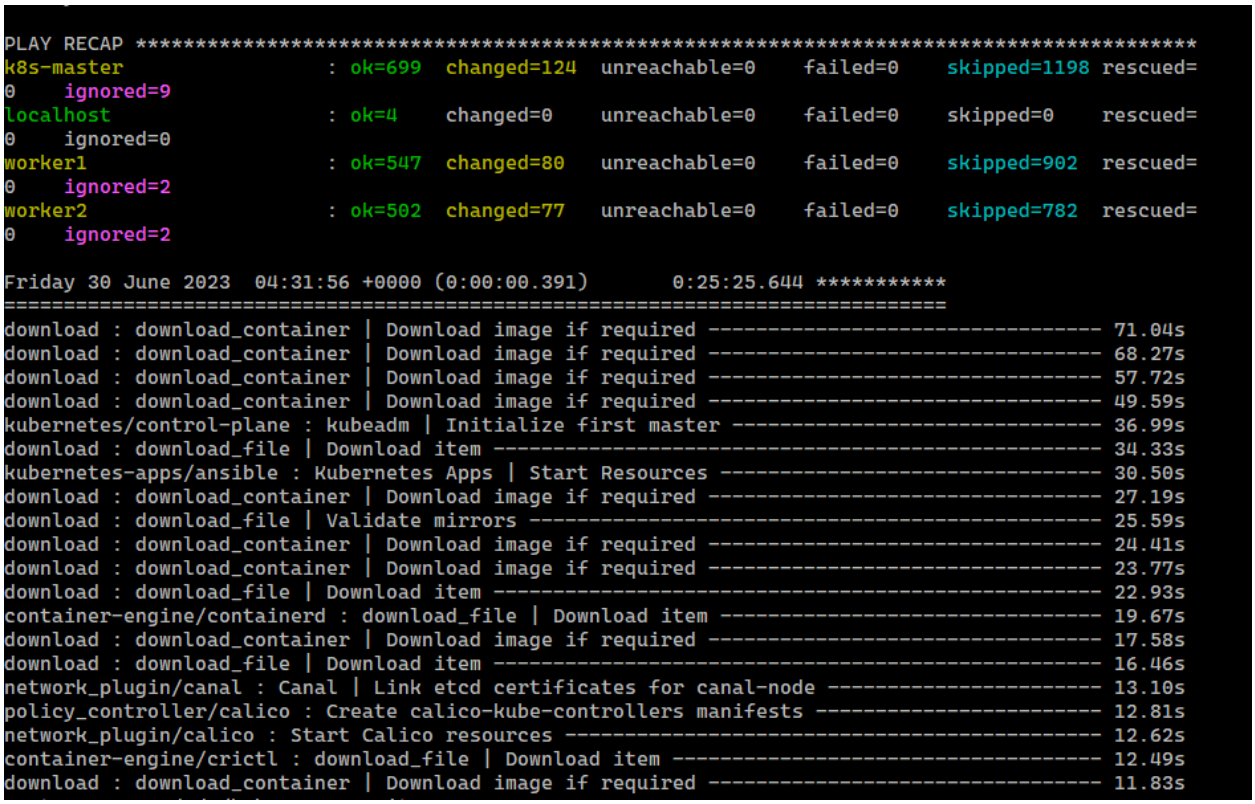
Lệnh để cài đặt:

```
ansible-playbook -i /inventory/hosts.yaml cluster.yml --user=rodion
--ask-pass --become --ask-become-pass
```

Khởi chạy ansible, cung cấp mật khẩu, ansile sẽ download/đẩy những cấu hình cần thiết qua các node trong cụm.

***Note**: Chú ý tên user của các node cần giống nhau. Coi như là một User dùng để install*

Kết quả:



Một số lỗi có thể gặp trong quá trình cài đặt:

Error because python-apt package missing: lỗi này xảy ra khi ansible tìm package python-apt. Tuy nhiên, một số phiên bản của hệ điều hành sử dụng python 3 nên không thể tìm thấy package này.

Cách fix:

Sửa các required_pkgs trong file cấu hình ubuntu.yml (~/kubernetes_installation/kubespray/roles/kubernetes/preinstall/vars/ubuntu.yml) python-apt -> python3-apt

Các phiên bản mới của kubespray đã chuyển thành python3-apt

Error because aufs-tools: Lỗi này cũng tương tự như lỗi python-apt, aufs-tools là một packet trong file ubuntu.yml

Cách fix:

Loại bỏ dòng - aufs-tools trong required_pkgs

Các phiên bản mới của kubespray đã loại bỏ package này.

Error because can't find containerd, docker.io… version: Lỗi xảy ra do sự không tương thích giữa các phiên bản của kubespray và docker

Cách fix:

Sử dụng các phiên bản mới hơn của kubespray

Error because syntax error:

Quick fix:

Comment out những dòng báo lỗi nếu không cần thiết.

Cài đặt và cấu hình kubectl:

Sau khi cài đặt xong thì trên master node đã có sẵn kubectl nên chỉ cần thiết lập:

```
mkdir -p $HOME/.kube

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

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

Kiểm tra trên master node:

```
rodion@k8s-master:~$ kubectl get nodes -o wide
NAME         STATUS   ROLES                  AGE     VERSION   INTERNAL-IP       EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION
k8s-master   Ready    control-plane,master   3h37m   v1.23.7   192.168.206.129   <none>        Ubuntu 22.04.2 LTS   5.15.0-76-gener
worker1      Ready    <none>                 3h36m   v1.23.7   192.168.206.135   <none>        Ubuntu 22.04.2 LTS   5.15.0-76-gener
worker2      Ready    <none>                 3h36m   v1.23.7   192.168.206.136   <none>        Ubuntu 22.04.2 LTS   5.15.0-76-gener
```

Cấu hình server chạy kubespray để join vào cluster (Làm manager)

```
mkdir -p $HOME/.kube

scp k8s-master:~/.kube/config  $HOME/.kube/

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

vi $HOME/.kube/config
```

-> sửa tham số "server: https://127.0.0.1:6443" thành "server: https://192.168.206.129:6443" và lưu lại (Ip master node)

-> có thể sử dụng kubectl từ server chạy kubespray

```
rodion@kubespray:~$ kubectl get nodes -o wide
NAME         STATUS   ROLES                  AGE     VERSION   INTERNAL-IP       EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION      CONTAINER-RUNTIME
k8s-master   Ready    control-plane,master   2d22h   v1.23.7   192.168.206.129   <none>        Ubuntu 22.04.2 LTS   5.15.0-76-generic   containerd://1.6.4
worker1      Ready    <none>                 2d21h   v1.23.7   192.168.206.135   <none>        Ubuntu 22.04.2 LTS   5.15.0-76-generic   containerd://1.6.4
worker2      Ready    <none>                 2d21h   v1.23.7   192.168.206.136   <none>        Ubuntu 22.04.2 LTS   5.15.0-76-generic   containerd://1.6.4
```

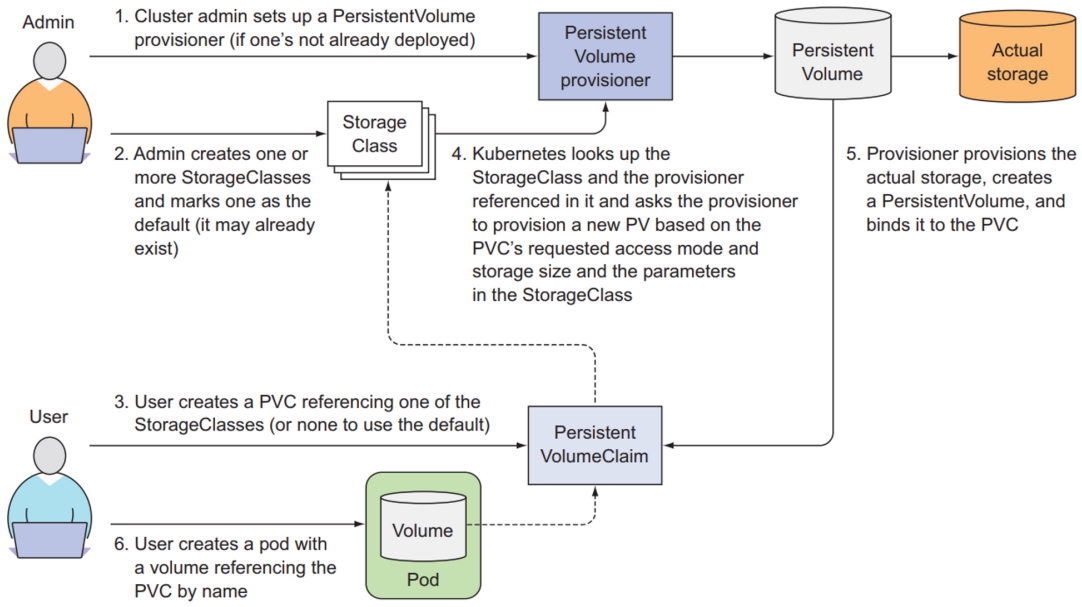
[Link tham khảo]

Thiết lập provisioner, Storageclass

Để dynamic provision các pv, cần phải thiết lập Storageclass. Storageclass sử dụng provisioner để provison các pv theo request từ PVC. Đối với một số môi trường như cloud thì provisoner đã được thiết lập sẵn. Nhưng đối với local thì không -> thiết lập provisioner thủ công.

https://github.com/rancher/local-path-provisioner

Cách hoạt động của PV, PVC, StorageClass.




```
rodion@kubespray:~$ kubectl get storageclass
NAME         PROVISIONER             RECLAIMPOLICY   VOLUMEBINDINGMODE      ALLOWVOLUMEEXPANSION   A
GE
local-path   rancher.io/local-path   Delete          WaitForFirstConsumer   false                  7
h34m
```

***Note**: trong bài lab sử dụng local path, provisioner này sẽ lấy trực tiếp trên node để tạo pv. Có thể thiết lập nhiều loại khác như nfs,...*

Mode của Storage class này là WaitForFirstConsumer. Volume binding mode gồm 2 loại:
1. **Immediate**: Khi volumeBindingMode được đặt thành **`Immediate`**, PVCs sẽ được ràng buộc (bind) ngay lập tức với PVs. Điều này có nghĩa là khi một PVC được tạo, Kubernetes sẽ ngay lập tức tìm kiếm một PV phù hợp và ràng buộc PVC với nó. Nếu không có PV phù hợp ngay lập tức, PVC sẽ không được ràng buộc và sẽ chờ cho đến khi một PV phù hợp xuất hiện.
2. **WaitForFirstConsumer**: Khi volumeBindingMode được đặt thành **`WaitForFirstConsumer`**, PVCs sẽ không được ràng buộc với PVs ngay lập tức. Thay vào đó, PV sẽ chỉ được ràng buộc khi PVC đầu tiên được sử dụng (consumer). Khi PVC được sử dụng cho một Pod, Kubernetes sẽ tìm kiếm một PV phù hợp và ràng buộc PVC với nó. Nếu không có PV phù hợp ngay lập tức, Kubernetes sẽ tạo một PV mới và ràng buộc PVC với nó.

Thiết lập Prometheus/ Grafana/ AlertManager cho quá trình monitoring

Prometheus là một hệ thống giám sát mã nguồn mở, được phát triển ban đầu bởi SoundCloud. Nó được thiết kế để thu thập và lưu trữ các số liệu giám sát từ các ứng dụng và hệ thống, sau đó phân tích và hiển thị chúng dưới dạng các biểu đồ và đồ thị. Prometheus sử dụng ngôn ngữ truy vấn PromQL để truy vấn dữ liệu giám sát và cung cấp các khả năng cảnh báo linh hoạt.

Grafana là một công cụ mã nguồn mở dùng để hiển thị dữ liệu giám sát. Nó cung cấp giao diện người dùng trực quan và linh hoạt để tạo biểu đồ, đồ thị và bảng điều khiển giám sát. Grafana hỗ trợ nhiều nguồn dữ liệu khác nhau, bao gồm cả Prometheus, và cho phép người dùng tạo ra các bản đồ tùy chỉnh và bảng điều khiển giám sát theo nhu cầu của mình.

Alertmanager là một thành phần của Prometheus được sử dụng để quản lý và gửi thông báo cảnh báo từ Prometheus tới các kênh nhận thông báo, như email, Slack hoặc các hệ thống gửi tin nhắn khác. Alertmanager cho phép người dùng tạo ra các quy tắc cảnh báo dựa trên các điều kiện giám sát và tùy chỉnh nội dung và hình thức của thông báo cảnh báo.

Cài đặt Prometheus

Thiết lập một volume để lưu trữ dữ liệu từ metrics:

Tạo PVC nối đến Storage Class đã tạo ở trên.

```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: prometheus-storage-pvc
  namespace: monitoring
spec:
  resources:
    requests:
      storage: 3Gi #Storage request
  accessModes:
    - ReadWriteOnce
  storageClassName: local-path
```

```
rodion@kubespray:~/kubernetes-prometheus$ kubectl get pvc -n monitoring
NAME                    STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS   AGE
prometheus-storage-pvc  Bound    pvc-ea6dd278-f7e3-428e-a5d9-18e166b1d19b   3Gi        RWO            local-path     6h50m
```

Khởi tạo volume với PVC đã tạo và mount vào thành thư mục lưu trữ cho prometheus trong file prometheus.yaml

```yaml
      containers:
        - name: prometheus
          image: prom/prometheus
          args:
            - "--config.file=/etc/prometheus/prometheus.yml"
            - "--storage.tsdb.path=/prometheus/data"
          ports:
            - containerPort: 9090
          volumeMounts:
            - name: prometheus-config-volume
              mountPath: /etc/prometheus/
            - name: prometheus-storage-volume
              mountPath: /prometheus/dât
      volumes:
        - name: prometheus-config-volume
          configMap:
            defaultMode: 420
            name: prometheus-server-conf
```

```
      - name: prometheus-storage-volume
        persistentVolumeClaim:
          claimName: prometheus-storage-pvc
```

Thêm cờ --storage.tsdb.retention.time (prometheus 2.7+) để tự động xóa dữ liệu sau một khoảng thời gian (default = 15d)

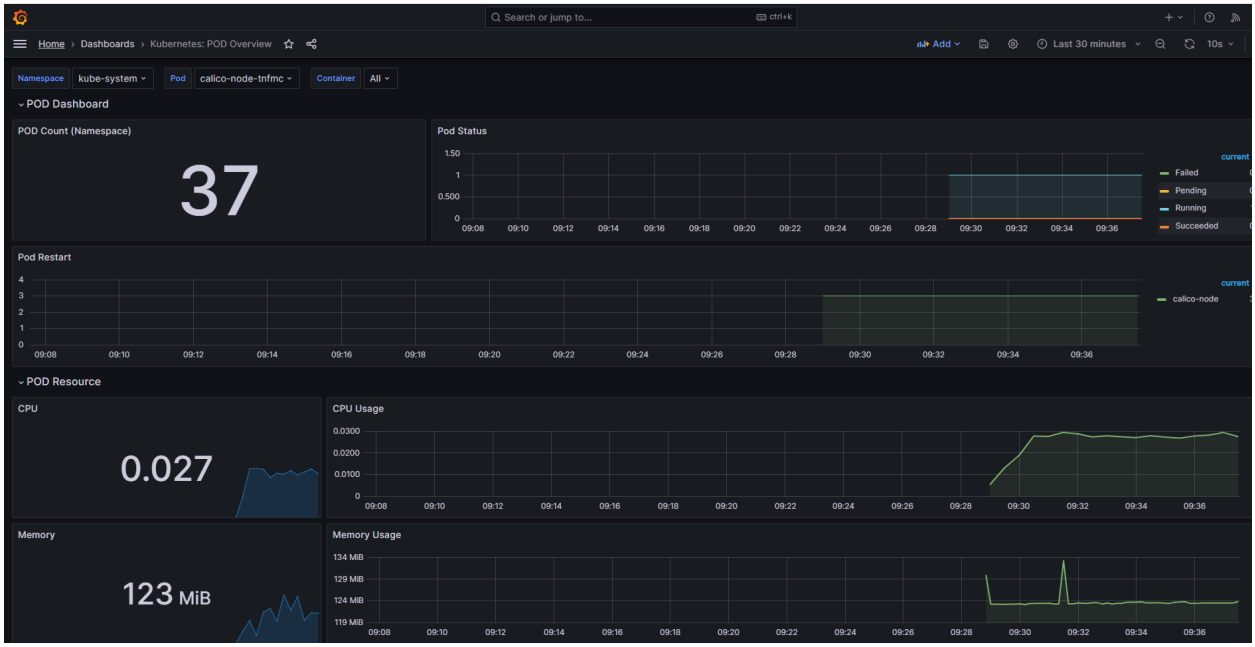
```
        args:
         - "--config.file=/etc/prometheus/prometheus.yml"
         - "--storage.tsdb.path=/prometheus/data"
         - "--storage.tsdb.retention.time=15d"
```

```
NAME                                       CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM                                STORAGECLASS   REASON   AGE
pvc-ea6dd278-f7e3-428e-a5d9-18e166b1d19b   3Gi        RWO            Delete           Bound    monitoring/prometheus-storage-pvc    local-path              6h51m
```

Thiết lập State Metrics

Thiết lập AlertManager

Thiết lập Grafana



Thiết lập Node Exporter để thu thập metrics về phần ứng/kernel của Linux

Thiết lập Metrics server để check Pod/Node resource usage

***Note**: Trường hợp metric-server bị dính lỗi liên quan đến certificate*

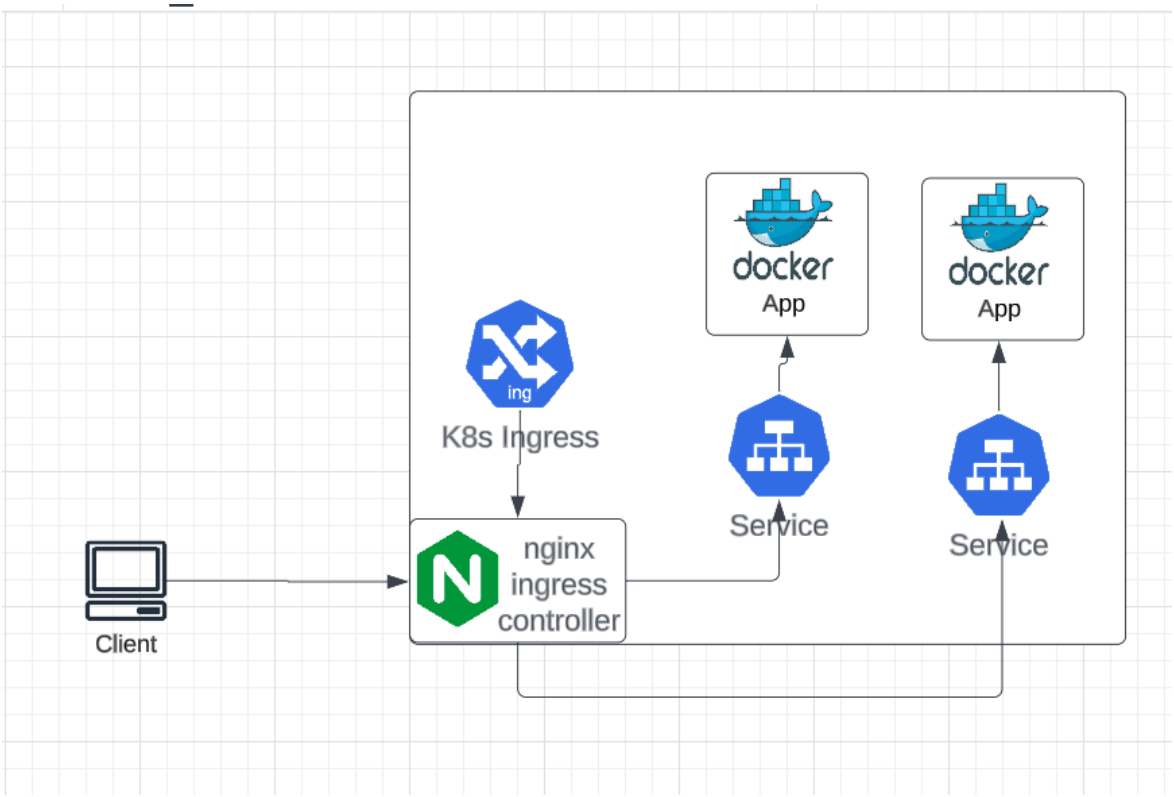
Quick fix: thêm --kubelet-insecure-tls vào args trong file yaml. Để bỏ qua tls, trong thực tế vẫn cần thiết vì lý do bảo mật.

***Note**: Trường hợp gặp lỗi DiskPressure khiến các pod liên tục bị envicted và tạo lại -> Tăng size disk của worker node lên và dùng các lệnh:*

```
#Give sda3 full space left
sudo growpart /dev/sda 3

# Increase the Physical Volume (pv) to max size
sudo pvresize /dev/sda3

# Expand the Logical Volume (LV) to max size to match
sudo lvresize -l +100%FREE /dev/mapper/ubuntu--vg-ubuntu--lv

# Expand the filesystem itself
sudo resize2fs /dev/mapper/ubuntu--vg-ubuntu--lv
```

***Node:** Có thể dùng kubeval để check cú pháp các file cấu hình k8s.*

Triển khai Ingress:



Sử dụng K8s ingress để tạo một entry cho các service ở bên trong Node.

Để K8s ingress resource hoạt động thì cần phải có Ingress controller. Ở đây dùng Nginx ingress controller

Thiết lập Nginx Ingress controller sử dụng helm chart

Install Helm 3

```
cd ~/

curl -fsSL -o get_helm.sh
https://raw.githubusercontent.com/helm/helm/master/scripts/get-helm-3

chmod 700 get_helm.sh

./get_helm.sh
```

```
rodion@k8s-master:~/test-ingress$ helm version
version.BuildInfo{Version:"v3.12.1", GitCommit:"f32a527a060157990e2aa86bf45010dfb3cc8b8d", GitTreeSt
ate:"clean", GoVersion:"go1.20.4"}
```

Deloy Nginx Ingress Controller trong cluster

```
controller_tag=$(curl -s
https://api.github.com/repos/kubernetes/ingress-nginx/releases/latest | grep tag_name | cut
-d '"' -f 4)

wget https://github.com/kubernetes/ingress-nginx/archive/refs/tags/${controller_tag}.tar.gz

tar xvf ${controller_tag}.tar.gz

cd ingress-nginx-${controller_tag}

cd charts/ingress-nginx/

kubectl create namespace ingress-nginx

helm install -n ingress-nginx ingress-nginx  -f values.yaml .
```

Kết quả:

```
rodion@k8s-master:~/test-ingress$ kubectl get all -n ingress-nginx
NAME                                        READY   STATUS    RESTARTS   AGE
pod/ingress-nginx-controller-7cdcd54cc6-nbbj7   1/1     Running   0          3h34m
pod/ingress-nginx-controller-7cdcd54cc6-rl7zp   0/1     Error     0          3h35m

NAME                                         TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)
                AGE
service/ingress-nginx-controller             NodePort    10.233.13.163   <none>        80:31562/TCP,
443:30626/TCP   3h35m
service/ingress-nginx-controller-admission   ClusterIP   10.233.42.169   <none>        443/TCP
                3h35m

NAME                                       READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/ingress-nginx-controller   1/1     1            1           3h35m

NAME                                                  DESIRED   CURRENT   READY   AGE
replicaset.apps/ingress-nginx-controller-7cdcd54cc6   1         1         1       3h35m
```

Trường hợp này deploy trên cụm K8s local, không có sẵn loadbalancer nên sử dụng Service của ingress controller là NodePort để truy cập vào Nginx Ingress controller thông qua Ip của Worker Node. NodePort chỉ dùng trong môi trường lab, thực tế vẫn cần triển khai qua loadbalancer.

Câu lệnh đổi type của service: (***Note**: Câu lệnh nhằm đổi nhanh type của service, tuy nhiên không nên dùng trong thực tế vì gây sai lệch giữa thực tế triển khai và file cấu hình. Thay vào đó nên update file cấu hình và apply lại*)

```
kubectl patch svc ingress-nginx-controller -n ingress-nginx --type='json' -p
'[{"op":"replace","path":"/spec/type","value":"NodePort"}]'
```

Triển khai app test

```
kind: Pod
apiVersion: v1
metadata:
  name: apple-app
  labels:
    app: apple
spec:
  containers:
    - name: apple-app
      image: hashicorp/http-echo
      args:
        - "-text=apple"
---

kind: Service
apiVersion: v1
metadata:
  name: apple-service
spec:
  selector:
    app: apple
  type: ClusterIP
  ports:
    - port: 80
      targetPort: 5678 # Default port for image
---
kind: Pod
apiVersion: v1
metadata:
  name: banana-app
  labels:
    app: banana
spec:
  containers:
    - name: banana-app
      image: hashicorp/http-echo
      args:
        - "-text=banana"
---

kind: Service
apiVersion: v1
metadata:
```

```
  name: banana-service
spec:
  selector:
    app: banana
  type: ClusterIP
  ports:
    - port: 80
      targetPort: 5678 # Default port for image
```

2 pod chạy và 2 service liên kết đến các pod với cổng 5678, type là ClusterIP. Tạo các resource trên bằng lệnh create.

```
rodion@k8s-master:~/test-ingress$ kubectl create -f demo-app.yml -n demo
pod/apple-app created
service/apple-service created
pod/banana-app created
service/banana-service created
```

```
rodion@k8s-master:~/test-ingress$ kubectl get pods -n demo
NAME         READY   STATUS    RESTARTS   AGE
apple-app    1/1     Running   0          53m
banana-app   1/1     Running   0          53m
rodion@k8s-master:~/test-ingress$ kubectl get svc -n demo
NAME             TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)   AGE
apple-service    ClusterIP   10.233.29.242   <none>        80/TCP    53m
banana-service   ClusterIP   10.233.51.71    <none>        80/TCP    53m
```

Khi các resource hoạt động bình thường, tạo Ingress resource để tạo entry cho các Service với mỗi Service là một path riêng.
Tạo một file .yaml với nội dung:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: webapp-ingress
spec:
  ingressClassName: nginx
  rules:
  - host: webapp.k8s.example.com
    http:
      paths:
        - path: /banana
          pathType: Prefix
```

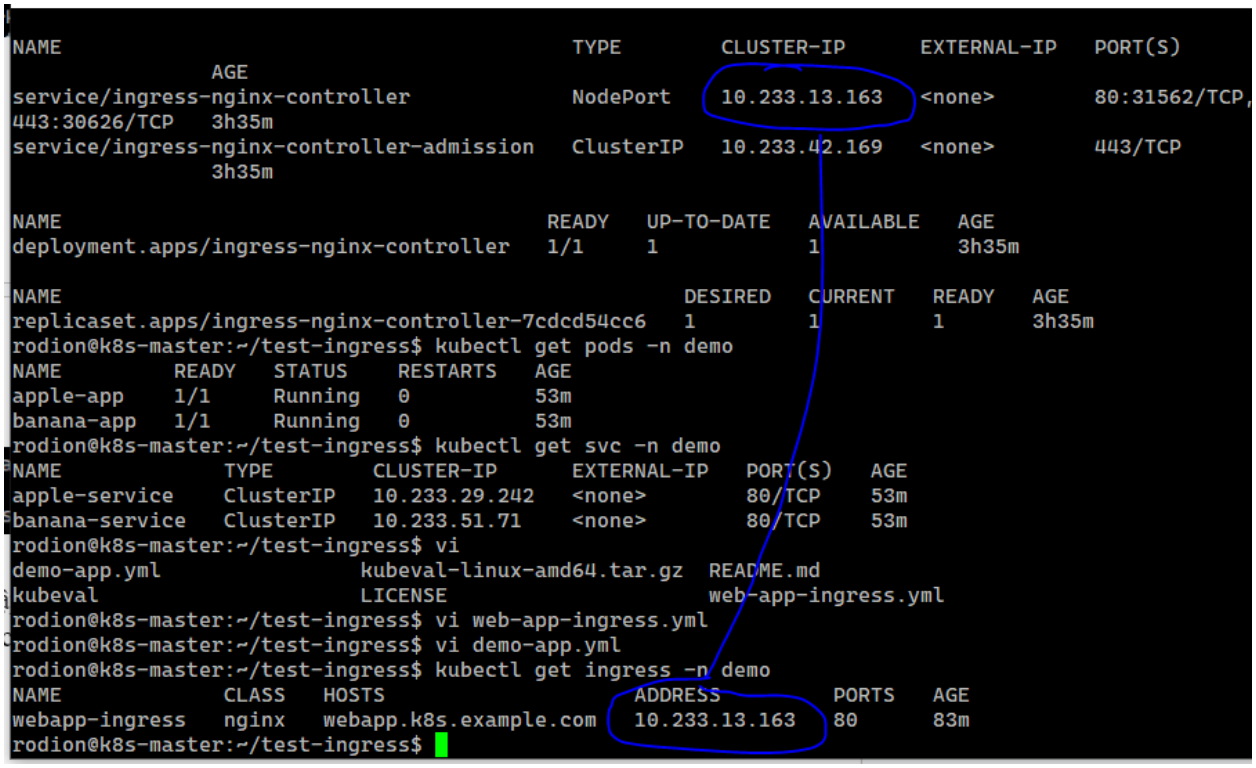
```
          backend:
            service:
              name: banana-service
              port:
                number: 80
        - path: /apple
          pathType: Prefix
          backend:
            service:
              name: apple-service
              port:
                number: 80
```

/banana sẻ trỏ đến banana-service, tương tự với /apple sẽ trỏ tới apple-service
Ingress sẽ để traffic vào port 80, Service nhận từ port 80 và forward đến targetPort 5678 theo cấu hình.

```
rodion@k8s-master:~/test-ingress$ kubectl apply -f web-app-ingress.yml -n demo
ingress.networking.k8s.io/webapp-ingress configured
```

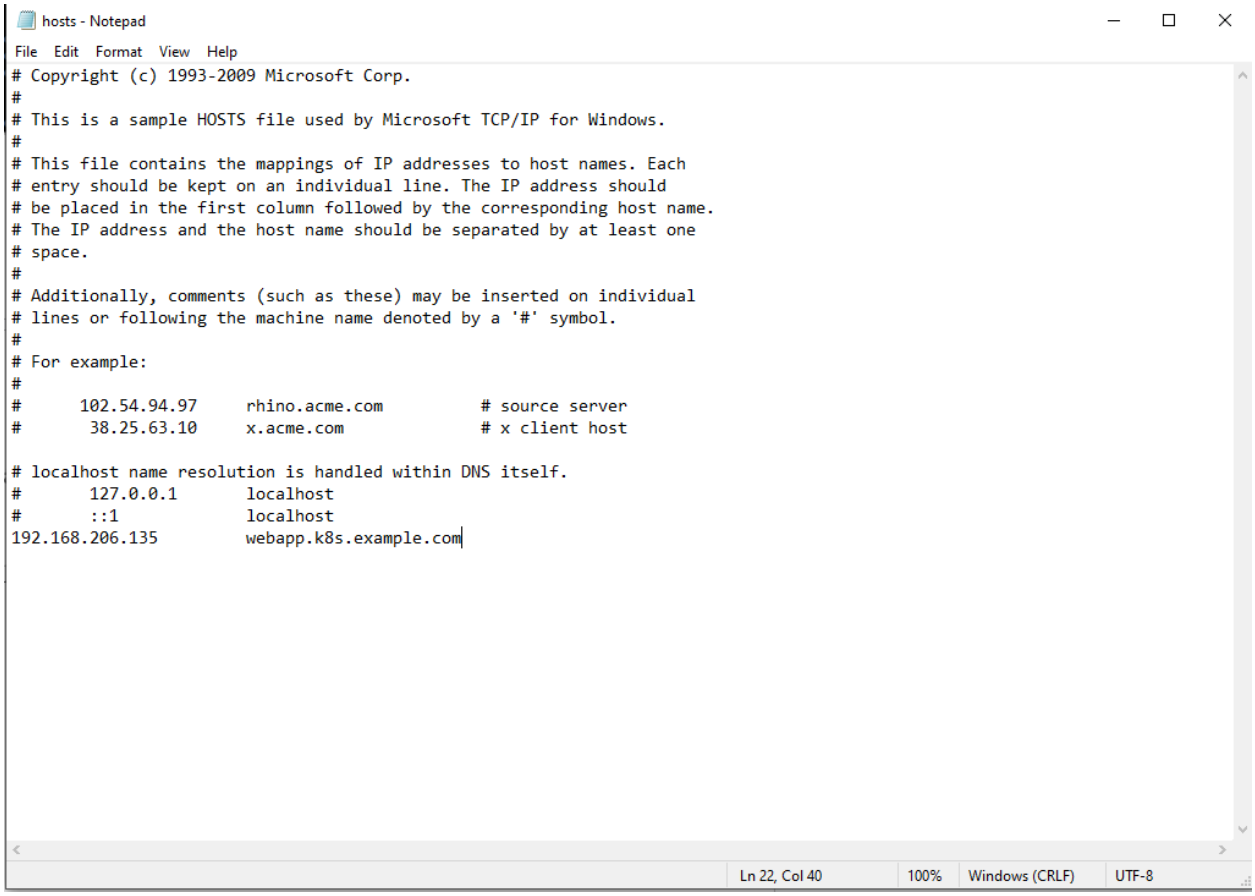
```
rodion@k8s-master:~/test-ingress$ kubectl get ingress -n demo
NAME             CLASS   HOSTS                   ADDRESS   PORTS   AGE
webapp-ingress   nginx   webapp.k8s.example.com             80      16s
```

Sau khi truy cập vào nginx ingress controller sẽ thấy địa chỉ của ingress controller được gắn vào phần address

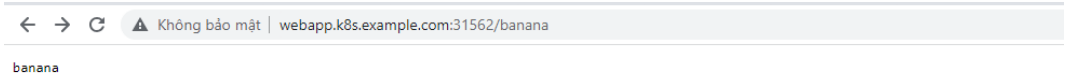


Những thiết lập từ Ingress resource sẽ được Nginx ingress controller đưa vào config.

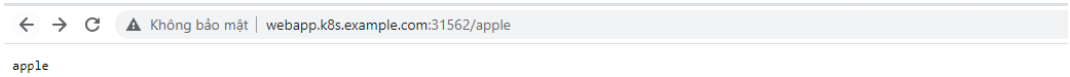
Vì tạm thời không có domain nên fake một domain trong file hosts trên window.



Domain trên trỏ tới địa chỉ IP của Worker Node 1.
Truy cập kiểm tra kết quả:

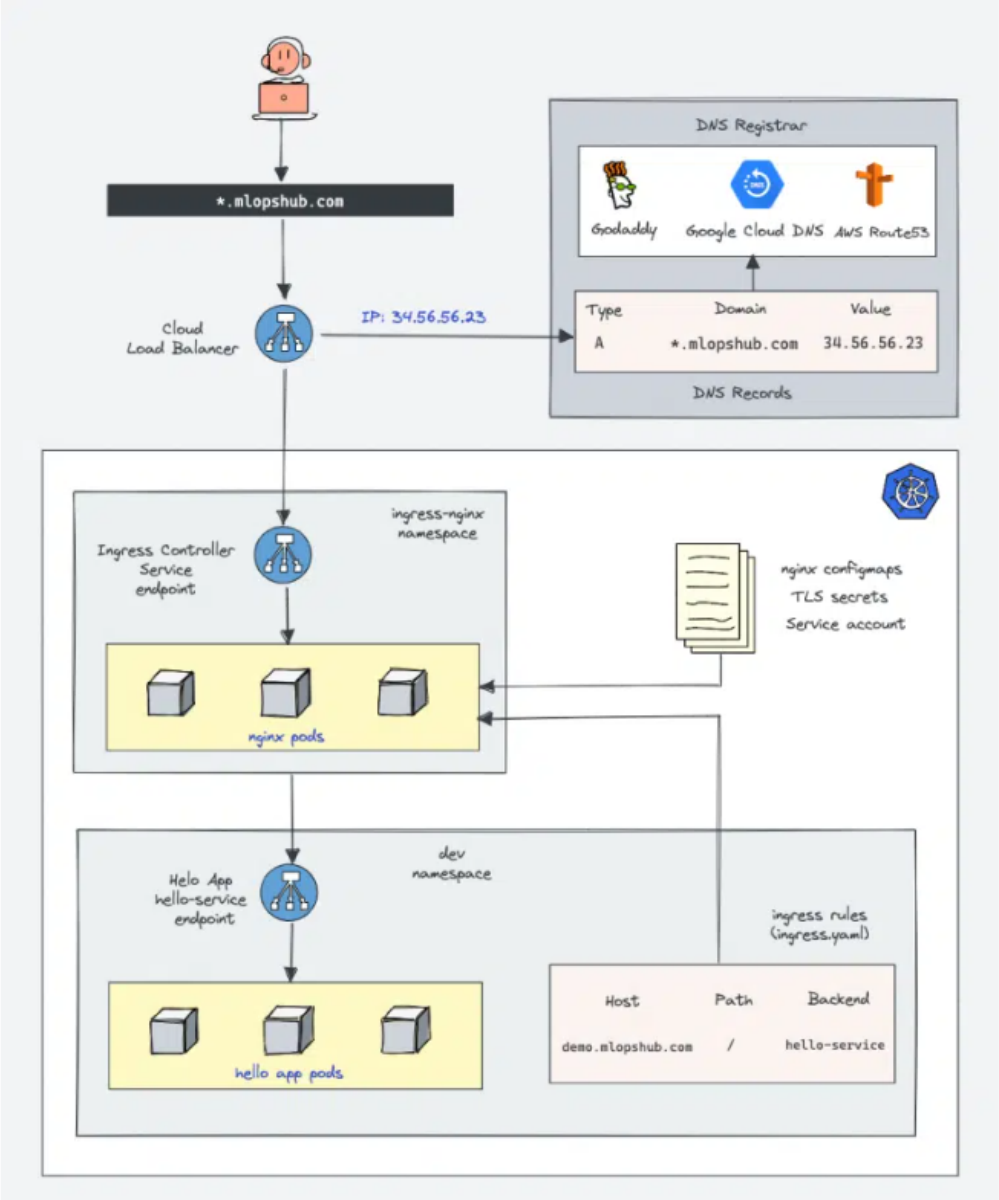


banana



Địa chỉ của domain tương ứng với phần domain đã thiết lập trong hosts của file Ingress resource. Port sau đó là của NodePort, và sau đó là path đến các service.

***Note**: Trường hợp truy cập trực bằng địa chỉ Ip của Worker Node (192.168.206.135:31562/<path>) sẽ không hoạt động, dù chúng hướng tới cùng một nơi. Cần truy cập bằng domain để có thể gửi kèm một DNS record type A. domain trong record này sẽ được so sánh với domain trong host đã được định nghĩa trong Ingress resource trước đó.*



Link tham khảo

https://computingforgeeks.com/deploy-nginx-ingress-controller-on-kubernetes-using-helm-chart/?expand_article=1

https://devopscube.com/setup-ingress-kubernetes-nginx-controller/

Triển khai Ingress cho Prometheus/ Grafana/ AlertManager

Đổi Type của Service sang Cluster IP

```yaml
apiVersion: v1
kind: Service
metadata:
  name: grafana
  namespace: monitoring
  annotations:
      prometheus.io/scrape: 'true'
      prometheus.io/port:   '3000'
spec:
  selector:
    app: grafana
  type: ClusterIP
  ports:
    - port: 3000
      targetPort: 3000
        #       nodePort: 32000
~
~
```

```
rodion@k8s-master:~$ kubectl get svc -n monitoring
NAME                TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE
alertmanager        ClusterIP   10.233.19.109   <none>        9093/TCP   5m26s
grafana             ClusterIP   10.233.43.128   <none>        3000/TCP   2m47s
node-exporter       ClusterIP   10.233.39.245   <none>        9100/TCP   25h
prometheus-service  ClusterIP   10.233.62.3     <none>        9090/TCP   13m
```

Viết file Ingress với liên kết với Service

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: grafana-ingress
  namespace: monitoring
spec:
  ingressClassName: nginx
  rules:
  - host: grafana.example.com
    http:
      paths:
        - path: /
          pathType: Prefix
          backend:
            service:
              name: grafana
              port:
                number: 3000
~
~
~
```

```
rodion@k8s-master:~/kubernetes-prometheus/kubernetes-grafana$ kubectl get ingress -n monitoring
NAME                CLASS   HOSTS                    ADDRESS         PORTS   AGE
alert-ingress       nginx   alert.example.com        10.233.13.163   80      7m46s
grafana-ingress     nginx   grafana.example.com      10.233.13.163   80      11m
prometheus-ingress  nginx   prometheus.example.com   10.233.13.163   80      25m
```

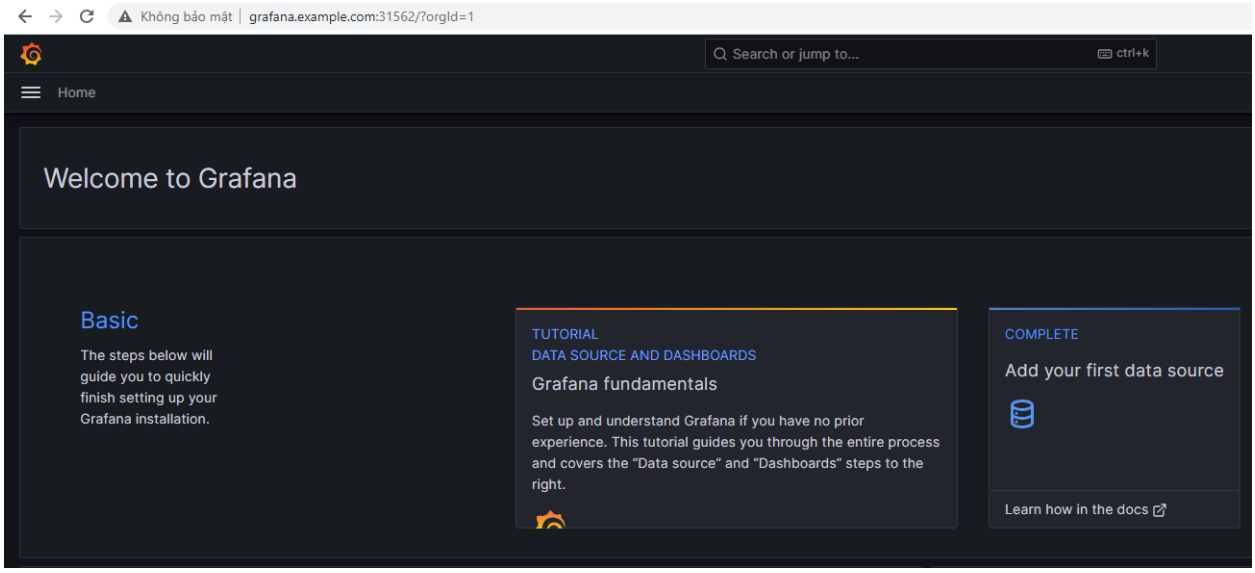
File host trong window

hosts - Notepad

File Edit Format View Help

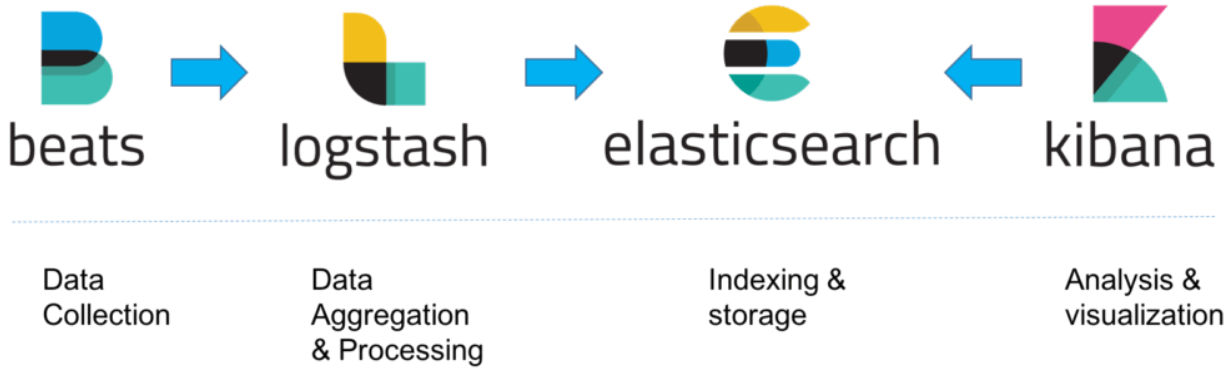
```
# Copyright (c) 1993-2009 Microsoft Corp.
#
# This is a sample HOSTS file used by Microsoft TCP/IP for Windows.
#
# This file contains the mappings of IP addresses to host names. Each
# entry should be kept on an individual line. The IP address should
# be placed in the first column followed by the corresponding host name.
# The IP address and the host name should be separated by at least one
# space.
#
# Additionally, comments (such as these) may be inserted on individual
# lines or following the machine name denoted by a '#' symbol.
#
# For example:
#
#      102.54.94.97     rhino.acme.com          # source server
#       38.25.63.10     x.acme.com              # x client host

# localhost name resolution is handled within DNS itself.
#       127.0.0.1       localhost
#       ::1             localhost
#192.168.206.135        webapp.k8s.example.com
192.168.206.136         prometheus.example.com
192.168.206.136         grafana.example.com
192.168.206.136         alert.example.com
```

Truy cập thông qua domain:



Triển khai hệ thống logging.

Sử dụng ElasticStack gồm các thành phần như sau:



Mô hình sử dụng filebeat để thu thập log từ các container trong /var/log/containers -> logstash -> elasticsearch -> kibana. Sử dụng helm chart để thiết lập các thành phần.

Tao thư mục chứa các thành phần

```
rodion@k8s-master:~/ELK2$ ls -l
total 16
drwxrwxr-x 3 rodion rodion 4096 Jul  7 02:48 elastic
drwxrwxr-x 3 rodion rodion 4096 Jul  7 04:26 filebeat
drwxrwxr-x 3 rodion rodion 4096 Jul  7 03:00 kibana
drwxrwxr-x 3 rodion rodion 4096 Jul  7 04:24 logstash
```

***Note**: Cách cài trong bài lab đang sử dụng trực tiếp helm để triển khai các thành phần. Trong tình huống thực thế nên trích xuất file cấu hình .yaml của resource để tự custom bằng lệnh* `kubectl get <resource> -n <namespace> -oyaml`

Tiển khai Elasticsearch:

```
cd elastic

helm repo add elastic https://Helm.elastic.co

helm search repo elastic/elasticsearch
```

```
helm pull elastic/elasticsearch --version 7.17.3

tar -xzf elasticsearch-7.17.3.tgz

cp elasticsearch/values.yaml value-elasticsearch.yaml

vi value-elasticsearch.yaml
```

***Note**: Phiên bản ElasticStack sử dụng trong bài lab là 7.17.3. Phiên bản mới nhất là 8.5.1. Có thể tìm phiên bản mới bằng lệnh helm search.*

Điều chỉnh cấu hình trong file .yaml:

***Note**: Do vấn đề tài nguyên, các thiết lập cấu hình là minimum và chỉ nên dành cho mục đích Lab. Tùy vào thực tế triển khai để điều chỉnh cho phù hợp.*

```
replicas: 1
minimumMasterNodes: 1
```

```
resources:
  requests:
    cpu: "1000m"
    memory: "1Gi"
  limits:
    cpu: "1000m"
    memory: "1Gi"
```

```
persistence:
  enabled: true
  labels:
    # Add default labels for the volumeCl
    enabled: false
  annotations: {}
```

Cấu hình sử dụng persistent volume để lưu trữ

```
# Hard means that by defa
# and that they will neve
antiAffinity: "soft"
```

Để antiAffinity thành soft trong trường hợp có ít node. Hard không phân bổ nhiều pod / node -> nhiều pod hơn node có thể dẫn đến pending.

***Note**: ở phần volume resource, thêm cấu hình storageClassName: "<storage-class>" vào để có thể nối PVC với storage class nhanh hơn, tránh tình trạng pending*

```
ingress:
  enabled: true
  annotations: {}
  # kubernetes.io/ingress.class: nginx
  # kubernetes.io/tls-acme: "true"
  className: "nginx"
  pathtype: ImplementationSpecific
  hosts:
    - host: elasticsearch.example.com
      paths:
        - path: /
  tls: []
  #  - secretName: chart-example-tls
  #    hosts:
  #      - chart-example.local
```

Cấu hình ingress với domain để có thể truy cập. (Lưu ý phải có ingress controller)

```
volumeClaimTemplate:
  accessModes: ["ReadWriteOnce"]
  resources:
    requests:
      storage: 1Gi
```

Cấu hình dung lượng lưu trữ. Vì mục đích bài lab nên chỉ để storage nhỏ.

***Note**: Trong trường hợp có nhiều storage class thì phải set một cái là default. Nhưng tốt hơn vẫn nên định nghĩa thẳng Storage class sử dụng trong volumeClaimTemplate*

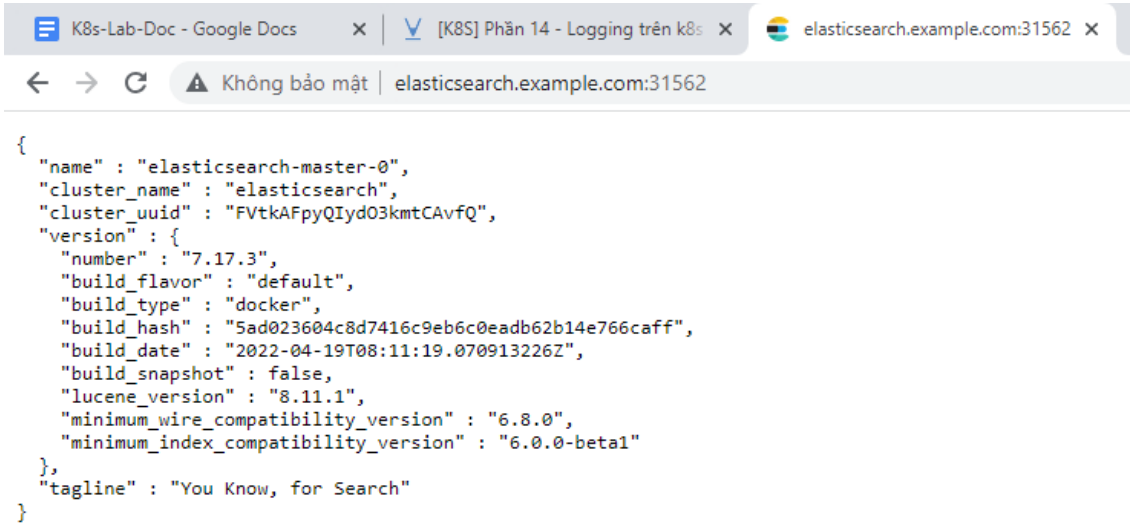
```
kubectl patch storageclass <storageclass-name> -p '{"metadata":
{"annotations":{"storageclass.kubernetes.io/is-default-class":"true"}}
}'
```

Cài đặt Elasticsearch bằng helm

```
helm -n prod install elasticsearch -f value-elasticsearch.yaml
elasticsearch
```

```
NAME                     READY   STATUS    RESTARTS   AGE
elasticsearch-master-0   1/1     Running   0          4h7m
```

Điền tên miền trong file hosts của window và truy cập từ web client



***Note**: Elasticsearch trong bài lab sử dụng 7.17.3, có thể nhận kết nối http. Với các phiên bản Elasticsearch mới hơn (ex 8.5.1) sẽ nhận mặc định là https.*

***Note**: Khi uninstall Elasticsearch bằng helm uninstall, PVC và PV được tạo ra trong quá trình triển khai vẫn được giữ lại.*

***Note**: Thay vì uninstall có thể scale các pod về 0 và apply lại cấu hình*

Triển khai Kibana:

Tương tự quá trình pull về và giải nén của Elasticsearch.

```
helm search repo elastic/kibana

helm pull elastic/kibana --version 7.17.3

tar -xzf kibana-7.17.3.tgz

cp kibana/values.yaml value-kibana.yaml

vi value-kibana.yaml
```

Điều chỉnh cấu hình kibana:

```
---
elasticsearchHosts: "http://elasticsearch-master:9200"

replicas: 1
```

Cấu hình kết nối đến Elasticsearch (trỏ tới dns domain của service của Elasticsearch) với cổng 9200. Với các phiên bản Elasticsearch nhận https. Cần đổi thành https và thêm cấu hình cert (Các phiên bản kibana mới mặc định có)

```
resources:
  requests:
    cpu: "1000m"
    memory: "1Gi"
  limits:
    cpu: "1000m"
    memory: "1Gi"
```

```
ingress:
  enabled: true
  className: "nginx"
  pathtype: ImplementationSpecific
  annotations: {}
  # kubernetes.io/ingress.class: nginx
  # kubernetes.io/tls-acme: "true"
  hosts:
    - host: kibana-example.com
      paths:
        - path: /
```

Cấu hình ingress để truy cập. (nếu không có ingress controller có thể dùng service)

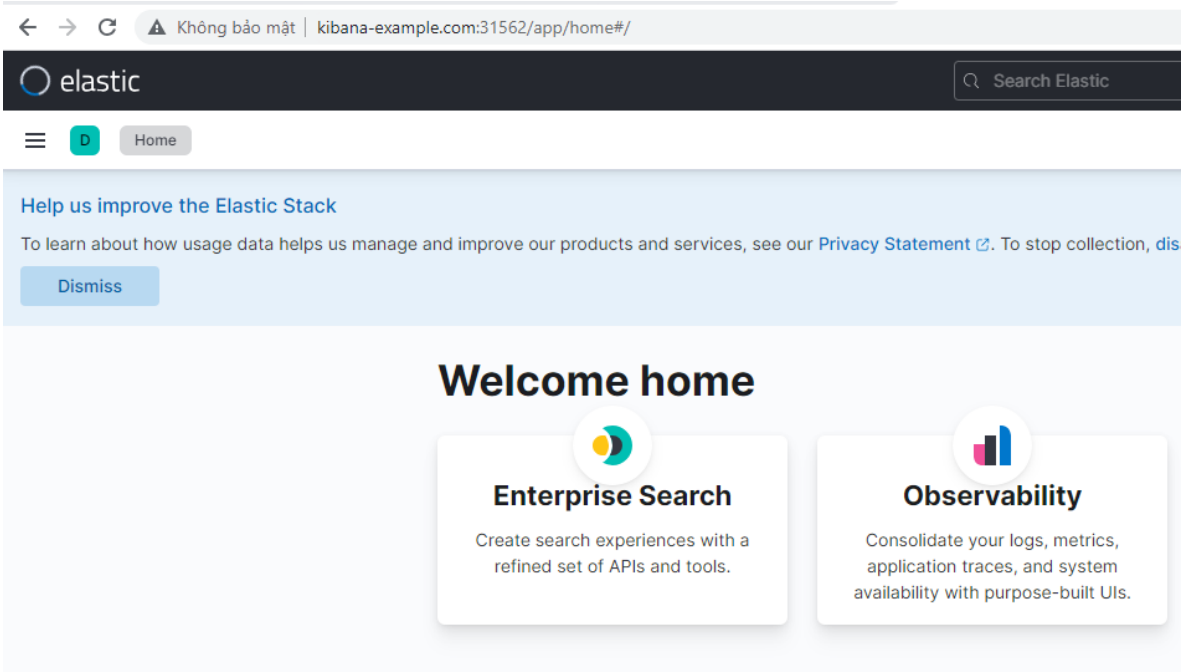
Tiến hành triển khai kibana:

```
helm -n prod install kibana -f value-kibana.yaml kibana
```

```
kibana-kibana-6f7c4b888c-cctfk   1/1     Running                0               4h57m
```

***Note**: Quá trình triển khai có thể tốn nhiều thời gian do phải thiết lập nhiều thành phần. Các pod pre-install-kibana sẽ được tạo ra để thực hiện các job.Tương tự với quá trình uninstall, sẽ có các pod post-delete-kibana được tạo ra. Các pod này có thể không hoạt động tốt dẫn đến các thành phần trên vẫn tồn tại gây cản trở quá trình uninstall hoặc install. Dùng các lệnh sau để xóa thủ công:*

```
kubectl delete configmap kibana-kibana-helm-scripts -n prod
kubectl delete serviceaccount pre-install-kibana-kibana -n prod
kubectl delete serviceaccount post-delete-kibana-kibana -n prod
kubectl delete roles pre-install-kibana-kibana -n prod
kubectl delete roles post-delete-kibana-kibana -n prod
kubectl delete rolebindings pre-install-kibana-kibana -n prod
kubectl delete rolebindings post-delete-kibana-kibana -n prod
kubectl delete job pre-install-kibana-kibana -n prod
kubectl delete post-delete-kibana-kibana -n prod
kubectl delete post-delete-kibana-kibana -n prod
kubectl delete secrets kibana-kibana-es-token
```



Triển khai logstash:

```
helm pull elastic/logstash --version 7.17.3

tar -xzf logstash-7.17.3.tgz

cp logstash/values.yaml value-logstash.yaml

vi value-logstash.yaml
```

Thiết lập cấu hình:

```
logstashPipeline:
  logstash.conf: |
    input {
        beats {
            port => "5044"
      }
    }
    filter {
        grok {
            add_field => [ "received_at", "%{@timestamp}" ]
        }
    }
    output {
    if "demo" in [tags] {
       elasticsearch {
         hosts => [ "http://elasticsearch-master:9200" ]
         index => "logstash_demo_%{+YYYY.MM.dd}"
         }
       }
       else {
       elasticsearch {
         hosts => [ "http://elasticsearch-master:9200" ]
         index => "logstash_default_%{+YYYY.MM.dd}"
         }
       }
    }
```

Theo cấu hình, logstash sẽ nhận input từ filebeat ở port 5044, filter theo timestamp. Sau đó log được gửi đến elasticsearch theo 2 loại, các log có tag demo và các log không.

***Note**: hosts thông thường chỉ cần trỏ đến tên service của Elasticsearch, nhưng phiên bản này gặp thì gặp một số lỗi. Log có thể vẫn được gửi đến elasticsearch:9200 thay vì elasticsearch-master. Sau khi thêm vào như trên và chạy lại thì http://elasticsearch-master:9200 được thêm vào Url pool của logstash (Tức là log được gửi đi ở cả http://elasticsearch:9200 và http://elasticsearh-master:9200. Đối với các phiên bản mới (ex 8.5.1) Elasticsearch nhận mặc định là https nên phải chuyển thành https://elasticsearch-master:9200 và thiết lập cert cho logstash.*

```
resources:
  requests:
    cpu: "100m"
    memory: "1536Mi"
  limits:
    cpu: "1000m"
    memory: "1536Mi"
```

```
persistence:
  enabled: false
  annotations: {}
```

Do logstash chỉ nhận nhiệm vụ parse log và gửi đến Elasticsearch nên không cần thiết lập volume để lưu trữ.

```
antiAffinity: "soft"
```

```
service:
  annotations: {}
  type: ClusterIP
  loadBalancerIP: ""
  ports:
    - name: beats
      port: 5044
      protocol: TCP
      targetPort: 5044
    - name: http
      port: 8080
      protocol: TCP
      targetPort: 8080
```

Enable service để filebeat có thể biết địa chỉ gửi log tới. Do không cần truy cập từ ngoài nên không cần enable ingress

Triển khai logstash:

```
helm -n prod install logstash -f value-logstash.yaml logstash
```

```
logstash-logstash-0              1/1     Running     2 (18m ago)   4h33m
```

***Note**: khi logstash bắt đầu vào trạng thái running, theo dõi log của nó để đảm bảo quá trình khởi động được diễn ra đúng. Sử dụng grep để tìm http://elasticsearch-master:9200 để đảm bảo đã được thêm vào url pool và có thể gửi log đến elasticsearch thành công.*

```
rodion@k8s-master:~/ELK2/elastic$ kubectl logs -n prod logstash-logstash-0 | grep elasticsearch-mast
er
[2023-07-07T08:00:21,084][INFO ][logstash.outputs.elasticsearch][main] New Elasticsearch output {:cl
ass=>"LogStash::Outputs::ElasticSearch", :hosts=>["http://elasticsearch-master:9200"]}
[2023-07-07T08:00:21,104][INFO ][logstash.outputs.elasticsearch][main] Elasticsearch pool URLs updat
ed {:changes=>{:removed=>[], :added=>[http://elasticsearch-master:9200/]}}
[2023-07-07T08:00:21,490][WARN ][logstash.outputs.elasticsearch][main] Restored connection to ES ins
tance {:url=>"http://elasticsearch-master:9200/"}
[2023-07-07T08:00:21,682][INFO ][logstash.outputs.elasticsearch][main] New Elasticsearch output {:cl
ass=>"LogStash::Outputs::ElasticSearch", :hosts=>["http://elasticsearch-master:9200"]}
[2023-07-07T08:00:21,762][INFO ][logstash.outputs.elasticsearch][main] Elasticsearch pool URLs updat
ed {:changes=>{:removed=>[], :added=>[http://elasticsearch-master:9200/]}}
[2023-07-07T08:00:21,841][WARN ][logstash.outputs.elasticsearch][main] Restored connection to ES ins
```

Triển khai Filebeat

```
helm pull elastic/filebeat --version 7.17.3

tar -xzf filebeat-7.17.3.tgz

cp filebeat/values.yaml value-filebeat.yaml

vi value-filebeat.yaml
```

Thiết lập cấu hình:

```
filebeatConfig:
   filebeat.yml: |
     filebeat.inputs:
```

```
    - type: container
      paths:
        - /var/log/containers/*.log
      processors:
      - add_kubernetes_metadata:
          host: ${NODE_NAME}
          matchers:
          - logs_path:
              logs_path: "/var/log/containers/"
    #output.elasticsearch:
    #  host: '${NODE_NAME}'
    #  hosts: '${ELASTICSEARCH_HOSTS:elasticsearch-master:9200}'
    output.logstash:
      hosts: [ "logstash-logstash:5044" ]
```

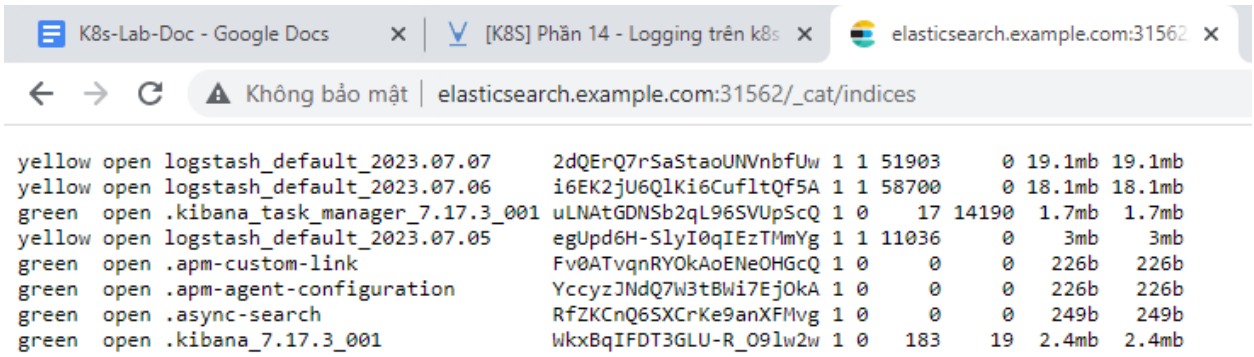
Thiết lập filebeat lấy tất cả log từ /var/log/containers/*.logs và gửi output đến logstash.

Triển khai filebeat

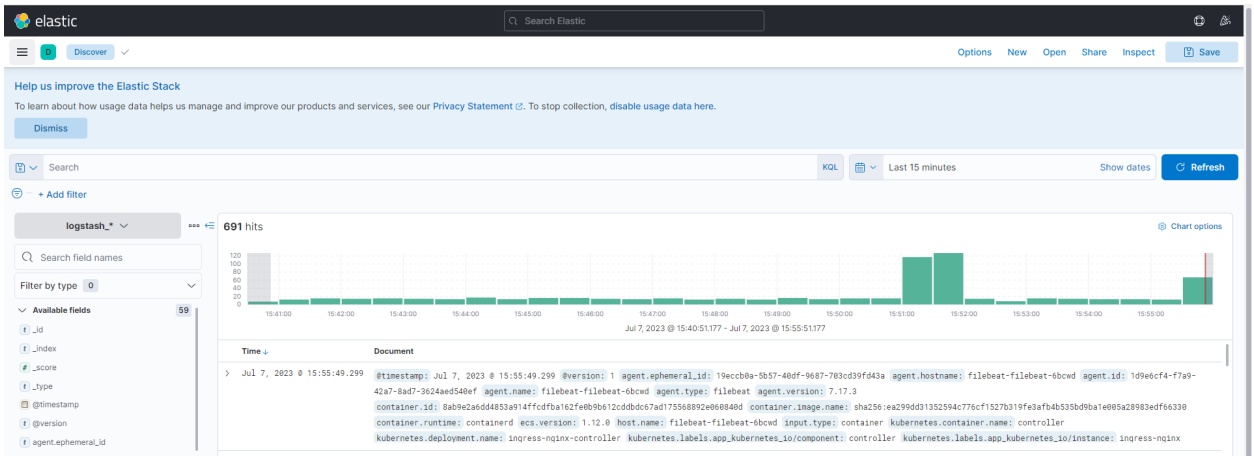
```
helm install -n prod filebeat -f value-filebeat.yaml filebeat
```

```
filebeat-filebeat-6bcwd          1/1     Running     0              4h1m
```

Kiểm tra xem log có được gửi lên Elastic search không.



Có các indices của logstash trên Elasticsearch.



Log được gửi từ logstash được visualize trên Kibana.

***Note**: Các triển khai ở đây chưa sử dụng HTTPS. Nên triển khai thêm cấu hình SSL để đảm bảo thông tin được bảo mật.*

[Link tham khảo]

Tổng hợp phiên bản sử dụng:

Ubuntu: 22.04.2
Kubespray: 2.19
Kubernetes: v1.23.7 (Kubespray 2.19 default)
Helm: 3.12.1
Prometheus Stack: latest
ElasticStack: 7.17.3

Các config files