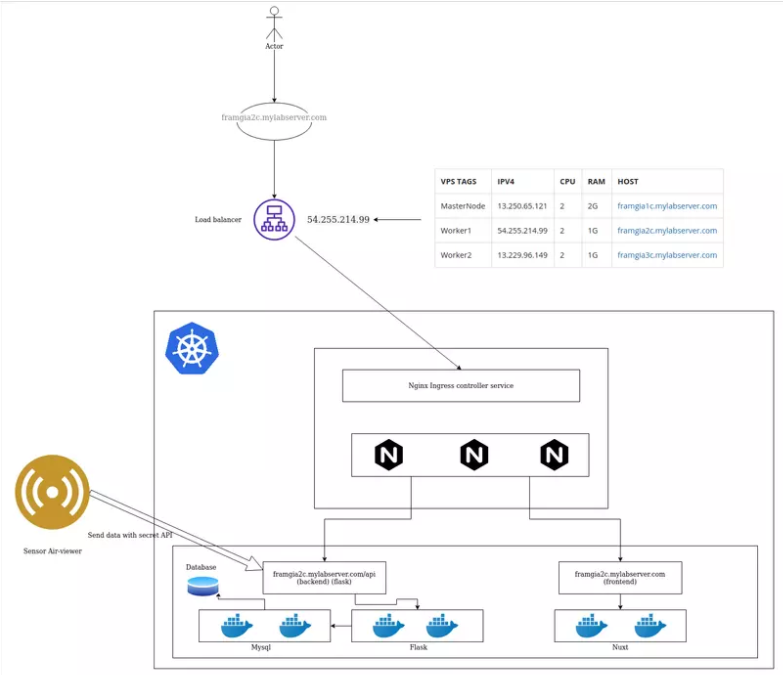
**Vậy flannel là gì ?**

Đầu tiên chúng ta có một định nghĩa: Agents (deamons, kuberlets) là node có thể giao tiếp với tất cả các pods trên node đó.

Flannel là một virtual network được thiết kế bới chính kubernetes. Mỗi host trong flannel cluster chạy một agent được gọi là flannelID. Chúng được cấp phép cho mỗi host một subnet trên địa chỉ IP trên container được chạy mỗi host. Containers có thể giao tiếp với các containers khác thông qua sử dụng địa chỉ IP. Flannel hỗ trợ multiple backend để đóng gói cá gói tin. Về sự lựa chọn thông thường thì là Virtual Extensible LAN (VXLAN), chúng chạy một layer có 2 network trên một kiến trúc của Layer 3. Flannel cũng hỗ trợ host-gw, chúng sẽ map chính xác các routes trong host được dùng bên trong Calico được miêu tả thêm ở link dưới.



## Deploy the Ingress Controller

Chúng ta có 2 lựa chọn để triển khai Ingress Controller

* Deployment: Nếu bạn có kế hoạch thay đổi số lượng Ingress controller replicas
* DaemonSet: Triển khai Ingress controller trên mỗi Node hoặc các node con (mình sẽ dùng cái này)

**Cách Deamon-Set**

apiVersion: apps/v1

kind: DaemonSet

metadata:

name: nginx-ingress

namespace: nginx-ingress

spec:

selector:

matchLabels:

app: nginx-ingress

template:

metadata:

labels:

app: nginx-ingress

#annotations:

#prometheus.io/scrape: "true"

#prometheus.io/port: "9113"

spec:

serviceAccountName: nginx-ingress

containers:

- image: nginx/nginx-ingress:edge

imagePullPolicy: Always

name: nginx-ingress

ports:

- name: http

containerPort: 80

hostPort: 80

- name: https

containerPort: 443

hostPort: 443

#- name: prometheus

#containerPort: 9113

securityContext:

allowPrivilegeEscalation: true

runAsUser: 101 #nginx

capabilities:

drop:

- ALL

add:

- NET\_BIND\_SERVICE

env:

- name: POD\_NAMESPACE

valueFrom:

fieldRef:

fieldPath: metadata.namespace

- name: POD\_NAME

valueFrom:

fieldRef:

fieldPath: metadata.name

args:

- -nginx-configmaps=$(POD\_NAMESPACE)/nginx-config

- -default-server-tls-secret=$(POD\_NAMESPACE)/default-server-secret

#- -v=3 # Enables extensive logging. Useful for troubleshooting.

#- -report-ingress-status

#- -external-service=nginx-ingress

#- -enable-leader-election

#- -enable-prometheus-metrics

để enable báo cáo trạng thái của ingress thì chúng ta uncomment 5 dòng cuối.

## Truy cập được vào Ingress Controller

Sẽ có 2 cách browser client truy cập. Cách thứ nhất có thể dùng kiểu NodePort sẽ để Nginx-Ingress chạy cùng với các Worker-Node (không dùng loadbalancer của các service như AWS, Gcloud, Azure... ). Cách dùng NodePort này không nên dùng trực tiếp đối với các serivces ngoài ingress này nhé vì an toàn bảo mật ở trong bài viết trước đã đề cập.

**Example Deploy Nginx after install Ingress**

apiVersion: apps/v1

kind: Deployment

metadata:

name: coffee

spec:

replicas: 2

selector:

matchLabels:

app: coffee

template:

metadata:

labels:

app: coffee

spec:

containers:

- name: coffee

image: nginxdemos/hello:plain-text

ports:

- containerPort: 80

---

apiVersion: v1

kind: Service

metadata:

name: coffee-svc

spec:

ports:

- port: 80

targetPort: 80

protocol: TCP

name: http

selector:

app: coffee

---

apiVersion: apps/v1

kind: Deployment

metadata:

name: tea

spec:

replicas: 3

selector:

matchLabels:

app: tea

template:

metadata:

labels:

app: tea

spec:

containers:

- name: tea

image: nginxdemos/hello:plain-text

ports:

- containerPort: 80

---

apiVersion: v1

kind: Service

metadata:

name: tea-svc

labels:

spec:

ports:

- port: 80

targetPort: 80

protocol: TCP

name: http

selector:

app: tea

2 web service coffee và tea và ta sẽ cho chúng nó dùng chung 1 Host là framgia2c.mylabserver.com của worker Node 1

**Ingress resource**

apiVersion: extensions/v1beta1

kind: Ingress

metadata:

name: cafe-ingress

spec:

tls:

- hosts:

- framgia2c.mylabserver.com

secretName: cafe-secret

rules:

- host: framgia2c.mylabserver.com

http:

paths:

- path: /tea

backend:

serviceName: tea-svc

servicePort: 80

- path: /coffee

backend:

serviceName: coffee-svc

servicePort: 80

# Deploy Air-viewer trên kubernetes sử dụng nginx ingress

Chúng ta có các services như Flask, mysql, nuxt.js được định tuyến chung với nginx-ingress sau đó sử dụng luôn worker node làm domain, loadbalancer. Dự án đo nồng độ ô nhiễm bụi trong không khí này cũng được xây dựng microservices các bạn có thể tham khảo tại đây <https://github.com/sun-asterisk-research/air-viewer>

## Clone Project Air-viewer

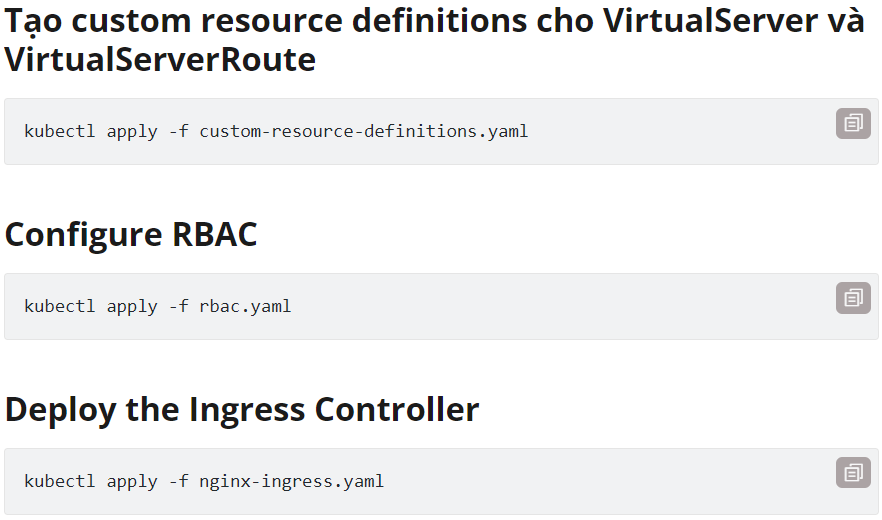
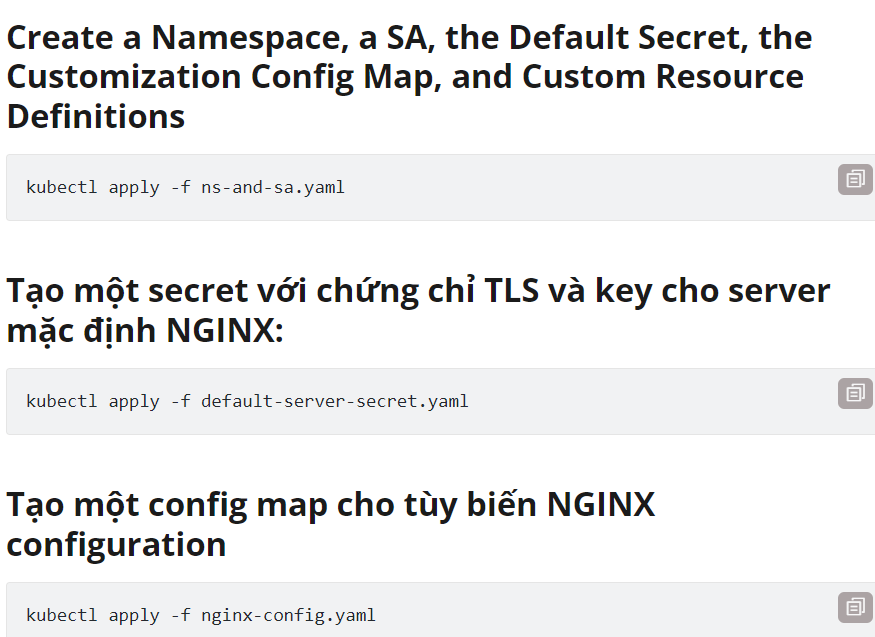
git clone https://github.com/sun-asterisk-research/air-viewer.git

**Install đối với các bạn chưa đọc bài trước**

Ở phần trước mình đã cài đặt kubernetes-ingress nên mình sẽ đi nhanh phần này nếu các bạn chưa cài đặt kubernetes-ingress. Từ giờ trở đi mặc định các câu lệnh thực thi trên master node thôi

# vào thư mục install

cd air-viewer/kubernetes/nginx-ingress/install



## Config Air-viewer

* Create Namespaces

kubectl create namespace air-viewer

* Deploy Mysql

Về deploy mysql chúng ta cần persistent Volume nơi chưa database, secret chứa các password của root và user name để che dấu password tránh người khác cùng quản lý các services khác đọc được thông qua câu lệnh describe deployment của mysql. Cuối cùng là các file chứa service và deployment

cd air-viewer/kubernetes/mysql

* Secret

echo -n "root" | base64

ouput: cm9vdA==

apiVersion: v1

kind: Secret

metadata:

name: mysql-secrets

namespace: air-viewer

type: Opaque

data:

ROOT\_PASSWORD: cm9vdA==

---

apiVersion: v1

kind: Secret

metadata:

name: mysql-pass-non-root

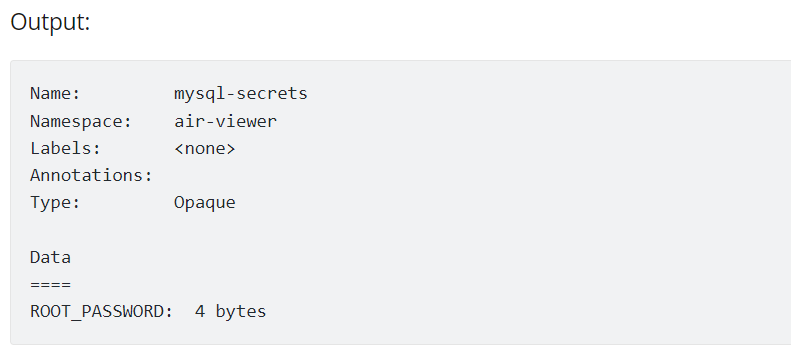
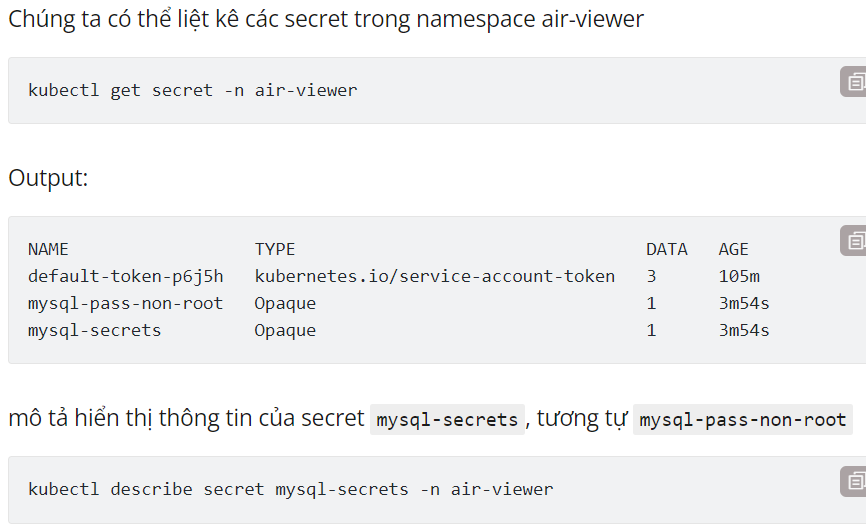
namespace: air-viewer

type: Opaque

data:

ROOT\_PASSWORD: ZXhhbXBsZQ==

kubectl apply -f secret.yaml



* Persistent Storage

Container là một cấu trúc ephemeral. Mọi thay đổi đối với container đang chạy sẽ bị mất dữ liệu khi container stop. Vì đó, container sẽ không phù hợp cho cơ sở dữ liệu lưu trữ, chúng ta bắt buộc phải sử dụng volume store để mount với các pod của mysql

# tạo kho 2Gi trống

apiVersion: v1

kind: PersistentVolume

metadata:

name: mysql-pv-volume

namespace: air-viewer

labels:

type: local

spec:

storageClassName: manual

capacity:

storage: 2Gi

accessModes:

- ReadWriteOnce

hostPath:

path: "/mnt/data"

---

# mysql sẽ sử dụng mysql-pv-claim thông qua deployment

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

name: mysql-pv-claim

namespace: air-viewer

spec:

storageClassName: manual

accessModes:

- ReadWriteOnce

resources:

requests:

storage: 2Gi

* Deployment

apiVersion: apps/v1

kind: Deployment

metadata:

name: mysql

labels:

app: mysql

namespace: air-viewer

spec:

replicas: 1

selector:

matchLabels:

app: mysql

strategy:

type: Recreate

template:

metadata:

labels:

app: mysql

spec:

containers:

- image: mysql:5.7

name: mysql

env:

- name: MYSQL\_ROOT\_PASSWORD

valueFrom:

secretKeyRef:

name: mysql-secrets

key: ROOT\_PASSWORD

- name: TZ

value: Asia/Ho\_Chi\_Minh

- name: MYSQL\_DATABASE

value: air\_viewer

- name: MYSQL\_USER

value: example

- name: MYSQL\_PASSWORD

valueFrom:

secretKeyRef:

name: mysql-pass-non-root

key: ROOT\_PASSWORD

ports:

- containerPort: 3306

name: mysql

volumeMounts:

- name: mysql-persistent-storage

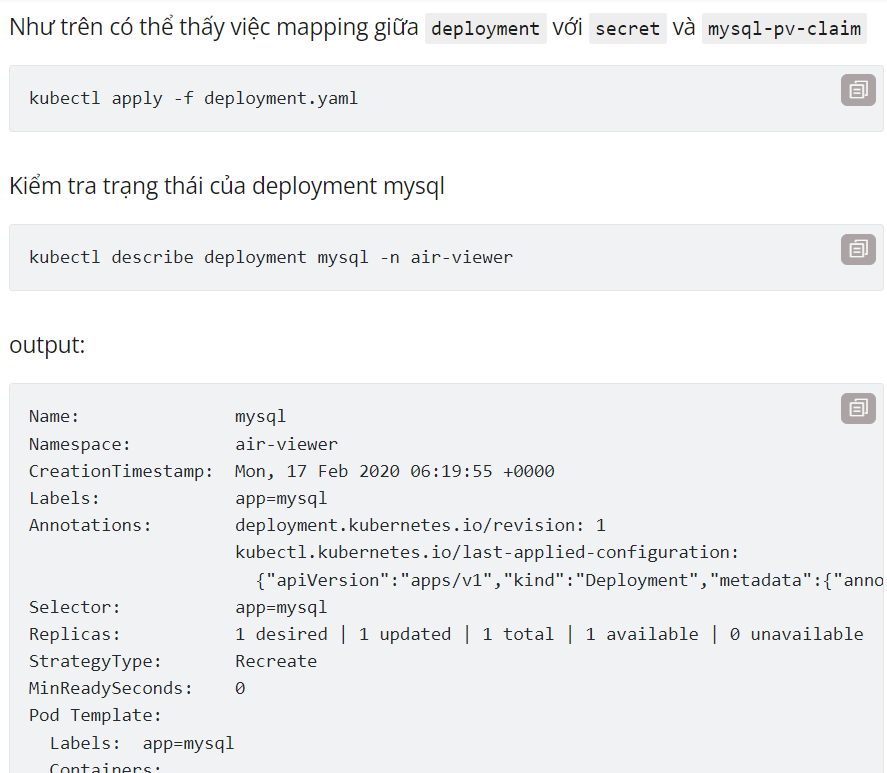
mountPath: /var/lib/mysql

volumes:

- name: mysql-persistent-storage

persistentVolumeClaim:

claimName: mysql-pv-claim

****  


### Service

Dựng service cho các pod(s)

apiVersion: v1

kind: Service

metadata:

name: mysql

namespace: air-viewer

labels:

app: mysql

spec:

type: ClusterIP

ports:

- port: 3306

selector:

app: mysql

## Deploy Backend (Flask, UWSGI)

### Deployment

cd air-viewer/kubernetes/backend

apiVersion: apps/v1

kind: Deployment

metadata:

name: backend-air-viewer

labels:

app: backend-air-viewer

namespace: air-viewer

spec:

replicas: 2

selector:

matchLabels:

app: backend-air-viewer

strategy:

type: Recreate

template:

metadata:

labels:

app: backend-air-viewer

spec:

containers:

- image: quanghung97/backend-air-viewer:v1

imagePullPolicy: Always

name: backend-air-viewer

env:

- name: DB\_HOST

value: mysql:3306

- name: DB\_DATABASE

value: air\_viewer

- name: DB\_USERNAME

value: example

- name: DB\_PASSWORD

valueFrom:

secretKeyRef:

name: mysql-pass-non-root

key: ROOT\_PASSWORD

ports:

- containerPort: 80

protocol: TCP

****

### Service

apiVersion: v1

kind: Service

metadata:

name: backend-air-viewer

namespace: air-viewer

labels:

app: backend-air-viewer

spec:

selector:

app: backend-air-viewer

ports:

- protocol: TCP

port: 80

targetPort: 80

## Deploy Frontend (Nuxt.js)

cd air-viewer/kubernetes/frontend

### Deployment

apiVersion: apps/v1

kind: Deployment

metadata:

name: frontend-air-viewer

labels:

app: frontend-air-viewer

namespace: air-viewer

spec:

replicas: 3

selector:

matchLabels:

app: frontend-air-viewer

strategy:

type: Recreate

template:

metadata:

labels:

app: frontend-air-viewer

spec:

containers:

- image: quanghung97/frontend-air-viewer:framgia2c

imagePullPolicy: Always

name: frontend-air-viewer

ports:

- containerPort: 3000

protocol: TCP

env:

- name: TZ

value: Asia/Ho\_Chi\_Minh

### Service

apiVersion: v1

kind: Service

metadata:

name: frontend-air-viewer

namespace: air-viewer

labels:

app: frontend-air-viewer

spec:

selector:

app: frontend-air-viewer

type: ClusterIP

ports:

- protocol: TCP

port: 80

targetPort: 3000

## Ingress controller config

Sơ sơ về VirtualServer: VirtualServer để định cấu hình cân bằng tải cho một tên miền

Ở phần này mình có chút yêu cầu về Virtual server route và virtual server tìm hiểu thêm [ở đây](https://docs.nginx.com/nginx-ingress-controller/configuration/virtualserver-and-virtualserverroute-resources/)

cd air-viewer/kubernetes/nginx-ingress

****

### Backend Virtual Server Route

Cấu hình route của backend với prefix /api

apiVersion: k8s.nginx.org/v1

kind: VirtualServerRoute

metadata:

name: backend-air-viewer

namespace: air-viewer

spec:

# change this to your DNS

host: framgia2c.mylabserver.com

upstreams:

- name: backend-air-viewer

service: backend-air-viewer

port: 80

subroutes:

- path: /api

action:

pass: backend-air-viewer

### Frontend Virtual Server Route

Cấu hình route của frontend với prefix /

apiVersion: k8s.nginx.org/v1

kind: VirtualServerRoute

metadata:

name: frontend-air-viewer

namespace: air-viewer

spec:

# change this to your DNS

host: framgia2c.mylabserver.com

upstreams:

- name: frontend-air-viewer

service: frontend-air-viewer

port: 80

subroutes:

- path: /

action:

pass: frontend-air-viewer

kubectl apply -f frontend-virtual-server-route.yaml

### Air-viewer Virtual Server

apiVersion: k8s.nginx.org/v1

kind: VirtualServer

metadata:

name: air-viewer

namespace: air-viewer

spec:

# change this to your DNS

host: framgia2c.mylabserver.com

tls:

secret: air-viewer-secret

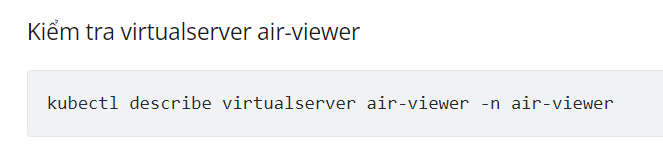
routes:

- path: /api

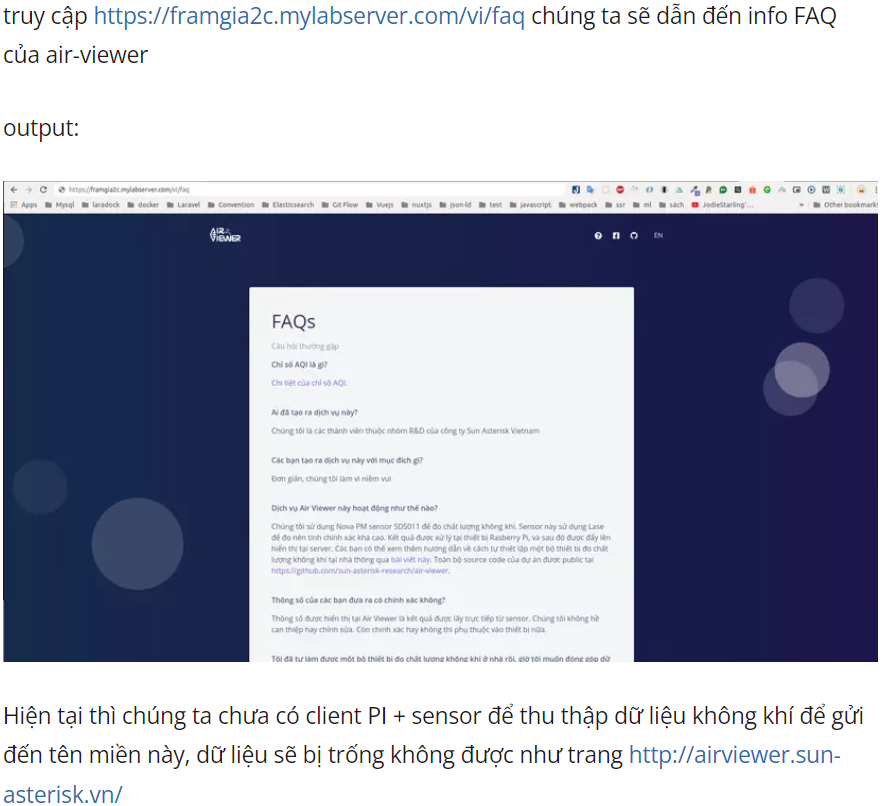
route: air-viewer/backend-air-viewer

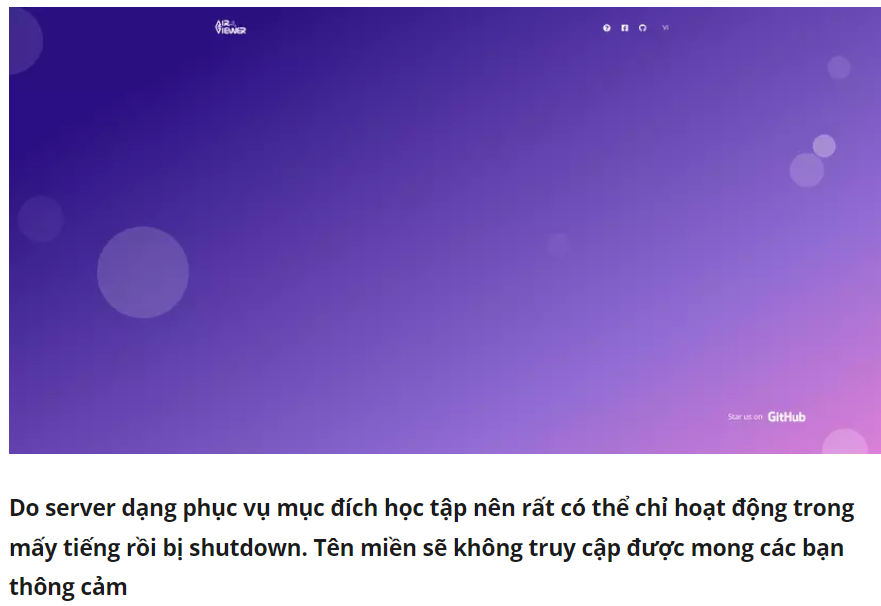
- path: /

route: air-viewer/frontend-air-viewer









## Một vài thí nghiệm nhỏ

### Thí nghiệm 1 (loại bỏ 1 worker node)

Chúng ta sẽ shut down worker node 2 có domain framgia3c.mylabserver.com tức ko ảnh hưởng đến domain hiện tại và cũng ko làm thay đổi master node. Nhận thấy rằng app vẫn chạy bình thường, kiểm tra các pods được dịch chuyển sang worker node 1 để gồng gánh các pods đã bị shutdown.

### Thí nghiệm 2(loại bỏ hẳn master node)

Chúng ta đang có 3 nodes, 1 master và 2 worker đang chạy và thử shutdown con master nodes. App vẫn chạy như bình thường, chỉ có chết 1 số service mà NuxtServerInit của Nuxt.js gọi API trên nền server kết nối với Flask sẽ bị chết, ngoài ra các services hay các API khác vẫn hoạt động như bình thường. Cũng giống như 1 tổ chức, leader nghỉ 1 hôm làm việc, member vẫn làm như bình thường theo lịch trình, nhưng chỉ cần thay đổi lịch trình, hay có vấn đề gì ở các member (shutdown tiếp cả worker node) lúc đó chắc chắn web của chúng ta sẽ bị chết. Do các pods sẽ không được điều phối nữa vì master node không còn quản lý hoạt động.

**Qua 2 thí nghiệm trên rút ra được zero downtime của kubernetes nó mạnh đến cỡ nào**

# Helm là gì ? Nó có liên quan gì đến series này ?

Với mục đích deploy air-viewer sử dụng helm, bài viết lần này chỉ giới thiệu về helm và deploy Nginx với Helm

## Lợi ích từ Helm

* Deployment đơn giản hơn mang tính chất lặp lại với chỉ vài câu lệnh ngắn
* Quản lý sự phụ thuộc của ứng dụng với các version cụ thể
* Thực hiện nhiều deployment với các môi trường khác nhau như: test, staging, production ...
* Thực thi các jobs liên quan đến chạy ứng dụng trước khi deployment
* Dễ dàng update rollback và test deployment khi có vấn đề xảy ra hay muốn cập nhật phiên bản mới (zero downtime server)

# Cài đặt Helm CLI

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

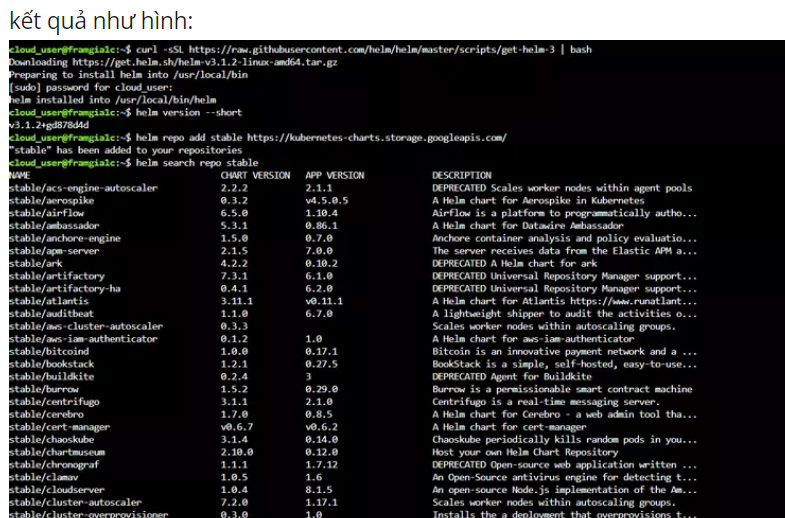
check version

helm version --short

Tải các kho lưu trữ Chart

helm repo add stable https://kubernetes-charts.storage.googleapis.com/

helm search repo stable



như các bạn thấy rất nhiều repo stable để cài đặt.

## Deploy Nginx với Helm

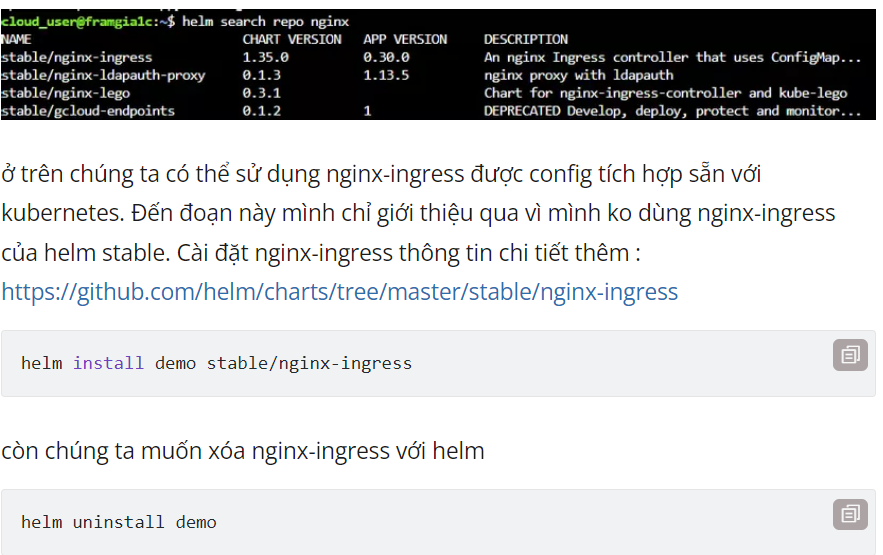
Cập nhật các repo

Để đảm bảo đã có đủ các repo stable có sẵn chúng ta sẽ chạy lệnh sau

helm repo update

## Tìm kiếm repo Nginx

helm search repo nginx



Tự tạo chart nginx cho riêng mình và deploy chúng

### create chart

chúng ta sẽ có 1 cấu trúc thư mục như sau

/example-project

/Chart.yaml # mô tả của chart

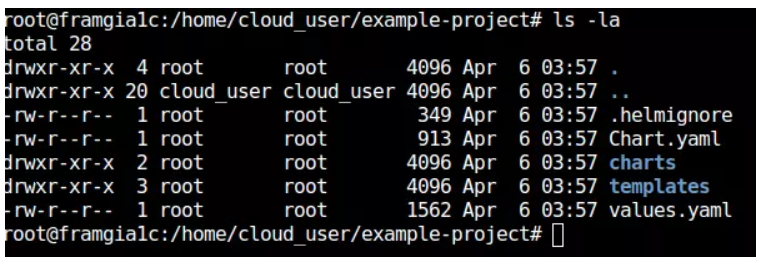
/values.yaml # các giá trị mặc định, chúng ta có thể thay đổi trong khi cài đặt hay nâng cấp project của chúng ta

/charts/ # subcharts

/templates/ # template file

Bằng việc chạy lệnh

helm create example-project



helm create example-project

### tùy chỉnh lại theo ý mình

trước khi tùy chỉnh chúng ta sẽ xem xét qua các nội dung của file ở phần templates chúng ta có các file như:

* deployment.yaml deployment với kubernetes
* \_helpers.tpl mẫu có sẵn có thể tái sử dụng trong chart
* ingress.yaml liệt kê triển khai các quyền để ứng dụng của bạn truy cập được kubernetes
* serviceaccount.yaml tạo tài khoản cho service mình dùng phần này có ví dụ cài đặt ingress-nginx ở 2 bài trước đó
* service.yaml triển khai các deployment với các service
* tests/ thư mục chứa các testing về chart

chúng ta sẽ tự tạo chart từ đầu và chúng ta sẽ xóa các mục sau:

rm -rf example-project/templates/

rm -rf example-project/Chart.yaml

rm -rf example-project/values.yaml

Tiếp theo chúng ta sẽ mô tả chart theo ý mình thích

cat <<EoF > example-project/Chart.yaml

apiVersion: v2

name: example-project

description: A Helm chart for Kubernetes Microservices application

version: 0.1.0

appVersion: 1.0

EoF

Sau đó, chúng ta sẽ tạo các deployment và service đơn giản bên trong templates

#tạo subfolders

mkdir -p example-project/templates/deployment

mkdir -p example-project/templates/service

Có 1 chút khác biệt so với deployment thông thường là có config {{ .Values.replicas }} hay {{ .Values.hello.image }}:{{ .Values.version }} được maping đến values.yaml sẽ khai báo sau



apiVersion: apps/v1

kind: Deployment

metadata:

name: coffee

spec:

replicas: {{ .Values.replicas }}

selector:

matchLabels:

app: coffee

template:

metadata:

labels:

app: coffee

spec:

containers:

- name: coffee

image: {{ .Values.hello.image }}:{{ .Values.version }}

ports:

- containerPort: 80



apiVersion: v1

kind: Service

metadata:

name: coffee-svc

spec:

ports:

- port: 80

targetPort: 80

protocol: TCP

name: http

selector:

app: coffee

Tạo values.yaml

cat <<EoF > example-project/values.yaml

# Default values for example-project.

# This is a YAML-formatted file.

# Declare variables to be passed into your templates.

# Release-wide Values

replicas: 3

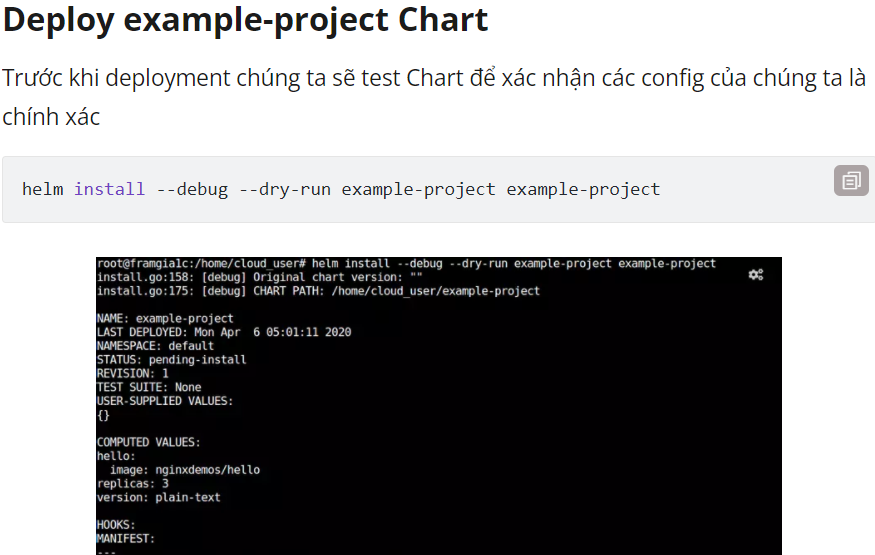
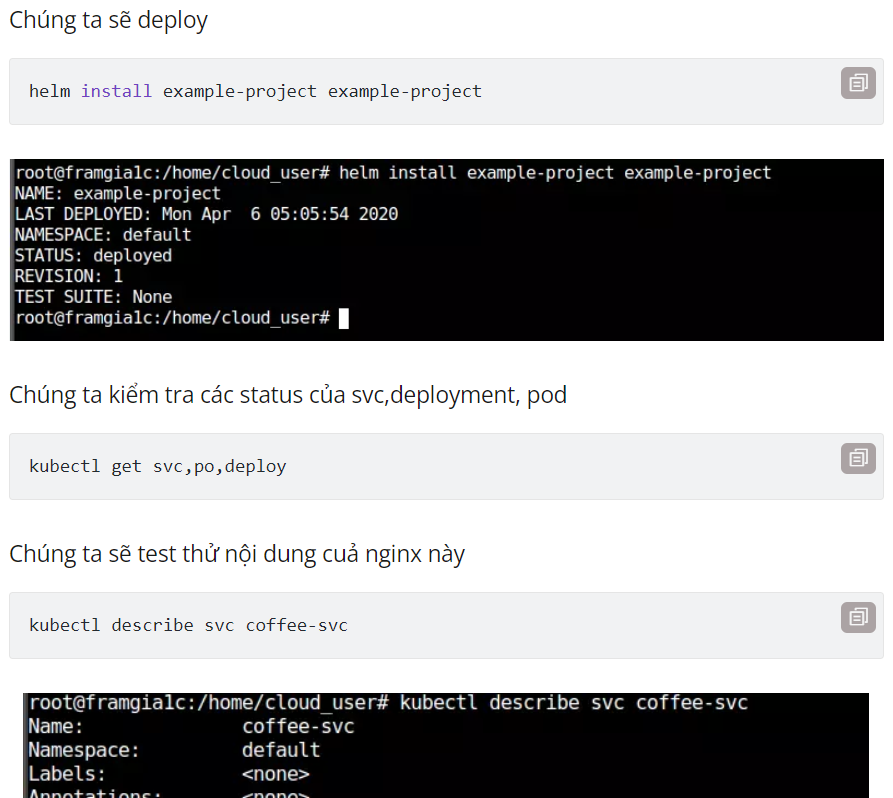
version: 'plain-text'

# Service Specific Values

hello:

image: nginxdemos/hello

EoF

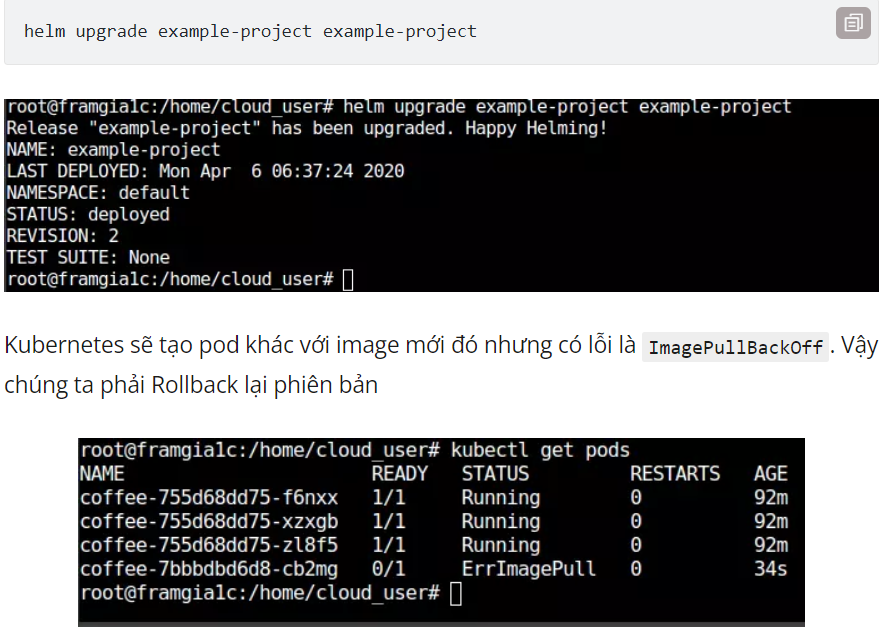
 

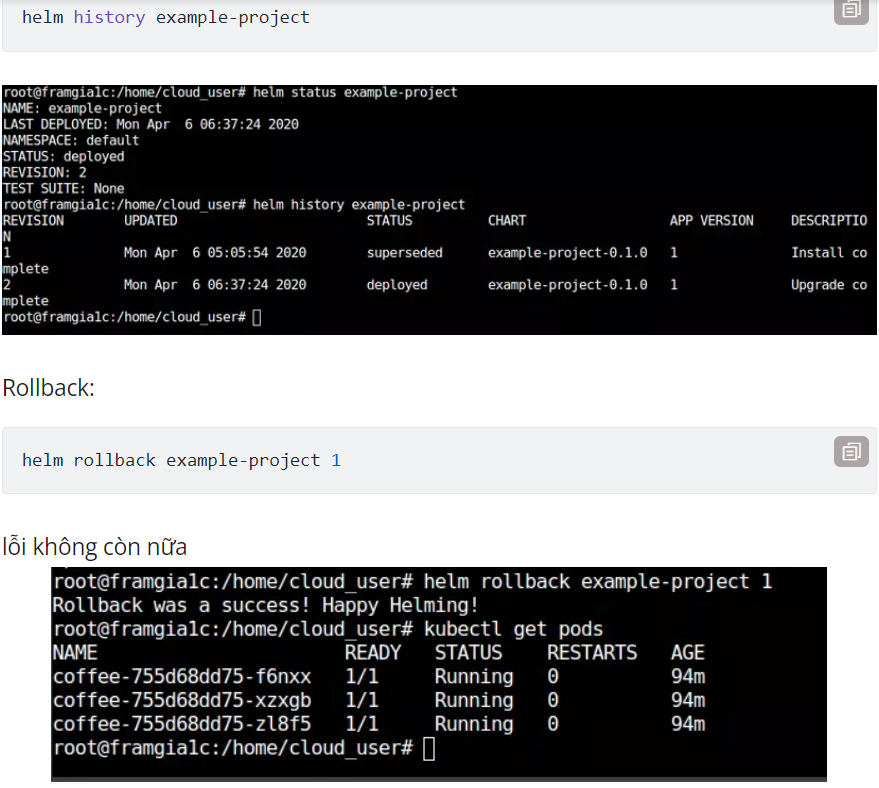
#chúng ta có 3 cái pods khác nhau, truy cập thử 1 cái

curl <Endpoint:>

## Lợi ích của việc sử dụng helm qua ví dụ minh họa

Giả sử chúng ta update example-project của chúng ta với image không tồn tại vì 1 lý do nào đó thành nginxdemos/hello-non-existing và cập nhật phiên bản mới của project của chúng ta





## Xóa example-project

Việc uninstall cũng nhanh không kém cạnh. Chúng ta không cần phải chạy 1 số câu lệnh lần lượt để delete svc, deployment, ingress, secret để xóa, mọi thứ chỉ 1 câu lệnh

helm uninstall example-project