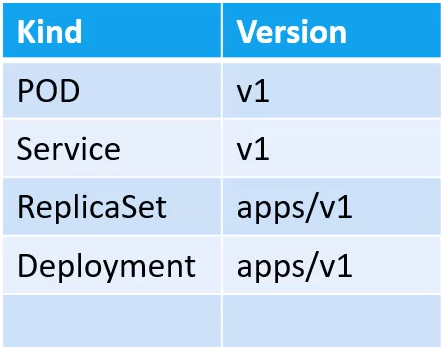


**1.apiVersion**

Vì mỗi kind phải có 1 apiVersion tương ứng và là khác nhau lên phải khai báo thêm apiVersion

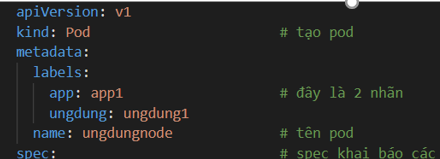


**2.Kind**

Pod, service, replicaSet, Deployment, ingress

**3.metadata**

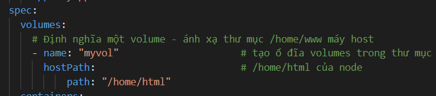
* name: cho từng loại kind
* labels: có thể có nhiều labels để map vs pod
* namespace: tên namespace



**4.spec**

**1.Volumes:**

* tên volume
* mount file vật lý máy local

****

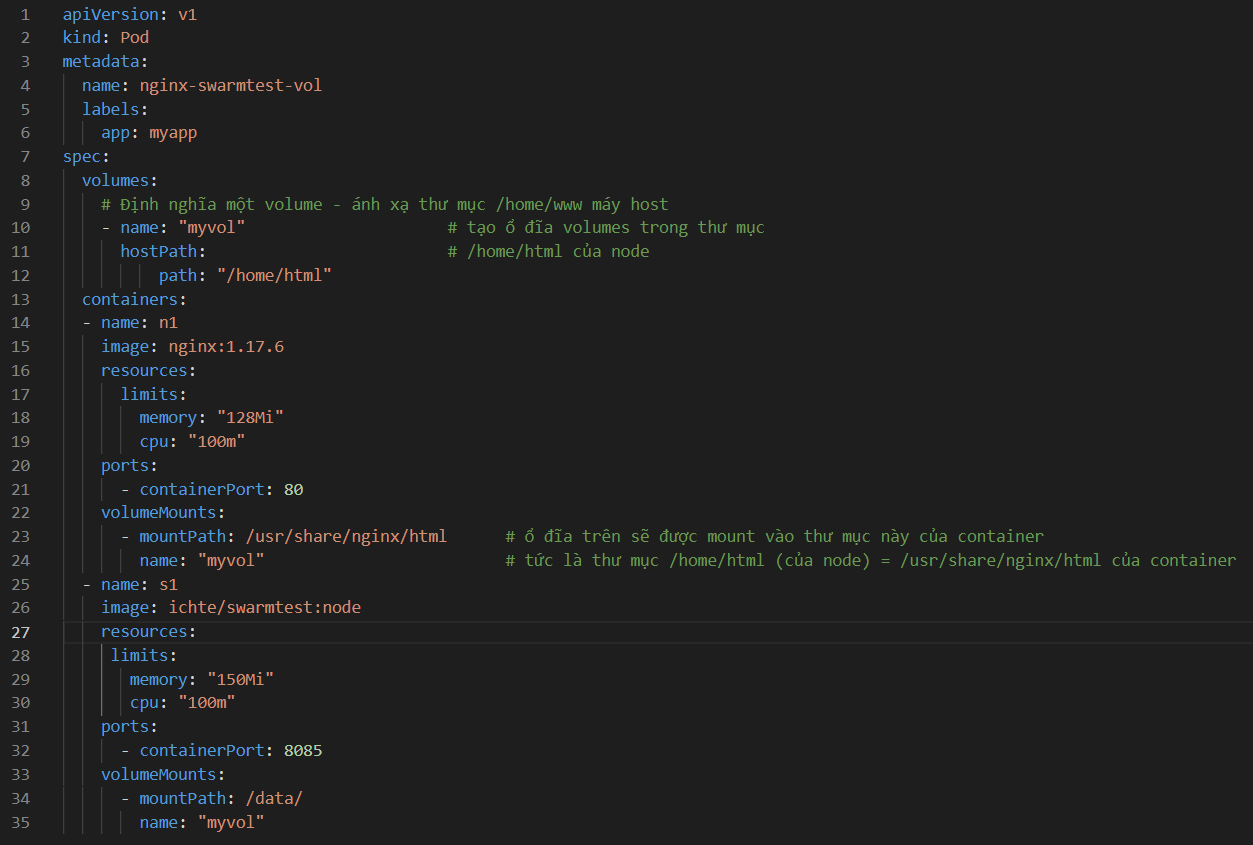
**2. container**

* container name
* image
* resources:
* limits: # giới hạn tài nguyên cho container

     memory: "128Mi"

    cpu: "100m"

* ports: # port container
* containerPort: 80
* volumeMounts:
  + name: "myvol" # sử dụng volume nào được khai báo tren local
* mountPath: /usr/share/nginx/html #mount vào container



**3.Replica**

* Số lượng replica pod

4. **selector:**

* Map với labels

**5.template**

* metadata
  + name của pod
  + labels
* spec
  + tạo ra container như trên (tức là nó sẽ tạo ra khi ko có sẵn pod )

**6.   nodeSelector:**

- phần này tạo ra để khi tạo ra pod chỉ định chạy trên node nào

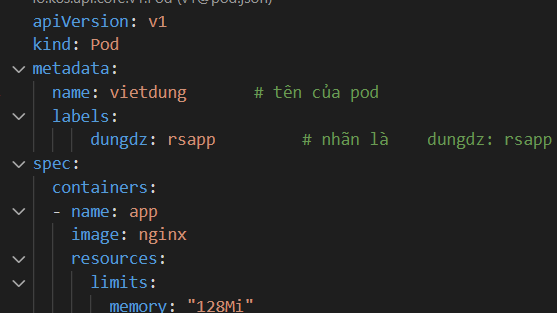
- thông tin nhãn để gán = lệnh kubectl describe no [nodename]

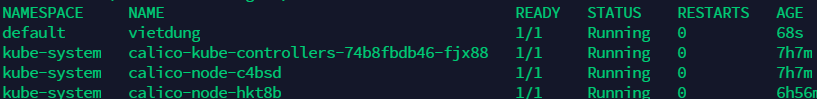
spec:

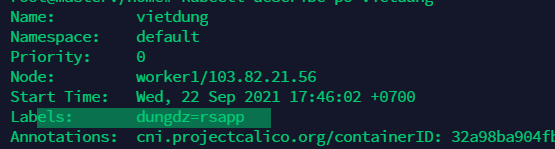
  nodeSelector:

    kubernetes.io/hostname: master

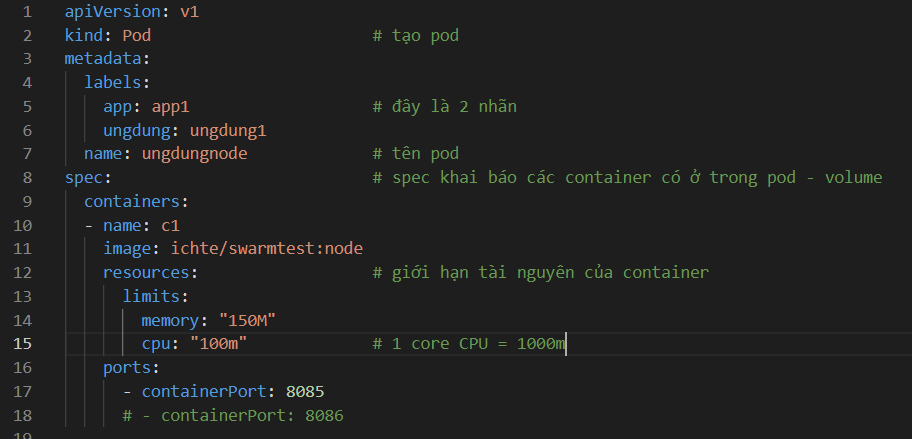
File yaml



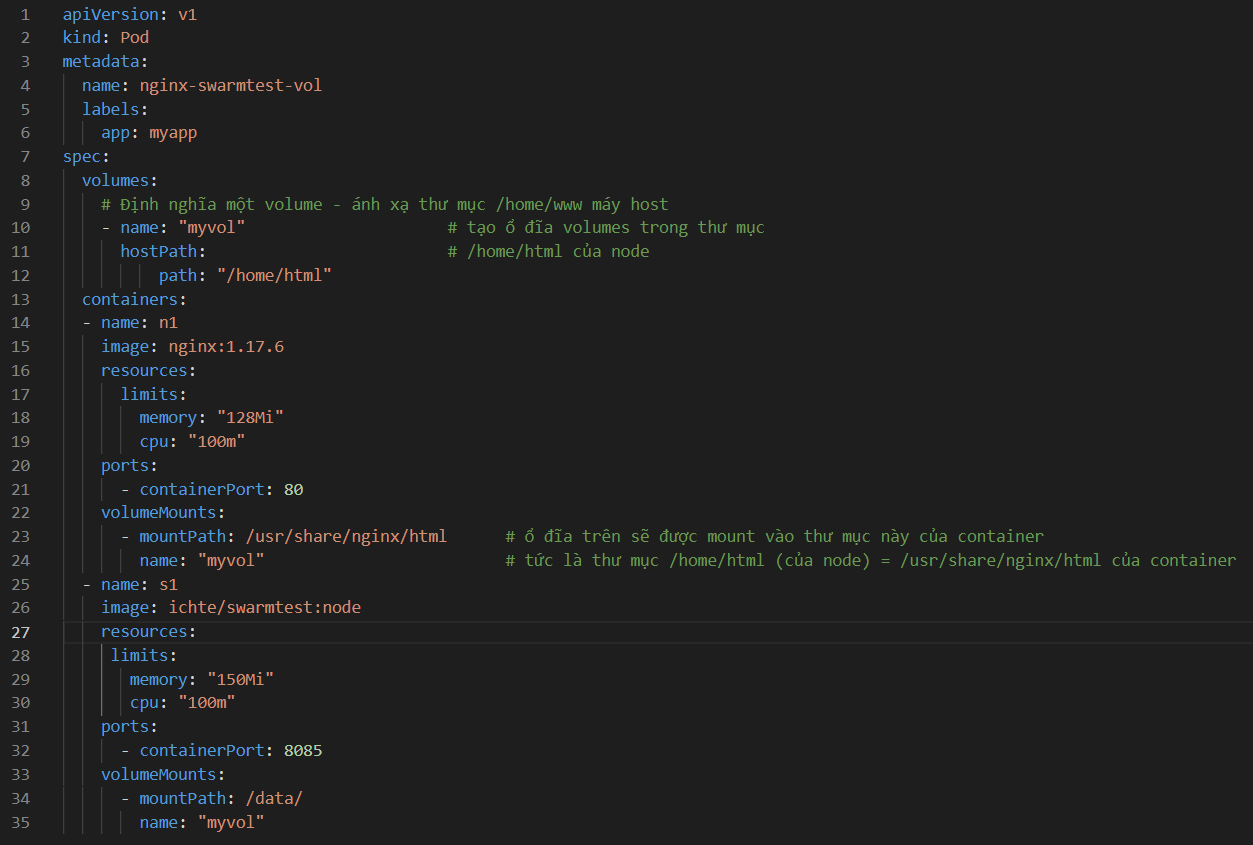




File pod

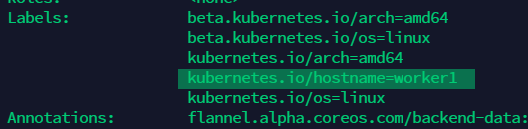


Tạo pod 2 container



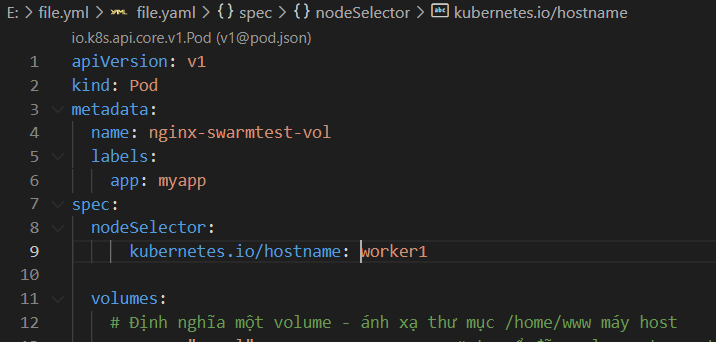
* ấn định 1 pod khi tạo ra chỉ chạy trên 1 node nào đó thôi

B1: kiểm tra cái label (nhãn nào đặc trưng cho node worker1)



Labels này mang giá trị của node worker1 và chỉ worker1 mới có nhãn này

Thêm vào file yaml của pod đó



**Replicaset**

apiVersion: apps/v1

kind: ReplicaSet

metadata:

  name: rsapp

spec:

  replicas: 3                         # số lượng replica pod tạo ra = 3

  selector:

    matchLabels:

      app: rsapp                      # quản lý các pod có nhãn là app: rsapp

  template:                           # tạo pod

    metadata:

      name: rsapp

      labels:

        app: rsapp

    spec:

      containers:

      - name: app

        image: ichte/swarmtest:node

        resources:

          limits:

            memory: "128Mi"

            cpu: "100m"

        ports:

          - containerPort: 8085

Phần **selector** là các nhãn của **pod** để **selector** chuyển hướng yêu cầu đến

Tức là tạo replicaset với 3 pod, trong đó tính cả những pod có labels là

**app: rsapp** và còn lại nếu thiếu n sẽ tự tạo ra pod và những pod đó được tạo từ template

Horizontal – Pod – Autoscaler (HPA)

apiVersion: autoscaling/v1

kind: HorizontalPodAutoscaler

metadata:

  name: rsapp-scaler

spec:

  scaleTargetRef:

    apiVersion: apps/v1

    kind: ReplicaSet

    name: rsapp

  minReplicas: 5

  maxReplicas: 10

  # Thực hiện scale CPU hoạt động ở 50% so với CPU mà POD yêu cầu

  targetCPUUtilizationPercentage: 50

file Deployment

apiVersion: apps/v1

kind: Deployment

metadata:

  # tên của deployment

  name: deployapp                 # tên của deployment

spec:

  # số POD tạo ra

  replicas: 3

  # thiết lập các POD do deploy quản lý, là POD có nhãn  "app=deployapp"

  selector:

    matchLabels:

      app: deployapp

  # Định nghĩa mẫu POD, khi cần Deploy sử dụng mẫu này để tạo Pod

  template:

    metadata:

      name: podapp

      labels:

        app: deployapp

    spec:

      containers:

      - name: node

        image: ichte/swarmtest:node

        resources:

          limits:

            memory: "128Mi"

            cpu: "100m"

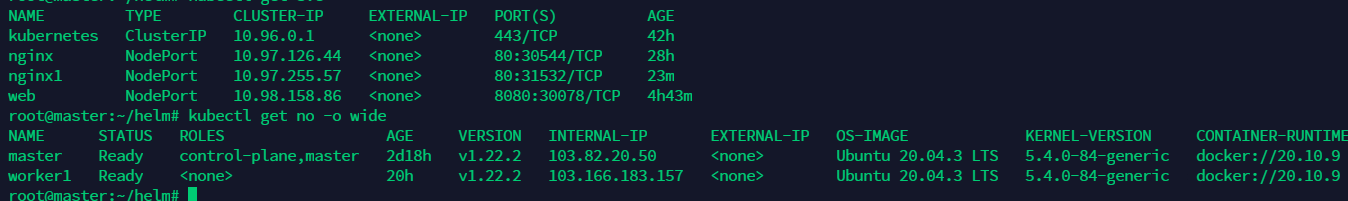
        ports:

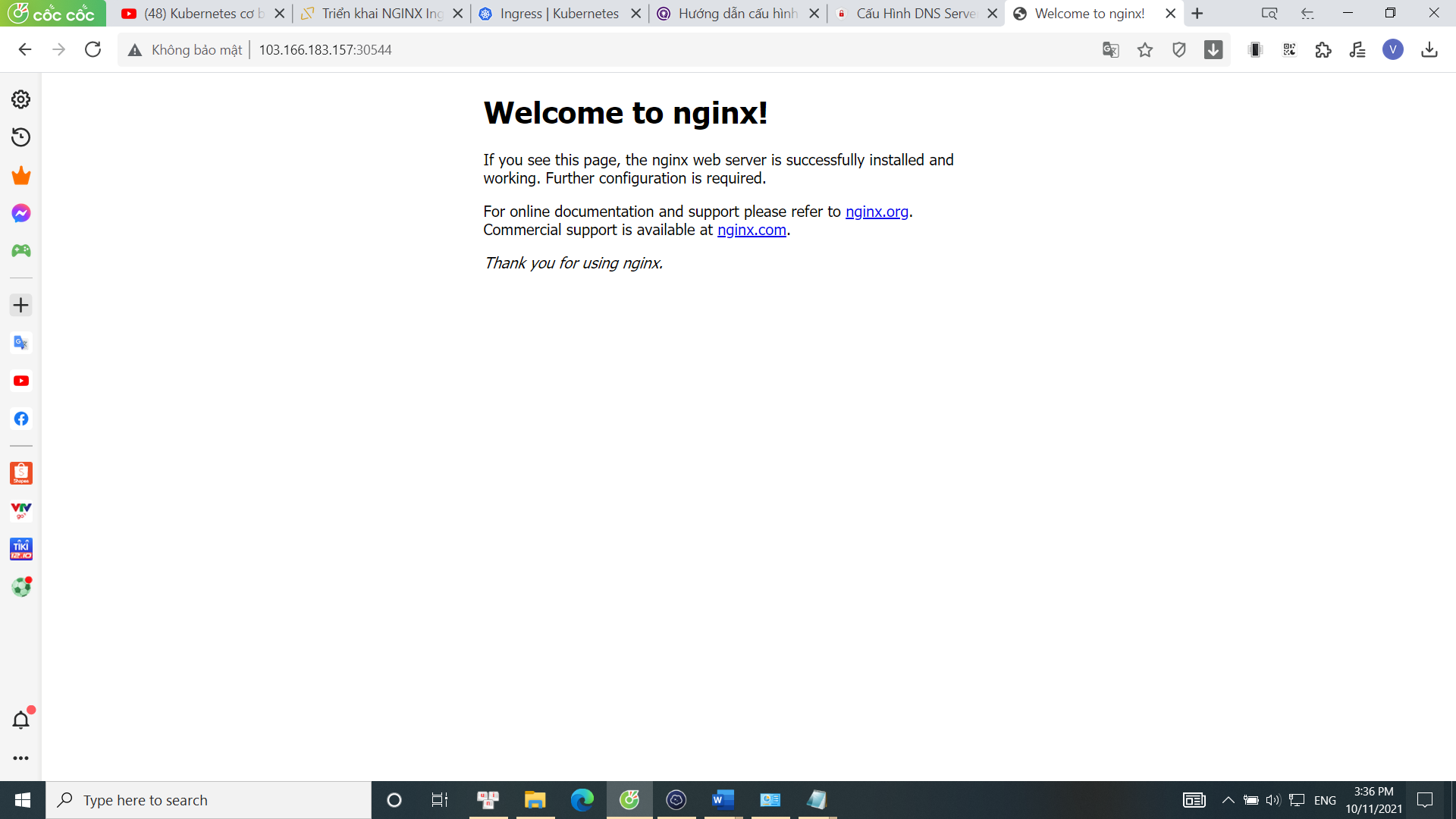
          - containerPort: 8085

File service.yaml

**Service**

Expose 1 trang web ra public có thể sử dụng service NodePort





apiVersion: v1

kind: Service

metadata:

  name: svc2

spec:

  selector:

     app: app1

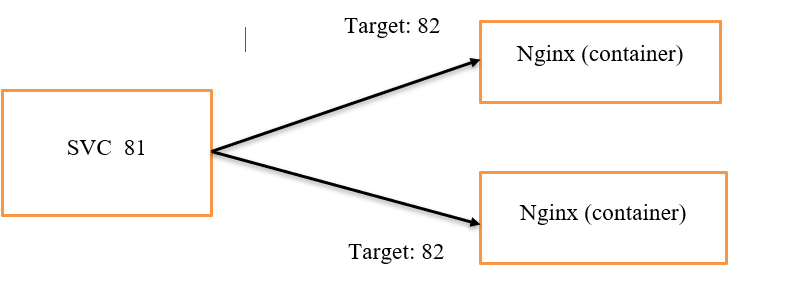
  type: ClusterIP

  ports:

    - name: port1

      port: 81        #IP port của service

      targetPort: 82  # expose port của container



TargetPort thì lúc này port container sẽ là 82, nhưng nginx chỉ hiểu port 80 nên port 82 n ko hiểu

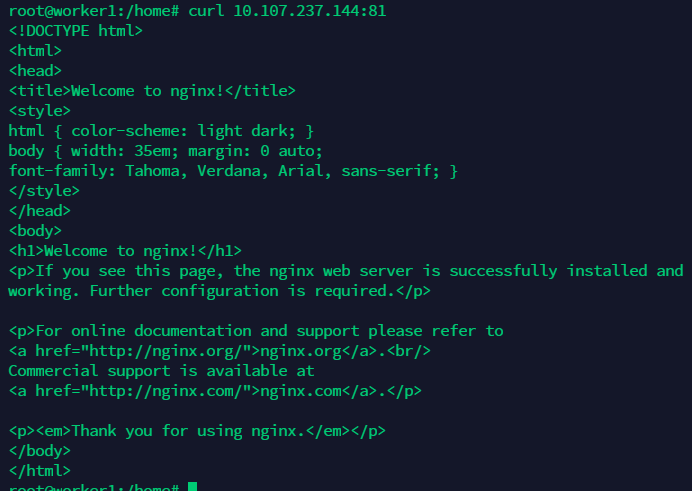
Giờ vào ip service port 81 là vào container nginx

Oke giờ chỉnh lại targetPort 80

# curl 10.107.237.144:81

Và ta truy cập service này bằng cluster IP

ở phần sau t sẽ truy cập bằng ip của nodePort bằng Type là NodePort



File pod nginx ở trên

apiVersion: v1

kind: Pod

metadata:

  name: myapp1

  labels:

    app: app1

spec:

  containers:

  - name: n1

    image: nginx

    resources:

      limits:

        memory: "128Mi"

        cpu: "100m"

    ports:

      - containerPort: 80  # cái này dell quan trọng

                           # chắc n chỉ là port để exec vào container

---

apiVersion: v1

kind: Pod

metadata:

  name: myapp2

  labels:

    app: app1

spec:

  containers:

  - name: n1

    image: httpd

    resources:

      limits:

        memory: "128Mi"

        cpu: "100m"

    ports:

      - containerPort: 80

Tạo thêm service NodePort

NodePort

apiVersion: v1

kind: Service

metadata:

  name: svc3

spec:

  selector:

     app: app1       # labels để map với 2 thằng container ở trên

  type: NodePort

  ports:

    - name: port1

      port: 80

      targetPort: 80

      nodePort: 31080

**Bài 11: ứng dụng Service, Deployment, Secret**

Phần này hay và quan trọng

**1) Chuẩn bị file cấu hình nginx.conf**

nginx.conf

error\_log /var/log/nginx/error.log warn;

pid /var/run/nginx.pid;

events {

worker\_connections 4096; ## Default: 1024

}

http {

include /etc/nginx/mime.types;

default\_type application/octet-stream;

log\_format main '$remote\_addr - $remote\_user [$time\_local] "$request" '

'$status $body\_bytes\_sent "$http\_referer" '

'"$http\_user\_agent" "$http\_x\_forwarded\_for"';

access\_log /var/log/nginx/access.log main;

sendfile on;

keepalive\_timeout 65;

#gzip on;

server {

listen 80;

server\_name localhost; # my-site.com

root **/usr/share/nginx/html**;

}

server {

listen 443 ssl;

server\_name localhost; # my-site.com;

ssl\_certificate **/certs/tls.crt**; # fullchain.pem

ssl\_certificate\_key **/certs/tls.key**; # privkey.pem

root **/usr/share/nginx/html**;

}

}

Chứng chỉ ssl ở thư mục **/certs/tls.crt**;

**2) Chuẩn bị file index.html**

index.html

<!DOCTYPE html>

<html>

<head><title>Nginx - Test!</title></head>

<body>

<h1>Chạy Nginx trên Kubernetes</h1>

</body>

</html>

**3) Xây dựng image mới**

Tạo Dockerfile xây dựng Image mới, từ image cơ sở nginx:1.17.6, có copy 2 file nginx.conf và index.html vào image mới này

Dockerfile

FROM nginx:1.17.6

COPY nginx.conf /etc/nginx/nginx.conf

COPY index.html /usr/share/nginx/html/index.html

4. nginx.yaml

apiVersion: apps/v1

kind: Deployment

metadata:

  name: nginx

spec:

  replicas: 1

  selector:

    matchLabels:

      app: nginx

  template:

    metadata:

      labels:

        app: nginx

    spec:

      volumes:                              # tạo ổ đĩa volume chứa file secret

        - name: cert-volume

          secret:                             # secret này được tạo ra có tên này trên kubernetes

             secretName: "secret-nginx-cert"  # map với secret vừa được tạo ra trên kubernetes

      containers:

      - name: nginx

        image: ichte/swarmtest:nginx

        imagePullPolicy: "Always"

        resources:

          limits:

            memory: "128Mi"

            cpu: "100m"

        ports:

        - containerPort: 80

        - containerPort: 443

        volumeMounts:

          - mountPath: "/certs"           # mount vào đường dẫn của container, đường dẫn cấu hình nginx nó đọc

            name: cert-volume             # tên volume muốn mount vào

---

apiVersion: v1

kind: Service

metadata:

  name: my-nginx

spec:

  type: NodePort

  ports:

  - port: 8080        # cổng dịch vụ ánh xạ vào cổng POD

    targetPort: 80    # cổng POD ánh xạ vào container

    protocol: TCP

    name: http

    nodePort: 31080   # cổng NODE ánh xạ vào cổng dịch vụ (chỉ chọn 30000-32767)

  - port: 443

    targetPort: 443

    protocol: TCP

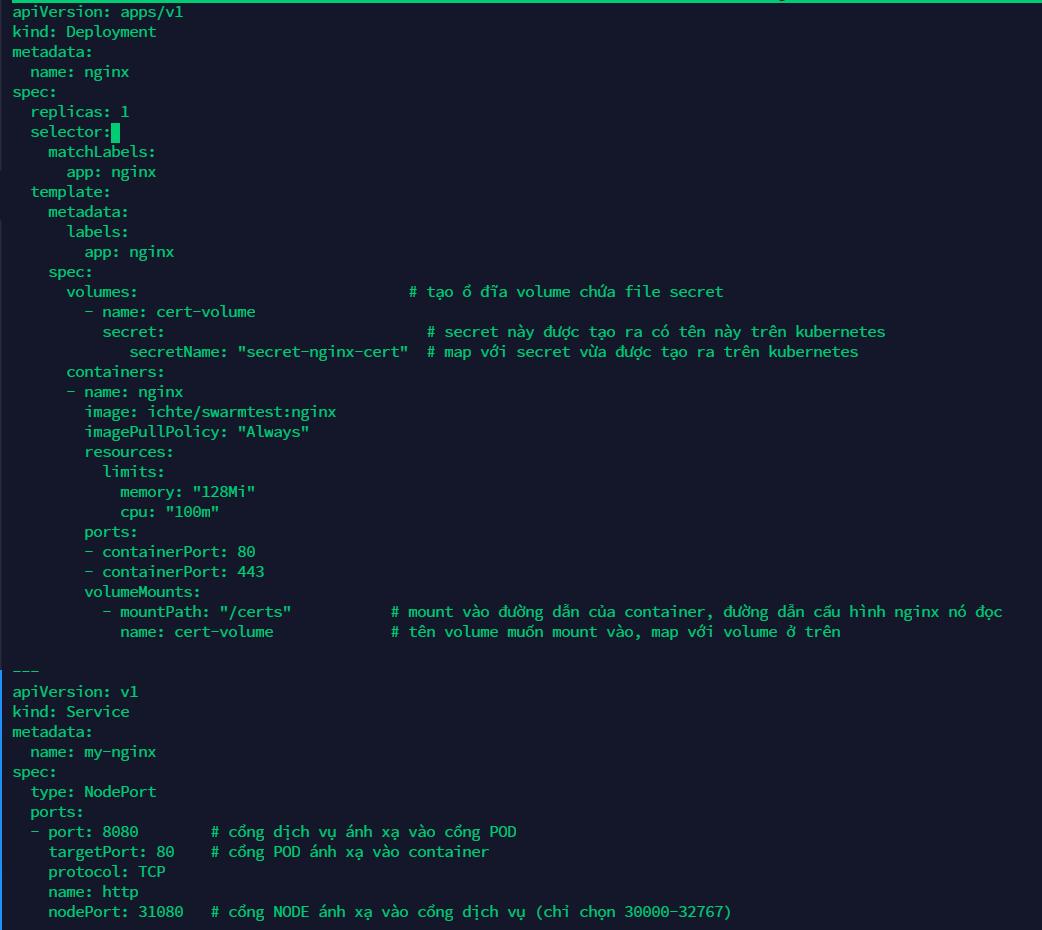
    name: https

    nodePort: 31443

  # Chú ý đúng với Label của POD tại Deployment

  selector:

    app: nginx

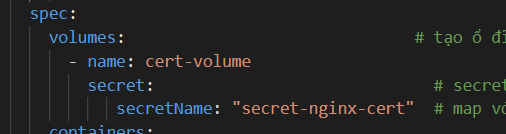


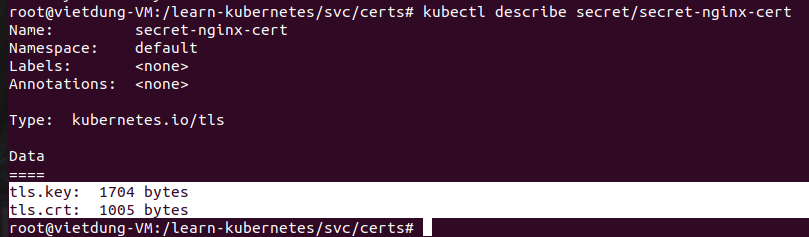
trong mục spec có 2 loại volume được tạo ra

* name volume

1. volume được tạo ra được lưu trên file vật lý của máy host gọi là **hostpath**
2. volume được tạo ra lưu trong file của container gọi là **mountPath** trong volumeMounts

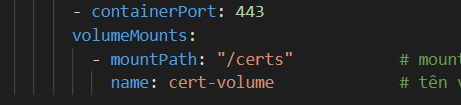
ở phần deploy nginx.yaml khai báo ở trên ta có thể thấy volume được tạo ra là

 volume này có dữ liệu của thằng secret trên hệ thống kubernetes chứa 2 file tls. Secret là thư mục lưu trữ để lưu file xác thực được ht kubernetes khuyến khích sử dụng



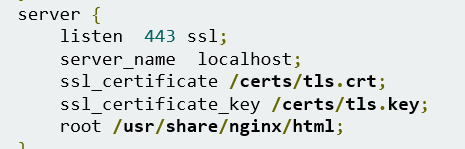
Trong secret của kubernetes có 2 file này

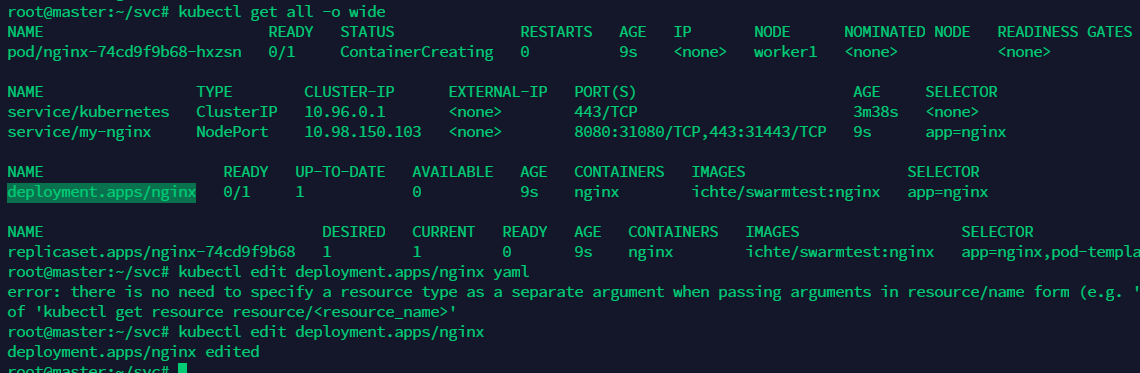
Sau khi có 2 file đó lưu trong volume, ta lại mount volume đó vào file /certs của thằng container bằng mountPath, lưu trong thư mục /certs của container



Trong thư mục /certs trên container lúc này đã có 2 file xác thực tls

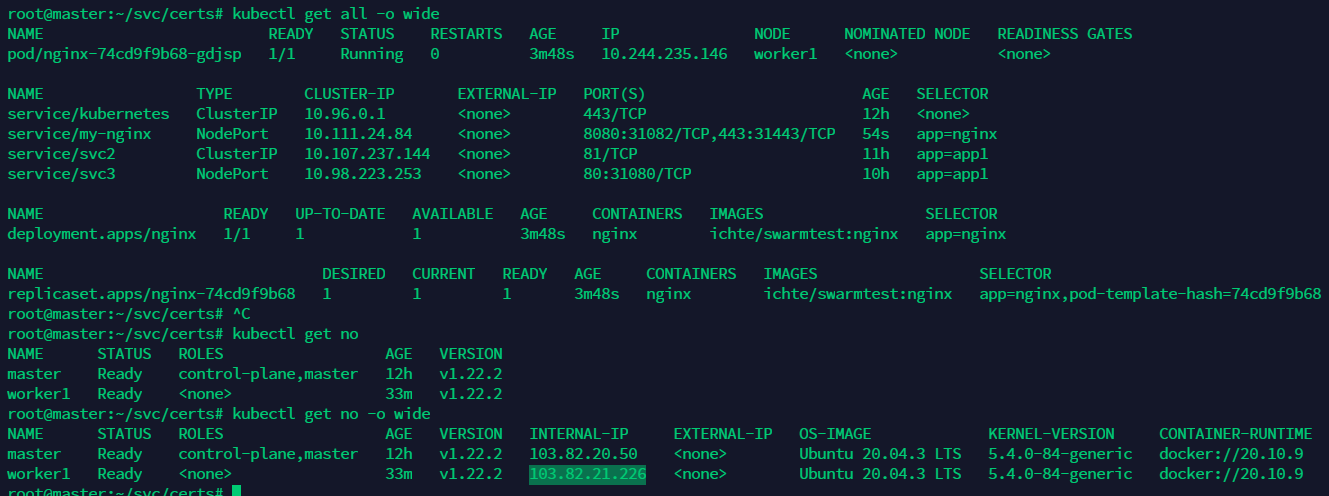
Và thư mục /certs này phải tạo đúng để thằng nginx thì trỏ đến file xác thực nó tìm thấy

 file deployment chạy trước, file nginx chạy sau



Để sửa nhanh thì **kubectl edit + NAME**

Giờ có thể truy cập từ địa chỉ IP của Node với cổng tương ứng (Kubernetes Docker thì http://localhost:31080 và <https://localhost:31443>)



8080, 443 là ip cluster

31082, 31443 là dùng ip worker node

**DaemonSet Job và CronJob trong Kubernetes**

1.DaemonSet

1.ds.yaml (DaemonSet)

apiVersion: apps/v1

kind: DaemonSet

metadata:

  name: dsapp

spec:

  selector:

    matchLabels:          # map với pod

      app: ds-nginx

  template:               # tạo pod nếu ko có sẵn pod, ko đủ pod

    metadata:

      labels:

        app: ds-nginx

    spec:

      containers:         # pod này chạy container nginx

      - name: nginx

        image: nginx

        resources:

          limits:

            memory: "128Mi"

            cpu: "100m"

        ports:

        - containerPort: 80

2. Job

02-job.yaml

# nhiệm vụ của job này là tạo ra 10 pod trong 120s chỉ để chạy ngày. nếu hthanh là thành công

apiVersion: batch/v1

kind: Job

metadata:

  name: myjob

spec:

  # Số lần chạy POD thành công

  completions: 10

  # Số lần tạo chạy lại POD bị lỗi, trước khi đánh dấu job thất bại

  backoffLimit: 3

  # Số POD chạy song song

  parallelism: 2

  # Số giây tối đa của JOB, quá thời hạn trên hệ thống ngắt JOB

  activeDeadlineSeconds: 120

  template:

    spec:

      containers:

      - name: busybox

        image: busybox

        command:

          - /bin/sh

          - -c

          - date; echo "Job executed"

      restartPolicy: Never

3.CronJob

3.cronjob.yaml

apiVersion: batch/v1beta1

kind: CronJob

metadata:

  name: mycronjob

spec:

  # Một phút chạy một Job

  schedule: "\*/1 \* \* \* \*"

  # Số Job lưu lại

  successfulJobsHistoryLimit: 3

  jobTemplate:

    spec:

      template:

        spec:

          containers:

          - name: busybox

            image: busybox

            args:

            - /bin/sh

            - -c

            - date; echo "Job in CronJob"

          restartPolicy: Never

**Persistent Volume – Persistent Volume C**

1.persistent-vol.yaml

apiVersion: v1

kind: PersistentVolume

metadata:

  name: pv1

  labels:

    name: pv1

spec:

  storageClassName: mystorageclass

  capacity:

    storage: 5Gi

  accessModes:

    - ReadWriteOnce

  hostPath:

    path: "/v1"

2.persistent-vol-claim.yaml

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

  name: pvc1

  labels:

    name: pvc1

spec:

  storageClassName: mystorageclass   # Map với storageName của PV

  accessModes:

    - ReadWriteOnce

  resources:

    requests:

      storage: 150Mi

Sử dụng PVC với Pod

apiVersion: apps/v1

kind: DaemonSet

metadata:

  name: myapp

spec:

  selector:

    matchLabels:

      name: myapp

  template:

    metadata:

      name: myapp

      labels:

        name: myapp

    spec:

      volumes:

      # Khai báo VL sử dụng PVC

      - name: myvolume

        persistentVolumeClaim:

          claimName: pvc1

      containers:

      - name: myapp

        image: busybox

        resources:

          limits:

            memory: "50Mi"

            cpu: "500m"

        command:

          - sleep

          - "600"

        volumeMounts:

        - mountPath: "/data"

          name: myvolume

accessModes:

* ReadWriteOnce -- the volume can be mounted as read-write by a single node
* ReadOnlyMany -- the volume can be mounted read-only by many nodes
* ReadWriteMany -- the volume can be mounted as read-write by many nodes
* ReadWriteOncePod -- the volume can be mounted as read-write by a single Pod. This is only supported for CSI volumes and Kubernetes version 1.22+.

In the CLI, the access modes are abbreviated to:

* RWO - ReadWriteOnce
* ROX - ReadOnlyMany
* RWX - ReadWriteMany
* RWOP - ReadWriteOncePod

## Tạo PersistentVolume NFS

## Tạo PersistentVolumeClaim NFS

Phần ingress, expose IP

<https://docs.microsoft.com/en-us/azure/aks/ingress-basic>

tạo ra các pod

apiVersion: v1

kind: Service

metadata:

  name: http-test-svc

  namespace: ingress-controller  #namespace của n là đây

spec:

  ports:

  - port: 80

    protocol: TCP

    targetPort: 80      # chuyển hướng đến cổng 80 của các pod

  selector:

    run: http-test-app

  sessionAffinity: None

  type: ClusterIP

---

apiVersion: apps/v1

kind: Deployment

metadata:

  labels:

    run: http-test-svc

  name: http-test-svc

  # Deploy này tạo tại namespace có Haproxy Ingress

  namespace: ingress-controller

spec:

  replicas: 2

  selector:

    matchLabels:

      run: http-test-app

  template:

    metadata:

      labels:

        run: http-test-app

    spec:

      containers:

      - image: nginx

        imagePullPolicy: IfNotPresent

        name: http

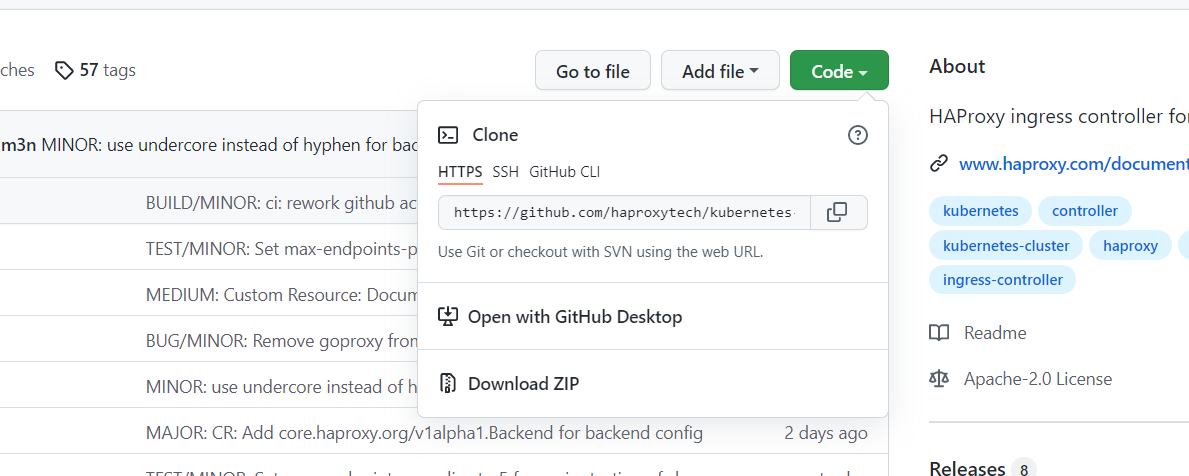
        ports:

        - containerPort: 80

          protocol: TCP

        resources: {}

bgio tạo ra Ingress để expose 1 IP public cho 2 pod trên



Để tải 1 cái trên git hub

* Vào phần deployment của n
* Copy file code kia
* git clone (+ file code vừa copy)

**service**

### **Type ExternalName**

**apiVersion**: v1

**kind**: Service

**metadata**:

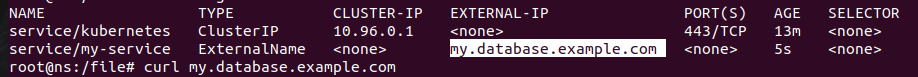
**name**: my-service

**namespace**: prod

**spec**:

**type**: ExternalName

**externalName**: my.database.example.com



Địa chỉ tên miền kia vẫn là local thôi, những cho những thành phần bên ngoài cluster đó truy cập được như máy thật window

Còn địa chỉ cluster-IP chỉ có những thành phần bên trong cluster đó truy cập được như node worker đó hay các pod chạy trong cluster đó

### **Type LoadBalancer**

**apiVersion**: v1

**kind**: Service

**metadata**:

**name**: my-service

**spec**:

**selector**:

**app**: MyApp

**ports**:

- **protocol**: TCP

**port**: 80

**targetPort**: 9376

**clusterIP**: 10.0.171.239

**type**: LoadBalancer

**status**:

**loadBalancer**:

**ingress**:

- **ip**: 192.0.2.127

### **External IPs**

**apiVersion**: v1

**kind**: Service

**metadata**:

**name**: my-service

**spec**:

**selector**:

**app**: MyApp

**ports**:

- **name**: http

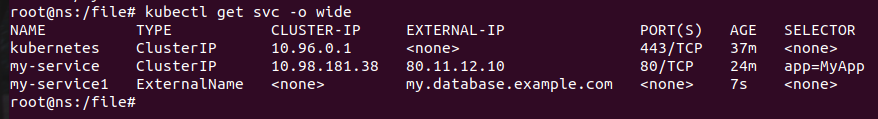
**protocol**: TCP

**port**: 80

**targetPort**: 9376

**externalIPs**:

- 80.11.12.10



**Tạo service expose tên miền**

1.Tạo 2pod

apiVersion: v1

kind: Pod

metadata:

  name: myapp1

  labels:

    app: MyApp          # nhãn cho svc quản lý

spec:

  containers:

  - name: n1

    image: nginx

    resources:

      limits:

        memory: "128Mi"

        cpu: "100m"

    ports:

      - containerPort: 80

---

apiVersion: v1

kind: Pod

metadata:

  name: myapp2

  labels:

    app: MyApp

spec:

  containers:

  - name: n1

    image: httpd

    resources:

      limits:

        memory: "128Mi"

        cpu: "100m"

    ports:

      - containerPort: 80

2.Tạo svc quản lý 2 pod và expose IP

apiVersion: v1

kind: Service

metadata:

  name: my-service

spec:

  selector:

    app: MyApp

  ports:

    - name: http

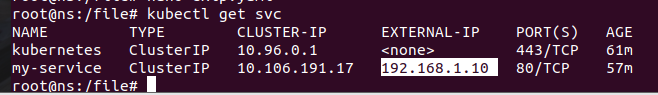
      protocol: TCP

      port: 80

      targetPort: 80

  externalIPs:

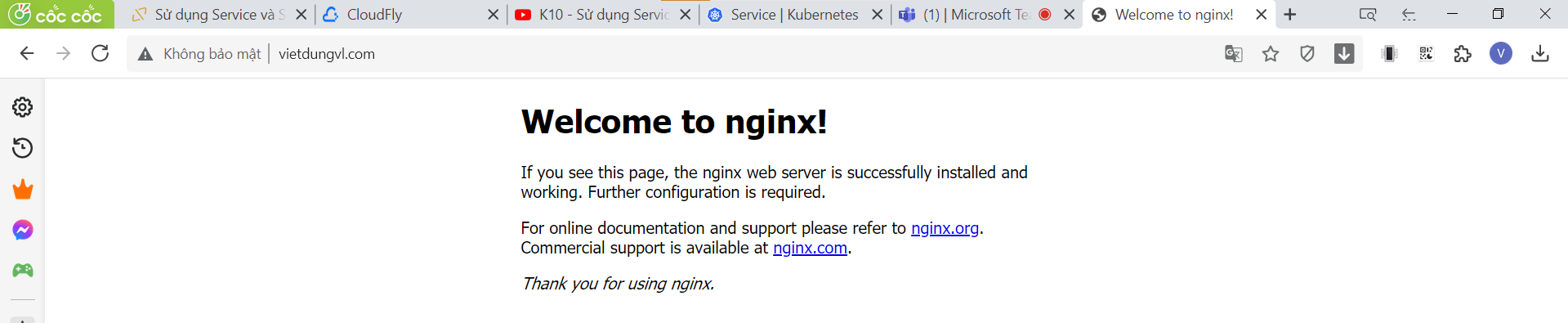
    - 192.168.1.10        # ip của worker node



Phần external-IP ta có thể đặt 1 IP khác được (1 IP ảo nào đấy) thì các thành phần trong cụm cluster đó có thể truy cập được.

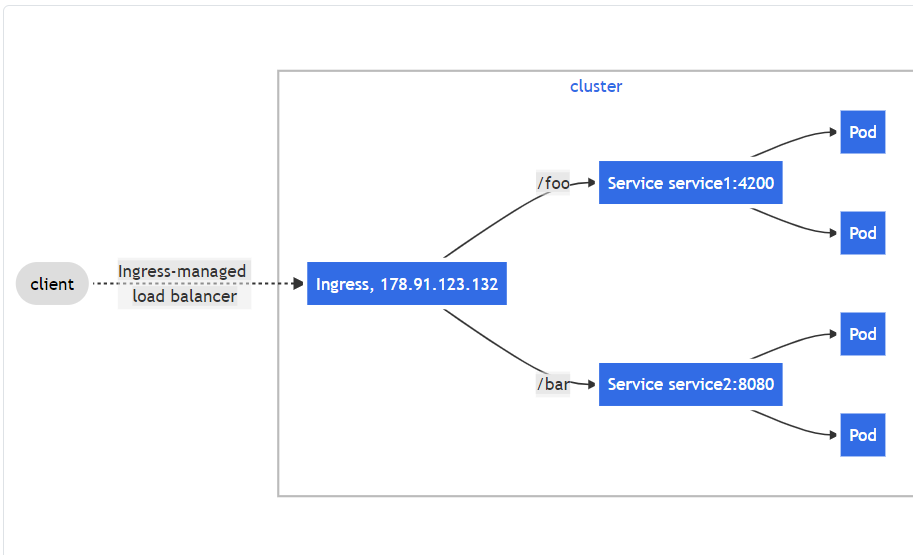
Nhưng thành phần bên ngoài thì ko truy cập được. phải external sang 1 IP worker

Vd 192.168.1.10 chính là địa chỉ ubuntu hay là IP worker này.

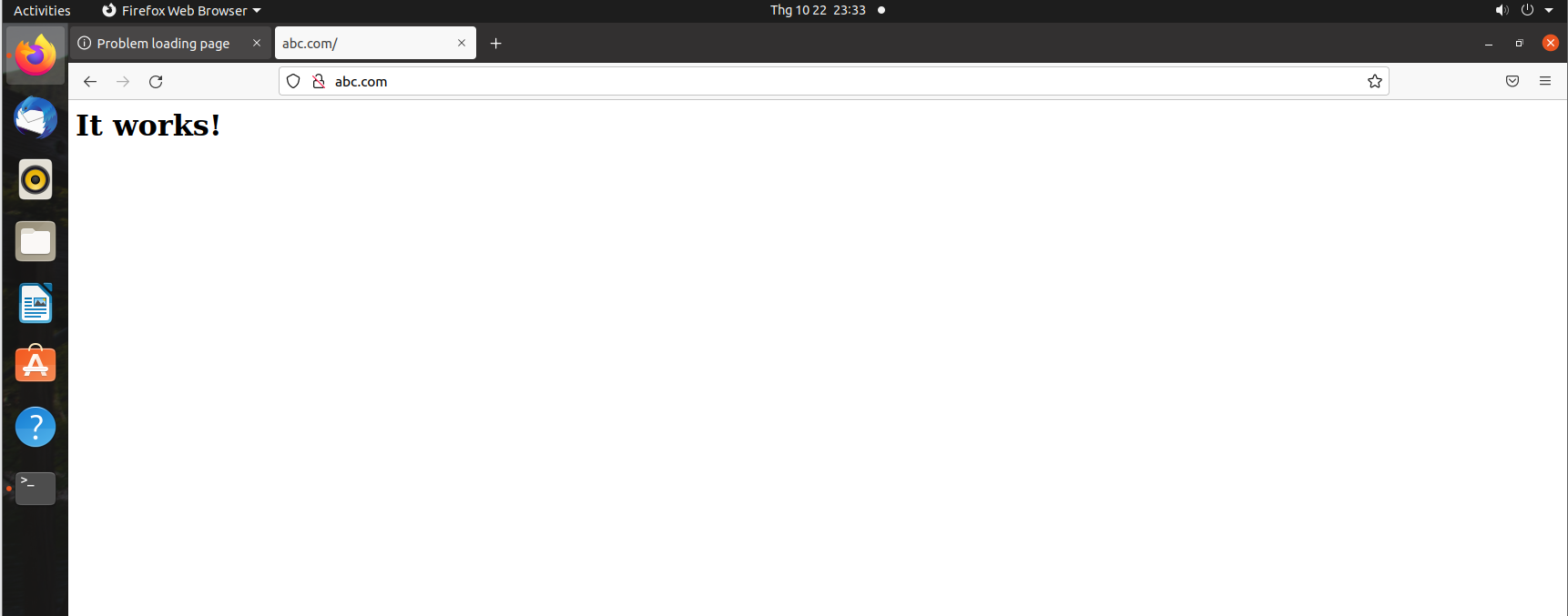


Ta đã vào được bằng ip external 192.168.1.10 hoặc địa chỉ tên miền **vietdungvl.com**

Vì ta đã cấu hình dns **vietdungvl.com = 192.168.1.10**



Vietdungvl.com



abc.com là địa chỉ của 1 ip của 1 con máy mà con máy này ko phải là 1worker node và ta vẫn vào được nhưng chỉ worker bên trong vào được

**Cấu hình ingress (Final)**

Cài ingress ở trang này

<https://kubernetes.github.io/ingress-nginx/deploy/>

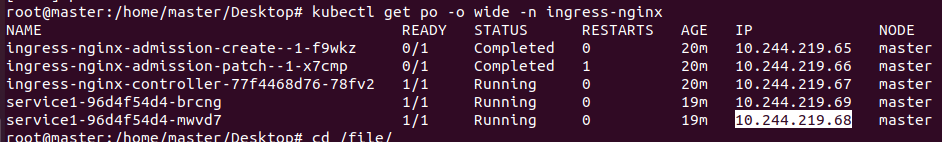
* lệnh cài ingress-controller

kubectl apply -f <https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v1.0.5/deploy/static/provider/cloud/deploy.yaml>

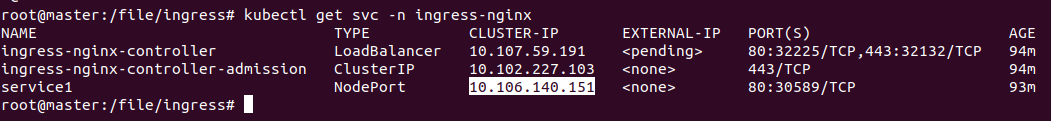
* file tạo ingress
* apiVersion: networking.k8s.io/v1
* kind: Ingress
* metadata:
* name: name-ingress
* namespace: ingress-nginx
* spec:
* rules:
* - host: first.bar.com
* http:
* paths:
* - pathType: Prefix
* path: "/"
* backend:
* service:
* name: service1
* port:
* number: 80
* File svc
* apiVersion: v1
* kind: Service
* metadata:
* name: service1
* namespace: ingress-nginx
* spec:
* ports:
* - port: 80
* protocol: TCP
* targetPort: 80
* selector:
* run: http-test-app
* sessionAffinity: None
* type: NodePort  # ClusterIP hay loại nào cũng dk
* ---
* apiVersion: apps/v1
* kind: Deployment
* metadata:
* labels:
* run: service1
* name: service1
* namespace: ingress-nginx
* spec:
* replicas: 2
* selector:
* matchLabels:
* run: http-test-app
* template:
* metadata:
* labels:
* run: http-test-app
* spec:
* containers:
* - image: nginx
* name: nginx
* ports:
* - containerPort: 80
* protocol: TCP
* resources: {}

thì trong file svc này ta có thể để chế độ nodeport hay clusterIP hay loại nào cũng dk

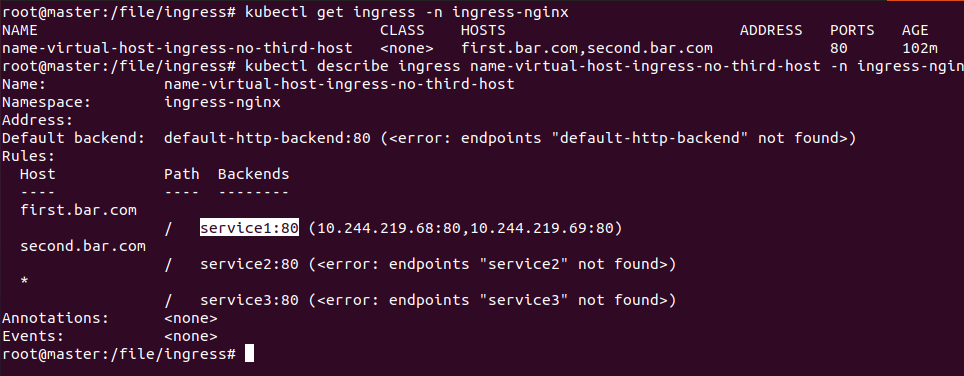
* Đây là các pod được tạo ra từ file svc ở trên trong namespace ingress-nginx



* Svc

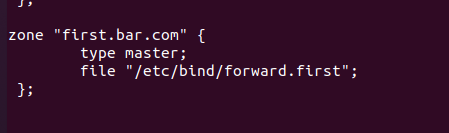


* Ingress



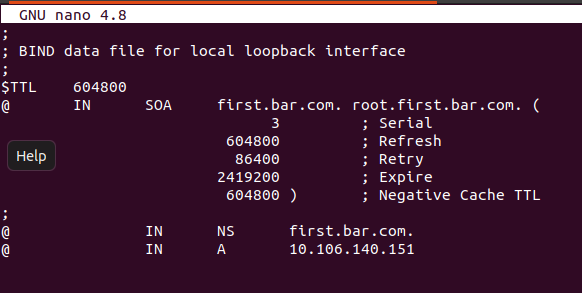
first.bar.com đã trỏ đến service1, IP phía sau là IP của pod

* Bgio cấu hình DNS để first.bar.com trỏ đến IP service1
* Tạo 1 zone



# nano forward.first

Cho first.bar.com trỏ đến IP của service là 10.106.140.151



# systemctl restart bind9

Kqua curl đén first.bar.com



Oke bgio chúng ta đã curl được rồi.

Vì là IP của service1 là IP ảo nên ta chỉ cơ đến được thôi, ko ping được đến tên miền first.bar.com

Bản chất ping là ta phải có kết nối đến IP đó, và IP đó tồn tại trên phần cứng

*# Tạo nhiều đối tượng YAML từ stdin*

cat <<EOF | kubectl apply -f -

apiVersion: v1

kind: Pod

metadata:

name: busybox-sleep

spec:

containers:

- name: busybox

image: busybox

args:

- sleep

- "1000000"

---

apiVersion: v1

kind: Pod

metadata:

name: busybox-sleep-less

spec:

containers:

- name: busybox

image: busybox

args:

- sleep

- "1000"

EOF

*# Tạo một secret với một số keys*

cat <<EOF | kubectl apply -f -

apiVersion: v1

kind: Secret

metadata:

name: mysecret

type: Opaque

data:

password: $(echo -n "s33msi4" | base64 -w0)

username: $(echo -n "jane" | base64 -w0)

EOF