Micro Services Application Deployment to Elastic Kubernetes Service

Table of Contents

I. Introduction

I.1. Target

The document aims at helping readers understand a few key features of **Kubernetes** such as“Pod”, “Deployment”, “StatefulSet”, “ConfigMap”, “Secret”, “Service”, “Ingress”, “StorageClass”, “PersistentVolume”, “PersistentVolumeClaim”, etc.

I.2. Demo Overview

The application will be deployed to AWS Elastic Kubernetes Service known as EKS.

The project used in this demo consists of 3 services:

- database
- backend
- frontend

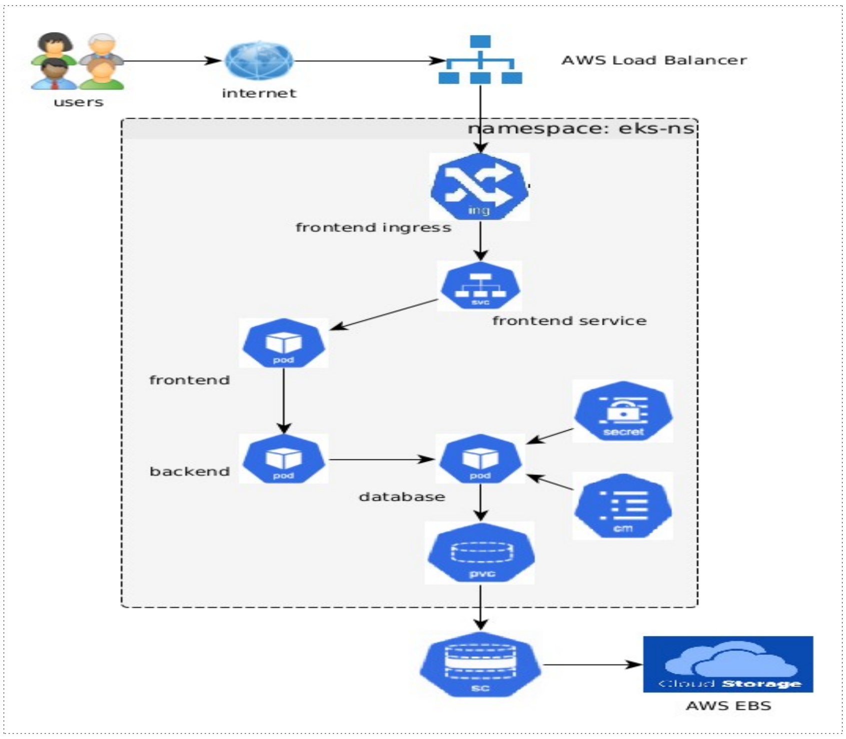


Figure 1: Application Components

There are 4 manifest yaml files for this project. They can be found in “Demo source code”

- mongodb.yaml
- backend.yaml
- frontend.yaml
- ingress.yml

Note: For the production environment, the database should be an external database service.

I.3. Prequiresites

- EKS CLI is installed (refer to Install EKS CLI)
- EKS cluster is already created (refer to Create EKS Cluster).
- EBS CSI driver is setup on the EKS cluster. See “Setup EBS CSI driver” for more details.
- Kubernetes CLI gets configured (kubectl) (refer Configure kubectl)

II. Instruction

II.1. General steps

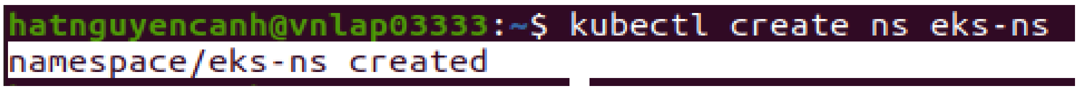
In order to get the application running, the following steps are required:

- Create a new namespace
- Create a new database
- Create backend and frontend applications

II.2. Execution

- <u>**Create a namespace:**</u>

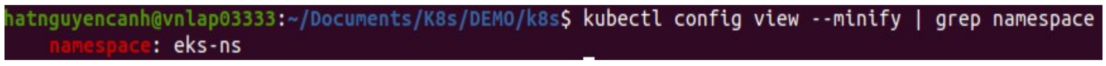
# create the a new namspace called "eks-ns" if it is not yet present:
kubectl create ns eks-ns



```
hatnguyencanh@vnlap03333:~$ kubectl create ns eks-ns
namespace/eks-ns created
```

# Change default working namespace to that one
kubectl config set-context --current --namespace eks-ns

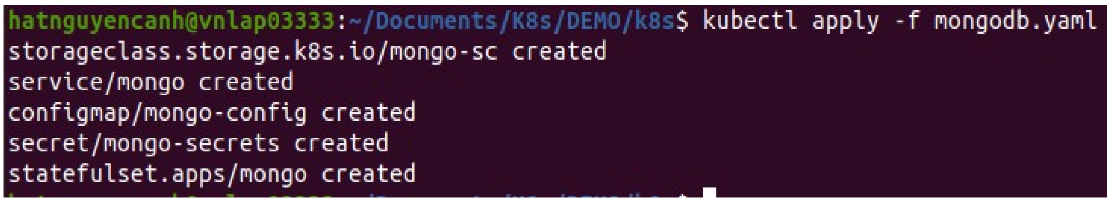
# verify if you've set to this namespace:
kubectl config view --minify | grep namespace



```
hatnguyencanh@vnlap03333:~/Documents/K8s/DEMO/k8s$ kubectl config view --minify | grep namespace
    namespace: eks-ns
```

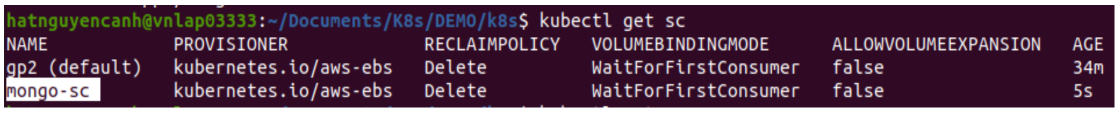
- <u>**Create a database**</u>

# create storageclass, pvc and database
kubectl apply -f mongodb.yaml



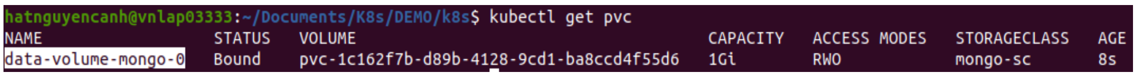
```
hatnguyencanh@vnlap03333:~/Documents/K8s/DEMO/k8s$ kubectl apply -f mongodb.yaml
storageclass.storage.k8s.io/mongo-sc created
service/mongo created
configmap/mongo-config created
secret/mongo-secrets created
statefulset.apps/mongo created
```

# verify storage class
kubectl get sc



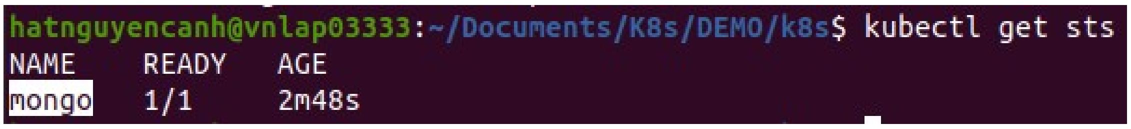
```
hatnguyencanh@vnlap03333:~/Documents/K8s/DEMO/k8s$ kubectl get sc
NAME            PROVISIONER             RECLAIMPOLICY   VOLUMEBINDINGMODE      ALLOWVOLUMEEXPANSION   AGE
gp2 (default)   kubernetes.io/aws-ebs   Delete          WaitForFirstConsumer   false                  34m
mongo-sc        kubernetes.io/aws-ebs   Delete          WaitForFirstConsumer   false                  5s
```

# verify pvc
kubectl get pvc



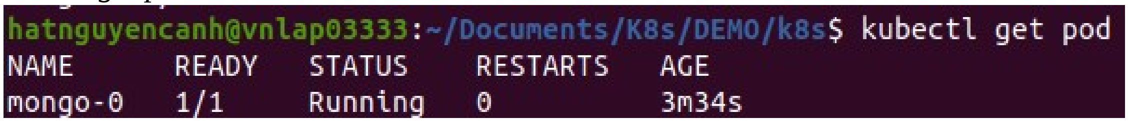
```
hatnguyencanh@vnlap03333:~/Documents/K8s/DEMO/k8s$ kubectl get pvc
NAME                  STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS   AGE
data-volume-mongo-0   Bound    pvc-1c162f7b-d89b-4128-9cd1-ba8ccd4f55d6   1Gi        RWO            mongo-sc       8s
```

# verify StatefulSet
kubectl get sts



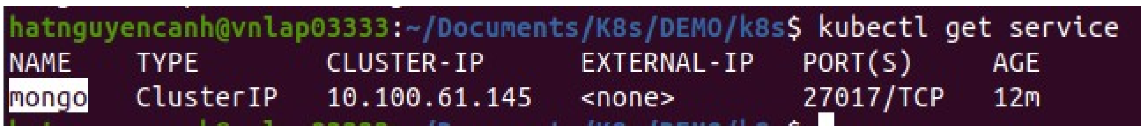
```
hatnguyencanh@vnlap03333:~/Documents/K8s/DEMO/k8s$ kubectl get sts
NAME    READY   AGE
mongo   1/1     2m48s
```

# verify database pods
kubectl get pod



```
hatnguyencanh@vnlap03333:~/Documents/K8s/DEMO/k8s$ kubectl get pod
NAME      READY   STATUS    RESTARTS   AGE
mongo-0   1/1     Running   0          3m34s
```

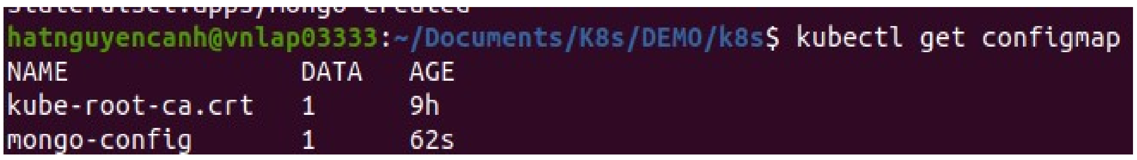
# verify database service
kubectl get service



```
hatnguyencanh@vnlap03333:~/Documents/K8s/DEMO/k8s$ kubectl get service
NAME    TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)     AGE
mongo   ClusterIP   10.100.61.145   <none>        27017/TCP   12m
```

# verify configmap

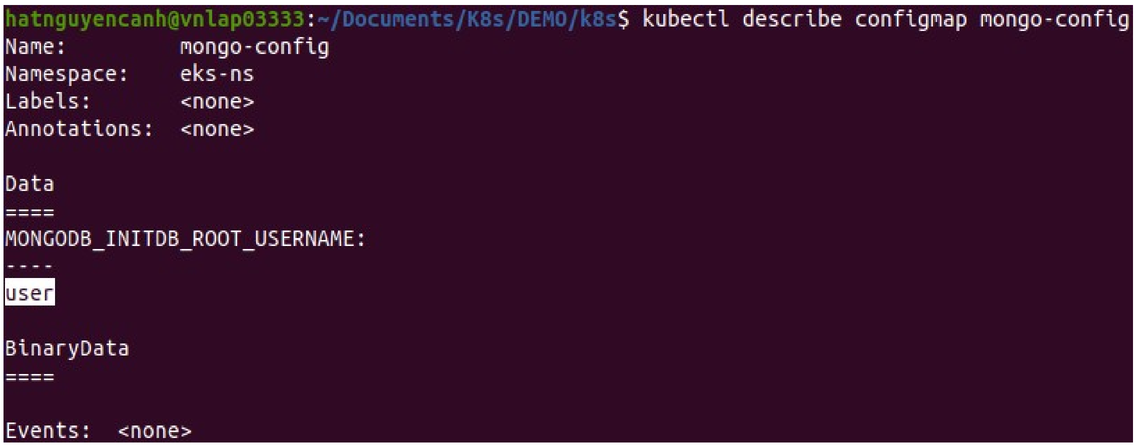
kubectl get configmap



```
hatnguyencanh@vnlap03333:~/Documents/K8s/DEMO/k8s$ kubectl get configmap
NAME               DATA   AGE
kube-root-ca.crt   1      9h
mongo-config       1      62s
```

# check configmap details

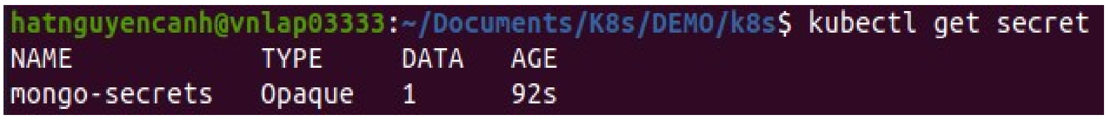
kubectl describe configmap mongo-config



```
hatnguyencanh@vnlap03333:~/Documents/K8s/DEMO/k8s$ kubectl describe configmap mongo-config
Name:         mongo-config
Namespace:    eks-ns
Labels:       <none>
Annotations:  <none>

Data
====
MONGODB_INITDB_ROOT_USERNAME:
----
user

BinaryData
====

Events:  <none>
```

# verify secret

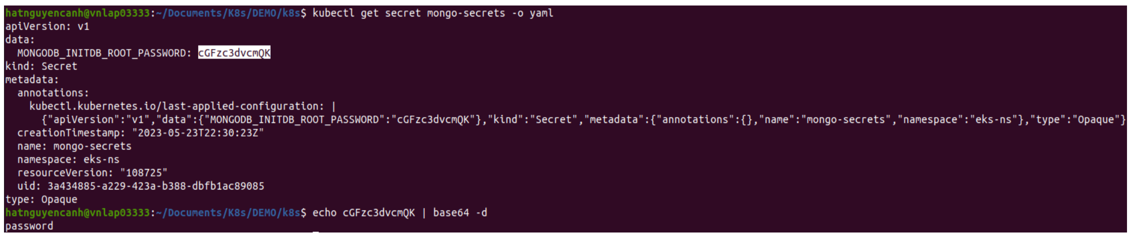
kubectl get secret



```
hatnguyencanh@vnlap03333:~/Documents/K8s/DEMO/k8s$ kubectl get secret
NAME            TYPE     DATA   AGE
mongo-secrets   Opaque   1      92s
```

# check secret details

kubectl get secret mongo-secrets -o yaml



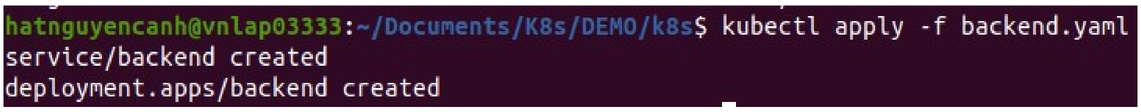
```
hatnguyencanh@vnlap03333:~/Documents/K8s/DEMO/k8s$ kubectl get secret mongo-secrets -o yaml
apiVersion: v1
data:
  MONGODB_INITDB_ROOT_PASSWORD: cGFzc3dvcmQK
kind: Secret
metadata:
  annotations:
    kubectl.kubernetes.io/last-applied-configuration: |
      {"apiVersion":"v1","data":{"MONGODB_INITDB_ROOT_PASSWORD":"cGFzc3dvcmQK"},"kind":"Secret","metadata":{"annotations":{},"name":"mongo-secrets","namespace":"eks-ns"},"type":"Opaque"}
  creationTimestamp: "2023-05-23T22:30:23Z"
  name: mongo-secrets
  namespace: eks-ns
  resourceVersion: "108725"
  uid: 3a434885-a229-423a-b388-dbfb1ac89085
type: Opaque
hatnguyencanh@vnlap03333:~/Documents/K8s/DEMO/k8s$ echo cGFzc3dvcmQK | base64 -d
password
```

- <u>**Create applications**</u>

Backend:

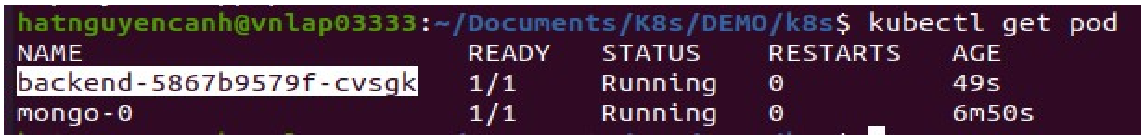
# Create backend

kubectl apply -f backend.yaml



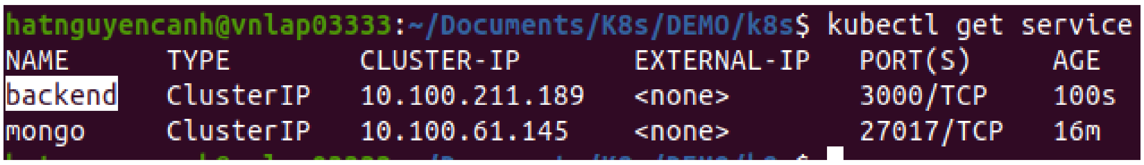
```
hatnguyencanh@vnlap03333:~/Documents/K8s/DEMO/k8s$ kubectl apply -f backend.yaml
service/backend created
deployment.apps/backend created
```

# verify backend pods
kubectl get pod



```
hatnguyencanh@vnlap03333:~/Documents/K8s/DEMO/k8s$ kubectl get pod
NAME                       READY   STATUS    RESTARTS   AGE
backend-5867b9579f-cvsgk   1/1     Running   0          49s
mongo-0                    1/1     Running   0          6m50s
```

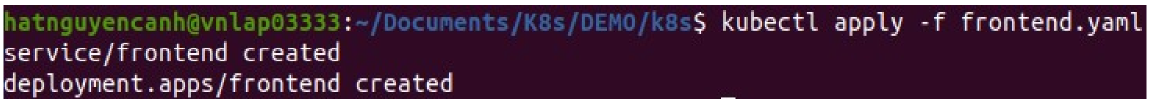
# verify backend service
kubectl get service



```
hatnguyencanh@vnlap03333:~/Documents/K8s/DEMO/k8s$ kubectl get service
NAME      TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)     AGE
backend   ClusterIP   10.100.211.189   <none>        3000/TCP    100s
mongo     ClusterIP   10.100.61.145    <none>        27017/TCP   16m
```

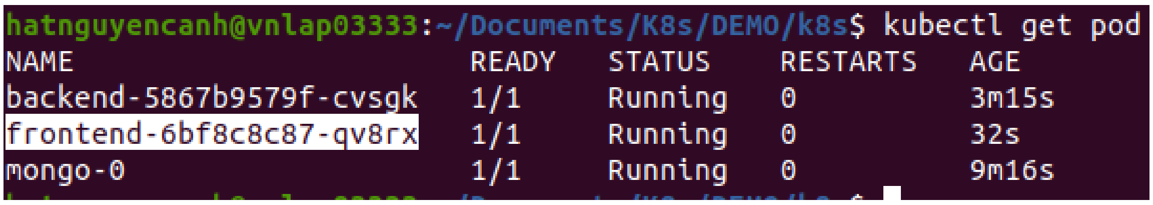
Frontend:

# Create frontend
kubectl apply -f frontend.yaml



```
hatnguyencanh@vnlap03333:~/Documents/K8s/DEMO/k8s$ kubectl apply -f frontend.yaml
service/frontend created
deployment.apps/frontend created
```

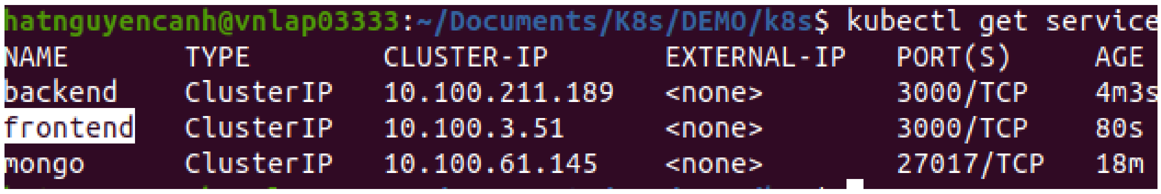
# verify frontend pods
kubectl get pod



```
hatnguyencanh@vnlap03333:~/Documents/K8s/DEMO/k8s$ kubectl get pod
NAME                        READY   STATUS    RESTARTS   AGE
backend-5867b9579f-cvsgk    1/1     Running   0          3m15s
frontend-6bf8c8c87-qv8rx    1/1     Running   0          32s
mongo-0                     1/1     Running   0          9m16s
```

# verify frontend service

kubectl get service

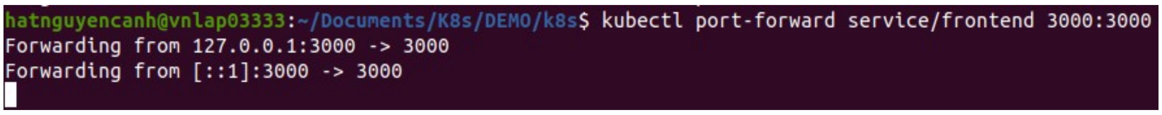


```
hatnguyencanh@vnlap03333:~/Documents/K8s/DEMO/k8s$ kubectl get service
NAME       TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)     AGE
backend    ClusterIP   10.100.211.189   <none>        3000/TCP    4m3s
frontend   ClusterIP   10.100.3.51      <none>        3000/TCP    80s
mongo      ClusterIP   10.100.61.145    <none>        27017/TCP   18m
```

Application verification by exposing frontend service

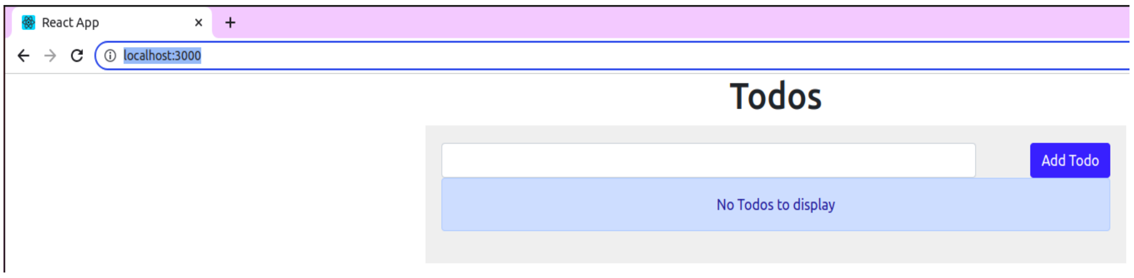
# expose frontend service to access application

kubectl port-forward service/frontend 3000:3000



```
hatnguyencanh@vnlap03333:~/Documents/K8s/DEMO/k8s$ kubectl port-forward service/frontend 3000:3000
Forwarding from 127.0.0.1:3000 -> 3000
Forwarding from [::1]:3000 -> 3000
```

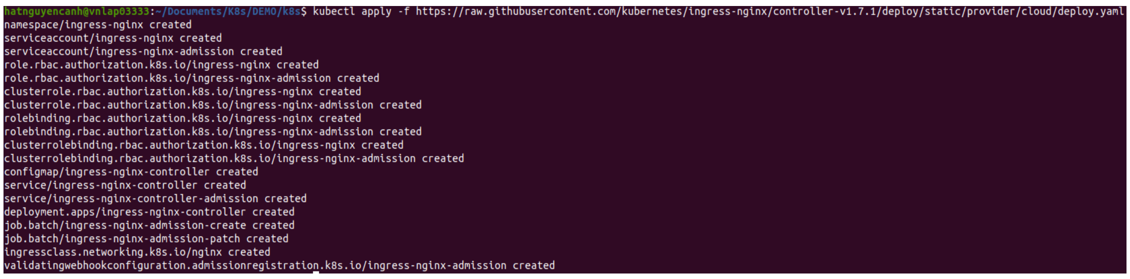
# open browser to access application at URL: locahost:3000



Application Ingress

# install NGINX ingress controller

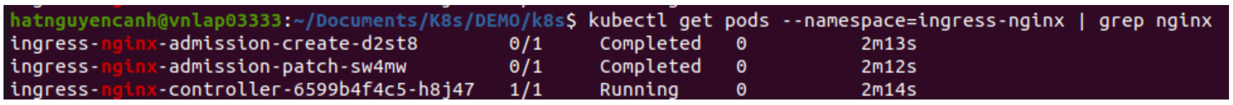
kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v1.7.1/deploy/static/provider/cloud/deploy.yaml



```
hatnguyencanh@vnlap03333:~/Documents/K8s/DEMO/k8s$ kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v1.7.1/deploy/static/provider/cloud/deploy.yaml
namespace/ingress-nginx created
serviceaccount/ingress-nginx created
serviceaccount/ingress-nginx-admission created
role.rbac.authorization.k8s.io/ingress-nginx created
role.rbac.authorization.k8s.io/ingress-nginx-admission created
clusterrole.rbac.authorization.k8s.io/ingress-nginx created
clusterrole.rbac.authorization.k8s.io/ingress-nginx-admission created
rolebinding.rbac.authorization.k8s.io/ingress-nginx created
rolebinding.rbac.authorization.k8s.io/ingress-nginx-admission created
clusterrolebinding.rbac.authorization.k8s.io/ingress-nginx created
clusterrolebinding.rbac.authorization.k8s.io/ingress-nginx-admission created
configmap/ingress-nginx-controller created
service/ingress-nginx-controller created
service/ingress-nginx-controller-admission created
deployment.apps/ingress-nginx-controller created
job.batch/ingress-nginx-admission-create created
job.batch/ingress-nginx-admission-patch created
ingressclass.networking.k8s.io/nginx created
validatingwebhookconfiguration.admissionregistration.k8s.io/ingress-nginx-admission created
```

# verify ingress controller installation

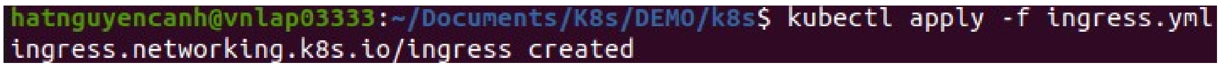
kubectl get pods --namespace=ingress-nginx | grep nginx



```
hatnguyencanh@vnlap03333:~/Documents/K8s/DEMO/k8s$ kubectl get pods --namespace=ingress-nginx | grep nginx
ingress-nginx-admission-create-d2st8        0/1     Completed   0          2m13s
ingress-nginx-admission-patch-sw4mw         0/1     Completed   0          2m12s
ingress-nginx-controller-6599b4f4c5-h8j47   1/1     Running     0          2m14s
```

# install application ingress

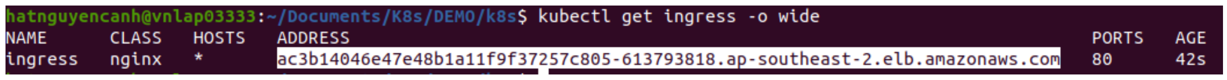
kubectl apply -f ingress.yml



```
hatnguyencanh@vnlap03333:~/Documents/K8s/DEMO/k8s$ kubectl apply -f ingress.yml
ingress.networking.k8s.io/ingress created
```

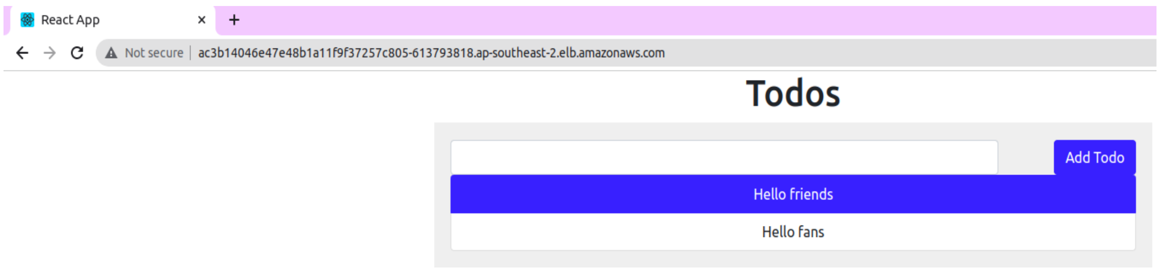
# verify application ingress

kubectl get ingress -o wide



```
hatnguyencanh@vnlap03333:~/Documents/K8s/DEMO/k8s$ kubectl get ingress -o wide
NAME      CLASS   HOSTS   ADDRESS                                                                    PORTS   AGE
ingress   nginx   *       ac3b14046e47e48b1a11f9f37257c805-613793818.ap-southeast-2.elb.amazonaws.com   80      42s
```

# open browser to access application at URL created by ingress controller (e.g: ac3b14046e47e48b1a11f9f37257c805-613793818.ap-southeast-2.elb.amazonaws.com)



React App
Not secure
ac3b14046e47e48b1a11f9f37257c805-613793818.ap-southeast-2.elb.amazonaws.com
Todos
Add Todo
Hello friends
Hello fans

III. Frequently Asked Questions

III.1. Setup EBS CSI driver

- Manually setup EBS CSI plugin, see at managing-ebs-csi
- Configure the EKS node role to have the AWS provided policy "AmazonEBSCSIDriverPolicy"

III.2. Demo source code

- https://github.com/nashtech-garage/kubernetes

IV. References

- EBS CSI driver setup: https://docs.aws.amazon.com/eks/latest/userguide/managing-ebs-csi.html
- NGINX Ingress controller setup: https://kubernetes.github.io/ingress-nginx/deploy/
- Source code for application: https://github.com/docker/awesome-compose/tree/master/react-express-mongodb