Lesson 06 Demo 04

Setting up Ingress Controller with Transport Layer Security

Objective: To implement the transport layer security by deploying an Ingress rule to generate an SSL certificate

Tools required: kubeadm, kubectl, kubelet, and containerd

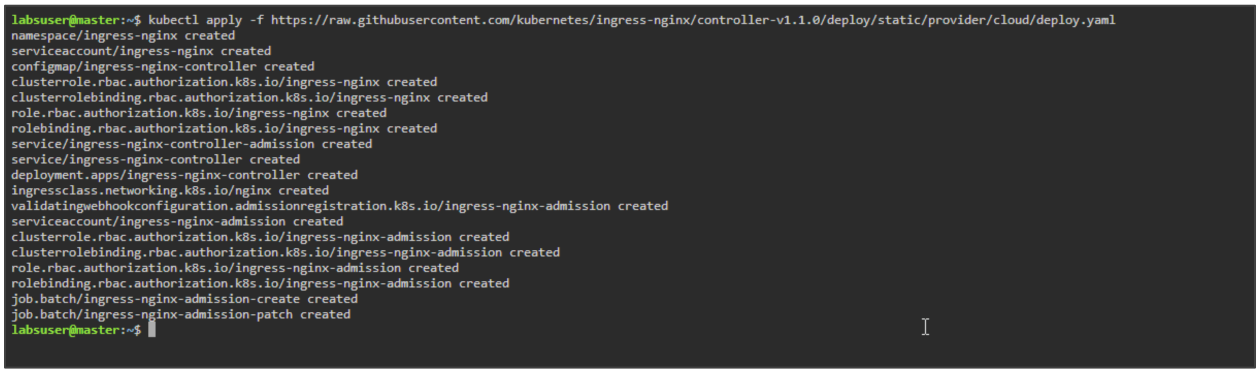
Prerequisites: A Kubernetes cluster should already be set up (refer to the steps in Lesson 02, Demo 01 for guidance).

Steps to be followed:

1. Deploy Ingress
2. Deploy HTTPD and OpenShift
3. Generate a self-signed SSL certificate and a TLS certificate
4. Verify the Ingress rule

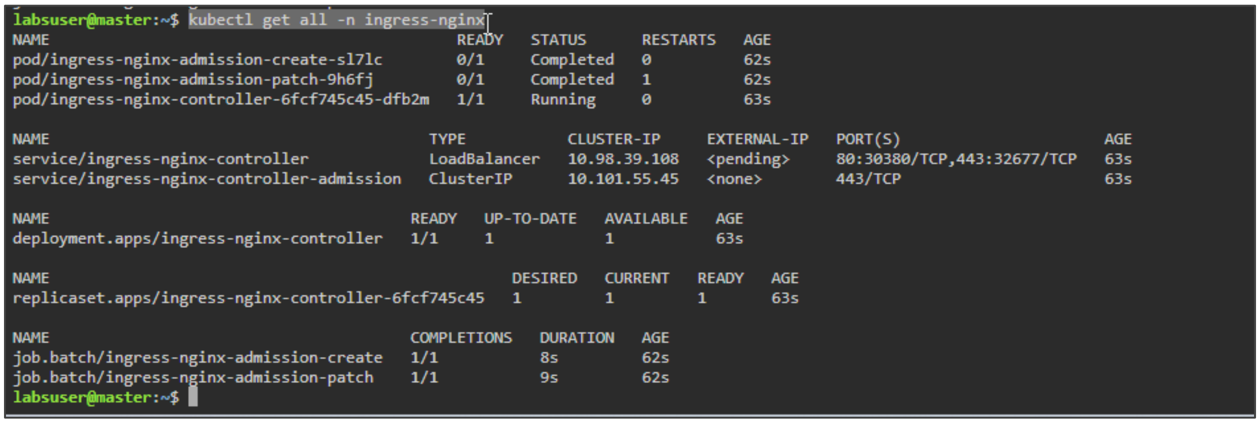
Step 1: Deploy Ingress

1.1 Deploy Ingress by using the following command:
kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v1.1.0/deploy/static/provider/cloud/deploy.yaml



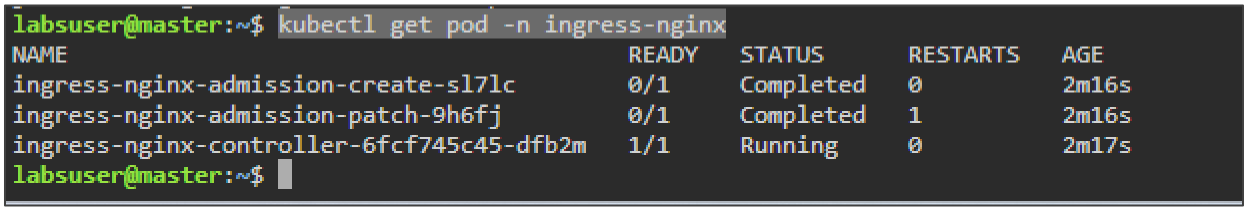
```
labsuser@master:~$ kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v1.1.0/deploy/static/provider/cloud/deploy.yaml
namespace/ingress-nginx created
serviceaccount/ingress-nginx created
configmap/ingress-nginx-controller created
clusterrole.rbac.authorization.k8s.io/ingress-nginx created
clusterrolebinding.rbac.authorization.k8s.io/ingress-nginx created
role.rbac.authorization.k8s.io/ingress-nginx created
rolebinding.rbac.authorization.k8s.io/ingress-nginx created
service/ingress-nginx-controller-admission created
service/ingress-nginx-controller created
deployment.apps/ingress-nginx-controller created
ingressclass.networking.k8s.io/nginx created
validatingwebhookconfiguration.admissionregistration.k8s.io/ingress-nginx-admission created
serviceaccount/ingress-nginx-admission created
clusterrole.rbac.authorization.k8s.io/ingress-nginx-admission created
clusterrolebinding.rbac.authorization.k8s.io/ingress-nginx-admission created
role.rbac.authorization.k8s.io/ingress-nginx-admission created
rolebinding.rbac.authorization.k8s.io/ingress-nginx-admission created
job.batch/ingress-nginx-admission-create created
job.batch/ingress-nginx-admission-patch created
labsuser@master:~$
```

1.2 Verify all Ingress deployments, pods, and services by using the following command:
kubectl get all -n ingress-nginx



```
labsuser@master:~$ kubectl get all -n ingress-nginx
NAME                                           READY   STATUS      RESTARTS   AGE
pod/ingress-nginx-admission-create-sl7lc       0/1     Completed   0          62s
pod/ingress-nginx-admission-patch-9h6fj        0/1     Completed   1          62s
pod/ingress-nginx-controller-6fcf745c45-dfb2m  1/1     Running     0          63s

NAME                                         TYPE           CLUSTER-IP     EXTERNAL-IP   PORT(S)                      AGE
service/ingress-nginx-controller             LoadBalancer   10.98.39.108   <pending>     80:30380/TCP,443:32677/TCP   63s
service/ingress-nginx-controller-admission   ClusterIP      10.101.55.45   <none>        443/TCP                      63s

NAME                                       READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/ingress-nginx-controller   1/1     1            1           63s

NAME                                                  DESIRED   CURRENT   READY   AGE
replicaset.apps/ingress-nginx-controller-6fcf745c45   1         1         1       63s

NAME                                       COMPLETIONS   DURATION   AGE
job.batch/ingress-nginx-admission-create   1/1           8s         62s
job.batch/ingress-nginx-admission-patch    1/1           9s         62s
labsuser@master:~$
```

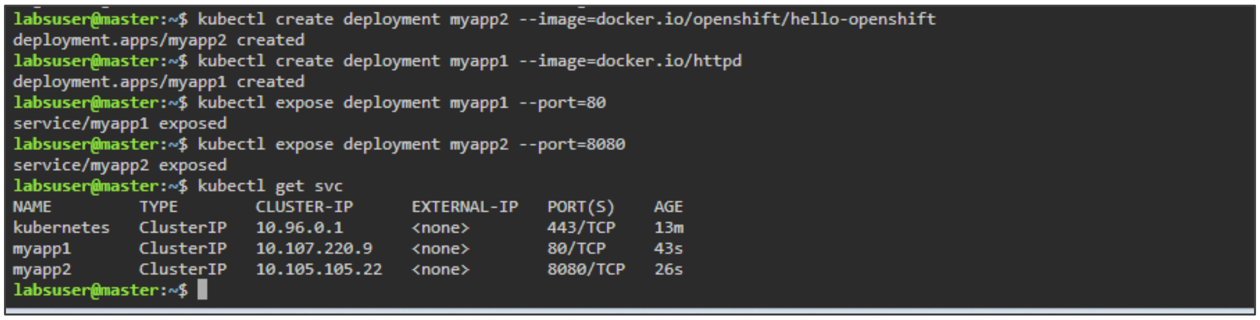
1.3 List the created Ingress pods using the following command:
kubectl get pod -n ingress-nginx



```
labsuser@master:~$ kubectl get pod -n ingress-nginx
NAME                                        READY   STATUS      RESTARTS   AGE
ingress-nginx-admission-create-sl7lc        0/1     Completed   0          2m16s
ingress-nginx-admission-patch-9h6fj         0/1     Completed   1          2m16s
ingress-nginx-controller-6fcf745c45-dfb2m   1/1     Running     0          2m17s
labsuser@master:~$
```

Step 2: Deploy HTTPD and OpenShift

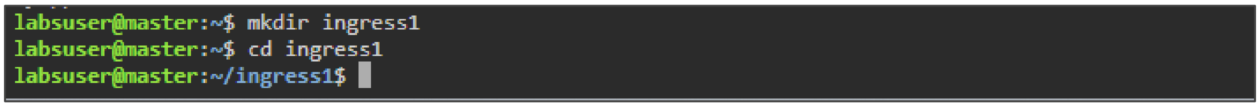
2.1 Deploy HTTPD and OpenShift deployments using the following command:
kubectl create deployment myapp2 --image=docker.io/openshift/hello-openshift
kubectl create deployment myapp1 --image=docker.io/httpd
kubectl expose deployment myapp1 --port=80
kubectl expose deployment myapp2 --port=8080
kubectl get svc



```
labsuser@master:~$ kubectl create deployment myapp2 --image=docker.io/openshift/hello-openshift
deployment.apps/myapp2 created
labsuser@master:~$ kubectl create deployment myapp1 --image=docker.io/httpd
deployment.apps/myapp1 created
labsuser@master:~$ kubectl expose deployment myapp1 --port=80
service/myapp1 exposed
labsuser@master:~$ kubectl expose deployment myapp2 --port=8080
service/myapp2 exposed
labsuser@master:~$ kubectl get svc
NAME         TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE
kubernetes   ClusterIP   10.96.0.1       <none>        443/TCP    13m
myapp1       ClusterIP   10.107.220.9    <none>        80/TCP     43s
myapp2       ClusterIP   10.105.105.22   <none>        8080/TCP   26s
labsuser@master:~$
```

2.2 Create a directory for Ingress TLS using the following commands:

mkdir ingress1
cd ingress1/

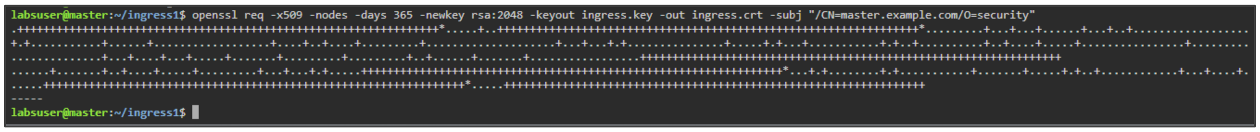


```
labsuser@master:~$ mkdir ingress1
labsuser@master:~$ cd ingress1
labsuser@master:~/ingress1$
```

Step 3: Generate a self-signed SSL certificate and a TLS certificate

3.1 Generate a self-signed SSL certificate using the OpenSSL command:

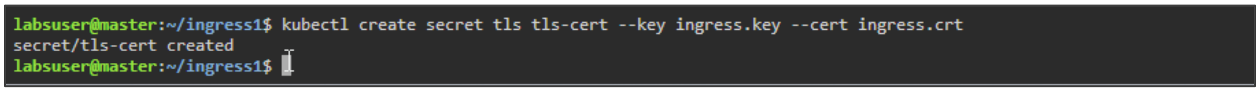
openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout ingress.key -out ingress.crt -subj "/CN=master.example.com/O=security"



```
labsuser@master:~/ingress1$ openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout ingress.key -out ingress.crt -subj "/CN=master.example.com/O=security"
.+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*.....+..++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*.........+...+...+......+...+..+...................
+.+...........+......+..................+....+..+....+.........+....................+...+...+.+...............+.....+.+...+...........+.+..+..........+..+....+.....+................+.........
.............+...+....+...+.....+.......+.........+.........+..+......+.......+................+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
......+.......+..+....+.....+.........+...+...+.+.....+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*...+.+........+.+...........+.......+.....+.+..+...........+...+....+.
.....+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*.....+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
-----
labsuser@master:~/ingress1$
```

3.2 Create a **tls-cert** certificate using the following command:

kubectl create secret tls tls-cert --key ingress.key --cert ingress.crt

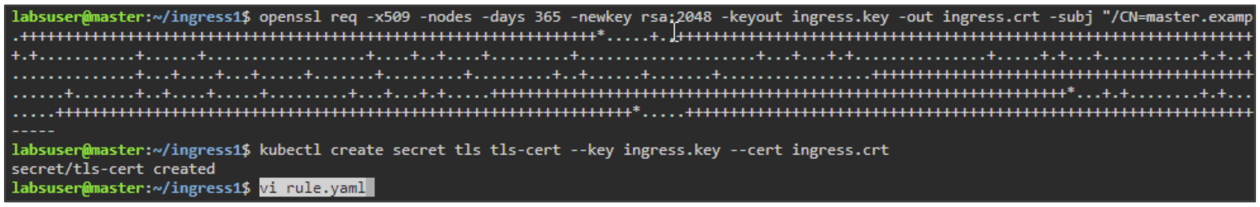


```
labsuser@master:~/ingress1$ kubectl create secret tls tls-cert --key ingress.key --cert ingress.crt
secret/tls-cert created
labsuser@master:~/ingress1$
```

Step 4: Verify the Ingress rule

4.1 Create a YAML file for the Ingress rule using the following command:

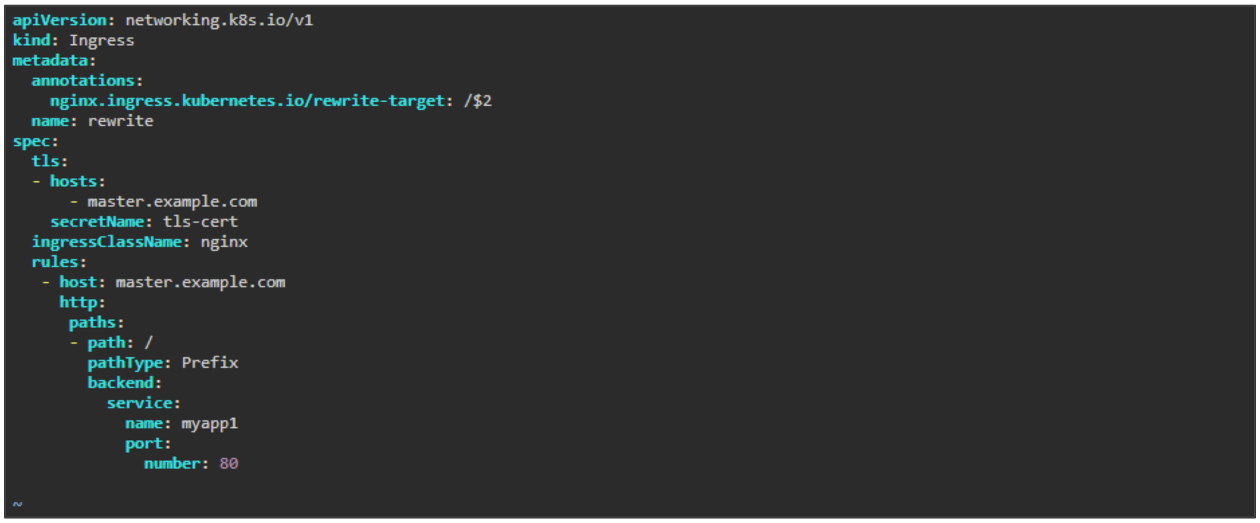
vi rule.yaml



```
labsuser@master:~/ingress1$ openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout ingress.key -out ingress.crt -subj "/CN=master.examp
.+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*.....+..+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+.+...........+......+..................+....+..+....+.........+....................+...+...+.+...............+.....+.+...+...........+.+..+
.............+...+....+...+.....+.......+.........+.........+..+......+.......+................+++++++++++++++++++++++++++++++++++++++++++++
......+.......+..+....+.....+.........+...+...+.+.....+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*...+.+........+.+...
.....+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*.....+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
-----
labsuser@master:~/ingress1$ kubectl create secret tls tls-cert --key ingress.key --cert ingress.crt
secret/tls-cert created
labsuser@master:~/ingress1$ vi rule.yaml
```

4.2 Write the following code within the **rule.yaml** file:

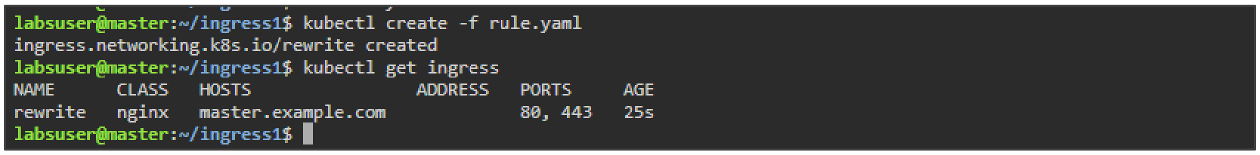
```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /$2
  name: rewrite
spec:
  tls:
  - hosts:
      - master.example.com
    secretName: tls-cert
  ingressClassName: nginx
  rules:
   - host: master.example.com
     http:
      paths:
      - path: /
        pathType: Prefix
        backend:

          service:
            name: myapp1
            port:
              number: 80
```



```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /$2
  name: rewrite
spec:
  tls:
  - hosts:
      - master.example.com
    secretName: tls-cert
  ingressClassName: nginx
  rules:
   - host: master.example.com
     http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: myapp1
            port:
              number: 80

~
```

4.3 Create and verify the Ingress rule using the following command:

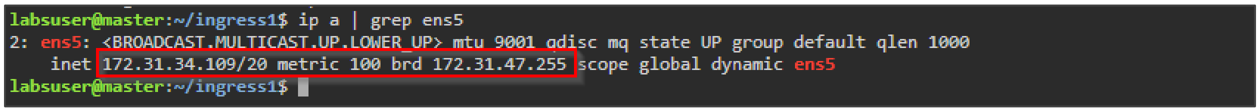
kubectl create -f rule.yaml
kubectl get ingress



```
labsuser@master:~/ingress1$ kubectl create -f rule.yaml
ingress.networking.k8s.io/rewrite created
labsuser@master:~/ingress1$ kubectl get ingress
NAME      CLASS   HOSTS                ADDRESS   PORTS     AGE
rewrite   nginx   master.example.com             80, 443   25s
labsuser@master:~/ingress1$
```

4.4 Execute the following command to get the IP address associated with ens5:

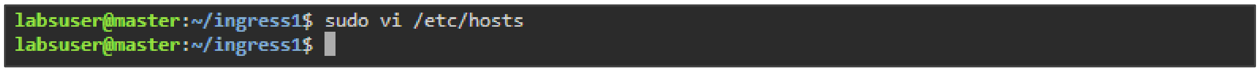
ip a | grep ens5



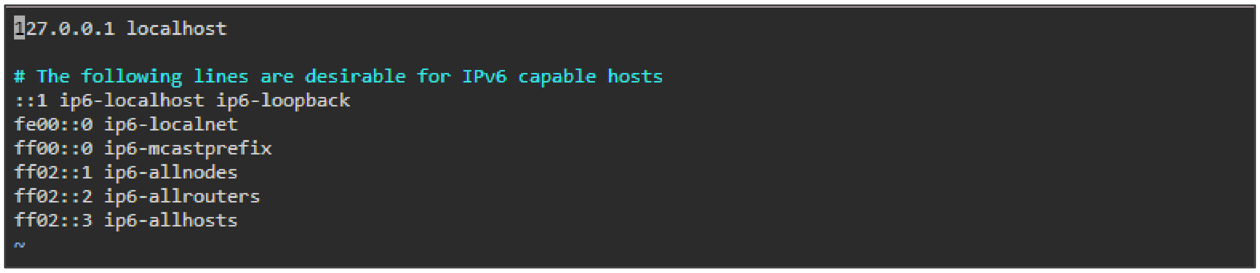
```
labsuser@master:~/ingress1$ ip a | grep ens5
2: ens5: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9001 qdisc mq state UP group default qlen 1000
    inet 172.31.34.109/20 metric 100 brd 172.31.47.255 scope global dynamic ens5
labsuser@master:~/ingress1$
```

4.5 Execute the following command to print the IP hostnames and addresses for the localhost:

sudo vi /etc/hosts



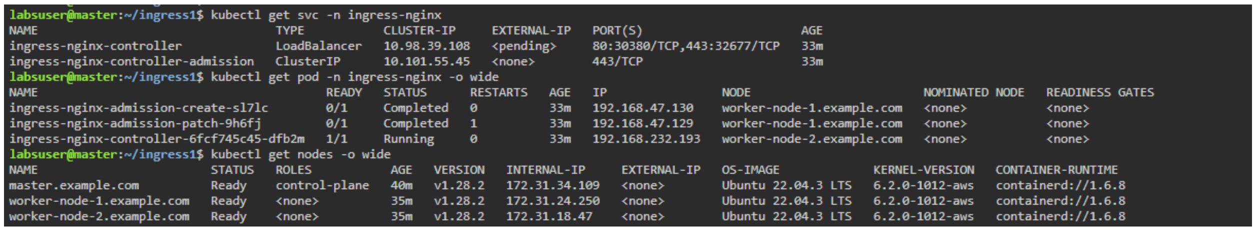
```
labsuser@master:~/ingress1$ sudo vi /etc/hosts
labsuser@master:~/ingress1$
```



```
127.0.0.1 localhost

# 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
ff02::3 ip6-allhosts
~
```

4.6 List the running Ingress pods using the following commands:

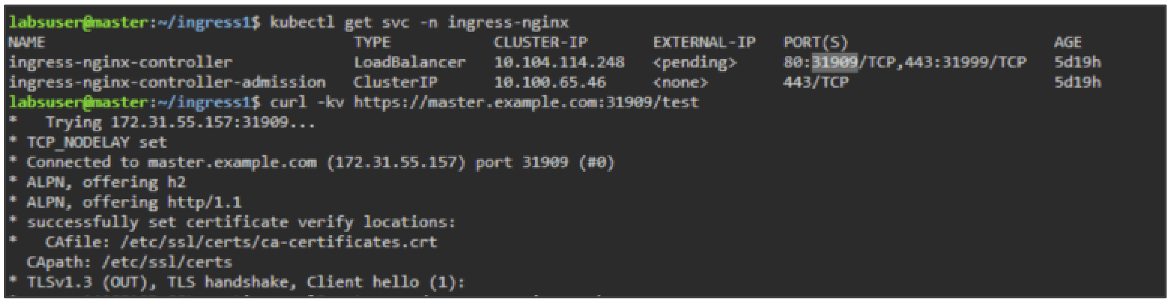
kubectl get svc -n ingress-nginx
kubectl get pod -n ingress-nginx -o wide
kubectl get nodes -o wide



```
labsuser@master:~/ingress1$ kubectl get svc -n ingress-nginx
NAME                                 TYPE           CLUSTER-IP     EXTERNAL-IP   PORT(S)                      AGE
ingress-nginx-controller             LoadBalancer   10.98.39.108   <pending>     80:30380/TCP,443:32677/TCP   33m
ingress-nginx-controller-admission   ClusterIP      10.101.55.45   <none>        443/TCP                      33m
labsuser@master:~/ingress1$ kubectl get pod -n ingress-nginx -o wide
NAME                                       READY   STATUS      RESTARTS   AGE   IP                NODE                         NOMINATED NODE   READINESS GATES
ingress-nginx-admission-create-sl7lc       0/1     Completed   0          33m   192.168.47.130    worker-node-1.example.com   <none>           <none>
ingress-nginx-admission-patch-9h6fj        0/1     Completed   1          33m   192.168.47.129    worker-node-1.example.com   <none>           <none>
ingress-nginx-controller-6fcf745c45-dfb2m  1/1     Running     0          33m   192.168.232.193   worker-node-2.example.com   <none>           <none>
labsuser@master:~/ingress1$ kubectl get nodes -o wide
NAME                        STATUS   ROLES           AGE   VERSION   INTERNAL-IP     EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION   CONTAINER-RUNTIME
master.example.com          Ready    control-plane   40m   v1.28.2   172.31.34.109   <none>        Ubuntu 22.04.3 LTS   6.2.0-1012-aws   containerd://1.6.8
worker-node-1.example.com   Ready    <none>          35m   v1.28.2   172.31.24.250   <none>        Ubuntu 22.04.3 LTS   6.2.0-1012-aws   containerd://1.6.8
worker-node-2.example.com   Ready    <none>          35m   v1.28.2   172.31.18.47    <none>        Ubuntu 22.04.3 LTS   6.2.0-1012-aws   containerd://1.6.8
```

This shows the **Node IP** and **NodePort** of the Ingress service.

4.7 Verify the generated certificate using the following commands:
kubectl get svc -n ingress-nginx
curl -kv https://master.example.com:31909/test



```
labsuser@master:~/ingress1$ kubectl get svc -n ingress-nginx
NAME                                 TYPE           CLUSTER-IP       EXTERNAL-IP   PORT(S)                      AGE
ingress-nginx-controller             LoadBalancer   10.104.114.248   <pending>     80:31909/TCP,443:31999/TCP   5d19h
ingress-nginx-controller-admission   ClusterIP      10.100.65.46     <none>        443/TCP                      5d19h
labsuser@master:~/ingress1$ curl -kv https://master.example.com:31909/test
*   Trying 172.31.55.157:31909...
* TCP_NODELAY set
* Connected to master.example.com (172.31.55.157) port 31909 (#0)
* ALPN, offering h2
* ALPN, offering http/1.1
* successfully set certificate verify locations:
*   CAfile: /etc/ssl/certs/ca-certificates.crt
  CApath: /etc/ssl/certs
* TLSv1.3 (OUT), TLS handshake, Client hello (1):
```

By following these steps, you can successfully implement transport layer security by generating an SSL certificate using Ingress.