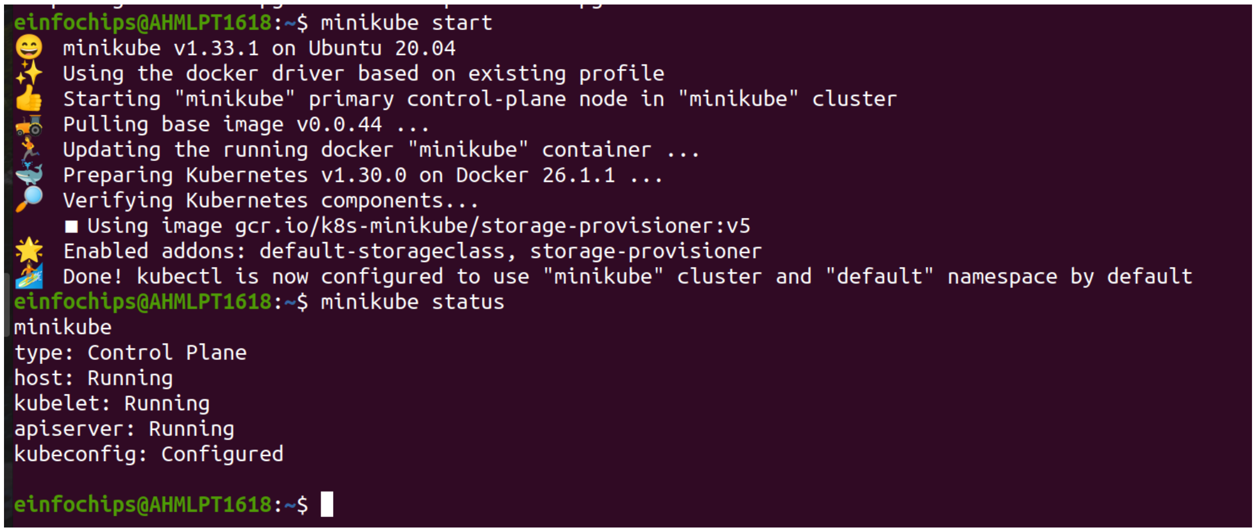
Project Overview

Participants are required to deploy a simple static web application on a Kubernetes cluster using Minikube, set up advanced ingress networking with URL rewriting and sticky sessions, and configure horizontal pod autoscaling to manage traffic efficiently. The project will be divided into stages, with each stage focusing on specific aspects of Kubernetes ingress, URL rewriting, sticky sessions, and autoscaling.

Requirements and Deliverables

Stage 1: Setting Up the Kubernetes Cluster and Static Web App

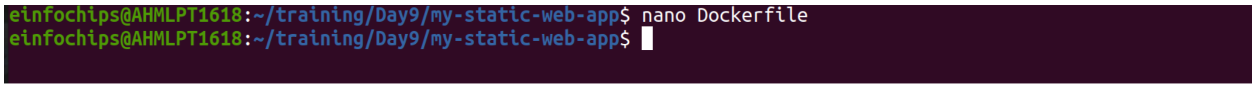
1. **Set Up Minikube:**
   - Ensure Minikube is installed and running on the local Ubuntu machine.
   - Verify the Kubernetes cluster is functioning correctly.



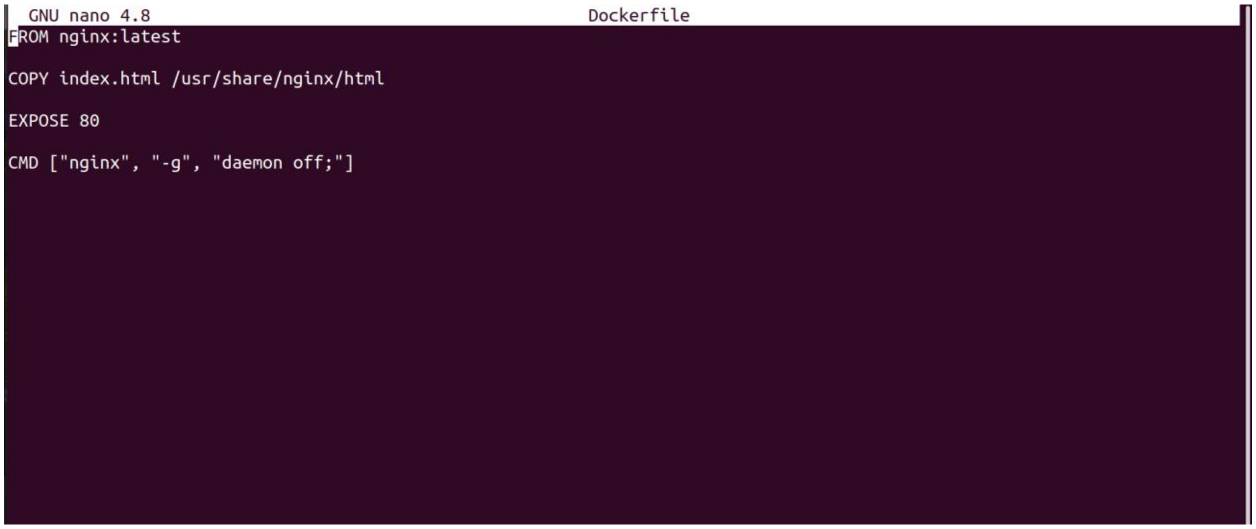
```
einfochips@AHMLPT1618:~$ minikube start
😄  minikube v1.33.1 on Ubuntu 20.04
✨  Using the docker driver based on existing profile
👍  Starting "minikube" primary control-plane node in "minikube" cluster
🚜  Pulling base image v0.0.44 ...
🏃  Updating the running docker "minikube" container ...
🐳  Preparing Kubernetes v1.30.0 on Docker 26.1.1 ...
🔎  Verifying Kubernetes components...
    ▪ Using image gcr.io/k8s-minikube/storage-provisioner:v5
🌟  Enabled addons: default-storageclass, storage-provisioner
🏄  Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
einfochips@AHMLPT1618:~$ minikube status
minikube
type: Control Plane
host: Running
kubelet: Running
apiserver: Running
kubeconfig: Configured

einfochips@AHMLPT1618:~$ 
```

2. Deploy Static Web App:

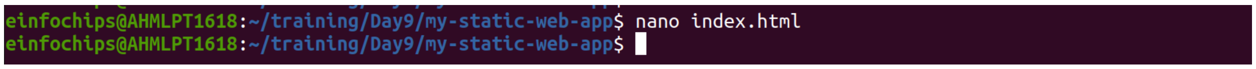
- Create a Dockerfile for a simple static web application (e.g., an HTML page served by Nginx).
- Build a Docker image for the static web application.
- Push the Docker image to Docker Hub or a local registry.



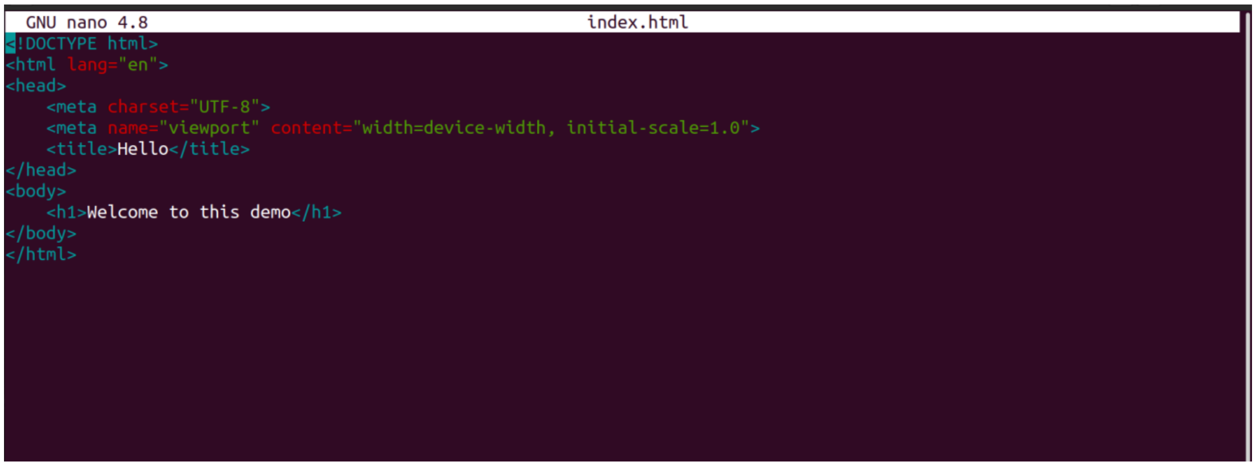
```
einfochips@AHMLPT1618:~/training/Day9/my-static-web-app$ nano Dockerfile
einfochips@AHMLPT1618:~/training/Day9/my-static-web-app$ 
```



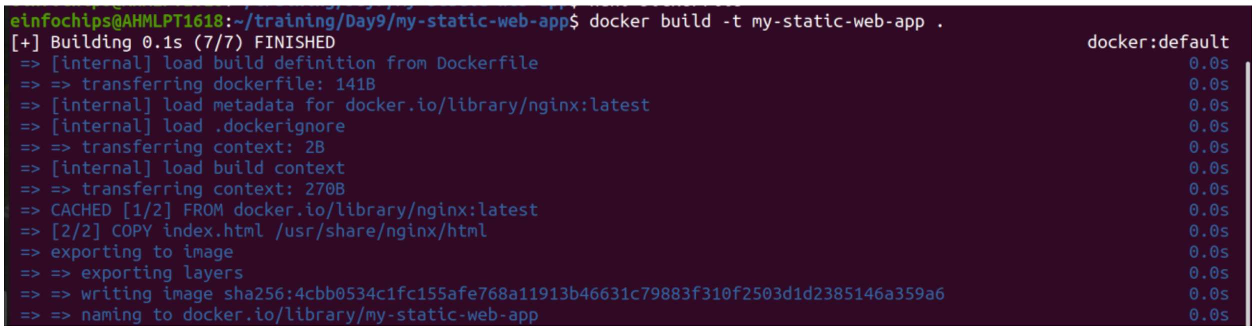
```
  GNU nano 4.8                                          Dockerfile
FROM nginx:latest

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

EXPOSE 80

CMD ["nginx", "-g", "daemon off;"]
```



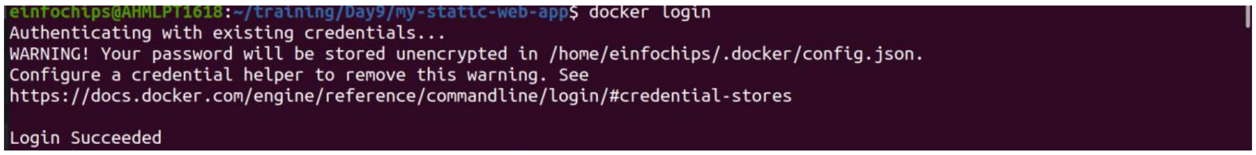
```
einfochips@AHMLPT1618:~/training/Day9/my-static-web-app$ nano index.html
einfochips@AHMLPT1618:~/training/Day9/my-static-web-app$ 
```



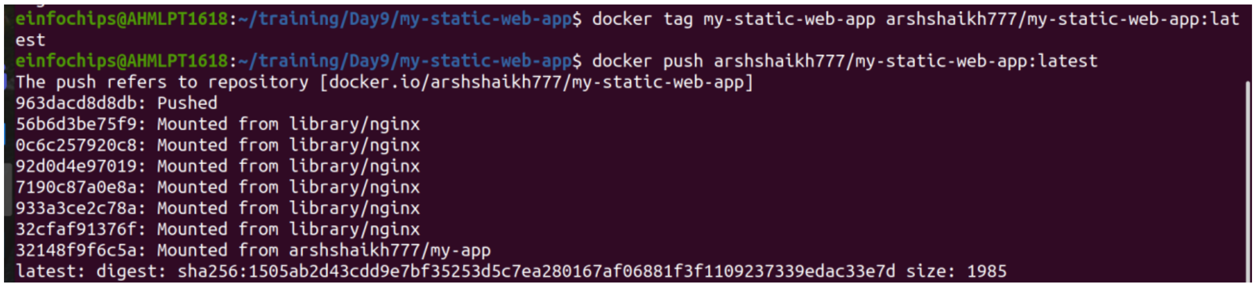
```
  GNU nano 4.8                                          index.html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Hello</title>
</head>
<body>
    <h1>Welcome to this demo</h1>
</body>
</html>
```



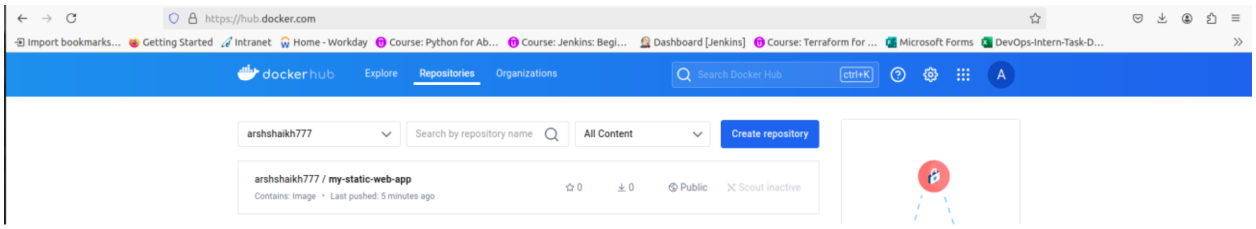
```
einfochips@AHMLPT1618:~/training/Day9/my-static-web-app$ docker build -t my-static-web-app .
[+] Building 0.1s (7/7) FINISHED                                                         docker:default
 => [internal] load build definition from Dockerfile                                               0.0s
 => => transferring dockerfile: 141B                                                               0.0s
 => [internal] load metadata for docker.io/library/nginx:latest                                    0.0s
 => [internal] load .dockerignore                                                                  0.0s
 => => transferring context: 2B                                                                    0.0s
 => [internal] load build context                                                                  0.0s
 => => transferring context: 270B                                                                  0.0s
 => CACHED [1/2] FROM docker.io/library/nginx:latest                                               0.0s
 => [2/2] COPY index.html /usr/share/nginx/html                                                    0.0s
 => exporting to image                                                                             0.0s
 => => exporting layers                                                                            0.0s
 => => writing image sha256:4cbb0534c1fc155afe768a11913b46631c79883f310f2503d1d2385146a359a6       0.0s
 => => naming to docker.io/library/my-static-web-app                                               0.0s
```



```
einfochips@AHMLPT1618:~/training/Day9/my-static-web-app$ docker login
Authenticating with existing credentials...
WARNING! Your password will be stored unencrypted in /home/einfochips/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credential-stores

Login Succeeded
```



```
einfochips@AHMLPT1618:~/training/Day9/my-static-web-app$ docker tag my-static-web-app arshshaikh777/my-static-web-app:lat
est
einfochips@AHMLPT1618:~/training/Day9/my-static-web-app$ docker push arshshaikh777/my-static-web-app:latest
The push refers to repository [docker.io/arshshaikh777/my-static-web-app]
963dacd8d8db: Pushed
56b6d3be75f9: Mounted from library/nginx
0c6c257920c8: Mounted from library/nginx
92d0d4e97019: Mounted from library/nginx
7190c87a0e8a: Mounted from library/nginx
933a3ce2c78a: Mounted from library/nginx
32cfaf91376f: Mounted from library/nginx
32148f9f6c5a: Mounted from arshshaikh777/my-app
latest: digest: sha256:1505ab2d43cdd9e7bf35253d5c7ea280167af06881f3f1109237339edac33e7d size: 1985
```

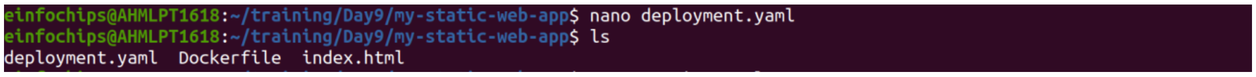


https://hub.docker.com

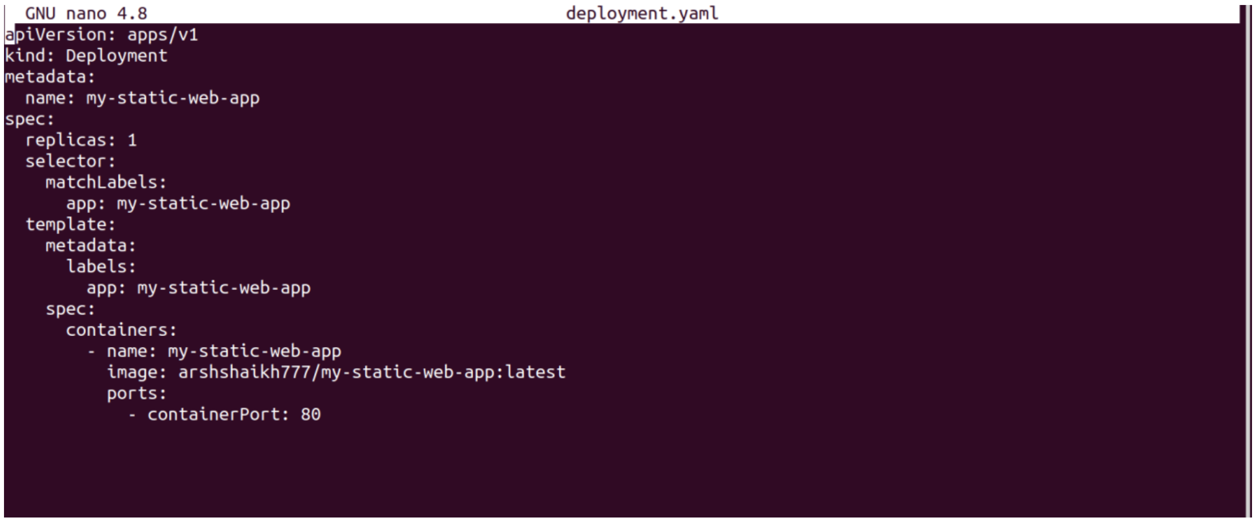
arshshaikh777 / my-static-web-app
Contains: Image • Last pushed: 5 minutes ago
☆ 0 ⤓ 0 Public Scout inactive

3. Kubernetes Deployment:

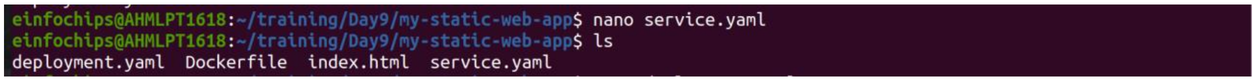
- Write a Kubernetes deployment manifest to deploy the static web application.
- Write a Kubernetes service manifest to expose the static web application within the cluster.
- Apply the deployment and service manifests to the Kubernetes cluster.

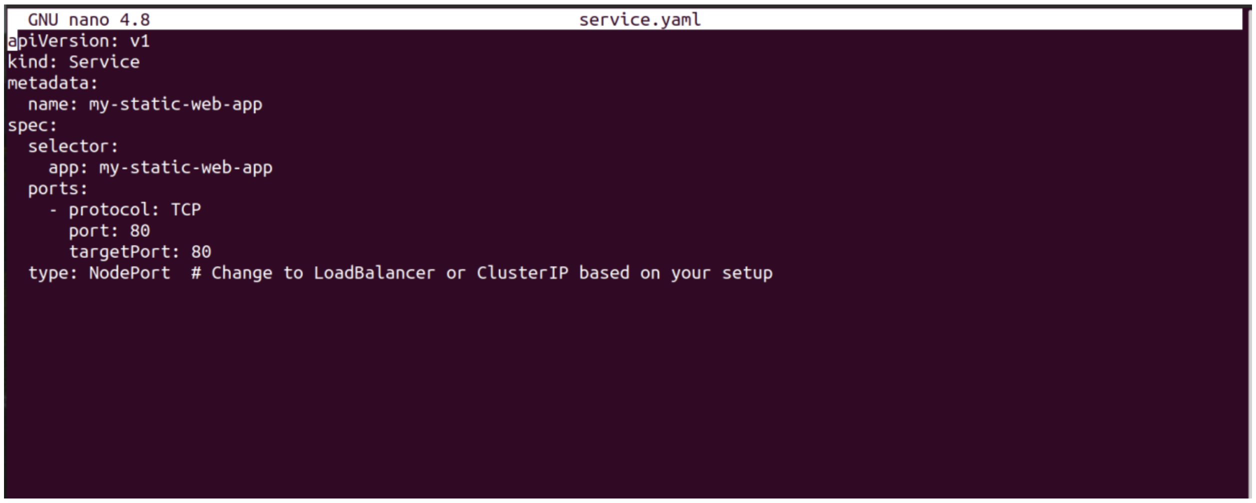


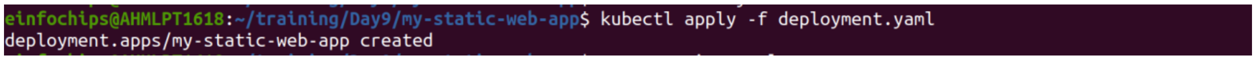
```
einfochips@AHMLPT1618:~/training/Day9/my-static-web-app$ nano deployment.yaml
einfochips@AHMLPT1618:~/training/Day9/my-static-web-app$ ls
deployment.yaml  Dockerfile  index.html
```

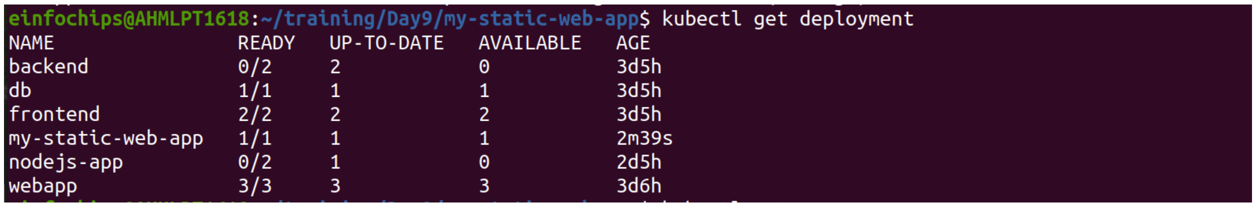


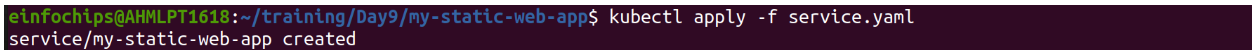
```
  GNU nano 4.8                                    deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-static-web-app
spec:
  replicas: 1
  selector:
    matchLabels:
      app: my-static-web-app
  template:
    metadata:
      labels:
        app: my-static-web-app
    spec:
      containers:
        - name: my-static-web-app
          image: arshshaikh777/my-static-web-app:latest
          ports:
            - containerPort: 80
```

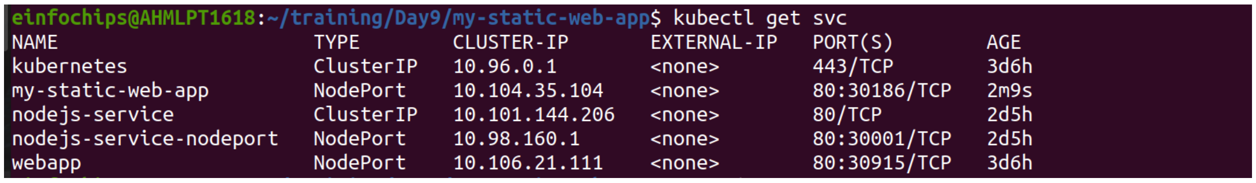

```
einfochips@AHMLPT1618:~/training/Day9/my-static-web-app$ nano service.yaml
einfochips@AHMLPT1618:~/training/Day9/my-static-web-app$ ls
deployment.yaml  Dockerfile  index.html  service.yaml
```

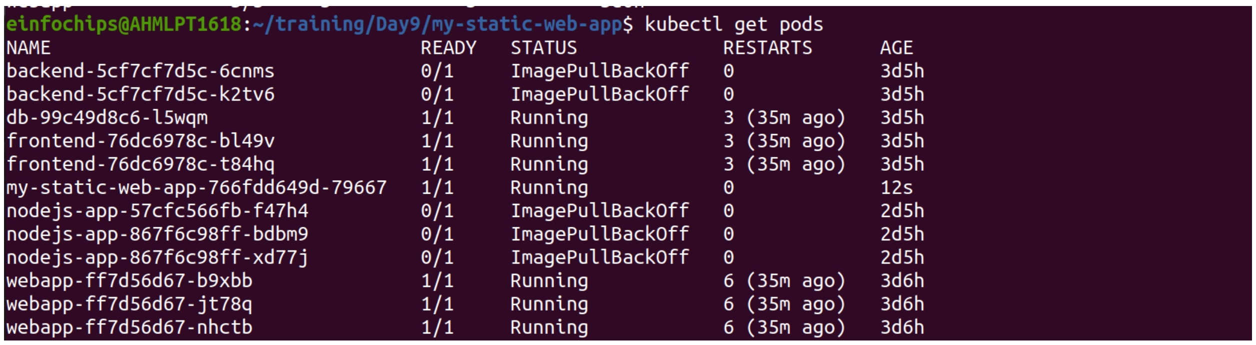

```
  GNU nano 4.8                                          service.yaml
apiVersion: v1
kind: Service
metadata:
  name: my-static-web-app
spec:
  selector:
    app: my-static-web-app
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
  type: NodePort  # Change to LoadBalancer or ClusterIP based on your setup
```


```
einfochips@AHMLPT1618:~/training/Day9/my-static-web-app$ kubectl apply -f deployment.yaml
deployment.apps/my-static-web-app created
```


```
einfochips@AHMLPT1618:~/training/Day9/my-static-web-app$ kubectl get deployment
NAME                READY   UP-TO-DATE   AVAILABLE   AGE
backend             0/2     2            0           3d5h
db                  1/1     1            1           3d5h
frontend            2/2     2            2           3d5h
my-static-web-app   1/1     1            1           2m39s
nodejs-app          0/2     1            0           2d5h
webapp              3/3     3            3           3d6h
```


```
einfochips@AHMLPT1618:~/training/Day9/my-static-web-app$ kubectl apply -f service.yaml
service/my-static-web-app created
```

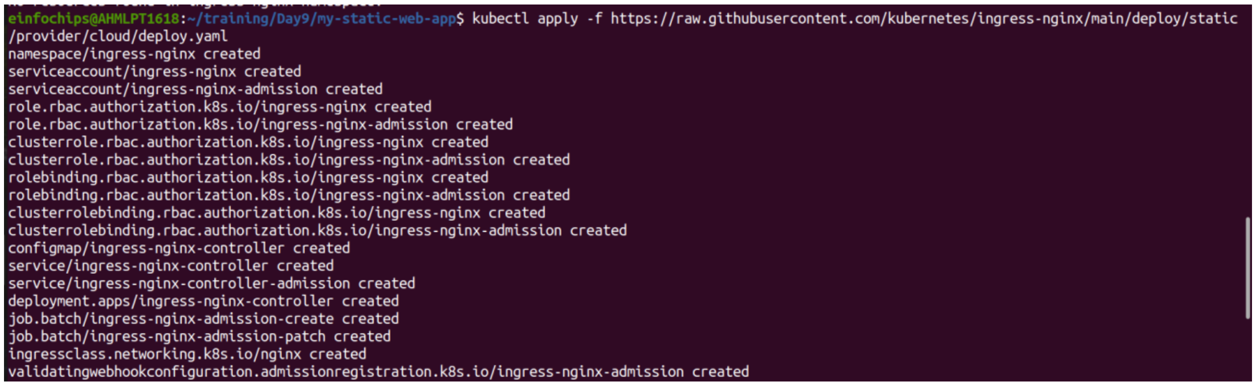

```
einfochips@AHMLPT1618:~/training/Day9/my-static-web-app$ kubectl get svc
NAME                     TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
kubernetes               ClusterIP   10.96.0.1        <none>        443/TCP        3d6h
my-static-web-app        NodePort    10.104.35.104    <none>        80:30186/TCP   2m9s
nodejs-service           ClusterIP   10.101.144.206   <none>        80/TCP         2d5h
nodejs-service-nodeport  NodePort    10.98.160.1      <none>        80:30001/TCP   2d5h
webapp                   NodePort    10.106.21.111    <none>        80:30915/TCP   3d6h
```



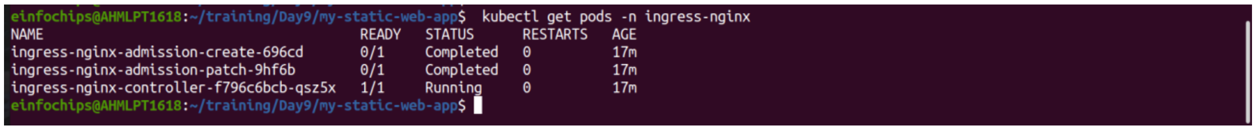
```
einfochips@AHMLPT1618:~/training/Day9/my-static-web-app$ kubectl get pods
NAME                                READY   STATUS             RESTARTS      AGE
backend-5cf7cf7d5c-6cnms            0/1     ImagePullBackOff   0             3d5h
backend-5cf7cf7d5c-k2tv6            0/1     ImagePullBackOff   0             3d5h
db-99c49d8c6-l5wqm                  1/1     Running            3 (35m ago)   3d5h
frontend-76dc6978c-bl49v            1/1     Running            3 (35m ago)   3d5h
frontend-76dc6978c-t84hq            1/1     Running            3 (35m ago)   3d5h
my-static-web-app-766fdd649d-79667  1/1     Running            0             12s
nodejs-app-57cfc566fb-f47h4         0/1     ImagePullBackOff   0             2d5h
nodejs-app-867f6c98ff-bdbm9         0/1     ImagePullBackOff   0             2d5h
nodejs-app-867f6c98ff-xd77j         0/1     ImagePullBackOff   0             2d5h
webapp-ff7d56d67-b9xbb              1/1     Running            6 (35m ago)   3d6h
webapp-ff7d56d67-jt78q              1/1     Running            6 (35m ago)   3d6h
webapp-ff7d56d67-nhctb              1/1     Running            6 (35m ago)   3d6h
```

Stage 2: Configuring Ingress Networking

4. **Install and Configure Ingress Controller:**
   - Install an ingress controller (e.g., Nginx Ingress Controller) in the Minikube cluster.
   - Verify the ingress controller is running and accessible.



```
einfochips@AHMLPT1618:~/training/Day9/my-static-web-app$ kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/main/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
```

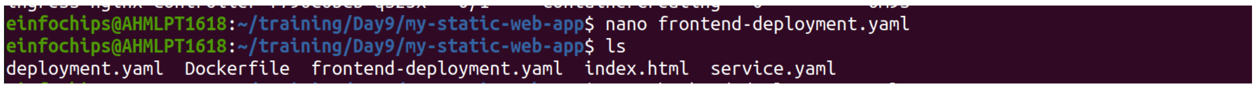


```
einfochips@AHMLPT1618:~/training/Day9/my-static-web-app$  kubectl get pods -n ingress-nginx
NAME                                       READY   STATUS      RESTARTS   AGE
ingress-nginx-admission-create-696cd       0/1     Completed   0          17m
ingress-nginx-admission-patch-9hf6b        0/1     Completed   0          17m
ingress-nginx-controller-f796c6bcb-qsz5x   1/1     Running     0          17m
einfochips@AHMLPT1618:~/training/Day9/my-static-web-app$ 
```

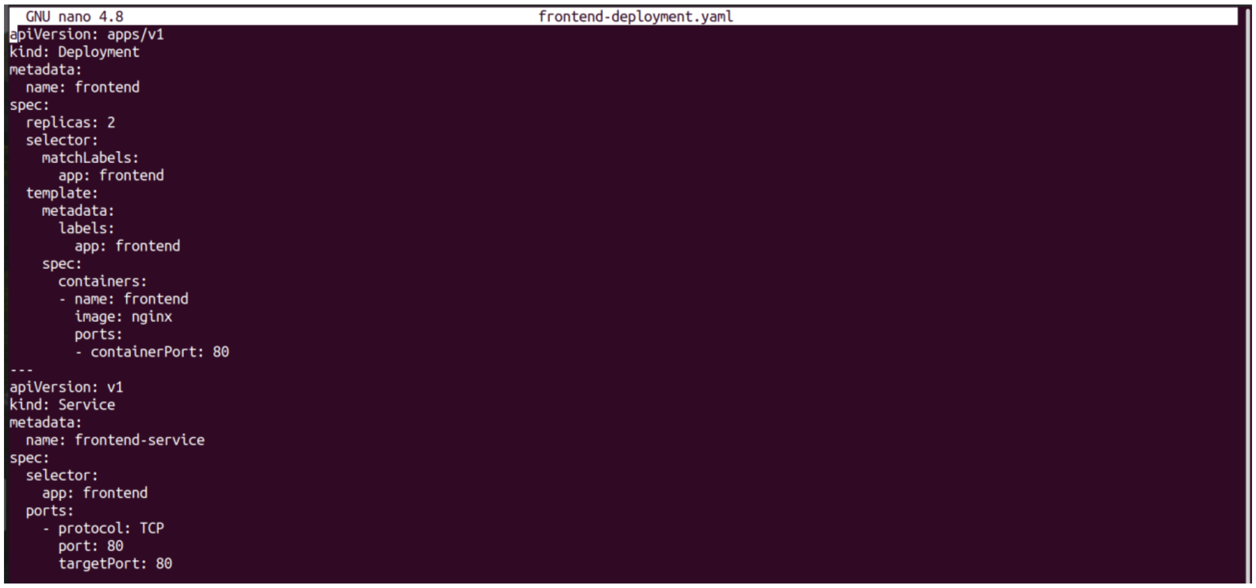
4. **Create Ingress Resource:**
   - Write an ingress resource manifest to route external traffic to the static web application.
   - Configure advanced ingress rules for path-based routing and host-based routing (use at least two different hostnames and paths).
   - Implement TLS termination for secure connections.
   - Configure URL rewriting in the ingress resource to modify incoming URLs before they reach the backend services.
   - Enable sticky sessions to ensure that requests from the same client are directed to the same backend pod.

Deliverables:

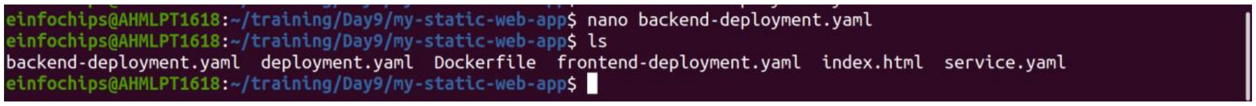
- Ingress controller installation commands/scripts
- Ingress resource YAML file with advanced routing, TLS configuration, URL rewriting, and sticky sessions



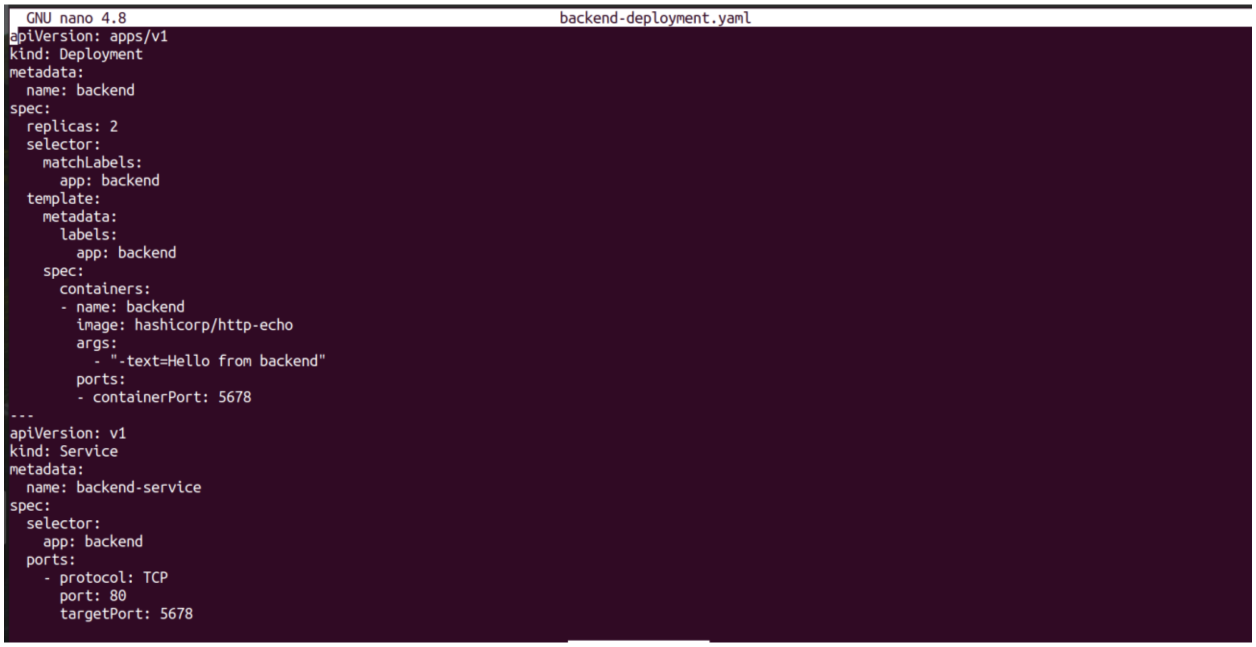
```
einfochips@AHMLPT1618:~/training/Day9/my-static-web-app$ nano frontend-deployment.yaml
einfochips@AHMLPT1618:~/training/Day9/my-static-web-app$ ls
deployment.yaml  Dockerfile  frontend-deployment.yaml  index.html  service.yaml
```



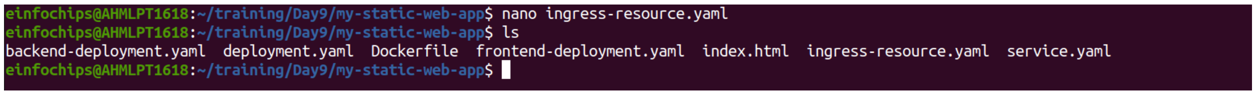
```
  GNU nano 4.8                                     frontend-deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: frontend
spec:
  replicas: 2
  selector:
    matchLabels:
      app: frontend
  template:
    metadata:
      labels:
        app: frontend
    spec:
      containers:
      - name: frontend
        image: nginx
        ports:
        - containerPort: 80
---
apiVersion: v1
kind: Service
metadata:
  name: frontend-service
spec:
  selector:
    app: frontend
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
```



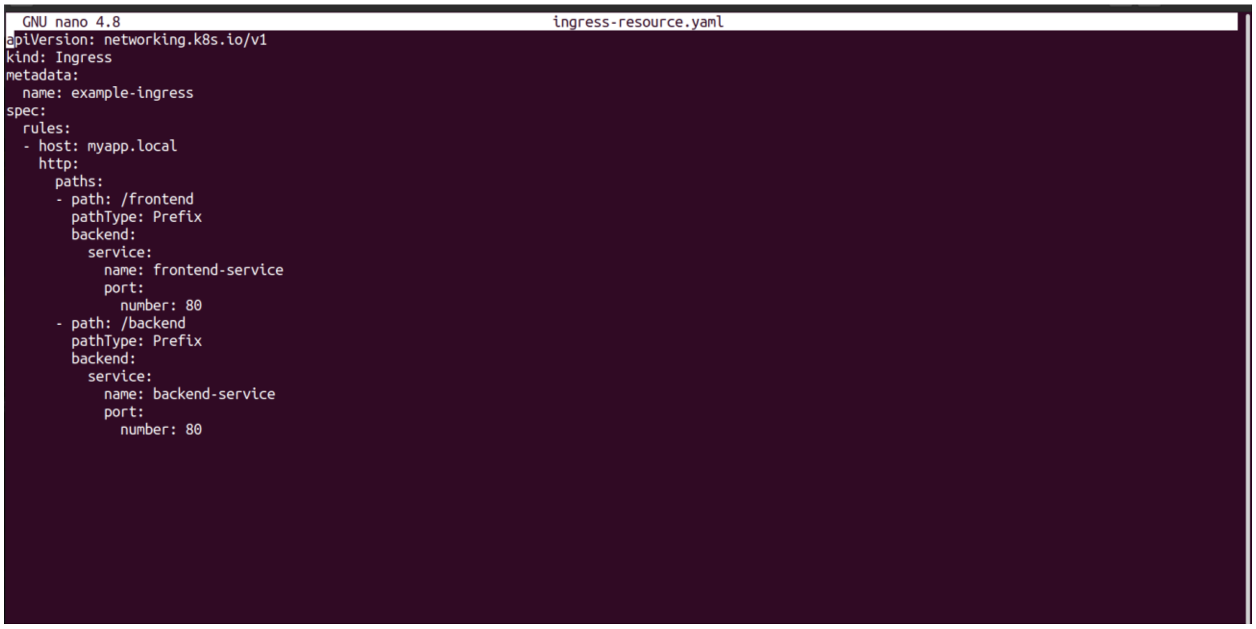
```
einfochips@AHMLPT1618:~/training/Day9/my-static-web-app$ nano backend-deployment.yaml
einfochips@AHMLPT1618:~/training/Day9/my-static-web-app$ ls
backend-deployment.yaml  deployment.yaml  Dockerfile  frontend-deployment.yaml  index.html  service.yaml
einfochips@AHMLPT1618:~/training/Day9/my-static-web-app$ 
```



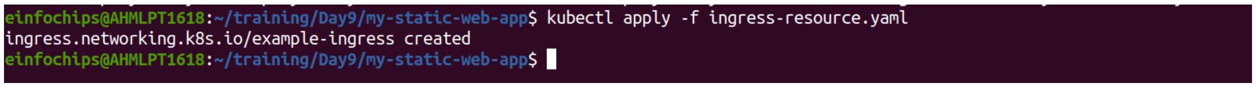
```
  GNU nano 4.8                                     backend-deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: backend
spec:
  replicas: 2
  selector:
    matchLabels:
      app: backend
  template:
    metadata:
      labels:
        app: backend
    spec:
      containers:
      - name: backend
        image: hashicorp/http-echo
        args:
          - "-text=Hello from backend"
        ports:
        - containerPort: 5678
---
apiVersion: v1
kind: Service
metadata:
  name: backend-service
spec:
  selector:
    app: backend
  ports:
    - protocol: TCP
      port: 80
      targetPort: 5678
```



```
einfochips@AHMLPT1618:~/training/Day9/my-static-web-app$ nano ingress-resource.yaml
einfochips@AHMLPT1618:~/training/Day9/my-static-web-app$ ls
backend-deployment.yaml  deployment.yaml  Dockerfile  frontend-deployment.yaml  index.html  ingress-resource.yaml  service.yaml
einfochips@AHMLPT1618:~/training/Day9/my-static-web-app$
```



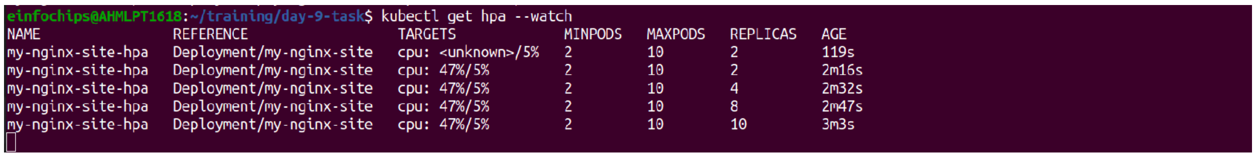
```
  GNU nano 4.8                                     ingress-resource.yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: example-ingress
spec:
  rules:
  - host: myapp.local
    http:
      paths:
      - path: /frontend
        pathType: Prefix
        backend:
          service:
            name: frontend-service
            port:
              number: 80
      - path: /backend
        pathType: Prefix
        backend:
          service:
            name: backend-service
            port:
              number: 80
```



```
einfochips@AHMLPT1618:~/training/Day9/my-static-web-app$ kubectl apply -f ingress-resource.yaml
ingress.networking.k8s.io/example-ingress created
einfochips@AHMLPT1618:~/training/Day9/my-static-web-app$
```

Stage 3: Implementing Horizontal Pod Autoscaling

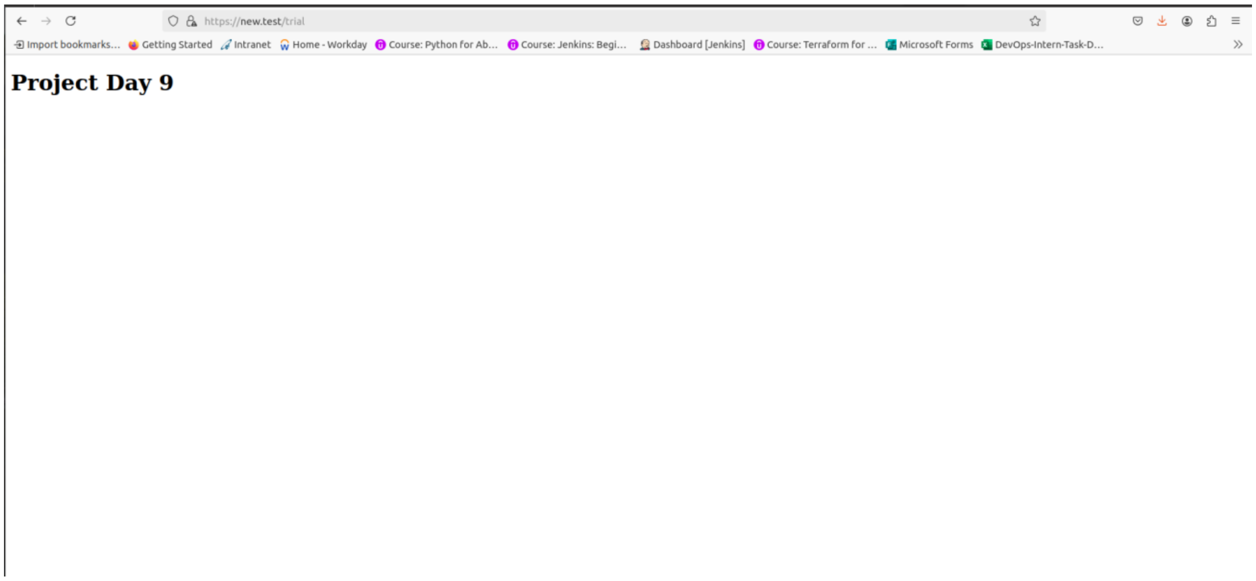
6. **Configure Horizontal Pod Autoscaler:**
   - Write a horizontal pod autoscaler (HPA) manifest to automatically scale the static web application pods based on CPU utilization.
   - Set thresholds for minimum and maximum pod replicas.
7. **Stress Testing:**
   - Perform stress testing to simulate traffic and validate the HPA configuration.
   - Monitor the scaling behavior and ensure the application scales up and down based on the load.



```
einfochips@AHMLPT1618:~/training/day-9-task$ kubectl get hpa --watch
NAME               REFERENCE                  TARGETS            MINPODS   MAXPODS   REPLICAS   AGE
my-nginx-site-hpa   Deployment/my-nginx-site   cpu: <unknown>/5%   2         10        2          119s
my-nginx-site-hpa   Deployment/my-nginx-site   cpu: 47%/5%         2         10        2          2m16s
my-nginx-site-hpa   Deployment/my-nginx-site   cpu: 47%/5%         2         10        4          2m32s
my-nginx-site-hpa   Deployment/my-nginx-site   cpu: 47%/5%         2         10        8          2m47s
my-nginx-site-hpa   Deployment/my-nginx-site   cpu: 47%/5%         2         10        10         3m3s
```

Stage 4: Final Validation and Cleanup

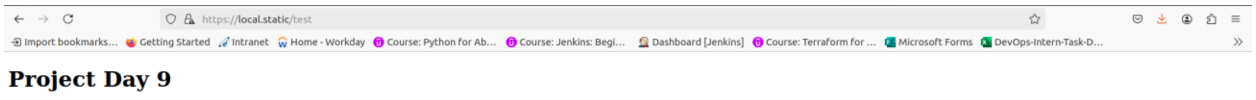
8. **Final Validation:**
   - Validate the ingress networking, URL rewriting, and sticky sessions configurations by accessing the web application through different hostnames and paths.
   - Verify the application's availability and performance during different load conditions.
9. **Cleanup:**
   - Provide commands or scripts to clean up the Kubernetes resources created during the project (deployments, services, ingress, HPA).



https://new.test/trial

Import bookmarks... Getting Started Intranet Home - Workday Course: Python for Ab... Course: Jenkins: Begi... Dashboard [Jenkins] Course: Terraform for ... Microsoft Forms DevOps-Intern-Task-D...

Project Day 9



Project Day 9