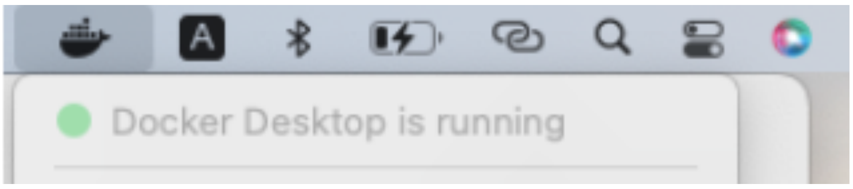
README

Challenge Steps

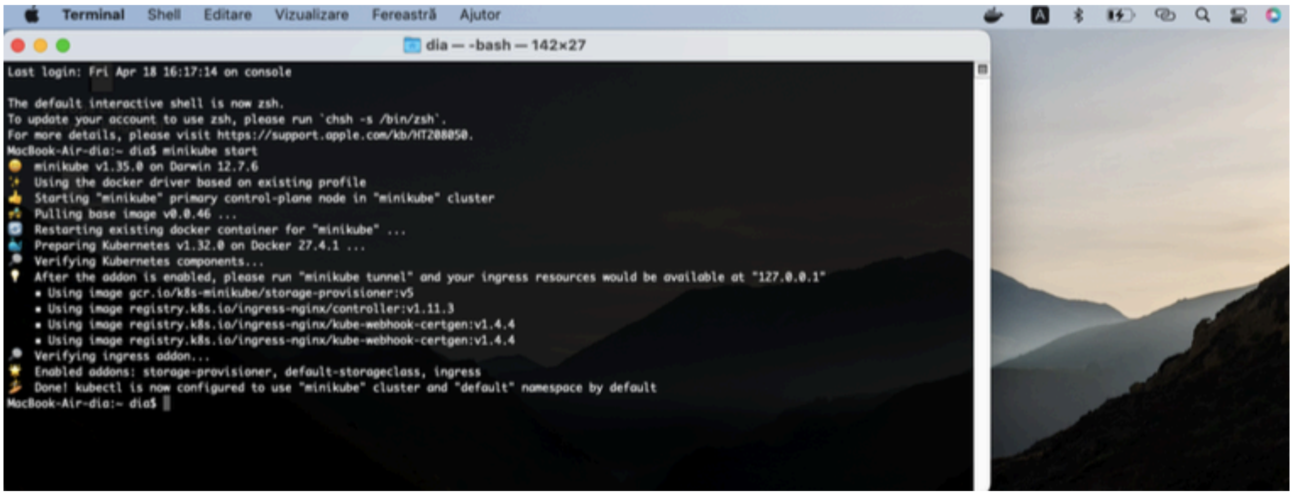
1. Complete the Kubernetes Manifest with an Ingress Controller

• Start Docker Desktop and ensure it is running.

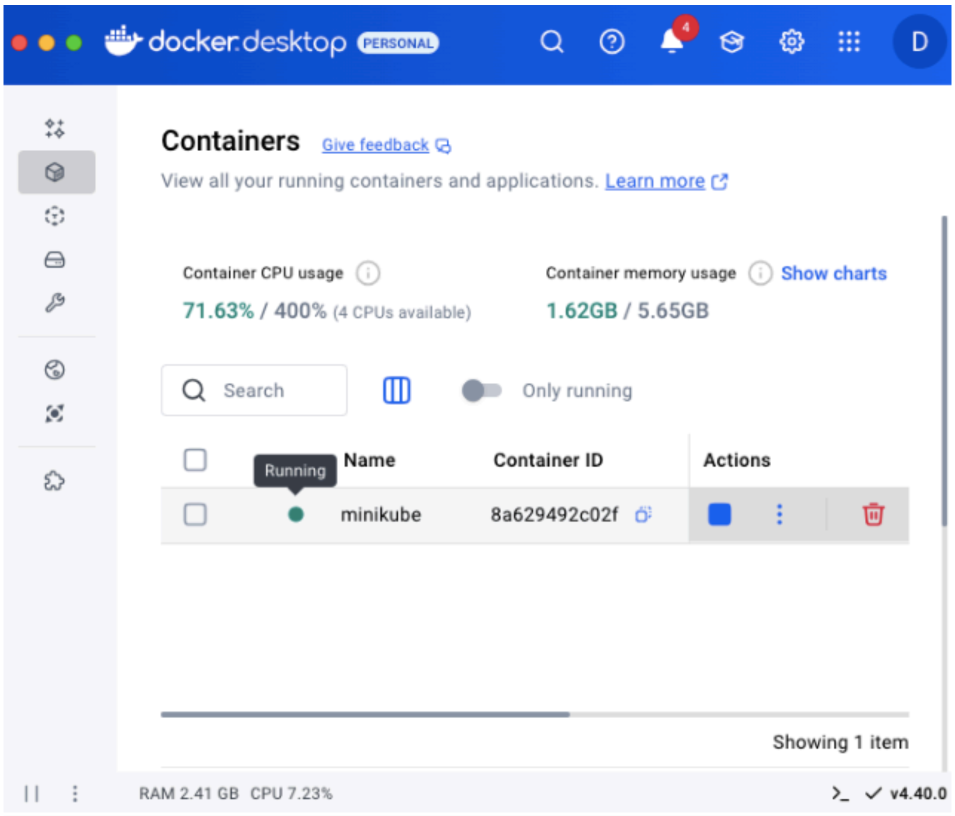


• Launch a local Kubernetes cluster using Minikube:

minikube start



```
Last login: Fri Apr 18 16:17:14 on console

The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
MacBook-Air-dia:~ dia$ minikube start
😄  minikube v1.35.0 on Darwin 12.7.6
✨  Using the docker driver based on existing profile
👍  Starting "minikube" primary control-plane node in "minikube" cluster
🚜  Pulling base image v0.0.46 ...
🔄  Restarting existing docker container for "minikube" ...
🐳  Preparing Kubernetes v1.32.0 on Docker 27.4.1 ...
🔎  Verifying Kubernetes components...
💡  After the addon is enabled, please run "minikube tunnel" and your ingress resources would be available at "127.0.0.1"
    ▪ Using image gcr.io/k8s-minikube/storage-provisioner:v5
    ▪ Using image registry.k8s.io/ingress-nginx/controller:v1.11.3
    ▪ Using image registry.k8s.io/ingress-nginx/kube-webhook-certgen:v1.4.4
    ▪ Using image registry.k8s.io/ingress-nginx/kube-webhook-certgen:v1.4.4
🔎  Verifying ingress addon...
🌟  Enabled addons: storage-provisioner, default-storageclass, ingress
🏄  Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
MacBook-Air-dia:~ dia$
```



Containers

View all your running containers and applications.

Container CPU usage 71.63% / 400% (4 CPUs available)

Container memory usage 1.62GB / 5.65GB

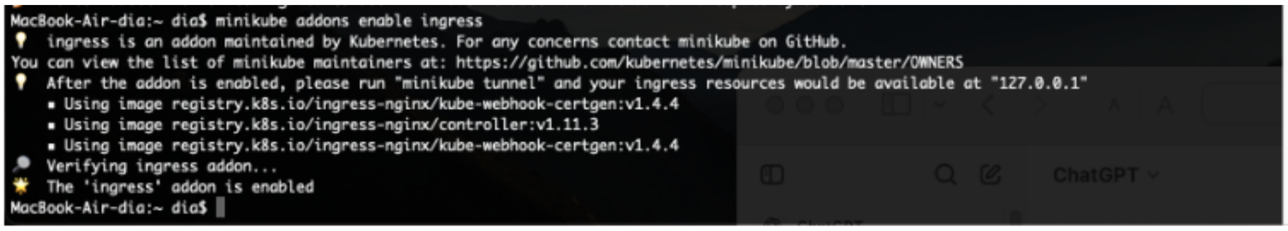
	Name	Container ID	Actions
●	minikube	8a629492c02f	

———————————————————————————————————

- We use an Ingress Controller to access applications through a custom domain (e.g. echo.local), we need to enable it again:

- Enable the Ingress Controller in Minikube:

minikube addons enable ingress



```
MacBook-Air-dia:~ dia$ minikube addons enable ingress
💡  ingress is an addon maintained by Kubernetes. For any concerns contact minikube on GitHub.
You can view the list of minikube maintainers at: https://github.com/kubernetes/minikube/blob/master/OWNERS
💡  After the addon is enabled, please run "minikube tunnel" and your ingress resources would be available at "127.0.0.1"
    ▪ Using image registry.k8s.io/ingress-nginx/kube-webhook-certgen:v1.4.4
    ▪ Using image registry.k8s.io/ingress-nginx/controller:v1.11.3
    ▪ Using image registry.k8s.io/ingress-nginx/kube-webhook-certgen:v1.4.4
🔎  Verifying ingress addon...
🌟  The 'ingress' addon is enabled
MacBook-Air-dia:~ dia$ 
```

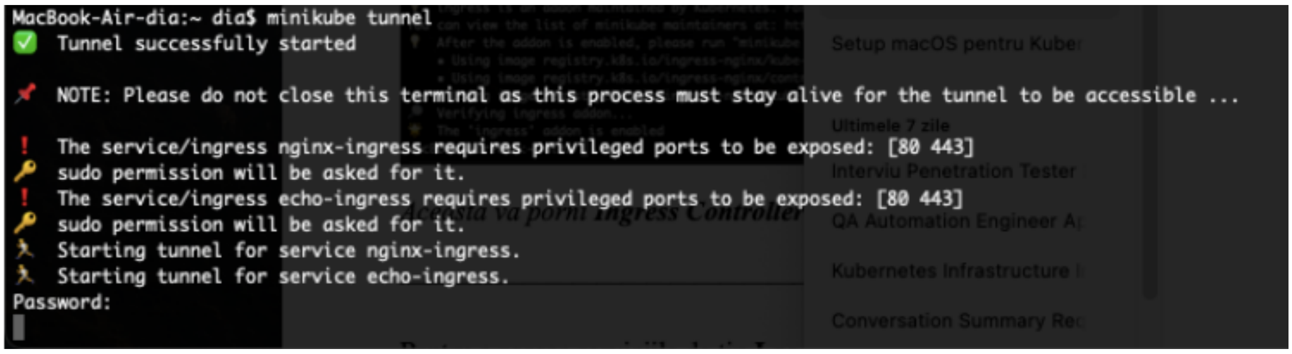
- This will start the Ingress Controller to allow us access to applications through the defined domain.

———————————————————————————————————

- To access LoadBalancer services (like ingress-nginx-controller), we need to restart the tunnel. This creates a network tunnel between Minikube and our localhost.

- Start a Minikube tunnel to allow access to LoadBalancer services:

minikube tunnel



```
MacBook-Air-dia:~ dia$ minikube tunnel
✅  Tunnel successfully started

📌  NOTE: Please do not close this terminal as this process must stay alive for the tunnel to be accessible ...

❗  The service/ingress nginx-ingress requires privileged ports to be exposed: [80 443]
🔑  sudo permission will be asked for it.
❗  The service/ingress echo-ingress requires privileged ports to be exposed: [80 443]
🔑  sudo permission will be asked for it.
🏃  Starting tunnel for service nginx-ingress.
🏃  Starting tunnel for service echo-ingress.
Password:
```

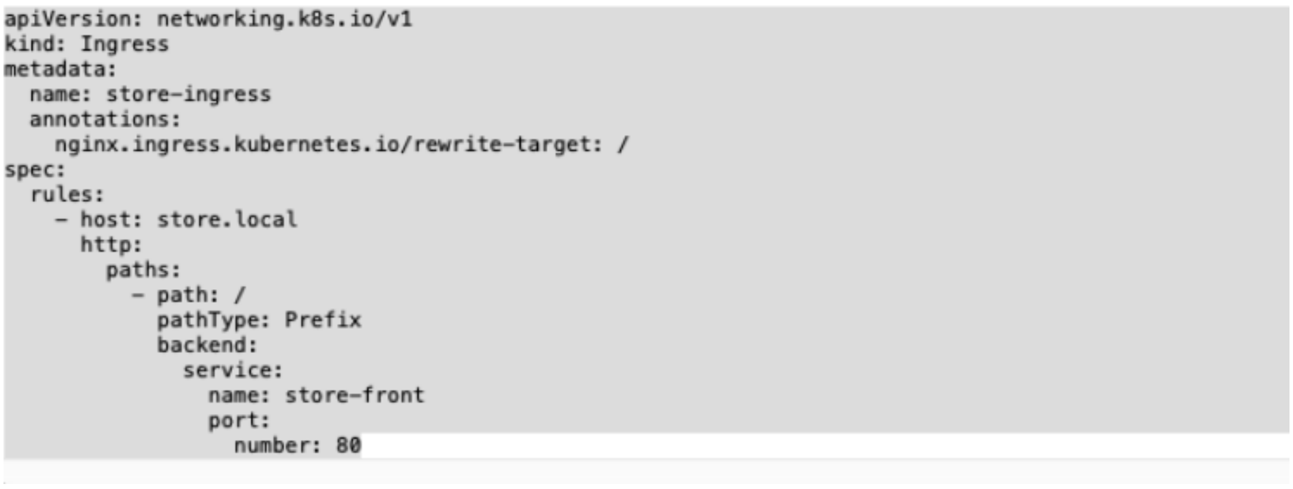
We leave the terminal open, this process will remain active and allow local access to our applications.

———————————————————————————————————

- Clone the official application repository:

git clone https://github.com/Azure-Samples/aks-store-demo.git
cd aks-store-demo

———————————————————————————————————



```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: store-ingress
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  rules:
    - host: store.local
      http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: store-front
                port:
                  number: 80
```

- Add an Ingress resource to the bottom of the `aks-store-quickstart.yaml` file to expose the frontend service (`store-front`) via a custom domain (`store.local`).

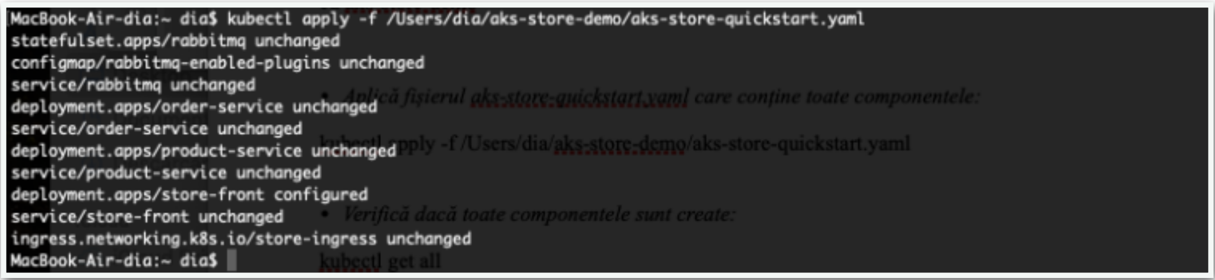
What this Ingress does:

- Creates a rule for the store.local domain
- Redirects all HTTP requests to the store-front service (port 80)
- Allows external access to the application through a single entry point Requires the Ingress
- Controller to be enabled in the cluster

————————————————————————————————————

- Apply the Kubernetes manifest:

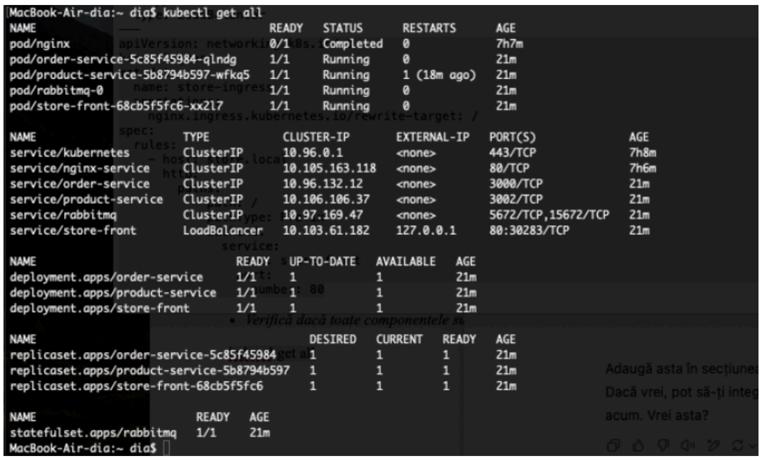
kubectl apply -f /path/to/aks-store-quickstart.yaml



```
MacBook-Air-dia:~ dia$ kubectl apply -f /Users/dia/aks-store-demo/aks-store-quickstart.yaml
statefulset.apps/rabbitmq unchanged
configmap/rabbitmq-enabled-plugins unchanged
service/rabbitmq unchanged
deployment.apps/order-service unchanged
service/order-service unchanged
deployment.apps/product-service unchanged
service/product-service unchanged
deployment.apps/store-front configured
service/store-front unchanged
ingress.networking.k8s.io/store-ingress unchanged
MacBook-Air-dia:~ dia$
```

- Verify all resources are running:

kubectl get all



```
MacBook-Air-dia:~ dia$ kubectl get all
NAME                                   READY   STATUS      RESTARTS       AGE
pod/nginx                              0/1     Completed   0              7h7m
pod/order-service-5c85f45984-qlndg     1/1     Running     0              21m
pod/product-service-5b8794b597-wfkq5   1/1     Running     1 (18m ago)    21m
pod/rabbitmq-0                         1/1     Running     0              21m
pod/store-front-68cb5f5fc6-xx2l7       1/1     Running     0              21m

NAME                      TYPE           CLUSTER-IP       EXTERNAL-IP   PORT(S)              AGE
service/kubernetes        ClusterIP      10.96.0.1        <none>        443/TCP              7h8m
service/nginx-service     ClusterIP      10.105.163.118   <none>        80/TCP               7h6m
service/order-service     ClusterIP      10.96.132.12     <none>        3000/TCP             21m
service/product-service   ClusterIP      10.106.106.37    <none>        3002/TCP             21m
service/rabbitmq          ClusterIP      10.97.169.47     <none>        5672/TCP,15672/TCP   21m
service/store-front       LoadBalancer   10.103.61.182    127.0.0.1     80:30283/TCP         21m

NAME                              READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/order-service     1/1     1            1           21m
deployment.apps/product-service   1/1     1            1           21m
deployment.apps/store-front       1/1     1            1           21m

NAME                                         DESIRED   CURRENT   READY   AGE
replicaset.apps/order-service-5c85f45984     1         1         1       21m
replicaset.apps/product-service-5b8794b597   1         1         1       21m
replicaset.apps/store-front-68cb5f5fc6       1         1         1       21m

NAME                        READY   AGE
statefulset.apps/rabbitmq   1/1     21m
MacBook-Air-dia:~ dia$
```

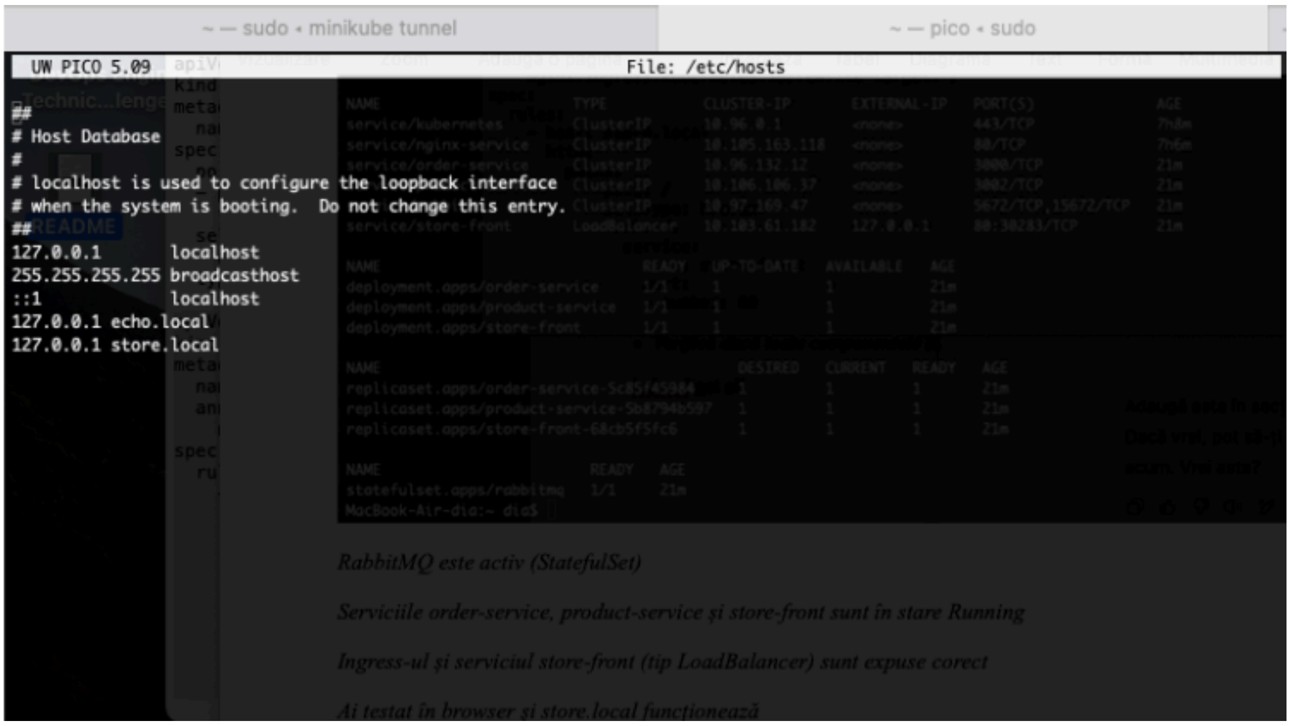
RabbitMQ is active (StatefulSet)
The order-service, product-service and store-front services are in Running state.
The ingress and store-front service (LoadBalancer type) are exposed correctly
You tested in the browser and store.local works

- Edit your /etc/hosts file to map store.local

sudo nano /etc/hosts

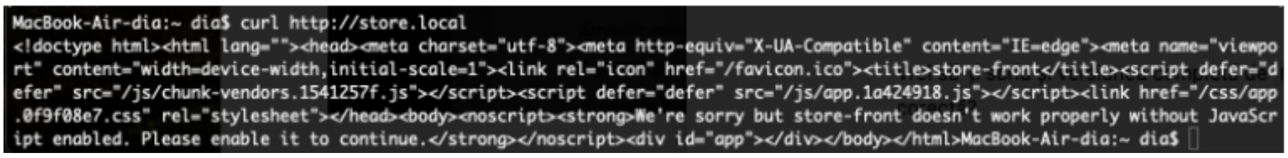
- Add the following line:

127.0.0.1 store.local



```
UW PICO 5.09                                File: /etc/hosts

##
# Host Database
#
# localhost is used to configure the loopback interface
# when the system is booting.  Do not change this entry.
##
127.0.0.1       localhost
255.255.255.255 broadcasthost
::1             localhost
127.0.0.1 echo.local
127.0.0.1 store.local
```

Test the application:
curl http://store.local



```
MacBook-Air-dia:~ dia$ curl http://store.local
<!doctype html><html lang=""><head><meta charset="utf-8"><meta http-equiv="X-UA-Compatible" content="IE=edge"><meta name="viewpo
rt" content="width=device-width,initial-scale=1"><link rel="icon" href="/favicon.ico"><title>store-front</title><script defer="d
efer" src="/js/chunk-vendors.1541257f.js"></script><script defer="defer" src="/js/app.1a424918.js"></script><link href="/css/app
.0f9f08e7.css" rel="stylesheet"></head><body><noscript><strong>We're sorry but store-front doesn't work properly without JavaScr
ipt enabled. Please enable it to continue.</strong></noscript><div id="app"></div></body></html>MacBook-Air-dia:~ dia$
```

This response confirms that:
- The Ingress Controller is up and running
- The store.local domain is correctly configured in the /etc/hosts file
- The store-front application is exposed and responding via Ingress
- All setup from Step 1 is functional

The application can also be accessed from the browser at: http://store.local

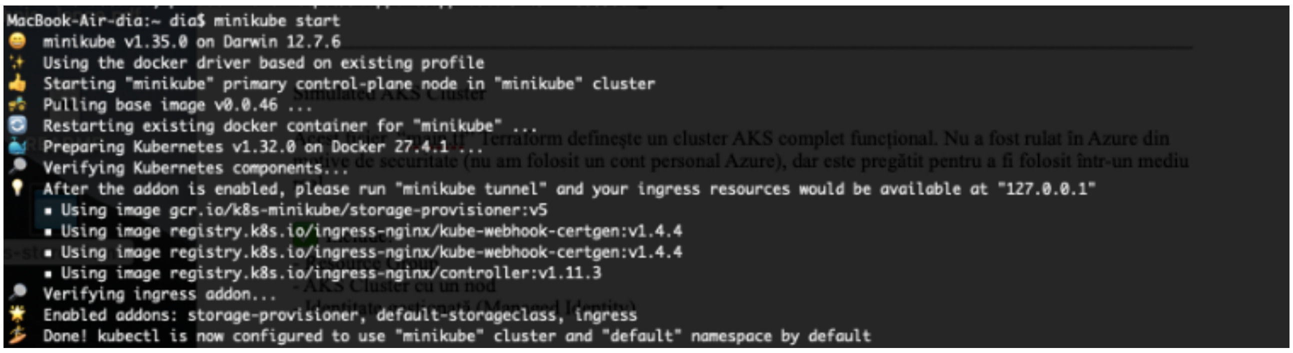
2. Create Kubernetes Cluster using Terraform

- A `main.tf` file is provided inside the `terraform/` directory, prepared to create an AKS cluster in Azure.
  Due to security reasons, an actual Azure account was not used during testing — all tests were performed locally using Minikube.
- main.tf includes:
  - Resource Group creation
  - AKS Cluster with a single node
  - Managed Identity configuration

Local Validation Steps (with Minikube):

- Start Minikube:

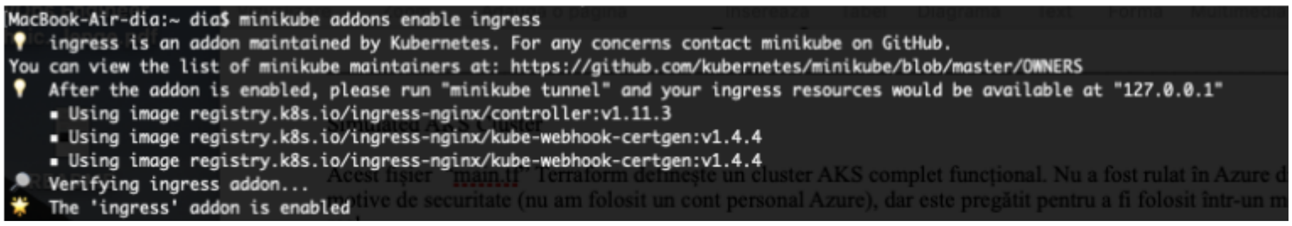
minikube start



```
MacBook-Air-dia:~ dia$ minikube start
😄  minikube v1.35.0 on Darwin 12.7.6
✨  Using the docker driver based on existing profile
👍  Starting "minikube" primary control-plane node in "minikube" cluster
🚜  Pulling base image v0.0.46 ...
🔄  Restarting existing docker container for "minikube" ...
🐳  Preparing Kubernetes v1.32.0 on Docker 27.4.1 ...
🔎  Verifying Kubernetes components...
💡  After the addon is enabled, please run "minikube tunnel" and your ingress resources would be available at "127.0.0.1"
    ▪ Using image gcr.io/k8s-minikube/storage-provisioner:v5
    ▪ Using image registry.k8s.io/ingress-nginx/kube-webhook-certgen:v1.4.4
    ▪ Using image registry.k8s.io/ingress-nginx/kube-webhook-certgen:v1.4.4
    ▪ Using image registry.k8s.io/ingress-nginx/controller:v1.11.3
🔎  Verifying ingress addon...
🌟  Enabled addons: storage-provisioner, default-storageclass, ingress
🏄  Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
```

- Enable Ingress Controller:

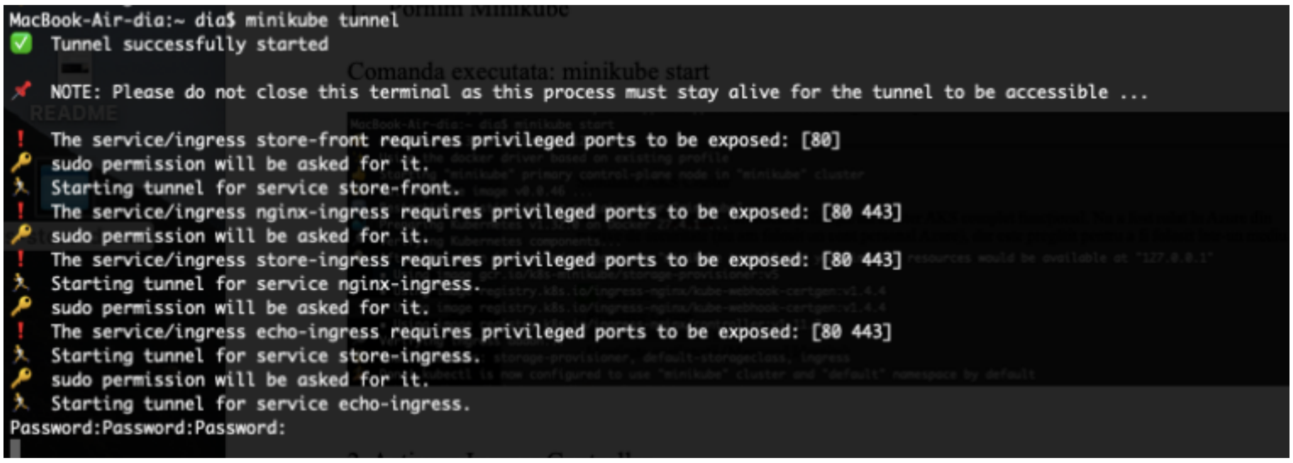
minikube addons enable ingress



```
MacBook-Air-dia:~ dia$ minikube addons enable ingress
💡  ingress is an addon maintained by Kubernetes. For any concerns contact minikube on GitHub.
You can view the list of minikube maintainers at: https://github.com/kubernetes/minikube/blob/master/OWNERS
💡  After the addon is enabled, please run "minikube tunnel" and your ingress resources would be available at "127.0.0.1"
    ▪ Using image registry.k8s.io/ingress-nginx/controller:v1.11.3
    ▪ Using image registry.k8s.io/ingress-nginx/kube-webhook-certgen:v1.4.4
    ▪ Using image registry.k8s.io/ingress-nginx/kube-webhook-certgen:v1.4.4
🔎  Verifying ingress addon...
🌟  The 'ingress' addon is enabled
```

- Start the Minikube tunnel:

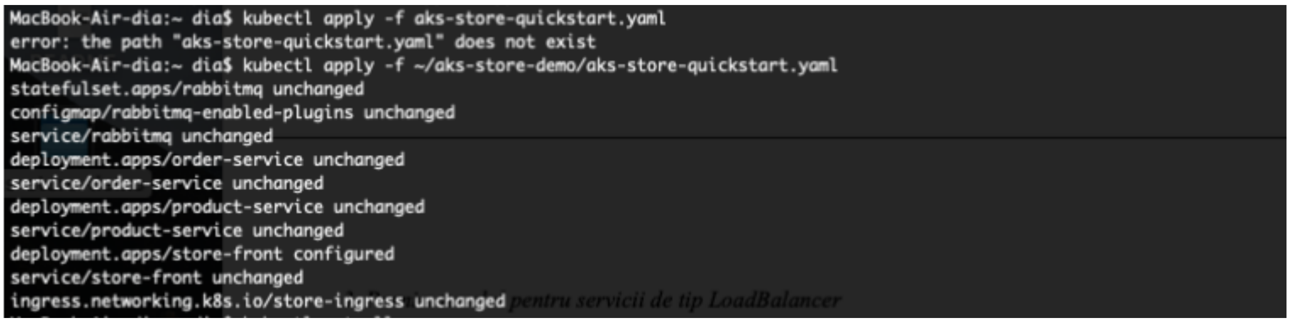
minikube tunnel



```
MacBook-Air-dia:~ dia$ minikube tunnel
✅  Tunnel successfully started

📌  NOTE: Please do not close this terminal as this process must stay alive for the tunnel to be accessible ...

❗  The service/ingress store-front requires privileged ports to be exposed: [80]
🔑  sudo permission will be asked for it.
🏃  Starting tunnel for service store-front.
❗  The service/ingress nginx-ingress requires privileged ports to be exposed: [80 443]
🔑  sudo permission will be asked for it.
❗  The service/ingress store-ingress requires privileged ports to be exposed: [80 443]
🏃  Starting tunnel for service nginx-ingress.
🔑  sudo permission will be asked for it.
❗  The service/ingress echo-ingress requires privileged ports to be exposed: [80 443]
🏃  Starting tunnel for service store-ingress.
🔑  sudo permission will be asked for it.
🏃  Starting tunnel for service echo-ingress.
Password:Password:Password:
```

- Apply the Kubernetes manifests:

kubectl apply -f aks-store-quickstart.yaml

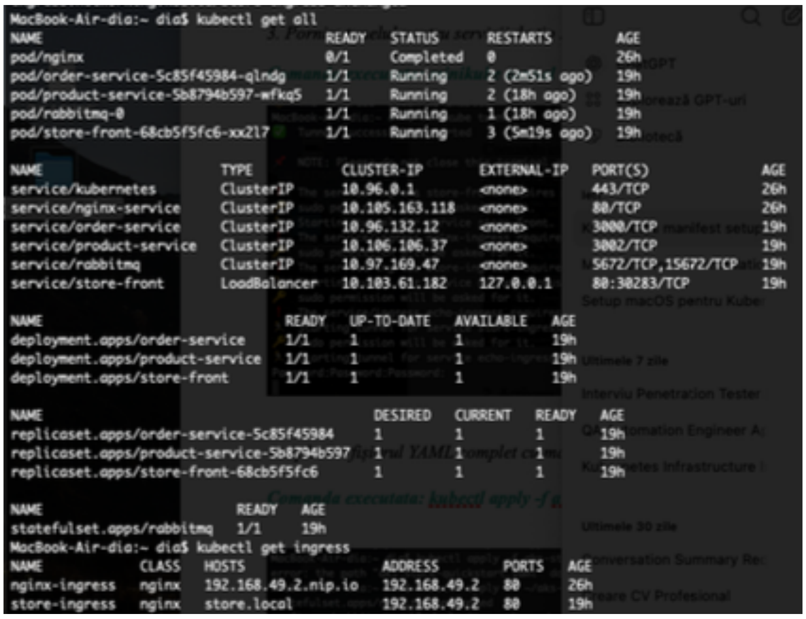


```
MacBook-Air-dia:~ dia$ kubectl apply -f aks-store-quickstart.yaml
error: the path "aks-store-quickstart.yaml" does not exist
MacBook-Air-dia:~ dia$ kubectl apply -f ~/aks-store-demo/aks-store-quickstart.yaml
statefulset.apps/rabbitmq unchanged
configmap/rabbitmq-enabled-plugins unchanged
service/rabbitmq unchanged
deployment.apps/order-service unchanged
service/order-service unchanged
deployment.apps/product-service unchanged
service/product-service unchanged
deployment.apps/store-front configured
service/store-front unchanged
ingress.networking.k8s.io/store-ingress unchanged
```

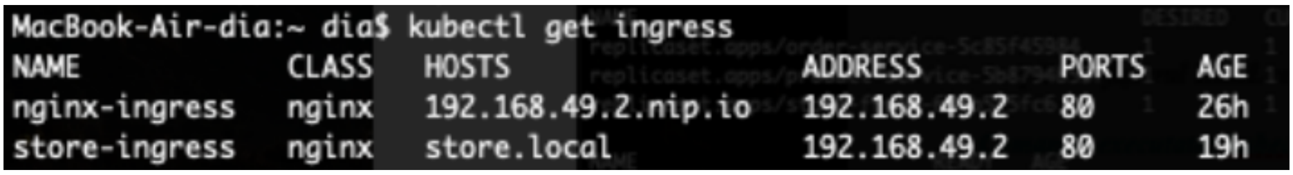
- Verify all resources:

kubectl get all

kubectl get ingress



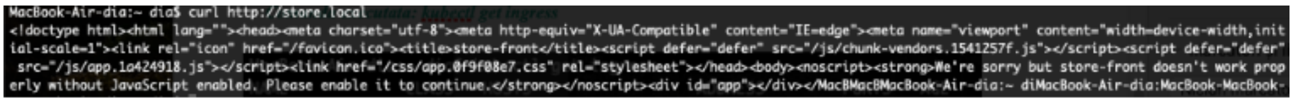
```
MacBook-Air-dia:~ dia$ kubectl get all
NAME                                   READY   STATUS      RESTARTS        AGE
pod/nginx                              0/1     Completed   0               26h
pod/order-service-5c85f45984-qlndg     1/1     Running     2 (2m51s ago)   19h
pod/product-service-5b8794b597-wfkq5   1/1     Running     2 (18h ago)     19h
pod/rabbitmq-0                         1/1     Running     1 (18h ago)     19h
pod/store-front-68cb5f5fc6-xx2l7       1/1     Running     3 (5m19s ago)   19h

NAME                      TYPE           CLUSTER-IP       EXTERNAL-IP   PORT(S)                AGE
service/kubernetes        ClusterIP      10.96.0.1        <none>        443/TCP                26h
service/nginx-service     ClusterIP      10.105.163.118   <none>        80/TCP                 26h
service/order-service     ClusterIP      10.96.132.12     <none>        3000/TCP               19h
service/product-service   ClusterIP      10.106.106.37    <none>        3002/TCP               19h
service/rabbitmq          ClusterIP      10.97.169.47     <none>        5672/TCP,15672/TCP     19h
service/store-front       LoadBalancer   10.103.61.182    127.0.0.1     80:30283/TCP           19h

NAME                              READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/order-service     1/1     1            1           19h
deployment.apps/product-service   1/1     1            1           19h
deployment.apps/store-front       1/1     1            1           19h

NAME                                         DESIRED   CURRENT   READY   AGE
replicaset.apps/order-service-5c85f45984     1         1         1       19h
replicaset.apps/product-service-5b8794b597   1         1         1       19h
replicaset.apps/store-front-68cb5f5fc6       1         1         1       19h

NAME                        READY   AGE
statefulset.apps/rabbitmq   1/1     19h
MacBook-Air-dia:~ dia$ kubectl get ingress
NAME            CLASS   HOSTS                ADDRESS        PORTS   AGE
nginx-ingress   nginx   192.168.49.2.nip.io  192.168.49.2   80      26h
store-ingress   nginx   store.local          192.168.49.2   80      19h
```



```
MacBook-Air-dia:~ dia$ kubectl get ingress
NAME            CLASS   HOSTS                 ADDRESS        PORTS   AGE
nginx-ingress   nginx   192.168.49.2.nip.io   192.168.49.2   80      26h
store-ingress   nginx   store.local           192.168.49.2   80      19h
```

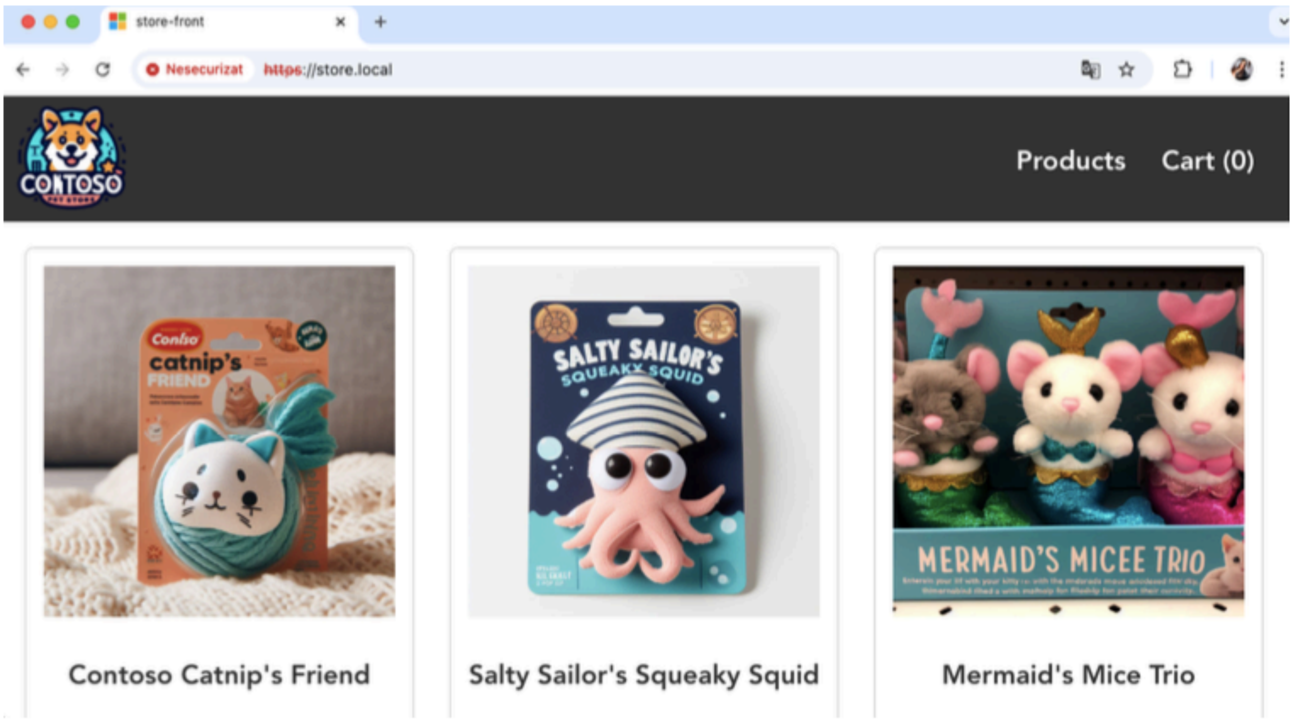
- Access the application:

curl http://store.local



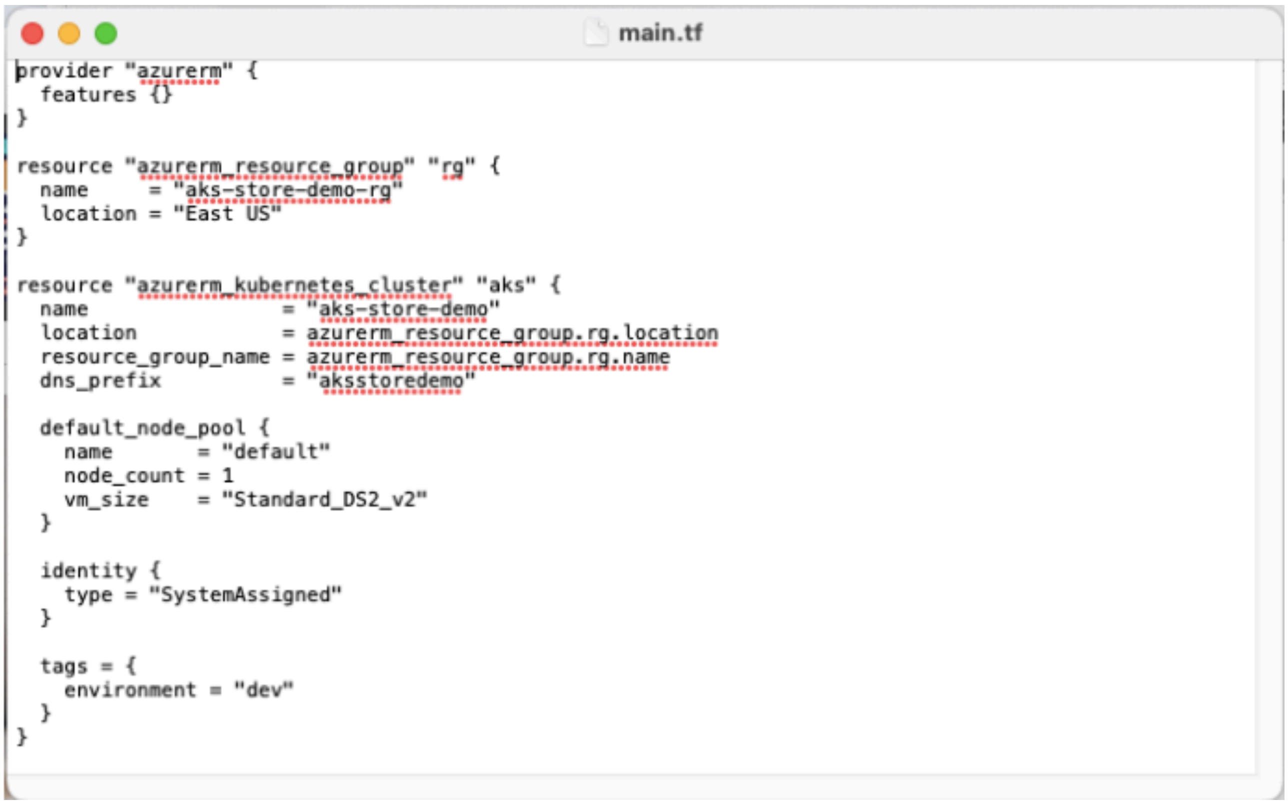
```
MacBook-Air-dia:~ dia$ curl http://store.local
<!doctype html><html lang=""><head><meta charset="utf-8"><meta http-equiv="X-UA-Compatible" content="IE=edge"><meta name="viewport" content="width=device-width,init
ial-scale=1"><link rel="icon" href="/favicon.ico"><title>store-front</title><script defer="defer" src="/js/chunk-vendors.1541257f.js"></script><script defer="defer"
 src="/js/app.1a424918.js"></script><link href="/css/app.0f9f08e7.css" rel="stylesheet"></head><body><noscript><strong>We're sorry but store-front doesn't work prop
erly without JavaScript enabled. Please enable it to continue.</strong></noscript><div id="app"></div></MacBMacBMacBook-Air-dia:~ diMacBook-Air-dia:MacBook-MacBook-
```

The application can also be accessed from the browser at: http://store.local



Ingress Controller is working
The store.local domain is correctly configured
The store-front application is exposed correctly
The local Kubernetes setup is 100% working

Content main.tf



```
provider "azurerm" {
  features {}
}

resource "azurerm_resource_group" "rg" {
  name     = "aks-store-demo-rg"
  location = "East US"
}

resource "azurerm_kubernetes_cluster" "aks" {
  name                = "aks-store-demo"
  location            = azurerm_resource_group.rg.location
  resource_group_name = azurerm_resource_group.rg.name
  dns_prefix          = "aksstoredemo"

  default_node_pool {
    name       = "default"
    node_count = 1
    vm_size    = "Standard_DS2_v2"
  }

  identity {
    type = "SystemAssigned"
  }

  tags = {
    environment = "dev"
  }
}
```

The code is related to setting up a Kubernetes cluster on Azure (AKS) with Terraform and configuring an Ingress controller to handle HTTP/HTTPS traffic within the application.

3. Create CI/CD for the Project

- Dockerfiles were created for each service and can be found in the `docker/` directory.
- Kubernetes YAML files for each service were created under `k8s/`.

Building and Deploying Locally:

- Example for the `order-service`:

docker build -t demo-order-service -f docker/Dockerfile.order-service .

kubectl apply -f k8s/order-service.yaml

kubectl get pods

kubectl get svc

- Repeat similarly for the other services.
- Alternatively, use automation scripts:

./build-all.sh

./deploy-all.sh

CI/CD Pipeline:

- Implemented using **GitHub Actions** (`.github/workflows/ci-cd.yml`).
- CI includes:
  - Building Docker images for all services.
  - Running basic service tests (e.g., simple `curl` tests).
- CD includes:
  - Deploying all services automatically into Minikube.
  - Using Kubernetes manifests and Ingress for service exposure.
- Local CI testing was performed using [`act`](act).

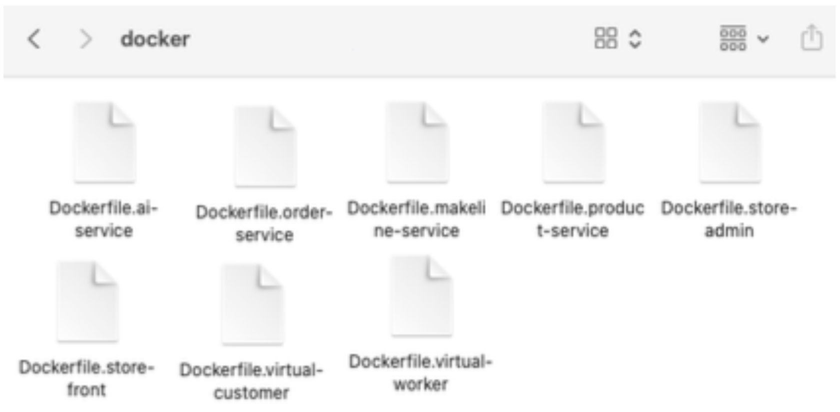
Commands:

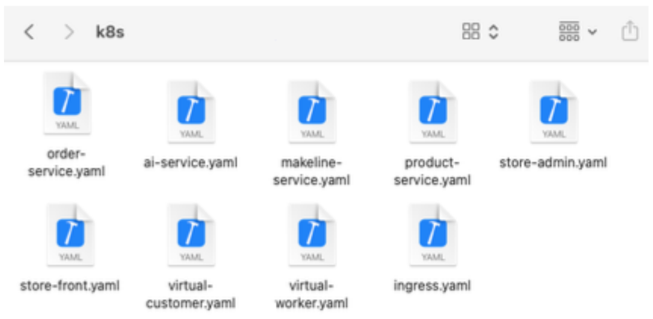
./build-all.sh

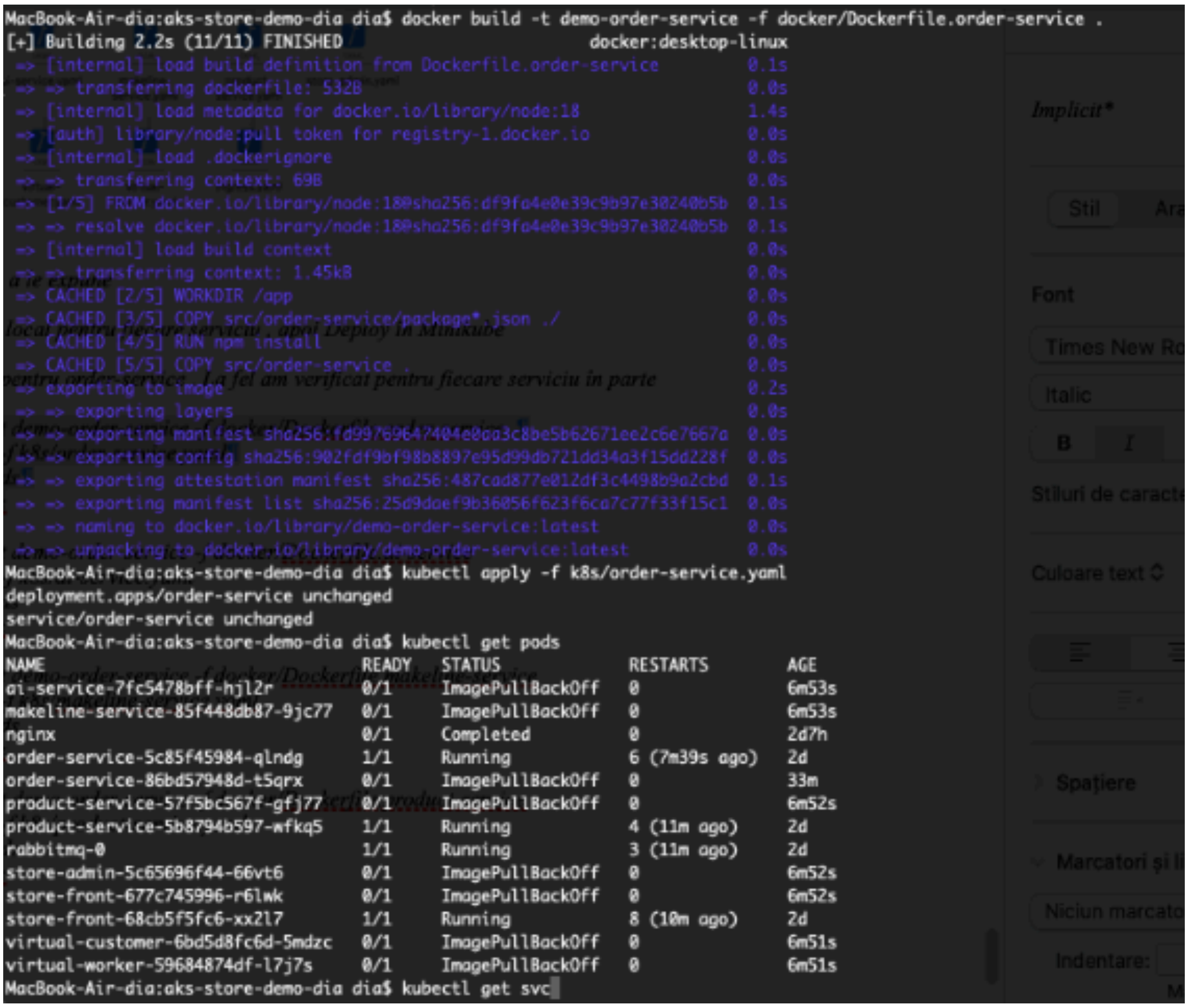
./deploy-all.sh

kubectl get pods

kubectl get ingress


docker
Dockerfile.ai-service
Dockerfile.order-service
Dockerfile.makeline-service
Dockerfile.product-service
Dockerfile.store-admin
Dockerfile.store-front
Dockerfile.virtual-customer
Dockerfile.virtual-worker





```
MacBook-Air-dia:aks-store-demo-dia dia$ docker build -t demo-order-service -f docker/Dockerfile.order-service .
[+] Building 2.2s (11/11) FINISHED                                          docker:desktop-linux
 => [internal] load build definition from Dockerfile.order-service                          0.1s
 => => transferring dockerfile: 532B                                                        0.0s
 => [internal] load metadata for docker.io/library/node:18                                  1.4s
 => [auth] library/node:pull token for registry-1.docker.io                                 0.0s
 => [internal] load .dockerignore                                                           0.0s
 => => transferring context: 69B                                                            0.0s
 => [1/5] FROM docker.io/library/node:18@sha256:df9fa4e0e39c9b97e30240b5b                   0.1s
 => => resolve docker.io/library/node:18@sha256:df9fa4e0e39c9b97e30240b5b                   0.1s
 => [internal] load build context                                                           0.0s
 => => transferring context: 1.45kB                                                         0.0s
 => CACHED [2/5] WORKDIR /app                                                               0.0s
 => CACHED [3/5] COPY src/order-service/package*.json ./                                    0.0s
 => CACHED [4/5] RUN npm install                                                            0.0s
 => CACHED [5/5] COPY src/order-service .                                                   0.0s
 => exporting to image                                                                      0.2s
 => => exporting layers                                                                     0.0s
 => => exporting manifest sha256:fd997696474 04e0aa3c8be5b62671ee2c6e7667a                  0.0s
 => => exporting config sha256:902fdf9bf98b8897e95d99db721dd34a3f15dd228f                   0.0s
 => => exporting attestation manifest sha256:487cad877e012df3c4498b9a2cbd                   0.1s
 => => exporting manifest list sha256:25d9daef9b36056f623f6ca7c77f33f15c1                   0.0s
 => => naming to docker.io/library/demo-order-service:latest                                0.0s
 => => unpacking to docker.io/library/demo-order-service:latest                             0.0s
MacBook-Air-dia:aks-store-demo-dia dia$ kubectl apply -f k8s/order-service.yaml
deployment.apps/order-service unchanged
service/order-service unchanged
MacBook-Air-dia:aks-store-demo-dia dia$ kubectl get pods
NAME                                  READY   STATUS             RESTARTS        AGE
ai-service-7fc5478bff-hjl2r           0/1     ImagePullBackOff   0               6m53s
makeline-service-85f448db87-9jc77     0/1     ImagePullBackOff   0               6m53s
nginx                                 0/1     Completed          0               2d7h
order-service-5c85f45984-qlndg        1/1     Running            6 (7m39s ago)   2d
order-service-86bd57948d-t5qrx        0/1     ImagePullBackOff   0               33m
product-service-57f5bc567f-gfj77      0/1     ImagePullBackOff   0               6m52s
product-service-5b8794b597-wfkq5      1/1     Running            4 (11m ago)     2d
rabbitmq-0                            1/1     Running            3 (11m ago)     2d
store-admin-5c65696f44-66vt6          0/1     ImagePullBackOff   0               6m52s
store-front-677c745996-r6lwk          0/1     ImagePullBackOff   0               6m52s
store-front-68cb5f5fc6-xx2l7          1/1     Running            8 (10m ago)     2d
virtual-customer-6bd5d8fc6d-5mdzc     0/1     ImagePullBackOff   0               6m51s
virtual-worker-59684874df-l7j7s       0/1     ImagePullBackOff   0               6m51s
MacBook-Air-dia:aks-store-demo-dia dia$ kubectl get svc
```

4. Bonus Steps

Helm Chart Creation:

- A Helm chart was created to manage the application deployment.
- It includes templates for Deployments, Services, ConfigMaps, etc.

Resource Limits and Requests:

```
resources:
  requests:
    cpu: "250m"
    memory: "256Mi"
  limits:
    cpu: "500m"
    memory: "512Mi"
```

This ensures better resource management inside the Kubernetes cluster.

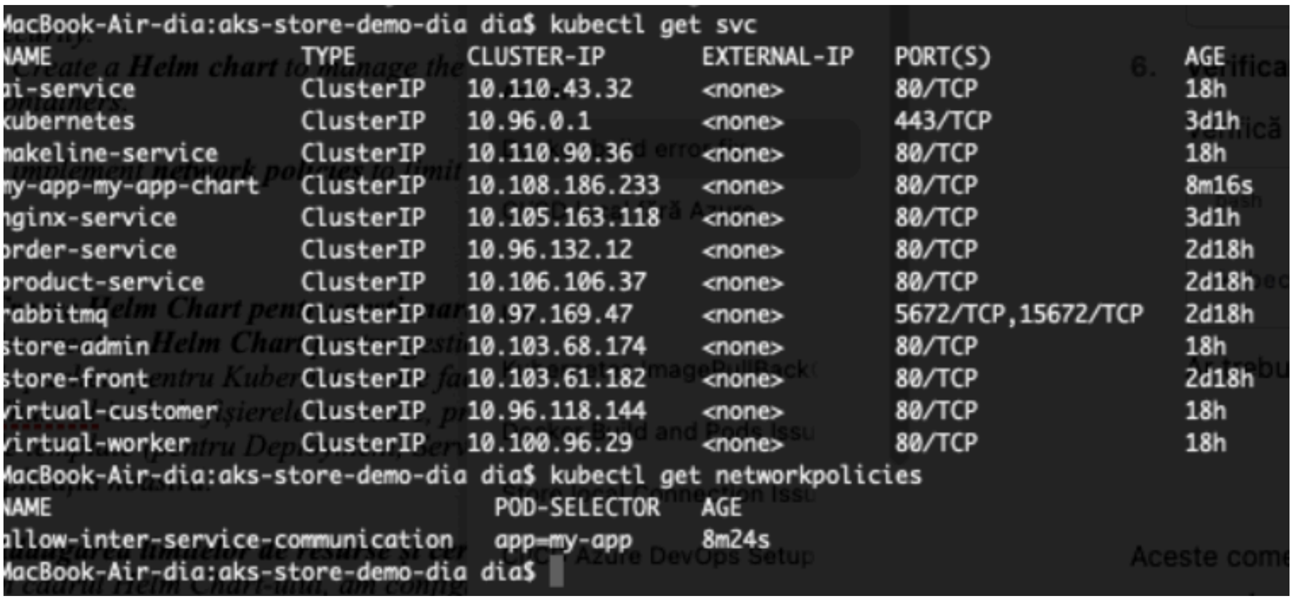
Network Policies:

- Implemented network policies to restrict inter-service communication.
- Example:

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-my-app-communication
spec:
  podSelector:
    matchLabels:
      app: my-app
  ingress:
    - from:
        - podSelector:
            matchLabels:
              app: my-app
```

Verification commands:

```
kubectl get pods
kubectl get svc
kubectl get networkpolicies
```



```
lacBook-Air-dia:aks-store-demo-dia dia$ kubectl get svc
IAME                   TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)              AGE
i-service              ClusterIP   10.110.43.32     <none>        80/TCP               18h
kubernetes             ClusterIP   10.96.0.1        <none>        443/TCP              3d1h
nakeline-service       ClusterIP   10.110.90.36     <none>        80/TCP               18h
ny-app-my-app-chart    ClusterIP   10.108.186.233   <none>        80/TCP               8m16s
nginx-service          ClusterIP   10.105.163.118   <none>        80/TCP               3d1h
order-service          ClusterIP   10.96.132.12     <none>        80/TCP               2d18h
product-service        ClusterIP   10.106.106.37    <none>        80/TCP               2d18h
rabbitmq               ClusterIP   10.97.169.47     <none>        5672/TCP,15672/TCP   2d18h
store-admin            ClusterIP   10.103.68.174    <none>        80/TCP               18h
store-front            ClusterIP   10.103.61.182    <none>        80/TCP               2d18h
virtual-customer       ClusterIP   10.96.118.144    <none>        80/TCP               18h
virtual-worker         ClusterIP   10.100.96.29     <none>        80/TCP               18h
lacBook-Air-dia:aks-store-demo-dia dia$ kubectl get networkpolicies
IAME                                 POD-SELECTOR   AGE
allow-inter-service-communication    app=my-app     8m24s
lacBook-Air-dia:aks-store-demo-dia dia$
```