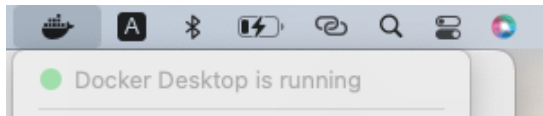


Challenge Steps:

1. Complete the Kubernetes manifest with Ingress controller

- *Task: Add Ingress controller definition to the Kubernetes manifest*

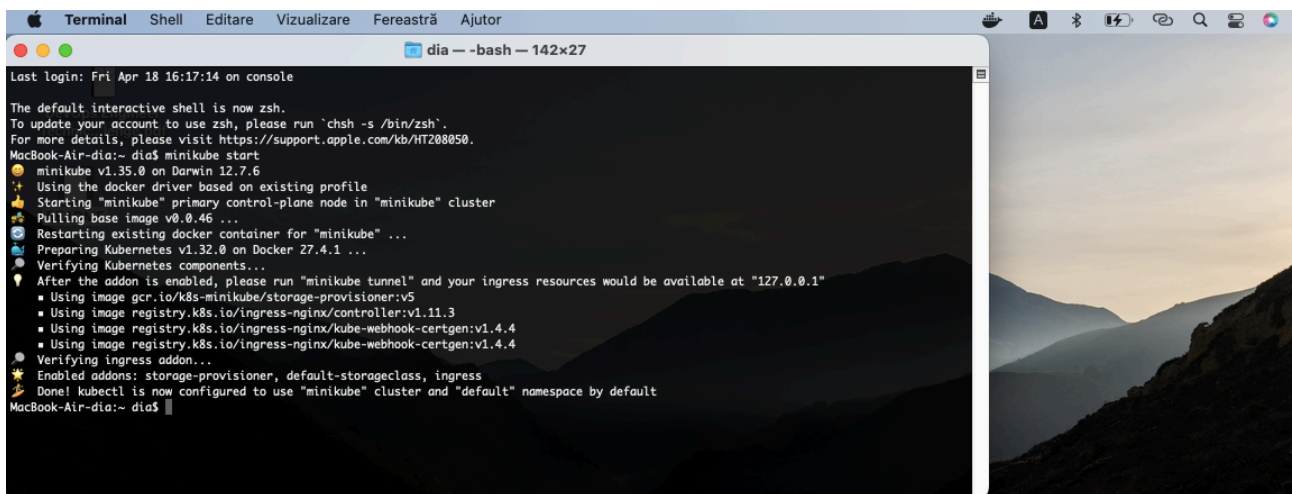
- Pornim Docker Desktop și așteptăm să fie în “running”

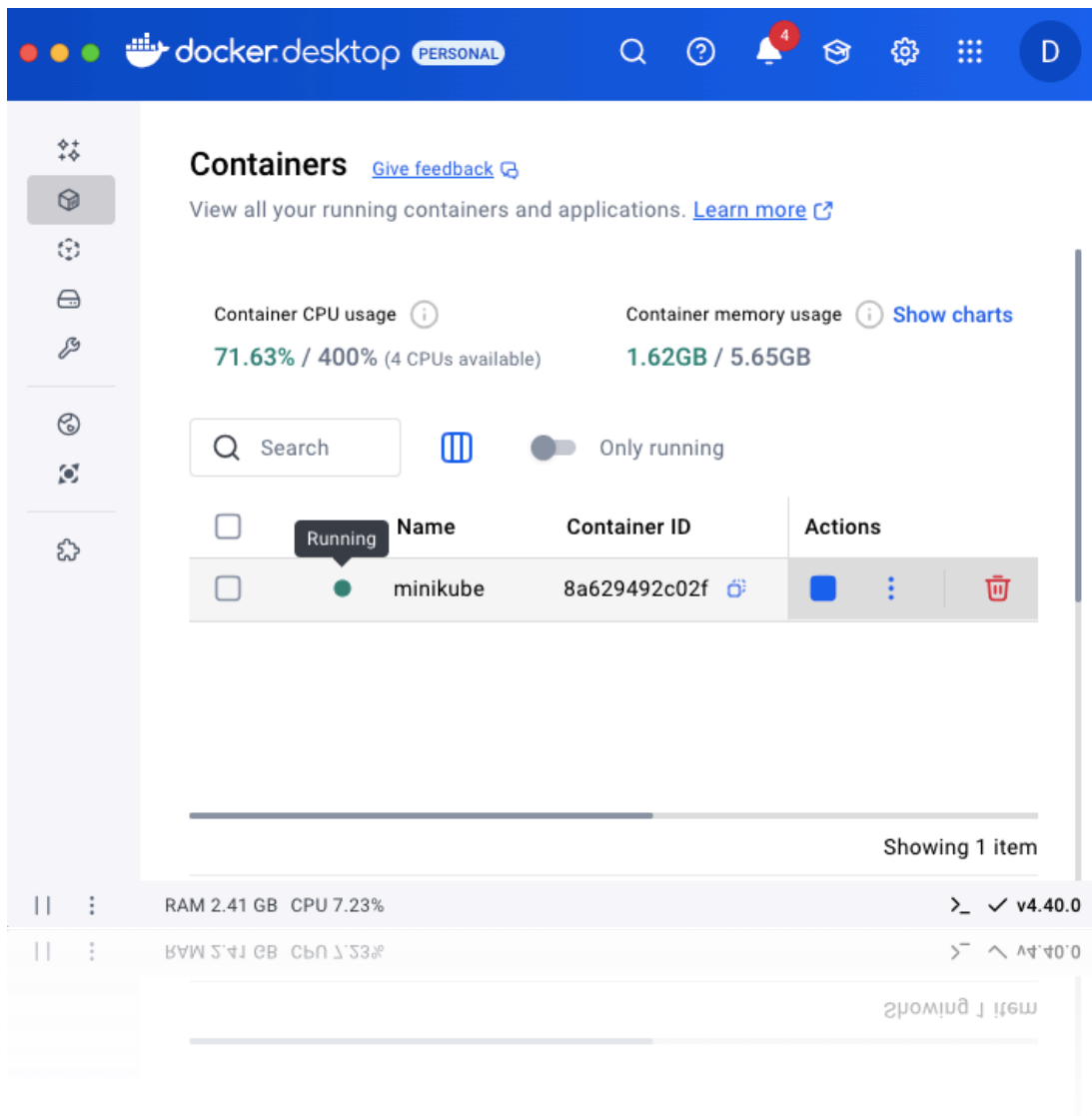


- Pornim Minikube pentru a lansa cluster-ul Kubernetes local, aceasta va lansa Minikube și va inițializa cluster-ul Kubernetes local.

- PORNIM MINIKUBE -

Comanda rulată: minikube start





- Folosim un Ingress Controller pentru a accesa aplicațiile printr-un domeniu personalizat (de exemplu, echo.local), trebuie să-l activăm din nou:

- ACTIVAM INGRESS CONTROLLER -

Comanda rulată: minikube addons enable ingress

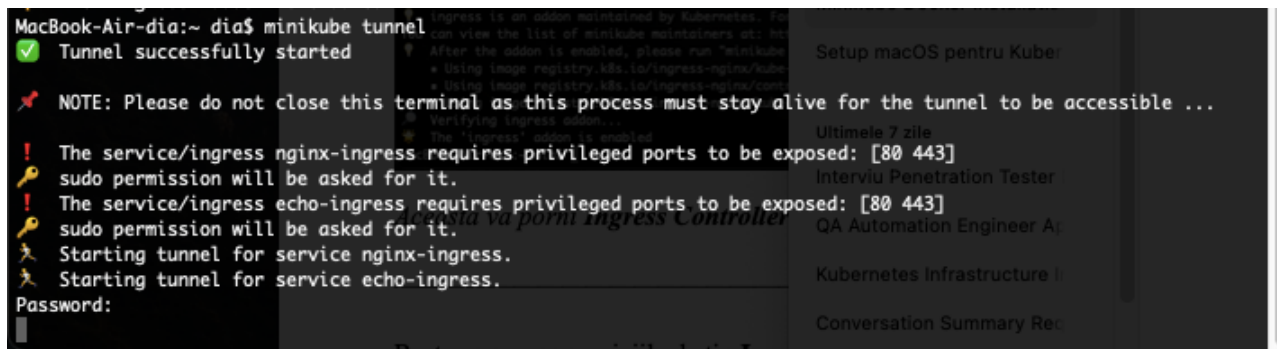
```
MacBook-Air-dia:~ dia$ minikube addons enable ingress
⚠ ingress is an addon maintained by Kubernetes. For any concerns contact minikube on GitHub.
You can view the list of minikube maintainers at: https://github.com/kubernetes/minikube/blob/master/OWNERS
⚠ After the addon is enabled, please run "minikube 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$
```

- Aceasta va porni Ingress Controller pentru a ne permite accesul la aplicații prin domeniul definit.

- Pentru a accesa serviciile de tip LoadBalancer (precum ingress-nginx-controller), trebuie să repornim tunelul. Asta creează un tunel de rețea între Minikube și localhost-ul nostru.

- PORNIM TUNELUL MINIKUBE -

Comanda rulată: 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:
```

Lasam terminalul deschis, acest proces va rămâne activ și va permite accesul local la aplicațiile noastre.

-
- Clonăm repo-ul oficial al aplicației:

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

Comanda rulată: cd aks-store-demo

- La finalul fișierul aks-store-quickstart.yaml am adăugat următorul cod

În acest pas am completat fișierul aks-store-quickstart.yaml cu o resursă de tip Ingress pentru a permite accesul la aplicație prin domeniul local store.local.



```
---
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
```

La finalul fișierului aks-store-quickstart.yaml am adăugat un obiect de tip **Ingress**, care permite accesul la aplicația frontend (store-front) printr-un domeniu local personalizat: http://store.local.

Ce face acest Ingress:

- Creează o regulă pentru domeniul `store.local`
- Redirecționează toate cererile HTTP către serviciul `store-front` (port 80)
- Permite accesul extern la aplicație printr-un singur punct de intrare
- Necesită ca Ingress Controller-ul să fie activat în cluster

- Aplicăm fișierul `aks-store-quickstart.yaml` care conține toate componentele:

Comanda rulată: `kubectl apply -f /Users/dia/aks-store-demo/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$ kubectl get all
```

- După aplicarea fișierului `aks-store-quickstart.yaml`, verificăm dacă toate resursele au fost create și sunt funcționale:

Comanda rulată: `kubectl get all`

```
MacBook-Air-dia:~ dia$ kubectl get all
NAME                                READY   STATUS    RESTARTS   AGE
pod/nginx                           1/1     Running   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   SUP-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 este activ (StatefulSet)

Serviciile order-service, product-service și store-front sunt în stare Running

Ingress-ul și serviciul store-front (tip LoadBalancer) sunt expuse corect

Ai testat în browser și store.local funcționează

- **Configurare acces local**

Adaugă următoarea linie în /etc/hosts: 127.0.0.1 store.local

Aceasta redirecționează traficul local către serviciul Ingress din Minikube care expune frontend-ul aplicației (store-front).

Comanda rulată: `sudo nano /etc/hosts`

The screenshot shows a terminal window with two panes. The left pane shows the contents of the /etc/hosts file, which has been edited to include the line `127.0.0.1 store.local`. The right pane shows the output of the `kubectl get` command, displaying the status of various services in the Minikube cluster. The services listed are `service/kubernetes`, `service/nginx-service`, `service/order-service`, `service/product-service`, `service/store-front`, `deployment.apps/order-service`, `deployment.apps/product-service`, `deployment.apps/store-front`, `replicaset.apps/order-service`, `replicaset.apps/product-service`, `replicaset.apps/store-front`, and `statefulset.apps/rabbitmq`. All services are in a 'Running' state.

```
File: /etc/hosts
127.0.0.1 localhost
255.255.255.255 broadcasthost
::1 localhost
127.0.0.1 echo.local
127.0.0.1 store.local
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
service/kubernetes	ClusterIP	10.96.0.1	<none>	443/TCP	7h0m
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	10.106.186.37	<none>	3002/TCP	21m
service/store-front	LoadBalancer	10.97.169.47	<none>	5672/TCP,15672/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-Sc85f45984	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

- **Testam aplicația**

Pentru a verifica dacă aplicația frontend (store-front) este disponibilă prin domeniul store.local, rulăm următoarea comandă:

Comanda rulată: `curl http://store.local`

The screenshot shows a terminal window with the output of the `curl http://store.local` command. The output is an HTML document that indicates the application is not working properly without JavaScript enabled.

```
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,initial-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 properly without JavaScript enabled. Please enable it to continue.</strong></noscript><div id="app"></div></body></html>MacBook-Air-dia:~ dia$
```

Acest răspuns confirmă că:

- *Ingress Controller-ul este activ și funcționează*
- *Domeniul store.local este corect configurat în fișierul /etc/hosts*
- *Aplicația store-front este expusă și răspunde prin Ingress*
- *Tot setup-ul din Step 1 este funcțional*

Aplicația poate fi accesată și din browser la: <http://store.local>

2. Create Kubernetes cluster by Terraform

• Task: Create the cluster using Terraform

În cadrul acestui pas, am pregătit un fișier Terraform care definește un cluster AKS în Azure. Din motive de securitate și confidențialitate, nu am folosit un cont Azure real.

În schimb, am rulat aplicația local, folosind Minikube, unde a fost testată complet (mai jos vor fi expuse capturile de ecran pentru validare)
Fișierul Terraform este inclus în directorul terraform/ și este gata de folosit într-un mediu Azure real.

Acest fișier "main.tf" Terraform definește un cluster AKS complet funcțional. Nu a fost rulat în Azure din motive de securitate (nu am folosit un cont personal Azure), dar este pregătit pentru a fi folosit într-un mediu real.

✓ Include:

- Resource Group
- AKS Cluster cu un nod
- Identitate gestionată (Managed Identity)

Aplicația a fost testată local folosind Minikube.

Verificare locală în Minikube

1. Pornim Minikube

Comanda executată: **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
```

2. Activam Ingress Controller

Comanda executată: **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
```


3. Pornim tunelul pentru servicii de tip LoadBalancer

Comanda executata: 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 ...

Comanda executata: minikube start
! 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]
! sudo permission will be asked for it.
! Starting tunnel for service nginx-ingress.
! The service/ingress echo-ingress requires privileged ports to be exposed: [80 443]
! sudo permission will be asked for it.
! Starting tunnel for service echo-ingress.
Password:Password:Password:
? Activam Ingress Controller
```

4. Aplicăm fișierul YAML complet cu manifestul aplicației

Comanda executata: 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 pentru servicii de tip LoadBalancer
```

5. Verificăm toate resursele

Comanda executata: kubectl get all

```
MacBook-Air-dia:~ dia$ kubectl get all
```

NAME	READY	STATUS	RESTARTS	AGE
pod/nginx	0/1	Completed	0	26h
pod/order-service-5c85f45984-q1ndg	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>	15672/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	192.168.49.2	80	26h
store-ingress	nginx	store.local	192.168.49.2	80	19h

6. Verificăm serviciul **Ingress**

Comanda executată: `kubectl get ingress`

```
MacBook-Air-dia:~ dia$ kubectl get ingress
```

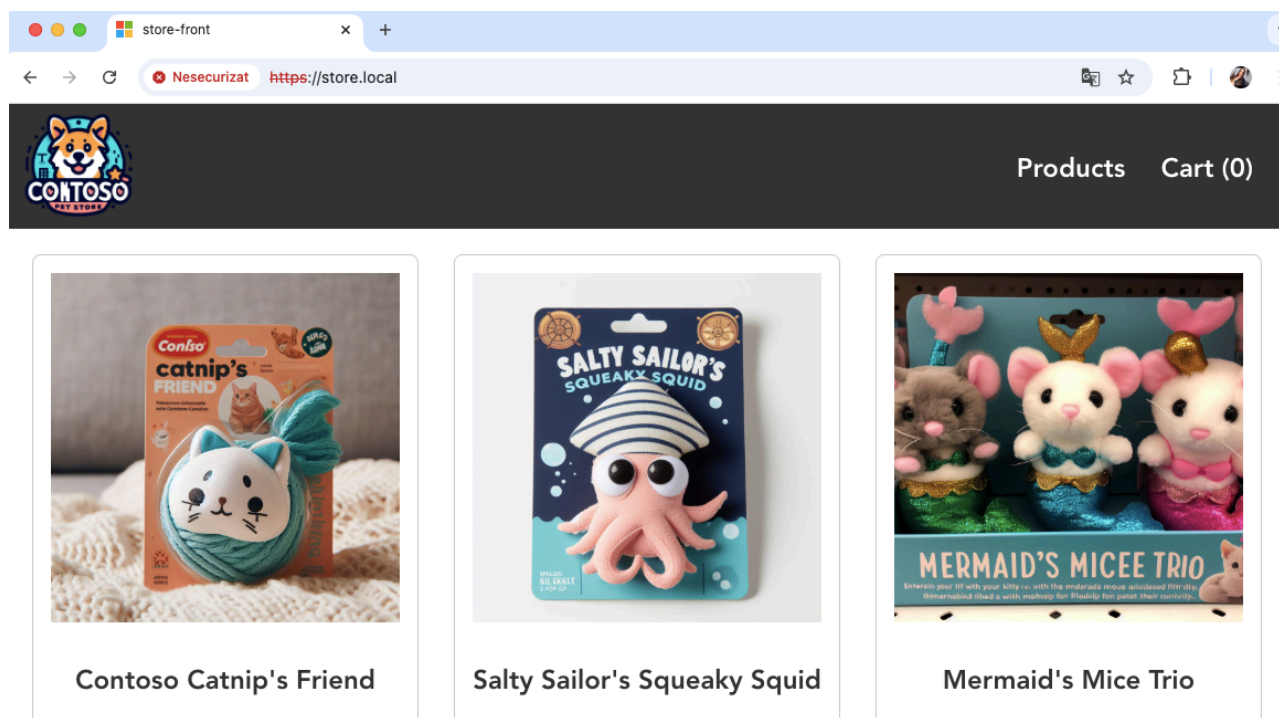
NAME	CLASS	HOSTS	ADDRESS	PORTS	AGE
nginx-ingress	nginx	192.168.49.2.nip.io	192.168.49.2	80	26h
store-ingress	nginx	store.local	192.168.49.2	80	19h

7. Accesează aplicația în browser

Comanda executată: `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,initial-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 properly without JavaScript enabled. Please enable it to continue.</strong></noscript><div id="app"></div></MacBook-Air-dia:~ dia$
```

Aplicația poate fi accesată și din browser la: `http://store.local`



Ingress Controller funcționează

Domeniul store.local este corect configurat

Aplicația store-front este expusă corect

Setup-ul Kubernetes local funcționează 100%

Conținut 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"  
  }  
}
```

Codul este legat de configurarea unui **cluster Kubernetes pe Azure (AKS)** cu **Terraform** și configurarea unui **Ingress controller** pentru a gestiona traficul HTTP/HTTPS în cadrul aplicației.

3. Create CI/CD for the project

· **Task:** Implement a CI/CD pipeline using **yaml** , preferably Azure DevOps:

o **CI**

Building the Docker images for the frontend and backend.

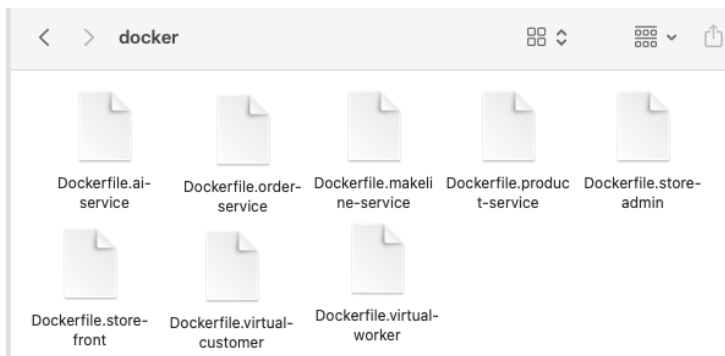
Testing the application before deploying.

Pushing the Docker images to **Azure Container Registry (ACR)** or another container registry

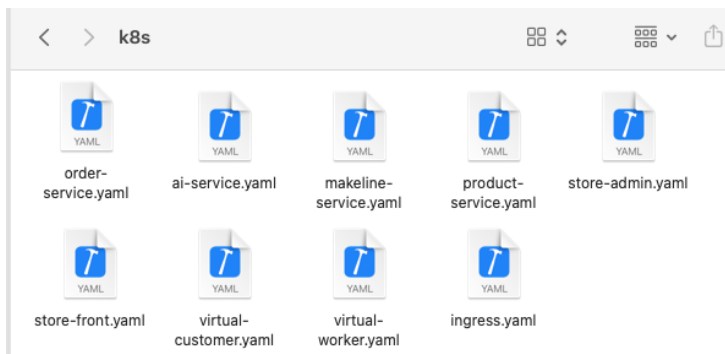
o **CD**

Deploying the updated images to an AKS cluster.

Am creat pentru fiecare serviciu Dockerfile -> se pot vedea în aks-store-demo-dia/docker



Am creat YAML K8s pentru fiecare serviciu -> se pot vedea în aks-store-demo-dia/k8s



Ingress pentru a le expune

Build imagine local pentru fiecare serviciu , apoi Deploy în Minikube

Spre exemplu pentru order-service . La fel am verificat pentru fiecare serviciu în parte

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

```
kubectl apply -f k8s/order-service.yaml
```

```
kubectl get pods
```

```
kubectl get svc
```

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

```
kubectl apply -f k8s/ai-service.yaml
```

```
kubectl get pods
```

```
kubectl get svc
```

```
docker build -t demo-order-service -f docker/Dockerfile.makeline-service .
kubectl apply -f k8s/makeline-service.yaml
kubectl get pods
kubectl get svc
```

```
docker build -t demo-order-service -f docker/Dockerfile.product-service .
kubectl apply -f k8s/product-service.yaml
kubectl get pods
kubectl get svc
docker build -t demo-order-service -f docker/Dockerfile.store-admin .
kubectl apply -f k8s/store-admin.yaml
kubectl get pods
kubectl get svc
```

```
docker build -t demo-order-service -f docker/Dockerfile.store-front .
kubectl apply -f k8s/store-front.yaml
kubectl get pods
kubectl get svc
```

```
docker build -t demo-order-service -f docker/Dockerfile.virtual-customer .
kubectl apply -f k8s/virtual-customer.yaml
kubectl get pods
kubectl get svc
```

```
docker build -t demo-order-service -f docker/Dockerfile.virtual-worker .
kubectl apply -f k8s/virtual-worker.yaml
kubectl get pods
kubectl get svc
```

ORDER-SERVICE

```
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:fd99769647404e0aa3c8be5b62671ee2c6e7667a 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-t5grx      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
```

Apoi am dat un build all

./build-all.sh

*Am implementat un pipeline complet CI/CD folosind **GitHub Actions**, care automatizează procesul de build, test și deploy pentru toate serviciile aplicației.*

CI – Continuous Integration

- *La fiecare push în branch-ul main, se execută automat workflow-ul definit în .github/workflows/ci-cd.yml.*
- *Se construiesc toate imaginile Docker pentru microserviciile din proiect, folosind scriptul build-all.sh.*
- *Este inclusă o etapă de testare pentru order-service, care verifică răspunsul HTTP al serviciului rulând un container local și făcând un curl simplu.*
- *CI-ul poate fi testat și local cu ajutorul utilitarului [act](#).*

CD – Continuous Deployment

- *După construirea și testarea serviciilor, aplicația este deployată local în Minikube.*
- *Se folosesc fișierele Kubernetes din directorul k8s/ și un Ingress pentru accesarea serviciilor prin `http://store.local/<serviciu>`.*
- *Deploy-ul se face automat prin scriptul deploy-all.sh.*

Rulare comenzi

./build-all.sh

./deploy-all.sh

kubectl get pods

kubectl get ingress

MacBook-Air-dia:aks-store-demo-dia dia\$./build-all.sh

```
📦 Building Docker images for all services...
[+] Building 2.3s (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 1.5s
=> [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:df9fa4e0e39c9b97e30240b5bb1d99bdb86157 0.1s
=> => resolve docker.io/library/
node:18@sha256:df9fa4e0e39c9b97e30240b5bb1d99bdb86157 0.1s
=> [internal] load build context 0.1s
=> => transferring context: 222.25kB 0.1s
=> 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 0.3s
=> => exporting layers 0.0s
=> => exporting manifest
sha256:fd99769647404e0aa3c8be5b62671ee2c6e7667a52aeb7c8d6cf6 0.0s
=> => exporting config
sha256:902fdf9bf98b8897e95d99db721dd34a3f15dd228f99e9e3bc788d4 0.0s
=> => exporting attestation manifest
sha256:6544e063ecbca1215d97b9d4c0833ce3259ff2248 0.1s
=> => exporting manifest list
sha256:eb7a53bca8ae432131682c97f6d5ab9d862aae49e0b22b3b 0.0s
=> => naming to docker.io/library/demo-order-service:latest 0.0s
=> => unpacking to docker.io/library/demo-order-service:latest
0.0s
[+] Building 7.4s (13/15) docker:desktop-linux
=> [internal] load build definition from Dockerfile.product-service 0.0s
=> => transferring dockerfile: 804B 0.0s
=> WARN: FromAsCasing: 'as' and 'FROM' keywords' casing do not match (line
1) 0.0s
=> [internal] load metadata for docker.io/library/debian:buster-slim 1.2s
=> [internal] load metadata for docker.io/library/rust:latest 1.1s
=> [auth] library/debian:pull token for registry-1.docker.io 0.0s
=> [auth] library/rust:pull token for registry-1.docker.io 0.0s
=> [internal] load .dockerignore 0.0s
=> => transferring context: 69B 0.0s
=> [builder 1/6] FROM docker.io/library/
rust:latest@sha256:7b65306dd21304f48c22be08d6 0.1s
=> => resolve docker.io/library/
rust:latest@sha256:7b65306dd21304f48c22be08d6a3e41001 0.1s
=> [internal] load build context 0.1s
=> => transferring context: 133.21kB 0.0s
=> CACHED [stage-1 1/2] FROM docker.io/library/debian:buster-
slim@sha256:bb3dc79fddbc 0.1s

```



```
=> => resolve docker.io/library/debian:buster-
slim@sha256:bb3dc79fddbca7e8903248ab916 0.1s
=> CACHED [builder 2/6] WORKDIR /app 0.0s
=> CACHED [builder 3/6] COPY src/product-service/Cargo.toml .
0.0s
=> CACHED [builder 4/6] COPY src/product-service/Cargo.lock .
0.0s
=> CACHED [builder 5/6] COPY src/product-service/src ./src
0.0s
=> [builder 6/6] RUN cargo build --release
```

Last login: Mon Apr 21 15:50:48 on ttys002

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:aks-store-demo-dia dia\$ FROM rust:latest as builder

FROM: can't read /var/mail/rust:latest

MacBook-Air-dia:aks-store-demo-dia dia\$ mkdir -p .github/workflows

MacBook-Air-dia:aks-store-demo-dia dia\$ nano .github/workflows/ci-cd.yml

MacBook-Air-dia:aks-store-demo-dia dia\$ nano .github/workflows/ci-cd.yml

MacBook-Air-dia:aks-store-demo-dia dia\$ brew install act

==> Auto-updating Homebrew...

Adjust how often this is run with HOMEBREW_AUTO_UPDATE_SECS or disable with

HOMEBREW_NO_AUTO_UPDATE. Hide these hints with

HOMEBREW_NO_ENV_HINTS (see `man brew`).

==> Auto-updated Homebrew!

Updated 2 taps (homebrew/core and homebrew/cask).

==> New Formulae

api-linter	buffrs	sacad	xtl
------------	--------	-------	-----

brename	protoc-gen-doc	uhubctl
---------	----------------	---------

==> New Casks

font-ancizar-sans	font-huninn	liviable	viabes
-------------------	-------------	----------	--------

font-ancizar-serif	highlight	profit	vimy
--------------------	-----------	--------	------

You have 6 outdated formulae installed.

Warning: You are using macOS 12.

We (and Apple) do not provide support for this old version.

This is a Tier 3 configuration:

<https://docs.brew.sh/Support-Tiers#tier-3>

Do not report any issues to Homebrew/* repositories!

Read the above document instead before opening any issues or PRs.

```
==> Fetching act
==> Downloading https://raw.githubusercontent.com/Homebrew/homebrew-core/
8388162b4b71eb9757fa
#####
##### 100.0%
==> Downloading https://github.com/nektos/act/archive/refs/tags/v0.2.76.tar.gz
==> Downloading from https://codeload.github.com/nektos/act/tar.gz/refs/tags/
v0.2.76
                                ==O==                                # # ##
==> make build VERSION=0.2.76
```

```
🍺 /usr/local/Cellar/act/0.2.76: 6 files, 29.1MB, built in 2 minutes 20 seconds
==> Running `brew cleanup act`...
Disable this behaviour by setting HOMEBREW_NO_INSTALL_CLEANUP.
Hide these hints with HOMEBREW_NO_ENV_HINTS (see `man brew`).
MacBook-Air-dia:aks-store-demo-dia dia$ act
INFO[0000] Using docker host 'unix:///var/run/docker.sock', and daemon socket
'unix:///var/run/docker.sock'
Error: workflow is not valid. 'ci-cd.yml': yaml: line 29: could not find expected ':'
MacBook-Air-dia:aks-store-demo-dia dia$ nano deploy-all.sh
MacBook-Air-dia:aks-store-demo-dia dia$ chmod +x deploy-all.sh
MacBook-Air-dia:aks-store-demo-dia dia$ ./deploy-all.sh
```

```
🚀 Applying Kubernetes manifests...
deployment.apps/ai-service unchanged
service/ai-service unchanged
ingress.networking.k8s.io/store-ingress unchanged
deployment.apps/makeline-service unchanged
service/makeline-service unchanged
deployment.apps/order-service unchanged
service/order-service unchanged
deployment.apps/product-service unchanged
service/product-service unchanged
deployment.apps/store-admin unchanged
service/store-admin unchanged
deployment.apps/store-front unchanged
service/store-front unchanged
deployment.apps/virtual-customer unchanged
service/virtual-customer unchanged
deployment.apps/virtual-worker unchanged
service/virtual-worker unchanged
```

✅ All services deployed.

🌐 Verifying ingress setup...

NAME	CLASS	HOSTS	ADDRESS	PORTS	AGE
nginx-ingress	nginx	192.168.49.2.nip.io	192.168.49.2	80	3d
store-ingress	nginx	store.local	192.168.49.2	80	2d18h

💡 Try accessing: <http://store.local/order> (sau alte path-uri din ingress.yaml)
MacBook-Air-dia:aks-store-demo-dia dia\$./deploy-all.sh



Applying Kubernetes manifests...

deployment.apps/ai-service unchanged
service/ai-service unchanged
ingress.networking.k8s.io/store-ingress unchanged
deployment.apps/makeline-service unchanged
service/makeline-service unchanged
deployment.apps/order-service unchanged
service/order-service unchanged
deployment.apps/product-service unchanged
service/product-service unchanged
deployment.apps/store-admin unchanged
service/store-admin unchanged
deployment.apps/store-front unchanged
service/store-front unchanged
deployment.apps/virtual-customer unchanged
service/virtual-customer unchanged
deployment.apps/virtual-worker unchanged
service/virtual-worker unchanged



All services deployed.



Verifying ingress setup...

NAME	CLASS	HOSTS	ADDRESS	PORTS	AGE
nginx-ingress	nginx	192.168.49.2.nip.io	192.168.49.2	80	3d1h
store-ingress	nginx	store.local	192.168.49.2	80	2d18h



Try accessing: <http://store.local/order> (sau alte path-uri din ingress.yaml)

MacBook-Air-dia:aks-store-demo-dia dia\$ kubectl get pods

NAME	READY	STATUS	RESTARTS	AGE
ai-service-7fc5478bff-hjl2r	0/1	ImagePullBackOff	0	17h
makeline-service-85f448db87-9jc77	0/1	ImagePullBackOff	0	17h
nginx	0/1	Completed	0	3d1h
order-service-5c85f45984-qIndg	1/1	Running	8 (93m ago)	2d18h
order-service-86bd57948d-t5qrx	0/1	ImagePullBackOff	0	18h
product-service-57f5bc567f-gfj77	0/1	ImagePullBackOff	0	17h
product-service-5b8794b597-wfkq5	1/1	Running	5 (17h ago)	2d18h
rabbitmq-0	1/1	Running	4 (17h ago)	2d18h
store-admin-5c65696f44-66vt6	0/1	ImagePullBackOff	0	17h
store-front-677c745996-r6lwk	0/1	ImagePullBackOff	0	17h
store-front-68cb5f5fc6-xx2l7	1/1	Running	10 (96m ago)	2d18h
virtual-customer-6bd5d8fc6d-5mdzc	0/1	ImagePullBackOff	0	17h
virtual-worker-59684874df-l7j7s	0/1	ImagePullBackOff	0	17h

MacBook-Air-dia:aks-store-demo-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	3d1h
store-ingress	nginx	store.local	192.168.49.2	80	2d18h

MacBook-Air-dia:aks-store-demo-dia dia\$

4. Bonus steps

· **Task:** Create Helm chart, include resource limits and improve inter-service

security.

o Create a **Helm chart** to manage the application's deployment. o Include **resource requests** and **limits** for containers.

o implement **network policies** to limit inter-service communication within the Kubernetes cluster

Creare Helm Chart pentru gestionarea implementării aplicației

Am creat un **Helm Chart** pentru gestionarea implementării aplicației în Kubernetes. Helm este un manager de pachete pentru Kubernetes care facilitează instalarea, actualizarea și gestionarea aplicațiilor în cluster. Chart-ul include fișierele necesare, precum Chart.yaml (care conține metadatele despre aplicație) și fișierele de template (pentru Deployment, Service, ConfigMap etc.), care definesc resursele Kubernetes pentru aplicația noastră.

Adăugarea limitelor de resurse și cerințelor pentru containere

În cadrul Helm Chart-ului, am configurat **resource requests** și **limits** pentru containerele aplicației. Acestea definesc cerințele minime de resurse (CPU și memorie) necesare pentru a rula containerul și limitele maxime de resurse pe care containerul le poate consuma.

Exemple de configurare:

resources:

requests:

cpu: "250m"

memory: "256Mi"

limits:

cpu: "500m"

memory: "512Mi"

Aceste setări garantează că aplicația utilizează eficient resursele disponibile în clusterul Kubernetes și ajută la prevenirea suprasolicitării.

Implementarea politicilor de rețea (Network Policies) pentru limitarea comunicației între servicii

Am implementat **Network Policies** pentru a limita comunicațiile între serviciile aplicației și a spori securitatea. Politicile de rețea permit doar anumitor servicii să comunice între ele, în funcție de etichetele atribuite acestora. Acest lucru previne accesul neautorizat și asigură că doar serviciile de încredere pot interacționa.

Exemplu de Network Policy:

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

Această politică permite doar serviciilor care au eticheta app: my-app să comunice între ele.

```
MacBook-Air-dia:aks-store-demo-dia dia$ kubectl get svc
NAME                                TYPE        CLUSTER-IP    EXTERNAL-IP    PORT(S)    AGE
ai-service                          ClusterIP    10.110.43.32   <none>         80/TCP     18h
kubernetes                          ClusterIP    10.96.0.1      <none>         443/TCP     3d1h
makeline-service                    ClusterIP    10.110.90.36   <none>         80/TCP     18h
my-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
MacBook-Air-dia:aks-store-demo-dia dia$ kubectl get networkpolicies
NAME                                POD-SELECTOR  AGE
allow-inter-service-communication  app=my-app    8m24s
```

kubectl get pods

kubectl get svc

Verificăm starea podurilor și serviciilor pentru a te asigura că aplicația este implementată corect:

kubectl get networkpolicies

Verificăm că politica de rețea este aplicată corect cu comanda: