GitHub Repository Mirroring

Objective

To mirror an existing GitHub repository into your own GitHub account, including all branches, tags, and commit history.

Steps Performed

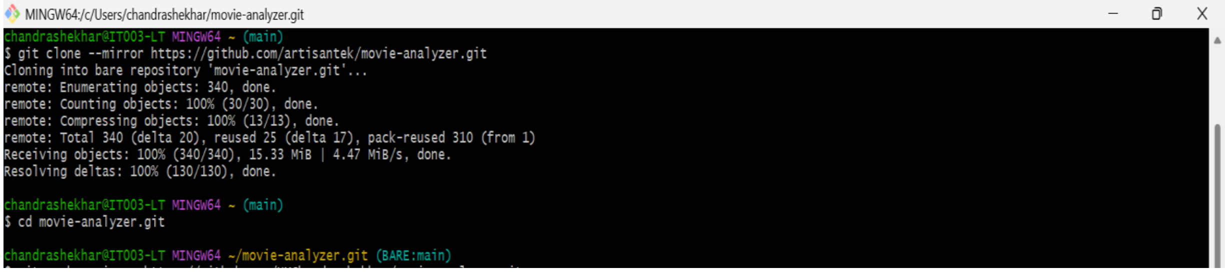
1. Clone the Repository as a Mirror

The following command was executed to create a bare mirror clone:

git clone --mirror https://github.com/artisantek/movie-analyzer.git

2. Navigate Into the Mirror Repository

cd movie-analyzer.git



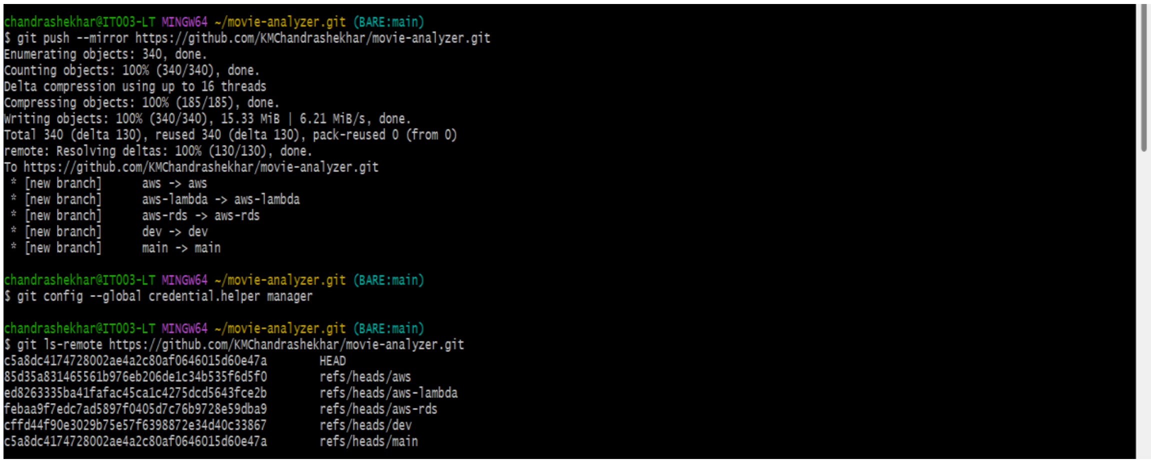
MINGW64:/c/Users/chandrashekhar/movie-analyzer.git

```
chandrashekhar@IT003-LT MINGW64 ~ (main)
$ git clone --mirror https://github.com/artisantek/movie-analyzer.git
Cloning into bare repository 'movie-analyzer.git'...
remote: Enumerating objects: 340, done.
remote: Counting objects: 100% (30/30), done.
remote: Compressing objects: 100% (13/13), done.
remote: Total 340 (delta 20), reused 25 (delta 17), pack-reused 310 (from 1)
Receiving objects: 100% (340/340), 15.33 MiB | 4.47 MiB/s, done.
Resolving deltas: 100% (130/130), done.

chandrashekhar@IT003-LT MINGW64 ~ (main)
$ cd movie-analyzer.git

chandrashekhar@IT003-LT MINGW64 ~/movie-analyzer.git (BARE:main)
```

3. Push the Mirror to Your GitHub Repository

A new empty repository named 'movie-analyzer' was created in your GitHub account. Then the mirror was pushed using the following command:

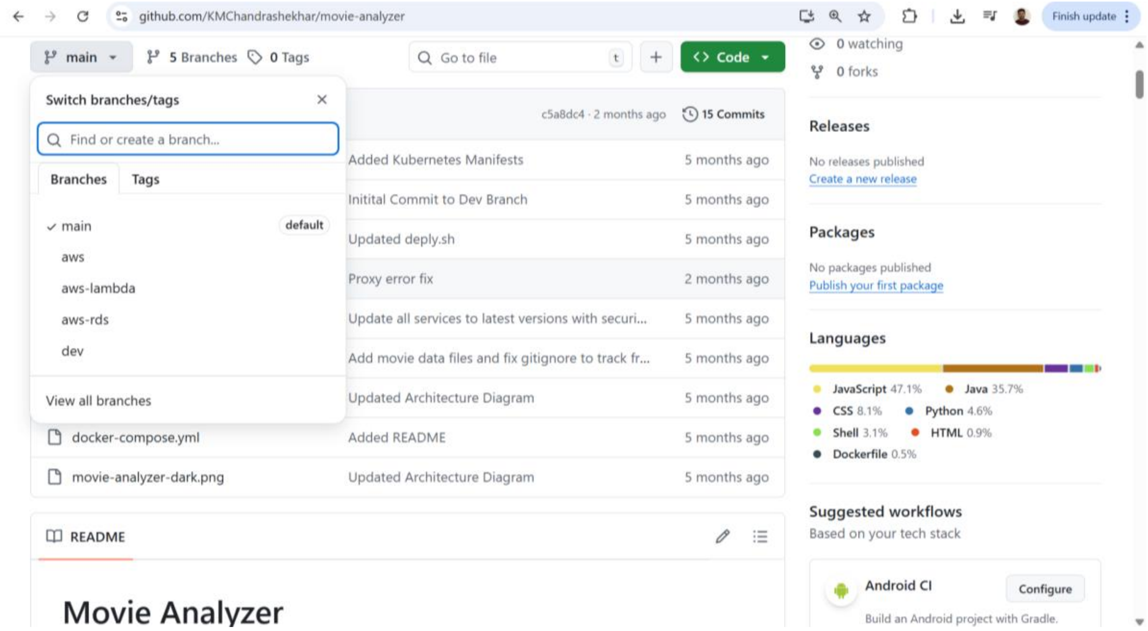
git push --mirror https://github.com/KMChandrashekhar/movie-analyzer.git



```
chandrashekhar@IT003-LT MINGW64 ~/movie-analyzer.git (BARE:main)
$ git push --mirror https://github.com/KMChandrashekhar/movie-analyzer.git
Enumerating objects: 340, done.
Counting objects: 100% (340/340), done.
Delta compression using up to 16 threads
Compressing objects: 100% (185/185), done.
Writing objects: 100% (340/340), 15.33 MiB | 6.21 MiB/s, done.
Total 340 (delta 130), reused 340 (delta 130), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (130/130), done.
To https://github.com/KMChandrashekhar/movie-analyzer.git
 * [new branch]      aws -> aws
 * [new branch]      aws-lambda -> aws-lambda
 * [new branch]      aws-rds -> aws-rds
 * [new branch]      dev -> dev
 * [new branch]      main -> main

chandrashekhar@IT003-LT MINGW64 ~/movie-analyzer.git (BARE:main)
$ git config --global credential.helper manager

chandrashekhar@IT003-LT MINGW64 ~/movie-analyzer.git (BARE:main)
$ git ls-remote https://github.com/KMChandrashekhar/movie-analyzer.git
c5a8dc4174728002ae4a2c80af0646015d60e47a        HEAD
85d35a831465561b976eb206de1c34b535f6d5f0        refs/heads/aws
ed8263335ba41fafac45ca1c4275dcd5643fce2b        refs/heads/aws-lambda
febaa9f7edc7ad5897f0405d7c76b9728e59dba9        refs/heads/aws-rds
cffd44f90e3029b75e57f6398872e34d40c33867        refs/heads/dev
c5a8dc4174728002ae4a2c80af0646015d60e47a        refs/heads/main
```

4. Verify the Remote Repository

To confirm that all branches and refs were mirrored, the following command was executed:



- ✓ main default
- aws
- aws-lambda
- aws-rds
- dev

c5a8dc4 · 2 months ago 15 Commits

	Added Kubernetes Manifests	5 months ago
	Initital Commit to Dev Branch	5 months ago
	Updated deply.sh	5 months ago
	Proxy error fix	2 months ago
	Update all services to latest versions with securi...	5 months ago
	Add movie data files and fix gitignore to track fr...	5 months ago
	Updated Architecture Diagram	5 months ago
docker-compose.yml	Added README	5 months ago
movie-analyzer-dark.png	Updated Architecture Diagram	5 months ago

Movie Analyzer

● JavaScript 47.1%	● Java 35.7%
● CSS 8.1%	● Python 4.6%
● Shell 3.1%	● HTML 0.9%
● Dockerfile 0.5%	

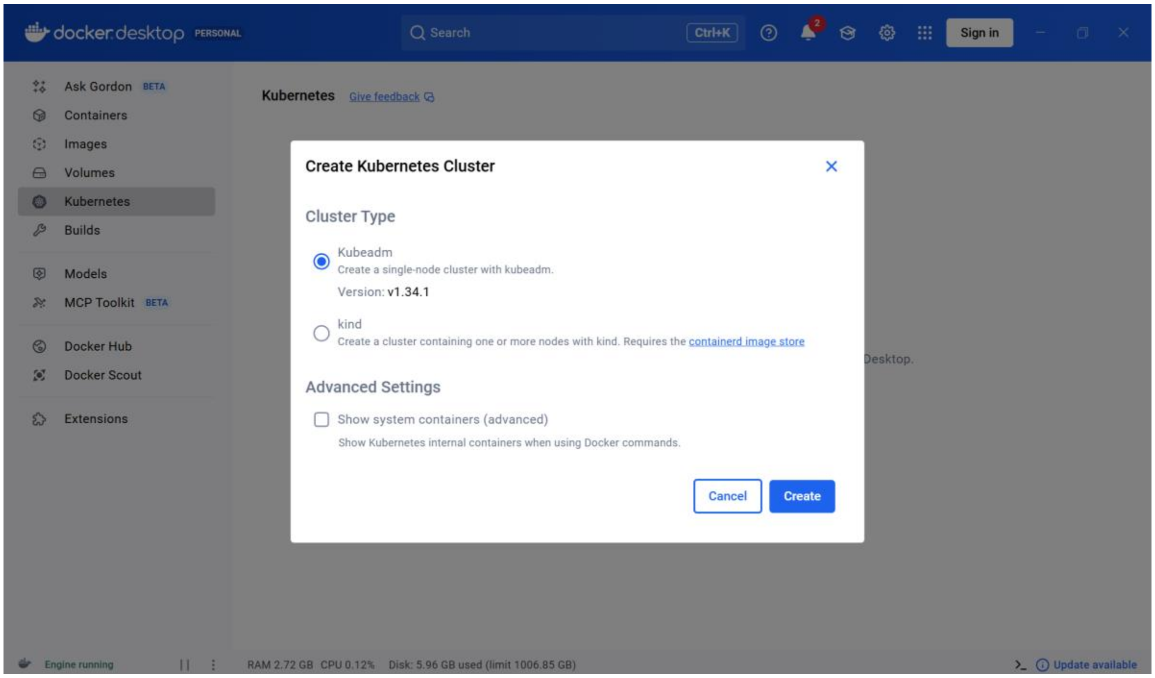
Conclusion

The GitHub repository was successfully mirrored. All branches, commit history, and references from the original repository were replicated into the new repository.

Create QA Environment using Helm & Docker Desktop Kubernetes

Objective

To create a new QA environment using a Helm chart and deploy it on a local Kubernetes cluster running inside Docker Desktop. The task includes setting up a Helm chart, creating environment-specific values files, installing the QA environment, and confirming successful deployment through Kubernetes service accessibility.



Create Kubernetes Cluster

Cluster Type

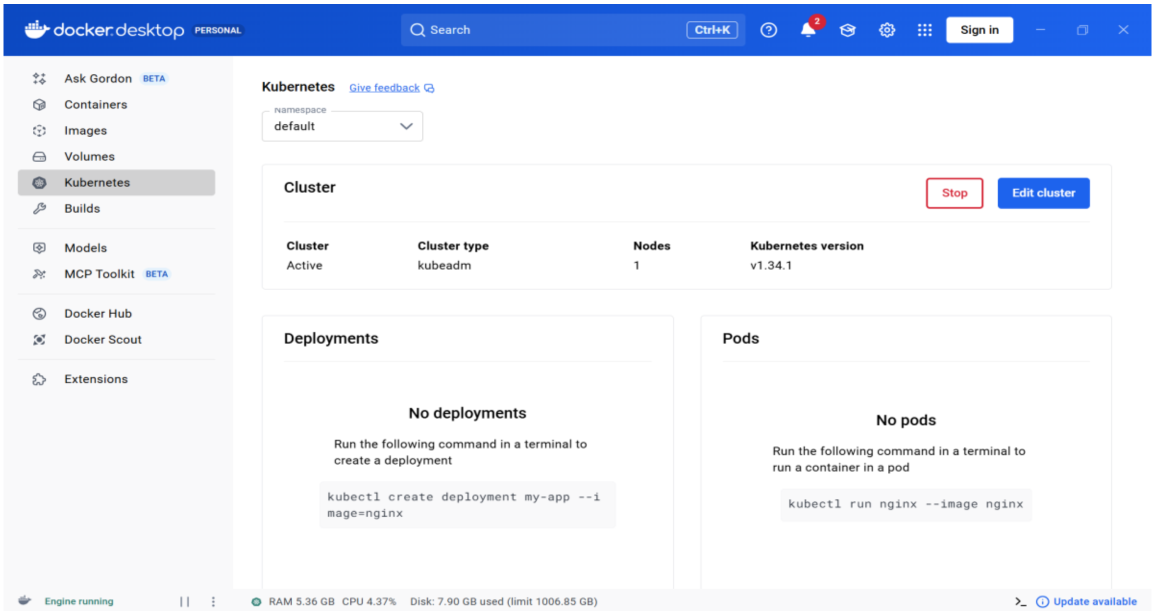
● Kubeadm
Create a single-node cluster with kubeadm.
Version: v1.34.1

○ kind
Create a cluster containing one or more nodes with kind. Requires the containerd image store

Advanced Settings

□ Show system containers (advanced)
Show Kubernetes internal containers when using Docker commands.

RAM 2.72 GB CPU 0.12% Disk: 5.96 GB used (limit 1006.85 GB)



Namespace: default

Cluster

Cluster	Cluster type	Nodes	Kubernetes version
Active	kubeadm	1	v1.34.1

Deployments

No deployments

Run the following command in a terminal to create a deployment

```
kubectl create deployment my-app --image=nginx
```

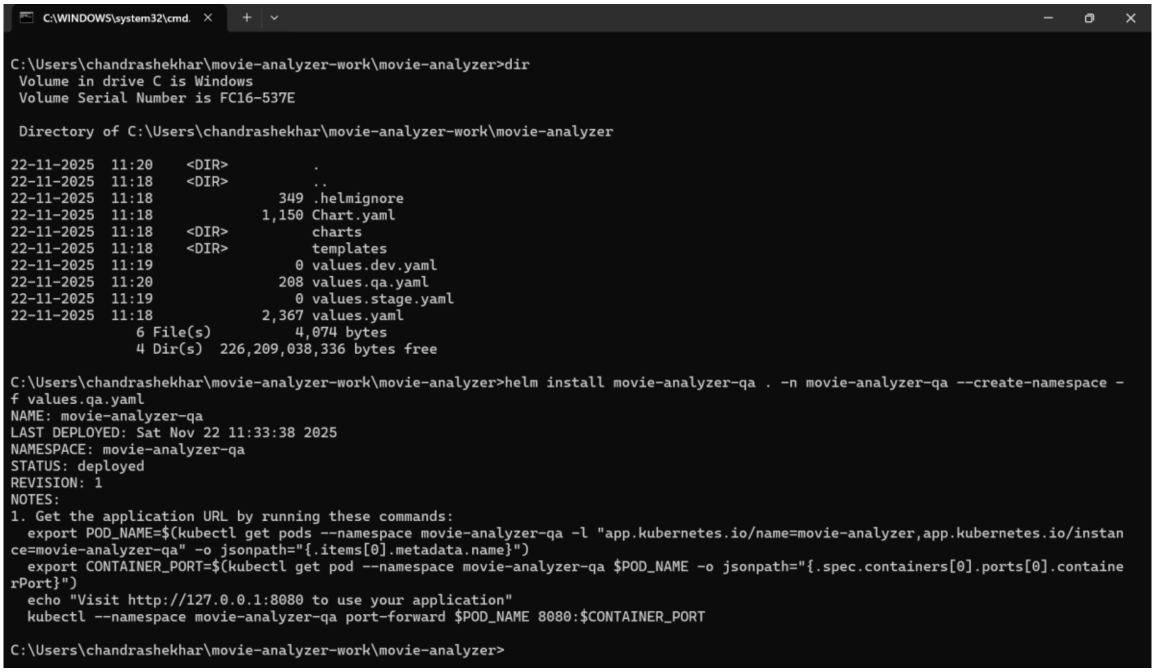
Pods

No pods

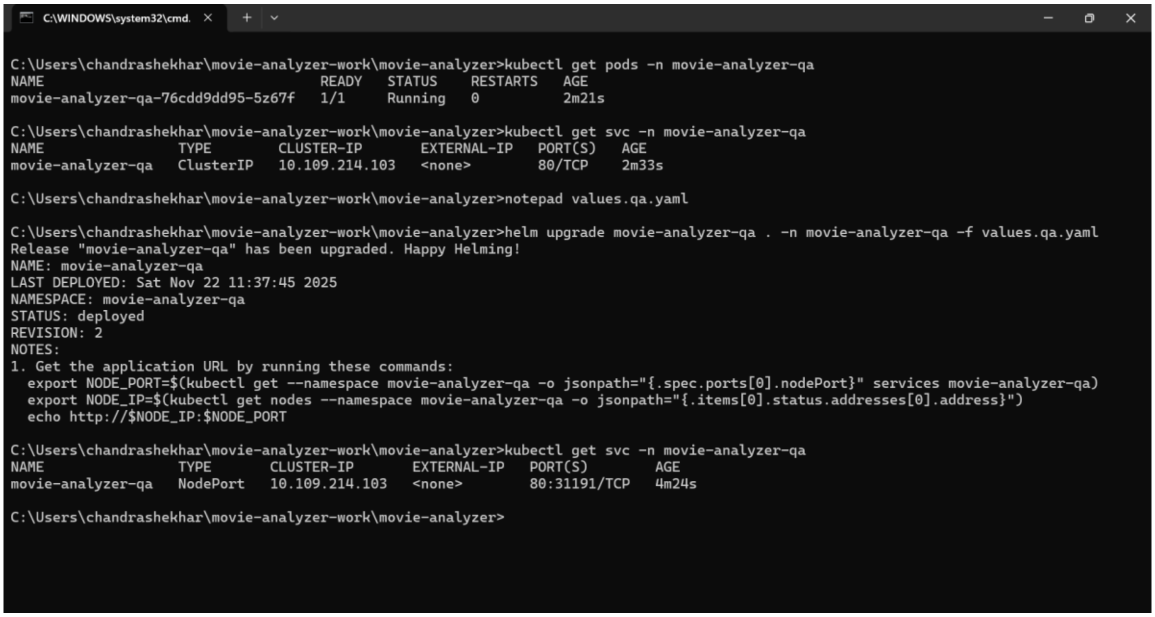
Run the following command in a terminal to run a container in a pod

```
kubectl run nginx --image nginx
```

RAM 5.36 GB CPU 4.37% Disk: 7.90 GB used (limit 1006.85 GB)



```
C:\Users\chandrashekhar\movie-analyzer-work\movie-analyzer>dir
 Volume in drive C is Windows
 Volume Serial Number is FC16-537E

 Directory of C:\Users\chandrashekhar\movie-analyzer-work\movie-analyzer

22-11-2025  11:20    <DIR>          .
22-11-2025  11:18    <DIR>          ..
22-11-2025  11:18               349 .helmignore
22-11-2025  11:18             1,150 Chart.yaml
22-11-2025  11:18    <DIR>          charts
22-11-2025  11:18    <DIR>          templates
22-11-2025  11:19                 0 values.dev.yaml
22-11-2025  11:20               208 values.qa.yaml
22-11-2025  11:19                 0 values.stage.yaml
22-11-2025  11:18             2,367 values.yaml
               6 File(s)          4,074 bytes
               4 Dir(s)  226,209,038,336 bytes free

C:\Users\chandrashekhar\movie-analyzer-work\movie-analyzer>helm install movie-analyzer-qa . -n movie-analyzer-qa --create-namespace -
f values.qa.yaml
NAME: movie-analyzer-qa
LAST DEPLOYED: Sat Nov 22 11:33:38 2025
NAMESPACE: movie-analyzer-qa
STATUS: deployed
REVISION: 1
NOTES:
1. Get the application URL by running these commands:
  export POD_NAME=$(kubectl get pods --namespace movie-analyzer-qa -l "app.kubernetes.io/name=movie-analyzer,app.kubernetes.io/instan
ce=movie-analyzer-qa" -o jsonpath="{.items[0].metadata.name}")
  export CONTAINER_PORT=$(kubectl get pod --namespace movie-analyzer-qa $POD_NAME -o jsonpath="{.spec.containers[0].ports[0].containe
rPort}")
  echo "Visit http://127.0.0.1:8080 to use your application"
  kubectl --namespace movie-analyzer-qa port-forward $POD_NAME 8080:$CONTAINER_PORT

C:\Users\chandrashekhar\movie-analyzer-work\movie-analyzer>
```



```
C:\Users\chandrashekhar\movie-analyzer-work\movie-analyzer>kubectl get pods -n movie-analyzer-qa
NAME                                 READY   STATUS    RESTARTS   AGE
movie-analyzer-qa-76cdd9dd95-5z67f   1/1     Running   0          2m21s

C:\Users\chandrashekhar\movie-analyzer-work\movie-analyzer>kubectl get svc -n movie-analyzer-qa
NAME                TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)   AGE
movie-analyzer-qa   ClusterIP   10.109.214.103  <none>        80/TCP    2m33s

C:\Users\chandrashekhar\movie-analyzer-work\movie-analyzer>notepad values.qa.yaml

C:\Users\chandrashekhar\movie-analyzer-work\movie-analyzer>helm upgrade movie-analyzer-qa . -n movie-analyzer-qa -f values.qa.yaml
Release "movie-analyzer-qa" has been upgraded. Happy Helming!
NAME: movie-analyzer-qa
LAST DEPLOYED: Sat Nov 22 11:37:45 2025
NAMESPACE: movie-analyzer-qa
STATUS: deployed
REVISION: 2
NOTES:
1. Get the application URL by running these commands:
  export NODE_PORT=$(kubectl get --namespace movie-analyzer-qa -o jsonpath="{.spec.ports[0].nodePort}" services movie-analyzer-qa)
  export NODE_IP=$(kubectl get nodes --namespace movie-analyzer-qa -o jsonpath="{.items[0].status.addresses[0].address}")
  echo http://$NODE_IP:$NODE_PORT

C:\Users\chandrashekhar\movie-analyzer-work\movie-analyzer>kubectl get svc -n movie-analyzer-qa
NAME                TYPE       CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
movie-analyzer-qa   NodePort   10.109.214.103   <none>        80:31191/TCP   4m24s

C:\Users\chandrashekhar\movie-analyzer-work\movie-analyzer>
```

Steps Performed

1. Create Helm Chart

A new Helm chart was created using the following command:

helm create movie-analyzer

2. Create Environment Values Files

Environment-specific values files were created using:

```
touch values.dev.yaml values.stage.yaml values.qa.yaml
```

3. Configure QA Environment (values.qa.yaml)

Below is the configuration used for the QA environment:

```
global:
  namespace: movie-analyzer-qa

backend:
  replicaCount: 1
  image:
    repository: nginx
    tag: latest

service:
  type: NodePort
  port: 80
  targetPort: 80
  nodePort: 31191
```

4. Install QA Environment using Helm

The QA environment was deployed using:

```
helm install movie-analyzer-qa . -n movie-analyzer-qa --create-namespace -f values.qa.yaml
```

5. Verify Deployment

Pods verification command:

```
kubectl get pods -n movie-analyzer-qa
```

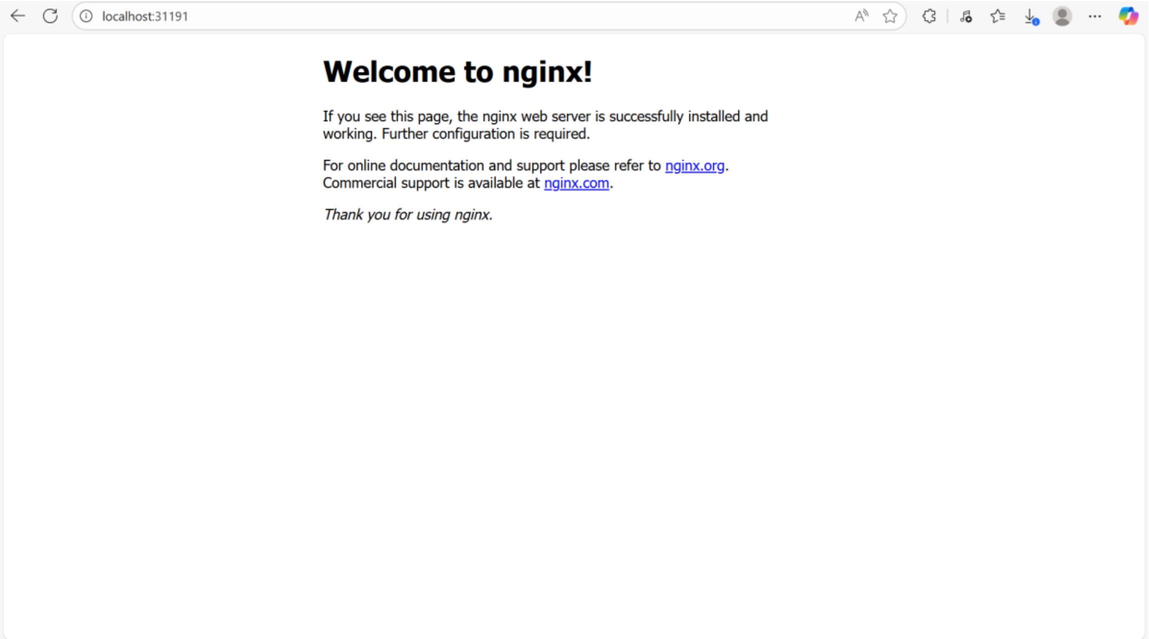
Service verification command:

```
kubectl get svc -n movie-analyzer-qa
```

6. Access Application

The application was accessed successfully:

http://localhost:31191



localhost:31191

Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

Conclusion

The QA environment was successfully created and deployed using Helm on a Docker Desktop Kubernetes cluster. All components worked correctly, and the application was accessed through the NodePort service.