Clone the repository:

git clone <https://github.com/swimlane/devops-practical>

podman build -t my-image .

podman run -d -p 8080:80 my-image

podman ps

<http://localhost:8080>

1. Access your containerized application by navigating to **http://localhost:8080** in your web browser.

MongoDB Docker Image:

podman pull docker.io/library/mongo

podman run -d --name mongodb -p 27017:27017 docker.io/library/mongo

Container’s Created:

PS C:\$18\_Practicals> podman ps

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

9d5e1cbe7598 localhost/ang-image:latest npm start 22 minutes ago Up 22 minutes 0.0.0.0:8080->80/tcp angry\_mclean

PS C:\$18\_Practicals> podman run -d --name mongodb -p 27017:27017 docker.io/library/mongo

c89ab19a4fdbb694f95964778c89b8ac081ec089fabdaeedaedf9ce014faeb0f

PS C:\$18\_Practicals>

PS C:\$18\_Practicals> podman ps

**CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES**

9d5e1cbe7598 localhost/ang-image:latest npm start 22 minutes ago Up 22 minutes 0.0.0.0:8080->80/tcp angry\_mclean

c89ab19a4fdb docker.io/library/mongo:latest mongod 5 seconds ago Up 6 seconds 0.0.0.0:27017->27017/tcp mongodb

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Created AKS cluster:

PS /home/pavan> az account set --subscription b4ee58f1-7bfb-4348-a5e9-18b5d8284c8d

PS /home/pavan> az aks get-credentials --resource-group vmfree-pavan --name swimlane --overwrite-existing

Merged "swimlane" as current context in /home/pavan/.kube/config

A screenshot of a computer

Description automatically generated

kubectl get deployments --all-namespaces=true

PS /home/pavan/devops-practical> helm package helm-chart

Successfully packaged chart and saved it to: /home/pavan/devops-practical/my-devops-practical-chart-1.0.0.tgz

PS /home/pavan/devops-practical>

**PS /home/pavan/devops-practical/helm-chart>**

**PS /home/pavan/devops-practical/helm-chart> ls**

**Chart.yaml templates**

apiVersion: v2

name: my-devops-practical-chart

description: A Helm chart for the DevOps Practical application

version: 1.0.0

PS /home/pavan/devops-practical> helm install my-devops-practical-chart my-devops-practical-chart-1.0.0.tgz

NAME: my-devops-practical-chart

LAST DEPLOYED: Sun May 5 17:12:13 2024

NAMESPACE: default

STATUS: deployed

REVISION: 1

TEST SUITE: None

**Create a Helm Chart:** You can create a Helm chart for your application by organizing your application's Kubernetes manifests into a specific directory structure. Here's a basic example of a Helm chart directory structure:

perl

Copy code

my-chart/ ├── Chart.yaml ├── values.yaml ├── templates/ │ ├── deployment.yaml │ ├── service.yaml │ └── ingress.yaml └── charts/

PS /home/pavan/devops-practical> ls

app Chart.yaml config ecosystem.config.js helm-chart LICENSE.txt my-devops-practical-chart-1.0.0.tgz package.json package-lock.json Procfile public README.md server.js test values.yaml

PS /home/pavan/devops-practical> helm install swimlanerelease ./helm-chart

NAME: swimlanerelease

LAST DEPLOYED: Sun May 5 17:15:12 2024

NAMESPACE: default

STATUS: deployed

REVISION: 1

TEST SUITE: None

PS /home/pavan/devops-practical>

**Security:**

**Get Cluster Credentials:**

**Obtain the necessary credentials (e.g., certificate authority, server URL, token) to authenticate with your Kubernetes cluster. These credentials are typically provided by your cluster administrator or obtained from your cloud provider's dashboard.**

**Set Cluster Context:**

**Use the kubectl config set-cluster command to define the cluster configuration. Replace cluster-name, cluster-server, and cluster-certificate-authority-data with the appropriate values:**

**kubectl config set-credentials cluster-username \**

**--token=your-auth-token**

**Set Authentication Credentials:** Use the **kubectl config set-credentials** command to configure authentication. Depending on your cluster setup, you may use a token, username/password, or client certificate for authentication:

Use the **kubectl config set-context** command to create a context that combines the cluster and authentication information:

kubectl config set-cluster cluster-name \

--server=cluster-server \

--certificate-authority=cluster-certificate-authority-data

kubectl config set-credentials cluster-username \

--token=your-auth-token

Errors:

PS C:\1\_Configs> kubectl get deployments --all-namespaces=true

E0506 21:25:48.145831 57480 memcache.go:265] couldn't get current server API group list: Get "https://swimlane-dns-18hm2u8f.hcp.eastus2.azmk8s.io:443/api?timeout=32s": read tcp 10.180.149.201:59843->52.184.219.137:443: wsarecv: An existing connection was forcibly closed by the remote host.

E0506 21:25:48.654300 57480 memcache.go:265] couldn't get current server API group list: Get "https://swimlane-dns-18hm2u8f.hcp.eastus2.azmk8s.io:443/api?timeout=32s": tls: failed to verify certificate: x509: certificate signed by unknown authority

E0506 21:25:49.151880 57480 memcache.go:265] couldn't get current server API group list: Get "https://swimlane-dns-18hm2u8f.hcp.eastus2.azmk8s.io:443/api?timeout=32s": tls: failed to verify certificate: x509: certificate signed by unknown authority

E0506 21:25:49.637189 57480 memcache.go:265] couldn't get current server API group list: Get "https://swimlane-dns-18hm2u8f.hcp.eastus2.azmk8s.io:443/api?timeout=32s": tls: failed to verify certificate: x509: certificate signed by unknown authority

E0506 21:25:50.129262 57480 memcache.go:265] couldn't get current server API group list: Get "https://swimlane-dns-18hm2u8f.hcp.eastus2.azmk8s.io:443/api?timeout=32s": tls: failed to verify certificate: x509: certificate signed by unknown authority

Unable to connect to the server: tls: failed to verify certificate: x509: certificate signed by unknown authority

C:\$18\_Practicals>npm start

> swimlane-devops-practical@4.0.0 start

> cross-env NODE\_ENV=development nodemon server.js

[nodemon] 2.0.2

[nodemon] to restart at any time, enter `rs`

[nodemon] watching dir(s): \*.\*

[nodemon] watching extensions: js,mjs,json

[nodemon] starting `node server.js`

(node:15440) Warning: Accessing non-existent property 'count' of module exports inside circular dependency

(Use `node --trace-warnings ...` to show where the warning was created)

(node:15440) Warning: Accessing non-existent property 'findOne' of module exports inside circular dependency

(node:15440) Warning: Accessing non-existent property 'remove' of module exports inside circular dependency

(node:15440) Warning: Accessing non-existent property 'updateOne' of module exports inside circular dependency

C:\$18\_Practicals\node\_modules\mongodb\lib\utils.js:725

throw error;

^

MongoServerSelectionError: connection timed out

at Timeout.\_onTimeout (C:\$18\_Practicals\node\_modules\mongodb\lib\core\sdam\topology.js:430:30)

at listOnTimeout (node:internal/timers:573:17)

at process.processTimers (node:internal/timers:514:7) {

reason: TopologyDescription {

type: 'Single',

setName: null,

maxSetVersion: null,

maxElectionId: null,

servers: Map(1) {

'mongo:27017' => ServerDescription {

address: 'mongo:27017',

error: MongoNetworkError: connection timed out

at connectionFailureError (C:\$18\_Practicals\node\_modules\mongodb\lib\core\connection\connect.js:406:14)

at Socket.<anonymous> (C:\$18\_Practicals\node\_modules\mongodb\lib\core\connection\connect.js:294:16)

at Object.onceWrapper (node:events:633:28)

at Socket.emit (node:events:519:28)

at Socket.\_onTimeout (node:net:590:8)

at listOnTimeout (node:internal/timers:573:17)

at process.processTimers (node:internal/timers:514:7),

roundTripTime: -1,

lastUpdateTime: 78761790,

lastWriteDate: null,

opTime: null,

type: 'Unknown',

minWireVersion: 0,

maxWireVersion: 0,

hosts: [],

passives: [],

arbiters: [],

tags: []

}

},

stale: false,

compatible: true,

compatibilityError: null,

logicalSessionTimeoutMinutes: null,

heartbeatFrequencyMS: 10000,

localThresholdMS: 15,

commonWireVersion: null

}

}

Node.js v20.13.0