To deploy your microservices (e.g., carts, orders, users) with MongoDB instances on \*\*Kubernetes\*\*, follow these steps to build the architecture and ensure high availability. This approach includes separating your MongoDB databases per service and automating your deployments with Kubernetes tools.

---

## \*\*Step 1: Prepare Kubernetes Cluster\*\*

1. If you don't have a cluster, you can set one up locally with:

- \*\*Minikube:\*\* for local testing.

- \*\*Kind:\*\* Kubernetes in Docker.

- \*\*Cloud Providers:\*\* GKE (Google Cloud), EKS (AWS), or AKS (Azure).

2. Ensure `kubectl` is installed and connected to the cluster:

```bash

kubectl cluster-info

```

---

## \*\*Step 2: Create Docker Images for Microservices\*\*

1. Build Docker images for each microservice and push them to a container registry (like DockerHub or a private registry):

```bash

docker build -t your-repo/carts ./carts

docker build -t your-repo/orders ./orders

docker push your-repo/carts

docker push your-repo/orders

```

---

## \*\*Step 3: Write Kubernetes Manifests\*\*

Each service will require \*\*Deployments\*\*, \*\*Services\*\*, and \*\*ConfigMaps\*\* or \*\*Secrets\*\*. Below are templates to deploy your microservices and MongoDB instances.

### \*\*Deployment for MongoDB (Per Service)\*\*

Create one MongoDB instance per microservice. Example for `carts-db`:

```yaml

# carts-mongo-deployment.yaml

apiVersion: apps/v1

kind: Deployment

metadata:

name: carts-mongo

spec:

replicas: 1

selector:

matchLabels:

app: carts-mongo

template:

metadata:

labels:

app: carts-mongo

spec:

containers:

- name: mongo

image: mongo:latest

ports:

- containerPort: 27017

env:

- name: MONGO\_INITDB\_ROOT\_USERNAME

value: "admin"

- name: MONGO\_INITDB\_ROOT\_PASSWORD

value: "secret"

- name: MONGO\_INITDB\_DATABASE

value: "carts-db"

---

apiVersion: v1

kind: Service

metadata:

name: carts-mongo

spec:

selector:

app: carts-mongo

ports:

- protocol: TCP

port: 27017

targetPort: 27017

```

Apply the MongoDB manifest:

```bash

kubectl apply -f carts-mongo-deployment.yaml

```

---

### \*\*Deployment for Microservices\*\*

Example for deploying the `carts` microservice:

```yaml

# carts-deployment.yaml

apiVersion: apps/v1

kind: Deployment

metadata:

name: carts

spec:

replicas: 2

selector:

matchLabels:

app: carts

template:

metadata:

labels:

app: carts

spec:

containers:

- name: carts

image: your-repo/carts:latest

ports:

- containerPort: 8080

env:

- name: QUARKUS\_MONGODB\_CONNECTION\_STRING

value: "mongodb://admin:secret@carts-mongo:27017/carts-db?authSource=admin"

---

apiVersion: v1

kind: Service

metadata:

name: carts

spec:

selector:

app: carts

ports:

- protocol: TCP

port: 8080

targetPort: 8080

```

Deploy it to the cluster:

```bash

kubectl apply -f carts-deployment.yaml

```

---

### \*\*Step 4: Automate Configuration Using ConfigMaps or Secrets\*\*

Store sensitive information using \*\*Secrets\*\*:

```bash

kubectl create secret generic mongo-credentials \

--from-literal=username=admin \

--from-literal=password=secret

```

Reference them in the deployment:

```yaml

env:

- name: MONGO\_USERNAME

valueFrom:

secretKeyRef:

name: mongo-credentials

key: username

- name: MONGO\_PASSWORD

valueFrom:

secretKeyRef:

name: mongo-credentials

key: password

```

---

### \*\*Step 5: Expose Services Using Ingress\*\*

Use \*\*Ingress\*\* to route traffic to different services from a single endpoint:

```yaml

apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

name: microservices-ingress

spec:

rules:

- host: example.com

http:

paths:

- path: /carts

pathType: Prefix

backend:

service:

name: carts

port:

number: 8080

- path: /orders

pathType: Prefix

backend:

service:

name: orders

port:

number: 8080

```

Apply it:

```bash

kubectl apply -f ingress.yaml

```

---

### \*\*Step 6: Enable Auto-Scaling with Horizontal Pod Autoscaler (HPA)\*\*

Set up HPA to automatically scale pods based on CPU usage:

```bash

kubectl autoscale deployment carts --cpu-percent=50 --min=1 --max=10

```

---

### \*\*Step 7: Use Helm for Simplified Management (Optional)\*\*

You can create Helm charts to manage these deployments more easily across environments.

---

### \*\*Step 8: Set Up CI/CD Pipeline\*\*

To automate deployments:

- Use \*\*Jenkins\*\*, \*\*GitHub Actions\*\*, or \*\*GitLab CI\*\*.

- Trigger deployments on commits to your microservices.

Example `.github/workflows/deploy.yaml` for GitHub Actions:

```yaml

name: Deploy to Kubernetes

on:

push:

branches:

- main

jobs:

deploy:

runs-on: ubuntu-latest

steps:

- name: Checkout code

uses: actions/checkout@v2

- name: Set up kubectl

uses: azure/setup-kubectl@v1

with:

version: 'v1.25.4'

- name: Deploy to Kubernetes

run: |

kubectl apply -f carts-deployment.yaml

kubectl apply -f orders-deployment.yaml

```

---

This setup ensures each microservice gets its own MongoDB instance, follows Kubernetes best practices, and enables scalability with HPA. Let me know if you encounter any issues!