Cloud

Final Project : Cloud Computing ECE 9016

Saeid Abedi: sabedi4@uwo.ca

Stephan Dhanraj: sdhanraj@uwo.ca

James Zhong: pzhong22@uwo.ca

Reza Motazed Monadjemi: rmotazed@uwo.ca

James Suresh: jsuresh3@uwo.ca

Group 07

Computing

Introduction

The project focuses on a simple web application. This application interacts with a deployed database container. The report covers container development and deployment processes. Deployment is to a Kubernetes cluster. Environments covered include both development and production settings.

Intro

Technologies Used

01 Google Cloud

02 Kubernetes (GKE)

03 Docker

04 Node.js

05 MongoDB

Containerized Application

Our application is structured around two main containers:

- a back-end MongoDB container for data storage
- a front-end Node.js container responsible for serving HTML and making database calls.

Cloud Computing

Cloud

Back-end Container

Uses standard MongoDB image from DockerHub for consistency.

Requires a locally mounted volume for data storage.

Data accessibility ensured through mounted volume.

YAML configuration files detail the volume mounting process.

Computing

Front-end Container

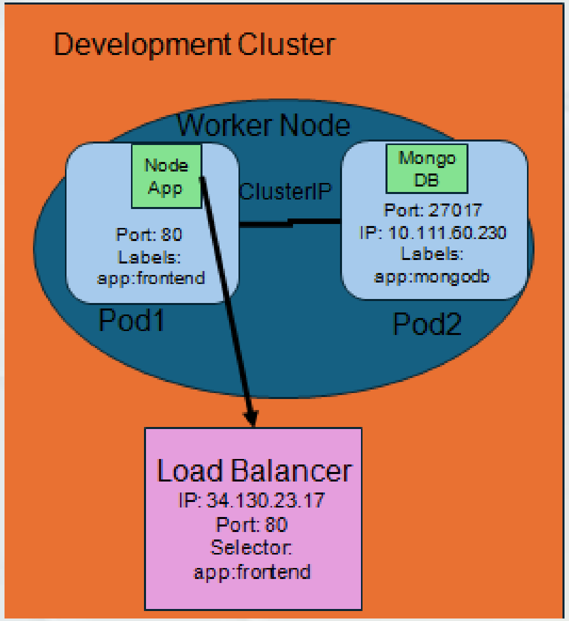
Containers handle HTML serving and database requests.

Utilizes Next.js server due to its MongoDB driver support.

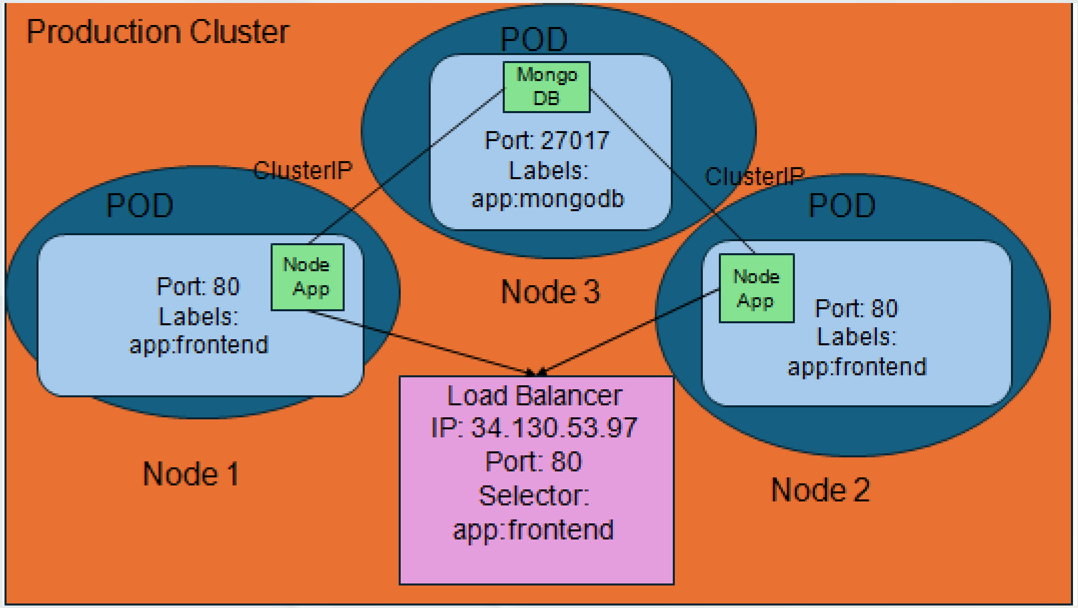
Installation of new dependencies, including Next and MongoClient, required.

Front-end container built on the standard Node.js image.

Development Environment



Production Environment



Deployment steps of Dev and Prod Env

1. Deploy the cluster
2. Mount the PVC using the PVC yaml file
3. Deploy the mongodb database pod
4. Create the mongodb clusterIP service
5. Give the docker image the IP of the cluster via the clusterIP
6. Build the new frontend container with the IP
7. Tag and push the new container image
8. Deploy the front-end pod
9. Deploy the front-end load balancer

The difference in the deployment is in the frontend of the production yaml file.

- gcloud container clusters **create** dev-**cluster**-realestate --num-nodes 1 --machine-type e2-standard-4
- Kubectl apply –f mongodb-pvc.yaml
- Kubectl apply –f db-deployment.yaml
- Kubectl apply –f mongodb service_ClusterIP.yaml
- docker build -t northamerica-northeast2-docker.pkg.dev/real-estate-group-416222/docker-rep3/frontend:latest .
- docker push northamerica-northeast2-docker.pkg.dev/real-estate-group-416222/docker-rep3/frontend:latest
- kubectl apply -f frontend-deployment.yaml
- kubectl apply -f frontend-service.yaml

Cloud

YAML Descriptions

Database Deployment Manifest:

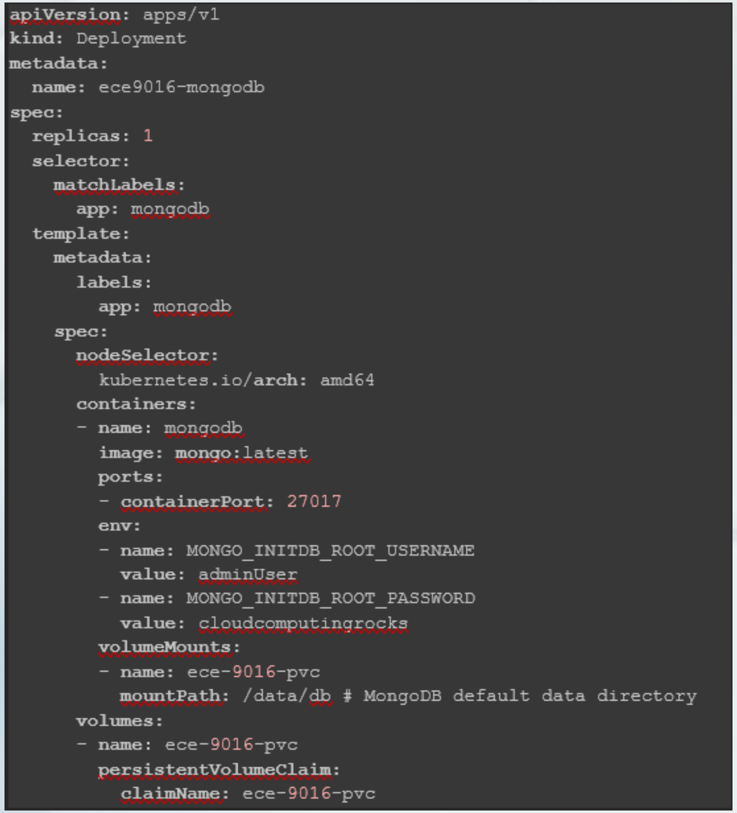
- Describes a Kubernetes YAML file for MongoDB setup named 'ece9016-mongodb'.
- Deployment utilizes one instance with the 'mongodb' label.
- Details include a MongoDB container on its standard port, secured with username and password.
- Connects to a persistent storage volume for data integrity across restarts.

Volume Mounting Manifest

- Requests Kubernetes to reserve persistent storage space via a PVC named 'ece-9016-pvc'.
- Ensures consistent data storage/retrieval, even with container changes.
- PVC seeks 10GB of 'ReadWriteOnce' storage, allowing single-pod read/write access.
- Specifies 'standard' storage class, common in Kubernetes, for reliable data maintenance.

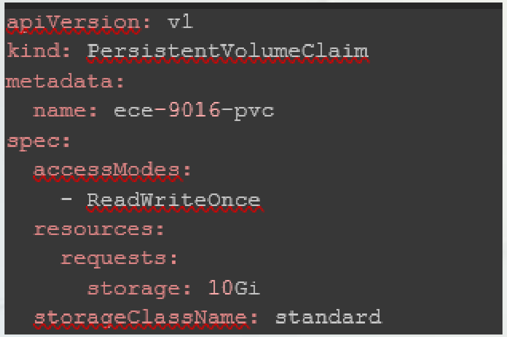
Computing

Database Deployment Manifest



```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: ece9016-mongodb
spec:
  replicas: 1
  selector:
    matchLabels:
      app: mongodb
  template:
    metadata:
      labels:
        app: mongodb
    spec:
      nodeSelector:
        kubernetes.io/arch: amd64
      containers:
      - name: mongodb
        image: mongo:latest
        ports:
        - containerPort: 27017
        env:
        - name: MONGO_INITDB_ROOT_USERNAME
          value: adminUser
        - name: MONGO_INITDB_ROOT_PASSWORD
          value: cloudcomputingrocks
        volumeMounts:
        - name: ece-9016-pvc
          mountPath: /data/db # MongoDB default data directory
      volumes:
      - name: ece-9016-pvc
        persistentVolumeClaim:
          claimName: ece-9016-pvc
```

Volume Mounting Manifest



```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: ece-9016-pvc
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
  storageClassName: standard
```

Cloud

YAML Descriptions

Database ClusterIP service Manifest:

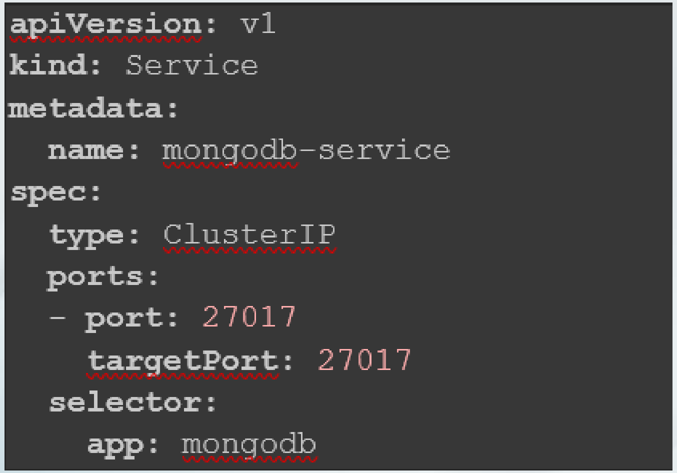
- Configures a Kubernetes network service 'mongodb-service' for MongoDB communication.
- Service lacks an external IP but is internally accessible within the cluster.
- Listens on MongoDB's standard port 27017, routing traffic to the matching pod.
- Uses 'app: mongodb' label to identify and direct traffic to the correct MongoDB pod.

Development Environment Frontend Deployment Manifest:

- Outlines Kubernetes config for a single-instance 'frontend' container service.
- Utilizes 'replicas: 1' for a solitary front-end instance in development.
- Labels deployment and pods as 'app: frontend' for simplified Kubernetes management.
- Configures to use a specific, latest Docker image from a repository for the front-end.
- Sets the container to listen on port 80 for HTTP traffic, ensuring web accessibility.
- Employs 'hostPort' to align container's port 80 with the host's, enabling direct host access.
- Designed to facilitate easy testing and development with the most updated front-end version.

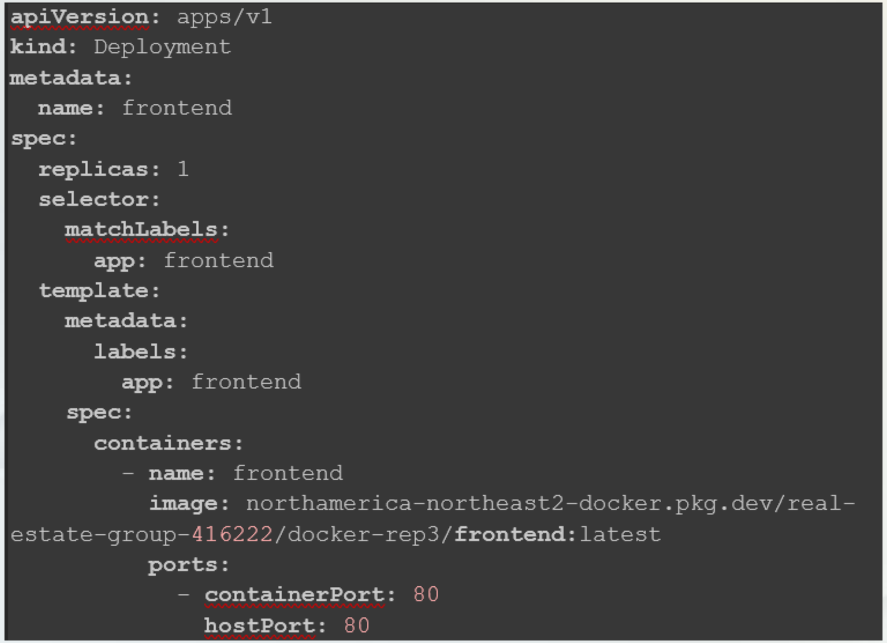
Computing

Database ClusterIP service Manifest



```yaml
apiVersion: v1
kind: Service
metadata:
  name: mongodb-service
spec:
  type: ClusterIP
  ports:
  - port: 27017
    targetPort: 27017
  selector:
    app: mongodb
```

Development Environment Frontend Deployment Manifest:



```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: frontend
spec:
  replicas: 1
  selector:
    matchLabels:
      app: frontend
  template:
    metadata:
      labels:
        app: frontend
    spec:
      containers:
        - name: frontend
          image: northamerica-northeast2-docker.pkg.dev/real-
estate-group-416222/docker-rep3/frontend:latest
          ports:
            - containerPort: 80
              hostPort: 80
```

Cloud

YAML Descriptions

Frontend Load Balancer Manifest

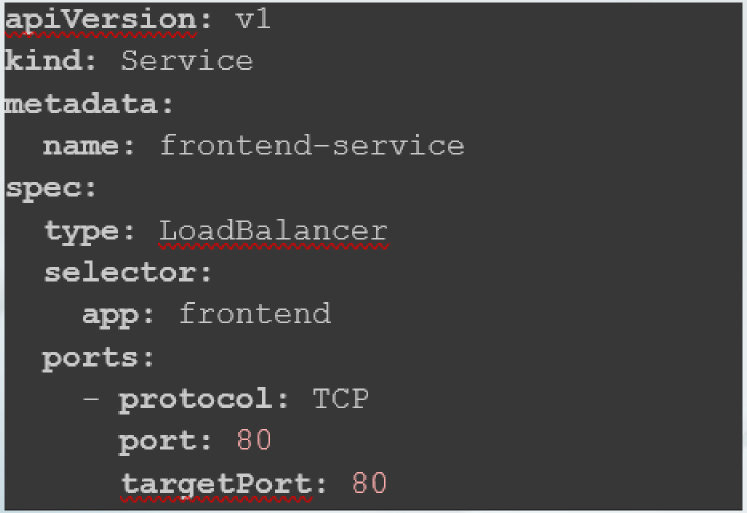
- Configures 'frontend-service' as a Kubernetes LoadBalancer to distribute web traffic.
- Enables external traffic reception on port 80, directing it to front-end containers.
- Searches for pods labeled 'app: frontend' for traffic distribution.
- Aims for efficient traffic management and enhanced application availability.

Production Environment Frontend Manifest

- Sets up two instances of the front-end in the production environment for better availability.
- Uses 'frontend' as the deployment name with the latest Docker image for consistency.
- Configures each front-end instance to operate on port 80 for standard web traffic handling.
- Employs 'affinity' settings, specifically 'podAntiAffinity', to distribute instances across different physical machines, enhancing reliability and failure resilience.

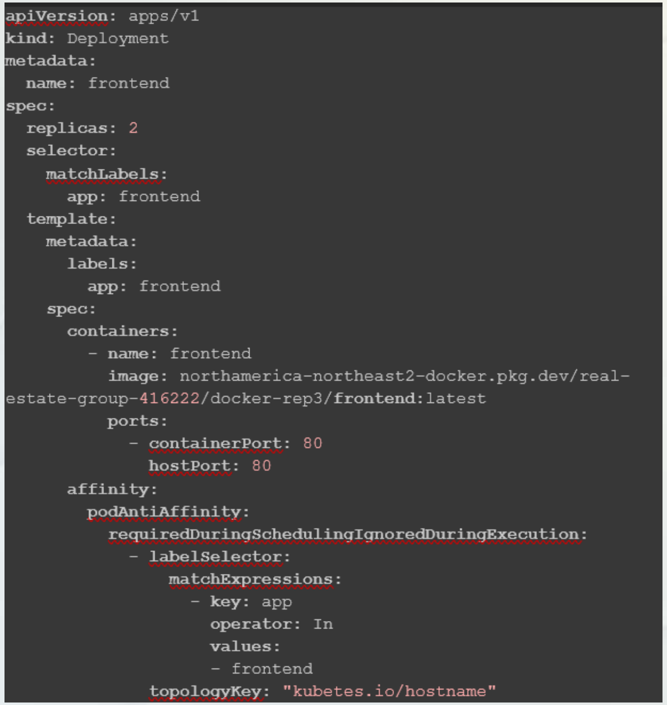
Computing

Frontend Load Balancer Manifest



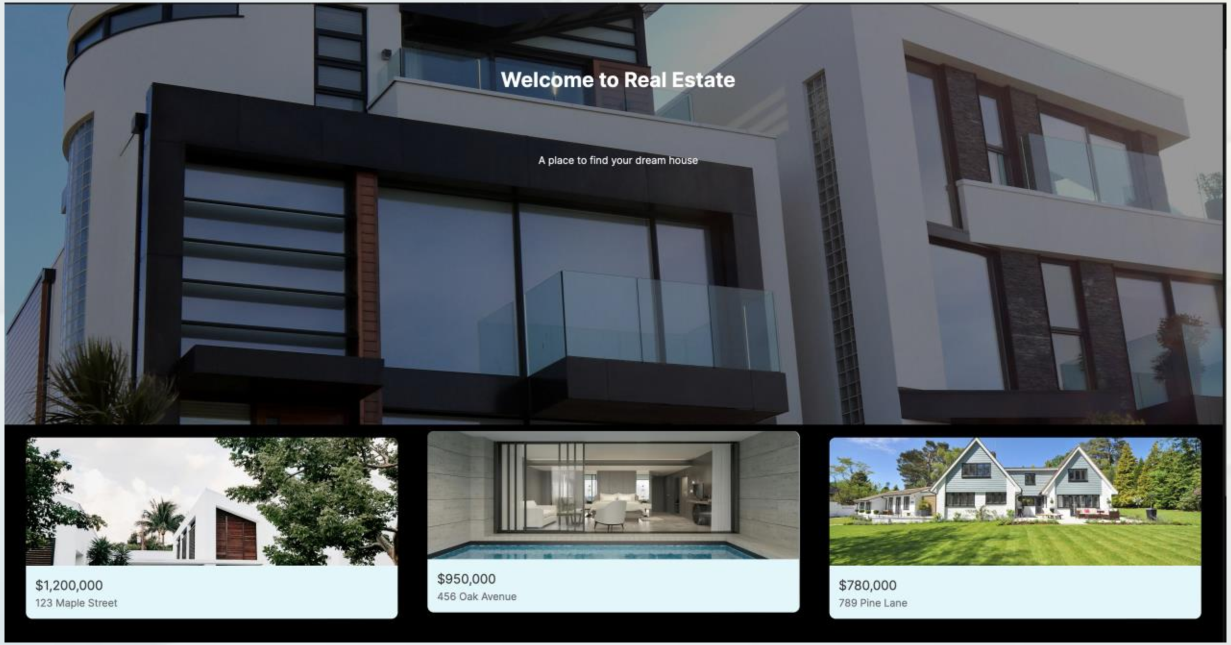
```yaml
apiVersion: v1
kind: Service
metadata:
  name: frontend-service
spec:
  type: LoadBalancer
  selector:
    app: frontend
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
```

Production Environment Frontend Manifest



```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: frontend
spec:
  replicas: 2
  selector:
    matchLabels:
      app: frontend
  template:
    metadata:
      labels:
        app: frontend
    spec:
      containers:
        - name: frontend
          image: northamerica-northeast2-docker.pkg.dev/real-
estate-group-416222/docker-rep3/frontend:latest
          ports:
            - containerPort: 80
              hostPort: 80
      affinity:
        podAntiAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            - labelSelector:
                matchExpressions:
                  - key: app
                    operator: In
                    values:
                    - frontend
              topologyKey: "kubetes.io/hostname"
```

Site



Production endpoint - http://34.130.53.97/
Dev endpoint – http://34.130.23.17/

Thank You!
question?