Project 01

In this project, you will develop a simple Node.js application, deploy it on a local Kubernetes cluster using Minikube, and configure various Kubernetes features. The project includes Git version control practices, creating and managing branches, and performing rebases. Additionally, you will work with ConfigMaps, Secrets, environment variables, and set up vertical and horizontal pod autoscaling.

Project 01

Project Steps

1. Setup Minikube and Git Repository

Start Minikube:

```
minikube start
```

1.2 Set Up Git Repository
Create a new directory for your project:

```
mkdir nodejs-k8s-project
cd nodejs-k8s-project
```

Initialize Git repository:

```
git init
```

Create a .gitignore file:

```
node_modules/
.env
```

Add and commit initial changes:

```
git add .
git commit -m "Initial commit"
```

```
vagrant@ubuntu2204:~/nodejs-k8s-project$ nano .gitignore
vagrant@ubuntu2204:~/nodejs-k8s-project$ git add .
vagrant@ubuntu2204:~/nodejs-k8s-project$ git commit -m "Initial commit"
[main (root-commit) 44f243d] Initial commit
 1 file changed, 3 insertions(+)
 create mode 100644 .gitignore
```

2. Develop a Node.js Application

2.1 Create the Node.js App

Initialize the Node.js project:

```
npm init -y
```

Install necessary packages:

```
npm install express body-parser
```

```
vagrant@ubuntu2204:~/nodejs-k8s-project$ npm init -y
Wrote to /home/vagrant/nodejs-k8s-project/package.json:

{
  "name": "nodejs-k8s-project",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "description": ""
}



vagrant@ubuntu2204:~/nodejs-k8s-project$ npm install express body-parser

added 64 packages, and audited 65 packages in 6s

12 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

Create app.js:

```
const express = require('express');
```

```
const bodyParser = require('body-parser');
const app = express();
const PORT = process.env.PORT || 3000;

app.use(bodyParser.json());

app.get('/', (req, res) => {
  res.send('Hello, World!');
});

app.listen(PORT, () => {
  console.log(`Server is running on port ${PORT}`);
});
```

Update `package.json` to include a start script:

```
"scripts": {
  "start": "node app.js"
}
```

2.2 Commit the Node.js Application

Add and commit changes:

```
git add .
git commit -m "Add Node.js application code"
```

3. Create Dockerfile and Docker Compose

3.1 Create a `Dockerfile`

Add `Dockerfile`:

```
# Use official Node.js image
FROM node:18

# Set the working directory
WORKDIR /usr/src/app

# Copy package.json and package-lock.json
COPY package*.json ./

# Install dependencies
RUN npm install
```

```
# Copy the rest of the application code
COPY . .

# Expose the port on which the app runs
EXPOSE 3000

# Command to run the application
CMD [ "npm", "start" ]
```

Create a .dockerignore file:

```
node_modules
.npm
```

3.2 Create docker-compose.yml (optional for local testing)
Add docker-compose.yml:

```
version: '3'
services:
  app:
    build: .
    ports:
      - "3000:3000"
```

Add and commit changes:

```
git add Dockerfile docker-compose.yml
git commit -m "Add Dockerfile and Docker Compose configuration"
```

4. Build and Push Docker Image

4.1 Build Docker Image
Build the Docker image:

```
docker build -t nodejs-app:latest .
```

```
vagrant@ubuntu2204:~/DevOps-class-assessment/Assessment9/nodejs-k8s-project$ docker build -t nodejs-app:latest
.
[+] Building 67.2s (11/11) FINISHED                                                                 docker:default
 => [internal] load build definition from Dockerfile                                                         0.0s
 => => transferring dockerfile: 398B                                                                         0.0s
 => [internal] load metadata for docker.io/library/node:18                                                   1.8s
 => [auth] library/node:pull token for registry-1.docker.io                                                  0.0s
 => [internal] load .dockerignore                                                                            0.0s
 => => transferring context: 58B                                                                             0.0s
 => [1/5] FROM docker.io/library/node:18@sha256:aabbaf118c7c0a6e9a3bda69bd2a94b0f6c4150bf80c501ef1c87ff       0.0s
 => [internal] load build context                                                                            0.0s
 => => transferring context: 3.26kB                                                                          0.0s
 => CACHED [2/5] WORKDIR /usr/src/app                                                                        0.0s
 => [3/5] COPY package*.json ./                                                                              0.0s
 => [4/5] RUN npm install                                                                                   65.1s
 => [5/5] COPY . .                                                                                           0.0s
 => exporting to image                                                                                       0.1s
 => => exporting layers                                                                                      0.1s
 => => writing image sha256:a034c2657275e9f40d26188052b65db289e3c35f747a2f47def5bdab8426adb1                 0.0s
 => => naming to docker.io/library/nodejs-app:latest                                                         0.0s
vagrant@ubuntu2204:~/DevOps-class-assessment/Assessment9/nodejs-k8s-project$
```

4.2 Push Docker Image to Docker Hub

Tag and push the image:

```
docker tag nodejs-app:latest your-dockerhub-username/nodejs-app:latest
docker push your-dockerhub-username/nodejs-app:latest
```

Add and commit changes:

```
git add .
git commit -m "Build and push Docker image"
```

```
vagrant@ubuntu2204:~/DevOps-class-assessment/Assessment9/nodejs-k8s-project$ docker image tag nodejs-app:latest daradesudarshan/centralrepo:nodejs-app-v1
vagrant@ubuntu2204:~/DevOps-class-assessment/Assessment9/nodejs-k8s-project$ docker push daradesudarshan/centralrepo:nodejs-app-v1
The push refers to repository [docker.io/daradesudarshan/centralrepo]
a39f2dcac1f4: Pushed
0dc026aa6f7b: Pushed
61feb683a6ae: Pushed
aab1b309311a: Pushed
0970e1a837f7: Mounted from library/node
d4061df7c236: Mounted from library/node
9487e6e19e60: Mounted from library/node
6ef00066aa6f: Mounted from library/node
b11bb163e263: Mounted from library/node
b779a72428fa: Mounted from library/node
8ada682d3780: Mounted from library/node
15bb10f9bb3a: Mounted from library/node
nodejs-app-v1: digest: sha256:47447e1fea1bcfdb6b8e93f0bf839324753c5ef560a0d8c71fbbf385a6f6c34c size: 2839
vagrant@ubuntu2204:~/DevOps-class-assessment/Assessment9/nodejs-k8s-project$ git add .
vagrant@ubuntu2204:~/DevOps-class-assessment/Assessment9/nodejs-k8s-project$ git commit -m "build and push docker image"
[main d1c821b] build and push docker image
 2 files changed, 3 insertions(+), 1 deletion(-)
 create mode 100644 .dockerignore
```

5. Create Kubernetes Configurations

5.1 Create Kubernetes Deployment

Create kubernetes/deployment.yaml:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nodejs-app-deployment
```

```
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nodejs-app
  template:
    metadata:
      labels:
        app: nodejs-app
    spec:
      containers:
      - name: nodejs-app
        image: your-dockerhub-username/nodejs-app:latest
        ports:
        - containerPort: 3000
        env:
        - name: PORT
          valueFrom:
            configMapKeyRef:
              name: app-config
              key: PORT
        - name: NODE_ENV
          valueFrom:
            secretKeyRef:
              name: app-secrets
              key: NODE_ENV
```

5.2 Create ConfigMap and Secret

Create kubernetes/configmap.yaml:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: app-config
data:
  PORT: "3000"
```

Create kubernetes/secret.yaml:

```
apiVersion: v1
kind: Secret
metadata:
  name: app-secrets
type: Opaque
data:
  NODE_ENV: cHJvZHVjdGlvbmFs # Base64 encoded value for "production"
```

Add and commit Kubernetes configurations:

```
git add kubernetes/
git commit -m "Add Kubernetes deployment, configmap, and secret"
```

```
vagrant@ubuntu2204:~/DevOps-class-assessment/Assessment9/nodejs-k8s-project$ nano kubernetes/deployment.yaml
vagrant@ubuntu2204:~/DevOps-class-assessment/Assessment9/nodejs-k8s-project$ mkdir kubernetes
vagrant@ubuntu2204:~/DevOps-class-assessment/Assessment9/nodejs-k8s-project$ ls
app.js  docker-compose.yml  Dockerfile  kubernetes  node_modules  package.json  package-lock.json
vagrant@ubuntu2204:~/DevOps-class-assessment/Assessment9/nodejs-k8s-project$ nano kubernetes/deployment.yaml
vagrant@ubuntu2204:~/DevOps-class-assessment/Assessment9/nodejs-k8s-project$ nano kubernetes/deployment.yaml
vagrant@ubuntu2204:~/DevOps-class-assessment/Assessment9/nodejs-k8s-project$ nano kubernetes/configmap.yaml
vagrant@ubuntu2204:~/DevOps-class-assessment/Assessment9/nodejs-k8s-project$ nano kubernetes/secret.yaml
vagrant@ubuntu2204:~/DevOps-class-assessment/Assessment9/nodejs-k8s-project$ git add kubernetes/
vagrant@ubuntu2204:~/DevOps-class-assessment/Assessment9/nodejs-k8s-project$ git commit -m "Add Kubernetes deployment,configmap and secret"
[main c6782d3] Add Kubernetes deployment,configmap and secret
 3 files changed, 45 insertions(+)
 create mode 100644 kubernetes/configmap.yaml
 create mode 100644 kubernetes/deployment.yaml
 create mode 100644 kubernetes/secret.yaml
vagrant@ubuntu2204:~/DevOps-class-assessment/Assessment9/nodejs-k8s-project$
```

5.3 Apply Kubernetes Configurations

Apply the ConfigMap and Secret:

```
kubectl apply -f kubernetes/configmap.yaml
kubectl apply -f kubernetes/secret.yaml
```

Apply the Deployment:

```
kubectl apply -f kubernetes/deployment.yaml
```

```
vagrant@ubuntu2204:~/DevOps-class-assessment/Assessment9/nodejs-k8s-project$ kubectl apply -f kubernetes/configmap.yaml
^[[Aconfigmap/app-config created
vagrant@ubuntu2204:~/DevOps-class-assessment/Assessment9/nodejs-k8s-project$ kubectl apply -f kubernetes/secret.yaml
secret/app-secrets created
vagrant@ubuntu2204:~/DevOps-class-assessment/Assessment9/nodejs-k8s-project$ kubectl apply -f kubernetes/deployment.yaml
deployment.apps/nodejs-app-deployment created
vagrant@ubuntu2204:~/DevOps-class-assessment/Assessment9/nodejs-k8s-project$
```

6. Implement Autoscaling

6.1 Create Horizontal Pod Autoscaler

Create kubernetes/hpa.yaml:

```
apiVersion: autoscaling/v2beta2
kind: HorizontalPodAutoscaler
metadata:
  name: nodejs-app-hpa
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: nodejs-app-deployment
```

```
  minReplicas: 2
  maxReplicas: 5
  metrics:
  - type: Resource
    resource:
      name: cpu
      target:
        type: Utilization
        averageUtilization: 50
```

Apply the HPA:

```
kubectl apply -f kubernetes/hpa.yaml
```

```
vagrant@ubuntu2204:~/DevOps-class-assessment/Assessment9/nodejs-k8s-project$ kubectl apply -f kubernetes/hpa.yaml
horizontalpodautoscaler.autoscaling/nodejs-app-hpa created
vagrant@ubuntu2204:~/DevOps-class-assessment/Assessment9/nodejs-k8s-project$
```

6.2 Create Vertical Pod Autoscaler
Create kubernetes/vpa.yaml:

```
apiVersion: autoscaling.k8s.io/v1beta2
kind: VerticalPodAutoscaler
metadata:
  name: nodejs-app-vpa
spec:
  targetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: nodejs-app-deployment
  updatePolicy:
    updateMode: "Auto"
```

Apply the VPA:

```
kubectl apply -f kubernetes/vpa.yaml
```

```
vagrant@ubuntu2204:~/DevOps-class-assessment/Assessment9/nodejs-k8s-project$ kubectl apply -f kubernetes/vpa.yaml
verticalpodautoscaler.autoscaling.k8s.io/nodejs-app-vpa created
vagrant@ubuntu2204:~/DevOps-class-assessment/Assessment9/nodejs-k8s-project$
```

7. Test the Deployment

7.1 Check the Status of Pods, Services, and HPA

Verify the Pods:

kubectl get pods

```
vagrant@ubuntu2204:~/DevOps-class-assessment/Assessment9/nodejs-k8s-project$ kubectl  get pods
NAME                                  READY   STATUS    RESTARTS   AGE
nodejs-app-deployment-696965b85b-9fm4h   1/1     Running   0          48s
nodejs-app-deployment-696965b85b-mr7l9   1/1     Running   0          48s
vagrant@ubuntu2204:~/DevOps-class-assessment/Assessment9/nodejs-k8s-project$
```

Verify the Services:

kubectl get svc

```
vagrant@ubuntu2204:~/DevOps-class-assessment/Assessment9/nodejs-k8s-project$ kubectl get svc
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP   2d
vagrant@ubuntu2204:~/DevOps-class-assessment/Assessment9/nodejs-k8s-project$
```

Verify the HPA:

kubectl get hpa

```
vagrant@ubuntu2204:~/DevOps-class-assessment/Assessment9/nodejs-k8s-project$ kubectl get hpa
NAME            REFERENCE                          TARGETS              MINPODS   MAXPODS   REPLICAS   AGE
nodejs-app-hpa   Deployment/nodejs-app-deployment   cpu: <unknown>/50%   2         5         2          85m
vagrant@ubuntu2204:~/DevOps-class-assessment/Assessment9/nodejs-k8s-project$
```

7.2 Access the Application

Expose the Service:

kubectl expose deployment nodejs-app-deployment --type=NodePort --name=nodejs-app-service

```
vagrant@ubuntu2204:~/DevOps-class-assessment/Assessment9/nodejs-k8s-project$ kubectl expose deployment nodejs-app-deployment --type=NodePort --name=nodejs-app-service
service/nodejs-app-service exposed
vagrant@ubuntu2204:~/DevOps-class-assessment/Assessment9/nodejs-k8s-project$
```

Get the Minikube IP and Service Port:

minikube service nodejs-app-service --url

- **Access the Application** in your browser using the URL obtained from the previous command.

```
vagrant@ubuntu2204:~/DevOps-class-assessment/Assessment9/nodejs-k8s-project$ minikube service nodejs-app-service --url
http://192.168.49.2:31028
vagrant@ubuntu2204:~/DevOps-class-assessment/Assessment9/nodejs-k8s-project$
```

```
vagrant@ubuntu2204:~/DevOps-class-assessment/Assessment9/nodejs-k8s-project$ curl http://192.168.49.2:31028
Hello, World!vagrant@ubuntu2204:~/DevOps-class-assessment/Assessment9/nodejs-k8s-project$
```

8. Git Version Control

8.1 Create a New Branch for New Features

Create and switch to a new branch:

```
git checkout -b feature/new-feature
```

Make changes and commit:

```
# Make some changes
git add .
git commit -m "Add new feature"
```

Push the branch to the remote repository:

```
git push origin feature/new-feature
```

8.2 Rebase Feature Branch on Main Branch

Switch to the main branch and pull the latest changes:

```
git checkout main
git pull origin main
```

```
vagrant@ubuntu2204:~/DevOps-class-assessment/Assessment9/nodejs-k8s-project/kubernetes$ git checkout main
Switched to branch 'main'
vagrant@ubuntu2204:~/DevOps-class-assessment/Assessment9/nodejs-k8s-project/kubernetes$ git pull origin main
From https://github.com/darade-sudarshan/DevOps-class-assessment
 * branch            main       -> FETCH_HEAD
```

Rebase the feature branch:

```
git checkout feature/new-feature
git rebase main
```

```
vagrant@ubuntu2204:~/DevOps-class-assessment/Assessment9/nodejs-k8s-project/kubernetes$ git branch
  feature/new-feature
* main
vagrant@ubuntu2204:~/DevOps-class-assessment/Assessment9/nodejs-k8s-project/kubernetes$ git checkout feature/new-feature
Switched to branch 'feature/new-feature'
vagrant@ubuntu2204:~/DevOps-class-assessment/Assessment9/nodejs-k8s-project/kubernetes$ git rebase main
Current branch feature/new-feature is up to date.
vagrant@ubuntu2204:~/DevOps-class-assessment/Assessment9/nodejs-k8s-project/kubernetes$
```

Resolve conflicts if any, and continue the rebase:

```
git add .
git rebase --continue
```

```
vagrant@ubuntu2204:~/DevOps-class-assessment/Assessment9/nodejs-k8s-project/kubernetes$ git checkout feature/new-featur
Switched to branch 'feature/new-feature'
vagrant@ubuntu2204:~/DevOps-class-assessment/Assessment9/nodejs-k8s-project/kubernetes$ git rebase main
Current branch feature/new-feature is up to date.
```

Push the rebased feature branch:

```
git push origin feature/new-feature --force
```

```
vagrant@ubuntu2204:~/DevOps-class-assessment/Assessment9/nodejs-k8s-project/kubernetes$ git push origin feature/new-feature --force
Username for 'https://github.com': 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
Password for 'https://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@github.com':
Everything up-to-date
vagrant@ubuntu2204:~/DevOps-class-assessment/Assessment9/nodejs-k8s-project/kubernetes$
```

9. Final Commit and Cleanup

Merge feature branch to main:

```
git checkout main
git merge feature/new-feature
```

```
vagrant@ubuntu2204:~/DevOps-class-assessment/Assessment9/nodejs-k8s-project/kubernetes$ git config pull.rebase false
vagrant@ubuntu2204:~/DevOps-class-assessment/Assessment9/nodejs-k8s-project/kubernetes$ git pull origin main
From https://github.com/darade-sudarshan/DevOps-class-assessment
 * branch            main       -> FETCH_HEAD
fatal: refusing to merge unrelated histories
vagrant@ubuntu2204:~/DevOps-class-assessment/Assessment9/nodejs-k8s-project/kubernetes$
```

Push the changes to the main branch:

```
git push origin main
```

Clean up:

```
git branch -d feature/new-feature
git push origin --delete feature/new-feature
```

```
vagrant@ubuntu2204:~/DevOps-class-assessment/Assessment9/nodejs-k8s-project$ git  branch -d feature/new-feature
Deleted branch feature/new-feature (was a103581).
vagrant@ubuntu2204:~/DevOps-class-assessment/Assessment9/nodejs-k8s-project$
```

Project 02

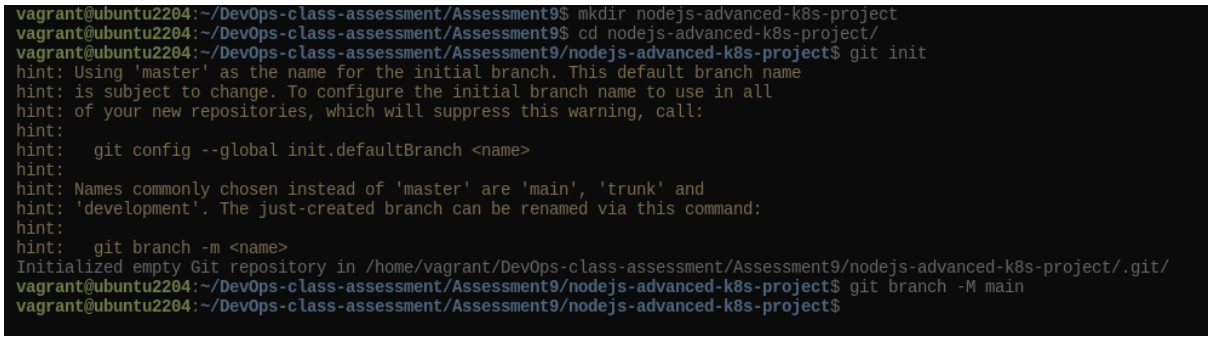
Deploy a Node.js application to Kubernetes with advanced usage of ConfigMaps and Secrets. Implement Horizontal Pod Autoscaler (HPA) with both scale-up and scale-down policies. The project will include a multi-environment configuration strategy, integrating a Redis cache, and monitoring application metrics.

Project Setup

1.1 Initialize a Git Repository

Create a new directory for your project and initialize Git:

```
mkdir nodejs-advanced-k8s-project
cd nodejs-advanced-k8s-project
git init
```



1.2 Create Initial Files

Create the initial Node.js application and Docker-related files:

```
npm init -y
npm install express redis body-parser
```

app.js

```
const express = require('express');
const bodyParser = require('body-parser');
const redis = require('redis');
const app = express();
const PORT = process.env.PORT || 3000;

// Connect to Redis
const redisClient = redis.createClient({
  url: `redis://${process.env.REDIS_HOST}:${process.env.REDIS_PORT}`
});
redisClient.on('error', (err) => console.error('Redis Client Error', err));

app.use(bodyParser.json());

app.get('/', async (req, res) => {
  const visits = await redisClient.get('visits');
  if (visits) {
```

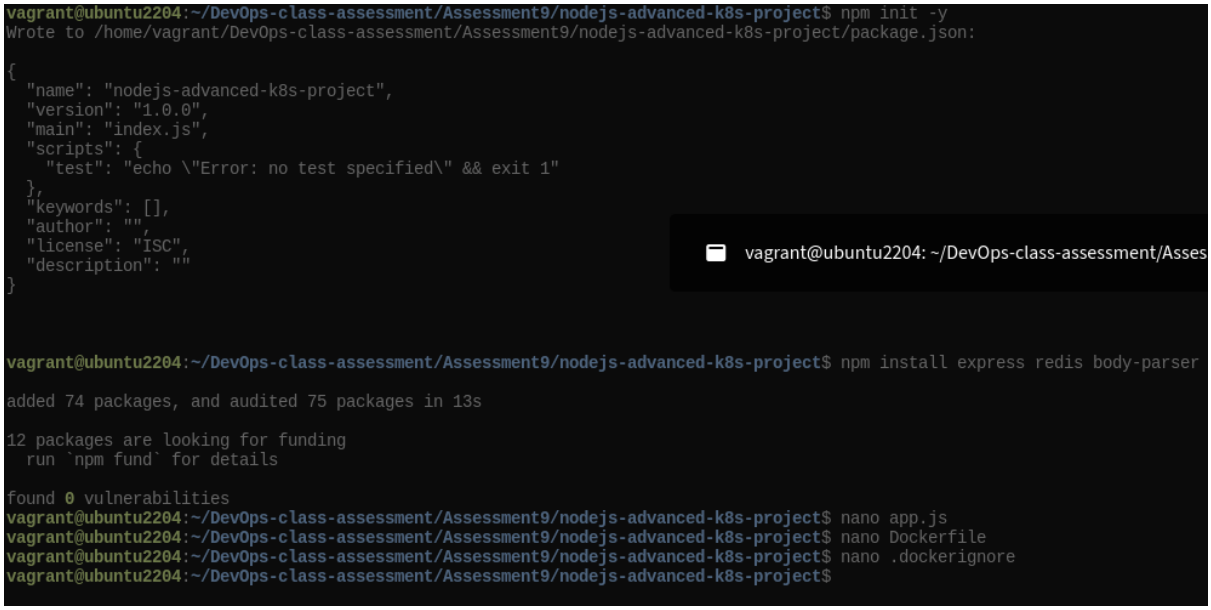
```
    await redisClient.set('visits', parseInt(visits) + 1);
  } else {
    await redisClient.set('visits', 1);
  }
  res.send(`Hello, World! You are visitor number ${visits || 1}`);
});

app.listen(PORT, () => {
  console.log(`Server is running on port ${PORT}`);
});
```

Dockerfile

```
FROM node:18

WORKDIR /usr/src/app

COPY package*.json ./

RUN npm install

COPY . .

EXPOSE 3000

CMD ["npm", "start"]
```

.dockerignore

```
node_modules
.npm
```



1. Build and push Docker image:

```
docker build -t your-dockerhub-username/nodejs-advanced-app:latest .
docker push your-dockerhub-username/nodejs-advanced-app:latest
```

Apply Kubernetes configurations:

```
kubectl apply -f kubernetes/
```

Access the application:

```
minikube service nodejs-advanced-app-service --url
```

2. Advanced Kubernetes Configuration

2.1 Deployment Configuration

Create `kubernetes/deployment.yaml` to deploy the Node.js application with Redis dependency:

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nodejs-advanced-app-deployment
spec:
  replicas: 2
  selector:
```

```
    matchLabels:
      app: nodejs-advanced-app
  template:
    metadata:
      labels:
        app: nodejs-advanced-app
    spec:
      containers:
      - name: nodejs-advanced-app
        image: your-dockerhub-username/nodejs-advanced-app:latest
        ports:
        - containerPort: 3000
        env:
        - name: PORT
          valueFrom:
            configMapKeyRef:
              name: app-config
              key: PORT
        - name: REDIS_HOST
          valueFrom:
            configMapKeyRef:
              name: redis-config
              key: REDIS_HOST
        - name: REDIS_PORT
          valueFrom:
            configMapKeyRef:
              name: redis-config
              key: REDIS_PORT
        - name: NODE_ENV
          valueFrom:
            secretKeyRef:
              name: app-secrets
              key: NODE_ENV
      - name: redis
        image: redis:latest
        ports:
        - containerPort: 6379
```

2.2 ConfigMap for Application and Redis

Create kubernetes/configmap.yaml to manage application and Redis configurations:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: app-config
```

```
data:
  PORT: "3000"

---
apiVersion: v1
kind: ConfigMap
metadata:
  name: redis-config
data:
  REDIS_HOST: "redis"
  REDIS_PORT: "6379"
```

2.3 Secret for Sensitive Data

Create kubernetes/secret.yaml to manage sensitive environment variables:

```
apiVersion: v1
kind: Secret
metadata:
  name: app-secrets
type: Opaque
data:
  NODE_ENV: cHJvZHVjdGlvbg== # Base64 encoded value for "production"
```

2.4 Service Configuration

Create kubernetes/service.yaml to expose the Node.js application:

```
apiVersion: v1
kind: Service
metadata:
  name: nodejs-advanced-app-service
spec:
  selector:
    app: nodejs-advanced-app
  ports:
  - protocol: TCP
    port: 80
    targetPort: 3000
  type: LoadBalancer
```

2.5 Horizontal Pod Autoscaler with Scale-Up and Scale-Down Policies

Create kubernetes/hpa.yaml to manage autoscaling:

```
apiVersion: autoscaling/v2beta2
kind: HorizontalPodAutoscaler
metadata:
  name: nodejs-advanced-app-hpa
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: nodejs-advanced-app-deployment
  minReplicas: 2
  maxReplicas: 5
  metrics:
  - type: Resource
    resource:
      name: cpu
      target:
        type: Utilization
        averageUtilization: 50
  - type: Resource
    resource:
      name: memory
      target:
        type: Utilization
        averageUtilization: 70
  behavior:
    scaleUp:
      stabilizationWindowSeconds: 30
      selectPolicy: Max
      policies:
      - type: Pods
        value: 2
        periodSeconds: 30
      - type: Resource
        resource: cpu
        value: 2
        periodSeconds: 30
    scaleDown:
      stabilizationWindowSeconds: 30
      selectPolicy: Min
      policies:
      - type: Pods
        value: 1
        periodSeconds: 30
      - type: Resource
```

```
      resource: memory
      value: 1
      periodSeconds: 30
```

2.6 Vertical Pod Autoscaler Configuration

Create kubernetes/vpa.yaml to manage vertical scaling:

```
apiVersion: autoscaling.k8s.io/v1beta2
kind: VerticalPodAutoscaler
metadata:
  name: nodejs-advanced-app-vpa
spec:
  targetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: nodejs-advanced-app-deployment
  updatePolicy:
    updateMode: "Auto"
```

2.7 Redis Deployment

Add a Redis deployment configuration to kubernetes/redis-deployment.yaml:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: redis-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      app: redis
  template:
    metadata:
      labels:
        app: redis
    spec:
      containers:
      - name: redis
        image: redis:latest
        ports:
        - containerPort: 6379
```

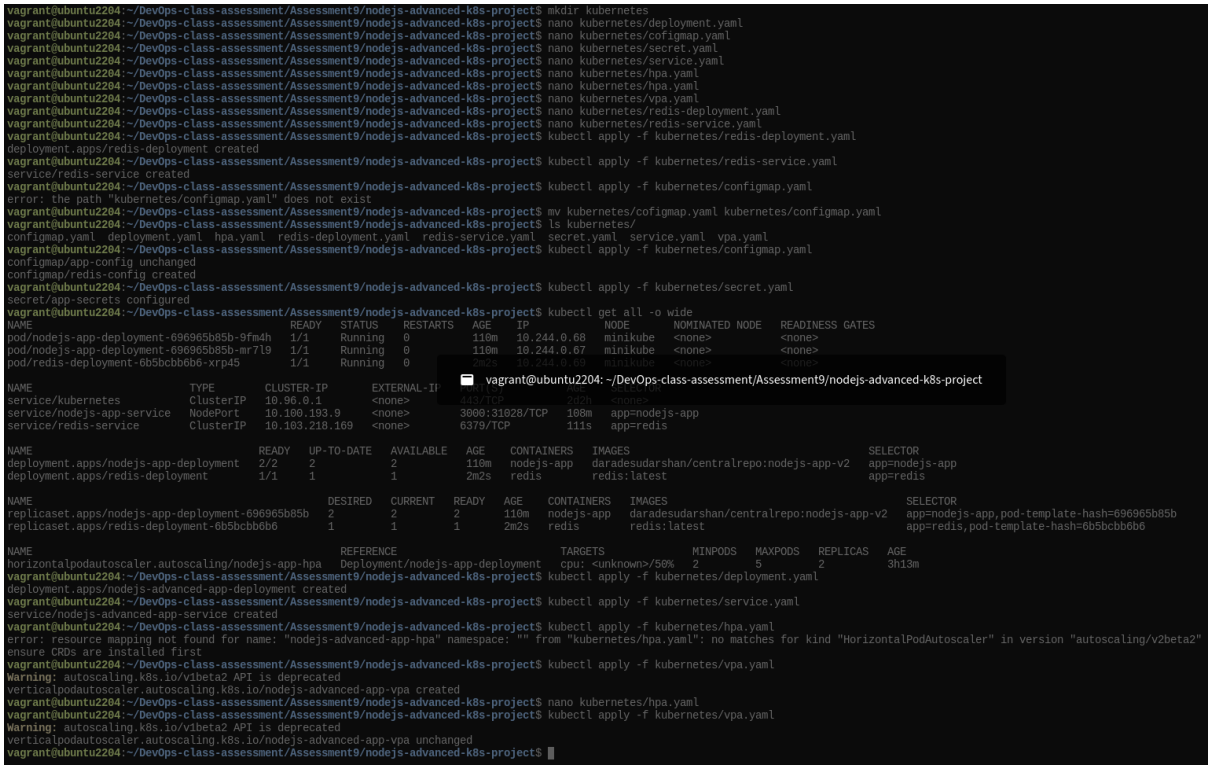
Add Redis service configuration to kubernetes/redis-service.yaml:

```
apiVersion: v1
kind: Service
metadata:
  name: redis-service
spec:
  selector:
    app: redis
  ports:
  - protocol: TCP
    port: 6379
    targetPort: 6379
  type: ClusterIP
```

2.8 Apply Kubernetes Configurations

Apply all configurations to your Minikube cluster:

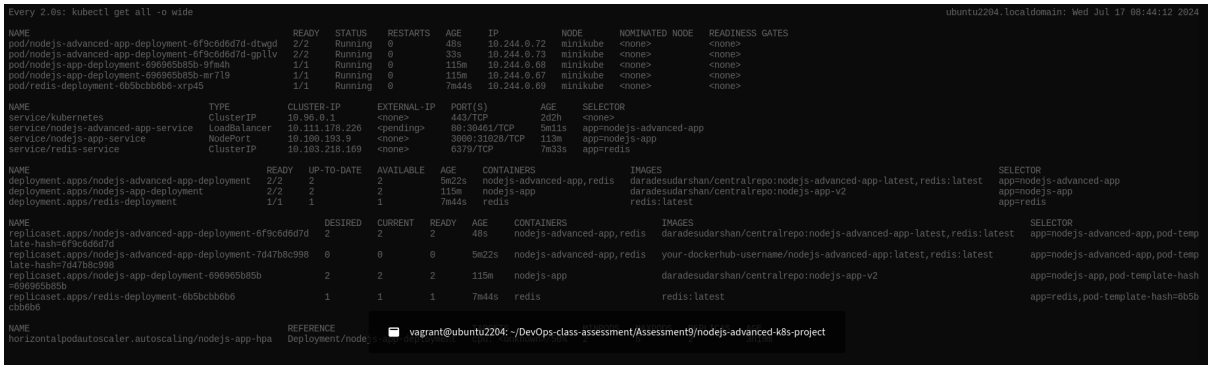
```
kubectl apply -f kubernetes/redis-deployment.yaml
kubectl apply -f kubernetes/redis-service.yaml
kubectl apply -f kubernetes/configmap.yaml
kubectl apply -f kubernetes/secret.yaml
kubectl apply -f kubernetes/deployment.yaml
kubectl apply -f kubernetes/service.yaml
kubectl apply -f kubernetes/hpa.yaml
kubectl apply -f kubernetes/vpa.yaml
```



2.9 Verify Deployments and Services

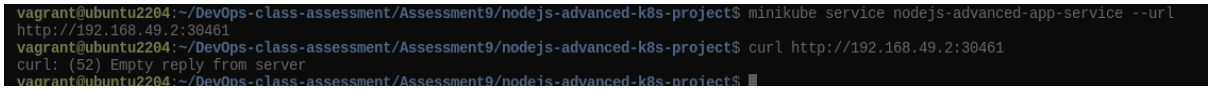
Check the status of your deployments and services:

kubectl get all



Access the application via Minikube:

minikube service nodejs-advanced-app-service --url



2.10 Testing Scaling

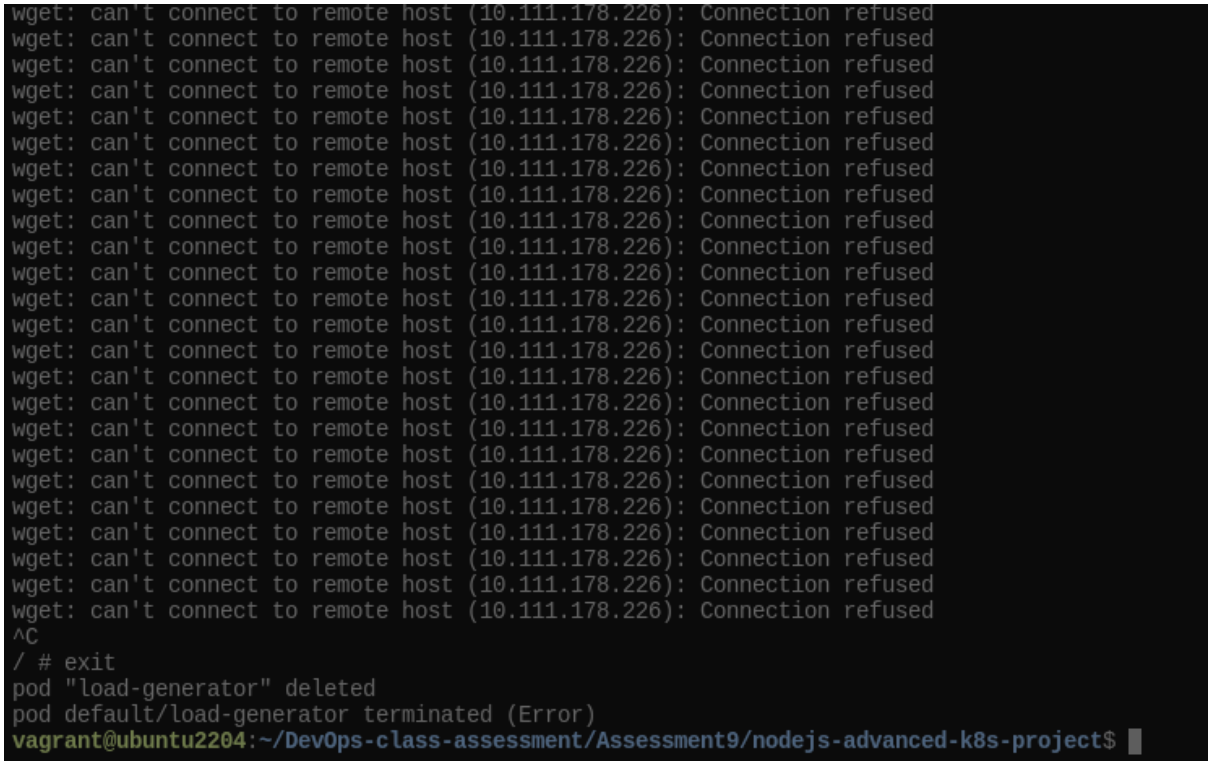
Simulate load on the application to test the HPA:

```
kubectl run -i --tty --rm load-generator --image=busybox --restart=Never --
/bin/sh
# Inside the pod, run the following command to generate load
while true; do wget -q -O- http://nodejs-advanced-app-service; done
```

2.11 Validate Autoscaling Behavior

Observe the HPA behavior:

```
kubectl get hpa
```



Watch the scaling events and verify that the application scales up and down based on the policies you configured.

3. Project Wrap-Up

3.1 Review and Clean Up

After completing the project, review the configurations and clean up the Minikube environment if needed:

minikube delete