Project details are available from page 13

1. ConfigMap

ConfigMap is a Kubernetes object that lets you store configuration data in key-value pairs. It is used to manage non-sensitive configuration information separately from the application code.

Creating a ConfigMap

You can create a ConfigMap from a literal value or from a file. Here's an example of creating a ConfigMap from literal values:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: web-config
data:
  DATABASE_URL: "jdbc:mysql://db-server:3306/mydatabase"
  APP_ENV: "production"
```

Using ConfigMap in a Pod

To use the ConfigMap in a Pod, you need to reference it in your Pod specification. Here's how you can inject ConfigMap values as environment variables:

```
apiVersion: v1
kind: Pod
metadata:
  name: web-app
spec:
  containers:
  - name: web-container
    image: my-web-app:latest
    env:
```

```
    - name: DATABASE_URL
      valueFrom:
        configMapKeyRef:
          name: web-config
          key: DATABASE_URL
    - name: APP_ENV
      valueFrom:
        configMapKeyRef:
          name: web-config
          key: APP_ENV
```

Example Use Case: Mounting ConfigMap as a File

Sometimes, an application may require configuration files. You can mount a ConfigMap as a file inside a container.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: config-files
data:
  config.yaml: |
    database:
      url: "jdbc:mysql://db-server:3306/mydatabase"
    environment: "production"
```

Mount the ConfigMap as a volume in the Pod:

```
apiVersion: v1
kind: Pod
```

```
metadata:
  name: web-app
spec:
  containers:
  - name: web-container
    image: my-web-app:latest
    volumeMounts:
    - name: config-volume
      mountPath: /etc/config
  volumes:
  - name: config-volume
    configMap:
      name: config-files
```

The configuration file `config.yaml` will be available at `/etc/config/config.yaml` inside the container.

2. Secrets

Secrets is a Kubernetes object designed to hold sensitive data such as passwords, OAuth tokens, and SSH keys. Secrets ensure that sensitive information is stored securely.

Creating a Secret

You can create a Secret from literal values or from files. Here's an example of creating a Secret from literal values:

```
apiVersion: v1
kind: Secret
metadata:
  name: db-credentials
type: Opaque
data:
  username: dXNlcm5hbWU=   # base64 encoded 'username'
  password: cGFzc3dvcmQ=   # base64 encoded 'password'
```

Using Secrets in a Pod

To use the Secret in a Pod, reference it in your Pod specification and inject it as environment variables:

```
apiVersion: v1
kind: Pod
metadata:
  name: web-app
spec:
  containers:
  - name: web-container
    image: my-web-app:latest
    env:
    - name: DB_USERNAME
      valueFrom:
        secretKeyRef:
          name: db-credentials
          key: username
    - name: DB_PASSWORD
      valueFrom:
        secretKeyRef:
          name: db-credentials
          key: password
```

Example Use Case: Mounting Secrets as Files

For applications that require secrets as files, you can mount the Secret as a volume inside a container.

```
apiVersion: v1
```

```
kind: Secret
metadata:
  name: ssh-keys
type: Opaque
data:
  ssh-privatekey: <base64-encoded-private-key>
  ssh-publickey: <base64-encoded-public-key>
```

Mount the Secret as a volume in the Pod:

```
apiVersion: v1
kind: Pod
metadata:
  name: web-app
spec:
  containers:
  - name: web-container
    image: my-web-app:latest
    volumeMounts:
    - name: ssh-volume
      mountPath: /etc/ssh
      readOnly: true
  volumes:
  - name: ssh-volume
    secret:
      secretName: ssh-keys
```

The SSH keys will be available at `/etc/ssh` inside the container.

3. Environment Variables

Environment variables are a way to pass configuration settings to applications running inside containers. They can be defined directly in the Pod specification or sourced from ConfigMaps and Secrets.

Example Use Case: Passing Configuration to a Container

Environment variables can be used to pass various configurations like application mode, API endpoints, and feature flags to the container.

Defining Environment Variables in Pod Specification

```
apiVersion: v1
kind: Pod
metadata:
  name: web-app
spec:
  containers:
  - name: web-container
    image: my-web-app:latest
    env:
    - name: APP_MODE
      value: "production"
    - name: API_ENDPOINT
      value: "https://api.example.com"
```

Example Use Case: Using Environment Variables from ConfigMaps and Secrets

Combining ConfigMaps and Secrets with environment variables provides a flexible and secure way to manage configurations.

Using ConfigMap and Secret Environment Variables Together

```
apiVersion: v1
kind: Pod
metadata:
  name: web-app
spec:
```

```
containers:
- name: web-container
  image: my-web-app:latest
  env:
  - name: DATABASE_URL
    valueFrom:
      configMapKeyRef:
        name: web-config
        key: DATABASE_URL
  - name: APP_ENV
    valueFrom:
      configMapKeyRef:
        name: web-config
        key: APP_ENV
  - name: DB_USERNAME
    valueFrom:
      secretKeyRef:
        name: db-credentials
        key: username
  - name: DB_PASSWORD
    valueFrom:
      secretKeyRef:
        name: db-credentials
        key: password
```

Autoscaling in Kubernetes

1. Horizontal Pod Autoscaler (HPA)

1.1. Define a Deployment

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: web-app
spec:
  replicas: 2
  selector:
    matchLabels:
      app: web-app
  template:
    metadata:
      labels:
        app: web-app
    spec:
      containers:
      - name: web-container
        image: my-web-app:latest
        ports:
        - containerPort: 80
        resources:
          requests:
            cpu: "500m"
          limits:
            cpu: "1"
```

1.2. Apply the Deployment

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

1.3. Create a Service

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

1.4. Apply the Service

```
kubectl apply -f service.yaml
```

1.5. Create an HPA

Define an HPA to scale the number of pods based on CPU utilization:

```
apiVersion: autoscaling/v2beta2
kind: HorizontalPodAutoscaler
metadata:
  name: web-app-hpa
spec:
```

```
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: web-app
  minReplicas: 2
  maxReplicas: 10
  metrics:
  - type: Resource
    resource:
      name: cpu
      target:
        type: Utilization
        averageUtilization: 50
```

1.6. Apply the HPA

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

Vertical Pod Autoscaler (VPA)

2.1. Define a Deployment

Create a Deployment for the batch job:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: batch-job
spec:
  replicas: 1
```

```
  selector:
    matchLabels:
      app: batch-job
  template:
    metadata:
      labels:
        app: batch-job
    spec:
      containers:
      - name: batch-container
        image: my-batch-job:latest
        resources:
          requests:
            cpu: "500m"
            memory: "1Gi"
          limits:
            cpu: "1"
            memory: "2Gi"
```

2.2. Apply the Deployment

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

2.3. Create a VPA

Define a VPA to manage the resource requests and limits for the Pod:

```
apiVersion: verticalpodautoscaler.k8s.io/v1
kind: VerticalPodAutoscaler
metadata:
  name: batch-job-vpa
spec:
```

```
  targetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: batch-job
  updatePolicy:
    updateMode: Auto
```

2.4. Apply the VPA

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

Linux Scripts

Viewing Processes (ps, top)

Use Cases:

1. **System Monitoring**:
    - **Example**: An administrator needs to check the status of all running processes to ensure that critical applications are running smoothly.

Commands:

```
ps aux  # Displays detailed information about all running processes

top      # Interactive view of system processes, updates in real time
```

2. **Troubleshooting Performance Issues**:
    - **Example**: A developer notices that the server is slow and needs to find out which processes are consuming the most CPU or memory.

Commands:

```
top  # Look for processes consuming high CPU or memory

ps -eo pid,comm,%cpu,%mem --sort=-%cpu | head  # Display top 10 processes by CPU usage
```

3. **Identifying Zombie Processes**:
    - **Example**: The system administrator is dealing with processes that are stuck in the “zombie” state.

Commands:

```
ps aux | grep 'Z'  # Finds processes in a zombie state
```

Examples:

Example 1:

```
ps aux | grep nginx
```

- Finds processes related to the `nginx` web server.

Example 2:

```
top -u username
```

- Displays processes owned by a specific user.

Managing Processes (kill, nice)

Use Cases:

1. **Stopping Unresponsive Applications**:
    - **Example**: A user needs to stop a process that has become unresponsive or is consuming excessive resources.

Commands:

```
kill -9 12345  # Forcefully terminates the process with PID 12345
```

    -

2. **Adjusting Process Priority**:
    - **Example**: A system administrator wants to lower the priority of a process to ensure it does not hog resources.

Commands:

```
nice -n 10 command  # Start a process with a lower priority

renice +10 -p 12345  # Change the priority of an existing process with PID 12345
```

    -

3. **Gracefully Stopping Services**:
    - **Example**: An admin needs to restart a service to apply configuration changes.

Commands:

```
kill -HUP 12345  # Sends a SIGHUP signal to the process to reload configuration
```

    -

Examples:

Example 1:

```
killall -9 firefox
```

- Kills all processes named `firefox`.

Example 2:

```
nice -n -10 ./heavy_script.sh
```

- Runs `heavy_script.sh` with a higher priority.

Configure SSH

Shell Scripts

Writing Basic Shell Scripts

Use Cases:

1. **Automating Routine Tasks**:
    - **Example**: A sysadmin wants to automate the backup of log files.

Commands:

```
#!/bin/bash
```

```
cp /var/log/syslog /backup/syslog-$(date +%F).log
```

-

2. **System Maintenance**:
    - **Example**: A developer creates a script to clean up temporary files.

Commands:

```
#!/bin/bash
rm -rf /tmp/*
```

3. **Batch Processing**:
    - **Example**: A data analyst needs to process multiple files in a directory.

Commands:

```
#!/bin/bash

process_file() {
  local file="$1"
  echo "Processing $file"
  # Add more commands to process the file here
}

for file in /data/*.csv; do
  process_file "$file"
done
```

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.

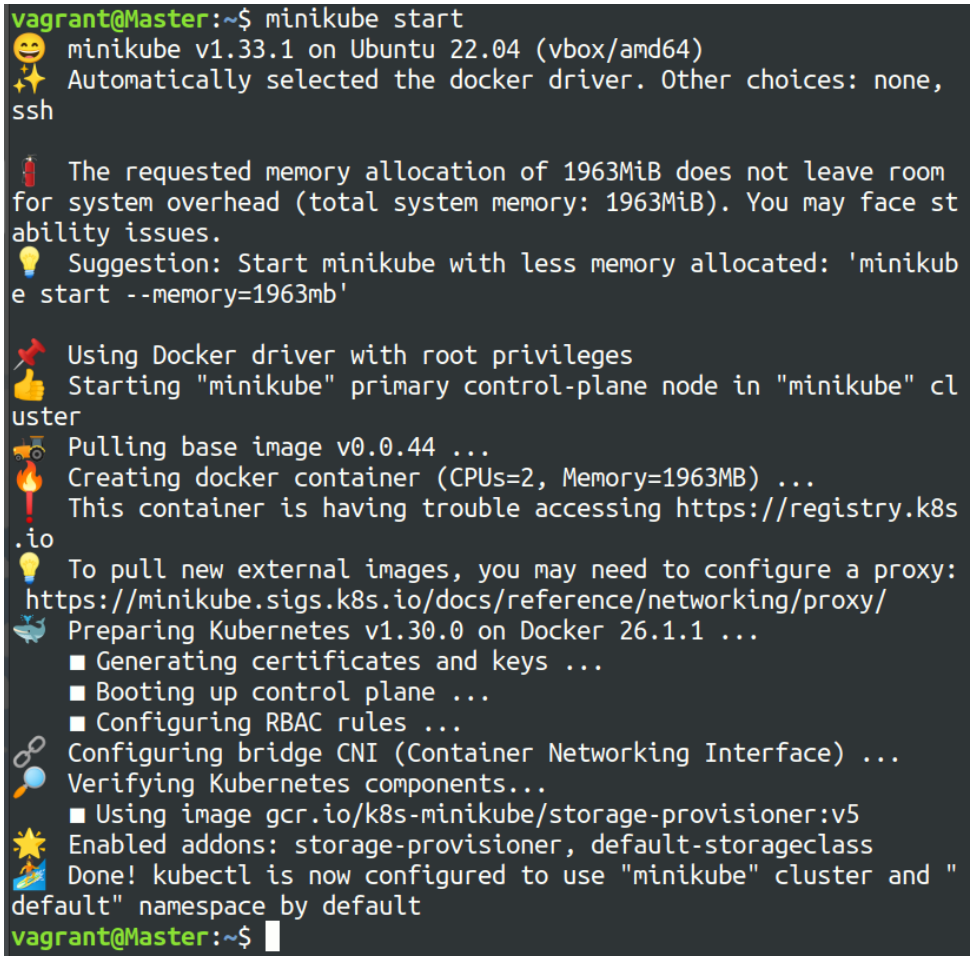
Project 01

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"
```

2. Develop a Node.js Application

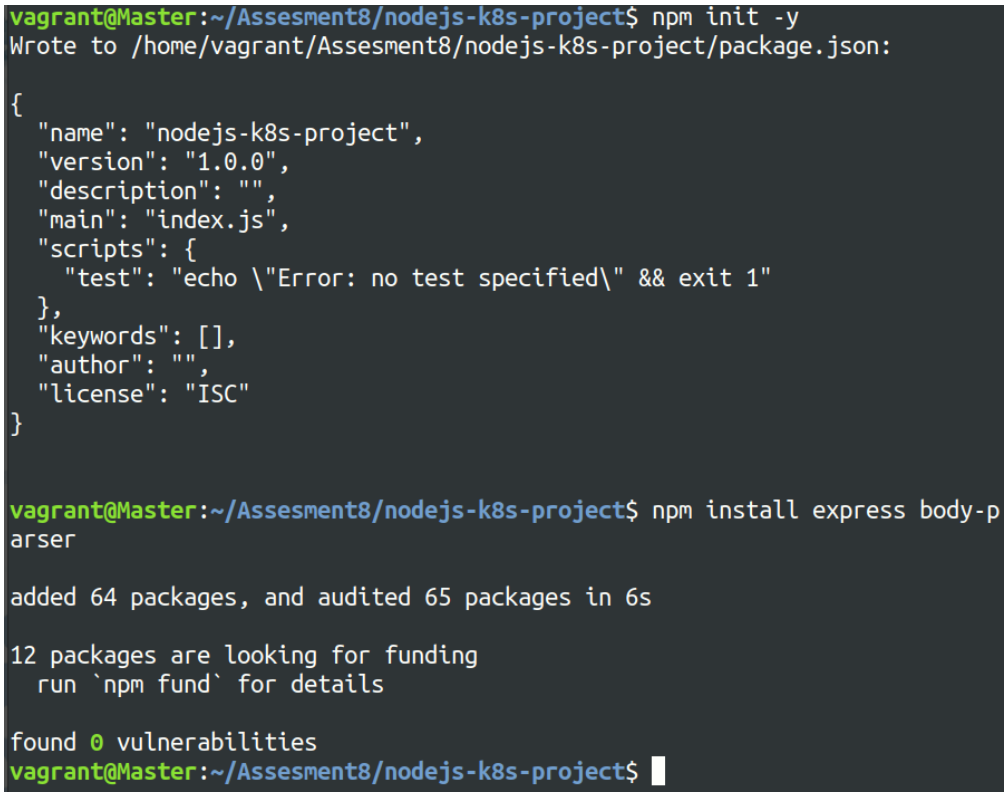
2.1 Create the Node.js App

Initialize the Node.js project:

```
npm init -y
```

Install necessary packages:

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



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"
}
```

```
vagrant@Master:~/Assesment8/nodejs-k8s-project$ vim app.js
vagrant@Master:~/Assesment8/nodejs-k8s-project$ vim package.json
vagrant@Master:~/Assesment8/nodejs-k8s-project$ cat package.json
{
  "name": "nodejs-k8s-project",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "start": "node app.js"
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "body-parser": "^1.20.2",
    "express": "^4.19.2"
  }
}
vagrant@Master:~/Assesment8/nodejs-k8s-project$
```

2.2 Commit the Node.js Application

Add and commit changes:

```
git add .

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

```
vagrant@Master:~/Assesment8/nodejs-k8s-project$ git add .
vagrant@Master:~/Assesment8/nodejs-k8s-project$ git commit -m "Add Node.js
 application code"
[master 4931d5d] Add Node.js application code
 3 files changed, 1219 insertions(+)
 create mode 100644 app.js
 create mode 100644 package-lock.json
 create mode 100644 package.json
vagrant@Master:~/Assesment8/nodejs-k8s-project$
```

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)

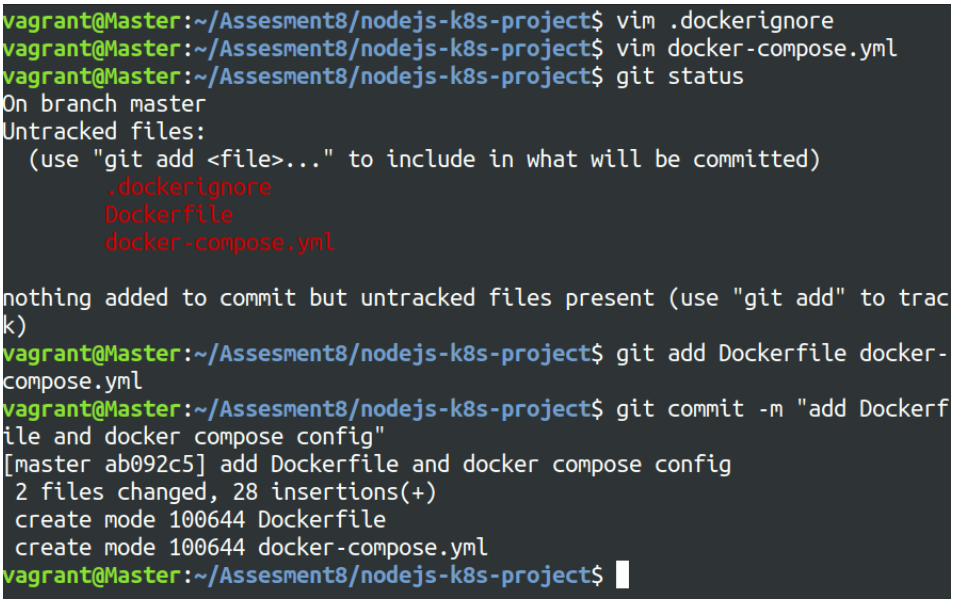
Add `docker-compose.yml`:

```
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@Master:~/Assesment8/nodejs-k8s-project$ docker build -t nodejs-app:latest .
[+] Building 68.6s (11/11) FINISHED                                    docker:default
 => [internal] load build definition from Dockerfile                           0.0s
 => => transferring dockerfile: 397B                                           0.0s
 => [internal] load metadata for docker.io/library/node:18                     2.3s
 => [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:aabbaf118c7c0a6e9a3bda69bd2a94b0  0.0s
 => [internal] load build context                                              0.0s
 => => transferring context: 3.35kB                                            0.0s
 => CACHED [2/5] WORKDIR /usr/src/app                                          0.0s
 => [3/5] COPY package*.json ./                                                0.0s
 => [4/5] RUN npm install                                                     65.9s
 => [5/5] COPY . .                                                             0.1s
 => exporting to image                                                         0.2s
 => => exporting layers                                                        0.2s
 => => writing image sha256:0e6c6d7cf96d8e4de4634b0bdca31f29959e59f5cc284d489f91  0.0s
 => => naming to docker.io/library/nodejs-app:latest                           0.0s
vagrant@Master:~/Assesment8/nodejs-k8s-project$
```

4.2 Push Docker Image to Docker Hub

Tag and push the image:

```
docker tag nodejs-app:latest chirag1212/nodejs-app:latest
```

```
docker push chirag1212/nodejs-app:latest
```

```
vagrant@Master:~/Assesment8/nodejs-k8s-project$ docker tag nodejs-app:latest chirag1212
/nodejs-app:latest
vagrant@Master:~/Assesment8/nodejs-k8s-project$ docker images
REPOSITORY                    TAG       IMAGE ID       CREATED         SIZE
chirag1212/nodejs-app         latest    0e6c6d7cf96d   4 minutes ago   1.09GB
nodejs-app                    latest    0e6c6d7cf96d   4 minutes ago   1.09GB
flask-k8s-app                 v2        16712ab1f865   19 hours ago    156MB
flask-k8s-app                 latest    ba3dfc19cefd   20 hours ago    156MB
nodejs-k8s-app                v2        139ef52cb13a   23 hours ago    919MB
nodejs-k8s-app                latest    b45a37e694fb   24 hours ago    919MB
chirag1212/my_repo            latest    5eef1184c621   46 hours ago    1.11GB
wordpress                     latest    d2a2d7e671fd   3 weeks ago     685MB
nginx                         latest    fffffc90d343   3 weeks ago     188MB
postgres                      latest    f23dc7cd74bd   2 months ago    432MB
gcr.io/k8s-minikube/kicbase   v0.0.44   5a6e59a9bdc0   2 months ago    1.26GB
mysql                         5.7       5107333e08a8   7 months ago    501MB
vagrant@Master:~/Assesment8/nodejs-k8s-project$ docker push chirag1212/nodejs-app:lates
t
The push refers to repository [docker.io/chirag1212/nodejs-app]
a3be8da658d9: Pushed
3812a0a39945: Pushed
c87422ff7bce: Pushed
058025a2e5c7: Pushed
0970e1a837f7: Mounted from library/node
d4061df7c236: Mounted from library/node
9487e6e19e60: Mounted from library/node
6ef00066aa6f: Mounted from library/node
b11bb163e263: Mounted from chirag1212/my_repo
b779a72428fa: Mounted from chirag1212/my_repo
8ada682d3780: Mounted from chirag1212/my_repo
15bb10f9bb3a: Waiting
```

Add and commit changes:

```
git add .
```

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

```
vagrant@Master:~/Assesment8/nodejs-k8s-project$ git add .
vagrant@Master:~/Assesment8/nodejs-k8s-project$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   .dockerignore
        modified:   package.json

vagrant@Master:~/Assesment8/nodejs-k8s-project$ git commit -m "Build and push Docker im
age"
[master 05835b9] Build and push Docker image
 2 files changed, 3 insertions(+), 1 deletion(-)
 create mode 100644 .dockerignore
vagrant@Master:~/Assesment8/nodejs-k8s-project$
```

5. Create Kubernetes Configurations

5.1 Create Kubernetes Deployment

Create `kubernetes/deployment.yaml`:

```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: chirag1212/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@Master:~/Assesment8/nodejs-k8s-project/kubernetes$ vim deployment.yaml
vagrant@Master:~/Assesment8/nodejs-k8s-project/kubernetes$ vim configmap.yaml
vagrant@Master:~/Assesment8/nodejs-k8s-project/kubernetes$ vim secret.yaml
vagrant@Master:~/Assesment8/nodejs-k8s-project/kubernetes$ cd ..
vagrant@Master:~/Assesment8/nodejs-k8s-project$ git add kubernetes/
vagrant@Master:~/Assesment8/nodejs-k8s-project$ git commit -m "Add kubernetes deploymen
t, configmap, and secret"
[master 5f35b1d] Add kubernetes deployment, configmap, and secret
 3 files changed, 46 insertions(+)
 create mode 100644 kubernetes/configmap.yaml
 create mode 100644 kubernetes/deployment.yaml
 create mode 100644 kubernetes/secret.yaml
vagrant@Master:~/Assesment8/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@Master:~/Assesment8/nodejs-k8s-project$ kubectl get all
NAME                 TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
service/kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP   63m
vagrant@Master:~/Assesment8/nodejs-k8s-project$ kubectl apply -f kubernetes/deployment.
yaml
deployment.apps/nodejs-app-deployment created
vagrant@Master:~/Assesment8/nodejs-k8s-project$ kubectl get all
NAME                                      READY   STATUS              RESTARTS   AGE
pod/nodejs-app-deployment-bd676bb7c-9bcp2   0/1     ContainerCreating   0          12s
pod/nodejs-app-deployment-bd676bb7c-zms2w   0/1     ContainerCreating   0          12s

NAME                 TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
service/kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP   64m

NAME                                   READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/nodejs-app-deployment   0/2     2            0           12s

NAME                                              DESIRED   CURRENT   READY   AGE
replicaset.apps/nodejs-app-deployment-bd676bb7c   2         2         0       12s
```

6. Implement Autoscaling

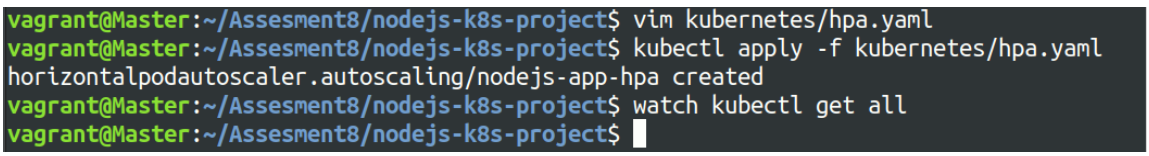
6.1 Create Horizontal Pod Autoscaler

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
```



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@Master:~/Assesment8/nodejs-k8s-project$ vim kubernetes/vpa.yaml
vagrant@Master:~/Assesment8/nodejs-k8s-project$ kubectl apply -f kubernetes/vpa.yaml
verticalpodautoscaler.autoscaling.k8s.io/nodejs-app-vpa created
```

7. Test the Deployment

7.1 Check the Status of Pods, Services, and HPA

Verify the Pods:

```
kubectl get pods
```

```
vagrant@Master:~/Assesment8/nodejs-k8s-project$ kubectl get pods
NAME                                    READY   STATUS    RESTARTS   AGE
nodejs-app-deployment-bd676bb7c-9bcp2   1/1     Running   0          63m
nodejs-app-deployment-bd676bb7c-zms2w   1/1     Running   0          63m
vagrant@Master:~/Assesment8/nodejs-k8s-project$
```

Verify the Services:

```
kubectl get svc
```

```
vagrant@Master:~/Assesment8/nodejs-k8s-project$ kubectl get svc
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP   127m
vagrant@Master:~/Assesment8/nodejs-k8s-project$
```

Verify the HPA:

```
kubectl get hpa
```

```
vagrant@Master:~/Assesment8/nodejs-k8s-project$ kubectl get hpa
NAME              REFERENCE                          TARGETS              MINPODS   MAXP
ODS   REPLICAS   AGE
nodejs-app-hpa    Deployment/nodejs-app-deployment   cpu: <unknown>/50%   2         5
      2          3m38s
vagrant@Master:~/Assesment8/nodejs-k8s-project$
```

7.2 Access the Application

Expose the Service:

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

```
vagrant@Master:~/Assesment8/nodejs-k8s-project$ kubectl expose deployment nodejs-app-de
ployment --type=NodePort --name=nodejs-app-service
service/nodejs-app-service exposed
vagrant@Master:~/Assesment8/nodejs-k8s-project$
```

Get the Minikube IP and Service Port:

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

```
vagrant@Master:~/Assesment8/nodejs-k8s-project$ minikube service nodejs-app-service --u
rl
http://192.168.49.2:30526
vagrant@Master:~/Assesment8/nodejs-k8s-project$
```

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

```
vagrant@Master:~/Assesment8/nodejs-k8s-project$ minikube service nodejs-app-service --u
rl
http://192.168.49.2:30526
vagrant@Master:~/Assesment8/nodejs-k8s-project$ curl http://192.168.49.2:30526
Hello, World!vagrant@Master:~/Assesment8/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"
```

```
vagrant@Master:~/Assesment8/nodejs-k8s-project$ git checkout -b feature/new-feature
Switched to a new branch 'feature/new-feature'
vagrant@Master:~/Assesment8/nodejs-k8s-project$ git add .
vagrant@Master:~/Assesment8/nodejs-k8s-project$ git commit -m "Add new feature"
[feature/new-feature f7f7454] Add new feature
 2 files changed, 31 insertions(+)
 create mode 100644 kubernetes/hpa.yaml
 create mode 100644 kubernetes/vpa.yaml
vagrant@Master:~/Assesment8/nodejs-k8s-project$
```

Push the branch to the remote repository:

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

```
vagrant@Master:~/Assesment8/nodejs-k8s-project$ git push origin feature/new-feature
Username for 'https://github.com': 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
Password for 'https://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@github.com':
Enumerating objects: 27, done.
Counting objects: 100% (27/27), done.
Delta compression using up to 2 threads
Compressing objects: 100% (24/24), done.
Writing objects: 100% (27/27), 10.90 KiB | 587.00 KiB/s, done.
Total 27 (delta 6), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (6/6), done.
remote:
remote: Create a pull request for 'feature/new-feature' on GitHub by visiting:
remote:      https://github.com/TankChirag-1212/knock_task1/pull/new/feature/new-featur
e
remote:
To https://github.com/TankChirag-1212/knock_task1.git
 * [new branch]      feature/new-feature -> feature/new-feature
vagrant@Master:~/Assesment8/nodejs-k8s-project$
```

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@Master:~/Assesment8/nodejs-k8s-project$ git checkout master
Already on 'master'
vagrant@Master:~/Assesment8/nodejs-k8s-project$ git pull origin master
From https://github.com/TankChirag-1212/knock_task1
 * branch            master     -> FETCH_HEAD
hint: You have divergent branches and need to specify how to reconcile them.
hint: You can do so by running one of the following commands sometime before
hint: your next pull:
hint:
hint:   git config pull.rebase false  # merge (the default strategy)
hint:   git config pull.rebase true   # rebase
hint:   git config pull.ff only       # fast-forward only
hint:
hint: You can replace "git config" with "git config --global" to set a default
hint: preference for all repositories. You can also pass --rebase, --no-rebase,
hint: or --ff-only on the command line to override the configured default per
hint: invocation.
fatal: Need to specify how to reconcile divergent branches.
vagrant@Master:~/Assesment8/nodejs-k8s-project$
```

Rebase the feature branch:

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

```
git rebase main
```

```
vagrant@Master:~/Assesment8/nodejs-k8s-project$ git checkout feature/new-feature
Switched to branch 'feature/new-feature'
vagrant@Master:~/Assesment8/nodejs-k8s-project$ git rebase master
Current branch feature/new-feature is up to date.
vagrant@Master:~/Assesment8/nodejs-k8s-project$
```

Resolve conflicts if any, and continue the rebase:

```
git add .
```

```
git rebase --continue
```

Push the rebased feature branch:

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

```
vagrant@Master:~/Assesment8/nodejs-k8s-project$ git push origin feature/new-feature
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'feature/new-feature' on GitHub by visiting:
remote:      https://github.com/TankChirag-1212/Assessment_8/pull/new/feature/new-featu
re
remote:
To https://github.com/TankChirag-1212/Assessment_8.git
 * [new branch]      feature/new-feature -> feature/new-feature
vagrant@Master:~/Assesment8/nodejs-k8s-project$
```

9. Final Commit and Cleanup

Merge feature branch to main:

```
git checkout main
```

```
git merge feature/new-feature
```

```
vagrant@Master:~/Assesment8/nodejs-k8s-project$ git checkout master
Switched to branch 'master'
vagrant@Master:~/Assesment8/nodejs-k8s-project$ git merge feature/new-feature
Updating 5f35b1d..f7f7454
Fast-forward
 kubernetes/hpa.yaml | 19 ++++++++++++++++++
 kubernetes/vpa.yaml | 12 ++++++++++++
 2 files changed, 31 insertions(+)
 create mode 100644 kubernetes/hpa.yaml
 create mode 100644 kubernetes/vpa.yaml
vagrant@Master:~/Assesment8/nodejs-k8s-project$
```

Push the changes to the main branch:

```
git push origin main
```

```
vagrant@Master:~/Assesment8/nodejs-k8s-project$ git push -u origin master
Enumerating objects: 27, done.
Counting objects: 100% (27/27), done.
Delta compression using up to 2 threads
Compressing objects: 100% (24/24), done.
Writing objects: 100% (27/27), 10.90 KiB | 413.00 KiB/s, done.
Total 27 (delta 6), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (6/6), done.
To https://github.com/TankChirag-1212/Assessment_8.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
vagrant@Master:~/Assesment8/nodejs-k8s-project$ git
```

Clean up:

```
git branch -d feature/new-feature

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

```
vagrant@Master:~/Assesment8/nodejs-k8s-project$ git branch -d feature/new-feature
Deleted branch feature/new-feature (was f7f7454).
vagrant@Master:~/Assesment8/nodejs-k8s-project$ git push origin --delete feature/new-fe
ature
To https://github.com/TankChirag-1212/Assessment_8.git
 - [deleted]         feature/new-feature
vagrant@Master:~/Assesment8/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
```

```
vagrant@Master:~/Assesment8/nodejs-k8s-project$ mkdir nodejs-advanced-k8s-project
vagrant@Master:~/Assesment8/nodejs-k8s-project$ cd nodejs-advanced-k8s-project/
vagrant@Master:~/Assesment8/nodejs-k8s-project/nodejs-advanced-k8s-project$ git init
Initialized empty Git repository in /home/vagrant/Assesment8/nodejs-k8s-project/nodejs-
advanced-k8s-project/.git/
vagrant@Master:~/Assesment8/nodejs-k8s-project/nodejs-advanced-k8s-project$
```

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"]
```

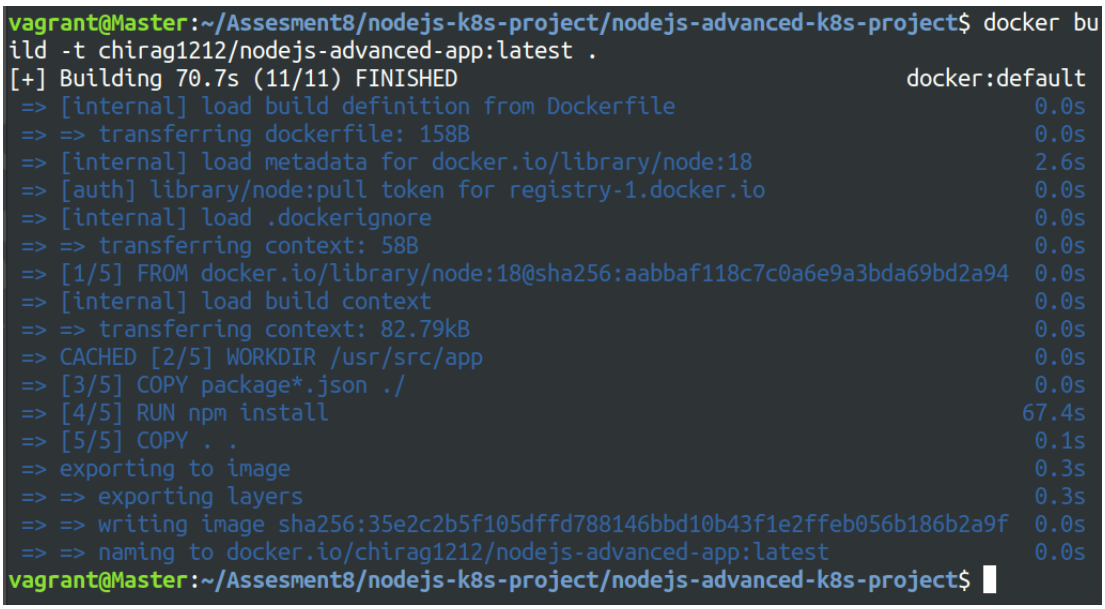
.dockerignore

```
node_modules

.npm
```

1. Build and push Docker image:

```
docker build -t chirag1212/nodejs-advanced-app:latest .
```




```
docker push chirag1212/nodejs-advanced-app:latest
```

```
vagrant@Master:~/Assesment8/nodejs-k8s-project/nodejs-advanced-k8s-project$ docker pu
sh chirag1212/nodejs-advanced-app:latest
The push refers to repository [docker.io/chirag1212/nodejs-advanced-app]
9b3f76123c87: Pushed
f2ae65fb9e37: Pushed
5e8e68c47cae: Pushed
058025a2e5c7: Mounted from chirag1212/nodejs-app
0970e1a837f7: Mounted from chirag1212/nodejs-app
d4061df7c236: Mounted from chirag1212/nodejs-app
9487e6e19e60: Mounted from chirag1212/nodejs-app
6ef00066aa6f: Mounted from chirag1212/nodejs-app
b11bb163e263: Mounted from chirag1212/nodejs-app
b779a72428fa: Mounted from chirag1212/nodejs-app
8ada682d3780: Mounted from chirag1212/nodejs-app
15bb10f9bb3a: Mounted from chirag1212/nodejs-app
latest: digest: sha256:ea5521f35fb6ed28a0e3c7b5337f6ff047589a8ed720bf052f64a309c6f691
95 size: 2839
vagrant@Master:~/Assesment8/nodejs-k8s-project/nodejs-advanced-k8s-project$
```

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: chirag1212/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
```

```yaml
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
```

```
vagrant@Master:~/Assesment8/nodejs-k8s-project/nodejs-advanced-k8s-project$ kubectl apply -f kubernetes/redis-deployment.yaml
deployment.apps/redis-deployment created
vagrant@Master:~/Assesment8/nodejs-k8s-project/nodejs-advanced-k8s-project$
```

```
kubectl apply -f kubernetes/redis-service.yaml
```

```
vagrant@Master:~/Assesment8/nodejs-k8s-project/nodejs-advanced-k8s-project$ kubectl apply -f kubernetes/redis-service.yaml
service/redis-service created
```

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

```
vagrant@Master:~/Assesment8/nodejs-k8s-project/nodejs-advanced-k8s-project$ kubectl apply -f kubernetes/configmap.yaml
configmap/app-config unchanged
configmap/redis-config created
vagrant@Master:~/Assesment8/nodejs-k8s-project/nodejs-advanced-k8s-project$
```

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

```
vagrant@Master:~/Assesment8/nodejs-k8s-project/nodejs-advanced-k8s-project$ kubectl apply -f kubernetes/secret.yaml
secret/app-secrets configured
vagrant@Master:~/Assesment8/nodejs-k8s-project/nodejs-advanced-k8s-project$
```

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

```
vagrant@Master:~/Assesment8/nodejs-k8s-project/nodejs-advanced-k8s-project$ kubectl apply -f kubernetes/deployment.yaml
deployment.apps/nodejs-advanced-app-deployment created
vagrant@Master:~/Assesment8/nodejs-k8s-project/nodejs-advanced-k8s-project$
```

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

```
vagrant@Master:~/Assesment8/nodejs-k8s-project/nodejs-advanced-k8s-project$ kubectl apply -f kubernetes/service.yaml
service/nodejs-advanced-app-service created
vagrant@Master:~/Assesment8/nodejs-k8s-project/nodejs-advanced-k8s-project$
```

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

```
vagrant@Master:~/Assesment8/nodejs-k8s-project/nodejs-advanced-k8s-project$ kubectl apply -f kubernetes/hpa.yaml
horizontalpodautoscaler.autoscaling/nodejs-advanced-app-hpa created
vagrant@Master:~/Assesment8/nodejs-k8s-project/nodejs-advanced-k8s-project$
```

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

```
vagrant@Master:~/Assesment8/nodejs-k8s-project/nodejs-advanced-k8s-project$ kubectl apply -f kubernetes/vpa.yaml
Warning: autoscaling.k8s.io/v1beta2 API is deprecated
verticalpodautoscaler.autoscaling.k8s.io/nodejs-advanced-app-vpa created
vagrant@Master:~/Assesment8/nodejs-k8s-project/nodejs-advanced-k8s-project$
```

2.9 Verify Deployments and Services

Check the status of your deployments and services:

```
kubectl get all
```

```
vagrant@Master:~/Assesment8/nodejs-k8s-project/nodejs-advanced-k8s-project$ kubectl get all
NAME                                                READY   STATUS    RESTARTS   AGE
pod/nodejs-advanced-app-deployment-5f98596dcd-2fr52   2/2     Running   0          13m
pod/nodejs-advanced-app-deployment-5f98596dcd-45vdj   2/2     Running   0          13m
pod/nodejs-app-deployment-bd676bb7c-9bcp2             1/1     Running   0          5h3m
pod/nodejs-app-deployment-bd676bb7c-zms2w             1/1     Running   0          5h3m
pod/redis-deployment-6b5bcbb6b6-dw8c2                 1/1     Running   0          19m

NAME                                  TYPE           CLUSTER-IP       EXTERNAL-IP   PORT(S)          AGE
service/kubernetes                    ClusterIP      10.96.0.1        <none>        443/TCP          6h7m
service/nodejs-advanced-app-service   LoadBalancer   10.111.207.3     <pending>     80:30440/TCP     13m
service/nodejs-app-service            NodePort       10.97.156.68     <none>        3000:30526/TCP   3h58m
service/redis-service                 ClusterIP      10.101.207.143   <none>        6379/TCP         15m

NAME                                           READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/nodejs-advanced-app-deployment   2/2     2            2           13m
deployment.apps/nodejs-app-deployment            2/2     2            2           5h3m
deployment.apps/redis-deployment                 1/1     1            1           19m

NAME                                                      DESIRED   CURRENT   READY   AGE
replicaset.apps/nodejs-advanced-app-deployment-5f98596dcd   2         2         2       13m
replicaset.apps/nodejs-app-deployment-bd676bb7c             2         2         2       5h3m
replicaset.apps/redis-deployment-6b5bcbb6b6                  1         1         1       19m

NAME                                                    REFERENCE                          TARGETS              MINPODS   MAXPODS   REPLICAS   AGE
horizontalpodautoscaler.autoscaling/nodejs-app-hpa      Deployment/nodejs-app-deployment   cpu: <unknown>/50%   2         5         2          4h3m
vagrant@Master:~/Assesment8/nodejs-k8s-project/nodejs-advanced-k8s-project$
```

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
```