**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**

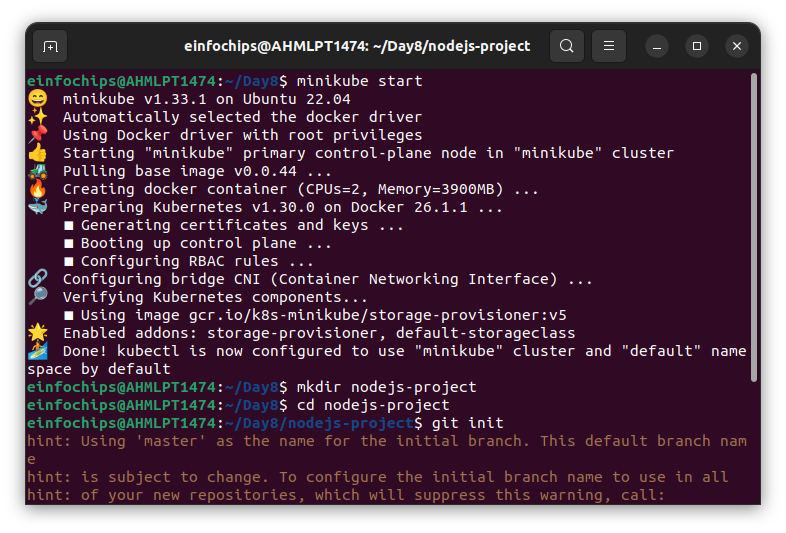
**Start Minikube**:  
  
minikube start

#### **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

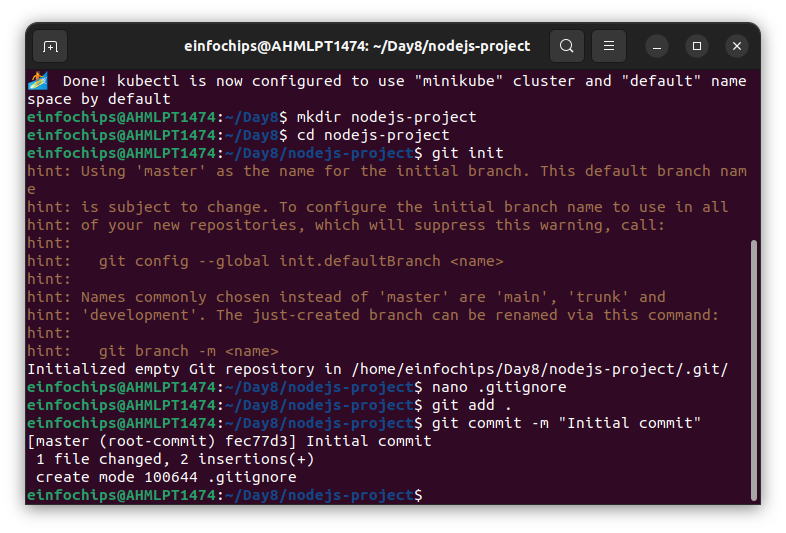


**Create a .gitignore file**:  
  
node\_modules/

.env

**Add and commit initial changes**:  
  
git add .

git commit -m "Initial commit"

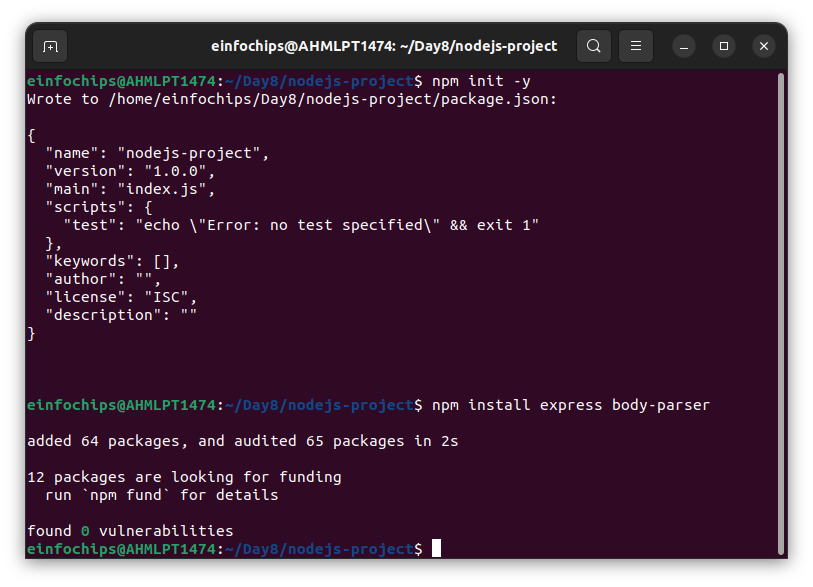


### **2. Develop a Node.js Application**

#### **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, () => {

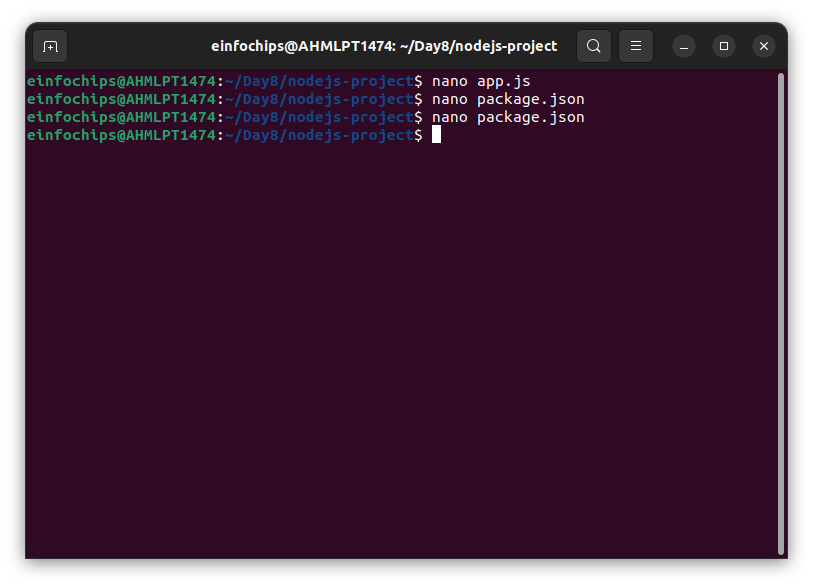
console.log(`Server is running on port ${PORT}`);

});

**Update package.json** to include a start script:  
  
"scripts": {

"start": "node app.js"

}



#### **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

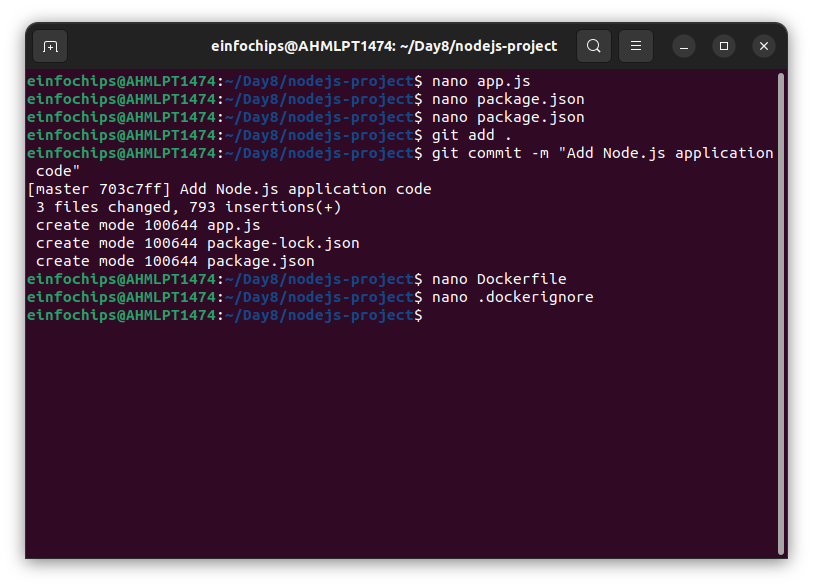
EXPOSE 3000

# 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:

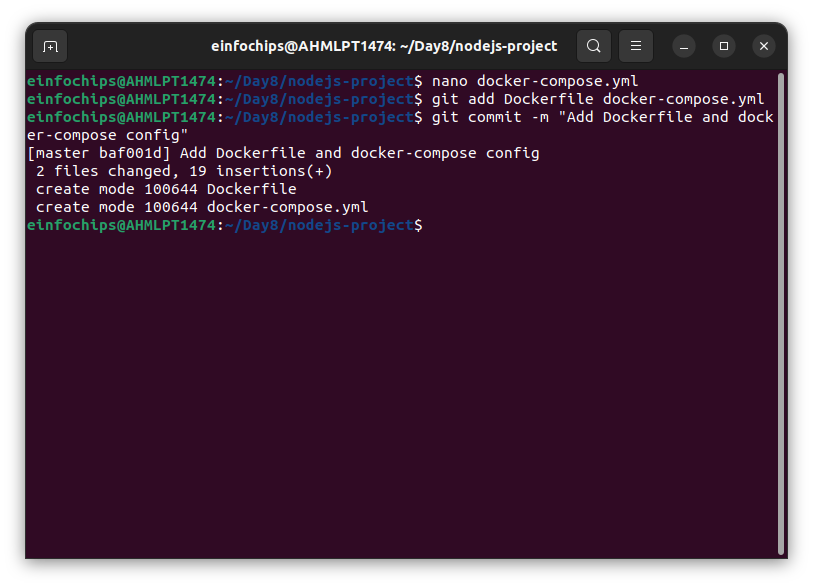
build: .

ports:

- "3000:3000"

**Add and commit changes**:  
  
git add Dockerfile docker-compose.yml

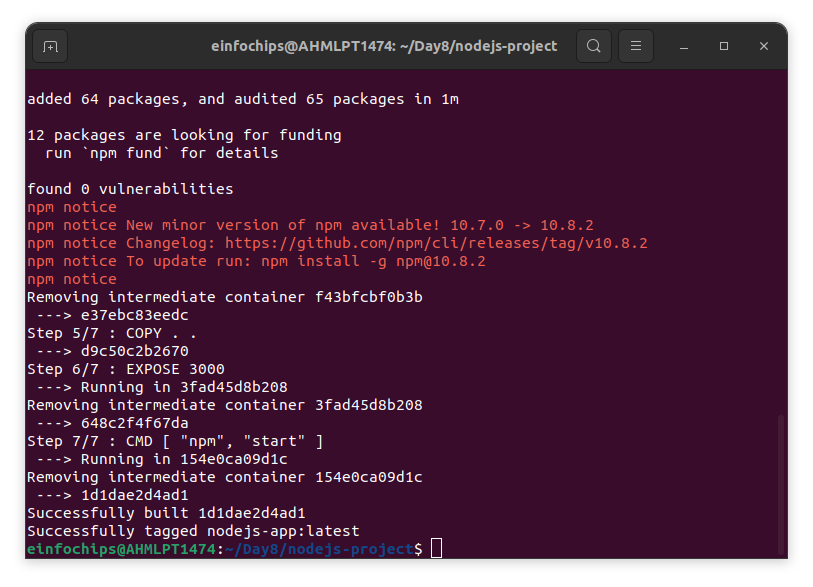
git commit -m "Add Dockerfile and Docker Compose configuration"



### **4. Build and Push Docker Image**

#### **4.1 Build Docker Image**

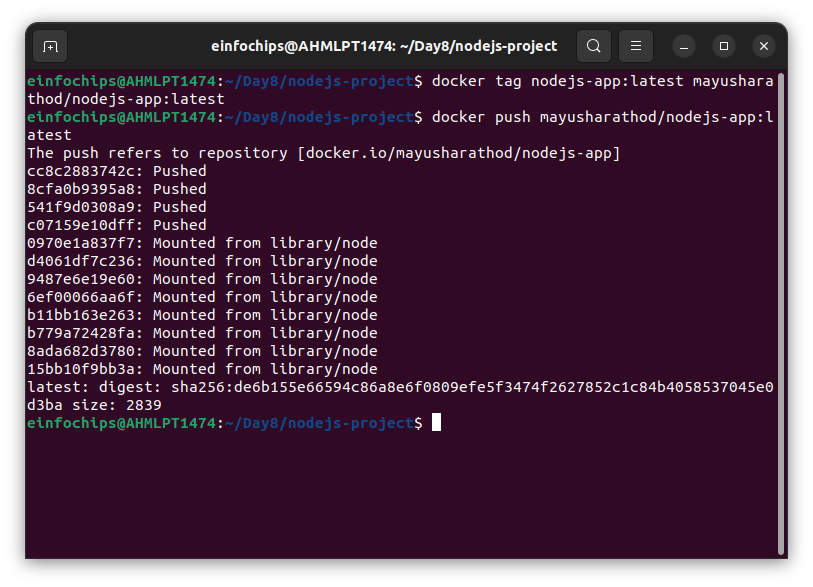
**Build the Docker image**:  
  
docker build -t nodejs-app:latest .



#### **4.2 Push Docker Image to Docker Hub**

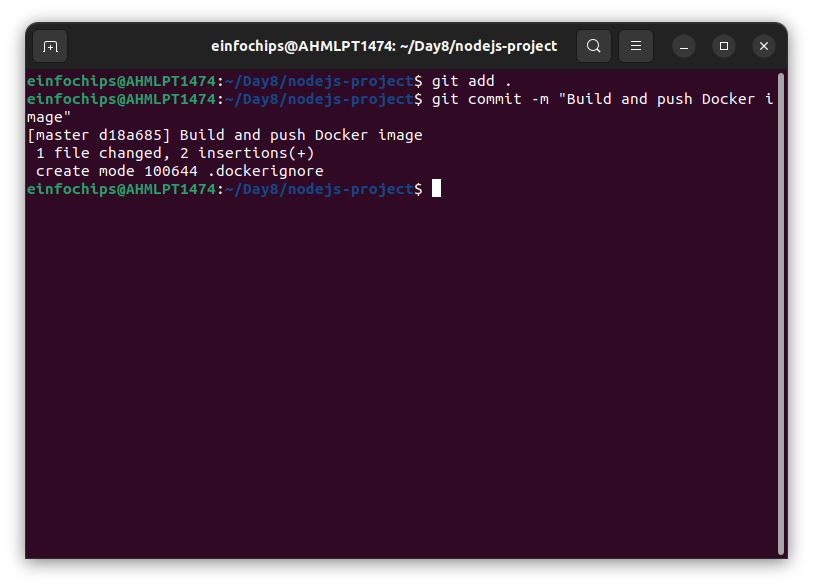
**Tag and push the image**:  
  
docker tag nodejs-app:latest your-dockerhub-username/nodejs-app:latest

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



**Add and commit changes**:  
  
git add .

git commit -m "Build and push Docker image"



### **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-secretsgit

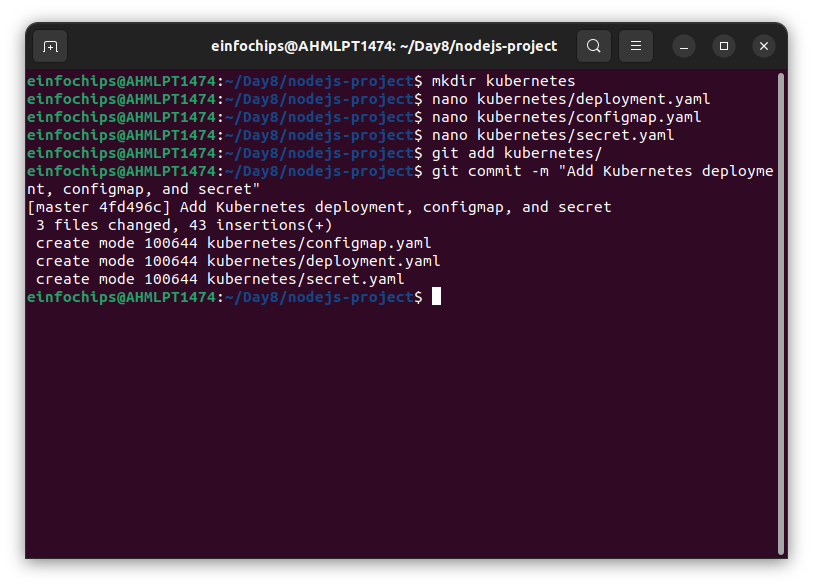
type: Opaque

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"

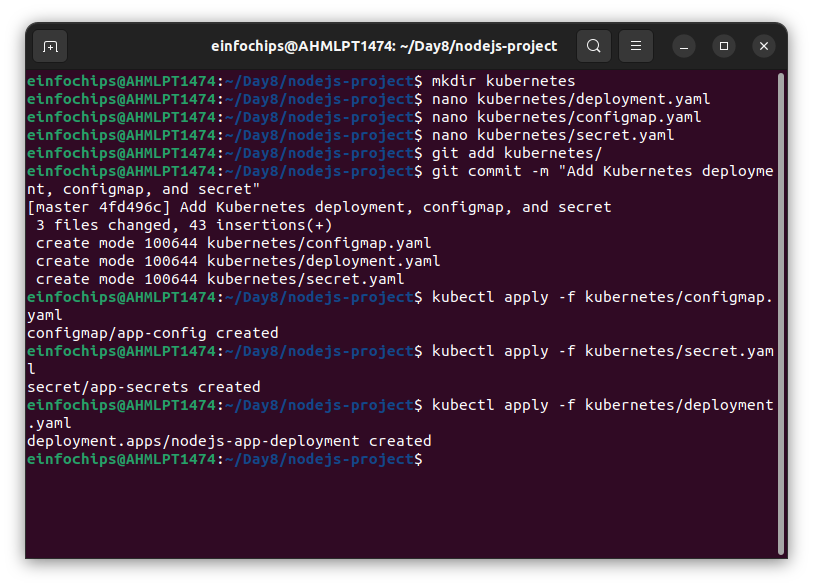


#### **5.3 Apply Kubernetes Configurations**

**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



### **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:

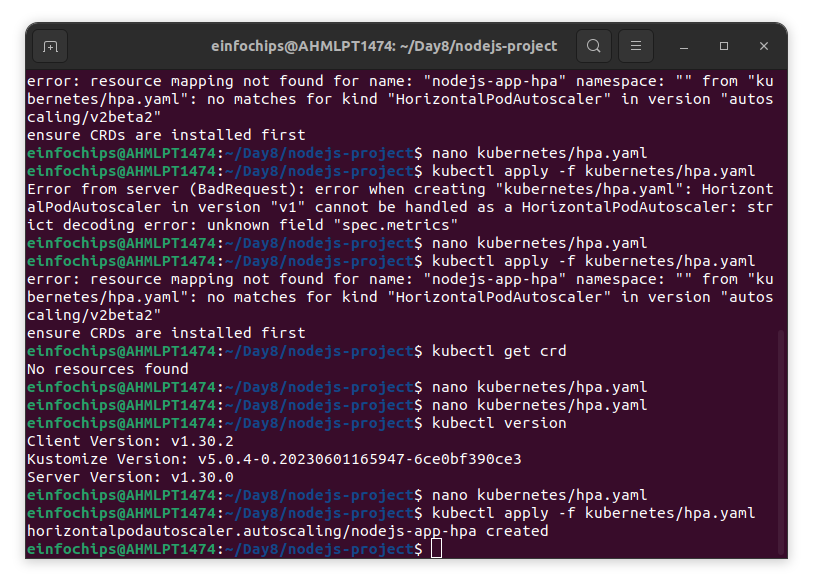
name: cpu

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

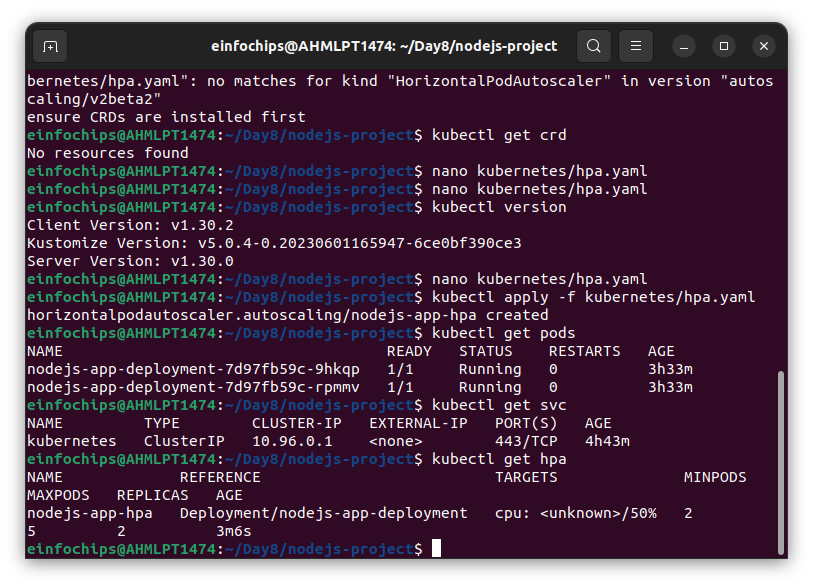
### **7. Test the Deployment**

#### **7.1 Check the Status of Pods, Services, and HPA**

**Verify the Pods**:  
  
kubectl get pods

**Verify the Services**:  
  
kubectl get svc

**Verify the HPA**:  
  
kubectl get hpa



#### **7.2 Access the Application**

**Expose the Service**:  
  
kubectl expose deployment nodejs-app-deployment --type=NodePort --name=nodejs-app-service

**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.

