Project 01

Deploying a Node.js App Using Minikube Kubernetes

Overview

This project guides you through deploying a Node.js application using Minikube Kubernetes. You'll use Git for version control, explore branching and fast-forward merges, and set up Kubernetes services and deployment pods, including ClusterIP and NodePort service types.

Prerequisites

- Minikube installed
- kubectl installed
- Git installed
- Node.js installed (https://nodejs.org/en/download/package-manager/all#debian-and-ubuntu-based-linux-distributions)

Project Steps

1. Set Up Git Version Control

1.1. Initialize a Git Repository

Create a new directory for your project:

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

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

Initialize a Git repository:

```
git init
```

1.2. Create a Node.js Application

Initialize a Node.js project:

```
npm init -y
```

```
einfochips@PUNELPT0728:~/vagrant-vms/k8s/kubernetes-the-hard-way/vagrant$ cd
einfochips@PUNELPT0728:~$ cd DevOps-assessment/
einfochips@PUNELPT0728:~/DevOps-assessment$ cd Assessment\ 7/
einfochips@PUNELPT0728:~/DevOps-assessment/Assessment 7$ ls
einfochips@PUNELPT0728:~/DevOps-assessment/Assessment 7$ mkdir nodejs-k8s-project
einfochips@PUNELPT0728:~/DevOps-assessment/Assessment 7$ cd nodejs-k8s-project/
einfochips@PUNELPT0728:~/DevOps-assessment/Assessment 7/nodejs-k8s-project$ git init
Initialized empty Git repository in /home/einfochips/DevOps-assessment/Assessment 7/nodejs-k8s-project/.git/
einfochips@PUNELPT0728:~/DevOps-assessment/Assessment 7/nodejs-k8s-project$ npm init -y
Wrote to /home/einfochips/DevOps-assessment/Assessment 7/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": ""
}
```

Install Express.js:

```
npm install express
```

```
einfochips@PUNELPT0728:~/DevOps-assessment/Assessment 7/nodejs-k8s-project$ npm install express

added 64 packages, and audited 65 packages in 3s

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

found 0 vulnerabilities
```

Create an index.js file with the following content:

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

const app = express();

const port = 3000;


app.get('/', (req, res) => {

    res.send('Hello, Kubernetes!');

});


app.listen(port, () => {

    console.log(`App running at http://localhost:${port}`);

});
```

1.

Create a .gitignore file to ignore node_modules:

node_modules

```
einfochips@PUNELPT0728:~/DevOps-assessment/Assessment 7/nodejs-k8s-project$ code index.js
einfochips@PUNELPT0728:~/DevOps-assessment/Assessment 7/nodejs-k8s-project$ code .gitignore
einfochips@PUNELPT0728:~/DevOps-assessment/Assessment 7/nodejs-k8s-project$ ls
index.js  node_modules  package.json  package-lock.json
einfochips@PUNELPT0728:~/DevOps-assessment/Assessment 7/nodejs-k8s-project$ ls -la
total 56
drwxrwxr-x  4 einfochips einfochips  4096 Jul 16 10:08 .
drwxrwxr-x  3 einfochips einfochips  4096 Jul 16 10:05 ..
drwxrwxr-x  7 einfochips einfochips  4096 Jul 16 10:05 .git
-rw-rw-r--  1 einfochips einfochips    13 Jul 16 10:08 .gitignore
-rw-rw-r--  1 einfochips einfochips   237 Jul 16 10:06 index.js
drwxrwxr-x 66 einfochips einfochips  4096 Jul 16 10:06 node_modules
-rw-rw-r--  1 einfochips einfochips   282 Jul 16 10:06 package.json
-rw-rw-r--  1 einfochips einfochips 25511 Jul 16 10:06 package-lock.json
```

1.3. Commit the Initial Code

Add files to Git:

git add .

Commit the changes:

git commit -m "Initial commit with Node.js app"

```
einfochips@PUNELPT0728:~/DevOps-assessment/Assessment 7/nodejs-k8s-project$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   .gitignore
        new file:   index.js
        new file:   package-lock.json
        new file:   package.json

einfochips@PUNELPT0728:~/DevOps-assessment/Assessment 7/nodejs-k8s-project$ git commit -m "Initial commit with Node.js app"
[master (root-commit) 01cd984] Initial commit with Node.js app
 4 files changed, 724 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 index.js
 create mode 100644 package-lock.json
 create mode 100644 package.json
```

2. Branching and Fast-Forward Merge

2.1. Create a New Branch

Create and switch to a new branch feature/add-route:

git checkout -b feature/add-route

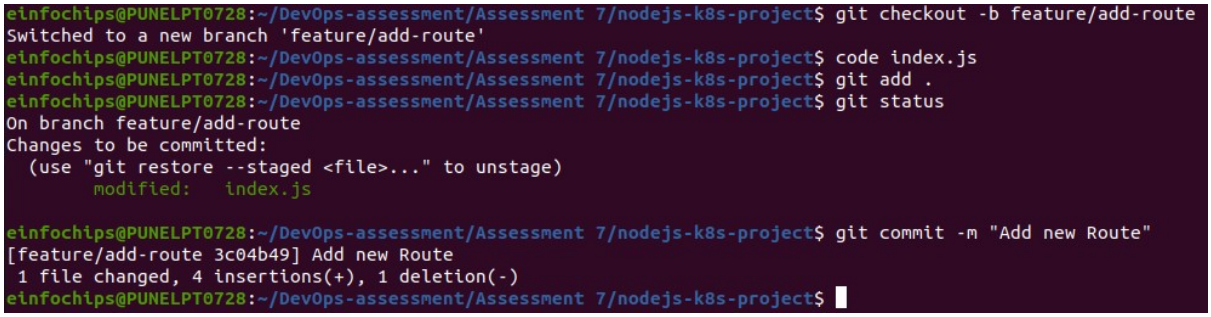
2.2. Implement a New Route

Modify index.js to add a new route:

```
app.get('/newroute', (req, res) => {
    res.send('This is a new route!');
});
```

Commit the changes:

```
git add .
git commit -m "Add new route"
```



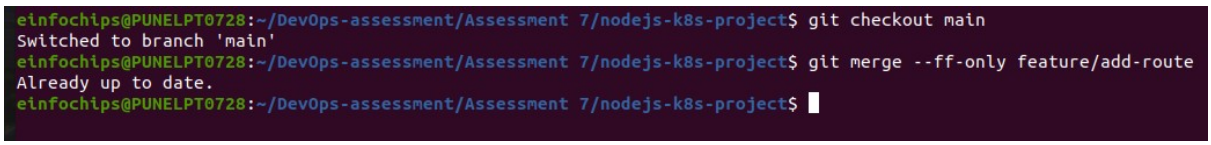
2.3. Merge the Branch Using Fast-Forward

Switch back to the main branch:

```
git checkout main
```

Merge the feature/add-route branch using fast-forward:

```
git merge --ff-only feature/add-route
```



Delete the feature branch:

```
git branch -d feature/add-route
```

3. Containerize the Node.js Application

3.1. Create a Dockerfile

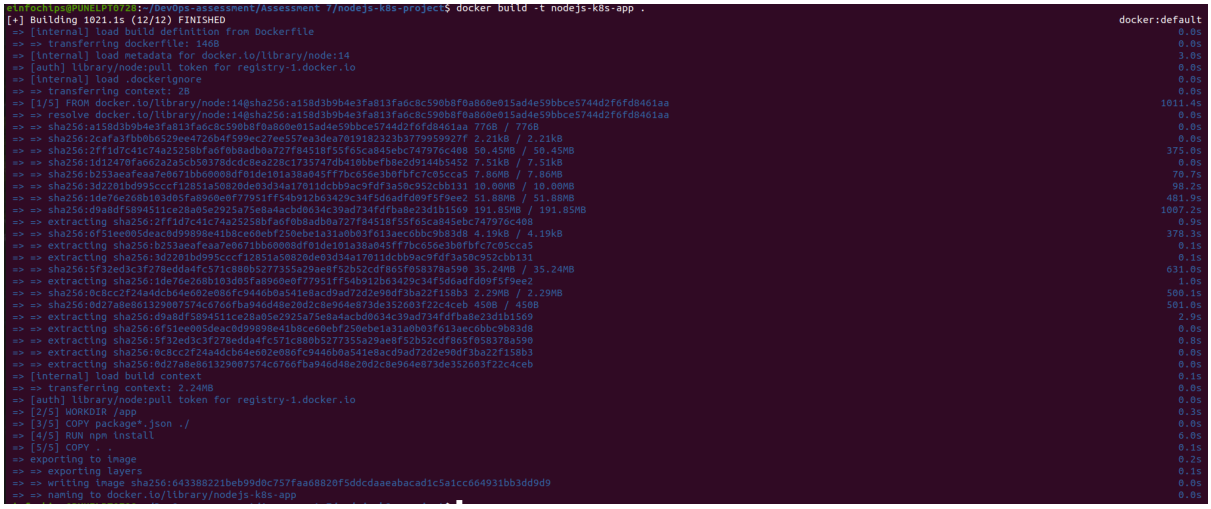
Create a Dockerfile with the following content:

```
FROM node:14

WORKDIR /app

COPY package*.json ./

RUN npm install

COPY . .

EXPOSE 3000

CMD ["node", "index.js"]
```

3.2. Build and Test the Docker Image

Build the Docker image:

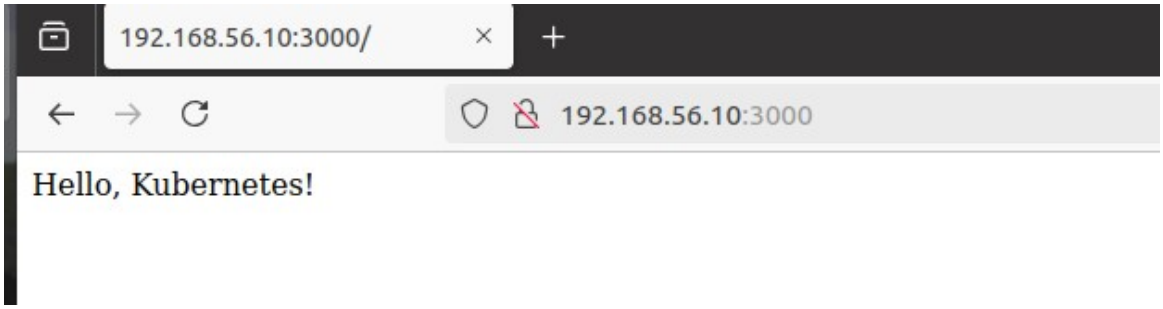
```
docker build -t nodejs-k8s-app .
```



Run the Docker container to test:

```
docker run -p 3000:3000 nodejs-k8s-app
```

1. Access http://localhost:3000 to see the app running.



4. Deploying to Minikube Kubernetes

4.1. Start Minikube

Start Minikube:

```
minikube start
```

4.2. Create Kubernetes Deployment and Service Manifests

Create a deployment.yaml file:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nodejs-app
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nodejs-app
  template:
    metadata:
      labels:
        app: nodejs-app
    spec:
      containers:
```

```
      - name: nodejs-app
        image: nodejs-k8s-app:latest
        ports:
        - containerPort: 3000
```

Create a service.yaml file for ClusterIP:

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

Create a service-nodeport.yaml file for NodePort:

```
apiVersion: v1
kind: Service
metadata:
  name: nodejs-service-nodeport
spec:
  selector:
    app: nodejs-app
  ports:
  - protocol: TCP
    port: 80
```

```
    targetPort: 3000
    nodePort: 30001
  type: NodePort
```

4.3. Apply Manifests to Minikube

Apply the deployment:

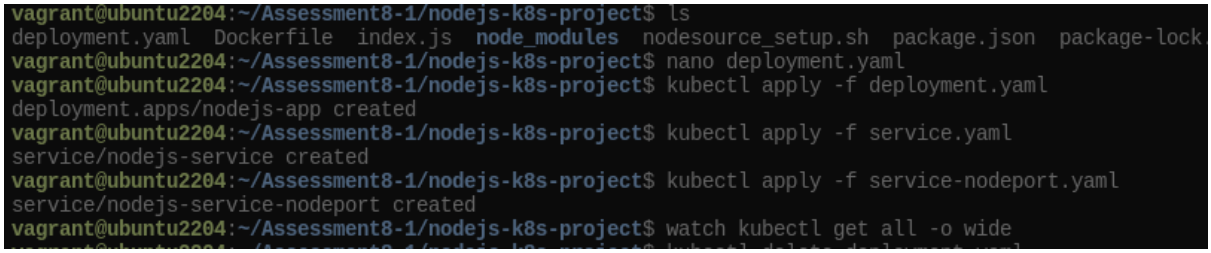
```
kubectl apply -f deployment.yaml
```

Apply the ClusterIP service:

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

Apply the NodePort service:

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



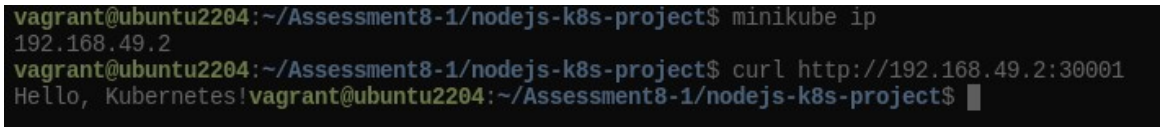
4.4. Access the Application

Get the Minikube IP:

```
minikube ip
```

1. Access the application using the NodePort:

   ```
   curl http://<minikube-ip>:30001
   ```



Making Changes to the App and Redeploying Using Kubernetes

6. Making Changes to the Node.js Application

6.1. Create a New Branch for Changes

Create and switch to a new branch feature/update-message:

```
git checkout -b feature/update-message
```

6.2. Update the Application

Modify index.js to change the message:

```
const express = require('express');
const app = express();
const port = 3000;

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

app.get('/newroute', (req, res) => {
    res.send('This is a new route!');
});

app.listen(port, () => {
    console.log(`App running at http://localhost:${port}`);
});
```

6.3. Commit the Changes

Add and commit the changes:

```
git add .
git commit -m "Update main route message"
```

```
vagrant@ubuntu2204:~/Assessment8-1/nodejs-k8s-project$ nano index.js
vagrant@ubuntu2204:~/Assessment8-1/nodejs-k8s-project$
vagrant@ubuntu2204:~/Assessment8-1/nodejs-k8s-project$ git add .
vagrant@ubuntu2204:~/Assessment8-1/nodejs-k8s-project$ git commit -m "Update main route message"
[feature/update-message ad40280] Update main route message
 5 files changed, 60 insertions(+), 1 deletion(-)
 create mode 100644 Dockerfile
 create mode 100644 deployment.yaml
 create mode 100644 service-nodeport.yaml
 create mode 100644 service.yaml
```

7. Merge the Changes and Rebuild the Docker Image

7.1. Merge the Feature Branch

Switch back to the main branch:

```
git checkout main
```

Merge the feature/update-message branch:

```
git merge --ff-only feature/update-message
```

Delete the feature branch:

```
git branch -d feature/update-message
```

```
vagrant@ubuntu2204:~/Assessment8-1/nodejs-k8s-project$ git checkout main
Switched to branch 'main'
vagrant@ubuntu2204:~/Assessment8-1/nodejs-k8s-project$ git merge --ff-only feature/update-message
Updating c571c6c..ad40280
Fast-forward
 Dockerfile           |  8 ++++++++
 deployment.yaml      | 19 ++++++++++++++++++
 index.js             |  7 ++++++-
 service-nodeport.yaml | 14 ++++++++++++++
 service.yaml         | 13 +++++++++++++
 5 files changed, 60 insertions(+), 1 deletion(-)
 create mode 100644 Dockerfile
 create mode 100644 deployment.yaml
 create mode 100644 service-nodeport.yaml
 create mode 100644 service.yaml
vagrant@ubuntu2204:~/Assessment8-1/nodejs-k8s-project$ git branch -d feature/update-message
Deleted branch feature/update-message (was ad40280).
```

7.2. Rebuild the Docker Image

Rebuild the Docker image with a new tag:

```
docker build -t nodejs-k8s-app:v2 .
```

8. Update Kubernetes Deployment

8.1. Update the Deployment Manifest

Modify deployment.yaml to use the new image version:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nodejs-app
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nodejs-app
  template:
    metadata:
      labels:
        app: nodejs-app
    spec:
      containers:
      - name: nodejs-app
        image: nodejs-k8s-app:v2
        ports:
        - containerPort: 3000
```

8.2. Apply the Updated Manifest

Apply the updated deployment:

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

```
vagrant@ubuntu2204:~/Assessment8-1/nodejs-k8s-project$ docker images
REPOSITORY                   TAG                IMAGE ID       CREATED          SIZE
nodejs-k8s-app               v2                 2a33e84394dd   11 seconds ago   919MB
daradesudarshan/centralrepo  latest             d271ed85b69a   2 hours ago      919MB
daradesudarshan/centralrepo  nodejs-k8s-app-v1  d271ed85b69a   2 hours ago      919MB
nodejs-k8s-app               latest             d271ed85b69a   2 hours ago      919MB
jenkins/jenkins              <none>             680da6c324fc   6 days ago       469MB
wordpress                    latest             d2a2d7e671fd   3 weeks ago      685MB
nginx                        latest             fffffc90d343   3 weeks ago      188MB
postgres                     <none>             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@ubuntu2204:~/Assessment8-1/nodejs-k8s-project$ docker image tag nodejs-k8s-app:v2 daradesudarshan/centralrepo:nodejs-k8s-app-v2
vagrant@ubuntu2204:~/Assessment8-1/nodejs-k8s-project$ docker push daradesudarshan/centralrepo:nodejs-k8s-app-v2
The push refers to repository [docker.io/daradesudarshan/centralrepo]
01a5308eb478: Pushed
e0b48c3227b5: Layer already exists
584012326bb4: Layer already exists
9dca9a1ce4b3: Layer already exists
0d5f5a015e5d: Layer already exists
3c777d951de2: Layer already exists
f8a91dd5fc84: Layer already exists
cb81227abde5: Layer already exists
e01a454893a9: Layer already exists
c45660adde37: Layer already exists
fe0fb3ab4a0f: Layer already exists
f1186e5061f2: Layer already exists
b2dba7477754: Layer already exists
nodejs-k8s-app-v2: digest: sha256:ad1efc7b923a29e76e4bca73c3e77f1b6fc5cdcc8b8f7f402d145caebc7b139a size: 3050
vagrant@ubuntu2204:~/Assessment8-1/nodejs-k8s-project$ nano deployment.yaml
vagrant@ubuntu2204:~/Assessment8-1/nodejs-k8s-project$ kubectl apply -f deployment.yaml
deployment.apps/nodejs-app configured
vagrant@ubuntu2204:~/Assessment8-1/nodejs-k8s-project$
```

minikube

8.3. Verify the Update

Check the status of the deployment:

```
kubectl rollout status deployment/nodejs-app
```

```
vagrant@ubuntu2204:~$ kubectl rollout status deployment/nodejs-app
deployment "nodejs-app" successfully rolled out
vagrant@ubuntu2204:~$
```

9. Access the Updated Application

9.1. Access Through ClusterIP Service

Forward the port to access the ClusterIP service:

```
kubectl port-forward service/nodejs-service 8080:80
```

1. Open your browser and navigate to http://localhost:8080 to see the updated message.

9.2. Access Through NodePort Service

1. Access the application using the NodePort:

   ```
   curl http://<minikube-ip>:30001
   ```

```
vagrant@ubuntu2204:~/Assessment8-1/nodejs-k8s-project$ curl http://192.168.49.2:30001
Hello, Kubernetes! Updated version.vagrant@ubuntu2204:~/Assessment8-1/nodejs-k8s-project$
```

Project 02

Deploying a Python Flask App Using Minikube Kubernetes

Overview

This project guides you through deploying a Python Flask application using Minikube Kubernetes. You'll use Git for version control, explore branching and fast-forward merges, and set up Kubernetes services and deployment pods, including ClusterIP and NodePort service types.

Prerequisites

- Minikube installed
- kubectl installed
- Git installed
- Python installed

Project Steps

1. Set Up Git Version Control

1.1. Initialize a Git Repository

Create a new directory for your project:

```
mkdir flask-k8s-project
```

```
cd flask-k8s-project
```

Initialize a Git repository:
sh
Copy code

```
git init
```

1.2. Create a Python Flask Application

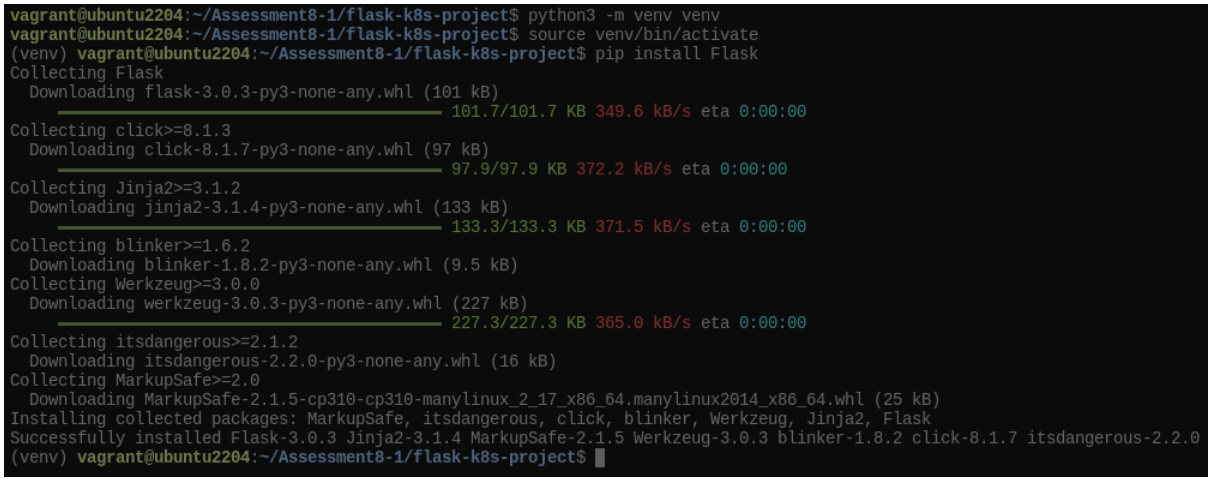
Create a virtual environment:

```
python -m venv venv
```

```
source venv/bin/activate
```

Install Flask:
sh
Copy code

```
pip install Flask
```



Create an app.py file with the following content:
python
Copy code

```
from flask import Flask


app = Flask(__name__)


@app.route('/')
def hello_world():
    return 'Hello, Kubernetes!'


if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)
```

Create a requirements.txt file to list the dependencies:
Copy code

```
Flask
```

Create a .gitignore file to ignore venv:
Copy code

```
venv
```

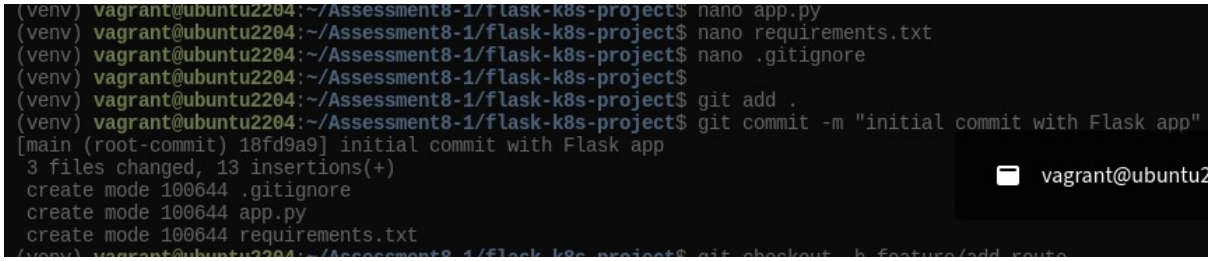
1.3. Commit the Initial Code

Add files to Git:

```
git add .
```

Commit the changes:

```
git commit -m "Initial commit with Flask app"
```



2. Branching and Fast-Forward Merge

2.1. Create a New Branch

Create and switch to a new branch feature/add-route:

```
git checkout -b feature/add-route
```

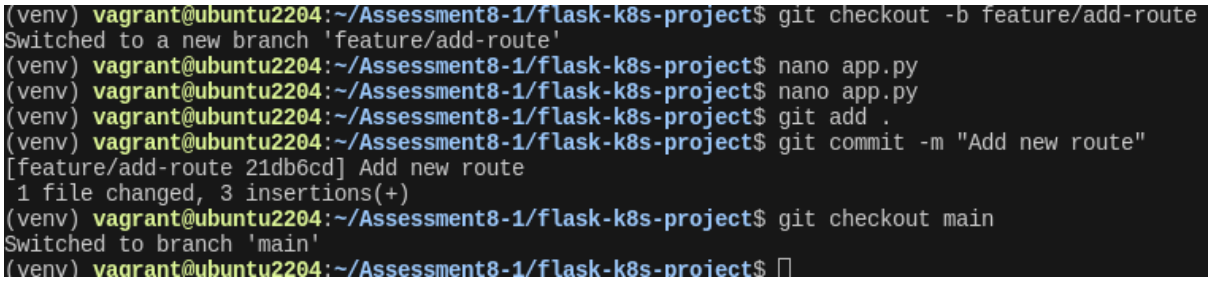
2.2. Implement a New Route

Modify app.py to add a new route:

```
@app.route('/newroute')
def new_route():
    return 'This is a new route!'
```

Commit the changes:

```
git add .
git commit -m "Add new route"
```



2.3. Merge the Branch Using Fast-Forward

Switch back to the main branch:

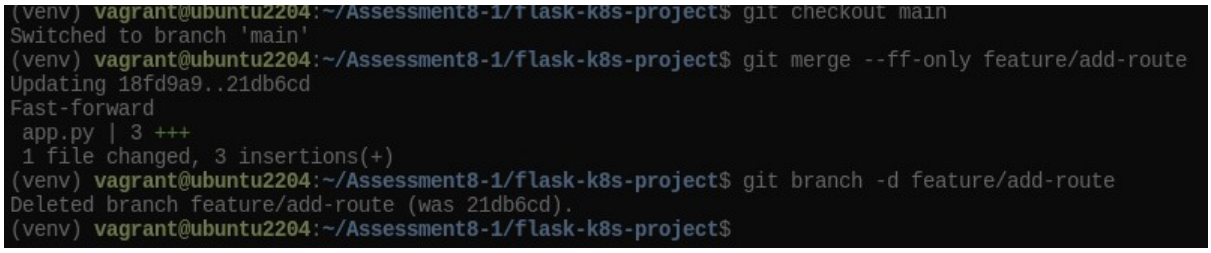
```
git checkout main
```

Merge the feature/add-route branch using fast-forward:

```
git merge --ff-only feature/add-route
```

Delete the feature branch:

```
git branch -d feature/add-route
```



3. Containerize the Flask Application

3.1. Create a Dockerfile

Create a Dockerfile with the following content:

```
FROM python:3.8-slim

WORKDIR /app

COPY requirements.txt requirements.txt
RUN pip install -r requirements.txt

COPY . .

EXPOSE 5000

CMD ["python", "app.py"]
```

3.2. Build and Test the Docker Image

Build the Docker image:

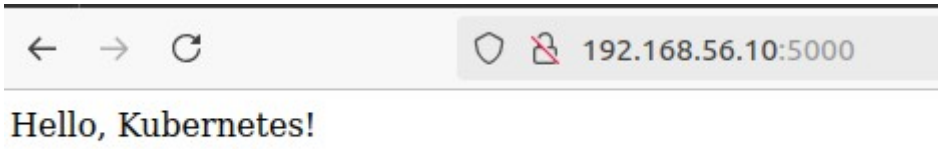
```
docker build -t flask-k8s-app .
```

```
(venv) vagrant@ubuntu2204:~/Assessment8-1/flask-k8s-project$ nano Dockerfile
(venv) vagrant@ubuntu2204:~/Assessment8-1/flask-k8s-project$ docker build -t flask-k8s-app .
[+] Building 85.8s (11/11) FINISHED                                                                 docker:default
 => [internal] load build definition from Dockerfile                                                         0.0s
 => => transferring dockerfile: 201B                                                                         0.0s
 => [internal] load metadata for docker.io/library/python:3.8-slim                                           3.4s
 => [auth] library/python:pull token for registry-1.docker.io                                                0.0s
 => [internal] load .dockerignore                                                                            0.0s
 => => transferring context: 2B                                                                              0.0s
 => [1/5] FROM docker.io/library/python:3.8-slim@sha256:463e5f5018b45cc2621ec7308df9ecaaf87deaf8fd88b2      52.0s
 => => resolve docker.io/library/python:3.8-slim@sha256:463e5f5018b45cc2621ec7308df9ecaaf87deaf8fd88b28      0.0s
 => => sha256:463e5f5018b45cc2621ec7308df9ecaaf87deaf8fd88b28502659adf24b1662a 10.41kB / 10.41kB             0.0s
 => => sha256:5af562618afca22211e56e80937c73d14923ad8fdef094f9084017fcdcdc6a8b 1.94kB / 1.94kB               0.0s
 => => sha256:aa3d4ef002c94a265ec8557e5348ef83dd1313beaab2a6bf56b430406505a039 6.95kB / 6.95kB               0.0s
 => => sha256:ac00c4d4c9c021c370a57f4867988627383ca8b1611ef85d566ab6f9f557de83 3.51MB / 3.51MB              21.6s
 => => sha256:c3c6f012f594262870ed238edaf0c7ee676ce92b61a02e8e4de81b0a92aeff7e 11.67MB / 11.67MB            51.2s
 => => sha256:1133e24b6550d31922d185bc08a8a5d1238cd26d05e06824d4c3fbb113f302b2 238B / 238B                   1.8s
 => => sha256:baafb1a12380ef970ecf90029340fe1d54cb1260deac3ae3e99a8ef6f14c406b 2.78MB / 2.78MB              35.5s
 => => extracting sha256:ac00c4d4c9c021c370a57f4867988627383ca8b1611ef85d566ab6f9f557de83                    0.1s
 => => extracting sha256:c3c6f012f594262870ed238edaf0c7ee676ce92b61a02e8e4de81b0a92aeff7e                    0.5s
 => => extracting sha256:1133e24b6550d31922d185bc08a8a5d1238cd26d05e06824d4c3fbb113f302b2                    0.0s
 => => extracting sha256:baafb1a12380ef970ecf90029340fe1d54cb1260deac3ae3e99a8ef6f14c406b                    0.2s
 => [internal] load build context                                                                            0.4s
 => => transferring context: 20.02MB                                                                         0.4s
 => [2/5] WORKDIR /app                                                                                       0.1s
 => [3/5] COPY requirements.txt requirements.txt                                                             0.0s
 => [4/5] RUN pip install -r requirements.txt                                                               29.6s
 => [5/5] COPY . .                                                                                           0.4s
 => exporting to image                                                                                       0.2s
 => => exporting layers                                                                                      0.2s
 => => writing image sha256:351529b10559e65649181b6ee5f65c2585c8cb8e9cdbe3867d51bd096aae24f0                 0.0s
 => => naming to docker.io/library/flask-k8s-app                                                             0.0s
```

Run the Docker container to test:

```
docker run -p 5000:5000 flask-k8s-app
```

1.
2. Access http://localhost:5000 to see the app running.



4. Deploying to Minikube Kubernetes

4.1. Start Minikube

Start Minikube:

```
minikube start
```

4.2. Create Kubernetes Deployment and Service Manifests

Create a deployment.yaml file:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: flask-app
spec:
  replicas: 2
  selector:
    matchLabels:
      app: flask-app
  template:
    metadata:
      labels:
        app: flask-app
    spec:
      containers:
      - name: flask-app
        image: flask-k8s-app:latest
        ports:
        - containerPort: 5000
```

Create a service.yaml file for ClusterIP:

```
apiVersion: v1
kind: Service
metadata:
  name: flask-service
```

```
spec:
  selector:
    app: flask-app
  ports:
  - protocol: TCP
    port: 80
    targetPort: 5000
  type: ClusterIP
```

Create a service-nodeport.yaml file for NodePort:

```
apiVersion: v1
kind: Service
metadata:
  name: flask-service-nodeport
spec:
  selector:
    app: flask-app
  ports:
  - protocol: TCP
    port: 80
    targetPort: 5000
    nodePort: 30001
  type: NodePort
```

4.3. Apply Manifests to Minikube

Apply the deployment:

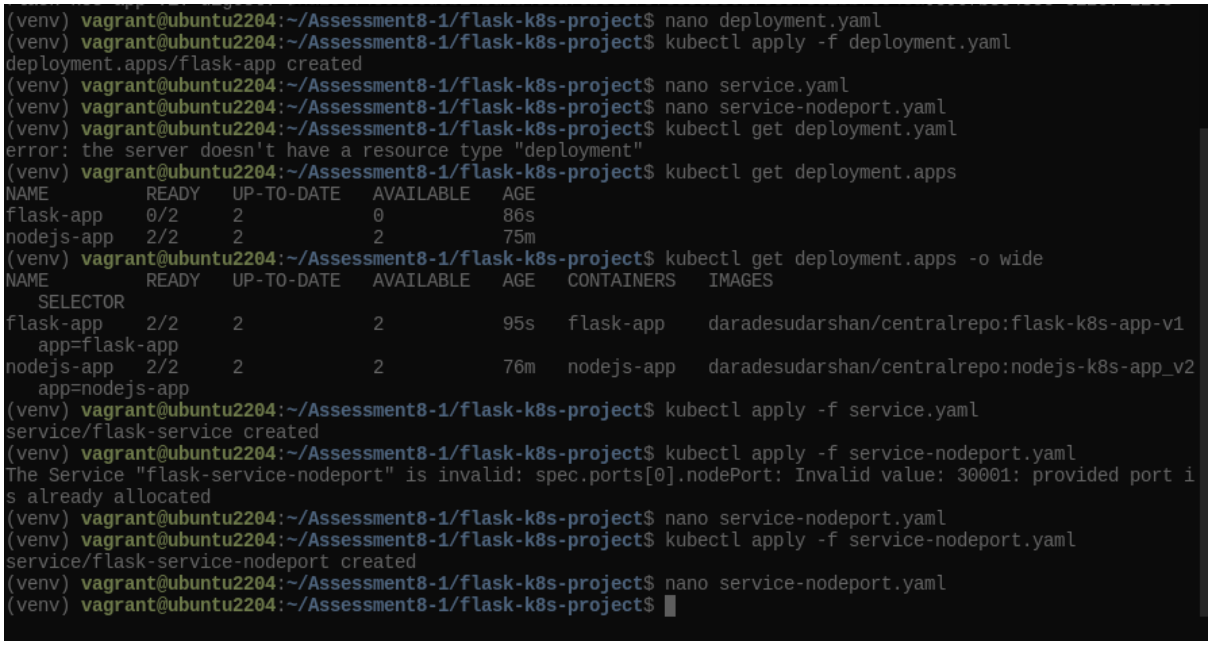
```
kubectl apply -f deployment.yaml
```

Apply the ClusterIP service:

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

Apply the NodePort service:

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



4.4. Access the Application

Get the Minikube IP:

```
minikube ip
```

Access the application using the NodePort:

```
curl http://<minikube-ip>:30001
```

5. Clean Up

Stop Minikube:

```
minikube stop
```

Delete Minikube cluster:

```
minikube delete
```

6. Making Changes to the Flask Application

6.1. Create a New Branch for Changes

Create and switch to a new branch feature/update-message:

```
git checkout -b feature/update-message
```

6.2. Update the Application

Modify app.py to change the message:

```
@app.route('/')
def hello_world():
    return 'Hello, Kubernetes! Updated version.'

@app.route('/newroute')
def new_route():
    return 'This is a new route!'
```

6.3. Commit the Changes

Add and commit the changes:

```
git add .
git commit -m "Update main route message"
```

```
(venv) vagrant@ubuntu2204:~/Assessment8-1/flask-k8s-project$ git checkout -b feature/update-message
Switched to a new branch 'feature/update-message'
(venv) vagrant@ubuntu2204:~/Assessment8-1/flask-k8s-project$ nano app.py
(venv) vagrant@ubuntu2204:~/Assessment8-1/flask-k8s-project$ git add .
(venv) vagrant@ubuntu2204:~/Assessment8-1/flask-k8s-project$ git commit -m " Update main route message"
[feature/update-message 1ba5552]  Update main route message
 5 files changed, 59 insertions(+), 1 deletion(-)
 create mode 100644 Dockerfile
 create mode 100644 deployment.yaml
 create mode 100644 service-nodeport.yaml
 create mode 100644 service.yaml
(venv) vagrant@ubuntu2204:~/Assessment8-1/flask-k8s-project$
```

7. Merge the Changes and Rebuild the Docker Image

7.1. Merge the Feature Branch

Switch back to the main branch:

```
git checkout main
```

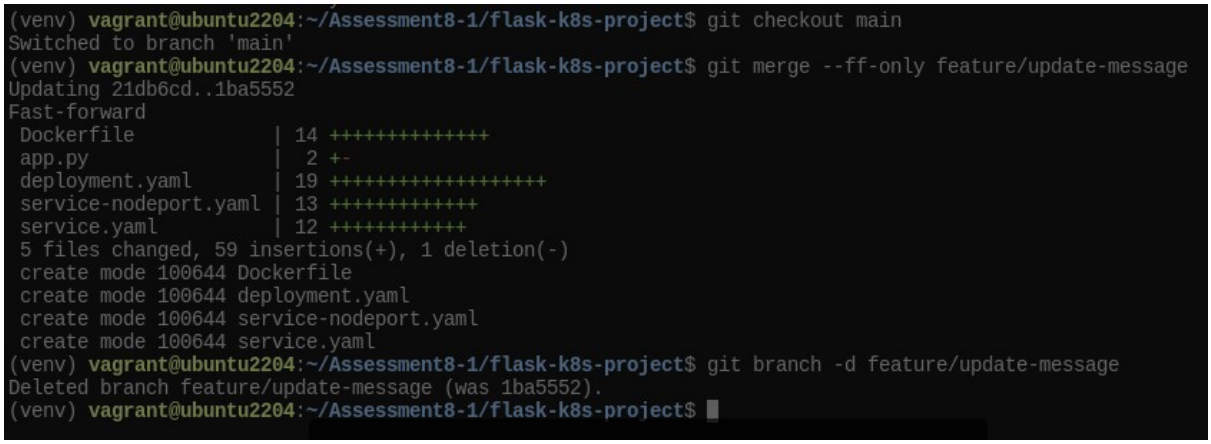
1.

Merge the feature/update-message branch:

```
git merge --ff-only feature/update-message
```

Delete the feature branch:

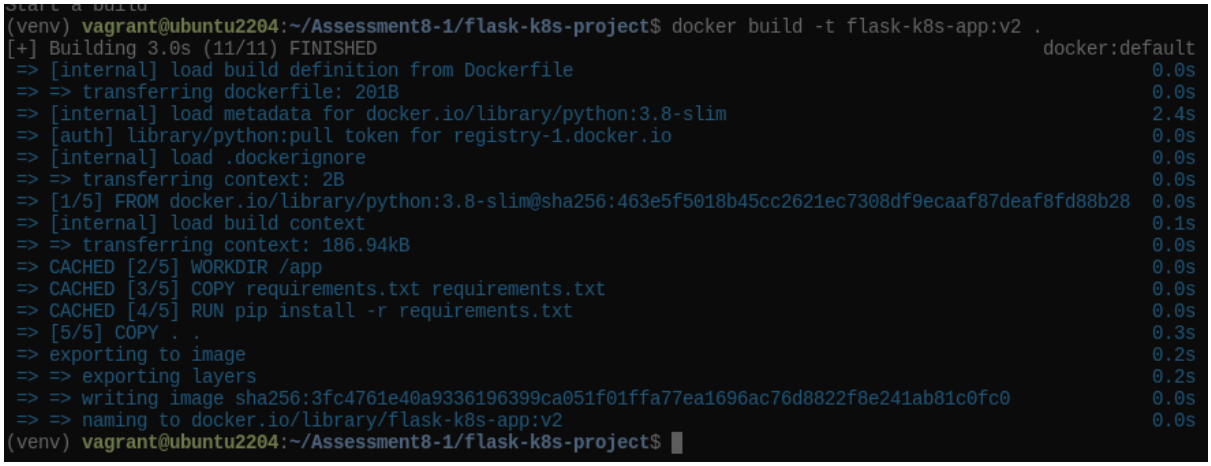
```
git branch -d feature/update-message
```



7.2. Rebuild the Docker Image

Rebuild the Docker image with a new tag:

```
docker build -t flask-k8s-app:v2 .
```



8. Update Kubernetes Deployment

8.1. Update the Deployment Manifest

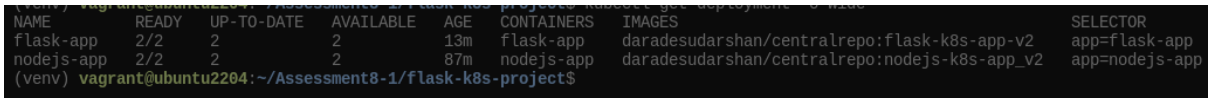
Modify deployment.yaml to use the new image version:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: flask-app
spec:
  replicas: 2
  selector:
    matchLabels:
      app: flask-app
  template:
    metadata:
      labels:
        app: flask-app
    spec:
      containers:
      - name: flask-app
        image: flask-k8s-app:v2
```

```
ports:

- containerPort: 5000
```

8.2. Apply the Updated Manifest

Apply the updated deployment:
sh
Copy code
```
kubectl apply -f deployment.yaml
```



8.3. Verify the Update

Check the status of the deployment:
sh
Copy code
```
kubectl rollout status deployment/flask-app
```

9. Access the Updated Application

9.1. Access Through ClusterIP Service

Forward the port to access the ClusterIP service:

```
kubectl port-forward service/flask-service 8080:80
```

1. Open your browser and navigate to http://localhost:8080 to see the updated message.

9.2. Access Through NodePort Service

1. Access the application using the NodePort:

   ```
   curl http://<minikube-ip>:30001
   ```