Project 01

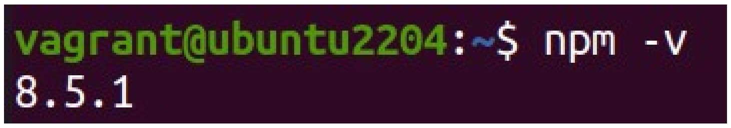
Deploying a Node.js App Using Minikube Kubernetes

Overview

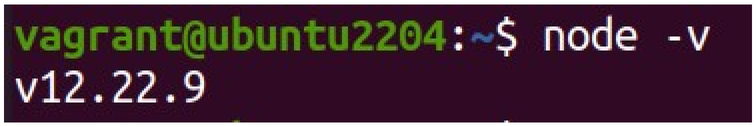
This project guides you through deploying a Node.js application using Minikube Kubernetes. You'll use Git for version control, explore branching and fast-forward merges, and set up Kubernetes services and deployment pods, including ClusterIP and NodePort service types.

Prerequisites

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



```
vagrant@ubuntu2204:~$ npm -v
8.5.1
```



```
vagrant@ubuntu2204:~$ node -v
v12.22.9
```

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
```

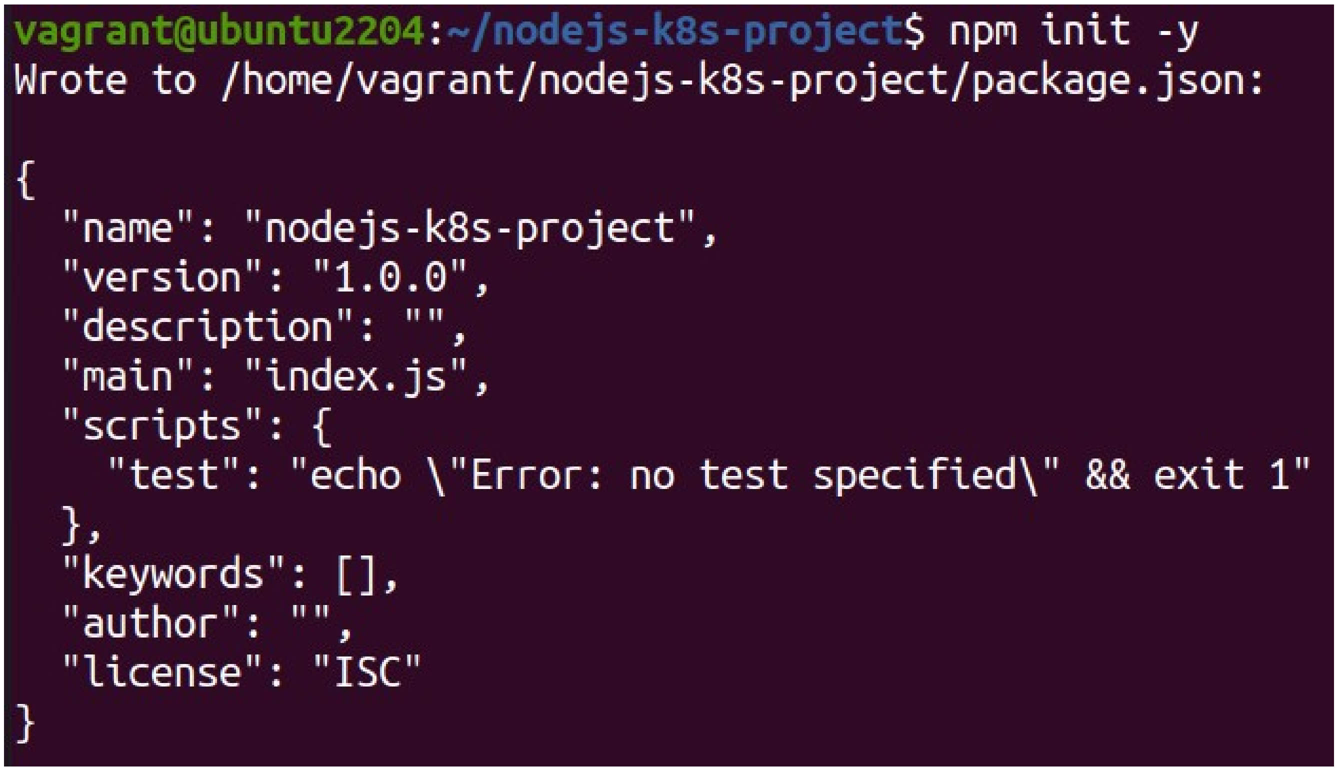
Initialize a Git repository:

```
git init
```

1.2. Create a Node.js Application

Initialize a Node.js project:

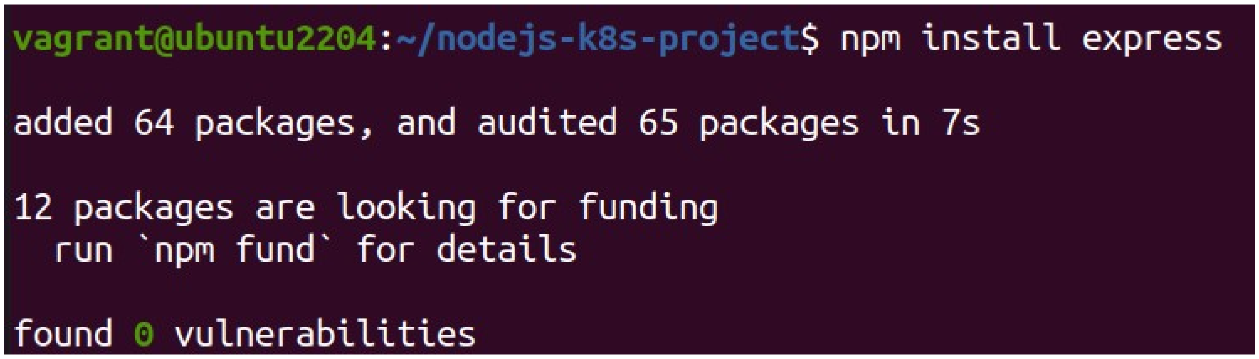
```
npm init -y
```



```
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",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}
```

Install Express.js:

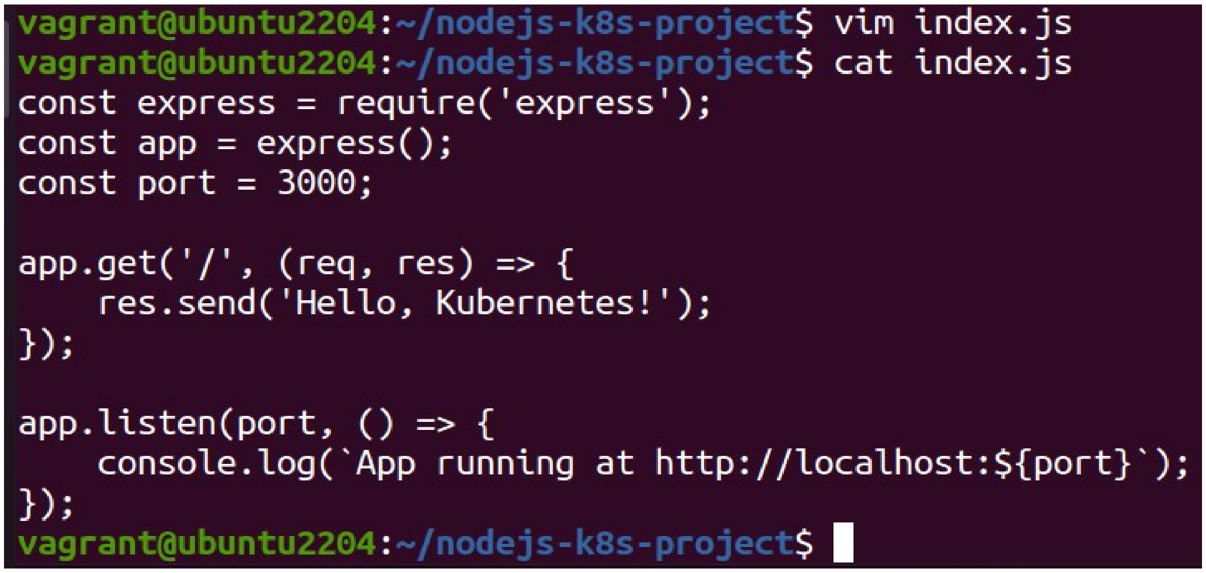
```
npm install express
```



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

added 64 packages, and audited 65 packages in 7s

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

found 0 vulnerabilities
```

Create an `index.js` file with the following content:

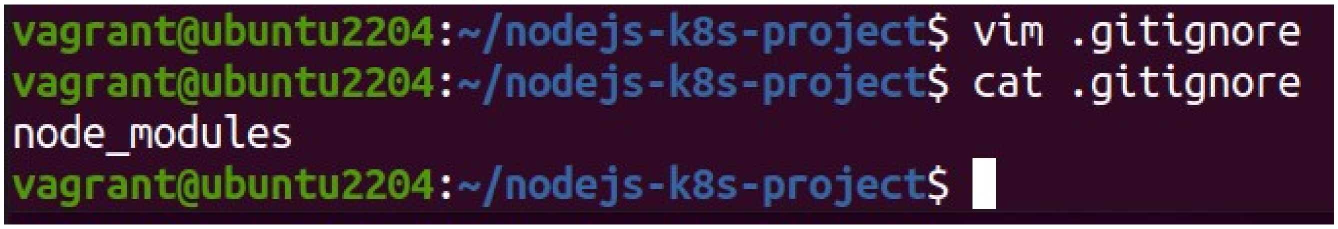
```
const express = require('express');

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}`);

});
```



```
vagrant@ubuntu2204:~/nodejs-k8s-project$ vim index.js
vagrant@ubuntu2204:~/nodejs-k8s-project$ cat index.js
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:${port}`);
});
vagrant@ubuntu2204:~/nodejs-k8s-project$
```

Create a .gitignore file to ignore node_modules:

```
node_modules
```

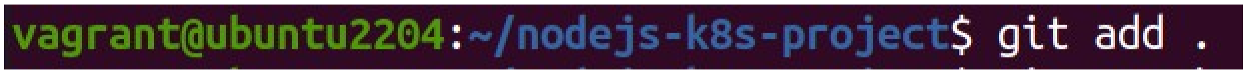


```
vagrant@ubuntu2204:~/nodejs-k8s-project$ vim .gitignore
vagrant@ubuntu2204:~/nodejs-k8s-project$ cat .gitignore
node_modules
vagrant@ubuntu2204:~/nodejs-k8s-project$
```

1.3. Commit the Initial Code

Add files to Git:

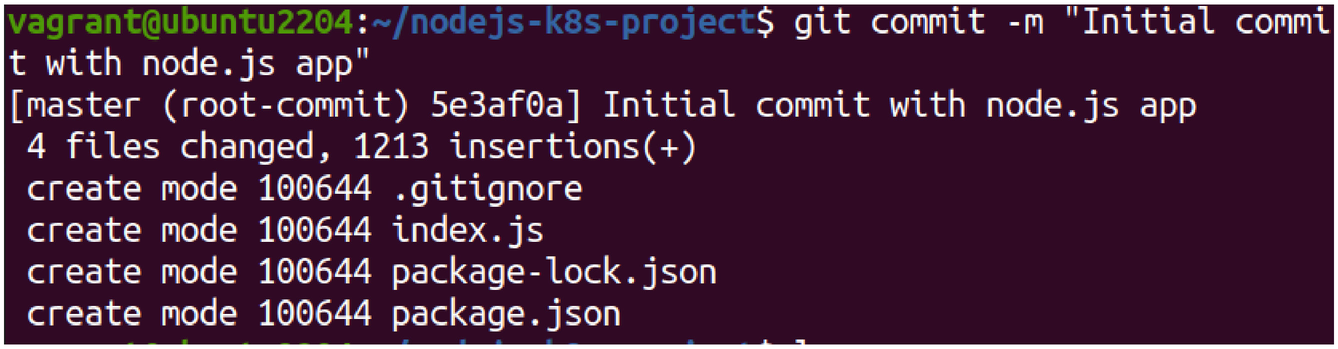
```
git add .
```



```
vagrant@ubuntu2204:~/nodejs-k8s-project$ git add .
```

Commit the changes:

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



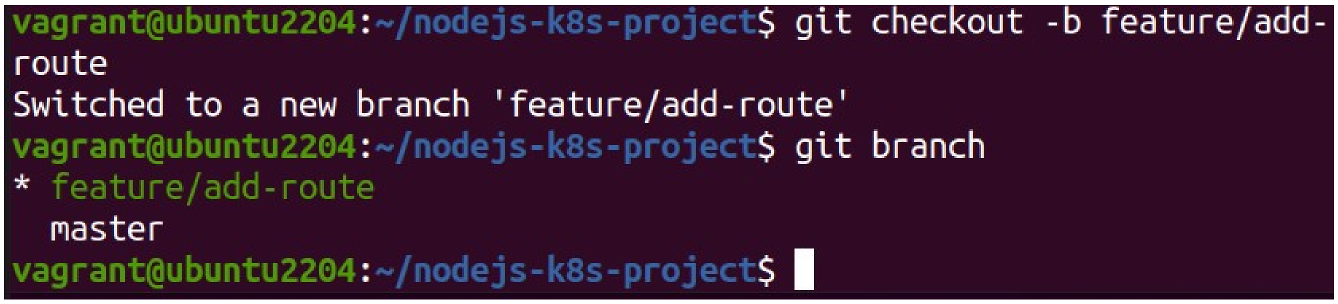
```
vagrant@ubuntu2204:~/nodejs-k8s-project$ git commit -m "Initial commi
t with node.js app"
[master (root-commit) 5e3af0a] Initial commit with node.js app
 4 files changed, 1213 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 index.js
 create mode 100644 package-lock.json
 create mode 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
```

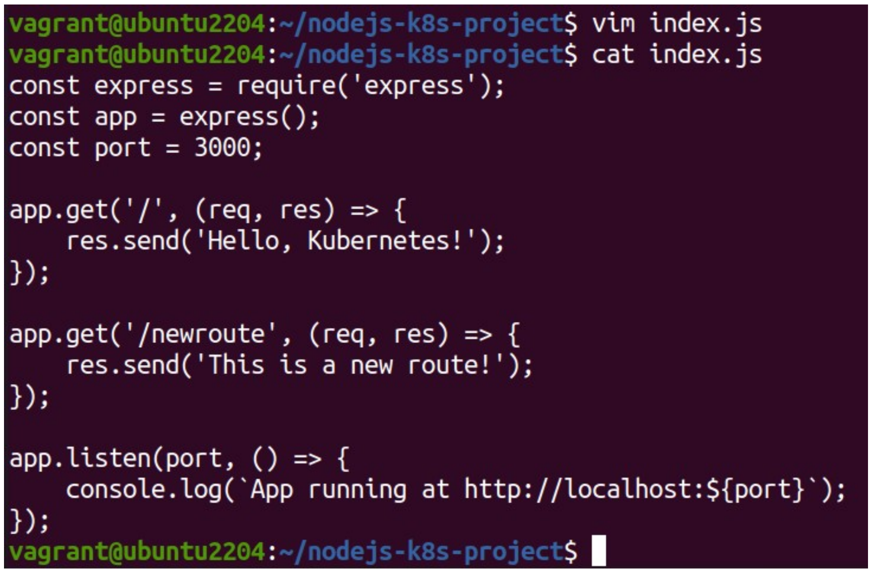


```
vagrant@ubuntu2204:~/nodejs-k8s-project$ git checkout -b feature/add-
route
Switched to a new branch 'feature/add-route'
vagrant@ubuntu2204:~/nodejs-k8s-project$ git branch
* feature/add-route
  master
vagrant@ubuntu2204:~/nodejs-k8s-project$
```

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!');

});
```

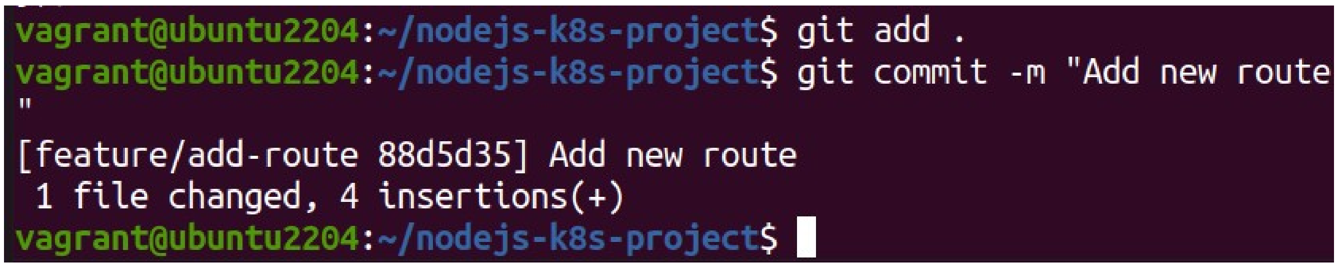


```
vagrant@ubuntu2204:~/nodejs-k8s-project$ vim index.js
vagrant@ubuntu2204:~/nodejs-k8s-project$ cat index.js
const express = require('express');
const app = express();
const port = 3000;

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

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

app.listen(port, () => {
    console.log(`App running at http://localhost:${port}`);
});
vagrant@ubuntu2204:~/nodejs-k8s-project$
```

Commit the changes:

git add .

git commit -m "Add new route"

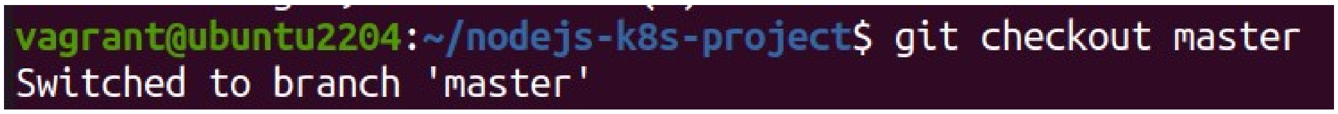


```
vagrant@ubuntu2204:~/nodejs-k8s-project$ git add .
vagrant@ubuntu2204:~/nodejs-k8s-project$ git commit -m "Add new route
"
[feature/add-route 88d5d35] Add new route
 1 file changed, 4 insertions(+)
vagrant@ubuntu2204:~/nodejs-k8s-project$
```

2.3. Merge the Branch Using Fast-Forward

Switch back to the main branch:

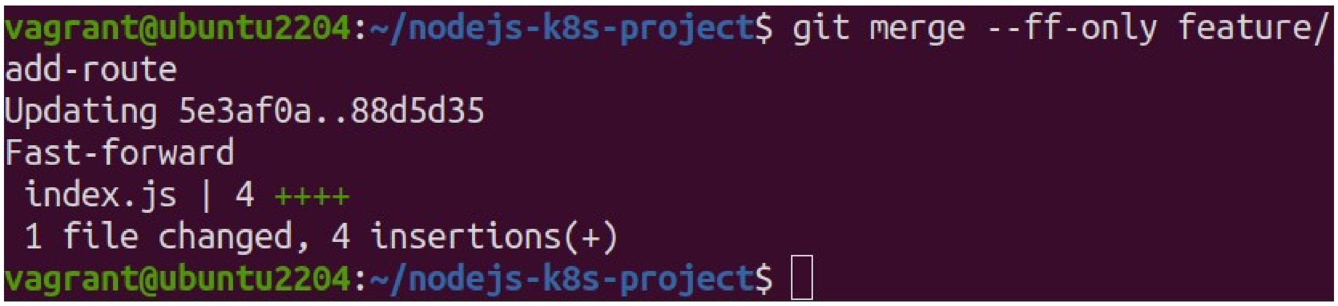
git checkout main



```
vagrant@ubuntu2204:~/nodejs-k8s-project$ git checkout master
Switched to branch 'master'
```

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

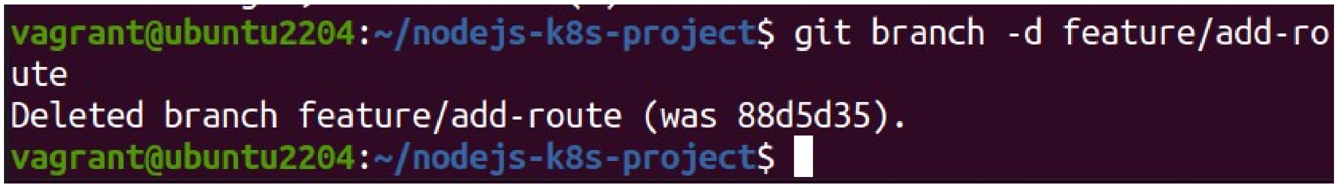
git merge --ff-only feature/add-route



```
vagrant@ubuntu2204:~/nodejs-k8s-project$ git merge --ff-only feature/
add-route
Updating 5e3af0a..88d5d35
Fast-forward
 index.js | 4 ++++
 1 file changed, 4 insertions(+)
vagrant@ubuntu2204:~/nodejs-k8s-project$
```

Delete the feature branch:

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



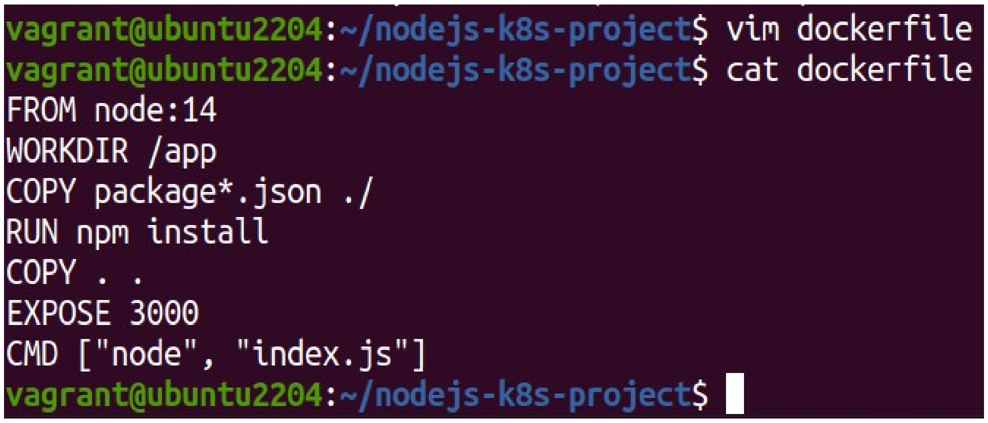
```
vagrant@ubuntu2204:~/nodejs-k8s-project$ git branch -d feature/add-ro
ute
Deleted branch feature/add-route (was 88d5d35).
vagrant@ubuntu2204:~/nodejs-k8s-project$ 
```

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"]
```

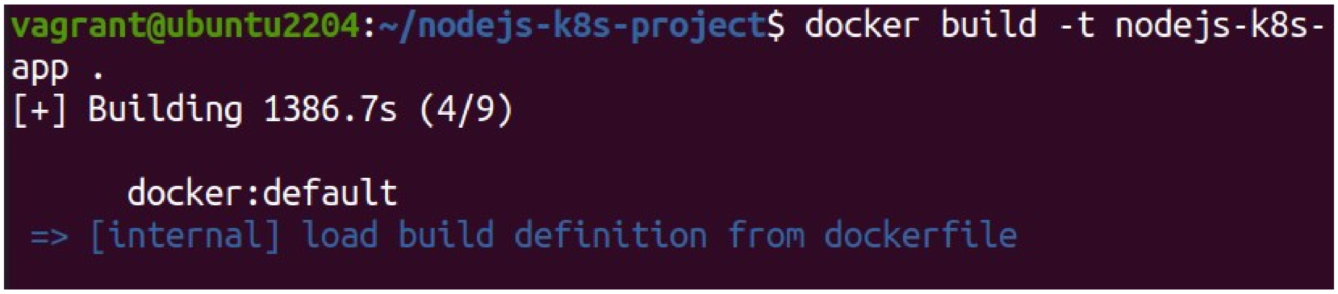


```
vagrant@ubuntu2204:~/nodejs-k8s-project$ vim dockerfile
vagrant@ubuntu2204:~/nodejs-k8s-project$ cat dockerfile
FROM node:14
WORKDIR /app
COPY package*.json ./
RUN npm install
COPY . .
EXPOSE 3000
CMD ["node", "index.js"]
vagrant@ubuntu2204:~/nodejs-k8s-project$ 
```

3.2. Build and Test the Docker Image

Build the Docker image:

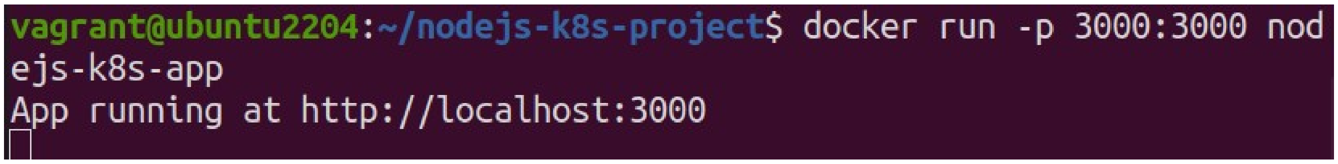
```
docker build -t nodejs-k8s-app .
```



```
vagrant@ubuntu2204:~/nodejs-k8s-project$ docker build -t nodejs-k8s-
app .
[+] Building 1386.7s (4/9)

      docker:default
 => [internal] load build definition from dockerfile
```

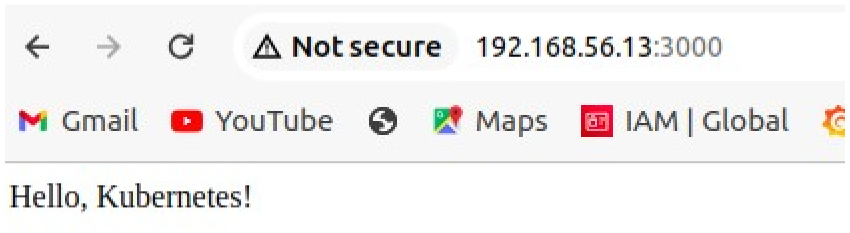
Run the Docker container to test:

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



```
vagrant@ubuntu2204:~/nodejs-k8s-project$ docker run -p 3000:3000 nod
ejs-k8s-app
App running at http://localhost:3000
```

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



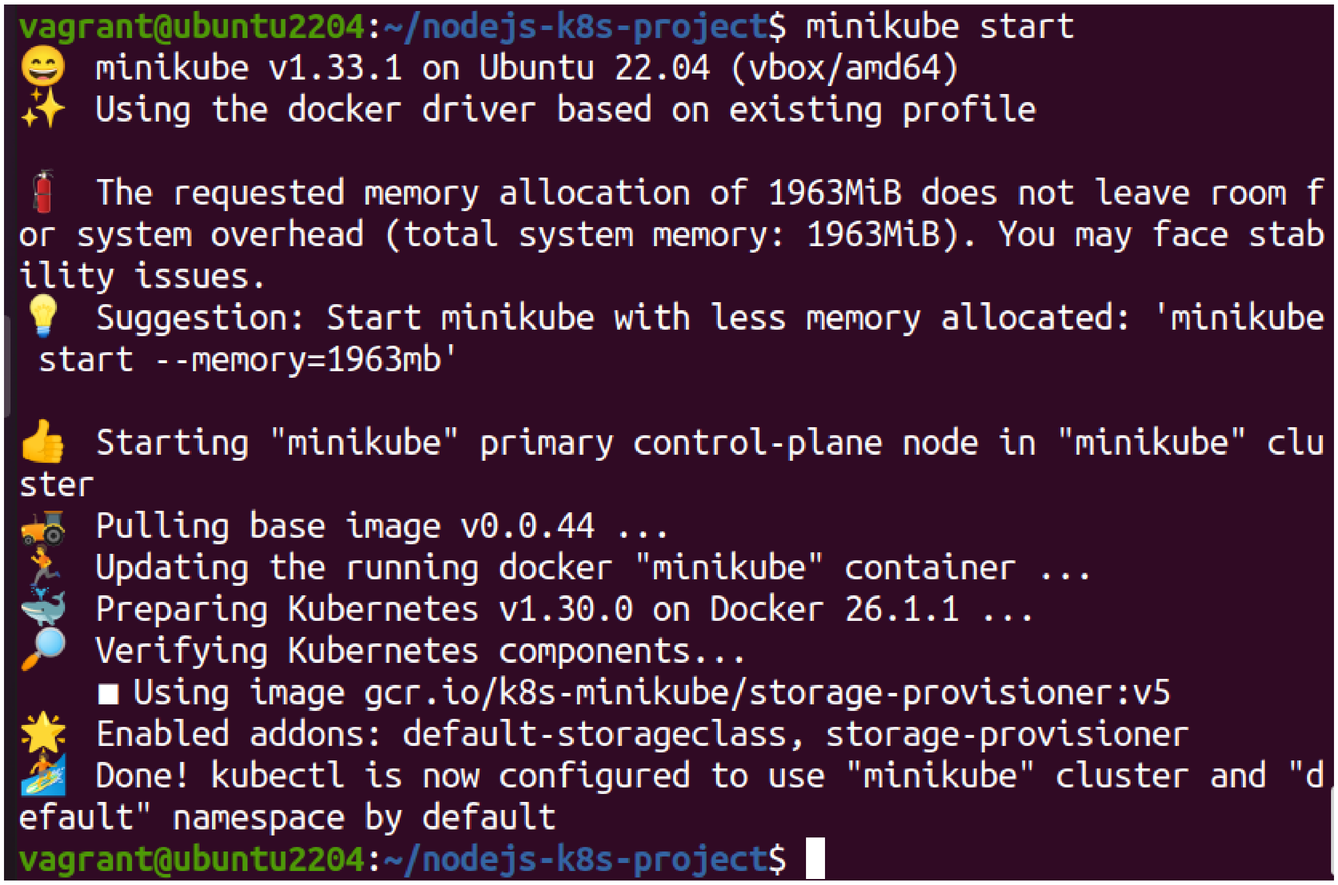
Hello, Kubernetes!

4. Deploying to Minikube Kubernetes

4.1. Start Minikube

Start Minikube:

minikube start



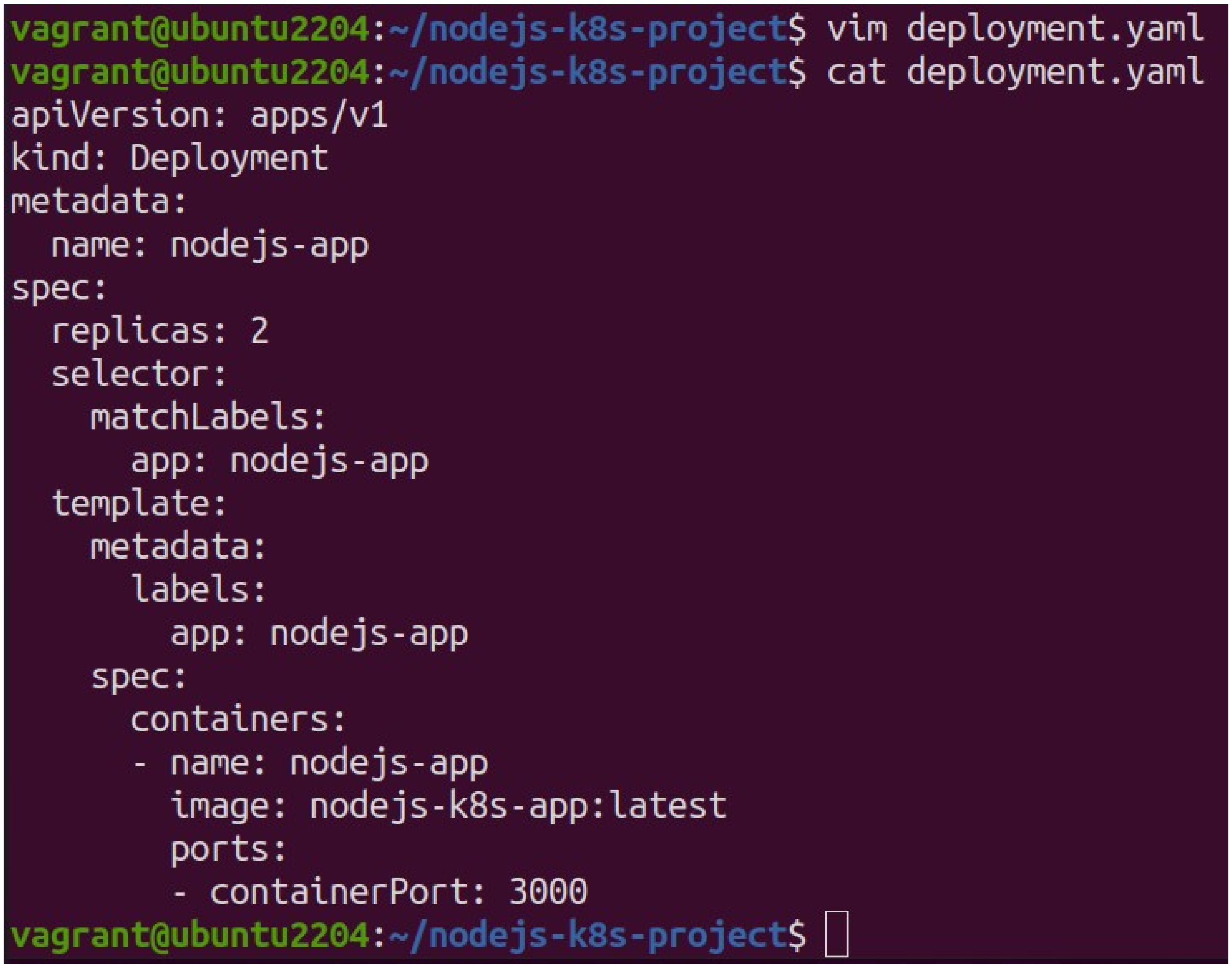
```
vagrant@ubuntu2204:~/nodejs-k8s-project$ minikube start
😄  minikube v1.33.1 on Ubuntu 22.04 (vbox/amd64)
✨  Using the docker driver based on existing profile

🧯  The requested memory allocation of 1963MiB does not leave room f
or system overhead (total system memory: 1963MiB). You may face stab
ility issues.
💡  Suggestion: Start minikube with less memory allocated: 'minikube
 start --memory=1963mb'

👍  Starting "minikube" primary control-plane node in "minikube" clu
ster
🚜  Pulling base image v0.0.44 ...
🏃  Updating the running docker "minikube" container ...
🐳  Preparing Kubernetes v1.30.0 on Docker 26.1.1 ...
🔎  Verifying Kubernetes components...
    ■ Using image gcr.io/k8s-minikube/storage-provisioner:v5
🌟  Enabled addons: default-storageclass, storage-provisioner
🏄  Done! kubectl is now configured to use "minikube" cluster and "d
efault" namespace by default
vagrant@ubuntu2204:~/nodejs-k8s-project$
```

4.2. Create Kubernetes Deployment and Service Manifests

Create a deployment.yaml file:

```
apiVersion: apps/v1

kind: Deployment

metadata:
```

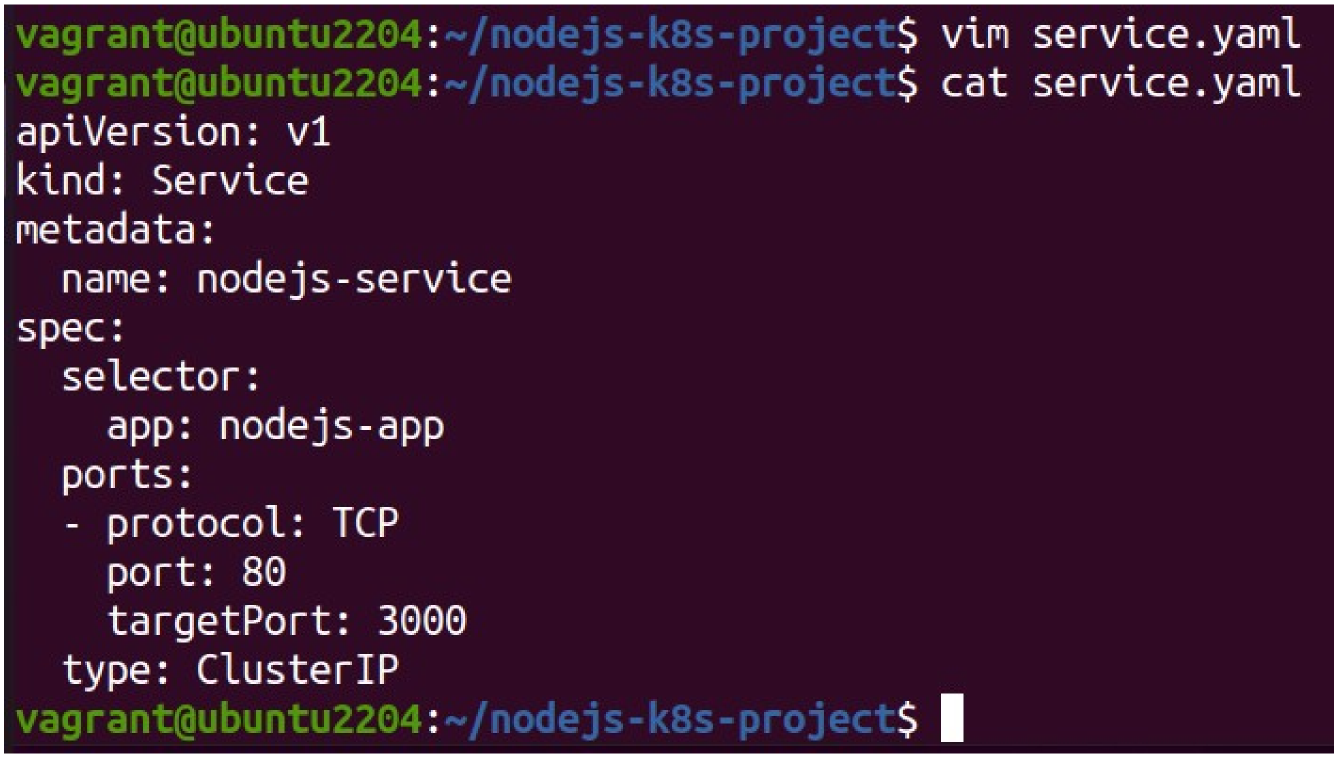
```yaml
  name: nodejs-app
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nodejs-app
  template:
    metadata:
      labels:
        app: nodejs-app
    spec:
      containers:
      - name: nodejs-app
        image: nodejs-k8s-app:latest
        ports:
        - containerPort: 3000
```



```
vagrant@ubuntu2204:~/nodejs-k8s-project$ vim deployment.yaml
vagrant@ubuntu2204:~/nodejs-k8s-project$ cat deployment.yaml
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
vagrant@ubuntu2204:~/nodejs-k8s-project$
```

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
```

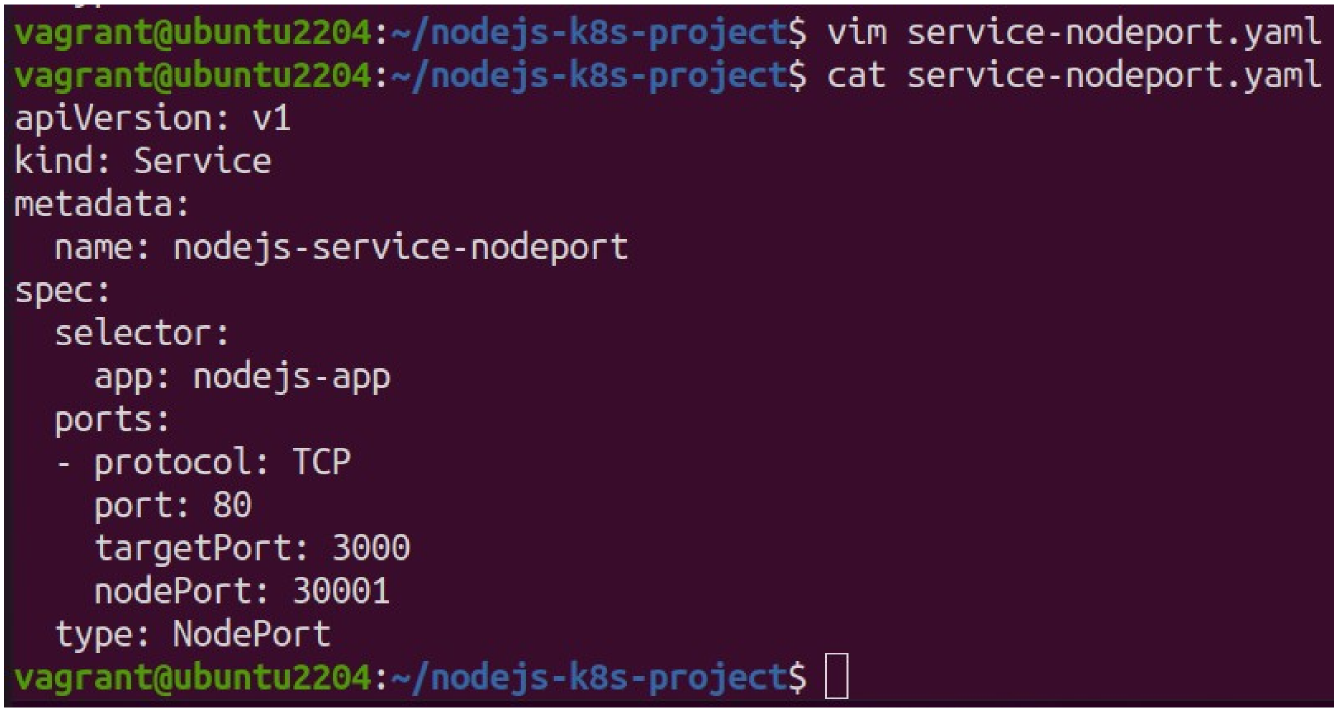


```
vagrant@ubuntu2204:~/nodejs-k8s-project$ vim service.yaml
vagrant@ubuntu2204:~/nodejs-k8s-project$ cat service.yaml
apiVersion: v1
kind: Service
metadata:
  name: nodejs-service
spec:
  selector:
    app: nodejs-app
  ports:
  - protocol: TCP
    port: 80
    targetPort: 3000
  type: ClusterIP
vagrant@ubuntu2204:~/nodejs-k8s-project$
```

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
```

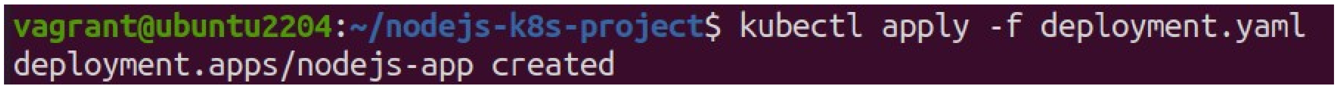


```
vagrant@ubuntu2204:~/nodejs-k8s-project$ vim service-nodeport.yaml
vagrant@ubuntu2204:~/nodejs-k8s-project$ cat service-nodeport.yaml
apiVersion: v1
kind: Service
metadata:
  name: nodejs-service-nodeport
spec:
  selector:
    app: nodejs-app
  ports:
  - protocol: TCP
    port: 80
    targetPort: 3000
    nodePort: 30001
  type: NodePort
vagrant@ubuntu2204:~/nodejs-k8s-project$
```

4.3. Apply Manifests to Minikube

Apply the deployment:

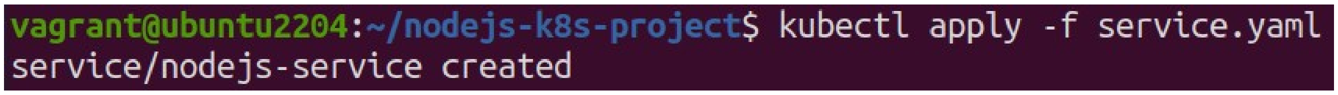
```
kubectl apply -f deployment.yaml
```



```
vagrant@ubuntu2204:~/nodejs-k8s-project$ kubectl apply -f deployment.yaml
deployment.apps/nodejs-app created
```

Apply the ClusterIP service:

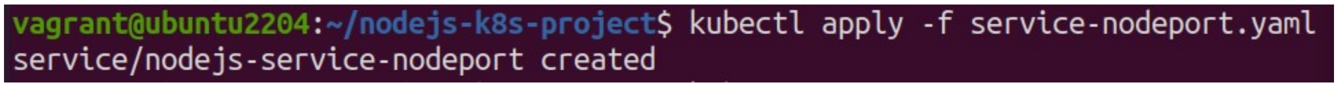
```
kubectl apply -f service.yaml
```



```
vagrant@ubuntu2204:~/nodejs-k8s-project$ kubectl apply -f service.yaml
service/nodejs-service created
```

Apply the NodePort service:

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

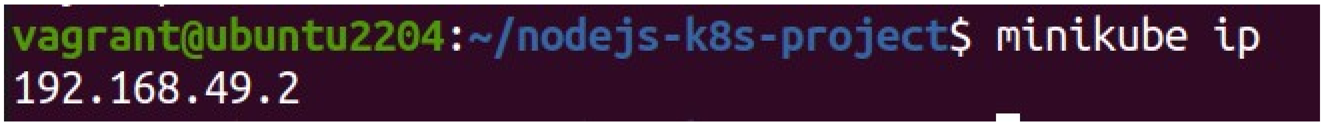


```
vagrant@ubuntu2204:~/nodejs-k8s-project$ kubectl apply -f service-nodeport.yaml
service/nodejs-service-nodeport created
```

4.4. Access the Application

Get the Minikube IP:

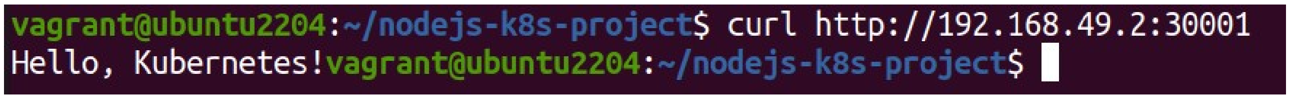
```
minikube ip
```



```
vagrant@ubuntu2204:~/nodejs-k8s-project$ minikube ip
192.168.49.2
```

1 Access the application using the NodePort:

```
curl http://192.168.49.2:30001
```



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

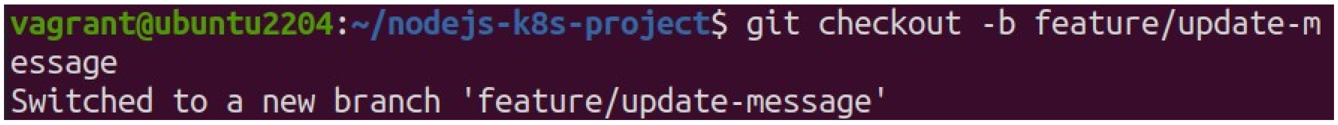
Making Changes to the App and Redeploying Using Kubernetes

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
```

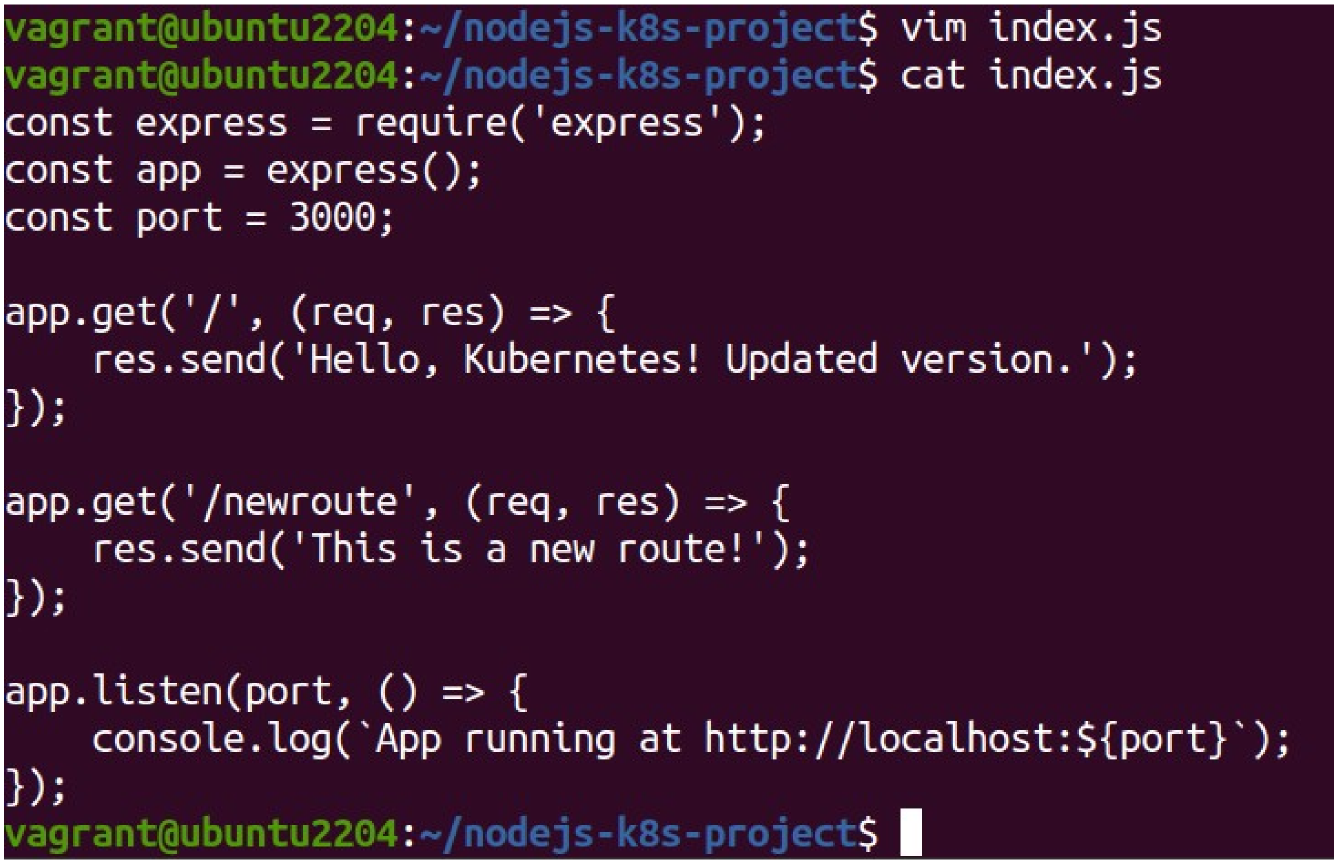


```
vagrant@ubuntu2204:~/nodejs-k8s-project$ git checkout -b feature/update-m
essage
Switched to a new branch 'feature/update-message'
```

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}`);

});
```

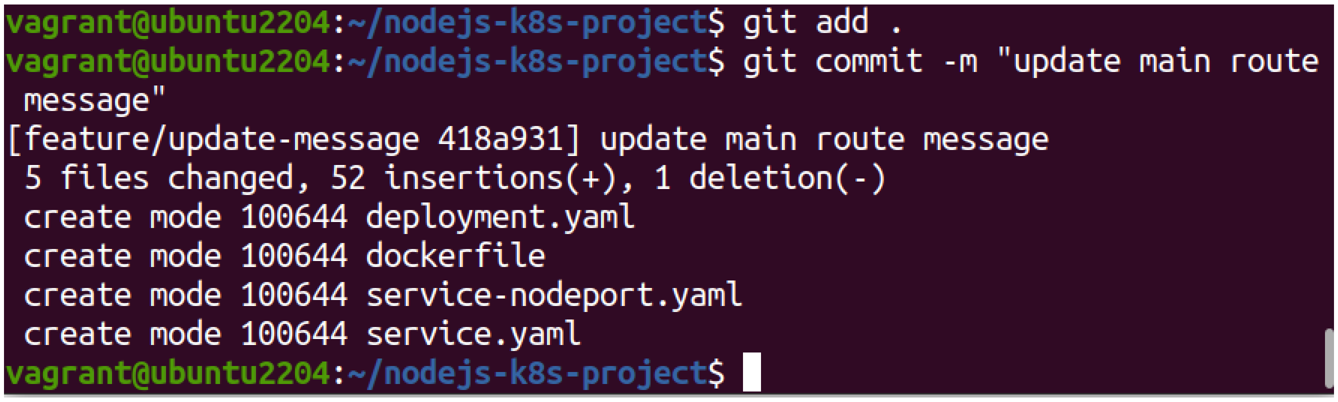


```
vagrant@ubuntu2204:~/nodejs-k8s-project$ vim index.js
vagrant@ubuntu2204:~/nodejs-k8s-project$ cat index.js
const express = require('express');
const app = express();
const port = 3000;

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:${port}`);
});
vagrant@ubuntu2204:~/nodejs-k8s-project$
```

6.3. Commit the Changes

Add and commit the changes:

```
git add .

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



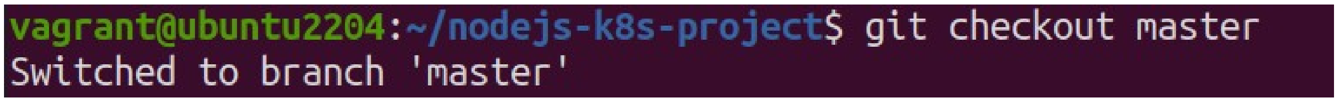
```
vagrant@ubuntu2204:~/nodejs-k8s-project$ git add .
vagrant@ubuntu2204:~/nodejs-k8s-project$ git commit -m "update main route
 message"
[feature/update-message 418a931] update main route message
 5 files changed, 52 insertions(+), 1 deletion(-)
 create mode 100644 deployment.yaml
 create mode 100644 dockerfile
 create mode 100644 service-nodeport.yaml
 create mode 100644 service.yaml
vagrant@ubuntu2204:~/nodejs-k8s-project$
```

7. Merge the Changes and Rebuild the Docker Image

7.1. Merge the Feature Branch

Switch back to the main branch:

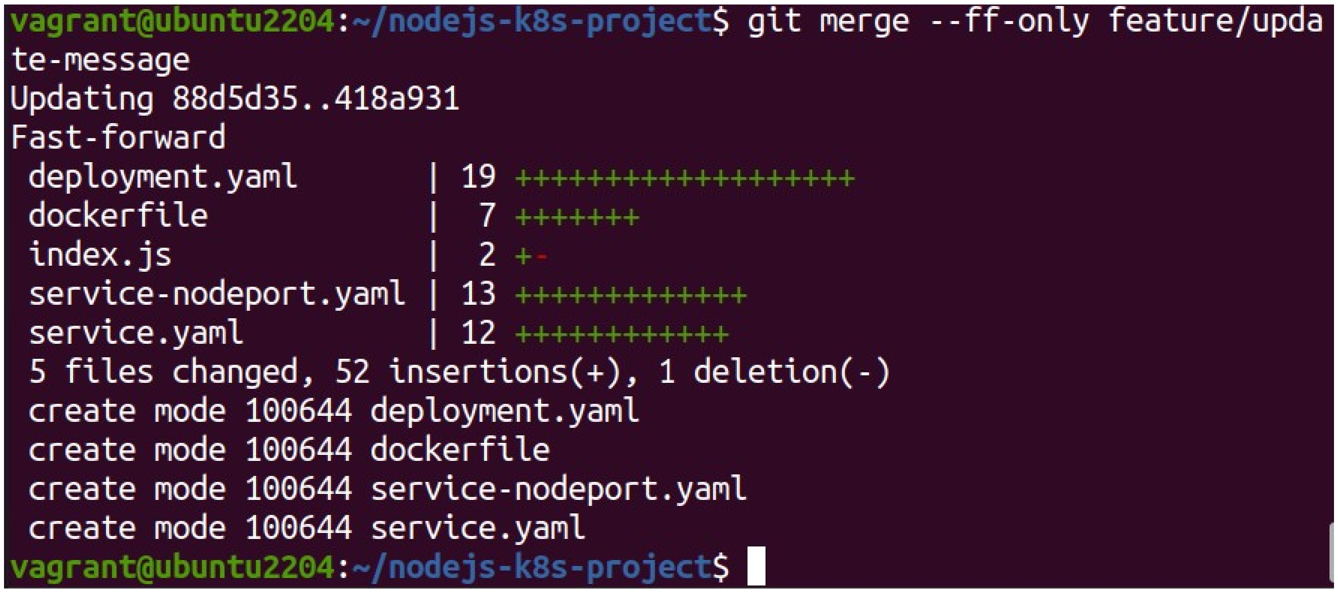
git checkout master



```
vagrant@ubuntu2204:~/nodejs-k8s-project$ git checkout master
Switched to branch 'master'
```

Merge the feature/update-message branch:

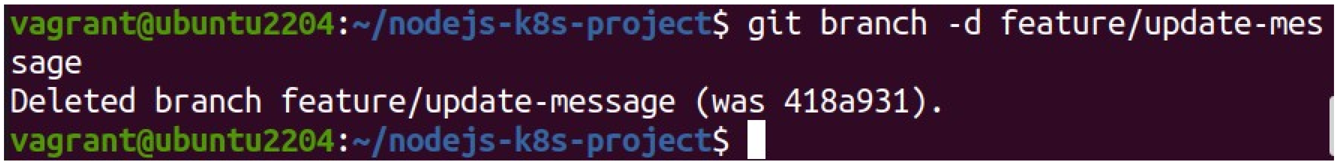
git merge --ff-only feature/update-message



```
vagrant@ubuntu2204:~/nodejs-k8s-project$ git merge --ff-only feature/upda
te-message
Updating 88d5d35..418a931
Fast-forward
 deployment.yaml       | 19 +++++++++++++++++++
 dockerfile            |  7 +++++++
 index.js              |  2 +-
 service-nodeport.yaml | 13 +++++++++++++
 service.yaml          | 12 ++++++++++++
 5 files changed, 52 insertions(+), 1 deletion(-)
 create mode 100644 deployment.yaml
 create mode 100644 dockerfile
 create mode 100644 service-nodeport.yaml
 create mode 100644 service.yaml
vagrant@ubuntu2204:~/nodejs-k8s-project$
```

Delete the feature branch:

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

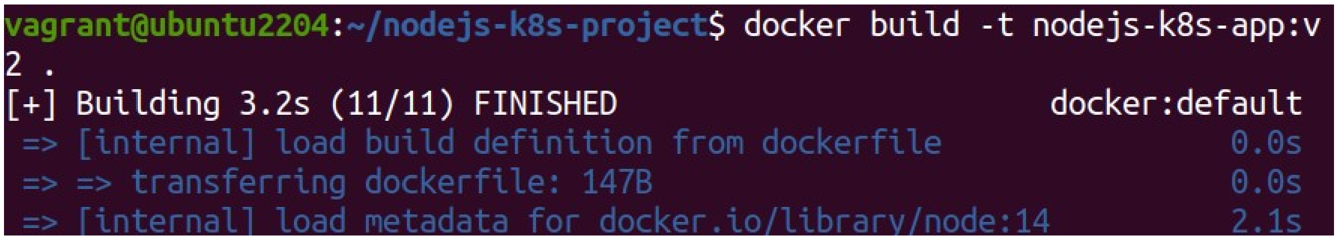


```
vagrant@ubuntu2204:~/nodejs-k8s-project$ git branch -d feature/update-mes
sage
Deleted branch feature/update-message (was 418a931).
vagrant@ubuntu2204:~/nodejs-k8s-project$
```

7.2. Rebuild the Docker Image

Rebuild the Docker image with a new tag:

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



```
vagrant@ubuntu2204:~/nodejs-k8s-project$ docker build -t nodejs-k8s-app:v
2 .
[+] Building 3.2s (11/11) FINISHED                          docker:default
 => [internal] load build definition from dockerfile                  0.0s
 => => transferring dockerfile: 147B                                  0.0s
 => [internal] load metadata for docker.io/library/node:14            2.1s
```

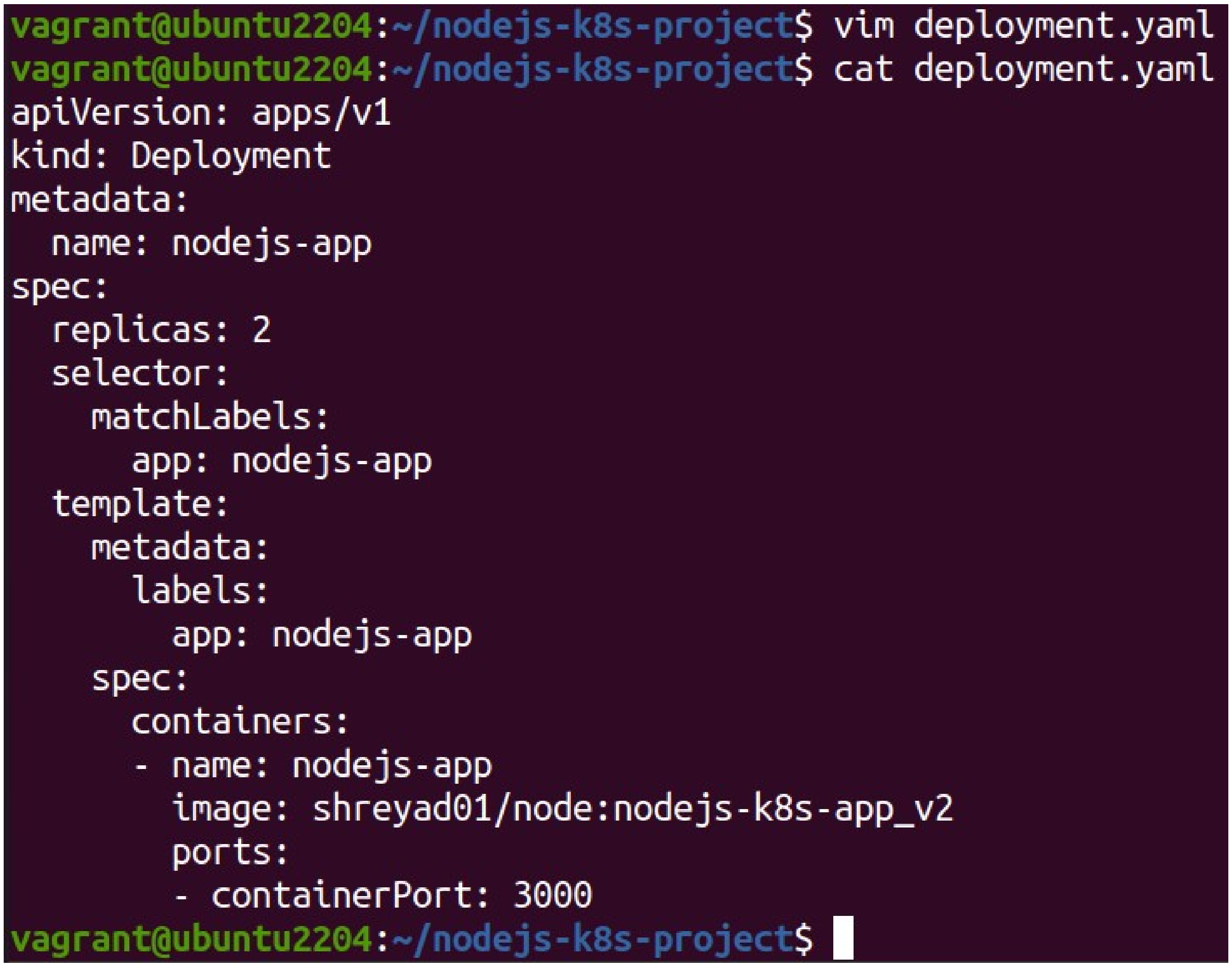
8. Update Kubernetes Deployment

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
```

```yaml
selector:
  matchLabels:
    app: nodejs-app
template:
  metadata:
    labels:
      app: nodejs-app
  spec:
    containers:
    - name: nodejs-app
      image: shreyad01/node:nodejs-k8s-app_v2
      ports:
      - containerPort: 3000
```

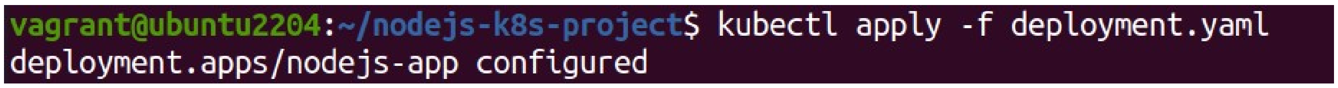


```
vagrant@ubuntu2204:~/nodejs-k8s-project$ vim deployment.yaml
vagrant@ubuntu2204:~/nodejs-k8s-project$ cat deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nodejs-app
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nodejs-app
  template:
    metadata:
      labels:
        app: nodejs-app
    spec:
      containers:
      - name: nodejs-app
        image: shreyad01/node:nodejs-k8s-app_v2
        ports:
        - containerPort: 3000
vagrant@ubuntu2204:~/nodejs-k8s-project$
```

8.2. Apply the Updated Manifest

Apply the updated deployment:

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

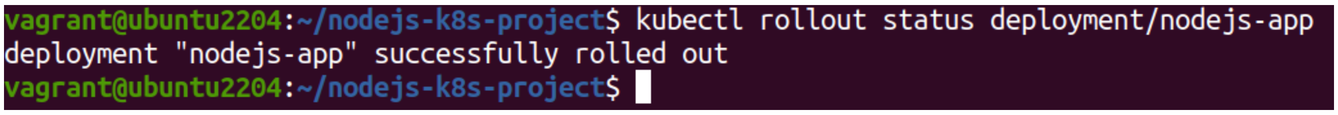


```
vagrant@ubuntu2204:~/nodejs-k8s-project$ kubectl apply -f deployment.yaml
deployment.apps/nodejs-app configured
```

8.3. Verify the Update

Check the status of the deployment:

kubectl rollout status deployment/nodejs-app



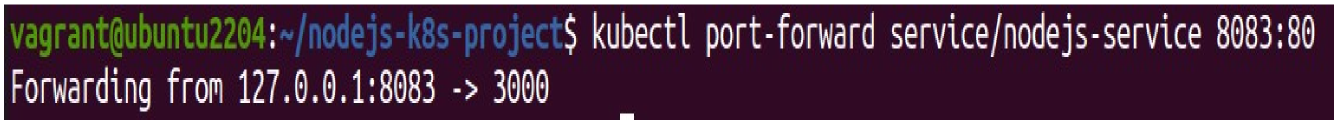
```
vagrant@ubuntu2204:~/nodejs-k8s-project$ kubectl rollout status deployment/nodejs-app
deployment "nodejs-app" successfully rolled out
vagrant@ubuntu2204:~/nodejs-k8s-project$
```

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 8083:80



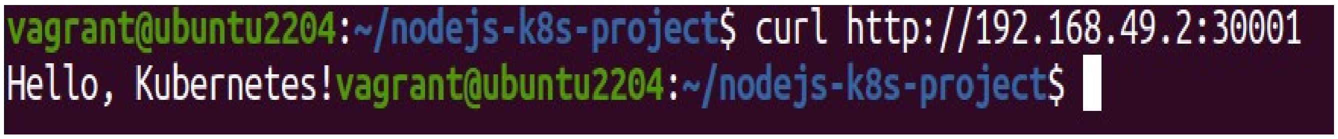
```
vagrant@ubuntu2204:~/nodejs-k8s-project$ kubectl port-forward service/nodejs-service 8083:80
Forwarding from 127.0.0.1:8083 -> 3000
```

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://192.168.49.2:30001



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

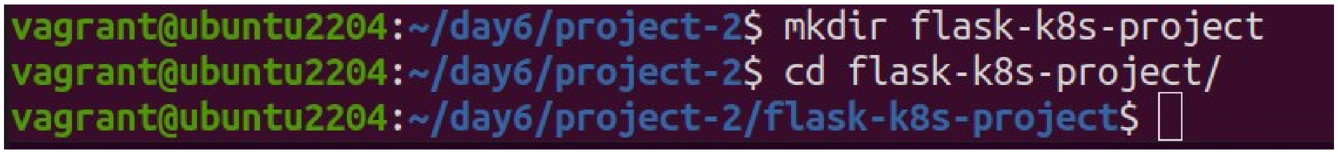
1. Set Up Git Version Control

1.1. Initialize a Git Repository

Create a new directory for your project:

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

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



```
vagrant@ubuntu2204:~/day6/project-2$ mkdir flask-k8s-project
vagrant@ubuntu2204:~/day6/project-2$ cd flask-k8s-project/
vagrant@ubuntu2204:~/day6/project-2/flask-k8s-project$
```

Initialize a Git repository:
sh
Copy code

```
git init
```

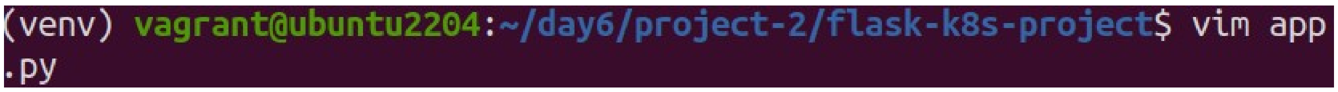
1.2. Create a Python Flask Application

Create a virtual environment:

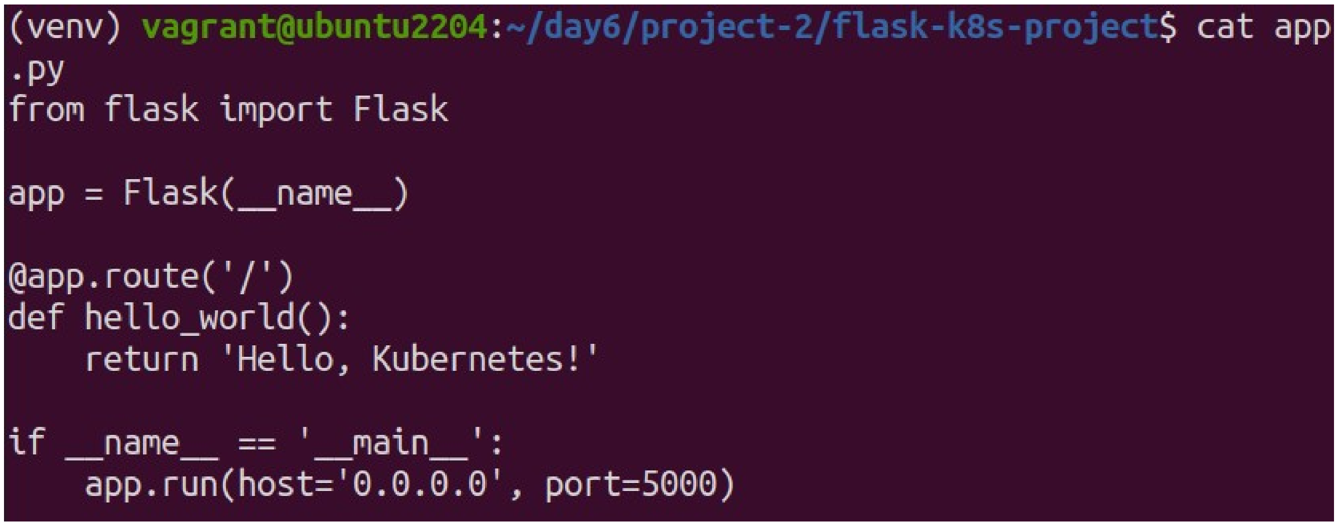
```
python -m venv venv

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)
```

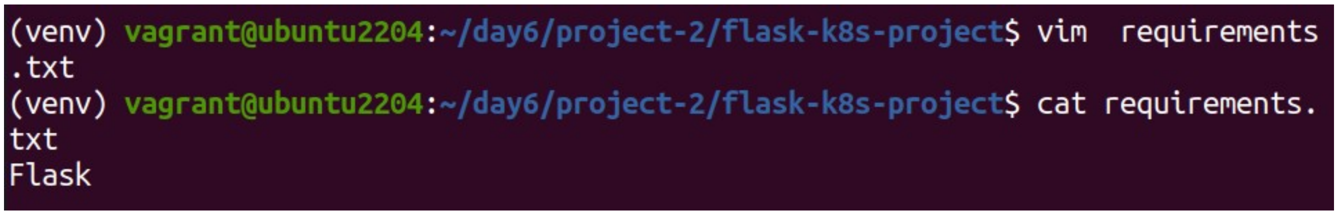


```
(venv) vagrant@ubuntu2204:~/day6/project-2/flask-k8s-project$ vim app
.py
```



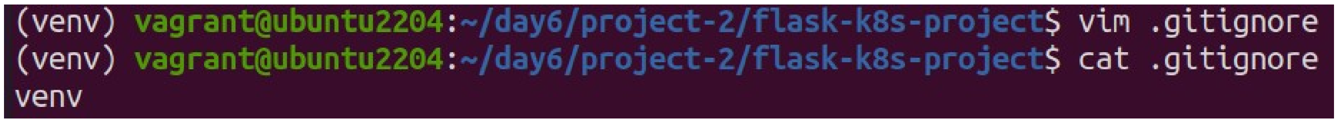
```
(venv) vagrant@ubuntu2204:~/day6/project-2/flask-k8s-project$ cat app
.py
from flask import Flask

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



```
(venv) vagrant@ubuntu2204:~/day6/project-2/flask-k8s-project$ vim  requirements
.txt
(venv) vagrant@ubuntu2204:~/day6/project-2/flask-k8s-project$ cat requirements.
txt
Flask
```

Create a .gitignore file to ignore venv:
Copy code
venv



```
(venv) vagrant@ubuntu2204:~/day6/project-2/flask-k8s-project$ vim .gitignore
(venv) vagrant@ubuntu2204:~/day6/project-2/flask-k8s-project$ cat .gitignore
venv
```

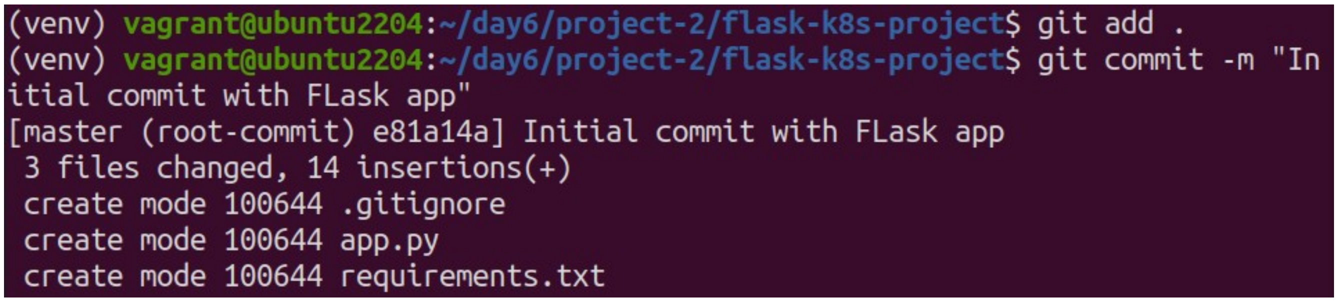
1.3. Commit the Initial Code

Add files to Git:

git add .

Commit the changes:

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



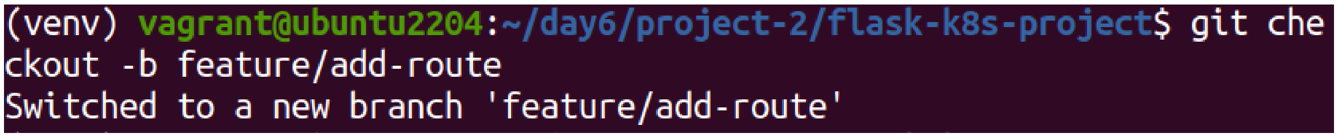
```
(venv) vagrant@ubuntu2204:~/day6/project-2/flask-k8s-project$ git add .
(venv) vagrant@ubuntu2204:~/day6/project-2/flask-k8s-project$ git commit -m "In
itial commit with FLask app"
[master (root-commit) e81a14a] Initial commit with FLask app
 3 files changed, 14 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 app.py
 create mode 100644 requirements.txt
```

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
```

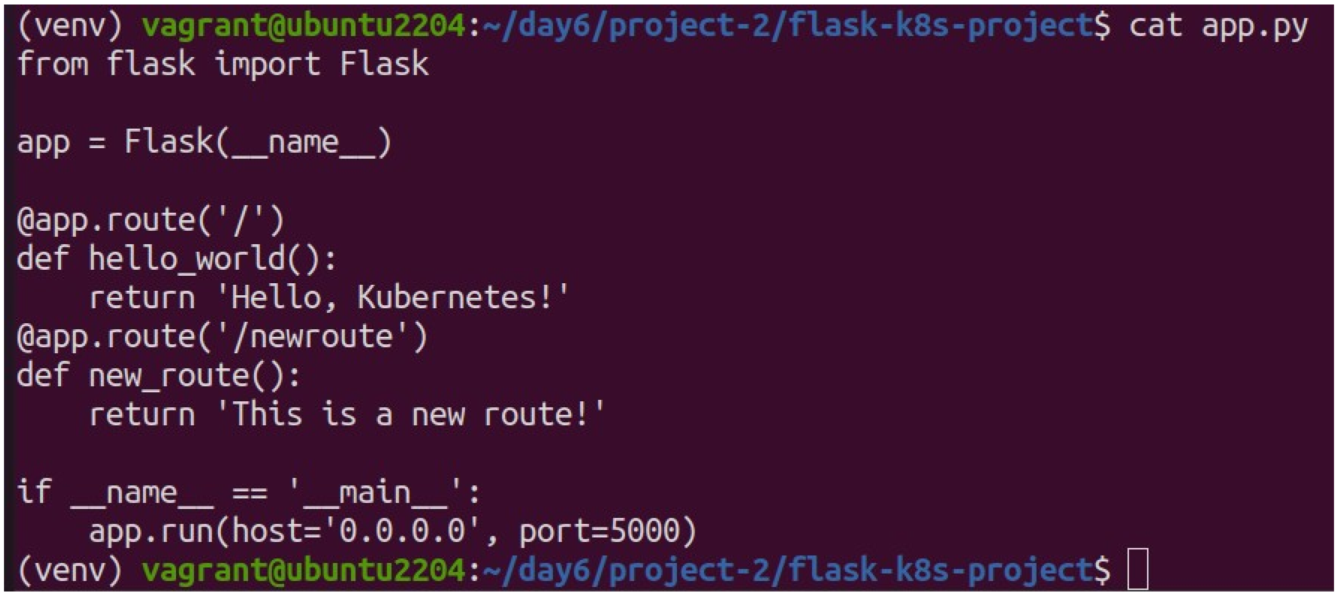


```
(venv) vagrant@ubuntu2204:~/day6/project-2/flask-k8s-project$ git che
ckout -b feature/add-route
Switched to a new branch 'feature/add-route'
```

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!'
```

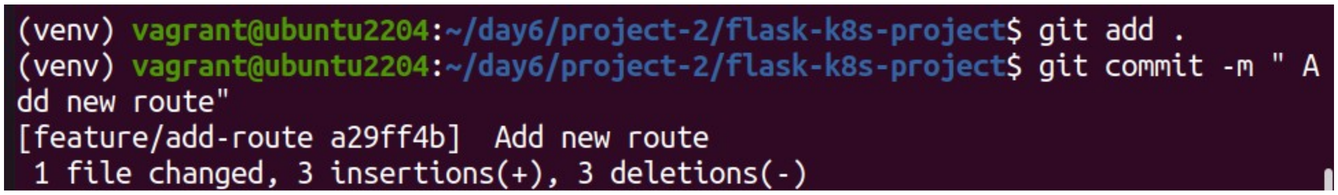


```
(venv) vagrant@ubuntu2204:~/day6/project-2/flask-k8s-project$ cat app.py
from flask import Flask

app = Flask(__name__)

@app.route('/')
def hello_world():
    return 'Hello, Kubernetes!'
@app.route('/newroute')
def new_route():
    return 'This is a new route!'

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)
(venv) vagrant@ubuntu2204:~/day6/project-2/flask-k8s-project$ 
```

Commit the changes:

git add .

git commit -m "Add new route"

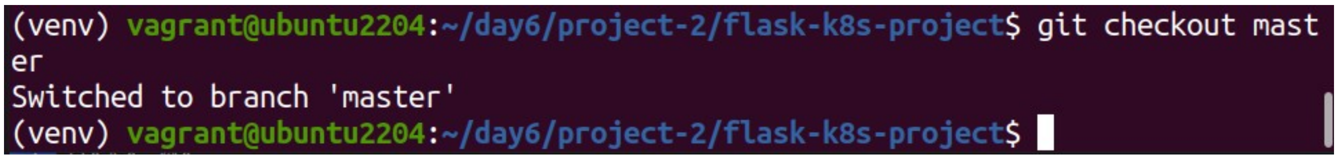


```
(venv) vagrant@ubuntu2204:~/day6/project-2/flask-k8s-project$ git add .
(venv) vagrant@ubuntu2204:~/day6/project-2/flask-k8s-project$ git commit -m " A
dd new route"
[feature/add-route a29ff4b]  Add new route
 1 file changed, 3 insertions(+), 3 deletions(-)
```

2.3. Merge the Branch Using Fast-Forward

Switch back to the main branch:

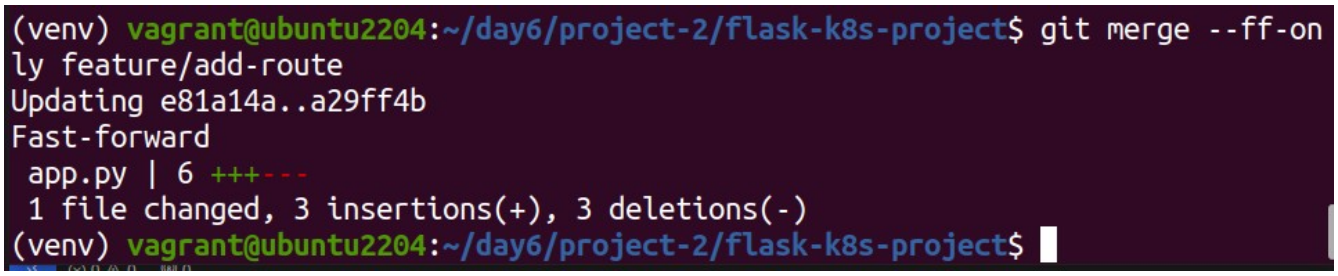
git checkout master



```
(venv) vagrant@ubuntu2204:~/day6/project-2/flask-k8s-project$ git checkout mast
er
Switched to branch 'master'
(venv) vagrant@ubuntu2204:~/day6/project-2/flask-k8s-project$ 
```

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

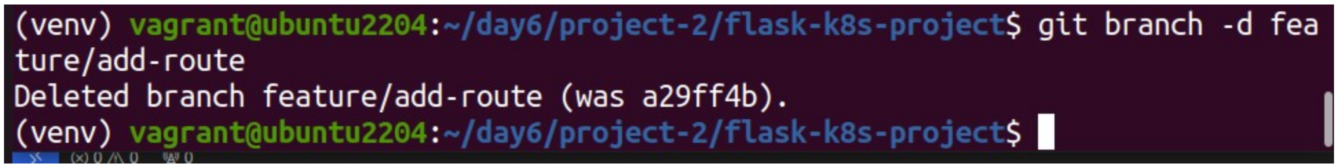
git merge --ff-only feature/add-route



```
(venv) vagrant@ubuntu2204:~/day6/project-2/flask-k8s-project$ git merge --ff-on
ly feature/add-route
Updating e81a14a..a29ff4b
Fast-forward
 app.py | 6 +++---
 1 file changed, 3 insertions(+), 3 deletions(-)
(venv) vagrant@ubuntu2204:~/day6/project-2/flask-k8s-project$
```

Delete the feature branch:

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



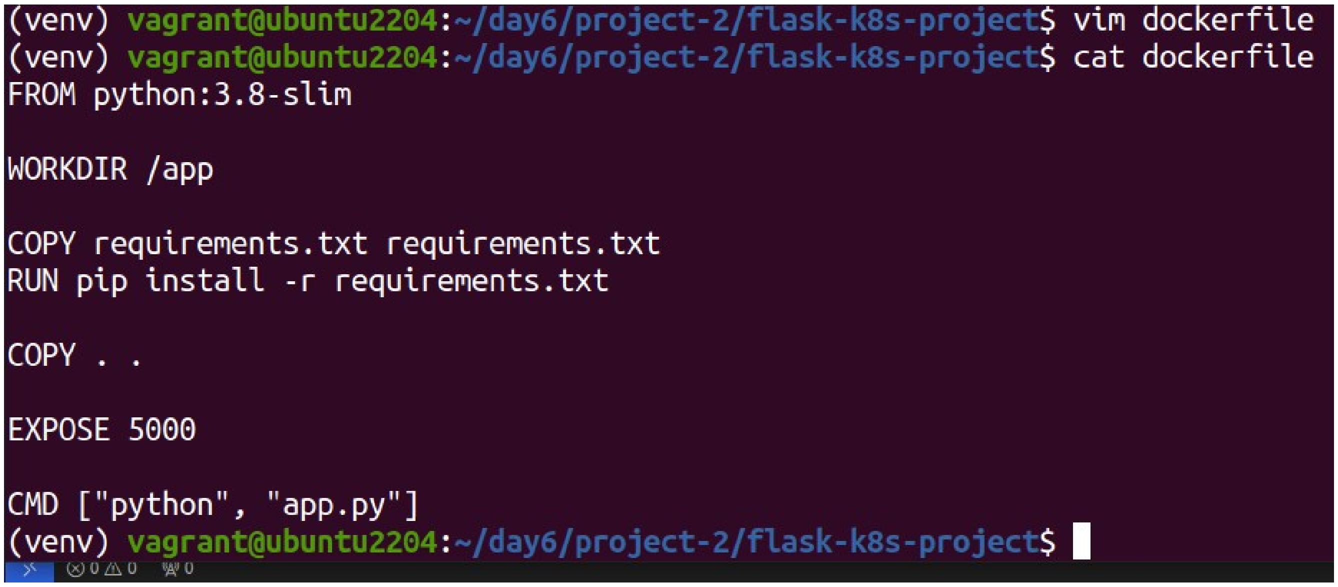
```
(venv) vagrant@ubuntu2204:~/day6/project-2/flask-k8s-project$ git branch -d fea
ture/add-route
Deleted branch feature/add-route (was a29ff4b).
(venv) vagrant@ubuntu2204:~/day6/project-2/flask-k8s-project$
```

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"]
```

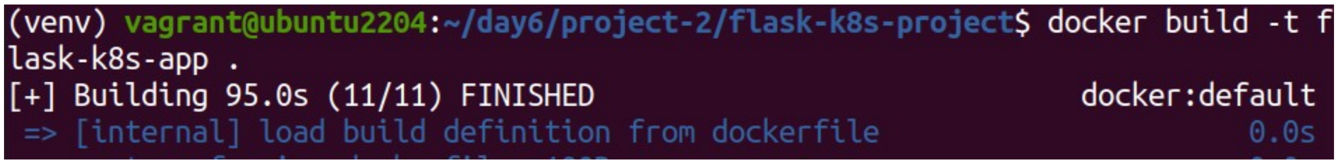


```
(venv) vagrant@ubuntu2204:~/day6/project-2/flask-k8s-project$ vim dockerfile
(venv) vagrant@ubuntu2204:~/day6/project-2/flask-k8s-project$ cat dockerfile
FROM python:3.8-slim

WORKDIR /app

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

COPY . .

EXPOSE 5000

CMD ["python", "app.py"]
(venv) vagrant@ubuntu2204:~/day6/project-2/flask-k8s-project$
```

3.2. Build and Test the Docker Image

Build the Docker image:

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

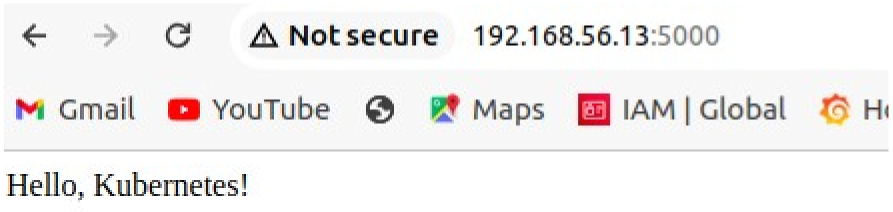


```
(venv) vagrant@ubuntu2204:~/day6/project-2/flask-k8s-project$ docker build -t f
lask-k8s-app .
[+] Building 95.0s (11/11) FINISHED                                docker:default
 => [internal] load build definition from dockerfile                          0.0s
```

Run the Docker container to test:

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

1
2 Access http:192.168.56.13:5000 to see the app running.



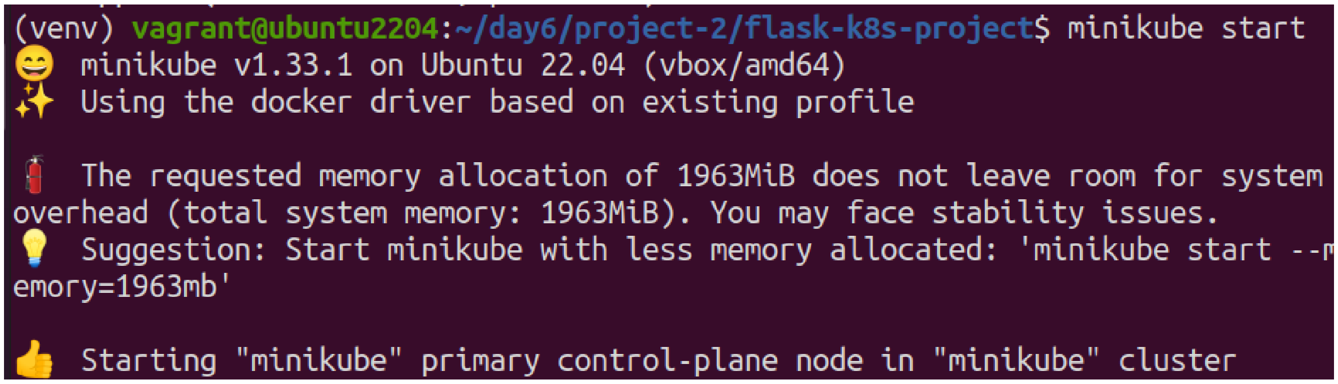
Hello, Kubernetes!

4. Deploying to Minikube Kubernetes

4.1. Start Minikube

Start Minikube:

```
minikube start
```



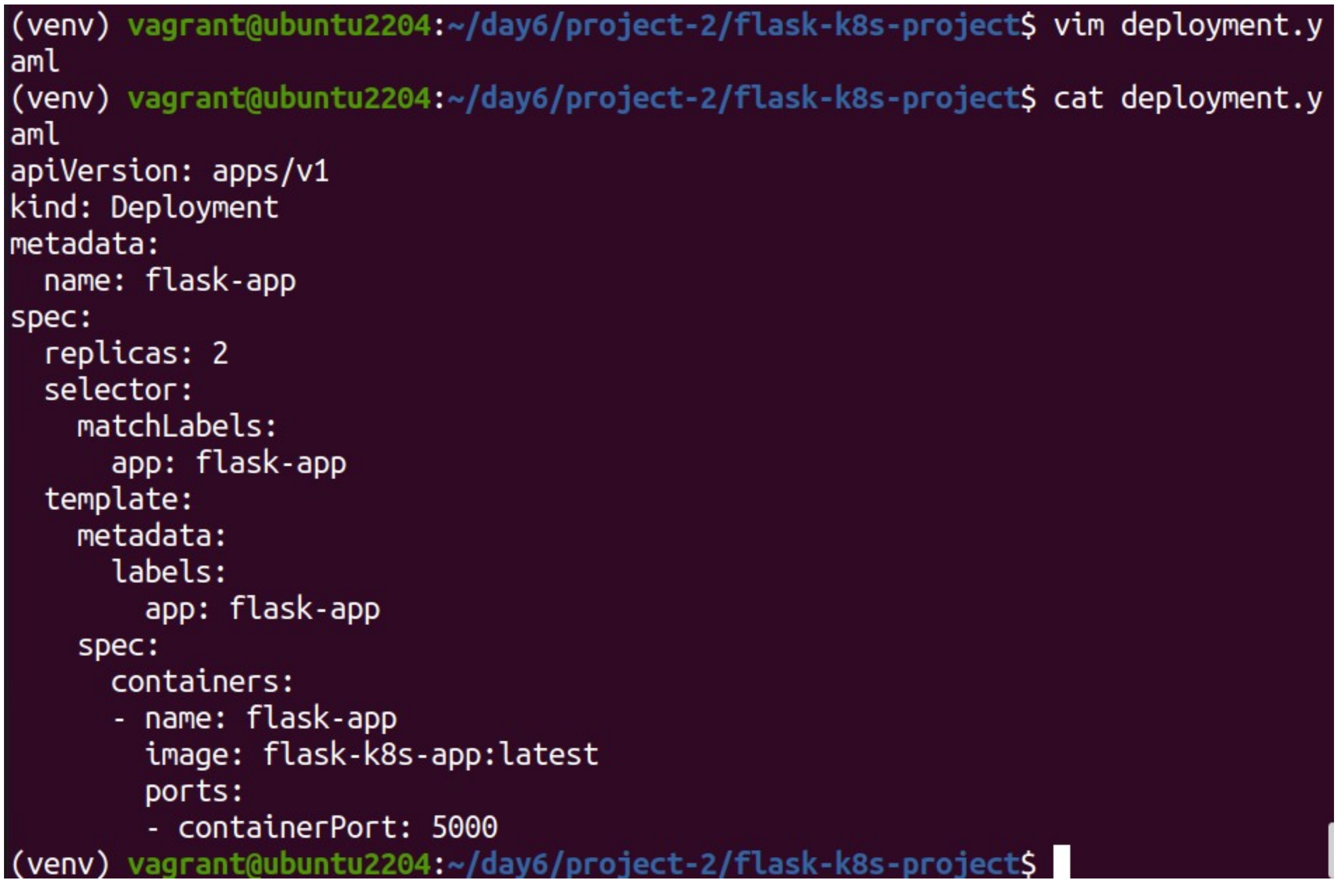
```
(venv) vagrant@ubuntu2204:~/day6/project-2/flask-k8s-project$ minikube start
😄  minikube v1.33.1 on Ubuntu 22.04 (vbox/amd64)
✨  Using the docker driver based on existing profile

🧯  The requested memory allocation of 1963MiB does not leave room for system
overhead (total system memory: 1963MiB). You may face stability issues.
💡  Suggestion: Start minikube with less memory allocated: 'minikube start --m
emory=1963mb'

👍  Starting "minikube" primary control-plane node in "minikube" cluster
```

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:
```

```yaml
    app: flask-app
template:
  metadata:
    labels:
      app: flask-app
  spec:
    containers:
    - name: flask-app
      image: flask-k8s-app:latest
      ports:
      - containerPort: 5000
```

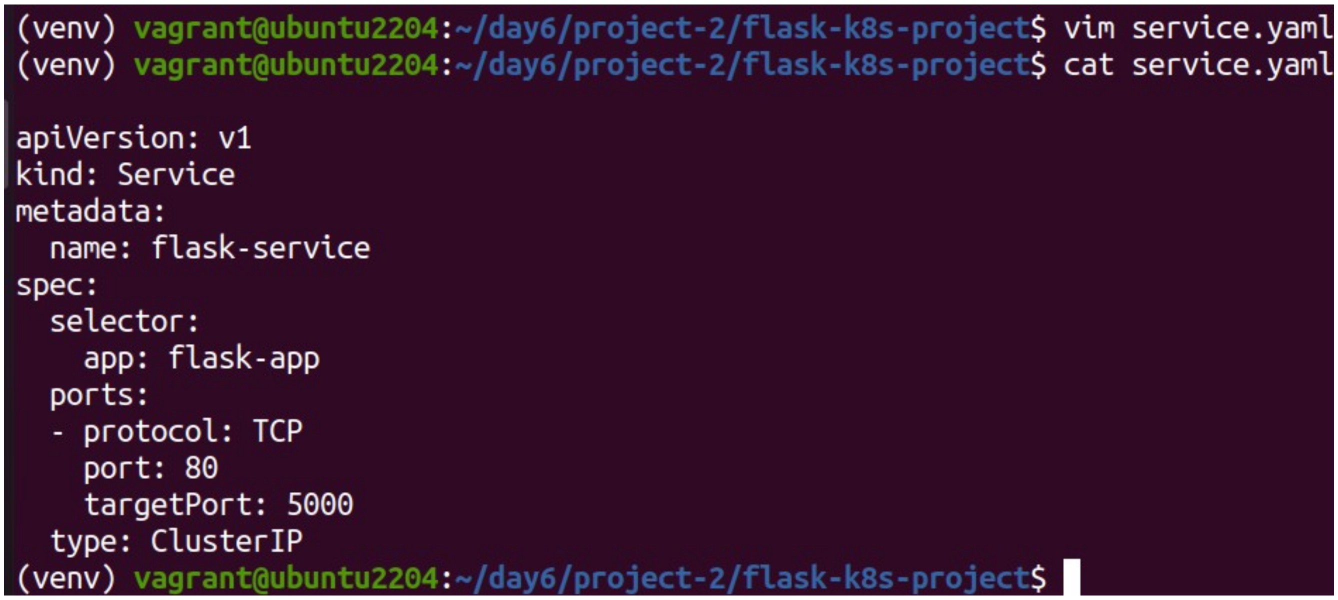


```
(venv) vagrant@ubuntu2204:~/day6/project-2/flask-k8s-project$ vim deployment.y
aml
(venv) vagrant@ubuntu2204:~/day6/project-2/flask-k8s-project$ cat deployment.y
aml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: flask-app
spec:
  replicas: 2
  selector:
    matchLabels:
      app: flask-app
  template:
    metadata:
      labels:
        app: flask-app
    spec:
      containers:
      - name: flask-app
        image: flask-k8s-app:latest
        ports:
        - containerPort: 5000
(venv) vagrant@ubuntu2204:~/day6/project-2/flask-k8s-project$
```

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
```

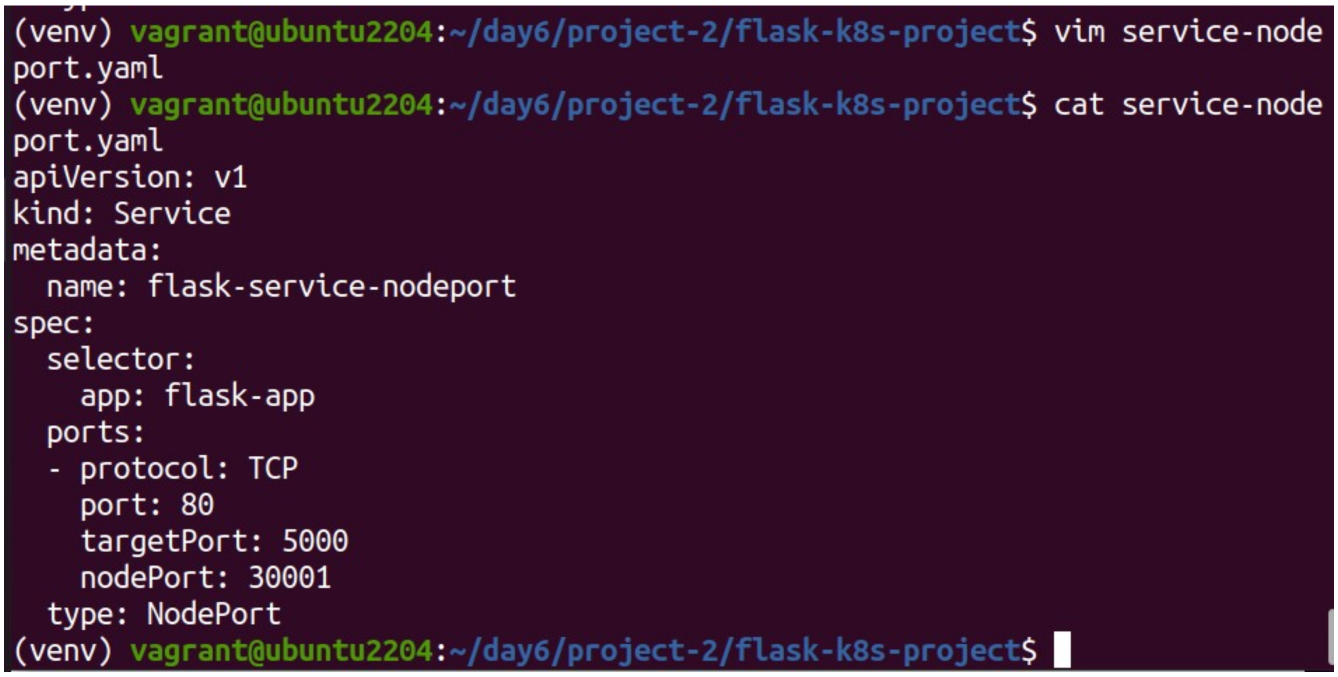


```
(venv) vagrant@ubuntu2204:~/day6/project-2/flask-k8s-project$ vim service.yaml
(venv) vagrant@ubuntu2204:~/day6/project-2/flask-k8s-project$ cat service.yaml

apiVersion: v1
kind: Service
metadata:
  name: flask-service
spec:
  selector:
    app: flask-app
  ports:
  - protocol: TCP
    port: 80
    targetPort: 5000
  type: ClusterIP
(venv) vagrant@ubuntu2204:~/day6/project-2/flask-k8s-project$
```

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
```

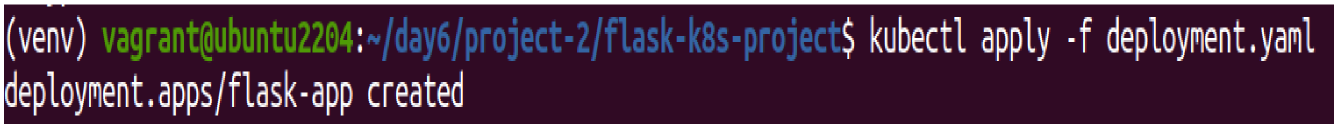


```
(venv) vagrant@ubuntu2204:~/day6/project-2/flask-k8s-project$ vim service-node
port.yaml
(venv) vagrant@ubuntu2204:~/day6/project-2/flask-k8s-project$ cat service-node
port.yaml
apiVersion: v1
kind: Service
metadata:
  name: flask-service-nodeport
spec:
  selector:
    app: flask-app
  ports:
  - protocol: TCP
    port: 80
    targetPort: 5000
    nodePort: 30001
  type: NodePort
(venv) vagrant@ubuntu2204:~/day6/project-2/flask-k8s-project$
```

4.3. Apply Manifests to Minikube

Apply the deployment:

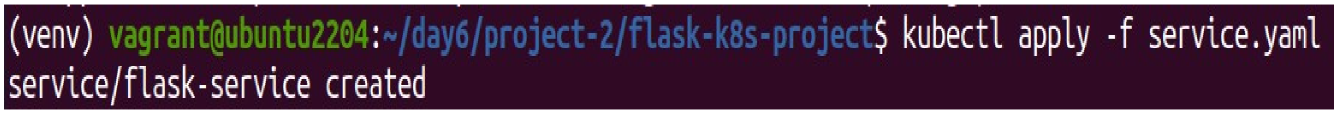
```
kubectl apply -f deployment.yaml
```



```
(venv) vagrant@ubuntu2204:~/day6/project-2/flask-k8s-project$ kubectl apply -f deployment.yaml
deployment.apps/flask-app created
```

Apply the ClusterIP service:

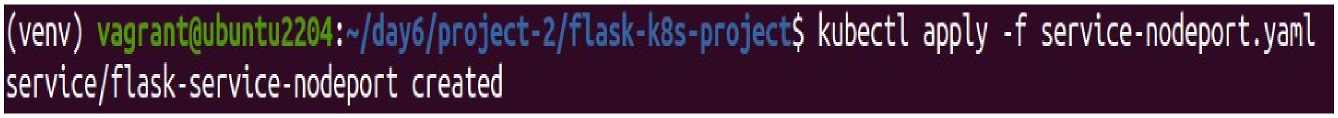
```
kubectl apply -f service.yaml
```



```
(venv) vagrant@ubuntu2204:~/day6/project-2/flask-k8s-project$ kubectl apply -f service.yaml
service/flask-service created
```

Apply the NodePort service:

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



```
(venv) vagrant@ubuntu2204:~/day6/project-2/flask-k8s-project$ kubectl apply -f service-nodeport.yaml
service/flask-service-nodeport created
```

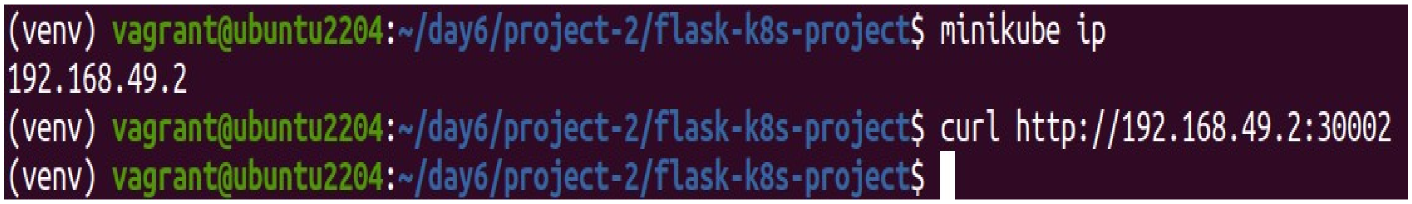
4.4. Access the Application

Get the Minikube IP:

```
minikube ip
```

Access the application using the NodePort:

```
curl http://192.168.49.2:30002
```

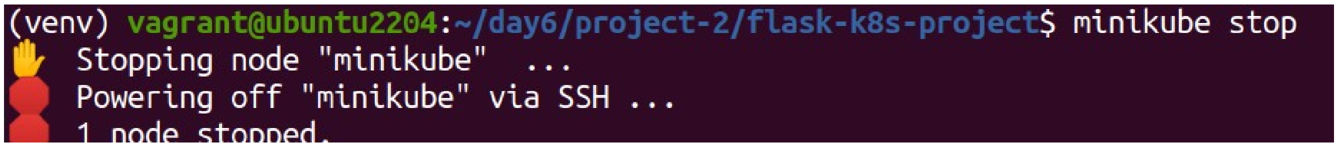


```
(venv) vagrant@ubuntu2204:~/day6/project-2/flask-k8s-project$ minikube ip
192.168.49.2
(venv) vagrant@ubuntu2204:~/day6/project-2/flask-k8s-project$ curl http://192.168.49.2:30002
(venv) vagrant@ubuntu2204:~/day6/project-2/flask-k8s-project$
```

5. Clean Up

Stop Minikube:

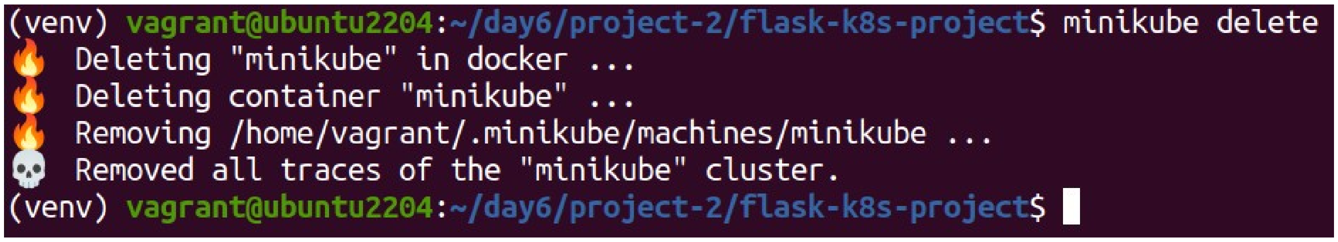
```
minikube stop
```



```
(venv) vagrant@ubuntu2204:~/day6/project-2/flask-k8s-project$ minikube stop
✋  Stopping node "minikube"  ...
🛑  Powering off "minikube" via SSH ...
🛑  1 node stopped.
```

Delete Minikube cluster:

```
minikube delete
```



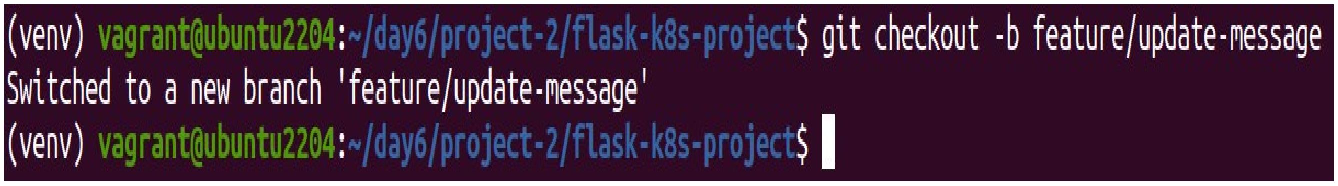
```
(venv) vagrant@ubuntu2204:~/day6/project-2/flask-k8s-project$ minikube delete
🔥  Deleting "minikube" in docker ...
🔥  Deleting container "minikube" ...
🔥  Removing /home/vagrant/.minikube/machines/minikube ...
💀  Removed all traces of the "minikube" cluster.
(venv) vagrant@ubuntu2204:~/day6/project-2/flask-k8s-project$ 
```

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
```



```
(venv) vagrant@ubuntu2204:~/day6/project-2/flask-k8s-project$ git checkout -b feature/update-message
Switched to a new branch 'feature/update-message'
(venv) vagrant@ubuntu2204:~/day6/project-2/flask-k8s-project$ 
```

6.2. Update the Application

Modify app.py to change the message:

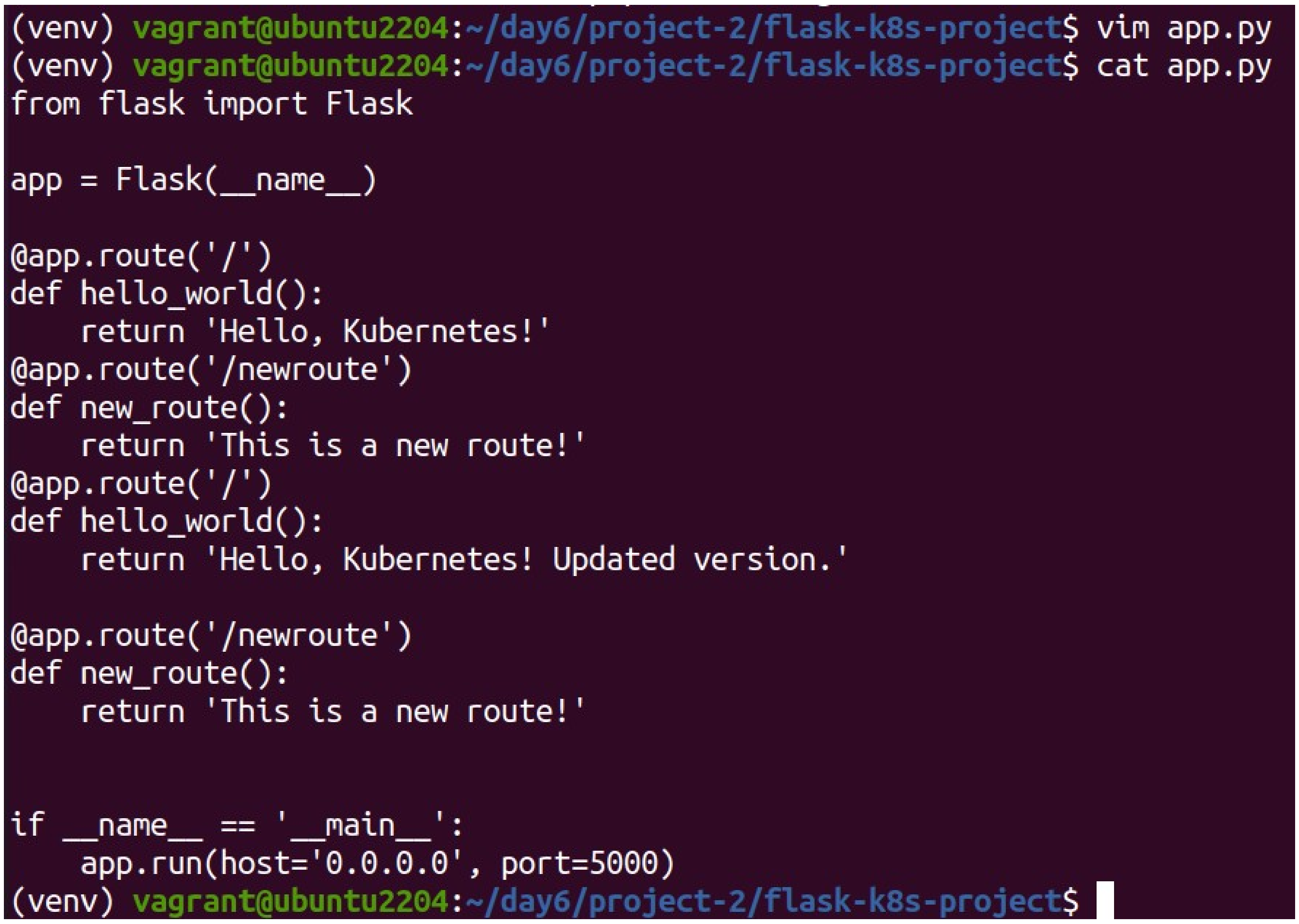
```
@app.route('/')

def hello_world():

    return 'Hello, Kubernetes! Updated version.'
```

cat

@app.route('/newroute')

def new_route():

    return 'This is a new route!'



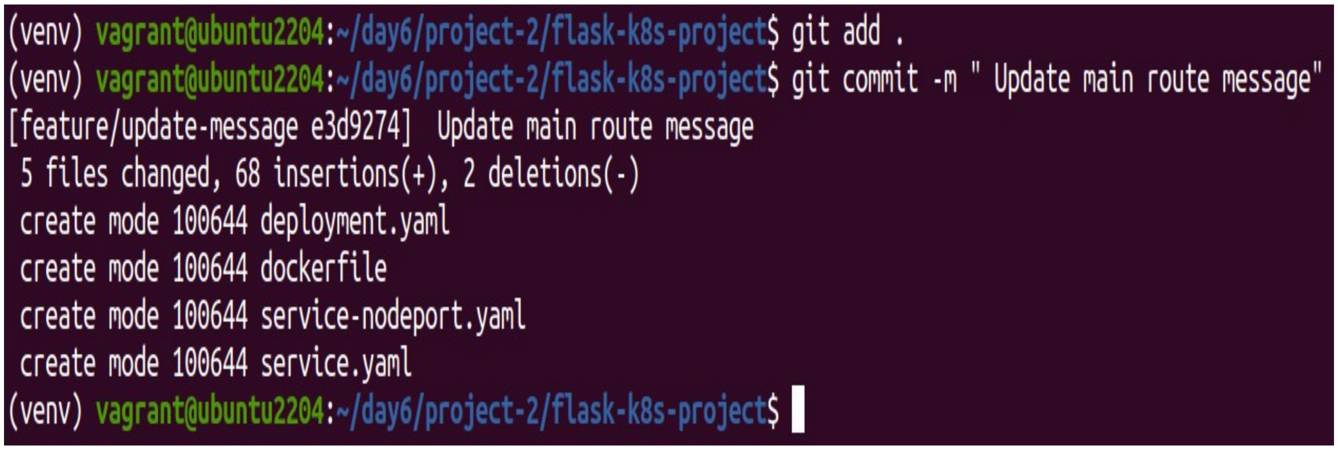
```
(venv) vagrant@ubuntu2204:~/day6/project-2/flask-k8s-project$ vim app.py
(venv) vagrant@ubuntu2204:~/day6/project-2/flask-k8s-project$ cat app.py
from flask import Flask

app = Flask(__name__)

@app.route('/')
def hello_world():
    return 'Hello, Kubernetes!'
@app.route('/newroute')
def new_route():
    return 'This is a new route!'
@app.route('/')
def hello_world():
    return 'Hello, Kubernetes! Updated version.'

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


if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)
(venv) vagrant@ubuntu2204:~/day6/project-2/flask-k8s-project$
```

6.3. Commit the Changes

Add and commit the changes:

```
git add .
```

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



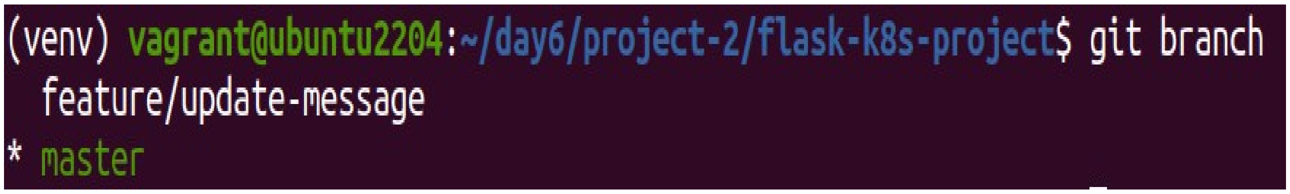
```
(venv) vagrant@ubuntu2204:~/day6/project-2/flask-k8s-project$ git add .
(venv) vagrant@ubuntu2204:~/day6/project-2/flask-k8s-project$ git commit -m " Update main route message"
[feature/update-message e3d9274]  Update main route message
 5 files changed, 68 insertions(+), 2 deletions(-)
 create mode 100644 deployment.yaml
 create mode 100644 dockerfile
 create mode 100644 service-nodeport.yaml
 create mode 100644 service.yaml
(venv) vagrant@ubuntu2204:~/day6/project-2/flask-k8s-project$ 
```

7. Merge the Changes and Rebuild the Docker Image

7.1. Merge the Feature Branch

Switch back to the `main` branch:

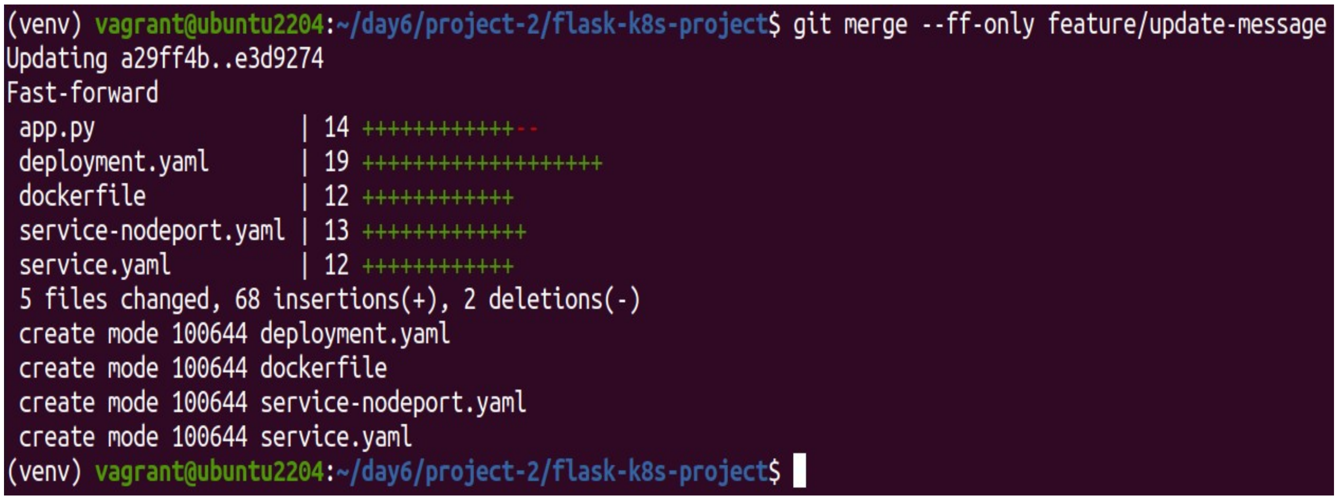
```
git checkout main
```



```
(venv) vagrant@ubuntu2204:~/day6/project-2/flask-k8s-project$ git branch
  feature/update-message
* master
```

Merge the `feature/update-message` branch:

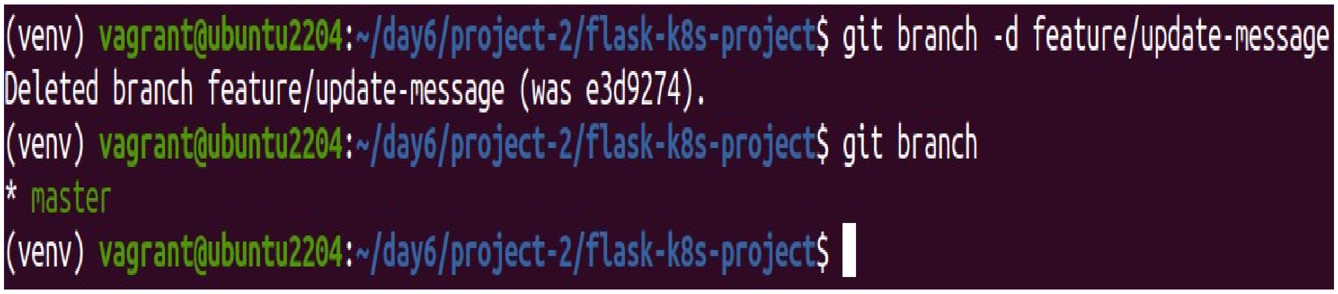
```
git merge --ff-only feature/update-message
```



```
(venv) vagrant@ubuntu2204:~/day6/project-2/flask-k8s-project$ git merge --ff-only feature/update-message
Updating a29ff4b..e3d9274
Fast-forward
 app.py                  | 14 ++++++++++++--
 deployment.yaml         | 19 +++++++++++++++++++
 dockerfile              | 12 ++++++++++++
 service-nodeport.yaml   | 13 +++++++++++++
 service.yaml            | 12 ++++++++++++
 5 files changed, 68 insertions(+), 2 deletions(-)
 create mode 100644 deployment.yaml
 create mode 100644 dockerfile
 create mode 100644 service-nodeport.yaml
 create mode 100644 service.yaml
(venv) vagrant@ubuntu2204:~/day6/project-2/flask-k8s-project$
```

Delete the feature branch:

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

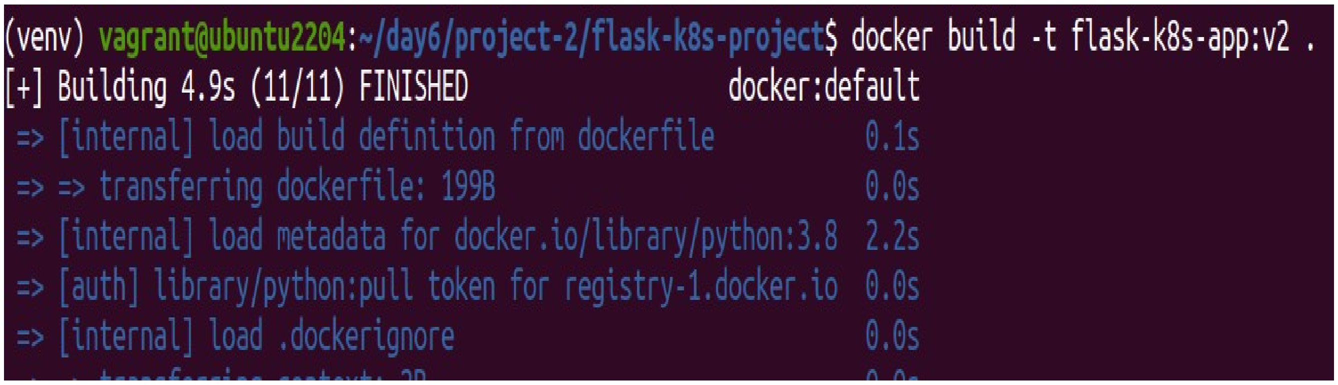


```
(venv) vagrant@ubuntu2204:~/day6/project-2/flask-k8s-project$ git branch -d feature/update-message
Deleted branch feature/update-message (was e3d9274).
(venv) vagrant@ubuntu2204:~/day6/project-2/flask-k8s-project$ git branch
* master
(venv) vagrant@ubuntu2204:~/day6/project-2/flask-k8s-project$
```

7.2. Rebuild the Docker Image

Rebuild the Docker image with a new tag:

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



```
(venv) vagrant@ubuntu2204:~/day6/project-2/flask-k8s-project$ docker build -t flask-k8s-app:v2 .
[+] Building 4.9s (11/11) FINISHED                          docker:default
 => [internal] load build definition from dockerfile                  0.1s
 => => transferring dockerfile: 199B                                  0.0s
 => [internal] load metadata for docker.io/library/python:3.8         2.2s
 => [auth] library/python:pull token for registry-1.docker.io         0.0s
 => [internal] load .dockerignore                                     0.0s
```

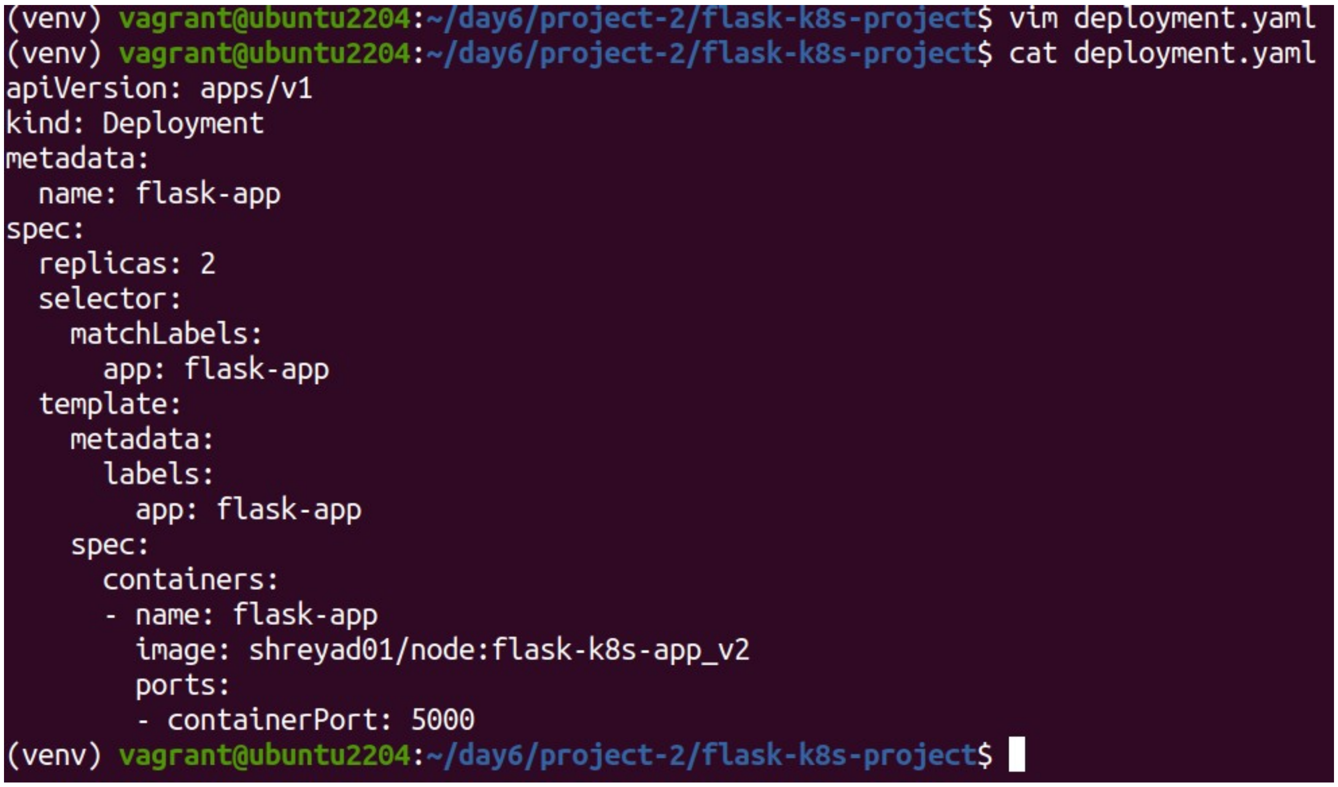
8. Update Kubernetes Deployment

8.1. Update the Deployment Manifest

Modify `deployment.yaml` to use the new image version:

```yaml
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: shreyad01/node:flask-k8s-app_v2
      ports:
- containerPort: 5000
```



```
(venv) vagrant@ubuntu2204:~/day6/project-2/flask-k8s-project$ vim deployment.yaml
(venv) vagrant@ubuntu2204:~/day6/project-2/flask-k8s-project$ cat deployment.yaml
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: shreyad01/node:flask-k8s-app_v2
        ports:
        - containerPort: 5000
(venv) vagrant@ubuntu2204:~/day6/project-2/flask-k8s-project$
```

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://192.168.49.2:8080` to see the updated message.

9.2. Access Through NodePort Service

1 Access the application using the NodePort:

```
curl http://192.168.49.2:30001
```