CMPE 272 Hello-World Assignment

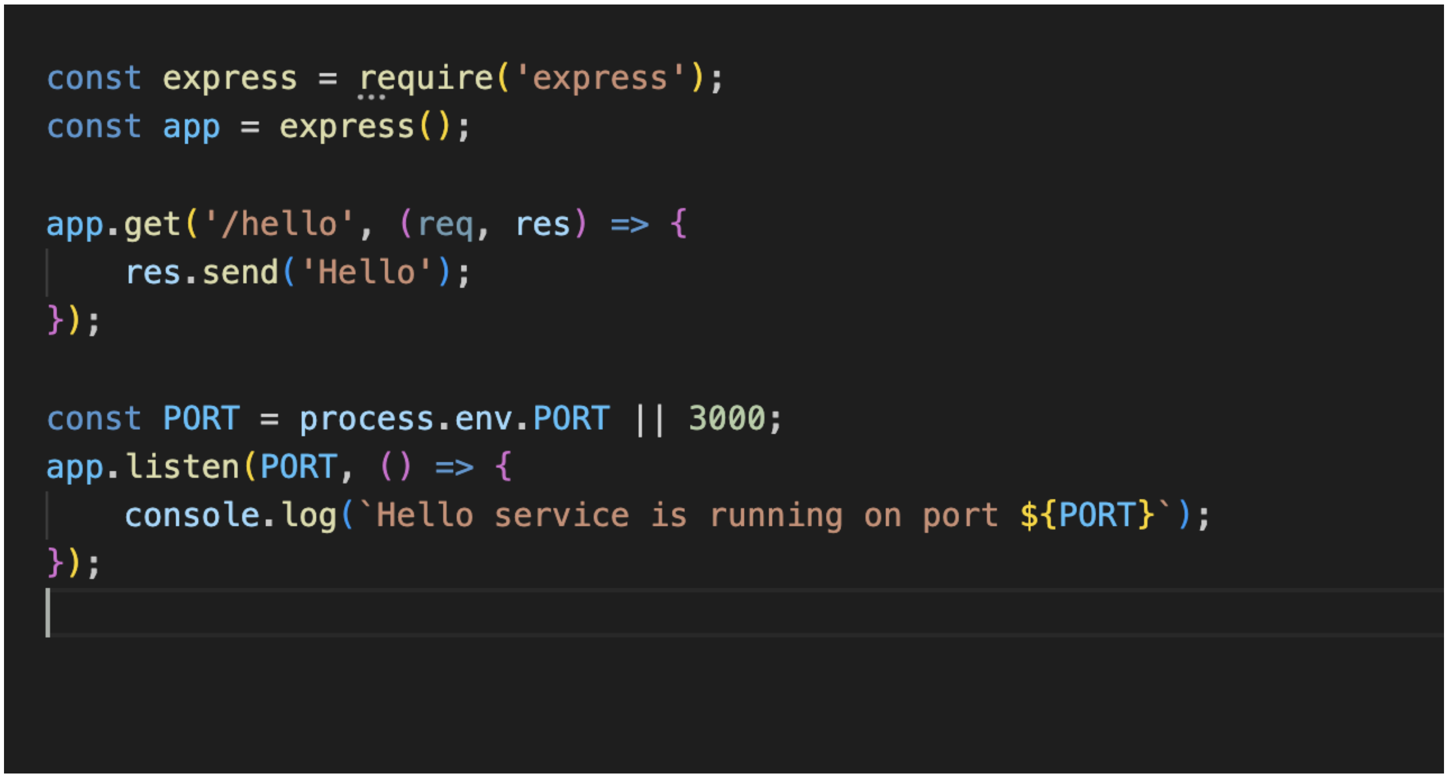
Step 1:

My Git Repository is https://github.com/DhruvilJayani/cmpe-272 and I have used Node and Express.JS to create my services.

Step 2:

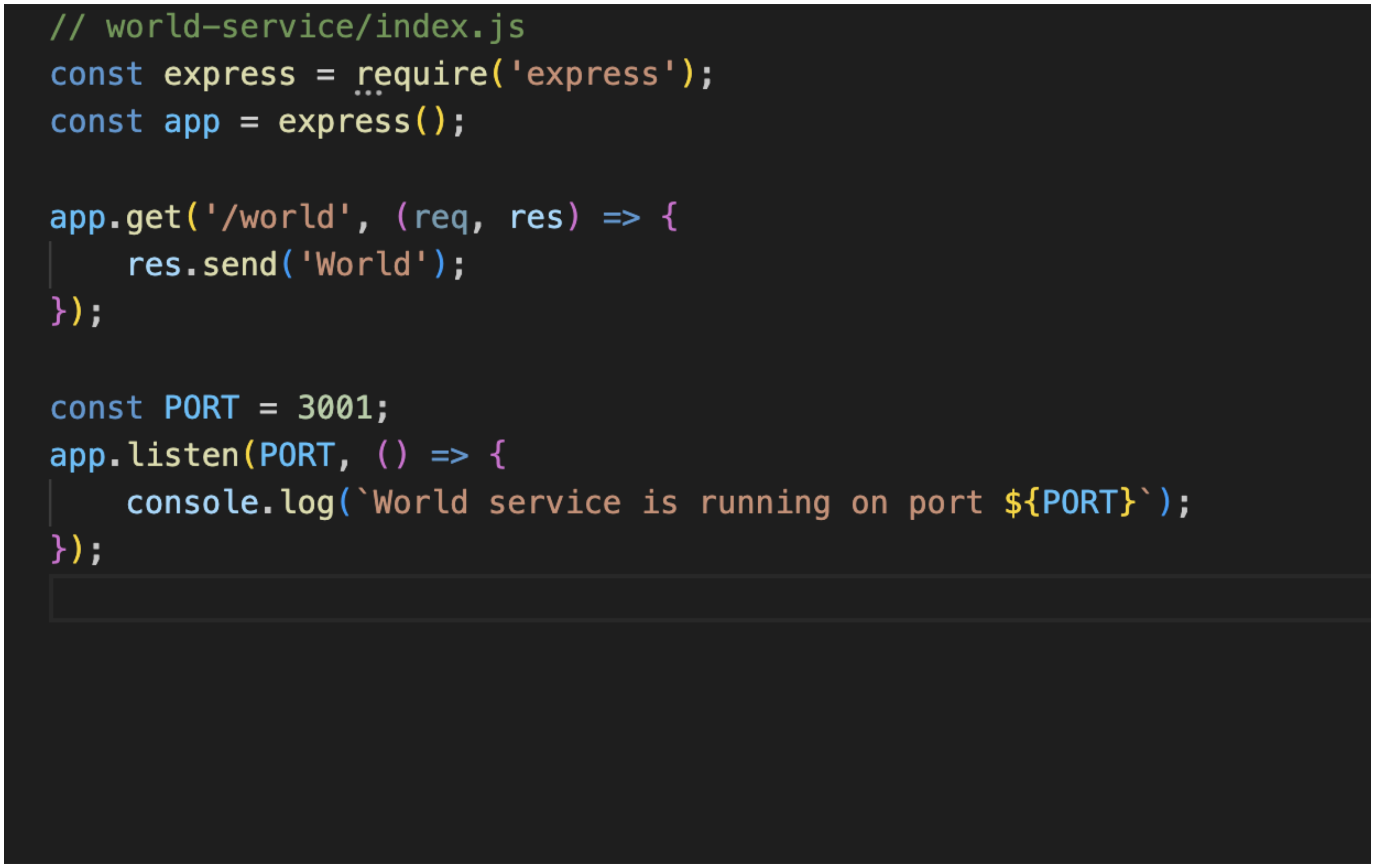
I created two services for Hello and World.
Each Service has a simple Express server which has an Endpoint which returns "Hello" and "World".

Code Snippet of Hello Service :



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

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

const PORT = process.env.PORT || 3000;
app.listen(PORT, () => {
    console.log(`Hello service is running on port ${PORT}`);
});
```

Code snippet for World service :

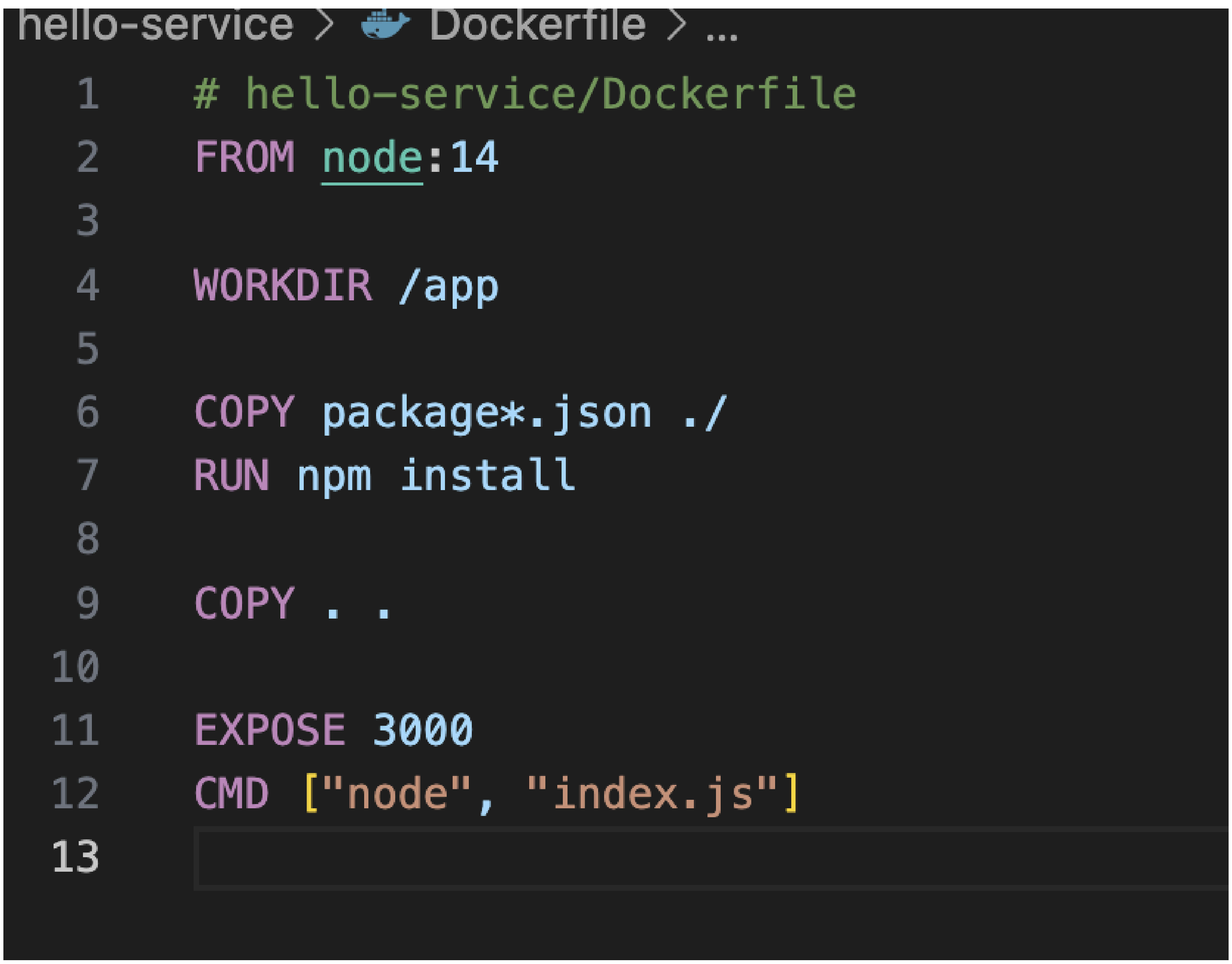


```
// world-service/index.js
const express = require('express');
const app = express();

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

const PORT = 3001;
app.listen(PORT, () => {
    console.log(`World service is running on port ${PORT}`);
});
```

I tested both of them on Postman and Browser.

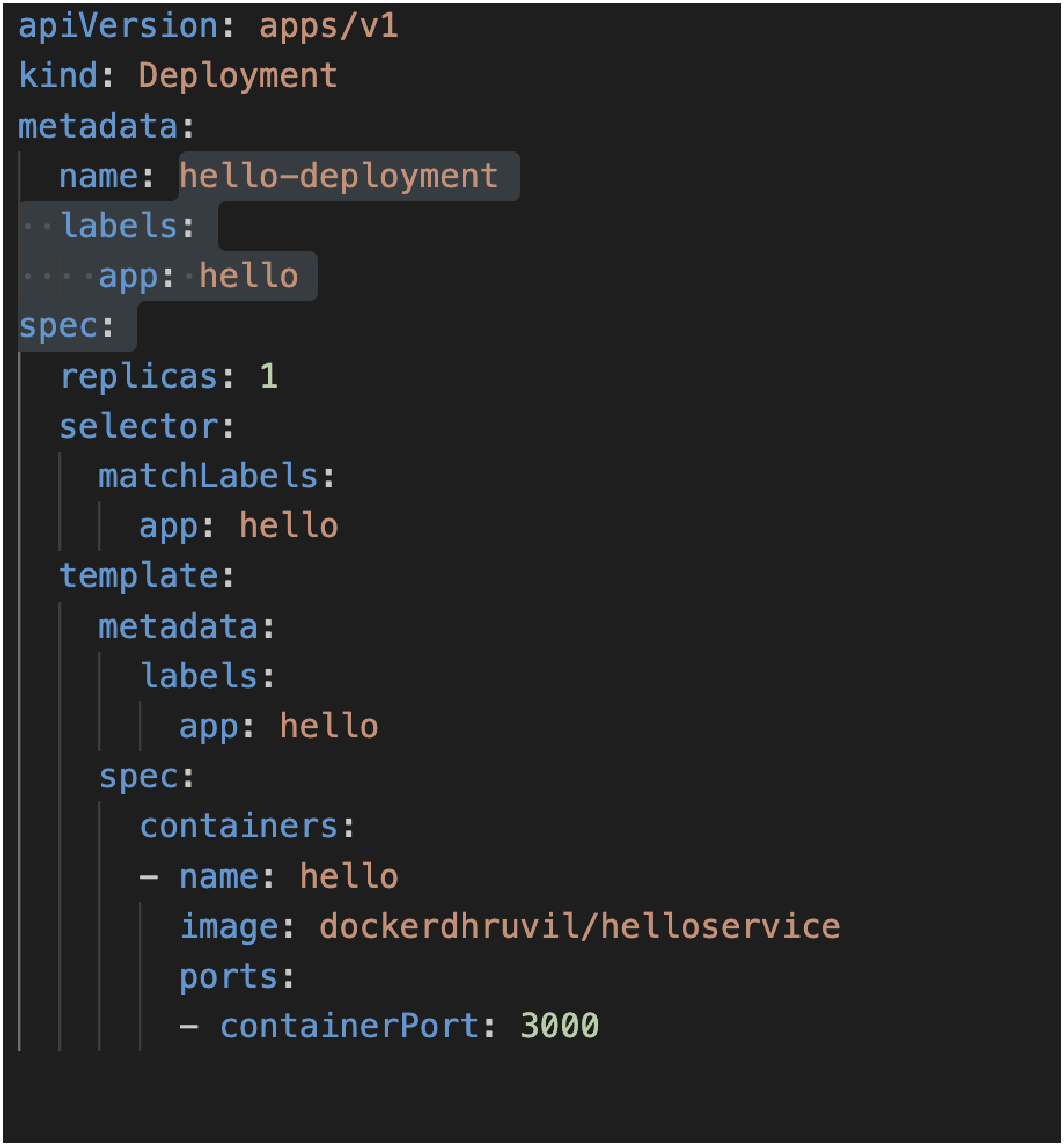
Step 3 :
Created a Docker File for Hello and World. That specified the node version and all necessary command lines to execute image in the container. I also tested them locally through port mapping and then I uploaded them on Docker Hub.

Docker File of Hello Service



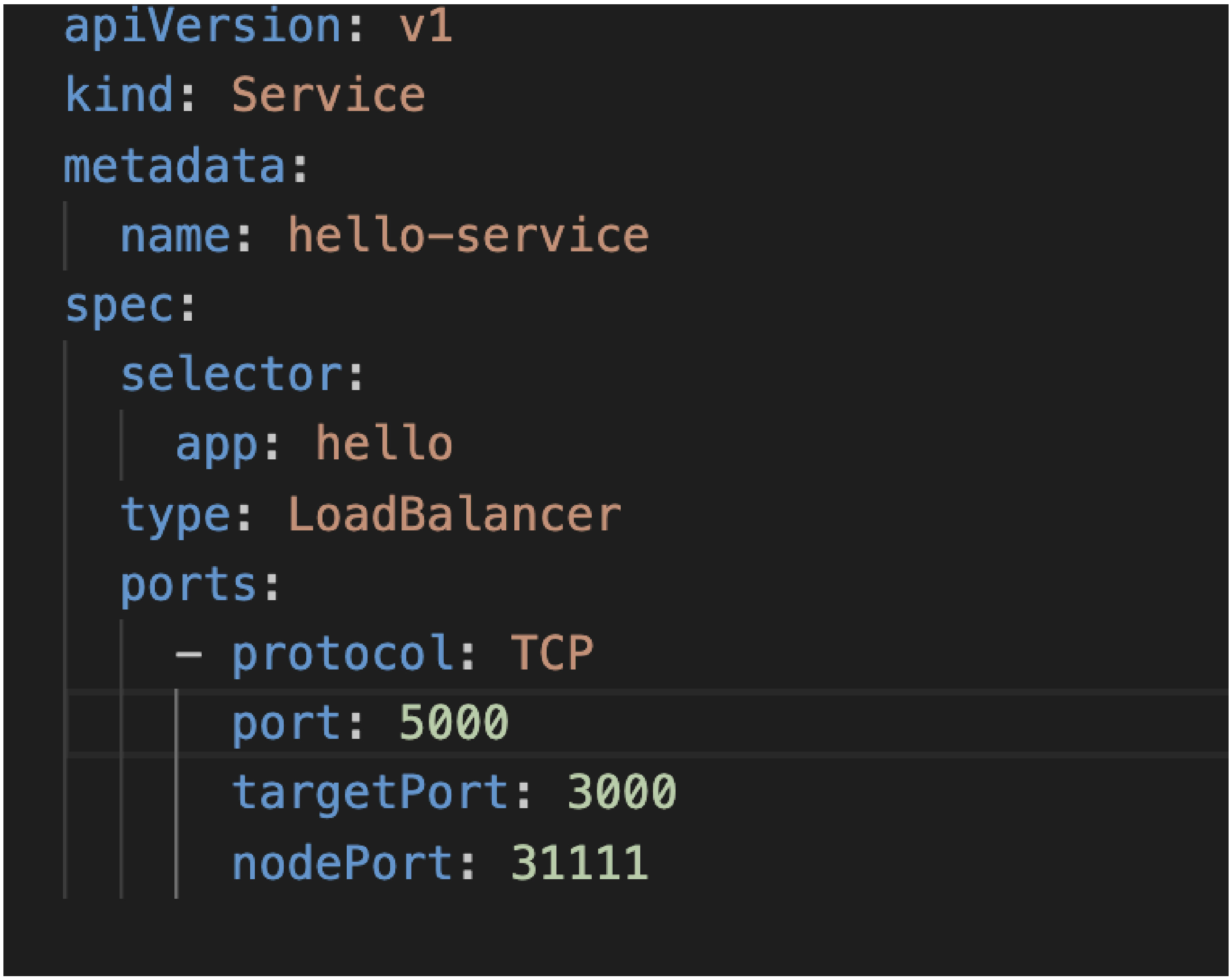
```dockerfile
# hello-service/Dockerfile
FROM node:14

WORKDIR /app

COPY package*.json ./
RUN npm install

COPY . .

EXPOSE 3000
CMD ["node", "index.js"]

```

Step 4 :
Then I chose minikube for my Offline Kubernetes deployment. I defined deployment.yml page and services.yml for each services. Now the deployment files fetches the docs images from docker hub and I used services file to create load balancer.



```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: hello-deployment
  labels:
    app: hello
spec:
  replicas: 1
  selector:
    matchLabels:
      app: hello
  template:
    metadata:
      labels:
        app: hello
    spec:
      containers:
      - name: hello
        image: dockerdhruvil/helloservice
        ports:
        - containerPort: 3000
```

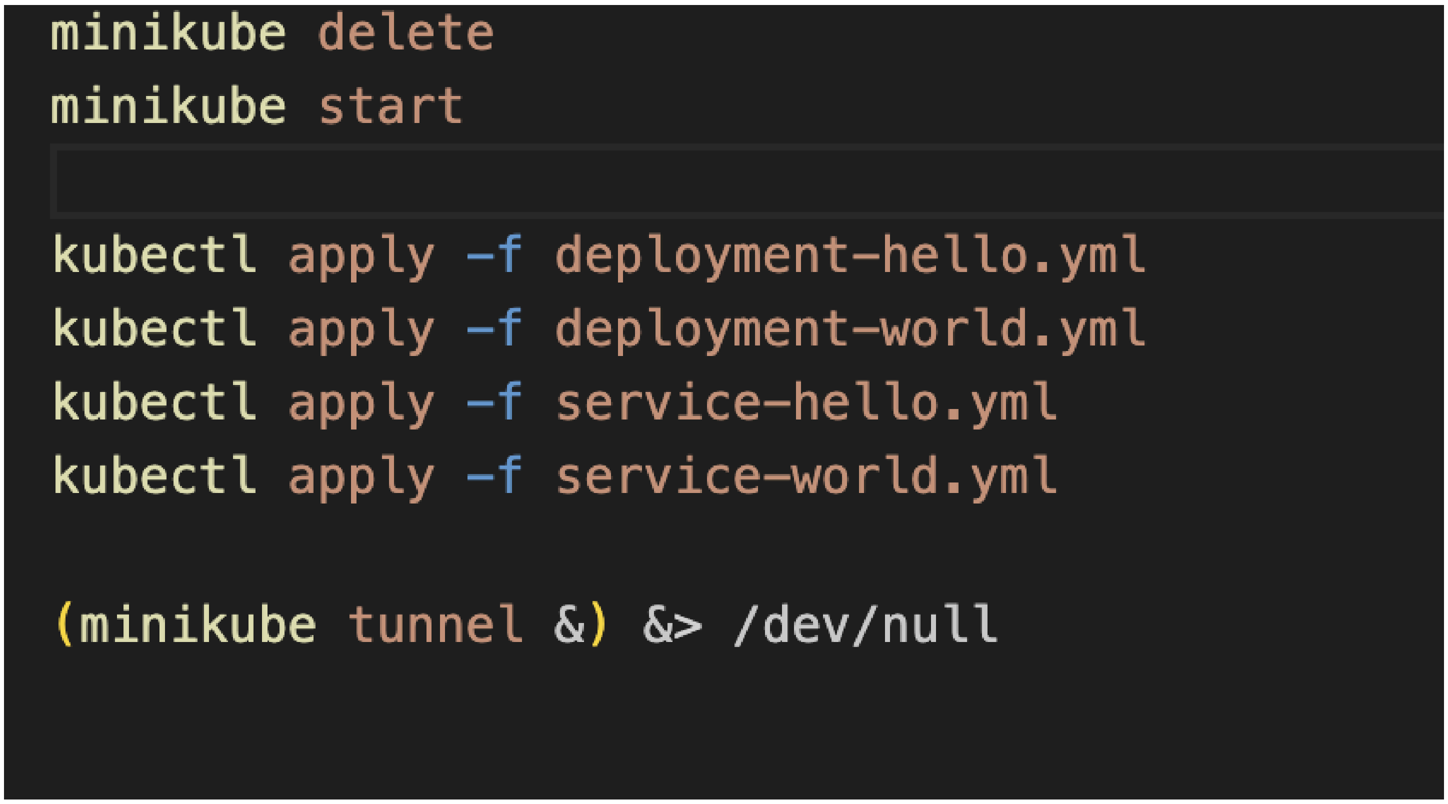
service-hello.yml



```yaml
apiVersion: v1
kind: Service
metadata:
  name: hello-service
spec:
  selector:
    app: hello
  type: LoadBalancer
  ports:
    - protocol: TCP
      port: 5000
      targetPort: 3000
      nodePort: 31111
```

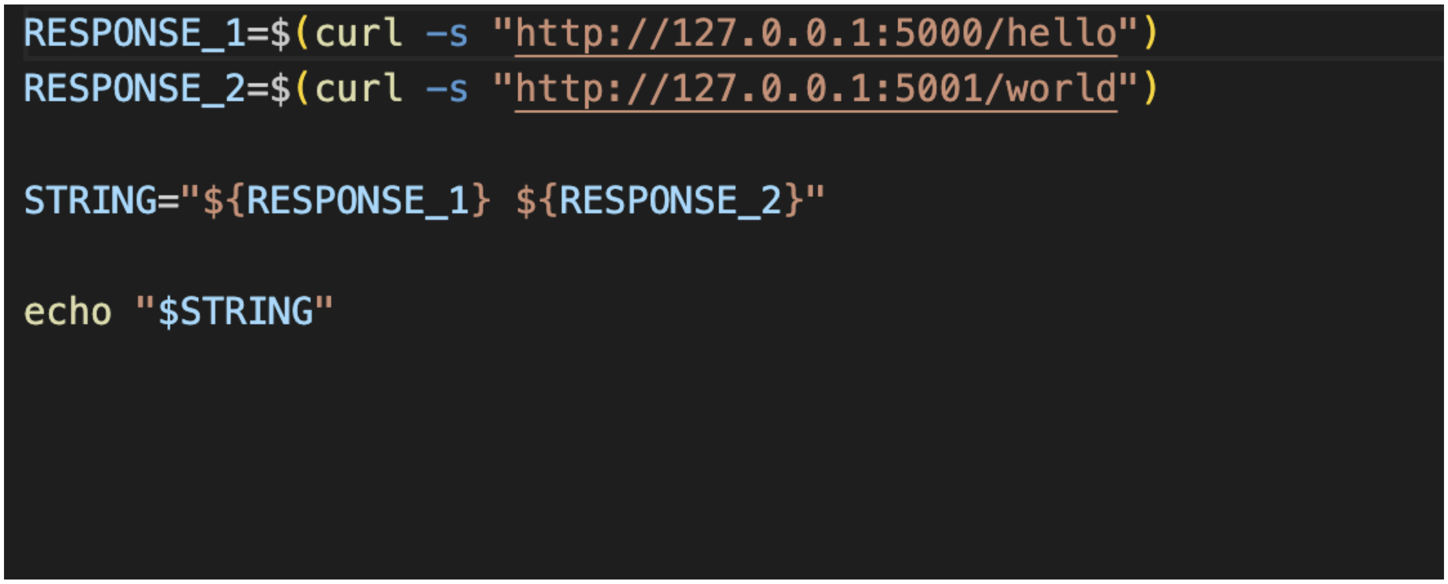
Step 5 :
Developed a comprehensive Bash script to set up and run the cluster and services. Additionally, created another script, app.sh, located in the root directory, which interacts with containers within the cluster, retrieves data from both services, and outputs the results.Once the Kubernetes services are deployed I used the tunnel command. This makes the required ports open to access.

Setup.sh



```
minikube delete
minikube start


kubectl apply -f deployment-hello.yml
kubectl apply -f deployment-world.yml
kubectl apply -f service-hello.yml
kubectl apply -f service-world.yml

(minikube tunnel &) &> /dev/null
```

app.sh



```
RESPONSE_1=$(curl -s "http://127.0.0.1:5000/hello")
RESPONSE_2=$(curl -s "http://127.0.0.1:5001/world")

STRING="${RESPONSE_1} ${RESPONSE_2}"

echo "$STRING"
```

Step 6 :
Created Documentation and Readme File to setup the project and test it.