

PROGRAM – 3

Project Structure:

1. Dockerfile
2. Server.js
3. package.json
4. node_modules

Dockerfile:

```
👉 Dockerfile > ...
1
2  FROM node:18-alpine
3
4
5  WORKDIR /app
6
7
8  COPY package*.json ./
9
10
11  RUN npm install
12
13
14  COPY . .
15  EXPOSE 3001
16
17  CMD [ "npm", "start" ]
```

Server.js

```
Devops-program4 > js server.js > ...
1  const express = require("express");
2  const app = express();
3  const port=3001
4
5  app.get("/", (req, res) => {
6    |  res.send("Hello Geeks");
7  });
8
9  app.get("/new", (req, res) => {
10   |  res.send("welcome to new page");
11 });
12
13 // Default route (404 handler)
14 app.use((req, res) => {
15   |  res.status(404).send("PAGE NOT FOUND");
16 });
17
18 // Server setup
19 app.listen(port, () => {
20   |  console.log(`Server listening on ${port}`);
21 }) ;
```

To run the dockerfile:

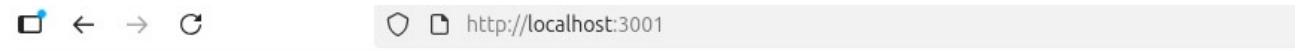
```
# To build the image  
sudo docker build -t thirdprg
```

```
# To run the image  
sudo docker run -d -p 3001:3001 thirdprg
```

OUTPUT:



```
PROMISES   SCHEMATIC   DEBUG CONSOLE   TERMINAL   PORTS  
ganeshv@ganeshv-HP-ENVY-x360-m6-Convertible:~/Desktop/devopslab/thirdprg$ sudo docker run -p 3001:3001 thirdprg  
> lab-3@1.0.0 start  
> node server.js  
  
Server listening on 3001  
^Cnpm error path /app  
npm error command failed  
npm error signal SIGINT  
npm error command sh -c node server.js  
npm error A complete log of this run can be found in: /root/.npm/_logs/2025-11-07T04_42_25_752Z-debug-0.log  
ganeshv@ganeshv-HP-ENVY-x360-m6-Convertible:~/Desktop/devopslab/thirdprg$ sudo docker run -p 3001:3001 thirdprg  
> lab-3@1.0.0 start  
> node server.js  
  
Server listening on 3001
```



Hello Geeks

PROGRAM -2

Project Structure:

1. node_modules
2. package.json
3. package_lock.json
4. src
 - index.js
5. Dockerfile

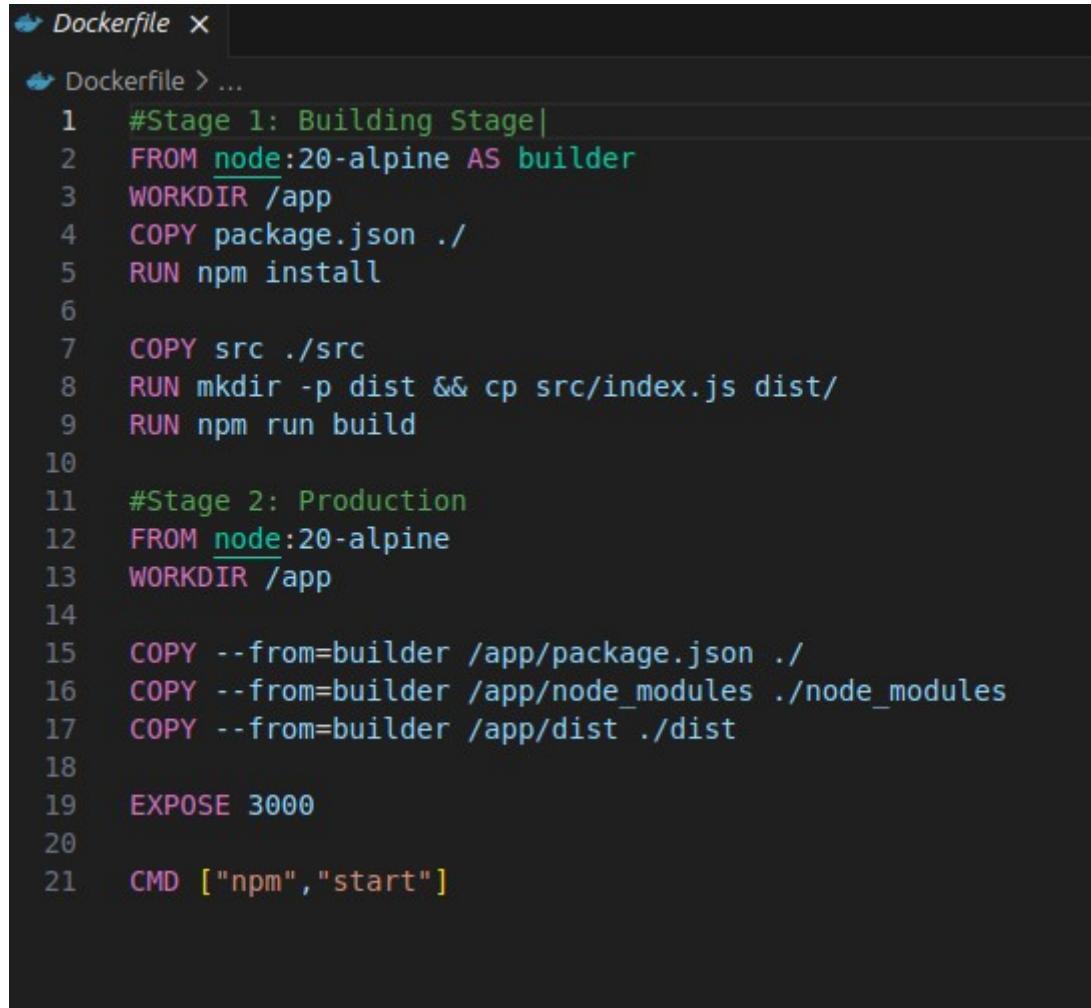
index.js

```
JS index.js  x

src > JS index.js > ...
1 const express = require('express');
2 const app = express();
3 const port = 3000;
4
5 app.get('/',(req,res)=>{
6   res.send(`
7     <script>alert("PRODUCTION!")</script>`)
8 });
9
10 app.listen(port, ()=>{
11   console.log("Server running on port 3000");
12 });


```

Dockerfile



```
 1  #Stage 1: Building Stage
 2  FROM node:20-alpine AS builder
 3  WORKDIR /app
 4  COPY package.json ./
 5  RUN npm install
 6
 7  COPY src ./src
 8  RUN mkdir -p dist && cp src/index.js dist/
 9  RUN npm run build
10
11 #Stage 2: Production
12 FROM node:20-alpine
13 WORKDIR /app
14
15 COPY --from=builder /app/package.json ./
16 COPY --from=builder /app/node_modules ./node_modules
17 COPY --from=builder /app/dist ./dist
18
19 EXPOSE 3000
20
21 CMD [ "npm", "start" ]
```

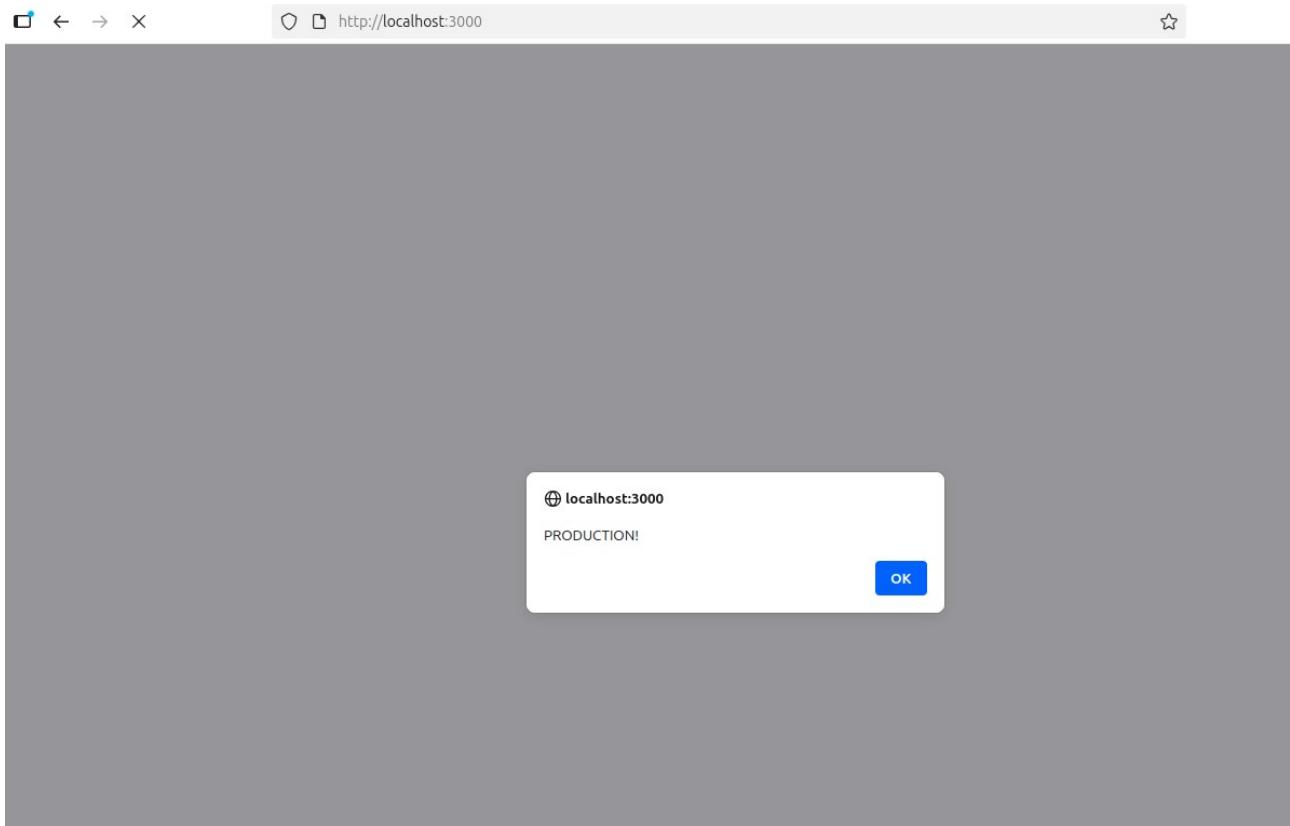
To run the dockerfile:

```
# To build the image
sudo docker build -t secondprg
```

```
# To run the image
sudo docker run -d -p 3000:3000 secondprg
```

OUTPUT:

```
ganeshv@ganeshv-HP-ENVY-x360-m6-Convertible:~/Desktop/devopslab/secondprogram$ sudo docker build -t secpngr .  
=> CACHED [builder 3/7] COPY package.json ./  
=> CACHED [builder 4/7] RUN npm install  
=> CACHED [builder 5/7] COPY src ./src  
=> CACHED [builder 6/7] RUN mkdir -p dist && cp src/index.js dist/  
=> CACHED [builder 7/7] RUN npm run build  
=> CACHED [stage-1 3/5] COPY --from=builder /app/package.json ./  
=> CACHED [stage-1 4/5] COPY --from=builder /app/node_modules ./node_modules  
=> CACHED [stage-1 5/5] COPY --from=builder /app/dist ./dist  
=> exporting to image  
=> => exporting layers  
=> => writing image sha256:e8ab6280169cfb0c67e4db237279e5d520c702dc5be9c85fa7f10df42c03e6f0  
=> => naming to docker.io/library/secpngr  
ganeshv@ganeshv-HP-ENVY-x360-m6-Convertible:~/Desktop/devopslab/secondprogram$ sudo docker run -p 3000:3000 secpngr  
  
> second@1.0.0 start  
> node dist/index.js  
  
Server running on port 3000
```

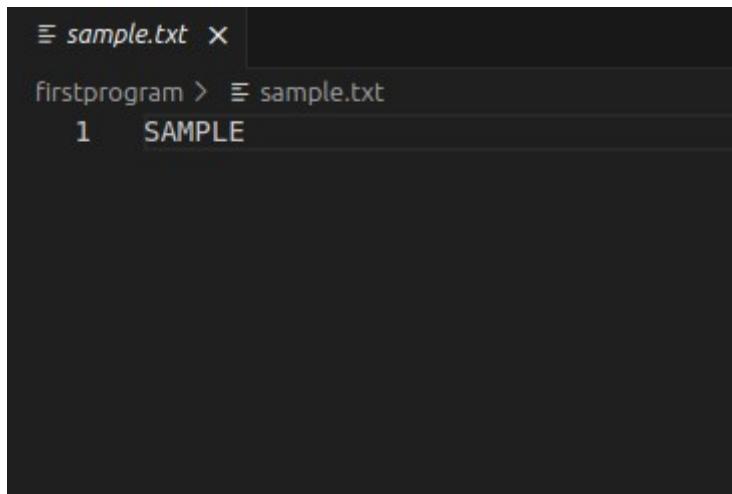


PROGRAM-1

Project Structure:

1. sample.txt
2. Dockerfile

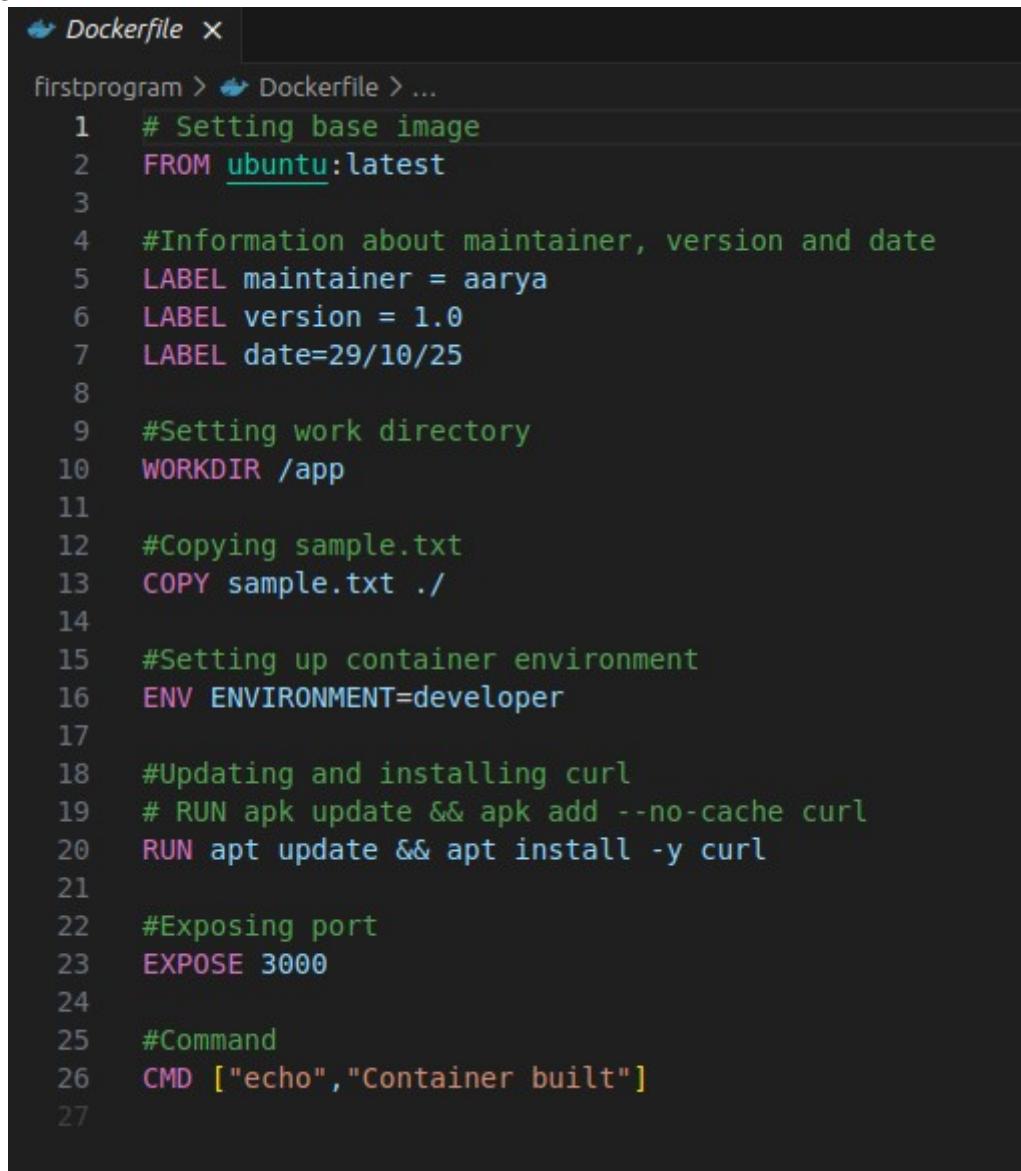
Sample.txt



A screenshot of a terminal window with a dark background. At the top, there is a tab labeled "sample.txt" with an "X" icon. Below the tabs, the text "firstprogram > sample.txt" is displayed. The main content area shows the text "1 SAMPLE".

```
sample.txt x
firstprogram > sample.txt
1 SAMPLE
```

Dockerfile



```
 1  # Setting base image
 2  FROM ubuntu:latest
 3
 4  #Information about maintainer, version and date
 5  LABEL maintainer = aarya
 6  LABEL version = 1.0
 7  LABEL date=29/10/25
 8
 9  #Setting work directory
10 WORKDIR /app
11
12 #Copying sample.txt
13 COPY sample.txt ./
14
15 #Setting up container environment
16 ENV ENVIRONMENT=developer
17
18 #Updating and installing curl
19 # RUN apk update && apk add --no-cache curl
20 RUN apt update && apt install -y curl
21
22 #Exposing port
23 EXPOSE 3000
24
25 #Command
26 CMD ["echo", "Container built"]
27
```

To run the dockerfile:

```
# To build the image
sudo docker build -t firstprg
```

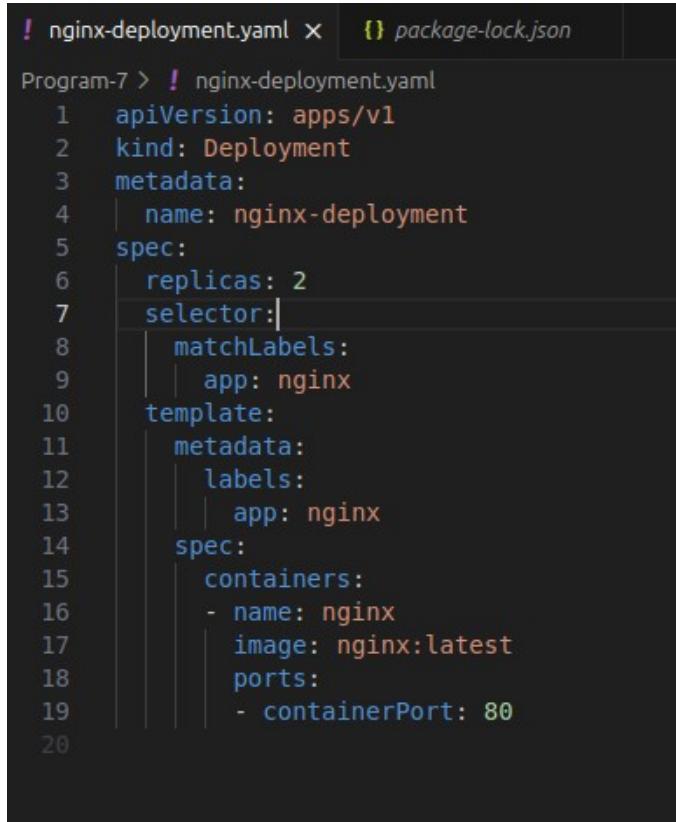
```
# To run the image
sudo docker run -d -p 3000:3000 firstprg
```

PROGRAM-7

Project Structure:

1. nginx-deployment.yaml

nginx-deployment.yaml file



```
! nginx-deployment.yaml x package-lock.json
Program-7 > ! nginx-deployment.yaml
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: nginx-deployment
5  spec:
6    replicas: 2
7    selector:
8      matchLabels:
9        app: nginx
10   template:
11     metadata:
12       labels:
13         app: nginx
14     spec:
15       containers:
16         - name: nginx
17           image: nginx:latest
18           ports:
19             - containerPort: 80
20
```

To run the .yaml file:

```
# Applying deployment configuration to the cluster
kubectl apply -f nginx-deployment.yaml
```

```
# Checking if the deployment was successful
kubectl get deployment
```

```
# List all pods
kubectl get pods
```

```
# Exposing the deployment as a service
kubectl expose deployment nginx-deployment --type=NodePort --port=80
```

```
# list all services
kubectl get services
```

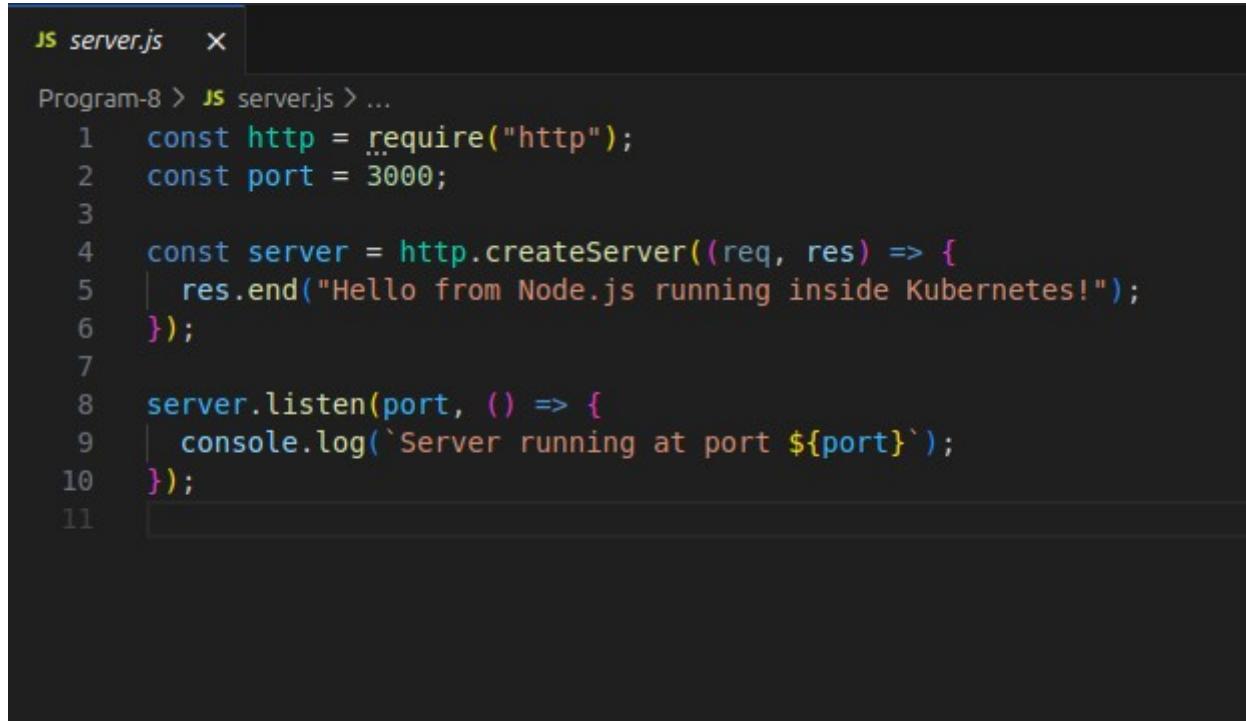
```
# Accessing the exposed services
minikube service nginx deployment
```

PROGRSM-8

Project Structure:

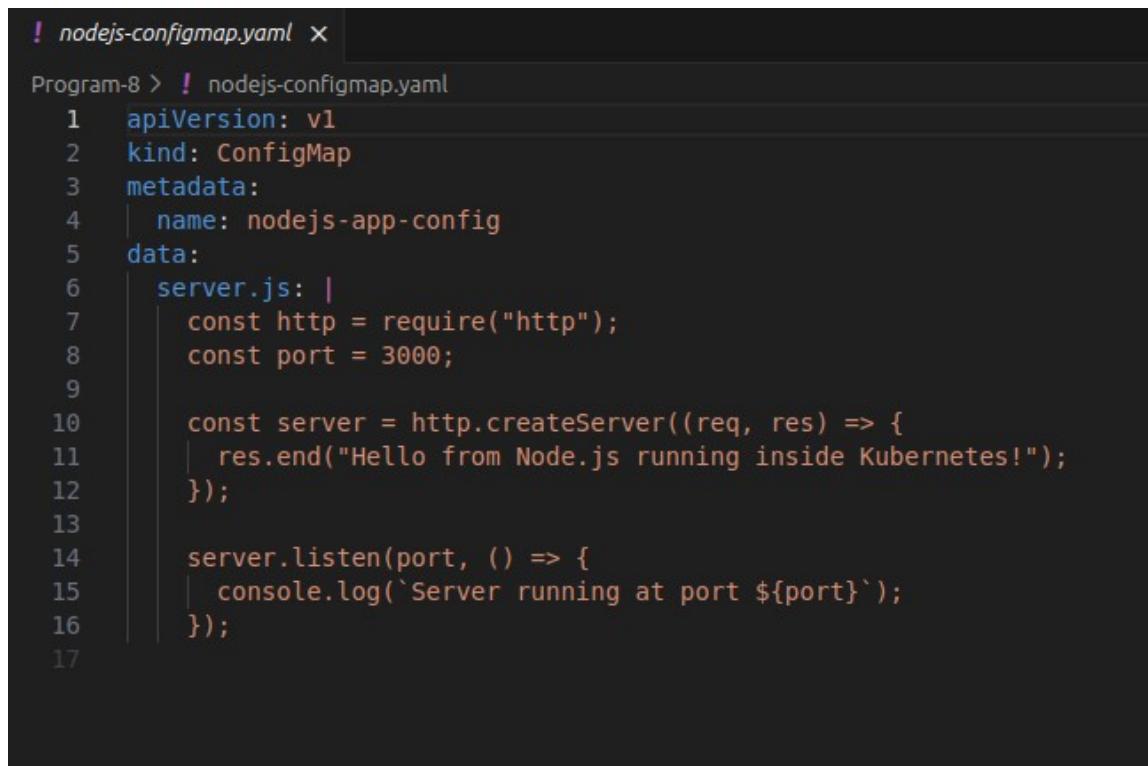
- 1 server.js
2. nodejs-config.yaml
- 3, nodejs-pod.yaml

Server.js:



```
JS server.js  X
Program-8 > JS server.js > ...
1 const http = require("http");
2 const port = 3000;
3
4 const server = http.createServer((req, res) => {
5   res.end("Hello from Node.js running inside Kubernetes!");
6 });
7
8 server.listen(port, () => {
9   console.log(`Server running at port ${port}`);
10 });
11
```

nodejs-config.yaml:



```
! nodejs-configmap.yaml  X
Program-8 > ! nodejs-configmap.yaml
1 apiVersion: v1
2 kind: ConfigMap
3 metadata:
4   name: nodejs-app-config
5 data:
6   server.js: |
7     const http = require("http");
8     const port = 3000;
9
10    const server = http.createServer((req, res) => {
11      res.end("Hello from Node.js running inside Kubernetes!");
12    });
13
14    server.listen(port, () => {
15      console.log(`Server running at port ${port}`);
16    });
17
```

nodejs-pod.yaml

```
! nodejs-pod.yaml ×  
Program-8 > ! nodejs-pod.yaml  
1  apiVersion: v1  
2  kind: Pod  
3  metadata:  
4    name: nodejs-pod  
5    labels:  
6      app: nodejs  
7  spec:  
8    containers:  
9      - name: nodejs  
10        image: node:18  
11        command: ["node", "/usr/src/app/server.js"]  
12  
13        volumeMounts:  
14          - name: app-code  
15            mountPath: /usr/src/app  
16  
17        ports:  
18          - containerPort: 3000  
19  
20    volumes:  
21      - name: app-code  
22        configMap:  
23          name: nodejs-app-config  
24
```

To run this program:

```
# Deploy  
kubectl apply -f nodejs-configmap.yaml  
kubectl app -f nodejs-pod.yaml  
  
# list pods  
kubectl get pods  
  
# expose  
kubectl port-forward-pod/prog-8 3000:3000
```