TASK

This is a simple nodejs app, you just need to do the following:

1- Push it on github so that you can share with us later

2- Dockerize it

3- Setup a minikube in your laptop and run the application .(minikube is just a tool that you can use to host kubernets locally)

4- once you finish it, let us know

==

Setting the things:

Nodejs folder and required file were shared in drive , I just unzip the folder and save it in my local machine.

Pushing Node.js App to github:

<u>Steps</u>

1. Created a repository in github
2. In my local directory where I saved the nodejs folder , I opened it using VS code
3. Following github operations were performed

Command	**Description**
git init	initialized a git repository
git remote add origin https://github.com/Adityakafle/nodejs-containerization.git	Linked my local repository to remote github repository.
git branch -m master main	renamed the default branch from "master" to "main" because it was bringing conflict
git add .	Added all the files to repository
git commit -m “Initial Commit”	Committed the changes made
git push -u origin main	Pushed the code to github

GITHUB LINK: ***https://github.com/Adityakafle/nodejs-containerization.git***

Running Node.js app locally in my machine

Following operations were performed.

Command	**Description**
npm init	Since it is node.js project so I ran this command for creating new package.json file and required information to create it was given.
npm install express	express.js was missing and error was showing so I installed it
npm install ejs	This was also a missing package which was showing error so it is also installed
node app.js	app.js script works fine and I was able to get desired output on port 7777 on localhost
npm start	wrote start script on package.json

Now the app is running on port 7777 on localhost in my machine.

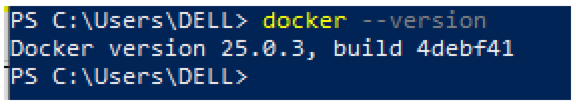
Dockerization of Nodejs Project

Docker Installation:

1. **Downloaded Docker Desktop for windows:**

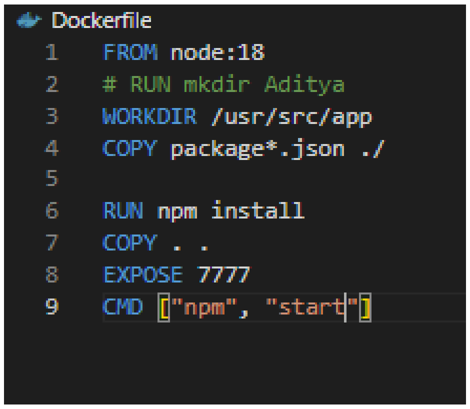
In official Docker website we can easily download docker at one-click, so I downloaded and installed **Docker Desktop Installer.exe** file and followed prompts to complete the installation.

2. **Verify Docker Installation**



```
PS C:\Users\DELL> docker --version
Docker version 25.0.3, build 4debf41
PS C:\Users\DELL>
```

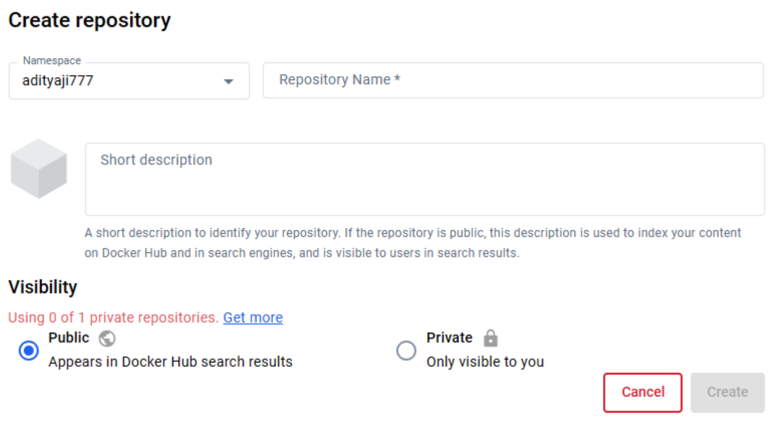
3. **Created Dockerfile:**



Dockerfile

```
FROM node:18
# RUN mkdir Aditya
WORKDIR /usr/src/app
COPY package*.json ./

RUN npm install
COPY . .
EXPOSE 7777
CMD ["npm", "start"]
```

4. **Login to Docker hub and created repository**



Create repository

Namespace: adityaji777

Repository Name *

Short description

A short description to identify your repository. If the repository is public, this description is used to index your content on Docker Hub and in search engines, and is visible to users in search results.

Visibility

Using 0 of 1 private repositories. Get more

[✓ Public] Appears in Docker Hub search results

Private Only visible to you

5. **Docker login :**

- ✓ Docker login command is executed .
- ✓ username and password for docker hub is provided.
- ✓ login is succeeded.

6. **Docker tag**

   Command: docker tag image_id username/repository:tagname

   docker tag my-nodejs-app adityaji7777/my-nodejs-app:latest

7. **Docker push**

   Command: docker push username/my-nodejs-app:tagname

   docker push adityaji777/my-nodejs-app:latest

(Instead of minikube I tried kind as minikube was consuming so much resources)

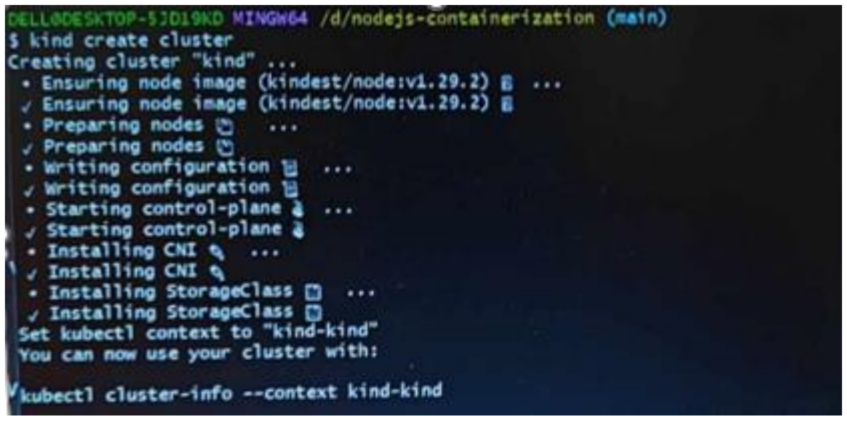
kind Installation:

kind is a tool for running local Kubernetes clusters using Docker container “nodes”.

kind was primarily designed for testing Kubernetes itself, but may be used for local development or CI

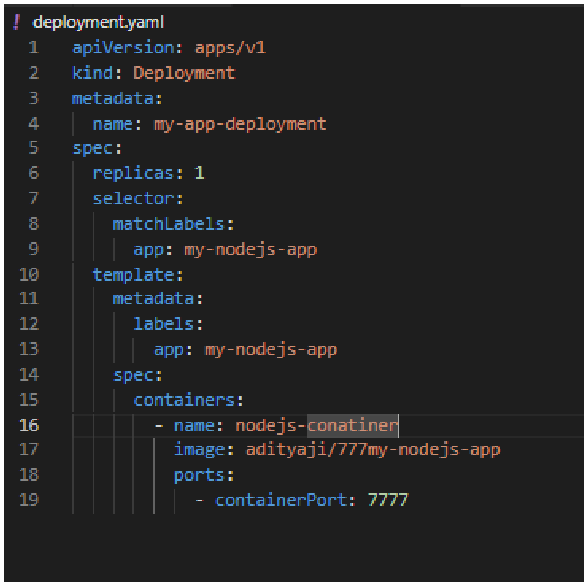
Steps :

- ✓ downloaded go 1.16+
- ✓ command:
  - *go install* sigs.k8s.io/kind@v0.22.0
  - *kind create cluster –name my-nodejs-app-cluster*



```
DELL@DESKTOP-5JD19KD MINGW64 /d/nodejs-containerization (main)
$ kind create cluster
Creating cluster "kind" ...
 • Ensuring node image (kindest/node:v1.29.2) 🖼  ...
 ✓ Ensuring node image (kindest/node:v1.29.2) 🖼
 • Preparing nodes 📦   ...
 ✓ Preparing nodes 📦
 • Writing configuration 📜  ...
 ✓ Writing configuration 📜
 • Starting control-plane 🕹️  ...
 ✓ Starting control-plane 🕹️
 • Installing CNI 🔌  ...
 ✓ Installing CNI 🔌
 • Installing StorageClass 💾  ...
 ✓ Installing StorageClass 💾
Set kubectl context to "kind-kind"
You can now use your cluster with:

kubectl cluster-info --context kind-kind
```

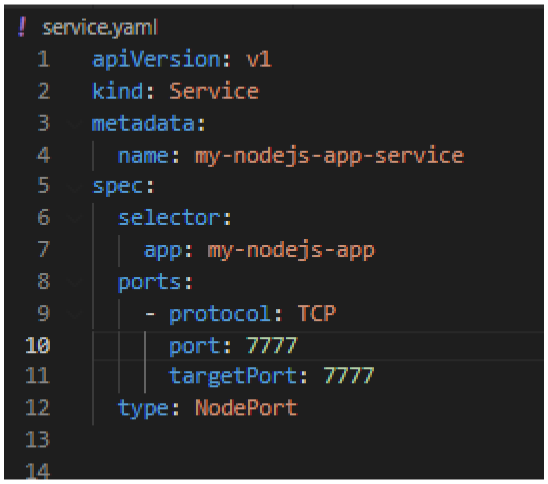
- ✓ Created deployment.yaml file



deployment.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-app-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      app: my-nodejs-app
  template:
    metadata:
      labels:
        app: my-nodejs-app
    spec:
      containers:
        - name: nodejs-conatiner
          image: adityaji/777my-nodejs-app
          ports:
            - containerPort: 7777
```

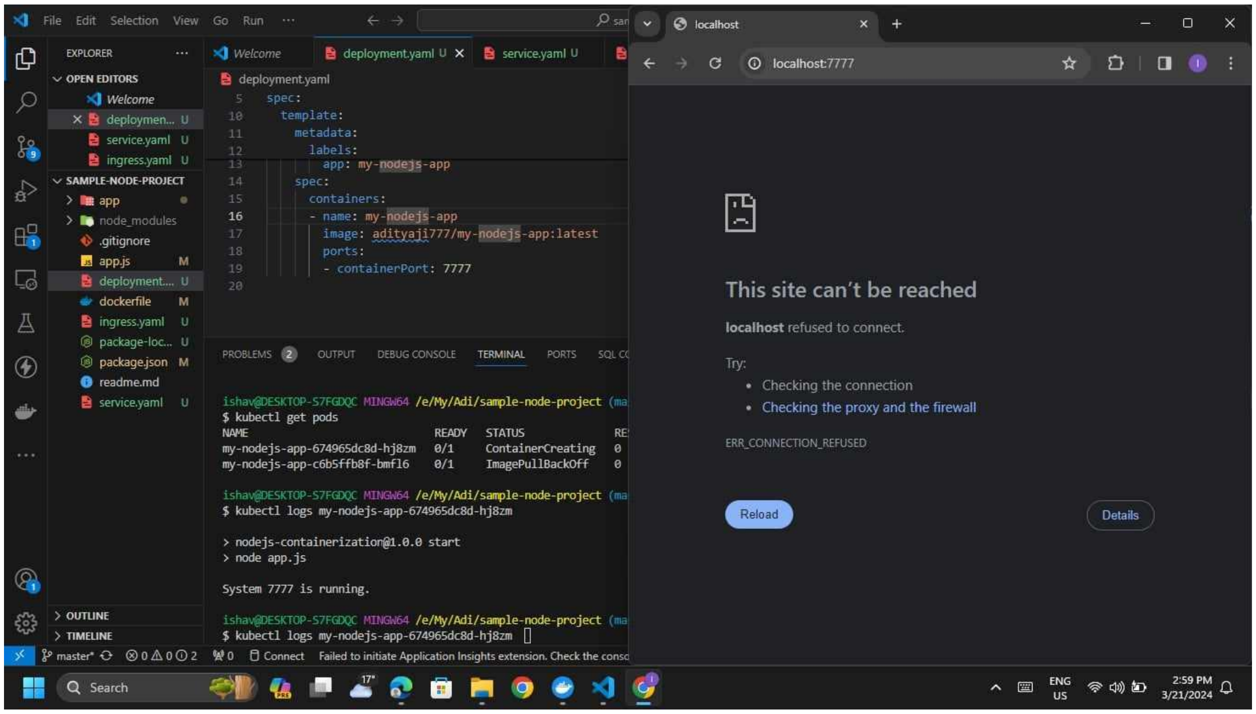
- ✓ Created service.yaml



service.yaml

```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-nodejs-app-service
spec:
  selector:
    app: my-nodejs-app
  ports:
    - protocol: TCP
      port: 7777
      targetPort: 7777
  type: NodePort
```

- ✓ After Creating deployment and service I tried to access using external ip but didn't get it so I forwarded the port to 7777 and accessed it from localhost.



```yaml
spec:
  template:
    metadata:
      labels:
        app: my-nodejs-app
    spec:
      containers:
      - name: my-nodejs-app
        image: adityaji777/my-nodejs-app:latest
        ports:
        - containerPort: 7777
```

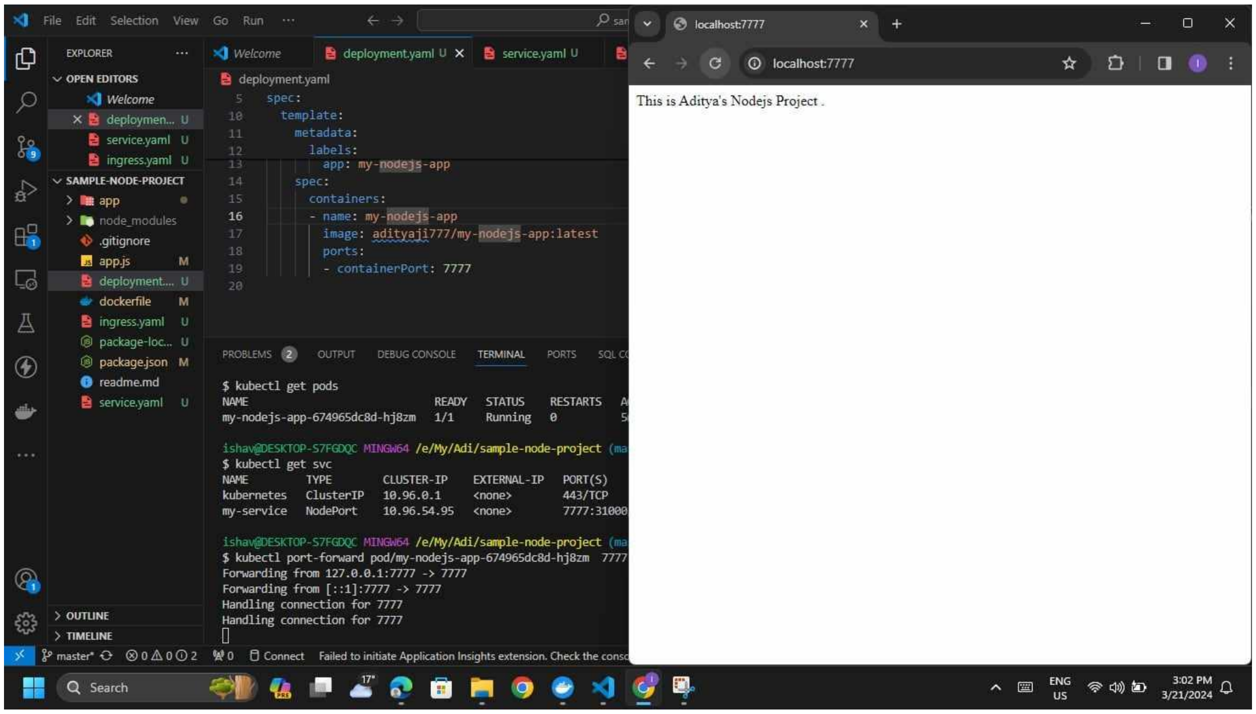
```
ishav@DESKTOP-S7FGDQC MINGW64 /e/My/Adi/sample-node-project (ma
$ kubectl get pods
NAME                             READY   STATUS              RE
my-nodejs-app-674965dc8d-hj8zm   0/1     ContainerCreating   0
my-nodejs-app-c6b5ffb8f-bmfl6    0/1     ImagePullBackOff    0

ishav@DESKTOP-S7FGDQC MINGW64 /e/My/Adi/sample-node-project (ma
$ kubectl logs my-nodejs-app-674965dc8d-hj8zm

> nodejs-containerization@1.0.0 start
> node app.js

System 7777 is running.

ishav@DESKTOP-S7FGDQC MINGW64 /e/My/Adi/sample-node-project (ma
$ kubectl logs my-nodejs-app-674965dc8d-hj8zm
```

This site can't be reached

localhost refused to connect.

Try:
- Checking the connection
- Checking the proxy and the firewall

ERR_CONNECTION_REFUSED



```
$ kubectl get pods
NAME                             READY   STATUS    RESTARTS   A
my-nodejs-app-674965dc8d-hj8zm   1/1     Running   0          5

ishav@DESKTOP-S7FGDQC MINGW64 /e/My/Adi/sample-node-project (ma
$ kubectl get svc
NAME         TYPE        CLUSTER-IP    EXTERNAL-IP   PORT(S)
kubernetes   ClusterIP   10.96.0.1     <none>        443/TCP
my-service   NodePort    10.96.54.95   <none>        7777:31000

ishav@DESKTOP-S7FGDQC MINGW64 /e/My/Adi/sample-node-project (ma
$ kubectl port-forward pod/my-nodejs-app-674965dc8d-hj8zm  7777
Forwarding from 127.0.0.1:7777 -> 7777
Forwarding from [::1]:7777 -> 7777
Handling connection for 7777
Handling connection for 7777
```

This is Aditya's Nodejs Project .

Now the app is running on pod and can be accessible from localhost.