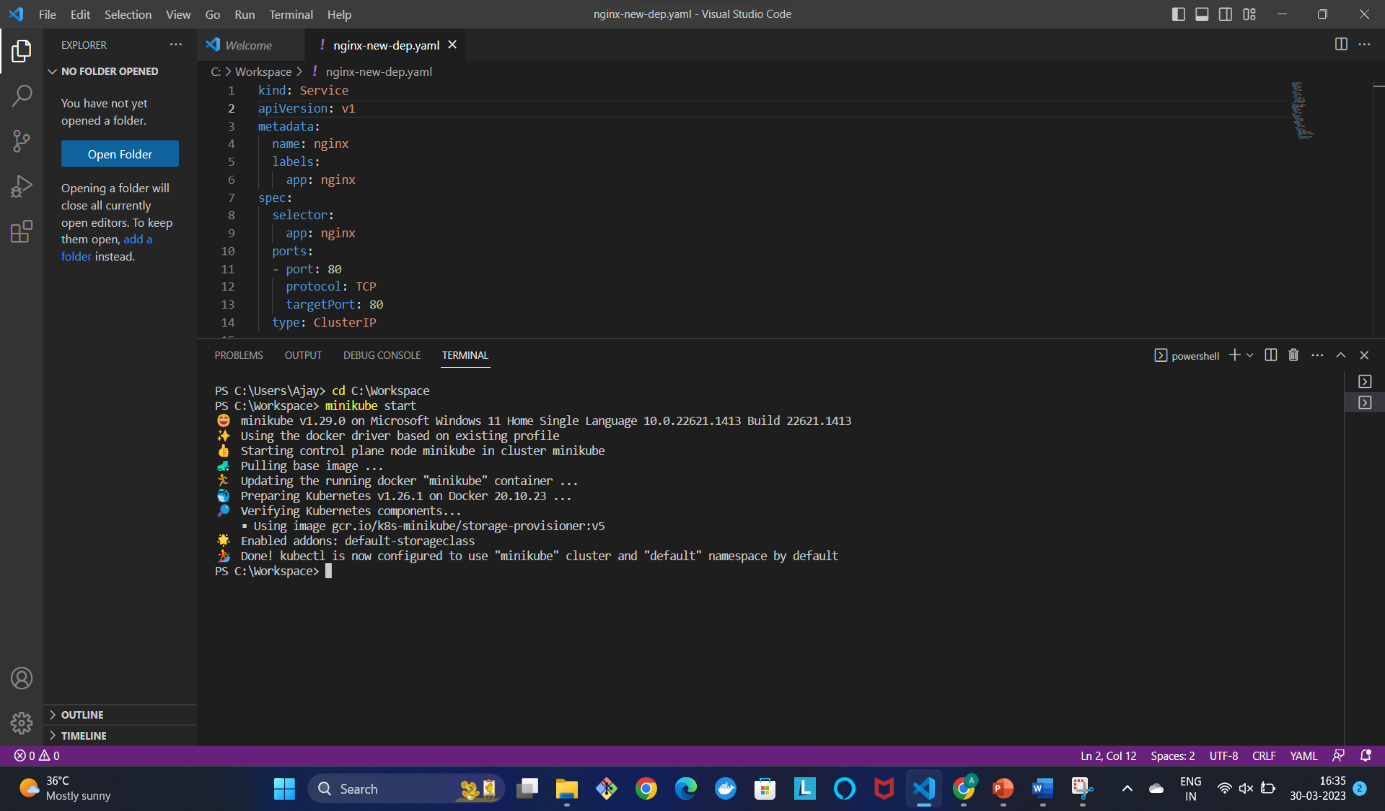
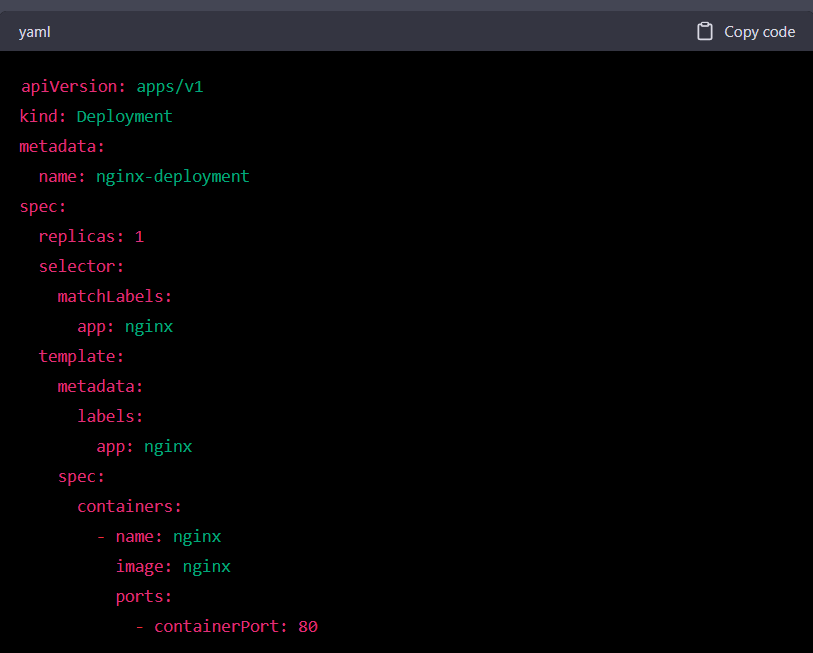
**Deploying Nginx in Minikube and modifying a Html page**

* First, we need to start minikube

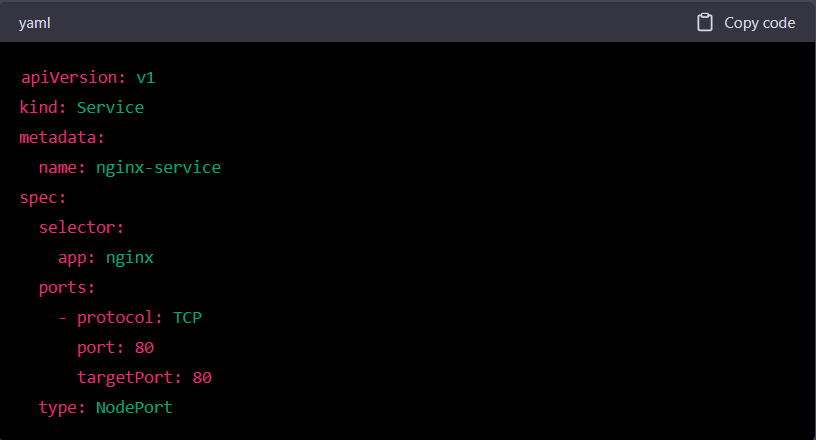
minikube start



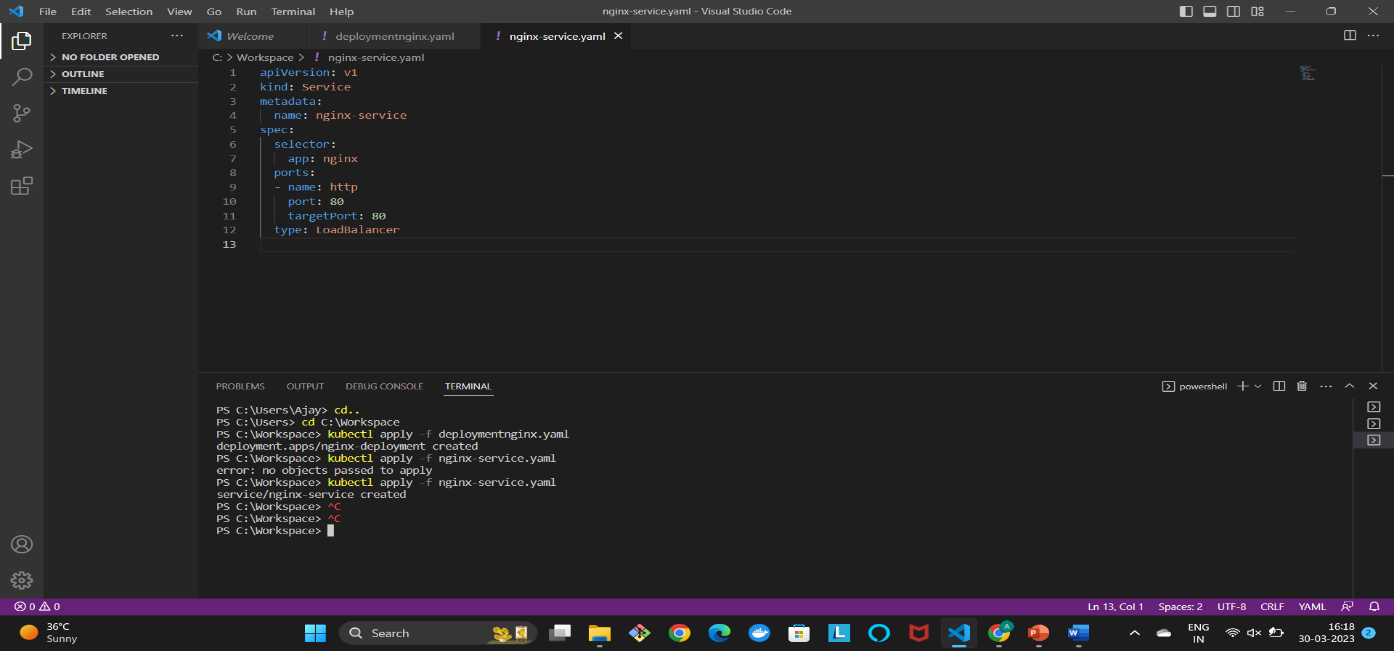
* Next, we need to create a new deployment file with a random name (ex: nginx-deployment) file in Visual Studio and save it in a local workspace.



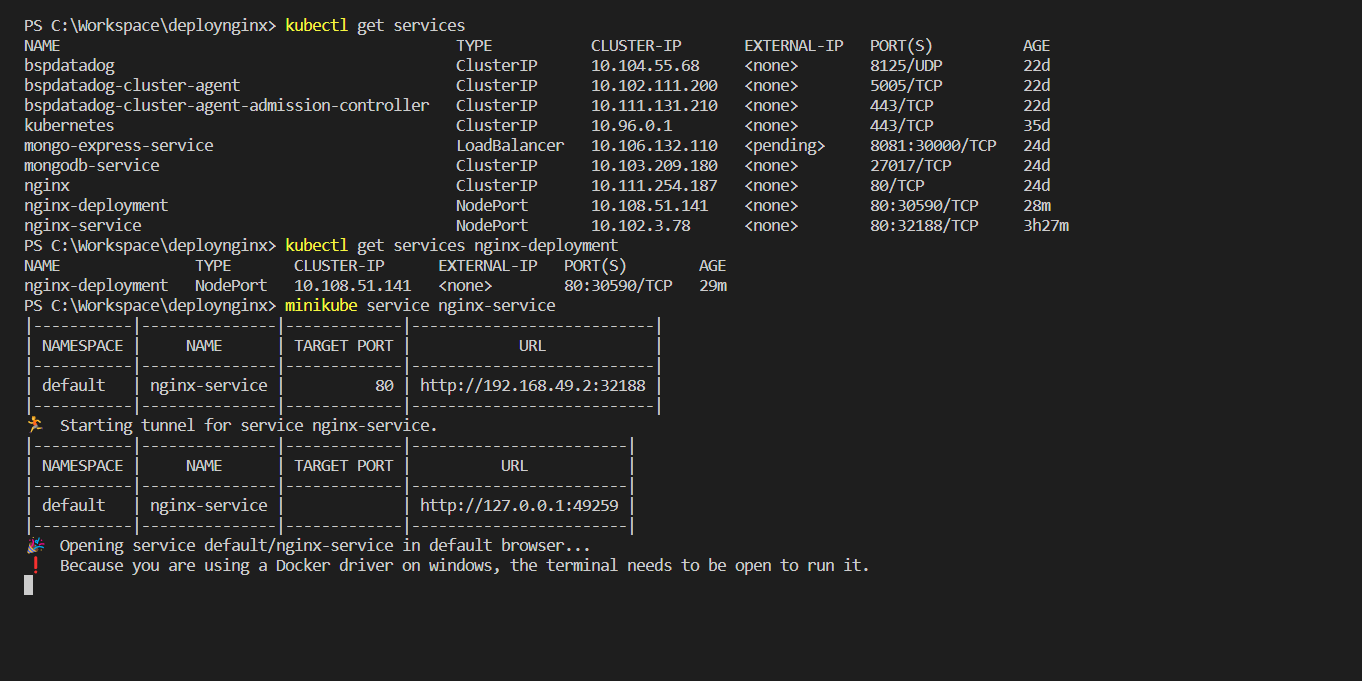
* Next, we need to create a new (ex: nginx-service.yaml) file with a random name and save it in our local with the same folder where the previously created nginx-deployment.yaml file is saved.

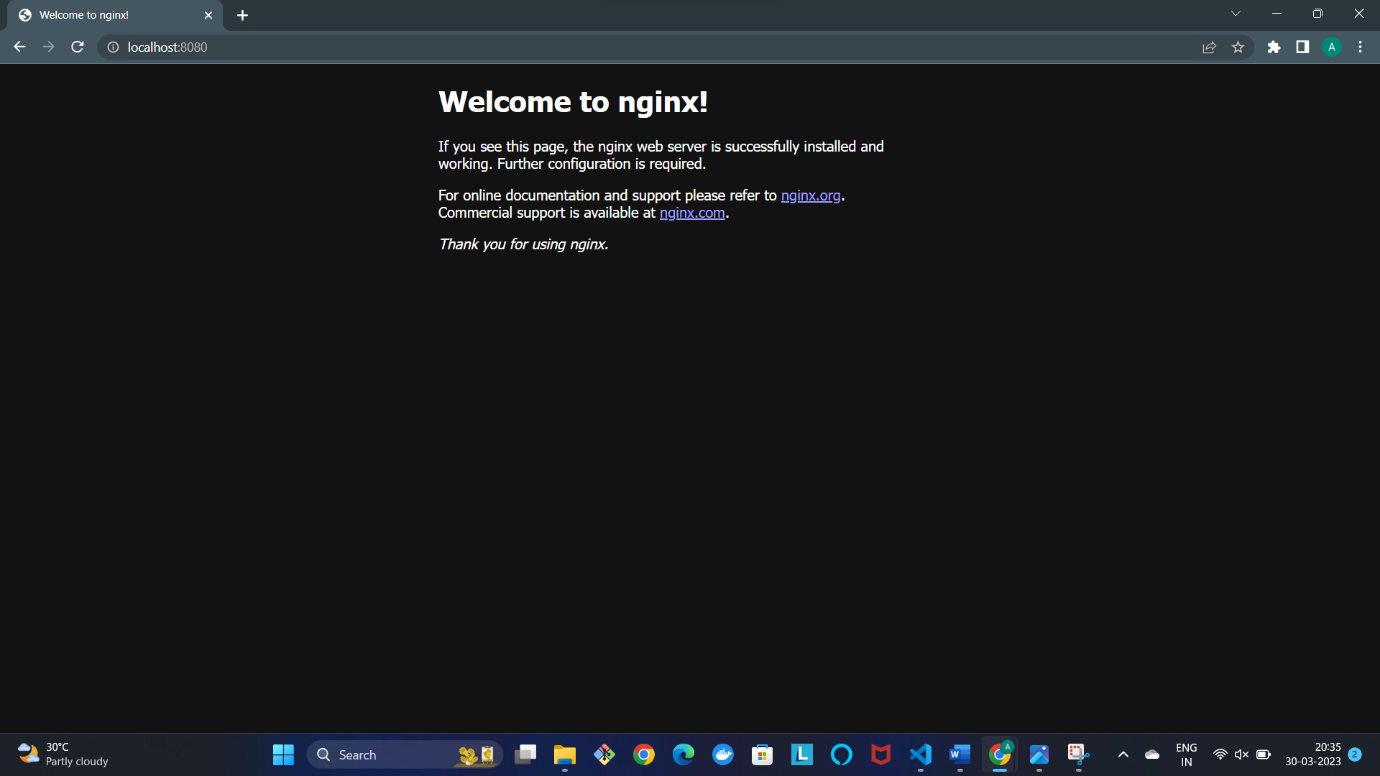


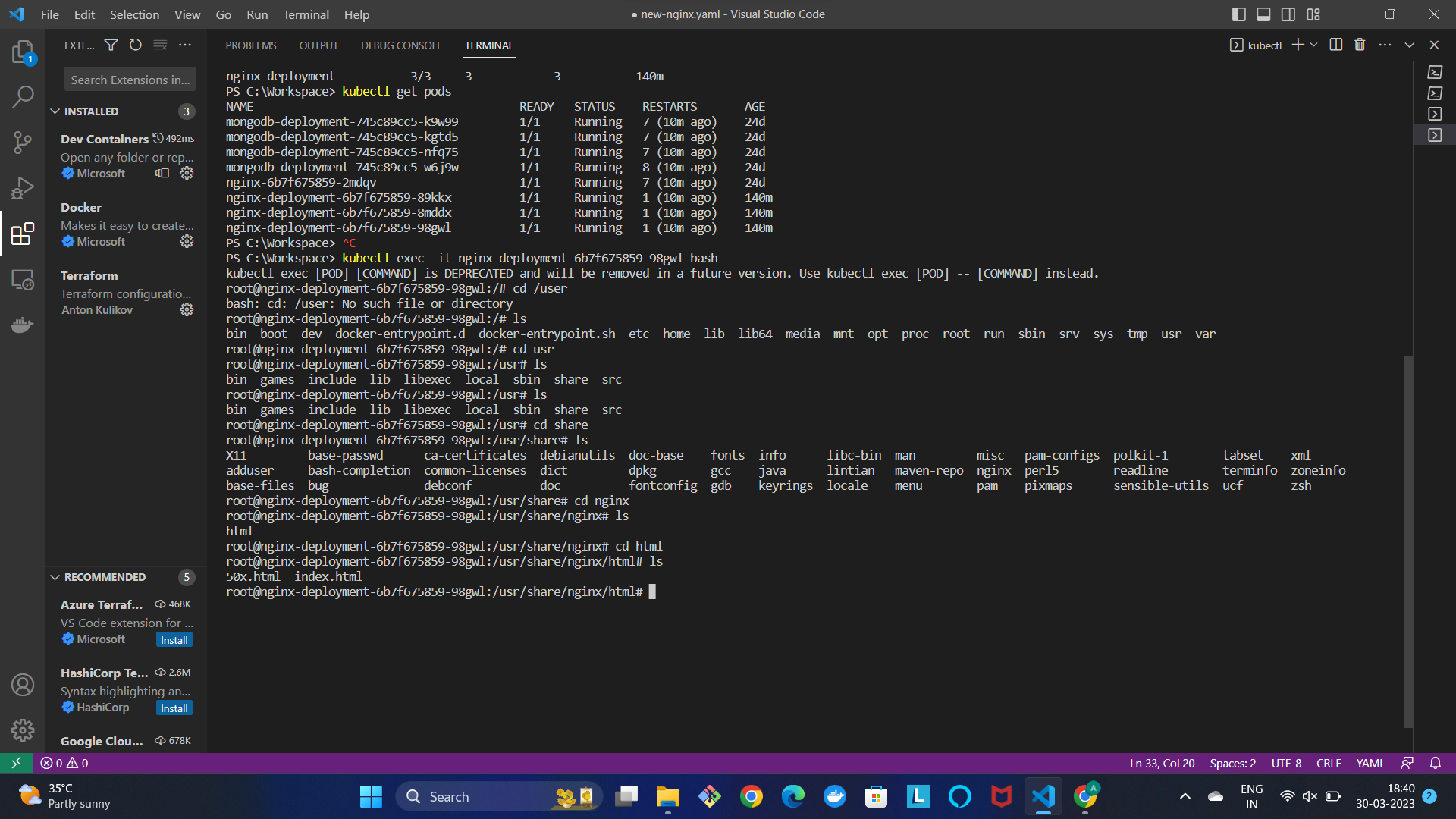
* Now, we need to open the terminal and change to the directory where these deployment and service . yaml files are stored
  + Cd C:\Workspace\deploynginx (Example), this command will change the working directory
* Now, we need to apply these files with the below commands
  + **kubectl apply -f nginx-deployment.yaml**
  + **kubectl apply -f nginx-service.yaml**



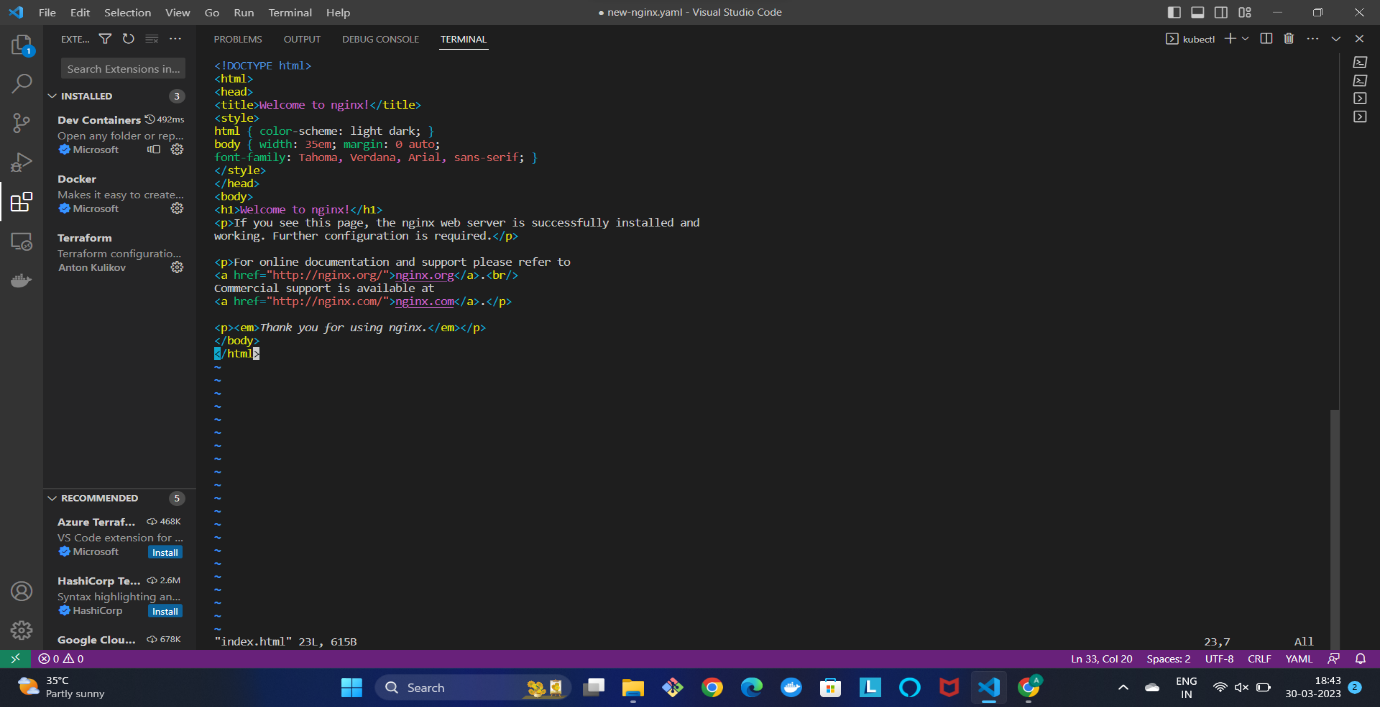
* After applying we need to run a command
  + kubectl get services (This will get all the available services)
  + We need to check the service we created and execute the below command



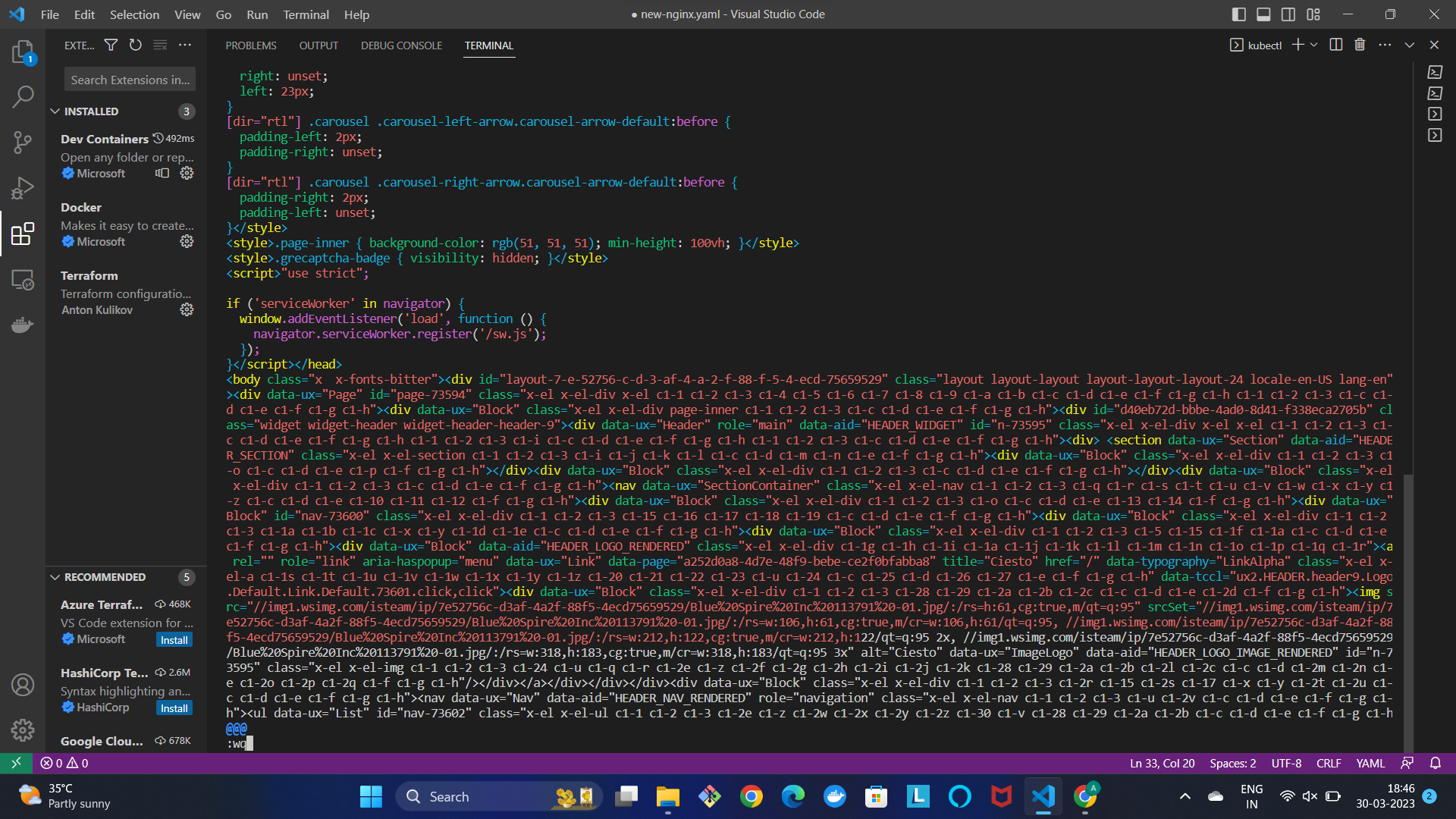
* Next, we need to execute the command
  + minikube service nginx-service(Service name)
* Next, we need to execute the command
  + [http://localhost:8080](http://localhost:8080/)
  + Then we can see the nginx home page
* Now we need to modify the nginx home html page by following below steps
  + Kubectl exec -it podname bash
  + Cd usr/share/nginx/html
  + Vi index.html



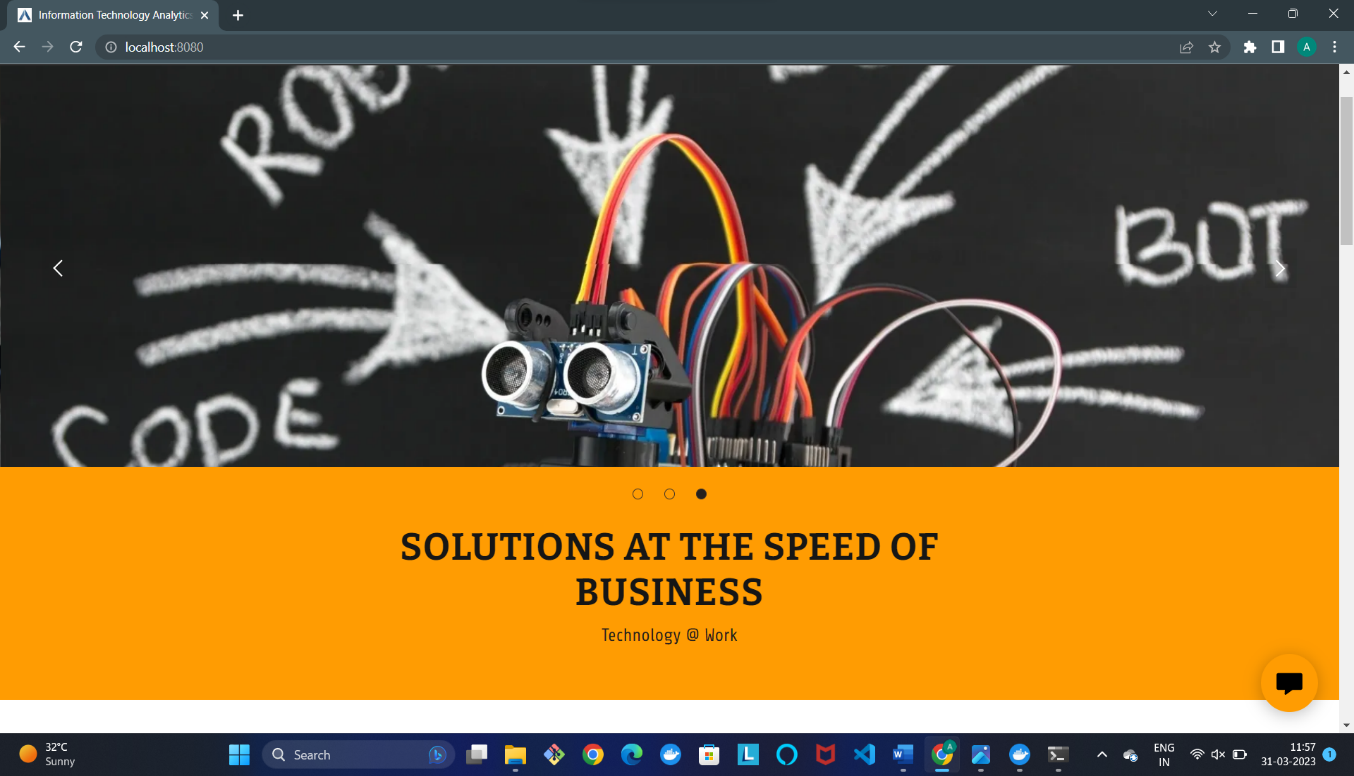
* After the command vi index.html, we can see a page as shown below



* We can remove the entire code and modify whatever code we want to deploy using steps given below
  + Press “Insert” Key
  + Modify the changes
  + Press “ESC” key, “:wq”(write and quit)
* If there is any issue in this command vi index.html, try below commands
  + apt update
  + apt install vim
  + vi index.html
* Inserted a sample html page as shown below



* Now open chrome and search, locolhost:8080



* Now, we can see our desired html page in localhost:8080

**Redis project deploying Guestbook**

**Objectives:**

* Create a namespace
* Start up a Redis leader
* Start up two Redis followers
* Start up the guestbook frontend
* Expose and view the frontend service
* Clean Up
* Creating a Namespace
  + Kubectl create ns guestbook
* Start up a Redis leader and its service
  + Create a redis deployment file with a random name (redis-deployment.yaml)

apiVersion: apps/v1

kind: Deployment

metadata:

name: guestbook

spec:

replicas: 1

selector:

matchLabels:

app: guestbook

template:

metadata:

labels:

app: guestbook

spec:

containers:

- name: guestbook

image: guestbook:latest

ports:

- containerPort: 5000

env:

- name: REDIS\_HOST

value: "redis.default.svc.cluster.local"

- name: REDIS\_PORT

value: "6379"

* To apply this redis deployment file, use the below command
  + kubectl plan -n guestbook -f redis-leader-deployment.yaml
  + kubectl apply -n guestbook -f redis-leader-deployment.yaml
* Next, we need to create the Redis leader service with a name (redis-leader-service.yaml)

apiVersion: v1

kind: Service

metadata:

name: redis-leader

labels:

app: redis

role: leader

tier: backend

spec:

ports:

- port: 6379

targetPort: 6379

selector:

app: redis

role: leader

tier: backend

* Now we need to apply the service
  + kubectl plan -n guestbook -f leader-service.yaml
  + kubectl apply -n guestbook -f leader-service.yaml

**Setting up redis follower**:

apiVersion: apps/v1

kind: Deployment

metadata:

name: redis-follower

labels:

app: redis

role: follower

tier: backend

spec:

replicas: 2

selector:

matchLabels:

app: redis

template:

metadata:

labels:

app: redis

role: follower

tier: backend

spec:

containers:

- name: follower

image: gcr.io/google\_samples/gb-redis-follower:v2

resources:

requests:

cpu: 100m

memory: 100Mi

ports:

- containerPort: 6379

* Apply redis deployment from the following redis follower file
  + kubectl plan -n guestbook -f redis-follower-deployment.yaml
  + kubectl apply -n guestbook -f redis-follower-deployment.yaml
* **Now, we need to create the redis follower service**

apiVersion: v1

kind: Service

metadata:

name: redis-follower

labels:

app: redis

role: follower

tier: backend

spec:

ports:

# the port that this service should serve on

- port: 6379

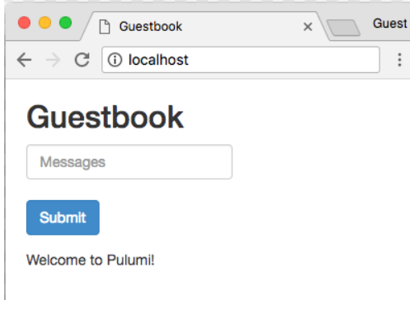
selector:

app: redis

role: follower

tier: backend

* We need to plan and apply redis follower service:
  + Kubectl plan -n guestbook -f redis-follower-service.yaml
  + Kubectl apply -n guestbook -f redis-follower-service.yam
* **Expose and view the frontend:**
  + minikube service -n guestbook

****

* we can see this web page automatically opened in our localhost