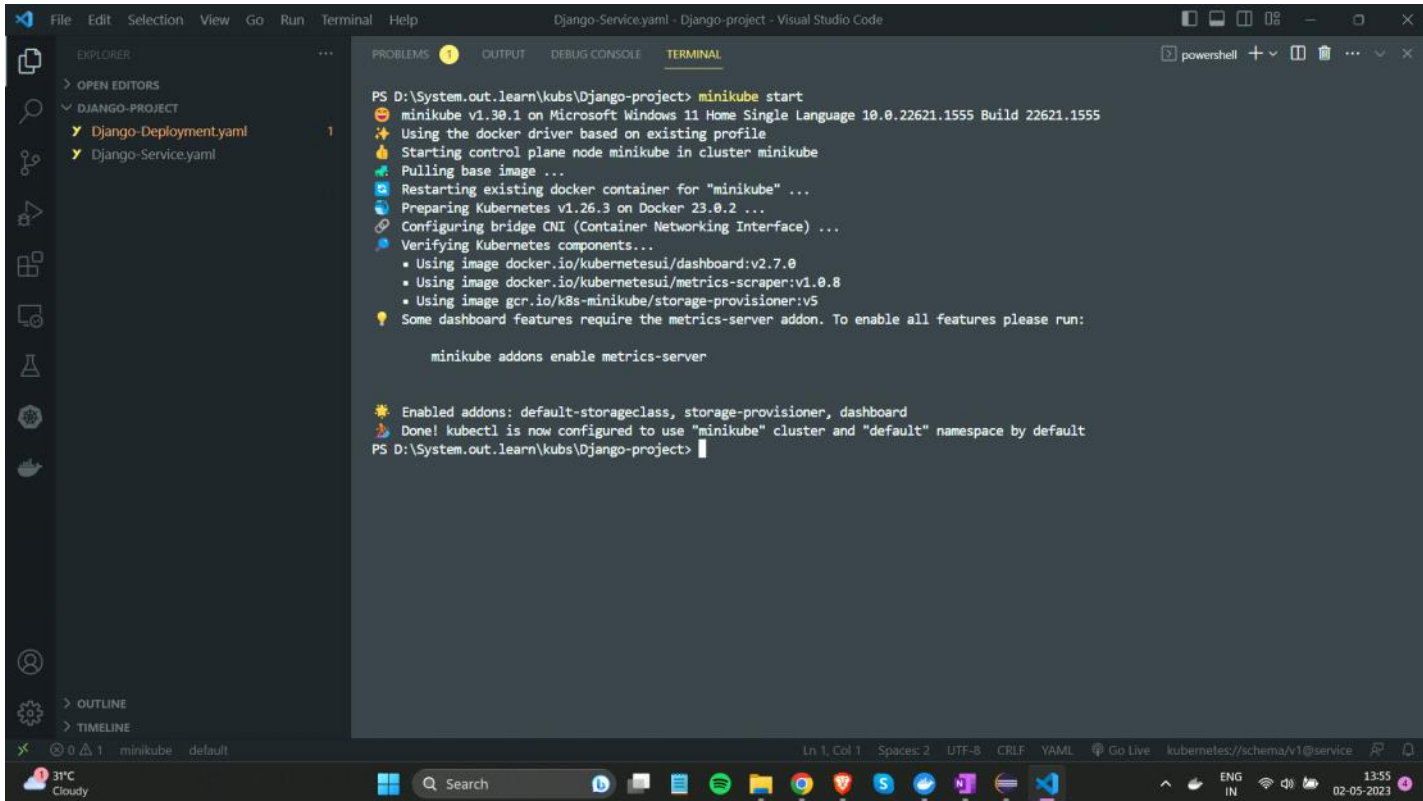


## Scaling up an application :

### 1. Starting minikube :



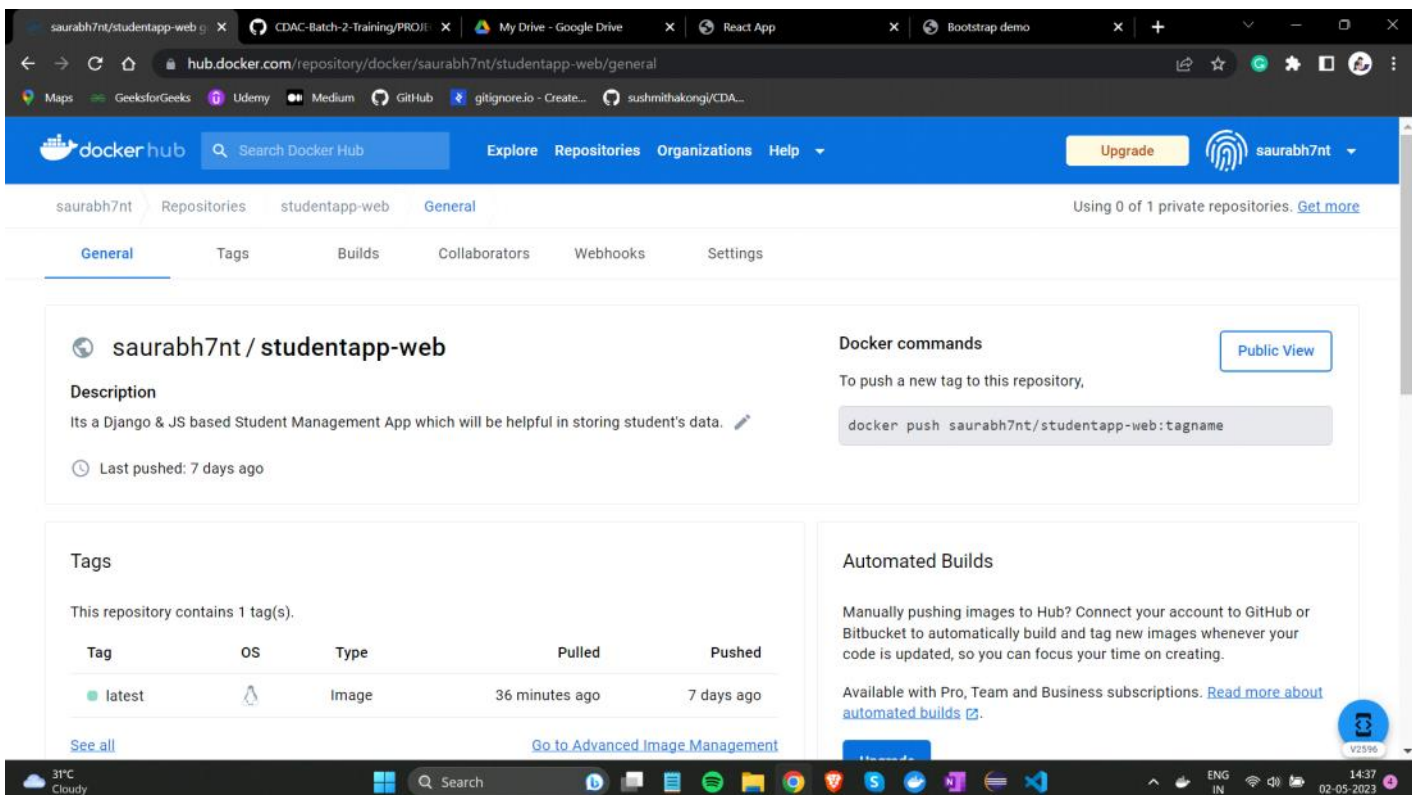
```

PS D:\System.out.learn\kubs\Django-project> minikube start
minikube v1.30.1 on Microsoft Windows 11 Home Single Language 10.0.22621.1555 Build 22621.1555
Using the docker driver based on existing profile
Starting control plane node minikube in cluster minikube
Pulling base image ...
Restarting existing docker container for "minikube" ...
Preparing Kubernetes v1.26.3 on Docker 23.0.2 ...
Configuring bridge CNI (Container Networking Interface) ...
Verifying Kubernetes components...
  Using image docker.io/kubernetes/dashboard:v2.7.0
  Using image docker.io/kubernetes/metrics-scraper:v1.0.8
  Using image gcr.io/k8s-minikube/storage-provisioner:v5
Some dashboard features require the metrics-server addon. To enable all features please run:

    minikube addons enable metrics-server

Enabled addons: default-storageclass, storage-provisioner, dashboard
Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
PS D:\System.out.learn\kubs\Django-project>
  
```

## Django project which is already deployed on docker-hub repository :



**saurabh7nt / studentapp-web**

**Description**  
It's a Django & JS based Student Management App which will be helpful in storing student's data. [Edit description](#)

Last pushed: 7 days ago

**Tags**  
This repository contains 1 tag(s).

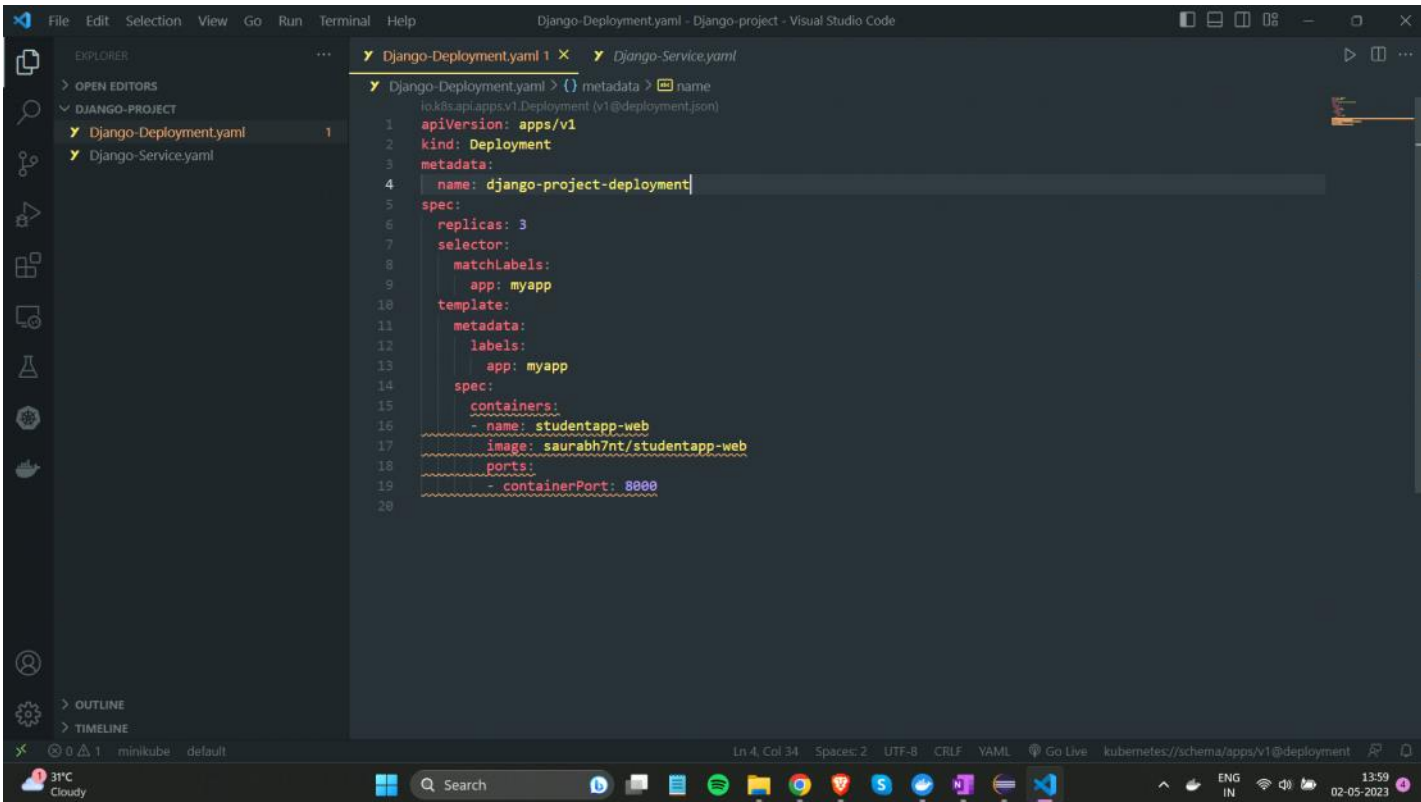
Tag	OS	Type	Pulled	Pushed
latest	linux	Image	36 minutes ago	7 days ago

[See all](#) [Go to Advanced Image Management](#)

**Docker commands**  
To push a new tag to this repository,  
`docker push saurabh7nt/studentapp-web:tagname`

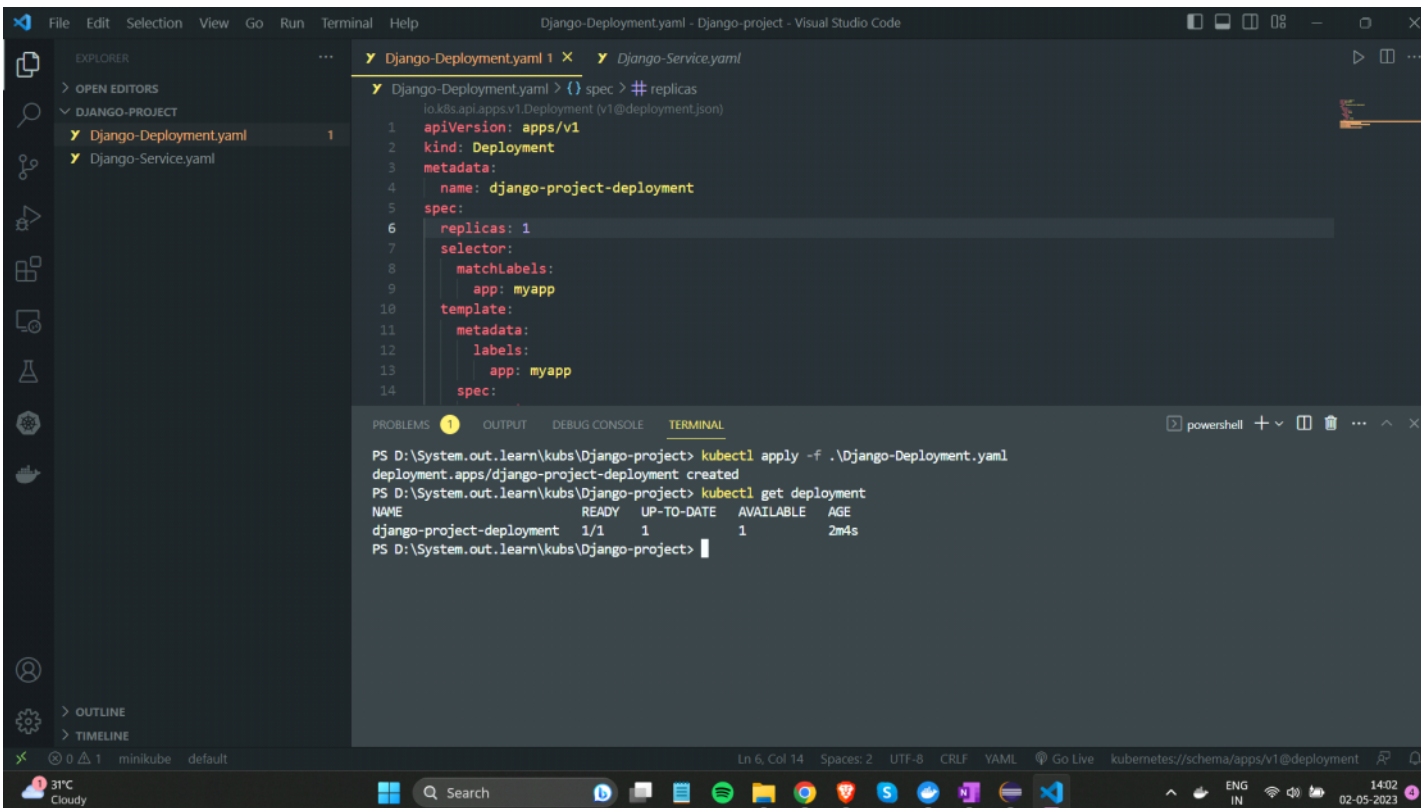
**Automated Builds**  
Manually pushing images to Hub? Connect your account to GitHub or Bitbucket to automatically build and tag new images whenever your code is updated, so you can focus your time on creating.  
Available with Pro, Team and Business subscriptions. [Read more about automated builds](#)

## 2. Deployment yaml file:



```
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: django-project-deployment
5  spec:
6    replicas: 3
7    selector:
8      matchLabels:
9        app: myapp
10   template:
11     metadata:
12       labels:
13         app: myapp
14     spec:
15       containers:
16         - name: studentapp-web
17           image: saurabh7nt/studentapp-web
18           ports:
19             - containerPort: 8000
```

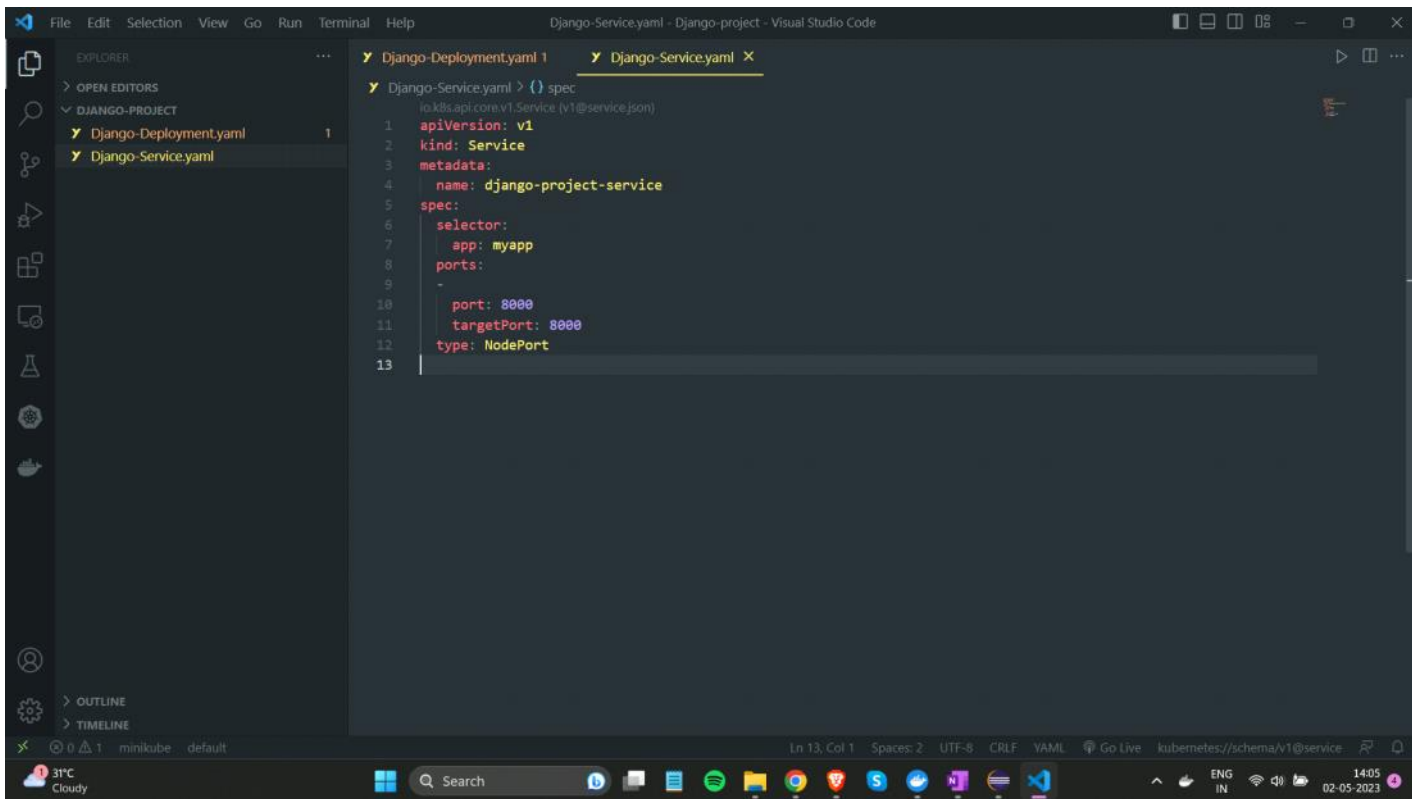
## 3. Creating a deployment by using yaml file:



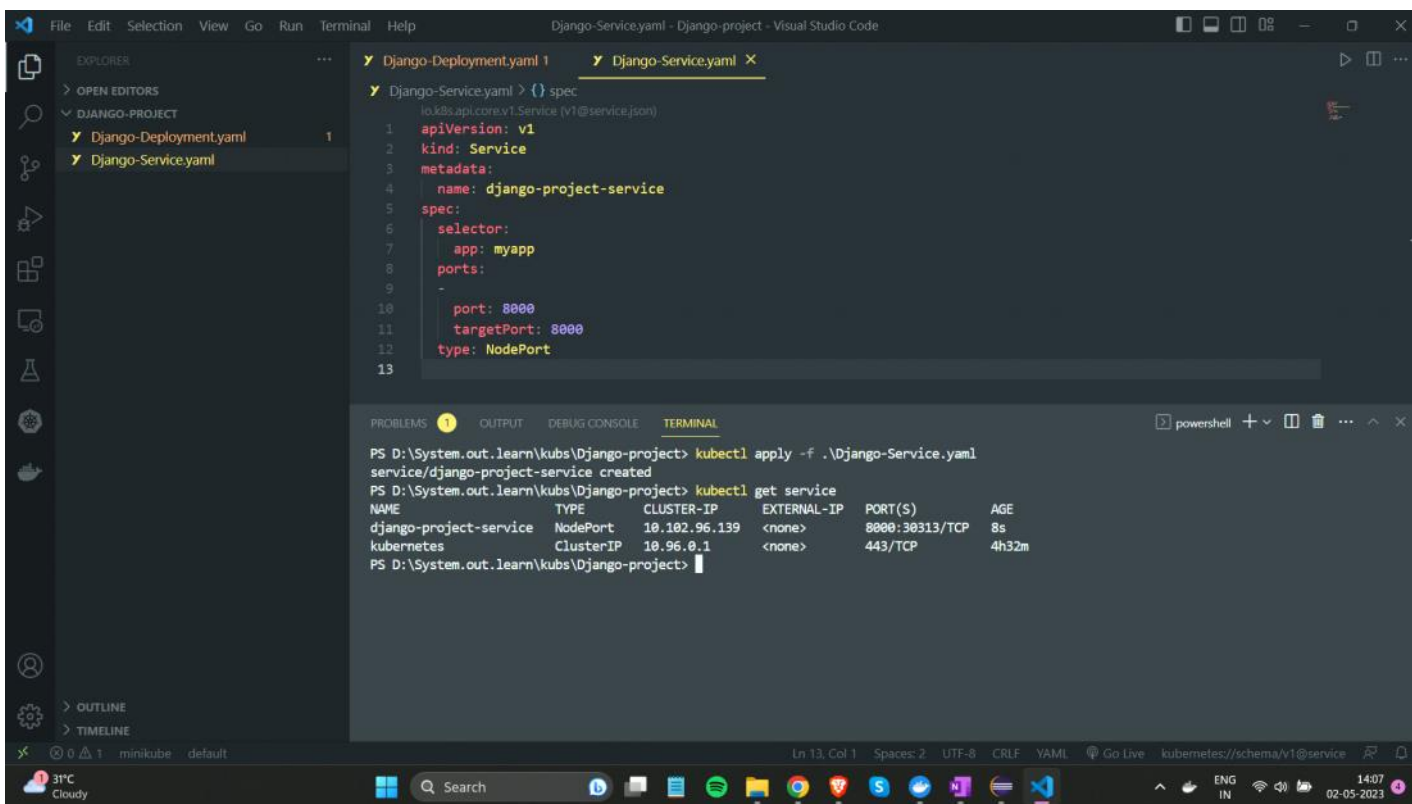
```
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: django-project-deployment
5  spec:
6    replicas: 1
7    selector:
8      matchLabels:
9        app: myapp
10   template:
11     metadata:
12       labels:
13         app: myapp
14     spec:
```

PS D:\System.out.learn\kubs\Django-project> kubectl apply -f .\Django-Deployment.yaml  
deployment.apps/django-project-deployment created  
PS D:\System.out.learn\kubs\Django-project> kubectl get deployment  
NAME READY UP-TO-DATE AVAILABLE AGE  
django-project-deployment 1/1 1 1 2m4s  
PS D:\System.out.learn\kubs\Django-project>

## 4. Service yaml file :



## 5. Creating a service by using yaml file:



## 6. Accessing newly created service

The screenshot shows the Visual Studio Code interface with the terminal open. The terminal displays the output of the following commands:

```
PS D:\System.out.learn\kubs\Django-project> kubectl get service
NAME                                TYPE           CLUSTER-IP      EXTERNAL-IP      PORT(S)          AGE
django-project-service             NodePort        10.102.0.139    <none>            8000:30313/TCP   109s
kubernetes                         ClusterIP       10.96.0.1       <none>            443/TCP          4h34m

PS D:\System.out.learn\kubs\Django-project> minikube service django-project-service
|-----|
| NAMESPACE | NAME             | TARGET PORT | URL               |
|-----|
| default   | django-project-service | 8000        | http://192.168.49.2:30313 |
|-----|
| * Starting tunnel for service django-project-service.
|-----|
| NAMESPACE | NAME             | TARGET PORT | URL               |
|-----|
| default   | django-project-service | 8000        | http://127.0.0.1:57052 |
|-----|
| Opening service default/django-project-service in default browser...
| Because you are using a Docker driver on windows, the terminal needs to be open to run it.
```





Student Management   Home   Add Student   Show Students  

## Student Management App

Student Name :

Roll no. :

Std :

Age :

Email :

Address :

Student Management   Home   Add Student   Show Students  

## Student Management App

### List of Students :

#SNO.	NAME	ROLL ID	EMAIL	STD	AGE	ADDRESS	ACTION
1	Saurabh Tajane	1	saurabhtajne07@gmail.com	9	13	Chandrapur	<input type="button" value="Delete"/>
2	Virat	2	Virat@gmail.com	9	14	Delhi	<input type="button" value="Delete"/>
3	Christiano Ronaldo	3	chris@gmail.com	9	14	Portugal	<input type="button" value="Delete"/>
4	Walter White	4	white@gmail.com	9	14	NY	<input type="button" value="Delete"/>
5	Jack Sparrow	5	sparrow@gmail.com	12	17	Oceans	<input type="button" value="Delete"/>

7. Scaling up an application :

Setting the replicas=5

This screenshot shows the Visual Studio Code editor with a file named `Django-Deployment.yaml` open. The file contains a Kubernetes Deployment manifest for a Django project. The manifest specifies the API version, kind, metadata, and a spec with replicas, selector, template, and containers.

```
1 apiVersion: apps/v1
2 kind: Deployment
3 metadata:
4   name: django-project-deployment
5 spec:
6   replicas: 5
7   selector:
8     matchLabels:
9       app: myapp
10  template:
11    metadata:
12      labels:
13        app: myapp
14    spec:
15      containers:
16      - name: studentapp-web
17        image: saurabh7nt/studentapp-web
18        ports:
19        - containerPort: 8000
20
```

This screenshot shows the Visual Studio Code editor with the `TERMINAL` panel open. It displays the output of two `kubectl` commands executed in a PowerShell terminal. The first command is `kubectl apply -f .\Django-Deployment.yaml`, which successfully creates the deployment. The second command is `kubectl get pods`, which shows the status of the pods.

```
PS D:\System.out.learn\kubs\Django-project> kubectl apply -f .\Django-Deployment.yaml
deployment.apps/django-project-deployment configured
PS D:\System.out.learn\kubs\Django-project> kubectl get pods
NAME                                READY   STATUS             RESTARTS   AGE
django-project-deployment-57cd9d767f-d7hp2    0/1     ContainerCreating   0           12s
django-project-deployment-57cd9d767f-lpzw8    1/1     Running             0           40m
django-project-deployment-57cd9d767f-sz7t9    0/1     ContainerCreating   0           12s
django-project-deployment-57cd9d767f-t4p5z    0/1     ContainerCreating   0           12s
django-project-deployment-57cd9d767f-v9fbb    1/1     Running             0           12s
PS D:\System.out.learn\kubs\Django-project> kubectl get pods
NAME                                READY   STATUS             RESTARTS   AGE
django-project-deployment-57cd9d767f-d7hp2    1/1     Running             0           23s
django-project-deployment-57cd9d767f-lpzw8    1/1     Running             0           41m
django-project-deployment-57cd9d767f-sz7t9    1/1     Running             0           23s
django-project-deployment-57cd9d767f-t4p5z    1/1     Running             0           23s
django-project-deployment-57cd9d767f-v9fbb    1/1     Running             0           23s
PS D:\System.out.learn\kubs\Django-project>
```