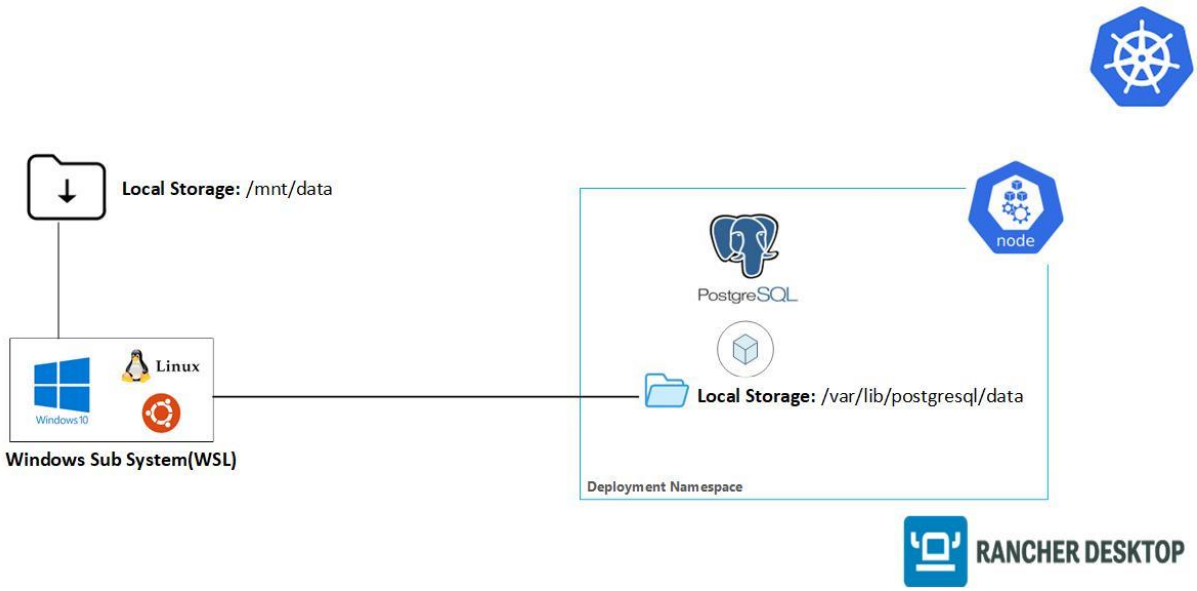
Steps to launch a PostgreSQL Database in a Kubernetes cluster and store its data in a persistent volume.

Note: *This configuration is intended only for Development,testing applications in a local Kubernetes Cluster(hosted in Windows using Rancher Dekstop and WSL)*

Download Source Code from here



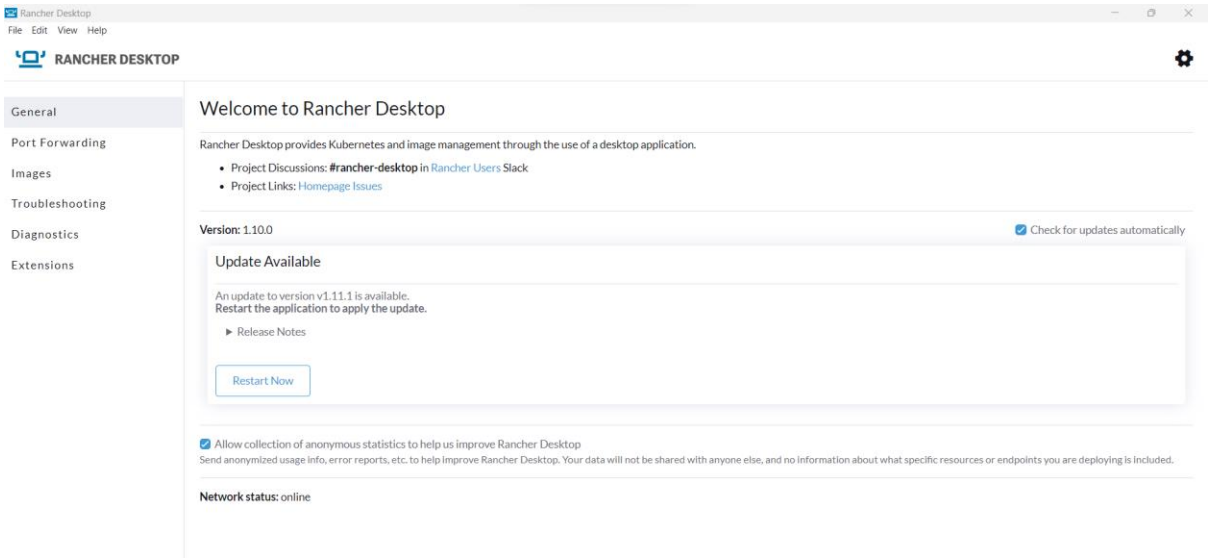
You can follow the steps outlined below.

Step 1: Install Rancher Desktop

- Visit the official Rancher Desktop GitHub releases page: Rancher Desktop Releases.
- Download the appropriate installer for your operating system (e.g., .exe for Windows).
- Execute the downloaded installer to start the installation process.
- Follow the on-screen instructions to complete the installation.

Step 2: Start Rancher Desktop

- Launch Rancher Desktop and ensure that it is running. This will start a local Kubernetes cluster.



Step 3: Enable Windows Subsystem for Linux

- Ensure that your system meets the requirements for WSL 2. It requires Windows 10 version 1903 or higher with Build 18362 or higher.
- Ensure that virtualization is enabled in your computer's BIOS settings. WSL 2 relies on Hyper-V, which requires virtualization support.
- Open PowerShell as Administrator and run the following command:
  dism.exe /online /enable-feature /featurename:Microsoft-Windows-Subsystem-Linux /all /norestart
- Run the following command in PowerShell:
  dism.exe /online /enable-feature /featurename:VirtualMachinePlatform /all /norestart
- Restart your computer to apply the changes.
- Download the WSL 2 Linux Kernel update package from the official Microsoft website.
- Open PowerShell and run the following command to set the WSL default version to 2:
  wsl --set-default-version 2

Step 3: Create a Kubernetes Secrets for (PostgreSQL DB) YAML file

kubectl create secret --from-literal= POSTGRES_USER=postgres --from-literal= POSTGRES_DB=postgres –from-literal= POSTGRES_PASSWORD=pass –dry-run=client -o yaml > postgresql-secret.yml

postgresql-secret.yml

```
apiVersion: v1
data:
  POSTGRES_DB: cG9zdGdyZXM=
  POSTGRES_PASSWORD: cGFzcw==
  POSTGRES_USER: cG9zdGdyZXM=
kind: Secret
```

```
metadata:
  name: psql-db-secrets
  labels:
    app: postgres-sql
```

Step 4: Create a Persistent Volume (PV) YAML file

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: psql-db-pv
  namespace: default
  labels:
    app: postgres-sql
spec:
  storageClassName: local-storage
  capacity:
    storage: 1Gi
  volumeMode: Filesystem
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Retain
  local:
    path: /mnt/c/data
  nodeAffinity:
    required:
      nodeSelectorTerms:
        - matchExpressions:
            - key: kubernetes.io/os
              operator: In
              values:
              - linux
```

We use local storage to store data hence use storageClassName as local-storage and make sure in C drive *"data"(/mnt/c/data = C:\data)* folder exists before running this file.

Step 5: Create a Persistent Volume Claim(PVC) YAML file

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: psql-db-pvc
  labels:
    app: postgres-sql
spec:
  selector:
```

```
    matchLabels:
      app: postgres-sql
  storageClassName: local-storage
  resources:
    requests:
      storage: 1Gi
  accessModes:
    - ReadWriteOnce
```

Step 6: Create PostgreSQL Deployment YAML file

kubectl create deployment postgresql-deployment –image=postgres:latest –dry-run=client -o yaml > postgresql-deployment.yml

Then adjust deployment yaml with environment, volume and volume mounts as below.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: postgres-sql
  name: postgresql-deployment
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: postgres-sql
  strategy: {}
  template:
    metadata:
      labels:
        app: postgres-sql
    spec:
      containers:
      - image: postgres:latest
        name: postgres-sql
        ports:
          - containerPort: 5432
        envFrom:
          - secretRef:
              name: psql-db-secrets
        volumeMounts:
          - mountPath: /var/lib/postgresql/data
            name: pgdbdatavol
      volumes:
        - name: pgdbdatavol
          persistentVolumeClaim:
            claimName: psql-db-pvc
```

Step 6: Create PostgreSQL Service YAML file

kubectl expose deployment postgresql-deployment –port=5432 –dry-run=client -o yaml > postgresql-svc.yml

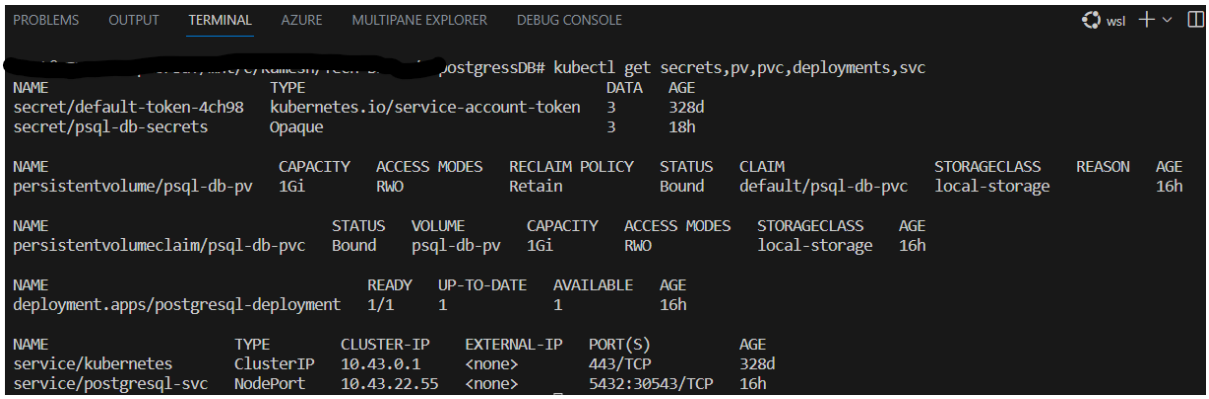
Adjust yaml to use Service type as NodePort

```yaml
apiVersion: v1
kind: Service
metadata:
  labels:
    app: postgres-sql
  name: postgresql-svc
spec:
  ports:
  - port: 5432
    protocol: TCP
    targetPort: 5432
    nodePort: 30543
  selector:
    app: postgres-sql
  type: NodePort
```

Step 7: Create resource in Kubernetes(Local Cluster created using Rancher Desktop)

kubectl config set-context rancher-desktop

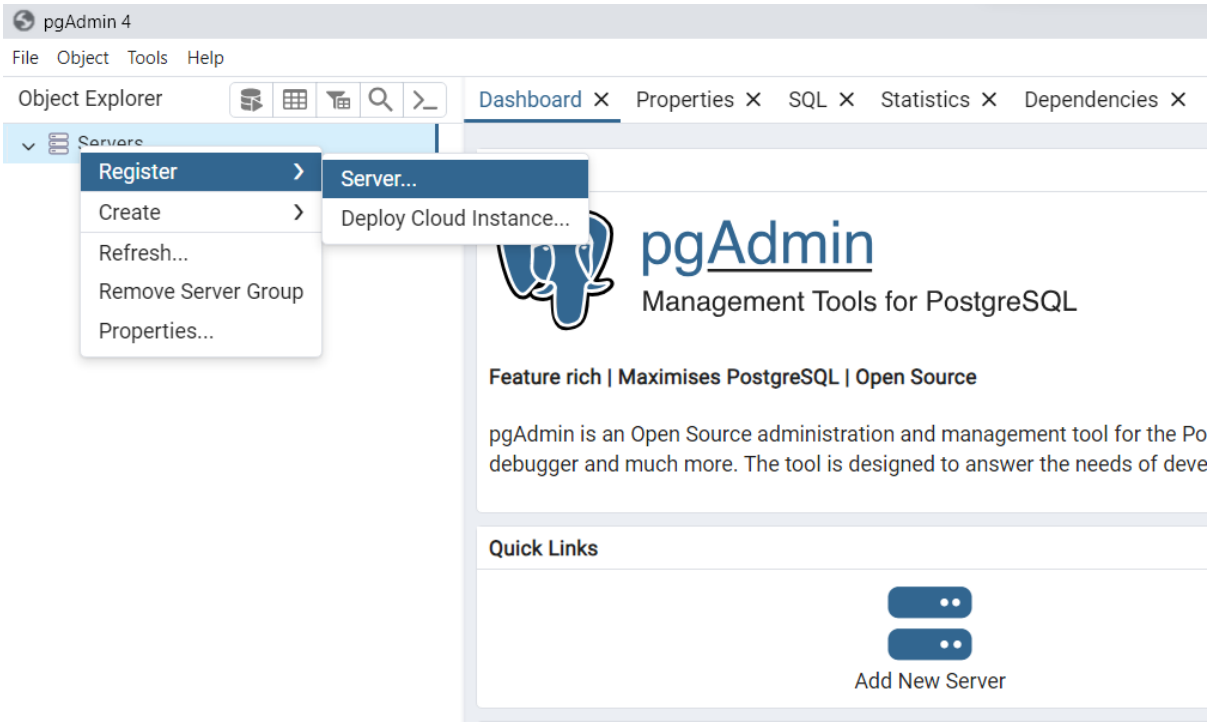
kubectl create -f configs/postgresql-secret.yml
kubectl create -f configs/postgresql-pv.yml
kubectl create -f configs/postgresql-pvc.yml
kubectl create -f deployment/postgresql-deployment.yml
kubectl create -f service/postgresql-svc.yml

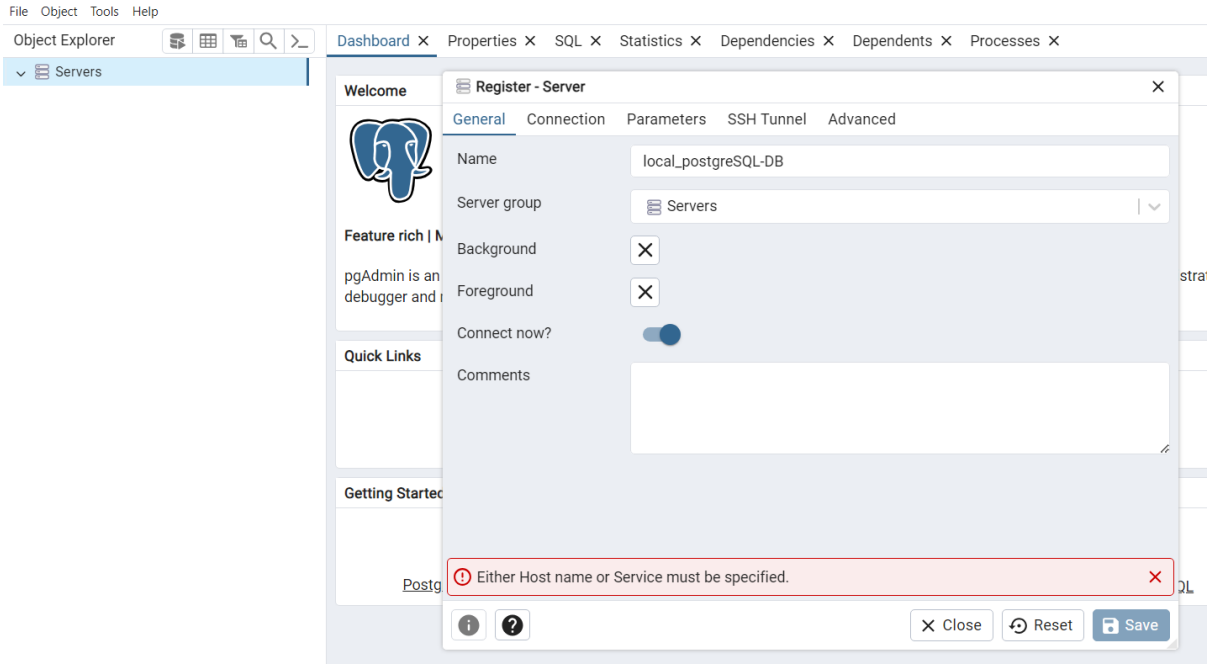


Test the Kubernetes PostgreSQL DB using pgAdmin

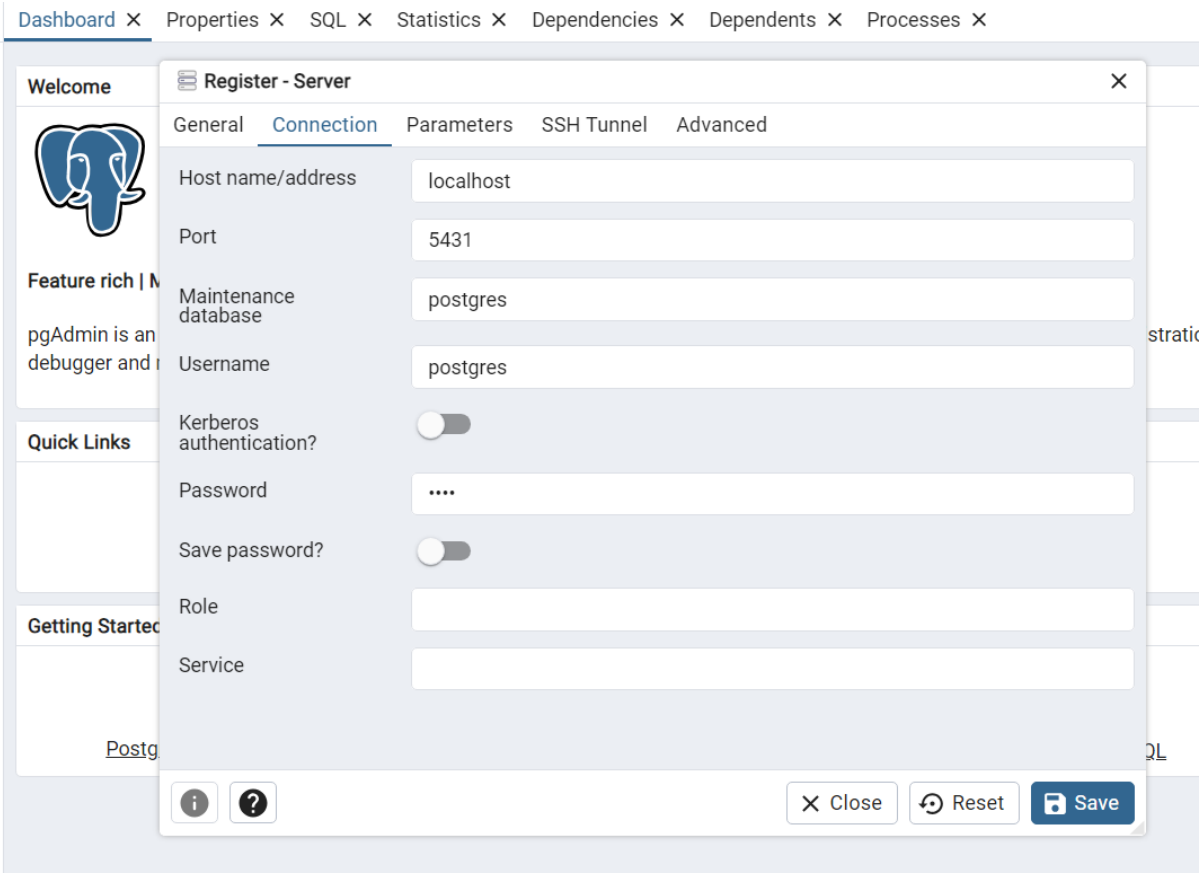
Install pgAdmin from here latest version: https://www.pgadmin.org/download/pgadmin-4-windows/

- Use Port Forward to access PostgreSQL service(service/postgresql-svc) from local as below
  kubectl port-forward service/postgresql-svc 5431:5432
  5431 – redirect to local port, 5432 is target port
- Steps to PostgreSQL DB using pgAdmin tool





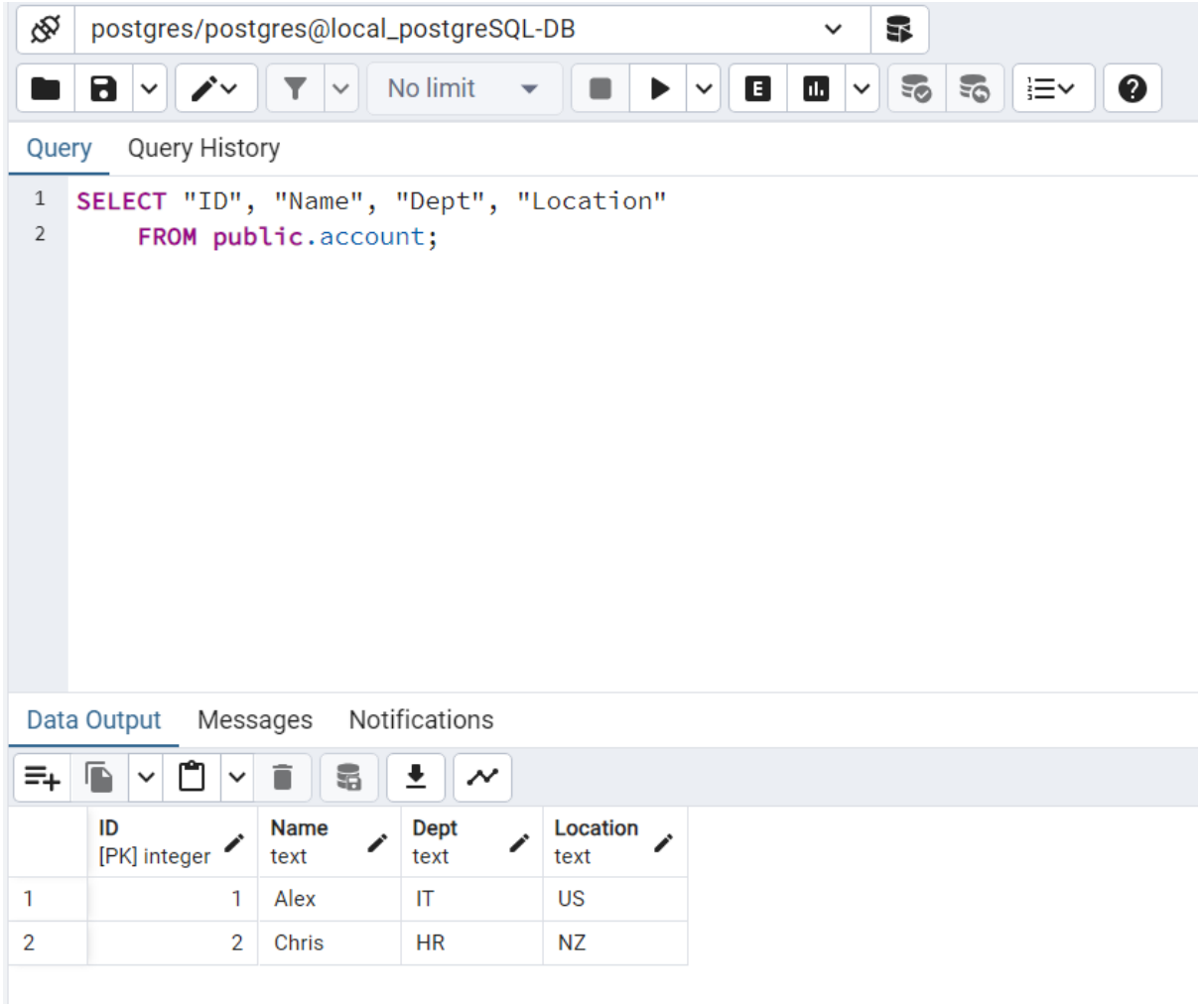
Host name as localhost/127.0.0.1, port as 5431, other details(database,Username,Passowrd) as mentioned in Kubernetes Secret and Click on Save button



Create Table and add data:

CREATE TABLE IF NOT EXISTS public.account
(
"ID" SERIAL PRIMARY KEY NOT NULL,
"Name" text NOT NULL,
"Dept" text NOT NULL,
"Location" text NOT NULL
)

INSERT INTO account("Name", "Dept", "Location") VALUES ('Alex', 'IT', 'US');
INSERT INTO account("Name", "Dept", "Location") VALUES ('Chris', 'HR', 'NZ');



Now Delete PostgresSQL Deployment and recreate it (using either below commands), to check the Data is Persists

kubectl replace -f deployment/postgresql-deployment.yml

OR

kubectl delete deployment postgresql-deployment

kubectl apply -f deployment/postgresql-deployment.yml