Kubernetes assignment 2

1. Deploy Postgres database using PVC & PV cluster.

To Deploy postgres database using PVC and PV cluster we just need to make a yaml manifest file. Official postgresSQL image can be taken from dockerhub. Persistent Volume and Persistent Volume Claim is used to store and persist the data even if the pods are deleted. So the yaml configuration for postgres is

```
apiVersion: apps/v1
kind: Deployment
metadata:
   name: postgres
spec:
   replicas: 1
   selector:
    matchLabels:
      app: postgres
   template:
      metadata:
         labels:
            app: postgres
      spec:
         volumes:
            - name: postgres-pv-storage
              persistentVolumeClaim:
                 claimName: postgres-pv-claim
         containers:
            - name: postgres
              image: postgres:11
              imagePullPolicy: IfNotPresent
              ports:
              - containerPort: 5432
              env:
              - name: POSTGRES_PASSWORD
                valueFrom:
                  secretKeyRef:
                    name: postgres-secret-config
                    key: password
              - name: PGDATA
                value: /var/lib/postgresql/data/pgdata

              volumeMounts:
              - mountPath: /var/lib/postgresql/data
                name: postgres-pv-storage
---
apiVersion: v1
```

```
kind: Secret

metadata:

   name: postgres-secret-config

type: Opaque

data:

   password: cG9zdGdyZXMK
---
apiVersion: v1
kind: PersistentVolume
metadata:
     name: postgres-pv-volume
     labels:
         type: local
spec:
     storageClassName: manual
     capacity:
         storage: 5Gi
     accessModes:
       - ReadWriteOnce
     hostPath:
         path: "/mnt/data"
---
apiVersion: v1

kind: PersistentVolumeClaim

metadata:
        name: postgres-pv-claim
spec:
        storageClassName: manual
        accessModes:
                - ReadWriteOnce
        resources:
                requests:
                        storage: 1Gi
```

Here first the postgresSQL pod is created using the **Deployment** component in the cluster, the root password is set using the **secrets** component, and finally the **Persistent Volume** and **Persistent Volume Claims** are created and configured with the deployment created.

To apply these rules, we will use **kubectl** command as:

```
~ kubectl apply -f postgres.yml
```

```
prajesh@prajesh-VirtualBox:~/postgres$ kubectl delete pv postgres-pv-volume
persistentvolume "postgres-pv-volume" deleted
prajesh@prajesh-VirtualBox:~/postgres$ vi postgres.yeml
prajesh@prajesh-VirtualBox:~/postgres$ vi postgres.yml
prajesh@prajesh-VirtualBox:~/postgres$ kubectl apply -f postgres.yml
deployment.apps/postgres created
secret/postgres-secret-config created
persistentvolume/postgres-pv-volume created
persistentvolumeclaim/postgres-pv-claim created
prajesh@prajesh-VirtualBox:~/postgres$ kubectl get pods
NAME                                    READY   STATUS    RESTARTS   AGE
hello-world-express-7447cfd7c7-mfb7v    1/1     Running   0          60m
mongo-intern-8c986c848-cd4s7            1/1     Running   0          43m
nginx-5668b5dfbb-g8r8g                  1/1     Running   0          83m
postgres-59f958ccd8-59jc9               1/1     Running   0          12s
prajesh@prajesh-VirtualBox:~/postgres$ kubectl get pods
NAME                                    READY   STATUS    RESTARTS   AGE
hello-world-express-7447cfd7c7-mfb7v    1/1     Running   0          61m
mongo-intern-8c986c848-cd4s7            1/1     Running   0          44m
nginx-5668b5dfbb-g8r8g                  1/1     Running   0          83m
postgres-59f958ccd8-59jc9               1/1     Running   0          19s
prajesh@prajesh-VirtualBox:~/postgres$ kubectl get deployments
NAME                   READY   UP-TO-DATE   AVAILABLE   AGE
hello-world-express    1/1     1            1           62m
mongo-intern           1/1     1            1           45m
nginx                  1/1     1            1           5h18m
postgres               1/1     1            1           117s
prajesh@prajesh-VirtualBox:~/postgres$ kubectl get pv
NAME                   CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM                         STORAGECLASS   REASON   AGE
postgres-pv-volume     5Gi        RWO            Retain           Bound    default/postgres-pv-claim     manual                  2m6s
prajesh@prajesh-VirtualBox:~/postgres$ kubectl get pvc
NAME                   STATUS   VOLUME                 CAPACITY   ACCESS MODES   STORAGECLASS   AGE
postgres-pv-claim      Bound    postgres-pv-volume     5Gi        RWO            manual         2m9s
prajesh@prajesh-VirtualBox:~/postgres$ kubectl get secrets
NAME                  TYPE                                  DATA   AGE
default-token-nzfml   kubernetes.io/service-account-token   3      46h
postgres-secret-config   Opaque                             1      2m20s
prajesh@prajesh-VirtualBox:~/postgres$
```

Again, expose the deployment as **Nodeport** to use the application

```
~  kubectl expose deployment postgres --type=NodePort --port=5432
```

```
prajesh@prajesh-VirtualBox:~/postgres$ kubectl expose deployment postgres --type=NodePort --port=5432
service/postgres exposed
prajesh@prajesh-VirtualBox:~/postgres$ kubectl get svc
NAME                  TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)          AGE
hello-world-express   NodePort    10.97.20.157     <none>        3000:31188/TCP   71m
kubernetes            ClusterIP   10.96.0.1        <none>        443/TCP          46h
mongo-intern          NodePort    10.100.50.214    <none>        27017:30797/TCP  54m
nginx                 NodePort    10.105.187.87    <none>        80:31293/TCP     112m
postgres              NodePort    10.105.134.208   <none>        5432:31427/TCP   8s
prajesh@prajesh-VirtualBox:~/postgres$
```

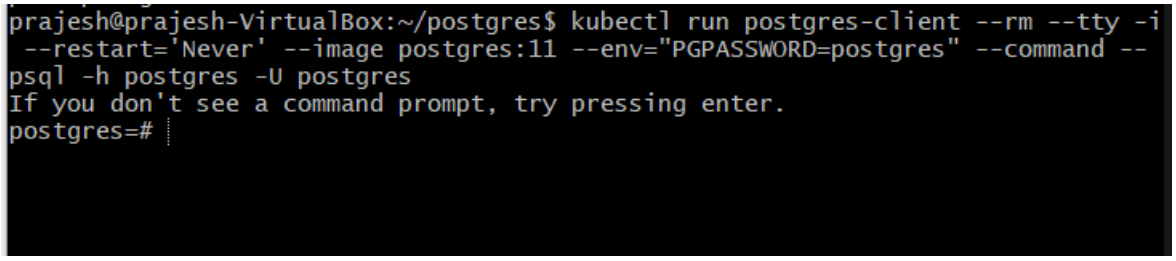
2. Deploy Postgres Client in cluster(psql)

3. Connect Postgres database from Postgres Client using core-dns's host name.

These two questions are done using one single command using kubectl, which prompts the psql terminal.

The command simply runs a pod with postgres Client and using the core dns inside the cluster connects to the database:

```
~ kubectl run postgres-client --rm --tty -i --restart='Never' --
   image postgres:11 --env="PGPASSWORD=postgres" --command -- psql
   -h postgres -U postgres
```



4. Create a database(internship) and few tables in database.

To Create database (internship) simple postgres SQL query can be written as

```
~ CREATE DATABASE internship;
```

```
postgres=# CREATE DATABASE internship
postgres-# ;
CREATE DATABASE
postgres=#
```

To create table, change into the database created and run the following query:

```
# \c internship
# CREATE TABLE members (
      id int,
      name varchar(255)
);
```

```
internship=# \c internship
You are now connected to database "internship" as user "postgres".
internship=# CREATE TABLE members (
internship(#  id int,
internship(#  name varchar(255)
internship(# );
CREATE TABLE
internship=# \dt
          List of relations
 Schema |  Name   | Type  |  Owner
--------+---------+-------+----------
 public | members | table | postgres
(1 row)

internship=#
```