1. Deploy Postgres database using PVC & PV cluster

To deploy PostgresDB, we need to configure database configmap for Postgres. For that, a file is created with the content shown in figure below:

Nano postgresconfig.yaml

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: postgres-config
  labels:
    app: postgres
data:
  POSTGRES_DB: postgresdb
  POSTGRES_USER: admin
  POSTGRES_PASSWORD: password
```

For deploying postgres, another file is created with following content:

Nano postgres-deployment.yaml

```
kind: Deployment
metadata:
  name: postgres
spec:
  replicas: 1
  selector:
    matchLabels:
      app: postgres
  template:
    metadata:
      labels:
        app: postgres
    spec:
      containers:
        - name: postgres
          image: postgres:10.1
          imagePullPolicy: "IfNotPresent"
          ports:
            - containerPort: 5432
          envFrom:
            - configMapRef:
                name: postgres-config
          volumeMounts:
            - mountPath: /var/lib/postgresql/data
              name: postgredb
      volumes:
        - name: postgredb
          persistentVolumeClaim:
            claimName: postgres-pv-claim
```

Again, for Persistent Volume and Persistent Volume Claim (PV and PVC cluster), another yaml configuration file is made and following content is saved:

Nano postgres-storage.yaml

```yaml
kind: PersistentVolume
apiVersion: v1
metadata:
  name: postgres-pv-volume
  labels:
    type: local
    app: postgres
spec:
  storageClassName: manual
  capacity:
    storage: 5Gi
  accessModes:
    - ReadWriteMany
  hostPath:
    path: "/mnt/data"
---
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: postgres-pv-claim
  labels:
    app: postgres
spec:
  storageClassName: manual
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 5Gi
```

To apply those configurations, following commands are used:

kubectl apply -f postgresconfig.yaml

kubectl apply -f postgres-deployment.yaml

kubectl apply -f postgres-storage.yaml

```
bj@batman:~/kubernetes$ kubectl apply -f postgres-deployment.yaml
deployment.apps/postgres created
bj@batman:~/kubernetes$ kubectl apply -f postgresconfig.yaml
configmap/postgres-config created
bj@batman:~/kubernetes$ kubectl apply -f postgres-storage.yaml
persistentvolume/postgres-pv-volume unchanged
persistentvolumeclaim/postgres-pv-claim unchanged
```

Now, with command kubectl get all, we can see the deployed pods and deployments which are active and ready.

```
bj@batman:~/kubernetes$ kubectl get all
NAME                            READY   STATUS    RESTARTS      AGE
pod/mongopod                    1/1     Running   1 (26m ago)   6h55m
pod/nginxpod                    1/1     Running   1 (26m ago)   6h57m
pod/nodepod                     1/1     Running   1 (26m ago)   6h56m
pod/postgres-7b9fb8d6c5-7hnj9   1/1     Running   1 (26m ago)   49m

NAME                 TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
service/kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP   6h59m

NAME                       READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/postgres   1/1     1            1           49m

NAME                                  DESIRED   CURRENT   READY   AGE
replicaset.apps/postgres-7b9fb8d6c5   1         1         1       49m
bj@batman:~/kubernetes$
```

Now with this command, we can enter the CLI of postgres database:

kubectl exec -it postgres-* -- psql -h localhost -U admin --password -p 5432:30733 postgresdb

When we provide the password mentioned in the above *postgresconfig.yaml* file, we can get access to the shell of the postgres database.

[In above command, the name of pod should be properly entered as shown below]

```
bj@batman:~/kubernetes$ kubectl exec -it postgres-7b9fb8d6c5-7hnj9 --  psql -h localhost -U admin --password -p 5432:30733 postgresdb
Password for user admin:
psql (10.1)
Type "help" for help.

postgresdb=#
```

2. Deploy Postgres Client in cluster (psql)

First of all, I made a namespace ‘client’ with command:

Kubectl create namespace client

And created service and deployment for namespace client using command:

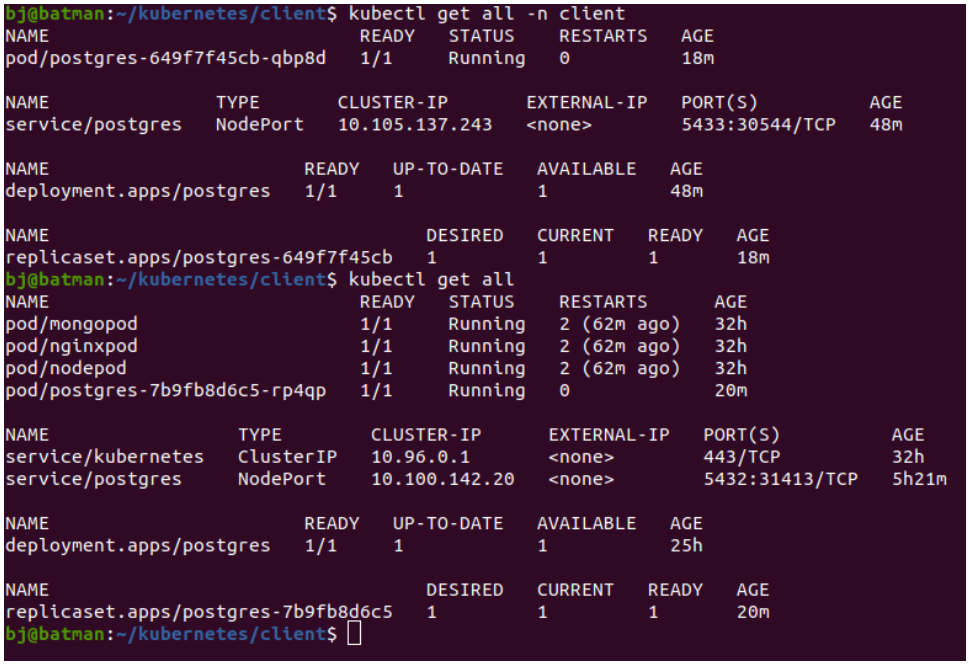
kubectl apply -f postgresconfig.yaml -n client

kubectl apply -f postgres-deployment.yaml -n client

kubectl apply -f postgres-storage.yaml -n client

With the command ***kubectl get all -n client,*** we can see all components for client namespace and with command ***kubectl get all,*** we can see components for default namespace.

[*the pod name may differ from QN1 since I was learning and removed the previous pods and made new ones, but with the same configurations*]



Here, we can see the running pods in both default namespace and client namespace.

3. <u>Connect Postgres database from Postgres Client using core-dns's host name.</u>

To run postgres database from the default namespace, this command is used in this case.

kubectl exec -it pod/postgres-7b9fb8d6c5-rp4qp -- psql -h localhost -U admin --password -p 5432 postgresdb

```
bj@batman:~/kubernetes/client$ kubectl get all -n client
NAME                            READY   STATUS    RESTARTS   AGE
pod/postgres-649f7f45cb-qbp8d   1/1     Running   0          46m

NAME               TYPE       CLUSTER-IP       EXTERNAL-IP   PORT(S)          AGE
service/postgres   NodePort   10.105.137.243   <none>        5433:30544/TCP   76m

NAME                       READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/postgres   1/1     1            1           76m

NAME                                  DESIRED   CURRENT   READY   AGE
replicaset.apps/postgres-649f7f45cb   1         1         1       46m
bj@batman:~/kubernetes/client$ kubectl exec -it pod/postgres-649f7f45cb-qbp8d -n client --  psql -h postgres.default -U admin --password -p 5432 postg
resdb
Password for user admin:
psql (10.1)
Type "help" for help.

postgresdb=# \l
                                 List of databases
    Name    |  Owner   | Encoding |  Collate   |   Ctype    |   Access privileges
------------+----------+----------+------------+------------+-----------------------
 internship | admin    | UTF8     | en_US.utf8 | en_US.utf8 |
 postgres   | postgres | UTF8     | en_US.utf8 | en_US.utf8 |
 postgresdb | postgres | UTF8     | en_US.utf8 | en_US.utf8 |
 template0  | postgres | UTF8     | en_US.utf8 | en_US.utf8 | =c/postgres          +
            |          |          |            |            | postgres=CTc/postgres
 template1  | postgres | UTF8     | en_US.utf8 | en_US.utf8 | =c/postgres          +
            |          |          |            |            | postgres=CTc/postgres
 testdb     | admin    | UTF8     | en_US.utf8 | en_US.utf8 |
(6 rows)
```

And this command is used to fun from client namespace for the same database which is named *postgresdb*:

kubectl exec -it pod/postgres-649f7f45cb-qbp8d -n client -- psql -h postgres.default -U admin --password -p 5432 postgresdb

```
bj@batman:~/kubernetes$ kubectl get all
NAME                            READY   STATUS    RESTARTS      AGE
pod/mongopod                    1/1     Running   2 (86m ago)   32h
pod/nginxpod                    1/1     Running   2 (86m ago)   32h
pod/nodepod                     1/1     Running   2 (86m ago)   32h
pod/postgres-7b9fb8d6c5-rp4qp   1/1     Running   0             44m

NAME                 TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
service/kubernetes   ClusterIP   10.96.0.1       <none>        443/TCP          32h
service/postgres     NodePort    10.100.142.20   <none>        5432:31413/TCP   5h45m

NAME                       READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/postgres   1/1     1            1           26h

NAME                                  DESIRED   CURRENT   READY   AGE
replicaset.apps/postgres-7b9fb8d6c5   1         1         1       44m
bj@batman:~/kubernetes$ kubectl exec -it pod/postgres-7b9fb8d6c5-rp4qp --  psql -h localhost -U admin --password -p 5432 postgresdb
Password for user admin:
psql (10.1)
Type "help" for help.

postgresdb=# \l
                                List of databases
    Name    |  Owner   | Encoding |  Collate   |   Ctype    |   Access privileges
------------+----------+----------+------------+------------+-----------------------
 internship | admin    | UTF8     | en_US.utf8 | en_US.utf8 |
 postgres   | postgres | UTF8     | en_US.utf8 | en_US.utf8 |
 postgresdb | postgres | UTF8     | en_US.utf8 | en_US.utf8 |
 template0  | postgres | UTF8     | en_US.utf8 | en_US.utf8 | =c/postgres          +
            |          |          |            |            | postgres=CTc/postgres
 template1  | postgres | UTF8     | en_US.utf8 | en_US.utf8 | =c/postgres          +
            |          |          |            |            | postgres=CTc/postgres
 testdb     | admin    | UTF8     | en_US.utf8 | en_US.utf8 |
(6 rows)
```

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

After logging in to the database, we can use database commands to create databases and tables.

Database is created using command:

CREATE DATABASE internship;

```
postgresdb=# CREATE DATABASE internship;
CREATE DATABASE
postgresdb=# \l
                                 List of databases
    Name    |  Owner   | Encoding |  Collate   |   Ctype    |   Access privileges
------------+----------+----------+------------+------------+-----------------------
 internship | admin    | UTF8     | en_US.utf8 | en_US.utf8 |
 postgres   | postgres | UTF8     | en_US.utf8 | en_US.utf8 |
 postgresdb | postgres | UTF8     | en_US.utf8 | en_US.utf8 |
 template0  | postgres | UTF8     | en_US.utf8 | en_US.utf8 | =c/postgres          +
            |          |          |            |            | postgres=CTc/postgres
 template1  | postgres | UTF8     | en_US.utf8 | en_US.utf8 | =c/postgres          +
            |          |          |            |            | postgres=CTc/postgres
(5 rows)
```

We can see available databases using command ***\l***

The database is selected using command:

\c internship

Now table named *information* is created in database *internship* using command:

CREATE TABLE information(id serial PRIMARY KEY, username VARCHAR(50) UNIQUE NOT NULL, useremail VARCHAR(50) UNIQUE NOT NULL, lastlogin TIMESTAMP);

```
internship=# CREATE TABLE information (id serial PRIMARY KEY , username VARCHAR(50) UNIQUE NOT NULL, useremail VARCHAR(50) UNIQUE NOT NULL, lastlogin
TIMESTAMP);
CREATE TABLE
internship=# \dt
```

We can see the table created with name *information* using command:

\d information

```
internship=# \d information
                                    Table "public.information"
  Column   |            Type             | Collation | Nullable |                  Default
-----------+-----------------------------+-----------+----------+-------------------------------------------
 id        | integer                     |           | not null | nextval('information_id_seq'::regclass)
 username  | character varying(50)       |           | not null |
 useremail | character varying(50)       |           | not null |
 lastlogin | timestamp without time zone |           |          |
Indexes:
    "information_pkey" PRIMARY KEY, btree (id)
    "information_useremail_key" UNIQUE CONSTRAINT, btree (useremail)
    "information_username_key" UNIQUE CONSTRAINT, btree (username)
```

Similarly, we also added another table for evaluation as shown in figure below: We can create and delete tables and databases using proper database commands for postgreSQL databases.

```
internship=# CREATE TABLE evaluation (username VARCHAR(50) UNIQUE NOT NULL, score INT NOT NULL, lastlogin TIMESTAMP);
CREATE TABLE
internship=# \dt
          List of relations
 Schema |    Name     | Type  | Owner
```