1. Deploy Postgres database using PVC & PV cluster

Making a directory mongo/data in host machine to persist data of postgres

mkdir -p mongo/data

```
bibek@bibek-LfTech:~/assignment/mongo$ ls
data
bibek@bibek-LfTech:~/assignment/mongo$ pwd
/home/bibek/assignment/mongo
bibek@bibek-LfTech:~/assignment/mongo$
```

Creating yaml file for PV and PVC in directory mongo

vi pv-pvc.yaml

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: creating-pv-local
  labels:
    type: local
spec:
  storageClassName: local-pv
  capacity:
    storage: 2Gi
  accessModes:
    - ReadWriteMany
  hostPath:
    path: "/home/bibek/assignment/mongo/data"

---

apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: creating-pvc-local
spec:
  storageClassName: local-pv
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
```

Here **storageClassName** of PV and PVC **should be identica**l.

And while PV is created of 2 GB , PVC claims for 1 GB out of 2 GB.

AccessMode represents "ReadWriteMany" which means the volume can be mounted as read-write by many nodes.

We can use other options as well like: **ReadWriteOnce, ReadOnlyMany,** etc

To created pv and pvc applying yaml conf file

sudo kubectl apply -f pv-pvc.yaml

```
bibek@bibek-LfTech:~/assignment/mongo$ sudo kubectl apply -f pv-pvc.yaml
[sudo] password for bibek:
persistentvolume/creating-pv-local created
persistentvolumeclaim/creating-pvc-local created
bibek@bibek-LfTech:~/assignment/mongo$ sudo kubectl get pv
NAME               CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM
                    STORAGECLASS   REASON   AGE
creating-pv-local   2Gi        RWX            Retain           Bound    default/
creating-pvc-local   local-pv                17s
bibek@bibek-LfTech:~/assignment/mongo$ sudo kubectl get pvc
NAME                 STATUS   VOLUME              CAPACITY   ACCESS MODES   STOR
AGECLASS   AGE
creating-pvc-local   Bound    creating-pv-local   2Gi        RWX            loca
l-pv       26s
bibek@bibek-LfTech:~/assignment/mongo$
```

If we run postgres pod without POSTGRES_PASSWORD env we get this error and pod doesn't starts.

```
stgres-image-container
  Warning  BackOff    4s (x2 over 5s)    kubelet            Back-off restarting
failed container
bibek@bibek-LfTech:~/assignment/mongo$ sudo kubectl logs postgres-deployment-557
cd7c558-h9tck
Error: Database is uninitialized and superuser password is not specified.
       You must specify POSTGRES_PASSWORD to a non-empty value for the
       superuser. For example, "-e POSTGRES_PASSWORD=password" on "docker run".

       You may also use "POSTGRES_HOST_AUTH_METHOD=trust" to allow all
       connections without a password. This is *not* recommended.

       See PostgreSQL documentation about "trust":
       https://www.postgresql.org/docs/current/auth-trust.html
```

Writing env variables in plain text inside deployment for pod is not a best practice

Creating configmap to pass environment variables.

vi configmap.yaml

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: postgres-config
data:
  POSTGRES_DB: testdb1
  POSTGRES_USER: test
  POSTGRES_PASSWORD: tes
```

I have not shown the password.

Using this configmap we can pass the environment value to postgres-pod

We created config map with kubectl apply command

sudo kubectl apply -f configmap.yaml

```
bibek@bibek-LfTech:~/assignment/mongo$ vi configmap.yaml
bibek@bibek-LfTech:~/assignment/mongo$ sudo kubectl apply -f configmap.yaml
configmap/postgres-config created
bibek@bibek-LfTech:~/assignment/mongo$ sudo kubectl get configmap
NAME               DATA   AGE
kube-root-ca.crt   1      2d12h
postgres-config    3      15s
bibek@bibek-LfTech:~/assignment/mongo$
```

We can see that the configmap is created and running.

Creating yaml file of deployment for postgres-pod

vi postgres.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: postgres-deployment
  labels:
    type: local
spec:
  selector:
    matchLabels:
      type: local
  replicas: 1
  template:
    metadata:
      name: postgres-pod-deployment
      labels:
        type: local
    spec:
      volumes:
        - name: postgres-storage-local
          persistentVolumeClaim:
            claimName: creating-pvc-local
      containers:
        - name: postgres-image-container
          image: postgres
          envFrom:
            - configMapRef:
                name: postgres-config
          volumeMounts:
            - name: postgres-storage-local
              mountPath: /var/lib/postgresql/data
              subPath: postgres
```

Here we illustrated the name of PVC (**claimName**) in volume section

We passed environment variables using **configMapRef** using **envFrom.**

We mount the postgres pod **/var/lib/postgresql/data** directory with our localhost for volume persisting.

Now creating postgres pod using kubectl apply

sudo kubectl apply -f postgres.yaml

```
bibek@bibek-LfTech:~/assignment/mongo$ vi postgres.yaml
bibek@bibek-LfTech:~/assignment/mongo$ sudo kubectl apply -f postgres.yaml
deployment.apps/postgres-deployment created
bibek@bibek-LfTech:~/assignment/mongo$ sudo kubectl get pods
NAME                                  READY   STATUS    RESTARTS   AGE
postgres-deployment-5cc9f47b97-z4bmq   1/1     Running   0          9s
bibek@bibek-LfTech:~/assignment/mongo$
```

We can see that the postgrespod is created and running.

Now creating service for the postgres-pod

vi service.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: service-postgres
spec:
  type: NodePort
  ports:
    - port: 7779
      targetPort: 5432
      nodePort: 30006
  selector:
    type: local
```

using NodePort type and assigning static nodeport 30006

Creating service using kubectl command

sudo kubectl apply -f service.yaml

```
bibek@bibek-LfTech:~/assignment/mongo$ vi service.yaml
bibek@bibek-LfTech:~/assignment/mongo$ sudo kubectl apply -f service.yaml
service/service-postgres created
bibek@bibek-LfTech:~/assignment/mongo$ sudo kubectl get services
NAME               TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)          A
GE
kubernetes         ClusterIP   10.96.0.1        <none>        443/TCP          2
d12h
service-postgres   NodePort    10.102.201.179   <none>        7779:30006/TCP   9
s
bibek@bibek-LfTech:~/assignment/mongo$
```

We can see the service is created and running with nodeport 30006.