# Config Maps for PostgreSQL Configurations

We will be using config maps for storing PostgreSQL related information. Here, we are using the database, user and password in the config map which will be used by the PostgreSQL pod in the deployment template.

File: postgres-configmap.yaml

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: postgres-config
  labels:
    app: postgres
data:
  POSTGRES_DB: postgresdb
  POSTGRES_USER: postgresadmin
  POSTGRES_PASSWORD: admin123
```

Create Postgres config maps resource

```
$ kubectl create -f postgres-configmap.yaml
configmap "postgres-config" created
```

```
[ec2-user@ip-172-31-84-37 PSQL]$ kubectl create -f postgres-configmap.yaml
configmap "postgres-config" created
```

# Persistent Storage Volume

As you all know that Docker containers are ephemeral in nature. All the data which is generated by or in the container will be lost after termination of the container instance.

To save the data, we will be using Persistent volumes and persistent volume claim resource within Kubernetes to store the data on persistent storages.

Here, we are using local directory/path as Persistent storage resource (/mnt/data)

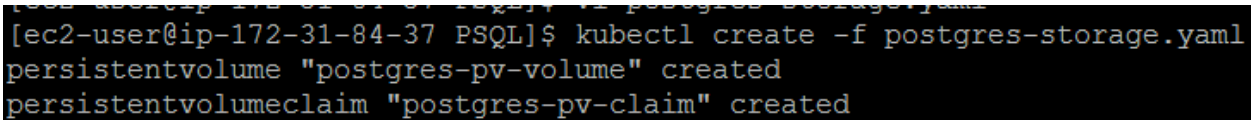
File: postgres-storage.yaml

```
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
```

Create storage related deployments

```
$ kubectl create -f postgres-storage.yaml
persistentvolume "postgres-pv-volume" created
persistentvolumeclaim "postgres-pv-claim" created
```




## PostgreSQL Deployment

PostgreSQL manifest for deployment of PostgreSQL container uses PostgreSQL 10.4 image. It is using PostgreSQL configuration like username, password, database name from the configmap that we created earlier. It also mounts the volume created from the persistent volumes and claims to make PostgreSQL container's data persists.

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: postgres
spec:
  replicas: 1
  template:
    metadata:
      labels:
        app: postgres
    spec:
      containers:
        - name: postgres
          image: postgres:10.4
          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
```

Create Postgres deployment

```
$ kubectl create -f postgres-deployment.yaml
deployment "postgres" created
```

```
[ec2-user@ip-172-31-84-37 PSQL]$ kubectl create -f postgres-deployment.yaml
deployment "postgres" created
```

Now pod for postgresql is running:

```
[ec2-user@ip-172-31-84-37 PSQL]$ kubectl get pods
NAME                                READY     STATUS    RESTARTS   AGE
nginx-deployment-75675f5897-6lq5x   1/1       Running   0          4h
nginx-deployment-75675f5897-98hxg   1/1       Running   0          4h
nginx-deployment-75675f5897-jfd9n   1/1       Running   0          4h
postgres-7ff9df5765-nqzbk           1/1       Running   0          2m
vault-xn256                         1/1       Running   0          3h
```

Now execute some command on this postgresql pod:

```
[ec2-user@ip-172-31-84-37 PSQL]$ kubectl exec -it postgres-7ff9df5765-nqzbk /bin/bash
root@postgres-7ff9df5765-nqzbk:/# su postgresadmin
No passwd entry for user 'postgresadmin'
root@postgres-7ff9df5765-nqzbk:/# su - postgres
No directory, logging in with HOME=/
$ psql vault
psql: FATAL:  database "vault" does not exist
$ psql postgresadmin
psql: FATAL:  database "postgresadmin" does not exist
$ psql postgresdb
psql (10.4 (Debian 10.4-2.pgdg90+1))
Type "help" for help.

postgresdb=# show tables;
ERROR:  unrecognized configuration parameter "tables"
```

Create a table:

```
CREATE TABLE vault_kv_store (
    parent_path TEXT COLLATE "C" NOT NULL,
    path        TEXT COLLATE "C",
    key         TEXT COLLATE "C",
    value       BYTEA,
    CONSTRAINT pkey PRIMARY KEY (path, key)
);
```

```
postgresdb=# CREATE TABLE vault_kv_store (
postgresdb(#     parent_path TEXT COLLATE "C" NOT NULL,
postgresdb(#     path        TEXT COLLATE "C",
postgresdb(#     key         TEXT COLLATE "C",
postgresdb(#     value       BYTEA,
postgresdb(#     CONSTRAINT pkey PRIMARY KEY (path, key)
postgresdb(# );
CREATE TABLE
```

Then create index:

```
CREATE INDEX parent_path_idx ON vault_kv_store (parent_path);
```

```
postgresdb=# CREATE INDEX parent_path_idx ON vault_kv_store (parent_path);
CREATE INDEX
```

Now that PostgreSQL is properly configured, we need to create a configuration file to inform Vault that its storage backend will be the Vault database inside this postgresql pod.