

# Deploy PostgreSQL on Kubernetes cluster

## Introduction

Kubernetes is an open-source container orchestration system for automating deployment, scaling and management of containerized applications. Running a PostgreSQL database on Kubernetes provides a way to provision stateful container using persistent volumes, stateful sets, etc.

Here I'm providing steps to run PostgreSQL database on Kubernetes cluster.

## Prerequisites

- Working Kubernetes Cluster

Need provision the Kubernetes cluster on any public cloud provider (like AWS, Azure or Google cloud) with different availability zone for single point for failure.

To Deploy PostgreSQL on Kubernetes following are the steps:

- Postgres Docker Image
- Config Maps for storing Postgres configurations
- Persistent Storage Volume
- PostgreSQL Deployment
- PostgreSQL Service

## PostgreSQL Docker Image

I'm using PostgreSQL 10.4 Docker image from the public registry. This image will provide the functionality of providing custom configurations/environment variables of PostgreSQL like username, password, database name and path, etc.

## 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: poStGres675#
```

## Create Postgres config maps resource

```
$ kubectl create -f postgres-configmap.yaml
```

## Persistent Storage Volume:

We 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, need to use Persistent volumes and persistent volume claim resource within Kubernetes to store the data on persistent storages.

Here, 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
```

## 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 I created

earlier. It also mounts the volume created from the persistent volumes and claims to make PostgreSQL container's data persists.

File: postgres-deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: postgres
  name: postgres
spec:
  replicas: 1
  selector:
    matchLabels:
      app: postgres
  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: postgresdb
volumes:
  - name: postgresdb
    persistentVolumeClaim:
      claimName: postgres-pv-claim

```

### Create Postgres deployment

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

## PostgreSQL Service

To access the deployment or container, need to expose PostgreSQL service. Kubernetes provides different type of services like ClusterIP, NodePort and LoadBalancer.

With ClusterIP can access PostgreSQL service within Kubernetes. NodePort gives the ability to expose service endpoint on the Kubernetes nodes. For accessing PostgreSQL externally, need to use a Load Balancer service type which exposes the service externally.

*File: postgres-service.yaml*

```

apiVersion: v1
kind: Service
metadata:
  name: postgres
  labels:
    app: postgres
spec:
  type: NodePort
  ports:
    - port: 5432
  selector:
    app: postgres

```


## Create Postgres Service

```
$ kubectl create -f postgres-service.yaml
```

## Connect to PostgreSQL

For connecting PostgreSQL, we need to get the Node port from the service deployment.

```
root@v06-k8smaster-main:/usr/local# kubectl get svc -o wide
NAME          TYPE          CLUSTER-IP    EXTERNAL-IP    PORT(S)          AGE    SELECTOR
kubernetes    ClusterIP     10.96.0.1     <none>         443/TCP          25d    <none>
postgres      NodePort      10.100.88.3   <none>         5432:31327/TCP   4h55m  app=postgres
```



Need to use port 31327 to connect to PostgreSQL from machine/node present in kubernetes cluster with credentials given in the configmap earlier.

```
$ psql -h localhost -U postgresadmin --password -p 31327 postgresdb
```

```
root@v06-k8smaster-main:/usr/local# psql -h localhost -U postgresadmin --password -p 31327 postgresdb
Password for user postgresadmin:
psql (10.15 (Ubuntu 10.15-0ubuntu0.18.04.1), server 10.4 (Debian 10.4-2.pgdg90+1))
Type "help" for help.

postgresdb=#
```

## Conclusion

Running PostgreSQL on Kubernetes help to utilize resources in a better way than when just using virtual machines. Kubernetes also provides isolation of other applications using PostgreSQL within the same virtual machine or Kubernetes cluster.

**-Monirul**

DevOps & Systems Analyst

Cell: +8801811456089, +8801912329277