**Deploying postgresql on kubernetes**

**Prerequisites**

* Working Kubernetes Cluster
* Basic understanding of Docker

You can provision the Kubernetes cluster on any public cloud provider like AWS, Azure or Google cloud, etc.

To Deploy PostgreSQL on Kubernetes we need to follow below steps:

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

## **PostgreSQL Docker Image**

We are 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.

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

## **Persistent Storage Volume**

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)

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 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

## **PostgreSQL Service**

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

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

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.

$ kubectl get svc postgres

We need to use the external-ip port to connect to PostgreSQL from machine/node present in kubernetes cluster with credentials given in the configmap earlier.

$ psql -h localhost -U postgresadmin1 --password -p 31070 postgresdb

## **Delete PostgreSQL Deployments**

For deletion of PostgreSQL resources, we need to use below commands.

# kubectl delete service postgres

# kubectl delete deployment postgres

# kubectl delete configmap postgres-config

# kubectl delete persistentvolumeclaim postgres-pv-claim

# kubectl delete persistentvolume postgres-pv-volume

By using the above steps you can provision a standalone PostgreSQL instance on a Kubernetes Cluster.