Design Document

for PostgreSQL database Backup

Table of Contents

Table of Figure

1. Overview:

The aim of this design document is to create a Kubernetes system that takes backup for a PostgreSQL database. To create a flawless backup system, we have to make sure the storage system is highly available for backup and restore cycle. In recent days data plays a vital role in many companies, and they give high importance and security to save those data for analytics or for reusing in case of any disaster with the existing system. This proposed design for a kubernetes system, firstly create a Helm charts which would be having the bunch of YAML files as a single package for creating a complete Kubernetes system with all the services, configMap, deployments, HPA, PVC, metrics and many more in it. A PgBackupRest is an open source backup tool, is been integrated with the backup schedular system which runs 2 cron jobs to take backup on the scheduled time. (Full weekly Backup and incremental Daily backup).

2. Problem Statement:

- As Containers are increasingly deployed in the operational environment, organizations are highly concerned about security, backup and recovery.
- Initially, most of the containerized applications were stateless which was very easy for them to deploy on public cloud.
- However, as now the stateful applications i.e. applications that require access to the database have become containerized. So, the need for backup and recovery has increased dramatically.
- Few common disasters such as accidently deleted a namespace, Kubernetes API Upgrade failure, cluster in unrecoverable state etc., So having a backup system will help to restore the data during this time.

3. System Scope:

- The scope of the Kubernetes backup system is to help the user, for maintain high availability service and able to take backup without data lose.
- By using helm charts we can achieve high productivity, easily modifiable reduces the complexity of the deployment and enables the adaption of cloud native applications.
- Helm charts can be maintained and can be used by various developers around the world by having the same templates.
- Easy deployment using Blue/green deployment.
- Using IBM Kubernetes services makes the end user very easy to access and deploy it very easily with high security.

4. Detailed Design

To achieve the system proposed, a step-by-step procedure has been carried out to create a Kubernetes system that takes backup for a PostgreSQL database. Below are the few basic steps that should be followed for creating a Kubernetes system using Helm-chart.

4.1. Creating Kubernetes Cluster:

This will be the initial step of a system, i.e. creating a Kubernetes system there are two ways to do this this is just few important steps while creating a cluster,

- Select an service provider and launch a kubernetes service
- From local system we have to install minikube

If the Kubernetes service is launched from Cloud Service provider (IBM cloud), first we have to create a cluster by selecting the plan (free or standard cluster) and the version of the kubernetes. Moreover, we have to select the network environment (VPC) and then set the subnet and location for VPC.

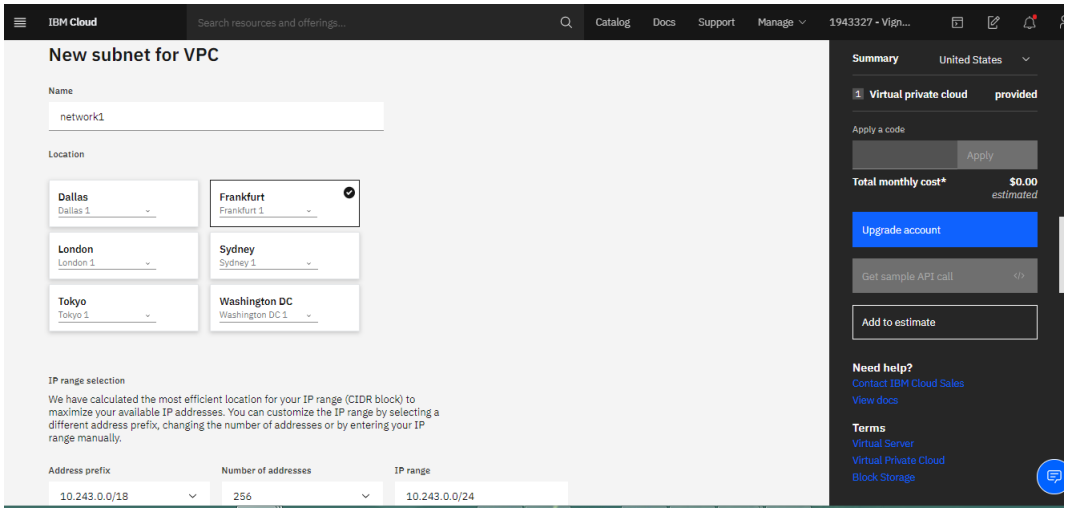

Figure 1 : Creating Subnet for VPC

- As we want our service in high availability we could also select location multi zone and set the geography. So, the worker nodes will be running across different region servers if one gets terminated or stopped, the worker zone will be automatically redirected to a different availability zone all this will be handled by IBM cloud.

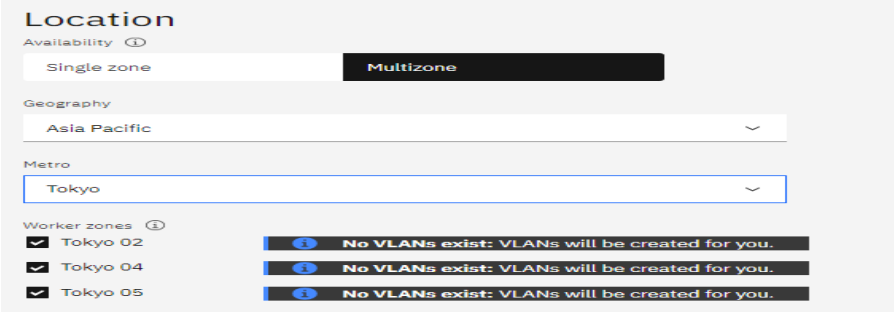

Figure 2: Selecting worker node Location

- We could select the number of worker nodes we want to be present, with the configuration of each worker node. This can be upgraded if the user or organization wants to be dependent on the traffic or load they handle.
- After doing all the required configuration, we are able to access the kubernetes dashboard, where we could virtualize and monitor the status/ usage of all the nodes, pods, services etc. We code either create a new service component (such as pod, PVC) from the Kubernetes UI or from the CLI (host terminal/ cloud CLI).
- We can start using the service by passing the command from the Cloud shell CLI, initially we have to login to the IBM cloud by providing the one time passcode that has been generated by the IBM cloud, after the basic authentication we could connect the Kubernetes cluster via CLI by passing credentials as below. From the CLI, we could perform all the kubectl commands and manage the cluster.

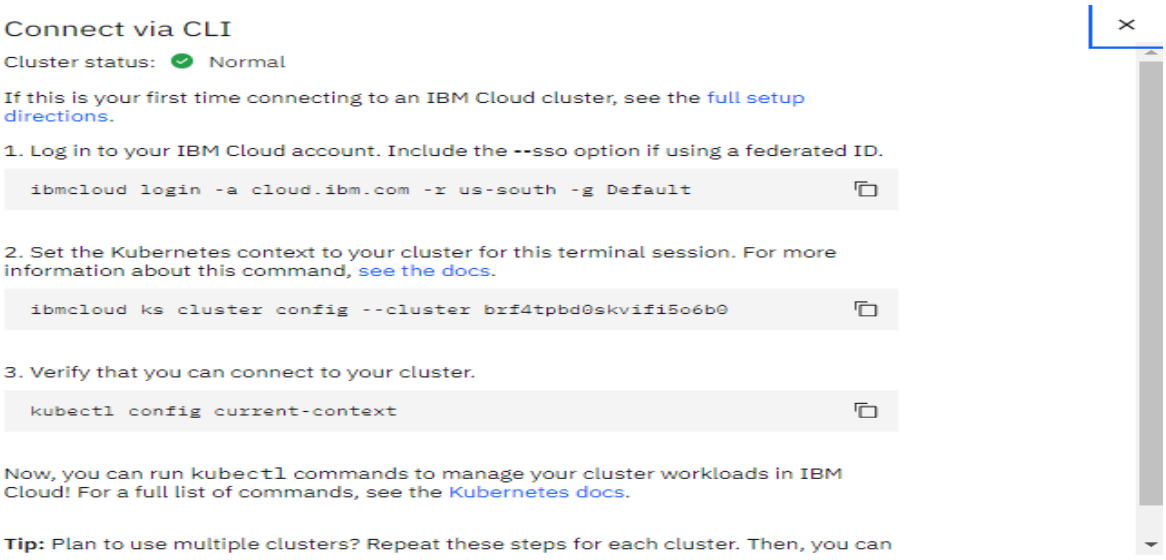

Figure 3: Connect via CLI

4.2. Creating Persistent Volume (PV) and Storage Classes (SC):

A Persistent Volume (PV) is a piece of storage in the cluster that has been provisioned by an administrator or dynamically provisioned using Storage Classes, so the Persistent Volume claim can request the resource that is needed for the particular node. All this persistent volume will be maintained by the IBM Kubernetes service, if we want to do any changes we could update the persistentvolumeclaim.yaml and change the required resource. So, in this design we will enable dynamic provisioning, in the storageclass.ymal file we could specify all the configuration needed. In `storageClassName` we provide the name of the storage class to be used to provision file storage. We can choose either IBM-provided storage class or create our own one. By default IBM provides `ibmc-file-bronze` default. Generally in an organization the PV and storage Class will be maintained by the administrator who will be having edit policy.

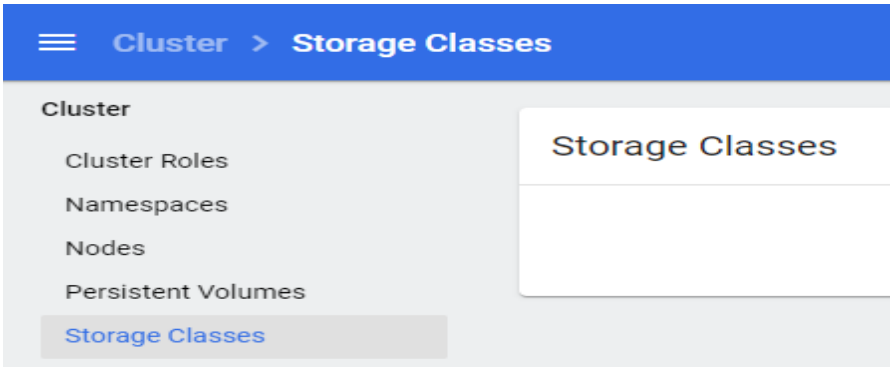

Figure 4: Creating SV and PV

4.3. Creating Helm Charts**:**

Helm is an application package manager for Kubernetes, which will maintain the download, installation and deployments of apps in the kubernetes cluster. Helm charts are the way we can define an application as a collection of related Kubernetes resources. So, helm makes the deployment of applications easily to manage through the templated approach. During the creation of helm chats there will be few default YAML files, generally the helm charts follow the same structure but still we could add or delete the files that are required to create the Kubernetes system. Creating Helm-Charts when necessary as they help harden your development practices and make your Kubernetes deployments much more reusable.

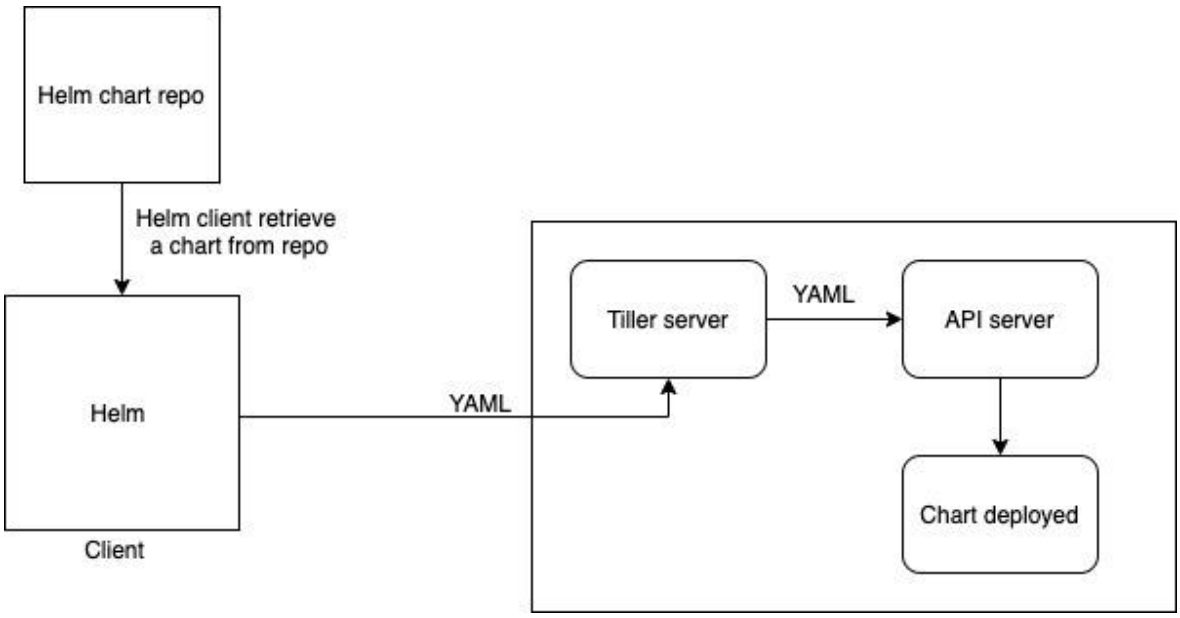

Figure 5: Working of helm / deploy chart

Generally we could pull the third party Helm charts that have been associated in the Helm repo from the Helm-Client or we could create our own charts and pass it to the Tiller. Tiller is the service that actually communicates with the Kubernetes API to manage our Helm packages. Once the Helm charts are installed from the CLI the bundle of YAML files will be passed to Tiller server, which will deploy the charts to the Kubernetes master and based on the resources configured in values files it will release the deployment model in the Kubernetes cluster. Helm will act as a desire state configuration, even if any deployment, pod, service etc. is deleted it will start recreates till we uninstall the release.

4.4. Helm Chart Structure:

- First we have to create a install all the prerequisites to run helm on the system
- From the Helm client we are able to create our own Helm charts, for example the name "**kubebackup**" is the name of this Helm-chart.

```
helm create kubebackup
Creating kubebackup
$ ls kubebackup /
Chart.YAML charts/ templates/ values.YAML
```

Helm Chart Structure

- Charts.YAML tells the overall outline of that Helm-chart and the values.YAML defines the values that will be deployed in the templates we created. Each time if we want to create a new Kubernetes system or modify the existing changes we can just edit the values.YAML file.

- Next is the most important part of the charts, the template directory which holds all the configurations of the application that will be deployed into the cluster.

```
$ ls templates/
NOTES.txt _helpers.tpl deployment.YAML metrics.YAML service.YAML Back Up.YAML configmap.YAML secrets.YAML tests/
```

- After adding all the required template YAML files and modified the values that the system should be created, we have to push the helm chart into the Kubernetes master from the Helm client CLI.

```
$ helm install backup-chart kubebackup/ --values kubebackup /values.YAML
Release "backup-chart" Happy Helming!
```

5. Proposed Design Working:

Initially after creating the IBM Kubernetes services and configuring all the prerequisites that are associated to deploy a Kubernetes system, we will be writing the Helm-charts. This is the most important and a bit complex structure, as per the Helm-chart designed above we will be having.

- One Master Postgres Node
- Two Slave Postgres Node
- One Backup scheduler pod in Master Node

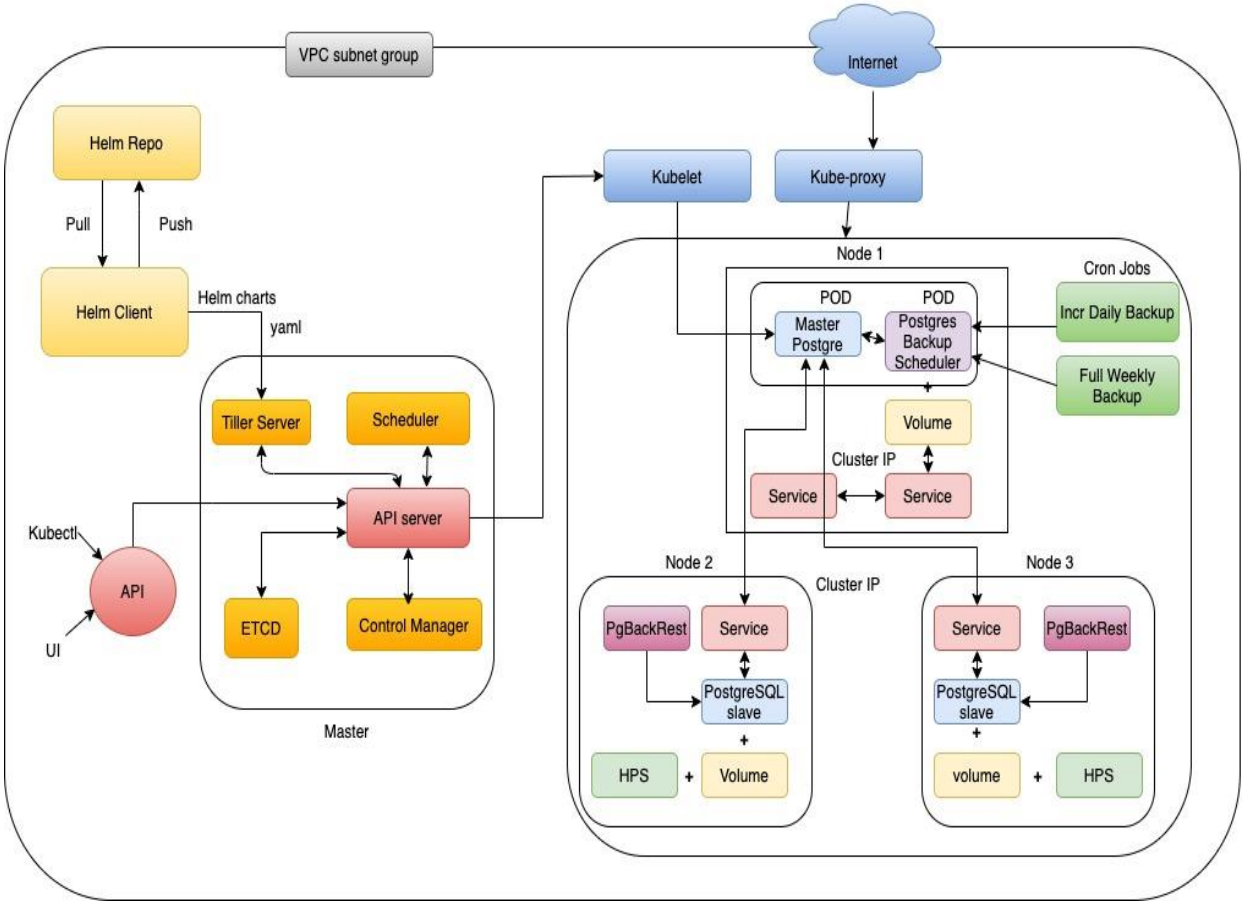

Figure 6: Detailed Architecture

Our Helm charts also deploy the Patroni agent which manages the end-to-end setup to facilitate automatic leader election and failover amongst replicas for HA Postgres. Even if the Primary/Master goes down, a replica will automatically take over as the new primary and the cluster will automatically reconfigure itself to avoid any downtime. Each node will have its own volume, so that even if the pod fails, kubernetes can automatically bring up a new instance of the database, reconnect it to the detached disk volume, recover the database and reintegrate it into the PostgreSQL DB cluster.

The Basic setup of every Pod in a nodes, each pod will be attached will be getting the memory resource that was requested by the **Persistent Volume claim (PVC)** from "IBM provided gid **Storage Classes**". We can also create our own customized storage class. Here, in this design it has been associated with ibmc-file-bronze-gid the default group ID is 65531 which would be specified in the "storageclass.YAML". Moreover, we should also make sure that the storage class should be extendable **"Elastic Memory Capacity"** the reason for this is when the pod utilizes the certain threshold then the system should dynamically extend the storage or should scale up the replicas with the same configuration. In case if the PV or storage is deleted, the data will be gone but helm will recreate it and it will attach with its associated node (all these data will be retained in etcd in Master which will have all the meta-data of a complete system).

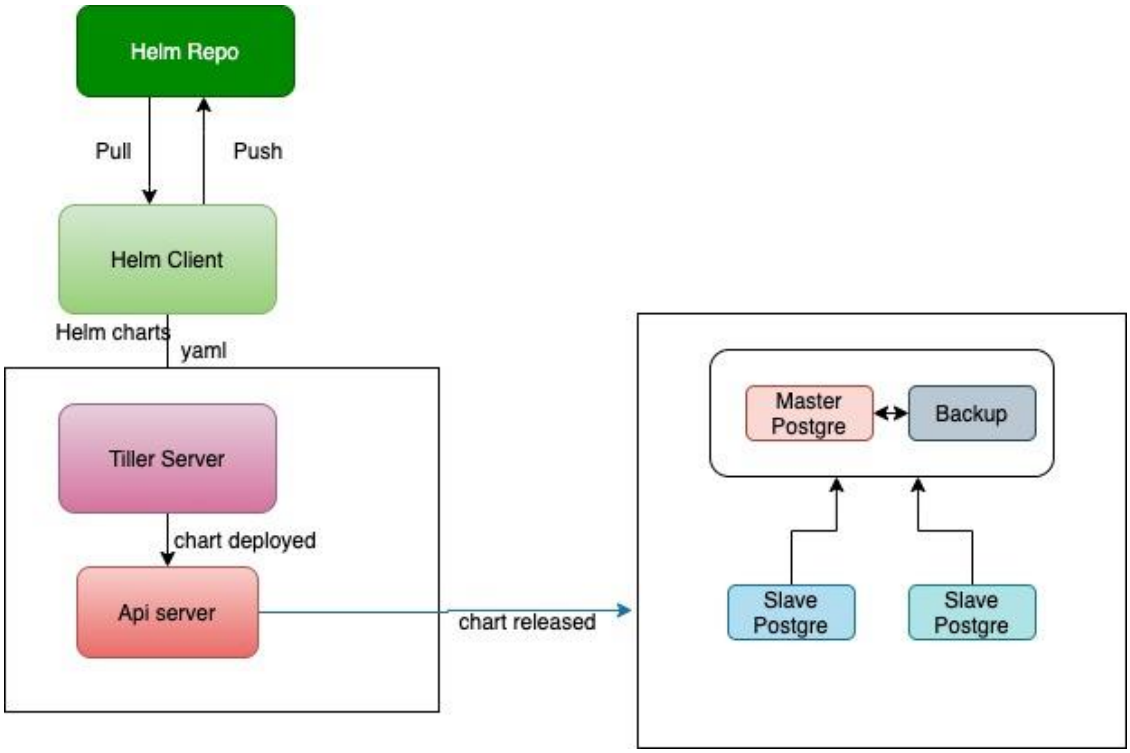

Figure 7:Helm chart released to create Kubernetes system

5.1. Checking Metric:

On the other hand the metric. YAML are configured with the slave pods, by setting the auto-scaling metrics and we have to write rules say, (the slave Postgres Pod utilizes 70% CPU – 70% Memory), the will trigger a request to the Horizontal Pod Auto-scaler (HPA). In addition to it the Deployment/ RC will scale up the slave Pods in the Postgres worker nodes.

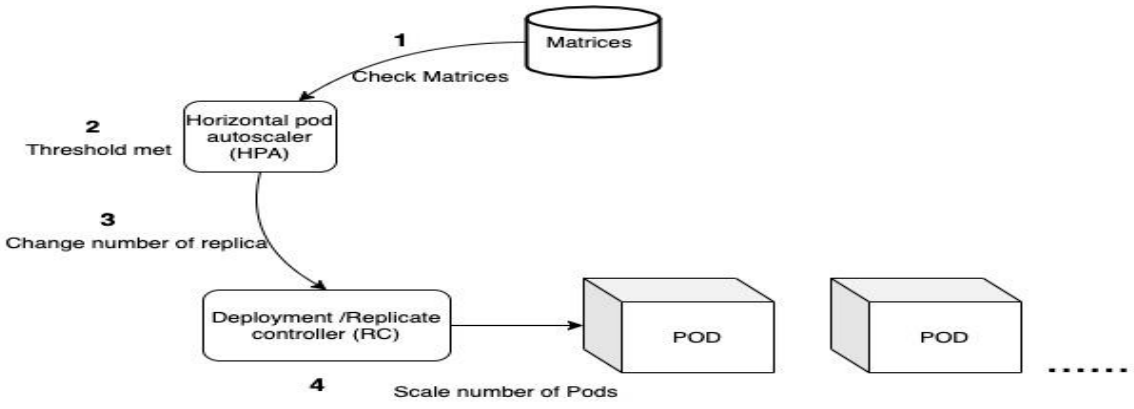

Figure 8: Checking for metrics

Next important component in "**Services",** each and every pod will be having its own service associated with that pod to communicate within the Kubernetes cluster with other nodes or pods. In this design we use Cluster-IP service, each slave Postgres has its Cluster-IP service running which will be connected to the master Pod service of Postgres. In case if the application is publicly accessible, we will be writing services like Node Port to establish the connection. Moreover as the kubernetes network is configured to Loader Balance instance for VPC in IBM cloud, to handle routing incoming traffic to the master pod (Control Balancer Service).

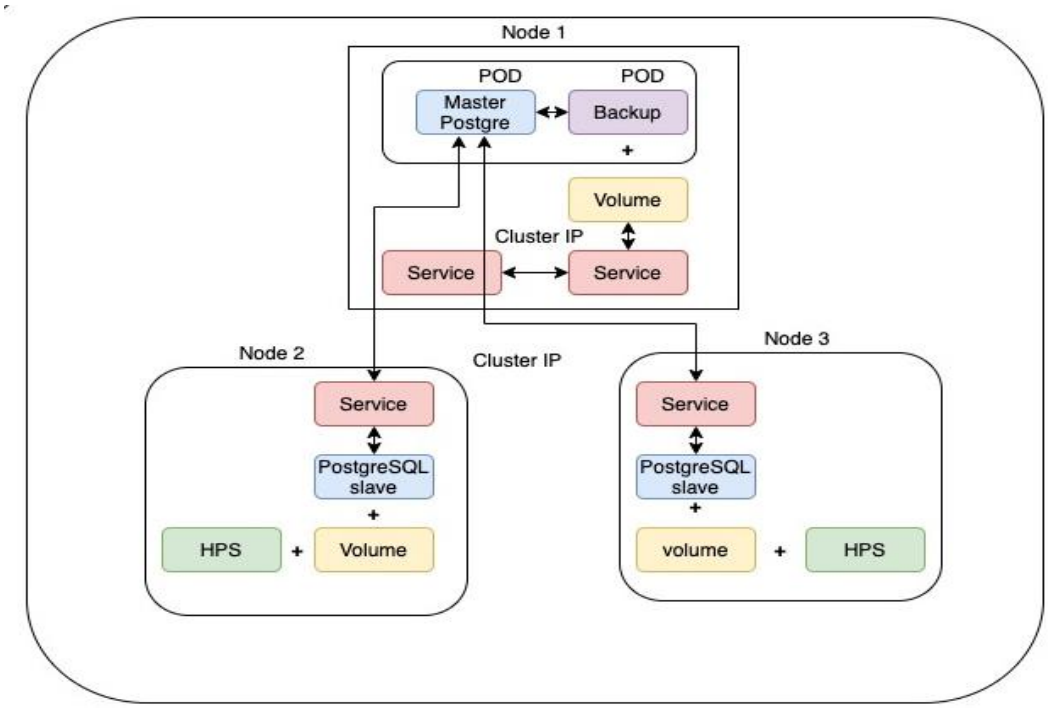

Figure 9: Services in the design system

On the other hand, for creating a Backup scheduler, we have to include a container running pgBackRest in each slave Postgres node. PgBackRest is an open source backup tool that creates different types of backups of the postgres db. Inside the container we have to install the pgBackRest package and can be used from the command line or locate the "pgbackrest.conf" file of pgBackRest. After setting all the required confirmation and parameters that are need to perform the backup. In addition to it 2 Cron Jobs will be created in the Postgres Backup scheduler. They are,

- Full Weekly Backup
- Incremental Daily Backup

So, basically Full Weekly Backup cron jobs will be scheduled to take the full backup of the Postgres Database every week. The primary need to perform this backup is, it copies all data available with a Single set of media. In this design the cloud provider

will handle all the data media. However, we could also integrate a plugin that is available on IBM cloud for storing data of all the resources of the Kubernetes system (PV, service, Pod etc.). An Incremental backup operation will result in coping only the data that has changed since the last backup operation of any type. So, this cron job will be scheduled and invoked daily. We could also track and record the date and time of backup operations. Based on the further usage the cron jobs can be updated and can also perform different kinds of backup that is needed for the application.

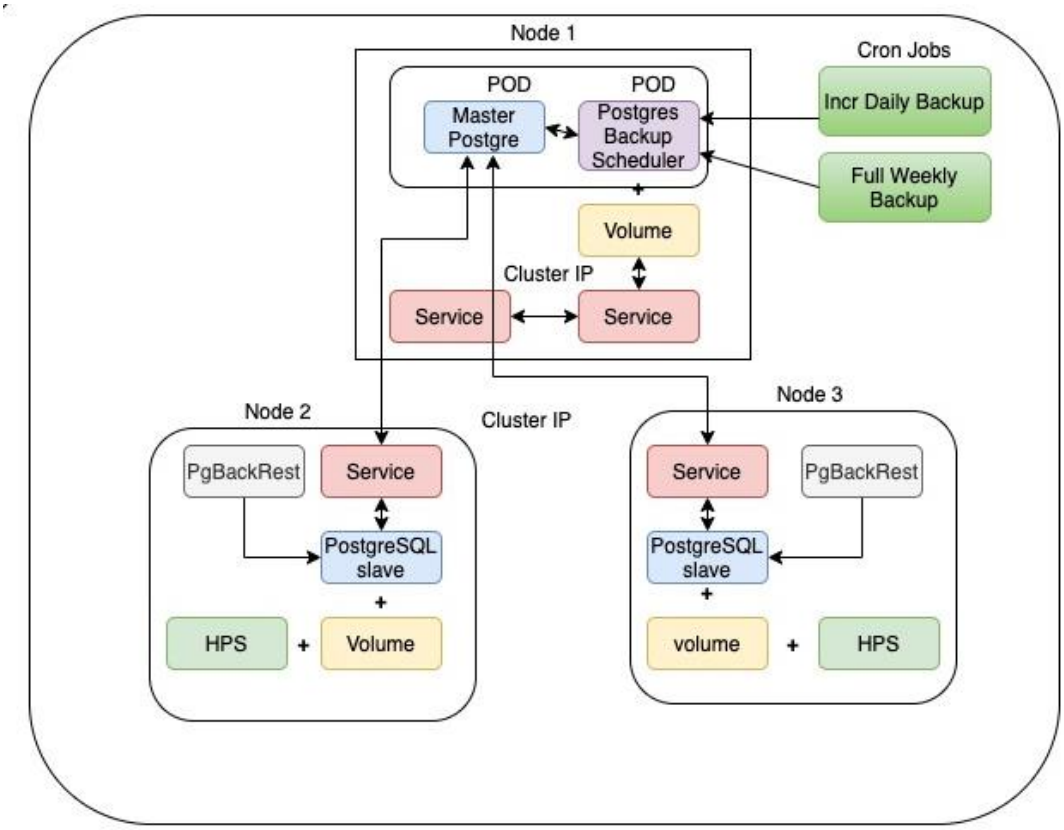

Figure 10: PgBackRest/Cron Jobs

6. Object storage Plug-in:

IBM Object storage Plug-in with a Helm chart can be integrated to set up pre-defined storage classes for IBM Cloud Object Storage for the application data. Store the credentials to access the cloud Object store in Kubernetes "secret.yaml". One of the main advantages of Installing IBM Cloud Object Storage Helm plug-in ibmc is, it has an automatic retrieve of the buckets in the storage classes.

Figure 11: Postgres and IBM cloud

7. Additional Advantage:

Blue/green Deployment

In the above mention system, in case if we want to upgrade it to the new version say from V1 to V2, then it will update the deployment using Blue/green method. The main advantage of this deployment method is Zero Down time, basically it will create a new node and will deploy the pod with the same configuration and new version pod in it (V2). Once, that is deployed successfully then the following node will be updated. For newer release/ update the existing this deployment method is followed.

8. Non-Functional Requirements:

Accessibility:

- The System is used only by the authenticated users or the administrator (in organization).
- IBM Cloud login will generate one time passcode for authentication.

Security:

- IBM Cloud provides high security to maintain the service and data stored
- Authentication is also provides security.

Extensibility:

- The system is designed in such a way that the functions can be extended an update is needed.
- Storage Volume is dynamically called when the system wants to extend.

Reusability:

- The Helm chart templates are been designed in a way that can be modified and redeploy with new values by editing just one file.

- In case of pod failure, the same Persistent Volume (PV) associated to that node will be redeployed to that respective pod.

Readability:

- The YAML file can be viewed and comments should be included in it to have better understanding and readable.
- The YAML should follow the standards based on the version and kind.

Serviceability:

- The Documentation of the system provides Serviceability.

9. Conclusion:

In a nutshell, this design document is to create a Kubernetes system that takes backup for a PostgreSQL database. This design basically, highlights the use of Helm charts and how it can be deploy for the high availability system as well as the major usage of IBM Kubernetes services. IBM cloud provides many plug-in services for storing and retaining the data from an IBM Object storage. However, the above mentioned few steps (4.1 - 4.3) are more important to understand the system and how it is been structured.