Setting up physical replication of PostgreSQL in Kubernetes

Pre-requisites :-

The setup for Kubernetes, Kubectl, and Minikube is complete. You can follow the instructions in this document. [Link]

How to Setup physical replication?

1.Create a Secret named `postgres-secret` in a YAML file of type Opaque, containing a base64-encoded PostgreSQL password under the key `postgres-password`.

```
apiVersion: v1
kind: Secret
metadata:
  name: postgres-secret
type: Opaque
data:
  postgres-password: <password>
```

Applies the configuration in postgres-secret.yaml.

```
kubectl apply -f postgres-secret.yaml
secret/postgres-secret created
```

2. Creates a Kubernetes secret named ghcr-secret in the default namespace using ghcr.io authentication credentials (docker-username, docker-password, and docker-email) for my Docker image.

```
kubectl create secret docker-registry ghcr-secret \
>   --docker-server=ghcr.io \
>   --docker-username=<username> \
>   --docker-password=<Personal Token> \
>   --docker-email=email -n default
secret/ghcr-secret created

```

3. Define a Cluster Role called postgres-cluster-role to manage pods, services, and more, and bind it to the default Service Account via Cluster Role Binding postgres-cluster-role-binding.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: postgres-cluster-role
rules:
- apiGroups: [""]
  resources: ["pods", "services", "endpoints", "persistentvolumeclaims"]
  verbs: ["get", "list", "watch", "create", "update", "patch", "delete"]
- apiGroups: ["apps"]
  resources: ["replicasets"]
  verbs: ["get", "list", "watch", "create", "update", "patch", "delete"]
- apiGroups: ["discovery.k8s.io"]
  resources: ["endpointslices"]
  verbs: ["get", "list", "watch", "create", "update", "patch", "delete"]
---
apiVersion: rbac.authorization.k8s.io/v1
```

```
kind: ClusterRoleBinding
metadata:
  name: postgres-cluster-role-binding
subjects:
- kind: ServiceAccount
  name: default
  namespace: default
roleRef:
  kind: ClusterRole
  name: postgres-cluster-role
  apiGroup: rbac.authorization.k8s.io
```

Applies the configuration in `auth.yaml`, creating or updating the defined Kubernetes resources like roles and role bindings.

```
kubectl apply -f auth.yaml
clusterrole.rbac.authorization.k8s.io/postgres-cluster-role created
clusterrolebinding.rbac.authorization.k8s.io/postgres-cluster-role-binding created
```

4. Defines a PostgreSQL Deployment named `postgres-primary` with one replica, utilizing a custom GHCR image, setting up PostgreSQL credentials and replication in the postStart hook, and creating a Service called `postgres-primary-service` to expose it on port 5432.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: postgres-primary
spec:
  replicas: 1
  selector:
    matchLabels:
      app: postgres-primary
  template:
    metadata:
      labels:
        app: postgres-primary
    spec:
      serviceAccountName: default  # Use the default service account
      imagePullSecrets:
      - name: ghcr-secret
      containers:
      - name: postgres
        image: ghcr.io/imtiazqa/almalinux_pgcustom:latest
        ports:
        - containerPort: 5432
        env:
        - name: POSTGRES_PASSWORD
          valueFrom:
            secretKeyRef:
              name: postgres-secret
              key: postgres-password
        - name: POSTGRES_USER
          value: postgres  # Explicitly setting the default PostgreSQL user
        - name: POSTGRES_DB
          value: postgres  # Specify the default database to be created
        - name: POSTGRES_REPLICATION_USER
          value: postgres  # Replication user
        - name: POSTGRES_REPLICATION_PASSWORD
          valueFrom:
            secretKeyRef:
              name: postgres-secret
```

```
              key: postgres-password  # Use the same password for replication
        securityContext:
          privileged: true  # Enables privileged mode
        lifecycle:
          postStart:
            exec:
              command:
                - "sh"
                - "-c"
                - |
                  # Configuration for replication
                  sed -i "/^#wal_level/s/^#//;s/^wal_level = .*/wal_level = replica/" /var/lib/pgsql/16/data/postgresql.conf
                  sed -i "/^#max_wal_senders/s/^#//;s/^max_wal_senders = .*/max_wal_senders = 10/" /var/lib/pgsql/16/data/postgresql.conf
                  sed -i "/^#wal_keep_segments/s/^#//;s/^wal_keep_segments = .*/wal_keep_segments = 64/" /var/lib/pgsql/16/data/postgresql.conf
                  # Restart PostgreSQL service
                  systemctl restart postgresql-16.service
                  echo "PostStartHook running"
---
apiVersion: v1
kind: Service
metadata:
  name: postgres-primary-service
spec:
  selector:
    app: postgres-primary
  ports:
    - port: 5432
      targetPort: 5432


```

Applies the configuration from postgres-primary.yaml to create or update the defined PostgreSQL deployment and service resources.

```
kubectl apply -f postgres-primary.yaml
deployment.apps/postgres-primary created
```

This command shows the current status of all pods in the Kubernetes cluster.

```
kubectl get pods
NAME                               READY   STATUS    RESTARTS   AGE
postgres-primary-858fb7dcff-tgdhb   1/1     Running   0          3m12s

```

5. The postgres-standby-1-pvc.yaml file defines a PersistentVolumeClaim named postgres-standby-1-pvc, requesting 10 GiB of storage with ReadWriteOnce access via the standard storage class.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: postgres-standby-1-pvc
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
```

```
  storageClassName: standard
```

Creating the `PersistentVolumeClaim called postgres-standby-1-pvc in Kubernetes.`

```
kubectl apply -f postgres-standby-1-pvc.yaml
persistentvolumeclaim/postgres-standby-1-pvc created

```

6. This YAML file creates a Kubernetes Deployment named postgres-standby-1 with two PostgreSQL standby replicas using a custom image. It configures environment variables for the PostgreSQL password and primary host, employs a postStart hook for cleanup and base backup, and includes a Service named postgres-standby-1-service to expose the replicas on port 5432.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: postgres-standby-1
spec:
  replicas: 2
  selector:
    matchLabels:
      app: postgres-standby-1
  template:
    metadata:
      labels:
        app: postgres-standby-1
    spec:
      serviceAccountName: default  # Use the default service account
      imagePullSecrets:
      - name: ghcr-secret
      containers:
      - name: postgres
        image: ghcr.io/imtiazqa/almalinux_pgcustom:latest
        ports:
        - containerPort: 5432
        env:
        - name: POSTGRES_PASSWORD
          valueFrom:
            secretKeyRef:
              name: postgres-secret
              key: postgres-password
        - name: POSTGRES_PRIMARY_HOST
          value: postgres-primary-service  # Service name of the primary node
        - name: POSTGRES_REPLICATION_USER
          value: postgres  # Replication user
        - name: POSTGRES_REPLICATION_PASSWORD
          valueFrom:
            secretKeyRef:
              name: postgres-secret
              key: postgres-password  # Use the same password for replication
        securityContext:
          privileged: true  # Enables privileged mode
        lifecycle:
          postStart:
            exec:
              command:
                - "su"
                - "postgres"
                - "-c"
                - |
```

```
                  # Remove existing files in data directory
                  rm -rf /var/lib/pgsql/16/data/* /var/lib/pgsql/16/data/.[!.]* /var/lib/pgsql/16/data/..?* ||
true
                  # Perform base backup
                  until pg_basebackup -h ${POSTGRES_PRIMARY_HOST} -U ${POSTGRES_REPLICATION_USER} -D
/var/lib/pgsql/16/data/ -Fp -Xs -P -R -W; do
                    echo "Waiting for primary to become available..."
                    sleep 120
                  done
---
apiVersion: v1
kind: Service
metadata:
  name: postgres-standby-1-service
spec:
  selector:
    app: postgres-standby-1
  ports:
    - port: 5432
      targetPort: 5432

```

This command creates a Deployment and a Service named postgres-standby-1, along with the associated Service postgres-standby-1-service, in Kubernetes. named `postgres-standby-1-service` in Kubernetes.

```
kubectl apply -f postgres-standby-1.yaml
deployment.apps/postgres-standby-1 created
service/postgres-standby-1-service created

```

Execute this command to check the status of all pods, including primary and standby.

```
kubectl get pods
NAME                                 READY   STATUS    RESTARTS   AGE
postgres-primary-858fb7dcff-tgdhb    1/1     Running   0          4m32s
postgres-standby-1-6bfb68444b-drt48  1/1     Running   0          42s
postgres-standby-1-6bfb68444b-hx5sq  1/1     Running   0          42s

```

7. Confirm that standby.signal exists on standby pods and that postgresql.auto.conf contains the primary pod information.

```
kubectl exec -it postgres-standby-1-6bfb68444b-drt48 -- bash -c "ls /var/lib/pgsql/16/data/standby.signal"
/var/lib/pgsql/16/data/standby.signal

```

```
kubectl exec -it postgres-standby-1-6bfb68444b-drt48 -- bash -c "cat
/var/lib/pgsql/16/data/postgresql.auto.conf"
# Do not edit this file manually!
# It will be overwritten by the ALTER SYSTEM command.
primary_conninfo = 'user=postgres passfile=''/var/lib/pgsql/.pgpass'' channel_binding=prefer host=''postgres-
primary-service'' port=5432 sslmode=prefer sslcompression=0 sslcertmode=allow sslsni=1
ssl_min_protocol_version=TLSv1.2 gssencmode=prefer krbsrvname=postgres gssdelegation=0 target_session_attrs=any
load_balance_hosts=disable'

```

```
kubectl exec -it postgres-standby-1-6bfb68444b-hx5sq -- bash -c "ls /var/lib/pgsql/16/data/standby.signal"
/var/lib/pgsql/16/data/standby.signal

```

```
kubectl exec -it postgres-standby-1-6bfb68444b-hx5sq -- bash -c "cat
/var/lib/pgsql/16/data/postgresql.auto.conf"
# Do not edit this file manually!
# It will be overwritten by the ALTER SYSTEM command.
primary_conninfo = 'user=postgres passfile=''/var/lib/pgsql/.pgpass'' channel_binding=prefer host=''postgres-
primary-service'' port=5432 sslmode=prefer sslcompression=0 sslcertmode=allow sslsni=1
ssl_min_protocol_version=TLSv1.2 gssencmode=prefer krbsrvname=postgres gssdelegation=0 target_session_attrs=any
load_balance_hosts=disable'

```

8. This command runs a SQL query in the postgres-primary pod to get process IDs and client addresses of two active replication connections from pg_stat_replication.

```
kubectl exec -it postgres-primary-858fb7dcff-tgdhb -- psql -U postgres -d postgres -c "SELECT
pid,client_addr,client_hostname FROM pg_stat_replication;;"
 pid | client_addr | client_hostname
-----+-------------+-----------------
  72 | 10.244.0.5  |
  71 | 10.244.0.4  |
(2 rows)

```

9. Create a table and populate it on the primary pod, then verify replication on the standby pods.

```
kubectl exec -it postgres-primary-858fb7dcff-tgdhb -- psql -U postgres -d postgres
psql (16.4)
Type "help" for help.

postgres=# CREATE TABLE departments (

    dept_no     CHAR(4)         NOT NULL,

    dept_name   VARCHAR(40)     NOT NULL,

    PRIMARY KEY (dept_no),

    UNIQUE      (dept_name)

);

INSERT INTO departments VALUES ('d001','Marketing'),('d002','Finance'),('d003','Human Resources');
CREATE TABLE
INSERT 0 3

```

```
kubectl exec -it postgres-standby-1-6bfb68444b-drt48 -- psql -U postgres -d postgres
psql (16.4)
Type "help" for help.

postgres=# select * from departments ;
 dept_no |    dept_name
---------+-----------------
 d001    | Marketing
 d002    | Finance
 d003    | Human Resources
(3 rows)

```

```
kubectl exec -it postgres-standby-1-6bfb68444b-hx5sq -- psql -U postgres -d postgres
psql (16.4)
Type "help" for help.

postgres=# select * from departments ;
 dept_no |    dept_name
---------+-----------------
 d001    | Marketing
 d002    | Finance
 d003    | Human Resources
(3 rows)


```

Scaling replica pods up and down.

1. Retrieve the list of all deployments.

```
kubectl get deployments
NAME                READY   UP-TO-DATE   AVAILABLE   AGE
postgres-primary    1/1     1            1           36m
postgres-standby-1  2/2     2            2           32m

```

2. Reduce the standby to one replica.

```
kubectl scale deployment --replicas=1 postgres-standby-1
deployment.apps/postgres-standby-1 scaled

```

Show the list of pods after scaling down.

```
kubectl get pods
NAME                                READY   STATUS        RESTARTS   AGE
postgres-primary-858fb7dcff-tgdhb   1/1     Running       0          37m
postgres-standby-1-6bfb68444b-drt48 1/1     Running       0          33m
postgres-standby-1-6bfb68444b-hx5sq 1/1     Terminating   0          33m
[rockylinux@rocky8 28_oc_2024_represume]$

```

```
[rockylinux@rocky8 28_oc_2024_represume]$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
postgres-primary-858fb7dcff-tgdhb   1/1     Running   0          37m
postgres-standby-1-6bfb68444b-drt48 1/1     Running   0          33m

```

Verify the replication after scaling down.

```
kubectl exec -it postgres-primary-858fb7dcff-tgdhb -- psql -U postgres -d postgres
psql (16.4)
Type "help" for help.

postgres=#
postgres=# insert into departments values('d004','test');

```

```
kubectl exec -it postgres-standby-1-6bfb68444b-drt48 -- psql -U postgres -d postgres
psql (16.4)
Type "help" for help.

postgres=# select * from departments ;
 dept_no |    dept_name
---------+-----------------
 d001    | Marketing
 d002    | Finance
 d003    | Human Resources
 d004    | test
(4 rows)

```

3. Increase the number of replicas to two.

Note: If you are unable to connect to the psql session, please start the PostgreSQL service manually.

```
kubectl scale deployment --replicas=2 postgres-standby-1
deployment.apps/postgres-standby-1 scaled
```

Show the list of pods after scaling up.

```
kubectl get pods
NAME                                 READY   STATUS              RESTARTS   AGE
postgres-primary-858fb7dcff-tgdhb    1/1     Running             0          39m
postgres-standby-1-6bfb68444b-drt48  1/1     Running             0          35m
postgres-standby-1-6bfb68444b-rbb58  0/1     ContainerCreating   0          4s

```

```
kubectl get pods
NAME                                 READY   STATUS    RESTARTS   AGE
postgres-primary-858fb7dcff-tgdhb    1/1     Running   0          40m
postgres-standby-1-6bfb68444b-drt48  1/1     Running   0          36m
postgres-standby-1-6bfb68444b-rbb58  1/1     Running   0          82s
```

Verify the replication after scaling up.

```
kubectl exec -it postgres-primary-858fb7dcff-tgdhb -- psql -U postgres -d postgres
psql (16.4)
Type "help" for help.

postgres=# insert into departments values('d005','test2');
INSERT 0 1

```

```
kubectl exec -it postgres-standby-1-6bfb68444b-drt48 -- psql -U postgres -d postgres
psql (16.4)
Type "help" for help.

postgres=# select * from departments ;
 dept_no |    dept_name
---------+-----------------
 d001    | Marketing
 d002    | Finance
 d003    | Human Resources
 d004    | test
 d005    | test2
(5 rows)
```

```

```

```
kubectl exec -it postgres-standby-1-6bfb68444b-rbb58 -- psql -U postgres -d postgres
psql: error: connection to server on socket "/run/postgresql/.s.PGSQL.5432" failed: No such file or directory
    Is the server running locally and accepting connections on that socket?

```

```
kubectl exec -it postgres-standby-1-6bfb68444b-rbb58 "bash"
[root@postgres-standby-1-6bfb68444b-rbb58 /]# systemctl start postgresql-16.service
```

```
[rockylinux@rocky8 28_oc_2024_represume]$ kubectl exec -it postgres-standby-1-6bfb68444b-rbb58 -- psql -U postgres -d postgres
psql (16.4)
Type "help" for help.

postgres=# select * from departments ;
 dept_no |    dept_name
---------+-----------------
 d001    | Marketing
 d002    | Finance
 d003    | Human Resources
 d004    | test
 d005    | test2
(5 rows)

```