**PostgreSQL Streaming Replication**

Streaming replication works by streaming any modifications from Primary Database to Standby Database instances (number of standby databases can be configured based on the requirement). Standby nodes can be used to offload read queries there by reducing the load on Primary Database.

A Standby Database instance can be promoted to Primary in case the Primary Database crashes. It is always advised not to configure Primary and Standby Database instances on the same disk or NFS drive.

Following diagram illustrates a Primary Database configured with 2 Standby Database instances on different NFS drives. Applications can use Primary Database for write operations and redirect reads to Standby instances.

[MuleSoft Code Review
MulSoft Code Analysis ](https://integralzone.com/wp-content/uploads/2020/04/StreamingReplication.png)

**Kubernetes – Configure Streaming Replication**

In this example we will be configuring a Primary Database instance and 1 Standby instance. Two NFS volume mounts will be used namely

* **nfs**– for Primary Database
* **nfs-standby** – for Standby Database

**Setup Primary Database**

Create a config map with environmental variables required for Postgres. Create a new file called *db-config.yaml* and pase the following contents.

apiVersion: v1

kind: ConfigMap

metadata:

name: izdb-config

namespace: default

labels:

type: config

app: izdb

data:

POSTGRES\_PASSWORD: "changeme"

TIMESCALEDB\_TELEMETRY: "off"

Execute the below command to create the config map –

kubectl create -f db-config.yaml

Deployment Pod definition for initialising Postgres primary database. Create a new file called *db-primary.yaml* and paste the below contents –

apiVersion: apps/v1

kind: Deployment

metadata:

name: izdb

spec:

replicas: 1

selector:

matchLabels:

app: izdb

template:

metadata:

labels:

type: deployment

app: izdb

spec:

containers:

- image: postgres:10

imagePullPolicy: IfNotPresent

name: izdb

ports:

- containerPort: 5432

envFrom:

- configMapRef:

name: izdb-config

volumeMounts:

- name: izdbmnt

mountPath: /var/lib/postgresql/data

subPath: izdb/data

volumes:

- name: izdbmnt

persistentVolumeClaim:

claimName: nfs

Execute the below command to create the pod –

kubectl create -f db-primary.yaml

Once **izdb** pod comes up the data directory in nfs i.e. **izdb/data** will be initialised with required files which is sufficient to run a standalone instance of Postgres server. But, to configure replication few parameters have to be changes in **postgresql.conf** and **pg\_hba.conf**

Execute the following commands in the **izdb** pod (use “*kubectl exec -it <pod\_id> bash*” to get into the pod/container). We are going to create a replication user and physical replication slot.

> su - postgres

> psql

# SET password\_encryption = 'scram-sha-256';

# CREATE ROLE repuser WITH REPLICATION PASSWORD 'changeme' LOGIN;

# SELECT \* FROM pg\_create\_physical\_replication\_slot('replica\_1\_slot');

Copy **postgresql.conf** and **pg\_hba.conf** from the **izdb** pod to a local directory called **config**.

$ sudo mkdir config

$ kubectl exec -it izdb-77dfb55dc9-b2jxs -n psql-dev -- cat /var/lib/postgresql/data/pg\_hba.conf > ~/config/pg\_hba.conf

$ kubectl exec -it izdb-77dfb55dc9-b2jxs -n psql-dev -- cat /var/lib/postgresql/data/ postgresql.conf > ~/config/ postgresql.conf

Append following contents to **postgresql.conf**

#Replication Settings

wal\_level = replica

max\_wal\_senders = 2

max\_replication\_slots = 2

synchronous\_commit = off

Append following contents to **pg\_hba.conf**

host replication repuser 0.0.0.0/0 scram-sha-256

**NOTE:** 0.0.0.0/0 can be replaced with the Standby Database ip.

Execute the following command to create conf files map from the **config** directory –

kubectl create configmap izdb-conf-files-config --from-file config

Now lets mount the updated **postgresql.conf** and **pg\_hba.conf** to postgres data directory to enable streaming replication. Replace the contents of *db-primary.yaml* with following –

apiVersion: apps/v1

kind: Deployment

metadata:

name: izdb

spec:

replicas: 1

selector:

matchLabels:

app: izdb

template:

metadata:

labels:

type: deployment

app: izdb

spec:

containers:

- image: postgres:10

imagePullPolicy: IfNotPresent

name: izdb

ports:

- containerPort: 5432

envFrom:

- configMapRef:

name: izdb-config

volumeMounts:

- name: izdbmnt

mountPath: /var/lib/postgresql/data

subPath: izdb/data

volumes:

- name: izdbmnt

persistentVolumeClaim:

claimName: nfs

- name: izdb-conf-files-config-mnt

configMap:

name: izdb-conf-files-config

initContainers:

- name: init-izdb

image: busybox:1.28

command: ['sh', '-c', ' cp /var/lib/postgresql/data/postgresql.main.conf /var/lib/postgresql/data/postgresql.conf && cp /var/lib/postgresql/data/pg\_hba.main.conf /var/lib/postgresql/data/pg\_hba.conf && chmod 600 /var/lib/postgresql/data/postgresql.conf && chmod 600 /var/lib/postgresql/data/pg\_hba.conf']

volumeMounts:

- name: izdbmnt

mountPath: /var/lib/postgresql/data

subPath: izdb/data

- name: izdb-conf-files-config-mnt

mountPath: /var/lib/postgresql/data/postgresql.main.conf

subPath: postgresql.conf

- name: izdb-conf-files-config-mnt

mountPath: /var/lib/postgresql/data/pg\_hba.main.conf

subPath: pg\_hba.conf

Expose the **izdb** pod as a service to be accessed by other applications. Create a new file called *izdb-service.yaml*

apiVersion: v1

kind: Service

metadata:

name: izdb

spec:

type: NodePort

selector:

app: izdb

ports:

- protocol: TCP

port: 5432

targetPort: 5432

Execute the following command to create service –

kubectl create -f izdb-service.yaml

**Setup Standby Database**

Before configuring the Standby instance we need to sync Standby instance with Primary Database state. We will be configuring Standby instance in a different nfs drive called **nfs-standby**

Create a new file called *izdb-standby.yaml* and paste the below contents –

apiVersion: apps/v1

kind: Deployment

metadata:

name: izdb-standby

spec:

replicas: 1

selector:

matchLabels:

app: izdb-standby

template:

metadata:

labels:

type: izdb-standby

app: izdb-standby

spec:

containers:

- image: busybox:1.28

imagePullPolicy: Always

name: izdb-standby

volumes:

- name: izdb-standby-mnt

persistentVolumeClaim:

claimName: nfs-standby

- name: izdb-standby-config-mnt

configMap:

name: izdb-standby-conf-files-config

initContainers:

- name: init-izdb-standby

image: postgres:10

command: ['sh', '-c', 'PGPASSWORD="changeme" pg\_basebackup -h izdb -D /var/lib/postgresql/data -U repuser -vP ']

volumeMounts:

- name: izdb-standby-mnt

mountPath: /var/lib/postgresql/data

subPath: izdb\_standby/data

Execute the following command to create a base backup of Primary Database instance –

kubectl create -f izdb-standby.yaml

**NOTE:** We are using a busy box instance as main container as we should not run Postgres in Primary mode after base backup. After base backup we need to start Postgres in Standby mode.

$ sudo mkdir ~/standby-config

$ sudo touch ~/config/ recovery.conf

$ kubectl exec -it izdb-77dfb55dc9-b2jxs -n psql-dev -- cat /var/lib/postgresql/data/ postgresql.conf > ~/standby-config/postgresql.conf

Copy **postgresql.conf** from the **izdb**-standby pod to a local directory called **standby-config** and append following contents –

hot\_standby = on

wal\_level = replica

max\_wal\_senders = 2

max\_replication\_slots = 2

synchronous\_commit = off

Create a new file called *recovery.conf* in **standby-config**directory with following contents –

standby\_mode = on

primary\_conninfo = 'host=izdb port=5432 user=repuser password=changeme application\_name=r1'

primary\_slot\_name = 'replica\_1\_slot' # Name of the replication slot we created on the master

trigger\_file = '/var/lib/postgresql/data/change\_to\_master'

Execute the following command to create conf files map from the **standby-config**directory –

kubectl create configmap izdb-standby-conf-files-config --from-file standby-config

Delete the pod **izdb-standby** and replace the contents of *izdb-standby.yaml* with following contents

apiVersion: apps/v1

kind: Deployment

metadata:

name: izdb-standby

spec:

replicas: 1

selector:

matchLabels:

app: izdb-standby

template:

metadata:

labels:

type: izdb-standby

app: izdb-standby

spec:

containers:

- image: postgres:10

imagePullPolicy: IfNotPresent

name: izdb-standby

ports:

- containerPort: 5432

envFrom:

- configMapRef:

name: izdb-config

volumeMounts:

- name: izdb-standby-mnt

mountPath: /var/lib/postgresql/data

subPath: izdb\_standby/data

volumes:

- name: izdb-standby-mnt

persistentVolumeClaim:

# use nfs for dev and nfs-standby for prod

#claimName: nfs

claimName: nfs-standby

- name: izdb-standby-config-mnt

configMap:

name: izdb-standby-conf-files-config

initContainers:

- name: init-izdb-standby

image: timescale/timescaledb:latest-pg10

command: ['sh', '-c', 'cp /var/lib/postgresql/data/postgresql.main.conf /var/lib/postgresql/data/postgresql.conf && cp /var/lib/postgresql/data/recovery.main.conf /var/lib/postgresql/data/recovery.conf && chmod 600 /var/lib/postgresql/data/postgresql.conf && chmod 600 /var/lib/postgresql/data/recovery.conf']

volumeMounts:

- name: izdb-standby-mnt

mountPath: /var/lib/postgresql/data

subPath: izdb\_standby/data

- name: izdb-standby-config-mnt

mountPath: /var/lib/postgresql/data/postgresql.main.conf

subPath: postgresql.conf

- name: izdb-standby-config-mnt

mountPath: /var/lib/postgresql/data/recovery.main.conf

subPath: recovery.conf

Execute the following command to create Standby Database instance

kubectl create -f izdb-standby.yaml

COMMAND PSQL

\du = list user

\l = list database

\dt = list tables

\d+ <tableName> = list columns of table

\q = quit

create user mentoruser with encrypted password 'm3nt0rpoliupg';

OR

CREATE ROLE mentoruser LOGIN PASSWORD 'm3nt0rpoliupg';

#Export DB

pg\_dump -U mentoruser mentordb > dbexport.pgsql

OR

pg\_dump -h 10.10.25.11 -p 31624 -U mentoruser mentordb > mentordbBackup.pgsql

#Import DB

pg\_dump -U mentoruser mentordb < dbexport.pgsql

OR

pg\_dump -h 10.10.25.11 -p 31624 -U mentoruser mentordb < mentordbBackup.pgsql