**Senior DBA – Case interview – IPSL 15/04/2025**

---Required Tools/Utilities

* Docker Desktop for virtual environment
* Minikube (small kubernetes setups)
* Helm 3 (bitnami helm chart)
* Chocolatey
* Python (pip)
* Libararies ( tdm,faker,pysopg2)
* PostgreSQL
* Powershell
* Git

**1. Install minikube and start with desired resources**

PS> minikube start --cpus=2 --memory=3072mb --disk-size=40g

**2. Add Bitnami Helm repository**

PS> helm repo add bitnami https://charts.bitnami.com/bitnami

PS> helm repo update

**4. Create namespace for PostgreSQL**

PS> kubectl create namespace database

**5. Create the helm chart and deploy. (** Ipsl-cluster-values.yaml )

architecture: replication

replication:

  enabled: true

  readReplicas:

  user: repl\_user

  password: rep2025

auth:

  username: ipsladm

  password: ipsl2025

  database: ipsldb

primary:

  resources:

    requests:

      cpu: 500m

      memory: 512Mi

  extendedConfiguration: |

    wal\_level = replica

    max\_wal\_senders = 10

    wal\_keep\_size = 64

    hot\_standby = on

  service:

    type: LoadBalancer

pgHbaConfiguration: |-

  host replication repl\_user 0.0.0.0/0 md5

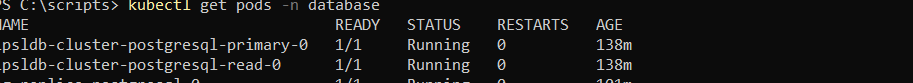
  host all all 0.0.0.0/0 md5

PS > *helm install pg-ipsldb-cluster oci://registry-1.docker.io/bitnamicharts/postgresql -f ipsldb-cluster-values.yaml --namespace database*

|  |  |
| --- | --- |
| Node | Pod name |
| primary | ipsldb-cluster-postgresql-primary-0 |
| replica | ipsldb-cluster-postgresql-read-0 |

PS > kubectl get pods -n database

kubectl get svc -A



**6. Test replication**

PS:> kubectl run -i --rm --tty psql-client --image=bitnami/postgresql – bash

PS: > kubectl exec -it ipsldb-cluster-postgresql-primary-0 -- psql -U postgres -c "SELECT pg\_is\_in\_recovery();" or run directly from psql query SELECT pg\_is\_in\_recovery()

IF it retuns “f” all is well.

psql# psql -h ipsldb-cluster-postgresql-primary-0 -U ipsladm -d ipsldb

**7. Perform a port forward from kubernetes environment to localhost as the script is hosted on my laptop**

PS> *kubectl port-forward svc/pg-ipsldb-cluster-postgresql-primary 5432:5432 -n database*

**6. Create SQL script and run to build the schema ( can include the data-load.py script to load 100000 records)**

import psycopg2

from faker import Faker

import random

from tqdm import tqdm

fake = Faker()

# Connect to the PostgreSQL running in Kubernetes

conn = psycopg2.connect(

    dbname="ipsldb",

    user="ipsladm",

    password="ipsl2025",

    host="localhost",

    port=5432

)

cur = conn.cursor()

# Drop and recreate the tables

cur.execute("""

DROP TABLE IF EXISTS orders;

DROP TABLE IF EXISTS customers;

CREATE TABLE customers (

    id SERIAL PRIMARY KEY,

    name TEXT NOT NULL,

    email TEXT NOT NULL UNIQUE

);

CREATE TABLE orders (

    id SERIAL PRIMARY KEY,

    customer\_id INTEGER NOT NULL,

    product TEXT NOT NULL,

    amount NUMERIC(10,2),

    order\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

    FOREIGN KEY (customer\_id) REFERENCES customers(id)

);

""")

conn.commit()

# Insert 100,000 customers

customer\_ids = []

print("Inserting customers...")

for \_ in tqdm(range(100000)):

    name = fake.name()

    email = fake.unique.email()

    cur.execute(

        "INSERT INTO customers (name, email) VALUES (%s, %s) RETURNING id",

        (name, email)

    )

    customer\_ids.append(cur.fetchone()[0])

conn.commit()

# Insert 100,000 orders

print("Inserting orders...")

for \_ in tqdm(range(100000)):

    customer\_id = random.choice(customer\_ids)

    product = fake.word()

    amount = round(random.uniform(10.0, 10000.0), 2)

    order\_date = fake.date\_time\_between(start\_date='-1y', end\_date='now')

    cur.execute("""

        INSERT INTO orders (customer\_id, product, amount, order\_date)

        VALUES (%s, %s, %s, %s)

    """, (customer\_id, product, amount, order\_date))

conn.commit()

cur.close()

conn.close()

print("Done populating database!")

**7: Initialize Git and push to GitHub**

git init

git add .

git commit -m "IPSL PostgreSQL k8 cluster setup with replication and data loads"

git remote add origin https://github.com/stevosine/ipsl-db.git

git branch -M main

git push -u origin main

**Database Architecture**

A diagram of a data flow

AI-generated content may be incorrect.

* Primary Pod as statefulset handles all write and read operations by default( bitnami helm chart).
* Data replicated to stateful read only replicas using streaming replication and synced to primary.
* Load balancer kubernetes service routes traffic to the appropriate pod.
* Each pod has its own PVC for data persistence.

