Spring Boot + GCP Cloud SQL (PostgreSQL) Deployment Guide

Step 1: Create Cloud SQL PostgreSQL

In Cloud Shell:

Git clone <https://github.com/sreenath20/gcp-postgres-demo.git>

cd gcp-postgres-demo

**Set gcloud environment variables:**

export INSTANCE\_NAME=my-postgres

export REGION=us-central1

export DB\_NAME=mydb

export DB\_USER=postgres

export DB\_PASS=MyPassword123

**Gcloud CLI to create PostgreSQL instance and DB [if not created on gcp web console]:**

gcloud sql instances create $INSTANCE\_NAME \

--database-version=POSTGRES\_14 \

--cpu=1 --memory=4GB \

--region=$REGION \

--root-password=$DB\_PASS

gcloud sql databases create $DB\_NAME --instance=$INSTANCE\_NAME

**Step 2: Configure application.properties**

spring.datasource.url=jdbc:postgresql:///mydb?cloudSqlInstance=PROJECT\_ID:us-central1:my-postgres&soc

ketFactory=com.google.cloud.sql.postgres.SocketFactory

spring.datasource.username=postgres

spring.datasource.password=MyPassword123

spring.datasource.driver-class-name=org.postgresql.Driver

spring.jpa.hibernate.ddl-auto=update

**Step 3: Add Dependencies to build.gradle**

// https://mvnrepository.com/artifact/com.google.cloud.sql/postgres-socket-factory  
*implementation*("com.google.cloud.sql:postgres-socket-factory:1.25.1")

**Step 4: Dockerfile**

# Stage 1: Build the application - multi stage docker for lean build

FROM gradle:8.4-jdk17 AS builder

# Copy project files to container

COPY . /app

# Run Gradle build - no test done to minimise build time

RUN cd /app && gradle clean build -x test

# Stage 2: Run the application - trusted eclipse temurin image for slimmer faster and secure image

FROM eclipse-temurin:17-jre

# Copy the JAR from the builder stage

COPY --from=builder /app/build/libs/\*.jar /app/app.jar

# Expose application port (optional, e.g., 8080)

EXPOSE 8080

# Run the JAR

ENTRYPOINT ["java", "-jar", "/app/app.jar"]

**Step 5: Push Image to Artifact Registry**

gcloud artifacts repositories create springboot-postgres-repo \

--repository-format=docker \

--location=us-central1

**Authenticate Docker**

gcloud auth configure-docker us-central1-docker.pkg.dev

**Build and Push**

docker build -t us-central1-docker.pkg.dev/<PROJECT\_ID>/springboot-postgres-repo/springboot-postgres-app .

note : check your image by [docker images ]

docker push us-central1-docker.pkg.dev/<PROJECT\_ID>/springboot-postgres-repo/springboot-postgres-app

**Step 6: Create Service Account**

gcloud iam service-accounts create springboot-cloudrun-sa \

--display-name "Spring Boot Cloud Run SA"

**Add permission to Spring boot app to access cloud sql [postgress]**

gcloud projects add-iam-policy-binding br1ghter-sun \

--member="serviceAccount:springboot-cloudrun-sa@br1ghter-sun.iam.gserviceaccount.com" \

--role="roles/cloudsql.client"

Step 7: Deploy to Cloud Run

gcloud run deploy springboot-api \

--image=us-central1-docker.pkg.dev/br1ghter-sun/springboot-postgres-repo/springboot-postgres-app \

--platform=managed \

--region=us-central1 \

--allow-unauthenticated \

--add-cloudsql-instances=br1ghter-sun:us-central1:my-postgres \

--service-account=springboot-cloudrun-sa@br1ghter-sun.iam.gserviceaccount.com \

--set-env-vars=SPRING\_PROFILES\_ACTIVE=prod

Optional: Clean Up

gcloud run services delete springboot-api --region=us-central1

gcloud sql instances delete my-postgres