

**Project Diagram**

Project: **qpathways-dev**

Project: **QInsights**

Proxy Client: qinishtsproxyclient

Cloud SQL

Instance: postgres-14-dev

DataStream

(shared VPC across all projects in network) (shared VPC across all projects in network)

Parent Project: **th-corporate**

Shared-VPC: triarqhealth-dev

**GCP Cloud for DataStream Overview**

We have parent project: **th-corporate**. In this project we have VPC name: **“triarqhealth-dev”**. This VPC is shared between other projects across the network.

Cloud SQL Instance name: **“postgres-14-dev”**, This is the postgres SQL instance created in **“qpathways-dev”.** It is created using Shared VPC as we have shared VPC across network. So our source is this Postgres SQL and we have our Bigquery which is our destination in **“QInsights”** project.

We have created Proxy Client instance in **“QInsights”** project named: qinsightsproxycliend. We can able to telnet from these instances to Cloud SQL instance.

(Note While creating this VM enabled IP forwarding, Select shared VPC and subnet in that VPC (to use shared VPC, we cannot select firewall options which shows while creating), IN Access scope: select “Allow full access to all Cloud APIs” also add below script in **“startup script”** option while creating it shows.

#!/bin/bash

wget https://dl.google.com/cloudsql/cloud\_sql\_proxy.linux.amd64 -O cloud\_sql\_proxy

chmod +x cloud\_sql\_proxy

./cloud\_sql\_proxy -instances=qpathways-dev:us-central1:postgres-14-dev=tcp:0.0.0.0:5432

OS installed on these VM : **Linux Debian 11**

In above script in ./cloud\_sql\_proxy.. which is auth proxy command in that “instances=**qpathways-dev:us-central1:postgres-14-dev**=tcp”, which shows here in bold is Cloud SQL instance connection name.

Then SSH in this new VM created. If it ask for “Authorize auth proxy” then click authorize, your official email address will display, just click continue then at linux bash shell prompt, run below command to install postgres client on this VM.

**$ sudo apt-get update**

**$ sudo apt-get install postgresql-client**

**$ !/bin/bash**

**$ sudo apt -y install wget**

**$ wget https://dl.google.com/cloudsql/cloud\_sql\_proxy.linux.amd64 -O cloud\_sql\_proxy**

**$ chmod +x cloud\_sql\_proxy**

**$ ./cloud\_sql\_proxy -instances=qpathways-dev:us-central1:postgres-14-dev=tcp:0.0.0.0:5432**

Then Auth proxy will be in running state in this Proxy client VM, so now open another SSH session for same VM while this keep open current SSH session, so that Auth proxy session will be running then just click on SSH once again on this VM.

Then at Linux prompt enter below command to login to postgres client.

**$ psql -h 127.0.0.1 -U postgres**

Then it will ask for password for this postgres user, enter it and then postgres prompt will show means you are able to login to postgres client here.

Then go to DataStream and create **“Private connectivity configuration”** first. In this we selected VPC, region and it needs available IP range, so we input that and able to create “Private connectivity configuration”.

Also we have created firewall rule for this available IP range we used while creating this **“Private connectivity configuration”.**

Now we have created **Source connection profile** in datastream, to establish connectivity to Source Postgres Cloud SQL Instance which is in

**“postgres-14-dev”,** **“qpathways-dev”** project. (As mentioned above)

While creating this, we have selected Postgres option, then connection name and region and as per suggested setup, IN connection details

we input IP address on Proxy client, which is in same project where datastream. Then we input in port filed DB port: **5432**

Then username and password and database of “postgres Cloud SQL” we input as per fields respectively.

Then we selected connectivity method as Private connectivity (VPC Peering) (As we need to connect using Private IP Address)

And in Private connectivity configuration we selected private connectivity configuration setup name which we created.

After this while running test, it should give below output:

