Project 1: PostgreSQL and Simulator Setup

CSC 6605 ML Production Systems

Overview

In this class, you're going to deploy all of your code to a MSOE server. On that server, each group has a dedicated virtual machine on dh-arm.hpc.msoe.edu. Each virtual machine has 8 cores, 32 GB of RAM, and 256 GB of storage. The username and password for each virtual machine are student and g0!r41d3rs.

You'll use Docker for a few aspects of this. Docker enables you to easily download images for various services you'll need such as the PostgreSQL database. Docker also allows you to test your images on your local machines before deploying to the server.

For this first part of the project, you'll connect to your virtual machine and start some services with Docker. Your instructor will log into your virtual machine and check that everything is running correctly.

Instructions

Part I: Connect to the Server

1. You'll need to be connected to MSOE's VPN to access the machine. Setup VPN access on your machine.

2. SSH into dh-arm.hpc.msoe.edu using your group's port:

SSH is port mapped onto the host system so that you can access your virtual machine by port:

```
$ ssh -p 22GG student@dh-arm.hpc.msoe.edu
```

where GG is your two digit group number. For example, group 4 would connect to:

```
$ ssh -p 2204 student@dh-arm.hpc.msoe.edu
```

The virtual machines are running Debian 12 (buster) and configured with Docker, various editors (emacs, vim, etc.), git, Python, tmux, and a basic set of compilers. Your account has sudo privileges, so you should be able to install anything else you may need via the apt package manager.

The VMs will be used primarily for deploying your system. They allow you to keep services such as databases running 24x7 for convenience. If you want to access services running inside the virtual machine from your local computer, use SSH port forwarding.

For example, to access PostgreSQL on your local computer:

```
$ ssh -L 5432:localhost:5432 -p 2204 student@dh-arm.hpc.msoe.edu
```

While the connection is open, you can use PostgreSQL by attaching to port 5432 on your local machine. (Note that you will generally access things by running programs on the VM's command-line interface.)

Part II: Start PostgreSQL using Docker

Most of the services we'll be using are going to be run as Docker containers. These avoids installing services system wide and makes it easy to restart services from scratch.

Okay, let's start by deploying the PostgreSQL database.

1. Create a Docker network

```
$ docker network create home-sale-event-system
```

2. Start the PostgreSQL service

```
$ docker run --name postgres --network home-sale-event-system -d -e
POSTGRES_PASSWORD="psql-password" -p 5432:5432 docker.io/postgres:latest
```

A few notes about the options:

- --name postgres specifies the name of the container. This will be the hostname that other containers attached to the same Docker network can use to access PostgreSQL.
- --network home-sale-event-system indicates that the container should be attached to the home-sale-event-system Docker network.
- -d indicates that the container should detach and run in the background
- -e VARIABLE=VALUE allows you to specify environmental variables. It's a convention in cloud applications to pass secrets like usernames and passwords via environmental variables. Here, we are setting the password for the "postgres" admin user to "psql-password".
- -p 5432:5432 map port 5432 inside the container to port 5432 on the host system. This will allow you to access Postgres outside of the containers.
- docker.io/postgres:latest specifies that we'll use an image hosted on Dockerhub

Check that the PostgreSQL service is running:

```
$ docker ps
CONTAINER ID      IMAGE COMMAND     CREATED     STATUS      PORTS NAMES
2effd8f10ac9   postgres:latest        "docker-entrypoint.s…"   2 weeks ago
Up 6 days     0.0.0.0:5432->5432/tcp   postgres
```

Part III: Create the Database and Tables for the Storing the Home Sale Events

Next, we'll connect to the PostgreSQL instance and create the necessary database and tables. First, you'll need to install the PostgreSQL client in your VM:

```
$ sudo apt install postgresql-client
```

It will ask for your password and then will confirm that you want to install the packages.

Next, start the PostgreSQL client:

```
$ psql -h 127.0.0.1 -U postgres postgres
Password for user postgres:
psql (15.10 (Debian 15.10-0+deb12u1), server 17.2 (Debian 17.2-1.pgdg120+1))
WARNING: psql major version 15, server major version 17.
         Some psql features might not work.
Type "help" for help.

postgres=#
```

You can then issue SQL and PostgreSQL-specific commands to create the database and switch to it:

```
postgres=# CREATE DATABASE house_price_prediction_service;
CREATE DATABASE
postgres=# \c house_price_prediction_service
psql (15.10 (Debian 15.10-0+deb12u1), server 17.2 (Debian 17.2-1.pgdg120+1))
WARNING: psql major version 15, server major version 17.
         Some psql features might not work.
You are now connected to database "house_price_prediction_service" as user
"postgres".
house_price_prediction_service=#

Lastly, you can create your tables:

house_price_prediction_service=# CREATE TABLE raw_home_sale_events (
                           id serial,
                           data JSONB NOT NULL,
                           event_date date NOT NULL
                           );

CREATE TABLE
house_price_prediction_service=# \d
                      List of relations
 Schema |              Name               |   Type   |  Owner
--------+---------------------------------+----------+----------
 public | raw_home_sale_events            | table    | postgres
 public | raw_home_sale_events_id_seq     | sequence | postgres
(2 rows)

house_price_prediction_service=# \q
```

Make sure that you create an index on the id column.

Part IV: Start the Home Sale Event Simulator

Next, start the simulator:

```
$ docker run --name "home-sale-event-simulator" --network home-sale-event-
system -d -e POSTGRES_USERNAME="postgres" -e POSTGRES_PASSWORD="psql-
```

```
password" -e POSTGRES_HOST="postgres" dh-arm.hpc.msoe.edu:5000/home-sale-
event-simulator
```

Check the simulator is running:

```
$ docker ps
CONTAINER ID   IMAGE                  COMMAND                  CREATED
STATUS        PORTS                   NAMES
0dc67f28a403   sale-event-simulator   "python3 simulator.p…"   6 days ago
Up 6 days                             sale-event-simulator
2effd8f10ac9   postgres:latest        "docker-entrypoint.s…"   2 weeks ago
Up 6 days     0.0.0.0:5432->5432/tcp   postgres
```

You can check that the simulator is working by looking for records in the database:

```
$ psql -h 127.0.0.1 -U postgres house_price_prediction_service
Password for user postgres:
psql (15.10 (Debian 15.10-0+deb12u1), server 17.2 (Debian 17.2-1.pgdg120+1))
WARNING: psql major version 15, server major version 17.
         Some psql features might not work.
Type "help" for help.

house_price_prediction_service=# SELECT * FROM raw_home_sale_events LIMIT 10;
```

Demonstration and Submission

Submit screenshots of the following to Canvas:

- `docker ps` output showing the two containers running for at least 10 minutes
- PostgreSQL `\d tablename` output for your table
- Output from selecting 10 records from your table

Rubric

This assignment is pass / fail. Everything must be working to receive credit.