Guided Exercise: Working with Databases

Set up a containerized PostgreSQL database for development and testing environments.

Outcomes

You should be able to:

- Create a containerized database that contains ephemeral data.
- Load the database schema and data on container creation.
- Use the PostgreSQL client in the PostgreSQL container to interact with the database.
- Create a containerized persistent database for a testing environment.
- Migrate the testing environment data to a container running a newer PostgreSQL version.

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command copies the database initialization scripts of a Raspberry Pi store to the lab directory in your system.

```
[student@workstation ~]$ lab start persisting-databases
```

Instructions

1. Create a PostgreSQL 12 container and call it `persisting-pg12`. Use the `load_db.sh` script in the lab directory to load the data.

   Examine the `/home/student/DO188/labs/persisting-databases/load_db.sh` script, which loads the database schema definitions and the data into the `model` and `inventory` tables.

   Create a PostgreSQL 12 container that runs interactively and call it `persisting-pg12`.

   Use the `registry.ocp4.example.com:8443/rhel8/postgresql-12:1-113` image.

   Mount the `/home/student/DO188/labs/persisting-databases` host directory into the `/opt/app-root/src/postgresql-start` container directory as a bind mount, with the `Z` option for SELinux.

   Provide the following environment variables:

   - `POSTGRESQL_USER=backend`
   - `POSTGRESQL_PASSWORD=secret_pass`
   - `POSTGRESQL_DATABASE=rpi-store`

   ```
   [student@workstation ~]$ podman run -it --rm \
     --name persisting-pg12 \
     -e POSTGRESQL_USER=backend \
     -e POSTGRESQL_PASSWORD=secret_pass \
     -e POSTGRESQL_DATABASE=rpi-store \
     -v ~/DO188/labs/persisting-databases:/opt/app-root/src/postgresql-start:Z \
     registry.ocp4.example.com:8443/rhel8/postgresql-12:1-113
   ...output omitted...
   server started
   /var/run/postgresql:5432 - accepting connections
   => sourcing /opt/app-root/src/postgresql-start/load_db.sh ...
   ...output omitted...
   ```

   On start, containers based on the `rhel8/postgresql-12` image run any scripts that end in `.sh`, found in the `/opt/app-root/src/postgresql-start` directory.

   Open a new terminal and run the `psql` PostgreSQL client in the container to query the `model` table.

   ```
   [student@workstation ~]$ podman exec -it persisting-pg12 \
     psql -d rpi-store -c "select * from model"

   id|     name          |  model |    soc    | memory_mb | ethernet |release_date
   ------------------------------------------------------------------------
   1 | Raspberry Pi      | B      | BCM2835   |       256 | t        |        2012
   2 | Raspberry Pi Zero | Zero   | BCM2835   |       512 | f        |        2015
   3 | Raspberry Pi Zero | 2W     | BCM2710A1 |       512 | f        |        2021
   4 | Raspberry Pi 4    | B      | BCM2711   |      4096 | t        |        2019
   6 | Raspberry Pi 4    | 400    | BCM2711   |      4096 | t        |        2020
   (5 rows)
   ```

2. Recreate the previous container to verify that the data loaded into it is not persistent.

Exit the `persisting-pg12` container by pressing **Ctrl**+**C**. The container is removed automatically because of the `--rm` option provided in the container creation.

Recreate the `persisting-pg12` container without loading the database scripts.

```
[student@workstation ~]$ podman run -it --rm \
  --name persisting-pg12 \
  -e POSTGRESQL_USER=backend \
  -e POSTGRESQL_PASSWORD=secret_pass \
  -e POSTGRESQL_DATABASE=rpi-store \
  registry.ocp4.example.com:8443/rhel8/postgresql-12:1-113
...output omitted...
```

In the previous terminal window, verify that the data you loaded in the preceding step is not present by querying the `model` table.

```
[student@workstation ~]$ podman exec -it persisting-pg12 \
  psql -d rpi-store -c "select * from model"
ERROR:  relation "model" does not exist
LINE 1: select * from model
```

3. Create a containerized PostgreSQL database for a testing environment. Use a volume to avoid losing data if the container is recreated.

   Exit the `persisting-pg12` container by pressing **Ctrl**+**C**. The container is removed automatically because of the `--rm` option.

   Create a volume called `rpi-store-data` to store the database files.

   ```
   [student@workstation ~]$ podman volume create rpi-store-data
   rpi-store-data
   ```

   Create the `persisting-pg12` container in detached mode and map the `rpi-store-data` volume to the `/var/lib/pgsql/data` directory in the container. Bind mount the `postgresql-start` directory like you did previously.

   ```
   [student@workstation ~]$ podman run -d \
     --name persisting-pg12 \
     -e POSTGRESQL_USER=backend \
     -e POSTGRESQL_PASSWORD=secret_pass \
     -e POSTGRESQL_DATABASE=rpi-store \
     -v rpi-store-data:/var/lib/pgsql/data \
     -v ~/DO188/labs/persisting-databases:/opt/app-root/src/postgresql-start:Z \
     registry.ocp4.example.com:8443/rhel8/postgresql-12:1-113
   c99b...7b7c
   ```

   The `/var/lib/pgsql/data` is the directory where the `rhel8/postgresql-12` image is configured to store the database files.

   Verify that the `rpi-store` database contains data by querying the `model` table.

   ```
   [student@workstation ~]$ podman exec -it persisting-pg12 \
     psql -d rpi-store -c "select * from model"
   ...output omitted...
   ```

4. Recreate the container to test that the data inserted in the `rpi-store` database persists after container recreation.

   Remove the container called `persisting-pg12`. Provide the `-f` option to stop and remove the container with the same command.

   ```
   [student@workstation ~]$ podman rm -f persisting-pg12
   persisting-pg12
   ```

   Recreate the `persisting-pg12` container and attach the `rpi-store-data` volume without binding the data loading scripts.

   ```
   [student@workstation ~]$ podman run -d \
     --name persisting-pg12 \
     -e POSTGRESQL_USER=backend \
     -e POSTGRESQL_PASSWORD=secret_pass \
     -e POSTGRESQL_DATABASE=rpi-store \
     -v rpi-store-data:/var/lib/pgsql/data \
     registry.ocp4.example.com:8443/rhel8/postgresql-12:1-113
   e37a...d0cb
   ```

   Query the `model` table to verify that the data is preserved.

```
[student@workstation ~]$ podman exec -it persisting-pg12 \
  psql -d rpi-store -c "select * from model"
...output omitted...
(5 rows)
```

5. Start a containerized pgAdmin interface to manage the database by using a web UI.

   Create a network to allow DNS name resolution between the pgAdmin and database containers.

```
[student@workstation ~]$ podman network create persisting-network
persisting-network
```

   Delete the existing `persisting-pg12` container.

```
[student@workstation ~]$ podman rm -f persisting-pg12
persisting-pg12
```

   Recreate the `persisting-pg12` container. The container must use the network called `persisting-network`.

```
[student@workstation ~]$ podman run -d \
  --name persisting-pg12 \
  -e POSTGRESQL_USER=backend \
  -e POSTGRESQL_PASSWORD=secret_pass \
  -e POSTGRESQL_DATABASE=rpi-store \
  -v rpi-store-data:/var/lib/pgsql/data \
  --network persisting-network \
  registry.ocp4.example.com:8443/rhel8/postgresql-12:1-113
ab8f...ce8c
```

   Create the pgAdmin container and call it `persisting-pgadmin`. Attach the container to the `persisting-network` to resolve the database container by using the `persisting-pg12` name.

   Use the `registry.ocp4.example.com:8443/crunchydata/crunchy-pgadmin4:ubi8-4.30-1` image.

   Map the `5050` container port to same host port.

   Use the following environment variables for the container:

   - `PGADMIN_SETUP_EMAIL=gls@example.com`
   - `PGADMIN_SETUP_PASSWORD=pga_secret_pass`

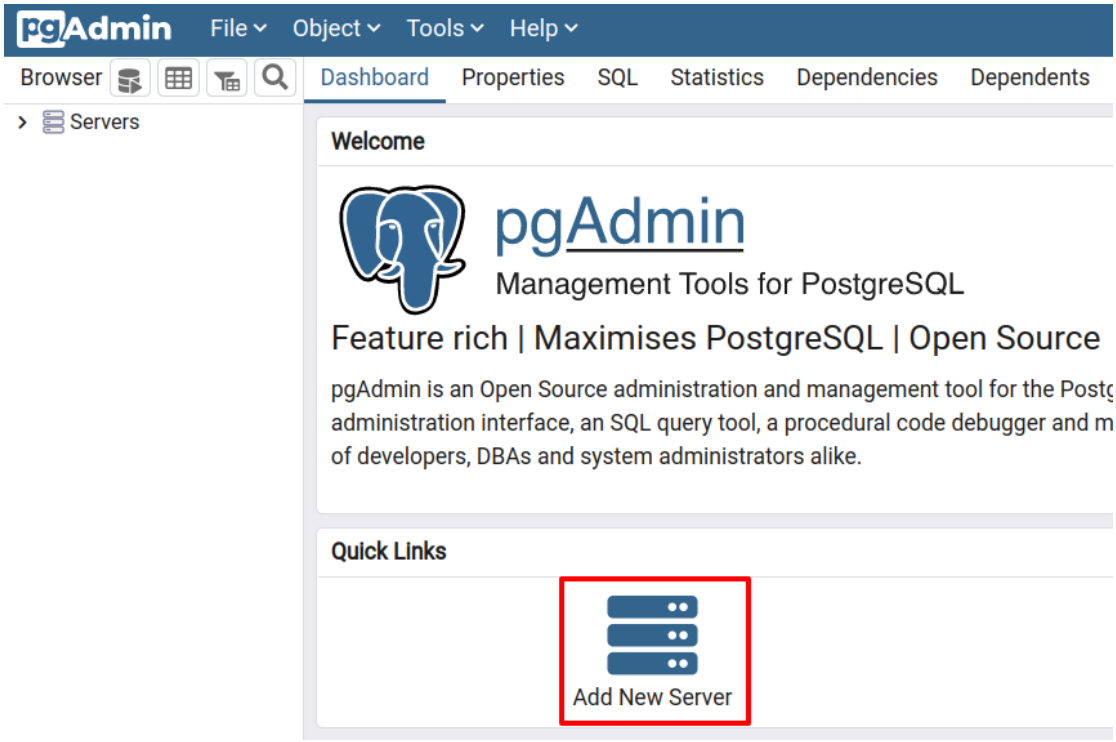
```
[student@workstation ~]$ podman run -d \
  --name persisting-pgadmin \
  -e PGADMIN_SETUP_EMAIL=gls@example.com \
  -e PGADMIN_SETUP_PASSWORD=pga_secret_pass \
  -p 5050:5050 \
  --network persisting-network \
  registry.ocp4.example.com:8443/crunchydata/crunchy-pgadmin4:ubi8-4.30-1
...output omitted...
cd9g...aa8f
```

   > **NOTE**
   >
   > After executing the previous command, you might see SELinux warnings. Those warnings are safe to ignore.

   Log in to pgAdmin by using a web browser and going to `http://localhost:5050`. Log in with the credentials that you used to start the container:

   - Email Address/Username: `gls@example.com`
   - Password: `pga_secret_pass`

   Connect to the `persisting-pg12` database container by clicking **Add New Server**.



On the `General` tab, set `rpi-store` as the name.

Switch to the `Connection` tab. Fill the form with the following data and leave the rest of the fields with their default values.

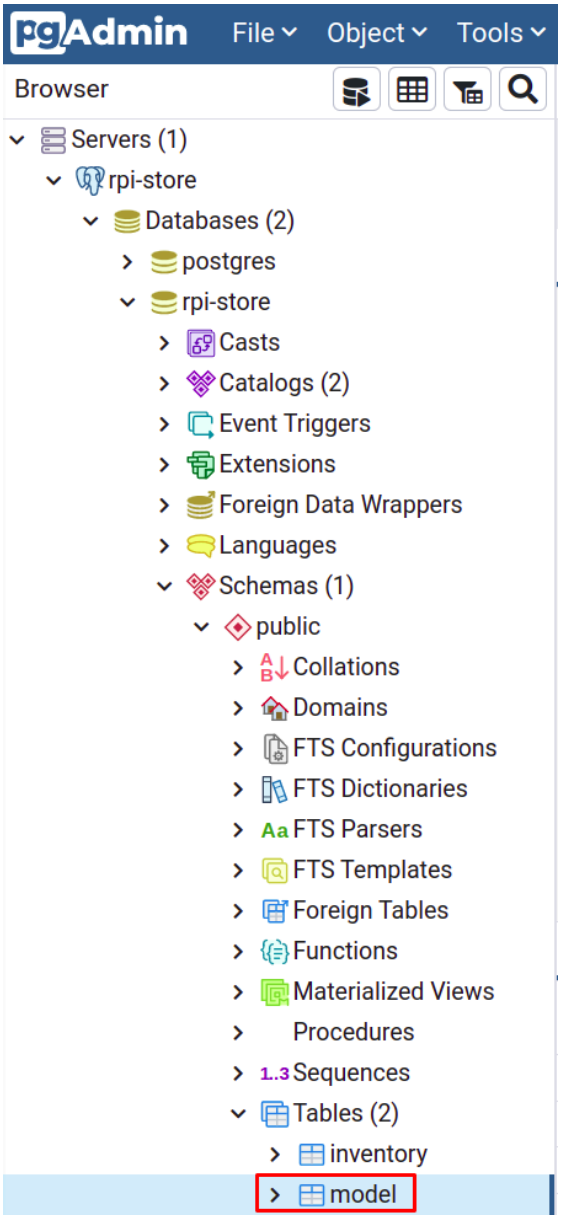
Field	**Value**
Hostname/address	`persisting-pg12`
Username	`backend`
Password	`secret_pass`

Click **Save**. The application verifies the connection before exiting the form.

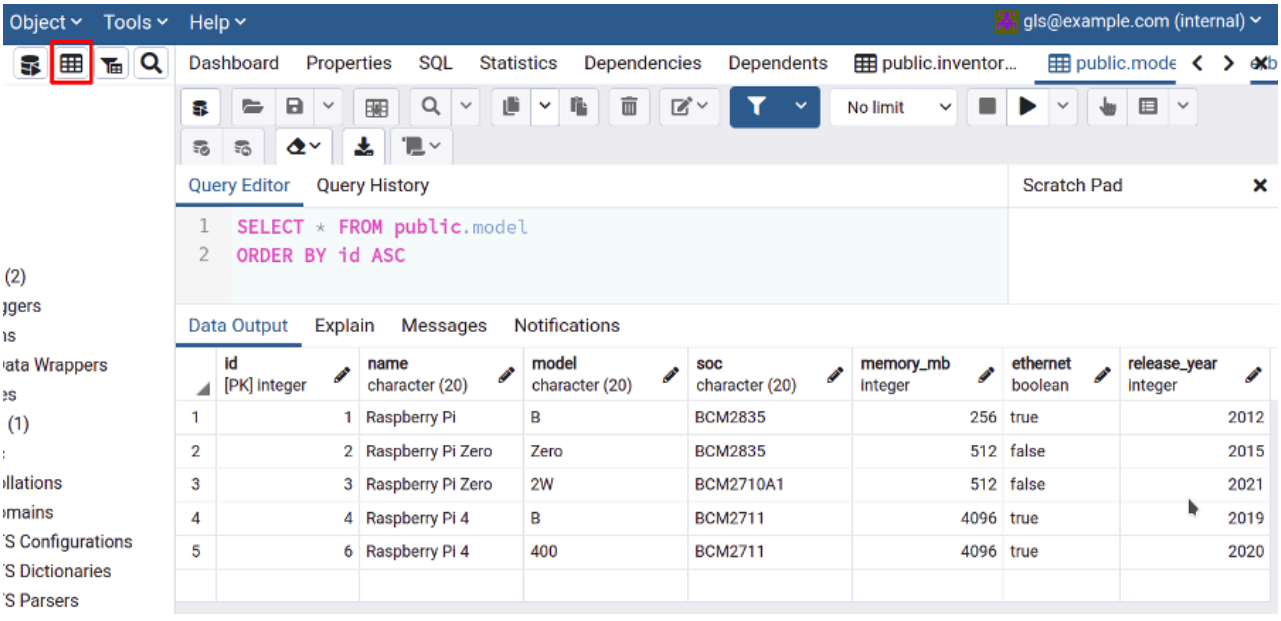
> **NOTE**
>
> Because the database is listening on the host machine, you might assume that the `localhost` value works for the `Host name` field. This does not work because pgAdmin runs in a container and `localhost` refers to the pgAdmin container itself.

View the data in the `model` table to verify the pgAdmin access to the `rpi-store` database in the `persisting-pg12` container.

Select the `model` table by clicking **Servers** > **rpi-store** > **Databases** > **rpi-store** > **Schemas** > **public** > **Tables** > **model**.



Click the **View Data** icon to verify that pgAdmin queries the data in the `persisting-pg12` container.



6. Migrate the testing environment data to a container running PostgreSQL 13, a newer version of the PostgreSQL database. Use a new data volume to avoid incompatibilities between the PostgreSQL versions.

Run the `pg_dump` command in the running database container to back up the `rpi-store`. A file called `/tmp/db_dump` stores the backup in the `persisting-pg12` container. Use the `-Fc` option to use the `custom` format which compresses the backup file.

```
[student@workstation ~]$ podman exec persisting-pg12 \
  pg_dump -Fc rpi-store -f /tmp/db_dump
no output expected
```

Use the `podman cp` command to copy the dump file to the host machine.

```
[student@workstation ~]$ podman cp persisting-pg12:/tmp/db_dump /tmp/db_dump
no output expected
```

Stop the `persisting-pg12` container.

```
[student@workstation ~]$ podman stop persisting-pg12
persisting-pg12
```

> **NOTE**
>
> You might see a `StopSignal` warning when you stop the container. You can safely ignore this warning.

Create a volume called `rpi-store-data-pg13` to store the database files.

```
[student@workstation ~]$ podman volume create rpi-store-data-pg13
rpi-store-data-pg13
```

Create a PostgreSQL 13 container that uses the `rpi-store-data-pg13` volume.

```
[student@workstation ~]$ podman run -d \
  --name persisting-pg13 \
  -e POSTGRESQL_USER=backend \
  -e POSTGRESQL_PASSWORD=secret_pass \
  -e POSTGRESQL_DATABASE=rpi-store \
  -v rpi-store-data-pg13:/var/lib/pgsql/data \
  registry.ocp4.example.com:8443/rhel9/postgresql-13:1
00e9...36a8
```

Copy the dump file to the new container by using the `podman cp` command.

```
[student@workstation ~]$ podman cp /tmp/db_dump persisting-pg13:/tmp/db_dump
no output expected
```

Run `pg_restore` to restore the `rpi-store` database in the `persisting-pg13` container. Set `rpi-store` as the destination database by using the `-d` option.

```
[student@workstation ~]$ podman exec persisting-pg13 \
  pg_restore -d rpi-store /tmp/db_dump
no output expected
```

Query the `model` table in the `persisting-pg13` container to validate that the migration worked.

```
[student@workstation ~]$ podman exec -it persisting-pg13 \
  psql -d rpi-store -c "select * from model"
...output omitted...
(5 rows)
```

Remove the container with the previous PostgreSQL version and its associated volume to eliminate unused resources.

```
[student@workstation ~]$ podman rm persisting-pg12
persisting-pg12
```

Because named volumes persist after deleting the container, you must delete the `rpi-store-data` volume manually.

```
[student@workstation ~]$ podman volume rm rpi-store-data
rpi-store-data
```

Finish

On the `workstation` machine, use the `lab` command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish persisting-databases
```