Hands-on Lab: Backup and Restore using PostgreSQL

Estimated time needed: 30 minutes

In this lab, you will learn how to use the PostgreSQL Command Line Interface (CLI) to restore a full database from a backup. Then using a combination of the CLI and pgAdmin, which is a Graphical User Interface (GUI) for postgreSQL, you will make some changes to this database and perform a full backup. Finally, you will then delete this database to practice a full restoration in the scenario of an accidental deletion.

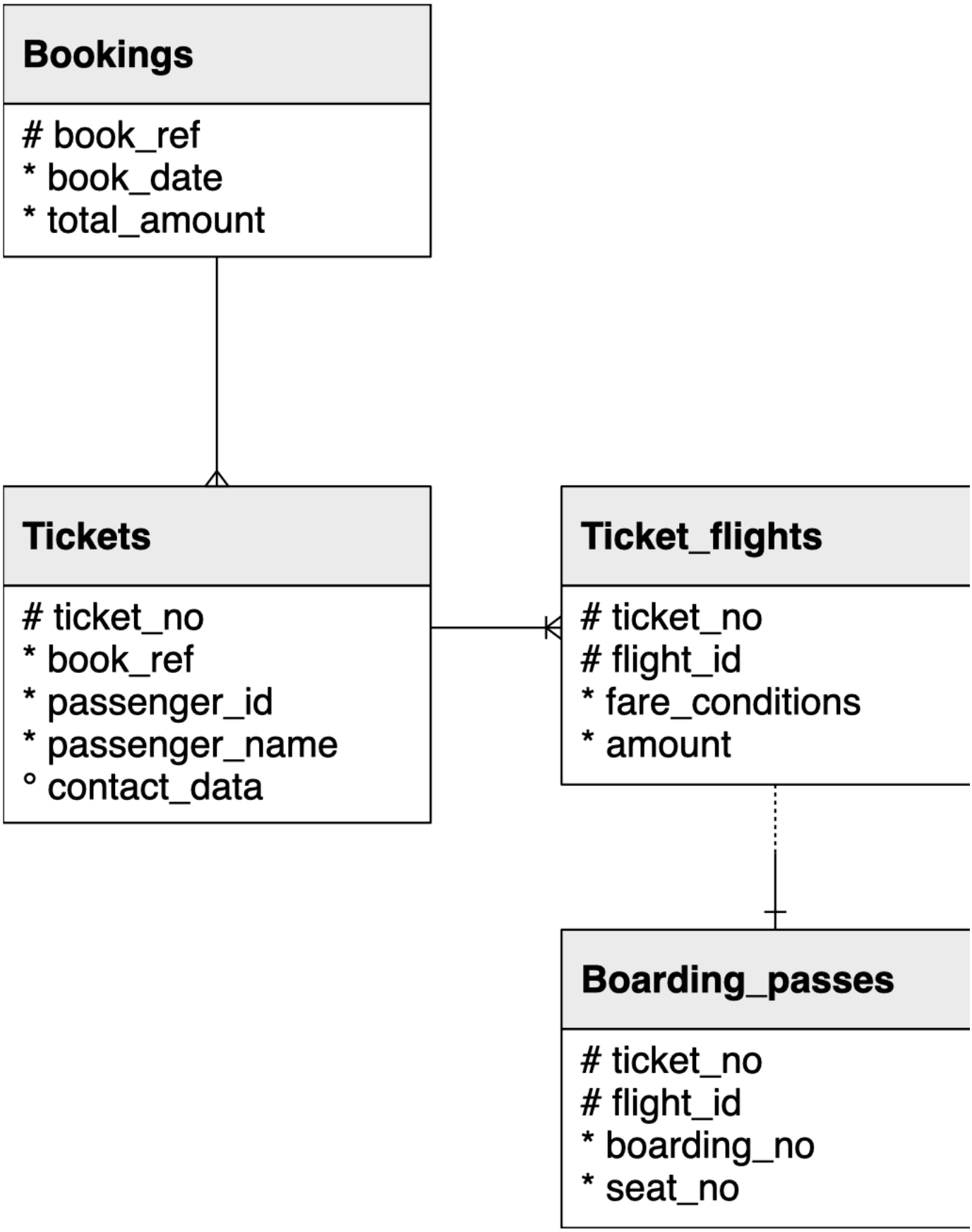
Software used in this Lab

In this lab, you will be using PostgreSQL. It is a popular open-source object Relational Database Management System (RDBMS) capable of performing a wealth of database administration tasks, such as storing, manipulating, retrieving, and archiving data.

To complete this lab, you will be accessing the PostgreSQL service through the IBM Skills Network (SN) Cloud IDE, which is a virtual development environnement you will utilize throughout this course.

Database used in this Lab

In this lab, you will use a database from https://postgrespro.com/education/demodb distributed under the PostgreSQL licence. It stores a month of data about airline flights in Russia and is organized according to the following schema:



Objectives

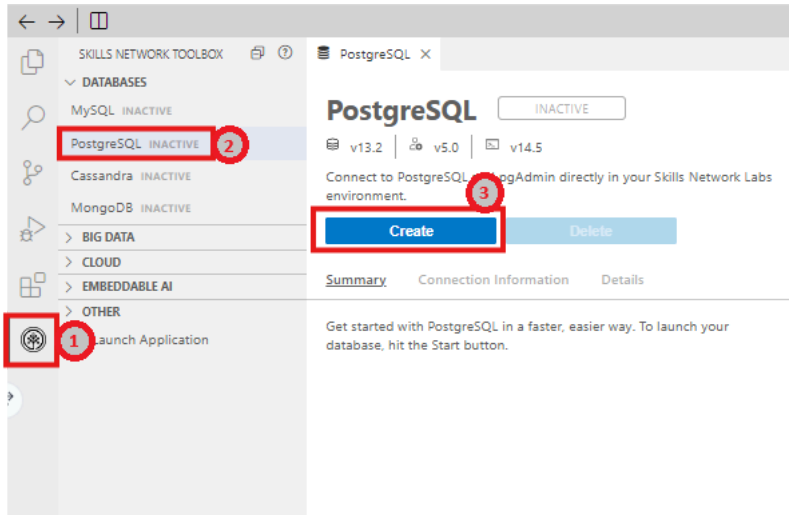
After completing this lab, you will be able to use the PostgreSQL CLI and pgAdmin to:

- Restore a full database from a backup
- Update a database and perform a full backup
- Drop a database and then restore it

Launching PostgreSQL in Cloud IDE

To get started with this lab, launch PostgreSQL using the Cloud IDE. You can do this by following these steps:

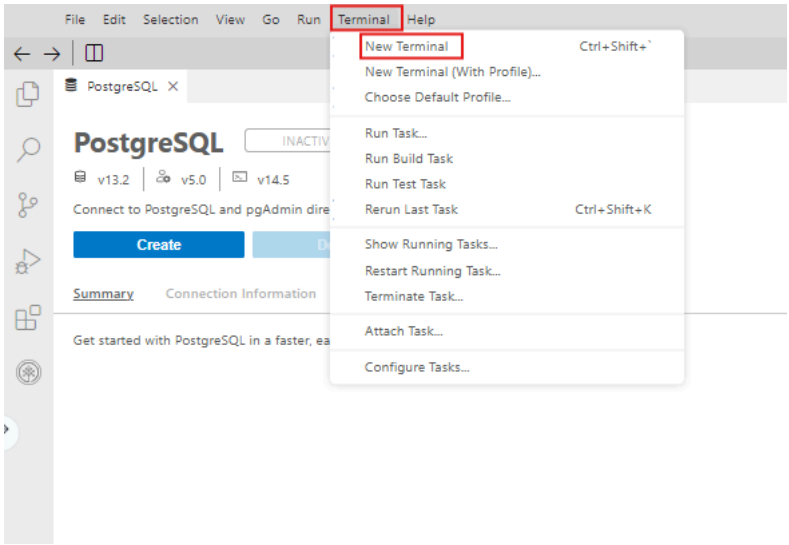
1. Click on the Skills Network extension button on the left side of the window.
2. Open the “DATABASES” drop down menu and click on “PostgreSQL”.
3. Click on the “Create” button. PostgreSQL may take a few moments to start.



Exercise 1: Restore a Full Database from a Backup

First, we will need to download the database.

1. Open a new terminal by clicking on the “New Terminal” button near the bottom of the interface.

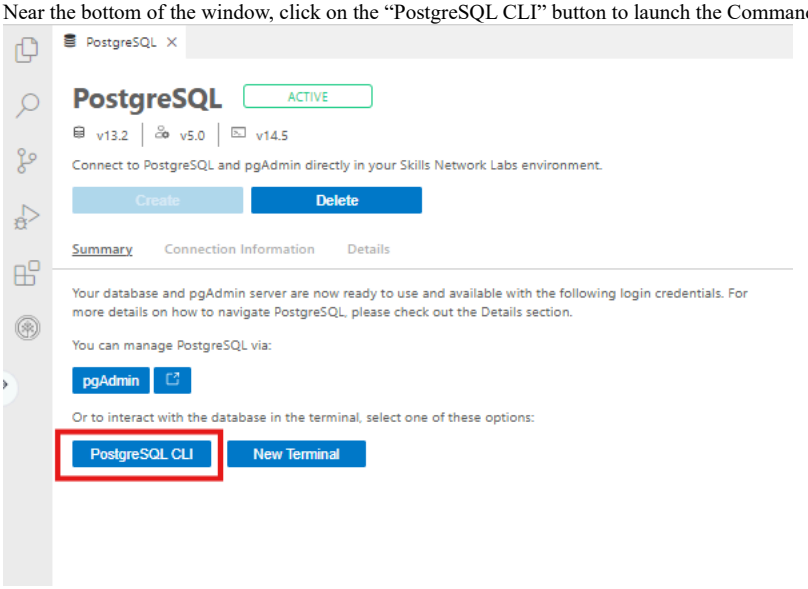


2. Run the following command in the terminal.

```
wget https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/example-guided-project/flights_RUSSIA_small.sql
```

The file which you downloaded is a full database backup of a month of flight data in Russia. Now, you can perform a full restoration of the dataset by first opening the PostgreSQL CLI.

3. Near the bottom of the window, click on the “PostgreSQL CLI” button to launch the Command Line Interface.



4. In the PostgreSQL CLI, type in the command `\i <file_name>`. In your case, the filename will be the name of the file you downloaded, `flights_RUSSIA_small.sql`. This will restore the data into a new database called `demo`.

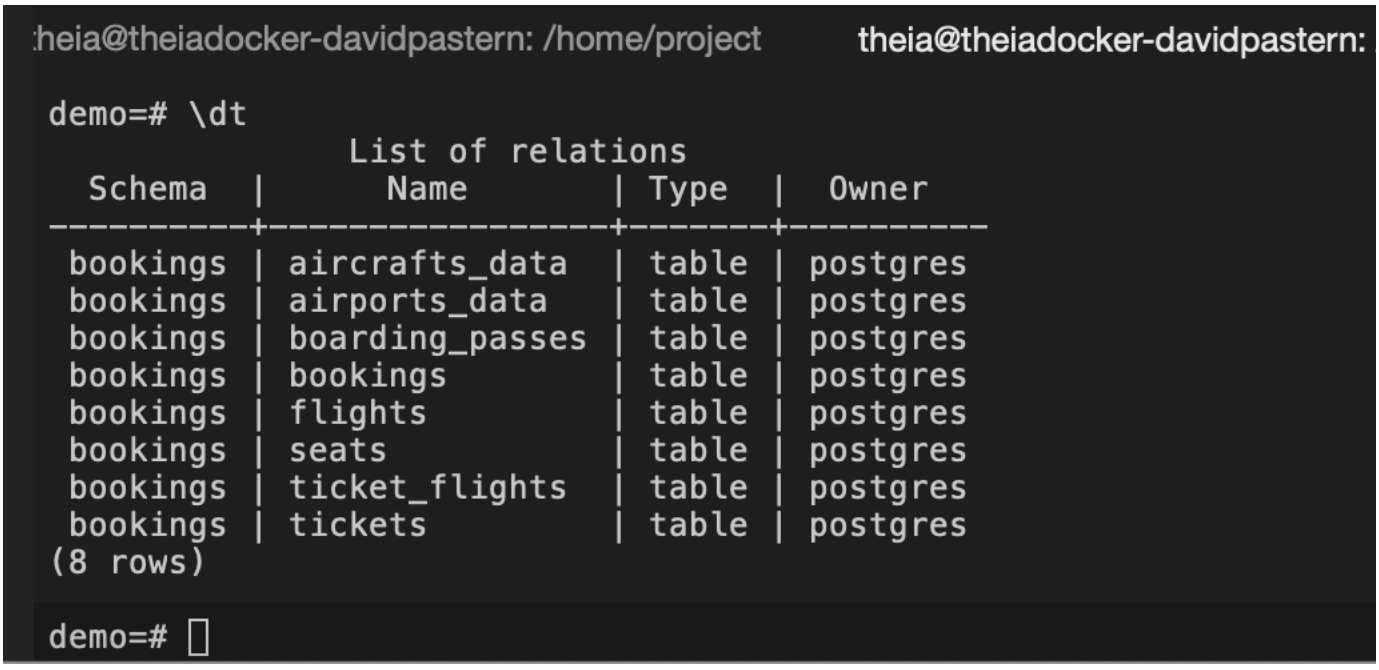
```
\i flights_RUSSIA_small.sql
```

The restorations may take a few moments to complete.

5. After the restoration completes, one way you can check that the database has been restored is with the following command, which lists all the tables in the current database schema.

```
\dt
```

You should see the following output:



Exercise 2: Modify the Database and Perform a Full Backup

Task A: Modify the Database with the CLI

1. One of the tables in the database schema is `aircrafts_data`. You can take a look at the contents of that table by executing the following command in the PostgreSQL **CLI**:

```
SELECT * FROM aircrafts_data;
```

This will show you the aircraft models in the database, their code, and their range in kilometers.

```
demo=# SELECT * FROM aircrafts_data;
 aircraft_code |              model               | range
---------------+----------------------------------+-------
 773           | {"en": "Boeing 777-300"}         | 11100
 763           | {"en": "Boeing 767-300"}         |  7900
 SU9           | {"en": "Sukhoi Superjet-100"}    |  3000
 320           | {"en": "Airbus A320-200"}        |  5700
 321           | {"en": "Airbus A321-200"}        |  5600
 319           | {"en": "Airbus A319-100"}        |  6700
 733           | {"en": "Boeing 737-300"}         |  4200
 CN1           | {"en": "Cessna 208 Caravan"}     |  1200
 CR2           | {"en": "Bombardier CRJ-200"}     |  2700
(9 rows)
```

2. Suppose a new model of aircraft is being added to the fleet, and you, as the database administrator, are responsible for updating the database to reflect this addition. The aircraft they wish to add is the Airbus A380, which has a range of 15,700 km and aircraft code "380". You can do this by executing the following command in the PostgreSQL CLI:

```
INSERT INTO aircrafts_data(aircraft_code, model, range) VALUES (380, '{"en": "Airbus A380-800"}', 15700);
```

3. To confirm that the information was entered into the database correctly, you can read out the `aircrafts_data` table again using:

```
SELECT * FROM aircrafts_data;
```

The output will look like this:

```
---------------+----------------------------------+-------
 773           | {"en": "Boeing 777-300"}         | 11100
 763           | {"en": "Boeing 767-300"}         |  7900
 SU9           | {"en": "Sukhoi Superjet-100"}    |  3000
 320           | {"en": "Airbus A320-200"}        |  5700
 321           | {"en": "Airbus A321-200"}        |  5600
 319           | {"en": "Airbus A319-100"}        |  6700
 733           | {"en": "Boeing 737-300"}         |  4200
 CN1           | {"en": "Cessna 208 Caravan"}     |  1200
 CR2           | {"en": "Bombardier CRJ-200"}     |  2700
 380           | {"en": "Airbus A380-800"}        | 15700
(10 rows)

demo=#
```

As you can see, there is a new entry in the table corresponding to the new aircraft added to the fleet.

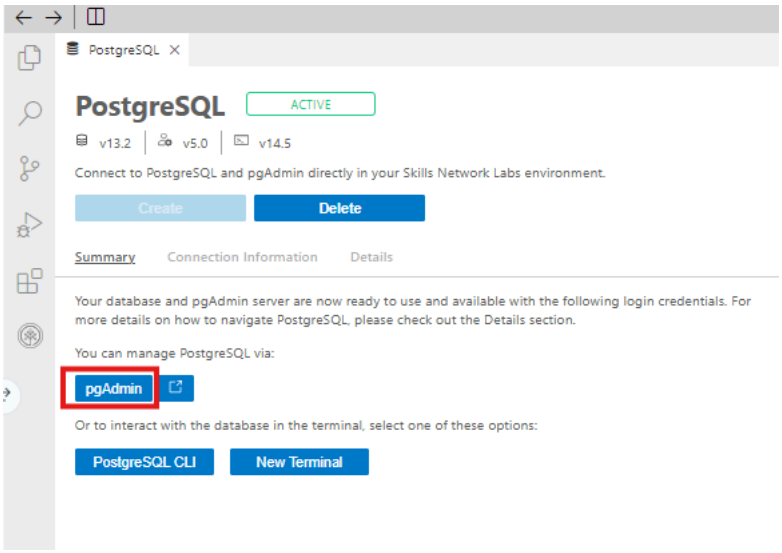
Task B: Backup your Database using pgAdmin

Now that you modified the database (minor modification for demonstration - in reality there would likely be far more additions) it is good practice to backup your database in case of accidental deletion.

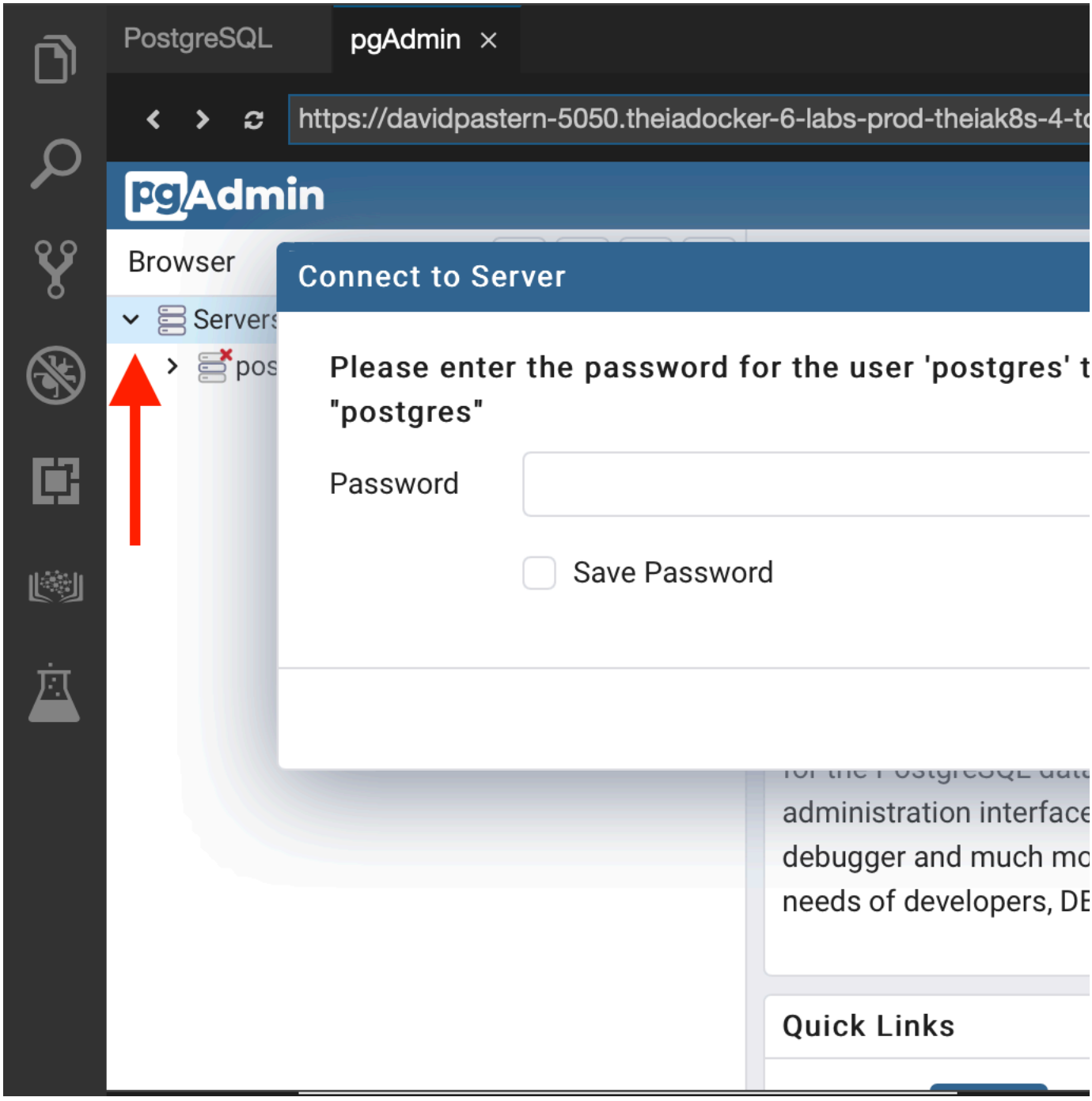
1. To back up the `demo` database, first exit the PostgreSQL CLI by either entering:

```
\q
```

2. Next, open the pgAdmin Graphical User Interface by clicking the "pgAdmin" button in the Cloud IDE interface.

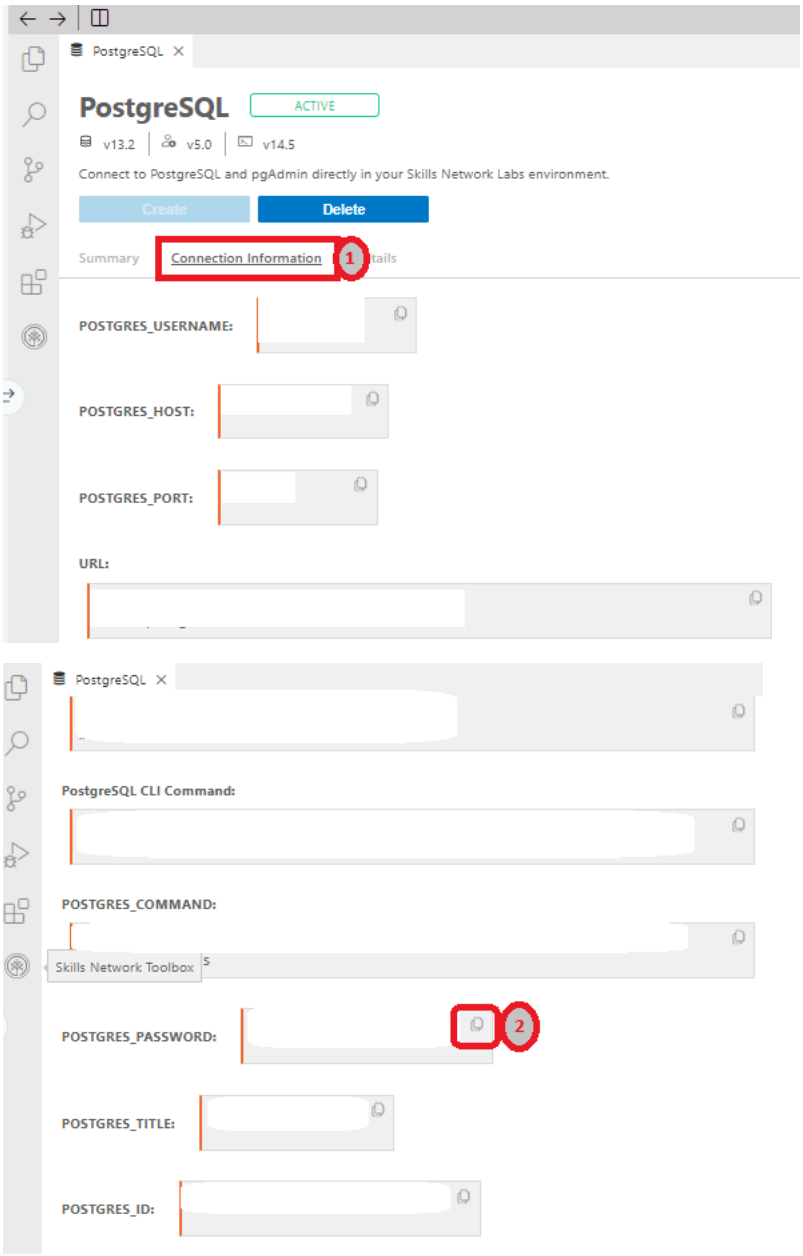


3. Once the pgAdmin GUI opens, click on the `Servers` tab on the left side of the page. You will be prompted to enter a password.

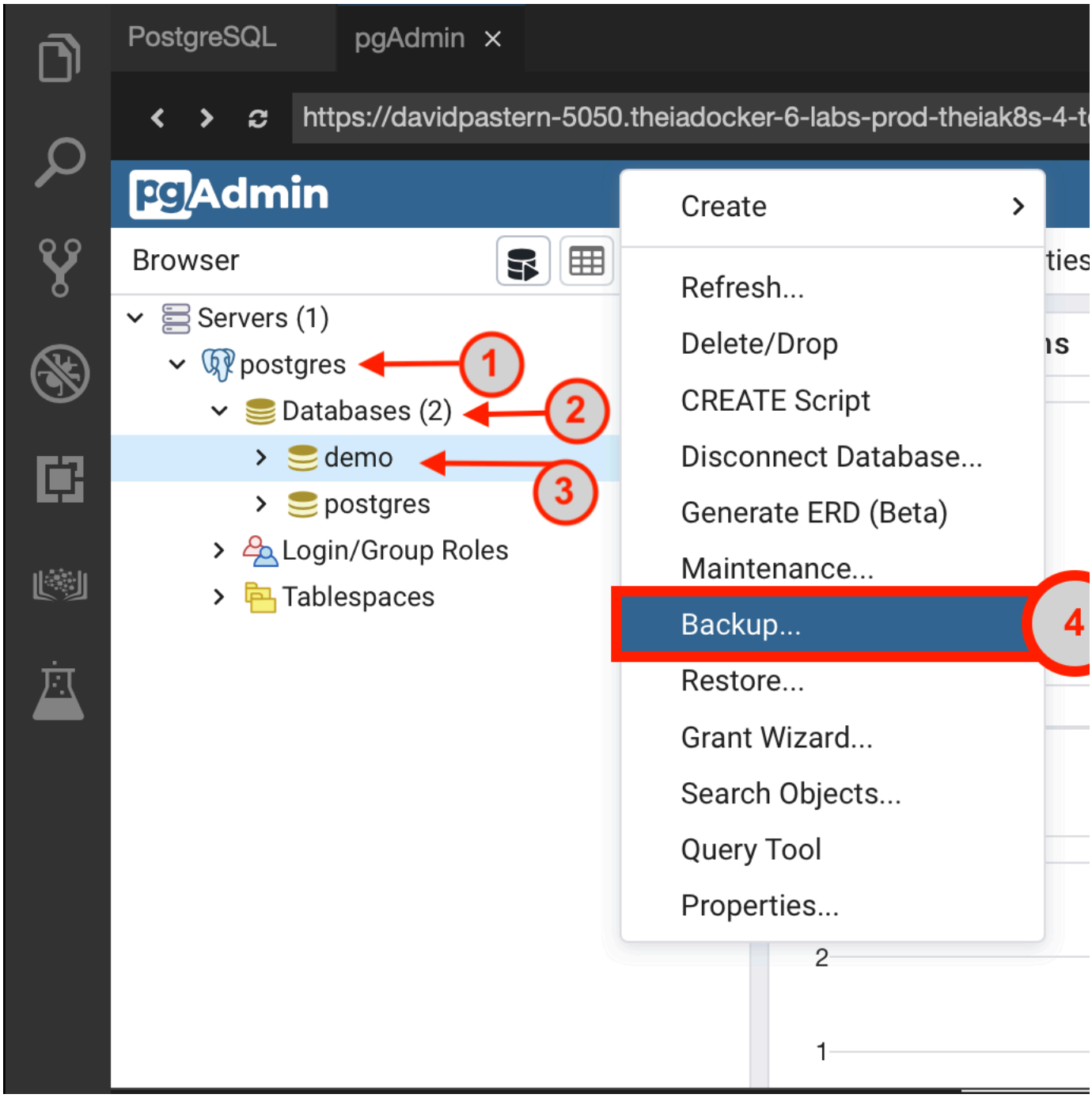


4. To retrieve your password, click on the “PostgreSQL” tab near the top of the interface.

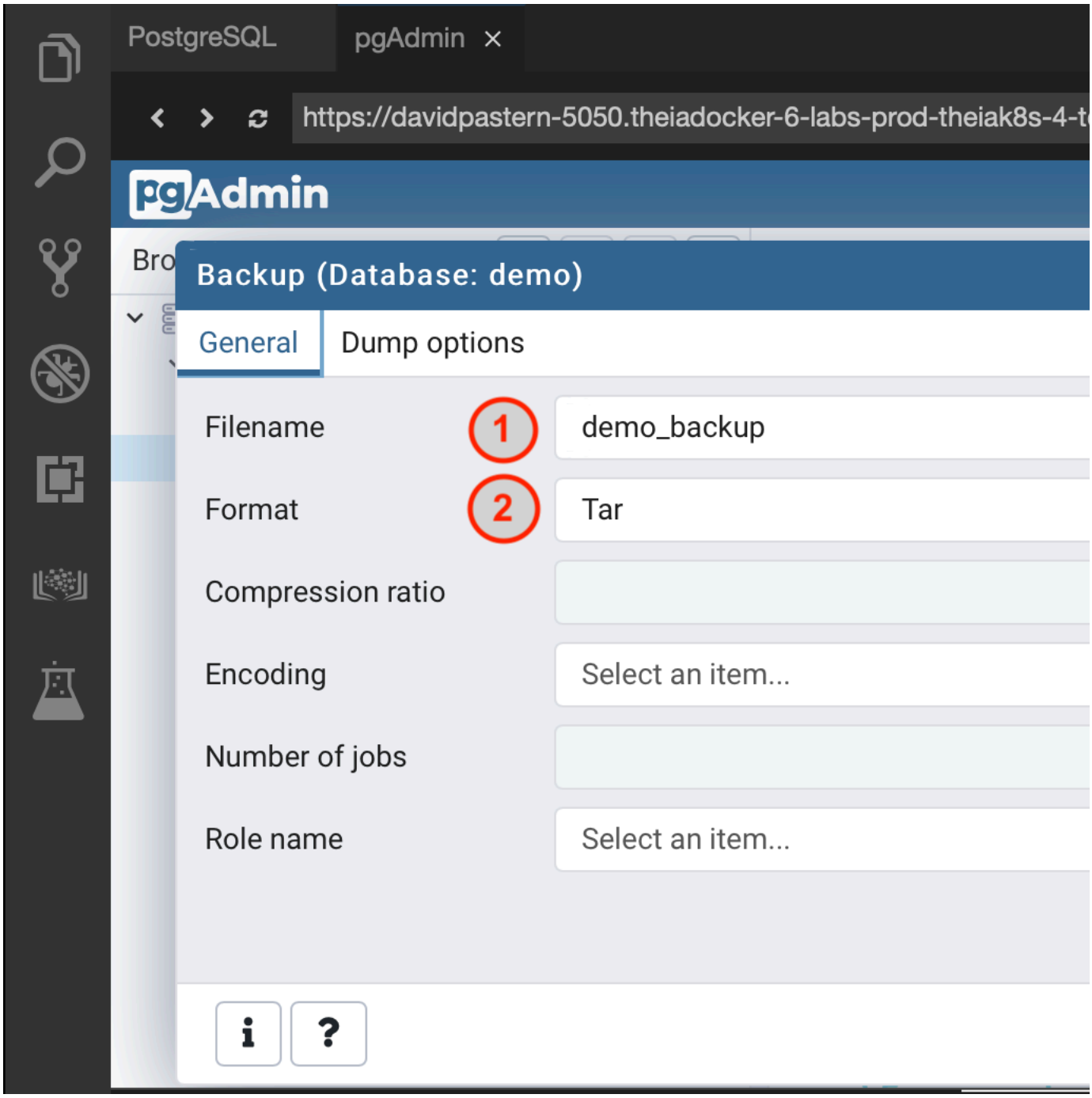
5. Click on the Copy icon to the left of your password to copy the session password onto your clipboard.



6. Navigate back to the “pgAdmin” tab and paste in your password, then click `OK`.
7. Click on `Postgres > Databases`.
8. Right click on `demo` and click the `Backup` button.



9. Enter a name for the backup (For example, “demo_backup”), set the `Format` to `Tar`, then click the “Backup” button.

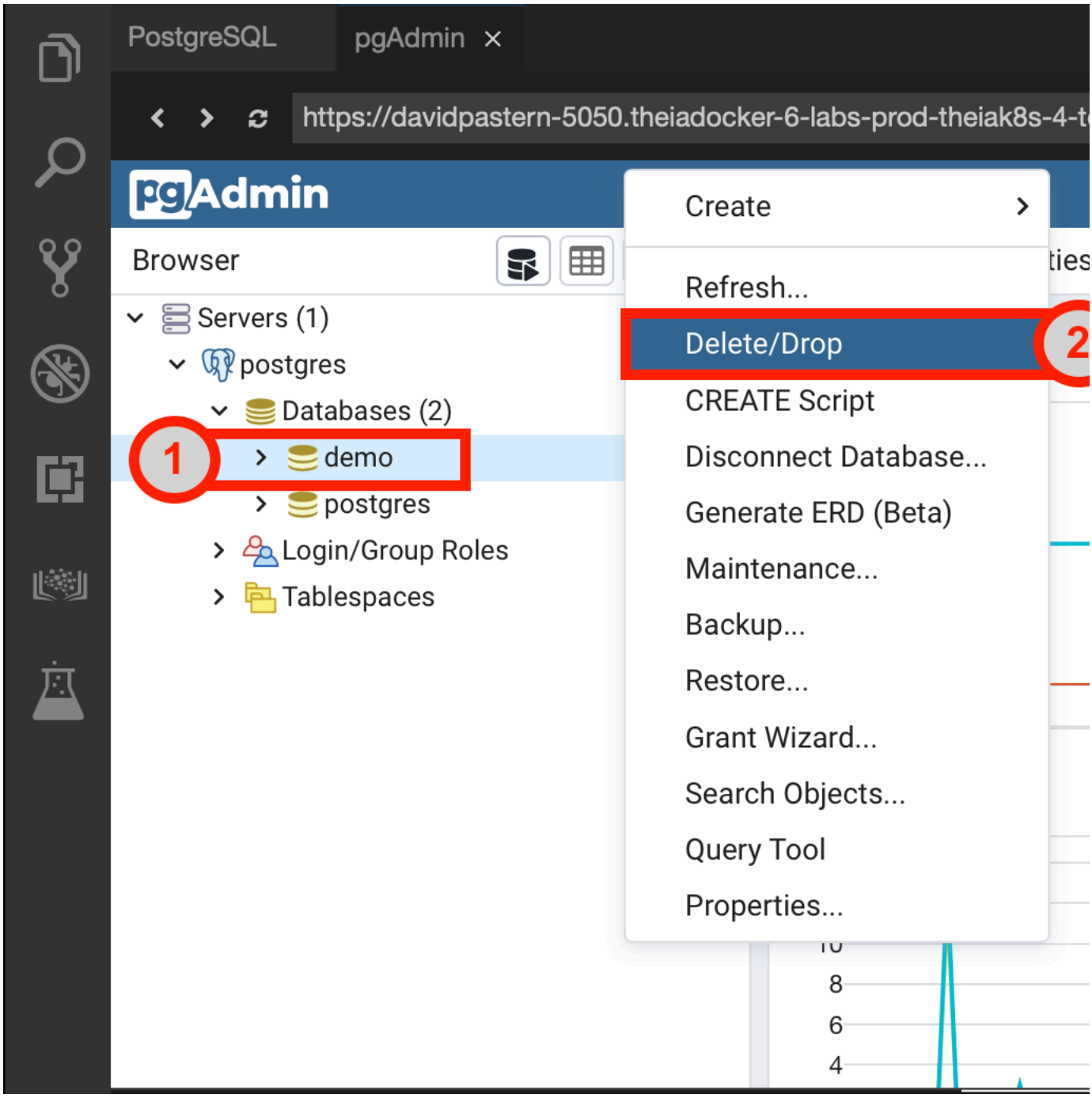


Exercise 3: Restore a Full Backup after Accidental Deletion

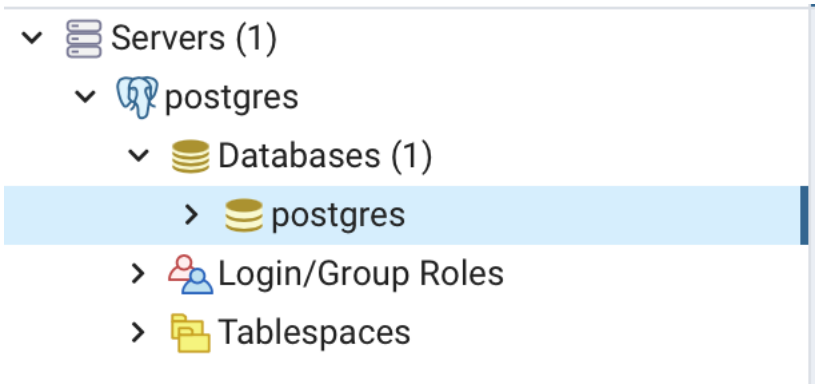
In this exercise, suppose you find yourself in a situation where you accidentally dropped the entire database. Fortunately, you made a full backup of the database in the previous exercise, which you will use to restore the database.

Task A: “Accidentally” Delete the Database

1. In the pgAdmin GUI, right click on the `demo` database and then click the “Delete/Drop” button.



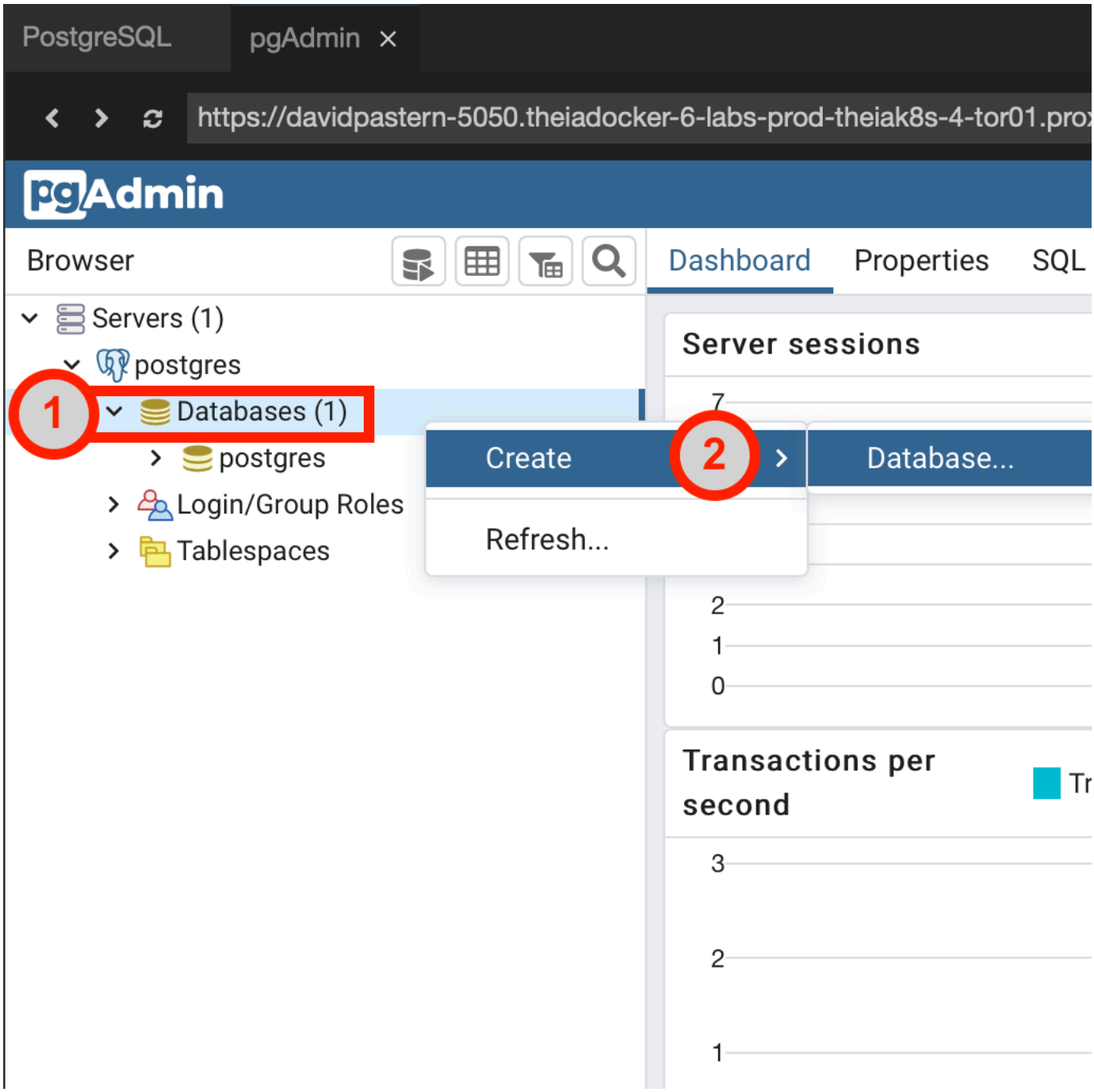
2. When prompted, click “Yes” to confirm the deletion of the database.
3. You will see that the `demo` database is no longer listed, which verifies that you have dropped it.



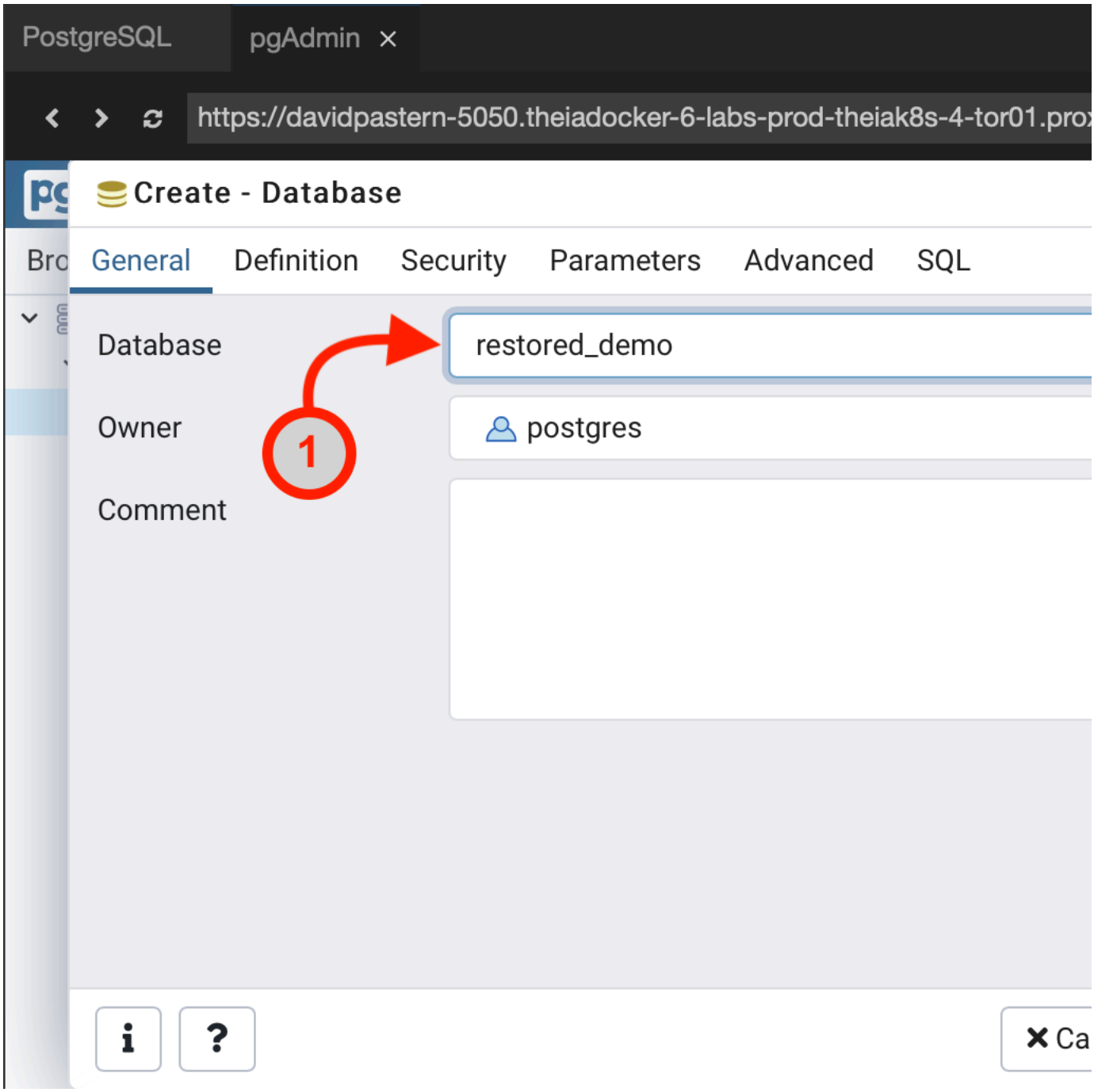
Task B: Restore the Database using the Full Backup

You will now use the full backup you created in Exercise 2 to restore the database which was deleted.

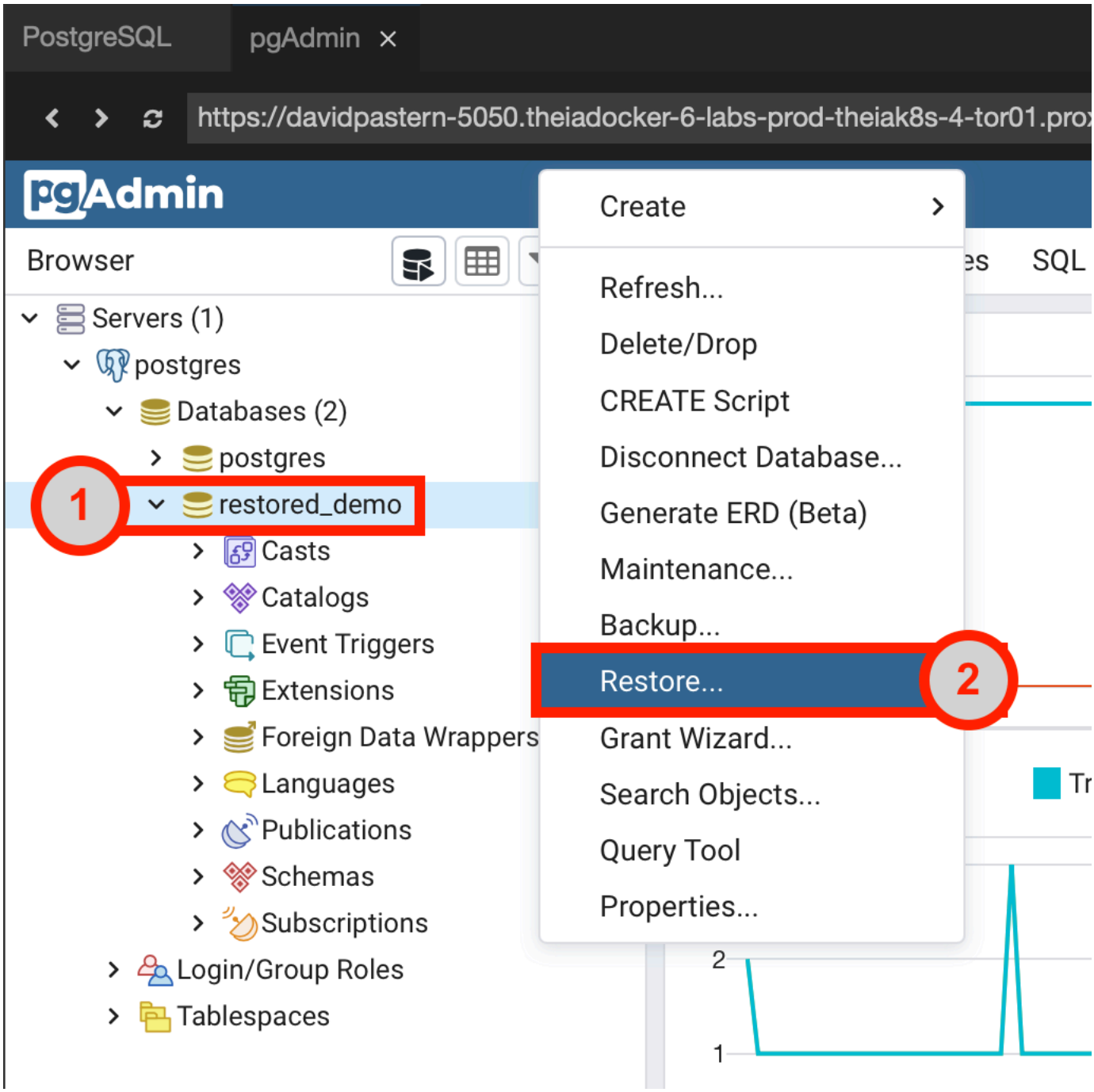
1. First, you will need an empty database in which to restore the `demo` database. Create a new database in pgAdmin by right clicking “Databases” then clicking “Create” > “Database…”.



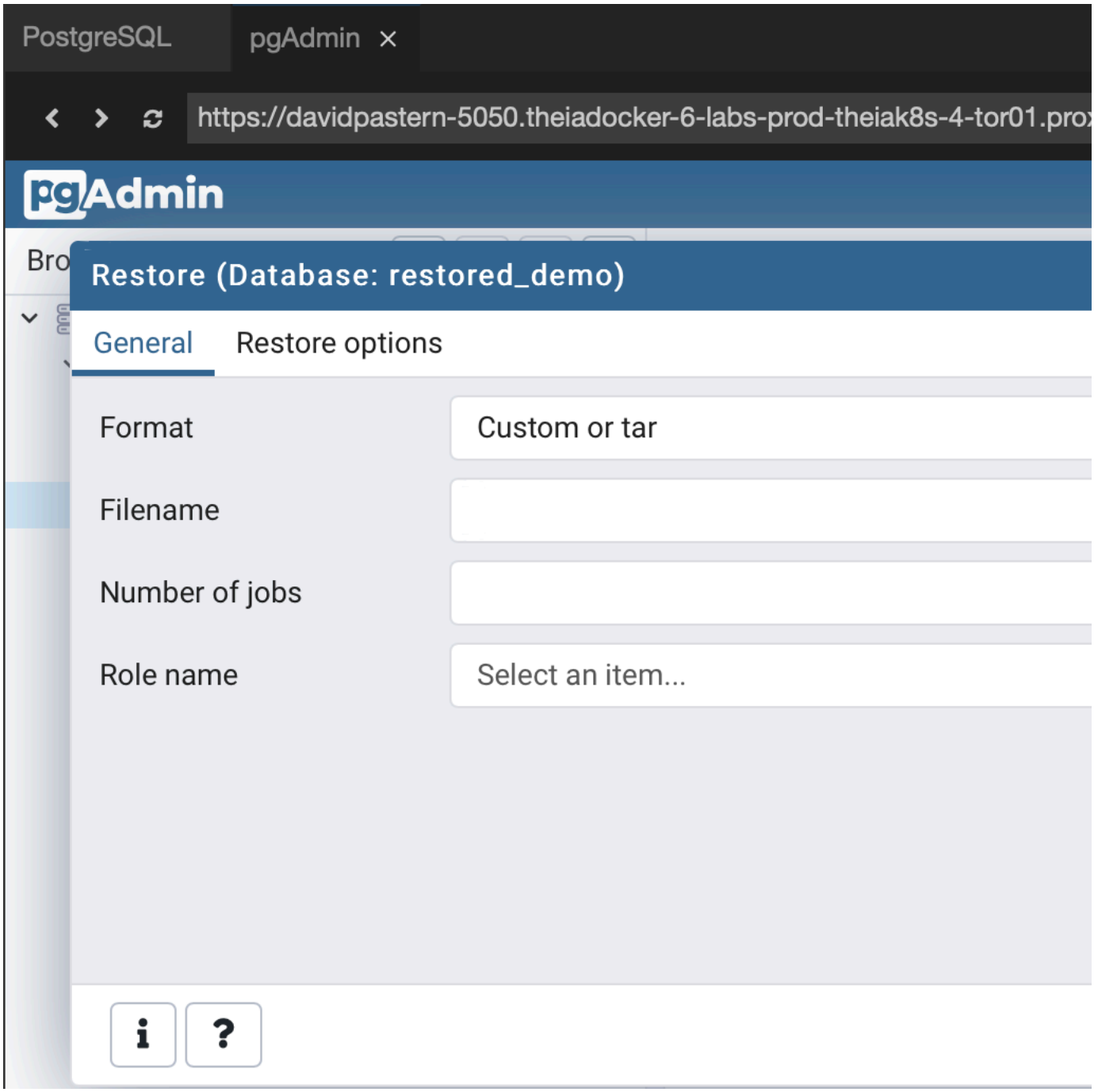
2. Name the database into which you will restore the original `demo` database (For example, `restored_demo`), then click the “Save” button on the bottom right.



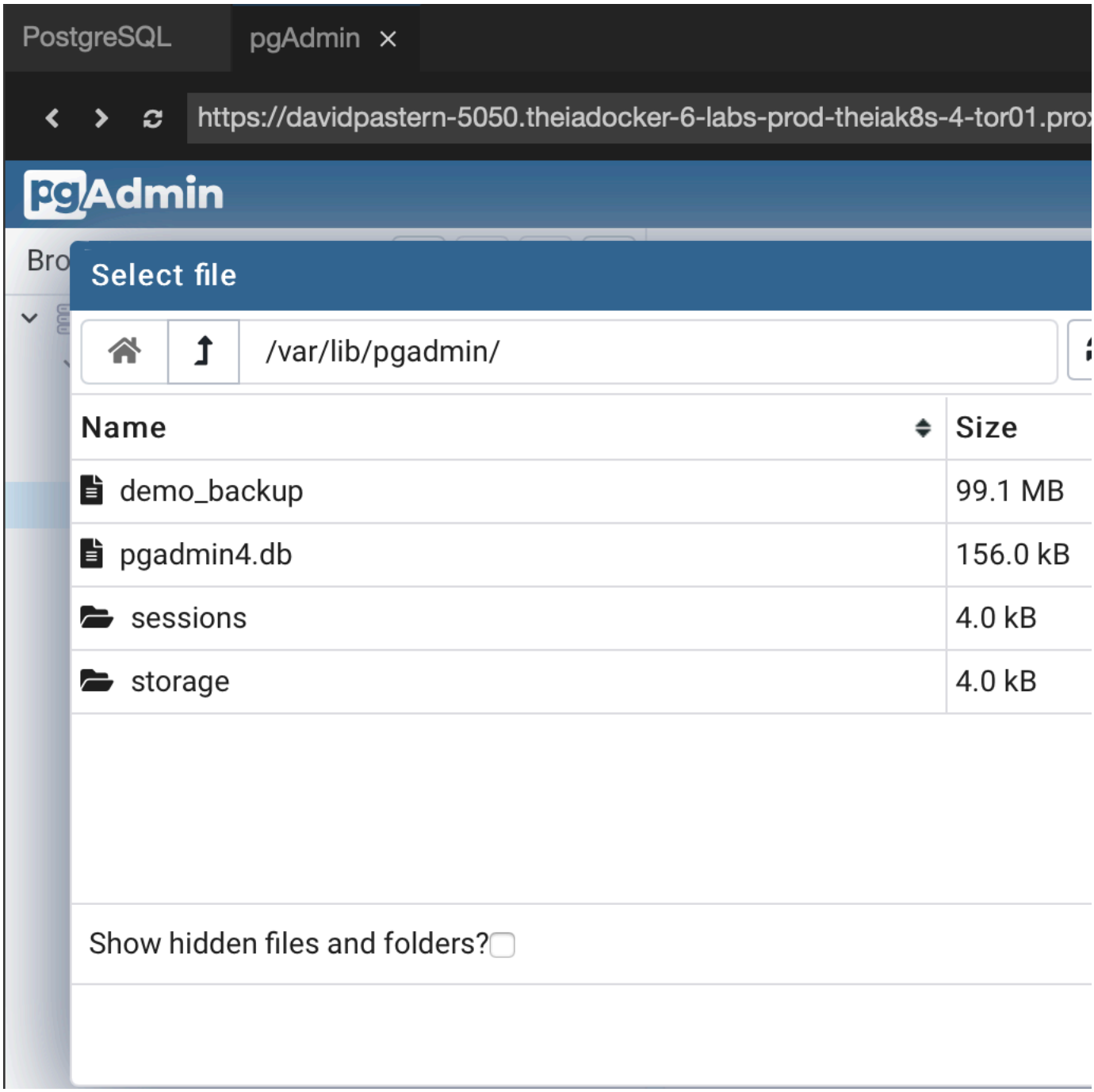
3. Next, to restore the backup you created in Task A into this new database, right click on the database you created (For example, `restored_demo`). Then click on the “Restore…” button.



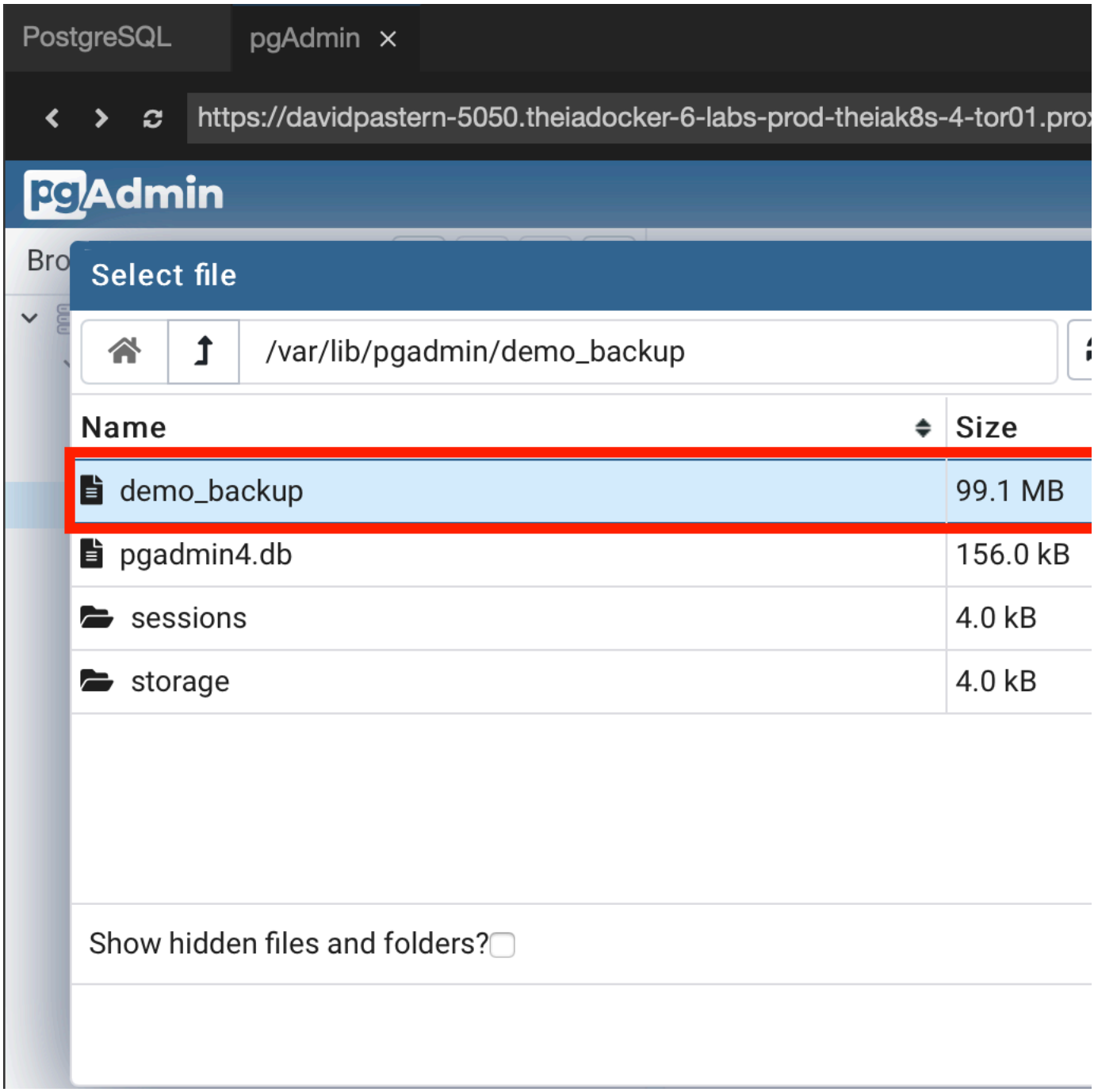
4. Click on the button containing three dots by the Filename box.



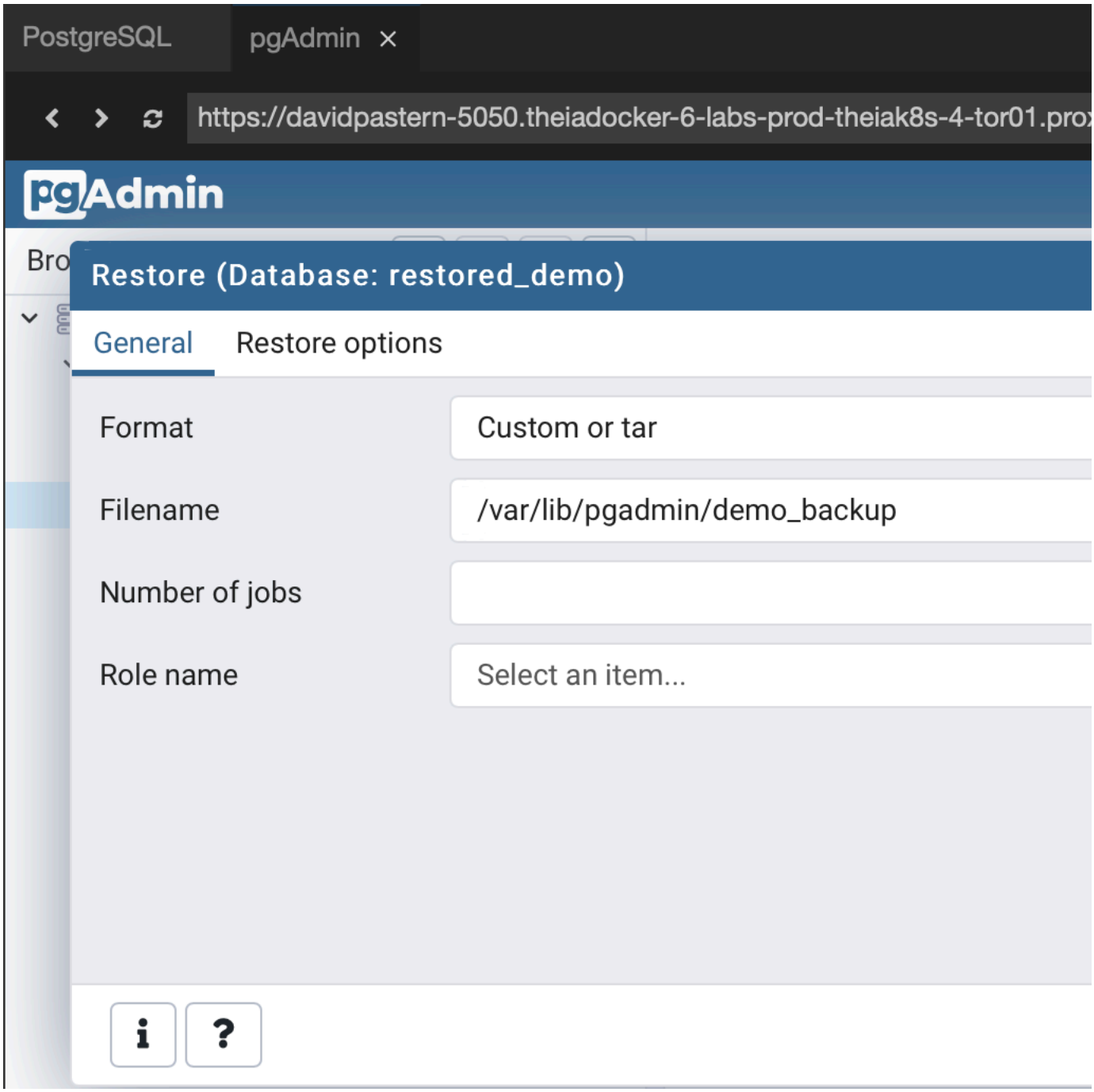
5. Near the bottom left of the window, open the “Format” drop down window and select “All files”.



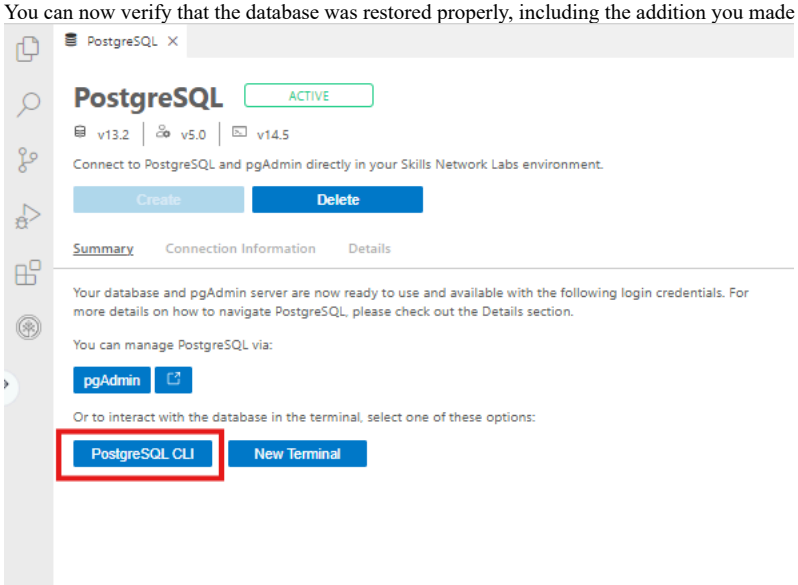
6. Select the backup you created in Task A (For example, `demo_backup`), then click the “Select” button near the bottom right of the window.



7. Then click on the “Restore” button at the bottom right of the window to restore the database.



8. You can now verify that the database was restored properly, including the addition you made to the `aircrafts_data` table. Open up the PostgreSQL CLI:



9. In the CLI, enter the command:

```
\connect restored_demo
```

10. To set the proper search path for your database, enter the following into the CLI:

```
SELECT pg_catalog.set_config('search_path', 'bookings', false);
```

11. To see the restored tables in the database, enter:

```
\dt
```

You will see the same tables as in the original `demo` database.

```
                List of relations
  Schema  |      Name       | Type  |  Owner
----------+-----------------+-------+----------
 bookings | aircrafts_data  | table | postgres
 bookings | airports_data   | table | postgres
 bookings | boarding_passes | table | postgres
 bookings | bookings        | table | postgres
 bookings | flights         | table | postgres
 bookings | seats           | table | postgres
 bookings | ticket_flights  | table | postgres
 bookings | tickets         | table | postgres
(8 rows)
```

12. Recall that you added a new aircraft model (Airbus A380) to the original database. Verify that this addition was successfully backed up and restored by entering the following command:

```
SELECT * FROM aircrafts_data;
```

```
restored_demo=# SELECT * FROM aircrafts_data;
 aircraft_code |              model               | range
---------------+----------------------------------+-------
 773           | {"en": "Boeing 777-300"}         | 11100
 763           | {"en": "Boeing 767-300"}         |  7900
 SU9           | {"en": "Sukhoi Superjet-100"}    |  3000
 320           | {"en": "Airbus A320-200"}        |  5700
 321           | {"en": "Airbus A321-200"}        |  5600
 319           | {"en": "Airbus A319-100"}        |  6700
 733           | {"en": "Boeing 737-300"}         |  4200
 CN1           | {"en": "Cessna 208 Caravan"}     |  1200
 CR2           | {"en": "Bombardier CRJ-200"}     |  2700
 380           | {"en": "Airbus A380-800"}        | 15700
(10 rows)
```

Notice that the Airbus A380 entry is there! Once again, you can enter `\q` to exit this view.

Practice Exercise

Scenario: *Suppose a passenger, Saniya Koreleva, books a flight with your airline company. Unfortunately, due to human error her first name was initially entered incorrectly into the database and you wish to correct it.*

In this practice exercise, you will apply the techniques you learned in this lab to modify the database to correct the spelling of the passenger's name. Then, you will practice performing a full backup of the database.

1. First, in the `restored_demo` database, take a look at the original entry for the passenger in the "tickets" table using the PostgreSQL Command Line Interface. Use the `booking_ref` parameter to query the database. The booking reference for this passenger is 0002D8.

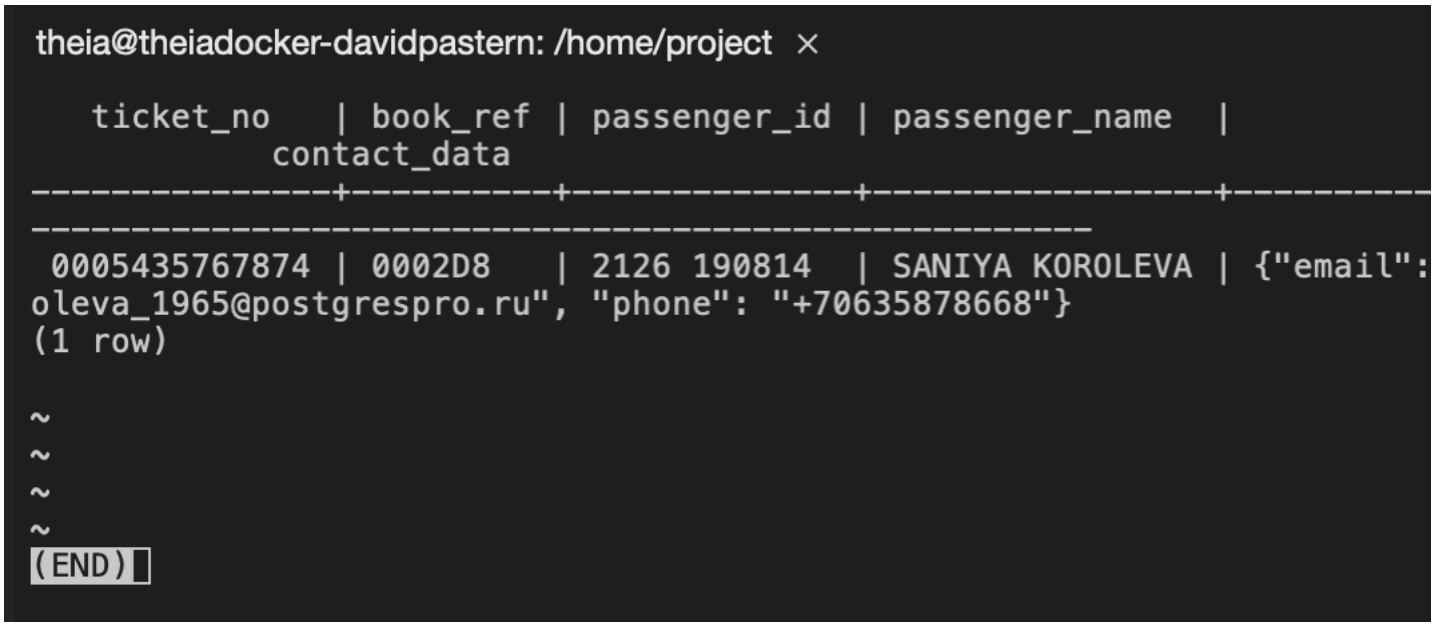
▼ Hint (Click Here)
To query a specific table in the database using some condition, enter a command in the CLI with the following format:

```
SELECT * FROM <table name> WHERE <condition>;
```

▼ Solution (Click Here)

```
SELECT * FROM tickets WHERE book_ref = '0002D8';
```

You should see see the following output showing the ticket associated with the booking reference 0002D8 under the name Saniya Koreleva. When you are done, type `\q` in the CLI to exit the view.



2. Next, suppose the passenger's first name is actually spelled "Sanya" instead of "Saniya". Modify the entry for this passenger to correct the spelling by changing the `passenger_name` to "SANYA KORELEVA".

▼ Hint (Click Here)
To change an entry in a table, enter a command in the CLI with the following format:

```
UPDATE <table name> SET <column A> = <condition A> WHERE <column B> = <condition B>;
```

▼ Solution (Click Here)

```
UPDATE tickets SET passenger_name = 'SANYA KORELEVA' WHERE book_ref = '0002D8';
```

You can then verify that your change was successful by entering the same command as you did in Step 1.

3. Now suppose that several other changes were done on the database, such as more bookings were created, flights scheduled, etc., and you wish to perform a full backup of the database. To complete this exercise, perform a full backup of the `restored_demo` database. Name the back up `restored_demo_backup.sql`.

▼ Hint (Click Here)
Recall that you performed a full backup earlier in this lab. Refer to Exercise 2 for a refresher.
▼ Solution (Click Here)
You can use pgAdmin in the same way as you did in Exercise 2 to perform a full backup.

An alternative method is to perform the backup using the terminal.

In the Cloud IDE terminal, enter the following command to create a backup of the `restored_demo` called `restored_demo_backup.sql` database:

```
pg_dump --username=postgres --host=localhost restored_demo > restored_demo_backup.sql
```

You can then verify that your change was successful by entering the same command as you did in Step 1.

Conclusion

Congratulations! You have successfully completed the lab and have gained some familiarity on how to perform a full backup and restoration of a database using PostgreSQL.

To summarize, recall that you covered the following objectives:

- Restore a full database from a backup
- Update a database and perform a full backup
- Drop a database and then restore it

Author

David Pasternak

Other Contributors

Sandip Saha Joy, Rav Ahuja