I'm pretty sure many of you have successfully completed the Python on ACG course with lots of practice and lots of love. I'm confident it wouldn't hurt to re-quote some important parts of the Python course from the perspective of a student like me. I guess there are crucial places for cloud engineering, and it would not be a waste of time to repeat it and it would be useful to focus on it again.

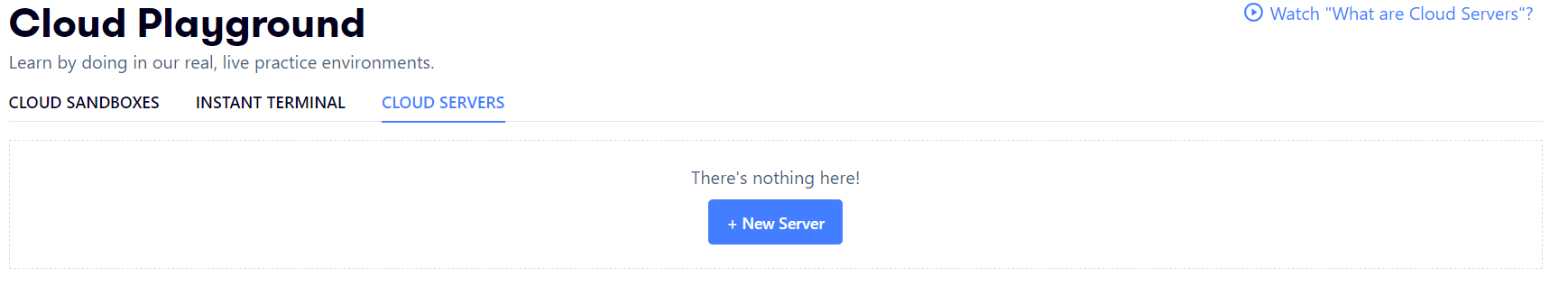
Because Keith's videos are from 5 years ago, some versions are not compatible with each other. Although it updated the db\_setup.sh file 3 years ago, I have seen in my experience that there is a conflict between PostgreSQL versions and Linux servers. If you want to see the image that Keith SQL is processing on your own computer now, you could fulfill this desire by using the commands I have given below.

First, let's start our Azure Sandbox from the Cloud Playground page[[link](https://learn.acloud.guru/cloud-playground/cloud-sandboxes)].

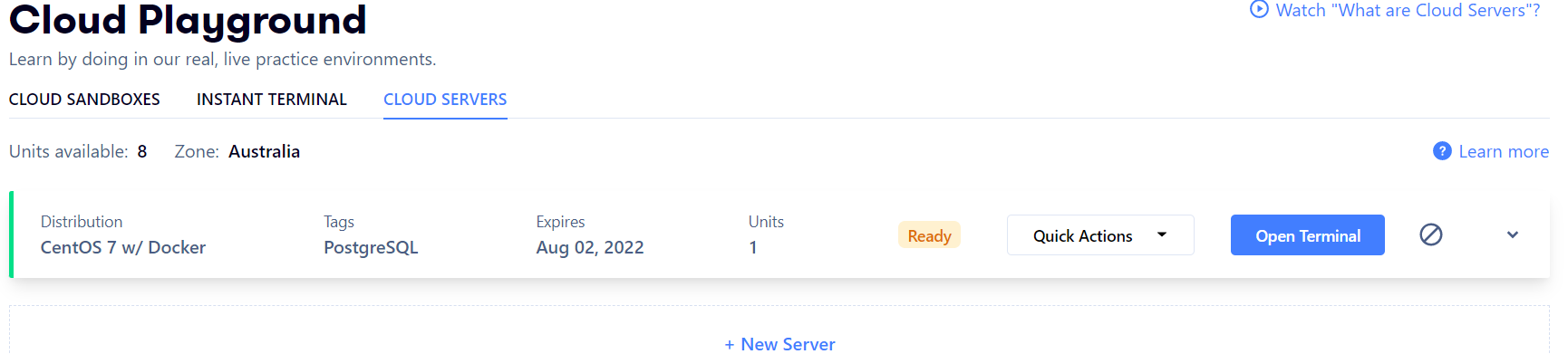
Graphical user interface, text, application, email

Description automatically generated

You could then use AWS and Google Cloud for practice.



After Open Sandbox, we first create a server for PostgreSQL. We choose Distribution CentOS 7 w/ Docker.



Open the terminal of the server we created and enter the following commands. And you will need to update your temp\_password. This password is for user sudo.

Text

Description automatically generated

After reaching the terminal page, we will install the database on this server. The installation process continues as follows:

**download database ->>**

*curl -o db\_setup.sh https://raw.githubusercontent.com/linuxacademy/content-python3-sysadmin/master/helpers/db\_setup.sh*

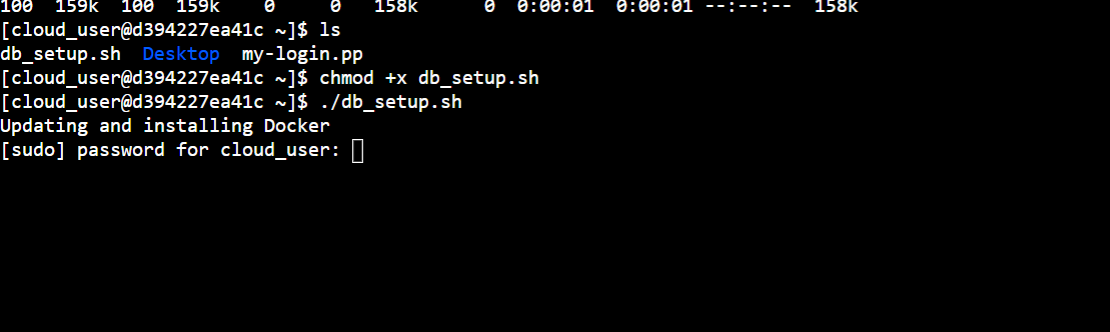
A computer screen capture

Description automatically generated with medium confidence

**then ->>**

*chmod +x db\_setup.sh*

*./db\_setup.sh*

**

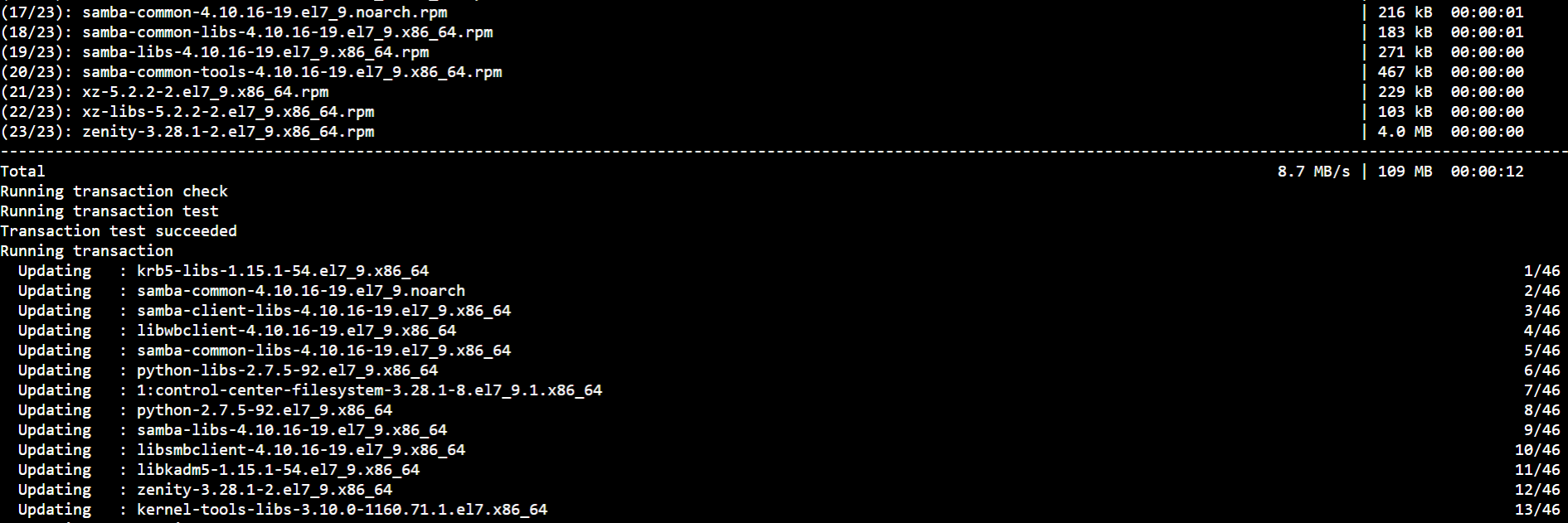
Entering your sudo password then the db\_setup.sh file will automatically make the necessary settings for us. Here is a good example of automation. As a cloud engineer, you will be able to create files in this way.

This may take some time.

Shape

Description automatically generated with low confidence

Still going on



We enter sudo password again.

A picture containing text

Description automatically generated

The rpm file is automatically installing.

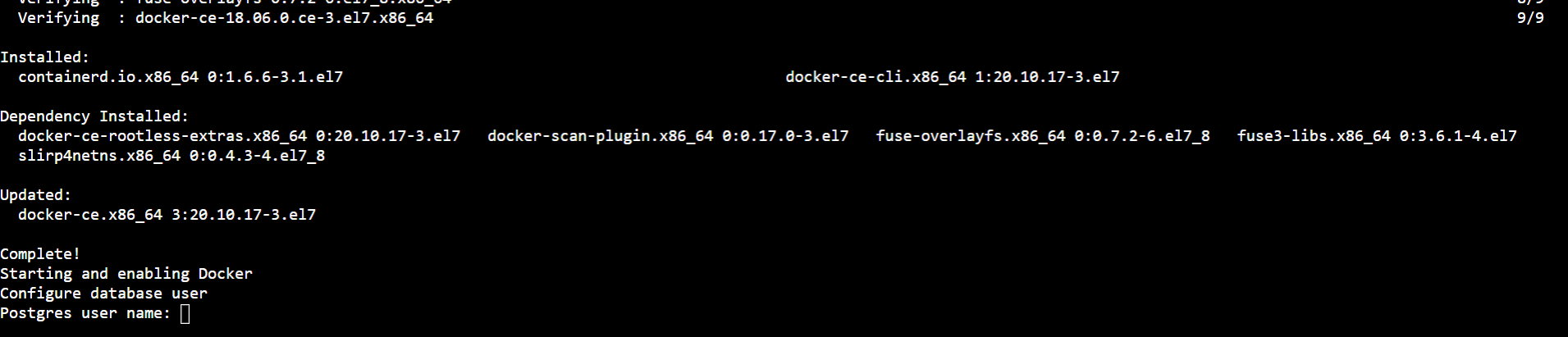
Text

Description automatically generated

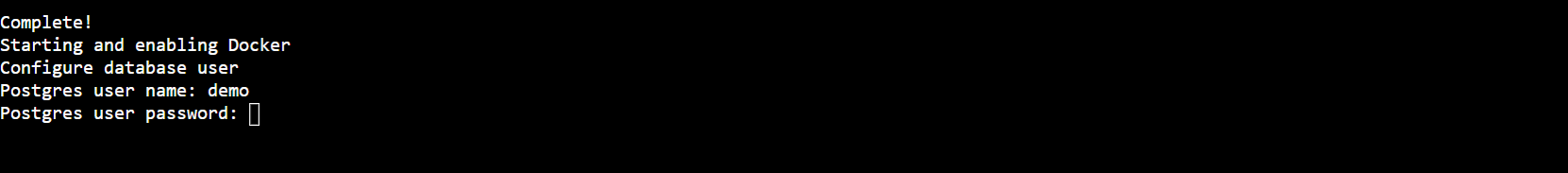
When you see this screen, you need to create a user and password for the database. The user and password we will use here will be used to access the database from the Workstation server we will create shortly. Making this username and password different from sudo's will avoid confusion.

We can make it as Keith did.

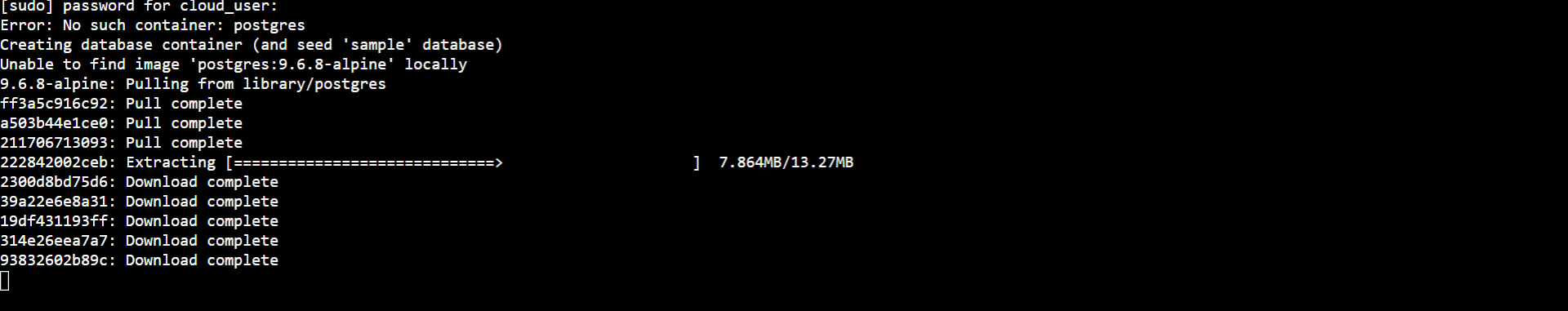
Postgres username: demo



Password: password



Automation starts again

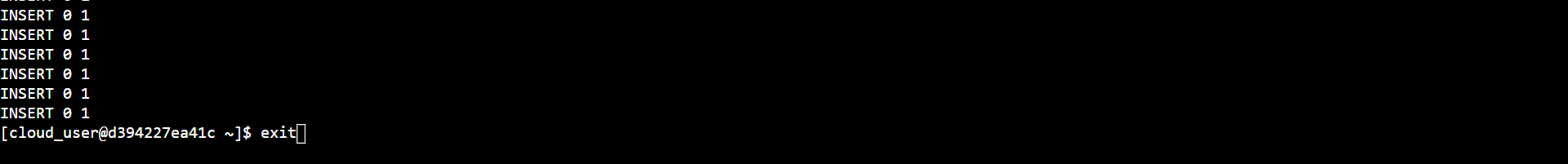


Finally, our sample database has been completed by installing it on the Linux server we have created via Azure Sandbox

Shape

Description automatically generated with medium confidence

**Then ->>** *exit* **from PostgreSQL server**

****

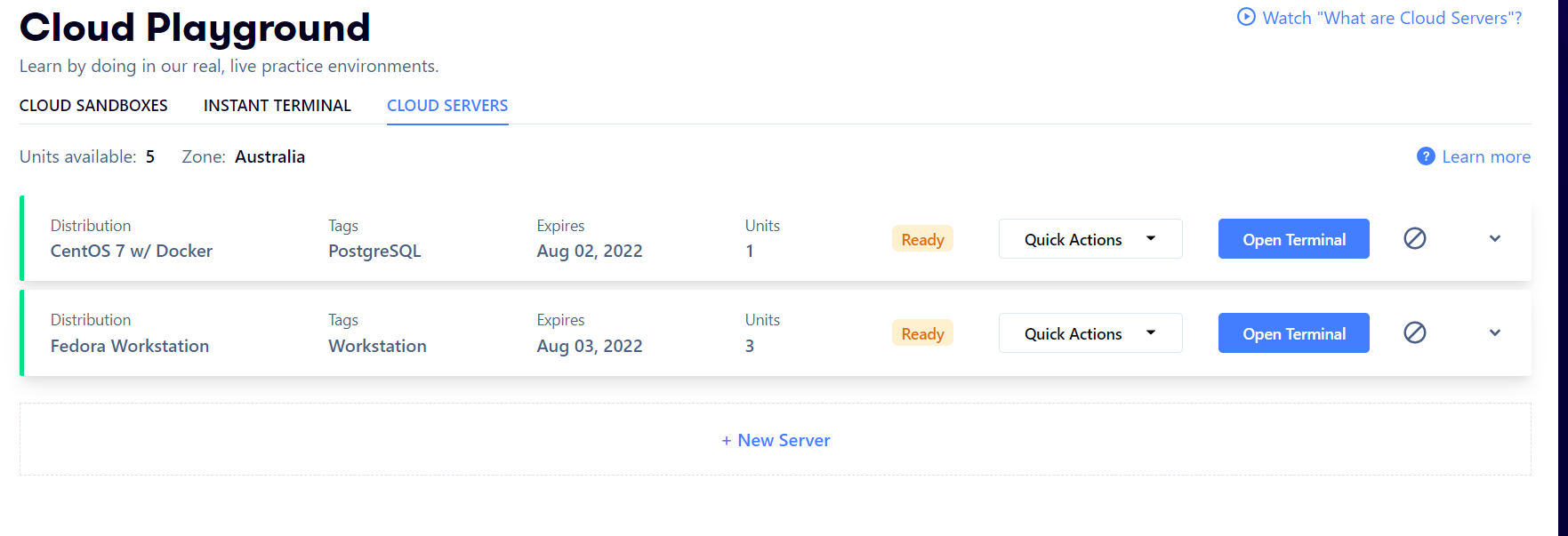
Now we could go to azure sandbox cloud server again and set up a server for workstation. I made the sizes randomly because of it is tutorial.

Graphical user interface, application, Teams

Description automatically generated

Here I chose fedora, another rpm distributor, because they made it a workstation functionally on the Azure Sandbox, and it installed PostgreSQL and plugins with a very high performance and without any problems.

Our Workstation server installation on Azure sandbox is completed



We need to update the password as in the server we set up for the database. We need to update password that every server we create it. I put the same sudo password, because it is tutorial .

Text

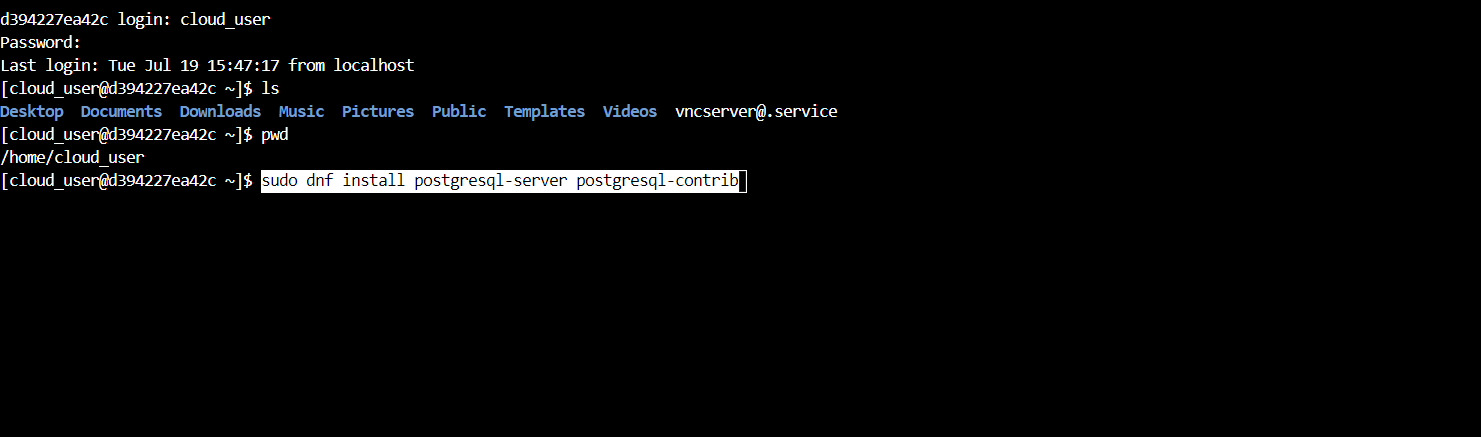
Description automatically generated

we need to install PostgreSQL on this server. We need to apply the following commands respectively.

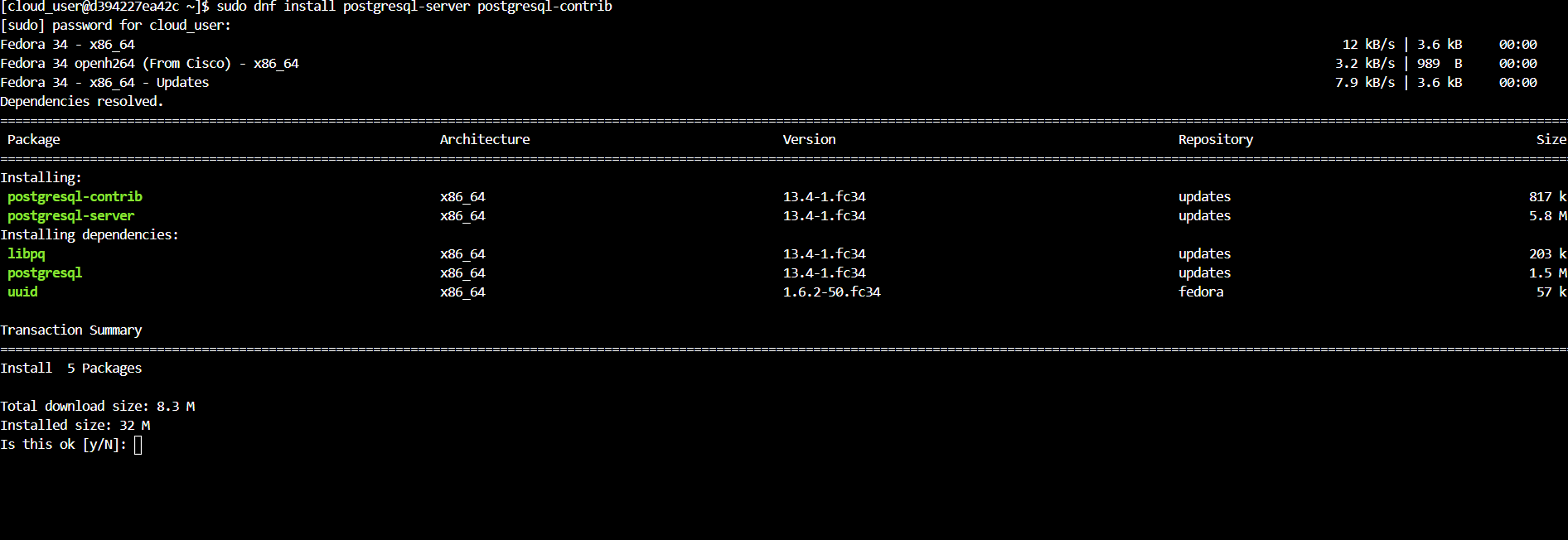
[[link](https://docs.fedoraproject.org/en-US/quick-docs/postgresql/)]

**install PostgreSQL on fedora ->>**

*sudo dnf install postgresql-server postgresql-contrib*

**

Tap in Y



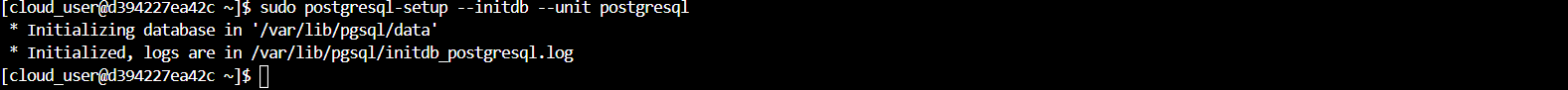
It is completed then we could switch to second command



*sudo systemctl enable postgresql*

**

*sudo postgresql-setup --initdb --unit postgresql*

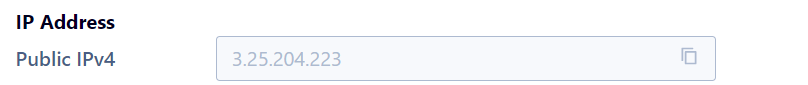
**

*sudo systemctl start postgresql*

**

From here, we can connect to our database server from the workstation. We will use the username "demo" and the password "password" that we specified while creating the database server.

connect to PostgreSQL server(database) from fedora34-workstaion server and play it using by SQL. Public IPv4(dynamic) will swap every time you start it, username, and password(static)



*psql postgres://demo:password@ 3.25.204.223:80/sample -c "SELECT \* FROM employees;"*

**

Here we can play on the database with simple or complex SQL commands. The exercises you will do with SQL will help you a lot in your python development. I did some practice with sql on the database for fun, it will be beneficial for you to do it too.

A picture containing text

Description automatically generated

*psql postgres://demo:password@3.25.204.223:80/sample -c "SELECT gender FROM employees WHERE gender = 'Female';"*

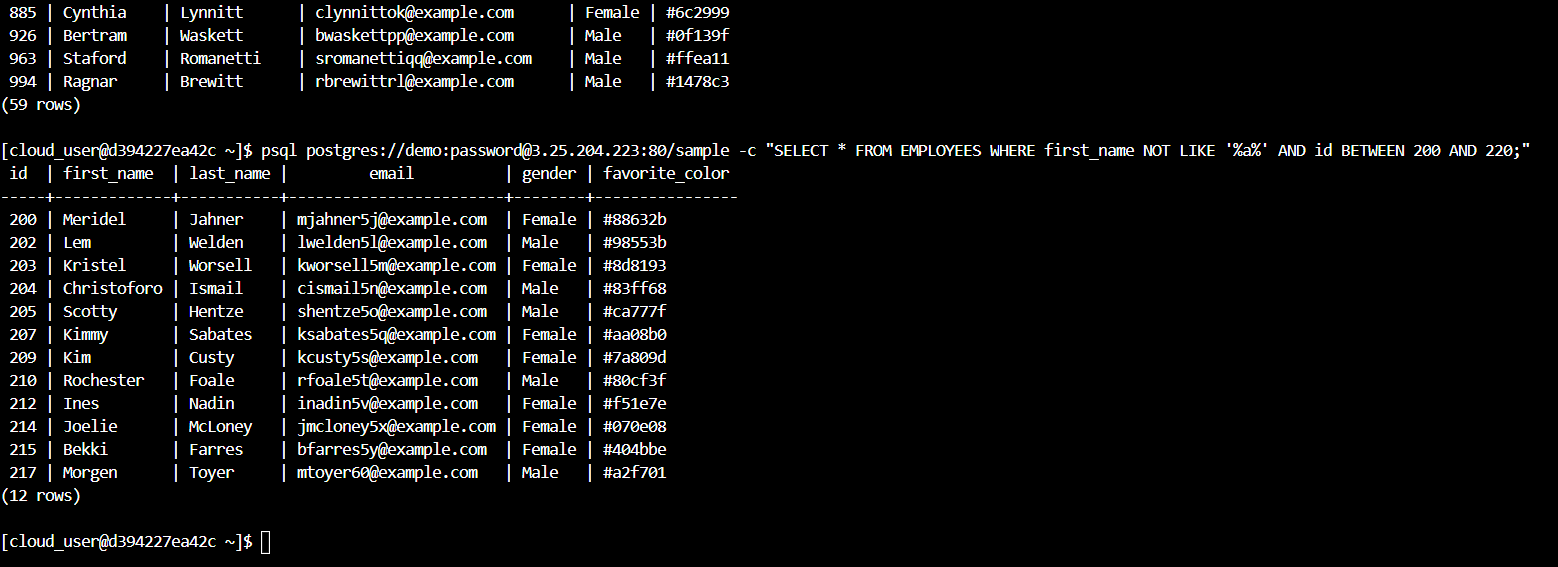
**

*psql postgres://demo:password@3.25.204.223:80/sample -c "SELECT \* FROM employees WHERE email LIKE '%tt%';"*

**

*psql postgres://demo:password@3.25.204.223:80/sample -c "SELECT \* FROM EMPLOYEES WHERE first\_name NOT LIKE '%a%' AND id BETWEEN 200 AND 220;"*

*\*sql-SQL*

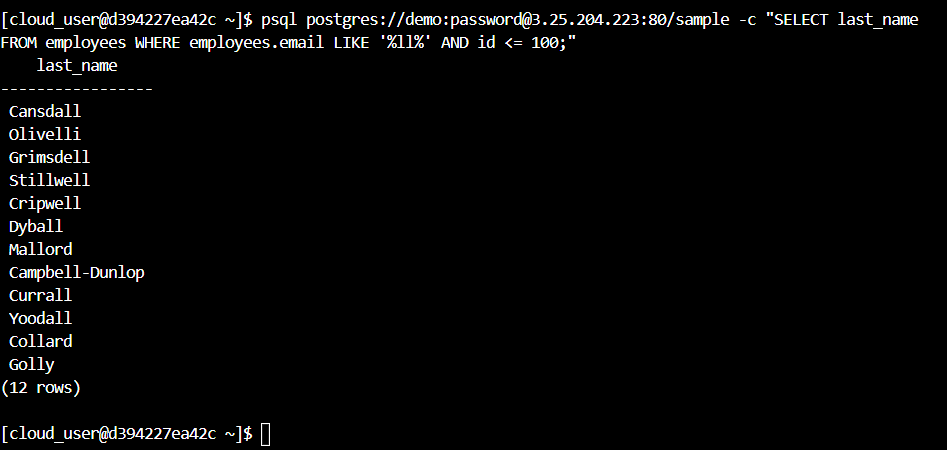
**

*psql postgres://demo:password@3.25.204.223:80/sample -c "SELECT \* FROM EMPLOYEES WHERE id>900 AND (last\_name LIKE '%tt%' OR last\_name LIKE '%ll%');"*

**

*psql postgres://demo:password@3.25.204.223:80/sample -c "SELECT last\_name*

*FROM employees WHERE employees.email LIKE '%ll%' AND id <= 100;"*

**

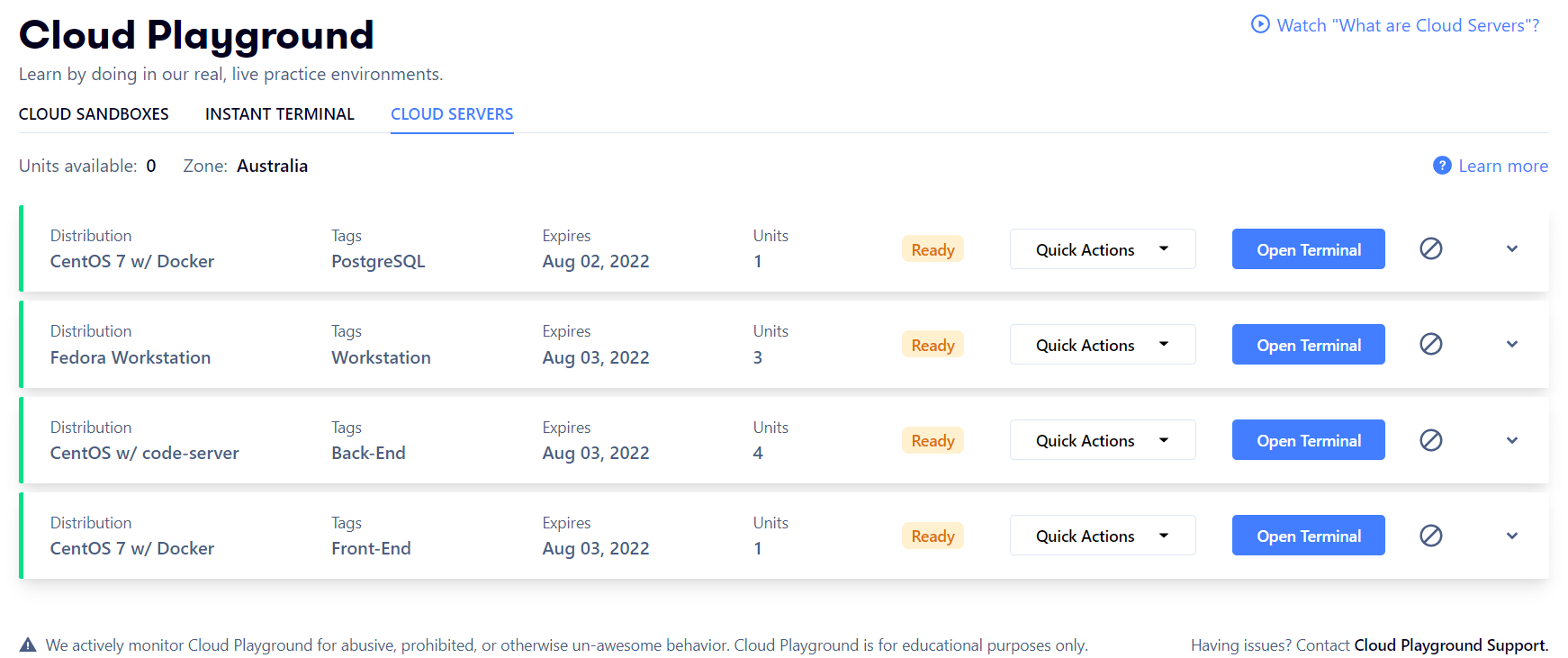
You could continue yourself to into SQL and do more exercise by using Cloud systems.

As you know, we are trying to create a web-based system. The first thing Keith does is set up a sample database server and manage it from another server running as Workstation. In our study, I used rmp, you can practice a lot by using dpkg(Debian-Ubuntu) and changing the variations by making Cloud servers AWS or Google Cloud.

We have just installed the workstation server and database now and there are many components such as front-end & back-end in cloud computing architecture.

As you know, In this way, instead of reloading the whole system by creating a server for the front-end, we can divide the front-end or back-end into parts using microservices. As a cloud engineer, we do the development, deployment and operations(DevOps) of the images of these services, which are update by the developers, and other contributor assigned in task manager.

As an example cloud servers that we can use for the website we will create



More power to your elbow.

Author : Mehmet A. Kir

Source : [[Link](https://github.com/mehmetalikir/CloudComputing/blob/main/Python/Building%20an%20IT%20System%20via%20Cloud%20Part%20I%20.pdf)]