Setting up PostgreSQL on Ubuntu 18.04

Installing PostgreSQL

First we need to update the apt repositories on our Ubuntu server. To do this, log into the server and type - sudo apt update

We then need to install PostgreSQL. To do this we use - sudo apt install postgresql postgresql-contrib

The postgresql-contrib adds additional functionality and is optional.

Once PostgreSQL is installed, we need to start it. To start the PostgreSQL software we use - sudo service postgresql start

Creating Roles for the database

By default PostgreSQL gives us a 'postgres' user. As we have no need to create multiple users for this project, we can just use the 'postgres' user account to set up our database.

To switch over to the 'postgres' user, use - sudo -i -u postgres

If you would like to create a new user, there are two ways to go about this. First, whilst in the postgres user account, type - createuser –interactive

If you prefer to do this from your normal account, you can just use –

sudo -u postgres createuser –interactive

You will then be prompted to add the name of the role you would like to add and if the user will be a superuser. Add your chosen name and decide whether the role will have superuser privileges.

To open the postgres prompt for the new user we need to ensure there is a matching linux user available. To add a new linux user of the same names as the new postgres user, type –

sudo adduser <username>

Once this has been done we can switch to the user and log into the postgres prompt using –

sudo -u <username> psql

Creating a New Database

By default, PostgreSQL creates a database for the user, using the username as the database name. For example, with the postgres user we automatically have a database called postgres.

To access the postgres prompt, type - psql -d postgres

Please note, we are using the default postgres user here, if you have created a user, replace postgres with your new user name in the above command.

If you would prefer to create a different database than the default one above, for a specific role, there are two ways to do this.

First, we can create a new database whilst logged in as the user (e.g postgres user). To do this we type – createdb <database name>

Alternatively, we can do this from the normal account using –

sudo -u postgres createdb <database name>

Once you are logged into the postgres prompt in your chosen role you can check your connection using - \conninfo

This will give you information on the database you are connected to and the user, as well as socket and port information.

Creating Tables

Once logged in to the database, we can now start to create the tables. To check the current tables in the database, type - \d

The first time you log in this should be empty.

For our project, we are using four tables (project, software, project_software and eol). To create these tables, use the following commands:

CREATE TABLE project (project_id serial PRIMARY KEY, host varchar(50) NOT NULL, name varchar(50) NOT NULL, username varchar(50) NOT NULL, password varchar(50), uptime varchar(50), scansuccessful Boolean, timestamp varchar(50), UNIQUE (name));

CREATE TABLE software (software_id serial PRIMARY KEY, name varchar(50) NOT NULL, latest_version varchar(50) NOT NULL);

CREATE TABLE project_software (project_id integer REFERENCES project(project_id), software_id integer REFERENCES software(software_id), installed_version varchar(50) NOT NULL, PRIMARY KEY (project_id, software_id));

CREATE TABLE eol (software_ name varchar(50) NOT NULL, version varchar(50) NOT NULL, eol_date date NOT NULL, PRIMARY KEY(software_name, version));

Please note that in the project and software tables we have used 'serial' data type as the primary key which uses a sequence to generate a surrogate key. When referring to these primary keys as foreign keys in the project_software and eol tables, the data type is integer. This is because the data type serial produces an integer and when we are referencing that column as a foreign key we do not want to create another sequence but rather refer to the integer produced by the sequence in the referenced table.

Insert Data into the Tables

Once the tables are created in the database, we can now add some test data. To do this we use the following commands:

Project

INSERT INTO project (host, name, username, password) VALUES (<insert IP address here>, 'Polarbears', <username>, <password>);

INSERT INTO project (host, name, username, password) VALUES (<insert IP address here>, 'Raahe', , <username>, <password>);

Software

INSERT INTO software (name, latest_version) VALUES ('nodejs', '15.8.0');

INSERT INTO software (name, latest_version) VALUES ('centOS', '8');

INSERT INTO software (name, latest_version) VALUES ('rails', '6.1.2.1');

INSERT INTO software (name, latest_version) VALUES ('ruby', '3.0.0');

INSERT INTO software (name, latest_version) VALUES ('java', '8');

INSERT INTO software (name, latest_version) VALUES ('mysql', '8.0.23');

INSERT INTO software (name, latest_version) VALUES ('postgresql', '13.1');

Project_software

INSERT INTO project_software (project_id, software_id, installed_version) VALUES ('1', '1', '10.23.1');

INSERT INTO project_software (project_id, software_id, installed_version) VALUES ('1', '7', '9.6.20');

EOL

INSERT INTO eol (software_name, version, eol_date) VALUES ('mysql', '10.23.1', '2025-06-01');

INSERT INTO eol (software_name, version, eol_date) VALUES ('CentOS', '9.6.20', '2030-02-01');