PostgreSQL: Getting started

How to start using PostgreSQL

> 👋**Welcome on Stackhero's documentation!**
> Stackhero provides PostgreSQL instances that are ready for production in just 2 minutes!
> Including TLS encryption (aka HTTPS), customizable domain name, PgAdmin, PostGIS, backups and updates in just a click.
> Try our managed PostgreSQL cloud in just 2 minutes

How to use the PostgreSQL `psql` CLI

PostgreSQL can be managed using its official `psql` CLI.
You can install it on your computer and manage your PostgreSQL service remotely.

You can also use it via Docker on your computer. This is the prefered method as it avoids to install `psql` directly on your computer and gives you the ability to switch between versions easily.

To get access to `psql` using Docker, simply run this command: `docker run -it postgres:14-alpine /bin/bash`
Don't forget to replace `14` with the major version number of your current PostgreSQL service.

Then use `psql` to connect to your service:

```
psql \
  --host=<STACKHERO_POSTGRESQL_HOST> \
  --username=admin \
  --dbname=admin
```

How to create a user and database in PostgreSQL using the `psql` CLI

Per default an `admin` user is created with admin rights.
A good practice is to create a dedicated user and database for each project you want to host.

Create a user

To create a user on PostgreSQL, you can use the `psql` CLI with the following SQL query:

```
CREATE ROLE "myProject" WITH
  LOGIN
  NOSUPERUSER
  NOCREATEDB
  NOCREATEROLE
  NOINHERIT
  NOREPLICATION
  CONNECTION LIMIT -1
  PASSWORD 'secretPassword';
```

> Don't forget to replace `myProject` with your project name and `secretPassword` with a secured password.
> Also a good practice is to use your project name as login and database name. If your project name is "superWebsite", we recommend to create a user named "superWebsite" as a database named "superWebsite".

> You can generate a secured password with this command line: `< /dev/urandom tr -dc _A-Z-a-z-0-9 | head -c${1:-32};echo;`

Create a database

To create a database on PostgreSQL, you can use the `psql` CLI with the following SQL query:

```
CREATE DATABASE "myProject"
  WITH
  OWNER = "myProject"
  ENCODING = 'UTF8'
  CONNECTION LIMIT = -1
```

```
IS_TEMPLATE = False;
```

A good practice is to define the same name for the database and the user.
For example, if your project name is "superWebsite", we recommend to create a user named "superWebsite" as a database named "superWebsite".

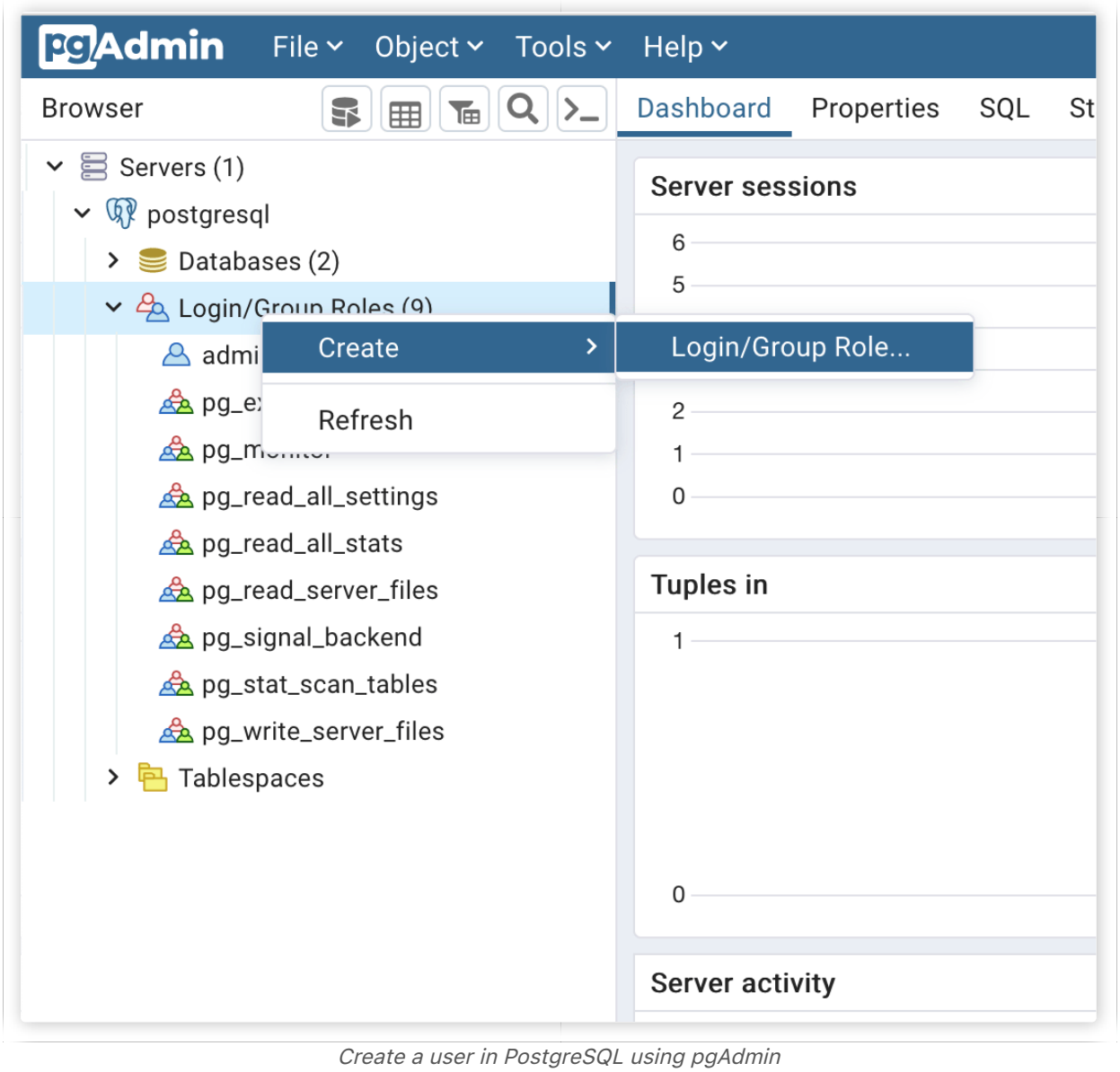
How to create a user and database in PostgreSQL using pgAdmin web UI

Per default an `admin` user is created with admin rights.
A good practice is to create a dedicated user and database for each project you want to host.

To connect to your pgAdmin web UI, simply connect to your PostgreSQL domain name using https (example: `https://XXXXXX.stackhero-network.com`).
Then login using `admin` as user and the password defined in your service configuration (in your Stackhero dashboard).

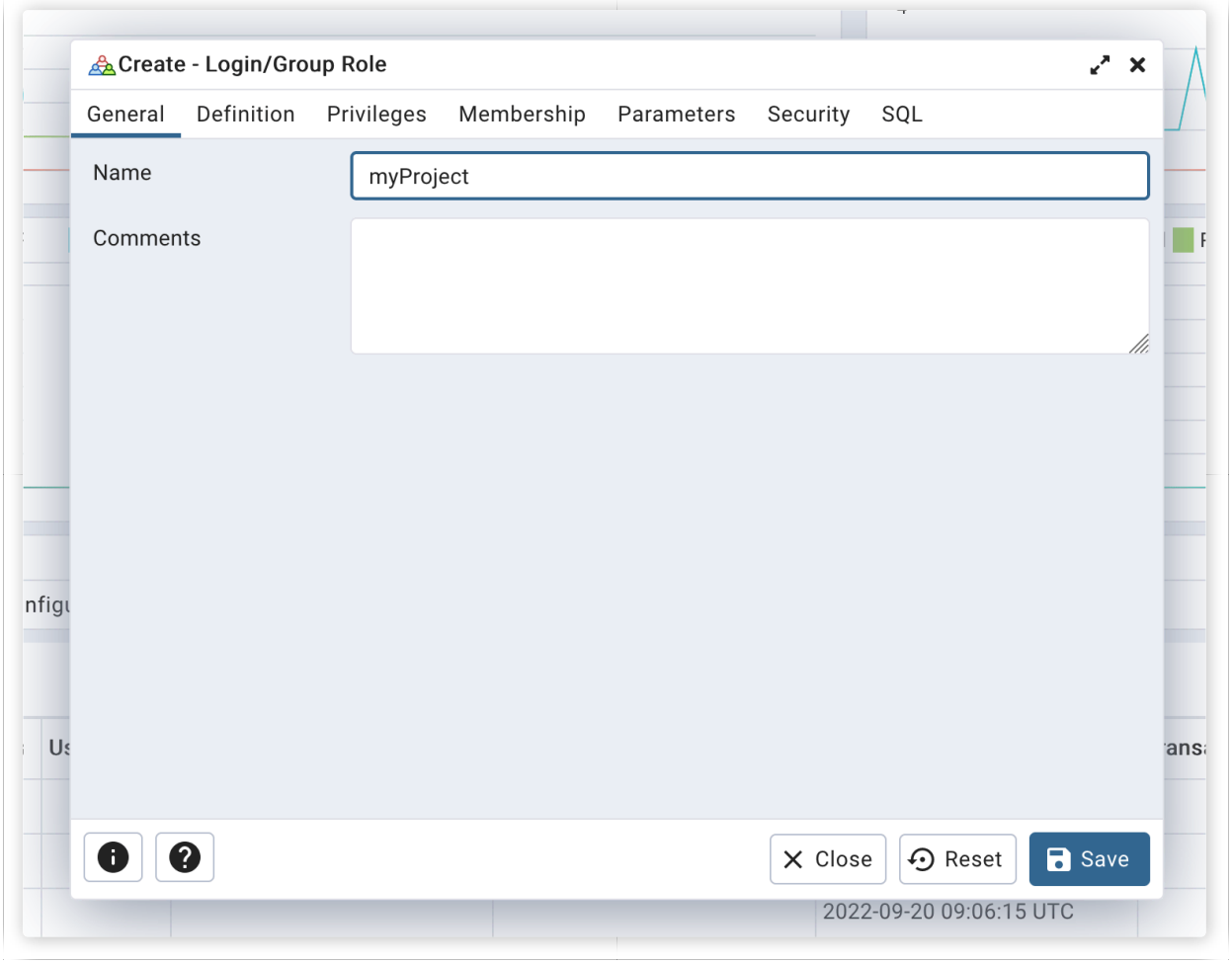
Create a user

Go to `Servers` / `postgresql`, click right on `Login/Group Roles` and select `Create` / `Login/Group Role`:



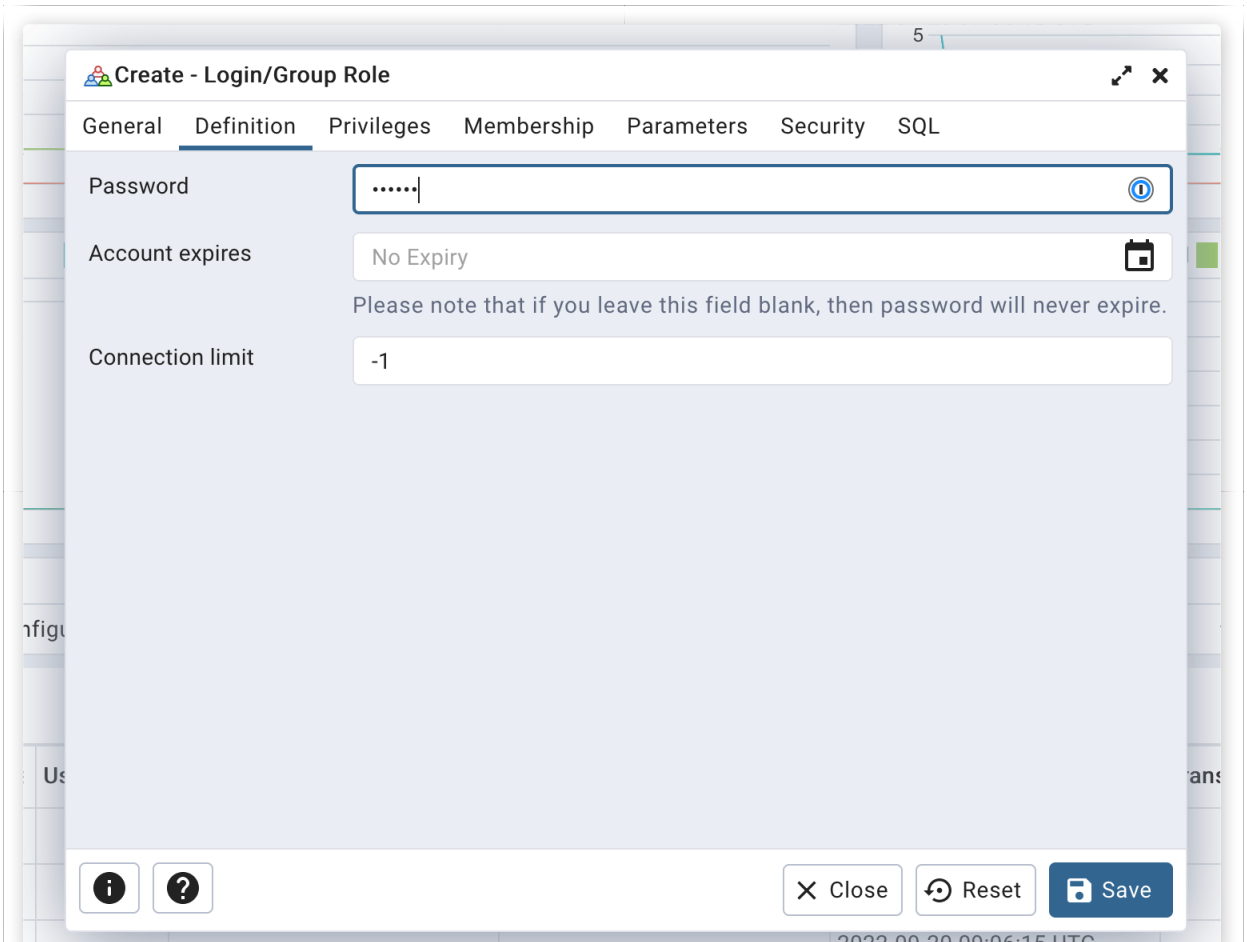
Create a user in PostgreSQL using pgAdmin

Then set the login name:



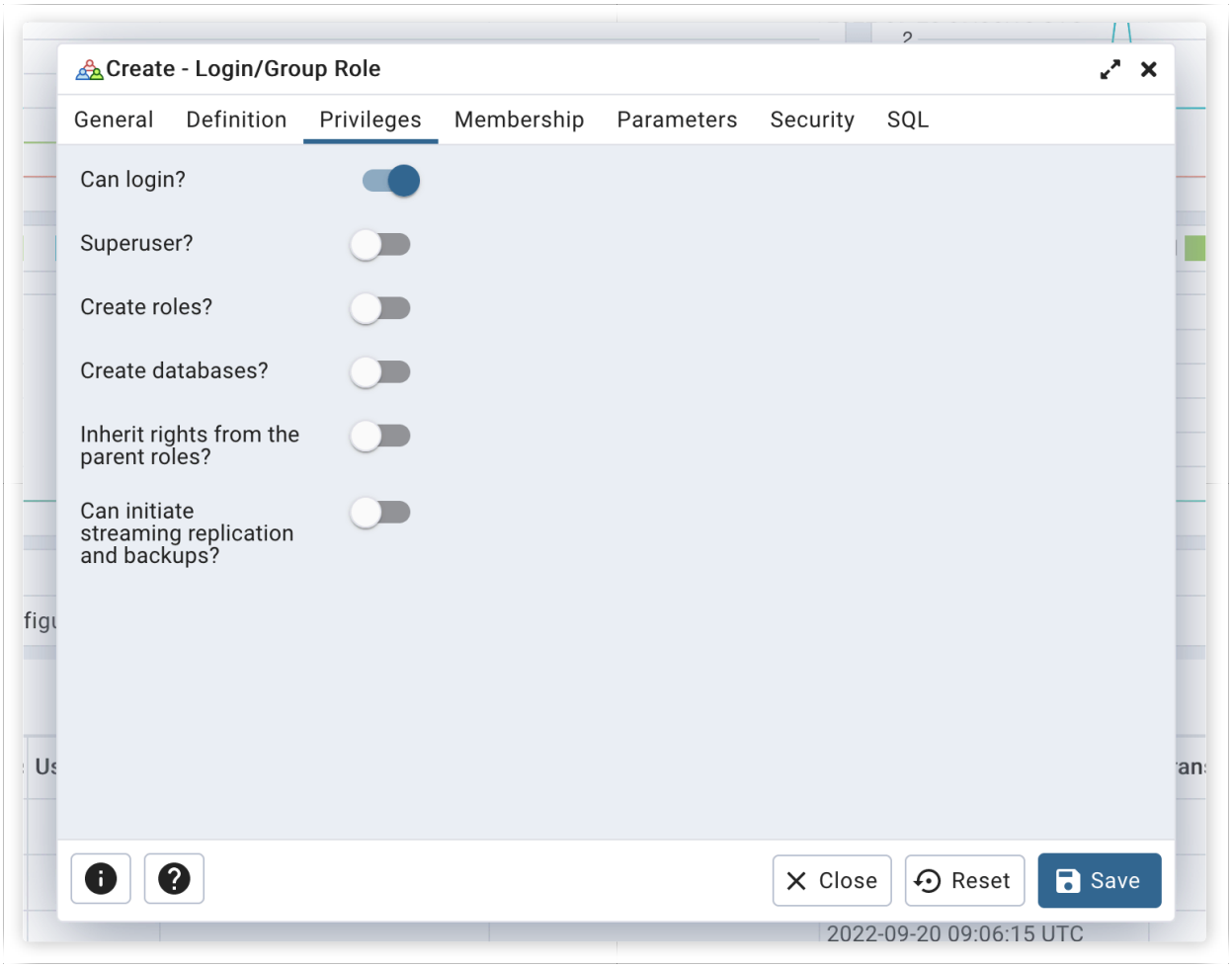
Define user login

And the password (please create a secured one to avoid bruteforce attacks!):



Define user password

And finally set the privileges to "Can login" only:

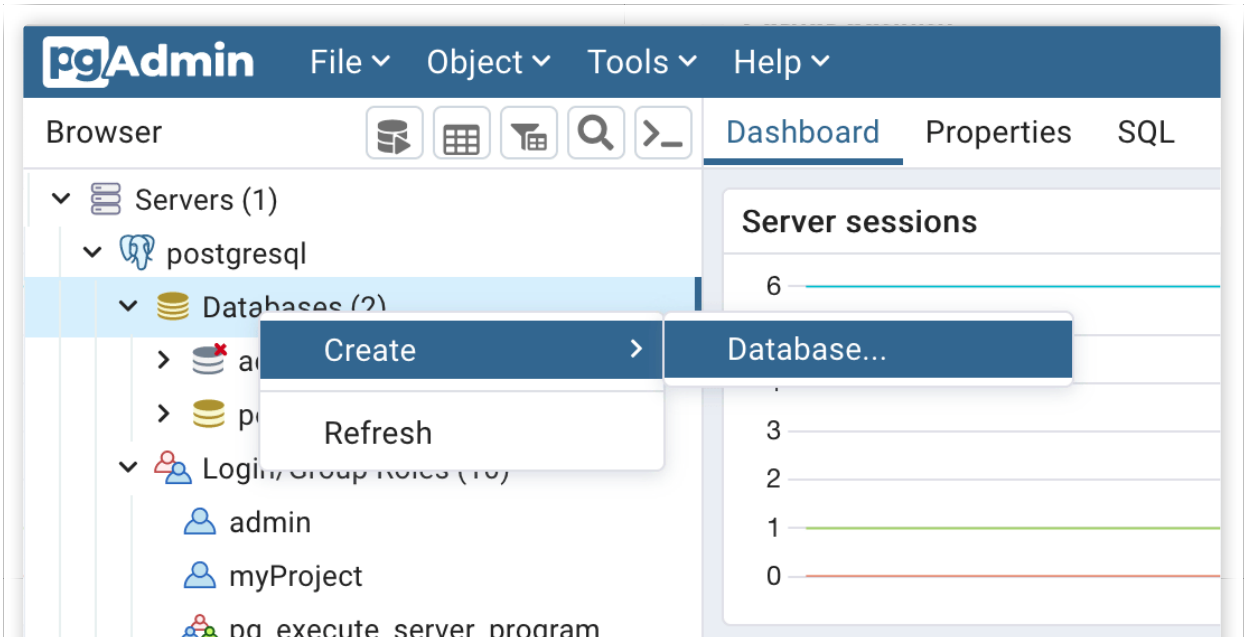


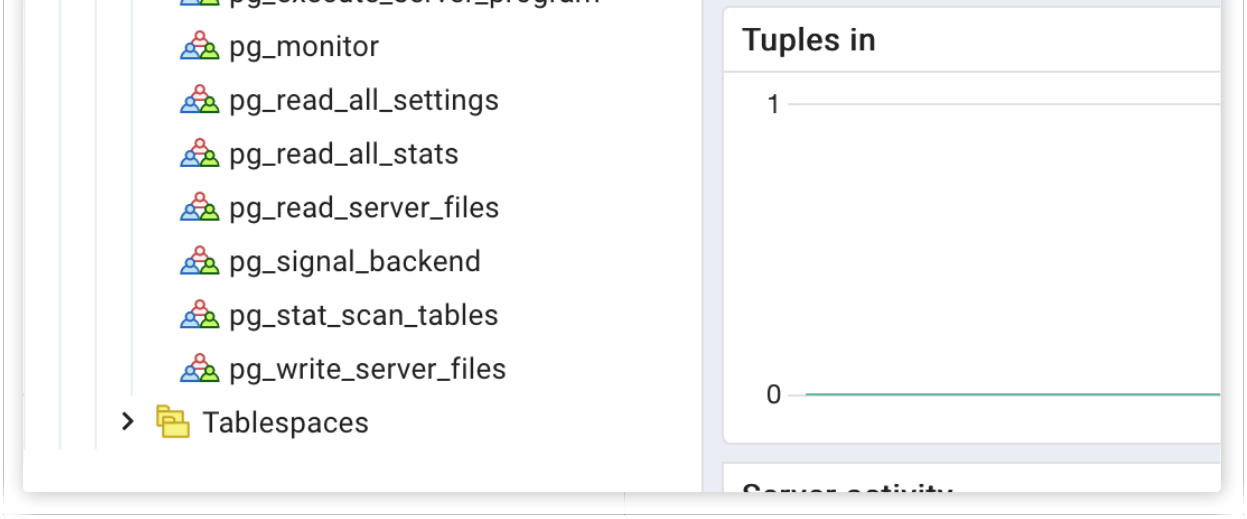
Define user rights

Click on the "Save" button and that's it, your user is created :)

Create a database

Go to `Servers` / `postgresql`, click right on `Databases` and select `Create` / `Database...`:

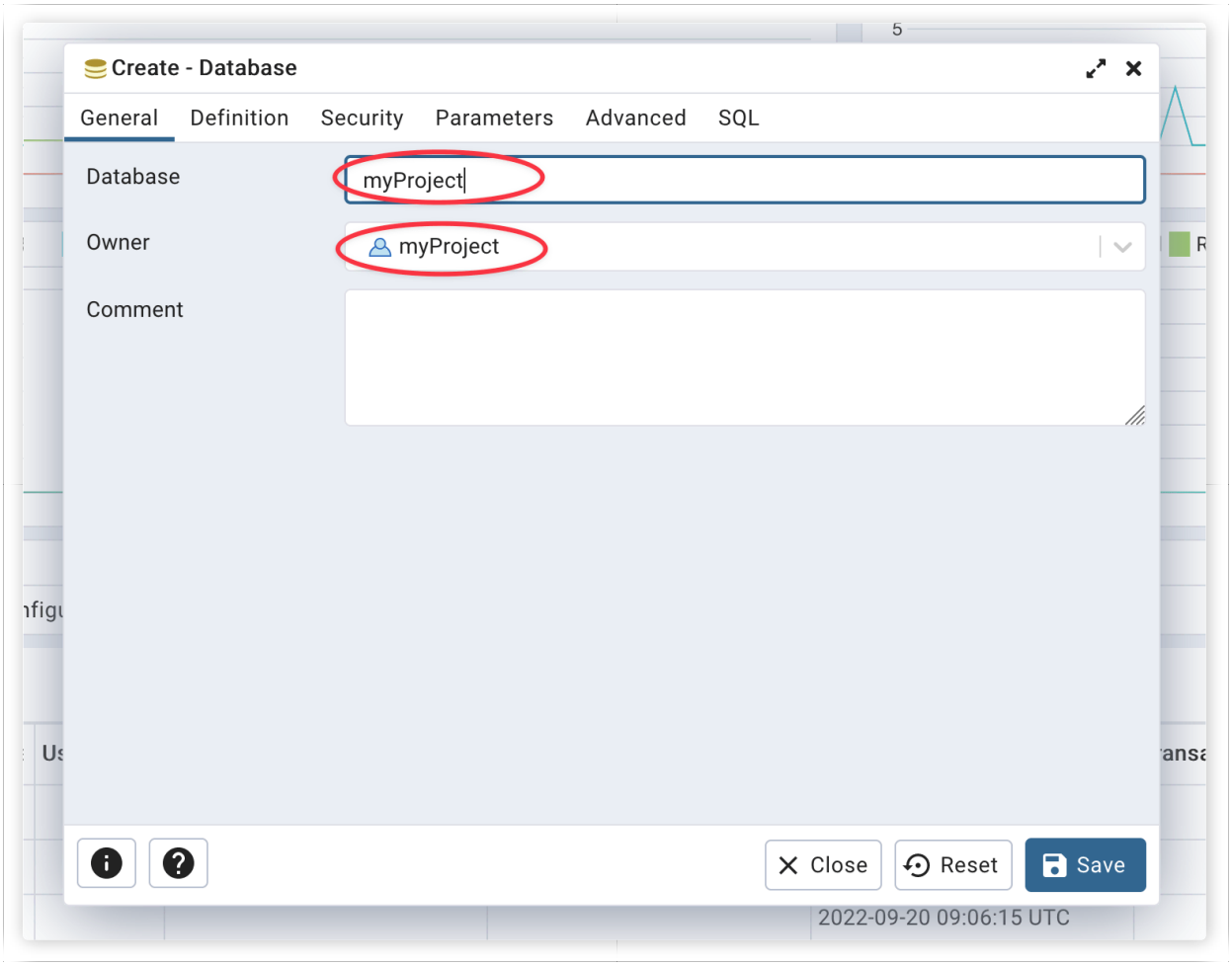




Create a database using pgAdmin

> A good practice is to define the same name for the database and the user.
> For example, if your project name is "superWebsite", we recommend to create a user named "superWebsite" as a database named "superWebsite".

Then set the database name and select the owner, which is the user you have just created before:



Defined database name and owner

That's it, your database is now created :)

How to import pgsql data from your computer to PostgreSQL

The easiest way to import data from your computer to your PostgreSQL instance is to use the PostgreSQL CLI.

On your computer, simply run this command (replace `<DB_NAME>` per your database name and `data.pgsql` with your SQL file):

```
psql \
  --host=<STACKHERO_POSTGRESQL_HOST> \
  --username=admin \
  --dbname=<DB_NAME> \
  < data.pgsql
```

How to export data from PostgreSQL to your computer

As for importing data, the easiest way to export them from your PostgreSQL instance to your computer is to use the PostgreSQL CLI.

On your computer, simply run one of these commands:

- To export the whole database (replace `<DB_NAME>` per your database name):

```
$ pg_dump \
  --host=<STACKHERO_POSTGRESQL_HOST> \
  --username=admin \
  --dbname=<DB_NAME> \
  > data.pgsql
```

- To export only a table (replace `<DB_NAME>` per your database name and `<TABLE_NAME>` per your table name):

```
$ pg_dump \
  --host=<STACKHERO_POSTGRESQL_HOST> \
  --username=admin \
  --dbname=<DB_NAME> \
  --table=<TABLE_NAME> \
  > data.pgsql
```

How to connect to PostgreSQL from Node.js

You will find a complete example of code to use PostgreSQL with Node.js and the pg library with async/await on this git repository: https://github.com/stackhero-io/postgresqlGettingStarted.

How to activate PostgreSQL PostGIS extension

PostGIS extension is included with our PostgreSQL service.

You have to activate it on each database where you want to use it.

To activate it, connect to your database and simply execute this query:
`CREATE EXTENSION postgis;`

Check then that PostGIS is working by checking its version:
`SELECT PostGIS_Full_Version();`

Or get the list of every PostGIS extensions installed:
`SELECT * FROM pg_extension WHERE extname LIKE 'postgis%';`

You can add these other extensions if necessary, but we strongly recommend to *NOT* activate extensions you will not use:

```
-- Enable PostGIS
CREATE EXTENSION postgis;

-- Enable raster support
CREATE EXTENSION postgis_raster;

-- Enable Topology
CREATE EXTENSION postgis_topology;

-- Fuzzy matching needed for Tiger
CREATE EXTENSION fuzzystrmatch;

-- Rule based standardizer
CREATE EXTENSION address_standardizer;

-- Example rule data set
CREATE EXTENSION address_standardizer_data_us;

-- Enable US Tiger Geocoder
CREATE EXTENSION postgis_tiger_geocoder;
```

> Caution: do not activate PostGIS on the database `postgres` !

How to deactivate PostgreSQL PostGIS extension

To deactivate PostGIS, simply connect to the database where PostGIS is installed and run this query: `DROP EXTENSION postgis;`

Connect from Node.js/TypeORM to PostgreSQL

To connect from Node.js/TypeORM, you have to add the `ssl` flag to true like in this example:

```
createConnection(
  driver: {
    type: 'postgres',
    host: '<STACKHERO_POSTGRESQL_HOST>',
    port: 5432,
    username: '<USERNAME>',
    password: '<PASSWORD>',
    database: '<DATABASE>',
    extra: {
      ssl: true
    }
  }
);
```

Our

vice

- GitLab
- Grafana
- Graylog
- InfluxDB
- Kibana
- MariaDB
- Matomo
- Mattermost
- Memcached
- Mercure-Hub
- MinIO
- Mosquitto
- MySQL
- Nextcloud
- Node-RED
- Node.js
- PHP
- Postfix
- PostgreSQL
- Prometheus
- Python
- RabbitMQ
- Redis®*
- RethinkDB

Privacy Policy

Documentations

Support

Status

English

Global