Setting up a local PostgreSQL instance using Docker to run the Astronomy on BlueSky services locally

Doug Hilton

April 2025

Abstract

The BlueSky Astronomy feed bots are written in Python, with an SQL backend running PostgreSQL on a cloud based Linux server. In order for the development team to be able to test new features and code changes, they need to have a local instance of the database running where they can freely make changes without fear of breaking production. This document walks a developer on the BlueSky Astronomy feed dev team through setting up that local environment, shortcutting much of the overhead of installing and administering PostgreSQL through the use of the official PostgreSQL Docker image. Once the environment is setup, developers will be able to refresh their local development database with sample data from production through a series of SQL scripts retrieved from a secure web-service running on the bot's Flask server.

1 Docker Install/Setup

- Download and install Docker Desktop from http://www.docker.com. You will need to create an account on the docker website in order to download the software.
- Once you have Docker installed, launch the application once to initialize the environment and complete setup
- Launch a shell on your local computer
- Grab the PostgreSQL docker image by executing:

```
docker pull postgres:alpine
```

- Create a volume to store store the data so docker restarts don't delete your database:

```
docker volume create postgres-data
```

2 Initialize BlueSky PostgreSQL Database

- Start Docker and create a new PostgreSQL instance (note: you will not repeat this step after the database is initialize. Your Docker and Postgres instance can be started/stopped via the Docker UI - see below):

```
docker run --name BlueSkyDB -e POSTGRES_PASSWORD={password} \
-d -p 5432:5432 -v postgres-data:/var/lib/postgresql/data \
-d postgres:alpine
```

- Log into the newly created BlueSkyDB PostgreSQL database using username: `postgres` and the password you chose in the last step.
- Create a new account to be used for the BlueSky application:

```
CREATE DATABASE bluesky_astronomy;
CREATE USER bluesky_astronomy WITH ENCRYPTED PASSWORD 'yourpass';
GRANT ALL PRIVILEGES ON DATABASE bluesky_astronomy TO bluesky_astronomy;
```

- Log out of the PostgreSQL database
- Locate the set of bootstrap .sql files stored on github and download those to a folder on your local machine.
- Create the database structure by executing the `bootstrap.sql` file with the below command. This will in turn execute each of the .sql files to create each table in your environment. Note that this will drop existing tables and recreate them with the new schema.

```
psql -U bluesky_astronomy -d bluesky_astronomy -a -f ./bootstrap.sql
```

3 Using PostgreSQL day-to-day

Once Docker is installed and your PostgreSQL database for BlueSky is initialized, you can use the Docker GUI to start/stop your container, which will also start/stop your PostgreSQL instance.

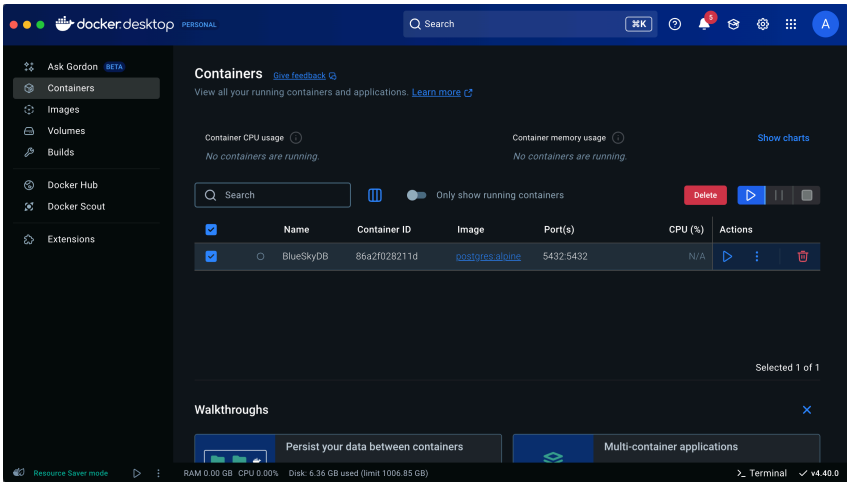

Figure 1: The main Docker screen showing a running PostgreSQL image

I highly recommend downloading and using a graphical user interface (GUI) to interact with your PostgreSQL database. pgAdmin is what I use, and it runs on any OS. There is even a Docker image if you want to go that route.

4 Reference

- Docker and PostgreSQL in [10 Minutes]
- Run Postgres in a Docker Container (Easiest PostreSQL Setup)