https://www.digitalocean.com/community/tutorials/how-to-move-a-postgresql-data-directory-to-a-new-location-on-ubuntu-16-04

digitalocean.com

How To Move a PostgreSQL Data Directory to a New Location on Ubuntu 16.04

By: Melissa Anderson

Go to page 5 to see quick process.

Introduction

Databases grow over time, sometimes outgrowing the space on their original file system. You can also run into I/O contention when they're located on the same partition as the rest of the operating system. RAID, network block storage, and other devices can offer redundancy and other desirable features. Whether you're adding more space, evaluating ways to optimize performance, or looking to take advantage of other storage features, this tutorial will guide you through relocating PostgreSQL's data directory.

Prerequisites

To complete this guide, you will need:

- **An Ubuntu 16.04 server with a non-root user with `sudo` privileges**. You can learn more about how to set up a user with these privileges in our Initial Server Setup with Ubuntu 16.04 guide.
- **A PostgreSQL server**. If you haven't already set one up, the How To Install and Use PostgreSQL on Ubuntu 16.04 guide can help you.

In this example, we're moving the data to a block storage device mounted at `/mnt/volume-nyc1-01`. If you are using Block Storage on DigitalOcean, this guide can help you mount your volume before continuing with this tutorial.

No matter what underlying storage you use, the following steps can help you move the data directory to a new location.

Step 1 — Moving the PostgreSQL Data Directory

To prepare for moving PostgreSQL's data directory, let's verify the current location by starting an interactive PostgreSQL session. In the line below, `psql` is the command to enter the interactive monitor, and `-u postgres` tells sudo to execute `psql` as the system's postgres user:

```
sudo -u postgres psql
```

Once you've entered the monitor, select the data directory:

```
SHOW data_directory;
```

```
Output       data_directory
------------------------------
/var/lib/postgresql/9.5/main
(1 row)
```

This output confirms that PostgreSQL is configured to use the default data directory, `/var/lib/postgresql/9.5/main`, so that's the directory we need to move. Once you've confirmed the directory on your system, type `\q` to quit.

To ensure the integrity of the data, we'll shut down PostgreSQL before we actually make changes to the data directory:

```
sudo systemctl stop postgresql
```

`systemctl` doesn't display the outcome of all service management commands. To verify you've succeeded, use the following command:

```
sudo systemctl status postgresql
```

You can confirm it's shut down if the final line of the output tells you the server is stopped:

```
Output. . .
Jul 22 16:22:44 ubuntu-512mb-nyc1-01 systemd[1]: Stopped PostgreSQL RDBMS.
```

Now that the server is shut down, we'll copy the existing database directory to the new location with `rsync`. Using the `-a` flag preserves the permissions and other directory properties while `-v` provides verbose output so you can follow the progress.

Note: Be sure there is no trailing slash on the directory, which may be added if you use tab completion. When there's a trailing slash, `rsync` will dump the contents of the directory into the mount point instead of transferring it into a containing `PostgreSQL` directory:

We’re going to start the `rsync` from the `postgresql` directory in order to mimic the original directory structure in our new location. By creating that `postgresql` directory within the mount-point directory and retaining ownership by the PostgreSQL user, we can avoid permissions problems for future upgrades. The version directory, `9.5` isn’t strictly necessary since we’ve defined the location explicitly in the `postgresql.conf` file, but following the project convention certainly won’t hurt, especially if there’s a need in the future to run multiple versions of PostgreSQL.

```
sudo rsync -av /var/lib/postgresql /mnt/volume-nyc1-01
```

Once the copy is complete, we'll rename the current folder with a .bak extension and keep it until we’ve confirmed the move was successful. By re-naming it, we’ll avoid confusion that could arise from files in both the new and the old location:

```
sudo mv /var/lib/postgresql/9.5/main /var/lib/postgresql/9.5/main.bak
```

Now we’re ready to turn our attention to configuration.

Step 2 — Pointing to the New Data Location

PostgreSQL has several ways to override configuration values. By default, the `data_directory` is set to `/var/lib/postgresql/9.5/main` in the `/etc/postgresql/9.5/main/postgresql.conf` file. Edit this file to reflect the new data directory:

```
sudo nano /etc/postgresql/9.5/main/postgresql.conf
```

Find the line that begins with `data_directory` and change the path which follows to reflect the new location.

In our case, the updated file looks like the output below:

/etc/postgresql/9.5/main/postgresql.conf

```
. . .
data_directory = '/mnt/volume-nyc1-01/postgresql/9.5/main'
. . .
```

Step 3 — Restarting PostgreSQL

We're ready to start PostgreSQL.

```
sudo systemctl start postgresql
sudo systemctl status postgresql
```

To make sure that the new data directory is indeed in use, start the PostgreSQL monitor.

```
sudo -u postgres psql
```

Look at the value for the data directory again:

```
SHOW data_directory;

Output            data_directory
----------------------------------------
/mnt/volume-nyc1-01/postgresql/9.5/main
(1 row)
```

Now that you've restarted PostgreSQL and confirmed that it's using the new location, take the opportunity to ensure that your database is fully functional. Once you've verified the integrity of any existing data, you can remove the backup data directory:

```
sudo rm -Rf /var/lib/postgresql/9.5/main.bak
```

Restart PostgreSQL one final time to be sure that it works as expected:

```
sudo systemctl restart postgresql
sudo systemctl status postgresql
```

Conclusion:

If you've followed along, your database should be running with its data directory in the new location and you've completed an important step toward being able to scale your storage. You might also want to take a look at 5 Common Server Setups For Your Web Application for ideas on how to create a server infrastructure to help you scale and optimize web applications.

Recapitulation:

Step 1 — Moving the PostgreSQL Data Directory

psql -U postgres -d postgres -h isdevelopment.us

SHOW data_directory;

Output
data_directory

/var/lib/postgresql/9.6/main
(1 row)

\q

sudo systemctl stop postgresql
sudo systemctl status postgresql

Output. . .
Jul 22 16:22:44 ubuntu-512mb-nyc1-01 systemd[1]: Stopped PostgreSQL RDBMS.

Start the rsync from the postgresql directory in order to mimic the original
directory structure in our new location.

sudo rsync -av /var/lib/postgresql /home/Data-1/

Once the copy is complete, we'll rename the current folder with a .bak extension and keep it until we've confirmed the move was successful.

sudo mv /var/lib/postgresql/9.6/main/ /var/lib/postgresql/9.6/main.bak

Step 2 — Pointing to the New Data Location

sudo vim /etc/postgresql/9.6/main/postgresql.conf

Find the line that begins with data_directory and change the path which follows to reflect the new location.

data_directory = '/home/Data-1/postgresql/9.6/main'

Step 3 — Restarting PostgreSQL

sudo systemctl start postgresql
sudo systemctl status postgresql

psql -U postgres -d postgres -h isdevelopment.us

SHOW data_directory;

```
 output     data_directory
--------------------------------
 /home/Data-1/postgresql/9.6/main
(1 row)
```

Now that you've restarted PostgreSQL and confirmed that it's using the new location, take the opportunity to ensure that your database is fully functional. Once you've verified the integrity of any existing data, you can remove the backup data directory:

sudo rm -Rf /var/lib/postgresql/9.6/main.bak/

Restart PostgreSQL one final time to be sure that it works as expected:

sudo systemctl restart postgresql
sudo systemctl status postgresql