PostgreSQL pg_dump Backup and pg_restore Restore Guide

Guide of PostgreSQL pg_dump Backup Command

You can use the pg_dump command to backup your PostgreSQL database. Even if others are accessing the database, pg_dump will still back it up, and it will not block others from reading or writing to it.

What is the pg_dump command?

PostgreSQL pg_dump is a database tool that helps you make automatic, consistent backups. For example, you can back up offline and online databases. The utility creates a set of SQL statements and processes them against the database instance to create a dump file that can use to restore the database later.

pg_dump example

With the pg dump command, a database can be exported as an archive or a script file, including SQL commands for reassembling the database. The main objective of this tool is to backup databases.

Requirements

- A server running Linux operating system with PostgreSQL installed.
- A root password is setup on your server.

A brief explanation of all available options is shown below:

1. **-U** : Specify the PostgreSQL username.
2. **-W** : Force the pg_dump command to ask for a password.
3. **-F** : Specify the format of the output file.
4. **-f** : Specify the output file.
5. **p** : Plain text SQL script.
6. **c** : Specify the custom formate.
7. **d** : Specify the directory format.
8. **t** : Specify tar format archive file.

Following is the basic syntax of the PostgreSQL pg_dump:

```
pg_dump -U username -W -F t database_name > c:\backup_file.tar
```

Backup a Single PostgreSQL Database

You will need to use the **pg_dump** tool to backup a PostgreSQL database. This tool will dump all content of a database into a single file.

The basic syntax to backup a PostgreSQL database is shown below:

```
pg_dump -U [option] [database_name] > [backup_name]
```

For example, create a backup of the PostgreSQL database named db1 in the tar format, and run the following command:

```
pg_dump -U postgres -F c db1 > db1.tar
```

If you want to save the backup in a directory format, run the following command:

```
pg_dump -U postgres -F d db1 > db1_backup
```

If your database is extensive and wants to generate a small backup file, you can use pg_dump with a compression tool such as gzip to compress the database backup.

```
pg_dump -U postgres db1 | gzip > db1.gz
```

You can also reduce the database backup time by dumping number_of_jobs tables simultaneously using the -j flag.

```
pg_dump -U postgres -F d -j 5 db1 -f db1_backup
```

Note: Remember that the above command will reduce the backup time and increase the server's load.

Backup All PostgreSQL Databases

PostgreSQL provides a simple tool (pg_dumpall) to back up all your databases using a single command. This tool will dump all

PostgreSQL databases of a cluster into one script file.

The basic syntax of the **pgdump_all** command is shown below:

```
pg_dumpall -f backupfile_name.sql
```

The above command will dump all databases to a single file named **backupfile_name.sql**.

Backup a Remote PostgreSQL Database

To perform the database backup on the remote PostgreSQL server, you must configure your PostgreSQL server to allow a remote connection.

The basic syntax to backup a remote PostgreSQL database is shown below:

```
pg_dump -h [remote-postgres-server-ip] -U [option] [database_name] > [backup_name]
```

For example, create a backup of the PostgreSQL database on the remote server (**192.168.0.100**) with the name **remote_db1** in the **tar** format, and run the following command:

```
pg_dump -h 192.168.0.100 -U postgres -F c remote_db1 > remote_db1.tar
```

Guide to pg_restore Restore Command

pg_restore is a database tool for restoring a backup in an access-friendly format created by pg_dump in one of the non-plain text formats. The pg_restore create database will reconstruct the data according to how it was originally saved and allow access to that information.

What is the pg_restore command?

The pg_restore command is utilized to restore a database from an archive generated by pg_dump (which produces and saves the data snapshot). This function will issue appropriate commands to reconstruct the contained database back into its initial state at the backup time.

pg_restore Example

Following is the basic syntax of the PostgreSQL pg_restore:

The basic syntax to restore a database with pg_restore is shown below:

```
pg_restore -U [option] [db_name] [db_backup]
```

A brief explanation of each option is shown below:

-c : Used to drop database objects before recreating them.

-C : Used to create a database before restoring it.

-e : Exit if an error has been encountered.

-F format: Used to specify the format of the archive.

Restore a Single PostgreSQL Database

If you choose custom, directory, or archive format when taking a database backup, you need to use the **pg_restore** command to restore your database.

For example, to restore a backup from the file db1.tar, you will need to consider two options:

- If the database already exists.
- The format of your backup.

If your database already exists, you can restore it with the following command:

```
pg_restore -U postgres -Ft -d db1 < db1.tar
```

If your database does not exist, you can restore it with the following command:

```
pg_restore -U postgres -Ft -C -d db1 < db1.tar
```

Restore All PostgreSQL Databases

You can use psql command to restore all PostgreSQL databases.

The basic syntax to restore all databases is shown below:

```
psql -f [db_backup.sql]
```

For example, restore a backup from the backupfile_name.sql file, run the following command:

```
psql -f backupfile_name.sql
```

Restore a Remote PostgreSQL Database

The basic syntax to restore a remote PostgreSQL database is shown below:

```
pg_restore -h [remote-postgres-server-ip] -U [option] [database_name] <
[backup_name]
```

For example, restore a database from the file **remote_db1.tar** on the remote server (**192.168.0.100**), and run the following command:

```
pg_dump -h 192.168.0.100 -U postgres -Ft remote_db1 < remote_db1.tar
```

Schedule PostgreSQL Database Backup Automatically

You can also use **cron jobs** to perform backups at regular intervals. Cron jobs are used to schedule tasks at specified intervals on your server.

You can edit the cron jobs by running the following command:

```
crontab -e
```

Add the following lines at the end of the as per your requirements:

```
30 20 * * * pg_dump -U postgres your_db > /root/backup_db.sql
```

Save and close the file when you are finished.

The above jobs will run daily at **8:30 PM** and create a backup file at **/root/backup_db.sql**.