How to automate database backups with backupninja

by Jack Wallen in Data Centers on January 10, 2017, 8:42 PM PST

Your databases need to be backed up with regularity. Learn how to do that with the help of the Linux tool backupninja.

Image: Jack Wallen

If you have Linux database servers in your data center, chances are you're going to want to back up those databases.

There are a number of routes to this success, all of which will employ various moving parts. One tool I prefer to use for my database backups is backupninja. This application is a command line tool that does an outstanding job of doing daily automated backups, so you don't have to worry about losing everything, should disaster strike.

Let's walk through the process of installing and using backupninja. I'll demonstrate on an instance of Ubuntu Server 16.04 running MySQL. **Note:** backupninja can back up more than just MySQL; it can also work with the following and more:

- maildir
- makecd
- PostgreSQL
- rdiff
- rsync

Data Center Must-Reads

- 8 data center predictions for 2020
- 7 networking predictions for 2020: Automation, edge computing, Wi-Fi 6, more
- Server virtualization best practices and tips on what not to do
- Quantum computing: Seven truths you need to know

Installing backupninja

To install backupninja, you need to log into your Ubuntu Server and issue the following commands:

```
sudo apt-get update
sudo apt-get install backupninja duplicity rdiff-backup
```

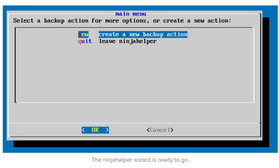
Once this completes, you are ready to set up your automated backup.

SEE: End user data backup policy (Tech Pro Research)

Setting up backupninja

The backupninja application comes with a handy wizard that will help you set up your backup. From the terminal window, issue the command *sudo ninjahelper*. When you see the prompt to create a new backup (**Figure A**), select New, tap down to OK, and hit Enter on your keyboard.

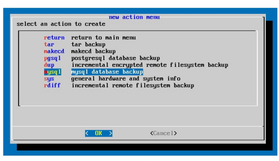
Figure A



The ninjahelper wizard is ready to go.

In the next screen, tap the down arrow button on your keyboard to highlight mysql (**Figure B**).

Figure B

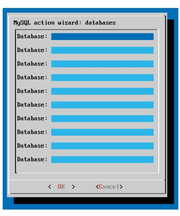


Selecting a MySQL backup.

In the next window, define where you want to store the backups (the default is /var/backups/mysql). Either use the default or set this to a directory of your choosing, tab down to OK, and hit the Enter key on your keyboard.

Now you'll be asked if you want to back up all of your databases. You can either select yes and continue on, or you can select No and then define which individual databases you want to back up. If you opt for individual databases, you will be required to enter the names of the databases to be backed up (**Figure C**).

Figure C



Naming individual databases for backup.

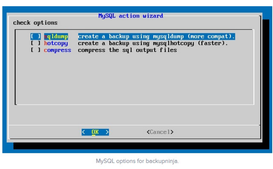
The next step is choosing an authentication method from three options:

- User: a standard Linux user
- Password: specify a MySQL user and password
- Debian: use default mysql user debian-sys-main

We'll go with password. Select the Password option, tab down to OK, and hit Enter. In the next window, enter the MySQL user (one that has full privileges to access all the databases to be backed up). After you type that user, tab down to OK and hit Enter. In the next window, type the MySQL user password, tab down to OK, and hit Enter.

Now you must select the options for your MySQL backup (**Figure D**). I suggest enabling sqldump and compress (if your databases are large), as that backup will be more compatible and the storage size is smaller. Use the arrow keys on your keyboard to highlight an option and then tap the spacebar to enable the option. After you make your selections, tab down to OK, and hit Enter.

Figure D



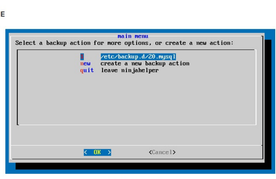
MySQL options for backupninja.

Next you must select a backup action. By default, backupninja will create a numbered action that is next in line for whatever other backups you have; so, if you already have 10.mysql, the wizard will create the next action to be 20.mysql. The actions are executed in numerical order, which means 10.mysql will execute before 20.mysql. You can manually create a new backup action by selecting new, tabbing down to OK, and hitting Enter. The new actions you can create include:

- **tar:** create a tar backup
- **makecd:** create a makecd backup
- **pgsql:** create a PostgreSQL backup
- **dup:** create an incremental encrypted remote filesystem backup
- **mysql:** create a MySQL backup
- **sys:** create a general hardware and system information backup
- **rdiff:** create an incremental remote filesystem backup

Since we already know we're creating a MySQL backup, we'll select the default option (**Figure E**), tab down to OK, and hit Enter. Your setup is complete.

Figure E



Selecting the backup action to use.

The next window will allow you to go back to the main window, view or edit the backup configuration, disable the action, change the filename for the action (backup), run the action, do a test run of the action, or remove the action. I highly recommend allowing backupninja to do a test run of your newly created action, so you can be sure it works as expected.

Your MySQL backup is set and will start running daily.

A simple backup solution

If you're looking for an easy way to set up an automated database backup for your data center, look no further than backupninja. It's a dependable, free, and open source solution to a critical component of your data center. Your company's executives will thank you.

Also see

- How to set up passwordless authentication for MySQL (TechRepublic)
- How to create and populate a database in MySQL (TechRepublic)
- How to install a LAMP stack on CentOS (TechRepublic)
- How to quickly audit a Linux system from the command line (TechRepublic)
- Linux 2017: With great power comes great responsibility (ZDNet)