Source: https://www.digitalocean.com/community/articles/how-to-install-rsnapshot-on-ubuntu-12-04

What the Red Means

The lines that the user needs to enter or customize will be in red in this tutorial!

The rest should mostly be copy-and-pastable.

Introduction

This tutorial will describe how to set up and use rsnapshot, a remote backup solution based on rsync. Rsnapshot leverages the power of rsync to create scheduled, incremental backups.

This procedure will demonstrate how to create local backups, as well as how to use another cloud server or your home computer as backup space. To complete the procedures in this tutorial, you will need to be logged in as root or preface all commands with "sudo".

Step One: Installing Rsnapshot and Configuring SSH Keys

The first step in getting your content backed up is to install rsnapshot on the machine that you would like to use as your backup server.

Rsnapshot is in Ubuntu's default repositories, so we can install it using apt-get:

```
sudo apt-get install rsnapshot
```

In order to backup another cloud server, your backup server will need to be able to connect through SSH to the cloud server you'd like to back up. We want rsnapshot to be able to use SSH without a password, so we need to generate SSH keys to authenticate.

To generate a public and private key run this command on your backup server:

```
sudo ssh-keygen -t rsa
```

We do not want a passphrase for this key because we want these computers to be able to connect to each other without our intervention. Press "ENTER" through all of the prompts to accept the defaults.

```
user@backupserver:~#: sudo ssh-keygen –t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
```

```
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
…
The key's randomart image is:
…
```

We now have a public and private key pair that will allow us to sign in to other servers from our backup server. We need to transfer our public key to the machine we will be backing up so that it knows that we are allowed to access it.

```
sudo ssh-copy-id -i /root/.ssh/id_rsa.pub root@example.com
```

Be sure to change “example.com” to the IP address or domain name of the server you will back up.

Once that command executes, you should check that you are able to log-in to your server from your backup server without a password.

```
sudo ssh root@example.com
```

Once you've verified that you can connect correctly, exit out so that you are able to work on your backup server again.

```
exit
```

Step Two: Configuring Rsnapshot

After you have installed rsnapshot, you will need to edit the configuration file.

```
sudo nano /etc/rsnapshot.conf
```

One thing to keep in mind while you are working with the rsnapshot configuration file is that each directory needs a trailing slash (/) to be read correctly.

The first decision you will need to make is where you would like to store your backups. We will use the directory “/backup” as our backup location. Search for and edit the following variable to set the backup location.

```
snapshot_root                    /backup/
```

If this directory does not already exist, rsnapshot will create it when it runs.

You will also want to uncomment the cmd_ssh line to allow for remote backups. Remove the “#” from in front of the following line so that rsnapshot can securely transfer your data to the backup server.

```
cmd_ssh                 /usr/bin/ssh
```

Remove the “#” from before the cmd_du line to enable accurate disk usage reports.

```
cmd_du                          /usr/bin/du
```

Next, we need to consider how many old backups we would like to keep. Since rsnapshot uses incremental backups, we can afford to keep older backups for awhile before removing them.

We set these up under the “BACKUP INTERVALS” section of the configuration. We tell rsnapshot to retain a specific number of backups of each kind of interval. For the purposes of this guide, the default setting is good, but we will also uncomment the “monthly” interval so that we have some longer-term backups in place. Edit this section to look like this:

```
#########################################
#           BACKUP INTERVALS            #
# Must be unique and in ascending order #
# i.e. hourly, daily, weekly, etc.      #
#########################################

retain          hourly  6
retain          daily   7
retain          weekly  4
retain          monthly 3
```

Finally, you’ll need to decide on what you would like to backup. If you are backing up locally to the same machine, this is as easy as specifying the directories that you want to save and following it with “localhost/” which will be a sub-directory in the snapshot_root that you set up earlier.

We will choose to backup the home directory and the etc directory.

```
backup          /home/          localhost/
backup          /etc/           localhost/
```

If you are backing up a remote server or another droplet (cloud server), this can be configured here too. You just need to tell rsnapshot where the server is and which directories you would like to back up.

```
backup           root@example.com:/home/                remote-droplet/
```

rsnapshot is pull-based. Meaning that, without additional workarounds and configuration, you cannot push the backups to a remote server. Rather, with the SSH keys, they will be pulled into the server with rsnapshot installed.

You’ll want to change the “example.com” portion with the IP address or domain name of the server you wish to back up. After your server name/IP address, you need to put a colon ":" followed by the directory path you would like to back up. The “remote-droplet” portion is the sub-directory where backups will be stored. You can name it whatever you’d like.

We are now finished with the initial configuration of rsnapshot. Save the /etc/rsnapshot.conf file before continuing. If you are using nano to edit, press “Ctrl-X” to exit and answer “Yes” at the prompt to save changes.

Testing Your Configuration

Before setting up the automation, we’ll want to test that everything works as expected. To test that your configuration has the correct syntax, run:

```
sudo rsnapshot configtest
```

If your file is error-free, you will receive a “Syntax OK” message. If you did not receive this message, you need to go back and fix the mistakes that it tells you about.

Next, we want to perform a dry run of one of the snapshot to make sure we are producing the expected results. We use the "hourly" parameter because hourly backups are the basic snapshot type that the other intervals will build off of. That means that we will get more information from "hourly" than if we had choosen "daily" or "weekly".

```
sudo rsnapshot -t hourly
```

If the generated output looks correct, you can remove the “-t” option to try your setup for the first time.

```
sudo rsnapshot hourly
```

This will run the backup that we setup in our configuration file. For this tutorial, rsnapshot created a /backup directory and then created the directory structure under it that organizes our files.

Automating the Process

With rsnapshot working correctly, the only thing left to do is to schedule it to run at certain intervals. We will use cron, a linux scheduler, to make this happen.

Luckily, rsnapshot includes a default cron file that we can edit to our liking. We’re going to edit this file with nano again.

```
sudo nano /etc/cron.d/rsnapshot
```

The scheduling is currently commented out. We’re going to remove the “#” character from the beginning of the scheduling section to activate these values.

```
# This is a sample cron file for rsnapshot.
# The values used correspond to the examples in /etc/rsnapshot.conf.
# There you can also set the backup points and many other things.
```

```
#
# To activate this cron file you have to uncomment the lines below.
# Feel free to adapt it to your needs.

0 */4          * * *          root    /usr/bin/rsnapshot hourly
30 3           * * *          root    /usr/bin/rsnapshot daily
0  3           * * 1          root    /usr/bin/rsnapshot weekly
30 2           1 * *          root    /usr/bin/rsnapshot monthly
```

These settings will run add a snapshot to the “hourly” directory within our “/backup/” directory every four hours, add a daily snapshot everyday at 3:30 am, add a weekly snapshot every Monday at 3:00 am, and add a monthly snapshot on the first of every month at 2:30 am.

It is important to stagger your backups and run larger backup intervals first. This means running the monthly backup first and progressing to shorter intervals from there in order, as we've done in this tutorial. This is necessary so that the program does not get caught up trying to do multiple backups at the same time, which can cause problems.

By Justin Ellingwood