

Remote Backups With Rsync

Contributed by Administrator
 Wednesday, 10 March 2010
 Last Updated Wednesday, 10 March 2010

This tutorial describes how to set up a nightly backup from remote client machines to a centralized server using rsync. It includes example shell scripts and crontab files.

Remote Backups With Rsync

By Noel Davis

In this article we explain how to automate the backup of files on remote machines to a centralized server using rsync.

rsync is a command line utility that is used to synchronize files between two computers over a network to synchronize files between two filesystems. It was written as a replacement for rcp but with many new features. For example it uses an algorithm that will only transfer files that have been modified. SSH will be used to authenticate between the machines and to encrypt the network traffic.

The situation: We have four machines named: server, machine1, machine2, and machine3. The server has a tape drive that is used to do nightly backups. machine1 is used as a development box and has files that need to be backed up in /src and in /home. machine2 is used for mail and needs /home and /mail to be backed up. machine3 is a web server and needs /home, /var/www, and /etc/httpd backed up.

Create a shell script for each machine. Simplify your maintenance by placing the scripts in a central location. I like to use /root/scripts. Decide on where you want to log your output. I like /root/logs but another common option is to have the script mail you the output.

Add entries to your crontab to call the scripts. Make sure you leave enough time before your normal backups of the server that the rsync jobs complete.

Each night the following will occur:

- rsync machine1 -> Server
 - rsync machine2 -> Server
 - rsync machine3 -> Server
 - backup server to tape
- Let's take a look at the flags used for rsync in the examples:

```
rsync -ave ssh --numeric-ids --delete machine1:/home /machine1
```

- -a:
Archive mode
- -v:
Verbose output
- -e ssh:
Specify the remote shell as ssh
- --numeric-ids:
Tells rsync to not map user and group id numbers local user and group names
- --delete:
Makes server copy an exact copy of the source by removing any files that have been removed on the remote machine
- machine1:/home:
The remote machine name, then the directory to be backed up
- /machine1:
The directory to place the backup

Next generate a public private key pair with ssh. Place the public key in the ~/.ssh/authorized_keys file in an account on machine1, machine2, and machine3 that has read access to the directories that need to be backed up. It is best not to use the root account on the remote machines, but you should evaluate the risk in your environment. Test that you can login to these accounts using ssh without using a password.

Test each one of the rsync scripts. The first time you run rsync will take the longest as it will need to copy all the files from the remote machines and not just the files that have changed.

Add the /machine1, /machine2, and /machine3 (or whatever you have named them) directories to the servers backup script.

While this process does not backup the entire remote machine, it will ensure that you will not lose irreplaceable data.

Starting with the example scripts included in this tutorial there are many changes that can be made to fit your specific circumstances.

The frequency of the rsyncs can be modified to occur more often or at different times. Simply by adding additional crontab lines the backup from the remote machines could be done everyday at lunch, multiple times a day or even hourly.

The scripts could also be changed to rotate between multiple backups on the server or could be changed to do some sort of processing on the files before they are backed up. For example if the home directories you are backing up contain web browser caches, they could be removed after the rsync but before the system backup.

Using this article as a starting point you should create a backup plan that fit your needs.

Example rsync script for machine1: `#!/bin/bash`

```
rsync -ave ssh --numeric-ids --delete machine1:/home /machine1
rsync -ave ssh --numeric-ids --delete machine1:/src /machine1
```

Example rsync script for machine2: `#!/bin/bash`

```
rsync -ave ssh --numeric-ids --delete machine2:/home /machine2
rsync -ave ssh --numeric-ids --delete machine2:/mail /machine2
```

Example rsync script for machine3: `#!/bin/bash`

```
rsync -ave ssh --numeric-ids --delete machine3:/home /machine3
rsync -ave ssh --numeric-ids --delete machine3:/var/www /machine3
rsync -ave ssh --numeric-ids --delete machine3:/etc/httpd /machine3
```

Example crontab file logging to a directory: `# Scripts to rsync machines`

```
59 20 * * * /root/scripts/sync-machine1.sh >/root/logs/sync-machine1.log 2>&1
59 21 * * * /root/scripts/sync-machine2.sh >/root/logs/sync-machine2.log 2>&1
59 22 * * * /root/scripts/sync-machine3.sh >/root/logs/sync-machine3.log 2>&1
#
# Nightly Backup script
59 23 * * * /root/scripts/backup.sh > /root/logs/backup.log 2>&1
```

Example crontab file mailing the output: `# Scripts to rsync machines`

```
59 20 * * * /root/scripts/sync-machine1.sh
59 21 * * * /root/scripts/sync-machine2.sh
59 22 * * * /root/scripts/sync-machine3.sh
#
# Nightly Backup script
59 23 * * * /root/scripts/backup.sh
```