Migrating a Linux server from the command line - preparing the servers

Sometimes you need to move everything from one server to another. It can take a while for all those files to transfer, but in this article we try to make the rest of the process as painless as possible.

Server migration

Migrating your data from one Linux server to another is only a simple affair if you've been running a simple server. If you have a lot of interdependent services or a highly customized setup then recreating your environment from scratch is an involved process. It gets less complex if you can copy over just the files you need without worrying about overwriting system files specific to the new server.

So that's what we're going to do here. We'll look at how to prepare for a migration and what tools will make the job go easier.

Full migration versus package migration

The first choice you need to make is whether you want to migrate the whole server, configuration and all, or if you can get away with just copying over the data for a couple services.

In this article we look at the process for a full migration. If you know you want to copy more than just a few datafiles this is the most straightforward approach.

If you prefer a per-package approach you may want to look at the **third article** in this series for advice.

Prepping the new server

Start by confirming that the destination server is accessible via SSH from the origin server. You'll also need to enable root logins via ssh on the destination server (in the /etc/ssh/sshd_config file) so rsync will be able to replace system and application files.

Check that rsync is installed on both the original server and the destination server (the package name is usually "rsync"). Running the command "which rsync" should let you know if it's installed where you can run it.

If you're performing a full migration it's much more likely to go smoothly if the destination server is as similar to the original server as possible. That includes the distribution used, the system architecture, and the kernel version.

Distribution

Make sure you're running the same distribution on each server. Try to match the version of the distribution as well. The location of system files isn't always consistent across different distributions, and sometimes when a distribution releases a new version they move some files around. If you do a straight copy without matching the distribution you may wind up with an unstable server.

If want to combine your server migration with a distribution upgrade it's safer to complete the migration before proceeding with the upgrade.

Architecture

Next make sure both servers are using the same architecture. You can check the architecture on Linux with the "uname -a" command:

```
$ uname -a
Linux demo 2.6.35.4-rscloud #8 SMP Mon Sep 20 15:54:33 UTC 2010 x86_64 Quad-Core AMD Opteron(tm) Processor 2374 HE AuthenticAMD GNU/Linux
```

After the date (which ends in "UTC 2010" above) you'll see a code representing your system's architecture. In this case "x86_64" means it's an x86 system running a 64-bit architecture. If you instead

see i686 for the architecture that means your system is 32-bit.

If the architectures don't match the copied programs won't run. Software compiled for 32-bit will generally not work well on a 64-bit system, and vice-versa. If the architectures don't match you'll have to migrate on a per-package basis instead.

Kernel version

Try to use the same kernel version on both servers. Sometimes a new kernel will add or change features, so a different kernel can throw a monkeywrench into the process.

You can check the kernel version by running "uname -a" like we did above. The kernel version is listed after the hostname, so in the example the kernel version was "2.6.35.4-rscloud".

It's generally not a good idea to copy kernels between servers. If you compile or install your own kernel (as opposed to using one provided by your hosting service) it's safer to perform that process manually on the destination server.

Versions

Finally, try to match the versions of any software that's already installed on the destination to what you're running on the original server. The easiest way to make sure both systems are running the same versions of any common packages is to run an update through your package manager before the migration.

We'll be replacing most software anyway so some version variance shouldn't actually make much difference. We are migrating a server though, and the paramount concern for any server is stability. We want to leave nothing to chance.

Optimizing before copying

The new server generally won't need a lot of the temporary files applications can leave lying around. The more stuff we have on the original server, the longer it will take to get everything onto the destination server.

Much of what goes on behind the scenes when you resize a virtual server is similar to what we'll do when we use rsync to copy from one server to the other. That means a lot of the tips in **our article about speeding up resizes** will apply here.

In a nutshell, remove any temporary or cache files you don't need or add their directories to the exclude file (explained below). Check the sizes of your log files and, if you can, archive or delete older logs.

Summary

We've compared the origin and destination servers to each other and prepared your filesystems for the copy. Now it's time to head to the **next article** in this series to start the servers syncing.

-- Jered

Tags: admin apache api arch at awstats backup capistrano centos cloud courier cron dapper debian dig django dns dstat ebook email fail2ban failover fedora feisty forensics ftp gentoo git gutsy ha hardy heartbeat ibex intrepid iotop iptables jaunty karmic kernel lenny logrotate logs lucid mail maverick migration mod_rails mongrel monitoring munin mysql nagios network nginx ntp open relay passenger permissions php postfix postgresql pv-grub rails rescue rescue mode resize rhel rootkit rsync security setup sftp shorewall slice slice administration slice manager slicemanager spam spamhaus spf ssh ssl subversion tar tcpdump telnet thin tomcat ubuntu untar upgrade vhosts windows winscp