APPENDIX A

List of Linux Commands

This appendix contains lists of the Linux commands that were used in this book, plus a few extra that are frequently important when working with Linux and the CentOS Linux distribution.

Linux commands usually follow the same basic format, based largely after human communication. The command is listed first because that is the name of the program which runs the command. Most commands take one or more *arguments* which can be considered similar to direct objects of a sentence. Additionally, many commands take options (also called flags, modifiers, or parameters) which modify the behavior of the command. These are very similar to adverbs. Options are usually given by prefixing them with a dash. For instance, to change the "list directory" command (`ls`) to show a long-form listing, you add the `-l` option to the command. Thus, the command to show the long-form listing is `ls -l`.

Sometimes options themselves take arguments. However, if they don't, you can oftentimes stack them. For instance, `ls` can also show hidden files with `-a`. You *can* do both options with `ls -l -a`, but you can also squeeze them together with `ls -la`. Many commands have both long options and short options, with the longer version of the option usually preceded by

J. Bartlett, *Building Scalable PHP Web Applications Using the Cloud*,
https://doi.org/10.1007/978-1-4842-5212-3

two dashes. For instance, instead of doing `ls -a`, you could instead type `ls --all`. If you don't know how to invoke a command, most commands have a help screen that you can invoke by doing `COMMAND -h` or `COMMAND --help`.

While each command has its own quirks and abilities, this general command format is common to most commands in Linux.

Additionally, you can get fuller information about a command by looking it up in the manual using the `man` command. For instance, to get the manual page on `ls`, type in `man ls`. Type `q` to exit the manual pages.

Basic Linux Commands

`cat`: This command simply spits out a given file to the screen.

`cd`: This changes your current directory. If the argument starts with a `/`, then it is an *absolute* path starting from the root of the filesystem hierarchy. If the argument starts with a `~`, it is a path that starts from your home directory. Otherwise, it is a *relative* path starting from your current directory. The special directory `..` represents the parent directory, and the special directory `.` represents the current directory. Therefore, to get to the parent directory of your current directory, just do `cd ..`.

`chgrp`: This is like `chown`, but just for group ownership.

`chmod`: This changes the permissions on a file. Each file has separate permissions for the user owner (`u`), the group owner (`g`), and everybody else (`o`). The basic set of permissions is read (`r`), write (`w`), and execute (`x`). To add execute permission to the owner

of a file, you would type `chmod u+x FILENAME`. To add read permission to everybody on a file, you would type `chmod a+r FILENAME`, where `a` refers to all users (`u`, `g`, and `o`). To remove group write permission on a directory and every file/directory under it, you would do `chmod -R g-w FILENAME`.

`chown`: This changes the user and group ownership of a file. If you do `chown fred FILENAME`, then `fred` becomes the owner of the file. If you do `chown fred:stonemasons FILENAME`, then `fred` becomes the owner and the group `stonemasons` becomes the group owner of the file. You can add the `-R` option to change the ownership of a directory and all of the file/directories underneath it.

`head`: This command gives you the first few lines of a file. `head -n NUM FILENAME` will give you the first `NUM` lines of the `FILENAME` file.

`ls`: This command lists all of the files in your current directory. Using the `-a` switch will show all files, including special and hidden files (files starting with a `.`), and the `-l` switch will show additional information about the files. You can also specify a specific directory to list at the end of the command.

`man`: This command gives the *man*ual page for a given command or configuration file. `man ls` tells you about the `ls` command. To get out of the manual, just type `q`.

`mkdir`: This creates a new directory within your current directory (or anywhere at all if you give it an absolute path starting with `/` or `~`).

`nano`: This is a simple text editor that comes with many Linux distributions. Common commands to use within `nano` are control-o which saves (outputs) the file, control-x which quits, and control-w which searches.

`pwd`: This prints out your current directory. It stands for "print working directory."

`rm`: Removes a given file. If you want to remove a whole directory tree (the directory and all of the files/directories in it), add the `-r` switch. Just be careful!

`scp`: This command (known as "secure copy") remote copies a file from one machine to another over an encrypted channel. The basic format is `scp LOCALFILE USERNAME@REMOTE.MACHINE.NAME:/PATH/TO/DESTINATION`.

`ssh`: This command (known as "secure shell") allows you to remote access other machines over an encrypted channel. The general format is `ssh` USERNAME@MACHINE.HOST.NAME.

`tail`: This command gives you the last few lines of a file. `tail -n NUM FILENAME` gives you the last `NUM` lines of the file `FILENAME`.

`telnet`: Telnet used to be the way that you accessed machines remotely, before encryption became a necessity. Now it is often used to make direct connections with remote servers for testing. For instance, to talk to a web server directly, you can type in `telnet REMOTE.MACHINE.NAME 80`, and it

will connect you directly to port 80 on the remote machine. Remember that `telnet` will display information directly to your screen, so be careful if sensitive data may be returned.

`vim`: This application (or its older brother, `vi`) is an editor that you will find on nearly *any* Linux-like system. It is very powerful, but please read a tutorial on it before attempting to use it. Its primary benefits are that its keyboard interface is based on where your hands naturally sit on the keyboard and that its small footprint and longstanding heritage mean that you will never be on a Linux or UNIX machine that doesn't have it installed.

Basic System Administration

`passwd`: This command changes the password of users. Without an argument, this changes your own password. Otherwise, it changes the password of the user you specify.

`useradd`: This creates a new user on the system. `useradd fred` adds a new user with the name `fred`. Use the `passwd` command to set their password.

`usermod`: This modifies a user on the system, normally to add them to a group. Use `usermod -a -G thegroup theusername` to add the user `theusername` to the group `thegroup`.

`systemctl`: This command handles starting and stopping background services on Linux. This command usually has the form `systemctl CMD`

SERVICE where CMD is the command you want to give, and SERVICE is the service you want to issue the command to. Service commands include start, stop, restart, enable (make sure the service starts on bootup), and disable (make sure the service does not start on bootup). The main services covered in this book include httpd, postgresql, and memcached.

firewall-cmd: This command handles the firewall. This command has several options. The option --add-service SERVICENAME allows you to open up a service to outside connections, where SERVICENAME is the specific service you want to allow outside users to connect to. The list of services is available by running firewall-cmd --get-service. If you just want to open up a port (i.e., one for which there is no service description), you can just use firewall-cmd --add-port 1234/tcp in order to open up TCP port 1234. To make the service available on reboot, you need to re-issue the command with the --permanent flag added. You can show everything enabled on the current firewall by using --list-all.

yum: The yum command is the automated package installer for CentOS and other Linux distributions. yum allows you to quickly and easily search, find, and install Linux packages onto your system. yum focuses on finding packages on the Internet and resolving dependencies between packages, and then uses rpm to do all of the dirty work of actually installing the packages. yum includes several subcommands, such

as `search`, `install`, `update`, and `uninstall`. `yum update` updates all of your installed packages, `yum search TERM` gives you a list of all available packages whose description includes the word `TERM`, and `yum install PACKAGENAME` will install `PACKAGENAME` and all of its dependencies for you. If for some reason `yum` stops working correctly, usually you can fix it by running `yum clean all`.

`rpm`: The `rpm` command is the low-level package manager for CentOS. It handles the work of actually taking a package file and installing it onto the system. This is pretty rare, as this is usually handled through `yum`. However, `rpm` also has a way of querying installed packages. A list of all of your installed packages can be found by running `rpm -qa`, and a list of all files that were modified after installation can be found by running `rpm -Va`.

`rkhunter`: This command, if installed, checks your system for various types of malware by running `rkhunter --check`.

`su`: This command stands for "switch user." Without any arguments, this switches the user to the `root` user. You usually want to add the `-l` option, which means to act as if you logged in with this user, which will take you to the user's home directory and run other login tasks. If you give it an argument, it will be the name of the user you want to switch to. You must enter that user's password in order to switch users.

`sudo`: This command lets you temporarily run a command as another user (normally as `root`). The configuration of this command is beyond the scope of this book, but `man sudo` should give you good information.

`pssh`: This command performs a parallel `ssh` session across multiple hosts. See Chapter 12 for more information about this command.

`pscp`: This command performs a parallel `scp` copy from a local file or directory to multiple destination hosts. See Chapter 12 for more information about this command.

`prsync`: This command does a parallel synchronization between a local directory and multiple destination hosts. See Chapter 12 for more information about this command.

`ss`: This command gives information about open sockets on your machine. The two commands we focus on are `ss -plnt` for looking at listening TCP connections and `ss -plnu` for looking at listening UDP connections. This command is critical for knowing potential attack vectors that an attacker may use to gain access to your system.

`netstat`: This is an older version of the `ss` command. This command gives you lots of information about active network connections on the system. The two ways this is normally called are `netstat -plant` (which gives a list of TCP session and listening sockets) and `netstat -planu` (which gives a similar list for UDP).

`ps`: This command gives you information about processes running on the system. This has numerous options that can give you almost any piece of information you want to know. However, my favorite way of calling it is `ps -afxww` which gives you a list of all of the processes currently running displayed as a tree so you know which process spawned which other process.

`top`: This command gives you information about which processes are using the most system resources. Use `q` to leave `top`.

`free`: This gives a short rundown of the current memory usage on the system. `free -h` gives the most readable output.

`uptime`: This gives a short rundown of the current load on the system. In Linux, the load is the number of processes that are wanting CPU time at any given moment. Therefore, for a machine with *x* processors, the machine is fully loaded near *x* and is falling behind when it goes above that number. I usually try to keep my machines only half loaded at most.

PostgreSQL Commands

`createdb`: This command creates a new database in PostgreSQL, usually called with the parameter `-U PGUSER`, where `PGUSER` is the PostgreSQL user who will create (and therefore own) this database.

`createuser`: This command creates new users in PostgreSQL. This takes the `-U PGUSER` parameter to tell which user to run as. The `-d` flag will give the new user permission to create new databases, and the `-P` parameter will prompt you to type in a password for the new user.

`pg_basebackup`: This command creates a binary backup of PostgreSQL. In this book, we use this command as a starting point for replication.

`postgresql-setup`: This is a script that aids the installation of PostgreSQL instances. In this book, we just use `postgresql-setup initdb` to create the initial instance data for PostgreSQL.

`psql`: This command gives you access to the PostgreSQL interactive SQL prompt. The `-U PGUSER` indicates which PostgreSQL user you will access the database as, and the command argument will be which database to connect to. The `-h HOSTNAME` option can allow you to access a PostgreSQL database on a different host.

Other Application-Specific Commands

`ab`: The ApacheBench command simulates a large number of requests to a web site and gathers response statistics. It is normally called `ab -n NUMREQUESTS -c NUMCONNS FULL_URL`, where `NUMREQUESTS` are the total number of requests for ApacheBench to make, `NUMCONNS` is the number of simultaneous connections to keep going, and `FULL_URL` is the destination URL you are trying to test.

`convert`: This command is a part of the ImageMagick package, and is a Swiss army knife for converting and modifying image files. To convert a JPEG file called `testme.jpg` to a PNG file, just do `convert testme.jpg testme.png`.

`pecl`: This is a package manager for PHP. This allows you to install add-ons to the system's PHP environment that aren't available via yum.

APPENDIX B

Important Files and Directories

This book mentions a number of important files and directories on CentOS. This appendix lists the files and directories from the book that you need to remember, plus a few more. Remember, each distribution has its own special way of doing things, so a file's location on CentOS may be slightly different than its location on Ubuntu. Additionally, each cloud provider may also set things up in certain ways. Custom-installed applications sometimes can wind up just about anywhere, depending on the application author, the person who packaged it, or the user who installed it. The Remi packages used in this book are a case in point—the `/opt/remi` hierarchy is entirely an invention of the packager.

Basic Linux Filesystem Directories

The first part of knowing where to find things is to know where Linux likes to put things. This section is a brief introduction to the standard structure of the Linux filesystem. Note that on Linux hard drives do not exist separately, but are instead "mounted" at certain locations on the

J. Bartlett, *Building Scalable PHP Web Applications Using the Cloud*,
https://doi.org/10.1007/978-1-4842-5212-3

filesystem. In other words, there is a single filesystem hierarchy even with multiple hard drives. The drives simply represent specific folders within the hierarchy.

`/`: This is the root of the filesystem tree. Every file and directory is contained somewhere in here.

`/boot`: This directory holds basic files for booting up (like the Linux kernel). You should usually stay out of this directory.

`/dev`: In Linux, devices are represented as files, and they live here.

`/etc`: This directory contains most of the configuration files for the computer.

`/proc`: This directory contains a file for every running process on the computer, plus files to represent operating system status information.

`/tmp`: This directory is used to store temporary files.

`/var`: This directory holds *var*iable files—files that are tied to programs and are intended to change during the operation of a program. For instance, your database is stored in a subdirectory of `/var` because it is changing and it is managed by the database software instead of by the user.

`/var/log`: This directory holds most of the log files for the system.

`/var/spool`: This directory is mostly used for transient data within a system, such as current mail for a mail server, jobs for a print service, and so on.

`/usr`: This directory is a mostly read-only directory used to store programs used by the machine in the course of its operation.

`/usr/bin`: The files in this directory are the programs (i.e., *bin*aries) that are normally used on the server by users and other programs. Also note that there is a directory `/bin` which has the programs that are necessary for proper bootup.

`/usr/sbin`: The files in this directory are the system programs (e.g., daemons and system tools) that are available for use on this machine. Also note that there is a directory `/sbin` which has the system programs that are necessary for proper bootup.

`/usr/lib`: The files that live here are the system *lib*raries that support the applications. There is often a `/usr/lib64` directory for 64-bit libraries. Also note that there is a directory `/lib` which has the libraries that are necessary for proper bootup.

`/usr/local`: This directory has almost an identical structure to `/usr`, but the programs installed here are usually compiled by the system administrator. Unlike `/opt` (where each program gets an entire directory to themselves), the programs installed here all share the same `bin`, `lib`, and `sbin` directories.

`/home`: This directory holds the home directory of each user on the machine except `root`.

`/root`: This is `root`'s home directory.

`/opt`: This is a directory that often has custom-installed applications.

In `/opt`, each program usually has its own directory.

Important Directories for Cloud Servers

These directories are special-purpose directories that you will want to know about for operating a cloud server. Different Linux distributions may put these in different locations, but these are the default CentOS locations.

`/etc/httpd`: This is the directory that holds the configuration files for the Apache web server.

`/etc/postfix`: This is the mail server's configuration directory.

`/etc/sysconfig`: This directory holds additional configuration of many system services.

`/etc/systemd`: This directory contains the configuration information that managed the `systemctl` command.

`/etc/ssh`: This directory contains the configuration of both the `ssh` client and server.

`/var/lib/pgsql/data`: This is the directory that holds PostgreSQL's database files.

`/var/lib/pgsql/data/pg_log/`: This is the directory that holds PostgreSQL's log information.

`/var/log/httpd`: This is the directory that holds Apache's server logs.

`/var/www/html`: This is the default directory for holding a web site.

`/opt/remi/php74/root`: This is the directory that holds the PHP 7.4 installation from the Remi repository that is used in this book.

Important Files

This book has covered many different files that have to be configured for properly running a server.

`/var/lib/pgsql/data/pg_hba.conf`: This file configures access controls for PostgreSQL.

`/var/lib/pgsql/data/postgresql.conf`: This is the main configuration file for PostgreSQL.

`/var/lib/pgsql/data/recovery.conf`: This file controls a PostgreSQL instance running as a replica server.

`/var/log/httpd/access_log`: This is the default location that logs every time your web server is accessed.

`/var/log/httpd/error_log`: This is the default location for errors from Apache and PHP.

`/var/log/maillog`: This is the log of all mail messages sent out from the system.

`/var/log/messages`: This is the default location for system error messages.

APPENDIX C

What to Do When It Doesn't Work

There are all sorts of things that can go wrong when following code from a book. This appendix focuses on the most common issues and what you can do about them.

Making Sure Everything Is Typed in Correctly

In any programming book, the first thing to check when something doesn't go right is to make sure it is typed in correctly.

- Be sure that everything is typed correctly.
- Make certain that everything is on the line that it is supposed to be (sometimes this matters, sometimes it doesn't—better safe than sorry).
- Check to see that everything uses the right punctuation (i.e., that you used colons and semicolons properly and used single and double quotes correctly).

J. Bartlett, *Building Scalable PHP Web Applications Using the Cloud*,
https://doi.org/10.1007/978-1-4842-5212-3

- Verify that your computer didn't autocorrect your punctuation into something more pretty. If you type in `"hello"` and your computer spits out "`hello`", then that will not work. Turn off automatic punctuation or whatever it is.
- If you are writing the PHP files locally, be sure that you are using a text editor, not a word process, and that the files are being saved text-only, using either an ASCII or UTF-8 character set.

Making Sure You Checked the Logs

If you think that you did everything right, and it still isn't working out for you, *check the logs*. This is the easiest way to spot a mistake. For PHP programs, the log files to check are `/var/log/httpd/error_log` and `/var/opt/remi/php74/log/php-fpm/www-error.log`. For PostgreSQL startup errors, the log files are in the directory `/var/lib/pgsql/data/pg_log`. Finally, for general system messages, check `/var/log/messages`.

Many log files are very long, and you only need to see the last few lines. The `tail` command will help you by just giving you the last few lines of a file. `tail /var/log/messages` or `tail /etc/httpd/logs/error_log` will potentially give you a lot of information about any problems you are having on your system.

Making Sure You Didn't Miss a Step

The instructions in this book contain a number of steps, and care was taken to be sure that they all worked when done in order. Therefore, be sure that you follow the steps in order when you are first learning. After you get the first thing up and running, *then* you should take the time and initiative to branch out and try variations.

Additionally, this book has a lot of things for you to install, and it is possible you missed one. Just in case, you might go each chapter and make sure you installed everything.

What If I Run a Different Version/ Distribution of Linux

Of course, there are a ton of different Linux distributions, and the specifics even within a distribution change for each version. If you are running a different version of Linux, be aware that the commands might be slightly different, that there may be slightly different ways of doing things, different things might come standard, and directories might be located in different places.

This book was written around the latest CentOS distribution at the time, so it will hopefully be perfectly suitable for several years. Nonetheless, the basic ideas will still work even if you need to tweak the directions slightly for the Linux distribution you are working with.

What If I Want to Use a Different Cloud Service?

While this book focuses on Linode, AWS, and GCP, there is very little in this book that doesn't directly translate to other services. I am a huge fan of Linode because they have a service that is simple yet powerful, and their servers are top quality. Most cloud services, however, are structured in essentially the same way. Therefore, if you are using a different service for some reason, the basic ideas in this book should still continue to hold. The point of this book is to get your mind thinking about cloud *architecture*—using a specific service merely helps you get started with a concrete starting point.

Where Else Can I Find Information?

Information abounds on the Web about troubleshooting programs that have gone awry. The first try for anything that happens that I don't understand is to Google the error message. If that doesn't work (or if you don't even have an error message to start from), the next step is probably a message board. The biggest message board is Stack Exchange, but there are other good ones as well.

Each of the technologies that this book deals with has excellent reference manuals. Most of these are really good, though it sometimes takes a while to find the specific solution that you need. The PHP reference manual even has an interactive section, where developers can ask questions and get answers.

Finally, you can go to a local developer meetup. Nearly every city has one. Even if it isn't PHP specific, if you find a developer group, chances are one of them will know enough PHP, Apache, or Linux to help you out.

Don't be stuck by yourself—get help and improve your skillset!

Afterword

Now that you have taken our small guestbook app through a variety of cloud configurations, you should be able to apply these principles to scaling any other application in the cloud.

A few tips to keep in mind as you go forward are

- Always look for ways to restructure your application so as to prevent a bottleneck of any one location or action.
- Keep an eye out for parts of your application that can be easily replicated to infinite scale through CDNs or similar services.
- Measure the scalability of different configurations of your application to find out where your problems are and where you quickly max out your system.

If you have followed through the exercises in this book, you can consider yourself experienced at scaling web applications in the cloud!

J. Bartlett, *Building Scalable PHP Web Applications Using the Cloud*,
https://doi.org/10.1007/978-1-4842-5212-3

Index

J. Bartlett, *Building Scalable PHP Web Applications Using the Cloud*,
https://doi.org/10.1007/978-1-4842-5212-3

D

E

F

G

H

I, J, K

L

M

N, O

P, Q

R

S

T, U

V

W, X

Y, Z