

EXPERT INSIGHT

The Software Developer's Guide to Linux

A practical, no-nonsense guide to using the Linux command line and utilities as a software developer



David Cohen
Christian Sturm

<packt>

The Software Developer's Guide to Linux

A practical, no-nonsense guide to using the Linux command line and utilities as a software developer

David Cohen

Christian Sturm



BIRMINGHAM—MUMBAI

The Software Developer's Guide to Linux

Copyright © 2024 Packt Publishing

All rights reserved. No part of this book may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior written permission of the publisher, except in the case of brief quotations embedded in critical articles or reviews.

Every effort has been made in the preparation of this book to ensure the accuracy of the information presented. However, the information contained in this book is sold without warranty, either express or implied. Neither the authors, nor Packt Publishing or its dealers and distributors, will be held liable for any damages caused or alleged to have been caused directly or indirectly by this book.

Packt Publishing has endeavored to provide trademark information about all of the companies and products mentioned in this book by the appropriate use of capitals. However, Packt Publishing cannot guarantee the accuracy of this information.

Senior Publishing Product Manager: Aaron Tanna

Acquisition Editor – Peer Reviews: Gaurav Gavas

Project Editor: Parvathy Nair

Content Development Editor: Matthew Davies

Copy Editor: Safis Editing

Technical Editor: Karan Sonawane

Proofreader: Safis Editing

Indexer: Tejal Soni

Presentation Designer: Ganesh Bhadwalkar

Developer Relations Marketing Executive: Meghal Patel

First published: January 2024

Production reference: 1230124

Published by Packt Publishing Ltd.

Grosvenor House

11 St Paul's Square

Birmingham

B3 1RB, UK.

ISBN 978-1-80461-692-5

www.packt.com

Contributors

About the authors

David Cohen has, for the past 15 years, worked as a Linux system administrator, software engineer, infrastructure engineer, platform engineer, site reliability engineer, security engineer, web developer, and a few other things besides. In his free time, he runs the *tutoriallinux* YouTube channel where he's taught hundreds of thousands of people the basics of Linux, programming, and DevOps. David has been at Hashicorp since 2019—first as an SRE, then as a reference architect, and now as a software engineer.

Thank you, Aleya, for your unwavering support over the past few years as I've been developing and writing this book. Without you, this would just be another of my promising-but-unfinished projects languishing in some forgotten "Archive" directory. Thanks to Christian, who has stuck with me for over a decade as a friend and a partner on practically every wild tech project idea I've come up with since we met. Finally, a big "thank you" is also due to my friends and colleagues at Hashicorp and everywhere else I've been over the past 15 years, who have made me a better engineer and encouraged projects like this.

Christian Sturm is a consultant on software and systems architecture, having worked in various technical positions for well over a decade. He has worked as an application developer for the frontend and backend at companies large and small, such as zoomsquare and Plutonium Labs. On top of that, he is also an active contributor to various open source projects and has a deep understanding of fields including operating systems, networking protocols, security, and database management systems.

About the reviewers

Mario Splivalo works as a consultant dealing with databases extended into modern cloud-based architectures. He also helps companies design their infrastructure using IaaS tools such as *Terraform* and *AWS Cloudformation*. For five years, Mario worked with *Canonical* as an OpenStack engineer.

Mario's fascination with computers started back when Commodore 64 dominated the user space. He took his first steps using BASIC on his dad's C64, quickly shifting to Assembler. He gradually moved to PCs, finding a great love for programming, systems design, and database administration. He switched to Linux (Knoppix, then Ubuntu, and never looked back) in the early 2000s, continuing as a database administrator, programmer, and system administrator.

Nathan Chancellor is an independent contractor working on the Linux kernel, based in Arizona, US. As a developer, his focus is on improving the compatibility between the Linux kernel and the LLVM toolchain. He has used Linux since 2016 and it has been his primary development operating system since 2018. His distributions of choice are Arch Linux and Fedora.

Learn more on Discord

To join the Discord community for this book – where you can share feedback, ask questions to the author, and learn about new releases – follow the QR code below:

<https://packt.link/SecNet>

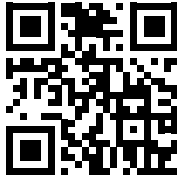


Table of Contents

Preface	xxiii
---------	-------

Chapter 1: How the Command Line Works	1
---------------------------------------	---

In the beginning...was the REPL	1
---------------------------------------	---

Command-line syntax (read)	3
----------------------------------	---

Command line vs. shell	4
------------------------------	---

How does the shell know what to run? (evaluate) • 5

A quick definition of POSIX • 6

Basic command-line skills	7
---------------------------------	---

Unix filesystem basics • 7

Absolute vs. relative file paths • 8

Absolute vs. relative pathname review • 10

Opening a terminal • 10

Looking around – command-line navigation • 10

pwd - print working directory • 10

ls - list • 11

Moving around • 12

cd – change directory • 12

find – find files • 13

Reading files • 13

less – page through a file • 14

Making changes • 14

<i>touch</i> – create an empty file, or update modification time for an existing one • 14	
<i>mkdir</i> – create a directory • 14	
<i>rmdir</i> – remove empty directories • 15	
<i>rm</i> – remove files and directories • 15	
<i>mv</i> – move or rename files and directories • 16	
Getting help	17
Shell autocompletion	18
Conclusion	20
 Chapter 2: Working with Processes	 21
Process basics	22
What is a Linux process made of? • 23	
Process ID (PID) • 24	
Effective User ID (EUID) and Effective Group ID (EGID) • 25	
Environment variables • 25	
Working directory • 25	
Practical commands for working with Linux processes	26
Advanced process concepts and tools	28
Signals • 28	
<i>Practical uses of signals</i> • 29	
<i>Trapping</i> • 29	
<i>The kill command</i> • 29	
lsof – show file handles that a process has open • 31	
Inheritance • 33	
Review – example troubleshooting session	33
Conclusion	35
 Chapter 3: Service Management with systemd	 37
The basics	38
init • 39	
Processes and services	39

systemctl commands	40
Checking the status of a service • 40	
Starting a service • 41	
Stopping a service • 41	
Restarting a service • 41	
Reloading a service • 42	
Enable and disable • 42	
A note on Docker	43
Conclusion	43
 Chapter 4: Using Shell History	 45
Shell history	45
Shell configuration files • 46	
History files • 46	
Searching through shell history • 47	
Exceptions • 47	
Executing previous commands with !	48
Re-running a command with the same arguments • 48	
Prepending a command to something in your history • 48	
Jumping to the beginning or end of the current line	49
Conclusion	49
 Chapter 5: Introducing Files	 51
Files on Linux: the absolute basics	52
Plaintext files • 52	
What is a binary file? • 52	
Line endings • 53	
The filesystem tree	53
Basic filesystem operations	54
ls • 54	
pwd • 55	

cd • 55	
touch • 56	
less • 57	
tail • 57	
mv • 57	
<i>Moving</i> • 57	
<i>Renaming</i> • 58	
cp • 58	
mkdir • 58	
rm • 58	
Editing files	59
File types	60
Symbolic links • 61	
Hard links • 62	
The file command • 62	
Advanced file operations	63
Searching file content with grep • 63	
Finding files with find • 64	
<i>Copying files between local and remote hosts with rsync</i> • 65	
<i>Combining find, grep, and rsync</i> • 66	
Advanced filesystem knowledge for the real world	67
FUSE: Even more fun with Unix filesystems • 68	
Conclusion	69
 Chapter 6: Editing Files on the Command Line	 71
Nano	72
Installing nano • 72	
Nano cheat sheet • 72	
<i>File handling</i> • 73	
<i>Editing</i> • 73	
<i>Search and replace</i> • 73	

Vi(m)	73
Vi/vim commands • 74	
Modes • 74	
<i>Command mode</i> • 74	
<i>Normal mode</i> • 75	
Tips for learning vi(m) • 76	
<i>Use vimtutor</i> • 76	
<i>Think in terms of mnemonics</i> • 76	
<i>Avoid using arrow keys</i> • 76	
<i>Avoid using the mouse</i> • 77	
<i>Don't use gvim</i> • 77	
<i>Avoid starting with extensive configuration or plugins</i> • 77	
Vim bindings in other software • 79	
Editing a file you don't have permissions for	79
Setting your preferred editor	79
Conclusion	80
 Chapter 7: Users and Groups	 81
What is a user?	81
Root versus everybody else	82
sudo	82
What is a group?	84
Mini project: user and group management	84
Creating a user • 85	
Create a group • 86	
Modifying a Linux user • 86	
<i>Adding a Linux user to a group</i> • 87	
<i>Removing a user from a group</i> • 87	
<i>Removing a Linux user</i> • 87	
<i>Remove a Linux group</i> • 87	
Advanced: what is a user, really?	88
User metadata / attributes • 88	

A note on scriptability	90
Conclusion	90
Chapter 8: Ownership and Permissions	93

Deciphering a long listing	93
File attributes	94
File type • 94	
Permissions • 95	
Number of hardlinks • 95	
User ownership • 95	
Group ownership • 95	
File size • 96	
Modification time • 96	
Filename • 96	
Ownership	96
Permissions	97
Numeric/octal • 98	
Common permissions • 99	
Changing ownership (chown) and permissions (chmod)	99
Chown • 99	
<i>Change owner • 99</i>	
<i>Change owner and group • 100</i>	
<i>Recursively change owner and group • 100</i>	
Chmod • 100	
<i>Using a reference • 101</i>	
Conclusion • 101	
Chapter 9: Managing Installed Software	103

Working with software packages	104
Update your local cache of repository state • 105	
Search for a package • 105	

Install a package • 106	
Upgrade all packages that have available updates • 106	
Remove a package (and any dependencies, provided other packages don't need them) • 106	
Query installed packages • 107	
Caution required – curl bash	107
Compiling third-party software from source	108
Example: compiling and installing htop • 110	
<i>Install prerequisites • 110</i>	
<i>Download, verify, and unarchive the source code • 110</i>	
<i>Configure and compile htop • 111</i>	
Conclusion	112
 Chapter 10: Configuring Software	 115
Configuration hierarchy	116
Command-line arguments	118
Environment variables	119
Configuration files	121
System-level configuration in /etc/ • 121	
User-level configuration in ~/.config • 121	
systemd units	122
Create your own service • 122	
Quick note: configuration in Docker	124
Conclusion	125
 Chapter 11: Pipes and Redirection	 127
File descriptors	128
What do these file descriptors reference? • 129	
Input and output redirection (or, playing with file descriptors for fun and profit)	129
Input redirection: < • 129	
Output redirection: > • 130	
<i>Use >> to append output without overwriting • 131</i>	

Error redirection with 2> • 132	
Connecting commands together with pipes ()	133
Multi-pipe commands • 133	
<i>Reading (and building) complex multi-pipe commands • 134</i>	
The CLI tools you need to know	134
cut • 134	
sort • 135	
uniq • 136	
<i>Counting • 136</i>	
wc • 137	
head • 138	
tail • 138	
tee • 138	
awk • 139	
sed • 139	
Practical pipe patterns	140
“Top X”, with count • 140	
curl bash • 140	
<i>Security considerations for curl sudo bash • 141</i>	
Filtering and searching with grep • 142	
grep and tail for log monitoring • 143	
find and xargs for bulk file operations • 143	
sort, uniq, and reverse numerical sort for data analysis • 144	
awk and sort for reformatting data and field-based processing • 145	
sed and tee for editing and backup • 145	
<i>ps, grep, awk, xargs, and kill for process management • 146</i>	
<i>tar and gzip for backup and compression • 146</i>	
Advanced: inspecting file descriptors	147
Conclusion	149

Chapter 12: Automating Tasks with Shell Scripts	151
Why you need Bash scripting basics	152
Basics	152
Variables • 152	
<i>Setting</i> • 152	
Getting • 153	
Bash versus other shells	153
Shebangs and executable text files, aka scripts	154
Common Bash settings (options/arguments) • 154	
/usr/bin/env • 155	
Special characters and escaping • 156	
Command substitution • 157	
Testing	157
Testing operators • 157	
[[file and string testing]] • 158	
<i>Useful operators for string testing</i> • 158	
<i>Useful operators for file testing</i> • 158	
((arithmetic testing)) • 159	
Conditionals: if/then/else	160
<i>ifelse</i> • 160	
Loops	160
C-style loops • 160	
for...in • 161	
While • 161	
Variable exporting • 162	
Functions	162
Prefer local variables • 163	

Input and output redirection	164
<: input redirection • 164	
> and >>: output redirection • 164	
Use 2>&1 to redirect STDERR and STDOUT • 164	
Variable interpolation syntax – \${}	165
Limitations of shell scripts	165
Conclusion	166
Citations	166
 Chapter 13: Secure Remote Access with SSH	 167
Public key cryptography primer	168
Message encryption • 168	
Message signing • 169	
SSH keys	169
Exceptions to these rules • 170	
Logging in and authenticating • 171	
Practical project: Set up a key-based login to a remote server	171
Step 1: Open your terminal on the SSH client (not the server) • 171	
Step 2: Generate the key pair • 172	
Step 3: Copy the public key to your server • 172	
Step 4: Test it out! • 172	
Converting SSH2 keys to the OpenSSH format	173
What we are trying to achieve • 173	
How to convert the SSH2-formatted key to OpenSSH • 174	
The other direction: Converting SSH2 keys to the OpenSSH format • 174	
SSH-agent	175
Common SSH errors and the -v (verbose) argument	176
File transfer	177
SFTP • 178	
SCP • 178	
Clever examples • 179	

<i>Without SFTP or SCP • 180</i>	
<i>Directory upload and .tar.gz compression • 180</i>	
Tunnels	181
Local forwarding • 181	
Proxying • 181	
The configuration file	182
Conclusion	183
 Chapter 14: Version Control with Git	 185
Some background on Git	186
What is a distributed version control system?	186
Git basics	187
First-time setup • 187	
Initialize a new Git repository • 187	
Make and see changes • 187	
Stage and commit changes • 187	
Optional: add a remote Git repository • 188	
Pushing and pulling • 188	
Cloning a repository • 188	
Terms you might come across	189
Repository • 189	
<i>Bare repository • 189</i>	
Branch • 189	
<i>Main/master branch • 190</i>	
<i>HEAD • 190</i>	
Tag • 190	
<i>Shallow • 190</i>	
Merging • 190	
<i>Merge commit • 191</i>	
Merge conflict • 191	
Stash • 191	

Pull request • 191	
Cherry-picking • 192	
Bisecting • 192	
Rebasing • 193	
Best practices for commit messages	196
Good commit messages • 196	
GUIs	197
Useful shell aliases	197
Poor man's GitHub	198
Considerations • 198	
1. Connect to your server • 198	
2. Install Git • 198	
3. Initialize a repository • 199	
4. Clone the repository • 199	
5. Edit the project and push your changes • 200	
Conclusion	200
 Chapter 15: Containerizing Applications with Docker	 203
How containers work as packages	204
Prerequisite: Docker install	205
Docker crash course	205
Creating images with a Dockerfile	208
Container commands	210
docker run • 210	
<i>docker image list • 211</i>	
docker ps • 211	
docker exec • 212	
docker stop • 212	
Docker project: Python/Flask application container	212
1. Set up the application • 213	
2. Create the Docker image • 215	

3. Start a container from your image • 215	
Containers vs. virtual machines	216
A quick note on Docker image repositories	217
Painfully learned container lessons	217
Image size • 218	
C standard library • 218	
Production is not your laptop: outside dependencies • 218	
Container theory: namespaces	219
How do we do Ops with containers?	220
Conclusion	220
 Chapter 16: Monitoring Application Logs	 223
Introduction to logging	224
Logging on Linux can get... weird • 224	
Sending log messages • 225	
The systemd journal	225
Example journalctl commands	226
Following active logs for a unit • 226	
Filtering by time • 226	
Filtering for a specific log level • 227	
Inspecting logs from a previous boot • 227	
Kernel messages • 227	
Logging in Docker containers	228
Syslog basics	228
Facilities • 229	
Severity levels • 230	
Configuration and implementations • 230	
Tips for logging	230
Keywords when using structured logging • 230	
Severity • 231	

Centralized logging	231
Conclusion	233
Chapter 17: Load Balancing and HTTP	235
Basic terminology	236
Gateway • 236	
Upstream • 237	
Common misunderstandings about HTTP	237
HTTP statuses • 237	
<i>Don't just check for 200 OK • 237</i>	
<i>404 Not Found • 238</i>	
<i>502 Bad Gateway • 238</i>	
<i>503 Service Unavailable • 239</i>	
<i>504 Gateway Timeout • 239</i>	
<i>Introduction to curl: checking HTTP response status • 239</i>	
HTTP headers • 240	
<i>Case-insensitive headers • 240</i>	
<i>Custom headers • 240</i>	
<i>Viewing HTTP headers with curl • 241</i>	
HTTP versions • 241	
<i>HTTP/0.9 • 241</i>	
<i>HTTP/1.0 and HTTP/1.1 • 242</i>	
<i>HTTP/2 • 242</i>	
<i>HTTP/3 and QUIC • 243</i>	
Load balancing • 244	
<i>Sticky sessions, cookies, and deterministic hashing • 245</i>	
<i>Round-robin load balancing • 246</i>	
<i>Other mechanisms • 246</i>	
<i>High availability • 246</i>	
<i>Troubleshooting redirects with curl • 247</i>	
<i>Using curl as an API testing tool • 248</i>	

<i>Accepting and displaying bad TLS certificates with curl</i> • 249	
CORS • 249	
Conclusion	251
Other Books You May Enjoy	255

Index	259
--------------	------------

Preface

Many software engineers are new to Unix-like systems, even though these systems are everywhere in the software engineering world. Whether developers know it or not, they're expected to work with Unix-like systems running in their work environment (macOS), their software development process (Docker containers), their build and automation tooling (CI and GitHub), their production environments (Linux servers and containers), and more.

Being skilled with the Linux command line can help software developers go beyond what's expected of them, allowing them to:

- Save time by knowing when to use built-in Unix tools, instead of writing thousand-line scripts or helper programs
- Help debug complex production outages, often involving Linux servers and their interface to the application
- Mentor junior engineers
- Have a more complete understanding of how the software they write fits into the larger ecosystem and tech stack

We hope that the theory, examples, and projects included in this book can take your Linux development skills to the next level.

Who this book is for

This book is for software developers who are new to Linux and the command line, or who are out of practice and want to quickly dust off their skills. If you still feel a bit insecure about your abilities when you're staring at a Linux command-line prompt on a production server at 2:00 in the morning, this book is for you. If you want to quickly fill a Linux skills gap to advance your career, this book is for you. If you're just curious, and you want to see what kind of efficiency gains you can make in your day-to-day development setup and routines by adding some command-line magic, this book will serve you as well.

What this book is not

One of the ways we have tried to fulfill our vision for this kind of uniquely useful book is by being extremely careful about what's included. We've tried to cut out everything that isn't essential to your life as a developer, or to a basic understanding of Linux and its core abstractions. In other words, the reason this book is useful is because of *all the things we left out*.

This book is not a full Linux course. It's not for people working as Linux system engineers or kernel developers. Because of this, it's not 750+ pages long, and you should be able to work through it in a few days, perhaps during a quiet sprint at work.

What this book covers

Chapter 1, How the Command Line Works, explains how a command-line interface works, what a shell is, and then immediately gives you some basic Linux skills. You'll get a bit of theory and then begin moving around on the command line, finding and working with files and learning where to look for help when you get stuck. This chapter caters to new developers by teaching the most important command-line skills. If you read nothing else, you'll still be better off than when you started.

Chapter 2, Working with Processes, will take you on a guided tour of Linux processes. You'll then dive into useful, practical command-line skills for working with processes. We'll add detail to a few aspects that are a common source of process-related problems that you'll encounter as a software developer, like permissions, and give you some heuristics for troubleshooting them. You'll also get a quick tour of some advanced topics that will come up again later in the book.

Chapter 3, Service Management with systemd, builds on the knowledge about processes learned in the previous chapter by introducing an additional layer of abstraction, the `systemd` service. You'll learn about what an `init` system does for an operating system, and why you should care. Then, we cover all the practical commands you'll need for working with services on a Linux system.

Chapter 4, Using Shell History, is a short chapter covering some tricks that you can learn to improve your speed and efficiency on the command line. These tricks revolve around using shortcuts and leveraging shell history to avoid repeated keystrokes.

Chapter 5, Introducing Files, introduces files as the essential abstraction through which to understand Linux. You'll be introduced to the **Filesystem Hierarchy Standard (FHS)**, which is like a map that you can use to orient yourself on any Unix system. Then it's time for practical commands for working with files and directories in Linux, including some special filetypes you probably haven't heard of. You'll also get a taste of searching for files and file content, which is one of the most powerful bits of knowledge to have at your fingertips as a developer.

Chapter 6, Editing Files on the Command Line, introduces two text editors – nano and vim. You will learn the basics of using these text editors for command-line editing while also becoming aware of common editing mistakes and how to avoid them.

Chapter 7, Users and Groups, will introduce you to how the concepts of users and groups form the basis for the Unix security model, controlling access for resources like files and processes. We'll then teach you the practical commands you'll need to create and modify users and groups.

Chapter 8, Ownership and Permissions, builds on the previous chapter's explanation of users and groups to show you how access control works for resources in Linux. This chapter teaches you about ownership and permissions by walking you through file information from a long listing. From there, we'll look at the common file and directory permissions that you'll encounter on production Linux systems, before engaging with the Linux commands for modifying file ownership and permissions.

Chapter 9, Managing Installed Software, shows you how to install software on various Linux distributions (and even macOS). First, we introduce package managers, which are the preferred way of getting software onto a machine: you'll learn the important theory and practical commands for the package management operations you'll need as a software developer. Then we'll introduce a few other methods, like downloading install scripts and the time-honored, artisanal Unix tradition of compiling your own software locally, from source (it's not as scary as it sounds!).

Chapter 10, Configuring Software, piggybacks off the previous chapter's focus on installing software by helping you with configuring software on a Linux system. You will learn about the places that most software will look for configuration ("the configuration hierarchy"). Not only will this knowledge come in handy during late-night troubleshooting sessions, but it can actually help you to write better software. We'll cover command-line arguments, environment variables, configuration files, and how all of this works on non-standard Linux environments like Docker containers. There's even a little bonus project: you'll see how to take a custom program and turn it into its own systemd service.

Chapter 11, Pipes and Redirection, will give you an introduction to what is possibly the "killer feature" of Unix: the ability to connect existing programs into a custom solution using pipes. We'll move through the prerequisite theory and practical skills you need to understand: file descriptors and input/output redirection. Then you'll jump into creating complex commands using pipes. You'll be introduced to some essential CLI tools and practical pipe patterns, which you'll still find yourself using long after you finish this book.

Chapter 12, Automating Tasks with Shell Scripts, serves as a Bash scripting crash course, teaching you how to go from typing individual commands in an interactive shell to writing scripts. We assume you're already a software developer, so this will be a quick introduction that shows you the core language features and doesn't spend a lot of time re-explaining the basics of programming. You'll learn about Bash syntax, best practices for script writing, and some important pitfalls to avoid.

Chapter 13, Secure Remote Access with SSH, explores the Secure Shell Protocol and the related command-line tools available to you. You'll learn the basics of **public-key cryptography (PKI)**, which is always useful for a developer to know, before diving into creating SSH keys and securely logging into remote systems over the network. You'll build on this knowledge and get some experience copying files over the network, using SSH to create ad-hoc proxies or VPNs, and see examples of various other tasks that involve moving data over an encrypted SSH tunnel.

Chapter 14, Version Control with Git, shows you how to use a tool you probably already know well – `git` – from the command line, instead of through your IDE or a graphical client. We quickly go through the basic theory behind git and then jump into the commands you'll need to use in a command-line environment. We'll cover two powerful features that it pays to understand – bisecting and rebasing – and then give you our take on best practices and useful shell aliases. Finally, the *Poor man's GitHub* section presents a small but legitimately useful project that you can do to practice and integrate the Linux skills you've learned up to this point.

Chapter 15, Containerizing Applications with Docker, gives you the basic theory and practical skills that will make it easy to work with Docker as a developer. We'll explore the problems that Docker solves, explain the most important Docker concepts, and take you through the core workflow and commands you'll use. You'll also see how to build your own images by containerizing a real application. And because we're approaching this from a software development and Linux perspective, you'll also develop a good intuition for how containerization works under the hood, and how it's different from virtual machines.

Chapter 16, Monitoring Application Logs, gives an overview of logging on Unix and Linux. We'll show you how (and where) logs are collected on most modern Linux systems using `systemd`, and how more traditional approaches work (you'll come across both in the real world). You'll build practical command-line skills finding and viewing logs and learn a bit about how logging is being done in larger infrastructures.

Chapter 17, Load Balancing and HTTP, covers the basics of HTTP for developers, with a special focus on the complexities that you'll come across when working with HTTP services in larger infrastructures. We'll correct some common misunderstandings about HTTP statuses, HTTP headers, and HTTP versions and how applications should handle them. We'll also introduce how load balancers and proxies work in the real world, and how they make the experience of troubleshooting a live application quite different from troubleshooting a development version on your laptop. Many of the Linux skills that you will have learned up to this point will come in handy here, and we'll introduce a new tool – `curl` – to help you troubleshoot a wide variety of HTTP-related issues.

To get the most out of this book

If you can get yourself to a Linux shell prompt – by installing Ubuntu in a virtual machine or running it as a Docker container, for example – you can follow along with everything in this book.

You can get away with even less – on Windows, there's WSL, and macOS is a bona-fide Unix operating system, so almost all of the practical commands you learn in this book (except those called out as Linux-only) will work out of the box. That said, for the best experience, follow along on a Linux operating system.

The skills required to get the most out of this book are only the basic computer skills that you already have as a software developer – editing text, working with files and folders, having some notion of what “operating systems” are, installing software, and using a development environment. Everything beyond that, we'll teach you.

Download the color images

We also provide a PDF file that has color images of the screenshots/diagrams used in this book. You can download it here: <https://packt.link/gbp/9781804616925>.

Conventions used

There are a number of text conventions used throughout this book.

CodeInText: Indicates code words in text, database table names, folder names, filenames, file extensions, pathnames, dummy URLs, user input, and Twitter handles. For example: “The `-f` flag stands for ‘follow,’ and the `-u` flag stands for ‘unit.’”

A block of command line is set as follows:

```
/home/steve/Desktop# ls
anotherfile  documents  somefile.txt  stuff
/home/steve/Desktop# cd documents/
/home/steve/Desktop/documents# ls
contract.txt
```

Bold: Indicates a new term, an important word, or words that you see on the screen. For instance, words in menus or dialog boxes appear in the text like this. For example: “When a file is set to be executable, Unix will do its best to execute it, either succeeding in the case of **ELF (Executable and Linkable Format)**, probably the most widely used executable format today) or failing.”



Warnings or important notes appear like this.



Tips and tricks appear like this.

Get in touch

Feedback from our readers is always welcome.

General feedback: Email feedback@packtpub.com, and mention the book’s title in the subject of your message. If you have questions about any aspect of this book, please email us at questions@packtpub.com.

Errata: Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you have found a mistake in this book we would be grateful if you would report this to us. Please visit, <http://www.packtpub.com/submit-errata>, selecting your book, clicking on the Errata Submission Form link, and entering the details.

Piracy: If you come across any illegal copies of our works in any form on the Internet, we would be grateful if you would provide us with the location address or website name. Please contact us at copyright@packtpub.com with a link to the material.

If you are interested in becoming an author: If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, please visit <http://authors.packtpub.com>.

Share your thoughts

Once you've read *The Software Developer's Guide to Linux*, we'd love to hear your thoughts! Please [click here](#) to go straight to the Amazon review page for this book and share your feedback.

Your review is important to us and the tech community and will help us make sure we're delivering excellent quality content.