Third Edition

The Official ubuntu Server Book

Kyle Rankin
Benjamin Mako Hill

Praise for *The Official Ubuntu Server Book*

"Murphy's Law is never truer than when it comes to administering a Linux server. You can pretty much count on something happening to your machine at a time when you need it the most. That's when a book with some basic troubleshooting instructions is worth every penny you paid for it. Chapter 11 covers the steps you should take when something goes wrong."

—Paul Ferrill, LinuxPlanet.com reviewer

"College-level collections catering to Linux programmers and developers will find *The Official Ubuntu Server Book,* a top addition to the collection, covering a complete, free server operating system in a guide to getting going quickly. From making the most of Ubuntu Server's latest technologies to automating installs and protecting the server using Ubuntu's built-in security tools, *The Official Ubuntu Server Book,* is packed with keys to success for any Ubuntu user."

—Jim Cox, *Midwest Book Review*

"This book will get you started on the path of the server admin, both within the context of Ubuntu server and in the larger realm of managing a server infrastructure. The desktop and server versions of Ubuntu are continuing to mature. Read this book if you want to keep up."

—James Pyles, author

This page intentionally left blank

The Official Ubuntu Server Book

Third Edition

Kyle Rankin
Benjamin Mako Hill

Pearson

Upper Saddle River, NJ • Boston • Indianapolis • San Francisco
New York • Toronto • Montreal • London • Munich • Paris • Madrid
Capetown • Sydney • Tokyo • Singapore • Mexico City

Cataloging-in-Publication Data is on file with the Library of Congress.

ISBN-13: 978-0-13-301753-3
ISBN-10: 0-13-301753-2
Text printed in the United States on recycled paper at Edwards Brothers Malloy in Ann Arbor, Michigan.
First printing, July 2013

I dedicate this book to my wife, Joy. It is not easy to balance a full-time job and writing a book while still having time for a family. She has endured many a book-writing process at this point and has always been my main source of support and motivation.

—Kyle Rankin

This page intentionally left blank

Contents at a Glance

This page intentionally left blank

Contents

Preface

WELCOME to the third edition of *The Official Ubuntu Server Book*!

When most people talk about Ubuntu these days, they tend to talk about the Ubuntu Desktop. After all, it's the easy-to-use, "just works" approach to the desktop that has made Ubuntu one of the most popular desktop Linux distributions. What has gotten less attention, although even that is starting to change, is Ubuntu Server. It turns out that desktop Linux users aren't the only ones who want their distribution to "just work"—system administrators appreciate that on their servers as well. In Ubuntu Server you will find all of the powerful server infrastructure from the Debian project plus that extra bit of Ubuntu polish, innovation, and focus on ease of use.

About This Book

This book is the result of the collaborative effort of not just the principal authors, but of the Ubuntu Server team itself. As it is the official, authorized book on Ubuntu Server, the focus has been on a server guide based on our collective experience. Beyond that, the goal is to have something to offer to both the beginner system administrator and the battle-hardened senior sysadmin. On the surface it might seem a tough balance to achieve, but in reality both groups ultimately want the same thing: for their servers to work. Now it's true that some administrators revel in doing things the hard way. Some even treat it as a point of pride. The thing is, all of us who have administered servers for years can do and have done things the hard way as well, but ultimately you realize that there's nothing particularly impressive in doing everything by hand—in the end you just have too much to do and any time-saving steps are welcome.

As you will see, most of this book takes a pragmatic approach to server management. Where Ubuntu offers new programs or features to ease administration and save time, you will find them mentioned here. If you are a beginner administrator, you will find that administering an Ubuntu server isn't nearly as difficult as you might think. Experienced administrators, especially those coming from other platforms, will find numerous time-saving tips and programs, as well as where Ubuntu has updated how a service is organized (Apache being a good example); you can treat this book as a map to point you to all of the right directories.

One great thing about Ubuntu as a server is that there are so many great server packages available for it. Of course, this creates a dilemma for us as writers: It's just not possible to feature every available e-mail and IMAP/POP3 server, for instance. In these cases we've tried to pick out programs that are easy to install, configure, and use under Ubuntu, as well as highlight programs that are preferred by the authors and server team. While doing that, there's a good chance that your favorite program for X, Y, and Z was left out. It's certainly no slight against any of those programs—we just had to draw the line somewhere.

How the Book Is Organized

Different people read tech books differently. Some people read them cover to cover, and others skip right ahead to the topic they need immediate help with. You will find that the way this book is organized lends itself well to both approaches. The first few chapters lay the foundation so you can install Ubuntu and navigate the system even if it's your first time. After that the chapters focus on particular server topics, from security to monitoring to system rescue.

- Chapter 1—Installation. In the first chapter you will learn how to use the default Ubuntu Server CD to install Ubuntu on a server. This guide includes a complete walk-through of the installation process from the initial boot screen to partitioning to your first login prompt.
- Chapter 2—Essential System Administration. If you are new to Ubuntu system administration, the amount of learning ahead of you might seem daunting. In this chapter you will find not only a solid foundation of instructions on how to navigate the Linux command

line, but also an introduction to the Ubuntu boot process and the standards behind all of the directories on an Ubuntu system. By the end of the chapter you should have a good basis to continue with the rest of the book.

- Chapter 3—Package Management. This chapter introduces you to packages and the packaging system—the way that Ubuntu handles the installation, removal, and management of software. We provide a solid foundation in what packages do and how they do it before drilling down into the details of how an administrator can manage software the Ubuntu way. In the final pages, we cover the way that administrators can switch from being consumers to producers and begin making their own packages.
- Chapter 4—Automated Ubuntu Installs. While you can certainly install Ubuntu step by step from the install CD, that method doesn't work so well when you have tens or hundreds of servers to install. This chapter covers the preseed method for automating Ubuntu installs along with Kickseed—Ubuntu's port of Kickstart. In addition to a description of how to use both of these technologies independently, you will find out it's even better when you use them together.
- Chapter 5—Guide to Common Ubuntu Servers. There is an enormous number of services you can run on an Ubuntu server. In this chapter we highlight some of the more popular services, from Web to e-mail to file services. If you are a new administrator, you will find a simple guide on how to install and configure these services for the first time. If you are an experienced administrator coming from another distribution, you will find this chapter a handy, how-to guide on how Ubuntu organizes all of the configuration files for your favorite services.
- Chapter 6—Security. Security is an important topic for any administrator. Ubuntu Server already is pretty secure by default, and in this chapter we highlight some of these mechanisms, along with steps you can take to increase your security even further. Some of the security topics include `sudo`, firewall configuration, an introduction to forensics, and even Ubuntu's AppArmor software.
- Chapter 7—Backups. There are two kinds of administrators: those who back up their servers and those who haven't lost valuable data yet. Backup software abounds for Linux as a whole and for Ubuntu

specifically, and in this chapter you will see a few easy-to-set-up approaches to keeping your data secure.

- Chapter 8—Monitoring. Monitoring is one of the most valuable systems an administrator can set up while simultaneously being the most annoying (why do servers always seem to page you in the middle of the night?). In this chapter we cover some different approaches to monitoring systems both for trending purposes and to alert you to any problems. By the end of the chapter you will no longer lose sleep wondering if a server is up—you'll lose sleep only when it goes down.

- Chapter 9—Virtualization and Cloud Computing. Virtualization and cloud computing are hot topics in system administration today. With increasingly powerful hardware out there, virtualization provides you with a way to squeeze the most efficiency out of your servers and cloud computing abstracts even further so that servers can be created and destroyed on a whim. In this chapter we cover one of the most popular server-based virtualization tools out there: KVM. We also cover how to use Amazon's EC2 cloud environment with command-line tools and also how to automate EC2 server deployment with a new Canonical tool called Juju.

- Chapter 10—Fault Tolerance. If a lot is riding on your servers and your downtime is measured in dollars and not minutes, you realize very quickly that your servers need fault tolerance. This chapter covers the Ubuntu software RAID, including steps to migrate from one type of RAID to another. Then we will cover how to set up redundant network connections and finish up with a guide to setting up your own Linux cluster. We also discuss how to get up and running with logical volume management (LVM).

- Chapter 11—Troubleshooting. No matter how great an administrator you are, eventually something on your servers will fail. Over the years you develop a series of troubleshooting steps you go through whenever you find a problem on your systems. In this chapter we condense years of troubleshooting experience into a series of step-by-step guides to walk you through common server and network problems and how to use standard Ubuntu tools and techniques to diagnose them.

- Chapter 12—Rescue and Recovery. We've often said that we've learned more about Linux from fixing a broken system than in any other way. In some environments when a system won't boot, an administrator might just install a new operating system. Under Ubuntu, however, you'll find that most common boot problems also have a common, easy solution. In this chapter we discuss how to use different stages of rescue modes both on Ubuntu and the Ubuntu Server install CD itself to repair your system.
- Chapter 13—Help and Resources. One great thing about Ubuntu is just how many support avenues there are when you need help. Whether it's documentation on the machine itself, guides on the official Ubuntu site, forums, or even professional Canonical support, when you are stuck you aren't alone. In this chapter we cover all of the different ways to get support for your Ubuntu server.
- Chapter 14—Basic Linux Administration. This chapter picks up where Chapter 2, Essential System Administration, left off. Here we discuss some of the core foundation concepts behind Linux administration, including file permissions, different file types, pipes, and other core Linux information. Beginner administrators will find this a very useful guide to flesh out any gaps in their command-line knowledge, and the experienced administrators will find it a good refresher on core concepts.
- Appendix—Cool Tips and Tricks. Over the years you develop all sorts of useful tips, one-liners, and other shell commands that make your life as an administrator easier. Here you will find some of our favorite time-saving tips and hacks in rapid-fire form.

Media with This Book

This book includes two versions of Ubuntu Server: Ubuntu 12.04.2 for 64-bit machines and the latest Ubuntu 13.04 release, so you can pick the version that best matches your needs.

Although we have included both Ubuntu 12.04.2 and 13.04 releases and have written the book for both versions, you might decide to try out a

newer Ubuntu release. In that case, just go to http://ubuntu.com and either download the CD image or request a copy to be sent to you. No matter which Ubuntu Server CD you pick, it's relatively easy to use the CDs. Just insert the version you want to install into your computer and boot from the CD-ROM. When the CD boots, you will see a number of options on the screen, but to install Ubuntu Server, just select Install Ubuntu Server. The installer that launches will ask some fairly straightforward questions common to most install discs, and if you get stuck, just turn to Chapter 1 for a more in-depth walk-through of the install process.

Acknowledgments

JORGE, I WOULDN'T HAVE been involved in this book if it weren't for you. I'm one in a long list of people using Ubuntu because of Jorge. His enthusiasm is infectious, and I can't count how many times he's introduced me to some cool new program or tool that I write off at first and then somehow find myself using eventually.

Debra and Mako, it has been great working with both of you on this project, and thank you for the opportunity and guidance. Also thanks to Matthew for his help on the support chapter. Robert, thanks so much for your great attention to detail and tracking down all the areas where I had made typos and mistakes. Thanks to Bill "The Cloud" Childers for providing me with equipment for the UEC section.

Extra thanks to Dustin, Nick, Jamie, Kees, Alan, Mathias, Thierry, and the rest of the Ubuntu Server team for all of your excellent feedback and help through this process.

—Kyle Rankin

This page intentionally left blank

About the Authors

Kyle Rankin is a senior systems administrator, the author of *DevOps Troubleshooting*, *Knoppix Hacks*, *Knoppix Pocket Reference*, *Linux Multimedia Hacks*, and *Ubuntu Hacks*, and he has contributed to a number of other O'Reilly books. Kyle is also an award-winning columnist for *Linux Journal* and has had articles featured in *PC Magazine*, *TechTarget*, and other publications.

Benjamin Mako Hill is a Ph.D. candidate at the Sloan School of Management and Media Lab at MIT, and, as of Fall 2013, an assistant professor of communications at the University of Washington. As part of the founding Ubuntu team, his charge at Canonical was to help grow the Ubuntu development and user community during the project's first year. Mako has continued his involvement with Ubuntu as a member of the Community Council governance board and through projects such as this book.

This page intentionally left blank

Introduction

THIS INTRODUCTION GIVES AN overview of Ubuntu and Ubuntu Server. After a quick welcome, it includes a brief history of free software, open source, and GNU/Linux and of the Ubuntu project itself, with a focus on some of the major players on the Ubuntu scene. This introduction ends where the rest of this book will continue: with a history of the Ubuntu Server project and an overview of that project's goals and accomplishments.

Welcome to Ubuntu Server

In the just over eight years of its life, Ubuntu has become one of the most popular GNU/Linux-based operating systems. In the process, however, public perception has been disproportionately focused on Ubuntu's role as a desktop-based operating system. While all popularity is certainly welcome for those of us involved in the project, this success has, at times, overshadowed the rock-solid server operating system that Ubuntu has been constructed to be. For those of us who have helped build out Ubuntu's server-specific features and who use it daily, this is both unfortunate and undeserved. Designed and used as a server since day one, Ubuntu has supported a server team that was one of the first active teams in the Ubuntu community and has been one of the most successful. Although perceptions have changed in large part, many prospective users—and even some current Ubuntu users—often continue to think of Ubuntu as something for desktops.

Perhaps it is just that people are so surprised at the usability of Ubuntu on the desktop—especially in the early days when expectations for desktop GNU/Linux distributions were low—that the public focus naturally has drifted away from Ubuntu's server offering. Lots of other GNU/Linux distributions run great on servers, but a solid desktop experience continues

to be surprising to many users. As a result, when people talk about Ubuntu, they often tend to talk about desktops. Perhaps, on the other hand, people just figured that such a well-polished desktop must have come at the cost of the server-oriented features and support. Of course, no such sacrifices were made.

To a large extent, times have changed. The Ubuntu Server team has continued its tireless work both to improve the experience for server users of Ubuntu and to help promote Ubuntu as a server solution. Documentation, testimonials, certification of server-based software, support contracts from a variety of sources, training courses, and more have all contributed to remaking Ubuntu into a powerful player on the server. Although its desktop credentials have not been diminished, Ubuntu's server chops are increasingly difficult to overlook. Over the past two years, Ubuntu has begun to become a major player in the GNU/Linux server market.

More than anything else, testimonials have spread and the small group of early Ubuntu Server users has spread the word. More and more people choose Ubuntu for their servers every day. In fact, this book is simply the latest striking example of just how far Ubuntu on servers has come. Not only do people now know that Ubuntu runs on a server, they know it runs well. This book is publishable only because there is a market for it. That market is made up of people who have heard good things about Ubuntu on the server and who are getting ready to take the plunge themselves. Welcome. We hope we can help make the process easier. We've come a long way, and we're still only just beginning.

Free Software, Open Source, and Linux

A history of Ubuntu Server must, in large part, be a history of Ubuntu itself. A history of Ubuntu must, in large part, be a history of the free software movement and of the Linux kernel. While thousands of individuals have contributed in some form to Ubuntu, the project has succeeded only through the contributions of many thousands more who have indirectly laid the technical, social, and economic groundwork for Ubuntu's success. While introductions to free software, open source, and GNU/Linux can be found in many other places, no introduction to Ubuntu is complete with-

out a brief discussion of these concepts and the people and history behind them. It is around these concepts and within these communities that Ubuntu was motivated and born. Ultimately, it is through these ideas that it is sustained.

Free Software and GNU

In a series of events that have almost become legend through constant repetition, Richard M. Stallman created the concept of free software in 1983. Stallman grew up with computers in the 1960s and 1970s, when computer users purchased very large and extremely expensive mainframe computers, which were then shared among large numbers of programmers. Software was, for the most part, seen as an add-on to the hardware, and every user had the ability and the right to modify or rewrite the software on his or her computer and to freely share this software. As computers became cheaper and more numerous in the late 1970s, producers of software began to see value in the software itself. Producers of computers began to argue that their software was copyrightable and a form of intellectual property much like a music recording, a film, or a book's text. They began to distribute their software under licenses and in forms that restricted its users' abilities to use, redistribute, or modify the code. By the early 1980s, restrictive software licenses had become the norm.

Stallman, then a programmer at MIT's Artificial Intelligence Laboratory, became increasingly concerned with what he saw as a dangerous loss of the freedoms that software users and developers had up until that point enjoyed. He was concerned with computer users' ability to be good neighbors and members of what he thought was an ethical and efficient computer-user community. To fight against this negative tide, Stallman articulated a vision for a community that developed liberated code—in his words, "free software." He defined free software as software that had the following four characteristics—labeled as freedoms 0 through 3 instead of 1 through 4 as a computer programmer's joke:

- The freedom to run the program for any purpose (freedom 0)
- The freedom to study how the program works and adapt it to your needs (freedom 1)

- The freedom to redistribute copies so you can help your neighbor (freedom 2)
- The freedom to improve the program and release your improvements to the public so that the whole community benefits (freedom 3)

Access to source code—the human-readable and modifiable blueprints of any piece of software that can be distinguished from the computer-readable version of the code that most software is distributed as—is a prerequisite to freedoms 1 and 3. In addition to releasing this definition of free software, Stallman began a project with the goal of creating a completely free OS to replace the then-popular UNIX. In 1984, Stallman announced this project and called it GNU—another joke in the form of a recursive acronym for "GNU's Not UNIX."

Linux

By the early 1990s, Stallman and a collection of other programmers working on GNU had developed a near-complete OS that could be freely shared. They were, however, missing a final essential piece in the form of a kernel—a complex system command processor that lies at the center of any OS. In 1991, Linus Torvalds wrote an early version of just such a kernel, released it under a free license, and called it Linux. Linus's kernel was paired with the GNU project's development tools and OS and with the graphical windowing system called X. With this pairing, a completely free OS was born—free both in terms of price and in Stallman's terms of freedom.

All systems referred to as Linux today are, in fact, built on the work of this collaboration. Technically, the term Linux refers only to the kernel. Many programmers and contributors to GNU, including Stallman, argue emphatically that the full OS should be referred to as GNU/Linux in order to give credit not only to Linux but also to the GNU project and to highlight GNU's goals of spreading software freedom—goals not necessarily shared by Linus Torvalds. Many others find this name cumbersome and prefer calling the system simply Linux. Yet others, such as those working on the Ubuntu project, attempt to avoid the controversy altogether by referring to GNU/Linux only by using their own project's name.

Open Source

Disagreements over labeling did not end with discussions about the naming of the combination of GNU and Linux. In fact, as the list of contributors to GNU and Linux grew, a vibrant world of new free-software projects sprouted up, facilitated in part by growing access to the Internet. As this community grew and diversified, a number of people began to notice an unintentional side effect of Stallman's free software. Because free software was built in an open way, *anyone* could contribute to software by looking through the code, finding bugs, and fixing them. Because software ended up being examined by larger numbers of programmers, free software was higher in quality, performed better, and offered more features than similar software developed through proprietary development mechanisms. In many situations, the development model behind free software led to software that was *inherently better* than proprietary alternatives.

As the computer and information technology industry began to move into the dot-com boom, one group of free software developers and leaders, spearheaded by two free software developers and advocates—Eric S. Raymond and Bruce Perens—saw the important business proposition offered by a model that could harness volunteer labor or interbusiness collaboration and create intrinsically better software. However, they worried that the term *free software* was problematic for at least two reasons. First, it was highly ambiguous—the English word *free* means both gratis, or at no cost (e.g., as in "free beer") and liberated in the sense of freedom (e.g., as in "free speech"). Second, there was a feeling, articulated most famously by Raymond, that all this talk of freedom was scaring off the very business executives and decision makers whom the free software movement needed to impress in order to succeed.

To tackle both of these problems, this group coined a new phrase—*open source*—and created a new organization called the Open Source Initiative. The group set at its core a definition of open source software that overlapped completely and exclusively both with Stallman's four-part definition of free software and with other community definitions that were also based on Stallman's.

One useful way to understand the split between the free software and open source movements is to think of it as the opposite of a schism. In religious

schisms, churches separate and do not work or worship together because of relatively small differences in belief, interpretation, or motivation. For example, most contemporary forms of Protestant Christianity agree on *almost* everything but have separated over some small but irreconcilable difference. However, in the case of the free software and open source movements, the two groups have fundamental disagreements about their motivation and beliefs. One group is focused on freedom, while the other is focused on pragmatics. Free software is most accurately described as a social movement, whereas open source is a development methodology. However, the two groups have no trouble working on projects hand in hand.

In terms of the motivations and goals, open source and free software diverge greatly. Yet in terms of the software, the projects, and the licenses they use, they are completely synonymous. While people who identify with either group see the two movements as being at odds, the Ubuntu project sees no conflict between the two ideologies. People in the Ubuntu project identify with either group and often with both. In this book, we may switch back and forth between the terms as different projects and people in Ubuntu identify more strongly with one term or the other. For the purposes of this book, though, either term should be read as implying the other unless it is stated otherwise.

A Brief History of the Ubuntu Project

A history of Ubuntu, born in April 2004, may seem premature. However, the last six years have been full ones for Ubuntu. With its explosive growth, it is difficult even for those involved most closely with the project to track and record some of the high points. Importantly, there are some key figures whose own history must be given for a full understanding of Ubuntu. This brief summary outlines the high points of Ubuntu's history to date and gives the necessary background knowledge to understand where Ubuntu comes from.

Mark Shuttleworth

No history of Ubuntu can call itself complete without a history of Mark Shuttleworth. Shuttleworth is, undeniably, the most visible and important person in Ubuntu. More important from the point of view of history,

Shuttleworth is also the originator and initiator of the project—he made the snowball that would eventually roll on and grow to become the Ubuntu project.

Shuttleworth was born in 1973 in Welkom, Free State, in South Africa. He attended Diocesan College and obtained a business science degree in finance and information systems at the University of Cape Town. During this period, he was an avid computer hobbyist and became involved with the free and open source software community. He was at least marginally involved in both the Apache project and the Debian project and was the first person to upload the Apache Web server, perhaps the single most important piece of server software on GNU/Linux platforms, into the Debian project's archives.

Seeing an opportunity in the early days of the Web, Shuttleworth founded a certificate authority and Internet security company called Thawte in his garage. Over the course of several years, he built Thawte into the second-largest certificate authority on the Internet, trailing only the security behemoth VeriSign. Throughout this period, Thawte's products and services were built and served almost entirely from free and open source software. In December 1999, Shuttleworth sold Thawte to VeriSign for an undisclosed amount that reached into the hundreds of millions in U.S. dollars.

With his fortune made at a young age, Shuttleworth might have enjoyed a life of leisure—and probably considered it. Instead, he decided to pursue his lifelong dream of space travel. After paying approximately $20 million to the Russian space program and devoting nearly a year to preparation, including learning Russian and spending seven months training in Star City, Russia, Shuttleworth realized his dream as a civilian cosmonaut aboard the Russian Soyuz TM-34 mission. On this mission, Shuttleworth spent two days on the Soyuz rocket and eight days on the International Space Station, where he participated in experiments related to AIDS and genome research. In early May 2002, Shuttleworth returned to Earth.

In addition to space exploration and a less-impressive jaunt to Antarctica, Shuttleworth played an active role as both a philanthropist and a venture capitalist. In 2001, he founded the Shuttleworth Foundation (TSF), a non-profit organization based in South Africa. The foundation was chartered

to fund, develop, and drive social innovation in the field of education. Of course, the means by which TSF attempts to achieve these goals frequently involves free software. Through these projects, the organization has been one of the most visible proponents of free and open source software in South Africa and even the world. In the venture capital area, Shuttleworth worked to foster research, development, and entrepreneurship in South Africa with strategic injections of cash into start-ups through a new venture capital firm called HBD, an acronym for "Here Be Dragons." During this period, Shuttleworth was busy brainstorming his next big project—the project that would eventually become Ubuntu.

The Warthogs

There has been no lack of projects attempting to wrap GNU, Linux, and other pieces of free and open source software into a neat, workable, and user-friendly package. Mark Shuttleworth, like many other people, believed that the philosophical and pragmatic benefits offered by free software put it on a course for widespread success. That said, none of the offerings were particularly impressive. Something was missing from all of them. Shuttleworth saw this as an opportunity. If someone could build the great free software distribution that helped push GNU/Linux into the mainstream, he or she would come to occupy a position of strategic importance.

Shuttleworth, like many other technically inclined people, was a huge fan of the Debian project (discussed in depth later). However, many things about Debian did not fit with Shuttleworth's vision of an ideal OS. For a period of time, Shuttleworth considered the possibility of running for Debian project leader as a means of reforming the Debian project from within. With time, though, it became clear that the best way to bring GNU/Linux into the mainstream would not be from within the Debian project—which in many situations had very good reasons for being the way it was. Instead, Shuttleworth would create a new project that worked in symbiosis with Debian to build a new, better GNU/Linux system.

To kick off this project, Shuttleworth invited a dozen or so free and open source software developers he knew and respected to his flat in London in April 2004. It was in this meeting (alluded to in the first paragraphs of this

introduction) that the groundwork for the Ubuntu project was laid. By that point, many of those involved were excited about the possibility of the project. During this meeting, the members of the team—which would in time grow into the core Ubuntu team—brainstormed a large list of the things that they would want to see in their ideal OS. The list is now a familiar list of features to most Ubuntu users. Many of these traits are covered in more depth later in this introduction. The group wanted

- Predictable and frequent release cycles
- A strong focus on localization and accessibility
- A strong focus on ease of use and user-friendliness on the desktop
- A strong focus on Python as the single programming language through which the entire system could be built and expanded
- A community-driven approach that worked with existing free software projects and a method by which the groups could give back as they went along—not just at the time of release
- A new set of tools designed around the process of building distributions that allowed developers to work within an ecosystem of different projects and that allowed users to give back in whatever ways they could

There was consensus among the group that actions speak louder than words, so there were no public announcements or press releases. Instead, the group set a deadline for itself—six short months in the future. Shuttleworth agreed to finance the work and pay the developers full-time salaries to work on the project. After six months, they would both announce their project and reveal the first product of their work. They made a list of goals they wanted to achieve by the deadline, and the individuals present took on tasks. Collectively, they called themselves the Warthogs.

What Does *Ubuntu* Mean?

At this point, the Warthogs had a great team, a set of goals, and a decent idea of how to achieve most of them. The team did not, on the other hand, have a name for the project. Shuttleworth argued strongly that they should call the project Ubuntu.

Ubuntu is a concept and a term from several South African languages, including Zulu and Xhosa. It refers to a South African ideology or ethic that, while difficult to express in English, might roughly be translated as "humanity toward others," or "I am because we are." Others have described ubuntu as "the belief in a universal bond of sharing that connects all humanity." The famous South African human rights champion Archbishop Desmond Tutu explained ubuntu in this way:

> A person with ubuntu is open and available to others, affirming of others, does not feel threatened that others are able and good, for he or she has a proper self-assurance that comes from knowing that he or she belongs in a greater whole and is diminished when others are humiliated or diminished, when others are tortured or oppressed.

Ubuntu played an important role as a founding principle in postapartheid South Africa and remains a concept familiar to most South Africans today.

Shuttleworth liked the term *Ubuntu* as a name for the new project for several reasons. First, it is a South African concept. While the majority of the people who work on Ubuntu are not from South Africa, the roots of the project are, and Shuttleworth wanted to choose a name that represented this. Second, the project emphasizes the definition of individuality in terms of relationships with others and provides a profound type of community and sharing—exactly the attitudes of sharing, community, and collaboration that are at the core of free software. The term represented the side of free software that the team wanted to share with the world. Third, the idea of personal relationships built on mutual respect and connections describes the fundamental ground rules for the highly functional community that the Ubuntu team wanted to build. *Ubuntu* was a term that encapsulated where the project came from, where the project was going, and how the project planned to get there. The name was perfect. It stuck.

Creating Canonical

In order to pay developers to work on Ubuntu full-time, Shuttleworth needed a company to employ them. He wanted to pick some of the best people for the jobs from within the global free software and open source communities. These communities, inconveniently for Shuttleworth, know no national and geographic boundaries. Rather than move everyone to a

single locale and office, Shuttleworth made the decision to employ these developers through a virtual company. While this had obvious drawbacks in the form of high-latency and low-bandwidth connections, different time zones, and much more, it also introduced some major benefits in the particular context of the project. On one hand, the distributed nature of employees meant that the new company could hire individuals without requiring them to pack up their lives and move to a new country. More important, it meant that *everyone* in the company was dependent on IRC, mailing lists, and online communication mechanisms to do their work. This unintentionally and automatically solved the water-cooler problem that plagued many other corporately funded free software projects—namely, that developers would casually speak about their work in person and cut the community and anyone else who didn't work in the office out of the conversation completely. For the first year, the closest thing that Canonical had to an office was Shuttleworth's flat in London. While the company has grown and now has several offices around the world, it remains distributed, and a large number of the engineers work from home. The group remains highly dependent on Internet collaboration.

With time, the company was named Canonical. The name was a nod to the project's optimistic goals of becoming the canonical place for services and support for free and open source software and for Ubuntu in particular. *Canonical*, of course, refers to something that is accepted as authoritative. It is a common word in the computer programmer lexicon. It's important to note that being canonical is like being standard; it is not coercive. Unlike holding a monopoly, becoming the canonical location for something implies a similar sort of success—but *never* one that cannot be undone and never one that is exclusive. Other companies will support Ubuntu and build operating systems based on it, but as long as Canonical is doing a good job, its role will remain central.

The Ubuntu Community

By now you may have noticed a theme that permeates the Ubuntu project on several levels. The history of free software and open source is one of a profoundly effective *community*. Similarly, in building a GNU/Linux distribution, the Ubuntu community has tried to focus on an ecosystem model—an organization of organizations—in other words, a community.

Even the definition of the term *ubuntu* is one that revolves around people interacting in a community.

It comes as no surprise, then, that an "internal" community plays heavily into the way that the Ubuntu distribution is created. While the Ubuntu 4.10 version (Warty Warthog) was primarily built by a small number of people, Ubuntu achieved widespread success only through contributions by a much larger group that included programmers, documentation writers, volunteer support staff, and users. While Canonical employs a core group of several dozen active contributors to Ubuntu, the distribution has, from day one, encouraged and incorporated contributions from *anyone* in the community and rewards and recognizes contributions by all. Rather than taking center stage, paid contributors are *not* employed by the Ubuntu project—instead they are employed by Canonical, Ltd. These employees are treated simply as another set of community members. They must apply for membership in the Ubuntu community and have their contributions recognized in the same way as anyone else. All non-business-related communication about the Ubuntu project occurs in public and in the community. Volunteer community members occupy a majority of the seats on the two most important governing boards of the Ubuntu project: the Technical Board, which oversees all technical matters, and the Community Council, which approves new Ubuntu members and resolves disputes. Seats on both boards are approved by the relevant community groups, developers for the Technical Board and Ubuntu members for the Community Council.

In order to harness and encourage the contributions of its community, Ubuntu has striven to balance the important role that Canonical plays with the value of empowering individuals in the community. The Ubuntu project is based on a fundamental belief that great software is built, supported, and maintained only in a strong relationship with the individuals who use the software. In this way, by fostering and supporting a vibrant community, Ubuntu can achieve much more than it could through paid development alone. The people on the project believe that while the contributions of Canonical and Mark Shuttleworth have provided an important catalyst for the processes that have built Ubuntu, it is the community that has brought the distribution its success to date. The project members believe that it is only through increasing reliance on the community that

the project's success will continue to grow. The Ubuntu community won't outspend the proprietary software industry, but it is very much more than Microsoft and its allies can afford.

Finally, it is worth noting that, while this book is official, neither of the authors is a Canonical employee. This book, like much of the rest of Ubuntu, is purely a product of the project's community.

Ubuntu Promises and Goals

So far, this Introduction has been about the prehistory, history, and context of the Ubuntu project. After this introduction, the book focuses on the distribution itself. Before proceeding, it's important to understand the goals that motivated the project.

Philosophical Goals

The most important goals of the Ubuntu project are philosophical in nature. The Ubuntu project lays out its philosophy in a series of documents on its Web site. In the most central of these documents, the team summarizes the charter and the major philosophical goals and underpinnings:

> Ubuntu is a community-driven project to create an operating system and a full set of applications using free and Open Source software. At the core of the Ubuntu Philosophy of Software Freedom are these core philosophical ideals:
>
> 1. Every computer user should have the freedom to run, copy, distribute, study, share, change, and improve their software for any purpose without paying licensing fees.
> 2. Every computer user should be able to use their software in the language of their choice.
> 3. Every computer user should be given every opportunity to use software, even if they work under a disability.

The first item should be familiar by now. It is merely a recapitulation of Stallman's free software definition quoted earlier in the section on free software history. In it, the Ubuntu project makes explicit its goal that every user of software should have the freedoms required by free software. This

is important for a number of reasons. First, it offers users all of the practical benefits of software that runs better, faster, and more flexibly. More important, it gives every user the capability to transcend his or her role as a user and a consumer of software. Ubuntu wants software to be empowering and to work in the ways that users want it to work. Ubuntu wants all users to have the ability to make sure it works for them. To do this, software *must* be free, so Ubuntu makes this a requirement and a philosophical promise.

Of course, the core goals of Ubuntu do not end with the free software definition. Instead, the project articulates two new, but equally important, goals. The first of these, that all computer users should be able to use their computers in their chosen languages, is a nod to the fact that the majority of the world's population does not speak English while the vast majority of software interacts only in that language. To be useful, source code comments, programming languages, documentation, and the texts and menus in computer programs must be written in *some* language. Arguably, the world's most international language is a reasonably good choice. However, there is no language that everyone speaks, and English is not useful to the majority of the world's population that does not speak it. A computer can be a great tool for empowerment and education, but only if the user can understand the words in the computer's interface. As a result, Ubuntu believes that it is the project's—and community's—responsibility to ensure that *every* user can easily use Ubuntu to read and write in the language with which he or she is most comfortable.

Finally, just as no person should be blocked from using a computer simply because he or she does not know a particular language, no user should be blocked from using a computer because of a disability. Ubuntu must be accessible to users with motor disabilities, vision disabilities, and hearing disabilities. It should provide input and output in a variety of forms to account for each of these situations and for others. A significant percentage of the world's most intelligent and creative individuals has disabilities. Ubuntu's impact should not be limited to any subset of the world when it can be fully inclusive. More important, Ubuntu should be able to harness the ability of these individuals as community members to build a better and more effective community.

Conduct Goals and Code of Conduct

If Ubuntu's philosophical commitments describe the *why* of the Ubuntu project, the Code of Conduct (CoC) describes Ubuntu's *how*. Ubuntu's CoC is, arguably, the most important document in the day-to-day operation of the Ubuntu community and sets the ground rules for work and cooperation within the project. Explicit agreement to the document is the only criterion for becoming an officially recognized Ubuntu activist—an Ubuntero—and is an essential step toward membership in the project.

The CoC covers "behavior as a member of the Ubuntu Community, in any forum, mailing list, wiki, Web site, IRC channel, install-fest, public meeting, or private correspondence." The CoC goes into some degree of depth on a series of points that fall under the following headings:

- Be considerate.
- Be respectful.
- Be collaborative.
- When you disagree, consult others.
- When you are unsure, ask for help.
- Step down considerately.

Many of these headings seem like common sense or common courtesy to many, and that is by design. Nothing in the CoC is controversial or radical, and it was never designed to be.

More difficult is that nothing is easy to enforce or decide because acting considerately, respectfully, and collaboratively is often very subjective. There is room for honest disagreements and hurt feelings. These are accepted shortcomings. The CoC was not designed to be a law with explicit prohibitions on phrases, language, or actions. Instead, it aims to provide a constitution and a reminder that considerate and respectful discussion is *essential* to the health and vitality of the project. In situations where there is a serious disagreement on whether a community member has violated or is violating the code, the Community Council is available to arbitrate disputes and decide what action, if any, is appropriate.

Nobody involved in the Ubuntu project, including Mark Shuttleworth and the other members of the Community Council, is above the CoC. The CoC is never optional and never waived. In fact, the Ubuntu community recently created a Leadership Code of Conduct (LCoC), which extends and expands on the CoC and describes additional requirements and expectations for those in leadership positions in the community. Of course, in no way was either code designed to eliminate conflict or disagreement. Arguments are at least as common in Ubuntu as they are in other projects and online communities. However, there is a common understanding within the project that arguments should happen in an environment of collaboration and mutual respect. This allows for *better* arguments with *better* results—and with less hurt feelings and fewer bruised egos.

While they are sometimes incorrectly used as such, the CoC and LCoC are not sticks to be wielded against an opponent in an argument. Instead, they are useful points of reference upon which consensus can be assumed within the Ubuntu community. Frequently, if a group in the community feels a member is acting in a way that is out of line with the code, the group will gently remind the community member, often privately, that the CoC is in effect. In almost all situations, this is enough to avoid any further action or conflict. Very few CoC violations are ever brought before the Community Council.

Technical Goals

While a respectful community and adherence to a set of philosophical goals provide an important frame in which the Ubuntu project works, Ubuntu is, at the end of the day, a technical project. As a result, it only makes sense that in addition to philosophical goals and a project constitution, Ubuntu also has a set of technical goals.

The first technical goal of the project, and perhaps the most important one, is the coordination of regular and predictable releases—something particularly important to server users. In April 2004, at the Warthogs meeting, the project set a goal for its initial proof-of-concept release six months out. In part due to the resounding success of that project, and in larger part due to the GNOME release schedule, the team has stuck to a regular and predictable six-month release cycle and has only once chosen

to extend the release schedule by six weeks and only after obtaining community consensus on the decision. The team then redoubled its efforts and made the next release in a mere four and a half months, putting its release schedule back on track. Frequent releases are important because users can then use the latest and greatest free software available—something that is essential in a development environment as vibrant and rapidly changing and improving as the free software community. Predictable releases are important, especially to businesses, because predictability means that they can organize their business plans around Ubuntu. Through consistent releases, Ubuntu can provide a platform upon which businesses and derivative distributions can rely to grow and build.

While releasing frequently and reliably is important, the released software must then be supported. Ubuntu, like all distributions, must deal with the fact that *all* software has bugs. Most bugs are minor, but fixing them may introduce even worse issues. Therefore, fixing bugs after a release must be done carefully or not at all. The Ubuntu project engages in major changes, including bug fixes, *between* releases only when the changes can be extensively tested. However, some bugs risk the loss of users' information or pose a serious security vulnerability. These bugs are fixed immediately and made available as updates for the released distribution. The Ubuntu community works hard to find and minimize all types of bugs before releases and is largely successful in squashing the worst. However, because there is always the possibility that more of these bugs will be found, Ubuntu commits to supporting *every* release for 18 months after it is released. In the case of Ubuntu 6.06 LTS (Dapper Drake), released in 2006, the project went well beyond even this and committed to support the release for three full years on desktop computers and for five years in a server configuration (LTS stands for LongTerm Support). This proved so popular with businesses, institutions, and the users of Ubuntu servers that Ubuntu 8.04 (Hardy Heron) was named as Ubuntu's second LTS release with similar three- and five-year desktop and server extended support commitments. These five-year support commitments are specifically designed for server users and make Ubuntu a much more attractive option for an important class of server users.

This bipartite approach to servers and desktops implies the third major technical commitment of the Ubuntu project and, in a sense, the most

important for this book: support for both servers and desktop computers in separate but equally emphasized modes. While Ubuntu continues to be more well known, and perhaps more popular, in desktop configurations, there exist teams of Ubuntu developers focused on both server and desktop users. The Ubuntu project believes that both desktops and servers are essential and provides installation methods on every CD for both types of systems. Ubuntu provides tested and supported software appropriate to the most common actions in both environments and documentation for each. LTS releases in particular mark an important step toward catering to users on the server.

Finally, the Ubuntu project is committed to making it as easy as possible for users to transcend their roles as consumers and users of software and to take advantage of each of the freedoms central to the Ubuntu philosophy. As a result, Ubuntu has tried to focus its development around the use and promotion of a single programming language, Python. The project has worked to ensure that Python is widely used throughout the system. By ensuring that many applications and many of the "guts" of the system are written in or extensible in Python, Ubuntu is working to ensure that users need to learn only one language in order to take advantage of, automate, and tweak many parts of their systems.

Canonical and the Ubuntu Foundation

While Ubuntu is driven by a community, several groups play an important role in its structure and organization. Foremost among these are Canonical, Ltd., a for-profit company introduced as part of the Ubuntu history description, and the Ubuntu Foundation, which is introduced later in this section.

Canonical, Ltd.

As mentioned earlier, Canonical, Ltd., is a company founded by Mark Shuttleworth with the primary goal of developing and supporting the Ubuntu distribution. Many of the core developers on Ubuntu—although no longer a majority of them—work full-time or part-time under contract for Canonical, Ltd. This funding by Canonical allows Ubuntu to make the type of support commitments that it does. Ubuntu can claim that it will

release in six months because releasing, in one form or another, is something that the paid workers at Canonical can ensure. As an all-volunteer organization, Debian suffered from an inability to set and meet deadlines—volunteers become busy or have other deadlines in their paying jobs that take precedence. By offering paying jobs to a subset of developers, Canonical can set support and release deadlines and ensure that they are met.

In this way, Canonical ensures that Ubuntu's bottom-line commitments are kept. Of course, Canonical does not fund all Ubuntu work, nor could it. Canonical can release a distribution every six months, but that distribution will be made *much* better and more usable through contributions from the community of users. Most features, most new pieces of software, almost all translations, almost all documentation, and much more are created outside of Canonical. Instead, Canonical ensures that deadlines are met and that the essential work, regardless of whether it's fun, gets done.

Canonical, Ltd., was incorporated on the Isle of Man—a tiny island nation between Wales and Ireland that is mostly known as a haven for international businesses. Since Canonical's staff is sprinkled across the globe and no proper office is necessary, the Isle of Man seemed as good a place as any for the company to hang its sign.

Canonical's Service and Support

While it is surprising to many users, fewer than half of Canonical's employees work on the Ubuntu project. The rest of the employees fall into several categories: business development, support and administration, and development on the Bazaar and Launchpad projects.

Individuals involved in business development help create strategic deals and certification programs with other companies—primarily around Ubuntu. In large part, these are things that the community is either ill suited for or uninterested in as a whole. One example of business development work is the process of working with companies to ensure that their software (usually proprietary) is built and certified to run on Ubuntu. For example, Canonical worked with IBM to ensure that its popular DB2 database would run on Ubuntu and, when this was achieved, worked to have

Ubuntu certified as a platform that would run DB2. Similarly, Canonical worked with Dell to ensure that Ubuntu could be installed and supported on Dell laptops as an option for its customers. A third example is the production of this book, which, published by Pearson Education's Prentice Hall imprint, was a product of work with Canonical.

Canonical also plays an important support role in the Ubuntu project in three ways. First, Canonical supports the development of the Ubuntu project. For example, Canonical system administrators ensure that the servers that support development and distribution of Ubuntu are running. Second, Canonical helps Ubuntu users and businesses directly by offering phone and e-mail support. Additionally, Canonical has helped build a large commercial Ubuntu support operation by arranging for support contracts with larger companies and organizations. This support is over and above the free (i.e., gratis) support offered by the community—this commercial support is offered at a fee and is either part of a longer-term flat-fee support contract or is pay-per-instance. By offering commercial support for Ubuntu in a variety of ways, Canonical aims to make a business for itself and to help make Ubuntu a more palatable option for the businesses, large and small, that are looking for an enterprise or enterprise-class GNU/Linux product with support contracts like those offered by other commercial GNU/Linux distributions.

Finally, Ubuntu supports other support organizations. Canonical does not seek or try to enforce a monopoly on Ubuntu support; it proudly lists *hundreds* of other organizations offering support for Ubuntu on the Ubuntu Web pages. Instead, Canonical offers what is called second-tier support to these organizations. Because Canonical employs many of the core Ubuntu developers, the company is very well suited to taking action on many of the tougher problems that these support organizations may run into. With its concentrated expertise, Canonical can offer this type of backup, or secondary, support to these organizations.

The Ubuntu Foundation

Finally, in addition to Canonical and the full Ubuntu community, the Ubuntu project is supported by the Ubuntu Foundation, which was announced by Shuttleworth with an initial funding commitment of $10 million. The foun-

dation, like Canonical, is based on the Isle of Man. The organization is advised by the Ubuntu Community Council.

Unlike Canonical, the foundation does not play an active role in the day-to-day life of Ubuntu. At the moment, the foundation is little more than a pile of money that exists to endow and ensure Ubuntu's future. Because Canonical is a young company, many companies and individuals find it difficult to trust that Canonical will be able to provide support for Ubuntu in the time frames (e.g., three to five years) that it claims it will be able to. The Ubuntu Foundation exists to allay those fears.

If something bad were to happen to Shuttleworth or to Canonical that caused either to be unable to support Ubuntu development and maintain the distribution, the Ubuntu Foundation exists to carry on many of Canonical's core activities well into the future. Through the existence of the foundation, the Ubuntu project can make the types of long-term commitments and promises it does.

The one activity in which the foundation can and does engage is receiving donations on behalf of the Ubuntu project. These donations, and only these donations, are then put to use on behalf of Ubuntu in accordance with the wishes of the development team and the Technical Board. For the most part, these contributions are spent on "bounties" given to community members who have achieved important feature goals for the Ubuntu project.

History of Ubuntu Server

The first "production" machines to run Ubuntu were Canonical's own development machines in its data center in London. In this sense, Ubuntu has been used on servers since day one, and Ubuntu has *always* been a server operating system. Of course, as we hinted in the welcome at the beginning of this Introduction, this has not always been universally recognized. After the first release, public perception was tilted so far toward the idea of Ubuntu as a desktop release that when the developers convened the first of their biannual developer summits after the first full release cycle, one of the most important items on the agenda was thinking about Ubuntu on servers and how to support it.

The Ubuntu Server project, as a result, was at least as much a marketing project as it was a technical project. Sure, there were ways that the team could make Ubuntu better for servers—and they spent plenty of time working and thinking about that—but the biggest problem they faced was simply communicating the message that Ubuntu already was great for servers to all their users and potential users.

Eventually Canonical funded the creation of a graphical installer, but in the first few releases there was just a single, nongraphical installer based on Debian's very descriptively named Debian Installer project. In the initial Ubuntu release, a user installing Ubuntu was given a choice between two modes: "Desktop"—which was self-explanatory enough—and "Custom." Custom, in the minds of the early developers, was what anyone would want for a server. Custom installed just the bare minimum set of packages and then put the users into this base install and prompted them to install the packages that they wanted on their system. It provided users with the bare-bones system and encouraged them to customize it. The first action of the Ubuntu Server project was purely superficial: The "Custom" install was renamed "Server." Although no code had changed, Ubuntu Server almost immediately began getting more recognition. If one had to pick a single point in time that the Ubuntu Server project was born, it would be this moment.

Ubuntu Server isn't actually any different from other flavors of Ubuntu. As the desktop has moved on to a new graphical installer based on a live CD, Ubuntu Server has its own installer that gives users access to features like RAID and LVM that are much more interesting to server users. Certainly, there are some pieces of software that are likely to end up on servers and unlikely to end up on desktops—things like Web servers and mail servers. When we say that the server edition will be supported, we mean these applications plus the core, so it certainly seems most accurate to refer to these as being within the purview of Ubuntu Server.

But at the end of the day, the server and desktop packages come out of a single repository. This fact, plus the integration among the teams of people working on different parts of the project—most core developers work on bits and pieces that get used and reused in server, desktop, and other editions—introduces a fuzziness that makes it hard to pin down just what

Ubuntu Server *is*. Of course, it also means that Ubuntu Server gets to benefit from the work, bug reporting, and bug fixing in those core parts of the operating system that every Ubuntu user shares.

Ubuntu Server now can roughly be interpreted to refer to a collection of resources that are particularly aimed at and used by server users. Most obviously, it involves the custom install discs that you'll be using when you install Ubuntu Server on your machine. It also refers to the collections of supported software that are installed primarily on servers—most of the software that the rest of this book will discuss in more detail. It also refers to a mass of documentation, to which this book represents the latest addition, that provides answers to questions. In a broader sense, certifications of software and training programs for administrators occupy another point in the growing Ubuntu Server constellation.

But most of all, and in the Ubuntu tradition, Ubuntu Server refers to a community. It's a community of developers who use Ubuntu on servers, who care deeply about Ubuntu on servers, and who work tirelessly to make sure that Ubuntu performs as well as possible on servers everywhere. Of course, Ubuntu Server also refers to the growing community of people who are primarily not contributing through code but who are at least as important. These people spend time in the support of IRC channels, send e-mail to the mailing lists, and post in the forums. These users help other users, file bugs, may contribute their own fixes to documentation, and contribute in myriad ways and in a variety of venues.

When you "graduate" beyond what this book can teach you, Ubuntu represents those people who will help you take your next steps. They are the people described in more depth in the server resources chapter (Chapter 13) of this book. This is the group you will join when you participate in the Ubuntu project. Let us be the first to welcome to you to the Ubuntu Server community.

Simple, Secure, Supported

Early on, the initial core Ubuntu team—of which one of this book's authors was lucky enough to be a part—resisted the idea of the server version of Ubuntu. Or rather, they resisted the idea of a server distribution in

the way that other GNU/Linux distributions had produced them and the way in which they were commonly thought of. The team was more than happy with running Ubuntu on servers, of course, but they resisted the idea of "server distributions" because of the way that Red Hat, SuSE, and the other big distributions built their businesses around "enterprise Linux" distributions that were big, clunky, and expensive. The result was, in the eyes of many of the early Ubuntu core developers and Canonical employees, top-heavy monstrosities. That's not what Ubuntu is about.

The big server-based GNU/Linux distributions seemed to be competing over who included more services, more features, and more bells and whistles. Distribution 1 would have a Web server, an FTP server, a DNS server, several file servers, and a mail server. Distribution 2 would have all of those plus a DHCP server! A brand-new install of one of these "server distributions" would be running dozens of daemons—each taking up many megabytes of memory, loads of disk space, and (most important) lots of administrator time when they failed or interfered with something else. But worst of all, most of these daemons lay completely unused on most installs.

And if that wasn't enough, the server installs would then run firewalls to keep people from accessing all these now-open services and to prevent users from exposing security vulnerabilities from their newly installed machines. Of course, there would be regular upgrades, security releases, and the like, to update all these now-firewalled services that nobody was using. Debian provided one alternative model that focused on custom installations of just what people needed. Among an elite group of sysadmins in the late 1990s and the early 2000s, Debian had become the server OS of choice. Because nearly everyone on the early Ubuntu team was a Debian developer, it was to this model and to Debian technology that the Ubuntu team first turned.

Of course, the commercial GNU/Linux server market was not all horrible. For example, the early Ubuntu developers liked the idea of commercial support for its servers. They liked the idea of regular, predictable releases. As Debian developers, they all knew someone who wanted to install a simple, custom version of Debian on a server but who, because of the lack of commercial support and accountability, had been rejected by a higher-

up in the company or organization. They liked the idea of a company using Debian's technology to offer simple, custom server installs but could offer a commercial support contract. The Warthogs, and lots of folks like them, had waited years for this, but nobody had stepped up to the plate.

As we described in the previous section, an Ubuntu server install was simply a bare-bones installation. We were all administrators—at least of our own machines—and when *we* installed servers, we started out with "naked" machines so that we could choose every application, every daemon, every service that would go onto the machine. As administrators, we wanted the *options* of the big enterprise distributions, but we wanted to be able to choose those options ourselves. Like all administrators, we used servers to solve problems and to offer services to our users. These problems and needs are unique and, as a result, the cookie-cutter model of GNU/Linux servers was always a poor match.

And so that is what the Warthogs built and it is what Ubuntu Server remains today. At first, some people were confused. Ubuntu's server offering was panned in several reviews for not including a firewall by default. But Ubuntu installed *no open ports by default*, so there was nothing to firewall! Of course, Ubuntu provided several firewalls for users to install *if they wanted one*, but Ubuntu left the decision to install a firewall, just like the decision to install services that might require one, up to the server's administrator. For all installations but for server installations in particular, Ubuntu's goal is to make the default installation simple and secure and to put the user in the driver's seat. Ubuntu's job, as distribution producer, is to make it as close to drop-dead simple for system administrators to do their jobs. In an Ubuntu Server install, every machine is exactly as complicated as the administrator has requested but never any more than necessary. No extra services or unnecessary features are included—although they are waiting in the wings for when they become necessary and are easily installable in ways that are described in Chapter 3.

One important effect of this simplicity is security. When there is less going on, there is simply less to go wrong. But, of course, the Ubuntu team has taken this many steps further and pursued proactive security in a number of other contexts. Ubuntu's first release was held up for one day because a single open port was found in the default release. The goal of a machine with

no open ports by default was more important than an on-time release. Ubuntu's CTO and the chairman of the Ubuntu technical board, Matt Zimmerman, is a longtime security-focused developer who made nearly all of Debian's security updates for more than a year before joining the Warthogs. As Ubuntu struggles over hard decisions about what to include or to pass up for inclusion in the distribution, the most important questions continue to be ones of security and support. "Can we—and we do want to—maintain security support and provide security releases for this software for the next 18 months?" Every piece of software included by default is subjected to this question, and many popular pieces of software are kept out because Ubuntu is reluctant to support them. Inclusion as an officially supported package means that a server admin can trust the software—both because Canonical has indicated that it trusts it and because Canonical has promised to clean up any security messes that occur through fixing important bugs and issuing a fixed package. Canonical's security guarantee goes beyond security bugs to other bugs that might result in data loss. While there are no guarantees beyond this, Canonical makes many dozens of new updates per release that fix other important bugs in the distribution as well. The result is a rock-solid system with a commitment to continue.

With customizability, security, and support, Ubuntu truly is ready for the data room. The rest of this book will show you how.

CHAPTER 1

Installation

I REMEMBER WHEN IT WAS QUITE the ordeal to install Linux. You had to download a complete set of floppy images and use some strange Linux or Windows tool to dump those images to floppies. After you had your set of floppies and started the installation process, you would quickly realize that the installation program assumed you already knew quite a bit about Linux and about computers in general. I am actively involved in a Linux users' group, and a staple of Linux users' groups is the install-fest—an event to which Linux newbies can bring their computers and have an expert walk them through the installation process.

Well, these days Linux distributions have made great strides to improve the install process. A desktop Ubuntu install asks very few questions that might stump a beginning user, and the server install (unlike some other server installs out there) doesn't assume you are an experienced Linux system administrator either. While the server install is pretty easy to navigate when you just accept the defaults, you'll find that it also allows the advanced administrator a lot of flexibility and control. What's better, you can get this control without toggling an "expert mode" that forces you to answer detailed questions about every aspect of the install. When the default suits you, you can accept it and move on, and when it doesn't, you can easily tweak only those settings you care about.

This chapter walks you through a standard Ubuntu install. Along the way, it covers each of the major parts of the installation process and also highlights areas where the experienced administrator can tweak and tune the settings for a custom server install.

Get Ubuntu

If you have this book, you should already have the Ubuntu install CD for Ubuntu 12.04 LTS, but just in case you don't, or if you need an extra copy, it's good to know how to get CDs from scratch. With some distributions you are required to register and log in to a Web site before you have access to CD images, but Ubuntu makes it relatively easy. At www.ubuntu.com/getubuntu/download you can access both desktop and server CDs for each of the platforms Ubuntu supports. Once you have downloaded the ISO image, you can burn it to a CD using your preferred CD-burning tool. Each program is different, so locate your program's option to burn a CD image.

If you are already running an Ubuntu desktop, you can just double-click the ISO image after it is downloaded to start the CD/DVD Creator. If you don't have the bandwidth to download the ISO image, you can also purchase CDs or request free CDs from the same page.

Boot Screen

Once you have your CD, insert it into your server's CD-ROM slot and boot the server. Server BIOSs can be quite different from each other. In some cases they are already set up to boot from a CD-ROM if one is present. In other cases you might have to hit a key such as Esc, Del, F12, or others as the server boots so that it presents you with a list of devices from which you can boot. On some systems you might even have to go into the BIOS configuration page to change the boot order.

After you have convinced your BIOS to boot from the CD, you are greeted with the Ubuntu boot splash screen. The very first part of the screen asks you for your language and then presents you with some boot options, as shown in Figure 1-1. If you just hit Enter and select the default option, you will start the Ubuntu install program. You can also perform some diagnostics

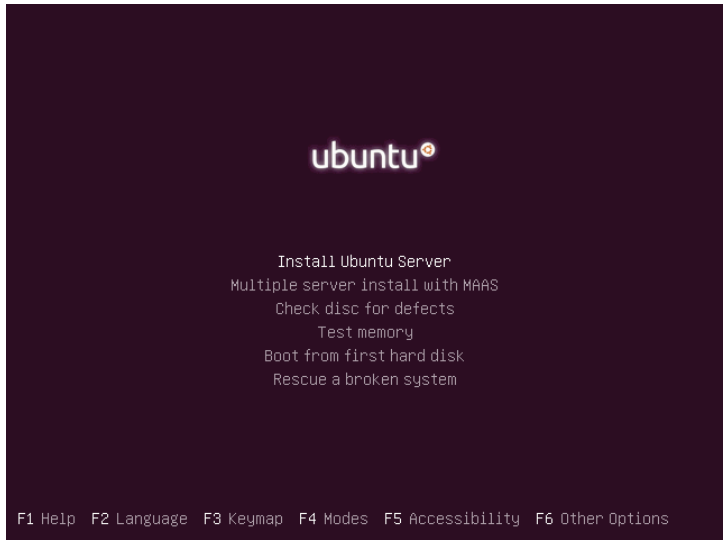

Figure 1-1 Default Ubuntu boot splash screen

from this screen. If you downloaded and burned your own CD from an ISO file, you might want to select the "Check CD for defects" option. You can also test your system memory from this screen—quite useful even later on if your server starts crashing at strange times and you suspect bad RAM might be the cause. This CD also doubles as a basic rescue disk, which is a broad enough topic that it has gotten its own chapter: Chapter 12, Rescue and Recovery. Finally, if you left the CD in the drive by mistake, you can select "Boot from first hard disk" to bypass the CD altogether.

In an ideal world, you could boot the CD, press Enter, and start the installation process with no problems. Unfortunately, for some machines you might have to tweak some of the boot options so the installation works for your system. The Ubuntu developers know this sort of thing happens and have prepared a rich set of options for you. Along the bottom of the boot splash screen are a number of different options you can access with function keys. For instance, if you hit F1 you will see an interactive help screen with documentation for the rest of the options. If you accidentally chose the wrong language at boot time, hit F2 to change it. The boot screen will automatically choose a keyboard mapping based on your language. If you want a different mapping (for instance, you speak English but are doing an install in Germany on a German keyboard), hit F3 to choose from a list of keyboard mappings. The boot screen also has a lot of great accessibility options. The F5 key brings up an accessibility menu that allows you to choose a high-contrast screen, a screen magnifier, a screen reader and Braille terminal, keyboard modifiers, and even an on-screen keyboard.

The F4 and F6 options allow you to actually control aspects of the install. The F4 key displays a list of install modes from which you can choose: you may install an original equipment manufacturer (OEM) install, a minimal system, and a minimal virtualization guest. The OEM install is available from manufacturers. The minimal virtualization guest gives you an easy way to install a virtualized version of Ubuntu. Other Ubuntu CDs allow you to choose text-only install modes, but that isn't necessary here because the Ubuntu server install is already text-only. The real power and control over the boot process are available once you hit the F6 key. Here you can see a menu of common arguments that help the CD to boot on difficult hardware. If you hit the Esc key, you will move from this menu to the boot prompt and can type any extra kernel boot parameters you might need for

your hardware. The F1 help screen lists a number of common boot parameters, including some kernel arguments for particular SCSI controllers. If you still can't seem to get the Ubuntu CD to boot after trying these arguments, check out some of the support options in Chapter 13, Help and Resources. Chances are, you aren't the first person to try to boot Ubuntu on that hardware, and someone else might have already discovered the magic list of options you need for it to work.

NOTE **For Headless Server Installation**
One downside to the pretty boot splash screen Ubuntu uses is that if you install Ubuntu on a headless server (a server without a monitor connected to it that outputs its display over the serial port), you'll notice that you can't see anything over your serial console. There is a workaround. After your system boots past the BIOS, the serial console screen will go completely blank. At that point you will have to type without being able to see the output. First hit Enter to accept the default language. Then hit F6 so you can tweak the boot prompt, and hit Esc to exit out of the F6 menu into the boot prompt. Finally, type `console=ttyS0,9600n8` and hit Enter. If your serial console is on a different port or uses different settings, you will want to change this argument accordingly. Since you can't see what you are typing, you will probably want to pay extra attention to each key. Once you hit Enter, the kernel will boot and within a few seconds you should start to see output over the serial port. At that point you will be able to complete the install over the serial console, and the next time you reboot you'll notice that the GRUB (Grand Unified Bootloader) prompt and the kernel arguments are already set up for you.

Disk Partitioning

Any Linux installer is essentially a series of questions, and the Ubuntu server install asks pretty simple questions for the most part. The first phase of the installation prompts you with questions about what language to use for the install, what hostname to use, and, if you don't have Dynamic Host Configuration protocol (DHCP) on your network, the network settings for the host. There aren't any real tricky decisions to be made until you get to the partitioning section of the installer.

What Is a Partition?

If you are relatively new to Linux, you might be wondering what a partition is, and why Linux is prompting you to partition in the first place. Think of a hard drive as a house with an open floor plan—no rooms and only outside walls. While many people do live in lofts that are organized this way, most people prefer a house that has rooms. With rooms you can organize your bed and all of your personal things in a bedroom, all of your