Chapter 1

Introduction to Linux

Chapter Objectives

1 Explain the purpose of an operating system.

2 Outline the key features of the Linux operating system.

3 Describe the origins of the Linux operating system.

4 Identify the characteristics of various Linux distributions and where to find them.

5 Explain the common uses of Linux in industry today.

6 Describe how Linux is used in the cloud.

Linux technical expertise is essential in today's computer workplace as more and more companies switch to Linux to meet their computing needs. Thus, it is important to understand how Linux can be used, what benefits Linux offers to a company, and how Linux has developed and continues to develop. In the first half of this chapter, you learn about operating system terminology and features of the Linux operating system, as well as the history and development of Linux. Later in this chapter, you learn about the various types of Linux, as well as the situations and environments in which Linux is used. Finally, you explore the ways that Linux can be hosted in the cloud, as well as the process and technologies used to add web apps to cloud-based Linux systems.

Operating Systems

Every computer has two fundamental types of components: hardware and software. You are probably familiar with these terms, but it's helpful to review their meanings so you can more easily understand how Linux helps them work together.

Hardware consists of the physical components inside a computer that are electrical in nature; they contain a series of circuits that manipulate the flow of information. A computer can contain many different pieces of hardware, including the following:

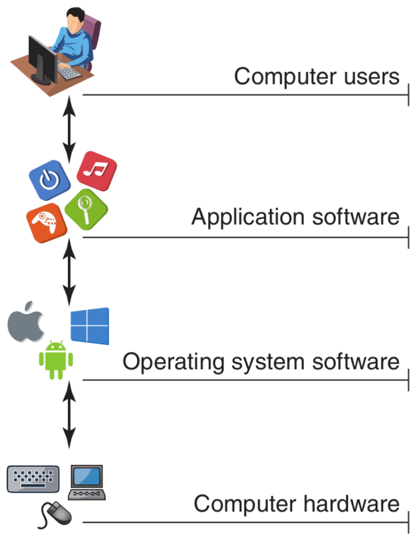
- A processor (also known as the central processing unit or CPU), which computes information
- Physical memory (also known as random access memory or RAM), which stores information needed by the processor
- Hard disk drives (HDDs) and solid state drives (SSDs), which store most of the information that you use
- CD/DVD drives, which read and write information to and from CD/DVD discs

- Flash memory card readers, which read and write information to and from removable memory cards, such as Secure Digital (SD) cards
- Sound cards, which provide audio to external speakers
- Video cards (also known as graphics processing units or GPUs), which display results to the computer monitor
- Network adapter cards, which provide access to wired and wireless (Wi-Fi or Bluetooth) networks
- Ports (such as USB, eSATA, GPIO, and Thunderbolt), which provide access to a wide variety of external devices including keyboards, mice, printers, and storage devices
- Mainboards (also known as motherboards), which provide the circuitry (also known as a bus) for interconnecting all other components

Software, on the other hand, refers to the sets of instructions or **programs** that allow the hardware components to manipulate data (or files). When a bank teller types information into the computer behind the counter at a bank, for example, that bank teller is using a program that understands what to do with your bank records. Programs and data are usually stored on hardware media, such as hard disk drives or SSDs, although they can also be stored on removable media or even embedded in computer chips. These programs are loaded into parts of your computer hardware (such as your computer's memory and processor) when you first turn on your computer and when you start additional software, such as word processors or Internet browsers. After a program is executed on your computer's hardware, that program is referred to as a **process**. In other words, a program is a file stored on your computer, whereas a process is that file in action, performing a certain task.

There are two types of programs. The first type, **applications (apps)**, includes those programs designed for a specific use and with which you commonly interact, such as word processors, computer games, graphical manipulation programs, and computer system utilities. The second type, **operating system** software, consists of a set of software components that control the hardware of your computer. Without an operating system, you would not be able to use your computer. Turning on a computer loads the operating system into computer hardware, which then loads and centrally controls all other application software in the background. At this point, the **user** (the person using the computer) is free to interact with the applications, perhaps by typing on the keyboard or clicking a mouse. Applications take the information the user supplies and relay it to the operating system. The operating system then uses the computer hardware to carry out the requests. The relationship between users, application software, operating system software, and computer hardware is illustrated in Figure 1-1.

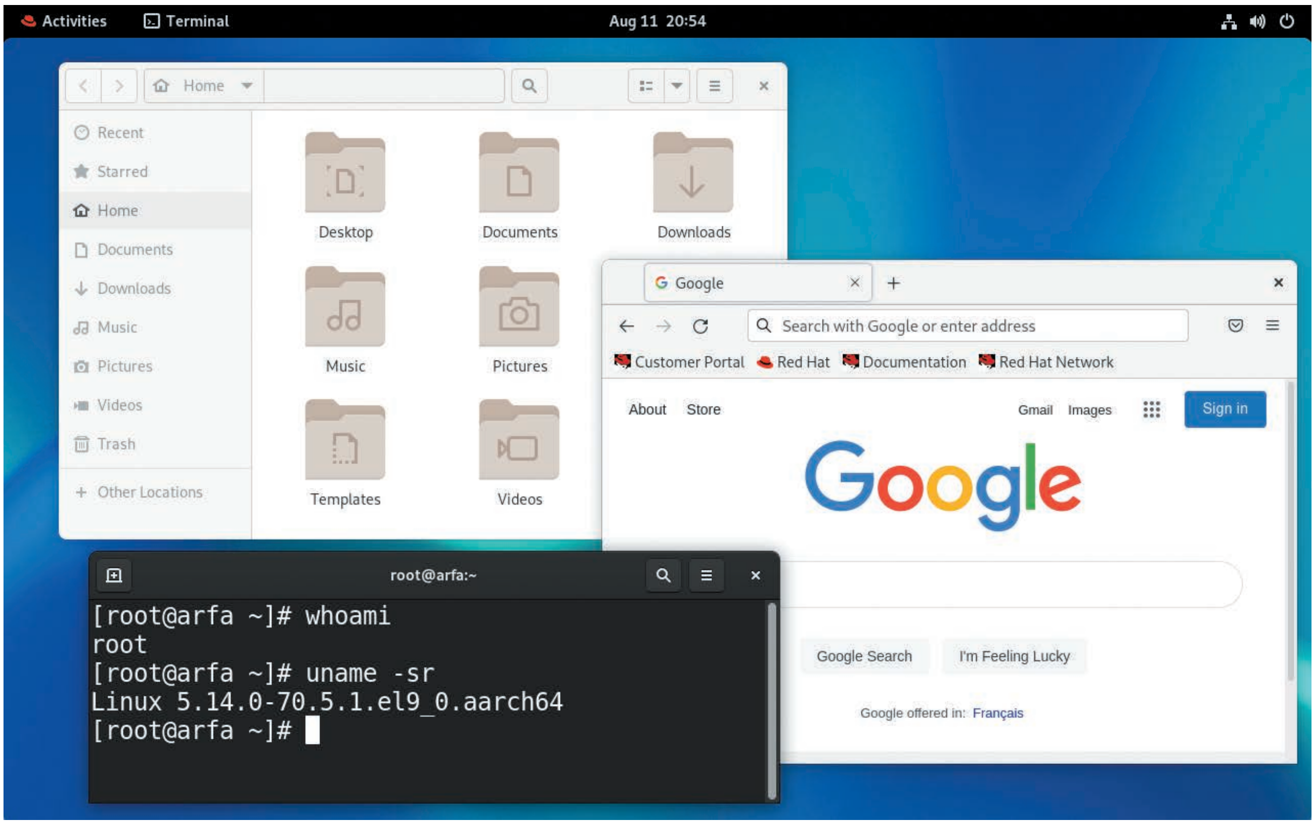
Figure 1-1 The role of operating system software



The operating system carries out many tasks by interacting with different types of computer hardware. For the operating system to accomplish the tasks, it must contain the appropriate device driver software for every hardware device in your computer. Each **device driver** tells the operating system how to use that specific device. The operating system also provides a **user interface**, which is a program that accepts user input indicating what to do, forwards this input to the operating system for completion, and, after it is completed, gives the results back to the user. The user interface can be a command-line prompt, in which the user types commands, or it can be a **graphical user interface (GUI)**, which consists of menus, dialog boxes, and symbols (known as icons) that the user can interact with via the keyboard or the mouse. A typical Linux GUI that also provides a command-line interface via an app is shown in Figure 1-2.

Finally, operating systems offer **services**, which are applications that handle system-related tasks, such as printing, scheduling programs, and network access. These services determine most of the functionality in an operating system. Different operating systems offer different services, and many operating systems allow users to customize the services they offer.

Figure 1-2 A Linux graphical and command-line user interface



```
[root@arfa ~]# whoami
root
[root@arfa ~]# uname -sr
Linux 5.14.0-70.5.1.el9_0.aarch64
[root@arfa ~]#
```

The Linux Operating System

Linux (pronounced "Lih-nucks") is an operating system that is used today to run applications on a variety of hardware. Similar to other operating systems, the Linux operating system loads into computer memory when you first power on your computer and initializes (or activates) all of the hardware components. Next, it loads the programs that display the interface. From within the interface, you can execute commands that tell the operating system and other applications to perform specific tasks. The operating system then uses the computer hardware to perform the tasks required by the applications.

Versions of the Linux Operating System

The core component of the Linux operating system is called the Linux **kernel**. The Linux kernel and supporting software (called function libraries) are written almost entirely in the C programming language, which is one of the most common languages that software developers use when creating programs.

Although a variety of software can be used to modify the appearance of Linux, the underlying kernel is common to all types of Linux. The Linux kernel is developed continuously; thus, you should understand the different version numbers of the Linux kernel to decide which kernel version is appropriate for your needs. Because the Linux kernel is directly responsible for controlling the computer's hardware (via device drivers), you might sometimes need to upgrade the Linux kernel after installing Linux to take advantage of new technologies or to fix problems (also known as bugs) related to your computer's hardware. Consequently, a good understanding of your system's hardware is important in deciding which kernel to use.

Note 1

For a complete list of kernels, kernel versions, and their improvements, see kernel.org.

In some cases, you can use updates in the form of a kernel module or a kernel patch to provide or fix hardware supported by the kernel. Kernel modules and kernel patches are discussed later in this book.

Identifying Kernel Versions

Linux kernel versions are made up of the following three components:

- Major number
- Minor number
- Revision number

Let's look at a sample Linux kernel version, 5.18.16. In this example, the **major number** is the number 5, which indicates the major version of the Linux kernel. The **minor number**, represented by the number 18, indicates the minor revision of the Linux kernel. As new features are added to the Linux kernel over time, the minor number is incremented. The major number is usually incremented when a major kernel feature is implemented, when the minor number versioning reaches a high number, or to signify a major event; for example, the 3.0 kernel was introduced to commemorate the twentieth anniversary of Linux.

Linux kernel changes occur frequently. Those changes that are very minor are represented by a **revision number** indicating the most current changes to the version of the particular kernel that is being released. For example, a 5.18.16 kernel has a revision number of 16. This kernel is the sixteenth release of the 5.18 kernel. Some kernels have over 100 revisions as a result of developers making constant improvements to the kernel code.

Note 2

Sometimes, a fourth number is added to a kernel version to indicate a critical security or bug patch. For example, a 5.18.16.1 kernel is a 5.18.16 kernel with a critical patch number of 1.

Modern Linux kernels that have a major, minor, and revision number are referred to as **production kernels**; they have been thoroughly tested by several Linux developers and are declared stable. **Development kernels** are not fully tested and imply instability; they are tested for vulnerabilities by people who develop Linux software. Most development kernels append the minor number with the letters -rc (release candidate) followed by a number that represents the version of the development kernel. For example, the 5.19-rc8 development kernel is the eighth release candidate for the 5.19 kernel; if Linux developers declare it stable after being thoroughly tested, it will become the 5.19.0 production kernel.

Note 3

Until Linux kernel 2.6.0, an odd-numbered minor number was used to denote a development kernel, and an even-numbered minor number was used to denote a production kernel.

Note 4

When choosing a kernel for a mission-critical computer such as a server, ensure that you choose a production kernel. This reduces the chance that you will encounter a bug in the kernel, which saves you the time needed to change kernels.

Table 1-1 shows some sample kernel versions released since the initial release of Linux.

Table 1-1 Sample Linux Kernel version history

Kernel Version	Date Released	Type
0.01	September 1991	First Linux kernel
0.12	January 1992	Production (stable)
0.99.15	March 1994	Development
1.0.8	April 1994	Production (stable)
1.3.100	May 1996	Development
2.0.36	November 1998	Production (stable)
2.3.99	May 2000	Development
2.4.17	December 2001	Production (stable)
2.5.75	July 2003	Development
2.6.35	August 2010	Production (stable)
3.0.0	July 2011	Production (stable)
3.15.10	August 2014	Production (stable)
4.0.0	April 2015	Production (stable)
4.12.5	August 2017	Production (stable)
5.0.0	March 2019	Production (stable)
5.19-rc8	July 2022	Development

Licensing Linux

Companies often choose Linux as their operating system because of the rules governing Linux licensing. Unlike most other operating systems, Linux is freely developed and continuously improved by a large community of software developers. For this reason, it is referred to as **open source software (OSS)**.

To understand OSS, you must first understand how source code is used to create programs. **Source code** refers to the list of instructions that a software developer writes to make up a program; an example of source code is shown in Figure 1-3.

Figure 1-3 Source code

```
#define MODULE
#include <linux/module.h>
int init_module(void){
        printk("My module has been activated.\n");
        return 0;
}
void cleanup_module(void){
        printk("My module has been deactivated.");
}
```

After the software developer finishes writing the instructions, the source code is compiled into a format that your computer's processor can understand and execute directly or via an interpreter program. To edit an existing program, the software developer must edit the source code and then recompile it.

The format and structure of source code follows certain rules defined by the **programming language** in which it was written. Programmers write Linux source code in many programming languages. After being compiled, all programs look the same to the computer operating system, regardless of the programming language in which they were written. As a result, software developers choose a programming language to create source code based on ease of use, functionality, and comfort level.

The fact that Linux is an open source operating system means that software developers can read other developers' source code, modify that source code to make the software better, and redistribute that source code to other developers who might improve it further. Like all OSS, Linux source code must be distributed free of charge, regardless of the number of modifications made to it. People who develop OSS commonly use the Internet to share their source code, manage software projects, and submit comments and fixes for bugs (flaws). In this way, the Internet acts as the glue that binds together Linux developers in particular and OSS developers in general.

Note 5

To read the complete open source definition, visit opensource.org.

Note 6

OSS is also called free and open source software (FOSS) or free/libre and open source software (FLOSS).

Here are some implications of the OSS way of developing software:

- Software is developed rapidly through widespread collaboration.
- Software bugs (errors) are noted and promptly fixed.
- Software features evolve quickly, based on users' needs.
- The perceived value of the software increases because it is based on usefulness and not on price.

As you can imagine, the ability to share ideas and source code is beneficial to software developers. However, a software company's business model changes drastically when OSS enters the picture. The main issue is this: How can a product that is distributed freely generate revenue? After all, without revenue any company will go out of business.

The OSS process of software development was never intended to generate revenue directly. Its goal was to help people design better software by eliminating many of the problems associated with traditional software development, which is typically driven by predefined corporate plans and rigid schedules. By contrast, OSS development relies on the unsystematic contributions of several software developers that have a direct need for the software that they are developing. While this process may seem haphazard, it ensures that software is constantly developed to solve real-world needs. Because the source code is scrutinized and improved by different developers, OSS often solves problems in the best possible way. Each developer contributes their strengths to a project, while learning new techniques from other developers at the same time.

By leveraging existing OSS, organizations can develop software much faster than they could otherwise. A typical software app today contains dozens or hundreds of well-designed OSS software components that the organization's developers didn't have to create themselves. To support this ecosystem, organization developers contribute improvements to existing OSS, as well as create and maintain OSS software components that they'd like other developers to evolve through contributions. In short, OSS saves companies a large amount of time and money when developing software.

Other companies make money by selling computer hardware that runs OSS, by selling customer support for OSS, or by creating **closed source software** programs that run on open source products such as Linux.

The OSS development process is, of course, not the only way to develop and license software. Table 1-2 summarizes the types of software you are likely to encounter. The following section explains these types in more detail.

Table 1-2 Software types

Type	Description
Open source	Software in which the source code and software can be obtained free of charge and optionally modified to suit a particular need
Closed source	Software in which the source code is not available; although this type of software might be distributed free of charge, it is usually quite costly and commonly referred to as commercial software
Freeware	Closed source software that is given out free of charge; it is sometimes referred to as freemium software
Shareware	Closed source software that is initially given out free of charge but that requires payment after a certain period of use

Types of Open Source Licenses

Linux adheres to the **GNU General Public License (GPL)**, which was developed by the **Free Software Foundation (FSF)**. The GPL stipulates that the source code of any software published under its license must be freely available. If someone modifies that source code, that person must also redistribute that source code freely, thereby keeping the source code free forever.

Note 7

"GNU" stands for "GNUs Not UNIX."

Note 8

The GPL is freely available at gnu.org.

The GPL is an example of a **copyleft license** as it contains strict restrictions on how the source code can be used in derivative software. To encourage adoption in software projects, most OSS instead use a **permissive license** that contains far fewer restrictions. The BSD, MIT, and Apache licenses are examples of permissive open source licenses.

Note 9

For a list of copyleft and permissive open source licenses, visit opensource.org.

Types of Closed Source Licenses

Closed source software can be distributed for free or for a cost; either way, the source code for the software is unavailable from the original developers. The majority of closed source software is sold commercially and bears the label of its manufacturer. Each of these software packages can contain a separate license that restricts free distribution of the program and its source code in many ways.

Note 10

Examples of closed source software are software created by companies such as Microsoft, Apple, and Electronic Arts (EA).

Another type of closed source software is **freeware**, in which the software program is distributed free of charge, yet the source code is unavailable. Freeware might also contain licenses that restrict the distribution of source code. Another approach to this style of closed source licensing is **shareware**, which is distributed free of charge, yet after a certain number of hours of usage or to gain certain features of the program, payment is required. Although freeware and shareware do not commonly distribute their source code under an open source license, some people incorrectly refer to them as OSS, assuming that the source code is freely shared as well.

Linux Advantages

The main operating systems in use today include Linux, Microsoft Windows, UNIX, and macOS. Notably, Linux is the fastest growing operating system released to date. Although Linux was only created in 1991, the number of Linux users estimated by Red Hat in 1998 was 7.5 million, and the number of Linux users estimated by Google in 2010 was over 40 million (including the number of Linux-based Android smartphone and device users). In 2013, LinuxCounter.net estimated that the number of Linux users was over 70 million, and Google estimated that over 900 million Linux-based Android devices had shipped by then. In 2018, Linux ran on the top 500 supercomputers, 60 percent of the computers purchased by schools in North America (due to widespread adoption of Linux-based Chromebooks), as well as nearly all web servers and Internet-connected hardware devices. In 2021, Gartner estimated that nearly half of the global population (3.5 billion people) were Linux users, whether they realized it or not.

Organizations have adopted Linux for many reasons. The following advantages are examined in the sections that follow:

- Risk reduction
- Meeting business needs
- Stability and security
- Support for different hardware
- Ease of customization
- Ease of obtaining support
- Cost reduction

Risk Reduction

Companies need software to perform mission-critical tasks, such as managing business operations and providing valuable services to customers over the Internet. However, changes in customer needs and market competition can cause the software a company uses to change frequently. Keeping the software up to date can be costly and time-consuming but is a risk that companies must take. Imagine that a fictitious company, ABC Inc., buys a piece of software from a fictitious software vendor, ACME Inc., to integrate its sales and accounting information with customers via the Internet. What would happen if ACME went out of business or stopped supporting the software due to lack of sales? In either case, ABC would be using a product that had no software support, and any problems that ABC had with the software after that time would go unsolved and could result in lost revenue. In addition, all closed source software is eventually retired after it is purchased, forcing companies to buy new software every so often to obtain new features and maintain software support.

If ABC instead chose to use an OSS product and the original developers became unavailable to maintain it, then ABC would be free to take the source code, add features to it, and maintain it themselves provided the source code was redistributed free of charge. Also, most OSS does not retire after a short period of time because collaborative open source development results in constant software improvement geared to the needs of the users.

Meeting Business Needs

Recall that Linux is merely one product of open source development. Many thousands of OSS programs are available, and new ones are created daily by software developers worldwide. Most open source Internet tools have been developed for quite some time now, and the focus in the Linux community in the past few years has been on developing application software, cloud technologies, and security-focused network services that run on Linux. Almost all of this software is open source and freely available, compared to other operating systems, in which most software is closed source and costly.

OSS is easy to locate on the web, at sites such as SourceForge (sourceforge.net), GitHub (github.com), GitLab (gitlab.com), and GNU Savannah (savannah.gnu.org). New software is published to these sites daily. SourceForge alone hosts over 500,000 software development projects.

Common software available for Linux includes but is not limited to the following:

- Scientific and engineering software
- Software emulators
- Web servers, web browsers, and e-commerce suites
- Desktop productivity software (e.g., word processors, presentation software, spreadsheets)
- Graphics manipulation software
- Database software
- Security software

In addition, companies that run the UNIX operating system (including macOS, which is a flavor of UNIX) might find it easy to migrate to Linux. For those companies, Linux supports most UNIX commands and standards, which eases a transition to Linux because the company likely would not need to purchase additional software or retrain staff. For example, suppose a company that tests scientific products has spent time and energy developing custom software that runs on the UNIX operating system. If this company transitioned to another operating system, its staff would need to be retrained or hired, and much of the custom software would need to be rewritten and retested, which could result in a loss of customer confidence. If, however, that company transitions to Linux, the staff would require little retraining, and little of the custom software would need to be rewritten and retested, hence saving money and minimizing impact on consumer confidence.

Companies that need to train staff on Linux usage and administration can take advantage of several educational resources and certification exams for various Linux skill levels. Certification benefits as well as the CompTIA Linux+ and LPIC-1 certifications are discussed in this book's Appendix A, "Certification."

In addition, for companies that require a certain development environment or need to support custom software developed in the past, Linux provides support for nearly all programming languages and software development frameworks.

Stability and Security

OSS is developed by people who have a use for it. This collaboration among software developers with a common need speeds up the software creation, and when bugs in the software are found, bug fixes are created quickly. Often, the users who identify the bugs can fix the problem because they have the source code, or they can provide detailed descriptions of their problems so that other developers can fix them.

By contrast, customers using closed source operating systems must rely on the operating system vendor to fix any bugs. Users of closed source operating systems must report the bug to the manufacturer and wait for the manufacturer to develop, test, and release a bug fix. This process might take weeks or even months, which is slow and costly for most companies and individuals. The thorough and collaborative open source approach to testing software and fixing software bugs increases the stability of Linux; it is not uncommon to find a Linux system that has been running continuously for months or even years without being turned off.

Security, a vital concern for most companies and individuals, is another Linux strength. Because Linux source code is freely available and publicly scrutinized, security loopholes are quickly identified and fixed by developers. In contrast, the source code for closed source operating systems is not released to the public for scrutiny, which means customers must rely on the operating system vendor to detect and fix security loopholes. A security loophole unnoticed by the vendor can be exploited by the wrong person. Every day, new malicious software (such as viruses and malware) is unleashed on the Internet with the goal of infiltrating operating systems and other software. However, most malicious software targets closed source operating systems and software. As of April 2008, Linux had fewer than 100 known viruses, whereas Windows had more than 1,000,000 known viruses. Compared to other systems, the amount of malicious software for Linux systems remains incredibly low.

Note 11

For a list of recent malicious software, visit cisecurity.org.

Support for Different Hardware

Another important feature of Linux is that it can run on a wide variety of hardware. Although Linux is most commonly installed on workstations and server systems that use an Intel CPU platform, it can also be installed on supercomputers that use an IBM POWER CPU or high-performance cloud servers that use an ARM CPU, such as Amazon's Graviton EC2. Linux can also be installed on other systems, such as point-of-sale terminals and sales kiosks that use an Intel CPU, as well as small custom hardware devices that use an ARM, MIPS, or RISC-V CPU. Small hardware devices that can connect to the Internet are collectively called the **Internet of Things (IoT)**.

The open source nature of Linux combined with the large number of OSS developers at work today makes Linux an attractive choice for manufacturers of mobile, custom, and IoT devices. NASA spacecrafts, Internet routers and firewalls, Google Android smartphones and tablets, Amazon Kindle eBook readers, GPS navigation systems, smart speakers, home automation equipment, and Wi-Fi access points all run Linux.

Few other operating systems run on more than two CPU platforms, making Linux the ideal choice for companies that use and support many different types of hardware. Here is a partial list of CPU platforms on which Linux can run:

- Intel x86/x86_64 (also implemented by AMD)
- ARM/ARM64
- MIPS/MIPS64
- RISC-V
- PPC (PowerPC, POWER)
- Mainframe (S/390, z/Architecture)

Ease of Customization

The ease of controlling the inner workings of Linux is another attractive feature, particularly for companies that need their operating system to perform specialized functions. If you want to use Linux as a web server, you can recompile the Linux kernel to include only the support needed to be a web server. This results in a much smaller and faster kernel.

Note 12

A small kernel performs faster than a large kernel because it contains less code for the processor to analyze. On high performance systems, you should remove any unnecessary features from the kernel to improve performance.

Today, customizing and recompiling the Linux kernel is a well-documented and easy process; however, it is not the only way to customize Linux. Only software packages necessary to perform certain tasks need to be installed; thus, each Linux system can have a unique configuration and set of applications available to the user. Linux also supports several system programming languages, such as shell and PERL, which you can use to automate tasks or create custom tasks.

Consider a company that needs an application to copy a database file from one computer to another computer, yet also needs to manipulate the database file (perhaps by checking for duplicate records), summarize the file, and finally print it as a report. This might seem like a task that would require expensive software; however, in Linux, you can write a short shell script that uses common Linux commands and programs to perform these tasks. This type of customization is invaluable to companies because it allows them to combine several existing applications to perform a certain task, which might be specific only to that company and, hence, not previously developed by another free software developer. Most Linux configurations present hundreds of small utilities, which, when combined with shell or PERL programming, can make new programs that meet many business needs.

Ease of Obtaining Support

For those who are new to Linux, the Internet offers a world of Linux documentation. A search of the word "Linux" on a typical Internet search engine such as google.com displays thousands of results, including Linux-related guides, information portals, and video tutorials.

In addition, several Internet **forums** allow Linux users to post messages and reply to previously posted messages. If you have a specific problem with Linux, you can post your problem on an Internet forum and receive help from those who know the solution. Linux forums are posted to frequently; thus, you can usually expect a solution to a problem within hours. You can find Linux-related forums on several different websites, including linux.org, linuxquestions.org, facebook.com (called groups), discord.com (called servers), and reddit.com (called subreddits).

Note 13

Appendix C, "Finding Linux Resources on the Internet," describes how to navigate Internet resources and lists some resources that you might find useful.

Although online support is the typical method of getting help, other methods are available, including **Linux User Groups (LUGs)**. LUGs are groups of Linux users who meet regularly to discuss Linux-related issues and problems. An average LUG meeting consists of several new Linux users (also known as Linux newbies), administrators, developers, and experts (also known as Linux gurus). LUG meetings are a resource to solve problems and learn about the local Linux community. Most LUGs host websites that contain a multitude of Linux resources, including summaries of past meetings and discussions. One common activity seen at a LUG meeting is referred to as an Installfest; several members bring in their computer equipment to install Linux and other Linux-related software. This approach to transferring knowledge is very valuable to LUG members because concepts can be demonstrated and the solutions to problems can be modeled by more experienced Linux users.

Note 14

To find a list of available LUGs in your region, search for the words "LUG cityname" on an Internet search engine such as google.com (substituting your city's name for "*cityname*"). When searching for a LUG, keep in mind that LUGs might go by several different names; for example, the LUG in the Kitchener-Waterloo area of Ontario, Canada is known as KW-LUG (Kitchener-Waterloo Linux Users Group). Many LUGs today are managed using custom websites, Facebook groups, or meeting sites such as meetup.com.

Cost Reduction

Linux is less expensive than other operating systems such as Windows because there is no cost associated with acquiring the software. In addition, a wealth of OSS can run on different hardware platforms running Linux, and a large community of developers is available to diagnose and fix bugs in a short period of time for free. While Linux and the Linux source code are distributed freely, implementing Linux is not cost free. Costs include purchasing the computer hardware necessary for the computers hosting Linux, hiring people to install and maintain Linux, and training users of Linux software.

The largest costs associated with Linux are the costs associated with hiring people to maintain the Linux system. However, closed source operating systems have this cost in addition to the cost of the operating system itself. The overall cost of using a particular operating system is known as the **total cost of ownership (TCO)**. Table 1-3 shows an example of the factors involved in calculating the TCO for operating systems.

Table 1-3 Calculating the total cost of ownership

Costs	Linux	Closed Source Operating System
Operating system cost	$0	Greater than $0
Cost of administration	Low: Stability is high and bugs are fixed quickly by open source developers.	Moderate/high: Bug fixes are created by the vendor of the operating system, which could result in costly downtime.
Cost of additional software	Low/none: Most software available for Linux is also open source.	Moderate/high: Most software available for closed source operating systems is also closed source.
Cost of software upgrades	Low/none	Moderate/high: Closed source software is eventually retired, and companies must buy upgrades or new products to gain functionality and stay competitive.

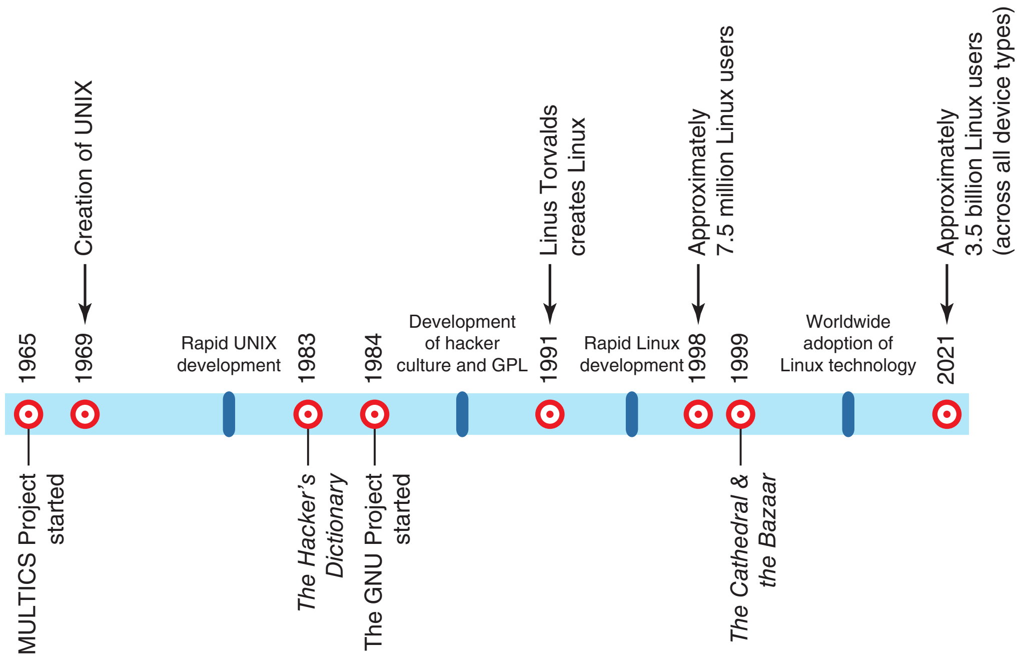
The History of Linux

Linux is based on the UNIX operating system developed by Ken Thompson and Dennis Ritchie of AT&T Bell Laboratories in 1969 and was developed through the efforts of many people as a result of the hacker culture that formed in the 1980s. Therefore, to understand how and why Linux emerged on the operating system market, you must first understand UNIX and the hacker culture. Figure 1-4 illustrates a timeline representing the history of the UNIX and Linux operating systems.

UNIX

The UNIX operating system has roots running back to 1965, when the Massachusetts Institute of Technology (MIT), General Electric, and AT&T Bell Laboratories began developing an operating system called **Multiplexed Information and Computing Service (MULTICS)**. MULTICS was a test project intended to reveal better ways of developing time-sharing operating systems, in which the operating system regulates the amount of time each process has to use the processor. The project was abandoned in 1969. However, Ken Thompson, who had worked on the MULTICS operating system, continued to experiment with operating systems. In 1969, he developed an operating system called **UNIX** that ran on the DEC (Digital Equipment Corporation) PDP-7 computer.

Figure 1-4 Timeline of UNIX and Linux development


1965
MULTICS Project started
1969
Creation of UNIX
Rapid UNIX development
1983
The Hacker's Dictionary
1984
The GNU Project started
Development of hacker culture and GPL
1991
Linus Torvalds creates Linux
Rapid Linux development
1998
Approximately 7.5 million Linux users
1999
The Cathedral & the Bazaar
Worldwide adoption of Linux technology
2021
Approximately 3.5 billion Linux users (across all device types)

Shortly thereafter, Dennis Ritchie invented the C programming language that was used on Ken Thompson's UNIX operating system. The C programming language was a revolutionary language. Most programs at the time needed to be written specifically for the hardware of the computer, which involved referencing volumes of information regarding the hardware in order to write a simple program. However, the C programming language was much easier to use to write programs, and it was possible to run a program on different machines without having to rewrite the code. The UNIX operating system was rewritten in the C programming language, and by the late 1970s, the UNIX operating system ran on different hardware platforms, something that the computing world had never seen until that time. Hence, people called UNIX a portable operating system.

Unfortunately, the company Ken Thompson and Dennis Ritchie worked for (AT&T) was restricted by a federal court order from marketing UNIX. In an attempt to keep UNIX viable, AT&T sold the UNIX source code to several companies, encouraging them to agree to standards among them. Each of these companies developed its own variety, or **flavor**, of UNIX yet adhered to standards agreed upon by all. AT&T also gave free copies of the UNIX source code to certain universities to promote widespread development of UNIX. One result was a UNIX version developed at the University of California, Berkeley in the early 1980s known as BSD (Berkeley Software Distribution). In 1982, one of the companies to whom AT&T sold UNIX source code (Sun Microsystems) marketed UNIX on relatively inexpensive hardware and sold thousands of computers that ran UNIX to companies and universities.

Throughout the 1980s, UNIX found its place primarily in large corporations that had enough money to purchase the expensive computing equipment needed to run UNIX (usually a DEC PDP-11, VAX, or Sun Microsystems computer). A typical UNIX system in the 1980s could cost over $100,000, yet it performed thousands of tasks for client computers (also known as dumb terminals). Today, UNIX still functions in that environment; many large companies employ different flavors of UNIX for their heavy-duty, mission-critical tasks, such as e-commerce and database hosting. Common flavors of UNIX today include FreeBSD, OpenBSD, NetBSD, HP-UX, Solaris, AIX, macOS, and iOS.

The Hacker Culture

The term **hacker** refers to a person who attempts to expand their knowledge of computing through experimentation. It should not be confused with the term **cracker**, which refers to someone who illegally uses computers for personal benefit or to cause damage.

In the early days of UNIX, hackers came primarily from engineering or scientific backgrounds, because those were the fields in which most UNIX development occurred. Fundamental to hacking was the idea of sharing knowledge. A famous hacker, Richard Stallman, promoted the free sharing of ideas while he worked at the Artificial Intelligence Laboratory at MIT. He believed that free sharing of all knowledge in the computing industry would promote development. In the mid-1980s, Stallman formed the Free Software Foundation (FSF) to encourage free software development. This movement was quickly accepted by the academic community in universities around the world, and many university students and other hackers participated in making free software, most of which ran on UNIX. As a result, the hacker culture was commonly associated with the UNIX operating system.

Unfortunately, UNIX was not free software, and by the mid-1980s some of the collaboration seen earlier by UNIX vendors diminished and UNIX development fragmented into different streams. As a result, UNIX did not represent the ideals of the FSF, and so Stallman founded the **GNU Project** in 1984 to promote free development for a free operating system that was not UNIX.

Note 15

For a description of the FSF and GNU, visit gnu.org.

This development eventually led to the publication of the GNU Public License (GPL), which legalized free distribution of source code and encouraged collaborative development. Any software published under this license must be freely available with its source code; any modifications made to the source code must then be redistributed free as well, keeping the software development free forever.

As more and more hackers worked together developing software, a hacker culture developed with its own implied rules and conventions. Most developers worked together without ever meeting each other; they communicated primarily via online forums and email. *The Hacker's Dictionary*, published by MIT in 1983, detailed the terminology regarding computing and computing culture that had appeared since the mid-1970s. Along with the FSF, GNU, and GPL, it served to codify the goals and ideals of the hacker culture. But it wasn't until the publication of Eric S. Raymond's *The Cathedral and the Bazaar*, in 1999, that the larger world was introduced to this thriving culture. Raymond, a hacker himself, described several aspects of the hacker culture:

- Software users are treated as codevelopers.
- Software is developed primarily for peer recognition and not for money.
- The original author of a piece of software is regarded as the owner of that software and coordinates the cooperative software development.
- The use of a particular piece of software determines its value, not its cost.
- Attacking the author of source code is never done. Instead, bug fixes are either made or recommended.
- Developers must understand the implied rules of the hacker culture before being accepted into it.

This hacker culture proved to be very productive, with several thousand free tools and applications created in the 1980s, including the famous Emacs editor, which is a common tool used in Linux today. During this time, many programming function libraries and UNIX commands also appeared because of the work on the GNU Project. Hackers became accustomed to working together via online forum and email correspondence. In short, the UNIX hacker culture, which supported free sharing of source code and collaborative development, set the stage for Linux.

Linux

Although Richard Stallman started the GNU Project to make a free operating system, the GNU operating system never took off. Much of the experience gained by hackers developing the GNU Project was later pooled into Linux. A Finnish student named **Linus Torvalds** first developed Linux in 1991 when he was experimenting with improving MINIX (Mini-UNIX, a small educational version of UNIX developed by Andrew Tannenbaum) for his Intel x86-based computer. The Intel x86 platform was fast becoming standard in homes and businesses around the world and was a good choice for any free development at the time. The key feature of the Linux operating system that attracted the development efforts of the hacker culture was that Torvalds had published Linux under the GNU Public License.

Since 1991, when the source code for Linux was released, the number of software developers dedicated to improving Linux has increased each year. The Linux kernel was developed collaboratively and was centrally managed; however, many Linux add-on packages were developed freely worldwide by those members of the hacker culture who were interested in their release. Linux was a convenient focal point for free software developers. During the early- to mid-1990s, Linux development proceeded at full speed, with hackers contributing their time to what turned into a large-scale development project. All of this effort resulted in several distributions of Linux. A **distribution (distro)** of Linux is a collection or bundle of software containing the commonly developed Linux operating system kernel and libraries, combined with add-on software specific to a certain use. Well-known distributions of Linux include Red Hat, openSUSE, Debian, Ubuntu, Gentoo, Linux Mint, Kali, and Arch.

This branding of Linux did not imply the fragmentation that UNIX experienced in the late-1980s. All distributions of Linux had a common kernel and utilities. Their blend of add-on packages simply made them look different on the surface. Linux still derived its usefulness from collaborative development.

Linux development continued to expand throughout the late-1990s as more developers grew familiar with the form of collaborative software development advocated by the hacker culture. By 1998, when the term "OSS" first came into use, there were already many thousands of OSS developers worldwide. Small companies formed to offer Linux solutions for business. People invested in these companies by buying stock in them. Unfortunately, this trend was short-lived. By the year 2000, most of these companies had vanished. At the same time, the OSS movement caught the attention and support of large companies (including IBM, Compaq, Dell, and Hewlett-Packard), and there was a shift in Linux development over the following decade to support the larger computing environments and mobile devices.

It is important to note that Linux is a by-product of OSS development. Recall that OSS developers are still members of the hacker culture and, as such, are intrinsically motivated to develop software that has an important use. Thus, OSS development has changed over time; in the 1980s, the hacker culture concentrated on developing Internet and programming tools, whereas in the 1990s, it focused on developing the Linux operating system. Since 2000, interest has grown in embedded Linux (Linux systems that run on custom, mobile, and IoT devices) and developing cloud- and security-focused software for use on the Linux operating system. As cloud computing and security continue to grow in importance, even more development in these areas can be expected from the OSS community over the next decade.

Note 16

For more information on the free software movement and the early development of Linux, watch the 2001 television documentary *Revolution OS* (available on YouTube). It features interviews with Linus Torvalds, Richard Stallman, and Eric S. Raymond.

Linux Distributions

It is time-consuming and inefficient to obtain Linux by first downloading and installing the Linux kernel and then adding desired OSS packages. Instead, it's more common to download a distribution of Linux containing the Linux kernel, common function libraries, and a series of OSS packages.

Note 17

Remember that although Linux distributions appear different on the surface, they run the same kernel and contain many of the same packages.

Despite the fact that varied distributions of Linux are essentially the same under the surface, they do have important differences. Different distributions might support different hardware platforms. Also, Linux distributions include predefined sets of software; some Linux distributions include many server-related software applications, such as web servers and database servers, whereas others include numerous workstation and development software applications. Others might include a set of open source tools that you can use to analyze and test the security of other systems on a network.

While Linux distributions use the same Linux kernel versions that are community developed, they can modify those kernels to provide fixes and optimizations that are specific to the distribution and used for long-term support. These are called **distribution kernels**, and list a patch version and distribution identifier following the major, minor, and revision number. For example, the 5.14.0-70.5.1.el9_0.aarch64 kernel is distribution release 70.5.1 of the Linux 5.14.0 production kernel used on a 64-bit ARM (aarch64) version of the Red Hat Enterprise Linux 9.0 distribution (el9_0).

Linux distributions that include many specialized tools might not contain a GUI; an example of this is a Linux distribution that fits within a single small flash memory chip and can be used as a home Internet router. Similarly, Linux distributions used by servers usually do not include a GUI. Workstation-focused distributions, however, do include a GUI that can be further customized to suit the needs of the user.

The core component of the GUI in Linux is referred to as **X Windows**. The original implementation of X Windows on Linux was called XFree86 but has since been replaced by X.org and Wayland. X.org is the latest implementation of X Windows based on the original MIT X Windows project that was released as OSS in 2004, and Wayland is an alternative to X.org that was designed to be easier to develop and maintain. In addition to X Windows, several Linux **desktop environments** are available, which provide for the look and feel of the Linux GUI. The two main competing desktop environments available in Linux are the **GNU Network Object Model Environment (GNOME)** and the **K Desktop Environment (KDE)**. These two desktop environments are comparable in functionality, although users might have a personal preference for one over the other. GNOME closely resembles the macOS desktop, while KDE closely resembles the Windows desktop. Most Linux distributions will ship with either GNOME or KDE but allow you to download and install other desktop environments. Figures 1-5 and 1-6 compare the GNOME and KDE desktop environments.

Note 18

In addition to GNOME and KDE, several other desktop environments are available to Linux systems. One example is Cinnamon, which is a desktop derived from GNOME that is particularly easy to use. Another example is XFCE, which is a lightweight desktop environment designed for Linux systems with few CPU and RAM resources.

Although the differences between Linux distributions can help narrow the choice of Linux distributions to install, one of the most profound reasons companies choose one distribution over another is support for package managers. A **package manager** is a software system that installs and maintains

Figure 1-5 The GNOME desktop environment

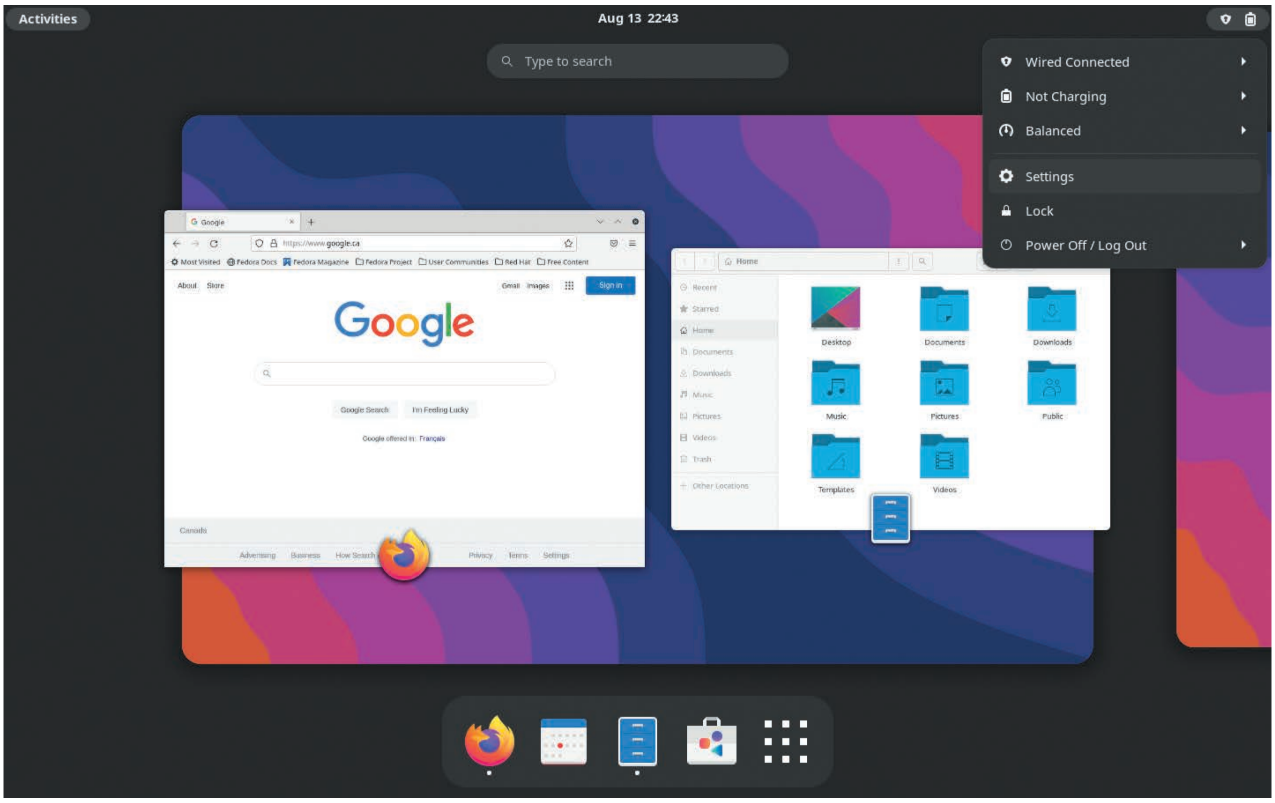
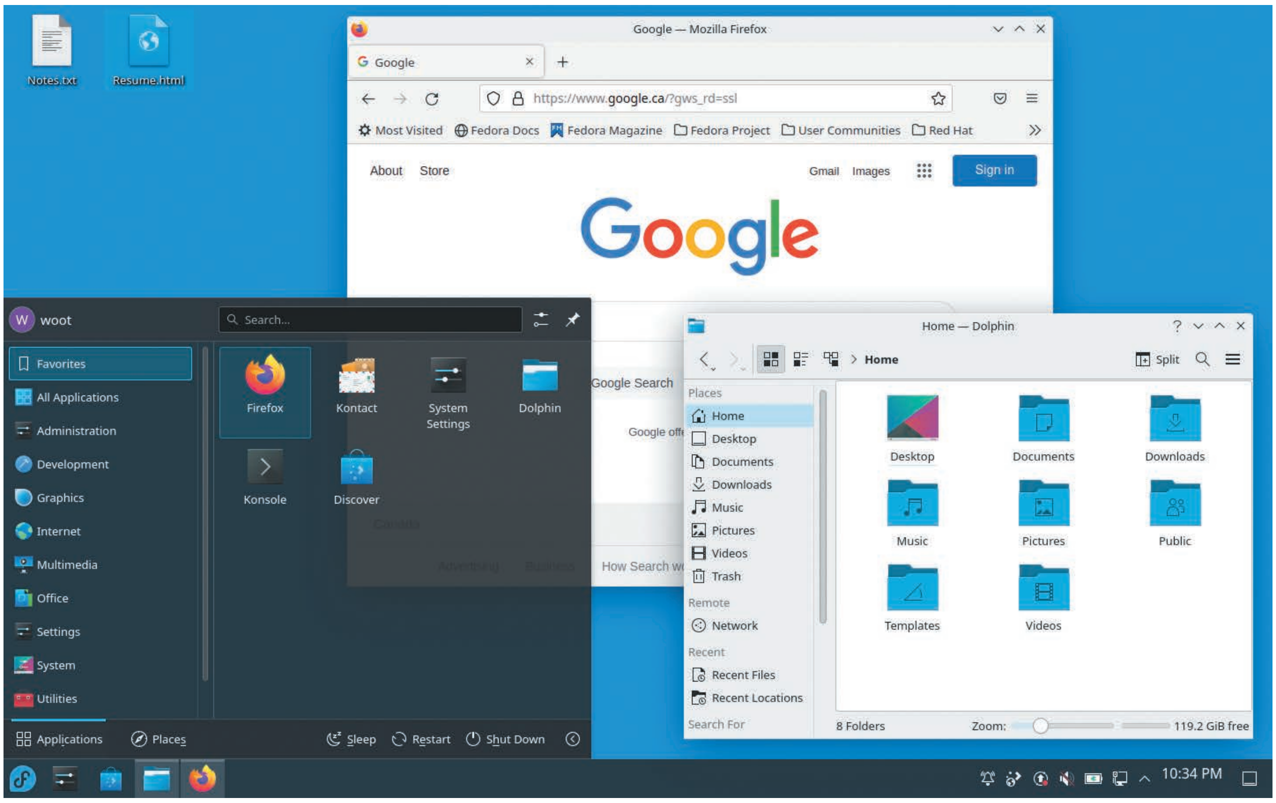


Figure 1-6 The KDE desktop environment



software. It keeps track of installed software, requires a standard format and documentation, and can manage and remove software from a system by recording all relevant software information in a central software database on your computer.

Note 19

A package manager in Linux is similar to the Apps and Features section within the Windows Settings app on a Windows system.

One of the most widely supported package managers is the Red Hat Package Manager (RPM). Most Linux software is available in RPM format, and the RPM is standard on many Linux distributions that were originally derived from the Red Hat Linux distribution. The Debian Package Manager (DPM) is also very common today; it offers the same advantages as the RPM but for systems that were originally derived from the Debian Linux distribution. Other, less common package managers available for Linux include Pacman (Arch Linux), Zypper (openSUSE Linux), and Portage (Gentoo Linux). There are even some package managers that install all related software components as a single unit (called a **sandbox**) for security and reliability. Snap, Flatpak, and AppImage are examples of sandbox-based packaged managers.

In addition to obtaining software in package manager format, you can download software in tarball format. A **tarball** is a compressed archive of files, like WinZip or RAR files, usually containing scripts that install the software contents to the correct location on the system, or source code that can be compiled into a working program and copied to the system. Unfortunately, tarballs do not update a central software database and, as a result, are very difficult to manage, upgrade, or remove from the system. Traditionally, most Linux software was available in tarball format, but package managers have since become the standard method for installing software.

Note 20

For a comprehensive list of Linux distributions, visit distrowatch.com.

Anyone can create a Linux distribution by packaging OSS with the Linux kernel. As a result, over 600 Linux distributions are publicly registered. Many are small, specialized distributions designed to fulfill certain functions, but some are mainstream Linux distributions used widely throughout the computing world. Typically, a distribution is associated with a website from which the distribution can be downloaded for free.

Table 1-4 describes some common mainstream Linux distributions, their features, and where to find them on the Internet.

Common Uses of Linux

As discussed earlier, an important feature of Linux is its versatility. Linux can provide apps and services to meet the needs of differing companies in a variety of situations. Some Linux configurations commonly used today include:

- Workstations
- Servers
- Supercomputers
- Network Devices
- Mobile and IoT Devices

Table 1-4 Common Linux distributions

Distribution	Features	Platforms	Location
Red Hat	One of the most used distributions within organizations today. Two distributions of Red Hat are available: the Red Hat Enterprise Linux (RHEL) distribution geared for enterprise environments and the Fedora distribution geared for all environments (servers, desktops, laptops, etc.).	Intel x86_64 ARM/ARM64 MIPS/MIPS64 RISC-V PPC Mainframe	redhat.com getfedora.org
openSUSE	Originally developed primarily in Europe, openSUSE is the oldest business-focused distribution. Like Red Hat, it is commonly used within many organizations today.	Intel x86/x86_64 ARM/ARM64 RISC-V PPC Mainframe	opensuse.org
Debian	One of the earliest distributions, Debian focuses on stability in its structure, release cycle, and available OSS packages. For these reasons, there are many other mainstream distributions based on Debian.	Intel x86/x86_64 ARM/ARM64 MIPS/MIPS64 RISC-V PPC Mainframe	debian.org
Ubuntu	A Debian-based distribution that is widely used in many cloud computing environments, Ubuntu is also commonly used on desktop and mobile devices.	Intel x86_64 ARM/ARM64 RISC-V PPC Mainframe	ubuntu.com
Kali	A Debian-based distribution that is designed for security monitoring, digital forensics, and penetration testing.	Intel x86/x86_64 ARM	kali.org
Gentoo	A distribution that focuses on hardware and software optimization. Each Gentoo component is compiled and optimized specifically for the hardware on the system.	Intel x86/x86_64 PPC/POWER ARM/ARM64	gentoo.org
Linux Mint	A distribution based on Ubuntu that is focused on providing a simple and intuitive desktop experience for novice Linux users.	Intel x86/x86_64	linuxmint.com
Arch	A very lightweight and customizable distribution designed for advanced Linux users that prefer to shape all aspects of their operating system.	Intel x86_64 ARM64 RISC-V PPC	archlinux.org

Workstations

While Windows and macOS remain the most common workstation operating systems today, cybersecurity professionals and software developers often choose to install Linux on their desktop or laptop workstation instead.

Cybersecurity professionals must use specialized tools to scan computers and networks for security vulnerabilities as well as attempt to break into systems to test the strength of their security measures (often called a penetration test). Additionally, cybersecurity professionals must continually monitor the security of systems and networks to determine when a system has been breached, as well as perform forensic analysis to investigate the nature of the breach and the damage incurred. Most of the tools used to perform these activities are available exclusively for Linux systems and preinstalled on cybersecurity-focused Linux distributions, such as Kali Linux.

The rich set of OSS development tools and frameworks available for Linux also makes Linux the preferred choice for software development workstations. For this reason, most major computer vendors provide a developer-focused laptop model that comes preinstalled with a Linux distribution instead of Windows. Some common examples include the HP Dev One, Dell XPS Developer Edition, and Lenovo ThinkPad (Linux Edition).

Modern Linux distributions contain drivers for nearly all desktop and laptop hardware, and often run faster than Windows and macOS on the same system. Consequently, organizations that have a limited budget for computer upgrades (e.g., elementary and secondary schools) often install Linux on legacy systems to extend their useful life.

Note 21

Many open source alternatives to common closed source software can also be installed on Linux workstations. For example, LibreOffice is an open source equivalent to Microsoft Office, and Krita is an open source equivalent to Adobe Photoshop.

Servers

Linux hosts a wide range of services that can be made available to other systems across a network such as the Internet. A computer that hosts one or more services that can be accessed by other systems across a network is commonly referred to as a **server**.

Note 22

On Linux systems, services are often called **daemons**.

Servers typically use more powerful hardware compared to workstations and may be located within your organization or in the cloud. While you can install Linux directly onto server hardware (called a **bare metal** installation), most organizations today run multiple server operating systems concurrently on the same server hardware for cost and power efficiency using **virtualization software** (also called a **hypervisor**). There are many different hypervisors available for servers, including the following:

- Hyper-V, a closed source hypervisor developed by Microsoft
- Elastic Sky X Integrated (ESXi), a closed source hypervisor developed by VMWare
- Kernel-based Virtual Machine (KVM), a hypervisor built into the Linux kernel
- Quick Emulator (QEMU), an open source hypervisor that is often used alongside KVM to provide fast access to hardware
- Xen, an open source hypervisor developed by the Linux Foundation

Each operating system that is run using a hypervisor is called a **virtual machine**. Modern server hardware can run dozens or hundreds of virtual machines, with each virtual machine hosting a separate Linux server.

Linux servers may also be hosted within a **container**. Containers are like virtual machines but lack an operating system kernel. Consequently, containers must use the kernel on an underlying operating system that has **container runtime** software installed. Docker, Podman, and LXC are common examples of container runtime software. Additionally, containers are usually smaller than virtual machines because they are configured with just enough filesystem, software libraries and supporting programs to run a particular service or app. This makes containers ideal for running in the cloud; a typical cloud server today can run thousands of small Linux containers.

Regardless of whether they are installed bare metal or hosted within a virtual machine or container, Linux servers can be configured to provide many different services to other systems on a network. The most common services found on Linux servers include the following:

- Web services
- DNS services
- DHCP services
- Time services
- Mail services
- File and print services
- Database services
- Authentication services
- Certificate services

Many of these services are discussed in more detail later in this course.

Web Services

The most popular tool for accessing resources on the Internet is the web browser, which can connect computers to web servers worldwide hosting information of many types: text, pictures, music, binary data, video, and much more. On a basic level, a web server is just a server using **Hypertext Transfer Protocol (HTTP)** to provide information to requesting web browsers running on other computers. However, web servers can also run **Common Gateway Interface (CGI)** programs to perform processing and access other resources, as well as encrypt communication using **Secure Sockets Layer (SSL)** or **Transport Layer Security (TLS)** to protect the information entered and shown within the web browser. You can tell SSL/TLS is in use when the *http://* in the browser's address bar changes to *https://*. While there are many open source web server software packages available for Linux, the two most common are Apache and Nginx.

DNS Services

Each computer on the Internet needs a unique way to identify itself and to refer to other computers. This is accomplished by assigning each computer a number called an **Internet Protocol (IP) address**. An IP address is a string of numbers that would be difficult for the typical person to remember. Thus, IP addresses are often associated with more user-friendly names. In particular, servers are identified by names like www.linux.org, which are known as **fully qualified domain names (FQDNs)**. When you use a web browser such as Google Chrome or Mozilla Firefox to request information from a web server, you typically type the web server's FQDN (e.g., www.linux.org) into your browser's address bar. However, FQDNs exist only for the convenience of the human beings who use computer networks. The computers themselves rely on IP addresses. Thus, before your browser can retrieve the requested information from the web server, it needs to know the IP address associated with the FQDN you typed into the address bar. Your web browser gets this information by contacting a server hosting a **Domain Name Space (DNS)** service. The server running the DNS service maintains a list of the proper FQDN to IP mappings, and quickly returns the requested IP address to your browser. Your browser can then use this IP address to connect to the target website.

For companies wanting to create a DNS server, Linux is an inexpensive solution, as many distributions of Linux ship with a DNS service known as BIND (Berkeley Internet Name Daemon).

DHCP Services

Recall that each computer on the Internet must have an IP address. While an administrator typically configures this address on servers, it is impractical to do the same to the plethora of workstations and other systems on the Internet. As a result, most computers on the Internet automatically receive an IP address and related configuration from a **Dynamic Host Configuration Protocol (DHCP)** server by broadcasting an IP address request on the network. This is normally performed at system startup and periodically afterwards. Any Linux computer can function as a DHCP server by adding and configuring the DHCP daemon, dhcpd.

Time Services

Most system components and network services require the correct date and time in order to function properly. The BIOS on each computer contains a system clock that stores the current date and time. Operating systems can choose to use the time from this system clock or obtain time information from other servers on the Internet or local network using the **Network Time Protocol (NTP)**. A Linux system can obtain time information from an NTP server or provide time information to other systems using NTP via the NTP daemon (ntpd) or Chrony NTP daemon (chronyd).

Mail Services

In the 1980s and early 1990s, email was a service that was found primarily in universities. Today, almost every Internet user has an email account and uses email on a regular basis. Email addresses are easy to acquire and can be obtained free of charge. Email is distributed via a network of email server services, also known as **Mail Transfer Agents (MTAs)**. Many MTAs are freely available for Linux, including sendmail, postfix, and exim.

File and Print Services

The most common and efficient method for transferring files over the Internet is by using the **File Transfer Protocol (FTP)**. Most FTP servers available on the Internet allow any user to connect and are, hence, called anonymous FTP servers. Furthermore, web browsers and FTP client programs that allow users to connect to these FTP servers are freely available for Linux, UNIX, Windows, and macOS systems. Although several FTP services are available for Linux, the most used is the Very Secure FTP Server daemon, vsftpd.

Most organizations use private networks to share resources, primarily printers and files. In business, it is not cost effective to purchase and install a printer next to the computer of every user who needs to print. It is far easier and cheaper to install one central printer on a server and let multiple users print to it across the computer network. Additionally, files must also be available to multiple users on the network to allow them to collaborate on projects or perform their daily jobs. By centrally storing these files on a server, users can access data regardless of the computer that they log into. Central storage also allows a company to safeguard its information by using devices to back up or make copies of stored data on a regular basis in case of computer failure. Most companies perform backups of data at least every week to ensure that if data is lost on the central server, it can be restored from a backup copy quickly.

Linux is well suited to the task of centrally sharing file and print resources on a corporate network to other Linux, UNIX, Windows, and macOS machines using services such as Network File System (NFS), Common UNIX Printing System (CUPS), and Samba.

Database Services

Most data used by applications is organized into tables of related information and stored in a **database**, where it is accessible via **database management system (DBMS)** software. Applications on workstations and other systems interact with a DBMS on a database server to create, manage, and retrieve the data stored within the database. Several free open and closed source DBMS software packages are available for Linux. The most popular open source DBMS packages include PostgreSQL, MySQL (My Structured Query Language), and MariaDB (based on MySQL), while the most popular closed source DBMS packages include Microsoft SQL Server and Oracle Database.

Authentication Services

To access any computer securely, you must log in using a valid username and password before gaining access to the user interface. This process is called **authentication**. However, users today often need to securely access resources on several servers within their organization, and each of these computers requires that you authenticate before access is granted. To ensure that you do not need to enter your username and password on every network computer that you access within your organization, you can configure your computer to log into an authentication service on a network server using a protocol, such as **Kerberos**. Once you log into an authentication service using Kerberos successfully, you receive a Kerberos ticket that your computer presents to all other computers within your organization to prove your identity. Microsoft Active Directory is one of the most common Kerberos-based authentication services, and every Linux computer can be easily configured to log into it. However, organizations can install alternate authentication services on Linux servers, including the Kerberos-based Apache Directory.

Note 23

You can learn more about Apache Directory at directory.apache.org.

Certificate Services

Since data is frequently passed across networks to other computers, the data within them could easily be intercepted and read by crackers. To prevent this, many technologies use an encryption algorithm to protect the data before it is transmitted on the network. An encryption algorithm uses a series of mathematical steps in sequence to scramble data. Because the steps within encryption algorithms are widely known, nearly all encryption algorithms use a random component called a **key** to modify the steps within the algorithm. **Symmetric encryption** algorithms are reversible; data can be decrypted by reversing the algorithm using the same key that was used to encrypt it. Unfortunately, it is difficult for two computers on a network to communicate this key. As a result, network technologies typically use **asymmetric encryption** to protect the data that travels across the network. Asymmetric encryption uses a pair of keys that are uniquely generated on each system: a **public key** and a **private key**. You can think of a public key as the opposite of a private key. If you encrypt data using a public key, that data can only be decrypted using the matching private key. Alternatively, if you encrypt data using a private key, that data can only be decrypted using the matching public key. Each system must contain at least one public/private key pair. The public key is freely distributed to any other host on the network, whereas the private key is used only by the system and never distributed.

Say, for example, that you want to send an encrypted message from your computer (host A) to another computer (host B). Your computer would first obtain the public key from host B and use it to encrypt the message. Next, your computer will send the encrypted message across the network to host B, at which point host B uses its private key to decrypt the message. Because host B is the only computer on the network with the private key that matches the public key you used to encrypt the message, host B is the only computer on the network that can decrypt the message.

You can also use private keys to authenticate a message. If host A encrypts a message using its private key and sends that message to host B, host B (and any other host on the network) can easily obtain the matching public key from host A to decrypt the message. By successfully decrypting the message, host B has proved that it must have been encrypted using host A's private key. Because host A is the only computer on the network that possesses this private key, host B has proven that the message was sent by host A and not another computer that has impersonated the sender.

Note 24

A message that has been encrypted using a private key is called a **digital signature**.

Note 25

Most network technologies use symmetric encryption to encrypt data that is sent on a network, and asymmetric encryption to securely communicate the symmetric encryption key between computers on the network.

Note 26

Common symmetric encryption algorithms include AES, 3DES, and RC4. Common asymmetric encryption algorithms include DH, RSA, and ECC.

Because public keys are transmitted across a network, a cracker could substitute their own public key in place of another public key to hijack encrypted communications. To prevent this, nearly all public keys are digitally signed by a trusted third-party computer called a **Certification Authority (CA)**. While some commercial CAs digitally sign certificates for a fee (e.g., DigiCert), others do it free of charge (e.g., Let's Encrypt). Alternatively, an organization can install their own CA to digitally sign public keys for use by their own computers when communicating across networks and the Internet. Common CA software packages for Linux include OpenSSL and OpenXPKI.

Note 27

A public key that has been digitally signed by a CA is called a **certificate**.

Note 28

An organization that installs one or more CAs is said to have a **Public Key Infrastructure (PKI)**.

Note 29

You can learn more about OpenSSL at openssl.org. To learn more about OpenXPKI, visit openxpki.org.

Supercomputers

Many companies and institutions use computers to perform extraordinarily large calculations for which most servers would be unsuitable. To accomplish these tasks, companies use specialized services to combine several servers in a way that allows them to function as one large supercomputer. Combining multiple computers to function as a single unit is called **clustering**.

Although it might seem logical to instead purchase computers with many processors, the performance of a computer relative to the number of processors decreases as you add processors to a computer. In other words, a computer with 64 processors does not handle 64 times as much work as one processor because of physical limitations within the computer hardware itself; a computer with 64 processors might only perform 50 times as much work as a single processor. The ability for a computer to increase workload as the number of processors increases is known as

scalability, and most computers, regardless of the operating system used, do not scale well with more than 32 processors. Clustering, however, results in much better scalability; 64 computers with one processor each working toward a common goal can handle close to 64 times as much as a single processor.

The supercomputing community has focused on Linux when developing clustering technology, and the most common method of Linux clustering is known as **Beowulf clustering**. One way to implement a Beowulf cluster is to have one main computer send instructions to several other computers, which compute parts of the calculation concurrently and send their results back to the main computer using a **Message Passing Interface (MPI)** software framework such as OpenMPI. This type of supercomputing breaks tasks into smaller units and executes them in parallel on many machines at once; thus, it is commonly referred to as parallel supercomputing. Beowulf parallel supercomputer technology has been aggressively developed since the mid-1990s and has been tested in various environments; currently, institutions, companies, and universities host thousands of Beowulf clusters worldwide.

Note 30

You can find more information about OpenMPI at open-mpi.org.

Network Devices

Specialized network devices are used to ensure that information can be sent between different networks and the Internet, as well as prevent malicious software from propagating to systems on the network. Most network devices are specialized Linux computers that provide key network-focused services, including the following:

- Routing services
- Firewall and proxy services
- Advanced security services

Routing Services

Routing is a core service that is necessary for the Internet to function. The Internet is merely a large network of interconnected networks; in other words, it connects company networks, home networks, and institutional networks so that they can communicate with each other. A **router** is a computer or special hardware device that provides this interconnection; it contains information regarding the structure of the Internet and sends information from one network to another. Companies can use routers to connect their internal networks to the Internet as well as to connect their networks together inside the company. Linux is a good choice for this service as it provides support for routing and is easily customizable; many small Linux distributions, each of which can fit within a few hundred megabytes of flash storage, provide routing capabilities.

Firewall and Proxy Services

The term **firewall** describes the structures that prevent a fire from spreading. In the automobile industry, a firewall protects the passengers in a car if a fire breaks out in the engine compartment. Just as an automobile firewall protects passengers, a computer firewall protects companies from outside intruders on the Internet. Most firewalls are routers that are placed between the company network and the company's connection to the Internet; all traffic must then pass through this firewall, allowing the company to control traffic at the firewall, based on a complex set of rules. Linux has firewall support built directly into the kernel. Utilities such as iptables and nftables are included with nearly all Linux distributions, and can be used to configure the rules necessary to make a router a firewall.

Because firewalls are usually located between a company's internal network and the Internet, they often provide other services that allow computers inside the company easy access to the Internet. The most common of these services are known as proxy services. A **proxy server** requests Internet resources, such as websites and FTP sites, on behalf of the computer inside the company. In other words, a workstation computer inside the company sends a request to the proxy server connected to the Internet. The proxy server then obtains and returns the requested information to the workstation computer. One proxy server can allow thousands of company workstation computers access to the Internet simultaneously without lengthy configuration of the workstation computers; proxy servers keep track of the information passed to each client by maintaining a **Network Address Translation (NAT)** table. Although routers and firewalls can be configured to perform NAT capabilities, Squid is the most common fully featured proxy server used on Linux. Squid retains (or caches) a copy of any requested Internet resources so that it can respond more quickly to future requests for the same resources.

Advanced Security Services

Today, most organizations use specialized server hardware running Linux to provide routing, firewall, and proxy services alongside additional advanced security services. These servers are called **security appliances** and are typically used to provide security between an organization's network and the Internet. Many Linux-based security appliances and security appliance software suites provide one or more of the following advanced security services:

- Malware, virus, and spam filtering
- Bot and intrusion prevention
- Advanced traffic throttling
- Virtual private network (VPN) functionality
- Centralized event logging for network devices using Simple Network Management Protocol (SNMP)
- Security Information and Event Management (SIEM) for centralized network and security monitoring

Note 31

Security appliances that provide multiple security functions are often called Next Generation Firewall (NGFW) or Unified Threat Management (UTM) appliances.

Note 32

A common Linux-based SIEM security appliance software suite is Alienvault OSSIM, which is available at cybersecurity.att.com/products/ossim.

Note 33

Some routers, firewalls, and security appliances can also separate requests for a specific resource, such as a website, across several servers that provide the same service; this feature is called **load balancing**, and can be used to provide fast, fault-tolerant access to specific services on other servers.

Mobile and IoT Devices

Over the past two decades, mobile devices such as tablets and smartphones have grown tremendously in popularity. Today, most of these devices are custom built to run either the UNIX-based Apple iOS or Linux-based Google Android operating system. However, some tablet and smartphone device models allow for the installation of other Linux-based mobile operating systems that have greater customization and privacy options. Some examples of Linux-based operating systems designed as an alternative to Android include postmarketOS, Ubuntu Touch, Mobian, and Manjaro.

Similarly, embedded technology advances in recent years have led to a wide range of IoT devices, such as personal fitness trackers and smart door locks. To avoid software licensing fees and remain cost-competitive on the market, IoT device manufacturers typically use a custom Linux operating system on these devices. This has led to a rich and well-documented IoT OSS ecosystem that anyone can take advantage of using general purpose IoT hardware devices, such as the Raspberry Pi. For example, computer hobbyists can configure Linux software on a Raspberry Pi to drive robotics, monitor agriculture, as well as perform a wide range of home and industrial automation tasks.

Linux in the Cloud

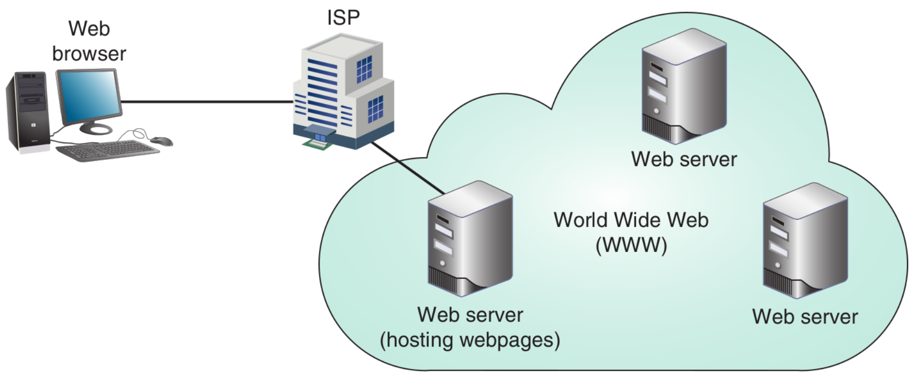
Many different types of cloud configurations and technologies are available today. To understand Linux's role in the cloud, you must first have a solid understanding of cloud concepts. More specifically, you should be able to define the cloud, as well as various cloud types and delivery models. Additionally, you should understand the process used to deploy new versions of web apps on cloud servers.

Defining the Cloud

In the early 1990s, the US government-funded Advanced Research Projects Agency Network (ARPANET) and National Science Foundation Network (NSFNET) infrastructures were sold to different companies to create the commercial Internet. These companies were called **Internet Service Providers (ISPs)** and provided Internet access to individuals and organizations for a fee.

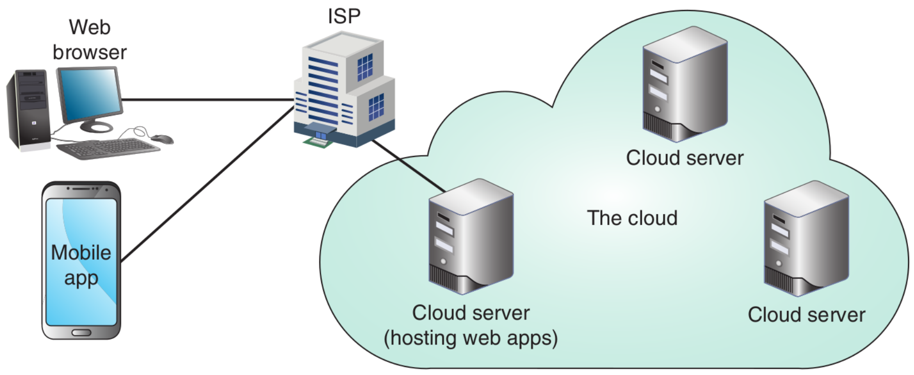
However, aside from providing physical network access to other computers, ISPs provided no resources for users to consume. Instead, most resources available on the Internet in the 1990s consisted of websites that contained webpages with information and media (e.g., pictures, music, video) for different topics and organizations. These websites were hosted on web servers and accessed using the HTTP or HTTPS protocols from a web browser on a user's computer. As shown in Figure 1-7, the worldwide collection of web servers on the Internet was called **World Wide Web (WWW)**.

Figure 1-7 The World Wide Web of the 1990s



Today, most web servers contain one or more **web apps** that process and communicate both data and media to users in complex ways. Search engines (e.g., Google), Office 365, video streaming services (e.g., Netflix), personal banking, social media websites, and the software on the Internet that smartphones and smart assistants (e.g., Amazon Alexa) communicate with are all examples of web apps. Web apps are software programs on web servers that can be accessed by other computers and IoT devices across the Internet via a program, web browser, or mobile app. Each web server that runs a web app is called a cloud server, and the worldwide collection of cloud servers is called the cloud, as shown in Figure 1-8.

Figure 1-8 The cloud



Note 34

Most web apps are accessed from a website on a web server. In this case, the website is said to provide the "front end" for the web app. Computer programs and mobile apps can also be used as the front end for web apps, either by connecting to a website using the HTTP or HTTPS protocol, or by communicating directly to a web app on a cloud server using a custom protocol.

The rapid growth of the WWW in the 1990s was largely due to the open source Apache web server software that ran on the open source Linux operating system. Apache and Linux allowed anyone with a computer and Internet access to host a web server to serve webpages without any software costs. Consequently, web technologies have evolved using the open source ecosystem since the 1990s, and most web apps in the cloud today run on Linux systems using open source **web app frameworks**. A web app framework is a collection of software packages or modules that allow software developers to write web apps using a programming language without having to implement many underlying system functionalities, such as network protocol usage and process management. For example, the Django web app framework allows programmers to easily write web apps using the Python programming language, whereas the React framework allows programmers to easily write web apps using JavaScript.

Cloud Types

Any organization that hosts cloud servers is called a **cloud provider**. A **public cloud** consists of cloud servers on the Internet that can be rented by others, whereas a **private cloud** consists of cloud servers that are used exclusively by the organization that owns them. **Hybrid cloud** refers to using both a public and private cloud together for a specific purpose. For example, you could provide access to a web app in a private cloud but redirect users to the same web app in a public cloud when your web app receives a large number of requests that consume too many hardware resources.

Note 35

There are many public clouds available on the Internet, including Amazon Web Services, Google Cloud Platform, Salesforce Cloud, Digital Ocean, and Microsoft Azure.

Note 36

Private clouds vary in size from one organization to another; they may consist of many cloud servers in a data center or a single cloud server within an organization.

Cloud Delivery Models

Regardless of whether you are using a public or private cloud, you can host web apps on a cloud server using different methods. These methods are called **cloud delivery models** and include **Infrastructure as a Service (IaaS)**, **Platform as a Service (PaaS)**, and **Software as a Service (SaaS)**.

With IaaS, the cloud provider offers a cloud platform that provides Internet access, IP addressing, and FQDN name resolution to virtual machines that you create on their hypervisor. You must manage each virtual machine as you would any other operating system, including installing and managing the web apps that users will access from across the Internet. This structure is shown in Figure 1-9.

PaaS allows you to run web app containers in the cloud. Recall that containers do not contain a full operating system. As a result, they must be run using container runtime software that allows access to a kernel on an underlying operating system (or "platform"), as shown in Figure 1-10.

Note 37

IaaS and PaaS can be used together. In this case, containers are run on the operating system in a virtual machine hosted on a cloud provider.

Unlike IaaS and PaaS, SaaS is not used to configure virtual machines or containers. Instead, the SaaS cloud provider maintains all aspects of the network, hardware, and operating system; it merely executes the web app that you provide. This structure is shown in Figure 1-11.

Because Linux is OSS and doesn't require expensive licensing, it is often used as the underlying operating system for SaaS, as well as the operating system used for containers (PaaS) and virtual machines (IaaS). In fact, Linux comprised over 90 percent of the public cloud in 2020, which is often reflected by posts on websites and social media sites as shown in Figure 1-12.

Figure 1-9 A sample IaaS structure

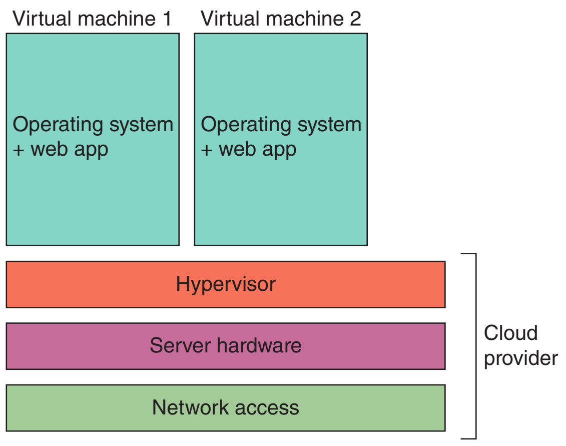


Figure 1-10 A sample PaaS structure

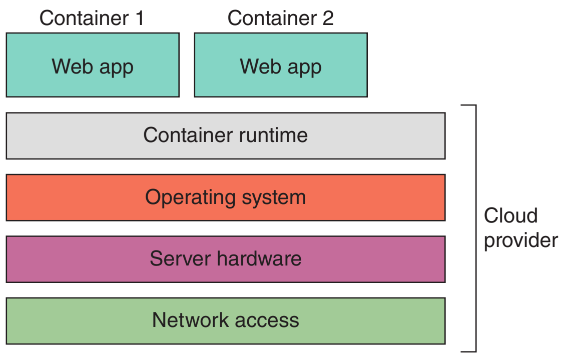
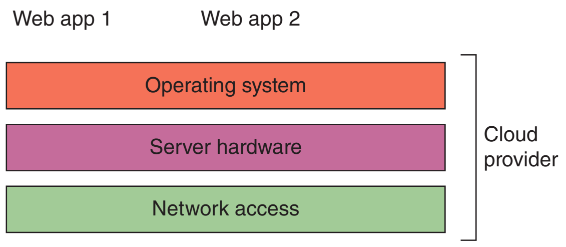


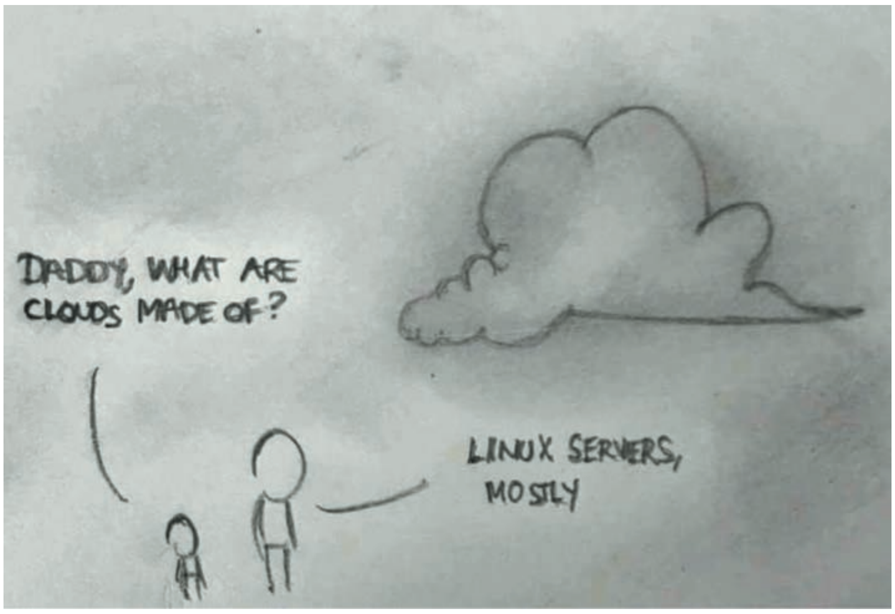
Figure 1-11 A sample SaaS structure



Note 38

Most public cloud providers allow you to choose between SaaS, PaaS, and IaaS.

Figure 1-12 A Reddit post regarding Linux cloud market share



Note 39

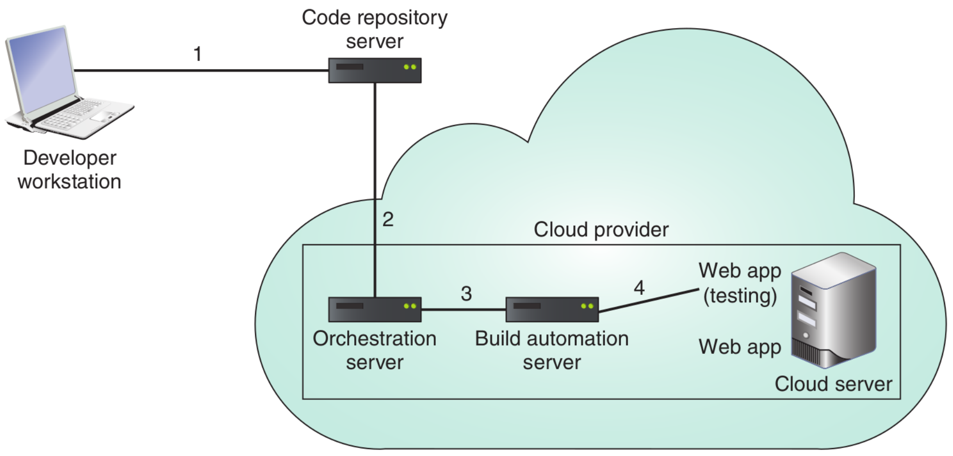
The words "as a service" are commonly used for marketing purposes. For example, Mobile as a Service (MaaS) can be used to describe web apps that manage smartphone devices, whereas Database as a Service (DBaaS) can be used to describe web apps that provide access to a database. These terms are collectively referred to as Anything as a Service (XaaS), and represent specific uses of either SaaS, PaaS, or IaaS.

Understanding Cloud Workflows

The prevalence of Linux in the cloud means that most web apps that software developers create are implemented using open source frameworks and developed on Linux workstations, or on Windows or macOS workstations that run a Linux virtual machine. These web apps must be revised on a continual basis to fix bugs and incorporate new features. While developers can create new versions of the web app on their workstation, they need to test these new versions on the cloud provider to ensure that they work as expected before replacing the existing web app with the new version. As part of the development process, new versions of the web app may need to be tested on a cloud provider several times per day. Each new version will require that a new container or virtual machine be created with the correct settings for the web app using **build automation** software. Additional software is required to move the web app from the developer workstation to the new container or virtual machine on the cloud provider, as well as ensure that the web app is ready for execution. This entire process is called a **continuous deployment (CD)** workflow, and the people that manage this workflow are called **DevOps** because they are system operators (ops) that support web app development (dev). Figure 1-13 illustrates a sample CD workflow that allows developers to push new versions of a web app to virtual machines in the cloud.

The first step shown in Figure 1-13 involves developers pushing a new version of their web app code to a **code repository** server, where it is peer-reviewed and eventually approved for use; a process called **continuous integration (CI)**. Next, **orchestration** software running on a server at the cloud provider obtains the new version of the web app from the code repository server (step 2) and compiles it to executable form. The web app is then sent to a build automation server (step 3),

Figure 1-13 A sample CD workflow



which creates a new container or virtual machine on the cloud server and adds the web app to it for testing (step 4). At this point, the new version of the web app can be tested by the web app developers or automated testing software. If this new version doesn't work as expected, the container or virtual machine is removed and the whole process is repeated for another new version of the web app. However, if this new version of the web app works as expected, the container or virtual machine used to test the web app will replace the publicly accessible container or virtual machine running the old version of the web app, and Internet users will immediately have access to the new version.

Note 40

As with developer workstations and cloud servers, the code repository, orchestration, and build automation servers shown in Figure 1-13 almost always run the Linux operating system.

Note 41

Hundreds of different software packages are available for implementing CD workflows. Today, Git is the most common code repository software used for CI, while Kubernetes is most used for orchestration. Ansible, Puppet, Chef, Terraform, and SaltStack are commonly used for build automation.

Summary

- Linux is an operating system whose kernel and associated software packages are freely developed and improved upon by a large community of software developers in collaboration. It is based on the UNIX operating system and has roots in the hacker culture perpetuated by the Free Software Foundation.
- Because Linux is published under the GNU Public License, it is referred to as open source software (OSS). Most additional software that is run on Linux is also OSS.
- Companies find Linux a stable, low-risk, and flexible alternative to other operating systems; it can be installed on several hardware platforms to meet business needs and results in a lower TCO.

- The Linux operating system is available in various distributions, all of which have a common kernel but are packaged with different OSS applications. Distributions, documentation, and other Linux resources are freely available online.
- Linux is an extremely versatile operating system that is used on workstations, servers, and supercomputers, as well as network, mobile, and IoT devices.
- The cloud is an evolution of the World Wide Web that includes complex web apps. Most of these web apps are hosted on Linux servers using the SaaS, PaaS, or IaaS cloud delivery models and updated frequently using a continuous deployment workflow.

Key Terms

application (app)
asymmetric encryption
authentication
bare metal
Beowulf clustering
build automation
certificate
Certification Authority (CA)
closed source software
cloud delivery model
cloud provider
clustering
code repository
Common Gateway Interface (CGI)
container
container runtime
continuous deployment (CD)
continuous integration (CI)
copyleft license
cracker
daemon
database
database management system (DBMS)
desktop environment
development kernel
device driver
DevOps
digital signature
distribution (distro)
distribution kernel
Domain Name Space (DNS)
Dynamic Host Configuration Protocol (DHCP)
File Transfer Protocol (FTP)
firewall
flavor
forum
Free Software Foundation (FSF)
freeware
fully qualified domain name (FQDN)
GNU General Public License (GPL)
GNU Network Object Model Environment (GNOME)
GNU Project
graphical user interface (GUI)
hacker
hardware
hybrid cloud
Hypertext Transfer Protocol (HTTP)
Infrastructure as a Service (IaaS)
Internet of Things (IoT)
Internet Protocol (IP) address
Internet Service Provider (ISP)
K Desktop Environment (KDE)
Kerberos
kernel
key
Linus Torvalds
Linux
Linux User Group (LUG)
load balancing
Mail Transfer Agent (MTA)
major number
Message Passing Interface (MPI)
minor number
Multiplexed Information and Computing Service (MULTICS)
Network Address Translation (NAT)
Network Time Protocol (NTP)
open source software (OSS)
operating system
orchestration
package manager
permissive license
Platform as a Service (PaaS)
private cloud
private key
process
production kernel
program
programming language
proxy server
public cloud
public key
Public Key Infrastructure (PKI)
revision number
router
sandbox
scalability
Secure Sockets Layer (SSL)
security appliance
server
service
shareware
software
Software as a Service (SaaS)
source code
symmetric encryption
tarball
total cost of ownership (TCO)
Transport Layer Security (TLS)
UNIX
user
user interface
virtual machine
virtualization software (hypervisor)
web app
web app framework
World Wide Web (WWW)
X Windows