O'REILLY®
Technical Guide

Linux on Azure

Deploying, Securing, and Monitoring Linux Workloads

Ned Bellavance
& Chris Hayner

Microsoft Azure

Securely Migrate and Modernize to Accelerate AI Innovation

Migrate and modernize on a proven platform for Linux and open-source

Maximize business agility

Modernize apps intelligently with AI, for AI

Unlock AI with unified data

Accelerate AI innovation by unifying data in the cloud

Secure from code to cloud

From foundational security to cloud-native application protection

Best-in-class cost and performance

Purpose-built for all Linux and open source workloads

Learn more

Linux on Azure

Deploying, Securing, and Monitoring Linux Workloads

Ned Bellavance and Chris Hayner

O'REILLY®

Linux on Azure
by Ned Bellavance and Chris Hayner

Published by O'Reilly Media, Inc., 141 Stony Circle, Suite 195, Santa Rosa, CA 95401.

O'Reilly books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (*https://oreilly.com*). For more information, contact our corporate/institutional sales department: 800-998-9938 or *corporate@oreilly.com*.

Acquisitions Editor: Megan Laddusaw
Development Editor: Gary O'Brien
Production Editor: Jonathon Owen
Copyeditor: Adam Lawrence
Proofreader: Shannon Turlington

Cover Designer: Karen Montgomery
Cover Illustrator: Ellie Volckhausen
Interior Designer: David Futato
Interior Illustrator: Kate Dullea

November 2025: First Edition

Revision History for the First Edition
2025-11-13: First Release

See *https://oreilly.com/catalog/errata.csp?isbn=9798341621428* for release details.

979-8-341-62139-8

[LSI]

Table of Contents

CHAPTER 1

The Importance of Open Source Software

Generally speaking, software is developed and distributed in one of two ways: open source or closed source. In the closed source model, users will either purchase the software outright or pay for it via a subscription. In either case, users will only receive the compiled executables—the machine-readable binaries that make the software run. The source code (the machine-readable instructions that tell the computer what to do) is not made available. This approach is common in commercial software, where companies are keen to protect their intellectual property by keeping the source code private.

Open source software (OSS) takes a different approach. When a company follows an OSS development and distribution model,[1] users get both the executable binaries and access to the source code. This source code is public, meaning that anyone—even non-paying customers—can examine it, understand how it works, and make suggestions on how to improve it. As we will see, sharing source code was actually the norm at first—it was only later that commercial products started restricting access. However, over the past decade or so, a remarkable revival has been taking place, with OSS taking center stage for many commercial programs.

1 It should be noted that access to the source code is not the only thing that makes a software package "open source." The definition has evolved many times over the decades, with an authoritative one being maintained by the Open Source Initiative. It is, in our opinion, well worth the read.

The concept of OSS has its roots in the early days of computing, long before the idea of sharing source code had a formal name. At that time, software development was largely tied to academic institutions where sharing source code was standard practice. Programmers openly distributed human-readable source code, enabling others to fix bugs, tailor functionality to suit their specific needs, and even introduce new features. These improvements were then shared back with the community in the spirit of collaboration and innovation that would come to define a formalized OSS culture later on. The rise of networks like ARPANET—and later, the internet—further amplified this collaborative culture, laying the groundwork for modern open source development.

While sharing source code was once the norm, this practice diminished as software became a commercial commodity. Companies began trying to protect their profits (and intellectual property) by keeping the source code private (or proprietary) and not available to customers. Once this started happening, OSS became more of a formalized movement. Richard Stallman was one early proponent, forming the GNU Project in late 1983 with the goal of creating only free and open source software. Later, Stallman wrote the GNU Manifesto, which further formalized what he believed were the key tenets of free and open source software. According to this document, you as the end user should have the freedom to:

- Run the program for any purpose
- Modify the program to suit your needs
- Redistribute copies of the program
- Distribute modified versions of the program

OSS was now part of a formalized movement.

Now, it's important to note that this movement does not require software to have a $0 price tag. Software was, and is, big business, and even Richard Stallman made money off of the GNU Project's efforts at times. The "free" in free software as he describes it refers to the freedoms the end user has. As Stallman himself said, "When I speak of free software, I'm referring to freedom, not price. So think of free speech, not free beer."

Today, OSS has evolved beyond just simply providing access to source code. Companies like Red Hat, SUSE, and even Microsoft

(much more on this later) celebrate and encourage the broader culture of OSS. According to Red Hat, the OSS model drives innovation via open and inclusive collaboration (Figure 1-1). The company describes it as a "Do-ocracy," where anyone with the ability to be involved should be involved—both in coding and decision making. The goal here is for everyone involved in a project, no matter how large or how small, to have a voice. This cultural foundation ensures that OSS remains dynamic, inclusive, and adaptable, enabling the innovation that is essential to solving complex problems.

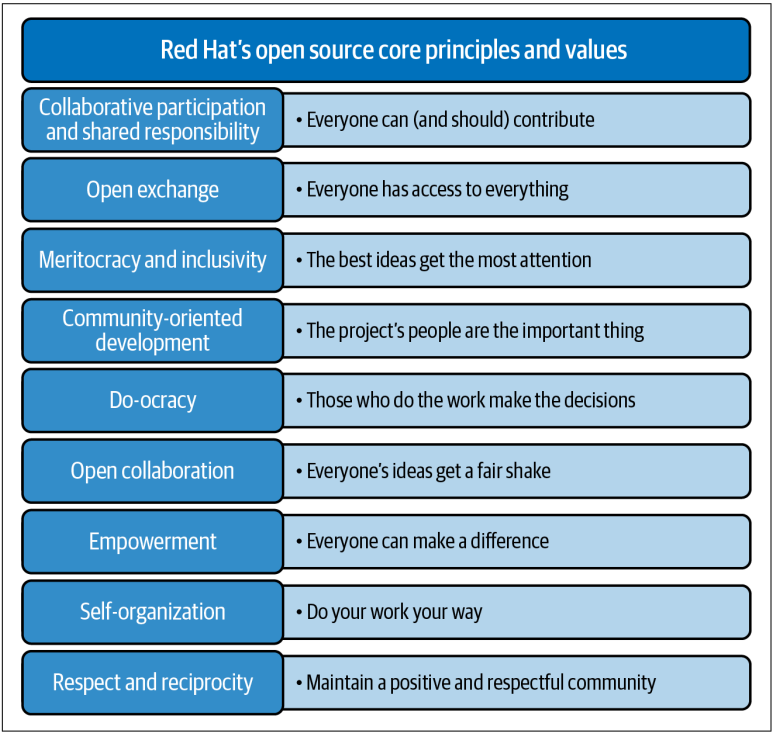

Figure 1-1. Red Hat's definition of open source core principles and values.[2] As you can see, when companies talk about OSS, they are usually talking about much more than just the source code being available.

2 Isabel Lee, "Open Source Culture: 9 Core Principles and Values," Red Hat, October 10, 2024, *https://www.redhat.com/en/blog/open-source-culture-9-core-principles-and-values*.

Linux: The Cradle of Open Source Software

By far, the most well-known (and most widely used) piece of open source software on the market is the Linux kernel. The Linux kernel has been developed by Linus Torvalds since 1991, when he was still an undergraduate at the University of Helsinki. Torvalds was an avid programmer, and as a side project, he wanted to create a free, full-featured operating system that mimicked the functionality of Minix, an educational OS popular at the time. Following OSS principles, he made the software widely available in both binary and source-code formats, and encouraged other programmers to join in and make improvements. In just a few years, Torvalds would see Linux go from his small student side project to being a popular operating system used by millions of people worldwide.

It's important to note that Linux was far from the only software project that was created using OSS principles. The strategy of software development made popular by the OSS model revolutionized how software was created across a plethora of software projects. Before OSS, software was generally built by small teams of experts working to produce a polished, perfect release. In OSS, since the code was easily accessible, people from the world over could, and did, contribute. Changes could be deployed and tested rapidly, and bug fixes could be suggested and deployed rapidly.

This development model was a game changer and a real shock to the software development establishment. Eric Raymond captured the evolution from the old to the new in his famous essay (and, later, book) *The Cathedral and the Bazaar*, which contrasts two distinct approaches to software development. The "cathedral" represents the older, more traditional style of software development where small groups work on code behind closed doors and release it methodically—when it's finished. This, Raymond says, is much like the building of a medieval cathedral. OSS-inspired software development, by contrast, followed the "bazaar" model, which meant everything happened in public, with rapid releases and people coming and going at their own pace and whim—much like a bustling marketplace in the center of a city.

About Linux itself, Raymond said:

> Linux evolved in a completely different way. From nearly the beginning, it was rather casually hacked on by huge numbers of volunteers coordinating only through the Internet. Quality was maintained not by rigid standards or autocracy but by the naively simple strategy of releasing every week and getting feedback from hundreds of users within days, creating a sort of rapid Darwinian selection on the mutations introduced by developers. To the amazement of almost everyone, this worked quite well.[3]

Raymond himself used the bazaar model on his own software project (Fetchmail), and the essay was instrumental in Netscape deciding to make Netscape Communicator open source and, later, in founding the open source Mozilla project.

It helped that there were a large number of other OSS programs that already existed that could be run on Linux. For example, packages from the GNU Project provided crucial capabilities to the Linux user such as compilation, file editing, compression, and more. Similarly, many utilities ended up being drawn from another operating system called BSD (short for Berkeley Software Distribution). BSD is another operating system similar to Linux that also had (and still has) an OSS-style license. The openly available source code allowed Linux developers to recompile (and, in some cases, refactor) these programs, which made this interworking possible and saved a ton of time that would otherwise have to be spent rewriting these utilities.

The Linux project quickly picked up fans and additional programmers who have made it into one of the most secure and resilient operating systems in existence. Its flexibility has made it indispensable across a wide variety of devices—from the world's most powerful supercomputers to tiny Internet of Things (IoT) gadgets. And if you're reading this on a smartphone, there's a good chance it's running Android, which is built on the Linux kernel. In short, Linux isn't just software—it's a cornerstone of modern computing.

Open Source Software Examples

The success of Linux directly led to the success of many other programs. While utilities like compilers and file editors provided

3 Eric S. Raymond, "Chapter 1," in *The Cathedral & the Bazaar* (O'Reilly, 2001).

basic functionality, Linux users also needed tools like web and email servers, databases, and communication software—features commonly found in established operating systems like UNIX and Windows. So they started writing them. Let's look at a few examples. We will start with the primary components of the LAMP stack (short for a system that runs Linux, Apache, MySQL, and PHP), which powered a lot of the websites on the early internet, and then look at some more modern examples such as Git, Kubernetes, and PostgreSQL.

Apache HTTP Server

The Apache HTTP Server Project (usually shortened to "Apache") started in 1995, and within a year or so, it became the most popular web server on the internet—maintaining that status for decades. Apache is still in continuous development, and like many open source projects, it has strong alternatives in the open source ecosystem, such as nginx, which has gained market share as a webserver of choice. Apache also demonstrates another feature of open source software. While the vast majority of instances run on Linux, Apache can be compiled to run on Windows and a variety of other operating systems.

MySQL and PostgreSQL

MySQL and PostgreSQL are both relational databases. (Relational databases work with tables of data that are rigorously structured, linked together for recordkeeping and data science, and powered by the Structured Query Language, or SQL.) PostgreSQL has a history extending all the way back to 1986, when it was being developed to be as powerful and reliable as possible. It is often used for enterprise, mission-critical applications requiring complex queries such as financial systems and analytics. MySQL was designed to be as fast as possible to support read-heavy workloads and web applications. Both databases continue to thrive, with PostgreSQL and MySQL being named as the top two preferred databases among developers.

PHP

PHP began its life as a set of tools created to track website visits. It was quickly expanded upon by enthusiastic users, becoming a fully featured scripting language and an open source project in 1995.

PHP was (and, in some cases, still is) the engine behind many web projects, including WordPress, Yahoo, and Facebook. While not as popular as it once was, it is still going strong in 2025.

Git

Git is an example of version control software that is essential in software development. Version control software programs are structured systems used by teams of developers to track changes, submit potential fixes, and even roll back changes to the source code of a program that turned out to be suboptimal. For years, Linux development relied on a closed source version control program called BitKeeper. BitKeeper had been made available to the Linux developer community—albeit under very onerous terms. In 2005, BitKeeper was made unavailable to the Linux developers due to what the maintainer of BitKeeper, Larry McVoy, determined was a violation of these terms. In response to this, Linus Torvalds himself started developing Git. Git quickly became the standard in version control software, outstripping competitors like BitKeeper, CVS, and Subversion. This was not solely because of its association with Linux, though—Git was just that good. Today there are multiple free and paid ways developers use Git, from standalone on-site installations to cloud-based software as a service (SaaS) tools—the most popular of these being GitHub.

Containers

One of the most transformative technologies to come out of Linux is the container. Put simply, a *container* is an isolated application runtime environment that utilizes a feature of the Linux kernel called control groups (or just cgroups). Cgroups enable strong isolation of different applications running within a single host, similar to how full virtualization enables multiple virtual machines (VMs) to run on one physical server (see Figure 1-2).

We will talk about containers more in subsequent sections of this book.

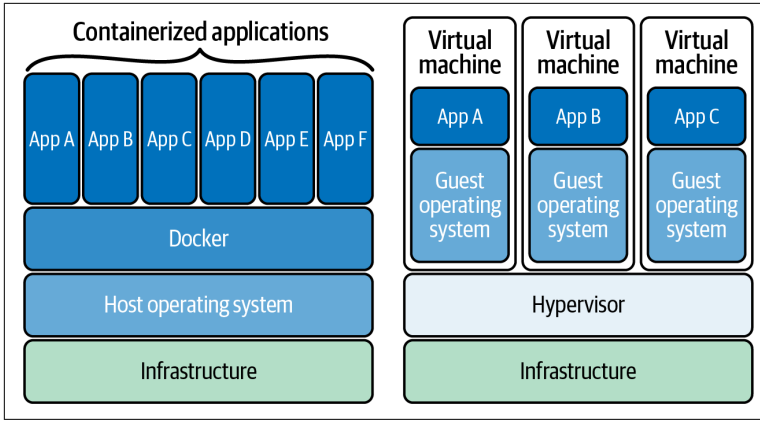

Figure 1-2. A comparison of a containerized system and a system running virtual machines.

Kubernetes

Containers can be a lot to set up manually, especially as the total number of containers you are managing increases. For this reason, container orchestration software like Kubernetes was born. Kubernetes helps manage the task of not only deploying containers but also managing the computers that run the containerized workloads. There is a lot that goes into Kubernetes that we don't have time for here (and a lot that goes into Kubernetes care and feeding), but basically it handles the deployment and management of containerized workloads with a feature set and operational reliability that are impractical to duplicate manually. Kubernetes was founded in 2014, and it has since gone on to become one of the most widely deployed software systems in the world.

Let's switch now, from looking at the wider OSS world as a whole, and take a look at how Microsoft viewed it over time.

Microsoft's Contributions to Open Source

For much of its early existence, Microsoft was strongly against Linux and open source software. To go back to our previous discussion of how software was written, Microsoft was a cathedral company in a sea of OSS bazaars. It was also a major player in the enterprise software space and did not want that to change. Microsoft believed,

like many companies of that time, that keeping source code closed was an essential strategy to protect intellectual property.

To the surprise of basically everyone, however, things did start to change. The rise of the modern internet led to an explosion of OSS usage due to the low cost, ease of access, and rapid updates to projects like the aforementioned LAMP stack. Fighting the prevailing tide became an exercise in futility, and Microsoft began to take a more nuanced stance on OSS. In 2004 Microsoft made the WiX Toolset (a .NET Windows Installer program) open source—the company's first foray into OSS. In 2006 Microsoft hired Sam Ramji as the head of Open Source Strategy with the goal of determining what to do next.

In 2008, just before he retired, none other than Bill Gates championed OSS and said it was essential for Microsoft to embrace it. In 2015, CEO Satya Nadella put out a presentation with a slide saying that "Microsoft ♥ Linux." In 2020, Brad Smith, president of Microsoft at the time, went on record in favor of OSS, saying, "Microsoft...was on the wrong side of history when open source exploded at the beginning of the century."

Microsoft has evolved into a major contributor to the open source community. The company became a platinum member of the Linux Foundation in 2016 and has embraced OSS across its entire portfolio. Many of Microsoft's flagship programming and development tools, such as .NET and PowerShell, have been transitioned to open source and now run natively on Linux. Microsoft actively contributes to open source projects such as PostgreSQL, VSCode, and TypeScript, and the company contributed a significant virtual networking tool called SONiC to the Linux Foundation. This is just a small sampling of the contributions Microsoft has made to open source projects.

Another reason Microsoft changed its approach and its views so dramatically was Azure. Customers were already running substantial amounts of their production environments on Linux, and as they migrated workloads to the cloud, they did not want to shift to a new operating system. Microsoft was well aware of that reality, and as it approached general availability of Azure IaaS (infrastructure as a service) in 2013, Microsoft made it clear to anyone who would listen that Azure would be offering Linux VMs "on day 1." The policy of supporting Linux from the very beginning of Azure has

been wildly successful. Today, over 65% of customer cores on Azure run Linux, and as we will see, many of Azure's service offerings are powered by Linux as well.

Today Microsoft not only supports running Linux on Azure but incorporates Linux and OSS into the very fabric that powers the cloud. From its own distribution of Azure Linux, to the open-sourcing of .NET and PowerShell, to the creation of cloud native projects like Dapr and Radius, Microsoft truly ♥s Linux and open source and has become a core contributor to and champion of the movement.

So now that we have a good background on the history of OSS and Linux, let's look at some of the more popular ways of using Linux on the Azure platform.

Running Linux on Azure

As is true of so many technologies, there's more than one way to run Linux on Microsoft Azure. In this section, we'll review the various options available and examine specific distributions and images you can leverage in your Azure environment.

Options for Running Linux

Microsoft Azure includes a myriad of services to host your applications. This includes infrastructure as a service (IaaS), platform as a service (PaaS), and functions as a service (FaaS). Each of these services allow you to leverage Linux to run your applications with different levels of abstraction. Depending on the service, you can select Linux as your operating system, host OS for containers, or host OS for application runtime.

IaaS options, like Azure Virtual Machines, give you full control over which operating system and version of that operating system is installed. With PaaS options, like Azure Kubernetes Service (AKS), you can select from a variety of supported operating systems for the hosts. FaaS options further abstract the underlying operating system by presenting you with possible runtime options, such as with Azure Functions. Table 1-1 breaks down the level of control you maintain over the underlying operating system on various service types.

Table 1-1. Comparison of OS control on different Azure service types

Service type	Example	Level of OS control
IaaS	Azure VMs	Full control
PaaS	Azure Kubernetes Service	Limited selection
FaaS	Azure Functions	Windows or Linux

We will discuss and compare IaaS and PaaS deployment options in greater detail in Chapter 2. For the time being, just be aware that IaaS provides the most freedom when selecting a Linux operating system and distribution, whereas PaaS has a more restricted set of options for a guided experience. Since FaaS abstracts the operating system almost entirely, we will not be touching deeply on its deployment options.

Source Image Options

The operating systems for IaaS and PaaS options need to come from somewhere, so what are your options for finding and selecting a source image for your compute needs? Microsoft includes several options for deploying an operating system on Azure. To better understand your options, we need to examine the different image types and where they are stored.

Source image types

Microsoft has three different image type categories: Marketplace, community, and custom, which can be further broken down by the image source, licensing, and level of support. Table 1-2 summarizes the following sections for quick reference.

Table 1-2. Comparison of image categories on Azure

Category	Location	Image source	Licensing	Support
Marketplace images	Azure Marketplace	Microsoft	PAYG or BYOL	Microsoft
		Endorsed partners	PAYG or BYOL	Microsoft and direct vendor support
		Partners	PAYG or BYOL	Microsoft and partner
Community images	Azure Compute Gallery	Community publishers	PAYG	Microsoft
Custom images		Self-published	PAYG or BYOL	Microsoft

We'll examine each category in greater detail in the following sections.

Marketplace images. As implied by the name, Marketplace images can be found on the Azure Marketplace. The images come from either Microsoft itself or through a set of Microsoft partners. These images all meet Microsoft's requirements for running Linux on Azure and include reasonable customer support from Microsoft. Reasonable support roughly means that Microsoft will help determine if the issue resides with Azure or the Linux image, and Microsoft will provide support for the image in line with its service-level agreements (SLAs) and a reasonable level of effort. Microsoft may refer you to the Linux vendor if it is unable to resolve the issue.

The licensing for Marketplace images will depend on the image and partner involved. Some images are freely available, while others have a pay-as-you-go (PAYG) or bring-your-own-license (BYOL) option.

There is a special subset of Marketplace images managed and maintained by endorsed Linux distribution partners like Red Hat, Canonical (Ubuntu), SUSE, CIQ (Rocky Linux), Oracle, AlmaLinux, Flatcar, and Credativ (Debian). These images are called platform images, and they are put through additional testing and have a well-defined update cadence as prescribed by the publisher. In addition to the standard level of Microsoft support, platform images also receive support directly from the partner.

Some platform images also include Azure-tuned kernels to enhance and optimize the Linux kernel to run on Azure. These customized kernels include performance enhancements, new features, and a faster update cadence than the default kernels for a given distribution. Running the Azure-tuned kernels provides access to things like full support for Accelerated Networking in Azure and Infiniband and remote direct memory access (RDMA) capabilities for Azure High-Performance Computing (HPC).

Community images. These images are created by the community, including open source projects and teams. They do not appear in the Azure Marketplace but can be found through the portal (if shared to your tenant) or using command-line tools. Anyone can publish community images through the Azure Compute Gallery. Some popular community images include Fedora and CentOS Stream.

It should be noted that Microsoft does not provide support for community images. Unlike platform images, there is no guaranteed level of testing or update cadence for community images. Support may be provided by the open source project or team behind a community image, but that is also not guaranteed.

The licensing for community images depends on the distribution of Linux and the community behind the project. You will want to read the license agreement for a community image before using it.

Custom images. These images are created by Azure customers and are not made available publicly. They can be stored as managed images or on the Azure Compute Gallery. Custom images need to follow the prerequisites identified by Microsoft for running Linux on Azure and often are built from existing platform or community images.

In terms of support, Microsoft will provide a reasonable level of support for the operating system and its integration with Azure. The licensing for custom images is the responsibility of the customer.

Creating images

Custom Linux images can be created using an existing virtual hard disk (VHD) or by customizing and capturing an Azure Virtual Machine. Before capture, you can customize and update the operating system to meet your needs. Once the operating system is ready, you can choose to generalize the image or leave it in its current state.

Generalized images can be modified during deployment to change the hostname or admin user or to run startup tasks during boot. Nongeneralized images are labeled as specialized on Azure and cannot be modified as part of the deployment process. A generalized image is preferred for scenarios where you will be using the same image across multiple VMs.

When creating a custom Linux image for use with Microsoft Azure, you will need to meet the prerequisites for preparing the image. Generalized images require either the Azure Linux Agent or cloud-init for deployment provisioning. Specialized images do not require a provisioning agent, but it is recommended to include it for handling Azure-specific extensions.

If you are using an existing image on Microsoft Azure to create your custom image, the necessary prerequisites will already be present.

Storing images

As mentioned previously, Marketplace images are stored on the Azure Marketplace and published by Microsoft or its partners. Unless you're a Microsoft partner, you won't be able to publish to the Marketplace. Instead, you have two storage options for images: managed images or Azure Compute Gallery.

Managed images. Managed images are a simple option for storing prepared images, but they are limited in terms of functionality. The stored images must come from a source VM or VHD and must be generalized before capture. Images are limited to the region, subscription, and tenant in which they were created but can be moved to another region via a copy (or they can be exported).

Managed images are considered a legacy option at this time. The preferred option is Azure Compute Gallery, which supports additional functionality, like specialized images and cross-subscription deployment.

Azure Compute Gallery. The Azure Compute Gallery is a more robust service that supports more options and provides better performance than managed images.

Azure Compute Gallery can capture images from an existing Azure VM, a VHD, a managed image, or other images stored in a gallery in the same subscription. The Compute Gallery also supports both operating system and data disks to assist with bundling complete applications together in a single image.

The images stored in a gallery can be specialized or generalized, and versioning is supported to assist in publishing updates to an image. Azure Compute Gallery also supports distributing images across regions and making images accessible across subscriptions and tenants. You can even publish your own community images that are accessible to everyone.

For large-scale deployments with frequent provisioning, multiple replicas of an image can be created in a region to support additional concurrent deployments. And galleries support zone-redundant storage to ensure images remain highly available.

Azure Linux

There is one other flavor of Linux that is a bit different from the usual vendor distributions you'll find in the Azure Marketplace, and that is Azure Linux.

Azure Linux is an open source Linux distribution originally developed under the project name CBL-Mariner. The distribution is meant for internal use by Microsoft on Azure and at the edge. Azure Linux is a lightweight distribution serving as a common core for packages to be layered on top, depending on the deployed VMs' intended purpose.

The goal behind Azure Linux is to provide a streamlined version of Linux that has a low deployment footprint, a reduced attack surface, and a focus on security by default configurations. Microsoft is responsible for fully testing and vetting each release of Azure Linux to ensure stability, security, and performance.

Microsoft currently uses Azure Linux internally on the Azure Stack HCI version of AKS, Azure IoT Edge, and most recently as a host option for AKS. Azure Linux Container Host for AKS runs Azure Linux 3.0 as the host operating system for nodes in an AKS cluster.

Summary

As Microsoft's portfolio of Azure services has expanded, so has its support for open source technologies and the Linux operating system. Linux-based workloads now encompass the majority of workloads running on Azure, and Microsoft is dedicated to ensuring that Linux users have a first-class experience on the platform.

By working closely with partners like Red Hat, Canonical, and SUSE, Microsoft can offer customized images for the most popular distributions of Linux that are optimized for performance and security on Azure. Endorsed partners and platform images create an ecosystem that includes additional support, consistent updates, and a stable base layer to build your applications on.

CHAPTER 2

IaaS and PaaS Deployment Options

Cloud services are broadly broken down into three main categories: infrastructure as a service (IaaS), platform as a service (PaaS), and software as a service (SaaS). These categories, as defined by NIST,[1] are intended to organize the cloud computing offerings around what is customer responsibility and what is cloud provider responsibility. The differences are defined in the Shared Responsibility Model of cloud computing. We'll take a look at the model in a moment, but first let's look at the difference between these cloud computing resource categories from a technical perspective.

IaaS assigns the highest amount of responsibility to the customer. In short, IaaS is basically like VMs running in the cloud—there is a defined level of resources (CPU, RAM, hard disk space, etc.) made available to the end user, and the user can install whatever applications they want on those resources, from the operating system down. The end user is fully responsible for the installation, configuration, and upkeep of those applications. IaaS gives the end user a maximum amount of control over the resources they are paying for, but it also requires the most work to keep things up and running. For example, say you have Accelerated Networking

1 It should be noted that while there are many other "as-a-service" options out there, including functions as a service, which were briefly mentioned in Chapter 1, these are the only three that are specifically defined and categorized as discrete cloud services by NIST.

running on an IaaS instance. It is your responsibility to make sure all drivers and kernel levels are correct and are functioning properly. If the drivers fail, it is also your responsibility to fix them. And if you don't, the instance will break (and stay broken until you fix it).

Historically, IaaS was a particularly popular first step for people very new to cloud—after all, it is an easy mental leap to take. Instead of running VMs on premises in your datacenter, you now have them running remotely as IaaS instances in a Microsoft datacenter, without needing to handle hardware refreshes.

At the other end of the customer responsibility spectrum is SaaS. SaaS is essentially a fully built, managed, and configured application that you pay to have access to. You are not responsible for (nor do you have any access to view or configure) the backend hardware, software, or data stores. Office 365 is an example of SaaS software—you pay for a subscription, and in return you get access to Outlook, Excel, PowerPoint, and so on. Where appropriate, you can manage users, create access policy, and shape the experience your users have—up to a limit defined by the SaaS provider. Generally, the only way to get the application to change how it operates is to submit a feature request to the developers and wait.

The whole point of SaaS is that you are hands-off from the underlying infrastructure—you do not have to manage anything that is not directly related to the software. SaaS is not going to feature very much in this book, but we wanted to at least mention it for the sake of completeness—and to help understand a little more clearly how it differs from PaaS.

PaaS is the middle ground between IaaS and SaaS. It's somewhat harder to explain because, as we will see, there is a lot of variability between products that fall into this category. Generally, though, a PaaS platform provides you with a layer of preconfigured infrastructure that you can deploy applications on top of. In a PaaS deployment, you are not responsible for infrastructure management (server provisioning, patching, network configuration, etc.), but you maintain control over the application layer. You are responsible for configuring your application's runtime settings, scaling parameters, and deployment specifications.

A good example here (and one we will talk about in depth later) is a database. Say you want to run Microsoft SQL Server (MSSQL) as a backend for your internet-facing application. If you went with an IaaS model, you would have to provision the instance, install the OS, install MSSQL, bring all patches up to date for both OS and application, and have all of this pass security scans—all before you can allow your database teams access to start configuration and database creation. With the PaaS model (Azure SQL), all of that is taken care of for you. You simply pay for X amount of Azure SQL, it is provisioned for you, and you can start building databases and connecting them to your applications.

PaaS provides a platform for subject matter experts to focus only on their application without worrying about the underlying infrastructure. What you get is a database administrator–level view of a deployed database that is sized to the database administrator's requirements (parameters required, database size, required redundancies, etc.). For a database team, this could be ideal: a database administrator is not necessarily a systems administrator, and may not want to take on the responsibilities that go along with running a full IaaS VM. With PaaS, they don't have to.

In all cases, there are things that will always be the customer's responsibility, and there are things that will always be the cloud vendor's responsibility. The different levels of control and responsibility between IaaS, PaaS, and SaaS are described in the Shared Responsibility Model. Reading Figure 2-1 from right to left, you can see the cloud provider taking on more and more responsibility as you go from a fully on-premises model, through IaaS, then PaaS, then SaaS.

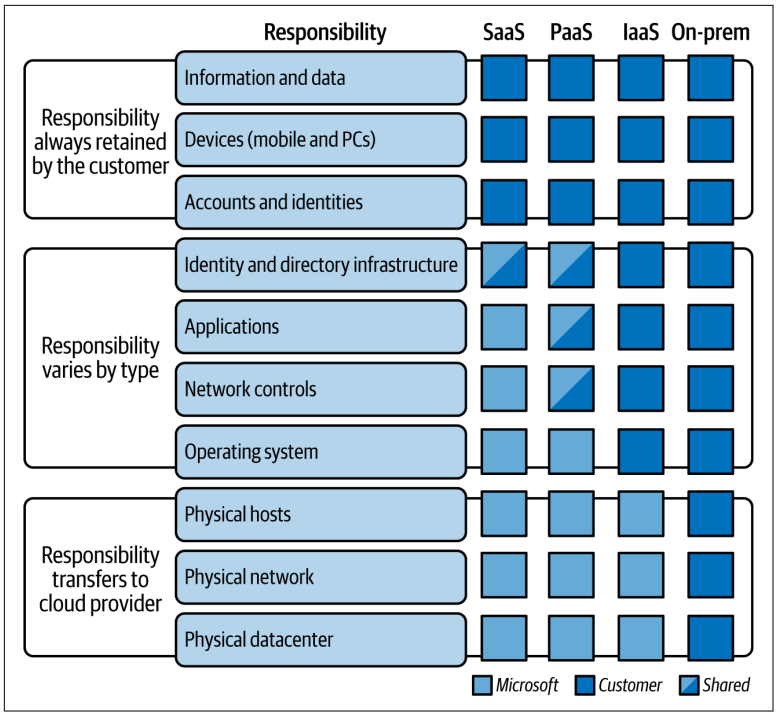

Figure 2-1. The Cloud Computing Shared Responsibility Model

Something that is important to note: when it comes to deciding between IaaS and PaaS, there is no automatic right answer. The infrastructure you deploy should simply be the best infrastructure to solve your particular computing problem while meeting your business and operational requirements. And there's no reason you can't have a mix. In some cases you might need the full flexibility of an IaaS model, while in others a PaaS solution fits the bill.

We should also take a moment to note that in this book, we will not be looking at every single product in the IaaS or PaaS categories—there is simply not enough space. The full list of Azure Products available to you is literally hundreds of items long. What we are going to do is look at some of the most common products in each of these categories and show you how they can be used to solve many of the most common cloud computing problems.

Now, with all of that said, let's take a look at some examples of the IaaS and PaaS solutions that are available on the Microsoft Azure platform.

IaaS

IaaS is a cloud computing model that provides you access to the essential building blocks of computing: CPU, memory, storage, and networking. Examples of IaaS products include cloud-hosted VMs, storage plans, and networking resources such as firewalls and connectivity services. IaaS resources can be consumed on an on-demand, pay-as-you-go basis, or they can be purchased in long-term subscription models such as Reserved Instances. This book will primarily focus on the technology rather than the billing models for these services. However, you can see exactly what the options are for the technology as well as the billing options online at the Microsoft Azure Pricing page, and we will discuss cost optimization strategies in Chapter 8.

It should be noted that Azure Virtual Machines are a key component of the larger category known as Azure Compute. This category contains a lot more services than just VMs, including Azure App Service, Functions, Containers, and more, but in this section we are just focusing on core IaaS. Some of these services will be included in the section dealing with PaaS offerings.

Now, let's take a look at three of the most popular IaaS options on the Microsoft Azure Platform: Azure Virtual Machines, Azure Storage, and Azure Networking.

Azure Virtual Machines

Azure Virtual Machines are, in short, exactly what they sound like: they are VMs that run in Azure. Microsoft has a wide variety of VM options available to the end user, organized by a very specific naming convention. VMs can run x86 or ARM CPUs from a variety of manufacturers, optionally have external GPU access, and can run local disk or remote storage of varying performance levels.

Local disk in this instance means that a small disk is generated and attached to the VM for temporary storage. Local disk storage is temporary and tied directly to the VM instance. It cannot be detached or moved to another VM. "Remote storage" is a storage instance that is instantiated separately from the VM and connected later. Remote storage exists independently of the VM and can be detached, moved to another VM, and in some cases even connected

to multiple VMs simultaneously. We will talk more about remote storage options later on in this chapter.

Azure VM instance naming convention

Figure 2-2 is an example of a common Azure VM, as shown in the Microsoft Azure Pricing Calculator.

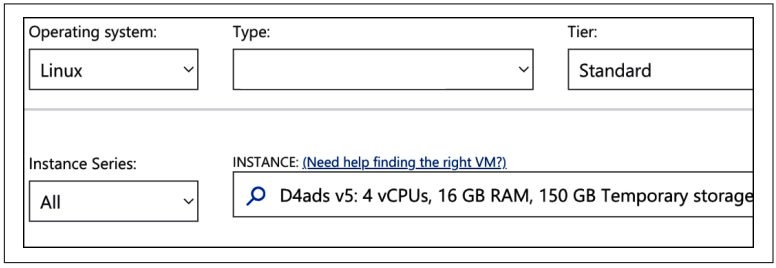

Figure 2-2. A screenshot of the selection options from the Azure Pricing Calculator

In the INSTANCE drop-down, you can see "D4ads v5." That's the name of a particular instance. If you do a little research (or just click more in that drop-down), you will see that there are hundreds of types of VM instances to choose from. So the question is, how to make sense of the differences between them just by the name?

The first letter in the name of any VM is its family. There are quite a lot of these, with common examples being "A" for entry-level, general-purpose workloads; "D" for general-purpose, production-level workloads that are expected to be more demanding; and "N" for GPU-accelerated options where a workload requires one or more GPUs. There are, of course, many others, and the full list of all VM options is available on the Azure Virtual Machines homepage.

The next capital letter (if present) is the name of a subfamily. Some instance types have no subcategories; some have many. The N instance family, for example, has (at the time of this writing) four: NC for compute- and graphics-intensive workloads; ND for large memory versions of NC instances; NG for virtual desktop infrastructure (VDI) and cloud gaming; and NV for video encoding and rendering. These subfamilies are intended for different use cases. Continuing the N-series example, ND is for AI and deep learning, NC is for compute-intensive GPU workloads, and NV

is for visualization and video. NG is less commonly used and is intended for specialized GPU workloads.

As is the case of our example in Figure 2-2, you will sometimes see a number next. This number shows the relative size of the instance in the family—generally based on how many CPUs will be allocated. In the D family, if you want 2 vCPUs, you'll select a D2, and if you want 4, you'll select a D4.

Next up are three optional lowercase letters that will let you know about various features that differentiate the VM. The first letter, if present, indicates which CPU is used. If the letter is not present, it's an Intel x86. If the letter is an "a," it's an AMD, and if the letter is a "p," it is an ARM-based CPU. (There can sometimes be exceptions to this—check the docs or ask Copilot if you are at all unsure.)

The second letter indicates if the system includes local disk ("d"), and the last letter ("s") indicates if it is capable of being connected to premium storage. (For more details on this, please see "Azure Storage" on page 26.) There are more letters that can indicate more discrete features, but these are the most common general-purpose examples.

In rare cases, there can be an underscore followed by a specific piece of hardware that helps distinguish one instance from another. For example, the ND_H100s and the ND_A100s are distinguished by name to let you know exactly which high-performance GPU (or GPUs) will be attached to the instance.

Finally, there is a lowercase "v,"" followed by a number. This indicates the version of the instance, and only appears if there have actually been different versions, with different hardware backing it, over time. Our example in Figure 2-2 is version 5.

So, putting it all together, you could choose an instance size of Das_v5, which is version 5 of the D series, with an AMD CPU, no local disk, and remote SSD. Or if you had a workload that needed serious GPU power, you might select the NVads-A10_v5, which would net you the fifth version of an N-series, subfamily V, with an AMD CPU, local disk, remote SSD, and connectivity to one or more NVIDIA A-10 GPUs.

Azure provides these categorizations (families, subfamilies, etc.) in order to quantify the myriad options that you have available to you as a customer. Still, the wide range of options can be overwhelming.

It can be a challenge to suss out exactly which instance (or instance version) would best suit your needs—and choosing the right VM for your needs is very important. In order to help you select the exact right VM for your workload, Microsoft recommends using Azure Copilot to narrow down the options. This relatively recent feature was announced in December 2024 on the Azure Compute blog post "Using Microsoft Copilot in Azure to Find the Best VM Size for You".

Not every Azure region supports every instance type, nor do they all support every version. Some versions have simply been retired over time. Since the instance versions are dependent on the hardware being available, some regions will get the newest versions faster than other regions. Eventually, as hardware refreshes are completed, the regions get close to parity. This is something that you should pay close attention to—particularly if you are managing deployments across multiple regions. For more detailed information on the availability of these services by region, you can refer to the Azure Product by Region page.

When talking about the instance size families as a whole, it is common for the names to be greatly simplified. For example, you will often see "Dv5" used when someone (or some documentation) is talking about the fifth generation of the D series in general, as opposed to a specific model. So in summary, D4ads v5 = family (D) + CPU count (4) + features (ads) + version (v5).

Azure VM CPU options

As stated earlier, the first lowercase letter indicates which CPU an instance will be running on. Instances that do not have a specific letter will be running an Intel x86 CPU. Similarly, instances that have a lowercase "a" will be running a version of an AMD x86 CPU. In both cases, newer instance versions correspond to newer CPUs. Sticking with our D instance version 4 and 5 example, the Dav4 runs the AMD EPYC 7452, and the Dav5 runs the newer AMD EPYC 7763v.

Some instance families are powered by an ARM-based processor. ARM processors are not suitable for all workloads, as not all OSs (or all applications) can run on them. For situations where they will work, however, the ARM processor option provides a cost-effective alternative to the standard x86 options. In keeping with the theme of

this book, many Linux workloads run just as well on ARM as they do on x86, making the value proposition even more compelling.

VM deployment options

Azure VMs can be deployed in a number of different ways. We talked about the various different options available for deploying prebuilt VMs based on images in Chapter 1, including Marketplace, community, and custom images. You can also simply select a VM instance, deploy it, log on to it, and start with a basic operating system. This is usually the first option people turn to when they start using cloud computing. This has the heaviest lift when it comes to sysadmin responsibilities, and it also gives you maximum control compared to using a purpose-built image.

There are many workloads, however, where customizing each VM OS by hand doesn't make sense at all. This could include a workload with performance demands that grow and shrink over time, such as with an ecommerce site on Black Friday or a big-data batch job that doesn't run 24/7. For this, Azure has a deployment option called Virtual Machine Scale Sets (VMSS).

Not unlike various container management products available on Azure (some of which will be discussed shortly), VMSS is intended to automate the deployment, deprovisioning, and management of a fleet of VMs. VMSS can handle many thousands of VMs, automatically making changes to the fleet based on predefined rules. VMs are deployed based on preexisting templates, which can be updated at your leisure, allowing you to make large-scale changes to your IaaS instances whenever major updates to OS or applications are required. VMSS can also mix and match a number of ways to pay for VMs, which makes cost optimization an important part of the VMSS deployment process.

Azure Linux Agent–VM extensions and kernel integrations. Azure VM extensions are small applications that run post-VM deployment to do any number of things to the new VM. These can be prebaked Microsoft extensions, or they can be provided by a third party. Some extensions can be used to install software, for example, or run scripts to create or modify configuration files. In order to run extensions on Linux VMs in Azure, you first need to have the Azure Linux Agent installed. The Agent is preinstalled on Azure Marketplace images, and for custom images it is free to use and can

be added to supported OSs as needed. Assuming all prerequisites are met, extensions can be run from the command line, as part of a template deployment, or from the portal. Extensions can also be updated and then applied to existing systems automatically or manually, depending on your needs.

These scripts can be custom written for your environment and deployed en masse—again, via automated or manual means. Much like any large-scale update, it is important to make sure that the code is good before deploying. Full details and documentation on the intricacies of writing Linux extensions can be found on the Microsoft Learn page for custom scripting on Linux.

In addition to enabling the VM extensions, the Azure Linux Agent customizes the kernel via certain configuration changes. Microsoft also publishes what it calls "tuned" Linux kernels that are optimized for Azure. These kernels are available on the Marketplace images for supported Linux distributions, and were developed alongside distribution partners such as Canonical and Red Hat to ensure optimal performance and make sure features like advanced networking are supported out of the box. This includes support and drivers for things like Infiniband, the Data Plane Development Kit, and the Microsoft Azure Network Adapter (this is a preview feature as of the time of this writing), all of which enhance the performance of VMs compared to the vanilla Linux kernel.

Azure Storage

Azure Storage is another fundamental Azure product. As we discussed in "Azure Virtual Machines" on page 21, VMs have two primary options for storage: local disk and remote disk. In both cases, these options appear to the VM as a directly attached disk. There are quite a few other options available, however, such as file-level storage, which we will discuss in detail within this section. (It should be noted that there are many others—such as Azure Storage filesystems for AI and Azure Elastic SAN—which are available but beyond the scope of this book.)

Storage types

There are three types of storage under the Azure Storage umbrella that you need to understand before we talk about the products themselves: file, block, and blob.

The lowest-level option from a technical perspective is block storage. Utilizing block storage requires the end user to provision the storage, attach it to a functional system, and create a filesystem. Much like IaaS versus PaaS, block storage provides the highest level of customization and control at the expense of configuration, expertise, and time requirements. These are most commonly used at the VM level when performance is a priority. Although this is rare, some specific applications can gain even more performance by writing directly to block storage without a filesystem—something that can only be done with this storage type.

In contrast, you do not have to create filesystems when file storage is provisioned. The storage is provisioned by size, and then can immediately be used to hold files and the metadata about files (permissions, time stamps, etc.) in a directory-based structure. All of the underlying disk management is handled by the systems and not by the end users. This is used most commonly for fileshare workloads and allows very easy migration from traditional application fileshares into cloud services.

Blob storage (also sometimes known as object storage) is another higher level of abstraction. Blob storage treats all items added to the provisioned storage account as a self-contained entity in an unstructured fashion. The end user can create logical folders or other file groups, but these separators do not actually exist—they are merely delimiters or identifiers in the blob's metadata. Blob is used to store massively large files (or massive amounts of files) and to provide read access to them in turn.

Now let's look at the Azure offerings that match up with each of these storage types.

Azure Files

Azure Files, as the name suggests, is file-level storage on Azure. A user simply provisions the amount of storage that they want, and then provides one or many systems with access to the share. Azure Files shares can be accessed using Server Message Block (SMB), Network File System (NFS), or the Azure FileREST API. Linux systems can connect to Azure Files shares using SMB or NFS, provided you are using a supported version of the kernel and have the other prerequisites covered (primarily installing the cifs-utils package). All relevant ports will need to be opened, usually 445 for SMB and 2049 for NFS.

Azure NetApp Files

Azure NetApp Files (ANF) is a high-performance file storage service designed to meet the needs of Linux workloads on Azure. Users can provision the amount of storage they need and provide access to one or multiple systems. ANF supports multiple protocols, including NFSv3, NFSv4.1, and SMB, and even allows simultaneous multiprotocol access, making it versatile for various Linux applications. Linux systems can connect to ANF shares using NFS, provided they are using a supported version of the kernel and have the necessary prerequisites covered. This includes configuring the appropriate NFS mount options, such as nconnect, to optimize performance.

Azure Blob

Azure Blob storage is Microsoft's blob/object storage solution. It was built specifically to store massive volumes of unstructured data without the user having to worry about filesystem limitations.

Blob storage is structured differently than traditional file storage. At the top level, you need a storage account, which is a unique namespace in your tenant. Inside this will be containers, which you can roughly think of as directories. Finally, in the containers will be the "blobs"—the files themselves. An image of this structure is shown in Figure 2-3. There can be an unlimited number of containers, and each container can contain an unlimited number of blobs. Note that while the number of blobs is unlimited, the size of blobs is not.

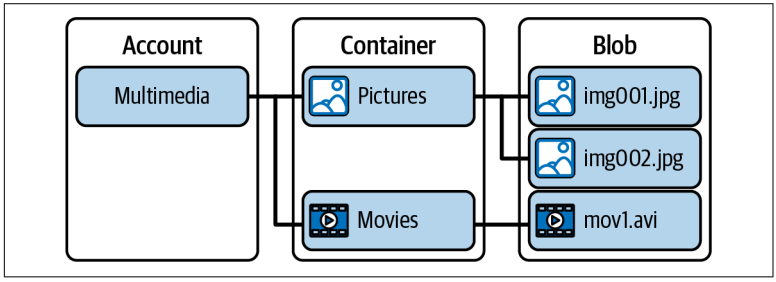

Figure 2-3. A simple breakdown of blob storage structure containing some organization for images and for movie files

Blob storage can be accessed via the REST API. One example of this could be from the URL *https://[STORAGE_ACCOUNT_NAME].blob.core.windows.net*. (Note: This is only one of many possible DNS endpoints—the URL for your specific deployment may vary.)

Storage accounts determine the type of blob storage available in each container. You will likely create a number of storage accounts and containers in order to account for job separation as well as cost management—not all blob storage is priced equally.

NOTE

The name for your storage account must be unique across all of Azure Public Cloud. This is because the storage account can be publicly accessible on the internet, so this uniqueness prevents URL collisions.

You will need to create at least one container, but again, you will likely end up with many. Containers control access to all the blobs below them, so these are an important security feature as well as an organizational one.

There are three types of blob storage available in Azure:

- *Block blobs* are the most fundamental level. They are made up of blocks of data that can be individually managed. These are intended for text and binary data. The maximum size of a block blob is approximately 190.7 TiB.
- *Append blobs* are similar to block blobs, but as their name indicates, they are built for append jobs. An example of an append job would be a target for log files from VMs.
- *Page blobs* are built for random access. Blobs in this storage type can be up to 8 TiB. These are used as hard disks (VM disks, or VMDs) for VMs.

Local disk

Unlike remote storage (such as VHDs), local disk does not require a storage account or container to be created. Local disk is created in the same space as the VM. It is ephemeral, meaning data saved on it cannot survive if the VM is deleted. Additionally, many features, such as image capture, snapshots, encryption, and Azure Backup, do not support local disk.

IBM Storage Ceph

Ceph is a scalable and open source storage package. Ceph was built to be a highly available distributed filesystem that provides reliability and persistence across multiple nodes. The deployment of Ceph is complex and outside of the scope of this book, but at a high level, a Ceph deployment includes:

- A Manager, which keeps track of the location of all nodes, and processes within the nodes. It also handles CLI queries about the cluster.
- A Monitor, which keeps the master copy of the cluster map.
- Ceph Object Storage Daemons (OSDs), which host copies of the actual data in the cluster and make it available when requested by the Ceph client.

There can be one or many of all of these Ceph components (and usually there are), which work together to optimize performance and accessibility to data wherever access is allowed. This allows Ceph to be built across environments to ensure organization-wide accessibility to data.

Ceph is a major storage product in the Linux/OSS world, and while it is not directly offered as a product on Azure, a cluster could certainly be built with the VMs and storage types that Azure makes available.

Azure Virtual Network

So you have your VMs, and you have your storage all ready to go. How do things get connected? Azure Virtual Network is the service that you will use to do public and private networking in Azure. Private networks connect all of your resources internally, with a number of gateway options and load balancers available to enable connections both out to and in from the internet.

VNets

A virtual network (VNet) is a logical private network forming a routing and security boundary around your applications. A VNet is built inside of a resource group, and then the VNet is subdivided into any number of subnets that will actually host resources. See Figure 2-4 for an example of this hierarchical structure.

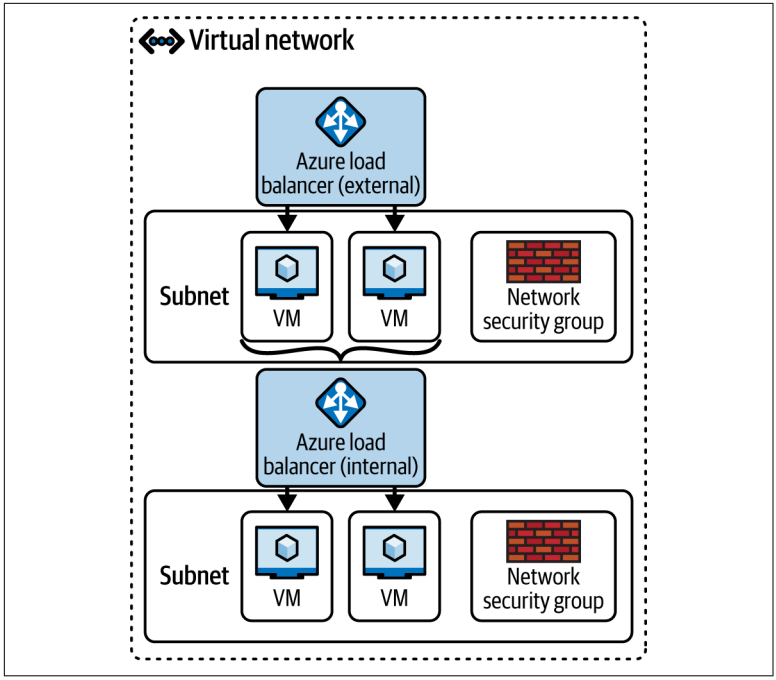

Figure 2-4. Sample image of a VNet containing subnets and VMs

Augmenting VNets and subnets are Network Security Groups (NSGs). These can be applied to several different types of Azure resources, including subnets and individual VMs. NSGs operate like a very basic firewall and define the allow/disallow rules for access from any network that is routed into the protected Azure resource, and they generally are created as default deny. It should be noted that Azure best practice is to have as many access rules attached to the subnet-level NSG as possible and to only use the VM-level rules as a last resort.

If you want more sophisticated security to be applied to your network traffic, Azure products such as Azure Firewall, Azure Web Application Firewall, and third-party network security virtual appliances are also available (subject to additional cost).

VM network interfaces

When you create a VM in Azure you have a few options for configuring network interface cards (NICs). Each will require the standard network configuration information, including IP and DNS settings,

and any NSG rules that you would like to apply to the NIC. You can use Azure-managed DNS servers or optionally use your own custom DNS servers. In the latter case, it is crucial that proper network access has been configured so that your VMs can access the external DNS, especially for DHCP-based deployments. You could have thousands of VMs fail to deploy properly if they cannot get to an IP address as they are expecting to.

Azure has Accelerated Networking available on supported VMs, which is an NIC type that greatly improves network performance. To achieve this, these NICs employ a feature called single root I/O virtualization (SR-IOV). SR-IOV is an extension of the PCIe (or Peripheral Component Interconnect Express) specification that allows network traffic to bypass the software switches in a virtualized environment—in this case, Azure's Hyper-V. Removing these additional steps greatly enhances performance. Full details of SR-IOV as implemented by Microsoft are available at the SR-IOV Overview page in the Azure Virtual Network documentation. Accelerated Networking is available for both Windows and Linux.

The NIC type and performance depends on its host VM's instance family as well as the size of the VM. Some VM instances, such as A version 2 and B version 1, do not have access to Accelerated Networking. Also, larger VMs (that is, VMs with more CPU and RAM allocated) will usually be allocated more bandwidth compared to smaller VMs.

For example, a Dv5 instance (intended for production-level general-purpose computing) maxes out at a very fast 60 Gbps on connected NICs, while an HBv4 instance (intended for high-performance computing with Infiniband connections for maximum network throughput) can do 400 Gbps!

So now you have the high-level knowledge of the major IaaS components. With just these three Azure services, you could build an entire infrastructure for your company or organization—and many have. These IaaS components are often referred to as primitives, as they are the fundamental building blocks of cloud infrastructure, and applications can be built on just these three as a foundation. Now, let's leave primitives aside for a moment and start to look at some services that are a little bit more abstracted from the primitives—Azure's PaaS offerings.

PaaS

Turning our focus to PaaS, for the sake of brevity we are going to focus on two platform types that are most relevant to open source and Linux-based workloads: containers as a service and databases as a service.

Container Deployment Options

Containers are a modern, cloud native option for developing and deploying applications using lightweight containers that share a common host system and kernel. The use of containers can speed up application development, simplify horizontal scaling, and enable the use of microservices in software.

When you think of running container-based workloads, you probably think of using Kubernetes. While that is an option, Azure offers a myriad of ways to deploy containers on the platform. In true PaaS fashion, each container deployment option has different levels of control and responsibility for the infrastructure layers supporting the containerized application.

Azure Container Instances

Azure Container Instances (ACI) is Microsoft's lightweight, serverless platform for running containers in the cloud, designed to make deploying and managing containerized applications as simple as possible. With ACI, you don't have to manage any infrastructure—you can just spin up containers on demand, pay only for what you use, and let Azure handle the rest. It's especially well suited for quick, short-lived workloads or situations where you need to scale dynamically without committing to a full Kubernetes setup.

ACI supports the deployment of containers using popular Linux distributions, leveraging the flexibility and compatibility Linux brings to modern development. The service integrates with Azure Container Registry or any other container registry. ACI does not support ARM-based container images, so you are limited to x86 images only. When provisioning a container instance, you can select the number of CPU cores, the amount of memory, and the source image, which determines whether it runs on a Windows or Linux host.

Much like Kubernetes Pods, container instances can include multiple containers launched together as a container group. The

container group shares the same host, local network, and storage. ACI also supports the use of Azure File shares for persistent storage, placement of container instances in an Azure virtual network, and the association of a managed identity to interact with other Azure services.

Azure Functions

Another serverless option that can run inside containers is Azure Functions. Azure Functions is based on event-driven triggers and bindings that execute your code on demand with little overhead. There is no need to manage any of the underlying infrastructure, including the operating system, application, runtime, and so on. Occasionally, the networking components can require some user configuration to enable proper access and the like.

Azure Functions is optimized for stateless tasks and scenarios where you can get away with deploying just your code rather than a full container. However, if you need control and customization of the container runtime but want to stay in the Functions framework, you can run Azure Functions in a container.

Functions require a hosting plan for deployment, and the Premium and Dedicated plans both include support for running Linux containers. It is also possible to use Azure Container Apps to host Azure Functions containers, providing greater scale, the ability to scale to zero, and dedicated hardware and GPUs. Speaking of Azure Container Apps…

Azure Container Apps

Azure Container Apps is a fully managed container service designed for microservices and application hosting. It sits between the simplicity of ACI and the complexity of AKS, which we'll discuss in depth in the next section, offering an environment for running containerized applications without requiring you to manage the underlying orchestration. With Azure Container Apps, you get built-in support for features like autoscaling, HTTP-based ingress, and seamless integration with Dapr (Distributed Application Runtime) for state management, pub/sub messaging, and more.

Unlike ACI, which is ideal for quick, short-lived workloads, Azure Container Apps is better suited for hosting long-running services or applications that need to scale dynamically based on demand.

Container Apps leverage workload profiles for access to dedicated hardware, including GPUs. Consumption workload profiles allow you to scale to zero, to reduce costs when the application isn't needed.

Even though Container Apps runs on Kubernetes under the hood, you cannot directly interact with the underlying Kubernetes API. Container Apps abstracts the underlying complexity, making it much more approachable for developers and administrators who don't want to deal with Kubernetes management. This abstraction does place limitations on what is available, and sometimes running a full-blown AKS cluster is the best option.

Azure Kubernetes Service

Kubernetes has emerged as the preferred orchestration model for running container-based applications. However, self-managing Kubernetes clusters effectively imposes a high administrative burden on organizations.

AKS is Microsoft's managed Kubernetes offering, designed to simplify deploying, scaling, and managing containerized applications in the cloud. It provides the full power of Kubernetes but with much of the operational complexity handled for you—things like managing the control plane, patching, and upgrades are taken care of by Azure. This lets you focus on deploying and managing your applications while still leveraging Kubernetes's flexibility, scalability, and ecosystem.

Kubernetes was built with Linux containers in mind, so you can run almost any Linux-based application in a containerized format, from simple APIs to complex, distributed microservices. AKS makes it easy to deploy workloads by integrating with container registries like Docker Hub and Azure Container Registry, while providing advanced capabilities like multinode pools, allowing you to isolate Linux workloads or optimize for specific resource needs.

AKS clusters leverage IaaS components, like the Azure VMSS we discussed earlier, to provide hosts with the cluster. Much like our earlier examination of IaaS integrations, AKS can take full advantage of the rest of Azure's services. For example, you can connect workloads to Azure services like Application Gateway, Azure Monitor, and Azure Storage.

AKS supports the vast majority of Azure VM families. That includes support for features like GPU-enabled nodes for machine learning or AI workloads. You can also choose between x86 or ARM-based VMs for more flexibility and potentially lower costs for workloads that support ARM.

AKS offers two options for a Linux-based host operating system on node pools: Ubuntu or Azure Linux. The Ubuntu version is a security-optimized version of Ubuntu LTS 22.04, with some unnecessary kernel module drivers disabled to reduce the attack surface. As an alternative to the standard Ubuntu image, Microsoft has developed Azure Linux based on CBL-Mariner, an open source Linux distribution created by Microsoft.

The Azure Linux container host has been specifically tailored for running container workloads on Azure, with all unnecessary components stripped out and additional security controls put in place. The Linux and AKS teams at Microsoft work in tandem to build, sign, test, and validate the host packages from source. The resulting OS has only five hundred packages included and takes up 5 GB of space on disk. The image ships with containerd for the container runtime and the upstream Linux kernel to maximize compatibility with existing container images.

While AKS provides a lot of power and flexibility, it can still be complex for smaller teams or those without deep Kubernetes expertise. AKS Automatic is a new managed mode of Azure Kubernetes Service designed to simplify Kubernetes operations for teams of all sizes while still maintaining full Kubernetes API access and flexibility. AKS Standard offers flexibility and control for custom configurations, while AKS Automatic prioritizes simplicity and best-practice defaults for faster low-ops deployment.

Database as a Service (DBaaS)

Since many containers are stateless in nature, persistent data needs to reside somewhere. That somewhere is usually a database solution. Broadly speaking, there are two common types of database solutions: relational and NoSQL.

Relational databases make use of tables to store information and create relationships between tables through the use of primary and foreign keys. SQL is most commonly used to interact with relational databases, and so they are often referred to as SQL databases.

In contrast to relational databases, NoSQL databases refer to any database structure that is not relational in nature. This could refer to a database that favors the use of key-value pairs, wide columns, or graphs. For data that is not formally structured and does not adhere to the normal forms outlined by relational databases, NoSQL ("not only SQL") databases are an excellent option.

Whether you choose a relational or NoSQL database solution for your application, deploying and managing your own database servers requires much more than simply installing the software on an Azure Virtual Machine. In an IaaS context, you are responsible for deploying the correct size and type of VM, selecting the correct operating system image and keeping it patched, and deploying and configuring the database software on top of that VM. And that's just a single database server! If you need additional functionality, like clustering or sharding, you're responsible for deploying a fleet of VMs and correctly configuring the cluster networking and replication.

DBaaS removes the additional management overhead, and enables you to simply select the database types and flavors that work best for your application. Azure offers DBaaS options for both traditional relational and NoSQL databases. Table 2-1 details the various DBaaS options sorted by database type.

Table 2-1. PaaS database services available in Azure and their database types

Service	Database type	Open source?
Azure SQL	SQL	No
Azure SQL Managed Instances	SQL	No
Azure SQL Hyperscale	SQL	No
Oracle Database@Azure	SQL	No
Azure Database for PostgreSQL	SQL	Yes
Azure Database for MySQL	SQL	Yes
Azure Cosmos DB	NoSQL	No
Azure Managed Redis	NoSQL	Source available
Azure Managed Instance for Cassandra	NoSQL	Yes

Since this guide focuses on open source and Linux-based solutions, we are going to examine the features and functionality of the open source database solutions on Azure.

Azure Managed Open Source Databases

As you may have gathered from the naming convention, Azure Database for MySQL and Azure Database for PostgreSQL share a common set of features and functionality. Both offer a fully managed database service that includes:

- High availability across multiple zones
- Automated patching within a maintenance window
- Automatic backups
- Virtual network integration
- Enterprise-grade security
- Monitoring and alerting

While there are some key differences in the underlying architecture that drives the two database engines, the choice of one versus the other comes down to your use case and application requirements.

Azure Database for MySQL. MySQL is one of the most popular open source relational database engines, widely used for web applications, particularly with LAMP stacks. Azure Database for MySQL provides a highly reliable and scalable solution for running MySQL workloads. It's an ideal choice for a broad range of use cases, including web applications, content management systems (like WordPress), and applications built with PHP, Python, or Ruby.

Its widespread adoption means there's a vast ecosystem of tools, libraries, and integrations, making it easy to find resources or community support. MySQL's maturity and stability make it a dependable option for many applications.

The architecture of Azure Database for MySQL-flexible server (shown in Figure 2-5) combines the use of an Azure Virtual Machine with Azure premium storage for data files and logs, with zone redundant storage (ZRS) being used for backups.

The server can run in a high-availability mode for autofailover or a scalable mode with a fully managed read replica located in either the same or a separate availability zone, depending on resiliency and latency requirements of the application workload.

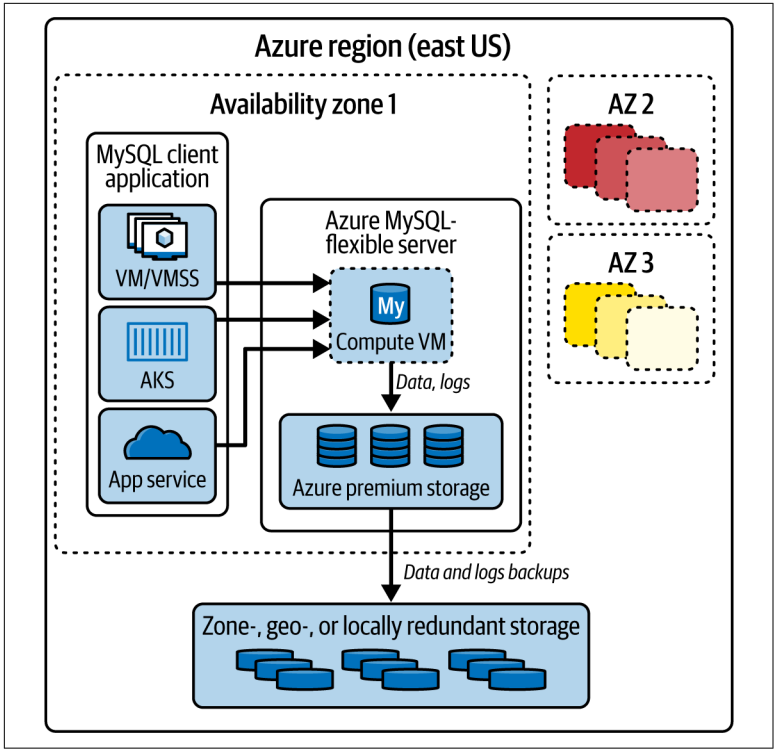

Figure 2-5. Microsoft's reference architecture for Azure MySQL-flexible server

There are three tiers of compute: Burstable, General Purpose, and Business Critical.[2] You can select the tier that most closely matches your application profile and scale your performance as needed on demand. You can also start and stop the server to save on costs for development, testing, or scheduled workloads.

Azure Database for PostgreSQL. PostgreSQL is a highly advanced, feature-rich relational database known for its standards compliance and extensibility. Azure Database for PostgreSQL offers a powerful platform for complex AI applications and agents, analytics, and use cases requiring advanced data types, indexing, and querying capabilities. PostgreSQL supports JSON, vector data (via pgvector), and

2 Microsoft has stated that "Business Critical" will be renamed as "Memory Optimized" sometime in the near future.

geospatial data (via PostGIS), making it ideal for applications requiring hybrid relational and nonrelational data storage or geospatial analytics.

PostgreSQL is a great fit for modern applications, financial systems, and applications requiring high consistency and complex queries. Additionally, PostgreSQL's support for advanced extensions and custom functions sets it apart from other solutions.

The architecture of Azure Database for PostgreSQL-flexible server shown in Figure 2-6 separates the compute from data storage, using one or more Linux hosts for the database engine and Azure zone-redundant storage for the data files. The service can be provisioned to support a single availability zone or multiple zones through the use of a warm standby.

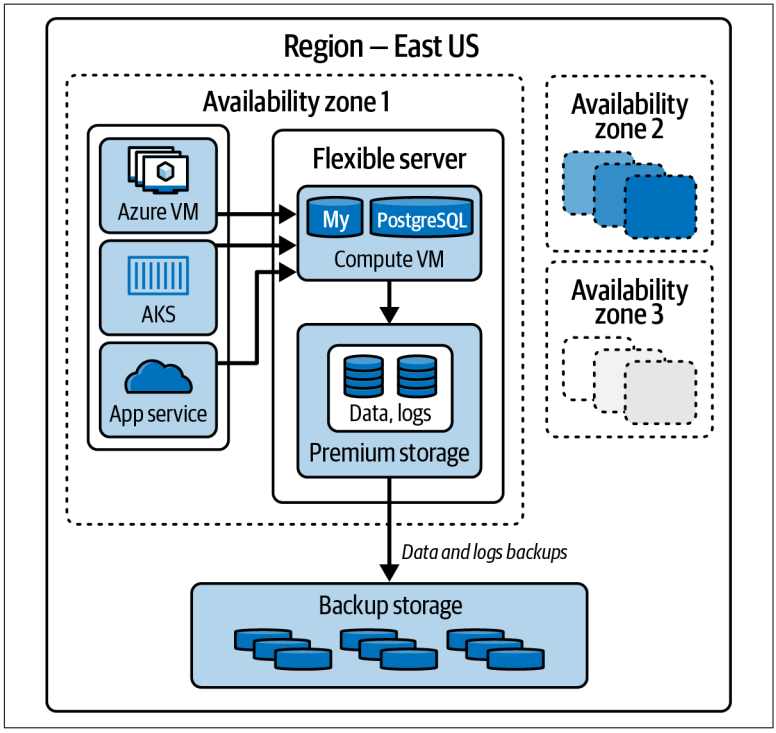

Figure 2-6. Microsoft's reference architecture for high availability with Azure Database for PostgreSQL-flexible server

There are three tiers of compute: Burstable, General Purpose, and Memory Optimized. You can select the tier that most closely

matches your application profile, and start and stop the server on demand to save on costs for development, testing, or scheduled workloads.

Azure Managed Instance for Apache Cassandra

Azure Managed Instance for Apache Cassandra is a fully managed, cloud native database service designed for running Cassandra workloads at scale with minimal operational overhead. Apache Cassandra is widely recognized for its high availability, fault tolerance, and ability to handle massive amounts of data across distributed clusters. By offering a managed version of Cassandra, Azure simplifies operations like provisioning, scaling, patching, and backup management, so you can focus on your application rather than the infrastructure.

The service uses Azure VMSS provisioned in an existing or new virtual network to create Apache Cassandra datacenters. The operating system and Cassandra software are patched on a regular basis using rolling upgrades of the VMs in the cluster. You can easily scale the cluster as needed through a simple command to add or remove nodes from your Cassandra ring.

One of the standout features is its hybrid and multicloud flexibility. You can extend or integrate your existing self-managed Cassandra clusters with Azure's managed service, creating a seamless hybrid deployment. This lets you scale your workloads dynamically into the cloud while still maintaining control over your on-premises clusters. The service also integrates natively with Azure Monitor for observability and supports virtual networks for secure, isolated deployments.

Summary

IaaS, PaaS, and SaaS define different levels of control and responsibility in cloud computing. IaaS gives customers full control over compute, storage, and networking but requires managing the OS and applications. PaaS removes infrastructure management, letting customers focus on application deployment and configuration. SaaS is fully managed by the provider, with customers only using the application as delivered. Organizations often mix these models—using IaaS for customization, PaaS for rapid development, and SaaS for turnkey solutions.

On Azure, IaaS offerings like Virtual Machines, Storage, and Networking provide the building blocks for custom environments. PaaS options, such as Azure Container Apps, AKS, and managed databases like PostgreSQL, MySQL, and Cassandra deliver preconfigured platforms that simplify scaling and operations. The choice between IaaS and PaaS depends on workload needs, desired control, and management overhead, with Azure enabling a tailored approach for performance, cost, and agility.

CHAPTER 3

Red Hat Enterprise Linux for Azure

As you move to adopt the cloud for your computing needs, you'll find that you have a large assortment of choices when it comes to Linux-based operating systems. In this chapter, we'll take a look at the benefits of choosing Red Hat Enterprise Linux (RHEL) as your de facto operating system in Microsoft Azure. We'll also endeavor to provide some practical advice and gotchas as you adopt RHEL for Azure.

Why Red Hat Enterprise Linux

Red Hat works closely with Microsoft to ensure their solutions are optimized for cloud performance, and RHEL is no exception. The golden images offered in the Azure marketplace have been developed by tenured engineers at Red Hat and Microsoft to maximize performance while maintaining security.

RHEL for Microsoft Azure comes with a wealth of guidance and experience from both teams when it comes to running applications at scale. Microsoft has worked to ensure that tools like Azure Monitor in Microsoft Azure integrate well with RHEL and that you'll have a first-class experience when deploying RHEL in your Azure environment.

If you're running RHEL on premises today or in other cloud environments, choosing RHEL for your Azure environment allows you to take advantage of the infrastructure you've already built and the experience you've gained from running Red Hat systems.

Management and automation tools like Red Hat Satellite and Red Hat's Ansible Automation Platform can easily be extended into your Azure environment, providing you with a consistent hybrid-cloud experience.

Running RHEL on Azure

Running RHEL for Azure is not limited to a single image or version from the Azure Marketplace. Red Hat and Microsoft have teamed up to allow you to run RHEL your way, offering multiple options when it comes to licensing, support, and management. Let's start by looking at the options when it comes to the base image you'll use to deploy Azure VMs.

Images

There are two kinds of images available in the Azure Marketplace: pay as you go (PAYG) and bring your own subscription or license (BYOS/BYOL). We'll discuss the difference between the two image types in more detail in "Licensing and Subscriptions" on page 45.

As we mentioned in Chapter 1, Red Hat is one of the platform image partners that Microsoft works with to create endorsed distributions. The images offered by Red Hat in the Azure Marketplace are endorsed, meaning that they must meet certain requirements.

Endorsed distributions are updated on a regular cadence to include the latest patches and security fixes. Red Hat and Microsoft engineers worked together to preconfigure RHEL images on Azure to provide a stable and consistent surface to build on without having to worry about the underlying technology. They jointly perform testing and provide support for these endorsed distributions. RHEL uses the same kernel everywhere, which makes it ideal for hybrid and multicloud environments. RHEL for Azure provides built-in tooling to create templated images for use in Azure, other clouds, and on premises. Red Hat and Microsoft also perform additional testing and provide support for endorsed distributions.

All RHEL marketplace images also include the Hyper-V drivers to run in Azure, the Azure Linux Agent, and a predefined partition layout—which can be customized by selecting RAW from the Azure Marketplace.

You can also choose to run your own custom image on Azure. This is especially useful if you need to heavily customize your operating system to meet certain standards or hardening requirements. If that's the case, you'll need to build and capture an image, installing all the required packages and services necessary to run RHEL for Azure. For more information, check out Red Hat's guide to building custom images.

Currently, RHEL versions 7, 8, 9, and 10 are available from the Azure Marketplace. In case you think that's a typo, RHEL 7 is indeed available and supported on Microsoft Azure under Extended Lifecycle Support (ELS), an optional subscription available from Red Hat. If you need somewhere to run your end-of-life versions of RHEL, Microsoft Azure has you covered.

Licensing and Subscriptions

As mentioned in the previous section, there are two image types available from the Azure Marketplace: PAYG and BYOS/BYOL. The major difference between these two images are how they receive updates and how you pay for them.

PAYG images are available in the Azure Marketplace. They are preconfigured to receive updates and CVE (common vulnerabilities and exposures) patches from the Red Hat Update Infrastructure—an Azure-hosted mirror of the Red Hat update servers with additional Azure-specific content—and both Azure-provided support and the Red Hat subscription cost are bundled into what you pay for the Azure VM. You do not need to register Azure VMs provisioned using PAYG images to Red Hat, and they do not count toward your current entitlements. Since the Red Hat subscription is included with the infrastructure cost of running the VM, PAYG images will cost more to run than BYOS-based images.

There is also an option to purchase PAYG instances directly from Red Hat. This will provide you the opportunity to use prenegotiated pricing and discounts, and RHEL support is delivered directly by Red Hat. Infrastructure support is still delivered by Microsoft, and the cost of the infrastructure remains the same, independent of the subscription cost for RHEL.

BYOS images can take two forms: Red Hat Gold Images and custom images you have published to a private image gallery. Gold Images are hosted in the Azure Marketplace, and are only available after

you have enabled your existing Red Hat subscriptions for Red Hat Cloud Access in Azure. Custom images are built or captured by you and hosted in an Azure Compute Gallery.

BYOS images are not automatically registered with the Red Hat Update Infrastructure in Azure. It's up to you to register the Azure VMs, by provisioning a BYOS subscription and associating it with Red Hat Subscription Manager or a Satellite instance to receive updates.

The cost of an Azure VM running a BYOS image only includes the infrastructure cost of the virtual machine and not the cost of the subscription, making it less expensive to run compared to PAYG images, at least from an Azure consumption standpoint.

It's important to note that you can potentially be double-billed for your RHEL Azure VMs! If you select a PAYG image and then register it with your Red Hat subscription, you are now paying for two RHEL subscriptions for a single VM. The good news is that you can convert a PAYG-based VM to BYOS through the Azure Hybrid Benefit option. The license change can be performed using the Azure CLI and doesn't require downtime. You can learn more about the Azure Hybrid Benefit option in Chapter 8.

Gold Images from the marketplace and custom images will not automatically register themselves with your Red Hat subscription. Red Hat has documented a process for enabling automated registration by building a custom image from a Gold Image or by updating your existing custom images. You could also leverage Ansible Automation or inject custom scripts at startup to perform the registration process.

Support

Support for RHEL for Azure is led by Microsoft for images sold by Microsoft and by Red Hat for images sold by Red Hat in the Azure Marketplace. In practical terms, if you're using one of the standard PAYG-type instances where the price of RHEL and the instance are rolled together, your first and second levels of support are delivered by Microsoft, who not only can troubleshoot the Azure infrastructure and services but also are experts in providing support to Red Hat products. If a bug is uncovered, Microsoft will escalate the issue to Red Hat on the customer's behalf, and Red Hat and

Microsoft will work together on finding a fix and getting it pushed out to customers.

If you choose a Red Hat–provided image from the Azure Marketplace, support for the operating system is provided directly through Red Hat. Microsoft will assist with support issues relating to the Azure platform, but the first line of support for RHEL will be the Red Hat support team.

Security and Compliance

Microsoft Azure includes several security features and offerings that integrate with your RHEL workloads. Some of these solutions we will discuss in more detail in Chapter 6, but here's a quick overview:

- Trusted Launch (Secure Boot) uses Trusted Platform Modules (TPMs) during the preboot sequence to verify the authenticity of an operating system before launching. Trusted Launch is supported for both Marketplace and custom images.
- Azure Disk Encryption enables both the operating system and data disks to be encrypted at rest.
- Defender for Cloud continuously scans and assesses the security posture of resources running in Microsoft Azure.
- Defender for Endpoint for Linux provides antivirus and anti-malware protection of Linux operating systems, including RHEL.
- Just-in-time VM Access provides on-demand access to Azure VMs and can use Entra ID as a certificate authority for SSH access.

Beyond these Azure-specific services, Microsoft has also published guidance on running Red Hat Identity Management and Red Hat Satellite.

Red Hat Identity Management provides a centralized way to administer and manage identities, authentication, policy, and authorization for Linux operating systems. Identity Manager can run as a standalone solution or integrate with Microsoft Entra ID or Windows Server Active Directory. For most cloud-based deployments, the recommendation is to integrate with Microsoft Entra ID and implement Red Hat SSO.

Red Hat Satellite provides lifecycle management and update support for RHEL servers. Microsoft and Red Hat provide guidance on deploying Satellite in Azure or extending your existing Satellite deployment by setting up a cluster of Satellite Capsule servers.

Azure RHEL Landing Zone Accelerator

In addition to providing guidance around Red Hat Identity Management and Satellite, Microsoft has worked with Red Hat to develop a holistic approach to adopting RHEL for Azure. An Azure landing zone is a program started by Microsoft that describes an architecture that employs the Cloud Adoption Framework. Figure 3-1 shows a subscription within a larger Azure landing zone deployment that is meant to support Red Hat components and Azure VMs running RHEL.

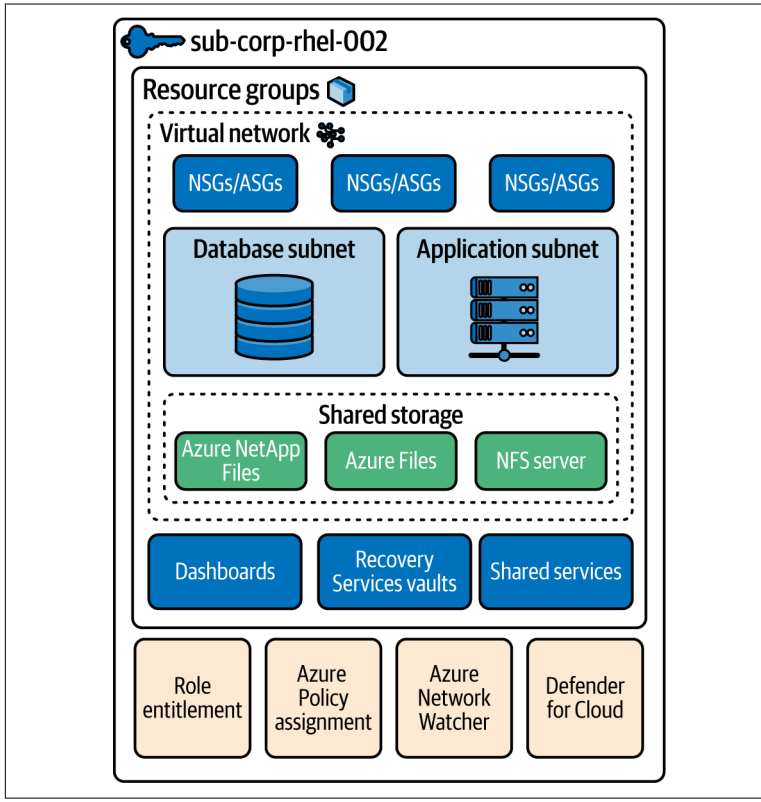

Figure 3-1. Red Hat–focused subscription as part of a larger Azure landing zone deployment

The conceptual architecture behind Azure landing zones is to provide clean separation of duties and permissions between different groups and teams within an organization. Landing zones rely heavily on the use of Azure management groups, subscriptions, and virtual networks to provide logical and physical separation between services like identity, connectivity, and applications.

The RHEL landing zone accelerator carves out sections of the landing zone that are specific to support Red Hat solutions, such as Identity Management, Satellite, and OpenShift. If you are considering a robust implementation of RHEL in your Azure environment, we'd highly recommend reviewing the planning and architecture behind this landing zone.

Expanded Offerings

Beyond running RHEL for Microsoft Azure, Red Hat and Microsoft provide a variety of complementary solutions that can enhance your cloud environment. These expanded offerings are designed to integrate seamlessly with Azure services while leveraging the enterprise-grade stability and security you expect from Red Hat. Whether your goal is to modernize application development, automate infrastructure, or deploy complex middleware, these tools offer a streamlined path to achieving your objectives.

Red Hat OpenShift brings a comprehensive application platform to Azure and helps organizations modernize their applications and infrastructure, build new cloud native applications, and accelerate their digital transformation. Azure Red Hat OpenShift (ARO) is a fully managed, Azure-native application platform for building, deploying, and scaling applications without the operational overhead of maintaining the underlying platform. Jointly operated and supported by Red Hat and Azure, the service integrates tightly with Azure's identity, networking, and security capabilities, making it a strong choice for organizations building with containers and pursuing hybrid or multicloud application strategies.

For infrastructure automation, the Red Hat Ansible Automation Platform enables you to codify IT workflows and standardize configurations across both on-premises and cloud environments. On Azure, Ansible can be used to automate provisioning of VMs, network configurations, and application deployments, reducing manual effort and minimizing the risk of configuration drift.

Finally, Red Hat JBoss Enterprise Application Platform (EAP) offers a robust, Java-based application server for building and running enterprise-grade applications in Azure. With its support for modern Jakarta EE specifications, lightweight architecture, and integration with CI/CD pipelines, JBoss EAP on Azure provides a scalable and secure runtime environment for mission-critical workloads.

Summary

By combining RHEL with these expanded offerings, you can create a cohesive, modernized infrastructure that supports both legacy and cloud native applications. The deep integration between Red Hat solutions and Microsoft Azure ensures consistent performance, enterprise-grade security, and operational efficiency across your environment. Whether you're migrating workloads, building new applications, or automating complex processes, the joint Red Hat–Microsoft ecosystem provides the flexibility and scalability needed to meet your business goals today—and to adapt to whatever comes next.

CHAPTER 4
Ubuntu and Canonical on Azure

Contributed by Dimple Kuriakose, Gauthier Jolly, Ijlal Loutfi, and Jehudi Castro-Sierra of Canonical

Whether you're migrating existing Linux deployments or architecting new cloud native solutions, Canonical, the company behind Ubuntu, has developed a comprehensive suite of tools that leverage Azure's infrastructure while maintaining the familiar Ubuntu experience. This chapter introduces Canonical's Azure-native tools and services, designed not just to "run Ubuntu" in the cloud but to build, scale, and secure production-grade Linux applications throughout the software lifecycle. As shown in Table 4-1, these tools span key areas that cloud architects commonly work with.

Table 4-1. Canonical's offerings from a cloud architect's perspective

Key considerations for architects	Solutions offered by Canonical
Application development platforms	Azure-optimized Ubuntu images
Security considerations	Handling of common vulnerabilities and exposures (CVE), Kernel Livepatch, AppArmor, Secure Boot and Measured Boot, Full Disk Encryption (FDE), confidential computing, Ubuntu Snapshot Service
Compliance requirements	FIPS-compliant images, hardened images based on DISA STIG or CIS benchmarks
Secure containerization options	Chiseled Ubuntu images, Container Build Service, Rocks
Kubernetes-related scale management options	Support for Ubuntu-based AKS worker nodes, Canonical Kubernetes for long-term support, and multicloud options

Key considerations for architects	Solutions offered by Canonical
Application management options	Lifecycle management using Juju and charms, managed apps and solutions (MLOps, data processing clusters, SQL and NoSQL databases, observability solutions, estate management)

Underpinning these technologies is Canonical's commitment to providing enterprise-grade support, ensuring you can confidently adopt and rely on secure open source solutions within your Azure environments. In the rest of this chapter, we will look at each of these offerings in some detail.

Azure-Optimized Ubuntu Images

Ubuntu is the most popular Linux environment for cloud developers, and it is officially endorsed by Microsoft for Azure. Through close collaboration with Microsoft, Ubuntu images are specifically fine-tuned to maximize performance on Azure infrastructure and support the latest cloud features as they are released. The images integrate with core Azure services, such as Azure Pricing, Azure Guest Patching Service (AzGPS), and Update Management Center. Multiple variants of these Azure-optimized images are available:

Server images
: These are general-purpose, customized images based on an Azure-optimized kernel. The customized kernel enables cloud-specific features such as accelerated networking for the InfiniBand-capable instances and support for single root I/O virtualization.

Minimal server images
: These images are designed for automated deployment at scale and have a reduced default package set. Installed documentation (manual pages) and tools intended for interactive usage are omitted. They are much smaller, boot faster, and require fewer security updates over time because they include fewer installed packages.

Ubuntu Pro images
: These are premium images that include optional FIPS-certified components, CIS hardening options, comprehensive open source security coverage for up to 12 years enabled by Expanded Security Maintenance (ESM), Kernel Livepatch service,

estate management service through Landscape, and an optional 24/7 enterprise-grade support. Minimal Ubuntu Pro server images are also available.

Ubuntu Pro Federal Information Processing Standards (FIPS) images
: These are images built on Ubuntu Pro, but with the FIPS-certified modules preenabled so that they are used from the first boot of the image and make it easy for US government Ubuntu users to move their workloads to Azure.

CIS hardened images
: These are prehardened (CIS) Ubuntu Pro minimal images, designed for those who want to implement security best practices for Ubuntu out of the box.

Confidential virtual machine (CVM) images
: These are confidential-compute capable images that support chipsets from both AMD SEV-SNP and Intel TDX.

NVIDIA virtual machine (VM) images
: These are images optimized for use in VMs running on NVIDIA GB200 hardware.

For a complete and updated list of the available images, refer to the "Find Ubuntu Images on Azure" page of the Ubuntu on Azure documentation.

Enhancing Security Through Ubuntu

Ubuntu's security strategy is a multilayered defense system designed to counter specific threats. Each security primitive addresses a unique threat, and understanding them enables you to choose the best defenses for a given environment.

CVE Handling: Defense Against Known Vulnerabilities

At the most basic level, security begins by defending against attackers who exploit known vulnerabilities. This means monitoring newly discovered vulnerabilities (CVEs) and patching them with security updates as soon as they are discovered.

Ubuntu LTS (Long-Term Support) releases come with five years of security updates for the approximately 2,300 packages available in the main repository. With ESM, available as part of Ubuntu Pro, the

coverage is expanded to up to 12 years and includes an additional 34,000 packages from the universe repository.

> **NOTE**
>
> The main repository includes Canonical-supported software, and security updates are accessible to anyone who has installed Ubuntu. The universe repository includes software that is maintained either by the community or by Canonical. To access security updates for it, you need an Ubuntu Pro subscription.

On Azure VMs running Ubuntu, if you check the assessment of Azure Update Manager under "Updates," you'll see that it highlights the security and critical updates that can be patched by enabling Ubuntu Pro.

Kernel Livepatch: Runtime Security Against Critical Vulnerabilities

While timely patching is crucial, applying kernel security updates often requires a system reboot, which can disrupt operations. To address this challenge, Ubuntu Pro includes the Kernel Livepatch service**.** This feature enables the safe patching of high- and critical-level kernel vulnerabilities at runtime, without requiring an immediate reboot. Livepatch helps maintain security and compliance while allowing you to schedule planned maintenance windows, deferring unplanned reboots for extended periods (potentially up to a year).

AppArmor: Protection Against Unauthorized Access

Even with timely patching, applications might have unknown (zero-day) vulnerabilities that could be exploited to gain unauthorized access or cause damage. A crucial security best practice to mitigate such risks is to apply the principle of least privilege, ensuring that applications can only access the specific resources they legitimately need to function.

Ubuntu incorporates AppArmor, a powerful mandatory access control (MAC) system designed to enforce this principle. AppArmor uses security profiles defined for each application, specifying exactly what files, network ports, and capabilities it can access. This effectively contains potential exploits by restricting an application's actions, even if it is compromised.

Ubuntu comes preinstalled with a range of AppArmor profiles for common applications. Furthermore, if your critical workload application doesn't have a profile, you can easily create one. To generate a new profile, AppArmor can be put into Complain mode while the application runs with its full range of capabilities. The actions taken by the application will be captured in a file. You can then use this file as a new profile for Enforce mode, where any deviations from the expected actions will be restricted.

Secure Boot, Measured Boot: Securing Early Boot Software

Another type of attack involves the attacker gaining physical access to your machine and modifying the early boot software—the foundational components responsible for initializing the hardware, verifying components, and loading the OS. Tampering with these allows malware to execute with the highest privileges, even before the OS security mechanisms are loaded.

To avoid this, you can use the following security primitives, both of which are available as selectable options while launching an Ubuntu VM on Azure:

Secure Boot
: This is a UEFI firmware feature that ensures that only trusted components can be used during the boot process. It uses signature databases and cryptographic keys to validate the authenticity and integrity of each component. However, it would still be possible for an attacker to replace a component with an older version that is signed and trusted but includes unfixed vulnerabilities. To avoid this, you will need to use another security primitive called measured boot.

Measured Boot
: This feature verifies the integrity of boot components using a hardware-based tamper-resistant log known as a Trusted Platform Module (TPM). During the boot process, each component being loaded is measured using a cryptographic hash function, and its hash value is stored in the TPM. Later, these values can be compared to a set of expected ones to ensure that no tampering has occurred. Specifically, in the case of Azure confidential VMs (which are explained ahead), a virtual TPM is used instead of a hardware-based TPM.

Full Disk Encryption: Securing Data at Rest

Securing the early boot software against an attacker with physical access is only effective if the hard disk containing your OS and other data is also independently protected. For this, you can use FDE, another security feature that is available as a selectable option during the launch of an Ubuntu VM on Azure. FDE ensures that all data on the disk is inaccessible without an encryption key.

Confidential Computing: Defense Against a Malicious Host

Next, consider an attacker capable of compromising the entire host environment, including its firmware, the host OS, and hypervisor, while you are running a VM on that host. Even if you were to use FDE to protect data at rest on disk, data remains vulnerable when in use—when loaded into RAM or processed by the CPU.

Confidential computing mitigates this problem by creating a hardware-based trusted execution environment (TEE). In this environment, the data stored in RAM is encrypted, and a specialized CPU with an AES encryption engine (such as AMD SEV or Intel TDX) is used to control access to this data.

Any VM deployed on and capable of using such hardware is called a *confidential virtual machine (CVM)*. While the goal is to significantly reduce the trust required in the host environment, current implementations still utilize certain host infrastructure elements, such as Azure's virtual TPMs (vTPMs), especially for persisting the vTPM state.

As part of Azure's Confidential Computing offering, Canonical currently provides Ubuntu CVMs powered by AMD SEV-SNP. For confidential AI workloads, the security model also extends to GPU processing. The NVIDIA H100 Tensor Core GPU implements its own TEE with encrypted data transfer between the CPU and GPU. Offerings based on Intel TDX are currently available in public preview mode, but will be generally available in the near future. Confidential computing is covered in more detail in Chapter 6.

Ubuntu Snapshot Service: Ensuring Consistent Rollout of Security Patches

Finally, consider large estates with hundreds or thousands of machines that require regular security patching. Ubuntu comes with an unattended upgrades feature, which ensures that new security updates are downloaded and installed every day. This works well for individual machines, but might not be a great idea for a whole fleet of machines, especially where workload disruption and service uptimes are important concerns. In this case, you might want to roll out the patches gradually after testing their effect on some machines. But since the Ubuntu archives are constantly updated with the latest patches, this approach might lead to the possibility of inconsistent patches across machines.

To solve this problem, you can use the Ubuntu Snapshot Service, which makes it possible to see and use the Ubuntu archive as it was at any specified date and time. This means you can select a specific snapshot of the archive and use it for rolling out updates across all your machines without worrying about the time lag between machines. The snapshot service has also been integrated with AzGPS to provide a comprehensive patching solution that includes health monitoring and pausing of rollout.

Ensuring Compliance Through Ubuntu

Customized Ubuntu images are available to help you with your compliance requirements, such as FIPS compliance and hardening based on DISA STIG and CIS benchmarks.

FIPS Compliance: Strengthening Cryptographic Integrity

FIPS 140 is a US and Canadian government standard that requires cryptographic modules to meet specific security requirements. It helps ensure that the cryptographic algorithms used are flawless and properly implemented.

FIPS-compliant images are handy for US federal agencies and anyone deploying systems and cloud services for federal government agency use, whether directly or through contractors. FIPS has also been adopted outside of the public sector in industries where data

security is heavily regulated, such as financial services (PCI-DSS) and healthcare (HIPAA).

Ubuntu Pro FIPS images are FIPS compliant. Ubuntu Pro provides access to FIPS-certified modules. You need to enable FIPS to get a FIPS-compliant image. These images ensure compliance by using FIPS-certified modules that handle all cryptographic operations, from hashing to encryption. This includes modules such as the Linux Kernel Crypto API, libgcrypt, strongSwan, OpenSSL, and OpenSSH.

DISA STIG and CIS Benchmarks: Hardening Your System

Inadequate system configurations, such as open or poorly secured settings, can create vulnerabilities that lead to security breaches. To prevent such attacks, you need to harden your system through robust and secure configurations.

Ubuntu is designed to be secure by default. However, depending on the use case, enabling all possible hardening options may not make sense. Ubuntu LTS releases are configured to strike a balance between usability, performance, and security. However, security best practices recommend hardening systems beyond the defaults to reduce the attack surface. This includes securing network services, such as SSH, removing unnecessary software packages, configuring strict file permissions, disabling unused hardware ports, and ensuring robust logging.

Comprehensive hardening guidance is available through established standards like the CIS Benchmarks (from the Center for Internet Security) and DISA STIGs (from the Defense Information Systems Agency). Based on these standards, Canonical produces tailored hardening profiles and tools for Ubuntu. Adopting configurations based on these benchmarks, especially the CIS benchmarks, is key to meeting the Azure Linux Security Baseline.

Applying these benchmarks manually can be time-consuming, as they often contain hundreds of detailed configuration steps. Canonical provides the Ubuntu Security Guide (USG) to simplify and automate this process. USG utilizes profiles derived from CIS benchmarks (Levels 1 and 2) and DISA STIGs to generate remediation scripts, allowing you to harden systems consistently and efficiently. It can also produce audit reports detailing compliance.

Canonical also provides prehardened Ubuntu images in the Azure Marketplace with these security configurations built in, making it easier to deploy secure systems right from the start.

Secure and Optimal Containerization Options

Containerized applications built using open source software are becoming increasingly widespread. But the way they are built and their reliance on external components expose them to considerable security risks and make them hard to maintain.

They face multiple issues:

- Open source from a wide range of sources is often included in the containers, making it hard to keep the components up to date, thereby leading to the risk of unpatched vulnerabilities.
- Obtaining containers from many different sources makes it difficult to verify the quality and trustworthiness (provenance) of the components used. This is needed to avoid supply chain threats.
- Inconsistencies across container development, testing, and production environments can negatively impact the developer experience and lead to further security vulnerabilities.
- Finally, due to the use of generic base images and package managers that suggest extra recommendations, it is easy to add more than what's needed. This leads to bigger containers, which means a broader attack surface, a lack of transparency, and more difficulty in obtaining regulatory compliance.

One approach to reduce the size of containers is to make them smaller by reducing them to their absolute essentials, by making them "distroless." This can be done by removing all the extra libraries and utilities that come along with a Linux distribution and ensuring that only the application, along with its runtime dependencies, are left intact. However, this process is quite error prone, and you might need specialized tooling and deep distro knowledge to create the expected container image. Moreover, the further away you get from a distro, the harder it is to maintain, update, harden, and audit the created container image. For instance, vulnerability scanners look at the metadata of a container image to determine the packages present and, in turn, list the vulnerabilities present. If

the metadata file is absent, the scanners will wrongly report zero vulnerabilities even when vulnerabilities actually exist.

So while containers are relatively easy to create, it is hard to make them minimal, secure, easily maintainable, and well organized. To solve this problem, Canonical has introduced the concept of chiseled Ubuntu images.

Chiseled Ubuntu Images

Much like distroless containers, chiseled Ubuntu container images only include the necessary components for an application to function, but unlike other distroless approaches, they are regularly updated and supported.

Chiseled Ubuntu images are built with the Chisel tool. This developer-friendly package manager transforms a full Ubuntu distribution, used to build the application, into a purpose-built, application-specific container. It is only used at build time and not shipped in the final image.

Chisel uses upstream Ubuntu package information and overlays it with additional data to extract a precise slice of the OS that is needed to run the containerized workload. The additional data is about package slices, which are sets of files required for specific runtime use cases. As shown in Figure 4-1, package slices (the numbered boxes) are subsets of Debian packages, with their own content and set of dependencies to other slices. Appropriate slices from each Debian (deb) package are chosen to create the final chiseled image.

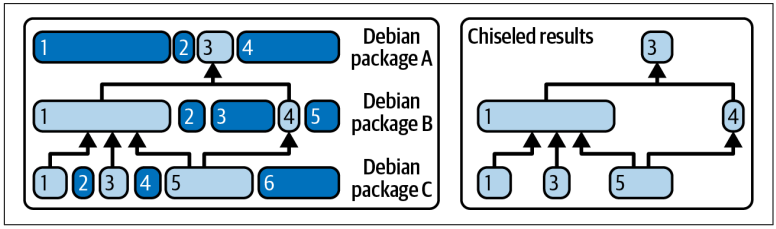

Figure 4-1. Creating chiseled images from Debian packages

The images built are ultrasmall (e.g., .NET chiseled images are over 100 MB smaller than those built using a standard Ubuntu image), and they are created using high-quality components from the Ubuntu OS. Canonical acts as the trusted vendor, and the same

consistent Ubuntu environment is used across development, testing, and production.

All Ubuntu-based container images, including those built by end users from Ubuntu LTS content, are fully supported by Canonical, on the same terms as classic Ubuntu images:

- Five-year free bug fixing and security patching for the main libraries
- Up to 12-year security patching for Ubuntu Pro customers, on all Ubuntu packages
- Optional 24/7 phone and ticket support

To simplify the process of creating these container images for you, Canonical also provides a service called the Container Build Service.

Container Build Service

Canonical's Container Build Service can be used to build optimal containers for your entire open source dependency tree, including open source that is not already packaged as a deb in Ubuntu. Based on your requirements specification, the Container Build Service will design a container image for your application and will produce a chiseled image. This image can then be used on any Open Container Initiative (OCI)–compliant container platform, such as Ubuntu, RHEL, VMware Kubernetes, or the different public cloud Kubernetes offerings such as AKS, Amazon Elastic Kubernetes Service (EKS), Google Kubernetes Engine (GKE), and Oracle Kubernetes Engine (OKE). This custom-built image will be security maintained for CVEs for up to 12 years and supported by Canonical on all those platforms through Ubuntu Pro subscriptions.

Finally, another offering from Canonical in the space of secure containers is a container variant called a rock.

Rocks

A *rock* is an OCI-compliant, Ubuntu LTS–based container image with a well-defined and opinionated design that meets the security and stability requirements of cloud native software.

The OCI specifications dictate what a container should look like and how it should be handled, ensuring portability of the container

across different containerization tools such as Docker and all major flavors of Kubernetes.

Being OCI compliant means that rocks are portable across tools, and Ubuntu LTS–based means they are based on an LTS version of Ubuntu.

A well-defined and opinionated design:

- Has a predictable interface, allowing easy management of your application
- Includes built-in metadata, allowing your application to easily infer information about its environment
- Uses a declarative syntax (YAML file instead of scripts), ensuring an easy experience for the container builder

Finally, since the rocks are based on LTS versions of Ubuntu, they are secure and stable.

Getting a rock

You can use the `docker pull` command to get access to existing rocks:

```
$ docker pull ubuntu/prometheus:2.46.0-22.04_stable
$ docker pull ubuntu/mlflow:2.1.1_1.0-22.04
$ docker pull ubuntu/alertmanager:0.25.0-22.04_stable
$ docker pull ubuntu/grafana-agent:0.35.2-22.04_stable
...
```

Building a rock: Rockcraft

In theory, a rock can be built using a Dockerfile and Docker Build, but that would be a highly cumbersome process. Crafting rocks using a tool called Rockcraft is much simpler.

Built and tailored for rocks by Canonical, Rockcraft abstracts away the repetitive boilerplate steps needed to build an image. It uses a simple declarative language and works with a *rockcraft.yaml* file. This is akin to Docker Build working with a Dockerfile.

Rockcraft can also be combined with Chisel to create minimal container images (as explained in the previous section). This ensures that the rocks remain small and specific, with minimal exposure to vulnerabilities.

If you want to try your hand at building a rock, you can use the demo code and instructions from the following pebble-to-rocks workshops:

1. Install Rockcraft.
2. Learn how to build a rock.
3. Inspect and run a rock.
4. Create a chiseled rock.

You can browse through some of the existing rocks on GitHub.

Scale Management Options Using Kubernetes

With a large number of containers, container orchestration and, in turn, Kubernetes becomes essential. On Azure, the primary way to leverage Kubernetes is through the managed AKS.

Support for Ubuntu on AKS Worker Nodes

Ubuntu is fully supported and widely used as the underlying operating system on AKS worker nodes. Using Ubuntu worker nodes ensures consistency with development environments and other deployments that run Ubuntu. It also helps in leveraging Canonical's security maintenance and optimizations for Azure. You can find details on using Ubuntu with AKS in the official documentation.

Canonical Kubernetes for Long-Term Support and Multicloud Options

If you need a consistent Kubernetes distribution across multiple clouds (Azure, other public clouds, private clouds, edge), or if you need longer-term support than standard Kubernetes releases offer, then you could consider Canonical Kubernetes. It is a conformant, hardened Kubernetes distribution that runs within standard Azure VMs and is available directly from Canonical.

It is a full implementation of upstream Kubernetes, delivered as a self-contained and secure snap package. It addresses complexities by including essential components like a container runtime, networking, and storage interfaces, often with sensible defaults. While developers can use its latest version to track the latest upstream

Kubernetes innovations, organizations prioritizing stability can utilize the LTS releases of Canonical Kubernetes. These LTS versions are backed by Canonical's long-term security commitment of up to 12 years with an Ubuntu Pro subscription.

Canonical Kubernetes can be deployed across multiple clouds to provide a unified, production-grade Kubernetes experience for developer workstations, cloud, datacenters, and edge deployments. Since it comes in the form of a snap, it is easy to install, maintain, and upgrade. It runs in complete isolation with minimal access to the underlying system's resources and also includes automatic patch upgrades that protect against known vulnerabilities.

Application Management Options

Most complex solutions are not composed of a single component or service. They need supporting components like databases, caches, event queues, and essential services like observability, patching, identity management, and so on. Configuring each of these to work well with each other is hard enough on a single cloud, and much worse in hybrid environments. Moreover, once you have set it all up, what happens when each component evolves with new versions and patch releases? What if you need to scale or migrate your solution?

These are cumbersome problems, and to solve them, you will need expert domain knowledge about operating each component and a means to ensure that the components work well together (both initially and in the long run). Canonical has created a simple cloud-agnostic paradigm to meet these lifecycle management needs.

Lifecycle Management Using Juju and Charms

Canonical supports lifecycle management through operators called charms, which define operational code for any given software, and through an orchestration engine called Juju, which ensures the operators work well together. For example, deploying a scalable web application typically involves coordinating a web server, database, and cache; Juju, using charms, can automate the complex setup, configuration, and linking across different cloud VMs or Kubernetes clusters, significantly simplifying operations.

Charms
: A charmed version of a software is essentially a wrapper that distills the software's relevant operational expertise into clean, maintainable Python code. The same charm can be used across different backing clouds, leading to complete portability. Canonical has created open source charms for many popular products such as Kubeflow, MySQL, PostgreSQL, OpenSearch, and Apache Kafka. For a complete list of the available charms, you can visit Charmhub, where all the charms are published along with their relevant documentation. You can also create your own charms using Charm SDK, a toolkit for building charms.

Juju
: Juju is the orchestration engine that enables the deployment, integration, and lifecycle management of charms—at any scale, on any platform. For instance, during deployment, Juju can build VMs, provision bare metal machines, deploy containers on Kubernetes, or deploy VMs on public or private clouds. Once deployed, it gives a rich language for defining integrations between charms (irrespective of the underlying platform on which a charm is deployed). Finally, it also includes actions for defining common operations such as scheduling backups, upgrading software, updating signatures, adding users, and so on.

If you prefer a more hands-off approach while taking advantage of these tools, Canonical also offers an option for managed services.

Managed Apps and Solutions

Many enterprises want to transition to open source but do not have sufficient resources or skills to do this with the required level of operational excellence. This is where Canonical's portfolio of Managed Apps and Services helps. They allow you to focus on your core applications while handling the setup and operation of all the underlying software and infrastructure.

The Managed Apps portfolio currently includes support for machine learning operations (MLOps), data processing clusters, SQL databases, and NoSQL databases. You can set them up on any public, private, or hybrid cloud infrastructure, including Azure. For instance, the managed MLOps solution is available on the Azure Marketplace as a one-click deployment option.

The solutions are natively compatible for use with a multicloud architecture, and tools are also available for observability, monitoring, and overall estate management.

Summary

Canonical and Microsoft maintain a close partnership focused on providing secure and reliable open source solutions on Microsoft Azure. Ubuntu is the foundation for many of these offerings, bringing established security practices and stability developed over many years.

As explored in this chapter, this collaboration provides capabilities across multiple areas that are particularly relevant for cloud architects—Azure-optimized VM images, security and compliance features of Ubuntu, secure containerization options, Kubernetes-related scale management options, and generic application management options.

CHAPTER 5

SUSE Linux on Azure

Contributed by Peter Schinagl and Derek Reinhardt of SUSE

Running SUSE Linux Enterprise Server (SLES) on Azure effectively requires Azure-specific optimization. This chapter provides technical details on enhancing performance, security, and management for your SUSE workloads, offering practical steps for deployment and configuration. With over 30 years in enterprise Linux development and a partnership with Microsoft, SUSE provides solutions grounded in a commitment to open source principles. SUSE actively fosters collaboration and community innovation, ensuring choice and preventing vendor lock-in. Ultimately, SUSE provides comprehensive, enterprise-ready technologies tailored for modern cloud deployments on platforms like Azure.

Features of SUSE Linux Enterprise Server on Azure

SLES on Azure provides a stable, secure, and optimized foundation for your applications and services, backed by robust support.

Performance

SLES is engineered to integrate with Azure to maximize the speed and efficiency of your applications and websites:

Azure-optimized kernel
: This option offers faster boot times, improved runtime performance, and advanced device support, configured specifically for the Azure environment.

Hyper-V integration
: Preinstalled drivers eliminate manual configuration, ensuring smooth operation with Azure's hypervisor.

I/O performance
: This utilizes optimized Linux I/O-Scheduler and Block-Multi-Queue settings, maximizing data handling efficiency within Azure's infrastructure.

Network management
: Built-in Azure Network Adapter drivers and cloud-init simplify network connectivity and system management.

Advanced virtualization
: This leverages Gen2 VMs for enhanced performance and support for large instances, including the M-family, for demanding workloads.

Security

Security is primary, with a suite of certifications, defenses, and regular updates to protect your data from evolving threats. Regularly patched images and a dedicated regional cloud update infrastructure deliver timely security updates, minimizing vulnerabilities.

Prehardened images
: Prehardened images are available to strengthen security and are designed to comply with public cloud frameworks.

Automated security auditing
: SLES's OpenSCAP profiles provide automated security auditing within the distribution (package: scap_security). These profiles are also available in the upstream ComplianceAsCode GitHub project.

Certifications
: SLES's security is validated by certifications like these:

- Common Criteria EAL 4+
- FIPS 140-3
- PCI DSS
- STIG/DISA
- SOC2
- SLSA L4
- ISO-IEC 27001/27701

For a complete list of certifications, refer to the SUSE documentation.

Management

Managing your SLES server on Azure gives you the ability to maintain control, streamline operations, and automate tasks:

Secure remote access
: Secure Shell (SSH) is enabled at boot for immediate and secure server access.

Precise time synchronization
: Preconfigured time synchronization with chrony ensures system consistency.

Customizable deployments
: Cloud-init enables customized server configurations, simplifying deployment and automation.

Flexible storage management
: Simplified disk resizing helps to adapt to evolving storage needs.

Azure tool integration
: Seamless integration is available with Azure CLI and Power Shell for streamlined administration.

Automated updates (PAYG)
: Automatic register to update channels for PAYG images ensures systems stay current.

Cost-effective solutions
: SLES offers flexible PAYG and BYOS options. Table 5-1 provides a comparison of the PAYG and BYOS models.

Table 5-1. SLES payment models on Azure

Pay as you go (PAYG)	Bring your own subscription (BYOS)
Operational expenditure (OPEX) model.	Capital expenditure (CAPEX) model.
You pay Microsoft for time-based use of resources, including the OS and infrastructure.	You pay Microsoft for infrastructure only. You buy the OS subscription from SUSE.
No direct contract with SUSE.	You are both a Microsoft and a SUSE customer.
Automatic registration within Azure Metadata and billing engine and Cloud Update infrastructure.	You register the OS subscription within the SUSE Customer Center.
Simplified management with Microsoft as the single point of contact.	Direct relationships with both Microsoft and SUSE.
Deployment is straightforward.	Migrate existing SUSE subscriptions to Azure.
Pay only for instance running time.	Predictable costs with upfront payment options.
Microsoft provides L1, L2, and L3 support. SUSE provides L3 to Microsoft.	Direct support from SUSE (L1, L2, L3) for the OS.
Automatic repository configuration.	Manual registration required.
Updates via Azure regional infrastructure.	Updates via the SUSE Customer Center.
OS usage compliant due to the PAYG model.	Usage monitoring required for license compliance.
No add-ons possible.	Add-ons like Live Patching, LCM, HA, and LTSS available.
SLES for SAP: Live Patching and SUSE Multi-Linux Manager Lifecycle Management included.	SLES for SAP: Live Patching and SUSE Multi-Linux Manager Lifecycle Management is a separate purchase.
Microsoft Azure Consumption Commitment (MACC) eligible.	Not eligible for MACC.

SUSE Linux Images Optimized for Azure

SUSE offers a variety of preconfigured solutions (both PAYG and BYOS) for different workloads:

- SLES is general purpose for a wide range of workloads.
- SLES Basic is free, without support.
- SUSE Linux Enterprise Server for SAP Applications (SLES for SAP) is optimized to run SAP applications, simplified with additional tooling and services. The PAYG version has the Live Patching Add-on, and a SUSE Multi-Linux Manager client subscription is included.
- SLES Hardened is a prehardened image version of SLES and SLES for SAP.

- SLES HPC can run high-performance computing (HPC) workloads.
- SUSE Multi-Linux Manager updates, secures, and monitors all Linux (not only SUSE) at scale.
- SUSE Linux Micro is a lightweight and secure OS platform purpose built for containerized and virtualized workloads with an immutable file system.

There are several options for deploying and managing SUSE images. You can easily find and deploy all SUSE products and images directly from the Azure Marketplace. SUSE also provides an ARM template example to automate the deployment of SLES VMs on Azure. In addition to ARM Templates (or Bicep), you can use Terraform or Ansible. Both are directly included within the Azure Cloud Shell, so no installation is required.

Finding and Using Images

To get started, simply use the Azure Marketplace and search for SUSE. This will bring up all our products and services SUSE offers through Azure.

If you deploy your instance with the help of the command line or automation, you need to know the image name. You can look this up with the help of the Azure "az" tool or through the SUSE Public Cloud Information Tracker (PINT), which provides not only the image URN but also more information like a changelog and the image lifecycle details.

The URNs can be used with the Azure CLI to deploy VMs quickly. When searching on PINT for 15 SP6, there are several images returned.

Names for SUSE's public cloud images consist of multiple parts that contain information about the product, its version, a time stamp indicating the release date of the image, and more.

The general naming scheme for SUSE's public cloud images is shown in Figure 5-1. The naming consists of four key identifiers that tell you the details of the image:

- Product (e.g., suse:sles-15-sp6, sles-sap-15-sp6, manager-server-5-0, etc.)

- Flavor (e.g., BYOS, hardened, basic, no flavor, or if it's basic, it means it's PAYG)
- Architecture (e.g., arm64, added if the architecture is ARM64; omitted for x86_64)
- Generation (e.g., Gen1, Gen2)
- Version (e.g., upload date of the image in the format YYYY.MM.DD)

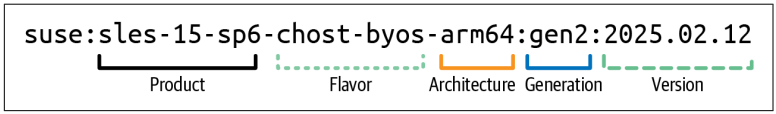

Figure 5-1. The naming scheme for SUSE's public cloud images

The following example command will deploy the PAYG version from January 29, 2025, with Azure Gen2 support for the product SLES 15 SP6:

```
az vm create -n MyVm -g MyResourceGroup \
--image suse:sles-15-sp6:gen2:2025.01.29
```

If deploying a BYOS instance, then a subscription key must be applied to the VM. This is accomplished by logging in to the VM and running the following command:

```
SUSEConnect -r <ActivationCode> -e <EmailAddress>
```

PINT additionally provides the IP addresses of our update server infrastructure for a possible whitelisting in your network to access the update infrastructure in Azure for PAYG images.

When dealing with cloud images, PINT provides changelogs to help track changes to the images themselves. It also provides details on the image changes, CVE fixes, package version changes, package changelogs, and SBOM (software bill of materials) in two formats (SPDX and CycloneDX).

A nice way to get familiar with SLES on Azure is the Microsoft Learn module "Customize a SUSE Linux Enterprise Server Virtual Machine on Azure". It will provide you with a quick overview of how to create an Azure VM and tweak the installation by installing additional packages with the help of our package management tools YaST or zypper.

Updates and Upgrades

SUSE is committed to providing regular updates to products over their standard lifecycle and beyond. The supported time frame depends on the product and can be viewed on the product lifecycle page.

SUSE Linux Enterprise, for example, has a new service pack (SP) released approximately once a year. After a major version of SLES has reached the normal end of life, it is possible to purchase long-term support to extend its usability for several more years. You can perform an online update to the latest service pack with YaST, a GUI system management tool, or through the command line with zypper.

Figure 5-2 provides an overview of supported upgrade paths for SLES 15 SP6.

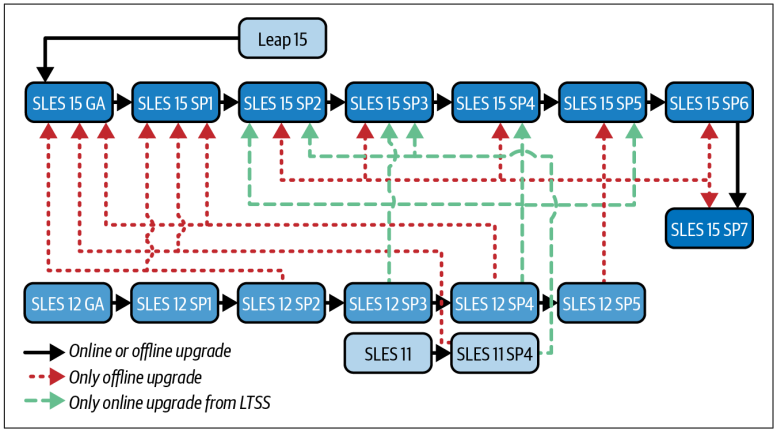

Figure 5-2. SLES 15 SP6 upgrade paths

Before doing anything, you will want to familiarize yourself with the supported upgrade paths. Consult the upgrade guide for the SUSE Linux Enterprise version you want to upgrade to and plan for some downtime, as an online update could require a reboot.

It is advisable to perform a preparation phase, where the instance is readied to be upgraded, and then a migration phase, where the instance is actually migrated.

The first step is preparing the instance, ensuring the instance is fully up to date with the latest updates. To do this, run:

```
sudo zypper patch
```

Once this is complete, the instance is ready for the migration phase. To initiate the migration of the instance, run:

```
sudo zypper migration
```

Follow the prompts to select your migration target. Once the upgrade completes, reboot the instance.

You can continue to run these two commands if you need to upgrade the instance further. For example, if the instance was just upgraded from SLES 15 SP3 to SLES 15 SP4, you can rerun these two commands to upgrade the instance from SLES 15 SP5 to SLES 15 SP6.

In the case of a major version update, such as from 12 to 15, a new installation is often better, as it ensures you benefit from all the latest features of such a major release. Because a new installation is not always possible, SUSE also allows migrations to new versions.

Such migrations typically require an offline installation, which is not possible in public clouds.

For such scenarios, SUSE provides a tool called the Distribution Migration System (DMS). Please follow the documentation carefully.

Overall, DMS involves a few more steps than the online upgrade, but it follows a similar process: prepare the system, then migrate the system. The steps of this process are:

1. Upgrade prerequisites.
2. Upgrade prechecks.
3. Install DMS.
4. Run the migration.
5. Check migration logs.

Specialized Solutions and Workloads

SUSE provides several specialized solutions, including:

SUSE Linux Enterprise HPC
: Extends SLES with supported HPC packages for workload management, cluster deployment, profiling, and libraries for parallel computing. It is available for x86_64 in the Azure Marketplace.

SLES for SAP
: Bundles SLES, the SLE HA-Extension, a SUSE SAP Priority Support contract, an extended maintenance period, an installation wizard for SLES and SAP NetWeaver HA Agents to automate HANA System Replication, predefined installation patterns, and tuning tools for SAP.

  SUSE and Microsoft collaborate to provide detailed SAP installation documentation, based on SUSE best practices and tested by Microsoft.

SLES Hardened
: SUSE offers prehardened images of SLES and SLES for SAP applications on Azure, aligned with CIS and DISA STIG benchmarks, for environments requiring enhanced security configurations out of the box. These images supplement the standard SLES images, which prioritize flexibility.

Certain rules can only be applied after instance creation, for example:

Rules that require having passwords set up
: Passwords would have to be public if configured during the image build. This would defeat the purpose of a secret password.

Rules that affect the network configuration
: Networking is set up during instance creation; therefore, it is not possible to limit access during image build.

Rules for custom partitioning
: SUSE's public cloud images are partitioned to meet the requirements of the framework in which they are released. If your system needs to meet standards that require separate filesystems for given directories, we recommend building your own images and using LVM or moving those directories onto attached disks to get the strictest data separation possible.

Rules to remove packages
: SUSE's public cloud images cater to a wide range of use cases. Even if the number of packages is limited, it is impossible to determine what packages an instance requires.

For a complete list of rules that have been applied during preharden-ing, refer to pcs-hardening.profile. This profile is a combination of the STIG and CIS profiles minus rules that can only be applied after instance creation.

Images of SLES for SAP are hardened using a modified version of the profile called pcs-hardening-sap.profile. Users may need to make additional modifications to the system configuration depending on individual application needs.

Building "Golden Images"

Organizations commonly use custom images to decrease configuration after deployment. This process is referred to as creating a *golden image*, or an image that can be used to deploy preconfigured instances quickly. The process of creating such a golden image requires deploying a base image, making the configuration updates, and then using the tools built into Azure to modify the disk image so that it can be used for new deployments.

There is an inherent challenge in using a running image as the base for a new golden image. During the very initial startup of a Linux system (e.g., the original base image), service identifiers are created on the disk, giving it an "identity" to help prevent device collisions, which conflicts with the goals of a golden image. Such identifiers need to be removed before the disk image can be used for new deployments.

You can use any SUSE base image required, whether the PAYG, the BYOS, or even the hardened image, as your starting place. After making the desired configuration changes, the following steps will remove the identifiers in the installation.

Ensure that the RPM package clone-master-clean-up is installed and simply run it as the last step before shutting down—this will remove the "identity" from the OS:

```
/usr/sbin/clone-master-clean-up
/usr/sbin/waagent -force -deprovision+user && \
export HISTSIZE=0 && sync
```

As the next step, you need to do something similar on the Azure side to create a generalized image. From the Azure command line or cloud console, run the following commands, which prepare the images as a new deployable Azure image:

```
az vm deallocate -g ${RG} -n ${SRC_VM_NAME}
az vm generalize -g ${RG} -n ${SRC_VM_NAME}
```

Now you have built your own "golden image," which you can use for your deployments. The benefit is that it inherits all the metadata from the chosen base PAYG or BYOS image.

For the case that you have started with openSUSE Leap instead of SLES, this is not a problem, as you can directly migrate it to SLES, since they share the same codebase. Take a look at the SUSE blog post "From openSUSE Leap to SUSE Linux Enterprise Server PAYG on Azure" for details.

Managing SUSE Systems at Scale

SUSE Multi-Linux Manager (formerly SUSE Manager or SUMA), available as BYOS and as a PAYG-like offering on Azure, provides a single console to manage, patch, and monitor your Linux estate across hybrid environments. Supporting 16+ Linux distributions, SUSE Multi-Linux Manager centralizes control, saving time and resources:

Centralized control
: Manage diverse environments from a single platform, whether you have a few servers or 10,000+, and whether they run SLES, RHEL, CentOS, Ubuntu, or other popular distributions.

Unified visibility
: Provides a clear view of physical, virtual, and cloud systems for monitoring and control

Content lifecycle management (CLM)
: Manages the lifecycle of software content, allowing you to test updates in staging environments before deploying them to production

Automated updates at scale
: Ensures security and saves time with automated patching across all systems, regardless of size or distribution

Maintain compliance across distributions
: Meet industry security standards with automated compliance checks and reporting, reducing the risk of noncompliance penalties, even as your infrastructure expands and diversifies

Simplified software deployment
: Streamlines software management, ensuring consistency and minimizing compatibility issues, regardless of scale or the mix of Linux distributions

Proactive monitoring
: Receive real-time alerts on system health, preventing potential issues across all distributions

Rapid deployment, no matter the number
: Automate the deployment of new systems and services, reducing manual effort and accelerating time to value

Containerized workload management made easy
: Deploy, manage, and scale containerized applications using modern development practices, regardless of the size of your container environment

Increased uptime, always
: Supports business continuity with high availability and disaster recovery capabilities, minimizing downtime and data loss across your entire infrastructure

Flexible integration for any environment
: Integrate SUSE Multi-Linux Manager with your existing IT tools and customize it to meet your specific needs and workflows

One of the newer features is the included "liberate" formula, which makes it possible to migrate systems from other enterprise Linux clients such as AlmaLinux, CentOS 7, Oracle Linux 9, Rocky Linux 9, or even RHEL 9 to SUSE Multi-Linux Support. With this formula, the conversion will be simple and will take place during the client onboarding on SUSE Multi-Linux Manager.

Azure users additionally can use SUSE Multi-Linux Manager on a monthly billing model (which is a kind of PAYG), paying only for the time of use and the number of managed and monitored systems.

This kind of PAYG offering in Azure needs to communicate with the Azure Billing API, making it a Managed Application and not a traditional VM. It has some core differences from a standard VM; mainly, it always gets deployed in its own virtual network, and it has two resource groups. More details can be found in the SUSE Multi-Linux Manager documentation.

SUSE Linux for Containers and Microservices

SUSE Linux Micro is a lightweight, low-maintenance, and secure OS platform purpose built for containerized and virtualized workloads anywhere. It's available in the Azure Marketplace.

SUSE Linux Enterprise Base Container Images (SLE BCI) offer secure, flexible container images and development tools. These container images, built on SLES, ensure strong security and compliance for containerized workloads and integrate with any Kubernetes distribution, including AKS.

The official SUSE registry provides tested, updated, and certified SLE BCI, ensuring regular updates and fixes in minimal, security-focused base images that simplify custom application development.

Container deployment typically involves:

1. Fetching a base image from a registry
2. Building a custom image using a Dockerfile
3. Deploying containers from the custom image

There are several versions of the BCI to fit your specific requirements:

- BCI-Base is a standard image including core packages and the zypper package manager.
- BCI-Init adds systemd to BCI-Base.
- BCI-Minimal does not include zypper but does include the RPM package tool.
- BCI-Micro does not include zypper or RPM.
- BCI-BusyBox uses BusyBox tooling instead of GNU Coreutils.

Beyond these standard images, several more images for specific applications or languages are built on top of BCI-Base, such as Python, Java, Nginx, and PostgreSQL.

Custom images, built on these bases, enable application-specific containerization. Images are stored in registries (Figure 5-3). Dockerfiles define build instructions, and layered image construction allows for modularity. For detailed information, refer to the official documentation.

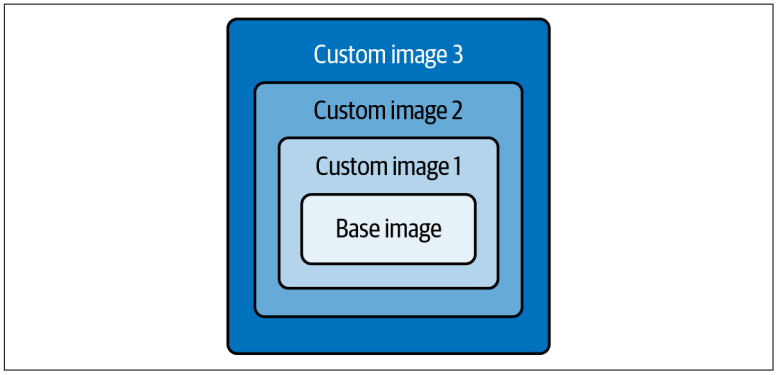

Figure 5-3. Custom images

SUSE Rancher Prime addresses the complexities of containerized workloads. It provides comprehensive Kubernetes support, including Azure AKS lifecycle management, and offers features like multicluster operations, unified security, and robust artifact management (SLSA, SBOM, OCI Prime Registry). SUSE Rancher Prime integrates with SUSE's cloud native portfolio, enabling seamless container workload deployment across datacenters, clouds, and edge environments. Key integrations include SUSE Kubernetes Engine (RKE2/K3s), SUSE Security, SUSE Observability, SUSE Storage, SUSE Virtualization, Elemental, and SUSE Linux Micro.

SUSE Rancher Prime's Application Collection streamlines code-to-production workflows by providing secure, standardized open source application deployment. The collection provides enterprise-grade distribution of minimal, hardened images with signatures and SBOMs, continuous CVE updates, OCI compliance, SLSA Level 3 security, signed containers, a Metadata REST API, multiarchitecture support, minimal image sizes, and continuous vulnerability/antivirus scanning.

SUSE Security (formerly known as NeuVector) is a Kubernetes-native application designed to monitor Kubernetes clusters for vulnerabilities. SUSE Security is able to mitigate problems within the cluster before they reach the underlying operating system or other containers.

SUSE Observability provides comprehensive monitoring, advanced analytics, and integration capabilities.

Support and Resources

Support for SUSE on Azure varies by licensing:

- BYOS: SUSE provides full product support.
- PAYG: Microsoft offers initial support and escalates complex issues to SUSE.

Microsoft provides extensive documentation for running SUSE on Azure. Find SUSE product documentation and best practices on the SUSE website.

The SUSE Customer Center (SCC) manages subscriptions and support. Submit support cases here if you have SUSE subscriptions. Consult the Technical Support Handbook on the SUSE website (under "Support") for initial troubleshooting.

The SUSE Knowledge Base offers solutions and tips.

Summary

Choosing the right enterprise Linux platform for Azure is an important consideration. You need a solution optimized for the cloud that securely supports a wide range of applications. SUSE Linux provides a platform designed to meet these needs, enabling innovation and achieving your cloud goals. SUSE on Azure provides flexible open source solutions you need to build, modernize, and scale applications effectively while controlling costs.

CHAPTER 6

Securing Linux Workloads in Azure

One of the primary concerns organizations have when adopting cloud is security. After all, the cloud introduces unique services and architectures that do not lend themselves to traditional security approaches. Simply re-creating your datacenter architecture and security practices in the cloud is not only inefficient but also counterproductive to achieving your security goals. In this chapter, we will review the various security features and services available in Microsoft Azure through the lens of securing Linux workloads.

Azure: A Code-to-Cloud Security Platform

The applications you host on Microsoft Azure rely on security delivered in multiple layers. While infrastructure security is a critical component of your security strategy, it should not be the only control. Infrastructure security should work in tandem with application security (AppSec) to provide a holistic solution for your organization.

While it is outside the scope of this chapter, we would like to briefly mention the practice of shifting left when it comes to security. This practice involves three key components:

- Involving security teams early on in the development process
- Establishing golden paths and patterns for developers to follow
- Integrating security scanning tools early in the continuous integration and continuous delivery (CI/CD) process

Microsoft Azure integrates well with most DevOps and security tools like GitHub Advanced Security and the Azure DevOps platform to help establish a DevSecOps practice from a tools and process perspective. As the practitioner, it is still incumbent on you to establish these processes and tools in your team and drive adoption in the larger organization.

What you are responsible for securing is a function of the shared responsibility model.

Shared Responsibility Model

Microsoft Azure, like all cloud providers, adopts a shared responsibility model when it comes to the security of applications running on the platform. Essentially, Microsoft is responsible for securing the underlying infrastructure that powers the services offered by Azure (such as AKS, Virtual Machines, etc.) and the security of the services themselves. You are responsible for properly configuring the features and functions of the services you consume, following best practices for your tenant, and securing the code driving your application. Full details can be found on Microsoft's Learn site.

The weight of responsibility for a platform largely depends on the type of platform you are leveraging for your application: infrastructure as a service (IaaS) or platform as a service (PaaS). There is an intrinsic trade-off between responsibility and level of control. IaaS provides the highest level of control and configurability and the highest burden of security on you. It is incumbent on you to make sure that your operating system is properly patched, that your network security groups are correctly configured, and that your application is free of vulnerabilities.

Choosing a more managed experience, such as Azure App Service, removes some options for configuration and control but also shifts the responsibility of security onto Microsoft Azure. While the decision on a platform will be composed of many factors, do not discount the question of security versus control. Microsoft manages more of the technology stack in PaaS services, but you still have significant responsibility for application code, configuration, and user/secret management.

Key Azure Security Services

Regardless of what type of platform you choose to host your application, Microsoft Azure provides a robust set of tools and services to help you secure it. Microsoft's security best practices help you achieve core functions of cybersecurity, including security posture, incident protection, detection and response, and the confidentiality, integrity, and availability (CIA) triad.

The CIA Triad

The CIA triad is a fundamental model in IT security, addressing all security across all aspects of systems (Figure 6-1). This includes sensitive data itself, as well as the systems that process, store, and transmit that data.

Confidentiality
: Preventing unauthorized access to sensitive data

Integrity
: Ensuring and confirming that data remains accurate and unaltered

Availability
: Making sure that data is accessible at all times

More information about this is available at the Microsoft Security 101 GitHub page about the CIA triad.

Figure 6-1. The CIA triad

We will further explore some of the services listed in Table 6-1 in the rest of this chapter, but be aware that even this table is not an exhaustive list of the security tooling in Azure.

Table 6-1. Some of the primary security tooling available within Microsoft Azure

Service	Description
Microsoft Entra ID	Identity and access control for Azure resources
Azure Network Security	Control the flow of network traffic in Azure
Azure Disk Encryption	Protect data on Azure VMs
Azure Storage Encryption	Protect data on Azure Storage accounts
Azure Backup	Protect and recover data
Azure Key Vault	Manage access and lifecycle of secrets
Microsoft Sentinel	AI-powered cloud security information and event management (SIEM) to secure multicloud, multiplatform environments
Microsoft Defender for Cloud	Cloud native application protection platform (CNAPP)
Microsoft Defender for Endpoint	Antimalware and endpoint detection and response (EDR)

You can leverage each of these services to secure your Linux-based workloads in Azure. Let's look at Microsoft's baseline security guidance for Linux workloads in Azure to better understand how.

Baseline Security Guidance for Linux Workloads in Azure

All Azure users have free access to the Microsoft Cloud Security Benchmark (MCSB). This benchmark provides security best practices based on Microsoft's vast experience in all kinds of workloads—not just Linux. The MCSB is the default benchmark used by Microsoft Security tools such as the Microsoft Defender suite of products, but it should be noted that if you have more specific needs regarding compliance (e.g., ISO27001), you can use other benchmarks as well. You can also use the basic version of Defender for Cloud for free on all your workloads. More on this in "Microsoft Defender for Cloud" on page 98.

The MCSB is designed to provide users with best practices and recommendations to maximize the security of the entirety of their cloud workloads, not just their Linux installations. (It should also be noted that the MCSB is not related to OS-level settings.) There are

subsections, or "domains," of the benchmark that describe areas of security interest—network security, identity management, and asset management, just to name a few. The benchmark even has guidance for how to run workloads on AWS. In addition, the benchmark's recommendations include information about how they correlate to industry standards such as CIS, NIST, and PCI. The entire benchmark is a public document and is available on GitHub for download in spreadsheet format.

Let's take a quick look at each section of the MCSB:

Network security
: This domain focuses on securing virtual networks (VNets in Azure parlance). The recommendations describe the best ways to implement network segmentation, connect one environment to another, and utilize advanced protections such as Microsoft Defender for DNS or intrusion prevention systems to protect workloads from external attacks. There are a total of 10 controls in this domain.

Identity management
: This domain focuses on implementing secure identity and access controls. Identity is a foundation of modern network security and does not just mean users—systems, programs, APIs, and so on—all have identities that need to be managed. The nine controls in this domain are intended to ensure that all identities in your network are managed securely.

Privileged access
: Identifying a user/system/API is one thing, but managing their access is another. The privileged access domain addresses what exactly those identities can see and do in your environment. This domain has eight controls covering lifecycle management, user accounts and permissions, the principle of least privilege, and emergency access.

Data protection
: This domain focuses on your organization's data. The eight controls cover data at rest and in transit, through whatever transfer or processing your applications apply. The goal is to minimize data access to prevent unauthorized access or data exfiltration.

Asset management
: This domain provides five controls that ensure security over anything in your Azure environment. This includes guidance on maintaining asset inventories, tagging, asset lifecycle management, and visibility into runtime concerns, such as application control in a VM.

Logging and threat detection
: All systems produce logs. This domain provides guidance on how these logs need to be collected, analyzed, and stored. The seven controls describe how best to log events, access these logs in an investigative manner if needed, and ensure all your systems have consistent logs that can be used collaboratively when a security incident occurs.

Incident response
: Speaking of security incidents, this domain is specifically designed to help you manage what could otherwise be a very stressful and disorganized situation. The seven controls describe the before, during, and after of an incident response in order to help resolve the issue as quickly as possible and provide the organization with details to remediate problems found during the investigation so that the incident doesn't happen again.

Posture and vulnerability management
: This domain focuses on assessing your overall cloud security posture. Effectively, this means looking at every single asset and determining if it is secure. Are there observable vulnerabilities, do apps or systems need patches, and, if so, how do you remediate? The seven controls in this domain take you from start to finish on protecting anything that has any kind of open access to anything else.

Endpoint security
: This domain focuses on endpoint detection and response capabilities. This is a shorter domain—only three controls—but essential to organizational security. Essentially, it asks if you are using an EDR tool that protects against malware. (Hint: you absolutely should be.)

Backup and recovery
: This domain focuses on backup strategy and makes sure that you are implementing it properly. The four controls describe

how to enact regular automated backups, protect that data, and make sure that the backup solution is tested on a regular basis.

DevOps security
: This domain is targeted at securing the development process. There are seven controls in this domain, looking at threat modeling, supply chain security, DevOps infrastructure security, and security of the code itself. All of this is intended to be deployed at the beginning, middle, and end of an application's lifecycle.

Governance and strategy
: By far, the least fun part of IT security is the policy and procedure that directs it. This domain consists of 11 controls covering just about every aspect of how an organization handles IT security from the top down. It touches on all of the other 11 domains and provides guidance on how to approach governance from a comprehensive and authoritative perspective.

Platform Security

Microsoft Azure provides a comprehensive suite of security tools to implement a robust defense-in-depth strategy to protect your cloud workloads. In this section, we will talk about many of them and see how they work together to protect your infrastructure at multiple layers. It is essential to understand these components, and how they work together in a layered fashion, in order to implement the best security strategy for your organization's Azure deployments.

Azure Networking

The MCSB is invaluable in understanding your current security posture, but what tools are used to make that security posture possible? Azure provides multiple tools that operate as layered security to keep your workloads safe. These components work together to provide the most comprehensive security posture possible. We will look at several of the most commonly used security tools in this section, roughly categorized by the kind of traffic they secure. (None of these are Linux-specific, of course; they are essential components used to protect any workload you have running in Azure.)

External security layer

This layer protects the outermost aspects of your Azure environment—that is, the direct contact made from internet users or customers into your applications and resources.

Web application firewall (WAF). This service protects applications from common exploits and vulnerabilities at layer 7. This means that it directly observes packets and interprets their actions, not just where they are coming from (or how many). The WAF service is deployed with public-facing IP addresses, so your infrastructure can be designed so that all traffic has to go through it in order to get to your applications or servers. The WAF can be configured using other Microsoft tools, such as Azure Front Door, Azure Application Gateway, and more, in order to protect traffic all through your IT footprint—not just in the Azure cloud. WAFs run at layer 7 and are primarily used to protect HTTP/S protocols.

Azure DDoS Protection. Distributed denial of service (DDoS) attacks are not about hackers breaking in—rather, they are about bad actors flooding your web servers and app portals so that legitimate traffic cannot get through. (From a security perspective, this would be the availability portion of the CIA triad.) DDoS Protection, as shown in Figure 6-2, prevents this by analyzing traffic at the layer 3 and layer 4 levels and automatically blocking attackers' traffic while allowing legitimate traffic through. This can be linked with the WAF in order to maximize security by utilizing both tools. This makes the service all the more effective, considering the WAF works at layer 7.

DDoS Protection is a paid service. At the time of this writing, there are two tiers of service available. You can pay for DDoS on an individual IP level, or you can pay to protect your entire network (up to one hundred IPs, with a small upcharge for any number of IPs above one hundred). The total number of IPs can be used across multiple subscriptions.

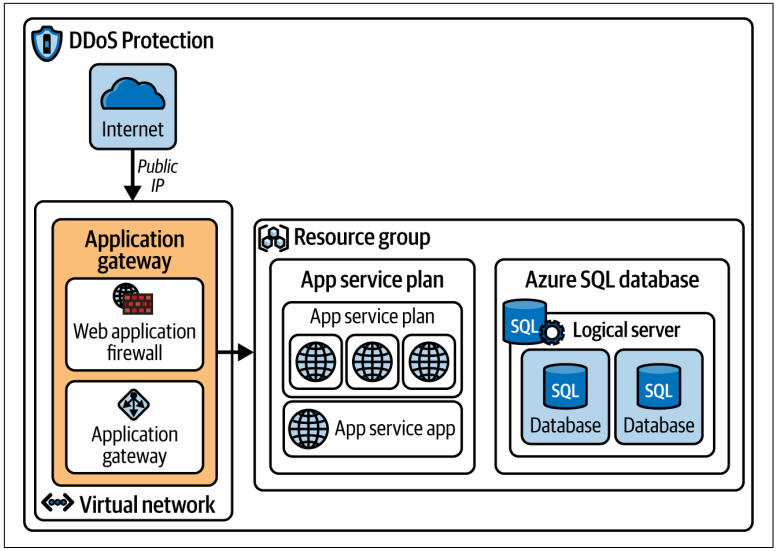

Figure 6-2. DDoS Protection working in Network Protection mode with connections to the web application firewall being used to protect all resources in a resource group

Perimeter security layer

These are combination tools that can protect traffic coming into your network or traversing from one area of your network to another. The main point is that they represent the edge protection for more specific areas and thus can be more precise in their configuration than the fully external tools listed in "External security layer" on page 90. Two of the most common ways of handling the perimeter are the Azure Firewall service, which is baked into Azure and managed by Microsoft, or third-party Network Virtual Appliances (NVAs).

Azure Firewall is a service similar to WAF. This service provides you the ability to control rules to secure inbound and outbound traffic to and from VNets. Azure Firewall is used to protect non-HTTP/S protocols (common ones include RDP, SSH, FTP, etc.) that are used to interact with systems inside your environment by other systems and administrators. The firewall is placed on its own subnet, and all traffic between VNets is routed through it, ensuring it has visibility and control and that it can log all access to and from your resources.

While the Azure Firewall is the standard for this level of security, users can use third-party networking tools if they so choose. These are referred to as NVAs and are essentially black box VMs that run in your environment. Similar to configuring Azure Firewall, you would put these in the routing path and use them to control access. These are often used if customers have a large footprint of other tools and don't want a heterogeneous network security environment.

Network control layer

The tools used to protect individual resources within your VNets are even more discrete in configurability. The most common of these is the Network Security Group (NSG).

In comparison to Azure Firewall, NSGs are much simpler and are used to protect traffic between resources with simple L3/L4 security directives. These are simple allow/deny rules that allow you to specify source and destination addresses, port, and protocol. NSGs are created at a subnet or NIC level, and then the subnet and/or NIC of a VM is associated with them.

It should be noted that these rules can be used as the only security you have protecting a resource from malicious internet traffic. However, this design is against best practices. NSGs are limited in scope and sophistication and should be considered as only one tool in your security toolbox.

Access control layer

Finally, we have the tools to administer and access your environment from an operational standpoint. By this, we mean logging in to the VMs themselves in order to configure or otherwise manage deployed applications, not logging in to the portal to make overall Azure configuration modifications.

It is generally against best practices to permit any kind of direct login to your IaaS resources. This means no direct RDP or SSH via public IP. The Azure Bastion PaaS service, as shown in Figure 6-3, is designed to allow you to access these protocols to administer your resources safely. Bastion is provisioned into a VNet, and you use it to connect to any system in that VNet and other peered VNets. Access is granted via either Azure CLI or directly through the Azure portal. There are a number of different service tiers you can configure, depending on the level of control and features you

want. It should be noted that once Bastion is deployed, it cannot be downgraded—if you want to go to a lower tier of support, it will have to be redeployed.

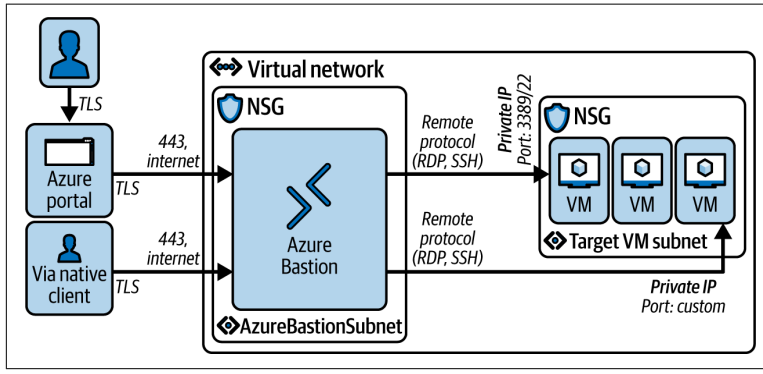

Figure 6-3. An example of the Basic SKU deployment of Azure Bastion and the various connectivity options

Azure Compute

VMs in Azure do not have to rely solely on network security tools for protection. There are a number of options available that can be used to protect operating systems and data from bad actors. Contrary to the more general security tools discussed in "Network control layer" on page 92, these will be more focused on Linux workloads—although it is safe to assume that these protections are also available for Windows workloads.

Azure Disk Encryption (ADE)

Protecting data at rest is just as important as protecting data in motion. This is accomplished using encryption to protect data written to persistent storage. In Azure Linux deployments, the DM-Crypt feature of Linux is used. This encryption is similar to the BitLocker feature of Windows and enables encryption at the operating system level.

There are requirements before Disk Encryption can be used. For example, it is not possible to encrypt disks on Basic, A-series, or v6-series VMs or machines with insufficient RAM (usually 2 GB, but that varies depending on how many volumes are being encrypted). Also, not all Linux distributions (or versions of those distributions) are supported. Check the details of your preferred instance type to

confirm that you will be able to use this feature before you deploy your workloads.

ADE integrates with Azure Key Vault to allow you to control and manage the keys used to encrypt drives. This would allow you to move managed disks around from VM to VM while maintaining the integrity of the encrypted data. Additionally, Microsoft Defender for Cloud will alert you if you have VMs that are not encrypted. This is considered a high severity problem. It is a best practice to encrypt all VMs if possible.

Microsoft Defender for Endpoint

Defender for Endpoint is a next-gen antivirus/antimalware and EDR tool available for most major operating systems. It is an agent-based system running directly on your IaaS workloads to provide security and help you protect against advanced security threats. It is tied directly into the rest of the Defender family of tools and Microsoft Sentinel, providing a landing spot for all logs and giving you a great place to start when it comes to investigating and responding to any threats that might have gotten through your defenses.

Azure Storage

Azure Storage refers to the Azure services that provide access to storage independent of VMs. These can be the blob, file, queue, etc. types of storage in a storage account and can be provisioned and applied however you like. While they can be connected to VMs, they can also exist independently.

Azure Service Side Encryption (SSE)

Any data that exists and is stored in the cloud needs to be encrypted. SSE is used to automatically encrypt any data that is uploaded, and is used to protect your data in most scenarios. The encryption happens on the Azure side; however, if you need to encrypt on the client side, that can be done for blob and queue storage types through different configurations (which are beyond the scope of this discussion). Note that for most scenarios, Microsoft recommends using service-side encryption features for ease of use in protecting your data.

SSE is enabled by default, does not incur an additional cost, and cannot be disabled. This provides a baseline of security for your

data. What you can do, however, is control the keys that are used to apply this encryption.

Microsoft-managed keys. The standard way is for Microsoft to handle the keys. You simply create the storage, and Azure automatically creates and manages the keys. This is the easiest way to do things, but it doesn't give you a lot of control.

Customer-managed keys. Instead of being handled by Microsoft entirely, customers have the ability to manage keys on their own. The customer will provide these from their own key management system, or they can be generated in Azure Key Vault. In either case, they must be hosted in Azure Key Vault.

Customer-provided keys. In the case where the customer needs to maintain the key store, the customer provides the key when making the read/write request. The data at rest is still encrypted; it will simply be inaccessible without access to the customer key. Key Vault can be used in this scenario to store the keys, or the customer can use their own key-management system to store them and provide them when encryption/decryption of data is required.

Azure Boost

Security is woven into Azure's infrastructure itself, with layers of logical isolation in Azure Boost reducing surfaces of attack. Azure Boost is available on a growing number of VM SKUs, and offers increased levels of security by enforcing hardware attestation across Azure Boost hardware, which allows only trusted machines to run production workloads. This cryptographically ensures that hardware and firmware have not been tampered with, which prevents untrusted machines from going into production.

From a security perspective, Azure Attestation enables protection from malicious firmware, preboot malware, bootkits, rootkits, compromised OS, and misconfigurations. Azure Attestation is resistant against replay and event log spoofing attacks while also offering alerting mechanisms for any nodes failing attestation due to any reason. All measurements used by the attestation service are backed by a hardware root of trust.

When a machine passes attestation, it receives a trusted identity certificate that enables it to enter production. Thus, only machines that

have passed attestation and received a trusted identity certificate can enter production, ensuring that only trusted and secure hardware and software are allowed to run production workloads.

Immutable storage

Immutable storage is blob storage that has been configured to be written exactly one time and then never rewritten or deleted. This is known as write once, read many (WORM). The policies around locking data and preventing its deletion can be based on a customer-specific time frame (time-based retention) or on a legal hold. Often (but not always) a legal hold means some kind of investigation is taking place requiring proof that would hold up in a court of law that data hasn't changed. This is not calendar based; the data will remain in a WORM state until the hold is lifted by an administrator.

Immutable storage is very valuable for regulatory compliance reasons, but it's also very helpful for backups. For example, many variants of ransomware have been observed in the wild specifically targeting backups for deletion. Having your backups in an immutable environment guarantees that a bad actor cannot delete the images and snapshots you would use to recover from.

Immutable storage can also be used for version control. You could have a dataset that is identified in this manner that would allow a new file to be put in place, but the old file would remain accessible for historical purposes. Each write would still be a one-time operation, but you would be able to write the same file many times.

Azure Key Vault

Microsoft's Azure Key Vault service is a bit of an outlier when it comes to security tools. It doesn't fit neatly into the networking, compute, or storage categories, and yet it is integral to supporting the security features of those categories.

Azure Key Vault stores and manages access to sensitive information and provides encryption services on demand. There are three types of data stored in Key Vault:

Certificates
: X.509 certificates that can be generated by Key Vault or uploaded from an external source

Keys
: Encryption keys available to perform cryptographic operations like encrypt, decrypt, sign, and verify

Secrets
: Arbitrary data written to Key Vault and available for retrieval

The sensitive information stored in Key Vault can be further protected by using a managed hardware security module (HSM). A managed HSM is a single-tenant, fully managed service that provides an additional layer of security and access control on top of the Azure Key Vault service.

Key Vault integrates with many of the services we've already discussed. Solutions like Azure Storage, AKS, and Azure VM disk encryption allow you to select a platform-managed key from Microsoft or a customer-managed key (CMK) you provide. If you choose a CMK, then Azure Key Vault or Azure Key Vault Managed HSM is where you will store and possibly generate the key to be used by the service.

Key Vault can also store and disseminate certificates to various running applications and services in Microsoft Azure. Compute resources and applications running in Azure can use their assigned managed security identity (MSI) to retrieve the certificate automatically and rotate it as needed.

Azure Confidential Computing

The Linux Foundation defines confidential computing as:

> [Protecting] data in use by performing computation in a hardware-based, attested Trusted Execution Environment. These secure and isolated environments prevent unauthorized access or modification of applications and data while in use, thereby increasing the security assurances for organizations that manage sensitive and regulated data.

In short, this is the third leg of data protection: we use network encryption to protect data in transit, disk encryption to protect data at rest, and confidential computing to protect data while it is in RAM or during computation.

An important concept in confidential computing is the trust boundary: the portion of the infrastructure that can be attested as being

trustworthy. The point of a boundary is to ensure that things outside the trust boundary can be breached without allowing access to the memory within the trust boundary. Confidential computing technologies help prevent Azure operators and unauthorized third parties, including users of other VMs on the same server, from accessing or altering data in server memory.

Confidential computing on VMs works by allowing the VM to bind encryption functions directly to the Virtual Trusted Platform Module (vTPM). This means that all content on the VM (all components, disks, etc.) is accessible only by the VM. You have the option to do this to just the data disks, or to use confidential computing on the OS disk and temp disks as well to maximize security. As with Disk Encryption, not all Linux distributions (or all versions of a particular distribution) are supported. Additionally, some regions have better support for confidential computing than others. Definitely check that your selected region has all these features available before you begin deployments.

The final component of confidential computing on Azure is attestation. Attestation is key to ensuring that sensitive data is not loaded into a confidential VM or a confidential container group until the underlying hardware and the software within the memory trust boundary is validated. Attestation is supported by Microsoft's first-party service on Azure, Microsoft Azure Attestation (MAA).

Azure provides this capability from IaaS instances that have the "C" subfamily indicator. One example is the DCasv5 series that runs on AMD, although there are others that run on various Intel chips as well. Encryption is again managed via Azure Key Vault to ensure your keys are safe and easily rotatable. Confidential computing is also possible in PaaS services such as Azure SQL, Azure Virtual Desktop, and Azure DB for PostgreSQL, among others. Confidential container groups are available as part of selected Azure PaaS and serverless offerings. These include AKS, ARO, and ACI.

Microsoft Defender for Cloud

Microsoft Defender for Cloud is a CNAPP designed to help protect workloads in hybrid and multicloud environments. *CNAPP* is a relatively new term defined by Gartner, and it's meant to describe a holistic solution that covers DevSecOps, cloud security posture

management, and cloud workload protection. Microsoft Defender for Cloud helps to build the code-to-cloud security story.

DevOps Security

Your Linux workloads are built from applications, and if you're developing those applications in-house, you can leverage the code scanning and pipeline integration offered by Defender for Cloud. There's a good chance your development team already uses static and dynamic code analysis tools in their CI/CD pipelines, and Defender for Cloud isn't meant to replace that functionality. Rather, it integrates with tools like GitHub Advance Security to provide visibility and risk analysis across your CI/CD pipelines and actual deployed applications and infrastructure.

Cloud Security Posture Management

An ounce of prevention is worth a pound of cure, and that's definitely the case with security. But you can't protect what you don't know about. To figure out your security posture, Microsoft Defender for Cloud can hook into Azure and other cloud providers to assess your current configurations and policies against the Microsoft cloud security benchmark and other compliance standards like PCI/DSS and HITRUST. It also provides contextual posture assessments and remediation guidance around posture issues.

If your Linux workloads will be supporting a regulated industry, or you just want to make sure you're aligned with the cloud security baseline, Defender for Cloud can scan your current deployments and give you a security score and detailed report. Defender for Cloud has a pretty generous free tier that includes the cloud security posture management component.

Cloud Workload Protection

Once you know what is actually running in your environment and where your potential vulnerabilities are, you can apply protections. Defender for Cloud monitors and protects Azure VMs running Linux through a few key features and services:

- Just-in-time network access
- File integrity monitoring

- Vulnerability scanning
- Threat detection and response

These built-in detection tools are designed to work with Linux environments, focusing on common attack vectors such as SSH brute-force attempts, anomalous process execution, and kernel-level exploits. Defender for Cloud continuously analyzes system behavior, generating security alerts based on deviations from expected activity. Additionally, it can leverage audit logs and system telemetry to provide deeper insights into security events, allowing teams to investigate and respond more effectively.

Your Linux workloads don't exist in a vacuum. They also rely on the underlying infrastructure and networking, and they might interact with managed databases and Kubernetes. Defender for cloud can also monitor and protect cloud databases and containers as well as analyze your infrastructure and networking for possible vulnerabilities.

We should note that some of these features require additional plans and licensing and may already overlap with tools your security team is using, so plan accordingly. Any robust security solution is going to be an amalgam of products, services, and processes.

Microsoft Sentinel

Microsoft Sentinel is a cloud native SIEM and security orchestration, automation, and response (SOAR) solution designed to provide real-time visibility and threat detection across an organization's infrastructure. For teams migrating or deploying Linux-based workloads in Azure, Microsoft Sentinel helps centralize log collection, detect anomalies, and automate incident response, making it easier to manage security at scale. Unlike traditional SIEM solutions that require extensive infrastructure management, Sentinel operates as a fully managed service, reducing the operational overhead of maintaining security analytics tools.

When running Linux workloads in Azure, security teams need deep visibility into system logs, authentication events, and network activity to detect potential threats. Sentinel integrates natively with Azure Monitor, Defender for Cloud, and third-party security tools to ingest and analyze logs from Linux VMs, containers, and Kubernetes clusters. It supports Syslog and Common Event Format (CEF),

allowing engineers to forward logs from Linux hosts with minimal configuration. Once ingested, Sentinel applies built-in threat intelligence and machine learning models to identify suspicious activity such as unauthorized access, privilege escalation, and anomalous process execution.

Microsoft Sentinel can then use custom or preconfigured Playbooks to automate threat detection response or vulnerability remediation. A catalog of common tasks is included with Microsoft Sentinel, but the Playbooks use Microsoft Logic Apps, so they are highly customizable to your needs.

Key Sentinel features for Linux workloads include:

Log ingestion and analysis
: Supports Syslog, CEF, and custom log sources for centralized monitoring of Linux workloads through the Azure Monitoring Agent (AMA)

Advanced threat detection
: Uses AI and behavior-based analytics to detect anomalies such as SSH brute-force attacks, unusual sudo activity, and process injection

Automated incident response
: Leverages playbooks to automate responses, such as isolating compromised VMs or notifying security teams

Threat intelligence integration
: Correlates security events with known threat indicators to detect and prioritize potential attacks

Custom analytics and alerts
: Allows practitioners to create custom KQL (Kusto Query Language) queries to generate security alerts tailored to their environment

Multicloud and hybrid support
: Extends security monitoring beyond Azure to include AWS, Google Cloud, and on-prem Linux environments

A key advantage of Sentinel for Linux security is its ability to perform real-time correlation of security events across multiple sources. For example, an SSH login attempt from an unfamiliar IP might seem low risk in isolation, but when combined with unusual sudo

activity and unexpected outbound network traffic, it could indicate a compromise. Sentinel's analytics engine helps identify these patterns, reducing the time it takes to detect and respond to security incidents.

Sentinel also includes heavy integration with all Azure services for log collection, monitoring, and response. For applications that span multiple services, Sentinel offers a complete picture of security incidents and how far they span.

Summary

Securing Linux workloads in Azure requires a thorough understanding of the Microsoft shared responsibility model as well as all of the security tools that are available within Azure. As we have seen, cloud security is not about a simple solution, nor is it simply a checkbox. Cloud security is an exercise in defense in depth that must consistently be reevaluated for efficacy and to potentially implement new features or capabilities. The key to success lies in designing your infrastructure with security in mind and leveraging the security tools available to you effectively.

CHAPTER 7

Automating Linux Deployments on Azure

As organizations scale their applications and infrastructure, manual deployments and configurations quickly become impractical and prone to errors. To address this, modern DevOps practices—GitOps, CI/CD, automation, and infrastructure as code—provide a framework for consistent, repeatable, and automated deployments. These practices are not just about efficiency; they fundamentally enhance the reliability, scalability, and security of Linux workloads on Microsoft Azure.

In this chapter, we explore how to leverage Azure's ecosystem of automation tools and integrations to deploy and manage Linux applications seamlessly. We'll cover:

Infrastructure as code (IaC)
: Using Azure Resource Manager (ARM), Bicep, Terraform, and Ansible to manage infrastructure as version-controlled code

Code hosting and automation platforms
: Deploying applications and infrastructure consistently with GitHub Actions and Azure DevOps

Application delivery models
: Exploring Azure App Service, Azure Container Apps, AKS, and Azure VMs

Microsoft Copilot for Azure
: Accelerating infrastructure planning with AI-powered automation

Infrastructure as Code (IaC)

IaC allows you to define infrastructure in human-readable configuration files, enabling automated, repeatable deployments. The rise of IaC has coincided with the growth of cloud computing, which provides a consistent API for IaC tools to hook into.

Azure supports multiple IaC tools, each with its own strengths and potential drawbacks.

Azure Resource Manager (ARM)

ARM is Azure's deployment and management service. ARM templates use JSON to define Azure resources and their configurations. When it comes to feature and service support, ARM templates are the gold standard.

While powerful, ARM templates can be verbose and difficult to maintain. Hand-editing JSON can be challenging, and ARM templates also lack support for common programming features, like inline comments and reusable modules. These shortcomings led to the development of Bicep as a simplified abstraction layer.

Bicep

Bicep is a domain-specific language for deploying Azure resources. It maintains full parity with ARM templates but is much easier to write and manage. Its primary benefits are:

Simplified syntax
: Cleaner and more readable than raw ARM JSON

Modular design
: Promotes reusable components for complex architectures

Native Azure integration
: Full compatibility with Azure services

Bicep has the same limitations as ARM templates in the sense that it only supports Azure services. For a more cloud-agnostic solution, Terraform provides a viable alternative.

Terraform

Terraform is the industry standard for multicloud IaC. Its declarative syntax allows you to define both Azure and multicloud resources consistently. The primary benefits of using Terraform are:

Cross-cloud compatibility
: Deploys infrastructure to Azure, Amazon Web Services (AWS), Google Cloud, and more

State management
: Tracks deployed resources, enabling drift detection and rollback

Modular configurations
: Supports reusable modules for simplified infrastructure scaling

Terraform supports all aspects of Azure through its AzureRM and AzAPI providers. The AzAPI provider, in particular, gives Terraform day-one support for all services and features available to ARM templates.

Ansible

While ARM, Bicep, and Terraform focus on infrastructure provisioning, Ansible excels in configuration management and application deployment. Ansible makes use of declarative configurations in the form of playbooks, which are typically stored in Git repositories.

Agentless management
: SSH-based configuration without agents

Idempotent operations
: Scripts can be safely rerun without risk of inconsistent state

Postdeployment automation
: Ideal for configuring Linux VMs, installing packages, and managing services

This is by no means an exhaustive list of IaC and configuration management tools. Azure also natively supports Power Shell Desired State Configuration, and other tools, such as Puppet and Chef, remain popular. However, in our experience, Terraform and Ansible tend to be the most popular tools for managing the infrastructure operating systems that support Linux workloads.

Deployment Workflows for Infrastructure

Through integrations with Azure DevOps and GitHub, Azure supports the principles of continuous integration (CI) and continuous delivery/deployment (CD), allowing infrastructure to be deployed consistently and securely. GitHub Actions and Azure DevOps Pipelines can both be leveraged to provide CI/CD pipelines for IaC that include:

CI
: Automated testing and validation of infrastructure code upon each commit

CD
: Automatic deployment to Azure upon successful tests

A key benefit of using IaC is the consistency of deployment across environments. This is typically accomplished through a code promotion workflow across multiple environments, with approval gates after each deployment.

Azure DevOps and GitHub Actions enable multistage pipelines, as shown in Figure 7-1, where code is promoted across three stages:

Development
: Automated testing and integration

Staging
: Integration and load testing in a production-like environment

Production
: Final deployment, often gated by manual approval or automated checks

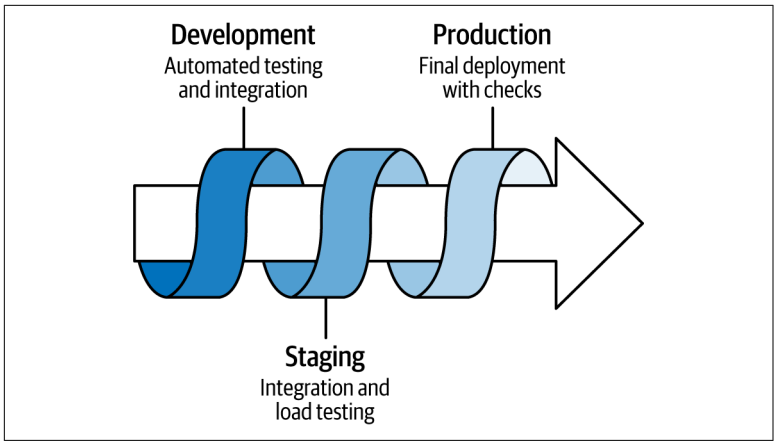

Figure 7-1. Example of a multistage pipeline for promotion of code through lower environments to production

Code Hosting and Automation Platforms

To automate Linux deployments on Azure, you need platforms for version control and pipeline automation and tools that can help you implement CI/CD of code for both your infrastructure and application. While there are many solutions available, the primary tools used in the Azure ecosystem are GitHub and Azure DevOps.

GitHub and GitHub Actions

GitHub is the world's largest platform for code hosting, version control, and collaboration. Augmented with GitHub Actions, it becomes a complete solution that automates testing, building, and deploying Linux applications directly to Azure. The key capabilities of GitHub are:

CI/CD
: Automate builds, tests, and deployments with every code push

IaC integration
: Deploy ARM, Bicep, or Terraform templates seamlessly

Secure secrets management
: Store access tokens, SSH keys, and passwords securely

Azure DevOps: Pipelines and Repos

Azure DevOps is a comprehensive platform that includes tools for source control, pipeline automation, artifact storage, and project tracking. It also includes tight integrations with Azure and Entra ID, allowing for seamless authentication and authorization flows that don't require additional software or plug-ins. The core components of Azure DevOps are:

Azure Repos
: Version-controlled storage for application code and infrastructure definitions

Azure Pipelines
: Multistage automation for building, testing, and deploying applications

Integration with Azure Services
: Simplified deployments to Azure App Service, VMs, and AKS

Azure DevOps is ideal for organizations that require strong integration with Azure and advanced release management capabilities.

The code you deploy to Azure using these platforms can take the form of traditional application code, configuration management manifests, or IaC. Since infrastructure forms the foundation of your application, we will investigate IaC options first.

Application Delivery Models

Once the infrastructure is in place, applications must be hosted and managed effectively. Azure provides several hosting options tailored for Linux applications. We reviewed these options in depth in Chapter 2, but now it's time to look at them from an application deployment perspective.

Azure App Service

Azure App Service supports both traditional web applications and the hosting of Azure Functions. Both web applications and Functions support the concept of deployment slots, which enable seamless switching between versions of code. Each integrates with Azure DevOps or GitHub for both source control and code deployment. While you can deploy code directly from the Azure CLI, configuring

CI/CD with Azure DevOps or GitHub Actions brings the benefits of automation, enhanced testing, and consistent deployment.

Azure Container Apps and Instances

Azure Container Apps is a serverless container platform designed for microservices and event-driven applications. Unlike AKS, it abstracts away the complexity of container orchestration, letting you focus on your application code.

Azure Container Instances are an even more lightweight, serverless container platform that allows you to deploy containers in groups, but it doesn't include the Kubernetes abstractions of Azure Container Apps.

For both of these services, there is no direct integration between GitHub Actions or Azure DevOps. However, you can leverage either of those services to write deployment scripts to deliver your application code as a container. A common workflow is to publish the container image to Azure Container Registry (ACR) and use the integration between ACR and Container Apps or Instances to roll out the new image containing your updated application.

Azure Kubernetes Service (AKS)

AKS is a fully managed Kubernetes service that simplifies the deployment, management, and scaling of containerized applications. AKS provides complete control over your Kubernetes clusters while offloading the heavy lifting of maintaining the control plane to Azure.

Since it is a fully functional Kubernetes cluster, unlike Azure Container Apps, you can leverage popular GitOps tools for application deployment. GitOps is the practice of driving application deployment and configuration through code stored in a Git repository. The repository serves as the source of truth, and GitOps tools are responsible for ensuring that deployed environments match the state described in the repository.

Common tools for GitOps deployment include Argo CD and Flux, both of which can be deployed and used within AKS to monitor application code repositories and deploy applications on a continuous basis.

Azure Virtual Machines (VMs)

For applications requiring full OS control or custom configurations, Azure VMs offer the greatest level of flexibility. As we've already discussed, Linux VMs on Azure support a wide range of distributions, including Ubuntu, Rocky Linux, Red Hat, and SUSE.

For application deployment, your options are wide open when it comes to Azure VMs. Terraform and Ansible are commonly used to bootstrap each VM and prepare it for application deployment. This might include installing prerequisites, applying patches, or deploying an agent to support application delivery. Azure DevOps pipelines and GitHub Actions are both capable of orchestrating Ansible or Terraform runs both when a machine is first deployed and when the configuration is altered.

Table 7-1 summarizes the various application deployment options available and the typical deployment methods for each option.

Table 7-1. Summarization of application deployment options on Azure

Feature	Azure App Service and Functions	Azure Container Apps and Instances	AKS	Azure VMs
Management level	Fully managed	Partially managed	Partially managed	Self-managed
Best for	Web apps, APIs	Microservices, APIs	Distributed apps, microservices	Legacy apps, custom OS
Application deployment methods	GitHub, DevOps	GitHub, Azure DevOps	GitOps (ArgoCD, FluxCD)	Custom scripts, Ansible

Microsoft Copilot for Azure

Microsoft Copilot for Azure is an AI-powered assistant that integrates directly into the Azure Portal and development environments. Its primary goal is to automate and accelerate cloud operations by providing intelligent recommendations, real-time assistance, and even automated code generation for infrastructure.

Key capabilities include:

Resource planning and estimation
: Quickly determine the best infrastructure setup based on workload requirements

Template generation
: Automatically generate ARM, Bicep, or Terraform templates for deploying VMs, AKS clusters, and more

Configuration inspection
: Audit existing deployments for security and compliance, suggesting improvements or optimizations

Automated scripting
: Generate Power Shell, CLI, or Terraform scripts to automate repetitive tasks

When planning out a new Linux workload deployment on Azure or analyzing your existing environment, Copilot for Azure can provide context-aware guidance and recommendations. For example, if you need to deploy a high-availability Ubuntu web server in Azure, you can simply tell Copilot: "Set up two VMs in East US for web hosting, with security best practices and cost optimization."

Copilot recommends the right VM sizes, estimates monthly costs, generates a ready-to-deploy Bicep or Terraform template, audits your network security rules, and even produces Azure CLI scripts—dramatically speeding up your design and deployment process.

Summary

Cloud native application delivery demands agility, reliability, and scalability, all of which are achieved through effective automation. In this chapter, we explored how Microsoft Azure provides a powerful platform for deploying and managing Linux workloads at scale. By leveraging IaC with tools like ARM, Bicep, Terraform, and Ansible, infrastructure provisioning becomes predictable and repeatable.

We also covered how GitHub Actions and Azure DevOps provide CI/CD pipelines, allowing for consistent application updates with little to no downtime. These automation pipelines not only streamline deployments but also enhance security and reliability by integrating automated testing and compliance checks.

CHAPTER 8
Azure Cost Optimization

Designing and managing an Azure environment that is secure, efficient, and economical requires a lot of technical acumen. But it also requires something else—namely, the money to pay the bills. Running a business in the cloud means that you are going to have difficult conversations with management about costs.

For decades, IT has been fighting for more funding just to keep the lights on, being tasked with doing more with less. Thankfully, these days it is becoming more and more common knowledge that effective cost management is a fundamental pillar of your cloud adoption strategy.

According to the Microsoft Cloud Adoption Framework, building a cost-conscious cloud infrastructure, as described in Figure 8-1, requires three main things: visibility (know where your money is going), accountability (be able to explain and defend costs and purchase decisions), and optimization (prove that every dollar spent is spent in an efficient way—the most bang for your buck).

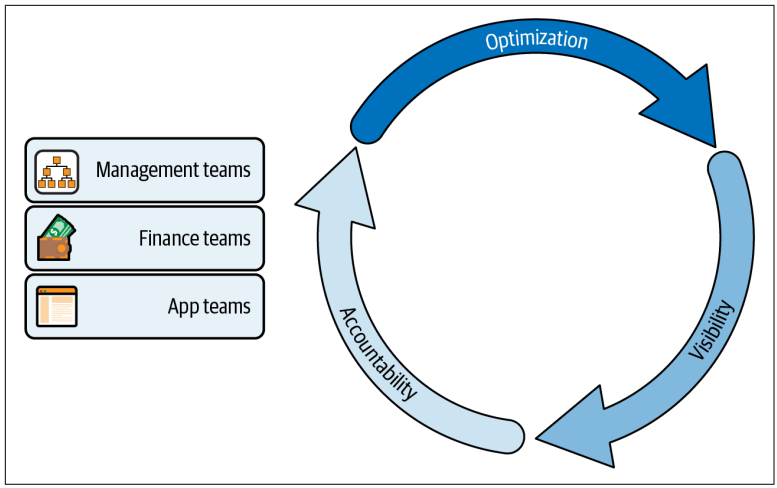

Figure 8-1. Microsoft outline of the cost-conscious organization from the Microsoft Cloud Adoption Framework for Azure[1]

Azure offers a multitude of services presenting opportunities and challenges for organizations operating in the cloud. From IaaS instance decisions to Entra ID licensing options to Microsoft Defender for Cloud feature decisions, every choice has a profound effect on your security, productivity, and monthly bill. The key is understanding how to effectively leverage Azure's pricing models, optimization tools, and security services without breaking the bank.

In this chapter, we'll explore how to approach the myriad options that Azure provides in order to keep things as effective as possible and keep the CFO happy. We'll cover everything from rightsizing your VMs and optimizing storage tiers to making smart decisions about security service subscriptions and compliance tools. The goal here is not to be prescriptive—every environment and situation is different—but to provide a variety of strategies that you can utilize to keep things safe and secure and within budget. What follows is a not completely comprehensive guide to doing exactly that.

1 "Build a Cost-Conscious Organization," Microsoft Ignite, February 28, 2023, *https://learn.microsoft.com/en-us/azure/cloud-adoption-framework/organize/cost-conscious-organization*.

Core Principles for Cost Optimization

In this section, we will talk about some of the ways that cost management can come into play with specific Azure services, but first let's look at some high-level principles that will help you contain costs:

Commitment offers
: Long-term commitments like Reserved Instances (RIs) deliver the largest savings across all of Azure.

Tagging and tracking
: Everything in Azure should be tagged so that its usage (and cost) can be tracked.

Rightsizing
: It's critical to use properly sized resources so that you get maximum performance at the lowest possible cost.

Minimum resource allotment
: If something is running in Azure that doesn't have a known purpose, it is probably wasting money.

Proactive monitoring
: Utilize budget alerts to know if any cost overruns are happening (or are close to happening).

Microsoft Cost Management

Azure has an amazing array of products and services available to you as you build out your infrastructure. Roughly speaking, any cloud provider will charge for these products and services on a utility model—you will be billed for use unless you pay in advance. Let's take a look at some of the ways that you can minimize your cloud bill while still maintaining a high level of both security and performance. We will talk about a few notable examples in the IaaS space, followed by some of the tools that Microsoft makes available to help you with cost optimization.

Azure IaaS

Let's start by taking a look at the options available to save money on VM instances that you spin up in your Azure environment. One thing to note right off the bat is that staying current and using more

modern instances is almost always a great way to save money. For example, a D4as v4 instance will cost 19.2 cents per hour to run. Upgrading to the latest version, the D4as v6 drops that price to 18.2 cents per hour. As always, the best way to check for the best price per instance is to check the Azure Pricing Calculator—or ask Copilot.

Azure Reserved VM Instances

There are a few ways to keep costs down with VMs. First is Reserved Instances (RIs). Long-term commitments deliver the biggest savings across Azure Services. RIs are done per VM and region at the time of purchase, and the discounts are significant. For example, at the time of this writing, a standard D4ads v5 Linux VM running in the East US region would cost you about $150/month to run in the standard PAYG billing model. Going to a one-year RI drops that price to around $89/month and a three-year RI down to $57/month! As shown by Figure 8-2, this reduces the total cost over three years by 41% for a one-year RI and 62% for a three-year RI. This alone should show you why RIs are one of the best values available in Azure. (It should be noted that Azure reservations are available for a variety of Azure services such as Azure AI Foundry, Azure SQL Database, and Microsoft Fabric.)

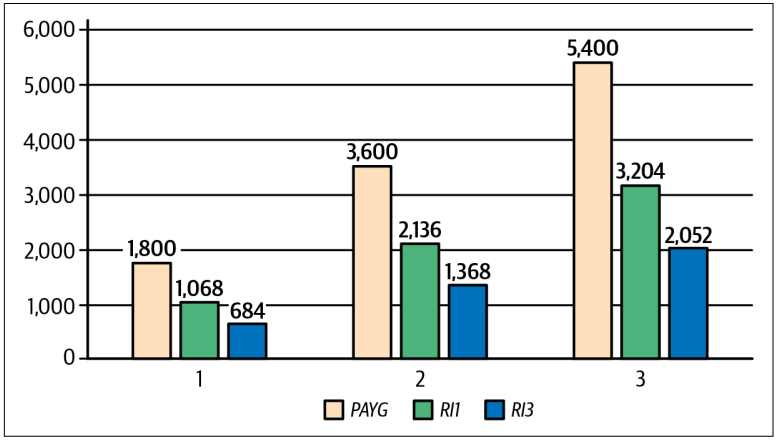

Figure 8-2. Comparison of costs for an Azure VM with PAYG, one-year RI and three-year RI options

RIs can be canceled, exchanged, and even in certain cases (and up to certain limits) refunded, which gives the user a lot of flexibility to use them to their fullest capacity even if things change in the future.

Azure Spot VMs

If you have a workload that can handle interruptions, then Azure Spot VMs are a great option. Spot VMs work by allowing you to spin up VMs that take advantage of unused capacity in an Azure datacenter. So, for example, if RIs and PAYG VMs are only using 25% of an Azure datacenter's compute capacity, then you can use some of that unused 75% at a reduced cost. The downside of Spot VMs is that if an RI or PAYG workload comes in and needs the compute that you are using for your Spot VMs, your VMs will be deallocated. Still, for things like batch jobs, distributed computation, big data, or even dev/test machines, this is a great option. The pricing is variable based on region and availability, and you can set a maximum cost that you are willing to pay. When the VMs are available at the price you are willing to pay, they will be spun up. Cost savings can be even greater with Spot VMs than with RIs—in many cases, up to 90% off of retail price. Microsoft has a recorded webinar available for your reference that goes into Spot VMs—how they are priced, how they are used, and so on.

Azure storage tiers

Picking the right storage tier for your data can help you with cost management much in the same way that picking the right VM instance can. There are five levels of storage available that range from Premium (the highest performance and lowest latency) all the way down to Archive (intended for extremely infrequent access). The difference in price is significant, so it (no pun intended) pays to really investigate what your workloads need and how best to spread the storage between these tiers. And of course, adding Reserved Capacity (the storage version of an RI) brings in even more significant cost savings.

Table 8-1 shows the pricing for storage as it stands at the time of this writing.

Table 8-1. Microsoft data storage pricing, PAYG, per GB per month

Data storage prices PAYG	Premium	Hot	Cool	Cold	Archive
First 50 terabytes (TB)/month	$0.15/GB	$0.018/GB	$0.01/GB	$0.0036/GB	$0.002/GB
Next 450 TB/month	$0.15/GB	$0.0173/GB	$0.01/GB	$0.0036/GB	$0.002/GB
Over 500 TB/month	$0.15/GB	$0.0166/GB	$0.01/GB	$0.0036/GB	$0.002/GB

Pricing Models and Programs

We talked about reservations so we won't rehash the cost benefits of those here. However, there are two other important programs available that you should be aware of—namely, Azure Savings Plan for Compute and Azure Hybrid Benefit for Linux.

Azure Savings Plan for Compute

In contrast with RIs, which provide individual savings on a specific VM and region, a savings plan commits you to spending a certain dollar amount per hour on select compute services. The savings can then be applied across your entire Azure environment—across subscriptions, resource groups, and so on regardless of VM and region. This provides a lot of flexibility and the ability to save even when your workloads are much more dynamic than a standard VM, but you know organizationally that they will still be running.

For example, say you have a set of VMs running as domain controllers in the cloud and a large file server. These workloads are very static and are ideal for RIs—they are likely to stick around for a long time with little to no need to change their compute requirements. If you have a large workload (or many disparate workloads) that you know will cost a certain amount of money but the resources will not remain consistent, then a savings plan makes sense.

To use a savings plan, you will commit to spending a certain amount of money on Azure resources for either one or three years. Instead of it only being applied to a specific VM in a specific region, this money will automatically be applied to the compute service that results in the largest savings. You will receive deeper discounts on a three-year savings plan than if you were on the one-year option. This gives you a stable bill each month in addition to the discounts being applied wherever resources are used within the savings plan. These fixed or upfront costs also make it easier to forecast your

cloud spend. See Figure 8-3 for an example of the various payment options and the various ways you can save money on IaaS instances.

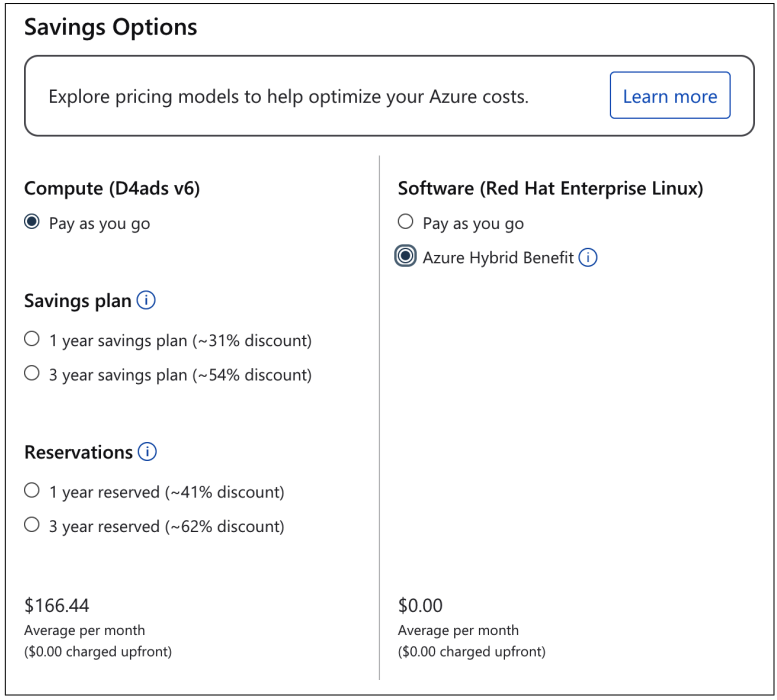

Figure 8-3. The Pricing Calculator

Azure Savings Plan is a great option—especially for large organizations committed to running variable amounts of compute across multiple regions—but there are caveats. For example, a savings plan cannot be canceled, refunded, or exchanged. All savings plan sales are final. Additionally, savings plan usage is defined hourly and is use-it-or-lose-it. You cannot roll time over from one hour to another.

Savings plan commitment recommendations can be obtained from Azure Advisor (more on that later in this chapter). You can find savings plan recommendations by simply searching for "savings plan" in the top search bar. Once in the savings plan section, you should be able to see recommendations based on your Azure usage simply by clicking on the "View recommendations" link. (If you do not already have savings plans, you will need to click the "Purchase

Now" button.) The recommendations on Azure Savings Plan sizing is based on your historical usage from a selectable period.

Azure Hybrid Benefit for Linux

The Azure Hybrid Benefit program applies to both Windows VMs and Linux VMs. In short, it gives you two options when it comes to licensing the OS. In a standard PAYG or RI VM, you have the option to pay for both the compute resources and the licensing cost of the OS as shown in Figure 8-4. Alternatively, if you have existing licenses purchased through a third party, you can elect to pay only for the compute resources you are using via Azure. Being able to convert from one model to the other is the Azure Hybrid Benefit.

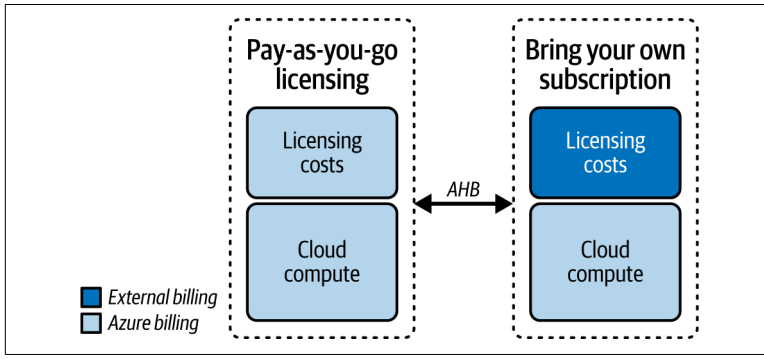

Figure 8-4. A comparison of Azure Hybrid Benefit licensing costs options

With Azure VMs, you are able to use Azure Hybrid Benefit for Linux on RHEL and SLES. This is useful when you have an enterprise license for one of these distributions already. The Azure Hybrid Benefit ensures that you are not accidentally paying for an OS license twice. Additionally, there are often bulk discounts that can be obtained from Red Hat or SUSE, meaning that the license you pay for from them will be less costly than the one that is bundled with the compute resource in the standard PAYG VM instance deployment.

Azure Dev/Test pricing

One cost-saving option many may not be aware of is Dev/Test pricing. A Visual Studio subscription is required in order to take advantage of this, but if you are able to use it, you can get savings

up to 55% across the board for nonproduction environments. The cost for a Visual Studio subscription can vary depending on your organization's needs and requirements, but the benefits from cost savings on Azure workloads alone more than make up for this expense. (Note: you also get Azure credits each month as a part of the Visual Studio subscription.)

Azure Dev/Test pricing is not just limited to IaaS—there are lower rates available for DevTest Labs (a service in Azure that allows dev teams to quickly create, deploy, and destroy VMs for development and testing), Azure Virtual Desktop, and more. The full details of a Visual Studio subscription are outside the scope of this book, but you can find a lot more information on what's available to you at the Microsoft homepage for Azure Dev/Test pricing.

Cost Management Tools

There are a number of tools built into Azure to help with cost management. As we discussed at the beginning of the chapter, Microsoft has put a lot of investment into helping you be cost conscious and make the most out of your Azure infrastructure. Think about it: it is in Microsoft's best interest to make sure you are confident your money is well spent. If you didn't think the price of Azure was worth the benefits, you would leave. To that end, Microsoft has built cost management into its Cloud Adoption Framework as well as into a number of tools available at no additional cost to you in the Azure portal. Let's take a look at them one by one.

Azure Advisor

If you are looking to optimize your environment (from a cost perspective or otherwise), Azure Advisor is an invaluable tool. Advisor is built directly into the Azure portal and will show you scoring from 0 to 100% on cost, security, performance, and more. It is built to allow you to drill down for more specific recommendations in all these categories. The recommendations they provide are listed in impact/priority, in a simple scale of high/medium/low.

From a strictly cost management perspective, the Advisor view of costs will give one- and three-year savings plans and reservation recommendations. It will also highlight underutilized resources and suggest rightsizing VMs or other infrastructure, and it will make recommendations about things in your environment that could be

deallocated. This is a key point: Advisor identifies things in your environment that are wasting your money.

Here are some examples of alerts that it will provide:

- "Consider a virtual machine Reserved Instance to save over the on-demand costs."
- "Consider purchasing a savings plan for compute to unlock lower prices."
- "Rightsize or shut down underutilized virtual machines."

Each of these alerts will be connected to a detailed report showing the impacted resources and a cost justification for each recommendation. This same level of reporting is available for all the other tiles in the Advisor dashboard. At the risk of sounding hyperbolic, we will say this: it is hard to overstate the value you can get out of Azure Advisor. If you are in charge of any amount of Azure infrastructure, Azure Advisor should be your first stop at the beginning of every workday.

Pricing Calculator

The Azure Pricing Calculator is a public website that gives you—*wait for it*—pricing on basically everything you can implement in your Azure environment. We have referenced it many times in this book, so we won't belabor the point, but use it whenever you need to see what retail pricing is at any given point in time. (This information is also available via Copilot through the Azure portal.)

The calculator is nice because you can build out resource what-if scenarios and make changes in real time. For example, what if we move 10 VMs from a D-class instance to an A-class instance, or vice versa? What if we move storage from Premium to Hot tier? What would this workload look like cost-wise PAYG versus built as RIs? And so on and so forth. The tool helps you model scenarios and compare pricing options so that you can make informed decisions about how to move forward with no surprises.

Cost Management and Billing

Finally, the Cost Management section of the portal provides automated recommendations and reports about your spending that help show exactly where each dollar went. You can get pricing estimates

here (same as from the Pricing Calculator), but you can also get a lot of reporting and analytics that help you make sense of the dollars spent. This is one place where tagging can really help you—the more accurate your tagging policy, the better reporting and insights you can get out of these tools.

It has also become increasingly common to leverage Azure subscriptions as a boundary for cost management reporting. Consider creating dedicated subscriptions for line-of-business applications and development teams to augment your existing tagging strategy.

Cost Management also provides monitoring and alerting features. These can be set on very specific pieces of your infrastructure so you can be aware of costs as the month passes. This is very helpful, for example, if you set a billing alert on a business unit or a project. If you have an expected cost, and that cost gets close, Cost Management will alert you. This can save you untold amounts of money in the event that, for example, an auto deploy script goes haywire and builds 1,000 VMs instead of 10.

There is a lot more in this tool that is worth looking into, including automation and optimization recommendations and capabilities. A full breakdown of everything that this tool can do is available at the feature homepage.

Summary

Managing Azure costs is not just a seesaw of cost versus performance—it's about creating a sustainable environment that satisfies both needs. Microsoft has invested heavily in tools to help you do exactly that. By implementing the core principles of cost optimization, you will be far along the road of maximizing the value of money spent in Azure. A sustainable cloud infrastructure requires sustainable (and defendable) spending, and using the tools and cost-saving programs available to you in Azure will help you get to that point without sacrificing anything when it comes to your organization's productivity.

About the Authors

Ned Bellavance is an IT professional and technical educator with more than 20 years of experience in the field. He's been a helpdesk operator, systems administrator, cloud architect, and product manager. Most recently, Ned runs Ned in the Cloud LLC, where he develops courses, runs multiple podcasts, writes books, and creates original content for technology vendors. Ned has been a Microsoft MVP since 2017 and a HashiCorp Ambassador since 2020. He has three central pillars: embrace discomfort, plan to fail, and be kind.

Chris Hayner is a seasoned IT professional with decades of experience spanning operating systems, infrastructure, cloud computing, and cybersecurity. His career began in the datacenter, where he managed a diverse range of systems, from mainframes to AlphaServers to white box x86 servers. From there, his scope expanded to include virtualization, cloud technologies, and overarching cybersecurity and IT strategies. For the past 15 years, Chris has worked in the consulting realm, serving as a subject matter expert, architect, and analyst. In these roles, he has helped hundreds of organizations bridge the gap between business objectives and IT solutions, driving innovation and operational success. In addition to holding a CISSP certification and numerous vendor and industry credentials, Chris earned an MBA from Temple University, equipping him with a unique blend of technical expertise and business acumen.