Microsoft Azure Fundamentals Certification and Beyond

A complete AZ-900 exam guide with online mock exams, flashcards, and hands-on activities

Third Edition

Foreword by

Peter De Tender
Lead Microsoft Technical Trainer

Steve Miles

packt

Microsoft Azure Fundamentals Certification and Beyond

Third Edition

A complete AZ-900 exam guide with online mock exams, flashcards, and hands-on activities

Steve Miles

‹packt›

Microsoft Azure Fundamentals Certification and Beyond

Third Edition

Portfolio Director: Kartikey Pandey
Relationship Lead: Aaron Tanna
Project Manager: Sonam Pandey
Content Engineer: Apramit Bhattacharya
Technical Editors: Sumant Jadhav and Simran Ali
Copy Editor: Safis Editing
Indexer: Tejal Soni
Proofreader: Apramit Bhattacharya
Production Designer: Ganesh Bhadwalkar
Growth Lead: Shreyans Singh

First published: January 2022
Second published: January 2024
Third Edition: March 2026

Production reference: 1170326

Published by Packt Publishing Ltd.
Grosvenor House
11 St Paul's Square
Birmingham
B3 1RB, UK.

ISBN 978-1-80670-241-1

Foreword

When this book was first written, the goal was clear: help people understand Azure well enough to pass the AZ 900 exam and confidently take their first steps into the cloud. The first edition explained the fundamentals. The second refined them. This third edition does something more important: **it places Azure fundamentals in the context of how the cloud is actually used today**.

Cloud computing is no longer just about learning service names or memorizing definitions for an exam. Azure now underpins **AI-driven workloads**, **global platforms**, **regulated environments**, and **real operational trade-offs**. Fundamentals are obviously still important, but instead of being OK enough to get started, they have also become more consequential. This edition reflects that reality.

The single biggest driver for this rewrite was alignment with the **latest 2026 Skills Measured updates**, but the changes go far beyond exam coverage. Throughout the book, you will notice a shift in tone and structure. The content is more conversational, less mechanical, and intentionally less focused on "exam style answers". Instead, it reflects how experienced practitioners like yourself think, explain, and reason about Azure in real environments. Which makes this third edition even more useful for anyone who not only wants to earn the credential but is also tasked with actually managing the platform.

Several chapters have been reworked and expanded to give core Azure building blocks the space they deserve. Compute, storage, and networking are no longer treated as subtopics under a single umbrella; they are explored as distinct decision areas, each with its own trade-offs, operating models, and architectural implications. This allows concepts to be positioned more solution-oriented, interconnected, and overall make sense beyond a checklist of features.

You will also see a stronger emphasis on **judgment over memorization**. Availability, latency, cost, governance, and composite SLAs are discussed as design considerations, not abstract definitions. Storage is framed as an architectural choice, not a menu of options. Azure Compute is explained in terms of operating models, not just service descriptions. These are the kinds of conversations that happen in real projects, and they are now reflected here.

Perhaps most importantly, this edition broadens its audience. While it remains an excellent guide for AZ 900 preparation, it is equally written for anyone seeking cloud literacy — business decision makers, technical professionals, and learners who want to understand why

Azure works the way it does, not just *what* to select on an exam. The goal is to help you build reasoning skills that last well beyond a single certification.

To me, this is really what a third edition should be. Not just a cosmetic refresh. Not marketing gloss. Not just a new cover and more pages. But a truly genuine evolution.

Moving from teaching nouns to teaching understanding, context, and confidence. If you are new to Azure, this book will ground you. If you already work with the cloud, it will sharpen how you think about it. Not just to pass the exam, but to be able to have foundational conversations with your peers, integrate true business value for your cloud workloads, and be able to think broader than resources and services, but truly understand cloud solutions and how to make them work for you and your business.

Welcome to the next stage of your Azure journey.

Peter De Tender

Microsoft Technical Trainer, Redmond, WA - Microsoft Worldwide Learning (WWL)

Contributors

About the author

Steve Miles works in a senior partner technology enablement role within the cloud practice at a multi-billion-dollar revenue European IT distributor.

He is a Microsoft Most Valuable Professional (MVP), Microsoft Certified Trainer (MCT), and Alibaba Cloud MVP. He has over 25 years of experience in technology, including hosted data center services, hybrid and multi-cloud platforms, as well as a previous military career in engineering, signals, and communications.

Steve is the author and technical reviewer of many books on Microsoft technologies, with a focus on security, Azure, AI, and data.

LinkedIn Profile: `https://www.linkedin.com/in/stevemiles70`

I want to thank the people who have been close to me and supported me, professionally and at home, especially my wife, Pippa, aka Mrs. Smiles, and my family.

About the reviewers

Shabaz Darr is a Senior IT Cloud Consultant, Microsoft MVP, and Microsoft Certified Trainer, with deep expertise in Azure IaaS, AVD, security, and Nerdio. With extensive experience in designing and delivering enterprise cloud solutions, he is recognized for his practical, real-world approach. As the creator of the I Am IT Geek brand, he produces community-focused technical content and speaks regularly at industry events. Shabaz is also the author of multiple books covering Microsoft cloud technologies.

I acknowledge the incredible support of the IT community, whose engagement drives my passion for sharing knowledge. I am grateful to everyone who has supported the I Am IT Geek brand, my books, and my journey as an MVP and MCT. Your encouragement continues to inspire me to produce clear, practical, and community-focused content.

Table of Contents

Preface

Azure is Microsoft's cloud computing platform, providing organizations with on-demand access to compute, storage, networking, data, and application services at a global scale. At its core, Azure enables businesses to build, deploy, and manage workloads without the constraints of traditional on-premises infrastructure.

As Azure has matured, the focus has shifted from simply provisioning resources to operating them responsibly and intelligently. Cost management, security posture, resilience, automation, operational efficiency, and governance are no longer optional considerations—they are essential competencies. Understanding shared responsibility, cloud service models, and Azure's global architecture is critical not only for exam success but for effective day-to-day Azure usage.

While the technology landscape continues to evolve rapidly, the skills measured for Azure fundamentals remain centered on how Azure delivers these capabilities reliably, securely, and cost-effectively. This third edition is written with those skills measured firmly in mind and aligned directly to the current certification objectives.

Whether you are new to cloud computing or refreshing your knowledge for certification, this book is designed to prepare you for the exam and help you build durable Azure fundamentals that translate into confident, effective skills with the platform long after the exam is complete.

Who this book is for

This book is for those in commercial, operational, or technical roles looking to pass the Microsoft certification exam *AZ-900: Microsoft Azure Fundamentals*. The exam is intended for candidates who are just beginning to work with cloud-based solutions and services or are new to Azure.

Importantly, though, this book does not stop at the skills measured. While every chapter is grounded in the official exam objectives, this book also aims to go beyond those objectives. By the end of the book, you will have an additional depth of knowledge that extends beyond the skills measured, offering practical context and insight that is of direct value in a day-to-day Azure-focused role.

You can learn more about the certification and exam at `https://learn.microsoft.com/en-us/credentials/certifications/azure-fundamentals/`.

What this book covers

This book is aligned with the revised syllabus of *AZ-900: Microsoft Azure Fundamentals* exam and encompasses the following topics:

Chapter 1, *Introduction to Cloud Computing*, introduces what cloud computing is, where it has evolved from, why cloud computing emerged, and the target audience. It also takes a look at the delivery and service models of cloud computing, how they compare, and the shared responsibility model.

Chapter 2, *The Benefits of Using Cloud Services*, discusses cloud computing as a digital transformation enabler and the cloud computing mindset, as well as the cloud computing operations model and the economics of cloud computing.

Chapter 3, *Azure Core Architectural Components*, outlines the Azure global infrastructure that comprises the Azure regions, geographies, and availability components. In addition, it covers Azure resource management topics.

Chapter 4, *Azure Compute Services*, focuses on the Azure platform's core building block compute resources: virtual machines, containers, application hosting, serverless, and Azure Virtual Desktop.

Chapter 5, *Azure Storage Services*, covers the core Azure storage offerings, including storage accounts, redundancy options, access tiers, and the different services used for file, object, and disk storage.

Chapter 6, *Azure Network Services*, introduces Azure networking fundamentals, including virtual networks and subnets, public and private connectivity, hybrid networking options, traffic security, and name resolution in Azure.

Chapter 7, *Azure Identity and Access*, explains the concepts of authentication and authorization and identity and access management (IAM), as well as exploring Microsoft Entra ID.

Chapter 8, *Azure Security*, discusses the threat landscape and explains the principles of security posture, Zero Trust, and defense-in-depth (DiD); in addition, it explores cloud posture management and workload protection with Microsoft Defender for Cloud and Microsoft Sentinel security tooling.

Chapter 9, *Azure Cost Management*, explores cost management and the factors that influence costs, as well as explaining the Azure Pricing Calculator.

Chapter 10, *Azure Governance and Compliance*, explores Microsoft's Purview in Azure, Azure Policy, resource locks, and tags, as well as Microsoft's core security, privacy, and security tenets.

Chapter 11, *Azure Resource Deployment and Management*, introduces the Azure portal, as well as Azure PowerShell, the Azure CLI, and Azure Cloud Shell. It also explains Azure Resource Manager templates and Azure Arc.

Chapter 12, *Azure Monitoring Tools*, explores actionable insights into Azure environments using tooling such as Azure Advisor, Azure Monitor, and Azure Service Health.

Appendix, *Assessing AZ-900 Exam Skills*, focuses on the Skills Measured by the certification and potential topics that the exam may cover. As you become competent in each skill area, this can help in tracking progress.

To get the most out of this book

This book's content is intended for candidates who are just beginning to work with cloud-based solutions and services or are new to Azure; no specific knowledge is assumed or required.

To carry out any tasks to further your learning using the Microsoft Azure cloud platform, you will require the following:

- Access to an internet browser.
- A Microsoft account. If you do not have a Microsoft account, you can create a free account using this link: `https://account.microsoft.com/account`.
- An Azure subscription that has permission to create and delete resources in the subscription. If you do not have an Azure subscription, you can create a free Azure account using this link: `https://azure.microsoft.com/free`.

Download the color images

We also provide a PDF file that has color images of the screenshots/diagrams used in this book. You can download it from here: `https://packt.link/gbp/978-1-80670-241-1`.

Conventions used

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

`CodeInText`: Indicates code words in text, database table names, folder names, filenames, file extensions, pathnames, dummy URLs, user input, and X (formerly, Twitter) handles. For example: "`Family` represents the VM family series."

Any command-line input or output is written as follows:

```
$PSVersionTable.PSVersion
```

Bold: Indicates a new term, an important word, or words that you see on the screen. For instance, words in menus or dialog boxes appear in the text like this. For example: "**Azure compute services** is one of the core Azure service categories."

Note

Warnings or important notes appear like this.

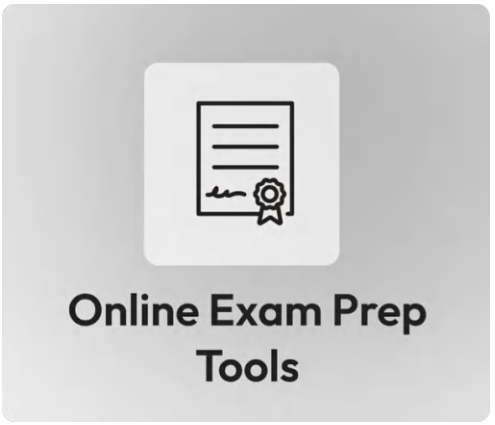
Tip

Tips and tricks appear like this.

Free Benefits with Your Book

This book includes free benefits designed to support your learning and help you ace your certification exam. Activate them now for instant access (see the "*How to Unlock*" section for instructions).

Here's a quick overview of what you can instantly unlock with your purchase:

PDF and ePub Copies	**Online Exam Prep Tools**
	
Access a DRM-free PDF copy of this book to read anywhere, on any device. Use a DRM-free ePub version with your favorite e-reader.	**Practice with purpose**: Reinforce your learning with online resources thoughtfully designed to reflect real exam standards and objectives. **Certified expertise**: Each resource is created and reviewed by professionals certified in the exact exam you're preparing for. **Exam ready from day one**: Begin practicing as you learn from this book so exam day feels familiar, not surprising.

You also get access this book on **Packt's next-gen Reader,** a web-based reader designed to help you seamlessly learn across devices: Sync your progress, highlight key ideas, take notes,

and bookmark important sections anywhere, anytime. It also includes dark and sepia modes for more comfortable reading.

How To Unlock

Unlock Exam Prep Tools and the PDF Copy

Scan the QR code (or go to `packtpub.com/unlock`). Search for this book by name, confirm the edition, and then follow the steps on the page.

Note: Keep your invoice handy. Purchases made directly from Packt don't require an invoice.

Note: Access to the online platform is reserved for the first owner of the publication and is non-transferable. Packt reserves the right to restrict and withdraw access to online resources to third parties.

Get in touch

Feedback from our readers is always welcome.

General feedback: If you have questions about any aspect of this book or have any general feedback, please email us at `customercare@packt.com` and mention the book's title in the subject of your message.

Errata: Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you have found a mistake in this book, we would be grateful if you reported this to us. Please visit `http://www.packt.com/submit-errata`, click **Submit Errata**, and fill in the form.

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

If you are interested in becoming an author: If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, please visit .

Stay Sharp in Cloud and DevOps – Join 44,000+ Subscribers of CloudPro

CloudPro is a weekly newsletter for cloud professionals who want to stay current on the fast-evolving world of cloud computing, DevOps, and infrastructure engineering.

Every issue delivers focused, high-signal content on topics like:

- AWS, GCP & multi-cloud architecture
- Containers, Kubernetes & orchestration
- **Infrastructure as Code** (**IaC**) with Terraform, Pulumi, etc.
- Platform engineering & automation workflows
- Observability, performance tuning, and reliability best practices

Whether you're a cloud engineer, SRE, DevOps practitioner, or platform lead, CloudPro helps you stay on top of what matters, without the noise.

Scan the QR code to join for free and get weekly insights straight to your inbox:

Share your thoughts

Once you've read *Microsoft Azure Fundamentals Certification and Beyond*, we'd love to hear your thoughts! Scan the QR code below to go straight to the Amazon review page for this book and share your feedback.

`https://packt.link/r/180670241X`

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

Part 1

Cloud Concepts

This section provides coverage of the knowledge and skills required for the *Skills Measured* of the *Describe Cloud Concepts* section of the exam. We also cover knowledge and skills that go beyond the exam content, so you are prepared for a real-world, day-to-day Azure-focused role.

This part of the book includes the following chapters:

- *Chapter 1, Introduction to Cloud Computing*
- *Chapter 2, The Benefits of Cloud Computing*

1

Introduction to Cloud Computing

We are firmly in the AI and cloud era, with **Microsoft Azure** established as one of the world's leading cloud computing platforms. Understanding how cloud computing works—and how it enables modern digital and AI-driven solutions—is now essential for both IT professionals and business decision-makers.

Modern AI solutions depend on elastic compute, rapid experimentation, and governed access to data. Cloud computing provides the operating model that makes these capabilities practical at scale, enabling organizations to provision resources on demand, scale dynamically, and apply consistent controls without directly managing underlying infrastructure.

This book provides a comprehensive introduction to cloud computing concepts and the Microsoft Azure platform. By the end of the book, you will have a solid understanding of cloud principles and Azure fundamentals, equipping you to approach the *AZ-900 Azure Fundamentals* certification exam with confidence while also building a strong foundation for real-world, day-to-day Azure-focused roles.

This chapter primarily focuses on the *Describe cloud concepts* module from the *Skills Measured* section of the *AZ-900 Azure Fundamentals* exam.

At this level, the exam assesses your understanding of core cloud computing ideas and terminology, rather than hands-on implementation or configuration.

> **Note**
>
> A full mapping of *AZ-900 Skills Measured* is available in the *Appendix*, *Assessing AZ-900 Exam Skills*.

By the end of this chapter, you will be able to confidently answer questions on the following topics:

- What is cloud computing?
- How has cloud computing evolved?
- Why do organizations adopt cloud computing?
- Who uses cloud computing?
- What is the cloud computing hierarchy of needs?
- What are the cloud computing delivery models?
- What are the cloud computing service types?
- How do the cloud computing service types compare?
- What is the shared responsibility model?

This chapter introduces cloud computing by exploring its origins, evolution, and core architectural models. It examines why organizations adopt cloud platforms, who cloud computing is designed for, and how responsibility is shared between cloud providers and customers. While the content is aligned to the AZ-900 exam, it is also intended to prepare you for practical, day-to-day Azure-focused roles.

Note

Free Benefits with Your Book

Your purchase includes a **free PDF copy of this book** along with **access to online cert prep platform** designed to help you ace your AZ-900 certification. Check the *Free Benefits with Your Book* section in the *Preface* to unlock them instantly and maximize your learning experience.

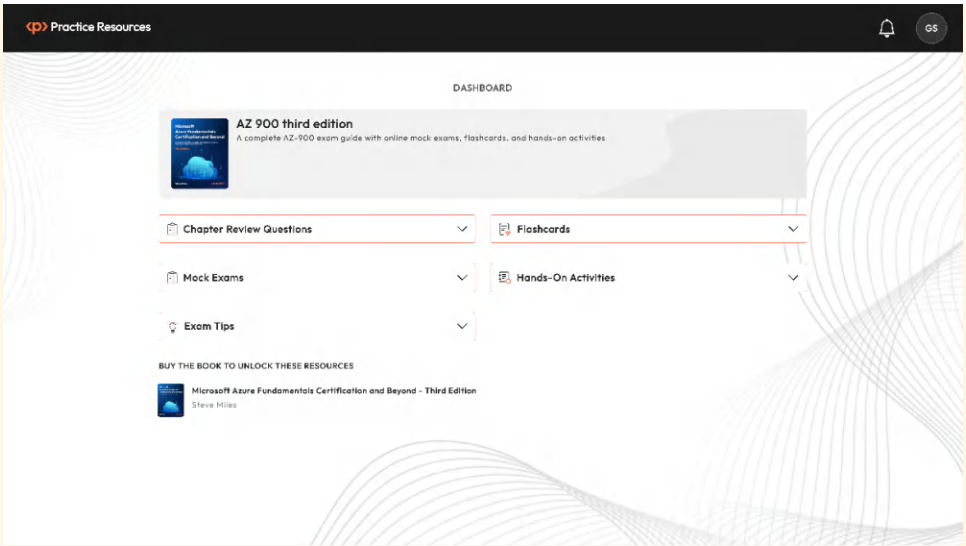


Figure 1.1: Dashboard interface of the online practice resources

Here's what you'll find on the online platform:

- 2 Mock Exams (40 questions each)
- 57 Flashcards
- 20 Exam Tips
- 10 Hands-On Activities
- 124 Chapter Review Questions

What is cloud computing?

Cloud computing is something of a misnomer. The term does not describe a specific technology, but rather a choice of computing model for delivering and consuming IT resources and services in an adaptive manner. In many ways, "the cloud" is as much a marketing term as it is a technical one. The primary value of a cloud platform lies in its ability to enable digital transformation and innovation.

Cloud platforms support faster time-to-market, improved agility, and economies of scale through a consumption-based cost model; these characteristics allow organizations to choose

how computing resources are delivered and consumed in a way that best aligns with their operating model and business goals.

Cloud computing represents not only a technical evolution but also a financial one. The cost model shifts from **Capital Expenditure** (**CapEx**), where hardware is purchased upfront, to **Operating Expenditure** (**OpEx**), where organizations pay only for the resources they consume.

Private cloud deployments may include both CapEx and OpEx elements; however, for the purposes of the exam objectives, private cloud is primarily associated with CapEx, as it typically involves owning and operating on-premises infrastructure; in real-world scenarios, *leased* hardware and software may also be treated as OpEx, although this still involves building and managing infrastructure outside of the public cloud.

Today's largest cloud platform providers include Microsoft, Amazon, Google, and Alibaba. These public cloud platforms operate on a multi-tenant model, where computing resources such as compute, storage, and networking are hosted in provider-managed data centers and shared securely across customers. Organizations consume only the resources they use and are billed accordingly, benefiting from the economies of scale achieved by these hyperscale cloud operations.

It is helpful to think of cloud computing as an **operating model** rather than a **physical location**. The defining characteristics of the cloud are how resources are provisioned, governed, and consumed—such as self-service delivery, elasticity, and metered usage—not simply where the infrastructure resides. This is why cloud capabilities can extend into hybrid and edge environments while still being managed through consistent control and operational practices.

Many providers also support **hybrid cloud models**, where cloud services extend beyond centralized data centers into customer or third-party locations; the hardware, software, services, and resources can now sit in these locations, with a **control plane** that operates over a network from the cloud provider's platform locations. Microsoft's examples of this are **Azure Local** and **Azure Arc**.

In the next sections, you will start with the official NIST definition of cloud computing, then look at how cloud computing has developed over time. You will also be introduced to the main ways cloud services are delivered, and the different types of cloud services organizations use today.

The NIST definition of cloud computing

The US government agency **National Institute of Standards and Technology** (**NIST**) defines "cloud computing" as follows:

> *Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model is composed of five essential characteristics, three service models, and three deployment models.*

The following are the five essential characteristics of the cloud computing model:

- **On-demand self-service**: Users can provision computing resources, such as virtual machines or storage, without requiring manual interaction with the service provider
- **Broad network access**: Cloud services are accessible over the network using standard mechanisms, such as the internet, from a wide range of devices
- **Resource pooling**: The cloud provider's resources are shared across multiple customers, with resources dynamically assigned and reassigned based on demand
- **Rapid elasticity**: Resources can be scaled up or down quickly, often automatically, to match changing workload requirements
- **Measured service**: Resource usage is monitored, measured, and billed based on actual consumption

The three service models are listed as follows:

- **Software as a Service (SaaS)**: The provider delivers a complete application that users access over the internet, while the provider manages the underlying infrastructure and platform
- **Platform as a Service (PaaS)**: The provider supplies the platform and runtime environment used to build and run applications, while customers manage their application code and data
- **Infrastructure as a Service (IaaS)**: The provider delivers core infrastructure resources such as virtual machines, storage, and networking, while customers manage the operating system and above

Note

The NIST definition groups cloud services into three broad service models: IaaS, PaaS, and SaaS. Newer terms, such as **Function as a Service (FaaS)**, are typically considered part of the PaaS model and emerged after the NIST definition was published, which is why they are not listed separately here.

The three deployment models listed include the following:

- **Private cloud**: Cloud infrastructure that is dedicated to a single organization, providing greater control over resources and data
- **Public cloud**: Cloud services that are owned and operated by a cloud provider and shared securely across multiple customers
- **Hybrid cloud**: A model that combines private and public cloud environments, allowing data and applications to be shared between them

Note

You can find more information about NIST at `https://csrc.nist.gov/pubs/ sp/800/145/final` and `https://nvlpubs.nist.gov/nistpubs/Legacy/ SP/nistspecialpublication800-145.pdf`.

Building upon this knowledge of cloud computing, we will look at its evolution next.

How has cloud computing evolved?

Historically, organizations began with bare-metal physical hardware deployed on-premises; this evolved into on-premises virtualization, enabling improved hardware utilization and operational efficiency. Over time, these virtualization concepts extended to service-provider-hosted and multi-tenant cloud environments, where infrastructure ownership and management shifted to the hosting services or cloud provider.

Cloud computing represents a major transformation from traditional computing platforms, offering organizations new ways to consume computing resources and services. Rather than owning and maintaining physical infrastructure, organizations increasingly rely on shared, scalable platforms delivered on demand. *Figure 1.2* illustrates this evolution:

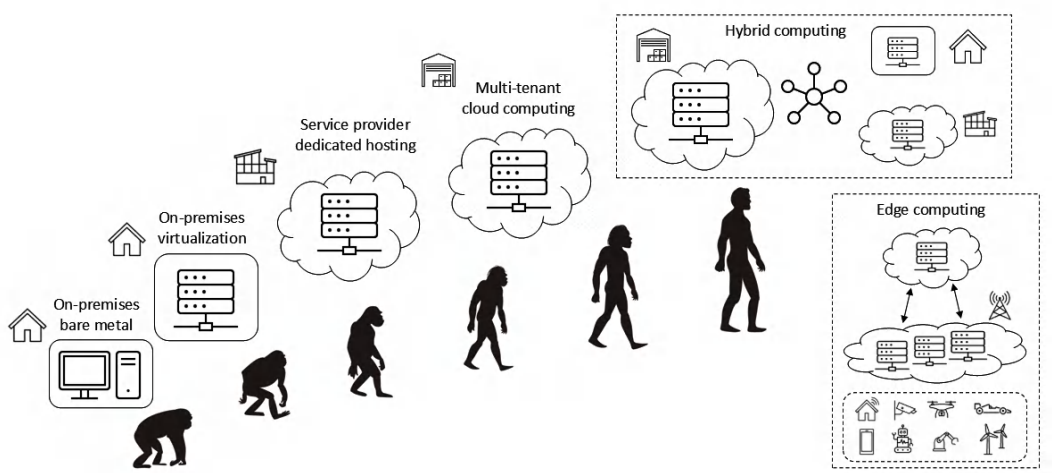


Figure 1.2: Evolution of cloud computing

As cloud adoption matured, compute models diversified, **virtual machines** remained common, while **containers** emerged to provide lighter-weight application packaging and portability. In parallel, **serverless architectures** introduced an even higher level of **abstraction,** allowing organizations to focus primarily on **business logic** (**outcomes**) while the platform automatically handles infrastructure provisioning, scaling, and availability when appropriate.

More recently, hybrid and edge computing models have extended cloud capabilities beyond centralized data centers, enabling workloads to run closer to users, devices, and data sources while remaining connected to centralized cloud services.

When working with different computing models, it is important to understand the key difference between public cloud computing and edge computing. As cloud computing has evolved beyond centralized service providers' data centers, newer models such as edge computing have emerged to support scenarios where processing needs to occur closer to users, devices, or data sources.

The key difference between public cloud computing and edge computing is where the data processing happens. The **public cloud** is based on a *centralized* data collection, processing, and analysis approach. In contrast, **edge computing** uses a *distributed* computing model approach, where the data is collected, processed, and analyzed locally.

This model has benefits in terms of **latency**, but also for those organizations concerned with or mandated by **data locality,** where compliance may place strict controls on where data is stored and processed.

Hybrid computing aims to provide a balance of computing resources and services available anywhere, anytime; it gives businesses options and the *power of choice* as to the most suitable technology platform and data location for any given workload, business initiative, or scenario that needs to be supported.

As computing platform environments have changed over time, so have the architectures. The following section will look at the evolution of **cloud computing architectures**.

Cloud computing architecture models

In this section, you are introduced to the concept of **cloud-native computing** and the shift in architectural mindset from **monolithic** virtual machine stacks to microservices, containers, and serverless functions.

This represents a fundamental shift from a *compute stack–centric* approach to a *business logic–centric* one, where the focus is on outcomes rather than infrastructure inputs; as a result, responsibility for many lower layers—such as runtimes and underlying compute—is shifted to the cloud provider and delivered as a service to consume. You provide the code, and the provider determines how it is executed, scaled, and managed.

The **serverless** computing model emerges from this architectural evolution at the compute layer and represents an extension of the PaaS model. When using PaaS resources to host a website, application, or execute code, servers are still involved, and a defined set of underlying compute resources is used and paid for; in traditional hosting models, this would typically be represented by a server farm.

As the name suggests, in a *serverless model*, you are not responsible for creating or managing **compute resources**; while servers still exist, the compute layer is fully provided and managed by the platform provider and abstracted from your control and operational responsibility. In this model, you supply the **business logic**, and the provider runs it on their **managed compute** infrastructure.

Building on the cloud service models introduced earlier, IaaS abstracts the underlying hardware, PaaS abstracts the compute platform and runtime environment, and serverless architectures extend this abstraction further by removing the need to manage the runtime entirely.

The term *abstract* refers to the removal of responsibility for a given layer; when a layer is abstracted, it becomes the cloud provider's responsibility to provision, scale, maintain availability, and manage that layer, allowing you to focus on higher-level concerns rather than infrastructure operations.

As abstraction increases, control does not disappear—it changes form. Instead of controlling physical infrastructure and individual servers, organizations rely more on configuration,

identity-based access, policy, and monitoring to enforce requirements. Understanding this shift helps explain why higher-level cloud services can reduce operational burden while still demanding strong governance and security practices.

Figure 1.3 illustrates this model of cloud computing architectures:

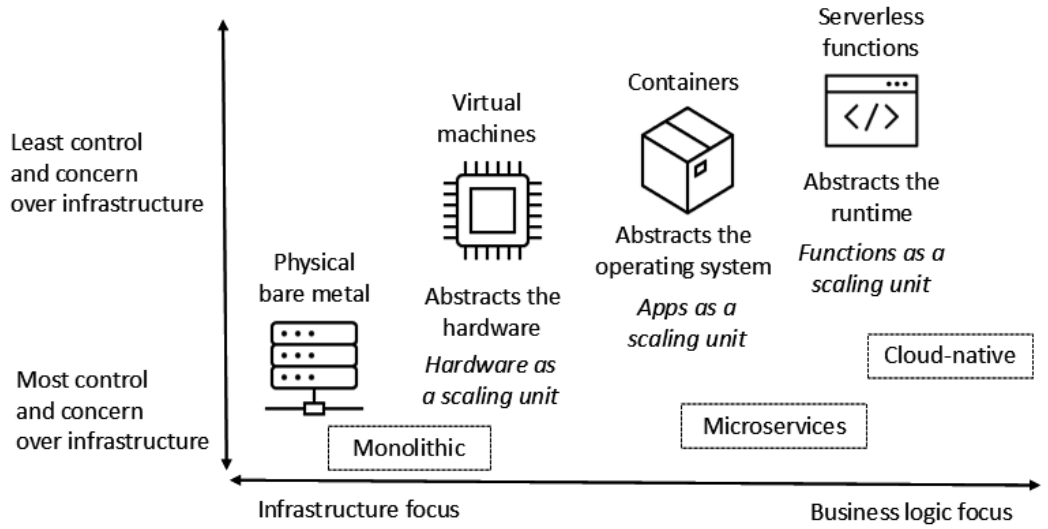


Figure 1.3: Cloud computing architectures

Figure 1.3 outlines where the architectures differ in their characteristics; it also outlines decision criteria to consider so that each architecture can be positioned to allow you to make the most appropriate choice for any given scenario.

This section looked at the evolution of computing platform environments and cloud computing architectures. In the *How do the cloud service types compare?* section later, you will learn about the degrees of control each model offers.

Next, you will learn why you need to adopt cloud computing.

Why do organizations adopt cloud computing?

The adoption of cloud computing is often driven more by the advantages of its business and operational model than by technology or location alone. The cloud computing model provides organizations with greater choice in how computing resources are delivered and consumed, allowing them to better align technology platforms with business operating models.

Cloud-based resources can be deployed in hybrid configurations, meaning workloads may run in cloud provider facilities, customer locations, or third-party environments. In hybrid

scenarios, the cloud platform can provide centralized control and operational capabilities, even when compute resources remain outside the provider's data centers.

Adopting cloud computing does not have to be a binary decision between fully public cloud services and traditional on-premises environments. In many cases, compute resources and data remain on-premises, while the cloud is used to provide the control and operations plane; this approach is often described as managing from the cloud rather than moving entirely to the cloud.

A hybrid model typically consists of a combination of traditional on-premises and cloud-based resources; in this context, *cloud* does not always mean shared public compute resources hosted in a provider's facility and accessed over the network. In this context, *cloud* refers to the operating model and control plane provided by the cloud platform, including centralized management, policy enforcement, automation, and monitoring, regardless of where the underlying compute resources are physically located or run.

The core benefits of the cloud computing model compared to traditional computing approaches are illustrated in *Figure 1.4*:

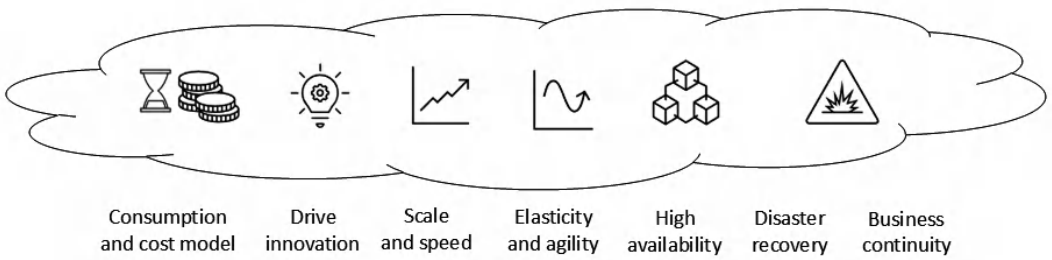


Figure 1.4: Benefits of cloud computing

Additional operational advantages include the removal of responsibility for replacing failed hardware or managing physical security in public cloud environments, as these concerns are handled by the cloud provider. This eliminates the need for CapEx to purchase hardware in public cloud scenarios; CapEx may still apply in private cloud deployments, where organizations own or control the underlying hardware and facilities, including scenarios where infrastructure is hosted by a third-party provider.

Despite these advantages, organizations remain responsible for securing and governing their systems and data in public cloud environments; this includes ensuring availability and redundancy, protecting identities, governing access permissions, and managing updates in line with shared responsibility principles.

However, the same speed and flexibility that make the cloud attractive also increase the importance of discipline. When resources can be created quickly, environments can grow

rapidly and become difficult to manage without clear ownership, consistent standards, and visibility into cost and security. This is one reason the shared responsibility model is so important—it clarifies what the provider handles and what the customer must still govern.

These advantages, along with the shared responsibility model, help explain why cloud computing appeals to a wide range of roles across an organization. Because responsibilities for infrastructure, platforms, and services are divided between the provider and the customer, different users benefit in different ways depending on their focus and responsibilities. The next section looks at who cloud computing is designed for and how various roles benefit from its adoption.

Who uses cloud computing?

Cloud computing can mean different things to different people, depending on their role, responsibilities, and perspective within an organization. A business leader may focus on cost optimization, agility, and time to market, while a technology leader may prioritize scalability, governance, and reliability. Developers, database administrators, and data scientists, on the other hand, each have their own distinct requirements, meaning cloud computing must address a broad and diverse set of needs.

In Azure, cloud services are commonly categorized at a high level to align with these different audiences and business functions. These categories include the following:

- Hosted infrastructure platform
- Application development platform
- Data, AI, and analytics platform
- Monitoring and management platform
- Hybrid and multi-cloud control plane platform
- Security operations platform
- Identity and compliance platform

These capabilities are designed to support a wide range of organizational roles, making the platform relevant to business, operational, and technical stakeholders alike.

For example, a business leader may use cloud services to improve agility and reduce upfront costs, focusing on faster delivery and predictable spending. An IT operations team may prioritize scalability, availability, and centralized management, while developers and data teams use cloud platforms to build, deploy, and analyze applications more quickly without managing underlying infrastructure. These differing needs can all be met using the same cloud platform, with each role interacting with it at a different level.

Azure also places strong emphasis on hybrid and multi-cloud scenarios by combining public cloud services with private and on-premises environments. Through capabilities such as Azure Local and Azure Arc, organizations can centrally manage and govern resources across Azure, on-premises infrastructure, and other cloud providers.

Understanding who cloud computing is designed for is an important part of successful adoption; beyond technology and processes, organizations must consider the people involved, including decision-makers, sponsors, and practitioners. Recognizing these stakeholders and the capabilities available to them helps ensure cloud computing initiatives deliver value across the entire organization.

Because different roles rely on different capabilities, it is important to understand how cloud services build on one another. The next section introduces the cloud computing hierarchy of needs, which explains how foundational services must be in place before higher-level business outcomes can be achieved.

What is the cloud computing hierarchy of needs?

The IT services delivery model can draw parallels to **Maslow's hierarchy of needs**, a psychological theory commonly represented as a hierarchical pyramid. In this model, lower-level needs must be satisfied before higher-level needs can be achieved, creating clear dependencies between layers.

In Maslow's theory, changes or failures in the lower levels directly affect the ability to achieve higher-level outcomes, with the ultimate goal being self-actualization. This concept can be applied to IT services delivery, where business outcomes such as applications, code, or functions depend on a series of underlying technology layers.

This theory can be applied to the IT services delivery model. In this model, the typical technology stack provides the outcome that needs to be met, such as that an app, code, or function needs the following:

- To exist on a lower layer of software
- To live on a lower layer of compute
- To exist on a lower layer of hardware
- To exist in a physical facility
- Power, cooling, physical security, facilities management, staffing, and so on

As with Maslow's hierarchy, higher-level outcomes depend on lower-level layers being fulfilled and maintained.

In traditional computing environments, a significant portion of the budget and effort is consumed by maintaining these lower layers. Resources are often concentrated on procuring

hardware and operating data centers, leaving less time and budget available for innovation and value-driven activities at the top of the stack.

This results in a CapEx-constrained model, where technology is frequently viewed as a *cost center* rather than an *enabler of innovation*.

At this point, the challenge is not understanding the individual layers, but seeing how effort and investment are concentrated at the bottom of the stack while higher-level outcomes depend on them. A visual comparison helps make this imbalance clear, which is why the following *Figure 1.5* contrasts Maslow's hierarchy of needs with a cloud computing hierarchy of needs.

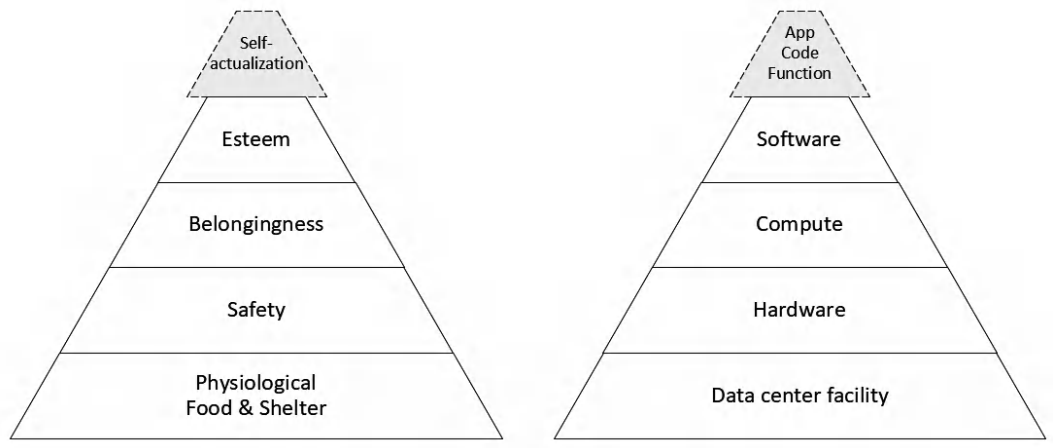


Figure 1.5: Maslow's hierarchy versus the cloud computing hierarchy of needs

The cloud computing model aims to rebalance this distribution of responsibility and investment. Cloud service providers assume responsibility for the lower layers, allowing organizations to focus their time, budget, and skills on higher-value activities such as application development, innovation, and business outcomes.

The goal is to adopt a more flexible and agile operating model, consuming only what is required at each layer to deliver the desired business outcome, rather than owning and managing the entire stack.

Understanding the cloud computing hierarchy of needs helps explain why different delivery models exist. As organizations move up the hierarchy—from foundational infrastructure to higher-level business outcomes—the way those layers are delivered and managed also changes. The next section explores the cloud computing delivery models and how each model aligns with different layers of the hierarchy.

What are the cloud computing delivery models?

Cloud computing generally has three delivery models: public, private, and hybrid. They are detailed as follows:

- **Public cloud**: This is a **shared entity** (*multi-tenant*) computing model. Hardware and resources such as compute, storage, and networking are owned and operated by the cloud provider and shared securely with other tenants on the platform. Customers typically have limited control over the underlying infrastructure and pay only for the resources they consume.
  This model can be compared to living in an apartment building, where multiple tenants share the same structure and facilities while renting individual units. In this analogy, the cloud provider acts as the landlord.
- **Private cloud**: This is a **dedicated entity** (*single-tenant*) computing model. Hardware and resources such as compute, storage, and networking are dedicated exclusively to a single organization, which retains full control over the environment.
  This model is comparable to living in a private house, where the building and resources are not shared with other tenants. A private cloud may consist of hardware owned and operated by the organization in its own facility, hosted by a third-party provider, located in a colocation data center, or delivered as dedicated infrastructure provided by a service provider.
- **Hybrid cloud**: This is a combination of the **shared entity** (*multi-tenant*) and **dedicated entity** (*single-tenant*) computing models. Organizations choose to run some workloads in a private cloud and others in a public cloud based on requirements such as control, compliance, performance, or scalability.
  This approach provides flexibility and agility by allowing organizations to place workloads in the most appropriate environment while adapting to changing business and technical demands.s

Figure 1.6 illustrates the key characteristics of the public, private, and hybrid cloud delivery models.

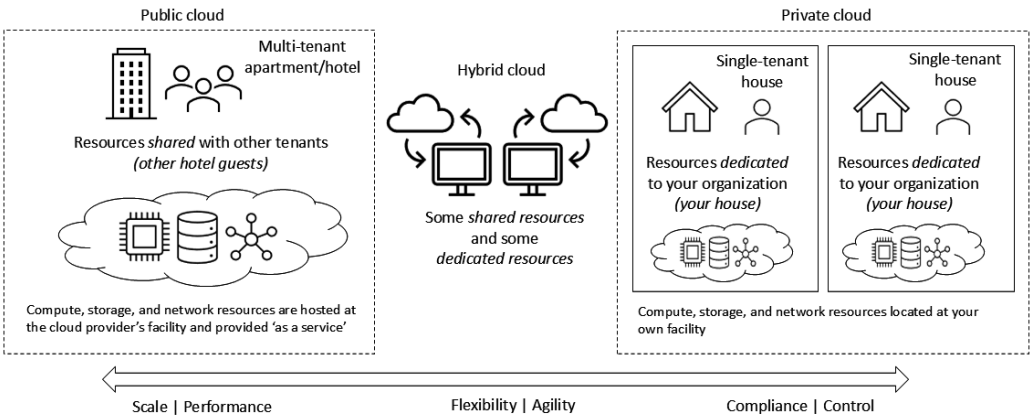


Figure 1.6: Cloud computing delivery models

Now that the core delivery models have been introduced, the next step is understanding how they differ in practice. Each model offers a different balance of control, flexibility, and responsibility, and those differences influence which model is best suited to a particular workload or scenario. The next section compares these delivery models to help clarify how those trade-offs affect decision-making.

Comparing the cloud computing delivery models

We will now examine the characteristics of each model in more detail to support informed decisions when choosing one model over another.

Each delivery model has distinct characteristics. The most appropriate choice depends on how much control, security, and management responsibility you require over resources, such as applications, code, data, networking, and security controls. A key consideration is whether resources are shared with other organizations or dedicated exclusively for your use.

The terms *multi-tenant* and *single-tenant* are commonly used to distinguish between models that share resources and those that provide dedicated resources to a single organization.

A helpful way to visualize this distinction is through the analogy of a hotel compared to a house. In a house, you have dedicated access to all resources, such as the front door, internal spaces, and utilities, with no sharing required. In contrast, a hotel provides a private room for your exclusive use, while common facilities such as entrances, stairways, restaurants, and services are shared with other guests.

Figure 1.7 illustrates this analogy and highlights how public, private, and hybrid cloud delivery models differ in terms of control, resource sharing, scale, and compliance considerations.

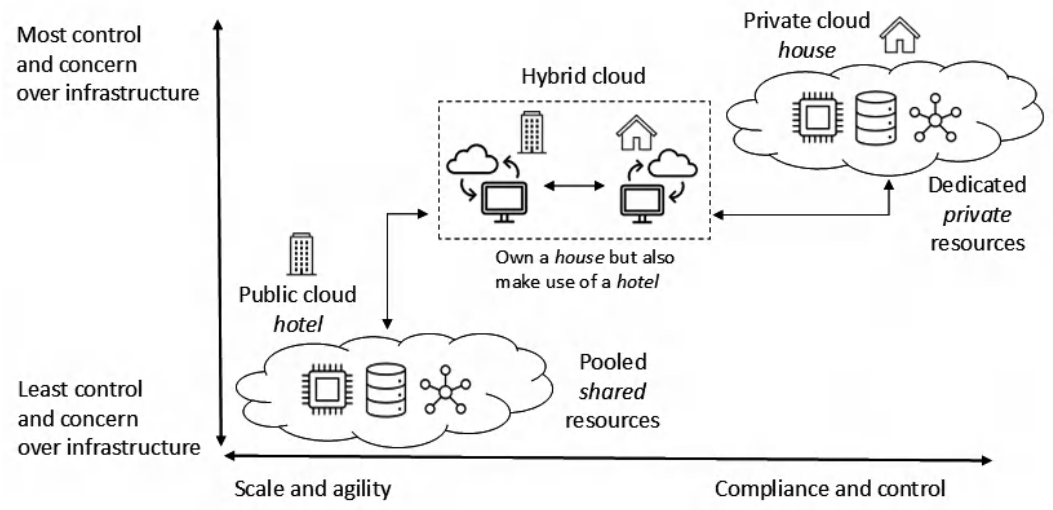


Figure 1.7: Comparing cloud computing delivery models

Understanding the characteristics of each delivery model is important because they directly affect day-to-day decisions such as cost control, governance, scalability, and compliance. These differences also form the basis of how cloud delivery models are tested in the AZ-900 exam, which is why each model is examined in more detail in the following sections.

Characteristics of public cloud computing resources

To recap, a public cloud is a shared entity (multi-tenant) computing model in which infrastructure and services are owned and operated by a cloud provider and securely shared across customers.

The key characteristics of public cloud computing resources include the following:

- **Consumption-based pricing** using metered, pay-as-you-go billing, where you pay only for the resources you consume, enabling effective cost control
- **Highly scalable resource availability**, allowing organizations to rapidly increase or decrease capacity to meet demand
- **Strong performance, scalability, and elasticity**, with automated, on-demand provisioning and deprovisioning of compute resources
- **Built-in availability and resilience**, including reliability, fault tolerance, and redundancy, provided by the cloud platform

- **Global access to resources**, typically over the internet or through private connectivity options such as **ExpressRoute**
- **Self-service management**, commonly delivered through web portals, command-line interfaces, and APIs
- **Reduced control over infrastructure security and compliance**, as customers do not have full control over the underlying physical environment in a public cloud model
- **Cloud-native identity and access management**, commonly provided through **Microsoft Entra ID**, with support for integrating traditional on-premises identity systems when required
- **Absence of direct deployment of customer-owned physical hardware**, as public cloud platforms deliver virtualized compute resources; in some cases, providers may offer dedicated hardware options
- **Potential reduction or removal of on-premises data center infrastructure**, allowing organizations to decommission local facilities
- **An OpEx cost model**, with no CapEx required for purchasing hardware or compute infrastructure

The following are some examples of public cloud platforms:

- Microsoft Azure
- Amazon Web Services
- Alibaba Cloud
- Google Cloud Platform

Because of these characteristics, public cloud platforms are commonly used by organizations that need to scale quickly, control costs, and avoid managing physical infrastructure.

They are well-suited to start-ups, digital-first businesses, development and test environments, and workloads with variable or unpredictable demand.

For many organizations, public cloud also provides a practical entry point into cloud computing by reducing upfront investment and operational complexity.

Characteristics of private cloud computing resources

To recap, a private cloud is a dedicated entity (single-tenant) computing model in which infrastructure and services are reserved exclusively for a single organization. The key characteristics of private cloud computing resources include the following:

- **Dedicated infrastructure**, where computing resources are deployed either on-premises at the organization's facility or hosted by a third-party provider. Available capacity is limited to the hardware provisioned.
- **A primarily CapEx cost model**, as organizations typically purchase and own the hardware, although leasing models may also be used.
- **Exclusive use of physical and virtualized hardware**, meaning servers, storage, and networking resources are not shared with other organizations. The organization is responsible for supporting and replacing failed hardware.
- **Full responsibility for availability and operations**, including fault tolerance, scalability, security, protection, patching, maintenance, and ongoing support.
- **Potential decommissioning of on-premises facilities** when private cloud infrastructure is hosted by a third-party provider, removing the need to operate local data centers.
- **Access over local or private networks**, typically with internet connectivity. In some scenarios, such as remote locations, regulated environments, or high-security facilities, private cloud resources may operate with limited, intermittent, or no internet connectivity. Internet access is not a defining characteristic of a private cloud.
- **Self-service management capabilities**, similar to those offered by public cloud platforms, while retaining full control over security, governance, and compliance. The organization remains responsible for procuring, implementing, and managing the underlying infrastructure.
- **Complete control over hardware, physical resources, security, and compliance**, making private cloud suitable for workloads with strict regulatory or operational requirements.
- **Identity and access management** is commonly provided through traditional Windows Server Active Directory, with the option to integrate **Microsoft Entra ID** when connecting to public cloud services.
- **Support for physical server deployment**, as private cloud environments can include both physical and virtualized infrastructure.

The following are examples of private cloud platforms:

- Azure Local
- Red Hat OpenShift
- VMware Cloud Foundation

Because of these characteristics, private cloud environments are commonly used by organizations with strict regulatory, security, or operational requirements, such as financial services, government, healthcare, and large enterprises with existing data center investments.

Private cloud is well-suited to workloads that require full control over infrastructure, data locality, or limited internet connectivity, while still benefiting from cloud-style management and automation.

Characteristics of hybrid cloud computing resources

To recap, a **hybrid cloud** combines the shared entity (multi-tenant) computing model and a dedicated entity (single-tenant) computing model. The key characteristics of hybrid cloud computing resources include the following:

- **High flexibility in resource placement**, allowing organizations to choose the most appropriate location and level of control for each workload.
- **A combination of public and private environments**, where some resources run on a public cloud platform, and others run on-premises or in a hosted private cloud environment. These environments are typically connected using the internet or private managed networks such as ExpressRoute.
- **Support for cloud bursting**, enabling workloads to temporarily extend capacity into the public cloud during periods of increased demand.
- **Clear responsibility boundaries**, where private cloud hardware (physical servers, virtualization platforms, and related infrastructure) is owned and maintained by the organization, while public cloud hardware is owned and supported by the cloud provider.
- **Flexible access options**, allowing computing resources to be accessed over the internet or private networks, depending on security and performance requirements.
- **Integrated connectivity between private and public clouds**, meaning private cloud resources are not isolated and can securely interact with public cloud services as part of a unified hybrid environment.
- **Greater flexibility in security, protection, and compliance controls**, enabling organizations to apply different governance models to different workloads based on risk and regulatory needs.

- **Hybrid identity integration**, where traditional Windows Server Active Directory commonly provides identity and authentication for private cloud resources, while Microsoft Entra ID is used for public cloud access. Directory synchronization enables a consistent or single sign-on experience across environments.
- **Physical server deployment within private cloud environments**, while public cloud resources are consumed as provider-owned infrastructure that cannot be physically owned by the customer.
- **Flexible expenditure models**, allowing organizations to combine CapEx for private cloud resources with OpEx for public cloud consumption.

Hybrid cloud platforms can be implemented using solutions such as Azure Local and Azure Arc. Common scenarios include managing on-premises virtual machines from Azure, replicating or backing them up to Azure, or running services such as **Azure Virtual Desktop** or **Azure Kubernetes Service** in a hybrid configuration.

You learned about the different cloud computing delivery models, how they compare, and the characteristics of each. We will take the same approach to looking at the cloud computing service models in the next section.

What are the cloud computing service types?

Cloud computing is generally considered to have four **service models**; these are also referred to as **categories**. These service models are as follows:

- IaaS
- PaaS
- FaaS, often referred to as *serverless*
- SaaS

Figure 1.8 shows the relationship between these service models:

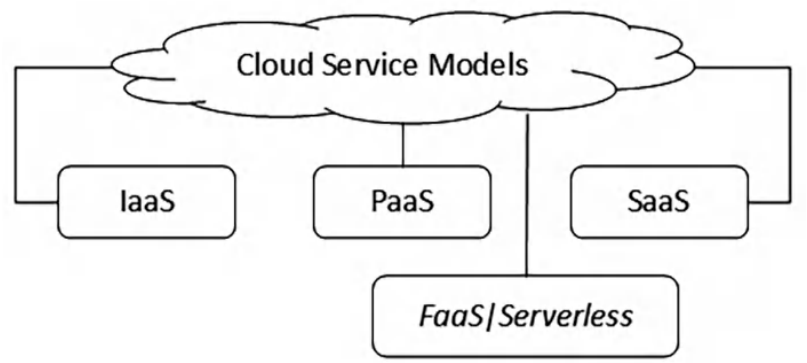


Figure 1.8: Cloud computing service models

Every cloud computing resource will fit into one of these categories; a solution will comprise one or more resources from each of these categories, and you select the resources from each category based on each solution's needs, taking a mix-and-match approach to tailoring a set of technology resources to map to business needs.

In the following section, you will cover quite a lot of ground, as there are many concepts and aspects to present information on, not only for the exam but for knowledge beyond. You will look closely at each service model and compare and contrast the characteristics in more detail.

A closer look at the cloud computing service models

Cloud computing is all about **abstraction**. This abstraction model approach removes layers that you no longer need to care about; the layer still exists, but it is being handled by somebody else, and it frees up resources to concentrate on other, more valuable layers.

The service models or categories of cloud computing define what layer of access and control the cloud service provider is responsible for and what the consumer of the cloud resources is responsible for. This is illustrated in *Figure 1.9*.

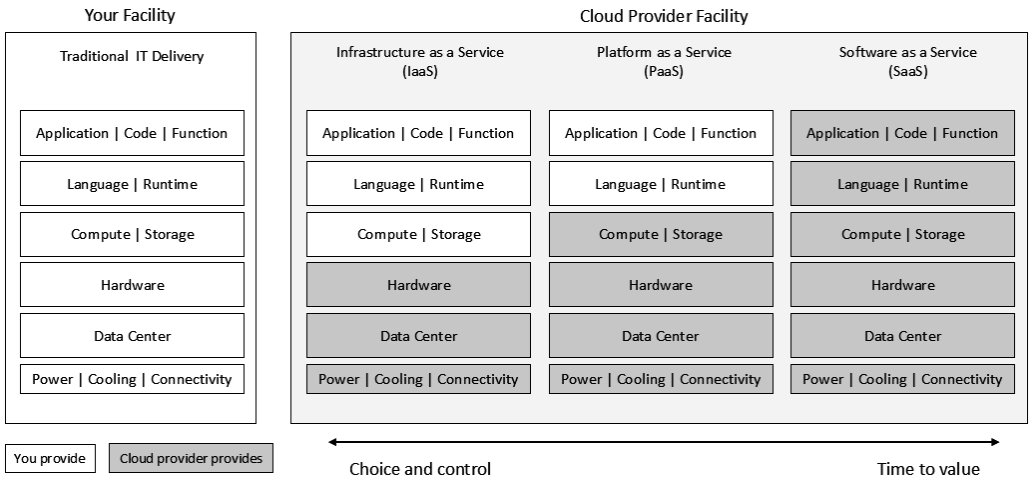


Figure 1.9: Cloud computing service models: layer providers

Figure 1.9 shows how responsibility and control shift across the service models as more layers are managed by the cloud provider. To make this shift easier to understand, the same concept is illustrated using a familiar, real-world analogy.

Figure 1.9 is translated into the analogy of pizza processing (*Figure 1.10*):

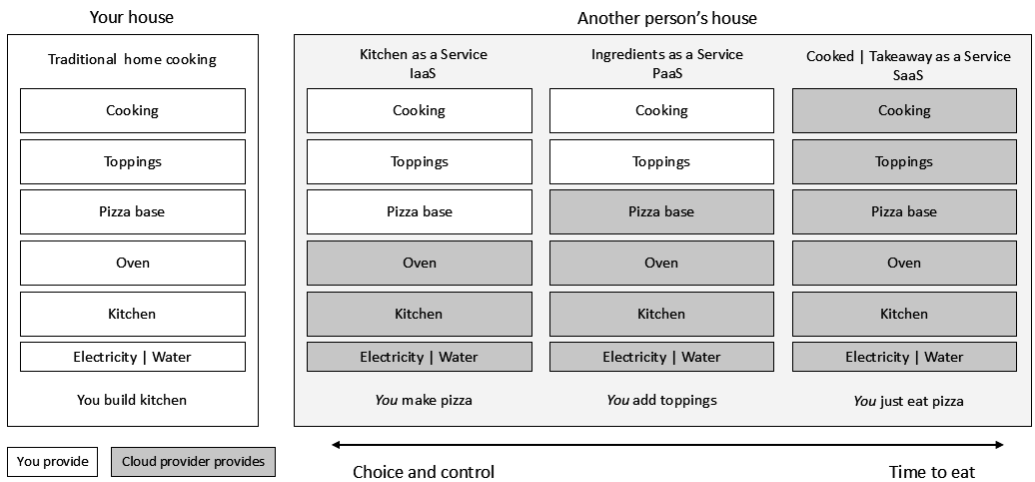


Figure 1.10: Cloud computing service models: layer analogy of pizza processing

As you can see in *Figure 1.10*, the service model you go for is determined by the level of input and control you need.

This analogy also helps explain why serverless computing represents a further shift in abstraction, where the unit of focus moves away from infrastructure and platforms toward individual applications or pieces of business logic.

Do you want to enjoy eating pizza as quickly as possible, and you're fine to let somebody else make it, select the toppings, and cook it? The trade-off is that you do not get a say in what ingredients they use to make the pizza base, what size it is, what toppings, or how well it is cooked. But in this scenario, you accept that you do not need to know or care, so this may be the perfect scenario for you. This is an example of SaaS, a ready-cooked and prepared meal ready to consume—that is, *takeaway as a service*.

However, to contrast that example, you may need to eat pizza but have some requirements you wish to specify. Maybe you want a particular ingredient to be used in the pizza base; maybe this is imperative, as you have a specific intolerance or medical condition, so you must have control over the ingredients. You cannot guarantee this with the takeaway example, as you only get that choice and control when you cook it yourself at home. So, the option with the most control is *kitchen as a service* or IaaS.

But maybe the pizza base is not of concern; you do not want the trouble of making your pizza, as that requires time, effort, and skill, even to know how to make a pizza base. So, you will let somebody else provide that for you, and it will probably be better, faster, and cheaper than how you would make it. This leaves you to focus on choosing your toppings, as this is where the value and fun bit is. Pizza bases are boring, but selecting the toppings is where it gets exciting, and you want to focus on the topping ingredients. This is where the *ingredients as a service*, or PaaS, model now appeals the most.

In summary, each service model has its place, depending on the level of control and responsibility you require.

This analogy focuses on IaaS, PaaS, and SaaS. Serverless computing is introduced in the next section as a model that further abstracts the runtime and execution layer.

What is serverless computing?

The following is Microsoft's definition of serverless computing:

> *Serverless computing enables developers to build applications faster by eliminating the need to manage infrastructure. With serverless applications, the cloud service provider automatically provisions, scales, and manages the infrastructure required to run the code.*

The term *serverless* itself is a bit of a misnomer, as in reality, there are servers involved, much like how *wireless* does have wires involved at a certain point in the solution. The servers do exist, but you do not need to know that they exist for you to have your desired outcome met.

This comes back to the topic of abstraction: the same layers still exist, and there are still servers that execute the code passed down from the runtime layer. It is just that the server layer is now further abstracted from you than it was in the IaaS and PaaS models.

The benefit of this further abstraction is that there are even fewer components to create and manage. This allows development teams to focus on writing their core code without considering what's running it; the provider automatically provides the resources needed to run the code. That means faster and more productive development teams, fewer operational overheads for DevOps teams, more significant innovation, and quicker time to value and return on investment for development resources.

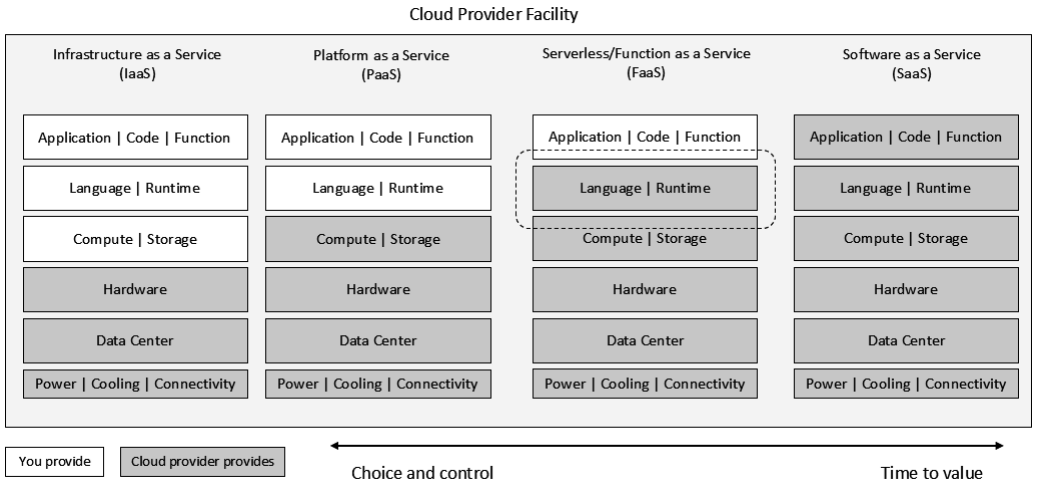


Figure 1.11: Cloud computing service models: compare and contrast

Some of the caveats of serverless are as follows:

- Event-based workloads are the best use case
- Long-running or highly stateful tasks may be less suitable, depending on platform limits, execution patterns, and cost considerations.
- The execution environment cannot be customized
- The cloud provider supports specific languages and runtimes

You learned, in this section, about the four service models of cloud computing: IaaS, PaaS, FaaS/serverless, and SaaS.

Understanding how these service models differ is important because each offers a different balance of control, responsibility, and flexibility. Comparing them helps clarify which model is best suited to a particular workload, scenario, or business requirement.

How do the cloud computing service types compare?

Each service model has its own characteristics. The most appropriate model is defined by how much you want or need to control, secure, and manage your resources, such as your apps, code, data, networks, and security. In the last section, you learned about the service models. In this section, you will look at the characteristics of each model in more detail to know when you may choose one service model over the other.

Characteristics of IaaS

In a nutshell, IaaS is a model where you can host your virtual machines and infrastructure services on hardware provided for you and shared with other tenants.

The cloud provider is responsible for providing all layers up to and including the hardware. You are responsible for providing all layers above the hardware layer (refer to *Figure 1.10*).

The following are the characteristics of IaaS:

- You create the **virtual machine** (**VM**) (installing the OS and software), storage, and computing resources as you would in a traditional on-premises computing model; this can be likened to a virtual data center. There is the ability to provide fault tolerance and redundancy through availability solutions in case of failure within an Azure data center, zone, or region.
- You can increase resources such as processors, memory, and virtual machine storage by using self-service without requiring redeployment or creating a new virtual machine of the required specifications.
- Based on the **OpEx** model, you only pay for resources you consume on a *pay-as-you-go basis*. You only pay for a virtual machine while it's running; therefore, your month-to-month running charges may differ across virtual machines if you run them for different amounts of hours in the month. You will not be charged while a machine is in the stopped/deallocated state. Storage costs will still be charged if you persist data on disks.
- You get the greatest control and flexibility to deploy, configure, manage, and support resources as you require. IaaS requires the greatest management, administrative, and operations overhead.
- You have direct access and complete control over virtual machines, the OS, and any roles/ services, such as web servers, application servers, and SQL servers, that may be

required to be installed/running on the virtual machines. You also have complete control over networking, security, and protection decisions.

The following are examples of Microsoft IaaS resources:

- Azure Virtual Machines
- Azure Storage
- Azure networking

IaaS is most commonly used when organizations need maximum control over operating systems, middleware, and network configuration. It is well-suited to legacy workloads, lift-and-shift migrations, and scenarios where specific OS-level customization or third-party software is required.

While IaaS offers the greatest flexibility, it also carries the highest operational responsibility, which is why many organizations look to PaaS when they want to reduce management over

Characteristics of PaaS

In a nutshell, PaaS is a model where you host your application, code, data, and business logic on compute and storage resources that are secured and isolated from other tenants.

The cloud provider is responsible for providing all layers up to and including the compute layer. You are responsible for providing all layers above the compute layer (refer to *Figure 1.10*).

The following are the characteristics of PaaS:

- PaaS provides a ready-to-use environment and platform for the fast deployment of hosting web applications, code, business logic, data, and so on. Using predeployed resources, development frameworks, languages, and runtimes provided as a service, there is a quicker time to value and consumption of the service.
- PaaS provides on-demand autoscaling.
- You have the ability to increase resources by changing the pricing tier of the service using self-service. You must still select the underlying compute resources sizing to host your app, code, and so on.
- You have no direct access to or control over the virtual machines or any applications, services, or roles that may be installed. Also, you do not get to specify or control which versions are available.
- PaaS gives you the least control and flexibility because the service provider is responsible for the compute and storage resources layer. Still, it requires the least management, administrative, and operations overhead.

The following are examples of Microsoft PaaS services:

- Azure App Service
- Azure SQL Database
- Cosmos DB
- Azure Files
- Entra Domain Services

PaaS is commonly used when organizations want to focus on application development and data rather than infrastructure management. It is well-suited to modern web applications, APIs, and data-driven workloads, where scalability, availability, and rapid deployment are more important than operating system–level control. By removing the need to manage servers and runtimes, PaaS reduces operational overhead while still giving development teams flexibility in how applications are built and deployed.

Characteristics of FaaS (serverless)

In a nutshell, FaaS is a model where you provide your code and business logic, and the cloud provider hosts it in their language, runtime, and compute environment, which is shared with other tenants.

The cloud provider is responsible for providing all layers up to and including the language, runtime, and compute. You are responsible for providing all layers above, that is, the business logic layer (refer to *Figure 1.11*).

The characteristics of FaaS are as follows:

- Azure Functions is a serverless code engine. It has the use case of events that trigger code; the Azure Functions code that is executed is a response or action based on an event trigger.
- Azure Logic Apps is a serverless workflow engine. It has the use case of events that trigger workflows. A Logic Apps workflow is a response or action based on an event trigger.
- You only have control over your application, code, and business logic layers. All other layers are provided as a service that you have no access to or control over. Essentially, you take the layers supplied to you and use them to execute (run/launch) your code/workflow. Using the pizza analogy covered earlier in this chapter, this is much like what could be described as a *take-and-bake* approach. Ultimately, you are still

responsible for cooking. Still, the FaaS has provided all the layers below that, so the solution is ready to cook, as it were, much like a microwave meal.

- FaaS provides the least control and flexibility but abstracts all the layers below, providing the least amount of deployment, configuration, and maintenance, so you can focus on the application, code, and business logic layers, where the value is, without needing to be concerned about the lower layers.

The following are examples of Microsoft serverless resources:

- Azure Functions
- Azure Logic Apps

Serverless computing is most commonly used for event-driven and highly scalable workloads, such as data processing pipelines, background jobs, APIs, and automation tasks. It is particularly effective when execution patterns are unpredictable or bursty, as resources are consumed only when code runs. By abstracting the runtime and execution environment entirely, serverless further reduces operational responsibility, allowing teams to focus exclusively on business logic rather than infrastructure or platform management.

Characteristics of SaaS

In a nutshell, SaaS is a model where you consume an application provided for you and share it with other tenants.

The cloud provider is responsible for providing all layers up to and including the applications. You are not responsible for any layers other than the configuration and consumption of the app (see *Figure 1.11*).

The following are the characteristics of SaaS:

- The cloud provider installs the application/solution and is responsible for its updates, scalability, availability, and security.
- SaaS provides the best time to value as there is no development time or resources required to create an application. It can be directly configured as needed and used instantly.
- In the pizza analogy, all the layers are taken care of, so all you need to do is just eat; this is the classic **takeaway** approach.
- SaaS provides the minimum amount of control and input on the lower layers.

The following are examples of a Microsoft SaaS solution:

- Microsoft Teams
- Microsoft Exchange Online

- Microsoft SharePoint Online
- Microsoft OneDrive
- Microsoft Dynamics 365

The following are examples of other vendors' SaaS solutions:

- Zoom
- Salesforce
- Dropbox
- Google Mail/Google Docs

This section covered the service models of IaaS, PaaS, and SaaS and introduced the concept of serverless computing as an extension of PaaS. By comparing these models, you explored how control, responsibility, and operational effort shift between the cloud provider and the customer as more layers are abstracted.

Since each service model places different responsibilities on the provider and the customer, understanding where those responsibilities sit becomes critical for making correct security, governance, and operational decisions. For this reason, the next section introduces the shared responsibility model, a foundational concept that explains how accountability is divided in cloud environments and why misunderstandings in this area are a common cause of security and compliance issues.

What is the shared responsibility model?

The **shared responsibility model** is a governance and control model that is critically important to grasp when operating resources and services within a public or hybrid cloud environment.

You should know when it *is* your responsibility to provide the appropriate level of control and when it is *not* your responsibility but that of the cloud service provider. This responsibility level may dictate what cloud computing service models you decide to deploy, such as IaaS, PaaS, or SaaS, to determine how much control and responsibility you must have or hand off to the cloud service provider.

There are three levels of responsibility to be considered. They are as follows:

- Responsibilities that the consumer of cloud services **always retains**
- Responsibilities that vary by **resource type**
- Responsibilities that will **transfer** to the cloud service provider

Figure 1.12 aims to visually set out the division or separation of responsibilities between the consumer of the cloud resources and the cloud service provider:

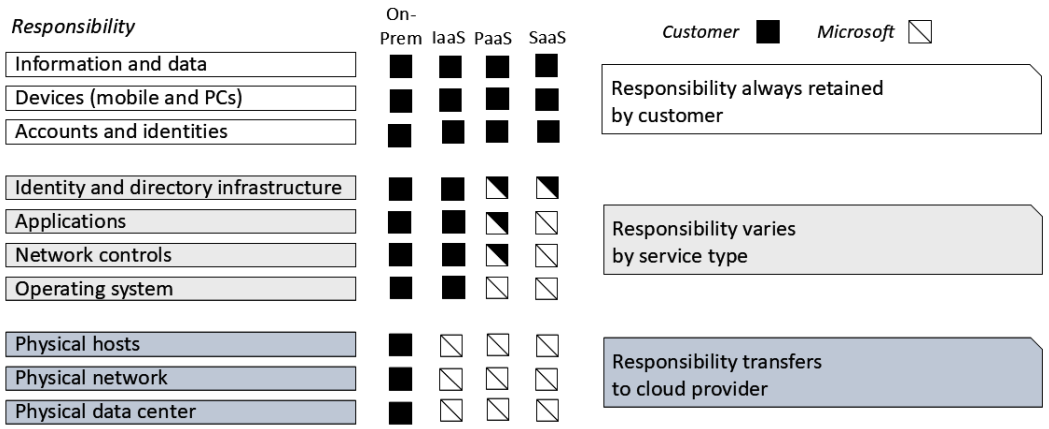


Figure 1.12: Shared responsibility model

Figure 1.12 illustrates how responsibility for different layers of the technology stack is divided between the customer and the cloud service provider across on-premises environments and the cloud service models of IaaS, PaaS, and SaaS. Each row represents a specific responsibility area, such as data, identity, applications, operating systems, or physical infrastructure, while each column shows how that responsibility is allocated depending on the service model being used.

The shaded indicators highlight which party is responsible for each layer. Responsibilities related to information, data, devices, and identities always remain with the customer, regardless of the service model. Responsibilities for platform, operating system, and application layers vary depending on whether IaaS, PaaS, or SaaS is used. Responsibilities for physical infrastructure—including data centers, physical hosts, and networking—transfer entirely to the cloud provider in all cloud service models.

An important consideration in the handoff for the responsibility of the cloud platform components is that the responsibility changes depending on the cloud service model, such as IaaS, PaaS, and SaaS. The cloud service provider has the most responsibility in the SaaS model, but the *least responsibility* in the IaaS model.

For example, the cloud provider is *always responsible* for the physical hosts, networks, and data centers across IaaS, PaaS, and SaaS. While the cloud provider has a *shared responsibility* with the customer for the applications and network controls for the PaaS model resources, it has *no responsibility* for the applications, network controls, or operating system for the IaaS model resources.

Note

Common cloud misconceptions to avoid

Cloud adoption is not automatically cheaper; cost optimization depends on design, governance, and ongoing management.

Serverless does not mean *no responsibility*—it means fewer infrastructure responsibilities and more emphasis on configuration, identity, and monitoring.

Hybrid cloud is not always a temporary step; for many organizations, it is an enduring operating model driven by latency, resiliency, and regulatory requirements.

Finally, security is never fully handled by the provider; the provider secures the platform, while customers must secure their identities, data, configurations, and access.

By now, you have learned how critically important the shared responsibility model is within a public or hybrid cloud environment. This concludes the learning content for this chapter.

Summary

This chapter included complete coverage of the *AZ-900 Azure Fundamentals* exam skills area, *Describe cloud concepts*.

You learned how to define cloud computing; describe the shared responsibility model; explain cloud delivery models, including public, private, and hybrid; identify appropriate use cases for each model; and understand the consumption-based pricing model.

The chapter also explored cloud service types, including IaaS, PaaS, and SaaS, along with their characteristics and appropriate use cases.

In addition to the required exam content, the chapter introduced broader concepts and real-world context to help prepare you for day-to-day work in Azure-focused roles. In the next chapter, you will explore the benefits of using cloud services.

Additional information and study references

This section provides links to additional exam information and study references:

- Microsoft Learn certification further information:
    - AZ-900 – Microsoft Azure Fundamentals exam guide: `https://learn.microsoft.com/en-us/learn/certifications/exams/az-900/`
    - AZ-900 – Microsoft Azure Fundamentals study guide: `https://learn.microsoft.com/en-us/credentials/certifications/resources/study-guides/az-900`
- Microsoft Learn training further information:
    - AZ-900 – Microsoft Azure Fundamentals course: `https://learn.microsoft.com/ en-us/training/courses/az-900t00`
    - AZ-900 – Microsoft Azure Fundamentals practice assessment: `https://learn.microsoft.com/en-us/credentials/certifications/azure-fundamentals/practice/assessment?assessment-type=practice&assessmentId=23&practice-assessment-type=certification`

Exam Readiness Drill – Chapter Review Questions

Apart from mastering key concepts, strong test-taking skills under time pressure are essential for acing your certification exam. That's why developing these abilities early in your learning journey is critical.

Exam readiness drills, using the free online practice resources provided with this book, help you progressively improve your time management and test-taking skills while reinforcing the key concepts you've learned.

HOW TO GET STARTED

- Open the link or scan the QR code at the bottom of this page
- If you have unlocked the practice resources already, log in to your registered account. If you haven't, follow the instructions in *Free Benefits with Your Book* section in this book's *Preface* and come back to this page.
- Once you log in, click the START button to start a quiz
- We recommend attempting a quiz multiple times till you're able to answer most of the questions correctly and well within the time limit.
- You can use the following practice template to help you plan your attempts:

Working On Accuracy		
Attempt	**Target**	**Time Limit**
Attempt 1	40% or more	Till the timer runs out
Attempt 2	60% or more	Till the timer runs out
Attempt 3	75% or more	Till the timer runs out
Working On Timing		
Attempt 4	75% or more	1 minute before time limit
Attempt 5	75% or more	2 minutes before time limit
Attempt 6	75% or more	3 minutes before time limit

The above drill is just an example. Design your drills based on your own goals and make the most out of the online quizzes accompanying this book.

First time accessing the online resources?

You need to unlock the online practice resources to access the link below. Check the *Free Benefits with Your Book* section in the *Preface* for detailed instructions on how to do this. You only need to unlock the resources once.

Open Quiz `https://packt.link/AZ-9003ech1` or scan the QR code

2

The Benefits of Using Cloud Services

In *Chapter 1*, *Introduction to Cloud Computing*, you learned how to define cloud computing, describe the delivery and service models, and compare characteristics and use case scenarios. In this chapter, we will look at the benefits and value of cloud computing and why cloud platforms act as key enablers of digital transformation. We will also be discussing the mindset shift required when moving from traditional computing models, and we will explore how cloud computing changes day-to-day operations and IT economics through elasticity, metered consumption, and cost management discipline.

Understanding these cloud concepts is essential not only for the AZ-900 exam but also for making informed decisions in real-world environments. A clear grasp of cloud benefits and economic principles helps you evaluate adoption strategies, discuss cost models with stakeholders, and translate business requirements into practical technical solutions.

This chapter primarily focuses on the *Describe cloud concepts* module from the *Skills Measured* section of the *AZ-900 Azure Fundamentals* exam. By the end of this chapter, you will be able to confidently answer exam questions related to the following:

- How does the cloud enable digital transformation?
- What is the cloud computing mindset?
- What is the cloud operations model?
- How does cloud change IT economics?

Where helpful to you, the chapter also connects exam concepts to real-world cloud adoption scenarios to reinforce understanding.

How does the cloud enable digital transformation?

Digital transformation reshapes how an organization delivers value by changing products, processes, and operating models to improve speed, customer outcomes, and time to value. *Figure 2.1* outlines the value that cloud computing as a digital transformation enabler can realize for an organization:

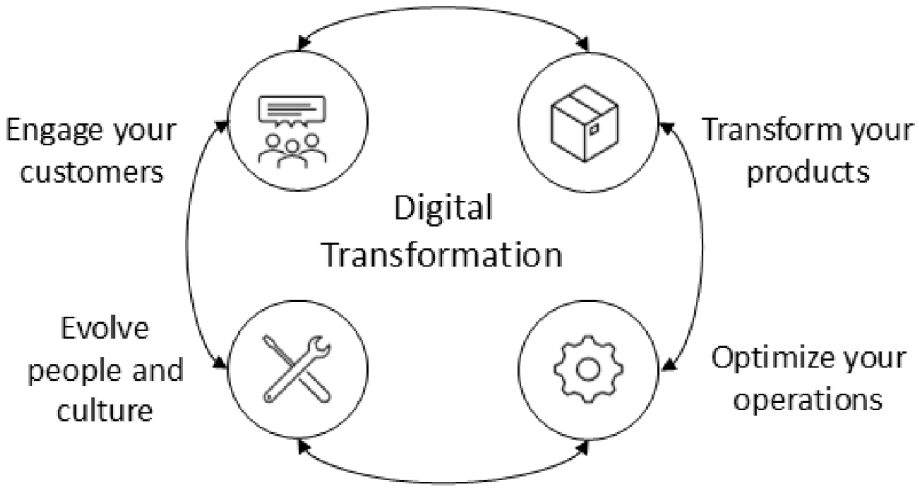


Figure 2.1: Cloud computing as a digital transformation enabler

Figure 2.1 highlights that digital transformation is not a single initiative but a continuous cycle. Organizations engage customers through new digital experiences, transform products by embedding intelligence and connectivity, optimize operations through automation and analytics, and evolve people and culture to support new ways of working. Cloud computing underpins each of these areas by providing scalable infrastructure, platform services, and rapid deployment capabilities that reduce friction between idea and execution.

For example, a retail organization moving from physical-only sales to an online and mobile model may use cloud services to launch a scalable e-commerce platform, integrate customer analytics to personalize offers, automate inventory management, and enable remote collaboration across teams. The technology enables change, but the transformation is realized through improvements in customer engagement, operational efficiency, and business agility.

Cloud computing can be seen as a vital part of any digital transformation journey. However, the reality is that it is less about the technology model and more about the business model, the people, and the process. Organizational change is often driven by clear business pressure

rather than technology alone. However, there must be a trigger to induce the action. Next, we will look at digital transformation triggers.

Digital transformation triggers

The reason to adopt any disruptive technology, especially one that can have a business impact, must start with a **trigger** that's relevant to the stakeholder's pain points or objectives and any business operations or technical operations drivers.

This move often begins with a **business directive-led** discussion rather than a **technology-led** one. A business leader will ask what business outcomes a cloud initiative delivers, not which technical features it includes.

The approach to digital transformation comprises steps to identify and resolve triggers. Once triggers have been identified, an approach can be planned. You can think of the triggers as the *why* and the approach as the *how*.

Figure 2.2 outlines some of the triggers for digital transformation initiatives:

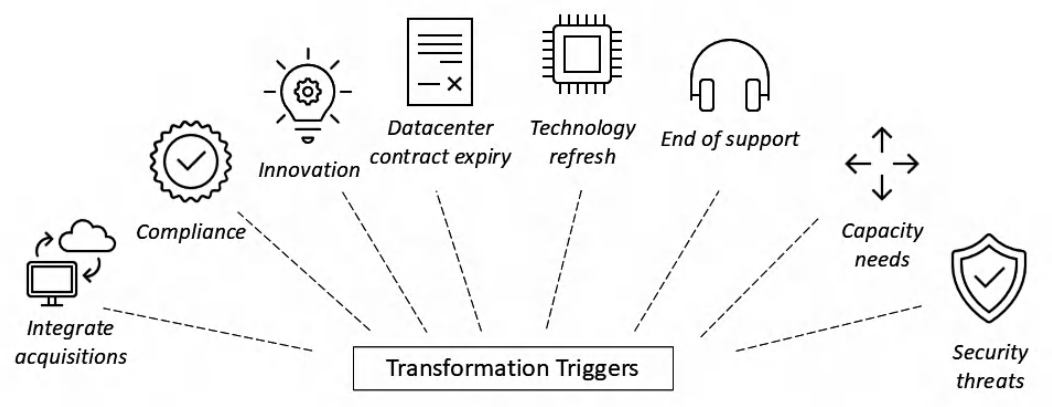


Figure 2.2: Digital transformation triggers

For example, consider an organization facing a data center contract expiry. Rather than renewing long-term infrastructure commitments, leadership may evaluate cloud services to reduce capital expenditure and gain flexibility. In this case, the trigger is contractual and financial, but it initiates a broader modernization discussion.

In another scenario, a company experiencing rapid growth may encounter capacity constraints that limit performance during peak demand. Capacity needs become the trigger for transformation, leading to the adoption of scalable cloud infrastructure that can expand and contract based on usage. What begins as a performance issue evolves into an operational and economic shift.

These triggers are an important aspect of defining your cloud adoption strategy and driving your priorities. Once these transformation triggers are identified and cloud adoption is justified, organizations must decide how workloads will be transitioned to cloud services in a structured and controlled way.

Migration approach

The next step in the transformation journey is to grasp the processes, methods, and services that could be used to execute the transition to cloud services.

Any cloud migration project is not simply the act of moving servers from one location to another. It involves evaluating workloads, selecting appropriate cloud service models, redesigning architectures where necessary, and preparing teams for new operational practices. While there are formal migration frameworks available, the following high-level approach provides a simplified structure to understand how organizations typically transition from on-premises environments to cloud services.

The following high-level migration approach is included for context and is not intended as a prescriptive framework for the exam. This is outlined in the following list, along with the example services that can be applied:

1. **Phase 1 | Assess**: This phase is an exercise in learning an organization's business operations and technical operations. It starts with discovery, which provides an inventory of people, processes, apps, data, infrastructure, and, critically, any dependencies. This will help you decide on the best strategy for each area of the captured inventory.
2. **Phase 2 | Move**: This phase is the physical exercise of moving inventory items to their cloud services target, for instance, moving to **Software as a Service** (**SaaS**), **Infrastructure as a Service** (**IaaS**), **Platform as a Service** (**PaaS**), or serverless cloud computing services.
3. **Phase 3 | Optimize**: This phase occurs after a period of bedding in – typically, a period of months. This may include right-sizing activities to suit the workload for cost and performance benefits.
4. **Phase 4 | Secure and Manage**: This ongoing operations phase should include governance and control, security, and the protection of the network, app, data, and identities.

Migration should be viewed as a milestone rather than an endpoint. After workloads move to the cloud, organizations typically continue to evolve their environments by improving security posture, refining operational practices, and optimizing cost and performance over time. This ongoing evolution reinforces the importance of adopting a cloud mindset rather than treating

migration as a one-time technical exercise. The following section will look closely at the cloud computing mindset.

What is the cloud computing mindset?

The biggest challenge to cloud adoption is not transforming the technology but changing the **mindset** and **culture** within an organization. A cloud mindset also favors standardization over reinvention. Instead of building every environment as a bespoke project, organizations use repeatable patterns, shared baselines, and automation to improve speed, consistency, and control at scale. The same thing has been happening with DevOps over the years. The realization over time has been that **DevOps** is not a technology or a job title but a cultural shift in a people and process model.

In the coming sections, let's look at the different mindsets that have been helpful in the adoption of cloud computing.

What is the traditional computing model mindset?

The following is an outline of the traditional computing model mindset:

- CapEx cost expenditure model
- Over-provision for peak demand and build in five-year growth from day one
- Fixed size and scaling approach
- Long procurement lead times and delayed resource availability
- Monolithic, tightly coupled infrastructure stack approach
- Always-on 24/7 operation

This traditional mindset was not inherently flawed. It developed in a context where infrastructure was costly, capacity was finite, and procurement cycles were long. Planning for peak demand and investing upfront in fixed infrastructure provided predictability and operational stability. For many years, this approach supported reliable enterprise systems and steady business growth.

However, as organizations began to compete in digital markets, the same characteristics became constraints. Over-provisioning tied up capital. Fixed capacity limited experimentation. Long lead times slowed innovation. What had once ensured control and reliability began to restrict speed and adaptability. The shift toward cloud computing emerged not because the traditional model was wrong, but because business demands changed.

The following section compares and contrasts the traditional computing model mindset with the cloud computing mindset. This shift in mindset directly shapes how cloud environments are operated day to day, leading to a fundamentally different cloud operations model.

What are the key characteristics of a cloud computing mindset?

The following outlines a cloud computing model mindset that you can compare with a traditional computing mindset:

- OpEx cost expenditure model
- Usage versus provisioning; just-in-time, demand-driven provisioning
- Cloud computing is designed to be elastic, scale in and out, and burst to meet demand
- Consume and pay only for resources while they are in use
- Microservices, loosely coupled, business logic-centric approach
- Platform-first thinking with portable design principles where appropriate

A shift in mindset does not remain theoretical; it changes how systems are designed, deployed, and operated. If the cloud computing mindset emphasizes elasticity, usage-based consumption, and rapid experimentation, then the operating model must support automation, real-time scaling, and continuous improvement. This is where the cloud operations model becomes critical.

What is the cloud operations model?

Cloud computing is elastic, scalable, agile, fault-tolerant, highly available, and designed to support disaster recovery. These operational model characteristics in cloud computing add value and benefit an organization's operational model. These characteristics allow cloud workloads to scale vertically and horizontally while maintaining availability in line with actual demand.

In practice, this means the platform supports rapid provisioning, automated scaling, and resilience patterns that would be expensive and slow to implement in traditional environments. This shift changes operations from capacity planning months in advance to managing policies, metrics, and automation that respond to demand in near-real time.

This ability to be elastic in nature allows the agility to provide a highly effective operations and economics model to flex with the changing demands of a business. By optimizing running hours and right-sizing resources in line with demand and changing requirements, switching to a consumption-based system of paying as you use resources allows monitored spending without the over-commitment of a traditional computing cost model.

Figure 2.3 outlines the computing resources demand model and shows the implications of actual demand against implemented resources based on predicted demand:

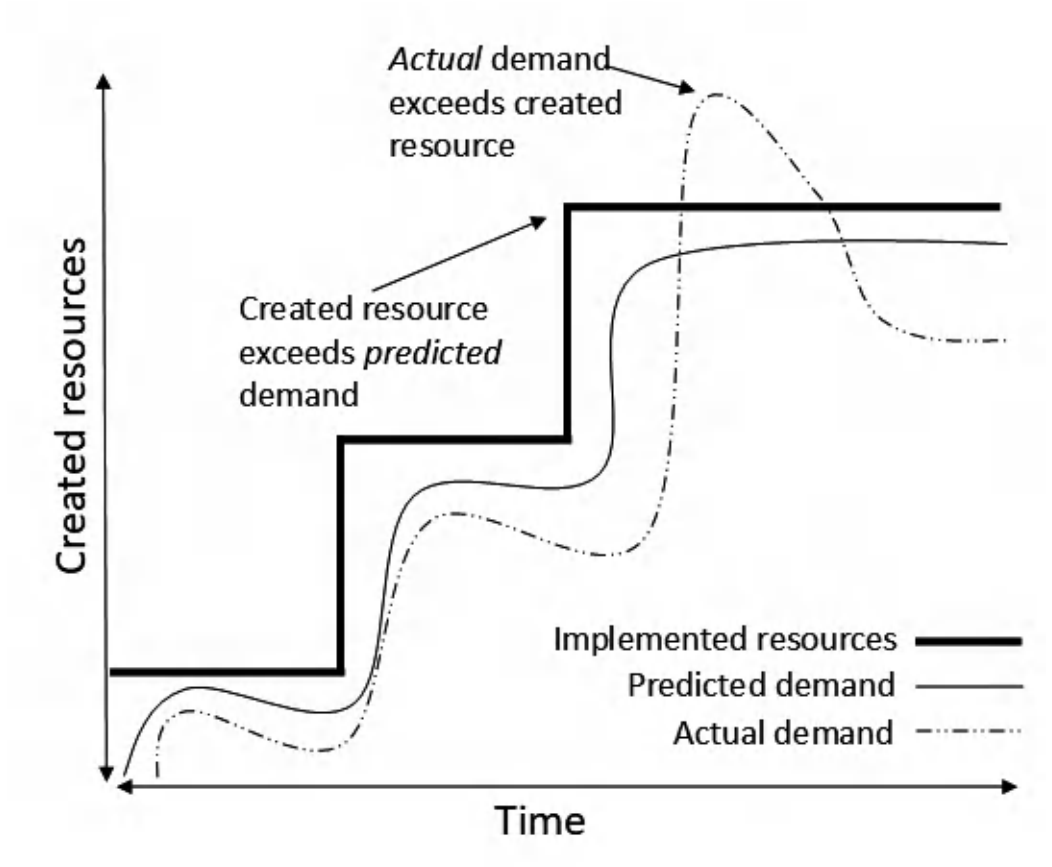


Figure 2.3: Cloud computing resource demand model

Figure 2.3 illustrates the traditional computing mindset through the stepped **Implemented resources** line. Resources are provisioned in large increments based on predicted demand, often exceeding actual need for long periods. When actual demand exceeds the predicted level, there is no immediate burst capacity because resources were sized in advance rather than dynamically scaled. By the time additional resources are implemented, demand may already have dropped, leaving excess capacity that is no longer needed. With the cloud computing mindset, resource utilization can be tracked and right-sized to demand. So, there is far less risk of over-provisioning and paying for more resources than are needed.

With this knowledge of the cloud computing operations model, we will look more closely at the characteristics of cloud computing that deliver benefits and value over the traditional computing model.

While cloud benefits such as scalability, high availability, security, and manageability apply across all cloud computing models, the way these benefits are delivered depends on the chosen service model:

- In IaaS, cloud platforms provide the underlying infrastructure, but benefits such as scalability, availability, and disaster recovery must be designed and configured by the customer. For example, virtual machines do not scale or provide high availability by default unless additional services such as availability zones, load balancing, or autoscaling are explicitly implemented.
- In PaaS and SaaS models, many cloud benefits are built into the service by default. High availability, fault tolerance, patching, and scaling are often handled automatically by the platform or service provider, reducing operational effort and management overhead.

Understanding this distinction is important when evaluating cloud benefits in exam scenarios. Questions that describe *greater customer control* and *responsibility* typically align with IaaS, while scenarios emphasizing *reduced management*, *built-in availability*, or *rapid deployment* usually point to PaaS or SaaS.

Tip

A reliable exam rule of thumb is that as you move from IaaS to PaaS to SaaS, you trade control for reduced operational responsibility and faster time to value.

The operational model we described is realized through a set of tangible benefits that distinguish cloud computing from traditional computing environments. Let's look at them next.

Operational benefits of cloud computing

Cloud computing platforms primarily provide the following operational benefits over traditional computing models:

- Scalability
- Elasticity
- Agility
- High availability (and geo-distribution)

- Disaster recovery
- Consumption-based cost model

These operational benefits may be an inherent built-in platform function that provides features as part of the service, as is typically the case with PaaS or **Function as a Service** (**FaaS**) and SaaS. These operational benefits just need to be enabled in some cases, if not automatically included as part of the services.

These operational benefits could also be something that needs to be designed into part of the solution as an individual set of resources that need to be implemented to enable these characteristics. For example, IaaS virtual machines will not provide scale, elasticity, high availability, and disaster recovery without these being designed into the solution and then implementing resources to provide the functionality to provide each of these characteristics.

The key takeaway is that cloud platform providers will generally provide these functions and characteristics. You may layer on additional functionality as your needs dictate.

Cloud benefits often come with trade-offs. Higher availability can increase cost, stronger security can add operational steps, and rapid deployment can create sprawl without governance. Understanding these trade-offs helps you choose the right service model and design approach in both exam scenarios and real-world deployments.

Of course, not everything is perfect with the cloud computing model. Here are some challenges that can be overcome but must be considered and provided for:

- Network dependency, i.e., reliability, stability, quality, and performance
- **Confidentiality, integrity, and availability** (**CIA**) of users, apps, and data
- Access control and operational governance
- Cost control

Now that we have considered the operational benefits and challenges, it is time to look at the benefits of cloud computing in more detail.

What is scalability?

Scalability refers to how to react to and increase resources based on demand, usually in an automated way triggered upon a metric such as a time or resource threshold being reached. The following two concepts are related to the scalability of computing resources:

- **Scaling up** (**vertical scaling**): This means capacity is increased within the resource, such as increasing the processor or memory by resizing a virtual machine; the opposite is **scaling down**, where resource capacity is decreased.
- **Scaling out** (**horizontal scaling**): This means additional resource instances, such as adding other virtual machines or compute node/scale units; the opposite is **scaling in**, where resource instances are de-allocated.

This concept is illustrated in *Figure 2.4*:

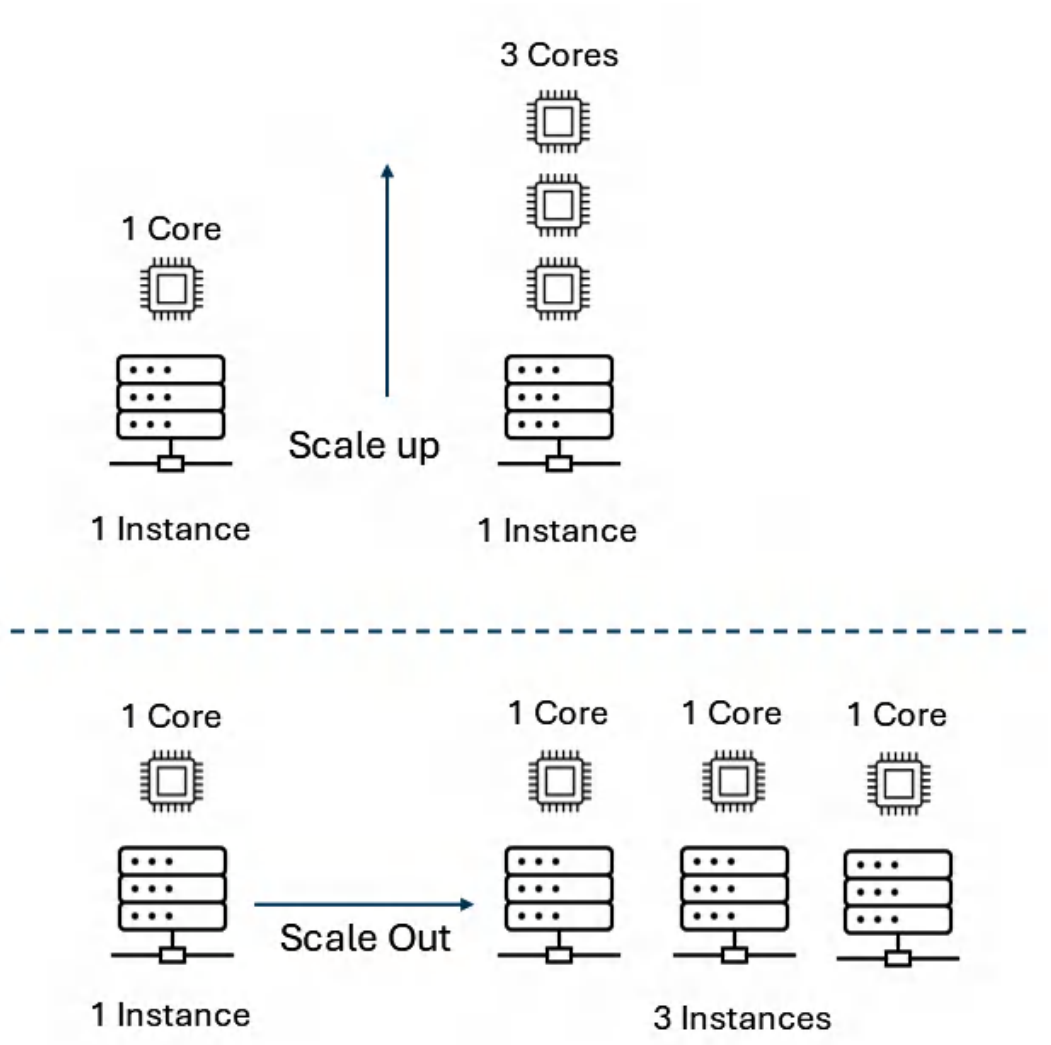


Figure 2.4: Vertical and horizontal scaling

Figure 2.4 illustrates the difference between vertical scaling (scale up) and horizontal scaling (scale out). In the upper example, a single instance is increased in size by adding more compute capacity, such as moving from one core to three cores. The workload still runs on one machine, but that machine becomes more powerful. In the lower example, capacity is increased by adding more instances of the same resource. Instead of enlarging a single server, multiple servers share the workload, improving resilience and flexibility.

Scalability should not be confused with fault tolerance, which moves a workload automatically to another resource or system when it detects a failure or unhealthy state.

What are elasticity and agility?

Elasticity refers to the ability to automatically adjust resources to match demand. When demand increases, additional capacity can be provisioned quickly. When demand decreases, resources can be scaled back down to avoid unnecessary cost and excess capacity. Elasticity focuses on adjusting capacity dynamically.

Agility, by contrast, focuses on how quickly new resources can be deployed and configured to respond to changing business requirements. Instead of waiting for hardware procurement or lengthy setup processes, cloud resources can be provisioned in minutes, enabling faster experimentation and quicker response to operational needs.

What is high availability?

High availability is explored in more depth than other operational characteristics because it is closely tied to **service-level agreements** (**SLAs**), service design, and decision-making. Before we look at SLA's, **high availability** (**HA**) can be understood as illustrated in *Figure 2.5*:

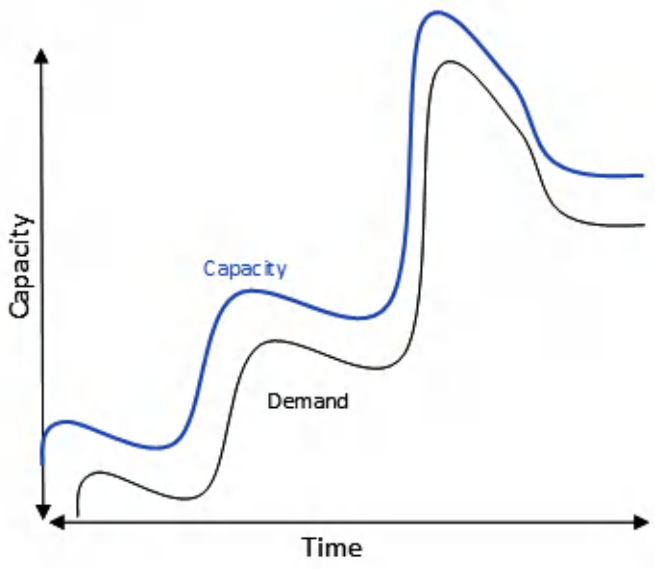


Figure 2.5: High availability

HA and **geo-distribution** mean deploying resources to operate within the required or mandated SLA for those resources. *Figure 2.5* shows demand fluctuating over time and the corresponding capacity provisioned to ensure that service remains available. HA requires that capacity consistently meets or exceeds demand to maintain SLA commitments.

An SLA sets out an expected level of service that a customer can expect from their service provider. This agreement will set out terms such as availability metrics, service availability, responsibilities, claims, and credit processes, as well as the vocabulary and terminology that will be used to express these aspects of the agreement. The SLA is a guaranteed measure of uptime, which is the amount of time services are online, available, and operational.

The following are the concepts of availability in the context of computing and systems:

- **Availability** is the percentage of time a resource is available to service requests.
- **Service availability** is expressed as the uptime percentage over time, for example, 99.9%.
- Availability depends on **resilient systems**, meaning that a system can continue to function after recovering from failures.
- Increasing availability often results in an increase in costs due to the complexity of the solutions required to deliver the level of availability.
- **Failover** is another critical factor in availability. This means one system takes over from another when a resource fails and becomes unavailable, and is part of an availability and disaster recovery strategy.

Microsoft defines an SLA as its commitment to service uptime and connectivity—that is, the amount of time services are *online*, *available*, and *operational*. Microsoft provides each service with an individual SLA that will detail what is covered by the agreement and any exceptions. For any service that does not meet the guarantees, a percentage of the monthly fees is eligible to be credited; each service has its own defined SLA.

While you see lots of references to *availability* and *uptime* when looking at an SLA that will be provided for a service, the customer and consumer of the services will want to know what that means in the real world and what impact any breach may have on them. Therefore, it is often the case that the real metric that matters is **downtime**, which means for a given SLA, how long that service is permitted to be down (that is, the service is not available from the service provider). You should scrutinize any SLA to determine whether that level of downtime is acceptable.

The service availability depends on the number of nines, as in three nines denote 99.9%, and five nines is 99.999% of the SLA.

Microsoft SLAs are typically expressed on a monthly basis, and small differences in the SLA percentage can translate into meaningful differences in permitted downtime, as shown in the following table:

SLA of a service	Permitted downtime per month
99.9%	43m 12s
99.95%	21m 36s
99.99%	4m 19s
99.999%	26s

Table 2.1: The SLA for a service indicating the acceptable level of downtime per month

SLA values vary by Azure service and service tier. While many Azure services provide SLAs such as 99.9% or higher, not all services offer an SLA, and SLA commitments differ depending on the service and configuration. It should also be noted that 100% availability cannot be guaranteed.

You should also be aware of the concept of a **composite SLA**. This means that when you combine services (such as virtual machines and the underlying services, including storage, networking components, and so on), the overall SLA is lower than the individual highest SLA on one of the services. This is because each service that you add increases the probability of failure and increases complexity.

The following actions will positively impact and increase your SLA:

- Using services that provide an SLA (or improve the service SLA), such as Entra ID Premium editions and Premium SSD managed disks
- Adding redundant resources, such as resources to additional/multiple regions
- Adding availability solutions, such as using availability sets and availability zones.

The following actions will negatively impact and decrease your SLA:

- Adding multiple services due to the nature of composite SLAs
- Choosing non-SLA-backed services or free services

The following actions will have no impact on your SLA:

- Adding multiple tenancies
- Adding multiple subscriptions
- Adding multiple admin accounts

The Azure status page (`https://azure.status.microsoft/`) provides a global overview of the service health across all regions. This should be the first place you visit if you suspect that there is a wider issue affecting the availability of services globally. From the status page, you can

click through to **Azure Service Health** in the Azure portal, which provides a personalized view of the availability of the services that are being used within your Azure subscriptions.

Service credits are paid through a claims process by a service provider when they do not meet the guarantees of the agreed service level; each service has its own defined SLA. You should evaluate all your services to ensure that, where required, you always have an SLA-backed service; as they say, there is often an operational impact that's felt from *free services*.

If you suspect that your services have been affected and that Microsoft has not been able to meet its SLA, then it is your responsibility to take action and pursue credit. You must submit a claim to receive service credit. For most services, you must submit the claim the month after the month in which the service was impacted. In partner-managed environments, service credit claims may be handled on the customer's behalf.

Note

You can find more details about Microsoft SLAs for Azure services and the composite SLA at the following links:

- `https://www.microsoft.com/licensing/docs/view/Service-Level-Agreements-SLA-for-Online-Services`
- `https://learn.microsoft.com/en-us/azure/well-architected/reliability/metrics`

What is disaster recovery?

Disaster recovery is based upon a set of practices or measures to ensure that when a system fails, it can be restored to operation by failing over to a replicated instance in another region.

A disaster recovery strategy will be determined by the required **recovery time objective** (**RTO**) and **recovery point objective** (**RPO**). Replication technologies allow for much shorter RTOs and RPOs that can be achieved with backups. The following are the crucial elements in creating comprehensive disaster recovery plans:

- **RTO**: This refers to the maximum duration of acceptable **downtime** for the system
- **RPO**: This refers to how much **data loss** is acceptable to a system.

This is represented in *Figure 2.6*:

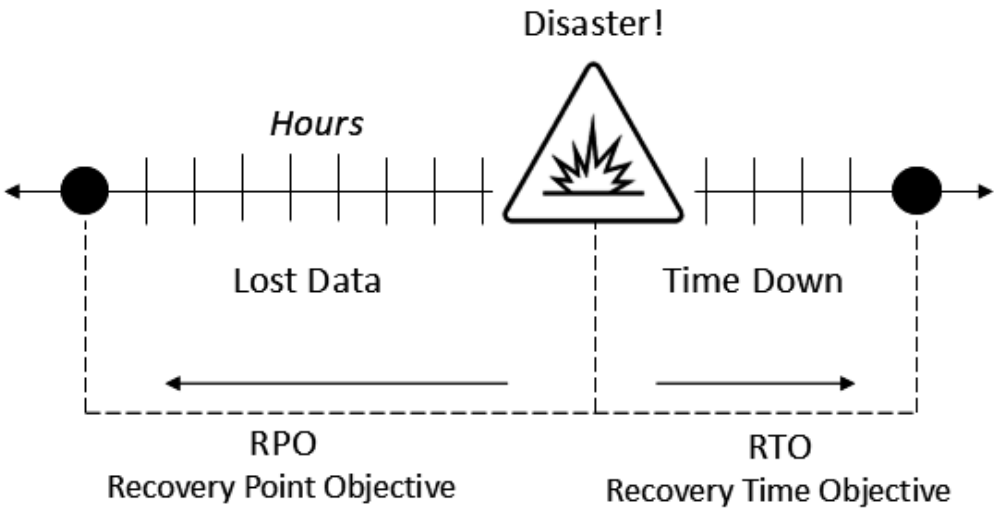


Figure 2.6: RTO and RPO

Figure 2.6 illustrates the difference between RPO and RTO along a timeline of a disaster event. The RPO represents how much data loss is acceptable, measured as the time between the last recoverable backup and the disaster. The RTO represents how long the system can remain unavailable after the disaster before service must be restored. Together, these objectives define acceptable data loss and acceptable downtime.

Having grasped the operational benefits, read on to compare disaster recovery to HA and backup concepts.

Comparing disaster recovery, HA, and backup

The following comparison highlights how HA, disaster recovery, and backup address different failure scenarios and protection goals within a cloud environment:

- **HA**: When systems fail and are not available, you can run a second instance in the same Azure region
- **Disaster recovery**: When systems fail and are not available, you can run a second instance in another Azure region
- **Backup**: When data is corrupted, deleted, lost, or irretrievable (perhaps due to ransomware), you can restore the instance from another copy of the system

Figure 2.7 outlines the three preceding points of HA, disaster recovery, and backup:

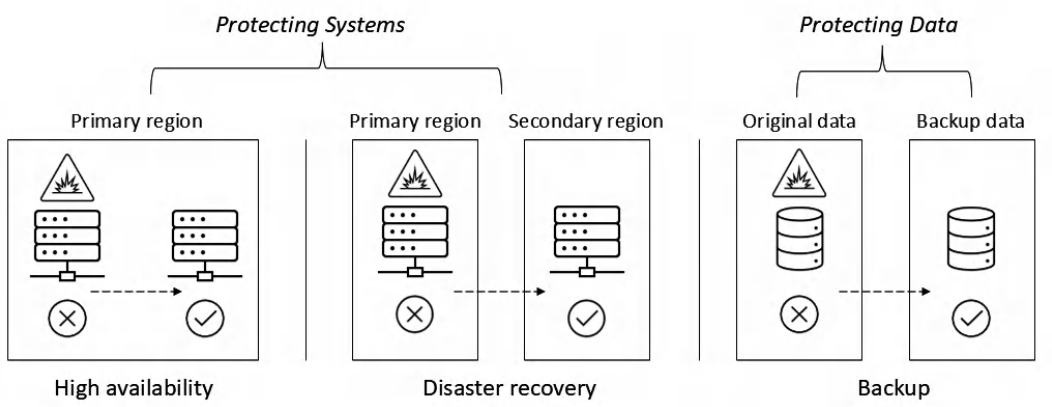


Figure 2.7: Comparing HA, disaster recovery, and backup

HA, disaster recovery, and backup should not be an either-or decision in a strategy for business continuity; any strategy should include all three as they serve different purposes.

Fault tolerance is a means of providing HA by ensuring that a system can continue operating when a component fails. It is different from autoscaling, which responds to increased demand by adding or removing capacity. Fault tolerance and failover are triggered by health checks that detect failures, whereas autoscaling is triggered by load or performance thresholds.

While these mechanisms strengthen resilience, implementing them as part of a broader business continuity strategy introduces practical design considerations and trade-offs.

Challenges of implementing business continuity

Implementing business continuity introduces a set of practical trade-offs related to cost, complexity, and compliance when designing for availability, recovery, and protection. **Cost**, **complexity**, and **compliance** are the biggest challenges for business continuity. These challenges result in systems that are often not covered by disaster recovery or protected by backups, which challenge your ability to comply with any regulatory or internal mandatory policy.

While many disaster recovery and business continuity strategies traditionally focus on technical failures and localized outages, threats to business operations can also arise from large-scale, unforeseen disruption events. These may include global pandemics, supply chain failures, regional incidents, workforce constraints, or sudden shifts in how and where people work. Historically, scenarios of this nature have not always been fully accounted for in continuity planning.

While a pandemic may not constitute a technical disaster or service outage in itself, it can cause significant and sustained disruption that is difficult to predict. Organizations that had

already adopted cloud services and remote working capabilities before the COVID-19 pandemic were generally better positioned to maintain continuity, scale access, and adapt operationally. Cloud adoption and modern workplace strategies can therefore improve resilience by enabling distributed access, flexible capacity, and faster recovery when disruption occurs.

Business disruption, therefore, is not limited to system failures or infrastructure outages. Operational resilience requires organizations to consider how services, data, and users remain accessible during widespread disruption, even when facilities are unavailable or normal working patterns are no longer possible. Planning for these scenarios extends disaster recovery beyond technology and into business operations and workforce enablement.

Figure 2.8 illustrates the transition from traditional operating models to a cloud adoption model, showing how cost structure, complexity, and compliance posture evolve.

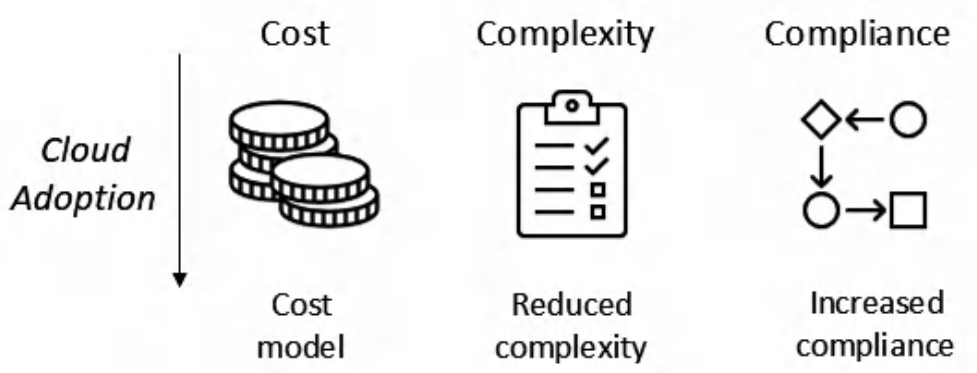


Figure 2.8: Challenges to implementing business continuity

Adopting a cloud strategy utilizing Microsoft Azure can address many of the cost, complexity, and compliance challenges associated with implementing business continuity.

The challenges are often centered around costs, and the benefit and driver can be the changing cost model that can be provided by the cloud. An additional benefit is that there is no need to maintain and purchase the resources required for a secondary site. With Microsoft Azure as the secondary site, only what is used is paid for in a consumption-based model.

With all that we have discussed in this section, we have covered all aspects of the cloud computing operations model, including aspects of the demand model, operational benefits, and compared disaster recovery, high availability, and backup. The operational characteristics discussed so far directly influence how cloud services are consumed and paid for, which leads to the economic model of cloud computing. The following section will cover the economics of cloud computing, the consumption model, and the cost-expenditure model.

How does cloud change IT economics?

This section will first look at the **consumption-based model**, one of the two economic characteristics of cloud computing. The second economic characteristic that cloud computing is based on is the **cost-expenditure model of operational expenditure**, which will be explored later.

Consumption model

In a nutshell, a consumption model means paying only for the time you use the resource. This can be likened to leasing/renting something instead of purchasing and owning the asset outright.

Some resources, such as virtual machines, can be stopped and started to reduce costs, so you only pay while running them. This is one of the key business benefits of the cloud computing cost model over a traditional computing cost model.

Now, let's take a closer look at the cost-expenditure models.

Defining the expenditure models

It is essential to know the finance terms **CapEx** and **OpEx**. The details are as follows:

- **CapEx**: This is the upfront commitment of a large amount of money to purchase assets, such as investment in data center facilities, network circuit implementation, physical hardware, or software, as a one-off payment that you then own for the lifespan of those assets. You can liken this to the model of purchasing a vehicle or a mobile phone handset outright with a sum of money. You own it until it needs replacing, so you have to find another lump sum of money to reinvest to purchase another one in a few years.
- **OpEx**: Essentially, a pay-as-you-go model for consuming assets and resources is the ongoing running costs, which require a recurring payment when the asset or resource is required and in use to deliver services or functionality to a business. This could include staffing costs, data center facility management costs, power, and software that is not purchased outright but leased on a subscription basis. There is no upfront commitment of money to buy assets. Every month, you pay only for the amount you use during a certain period. You can liken this to the model of renting a vehicle or mobile phone handset and paying every month. You never own the asset but use it for as long as you need. You may swap it, upgrade it, and downgrade it as you wish within the terms.

Figure 2.9 aims to represent this:

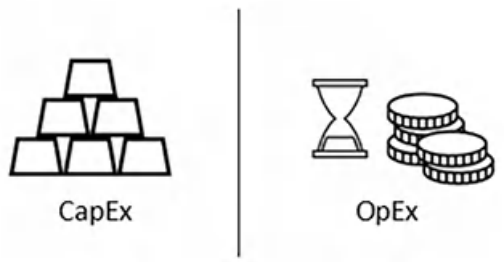


Figure 2.9: Cost expenditure models

For many organizations, the primary value of cloud computing lies in its economic model rather than the underlying technology. Because cloud spending is consumption-based and can change daily, many organizations adopt **FinOps** practices—a collaborative approach between technology, finance, and business teams—to improve cost visibility, accountability, and optimization. The goal is not only to reduce cost, but to ensure spending aligns with business value and to prevent waste as environments scale.

The utilization of the cost model is often the value driver, with the technology being secondary.

In traditional computing environments, expenditure is typically CapEx, whereas cloud computing primarily adopts an operational expenditure OpEx model.

Applying cost expenditure models to cloud computing

In the traditional computing model (pre-cloud platforms), hardware resources and software were purchased upfront with a one-off lump sum. These business assets would be seen as depreciating assets; this is the CapEx model. In contrast to the public cloud computing model, resources provided by the cloud platform are shared with others. This means there has been no CapEx to provision hardware, so you can start using these resources on demand. These resources are treated as day-to-day OpEx.

The exception is **reserved instance resources**, where you can commit to a usage term of one to three years, paying upfront (or monthly) on a CapEx basis to reserve predicted resource usage. This is more cost-effective over a one- to three-year period than paying on the pay-as-you-go consumption model.

In the case of the private cloud computing model, cloud computing technologies are hosted on dedicated, single-tenant hardware resources; typically, these technologies are self-hosted on hardware in a facility (colocation) but may also be hosted with a third-party hosting provider. In this model, there is an element of CapEx and OpEx. The most significant portion is CapEx for purchasing hardware and assets, with usage (and often licensing and software) as OpEx.

The **hybrid cloud approach** will give the greatest flexibility in the expenditure model. This approach will allow you to decide the resources and respective expenditure models that they utilize to fit the business's needs best (*Figure 2.10*):

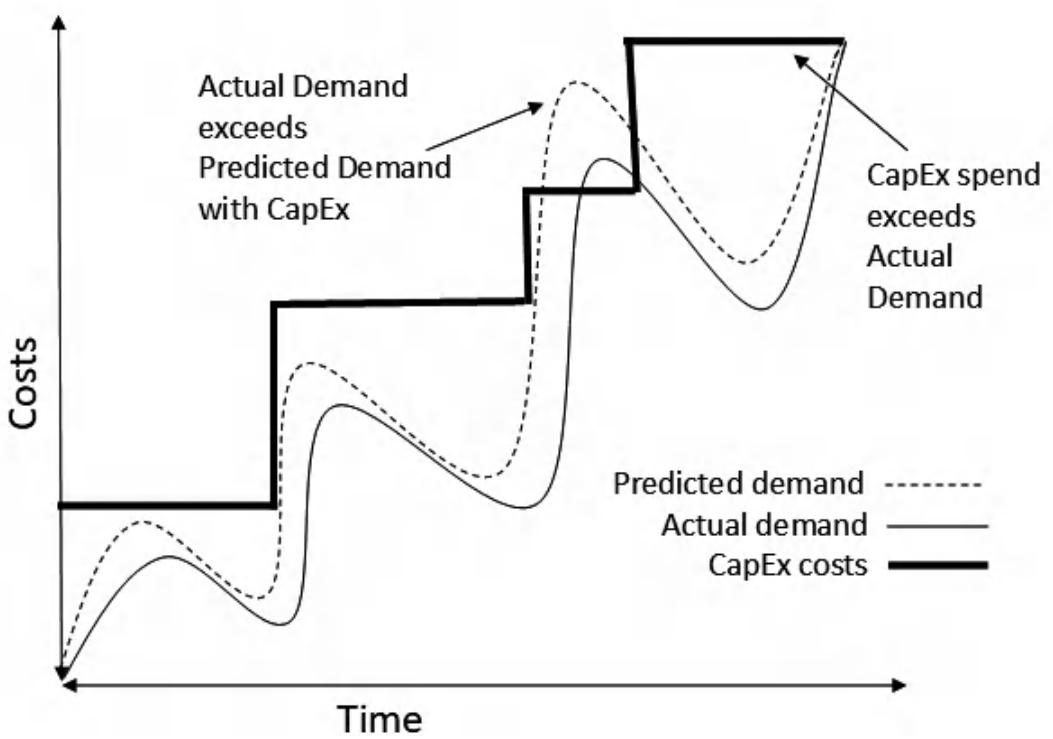


Figure 2.10: Application of cost expenditure models

The consumption cost model of cloud computing ensures not committing large amounts of capital expenditure on depreciating assets. Instead, by keeping that money within the business to use for day-to-day expenses, paying for what you need when you need it, and only for as long as you need it, you can keep your costs much closer to actual demand rather than over-committing to costs for a predicted demand. When resource demand exceeds predicted demand, the nature of the cloud computing model will ensure you have resources that can be scaled in and out to meet that demand.

Summary

This chapter included complete coverage of the *Describe the benefits of using cloud services* skills area of the *AZ-900 Azure Fundamentals* exam. This content progressed from why organizations adopt cloud services to how cloud environments are operated and how cloud consumption is paid for.

You learned about the benefits of using cloud services. You looked at cloud computing as a digital transformation enabler, the triggers, the cloud migration approach, and adopting a cloud mindset. In addition, you learned about various aspects of a cloud operations model and the economics of cloud computing. Further knowledge beyond the required exam content was provided to prepare you for a real-world, day-to-day Azure-focused role.

The next chapter will look at Azure's core architectural components.

Exam Readiness Drill – Chapter Review Questions

Apart from mastering key concepts, strong test-taking skills under time pressure are essential for acing your certification exam. That's why developing these abilities early in your learning journey is critical.

Exam readiness drills, using the free online practice resources provided with this book, help you progressively improve your time management and test-taking skills while reinforcing the key concepts you've learned.

HOW TO GET STARTED

- Open the link or scan the QR code at the bottom of this page
- If you have unlocked the practice resources already, log in to your registered account. If you haven't, follow the instructions in *Free Benefits with Your Book* section in this book's *Preface* and come back to this page.
- Once you log in, click the START button to start a quiz
- We recommend attempting a quiz multiple times till you're able to answer most of the questions correctly and well within the time limit.
- You can use the following practice template to help you plan your attempts:

Working On Accuracy		
Attempt	**Target**	**Time Limit**
Attempt 1	40% or more	Till the timer runs out
Attempt 2	60% or more	Till the timer runs out
Attempt 3	75% or more	Till the timer runs out
Working On Timing		
Attempt 4	75% or more	1 minute before time limit
Attempt 5	75% or more	2 minutes before time limit
Attempt 6	75% or more	3 minutes before time limit

The above drill is just an example. Design your drills based on your own goals and make the most out of the online quizzes accompanying this book.

First time accessing the online resources?

You need to unlock the online practice resources to access the link below. Check the *Free Benefits with Your Book* section in the *Preface* for detailed instructions on how to do this. You only need to unlock the resources once.

Open Quiz `https://packt.link/AZ-9003ech2` or scan the QR code

Part 2

Azure Architecture and Services

This section provides coverage of the knowledge and skills required for the *Skills Measured* of the *Describe Azure Architecture and Services* section of the exam. We also cover knowledge and skills that go beyond the exam content, so you are prepared for a real-world, day-to-day Azure-focused role.

This part of the book includes the following chapters:

- *Chapter 3, Azure Core Architectural Components*
- *Chapter 4, Azure Compute Services*
- *Chapter 5, Azure Storage Services*
- *Chapter 6, Azure Network Services*
- *Chapter 7, Azure Identity and Access*
- *Chapter 8, Azure Security*

3

Azure Core Architectural Components

In *Chapter 2*, *The Benefits of Using Cloud Services*, you explored why organizations adopt cloud services and how cloud platforms enable digital transformation. This chapter builds on that foundation by introducing core architectural components of Azure and explaining how those components work together to deliver global scale, resilience, and governed resource management.

This content examines Azure's architectural components from two angles. First, it looks at the **physical foundation**—data centers, Microsoft's global network, regions, and availability. Next, it explores the **logical foundation**—how Azure organizes and manages resources using management groups, subscriptions, **Azure Resource Manager** (**ARM**), and resource groups.

Think of it this way: the **physical layer** determines *where* your workloads can live and how resilient they can be, while the **management layer** determines *how* those workloads stay organized, governed, and controllable as you scale.

This chapter primarily focuses on the *Describe Azure Architecture and Services* module from the *Skills Measured* section of the *AZ-900 Azure Fundamentals* exam.

Note

You can find a detailed *AZ-900 Azure Fundamentals* exam skills area in the *Appendix* section of this book, under *Assessing AZ-900 Exam Skills*.

By the end of this chapter, you will be able to explain the following topics clearly and confidently:

- Azure global infrastructure
- Azure resource management

In addition, this chapter includes practical context that goes beyond assessment objectives to help you apply these concepts in real-world Azure environments.

Azure global infrastructure

The key components of the Azure global infrastructure are the **physical data centers**, the **edge infrastructure**, and the **global network**, sometimes called **premises** and **pipes**.

An **Azure physical data center** is a secure facility that hosts physical computing, storage, and networking facilities that provide Azure cloud computing platform resources. Each data center has independent, isolated, redundant power supplies and cooling systems for resilience against outages.

Edge locations are secure facilities where traffic enters and leaves the Microsoft global network. These locations can provide computing resources closer to users for improved network latency, allowing fewer network hops through fewer providers. If required, the traffic can stay longer on the Microsoft backbone network without transiting the internet.

The **Microsoft global network** is one of the largest private networks in the world, with global/cross-ocean fiber that is leased or owned by Microsoft. This global network connects data centers to regional gateways and edge locations where traffic can enter and leave the Microsoft **wide area network** (**WAN**).

Microsoft also provides **ExpressRoute**, a service that allows customers to create private network connections to Azure regions from specific peering locations. This allows traffic from a customer's **Multi-Protocol Label Switching** (**MPLS**) WAN to enter the Microsoft network, bypassing the internet entirely and offering a **low-latency** private connection.

Figure 3.1 outlines the topology of Azure global infrastructure components:

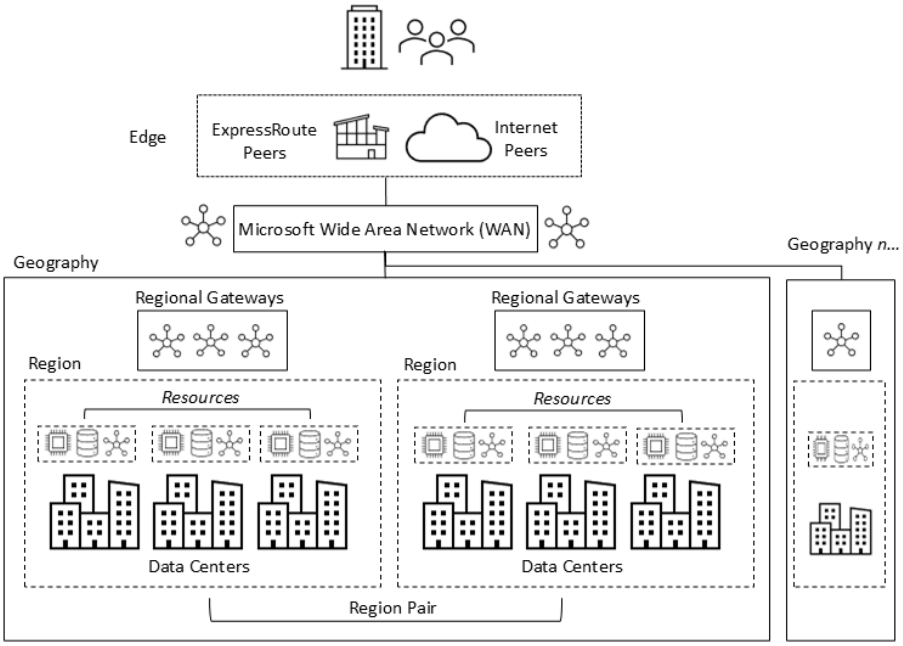


Figure 3.1: Azure global infrastructure key component topology

Figure 3.1 illustrates how Azure's global infrastructure is organized from edge connectivity through to regional deployment boundaries. At the top, customer environments connect through ExpressRoute or internet peering at the network edge. That traffic enters the Microsoft WAN, which provides global backbone connectivity across geographies. Within each geography, regional gateways connect traffic into individual Azure regions. Each region contains one or more data centers that host compute, storage, and networking resources. Regions are paired to support resilience and **disaster recovery** (**DR**) design. Together, these layers show how global connectivity, geographic boundaries, and regional resource deployment are structurally related.

The location for a **resource** will be the one that best meets the organization's needs. It may be a **technical driver**, such as service capability and availability or latency in that region, or a **business driver**, such as compliance, data residency, or even cost.

These physical components are organized into regions and geographies, which define where you can deploy resources and boundaries that shape availability and compliance.

Azure regions and geographies

Microsoft data centers are grouped into collections across different sites to provide redundancy and **high availability** (**HA**) for resources, such as computing and storage, hosted within the data centers.

These sets of grouped data centers are known as **regions** and are exposed as the deployment location when creating resources. There are multiple **geographies**, such as Europe, the UK, the US, and Asia, which contain **multiple regions** with **multiple data centers** in each region.

Regions should not be viewed as isolated locations; instead, they act as building blocks that combine technical capability with regulatory and business constraints.

The following examples illustrate how regions are grouped within different geographies:

- The **Europe** geography includes the following:
    - The North Europe region, located in Ireland
    - The West Europe region, located in the Netherlands
- The **UK** geography includes the following:
    - The UK South region, located in London
    - The UK West region, located in Cardiff
- The **US** geography includes the following:
    - The Central US region, located in Iowa
    - The East US region, located in Virginia
    - The East US 2 region, located in Virginia
    - The East US 3 region, located in Georgia
    - The North Central US region, located in Illinois
    - The South Central US region, located in Texas
    - The West Central US region, located in Wyoming
    - The West US region, located in California
    - The West US 2 region, located in Washington
    - The West US 3 region, located in Arizona
- The **Asia Pacific** geography includes the following:
    - The East Asia region, located in Hong Kong
    - The Southeast Asia region, located in Singapore

Note

A complete list of geographies and regions can be found at `https://learn.microsoft.com/en-us/azure/reliability/regions-list`.

The data centers in a region are connected by a **low-latency network** for availability, which provides the basis for **availability zones**, which you will cover in the next section.

To further support resilience within a geography, regions are "paired" to reduce the impact of widespread events and to enable prioritized platform recovery scenarios.

Microsoft designs paired regions to reduce the likelihood that a single widespread event impacts both locations, and to support prioritized recovery behavior for some platform scenarios. However, paired regions do not mean your workloads automatically fail over by default. For most customer-managed solutions, cross-region failover requires explicit design choices, such as deploying into multiple regions and implementing the appropriate replication, routing, and recovery mechanisms.

Some Microsoft-managed services may use paired regions as part of their built-in resilience strategy; however, for customer-managed workloads, cross-region recovery typically requires explicit design and configuration.

A geography can have multiple regions and define a **data residency** and **compliance** boundary.

Figure 3.2 shows the relationship between Azure regions and geographies:

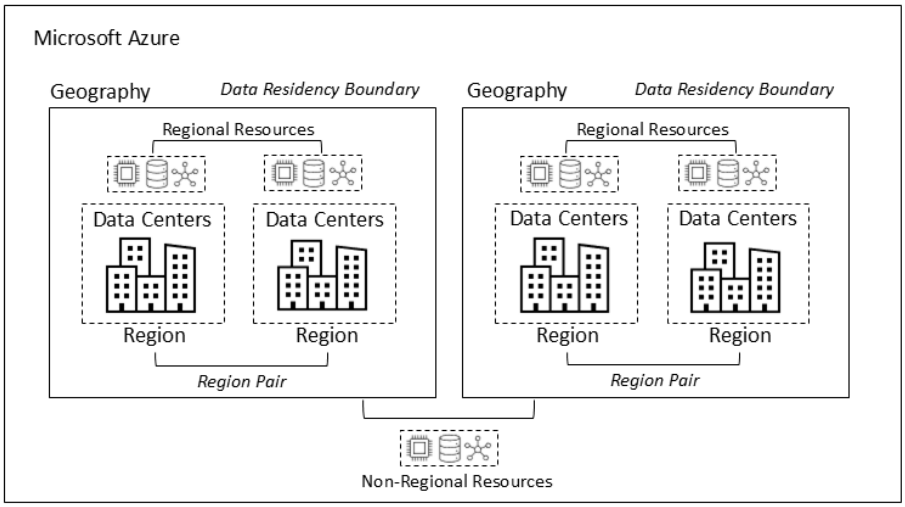


Figure 3.2: Azure regions and geographies relationship

Figure 3.2 illustrates that Azure regions are grouped within geographies, which define data residency and compliance boundaries, and that regions are paired within a geography to support HA and DR, while certain services operate as non-regional resources across the broader geography.

Although Azure resources are physically located in data center facilities, these data centers are not considered individual facilities in which you can create resources. When creating an Azure resource, such as a **virtual machine** (**VM**), it is not possible to directly select a particular data center; only a region can be selected. When creating Azure resources, the region list in the Azure portal may present suggested or commonly used regions depending on factors such as your location, capacity, and service availability. In exam and real-world scenarios, the key point is that *not every Azure service is available in every region*, and availability zone support varies by region and by service. Always validate regional availability and zone support for the specific service you are deploying.

Figure 3.3 shows region selection in the Azure portal when creating resources:

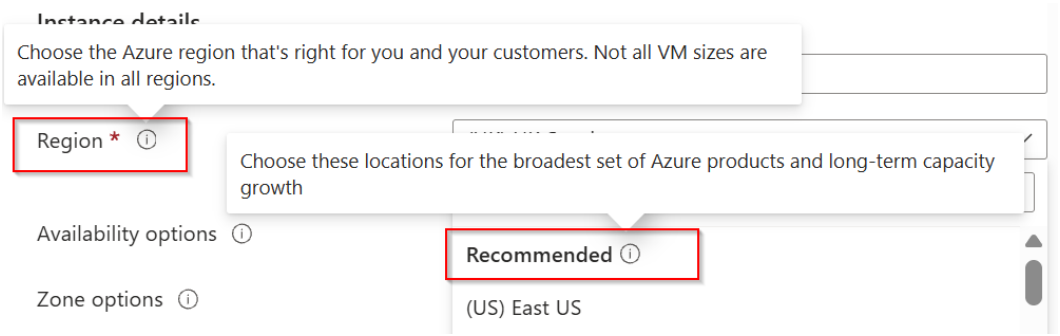


Figure 3.3: Azure region selection for creating resources

When creating Azure resources, regions are categorized under the **Recommended** or **Other** headings in the drop-down menu. The appropriate option should be selected for the scenario.

A **Recommended** region is designed to support availability zones now or planned for the future. Regions categorized as **Other** (not shown in *Figure 3.3* for simplicity) refer to the data pairing of regions to provide data residency boundaries and low latency for DR purposes. These are regions that do not support AZs.

Once the appropriate option is selected, a resource, such as a VM, is created in any of the data centers in that region. These resources are called **regional services** as they are dependent on being co-located in that specific region's data centers.

Some Azure services are designed to be **globally distributed** and are not deployed "into" a single region in the same way as a VM or storage account. These services still have configuration and metadata you manage, but the service itself operates across Microsoft's global infrastructure.

Some examples of non-regional services are **Traffic Manager** and **Front Door**. In contrast, **DNS** and **Microsoft Entra Domain Services** (formerly **Azure Active Directory Domain Services**) are **geo-based** to ensure data is kept within that geography.

When planning to create resources between regions, it should be considered that any data transfer leaving a region or **egress** is billed; however, traffic "within" a region or **ingress** is free.

While regions address availability and compliance, some scenarios require compute to sit even closer to users, devices, or data sources; so, it is now time to learn about Azure Edge Zones.

Azure Edge Zones

Microsoft is extending the Azure platform to incorporate **edge computing**. Edge computing allows data processing and code to be executed closer to the end user when sensitivity to network latency is critical.

Azure Edge Zones extend Azure closer to where users, devices, or data are generated, which helps when latency, local processing, or connectivity constraints matter. They provide Azure-consistent deployment and management, but with compute placed outside traditional Azure data center regions—such as at edge locations, partner or carrier sites, or customer-controlled environments—depending on the Edge Zone model used. The common idea is the same: *run workloads closer to the point of use while keeping them aligned to Azure operations and governance*.

Figure 3.4 shows the positioning of Azure data center regions and Azure Edge Zones:

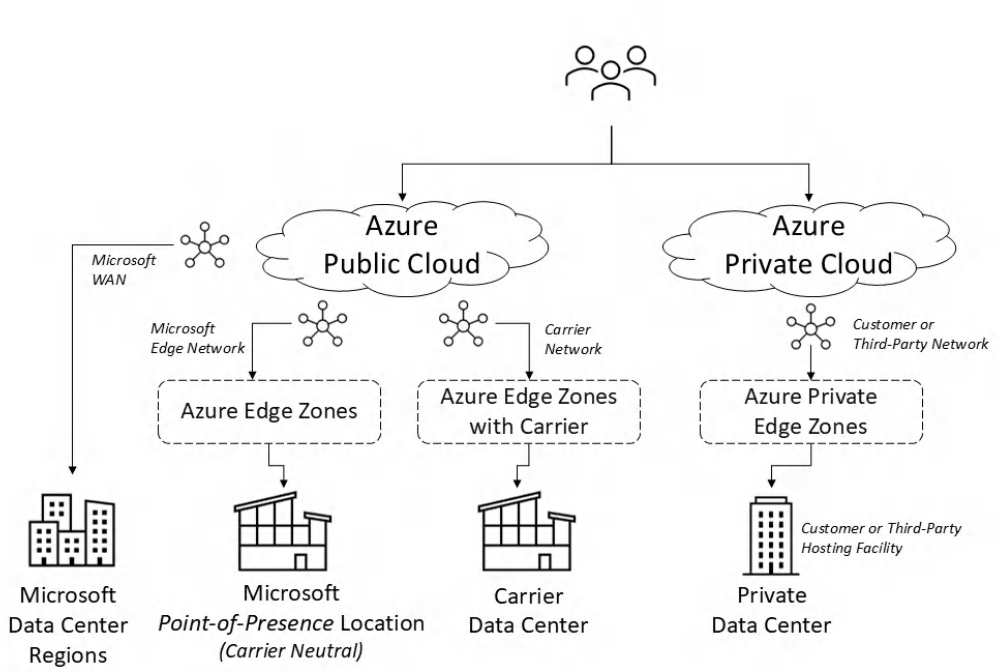


Figure 3.4: Azure Edge Zone relationships

The following describes the three Edge Zone scenarios depicted in *Figure 3.4*:

- **Azure Edge Zones**: These are still Azure public cloud (*shared*) platform resources (*Microsoft hardware*) within Microsoft's **Point-of-Presence** (**POP**) edge locations. These edge locations provide services such as ExpressRoute, **Content Delivery Network** (**CDN**), and the Azure Front Door service. These zones are connected to Microsoft's global network.

- **Azure Edge Zones with the carrier**: These are still Azure public cloud (*shared*) platform resources (*Microsoft hardware*), but are now shipped to exist within the carrier's data center locations. These zones are part of the carrier's global network.
- **Azure private Edge Zones**: These are a part of the Azure Local (*private*) cloud platform (*customer or third-party hardware*) within the customer's location or a third-party hosting provider. These zones are part of the customer or hosting provider's private network and not part of a carrier or Microsoft network. These zones can be connected to Microsoft data center regions using a VPN or ExpressRoute to establish hybrid connectivity if required.

With your knowledge of the concepts of Azure Edge Zones, you can now explore Azure sovereign regions.

Azure sovereign regions

Azure provides what is referred to as **sovereign regions**, which support greater compliance for specific markets.

These are isolated cloud-platform instances that run dedicated hardware and networks separate from the global Azure public cloud, as shown in *Figure 3.5*:

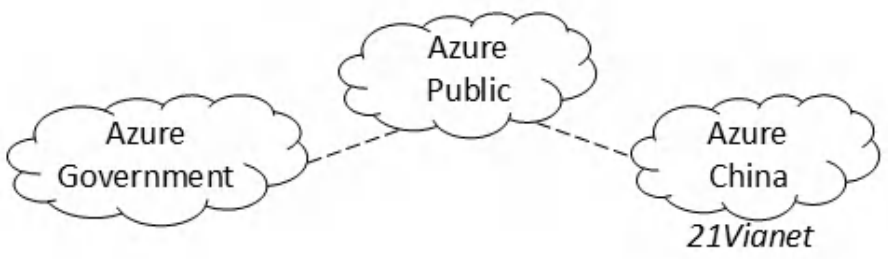


Figure 3.5: Provided Azure sovereign regions

Azure Public represents the global commercial cloud, whereas **Azure Government** and **Azure China** (**21Vianet**) operate as separate sovereign instances with independent infrastructure, management boundaries, and compliance boundaries.

The sovereign region platforms also have their own portals with different URLs and different service endpoints in the DNS. Here are some examples of URLs and endpoints to connect to in the DNS:

- **Azure Government**: This is a separate instance of the Azure platform operated by Microsoft. It is solely used by US government bodies (*and partners*):
    - The Azure portal can be accessed via a dedicated URL: `https://portal.azure.us`
    - The service endpoints to connect to in DNS are in the form of `*.azurewebsites.us`

> **Note**
>
> You can find more information on Azure Government at `https://azure.microsoft.com/en-gb/explore/global-infrastructure/government/`.

- **Azure China (21Vianet)**: This is a separate instance of the Azure platform operated by 21Vianet. It is for compliance with Chinese government regulations:
    - The Azure portal can be accessed via a dedicated URL: `https://portal.azure.cn`
    - The service endpoints to connect to in DNS are in the form of `*.chinacloudsites.cn`

> **Note**
>
> You can find more information on Azure China (21Vianet) at `https://www.azure.cn`.

Now that the structural boundaries of Azure infrastructure—regions, geographies, edge connectivity, and sovereign clouds—have been established, the focus shifts to how these boundaries influence availability design and risk mitigation strategies.

Availability components

Building on the availability concepts introduced in the cloud operations model, Azure provides a set of architectural components that allow availability to be designed intentionally rather than assumed. You should think back to the cloud computing services models of **Infrastructure as a Service** (**IaaS**), **Platform as a Service** (**PaaS**), **Serverless/Functions as a Service** (**FaaS**), and **Software as a Service** (**SaaS**), with the same principles and approach being applied to responsibility as for the *availability* of resources. This means you should consider how you will make data, app, compute, and other services available in the case of a failure at the app or VM, hardware, data center, or region level.

Depending on the service, Microsoft is responsible for providing a range of availability options. However, you are responsible for implementing them to ensure you have designed an availability strategy that meets your **service-level agreement** (**SLA**) requirements. Microsoft's SLAs for its services are based on financial compensation when the defined SLA is not met; however, this is not paid automatically and must be claimed by the customer.

> **Note**
>
> Microsoft SLAs for all services can be found at `https://www.microsoft.com/licensing/docs/view/Service-Level-Agreements-SLA-for-Online-Services`.

Figure 3.6 summarizes these components by looking at VMs and storage resources. You will see how you can increase the SLA and durability by using the following components:

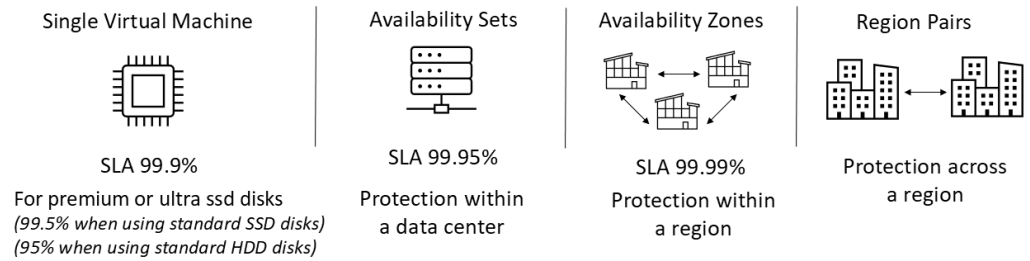


Figure 3.6: Azure availability components relationship

Figure 3.6 illustrates how Azure availability options increase resilience by expanding the scope of protection—from a single VM within one data center, to availability sets within a data center, to availability zones across physically separate facilities within a region, and finally to region pairs that provide protection at the regional level. Each component represents a broader fault domain boundary and a corresponding increase in SLA and durability.

An availability solution is based on the resources to be protected, the nature of the failure to protect the resources, and the SLA requirements.

Here are the *availability components* that Azure can provide to build a solution. The following components can be considered as building blocks to use as required:

- Within a data center component:
    - **Example failures that could occur**: Hardware rack failure, power, or network
    - **Availability component**: Availability set in the case of a VM, or locally redundant storage in the case of storage
- Within a region component:
    - **Example failures that could occur**: Data center failure, power, network, or cooling
    - **Availability component**: The availability zone in the case of a VM, or zone-redundant storage in the case of storage; synchronous replication is used
- Across a region component:
    - **Example failures that could occur**: Entire region failure—multiple data centers in a region suffer failures such as power, network, or cooling
    - **Availability component**: Azure Site Recovery in the case of a VM, or in the case of storage, geo-redundancy with asynchronous replication

Consider a business-critical web application. A single VM may meet basic SLA requirements, but placing the workload in an availability set protects against rack-level failure within a data center. Deploying across availability zones protects against data center failure within a region. For mission-critical scenarios requiring regional resilience, replication to a paired region using Azure Site Recovery or geo-redundant storage ensures continuity even in the event of a regional outage. The selected combination reflects the organization's risk tolerance and SLA objectives.

This leads us to how availability choices are combined into an overall availability model.

Adopting an availability model

Much like in the security area, where you looked at a **risk model** and you took a **defense-in-depth** (**DiD**) approach, you should consider taking a similar approach to implementing **availability** in a solution.

You should know the possible failures at the level you are concerned with, determine the **impact** and **probability**, and then implement the most appropriate solution. This availability model should be driven by determining your required SLA; that is, the uptime required for a

resource. The SLA of a resource can be increased by combining different availability components.

The first building block in this availability model is the availability set.

Availability sets

Availability sets are logical groupings of VMs designed to improve availability within a single data center and provide the VMs with a 99.95% SLA. Availability sets deploy VMs across different hardware instances through what is known as a **fault domain**, meaning the VMs are placed on hardware that is in *physically separated racks* when they are created. If there is a *within-rack failure*, then not all VMs are impacted; only a subset will be impacted, as you have *spread your eggs between several baskets*. However, without availability sets, you have *kept all your eggs in the same basket*.

Without VMs being part of an availability set, the best SLA for a single VM will be 99.9% when Ultra or Premium SSD disks are used; this drops to 99% when Standard SSD disks are used and 95% when a Standard HDD is used.

If a VM is not deployed in an availability set, the highest SLA is 99.9% when Ultra or Premium SSD disks are used; this decreases to 99% with Standard SSD disks and 95% with Standard HDD disks.

These SLAs equate to the following figures:

- 99.5% availability results in a period of allowed downtime/unavailability of 3h 37m 21s
- 99.9% availability results in a period of allowed downtime/unavailability of 43m 28s
- 99.99% availability results in a period of allowed downtime/unavailability of 4m 21s

Figure 3.7 helps to visualize resource placement in an availability set:

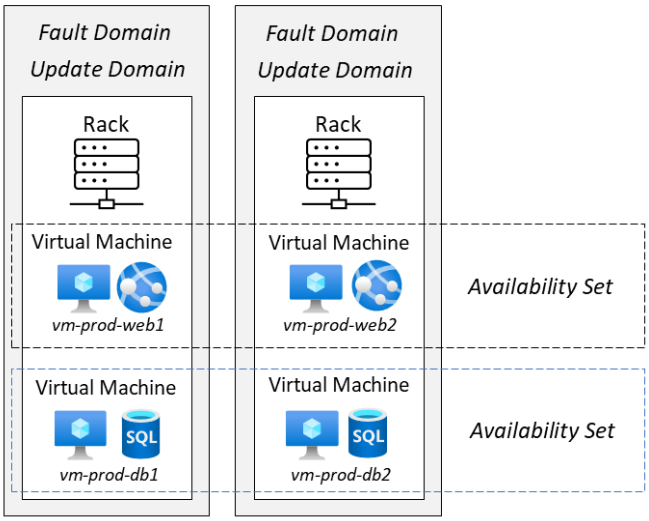


Figure 3.7: Providing Azure availability set functionality in a solution

Figure 3.7 illustrates how VMs within an availability set are distributed across separate fault domains and update domains. Each fault domain represents a physically separate infrastructure, such as distinct racks with independent power and networking. Update domains group VMs for coordinated maintenance, ensuring that only one subset is updated or restarted at a time. By spreading workloads across these boundaries, an availability set reduces the impact of both hardware failures and planned maintenance events.

Availability sets also act as an update patching boundary. An update domain is automatically assigned to all VMs that are part of an availability set. Updates to one set of VMs are isolated from other VMs to ensure that only one update domain gets updated at a time. If an update impacted the service of VMs in one update domain, that update would not be applied to other update domains.

Some points to consider regarding availability sets are as follows:

- Availability sets are not billable items.
- A VM is part of a single availability set.
- VMs can only be added to an availability set when created. It is not possible to add to an availability set or change the availability set after VMs have been created. In such cases, you would need to delete the VM and redeploy it, but this time as part of an availability set. Therefore, planning is critical for this purpose.

- You might consider deploying a single VM from the start as part of an availability set. While it will not help meet the availability set SLA, as that requires at least two instances or more, it would prepare your workload for the future without needing to redeploy.
- You will need to create the VM in the same resource group as the availability set to be able to add the VM to the availability set.
- Three fault domains and five update domains (a maximum of 20 can be configured) are part of an availability set.
- The assignment is carried out automatically and distributed sequentially across fault domains—for example, `vm1` in fault domain *#1*, `vm2` in fault domain *#2*, and `vm3` in fault domain *#3*. Because only three fault domains exist in a standard availability set, `vm4` would then be assigned back to fault domain *#1*.

Fault and update domains explain how availability sets isolate failures and maintenance events.

Availability sets improve resilience by distributing VMs across distinct failure and maintenance boundaries. These boundaries are implemented through fault domains and update domains, which define how infrastructure is physically separated and how planned maintenance is sequenced. To understand how availability sets achieve isolation, it is necessary to examine these domains more closely.

Fault domains

A fault domain is a group of resources in a data center rack that share the same power and network. When a failure occurs in a fault domain, all resources in that fault domain become unavailable.

Figure 3.8 shows each web server and database server in a separate fault domain:

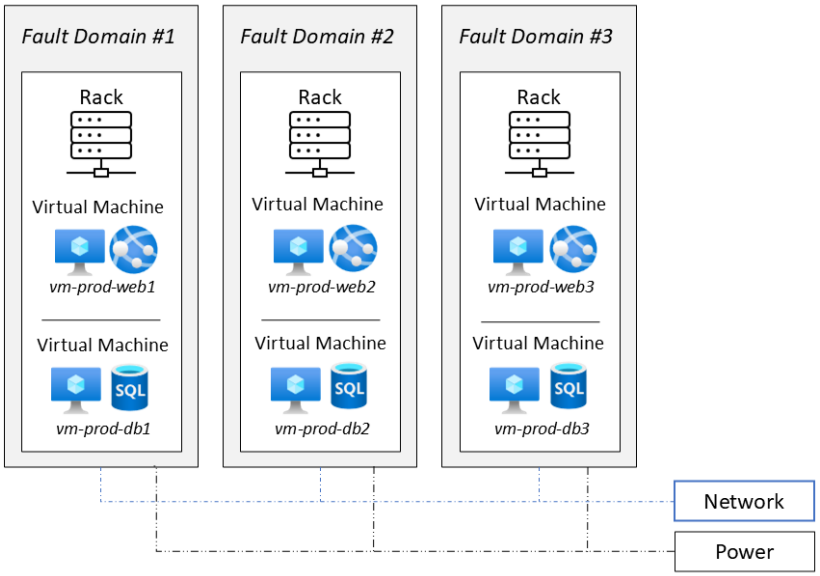


Figure 3.8: Providing availability set fault domain functionality in a solution

A failure can occur in one of the fault domains, but only one web server and one database server will be offline. As discussed earlier, this is the approach of *spreading your eggs across different baskets*. If the servers shared the same fault domain, the failure would affect all of them.

While fault domains protect against physical infrastructure failures, update domains address the operational impact of planned maintenance.

Update domains

Update domains are similar to fault domains but relate to *patching schedules*. You should put each web server in its own update domain so that they are rebooted independently and there is always a web server available to handle requests. The same approach should be taken for the database servers so that a database server is always available to handle requests from the web servers. This precludes both database servers from being rebooted simultaneously and potentially from both being unavailable.

Figure 3.9 helps to visualize the assignment of VMs across update domains:

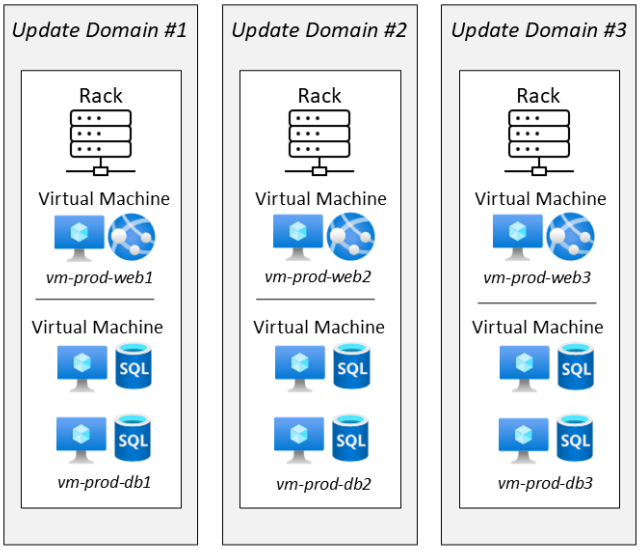


Figure 3.9: Providing availability set update domain functionality in a solution

In a nutshell, all VMs in the same update domain will receive updates and be rebooted simultaneously.

The key takeaway from this section is not to put two web servers, for example, in the same update domain, as both will be rebooted simultaneously.

Availability zones extend these availability concepts beyond a single data center.

Availability zones

Some regions are further divided into availability zones, although they are not available in all regions, and not all resources support availability zones. The purpose of an availability zone is to provide redundancy within a region; that is, to protect against a *data center failure* so that a failure related to something such as power, cooling, or connectivity does not impact the operations of any resources running in a single data center.

Note

Availability zones protect resources from a data center failure, not from a region failure. The scope of availability sets is to protect a VM from an in-rack failure, for example, within a data center.

Figure 3.10 outlines the Azure availability zone topology:

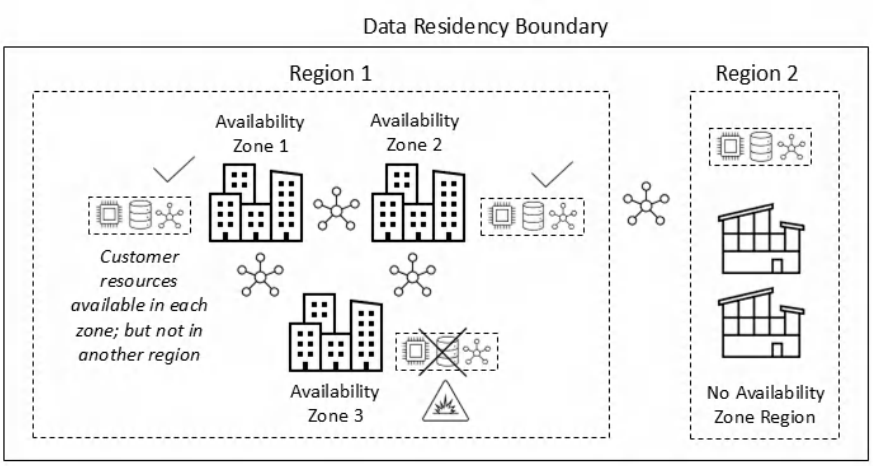


Figure 3.10: Azure availability zones providing redundancy within a region

Availability zones are designed so that each zone is protected by independent power, cooling, and networking infrastructure, helping isolate zone-level failures within a region. When resources are deployed across multiple availability zones, an outage in one zone does not inherently affect resource availability in the remaining zone or zones.

When VMs are deployed across two or more availability zones using supported configurations, Microsoft provides a higher zonal infrastructure SLA. This SLA reflects the resilience of the underlying platform, while application-level availability still depends on how the workload is architected across zones.

For VMs, availability zones enable VM instances to be deployed across physically separate facilities within the same region, reducing the impact of a single data center failure.

Microsoft provides the zonal building blocks, but you design the solution so that it can continue operating when one zone becomes unavailable—typically by deploying multiple VM instances across zones and using load balancing and zone-aware architecture choices. Availability-zone SLAs and requirements vary by service and configuration, so you should always validate what the SLA applies to for the specific design you choose.

Without availability zones (or another redundancy design), a single-instance resource running in one data center can be unavailable until that data center is restored. Availability zones keep resources, connectivity, and traffic within the primary region while hosting them across physically isolated data center locations inside that region.

To protect against an entire region becoming unavailable, resources can be replicated to a secondary region as part of a cross-region DR strategy. In some scenarios, this may not be viable due to compliance, latency, connectivity constraints, or dependencies such as ExpressRoute or VPN designs that are not configured for regional failover.

Azure resources are placed into one of three categories, as outlined here:

- **Zonal services**: These provide the ability to select an availability zone for resources. For instance, resources can be pinned to a specific zone based on your needs, performance, or latency.
- **Zone-redundant services**: The replication of resources across zones is automatic, and you cannot define replication settings governing how resources are distributed across zones.
- **Non-regional services**: Services are available in all geographies and are not affected by zone-wide or region-wide outages.

Consider an e-commerce platform running in a region that supports availability zones. The web tier consists of VMs deployed across two availability zones to achieve zonal resilience and qualify for a higher-infrastructure SLA. The database tier uses a zone-redundant managed service, so replication across zones is handled automatically. However, a latency-sensitive analytics component requires low network latency between compute and storage, so those resources are pinned within a single availability zone and placed in a proximity placement group (covered in the following section). This design balances resilience and latency by deliberately mixing zonal, zone-redundant, and zone-specific services.

While availability zones focus on resilience and fault isolation, some designs are driven primarily by latency, which introduces a different placement consideration.

Proximity placement groups

Proximity placement groups are a logical entity and an architectural component that should be considered in any solution design where low latency between Azure infrastructure and compute resources is required. Proximity placement groups ensure that compute resources are physically adjacent and collocated within the same physical data center and not across data centers.

Proximity placement groups are important where *latency* is considered. Placing resources closer within a single availability zone is possible, but the challenge is that an availability zone can expand to span multiple physical data centers. VMs that are part of the web frontend tier for the solution may be in one data center, but the database tier is in a different data center's rack; the latency impact here should be readily apparent. VM-accelerated networking was one solution to latency; however, proximity placement groups alleviate this.

Figure 3.11 outlines the placement of VMs without a proximity placement group:

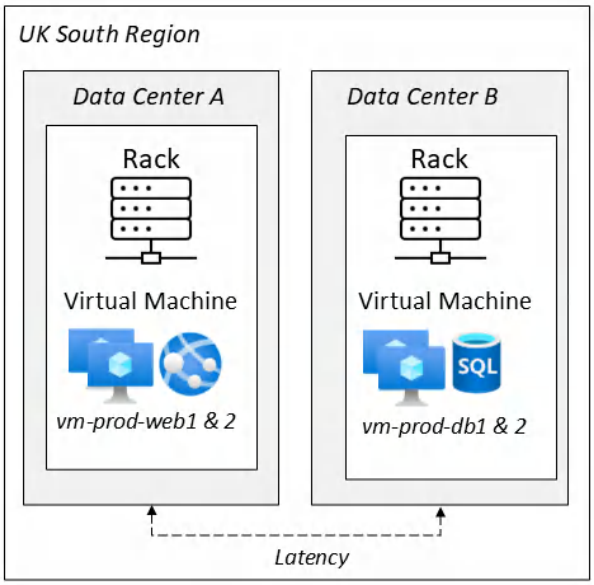


Figure 3.11: Providing VM placement without proximity placement groups in a solution

In *Figure 3.11*, two VMs that need to communicate with each other could be placed by Microsoft in different data centers several kilometers apart. This may cause issues due to latency.

Figure 3.12 outlines the Azure proximity placement groups topology and how this can reduce latency. Observe that the VMs are now located within the same data center, and latency is now reduced to within the same rack. This optimization introduces an important trade-off between latency and availability that must be addressed deliberately.

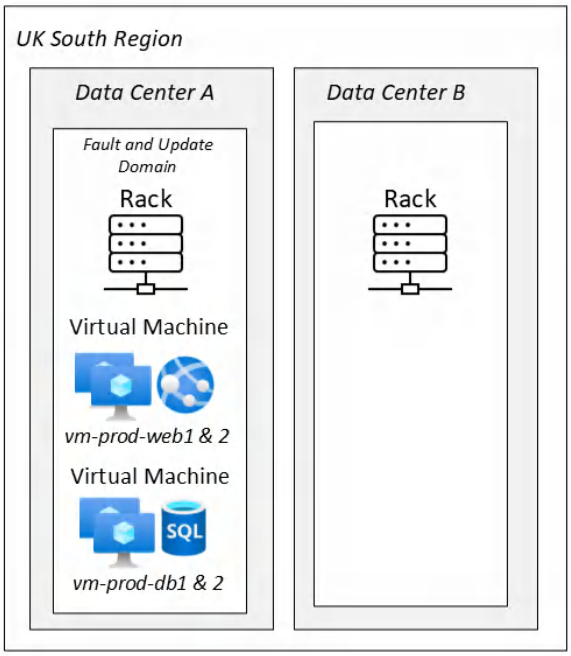


Figure 3.12: Providing VM placement with proximity placement groups in a solution

Figure 3.12, while solving a latency problem, introduces an availability issue. Now, as both VMs are located in the same rack, they also share the same fault and update domains. This issue can be resolved using an availability set to place each VM in a different rack with its own fault and update domains. *Figure 3.13* outlines this concept:

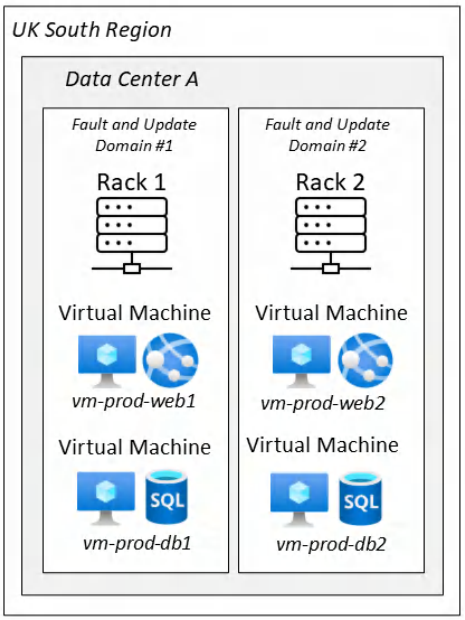


Figure 3.13: Providing proximity placement groups and availability sets in a solution

Here are a few things to note concerning proximity placement groups:

- They are Azure resources and need to be created before they can be used
- They can be used with VMs, Virtual Machine Scale Sets, and availability sets
- When creating a compute resource, specify the proximity placement group previously created
- To move existing compute resources into a proximity placement group, the resource will need to be stopped (deallocated)
- They are set at the resource level, not the individual VM level, for availability and Virtual Machine Scale Sets
- A workload such as SAP HANA with SAP NetWeaver is a good example of the importance of having the machines separated for HA/SLA, but close enough to not face latency issues in communication

Taken together, availability sets, fault and update domains, availability zones, and proximity placement groups illustrate that resilience and performance are architectural design choices, not automatic outcomes. Each decision influences SLA, fault isolation, latency, and operational behavior. Once these placement and resilience considerations are understood, the next

challenge is ensuring that resources are consistently organized, governed, and controlled as environments grow in size and complexity.

Azure resource management

It is important to ensure that any data or workloads running in a cloud computing environment are managed in a **governed**, **controlled**, **secured**, and **protected** manner, as they would be for a traditional computing model.

The foundation for all management within Azure is ARM, which is an **Application Programming Interface** (**API**) and the **management** and **deployment** service for Azure. All user and resource interactions pass through the **ARM API**.

You can think of the ARM API as the *brain* or *engine* of Azure. It provides a single consistent management layer and orchestration engine that processes all requests for every interaction with the Azure platform. Be it the portal, **command-line interfaces** (**CLIs**), templates, Microsoft utilities, or third-party software, they all call the ARM API.

All deployable Azure services and configuration objects discussed so far, such as VMs, availability sets, load balancers, storage accounts, and proximity placement groups, are created, managed, and governed through ARM.

Figure 3.14 illustrates this:

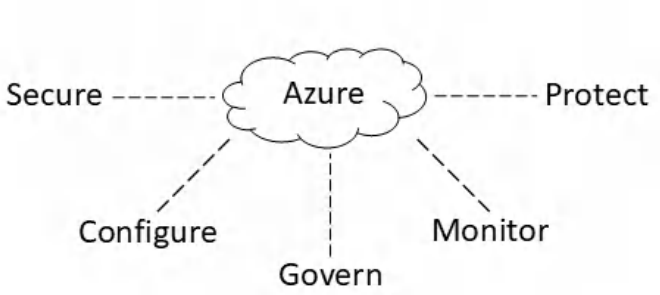


Figure 3.14: Azure resource management

Azure resource management solutions should include the following:

- **Protection**: This is managed through backup and DR
- **Security**: This is managed through threat protection and security posture management
- **Configuration**: This is managed through automation, scripting, and update management

- **Governance**: This is managed through access control, compliance, and cost management
- **Monitoring**: This is managed through the collection of security incidents, events, resource health, performance metrics, logs, and diagnostics

Note

Further knowledge

Key takeaways from this section are that governance and planning are essential stages when creating resources in Azure. Doing this correctly can prevent re-work and potentially tearing down aspects that may have been implemented without fully appreciating what should be considered and grasping all options and their benefits and limitations on the desired outcome.

A development team deploys multiple VMs directly into a subscription without applying naming standards, tagging, **Role-Based Access Control** (**RBAC**) boundaries, or policy controls. Because all resources are created through ARM, the deployments succeed. However, months later, the organization cannot easily identify ownership, cost allocation, or compliance posture. Remediation requires re-tagging, role reassignment, and, in some cases, redeployment into correctly structured resource groups. ARM enables deployment, but governance planning determines whether that deployment is sustainable.

These governance principles are applied in Azure through a defined set of management scopes.

Azure management scopes

Azure provides the following four management scopes so that RBAC and Azure Policy can be targeted at the following levels:

- Management group level
- Subscription level
- Resource group level
- Resource level

Figure 3.15 outlines these four management scopes and their hierarchy:

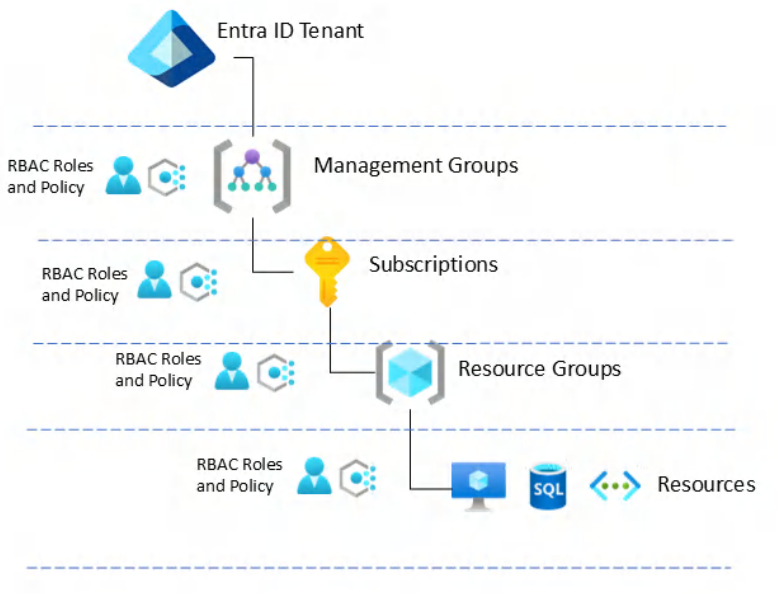


Figure 3.15: Azure management scopes relationship

The hierarchy begins at the Microsoft Entra ID tenant, which forms the top-level identity boundary for the organization. Beneath the tenant are management groups, which allow governance controls such as RBAC and Azure Policy to be applied across multiple subscriptions. Subscriptions sit under management groups and act as billing and isolation boundaries. Within each subscription are resource groups, which logically organize related resources. At the lowest level are the individual Azure resources themselves. RBAC roles and policy assignments can be applied at each level and are inherited downward through the hierarchy.

Azure management scopes define *where* access control and policies can be applied; management groups explain *how* those controls are organized and enforced across multiple subscriptions.

Azure management groups

Azure management groups are **logical containers** that group together Azure subscriptions. They can be considered a governance and management layer to implement access control and policies. Management groups are the most effective governance scope when multiple Azure subscriptions exist in a tenant.

However, there are some limitations of management groups to be aware of:

- Management groups can only contain other management groups and subscriptions. They cannot include resource groups or resources.
- The scope is per tenant and not across tenants.
- The parent management group cannot be deleted or moved. However, it can be renamed.
- Only one parent management group is supported, and only one parent hierarchy is a single tenant.
- They can support many child management groups (up to 1,000).
- A hierarchy of up to six levels is supported.

This hierarchy may be applied across **business units** (**BUs**), regions, or environments, wherever you may need to define billing or access control boundaries. In practice, planning is critical.

Figure 3.16 shows just one example of the hierarchy and the relationships between the different subscription types within a business:

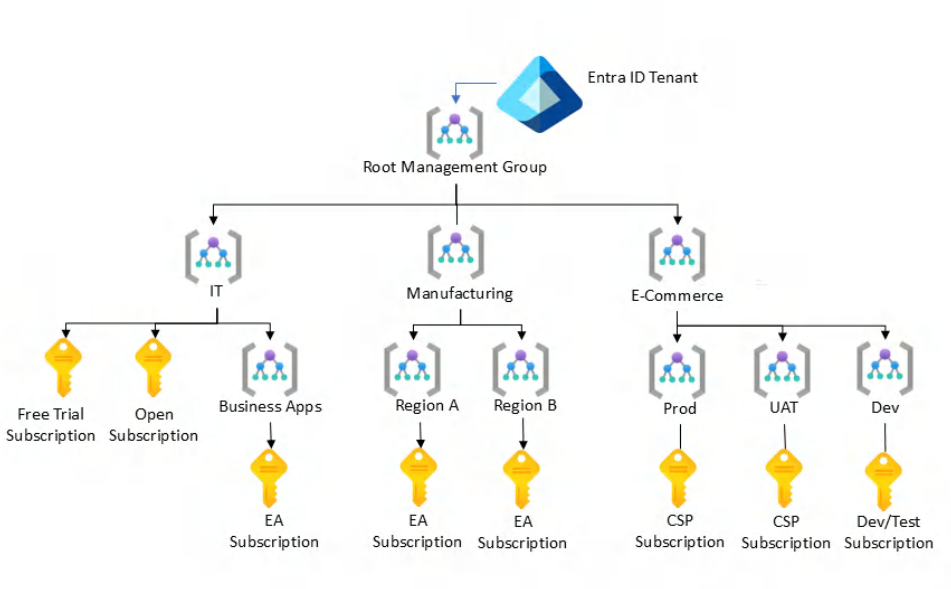


Figure 3.16: Azure management group and subscription relationships

Figure 3.16 illustrates a typical enterprise hierarchy within a single Microsoft Entra ID tenant. At the top is the tenant boundary, under which a **root management group** is created. Beneath the root are **child management groups** aligned to organizational structures such as IT,

manufacturing, and e-commerce. Each of these management groups contains **subscriptions** that represent workload types, regions, or environments—for example, **production** (**Prod**), **user acceptance testing** (**UAT**), **development** (**Dev**), or region-specific subscriptions such as **Region A** and **Region B**.

Different subscription types are also shown, such as **Enterprise Agreement** (**EA**), **Cloud Solution Provider** (**CSP**), **Dev/Test**, and **Free Trial** subscriptions. Regardless of subscription type, each subscription is placed under a management group so that governance policies and RBAC assignments can be applied consistently. Controls assigned at a higher management group are inherited by all child management groups and subscriptions beneath it.

While management groups define how governance is structured across multiple subscriptions, the subscription itself is the fundamental billing and isolation boundary within Azure.

Azure subscriptions

Azure subscriptions can be considered the *logical containers* of Azure resources that share the same *billing* and *access control* boundaries. An Azure subscription is a billing mechanism for the Azure resources you consume within a tenancy. It can be likened to a *bar tab*—a method of paying for everything you have consumed, all together and itemized at the end of the night, instead of paying for each item individually every time you order a drink. To continue the analogy, when a bar tab is opened, a payment method (such as a credit card) must be provided so that there is one bill to settle for everything put on the tab at the end of the night.

Every time a resource is created within Azure, an Azure subscription must be selected for the allocation of the consumption of resources (or "tab" in the previous example). This means the Azure subscription(s) are created first before creating any resources. At the end of the month, all Azure resources created in that subscription will be billed in a single itemized invoice.

Figure 3.17 outlines Azure subscriptions:

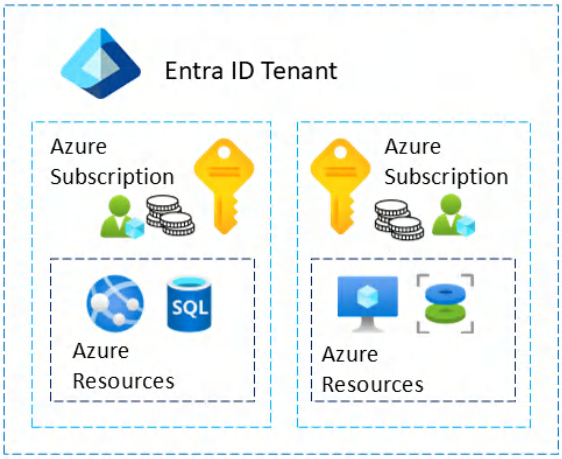


Figure 3.17: Relationship between Azure subscriptions and resources within a tenant

An Azure subscription should first be created in a new or existing Entra ID tenancy, as this is the foundation and starting point. Management groups and resource groups are then planned as needed.

Subscriptions cannot be merged; however, resources can be moved into another subscription. A subscription that no longer holds any resources after the move can be deleted or allowed to expire. When a subscription is deleted, all resources within the subscription are deleted. However, when a subscription expires, the resources remain. In either case, the tenancy remains in place and is not removed. The billing owner of a subscription can also be changed, much like a bar tab could be run up and transferred to somebody else to take it over and settle the tab.

When planning to start creating resources in Azure, you must first define access and billing control. Defining these will shape the subscription and resource group strategy. You should create a tenancy, subscription, and resource groups in that order. Planning is necessary for your subscriptions, and you need to determine whether multiple subscriptions will exist within a tenancy. You may wish to have one for production resources and another for development and testing purposes to isolate access to the resources through the subscription or to control who will receive an invoice for any resources created within each subscription. This will depend on your needs and organizational requirements.

Now, it is time to look at the relationship between subscriptions and tenants.

Relationship between subscriptions and tenants

As you saw in the last section, there are certain relationships between Azure subscriptions and Entra ID tenants. The relationships also extend to other entities, such as resource groups and Azure resources. There are existing relationships between users, groups, and apps, such as a Microsoft 365 subscription. Resource groups are logical containers for resources, and you will explore this later in this chapter.

Figure 3.18 outlines the relationship between these entities:

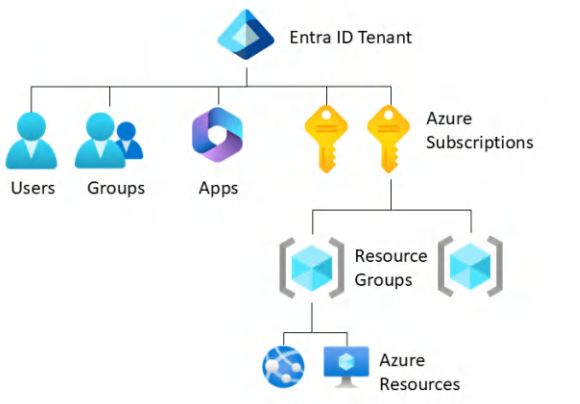


Figure 3.18: Azure subscription and tenant relationship

At a high level, the following relationships exist:

- **Entra ID tenant**: This contains users, groups, and Microsoft 365 subscriptions
- **Azure subscription**: This contains resource groups, which include resources

The following additional relationship rules apply:

- Each subscription in Azure can be associated with **only one** Entra ID tenant in a parent-and-child relationship, with the Entra ID tenant being the parent.
- A subscription can be moved to a different Entra ID tenant.
- The subscription billing owner can be changed, such as moving from being invoiced by Microsoft to being invoiced by a channel CSP.
- Each subscription can have multiple resource groups. Each resource group can be part of **only one** subscription.
- Each resource can be associated with **only one** resource group. In a parent-and-child relationship, the resource group is the parent.

- Each user can be associated with **only one** Entra ID tenant, with the Entra ID tenant being the parent.
- Each user can access **more than one** Entra ID tenant.
- Each group can be associated with **only one** Entra ID tenant, with the Entra ID tenant being the parent.
- Each **Microsoft 365 subscription** can be associated with **only one** Entra ID tenant (a domain), with the Entra ID tenant being the parent.

These relationships shape how subscriptions are used as both access control and billing boundaries, which leads us to Azure subscription management.

Multiple subscriptions

An organization is not limited to a single subscription. Resources can be consumed from multiple subscriptions within the same tenant or another tenant that the organization owns. Any additional subscriptions can be created for access control management or billing management purposes, as a subscription acts as a billing and resource management boundary.

There are subscription soft quota limits, which means you need to contact Microsoft support to increase resource quota limits through a service request. Typically, this request would be for a VM core count that has exceeded the value set for the number of cores in the subscription quota. The subscription quota would need to be increased to set the number of VM cores needed.

A subscription can be a boundary of access control. A subscription could be created for each environment, such as creating a subscription for UAT, production, or staging. Alternatively, a subscription could be created per team or project, as shown in *Figure 3.19* in this section.

This approach to access control ensures control of access to resources, with individuals or groups having the least amount of access they need to the minimum amount of resources. This is one of the core principles of governance and compliance, and through RBAC, the approach and principle of **least privilege** can be adopted. The subscription can also be used as a layer to enforce Azure Policy.

In addition to being used as a boundary of access control, a subscription can also be used as a boundary of **billing control**. This means costs incurred from Azure resources created can be allocated with a *charge-back* or *show-back* approach, where charge-back bills departments directly for their consumption, and show-back reports usage without formally billing.

The relevant entity can see precisely which environment, team, or project incurred the costs without all resource costs being aggregated into one subscription.

> **Note**
>
> Tags can be used to break down resources within a subscription. Since they are out of the scope of this section, they will be covered in *Chapter 10, Azure Governance and Compliance*.

Figure 3.19 outlines the relationships between multiple subscriptions, tenants, and resources.

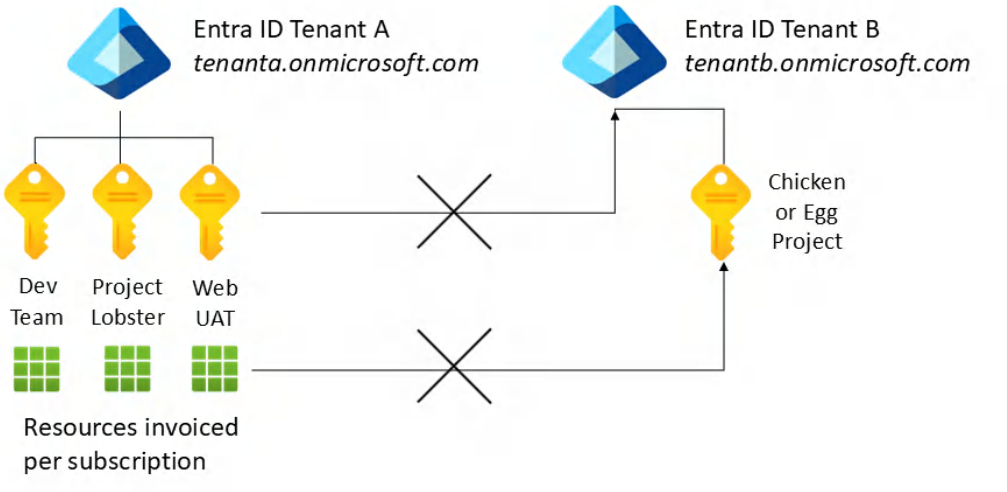


Figure 3.19: Multiple Azure subscriptions

Looking back to the "bar tab" analogy from the previous section, each resource can only be billed to one bar tab. However, resources can be moved between subscriptions so that another subscription can be used to pick up the bill (their own bar tab) for those resources consumed.

For instance, some resources that you no longer wish to be billed for under the existing resources' subscription can be moved to another subscription that may be more appropriate to pay the bill for those consumed resources. The resource move is a simple operation that can be done within the Azure portal.

You can see from the example shown in *Figure 3.19* that the three subscriptions associated with **Tenant A** cannot be associated with **Tenant B** simultaneously. These subscriptions can only have their own parent owner.

Likewise, resources can only exist within one subscription that acts as the billing mechanism for the consumption of those resources. The resources cannot be associated with another subscription within another tenancy simultaneously. They can move owners from

subscription to subscription within the same tenancy and from the parent subscription to a *different* subscription in a *different* tenancy.

You learned about Azure subscription management in this section, including reasons for and benefits of creating multiple Azure subscriptions in a tenancy. In the next section, you will explore more about ARM and the functionality it provides.

ARM

In a nutshell, **ARM** is the **deployment** and **management** service for Azure. It provides a consistent management and control layer for the creation, update, and deletion of resources as well as control access, application of policy, governance, and compliance.

Figure 3.20 outlines the ARM architecture:

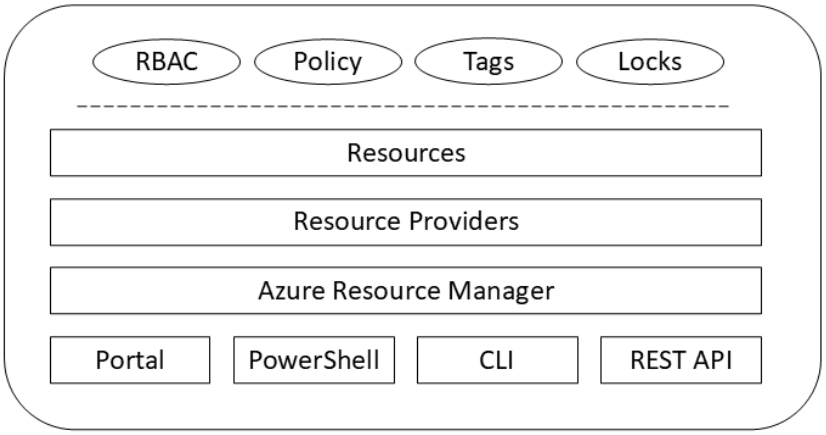


Figure 3.20: ARM architecture showing the relationship between its components

Figure 3.20 illustrates the layered architecture of ARM. At the lowest level, users interact with Azure through tools such as the Azure portal, PowerShell, the Azure CLI, or directly via the REST API. All of these interfaces communicate with the ARM control plane. ARM processes the request and routes it to the appropriate resource provider, which is responsible for managing a specific Azure service, such as compute, networking, or storage. Resource providers then create or manage underlying Azure resources. Cross-cutting governance controls—such as RBAC, Azure Policy, tags, and resource locks—are applied through ARM and enforced consistently across resources.

ARM provides the following functionality:

- Deployment, management, and monitoring as it pertains to resource grouping rather than individual resources

- Application of access control and policies at the resource-group level, inherited by all resources in that group
- Application of tags to resources for billing, logical grouping, and management
- Repeatable lifecycle deployments that result in a consistent state
- Use of deployment template files through **JavaScript Object Notation** (**JSON**) or **Bicep** to define the deployment of resources
- Creation of resource dependencies

As the name implies, ARM takes a resource-centric approach to every Azure object or entity classed as a resource.

The main component of the ARM logical architecture is the resource group. The resource group can then be a target for applying access control and policies instead of individual resources.

Here are some terms to know when working with ARM:

- **Resource**: In a nutshell, this is a billable item that is consumed—the lowest-level element that can be broken down in Azure. Resources can be thought of as the cells, molecules, or building blocks of all services and solutions for the Azure platform. Billing takes place on the resource level, and each resource will have an ID with a billing meter that calculates the amount consumed and the rate at which the customer should be billed. Billing meters are not used for Entra tenants, subscriptions, management groups, resource groups, and so on. These are logical entities and have no billing meters.
- **Resource group**: A logical container for resources. Resources that are to be managed together as a lifecycle are grouped to target access control and compliance through policies. Resources are grouped into resource groups in the most appropriate way for management purposes.
- **Resource provider**: These are not logical, but rather the actual service that contains and provides Azure resources. You could consider a book as a reading resource and the library as the provider.
- **ARM templates**: These are JSON or Bicep files that provide a repeatable and automated deployment methodology. These form the foundation of the **Infrastructure as Code** (**IaC**) methodology and use a **declarative** syntax instead of an **imperative** syntax, which is the approach with PowerShell.

With ARM established as the control plane that processes and governs all resource interactions, the next logical step is to examine resource groups, which define how those resources are organized and managed within a subscription.

Resource groups

Resource groups are one of the core foundational elements of the Azure platform and ARM. The Azure platform is built around resource-centric principles. Resource groups are **logical containers** for Azure resources and are used as a *management scope* for access management and policy.

The relationship between a resource group, subscription, and the resource itself is outlined in *Figure 3.21*:

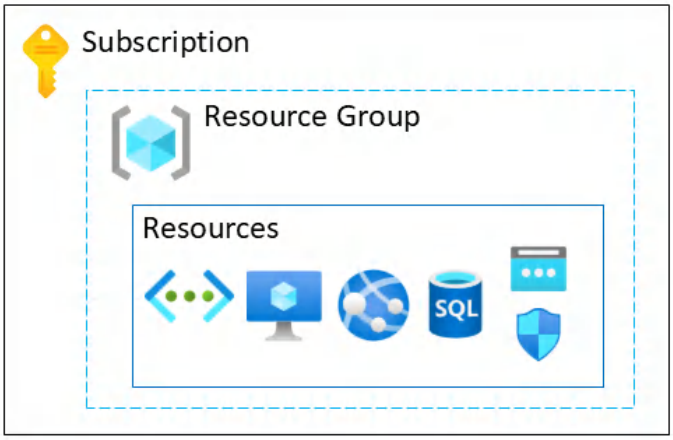


Figure 3.21: Azure resource group relationships

This section introduced the concept of resource groups. The key takeaway from this section is the flexibility that using resource groups provides, and how you can use resource groups to logically group resources in a way that means something to a given organization. For instance, it may be grouped by resource type, environment, BU, project, or location.

The following section looks at some of the characteristics of resource groups.

Resource group characteristics

The following are characteristics of resource groups:

- Resources must belong to a resource group and can only exist in one resource group. However, they can be moved between resource groups.
- Resources can interact with other resources in the same resource group, other resource groups, and subscriptions.

- Resources work at the data-plane level, while resource groups and subscriptions work at the management-plane level.
- Resource groups do not have to use the same region. They can include resources from other regions.
- Resource groups do not contain subscriptions, but subscriptions contain resource groups.
- Resource groups are not physical. They are logical entities and not billable items.
- Resource groups contain metadata about the resources they include.
- Resources inherit all permissions set at the resource-group level they belong to by default.
- When adding new resources to a resource group, they inherit those permissions and any access assignments.
- When moving resources, the resources lose permissions of the resource group they belonged to and inherit those of the new resource group they are moved to.
- If access and permissions are assigned at the resource-group level, all resources in that resource group can be managed.
- Deleting a resource group will remove all resources within that resource group. However, it would not delete the subscription or tenant.
- Since all resources are contained in the same resource group, it is easy to act on all resources with a single activity. All resources within the resource group inherit access assignments and policies set at the resource-group level.
- When assigning tags to a resource group, the resources in that resource group do not inherit those tags. You would have to individually apply tags to each resource in that group.

Now that you have explored the characteristics of resource groups, it is time to learn more about resource group logical organization.

Azure resource group organization

You must first define access and billing control when planning to create Azure resources. This will shape the **subscription** and **resource group** strategy. Here is the order in which these elements should be created:

- Tenancy
- Subscription
- Management groups
- Resource groups

A resource group is typically used to group all resources that share the same **lifecycle** or have some form of **dependency**, or the same **access control** or **management** requirements. Resources could be grouped by **environment**, **BU**, **region**, or another combination as appropriate.

Creating the appropriate resource group requires planning to find the optimal way to organize all Azure resources logically. This will become increasingly important as the environment grows to hundreds or thousands of resources. It is best to plan a strategy for both the logical grouping of resources and a naming convention to make identifying resources easier for management purposes.

This resource group organization is entirely subjective, as it all depends on the organization's decision on how resource groups will be used. Some may like to see **location** grouping, and some **resource-type** groupings, such as networks or VMs.

Figure 3.22 outlines different methods that could be used:

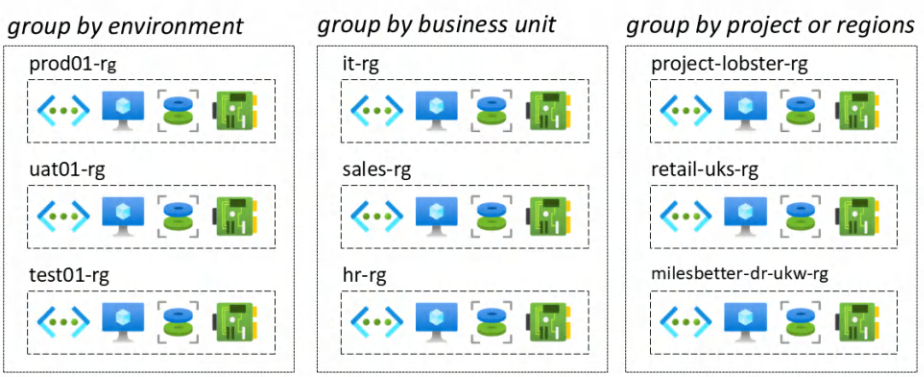


Figure 3.22: Azure resource group organization showing grouping methodologies

Figure 3.22 illustrates three common approaches to organizing resource groups. Resources can be grouped by environment (for example, production, UAT, and test), which supports lifecycle isolation and access separation. They can be grouped by BU, such as IT, Sales, or HR, aligning resources to organizational ownership and cost accountability. Alternatively, they can be grouped by project or geographic region, which is useful for regional compliance, DR planning, or workload-specific management. The appropriate structure depends on governance, billing, and operational requirements.

When managing resource groups as part of day-to-day operations, you can delete individual resources as well as the resource group and all resources associated with it. Deleting at the resource-group level is useful in cases when several resources in the resource group are no

longer needed and allows a bulk delete, rather than deleting each resource. This can also be risky as there may be resources in a resource group that have dependencies and are required not to be deleted.

To avoid this, governance controls can be put in place—that is, removing the ability to delete resource groups through RBAC. In addition, **locks** can also be used to prevent deletion, which will be covered in *Chapter 10*, *Azure Governance and Compliance*.

You learned in this section all about the components of ARM, which concludes this chapter on Azure core architectural components.

Summary

This chapter covered the *Describe Azure architecture and services* skills area of the *AZ-900 Azure Fundamentals* exam.

The chapter examined Azure's core architectural components from both a physical and logical perspective, including data centers, regions, global networking, and availability constructs, alongside the mechanisms used to organize and manage resources at scale.

You learned how Azure organizes resources through a hierarchical structure of management groups, subscriptions, resource groups, and individual resources. You saw how ARM acts as the control plane that enforces governance, applies RBAC and policies, and ensures consistent deployment across environments. You also explored how billing, access control, and lifecycle management are structured to provide clarity, accountability, and operational control at scale.

The next chapter introduces **Azure Compute Services**, continuing the progression through the *Azure Core Resources* skill area and building on architectural foundations established here.

Exam Readiness Drill – Chapter Review Questions

Apart from mastering key concepts, strong test-taking skills under time pressure are essential for acing your certification exam. That's why developing these abilities early in your learning journey is critical.

Exam readiness drills, using the free online practice resources provided with this book, help you progressively improve your time management and test-taking skills while reinforcing the key concepts you've learned.

HOW TO GET STARTED

- Open the link or scan the QR code at the bottom of this page

- If you have unlocked the practice resources already, log in to your registered account. If you haven't, follow the instructions in *Free Benefits with Your Book* section in this book's *Preface* and come back to this page.
- Once you log in, click the START button to start a quiz
- We recommend attempting a quiz multiple times till you're able to answer most of the questions correctly and well within the time limit.
- You can use the following practice template to help you plan your attempts:

Working On Accuracy		
Attempt	**Target**	**Time Limit**
Attempt 1	40% or more	Till the timer runs out
Attempt 2	60% or more	Till the timer runs out
Attempt 3	75% or more	Till the timer runs out
Working On Timing		
Attempt 4	75% or more	1 minute before time limit
Attempt 5	75% or more	2 minutes before time limit
Attempt 6	75% or more	3 minutes before time limit

The above drill is just an example. Design your drills based on your own goals and make the most out of the online quizzes accompanying this book.

First time accessing the online resources?

You need to unlock the online practice resources to access the link below. Check the *Free Benefits with Your Book* section in the *Preface* for detailed instructions on how to do this. You only need to unlock the resources once.

Open Quiz `https://packt.link/AZ-9003ech3` or scan the QR code

Join the CloudPro Newsletter with 44000+ Subscribers

Want to know what's happening in cloud computing, DevOps, IT administration, networking, and more? Scan the QR code to subscribe to **CloudPro**, our weekly newsletter for 44,000+ tech professionals who want to stay informed and ahead of the curve.

`https://packt.link/cloudpro`

4

Azure Compute Services

Chapter 3, *Azure Core Architectural Components*, introduced the Azure platform's core components from both physical and logical perspectives. This chapter builds on that foundation by exploring **Azure Compute Services**—one of the core building blocks used to run applications, processes, and workloads in Azure.

The chapter primarily aligns with the *Describe Azure Architecture and Services* module in the *Skills Measured* section of the *AZ-900 Azure Fundamentals* exam. Understanding these services is central to progressing in Azure. Compute underpins nearly every workload you will deploy, and selecting the correct execution model affects scalability, management responsibility, and cost. The sections that follow will help you recognize the differences between these models and apply them appropriately as your Azure knowledge develops.

By the end of this chapter, you will be able to confidently answer questions on the following topics:

- Azure Compute Services overview
- Virtual machines in Azure
- Containers in Azure
- Azure App Service
- Azure Functions
- Azure Virtual Desktop

The concepts explored here are not only exam-relevant but also foundational to understanding how Azure workloads are designed and operated in practice.

Azure Compute Services overview

Azure Compute Services is one of the core Azure service categories, and the focus of this chapter. These services utilize resources that provide the capability to run operating systems, host applications, execute code, and process instructions and data. The term *compute* can be described simply as a platform to execute code; it runs your software and processes your data.

Every workload deployed—whether it is a website, a business application, a virtual desktop, or a process—relies on one or more compute services to function. In practice, Azure compute choices are less about whether code can run and more about the operating model—specifically, how much of the platform Microsoft manages versus how much the customer manages. In modern cloud environments, organizations often standardize on managed platform services for day-to-day workloads, using **virtual machines (VMs)** mainly when full operating system control, legacy dependencies, or specialized configurations are required. Rather than offering a single compute resources model, Azure provides a range of compute services that differ in how much control and responsibility you retain versus how much it is managed by Microsoft.

Some services give you full control over the operating system and runtime environment, while others abstract these details so that you can focus only on your application or code. Selecting a compute service is therefore primarily a decision about the operational model rather than raw compute capability. When choosing a compute service, you should consider factors such as the level of control required, how predictable the workload demand is, how quickly the service needs to scale, and how much operational management is acceptable.

Compute services are designed to integrate with storage, networking, identity, monitoring, and security services to form a complete end-to-end Azure business solution. The following sections will review the following compute resources outlined in the AZ-900 exam's *Skills Measured* section:

- Virtual machines and scale sets
- Containers
- App services
- Serverless
- Azure Virtual Desktop

The chapter begins with virtual machines, the most fundamental and flexible compute service in Azure.

Virtual machines in Azure

VMs are a core compute service outlined in the AZ-900 exam's *Skills Measured* section: *Describe Azure Compute and Networking Services*. VMs are **Infrastructure-as-a-Service** (**IaaS**) compute service resources that form a common building block of Azure solutions. VMs virtualize physical CPU and memory resources, providing a software-based emulation of hardware computers.

VMs are the most appropriate compute service in the following scenarios:

- When there is a need to provide complete control of the **operating system** (**OS**) and any software installed.
- When there are customization requirements, such as customizing the OS, the software/applications, and their runtimes. Custom and Azure Marketplace images can also be used; the OS can be a Windows or Linux distribution.
- When the workload cannot be containerized.
- When there is a need to extend on-premises computing capacity, perhaps for development and testing, disaster recovery, and business continuity scenarios. Examples can be VMs connected to an organization's network using a VPN over the internet or the Microsoft ExpressRoute service, a private network connection that bypasses the internet for better performance and low-latency requirements. Alternatively, Azure VMs can be isolated and not connected to on-premises environments.

Because VMs are an IaaS offering, customers retain responsibility for several operational tasks under the shared responsibility model. This responsibility means that these are tasks you need to carry out as you would for on-premises compute resources, such as configuring and patching the OS, installing and configuring any software, creating backups, securing the VM with network security controls, and managing/securing user account access. As a result, Microsoft does not automatically patch the guest OS or back up data inside VMs unless additional services are explicitly configured.

Microsoft Azure, as a platform, also provides optional management capabilities to help standardize patching, monitoring, and policy-based governance at scale, but the key principle remains the same: with IaaS VMs, you retain responsibility for the guest OS and what runs inside it.

Before exploring additional VM features and configurations, it is useful to briefly step back and understand how VMs evolved from traditional physical server deployments. The next section

compares physical and virtual machines to clarify the architectural shift that virtualization introduced.

Physical machines versus virtual machines

To understand the benefits of VMs over physical machines, you will first explore the physical hardware operating model. Historically, for each application that an organization wishes to operate, there is typically a **one-to-one mapping** between the application and a dedicated instance of an OS and physical machine, meaning that each app required its own physical machine.

In this traditional one-to-one deployment model, many physical machines would be required. An organization would have to have individual dedicated physical machines to run workloads such as Exchange, SharePoint, Active Directory, files, print, the web, databases, line-of-business applications, and so on.

Figure 4.1 outlines this typical traditional physical hardware operating model, which has a common shared infrastructure connecting and supporting multiple physical machines:

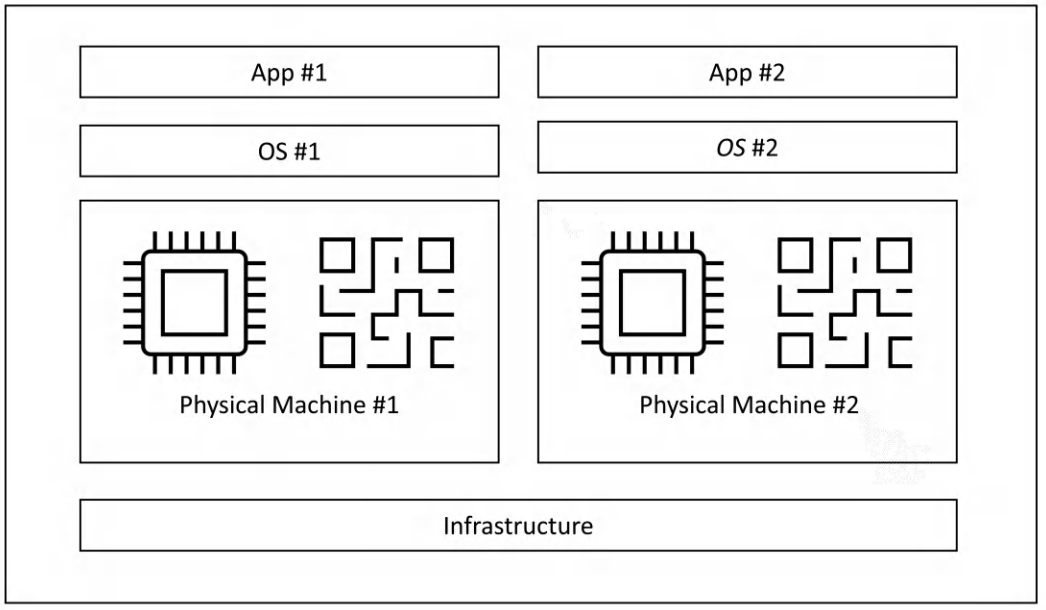


Figure 4.1: Traditional approach supporting a single app per physical machine

Figure 4.1 illustrates the traditional approach to deploying one physical machine per app. It was (and still is) a costly and operationally inefficient mode. However, virtualization is an alternative to this and has many benefits to an organization when running on-premises.

When running VMs in the context of a cloud platform, there can be even greater operational and financial benefits and value, as you learned in *Chapter 1*, *Introduction to Cloud Computing*. Adopting public cloud virtualization removes the need to purchase and maintain hardware in an on-premises facility and shifts infrastructure costs away from an upfront CapEx model, although on-premises hardware may also be leased as OpEx.

But before we cover VMs, we need to know what virtualization is as a technology concept, its benefits, and what value it provides to an organization's technology and business personas. **Virtualization**, as a technology concept, is based on abstraction and, specifically, hardware abstraction. The technology layer that is used to achieve the hardware abstraction is known as a **hypervisor**. **Abstraction** means removing a dependency and filtering out the facts of some characteristics no longer relevant, which allows you to focus on what is important.

So, for example, in the case of virtualization, the VM is no longer dependent on the hardware. If you have abstracted or removed the hardware from the equation, you no longer need that layer or its detail; the hardware is somebody else's responsibility to handle. Likening this to the analogy of pizza cooking and the *Pizza-as-a-Service* example from *Chapter 1*, *Introduction to Cloud Computing*, you could say that your favorite franchised pizza outlet has abstracted the cooking process in that they have abstracted the ingredients, the recipe, the oven, as well as the whole kitchen; for example, you request from a pizza outlet a specific pizza size that meets your requirements, and all you have to do is consume it.

You have removed/filtered out (no longer your concern or care) the details of stocking the ingredients and knowing recipes, the ability to cook, the cooking premises, and having a special pizza oven, as well as removing the actual cooking process, adhering to health and safety regulations, meeting food hygiene standards, and so on.

In the *Virtualization versus containerization* section later, you will see that containerization is all about abstraction at the OS level, while virtualization means abstraction at the hardware level.

Figure 4.2 visualizes the approach of virtualization:

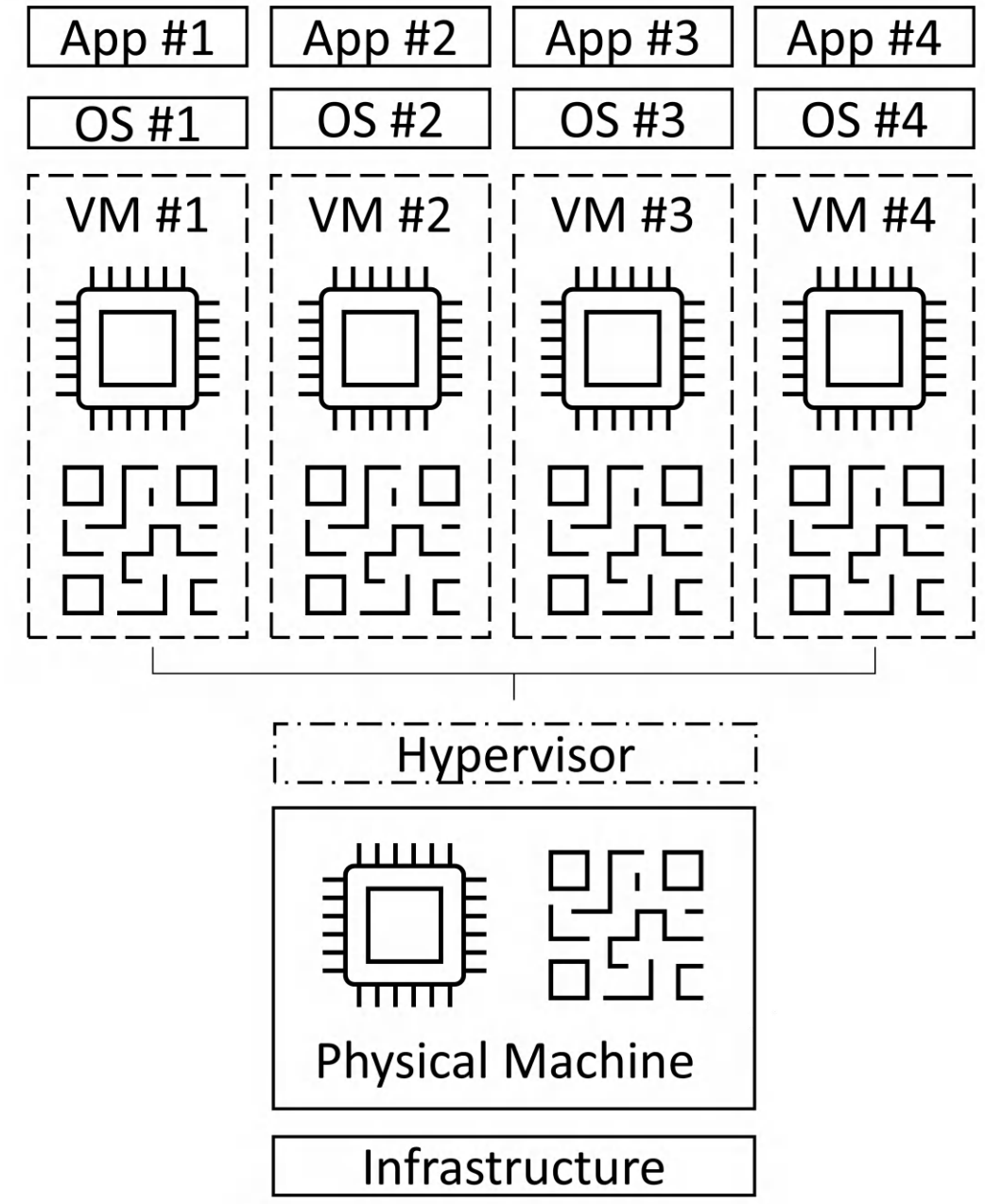


Figure 4.2: Virtualization approach supporting multiple apps per physical machine

This model allows fewer physical machines to run more applications. In *Figure 4.1*, you had two physical machines to run two apps. With virtualization, you can now run four applications from just one physical machine. Each VM shares the underlying hardware resources on which the VM is located, referred to as the **host resources**. You can create many virtual CPU and memory resources from one physical CPU resource; how many VMs you can create on a physical host machine will vary depending on the underlying host's resources… your mileage may vary. That variability is exactly why Azure offers different VM types and sizes, each designed for specific workload characteristics.

VM types

Different workloads have different requirements and therefore need different solutions. To address this, Azure has different VM types (and sizes), and each VM type is tailored to a specific use case and workload type.

VM types are broken down into categories and family series. The family series identifier is a leading alphabetic character. In addition, a naming convention is used to break down the VM types based on their intended use case. Some examples of this include the following:

- Subfamilies
- The number of virtual CPUs (this can be expressed as the number of vCPUs)
- Additive features
- Versions

VM naming conventions are examined later in this section.

Now, take a look at the following categories, family series, and the intended purpose of various VMs:

- **General-purpose (A, B, D, DC family series)**: These VMs have a balanced CPU-to-memory ratio. They are best suited for testing and development (A series only), burstable workloads (B series only), and general-purpose workloads (D series only).
- **Compute optimized (F, FX family series)**: These VMs have a high CPU-to-memory ratio. They are best suited for web servers, application servers, network appliances, batch processes, and any workload where bottlenecks and a lack of resources will typically originate in the CPU rather than in memory.
- **Memory optimized (E, Eb, EC, and M family series)**: These VMs have a high memory-to-CPU ratio. They are best suited for relational databases, in-memory analytics, and any workload where bottlenecks and a lack of resources will typically originate in the memory over the CPU.

- **Storage optimized (L family series)**: These VMs have high disk throughput and I/O. They are best suited for data analytics, data warehousing, and any workload where bottlenecks and a lack of resources will typically relate to the disk over the memory and CPU.
- **GPU optimized (NC, ND, NG, NV family series)**: These VMs have **graphics processing units** (**GPUs**). They are best suited for compute-intensive, graphics/gaming-intensive, visualization, and video conferencing/streaming workloads.
- **FPGA optimized (NP family series)**: These VMs use FPGAs to provide hardware-level acceleration for specialized algorithms. They are best suited for workloads requiring very low latency and high throughput, such as real-time processing, network and security inspection, and custom compute acceleration.
- **High performance (HB, HC, HX family series)**: These are the most powerful CPU VMs that Azure provides, offering high-speed throughput network interfaces. They are best suited for compute and network workloads such as SAP HANA.

While familiarity with these VM family series is sufficient for exam purposes, understanding Azure's naming conventions helps when selecting appropriate VM sizes in real-world scenarios.

These naming conventions can help you identify the many VMs you can choose from when you create a VM; this section will take a closer look at the Azure naming conventions involved.

For simplicity and readability, the core aspects can be presented as follows:

```
[Family] + [Sub-Family] + [# of vCPUs] + [Additive Features] + [Version]
```

Here is some information on the core aspects represented in the VM series naming:

- `Family`: This represents the VM family series.
- `Sub-Family`: This represents the specialized VM differentiations.
- `# of vCPUs`: This represents the number of vCPUs of the VM.
- **Additive features examples** (more exist):
    - `a`: This indicates an AMD-based processor.
    - `d`: This indicates a VM with a local temp disk.
    - `i`: This indicates an isolated size.
    - `l`: This indicates a low memory size.
    - `m`: This indicates a memory-intensive size.

- `P`: This indicates an ARM-based processor.
- `s`: This indicates the presence of Premium Storage capabilities. However, some newer VM options without this attribute can still support Premium Storage.
- `t`: This indicates a tiny memory size.

> **Note**
>
> Further details on Azure VM size families, series, and naming conventions can be found at `https://learn.microsoft.com/en-us/azure/virtual-machines/sizes/overview`.

- `Version`: This represents the version of the VM family series. Some example breakdowns are as follows:
  - `B2s v2`: B series family, two vCPUs, memory-intensive, Premium Storage-capable, version 2
  - `D4ds v6`: D series family, four vCPUs, local temp disk, Premium Storage-capable, version 6
  - `D4as v7`: D series family, four vCPUs, AMD, Premium Storage-capable, version 7
  - `E8s v6`: E series family, eight vCPUs, Premium Storage-capable, version 6
  - `NV16as v4`: N series family, NVIDIA GRID, 16 vCPUs, AMD, Premium Storage-capable, version 4

Understanding VM types is only part of the decision. Once you have selected an appropriate VM size and family, you must also consider how that VM will be deployed, secured, scaled, and managed within your Azure environment.

VM deployment considerations

When it comes to deploying VMs on an Azure subscription, you should think of your solution more broadly, not just focusing on deploying resources in isolation without further consideration of other elements, such as the operational aspects. Some additional elements to consider when creating VMs for a solution are outlined in this section and visualized in *Figure 4.3*:

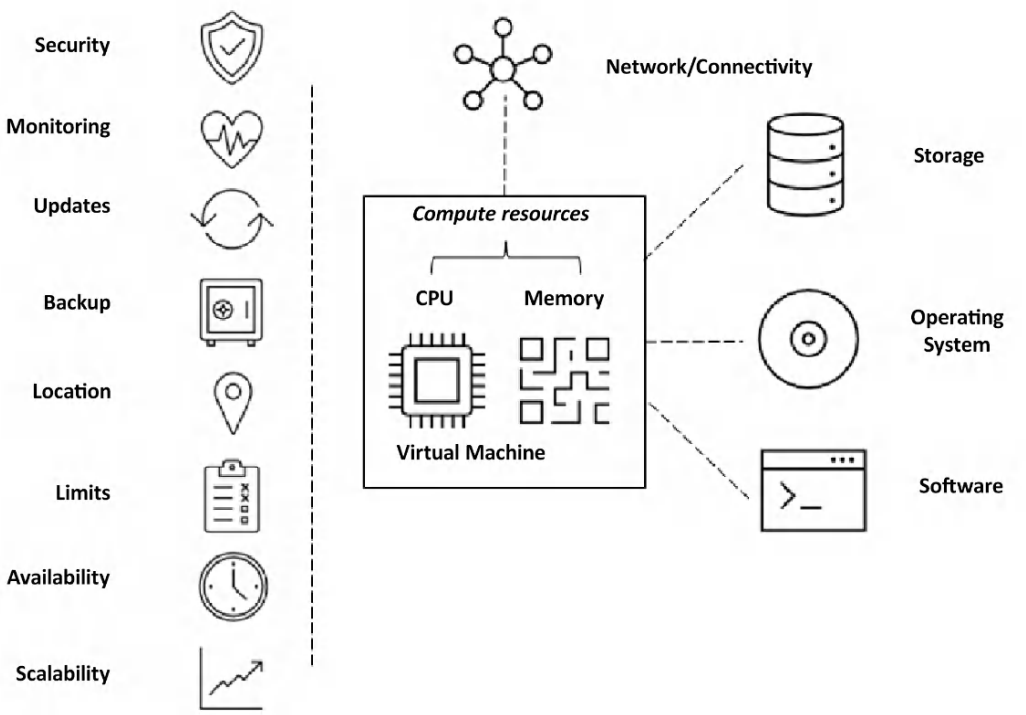


Figure 4.3: VM components and considerations

Several additional elements should be considered when deploying VMs:

- **Additional VM resources**: A VM includes a virtual CPU and memory as its core components. However, you must also provide supporting resources such as storage and networking, and in many exam scenarios, this includes assigning a public IP address to enable direct inbound connectivity from the internet.
- **Location and data residency**: Not all VM types and sizes are available in all regions. Check whether the VM types and sizes required for your solution are available in the region where you will be creating resources. Data residency may also be an important point to ensure you meet any compliance needs mandated for your organization. Different regions will also have different costs for VM creation.
- **VM quota limits**: Each Azure subscription has default quota limits that could impact the creation of VMs. In addition, there are limits on the number of VMs, the VM's total cores, and the VMs per series. The specific default limits depend on the Azure subscription billing type. Quota limits for SQL VMs and virtual desktop VMs that

require the NV series type, for example, can commonly be exceeded, but this can be resolved by requesting a quota limit increase from Microsoft Support.

> **Note**
>
> You can find further information on these limits at `https://learn.microsoft.com/en-us/azure/azure-resource-manager/management/azure-subscription-service-limits`.

- **Monitoring**: It is vital to have visibility into the performance metrics and operational and security event logs to gain insights; you cannot respond to what you do not know about. Unfortunately, Microsoft does not automatically monitor VMs or their resources. There are automatically captured activity logs, but this is more of an audit and governance control than performance metrics or operational and security event logs; to have more visibility, you must enable monitoring.
  You can use both **Azure Monitor** and **Azure Advisor** to gain insights into your VM's performance and operational health. In addition, a cloud-native **security incident and event management** (**SIEM**) solution, such as Microsoft Sentinel, can be used as a single pane of glass and provides a bird's-eye view across all Azure, other clouds, and on-premises resources.
- **Backup**: Microsoft does not automatically back up the OS or software running on your VMs. Under the shared responsibility model, you have complete control of that aspect, and it is your responsibility to protect it. However, Microsoft's responsibility is to protect the underlying host's hardware and software.
- **Update management**: Microsoft does not automatically update the OS or software running on your VMs. As with backups, you have complete control over this aspect, and it is your responsibility under the shared responsibility model. However, updating the underlying host's hardware and software is Microsoft's responsibility. This leads to the next important aspect, the question of availability.
- **Availability**: This is the percentage of time a service is available for use. The two core components for addressing SLA requirements for VMs are availability sets and availability zones, each providing an SLA.
  Availability sets are designed to protect against localized host and rack failures, as well as planned maintenance events, by distributing VMs across fault domains and update domains within a data center.

Availability zones provide a higher level of resilience by placing resources across physically separate data centers within the same Azure region, protecting against data center-level outages.

This is important to consider because, as your VM is provided by Microsoft's hardware and infrastructure, it can and will fail in unavoidable ways. It is more a case of how you handle the failure when it happens than preventing it.

It may be a planned or unplanned update/maintenance event that Microsoft carries out. So, it may not be an outright failure; it may mean that your VM gets rebooted or moved to another host. This could result in a short service interruption.

These interruptions do not impact how your service operates when you implement measures provided by Microsoft, such as availability sets and availability zones. It is important to note that these measures need to be actioned by you, as they are not configured by default.

- **Scalability**: This is the ability of a system to handle increased loads while still meeting availability goals. **Scaling up** (**vertical scaling**) means changing the VM size to add more CPU and memory to a single VM. **Scaling out** (**horizontal scaling**) means increasing the number of VM instances to distribute the workload across multiple VMs VMs do not typically support scaling or auto-scaling; however, an IaaS resource solution, such as **scale sets**, can provide this functionality for VMs. When the requirement is to deploy and automatically scale multiple identical VMs, the intended solution is typically a VM scale set.

  A **scale set** is an IaaS resource service with built-in auto-scale features for VM-based workloads such as web and application services and batch processing. You specify the number of VMs and OS type to be deployed in the scale set, and a number of identical VMs will then be created.

  You benefit from multiple **fault domains** and **update domains** with scale sets as they are automatically deployed in availability sets and compatible with availability zones to protect against data center failures. This provides the automatic scaling, load balancing, availability, and fault tolerance functionality expected from a cloud model.

You can now identify and select VM types based on workload characteristics. However, selecting a VM is only part of designing a reliable solution. Without understanding how to configure availability, scaling, monitoring, and governance, the workload may not meet production expectations. That is why the next section examines the broader deployment considerations that shape how VMs operate in real-world environments.

Containers in Azure

Container services are a form of compute service outlined in the exam's *Skills Measured* section: *Describe Azure compute and networking services*. The Azure container services are described as follows:

- **ACI**: This is a **Platform-as-a-Service** (**PaaS**) service for containers running in Azure.
- **Azure Container Apps**: These have additional benefits over ACI for running your containers by providing scaling and load balancing, and permitting greater flexibility in your design. They remove the need for container management and also run as a PaaS service.
- **AKS**: This is a container-hosting platform and orchestration service for managing containers at scale.

In many real-world designs, solutions often start with a fully managed container option for simplicity and scaling behavior and move to AKS when they need deep orchestration control, advanced networking patterns, or a standardized Kubernetes platform for larger environments.

Before looking at container services, it is a sound idea to first explore the differences between virtualization and containerization.

Virtualization versus containerization

These compute service approaches can be defined as follows:

- **Virtualization**: This is an approach where the hardware is abstracted. This allows many compute instances (VMs) to share a single host's hardware resources. Each compute instance runs with its own isolated OS.
- **Containerization**: This is an approach where the OS is abstracted. Here, *abstract* means *remove*—that is, remove the requirement to provide that layer. You make the abstracted (removed from thought and consideration) layer the cloud provider's responsibility to keep available, maintain, and so on; you still consume resources from it, but it is a layer that you no longer need to know or care about.

Containers, as a concept, are built around encapsulation and creating standardized software units, packaging an application's code and its dependencies such that they can be deployed equally seamlessly into a development or production environment with expected, repeatable, and consistent results. They are advantageous as they are intended to be portable, self-contained computing environments.

A container is a compute unit similar to a VM. Containers and VMs have the same goals—that is, to host and execute code. However, VMs have a lot of overhead, are big, utilize a lot of resources, and have slow boot times. Containers differ significantly from VMs in both size and startup behavior, offering a more lightweight and agile compute option. They are lightweight, smaller in size, utilize fewer resources, and have quicker boot times. So, they are more agile, have more efficient compute units than a VM, and are therefore a perfect fit for your digital transformation journey and modernizing your data center and workloads.

You may even be brave enough to say that containerization, at some point, will do to virtualization what virtualization did to the process of installing an OS in the physical server bare-metal approach. This containerization approach allows many compute instances (containers) to share the host software resources of one OS on a single host. Thus, it can be thought of as more of an application delivery model than a virtualization model.

Figure 4.4 visualizes the concepts of virtualization and containerization against the traditional physical approach:

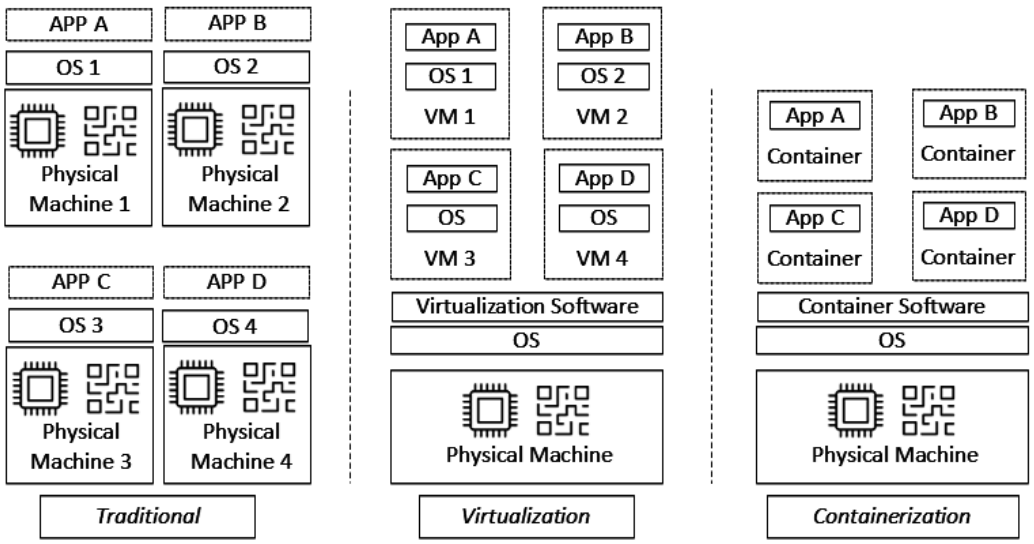


Figure 4.4: Concept differences in traditional versus virtualization versus containerization

Looking closely, you will observe that *Figure 4.4* visualizes the different approaches that can be used to host your application. These are explained in detail in the following sections.

Traditional approach

Before virtualization and containerization technologies, applications were installed on an OS that was installed directly on the hardware. This process is usually referred to as being installed on **bare metal**. This means there is no software (such as a hypervisor) between the OS and the hardware.

The early physical approaches installed many apps on one OS and one physical server (often referred to as a *piece of tin*), but this often resulted in resource starvation, where the resource-hungry app would consume all the resources or raise software conflicts with processes and libraries (referred to as **DLL hell**, where **DLL** stands for **Dynamic Link Libraries**).

This was solved by running each application on its own physical server. However, this brought other problems as you had over-resourced and under-utilized servers, costing a lot of money with little output. In addition, it brought physical tin sprawl across the data center, which meant that you needed more physical rack space. Expanding physical rack space required more floor space, cabling, power, cooling, and so on. Finally, this meant that the IT focus was on data center capacity management, and this distracted engineers from their core task—the job of deploying and operating apps for the business.

The bottom line is that the ability of one physical server to deliver one app introduced an inefficiency-of-scale problem that was seen as a cost to the business. Hence, IT was seen as a cost center, not an innovation or value-creation center.

Virtualization approach

Virtualization was an opportunity to change the previous scenario of one app per physical server. This ability allowed abstracting (removing dependency on) the hardware.

The benefits that virtualization provides include tackling the two key challenges of the traditional physical approach (i.e., scale and utilization). Virtualization meant that you could run multiple VMs and, therefore, multiple applications on one physical server, meaning greater resource utilization and less floor space, power, and cooling are required.

In conclusion, fewer physical servers are required to deliver more apps, translating to fewer costs to deliver the same applications to the business. Due to this, you have a leaner, more agile, tangible, and value-delivering IT team.

Containerization approach

Containerization is another evolution in delivering applications to a business. This approach changed from abstracting the hardware to abstracting (removing dependency on) the software, bringing about the benefits of containerization.

The key benefits and metrics are greater resource utilization, efficiency, and isolation, smaller sizes, faster boot-up, agile app deployment, and no requirement to provide a guest OS for each app, as they all share the underlying host OS kernel.

The bottom line is that fewer physical servers and OS instances are required to deliver more apps, which, again, means fewer costs to deliver more applications to the business in a more demand-driven, agile, and standardized manner.

Having established how containerization differs from virtualization and why it enables lighter, faster application delivery, the next step is to examine how Azure implements this model in practice through Azure Container Instances.

Azure Container Instances

Azure Container Instances (**ACI**) is a multitenant, serverless **Containers-as-a-Service** (**CaaS**) platform intended for running single-instance containers in isolation. ACI operates as a PaaS offering, allowing containers to be created and run without deploying or managing VMs or container orchestration platforms.

Containers share the kernel of the host OS they are running on. This means containers are very lightweight and have faster boot times. ACI containers can be available in seconds, compared to VMs, which have to host their own guest OSs and have much slower start times. The important fact is that there is no overhead when creating container platforms, no need to manage VMs, and no orchestration components. Both Windows and Linux containers are supported on the ACI container platform.

It should be noted that there are VMs underneath; it is that misnomer of **serverless** again. This term just means that you do not need to concern yourself with the details. Microsoft provides and manages the underlying hosts, the VMs, the container engine, the orchestration components, and so on, leaving a compute unit available to host and execute application code.

The primary benefit is that it allows for the most straightforward and quickest entry point to start running your apps in Azure containers without the need to manage servers or orchestration components.

ACI is best suited to running jobs that are rarely used or scheduled (quarterly, yearly, or ad hoc), including batch processing, and those that are temporary or demand-driven/event-driven. This also includes jobs that do not interact with other container instances/services or require advanced orchestration, load balancing, or auto-scaling, given that these are single-instance containers and are best for single-use and isolated jobs.

A simple rule of thumb is that ACI is best suited for running single, short-lived, or isolated containers, while a Kubernetes-based platform is intended for orchestrating and managing containers at scale.

Azure Container Apps build on this model by adding scaling and application-level orchestration capabilities.

Azure Container Apps

So what is **Azure Container Apps** (**ACA**)? In a nutshell, it's a fully managed serverless container service platform that is powered by Kubernetes, DAPR, **Kubernetes Event-Driven Autoscaling** (**KEDA**), and Envoy open source technologies. It allows Kubernetes-style applications and microservices to be built without having to manage the complexities of container orchestration. No direct access is given to the underlying APIs, control plane, and cluster management of Kubernetes.

You can learn more about DAPR and KEDA at the following URLs:

- `https://docs.dapr.io/concepts/overview/`
- `https://keda.sh/docs/2.11/concepts`

For scenarios that require greater orchestration control and scale, Azure Kubernetes Service provides the next level of container management.

Azure Kubernetes Service

What is the **Azure Kubernetes Service** (**AKS**)? In a nutshell, it is a container hosting platform and orchestration service for managing containers at scale. This approach brings the power of the open source Kubernetes platform but is delivered as a PaaS-managed platform by Microsoft; think of this as CaaS. The service is designed for highly customizable, scalable, and portable workload scenarios.

When comparing ACI with AKS, it is important to know the non-functional aspects of cost, complexity, and operations. In other words, you can think of each of these solutions at opposing ends of the scale. ACI is the simplest in terms of how it operates and also has the lowest cost, while AKS has the greatest functionality, features, and scale. However, this means AKS comes with complexity in terms of how it operates, and it is also expensive.

Figure 4.5 visualizes the difference between the ACI and AKS container services:

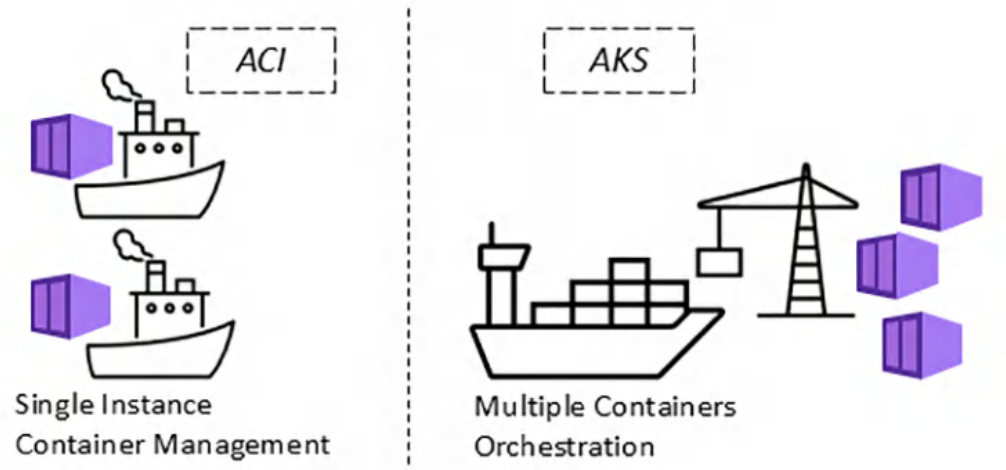


Figure 4.5: ACI versus AKS for containerization compute services

Unlike ACI, which runs individual containers without orchestration, AKS provides a full Kubernetes-based orchestration platform for managing containerized workloads at scale.

The AKS architecture has two essential elements—the master node and the worker node. The master node is the control plane managed by Microsoft, while the worker nodes contain Pods that the customer manages.

Figure 4.6 shows the high-level AKS architecture:

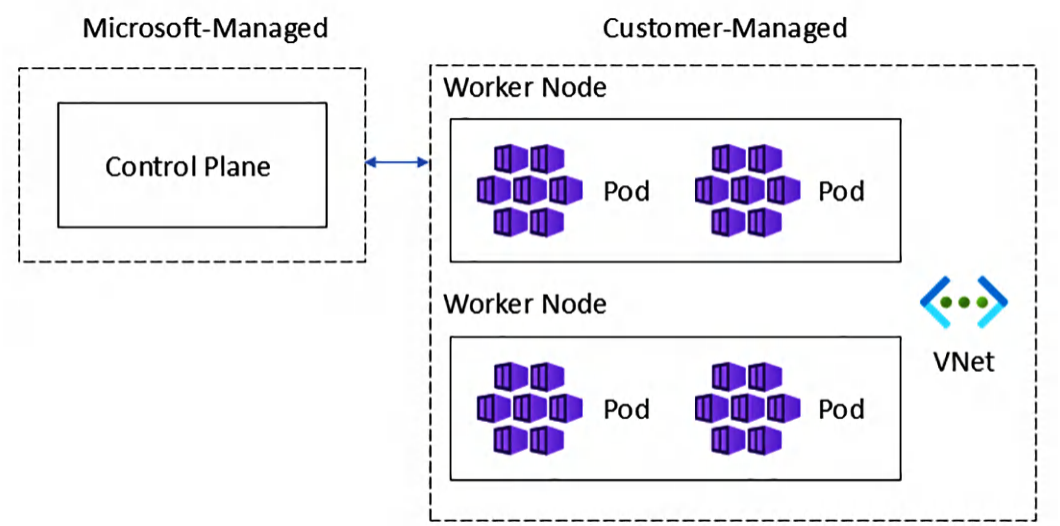


Figure 4.6: AKS high-level service architecture showing Microsoft's responsibilities versus the customer-managed components

The **master node** is responsible for scheduling the underlying worker nodes, Pods, and the **worker nodes** are the VMs that run the node's components and the container runtime. *Figure 4.7* shows the high-level worker node service architecture:

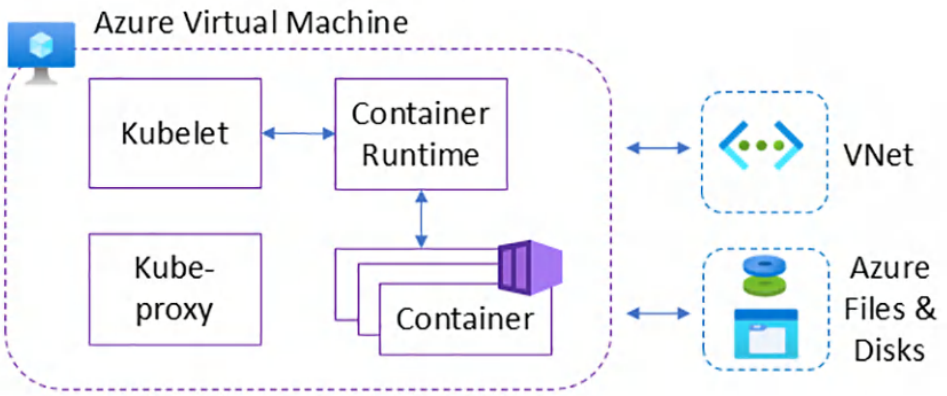


Figure 4.7: Worker node high-level service architecture

You should size and scale out the VMs in your nodes to support the performance and capacity demands you need to meet. The Azure VM for cluster nodes is based on Windows Server 2019

or Ubuntu Linux. Microsoft manages the process of creating and scaling these VMs. These nodes are billed as regular VMs, so discounts such as reservations can be applied.

While VMs and container platforms such as ACI and AKS provide varying degrees of infrastructure control, not every workload requires that level of management responsibility. Many web applications and APIs can be hosted without managing servers or orchestration layers. For these scenarios, Azure App Service provides a fully managed platform approach.

Azure App Service

Azure App Service is a compute service outlined in the exam's *Skills Measured* section: *Describe Azure compute and networking services*.

Azure App Service is a PaaS resource and provides a website and code-hosting platform fully managed by Microsoft. The platform supports Windows and Linux workloads, multiple frameworks, and programming languages such as .NET, .NET Core, Node.js, Python, PHP, and more. *Figure 4.8* visualizes how you can host a website or code in the IaaS model versus the PaaS model:

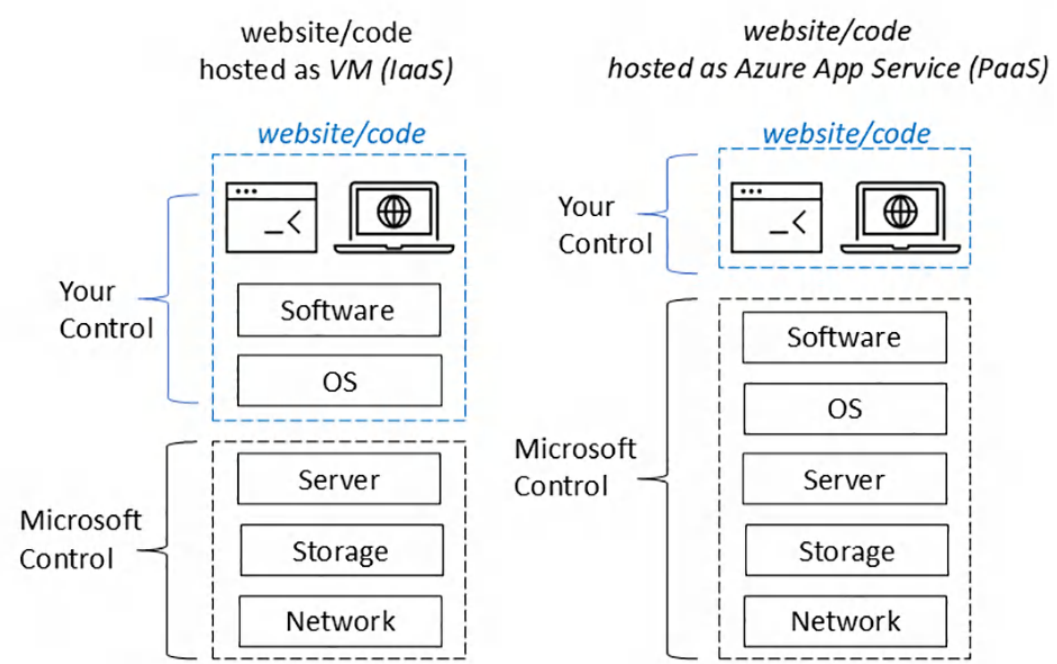


Figure 4.8: IaaS hosting with Azure VM versus PaaS hosting with Azure App Service

Figure 4.8 compares the shared responsibility boundaries between hosting a web application on an Azure VM (IaaS) and hosting it on Azure App Service (PaaS).

In the IaaS model, you retain control over the OS and installed software, while Microsoft manages the underlying infrastructure. In the PaaS model, Microsoft manages not only the infrastructure but also the OS and runtime platform, leaving you responsible primarily for your application code.

Being a PaaS service means that this is a hosting platform managed by Microsoft; Azure App Service is responsible for the underlying compute service that will host your website or code. Operationally, App Service shifts the focus from server maintenance to application lifecycle tasks such as deployment, configuration, scaling, and monitoring. This is one of the reasons PaaS hosting remains a common default for web workloads when full OS control is not required. As a result, you do not need to manage VMs or containers and are only responsible for providing the website or code to be hosted.

Workloads that require deep OS control, custom drivers, or administrative access typically align better with VMs or container-based services than with Azure App Service. Each app service runs inside an **App Service plan**. You can think of an App Service plan as a TV subscription service plan or a mobile/broadband service provider plan. Each has a tariff with a set of features and functionalities you have enabled and can access as a subscriber. The more fully-featured the plan is, the more choices and options you will have. You just need to choose a plan with the most appropriate features that meet your needs.

Figure 4.9 outlines the relationship between all the components of Azure App Service:

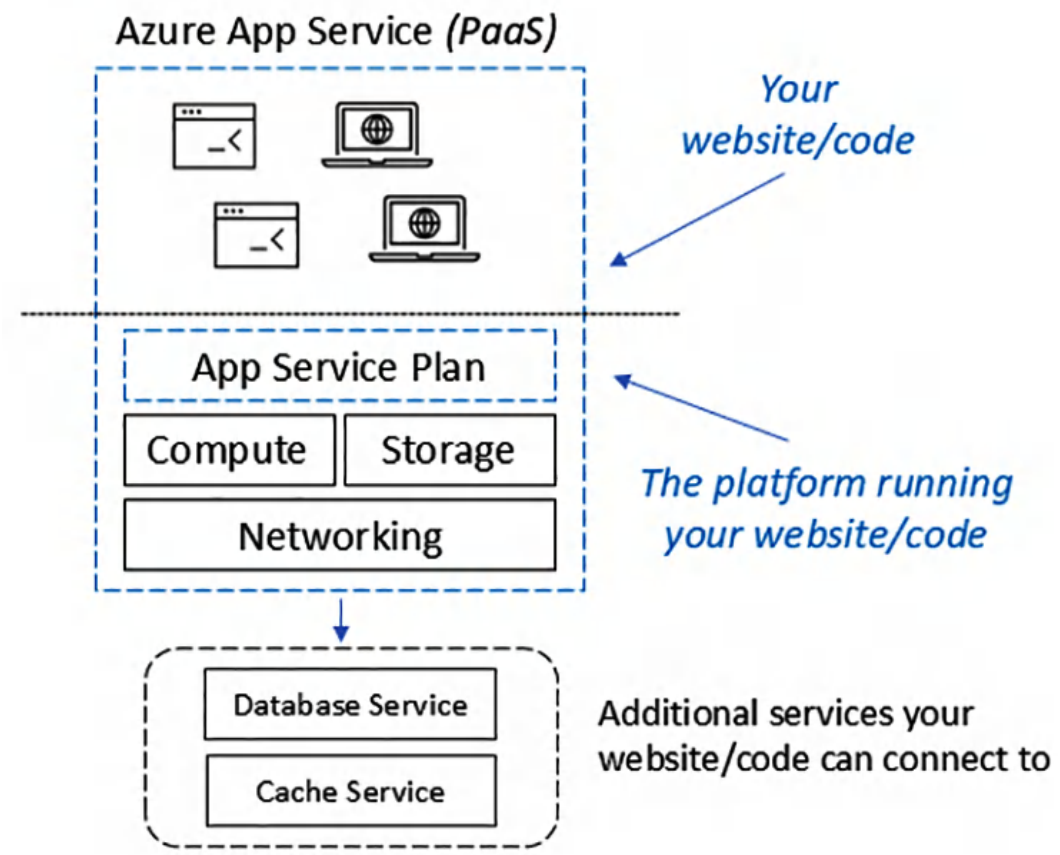


Figure 4.9: Azure App Service plan components relationship

Figure 4.9 illustrates the logical relationship between your web application and the App Service plan that hosts it. The web app represents your code, while the App Service plan provides the underlying compute, storage, and networking resources that run that code. External services such as databases and cache services sit outside the plan and are consumed by the application as needed.

In reality, the App Service plan is closer to that of a server farm; it defines the number of VMs your App Service instance will use, the type and size of VM, and its properties, and the region in which the VMs will be created. One App Service plan (server farm) can be used to host multiple web apps if enough resources are available from the underlying VM resources.

A web app can only be hosted on a single App Service plan (one set of compute instances, one server farm), but an App Service plan may host many web apps. A web app, when it is being

created, can be added to an existing App Service plan. Alternatively, you can create a new App Service plan to host your web app/code.

You can select an OS to host your app/code, as well as a runtime stack such as .NET, PHP, or Java, and its version. You can also select the region where your website/code will be hosted. This could have cost or data compliance implications if you must use certain regions based on compliance rules.

App Service has pricing tiers that define the features and functionality available in each tier. The following are the price tiers available:

- **Free and shared**: This is intended for development and testing purposes. No SLA is provided, and your app will run on the same compute service resources as other customers. There is no support for autoscaling, hybrid, or VNet connectivity. Also, custom domains are only available in the shared tier.
- **Basic service plan**: This is intended for low-traffic usage that does not require advanced autoscaling and traffic management features. It has built-in basic load balancing across instances. Custom domains and hybrid connectivity are supported, but not VNet connectivity.
- **Standard service plan**: This is intended for running production workloads. It supports custom domains, autoscaling, hybrid, and VNet connectivity.
- **Premium v2/v3/v4**: This is intended for larger-scale and performance production workloads. It provides all the features of Standard, plus private endpoints.
- **Isolated/Isolated v2**: This is intended for mission-critical workloads required to run on a virtual network in a private, dedicated environment.

Note

More information on pricing tiers can be found at `https://learn.microsoft.com/en-us/azure/app-service/overview-hosting-plans`.

An App Service plan is billed even when no web apps are running. This is because the VM is created as part of the plan and will remain running until it is deleted, even with no web apps running on the VMs. The only way to stop billing is to delete the App Service plan. An App Service plan has **scale-up** and **scale-out** properties in the portal when configuring an App Service plan. The scale-up property adds functionality and features. These are part of a pricing tier, and you are, in effect, selecting a VM with a preconfigured runtime stack such as .NET, PHP, and Java. You can select any of the different versions available.

The number of compute instances that can be scaled is determined by the pricing tier. This compute instance scaling is only available for the Basic, Standard, and Premium pricing tiers. An App Service pricing tier can be changed after creation. Changing this will modify the functions, features, and the number of compute instances available to run your web app.

The scale-out property adds additional compute instance resources to support running these apps. You can scale manually or set up custom autoscaling on a schedule based on any metric available. Your pricing tier will determine your scale-out limit, which could be either Basic, Standard, or Premium.

This section looked at Azure App Service. You learned how you can use a PaaS approach to host a website or code versus the IaaS approach, as well as the components' relationship embodied in the Azure App Service plan. However, not all workloads require a continuously running web application or dedicated compute plan. In scenarios where code needs to run only in response to events, a different compute model is more appropriate. Next, you will take a brief look at another Azure compute function, Azure Functions.

Azure Functions

Azure Functions is a serverless code engine. It is a **Function-as-a-Service** (**FaaS**) or serverless model where you provide your code and business logic, and the cloud provider hosts it in their language, runtime, and compute environment shared with other tenants.

It is designed for event-driven workloads, where code runs in response to a trigger such as an HTTP request, a timer schedule, a message arriving in a queue, or an event from another Azure service. Microsoft manages the underlying infrastructure, runtime environment, and scaling behavior, while you provide only the function code and configuration. This model is well-suited to short-running, on-demand tasks such as automation, lightweight APIs, data processing, and integration workflows. In exam scenarios, when the requirement is to run code in response to an event without managing servers, Azure Functions is typically the intended compute service.

This approach aligns well with modern event-driven and automation patterns, where workloads scale dynamically, and cost is optimized by running only when triggered. Beyond application and code hosting, Azure also provides compute services focused on end user access, such as Azure Virtual Desktop.

Azure Virtual Desktop

The **Virtual Desktop** service is a compute service that you must learn about, as outlined in the exam's *Skills Measured* section: *Describe Azure compute and networking services*.

Microsoft's Azure Virtual Desktop service is a desktop and application virtualization service that provides access to your desktop and applications from virtually anywhere and on any device. It is provided as a PaaS service that allows remote users to connect from their devices to their hosted desktops and remote applications in Azure. This access is provided securely and reliably from any location with an internet connection or over a private managed network, such as Microsoft's ExpressRoute service. Many device types and Virtual Desktop clients are available to provide the broadest range of connectivity and access methods. The following are some of the benefits of Virtual Desktop:

- It provides a full Windows client and Office 365 experience.
- It delivers the only multi-session Windows 10/11 experience.
- It paves a migration path for **Remote Desktop Services** (**RDSs**).
- It deploys and scales in minutes.
- It allows access to the cloud service from anywhere, from a web client to a desktop client such as Windows or macOS, and iOS, Android, and Linux devices.
- It provides centralized identity management and security using Entra ID for role-based access control with Conditional Access, Microsoft Intune, and Microsoft Defender support.
- It provides improved remote access security using reverse connect technology, meaning no inbound ports to the session host's VMs, thus reducing the attack surface area.
- It provides the most cost-effective service using pooled desktops through the traditional RDS session host model approach, where many users share one session host VM. Alternatively, virtual desktops can provide the most performant and secure solution through the traditional VDI approach, where power users or security requirements mandate a level of isolation between user sessions, with one user per VM (where a desktop cannot be shared between users). This is the personal desktop approach.
- Where a full desktop experience is not required, applications can be published without exposing the full desktop.

Figure 4.10 shows the Azure Virtual Desktop approach and its components:

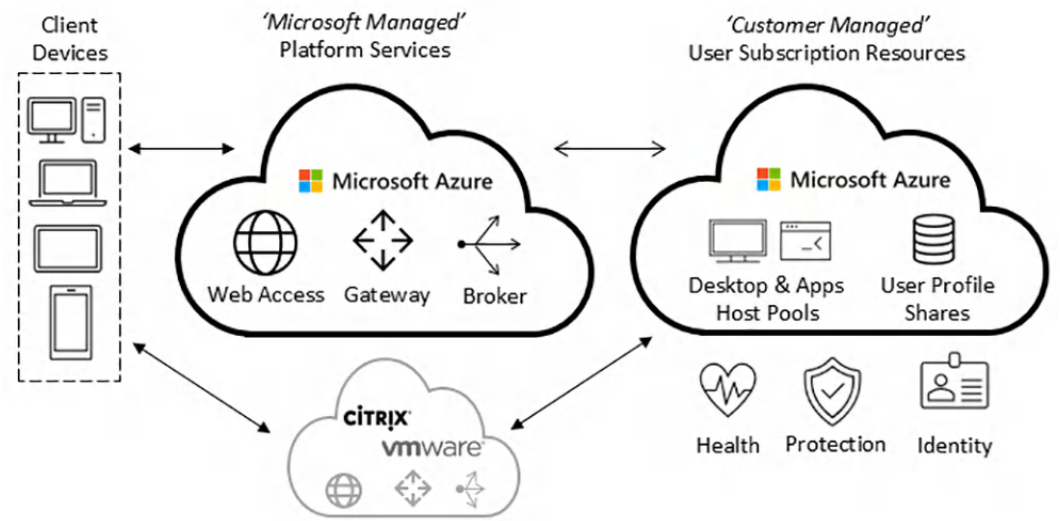


Figure 4.10: Azure Virtual Desktop service providing access from virtually anywhere

As shown in *Figure 4.10*, the Azure Virtual Desktop service has the following elements:

- **Microsoft-provided and managed platform services**: These are PaaS functions where Microsoft provides the managed services of the web access, gateway, and broker roles. These provide a secure connectivity service layer that connects users with their desktops and published apps.
- **Host pools**: These are collections of VMs. They are the user-assignable entities that will run your users' desktops and publish the remote applications. For example, you may have multiple host pools to meet your needs, and each pool is a collection of VMs that share the same image and configuration that may be assigned to a different set of users. Different images and configurations could be available to different users and teams based on Microsoft Entra ID group assignments to different pools.
  In addition, there are two load-balancing methods—that is, depth mode and breadth mode. **Depth mode** saves costs by fully utilizing a single VM to host users' sessions before placing a session on the next VM. This is known as **vertical load balancing**. On the other hand, **breadth mode** is the best load-balancing method for performance since each user session is created on the next available VM and never on the same VM as the last session it was connected to (where two or more VMs are available in the host pool). This is known as **horizontal load balancing**.
- **Customer-created desktops**: These infrastructure resources provide desktops and applications for end users. Host pools contain the session host VMs that provide the

user's desktop, published apps, and file shares; file shares will be required to host the user's profile containers.

- **Customer-created remote applications**: Applications no longer need to be installed on the VM OS itself or included as part of the image. Through the MSIX app attach functionality, the apps are decoupled from the OS and are then dynamically attached to the VM that a user is connected to for their remote session.
- **User profiles**: Here, the **FSLogix** profile container technology is used, which allows a user profile to be stored on a **virtual hard disk** (**VHD**) in the file share location. This is then dynamically attached to the VM that a user is connected to for their remote session, whether it be the full desktop or just the application that has been published.
- **Third-party provided and managed virtual desktop platform services**: Vendors such as Citrix and VMware are part of the ecosystem that provides additional functionalities. You can utilize their presentation and management layers to connect to your virtual desktops and apps.
- **Cost savings**: Cost savings can be made by using autoscaling and running the VMs only when users connect to them. You can also consider using ephemeral disks to negate the need to persist disks in storage. In contrast to shutting VMs down to save costs, if the VMs must be running 24/7, then you can save on pay-as-you-go metered pricing by committing to a 1- or 3-year commitment to reserve the compute capacity, which is called **Reserved Instances** (**RIs**) or **VM reservations**. Note that this only discounts the compute costs of the VM. You will still have to pay the OS costs for storage, networking, and so on.
  Finally, through the **Azure Hybrid Benefit** (**AHB**), you can also negate the OS costs for the VMs if you have eligible software licensing. You should contact Microsoft Support or a licensing specialist partner for guidance in this area.

Together, these Azure compute services illustrate how different operational models support a wide range of application, infrastructure, and end user scenarios, from full OS control to fully managed, event-driven execution.

Summary

This chapter covered the Azure compute skills measured area of the *AZ-900 Azure Fundamentals* exam, showing how Azure compute options support common workload scenarios.

By this point, you should be able to distinguish between infrastructure-based, platform-based, and serverless compute models, and understand when each is appropriate. You learned how VMs provide full control at the cost of greater management responsibility, how containers

improve efficiency and portability, and how serverless options such as Azure Functions further abstract infrastructure. You also examined supporting constructs such as scale sets, availability sets, and App Service plans, enabling you to reason about scalability, resilience, and operational overhead when selecting a compute option.

In the next chapter, you will build on this foundation by exploring Azure Storage services. Since every compute workload requires data persistence, understanding storage options is the natural next step in designing complete Azure solutions.

Exam Readiness Drill — Chapter Review Questions

Apart from mastering key concepts, strong test-taking skills under time pressure are essential for acing your certification exam. That's why developing these abilities early in your learning journey is critical.

Exam readiness drills, using the free online practice resources provided with this book, help you progressively improve your time management and test-taking skills while reinforcing the key concepts you've learned.

HOW TO GET STARTED

- Open the link or scan the QR code at the bottom of this page
- If you have unlocked the practice resources already, log in to your registered account. If you haven't, follow the instructions in *Free Benefits with Your Book* section in this book's *Preface* and come back to this page.
- Once you log in, click the START button to start a quiz
- We recommend attempting a quiz multiple times till you're able to answer most of the questions correctly and well within the time limit.
- You can use the following practice template to help you plan your attempts:

Working On Accuracy		
Attempt	**Target**	**Time Limit**
Attempt 1	40% or more	Till the timer runs out
Attempt 2	60% or more	Till the timer runs out
Attempt 3	75% or more	Till the timer runs out
Working On Timing		
Attempt 4	75% or more	1 minute before time limit
Attempt 5	75% or more	2 minutes before time limit
Attempt 6	75% or more	3 minutes before time limit

The above drill is just an example. Design your drills based on your own goals and make the most out of the online quizzes accompanying this book.

First time accessing the online resources?

You need to unlock the online practice resources to access the link below. Check the *Free Benefits with Your Book* section in the *Preface* for detailed instructions on how to do this. You only need to unlock the resources once.

Open Quiz `https://packt.link/AZ-9003ech4` or scan the QR code

5

Azure Storage Services

In *Chapter 4*, *Azure Compute Services*, you gained knowledge of the Azure platform's core components. This chapter builds on those foundations by examining how Azure Storage services provide the durability, availability, and scalability required to support modern cloud workloads.

This chapter primarily focuses on the *Describe Azure Architecture and Services* module from the *Skills Measured* section of the *AZ-900 Azure Fundamentals* exam.

Building on your understanding of Azure compute services, this chapter completes the core infrastructure picture by focusing on how data is stored, replicated, and protected in Azure. Since every workload depends on reliable data persistence, understanding storage services enables you to evaluate durability, redundancy, and access patterns when designing solutions. This knowledge strengthens your ability to reason about real-world cloud architectures rather than viewing services in isolation.

Azure provides several distinct storage services designed for different data access patterns and workload requirements. Some services are optimized for unstructured object storage, others for file shares, messaging, or high-performance block storage. While they appear separate, they are all organized and governed through storage accounts, which act as the foundational management and billing boundary. Understanding how these services differ—and how they are structured under a storage account—provides the context needed before examining configuration details such as replication, tiers, and security. The following core storage services and concepts form the foundation of this chapter:

- Azure Files, containers (Blob Storage), Queue Storage, and disk storage
- Storage accounts, tiers, replication, storage copying
- Data stores

By the end of this chapter, you will be able to confidently answer questions on the following topics:

- Storage account types in Azure
- Storage redundancy options
- Storage tiers in Azure
- File, object, and disk storage options
- Moving data into Azure Storage

Beyond exam objectives, this chapter aims to help you think critically about how data should be stored, protected, and accessed in different workload scenarios, since storage design decisions often determine performance, cost, and resilience outcomes in real Azure environments.

Storage account types in Azure

At the center of all Azure Storage services is the **storage account**, which provides both the management boundary and the data boundary for stored objects. In addition to working as data planes, they can also be considered management plane consoles.

Storage accounts are used to define the settings and configurations that can be applied to the data held in the storage account, such as tiering, redundancy, security, access control, protection, monitoring, and so on. Storage accounts are automatically encrypted for each storage service. They can be monitored using Azure Storage Analytics and Azure Monitor.

While storage services generate metrics and logs, Azure Monitor stores platform monitoring data in Log Analytics workspaces, not in general-purpose storage accounts by default; storage accounts are not used as the primary store for Azure Monitor performance data.

You can create your first storage account using the Azure portal by navigating to the **Storage accounts** blade and selecting the **Create** button, as illustrated in *Figure 5.1*:

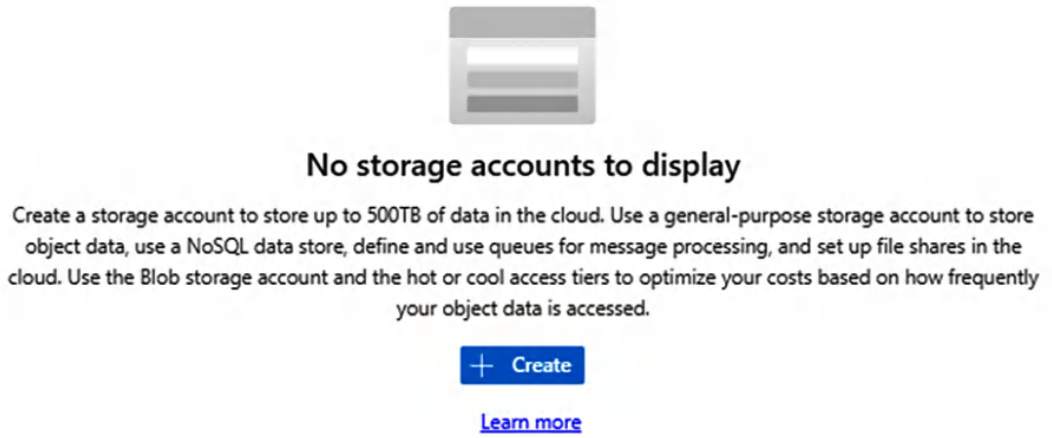


Figure 5.1: Creating a storage account in Azure

You can also use **PowerShell**, the **Azure CLI**, or an **Infrastructure-as-code** (**IaC**) deployment language such as **Terraform** or **Bicep** with **ARM** templates.

Once created, the storage account can be accessed via HTTP/HTTPS through a globally unique namespace; these are known as **endpoints** and can be public or private:

- A **public endpoint** is assigned a public IP and can be accessed over the internet
- A **private endpoint** is assigned a private IP address from a **virtual network** (**VNet**) and is only accessible from the VNet

The following types of storage accounts can be created:

- **Standard general-purpose v2**: This is a *standard* storage account that supports blobs, file shares, tables, and queues
- **Premium block blobs**: This is a *premium* storage account that supports block blob and append blob types
- **Premium page blobs**: This is a *premium* storage account that supports page blobs only
- **Premium File shares**: This is a *premium* storage account type that supports file shares only

Figure 5.2 illustrates these storage account types:

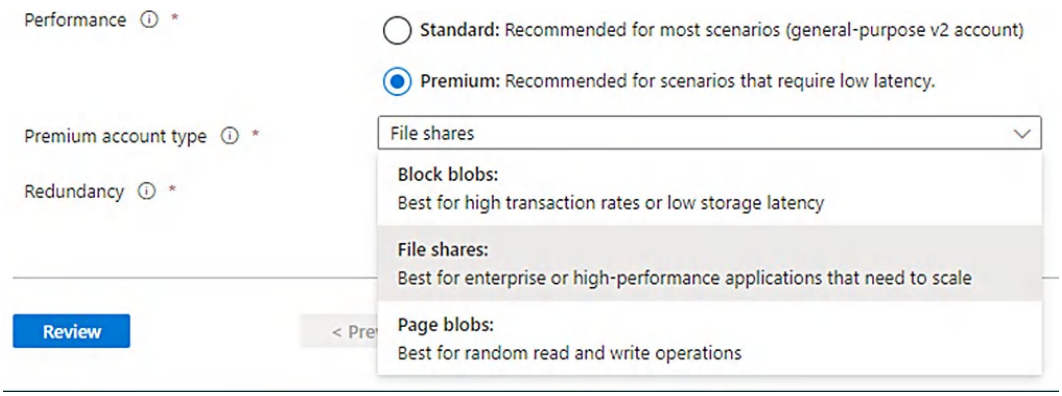


Figure 5.2: Storage account types

You cannot change the performance type setting after creating a storage account. This is clearly stated, as shown in *Figure 5.3*:

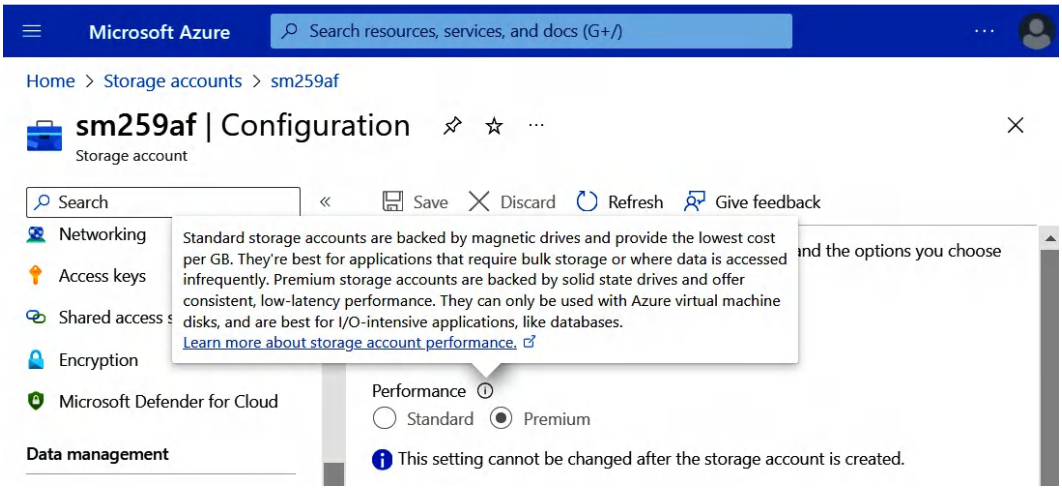


Figure 5.3: Storage account performance type setting

You must create a new storage account of the performance type you need and then move the data to the created storage account. The storage account also provides a unique namespace (and endpoint) for each data object to address each storage account uniquely.

The unique identifier for a storage object is constructed in the following format:

```
https://<storage account name>.<service name>.core.windows.net
```

The following are the endpoints of each of the Azure storage services:

- **Blob storage**: `https://<storage-account>.blob.core.windows.net`
- **Data Lake Storage Gen2**: `https://<storage-account>.dfs.core.windows.net`
- **Azure Files**: `https://<storage-account>.file.core.windows.net`
- **Queue Storage**: `https://<storage-account>.queue.core.windows.net`
- **Table storage**: `https://<storage-account>.table.core.windows.net`

While understanding service endpoints helps you see how Azure storage is accessed, designing reliable solutions also requires understanding how that data is protected against failure. The next section, therefore, examines storage redundancy options and how replication models affect durability and availability.

Storage redundancy options

Microsoft provides **redundancy** of your data in Azure through several **replication** options. You will look at all of that in this section. However, note that under the shared responsibility model, you are responsible for ensuring that your data is available and protected by selecting the most appropriate replication and protection model to meet your needs.

The following are the redundancy options that are available within a **primary region**, though some regions offer more replication types:

- **Locally redundant storage (LRS)**: This provides three copies of your data, replicated synchronously within a single physical location in the primary region. This is the lowest cost option, but it has the lowest availability and durability.
- **Zone-redundant storage (ZRS)**: This provides synchronously replicated copies of your data across three availability zones in the primary region. This is a higher-cost option but has the highest availability and durability within a region.

The following are the redundancy options that are available within a **secondary region**:

- **Geo-redundant storage (GRS)**: This provides three copies of your data, replicated synchronously within a single physical location in the primary region. Your data is also asynchronously replicated to a single physical location in the secondary region. This secondary region provides three copies of your data, replicated synchronously as LRS.
- **Geo-zone-redundant storage (GZRS)**: It provides synchronously replicated copies of your data across three availability zones in the primary region.

Your data is also asynchronously replicated to a **single physical location** in the secondary region. This secondary region provides three copies of your data, replicated synchronously as LRS.

In scenarios where it is required to use the minimum and lowest-cost replication option available in Azure Storage, then LRS is the correct option; it maintains three copies of data within a single datacenter. Any other storage option that provides replication across availability zones or regions exceeds the minimum redundancy requirements.

While redundancy determines how your data is protected and replicated, it does not determine how that data is accessed or how much it costs to store over time. The next section, therefore, examines storage tiers, which define the performance characteristics and cost model based on data access patterns.

Storage tiers in Azure

Azure Storage services provide different storage tiers to store your data objects in the most cost-effective and performant way that meets your needs. The tier you choose should be based on how frequently data is accessed and whether data can be moved between tiers. At this time, you should also consider the data access/retrieval (rehydration) costs.

The four available storage tiers are as follows:

- **Hot tier**: This is optimized for *frequently accessed* data.
  This tier has the following characteristics:
    - The highest storage costs
    - The lowest access costs
- **Cool tier**: This is optimized for *infrequently accessed* data (i.e., that is stored for at least 30 days).
  This tier has the following characteristics:
    - Lower storage costs
    - Higher access costs
- **Cold tier**: This is optimized for *infrequently accessed* data (i.e., that is stored for at least 90 days).
  This tier has the following characteristics:
    - Lower storage costs than the Cool tier
    - Higher access costs than the Cool tier
- **Archive tier**: This is optimized for *rarely accessed* data (i.e., that is stored for at least 180 days).

This tier has the following characteristics:

- The lowest storage costs
- The highest access costs and data retrieval (rehydration) costs
- Data is offline and will take several hours for the first byte to be accessible

Selecting the appropriate storage tier is therefore a balance between cost and access requirements, with colder tiers offering significant savings at the expense of retrieval time and access charges.

While storage tiers determine how frequently data is accessed and at what cost, they do not define the actual storage service used for different workload types. To design an effective solution, you must also understand the distinct storage options available in Azure and when each is appropriate. The next section examines file, object, and disk storage services in more detail.

File, object, and disk storage options

The following storage types are available in Azure:

- **Files**: This service provides fully managed, highly available serverless **network file shares** using the **Server Message Block** (**SMB**) protocol. These traditional mapped drives use port 445 and the SMB 3.x protocol for communication.
- **Binary Large Object** (**Blob**): This provides cost-effective, massive scale-out unstructured data storage. The following are the three types of Blobs available:
  - **Page Blob**: This is used to store random-access data objects, such as VM disks
  - **Block Blob**: This is used to store ordered data objects, such as backups
  - **Append Blob**: This is used to store data objects, such as log files, consecutively as they are added
- **Table storage**: This provides a data store for non-relational (NoSQL) structured and semi-structured data.
- **Queue Storage**: This provides a data store for large amounts of asynchronous messages.

Selecting the right data storage solution to match your workload type is a key design decision to provide an operational solution at optimal cost and performance. You may need to implement different storage types in a single solution, as no Azure Storage service will meet all needs.

When a scenario requires mapped drives, Windows file shares, or SMB access, then the correct storage service to use would be Azure Files. Although Blob Storage can store files, it does not

provide SMB-based file shares and is not used for mapped network drives. Virtual machine OS and data disks always use Azure managed disks, not Blob containers.

File storage

File storage provides serverless **SMB file shares**. Serverless PaaS deployments mean there are no file servers to manage; the file-server layer has been abstracted so you can focus on creating shares and managing their access. These shares are highly available, fully managed network file shares that can be accessed using the traditional SMB protocol and from anywhere over port 445. You should ensure that this port and protocol are not blocked.

Azure Files supports a maximum file size of 4 TB and a maximum file share size of 100 TB (at the time of writing). The file shares created provide a traditional mapped drives storage solution to an existing solution, a lift-and-shift scenario. Azure Files can also be used for backup and disaster recovery scenarios for file servers located on-premises. Azure File Sync can replicate file shares from on-premises file servers, providing a cloud tiering capability.

Azure File Sync uses Azure Files to keep file shares centralized in Azure, but retains on-premises file servers to maintain existing operations, compatibility, and performance. Azure file shares can also be cached locally on on-premises file servers, so they are closest to the location where they are required. The file share data is still accessible locally via SMB, **Network File System** (**NFS**), and **File Transfer Protocol** (**FTP**).

The next section will look at container (Blob) storage.

Container (Blob) storage

Container storage provides massive-scale, cost-effective storage for unstructured data such as text, video, photos, files, and more. Containers are used to store Blob data objects.

Here are the three forms of Blob data that can be stored in a container:

- **Page Blob**: This is used to hold random-access files, such as objects that have been randomly written and read, in no particular order. An example would be VM disks (managed disks are stored in Microsoft storage accounts).
- **Block Blob**: This is used to hold text or binary files, such as objects that have been ordered, are consecutive, and are not random. An example would be backups.
- **Append Blob**: This is used for objects that have been consecutively added after the last piece of information stored in the Blob. An example of its use is for storing log files.

With this knowledge of container storage, you will now learn about Queue Storage.

Queue Storage

Asynchronous messages can be stored in **Azure Queue Storage**. The message queues are managed within an application programmatically. The programmatic management of message queues is done when a URL is used to access messages in Azure queues, and all requests are authenticated.

A single message can have a maximum size of 64 KB, and a message queue can have a maximum size of approximately 550 TB. A message is given a "time-to-live," which is, by default, set to seven days (this can be changed for messages added to a queue). A message is "deleted" from the queue when the time-to-live value expires.

The next section will look at disk storage.

Disk storage

Disk storage provides disks that can be attached to VMs. These provide the OS disk and temp disk used in VMs. Data disks may also be attached to VMs to provide additional storage capacity and additional volumes to meet disk layout needs, such as a VM that will run SQL Server. In addition, these data disks may be used as an installation path for applications or to store data that you cannot store on the system volume.

Azure Managed Disks is the recommended disk storage for Azure VMs for persistent data storage; Microsoft manages these disks.

The following are the types of managed disks that can be attached to a VM:

- **Standard HDD**: Low-cost storage
- **Standard SSD**: Consistent performance and low latency
- **Premium SSD**: High performance and low latency
- **Ultra disk**: Sub-millisecond latency

Selecting the appropriate storage type defines how data is stored and accessed once it resides in Azure. However, workloads must first be populated or migrated into those storage services. The next section, therefore, examines the tools and capabilities available for moving data into Azure Storage.

Moving data into Azure Storage

Several native Azure platform capabilities can copy (move) data into and out of Azure and between Azure resources.

The various Azure platform capabilities are as follows:

- **AzCopy**: This command-line utility helps you transfer data to or from a storage account.
- **Storage Explorer**: This GUI tool allows you to interact with your Azure storage resources. You can upload data and manage storage resources using Storage Explorer.
- **Azure Migrate**: This Azure discovery, assessment, and migration service can move your on-premises VM-based disk storage to Azure.
- **Databox**: This is an Azure service for transferring large amounts of data into Azure that may not be possible with other methods.
- **Azure File Sync**: This Azure service enables hybrid file shares. Windows Server can be utilized as a local cache for Azure file shares and provide centralized file storage in Azure while retaining on-premises local file servers.

While the previous sections focused on how Azure stores, protects, and moves data, designing an effective solution also requires choosing the appropriate data store based on the nature of the data itself.

The next section examines different data store types and how they align with structured, semi-structured, and unstructured data scenarios.

Data stores

Choosing the right **data store** to match your data type is a key design decision. No single storage solution fits all data types, and several stores might be needed to provide a complete and optimal solution. Your solution will largely be determined by the data type(s) you wish to store.

Here are the key factors in deciding on an optimal storage solution:

- How will you classify your data: structured, semi-structured, unstructured, streaming, relational, or non-relational data?
    - An example of structured data (also referred to as relational data) would be data stored in databases, such as a CRM system. This includes anything with a strict schema, such as most operational data stored in a SQL database or business data stored in a data warehouse for analysis and decision-making.
    - Examples of semi-structured data are key/value pairs, JSON files, and XML files.
    - Some examples of unstructured data include media files such as photos, videos, and audio; Office documents such as Word, PDF, and text; and log files.
- How will the primary operations use your data—analytical, transactional, read/write, search/ lookup, upload, change, and so on—carried out on each data type?

- How can you get the best performance out of your application?
- How can you get the best durability, availability, recovery, and security?
- How can you be the most cost-effective?

Figure 5.4 provides a simple data store selection guide; for clarity, not all the storage service options and decision points have been shown:

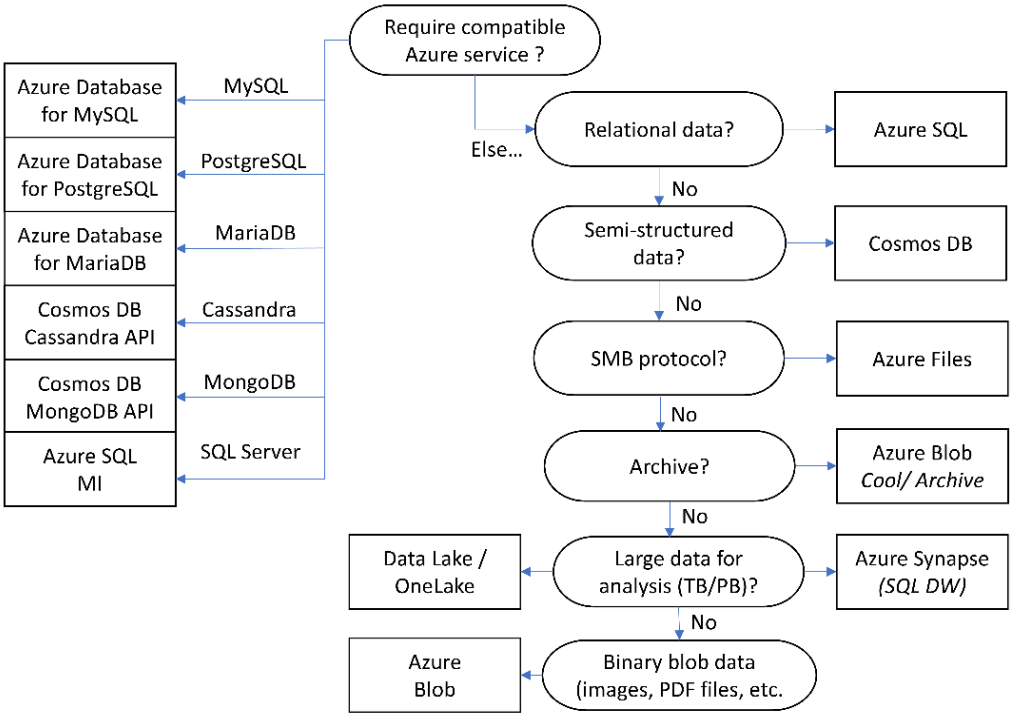


Figure 5.4: The flow for choosing a data store

Consider a company transforming and modernizing its application portfolio; its e-commerce platform processes orders and payments. Because this workload requires relational consistency and structured schema enforcement, the decision flow in *Figure 5.4* directs it toward **Azure SQL**.

The same organization collects global clickstream telemetry and user activity logs. This semi-structured, high-volume data aligns with **Azure Cosmos DB**, as indicated by the semi-structured branch in the diagram.

Product images, PDF invoices, and marketing assets represent large binary files. Following the binary data path in the figure leads to **Azure Blob Storage**.

Finally, historical transaction records must be retained for compliance, but are rarely accessed. In this case, the **Archive tier** within **Blob Storage** satisfies the durability requirement at minimal cost.

In this section, you learned about the Azure Storage services and grasped the concepts behind storage types, account types, tiers, replication, copying, and data stores. Together, these storage services illustrate how Azure enables organizations to store, protect, and access data efficiently across a wide range of workloads and access patterns.

Summary

This chapter addressed the *AZ-900 Azure Fundamentals* skills measured for *Describe Azure Storage Services* by exploring how Azure stores, protects, and manages data across different workload scenarios.

You examined the role of the storage account as the management and billing boundary for storage services, and how it governs replication, tiers, security, and monitoring. You compared core storage services, including Blob Storage, Azure Files, disk storage, and Queue Storage, and learned when each is appropriate based on access patterns and workload requirements.

You explored redundancy models such as LRS and ZRS, understanding how replication scope affects durability and availability. You also analyzed storage tiers and how access frequency directly impacts cost and retrieval characteristics, particularly in archive scenarios.

Finally, you examined data movement tools such as AzCopy, Azure Migrate, and Data Box, and learned how to choose appropriate data stores based on whether data is relational, semi-structured, or unstructured.

From an exam perspective, focus on identifying the correct storage service for a given workload, understanding replication copy counts and regional scope, recognizing tier access limitations, and distinguishing between data store types.

In the next chapter, you will look at **Azure networking services**.

Exam Readiness Drill — Chapter Review Questions

Apart from mastering key concepts, strong test-taking skills under time pressure are essential for acing your certification exam. That's why developing these abilities early in your learning journey is critical.

Exam readiness drills, using the free online practice resources provided with this book, help you progressively improve your time management and test-taking skills while reinforcing the key concepts you've learned.

HOW TO GET STARTED

- Open the link or scan the QR code at the bottom of this page
- If you have unlocked the practice resources already, log in to your registered account. If you haven't, follow the instructions in *Free Benefits with Your Book* section in this book's *Preface* and come back to this page.
- Once you log in, click the START button to start a quiz
- We recommend attempting a quiz multiple times till you're able to answer most of the questions correctly and well within the time limit.
- You can use the following practice template to help you plan your attempts:

Working On Accuracy		
Attempt	**Target**	**Time Limit**
Attempt 1	40% or more	Till the timer runs out
Attempt 2	60% or more	Till the timer runs out
Attempt 3	75% or more	Till the timer runs out
Working On Timing		
Attempt 4	75% or more	1 minute before time limit
Attempt 5	75% or more	2 minutes before time limit
Attempt 6	75% or more	3 minutes before time limit

The above drill is just an example. Design your drills based on your own goals and make the most out of the online quizzes accompanying this book.

First time accessing the online resources?

You need to unlock the online practice resources to access the link below. Check the *Free Benefits with Your Book* section in the *Preface* for detailed instructions on how to do this. You only need to unlock the resources once.

Open Quiz `https://packt.link/AZ-9003ech5` or scan the QR code

6

Azure Network Services

In *Chapter 5*, *Azure Storage Services*, you gained knowledge of the Azure platform's core components. This chapter builds on those foundations by examining how **Azure networking services** connect, isolate, and secure resources across Azure, the internet, and on-premises environments.

This chapter primarily focuses on the *Describe Azure architecture and services* module from the *Skills Measured* section of the AZ-900 Azure Fundamentals exam.

This chapter builds your understanding of how Azure networks are structured, how traffic moves within and between networks, and how Azure connects to on-premises environments and the internet. You will learn how virtual networks define boundaries, how routing controls traffic flow, how connectivity options such as VPN Gateway and ExpressRoute extend your infrastructure, and how DNS enables name resolution across hybrid environments.

By mastering these foundational concepts, you will not only be prepared for exam questions but also able to interpret real-world Azure network designs, understand hybrid connectivity diagrams, and make informed decisions about connectivity, segmentation, and traffic control.

By the end of this chapter, you will be able to confidently answer questions on the following topics:

- Azure networking overview
- Virtual networks and subnets
- Virtual Private Network gateways
- Name resolution in Azure

While the focus remains aligned to the AZ-900 exam objectives, this chapter also strengthens your understanding of how Azure networking components work together in practical deployment scenarios.

Azure networking overview

You must know about the following **Azure network services** since they are outlined in the exam:

- **Virtual Network (VNet)**
- VNet peering
- **Virtual Private Network (VPN)** gateway
- ExpressRoute
- Azure DNS

The services listed represent the core Azure networking components required for the AZ-900 exam. To understand how these services relate to one another in practice, it is helpful to view them as part of a layered networking model. The following diagram illustrates how segmentation, connectivity, traffic control, and name resolution combine to form a complete Azure networking architecture.

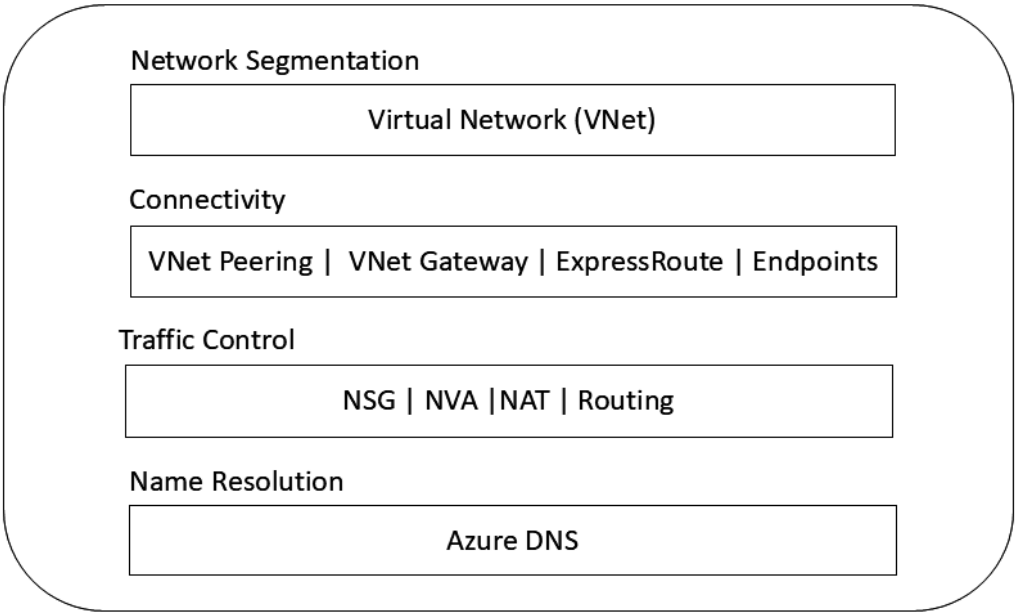


Figure 6.1: Azure networking services overview model

Figure 6.1 presents a layered view of Azure networking. The exam-scoped services, such as virtual networks, VNet peering, VPN Gateway, ExpressRoute, and Azure DNS, form the foundational elements, while additional components, including **Network Security Groups**

(**NSGs**), routing, NAT, and **Network Virtual Appliances** (**NVAs**), provide traffic inspection, filtering, and control. Together, these elements deliver secure and structured communication across Azure resources and hybrid environments.

Azure networking provides the following capabilities:

- **Communication between Azure resources**: Communication paths **between Azure resources**, such as VMs, are provided through **Virtual Networks** (**VNets**). Communication **between VNets** is enabled using **VNet peering** (within the same region and across regions). Additionally, **service endpoints** can be used to enable optimized connectivity from VNets to supported PaaS resources such as storage services and database services.
- **Communication with on-premises resources**: You can extend your local on-premises networks into Azure through **Virtual Private Networks** (**VPNs**), which provide encrypted connections over the internet, or ExpressRoute, which allows private, low-latency connections that bypass the internet.
- **Communication with third-party services**: You can communicate with other cloud providers, carriers, hosting and data center service providers, and software vendors, for example, to access the APIs or services offered by these third parties.
- **Communication with the internet**: Azure can be used as an **internet breakout** point for Azure resources or for on-premises purposes to access Azure resources. Azure resources, such as VMs, can accept incoming internet traffic through a public IP address. However, all VMs will typically have private IP addresses for security purposes, and the public IP address should be assigned via a load balancer, **Network Address Translation** (**NAT**) Gateway, or other internet-facing gateway resource. This approach improves security and control by ensuring that outbound connectivity is intentionally configured rather than automatically available.

Note

At the time of writing, Microsoft is retiring **default outbound access** for Azure virtual machines, which means outbound internet connectivity should be explicitly designed using Microsoft services such as **NAT Gateway**, a **public load balancer**, or **Azure Firewall** rather than being implicitly available from a subnet.

Read more at `https://azure.microsoft.com/en-us/updates?id=default-outbound-access-for-vms-in-azure-will-be-retired-transition-to-a-new-method-of-internet-access`.

- **Isolation and segmentation of network traffic**: VNets are *communication boundaries* that provide *isolation* of the address space(s) (also referred to as prefixes) used within that VNet. To allow VNets to communicate, a VPN gateway, **Network Virtual Appliance** (**NVA**), or VNet peering must be used.
  In addition, a VNet's address space can be segmented with subnets to allow traffic filtering and routing to be defined at the subnet level. Typically, a VNet will contain one or more private address spaces, although public prefixes are supported.
- **Routing of network traffic**: Azure uses a set of default system routes or traffic direction/destination paths. You can think of this as a set of defined **Satellite-Navigation** (**Sat-Nav**) traffic routes that define how to get to a destination when coming from a given location. These routes will direct traffic between the subnets of all interconnected VNets to on-premises networks and the internet; no traffic inspection or filtering is applied by default. You can also apply **User-Defined Routes** (**UDRs**), which you will read about in the *Internal Virtual Network (VNet) routing* section.
- **Filtering of network traffic**: By default, no traffic filtering occurs within a VNet. However, a **Network Security Group** (**NSG**) can be used, which is a resource that allows you to define inbound and outbound allow and deny rules. You can filter this traffic based on factors such as source IP, destination IP, port, protocol, and more. Alternatively, you can use an NVA—a VM that runs network routing and traffic control/filtering software provided by a third party (Barracuda, Fortinet, and WatchGuard, to name a few); this acts as a firewall/packet filter in this scenario.
- **Encryption of network traffic**: Through a **VPN gateway**, traffic that has been sent and received from on-premises or another VNet can be encrypted.

- **Name resolution services through DNS services**: Azure has an integrated name resolution service for all resources in a VNet. You can configure the VNet with a *custom DNS entry* for any internal or external DNS server you have. For example, you can set the custom entry to the IP addresses of domain controllers if these handle the DNS for your VMs.
- **Cost implications**: No traffic *entering* (ingress) a VNet within a region is billed, but all traffic *leaving* (egress) a VNet and region is; this also applies to VNet peering; traffic *between* VNets will be billed for egress, not ingress.

> **Tip**
>
> For exam scenarios, use the rule of thumb: If traffic *crosses regions*, assume *egress costs*.

In modern Azure architectures, private endpoints are commonly used to access PaaS services privately from a virtual network across the Microsoft backbone without exposing a public endpoint. If the scenario involves extending connectivity between Azure and an on-premises environment, a VPN gateway is used when encrypted connectivity over the internet is sufficient, whereas ExpressRoute is selected when private, low-latency connectivity that bypasses the public internet is required. When the focus shifts to isolating or segmenting workloads within Azure, VNets and subnets define the network boundaries, with routing behavior determined by system routes or user-defined routes.

Traffic control requirements then determine the security component; NSGs are used for basic inbound and outbound allow-and-deny filtering at the subnet or network interface level, while scenarios that require centralized inspection or control across multiple subnets or VNets point to an NVA.

Next, we'll take a closer look at Azure VNets.

Virtual networks and subnets

An Azure **Virtual Network** (**VNet**) is an IaaS resource that enables communication with Azure resources. A VNet represents a software-defined, single-tenant, private network in a single Azure region that your resources communicate on. A VNet is dedicated to you and is not shared with other tenants in Azure. Resources in the Azure VNet can access the internet and on-premises resources via public and private networks when appropriate connectivity services, such as NAT Gateway, VPN gateways, or firewalls, are configured.

Users and on-premises resources can also access and communicate with resources in the Azure VNet via these public and private networks. You may have one or more VNets; each can be

connected or isolated to meet your goals. In addition, several connectivity options support inter-VNet and cross-premises connectivity scenarios.

Figure 6.2 visualizes the VNet and connectivity approach:

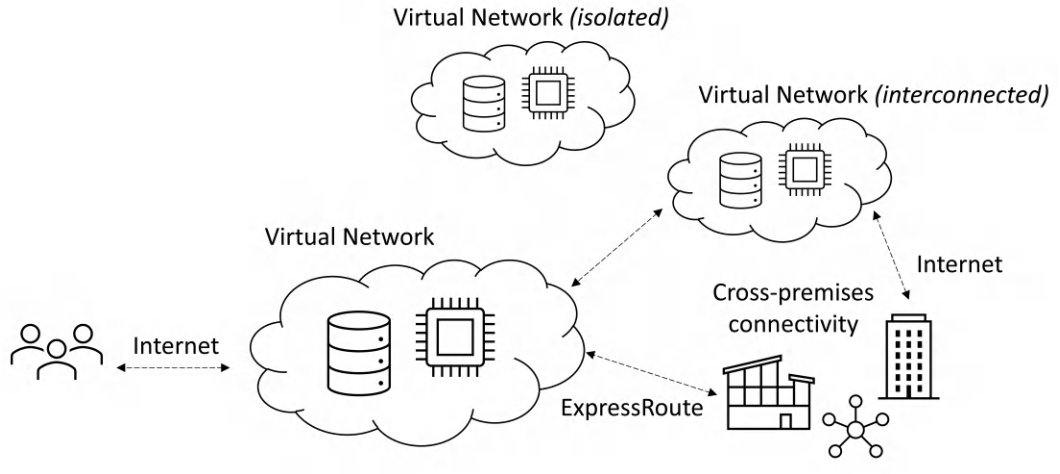


Figure 6.2: Azure virtual networks schema showing communication between resources

Figure 6.2 illustrates how a VNet acts as a private boundary for Azure resources while still enabling controlled connectivity. The diagram shows isolated VNets, interconnected VNets using peering, internet connectivity, and cross-premises connectivity using services such as ExpressRoute and VPN. It demonstrates that while VNets provide isolation by default, connectivity can be deliberately configured to extend communication between networks and external environments.

The following are some characteristics of VNets:

- VNets act as **communication boundaries**; by default, Microsoft routes traffic to all the resources in the **same** VNet to allow them to communicate with each other. However, if you wish to ensure that, by default, the VMs don't communicate (such as for compliance reasons), place each VM in a **separate** VNet.
  If you want the VMs in different VNets to communicate with each other, you must ensure that VNet peering, or a VPN, is used to connect them.
- You can control traffic into a VNet using a VPN gateway (one per VNet), an NSG, or a firewall. A VPN gateway is a **per-VNet** resource, while **NSGs** apply at the **subnet** or **network interface** level.
  If you have multiple VNets to control traffic, then an Azure firewall or third-party NVA is the most efficient option. This has a one-to-many ratio in that one firewall or NVA

can control access to multiple VNets across multiple subscriptions. An NSG cannot be applied at the VNet level, but only at the subnet or network interface level.

- A VNet belongs to a resource group that belongs to a subscription and can only be part of one region. VNets can span across data center zones, but are not available across regions.
- VMs in different subscriptions can communicate with each other so long as they are part of the same VNet or VNet peering, or if a VPN gateway connects the VNets and they don't have conflicting IP ranges.
- All traffic entering (ingress) a VNet and region is **"not billed**," but all traffic leaving (egress) a VNet and region is **"billed**." This also applies to VNet peering, since the traffic between these VNets will be billed for egress, not ingress.
- VNets contain one or more **IP address spaces**. **Private IP prefixes** are typically used, but **public address spaces** can also be used.
- **VNet address spaces** can be segmented into **smaller subnets**, the same as in on-premises networks. However, note that Azure networking does not allow control at **Layer 2** (Data Layer | MAC address) of the **Open Systems Interconnect** (**OSI**) model; only **Layer 3** (Network Layer | IP address) can be controlled. The OSI model of network functions is beyond the scope of this book and exam objectives, but it has been mentioned here to give a fuller picture.
  If you are connecting Azure VNets to on-premises networks, you should ensure that these addresses **do not overlap** and are **not duplicated**. For example, you must ensure that the same matching address space(s) do not exist on-premises. In this scenario, you should create Azure VNets with unique address spaces that differ from those used on-premises.

A **VNet** defines the private network boundary and IP address space in which Azure resources operate and is required for virtual machine networking. **Subnets** are used to segment a VNet into smaller network areas for workload separation. **NSGs** do not provide network isolation; they are used only to allow or deny traffic at the subnet or network interface level.

To fully understand how VNets provide isolation and controlled communication, you must also understand how they are structured internally. This leads directly to VNet segmentation, where address spaces are divided into subnets to separate workloads and control traffic flow.

VNet segmentation

VNets, in their simplest form, are a set of communication paths that interconnect systems and other resources and services. You can think of them as **corridors** or **elevators** in a building. There are always entry and exit points in buildings; these are **doors**. But in VNets, they can be considered gateways; there are external doors on the buildings and internal doors that separate rooms from corridors. You also have different floors, and you may even have buildings connected by interconnecting walkways or skyways in a campus environment.

In VNets, you can create similar constructs to **segment a network** into smaller sections called **subnets**. In the building example, a **VNet** is a *building*, an **address space** (also referred to as an address prefix) is a *floor* (VNets can have multiple address spaces, similar to how a building can have multiple floors), and the **subnets** are *rooms*. These **rooms (subnets)** are then all connected by *corridors*, *stairs*, and *elevators*, which act as the (*human*) *traffic paths*.

When you create a VNet, you define an **address space**; this can be broken down into **subnets**. Each resource within the VNet will be assigned an **IP address**. *Figure 6.3* visualizes this concept of VNets and segmentation:

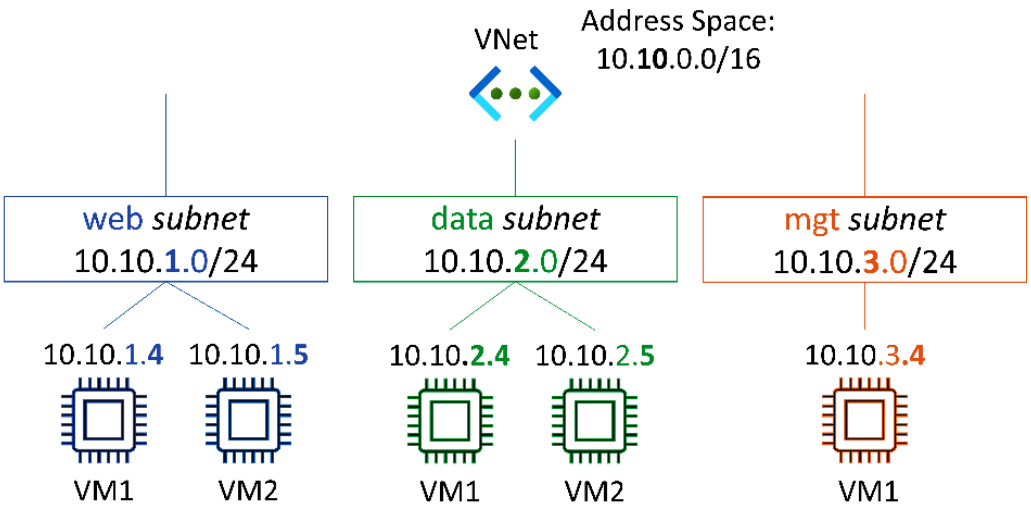


Figure 6.3: The concept of segmenting a VNet into subnets

Figure 6.3 shows an address space created using the **Classless Inter-Domain Routing** (**CIDR**) format of `10.10.0.0/16`. This has been further segmented into **smaller subnets** (`/24`), with each resource **assigned an IP address** from this **address range**.

By default, all traffic can be routed between all the resources in all subnets, routed to other VNets, and routed externally to the internet and on-premises. You can define your routes based

on how traffic travels to its destination through UDRs. It is recommended that you use the following **private non-routable** (RFC1918) address ranges with Azure VNets:

- `10.0.0.0` to `10.255.255.255` (10/8 prefix)
- `172.16.0.0` to `172.31.255.255` (172.16/12 prefix)
- `192.168.0.0` to `192.168.255.255` (192.168/16 prefix)

Azure **reserves five IP addresses** in any subnet. When creating a VM, you will notice that the **first IP assigned** in any given subnet will be a .4 address because Azure reserves the **first four IP addresses** (ranges start at 0, not 1) and the **last IP address**.

They are detailed as follows:

- The **first IP address** is reserved for the VNet's network address—that is, `x.x.x.0`.
- The **second IP address** is reserved for the VNet's default gateway—that is, `x.x.x.1`.
- The **third** and **fourth IP addresses** are reserved for the VNet's Azure DNS—that is, `x.x.x.2` and `x.x.x.3`, respectively.
- The **fifth IP address** is reserved for the VNet broadcast address—that is, `x.x.x.255`.

  The following section looks at internal routing.

These reserved addresses ensure consistent internal routing and platform-level functionality within each subnet. With the subnet structure now defined, the next step is to understand how Azure determines the path that traffic takes between resources.

Internal Virtual Network (VNet) routing

As you learned earlier in this chapter, VNets have a set of communication paths that interconnect systems. Azure uses a set of default **"system routes"** or traffic direction/destination paths that allow resources to communicate with each other.

You cannot edit or remove these routes, but you can create custom route tables to control (redirect) the direction of traffic (packets). This is called **UDR** and allows you to override the default Azure system routes. Again, similar to the Sat-Nav analogy, this allows you to load in your custom traffic directions to override the manufacturer's predefined ones. You can apply UDR through a **"route table"** resource within a VNet, combined with an NVA. This approach allows network segmentation and traffic control to occur much as they would in a traditional network.

Figure 6.4 outlines this approach:

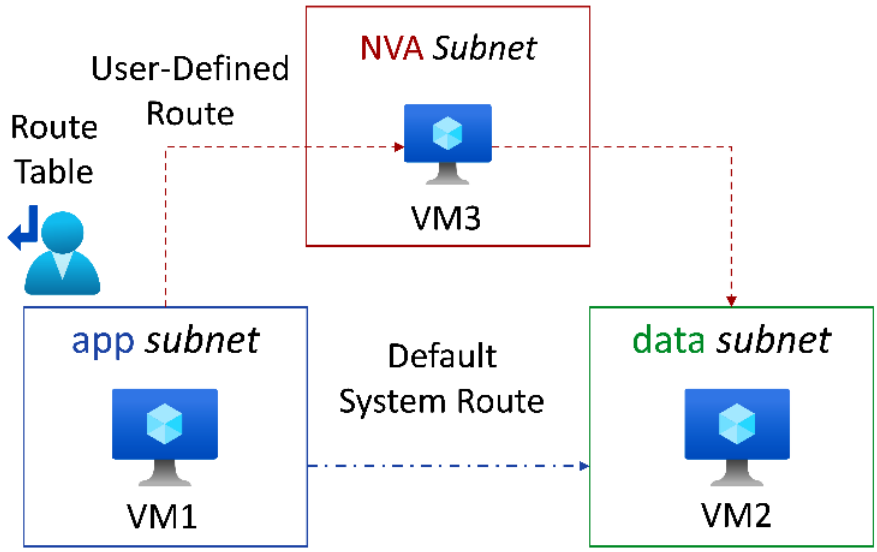


Figure 6.4: VNet showing internal routing and segmentation through the use of an NVA

Figure 6.4 shows that, in the default system routing, there is a communication path between VM1 and VM2. This could mean that if VM1 were to be compromised, it could contact VM2. However, if you apply a route table to the subnet where VM1 is, then any traffic leaving VM1 will be overridden by the UDR route table and directed to VM3. VM3 will then inspect the traffic and either block or allow the traffic based on the security rules set on the appliance (firewall). In this scenario, if VM1 is compromised, it will no longer communicate directly with VM2 and must be passed to VM3 first.

While internal routing determines how traffic flows within a VNet, Azure must also decide how traffic exits its network boundary and reaches external destinations. This shift from internal communication to outbound path selection introduces the concept of external VNet routing.

External Virtual Network (VNet) routing

External routing is where traffic leaving (egressing) Azure has its path to its destination determined by **cold-potato routing**; the alternative to this is **hot-potato routing**.

These terms are defined as follows:

- **Cold-potato routing**: This ensures that traffic remains on the Microsoft global network for as long as possible before it is handed off to a downstream **"ISP edge" Point of Presence** (**PoP**) to be delivered to its final destination. This is the most

expensive form of egress data from Microsoft regions but has the lowest latency, the best quality, and the fastest performance. This can be likened to a plane courier versus a boat postal service.

- **Hot-potato routing**: This means that traffic leaves (egresses) the Microsoft global network at the earliest opportunity and travels to its destination via the internet, which means passing the packet between points (hops) on the internet multiple times. When the packet arrives at a point, it is passed on as soon as possible, if not immediately. This is where the analogy of *passing a hot potato* comes from. This is the cheapest option for data egress as it does not stay on the Microsoft network for any length of time. It exits onto the public network as quickly as possible and is the closest point to the data center. It is delivered by a cheaper, slower, and less reliable boat postal service.

Figure 6.5 outlines the hot- versus cold-potato routing approach. In the diagram, the cold-potato-routed traffic gets delivered to its final destination quicker with lower latency and more reliability as it stays on the Microsoft global backbone network for longer:

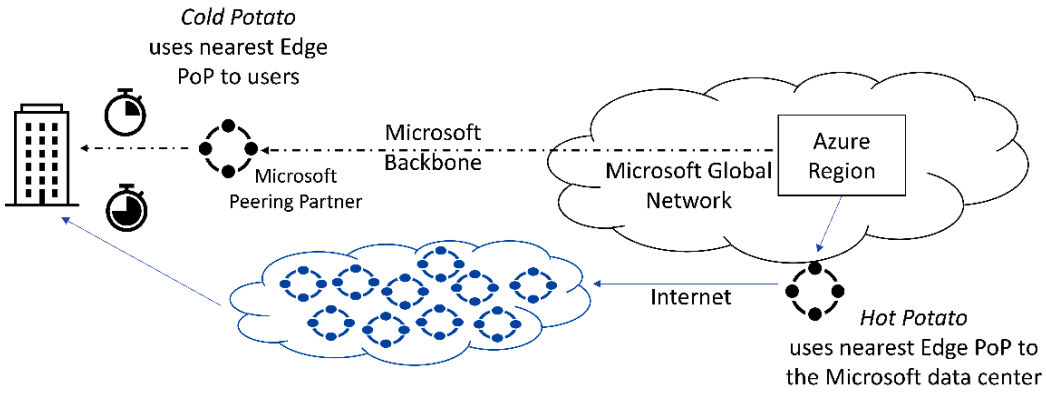


Figure 6.5: Hot versus cold-potato routing to carry traffic to its destination

Figure 6.6 likens this to delivering a packet to its final destination by using the perspective of air travel versus shipping:

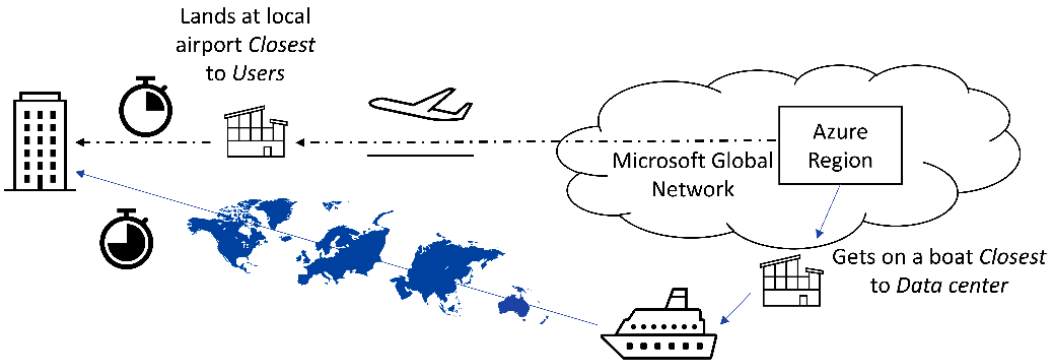


Figure 6.6: Air travel versus shipping routes—similar to hot versus cold-potato routing

Figure 6.6 reinforces the routing concept by illustrating how different transport paths affect delivery speed, cost, and reliability. In Azure, these routing decisions determine how traffic exits a region and reaches external destinations.

While external routing governs how traffic leaves Azure, communication within Azure regions can also be optimized. Rather than traversing the public internet, virtual networks can be directly interconnected over the Microsoft backbone. This introduces the concept of virtual network peering.

Virtual network peering

VNet peering allows two or more VNets to be connected seamlessly so that, from a communication point of view, the traffic flow appears as though it is on the same network. Unlike a VPN gateway whose traffic must be routed over the internet, traffic between the VNets, when using VNet peering, is routed over the Microsoft backbone, meaning fast, reliable, low-latency, and secure connectivity between your resources in different VNets. This is the same as traffic being routed between resources within the same VNet.

Figure 6.7 visualizes this inter-VNet connectivity:

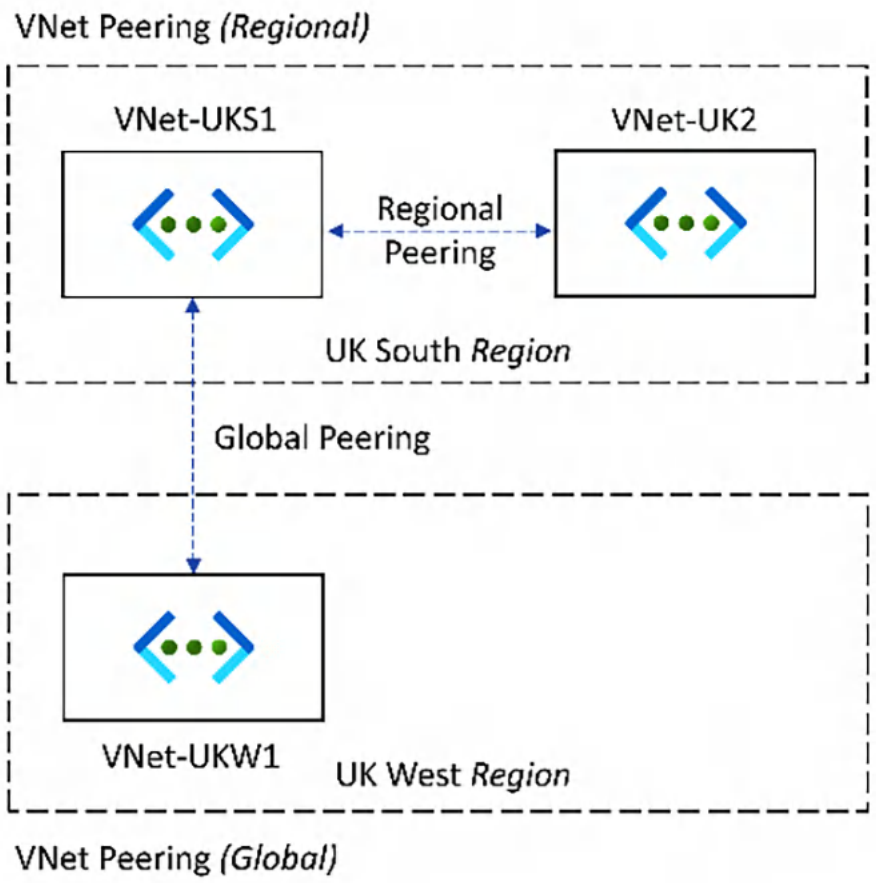


Figure 6.7: Inter-VNet connectivity shown using peering

VNet peering also supports **regional** and **global** VNet peering, as shown in *Figure 6.7*.

They are detailed as follows:

- **Regional VNet peering**: This is used to connect VNets from the same region
- **Global VNet peering**: This is used to connect VNets from different regions

As seen in *Figure 6.7*, a hub-and-spoke design is the most common deployment topology. This design is widely used in modern Azure-architecture-frameworks-designed environments to centralize shared services such as firewalls, VPN gateways, and routing controls while keeping application workloads isolated.

Understanding virtual networks, subnets, routing behavior, and peering establishes the foundation of Azure networking within and between regions. However, many environments must also securely extend connectivity beyond Azure to on-premises networks. To achieve this, Azure provides gateway services that enable encrypted communication between the cloud and local infrastructure.

Virtual Private Network gateways

A **Virtual Private Network (VPN) gateway** is a service that allows encrypted traffic to be sent through private and secure connections between resources in an on-premises location and an Azure VNet by using the internet. This allows users and resources to seamlessly connect as though they were part of the same local network.

A VPN gateway requires several resources to be deployed;

- VNet
- Gateway subnet
- Public IP address
- Local network gateway

They are detailed as follows:

- **VNet**: A VNet can only have one VPN gateway, which is a resource that can be associated with only one VNet. When connecting Azure VNets to on-premises networks, you should ensure that these addresses do not overlap, are not duplicated, and that the same matching address space does not exist on-premises. In this scenario, you should create Azure VNets with a unique address space that differs from those used on-premises.
- **Gateway subnet**: A dedicated subnet named `GatewaySubnet` is created within the VNet. The VNet address space must have enough space available to allocate this. A /27 (/26 ...) address or larger is recommended since this ensures enough IP addresses are available.
- **Public IP address**: This resource is the public IP address used to connect to the on-premises VPN device (or another VNet's gateway).
- **Local network gateway**: This resource is used to represent your on-premises network location. It defines the VPN appliance that connections will be made through, as well as the network address spaces (also referred to as address prefixes) that you will be connecting from to establish a connection to the Azure resources. A local network

gateway can specify a single on-premises network address space or multiple network address spaces.

    - You should be able to distinguish between the virtual network gateway and the local network gateway. The virtual network gateway is deployed inside the Azure VNet and must reside in a dedicated subnet named `GatewaySubnet`; the local network gateway represents the on-premises environment by defining the on-premises VPN device public IP address (or FQDN) and the on-premises address spaces.
    - The local network gateway is configured with an IP address or the **Fully Qualified Domain Name** (**FQDN**) of the on-premises VPN appliance you will make a connection to. You also have to specify the IP ranges from the local on-premises network location that you will connect from.
    - You can modify your local network gateway to reflect any changes in your on-premises network. For example, you can add or remove address spaces (prefixes) if you added a new on-premises network and now need to connect to Azure resources over the VPN associated with this local gateway, and then add these new IP spaces to allow this communication.
    - Suppose you needed to prevent a network from communicating, you could remove these on-premises network spaces from the local gateway. By doing this, they will no longer be able to access resources in Azure over the VPN.

- **Connection**: This resource creates a logical connection between the local network gateway and the VPN gateway.

Figure 6.8 shows these resources and their relationships:

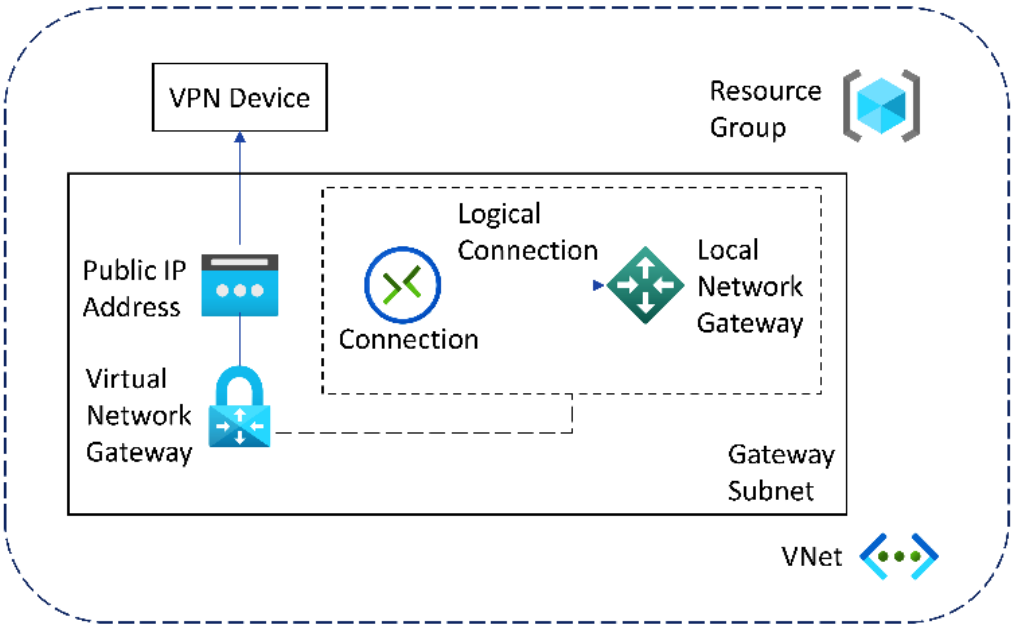


Figure 6.8: VPN gateway resources allowing encrypted traffic to be sent through secure connections

A VPN gateway supports both the **Point-to-Site** VPN and **Site-to-Site** VPN connectivity methods, each serving different architectural needs. In the next few sections, you will look at these two connectivity methods in more detail.

Having established how Azure connects networks at a broader level, it is logical to begin with the simplest use case: enabling secure access for individual users connecting from remote locations. This scenario introduces the Point-to-Site VPN model.

Point-to-Site VPN

This allows you to make a **"secure, encrypted connection"** to Azure resources using a single user from a client device. This is intended for remote users who are not connected to the corporate network (and cannot use a Site-to-Site VPN), such as home workers or those traveling; the **OpenVPN**, **SSTP**, and **IKEv2** protocols are used.

Figure 6.9 shows a Point-to-Site VPN:

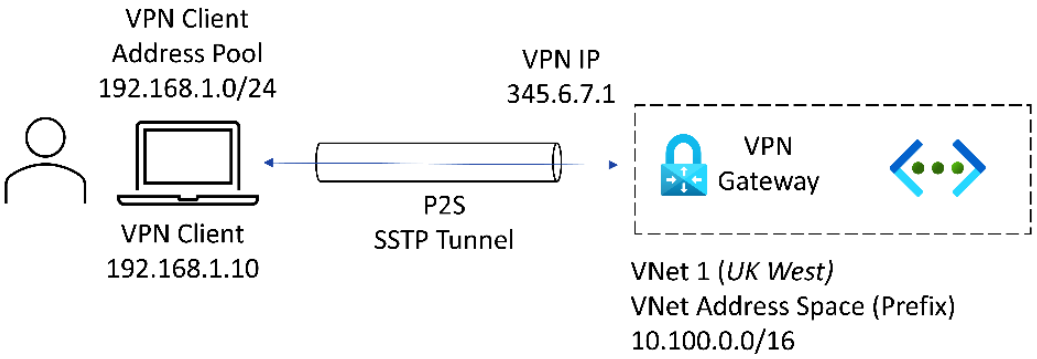


Figure 6.9: Point-to-Site VPN

In *Figure 6.9*, a user initiates a VPN session through a VPN client to connect them to the Azure VNet. The user's device is allocated an IP address from the VPN client address pool, which is used for communication with Azure VNet resources.

While a Point-to-Site VPN enables secure access for individual users and devices, it does not address scenarios where an entire on-premises network must connect to Azure. Organizations often require persistent, network-to-network connectivity to support hybrid workloads, shared services, and centralized infrastructure. This broader requirement introduces the Site-to-Site VPN model.

Site-to-Site VPN

This type of VPN allows you to make a secure, encrypted connection to Azure resources in a cross-premises scenario to support a hybrid connectivity solution. The Azure VPN gateway service provides both **"policy-based"** VPNs (also referred to as static routing) and **"route-based"** VPNs (also referred to as dynamic routing).

Policy-based VPNs may be considered for more legacy scenarios since this is the older approach and is generally used to connect on-premises devices that only support **policy-based** (*static*) connections. This is typical since firewalls provide more VPN functionality than a fully-fledged and dedicated VPN device. Due to this, **route-based VPNs** (**dynamic**) are the recommended approach.

Both support the **Internet Protocol Security** (**IPSec**) protocol, known as **Internet Key Exchange** (**IKE**)—both versions 1 and 2. The only authentication method that is supported is **"pre-shared keys**."

Figure 6.10 shows how to connect an on-premises network to Azure using a Site-to-Site VPN. This topology requires a VPN device to be located at the on-premises site, and it must have a public IP address:

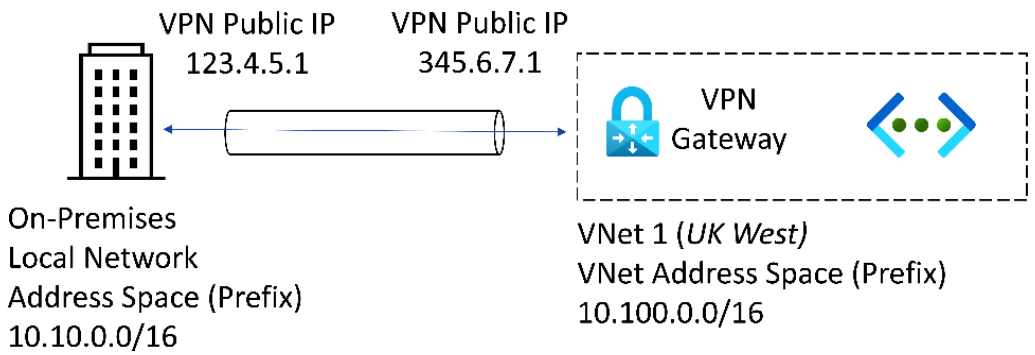


Figure 6.10: Site-to-Site VPN

Figure 6.10 demonstrates the traditional Site-to-Site VPN scenario, where an on-premises network establishes an encrypted IPSec tunnel to an Azure VNet through a VPN gateway. This model is commonly used to extend corporate infrastructure into Azure as part of a hybrid deployment.

The same Site-to-Site VPN mechanism can also be used to connect two Azure virtual networks when peering is not suitable or when encryption over IPSec is specifically required.

Figure 6.11 illustrates how to connect two Azure VNets using a Site-to-Site VPN:

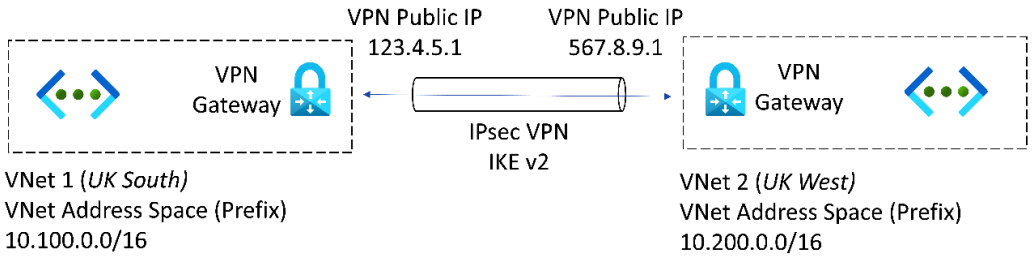


Figure 6.11: VNet-to-VNet VPN

The alternative to using a Site-to-Site VPN to connect two Azure VNets is to use VNet peering. However, this uses private addresses and does not encrypt the traffic between the two VNets. This is a private connection that stays on the Microsoft backbone network, so it may not be a suitable alternative requirement.

While Site-to-Site VPN provides secure connectivity over the public internet, it may not meet the performance, reliability, and bandwidth requirements of large or mission-critical workloads. Organizations that require predictable latency, higher throughput, and private

connectivity that does not traverse the public internet often need a more dedicated solution. This requirement introduces ExpressRoute.

ExpressRoute

ExpressRoute (using an **ExpressRoute circuit**) is a true extension of your on-premises networks into Microsoft's data centers. It provides a fast, low-latency, enterprise-quality (reliable, stable) connection to your Azure resources for cross-premises scenarios.

Unlike a VPN network, traffic does not route (traverse) the internet. Traffic is carried over a privately managed network between your resources and Microsoft's data centers. This network is facilitated by a global telecom carrier, which acts as the connectivity provider. As expected, this enterprise-grade connectivity has built-in redundancy that uses dynamic routing and the **Border Gateway Protocol** (**BGP**).

Figure 6.12 outlines the topology of an ExpressRoute circuit connection using a connectivity provider alongside a Site-to-Site VPN connection:

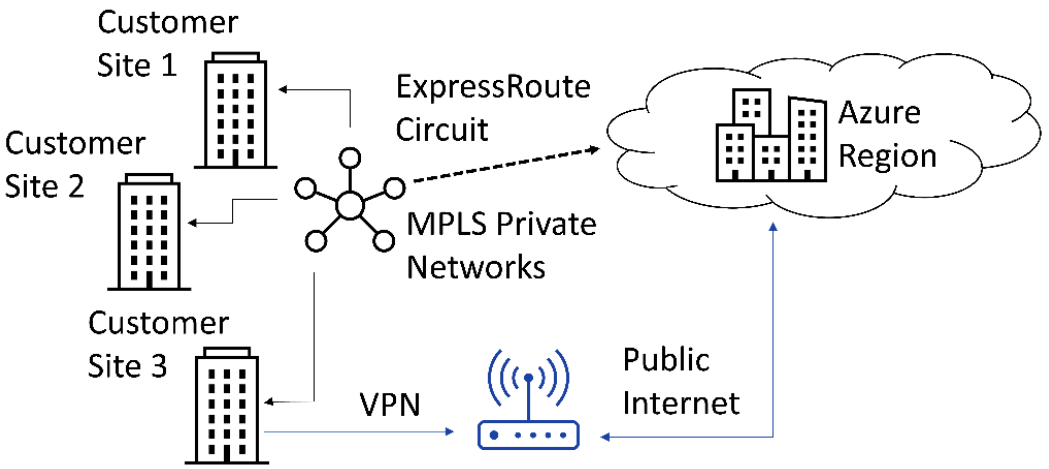


Figure 6.12: ExpressRoute showing the connection for cross-premises to Azure resources

As you can see, by working with a connectivity provider, ExpressRoute circuits can make Azure data centers appear just like any other site on an MPLS circuit by extending the on-premises network's reach globally and the data center's capacity.

In many modern hybrid designs, ExpressRoute is combined with a Site-to-Site VPN to provide failover connectivity if the private circuit becomes unavailable.

In scenarios that have requirements that refer to private, dedicated connectivity that does not traverse the public internet and explicitly mention OSI Layer 3 or BGP, the correct service is ExpressRoute.

While networking establishes how traffic moves between resources, name resolution determines how those resources are located in the first place. Without DNS, systems cannot reliably translate human-readable names into IP addresses, regardless of how well the network itself is designed. For this reason, understanding name resolution is a necessary complement to the connectivity and routing concepts discussed so far.

Name resolution in Azure

Name resolution in Azure can be provided by a traditional **Windows Server DNS** running on a VM or by **Azure DNS**, which is a managed service that is fully VNet-integrated to provide both **"public"** and **"private"** zones.

Azure DNS is a Microsoft-managed **DNS-as-a-Service** solution for the name resolution of Azure resources. You do not need to provide your own DNS servers for name resolution. In most modern Azure deployments, Azure DNS is preferred for simplicity and availability, with custom DNS servers used only when advanced features such as conditional forwarding or zone transfers are required.

The Azure DNS service provides hosting for both your **public DNS zones** and **private DNS zones**, for which Azure can be authoritative. The following are the DNS services available:

- **Public DNS**: This hosts your DNS domains and provides name resolution for internet-facing domains
- **Private DNS**: This allows for hostname resolution within a VNet and between VNets

Once your DNS zones are *"migrated* "to Azure DNS, Microsoft's DNS name servers will respond to queries for resources in these zones. **Anycast DNS** is used by the DNS service, which means that the query will be responded to by the DNS server that is closest geographically to the query. The **Azure portal**, **Azure PowerShell**, or **Azure CLI** can create and manage the Azure public and private zones.

The following outlines the core steps to implement a public zone:

1. Create a public zone.
2. Add records to the zone.
3. Validate name resolution for the zone.

The following outlines the core steps to implement a private zone:

1. Create a private zone.

2. Link (publish) the zone to the VNets for name resolution using the Azure DNS service.
3. Add records to the zone if required and enable "auto-registration."
4. Validate name resolution for the zone.

At this stage, you have seen how public and private DNS zones are implemented and integrated with Azure Virtual Networks. These configurations form the foundation for how resources are discovered and accessed within both internet-facing and internal environments. Understanding how zones are created and linked is essential before examining the capabilities, limitations, and design considerations that influence how DNS should be deployed in real-world architectures.

Azure DNS private zones

Once a private DNS zone is created and linked to one or more VNets, it becomes the internal name resolution authority for those virtual networks. While public zones focus on internet-facing resolution, private zones address internal name resolution within Azure and hybrid environments.

The capabilities of Azure DNS private zones include the following:

- Automatic registration from a private zone linked to the VNet of VMs
- DNS resolution forwarding across private zone-linked VNets
- Reverse DNS lookup within the scope of the VNet

These capabilities make private zones particularly useful for internal service discovery, application tier separation, and hybrid scenarios where name resolution must remain isolated from public DNS. However, Azure DNS is not a full replacement for every traditional DNS scenario, which leads to some important limitations.

Azure DNS drawbacks

The following are some limitations of Azure DNS:

- There is no support for **conditional forwarding**. You should use your own DNS server in place of this if this capability is required.
- There is no support for **Domain Name System Security Extensions** (**DNSSEC**). You should use your own DNS servers in place of this if this capability is required.
- There is no support for **zone transfers**. You should use your own DNS servers in place of this if this capability is required.
- Reverse DNS is only possible with private address spaces linked to the VNet.

- Multiple VNets can be linked to a private DNS zone, although only one VNet can be configured for auto-registration.
- There is a limit to the number of zones and records per subscription. Next, you will look at Windows Server DNS hosted on Azure VMs.

These limitations highlight that Azure DNS is designed for managed, cloud-native name resolution scenarios. When requirements extend to advanced forwarding logic, DNSSEC, zone transfers, or full traditional DNS server control, organizations may need to deploy and manage their own DNS infrastructure. One common approach is to run Windows Server DNS on Azure virtual machines, which provides the flexibility of a traditional DNS server while remaining integrated within the Azure network.

Windows Server DNS on Azure VMs

The **DNS server role** can be installed on Azure VMs running the Windows Server OS using Server Manager in the same way as an on-premises installation. You can specify that a VNet uses your own Windows DNS servers instead of Azure DNS; these DNS servers could be Azure VM DNS servers or on-premises DNS servers connected to the Azure VNet.

You may decide to implement Windows Server DNS on Azure VMs to replace (or in addition to) Azure DNS in the following scenarios:

- When you have a need for name resolution between VNets
- When you have a need for name resolution of Azure resources from on-premises
- When you have a need for conditional forwarders
- When you have a need for zone transfers

Your own Windows DNS servers can be set in the Azure VNet **DNS servers Custom** setting, as shown in *Figure 6.13*:

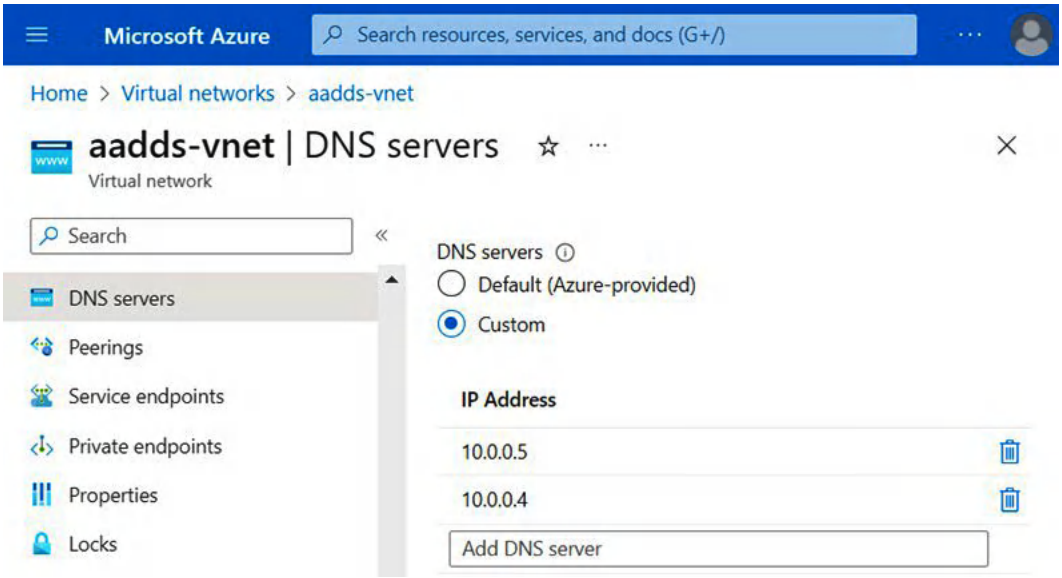


Figure 6.13: Custom DNS server setting

Deploying Windows Server DNS on Azure virtual machines introduces the ability to control advanced name resolution behavior within the virtual network. In hybrid environments, however, name resolution rarely remains isolated to Azure alone. Organizations must ensure that Azure-hosted resources and on-premises systems can resolve each other's names consistently and securely. This requirement leads directly to the concept of hybrid name resolution.

Hybrid name resolution

Hybrid name resolution refers to an approach where names can be resolved for Azure-hosted and on-premises resources using both the Windows Server DNS and Azure DNS.

Figure 6.14 illustrates hybrid name resolution:

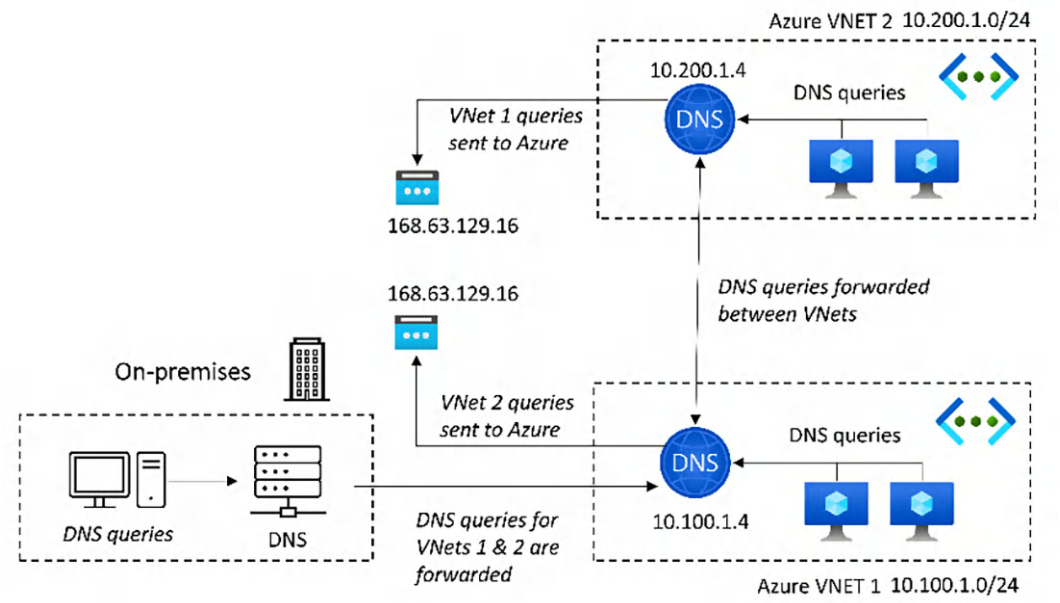


Figure 6.14: An example solution demonstrating the hybrid name resolution approach

In *Figure 6.14*, Windows Server DNS servers attached to a VNet can respond to queries for its on-premises domain, such as its own authoritative zones, and use the recursive resolvers in Azure for forwarded queries that need to resolve Azure hostnames that it will not have a record of. The address to use for the Azure DNS recursive DNS resolvers is `168.63.129.16`.

Summary

This chapter provided complete coverage of the AZ-900 Azure Fundamentals exam's following **Skills Measured** section: **Describe Azure compute and networking services** and **Describe Azure storage services**.

With this content, you built a foundational understanding of how Azure networking creates secure and segmented environments in the cloud. You explored how virtual networks define isolation boundaries, how subnets and routing control traffic flow, and how peering enables communication between networks across regions. You examined connectivity options such as VPN Gateway and ExpressRoute, understanding when encrypted internet-based tunnels are sufficient and when private, dedicated circuits are required. Finally, you learned how Azure DNS and Windows Server DNS provide name resolution within and across hybrid environments, ensuring that resources can be located reliably and securely.

In the next chapter, you will look at **Azure Identity and Access**.

Exam Readiness Drill – Chapter Review Questions

Apart from mastering key concepts, strong test-taking skills under time pressure are essential for acing your certification exam. That's why developing these abilities early in your learning journey is critical.

Exam readiness drills, using the free online practice resources provided with this book, help you progressively improve your time management and test-taking skills while reinforcing the key concepts you've learned.

HOW TO GET STARTED

- Open the link or scan the QR code at the bottom of this page
- If you have unlocked the practice resources already, log in to your registered account. If you haven't, follow the instructions in *Free Benefits with Your Book* section in this book's *Preface* and come back to this page.
- Once you log in, click the START button to start a quiz
- We recommend attempting a quiz multiple times till you're able to answer most of the questions correctly and well within the time limit.
- You can use the following practice template to help you plan your attempts:

Working On Accuracy		
Attempt	**Target**	**Time Limit**
Attempt 1	40% or more	Till the timer runs out
Attempt 2	60% or more	Till the timer runs out
Attempt 3	75% or more	Till the timer runs out
Working On Timing		
Attempt 4	75% or more	1 minute before time limit
Attempt 5	75% or more	2 minutes before time limit
Attempt 6	75% or more	3 minutes before time limit

The above drill is just an example. Design your drills based on your own goals and make the most out of the online quizzes accompanying this book.

First time accessing the online resources?

You need to unlock the online practice resources to access the link below. Check the *Free Benefits with Your Book* section in the *Preface* for detailed instructions on how to do this. You only need to unlock the resources once.

Open Quiz `https://packt.link/AZ-9003ech6` or scan the QR code

7

Azure Identity and Access

In *Chapter 6*, *Azure Network Services*, you saw the core building block resources available in Azure for networking. This chapter will outline the core building blocks of identity and access services available in Azure. You will learn about **directory services**, **authentication methods**, **external identities**, **conditional access**, and **role-based access control** (**RBAC**) in Azure.

Understanding these identity foundations will help you recognize how Azure verifies users, protects resources, and enforces access boundaries—concepts that are central to both exam success and secure cloud infrastructure operation.

This chapter aligns with the *Describe Azure Architecture and Services Skills Measured* area of the *AZ-900 Azure Fundamentals* exam. By the end of this chapter, you will be able to confidently answer questions on the following topics:

- Authentication and authorization
- Microsoft Entra ID
- Identity and access management in Azure

Identity underpins every secure Azure deployment. Mastering these concepts will help you understand how access is granted, controlled, and protected across services—an essential foundation for both the exam and any Azure environment you work in.

Authentication and authorization

Accessing resources is based on a two-stage concept of first authenticating and then authorizing, that is, identifying *who you are* and determining *what you can do*.

Authentication, also referred to as **authN**, is the process of establishing and proving the identity of a person (or service). This can be done by *validating* the provided access credentials information against stored or known identifying information.

Authorization, also referred to as **authZ**, is the process of establishing *what level of access* the authenticated person (or service) has to the resources, that is, *what they can access* and *what actions they may perform*.

Figure 7.1 visualizes the concepts of authentication and authorization.

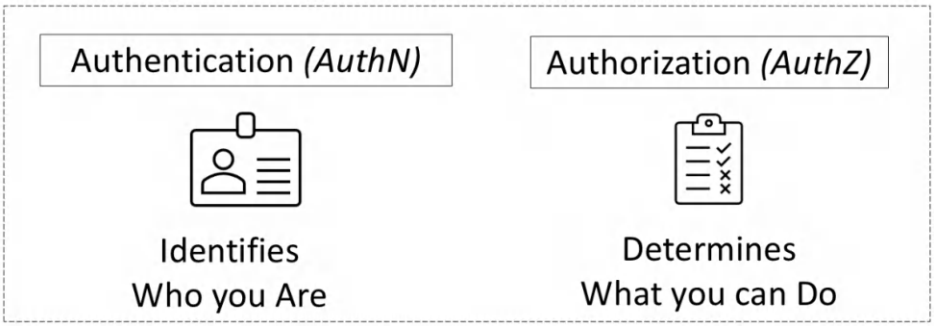


Figure 7.1: The concepts of authentication and authorization

In Azure, authentication is provided by **Microsoft Entra ID**, while authorization is enforced using Azure RBAC.

These authentication and authorization concepts form the foundation for how identity is evaluated and access is enforced in Azure, which is implemented through Microsoft Entra ID.

Microsoft Entra ID

Microsoft Entra ID (formerly Azure Active Directory) is a cloud-based, multi-tenant **identity and access management** (**IAM**) solution within Microsoft's Entra platform. It acts as a centralized **identity provider** (**IDP**) and **directory service**, storing objects with their associated attributes.

For **user identities**, Microsoft Entra ID stores attributes that are evaluated to authenticate the user and determine what level of access is permitted, including the **sign-in name**, **User Principal Name** (**UPN**), **assigned roles**, **group membership**, **devices**, **licenses**, and **authentication methods**.

The directory service is the foundation of granting identities access to resources through IAM in cloud-only and hybrid environments. It provides authentication and authorization for users, apps, machines, and devices.

Figure 7.2 visualizes Microsoft Entra ID as the centralized cloud IDP:

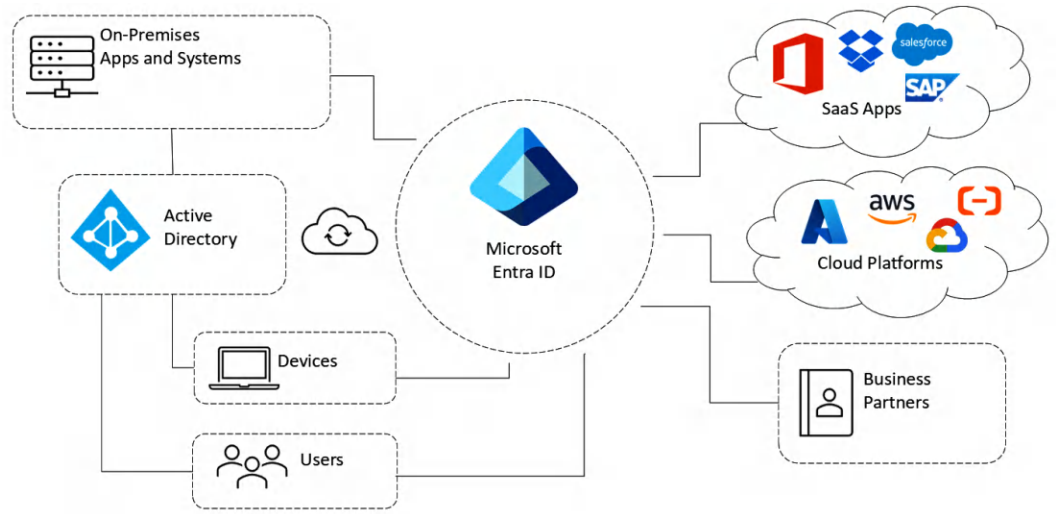


Figure 7.2: Microsoft Entra ID as a centralized cloud IDP and IAM solution

In addition to an organization's user management, Microsoft Entra ID provides the following:

- Device management
- Application management
- External identity services
- **Multi-factor authentication** (**MFA**) and conditional access
- Integration with Active Directory Federation Services
- Kerberos support
- Microsoft **Entra Domain Services** (**Entra DS**)

Unlike **Active Directory** (**AD**), Microsoft Entra ID is a multi-tenant **software-as-a-service** (**SaaS**) service. An instance of Microsoft Entra ID—a directory—is implemented for *each* Microsoft tenant that is created; each `<tenant>.onmicrosoft.com` domain represents an instance of Microsoft Entra ID. You will only have *one* directory per Microsoft tenant domain, and by creating a new `<tenant>.onmicrosoft.com` domain, you will create a new directory. You can create multiple tenant domains (and, therefore, directories) to meet your needs.

There are four key editions of Microsoft Entra ID, as follows:

- **Microsoft Entra ID Free**: This edition is included when you create a new tenant and is created with the provisioning of a Microsoft online service, such as Microsoft 365, Dynamics 365, and Azure.

- **Office 365**: This edition is included with Microsoft 365. It includes an SLA of 99.9% availability and additional functionality such as organization branding and two-way synchronization of objects between AD and Microsoft Entra ID.
- **Microsoft Entra ID P1**: This edition provides additional identity protection and identity governance functionality on top of the basic functionality.
- **Microsoft Entra ID P2**: This edition provides further identity protection and identity governance functionality on top of the Microsoft Entra ID Premium P1 functionality.

Those four are the identity editions often discussed in licensing contexts. Beyond those, Microsoft also offers the **Entra Suite**, which bundles additional services (governance, private access, internet access, etc.) on top of Entra ID, and other related products, such as Entra External ID and Entra Workload ID, which have their own licensing models—but the core identity service editions are the four outlined previously. In summary, there are four Entra ID editions (**Free**, **Basic**, **P1**, and **P2**), with the Entra Suite and other products extending the core service.

Microsoft Entra ID has objects referred to as **security principals** that provide the basis for identities. They can be one of the following types:

- **User**: This is an entity that Microsoft Entra ID can manage. The user referred to here can be a member of the organization's tenancy or a guest user who does not belong to your organization:
    - Microsoft Entra ID supports guest users through a feature called business-to-business (B2B). B2B allows access to resources in your organization's tenancy for users not part of your organization, such as business partners.
    - Microsoft Entra ID also supports **business-to-consumer** (**B2C**), allowing access to Microsoft Entra ID resources via an external IDP account such as Facebook or Google.
- **Application service principal**: This is an entity that represents the identity of a service or application in Azure.
- **Managed identity service principal**: This is an entity representing a special kind of service principal identity for a service or application to use in place of a user identity. There are system-assigned and user-assigned managed identities.
- **Device**: This is a physical entity, such as a laptop, tablet, phone, **virtual machine** (**VM**), or similar device.

Understanding Microsoft Entra ID makes it easier to see how it differs from traditional AD.

Comparing AD and Microsoft Entra ID

AD and Microsoft Entra ID are both classed as IDPs. However, they function very differently.

AD was introduced in Windows 2000 as Microsoft's directory service. AD is a role installed as part of **Windows Server**, and servers running this role are called **domain controllers (DCs)**. It allows access to multiple resources stored as computer objects within the directory service, with identities stored as user objects. AD does not simply serve a single function and has several services that can be provided; the core services are as follows:

- **AD Domain Services**
- **AD Certificate Services**
- **AD Federation Services**

Moving to modern authentication and identity services can transform your IAM strategy and approach, so these legacy services can be replaced and retired by phasing them out. However, many of these services are still required and will be around for years.

Figure 7.3 visualizes the relationship between AD and Microsoft Entra ID as IDPs:

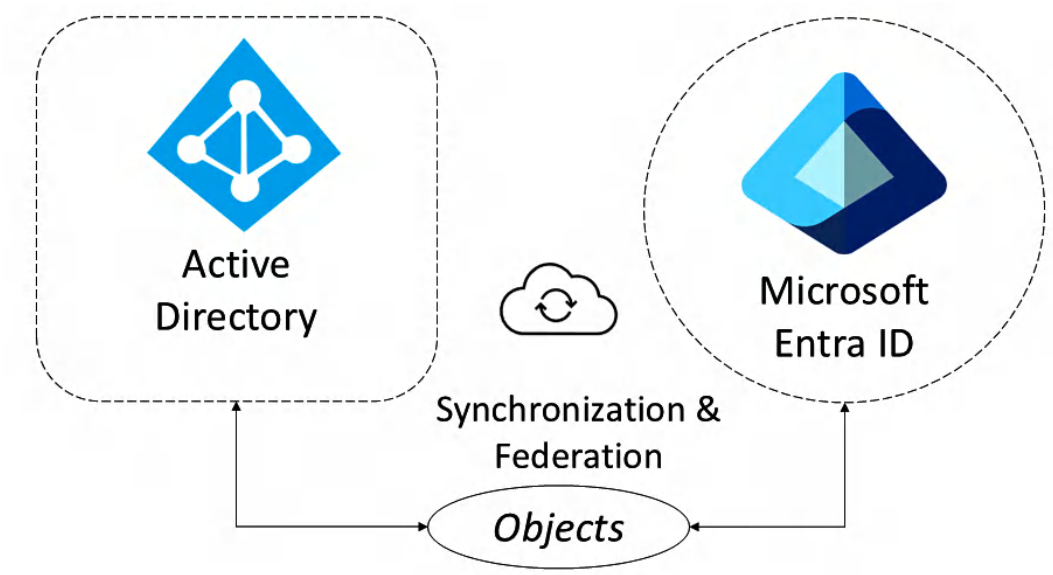


Figure 7.3: Connection between AD and Microsoft Entra ID

There is a misconception that Microsoft Entra ID is the cloud equivalent of the traditional Windows Server-based AD. However, this is not the case; Microsoft Entra ID is designed for cloud-native identity scenarios and does not provide full DC functionality.

Microsoft Entra ID can integrate with Kerberos-based authentication for specific scenarios, allowing certain legacy workloads to authenticate without requiring organizations to deploy or manage traditional DCs in Azure.

Unlike AD, there is nothing to install for Microsoft Entra ID; no DCs as VMs are required. Despite the lack of installation and VMs, Microsoft Entra ID still offers the same functionality of allowing users and computers to appear as objects in the directory. Microsoft provides Microsoft Entra ID as a fully managed IDP platform provided as SaaS.

AD can be connected to Microsoft Entra ID using **Microsoft Entra Connect**, which is a free download tool with which an organization can establish a hybrid identity. This tool synchronizes user identities, attributes, and objects between both IDPs.

A hybrid identity provides users with a common identity and an approach that allows them to access resources in Microsoft Entra ID using their AD identity. The same username and password are used to access resources in both IDP environments.

Despite hybrid identity enabling a single set of credentials across environments, Microsoft Entra ID does not provide the full set of traditional DC capabilities. This gap explains why Microsoft Entra DS exists as a managed option for workloads that still depend on domain services features such as LDAP, Kerberos, or domain join.

Microsoft Entra Domain Services

Microsoft Entra DS is a managed Azure identity service provided as **platform-as-a-service** (**PaaS**).

In simple terms, it provides domain services as a service. When you implement Microsoft Entra DS, you create a Microsoft-managed domain. This domain is a Microsoft-managed implementation of AD DS. It provides the functions of Kerberos/NTLM authentication, LDAP, domain join, and group policy.

The use case for Microsoft Entra DS is where workloads running in Azure depend on Entra DS functions and apps or services that cannot be modified or rewritten to utilize native Microsoft Entra ID and modern authentication, such as **OAuth**, **Security Assertion Markup Language** (**SAML**), or **REST**.

You may also wish to integrate your Azure resources into an existing instance of Windows Server AD or perform a lift-and-shift-style of resources that will still depend on Entra DS functionality but without wanting to run and manage your own DC VMs.

Figure 7.4 shows how to provide the same on-premises domain functions for workloads once they are moved into Azure:

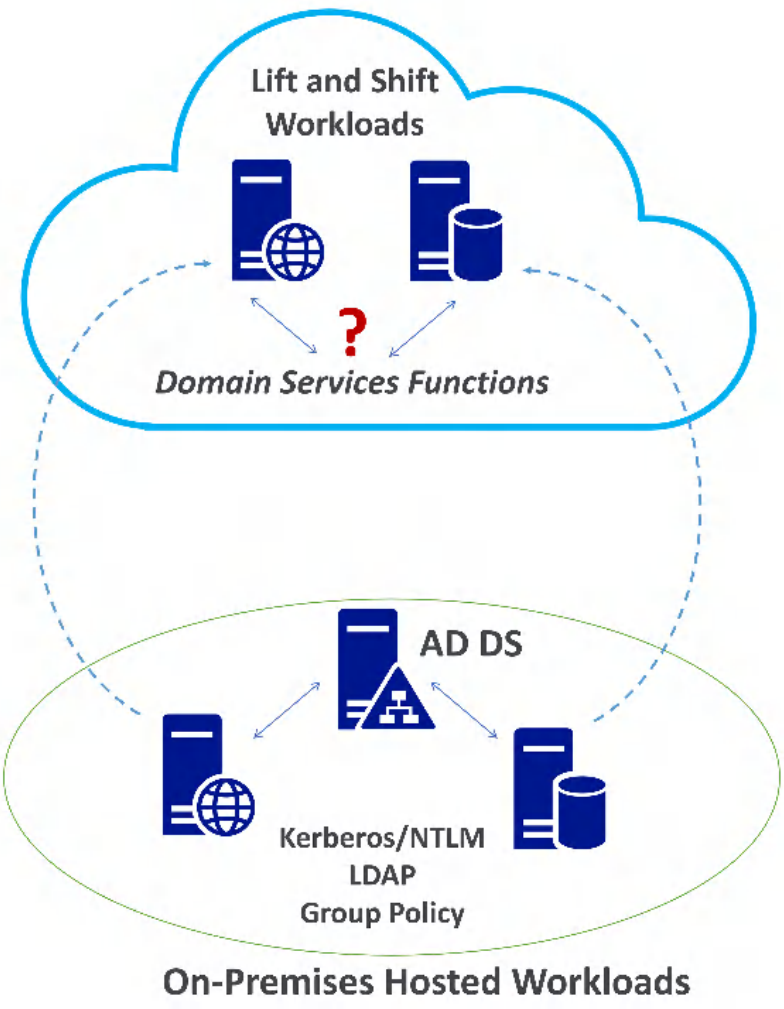


Figure 7.4: Providing AD Domain Services functions for Azure workloads

For the scenario in *Figure 7.4*, the solution could be one of the following options:

- Implement a VPN so that Azure workloads can authenticate to on-premises AD DCs
- Deploy AD DCs as VMs in Azure
- Implement Microsoft Entra DS, that is, Microsoft-managed AD DCs in Azure

The advantage of providing Microsoft-managed DCs as PaaS is that you do not need to consider the VM overhead, the physical infrastructure components, or the logical components, such as the data store, schema, trusts, partitions, availability, and protection.

The Microsoft Entra DS managed domain instance is created by Microsoft with two DCs for high availability, with additional region redundancy across two availability zones, if available in that region. Microsoft will automatically configure the distribution of DCs across the zones.

The following are some characteristics of a managed domain:

- It is Microsoft-managed, not self-managed.
- It is a standalone domain. So, extensions and joining the managed domain to an existing AD DS forest are not possible.
- Trusts are one-way—outbound forest trusts only.
- There are no domain or enterprise administrator privileges.
- It supports domain join, LDAP, and Kerberos/NTLM.
- Two built-in **organizational unit** (**OU**) containers for computers and users are provided, with an associated built-in **group policy object** (**GPO**).
- It supports a custom OU structure and group policy via a flat structure.
- It has no schema extensions.
- DCs are provided as a service and managed by Microsoft.
- Microsoft Entra DS is integrated into the Microsoft Entra ID tenant for cloud-only identity scenarios and can be combined with AD DS for hybrid identity scenarios.
- Microsoft Entra DS can be managed similarly to AD DS through the RSAT tools.
- Object sync occurs in only one direction—from Microsoft Entra ID to Microsoft Entra DS. Any changes made directly to Microsoft Entra DS will be overwritten at the next sync. These should be considered read-only DCs with no direct creation of or changes to objects with DCs.

Passwords cannot be reset in the managed domain; they must be reset in AD or Microsoft Entra ID and synchronized into the managed domain. In a cloud-only identity scenario, users, passwords, and groups are created and managed in the Microsoft Entra ID tenant.

A Microsoft Entra ID tenant acts as the authoritative directory service and IDP.

These directory objects are synchronized to the Microsoft Entra DS managed domain. There are no direct changes to objects in the managed domain. Any changes should be made in the authoritative IDP, which is Microsoft Entra ID in this scenario, and those changes must then be synchronized to the managed domain to become effective.

Figure 7.5 shows the relationship between Microsoft Entra ID and the Microsoft Entra DS managed domain in a cloud-only identity scenario:

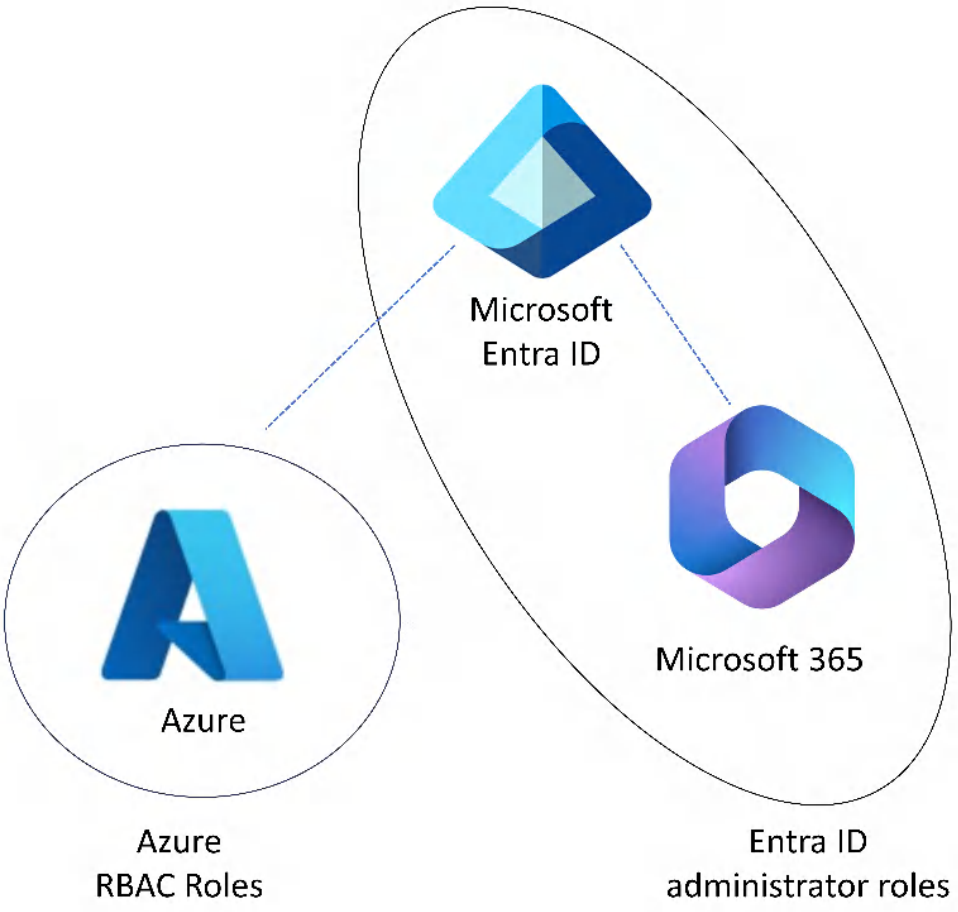


Figure 7.5: Making changes effective by synchronizing Microsoft Entra ID to the managed domain

In *Figure 7.5*, you can see that objects from the Microsoft Entra ID tenant directory are synchronized one-way to the Microsoft Entra DS managed domain directory.

Azure VMs can join the Microsoft Entra DS managed domain, allowing them to use traditional domain-based authentication such as Kerberos and LDAP.

If a user's password needs to be reset, the user's password is reset in Microsoft Entra ID. This change is then synchronized to the DCs of the managed domain. Enabling **Self-Service Password Reset** (**SSPR**) in the AD tenant is recommended.

Figure 7.6 shows the relationship between AD, Microsoft Entra ID, and the Microsoft Entra DS managed domain in a hybrid identity scenario:

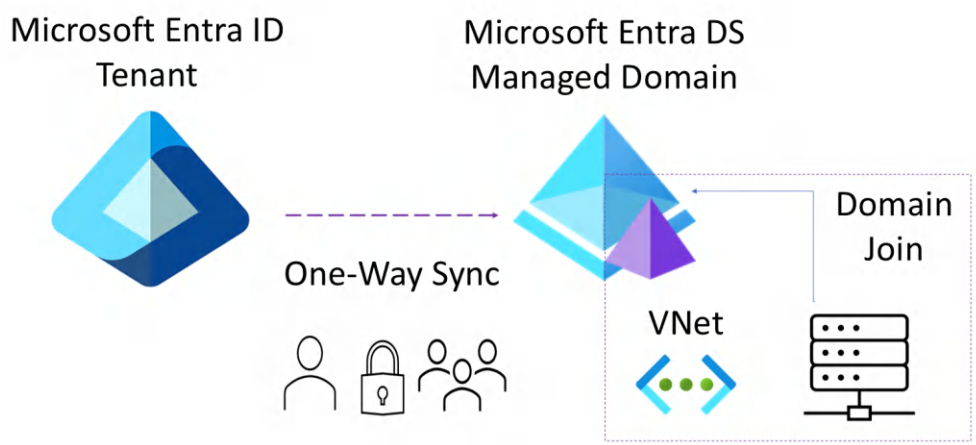


Figure7.6: Hybrid identity scenario showing the connection between AD, Microsoft Entra ID, and the Microsoft Entra DS managed domain

In a hybrid identity scenario, users, passwords, and groups are created and managed in the AD forest, which acts as the authoritative directory service and IDP. These directory objects are then synchronized *two-way* into the Microsoft Entra ID tenant using Microsoft Entra Connect, and then the objects are synchronized *one-way* to the Microsoft Entra DS managed domain.

There are no direct changes to objects to be made in the managed domain. Changes should be made in the authoritative IDP; in this case, it is the AD forest.

These changes must then be synchronized to the Microsoft Entra ID tenant using Microsoft Entra Connect and then synchronized from the Microsoft Entra ID tenant to the managed domain to become effective.

To summarize, in both cloud-only and hybrid identity scenarios, no changes are made directly in the Microsoft Entra DS managed domain.

All objects are synchronized from the Microsoft Entra ID tenant, so the only thing left to figure out is how the objects get created in Microsoft Entra ID in the first instance. This will be done by creating objects directly in the Microsoft Entra ID tenant in the case of cloud-only accounts or by allowing these objects to be created in the Microsoft Entra ID tenant from the Microsoft Entra Connect sync from the AD DCs.

Once identities are synchronized and available across environments, the next consideration is how users access applications efficiently. This is where authentication experience becomes just as important as directory architecture.

Single sign-on

With **single sign-on** (**SSO**), you only need to enter a set of credentials once, where a single authentication allows access to all resources enabled for SSO within an organization. In SSO, you are not prompted to sign in again to access each needed resource. This simplifies access to multiple applications while reducing repeated sign-in prompts for users. It also allows a governed leavers process when users no longer need access to an app.

You can configure Microsoft Entra ID as the trusted IDP for each app on which you wish to enable SSO through a centralized portal; these apps can be cloud apps, public cloud provider platforms, or on-premises apps. You can enforce secure access with identity protection through MFA, conditional access, and risk-based access policies.

While SSO simplifies access, MFA and conditional access determine when additional verification is required.

MFA and conditional access

MFA, which includes **two-factor authentication** (**2FA**), provides an additional layer of security for identifying a user. It does this by necessitating that the user submit two or more elements for authentication.

MFA is based on the following principles:

- **Knowledge**: This is something that only the user knows, such as a password or PIN—*something you know*
- **Possession**: This is something that only the user has, such as a code sent to a phone, a token, or a key—*something you have*
- **Inherent**: This is something that can only belong to the user, such as biometrics—*something you are*
- **Conditional access**: This works alongside MFA to provide more granular levels of access control

For conditional access, information is collected from the sign-in process (signals). Then, decisions are made based upon that information to determine whether access to the requested resource will be granted or denied and whether the user will require additional factors of authentication or require taking other action, such as resetting their password.

Conditional access works as an if-then model, visualized in *Figure 7.7*:

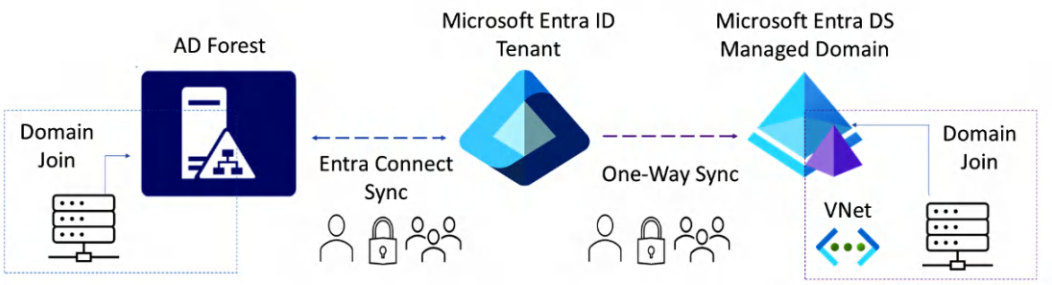


Figure 7.7: Granting access using conditional access based on the information collected from the sign-in process

In *Figure 7.7*, identities originate in the AD forest and are synchronized to the Microsoft Entra ID tenant using Microsoft Entra Connect. From there, selected objects are synchronized one-way into the Microsoft Entra DS managed domain. Azure VMs within the **virtual network (VNet)** can then join this managed domain, enabling traditional domain-based authentication without requiring you to deploy or manage your own DCs in Azure.

In practice, access decisions are not treated as one-time checks at sign-in but are increasingly evaluated continuously as conditions and risk signals change.

This could mean that if the user is trying to access the resource from a location that is unknown, a non-managed/personal device, or a device that does not have the latest security updates, then the user will be required to authenticate with a second factor of authentication or even be denied access to the resource.

Conditional access is a licensed function that requires a Microsoft Entra ID P1 license or a Microsoft Entra ID P2 license. This functionality is also included in some other Microsoft 365 license types. These controls set the stage for passwordless authentication as an alternative to traditional credentials.

Passwordless authentication

Passwordless authentication provides users with a more hassle-free user authentication process while still providing the same level of security compared to traditional password-based methods.

Passwordless authentication represents a strategic direction for reducing reliance on passwords, rather than a universal replacement for all authentication scenarios.

MFA is still used; however, users now authenticate using something they have, such as a security key or device, combined with a biometric of something they are. The passwordless authentication options are as follows:

- Windows Hello for Business
- Microsoft Authenticator
- FIDO2 security keys

Note

You can find more information on these passwordless authentication options in the following links:

- `https://learn.microsoft.com/en-us/entra/identity/authentication/concept-authentication-passkeys-fido2`
- `https://learn.microsoft.com/en-us/entra/identity/authentication/concept-authentication-windows-hello`
- `https://learn.microsoft.com/en-us/entra/identity/authentication/concept-authentication-authenticator-app`

These identity capabilities provide the foundation for implementing IAM controls across Azure.

Identity and access management in Azure

IAM in Azure is implemented through RBAC, subscription access control, Azure roles, and external identity access. Let's discuss each of them in more detail in the following subsections.

Role-based access control

RBAC is a concept that refers to authorized user access based on defined roles that have been assigned.

This model enables consistent authorization as Azure environments scale and change, without relying on static, per-resource permission assignments.

It allows you to create granular access control to Azure resources through defined roles and custom roles. You can segregate duties by granting only the access required to perform the required tasks.

It is an effective governance practice to only allow the minimum access required to complete a task; this is the basis for the principle of least privilege and should always be adopted. So, users are only given access through a role(s) that is the most appropriate for the tasks they need to carry out.

This least-privilege approach enhances governance and control of user access management as the permissions are not given directly to individuals or groups but to roles. Each role has a set of associated permissions. When a user or group is assigned that role, they receive the role's associated permissions.

It is a sound practice not to assign permissions to individual users but to add users to groups and then assign permissions to the groups for easier governance and control of group membership. Roles are specified at what level users apply and are inherited from the parent level. So, RBAC roles at the subscription level are inherited by resource groups, and subsequently, RBAC roles at the resource group level are inherited by resources. RBAC assignment is managed via the **Access Control (IAM)** blade in the portal for each resource you wish to control access over.

RBAC is based on four elements, and these are as follows:

- **Security principal**: This represents an identity that can be a user, group, service principal, or managed identity.
- **Role (definition)**: This represents a collection of permissions that the security principal will receive and the actions they can take on the resource, such as delete and write. There are several built-in roles, and you can also create custom roles.
- **Scope**: This represents the resource level that this access will apply to. The scopes are structured in a parent-child relationship from the broadest to the narrowest—management group, subscription, resource group, and resource. A recommended practice is not to set the scope too wide, but only at the lowest level needed.
- **Role assignments**: This represents the process of attaching a role definition to a security principal to provide access, creating a role assignment to grant access, and removing a role assignment to revoke access.

The following are the three core RBAC roles for controlling access to Azure resources:

- **Owner**: This role has full access to resources. In addition, it can assign access to others.
- **Contributor**: This role is the same as the Owner role; apart from that, it cannot assign access to others.
- **Reader**: This role can only view resources. It cannot create, edit, delete, or manage any resources.

If you feel that the default permissions of a built-in role may provide too much control or be too restrictive, then a more granular custom role can be created with only the permissions associated with your required role. These custom roles can then be assigned to users, groups, and service principals at the management group, subscription, and resource group scope, the same as for the built-in roles.

Up to this point, you have examined how RBAC assigns permissions to individual resources. The next step is understanding how access is controlled at the subscription level, where governance boundaries and ownership responsibilities are defined. RBAC is applied at multiple scopes, with subscription-level access control forming a key governance boundary.

Azure subscription access control

It is essential to control access to subscriptions to ensure only the required users can create resources within a subscription; this utilizes RBAC. As you learned in the previous section, RBAC allows multiple accounts with different access levels to a subscription as needed by an organization. There can be multiple owners for a subscription, although the number of owners should be limited to a maximum of three for the best operational security practices defined in Defender for Cloud. Microsoft Entra ID accounts are primarily used to assign access to Azure subscriptions through role assignments.

As discussed, there are two levels of access control—authentication and authorization:

- Authentication means proving who you say you are
- Authorization refers to what actions you are allowed to take as that identity, such as **reading data**, **modifying data**, **deleting data**, or **giving others access to data**

In managing access to an Azure subscription, authentication comes from Microsoft Entra ID, and authorization comes from Azure RBAC roles.

Figure 7.8 shows the relationship between roles:

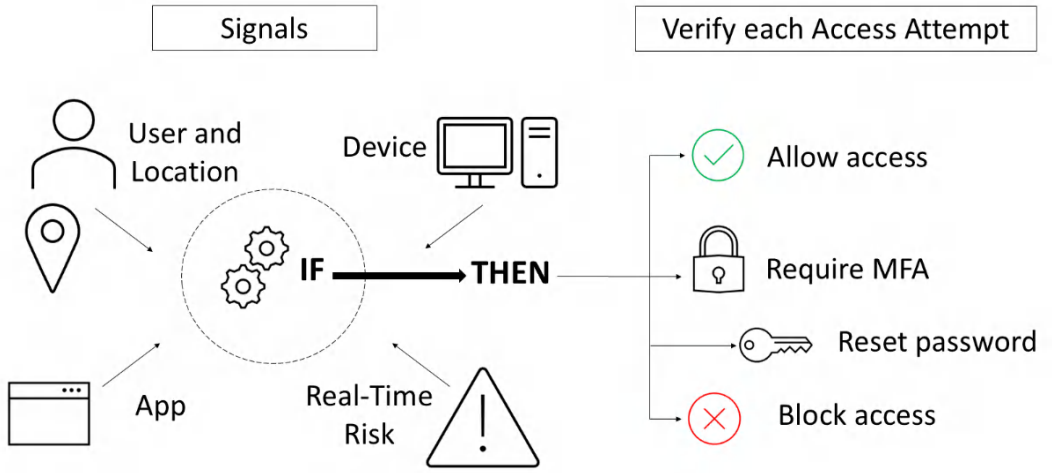


Figure 7.8: Managing access with authentication from Microsoft Entra ID and authorization from Azure RBAC

The following should be noted from *Figure 7.8*:

- By default, the Azure roles and Microsoft Entra ID are isolated and do not overlap with Microsoft Entra ID; a similar separation or isolation is maintained in Microsoft 365 roles
- Being assigned a Microsoft Entra ID role does not mean being assigned any Azure roles
- By default, there is no access to Azure resources for an account with the Global Administrator role
- Access to Azure resources must be explicitly allowed and can only be granted with an account that has the Owner role for a subscription

You now understand that subscription-level access is more than just permissions—it defines accountability. By controlling who holds Owner, Contributor, or Reader access, an organization determines who can deploy resources, assign roles, and govern the entire subscription boundary. These built-in roles are the mechanism through which that governance is enforced.

Azure roles

In this section, you will learn about the fine-grained roles from the **Azure Resource Manager** (**ARM**) deployment model, which replaced the legacy **Azure Service Manager** (**ASM**), now referred to as the classic model. In addition, custom roles can now be defined within the ARM model.

The four core ARM roles are as follows:

- **Owner**: This role has full access to all resources and can delegate access to others
- **Contributor**: This role has full access to all resources but cannot delegate access to others
- **Reader**: This role has only read access to all resources
- **User Access Administrator**: This role can only manage user access to all resources

The other 70+ roles are specific to individual Azure resources and give granular, resource context-specific allow actions, as follows:

- **Backup Operator**: This role allows the management of backup services but does not allow the creation of vaults, removing backups, or giving access to others
- **Storage Account Contributor**: This role allows the management of storage accounts and provides access to the account key
- **VM Administrator Login**: This role allows viewing VMs in the portal and logging in as the administrator, but does not allow VM creation or disk and network management

These roles can be further customized to any individual permission required, combining a few of the 1,000 possible roles. The roles and their assignments are managed via the **Access Control (IAM)** blade for the subscription within the Azure portal.

Access control also extends beyond internal users through external identity access.

External identity access

As well as internal tenant users, B2B and guest users can be invited to access a Microsoft Entra ID tenant through the external identity access capability of Microsoft Entra B2B. A Microsoft Entra ID account from another tenancy can be given access to the subscription within a different tenancy.

Azure Lighthouse can also be used for managed security service providers; although this is not within the scope of the exam objectives, it is mentioned here for completeness. Azure Lighthouse provides delegated admin access at the subscription or resource group level to an organization's tenant from an MSP's managing tenant.

Together, these capabilities show how Azure enforces identity-driven access control across users, applications, and external collaborators.

Summary

This chapter examined identity and access as a core architectural building block in Azure, focusing on how access to cloud resources is established, evaluated, and enforced. By separating authentication from authorization, the chapter established a clear foundation for understanding how Azure determines who a user or service is and what actions they are permitted to perform.

You explored Microsoft Entra ID as Azure's cloud-based IDP and directory service, and how it differs from traditional AD while still supporting hybrid identity scenarios. The chapter showed how identity attributes, authentication methods, and access policies work together to enable secure access for users, devices, applications, and external collaborators.

The chapter also introduced key access control mechanisms, including SSO, MFA, conditional access, and RBAC. Together, these capabilities demonstrate how Azure applies identity-driven decisions to protect resources while supporting scalability, governance, and least-privilege access.

These identity and access fundamentals form the foundation for understanding how Azure enforces governance boundaries and applies security controls at scale—areas you will explore in the next chapter.

Exam Readiness Drill – Chapter Review Questions

Apart from mastering key concepts, strong test-taking skills under time pressure are essential for acing your certification exam. That's why developing these abilities early in your learning journey is critical.

Exam readiness drills, using the free online practice resources provided with this book, help you progressively improve your time management and test-taking skills while reinforcing the key concepts you've learned.

HOW TO GET STARTED

- Open the link or scan the QR code at the bottom of this page
- If you have unlocked the practice resources already, log in to your registered account. If you haven't, follow the instructions in *Free Benefits with Your Book* section in this book's *Preface* and come back to this page.
- Once you log in, click the START button to start a quiz

- We recommend attempting a quiz multiple times till you're able to answer most of the questions correctly and well within the time limit.
- You can use the following practice template to help you plan your attempts:

Working On Accuracy		
Attempt	**Target**	**Time Limit**
Attempt 1	40% or more	Till the timer runs out
Attempt 2	60% or more	Till the timer runs out
Attempt 3	75% or more	Till the timer runs out
Working On Timing		
Attempt 4	75% or more	1 minute before time limit
Attempt 5	75% or more	2 minutes before time limit
Attempt 6	75% or more	3 minutes before time limit

The above drill is just an example. Design your drills based on your own goals and make the most out of the online quizzes accompanying this book.

First time accessing the online resources?

You need to unlock the online practice resources to access the link below. Check the *Free Benefits with Your Book* section in the *Preface* for detailed instructions on how to do this. You only need to unlock the resources once.

Open Quiz `https://packt.link/AZ-9003ech7` or scan the QR code

Join the CloudPro Newsletter with 44000+ Subscribers

Want to know what's happening in cloud computing, DevOps, IT administration, networking, and more? Scan the QR code to subscribe to **CloudPro**, our weekly newsletter for 44,000+ tech professionals who want to stay informed and ahead of the curve.

`https://packt.link/cloudpro`

8

Azure Security

In *Chapter 7*, *Azure Identity and Access*, you learned how to describe identities, identity provider services, authentication methods, and access control. Building on the identity and access foundations from the previous chapter, this chapter broadens the lens to examine how Azure approaches security as a system—across people, processes, and technology. It introduces the core security concepts that underpin Azure protection, including security posture, **Zero Trust**, **defense in-depth** (**DiD**), and Microsoft Defender for Cloud.

This chapter equips you to evaluate security in Azure beyond individual features. By understanding security posture, Zero Trust principles, and layered protection strategies, you will be able to interpret security recommendations, recognize architectural weaknesses, and explain how Azure security services work together to reduce risk. This foundation enables you to make informed cloud security design decisions with confidence rather than viewing security tools as isolated configurations.

The content in this chapter aligns with the *Describe Azure identity, Access, and Security Skills Measured* area of the AZ-900 Azure Fundamentals exam. By the end of this chapter, you will be able to confidently answer questions on the following topics:

- Threat landscape
- Security posture
- Zero Trust
- Defense in-depth
- Microsoft Defender for Cloud

In addition, this chapter encourages you to think about security as an architectural discipline rather than a checklist of features, helping you connect individual services to an overall risk reduction strategy.

Before exploring specific security models and controls, it is important to understand the types of threats modern cloud environments are designed to defend against.

Threat landscape

The profile of attackers changes continuously. Threats may come from foreign nations, competitors, criminal hackers, hacktivists, script-kiddies, or opportunists. Internal threats are often the hardest to detect and guard against, and attackers are not always strangers; they could be insiders or disgruntled ex-employees, which could have the greatest impact due to their insights into your systems and data.

To avoid becoming an easy target for opportunists and targeted attacks, controls should raise the attacker's cost and effort significantly. This forces attackers to divert their resources and activities to an easier attack target with a higher return on their attack investment.

> **Note**
>
> To learn about the reality of insider threats, search for **Sly Dog Gang** in your search engine to learn about the threat of a real espionage attack on one of the highest-profile manufacturers of electric vehicles.

The aim of an attack may be specific to an organization, which might be to steal data, deface a website, alter the integrity of an app or a service, or extort money through ransom. Usually, there are two motivations of attackers: **money** or a **mission**. *Figure 8.1* visualizes these motivations:

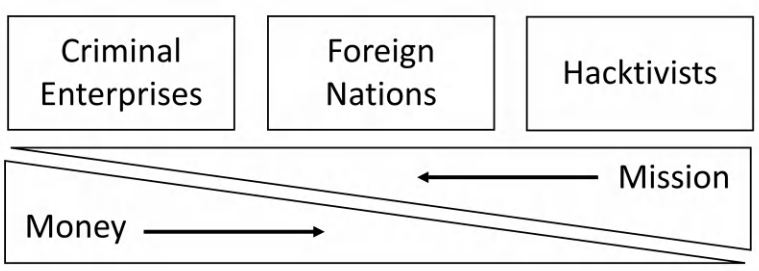


Figure 8.1: Attack motivations

Figure 8.1 illustrates how different threat actors sit along a spectrum between financial gain and ideological or strategic mission. While criminal enterprises typically prioritize profit, nation-state actors and hacktivists may be driven more strongly by political, strategic, or ethical objectives. The arrows emphasize that motivation is not binary; actors can shift along this spectrum depending on context.

The motivation is more apparent in **money-driven attacks** involving a calculated assessment by the attacker of their **Return on Investment** (**ROI**) before deciding to give up on the current target and move on to another one.

However, a **mission-driven attack** is often more of a **moral standard** and a matter of ethics, principles, politics, and control than money. Thus, the attacks may be more sustained, and the attackers may be determined to succeed at any cost because the reward may not have a price that can be attached.

These motivations translate into a small number of recurring attack patterns that organizations consistently need to defend against.

Common threats

Some of the most common threats to protect against are as follows:

- **Ransomware**: This is malware that will encrypt files and folders in an attempt to extort money.
- **Data breach**: This includes **phishing**, **spear phishing**, **Structured Query Language** (**SQL**) injection, stealing passwords/bank details/other sensitive information, luring somebody to click a link, and opening a file.
- **Dictionary attack**: This is an **identity theft attack** (put simply, a **password-guessing attack**), also known as a **brute-force attack**. In this attack, *known passwords* are used against an account to steal an identity.
- **Disruptive attack**: This is a **network and workload attack**. A **distributed-denial-of-service** (**DDoS**) attack is disruptive as it attempts to make a network or workload (such as an application or service) unavailable by flooding it with requests and attempting to exhaust its resources.

While individual threats vary in technique, successful attacks tend to follow predictable stages rather than occurring as isolated events.

As these attack stages have become well understood, adversaries have increasingly sought ways to accelerate, automate, and scale them—an evolution that **artificial intelligence** (**AI**) now significantly amplifies.

Emerging threats: AI-amplified attacks

AI has reduced the cost and skill barrier required to launch cyberattacks. Techniques that once required specialist knowledge—such as phishing, social engineering, and password guessing —can now be automated and scaled using AI models.

In practice, AI does not introduce entirely new attack categories. Instead, it accelerates existing threats, making them faster, more convincing, and harder to detect. This reinforces the need

for identity-centric security controls, behavioral detection, and layered defenses rather than reliance on static perimeter protections.

Consider a phishing campaign a decade ago. Crafting convincing messages required language fluency, manual targeting, and time-intensive reconnaissance. Today, AI models can generate tailored phishing emails at scale, mimic writing styles, and adapt messaging based on prior responses. The attack pattern remains the same, but the speed, personalization, and volume increase dramatically. Defending against this shift requires stronger identity verification, continuous monitoring of user behavior, and controls that assume compromise rather than relying solely on email filtering.

These shifts underscore a broader reality: as threats evolve in terms of speed and sophistication, organizations must continuously reassess their security posture rather than rely on static defenses.

Kill chains

Attackers plan and structure their attacks to stay undetected on the network and in the user's systems without the victim being alerted. Attacks follow a sequence or chain of events, known as an **attack chain** or **kill chain**. A common chain is represented in *Figure 8.2*:

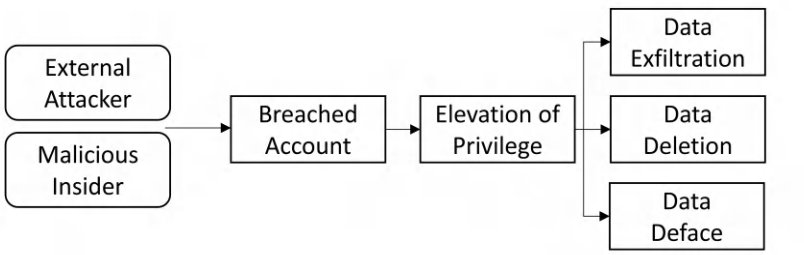


Figure 8.2: An attack chain showing the sequence of an attack

When a user account is compromised, it can access the network and then work to elevate privileges to an admin account that can move laterally within the network. This access can then be used to access systems and execute activities, such as stealing, deleting, corrupting, and encrypting data. It can also manipulate a service, such as by adding/editing **Domain Name Service** (**DNS**) records and **Active Directory** (**AD**) objects, installing software, enabling service credits, purchasing goods/services, as well as using the compromised access for further abuse.

Disrupting this chain of events requires security models that assume compromise and limit attacker movement at every stage. This can be done through a Zero Trust and **Defense-in-Depth** (**DiD**) approach to prevent and disrupt this chain of events. You need to put multiple

obstacles in the attacker's way and increase their attack costs so that they will move on to launching an easier attack elsewhere that offers less resistance. This will be discussed later in this chapter in more detail.

Security must be in place before a single line of code is written, a system is created, or data is stored. However, this can often be seen as the *anti-pattern* of operations, availability, and productivity. Security controls are sometimes perceived as obstacles to productivity when they are introduced without alignment to operational goals. A culture akin to **Development-Operations** (**DevOps**) that fosters trust between all teams and security teams must exist, and leaders must introduce the concept and culture of **Security-Development-Operations** (**SecDevOps**) to an organization. The bottom line is that security is not just somebody else's problem but **everybody's responsibility**.

Understanding how attacks progress highlights the need to prioritize defensive effort where it most effectively interrupts attacker paths. This leads us to the following question.

What can be done to prevent threats?

The approach that should be taken is to adopt a **threat priority model**. To determine the risk priority, you must determine the impact and probability. A risk solution can be derived from the risk classification given by the risk analysis.

The visualization of the threat priority model is represented in *Figure 8.3*:

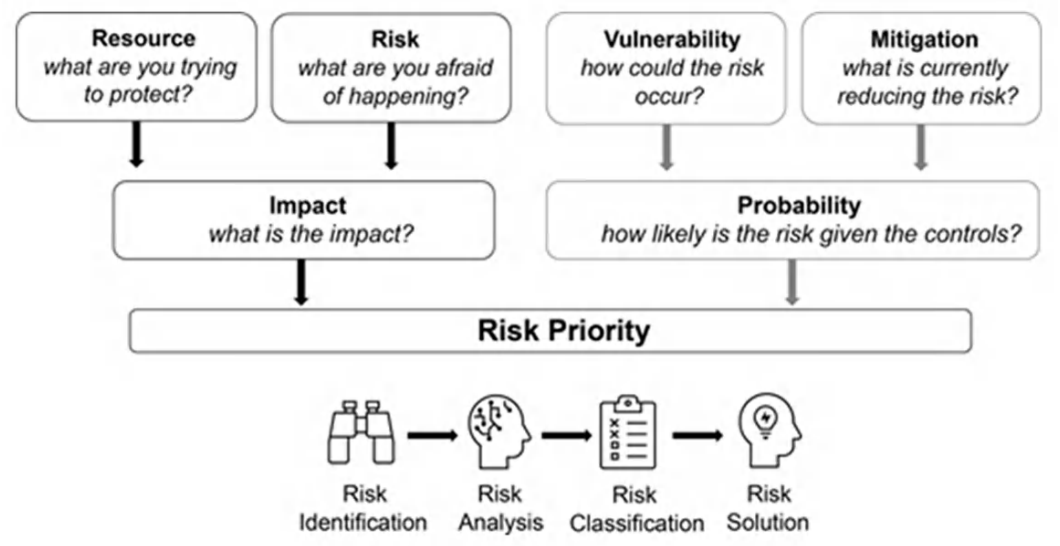


Figure 8.3: Threat priority model showing the adopted risk solution based on risk priority

The threat priority model can aid in identifying your risk priorities, as well as where security investments should be made to reduce your **security operations** (**SecOps**) costs and increase your attacker's kill-chain costs.

For example, consider an organization that stores sensitive customer data in a cloud-based application. The resource to protect is the customer database. A key risk might be unauthorized access through compromised credentials. The vulnerability could stem from weak password policies or a lack of multifactor authentication, while existing mitigation measures may include basic access controls. By assessing the potential impact of data exposure and the probability of credential compromise, the organization can assign a risk priority. If the impact is high and the likelihood is significant, strengthening identity controls—such as enforcing multifactor authentication and monitoring anomalous sign-ins—becomes a prioritized security solution.

Risk prioritization informs decision-making, but it must be grounded in a clear understanding of an organization's overall security readiness.

Security posture

A **security posture** is an organization's *threat-protection* and *response capabilities*. This ensures that the organization has the ability for systems, data, and identities to be recoverable and operational should an attack be successful.

Any security approach must start from an inward look at the current security position. Secure Score can be thought of as a *credit-rating* score you receive to see how likely you are to be accepted for a finance agreement. In other words, Secure Score will indicate your security posture. However, in security terms, it looks at where you are on the attack *vulnerability scale* of 1 to 10.

It is critical to know that you cannot prevent or eliminate threats and attacks. An attacker only must be successful once, while you must protect everything all the time. A security posture's goal should be to reduce exposure to threats, shrinking attack surface areas and vectors while building resilience to attacks, as attacks cannot be eliminated.

A security strategy and security posture should use the guiding principles of **Confidentiality, Integrity, and Availability** (**CIA**). This is also referred to as the **CIA triangle**, which is a common industry model used by security professionals. There is no perfect threat prevention or security solution; there will always be a trade-off, and the CIA model is a way to consider that. The CIA triangle model can be visualized as represented in *Figure 8.4:*

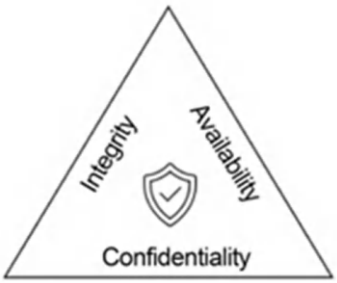


Figure 8.4: A security posture CIA triangle

The following are the guiding principles in more detail:

- **Confidentiality**: This is a requirement that sensitive data be kept protected and can only be accessed by those who should have access through the principle of least privilege. This is the confidence that the data cannot be accessed, read, or interpreted by anybody other than those intended to read and access this data; this can be achieved by encrypting the data. The encryption keys also need to be made confidential and available to those who need access to the data.
- **Integrity**: This means that the data transferred is the same as the data received; the bytes sent are the same bytes received. This is the confidence that the data has not been altered from its original form or tampered with. This can be achieved by hashing the data. Malware can threaten the integrity of systems and data.
- **Availability**: This means that data and systems are available to those who need them, including access to encryption keys, but in a secure and governed manner. This is a trade-off between the triangle's three sides and a balance between being locked down for security, but accessible for operational needs and productivity. A DDoS attack will threaten the availability of systems, data, and encryption keys.

We just established how organizations assess their security posture using the CIA principles and Secure Score. Once an organization understands its current security posture, the next step is to define a strategy that governs how trust and access decisions are made.

Zero Trust

You need to think beyond traditional network perimeter-based security and adopt a holistic approach to security. An important concept to consider in a security strategy is **Zero Trust**, which uses the approach of never trust and always verify. Zero Trust is not a product, service, or solution, but a broader security strategy and framework to adopt. It works on the notion of ensuring compliance and securing access to the resource, rather than trusting the location or network on which the resource resides. You must not assume trust because of the resource's network or location.

The Zero Trust framework is built upon the following foundational principles:

- **Assume breach**: This principle assumes that systems have already been compromised and infiltrated by an attacker; they are already inside your systems. The impact can be mitigated by requiring the use of DiD measures with segmented access to prevent lateral movement.
- **Verify explicitly**: Regardless of the scenario, all data points should be used to ensure authentication and authorization are always performed and never skipped, due to implicit data signals. This requires the use of multiple factors or authentication, such as combining biometric or authenticator codes with usernames and passwords with Conditional Access controls.
- **Use least privilege**: Access is limited to those only required to perform specific tasks or activities. This requires the use of the **Just in Time** (**JIT**) and **Just Enough Access** (**JEA**) principles.

The illustration *Figure 8.5* aims to visualize principles of the Zero Trust framework.

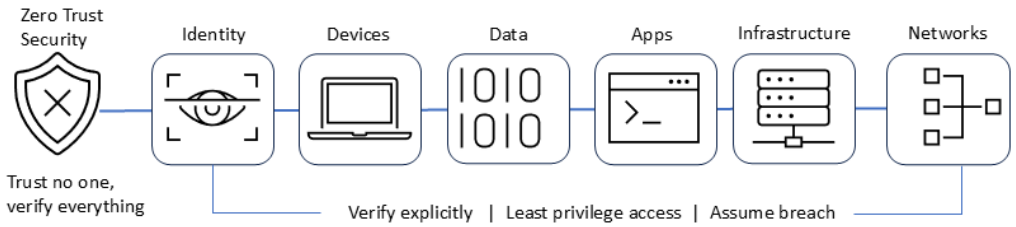


Figure 8.5: Zero Trust Framework

The following are the Zero Trust framework's six foundational elements:

- **Identities**: This represents users, services, and devices; each represents an element to be compromised
- **Devices**: This represents an attack surface and threat vector for data flows
- **Applications**: This represents the consumer of the data flows
- **Data**: This represents the data stored that is to be protected
- **Infrastructure**: This represents an attack surface and threat vector, whether locally on-premises or remotely hosted by a cloud provider
- **Network**: This represents an attack surface and threat vector and should be segmented

The Zero Trust framework encompasses identities as one of its six foundational elements. In this new world of hybrid work, where organizations' traditional firewalls and security service-controlled network perimeters have vanished, you must now consider identity as the new perimeter and control pane.

Why identity became the new perimeter

Traditional network perimeters assumed that users, devices, and workloads were physically located inside a trusted boundary. Cloud services, SaaS applications, mobile devices, and remote work have dissolved that assumption. As a result, identity signals—who the user is, what device they are using, and under what conditions they are accessing a resource—now provide a more reliable trust boundary than network location alone. Zero Trust strategies, therefore, focus on validating identity and context continuously, rather than trusting where a request originates.

Zero Trust defines how trust decisions are made; DiD addresses how protections are layered across the environment. While Zero Trust focuses on verifying identity and context at every

access point, DiD expands that principle by ensuring protection is layered throughout the entire architecture.

Defense in-depth

DiD is a security strategy that places multiple layers of different forms of defense between attackers and the resources that need to be protected. Adopting a DiD strategy allows an organization to adopt a strong security posture and helps ensure that all systems, data, and users are better protected from threats and compromise. A DiD strategy means no single layer of protection or security service is solely responsible for protecting resources.

Designing for control failure

DiD acknowledges that individual security controls will eventually fail. Configuration errors, zero-day vulnerabilities, and human mistakes are unavoidable in complex environments. The purpose of layering controls is therefore not to guarantee prevention, but to ensure controlled failure—so that when one layer is bypassed, subsequent layers restrict lateral movement, limit blast radius, and preserve recovery options.

Multiple layers of protection in a DiD strategy slow down attack paths by applying different defenses at each layer. Attackers may successfully breach one defensive layer but will be halted by subsequent protection layers, preventing the protected resource from being exposed. *Figure 8.6* depicts DiD as a concept that can be considered as the medieval castle approach to protecting resources:

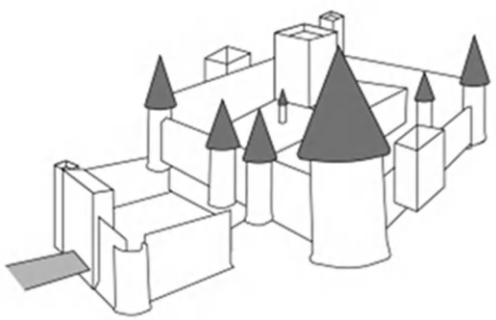


Figure 8.6: The medieval castle defense approach

The medieval castle approach should be part of your strategy for building your resources in Azure. In the medieval castle analogy, each layer from the center to the outside and back to the center provides its own independent protection service, tailored to protect the characteristics of that layer best.

The DiD approach layers, from the outside layer to the inside layer, can be represented as follows:

- Application
- Compute
- Network
- Perimeter
- Identity and access
- Physical security
- Data

Figure 8.7 helps to visualize the layered security approach that defines a DiD strategy for a protected resource:

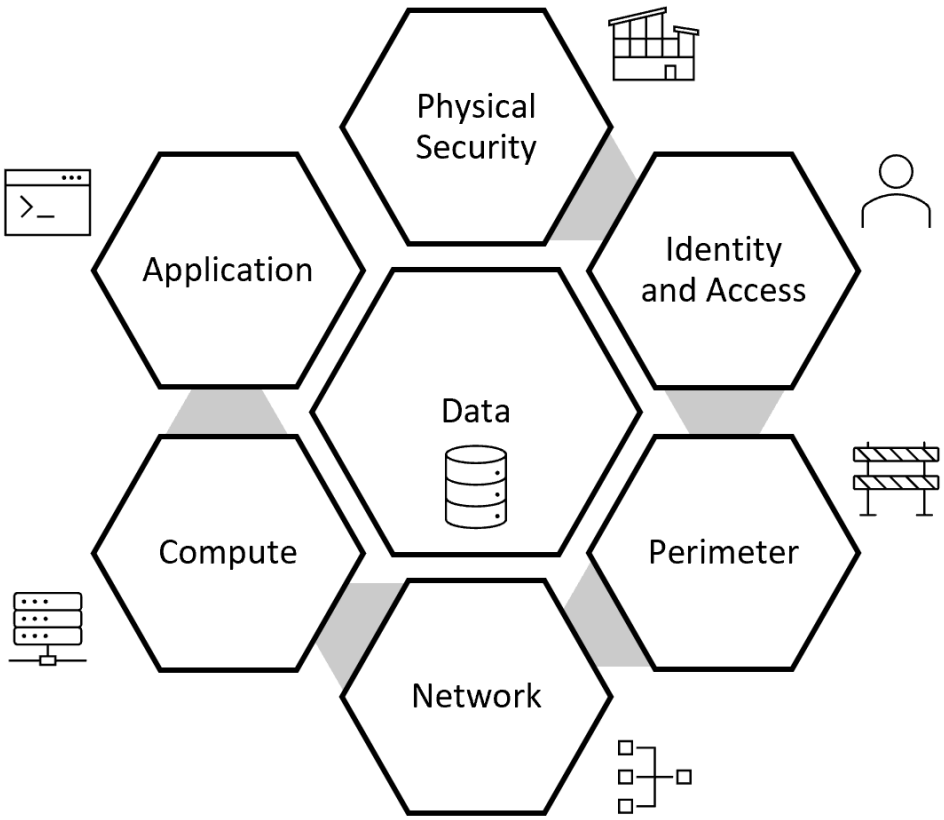


Figure 8.7: The DiD approach layers

As there is no one-size-fits-all security service that can protect all the layers, you must have security services at each layer that work in conjunction and complement the outside and inside layers. There must be a unified view so that telemetry and threat intelligence can be passed between each layer and enhance protection at each layer.

Microsoft uses AI, threat intelligence, and analytics to enhance these capabilities. These layered principles are operationalized in Azure through **Microsoft Defender for Cloud**.

Microsoft Defender for Cloud

Microsoft Defender for Cloud **{ XE "Microsoft Defender for Cloud" }**is a cloud-native security solution that is used by an organization to improve its security posture and workload protection. *Figure 8.8* shows Microsoft Defender for Cloud in the Azure portal.

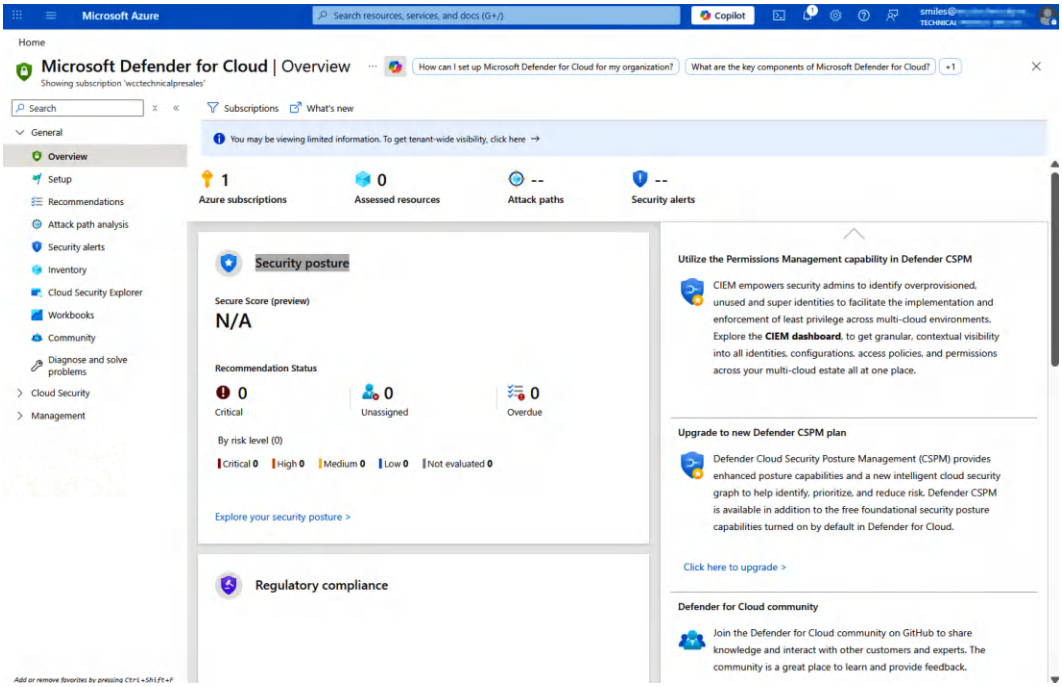


Figure 8.8: Defender for Cloud

Defender for Cloud is automatically enabled at no cost for all subscriptions. However, only basic **Cloud Security Posture Management** (**CSPM**) functionality is provided to ensure a security posture management capability is available as a default.

The Secure Score function of the CSPM capability in Defender for Cloud measures an organization's security posture. The security posture and Secure Score are determined by the

number of security recommendations and controls met. This Secure Score function provides information on the current and maximum scores available and the potential increase by implementing the recommendations provided. Microsoft Defender for Cloud also compares an environment with industry standards and regulatory compliance that may need to be conformed to, such as the **International Organization for Standardization** (**ISO**) **27001** or the **Payment Card Industry** (**PCI**). These reports can also be exported.

The continuous assessments, security recommendations, and Secure Score are part of the included capabilities enabled by default within Defender for Cloud.

Security posture management versus workload protection

Within Microsoft Defender for Cloud, it is important to distinguish between two complementary security approaches. Security posture management focuses on reducing risk before an attack occurs by identifying misconfigurations and missing security controls. Workload protection focuses on detecting and responding to threats after workloads are running. Both approaches are required: posture management lowers the likelihood of compromise, while workload protection reduces impact when preventive controls fail.

However, to take full advantage of the **Cloud Workload Protection** (**CWP**) capabilities provided by Defender for Cloud, you need to enable the enhanced features. Once enabled, these extended security posture, detection, and response features are available to improve your security posture and workload protection. These features can be enabled for subscriptions via the paid-for Defender plans.

Defender for Cloud can also be integrated with Microsoft Sentinel as the **security information and event management** (**SIEM**) solution, providing seamless integration and sharing threat intelligence and signal data.

The following are some capabilities and terminology used with Defender for Cloud:

- **SecOps**: The SecOps function deals with managing the inclusion of the day-to-day security monitoring needs of an organization in IT operations.
- **Security posture**: This is the status of an organization's cybersecurity measures and its ability to detect and react to security threats.
- **Secure Score**: This is a percentage-based score based on Microsoft's best practice security recommendations. It measures your security posture; the higher your score, the greater your security posture.
- **CSPM**: CSPM is the means to measure an organization's security posture; it is a proactive security service that provides a Secure Score to measure your security

protection levels. It provides actionable recommendations and insights for the remediation of identified threat vectors and vulnerabilities.

- **CWP**: These are reactive security measure tools that can be implemented to protect workloads identified as potentially at risk from the CSPM insights.

Both Defender for Cloud and Sentinel go beyond the scope of Azure. They also allow agent-based hybrid **virtual machine** (**VM**) scenarios from on-premises, as well as integrating AWS and GCP cloud security information to get a single source of truth.

Why security fundamentals still matter

While security technologies continue to evolve—driven by automation, cloud scale, and AI—the foundational principles of Zero Trust, DiD, and security posture assessment remain constant. New attack techniques change how attacks are executed, but not why they succeed.

Organizations that understand these fundamentals can adapt to new threats without redesigning their security strategy from scratch.

We have now explored how adopting Defender for Cloud improves an organization's security posture and workload protection capabilities, and with that, we have reached the end of this chapter.

Summary

This chapter aligns with the AZ-900 Azure Fundamentals exam *Skills Measured* area of *Describe Azure identity, Access, and Security*, by bringing together the core concepts that shape how Azure environments are protected.

It linked the modern threat landscape to foundational security principles, showing how security posture, Zero Trust, and DiD influence the way risks are identified and controlled. Microsoft Defender for Cloud was presented as the service that brings these principles into practice, providing visibility into the security posture and protection for running workloads. Taken together, these concepts highlight that effective cloud security relies on coordinated decisions across people, processes, and technology, rather than any single control or service.

The next chapter shifts focus to **Azure Cost Management**, where you will look at how cloud costs are influenced, measured, and controlled using Azure's pricing and cost management tools.

Exam Readiness Drill – Chapter Review Questions

Apart from mastering key concepts, strong test-taking skills under time pressure are essential for acing your certification exam. That's why developing these abilities early in your learning journey is critical.

Exam readiness drills, using the free online practice resources provided with this book, help you progressively improve your time management and test-taking skills while reinforcing the key concepts you've learned.

HOW TO GET STARTED

- Open the link or scan the QR code at the bottom of this page
- If you have unlocked the practice resources already, log in to your registered account. If you haven't, follow the instructions in *Free Benefits with Your Book* section in this book's *Preface* and come back to this page.
- Once you log in, click the START button to start a quiz
- We recommend attempting a quiz multiple times till you're able to answer most of the questions correctly and well within the time limit.
- You can use the following practice template to help you plan your attempts:

Working On Accuracy		
Attempt	**Target**	**Time Limit**
Attempt 1	40% or more	Till the timer runs out
Attempt 2	60% or more	Till the timer runs out
Attempt 3	75% or more	Till the timer runs out
Working On Timing		
Attempt 4	75% or more	1 minute before time limit
Attempt 5	75% or more	2 minutes before time limit
Attempt 6	75% or more	3 minutes before time limit

The above drill is just an example. Design your drills based on your own goals and make the most out of the online quizzes accompanying this book.

First time accessing the online resources?

You need to unlock the online practice resources to access the link below. Check the *Free Benefits with Your Book* section in the *Preface* for detailed instructions on how to do this. You only need to unlock the resources once.

Open Quiz `https://packt.link/AZ-9003ech8` or scan the QR code

Part 3

Azure Management and Governance

This section provides coverage of the knowledge and skills required for the *Skills Measured* of the *Describe Azure Management and Governance* section of the exam. We also cover knowledge and skills that go beyond the exam content, so you are prepared for a real-world, day-to-day Azure-focused role.

This part of the book includes the following chapters:

- *Chapter 9, Azure Cost Management*
- *Chapter 10, Azure Governance and Compliance*
- *Chapter 11, Azure Resource Deployment and Management*
- *Chapter 12, Azure Monitoring Tools*

9

Azure Cost Management

In *Chapter 8*, *Azure Security*, you explored how Azure approaches protection through principles such as Zero Trust, defense-in-depth, and security posture management. This chapter explains how Azure costs are incurred, monitored, and controlled in a consumption-based cloud model, and why understanding cost drivers and management tools is essential as cloud usage becomes more dynamic.

This chapter primarily focuses on the *Describe Azure Management and Governance* module from the *Skills Measured* section of the *AZ-900 Azure Fundamentals* exam.

> **Note**
>
> A full mapping of *AZ-900 Skills Measured* is available in the *Appendix*, *Assessing AZ-900 Exam Skills*.

By the end of this chapter, you will be able to confidently answer questions on the following topics:

- Cost factors in Azure
- Azure Cost Management
- Azure Pricing Calculator

Beyond preparing you for exam questions, this chapter strengthens your ability to evaluate cost decisions in practical scenarios—helping you understand how pricing structures, usage patterns, and governance choices directly affect organizational budgets and long-term cloud sustainability.

As you deepen your understanding of Azure, this chapter encourages you to think beyond exam objectives and consider how cost decisions shape day-to-day cloud operations.

Understanding how Azure charges for resources starts with recognizing the factors that influence cost behavior over time.

Cost factors in Azure

Each Azure consumption-based service (usage-based) has one or more **usage meters** that define the **price rate** and **unit of cost**; the **service type** determines the different units of cost. Since Azure pricing is based on usage, cost behavior is influenced by not only *which* services are used, but also *how* those services are consumed over time.

Two deployments using the same Azure service can generate very different costs depending on running duration, scale, configuration choices, and data movement patterns. This variability is one of the key differences between cloud and traditional on-premises computing environments.

Instead of sizing infrastructure for **peak capacity** and spending on resources as **capital expenditure** (**CapEx**), Azure allows resources to scale dynamically and charges only for measured usage, allowing spending on resources as **operational expenditure** (**OpEx**). While this model provides flexibility and efficiency, it also requires greater awareness of how architectural and operational decisions translate into ongoing running costs.

As workloads evolve, usage patterns may change significantly from month to month, new features may be enabled, data volumes may grow, or demand may fluctuate based on business activity; understanding the factors that influence Azure costs, therefore, enables more accurate forecasting, better cost control, and more informed design decisions.

In practice, these cost factors are rarely independent; choices made across design and operation often interact in ways that amplify or offset one another over time; understanding these interactions helps organizations anticipate how costs may change as workloads scale or usage patterns evolve.

Azure costs are often determined long before a workload is deployed; design-time architectural choices—such as choosing between single-region and multi-region deployments, selecting managed services instead of self-managed infrastructure, or adopting event-driven architectures—have a direct and lasting impact on cost behavior.

For example, deploying workloads across multiple regions can improve resiliency but may introduce additional costs related to data replication and outbound network traffic. Similarly, selecting higher availability or performance tiers may reduce operational risk but increase baseline service spend. Understanding these trade-offs early helps ensure that cost is considered alongside availability, security, and performance requirements rather than treated as an afterthought.

In Azure, cost-efficient architectures typically balance scalability with predictability; designs that scale dynamically with demand can reduce waste during low-usage periods, but they also require monitoring to prevent unexpected cost spikes during periods of high activity.

These usage patterns are reflected in Azure's monthly billing, which is calculated per subscription and based on resource consumption recorded by usage meters. As a result, monthly invoices may vary depending on which resources were used, how long they ran, where they were deployed, and how much data was processed or transferred.

The following are the primary factors that can affect costs:

- Purchasing model
- Resource type and selected service tier (SKU)
- Location (region)
- Usage period
- Network traffic
- Unused billable resources

These aspects are visualized in *Figure 9.1*:

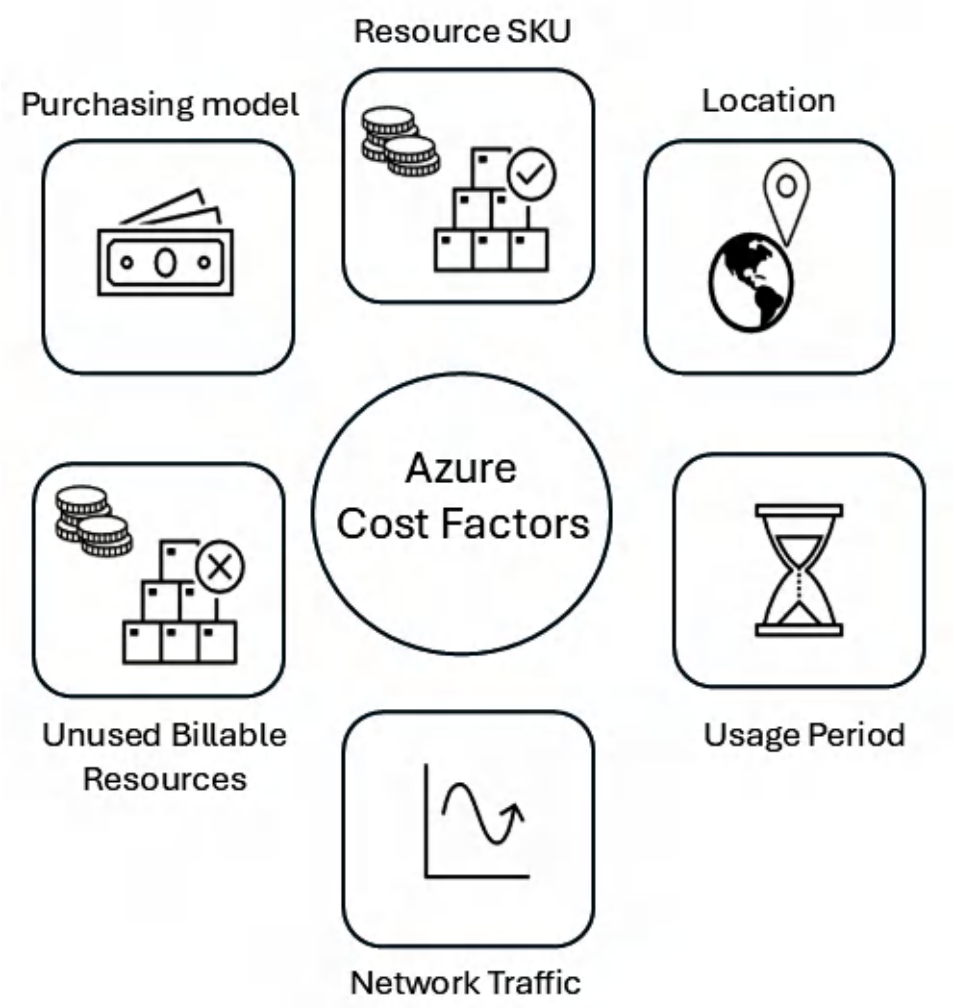


Figure 9.1: Azure cost factors

Each one of these is explored further in the following sections.

Purchasing model

The cost of Azure resources can differ depending on *how* Azure is purchased. Organizations may purchase Azure directly from Microsoft or through a Microsoft partner, which can affect billing relationships, invoicing, and how cost management features are accessed.

While the underlying metered pricing of Azure services remains consistent, the purchasing model can influence how charges may be discounted, consolidated, reported, and supported.

Understanding the purchasing model in use helps organizations align billing processes with operational and financial requirements.

> **Note**
>
> You can find the Azure purchase options at `https://azure.microsoft.com/en-us/pricing/purchase-options`.

Resource type and selected service tier (SKU)

Azure costs are specific to the **resource type** and its selected **service tier**, often referred to as a **stock keeping unit** (**SKU**). Each resource exposes one or more billing meters that define how usage is measured and charged; different resource types use different billing units. For example, data storage and data transfer are typically billed *per gigabyte*, **virtual machines** (**VMs**) are billed *per hour*, and Premium SSD managed disks are billed *per month*.

Higher service tiers or SKUs typically provide increased performance, scalability, or availability and therefore incur higher costs. In addition, some storage services may charge for read and write operations depending on the storage type used. Understanding the billing units and service tiers for each resource is essential when estimating and controlling Azure costs.

One more thing to consider is that different Azure **service models** introduce different cost characteristics. **Infrastructure as a service** (**IaaS**) workloads often provide the greatest control but require ongoing management of VMs, storage, and networking resources, all of which contribute to cost. **Platform as a service** (**PaaS**) and **serverless** offerings typically reduce operational overhead by abstracting infrastructure management. However, while these services may appear more expensive at first glance, they often include built-in scaling, availability, and maintenance that would otherwise require additional resources in an IaaS model.

Selecting the appropriate service model involves evaluating not only direct service costs but also the operational effort required to manage the workload over time.

Location (region)

Azure pricing varies **by region**, meaning the same resource deployed in different locations may incur different costs. These differences reflect factors such as infrastructure availability, demand, and operational overhead within each region.

While cost is an important consideration, regions should not be selected based solely on price. Latency requirements, compliance obligations, service availability, and potential data transfer charges must also be considered when choosing a deployment region.

Usage period

Many Azure resources are billed based on *how long* they run. This means that two identical VMs configured with the same specifications can incur very different costs if one runs continuously and the other is stopped and deallocated when not needed.

Be sure to remember scenarios such as when a VM is deallocated, compute charges stop; however, storage costs continue as long as disks are retained.

Some services—such as **Azure Bastion** (non-free SKU), **Azure VPN Gateway**, **App Service plans**, and **Entra Domain Services**—can continue to incur charges once created, even if they are not actively used. For these services, deleting the resource is the only way to stop charges.

Network traffic

Inbound data transfer into Azure is free. However, remember that outbound data transfer from Azure is billed on a per-gigabyte basis, regardless of whether traffic exits the region via the public internet, a VPN connection, or an ExpressRoute circuit.

Data transfer between resources within the same Azure region is not charged; architectures that minimize outbound and cross-region traffic can therefore help reduce ongoing network costs.

Unused billable resources

Some Azure resources incur charges even when they are not actively used. A common example is **public IP addresses** or a **VPN gateway**, which remain billable until they are explicitly removed. Regularly identifying and deleting unused resources is an important cost-optimization activity. Removing resources that are no longer required helps prevent unnecessary ongoing charges and improves overall cost efficiency.

In real-world environments, unused resources often accumulate gradually as projects evolve, test environments are abandoned, or architectures change. Regular reviews help ensure that temporary or transitional resources do not become permanent cost drivers.

You should also be aware that certain Azure resources do not incur charges and have no billing meters; examples include the following:

- User accounts or groups
- Resource groups
- Virtual networks
- Virtual network peering (data transfers are billed)
- Network interfaces

- Network security groups
- Availability sets

> **Note**
>
> Removing any of these will not reduce your costs or the total on the invoice you receive.

Knowing which resources incur costs and which ones do not is crucial.

With this, you have a fair understanding of the factors that affect costs, in addition to learning about some resource types that are free and have no billing meter or cost implications. With a clear view of what drives Azure costs, the next step is learning how to influence and optimize them.

Reducing and controlling costs

Effective cost control is an ongoing operational practice rather than a one-time activity. As workloads evolve and usage patterns change, previously efficient configurations may no longer represent the best balance between capability and cost. Cost optimization should not be approached as a purely technical exercise; business requirements such as security, compliance, availability, performance, and user experience often justify higher costs in certain areas of the environment. For example, production workloads supporting critical business functions may require higher service tiers or reserved capacity to ensure stability, while development or test environments can often tolerate lower-cost and operational configurations. Effective cost management involves aligning spending decisions with business priorities rather than minimizing costs indiscriminately.

By categorizing workloads based on business impact, organizations can apply different optimization strategies that reflect real operational needs. Effective cost management also depends on clear ownership and alignment with business priorities. When responsibility for cost is shared across teams, it becomes easier for inefficiencies to persist unnoticed. Assigning accountability for resource usage helps ensure that cost considerations are part of everyday operational decisions.

Cost control can be achieved through a combination of operational practices, tooling, and governance; Microsoft's **Cloud Adoption Framework** (**CAF**) includes cost management as a core governance discipline.

> **Note**
>
> You can find more information on this at `https://learn.microsoft.com/en-us/azure/cloud-adoption-framework/plan/estimate-total-cost-of-ownership`.

The following are some of the ways you can reduce and control costs:

- Optimize resources
- Azure Hybrid Benefit
- Azure reservations
- Spot pricing

Each of these is explored further in the following sections.

Optimize resources

Resource optimization is an operational activity that includes identifying unused resources and deleting them, right-sizing underutilized resources, and shutting down workloads that do not need to run continuously. It also involves evaluating whether IaaS workloads can be moved to PaaS, serverless, or SaaS alternatives. Azure Advisor identifies underutilized or unused resources and recommends actions to reduce costs.

Consider a development environment where VMs are provisioned to mirror production specifications but are only actively used during business hours. If these VMs run continuously, they incur unnecessary compute costs overnight and during weekends. By scheduling automated shutdowns outside of working hours or resizing the VMs to smaller SKUs, organizations can significantly reduce monthly costs without affecting developer productivity.

Over time, small optimizations such as these can result in substantial savings across large environments. Azure Advisor helps identify these opportunities by analyzing resource usage patterns and highlighting underutilized assets that may benefit from resizing or removal.

Azure Hybrid Benefit

This is a licensing benefit and allows an organization to maximize any investment in existing on-premises **software assurance** (**SA**)-enabled Windows Server or SQL licenses (or eligible subscription-based licenses). This removes the need to license and pay with the **pay-as-you-go** (**PAYG**) consumption model.

For a VM, this does not discount or remove the compute costs or any storage or networking costs; you are still liable for those and need to factor this into the total operating costs of a VM.

Azure reservations

This is a resource benefit and acts as a billing discount mechanism to reduce PAYG consumption charges. It does this by allowing you to commit to paying for an amount of capacity for a fixed term at a discounted rate compared to the PAYG consumption rate.

Reservations are available for a range of resources. For example, VMs make the most sense and are best used where the workloads must run for long periods of time or 24/7. Though compute costs are usually reduced by shutting down the VMs to save costs, this is no longer possible

with reservations and has no cost-saving benefit. In addition, for VMs, the reservations do not remove the software license costs or any storage or networking costs; you are still liable for those and need to factor this into the total operating costs of a VM.

Spot pricing

This is a resource benefit and allows an organization to make considerable savings by taking advantage of the unused capacity. Spot pricing is best used for workloads such as test/dev, analytics, machine learning, batch processes, and rendering that do not need a specific period in which they must run.

An analogy can be used for travel. If, for example, you plan to travel to a location and stay overnight at any time over the next week but are not sure which specific day or time to arrive, then you can take advantage of the days on which travel and accommodation will be the cheapest. You may have decided to take a vacation in the first week of the month, but if that is not a specific requirement, then you may be able to take advantage of cheaper pricing if you take your vacation in the last week of the month instead, or midweek instead of a weekend or public holiday. There is a risk, however, that those spot VMs can be evicted at any time when Azure compute demand goes up. Therefore, while bringing in a big cost benefit (up to 90% discount), it should be used for workloads that can handle interruptions.

Cost optimization should be reviewed regularly. Gradual usage growth, seasonal demand changes, or shifting workload requirements can introduce inefficiencies over time. Regular review helps ensure that cost-saving measures remain aligned with actual usage.

We have now looked at the factors influencing costs in Azure and learned about the purchasing model, the resource type, the location, the usage period, and the network traffic. We have also looked at how to reduce and control costs as part of the cost governance discipline. While understanding cost drivers and optimization techniques is essential, visibility is required to apply them effectively. The next section introduces Azure Cost Management, which provides the tools needed to monitor, analyze, and control Azure spending.

Azure Cost Management

Azure Cost Management is accessed through the **Cost Management + Billing** dashboard in the Azure portal and provides organizations with visibility into Azure spending across subscriptions, resource groups, and services. This visibility helps teams understand where costs are generated and how spending changes over time, supporting both operational and financial decision-making.

Azure Cost Management enables organizations to analyze historical cost data to identify trends, forecast future spending, and assess the financial impact of architectural and workload decisions. Administrators can also define budgets and configure alerts that provide early

warnings when spending approaches or exceeds defined thresholds, helping enforce financial governance and accountability.

To support cost optimization, Azure Advisor analyzes resource utilization patterns to identify underutilized or unused resources that may be contributing to unnecessary costs. Azure Advisor provides cost optimization recommendations that help organizations reduce spending, while Azure Cost Management continues to track, measure, and report on the financial outcomes of those optimization actions.

Figure 9.2 shows the cost analysis screen in the Azure portal:

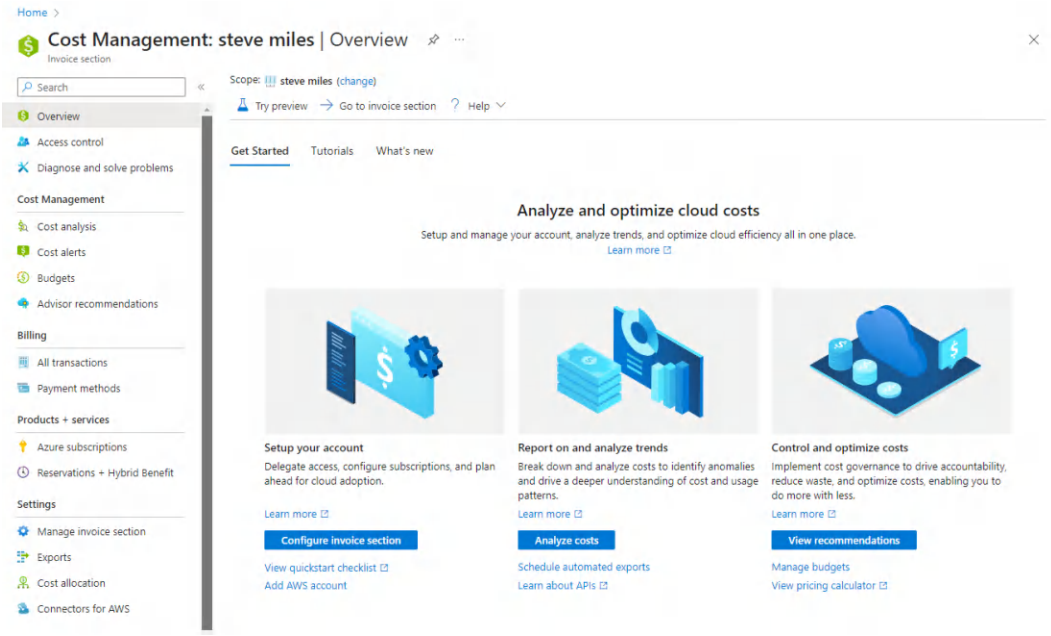


Figure 9.2: Managing costs associated with Azure services using Azure Cost Management

The two most important capabilities provided within the **Cost Management + Billing** function in the Azure portal are as follows:

- **Cost Management**: You can perform cost analysis, set cost alerts, and create budgets
- **Billing**: You can view and download invoices, view payment methods, and make payments

The **Azure Pricing Calculator** (covered in the following section) is particularly valuable during solution design, when multiple architectural options are being evaluated. By creating separate estimates for alternative designs, such as different regions or service tiers, organizations can compare expected costs before committing to a specific approach.

This use of the tools encourages cost-aware design decisions and helps avoid rework later in the deployment life cycle. While estimates are not guarantees, they provide useful directional insight when combined with an understanding of workload requirements.

In this section, we have understood that Azure Cost Management is used to monitor and analyze costs, set budgets, generate invoices and reports, and perform other tasks related to the financial aspects of using Azure resources.

While Azure Cost Management focuses on monitoring and controlling existing spend, estimating costs before deployment is equally important—this is where the Azure Pricing Calculator plays a role.

Azure Pricing Calculator

The Azure Pricing Calculator is a publicly accessible, browser-based tool used to estimate the cost of services before they are deployed in Azure. It allows organizations to explore expected costs based on selected services, configurations, and usage assumptions, helping to support cost-aware planning and design decisions.

Cost estimation tools are intended to provide guidance rather than guaranteed pricing. Actual Azure costs may vary due to changes in usage patterns, service availability, pricing updates, or regional differences. As a result, estimates produced by the calculator should be treated as indicative rather than final.

The Azure Pricing Calculator is shown in *Figure 9.3*:

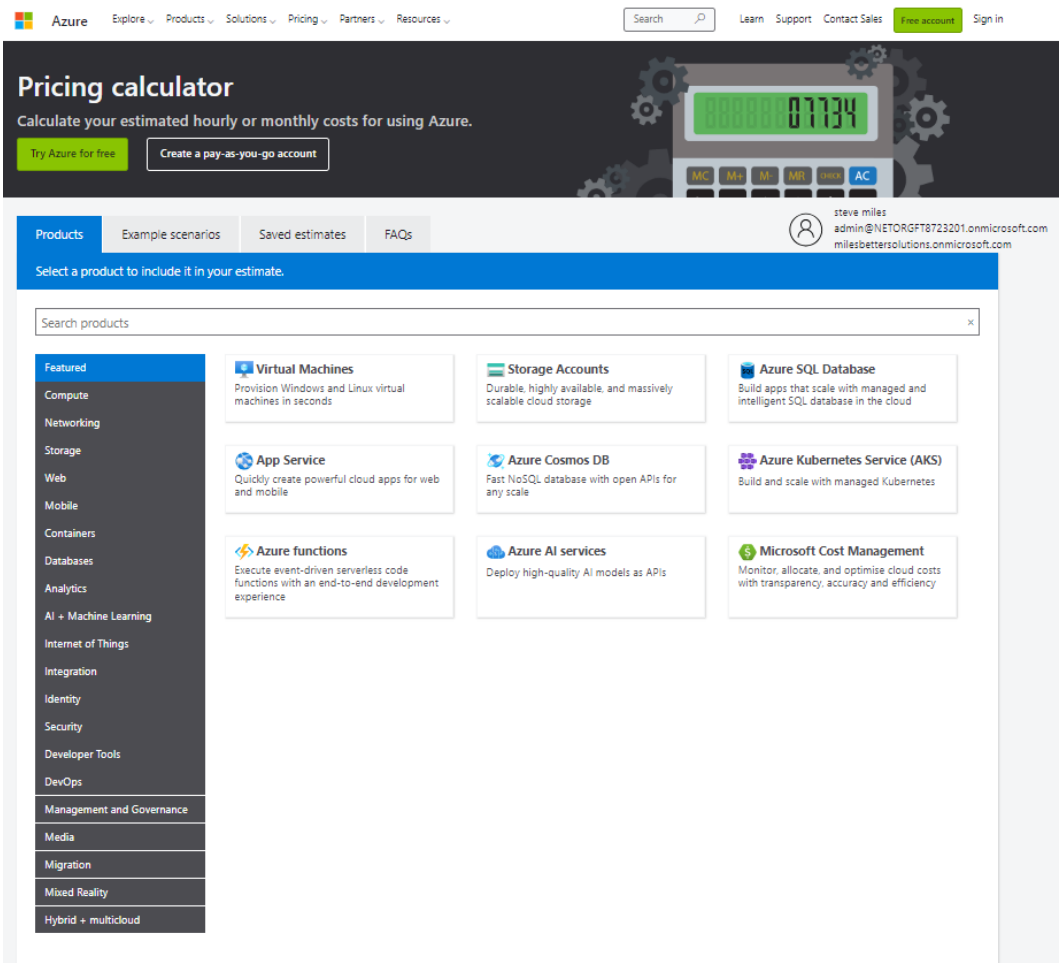


Figure 9.3: Azure Pricing Calculator showing categorized resources

All Azure resources available for purchase are organized into categories within the Pricing Calculator, allowing users to browse or search for specific services. When a resource is added to an estimate, configuration options such as region, service tier, and expected usage can be adjusted to reflect anticipated deployment scenarios.

Each resource includes a link to its detailed pricing page, which explains how costs are calculated and which configuration choices influence pricing. As resources are added, the calculator produces a consolidated estimate with a per-service cost breakdown, and supports changing currency, reviewing line items, and exporting or sharing estimates.

Because the calculator is based on user-defined assumptions, it is best suited for early-stage planning and comparing alternative architectures, rather than predicting exact monthly costs. Adjusting variables such as region, tier, or usage helps illustrate how design decisions affect overall spend.

> **Note**
>
> You can access the Azure pricing calculator at `https://azure.microsoft.com/en-us/pricing/calculator/`.
>
> At the time of writing, Microsoft has announced the deprecation of the Azure TCO Calculator. However, the concept of TCO estimation remains relevant for understanding Azure cost comparisons.

Used alongside cost monitoring tools, the Pricing Calculator helps organizations make informed decisions early, before resources are deployed and costs are incurred.

Cost allocation using tags is covered in *Chapter 10*, *Azure Governance and Compliance*.

Summary

This chapter included coverage of the AZ-900 Azure Fundamentals exam *Skills Measured* area *Describe Azure Management and Governance*.

In this chapter, you saw how Azure's consumption-based pricing model works, learned about the key factors that influence cost behavior, explored how Azure Cost Management supports monitoring and budgeting, and used the Pricing Calculator to estimate costs before deployment.

This chapter balanced exam-focused knowledge with practical context, enabling you to understand Azure cost behavior for both the AZ-900 exam and real-world Azure usage.

In the next chapter, you will learn about Azure governance and compliance.

Exam Readiness Drill – Chapter Review Questions

Apart from mastering key concepts, strong test-taking skills under time pressure are essential for acing your certification exam. That's why developing these abilities early in your learning journey is critical.

Exam readiness drills, using the free online practice resources provided with this book, help you progressively improve your time management and test-taking skills while reinforcing the key concepts you've learned.

HOW TO GET STARTED

- Open the link or scan the QR code at the bottom of this page
- If you have unlocked the practice resources already, log in to your registered account. If you haven't, follow the instructions in *Free Benefits with Your Book* section in this book's *Preface* and come back to this page.
- Once you log in, click the START button to start a quiz
- We recommend attempting a quiz multiple times till you're able to answer most of the questions correctly and well within the time limit.
- You can use the following practice template to help you plan your attempts:

Working On Accuracy		
Attempt	**Target**	**Time Limit**
Attempt 1	40% or more	Till the timer runs out
Attempt 2	60% or more	Till the timer runs out
Attempt 3	75% or more	Till the timer runs out
Working On Timing		
Attempt 4	75% or more	1 minute before time limit
Attempt 5	75% or more	2 minutes before time limit
Attempt 6	75% or more	3 minutes before time limit

The above drill is just an example. Design your drills based on your own goals and make the most out of the online quizzes accompanying this book.

First time accessing the online resources?

You need to unlock the online practice resources to access the link below. Check the *Free Benefits with Your Book* section in the *Preface* for detailed instructions on how to do this. You only need to unlock the resources once.

Open Quiz `https://packt.link/AZ-9003ech9` or scan the QR code

10

Azure Governance and Compliance

In *Chapter 9*, *Azure Cost Management*, we looked at cost management, the factors that can affect costs in Azure, and the tools and resources available, such as **Azure Cost Management** and the **Azure pricing calculator**. This chapter primarily focuses on the *Describe Azure Management and Governance* module from the *Skills Measured* section of the *AZ-900 Azure Fundamentals* exam.

In this particular section, candidates are assessed on their understanding of how Azure enforces governance, manages compliance, and applies organizational controls across resources. The topics covered here, such as Azure Policy, Resource Locks, tags, and Microsoft Purview, represent the core governance capabilities you are expected to recognize and describe at a foundational level.

Together, these topics explain how Azure translates governance requirements into practical controls, which is why understanding their purpose and role will help you confidently answer governance and compliance questions in the AZ-900 exam.

A detailed breakdown of how these topics map to individual exam skills is provided in the appendix for reference, but this chapter contains everything required to understand and answer governance-related questions on the exam.

By the end of this chapter, you will be able to confidently answer questions on the following topics:

- Microsoft Purview in Azure
- Azure Policy
- Azure Resource Locks

- Tags
- Microsoft Trusted Cloud Principles

In addition, this chapter's goal is to take your knowledge beyond the exam's content to prepare you for a real-world, day-to-day Azure-focused role.

Microsoft Purview in Azure

Microsoft Purview in Azure (previously *Azure Purview*) is a data governance service. It helps organizations discover, classify, protect, and manage sensitive information across cloud and hybrid environments. Rather than focusing on servers or virtual machines, Purview focuses on the data itself.

Microsoft Purview in Azure is designed for roles responsible for creating, consuming, and governing data, including data producers, data consumers, and data officers. To better understand how Microsoft Purview operates across an organization's data estate, consider the high-level architecture shown in the following diagram.

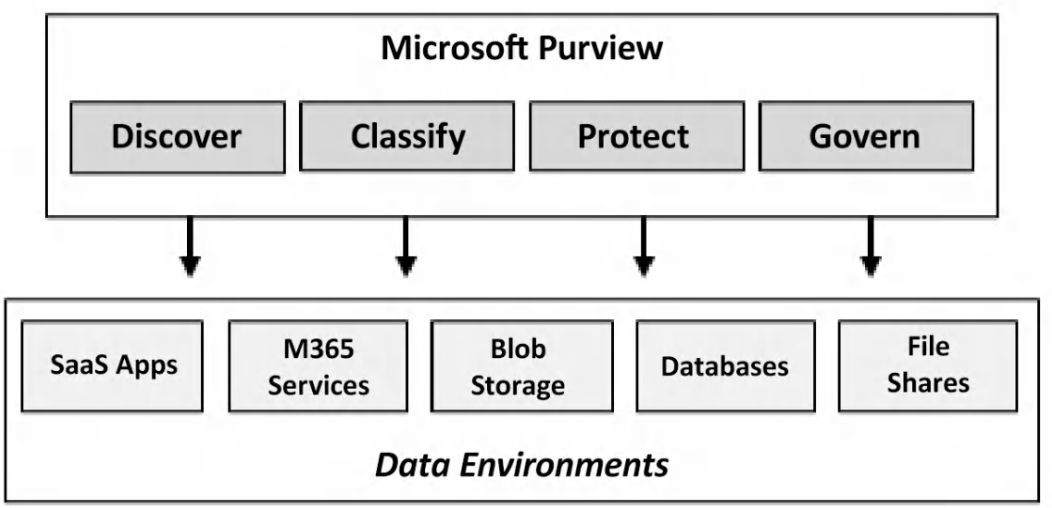


Figure 10.1: How Microsoft Purview governs data across the data estate

The diagram illustrates how Microsoft Purview delivers its core governance functions—**Discover**, **Classify**, **Protect**, and **Govern**—across diverse data environments, including SaaS applications, Microsoft 365 services, Blob storage, databases, and file shares. Rather than being tied to specific infrastructure, Purview applies these capabilities consistently across distributed data sources, reinforcing its data-centric approach to governance in cloud and hybrid

environments. Microsoft Purview in Azure enables you to gain insights and stay informed about your distributed data landscape through the following capabilities:

- **Data discovery automation**, allowing organizations to automatically locate data across cloud, on-premises, and hybrid sources
- **Mapping of an entire data estate**, providing a unified view of where data resides and how it is organized
- **Classification of sensitive data**, enabling identification of personal, financial, and regulated information
- **Lineage of data end-to-end**, showing how data moves, transforms, and is consumed across systems
- **Data storage and usage insights**, offering visibility into how and where data is used
- **Data access management at scale**, supporting consistent governance across large and distributed environments
- **A governed data security environment**, where protection policies are applied consistently and transparently

Note

You can find more information on data governance solutions at `https://learn.microsoft.com/en-us/purview/data-governance-overview`.

These capabilities position Microsoft Purview as a **data-centric governance service**.

Rather than focusing on the infrastructure that stores data, Microsoft Purview focuses on the information itself, ensuring that it remains discoverable, classified, and protected regardless of where it is stored or how it is shared. This information-centric approach begins with understanding what data exists and how sensitive it is; therefore, data classification is a foundational capability of Microsoft Purview.

Data classification and sensitivity labels

A core capability of Microsoft Purview is **data classification**. Data classification enables organizations to identify information that requires special handling, such as personal, financial, or regulated data; once identified, this data can be protected using **sensitivity labels**.

Sensitivity labels define how data should be treated. Labels can be applied **manually** by users, **automatically** based on defined rules, or suggested to users when certain conditions are met; for example, a document containing financial information can automatically receive a label that identifies it as confidential.

Unlike traditional access controls, sensitivity labels are applied **directly to the data**; this means the protection remains in place even if the data is moved, copied, or shared outside its original location. This model reflects the realities of modern, distributed cloud environments where data frequently crosses system and organizational boundaries.

Protecting sensitive information

Once a sensitivity label is applied, Microsoft Purview can enforce **protection controls** on the data. These controls may include encryption, access restrictions, and visual markings such as headers, footers, or watermarks. Visual markings provide a clear indication of the sensitivity of the information and help users understand how the data should be handled.

Microsoft Purview can identify common **sensitive information types** (**SITs**), such as financial data or personal identifiers, and apply protections automatically. This reduces the risk of accidental data exposure while supporting secure collaboration across teams and organizations.

It is important to note that Microsoft Purview operates at the **content level**; it protects documents and emails rather than managing or restricting Azure resources. This distinction separates Microsoft Purview from governance tools that focus on infrastructure configuration or access control.

Microsoft Purview and Azure governance tools

Microsoft Purview complements other Azure governance and management services while serving a distinct role. Services such as **Azure Policy**, **Resource Locks**, and **role-based access control** (**RBAC**) govern how Azure resources are created, configured, or accessed. Microsoft Purview governs **the data stored within those resources**.

By separating **infrastructure governance** from **data governance**, Azure enables organizations to apply the right controls at the right layer. Microsoft Purview provides the data-focused layer that ensures sensitive information remains protected, even as it moves across services, users, and environments.

With this distinction between infrastructure governance and data governance established, it's useful to see how Microsoft Purview is applied in real-world environments.

Microsoft Purview in practice

In practice, Microsoft Purview is used to establish consistent data governance across distributed environments where information is created, shared, and stored across multiple services and locations. As organizations adopt cloud platforms and collaborative working models, data often moves beyond traditional network boundaries, making location-based security controls insufficient on their own.

By embedding classification and protection directly into the data, Microsoft Purview enables organizations to maintain control over sensitive information regardless of where it resides. This approach allows data to be shared internally or externally while still enforcing organizational handling requirements, such as encryption, access restrictions, or visual markings.

Microsoft Purview is commonly used alongside other Azure governance services; so, while Azure Policy and Resource Locks govern how infrastructure is deployed and protected, Microsoft Purview governs the information stored within that infrastructure. This separation of responsibilities ensures that governance controls are applied at the appropriate layer, reducing complexity while improving overall security and compliance posture.

Used effectively, **Microsoft Purview helps organizations balance data protection, regulatory requirements, and user productivity**, providing visibility and control without restricting legitimate collaboration.

You learned in this section how Microsoft Purview relates to Azure through its data governance solution capabilities. Next, you will look at Azure Policy.

Azure Policy

Azure Policy is a governance service that enables organizations to **define**, **assess**, and **enforce rules** for how Azure resources are created and managed. It operates consistently across environments, helping ensure that deployments align with organizational standards, regulatory requirements, and architectural decisions.

Azure Policy is applied at scale and is not dependent on individual user permissions; instead, it evaluates resource properties and configurations, ensuring that resources conform to defined rules throughout their lifecycle. Azure Policy is a set of resource creation and management rules that apply across multiple subscriptions; it provides the following functions and capabilities:

- **Defines what actions are allowed within a subscription**, regardless of which user is performing them
- **Assesses resources to ensure that compliance standards are met**, providing visibility into configuration drift
- **Enforces organization mandates**, such as architectural standards or regulatory controls
- **Automates the remediation process**, addressing drift or non-compliance where possible

One typical example use case of Azure Policy is to limit which regions can be used when creating resources, ensuring that **data sovereignty and regulatory requirements** are met; by restricting deployments to approved regions, organizations can reduce compliance risk while maintaining centralized control.

Azure Policy can also be used to limit which **virtual machine sizes**, **storage account types**, or other resource options are allowed. This helps prevent the deployment of expensive, unsupported, or operationally inefficient resources and promotes consistency across environments.

While Azure Policy defines what resources are allowed to exist, understanding how access to those resources is controlled requires examining Azure **role-based access control** (**RBAC**).

Azure Policy versus Azure role-based access control

Azure Policy is often compared with Azure RBAC, but the two services solve different governance problems. Azure Policy controls what can exist, while RBAC controls who can perform actions.

Table 10.1 lists the key differences to comprehend between Azure Policy and Azure RBAC:

Azure Policy	**Azure RBAC**
Controls what can be done (*regardless of the user*)	Controls who can do what (*specific to each user*)
Focuses on *resource properties*	Focuses on *user actions*
Applied during the creation or change of Azure resources	Applied to each action—create, read, update, and delete
No specific roles model in place	Built-in and custom roles
Supports different effects such as deny, audit, and `deployIfNotExists`	Default deny, explicit allow

Table 10.1: Distinguishing factors between Azure Policy and RBAC

An example of Azure Policy in action would be a user who has contributor permissions on a resource group but is blocked from deploying a virtual machine in an unapproved region due to a policy assignment. In this case, the user's permissions are valid, but the resource configuration violates an organizational governance rule.

Another example would be a restriction preventing the deployment of an older or unsupported virtual machine type or size, such as a specific D-series generation. Even though the user has permission to create virtual machines, the policy ensures that only approved configurations are allowed.

By contrast, an RBAC example would involve granting a user contributor permissions in a development subscription while limiting them to virtual machine administrator permissions in a production subscription. Here, **the control is based on who the user is and what role they have**, not on the properties of the resources being deployed.

With the distinction between policy-based control and identity-based access established, the next step is to understand how Azure Policy is defined and enforced in practice.

> **Note**
>
> If you would like a refresher on identity-based access control, see *Chapter 7, Azure Identity and Access*, for a deeper discussion of RBAC.

How Azure Policy works

Azure Policy rules are defined using **policy definitions**, which describe the conditions under which a policy applies and the effect that should be enforced.

Policy definitions are written in **JavaScript Object Notation** (**JSON**) format and represent the business rules that organizations want to apply consistently. Multiple policy definitions can be grouped into a **policy initiative** (also known as a **policy set**). Policy initiatives simplify governance by allowing related rules to be assigned and managed as a single unit, such as a baseline set of security or compliance requirements.

Once defined, policies or initiatives are applied through **policy assignments**. These assignments determine the scope at which the policy is enforced; Azure Policy can be assigned at the subscription, resource group, or individual resource level, allowing organizations to balance centralized governance with localized flexibility.

After the assignment, Azure Policy evaluates all resources within the defined scope and identifies whether they comply with the assigned rules. Depending on the effect configured, Azure Policy may block non-compliant deployments, audit resources for reporting purposes, or deploy required configurations automatically.

Figure 10.2 illustrates how Azure Policy can be scoped at different levels within Azure:

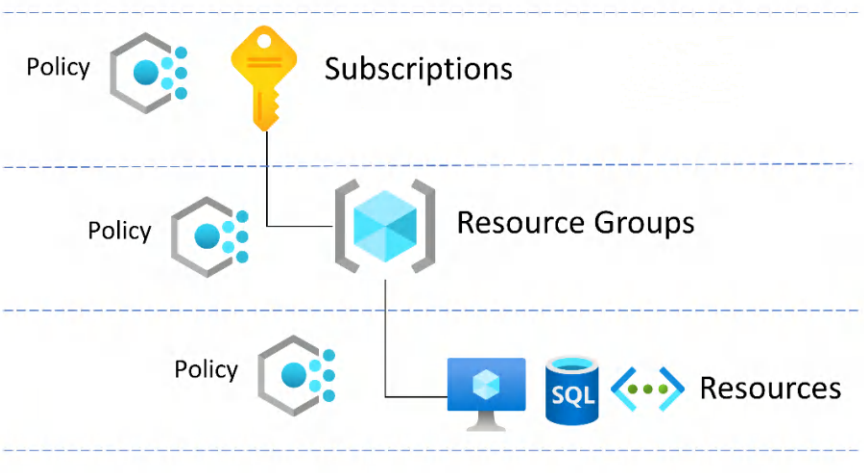


Figure 10.2: Azure Policy scope level

Figure 10.2 shows how Azure Policy assignments can be scoped at different levels within Azure, allowing governance rules to be applied broadly or narrowly depending on organizational needs. A policy assigned at the subscription level is inherited by all resource groups and resources beneath it, ensuring consistent enforcement across an entire environment. Assigning a policy at the resource group level limits its scope to a specific workload or application boundary, while assigning it directly to a resource enforces rules only on that individual resource.

For example, an organization might assign a policy at the subscription level to restrict deployments to approved regions, while applying a more specific policy at a resource group level to enforce allowed virtual machine sizes for a particular application. This hierarchical scoping model enables centralized governance without removing flexibility where exceptions are required.

With an understanding of how Azure Policy definitions are scoped and evaluated, the next section explores how these controls are applied in real-world scenarios to enforce governance consistently across environments.

Note

Management groups, while not referenced directly in this context, were explored in *Chapter 3, Azure Core Architectural Components*, and provide an additional layer for applying governance across multiple subscriptions.

Azure Policy in practice

In practice, **Azure Policy acts as a preventive governance control**. It helps organizations move away from manual reviews and reactive enforcement by embedding rules directly into the resource deployment process; this ensures that standards are applied consistently, even in large or rapidly changing environments.

By combining assessment, enforcement, and remediation capabilities, Azure Policy supports both governance and operational efficiency. It reduces the risk of misconfiguration, simplifies compliance reporting, and helps maintain architectural consistency across cloud environments.

With an understanding of Azure Policy and how it differs from RBAC, the next section explores **Resource Locks**, which provide a different form of protection focused on preventing accidental changes or deletion.

Azure Resource Locks

Azure Resource Locks are used to protect Azure resources from **accidental modification** or **deletion**. This differs from RBAC, which determines who can perform actions on resources; Resource Locks act as a **final safeguard** by preventing destructive operations, even when a user has elevated-level permissions.

The most common and important use of Resource Locks is to prevent **accidental deletion** of critical resources. Locks **override** permissions that have been granted through RBAC, meaning that even administrators or contributors **cannot delete** or **modify** locked resources unless the lock is first **removed**.

Resource Locks are managed at the **subscription**, **resource group**, and **individual resource levels** and can be configured as one of the following types:

- **Read-only lock**: Administrators *cannot* delete or update a resource
- **Cannot-delete lock**: Administrators *can* update a resource, but *cannot* delete it

These two lock types serve different protection purposes.

A **read-only lock** is typically used for highly sensitive or stable resources that should not be changed, while a **cannot-delete lock** is commonly applied to resources that may require configuration updates but must not be removed.

Lock inheritance and scope

Unlike resource tags, **Resource Locks are inherited by child resources**. This means that when a lock is applied at a higher scope, such as a resource group or subscription, **all resources** within that scope automatically **inherit the lock**. For example, applying a cannot-delete lock to a resource group ensures that none of the resources within that group can be deleted, even if individual resources do not have locks applied directly. This inheritance model makes Resource Locks particularly effective for protecting foundational or shared infrastructure components.

It is also possible to apply **multiple locks** to a single resource. When this occurs, the **most restrictive lock takes precedence**. For instance, if a resource inherits a cannot-delete lock from a resource group and also has a read-only lock applied directly, the read-only lock will be enforced.

Figure 10.3 illustrates the different levels at which locks can be applied and how they are inherited by child resources.

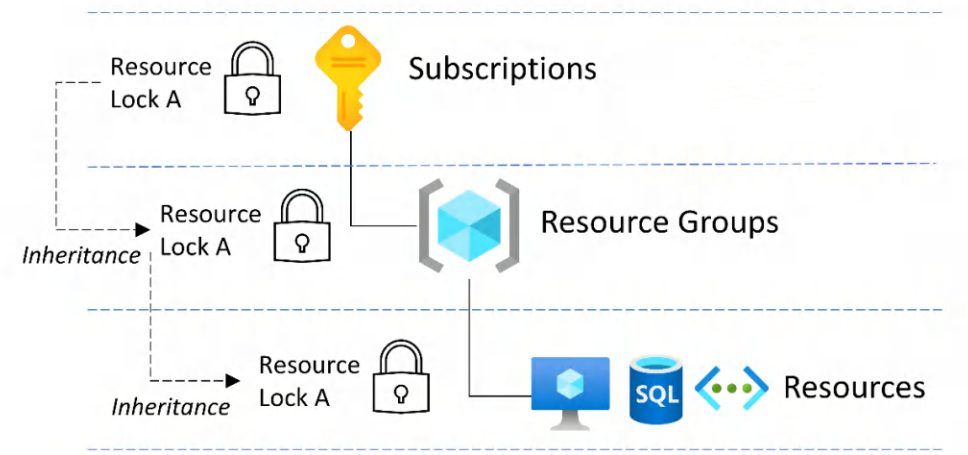


Figure 10.3: Azure Resource Locks preventing accidental deletion of resources

Figure 10.3 shows how Resource Locks are applied at different scopes in Azure and how they are inherited by child resources. When a lock is applied at the subscription or resource group level, it is automatically inherited by all resources within that scope, even if those resources do not have locks applied directly. This means that a single lock at a higher level can protect multiple dependent resources consistently. The diagram also illustrates that inheritance flows downward only, ensuring that protections applied to foundational scopes cannot be bypassed unintentionally at lower levels.

Resource Locks play a distinct role within Azure governance. While Azure Policy is used to enforce rules about how resources are created and configured, Resource Locks are focused on protecting existing resources from change or removal; they do not evaluate compliance, enforce standards, or control deployment options. Because locks override RBAC permissions, they are particularly useful in environments with multiple administrators or shared operational responsibility. Locks provide an additional layer of protection against human error, ensuring that critical resources remain intact even during routine maintenance or troubleshooting activities.

Managing Resource Locks

Resource Locks can be created and managed using multiple Azure management tools, including the Azure portal, Azure PowerShell, the Azure CLI, ARM templates, and the REST API. This flexibility allows locks to be applied both manually and as part of automated deployment or governance processes.

For example, a cannot-delete lock is commonly applied to a production resource group to prevent accidental deletion of critical services during routine administration.

To create or delete a Resource Lock, a user must have appropriate permissions. Typically, this requires ownership of the resource or assignment of the User Access Administrator role. In more controlled environments, a custom role can be created that grants only the specific permissions required to manage locks, reducing the risk of excessive privilege.

Resource Locks in practice

In real-world Azure environments, Resource Locks are commonly used to protect critical components such as networking infrastructure, identity-related resources, or shared services that support multiple workloads. By preventing accidental deletion, locks help ensure service continuity and reduce the likelihood of outages caused by operational mistakes.

Resource Locks are most effective when used selectively and intentionally. Overuse of locks can make environments difficult to manage, particularly when legitimate changes are required. For this reason, locks are often combined with clear operational processes and documentation to ensure that protections can be adjusted safely when necessary.

In summary, Azure Resource Locks provide a simple but powerful mechanism for protecting resources from accidental modification or deletion. By overriding RBAC permissions and supporting inheritance across scopes, locks ensure that critical resources remain protected regardless of user permissions. As part of Azure governance, Resource Locks complement policy-based enforcement and access control by adding a focused layer of protection for existing resources.

While Resource Locks focus on protecting existing resources from accidental change or deletion, governance also requires a way to consistently organize and describe resources at scale. This is where tags come in.

Tags

Tags provide **metadata**, or **descriptive information**, for Azure resources. Metadata is a way to *describe* data, adding context that is not inherently part of the resource configuration itself. In Azure, tags are used to label resources with additional information that helps organizations organize, manage, and understand their environments.

A useful way to think about tags is as **sticky notes or comments** attached to resources. Much like a comment in a document or a tooltip in an application, a tag does not change how a resource behaves, but it provides meaningful information about what the resource represents, who owns it, or how it should be managed.

Tags can be created and managed using a variety of Azure tools, including the Azure portal, Azure PowerShell, the Azure CLI, ARM templates, and the REST API. Tags can also be governed through Azure Policy, which allows organizations to enforce consistent tagging practices or identify resources that are missing required tags.

Tag structure and behavior

Each tag consists of a **tag name** and a **tag value**, forming a simple key-value pair. Both the name and the value are user-defined, allowing organizations to design tagging schemes that align with their internal processes and reporting requirements.

Up to **15 tags** can be applied to each resource. Tags do not inherit automatically. If a tag is applied at the resource group level, it applies only to that resource group and does not flow down to the resources contained within it. This behavior allows resources to be grouped logically by resource group while still being labeled independently with metadata that is not tied to the subscription or group structure.

Figure 10.4 illustrates how tags are applied to resources and how key-value pairs are used to describe resource metadata.

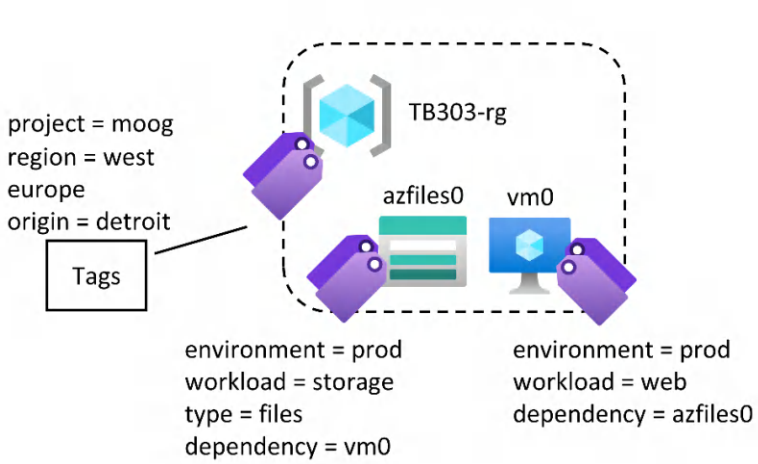


Figure 10.4: Tags used for resource metadata

Figure 10.4 shows how tags are applied directly to individual Azure resources, rather than being inherited automatically from the resource group. In the example, resources within the same resource group share some common tags, such as environment, while also having resource-specific tags that describe workload type and dependencies. This demonstrates how tags can be used to add meaningful metadata that reflects business context, relationships, and purpose, independent of the resource group or subscription structure.

Once applied, tags can be used to logically organize resources that share common characteristics. Resources with the same tag values can be filtered, grouped, or targeted for reporting and management activities. While tags do not enforce behavior on their own, they provide critical context that enables governance processes and operational insight.

Tags can also be referenced by other Azure services. For example, Azure Policy can be used to identify resources that are missing required tags or to enforce tagging standards during resource creation. Automation and scripts can also use tags to perform actions against groups of resources that share the same metadata.

Common tag use cases include the following:

- **Billing information**, such as cost center, billing ID, project ID, team, business unit, or region
- **Ownership information**, including department or team name, named point of contact, or stakeholder

- **Purpose information**, such as environment (development, test, staging, or production) or application name
- **Classification and compliance information**, including service-level agreements, confidentiality markings, region, or compliance standards

These tagging categories help create a shared taxonomy that can be consistently applied across Azure environments.

Beyond organizing and classifying resources, tags also play a critical role in making cloud spend visible and accountable, which naturally leads to how they are used for cost management.

Tags and cost management

One of the most common and practical uses of tags is **cost management**. By tagging resources with **billing-related metadata**, organizations can break down costs by project, department, team, or workload. This enables more granular visibility into spending and supports internal reporting models such as show-back or charge-back.

For example, by tagging all production resources with `environment=prod` and `costcenter=finance`, an organization can quickly attribute monthly Azure spend to the correct business unit without restructuring subscriptions.

Tags appear in the CSV export of Azure billing and usage data; this allows cost data to be filtered, grouped, and processed using automation or external reporting tools.

By combining tags with cost analysis, organizations can better understand how resources are being consumed and identify opportunities for optimization.

Tags in practice

In practice, tags are most effective when used consistently and intentionally. A well-defined tagging strategy helps prevent sprawl, improves visibility, and simplifies reporting across large or complex Azure environments. While tags are flexible by design, organizations often define a standard set of required tags to ensure that critical metadata is always present.

Tags do not replace governance controls such as Azure Policy or resource locks. Instead, they complement those controls by providing descriptive context that helps teams understand the purpose and ownership of resources. When used together, tags, policies, and locks form a layered governance approach that balances flexibility with control.

In summary, tags provide a lightweight but powerful mechanism for adding descriptive metadata to Azure resources. By using key-value pairs to capture information such as ownership, purpose, classification, and cost allocation, tags help organizations organize resources, improve visibility, and support governance and cost management processes. As part

of an Azure governance strategy, tags enable better insight and operational clarity without altering how resources function.

By adding descriptive context to resources, tags help organizations understand *what* exists and *why* it exists, but they do not define how Microsoft operates or secures the cloud itself. To complete the governance picture, it is also important to understand the foundational principles Microsoft applies to operating Azure as a trusted cloud platform.

Microsoft Trusted Cloud Principles

The **Microsoft Trusted Cloud Principles** describe the foundational commitments that underpin how cloud services are designed, operated, and governed. These principles focus on **security**, **privacy**, **compliance**, and **transparency**, and they establish the basis of trust between customers and Microsoft when adopting cloud services.

These principles are not abstract concepts. They are embedded in Microsoft's cloud architecture, operational practices, and contractual commitments, and they define how responsibilities are shared between the cloud provider and the customer in a cloud operating model.

Figure 10.5 highlights the Microsoft Trusted Cloud Principles.

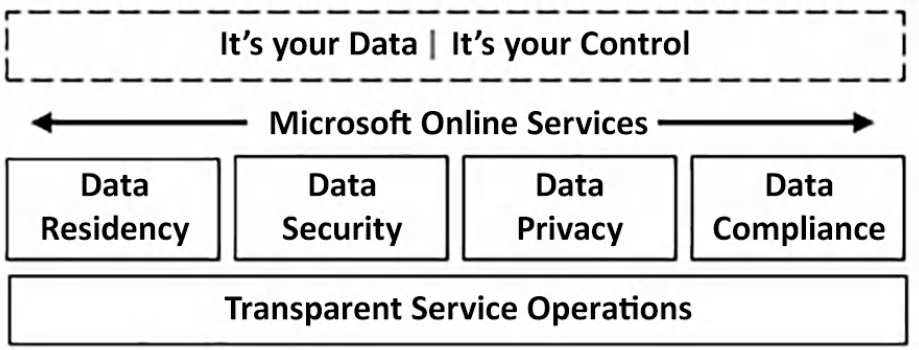


Figure 10.5: Microsoft Trusted Cloud Principles

As illustrated in *Figure 10.5*, customers retain ownership and control of their data, while Microsoft is responsible for delivering and operating a cloud services platform that supports data residency requirements and ensures data is kept secure, private, and compliant with recognized regulatory and compliance standards. These commitments are not merely aspirational; they are backed by enforceable contractual guarantees.

The Microsoft Trusted Cloud Principles provide the **conceptual foundation for Azure governance**.

Services such as Microsoft Purview, Azure Policy, Resource Locks, and tags translate these principles into practical controls that allow customers to manage data, enforce standards, protect resources, and maintain visibility across their environments.

By combining platform-level assurances with customer-controlled governance mechanisms, Azure enables organizations to adopt cloud services with confidence while retaining responsibility and control over their security, privacy, and compliance outcomes.

In summary, the Microsoft Trusted Cloud Principles define how trust is established in the cloud through shared responsibility, strong security foundations, data privacy protections, regulatory compliance, and operational transparency. Together, they underpin Azure governance capabilities and help organizations design, manage, and operate cloud environments that are secure, compliant, and trustworthy.

With this understanding of Microsoft's Trusted Cloud Principles, this concludes the learning content for this chapter.

Summary

In this chapter, you learned how Azure management and governance services help organizations control resources, data, and costs in the cloud. Microsoft Purview was introduced as a data-centric governance service for discovering, classifying, and protecting information. Azure Policy was covered as a mechanism for enforcing rules and standards across Azure environments, while Resource Locks were explained as a way to protect critical resources from accidental change or deletion. Tags were explored as a lightweight method for organizing resources, improving visibility, and supporting cost management. The chapter concluded with an introduction to the Microsoft Trusted Cloud Principles, which provide the foundational trust and shared responsibility model that underpins Azure governance.

Further knowledge beyond the required exam content was provided to prepare for a real-world, day-to-day Azure-focused role.

The next chapter will look at **Azure Resource Deployment and Management**.

Exam Readiness Drill – Chapter Review Questions

Apart from mastering key concepts, strong test-taking skills under time pressure are essential for acing your certification exam. That's why developing these abilities early in your learning journey is critical.

Exam readiness drills, using the free online practice resources provided with this book, help you progressively improve your time management and test-taking skills while reinforcing the key concepts you've learned.

HOW TO GET STARTED

- Open the link or scan the QR code at the bottom of this page
- If you have unlocked the practice resources already, log in to your registered account. If you haven't, follow the instructions in *Free Benefits with Your Book* section in this book's *Preface* and come back to this page.
- Once you log in, click the START button to start a quiz
- We recommend attempting a quiz multiple times till you're able to answer most of the questions correctly and well within the time limit.
- You can use the following practice template to help you plan your attempts:

Working On Accuracy		
Attempt	**Target**	**Time Limit**
Attempt 1	40% or more	Till the timer runs out
Attempt 2	60% or more	Till the timer runs out
Attempt 3	75% or more	Till the timer runs out
Working On Timing		
Attempt 4	75% or more	1 minute before time limit
Attempt 5	75% or more	2 minutes before time limit
Attempt 6	75% or more	3 minutes before time limit

The above drill is just an example. Design your drills based on your own goals and make the most out of the online quizzes accompanying this book.

First time accessing the online resources?

You need to unlock the online practice resources to access the link below. Check the *Free Benefits with Your Book* section in the *Preface* for detailed instructions on how to do this. You only need to unlock the resources once.

Open Quiz `https://packt.link/AZ-9003ech10` or scan the QR code

11

Azure Resource Deployment and Management

In *Chapter 10*, *Azure Governance and Compliance*, you were introduced to Microsoft Purview in Azure, Azure Policy, resource locks, and tags. This chapter will outline resource deployment, management capabilities, and tooling in Azure. It primarily focuses on the *Describe Azure Management and Governance* module from the *Skills Measured* section of the AZ-900 Azure Fundamentals exam.

Together, these tools and services show how Azure resources are created, managed, and controlled in practice, which is why understanding their purpose and usage patterns will help you answer deployment and management questions in the AZ-900 exam.

By the end of this chapter, you will be able to confidently answer questions on the following topics:

- Azure portal
- Azure PowerShell
- Azure CLI
- Azure Cloud Shell
- ARM templates in Azure
- Azure Arc
- Microsoft Copilot in Azure

In addition, this chapter's goal is to take your knowledge beyond the exam's content to prepare you for a real-world, day-to-day Azure-focused role. By exploring multiple management interfaces, automation approaches, and deployment patterns, the chapter shows not only *what*

the Azure tools are, but *how* they are used together in practical operational scenarios you are likely to encounter in production environments.

Azure portal

The **Azure portal** is a public-facing browser-based **Graphical User Interface** (**GUI**) console for interacting with Azure resources; it can be accessed at `https://portal.azure.com`.

The Azure portal is designed for *self-service* and is the most common method for creating and managing your Azure environments. It is the quickest way for anybody new to Azure to get started and carry out simple tasks.

Figure 11.1 shows the Azure portal home page, highlighting the primary navigation elements and common entry points used to create, view, and manage Azure resources.

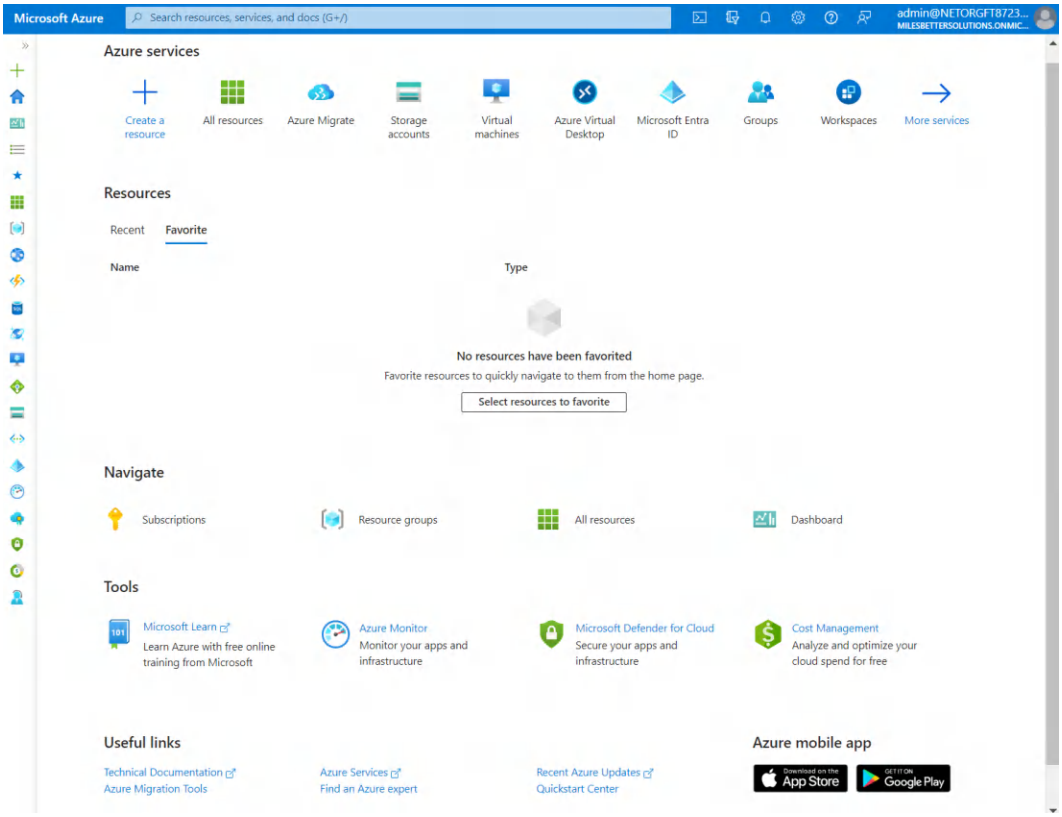


Figure 11.1: The Azure portal

The Azure portal allows users to create, change, and delete resources directly within the portal as a seamless single-user-interface experience. It provides a quick and straightforward way to

access Azure resources, often through dashboard views or high-level overview visualizations; these can also be exported in various formats, such as reports and Excel, based on the resource type.

Being a **web-based console**, the portal can be accessed from anywhere with an internet connection and from all modern desktops, laptops, and tablet devices running Windows, Android, and so on. The only limitation is that your browser must have JavaScript enabled, and your OS must be capable of running one of the following supported browsers:

- Microsoft Edge (*latest version*)
- Safari (*latest version, Mac only)*
- Chrome (*latest version*)
- Firefox (*latest version*)

The portal user experience can be customized through personal settings. These settings can be accessed by clicking **Settings** (*the gears icon*) at the top right of the portal toolbar. Once clicked, the following settings are displayed (*at the time of writing, GUI changes may occur over time*):

- **Directories + subscriptions**
- **Appearance**
- **Language + region**
- **My information**
- **Signing out + notifications**

In addition to the preceding settings, **customized dashboards** can be created.

Finally, there are other ways to interact with Azure services and resources; if you want to create and manage more complex resources or perform automation tasks, you can use a **Command-Line Interface** (**CLI**) such as **PowerShell** or **Bourne Again Shell** (**Bash**).

All tasks carried out in the portal use the underlying **Azure Resource Manager** (**ARM**) **Application Programming Interface** (**API**). The **ARM API** performs all the activities in the background and will **notify** you of the progress and whether any tasks have been completed or failed.

Full deployment and activity logging are provided, which detail all the activities/tasks initiated by users and the Azure platform. This can help troubleshoot issues and be helpful for audit and compliance purposes.

Governance can also be put in place through **Azure Policy**, and **access controls** can be put in place through **RBAC**. These access controls can prevent users from adding, changing, or deleting resources, and locks can also be added to resources to prevent accidental deletion.

While the Azure portal is often the starting point for exploring and managing resources, day-to-day Azure operations quickly require consistent, repeatable actions across environments, which is where command-line tools such as Azure PowerShell become essential.

Azure PowerShell

PowerShell is a **cross-platform CLI management tool** for interacting with Azure resources. This tool can be used instead of the GUI of the Azure portal or desktop/mobile apps for creating and managing your Azure environments. **Azure PowerShell** is a **module** installed as part of **Windows PowerShell**. The **AzPowerShell** module contains a set of **cmdlets** (pronounced as "commandlets") for PowerShell that allows you to manage resources from within PowerShell directly at the CLI without the need to access the portal.

PowerShell is a popular management tool where the focus is on Windows systems, and it is designed for complex automation tasks. It can be used interactively, meaning commands can be entered manually by typing them directly into the shell command prompt. Its cross-platform support—from **PowerShell Core 6.x** and **PowerShell 7.x**—means that you can install and use the Az PowerShell module on Windows, Linux, or macOS to interact with your Azure environments.

The **Az module** provides a set of cmdlets for resource management and is based on the .NET Standard library. The Az module for PowerShell replaces **AzureRM** for Azure interaction as it now has the full capabilities of AzureRM, plus more.

> **Note**
>
> The AzureRM PowerShell modules were retired on February 29, 2024.

You can also access the CLI through the Azure portal using **Cloud Shell**, which provides access to the **Az PowerShell module** cmdlets.

PowerShell 7 is the recommended version to use with the Az PowerShell module on all platforms. Exact version requirements may change over time, but PowerShell 7 remains the preferred and supported option for modern Azure management.

To use the Az PowerShell module, you must install it (it is pre-installed in Azure Cloud Shell).

The preferred installation method for the Az module is to use the `Install-Module` cmdlet, and the recommended installation scope is for the **current user** only.

The **syntax** of PowerShell commands uses the **verb-noun** pair format, and the **data** (response or answer to your question) that is returned is called an **object**. These verb-noun pairs work in

a manner in which the **verb** (first part) of the pair refers to the **action that the cmdlet must perform**. The following are the actions:

- `Get`
- `Set`
- `Show`
- `Find`
- `New`
- `Resize`

The noun (second part) of the pair refers to the target resource that the action is performed against—that is, the resource the command operates on.

PowerShell cmdlets follow a consistent verb-noun naming convention, where the verb describes the action being performed and the noun identifies the target resource. Common examples include the following:

- `Connect-AzAccount`: Uses the `Connect` verb to authenticate and sign in to Azure
- `Get-AzResourceGroup`: Uses the `Get` verb to retrieve a list of resource groups
- `New-AzResourceGroup`: Uses the `New` verb to create a new resource group
- `New-AzVM`: Uses the `New` verb to create a new virtual machine

These examples show how different approved PowerShell verbs are used consistently to describe authentication, retrieval, and creation actions against Azure resources.

You can also check the PowerShell version by running the following command from within PowerShell:

```
$PSVersionTable.PSVersion
```

The version number returned will vary depending on when PowerShell was installed or last updated.

PowerShell can also run PowerShell **scripts**. These scripts contain PowerShell cmdlets and code and can only be run (executed) from within PowerShell. Scripts automate repetitive or complex tasks that have many steps and actions to be performed against many different entities.

When using PowerShell scripts, a series of commands can be assembled in the syntax format of the shell being used and then executed together by issuing a single command at the PowerShell prompt.

With an understanding of how Azure resources can be managed using PowerShell, it is important to recognize that Azure provides **multiple command-line tools** to support different operating systems, workflows, and administrator preferences. For this reason, the next section introduces the Azure CLI as an alternative, cross-platform approach to managing Azure resources.

Azure CLI

The **Azure CLI** is a cross-platform CLI management tool for interacting with Azure resources. It is written in Python. The CLI can be used instead of the GUI for creating and managing your Azure environments. The Azure CLI provides cross-platform support, meaning you can install and use it on Windows, Linux, or macOS. It draws parallels to Bash scripting and is a popular management tool choice when the focus is on Linux systems, and is designed for complex and automated tasks.

Much like the Az PowerShell module, commands can be executed using interactive commands. You can use commands directly from the shell prompt or scripts to automate repetitive or complex tasks that have many steps and require actions to be performed against many different entities.

When using Azure CLI scripts, a series of commands is assembled in the syntax format of the shell being used, and the script is then executed by issuing a single command at the shell prompt. This is done within the shell of the OS you have installed the Azure CLI on, for example, `cmd.exe` for Windows or Bash for Linux and Mac.

> **Note**
>
> Cloud Shell is a Microsoft-provided shell environment hosted for you on Ubuntu Linux containers (which Microsoft manages and maintains, and you do not pay for). You can think of this as a managed, cloud-hosted shell environment provided as a service – **Shell Environment as a Service** (**SEaaS**).
>
> The quickest way to start using the Azure CLI is by running it in Azure Cloud Shell instead of a local shell environment hosted on Windows, Linux, or macOS machines.

You can start using the Azure CLI to perform creation and management tasks as you would in the portal or via the Az PowerShell module (Azure PowerShell). So, first sign in by using the following command:

```
az login
```

This command will load an Azure sign-in page; this sign-in method is referred to as an interactive sign-in.

The syntax of the Azure CLI follows a similar (but still different) pattern to PowerShell; it uses the following format:

```
az <command group> <parameters>
```

The following are the same example tasks you looked at in the *Azure PowerShell* section, but this time using the Azure CLI syntax:

- `az login`: This command can be used to log in to Azure
- `az group`: This command lists all resource groups
- `az group`: This command will create a new resource group
- `az vm`: This command will create a new VM

These are just a few Azure CLI examples.

After installing the Azure CLI, you can check the currently installed version by running the following `az` command:

```
az version
```

The following screenshot shows the command output for Windows PowerShell:

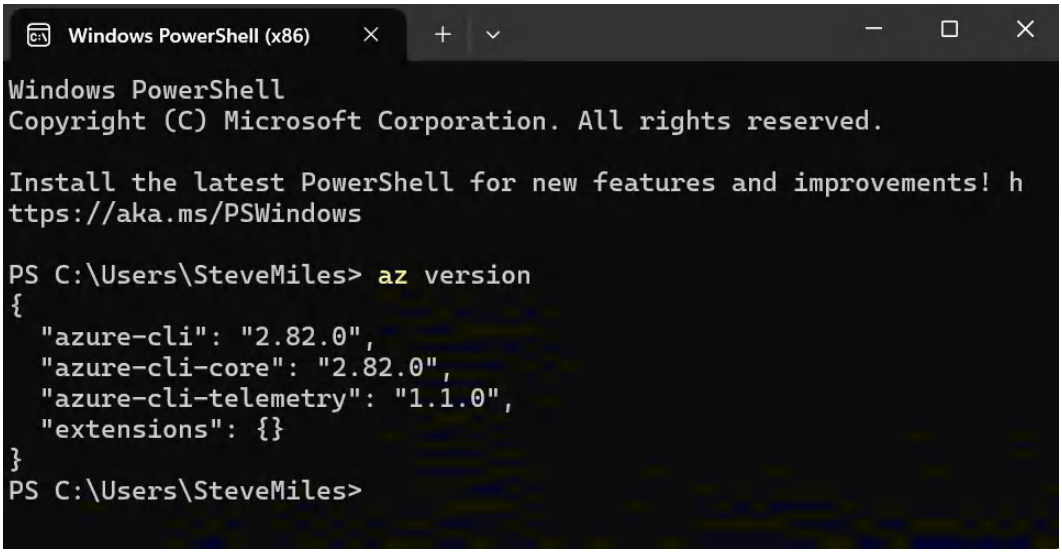


Figure 11.2: The azure-cli version command in PowerShell

This output confirms the installed Azure CLI version along with related components such as the core CLI and telemetry modules.

To illustrate that the Azure CLI behaves consistently across different local shell environments, the same command is shown running in Windows Command Prompt.

The following screenshot shows the command output for Windows Command Prompt:

Figure 11.3: The azure-cli version command in Windows Command Prompt

The version numbers shown are examples and will vary depending on when the Azure CLI was installed or last updated.

Now that you are familiar with the Azure CLI as an Azure management tool, it is time to look at Azure Cloud Shell as a browser-based alternative to a shell environment on a physical or virtual machine.

Azure Cloud Shell

Azure Cloud Shell is a cross-platform, interactive hosted shell and scripting environment that Microsoft provides, hosted on Ubuntu containers. As stated previously, you can think of this as a **Shell Environment-as-a-Service** (**SEaaS**).

Cloud Shell enables a **browser-based CLI**. The benefit of using Cloud Shell is that you do not need to download, install, or update any CLI management tools in a local shell environment on a device or machine. Being a cross-platform management tool, all you need is a browser to run shell commands and the Az PowerShell module.

Imagine that you were to move between devices; you might not have access to the necessary CLI tools, and you may not have the PowerShell modules or updates. This means that your scripts may fail to run or your interactive commands may return errors as they are intended for a different version than the one you have installed locally. However, with Cloud Shell, wherever you have access to a browser, you have access to a CLI; you will have a consistent and regularly updated experience without needing to manage local tool versions. You can also use Cloud Shell from a browser to have a shell environment from anywhere, anytime.

Cloud Shell provides two shell environments so that you can choose the one that best suits your requirements:

- **Bash**: This comes with the Azure CLI installed
- **PowerShell**: This comes with the Azure PowerShell module installed

Cloud Shell can be accessed directly within the Azure portal via the code icon at the top of the portal toolbar, as shown in *Figure 11.4*:

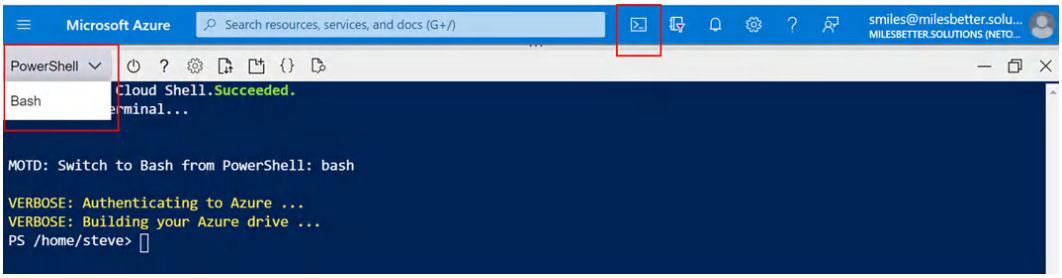


Figure 11.4: Azure Cloud Shell with the PowerShell interface via the Azure portal

You can select your required shell environment from a dropdown, depending on whether you wish to run **Bash** or **PowerShell** commands. You can drag the Cloud Shell pane up and down to resize it and have a split view, as shown in *Figure 11.5*:

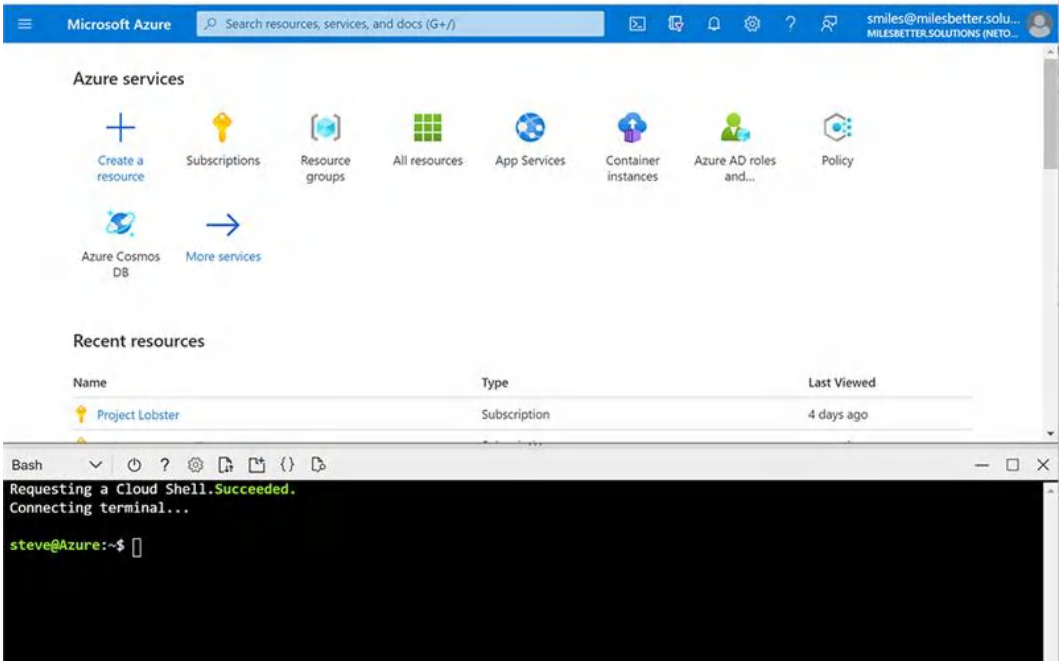


Figure 11.5: Azure Cloud Shell with the Bash interface via the Azure portal

Cloud Shell is also available via a standalone browser-based shell without the need to access it via the Azure portal. You can access this standalone browser-based shell experience at `https://shell.azure.com/`.

As mentioned in the *Azure CLI* section of this chapter, the Azure CLI can be accessed either via the PowerShell or Bash shell environment within Cloud Shell. This is pre-installed, and unlike running it on your device, it is always available wherever you have access to a browser. This feature makes it portable by nature, unlike an installation that needs to be carried out on every device that you may access.

You learned that Cloud Shell, as an Azure management tool, is utilized for writing scripts and automating various tasks. The next section will look at ARM templates, which are integrated within these scripts to automate the deployment of Azure resources.

ARM templates in Azure

ARM templates are an **Infrastructure as Code** (**IAC**) approach to resource deployment. This approach allows the repeatable and reliable deployment of multiple dependent resources into an Azure subscription in an automated and governed manner. ARM templates are **JSON** files—a minimal, readable format for structuring data as an alternative to **Extensible Markup Language** (**XML**)—that define the configuration to be used for resource deployment.

With ARM templates, you define your desired deployment outcome with all resources you want to be created and any properties you wish to be specified in a single request. This is achieved by utilizing a **declarative** syntax (as opposed to an imperative syntax) in the templates.

Figure 11.6 shows this approach of ARM versus non-ARM IAC:

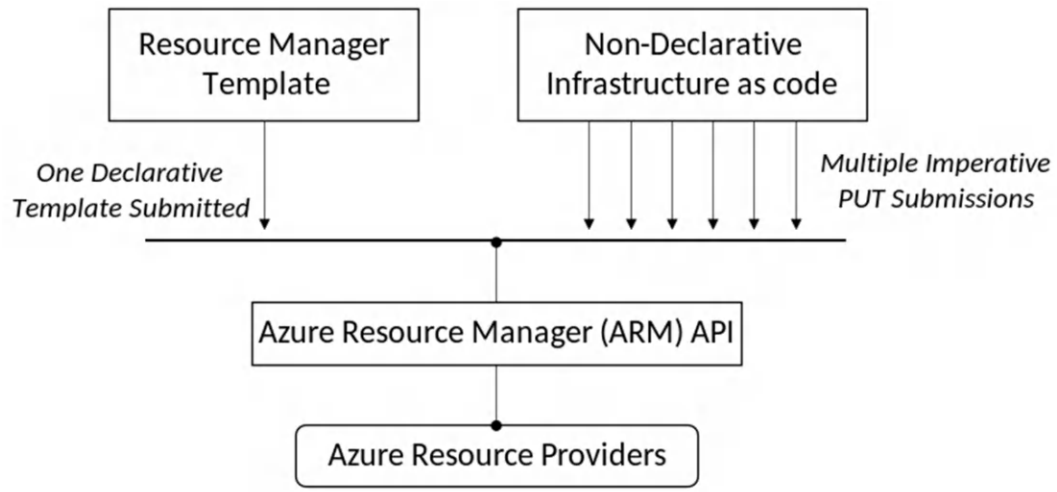


Figure 11.6: ARM template declarative approach

This diagram illustrates the difference between declarative and non-declarative infrastructure approaches in Azure. With ARM templates, a single declarative template is submitted to the **Azure Resource Manager** (**ARM**) API, which determines the required actions and coordinates the appropriate resource providers to reach the desired state. In contrast, non-ARM (imperative) approaches require multiple individual `PUT` requests, where each step must be explicitly defined and executed in sequence by the user or tool.

Instead of issuing multiple commands at each stage of the process to create each resource individually with no orchestration, you must also consider dependencies and the order in which the commands are processed.

We have now learned how Azure Resource Manager provides a consistent deployment and control plane for Azure resources. The next section extends this model beyond native Azure services by introducing Azure Arc, which allows the same ARM-based management, governance, and policy controls to be applied to physical servers and virtual machines running outside Azure.

Azure Arc

Azure Arc is a hybrid management and governance tool that supports physical and virtual machines. These hybrid servers can be on-premises, in provider edge locations, or hosted on other cloud providers' platforms. This is represented in *Figure 11.7*:

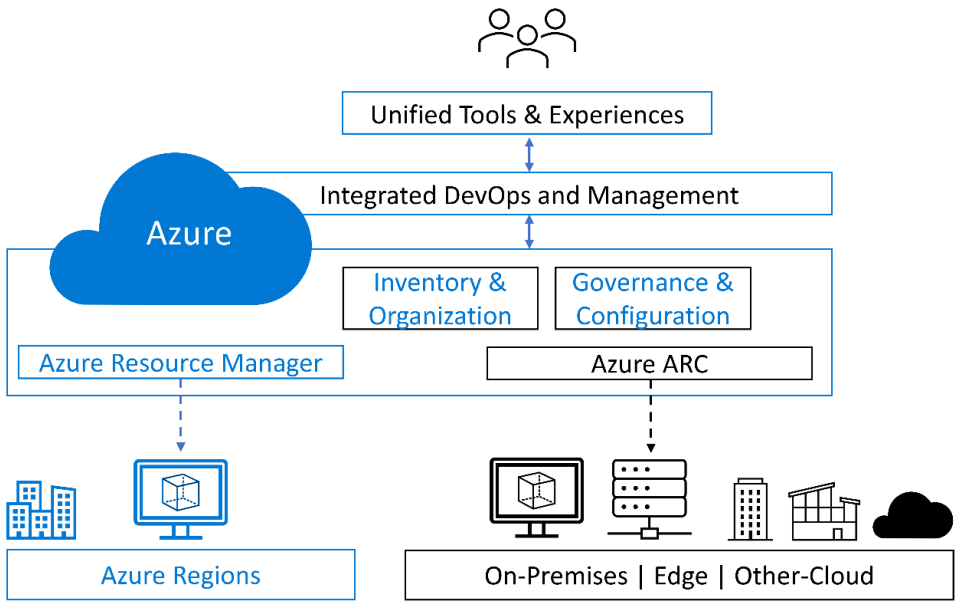


Figure 11.7: Azure Arc for servers

When connected in this way, a hybrid server becomes an **Azure resource** that can be controlled, secured, and managed the same as an Azure native VM.

Each hybrid machine is given an **Azure resource ID**, allowing the machine to be added to a resource group and be managed by ARM. These Azure Arc-managed servers are classed as **Arc-enabled servers**.

To connect a Windows server to Azure Arc, an **Azure Connected Machine agent** is deployed and configured on the server. It should be noted that this does not replace the **Azure Monitor agent** for Windows servers; both agents are required for Arc-enabled servers.

The following hybrid functionality and cloud operations can be performed when a hybrid server is Azure Arc-enabled:

- **Configure**: Azure Automation can be used to speed up routine management tasks. **Change Tracking and Inventory** capabilities are used for configuration changes, software installation, services, and registry updates.
- **Govern**: Azure Policy **guest configurations** can be assigned to perform machine settings audits.
- **Monitor: VM Insights** is used to monitor the OS, components, processes, services, and software applications. Log data can be collected and analyzed through **Log Analytics**.
- **Protect**: **Microsoft Defender for Cloud** provides endpoint protection through **Microsoft Defender for Endpoint**. **Microsoft Sentinel** can provide **Security Information and Event Management** (**SIEM**) and orchestrated and automated responses.

Azure Arc can be enabled manually on servers or automated through the **Windows Admin Center** (**WAC**). This can be used to deploy the **Connected Machine agent** and perform the **Azure registration** in a single place.

This section showed how Azure Arc extends Azure's management, governance, and security model to servers running outside Azure. Having established how resources—both cloud-native and hybrid—are managed consistently through Azure, the next section introduces Microsoft Copilot in Azure, which builds on this foundation by helping administrators interact with these management tools more efficiently using natural language.

Microsoft Copilot in Azure

Microsoft Copilot in Azure is an **AI-powered assistant** integrated into the Azure management experience that helps administrators, operators, and architects interact with Azure resources using natural language. While **Microsoft Copilot in Azure** is not currently assessed in the AZ-900 Azure Fundamentals exam, it represents an important evolution in how Azure environments are managed and operated.

Microsoft Copilot in Azure is not intended to, and does not, replace **ARM**, Azure Policy, RBAC, or other governance and security controls; instead, it is an **AI operations assistant** that operates on top of the existing Azure control plane, using the same permissions, policies, and enforcement mechanisms already in place. Any action suggested or performed through

Microsoft Copilot in Azure is still subject to the same authorization and governance boundaries as actions performed through the Azure portal, Azure CLI, PowerShell, or ARM templates.

From an operational perspective, Microsoft Copilot in Azure acts as the conversational interface across Azure management tools. It can help users understand deployed resources, summarize configurations, explain cost drivers, and surface insights from services such as Azure Advisor, Azure Monitor, and Azure Policy. For example, an administrator might ask Microsoft Copilot in Azure to explain why costs have increased in a subscription, summarize security recommendations for a workload, or describe the configuration of a virtual network and its associated resources.

Microsoft Copilot in Azure is particularly useful in complex environments where multiple services, subscriptions, and governance controls are in place. Rather than navigating multiple blades in the Azure portal or composing queries manually, users can ask questions in natural language and receive contextual explanations grounded in their actual Azure environment. This can improve productivity, reduce operational friction, and help less experienced administrators understand Azure concepts more quickly.

It is important to understand that Microsoft Copilot in Azure is an assistive capability, not an autonomous management system. It does not independently deploy resources, bypass approval processes, or override organizational controls. All deployments, changes, and deletions remain governed by ARM, Azure Policy, resource locks, and RBAC, and require appropriate permissions.

While not required knowledge for the AZ-900 exam, Microsoft Copilot in Azure reflects Microsoft's broader direction toward AI-assisted cloud operations. As Azure environments continue to grow in scale and complexity, tools such as Microsoft Copilot in Azure are intended to help organizations manage resources more efficiently while maintaining strong governance and security foundations.

Summary

This chapter included coverage of the following AZ-900 Azure Fundamentals exam **Skills Measured** area: **Describe Azure management and governance**.

In this chapter, you learned how Azure resources are deployed, managed, and governed using Azure's core management tools and services. You explored the **Azure portal** as the primary graphical interface for managing Azure resources, alongside **Azure PowerShell** and the **Azure CLI** as scriptable, automation-friendly management tools. You also learned how **Azure Cloud Shell** provides a browser-based, preconfigured environment for running PowerShell and CLI commands without local installation.

The chapter introduced **ARM templates** as the foundation for infrastructure as code in Azure, enabling consistent, repeatable, and governed deployments across environments. You also learned about **Azure Arc**, which extends Azure management and governance capabilities to on-premises, multi-cloud, and edge resources.

Finally, you were introduced to **Azure Copilot**, an AI-powered assistant integrated into the Azure management experience. While not required knowledge for the AZ-900 exam, Microsoft Copilot in Azure represents an important evolution in how administrators and operators interact with Azure environments. It provides a conversational interface that helps explain configurations, surface insights, and understand costs and governance, while still operating fully within existing Azure control-plane mechanisms such as ARM, RBAC, Azure Policy, and resource locks.

Together, these tools demonstrate how Azure provides multiple ways to deploy, manage, automate, and govern cloud resources—ranging from graphical interfaces and command-line tools to declarative templates and AI-assisted operations—while maintaining consistent security, compliance, and governance controls.

In the next chapter, you will look at **Azure Monitoring and Tools**.

Exam Readiness Drill – Chapter Review Questions

Apart from mastering key concepts, strong test-taking skills under time pressure are essential for acing your certification exam. That's why developing these abilities early in your learning journey is critical.

Exam readiness drills, using the free online practice resources provided with this book, help you progressively improve your time management and test-taking skills while reinforcing the key concepts you've learned.

HOW TO GET STARTED

- Open the link or scan the QR code at the bottom of this page
- If you have unlocked the practice resources already, log in to your registered account. If you haven't, follow the instructions in *Free Benefits with Your Book* section in this book's *Preface* and come back to this page.
- Once you log in, click the START button to start a quiz
- We recommend attempting a quiz multiple times till you're able to answer most of the questions correctly and well within the time limit.
- You can use the following practice template to help you plan your attempts:

Working On Accuracy		
Attempt	**Target**	**Time Limit**
Attempt 1	40% or more	Till the timer runs out
Attempt 2	60% or more	Till the timer runs out
Attempt 3	75% or more	Till the timer runs out
Working On Timing		
Attempt 4	75% or more	1 minute before time limit
Attempt 5	75% or more	2 minutes before time limit
Attempt 6	75% or more	3 minutes before time limit

The above drill is just an example. Design your drills based on your own goals and make the most out of the online quizzes accompanying this book.

First time accessing the online resources?

You need to unlock the online practice resources to access the link below. Check the *Free Benefits with Your Book* section in the *Preface* for detailed instructions on how to do this. You only need to unlock the resources once.

Open Quiz `https://packt.link/AZ-9003ech11` or scan the QR code

Join the CloudPro Newsletter with 44000+ Subscribers

Want to know what's happening in cloud computing, DevOps, IT administration, networking, and more? Scan the QR code to subscribe to **CloudPro**, our weekly newsletter for 44,000+ tech professionals who want to stay informed and ahead of the curve.

https://packt.link/cloudpro

12

Azure Monitoring Tools

In *Chapter 11*, *Azure Resource Deployment and Management*, you explored how Azure resources are deployed, managed, and controlled using the Azure portal, command-line tools, and Azure Resource Manager. This chapter primarily focuses on the *Describe Azure Management and Governance* module from the *Skills Measured* section of the *AZ-900 Azure Fundamentals* exam. By the end of this chapter, you will be able to confidently answer questions on the following topics:

- Azure Advisor
- Azure Monitor
- Azure Service Health

In addition, this chapter's goal is to take your knowledge beyond the exam by helping you understand how monitoring, health insights, and advisory recommendations support operational decision-making, cost optimization, performance tuning, and proactive issue resolution in real-world Azure environments.

> **Note**
>
> No prior technical experience is required to understand these concepts; however, a basic familiarity with navigating the Azure portal will help you contextualize how these services appear and operate in practice.

Azure Advisor

Azure Advisor is an included, no-cost service that provides advice on optimizing your Azure resources. It provides personalized and actionable best-practice recommendations based on usage analysis and can be accessed directly within the portal. It is recommended to regularly review Azure Advisor recommendations and take action to reduce costs, improve performance, strengthen security posture, and enhance reliability across your Azure workloads.

To view Azure Advisor, as shown in *Figure 12.1*, log in to the Azure portal:

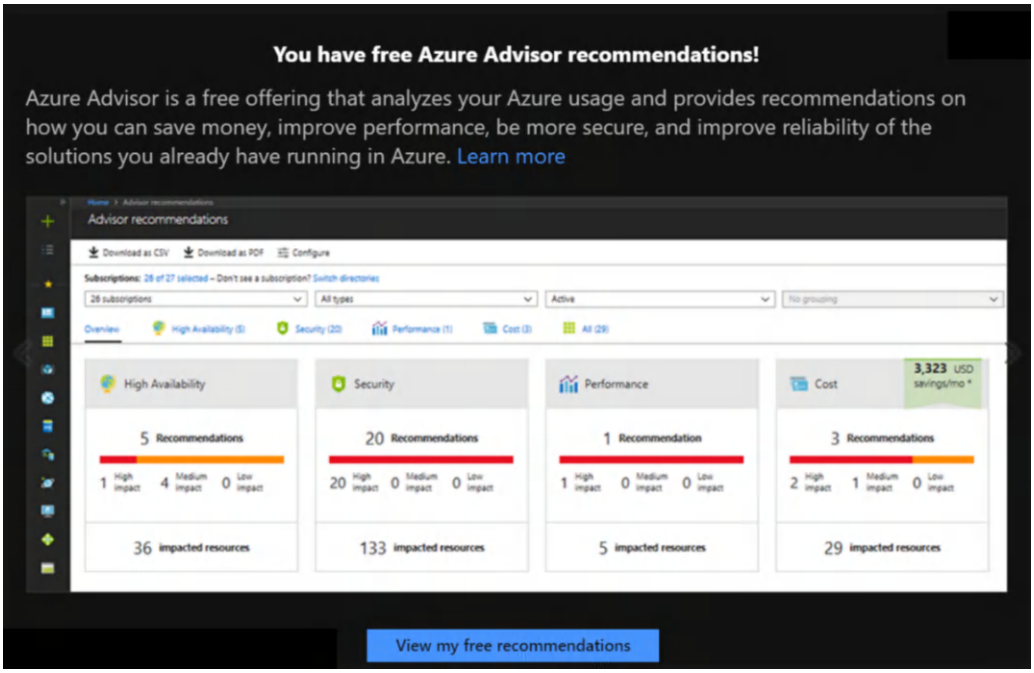


Figure 12.1: Azure Advisor recommendations prompt screen

The screenshot in *Figure 12.1* shows the Azure Advisor dashboard in the Azure portal, where recommendations are categorized by **High Availability**, **Security**, **Performance**, and **Cost**. Each category displays the number of recommendations, their impact level, and the number of affected resources. From this central view, administrators can review suggested actions and assess potential monthly savings or risk reduction before implementing changes.

Purpose and role of Azure Advisor

At a fundamental level, Azure Advisor exists to help organizations continuously improve the way they use Azure after resources have already been deployed. Unlike services that focus on deployment, enforcement, or monitoring, Azure Advisor analyzes the current state of running resources and compares them against Microsoft-recommended best practices.

This positioning is important for both the exam and real-world understanding. Azure Advisor does not prevent misconfiguration, block deployments, or automatically correct issues. Instead, it highlights opportunities for improvement and leaves the final decision to the customer. This advisory-only model allows organizations to balance cost, performance, security, and operational priorities according to their own business requirements.

For the AZ-900 exam, Azure Advisor should be understood as a recommendation engine rather than a monitoring or enforcement tool.

Azure Advisor recommendations are available across the following categories (*Table 12.1*):

Category	**Recommendation**
Cost	Provided for optimizing and reducing Azure subscription spend
Operational Excellence	Provided for process and workflow efficiency, resource manageability, and deployment best practices
Performance	Provided for optimizing the workload's speed and responsiveness to demand
Reliability (formerly High Availability)	Provided to ensure and improve the continuity of your workloads
Security	Provided to protect against vulnerabilities and threats

Table 12.1: Advisor recommendations for each category

Each category aligns to a specific area of cloud optimization and represents a common focus area in both exam questions and real-world cloud operations. Together, these categories

reinforce the idea that Azure Advisor looks at environments holistically rather than focusing on a single technical domain.

Understanding these categories conceptually is important, but in practice, administrators must review, filter, and act on the specific recommendations surfaced within each area.

Viewing and managing recommendations

Figure 12.2 shows the **Advisor | Overview** blade displaying these recommendations in a dashboard view:

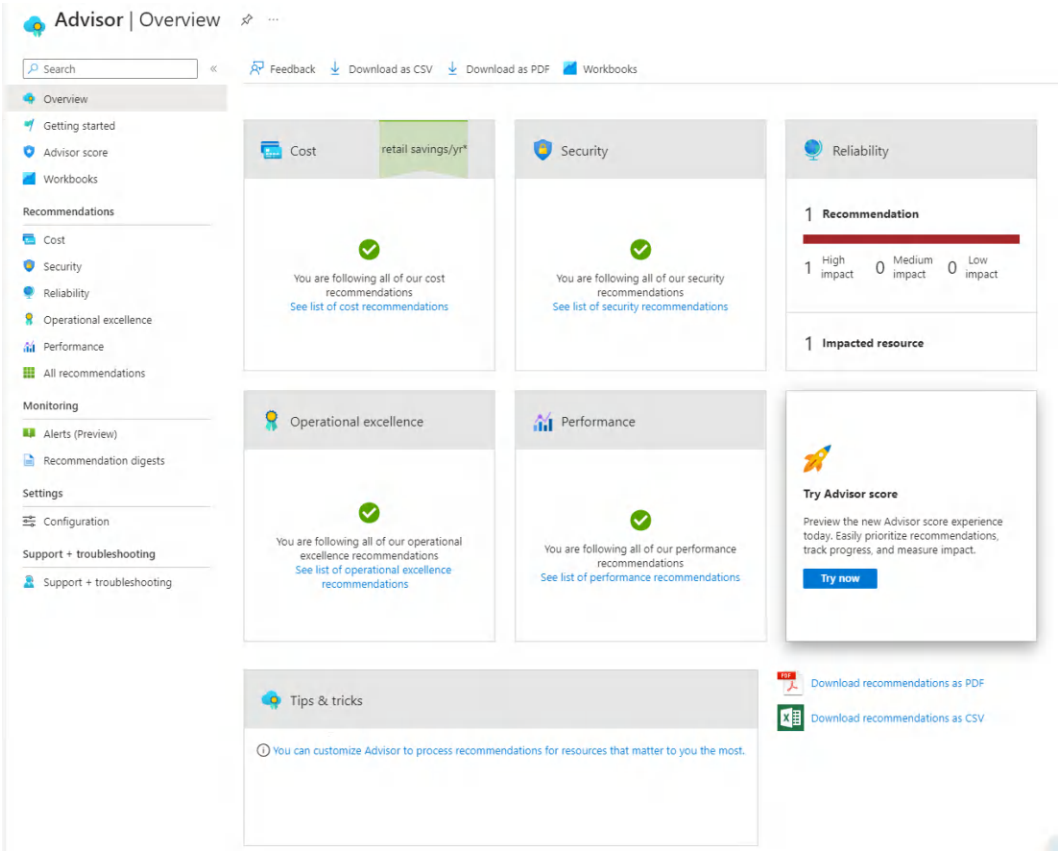


Figure 12.2: Azure Advisor recommendations displayed on the dashboard

From the **Overview** blade, recommendations are grouped by category and severity, allowing administrators to quickly identify which areas of the environment may benefit most from attention. This centralized view helps teams prioritize improvements without needing to inspect individual resources manually.

You can also download the recommendations in PDF and Excel format. A **Tips & tricks** section at the bottom of the **Overview** blade is provided; Microsoft will use this information pane to provide useful hints for you to get the most out of Azure Advisor. These features are particularly useful when recommendations need to be shared with stakeholders outside the Azure portal.

While the **Overview** blade shows the results of Azure Advisor's analysis, it is equally important to understand how those recommendations are generated behind the scenes.

How Azure Advisor generates recommendations

Azure Advisor recommendations are based on the approach of notifying users of aspects that are not implemented rather than things that are implemented. This proactive approach focuses attention on potential gaps and improvement opportunities before they result in performance issues, security risks, or unnecessary costs.

For example, Azure Advisor will generate the following types of recommendations:

- VMs that are not protected by a network security group
- VMs that do not have backup enabled, updates applied, a premium SSD when capable, and so on
- Storage accounts that do not have soft delete enabled, and so on

You can click on each recommendation category to get more information on the recommendations and the actions that can be taken. Each recommendation includes contextual guidance explaining why it was generated and what impact addressing it may have.

Understanding the reasoning behind each recommendation naturally leads to the next step—deciding whether and how to implement it.

Acting on Azure Advisor recommendations

Once security recommendations are implemented, you will see the organization's Secure Score increase. However, there is no requirement for any recommendations to be implemented to be supported by Microsoft. You can decide whether to implement these recommendations or not; they can be dismissed or postponed.

This flexibility reflects real-world operational decision-making. In some scenarios, a recommendation may be technically valid but not appropriate due to cost constraints, workload requirements, or business priorities. Azure Advisor supports this by remaining advisory rather than prescriptive.

Azure Advisor in an exam and operational context

For the AZ-900 exam, Azure Advisor is commonly tested through scenarios that ask which service provides best-practice guidance to optimize existing Azure resources. In these cases, Azure Advisor is the correct choice over services that monitor activity, enforce configuration, or respond to incidents.

From an operational perspective, Azure Advisor is often used as a starting point for optimization discussions, helping organizations identify quick wins and longer-term improvement opportunities across cost, security, reliability, performance, and operational excellence.

Azure Advisor highlights where improvements can be made; Azure Monitor shifts the focus to understanding what is happening across your environment in real time and over time.

To build on this optimization mindset, we now turn to Azure Monitor, which focuses not on recommendations but on continuous visibility and observability across your Azure resources.

Azure Monitor

Azure Monitor is an included service that provides actionable insights into the health, availability, and performance of Azure and on-premises environments by collecting and analyzing logs and metrics (telemetry). It allows you to find and fix problems faster, optimize your workload's performance, and then provide actions to remediate and alert; it provides all these insights from within the portal.

Where Azure Advisor focuses on identifying optimization opportunities and recommended improvements, Azure Monitor concentrates on observing live and historical telemetry to provide visibility into the actual behavior and health of your environment.

At a fundamental level, Azure Monitor acts as the central observability platform for Azure. It enables organizations to understand what is happening across their resources, how those resources are performing, and whether intervention is required.

Purpose and scope of Azure Monitor

Azure Monitor is designed to answer operational questions such as whether resources are healthy, whether performance is meeting expectations, and whether abnormal behavior is occurring. Unlike Azure Advisor, which provides recommendations for improvement, Azure Monitor focuses on the **current and historical state** through telemetry.

Viewing and managing recommendations

To view Azure Monitor, as shown in *Figure 12.3*, log in to the Azure portal:

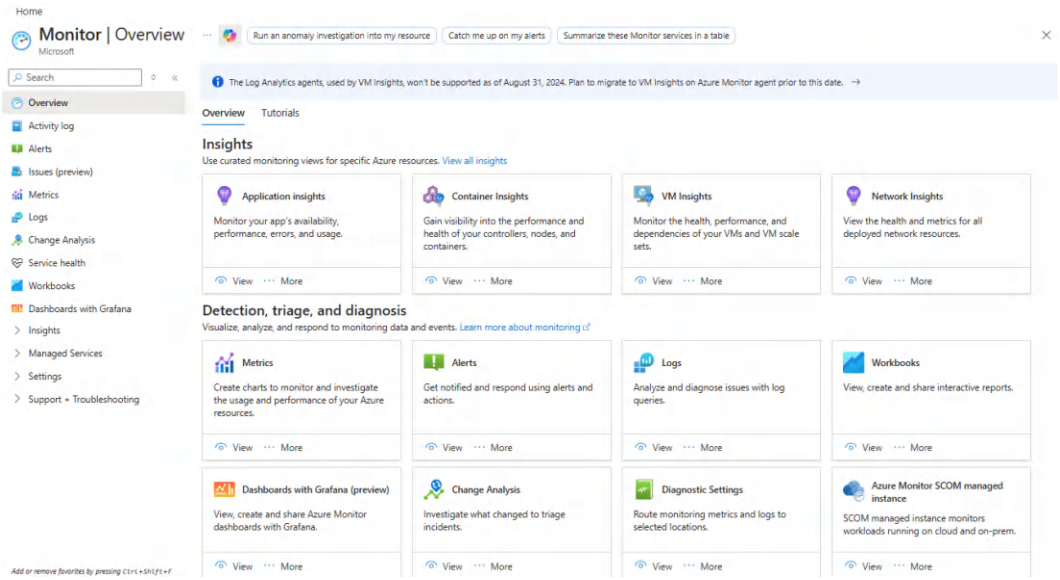


Figure 12.3: Azure Monitor

The **Monitor | Overview** blade, shown in *Figure 12.3*, centralizes monitoring capabilities into a single dashboard. From here, administrators can access insights such as **Application insights**, **VM Insights**, **Container Insights**, and **Network Insights**, as well as core monitoring tools, including **Metrics**, **Logs**, **Alerts**, **Workbooks**, and **Diagnostic Settings**. This unified interface allows teams to track resource health, performance trends, configuration changes, and security-related telemetry across both Azure and hybrid environments.

From an AZ-900 perspective, Azure Monitor should be understood as the *umbrella service* that brings together monitoring data, analysis tools, and alerting capabilities across Azure and connected environments.

How Azure Monitor generates recommendations

Azure Monitor collects resources and platform data from the following data sources:

- PaaS resources such as applications and databases
- IaaS resources, including VMs, containers, virtual desktops, databases, and storage
- On-premises resources via Azure Monitor, using agent-based or Syslog-based data collection for supported appliances

This broad scope allows Azure Monitor to provide a unified monitoring experience across cloud and hybrid environments. For organizations operating mixed workloads, this consistency reduces the need to manage separate monitoring solutions for different platforms.

To understand how Azure Monitor delivers this unified monitoring capability, it is helpful to examine its underlying architecture and the core components that support data collection, storage, and analysis.

Azure Monitor architecture overview

Figure 12.4 visualizes the Azure Monitor service and its three core elements—that is, data sources, data stores, and data functions:

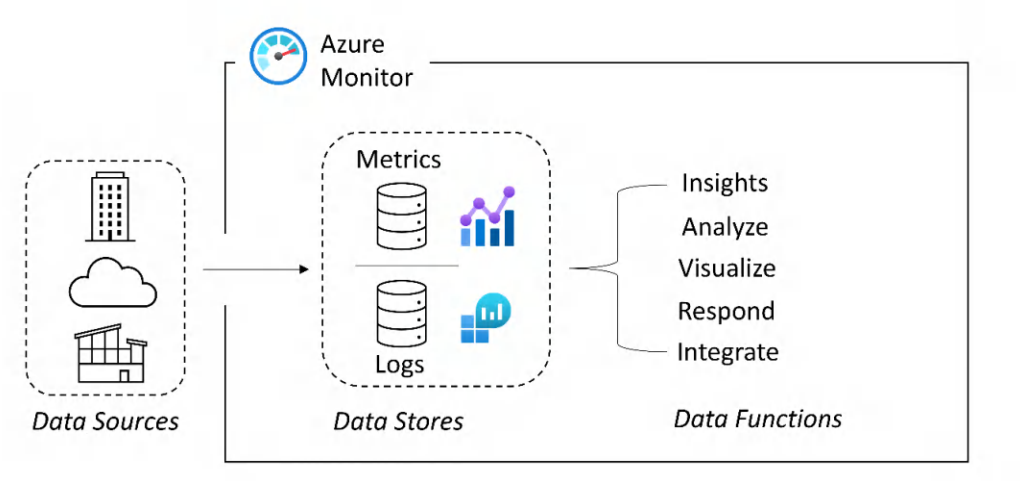


Figure 12.4: Azure Monitor components architecture

Figure 12.4 illustrates the end-to-end telemetry pipeline within Azure Monitor.

On the left, **data sources** represent Azure resources, applications, VMs, containers, and on-premises systems that generate monitoring signals. These signals are then ingested into centralized **data stores**—specifically the metrics store for lightweight, time-series performance data and the logs store for more detailed, queryable event and diagnostic data.

On the right, **data functions** show how this stored telemetry is used: insights are generated and data is analyzed through queries, visualized in dashboards and workbooks, responded to through alerts and automation, and integrated with other tools and services.

Azure Monitor follows a structured telemetry flow. Monitoring data is generated by data sources, stored centrally, and then consumed by analysis and visualization tools. This architecture allows Azure Monitor to support near-real-time insights while also enabling historical analysis.

Logs and metrics as fundamental data types

The Azure Monitor service uses two fundamental data types: **logs** and **metrics**.

Monitoring data sources populate these data stores, and data functions consume them, with each data type serving a distinct purpose.

Logs

Logs record the activities of a data source and represent actions taken against resources. This includes events and operations that occur within the environment.

Examples of log data include the following:

- Capturing a login
- A create, delete, or update action
- Listing storage account keys
- Whether a policy was evaluated
- Whether an automation job started

Log data is presented in a structured, column-based format and is analyzed using Log Analytics, as shown in *Figure 12.5*.

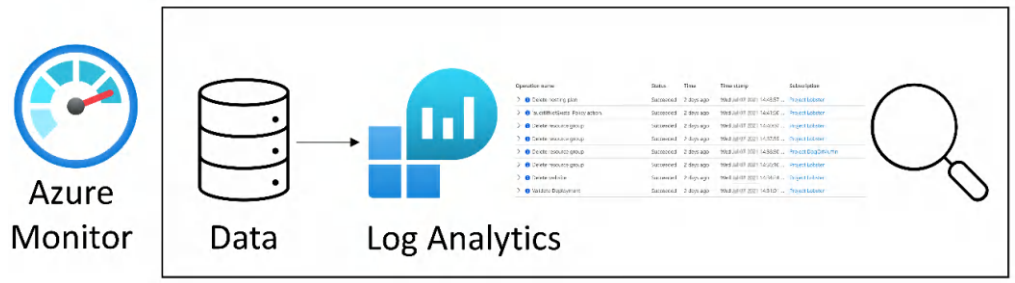


Figure 12.5: Azure Monitor Log Analytics showing ingested log data

Logs are particularly valuable for troubleshooting, auditing, and forensic analysis because they provide detailed contextual information about what happened and when it happened.

Metrics

Metrics record performance and consumption data and represent numerical measurements collected at regular intervals. They are optimized for near-real-time analysis.

Metrics can capture information such as the following:

- CPU utilization
- Memory usage
- Network bandwidth
- Disk throughput

Metrics data is presented visually using charts and graphs through service metrics, as shown in *Figure 12.6*.

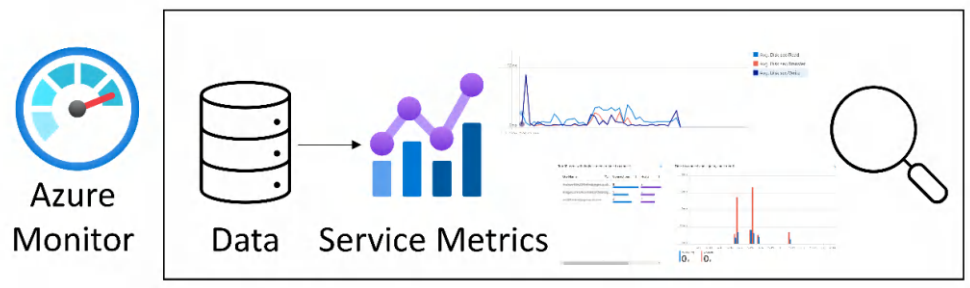


Figure 12.6: Azure Monitor service metrics showing ingested metrics data

Metrics are commonly used to assess performance trends, detect anomalies, and trigger alerts based on thresholds.

Understanding the difference between logs and metrics is a key exam concept. Logs answer detailed *what happened* questions, while metrics answer high-level *how is it performing* questions.

Alerting and notifications in Azure Monitor

With Azure Monitor, you can create alerts to notify individuals, such as an administrator or resource owner, when certain conditions are triggered. This could occur when a resource exceeds a defined threshold, when a resource is stopped, or when a configuration change occurs, such as the deletion of a storage account.

Alerts enable proactive response to issues rather than reactive troubleshooting. They are a key component of operational readiness, ensuring that teams are informed when attention is required.

There are cost implications with using Azure Monitor, especially Log Analytics, depending on the volume of data ingested, the data sources used, and the retention period configured. While cost management is not deeply tested at the AZ-900 level, it reinforces the principle that monitoring design should balance visibility with cost efficiency.

While Azure Monitor provides broad infrastructure and platform visibility, many organizations also require deeper insight into how their applications themselves behave and perform.

Application Insights within Azure Monitor

Application Insights is a feature of Azure Monitor designed specifically for monitoring application performance and health. It can be used for applications hosted in Azure, in other cloud platforms, or on-premises.

Applications can be monitored by installing a **software development kit** (**SDK**) or the Application Insights agent, which supports .NET, Node.js, Python, and Java. This enables application-level telemetry to be collected without requiring extensive custom instrumentation.

Once Application Insights is configured, it can collect a wide range of application telemetry, including the following:

- CPU, memory, and network usage through Windows and Linux performance counters
- Session and user counts
- Browser load performance and page views
- Response times, failure rates, request rates, and dependency calls
- Web page telemetry captured through AJAX

In addition, during periods of low activity, Application Insights can perform synthetic application requests to help validate availability and responsiveness.

Figure 12.7 shows the Application Insights dashboard:

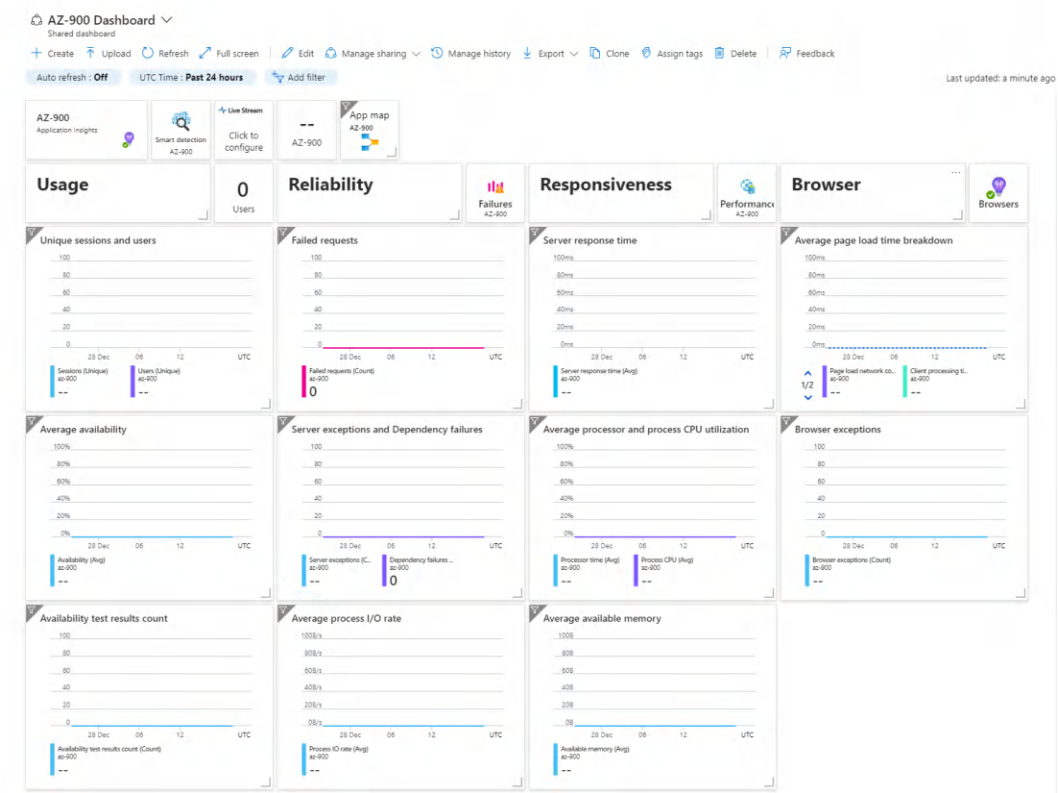


Figure 12.7: Application Insights dashboard showing application performance and health at a glance

The Application Insights dashboard provides a consolidated view of key application telemetry. From this interface, teams can track usage metrics such as unique users and sessions, reliability indicators including failed requests and server exceptions, responsiveness metrics such as server response time and page load duration, and browser-level insights that highlight client-side performance and errors. Additional metrics such as availability test results, dependency failures, processor and memory utilization, and I/O activity allow both developers and operations teams to quickly assess application health, detect anomalies, and identify performance bottlenecks.

Application Insights extends Azure Monitor beyond infrastructure visibility, enabling teams to understand how applications behave from both a system and user perspective.

We looked at Azure Monitor in this section and learned about the three core elements of the service—data sources, data stores, and data functions—and the fundamental data types of

logs and metrics. We then looked at Application Insights as a specialized monitoring capability for applications.

For the AZ-900 exam, Azure Monitor is typically tested as the service responsible for collecting telemetry, visualizing performance, and generating alerts. It should be distinguished from Azure Advisor, which provides recommendations, and Azure Service Health, which focuses on Microsoft-initiated service issues.

Azure Monitor is better when used alongside Azure Service Health. Azure Monitor tells you what is happening inside your environment, while Azure Service Health explains platform-level events that may be affecting your resources.

Azure Monitor focuses on conditions within your environment, while Azure Service Health provides context for platform-level events initiated by Microsoft.

Azure Service Health

Azure Service Health is also an included, no-cost service that provides a personalized view of the health of all your Azure resources. It provides guidance and notifications, such as planned maintenance and other advisories on the resource health that are specific to you.

At a high level, Azure Service Health exists to answer a different class of question than Azure Monitor or Azure Advisor; instead of focusing on how your resources are configured or performing, Azure Service Health focuses on *events initiated by Microsoft* that may affect your services.

Purpose and positioning of Azure Service Health

Azure Service Health is designed to keep customers informed about Azure platform events that have a direct impact on their resources. This includes incidents, planned maintenance, and service advisories that are relevant to the regions and services in use within a subscription.

For AZ-900, it is important to understand that Azure Service Health is not a general monitoring tool and does not collect telemetry from your workloads. Instead, it provides **contextual awareness** of platform-level issues so that customers can respond appropriately. To see how this platform-level awareness fits into the broader monitoring ecosystem, it is important to understand how Azure Service Health integrates with Azure Monitor.

Azure Service Health and Azure Monitor integration

Insights such as **service incidents**, planned maintenance notifications, health advisories, and post-incident **root cause analyses** (**RCAs**) are provided directly within the portal as a subset of the Azure Monitor service.

This integration allows Azure Service Health to complement Azure Monitor, which may show symptoms such as degraded performance or unavailable resources, while Azure Service Health explains whether those symptoms are caused by an underlying Azure service issue.

From an operational perspective, this reduces diagnostic effort by clarifying whether a problem originates within the customer environment or within the Azure platform itself.

Types of information provided by Azure Service Health

Azure Service Health communicates several types of platform-related events, including incidents, planned maintenance, and service advisories. These notifications are filtered automatically so that only events relevant to your subscriptions and deployed resources are shown.

This personalized filtering is a key distinction. Unlike global status dashboards, Azure Service Health focuses specifically on what affects you, helping teams prioritize response efforts and avoid unnecessary investigation into unrelated issues.

Azure Service Health versus the Azure status page

The difference between Azure Service Health and the **Azure status page** is that the status page is a public-facing website that requires no login.

It has a global view of all the services across all regions and is useful to get a quicker and bigger picture of incidents that have a widespread impact.

> **Note**
>
> You can access the Azure status page at `https://azure.status.microsoft/`.

The Azure status page provides a high-level overview of Azure service availability worldwide, but it *does not* reflect subscription-specific impact. It is typically used for situation awareness rather than operational decision-making.

Azure Service Health, by contrast, *requires authentication* and presents a tailored view that highlights which services and regions *within your environment* may be affected.

Navigating from the Azure status page to Azure Service Health

Figure 12.8 shows the Azure status page, which links to Azure Service Health to display issues specific to your resources that may be impacted.

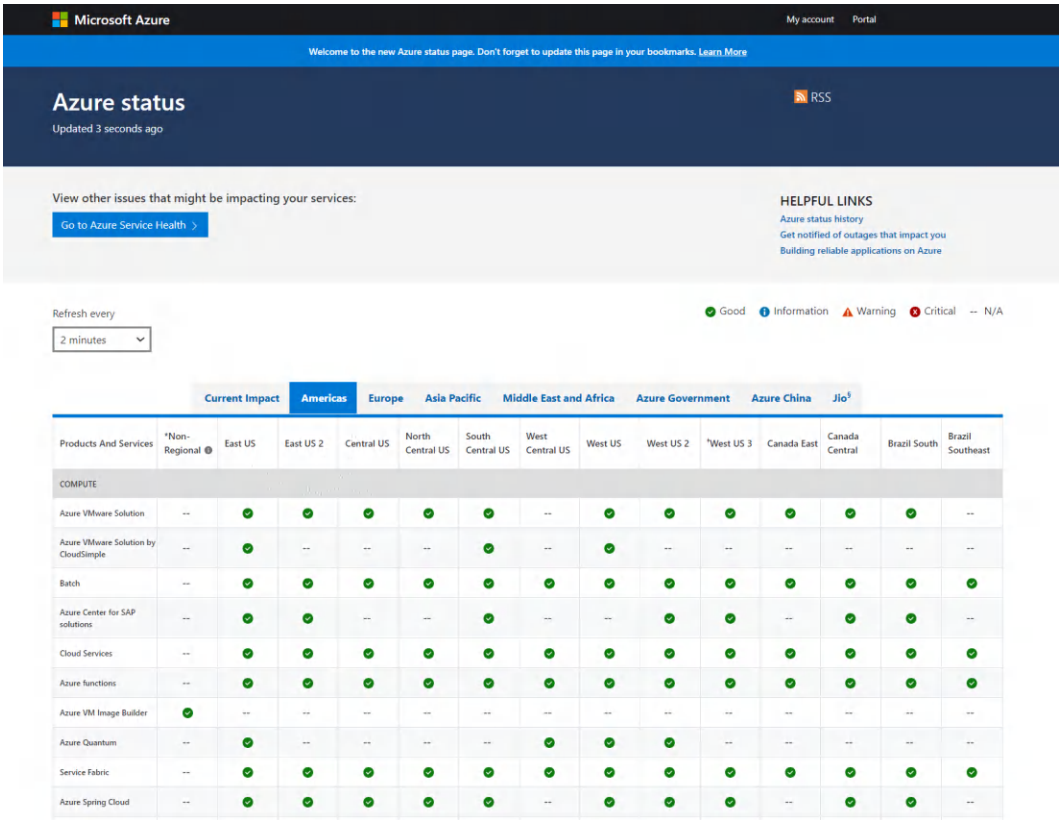


Figure 12.8: Going from the Azure status page to Azure Service Health to view issues

Upon clicking **Go to Azure Service Health**, you will be redirected to sign in to the Azure portal.

Once authenticated, you can view detailed information about how the incident or maintenance event relates to your subscriptions, along with guidance and any available RCA documentation. This linkage allows users to move seamlessly from a global awareness view to a personalized operational view.

In this section, you looked at Azure Service Health, which offers a customized view of the well-being of your Azure resources by providing a health status change so that you can evaluate the situation and take any actions you deem necessary. This section was completed with a look at the individual Azure management tools that can be utilized.

For the AZ-900 exam, Azure Service Health is commonly tested through scenarios involving service outages, planned maintenance, or platform advisories. In these cases, Azure Service Health is the correct choice over Azure Monitor or Azure Advisor because it reflects Microsoft-initiated events rather than customer-generated telemetry or recommendations.

From an operational standpoint, Azure Service Health is most effective when used alongside Azure Monitor. Azure Monitor highlights what is happening inside your environment, while Azure Service Health explains whether Azure itself is contributing to those conditions.

Summary

This chapter included coverage of the AZ-900 Azure Fundamentals exam Skills Measured area *Describe Azure management and governance*.

In this chapter, you learned how to distinguish between optimization, monitoring, and platform-level health services within Azure. You saw how Azure Advisor identifies improvement opportunities, Azure Monitor collects and analyzes telemetry to provide operational visibility, and Azure Service Health communicates platform incidents and maintenance events. You also learned how these services integrate to support informed decision-making across cost, performance, reliability, and security.

With this chapter, you have reviewed the core management and governance concepts required for the AZ-900 Azure Fundamentals exam. You should now be able to clearly differentiate between optimization guidance, operational monitoring, and platform health awareness within Azure and understand how these services work together to support informed cloud management decisions.

As you move into the review questions and final preparation, focus on reinforcing these distinctions, as they are central to exam success and real-world clarity.

After that, refer to the *Appendix*, *Assessing AZ-900 Exam Skills*, for an overview of the skills measured by the certification and the potential topics the exam may cover.

Exam Readiness Drill – Chapter Review Questions

Apart from mastering key concepts, strong test-taking skills under time pressure are essential for acing your certification exam. That's why developing these abilities early in your learning journey is critical.

Exam readiness drills, using the free online practice resources provided with this book, help you progressively improve your time management and test-taking skills while reinforcing the key concepts you've learned.

HOW TO GET STARTED

- Open the link or scan the QR code at the bottom of this page
- If you have unlocked the practice resources already, log in to your registered account. If you haven't, follow the instructions in *Free Benefits with Your Book* section in this book's *Preface* and come back to this page.
- Once you log in, click the START button to start a quiz
- We recommend attempting a quiz multiple times till you're able to answer most of the questions correctly and well within the time limit.
- You can use the following practice template to help you plan your attempts:

Working On Accuracy		
Attempt	**Target**	**Time Limit**
Attempt 1	40% or more	Till the timer runs out
Attempt 2	60% or more	Till the timer runs out
Attempt 3	75% or more	Till the timer runs out
Working On Timing		
Attempt 4	75% or more	1 minute before time limit
Attempt 5	75% or more	2 minutes before time limit
Attempt 6	75% or more	3 minutes before time limit

The above drill is just an example. Design your drills based on your own goals and make the most out of the online quizzes accompanying this book.

First time accessing the online resources?

You need to unlock the online practice resources to access the link below. Check the *Free Benefits with Your Book* section in the *Preface* for detailed instructions on how to do this. You only need to unlock the resources once.

Open Quiz `https://packt.link/AZ-9003ech12` or scan the QR code

13

Appendix: Assessing AZ-900 Exam Skills

You have covered several topics throughout this book and learned the skills to prepare for the Microsoft certification exam *AZ-900: Microsoft Azure Fundamentals*.

This *Appendix* will provide an overview of the *Skills Measured* by the certification and the potential topics the exam may cover. This may help you track and record your progress as you become competent in each skill area.

Skills measured

The three *Skills Measured* areas of coverage are listed as follows:

1. Describe Cloud Concepts (25–30%)
2. Describe Azure Architecture and Services (35–40%)
3. Describe Azure Management and Governance (30–35%)

Each area is broken down into individual *Skills Measured* topics, as shown in the following table:

Module	Topics	Concepts
1. Describe Cloud Concepts (25–30%)	1. Describe cloud computing	1. Define cloud computing
		2. Describe the shared responsibility model

Module	Topics	Concepts
		3. Define cloud models, including public, private, and hybrid
		4. Identify appropriate use cases for each cloud model
		5. Describe the consumption-based model
		6. Compare cloud pricing models
		7. Describe serverless
	2. Describe the benefits of using cloud services	1. Describe the benefits of high availability and scalability in the cloud
		2. Describe the benefits of reliability and predictability in the cloud
		3. Describe the benefits of security and governance in the cloud
		4. Describe the benefits of manageability in the cloud
	3. Describe cloud service types	1. Describe Infrastructure-as-a-Service (IaaS)
		2. Describe Platform-as-a-Service (PaaS)
		3. Describe Software-as-a-Service (SaaS)

Module	Topics	Concepts
		4. Identify appropriate use cases for each cloud service (IaaS, PaaS, and SaaS)
2. Describe Azure Architecture and Services (35–40%)	1. Describe the core architectural	1. Describe Azure regions, region pairs, and sovereign regions
	components of Azure	2. Describe availability zones
		3. Describe Azure datacenters
		4. Describe Azure resources and resource groups
		5. Describe subscriptions
		6. Describe management groups
		7. Describe the hierarchy of resource groups, subscriptions, and management groups
	2. Describe Azure compute and networking services	1. Compare compute types, including containers, virtual machines,
		and functions
		2. Describe virtual machine options, including Azure Virtual Machines, Azure Virtual Machine Scale Sets, availability sets, and Azure Virtual Desktop

Module	Topics	Concepts
		3. Describe the resources required for virtual machines
		4. Describe application hosting options, including web apps, containers, and virtual machines
		5. Describe virtual networking, including the purpose of Azure virtual networks, Azure virtual subnets, peering, Azure DNS, Azure VPN Gateway, and ExpressRoute
		6. Define public and private endpoints
	3. Describe Azure storage services	1. Compare Azure storage services
		2. Describe storage tiers
		3. Describe redundancy options
		4. Describe storage account options and storage types
		5. Identify options for moving files, including AzCopy, Azure Storage Explorer, and Azure File Sync
		6. Describe migration options, including Azure Migrate and Azure Data Box

Module	Topics	Concepts
	4. Describe Azure identity, access, and security	1. Describe directory services in Azure, including Microsoft Entra ID, and Microsoft Entra Domain Services
		2. Describe authentication methods in Azure, including Single
		Sign-On (SSO), Multi-Factor Authentication (MFA), and passwordless
		3. Describe external identities in Azure
		4. Describe Microsoft Entra Conditional Access
		5. Describe Azure Role-Based Access Control (RBAC)
		6. Describe the concept of Zero Trust
		7. Describe the purpose of the defense-in-depth model
		8. Describe the purpose of Microsoft Defender for Cloud
3. Describe Azure Management and Governance (30–35%)	1. Describe cost management in Azure	1. Describe factors that can affect costs in Azure
		2. Explore the pricing calculator

Module	Topics	Concepts
		3. Describe cost management capabilities in Azure
		4. Describe the purpose of tags
	2. Describe features and tools in Azure for governance	1. Describe the purpose of Microsoft Purview in Azure
	and compliance	2. Describe the purpose of Azure Policy
		3. Describe the purpose of resource locks
	3. Describe features and tools for managing and deploying	1. Describe the Azure portal
	Azure resources	2. Describe Azure Cloud Shell, including the Azure Command-Line Interface (CLI) and Azure PowerShell
		3. Describe the purpose of Azure Arc
		4. Describe Infrastructure as Code (IaC)
		5. Describe Azure Resource Manager (ARM) and ARM templates
	4. Describe monitoring tools in Azure	1. Describe the purpose of Azure Advisor
		2. Describe Azure Service Health

Module	Topics	Concepts
		3. Describe Azure Monitor, including Log Analytics, Azure Monitor alerts, and Application Insights

Table 13.1: AZ-900 modules, topics, and their concepts

You can find information about these *Skills Measured* areas in Microsoft's online study guide for *AZ-900 Microsoft Azure Fundamentals* exam. The official study guide also includes any information regarding updates or changes to the exam.

Tip

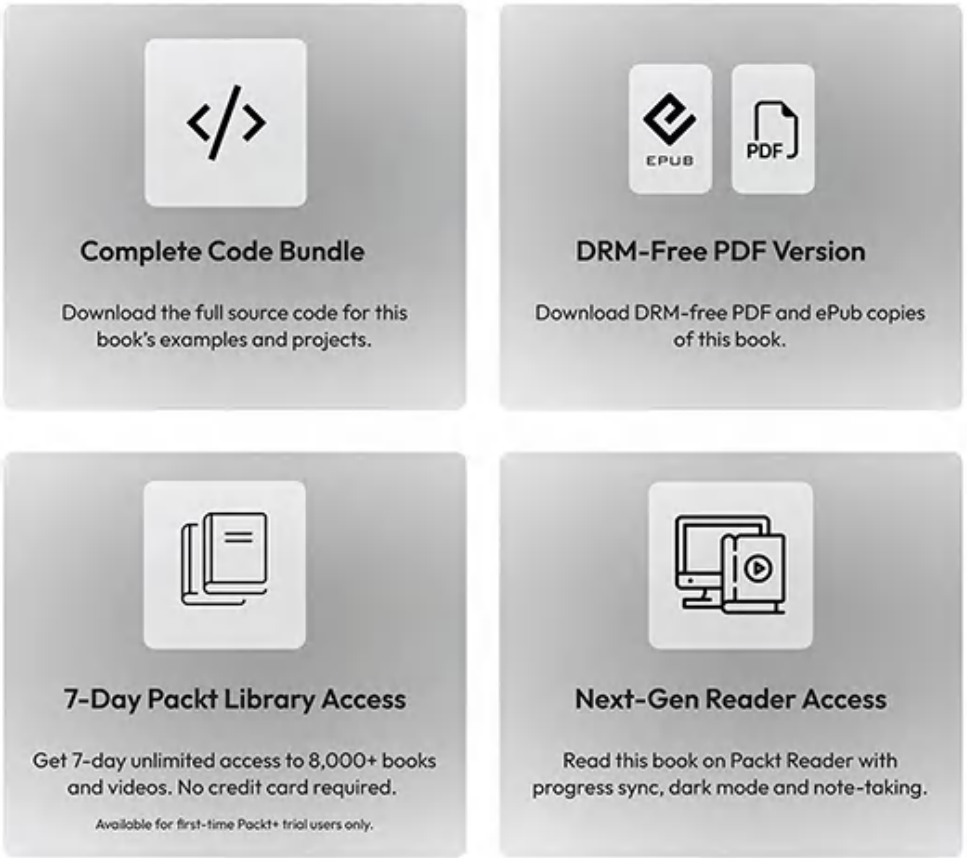
You can find Microsoft's online study guide at `https://learn.microsoft.com/en-us/credentials/certifications/resources/study-guides/az-900`.

You are now closer than ever to becoming **Microsoft Certified for Azure Fundamentals and Beyond**. Keep up the good spirit, continue with passion, and carry on with the next step toward honing your skills and achieving your career goals.

14

Unlock Your Exclusive Benefits

Your copy of this book includes the following exclusive benefits:



Follow the guide below to unlock them. The process takes only a few minutes and needs to be completed once.

Unlock this Book's Free Benefits in 3 Easy Steps

Step 1

Keep your purchase invoice ready for *Step 3*. If you have a physical copy, scan it using your phone and save it as a PDF, JPG, or PNG.

For more help on finding your invoice, visit `https://www.packtpub.com/en-us/unlock?step=1`.

> **Note**
>
> **Note**: If you bought this book directly from Packt, no invoice is required. After *Step 2*, you can access your exclusive content right away.

Step 2

Scan the QR code or go to `packtpub.com/unlock`.

On the page that opens (similar to *Figure 14.1* on desktop), search for this book by name and select the correct edition.

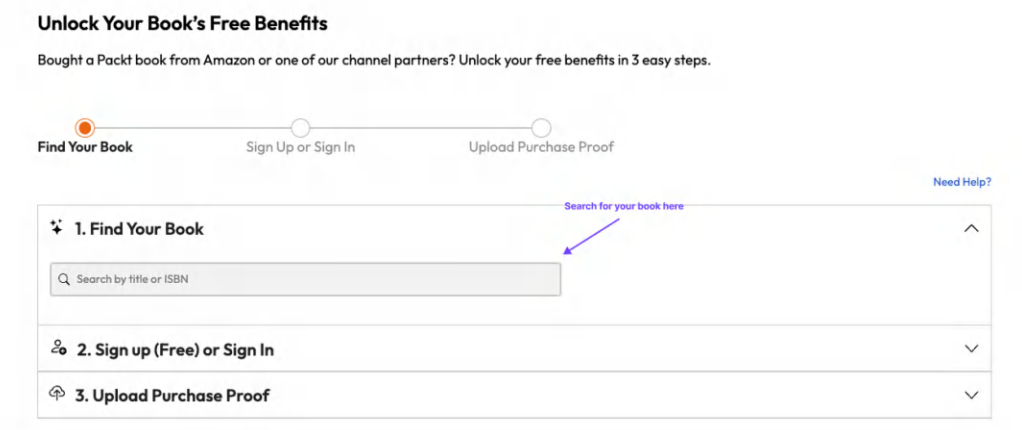


Figure 14.1: Packt unlock landing page on desktop

Step 3

After selecting your book, sign in to your Packt account or create one for free. Then upload your invoice (PDF, PNG, or JPG, up to 10 MB). Follow the on-screen instructions to finish the process.

Need Help

If you get stuck and need help, visit `https://www.packtpub.com/unlock-benefits/help` for a detailed FAQ on how to find your invoices and more. This QR code will take you to the help page.

Note

Note: If you are still facing issues, reach out to `customercare@packt.com`.

packtpub.com

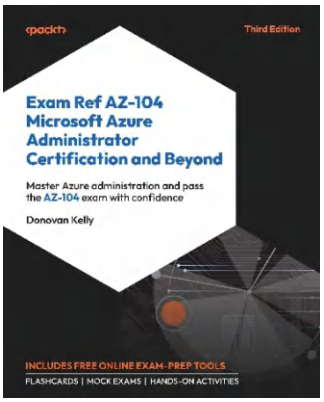
Subscribe to our online digital library for full access to over 7,000 books and videos, as well as industry leading tools to help you plan your personal development and advance your career. For more information, please visit our website.

Why subscribe?

- Spend less time learning and more time coding with practical eBooks and Videos from over 4,000 industry professionals
- Improve your learning with Skill Plans built especially for you
- Get a free eBook or video every month
- Fully searchable for easy access to vital information
- Copy and paste, print, and bookmark content

At packtpub.com, you can also read a collection of free technical articles, sign up for a range of free newsletters, and receive exclusive discounts and offers on Packt books and eBooks.

Other Books You May Enjoy

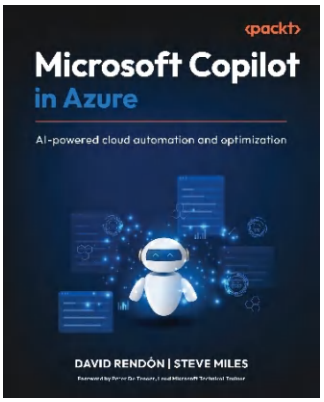


Exam Ref AZ-104 Microsoft Azure Administrator Certification and Beyond

Donovan Kelly

ISBN: 978-1-80512-285-2

- Manage Azure AD users, groups, and RBAC
- Handle subscription management and governance implementation
- Customize and deploy Azure Resource Manager templates
- Configure containers and Azure app services
- Manage and secure virtual networks comprehensively
- Utilize Azure Monitor for resource monitoring
- Implement robust backup and recovery solutions



Microsoft Copilot in Azure

David Rendón, Steve Miles

ISBN: 978-1-83620-024-6

- Set up and configure Microsoft Copilot in the Azure Portal
- Deploy and manage Azure Kubernetes Service (AKS) and App Services using Copilot
- Integrate Copilot with Azure Functions, Blob Storage, and Azure SQL
- Secure resources using Microsoft Defender for Cloud and compliance policies
- Monitor cloud workloads and troubleshoot issues using Copilot diagnostics
- Leverage AI-driven cost optimization and performance recommendations
- Strengthen cloud security posture using AI-enhanced threat detection and remediation
- Master prompt engineering techniques to get the most accurate and effective results from Copilot

Packt is searching for authors like you

If you're interested in becoming an author for Packt, please visit `authors.packt.com` and apply today. We have worked with thousands of developers and tech professionals, just like you, to help them share their insight with the global tech community. You can make a general application, apply for a specific hot topic that we are recruiting an author for, or submit your own idea.

Share your thoughts

Now you've finished *Microsoft Azure Fundamentals Certification and Beyond*, we'd love to hear your thoughts! Scan the QR code below to go straight to the Amazon review page for this book and share your feedback or leave a review on the site that you purchased it from.

https://packt.link/r/180670241X

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

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

T

U

V

W

Z