

Visual Guide to Computer Systems

Azure A Visual Guide

Sample Book — Chapters 1 to 2

Essential Cloud Foundations for New Systems Engineers

Reiner Braun

Introduction

This book is designed for readers who are taking their first steps with Azure, including systems engineers who want to build the right foundation from the start.

Rather than asking you to memorize more than you need, the book focuses on the essential concepts that will support your future study and day-to-day work. More advanced topics are included only when they are likely to become useful to many readers, and even then they are introduced only far enough to explain their purpose, context, and overall role.

Processes, relationships, and structures that can be difficult to picture from text alone are supported by diagrams. By moving between explanations, visuals, and examples, you can build a clearer and more practical understanding of the subject.

Product names, service names, logos, icons, and other identifying marks mentioned in this book are trademarks or registered trademarks of their respective owners. This book is not an official publication, certified training resource, or endorsed learning material of any company or organization.

How to Use This Book

1. **Read the chapters in order.** The sequence builds the required background step by step, beginning with Azure's operating model and then connecting identity, networking, compute, data, security, governance, operations, recovery, automation, migration, and production design.
2. **Review “What You’ll Be Able to Do” before each chapter.** These outcomes identify the concepts and practical abilities that should be clear by the time you finish the chapter.
3. **Use diagrams to understand relationships, processes, and structures.** A diagram supports the surrounding explanation; it does not replace the definitions, assumptions, or operational context in the text.
4. **Try the commands, code, and hands-on examples where practical.** Use an approved non-production environment, confirm prerequisites and active scope, and compare the observed result with the expected result described in the guide.
5. **Use each Chapter Summary and the Review Questions to check retention.** Explain the answers in your own words and return to the relevant section when a relationship or responsibility remains unclear.
6. **Treat advanced-topic overviews as signposts.** You do not need to memorize them at this stage. Remember their purpose and the conditions that indicate when deeper study or specialist support may be required.

Table of Contents

Introduction

How to Use This Book

Part I: Build the Azure Mental Model

Chapter 1: Azure in One Picture

- What You'll Be Able to Do
- Cloud computing is a delivery model
- The service model determines who operates each layer
- Four words that prevent confusion
- Control plane and data plane
- What systems engineers still operate
- A resource has a lifecycle
- Practical decision: virtual machine or managed platform
- Misconceptions that cause operational errors
- Chapter Summary
- Review Questions

Chapter 2: Regions, Availability Zones, and Reliability Boundaries

- What You'll Be Able to Do
- From geography to datacenter
- Regional, zonal, zone-redundant, and global resources
- Four protections that are not interchangeable
- The reliability vocabulary
- Three common design levels
- Shared dependencies can cancel redundancy
- Choosing a region step by step
- Capacity and quota are reliability concerns
- Applying the concepts to the example
- Reliability mistakes to avoid
- Chapter Summary
- Review Questions

Part I: Build the Azure Mental Model

Chapter 1: Azure in One Picture

The central idea is that **Azure changes the division of operational work, but it does not remove the customer's responsibility for the workload.** Microsoft operates the physical cloud platform and, depending on the service, may also operate the guest operating system, runtime, or database engine. The customer still owns decisions about identities, data, access, configuration, availability, and the application itself.

What You'll Be Able to Do

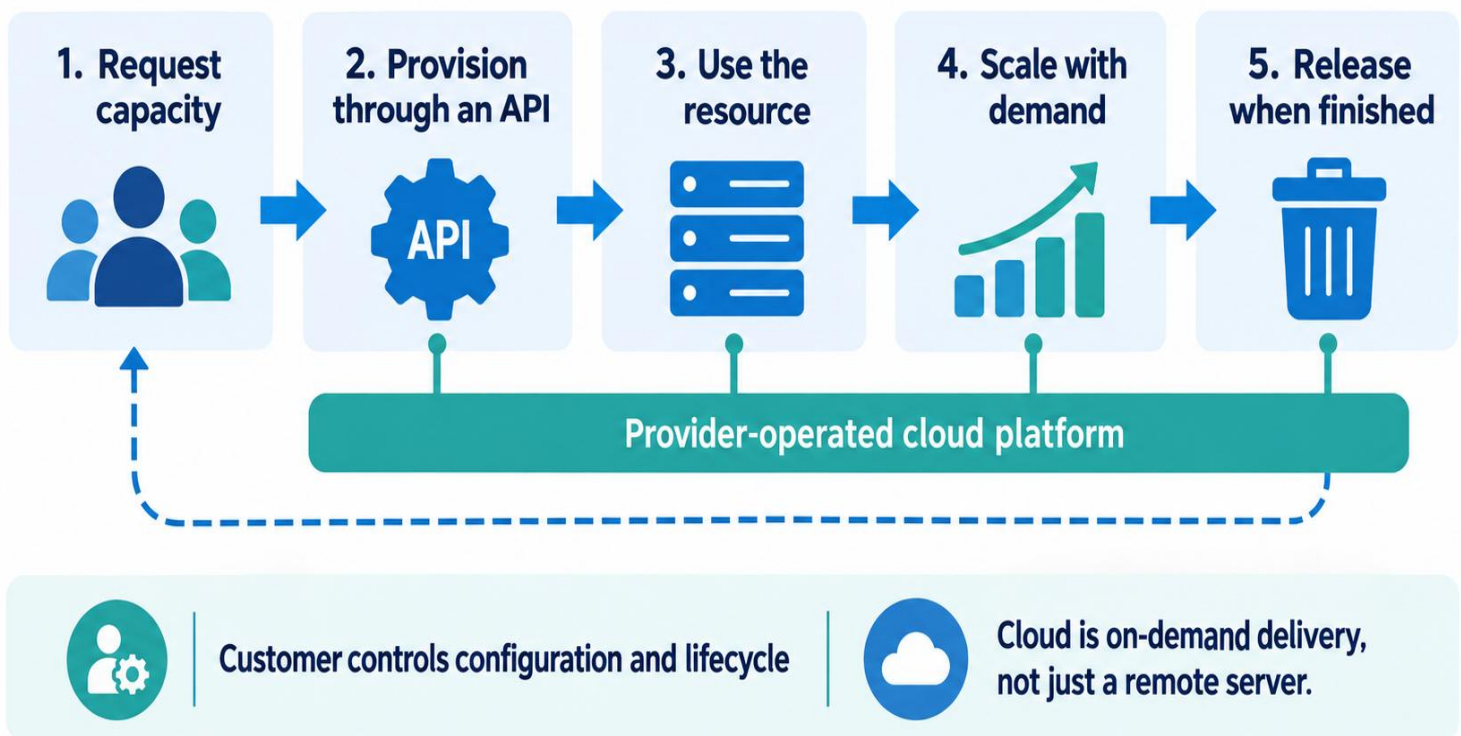
- Explain Azure as a provider-operated platform for customer workloads rather than a catalog of unrelated products.
- Distinguish IaaS, PaaS, SaaS, and serverless models by the operational responsibilities retained by the customer.
- Separate control-plane operations from data-plane access and describe the basic lifecycle of an Azure resource.

Cloud computing is a delivery model

Cloud computing is the on-demand delivery of computing capabilities—such as processing, storage, networking, and managed software—through a provider-operated platform. Instead of purchasing hardware before it is needed, a team can request resources through an API and release them when they are no longer required.

Chapter 1: Azure in One Picture

1-1 Cloud Delivery Lifecycle



001

Figure 1-1: Cloud delivery lifecycle

Three properties matter to a systems engineer:

- Resources can be provisioned through software rather than physical installation.
- Capacity can be increased or reduced without replacing customer-owned hardware.
- Responsibility is shared between the provider and the customer.

Azure is a **public cloud**, meaning Microsoft operates a shared global platform and isolates each customer logically. A **private cloud** provides cloud-like automation on infrastructure dedicated to one organization. A **hybrid cloud** combines

cloud services with on-premises or edge systems. **Multicloud** means an organization intentionally uses services from more than one cloud provider.

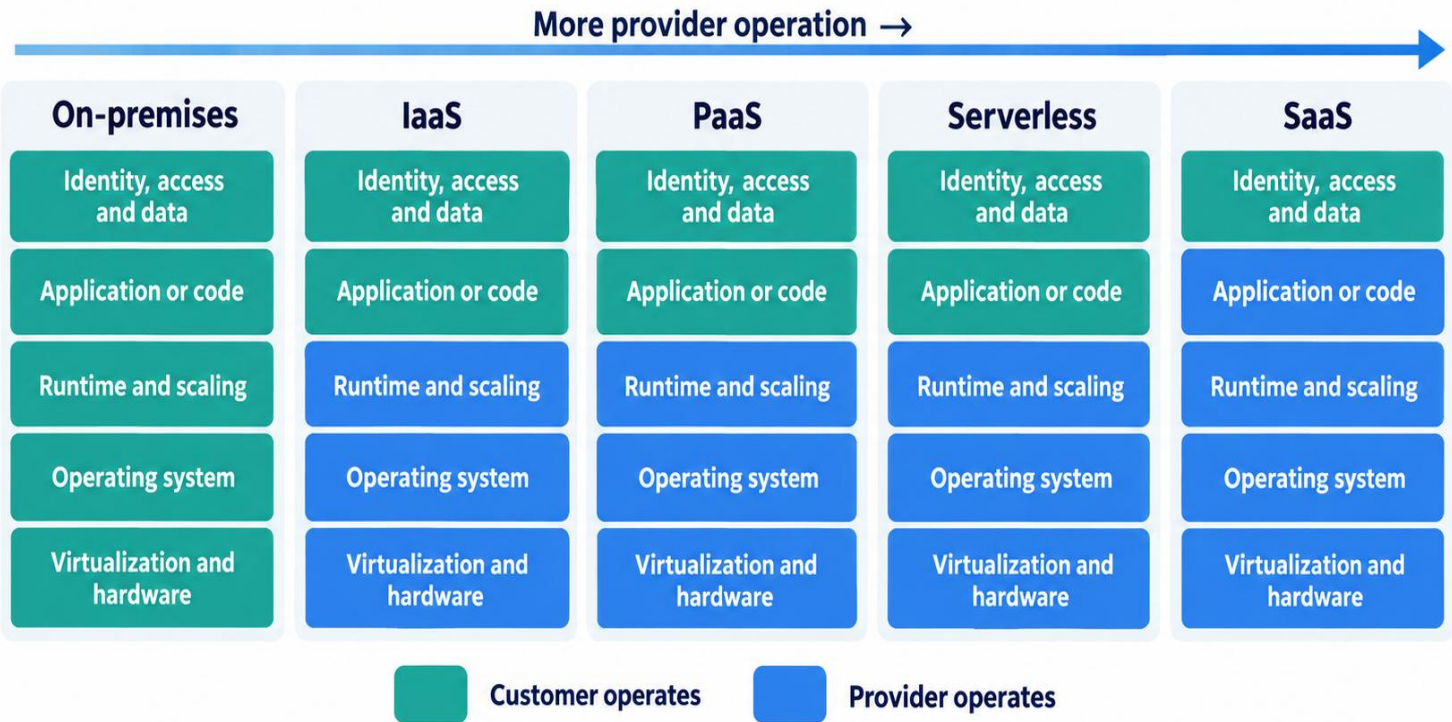
These terms describe operating models, not automatic quality levels. A hybrid design can be well managed or poorly managed; a public-cloud design can be highly available or dependent on one fragile component.

The service model determines who operates each layer

An everyday analogy is useful. Renting an empty apartment gives you more control, but you must furnish and maintain more of it. Staying in a hotel gives you less control, but the operator handles more daily work. Cloud service models follow the same tradeoff.

Chapter 1: Azure in One Picture

1-2 Shared Responsibility Across Service Models



Responsibility moves; customer accountability remains.

002

Figure 1-2: Shared responsibility across service models

Service model	What the customer primarily manages	What the provider primarily manages	Common Azure example
On-premises	Physical facilities, hardware, network, operating system, applications, and data	Nothing in the customer environment	A server room owned by the organization
Infrastructure as a service (IaaS)	Guest operating system, installed software, configuration, identities, and data	Datacenter, physical network, hosts, and virtualization	Azure Virtual Machines
Platform as a service (PaaS)	Application, service configuration, identities, access, and data	Infrastructure, operating system, and much of the runtime	Azure App Service or Azure SQL Database
Software as a service (SaaS)	Users, access policy, data handling, and business configuration	The complete application and its underlying platform	Microsoft 365
Serverless computing	Code or container, triggers, permissions, and data	Capacity allocation, much of scaling, and underlying hosts	Azure Functions

Serverless does not mean that servers do not exist. It means the service hides most server lifecycle work and allocates execution capacity according to the service model.

The shared responsibility boundary moves as the service becomes more managed. With a virtual machine, the customer patches the guest operating system. With App Service, Microsoft patches the platform operating system, but the customer still updates application dependencies and configures secure access. With SaaS, the customer still decides who can sign in and what data may be stored.

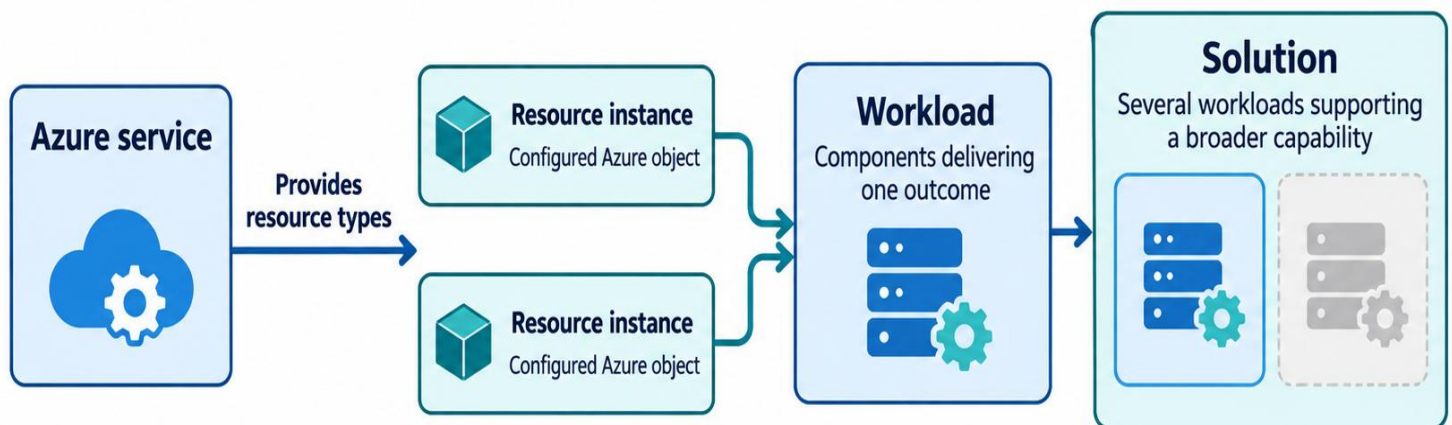
Four words that prevent confusion

Azure discussions often mix several levels of meaning. Keep these definitions separate:

- A **workload** is a collection of components that together deliver a business or technical outcome. The asset-request application is a workload.
- A **service** is an Azure capability offered under a product name, such as Azure Virtual Machines or Azure Storage.
- A **resource** is one configured instance managed in Azure, such as a particular virtual machine or storage account.
- A **platform** is the operating foundation on which workloads run, including management, identity, networking, and service capabilities.

Chapter 1: Azure in One Picture

1-3 Service, Resource, Workload, and Solution Relationship



Example: Azure Storage → stassetprod01 → Asset-request workload



Service ≠ resource ≠ workload ≠ solution

003

Figure 1-3: Service, resource, workload, and solution relationship

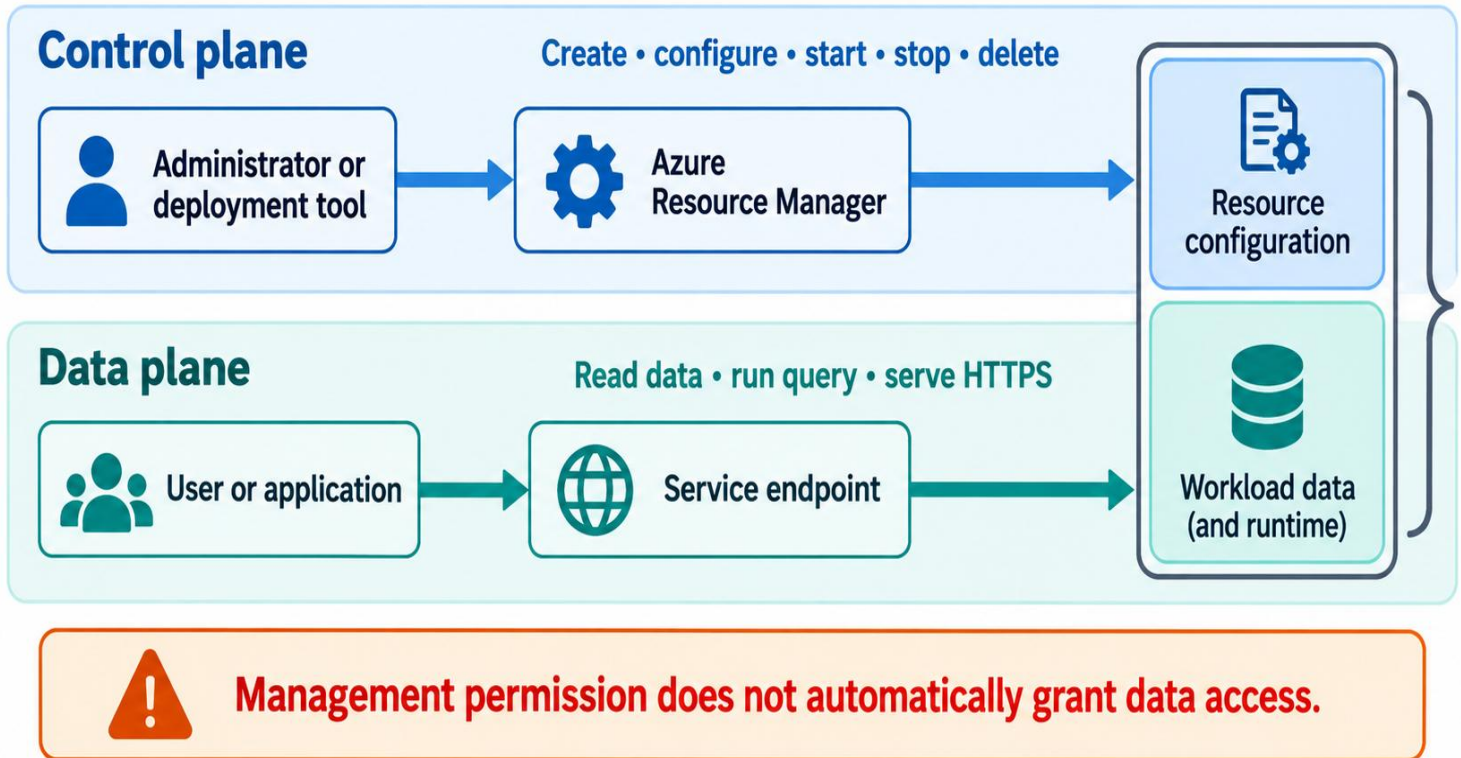
For example, "Azure Storage" is a service. `stassetprod01` might be one storage account resource. The files and objects inside that account support the asset-request workload.

Control plane and data plane

Azure operations occur through two broad paths.

Chapter 1: Azure in One Picture

1-4 Control-Plane and Data-Plane Paths



004

Figure 1-4: Control-plane and data-plane paths

The **control plane** is the management path used to create, configure, start, stop, tag, or delete resources. A request such as “resize this virtual machine” is a control-plane operation, usually processed through Azure Resource Manager.

The **data plane** is the path used to consume the function or data of a resource. Reading a blob, connecting to a database, opening an HTTPS page, or signing in to a virtual machine through SSH are data-plane activities.

Think of a hotel:

- The control plane is the front desk changing a reservation or assigning a room.
- The data plane is the guest entering the room and using it.

This distinction is essential during troubleshooting. A virtual machine can show **Running** in the control plane while its application is unavailable in the data plane. Conversely, a user might be able to read application data but lack permission to change the resource configuration.

What systems engineers still operate

Azure reduces hardware work, but it increases the importance of configuration and coordination. Systems-engineering responsibilities commonly include:

- Confirming the correct tenant, subscription, and resource scope before making a change.
- Managing operating systems where IaaS is used.
- Designing IP addressing, routing, name resolution, and traffic controls.
- Applying least-privilege access for people and workloads.

- Selecting availability, backup, and recovery mechanisms that match business needs.
- Collecting evidence for troubleshooting and change review.
- Managing capacity, quotas, service limits, and lifecycle decisions.

The exact owner differs by organization. A platform team might operate shared networking, an identity team might administer Microsoft Entra ID, and a workload team might own the application. **Responsibility must be explicit even when technical control is shared.**

A resource has a lifecycle

Treat every Azure resource as something that moves through an operational lifecycle:

1. **Plan:** Define the owner, purpose, location, access, connectivity, availability, data handling, and expected lifetime.
2. **Deploy:** Create the resource with reviewed configuration.
3. **Configure:** Apply service, operating-system, application, security, and monitoring settings.
4. **Operate:** Patch, observe, support, back up, and verify the resource.
5. **Change:** Modify capacity or behavior through a controlled process.
6. **Recover:** Restore service or data after failure.
7. **Retire:** Confirm dependencies and retention requirements, then remove access and resources.

Chapter 1: Azure in One Picture 1-6 Azure Resource Lifecycle



005

Figure 1-5: Azure resource lifecycle

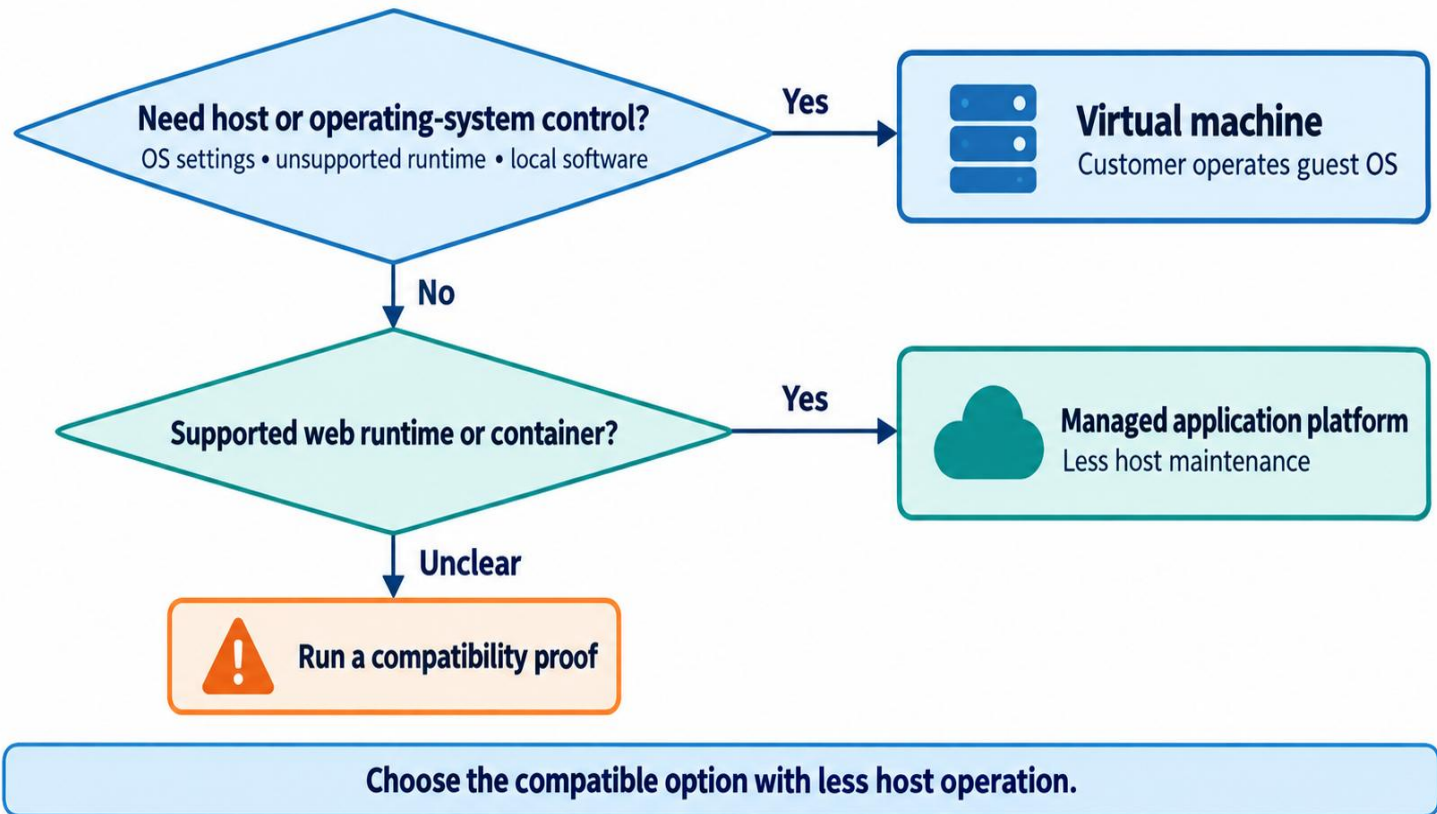
A resource created without an owner or retirement condition often becomes an abandoned cost or security risk. The lifecycle therefore begins before deployment and ends after final verification, not when a portal page says that deletion succeeded.

Practical decision: virtual machine or managed platform

Suppose the asset-request application is a conventional web application. The team can host it on a virtual machine or on App Service.

Chapter 1: Azure in One Picture

1-7 Virtual Machine or Managed Platform Decision



006

Figure 1-6: Virtual machine or managed platform decision

A virtual machine is a reasonable candidate when the application requires a specific operating-system configuration, an unsupported runtime, local software installation, or deep administrative control. The team must then patch the operating system, harden remote access, manage the web server, and plan scaling.

App Service is a reasonable candidate when the application fits a supported web runtime or container model and does not require control of the underlying host. The team gives up some operating-system control but removes routine host maintenance from its workload.

The correct question is not “Which service is more powerful?” It is **“Which service meets the requirement with the least unnecessary operational responsibility?”**

Misconceptions that cause operational errors

“Microsoft operates everything after a workload moves to Azure.” Microsoft operates the cloud platform. The customer remains responsible for the portions of the stack exposed by the selected service and for the workload’s identities, data, and configuration.

“PaaS removes operations.” PaaS removes some host operations. Application patching, access control, network integration, scaling policy, data protection, and service configuration still require ownership.

“The Azure portal is Azure.” The portal is one client of Azure management APIs. The same resources can be managed through Azure CLI, Azure PowerShell, templates, REST APIs, and SDKs. A portal display is therefore a view of the platform, not the platform itself.

Keep this mental model: Azure provides layers of managed capability. Every design decision chooses both a technical service and a responsibility boundary.

Chapter Summary

Azure changes the boundary between provider and customer operations, but it does not remove customer accountability for workload configuration, identity, access, data, and service outcomes. A stable vocabulary for services, resources, workloads, control planes, data planes, and lifecycle stages makes later Azure decisions easier to reason about.

Review Questions

1. Why can a workload on a managed platform still require substantial customer operations?
2. How does a control-plane failure differ from a data-plane failure?
3. What operational factors would make a virtual machine more appropriate than a managed application platform?

Chapter 2: Regions, Availability Zones, and Reliability Boundaries

Boundaries

Placing a workload in Azure does not automatically make it highly available or recoverable. Reliability comes from identifying the failures that matter, placing redundant components across appropriate boundaries, and ensuring that shared dependencies do not defeat the design.

What You'll Be Able to Do

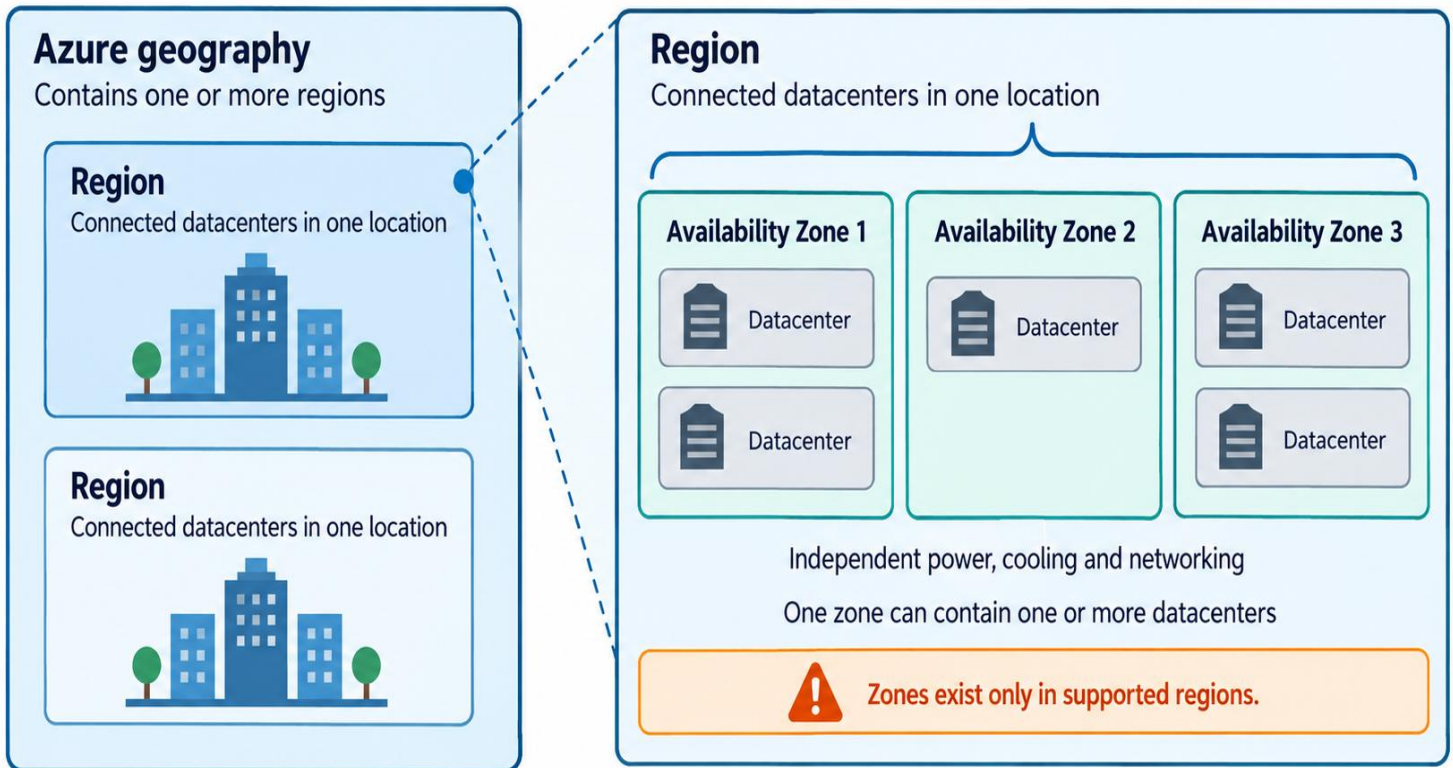
- Distinguish geographies, regions, availability zones, and datacenters as different placement and failure boundaries.
- Explain the separate purposes of high availability, fault tolerance, backup, and disaster recovery.
- Use SLO, SLA, RTO, and RPO concepts when discussing reliability requirements and regional design.

From geography to datacenter

Azure organizes physical infrastructure in several levels.

Chapter 2: Regions, Availability Zones, and Reliability Boundaries

2-1 Azure Physical Geography Hierarchy



007

Figure 2-1: Azure physical geography hierarchy

An Azure **geography** is a market and data-residency boundary that contains one or more regions. An Azure **region** is a defined area containing one or more datacenters connected by a low-latency network. An **availability zone** is a separated group of datacenters within a region, with independent supporting power, cooling, and networking.

A useful analogy is a city campus:

- The geography is the country or major legal market.
- The region is the campus in one metropolitan area.
- Availability zones are separate buildings or building groups on that campus.

A fire in one building might not affect another building on the same campus. A metropolitan disaster could affect the whole campus. This is why zones help with datacenter-scale failures, while multiple regions address regional failures.

Not every service is available in every region, and not every service uses availability zones in the same way. Availability must be checked for the exact service, tier, and region before deployment.

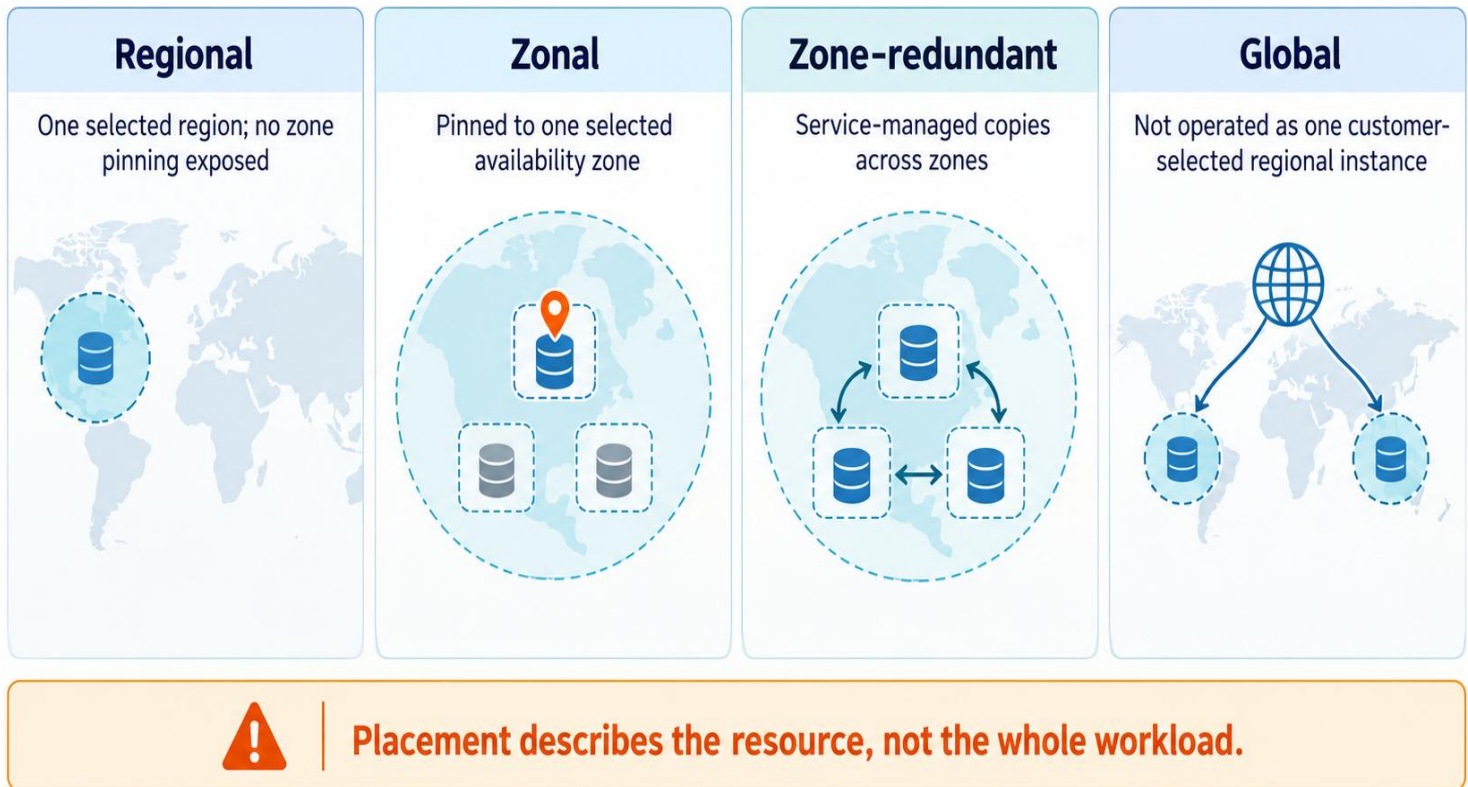
Regional, zonal, zone-redundant, and global resources

Azure resources have different placement behaviors:

- A **regional resource** is deployed in a region but might not expose a choice of zone.
- A **zonal resource** is pinned to a selected availability zone, such as a virtual machine deployed in Zone 1.
- A **zone-redundant resource** has service-managed replicas across multiple zones in a region.
- A **global resource** is not operated as a single customer-selected regional instance, although it can depend on regional endpoints.

Chapter 2: Regions, Availability Zones, and Reliability Boundaries

2-2 Regional, Zonal, Zone-Redundant, and Global Placement



008

Figure 2-2: Regional, zonal, zone-redundant, and global placement

These labels describe resource placement, not complete application reliability. A zone-redundant front end can still fail if it depends on a single zonal database or a DNS configuration stored and managed incorrectly.

Four protections that are not interchangeable

Protection	Primary purpose	Typical example	What it does not automatically provide
High availability	Keep service accessible during expected component failures	Two application instances across zones	Historical recovery after deletion or corruption
Fault tolerance	Continue operating with little or no interruption when a component fails	Redundant components with automatic failover	Protection from every regional or logical failure
Backup	Preserve recoverable copies from earlier points in time	Scheduled VM or database restore points	Immediate service continuity
Disaster recovery	Restore or fail over a workload after a major outage	A tested secondary-region recovery plan	Protection from bad data replicated to both sites

A backup answers, “Can the data be restored?” High availability answers, “Can the service continue when a component fails?” Disaster recovery answers, “Can the workload be restored or moved after a larger failure?” A mature design can need all three.

Chapter 2: Regions, Availability Zones, and Reliability Boundaries
2-3 Backup, High Availability, Disaster Recovery, and Replication

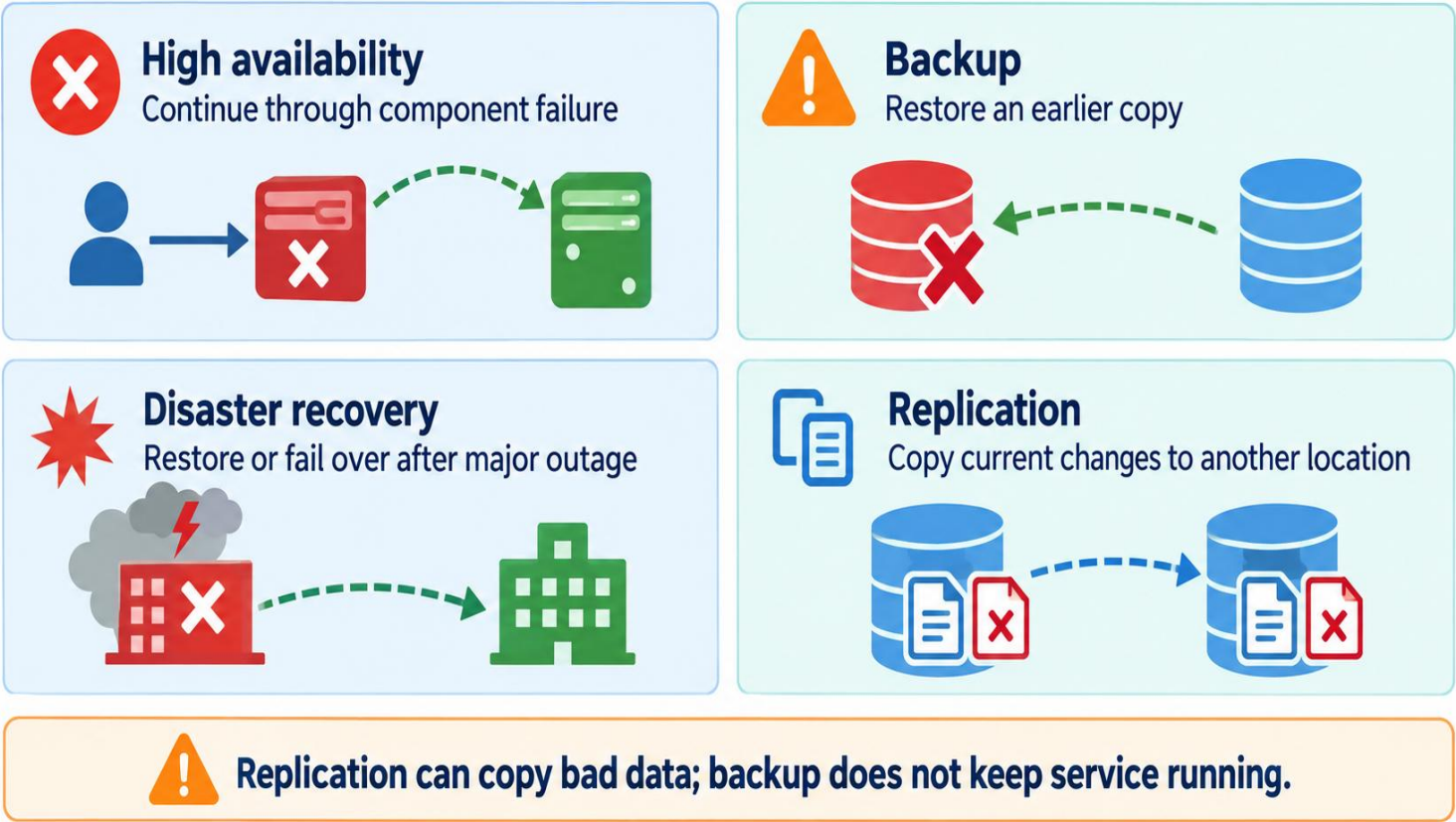


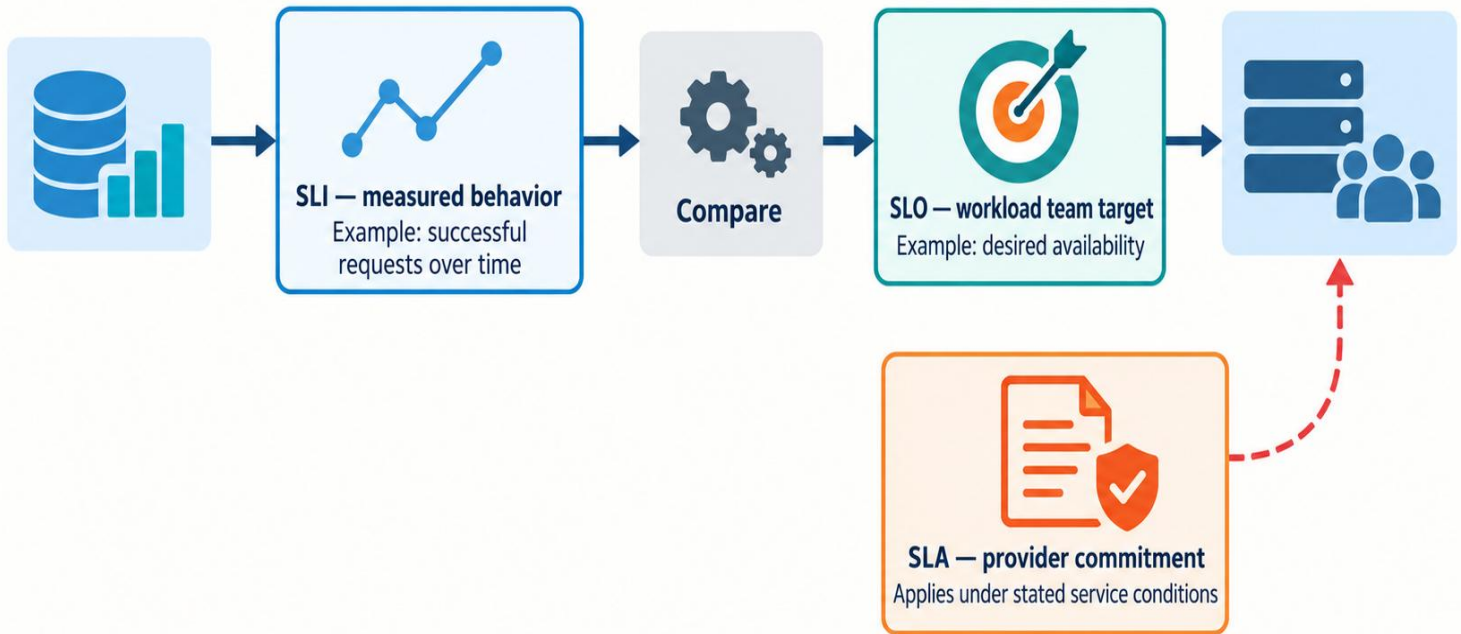
Figure 2-3: Backup, high availability, disaster recovery, and replication

The reliability vocabulary

A **service-level agreement (SLA)** is a provider's contractual availability commitment for a service under stated conditions. A **service-level objective (SLO)** is the workload team's internal reliability target. An application SLO is not automatically equal to the SLA of one Azure service because the application depends on several components.

Chapter 2: Regions, Availability Zones, and Reliability Boundaries

2-4 SLI, SLO, and SLA Relationship



Measure the workload; do not substitute one service SLA for the application SLO.

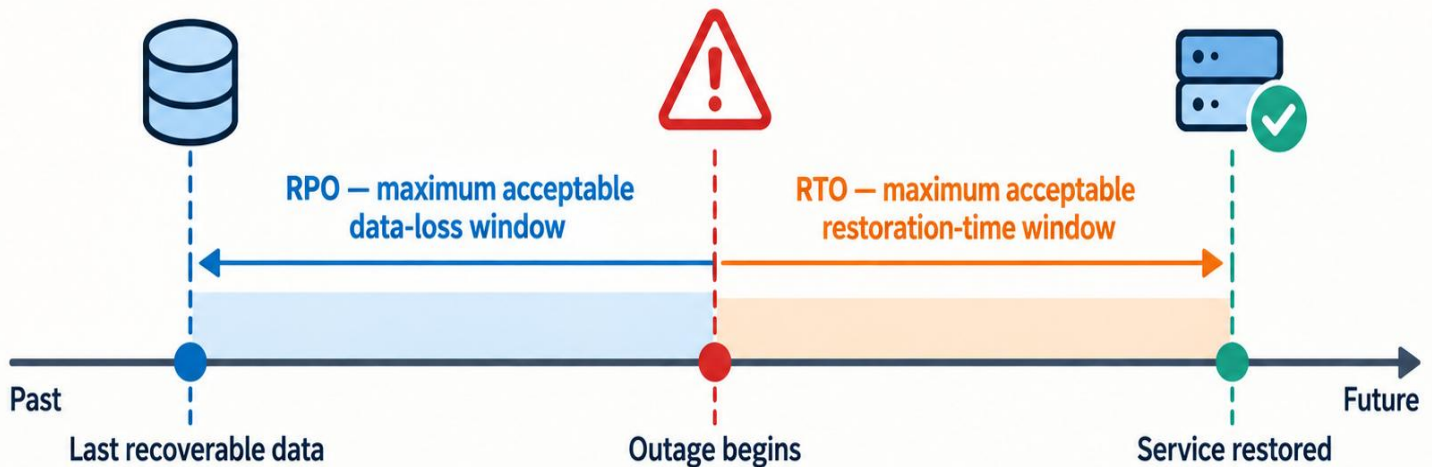
010

Figure 2-4: SLI, SLO, and SLA relationship

A **recovery time objective (RTO)** is the maximum acceptable time to restore a service after disruption. A **recovery point objective (RPO)** is the maximum acceptable amount of data loss measured in time.

Chapter 2: Regions, Availability Zones, and Reliability Boundaries

2-4 RTO and RPO on an Outage Timeline



RPO concerns data; RTO concerns time to restore service.

011

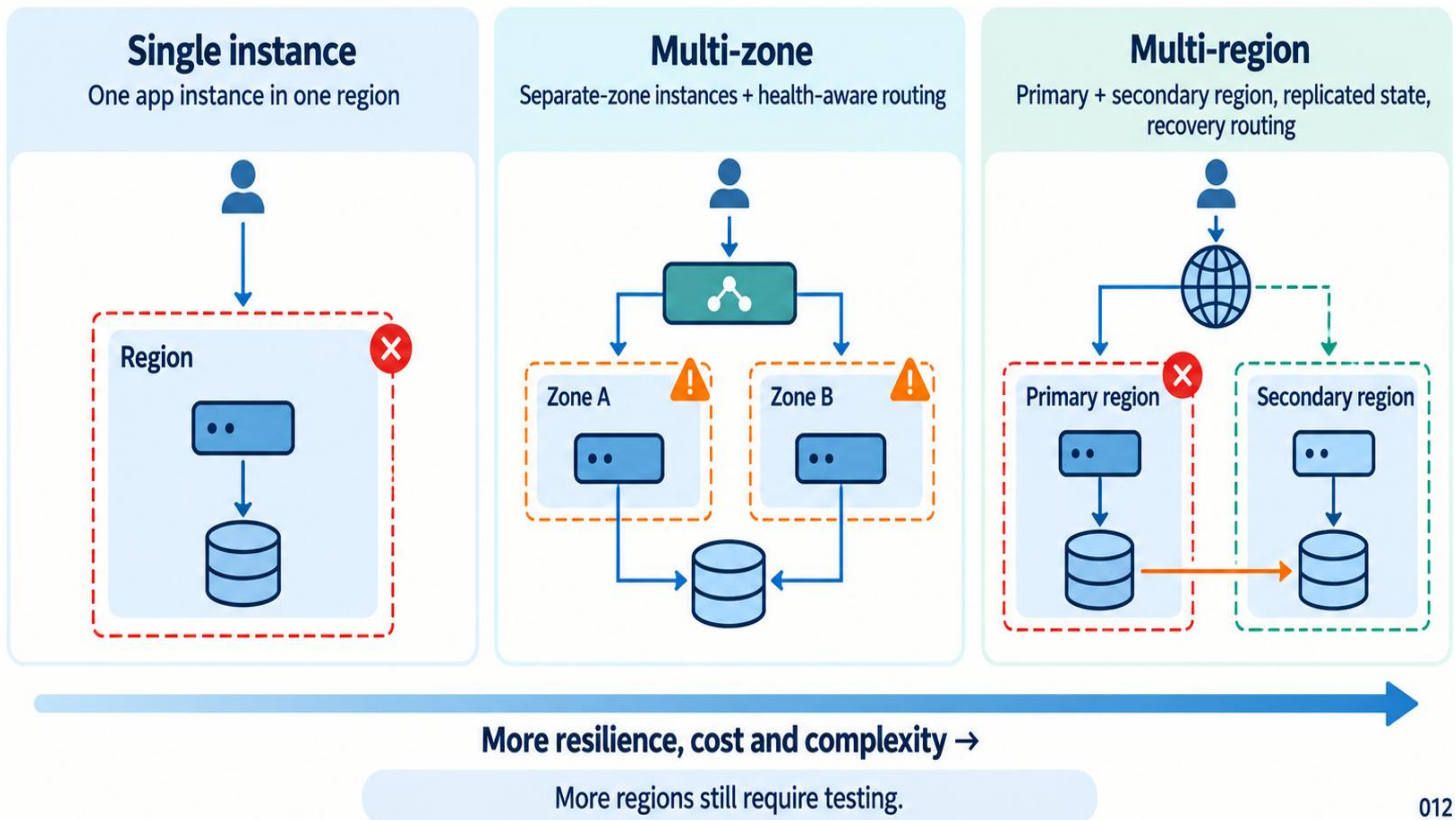
Figure 2-5: RTO and RPO on an outage timeline

For example, an RTO of two hours means the business expects the application to return within two hours. An RPO of fifteen minutes means losing more than the latest fifteen minutes of committed data is unacceptable. These are business requirements that guide technical choices; they are not settings that Azure infers automatically.

Three common design levels

A single-instance design uses one application instance in one region. It is simple and inexpensive but depends on that instance and its placement. It can be suitable for a noncritical development environment when downtime is acceptable.

Chapter 2: Regions, Availability Zones, and Reliability Boundaries 2-5 Single-Instance, Multi-Zone, and Multi-Region Designs



012

Figure 2-6: Single-instance, multi-zone, and multi-region designs

A multi-zone design places redundant components across availability zones or uses zone-redundant services. It can continue through many datacenter-scale failures within the region. The application must have at least two healthy instances where needed, and traffic distribution must stop sending requests to failed instances.

A multi-region design runs or prepares workload components in more than one region. It can support regional disaster recovery or active service from multiple regions. It also introduces data replication, routing, consistency, operational, and cost complexity.

More regions do not automatically mean better reliability. A poorly tested multi-region system can fail in more complicated ways than a well-operated single-region system.

Shared dependencies can cancel redundancy

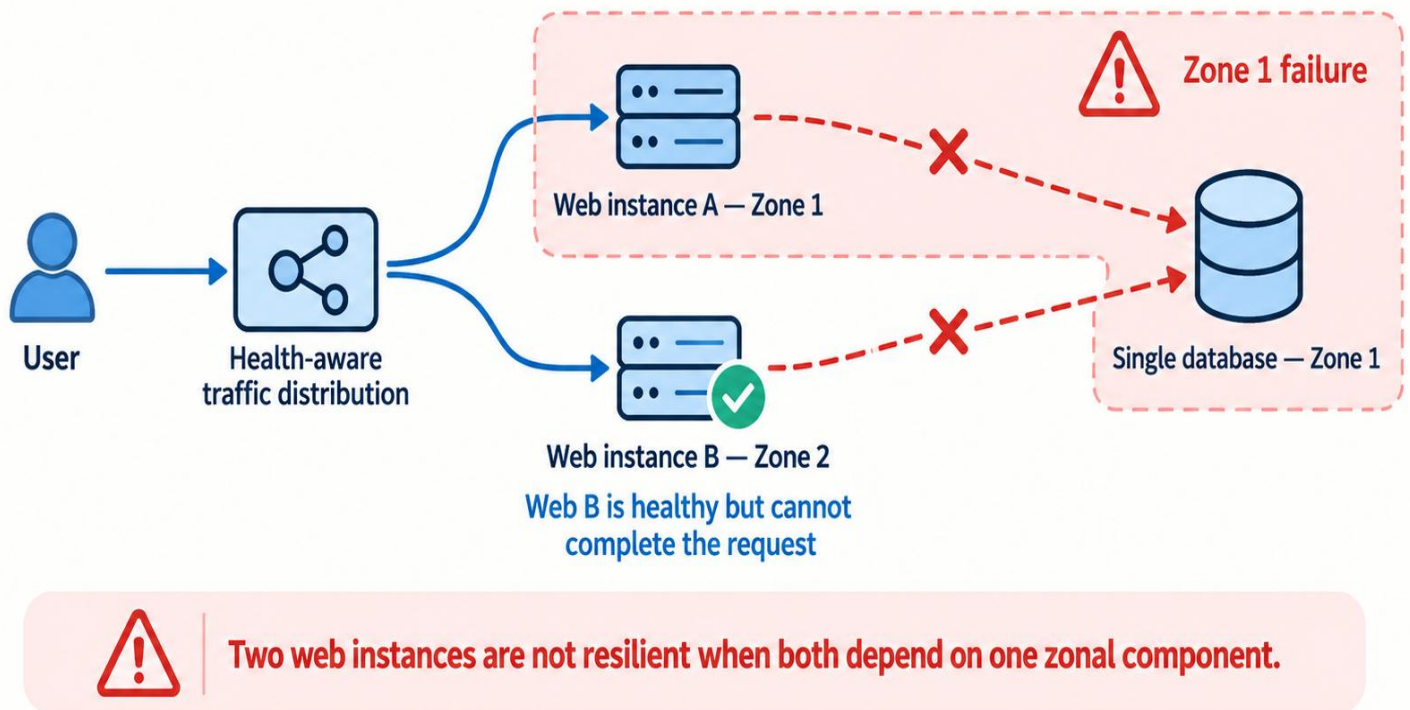
Suppose the asset-request application has two web servers in separate zones. The design is not truly zone resilient if both servers depend on:

- One database instance in a single zone.
- One firewall appliance without redundancy.
- One certificate that no one can renew.
- One DNS server reachable through a failed network path.

- One identity or secret configuration that cannot be accessed during the incident.

Chapter 2: Regions, Availability Zones, and Reliability Boundaries

2-6 Hidden Shared Dependency Cancels Redundancy



013

Figure 2-7: Hidden shared dependency cancels redundancy

Reliability analysis must follow the request from the user through every dependency. A redundant compute tier is only one part of the chain.

Choosing a region step by step

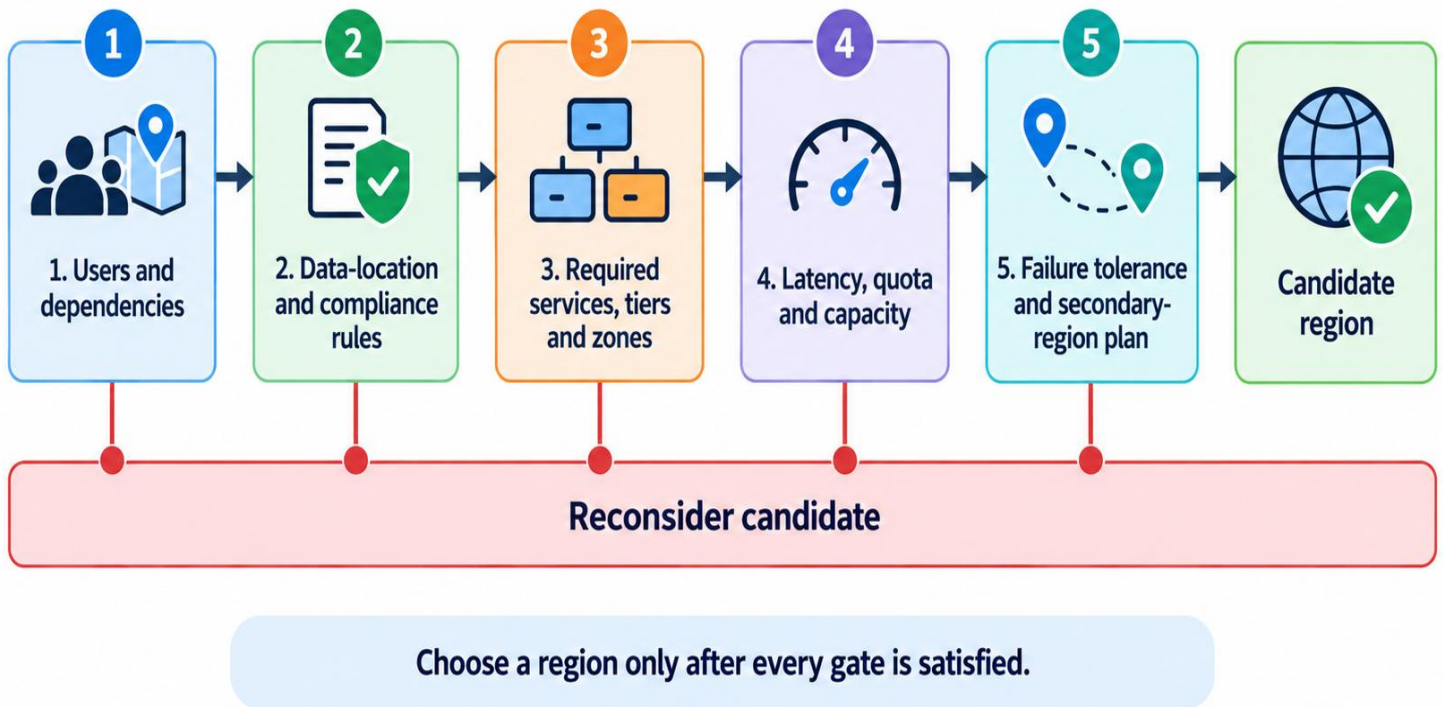
For a new internal workload, use a repeatable sequence:

1. Identify where users and dependent systems are located.
2. Identify legal, contractual, and organizational data-location requirements.
3. Confirm that required Azure services, tiers, virtual machine sizes, and zone features are available.
4. Estimate latency to users and hybrid dependencies.
5. Confirm quotas and likely regional capacity for the intended resources.
6. Decide which failures must be tolerated in the primary region.

7. Decide whether a secondary region is required for recovery.

Chapter 2: Regions, Availability Zones, and Reliability Boundaries

2-7 Region-Selection Sequence



014

Figure 2-8: Region-selection sequence

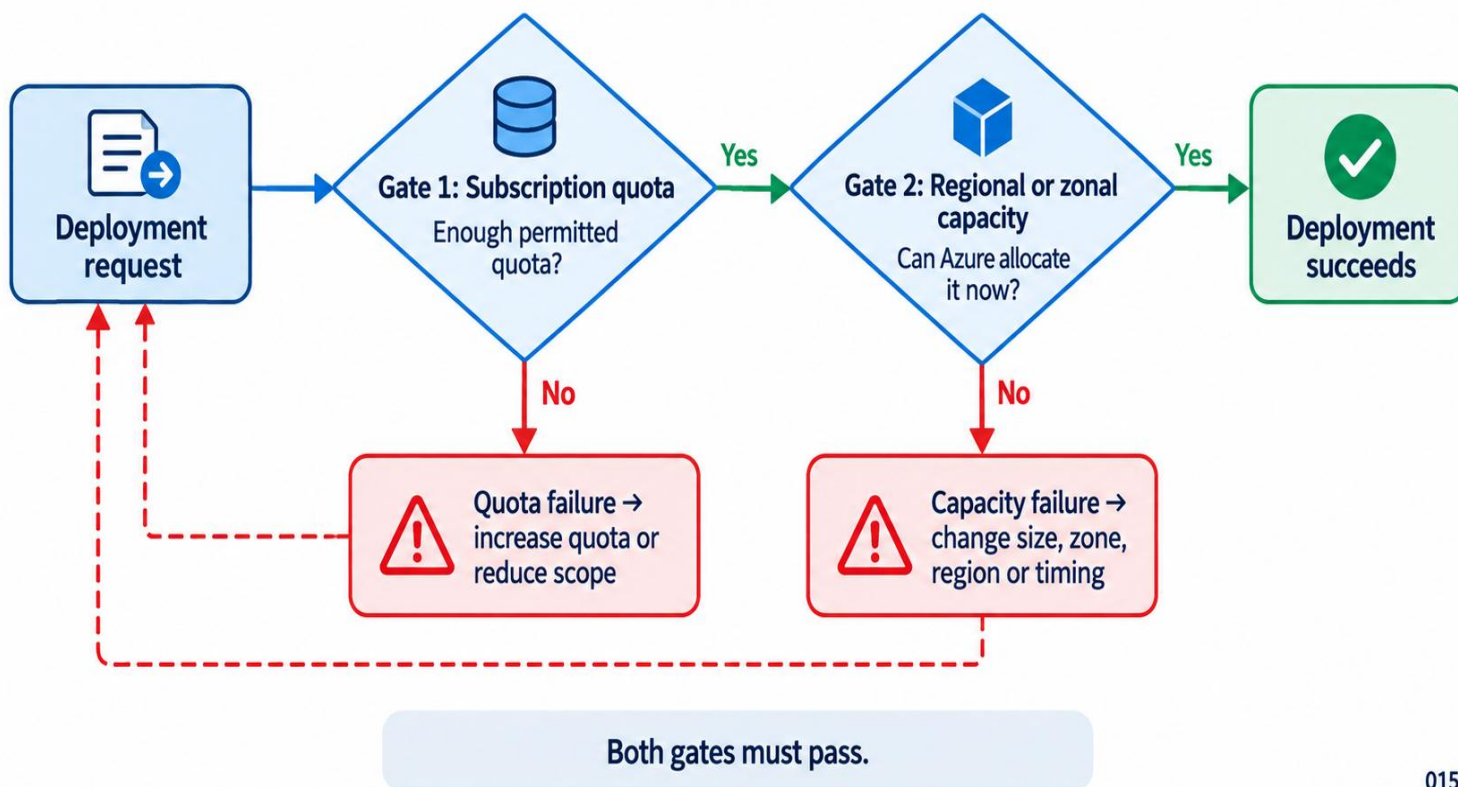
Do not select a region solely because its name looks geographically close. Network latency, service availability, compliance requirements, and organizational standards all matter.

Capacity and quota are reliability concerns

A **quota** is a subscription limit on the amount or number of a resource that can be deployed. Regional capacity is the provider's ability to allocate the selected resource in that location at that time.

Chapter 2: Regions, Availability Zones, and Reliability Boundaries

2-8 Quota and Regional Capacity as Deployment Gates



015

Figure 2-9: Quota and regional capacity as deployment gates

A design can be valid on paper and still fail during deployment because the subscription lacks virtual CPU quota or a requested virtual machine size has no capacity in a zone. Validate quotas and availability before a migration window. For critical scale-out or recovery plans, confirm that the secondary location can actually allocate the required capacity.

Applying the concepts to the example

The asset-request application is initially non-production. A single application instance might be acceptable if the team explicitly accepts downtime. When the application becomes important to business operations, the team could deploy two instances across zones behind a health-aware traffic distributor and choose zone-resilient data services.

If the business requires recovery after a regional outage, the team must also decide how application configuration, data, DNS, identity access, and network connectivity will be restored in another region. Copying only the web server is not a disaster-recovery plan.

Reliability mistakes to avoid

“A region and an availability zone are the same.” A zone is a failure boundary inside a region. Multiple zones do not protect against every region-wide event.

“The service SLA is the application SLA.” An application combines services and customer configuration. Its availability must be designed and measured as a complete workload.

“Backup is disaster recovery.” Backup is one recovery input. Recovery also requires compute, identity, networking, DNS, configuration, dependencies, procedures, and tested timing.

“Two instances are redundant wherever they are placed.” Two instances in the same failure boundary can fail together. Their placement and shared dependencies determine the actual protection.

The practical conclusion is simple: **design around explicit failure boundaries, not around the assumption that cloud resources are automatically resilient.**

Chapter Summary

Azure reliability depends on deliberate placement, independent dependencies, measurable objectives, and tested recovery paths. A resource being in Azure, or even being duplicated, does not establish which failures the complete workload can survive.

Review Questions

1. What failure does a multi-zone design address that two instances in one zone do not?
2. Why is a service SLA not automatically the availability objective for the whole application?
3. How do RTO and RPO lead to different technical decisions?