100 XP

Introduction

2 minutes

Imagine that you're building a new system on the cloud or migrating an existing system to the cloud. How do you instill confidence in your customers that their data is secure? Can your architecture handle a spike in traffic if a media report goes viral? Can your architecture handle the failure of one or more critical components? Are you using resources in the most efficient way?

The Azure Well-Architected Framework will help you to design, build, and continuously improve a secure, reliable, and efficient application. In this module, we'll introduce you to the framework, along with the pillars and principles that are essential to a great Azure architecture.

The concepts discussed in this module are not all-inclusive. They represent some of the important considerations when you're building a solution on the cloud. For more details on the Azure Well-Architected Framework, visit the Azure Architecture Center as you start planning and designing your architecture.

Learning objectives

By the end of this module, you'll be able to:

- Describe the pillars of the Azure Well-Architected Framework
- Identify key principles for creating a solid architectural foundation

Prerequisites

- Experience building or operating solutions by using core infrastructure technology such as data storage, compute, and networking
- Experience building or operating technology systems to solve business problems

Next unit: Azure Well-Architected Framework pillars

Continue >

100 XP

Azure Well-Architected Framework pillars

10 minutes

The cloud has changed the way organizations solve their business challenges, and how applications and systems are designed. The role of a solution architect is not only to deliver business value through the functional requirements of the application. It's also to ensure that the solution is designed in ways that are scalable, resilient, efficient, and secure.

Solution architecture is concerned with the planning, design, implementation, and ongoing improvement of a technology system. The architecture of a system must balance and align the business requirements with the technical capabilities that are needed to execute those requirements. The finished architecture is a balance of risk, cost, and capability throughout the system and its components.

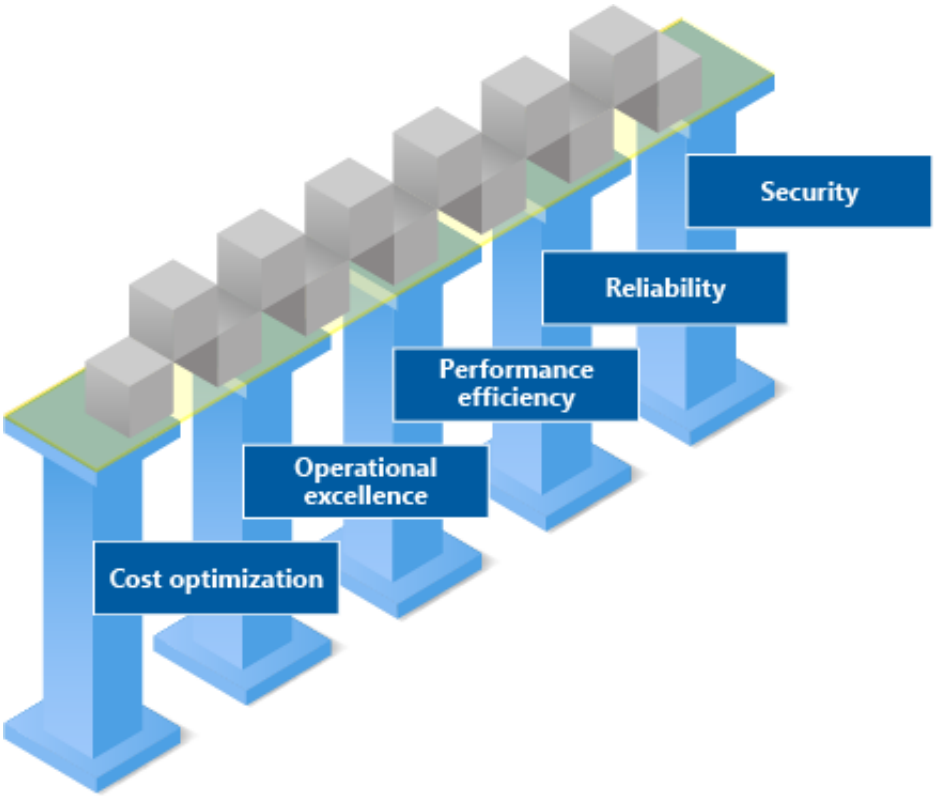
Azure Well-Architected Framework

The Azure Well-Architected Framework is a set of guiding tenets to build high-quality solutions on Azure. There's no one-size-fits-all approach to designing an architecture, but there are some universal concepts that will apply regardless of the architecture, technology, or cloud provider.

These concepts are not all-inclusive, but focusing on them will help you build a reliable, secure, and flexible foundation for your application.

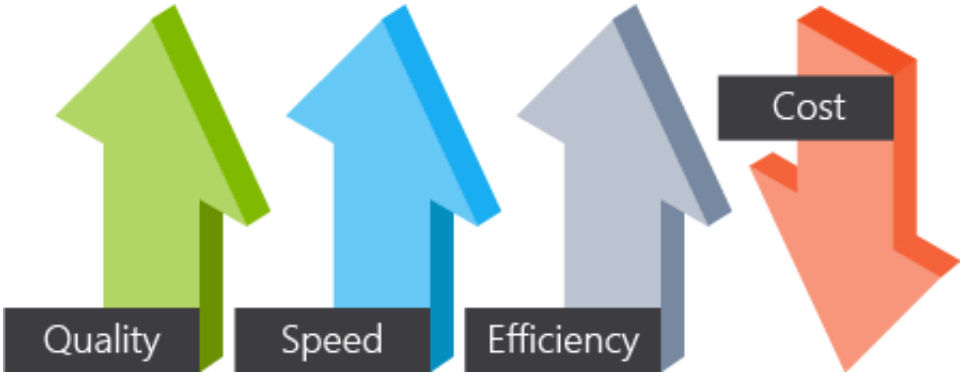
The Azure Well-Architected Framework consists of five pillars:

- Cost optimization
- Operational excellence
- Performance efficiency
- Reliability
- Security



Cost optimization

You'll want to design your cloud environment so that it's cost-effective for operations and development. Identify inefficiency and waste in cloud spending to ensure you're spending money where you can make the greatest use of it.



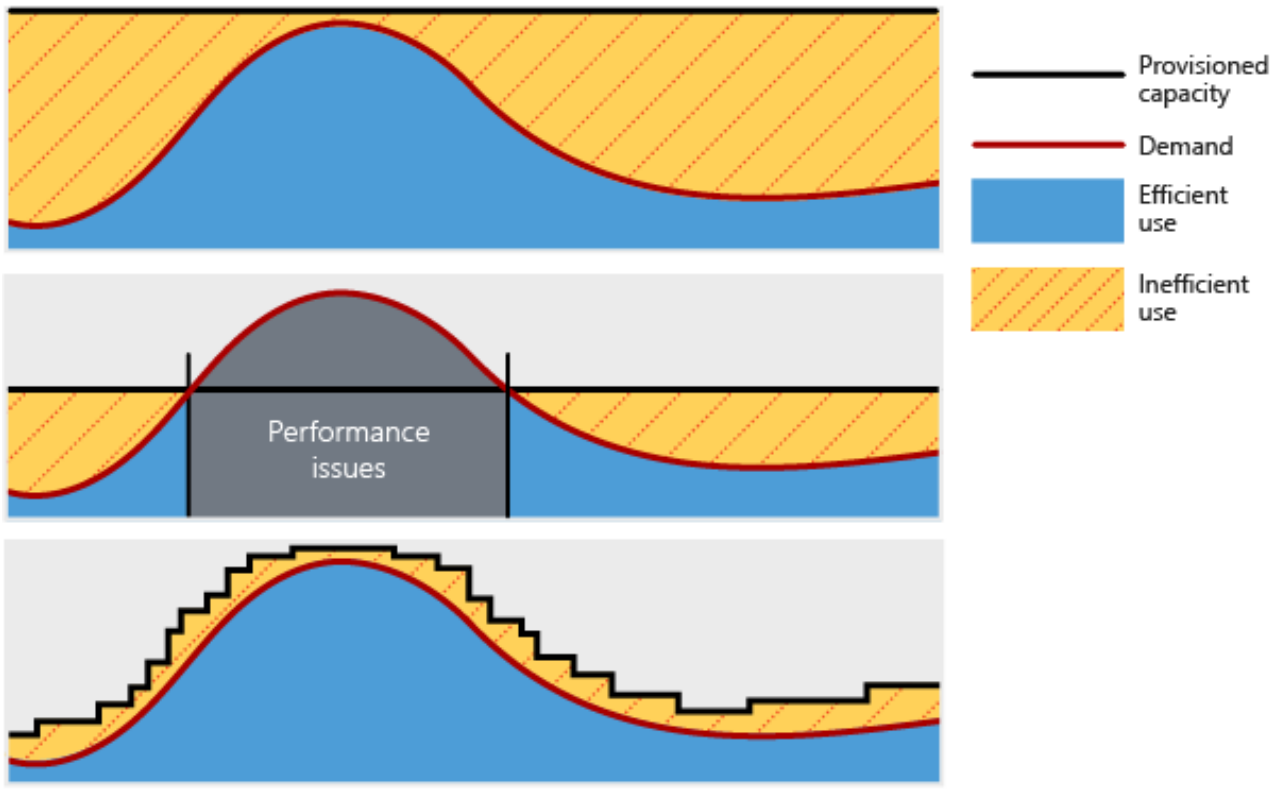
Operational excellence

By taking advantage of modern development practices such as DevOps, you can enable faster development and deployment cycles. You need to have a good monitoring architecture in place so that you can detect failures and problems before they happen or, at a minimum, before your customers notice. Automation is a key

aspect of this pillar to remove variance and error while increasing operational agility.

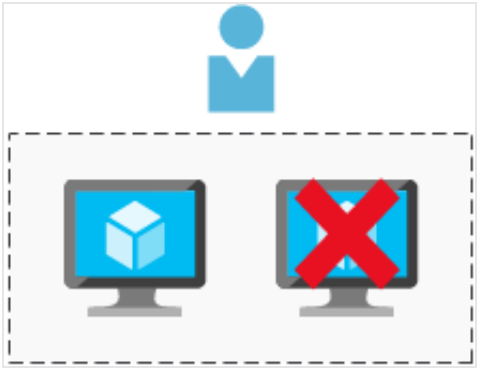
Performance efficiency

For an architecture to perform well and be scalable, it should properly match resource capacity to demand. Traditionally, cloud architectures accomplish this balance by scaling applications dynamically based on activity in the application. Demand for services changes, so it's important for your architecture to be able to adjust to demand. By designing your architecture with performance and scalability in mind, you'll provide a great experience for your customers while being cost-effective.



Reliability

Every architect's worst fear is having an architecture fail with no way to recover it. A successful cloud environment is designed in a way that anticipates failure at all levels. Part of anticipating failures is designing a system that can recover from a failure within the time that your stakeholders and customers require.



Security

Data is the most valuable piece of your organization's technical footprint. In this pillar, you'll focus on securing access to your architecture through authentication and protecting your application and data from network vulnerabilities. You should protect the integrity of your data too, through tools like encryption.

You must think about security throughout for the entire lifecycle of your application, from design and implementation to deployment and operations. The cloud provides protections against a variety of threats, such as network intrusion and DDoS attacks. But you still need to build security into your application, processes, and organizational culture.

General design principles

In addition to each of these pillars, there are some consistent design principles that you should consider throughout your architecture.

- **Enable architectural evolution**: No architecture is static. Allow for the evolution

of your architecture by taking advantage of new services, tools, and technologies when they're available.

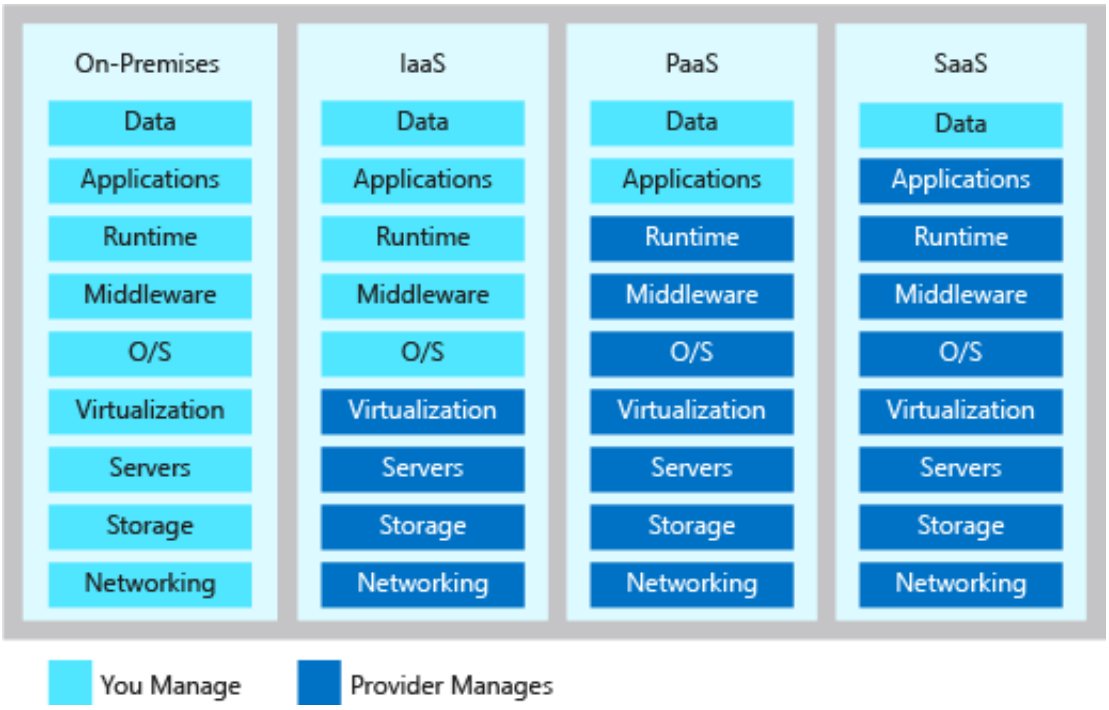
- **Use data to make decisions**: Collect data, analyze it, and use it to make decisions surrounding your architecture. From cost data, to performance, to user load, using data will guide you to make the right choices in your environment.
- **Educate and enable**: Cloud technology evolves quickly. Educate your development, operations, and business teams to help them make the right decisions and build solutions to solve business problems. Document and share configurations, decisions, and best practices within your organization.
- **Automate**: Automation of manual activities reduces operational costs, minimizes error introduced by manual steps, and provides consistency between environments.

Shared responsibility

Moving to the cloud introduces a model of shared responsibility. In this model, your cloud provider will manage certain aspects of your application, leaving you with the remaining responsibility.

In an on-premises environment, you're responsible for everything. As you move to infrastructure as a service (IaaS), then to platform as a service (PaaS) and software as a service (SaaS), your cloud provider will take on more of this responsibility.

This shared responsibility will play a role in your architectural decisions, because these decisions can have implications on cost, security, and your application's technical and operational capabilities. By shifting these responsibilities to your provider, you can focus on bringing value to your business and move away from activities that aren't a core business function.



Design choices

In an ideal architecture, you would build the most secure, high-performance, highly available, and efficient environment possible. However, as with everything, there are trade-offs.

To build an environment with the highest level of all these pillars, there's a cost. That cost might be in money, time to deliver, or operational agility. Every organization will have different priorities that will affect the design choices that are made in each pillar. As you design your architecture, you'll need to determine which trade-offs are acceptable, and which are not.

When you're building an Azure architecture, there are many considerations to keep in mind. You want your architecture to be secure, scalable, available, and recoverable. To make that possible, you'll have to make decisions based on cost, organizational priorities, and risk.

Next unit: Cost optimization

Continue >

200 XP

Cost optimization

10 minutes

Your organization has moved a majority of its systems to the cloud, but you're now seeing cost increases in areas you didn't expect. After some observation, you realize that you're inefficient across your environment, and you're still doing manual operational work.

In this unit, you'll learn about cost optimization. You'll also look at ways to reduce unnecessary expenses and improve operational efficiencies.

What is cost optimization?

Cost optimization is ensuring that the money your organization spends is being used to maximum effect. Cloud services provide computing as a utility. Technologies in the cloud are provided under a service model, to be consumed on demand. This on-demand service offering drives a fundamental change that directly affects planning, bookkeeping, and organizing.

When an organization decides to own infrastructure, it buys equipment that goes onto the balance sheet as assets. Because a capital investment was made, accountants categorize this transaction as a capital expense (CapEx). Over time, to account for the assets' limited useful lifespan, assets are depreciated or amortized.

Cloud services, on the other hand, are categorized as an operating expense (OpEx), because of their consumption model. Under this scheme, there is no asset to amortize. Instead, OpEx has a direct impact on net profit, taxable income, and the associated expenses on the balance sheet.

When an organization adopts a cloud platform, it must shift away from CapEx-oriented budgeting toward OpEx. This move reflects the shift from owning infrastructure to leasing solutions. Some organizations can derive value just from this

new accounting model. For example, a startup company can attract investors by demonstrating a profitable idea at large scale, without needing a large investment up front to purchase infrastructure.

To optimize costs in your organization's architecture, you can use several principles.

Plan and estimate costs

For any cloud project, whether it's the development of a new application or the migration of an entire datacenter, it's important to get an estimate of your costs. This estimate involves identifying any current resources to move or redevelop, understanding business objectives that might affect sizing, and selecting the appropriate services for the project.

With the requirements identified, you can use cost-estimation tools to provide a more concise estimate of the resources that would be required. Transparency is important here, so that all stakeholders can review for accuracy and have visibility into the costs that are associated with the project.

Provision with optimization

Provisioning services that are optimized for cost from the outset can reduce your work effort in the future. For example, you should ensure that you're selecting the appropriate service level for your workload, and take advantage of services that let you adjust the service level. You should also use discounts when they're available, such as reserved instances and bring-your-own-license offers.

Where possible, you want to move from IaaS to PaaS services. PaaS services typically cost less than IaaS, and they generally reduce your operational costs.

With PaaS services, you don't have to worry about patching or maintaining VMs, because the cloud provider typically handles those activities. Not all applications can be moved to PaaS, but with the cost savings that PaaS services provide, it's worth considering.

Use monitoring and analytics to gain cost insights

If you're not monitoring your spending, you don't know what you can save. Take advantage of cost management tools and regularly review billing statements to better understand where money is being spent.

Take time to conduct regular cost reviews across services to understand if the expenditure is appropriate for the resource requirements of the workload. Adjust expenditures as necessary. Identify and track down any cost anomalies that might show up on billing statements or through alerts. If you notice a large spike in cost associated with network traffic, it might uncover both cost savings and potential technical issues.

Maximize efficiency of cloud spend

Efficiency is focused on identifying and eliminating unnecessary expenses within your environment. The cloud is a pay-as-you-go service, and avoidable expenses are typically the result of provisioning more capacity than your demand requires. Operational costs can also contribute to unnecessary or inefficient costs. These inefficient operational costs show up as wasted time and increased error. As you design your architecture, identify and eliminate waste across your environment.

Waste can show up in several ways. Let's look at a few examples:

- A virtual machine that's always 90 percent idle
- Paying for a license included in a virtual machine when a license is already owned
- Retaining infrequently accessed data on a storage medium optimized for frequent access
- Manually repeating the build of a non-production environment

In each of these cases, you're spending more money than you should. Each case presents an opportunity for cost reduction.

As you evaluate your cost, take the opportunity to optimize environments. Capacity demands can and will change over time, and many cloud services can manually or dynamically adjust the provisioned resources to meet the demands. These adjustments can drive the balance between a well-running application and the most cost-effective size.

Optimize your systems at every level. At the network level, ensure that data transfer is efficient and meets the expectations of your customers. Use services to cache data to increase application performance and reduce the transaction load on your data-storage services. Identify and decommission unused resources. Take advantage of lower-cost data-storage tiers to archive infrequently accessed data.

Check your knowledge

1. Which of the following is an example of waste, resulting in an increased resource cost?

- ○ Archiving infrequently accessed data to an archive storage tier.
- ○ Using a service that automatically adjusts resources that are provisioned to match user load.
- ○ Pooling databases to share provisioned capacity.
- ○ Running a development environment overnight that is used only during business hours.

2. Which of the following is a good practice to reduce costs?

- ○ Conducting regular reviews of cloud bills to identify abnormal increases in spend.
- ○ Letting all IT teams have access to provision virtual machines of any size.
- ○ Provisioning the same capacity in development environments as for production, even though resource requirements are substantially lower in development environments.
- ○ Provisioning virtual machines that include licensing costs rather than using a bring-your-own-license image.

3. Suppose you have recently moved your application to the cloud and your monthly bill seems higher than expected. The utilization level of your VM is high enough that

you're hesitant to downsize. What might be a reasonable next step you can take to help you find inefficiencies?

- ○ Wait a month and recheck your bill.
- ○ Increase the amount of application testing you do before each release.
- ○ Add monitoring and instrumentation to your application.

Check your answers

200 XP

Operational excellence

10 minutes

Moving just your resources to the cloud is taking advantage of only a small portion of what the cloud can bring to your organization. Along with the technical capabilities that the cloud brings, you can improve your operational capabilities as well. From improving developer agility to improving your visibility into the health and performance of your application, you can use the cloud to improve the operational capabilities of your organization.

In this unit, we'll look at the pillar of operational excellence.

What is operational excellence?

Operational excellence is about ensuring that you have full visibility into how your application is running, and ensuring the best experience for your users. Operational excellence includes making your development and release practices more agile, which allows your business to quickly adjust to changes. By improving operational capabilities, you can have faster development and release cycles, and a better experience for your application's users.

You can use several principles when driving operational excellence through your architecture.

Design, build, and orchestrate with modern practices

Modern architectures should be designed with DevOps and continuous integration in mind. A modern architecture will give you the ability to automate deployments by using infrastructure as code, automate application testing, and build new environments as needed. DevOps is as much cultural as it is technical, but can bring many benefits to organizations that embrace it.

Regardless of whether your project is a greenfield application that uses full continuous integration and continuous deployment (CI/CD) and containers, or if it's a legacy application that you're continuing to service, there are DevOps practices you can bring into your organization.

Breaking down silos within an organization is a common thread throughout DevOps. So is working collaboratively across every stage in a project, including change management. Creating a culture of sharing, collaboration, and transparency will bring operational excellence to your organization.

Use monitoring and analytics to gain operational insights

Throughout your architecture, you want to have a thorough monitoring, logging, and instrumentation system. By creating an effective system for monitoring what's going on in your architecture, you can ensure that you'll know when something isn't right before your users are affected. With a comprehensive approach to monitoring, you'll be able to identify performance issues and cost inefficiencies, correlate events, and gain a greater ability to troubleshoot issues.

Operationally, it's important to have a robust monitoring strategy. This helps you identify areas of waste, troubleshoot issues, and optimize the performance of your application. A multilayered approach is essential. Gathering data points from components at every layer will help alert you when values are outside acceptable ranges and help you track spending over time.

Use automation to reduce effort and error

You should automate as much of your architecture as possible. The human element is costly, injecting time and error into operational activities. This increased time and error will result in increased operational costs. You can use automation to build, deploy, and administer resources. By automating common activities, you can eliminate the delay in waiting for a human to intervene.

Test

You should include testing in your application deployment and your ongoing operations. A good testing strategy will help you identify issues in your application

before it's deployed, and ensure that dependent services can properly communicate with your application.

A good testing strategy can also help identify performance issues and potential security vulnerabilities in both pre-production and production deployments. A robust testing plan can uncover issues with infrastructure deployments that can affect the user experience, and testing will help you provide a great experience for your users.

Check your knowledge

1. Which of the following is a good example of using testing in your environment?

- ○ Waiting for users to reach out to you with reports of errors in your application.
- ○ Performing functionality tests in the development environment that are different from functionality tests in the production environment.
- ○ Omitting infrastructure deployment from test plans.
- ○ Performing regular security tests of your application code in development and production environments.

2. Which of the following is a good example of using automation to improve operational excellence?

- ○ Manually provisioning development environments every day for your development teams.
- ○ Logging on Linux VMs after deployment and installing the software packages that are required for the application.
- ○ Using a configuration template to deploy infrastructure in each environment.
- ○ Manually copying application binaries from your build system to your deployment infrastructure.

Check your answers

200 XP

Performance efficiency

10 minutes

Imagine that a news story was just published about one of your organization's recent product announcements. The additional publicity from the news story will undoubtedly bring a large influx of traffic to your website. Will your website be able to handle this traffic increase, or will the additional load cause your site to be slow or unresponsive?

In this unit, you'll look at some of the basic principles of ensuring outstanding application performance by using scaling and optimization principles that make up the performance efficiency pillar.

What is performance efficiency?

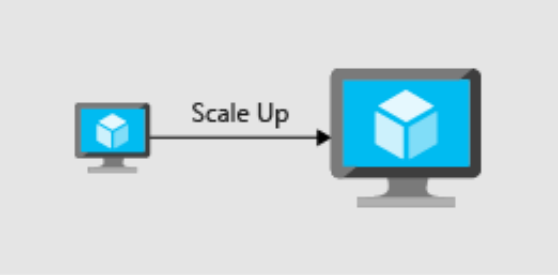
Performance efficiency is matching an application's available resources with the demand that it's receiving. Performance efficiency includes scaling resources, identifying and optimizing potential bottlenecks, and optimizing your application code for peak performance.

Let's look at some patterns and practices that can enhance the scalability and performance of your application.

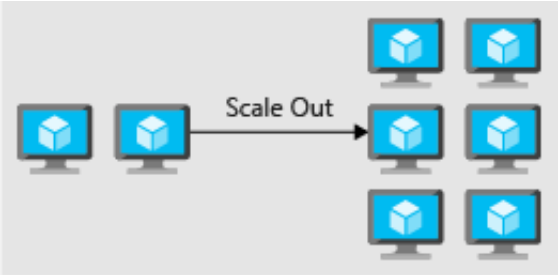
Scale up and scale out

Compute resources can be scaled in two directions:

- Scaling *up* is adding more resources to a single instance. This is also known as *vertical scaling*.



- Scaling *out* is adding more instances. This is also known as *horizontal scaling*.



Scaling up is concerned with adding more resources, such as CPU or memory, to a single instance. This instance might be a virtual machine or a PaaS service.

The act of adding more capacity to the instance increases the resources that are available to your application, but it does come with a limit. Virtual machines are limited to the capacity of the host that they run on, and hosts themselves have physical limitations. Eventually, when you scale up an instance, you can run into these limits. They restrict your ability to add more resources to the instance.

Scaling out is concerned with adding more instances to a service. They can be virtual machines or PaaS services. Instead of adding more capacity by making a single instance more powerful, we add capacity by increasing the total number of instances.

The advantage of scaling out is that you can conceivably scale out forever if you have more machines to add to the architecture. Scaling out requires some type of load distribution. This might be in the form of a load balancer that distributes requests across available servers, or it might be a service-discovery mechanism for identifying active servers to send requests to.

In both types of scaling, resources can be reduced, which brings cost optimization into the picture.

Autoscaling is the process of dynamically allocating resources to match performance requirements. As the volume of work grows, an application might need more

resources to maintain the desired performance levels and satisfy service-level agreements (SLAs). As demand slackens and the additional resources are no longer needed, they can be deallocated to minimize costs.

Autoscaling takes advantage of the elasticity of cloud-hosted environments while easing management overhead. It reduces the need for an operator to continually monitor the performance of a system and make decisions about adding or removing resources.

Optimize network performance

When you're optimizing for performance, you'll look at network and storage performance to ensure that their levels are within acceptable limits. These performance levels can affect the response time of your application. Selecting the right networking and storage technologies for your architecture will help you ensure that you're providing the best experience for your consumers.

Adding a messaging layer between services can have a benefit to performance and scalability. A messaging layer creates a buffer so that requests can continue to flow in without error if the receiving application can't keep up. As the application works through the requests, they'll be answered in the order in which they were received.

Optimize storage performance

In many large-scale solutions, data is divided into separate partitions that can be managed and accessed separately. The partitioning strategy must be chosen carefully to maximize the benefits while minimizing adverse effects. Partitioning can help improve scalability, reduce contention, and optimize performance.

Use caching in your architecture to help improve performance. Caching is a mechanism to store frequently used data or assets (webpages, images) for faster retrieval. You can use caching at different layers of your application. You can use caching between your application servers and a database in order to decrease data retrieval times.

You can also use caching between your users and your web servers by placing static content closer to users and decreasing the time it takes to return webpages to the

users. This has a secondary effect of offloading requests from your database or web servers, increasing the performance for other requests.

Identify performance bottlenecks in your application

Distributed applications and services running in the cloud are complex pieces of software that comprise many moving parts. In a production environment, it's important to be able to track the way in which users utilize your system, trace resource utilization, and generally monitor the health and performance of your system. You can use this information as a diagnostic aid to detect and correct issues. You can also use this information to help spot potential problems and prevent them from occurring.

Performance optimization will include understanding how the applications themselves are performing. Errors, poorly performing code, and bottlenecks in dependent systems can all be uncovered through an application performance-management tool. Often, these issues might be hidden or obscured for users, developers, and administrators, but they can have an adverse impact on the overall performance of your application.

Look across all layers of your application and identify and remediate performance bottlenecks. These bottlenecks might be poor memory handling in your application, or even the process of adding indexes into your database. It might be an iterative process as you relieve one bottleneck and then uncover another that you were unaware of.

With a thorough approach to performance monitoring, you'll be able to determine the types of patterns and practices that your architecture will benefit from.

Check your knowledge

1. Which of the following is an example of scaling up (vertical scaling)?

- ○ Updating your application to use a queuing service
- ○ Adding more web servers into a web farm
- ○ Adding another virtual machine into a database cluster

- ○ Updating a virtual machine to a larger size

2. Which of the following is an example of scaling out (horizontal scaling)?

- ○ Updating a virtual machine to a larger size
- ○ Adding more storage to a virtual machine
- ○ Adding more web servers into a web farm
- ○ Replicating backups to another region

Check your answers

200 XP

Reliability

10 minutes

Imagine that you run a clinical system for a healthcare organization. Clinicians and caregivers have little tolerance for downtime. They need to have access to clinical IT systems around the clock to ensure that they're providing the highest-quality care at all times.

To meet the around-the-clock demands of clinicians, applications must be able to handle failures with minimal impact to their users. How do they keep their applications operational, both for localized incidents and for large-scale disasters?

In this unit, you'll learn how to include elements from the reliability pillar in your architecture design.

What is reliability?

In a complex application, any number of things can go wrong at any scale. Individual servers and hard drives can fail. A deployment issue might unintentionally drop all tables in a database. Whole datacenters might become unreachable. A ransomware incident might maliciously encrypt all your data. It's critical that your application stays reliable and can handle both localized and broad-impact incidents.

Designing for reliability includes maintaining uptime through small-scale incidents and temporary conditions like partial network outages. You can ensure that your application can handle localized failures by integrating high availability into each component of the application and eliminating single points of failure. Such a design also minimizes the impact of infrastructure maintenance. High-availability designs typically aim to eliminate the impact of incidents quickly and automatically, and to ensure that the system can continue to process requests with little to no impact.

Designing for reliability also focuses on recovery from data loss and from larger-scale

disasters. Recovery from these types of incidents often involves active intervention, though automated recovery steps can reduce the time needed to recover. These types of incidents might result in some amount of downtime or permanently lost data. Disaster recovery is as much about careful planning as it is about execution.

Including high availability and recoverability in your architecture design protects your business from financial losses that result from downtime and lost data. They ensure that your reputation isn't negatively affected by a loss of trust from your customers.

Architecting for reliability ensures that your application can meet the commitments you make to your customers. This includes ensuring that your systems are *available* to end users and can *recover* from any failures.

Build a highly available architecture

For availability, identify the service-level agreement (SLA) to which you're committing. Examine the potential high-availability capabilities of your application relative to your SLA, and identify where you have proper coverage and where you'll need to make improvements. Your goal is to add redundancy to components of the architecture so that you're less likely to experience an outage.

Examples of high-availability design components include clustering and load balancing:

- Clustering replaces a single VM with a set of coordinated VMs. When one VM fails or becomes unreachable, services can fail over to another one that can service the requests.
- Load balancing spreads requests across many instances of a service, detecting failed instances and preventing requests from being routed to them.

Build an architecture that can recover from failure

For recoverability, you should perform an analysis that examines your possible data loss and major downtime scenarios. Your analysis should include an exploration of recovery strategies and the cost/benefit tradeoff for each. This exercise will give you important insight into your organization's priorities, and help clarify the role of your application. The results should include the application's:

- **Recovery point objective (RPO)**: The maximum duration of acceptable data loss. RPO is measured in units of time, not volume. Examples are "30 minutes of data," "four hours of data," and so on. RPO is about limiting and recovering from data *loss*, not data *theft*.

- **Recovery time objective (RTO)**: The maximum duration of acceptable downtime, where "downtime" is defined by your specification. For example, if the acceptable downtime duration is eight hours in the event of a disaster, then your RTO is eight hours.

With RPO and RTO defined, you can design backup, restore, replication, and recovery capabilities into your architecture to meet these objectives.

Every cloud provider offers a suite of services and features that you can use to improve your application's availability and recoverability. When possible, use existing services and best practices, and try to resist creating your own.

Hard drives can fail, datacenters can become unreachable, and hackers can attack. It's important that you maintain a good reputation with your customers by using availability and recoverability. Availability focuses on maintaining uptime through conditions like network outages, and recoverability focuses on retrieving data after a disaster.

Check your knowledge

1. Suppose you want to increase the availability of your system to provide a better service-level agreement (SLA) to your customers. Which of the following is a guiding principle you can use?

- ○ Reduce your target for maximum duration of acceptable data loss
- ○ Encrypt all data at rest
- ○ Eliminate single point of failure

2. Which of the following would be affected by your defined recovery point objective (RPO)?

- ○ The frequency of database backups
- ○ The number of regions that data is replicated to
- ○ The number of instances in a database cluster
- ○ The type of load-balancing technology used in your application

Check your answers

200 XP

Security

10 minutes

Healthcare organizations store personal and potentially sensitive customer data. Financial institutions store account numbers, balances, and transaction history. Retailers store purchase history, account information, and demographic details of customers. A security incident might expose this sensitive data, which might cause personal embarrassment or financial harm. How do you ensure the integrity of their data and ensure that your systems are secure?

In this unit, you'll learn about the important elements of the security pillar.

What is security?

Security is ultimately about protecting the data that your organization uses, stores, and transmits. The data that your organization stores or handles is at the heart of your securable assets. This data might be sensitive data about customers, financial information about your organization, or critical line-of-business data that supports your organization. Securing the infrastructure on which the data exists, along with the identities used to access it, is also critically important.

Your data might be subject to additional legal and regulatory requirements, depending on where you're located, the type of data you're storing, or the industry in which your application operates.

For instance, in the healthcare industry in the United States, there's a law called the Health Insurance Portability and Accountability Act (HIPAA). In the financial industry, the Payment Card Industry Data Security Standard is concerned with the handling of credit card data. Organizations that store data that's in scope for these laws and standards are required to ensure that certain safeguards are in place for the protection of that data. In Europe, the General Data Protection Regulation (GDPR) lays out the rules of how personal data is protected, and defines individuals' rights

related to stored data. Some countries require that certain types of data do not leave their borders.

When a security breach occurs, there can be substantial impacts to the finances and reputation of both organizations and customers. This breaks down the trust that customers are willing to instill in your organization, and can affect the organization's long-term health.

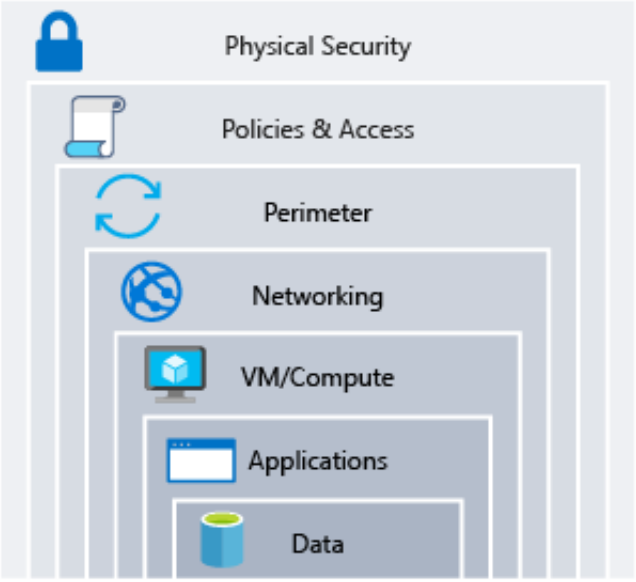
Defense in depth

A multilayered approach to securing your environment will increase the security posture of your environment. Commonly known as *defense in depth*, we can break down the layers as follows:

- Data
- Applications
- VM/compute
- Networking
- Perimeter
- Policies and access
- Physical security

Each layer focuses on a different area where attacks can happen, and creates a depth of protection if one layer fails or is bypassed by an attacker. If you were to focus on just one layer, an attacker would have unfettered access to your environment if they got through this layer.

Addressing security in layers increases the work an attacker must do to gain access to your systems and data. Each layer will have different security controls, technologies, and capabilities that will apply. When you're identifying the protections to put in place, cost is often of concern. You'll need to balance cost with business requirements and overall risk to the business.



Defense in-depth security layers

No single security system, control, or technology will fully protect your architecture. Security is more than just technology; it's also about people and processes. Creating an environment that looks holistically at security and makes it a requirement by default will help ensure that your organization is as secure as possible.

Protect from common attacks

At each layer, there are some common attacks that you'll want to protect against. The following list isn't all-inclusive, but it can give you an idea of how each layer can be attacked and what types of protections you might need.

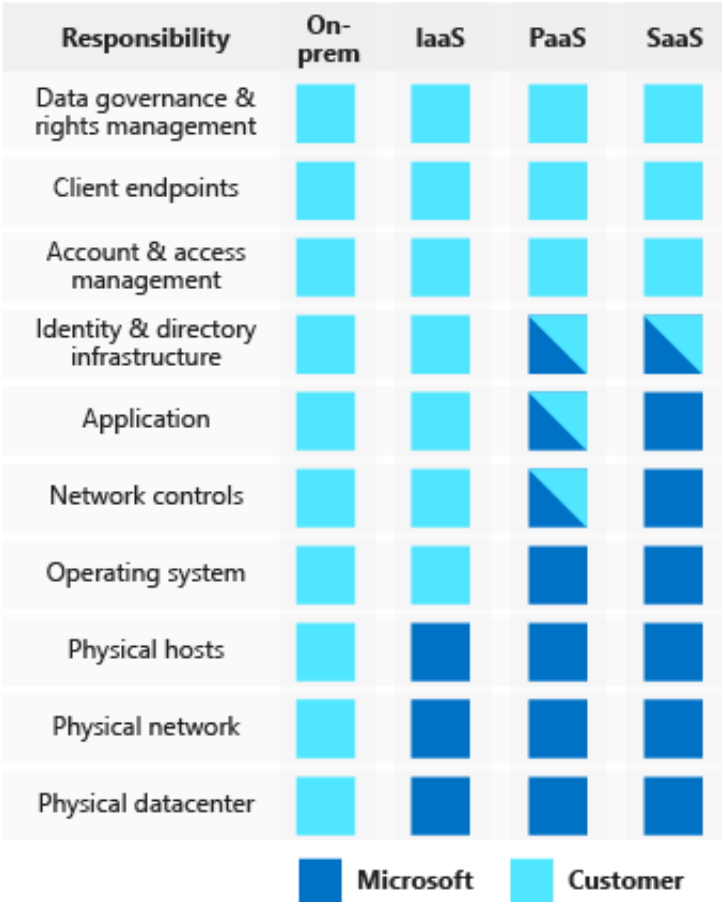
- **Data layer**: Exposing an encryption key or using weak encryption can leave your data vulnerable if unauthorized access occurs.
- **Application layer**: Malicious code injection and execution are the hallmarks of application-layer attacks. Common attacks include SQL injection and cross-site scripting (XSS).
- **VM/compute layer**: Malware is a common method of attacking an environment, which involves executing malicious code to compromise a system. After malware is present on a system, further attacks that lead to credential exposure and lateral movement throughout the environment can occur.
- **Networking layer**: Unnecessary open ports to the internet are a common

method of attack. These might include leaving SSH or RDP open to virtual machines. When these protocols are open, they can allow brute-force attacks against your systems as attackers attempt to gain access.

- **Perimeter layer**: Denial-of-service (DoS) attacks often happen at this layer. These attacks try to overwhelm network resources, forcing them to go offline or making them incapable of responding to legitimate requests.
- **Policies and access layer**: This layer is where authentication occurs for your application. This layer might include modern authentication protocols such as OpenID Connect, OAuth, or Kerberos-based authentication such as Active Directory. The exposure of credentials is a risk at this layer, and it's important to limit the permissions of identities. You also want to have monitoring in place to look for possible compromised accounts, such as logins coming from unusual places.
- **Physical layer**: Unauthorized access to facilities through methods, such as door drafting and theft of security badges, can happen at this layer.

Shared security responsibility

Revisiting the model of shared responsibility, we can reframe this in the context of security. Depending on the type of service you select, some security protections will be built in to the service, while others will remain your responsibility. Careful evaluation of the services and technologies that you select will be necessary, to ensure that you're providing the proper security controls for your architecture.



Check your knowledge

1. Which of the following types of data might need to have security protections?

- ○ Customer data that contains personal information
- ○ Financial data that supports business operations
- ○ Intellectual property
- ○ All of the above might need security protections

2. Which of the following is an example of an attack you might see at the policies and access layer?

- ○ Exposed credentials posted online
- ○ A SYN flood attack
- ○ Following an employee into a datacenter without presenting credentials

○ Ransomware that encrypts the disks of a virtual machine

Check your answers

100 XP

Introduction

2 minutes

How efficient is your cloud environment? Are you maximizing your resources and minimizing your cloud spend? Do you have visibility into your costs, both historical and projected? How do you maximize the licenses and resources you have already purchased on Azure? In this module, you'll learn about the *cost optimization* pillar of the Azure Well-Architected Framework.

The concepts described in this module are not all-inclusive, but they represent some of the important considerations when you're building a solution on the cloud. For more details about the Azure Well-Architected Framework, visit the Azure Architecture Center as you start planning and designing your architecture.

Learning objectives

By the end of this module, you'll be able to:

- Use monitoring and analytics to gain cost insights
- Maximize the efficiency of your cloud environment
- Provision resources that are optimized for cost

Prerequisites

- Experience building or operating solutions by using core infrastructure technology such as data storage, compute, and networking
- Experience building or operating technology systems to solve business problems

---

Next unit: Plan and estimate your Azure costs

Continue

200 XP

Plan and estimate your Azure costs

12 minutes

Whether your organization wants to build a new application on Azure, or you're looking to move an entire datacenter to the cloud, estimating costs is a key part of your planning process to ensure a successful project. For example, how do you know which services to select, or which service tier or virtual machine (VM) size to choose? Do you provision VMs for your workload, or do you take advantage of higher-level services that can reduce operational costs?

Proper planning is incredibly important to any cloud project. Let's look at what you need to consider.

Capture requirements

Before you start any cloud project, take time to plan properly. That's especially important when you're considering costs.

Start by identifying the stakeholders for the project. This should include the business teams that are driving the organizational outcomes. It should also include the technical teams involved in the project. Bring everyone together and foster a culture of transparency. All teams involved in the project should have visibility into the decisions that will affect costs.

Identify the business and technical requirements of your project:

- Business requirements might be an API to enable partner communications or a reporting interface for the accounting department to view financial transactions.
- Technical requirements might be the ability to store relational data, or the ability for users to use a personal identity to access applications.

Both of these requirements will affect the overall cost of the project. They'll also

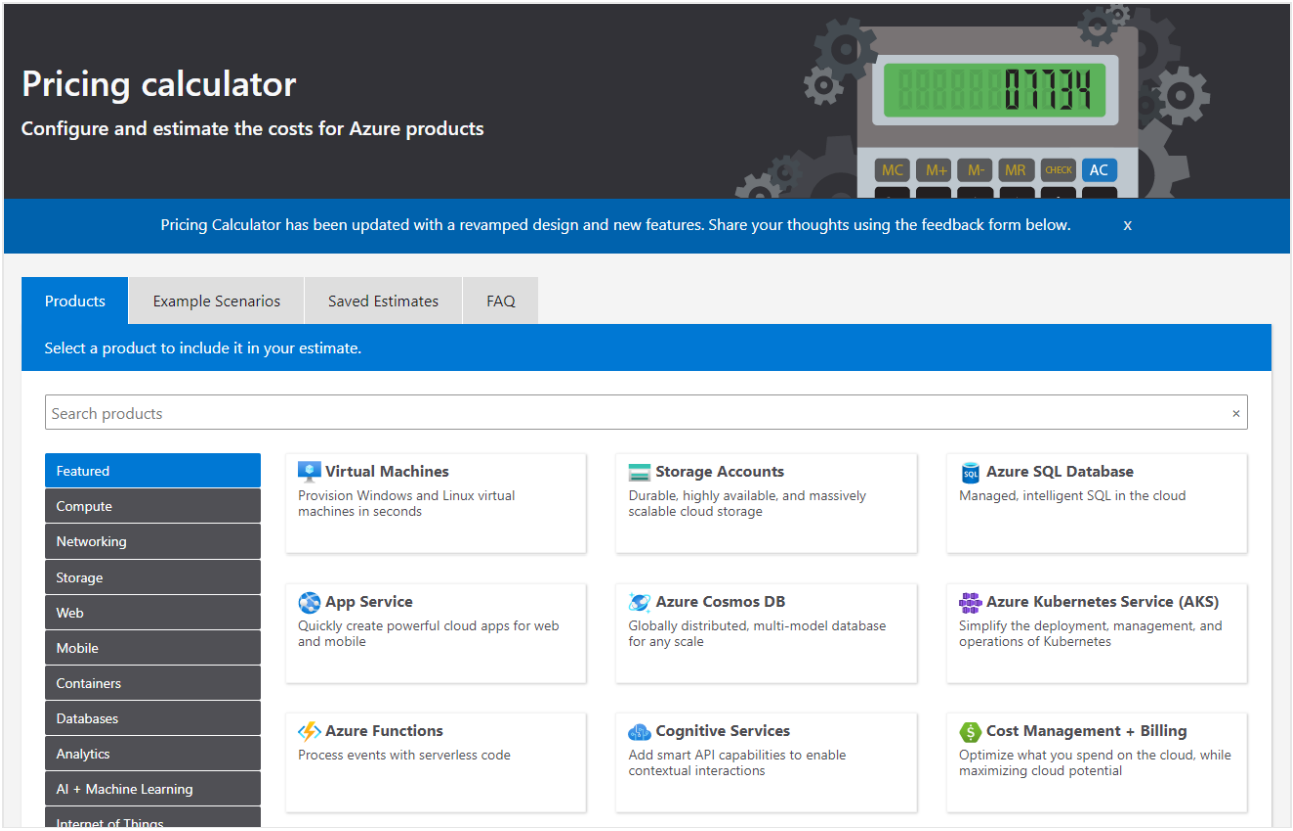
influence your selection of services.

After you have identified your requirements, you'll want to define the workloads that are in scope to use cloud services, and identify the services and resources you'll use. This effort will include evaluating and comparing service options in order to select the best services to meet your requirements.

When you have listed all of the requirements, services, and resources for your project, you can begin to estimate your costs.

Estimate costs

With your list of services captured, you can use the Azure Pricing Calculator to create estimates of the cost of your application. You can use the calculator to create, save, and share estimates for all Azure services.



As part of your cost estimation, it's also important to understand the subscription and billing models that are available on Azure. Two of the most common models are *pay-as-you-go* and *enterprise agreement*:

- *Pay-as-you-go* subscriptions give you the flexibility to purchase and use the

services you need, with the advantage of having no up-front commitments.

- *Enterprise agreements* let organizations take advantage of discounts through up-front commitments. These agreements allow organizations to centralize their Azure costs and billing. They can include other Microsoft services, such as Microsoft 365.

There are additional billing models. Each gives you access to the full suite of Azure services with the flexibility to purchase only what you need, when you need it.

No architecture is static, so you'll want your estimates to include any investments that you might be planning to make in the future. Taking the extra time to plan effectively for your application's future will give you better visibility into the long-term costs for your project, even if your costs might shift over time.

Evolving your architecture can reduce resources costs, such as moving from virtual machines to app services. It can also reduce operational costs by requiring less downtime for maintenance.

Organize resources for cost awareness

It's also important to set up an organizational framework to enable the control, reporting, and attribution of costs throughout your environment.

Use Azure Policy to create limitations for the size or tier of resources that can be provisioned. For example, you can prevent users from creating virtual machines from the G or M series, which have higher costs. If a need arises for VMs of those sizes, you can flow requests for those VMs through an exception process, where individual scenarios can be reviewed and approved on a case-by-case basis. These types of policies will prevent your organization from facing unexpected large bills from your users creating resources that are larger than your projects need.

Let your users view reports and billing as needed by creating roles that allow them to view services such as Microsoft Cost Management. Allowing your users to view costs will help them see the impact of their business decisions. It also provides for transparency across the organization with respect to cloud resource costs.

Organize your resources into resource groups or subscriptions. They can serve as

boundaries for projects, business units, or services. You can also use Azure Policy to enforce the tagging of resources. Subscriptions, resource groups, and tags are exposed in billing reports. These reports will allow you to account for the usage of resources by product, business unit, or project.

Budget for education

Educating your engineers, developers, and users is an important piece of a successful cloud project. The cloud is a transformational shift for an organization, and is an ever-changing set of services and technologies. Your organization will need to ensure that your staff is properly trained to build and maintain resources on Azure. Microsoft provides full product documentation on Microsoft Docs, and it provides self-paced learning on Microsoft Learn. Microsoft also has a large network of partners to deliver in-person and custom training for your staff.

Identify both the initial and ongoing training needs for your organization. Include this as part of the cost of your project.

Check your knowledge

1. Which of the following tools can you use to estimate costs for a new application you are deploying on Azure?

- ○ Azure Resource Manager
- ○ Azure Policy
- ○ Microsoft Cost Management
- ○ Microsoft Pricing Calculator

2. Which of the following would be considered a technical requirement for your application that would affect Azure resource costs?

- ○ The ability for users to cancel orders.
- ○ The ability for customers to upload and store image files in your application.

- ○ Sending notifications to procurement when inventory is low.
- ○ Providing a form for customers to initiate a product return.

Check your answers

200 XP

Provision with optimization

12 minutes

When provisioning resources, you'd ideally make them as efficient as possible from the start. Resource demands and technical requirements can change over time, but if you start with a workload that was optimized for cost when you initially designed it, it will set you up for success down the line. Let's look at some ways to provision your resources with cost optimization in mind.

Select appropriate service tiers and sizes

When you're provisioning resources on the cloud, selecting the right SKU or tier will have a direct impact on the Azure service's capabilities, capacity, and performance. This selection is tied directly to cost. Carefully evaluate the workload requirements for your application and select the SKU or tier that matches your resource requirements.

There's a wide variety of virtual machine types to choose from when you're provisioning for VM-based workloads. Each VM SKU comes with an assigned amount of CPU, memory, and storage. Assess the resource requirements for your workload and select the VM SKU that most closely matches your needs.

Provisioning VM sizes can often be challenging. You might be deploying for your maximum workload, even though your application needs that capacity for only a portion of its running time. Choosing a VM size is not a permanent decision. You can modify your VM size at any time, but in most cases it will require you to restart your VM.

Pay only for consumption

Many cloud services provide a consumption billing model. With consumption models, you pay for only the amount of transactions, CPU time, or run time of your

application. This can bring cost savings and efficiency to your application, because you aren't paying for the resources to run your application when it's not being used. Let's look at a few examples of Azure services that have a consumption cost model:

- **Azure Functions** is an event-driven, serverless compute platform that provides a consumption plan. When you're using the consumption plan, you're charged for compute resources only when your functions are running. Billing is based on the number of executions, the length of time running, and the amount of memory used. As an added benefit, your function scales automatically. Instances of the Azure Functions host are dynamically added and removed based on the number of incoming events. Function execution times out after a configurable period of time.

- **Azure Logic Apps** is a service that helps you create automated integration workflows in the cloud. Logic Apps provides a consumption tier where you only pay per execution of a connector.

- **Azure SQL Database** is service that allows you to store relational data in the cloud. Azure SQL Database has a serverless tier where you can reduce your costs by pausing the database when it's not in use. Azure SQL Database serverless is price-performance optimized for single databases with intermittent, unpredictable usage patterns that can afford some delay in compute warm-up after idle usage periods.

- **Azure API Management** is a service that provides centralized API administration, proxy, and deployment. API Management has a consumption tier that bills per execution, and will scale out automatically as requests change over time. The consumption tier enables the service to be used in a serverless fashion, with instant provisioning, automated scaling, built-in high availability, and pay-per-action pricing.

Use spot instances for low-priority workloads

You can use spot VMs to take advantage of unused capacity on Azure at a significant cost savings. At any point when Azure needs the capacity back, the Azure infrastructure will evict spot VMs. Spot VMs are great for workloads that can handle interruptions like batch processing jobs, development/test environments, and large

compute workloads.

Take advantage of reserved instances

Azure reservations help you save money by committing to one-year or three-year plans for multiple products. Committing to one of these plans gets you a discount on the resources you use. Reservations can reduce your resource costs up to 72 percent on pay-as-you-go prices. Reservations provide a billing discount and don't affect the runtime state of your resources. After you purchase a reservation, the discount automatically applies to matching resources.

Reservations are available for services such as:

- Windows and Linux virtual machines
- Azure SQL Database
- Azure Cosmos DB
- Azure Synapse Analytics
- Azure Storage

If you have consistent resource usage that supports reservations, buying a reservation provides you the option of reducing your costs. For example, when you continuously run instances of a service without a reservation, you're charged at pay-as-you-go rates. When you buy a reservation, you immediately get the reservation discount. The resources are no longer charged at the pay-as-you-go rates.

Use managed services when possible

Whenever possible, take advantage of combining lower resource costs and lower operational costs by using managed services. These services come with lower operational costs because you don't need to patch and manage the underlying infrastructure and services. Deploying applications on VMs comes with the administration and maintenance of the operating system, as well as any layered software.

Azure SQL Database is a great example of a managed service. You can deploy a single or pooled database, or a managed instance, and each of these is fully managed. You don't need to patch the underlying database software, and operational items like

backup are built in and provided for you.

Azure App Service is another example of a managed service that is designed to host web applications. Rather than deploying and managing VMs to host your web applications, you can deploy your applications directly to App Service and dramatically reduce the amount of effort that's required to maintain infrastructure.

Check your knowledge

1. Which of the following scenarios would benefit from the use of reserved instances?

- ○ You have a workload that needs a VPN connection between Azure and an on-premises datacenter.
- ○ You have a website that needs the flexibility to scale from hundreds to millions of users at any point in time.
- ○ You have an enterprise resource planning (ERP) system with consistent resource utilization.
- ○ You have a small administrative task that uses PowerShell that you want to run automatically.

2. Which of the following scenarios would be well suited for spot instances?

- ○ A database for a business-critical financial application.
- ○ Virtual desktops for employees.
- ○ A system that enables patients to access health records.
- ○ A system that processes batches of data that is sent from partners.

Check your answers

200 XP

Use monitoring and analytics to gain cost insights

8 minutes

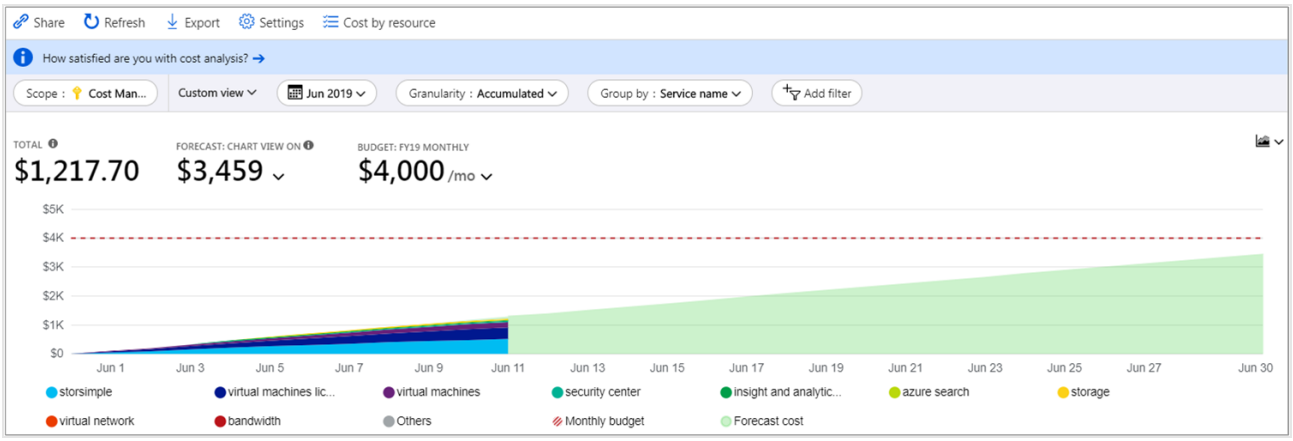
You've deployed your application by using infrastructure and services that are as cost-effective as possible. But what do you do when your business, customer demand, or application changes? How do you ensure that your costs aren't growing out of control relative to the resources that are required to run them? How do you detect areas to improve efficiency in your environment? Architectures aren't static, resource demands will shift over time, and cloud services will evolve to introduce new features and cost savings.

Track your cloud spend

To make intelligent decisions, you need data. By analyzing where your money is going, you can compare your costs to your utilization to discover where you might have waste within your environment.

An export of your billing data is available at any time. By using your billing data, you can track where your costs are going and how they're allocated across your resources. One challenge for you is that the billing data shows your costs, but not your utilization. You'll have data that indicates you're paying for a large VM, but how much are you actually using it?

Microsoft Cost Management gives you insights into where your spend is going, as well as underutilized resources. Microsoft Cost Management tracks your total spend, cost by service, and cost over time. You can drill down into resource types and instances. You can also break down your costs by organization or cost center by tagging resources with those categories.



Azure Advisor also has a cost component that:

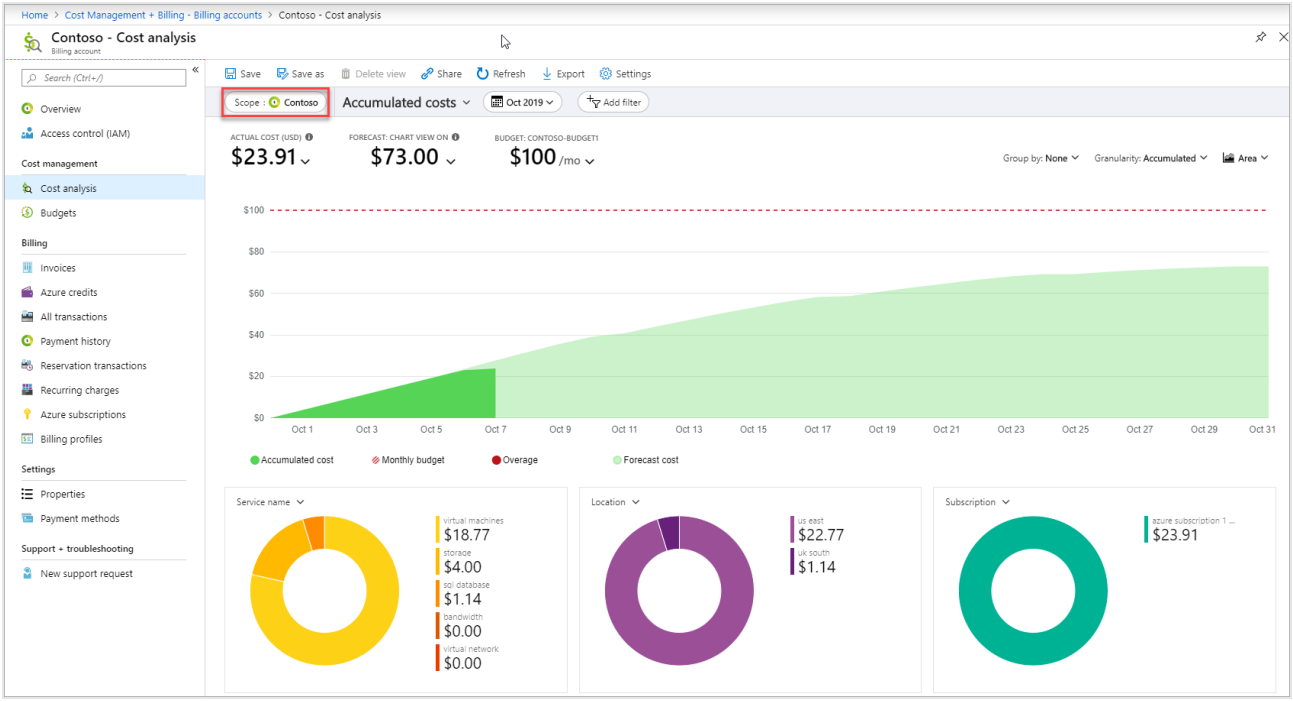
- Recommends VM resizing when necessary
- Identifies unused Azure ExpressRoute circuits and idle virtual network gateways
- Advises when to consider buying reserved instances because that might be more cost-effective than using pay-as-you-go instances

Azure Advisor makes additional recommendations in the areas of performance, high availability, and security.

The important part of optimization is to take time to review your spend and evaluate where your money is going. Effective analysis will help you identify areas of inefficiency and ensure you're operating as cost-effectively as possible.

Conduct cost reviews

After you have your Azure services running, you should regularly check your costs to track your Azure spending. You can use cost analysis to understand where the costs originated for your Azure usage.



Take time as an organization to regularly meet and review billing and expenditures that are related to cloud services. Review the respective expenditures with the technical and business stakeholders for each application. This brings increased visibility to the costs that are associated with an application and the decisions made from a cost perspective.

Respond to cost alerts

One of the key features of Microsoft Cost Management is the ability to configure alerts that are based on spending. These alerts can provide immediate visibility into spending that might be exceeding your budget. You can then take steps to address these costs. There are three types of cost alerts:

- *Budget alerts* notify you when spending, based on usage or cost, reaches or exceeds the amount defined in the alert condition of the budget. Budgets in Microsoft Cost Management help you plan for and drive organizational accountability.

  With budgets, you can account for the Azure services that you consume or subscribe to during a specific period. They help you to proactively inform others about their spending and to monitor how spending progresses over time. When you exceed the budget thresholds that you've created, alerts can be sent to the

appropriate teams. You can set budgets at varying levels, from resource groups to subscriptions to enterprise agreements.

- *Credit alerts* notify you when your Azure credit monetary commitments are consumed. Monetary commitments are for organizations with enterprise agreements.
- *Department spending quota alerts* notify you when department spending reaches a fixed threshold of the quota. You can configure spending quotas in the Azure Enterprise Agreement portal. When you meet a threshold, an email is sent to department owners and a notification appears in cost alerts.

Report anomalies

When an anomaly in spending is identified through your data collection, cost reviews, or cost alerts, you should report it to the necessary stakeholders. Active engagement on cost can ensure that you identify a potential for cost overrun before it becomes problematic. Transparency with stakeholders is important so they can fully understand any technical or business decisions that caused abnormal cloud costs.

Check your knowledge

1. Which of the following would be an example of a cost anomaly that you should investigate further?

- ○ A 1% increase in compute costs from the previous month.
- ○ A decrease in compute costs after the decommission of a website.
- ○ A 5% increase in storage costs after the migration of a large database.
- ○ A 25% increase in ExpressRoute circuit utilization from the previous month.

2. Which of the following is not used to monitor your costs?

- ○ Microsoft Cost Management
- ○ Azure Functions
- ○ Azure Advisor
- ○ Azure billing reports

Check your answers

200 XP

Maximize efficiency of cloud spend

15 minutes

You're a solution architect, and your organization, Lamna Healthcare, has moved its workloads to the cloud. Recently, the bill for these resources and workflows has increased more than Lamna had anticipated. You've been asked to determine whether the increase is the result of natural, efficient growth, or whether the cost can be reduced by improving efficiency with the organization's cloud resources.

How the cloud changes your expenses

One of the differences between the public cloud and on-premises infrastructure is how you pay for the services that you use.

In an on-premises datacenter, hardware procurement is a long process. Physical hardware is sized for maximum capacity. Some of the costs, such as computer power and storage space, can be hidden from the business units that are consuming those resources. Purchasing physical infrastructure ties up investments in long-term assets, which hinders your ability to be agile with your resources.

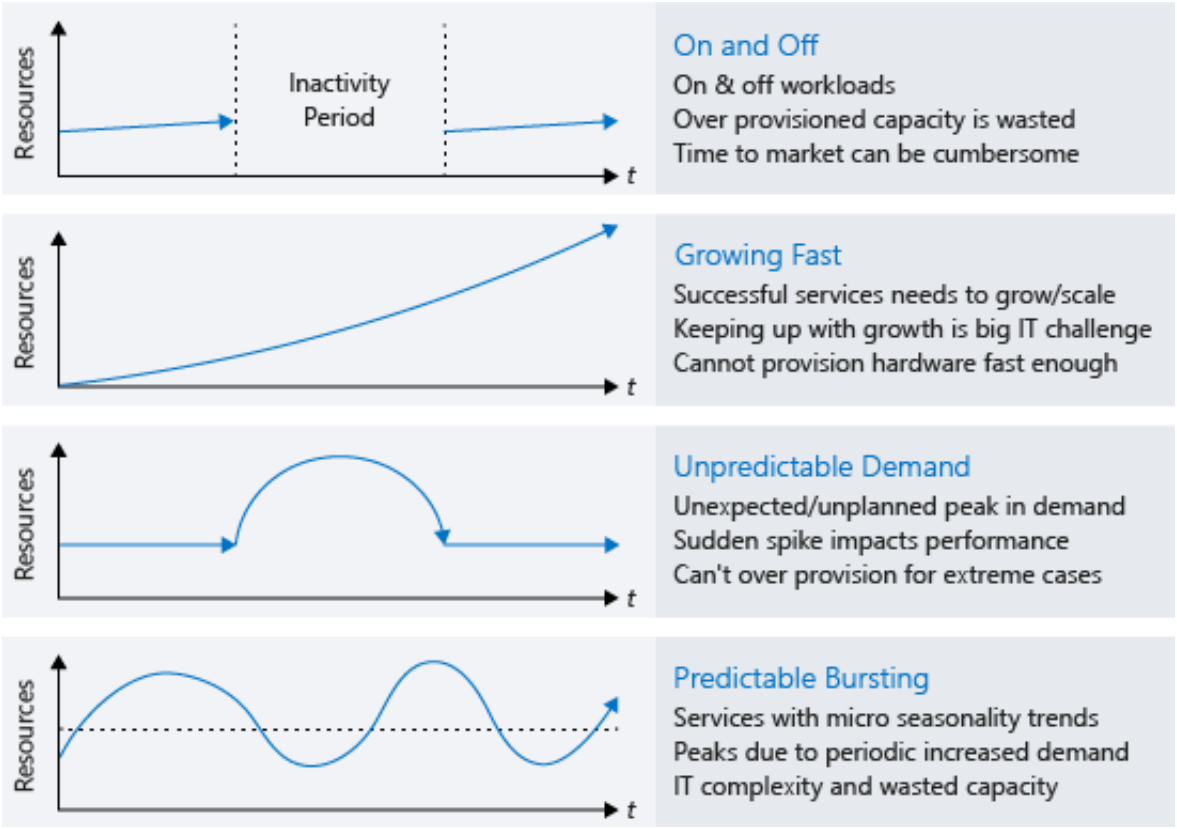
Shifting to the cloud replaces the burgeoning costs of maintaining physical infrastructure with a pay-for-what-you-use cost model. You no longer need to tie up investments in physical assets. If your resource requirements change, you can respond by adding, moving, or removing resources.

Workloads vary between and within services, demand can be unpredictable, and your growth patterns shift over time. Because you pay for only what you use in the cloud, your cost structure can move in sync with the changes in resources.

Cloud infrastructure can handle fluctuating resource-usage scenarios. Resources that have significant periods of inactivity can be shut down when not in use, and then not incur any cost at all. Resource allocation can grow automatically with a successful

service as demand increases, rather than having to wait for the next procurement cycle. Additional resources can be dynamically added and removed to respond to predictable and unpredictable bursts of demand.

The following illustration shows why an on-premises infrastructure can't handle all of these fluctuating scenarios.



In an efficient architecture, resources are provisioned to match the demand. If a virtual machine is less than 10 percent utilized the majority of the time, you're wasting resources, both in compute and cost. Conversely, a virtual machine that is running 90 percent utilized is using the majority of the available resources, and is an efficient use of money.

Running a system at 100 percent utilization runs the risk of introducing performance issues. It's important to ensure that maximizing efficiency doesn't negatively affect the performance of your system. Demand is rarely constant, so adjusting resources when possible to match demand is important to ensure efficiency.

Optimize IaaS costs

When you're using infrastructure as a service (IaaS) resources such as virtual machines as part of your solution, the cost associated with VMs is often the biggest portion of your spending. The compute costs are typically your largest expense, followed by storage costs. Taking time to optimize pay-for-what-you-use resources can have a large impact on the size of your monthly bill.

Let's take a look at best practices to reduce your compute and storage costs.

Compute

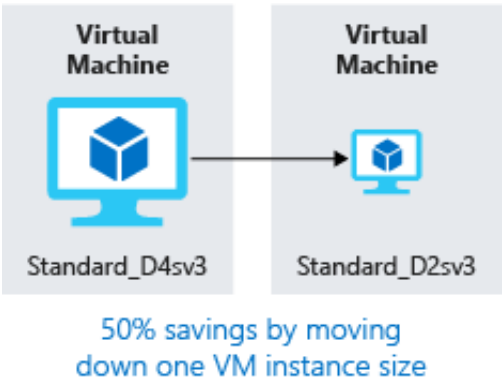
A few options are available to achieve cost savings for virtual machines:

- Choose a smaller size for the virtual machine instance
- Reduce the number of hours a virtual machine runs
- Use discounts for the compute costs

Rightsize virtual machines

Rightsizing a virtual machine is the process of matching virtual machine sizes with the respective requirements for resource demand. If a VM is running 25 percent idle, reducing the size of the VM will immediately reduce your cost. Virtual machine costs are linear within an instance family; each next size larger will double your cost. Conversely, reducing a VM by a single instance size will reduce your cost by half.

The following illustration shows a 50 percent savings achieved by moving one size down within the same series.



Azure Advisor identifies which virtual machines are underutilized. Azure Advisor monitors your virtual machine usage for 14 days, then it identifies any underutilized

virtual machines. Virtual machines with a CPU utilization of 5 percent or less and network usage of 7 MB or less for four or more days are considered underutilized.

Implement shutdown schedules for virtual machines

If you have VM workloads that are used only periodically, but are running continuously, you're wasting money. You can shut down these VMs when they're not in use, which saves your compute costs while the VM is deallocated. For example, a development environment is a good candidate for shutdown during your organization's off hours, because development generally happens only during business hours.

You have several options to deallocate a VM. For example:

- You can use Azure Automation to run your VMs only during times that your workloads require
- You can use the auto-shutdown feature on a virtual machine to schedule a one-off automated shutdown
- You can manually stop a VM in the Azure portal

You should always use the Azure controls to stop your VMs. Shutting down the OS from inside a VM doesn't deallocate its Azure resource, so you'll continue to accrue costs.

Apply compute cost discounts

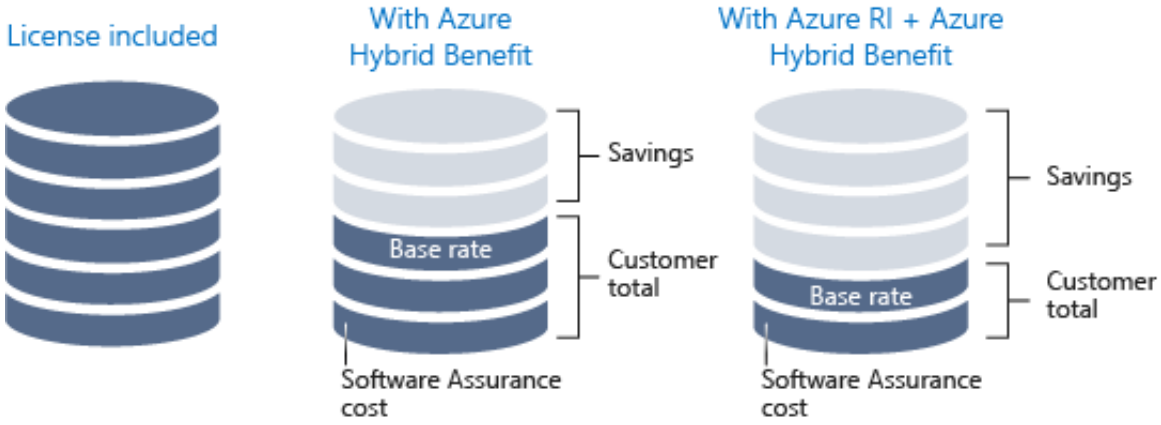
Azure Hybrid Benefit offers an additional way to optimize the costs of your Windows Server and SQL Server instances. It lets you use your licenses for your on-premises computers running Windows Server or SQL Server with Software Assurance as a discount toward the compute cost of these VMs. You can then reduce or eliminate the costs for Windows Server and SQL Server on enabled instances.

Some virtual machines need to be up and running all the time. Perhaps you have a web application server farm for a production workload. Or maybe you have a domain controller that supports various servers on a virtual network. If you know with certainty that these virtual machines will run continuously throughout the coming year or longer, you can reduce your costs even more by purchasing a reserved

instance.

Azure Reserved Virtual Machine Instances (Azure RI) lets you purchase compute capacity for a one-year or three-year commitment. It offers you significant savings — up to 72 percent — when compared to pay-as-you-go compute resources.

The following illustration shows savings achieved when you combine your on-premises licenses with Azure Hybrid Benefit. It also shows savings achieved when you combine your on-premises licenses with both Azure Reserved Virtual Machine Instances and Azure Hybrid Benefit.



Cost optimization for VM disk storage

For workloads that don't require high reliability and performance disks, you can use the reduced-cost standard storage. For example, you might choose to use standard storage for your development and test environments that are not required to be an identical match for your production workloads.

Disks that aren't associated with a VM still incur storage costs, so you should make sure you don't have any orphaned disks remaining in your environment. If you've removed a VM but not its associated disks, you can reduce your storage costs by identifying and removing these orphaned disks from your environment.

You should also make sure that you don't have any orphaned snapshots remaining in your environment. Pricing for snapshots is lower than pricing for the disks themselves, but it's still a good practice to eliminate costs for unnecessary resources.

Optimize PaaS costs

Platform as a service (PaaS) services are typically optimized for costs over IaaS services. But there are opportunities to identify waste and optimize for minimal costs in your PaaS services as well. Let's look at ways to reduce Azure SQL Database and Azure Blob storage costs.

Optimize Azure SQL Database costs

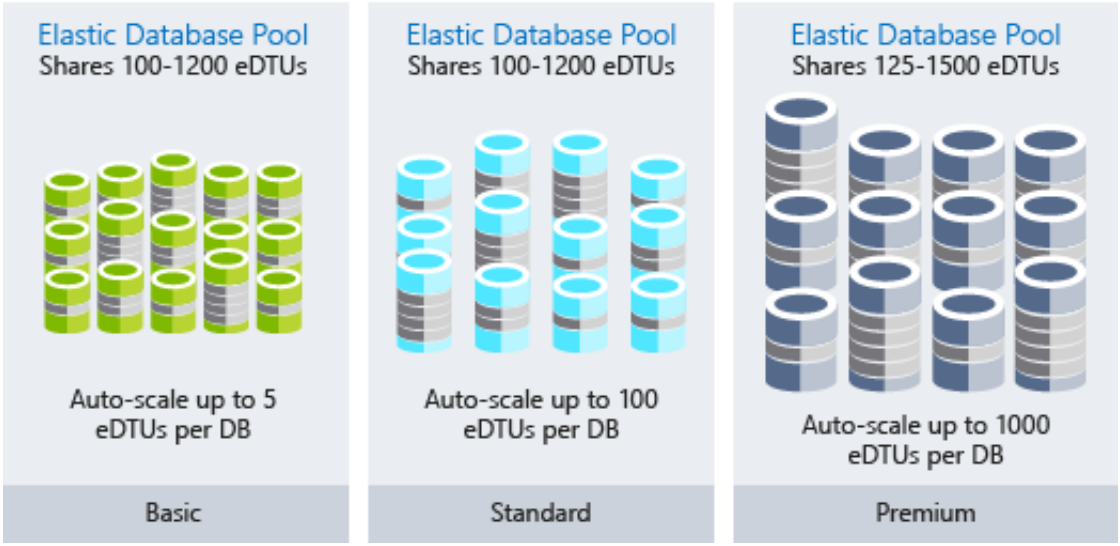
When creating an Azure SQL database, you have to select a server and decide on a performance tier. Each tier provides a performance level either in database transaction units (DTUs) or virtual cores (vCores).

For steady database loads, it's easy to optimize by selecting the appropriate tier size for the performance needs. But what if your database has unpredictable bursts or spikes in activity? When you're dealing with unpredictable workloads, elastic pools can help you reduce your costs.

SQL Database elastic pools are a simple, cost-effective solution for managing and scaling several databases that have varying and unpredictable usage demands. The databases in an elastic pool are on a single Azure SQL Database server, and share a set number of resources at a set price. Pools are well suited for a large number of databases with specific utilization patterns. For a given database, this pattern is characterized by low average utilization with relatively infrequent utilization spikes.

The more databases you can add to a pool, the greater your savings become. The following illustration shows the capabilities of the three types of elastic database pools:

- *Basic* autoscales up to 5 eDTUs per database
- *Standard* autoscales up to 100 eDTUs per database
- *Premium* autoscales up to 1,000 eDTUs per database



Elastic pools are a great way to spread costs across multiple databases. They can make a significant impact on reducing your Azure SQL Database costs.

Optimize Blob Storage costs

Blob Storage is a cost-effective way to store data, but as the amount of data grows, your bill can benefit from optimizing how the data is stored.

Azure Storage offers three tiers for blob object storage:

- **Hot access tier**: Highest storage costs but lowest access costs. This tier is optimized for storing data that's accessed often.
- **Cool access tier**: Lower storage costs and higher access costs compared to hot storage. This tier is optimized for storing data that's infrequently accessed and stored for at least 30 days.
- **Archive access tier**: Lowest storage cost and highest data-retrieval cost compared to hot and cool storage. This tier is optimized for storing data that is rarely accessed and stored for at least 180 days, with flexible latency requirements (for example, several hours of retrieval latency).

Consumption pricing models

Moving to PaaS services can take the pay-as-you-go model even further into a true consumption pricing model. Services such as Azure Functions have the ability to use

consumption plans.

When you're using a consumption plan, instances of the Azure Functions host are dynamically added and removed based on the number of incoming events. This serverless plan scales automatically, and you're charged for compute resources only when your functions are running. On a consumption plan, a function execution times out after a configurable period of time. Billing is based on the number of executions, the length of execution time, and the amount of memory used. Billing is aggregated across all functions within a function app.

Moving to services that use a consumption pricing model can bring a new approach to cost savings into your architecture.

Check your knowledge

1. Your business stores PDF copies of all purchase orders. These files are accessed infrequently after their initial upload, and there is no time criticality associated with their retrieval. Which of the following storage tiers would be the best choice to reduce costs for long-term storage?

- ○ Hot access tier
- ○ Cool access tier
- ○ Archive access tier

2. Which of the following databases would be well suited for elastic pools?

- ○ An ERP database that has consistent usage around the clock.
- ○ Customer databases that have usage during the same period of time.
- ○ Customer databases that are spread across time zones and have periods of intermittent usage.

Check your answers

100 XP

Introduction

2 minutes

When you analyze your cloud-based solutions, you should be asking yourself a series of questions that will help you determine whether your applications are following operational best practices. For example: how operationally mature is your cloud environment? Are you incorporating modern practices such as DevOps into the way your organization operates? Are you able to assess the health of your infrastructure? Are adverse events visible and actionable? Are you provisioning resources manually, or have you automated them to reduce risk and increase efficiency?

In this module, you'll learn about the *Operational Excellence* pillar of the Azure Well-Architected Framework, which will allow you to answer these types of questions and improve the operations of your Azure cloud deployments. The concepts discussed in this module are not all-inclusive, but they represent some of the important considerations when building a solution on the cloud. For more details on the Azure Well-Architected Framework, visit the Azure Architecture Center as you start planning and designing your architecture.

Learning objectives

By the end of this module, you'll be able to:

- Apply modern practices to design, build, and orchestrate resources on Azure
- Gain operational insights by using monitoring and analytics
- Reduce effort and error by using automation
- Identify issues and improve quality in your application by using tests

Prerequisites

- Experience building or operating solutions using core infrastructure technology such as data storage, compute, and networking
- Experience building or operating technology systems to solve business problems

200 XP

Design, build, and orchestrate with modern practices

15 minutes

Development and operational practices have evolved over the years to become more seamless and integrated. Modern practices involve organizational shifts and adopting new tooling in order to improve an organization's operational agility. In this unit, you'll learn some of the key concepts to improving your organization's ability to build and deploy applications.

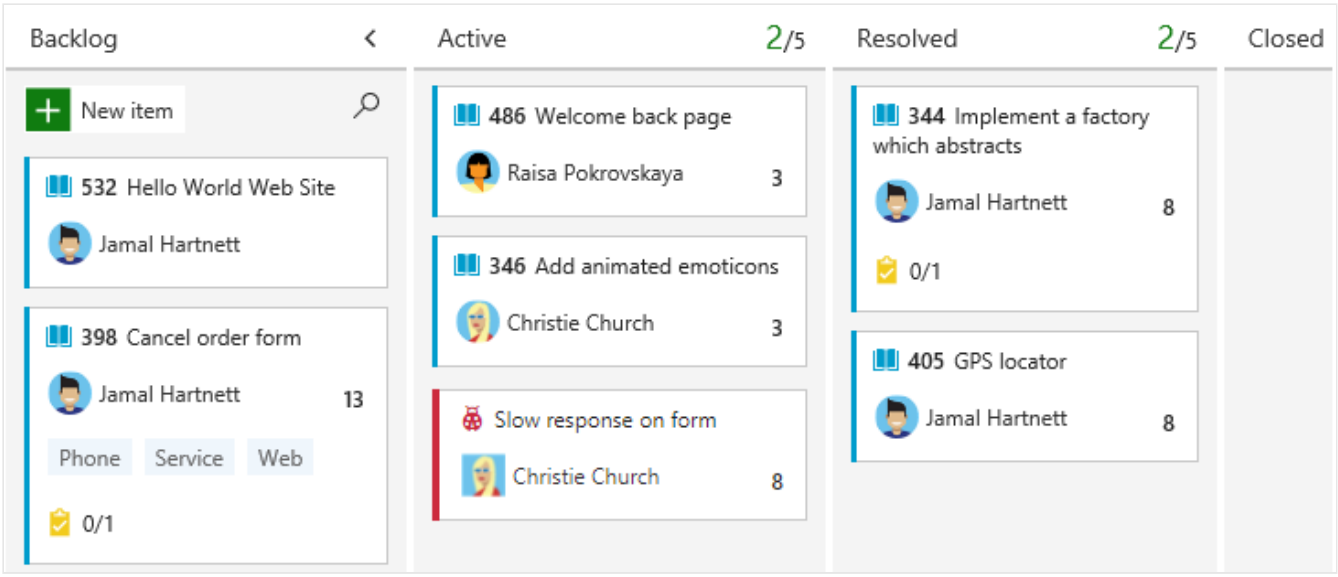
DevOps

DevOps is the union of people, processes, and products to enable continuous delivery of value to end users. DevOps focuses on bringing the development and operations functions together, and breaking down the existing barriers between them. This combination creates multidisciplinary teams that work together with shared and efficient practices and tools. Essential DevOps practices include agile planning, continuous integration, continuous delivery, and monitoring of applications.

The DevOps culture stresses small, multidisciplinary teams that work autonomously and take collective accountability for how end users will experience their software. DevOps teams apply agile practices and include operations in each team's responsibilities. Teams work in small batches that focus on improving the end-to-end delivery of customer value, and they strive to eliminate waste and impediments along the way. There are no silos and no blame games, because each team is mutually accountable.

There are several services and tools that are available from Microsoft to help an organization adopt and develop DevOps practices. Azure DevOps is a suite of products and tools that teams adopting DevOps practices can use to plan, develop, deliver, and operate their solutions.

Azure Boards is a part of Azure DevOps that helps teams plan and track work. Azure Boards has modern agile tools like Kanban boards, backlogs, dashboards, and scrum boards to help your team to have greater visibility into the work that's planned and what's been delivered.



GitHub is the most widely used platform to build, deliver, and share software. At its core, GitHub is a version-control platform that allows a global community of individuals and teams to collaborate on software development projects, whether these projects are kept private within their organizations or shared through public communities of disparate software developers. GitHub also includes features to build and test deployments, track issues, and create custom workflows in your repositories.

Azure DevOps and GitHub integrate together, and organizations that are operating or adopting a DevOps model often use these services together.

Continuous Integration and Continuous Delivery (CI/CD)

Continuous Integration (CI) is the practice of building and testing code every time a team member commits changes to version control. CI encourages developers to share their code and unit tests by merging their changes into a shared version-control repository after every small task completion. Committing code triggers an automated build system to grab the latest code from the shared repository, and then build, test, and validate the full main branch.

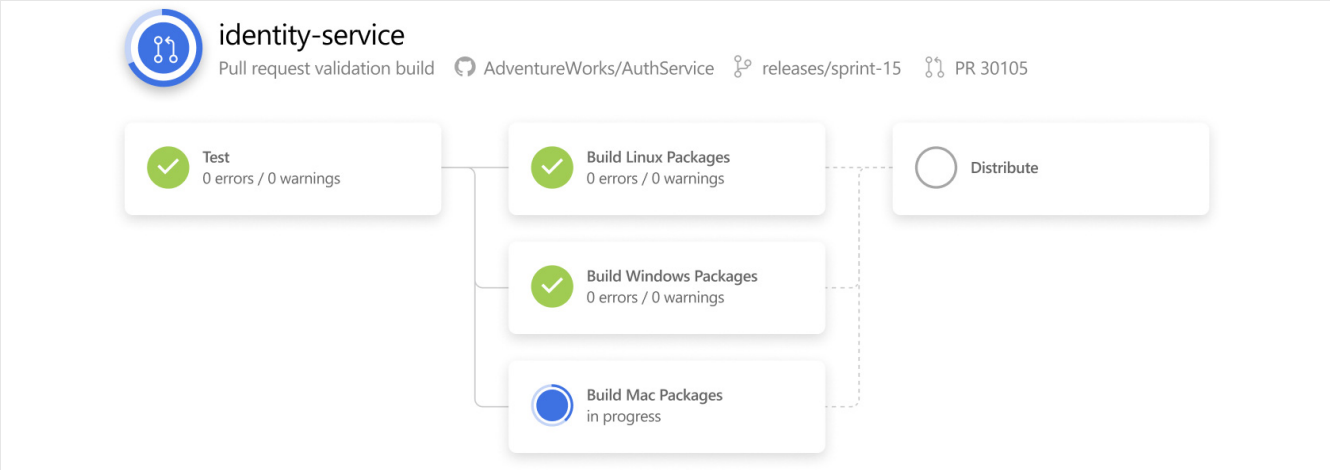
CI helps developers to identify bugs earlier, and it improves software quality since code is checked in, built, and verified more frequently. Instead of working on code for a month and discovering numerous issues when changes are eventually checked in, developers can check in smaller sets of changes and be confident that their code doesn't introduce large volumes of issues into the main branch.

Continuous Delivery (CD) is the process to build, test, configure and deploy from a build environment to a production environment. Multiple testing or staging environments create a release pipeline to automate the creation of infrastructure and deployment of a new build.

Successive environments support progressively longer-running activities of integration, load, and user-acceptance testing.

Continuous integration and continuous delivery are often combined into a single pipeline known as *CI/CD*. Continuous integration starts the continuous delivery process, and the CI/CD pipeline stages changes from each successive environment to the next upon successful completion of the tests that are defined at each stage. As a developer, you can check in code, validate that it passes all tests and introduces no new issues into the main branch, then roll it out to production with the confidence that it will not impact the operation of your production environment.

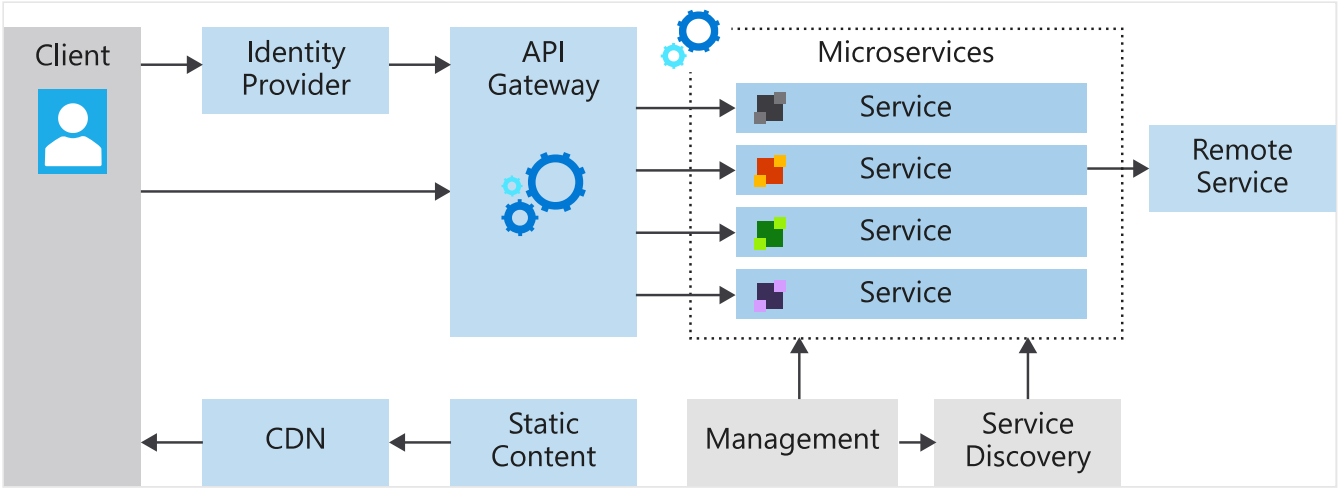
Azure Pipelines is a cloud service you can use to automatically build and test your code project and make it available to others. It works with just about any language or project type, and integrates with GitHub, GitHub Enterprise, Azure Repos, and other version-control systems. Azure Pipelines combines continuous integration (CI) and continuous delivery (CD) to constantly and consistently test and build your code and ship it to any target.



You can also use GitHub Actions to build CI/CD capabilities in your GitHub repositories. With GitHub Actions, you can build workflows that are custom automated processes to build, test, package, release, and deploy code.

Microservices

A microservices architecture consists of services that are small, independent, and loosely coupled. Each service can be deployed and scaled independently. Microservice architectures are often adopted for new applications that are adopting DevOps practices.



A microservice is small enough that a single, small team of developers can write and maintain it. Because services can be deployed independently, a team can update an existing service without rebuilding and redeploying the entire application.

Each service is typically responsible for its own data. Its data structure is isolated, so upgrades or changes to schema aren't dependent on other services. Requests for data are typically handled through APIs, and provide a well-defined and consistent access model. Internal implementation details are hidden from service consumers.

Because each service is independent, they can use different technology stacks, frameworks, and SDKs. It's common to see services rely on REST calls for service-to-service communication by using well-defined APIs instead of RPC or other custom communication methods.

Microservice architectures are technology agnostic, but you often see containers or serverless technologies used for their implementation. Continuous deployment and continuous integration (CI/CD) is frequently used to increase the speed and quality of development activities.

Environment consistency

A key piece of ensuring that you can develop and deploy applications with confidence is by making sure that your environments are consistent between development, test, and production. As your CI/CD processes move your code through your environments, any variation risks introducing areas where testing can fail or overlook defects. Through automation, you can spin up and tear down environments as needed, which can be included as part of your CI/CD processes.

Imagine an environment where you're building a .NET Core application, and your test and production environments are running different versions. Your deployment may succeed in your test environment, but cause issues in your production environment because it's running a different version of your application. Including your environment definitions as part of your

deployment will help ensure that your code is built and deployed on a consistent, end-to-end infrastructure.

Check your knowledge

1. Which of the following is a benefit of incorporating CI/CD into your development processes?

- ○ Decreased confidence when deploying your code to production.
- ◉ Longer time between validation of developer code.

  ✗ **Code is checked in and validated more frequently when using CI/CD.**
- ○ Build processes revert to manual compilation, thereby allowing you to catch potential issues.
- ○ Earlier identification of code defects.

  ✓ **Your developers will be able to identify and resolve code defects earlier in the development cycle.**

2. Which of the following is true of microservice architectures?

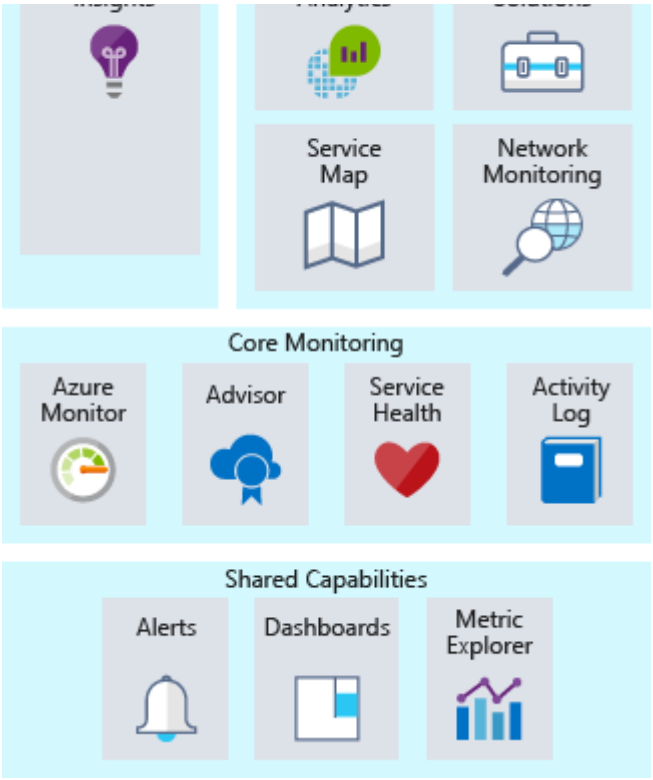
- ○ One large development team builds and maintains the entire code base.
- ○ Communication between services is typically through REST calls.

  ✓ **REST calls are the most common mechanism used for communication.**
- ◉ All services typically run on the same virtual machine.

  ✗ **Microservices are distributed across hardware and cloud services and are not run on the same virtual machine.**
- ○ Microservices require that all services be developed with the same language.

Next unit: Use monitoring and analytics to gain operational insights

Continue >



Core monitoring

Core monitoring provides fundamental, required monitoring across Azure resources. When we talk about fundamental monitoring, you can think of it as monitoring what's happening with your resources at the Azure platform level. This area of focus gives you insight into things like the health of the Azure platform, changes being made to your resources, and performance metrics. Using services from this area gives you the ability to monitor the basic pieces you need to keep your application running.

Azure provides services to give you visibility into four key core monitoring areas: activity logging, the health of services, metrics and diagnostics, and recommendations on best practices.

These services are built into Azure and take little to no configuration to enable and set up. Let's take a closer look at each of these services.

Activity logging

Activity logging is an incredibly important source of information about what's happening with your resources at the Azure platform level. Every change submitted to the Azure platform is tracked in the Azure Activity Log, which gives you the ability to trace any action taken on your

resources. The Activity Log will contain detailed information on activities to help you answer questions like:

- Who has attached a disk to this virtual machine?
- When was this machine shut down?
- Who changed the load balancer configuration?
- Why did the autoscale operation on my virtual machine scale set fail?

Using Activity Log to answer these types of questions will help you troubleshoot issues, track changes, and provide auditing of what's happening in your Azure environment. Activity Log data is only retained for 90 days, although you can archive your data to a storage account, or you can send your data to Azure Log Analytics for longer retention and further analysis.

Health of cloud services

At some point, any system can have issues, and that's true for Azure services as well. Staying informed of the health of Azure services will help you understand if and when an issue that is impacting an Azure service is impacting your environment. What may seem like a localized issue could be the result of a more widespread issue, and Azure Service Health provides this insight. Azure Service Health identifies any issues with Azure services that might affect your application. Service Health also helps you plan for scheduled maintenance.

Metrics and diagnostics

For issues that are more localized in nature, it's important to have visibility into what's happening on your system or service instance. The ability to view metrics and diagnostic information is critical for troubleshooting performance issues and staying notified when something goes wrong. To provide this visibility, Azure services have a common way of showing health, metric, or diagnostic information. Azure Monitor enables core monitoring for Azure services by allowing the collection, aggregation, and visualization of metrics, activity logs, and diagnostic logs.

Metrics are available that provide performance statistics for different resources, and even the operating system inside a virtual machine. You can view this data with one of the explorers in the Azure portal, and create alerts based on these metrics. Azure Monitor provides a fast metrics pipeline, so you should use it for time-critical alerts and notifications.

Recommendations on best practices

When we think of monitoring, we typically think of the current health of a resource. But even when a resource is healthy, there could be adjustments that would result in greater availability,

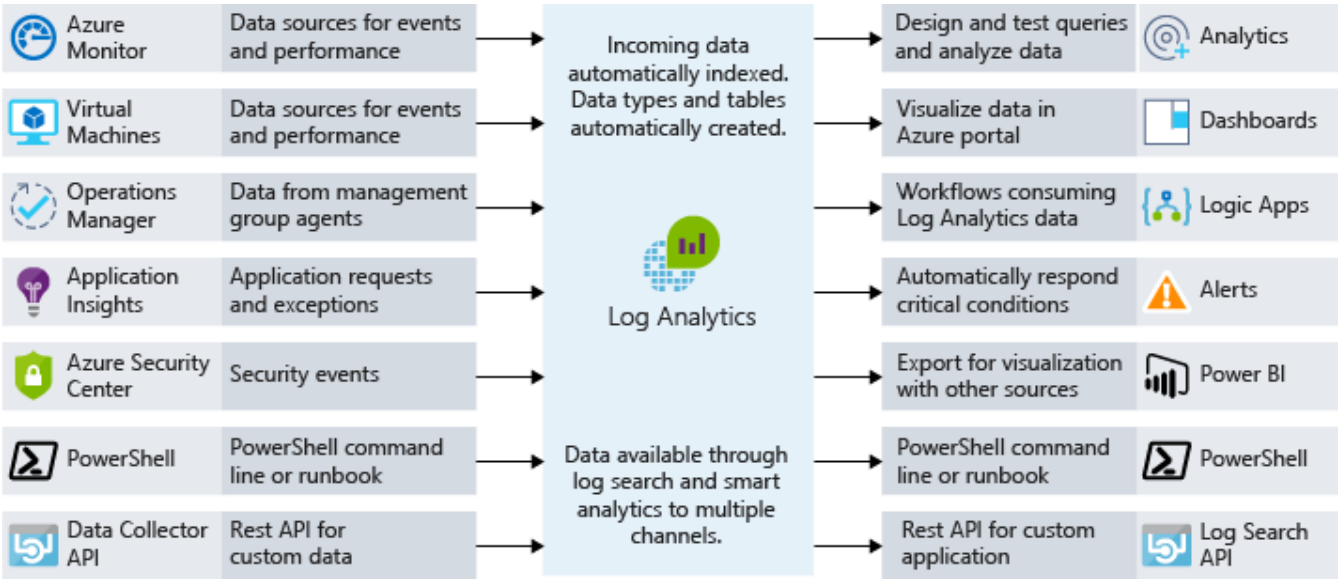
reduced cost, or improved security. Azure Advisor can help by keeping an eye out for potential performance, cost, high availability, or security issues within your resources. Advisor makes personalized recommendations based on resource configuration and telemetry and provides

personalized recommendations based on resource configuration and telemetry and provides you with guidance that most traditional monitoring platforms don't provide.

Deep infrastructure monitoring

While the monitoring components we've covered thus far are great at offering insights, they only give visibility to the Azure platform. For typical IaaS workloads, there's more diagnostic information and metrics to gather from the network or the actual operating systems. For example: Log Analytics can provide deep insights by pulling information from SQL Server to ensure it's properly configured, analyzing free disk space across all the servers in your environment, or visualizing the network dependencies between your systems and services.

When you're designing a monitoring strategy, it's important to include every component in the application chain so you can correlate events across services and resources. You can easily configure services that support Azure Monitor to send their data to a Log Analytics workspace. Virtual machines (both in the cloud and on-premises) can have an agent installed to send data to Log Analytics. You can submit custom data to Log Analytics through the Log Analytics API. The following illustration shows how Log Analytics acts as a central hub for monitoring data; Log Analytics receives monitoring data from your Azure resources and makes that data available to consumers for analysis or visualization.



With this data in Log Analytics, you can query the raw data for troubleshooting, root-cause identification, and auditing purposes. For several known services, like SQL Server and Windows Server Active Directory, there are readily available management solutions that visualize monitoring data and uncover compliance with best practices.

Log Analytics allows you to create queries and interact with other systems based on those queries. The most common example is an alert. Maybe you want to receive an email when a system runs out of disk space, or when a best practice on your SQL Servers is no longer being

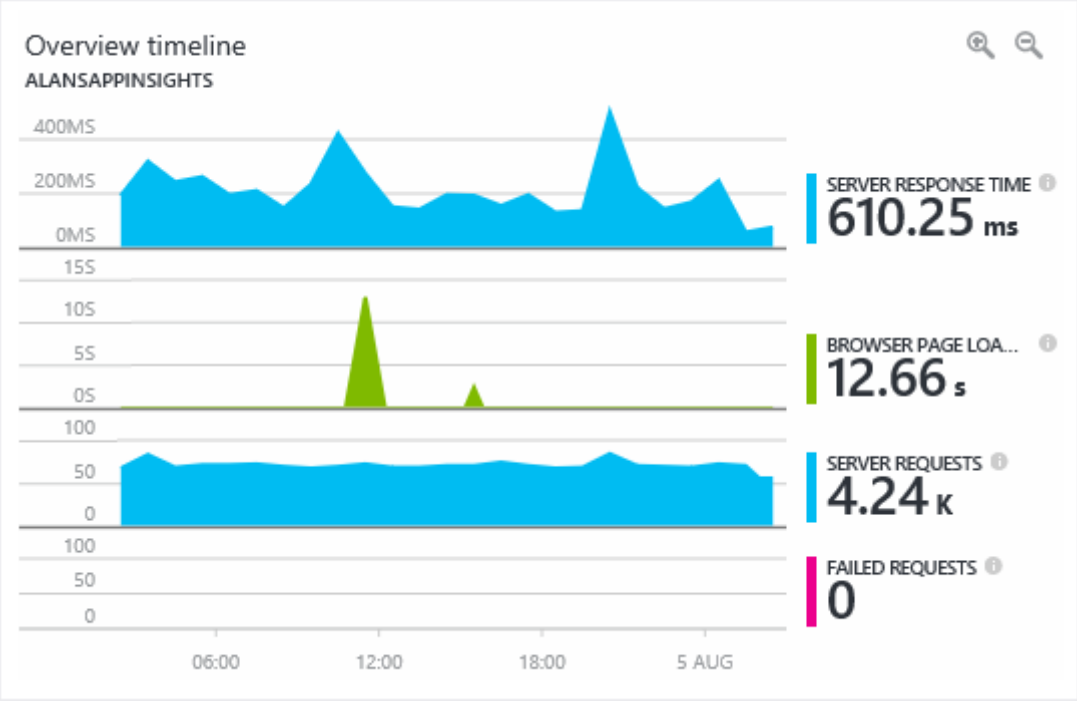
system runs out of disk space, or when a best practice on your SQL servers is no longer being followed. Log Analytics can send alerts, kick off automation, and even hook into custom APIs for things like integration with IT service management (ITSM).

Deep application monitoring

It's important to understand how core services and infrastructure are performing, but you can take your monitoring capabilities even further by looking deep into your applications to identify performance issues, usage trends, and the overall availability of services you develop and depend on. By using an application performance-management tool, you can better detect and diagnose issues that occur within your web apps and services.

Azure Application Insights allows you to do exactly that. Application Insights provides telemetry collection, query, and visualization capabilities. Instrumenting this level of monitoring requires little to no code changes; you only have to install a small instrumentation package into your application. Application Insights is cross platform, supporting .NET, Node.js, or Java.

For instance, the response time for one of your applications might be complex to troubleshoot. For example: is the web server being overloaded? Is a specific SQL query not optimized? Is an API that you're calling performing slower than usual? Application performance monitoring solutions can help uncover the underlying issues that basic metric monitoring can't expose. The following screenshot shows a graphical display of an application's performance details provided by Azure Application Insights.



An application performance-monitoring solution will help you monitor usage, performance, and availability, allowing you to respond to failure much faster, and should be included in any monitoring strategy.

monitoring strategy.

Check your knowledge

1. Which of the following would you use to troubleshoot the performance of API calls in your application?

- ○ Service Health
- ◉ Network Watcher

  ✗ **Azure Network Watcher will help you troubleshoot network performance issues, but will not provide insight into your application.**
- ○ Activity Log
- ○ Application Insights

  ✓ **Application Insights will help you detect and diagnose issues and understand usage for your web apps, services, and API calls in your application.**

2. Which of the following services would you use to identify who deleted a virtual machine?

- ○ Service Health
- ○ Network Watcher
- ◉ Activity Log

  ✓ **Azure Activity Log records all events that have been performed on your resources on the Azure platform, including deletion of resources.**
- ○ Application Insights

Next unit: Use automation to reduce effort and error

Continue >

How are we doing? ☆ ☆ ☆ ☆ ☆

200 XP

Use automation to reduce effort and error

12 minutes

Managing the infrastructure for any type of workload involves configuration tasks. You can do this configuration manually, but manual steps don't scale well; in addition, they can be labor-intensive, prone to error, and inefficient. What if you're assigned to lead a project that requires the deployment of hundreds of systems on Azure? How would you build and configure these resources? How long would this take? Could you ensure that each system is configured properly, with no variance between systems?

By using automation in your architecture design, you can work past these challenges. In this unit, you'll learn about some of the ways you can automate on Azure.

Infrastructure as code

Infrastructure as code (IaC) is the management of infrastructure — such as networks, virtual machines, load balancers, and connection topology — in a descriptive model, using a versioning system that's similar to what's used for source code. When you are creating an application, the same source code will generate the same binary every time it's compiled. In a similar manner, an IaC model generates the same environment every time it's applied. IaC is a key DevOps practice, and it's often used in conjunction with continuous delivery.

IaC evolved to solve the problem of *environment drift*. Without IaC, teams must maintain the settings of individual deployment environments. Over time, each environment becomes a *snowflake* that is increasingly unique, and cannot be reproduced automatically. The administration and infrastructure maintenance of these snowflake environments involves manual processes that are hard to track and that contribute to errors. The resulting inconsistencies between these snowflake environments lead to issues during deployments.

When automating the deployment of services and infrastructure, there are two different approaches you can take: *imperative* and *declarative*.

- With an **imperative** approach, you explicitly state the commands that are executed to produce the outcome you are looking for.
- With a **declarative** approach, you specify what you want the outcome to be instead of specifying how you want it done.

Both approaches are valuable, so there's no wrong choice. What do these different approaches look like on Azure, and how do you use them?

Imperative automation

Let's start with imperative automation. With imperative automation, we're specifying *how* things are to be done. This is typically done programmatically through a scripting language or SDK. For Azure resources, we could use the Azure CLI or Azure PowerShell. Let's take a look at an example that uses the Azure CLI to create a storage account.

Azure CLI Copy

```
az group create \
    --name storage-resource-group \
    --location eastus

az storage account create \
    --name mystorageaccount \
    --resource-group storage-resource-group \
    --kind BlobStorage \
    --access-tier hot
```

In this example, we're specifying how to create these resources: execute a command to create a resource group, then execute another command to create a storage account. We're explicitly telling Azure which commands to run in order to produce the output we need.

With this approach, we're able to fully automate our infrastructure. We can provide areas for input and output, and can ensure that the same commands are executed every time. By automating our resources, we've taken the manual steps out of the process, making resource administration operationally more efficient.

However, there are some downsides to this approach. Scripts to create resources can quickly become complex as the architecture becomes more complex. Error handling and input validation may need to be added to ensure full execution. Commands may change, requiring ongoing maintenance of the scripts.

Declarative automation

With declarative automation, we're specifying *what* we want our result to be, leaving the details of how it's done to the system we're using. On Azure, declarative automation is accomplished through the use of Azure Resource Manager (ARM) templates, which are JSON-structured files that specify what we want created. ARM templates have four sections: *parameters*, *variables*, *resources*, and *outputs*.

- *Parameters* handle input to be used within the template
- *Variables* provide a way to store values for use throughout the template
- *Resources* are the things that are being created
- *Outputs* provide details to the user of what was created

In the example below, we're telling Azure to create a storage account with the names and properties that we specify. The actual steps that are executed behind the scenes to create this storage account are left for Azure to decide.

JSON Copy

```
{
    "$schema": "https://schema.management.azure.com/schemas/2015-01-
01/deploymentTemplate.json#",
    "contentVersion": "1.0.0.0",
    "parameters": {
        "name": {
            "type": "string"
        },
        "location": {
            "type": "string"
        },
        "accountType": {
            "type": "string",
            "defaultValue": "Standard_RAGRS"
        },
        "kind": {
            "type": "string"
        },
        "accessTier": {
            "type": "string"
        },
        "httpsTrafficOnlyEnabled": {
            "type": "bool",
            "defaultValue": true
        }
    },
    "variables": {
    },
    "resources": [
        {
            "apiVersion": "2018-02-01",
            "name": "[parameters('name')]",
            "location": "[parameters('location')]",
            "type": "Microsoft.Storage/storageAccounts",
            "sku": {
                "name": "[parameters('accountType')]"
            },
            "kind": "[parameters('kind')]",
            "properties": {
                "supportsHttpsTrafficOnly": "
[parameters('httpsTrafficOnlyEnabled')]",
```

```
                "accessTier": "[parameters('accessTier')]",
                "encryption": {
                    "services": {
                        "blob": {
                            "enabled": true
                        },
                        "file": {
                            "enabled": true
                        }
                    },
                    "keySource": "Microsoft.Storage"
                }
            },
            "dependsOn": []
        }
    ],
    "outputs": {
        "storageAccountName": {
            "type": "string",
            "value": "[parameters('name')]"
        }
    }
}
```

You can use ARM templates to create and manipulate most services on Azure. You can store your templates in code repositories and inherit all the benefits of using a source-control system, and you can share your templates across environments to ensure that the infrastructure you use for your development environment matches your production environment. ARM templates are a great way to automate deployments, and they help ensure consistency, eliminate deployment misconfigurations, and can increase operational speed.

Automating your infrastructure deployment is a great first step. But when you're deploying virtual machines, there's still more work to do. Let's take a look at a couple of approaches to automating configuration post deployment.

VM images vs. post-deployment configuration

For many virtual machine deployments, your job isn't done when a VM is simply up and running. In many situations, it's quite likely there's additional configuration that you need to take care of before the VM can actually serve its intended purpose. For example: additional disks might need formatting, the VM might need to be joined to a domain, an agent for a management software package might need to be installed, and most likely the actual workload requires installation and configuration as well.

There are two common strategies that you can use for the configuration work, which for all intents and purposes are considered to be part of the configuration process for the VM itself.

Each of these strategies has its advantages and disadvantages.

- **Custom images** are generated by deploying a virtual machine, and then configuring or installing software on that running instance. When everything is configured correctly, the virtual machine can be shut down, and an image is created from the VM. This image can then be used as a base for deploying other new virtual machines. Working with custom images can speed up your overall deployment time, because as soon as the virtual machine is deployed and running, no additional configuration would be needed. If deployment speed is an important factor, custom images are definitely worth exploring.

- **Post-deployment scripting** typically leverages a basic base image, and then relies on scripting or a configuration-management platform to perform the necessary configuration after the VM is deployed. The post-deployment scripting could be done by executing a script on the VM through the Azure Script Extension, or by leveraging a more robust solution such as Azure Automation Desired State Configuration (DSC).

Each of these approaches has a few specific and some shared considerations to keep in mind.

- When using images, you'll need to ensure there's a process to handle image updates, security patches, and inventory management of the images themselves.

- With post-deployment scripting, build times can be extended since the VM can't be added to live workloads until the build is complete. This may not be a significant issue for standalone systems, but when using services that autoscale (such as virtual machine scale sets), this extended build time can impact how quickly you can scale.

- With both approaches, you'll want to ensure that you address configuration drift; as new configuration is rolled out, you'll need to ensure that existing systems are updated accordingly.

Automating resource deployment can be a massive benefit to your environment. The amount of time saved and the number of errors that are reduced can move your operational capabilities to another level.

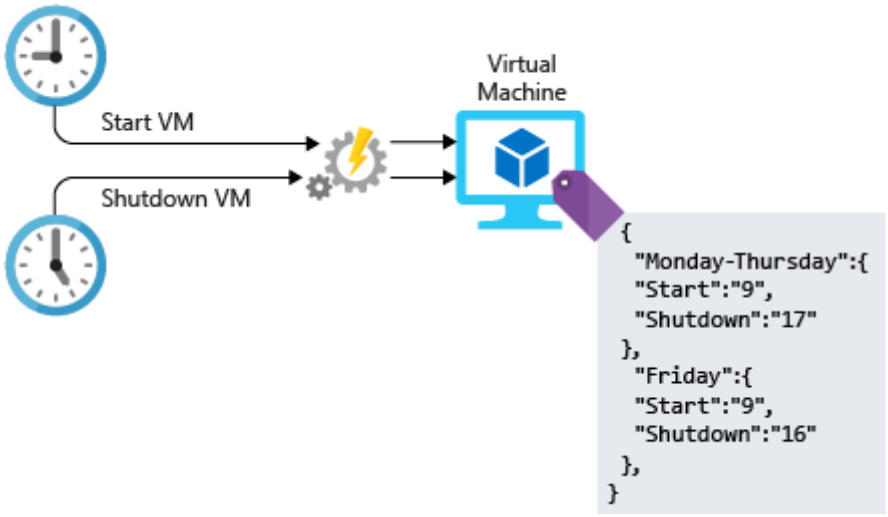
Automation of operational tasks

Once your solutions are up and running, there are ongoing operational activities that you can also automate. Automating these tasks with Azure Automation reduces manual workloads, enables configuration and update management of compute resources, provides a framework for running any type of Azure task, and centralizes shared resources such as schedules, credentials, and certificates.

Examples of this automation might include:

- Periodically searching for orphaned disks
- Installing the latest security patches on VMs
- Searching for and shutting down virtual machines in off hours
- Running daily reports and producing a dashboard to report to senior management

As an example, let's suppose that you want to reduce your compute costs by configuring one of your development virtual machines to run only during your organization's business hours. You can write a script to start the VM in the morning and shut it down in the evening, and you can configure Azure Automation to run the script at set times. The following illustration shows the role of Azure Automation in this process.



Automating development environments

At the other end of your cloud infrastructure pipeline are a collection of development machines, which your developers use to write the applications and services that are the core of your business. You can use Azure DevTest Labs to deploy VMs with all of the correct tools and repositories that your developers need. Developers working on multiple services can switch between development environments without having to provision a new VM themselves. These development environments can be shut down when they are not in use, and then restarted when they are required again.

Check your knowledge

1. Which of the following cannot be used to automate infrastructure deployment?

- ○ Azure PowerShell
- ○ Azure CLI
- ◉ Azure Resource Manager templates

✗ **With Azure Resource Manager templates you can fully automate the deployment of infrastructure.**

○ Application Insights

✓ **Application Insights helps you monitor and analyze your application, it does not offer automation.**

2. Which of the following is not a common strategy for configuring virtual machines?

○ Custom images

◉ Auto-boot ISO images

✓ **Mounting ISO images with custom OS installations is not a common strategy for configuring VM.**

○ Post-deployment scripting

3. True or false: Azure Automation could be used to automate the start/stop of virtual machines during your off-hours.

○ True

✓ **Azure Automation is a great way to automate the start/stop of virtual machines in off-hours.**

◉ False

✗ **Azure Automation can automate the start/stop of virtual machines on a schedule.**

Next unit: Testing strategies for your application

Continue >

How are we doing? ☆ ☆ ☆ ☆ ☆

200 XP

Testing strategies for your application

12 minutes

Testing is one of the fundamental components of DevOps and agile development in general. If automation gives DevOps the required speed and agility to deploy software quickly, only through extensive testing will those deployments achieve the required reliability that customers demand.

A main tenet of a DevOps practice to achieve system reliability is the *shift left* principle. If your process for developing and deploying an application is depicted as a series of steps that are listed from left to right, your testing should be shifted as much as possible toward the beginning of your process (e.g. to the left), and not just at the very end of your process (e.g. to the right). Errors are far cheaper to repair when they're caught early, and issues can be expensive or impossible to fix later in your application's lifecycle.

Testing should occur on both application code and infrastructure code, and they should both be subject to the same quality controls. The environment where applications are running should be version-controlled, and deployed through the same mechanisms as application code. As a result, you can test and validate both application code and infrastructure code using DevOps testing paradigms.

You can use your favorite testing tool to run your tests, including Azure Pipelines for automated testing and Azure Testing Plans for manual testing.

There are multiple stages at which you can perform tests in the code's lifecycle, and each type of testing has several considerations that are important for you to understand. In this unit, you can find a summary from several of the different tests that you should consider when you're developing and deploying applications.

Automated Testing

Automating tests is the best way to make sure that they're executed. Depending on how frequently your automated tests are performed, they're typically limited in duration and scope, and the following descriptions will list some of the things you need to consider when creating your test strategy.

Unit Testing

Unit tests are tests typically run by each new version of code that's committed into your version-control system. Unit Tests should be extensive (they should ideally cover 100% of the code), and quick (typically under 30 seconds, although this number isn't a rule set in stone). Unit testing could verify things like the syntax correctness of application code, Resource Manager templates or Terraform configurations, that the code is following best practices, or that the code produces the expected results when provided certain inputs.

Unit tests should be applied both to application code and infrastructure code.

Smoke Testing

Smoke tests are more exhaustive than unit tests, but still not as much as integration tests. Smoke tests normally run in less than 15 minutes. While smoke tests don't verify the interoperability of your different components with each other, they verify that each component can be correctly built, and each component meets your criteria for expected functionality and performance.

Smoke tests usually involve building the application code, and if you're deploying infrastructure as part of your process, then possibly testing the deployment in a test environment.

Integration Testing

After making sure that your different application components operate correctly individually, integration testing determines whether your components can interact with each other as they should. Integration tests usually take longer than smoke testing, and consequently they're sometimes executed less frequently. For example, running integration tests every night still offers a good compromise between the different types of automated testing; your integration testing will detect interoperability issues between application components no later than one day after they were introduced.

Manual Testing

Manual testing is much more expensive than automated testing, and consequently it's used less frequently than automated testing. However, manual testing is fundamental for the correct functioning of the DevOps feedback loop; manual testing is used to correct errors before they become too expensive to repair, or before they cause customer dissatisfaction.

Acceptance Testing

There are many different ways of confirming that the application is doing what it should.

- **Blue/Green deployments**: when deploying a new application version, you can deploy it in parallel to the existing one. This way, you can start redirecting your clients to the new version. If everything goes well, you'll decommission the old version. If there's any problem with the new deployment, you can always redirect your clients back to the older deployment.
- **Canary releases**: you can expose new functionality of your application (ideally using feature flags) to a select group of users. If these users are satisfied with the new functionality, you can extend it to the rest of your user community. In this scenario, we're talking about releasing functionality, and not necessarily about deploying a new version of the application.
- **A/B testing**: A/B testing is similar to canary release testing, but while canary releases focus on mitigating risk, A/B testing focuses on evaluating the effectiveness of two similar ways of achieving the same goal. For example, if you have two versions of the layout of a certain area of your application, you could send half of your users to one version and the other half your users to the other, then you could use some metrics to see which layout works better for your application goals.

An important aspect to consider is how to measure the effectiveness of new features in an application. One way to measure that is through the Application Insights User Behavior Analytic, which you can use to determine how people are using your application. By analyzing the results, you can decide whether a new feature has increased or decreased your application's usability.

Certain services in Azure offer functionality that can help you implement these kind of tests. For example: the deployment slot functionality in the Azure App Service allows you to have two different versions of the same application running at the same time, and you can redirect your users to one version or the other.

Stress tests

As other sections of this framework have explained, designing your application code and infrastructure for scalability is of paramount importance. With this in mind, it's critical that you test whether your application and infrastructure code will both be able to adapt to changing load conditions. For example, if there's a spike in user activity, you need to be confident that your application and infrastructure will scale automatically to meet the increased demand.

During your stress tests, it's critical that you monitor all the components of the system in order to identify whether there are any scale limitations. Every component of the system that's not able to scale out can turn into a bottleneck (such as active/passive network components or databases). It's important for you to know the limits for each of your components so you can mitigate their impact into your application scale. As you learn more about the performance characteristics for each of your components, the discoveries that you make along the way might motivate you to replace some of your components with more scalable counterparts.

It's equally important to verify that after the stress test is concluded, your infrastructure scales back down to its normal condition in order to keep your costs under control.

Fault injection

Your application should be resilient to infrastructure failures, and introducing faults in the underlying infrastructure and observing how your application behaves is fundamental for increasing the trust in your redundancy mechanisms. For example: ungracefully shutting down infrastructure components, degrading the performance of certain elements such as network equipment, or purposely introducing faults in the environment are ways of verifying that your application is going to continue to behave or react as expected, should these situations ever occur in real life.

Most companies use a controlled way of injecting faults into the system, although if you're confident with your application resiliency, you could use automated frameworks. Chaos engineering is a practice adopted by some organizations to identify areas where faults may occur by purposefully making key pieces of infrastructure unavailable.

Security tests

Another critical component of your test strategy should be routine testing of your application for security vulnerabilities. You should regularly perform security tests against your application to identify any application vulnerabilities that are introduced through code defects or through software dependencies. These tests can include automated security scans to test against common vulnerabilities, such as cross-site scripting or SQL injection. Your security tests can also include *red team* exercises, where security teams attempt to compromise your application.

Use the results from these tests to provide feedback through your entire development process and resolve any security issues that you find in your code or software dependencies.

Check your knowledge

1. Which of the following would be an example of a stress test?

- ○ Scanning for SQL injection vulnerabilities.
- ○ Submitting an API call for each of the published APIs that your service exposes to validate an appropriate response.
- ◉ Testing every public method in your code for accuracy.

  ✗ **This would be an example of a unit test.**
- ○ Running an increasing number of API calls against your service.

  ✓ **This would test the load of the system and is an example of a stress test.**

2. If you had a new feature that you wanted to gain insight into whether or not it improves the user experience, which acceptance testing strategy would be the most appropriate?

- ○ Blue/Green deployment
- ◉ Canary release

  ✗ **Canary tests are intended to determine if a particular feature is rolled out to users, but is not the best strategy for this scenario.**
- ○ A/B test

  ✓ **This would allow you to test one experience with some users, and different experience with others. You could then analyze the results to determine the impact of the new user experience.**

Next unit: Summary

Continue >

How are we doing? ☆ ☆ ☆ ☆ ☆

100 XP

Introduction

2 minutes

Whether you're running a public-facing application that handles massive amounts of traffic or an internal business API that manages critical data for internal systems, your users expect a system that performs well. Scaling your system to handle load, identifying network bottlenecks, and optimizing your storage performance are important tasks to ensure your users have the best experience.

In this module, you'll learn about the *performance efficiency* pillar of the Azure Well-Architected Framework.

The concepts discussed in this module aren't all-inclusive. They represent some of the important aspects to consider when you build a solution on the cloud. For more details on the Azure Well-Architected Framework, go to the Azure Architecture Center as you start to plan and design your architecture.

Learning objectives

By the end of this module, you'll be able to:

- Scale your capacity based on workload
- Optimize network performance
- Optimize storage and database performance
- Improve application performance by identifying bottlenecks

Prerequisites

- Experience building or operating solutions using core infrastructure technology such as data storage, compute, and networking
- Experience building or operating technology systems to solve business problems

---

Next unit: Use scaling up and scaling out in your architecture

Continue >

200 XP

Use scaling up and scaling out in your architecture

15 minutes

It's rare that we can exactly predict the load on our systems. A public-facing application might grow rapidly in popularity and usage, or an internal application might need to support a larger user base as the business grows. Even when we can predict load, it's rarely flat. Retailers have more demand during the holidays, and sports websites peak during the playoff seasons.

In this unit, we will:

- Define scaling up or down and scaling out or in
- Discuss some ways Azure can improve your scaling capabilities
- Look at how serverless and container technologies can improve your architecture's ability to scale

What is scaling?

When we look at ways to increase or decrease the compute capacity and relative costs for your applications, it's important to define two key concepts: *scaling* and *resources*.

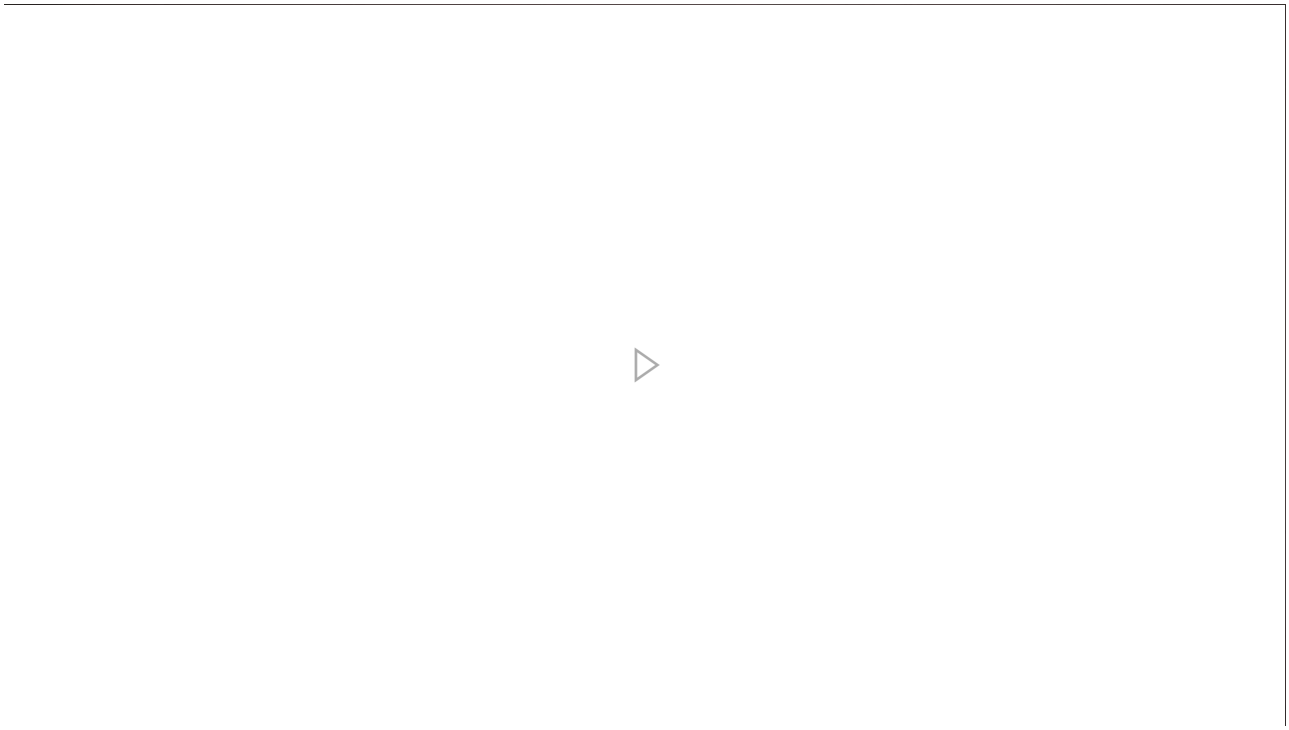
- **Scaling** is the process of managing your resources to help your application meet a set of performance requirements. When you have too many resources serving your users, you won't be using those resources efficiently, and you'll be wasting money. When you have too few resources available, the performance of your application can be adversely affected. Your goal is to meet your defined performance requirements while optimizing for cost.
- **Resources** can refer to anything you need to manage and run your applications. Memory and CPUs for virtual machines are the most obvious resources, but some Azure services might require you to consider bandwidth or abstractions, like Request Units (RUs) for Azure Cosmos DB.

In a hypothetical world where application demand was constant, it would be easy to predict the right amount of resources you'd need. But in the real world, the demands of applications change over time, so the right amount of resources you'll need can be harder to predict. If you're lucky, that change will be predictable or seasonal, but that's not typical of all scenarios.

Ideally, you want to provision the right amount of resources to meet demand, then adjust the amount as demand changes.

Scaling is difficult in an on-premises scenario, where you purchase and manage your own servers. Adding resources can be costly, and often it takes too much time to bring resources online. Sometimes it can take longer than your actual need for the increased capacity. It also can be difficult to reduce your resources during times of low demand on the system, so you might be stuck with the increased cost.

Easy scaling is a key benefit of Azure. Most Azure resources let you easily add or remove resources as demand changes. Many services have automated options that monitor demand and adjust for you. This automatic scaling capability is commonly known as *autoscaling*. Autoscaling lets you set thresholds for the minimum and maximum level of instances that should be available. It also adds or removes instances based on a performance metric. An example is CPU utilization.

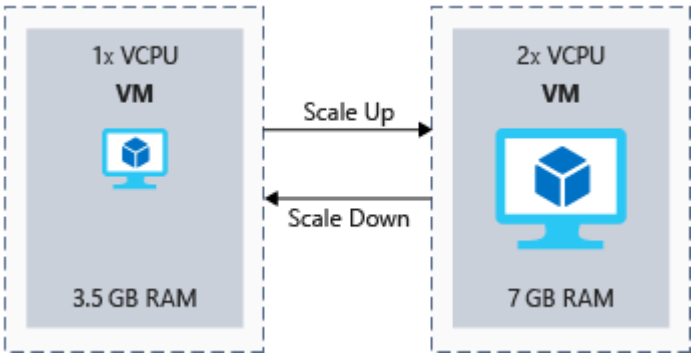


What is scaling up or down?

When you use a single instance of a service, such as a virtual machine, you might need to scale the number of resources that are available to your instance.

- **Scaling up** is the process where you increase the capacity of a given instance. For example, to increase your processing capacity, you might increase a virtual machine from 1 vCPU and 3.5 GB of RAM to 2 vCPUs and 7 GB of RAM.
- **Scaling down** is the process where you decrease the capacity of a given instance. For example, you might decrease a virtual machine's capacity from 2 vCPUs and 7 GB of RAM to 1 vCPU and 3.5 GB of RAM. In this way, you reduce capacity and cost.

The following illustration shows an example of changing the size of a virtual machine.



Let's look at what scaling up or down means in the context of Azure resources:

- In Azure virtual machines, you scale based on a virtual machine size. Each VM size has a certain amount of vCPUs, RAM, and local storage associated with it. For example, you could scale up from a Standard_DS1_v2 virtual machine (1 vCPU and 3.5 GB of RAM) to a Standard_DS2_v2 virtual machine (2 vCPUs and 7 GB of RAM).
- Azure SQL Database is a platform as a service (PaaS) implementation of Microsoft SQL Server. You can scale up a database based on the number of database transaction units (DTUs) or vCPUs. DTUs are an abstraction of underlying resources and are a blend of CPU, IO, and memory. For example, you could scale your database in Azure SQL Database from a size of P2 with 250 DTUs up to a P4 with 500 DTUs to give the database more throughput and capacity.
- Azure App Service is a PaaS website-hosting service on Azure. Websites run on a virtual server farm, which is also known as an App Service plan. You can scale the App Service plan up or down between tiers. You also have capacity options within those tiers. For example, an S1 App Service plan has 1 vCPU and 1.75 GB of RAM per instance. You could scale up to an S2 App Service plan, which has 2 vCPUs and 3 GB of RAM per instance.

To scale these types of capabilities in an on-premises environment, you typically have to wait for procurement and installation of the necessary hardware before you can start using the new level of scale. In Azure, the physical resources are already deployed and available for you. You simply need to select the alternate level of scale that you want to use.

You might need to consider the impact of scaling up in your solution. Your decision depends on the cloud services that you chose.

For example, if you choose to scale up in Azure SQL Database, the service deals with scaling up individual nodes and continues the operation of your service. Changing the service tier or performance level of a database creates a replica of the original database at the new performance level. Connections are then switched over to the replica. No data is lost during this process. There's only a brief interruption when the service switches over to the replica. The interruption is typically less than four seconds.

Alternatively, if you choose to scale a virtual machine up or down, you'll do so by selecting a different instance size. In most situations, if the VM is already running, you're required to restart the VM. With that in mind, expect that a reboot will be required when you scale your VMs. You'll need to account for that eventuality in your planning.

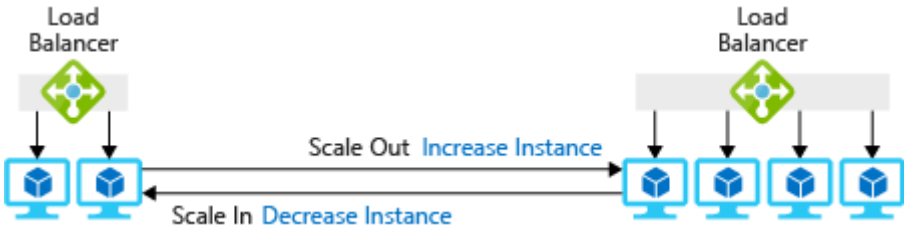
Finally, you should always look for places where scaling down is an option. If your application can provide adequate performance at a lower price tier, your Azure bill could be reduced.

What is scaling out or in?

You now know that scaling up and down adjusts the amount of resources a single instance has available. Scaling out and in adjusts the total number of instances.

- **Scaling out** is the process of adding more instances to support the load of your solution. For example, if your website front ends were hosted on virtual machines, you could increase the number of virtual machines if the level of load increased.
- **Scaling in** is the process of removing instances that are no longer needed to support the load of your solution. If your website front ends have low usage, you might want to lower the number of instances to save cost.

The following illustration shows an example of changing the number of virtual machine instances.



Here are some examples of what scaling out or in means in the context of Azure resources:

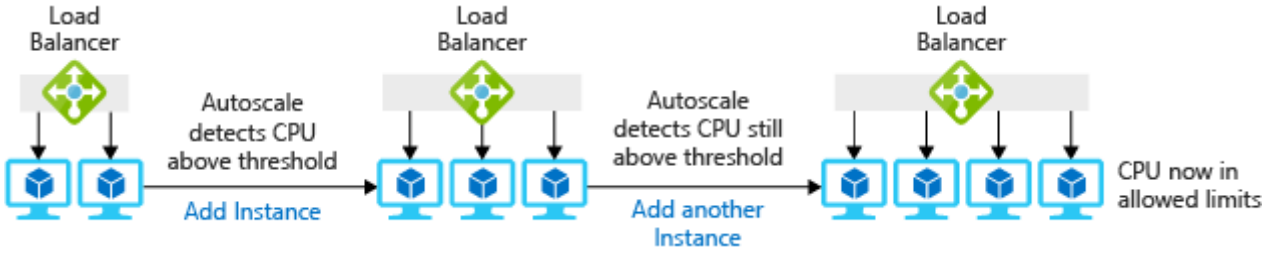
- For the infrastructure layer, you would likely use virtual machine scale sets to automate the addition and removal of extra instances.
  - Virtual machine scale sets let you create and manage a group of identical, load-balanced VMs.
  - The number of VM instances can automatically increase or decrease in response to demand or a defined schedule.
- In an Azure SQL Database implementation, you could share the load across database instances by *sharding*. Sharding is a technique to distribute large amounts of identically structured data across a number of independent databases.
- In Azure App Service, the App Service plan is the virtual web server farm that hosts your application. Scaling out in this way means that you're increasing the number of virtual

machines in the farm. As with virtual machine scale sets, the number of instances can be automatically raised or lowered in response to certain metrics or a schedule.

Scaling out is easily performed via the Azure portal, command-line tools, or Azure Resource Manager templates. In most cases, it's seamless to the user.

Autoscale

You can configure some of these services to use a feature called *autoscale*. With autoscale, you no longer have to worry about scaling services manually. Instead, you can set a minimum and maximum threshold of instances. You can scale based on specific metrics, like queue length or CPU utilization. You also can scale based on schedules. An example is weekdays between 5:00 PM and 7:00 PM. The following illustration shows how the autoscale feature manages instances to handle the load.



Considerations when you scale in and out

When you scale out, the startup time of your application can impact how quickly your application can scale. If your web app takes two minutes to start up and become available for users, that means each of your instances will take two minutes until they're available to your users. You need to consider this startup time when you determine how fast you want to scale.

You also need to think about how your application handles state. When the application scales in, any state that was stored on the instance that's removed from your environment will be lost. If a user connects to an instance that doesn't have its state, your application might force the user to sign in or reselect their data. The result is a poor user experience. A common pattern is to externalize your state to another service, like Azure Cache for Redis or SQL Database, which makes your web servers stateless. Now that your web front ends are stateless, you don't need to worry about which individual instances are available. They're all doing the same job and are deployed in the same way.

Throttling

We've established that the load on an application varies over time. This variation might be because of the number of active or concurrent users and the activities being performed. You

can use autoscaling to add capacity, but you can also use a throttling mechanism to limit the number of requests from a source. You can safeguard performance limits by putting known limits into place at the application level. In this way, you prevent the application from breaking. Throttling is most frequently used in applications that expose API endpoints.

After the application has identified that it would breach a limit, throttling could begin and ensure the overall system SLA isn't breached. For example, if you exposed an API for customers to get data, you could limit the number of requests to 100 per minute. If any single customer exceeded this limit, you could respond with an HTTP 429 status code and include the wait time before another request can successfully be submitted.

Serverless

Serverless computing provides a cloud-hosted execution environment that runs your apps, but completely abstracts the underlying environment. You create an instance of the service, and you add your code. No infrastructure management or maintenance is required or even allowed.

You configure your serverless apps to respond to events. An event can be a REST endpoint, a timer, or a message received from another Azure service. The serverless app runs only when it's triggered by an event.

When you work with serverless apps, infrastructure isn't your responsibility. Scaling and performance are handled automatically. You're only billed for the resources you use. There's no need to reserve capacity. Azure Functions, Azure Container Instances, and Azure Logic Apps are examples of serverless computing available on Azure.

Containers

A container is a method of running applications in a virtualized environment. A virtual machine is virtualized at the hardware level, where a hypervisor makes it possible to run multiple virtualized operating systems on a single physical server. Containers take the virtualization up a level. The virtualization is done at the OS level, which makes it possible to run multiple identical application instances within the same OS.

Containers are lightweight and well suited to scale-out scenarios. They're designed to be created, scaled out, and stopped dynamically as your environment and demands change. Another benefit of using containers is the ability to run multiple isolated applications on each virtual machine. Because containers are secured and isolated at a kernel level, you don't necessarily need to separate VMs for separate workloads.

Although you can run containers on virtual machines, there are a couple of Azure services that ease the management and scaling of containers:

- **Azure Kubernetes Service (AKS)**

  With Azure Kubernetes Service, you can set up virtual machines to act as your nodes. Azure hosts the Kubernetes management plane. You're charged only for the running worker nodes that host your containers.

  To increase the number of your worker nodes in Azure, you can use the Azure CLI to increase that number manually. At the time of this writing, you can use Cluster Autoscaler on AKS to autoscale your worker nodes. On your Kubernetes cluster, you can use the Horizontal Pod Autoscaler to scale out the number of instances of the container to be deployed.

  AKS can also scale with the Virtual Kubelet, which is described in the next section.

- **Azure Container Instances**

  Azure Container Instances is a serverless approach that lets you create and execute containers on demand. You're charged only for the execution time per second.

  You can use virtual nodes to connect Azure Container Instances into your Kubernetes environment, which includes AKS. The virtual nodes add-on for AKS is based on the open-source Virtual Kubelet. With virtual nodes, when your Kubernetes cluster demands additional container instances, those demands can be met from Container Instances. Because Container Instances is serverless, there's no need to have reserved capacity. You can take advantage of the control and flexibility of Kubernetes scaling with the per-second-billing of serverless.

Check your knowledge

1. Which is the most accurate description of scaling out?

- ○ Increasing the amount of resources allocated to an instance
- ○ Increasing the number of instances serving requests

  ✔ **Scaling out increases the number of instances service requests.**

- ◉ Adding additional storage to a virtual machine

  ✘ **Adding additional storage isn't an example of scaling out.**

- ○ Reaching the maximum level of scale for your application

2. Which is the most accurate description of scaling down?

- ○ Decreasing the number of instances serving requests
- ◉ Taking ownership of how your application scales

  ✗ **Taking ownership of how your application scales isn't scaling down.**
- ○ Decreasing the amount of resources allocated to an instance

  ✓ **Scaling down is the reduction of resources that a single instance has available.**
- ○ Remaining below the maximum level of scale for your application

3. Which of the following is *not* a consideration when you build a scaling strategy into your application?

- ○ The backup retention policies for your instances

  ✓ **While it's important to define the backup retention policies, it's unrelated to the ability of your application to scale.**
- ◉ State management of your application

  ✗ **State management is important to consider when you build a scaling strategy. Scaling requires you to store user state outside of the individual instances to ensure the user experience isn't broken as instances are added and removed.**
- ○ Startup time of your instances
- ○ Automating the scaling of your instances based on a metric or schedule

Next unit: Optimize network performance

Continue >

How are we doing? ☆ ☆ ☆ ☆ ☆

200 XP

Optimize network performance

10 minutes

Network performance can have a dramatic impact on a user's experience. In complex architectures with many different services, minimizing the latency at each network hop can affect the overall performance. In this unit, you'll learn about the importance of network latency and how to reduce it within your architecture. We'll also discuss strategies to minimize network latency between Azure resources and between users and Azure.

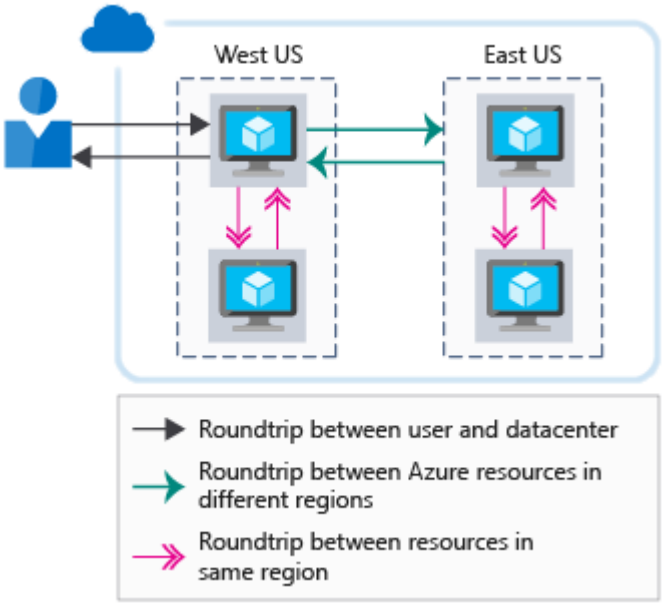
The importance of network latency

Latency is a measure of delay. Network latency is the time that it takes for data to travel between a source to a destination across a network. The time that it takes for data to travel from the source to a destination and for the destination to respond is commonly known as a round-trip delay.

In a traditional datacenter environment, latency might be minimal because resources often share the same location and a common set of infrastructure. The time taken to get from source to destination is typically lower when resources are physically close together.

In comparison, a cloud environment is built for scale. Cloud-hosted resources might not be in the same rack, datacenter, or region. This distributed approach can have an impact on the round-trip time of your network communications. While all Azure regions are interconnected by a high-speed fiber backbone, the speed of light is still a physical limitation. Calls between services in different physical locations will still have network latency directly correlated to the distance between them.

In addition, depending on the communication needs of an application, more round trips might be required. Each round trip comes with a latency tax, and each round trip adds to the overall latency. The following illustration shows how the latency perceived by the user is the combination of the round trips required to service the request.



Let's look at how to improve performance between Azure resources, and also from your users to your Azure resources.

Latency between Azure resources

Imagine that you work for a healthcare organization that's pilot testing a new patient booking system. This system runs on several web servers and a database. All of the resources are located in the West Europe Azure region. The scope of your pilot test is available only for users in Western Europe. This architecture minimizes your network latency, because all of your resources are colocated inside a single Azure region.

Suppose that your booking system's pilot testing was successful. As a result, the scope of your pilot test has expanded to include users in Australia. When the users in Australia view your website, they'll incur the additional round-trip time that's necessary to access all of the resources that are located in West Europe. Their experience will be diminished because of the additional network latency.

To address your network latency issues, your IT team decides to host another front-end instance in the Australia East region. This design helps reduce the time for your web servers to return content to users in Australia. But their experience is still diminished, because there's significant latency for data that's being transferred between the front-end web servers in Australia East and the database in West Europe.

There are a few ways you could reduce the remaining latency:

- Create a read-replica of the database in Australia East. A read replica allows reads to perform well, but writes still incur latency. Azure SQL Database geo-replication allows for read-replicas.

- Sync your data between regions with Azure SQL Data Sync.
- Use a globally distributed database such as Azure Cosmos DB. This database allows both reads and writes to occur regardless of location, but it might require changes to the way your application stores and references data.
- Use caching technology, such as Azure Cache for Redis, to minimize high-latency calls to remote databases for frequently accessed data.

The goal here is to minimize the network latency between each layer of the application. How you'll improve your network latency depends on your application and data architecture. Azure provides mechanisms to resolve latency issues through several services.

Latency between users and Azure resources

You've looked at the latency between your Azure resources, but you should also consider the latency between your users and your cloud application. You want to optimize delivery of the front-end user interface to your users. Let's look at some ways to improve the network performance between your users and your application.

Use a DNS load balancer for endpoint path optimization

In our example scenario, your IT team created an additional web front-end node in Australia East. But users have to explicitly specify which front-end endpoint they want to use. As the designer of a solution, you want to make the experience as smooth as possible for users.

Azure Traffic Manager could help. Traffic Manager is a DNS-based load balancer that you can use to distribute traffic within and across Azure regions. Rather than having the user browse to a specific instance of your web front end, Traffic Manager can route users based on a set of characteristics:

- **Priority**: You specify an ordered list of front-end instances. If the one with the highest priority is unavailable, Traffic Manager routes the user to the next available instance.
- **Weighted**: You set a weight against each front-end instance. Traffic Manager then distributes traffic according to those defined ratios.
- **Performance**: Traffic Manager routes users to the closest front-end instance based on network latency.
- **Geographic**: You set up geographical regions for front-end deployments and route your users based on data-sovereignty mandates or localization of content.

Traffic Manager profiles can also be nested. For example, you could initially route your users across different geographies (such as Europe and Australia) by using geographic routing. Then

you can route them to local front-end deployments by using the performance routing method.

Recall that the organization in our example scenario deployed a web front end in West Europe and another front end in Australia. Let's assume that they deployed Azure SQL Database with their primary deployment in West Europe and a read replica in Australia East. Let's also assume the application can connect to the local SQL instance for read queries.

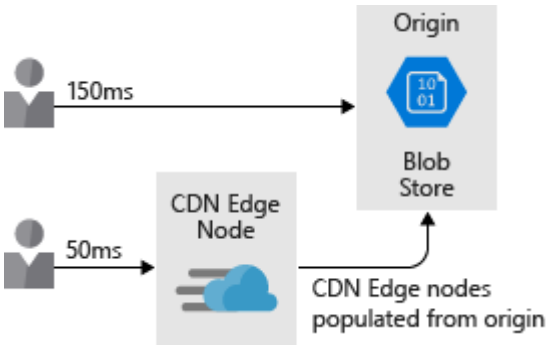
Your team deploys a Traffic Manager instance in performance mode and adds the two front-end instances as Traffic Manager profiles. As a user, you navigate to a custom domain name (for example, contoso.com) which routes to Traffic Manager. Traffic Manager then returns the DNS name of the West Europe or Australia East front end based on the best network latency performance.

It's important to note that this load balancing is only handled via DNS. No inline load balancing or caching is happening here. Traffic Manager simply returns the DNS name of the closest front end to the user.

Use a CDN to cache content close to users

Your website likely uses some form of static content, either whole pages or assets such as images and videos. This static content can be delivered to users faster by using a content delivery network (CDN) such as Azure Content Delivery Network.

With content deployed to Azure Content Delivery Network, those items are copied to multiple servers around the globe. Let's say one of those items is a video served from blob storage: `HowToCompleteYourBillingForms.MP4`. The team then configures the website so that each user's link to the video references the CDN edge server nearest them rather than referencing blob storage. This approach puts content closer to the destination, which reduces latency and improves the user experience. The following illustration shows how using Azure Content Delivery Network puts content closer to the destination, which reduces latency and improves the user experience.



You *can* also use content delivery networks to host cached dynamic content. Extra consideration is required, though, because cached content might be out of date compared

with the source. You can control context expiration by setting a time to live (TTL). If the TTL is too high, out-of-date content might be displayed and the cache would need to be purged.

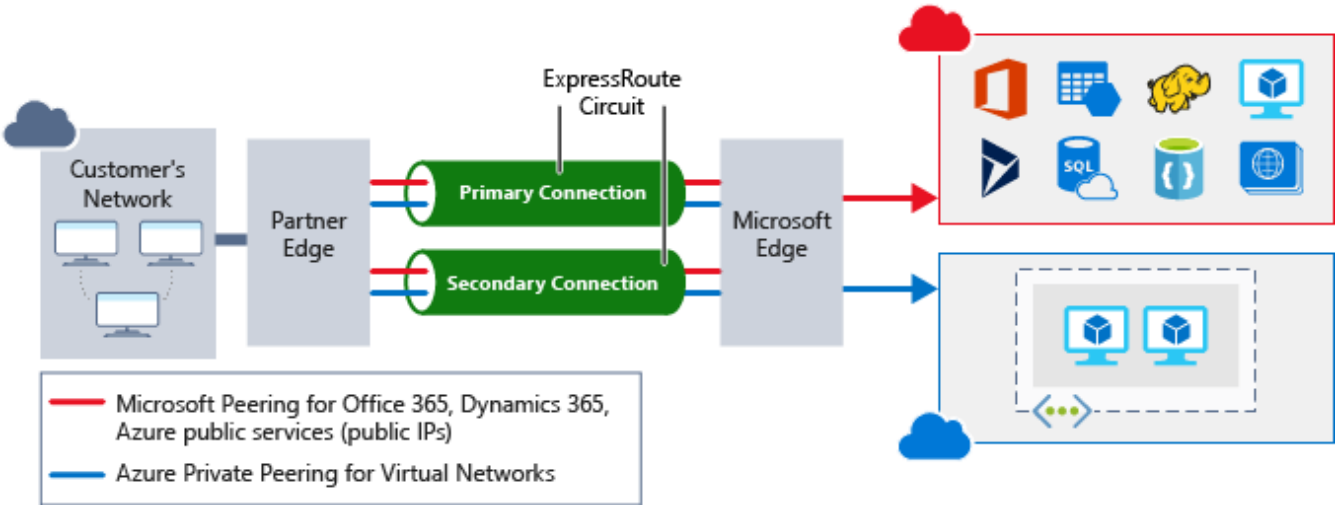
One way to handle cached content is with a feature called *dynamic site acceleration*, which can increase performance of webpages with dynamic content. Dynamic site acceleration can also provide a low-latency path to additional services in your solution. An example is an API endpoint.

Use ExpressRoute for connectivity from on-premises to Azure

Optimizing network connectivity from your on-premises environment to Azure is also important. For users who connect to applications, whether they're hosted on virtual machines or on platform as a service (PaaS) offerings like Azure App Service, you'll want to ensure they have the best connection to your applications.

You can always use the public internet to connect users to your services, but internet performance can vary and might be affected by outside issues. Also, you might not want to expose all of your services over the internet. You might want a private connection to your Azure resources. Site-to-site VPN over the internet is also an option. VPN overhead and internet variability can have a noticeable impact on network latency for high-throughput architectures.

Azure ExpressRoute can help. ExpressRoute is a private, dedicated connection between your network and Azure. It gives you guaranteed performance and ensures that your users have the best path to all of your Azure resources. The following illustration shows how an ExpressRoute circuit provides connectivity between on-premises applications and Azure resources.



If we consider our example scenario once again, your team decides to further improve user experience for users who are in their facilities by provisioning an ExpressRoute circuit in both Australia East and West Europe. This option gives users a direct connection to their booking system. It also ensures the lowest latency possible for their application.

Considering the impact of network latency on your architecture is important to ensure the best possible performance for your users. In this unit, we've looked at some options to lower network latency between users and Azure and between Azure resources.

Check your knowledge

1. Which of the following is the definition of the term "network latency"?

- ○ The amount of throughput available on the network
- ○ The amount of time it takes for information to be sent from a source to a destination

  ✔ **Latency is a measure of network transit time.**
- ◉ The TTL of an asset in a content delivery network

  ✘ **The TTL of an asset on a content delivery network relates to how often it's refreshed from the original source.**
- ○ A private circuit between an on-premises network and Azure

2. Suppose your web application is hosted in East US. Which of the following is the most cost-effective way to optimize network latency for users around the world?

- ○ Deploy Azure ExpressRoute to each of your users.
- ◉ Use a content delivery network to place assets and content closer to your users.

  ✔ **By using a content delivery network, assets are cached at locations physically closer to users, which reduces the amount of time to load them. A content delivery network cost-effectively improves network latency for users around the globe.**
- ○ Use Azure Traffic Manager in performance routing mode.
- ○ Deploy more instances in East US to serve the extra user load.

Next unit: Optimize storage performance

Continue >

200 XP

Optimize storage performance

9 minutes

It's important to include storage performance considerations in your architecture. Just like network latency, poor performance at the storage layer can affect your user experience. How would you optimize your data storage? What things do you need to consider to make sure that you're not introducing storage bottlenecks into your architecture?

In this unit, you'll learn about optimizing your storage performance in your architecture.

Optimize virtual machine storage performance

Let's first look at how to optimize storage for virtual machines (VMs). Disk storage plays a critical role in the performance of your VMs. Selecting the right disk type for your application is an important decision.

Different applications have different storage requirements. Your application might be sensitive to latency of disk reads and writes, or it might require the ability to handle a large number of input/output operations per second (IOPS) or greater overall disk throughput.

When you build an infrastructure as a service (IaaS) workload, which type of disk should you use? There are four options:

- **Local SSD storage**: Each virtual machine has a temporary disk that's backed by local SSD storage. The size of this disk varies depending on the size of the virtual machine. Because this SSD is local to the virtual machine, the performance is high. But data could be lost during a maintenance event or a redeployment of the VM. This disk is only suitable for temporary storage of data that you don't need permanently. For example, this disk is great for the VM's page or swap file, and for things like tempdb in Azure SQL Server. There's no charge for this storage. It's included in the cost of the VM.
- **Standard storage HDD**: This type of storage is spindle disk storage. It might fit well where your application isn't bound by inconsistent latency or lower levels of throughput. A dev/test workload where guaranteed performance isn't required is a great use case for this disk type.
- **Standard storage SSD**: This SSD-backed storage has the low latency of an SSD, but with lower levels of throughput. A non-production web server is a good use case for this disk type.

- **Premium storage SSD**: This SSD-backed storage is well suited for those workloads that are going into production and require the greatest reliability, demand consistent low latency, or need high levels of throughput and IOPS. Because these disks have greater performance and reliability capabilities, they're recommended for all production workloads.

Premium storage can attach only to specific VM sizes. Premium-storage-capable sizes are designated with an "s" in the name. Examples are D2*s*_v3 or Standard_F2*s*_v2. Any virtual machine type (with or without an "s" in the name) can attach standard storage HDD or SSD drives.

Disks can be striped by using a striping technology like Storage Spaces on Windows or mdadm on Linux. Striping increases the throughput and IOPS by spreading disk activity across multiple disks. You can use disk striping to push the limits of performance for disks. Striping is often seen in high-performance database systems and other systems with intensive storage requirements.

When you rely on virtual machine workloads, you need to evaluate the performance requirements of your application to determine the underlying storage that you'll provision for your virtual machines.

Optimize storage performance for your application

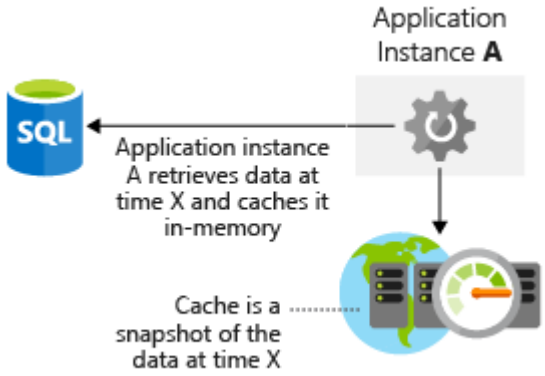
You've just learned how you can use different types of storage technologies to improve raw disk performance. You can also address the performance of access to data at the application layer. Let's look at a few ways you can do this.

Caching

A common approach to improve application performance is to integrate a caching layer between your application and your data store.

A cache typically stores data in memory and allows for fast retrieval. This data can be frequently accessed data, data that you specify from a database, or temporary data such as user state. You have control over the type of data stored, how often it refreshes, and when it expires. By colocating this cache in the same region as your application and database, you reduce the overall latency between the two. Pulling data out of the cache is almost always faster than retrieving the same data from a database. By using a caching layer, you can substantially improve the overall performance of your application.

The following illustration shows how an application retrieves data from a database, stores it in a cache, and uses the cached value as needed.

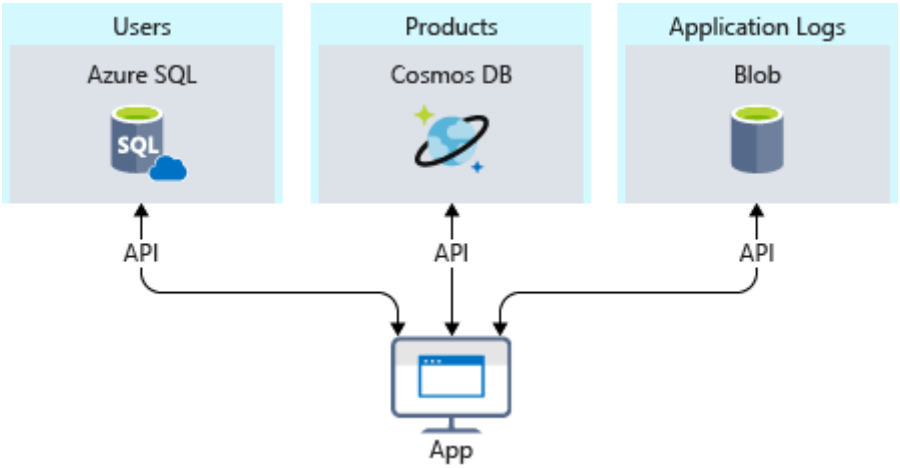


Azure Cache for Redis is a caching service on Azure that stores data in memory. It's based on the open-source Redis cache and is a fully managed service offering by Microsoft. You'll select the performance tier that you require, then you'll configure your application to use the service.

Polyglot persistence

Polyglot persistence is the use of different data storage technologies to handle your storage requirements.

Consider the following e-commerce example. Suppose that you store application assets in a blob store, product reviews and recommendations in a NoSQL store, and user profile or account data in a SQL database. The following illustration shows how an application might use multiple data storage techniques to store different types of data.



It's important to know that different data stores are designed for certain use cases or might be more accessible because of cost. As an example, storing blobs in a SQL database might be costly and slower to access than directly from a blob store.

Maintaining data consistency across distributed data stores can be a significant challenge. The issue is that strategies such as serialization and locking only work well if all application instances share the same data store and the application is designed to ensure that the locks are short-lived. But if data is partitioned or replicated across different data stores, locking and

serializing data access to maintain consistency can become an expensive overhead that affects the throughput, response time, and scalability of a system. As a result, most modern distributed applications don't lock the data that they modify. They take a more relaxed approach to consistency, which is known as *eventual consistency*.

Eventual consistency means that replica data stores eventually converge if there are no further writes. If a write is made to one of the data stores, reads from another data store might provide slightly out-of-date data. Eventual consistency enables higher scale because there's a low latency for reads and writes, instead of waiting to check if information is consistent across all stores.

Check your knowledge

1. What is polyglot persistence?

- ○ Using Azure SQL Database to store all of your data
- ◉ A way to keep connections alive for data transfers

  ✗ **Keepalives are methods of keeping connections alive for data transfers and aren't related to polyglot persistence.**
- ○ The use of different data storage technologies for different purposes in your application

  ✓ **Polyglot persistence is the use of different data storage technologies in the same application so that you can use the best approach for each type of data.**
- ○ An approach to deal with transient data faults

2. True or false: Azure Cache for Redis improves the performance of your application by storing a snapshot of data in memory.

- ○ True

  ✓ **Azure Cache for Redis stores application data in memory, which leads to fast retrieval times and an improvement in application performance related to frequently accessed data.**
- ◉ False

  ✗ **Azure Cache for Redis stores application data in memory, which leads to fast retrieval times and an improvement in application performance related to frequently accessed data.**

200 XP

Identify performance bottlenecks in your application

9 minutes

In today's fast-paced digital world, your users expect more from your applications. You want them to have a great experience and not be affected by performance issues. How do you identify potential performance bottlenecks in your architecture?

In this unit, you'll look at processes and tools that can help you make sure that your application performs well. They also can help you discover the cause if your application doesn't perform well.

Performance requirements

Before we talk about performance, it's important to talk about requirements. In theory, you could keep improving scalability and performance further and further without end. At some point, though, more improvement is prohibitively expensive, difficult, and doesn't have enough business impact to be worthwhile.

Nonfunctional requirements help you find that point. These particular requirements don't tell you what your app must *do*. Instead, they tell you what quality levels it must meet. For example, you can define nonfunctional requirements to discover:

- How fast a transaction must return under a given load
- How many simultaneous connections your application needs to support before it begins to return errors
- If there's server failure, the maximum amount of time your application is allowed to be down before a backup is online

Defining these requirements in advance of building your solution is critical. They ensure that your application meets expectations but doesn't require more effort or cost more money than necessary. You can also plan your monitoring and operations rules around these nonfunctional requirements.

You should discuss requirements with your stakeholders or customers, document them, and communicate them broadly to ensure that everyone agrees on what good performance

means.

In the following section, let's look at some ways to track performance in your applications.

Performance monitoring options in Azure

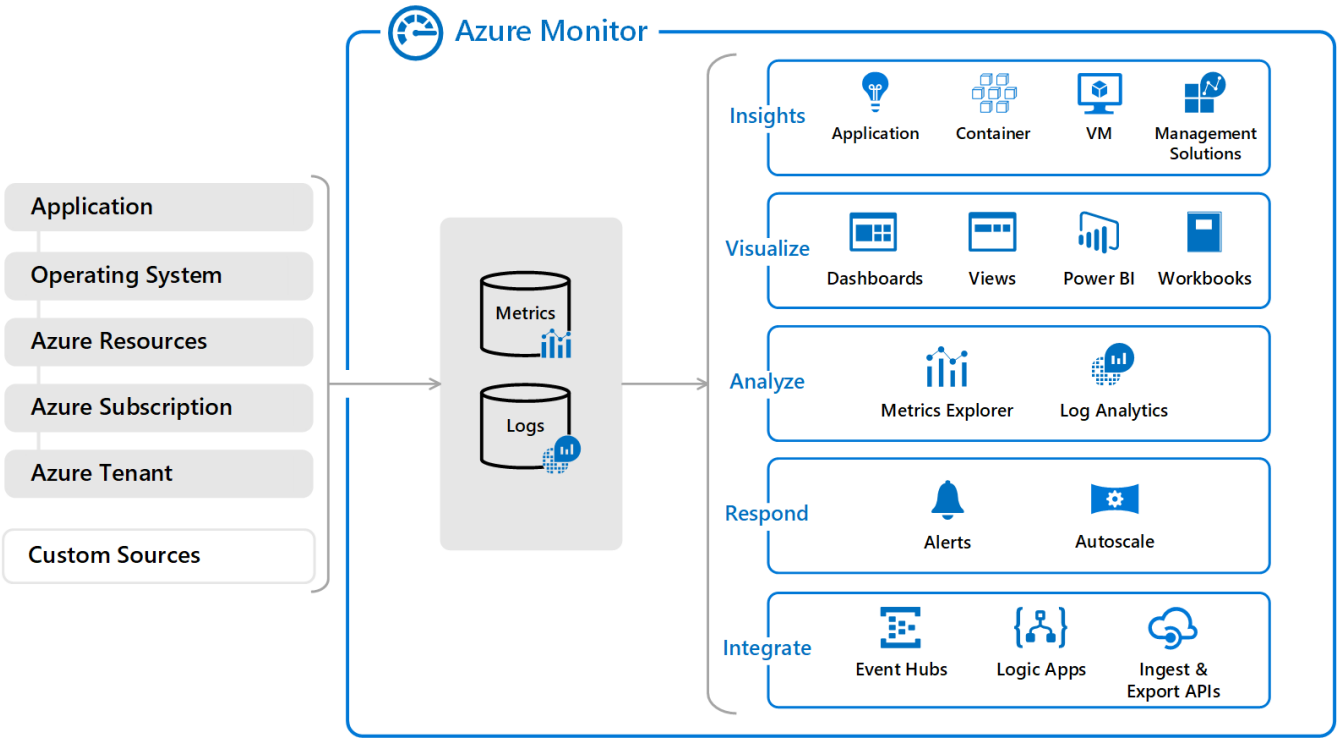
Monitoring is the act of collecting and analyzing data to determine the performance, health, and availability of your business application and associated resources.

You want to be kept informed that your applications are running smoothly. You can use proactive notifications to inform about critical issues that arise. There are many layers of monitoring to consider. Let's focus on the infrastructure layer and the application layer.

Azure Monitor

Azure Monitor provides a single management point for infrastructure-level logs and monitoring for most of your Azure services. Azure Monitor maximizes the availability and performance of your applications by delivering a comprehensive solution for collecting, analyzing, and acting on telemetry from your cloud and on-premises environments.

The following diagram depicts a high-level view of Azure Monitor. At the center of the diagram are the data stores for metrics and logs. These are the two fundamental types of data that Azure Monitor uses. On the left side are the sources of monitoring data that populate these data stores. On the right side are the different functions that Azure Monitor performs with this collected data, such as analysis, alerting, and streaming to external systems.



Azure Monitor can collect data from a variety of sources. You can think of monitoring data for your applications as occurring in tiers that range from your application to any OS and the services it relies on to the platform itself. Azure Monitor collects data from each of the following tiers:

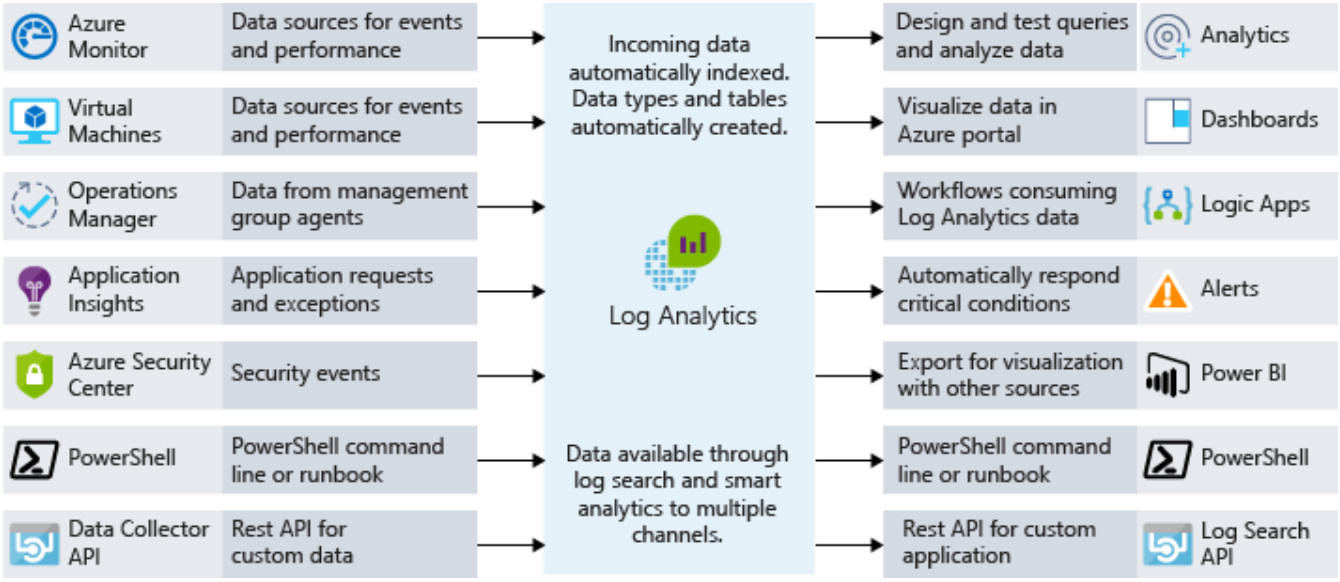
- **Application monitoring data**: Data about the performance and functionality of the code you've written, regardless of its platform.
- **Guest OS monitoring data**: Data about the OS on which your application is running. This OS might be in Azure, another cloud, or on-premises.
- **Azure resource monitoring data**: Data about the operation of an Azure resource.
- **Azure subscription monitoring data**: Data about the operation and management of an Azure subscription, and data about the health and operation of Azure itself.
- **Azure tenant monitoring data**: Data about the operation of tenant-level Azure services, such as Azure Active Directory (Azure AD).

As soon as you create an Azure subscription and start adding resources, such as VMs and web apps, Azure Monitor starts collecting data. Activity logs record when resources are created or modified. Metrics tell you how the resource is performing and the resources that it's consuming. You can also extend the data you collect. You can enable diagnostics in your apps and add agents to collect telemetry data from Linux and Windows or Application Insights.

Azure Monitor is the place to start for all your resource metric insights gathered in near-real time. Many Azure resources start outputting metrics automatically after they're deployed. For example, web apps built with the Web Apps feature of Azure App Service output compute and application-request metrics. Metrics from Application Insights are also collated here, in addition to VM host diagnostic metrics. VM guest diagnostic metrics also appear after you opt in.

Log Analytics

Centralized logging can help you uncover hidden issues that might be difficult to track down. With Log Analytics, you can query and aggregate data across logs. This cross-source correlation can help you identify issues or performance problems that might not be evident when you look at logs or metrics individually. The following illustration shows how Log Analytics acts as a central hub for monitoring data. Log Analytics receives monitoring data from your Azure resources and makes it available to consumers for analysis or visualization.



You can collate a wide range of data sources, security logs, Azure activity logs, and server, network, and application logs. You can also push on-premises System Center Operations Manager data to Log Analytics in hybrid deployment scenarios. Then Azure SQL Database can send diagnostic information directly into Log Analytics for detailed performance monitoring.

Centralized logging can be highly beneficial for troubleshooting all types of scenarios. You can use it to troubleshoot performance issues. Centralized logging is a key part of a good monitoring strategy for any architecture.

Application performance management

Deep application issues are often tricky to track down. This type of scenario is when it can be beneficial to integrate telemetry into your application by using an application performance management (APM) solution. An APM solution helps you to track down low-level application performance and behavior. Telemetry can include individual page request times, exceptions within your application, and even custom metrics to track business logic. This telemetry can provide a wealth of insight into what's going on within your application.

Application Insights is an Azure service that provides this deep application performance management. You install a small instrumentation package in your application and then set up an Application Insights resource in the Azure portal. The instrumentation monitors your app and sends telemetry data to the portal.

You can use Application Insights to consume telemetry from the host environments, such as performance counters, Azure diagnostics, and Docker logs. You can also set up web tests that periodically send synthetic requests to your web service. You could even configure your application to send custom events and metrics that you write yourself in the client or server code. For example, you can track application-specific events such as items sold or games won.

Application Insights stores its data in a common repository, and metrics are shared with Azure Monitor. Application Insights can also take advantage of shared functionality such as alerts, dashboards, and deep analysis with the Log Analytics query language.

A common pattern used to determine the availability of a web application is the health endpoint monitoring pattern. This pattern is used to monitor web applications and their associated back-end services to ensure that they're available and performing correctly. The pattern is implemented by querying a particular URI. The endpoint checks on the status of many components. Even the back-end services that the app depends on are checked, instead of only the availability of the front end itself. The health endpoint monitoring pattern acts as a service-level health check that returns an indication of the overall health of the service.

You should use an APM solution such as Application Insights to gain a deep understanding of your application and to correlate activity across your application. An APM solution can help you understand how a specific action works in the client browser, on the server, and through to downstream services. It also provides insight into trends. An APM solution provides notifications when there's a problem, helps to identify where the problem is, and informs you how to fix it before your users are aware of it.

Check your knowledge

1. Which of the following best describes Application Insights?

- ○ A single view for all your monitoring needs in Azure
- ◉ An implementation of the health endpoint monitoring pattern

  ✗ **The health endpoint monitoring pattern focuses on monitoring endpoints exposed by your application for overall health.**
- ○ A workspace to enable collection of logs across infrastructure and applications
- ○ A service to monitor availability, performance, and the use of web applications

  ✓ **Application Insights is a service to monitor availability, performance, and the use of web applications.**

2. True or false: Log Analytics tracks custom logs and metrics.

- ○ True

  ✓ **For custom logs and metrics, you can use the HTTP Data Collector API to write data to Log Analytics from any REST API client or an**

Azure Logic App to write data from a custom workflow.

◉ False

✗ For custom logs and metrics, you can use the HTTP Data Collector API to write data to Log Analytics from any REST API client or an Azure Logic App to write data from a custom workflow.

Next unit: Summary

Continue >

How are we doing? ☆ ☆ ☆ ☆ ☆

100 XP

Introduction

1 minute

Your business relies on access to the systems and data that make it run. Every minute that your customers or your internal teams don't have access to what they need can result in a loss of revenue. It is your job to make sure that doesn't happen. Here, you will learn about the *Reliability* pillar of the Azure Well-Architected Framework.

The concepts discussed in this module are not all-inclusive, but represent some of the important considerations when building a solution on the cloud. For more details on the Azure Well-Architected Framework, visit the Azure Architecture Center as you start planning and designing your architecture.

Learning objectives

By the end of this module, you'll be able to:

- Design a highly available application by leveraging Azure services
- Incorporate Azure disaster-recovery capabilities into your architecture
- Protect your application from data loss or corruption

Prerequisites

- Experience building or operating solutions using core infrastructure technology such as data storage, compute, and networking
- Experience building or operating technology systems to solve business problems

Next unit: Build a highly available architecture

Continue

How are we doing? ☆ ☆ ☆ ☆ ☆

200 XP

Build a highly available architecture

20 minutes

High availability (HA) ensures your architecture can handle failures. Imagine you're responsible for a system that must be always fully operational. Failures can and will happen, so how do you ensure that your system can remain online when something goes wrong? How do you perform maintenance without service interruption?

Here, you'll learn the need for high availability, evaluate application high-availability requirements, and see how the Azure platform helps you meet your availability goals.

What is high availability?

A highly available service is a service that absorbs fluctuations in availability, load, and temporary failures in dependent services and hardware. The application remains online and available (or maintains the appearance of it) while performing acceptably. This availability is often defined by business requirements, service-level objectives, or service-level agreements.

High availability is ultimately about the ability to handle the loss or severe degradation of a component of a system. This might be due to a virtual machine that's hosting an application going offline because the host failed. It could be due to planned maintenance for a system

upgrade. It could even be caused by the failure of a service in the cloud. Identifying the places where your system can fail, and building in the capabilities to handle those failures, will ensure that the services you offer to your customers can stay online.

High availability of a service typically requires high availability of the components that make up the service. Think of a website that offers an online marketplace to purchase items. The service that's offered to your customers is the ability to list, buy, and sell items online. To provide this service, you'll have multiple components: a database, web servers, application servers, and so on. Each of these components could fail, so you have to identify how they could fail and where your failure points are, then determine how to address these failure points in your architecture.

Evaluate high availability for your architecture

There are three steps to evaluate an application for high availability:

1. Determine the service-level agreement of your application
2. Evaluate the HA capabilities of the application
3. Evaluate the HA capabilities of dependent applications

Let's explore these steps in detail.

Determine the service-level agreement of your application

A service-level agreement (SLA) is an agreement between a service provider and a service consumer, in which the service provider commits to a standard of service based on measurable metrics and defined responsibilities. SLAs can be strict, legally bound, contractual agreements, or assumed expectations of availability by customers. Service metrics typically focus on service throughput, capacity, and availability, all of which can be measured in various ways. Regardless of the specific metrics that make up the SLA, failure to meet the SLA can have serious financial ramifications for the service provider. A common component of service agreements is guaranteed financial reimbursement for missed SLAs.

Service-level objectives (SLO) are the values of target metrics that are used to measure performance, reliability, or availability. These could be metrics defining the performance of request processing in milliseconds, the availability of services in minutes per month, or the number of requests processed per hour. By evaluating the metrics exposed by your application and understanding what customers use as a measure of quality, you can define the acceptable and unacceptable ranges for these SLOs. By defining these objectives, you clearly set goals and expectations with both the teams supporting the services and customers who

are consuming these services. These SLOs will be used to determine if your overall SLA is being met.

The following table shows the potential cumulative downtime for various SLA levels.

SLA	Downtime per week	Downtime per month	Downtime per year
99%	1.68 hours	7.2 hours	3.65 days
99.9%	10.1 minutes	43.2 minutes	8.76 hours
99.95%	5 minutes	21.6 minutes	4.38 hours
99.99%	1.01 minutes	4.32 minutes	52.56 minutes
99.999%	6 seconds	25.9 seconds	5.26 minutes

Higher availability is better, everything else being equal. But as you strive for more 9's, the cost and complexity to achieve that level of availability grows. An uptime of 99.99% translates to about 5 minutes of total downtime per month. Is it worth the additional complexity and cost to reach five 9s? The answer depends on the business requirements.

Here are some other considerations when defining an SLA:

- To achieve four 9's (99.99%), you probably can't rely on manual intervention to recover from failures. The application must be self-diagnosing and self-healing.
- Beyond four 9's, it's challenging to detect outages quickly enough to meet the SLA.
- Think about the time window against which your SLA is measured. The smaller the window, the tighter the tolerances. It probably doesn't make sense to define your SLA in terms of hourly or daily uptime.

Identifying SLAs is an important first step when determining the high availability capabilities that your architecture will require. These will help shape the methods you'll use to make your application highly available.

Evaluate the HA capabilities of the application

To evaluate the HA capabilities of your application, perform a failure analysis. Focus on single points of failure and critical components that would have a large impact on the application if they were unreachable, misconfigured, or started behaving unexpectedly. For areas that do

have redundancy, determine whether the application is capable of detecting error conditions and self-healing.

You'll need to carefully evaluate all components of your application, including the pieces designed to provide HA functionality, such as load balancers. Single points of failure will either need to be modified to have HA capabilities integrated, or will need to be replaced with services that can provide HA capabilities.

Evaluate the HA capabilities of dependent applications

You'll need to understand not only your application's SLA requirements to your consumer, but also the provided SLAs of any resource that your application may depend on. If you are committing an uptime to your customers of 99.9%, but a service your application depends on only has an uptime commitment of 99%, this could put you at risk of not meeting your SLA to your customers. If a dependent service is unable to provide a sufficient SLA, you may need to modify your own SLA, replace the dependency with an alternative, or find ways to meet your SLA while the dependency is unavailable. Depending on the scenario and the nature of the dependency, you can temporarily work around failing dependencies with solutions like caches and work queues.

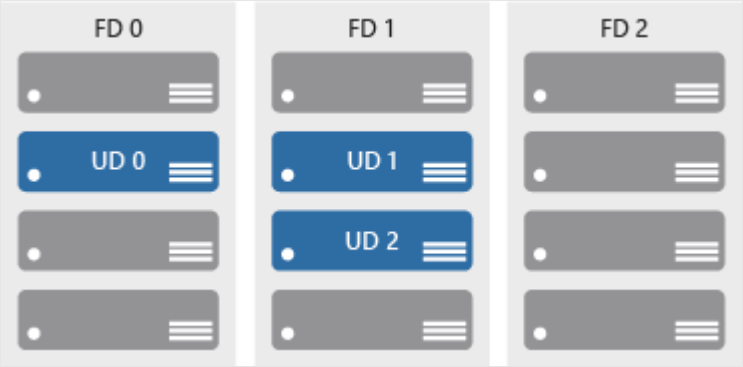
Azure's highly available platform

The Azure cloud platform has been designed to provide high availability throughout all its services. Like any system, applications may be affected by both hardware and software platform events. The need to design your application architecture to handle failures is critical, and the Azure cloud platform provides you with the tools and capabilities to make your application highly available. There are several core concepts when considering HA for your architecture on Azure:

- Availability sets
- Availability zones
- Load balancing
- Platform as a service (PaaS) HA capabilities

Availability sets

Availability sets are a way for you to inform Azure that VMs that belong to the same application workload should be distributed to prevent simultaneous impact from hardware failure and scheduled maintenance. Availability sets are made up of *update domains* and *fault domains*.



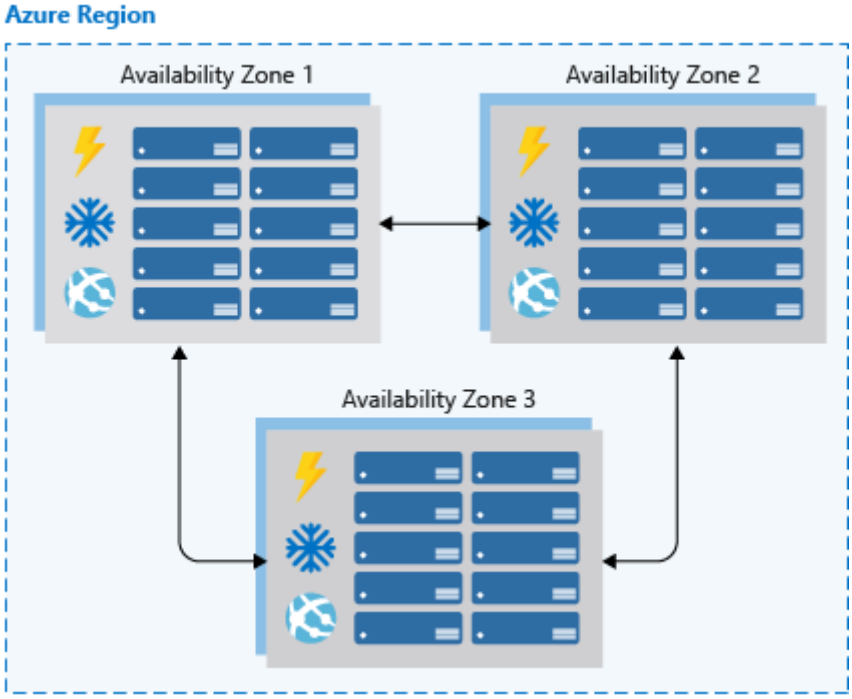
Update domains ensure that a subset of your application's servers always remain running when the virtual machine hosts in an Azure datacenter require downtime for maintenance. Most updates can be performed with no impact to the VMs running on them, but there are times when this isn't possible. To ensure that updates don't happen to a whole datacenter at once, the Azure datacenter is logically sectioned into update domains (UD). When a maintenance event occurs, such as a performance update and critical security patch that needs to be applied to the host, the update is sequenced through update domains. The use of sequencing updates using update domains ensures that the whole datacenter isn't unavailable during platform updates and patching.

While update domains represent a logical section of the datacenter, fault domains (FD) represent physical sections of the datacenter and ensure rack diversity of servers in an availability set. Fault domains align to the physical separation of shared hardware in the datacenter. This includes power, cooling, and network hardware that supports the physical servers located in server racks. In the event the hardware that supports a server rack has become unavailable, only that rack of servers would be affected by the outage. By placing your VMs in an availability set, your VMs will be automatically spread across multiple FDs so that in the event of a hardware failure, only part of your VMs will be impacted.

With availability sets, you can ensure your application remains online if a high-impact maintenance event is required or hardware failures occur.

Availability zones

Availability zones are independent physical datacenter locations within a region that include their own power, cooling, and networking. By taking availability zones into account when deploying resources, you can protect workloads from datacenter outages while retaining presence in a particular region. Services like virtual machines are *zonal services* and allow you to deploy them to specific zones within a region. Other services are *zone-redundant services* and will replicate across the availability zones in the specific Azure region. Both types ensure that there are no single points of failure within an Azure region.



Supported regions contain a minimum of three availability zones. When creating zonal service resources in those regions, you'll have the ability to select the zone in which the resource should be created. This will allow you to design your application to withstand a zonal outage and continue to operate in an Azure region before having to evacuate your application to another Azure region.

Availability zones are a newer high-availability configuration service for Azure regions, and are currently available for certain regions. It's important to check the availability of this service in the region in which you're planning to deploy your application if you want to consider this functionality. Availability zones are supported when using virtual machines, as well as several PaaS services. Availability zones are mutually exclusive with availability sets. When using availability zones, you no longer need to define an availability set for your systems. You'll have diversity at the data-center level, and updates will never be performed to multiple availability zones at the same time.

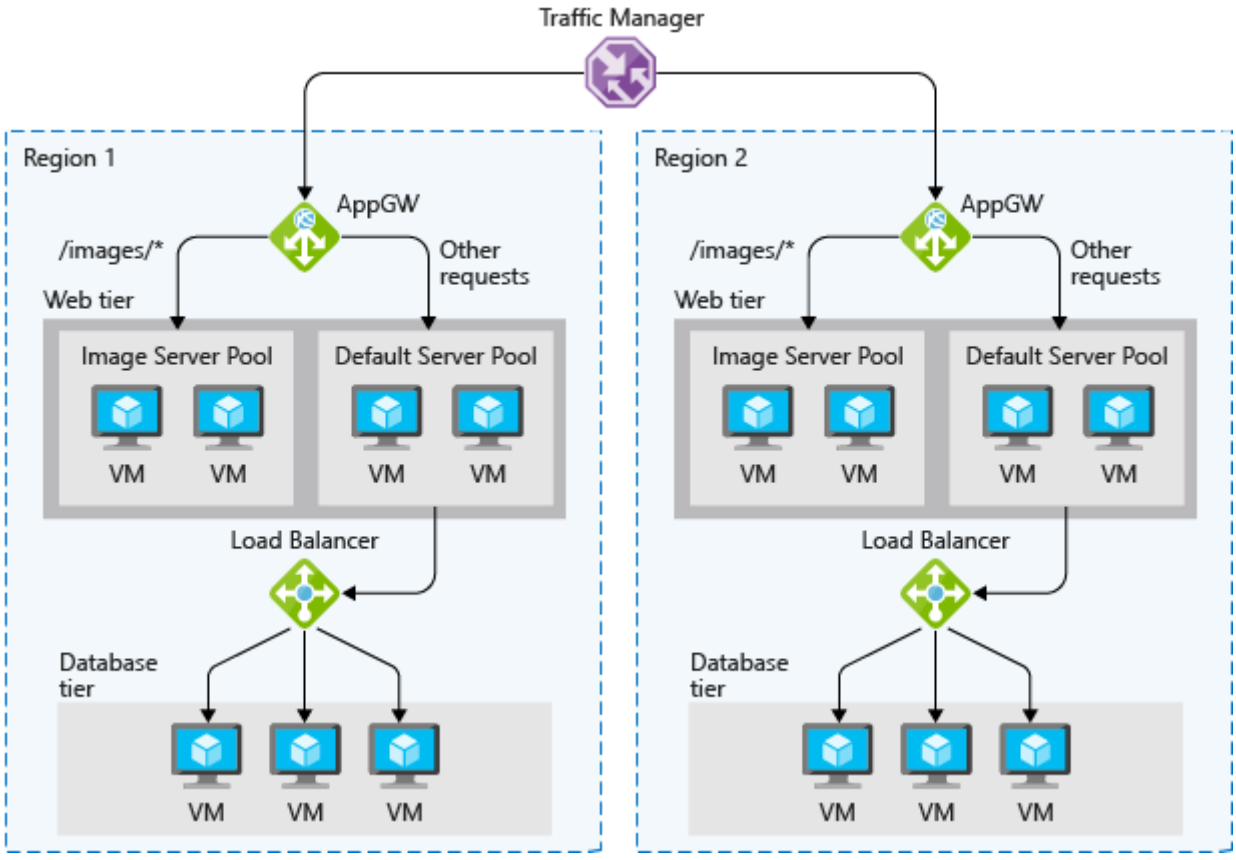
Load balancing

Load balancers manage how network traffic is distributed across an application. Load balancers are essential in keeping your application resilient to individual component failures and to ensure your application is available to process requests. For applications that don't have service discovery built in, load balancing is required for both availability sets and availability zones.

Azure possesses three load-balancing technology services that are distinct in their abilities to route network traffic:

- **Azure Traffic Manager** provides global DNS load balancing. You'd consider using Traffic Manager to provide load balancing of DNS endpoints within or across Azure regions. Traffic manager will distribute requests to available endpoints and use endpoint monitoring to detect and remove failed endpoints from load.
- **Azure Application Gateway** provides Layer 7 load-balancing capabilities, such as round-robin distribution of incoming traffic, cookie-based session affinity, URL path-based routing, and the ability to host multiple websites behind a single application gateway. Application Gateway monitors the health of all resources in its back-end pool by default, and automatically removes any resource considered unhealthy from the pool. Application Gateway continues to monitor the unhealthy instances and adds them back to the healthy back-end pool once they become available and respond to health probes.
- **Azure Load Balancer** is a Layer 4 load balancer. You can configure public and internal load-balanced endpoints and define rules to map inbound connections to back-end pool destinations by using TCP and HTTP health-probing options to manage service availability.

One or a combination of all three Azure load-balancing technologies can ensure you have the necessary options available to architect a highly available solution to route network traffic through your application.



PaaS HA capabilities

PaaS services come with high availability built in. Services such as Azure SQL Database, Azure App Service, and Azure Service Bus include high availability features and ensure that failures of an individual component of the service will be seamless to your application. Using PaaS services is one of the best ways to ensure that your architecture is highly available.

When architecting for high availability, you'll want to understand the SLA that you're committing to your customers. Then you'll evaluate both the HA capabilities that your application has and the HA capabilities and SLAs of dependent systems. After those have been established, use Azure features, such as availability sets, availability zones, and various load-balancing technologies to add HA capabilities to your application. Any PaaS services you should choose to use will have HA capabilities built in.

Check your knowledge

1. An SLA is important to define for your service because:

- ○ It helps you determine your backup retention requirements.
- ○ It helps you determine the availability of your service that you will commit to for your customers.

  ✔ **Service level agreement is a commitment you are making to your customers for the availability of your service.**
- ◉ You can use it to convince your manager to automate virtual-machine creation.

  ✘ **While this is a good discussion to have, your SLA will likely not have any impact on the outcome.**
- ○ You use it to define the access control settings for your resources.

2. Which of the following is not used in a highly available architecture?

- ○ Load balancing
- ◉ Availability zones

  ✘ **Availability zones are used in a highly available architecture to distribute systems across multiple datacenters within a region.**
- ○ Availability sets
- ○ A recovery service vault

  ✔ **Recovery service vaults provide a destination for backups, but they are not part of a highly available architecture.**

200 XP

Develop a disaster recovery strategy

20 minutes

Designing for high availability helps keep an application or process running despite unfavorable events and adverse conditions. But what do you do when something so significant happens that you've lost data, and it's impossible to keep your apps and processes from going down? When disaster strikes, you need to have a plan to get your services running again. You should know what your goals and expectations are for recovering, the costs and limitations of your plan, and how to execute on it.

What is disaster recovery?

Disaster recovery is about recovering from high-impact events that result in downtime and data loss. A disaster is a single, major event with an impact much larger and long-lasting than the application can mitigate through the high-availability portion of its design.

The word *disaster* often evokes thoughts of natural disasters and external events (earthquakes, floods, tropical storms, and so on) but many other kinds of disasters exist as well. A failed deployment or upgrade can leave an app in an unrecognizable state. Malicious hackers can encrypt or delete data and inflict other kinds of damage that take an app offline or eliminate some of its functionality.

Regardless of its cause, the best remedy for a disaster once it has occurred is a well-defined, tested disaster recovery plan and an application that actively supports disaster recovery efforts through its design.

How to create a disaster recovery plan

A disaster recovery plan is a single document that details the procedures that are required to recover from data loss and downtime caused by a disaster and identifies who's in charge of directing those procedures. Operators should be able to use the plan as a manual to restore application connectivity and recover data after a disaster occurs. A detailed, written plan that's dedicated to disaster recovery is critical to ensuring a favorable outcome. The process of creating the plan will help to assemble a complete picture of the application. The resulting written steps will promote good decision-making and follow-through in the panicked, chaotic aftermath of a disaster event.

Creating a disaster recovery plan requires expert knowledge of the application's workflows, data, infrastructure, and dependencies.

Risk assessment and process inventory

The first step in creating a disaster recovery plan is performing a risk analysis that examines the impact of different kinds of disasters on the application. The exact nature of a disaster isn't as important to the risk analysis as its potential impact through data loss and application downtime. Explore various kinds of hypothetical disasters and try to be specific when thinking about their effects. For example, a targeted malicious attack may modify code or data that results in a different kind of impact than an earthquake that disrupts network connectivity and datacenter availability.

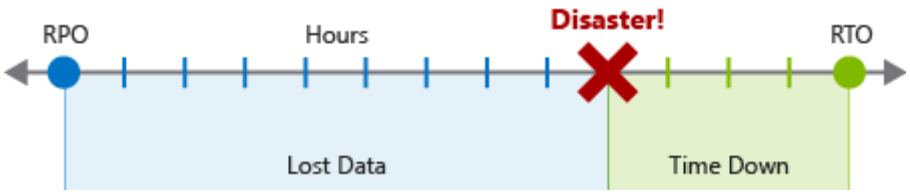
The risk assessment needs to consider *every* process that can't afford unlimited downtime, and every category of data that can't afford unlimited loss. When a disaster that affects multiple application components occurs, it's critical that the plan owners can use the plan to take a complete inventory of what needs attention and how to prioritize each item.

Some apps may only consist of a single process or classification of data. This is still important to note, as the application will likely be one component of a larger disaster recovery plan that includes multiple applications with the organization.

Recovery objectives

A complete plan needs to specify two critical business requirements for each process implemented by the application:

- **Recovery Point Objective (RPO)**: The maximum duration of acceptable data loss. RPO is measured in units of time, not volume: "30 minutes of data", "four hours of data", and so on. RPO is about limiting and recovering from data *loss*, not data *theft*.
- **Recovery Time Objective (RTO)**: The maximum duration of acceptable downtime, where "downtime" needs to be defined by your specification. For example, if the acceptable downtime duration is eight hours in the event of a disaster, then your RTO is eight hours.



Each major process or workload that's implemented by an app should have separate RPO and RTO values. Even if you arrive at the same values for different processes, each one should be

generated through a separate analysis that examines disaster-scenario risks and potential recovery strategies for each respective process.

The process of specifying an RPO and RTO is effectively the creation of disaster recovery requirements for your application. It requires establishing the priority of each workload and category of data and performing a cost-benefit analysis. The analysis includes concerns, such as implementation and maintenance cost, operational expense, process overhead, performance impact, and the impact of downtime and lost data. You'll need to define exactly what "downtime" means for your application, and in some cases, you may establish separate RPO and RTO values for different levels of functionality. Specifying RPO and RTO should be more than simply choosing arbitrary values. Much of the value of a disaster recovery plan comes from the research and analysis that goes into discovering the potential impact of a disaster and the cost of mitigating the risks.

Detailing recovery steps

The final plan should go into detail about exactly what steps should be taken to restore lost data and application connectivity. Steps often include information about:

- **Backups**: How often they're created, where they're located, and how to restore data from them
- **Data replicas**: The number and locations of replicas, the nature and consistency characteristics of the replicated data, and how to switch over to a different replica
- **Deployments**: How deployments are executed, how rollbacks occur, and failure scenarios for deployments
- **Infrastructure**: On-premises and cloud resources, network infrastructure, and hardware inventory
- **Dependencies**: External services that are used by the application, including SLAs and contact information
- **Configuration and notification**: Flags or options that can be set to gracefully degrade the application, and services that are used to notify users of application impact

The exact steps that are required will depend heavily on implementation details of the app, making it important to keep the plan updated. Routinely testing the plan will help identify gaps and outdated sections.

Designing for disaster recovery

Disaster recovery is not an automatic feature. It must be designed, built, and tested. An app that needs to support a solid disaster recovery strategy must be built from the ground up with

disaster recovery in mind. Azure offers services, features, and guidance to help you create apps that support disaster recovery, but it's up to you to include them in your design.

Designing for disaster recovery has two main concerns:

- **Data recovery**: Using backups and replication to restore lost data
- **Process recovery**: Recovering services and deploying code to recover from outages

Data recovery and replication

Replication duplicates stored data between multiple data store replicas. Unlike *backup*, which creates long-lived, read-only snapshots of data for use in recovery, replication creates real-time or near-real-time copies of live data. The goal of replication is to keep replicas synchronized with as little latency as possible while maintaining application responsiveness. Replication is a key component of designing for high availability and disaster recovery, and is a common feature of production-grade applications.

Replication is used to mitigate a failed or unreachable data store by executing a *failover*: changing application configuration to route data requests to a working replica. Failover is often automated, triggered by error detection built into a data-storage product, or detection that you implement through your monitoring solution. Depending on the implementation and the scenario, failover may need to be manually executed by system operators.

Replication is not something you implement from scratch. Most fully featured database systems and other data-storage products and services include some kind of replication as a tightly integrated feature due to its functional and performance requirements. However, it's up to you to include these features in your application design and make appropriate use of them.

Different Azure services support various levels and concepts of replication. For example:

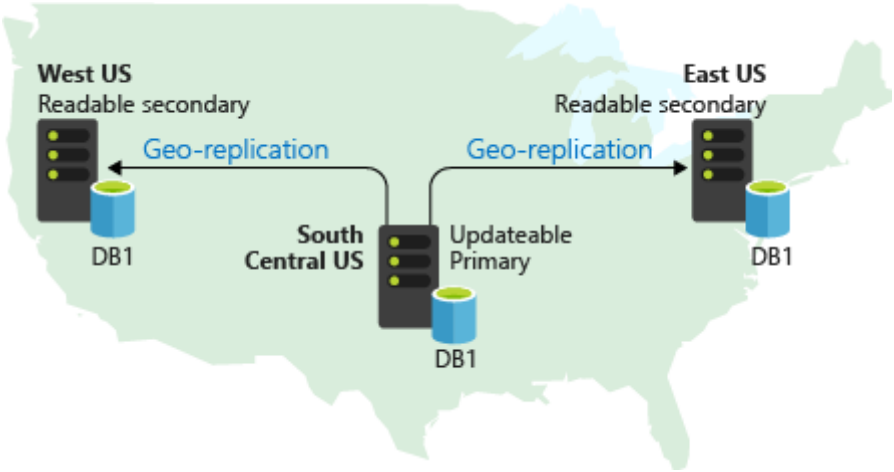
- **Azure Storage** replication capabilities depend on the type of replication that is selected for the storage account. This replication can be local (within a datacenter), zonal (between data centers within a region), or regional (between regions). Neither your application nor your operators interact with it directly. Failovers are automatic and transparent, and you simply need to select a replication level that balances cost and risk.

- **Azure SQL Database** replication is automatic at a small scale, but recovery from a full Azure datacenter or regional outage requires geo-replication. Setting up geo-replication is manual, but it's a first-class feature of the service, and well-supported by documentation.

- **Azure Cosmos DB** is a globally distributed database system, and replication is central to its implementation. With Azure Cosmos DB, you can configure options related to regions

associated with your database, data partitioning, and data consistency.

Many different replication designs exist that place different priorities on data consistency, performance, and cost. *Active* replication requires updates to take place on multiple replicas simultaneously, guaranteeing consistency at the cost of throughput. In contrast, *passive* replication performs synchronization in the background, removing replication as a constraint on application performance, but increasing RPO. *Active-active* or *multi-master* replication enables multiple replicas to be used simultaneously, enabling load balancing at the cost of complicating data consistency, while *active-passive* replication reserves replicas for live use only during failover.



Important

Neither replication nor backup are complete disaster recovery solutions on their own. Data recovery is only one component of disaster recovery, and replication will not fully satisfy many kinds of disaster recovery scenarios. For example, in a data-corruption scenario, the nature of the corruption may allow it to spread from the primary data store to the replicas, rendering all the replicas useless and requiring a backup for recovery.

Process recovery

After a disaster, business data isn't the only asset that needs recovering. Disaster scenarios will also commonly result in downtime, whether it's due to network connectivity problems, datacenter outages, or damaged VM instances or software deployments. The design of your application needs to allow you to restore it to a working state.

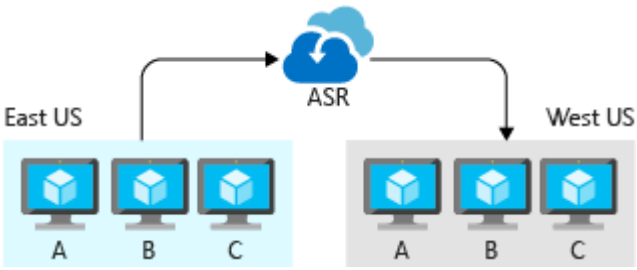
In most cases, process restoration involves failover to a separate, working deployment. Depending on the scenario, geographic location may be a critical aspect. For example, a large-scale natural disaster that brings an entire Azure region offline will necessitate restoring service in another region. Your application's disaster recovery requirements, especially RTO,

should drive your design and help you decide how many replicated environments you should have, where they should be located, and whether they should be maintained in a ready-to-run state or should be ready to accept a deployment in the event of disaster.

Depending on the design of your application, there are a few different strategies and Azure services and features that you can take advantage of to improve your app's support for process recovery after a disaster.

Azure Site Recovery

Azure Site Recovery is a service that's dedicated to managing process recovery for workloads running on VMs deployed to Azure, VMs running on physical servers, and workloads running directly on physical servers. Site Recovery replicates workloads to alternate locations and helps you to failover when an outage occurs and supports testing of a disaster recovery plan.

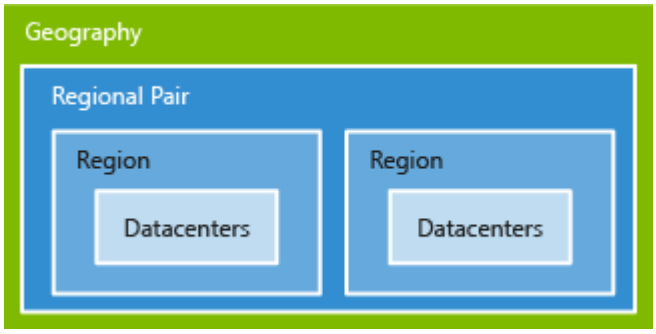


Site Recovery supports replicating whole VMs and physical server images as well as individual *workloads*, where a workload may be an individual application or an entire VM or operating system with its applications. Any application workload can be replicated, but Site Recovery has first-class integrated support for many Microsoft server applications, such as SQL Server and SharePoint, as well as a handful of third-party applications like SAP.

Any app that runs on VMs or physical servers should at least investigate the use of Azure Site Recovery. It's a great way to discover and explore scenarios and possibilities for process recovery.

Service-specific features

You might also have apps that run on Azure PaaS offerings like App Service. Most such services offer features and guidance for supporting disaster recovery. For certain scenarios, you can use service-specific features to support fast recovery. For example, Azure SQL Server supports geo-replication for quickly restoring service in another region. Azure App Service has a Backup and Restore feature, and the documentation includes guidance for using Azure Traffic Manager to support routing traffic to a secondary region.



Testing a disaster recovery plan

Disaster recovery planning doesn't end once you have a completed plan in hand. Testing the plan is a crucial aspect of disaster recovery to ensure that the directions and explanations are clear and up-to-date.

Choose intervals to perform different types and scopes of tests, such as testing backups and failover mechanisms every month, and performing a full-scale disaster recovery simulation every six months. Always follow the steps and details exactly as they're documented in the plan, and consider having someone unfamiliar with the plan give perspective on anything that could be made clearer. As you execute the test, identify gaps, areas of improvement, and places to automate and add these enhancements to your plan.

Make sure to include your monitoring system in your testing as well. For example, if your application supports automated failover, introduce failures in a dependency or other critical component to ensure that the application behaves correctly end-to-end, including detection of the failure and triggering of the automated failover.

By carefully identifying your requirements and laying out a plan, you'll be able to determine what types of services you'll need to use to meet your recovery objectives. Azure provides several services and features to help you meet these objectives.

Check your knowledge

1. Which of the following can be excluded from your recovery steps in your recovery plan?

- ○ How to restore backups
- ○ How to fail over to replicated systems
- ◉ How to reestablish network connectivity to on-premises or external networks

✗ Network connectivity to on-premises or external networks are often critical pieces of infrastructure. The steps to reestablish after failure

are critical pieces of a recovery plan.

- ○ None of the above should be excluded in a recovery plan.

✔ All these procedures are necessary in a recovery plan.

2. True or false: data replication and data backup are the same thing.

- ○ True
- ◉ False

✔ Replication creates real-time or near-real-time copies of live data for use in failover during execution. Backup creates long-lived, read-only snapshots of data for use in recovery,

Next unit: Protect your data with backup and restore

Continue >

How are we doing? ☆ ☆ ☆ ☆ ☆

200 XP

Protect your data with backup and restore

15 minutes

Backup is the final and most powerful line of defense against permanent data loss. An effective backup strategy requires more than simply making copies of data. It needs to take your application's data architecture and infrastructure into consideration. Your app may manage many kinds of data of varying importance, spread widely across filesystems, databases, and other storage services both in the cloud and on-premises. Using the right services and products for the job will simplify your backup process and decrease recovery time if a backup needs to be restored.

Establish backup and restoration requirements

As with a disaster recovery strategy, backup requirements are based on a cost-benefit analysis. Analysis of your app's data should be guided by the relative importance of the different categories of data the app manages, as well as external requirements, such as data retention laws.

To establish backup requirements for your app, group your application's data based on the following requirements:

- How much of this type of data can afford to be lost, measured in duration
- The maximum amount of time a restore of this type of data should require
- Backup retention requirements: how long and at what frequency do backups need to remain available

These concepts map neatly to the concepts of Recovery Point Objective and Recovery Time Objective (RPO and RTO). The duration of acceptable loss will generally translate directly to required backup intervals and RPO. The maximum amount of time a restore takes corresponds to the RTO for the data component of your application. Both requirements should be developed relative to the cost of achieving them. Every organization would like to say that they truly can't afford to lose *any* data, but often that's not the case when the cost of achieving that requirement is considered.

Backup absolutely plays a role in disaster recovery (DR), but backups, restores and their associated scenarios extend beyond the scope of DR. Backups may need to be restored in non-disaster situations, including those where RTO and RPO aren't of great concern. For

example, if a small amount of data older than your backup interval is corrupted or deleted, but the application doesn't experience downtime, your application may never be in danger of missing its SLA and a successful restore will result in no data lost. Your disaster recovery plan may or may not include guidance on performing restores in non-disaster situations.

> **Tip**
>
> Don't confuse *archival*, *replication*, and *backup*. Archival is the storage of data for long-term preservation and read access. Replication is the near-real-time copying of data between replicas to support high availability and certain disaster recovery scenarios. Some requirements, such as data retention laws, may influence your strategies for all three of these concerns. Archival, replication, and backup all require separate analysis and implementation.

Azure backup and restore capabilities

Azure offers several backup-related services and features for various scenarios, including data in Azure as well as on-premises data. Most Azure services offer some kind of backup functionality. Here, we'll look at a few of the most popular backup-related Azure offerings.

Azure Backup

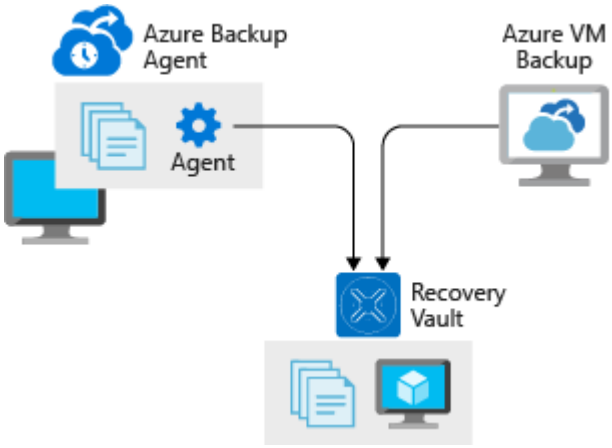
Azure Backup is a family of backup products that back up data to Azure Recovery Services vaults for storage and recovery. Recovery Service vaults are storage resources in Azure that are dedicated to holding data and configuration backups for virtual machines, servers, and individual workstations and workloads.

Azure Backup serves as a general-purpose backup solution for cloud and on-premises workflows that run on VMs or physical servers. It's designed to be a drop-in replacement for traditional backup solutions that stores data in Azure instead of archive tapes or other local physical media.

Four different products and services can use Azure Backup to create backups:

- **Azure Backup Agent** is a small Windows application that backs up files, folders, and system state from the Windows VM or server on which it's installed. It works in a way that's similar to many consumer cloud-based backup solutions, but requires configuration of an Azure Recovery vault. Once you download and install it onto a Windows server or VM, you can configure it to create backups up to three times a day.

- **System Center Data Protection Manager** is a robust, fully featured, enterprise-level backup and recovery system. Data Protection Manager is a Windows Server application that can back up file systems and virtual machines (Windows and Linux), create bare-metal backups of physical servers, and perform application-aware backup of many Microsoft server products, such as SQL Server and Exchange. Data Protection Manager is part of the System Center family of products and is licensed and sold with System Center, but it's considered part of the Azure Backup family because it can store backups in an Azure Recovery vault.
- **Azure Backup Server** is similar to Data Protection Manager, but it's licensed as part of an Azure subscription and doesn't require a System Center license. Azure Backup Server supports the same functionality as Data Protection Manager except for local tape backup and integration with the other System Center products.
- **Azure IaaS VM Backup** is a turnkey backup and restore feature of Azure Virtual Machines. VM backup supports once-per-day backups for Windows and Linux virtual machines. It supports recovery of individual files, full disks, and entire VMs, and can also perform application-consistent backups. Individual applications can be made aware of backup operations and get their filesystem resources into a consistent state before the snapshot is taken.



Azure Backup can add value and contribute to the backup and restore strategy for IaaS and on-premises applications of virtually any size and shape.

Azure Blob storage

Azure Storage doesn't include an automated backup feature, but blobs are commonly used to back up all kinds of data from various sources. Many services that provide backup capabilities use blobs to store their data, and blobs are a common target for scripts and tools in every kind of backup scenario.

General Purpose v2 storage accounts support three different blob-storage tiers of varying performance and cost. **Cool** storage offers the best cost-to-performance ratio for most backups, as opposed to **hot** storage, which offers lower access costs but higher storage costs.

Archive-tier storage may be appropriate for secondary backups or backups of data with low expectations for recovery time. It's low in cost, but requires up to 15 hours of lead time to access.

Immutable blob storage is configurable to be non-erasable and non-modifiable for a user-specified interval. Immutable blob storage was designed primarily to fulfill strict requirements for certain kinds of data, such as financial data. It's a great option for ensuring that backups are protected against accidental deletion or modification.

Azure SQL Database

Comprehensive, automatic backup functionality is included with Azure SQL Database at no extra charge. Full backups are created weekly, with differential backups performed every 12 hours, and log backups created every five minutes. Backups created by the service can be used to restore a database to a specific point in time, even if it's been deleted. Restores can be performed using the Azure portal, PowerShell, or the REST API. Backups for databases encrypted with Transparent Data Encryption, enabled by default, are also encrypted.

SQL Database backup is enterprise-grade, production ready, and enabled by default. If you're evaluating different database options for an app, it should be included as part of cost-benefit analysis, as it's a significant benefit of the service. Every app that uses Azure SQL Database should take advantage of it by including it in their disaster recovery plan and backup/restore procedures.

Azure App Service

Web applications hosted in the Azure App Service Standard and Premium tiers support turnkey-scheduled and manual backups. Backups include configuration and file contents, as well as contents of databases used by the app. They also support simple filters for excluding files. Restore operations can target different App Service instances, making App Service backup a simple way to move one app's contents to another.

App Service backups are limited to 10 GB total, including app and database content. They're a good solution for new apps under development and small-scale apps. More mature applications won't generally use App Service backup. They will instead rely on robust deployment and rollback procedures, storage strategies that don't use application disk storage, and dedicated backup strategies for databases and persistent storage.

Verify backups and test restore procedures

No backup system is complete without a strategy for verifying backups and testing restore procedures. Even if you use a dedicated backup service or product, you should still document and practice recovery procedures to ensure that they're well-understood and return the system to the expected state.

Strategies for verifying backups vary and will depend on the nature of your infrastructure. You may want to consider techniques like creating a new deployment of the application, restoring the backup to it, and comparing the state of the two instances. In many cases, this technique closely mimics actual disaster recovery procedures. Simply performing a comparison of a subset of the backup data with the live data immediately after creating a backup is enough. A common component of backup verification is attempting to restore old backups to ensure that they're still available and operational, and that the backup system hasn't changed in a way that renders them incompatible.

Any strategy is better than finding out that your backups are corrupted or incomplete while attempting to recover from a disaster.

A backup and restore strategy is an important part of ensuring your architecture can recover from the loss or corruption of data. Review your architecture to define your backup and restore requirements. Azure provides several services and features to provide backup and restore capabilities to any architecture.

Check your knowledge

1. True or false: all Azure data storage options include automatic backup of data?

- ○ True
- ◉ False

✔ **Most Azure services offer integrated backup functionality, but it is not always enabled by default. You can also add services like Azure Backup Agent as needed to make sure all your data is protected.**

2. Why is it important to test restores regularly?

- ○ To validate the integrity of your backups.
- ◉ To become familiar with the restoration process.

✘ **Testing restores regularly does allow you to become more familiar with the restoration process, but this is not the only correct choice.**

- ○ To identify gaps in your backup and restore processes.

○ All of the above.

✔ All are benefits of regularly testing your restore process.

Next unit: Summary

Continue >

How are we doing? ☆ ☆ ☆ ☆ ☆

100 XP

Introduction

2 minutes

Security is one of the most important aspects of any architecture. Ensuring that your business data and customer data are secure is critical. A public data breach can ruin a company's reputation, as well as cause significant personal and financial harm.

Gone are the days when security focused solely on a strong perimeter defense to keep out malicious hackers. Anything outside the perimeter was treated as hostile, whereas inside the wall, an organization's systems were trusted. Today's security posture is to assume breach and use the *Zero Trust model*.

Security professionals no longer focus on perimeter defense. Modern organizations have to support access to data and services evenly from both inside and outside the corporate firewall.

Here, you'll learn about the *Security* pillar of the Azure Well-Architected Framework.

The concepts discussed in this module are not all-inclusive, but represent some of the important considerations for building a solution on the cloud. For more details on the Azure Well-Architected Framework, visit the Azure Architecture Center as you start planning and designing your architecture.

Learning objectives

By the end of this module, you'll be able to:

- Develop a defense-in-depth approach to securing your architecture
- Choose the technologies to secure your Azure infrastructure
- Develop a strategy for secure identity management

Prerequisites

- Experience building or operating solutions by using core infrastructure technology such as data storage, compute, and networking
- Experience building or operating technology systems to solve business problems

200 XP

Defense in depth

10 minutes

There's no easy solution that solves all your problems from a security perspective. Let's imagine you work for an organization that has neglected security in its environment. The company has realized that it needs to put some major focus in this area. The company isn't sure where to start, or if it's possible to just buy a solution to make the environment secure. The company knows it needs a holistic approach, but is unsure what fits into that.

Here, we'll identify key concepts of defense in depth and identify key security technologies and approaches to support a defense-in-depth strategy. We'll also discuss how to apply these concepts when you're architecting your own Azure services.

Zero Trust model

The analyst firm Forrester Research introduced the *Zero Trust model*, which states that you should never assume trust, but instead continually validate trust. When users, devices, and data all resided inside the organization's firewall, they were assumed to be trusted. This assumed trust allowed for easy lateral movement after a malicious hacker compromised an endpoint device.

Most users now access applications and data from the internet, and many companies now allow users to use their own devices at work (*bring your own device*, or BYOD). Most components of the transactions — the users, network, and devices — are no longer completely under organizational control. The Zero Trust model relies on verifiable user and device trust claims to grant access to organizational resources. No longer is trust assumed based on the location inside an organization's perimeter.

This model has forced security researchers, engineers, and architects to rethink the approach applied to security and use a layered strategy to protect their resources.

Defense in depth: A layered approach to security

Defense in depth is a strategy that employs a series of mechanisms to slow the advance of an attack that's aimed at acquiring unauthorized access to information. Each layer provides

protection so that if one layer is breached, a subsequent layer is already in place to prevent further exposure.

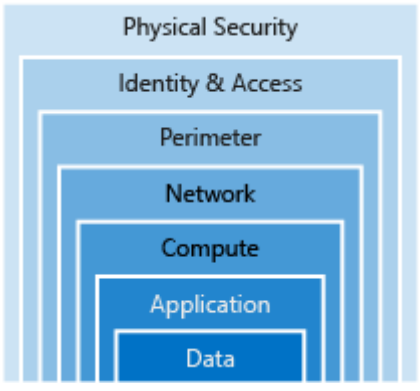
Microsoft applies a layered approach to security, both in its physical datacenters and across Azure services. The objective of defense in depth is to protect information and prevent it from being stolen by individuals who aren't authorized to access it. The common principles that help define a security posture are confidentiality, integrity, and availability, known collectively as CIA.

- **Confidentiality**: The principle of least privilege restricts access to information only to individuals explicitly granted access. This information includes protection of user passwords, remote access certificates, and email content.
- **Integrity**: The goal is to prevent unauthorized changes to information at rest or in transit. A common approach used in data transmission is for the sender to create a unique fingerprint of the data by using a one-way hashing algorithm. The hash is sent to the receiver along with the data. The receiver recalculates the data's hash and compares it to the original to ensure that the data wasn't lost or modified in transit.
- **Availability**: Ensure that services are available to authorized users. Denial-of-service attacks are a common cause of loss of availability to users. Natural disasters also drive system design to prevent single points of failure and deploy multiple instances of an application to geo-dispersed locations.

Security layers

You can visualize defense in depth as a set of concentric rings with the data to be secured at the center. Each ring adds a layer of security around the data. This approach removes reliance on any single layer of protection. It also acts to slow down an attack and provide alert telemetry that can be acted upon either automatically or manually.

Let's look at each of the layers.



Each layer can implement one or more of the CIA concerns:

#	Ring	Example	Principle
1	Data	Data encryption at rest in Azure Blob Storage	Integrity
2	Application	SSL/TLS encrypted sessions	Integrity
3	Compute	Regular application of OS and layered software patches	Availability
4	Network	Network security rules	Confidentiality
5	Perimeter	DDoS protection	Availability
6	Identity and access	Azure Active Directory user authentication	Integrity
7	Physical security	Azure datacenter biometric access controls	Confidentiality

Data

In almost all cases, attackers are after data:

- Stored in a database
- Stored on disk inside virtual machines
- Stored on a software as a service (SaaS) application such as Microsoft 365
- Stored in cloud storage

The people who store and control access to data are responsible for ensuring that it's properly secured. Often, regulatory requirements dictate the controls and processes that must be in place to ensure the confidentiality, integrity, and availability of the data.

Applications

- Ensure that applications are secure and free of vulnerabilities
- Store sensitive application secrets in a secure storage medium
- Make security a design requirement for all application development

Integrating security into the application development life cycle will help reduce the number of vulnerabilities introduced in code. Encourage all development teams to make their applications secure by default. Make security requirements non-negotiable.

Compute

- Secure access to virtual machines
- Implement endpoint protection and keep systems patched and current

Malware, unpatched systems, and improperly secured systems open your environment to attacks. The focus in this layer is on making sure that your compute resources are secure, and that you have the proper controls in place to minimize security issues.

Networking

- Limit communication between resources through segmentation and access controls
- Deny by default
- Restrict inbound internet access and limit outbound where appropriate
- Implement secure connectivity to on-premises networks

At this layer, the focus is on limiting network connectivity across all your resources. Segment your resources and use network-level controls to restrict communication to only what's needed. By limiting this communication, you reduce the risk of lateral movement throughout your network.

Perimeter

- Use distributed denial-of-service (DDoS) protection to filter large-scale attacks before they can cause a denial of service for users
- Use perimeter firewalls to identify and alert on malicious attacks against your network

At the network perimeter, it's about protecting from network-based attacks against your resources. Identifying these attacks, eliminating their impact, and alerting on them are important to keep your network secure.

Identity and access

- Control access to infrastructure (change control)
- Use single sign-on and multifactor authentication
- Audit events and changes

The identity and access layer is all about ensuring that identities are secure, that access granted is only what's needed, and that changes are logged.

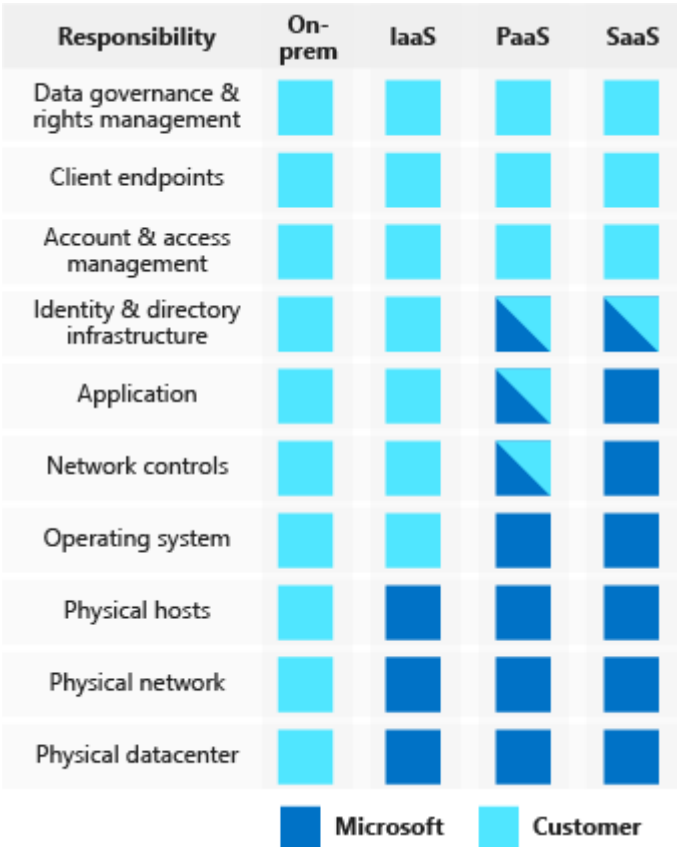
Physical security

Physical building security and controlling access to computing hardware within the datacenter are the first line of defense.

With physical security, the intent is to provide physical safeguards against access to assets. This ensures that other layers can't be bypassed, and that loss or theft is handled appropriately.

Shared responsibilities

As computing environments move from customer-controlled datacenters to cloud datacenters, the responsibility of security also shifts. Security is now a concern that both cloud providers and customers share.



Continuous improvement

The threat landscape is evolving in real time and at massive scale, so a security architecture is never complete. Microsoft and its customers need the ability to respond to these threats intelligently, quickly, and at scale.

Microsoft Defender for Cloud provides customers with unified security management and advanced threat protection to understand and respond to security events on-premises and in Azure. In turn, Azure customers have a responsibility to continually reevaluate and evolve their security architecture.

Check your knowledge

1. True or false: *defense in depth* is a strategy aimed to protect you against attacks that try to gain access to your information.

- ○ True
  - ✔ ***Defense in depth*** **is a layered system that provides protection from unauthorized access. If one layer is breached, another layer prevents further exposure.**
- ◉ False
  - ✘ ***Defense in depth*** **is a layered system that provides protection from unauthorized access. If one layer is breached, another layer prevents further exposure.**

2. True or false: by moving to the cloud, your architecture is fully secure and you can hand off all security responsibilities to your cloud provider.

- ◉ True
  - ✘ **Moving to the cloud does offload some responsibilities to the cloud provider, but a model of shared responsibility is still in effect.**
- ○ False
  - ✔ **Moving to the cloud does offload some responsibilities to the cloud provider, but a model of shared responsibility is still in effect.**

Next unit: Identity management

Continue >

How are we doing? ☆ ☆ ☆ ☆ ☆

200 XP

Identity management

10 minutes

Imagine you work for a healthcare organization that hosts an internal application and web portal for its clinicians to manage patient health data. The organization has received many requests for this application to be available to caregivers, who are often on-site with patients and therefore outside the network.

A recent data leak by malicious agents has forced the company to tighten its password policies. The company now requires users to change their passwords more frequently and use longer, more complex passwords. This has led to the unwanted side effect of users recording complex passwords insecurely as they struggle to remember multiple sets of credentials created for different administrative roles.

Here, we'll discuss identity as a security layer for internal and external applications. We'll also discuss the benefits of single sign-on (SSO) and multifactor authentication to provide identity security, and why you should consider replicating on-premises identities to Azure Active Directory (Azure AD).

Identity as a layer of security

Digital identities are an integral part of today's business and social interactions on-premises and online. In the past, identity and access services were restricted to operating within a company's internal network. Protocols such as Kerberos and LDAP were designed for this purpose.

More recently, mobile devices have become the primary way that people interact with digital services. Customers and employees alike expect to be able to access services from anywhere at any time. This expectation has driven the development of identity protocols that can work at internet scale across many disparate devices and operating systems.

As your organization evaluates the capabilities of its architecture around identity, it's looking at ways to bring the following capabilities into the application:

- Provide single sign-on to application users
- Enhance the application to use modern authentication with minimal effort
- Enforce multifactor authentication for all sign-ins outside the company's network

- Develop an application to allow patients to enroll and securely manage their account data

Single sign-on

The more identities a user has to manage, the greater the risk of a credential-related security incident. More identities mean more passwords to remember and change. Password policies can vary between applications. As complexity requirements increase, it's more difficult for users to remember them.

On the other side is the management required for all those identities. Additional strain is placed on help desks as they deal with account lockouts and password-reset requests. If a user leaves an organization, tracking down all those identities and ensuring that they're disabled can be challenging. An overlooked identity can allow access that should have been eliminated.

With single sign-on, users need to remember only one ID and one password. Access across applications is granted to a single identity tied to a user, simplifying the security model. As users change roles or leave an organization, access modifications are tied to the single identity, greatly reducing the effort needed to change or disable accounts.

Using single sign-on for accounts will make it easier for users to manage their identities. It will also increase the security capabilities in your environment.

SSO with Azure Active Directory

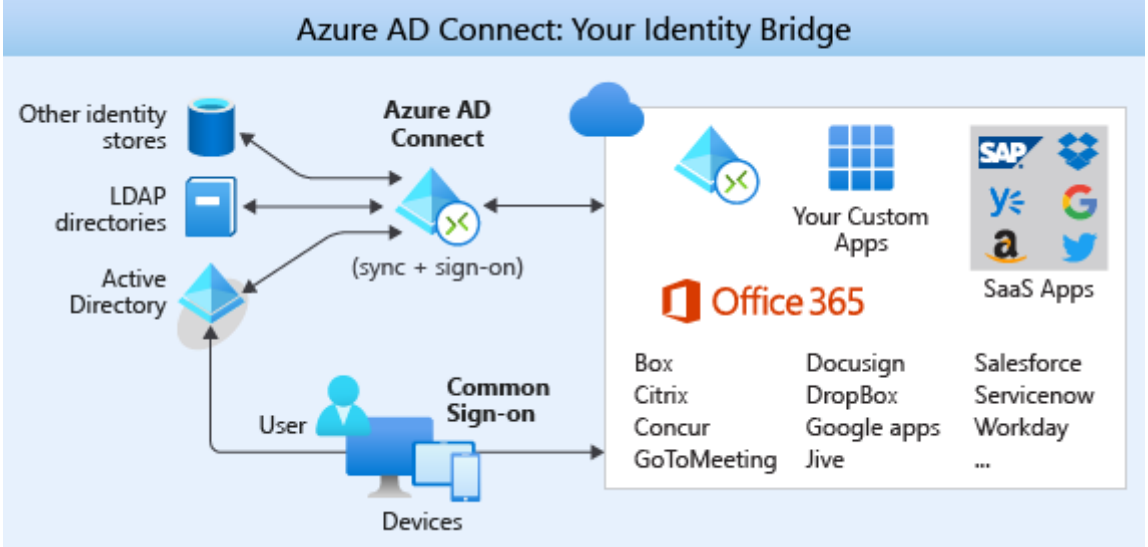
Azure AD is a cloud-based identity service. It has built-in support for synchronizing with your on-premises Active Directory instance, or it can be used on its own. This means that all your applications, whether on-premises, in the cloud (including Microsoft 365), or even mobile, can share the same credentials. Administrators and developers can control access to data and applications by using centralized rules and policies configured in Azure AD.

By using Azure AD for SSO, you'll also have the ability to combine multiple data sources into an intelligent security graph. This security graph can help you provide threat analysis and real-time identity protection to all accounts in Azure AD, including accounts that are synchronized from on-premises Active Directory. By using a centralized identity provider, you'll have centralized the security controls, reporting, alerting, and administration of your identity infrastructure.

Synchronize directories with Azure AD Connect

Azure AD Connect can integrate your on-premises directories with Azure Active Directory. Azure AD Connect provides the newest capabilities, and replaces older versions of identity-integration tools such as DirSync and Azure AD Sync.

It's a single tool to provide an easy deployment experience for synchronization and sign-in.



Your organization requires that authentication occurs primarily against on-premises domain controllers, but it also requires cloud authentication in a disaster recovery scenario. It doesn't have any requirements that Azure AD doesn't already support.

Your organization has made the decision to move forward with the following configuration:

- Use Azure AD Connect to synchronize groups, user accounts, and password hashes stored in on-premises Active Directory to Azure AD.

  This can be a backup if pass-through authentication is unavailable.

- Configure pass-through authentication by using an on-premises authentication agent installed on Windows Server.

- Use the seamless SSO feature of Azure AD to automatically sign in users from on-premises domain-joined computers.

  SSO reduces user friction by suppressing multiple authentication requests.

Authentication and access

Your organization's security policy requires that all sign-ins that occur outside the company's perimeter network are authenticated with an additional factor of authentication. This requirement combines two aspects of the Azure AD service: multifactor authentication and Conditional Access policies.

Multifactor authentication

Multifactor authentication provides additional security for your identities by requiring two or more elements for full authentication. These elements fall into three categories:

- *Something you know*: A password or the answer to a security question
- *Something you have*: A mobile app that receives a notification, or a token-generating device
- *Something you are*: Some sort of biometric property such as a fingerprint or face scan used on many mobile devices

Using multifactor authentication increases the security of your identity by limiting the impact of credential exposure. An attacker who has a user's password would also need to have possession of their phone or their face in order to fully authenticate. Authentication with only a single factor verified is insufficient, and the attacker would be unable to use those credentials to authenticate. The benefits that this approach brings to security are huge, so organizations should enable multifactor authentication wherever possible.

Azure AD has multifactor authentication capabilities built in and will integrate with other multifactor authentication providers. Basic multifactor authentication features are available to Microsoft 365 and Azure AD administrators for no extra cost. If you want to upgrade the features for your admins or extend multifactor authentication to the rest of your users, you can purchase more capabilities.

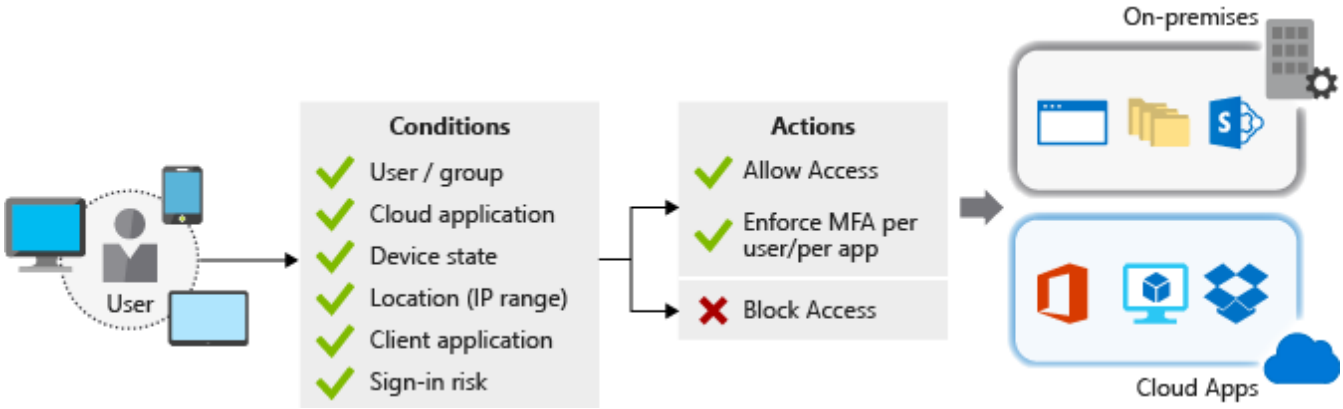
Conditional Access policies

Along with multifactor authentication, ensuring that additional requirements are met before granting access can add another layer of protection. Blocking logins from a suspicious IP address, or denying access from devices without malware protection, can limit access from risky sign-ins.

Azure Active Directory provides Conditional Access policies based on group, location, or device state. The location feature allows your organization to differentiate IP addresses that don't belong to the network, and it satisfies the security policy to require multifactor authentication from all such locations.

Your organization has created a Conditional Access policy that requires users who access the application from an IP address outside the company network to be challenged with multifactor authentication.

In the following illustration, user requests to access the on-premises and cloud applications are first checked against a list of conditions. The requests are either allowed access, forced to go through multifactor authentication, or blocked based on the conditions that they satisfy.



Securing applications

Your employees require secure remote access to their administrative application hosted on-premises. Users currently authenticate to the application by using Windows Integrated Authentication from their domain-joined machines behind the corporate firewall.

Although a project to incorporate modern authentication mechanisms into the application has been planned, there's considerable business pressure to enable remote-access capabilities as soon as possible. Azure AD Application Proxy can allow users to access the application remotely without any code changes.

Azure AD Application Proxy is:

- Simple
  - You don't need to change or update your applications to work with Application Proxy.
  - Your users get a consistent authentication experience. They can use the MyApps portal to get single sign-on to both SaaS apps in the cloud and your apps on-premises.
- Secure
  - When you publish your apps by using Azure AD Application Proxy, you can take advantage of the authorization controls and security analytics in Azure. You get cloud-scale security and Azure security features like Conditional Access and two-step verification.
  - You don't have to open any inbound connections through your firewall to give your users remote access.
- Cost-effective
  - Application Proxy works in the cloud, so you can save time and money. On-premises solutions typically require you to set up and maintain perimeter networks, edge servers, or other complex infrastructures.

Azure AD Application Proxy has two components. The first is a connector agent that sits on a server running Windows within your corporate network. The second is an external endpoint,

either the MyApps portal or an external URL. When a user goes to the endpoint, they authenticate with Azure AD and are routed to the on-premises application via the connector agent.

Working with consumer identities

Since your organization integrated modern authentication with its existing application, it has quickly acknowledged the benefits of a managed-identity system such as Azure AD. The leadership team is now interested in exploring other ways that Microsoft identity services can add business value. The team is focusing its attention on external customers, and how modernization of existing customer interactions might provide tight integration with identity providers like Google, Facebook, and LinkedIn.

Azure AD B2C is an identity management service that's built on the foundation of Azure Active Directory. It lets you customize and control how customers sign up, sign in, and manage their profiles when using your applications. This includes applications developed for iOS, Android, and .NET, among others.

Azure AD B2C provides a social identity login experience, while at the same time protecting your customer identity profile information. Azure AD B2C directories are distinct from standard Azure AD directories and can be created in the Azure portal.

Check your knowledge

1. Which of the following is *not* a benefit of single sign-on?

- ○ Increased complexity in assigning permissions to users

  ✔ **Single sign-on can reduce the complexity needed to assign permissions to users because there are fewer identities to manage.**

- ○ Fewer IDs and passwords for users to remember
- ◉ Lower administration effort when users change roles or leave an organization

  ✘ **Single sign-on can reduce the complexity needed to assign permissions to users because there are fewer identities to manage.**

- ○ Ensuring a consistent password policy across applications

2. Which of the following would be a valid second element for multifactor authentication, when combined with a password?

- ○ A driver's license
- ○ A token-generating device

  ✔ A token-generating device or mobile app that receives a notification are great components to use for multifactor authentication.
- ◉ Your account number

  ✘ Your account number can't be used for multifactor authentication.
- ○ Your car keys

Next unit: Infrastructure protection

Continue >

How are we doing? ☆ ☆ ☆ ☆ ☆

200 XP

Infrastructure protection

10 minutes

Imagine your organization recently experienced a significant outage to a customer-facing web application. An engineer was granted full access to a resource group that contains the production web application. This engineer accidentally deleted the resource group and all child resources, including the database that hosts live customer data.

Fortunately, the application's source code and resources were available in source control, and regular database backups were running automatically on a schedule. The service was reinstated relatively easily. Here, we'll explore how the organization could have avoided this outage by using capabilities in Azure to protect access to the infrastructure.

Criticality of infrastructure

Cloud infrastructure is becoming an essential piece of many businesses. It's critical to ensure that people and processes have only the rights they need to get their job done. Assigning incorrect access can result in data loss, data leakage, or unavailability of services.

System administrators can be responsible for a large number of users, systems, and permission sets, so correctly granting access can quickly become unmanageable and can lead to a "one size fits all" approach. This approach can reduce the complexity of administration, but makes it far easier to inadvertently grant more permissive access than required.

Role-based access control

Role-based access control (RBAC) offers a slightly different approach. Roles are defined as collections of access permissions. Security principals are mapped to roles directly or through group membership. Separating security principals, access permissions, and resources provides simplified access management and more detailed control.

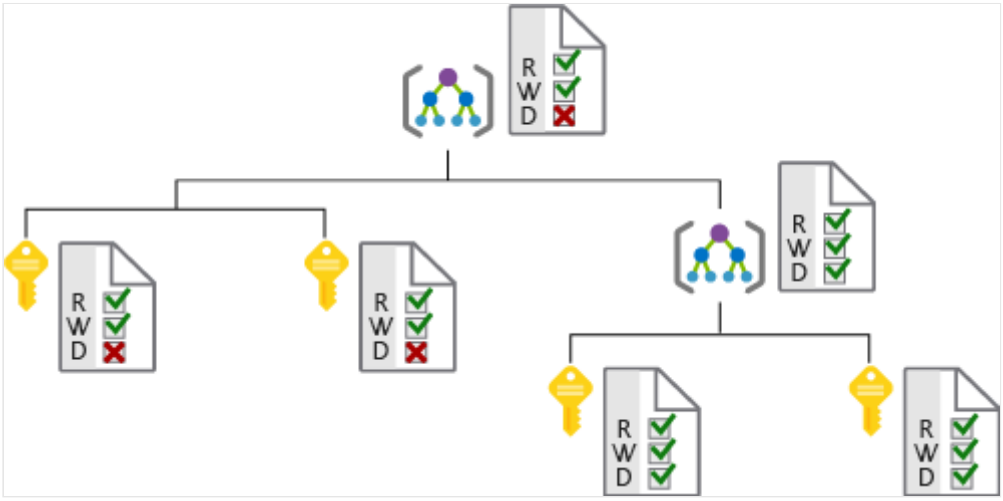
On Azure, users, groups, and roles are all stored in Azure Active Directory (Azure AD). The Azure Resource Manager API uses role-based access control to secure all resource access management within Azure.

Roles and management groups

Roles are sets of permissions, like *read-only* or *contributor*, that users can be granted to access an Azure service instance. Roles can be granted at the level of an individual service instance, but they also flow down the Azure Resource Manager hierarchy. Roles assigned at a higher scope, like an entire subscription, are inherited by child scopes, like service instances.

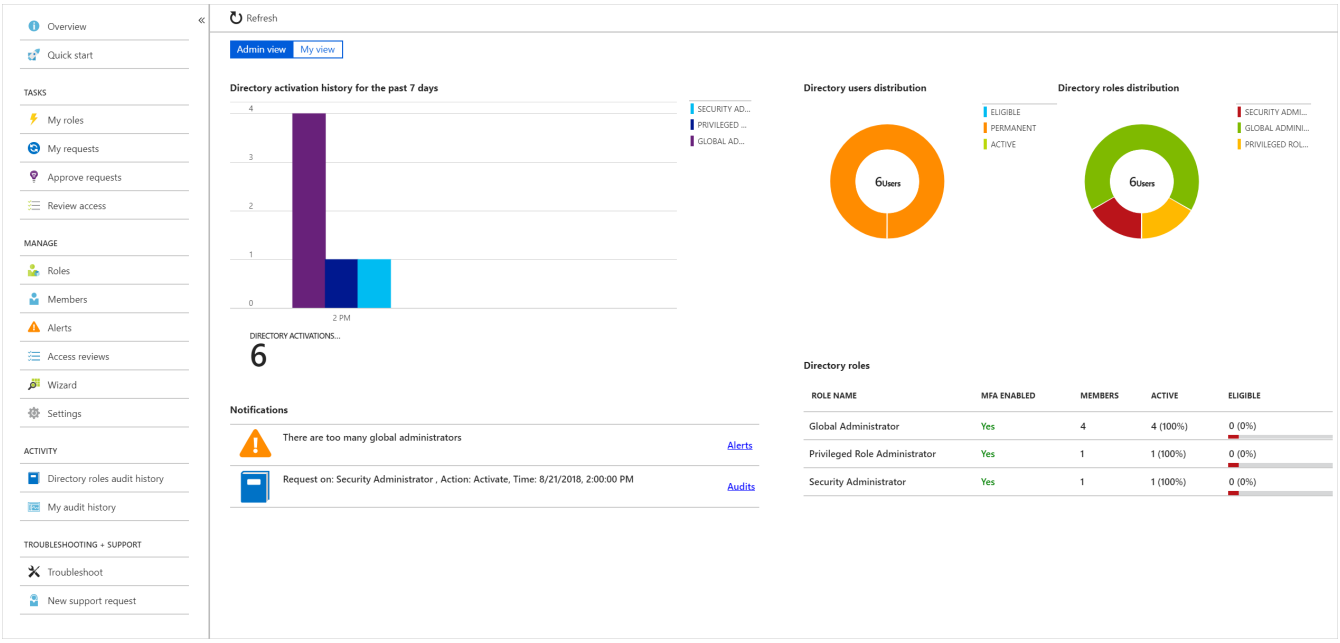
Management groups are an additional hierarchical level recently introduced into the RBAC model. Management groups add the ability to group subscriptions together and apply policy at an even higher level.

The ability to flow roles through an arbitrarily defined subscription hierarchy also allows administrators to grant temporary access to an entire environment for authenticated users. For example, an auditor might require temporary read-only access to all subscriptions.



Privileged Identity Management

In addition to managing Azure resource access with RBAC, a comprehensive approach to infrastructure protection should consider including the ongoing auditing of role members as the organization changes and evolves. Azure AD Privileged Identity Management (PIM) is an additional paid-for offering that provides oversight of role assignments, self-service, and just-in-time (JIT) role activation.



With the Azure AD PIM service, you can manage, control, and monitor access to important resources in your organization. This includes access to resources in Azure AD, Azure, and other Microsoft Online Services like Microsoft 365 and Microsoft Intune. This control does not eliminate the need for users to carry out privileged operations in Azure AD, Azure, Microsoft 365, and software as a service (SaaS) apps.

Organizations can give users JIT privileged access to Azure resources and Azure AD. Oversight is needed for what those users do with their administrator privileges. PIM helps mitigate the risk of excessive, unnecessary, or misused access rights.

Here are some of the key features of PIM:

- Providing just-in-time privileged access to Azure AD and Azure resources
- Assigning time-bound access to resources by using start and end dates
- Requiring approval to activate privileged roles
- Enforcing Azure AD multifactor authentication to activate any role
- Using justification to understand why users activate
- Getting notifications when privileged roles are activated
- Conducting access reviews to ensure that users still need roles
- Downloading an audit history for an internal or external audit

To use PIM, you need one of the following paid or trial licenses:

- Azure AD Premium P2
- Enterprise Mobility + Security (EMS) E5

Providing identities to services

It's often valuable for services to have identities. Often, and against best practices, credential information is embedded in configuration files. With no security around these configuration files, anyone with access to the systems or repositories can access these credentials and risk exposure.

Azure AD addresses this problem through two methods: service principals and managed identities for Azure services.

Service principals

To understand service principals, it's useful to first understand the words *identity* and *principal* as they're used in the world of identity management.

An *identity* is just a thing that can be authenticated. Obviously, this includes users with usernames and passwords. But it can also include applications or other servers, which might authenticate with secret keys or certificates. As a bonus definition, an *account* is data associated with an identity.

A *principal* is an identity that acts with certain roles or claims. Consider the use of Sudo on a Bash prompt or on Windows via **Run as administrator**. In both of those cases, you're still signed in as the same identity as before, but you've changed your role.

So, a *service principal* is literally named. It's an identity that a service or application uses. Like other identities, it can be assigned roles.

For example, your organization can assign its deployment scripts to run authenticated as a service principal. If that's the only identity that has permission to perform destructive actions, your organization has gone a long way toward making sure that it doesn't repeat the accidental resource deletion.

Managed identities for Azure resources

The creation of service principals can be a tedious process. There are also many touchpoints that can make maintaining service principals difficult. Managed identities for Azure resources are much easier, and will do most of the work for you.

A managed identity can be instantly created for any Azure service that supports it. (The list is constantly growing.) When you create a managed identity for a service, you're creating an account on the Azure AD tenant. Azure infrastructure will automatically take care of authenticating the service and managing the account. You can then use that account like any other Active Directory account, including letting the authenticated service securely access other Azure resources.

Check your knowledge

1. Which of the following scopes cannot be secured with Azure role-based access control?

- ○ Subscription
- ◉ Resource group

  ✗ **Role-based access control can be applied at the resource group scope.**
- ○ Files and folders within a Linux file system

  ✓ **Azure role-based access control can be applied to only Azure resources. Files and folders within a Linux file system can be secured with various methods, but not with Azure role-based access control.**
- ○ Resource

2. True or false: a managed identity for Azure resources can be assigned to a virtual machine to give it rights to start and stop other virtual machines.

- ◉ True

  ✓ **Managed identities for Azure resources are a method of assigning an identity to services. Through this assignment, role-based access control can be granted to manage Azure services, such as starting and stopping virtual machines.**
- ○ False

Next unit: Encryption

Continue >

How are we doing? ☆ ☆ ☆ ☆ ☆

200 XP

Encryption

10 minutes

Data is an organization's most valuable and irreplaceable asset. Encryption serves as the last and strongest line of defense in a layered security strategy for data.

Imagine you work for a healthcare organization that stores large amounts of sensitive data. The organization recently experienced a breach that exposed the unencrypted sensitive data of patients. The organization is now fully aware that it has gaps in its data-protection capabilities. It wants to understand how it could have better used encryption to protect itself and its patients from this type of incident.

Here, we'll take a look at what encryption is, how to approach the encryption of data, and what encryption capabilities are available on Azure.

What is encryption?

Encryption is the process of making data unreadable and unusable. To use or read the encrypted data, it must be *decrypted*, which requires the use of a secret key. There are two top-level types of encryption: *symmetric* and *asymmetric*.

Symmetric encryption uses the same key to encrypt and decrypt the data. Consider a password manager application. You enter your passwords, and they're encrypted with your own personal key. (Your key is often derived from your master password.) When the data needs to be retrieved, the same key is used and the data is decrypted.

Asymmetric encryption uses a public key and private key pair. Either key can encrypt but can't decrypt its own encrypted data. To decrypt, you need the paired key. Asymmetric encryption is used for things like TLS (used in HTTPS) and data signing.

Both symmetric and asymmetric encryption play a role in properly securing your data.

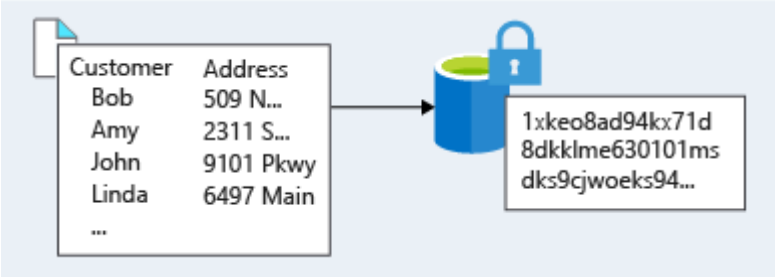
Encryption is typically approached in two ways: encryption at rest and encryption in transit.

Encryption at rest

Data at rest is the data that has been stored on a physical medium. This might be data stored on the disk of a server, data stored in a database, or data stored in a storage account.

Regardless of the storage mechanism, encryption of data at rest ensures that the stored data is unreadable without the keys and secrets needed to decrypt it. If an attacker obtained a hard drive with encrypted data and didn't have access to the encryption keys, the attacker would have great difficulty compromising the data. In such a scenario, an attacker would have to attempt attacks against encrypted data, which is much more complex and resource consuming than accessing unencrypted data on a hard drive.

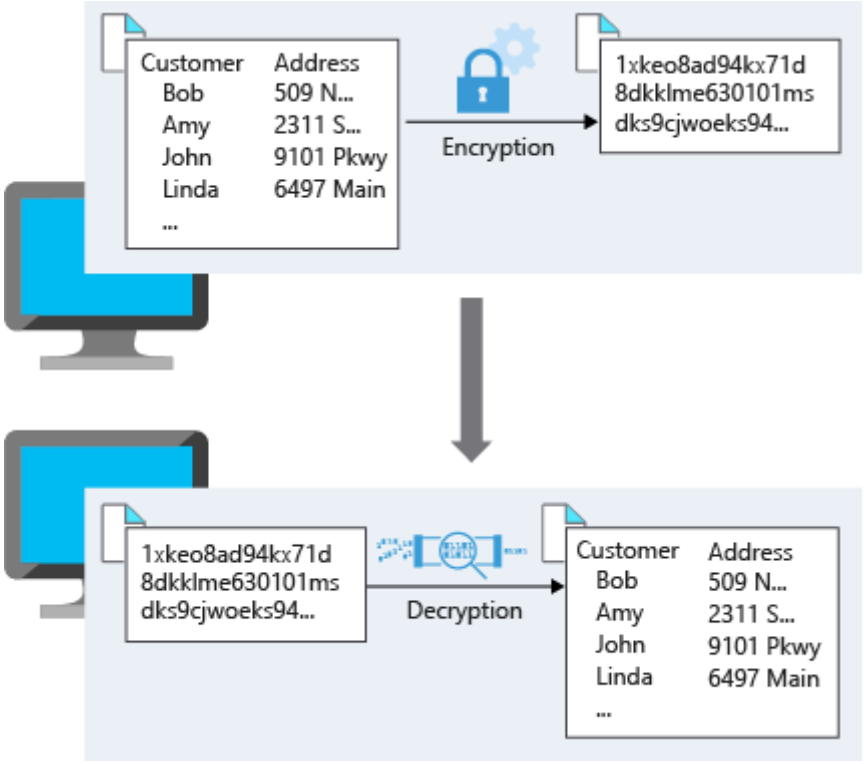
The data that's encrypted can vary in its content, usage, and importance to the organization. It might be financial information that's critical to the business, intellectual property that has been developed by the business, personal data that the business stores about customers or employees, and even the keys and secrets used for the encryption of the data itself.



Encryption in transit

Data in transit is the data that's actively moving from one location to another, such as across the internet or through a private network. An organization can handle secure transfer by encrypting the data before sending it over a network, or setting up a secure channel to transmit unencrypted data between two systems. Encrypting data in transit protects the data from outside observers and provides a mechanism to transmit data while limiting risk of exposure.

The following illustration is an example of encryption in transit. The data is encrypted before it's transferred. After the data reaches the destination, it's decrypted.



Identify and classify data

Let's revisit the problem that your organization is trying to solve. The organization has had previous incidents that exposed sensitive data, so there's a gap between what's being encrypted and what should be encrypted. The organization needs to start by identifying and classifying the types of data that it's storing, and align this with the business and regulatory requirements for the storage of data.

It's beneficial to classify this data as it relates to the impact of exposure to the organization, its customers, or its partners. An example classification might be as follows:

Data classification	Explanation	Examples
Restricted	Data classified as restricted poses significant risk if exposed, altered, or deleted. This data requires strong levels of protection.	Data that contains Social Security numbers, credit card numbers, personal health records
Private	Data classified as private poses moderate risk if exposed, altered, or deleted. This data requires reasonable levels of protection. Data that isn't classified as restricted or public will be classified as private.	Personal records that contain information such as address, phone number, academic records, customer purchase records

Data classification	Explanation	Examples
Public	Data classified as public poses no risk if exposed, altered, or deleted. This data doesn't require any protection.	Public financial reports, public policies, product documentation for customers

By taking an inventory of the types of data being stored, the organization can get a better picture of where sensitive data might be stored and where existing encryption might or might not be happening.

A thorough understanding of the regulatory and business requirements that apply to data that the organization stores is also important. The regulatory requirements to which an organization adhere often drive a large part of the data encryption requirements.

Your organization is storing sensitive data that falls under the Health Insurance Portability and Accountability Act (HIPAA), which contains requirements on how to handle and store patient data. Other industries fall under different regulatory requirements. A financial institution might store account information that falls within Payment Card Industry (PCI) standards. An organization that does business in the EU might fall under the General Data Protection Regulation (GDPR), which defines the handling of personal data in the EU.

Business requirements might also dictate that any data that could put the organization at financial risk needs to be encrypted. Competitive information falls in this category.

After you've classified the data and defined your requirements, you can take advantage of tools and technologies to implement and enforce encryption in your architecture.

Encryption on Azure

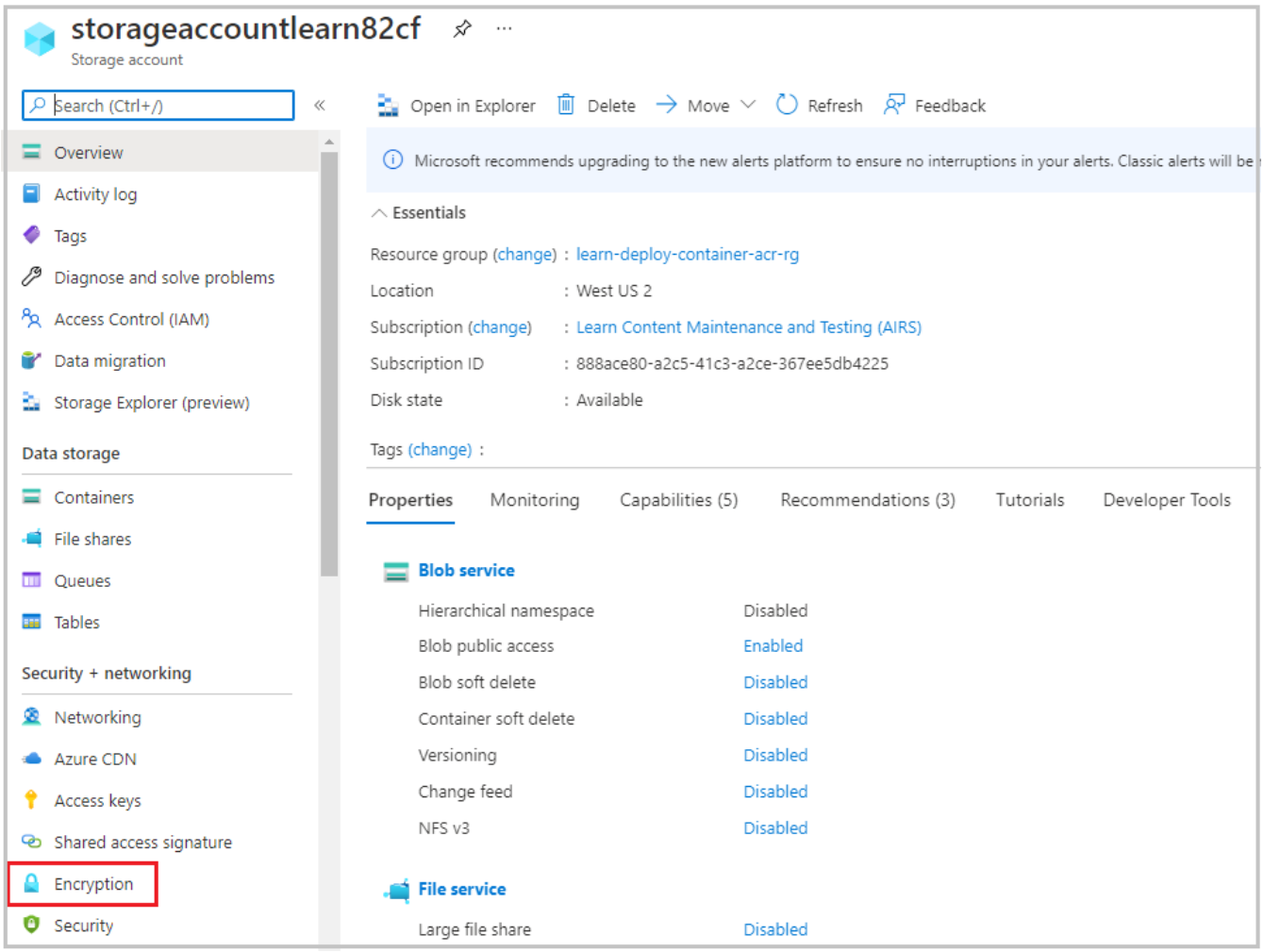
Let's take a look at some ways that Azure enables you to encrypt data across services.

Encrypt raw storage

Azure Storage encryption for data at rest helps you protect your data to meet your organizational security and compliance commitments. The Azure Storage platform automatically encrypts your data with 256-bit Advanced Encryption Standard (AES) encryption before persisting it to disk and then decrypts the data during retrieval. This handling of encryption, encryption at rest, decryption, and key management in Azure Storage is

transparent to applications that use the service. You don't need to add any code or turn on any features.

You can use Microsoft-managed encryption keys with Azure Storage encryption, or you can use your own encryption keys by selecting the option in the Azure portal.



Azure Storage automatically encrypts data in:

- All Azure Storage services, including Azure Managed Disks, Azure Blob Storage, Azure Files, Azure Queue Storage, and Azure Table Storage
- Both performance tiers (Standard and Premium)
- Both deployment models (Azure Resource Manager and classic)

For your organization, Azure Storage encryption means that whenever someone is using services that support Azure Storage encryption, their data is encrypted on the physical medium of storage. In the unlikely event that someone gets access to the physical disk, the data will be unreadable.

Encrypt virtual machines

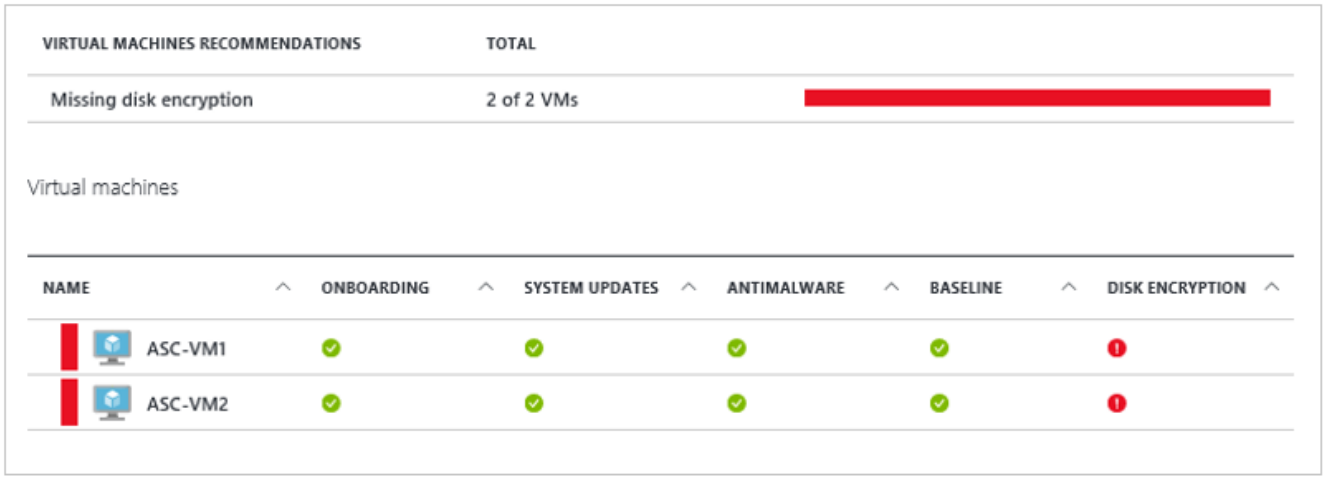
Azure Storage provides low-level encryption protection for data written to physical disk, but how do you protect the virtual hard disks (VHDs) of virtual machines (VMs)? If a malicious attacker gained access to your Azure subscription and exfiltrated the VHDs of your virtual machines, how would you ensure they'd be unable to access data stored on the VHD?

Azure Disk Encryption is a capability that helps you encrypt your Windows and Linux IaaS virtual machine disks. Azure Disk Encryption uses the industry-standard BitLocker feature of Windows and the DM-Crypt feature of Linux to provide volume encryption for the OS and data disks. The solution is integrated with Azure Key Vault to help you control and manage the disk-encryption keys and secrets. (And you can use managed identities for Azure services for accessing the key vault.)

Disk Encryption for Windows IaaS and Linux VMs is in general availability in all Azure public regions and Azure Government regions for Standard and Premium VMs. When you apply the Disk Encryption management solution, you can satisfy the following business needs:

- IaaS VMs are secured at rest through industry-standard encryption technology to address organizational security and compliance requirements.
- IaaS VMs start under customer-controlled keys and policies. You can audit their usage in your key vault.

In addition, if you use Microsoft Defender for Cloud, you're alerted if you have VMs that aren't encrypted. The alerts appear as High Severity, and the recommendation is to encrypt these VMs.



Your organization can apply Disk Encryption to its virtual machines to be sure that any data stored on VHDs is secured to organizational and compliance requirements. Because startup disks are also encrypted, the organization can control and audit usage.

Encrypt databases

Your organization has several databases that store data that needs more protection. The organization has moved many databases to Azure SQL Database, and wants to ensure that data is encrypted there. The organization wants to make sure that if the data files, log files, or backup files are stolen, they're unreadable without access to the encryption keys.

Transparent data encryption helps protect Azure SQL Database and Azure Data Warehouse against the threat of malicious activity. It performs real-time encryption and decryption of the database, associated backups, and transaction log files at rest without requiring changes to the application. By default, transparent data encryption is enabled for all newly deployed Azure SQL databases.

Transparent data encryption encrypts the storage of an entire database by using a symmetric key called the database encryption key. By default, Azure provides a unique encryption key per logical SQL Server instance and handles all the details. *Bring your own key* is also supported with keys stored in Azure Key Vault.

Because transparent data encryption is enabled by default, your organization can be confident that it has the proper protections in place for data stored in its databases.

For its on-premises SQL Server databases, your organization has turned on the SQL Server Always Encrypted feature. Always Encrypted is designed to protect sensitive data, such as client personal information or financial data. This feature helps protect column data at rest and in transit by having the client application handle the encryption and decryption outside the SQL Server database through an installed driver. This allows your organization to minimize exposure of data, because the database never works with unencrypted data.

The Always Encrypted client driver performs the encryption and decryption processes. It rewrites the T-SQL queries as necessary to encrypt data passed to the database and decrypt the results, while keeping these operations transparent to the application.

Encrypt secrets

We've seen that the encryption services all use keys to encrypt and decrypt data. How do we ensure that the keys themselves are secure? You might also have passwords, connection strings, or other sensitive pieces of information that you need to securely store.

Azure Key Vault is a cloud service that works as a secure store for secrets. Key Vault allows you to create multiple secure containers, called vaults. These vaults are backed by hardware security modules (HSMs). Vaults help reduce the chances of accidental loss of security information by centralizing the storage of application secrets. Vaults also control and log access to anything stored in them.

Azure Key Vault can handle requesting and renewing Transport Layer Security (TLS) certificates, to provide a robust certificate lifecycle management solution. Key Vault is designed to support any type of secret. These secrets can be passwords, database credentials, API keys, and certificates.

Because you can grant Azure Active Directory identities access to use Key Vault secrets, applications that use managed identities for Azure services can automatically and seamlessly acquire the secrets they need.

Your organization can use Key Vault for the storage of all of its sensitive application information. That information includes the TLS certificates that the organization uses to secure communication between systems.

Encrypt backups

Encrypting all of its data won't help your organization if the daily backups of systems aren't also encrypted. Your organization uses Azure Backup to back up data from on-premises machines and Azure VMs. Azure Backup lets the IT department back up and recover data at a granular level. The backups include files, folders, machine system state, and app-aware data.

Luckily for your hard-working IT department, there's no work to do here because all the data is stored encrypted at rest. Azure Backup encrypts local backups by using AES256 and a key created from the passphrase configured by the administrator. The data is securely transferred to Azure through HTTPS. The already-encrypted data is then stored on disk. Azure VMs are also automatically encrypted at rest because they use Azure Storage for their disks.

Check your knowledge

1. True or false: only Windows virtual machines can use Azure Disk Encryption.

- ○ True
- ◉ False

✔ **Azure Disk Encryption is available for both Windows and Linux virtual machines.**

2. When you're classifying data, which of the following is a factor?

- ○ Level of risk posed to customers if exposed

✔ **Classifying data that should be encrypted is commonly based on the impact it can have to customers if it's exposed.**

- ◉ Method of data transport

  ✗ **Classifying data that should be encrypted is commonly based on the impact it can have to customers if it's exposed.**

- ○ Whether the data is stored on virtual machines or in a database
- ○ The amount of data stored

Next unit: Network security

Continue >

How are we doing? ☆ ☆ ☆ ☆ ☆

200 XP

Network security

10 minutes

Securing your network from attacks and unauthorized access is an important part of any architecture. As part of preparation for its cloud migration, your company took the time to plan its network infrastructure. The company wanted to ensure that it had network security controls in place to protect the network infrastructure from attack.

Here, we'll look at what network security is, how to integrate a layered approach into your architecture, and how Azure can help you provide network security for your environment.

What is network security?

Network security is protecting the communication of resources within and outside your network. The goal is to limit exposure at the network layer across your services and systems. By limiting this exposure, you decrease the likelihood that your resources can be attacked. For network security, an organization can focus its efforts on the following areas:

- *Securing traffic flow between applications and the internet* focuses on limiting exposure outside your network. Network attacks will most often start outside your network, so by limiting the internet exposure and securing the perimeter, you can reduce the risk of being attacked.
- *Securing traffic flow among applications* focuses on data between applications and their tiers, between different environments, and in other services within your network. By limiting exposure between these resources, you reduce the effect that a compromised resource can have. This can help reduce further propagation within the network.
- *Securing traffic flow between users and an application* focuses on securing the network flow for your users. This limits the exposure that your resources have to outside attacks, and it provides a secure mechanism for users to utilize your resources.

Layered approach to network security

A common thread throughout this module has been taking a layered approach to security, and this approach is no different at the network layer. It's not enough to just focus on securing the network perimeter, or focusing on the network security between services inside a network.

A layered approach provides multiple levels of protection, so that if an attacker gets through one layer, further protections are in place to limit the attack.

Let's look at how Azure can provide the tools for a layered approach to securing your network footprint.

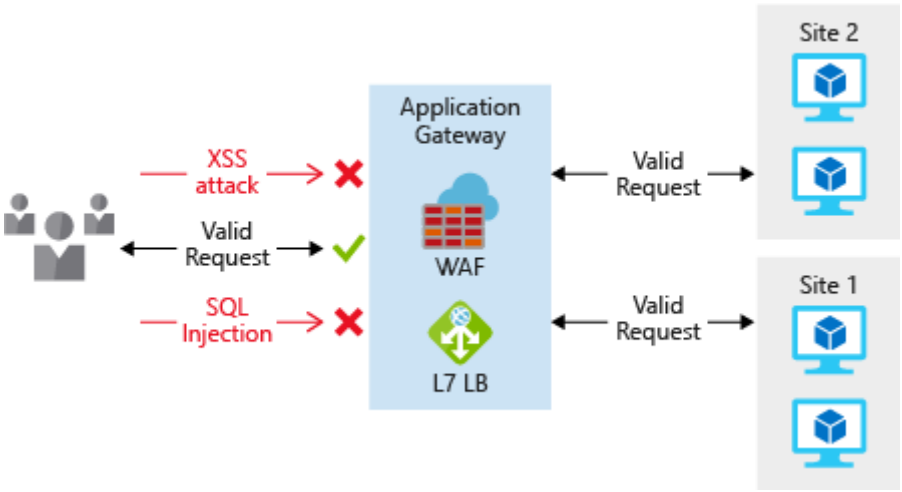
Internet protection

If you start on the perimeter of the network, you're focused on limiting and eliminating attacks from the internet. A great place to start is to assess the resources that are internet-facing, and allow inbound and outbound communication only where necessary. Identify all resources that are allowing inbound network traffic of any type. Ensure that they're necessary and restricted to only the required ports and protocols.

You can look for this information in Microsoft Defender for Cloud, which will identify internet-facing resources that don't have network security groups associated with them. It will also identify resources that aren't secured behind a firewall.

There are a couple of ways to provide inbound protection at the perimeter. Azure Application Gateway is a Layer 7 load balancer that also includes a web application firewall (WAF) to provide advanced security for your HTTP-based services. The WAF is based on rules from the OWASP 3.0 or 2.2.9 core rule sets. It provides protection from commonly known vulnerabilities such as cross-site scripting and SQL injection.

In the following diagram, the WAF feature of the application gateway protects the system from malicious attacks. The load balancer distributes the legitimate requests among virtual machines.

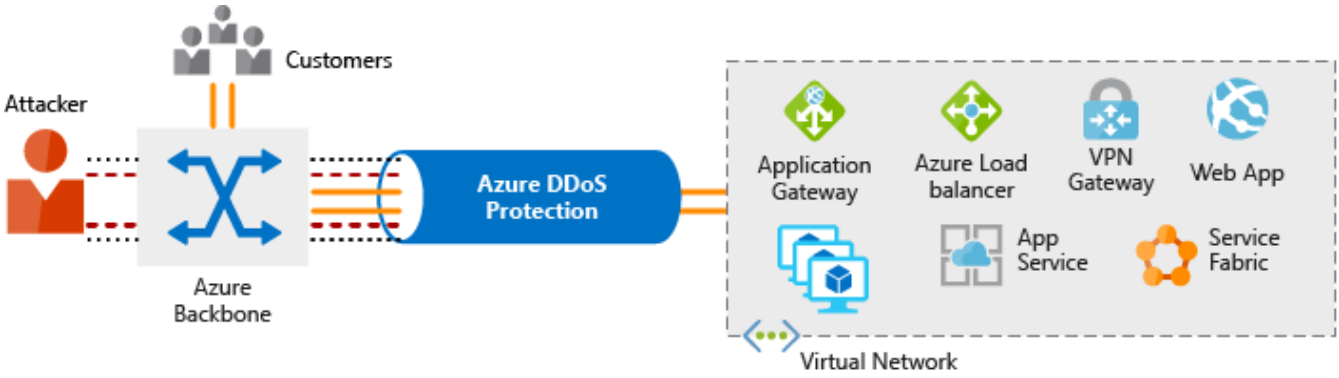


For protection of non-HTTP-based services or for increased customization, you can use network virtual appliances (NVAs) to secure your network resources. NVAs are similar to firewall appliances that you might find in on-premises networks, and are available from popular network security vendors. NVAs can provide greater customization of security for

those applications that require it. But they increase complexity, so we recommend that you carefully consider your requirements.

Any resource exposed to the internet is at risk for a denial-of-service attack. These types of attacks try to overwhelm a network resource by sending so many requests that the resource becomes slow or unresponsive.

To mitigate these attacks, Azure DDoS Protection provides basic protection across all Azure services and enhanced protection for further customization for your resources. DDoS Protection blocks attack traffic and forwards legitimate traffic to its intended destination. Within a few minutes of attack detection, you're notified through Azure Monitor metrics.

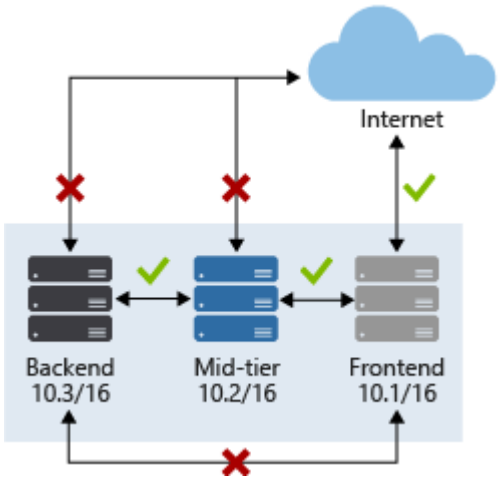


Virtual network security

Inside a virtual network, it's important to limit communication between resources to only what's required.

For communication between virtual machines, network security groups are a critical piece to restrict unnecessary communication. Network security groups operate at layers 3 and 4. They provide a list of allowed and denied communication to and from network interfaces and subnets. Network security groups are fully customizable, and they enable you to lock down network communication to and from your virtual machines. By using network security groups, you can isolate applications between environments, tiers, and services.

The following diagram shows how a network security group restricts the back end and middle tier from communicating directly with the internet. The front end receives the internet requests and then passes them to the middle tier. The middle tier communicates with the back end.



To isolate Azure services to allow communication only from virtual networks, use virtual network service endpoints. With service endpoints, you can secure Azure service resources to your virtual network.

Securing service resources to a virtual network provides improved security by fully removing public internet access to resources and allowing traffic only from your virtual network. This technique:

- Reduces the attack surface for your environment
- Reduces the administration required to limit communication between your virtual network and Azure services
- Provides optimal routing for this communication

Network integration

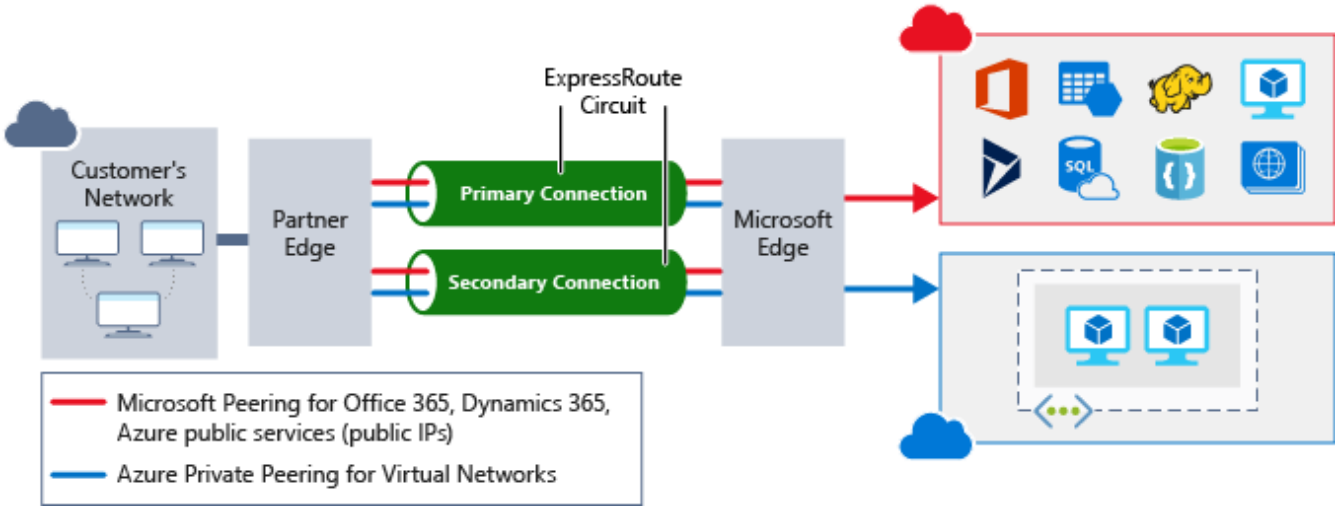
It's common to have existing network infrastructure that needs to be integrated to provide communication from on-premises networks, or to provide improved communication between services in Azure. There are a few key ways to handle this integration and improve the security of your network.

Virtual private network (VPN) connections are a common way of establishing secure communication channels between networks. This is no different when you're working with virtual networking on Azure. Connection between Azure virtual networks and an on-premises VPN device is a great way to provide secure communication between your network and your virtual machines on Azure.

To provide a dedicated, private connection between your network and Azure, you can use Azure ExpressRoute. ExpressRoute lets you extend your on-premises networks into the Microsoft cloud over a private connection facilitated by a connectivity provider.

With ExpressRoute, you can establish connections to Microsoft cloud services, such as Azure, Microsoft 365, and Dynamics 365. This improves the security of your on-premises

communication by sending this traffic over the private circuit instead of over the internet. You don't need to allow access to these services for your users over the internet, and you can send this traffic through appliances for further traffic inspection.



To easily integrate multiple virtual networks in Azure, virtual network peering establishes a direct connection between designated virtual networks. After a connection is established, you can use network security groups to provide isolation between resources in the same way that you secure resources within a virtual network. This integration gives you the ability to provide the same fundamental layer of security across any peered virtual networks. Communication is allowed only between directly connected virtual networks.

Check your knowledge

1. Azure network security groups can be used to secure communication between which of the following?

- ○ Communication between Azure virtual machines and the internet
- ◉ Communication between Azure virtual machines within a virtual network

  ✗ **Network security groups can be used to secure this communication flow, but this is not the only correct choice.**
- ○ Communication between Azure virtual machines and systems in an on-premises network
- ○ All of the above

  ✓ **Network security groups are fully customizable, and they give you the ability to fully lock down network communication to and from your virtual machines.**

2. Which of the following is not a method for protecting internet-facing services from network attacks?

- ○ Azure DDoS Protection
- ○ Azure Application Gateway WAF
- ◉ Azure Disk Encryption

  ✔ **Azure Disk Encryption protects your virtual machine VHDs from exposure, but it doesn't provide protection from network-based attacks.**

- ○ A network virtual appliance

Next unit: Application security

Continue >

How are we doing? ☆ ☆ ☆ ☆ ☆

100 XP

Application security

8 minutes

Hosting applications on a cloud platform provides advantages over traditional on-premises deployments. The cloud's shared-responsibility model moves security at the physical network, building, and host levels under the control of the cloud provider. An attacker who tries to compromise the platform at this level would see diminishing returns versus the considerable investment and insight that providers make in securing and monitoring their infrastructure.

It's far more effective for attackers to pursue vulnerabilities introduced at the application level by cloud-platform customers. Furthermore, by adopting platform as a service (PaaS) to host their applications, customers can free resources from managing operating system security and deploy them to harden application code and monitor the identity perimeter around the application.

Here, we'll discuss some of the ways that you can improve application security through design.

Scenario

Imagine you work for a healthcare organization whose customers require access to their personal medical records through an online web portal. Compliance with the Health Insurance Portability and Accountability Act (HIPAA) is mandatory, and puts the company at significant risk of financial penalties if a breach of personal data occurs. Securing the application and personal data with which it interacts is paramount.

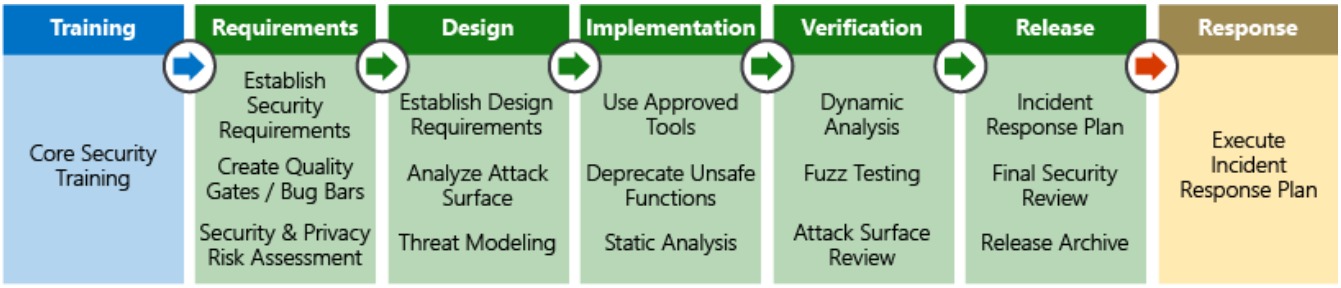
The primary areas that concern customer applications are:

- Secure application design
- Data security
- Identity and access management
- Endpoint security

Security Development Lifecycle

You can use the Microsoft Security Development Lifecycle (SDL) process during the application-design stage to ensure that security concerns are incorporated in the software

development lifecycle. Security and compliance issues are far easier to address when you're designing an application, and you can mitigate many common errors that can lead to security flaws in the final product. Fixing issues early in the software development journey is also far less costly. A software project can use this typical sequence of SDL steps:



The SDL is as much a cultural aspect as it is a process or set of tools. Building a culture where security is a primary focus and requirement of any application development can make great strides in evolving an organization's capabilities around security.

Operational security assessment

After an application has been deployed, it's essential to continually evaluate its security posture, determine how to mitigate any issues that are discovered, and feed the knowledge back into the software-development cycle. The depth to which an organization performs this evaluation is a factor of the maturity level of the software-development and operational teams, as well as the data-privacy requirements.

Software services that scan for security vulnerabilities are available to help automate this process and assess security concerns on a regular cadence. Such services offer these benefits without burdening teams with costly manual processes, such as penetration testing.

Microsoft Defender for Cloud is a free service that's now enabled by default for all Azure subscriptions. It's tightly integrated with other Azure application-level services, such as Azure Application Gateway and Azure Web Application Firewall. By analyzing logs from these services, Defender for Cloud can report on known vulnerabilities in real time and recommend responses to mitigate them. You can even configure Defender for Cloud to automatically execute playbooks in response to attacks.

Identity as the perimeter

Identity validation is becoming the first line of defense for applications. Restricting access to a web application by authenticating and authorizing sessions can drastically reduce the attack surface area.

Azure Active Directory (Azure AD) and Azure Active Directory B2C (Azure AD B2C) offer an effective way to offload the responsibility of identity and access to a fully managed service. Azure AD Conditional Access policies, Privileged Identity Management, and identity protection controls further enhance your ability to prevent unauthorized access and audit changes.

Data protection

Customer data is the target for most, if not all, attacks against web applications. The secure storage and transport of data between an application and its data storage layer is paramount.

Your organization stores and accesses sensitive patient medical record data. HIPAA, enacted by the United States Congress in 1996, among other controls, defines the national standards for electronic healthcare transactions by healthcare providers and employers. Healthcare providers and employers must ensure that patients and authorized parties, such as physicians, have secure access to medical data.

To comply with these requirements, your organization has modified its applications to encrypt all patient data at rest and in transit. For example, the organization uses Transport Layer Security (TLS) to encrypt data exchanged between the web application and back-end SQL databases. Data is also encrypted at rest in SQL Server through transparent data encryption. Encryption at rest ensures that even if the environment is compromised, data is effectively useless to anyone without the correct decryption keys.

To encrypt data stored in Azure Blob Storage, you can use client-side encryption to encrypt the data in memory before it's written to the storage service. Libraries that support this encryption are available for .NET, Java, and Python. These libraries enable the integration of data encryption directly into applications to enhance data integrity.

Secure key and secret storage

Separating application secrets (like connection strings or passwords) and encryption keys from the application that's used to access data is vital. Encryption keys and application secrets should never be stored in the application code or configuration files.

Instead, use a secure store such as Azure Key Vault. Access to this sensitive data can then be limited to application identities through managed identities for Azure resources. You can rotate keys on a regular basis to limit exposure if encryption keys are leaked.

You can also choose to use your own encryption keys generated by on-premises hardware security modules (HSMs). You can even mandate that Azure Key Vault instances are implemented in single-tenant, discrete HSMs.