Welcome to Module 2: Introducing Cloud Architecting.

Module overview

Sections

1. What is cloud architecting?
2. The Amazon Web Services (AWS) Well-Architected Framework
3. Best practices for building solutions on AWS
4. AWS global infrastructure

Knowledge check

This module includes the following sections:

1. What is cloud architecting?
2. The Amazon Web Services (AWS) Well-Architected Framework
3. Best practices for building solutions on AWS
4. AWS global infrastructure

At the end of this module, you will be asked to complete a knowledge check that will test your understanding of key concepts covered in this module.

Module objectives

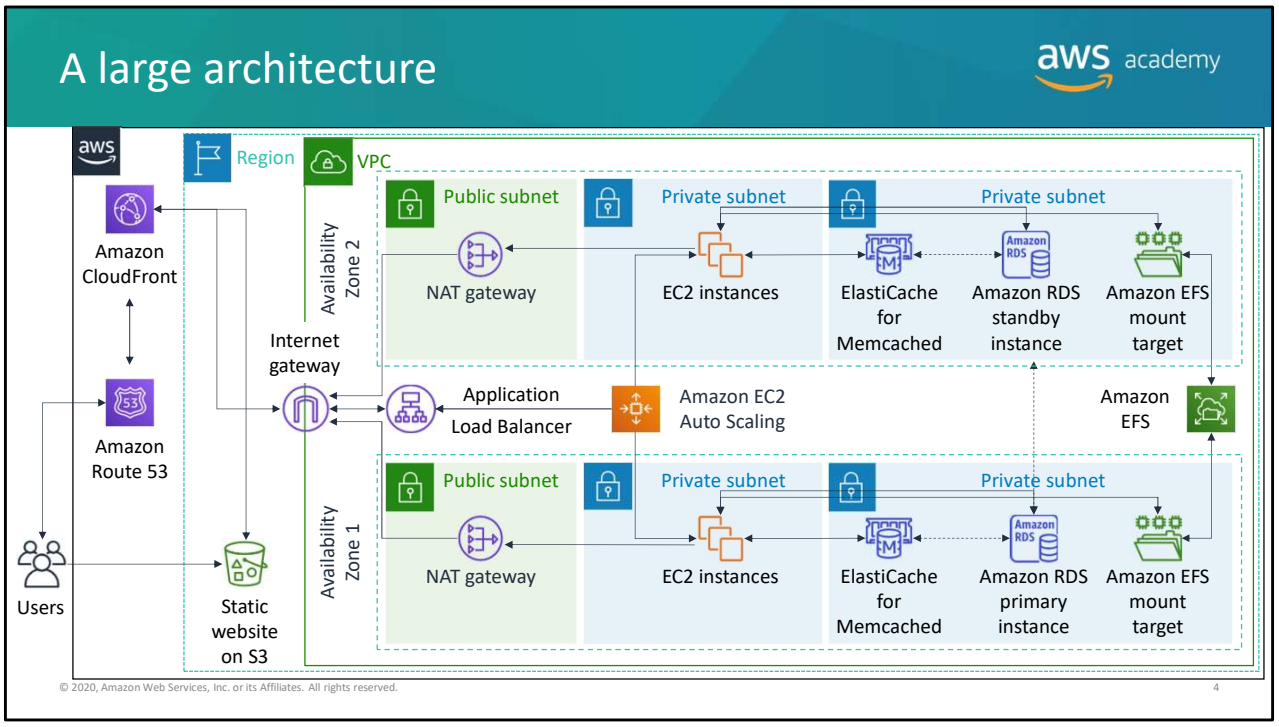
aws academy

At the end of this module, you should be able to:

- Define cloud architecture
- Describe how to design and evaluate architectures using the AWS Well-Architected Framework
- Explain best practices for building solutions on AWS
- Describe how to make informed decisions on where to place AWS resources

At the end of this module, you should be able to:

- Define cloud architecture
- Describe how to design and evaluate architectures using the AWS Well-Architected Framework
- Explain best practices for building solutions on AWS
- Describe how to make informed decisions on where to place AWS resources



By the end of this course, you will have learned about all the components in this architecture diagram. You should also be able to construct your own solution architectures that are as large and robust as this example. You will see this diagram repeated at the start of most modules in the course. New components in the diagram will be revealed as they are introduced in the course.

Module 2: Introducing Cloud Architecting

Section 1: What is cloud architecting?

aws academy

Introducing Section 1: What is cloud architecting?

Architectural need

Around 2000, Amazon was struggling to make its new shopping website highly available and scalable.

To understand what cloud architecting is and why it's important, first consider an example of what software development is like in its absence.

Around 2000, Amazon was trying to create an ecommerce service that would enable third-party sellers to build their own online shopping sites on top of the Amazon ecommerce engine. The company was struggling to make its new shopping website highly available and scalable.

Origins of AWS

- According to AWS CEO Andy Jassy, at the time, Amazon ecommerce tools were “a jumbled mess”
  - Applications and architectures were built without proper planning
  - It was difficult to separate services from each other
- Solution: Amazon created a set of well-documented APIs, which became the company’s standard for service development

In a TechCrunch interview about the genesis of AWS, AWS Chief Executive Officer (CEO) Andy Jassy said that in the beginning, the Amazon ecommerce tools were a “jumbled mess.” Applications and architectures were being built without proper planning. Jassy also said that it was “a huge challenge to separate the various services to make a centralized development platform.”

The solution to this problem was to create a set of well-documented application programming interfaces (APIs) to organize the development environment.

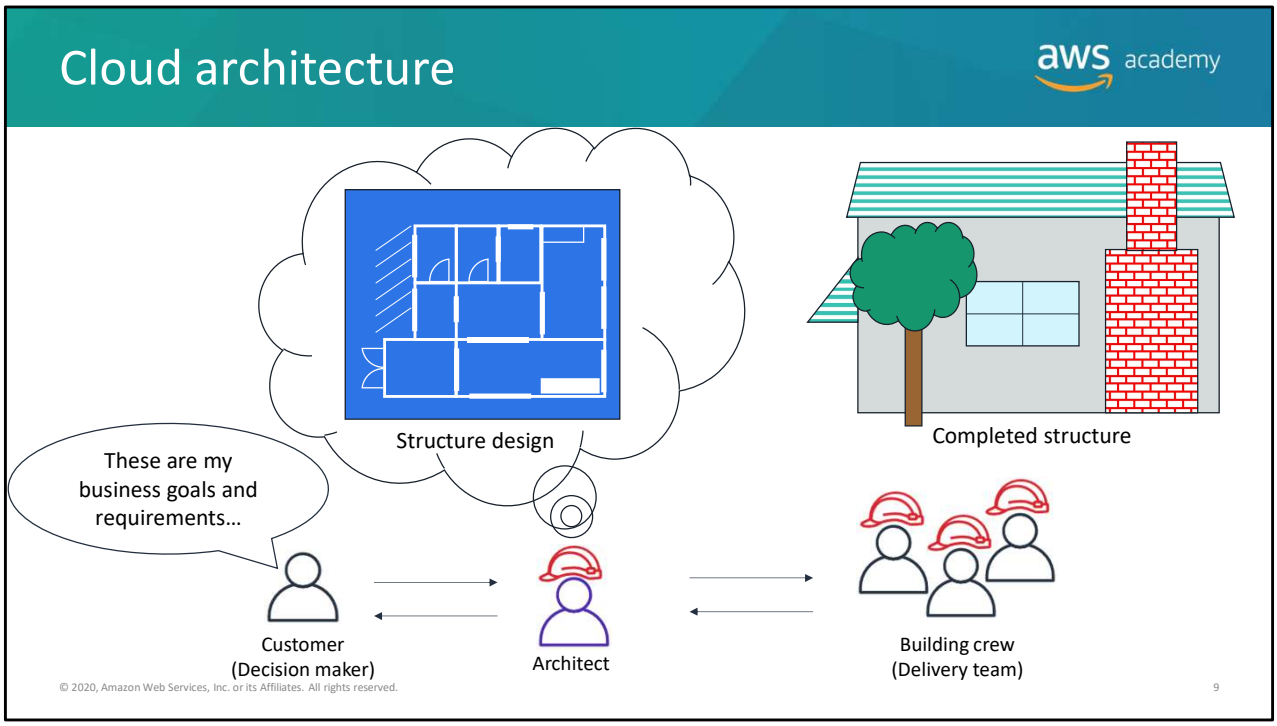
Problems persisted

- Amazon still struggled to build applications quickly.
  - Database, compute, and storage components took 3 months to build.
  - Each team built their own resources, with no planning for scalability or re-usability.
- Solution: Amazon built internal services to create highly available, scalable, and reliable architectures on top of its infrastructure. In 2006, Amazon started selling these services as AWS.

However, Amazon still struggled to build applications quickly as the company grew and more software engineers were hired:

- It took 3 months to build database, compute, and storage components for an entire project that was expected to take 3 months.
- Each team built their own resources without planning for scalability or reusability.

The solution was to build internal services to create highly available, scalable, and reliable architectures on top of the Amazon infrastructure. In 2006, Amazon started selling these services as AWS.



So, what is cloud architecture? Cloud architecture is the practice of applying cloud characteristics to a solution that uses cloud services and features to meet an organization's technical needs and business use cases. A solution is similar to a blueprint for a building.

Software systems require architects to manage their size and complexity.

Cloud architects:

- Engage with decision makers to identify the business goals and the capabilities that need improvement.
- Ensure alignment between technology deliverables of a solution and the business goals.
- Work with delivery teams that are implementing the solution to ensure that the technology features are appropriate.

Having well-architected systems increases the likelihood that the technology deliverables will help meet business goals.

Section 1 key takeaways

- Cloud architecture is the practice of applying cloud characteristics to a solution that uses cloud services and features to meet an organization's technical needs and business use cases
- You can use AWS services to create highly available, scalable, and reliable architectures

Some key takeaways from this section of the module include:

- Cloud architecture is the practice of applying cloud characteristics to a solution that uses cloud services and features to meet an organization's technical needs and business use cases
- You can use AWS services to create highly available, scalable, and reliable architectures

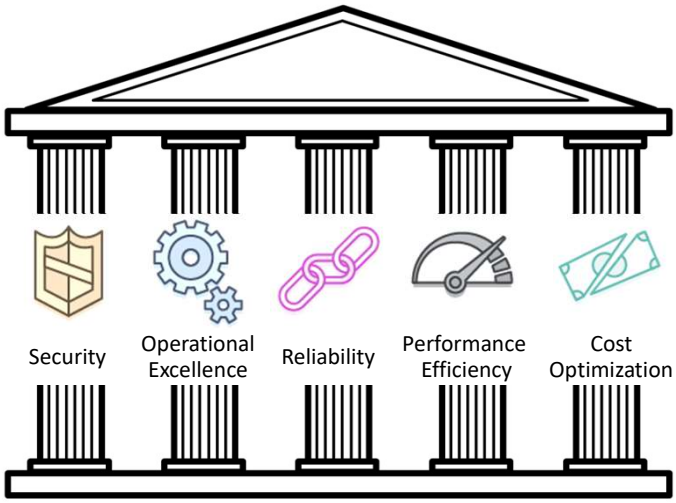
Module 2: Introducing Cloud Architecture

Section 2: The AWS Well-Architected Framework

aws academy

Introducing Section 2: The AWS Well-Architected Framework.

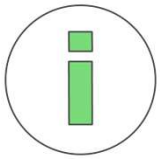
The AWS Well-Architected Framework



The AWS Well-Architected Framework is a guide that is designed to help you build the most secure, high-performing, resilient, and efficient infrastructure possible for your cloud applications and workloads. It provides a consistent approach to evaluate cloud architectures, and guidance to help implement designs. It documents a set of foundational questions and best practices that enable you to understand if a specific architecture aligns well with cloud best practices. AWS developed this framework after reviewing thousands of customer architectures on AWS.

The AWS Well-Architected Framework is organized into five pillars: Operational Excellence, Security, Reliability, Performance Efficiency, and Cost Optimization.

Security pillar



Identity foundation

Traceability

Security at all layers

Risk assessment and mitigation strategies

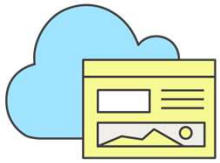
The Security pillar addresses the ability to protect information, systems, and assets while delivering business value through risk assessments and mitigation strategies.

Your architecture will present a much stronger security presence if you implement a strong identity foundation, enable traceability, apply security at all layers, automate security best practices, and protect data in transit and at rest.

For more information about security best practices, see the Security Pillar whitepaper.

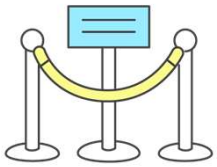
Operational Excellence pillar

- The ability to run and monitor systems
- To continuously improve supporting process and procedures



Deployed

Updated



Operated

The Operational Excellence pillar addresses the ability to run systems and gain insight into their operations to deliver business value. It also addresses the ability to continuously improve supporting processes and procedures.

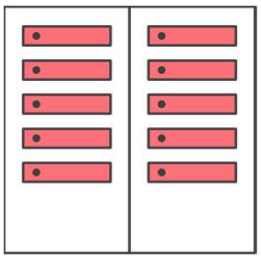
When you design a workload for operations, you must be aware of how it will be deployed, updated, and operated. Implement engineering practices that align with defect reductions and quick, safe fixes. Enable observation with logging, instrumentation, and business and technical metrics so that you can gain insight into what is happening inside your architecture.

In AWS, you can view your entire workload (applications, infrastructure, policy, governance, and operations) as code. It can all be defined in and updated using code. This means that you can apply the same engineering discipline that you use for application code to every element of your stack.

For more information about operational excellence best practices, see the Operational Excellence Pillar whitepaper.

Reliability pillar

- Recover quickly from infrastructure or service disruptions
- Dynamically acquire computing resources to meet demand
- Mitigate disruptions such as:
  - Misconfigurations
  - Transient network issues



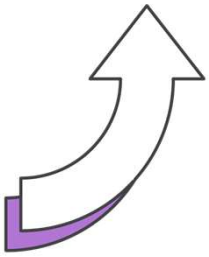
The Reliability pillar addresses the ability of a system to recover from infrastructure or service disruptions and dynamically acquire computing resources to meet demand. It also addresses the ability of a system to mitigate disruptions such as misconfigurations or transient network issues.

It can be difficult to ensure reliability in a traditional environment. Issues arise from single points of failure, lack of automation, and lack of elasticity. By applying the best practices outlined in the Reliability pillar, you can prevent many of these issues. It will help you and your customers to have a properly designed architecture with respect to high availability, fault tolerance, and overall redundancy.

For more information about reliability best practices, see the Reliability Pillar whitepaper.

Performance Efficiency pillar

- Choose efficient resources and maintain their efficiency as demand changes
- Democratize advanced technologies
- Employ mechanical sympathy



When you consider performance, you want to maximize your performance by using computation resources efficiently. You also want to maintain that efficiency as the demand changes.

It is also important to democratize advanced technologies. In situations where technology is difficult to implement yourself, consider using a vendor. By implementing the technology for you, the vendor handles the complexity and the knowledge, freeing your team to focus on more value-added work.

Mechanical sympathy is when you use a tool or system with an understanding of how it operates best. Use the technology approach that aligns best to what you are trying to achieve. For example, consider data access patterns when you select database or storage approaches.

For more information about performance best practices, see the Performance Efficiency Pillar whitepaper.

Cost Optimization pillar

- Measure efficiency
- Eliminate unneeded expense
- Consider using managed services

Cost optimization is an ongoing requirement of any good architectural design. The process is iterative, and it should be refined and improved throughout your production lifetime. Understanding how efficient your current architecture is in relation to your goals can remove unneeded expense. Consider using managed services because they operate at cloud scale, and they can offer a lower cost per transaction or service.

For more information, see the Cost Optimization Pillar whitepaper.

The AWS Well-Architected Tool

- Helps you review the state of your workloads and compares them to the latest AWS architectural best practices
- Gives you access to knowledge and best practices used by AWS architects, when you need it
- Delivers an action plan with step-by-step guidance on how to build better workloads for the cloud
- Provides a consistent process for you to review and measure your cloud architectures

If you would like help with designing a well-architected solution, you can use the AWS Well-Architected Tool. The AWS Well-Architected Tool is a self-service tool that provides you with on-demand access to current AWS best practices. These best practices can help you build secure, high-performing, resilient, and efficient application infrastructure on AWS.

The AWS Well-Architected Tool helps you review the state of your workloads and compare them to the latest AWS architectural best practices. It gives you access to knowledge and best practices used by AWS architects, when you need it.

This tool is available in the AWS Management Console. You define your workload and answer a series of questions in the areas of operational excellence, security, reliability, performance efficiency, and cost optimization. The AWS Well-Architected Tool then delivers an action plan with step-by-step guidance on how to improve your workload for the cloud.

The AWS Well-Architected Tool provides a consistent process for you to review and measure your cloud architectures. You can use the results that the tool provides to identify next steps for improvement, drive architectural decisions, and bring architecture considerations into your corporate governance process.

To learn more about the AWS Well-Architected Tool, see the AWS Well-Architected Tool website.

Section 2 key takeaways

- The AWS Well-Architected Framework provides a consistent approach to evaluate cloud architectures and guidance to help implement designs
- The AWS Well-Architected Framework is organized into five pillars
- Each pillar documents a set of foundational questions that enable you to understand if a specific architecture aligns well with cloud best practices
- The AWS Well-Architected Tool helps you review the state of your workloads and compares them to the latest AWS architectural best practices

Some key takeaways from this section of the module include:

- The AWS Well-Architected Framework provides a consistent approach to evaluate cloud architectures and guidance to help implement designs.
- The AWS Well-Architected Framework is organized into five pillars.
- Each pillar documents a set of foundational questions that enable you to understand if a specific architecture aligns well with cloud best practices.
- The AWS Well-Architected Tool helps you review the state of your workloads and compares them to the latest AWS architectural best practices.

Module 2: Introducing Cloud Architecting

Section 3: Best practices for building solutions on AWS

aws academy

Introducing Section 3: Best practices for building solutions on AWS.

Design tradeoffs

- Evaluate tradeoffs so you can select an optimal approach
- Examples of tradeoffs include:
  - Trade consistency, durability, and space for time and latency to deliver higher performance
  - Prioritize speed to market of new features over cost
- Base design decisions on empirical data

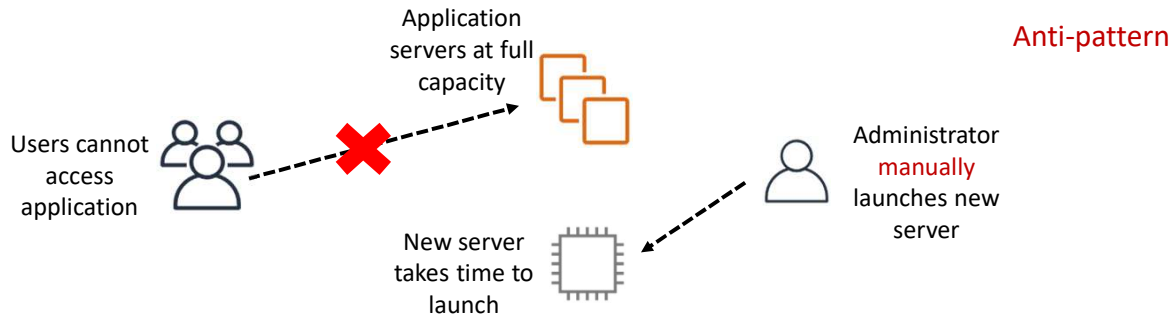
As you design a solution, think carefully about tradeoffs so that you can select an optimal approach. For example, you might trade consistency, durability, and space for time and latency to deliver higher performance. Or, you might prioritize speed to market over cost.

Tradeoffs can increase the cost and complexity of your architecture, so your design decisions should be based on empirical data. For example, you might need to perform load testing to ensure that a measurable benefit is obtained in performance. Or, you might need to perform benchmarking to achieve the most cost-optimal workload over time. When you evaluate performance-related improvements, you will also want to consider how your architecture design choices will impact customers and workload efficiencies.

In this section, you will learn about best practices for designing solutions on AWS. You will also learn about anti-patterns (or bad solution designs) to avoid.

1. Enable scalability (1 of 2)

Ensure that your architecture can handle changes in demand.

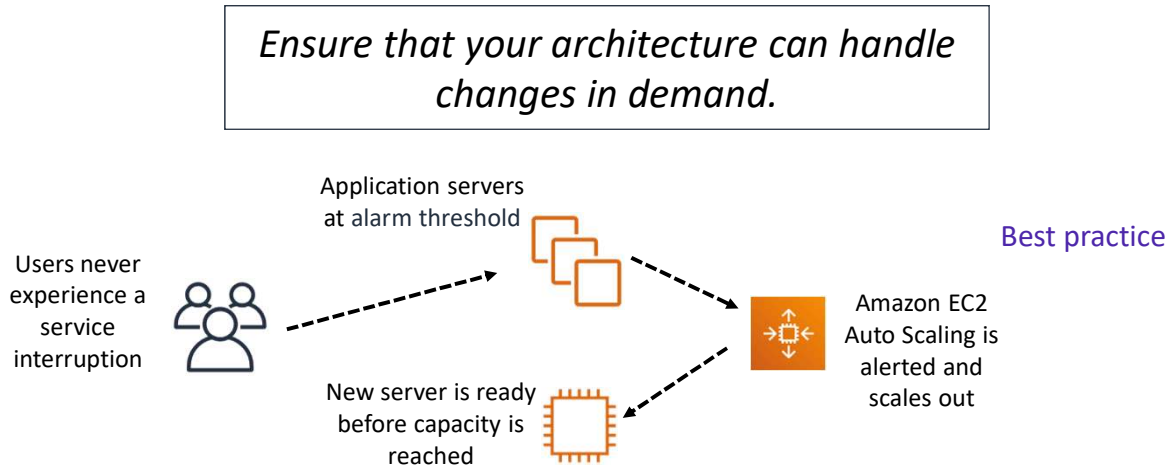


When you run your workloads on the AWS Cloud, you can scale your infrastructure quickly and proactively. Make sure that you implement scalability at every layer of your infrastructure.

To understand the importance of scalability, consider this anti-pattern, where scaling is done reactively and manually.

In this scenario, when application servers reach full capacity, users are prevented from accessing the application. Administrators then manually launch one or more new instances to manage the load. Unfortunately, it takes a few minutes for an instance to become available for use after it's launched. That increases the time that users can't access the application.

1. Enable scalability (2 of 2)



By enabling scalability, you can improve your design to anticipate the need for more capacity and deliver it before it's too late.

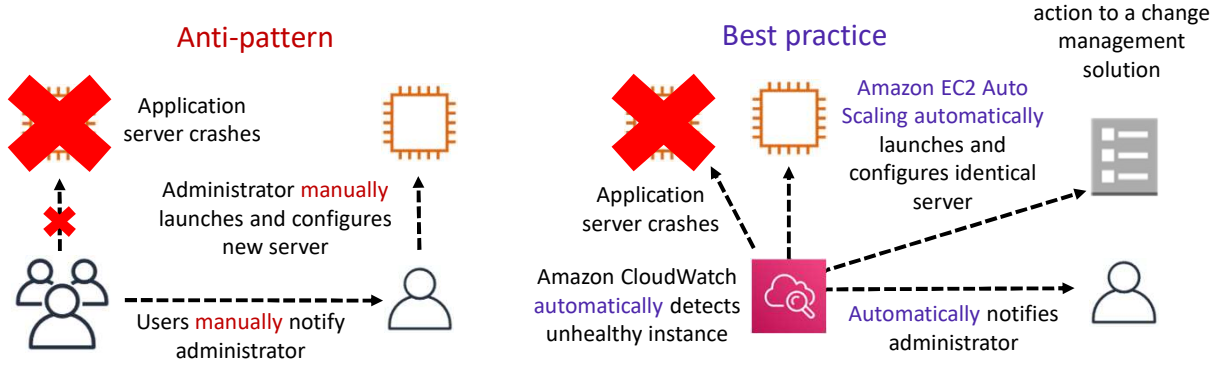
For example, you can use a monitoring solution like Amazon CloudWatch to detect whether the total load across your fleet of servers has reached a specified threshold. You can define this threshold to be *Stayed above 60% CPU utilization for longer than 5 minutes*, or anything related to the use of resources. With CloudWatch, you can also design custom metrics based on specific applications that can trigger the resource scaling that is required.

When an alarm is triggered, Amazon EC2 Auto Scaling immediately launches a new instance. That instance is then ready before capacity is reached, which provides a seamless experience for users.

Ideally, you should also design your system to *scale in* when demand drops off so that you're not running (and paying for) instances that you no longer need.

2. Automate your environment

Where possible, automate the provisioning, termination, and configuration of resources.



AWS offers built-in monitoring and automation tools at virtually every layer of your infrastructure. Take advantage of these tools to ensure that your infrastructure can respond quickly to changes.

You can use tools like CloudWatch and Amazon EC2 Auto Scaling to detect unhealthy resources and automate the launch of replacement resources. You can also be notified when resource allocations change.

3. Treat resources as disposable

Take advantage of the dynamically provisioned nature of cloud computing.

Anti-pattern	Best practice
• Over time, different servers end up with different configurations	• Automate deployment of new resources with identical configurations
• Resources run when they're not needed	• Terminate resources that are not in use
• Hardcoded IP addresses prevent flexibility	• Switch to new IP addresses automatically
• It can be difficult or inconvenient to test new updates on hardware that's in use	• Test updates on new resources, and then replace old resources with updated ones

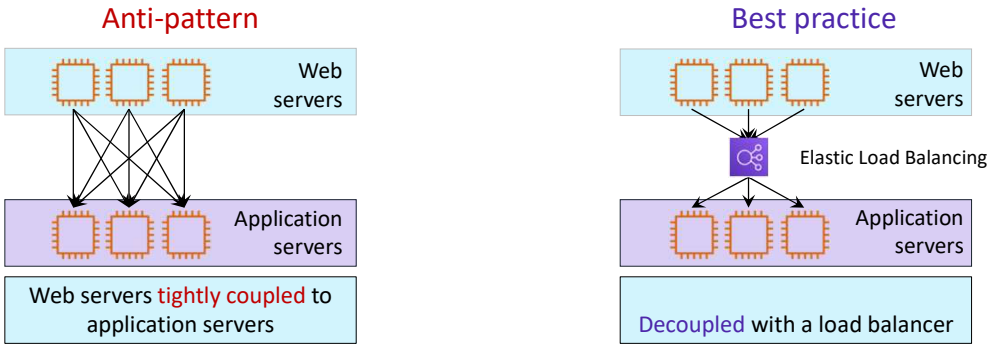
The best practice of treating resources as disposable refers to the idea of thinking about your infrastructure as software instead of hardware.

With hardware, it's easy to buy more specific components than you need so that you are prepared for spikes in usage. That's expensive and inflexible—it's harder to upgrade because of the sunk cost.

Instead, when you treat your resources as disposable, migrating between instances or other discrete resources is fairly straightforward. You can quickly respond to changes in capacity needs, upgrade applications, and manage the underlying software.

4. Use loosely coupled components

Design architectures with independent components.



Traditional infrastructures have chains of tightly integrated servers, each with a specific purpose. The problem is that when one of those components or layers goes down, the disruption to the system can be fatal. It also impedes scaling. If you add or remove servers at one layer, you must also connect every server on each connecting layer.

The example on the left illustrates a collection of web and application servers that are tightly coupled. If one application server goes down, an error will be caused because the web servers try and fail to connect to it.

With loose coupling, you use managed solutions as intermediaries between the layers of your system. With this design, the intermediary automatically handles both failures and the scaling of components or layers.

The example on the right shows a load balancer (in this case, an Elastic Load Balancing load balancer) that routes requests between the web servers and the application servers. If one application server goes down, the load balancer will automatically start directing all traffic to the two healthy servers.

Two primary solutions for decoupling your components are **load balancers** and **message queues**.

5. Design services, not servers

Use the breadth of AWS services.
Don't limit your infrastructure to servers.

Anti-pattern

- Simple applications run on persistent servers
- Applications communicate directly with one another
- Static web assets are stored locally on instances
- Backend servers handle user authentication and user state storage

Best practice

- When appropriate, consider using containers or a serverless solution
- Message queues handle communication between applications
- Static web assets are stored externally, such as on Amazon Simple Storage Service (Amazon S3)
- User authentication and user state storage are handled by managed AWS services

The next best practice is to design services, not servers. Although Amazon Elastic Compute Cloud (Amazon EC2) offers tremendous flexibility for designing and setting up your solution, it shouldn't always be the first (or only) solution that you use for every need. In some cases, containers or a serverless solution might be more appropriate. Therefore, it's important to consider what your needs are and which solution is appropriate.

With AWS serverless solutions and managed services, you don't need to provision, configure, and manage an entire Amazon EC2 instance.

Managed solutions that have a lower profile and are more performant can replace server-based solutions at a lower cost. A few examples are AWS Lambda, Amazon Simple Queue Service (Amazon SQS), Amazon DynamoDB, Elastic Load Balancing, Amazon Simple Email Service (Amazon SES), and Amazon Cognito.

6. Choose the right database solution

aws academy

Match technology to the workload, not the other way around.

Things to consider:

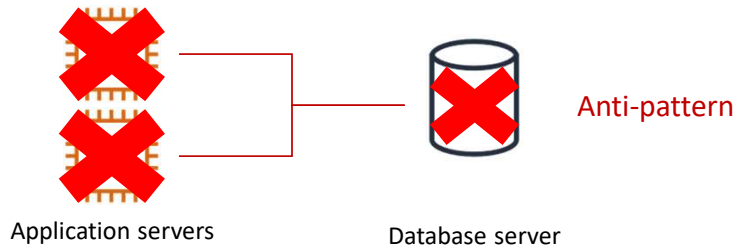
- Read and write needs
- Total storage requirements
- Typical object size and nature of access to these objects
- Durability requirements
- Latency requirements
- Maximum concurrent users to support
- Nature of queries
- Required strength of integrity controls

It is important that you choose the right database solution. In traditional data centers and on-premises environments, limits on available hardware and licenses can constrain your choice of a data store solution. AWS recommends that you choose a data store based on your needs for your application environment.

7. Avoid single points of failure (1 of 2)

Assume everything fails. Then, design backward.

Where possible, use redundancy to prevent single points from bringing down an entire system.

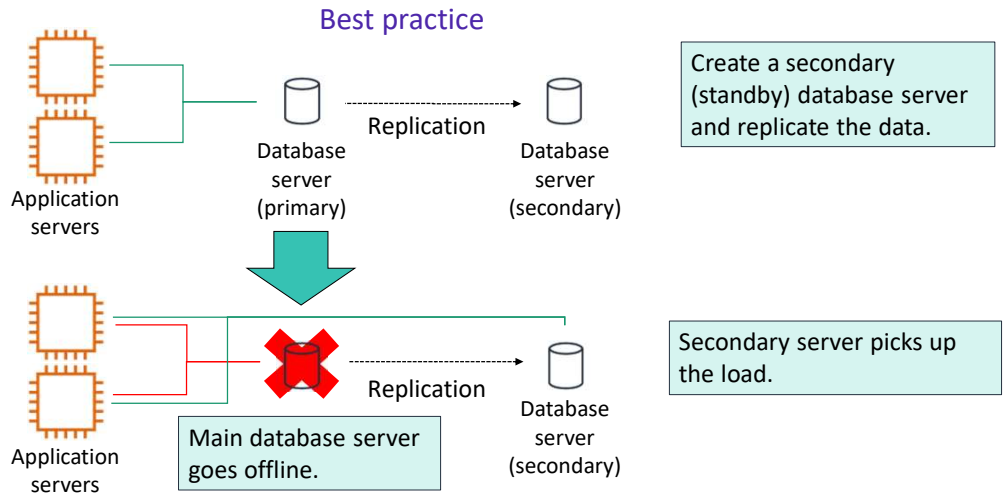


Where possible, eliminate single points of failure from your architecture. This doesn't mean that you must always duplicate every component. Depending on your downtime service-level agreements (SLAs), you can use automated solutions that only launch components when needed. You can also use a managed service, where AWS automatically replaces malfunctioning underlying hardware for you.

This simple system shows two application servers connected to a single database server. The database server represents a single point of failure and should be avoided. When it goes down, the application servers also go down.

Application servers should continue to function even if the underlying physical hardware fails, is removed, or replaced.

7. Avoid single points of failure (2 of 2)



A common way to avoid single points of failure is to create a secondary (standby) database server and replicate the data. This way, if the main database server goes offline, the secondary server can pick up the load.

In this example, when the main database goes offline, the application servers automatically send their requests to the secondary database. This example also exemplifies Best Practice #3: Treat resources as disposable, and design your applications to support changes in hardware.

8. Optimize for cost

aws academy

Take advantage of the flexibility of AWS to increase your cost efficiency.

Things to consider:

- Are my resources the right size and type for the job?
- What metrics should I monitor?
- How do I make sure to turn off resources that are not in use?
- How often will I need to use this resource?
- Can I replace any of my servers with managed services?

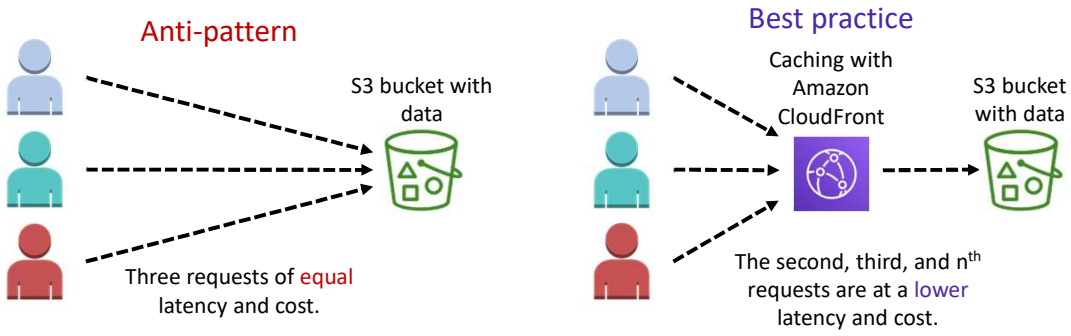
Cloud computing allows you to trade capital expense for variable expense. *Capital expenses (capex)* are funds that a company uses to acquire, upgrade, and maintain physical assets such as property, industrial buildings, or equipment. Under this model, you pay for the servers in the data center, whether they are active or not.

By contrast, AWS services use a *variable expense* cost model, which means that you pay only for the individual services you need, for as long as you use them. Within each service, you can optimize for cost. Many services offer different pricing tiers, models, or configurations.

Keep in mind that it can be very expense to replicate an on-premises data center setup of servers running 24/7 in the cloud. Therefore, the best way to build your infrastructure, from a cost perspective, is to only provision the resources that you need and stop services when they are not being used.

9. Use caching

Caching minimizes redundant data retrieval operations, improving performance and cost.



Caching is a technique to make future requests faster and reduce network throughput by temporarily storing data in an intermediary location between the requester and the permanent storage.

In the anti-pattern example, no caching service is used. When anyone requests a file from one of the Amazon Simple Storage Service (Amazon S3) buckets, each request takes the same amount of time to complete, and each request costs the same.

In the best-practice-pattern example, the infrastructure uses Amazon CloudFront in front of Amazon S3 to providing caching. In this scenario, the initial request checks for the file in Amazon CloudFront. If it is not found, CloudFront requests the file from Amazon S3. CloudFront then stores a copy of the file at an edge location close to the user, and sends a copy to the user who made the request. Subsequent requests for the file are retrieved from the (now closer) edge location in CloudFront instead of Amazon S3.

This reduces latency *and* cost because, after the first request, you no longer pay for the file to be transferred out of Amazon S3.

10. Secure your entire infrastructure

Build security into every layer of your infrastructure.

Things to consider:

- Isolate parts of your infrastructure
- Encrypt data in transit and at rest
- Enforce access control granularly, using the principle of least privilege
- Use multi-factor authentication (MFA)
- Use managed services
- Log access of resources
- Automate your deployments to keep security consistent

Security isn't only about getting through the outer boundary of your infrastructure. It also involves ensuring that your individual environments and their components are secured from each other.

For example, in Amazon EC2, you can create security groups that allow you to determine which ports on your instances can send and receive traffic. Security groups can also determine where that traffic can come from or go to.

You can use security groups to reduce the probability that a security threat on one instance will spread to every other instance in your environment. You should take similar precautions with other services. Specific ways to implement this best practice are discussed throughout the course.

Section 3 key takeaways

- As you design solutions, evaluate tradeoffs and base your decisions on empirical data
- Follow these best practices when building solutions on AWS –
  - Enable scalability
  - Automate your environment
  - Treat resources as disposable
  - Use loosely-coupled components
  - Design services, not servers
  - Choose the right database solution
  - Avoid single points of failure
  - Optimize for cost
  - Use caching
  - Secure your entire infrastructure

The key takeaways from this section of the module are:

- As you design solutions, evaluate tradeoffs and base your decisions on empirical data
- Follow these best practices when building solutions on AWS –
  - Enable scalability
  - Automate your environment
  - Treat resources as disposable
  - Use loosely-coupled components
  - Design services, not servers
  - Choose the right database solution
  - Avoid single points of failure
  - Optimize for cost
  - Use caching
  - Secure your entire infrastructure

Module 2: Introducing Cloud Architecting

Section 4: AWS global infrastructure

aws academy

Introducing Section 4: AWS global infrastructure.

AWS Regions

- An AWS Region is a geographical area
- Each AWS Region consists of two or more Availability Zones
- Communication between Regions uses AWS backbone network infrastructure
- You enable and control data replication across Regions

Example: London Region

The AWS Cloud infrastructure is built around Regions. AWS has 22 Regions worldwide. An *AWS Region* is a physical geographical location with two or more *Availability Zones*. Availability Zones in turn consist of one or more *data centers*.

AWS Regions are connected to multiple Internet Service Providers (ISPs). Regions are also connected to a private global network backbone, which provides lower cost and more consistent cross-Region network latency when compared with the public internet.

AWS Regions that were introduced before March 20, 2019 are *enabled* by default. Regions that were introduced after March 20, 2019—such as Asia Pacific (Hong Kong) and Middle East (Bahrain)—are *disabled* by default. You must enable these Regions before you can use them. You can use the AWS Management Console to enable or disable a Region.

Some Regions have restricted access. An AWS (**China**) account provides access to the Beijing and Ningxia Regions only. To learn more about AWS in China, see the AWS in China page. The isolated **AWS GovCloud (US)** Region is designed to allow US government agencies and customers to move sensitive workloads into the cloud by addressing their specific regulatory and compliance requirements.

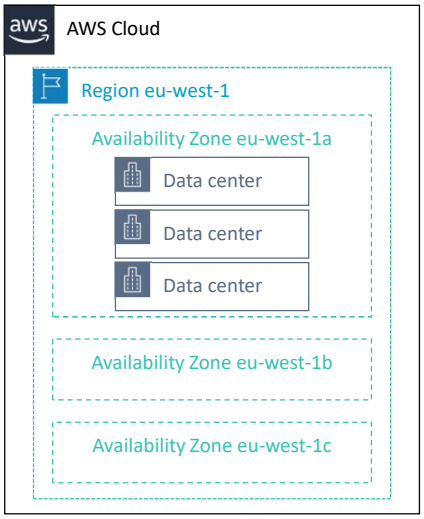
To achieve fault tolerance and stability, Regions are isolated from one another. Resources in one Region are not automatically replicated to other Regions. When you store data in a specific Region, it is not replicated outside that Region. It is your responsibility to replicate data across Regions, if your business needs require it. AWS provides information about the country and—where applicable—the state where each Region resides. You are responsible for selecting the Region where you should store data, based on your compliance and network latency requirements.

AWS products and services are available by Region, so you might not see all Regions available for a given service. For a list of AWS services offered by Region, see the Region Table.

For more information about the AWS global cloud infrastructure, see the Global Infrastructure website. For a current and interactive map of the AWS global infrastructure, see the Interactive AWS Global Infrastructure map.

AWS Availability Zones

- Each Availability Zone is –
  - Made up of one or more data centers
  - Designed for fault isolation
  - Interconnected with other Availability Zones in a Region using high-speed private links
- For certain services, you can choose your Availability Zones
- AWS recommends replicating across Availability Zones for resiliency



Each AWS Region consists of two or more isolated locations that are known as *Availability Zones*. Each Availability Zone comprises one or more data centers, with some Availability Zones having as many as six data centers. However, no data center can be a part of two Availability Zones.

Each Availability Zone is designed as an independent failure zone. This means that Availability Zones are physically separated within a typical metropolitan Region. They are also located in lower-risk flood plains (specific flood-zone categorization varies by Region). In addition to having a discrete uninterruptible power supply and onsite backup generation facilities, they are each fed via different grids from independent utilities to further reduce single points of failure. Availability Zones are all redundantly connected to multiple tier-1 transit providers.

An Availability Zone is the most granular level of specification that you can make for certain services, such as Amazon EC2.

You are responsible for selecting the Availability Zones where your systems will reside. Systems can span multiple Availability Zones. You should design your systems to survive temporary or prolonged failure of an Availability Zone if a disaster occurs. Distributing applications across multiple Availability Zones enables them to remain resilient in most failure situations, including natural disasters or system failures.

AWS Local Zones

- Enable you to run latency-sensitive portions of applications closer to end users and resources in a specific geography
- Are an extension of an AWS Region where you can use AWS services in geographic proximity to end users
- Let you place AWS compute, storage, database, and other select services closer to large population, industry, and IT centers where no Region exists today
- Are managed and supported by AWS
- Los Angeles (LA) AWS Local Zone is available by invitation

AWS Local Zones are a new type of AWS infrastructure deployment that places AWS compute, storage, database, and other select services closer to large population, industry, and IT centers where no AWS Region exists today. With AWS Local Zones, you can run latency-sensitive portions of applications closer to end users and resources in a specific geography. You can use AWS Local Zones to deliver single-digit millisecond latency for use cases such as media and entertainment content creation, real-time gaming, reservoir simulations, electronic design automation, and machine learning.

Each AWS Local Zone location is an extension of an AWS Region. You can run latency-sensitive applications in an AWS Local Zone by using AWS services such as Amazon EC2, Amazon Virtual Private Cloud (Amazon VPC), Amazon Elastic Block Store (Amazon EBS), Amazon FSx, and Elastic Load Balancing in geographic proximity to end users. AWS Local Zones provide a high-bandwidth, secure connection between local workloads and workloads that run in the AWS Region. Thus, AWS Local Zones enable you to seamlessly connect back to your other workloads that are running in AWS and connect to the full range of in-Region services through the same APIs and toolsets.

AWS Local Zones are managed and supported by AWS, and provide you with all the elasticity, scalability, and security benefits of the cloud. With AWS Local Zones, you can build and deploy latency-sensitive applications closer to your end users by using a consistent set of AWS services. You can also scale up or scale down, and you pay only for the resources that you use.

The Los Angeles AWS Local Zone is generally available by invitation today, and you can expect more Local Zones to come.

To learn more about AWS Local Zones, see the AWS Local Zones page.

AWS data centers

- Data centers are where the data resides and data processing occurs
- A data center typically has tens of thousands of servers
- All data centers are online and serving customers
- AWS custom network equipment –
  - Is sourced from multiple ODMs
  - Has a customized network protocol stack

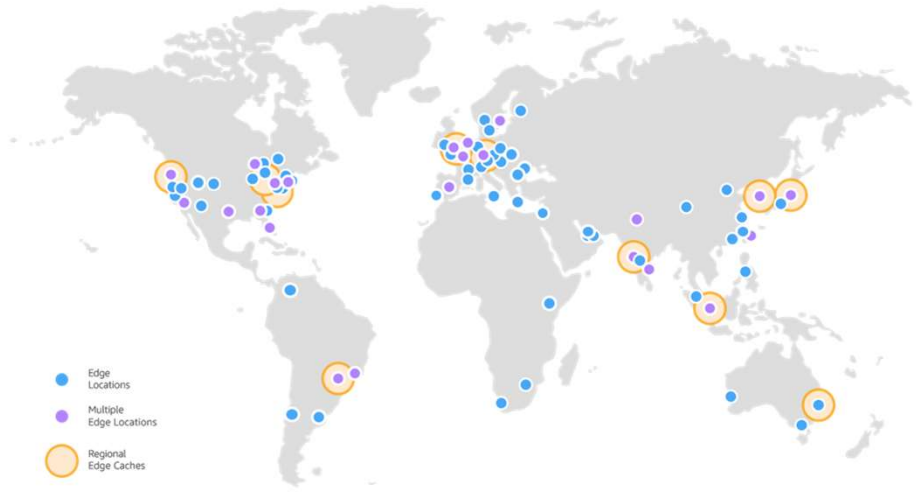
The foundation for the AWS infrastructure is the data centers. You do not specify a data center for the deployment of resources. However, a data center is the location where the actual data resides. Amazon operates state-of-the-art, highly available data centers. Though rare, failures that affect the availability of instances in the same location can occur. If you host all your instances in a single location that is affected by such a failure, none of your instances will be available.

All data centers are online and serving customers. In case of failure, automated processes move customer data traffic away from the affected area. Core applications are deployed in an N+1 configuration, so that in the event of a data center failure, there is sufficient capacity to enable traffic to be load balanced to the remaining sites.

AWS uses custom network equipment sourced from multiple original device manufacturers (ODMs). ODMs design and manufacture products based on specifications from a second company. The second company then rebrands the products for sale.

For more information about AWS data centers, see Learn how we secure AWS data centers by design.

AWS Points of Presence



To deliver content to end users with lower latency, Amazon CloudFront uses a global network that includes over 200 Points of Presence that are comprised of Edge Locations and Regional Edge Caches.

Edge locations are located in North America, Europe, Asia, Australia, South America, the Middle East, Africa, and China. Edge locations support AWS services like Amazon Route 53 and Amazon CloudFront.

Regional edge caches are used by default with Amazon CloudFront. They are used when you have content that is not accessed frequently enough to remain in an edge location. Regional edge caches absorb this content and provide an alternative to fetching the content from the origin server.

For more information about the Amazon CloudFront infrastructure, see **Amazon CloudFront infrastructure** at https://aws.amazon.com/cloudfront/features/?whats-new-cloudfront.sort-by=item.additionalFields.postDateTime&whats-new-cloudfront.sort-order=desc#Amazon_CloudFront_Infrastructure

Section 4 key takeaways

- The AWS global infrastructure consists of Regions, Availability Zones, and edge locations
- Your choice of a Region is typically based on compliance requirements or to reduce latency
- Each Availability Zone is physically separate from other Availability Zones and has redundant power, networking, and connectivity
- Edge locations and Regional edge caches improve performance by caching content closer to users

Some key takeaways from this section of the module include:

- The AWS global infrastructure consists of Regions and Availability Zones
- Your choice of a Region is typically based on compliance requirements or to reduce latency
- Each Availability Zone is physically separate from other Availability Zones and has redundant power, networking, and connectivity
- Edge locations and Regional edge caches improve performance by caching content closer to users

Course capstone project

The capstone project provides you with the opportunity to apply the skills and knowledge you develop in the course to a real-world scenario. Details about the project are included in the Bridging to certification module.

Module 2: Introducing Cloud Architecting

Module wrap-up

aws academy

It’s now time to review the module and wrap up with a knowledge check.

Module summary

In summary, in this module, you learned how to:

- Define cloud architecture
- Describe how to design and evaluate architectures using the AWS Well-Architected Framework
- Explain best practices for building solutions on AWS
- Describe how to make informed decisions on where to place AWS resources

In summary, in this module, you learned how to:

- Define cloud architecture
- Describe how to design and evaluate architectures using the AWS Well-Architected Framework
- Explain best practices for building solutions on AWS
- Describe how to make informed decisions on where to place AWS resources

Complete the knowledge check

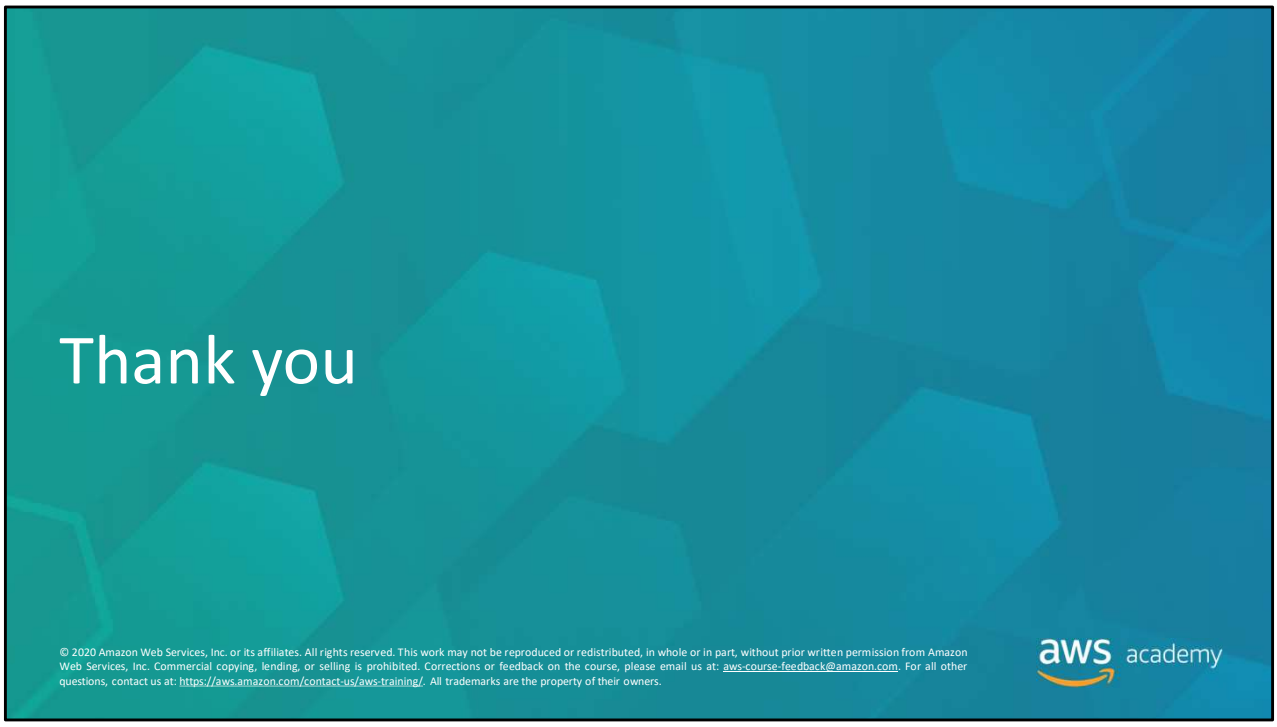
It is now time to complete the knowledge check for this module.

Additional resources

- AWS Global Infrastructure page
- Interactive AWS Global Infrastructure map
- AWS Well-Architected Framework whitepaper
- Security Pillar whitepaper
- Operational Excellence Pillar whitepaper
- Reliability Pillar whitepaper
- Performance Efficiency Pillar whitepaper
- Cost Optimization Pillar whitepaper

If you want to learn more about the topics covered in this module, you might find the following additional resources helpful:

- AWS Global Infrastructure page
- Interactive AWS Global Infrastructure map
- AWS Well-Architected Framework whitepaper
- Security Pillar whitepaper
- Operational Excellence Pillar whitepaper
- Reliability Pillar whitepaper
- Performance Efficiency Pillar whitepaper
- Cost Optimization Pillar whitepaper



Thank you for completing this module.