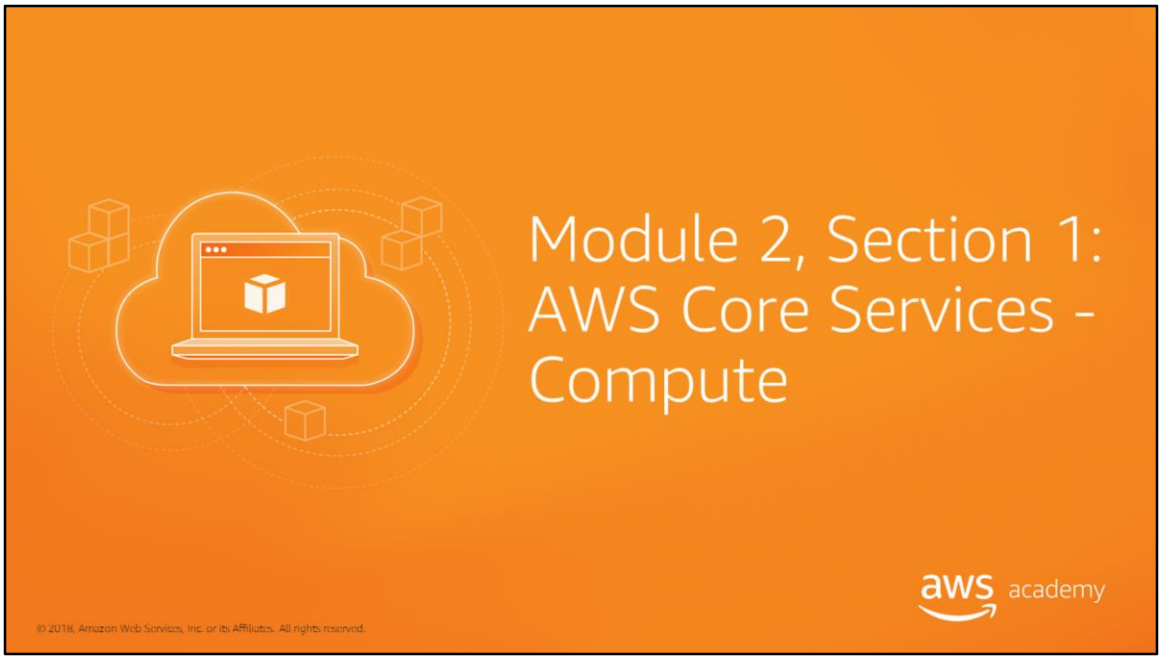
AWS Academy - Cloud Foundations
Module 02 Student Guide
Version 1.0.0

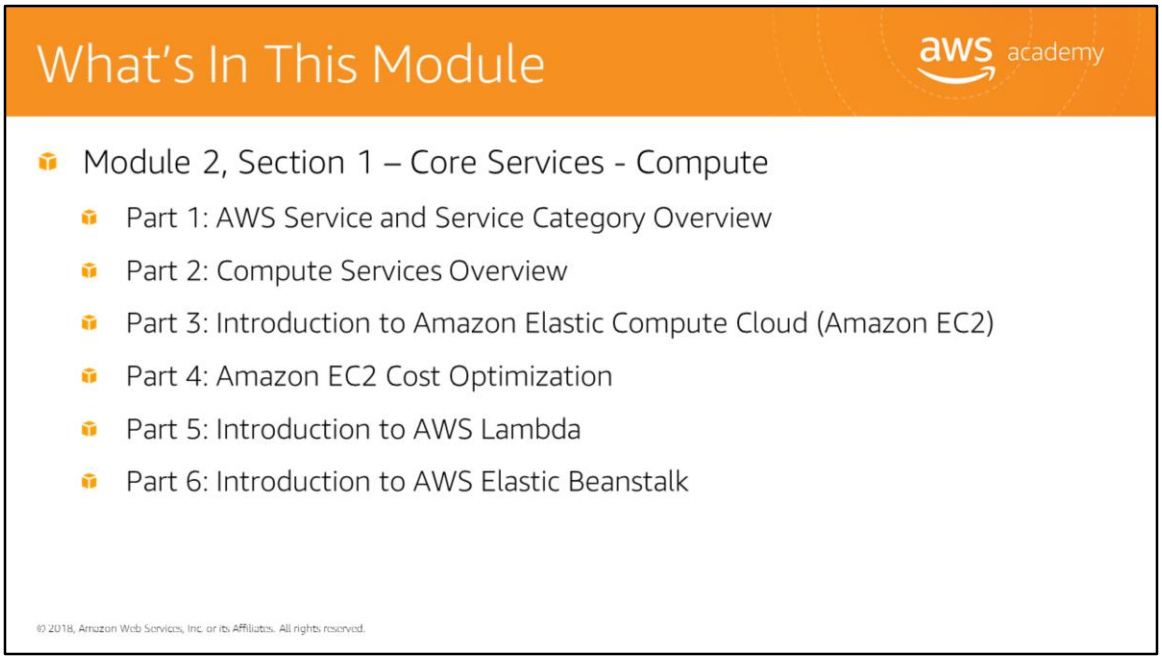
100-ACFNDS-10-EN-SG

Corrections or feedback on the course, please email us at:
aws-course-feedback@amazon.com.
For all other questions, contact us at:
https://aws.amazon.com/contact-us/aws-training/.

Contents

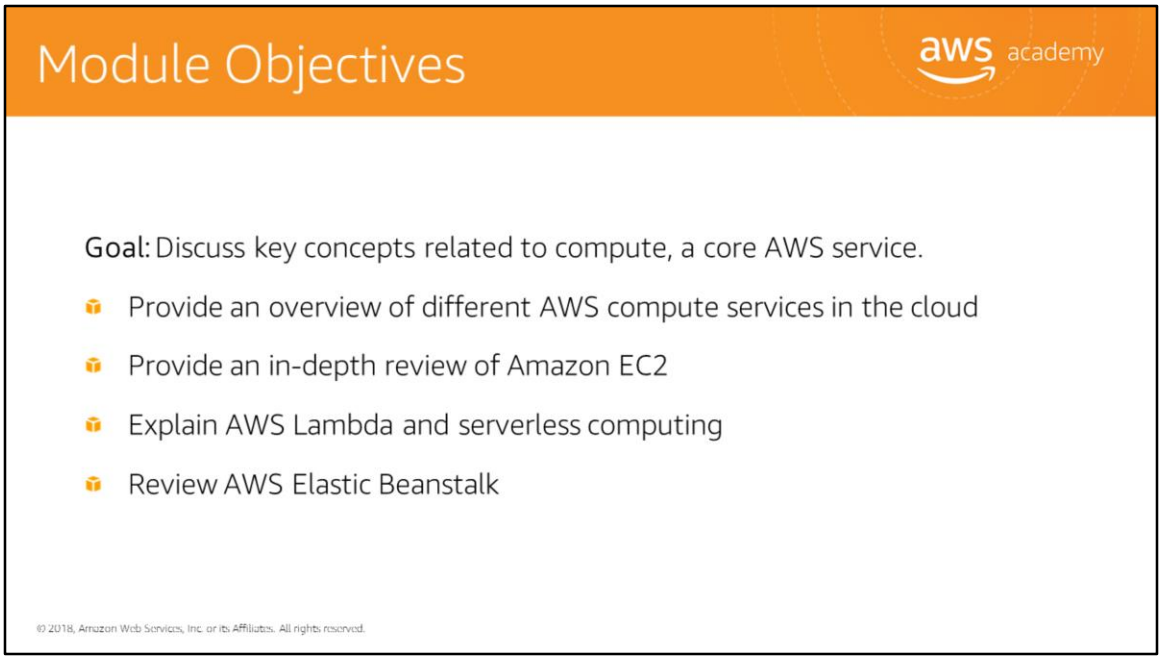


Welcome to Module 2, Section 1 – AWS Core Services – Compute. In this module, we will review AWS core services starting with Compute.



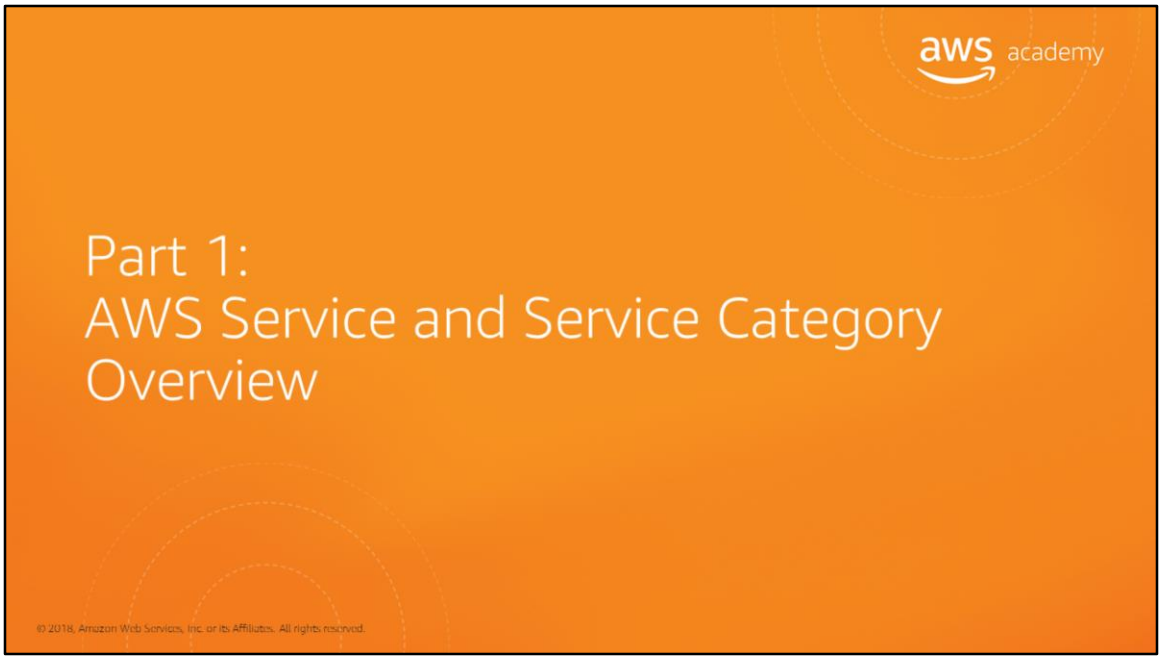
Amazon Web Services provides multiple services to build out a solution. Some of those services provide the foundation to all solutions. We refer to those as the *core services*. In the this module, we provide insight into the offerings of each service category and look at our first group of services, compute.

Whether you're building mobile apps or running massive clusters to sequence the human genome, building and running your business starts with compute. AWS has a broad catalog of compute services. Everything from simple application services to flexible virtual servers, and even serverless computing. This module introduces compute services.

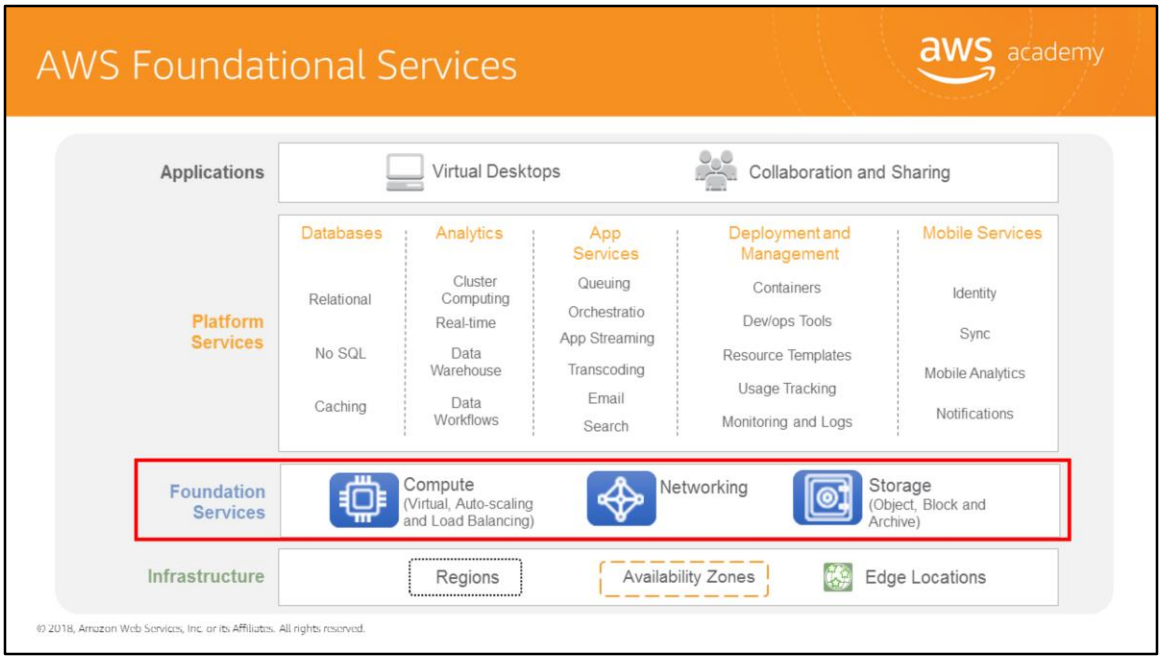


The goal of this module is to help you understand the compute resources that are available to power your solution. We will also review pricing options so you can begin to understand how different choices impact the cost of your solution.

Then, you have an opportunity to jump back into the lab. You will create an Amazon EC2 instance and launch Microsoft Windows.

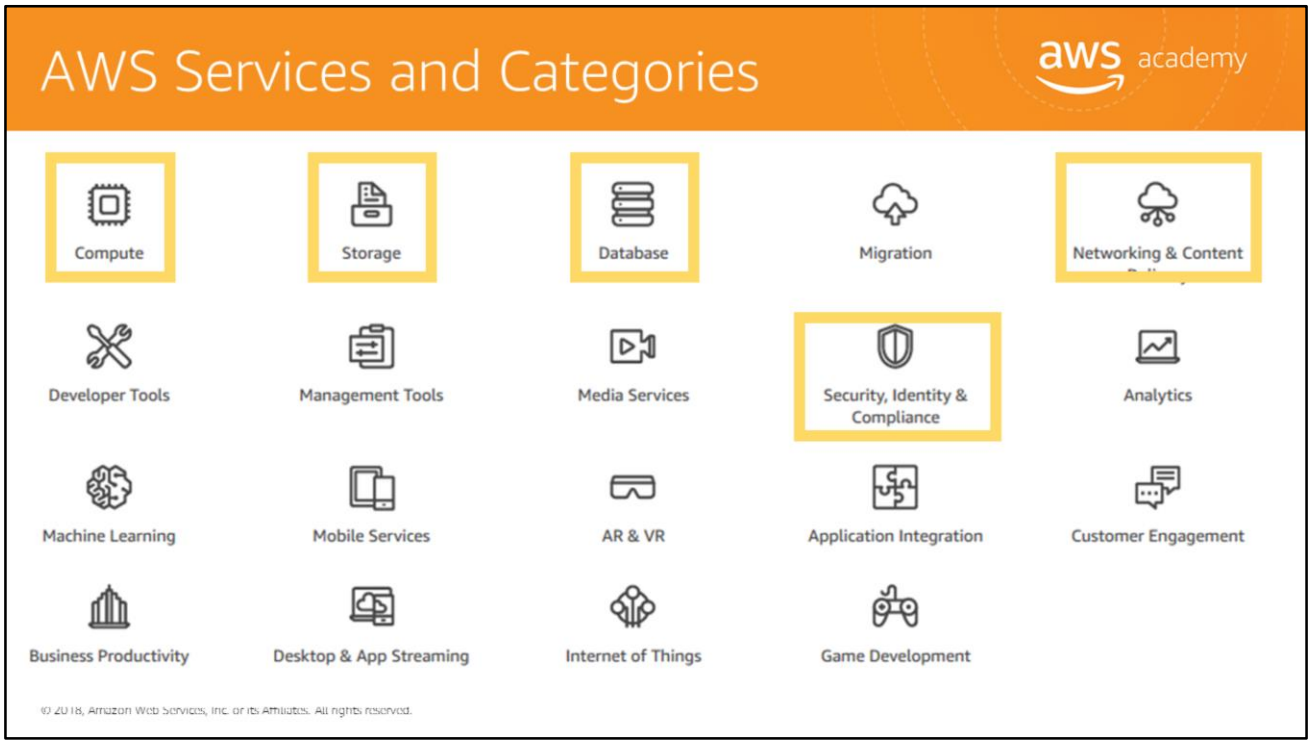


AWS offers a broad set of global cloud based products that can be used as building blocks for common cloud architectures. Let's look at how these cloud based products are organized.



As discussed previously, AWS's global infrastructure can be broken down into three elements: Regions, Availability Zones, and edge locations. This infrastructure provides the platform for a broad set of services, such as networking, storage, compute power, and databases, delivered as an on-demand utility that is available in seconds, with pay-as-you-go pricing.

Now, let's shift our focus to the core services and take a more in-depth look at what these are and what each offers you for building your cloud solution.

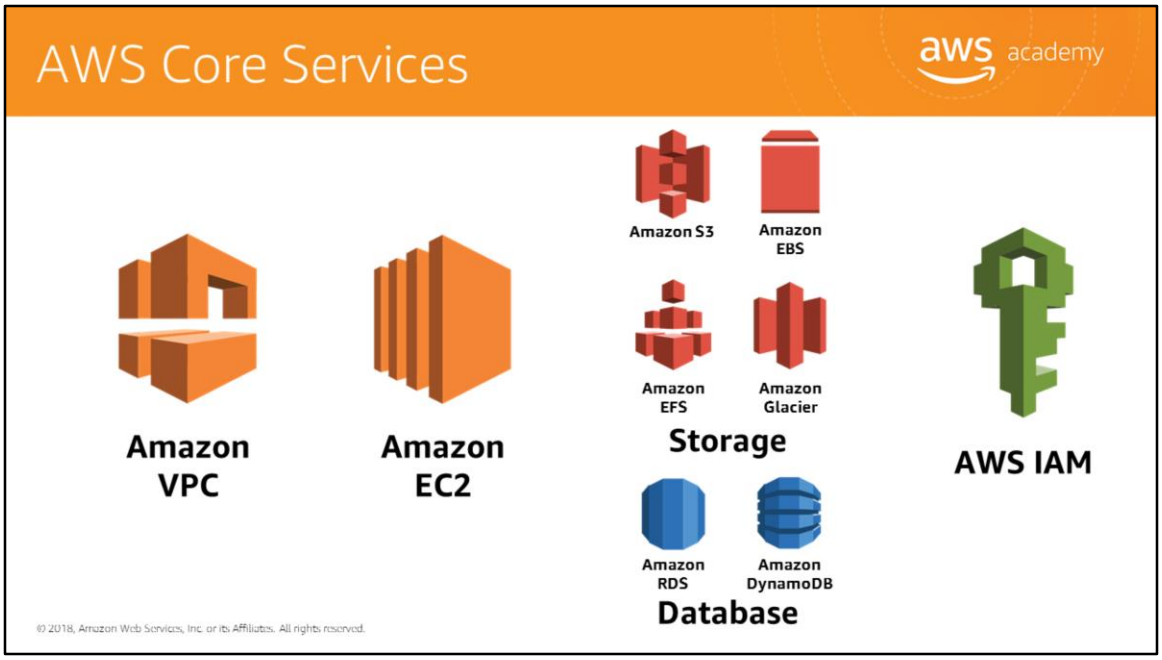


AWS offers a broad set of global cloud-based services that can be used as building blocks for common cloud architectures. Some of the categories we will discuss in this module include Compute, Storage, Database, Networking & Content Delivery and Security, Identity & Compliance.

If you go to the AWS front page, aws.amazon.com, and scroll down a little bit, you will find the section that allows you to explore the products. It places all of the products and services into different categories. For example, click on **Compute** and you will see Amazon EC2 is first on the list. There are also a lot of other products and services that appear in the compute category.

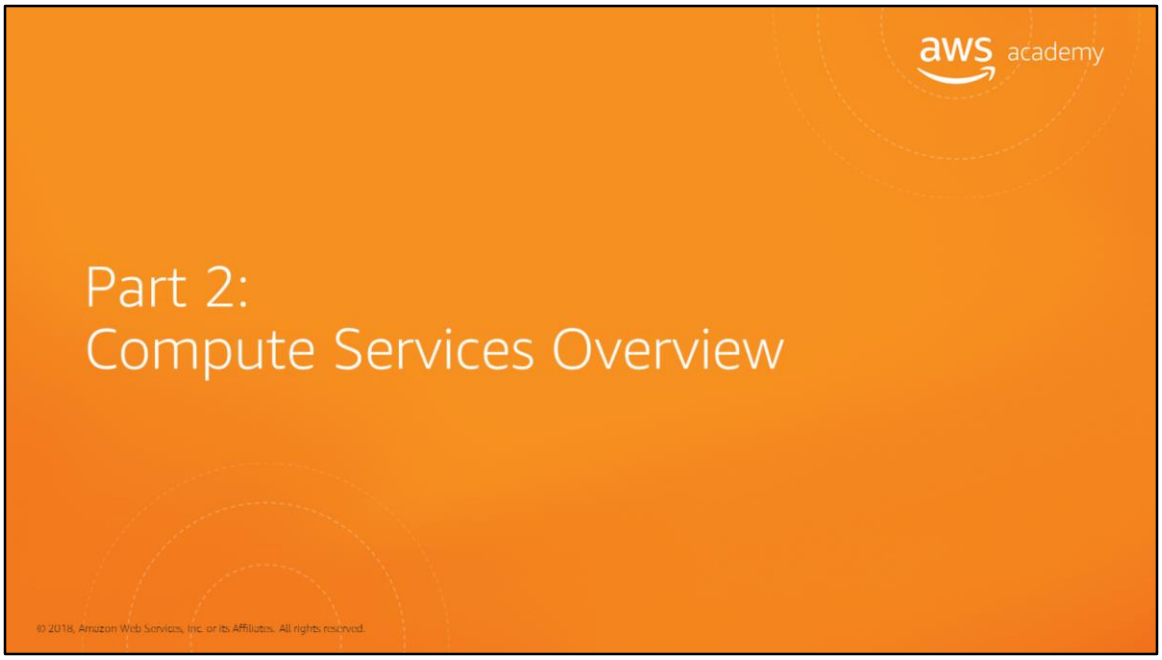
If you click **Amazon EC2**, it brings you to the Amazon EC2 main page http://aws.Amazon.com/EC2. It gives you a detailed description of the product and lists some of the benefits. Additionally, there are links for Product Details, Instance Types, Pricing, Getting Started, FAQs, and Resources. When you click on Product Details, there is more detailed information about Amazon EC2.

Explore the different service groups to understand the categories and services within them. Now that you know how to locate information about different services, let's narrow our discussion to the AWS Core Services.



As you probably noticed, Amazon Web Services has a lot of services available. In Part 2, we review Core AWS services including:

- AWS Virtual Private Cloud (VPCs)
- Security groups
- Compute services that use Amazon EC2
- Storage services including Amazon Simple Storage Service (Amazon S3), Amazon Elastic Block Store (Amazon EBS), Amazon Elastic File System (Amazon EFS), and Amazon Glacier
- AWS Identity and Access Management (IAM)



Running servers on premises is an expensive undertaking. Hardware needs to be procured, often based on project plans rather than the reality of usage. Data centers are expensive to build, staff, and maintain. You need to provision resources for the worst case. Servers need to be able to handle traffic spikes and events. Once built out, you often have capacity lying idle.

AWS offers flexibility and cost effectiveness. With AWS, you can scale your compute needs to your workload. Scalability is built into our compute services so that as demand increases, you can easily scale up. When demand drops, say at night or on weekends, you can scale down to save money and resources. You don't need to pay for what you're not using.

Let’s start with an overview of Compute Services to understand how they can address these issues.

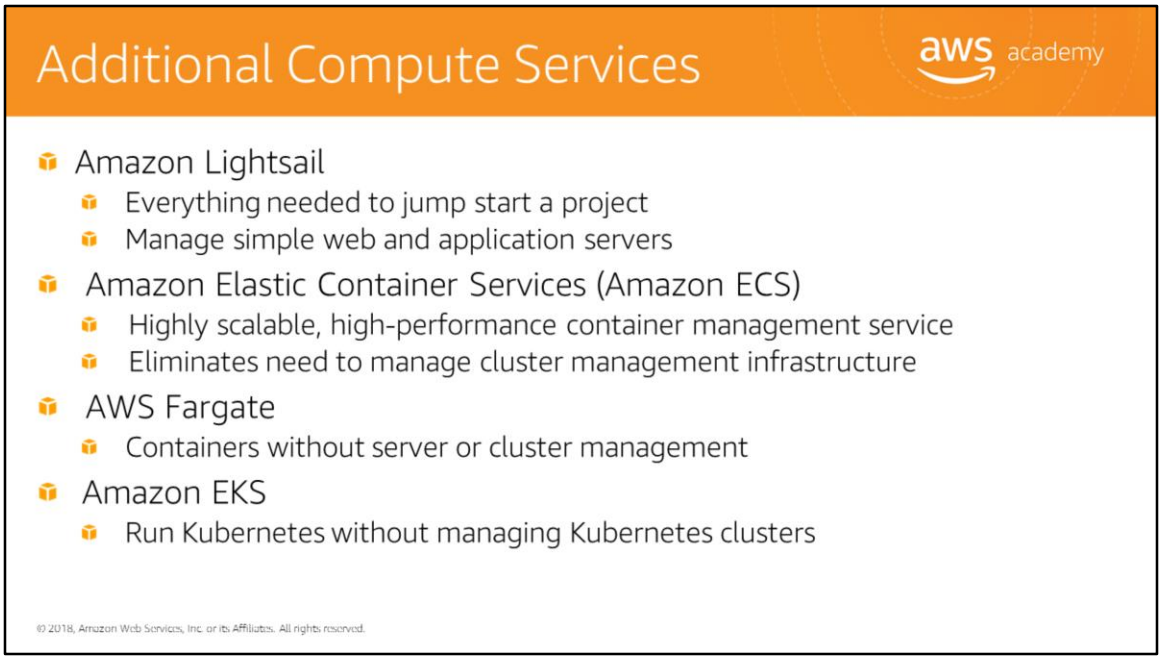
Amazon EC2 is a web service that provides resizable compute capacity in the cloud. It allows organizations to obtain and configure virtual compute capacity in the cloud. You can select from a variety of operating systems and resource configurations (memory, CPU, storage, etc.) that are optimal for the application profile of each workload. You can cost-effectively scale resources up and down to meet your needs.

AWS Lambda is a zero-administration compute platform. AWS Lambda lets you run code without provisioning or managing servers. You pay only for the compute time you consume. There's no charge when your code's not running. With Lambda, you can run code for virtually any type of application or backend service: mobile, internet of Things (IoT), streaming service — all with zero administration.

Auto Scaling allows organizations to scale Amazon EC2 capacity up or down automatically according to conditions defined for a particular workload. It be used to help maintain application availability and ensure that the desired number of Amazon EC2 instances are running, but it also allows resources to scale in and out to match workload demand.

Elastic Load Balancer automatically adjusts to incoming traffic and rapid changes in network traffic patterns by distributing the it across multiple Amazon EC2 instances in the cloud without manual intervention. This enables you to achieve higher levels of fault tolerance with your applications.

AWS Elastic Beanstalk is a Platform as a Service that facilitates quick deployment of your applications by providing all the applications services that you need for your application.



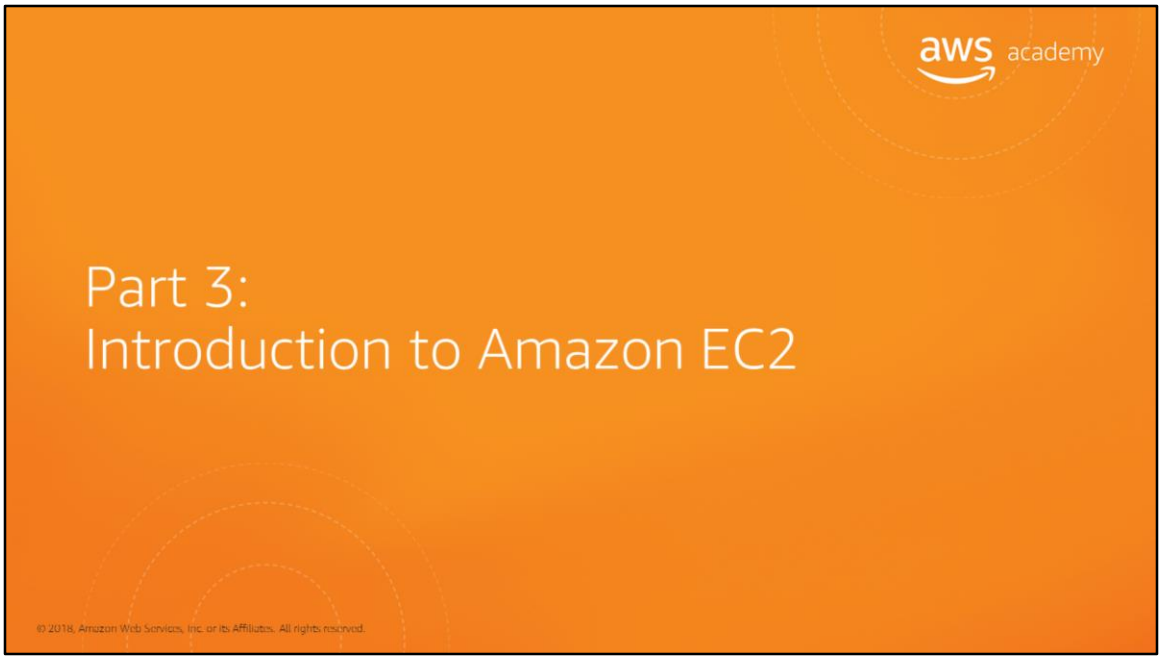
Amazon Lightsail includes everything you to jump-start your project—a virtual machine, SSD-based storage, data transfer, DNS management, and a static IP address—for a low, predictable price.

Amazon Elastic Container Service (Amazon ECS) is a highly scalable, high-performance container management service that supports Docker containers and allows you to easily run applications on a managed cluster of Amazon EC2 instances.

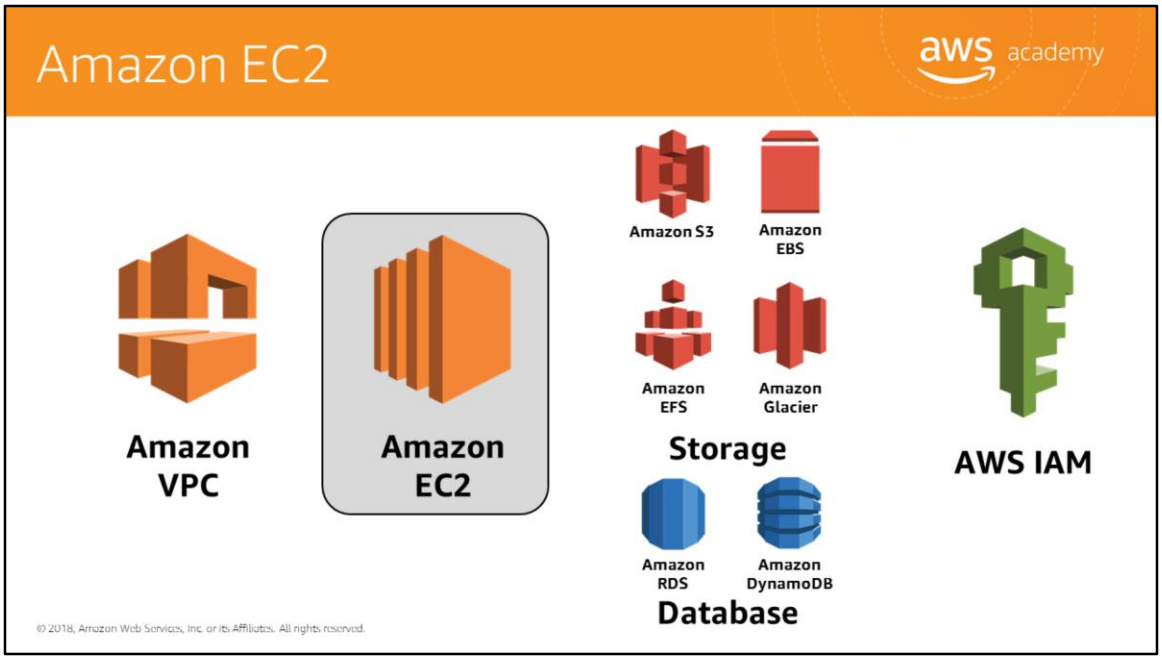
AWS Fargate is a technology for Amazon ECS and EKS that allows users to run containers without having to manage servers or clusters. It removes the need to interact with or think about servers/clusters.

Amazon Elastic Container Service for Kubernetes (Amazon EKS) is a managed service that makes it simple to run Kubernetes on AWS without needing to install/operate your own Kubernetes clusters.

For additional information about AWS compute services see https://aws.amazon.com/products/compute/.

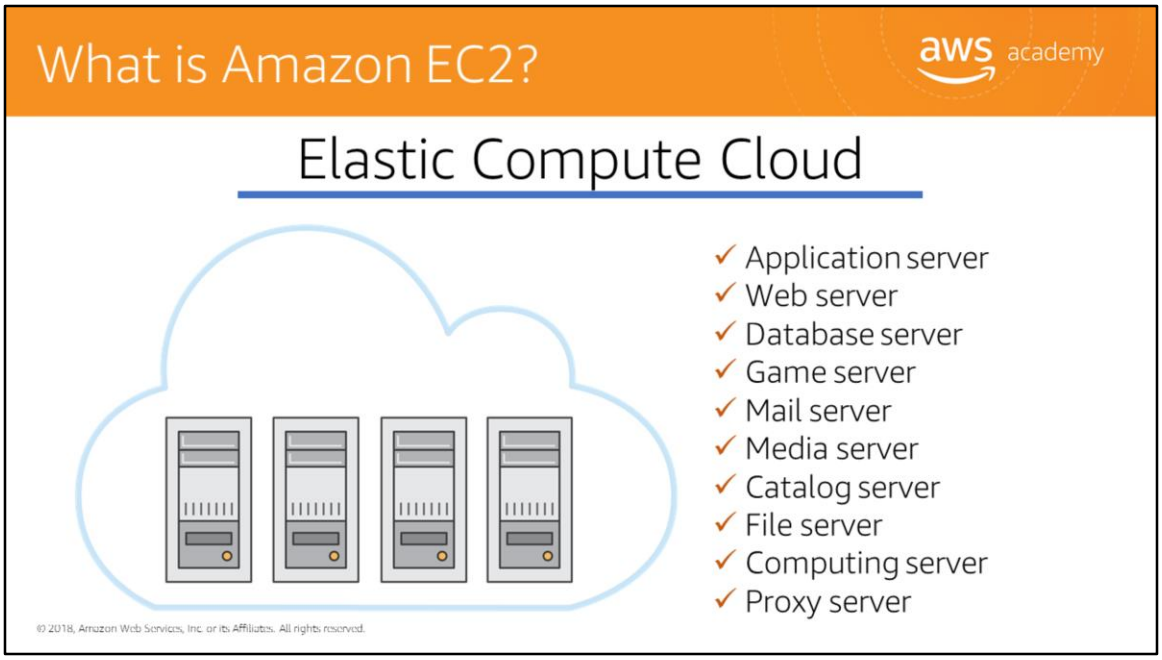


.Amazon EC2 is one of the core AWS services; it provides compute resources. Let’s start by looking at some Amazon EC2 basic facts.



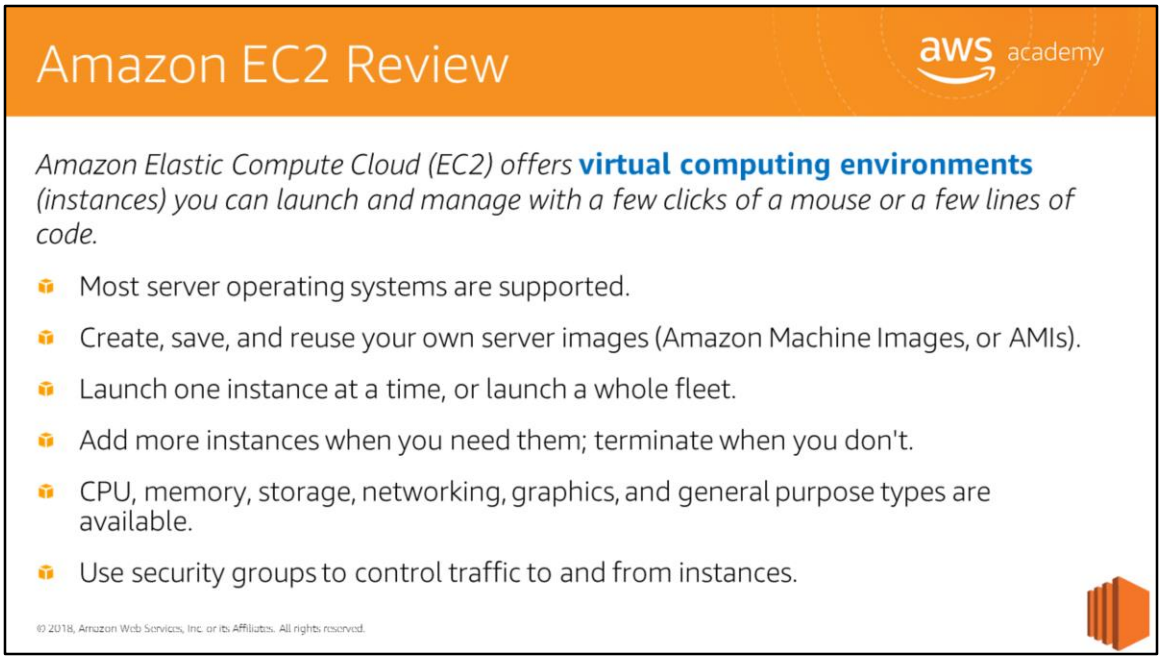
Compute refers to the amount of computational power required to fulfill your workload. If your workload is small, such as a website that receives few visitors, then your compute needs (or workload) is very small. A larger workload, such as screening a bacteria against thousands of different combinations of antibiotics for sensitivity, may require a great deal of compute.

Let's look at how Amazon EC2 handles these workloads.



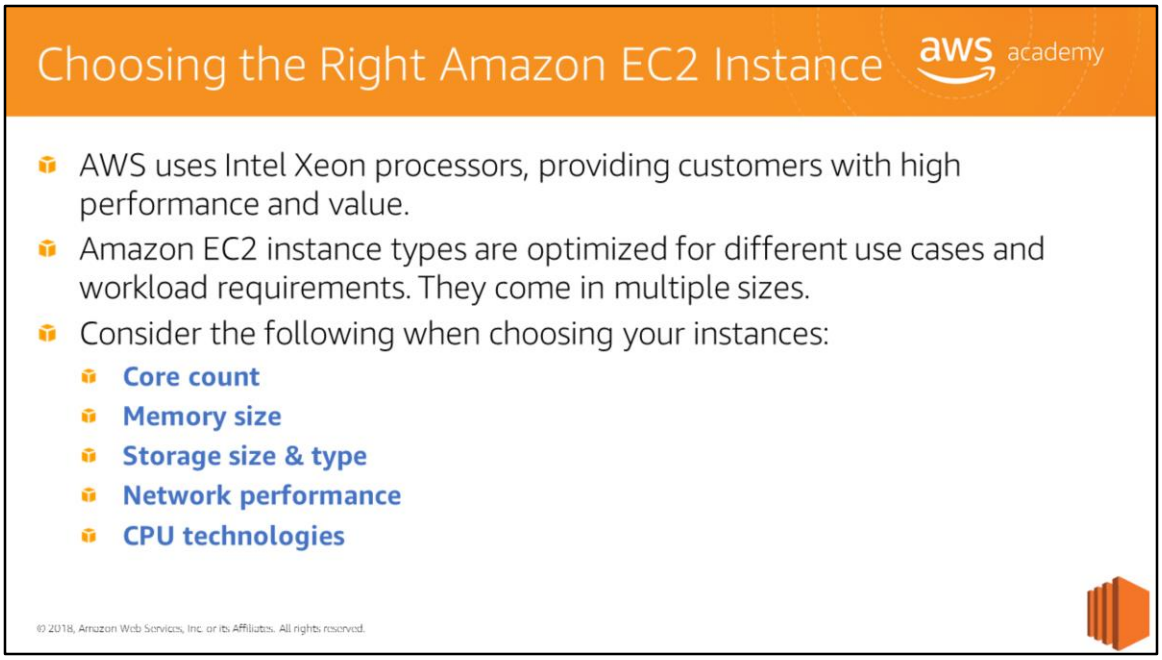
First, what is Amazon EC2? *EC2* stands for *Elastic Compute Cloud.*

- ***Elastic*** refers to the fact that if properly configured, you can increase or decrease the amount of servers required by an application automatically, according to the current demands on that application.
- ***Compute*** refers to the compute, or *server*, resources that are being presented.
- ***Cloud*** refers to the fact that these are cloud-hosted compute resources.



Amazon EC2 is essentially a computer in the cloud. Virtually anything you can do with a server, you can do with an Amazon EC2 instance. When combined with the other services from AWS, with which Amazon EC2 is optimized to work, it enables you to do even more.

- Most server operating systems are supported: Windows 2003, 2008, and 2012, Red Hat, SUSE, Ubuntu, and Amazon Linux.
- You can create images of your servers at any time with a few clicks or a simple API call. These images are referred to as *AMIs* (Amazon Machine Images) and can be reused to launch instances in the future.
- You can launch one instance or an entire fleet of instances with a few clicks or a simple API call.
- Amazon EC2 instances in Amazon VPC now offer native support for the IPv6 protocol. IPv6 can be enabled for existing and new VPCs through the AWS Management Console, Software Development Kit (SDK), and Command Line Interface (CLI).
- Scalable: Add more instances when you need them; terminate them when you don’t.
- Optimizable: Choose from instance types optimized for compute, memory, storage, accelerated computing and general purpose. Each instance type has a variety of sizes. To learn more about instance types see https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/instance-types.html.

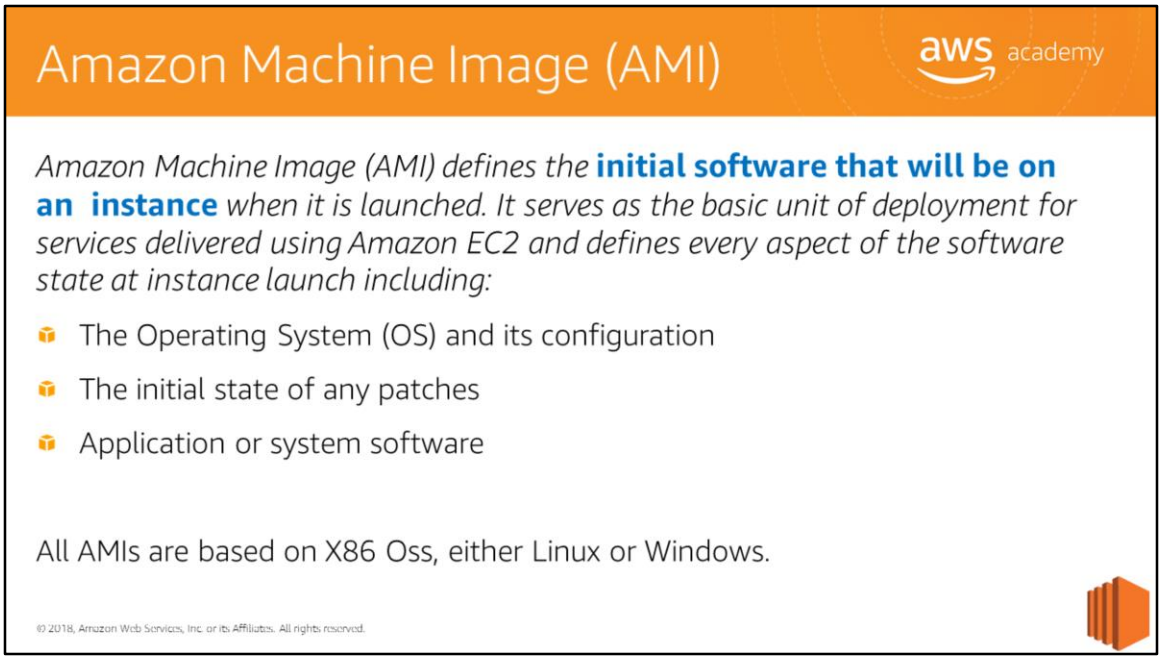


AWS has a wide variety of Amazon EC2 compute instances and choosing the right instance type matters.

There are two key dimensions of new instances that are controlled by the instance type and Amazon Machine Image (AMI) for images launched on AWS:
1. Amount of virtual hardware dedicated to the instance
2. Software loaded on the instance

Each instance type or family is optimized for different workloads or use cases. Within each type or family, there are multiple sizes: Large, XLarge, 2XLarge, etc. Amazon EC2 provides different instance types to enable you to choose the CPU, memory, storage, and networking capacity that you need to run your applications. When you choose your instance type, consider the several different attributes of each family; such as number of cores, amount of memory, amount and type of storage, networks performance, and Intel processor technologies.

Larger instances are better for workloads that scale.

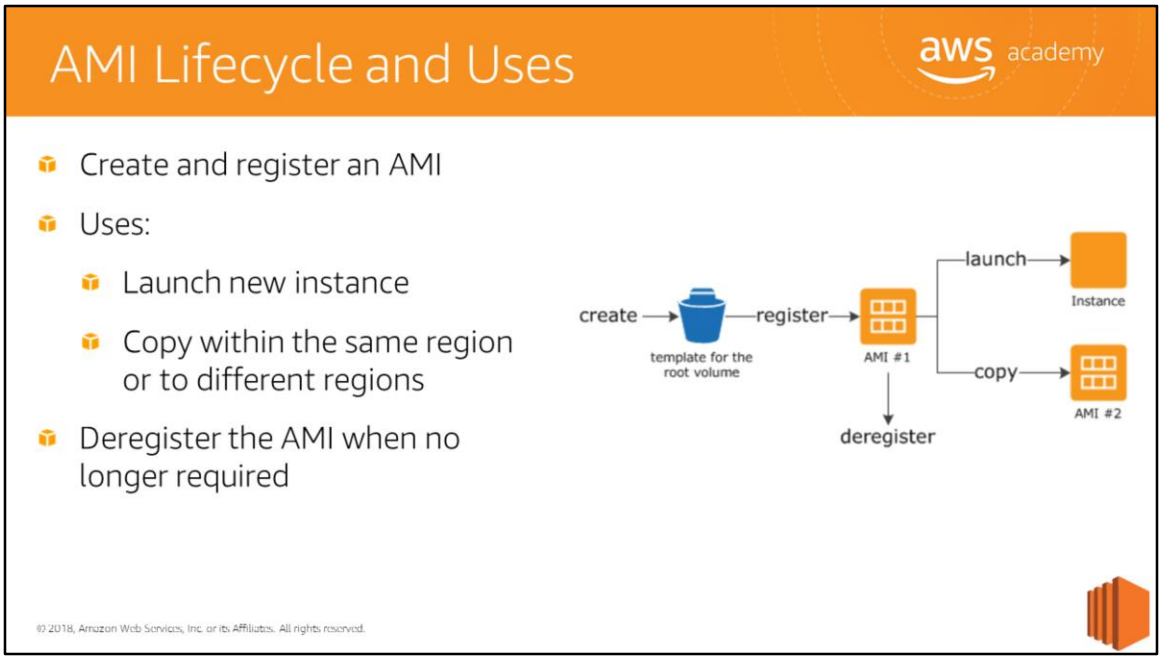


An ***AMI*** is a template that contains a software configuration (for example, an operating system, an application server, and applications). From an AMI, you launch an *instance*, which is a copy of the AMI running as a virtual server in the cloud.

You must specify a source AMI when you launch an Amazon EC2 instance. You can launch multiple instances from a single AMI when you need multiple instances with the same configuration. You can use different AMIs to launch instances when you need instances with different configurations.

An AMI includes the following:

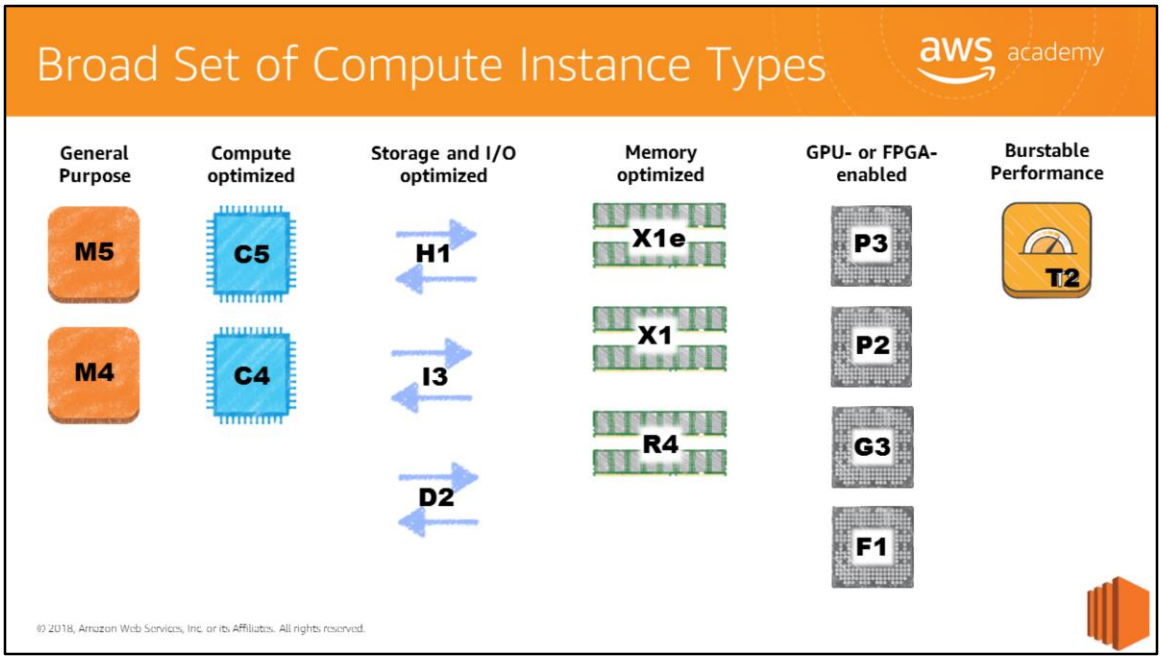
- A template for the root volume for the instance (for example, an operating system, an application server, and applications)
- Launch permissions that control which AWS accounts can use the AMI to launch instances
- A block device mapping that specifies the volumes to attach to the instance when it's launched



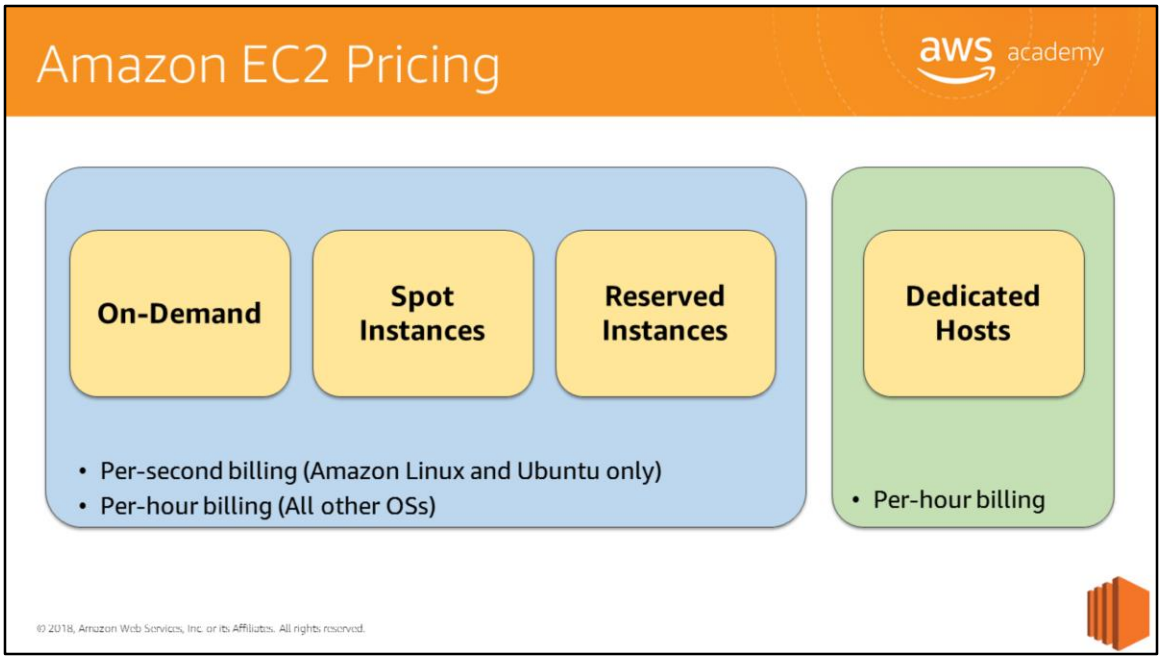
After an AMI is created, it is registered. Once this is done, the AMI can be used to launch new instances. You can also use an AMI you don't own to launch instances if the AMI owner grants you launch permissions. You can copy an AMI within the same region or to different regions.

When you no longer need an AMI, you can deregister it.

You can learn more about AMIs at https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/AMIs.html.



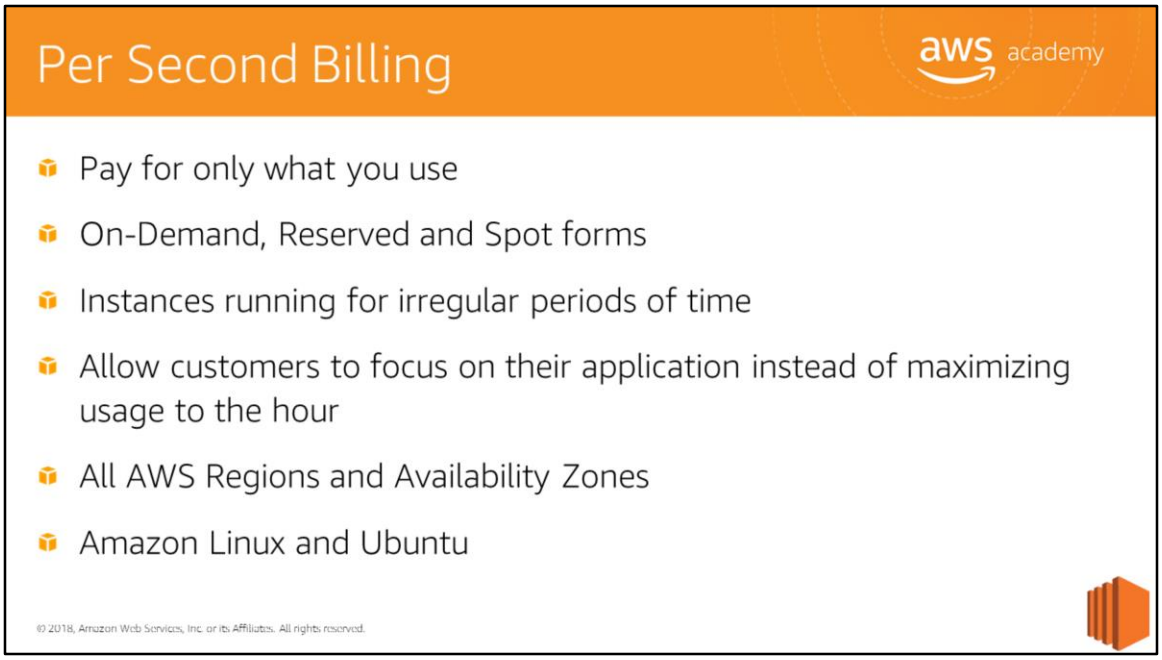
Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instance types comprise varying combinations of CPU, memory, storage, and networking capacity and give you the flexibility to choose the appropriate mix of resources for your applications. Each instance type includes one or more instance sizes, which allows you to scale your resources to the requirements of your target workload.



There are four ways to pay for Amazon EC2 instances: On-Demand, Reserved Instances, Spot Instances, and Dedicated Hosts.

Amazon EC2 usage of Linux- and Ubuntu-based instances that are launched in On-Demand, Spot, and Reserved are billed on one-second increments, with a minimum of 60 seconds. All other types of OS are billed by the hour. The minimum unite of time that will be charged is a minute, but after your first minute of time, we can account for seconds. However if you start then stop an instance in 10 seconds, you will be charged the 60 seconds not 10.

Dedicated Hosts provide you with Amazon EC2 instance capacity on physical servers dedicated for your use.



With per-second billing, you pay for only what you use. It takes cost of unused minutes and seconds in an hour off of the bill, so you can focus on improving your applications instead of maximizing usage to the hour. This is of particular value if you manage instances running for irregular periods of time, such as dev/testing, data processing, analytics, batch processing and gaming applications.
Amazon EC2 usage are billed on one second increments, with a minimum of 60 seconds. Similarly, provisioned storage for EBS volumes will be billed per-second increments, with a 60-second minimum. Per-second billing is available for instances launched in:

- On-Demand, Reserved and Spot forms
- All AWS Regions and Availability Zones
- Amazon Linux and Ubuntu

Amazon EC2 Pricing: Costs

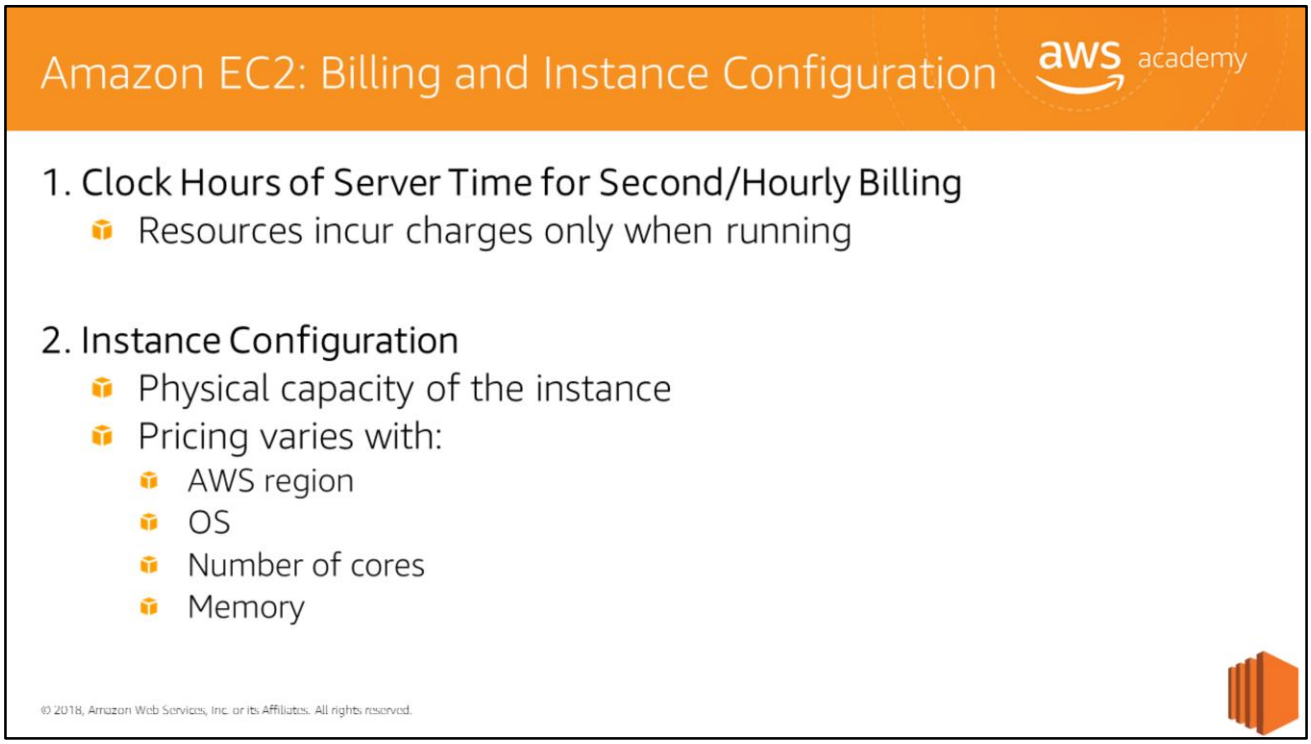
On-Demand Instances	Spot Instances	Reserved Instances	Dedicated Hosts
• Pay for what you use • Per-second billing	• Spot price based on supply and demand • Per-second billing	• Pay low or no upfront fee; overall cost is lower • Per-second billing	Pay the On-Demand rate for every hour the host is active in the account

Amazon EC2 Pricing Options: Benefits

On-Demand Instances	Spot Instances	Reserved Instances	Dedicated Hosts
Low cost and flexibility	Large scale, dynamic workload	Predictability ensures compute capacity is available when needed	• Save money on licensing costs • Help meet compliance and regulatory requirements

Amazon EC2 Pricing Options: Use Cases

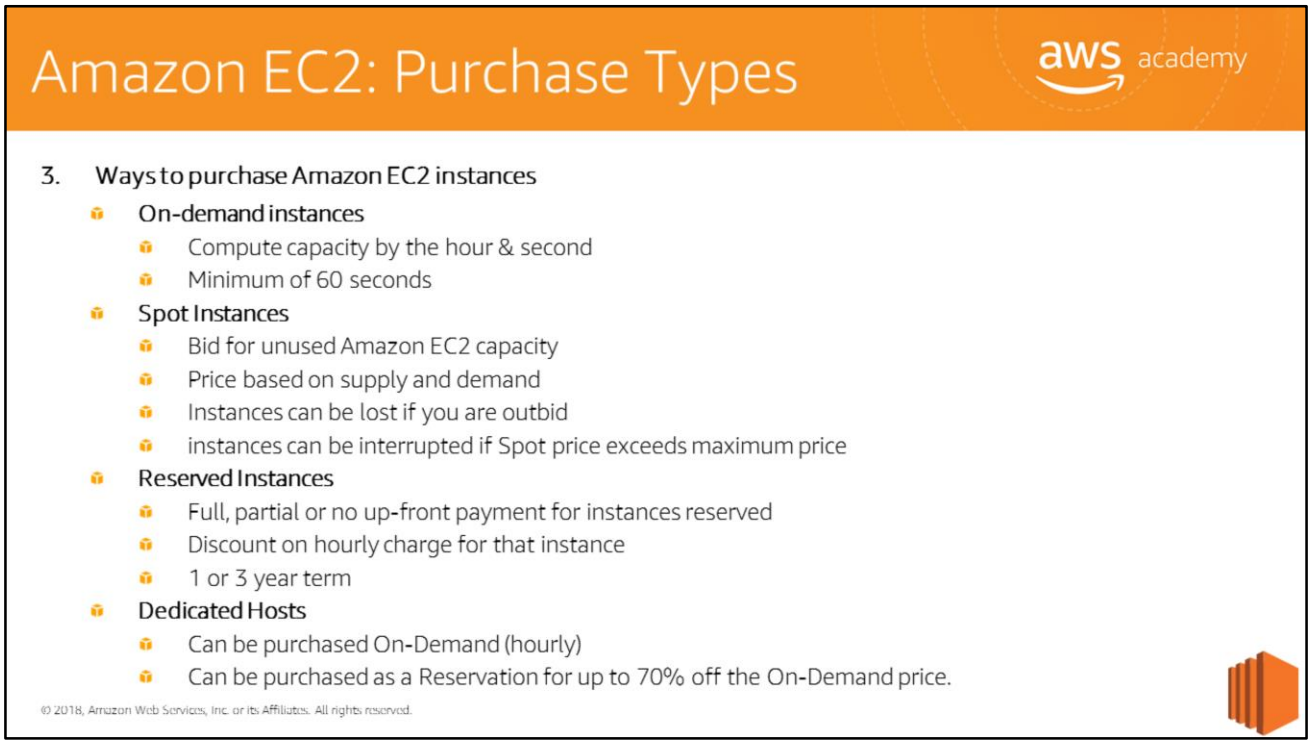
On-Demand Instances	Spot Instances	Reserved Instances	Dedicated Hosts
• Short-term, spiky, or unpredictable workloads • Application development or testing	• Applications with flexible start and end times • Applications only feasible at very low compute prices • Users with urgent computing needs for large amounts of additional capacity	• Steady state or predictable usage workloads • Applications that require reserved capacity, including disaster recovery • Users able to make upfront payments to reduce total computing costs even further	• Bring your own license (BYOL) • Compliance and regulatory restrictions • Usage and licensing tracking • Control instance placement



When you begin to estimate the cost of using Amazon EC2, you need to consider nine factors:

1. **Clock Hours of Server Time** – Resources incur charges when they are running. For example, charges are incurred from the time Amazon EC2 instances are launched until they are terminated, or from the time Elastic IPs are allocated until the time they are de-allocated.

2. **Instance Configuration** – Consider the physical capacity of the Amazon EC2 instance you choose. Instance pricing varies with the AWS region, OS, number of cores, and memory.



3. There are several ways to purchase Amazon EC2 instances:
    - With On-Demand Instances, you pay for compute capacity by the hour or second with no required minimum commitment. ***Note***: per second billing only available for Linux- and Ubuntu-based instances.
    - Reserved Instances give you the option to make a low one-time payment – or no payment at all – for each instance you want to reserve and in turn receive a significant discount on the hourly usage charge for that instance.
    - With Spot Instances, you can bid for unused Amazon EC2 capacity.

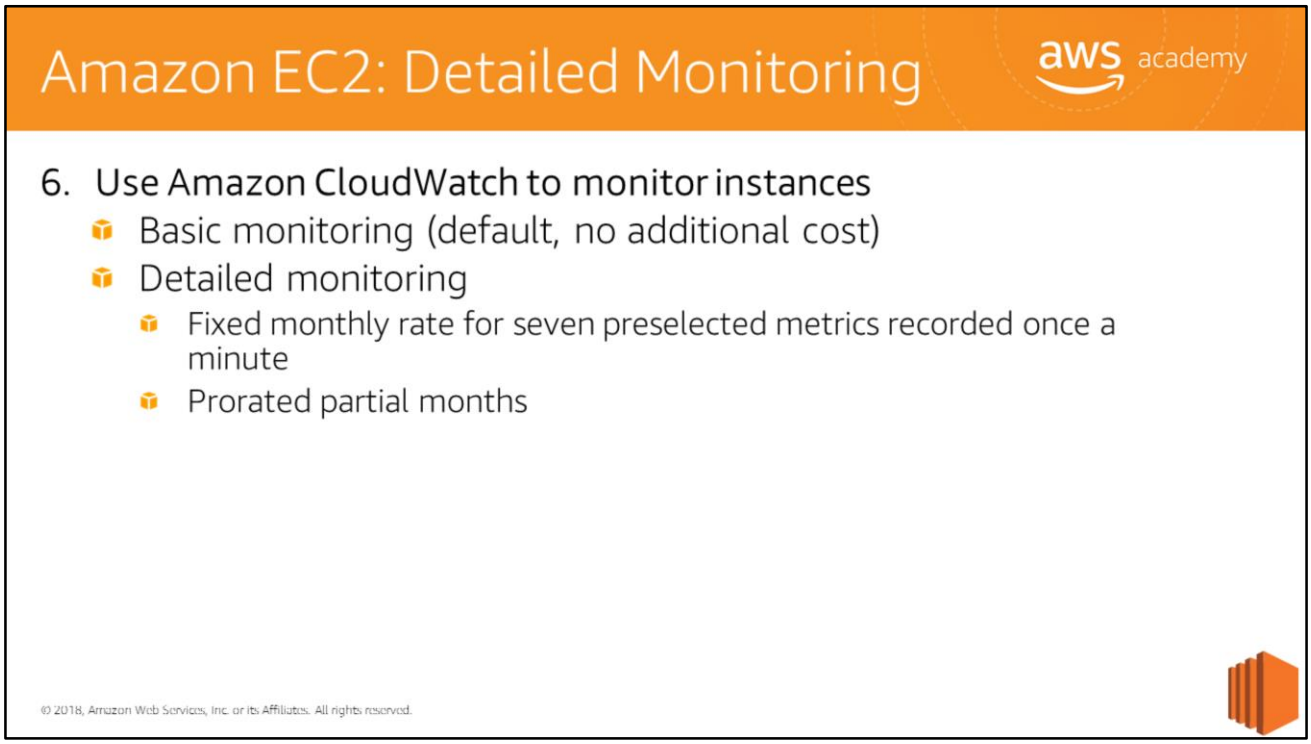
Amazon EC2: Number of Instances and Load Balancing

4. Number of Instances
   - Provision multiple instances to handle peak loads and shut them down when they are no longer needed.. Pay for only the capacity that you actually use.

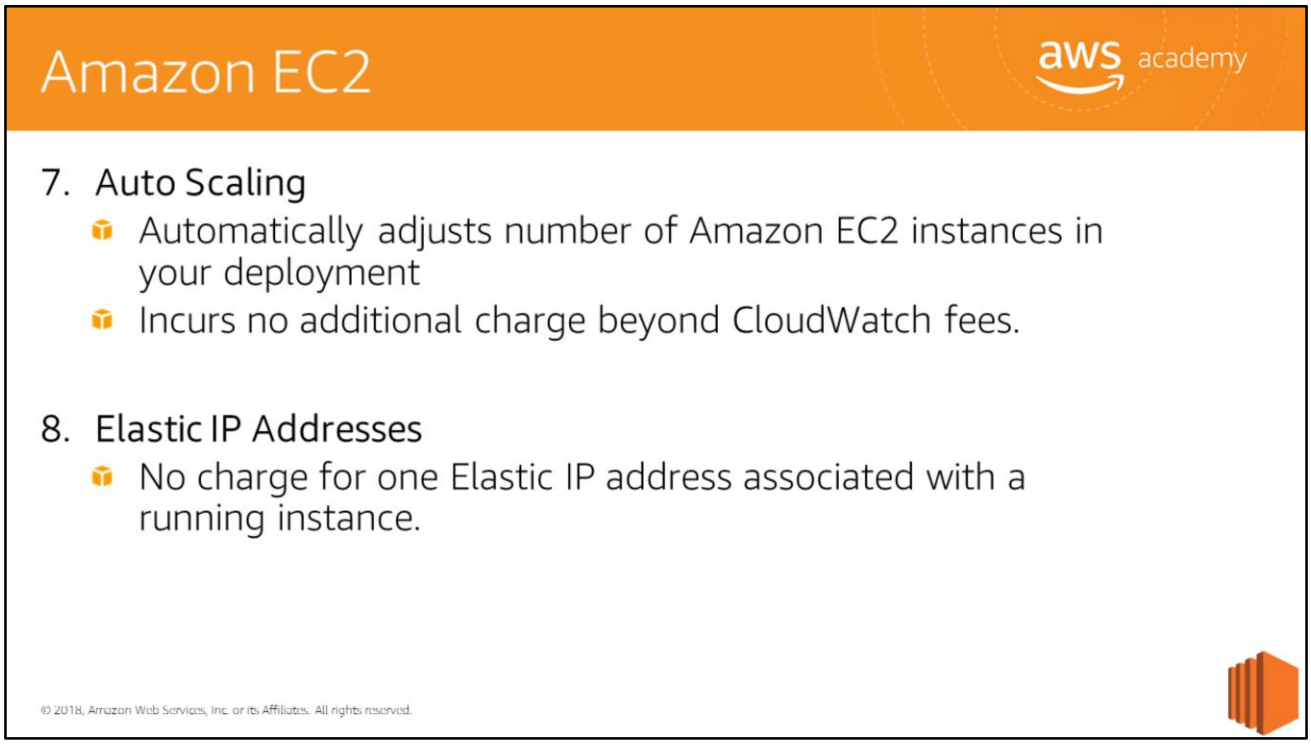
5. **Load Balancing -** Uses Elastic Load Balancing to distribute traffic among Amazon EC2 instances
   - Calculates monthly cost based on:
     - Hours load balancer runs
     - Data load balancer processes

4. **Number of Instances** – You can provision multiple instances of your Amazon EC2 and Amazon EBS resources to handle peak loads.

5. **Load Balancing** – An Elastic Load Balancer can be used to distribute traffic among Amazon EC2 instances. The number of hours the Elastic Load Balancer runs and the amount of data it processes contribute to the monthly cost.

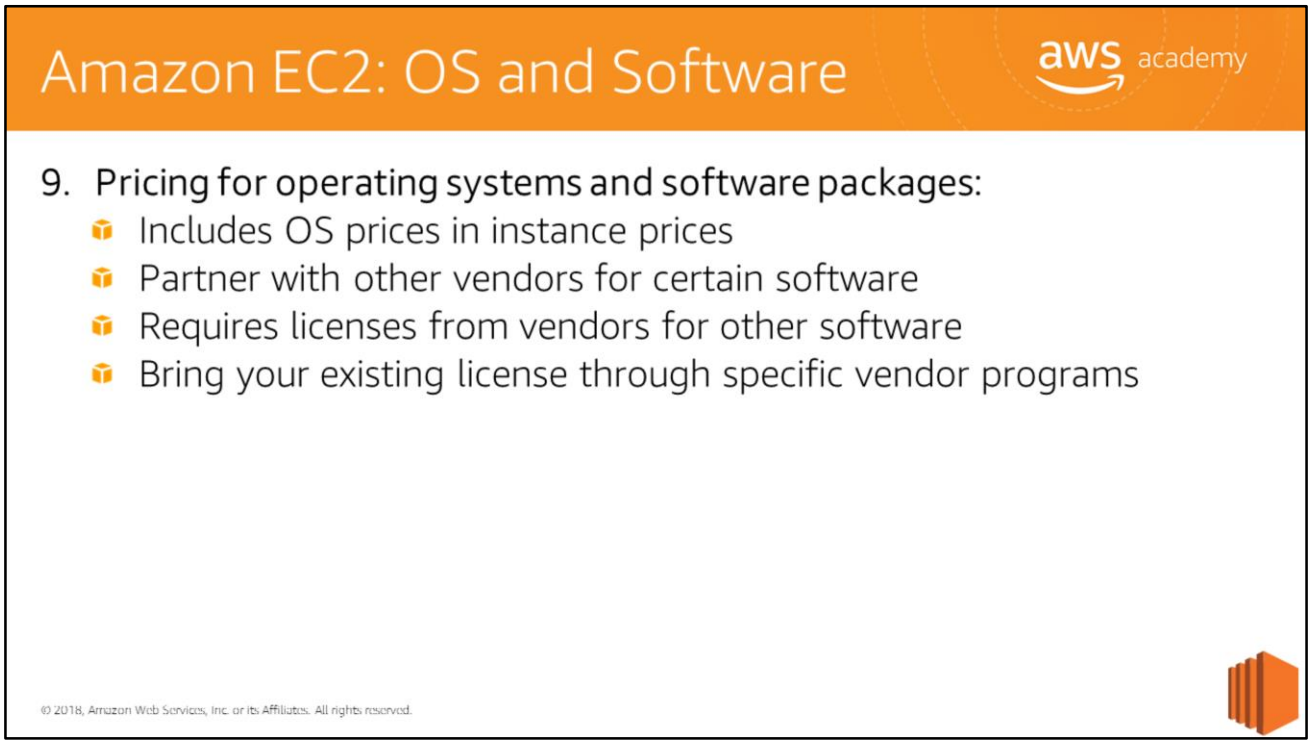


6. **Detailed Monitoring** – You can use Amazon CloudWatch to monitor your Amazon EC2 instances. By default, basic monitoring is enabled (and available at no additional cost); however, for a fixed monthly rate, you can opt for detailed monitoring, which includes seven preselected metrics recorded once a minute. Partial months are charged on an hourly pro rata basis, at a per instance-hour rate.

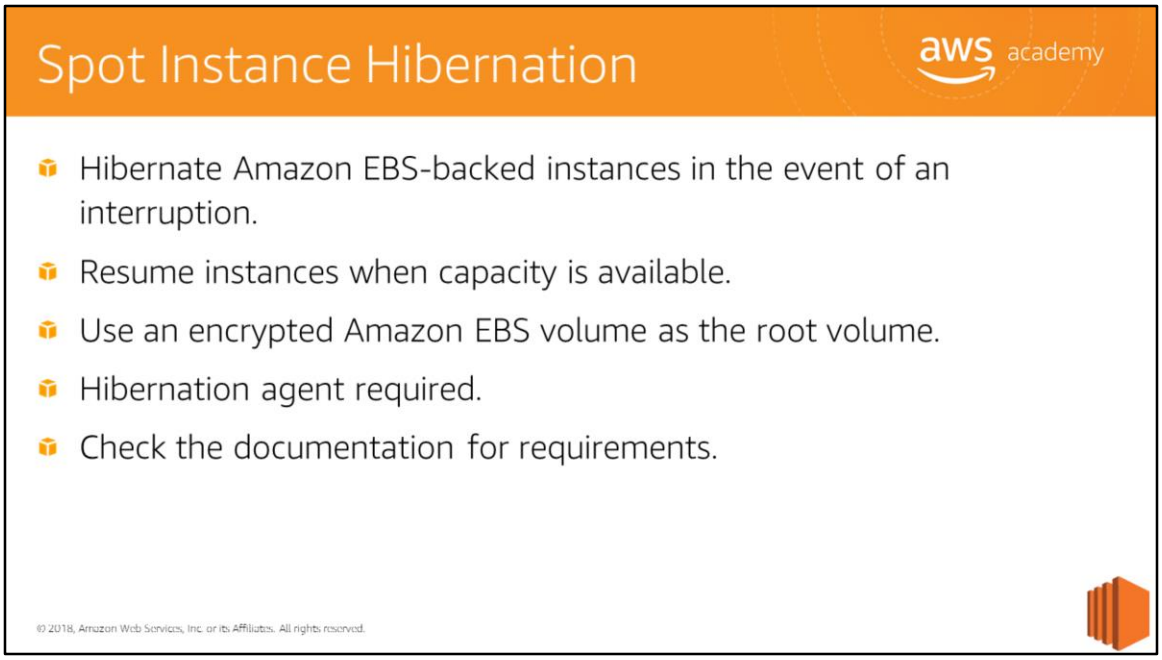


7. **Auto Scaling –** Auto Scaling automatically adjusts the number of Amazon EC2 instances in your deployment according to conditions you define. This service is available at no additional charge beyond Amazon CloudWatch fees.

8. **Elastic IP Addresses** – You can have one Elastic IP (EIP) address associated with a running instance at no charge.



9. **Operating Systems and Software Packages** – Operating System prices are included in the instance prices. AWS has made it easy for you and has partnered with Microsoft, IBM, and several other vendors to simplify running certain commercial software packages running on your Amazon EC2 instances (for example, Microsoft SQL Server on Windows, IBM Software). For commercial software packages that AWS does not provide, such as nonstandard operating systems, Oracle Applications, Windows Server applications such as Microsoft SharePoint and Microsoft Exchange, you need to obtain a license from the vendors. You can also bring your existing license to the cloud through specific vendor programs such as Microsoft License Mobility through Software Assurance Program.

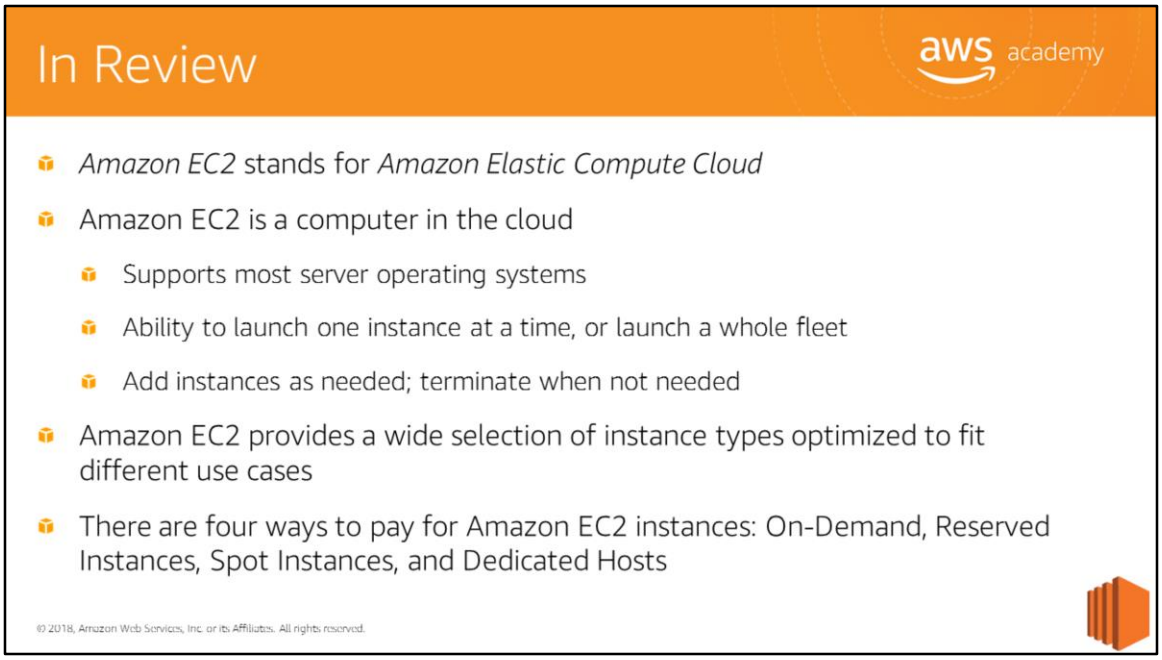


Amazon EC2 Spot can now hibernate Amazon EBS-backed instances in the event of an interruption. Spot can fulfill your request by resuming instances from a hibernated state when capacity is available. *Hibernating* is just like closing and opening your laptop lid, with your application starting up right where it left off.

After a Spot Instance is hibernated by the Spot service, it can only be resumed by the Spot service. The Spot service resumes the instance when capacity becomes available with a Spot price that is less than your specified maximum price.

We strongly recommend that you use an encrypted Amazon EBS volume as the root volume, because instance memory is stored on the root volume during hibernation. This ensures that the contents of memory (RAM) are encrypted when the data is at rest on the volume and when data is moving between the instance and volume. If your AMI does not have an encrypted root volume, you can copy it to a new AMI and request encryption.

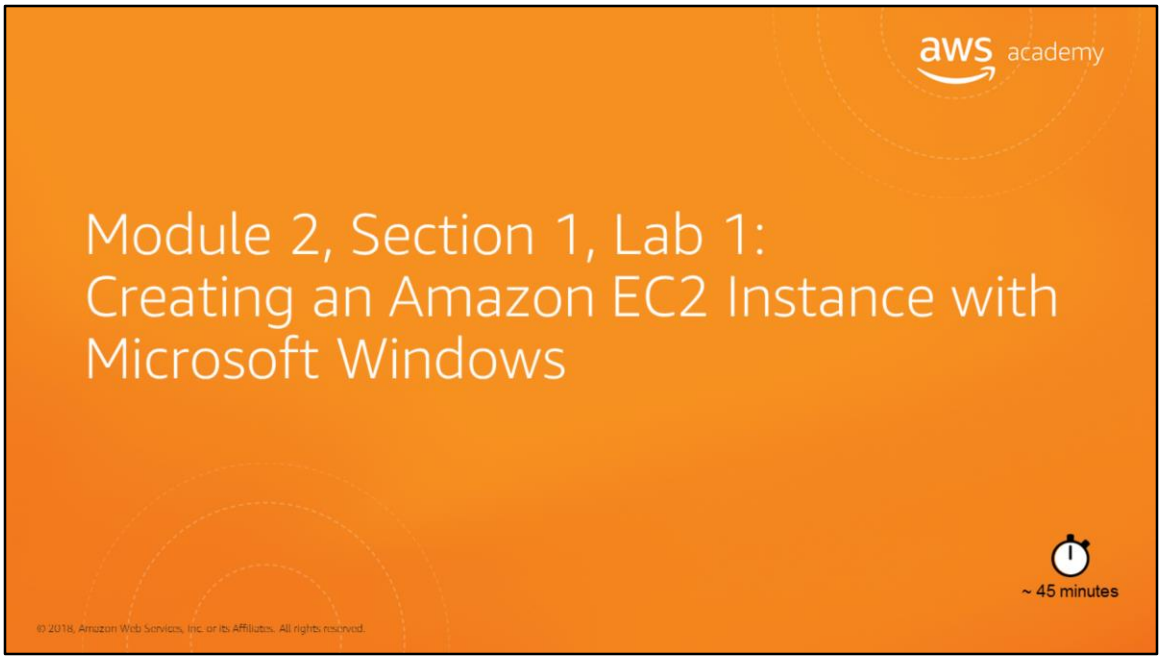
For information on the requirements (including the agent), see http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/spot-interruptions.html#interruption-behavior



In review, Amazon EC2 is essentially a computer in the cloud. Virtually anything you can do with a server, you can do with an Amazon EC2 instance. Amazon EC2 supports most operating systems. Amazon EC2 is scalable -- you can launch a single instance or a fleet of instances and terminate them when they are not needed. Instance types are optimized for CPU, memory, storage, networking capacity, graphics, and general purpose to enable you to select the best type for your needs. Just like you can optimize your instance type, Amazon EC2 offers four ways to pay for instances so you can minimize costs.

Please review the Amazon EC2 demonstration: M2_S2_EC2 Set Up Demo Project v2.0.mp4.

This video demonstration can be found in the learning management system.



Amazon EC2 is a web service that provides secure, resizable compute capacity in the cloud. It is designed to make web-scale cloud computing easier for developers.

After completing this lab, you will be able to:

- Launch an Amazon EC2 server instance from an AMI.
- Create a security group to permit access to server resources.
- Log in to an instance.
- Configure an IIS web server.

Duration: ~45 minutes

Lab 1: Tasks

Launch a **Microsoft Windows Server Instance** (on Amazon EC2).

Create a **VPC Security Group.**

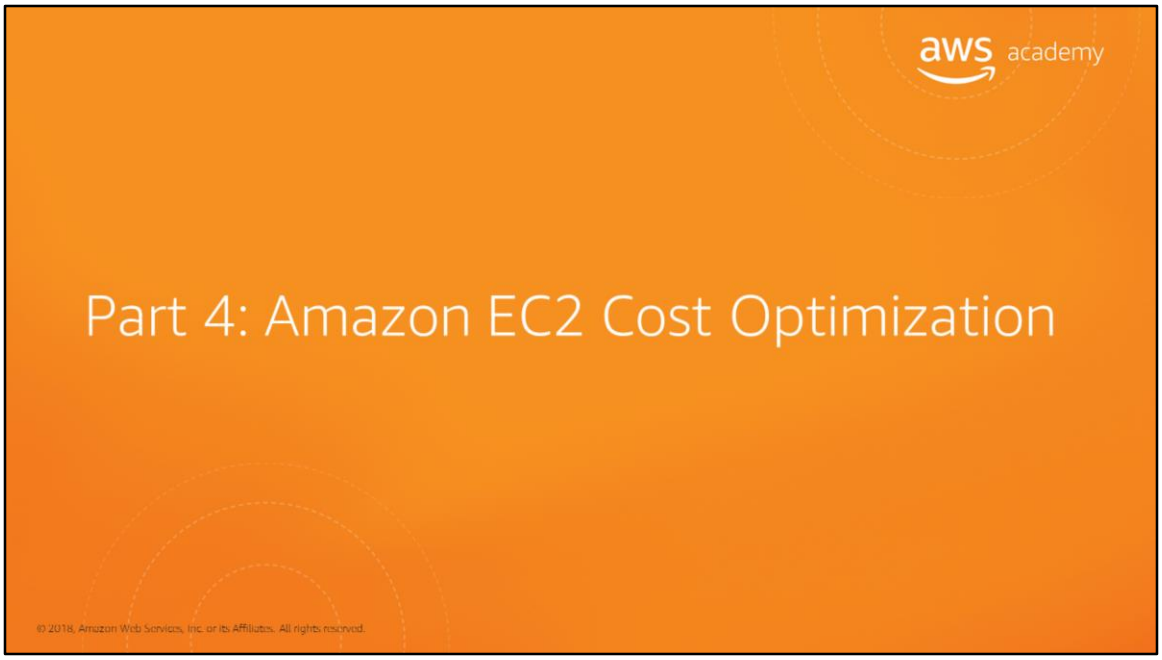
Launch an **Internet Information Services (IIS)** Web Server **(on Amazon EC2).**

Lab 1: Final Product

In this lab, you:

- Launched an Amazon EC2 instance from an AMI.
- Created a security group to permit access to server resources.
- Logged in to the new instance and created an IIS web server.

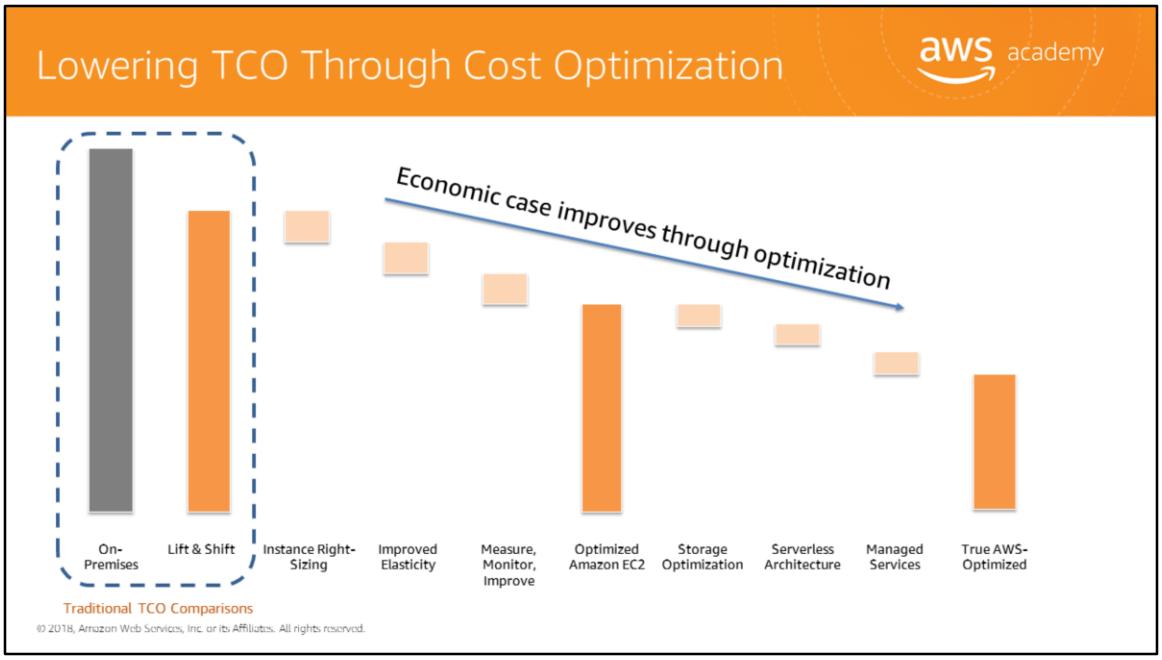
~ 45 minutes



One of the most common reasons to move into the cloud is to **reduce costs**. In Part 4: Cost Optimization, using Amazon EC2 as our example, we will review important cost optimization elements.

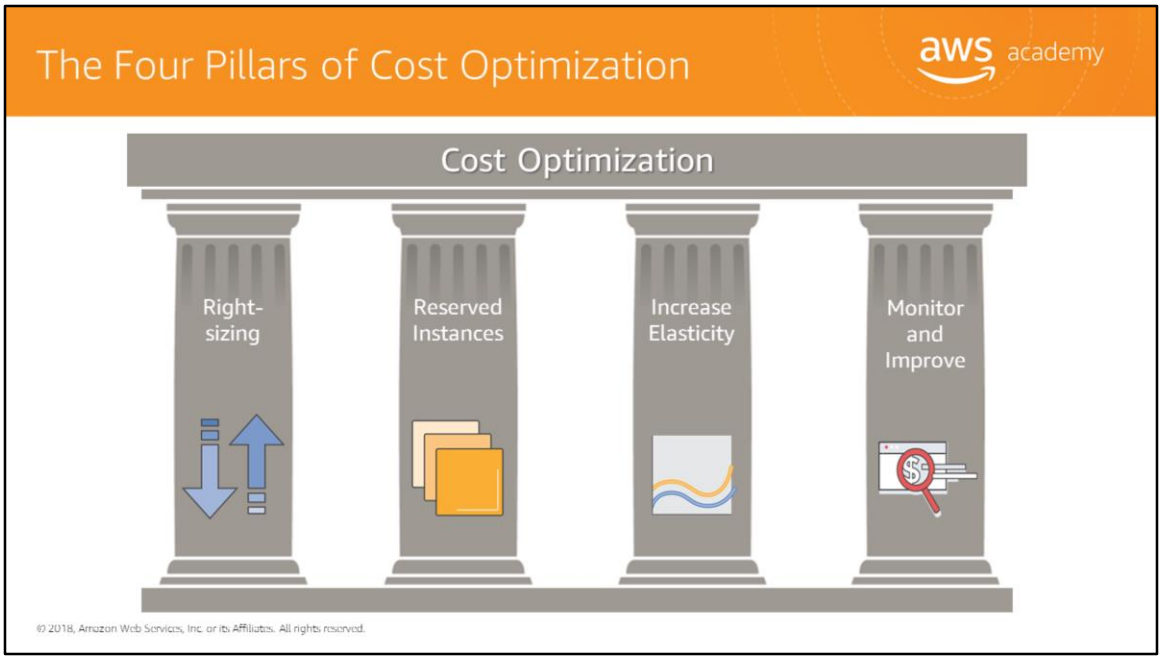
To reduce costs, it's very important to be able to **optimize spend** and **pay only for what you need** and **when you need it**. When you optimize costs, you can help your organization get the most out of your investments helping meet demand and capacity while using the most economically effective options.

Cost optimization using cloud has brought about new business models enabling organizations to be lean with what they use and reduce their spend dramatically.



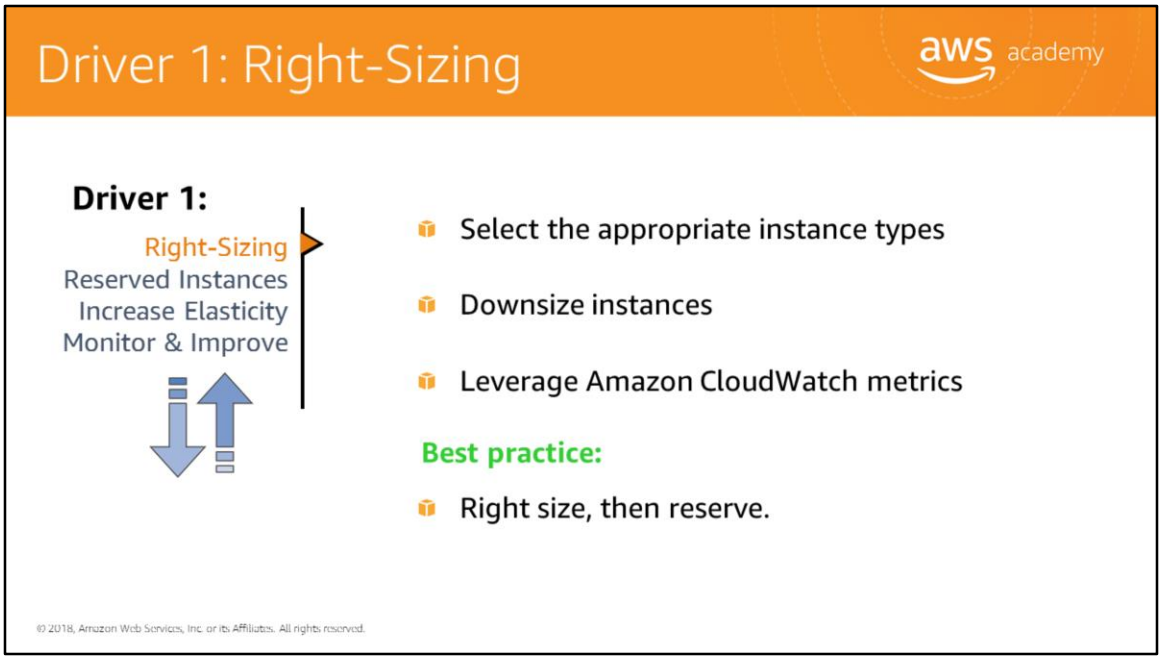
Lift and shift is a strategy for moving an application or operation from one environment to another – without redesigning the application.

The initial lift and shift model doesn't fully capture the on-going economic case for the cloud. Cost optimization over time continues to drive down costs through ongoing improvements, managed services, and an expanded scope of analysis beyond just Amazon EC2 (for example, Amazon Relational Database Service (Amazon RDS), Lambda, and storage), etc.



To optimize costs, you need to consider four consistent, powerful drivers:

- **Right-sizing** - choose the right balance of instance types
- **Reserved Instances** - Leverage reserved instances when you have long-term workloads with predictable usage patterns
- **Elasticity** - Increase elasticity using auto scaling
- **Monitor and Improve** - Monitor by measuring and analyzing your system. Continually improve and adjust as you go.



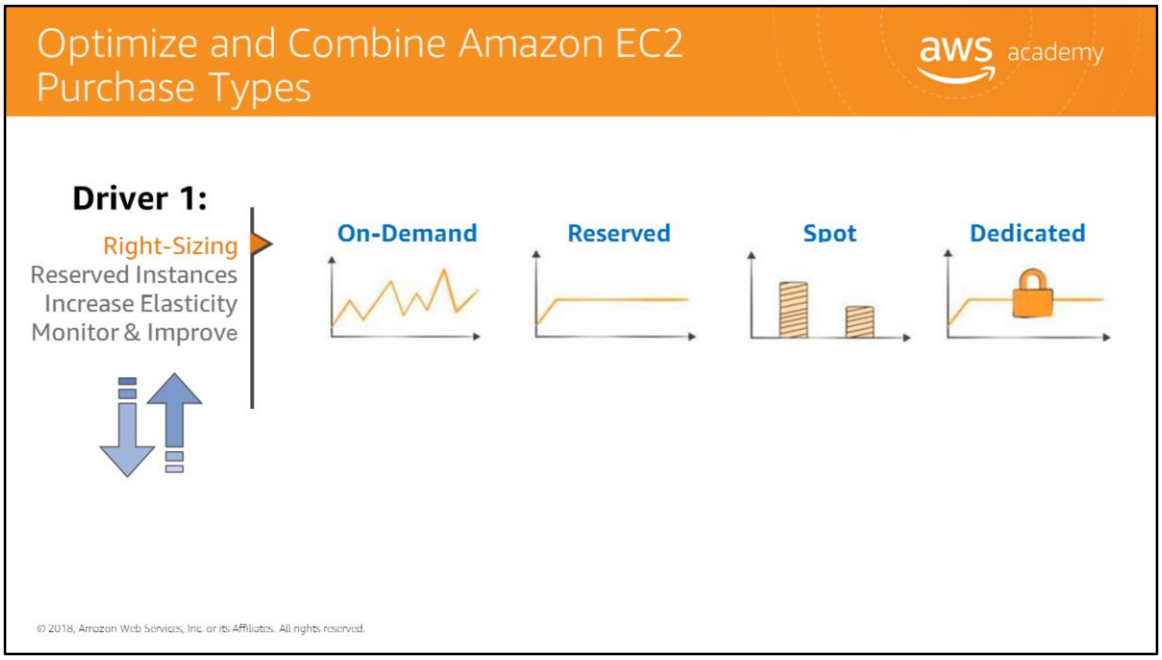
Let's look at right-sizing first. AWS offers approximately 60 instance types and sizes (https://aws.amazon.com/ec2/instance-types/). This is great for customers because it allows them to select the best fit instance for their workload. It can be difficult to know where to start and what instance is best not just from a technical but also from a cost perspective. **Right-sizing** is the process of looking at deployed resources and looking for opportunities to downsize when possible.

To Right-Size:

- **Select** the cheapest instance available that still meets your performance requirements. Right-size is defined as the cheapest instance or storage type available that meets performance requirements.
- **Review** CPU, RAM, storage, and network utilization to identify potential instances that can be ***downsized***. Also, testing is cheap so you can easily provision any type and size of instance to test your application on to identify performance requirements. Use this to your advantage for right-sizing.
- **Leverage** Amazon CloudWatch metrics and set up custom metrics. A metric represents a time-ordered set of values that are published to CloudWatch. For example, the CPU usage of a particular Amazon Amazon EC2 instance. Data points can come from any application or business activity for which you collect data. For further information on CloudWatch metrics see:

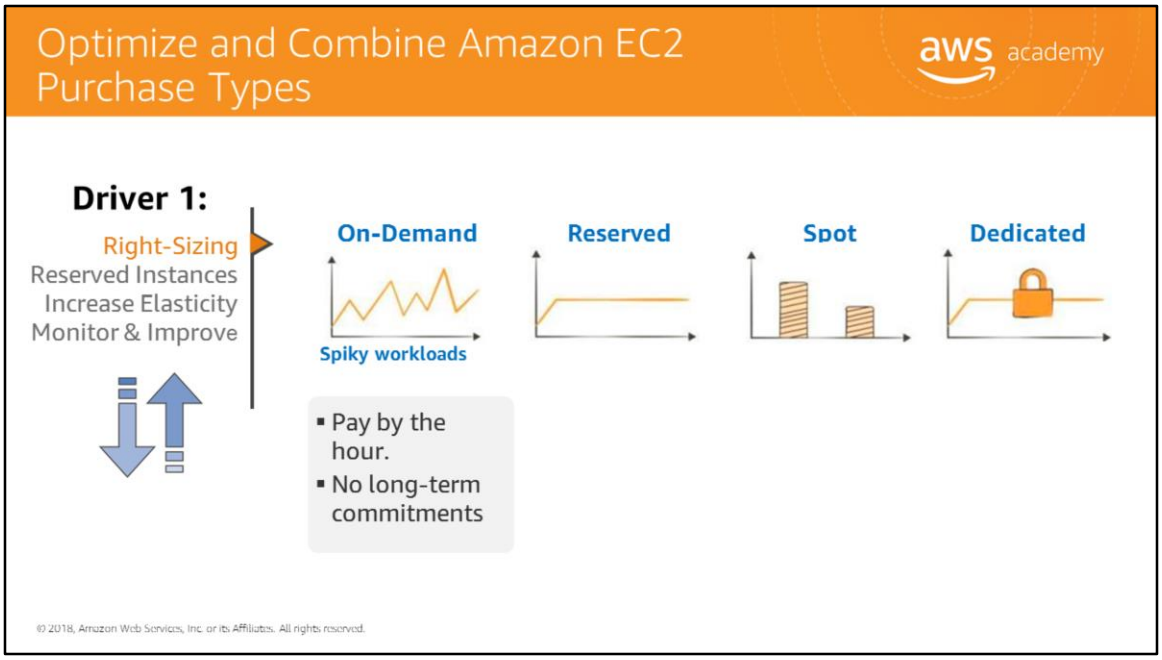
https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/cloudwatch_concepts.html#Metric.

Best Practice: Right size, then reserve.



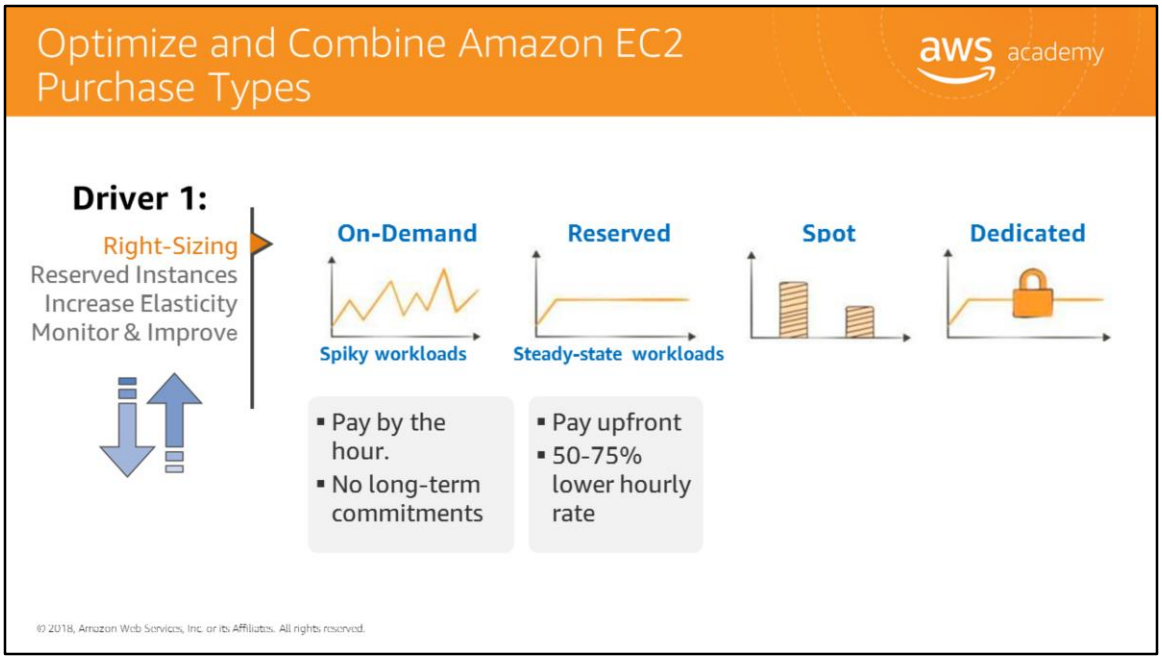
AWS provides a number of pricing models for Amazon EC2 to help customers save money. Customers can combine multiple purchase types to optimize pricing based on their current and forecast capacity needs. There are four Amazon EC2 purchase types: On Demand, Reserved, Spot, and Dedicated.

Let's review each of these in more detail.



On Demand:

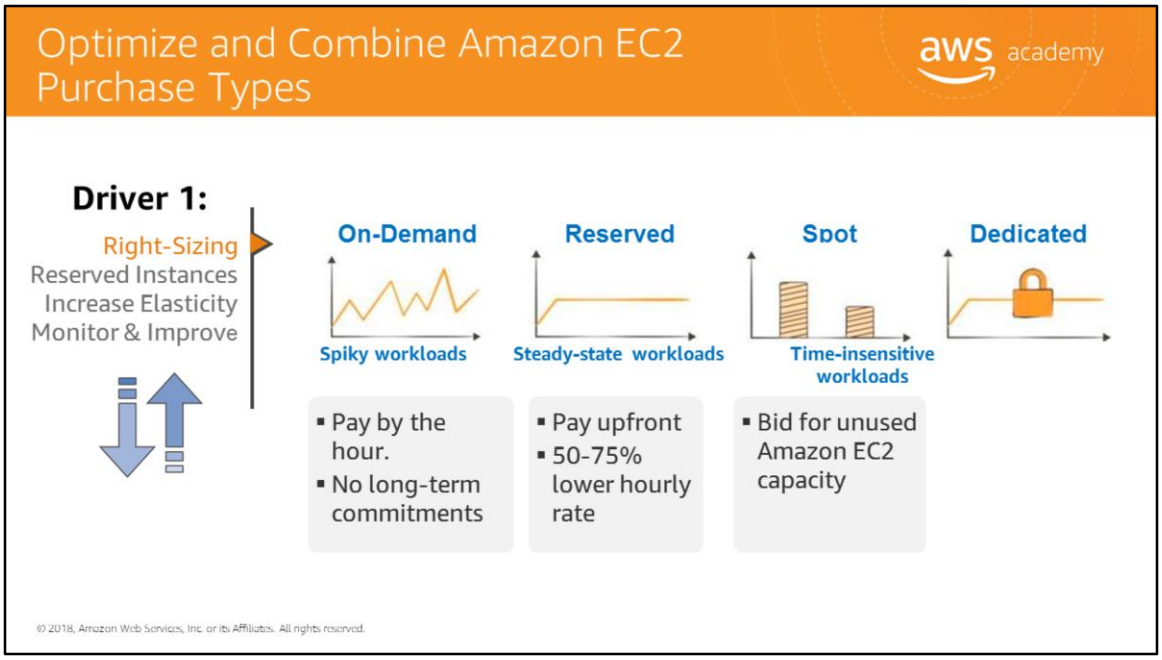
- Pay for compute capacity by per hour or per second depending on which instances you run.
- No long-term commitments or upfront payments are needed.
- Increase or decrease your compute capacity depending on the demands of your application and only pay the specified per hourly rates for the instance you use.



Reserved Instances give you the option to make one upfront payment for each instance you want to reserve at a significant discount. Reserved Instances provide you with a significant discount (up to 75%) compared to On-Demand instance pricing. When you purchase a Reserved Instance, you can choose between a Standard or Convertible offering class.

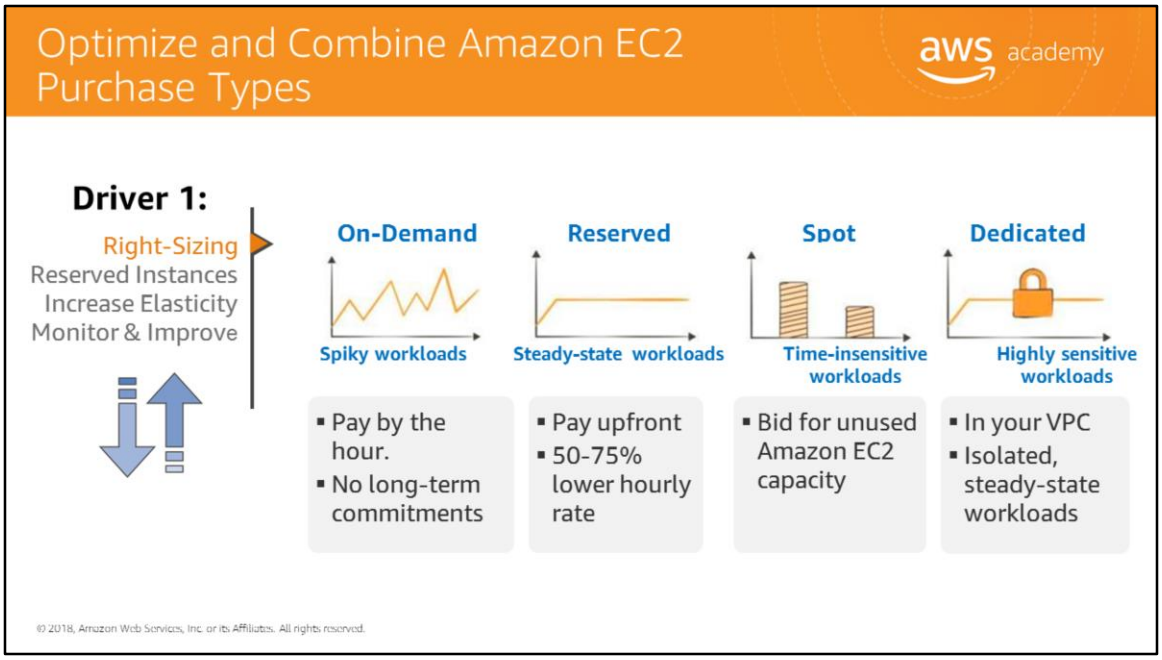
- **Standard**: With standard, some attributes, such as instance size, can be modified during the term; however, the instance type cannot be modified. You cannot exchange a Standard Reserved Instance, only modify it.
- **Convertible**: With convertible, the instance can be exchanged during the term for another Convertible Reserved Instance with new attributes including instance family, instance type, platform, scope, or tenancy

Reserved pricing works well for steady-state workloads with committed utilization.



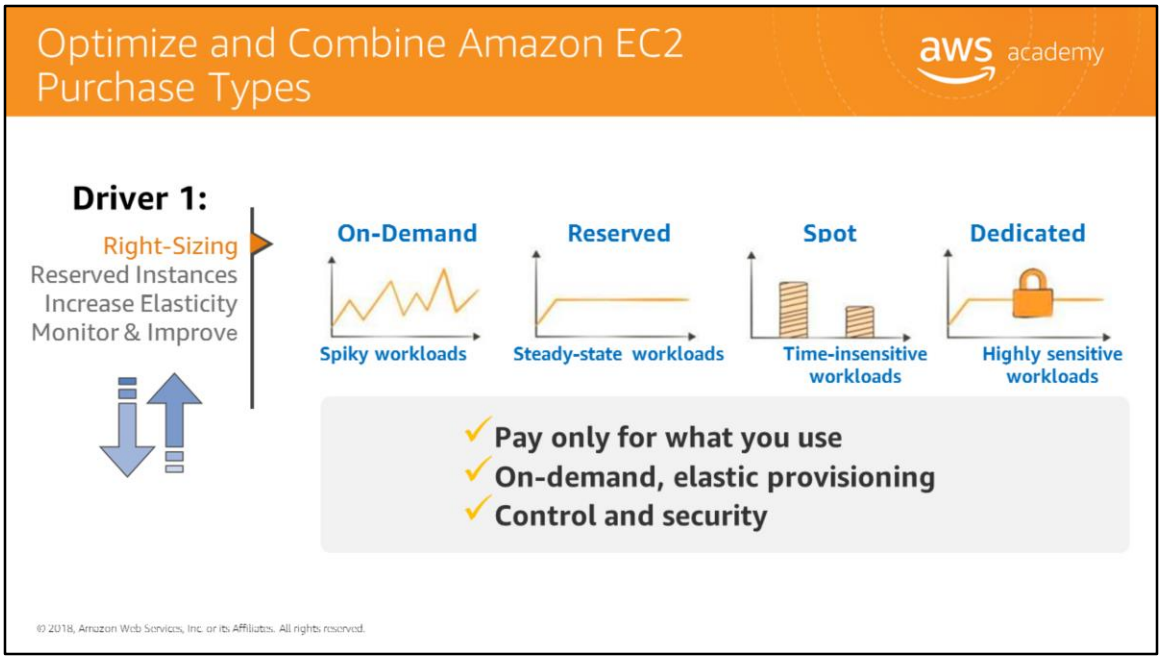
Spot Instances enable you to bid for unused Amazon EC2 capacity. Instances are charged the Spot Price, which is set by Amazon and fluctuates depending on the supply of and demand for Spot Instance capacity. Amazon EC2 Spot instances allow you to request spare Amazon EC2 computing capacity for up to 90% off the On-Demand price.

Spot pricing offers the best hourly rate and works best for workloads that are not time-dependent and which can afford to be interrupted.

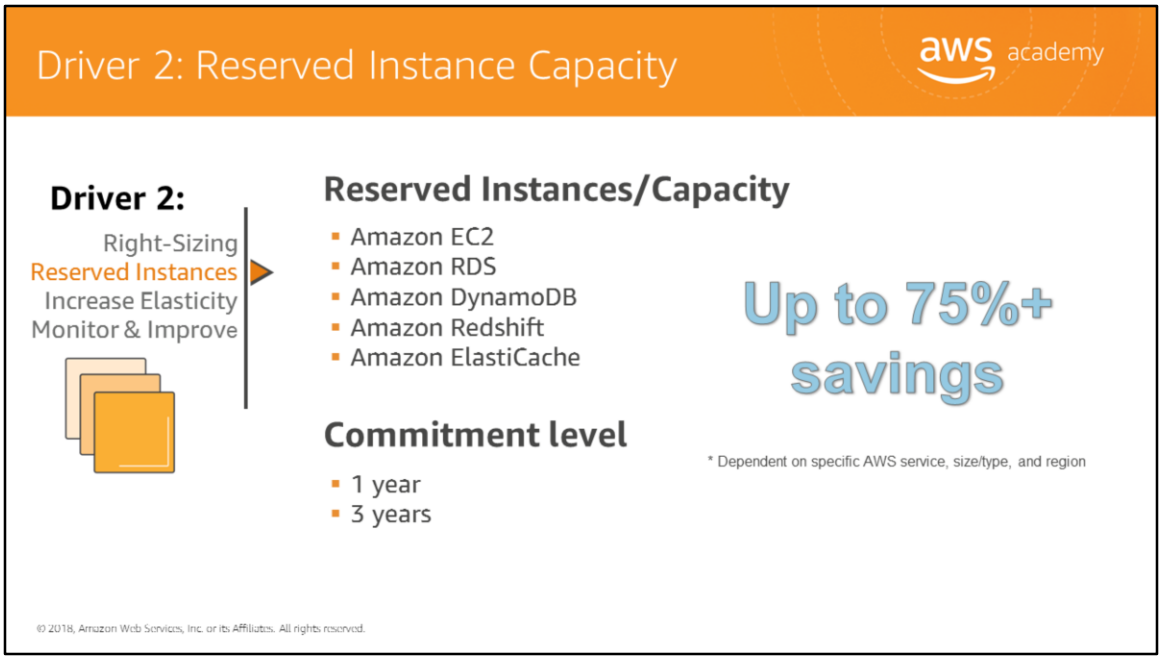


Dedicated Instances run on hardware dedicated to a single customer. Dedicated Instances ensure that your Amazon EC2 compute instances are isolated at the hardware level.

Some customers use Dedicated Instances to allow them to run third-party software, where the licensing model demands that the hardware is dedicated to one tenant. They then fill the rest of their workloads with either On-Demand or Spot Instances.



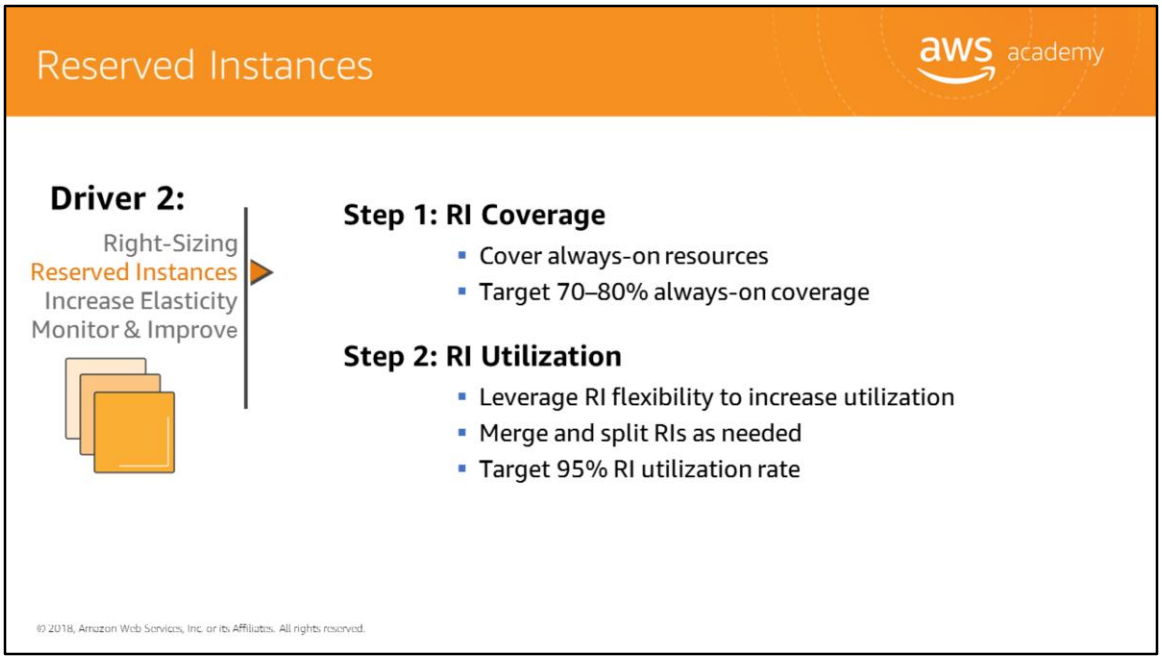
The AWS pricing models help you optimize your cost savings based on your unique requirements. You can take full advantage of cloud computing benefits and on-demand elastic provisioning with the control and security your applications require, while paying only for what you use.



After you have settled on an instance type, you have the option of purchasing a Reserved Instance. This is an **upfront commitment to purchase capacity** in a particular AWS region, which will **dramatically reduce your running costs**. A **Reserved Instance is a billing construct**; it ensures you have capacity available in the Availability Zones you have selected and purchased for that instance type. Reserved Instances are currently offered as one- or three-year commitments, and your requirements may change before the Reserved Instance commitment expires.

Besides treating a Reserved Instance as a 24x7 resource, it is also possible for you to combine a Reserved Instance if your workload is time-dependent. For example, let's assume you have a Reserved Instance for a multi-purpose instance type like an m4.large. You only need to run this instance during office hours for a total of nine hours (8:00am to 5:00pm). However, you have another workload in the same Availability Zone that can use the same instance type and be run after office hours (5:00pm to 8:00am). You could select the same instance type (m4.large) and start the evening workload on that instance after the daytime instance has shut down. After the first instance is shut down, the Reserved Instance hourly rate will apply to the after-hours instance, thus maximizing your overall cost efficiency.

It's important to **continually reevaluate your instance selection**, because workloads and instance types will change over time.

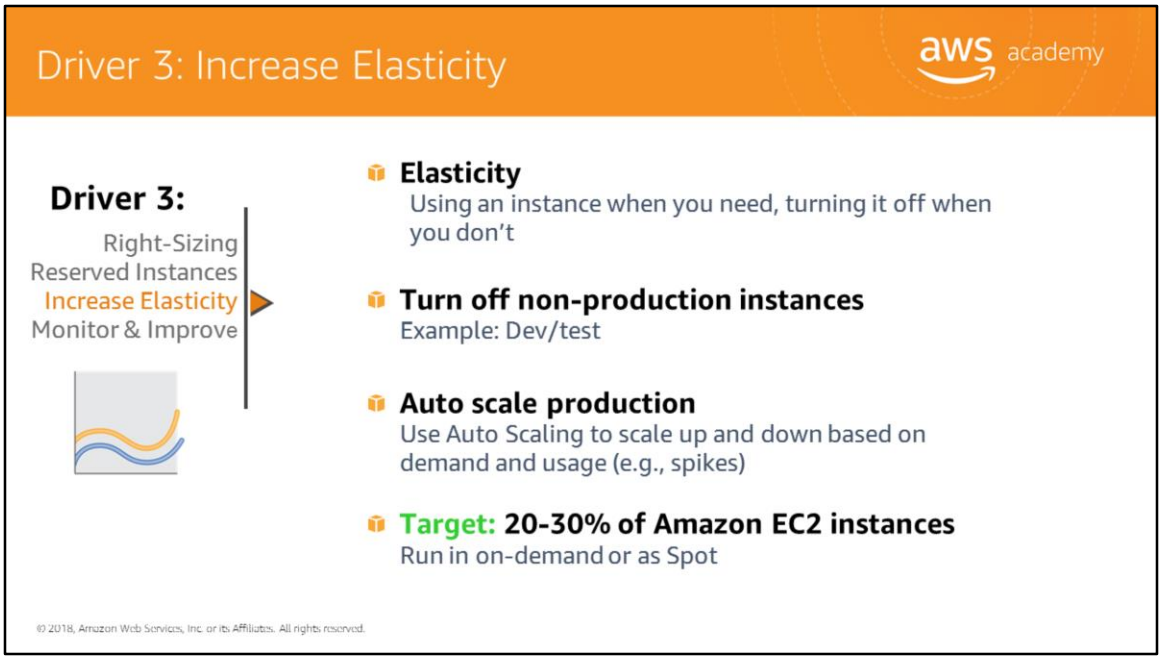


Reserved Instances:

Step 1: Reserved Instance Coverage - Cover always-on resources with standard or convertible Reserved Instances

Step 2: Increase Reserved Instance Utilization

- Known architectures: Leverage Standard Reserved Instance flexibility to increase utilization.
- Growing or changing architectures: Leverage Convertible Reserved Instances across families, sizes, and OS.
- Regional Benefit: Consolidated billing, reservation not critical

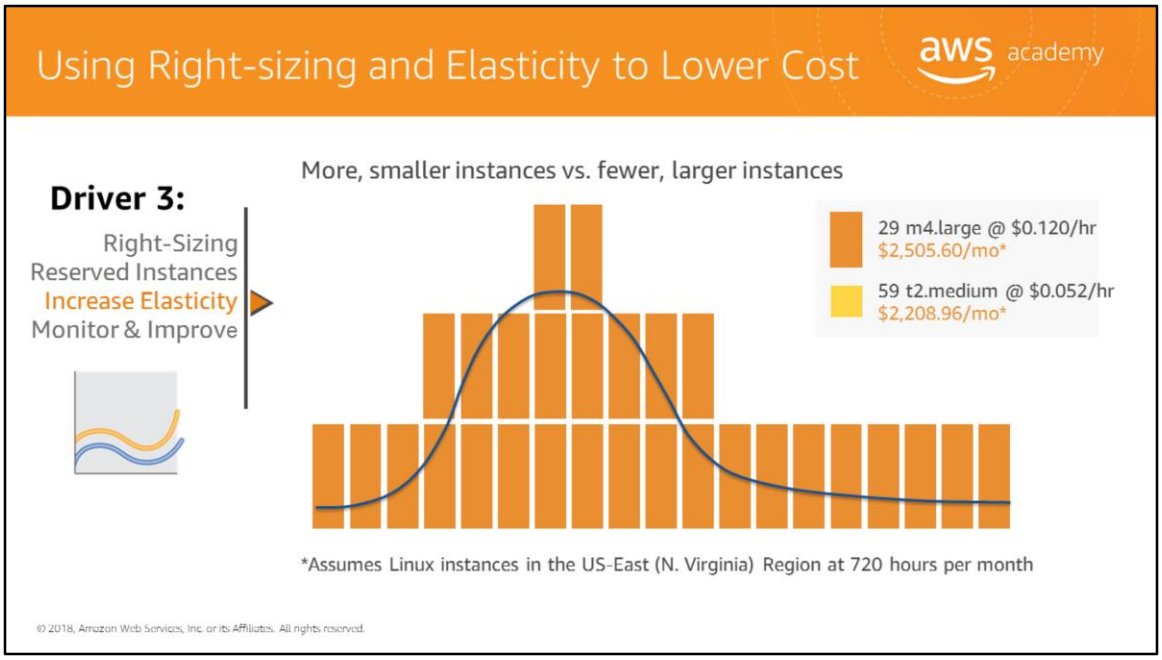


Elasticity is using an instance when you need it, but turning it off when you don't. It's one of the most central tenets of the cloud, but we often see customers go through a learning process to operationalize this in order to drive cost savings.

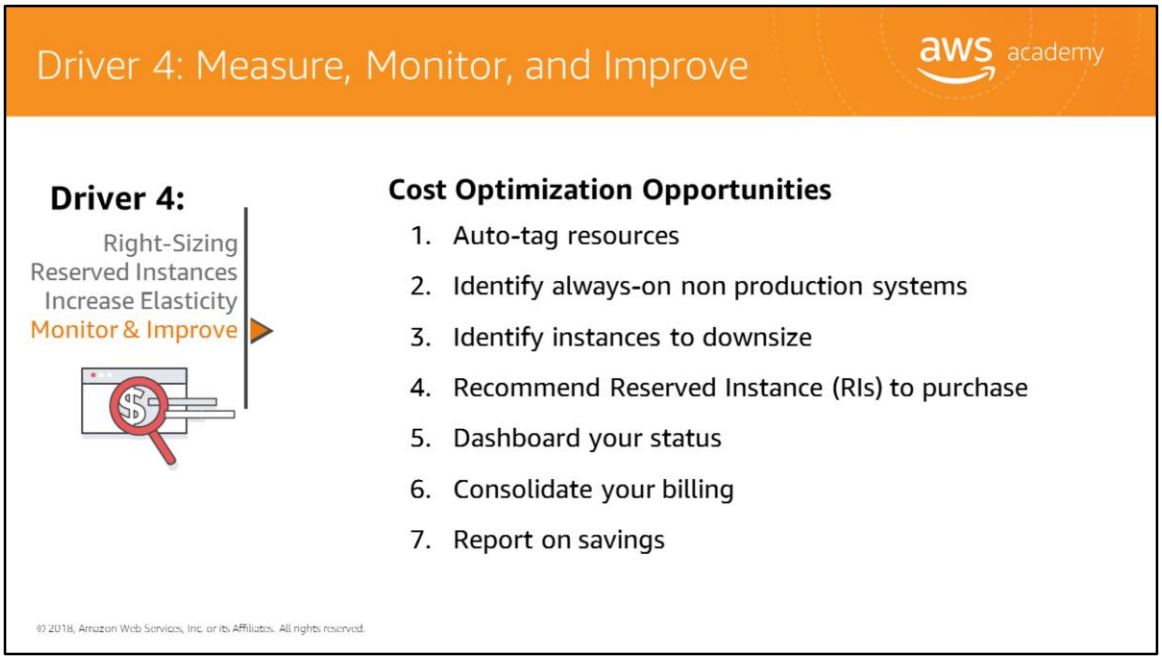
The easiest way for large customers to leverage this is to look for the "low-hanging fruit, such as non-production environments or dev/test workloads. If you're running dev/test out of a single time zone, for example, you can easily **turn off those instances outside of business hours** and reduce their cost by 80%. There's a reason why the light switch is by the door: *turn off the lights on your way out of the office each night.*

For production workloads, getting more precise and granular with auto scaling is going to help ensure that you're able to take advantage of horizontal scaling in order to meet peak capacity needs, while not paying for peak capacity.

As a rule of thumb: You should be targeting 20-30% of your Amazon EC2 instances running on-demand or as spot, and you should be looking to maximize elasticity within this group.



Do not manage the cloud like you would manage a data center: this is a new operational model. Use more and smaller instance as opposed to fewer, larger instances. The savings can be significant, for example, there is a14% savings from using t2s vs. m4s.

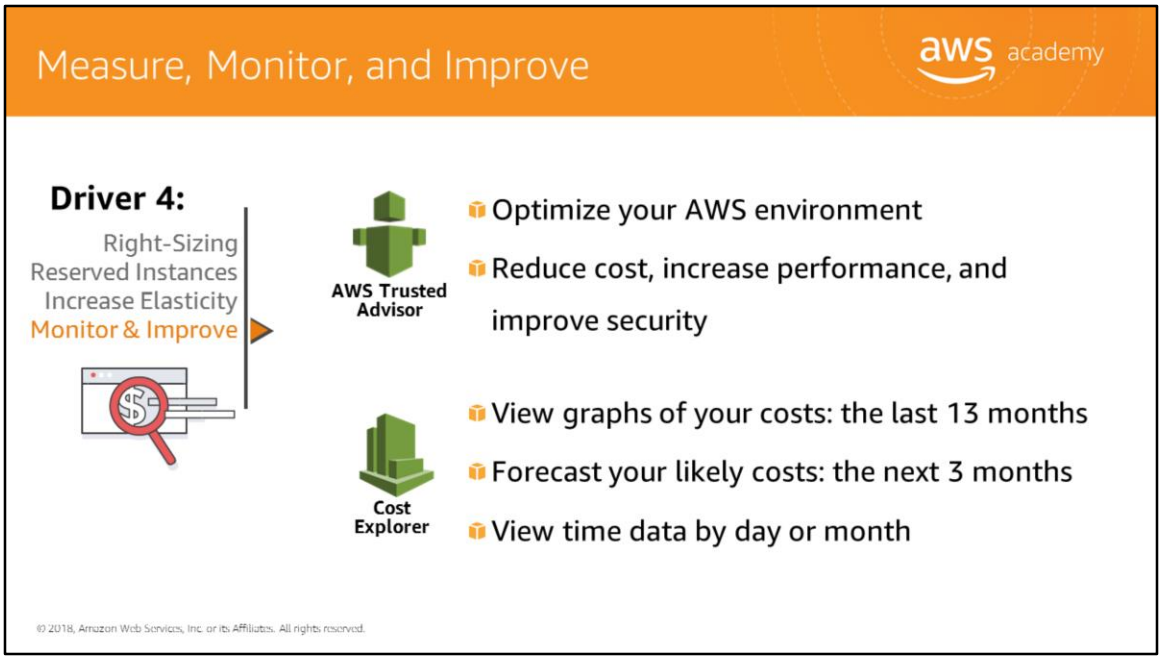


At Amazon, automation is the key to success at scale. There are a few things that can provide the insights needed to drive cost optimization:

- **First**, set up tools that help you understand the opportunity. Tagging helps provide information about *what resources* are being used *by whom* and *for what purpose*. **Tags** are Key Value Pairs attached to AWS resources. They contain Metadata (data about data). Tags can sometimes be inherited to help you keep track of who provisioned resources (Auto scaling, CloudFormation, and Elastic Beanstalk can create other resources). This help you keep track what resources are doing and who provisioned the service.
- **Second**, you want tools that identify those resources that you can take action on quickly. Set up automated reports that identify instances not being turned off or that run at the wrong size.
- **Third**, set up an automated report to determine what instances to downsize. This is important because looking at thousands of instances and trying to determine what type of instance should being run is challenging. An automated tool or report can easily do that for you.
- **Fourth**, you need a tool to recommend which Reserved Instances (RIs) to buy. AWS provides recommendations through Trusted Advisor, and several partners (including CloudAbility, Cloud Checkr, Cloudyn, and Cloud Health) who also have good tools.
- **Fifth**, consolidate your billing. Consolidated billing has the following benefits: **One Bill** – You get one bill for multiple accounts, **Easy Tracking** – You can easily track each account's charges and download the cost data in CSV format, **Combined Usage** – If you have multiple accounts today, your charges might decrease because AWS combines usage from all accounts in the organization

to qualify you for volume pricing discounts.

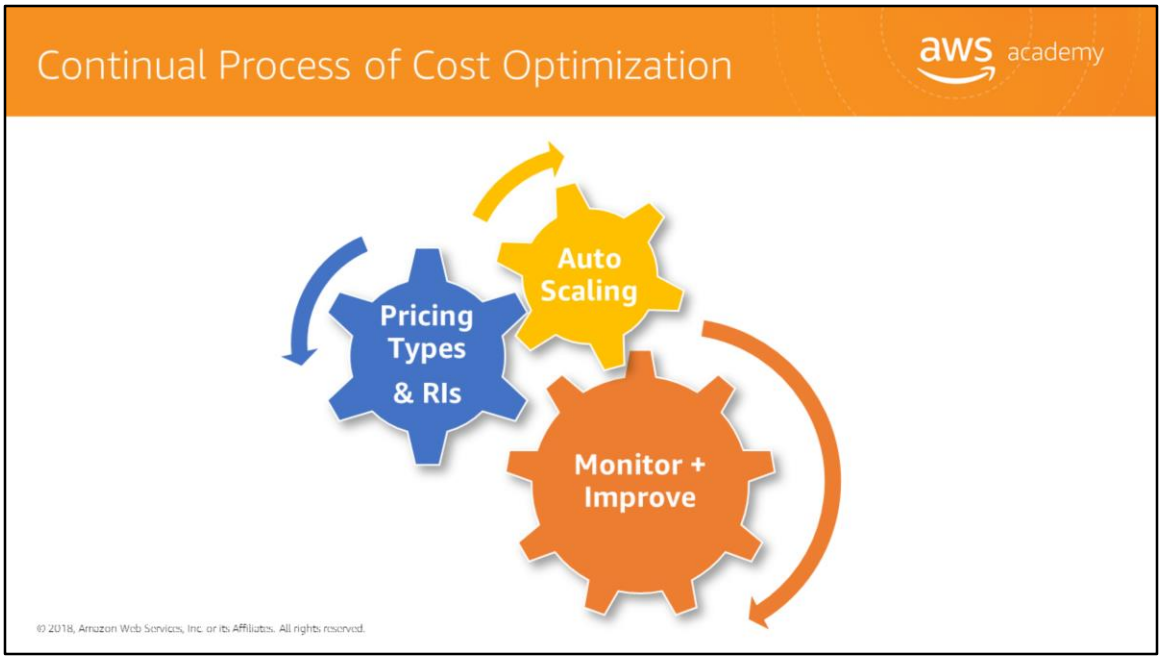
- **Finally**, it's important to report on cost optimization in order to show the opportunities that exist and show how you're progressing. Create a dashboard that can report on the savings your cost optimization efforts have achieved.



Utilize online resources to help you reduce cost, increase performance, and improve security by optimizing your AWS environment:

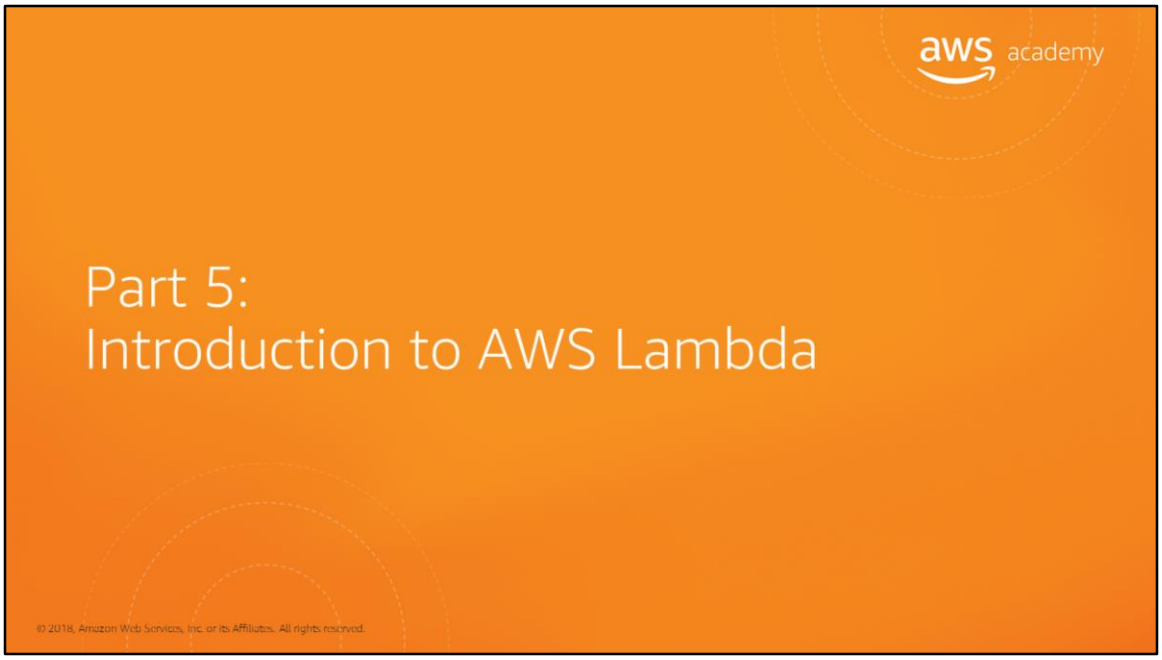
- **AWS Trusted Advisor** provides real time guidance to help you provision your resources following AWS best practices. To learn more about AWS Trusted Advisor, see: http://aws.amazon.com/premiumsupport/trustedadvisor.

- **AWS Cost Explorer** is a free tool that you can use to view graphs of your costs (also known as spend data) for up to the last 13 months, and forecast how much you are likely to spend for the next three months. You can use Cost Explorer to see patterns in how much you spend on AWS resources over time, identify areas that need further inquiry, and see trends that you can use to understand your costs. You can also specify time ranges for the data you want to see, and you can view time data by day or by month. To learn more about AWS Cost Explorer, see: http://aws.amazon.com/aws-cost-management/aws-cost-explorer.

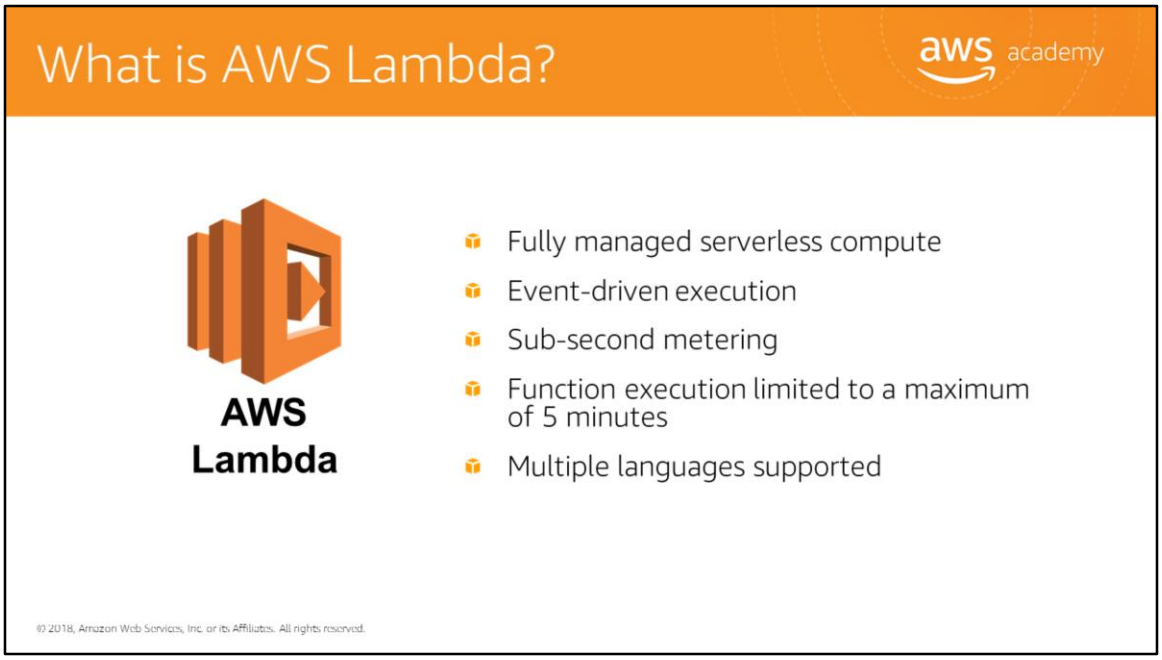


Cost optimization is an continual and interdependent process.

- Select the appropriate pricing models (instance types), and leverage Reserved Instances (RIs) according to your business requirements.
- Increase your elasticity by using auto scaling and turning off non-production instances.
- Leverage AWS tools to analyze, monitor, and improve your costs.

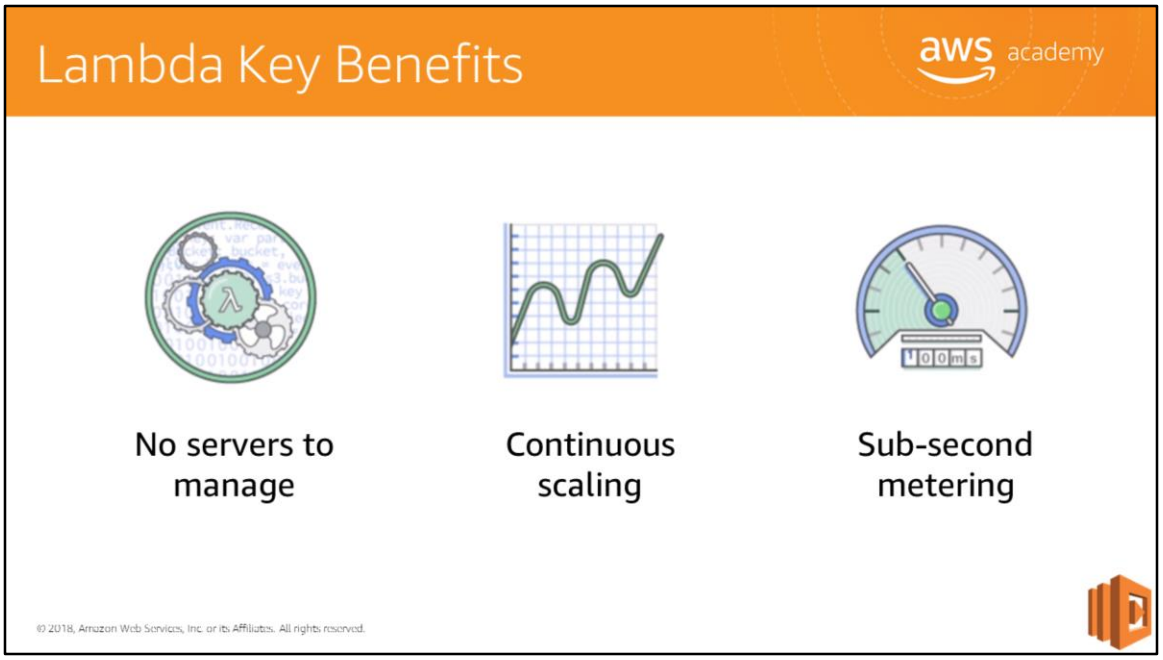


Let’s look another compute service, AWS Lambda, an event-driven serverless compute service. Lambda lets you run code without provisioning or managing servers. You pay only for the compute time you consume - there is no charge when your code is not running.



Lambda executes your code only when needed and scales automatically to thousands of requests per second.

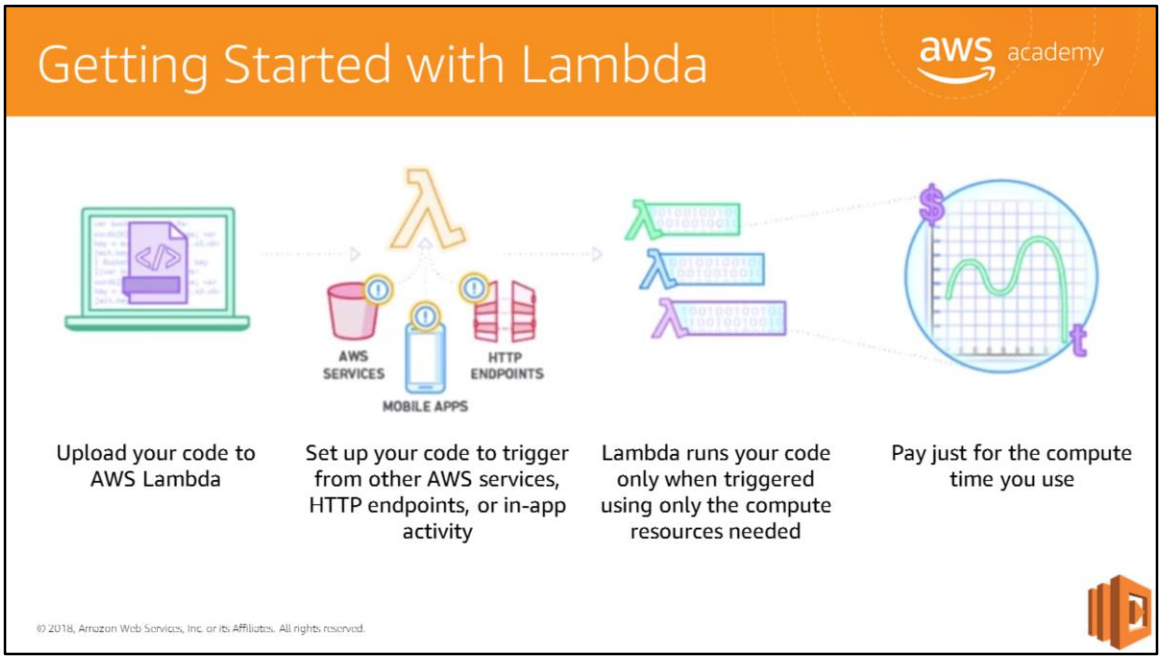
With Lambda, you can run code for virtually any type of application or backend service - all with zero administration. Just upload your code and Lambda takes care of everything required to run and scale your code with high availability. You can set up your code to automatically trigger from other AWS services or call it directly from any web or mobile app. It should be noted that function execution on Lambda is limited to a maximum of 5 minutes.



Lambda offers several benefits:

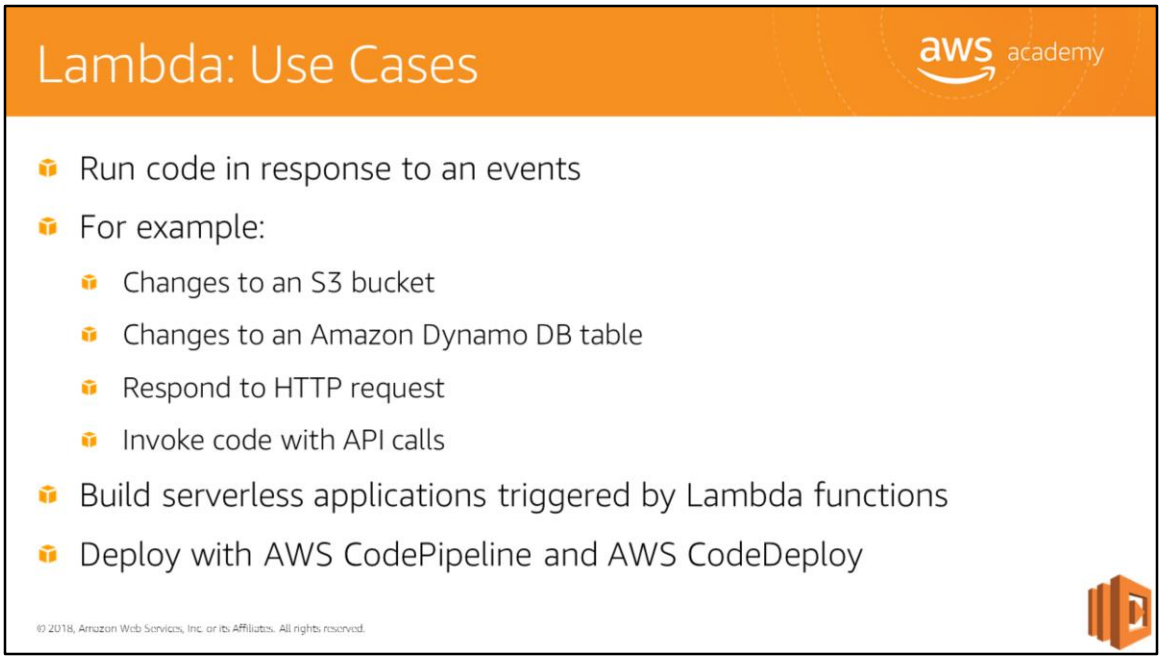
- You only pay for the compute you use. You don't pay for compute time when your code is not running. This makes AWS Lambda ideal for variable in intermittent work loads.
- You can run code for virtually any application or backend service, all with zero administration, including server and operating system maintenance. Just upload your code and Lambda takes care of everything required to run and scale your code with high availability.
- You can set up your code to automatically trigger from other AWS services, or call it directly from any web or mobile app.

Lambda supports a variety of different programming languages including Go, NodeJS, Java, C#, and Python.



It is really simple to build your Lambda function. You configure your environment, then you upload your code and watch it run. It is as simple as that.

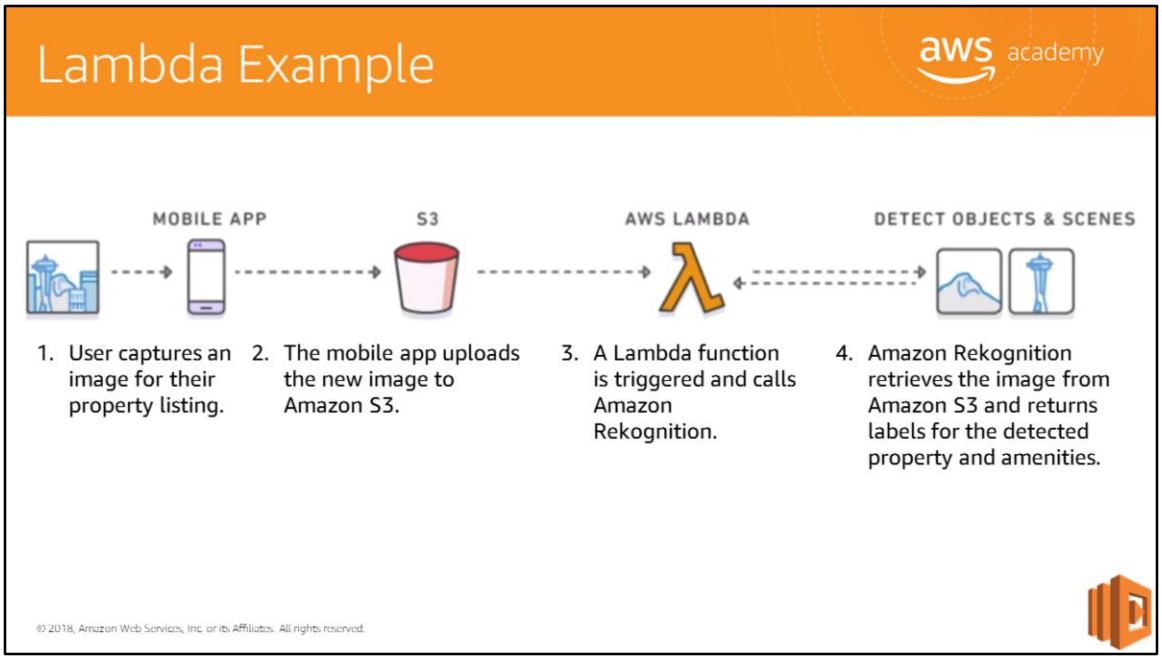
Lambda is billed on the number times your code is triggered and for each 1/100 millisecond of execution time.



So, how can you use Lambda?

You can use it for event driven computing. For example:

- You can run code in response to events, including changes to an S3 bucket or an Amazon DynamoDB table.
- You can respond to HTTP requests using Amazon API Gateway.
- You can invoke your code using API calls made using the AWS SDKs.
- You can build serverless applications that are triggered by Lambda functions.
- You can automatically deploy them using AWS CodePipeline and AWS CodeDeploy.

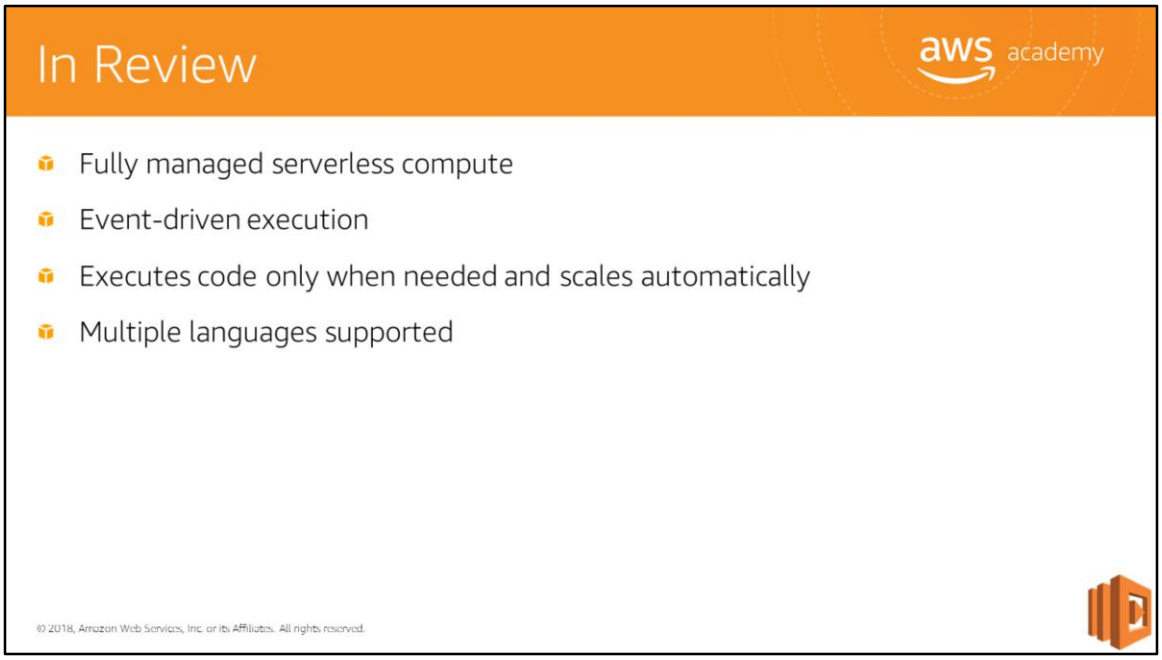


Here is an example illustrating the use of Lambda for an image recognition application. Here's how it works.

First, the user captures an image for their property using an app on their mobile phone. The mobile app then uploads the new image to Amazon S3. Adding this image to Amazon S3 triggers a Lambda function and calls Amazon Rekognition. Amazon Rekognition can identify objects, people, text, scenes, and activities. It provides highly accurate facial analysis and recognition. Lambda will retrieve the image from Amazon S3 and return labels for the property and its amenities.

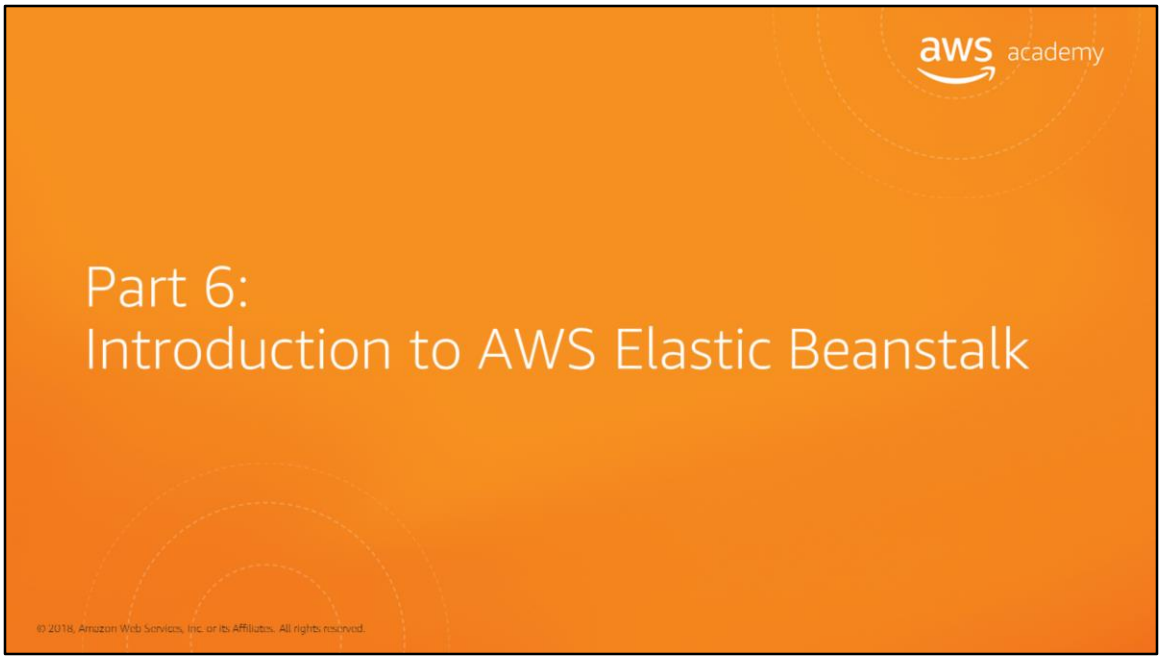
This is just one example of an Lambda use case. With Lambda, we can run code for virtually any application or backend service. Other Lambda use cases include:

- Automated backups
- Processing objects uploaded to Amazon S3
- Event-driven log analysis
- Event-driven transformations
- Internet of Things (IoT)
- Operating serverless websites



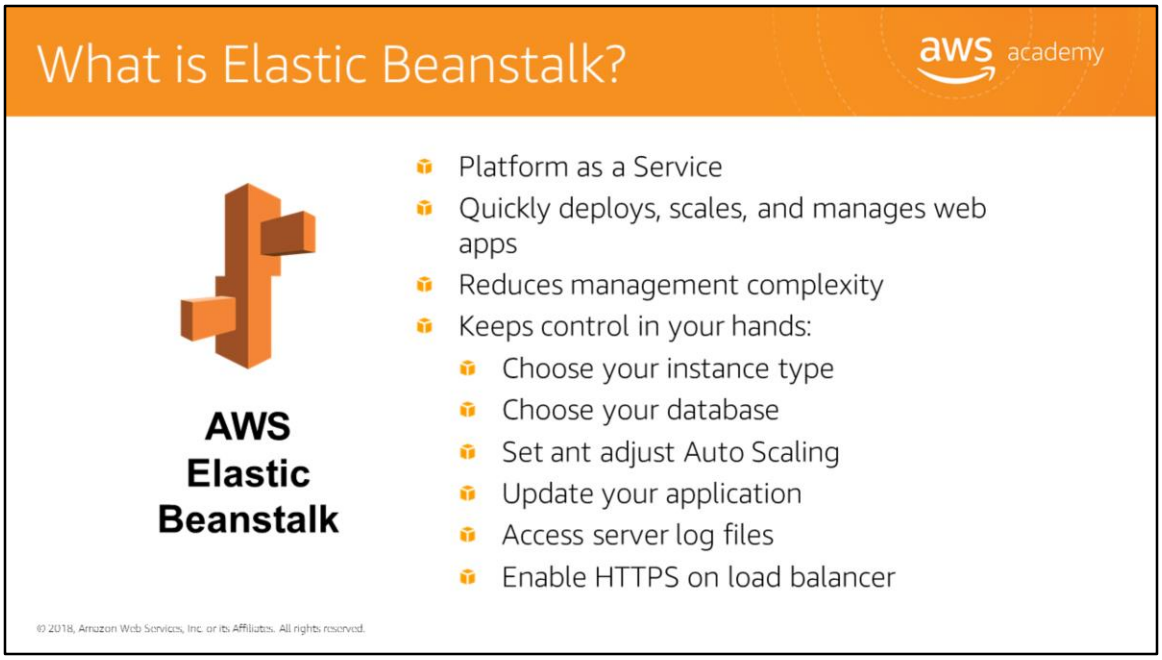
In summary, Lambda is the connective tissue for AWS services from building micro services architectures to running your applications. Lambda executes your code only when needed and scales automatically to thousands of requests per second. With Lambda, you can run code for virtually any type of application or backend service - all with zero administration.

For more information about Lambda, see https://aws.amazon.com/lambda/.

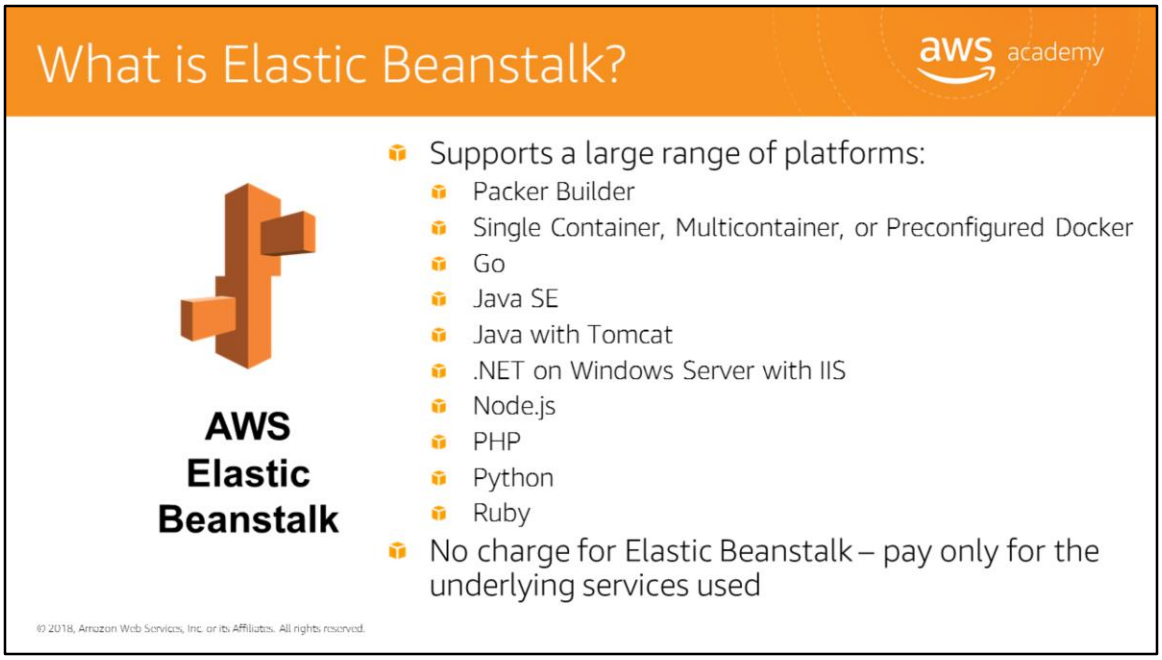


Let's look another compute service, AWS Elastic Beanstalk. It's is an easy-to-use service for deploying and scaling web applications and services developed with Java, .NET, PHP, Node.js, Python, Ruby, Go, and Docker on familiar servers such as Apache, Nginx, Passenger, and IIS.

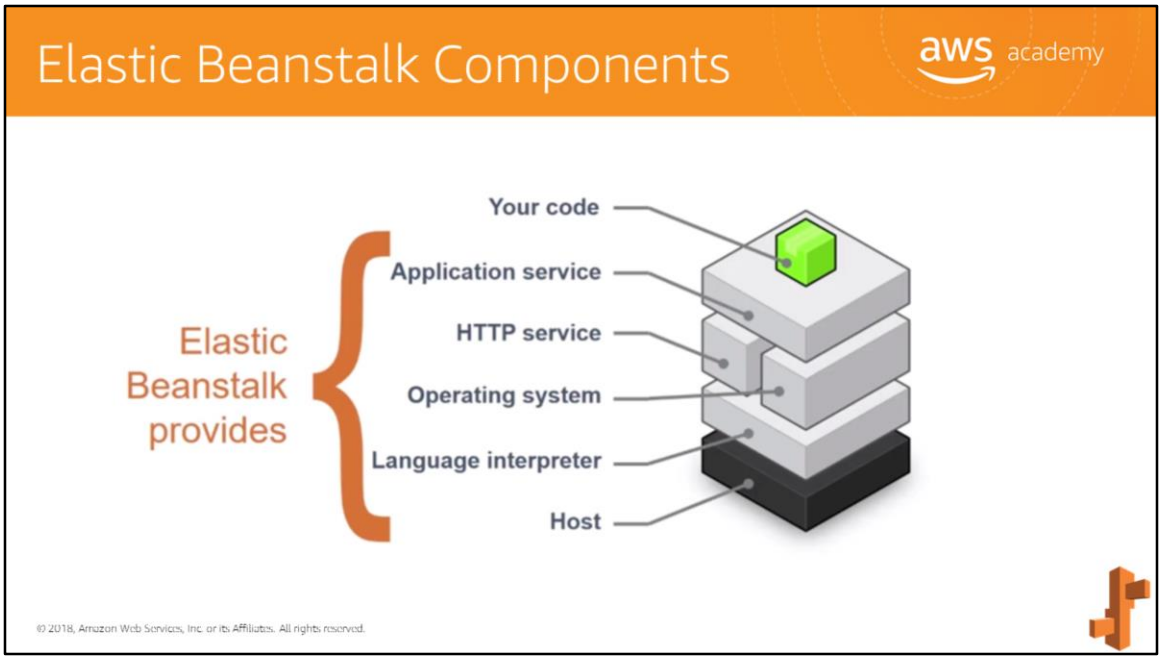
You can simply upload your code and Elastic Beanstalk automatically handles the deployment, from capacity provisioning and load balancing to automatic scaling and application health monitoring. At the same time, you retain full control over the AWS resources powering your application and can access the underlying resources at any time.



Elastic Beanstalk is a Platform as a Service. With the entire platform already built, you simply upload your code. This facilitates quick deployment of your applications.

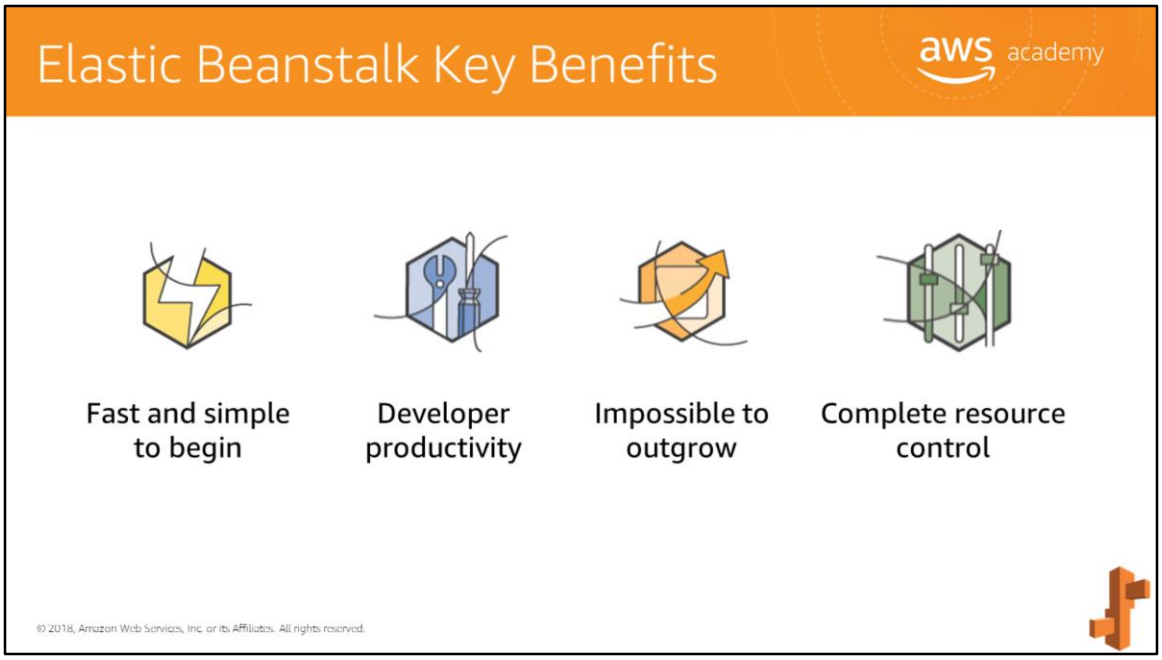


Elastic Beanstalk supports a large range of platforms. Platforms supported include Packer Builder, Single Container / Multicontainer or Preconfigured Docker, Go, JavaSE, Java with Tomcat, .NET on Windows Server with IIS, Node.js, PHP, Python, and Ruby. So, you can develop your application to meet your requirements and simply deploy on Elastic Beanstalk.



Elastic Beanstalk provides all the applications services that you need for your application. The only thing you need to create your code, deploy it according to your needs. This makes is very quick and easy to deploy your application.

Updates to your application are easy when you deploy it. You simply upload the new code.

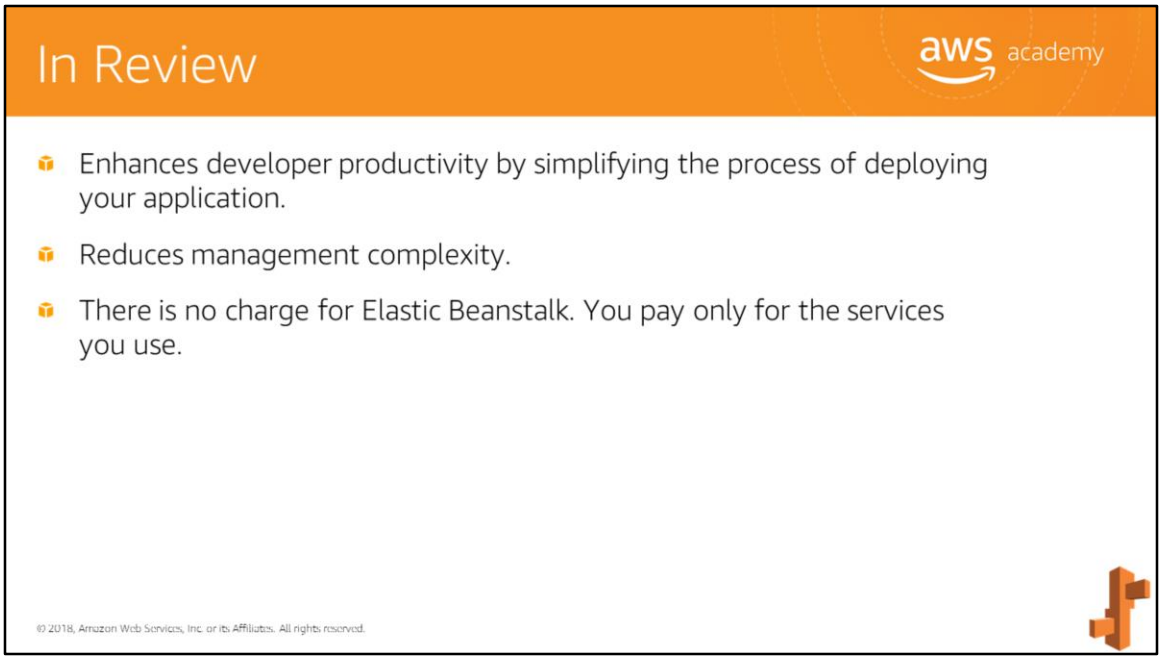


Elastic Beanstalk is the fastest and simplest way to deploy your application on AWS. Use the AWS Management Console, a Git repository, or an integrated development environment (IDE) such as Eclipse or Visual Studio to upload your application, and Elastic Beanstalk automatically handles the deployment details of capacity provisioning, load balancing, automatic scaling, and application health monitoring.

You can improve your productivity by focusing on writing code rather than spending time managing and configuring servers, databases, load balancers, firewalls, and networks. Elastic Beanstalk provisions and operates the infrastructure and manages the application stack (platform) for you, so you don't have to spend the time or develop the expertise. It keeps the underlying platform running your application up-to-date with the latest patches and updates.

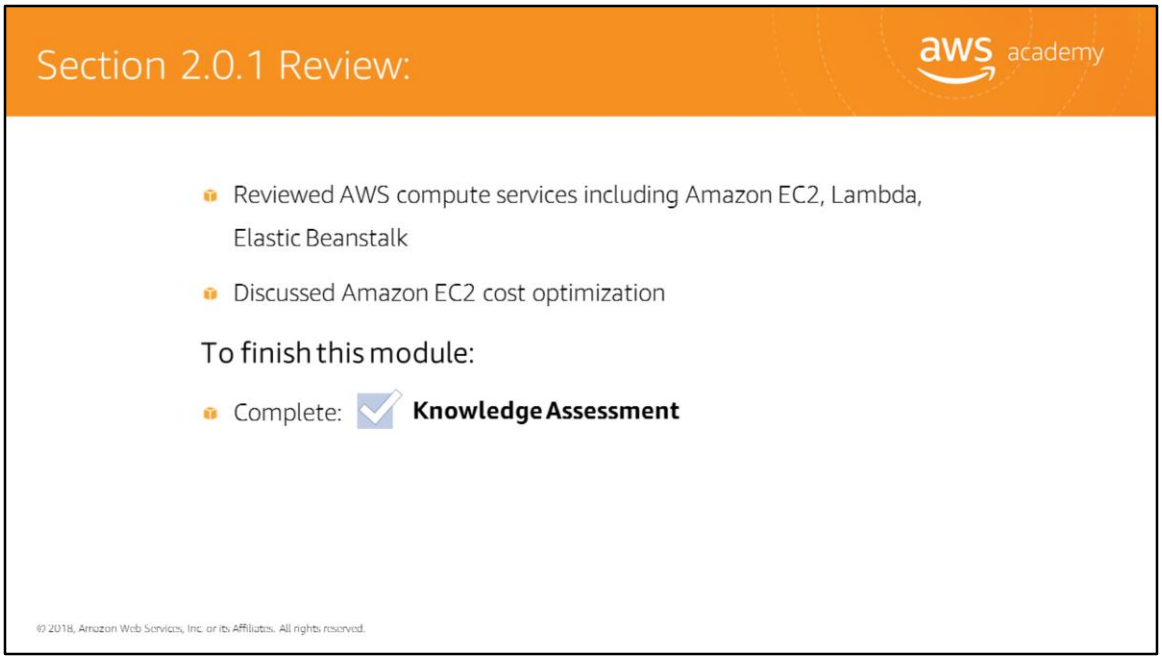
With Elastic Beanstalk, your application can handle peaks in workload or traffic while minimizing your costs. It automatically scales your application up and down based on your application's specific need using easily adjustable Auto Scaling settings. You can use CPU utilization metrics to trigger Auto Scaling actions.

You have the freedom to select the AWS resources, such as Amazon EC2 instance type, that are optimal for your application. Elastic Beanstalk lets you "open the hood" and retain full control over the AWS resources powering your application. If you decide you want to take over some (or all) of the elements of your infrastructure, you can do so seamlessly by using Elastic Beanstalk's management capabilities.



In summary, AWS Elastic Beanstalk can help you get your application up and running on AWS. Simply upload your application code and the service automatically handles all the details such as resource provisioning, load balancing, automatic scaling, and monitoring.

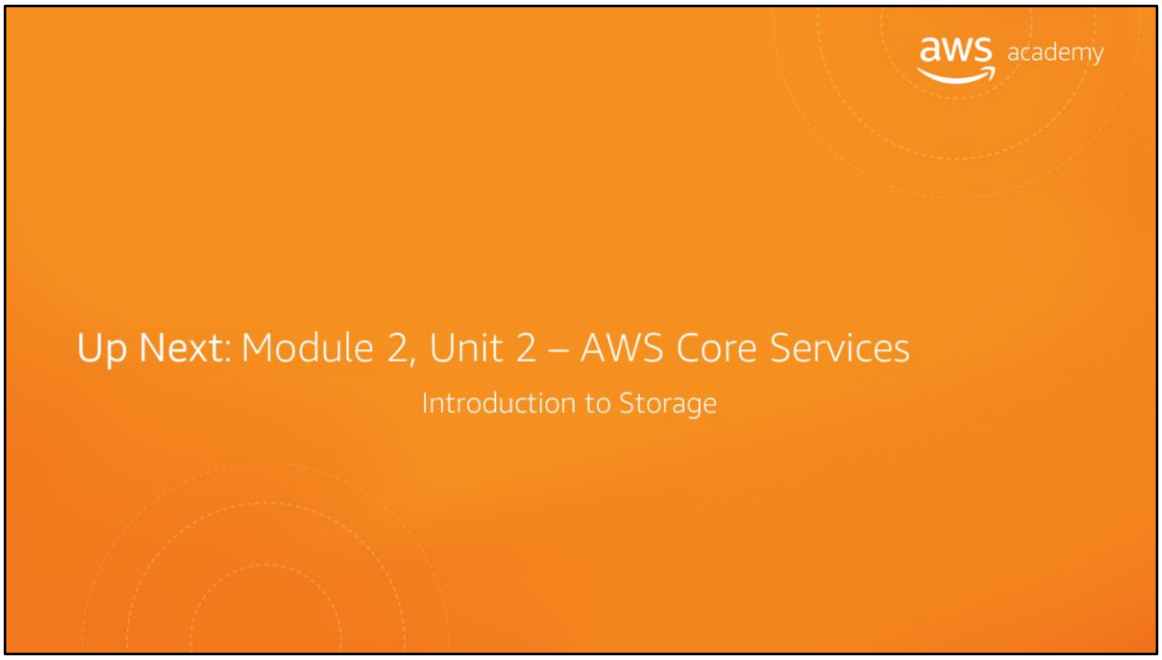
To learn more about Elastic Beanstalk, see: https://aws.amazon.com/elasticbeanstalk/.



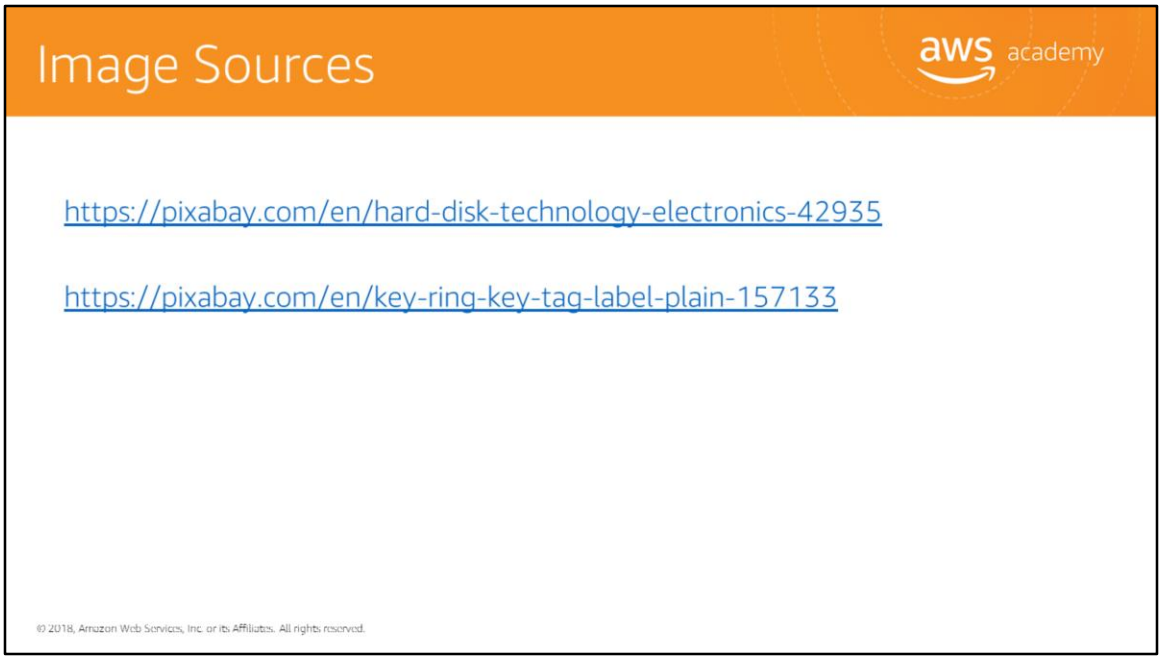
In review, we:

- Described compute services in the cloud
- Explained Amazon EC2 and cost optimization opportunities
- Explored serverless computing with Lambda
- Reviewed Elastic Beanstalk and its ability to facilitate quick application deployment

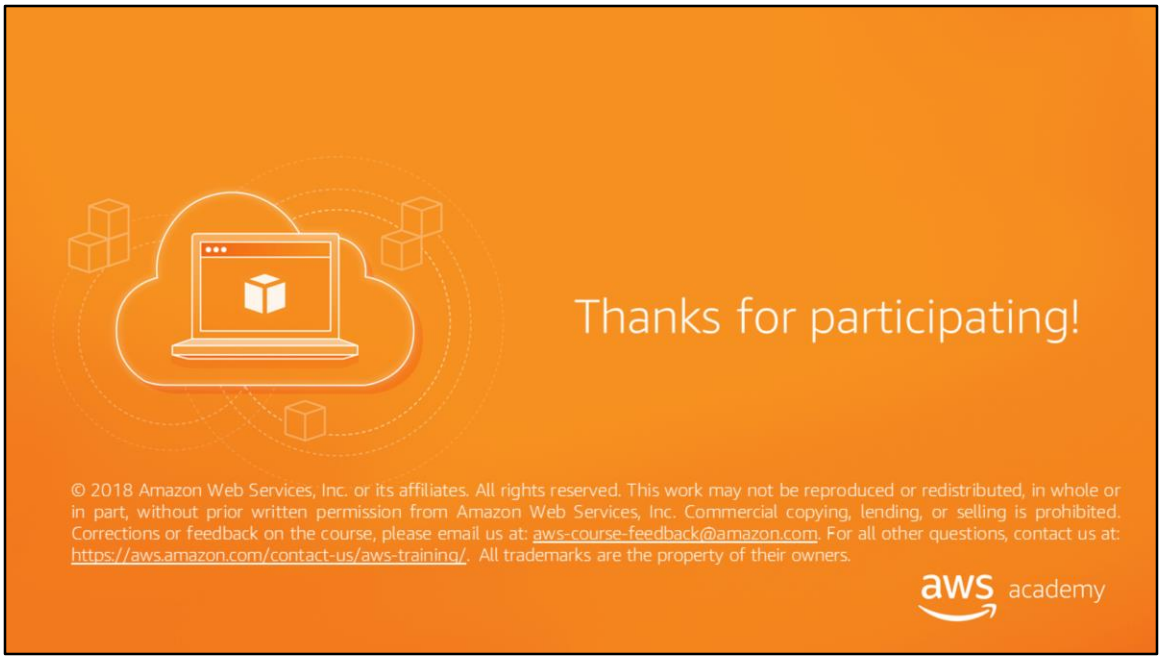
To finish this module, please complete the lab and the corresponding knowledge assessment.

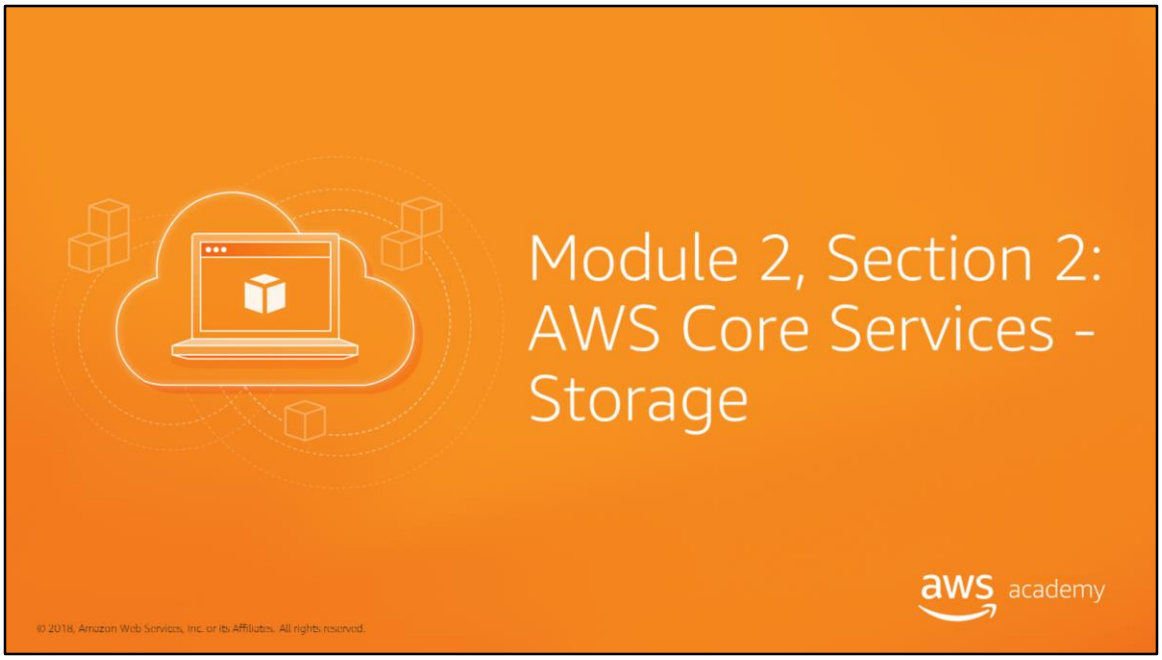


Now that we have a better understanding some of the compute services offered by AWS, in unit 2.0.2, Storage, we gain an understanding of cloud storage options.

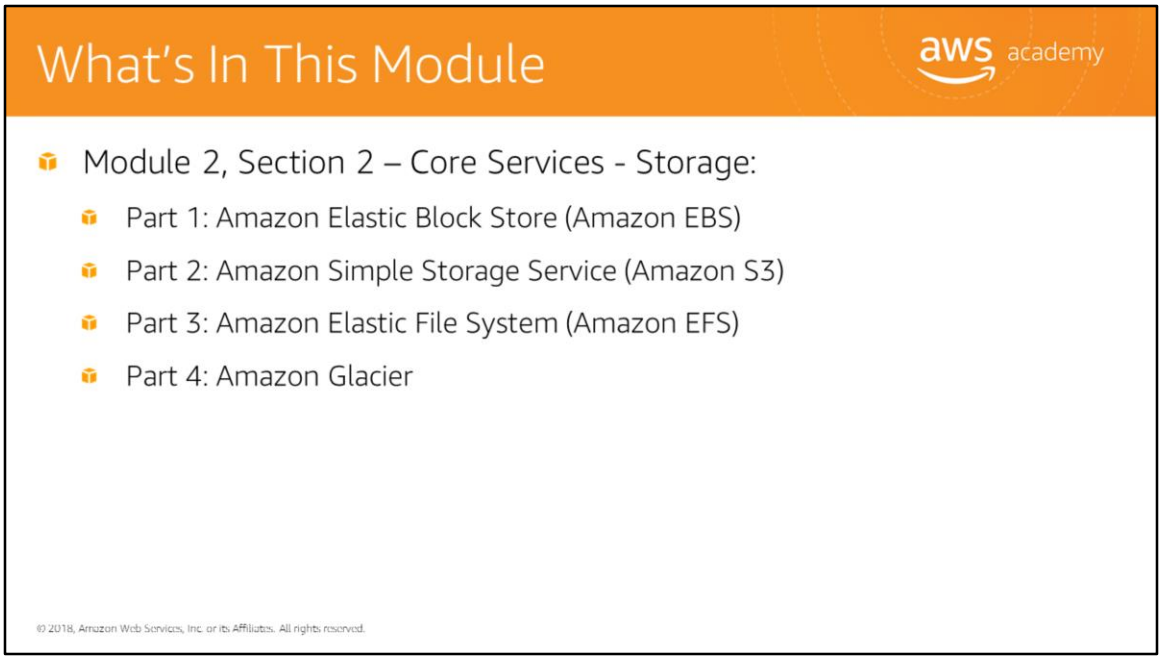


This slide contains attributions for any Creative Commons-licensed images used within this module.

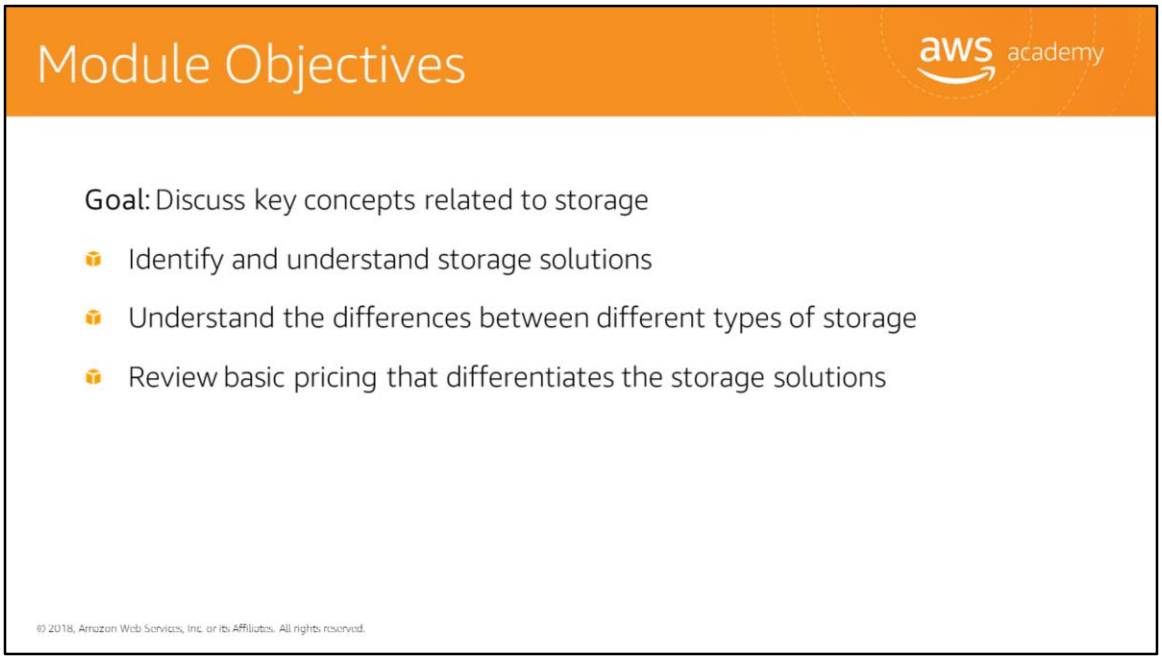




Welcome to Module 2, Section 2 – AWS Core Services - Storage.

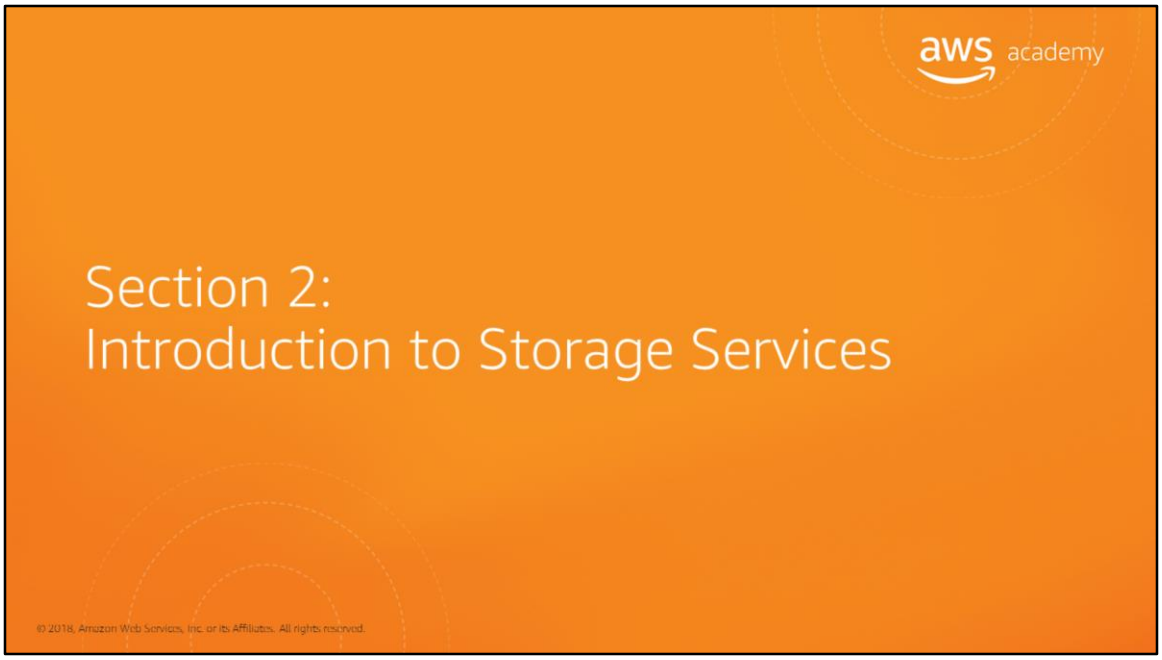


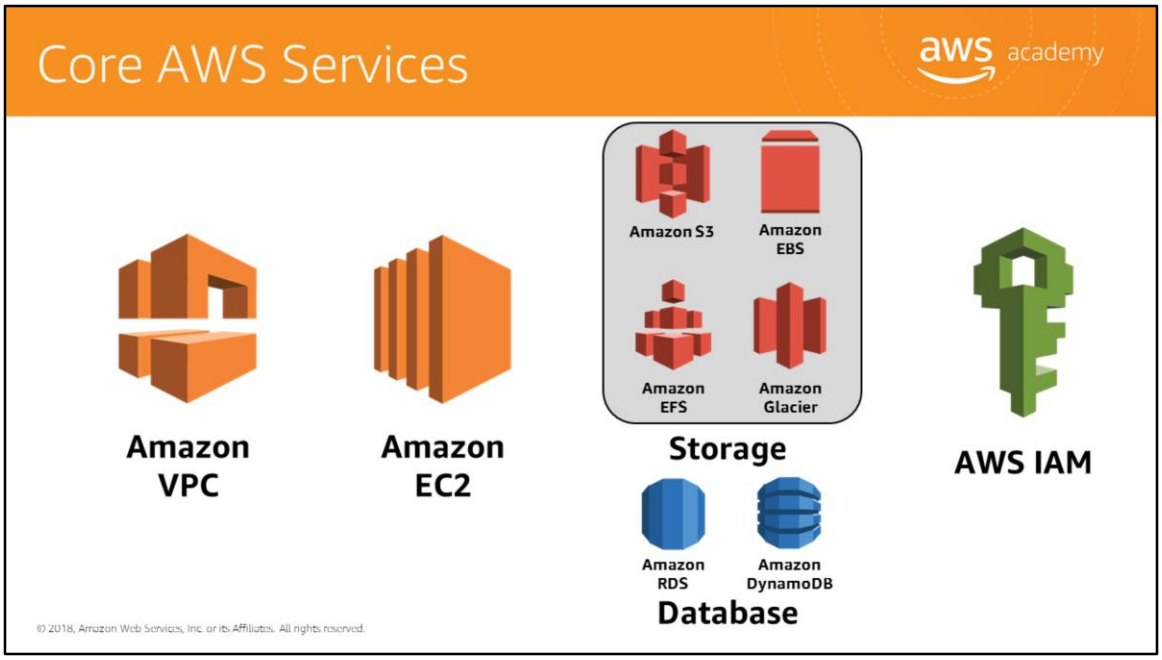
Cloud storage is a critical component of cloud computing, holding the information used by applications. Big data analytics, data warehouses, Internet of Things (IoT), databases, and backup and archive applications all rely on some form of data storage architecture. Cloud storage is typically more reliable, scalable, and secure than traditional on-premises storage systems.



The goal of this module is to help you understand the storage resources that are available to power your solution. We will also review the different pricing options that are available so you can begin to understand how different choices impact your solution cost.

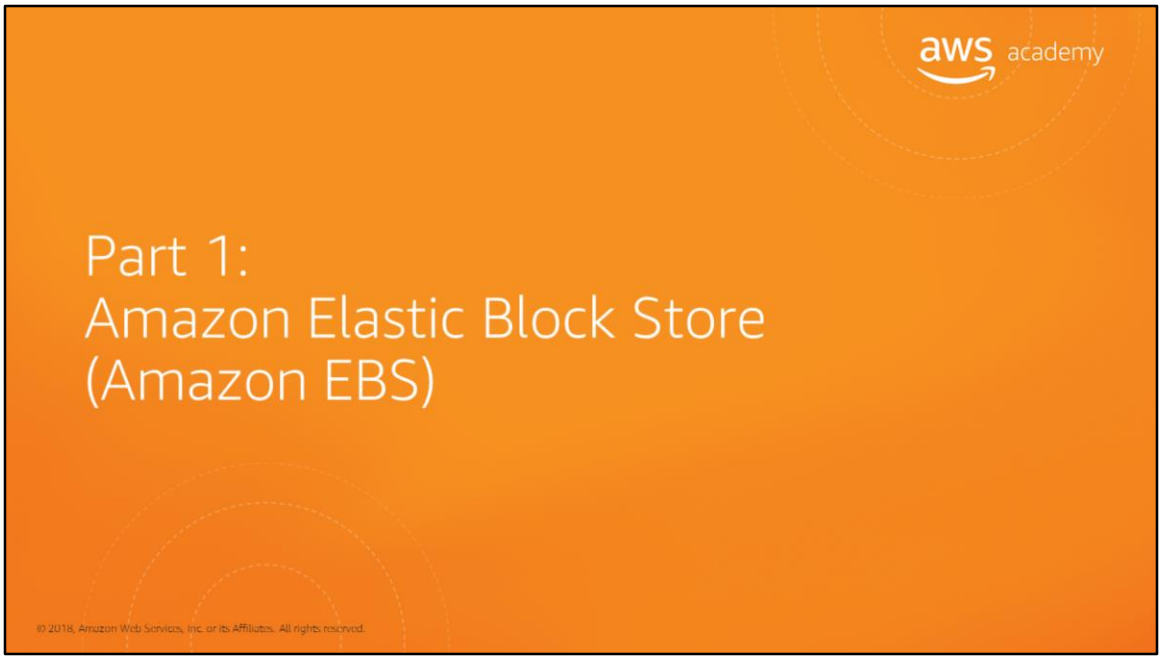
Then, you have your next opportunity to jump back into the lab and complete a Knowledge Assessment.



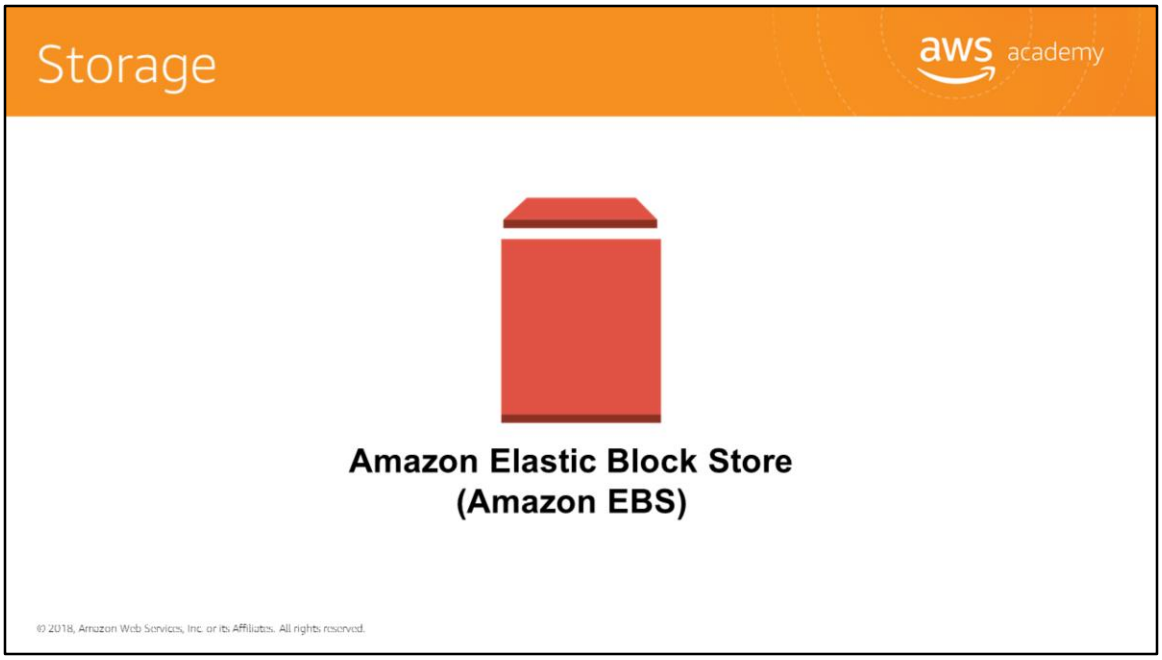


It is not surprising that storage is another AWS core service. There are three broad categories of storage: instance store ("ephemeral"), Amazon EBS, and Amazon S3.

Instance store, or *ephemeral storage*, is a **temporary storage** that it is added to your Amazon EC2 instance. Amazon EBS is **persistent**, **mountable storage**; it can be mounted as a device to an Amazon EC2 instance. Amazon EBS can only be mounted to an Amazon EC2 instance in the same Availability Zone. Like Amazon EBS, Amazon S3 is persistent storage but it can be accessed from anywhere.

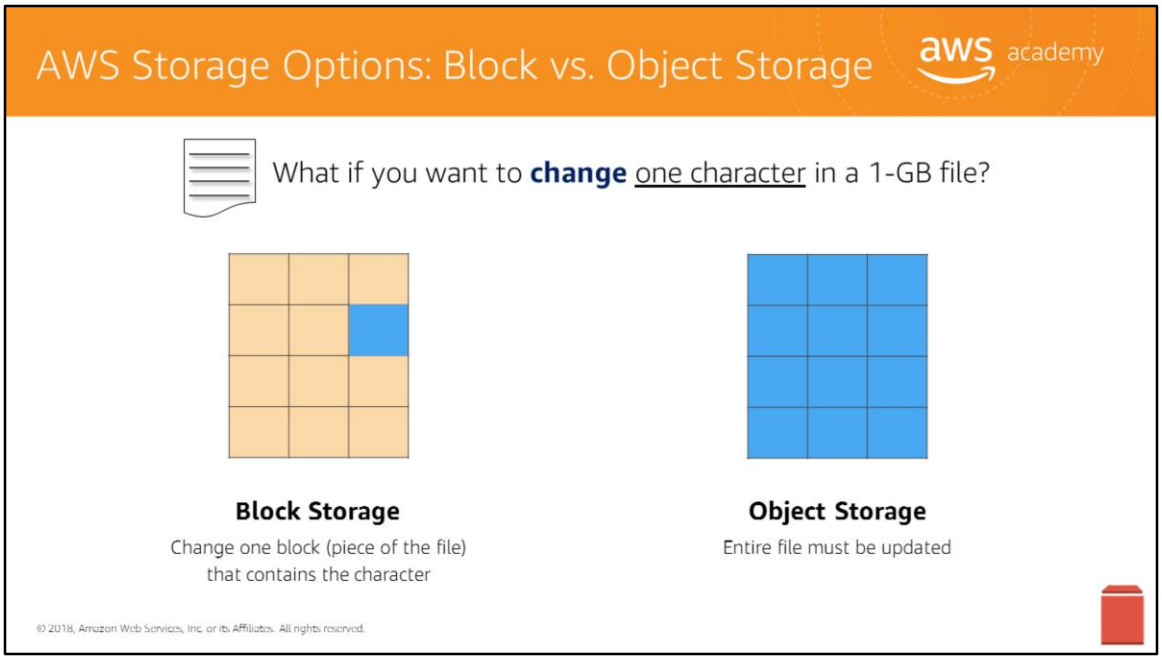


We just finished discussing compute options to power your solution. Amazon Elastic Block Store (Amazon EBS) is an AWS block storage system that is best used for storing persistent data. Amazon EBS provides highly available block level storage volumes for use with Amazon EC2 instances.

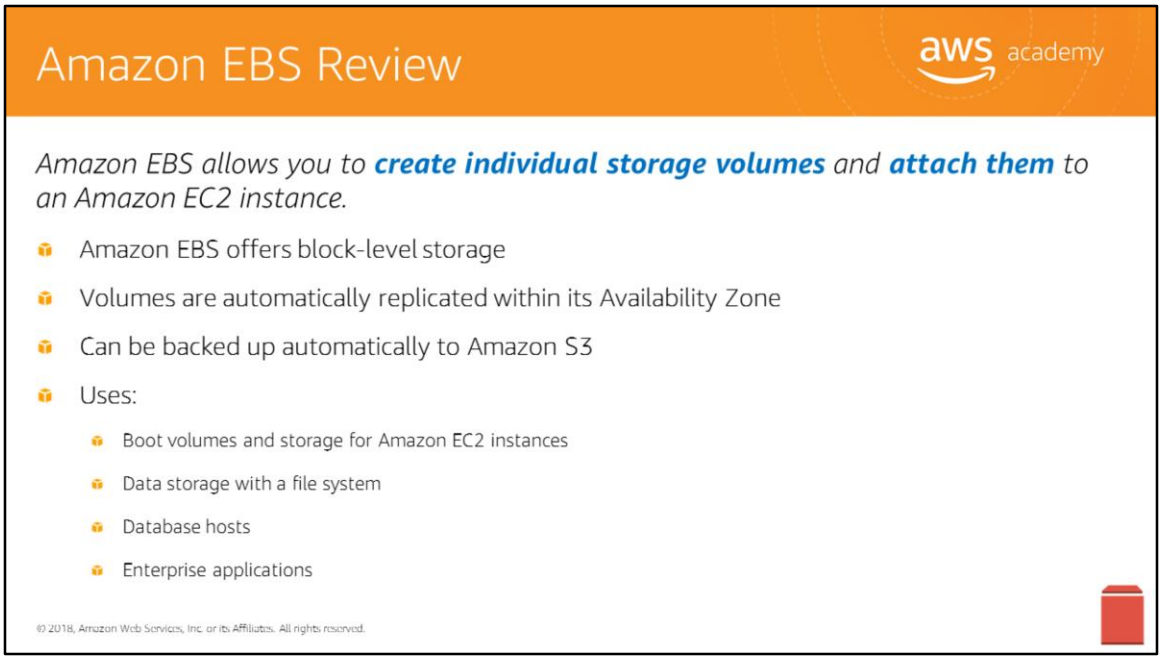


Amazon EBS provides persistent block storage volumes for use with Amazon EC2 instances in the cloud. Persistent storage is any data storage device that retains data after power to that device is shut off. It is also sometimes referred to as *non-volatile storage*.

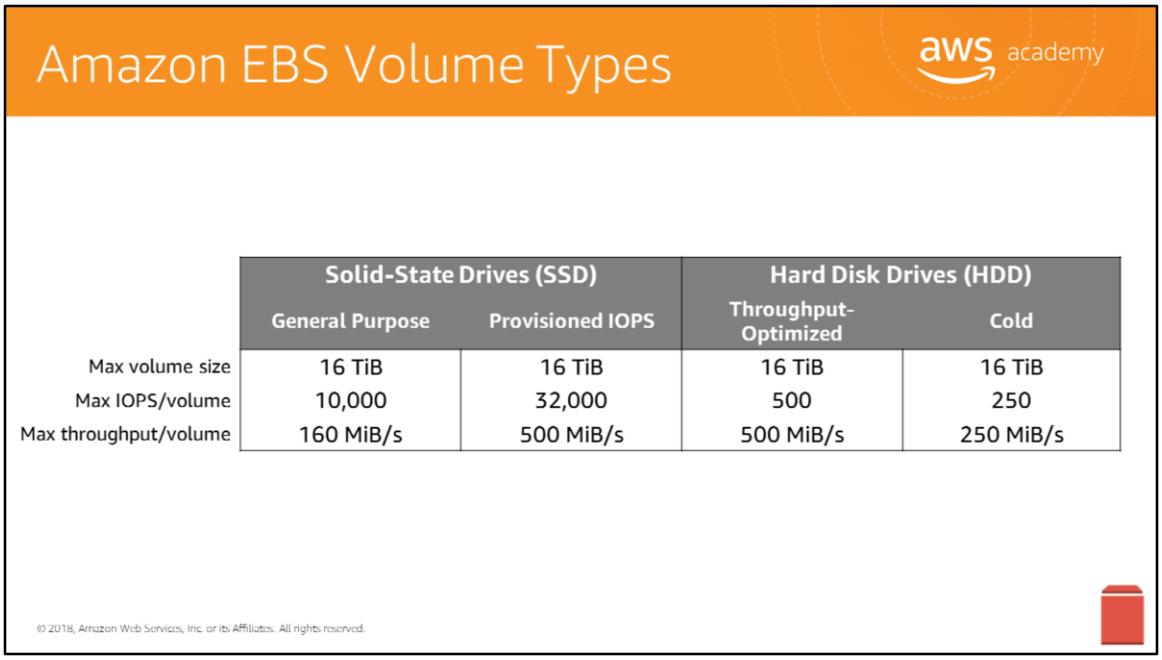
Each Amazon EBS volume is automatically replicated *within* its Availability Zone to protect you from component failure, offering high availability and durability. Amazon EBS volumes offer the consistent and low-latency performance needed to run your workloads. With Amazon EBS, you can scale your usage up or down within minutes – all while paying a low price for only what you provision.



One of the critical concepts to understanding the differences between some storage types is whether they offer "block-level" storage or "object-level" storage. This difference has a major impact on the throughput, latency, and cost of your storage solution: block storage solutions are typically faster and use less bandwidth but cost more than object-level storage.



Amazon EBS volumes provide durable, detachable, block-level storage (like an external hard drive) for your Amazon EC2 instances. Because they are directly attached to the instances, they can provide extremely low latency between where the data is stored and where it might be used on the instance. For this reason, they can be used to run a database with an Amazon EC2 instance. Amazon EBS volumes can also be used to back up your instances into Amazon Machine Images (AMI), which are stored in Amazon S3 and can be reused to create new Amazon EC2 instances later.

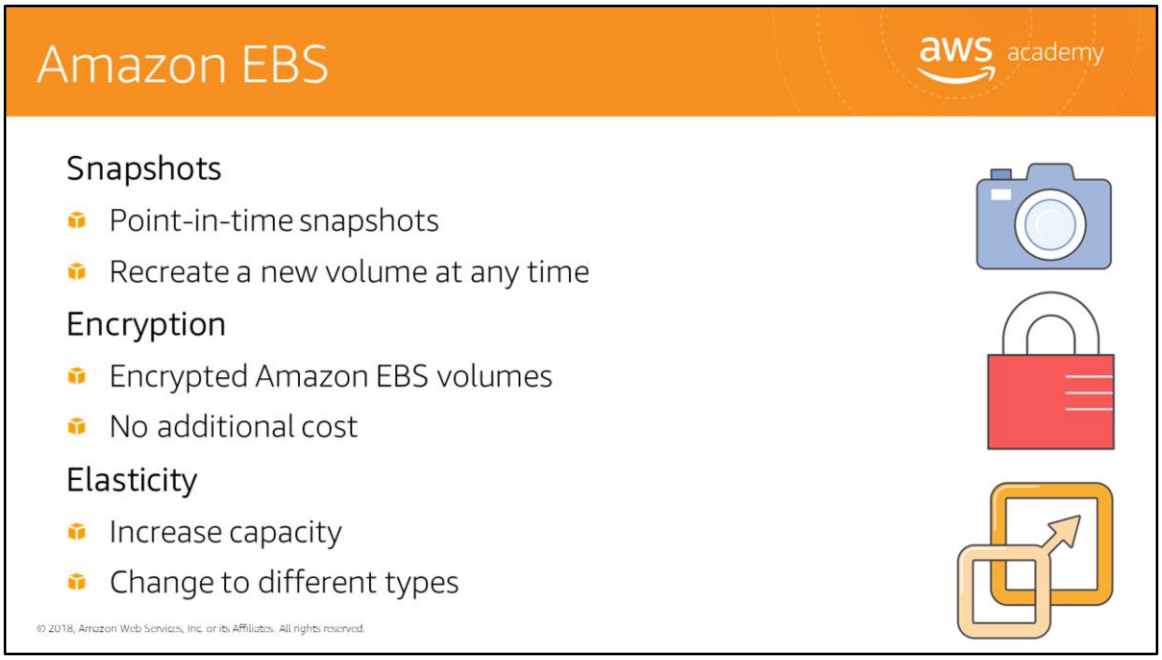


Matching the correct technology to your workload is a key best practice for reducing storage costs. Provisioned IOPS SSD-backed Amazon EBS volumes can give you the highest performance, but if your application doesn't require or won't use performance that high, one of the lower-cost options might be a better solution.

For more information, see http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EBSVolumeTypes.html.

Amazon EBS Volume Types

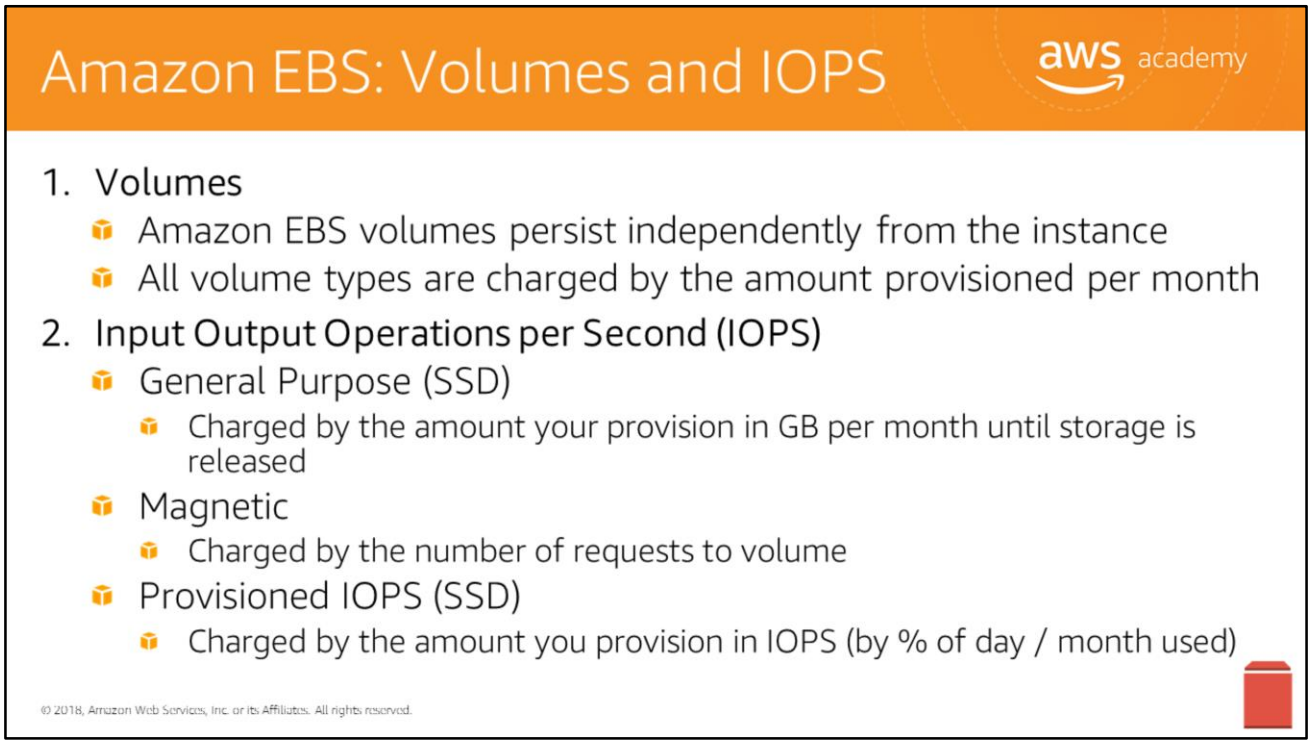
	Solid-State Drives (SSD)		Hard Disk Drives (HDD)	
	General Purpose	Provisioned IOPS	Throughput-Optimized	Cold
Use Cases	• Recommended for most workloads • System boot volumes • Virtual desktops • Low-latency interactive apps • Development and test environments	• I/O-intensive workloads • Relational DBs • NoSQL DBs	• Streaming workloads requiring consistent, fast throughput at a low price • Big data • Data warehouses • Log processing • Cannot be a boot volume	• Throughput-oriented storage for large volumes of data that is infrequently accessed • Scenarios where the lowest storage cost is important • Cannot be a boot volume



To provide an even higher level of data durability, Amazon EBS gives you the ability to create point-in-time snapshots of your volumes, and AWS allows you to recreate a new volume from a snapshot at any time. Share snapshots or even copy snapshots to different AWS Regions for even greater disaster recovery (DR) protection. You can, for example, encrypt and share your snapshots from Virginia to Tokyo.

You could also have encrypted Amazon EBS volumes at no additional cost. The encryption occurs on the Amazon EC2 side, so the data moving between the Amazon EC2 instance and the Amazon EBS volume inside AWS data centers will be encrypted in transit.

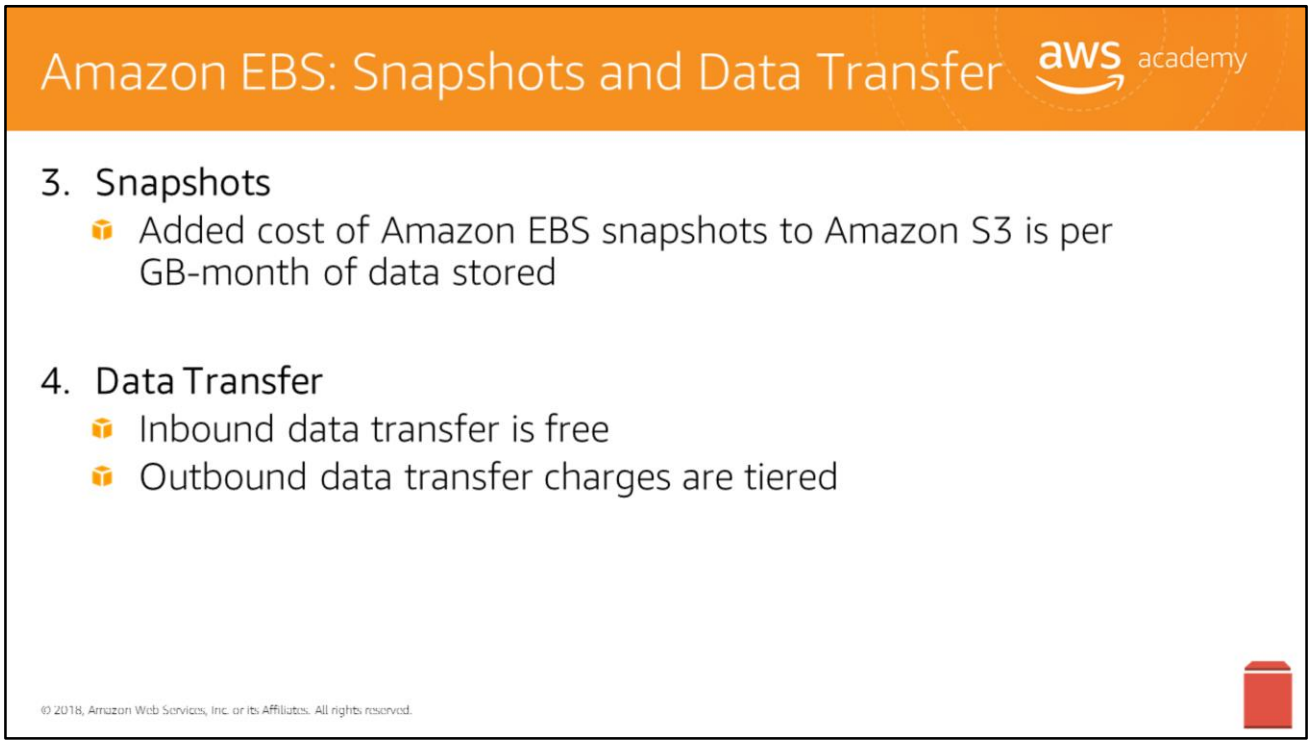
As your company grows, the amount of data stored on your Amazon EBS volumes will likely also grow. Amazon EBS volumes have the ability to increase capacity and change to different types, meaning that you can change from hard disk to SSD or increase from a 50-gigabyte volume to a 16-terabyte volume. For example, you can do this resize operation on the fly without needing to stop the instances.



When you begin to estimate the cost for Amazon EBS, you need to consider the following:

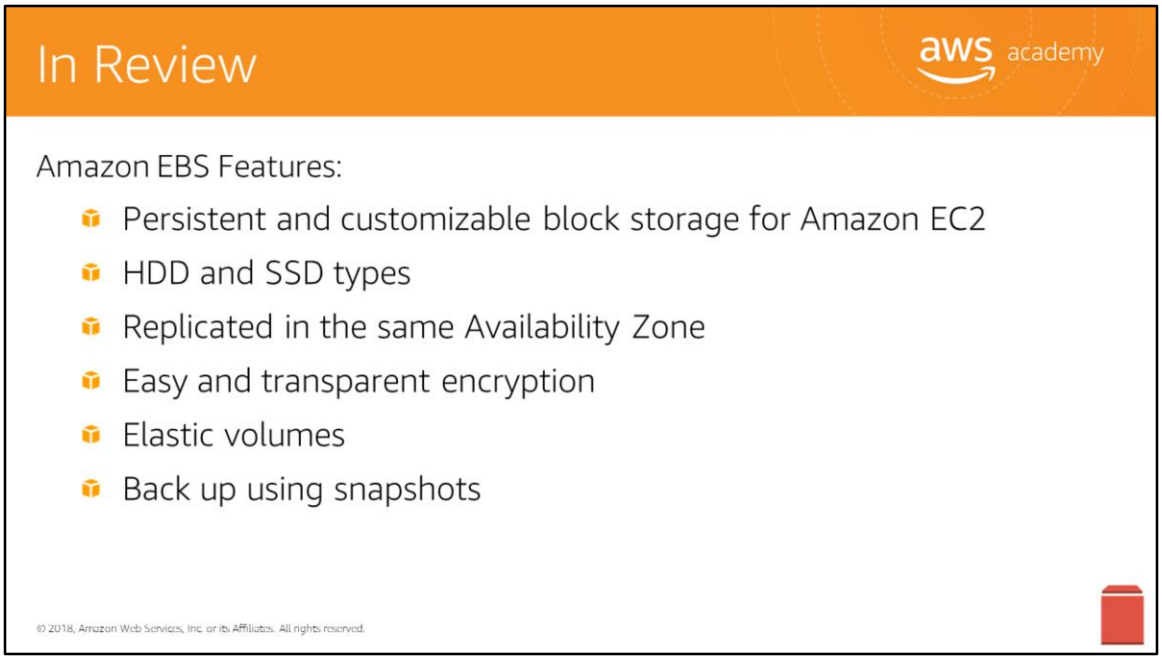
1. **Volumes** – Volume storage for all Amazon EBS volume types is charged by the amount you provision in GB per month, until you release the storage.

2. **Input Output Operations per Second (IOPS)** – I/O is included in the price of General Purpose (SSD) volumes, while for Amazon EBS Magnetic volumes, I/O is charged by the number of requests you make to your volume. With Provisioned IOPS (SSD) volumes, you are also charged by the amount you provision in IOPS (multiplied by the percentage of days you provision for the month).



3. **Snapshot** – Amazon EBS provides the ability to back up snapshots of your data to Amazon S3 for durable recovery. If you opt for Amazon EBS snapshots, the added cost is per GB-month of data stored.

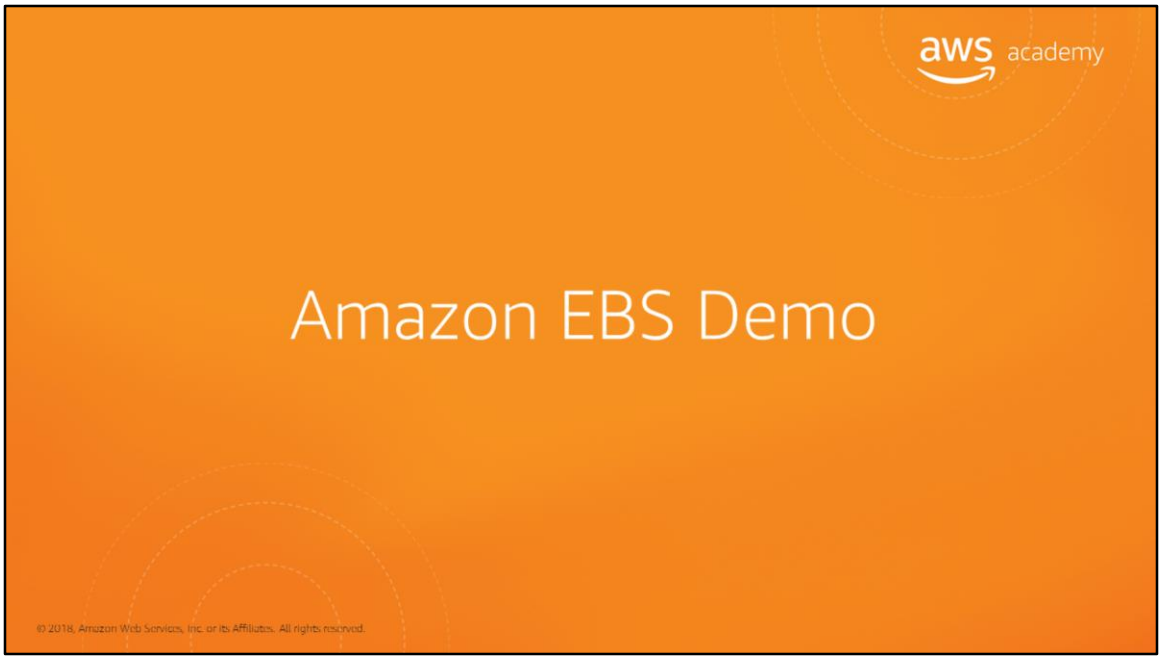
4. **Data Transfer** – Take into account the amount of data transferred out of your application. Inbound data transfer is free, and outbound data transfer charges are tiered.



Amazon EBS provides block level storage volumes for use with Amazon EC2 instances. Amazon EBS volumes are off-instance storage that persists independently from the life of an instance. They are analogous to virtual disks in the cloud. Amazon EBS provides three volume types: General Purpose (SSD), Provisioned IOPS (SSD), and Magnetic.

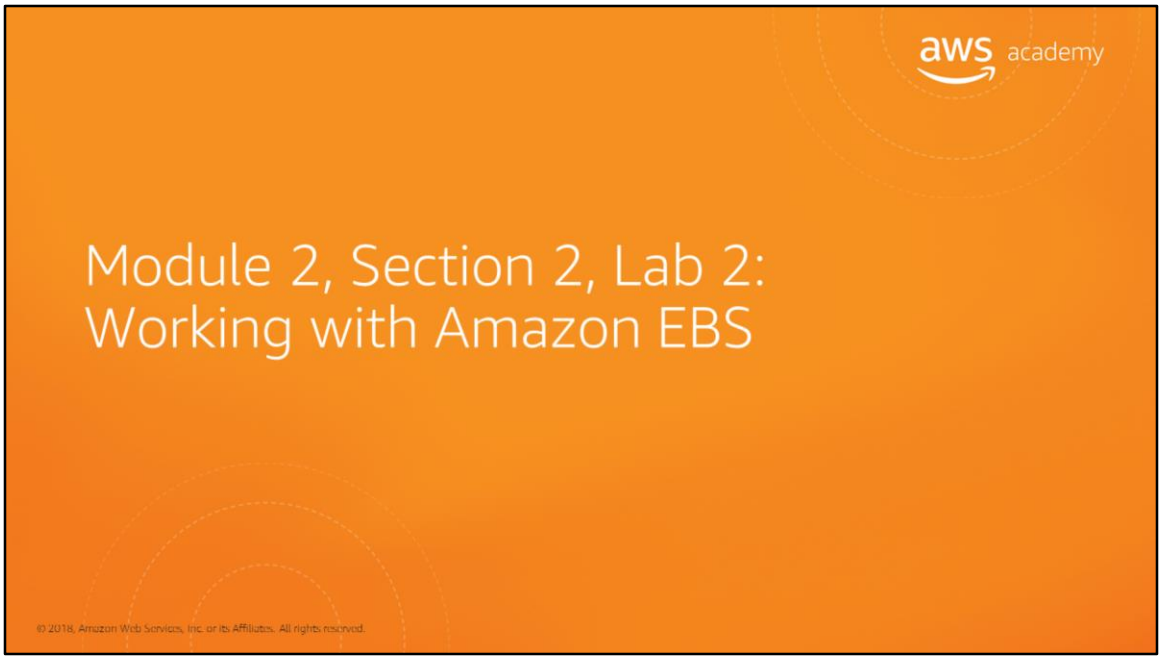
The three volume types differ in performance characteristics and cost, so you can choose the right storage performance and price for the needs of your applications.

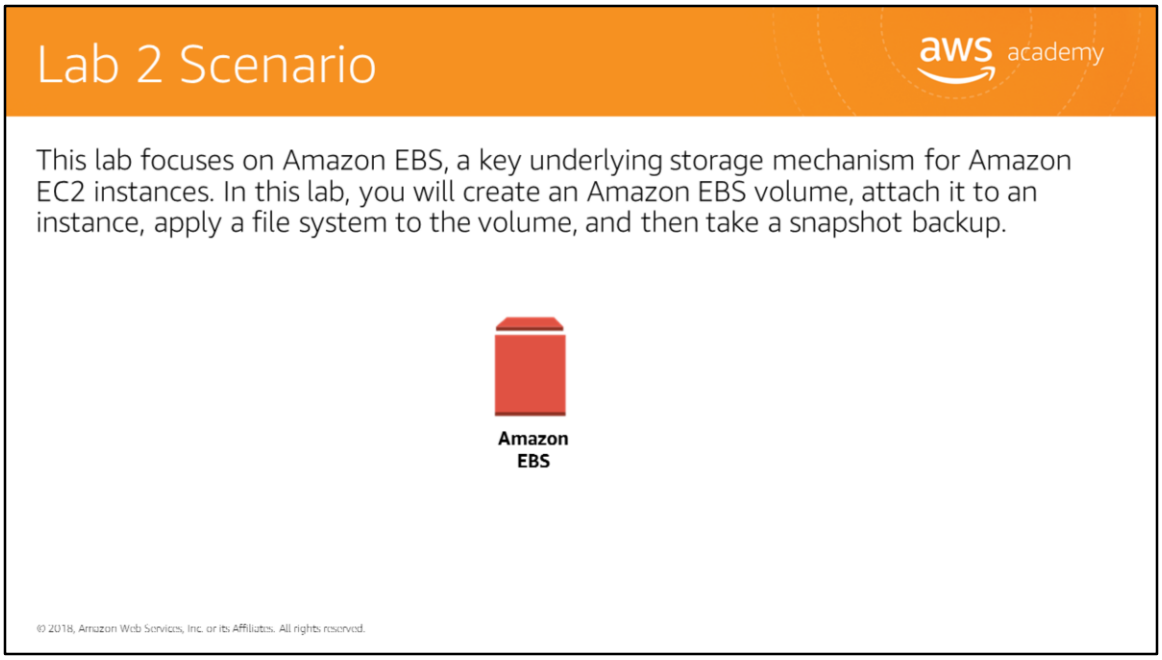
To learn more about Amazon EBS see, https://aws.amazon.com/ebs/.



Please review the Amazon EBS demonstration: M2_S3_EBS v2.0.mp4.

This video demonstration can be found in the learning management system.





Amazon EBS provides persistent block storage volumes for use with Amazon EC2 instances in the AWS cloud. Each Amazon EBS volume is automatically replicated within its Availability Zone to protect you from component failure, offering high availability and durability.

After completing this lab, you will be able to:

- Create an Amazon EBS volume
- Attach the volume to an instance
- Configure the instance to use the virtual disk
- Create an Amazon EBS snapshot
- Restore the snapshot

Duration: ~45 minutes

Lab 2: Tasks

Create a new **EBS volume.**

Attach the volume to an Amazon **EC2 instance.**

Create and configure your **file system.**

Create a **snapshot.**

Lab 2: Final Product

In this lab, you:

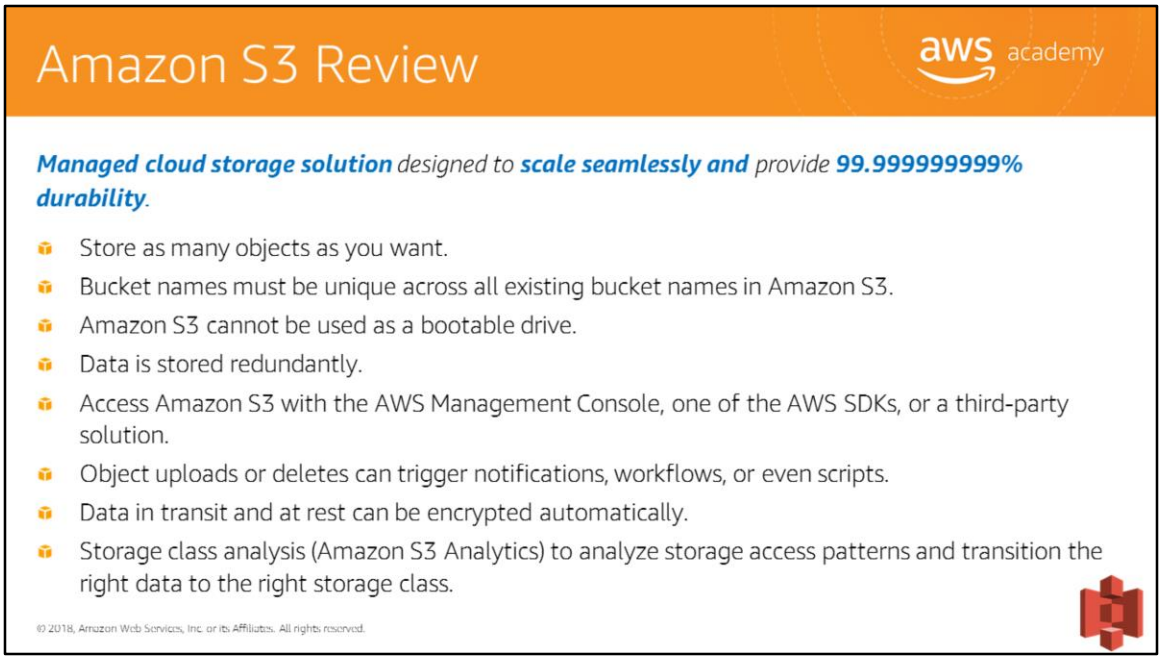
- Created an Amazon EBS volume
- Attached that volume to an instance
- Configured the instance to use the virtual disk
- Created an Amazon EBS snapshot
- Restored the snapshot

~ 45 minutes

Companies today need the ability to simply and securely collect, store, and analyze their data at a massive scale. Amazon S3 is object storage built to store and retrieve any amount of data from anywhere – web sites and mobile apps, corporate applications, and data from IoT sensors or devices.

Amazon S3 is object-level storage, which means that if you want to change a part of a file, you have to make the change and then re-upload the entire modified file. Amazon S3 stores data as objects within resources called *buckets*.

Let's take a closer look Amazon S3.



You can store as many objects as you want within a bucket, and write, read, and delete objects in your bucket. Bucket names are universal and must be unique across all existing bucket names in Amazon S3. Objects can be up to 5 terabytes in size. By default, data in Amazon S3 is stored redundantly across multiple facilities and multiple devices in each facility.

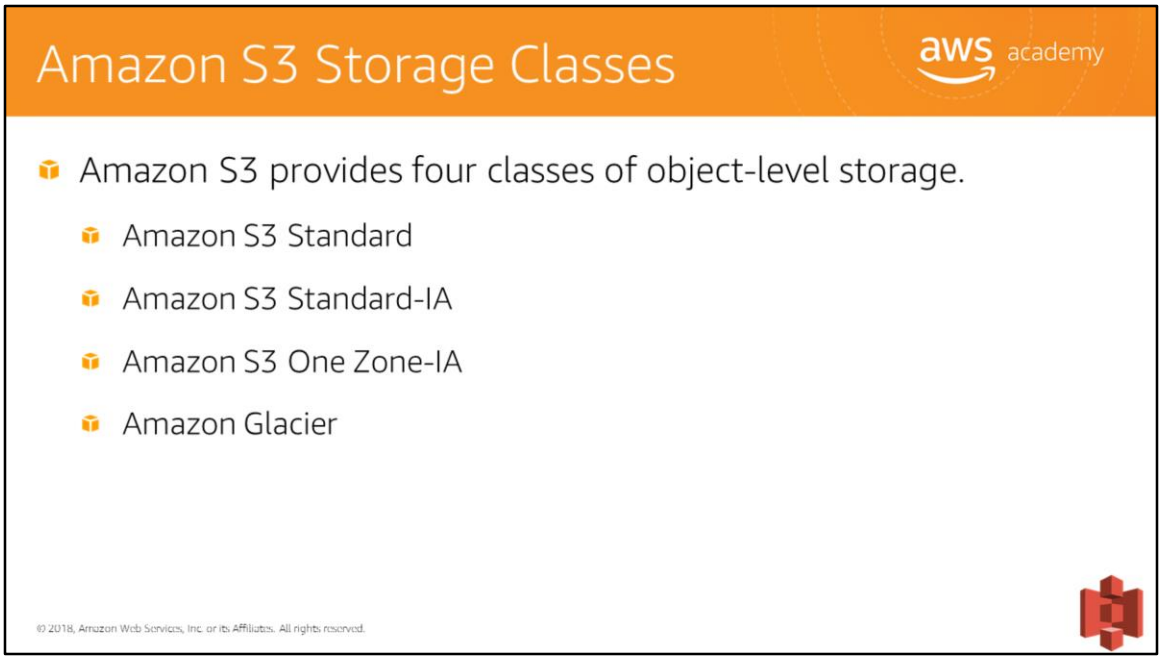
Amazon S3 is a fully managed storage service that provides a simple API for storing and retrieving data. This means that the data you store in Amazon S3 isn't associated with any particular server, and you don't have to manage any infrastructure yourself. You can put as many objects into Amazon S3 as you want. Amazon S3 holds trillions of objects and regularly peaks at millions of requests per second. Objects can be almost any data file, such as images, videos, or server logs. Since Amazon S3 supports objects as large as several terabytes in size, you could even store database snapshots as objects. Amazon S3 also provides low-latency access to the data over the internet by HTTP or HTTPS, so you can retrieve data anytime from anywhere. You can also access Amazon S3 privately through a virtual private cloud endpoint. You get fine-grained control over who can access your data using identity and access management policies, S3 bucket policies, and even per-object access control lists.

By default, none of your data is shared publicly. You can also encrypt your data in transit and choose to enable server-side encryption on your objects.

Amazon S3 can be accessed via the web-based AWS Management Console, programmatically via the API and SDKs, or with third-party solutions (which use the API/SDKs).

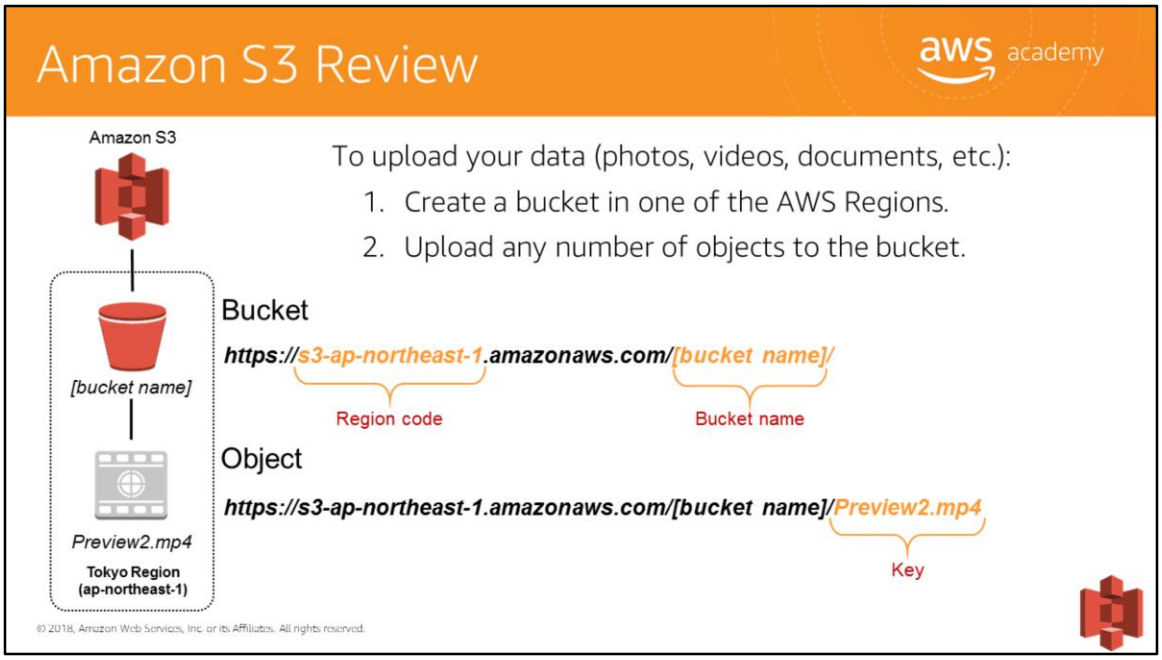
Amazon S3 includes event notifications that allow you to set up automatic notifications when certain events occur, such as an object being uploaded to or deleted from a specific bucket. Those notifications can be sent to you, or they can be used to trigger other processes, such as AWS Lambda scripts.

With storage class analysis, you can analyze storage access patterns and transition the right data to the right storage class. This new Amazon S3 Analytics feature automatically identifies the optimal lifecycle policy to transition less frequently accessed storage to Amazon S3 Standard – Infrequent Access (S3 Standard-IA). You can configure a storage class analysis policy to monitor an entire bucket, a prefix, or object tag. Once an infrequent access pattern is observed, you can easily create a new lifecycle age policy based on the results. Storage class analysis also provides daily visualizations of your storage usage in the AWS Management Console. You can export these to an S3 bucket to analyze using the business intelligence tools of your choice, such as Amazon QuickSight.



You can select from four different storage classes to store your data in Amazon S3:

- **Amazon S3 Standard**: Amazon S3 Standard offers high durability, availability, and performance object storage for frequently accessed data. Because it delivers low latency and high throughput, Amazon S3 Standard is perfect for a wide variety of use cases including cloud applications, dynamic websites, content distribution, mobile and gaming applications, and Big Data analytics.
- **Amazon S3 Standard-IA**: Amazon S3 Standard-Infrequent Access (Amazon S3 Standard-IA) is an Amazon S3 storage class for data that is accessed less frequently, but requires rapid access when needed. Amazon S3 Standard-IA offers the high durability, high throughput, and low latency of Amazon S3 Standard, with a low per GB storage price and per GB retrieval fee. This combination of low cost and high performance make Amazon S3 Standard-IA ideal for long-term storage, backups, and as a data store for disaster recovery.
- **Amazon S3 One Zone-IA**: Amazon S3 One Zone-Infrequent Access (Amazon S3 One Zone-IA) is an Amazon S3 storage class for data that is accessed less frequently, but requires rapid access when needed. Unlike other Amazon object storage classes, which store data in a minimum of three Availability Zones (AZs), Amazon S3 One Zone-IA stores data in a single AZ.
- **Amazon Glacier**: Amazon Glacier is a secure, durable, and extremely low-cost storage service for data archiving. You can reliably store any amount of data at costs that are competitive with or cheaper than on-premises solutions.



To get the most out of Amazon S3, you need to understand a few simple concepts. First, Amazon S3 stores data inside *buckets*. Buckets are essentially the prefix for a set of files, and as such must be uniquely named across all of Amazon S3. Buckets are logical containers for objects. You can have one or more buckets in your account. For each bucket, you can control access, in other words, who can create, delete and list objects in the bucket. You can also view access logs for the bucket, and its objects, and choose the geographical region where Amazon S3 will store the bucket and its contents.

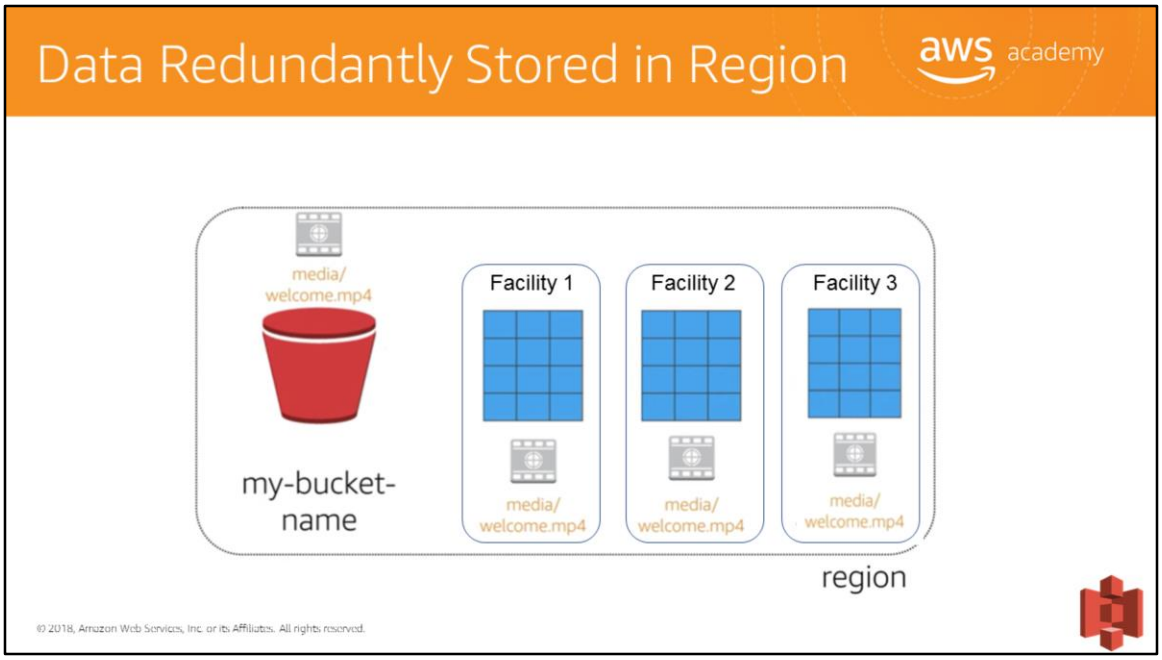
In the example, we've used Amazon S3 to create a bucket in the Tokyo region, which is identified within AWS formally by its region code: "ap-northeast-1").

The URL for a bucket is structured as displayed here, with the region code first, followed by amazonaws.com, followed by the bucket name.

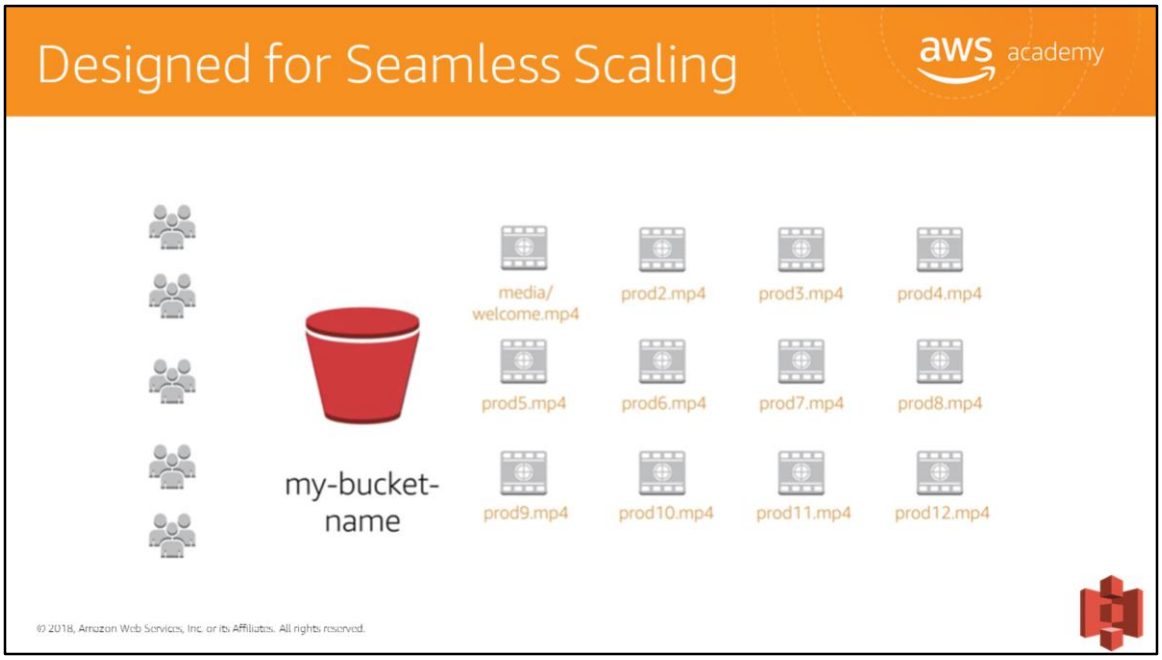
Amazon S3 refers to files as *objects*. Once you have a bucket, you can store any number of objects inside of it. An object is composed of data, and any metadata that describes that file. To store an object in Amazon S3, you upload the file you want to store into a bucket.

When you upload a file, you can set permission on the data as well as any metadata.
In this example, we're storing the object "Preview2.mp4" inside of our bucket.

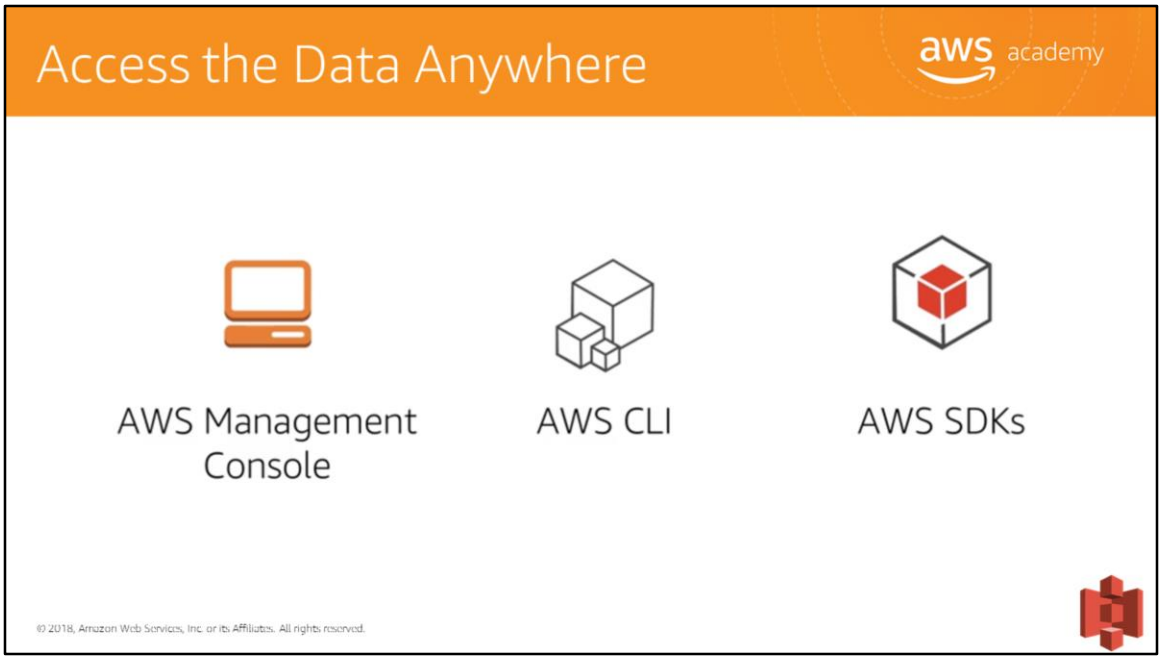
The URL for the file includes the object name at the end.



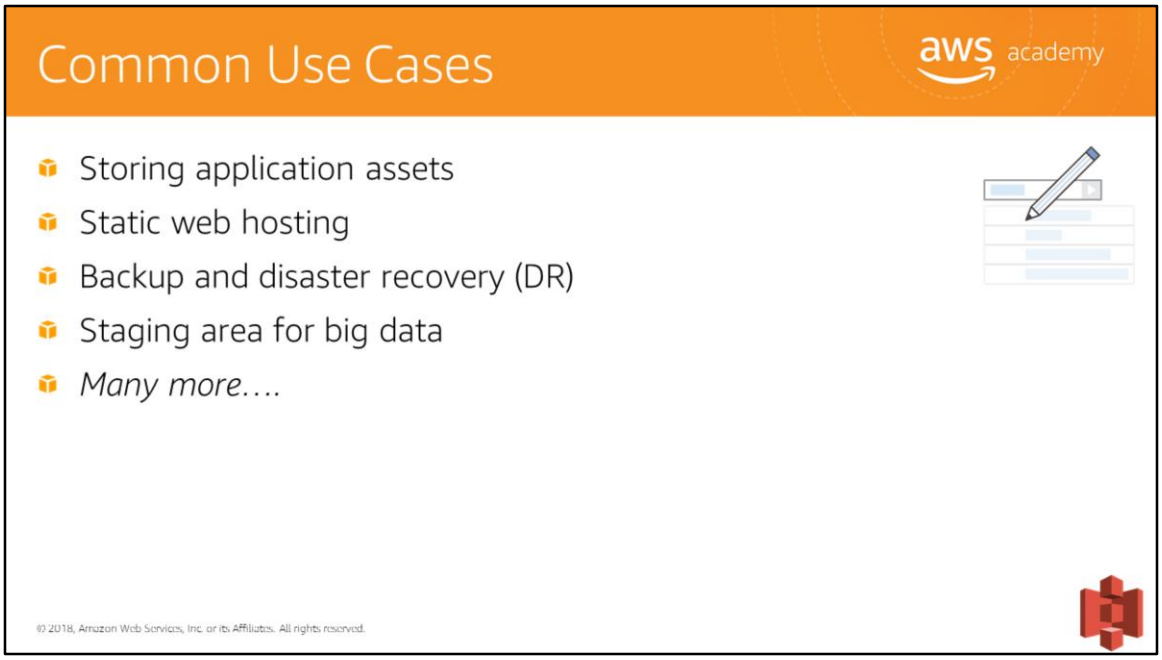
When you create a bucket in Amazon S3, it's associated with a particular AWS Region. Whenever you store data in the bucket, it is redundantly stored across multiple AWS facilities within your selected region. Amazon S3 is designed to durably store your data, even in the case of concurrent data loss in two AWS facilities.



Amazon S3 will automatically manage the storage behind your bucket even as your data grows. This allows you to get started immediately and to have your data storage grow with your application needs. Amazon S3 will also scale to handle a high volume of requests. You don't have to provision the storage or throughput, and you'll only be billed for what you use.

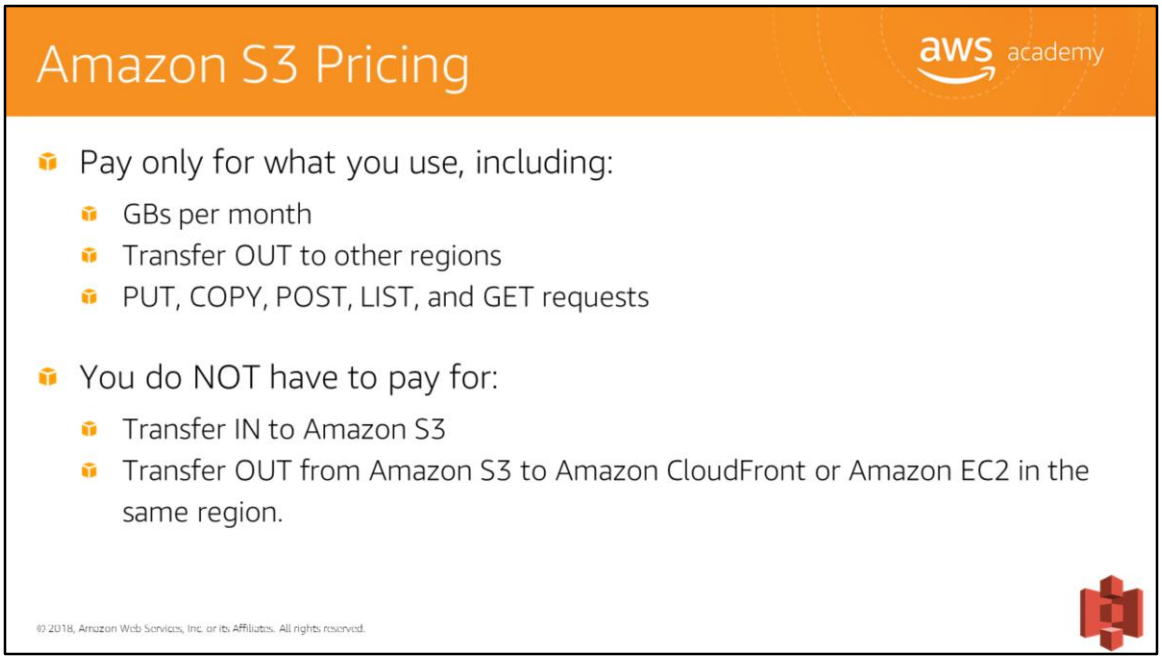


You can access Amazon S3 via the console, AWS CLI, or AWS SDK. Additionally, you can also access the data in your bucket directly via the rest endpoints. These support HTTP or HTTPS access. To support this type of URL-based access, S3 bucket names must be globally unique and DNS-compliant. Also, object keys should be using characters that are safe for URLs.

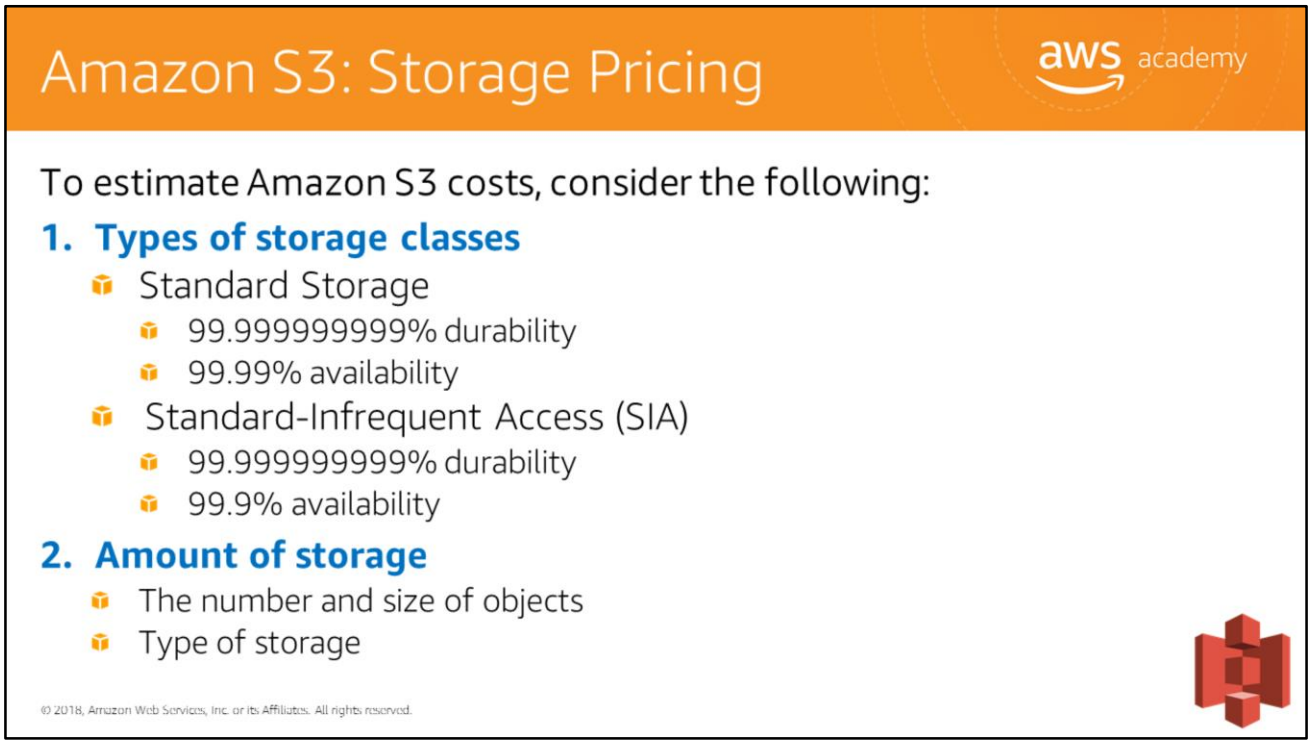


This flexibility to store a virtually unlimited amount of data and access that data from anywhere makes Amazon S3 suitable for a wide range of scenarios. Let's look at some use cases for Amazon S3:

- As a location for any application data, Amazon S3 buckets provide that shared location for storing objects that any instances of your application can access, including applications on Amazon EC2 or even traditional servers. This can be useful for user-generated media files, server logs, or other files your application needs to store in a common location. Also, because the content can be fetched directly over the web, you can offload serving of that content from your application and allow clients to directly fetch the data themselves from Amazon S3.
- For static web hosting, Amazon S3 buckets can serve up the static contents of your website, including HTML, CSS, JavaScript, and other files.
- The high durability of Amazon S3 makes it a good candidate to store backups of your data. For even greater availability and disaster recovery capability, Amazon S3 can even be configured to support cross-region replication such that data put into an Amazon S3 bucket in one region can be automatically replicated to another Amazon S3 region. The scalable storage and performance of Amazon S3 make it a great candidate for staging or long-term storage of data you plan to analyze using a variety of big data tools. Given how simple it is to store and access data with Amazon S3, you'll find yourself using it frequently with AWS services and for other parts of your application.

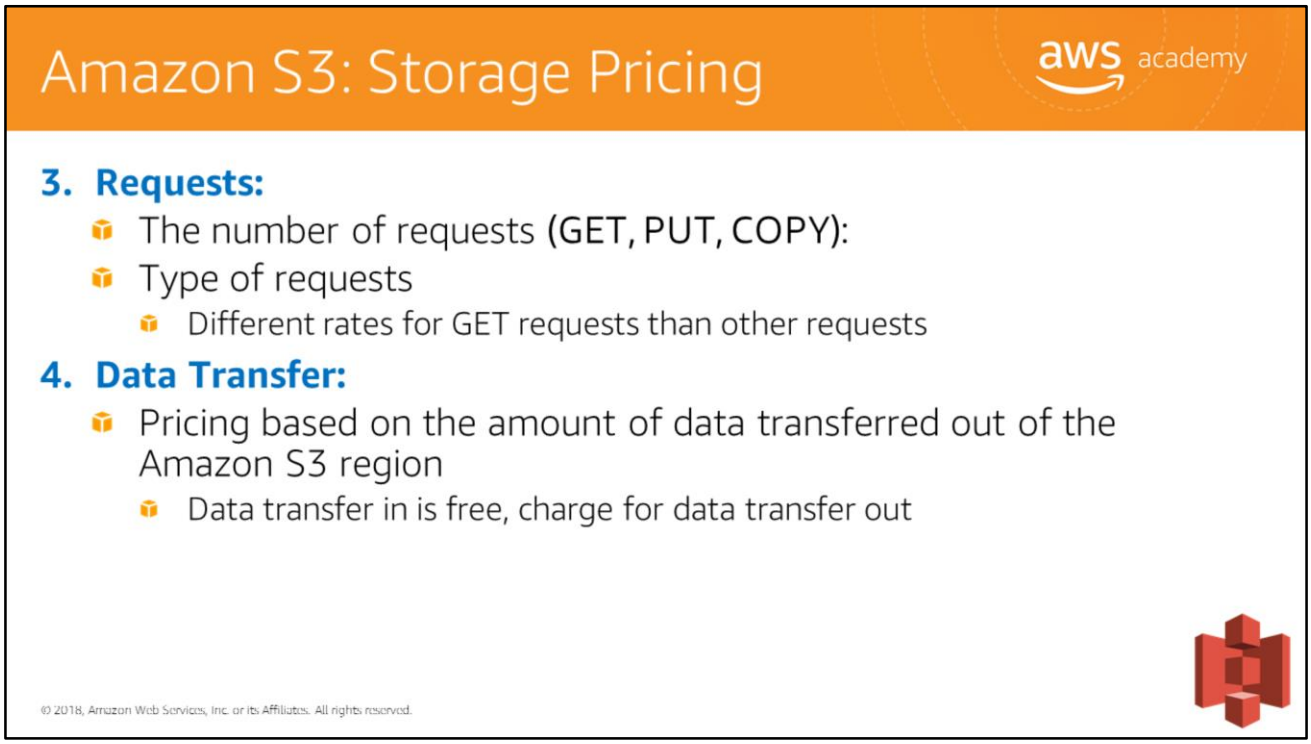


Specific costs may vary depending on region and the specific requests made.
As a general rule, you only pay for transfers that cross the boundary of your region, which means you do not pay for transfers to Amazon CloudFront's edge locations within that same region.



When you begin to estimate the costs of Amazon S3, you need to consider the following:

1. **Storage Class:**
   - Standard Storage is designed to provide 99.999999999% durability and 99.99% availability.
   - Standard – Infrequent Access (SIA) is a storage option within Amazon S3 that you can use to reduce your costs by storing less frequently accessed data at slightly lower levels of redundancy than Amazon S3's standard storage. Standard – Infrequent Access is designed to provide the same 99.999999999% durability as Amazon S3 with 99.9% availability in a given year. It's important to note that each class has different rates.
2. **Storage** – The number and size of objects stored in your Amazon S3 buckets as well as type of storage.

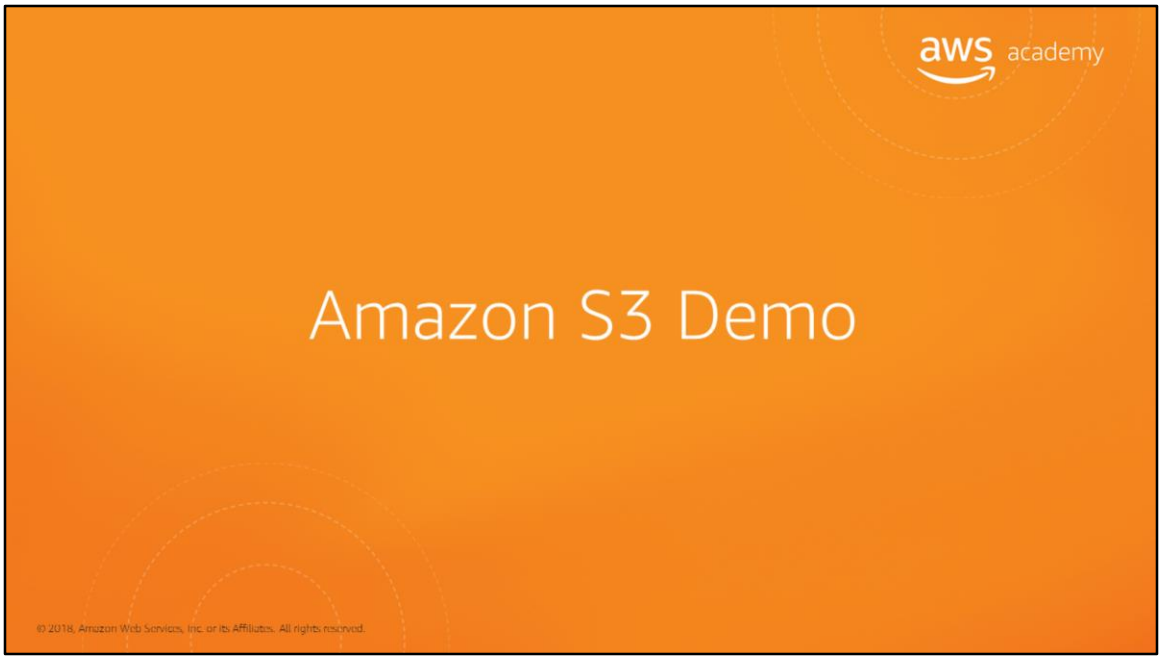


3. **Requests** – The number and type of requests. GET requests incur charges at different rates than other requests, such as PUT and COPY requests.
    - Get: Retrieves an object from Amazon S3. You must have READ access to use this operation.
    - Put: Adds an object to a bucket. You must have WRITE permissions on a bucket to add an object to it.
    - Copy: Creates a copy of an object that is already stored in Amazon S3. A PUT copy operation is the same as performing a GET and then a PUT.

4. **Data Transfer** – The amount of data transferred out of the Amazon S3 region. Remember that data transfer in is free, but there is a charge for data transfer out.

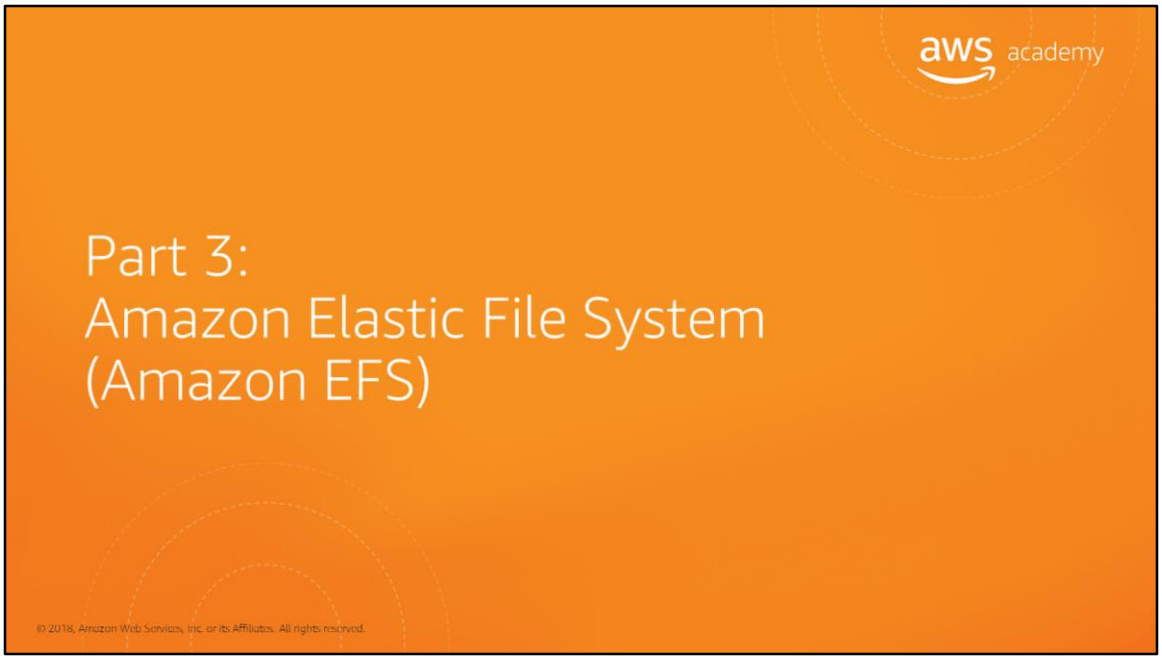
We've covered an introduction to Amazon S3 including key features and some common use cases.

For more information about Amazon S3, see https://aws.amazon.com/s3/.

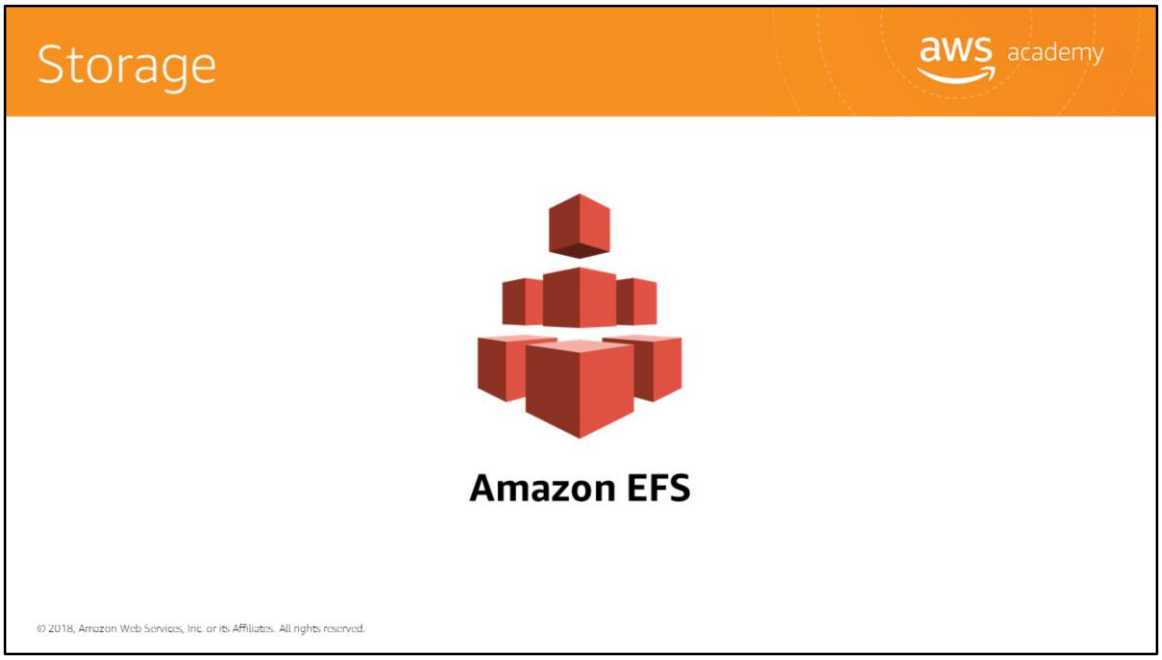


Please review the Amazon S3 demonstration: M2_S3_S3 v2.0.mp4.

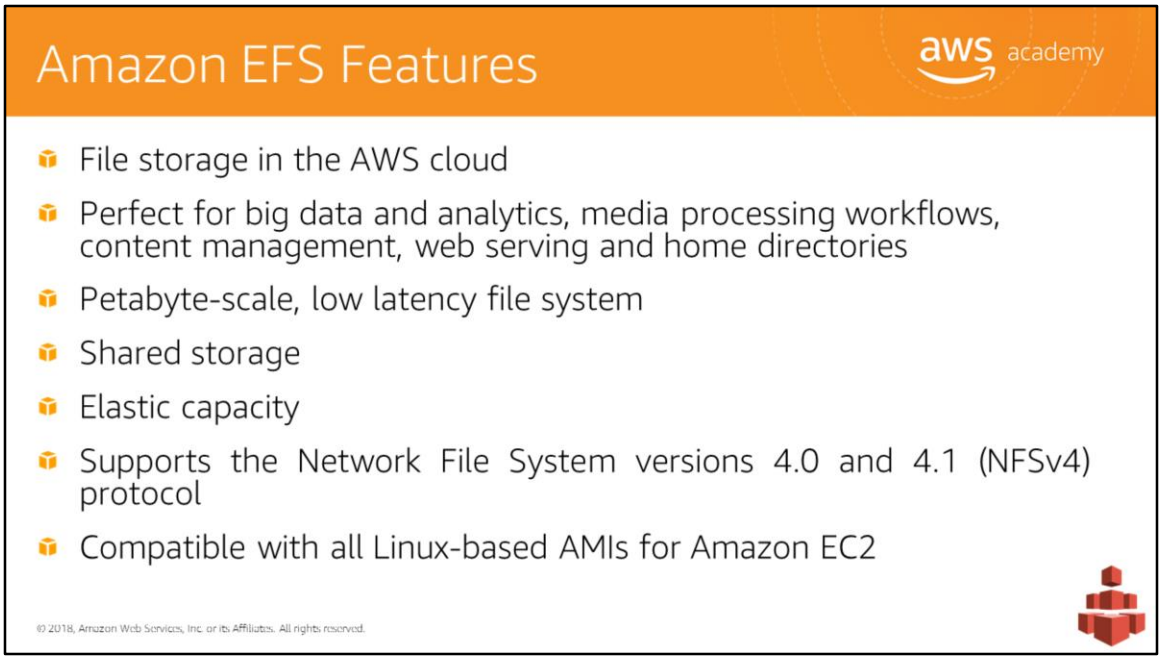
This video demonstration can be found in the learning management system.



Amazon Elastic File System (Amazon EFS) provides simple, scalable file storage for use with Amazon EC2 instances in the AWS Cloud. Amazon EFS is easy to use and offers a simple interface that allows you to create and configure file systems quickly and easily.

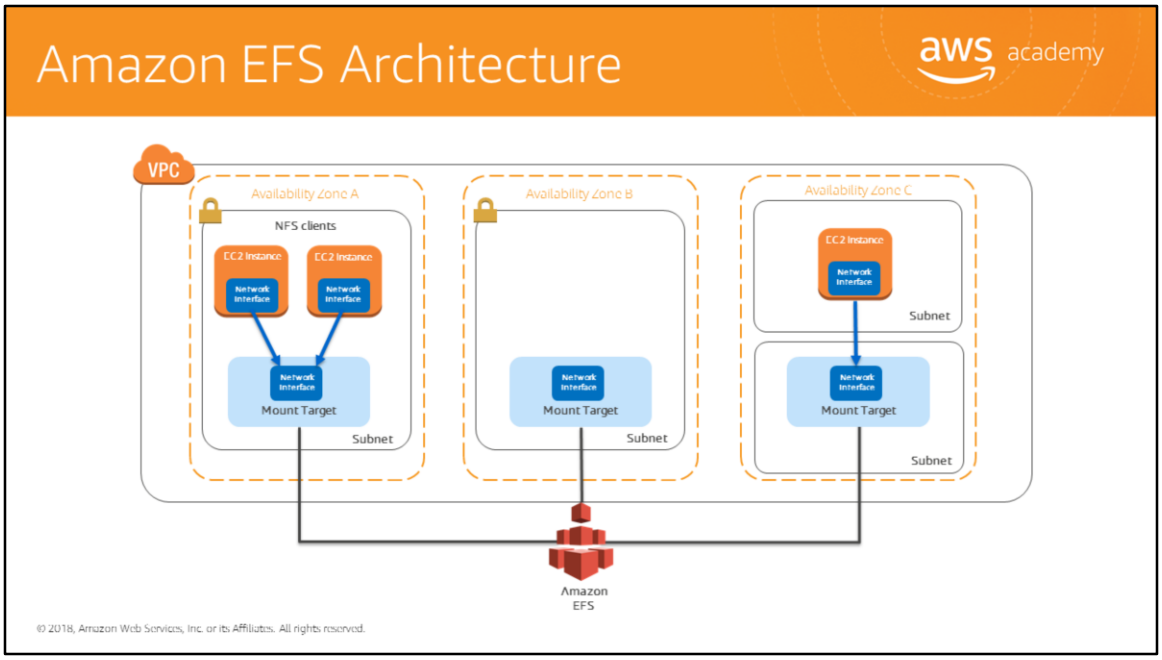


Amazon EFS provides simple, scalable, elastic file storage for use with AWS services and on-premises resources. It is easy to use and offers a simple interface that allows you to create and configure file systems quickly and easily. Amazon EFS is built to elastically scale on demand without disrupting applications, growing and shrinking automatically as you add and remove files, so your applications have the storage they need, when they need it.



Amazon EFS is a fully managed service that makes it easy to set up and scale file storage in the AWS cloud. It is the easiest way to build a file system for big data and analytics, media processing workflows, content management, web serving and home directories. You can create file systems that are accessible to Amazon EC2 instances via a file system interface (using standard operating system file I/O APIs) and that support full file system access semantics (such as strong consistency and file locking).

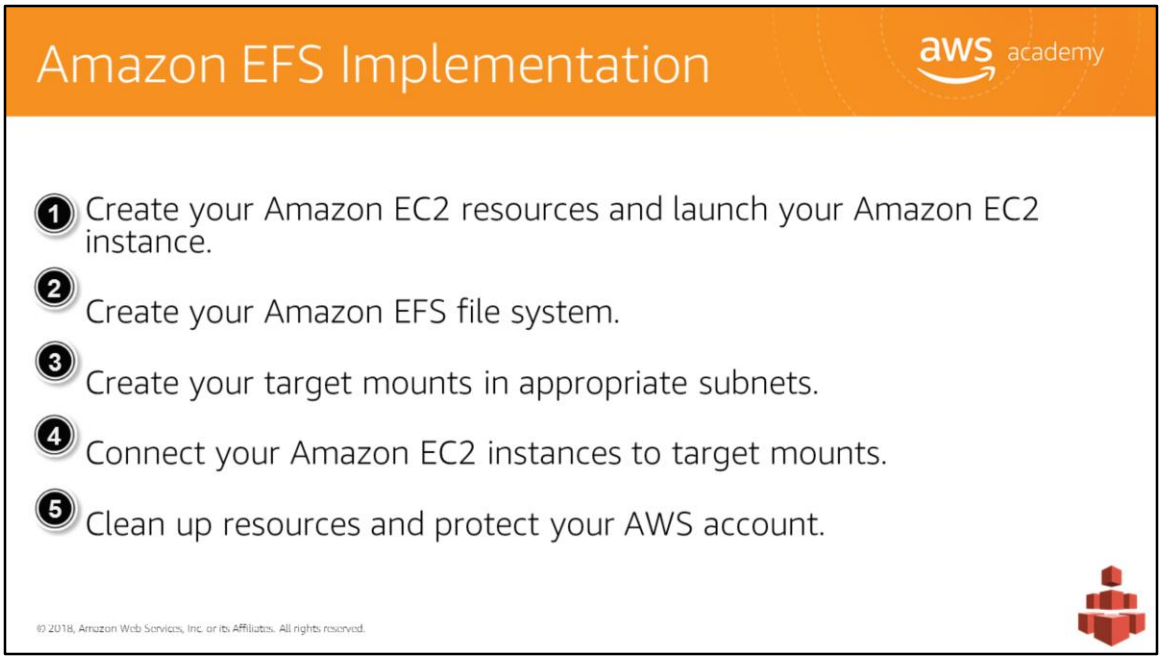
Amazon EFS file systems can automatically scale from gigabytes to petabytes of data without needing to provision storage. Thousands of Amazon EC2 instances can access an Amazon EFS file system at the same time, and Amazon EFS provides consistent performance to each Amazon EC2 instance. Amazon EFS is designed to be highly durable and highly available. With Amazon EFS, there is no minimum fee or setup costs, and you pay only for the storage you use.



Amazon EFS provides file storage in the cloud. With Amazon EFS, you can create a file system, mount the file system on an Amazon EC2 instance, and then read and write data from to and from your file system. You can mount an Amazon EFS file system in your VPC, through the Network File System versions 4.0 and 4.1 (NFSv4) protocol.

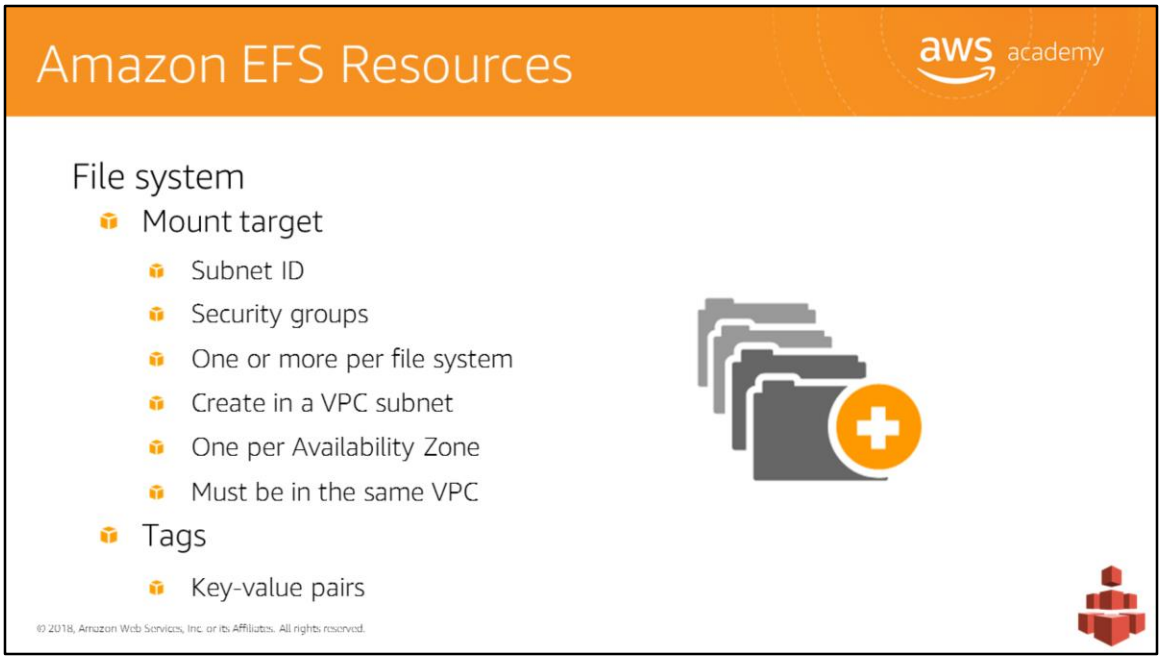
You can access your Amazon EFS file system concurrently from Amazon EC2 instances in your Amazon VPC, so applications that scale beyond a single connection can access a file system. Amazon EC2 instances running in multiple Availability Zones within the same AWS Region can access the file system, so that many users can access and share a common data source.

In this illustration, the VPC has three Availability Zones, and each has one mount target created in it. We recommend that you access the file system from a mount target within the same Availability Zone. Note that one of the Availability Zones has two subnets. However, a mount target is created in only one of the subnets.



There are five steps you need to perform to create and use your first Amazon EFS file system, mount it on an Amazon EC2 instance in your VPC, and test the end-to-end setup.

1. Create your Amazon EC2 resources and launch your instance. (Before you can launch and connect to an Amazon EC2 instance, you need to create a key pair, unless you already have one.)
2. Create your Amazon EFS file system.
3. In the appropriate subnet, create your target mounts.
4. Next, connect to your Amazon EC2 instance and mount the Amazon EFS file system.
5. Finally, clean up your resources and protect your AWS account.

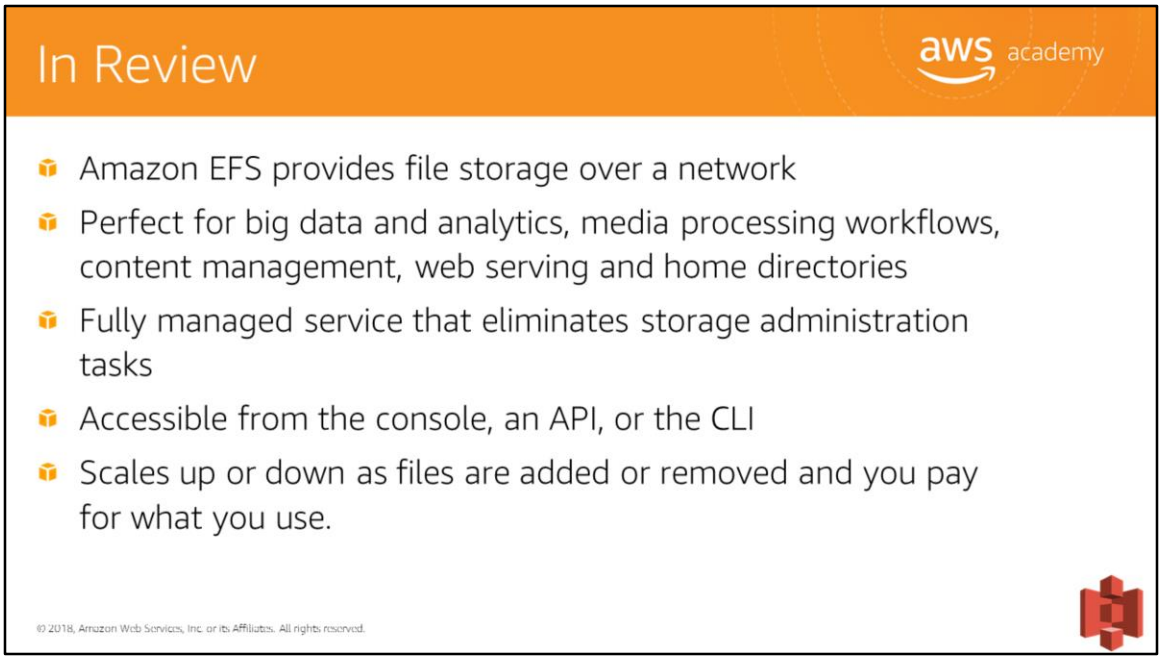


In Amazon EFS, a file system is the primary resource. Each file system has properties such as ID, creation token, creation time, file system size in bytes, number of mount targets created for the file system, and the file system state.

Amazon EFS also supports other resources to configure the primary resource. These include mount targets and tags.
Mount target: To access your file system, you must create mount targets in your VPC. Each mount target has the following properties:
- The mount target ID
- The subnet ID in which it is created
- The file system ID for which it is created
- An IP address at which the file system may be mounted
- The mount target state.

You can use the IP address or the DNS name in your mount command. Each file system has a DNS name of the following form.
Tags: To help organize your file systems, you can assign your own metadata to each of the file systems you create. Each tag is a key-value pair.

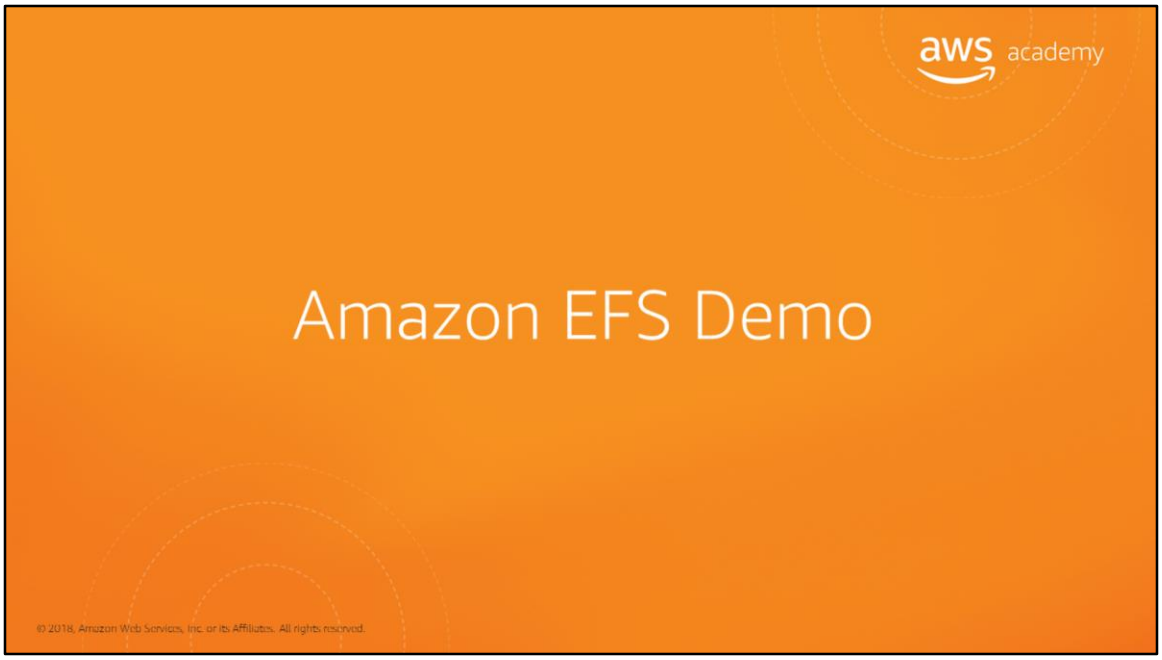
Think of mount targets and tags as subresources that don't exist without being associated with a file system.



We've covered an introduction to Amazon EFS, including key features and key resources. It provides file storage in the cloud that is perfect for big data and analytics, media processing workflows, content management, web serving and home directories. Amazon EFS scales up or down as files are added or removed and you pay for only what you are using.

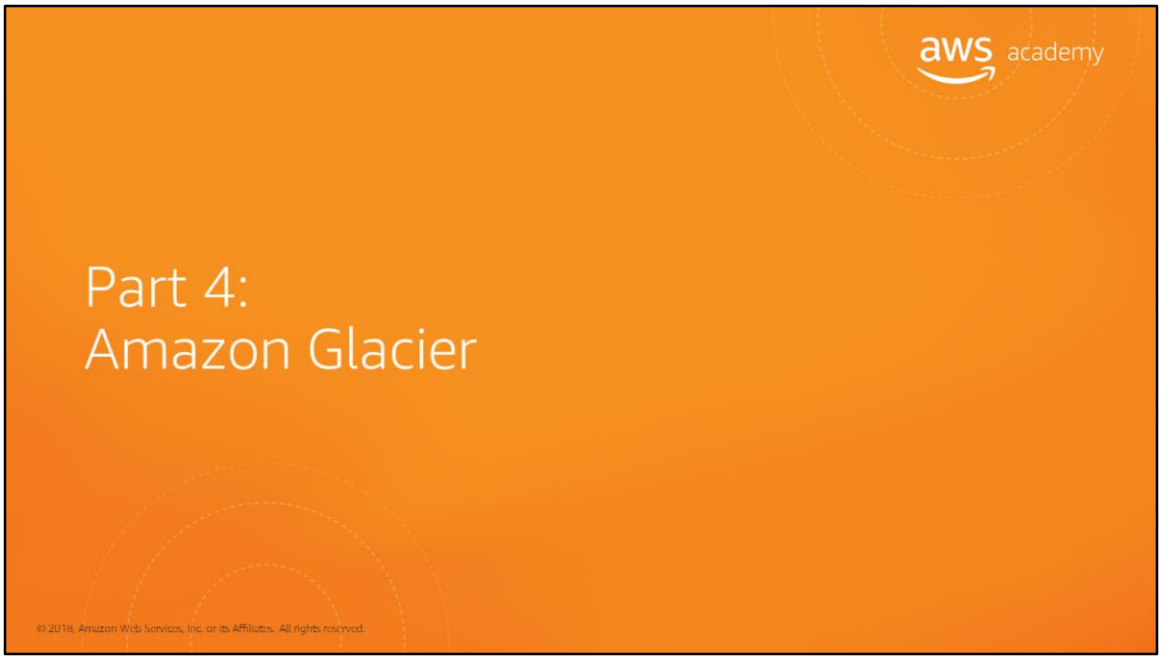
Amazon EFS is a fully managed service that is accessible from the console, an API, or the CLI.

For more information about Amazon S3, see https://aws.amazon.com/efs/.



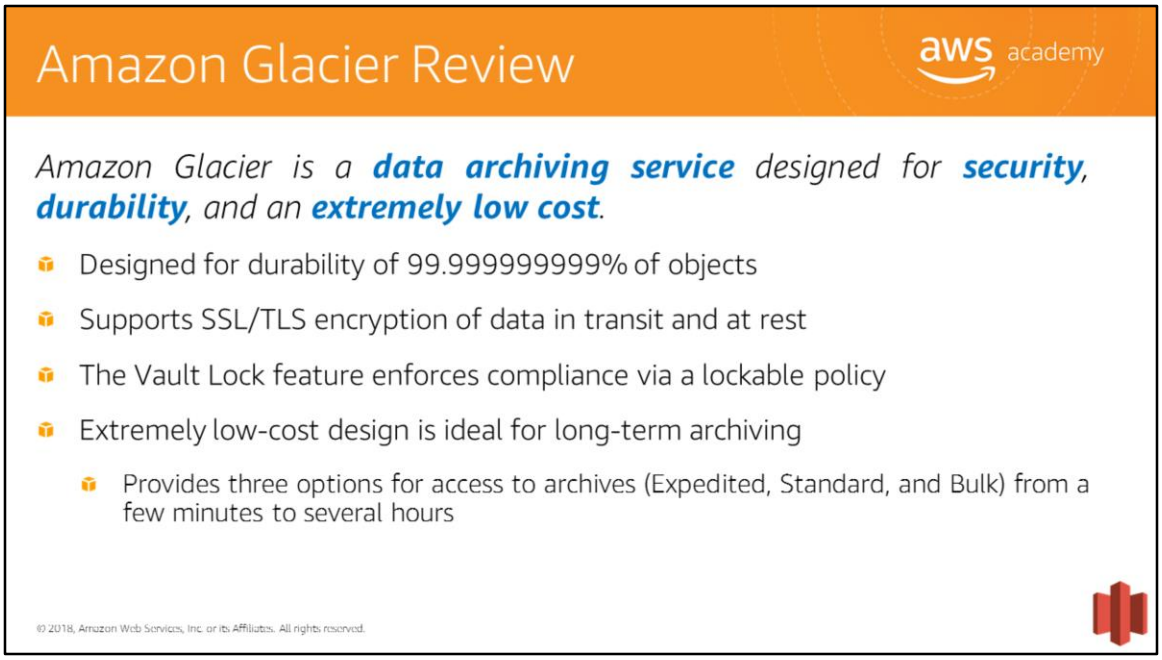
Please review the Amazon EFS demonstration: M2_S3_EFS v2.0.mp4.

This video demonstration can be found in the learning management system.



Amazon Glacier is a secure, durable, and extremely low-cost cloud storage service for data archiving and long-term backup.

Let’s take a closer look at Amazon Glacier.



Amazon Glacier's data archiving means that although you can store your data at an extremely low cost (even in comparison to Amazon S3), you cannot retrieve your data immediately when you want it. Data stored in Amazon Glacier takes several hours to retrieve, which is why it's ideal for archiving.

There are three key Amazon Glacier terms that you should be familiar with:

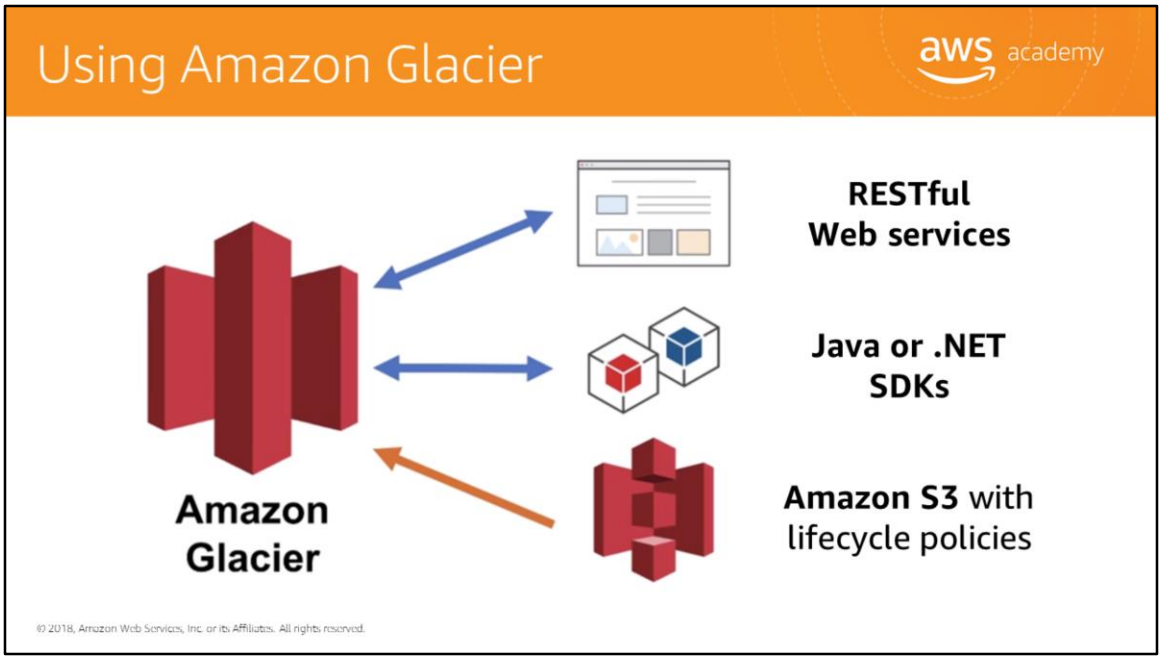
- **Archive**: Any object such as a photo, video, file, or document that you store in Amazon Glacier. It is the base unit of storage in Amazon Glacier. Each archive has its own unique ID and can also have a description.
- **Vault**: A container for storing archives. When you create a vault, you specify the vault name and the region in which you would like the vault located.
- **Vault Access Policy**: Determine who can and cannot access the data stored in the vault as well as what operations users can and cannot perform. One vault access permissions policy can be created for each vault to manage access permissions for that vault. You can also use a vault lock policy to make sure a vault cannot be altered. Each vault can have one vault access policy and one vault lock policy attached to it.

There are three options for retrieving data with varying access times and cost: Expedited, Standard, and Bulk retrievals, as follows:

- Expedited retrievals are typically made available within 1 – 5 minutes (highest cost).

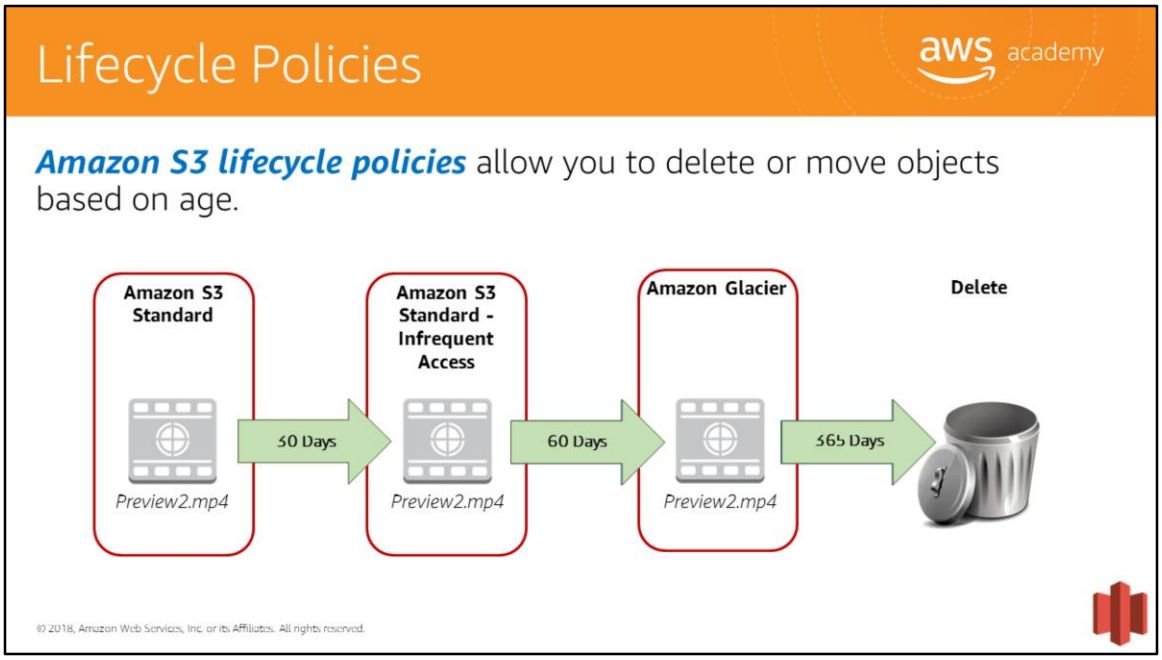
- Standard retrievals typically complete within 3 – 5 hours (less than expedited, more than bulk).
- Bulk retrievals typically complete within 5 – 12 hours (lowest cost).

Think of it as being like choosing the cost to ship a package.



To store and access data in Amazon Glacier, you can use the AWS Management Console; however only a few operations, such as creating and deleting vaults and creating and managing archive policies, are available in the console. Almost all other operations require that you use either the Amazon Glacier REST API, or AWS Java or .NET SDKs to interact with Amazon Glacier via the CLI.

You can also archive data into Amazon Glacier using lifecycle policies. Let's take a closer look at what that means.



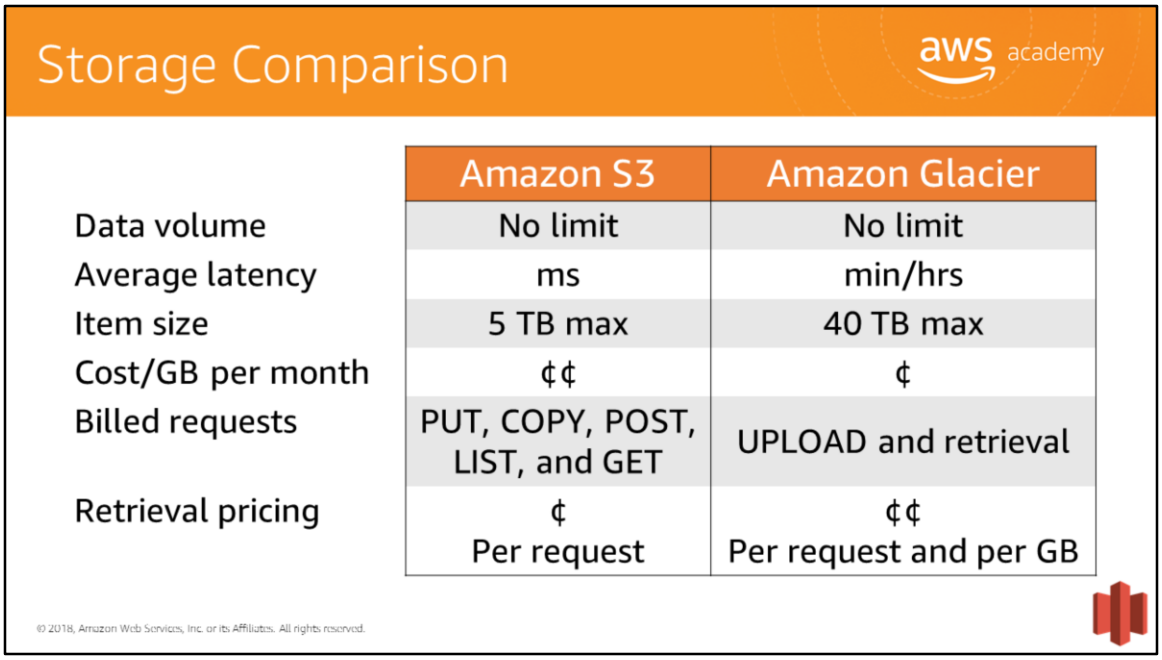
You should automate the lifecycle of your data stored in Amazon S3. Using lifecycle policies, you can have data cycled at regular intervals between different Amazon S3 storage types. This reduces your overall cost, because you are paying less for data as it becomes less important with time.

In addition to being able to set lifecycle rules per object, you can also set lifecycle rules per bucket.

Let's take a look at an example of a lifecycle policy that moves data as it ages from Amazon S3 Standard to Amazon S3 Standard – Infrequent Access and, finally, into Amazon Glacier before it is deleted. Let's say that the user uploads a video to your application and your application generates a thumbnail preview of the video. This video preview is stored to Amazon S3 Standard, because it is likely that the user will want to access it right away.

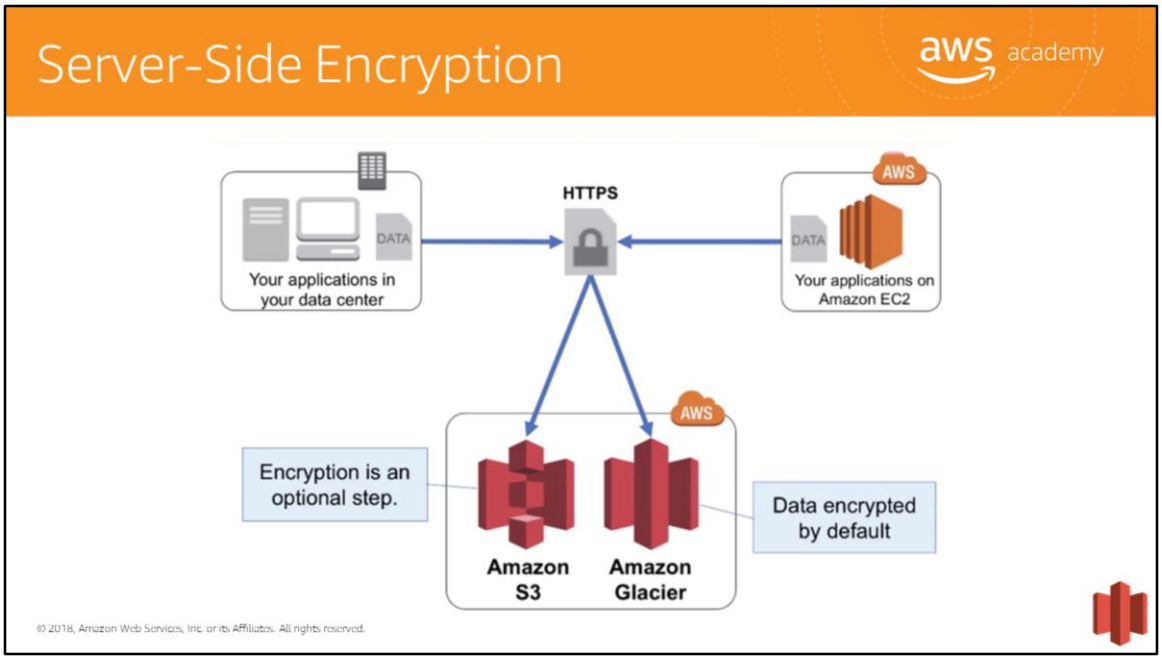
Your usage data indicates that most thumbnail previews are not accessed after 30 days. So, your lifecycle policy will take this previews and move them to Amazon S3 infrequent access after 30 days. Once another 30 days have lapsed, it is highly unlikely that it the preview will be accessed again, so it is moved to Amazon Glacier where it remains for 1 year. After one year, the preview is deleted. The important thing to note is that the lifecycle policy manages all of this movement automatically.

For more information, see http://docs.aws.amazon.com/AmazonS3/latest/dev/object-lifecycle-mgmt.html.



While Amazon S3 and Amazon Glacier are both object storage solutions that allow you to store an unlimited amount of data, there are some critical differences between them that are outlined in this chart.

1. Be careful when deciding which storage solution is correct for your needs. These are two very different services for storage needs. Amazon S3 is designed for frequent, low-latency access to your data, while Amazon Glacier is designed for low-cost, long-term storage of infrequently accessed data.
2. The maximum item size in Amazon S3 is 5 TB, whereas Amazon Glacier can store items up to 40 TB in size.
3. Because Amazon S3 gives you faster access to your data, the storage cost per gigabyte is higher than it is with Amazon Glacier.
4. While both services have per request charges, Amazon S3 charges for PUT, COPY, POST, LIST, GET while Amazon Glacier charges for UPLOAD and retrieval.
5. Because Amazon Glacier was designed for less frequent access to data, it costs more for each retrieval request than Amazon S3. Both the cost per retrieval and the cost per GB are higher for Amazon Glacier.



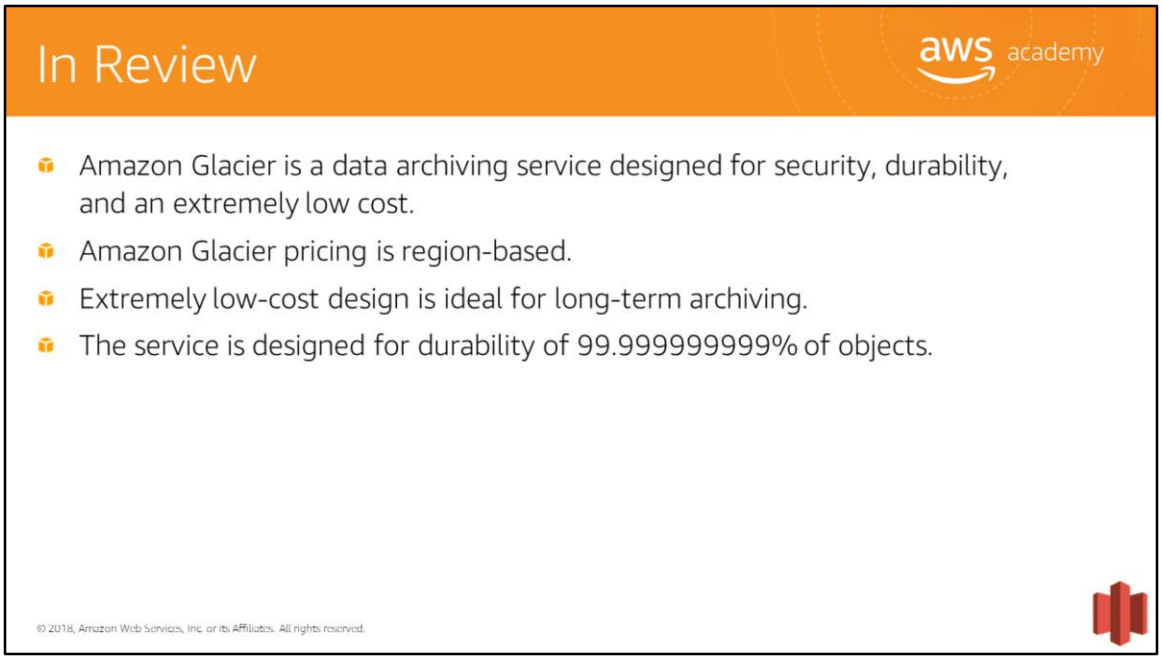
Another important difference between Amazon S3 and Amazon Glacier is how data is encrypted. Server-side encryption is about protecting data at rest. With both solutions, you can securely transfer your data over HTTPS. Any data archived in Amazon Glacier is encrypted by default. With Amazon S3, your application must initiate server-side encryption. There are several ways to accomplish this:

- Server-side encryption with Amazon S3-managed encryption keys (**SSE-S3**) employs strong multi-factor encryption. Amazon S3 encrypts each object with a unique key. As an additional safeguard, it encrypts the key itself with a master key that it regularly rotates. Amazon S3 server-side encryption uses one of the strongest block ciphers available, 256-bit Advanced Encryption Standard (AES-256), to encrypt your data.
- AWS Key Management Service (**AWS KMS**) is a service that combines secure, highly available hardware and software to provide a key management system scaled for the cloud. AWS KMS uses customer master keys (CMKs) to encrypt your Amazon S3 objects. You use AWS KMS via the Encryption Keys section in the IAM console or via AWS KMS APIs to centrally create encryption keys, define the policies that control how keys can be used, and audit key usage to prove they are being used correctly. You can use these keys to protect your data in Amazon S3

buckets.

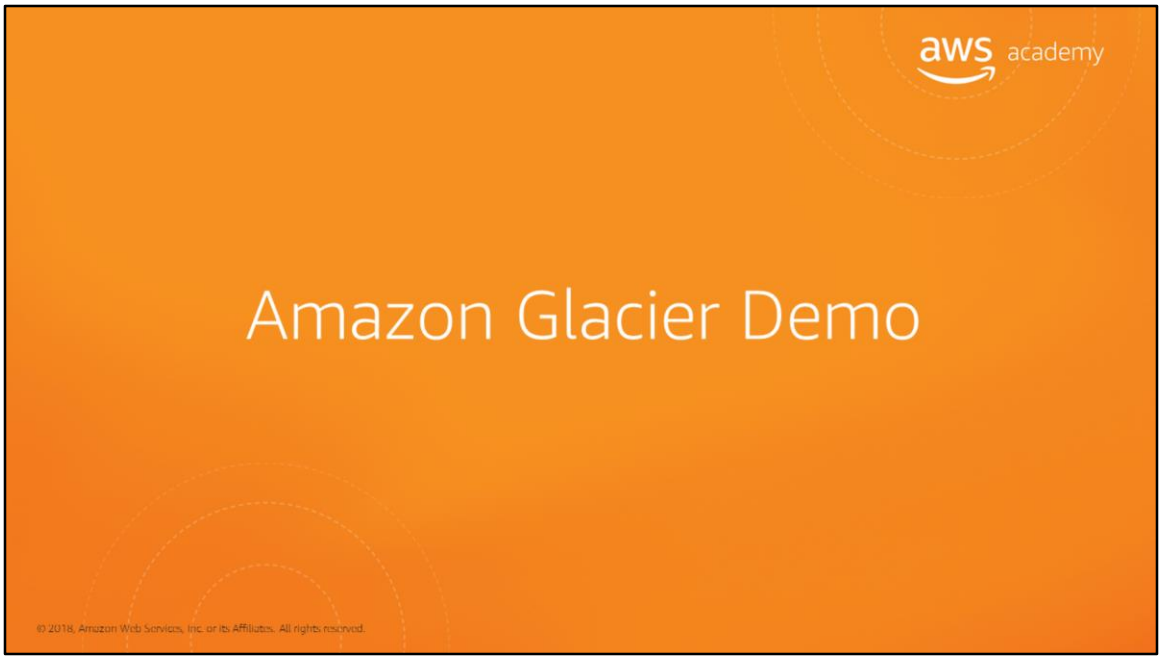
- Using server-side encryption with customer-provided encryption keys (**SSE-C**) allows you to set your own encryption keys. With the encryption key you provide as part of your request, Amazon S3 manages both encryption (as it writes to disks) and decryption (when you access your objects).

By default, only you can access your data. You can enable and control access to your data in Amazon Glacier by using AWS IAM. You just set up an AWS IAM policy that specifies user access.



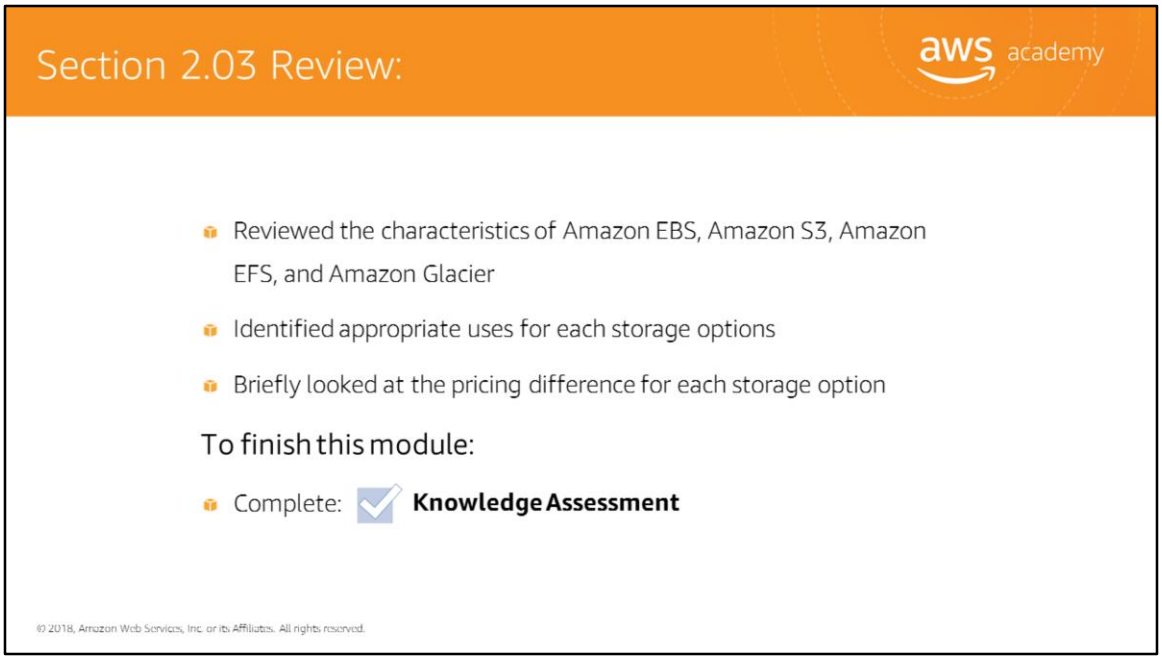
We've covered an introduction to Amazon Glacier including key differences between Amazon S3 and Amazon Glacier.

For more information about Amazon Glacier, see https://aws.amazon.com/glacier/.



Please review the Amazon Glacier demonstration: M2_S3_Glacier v2.0.mp4.

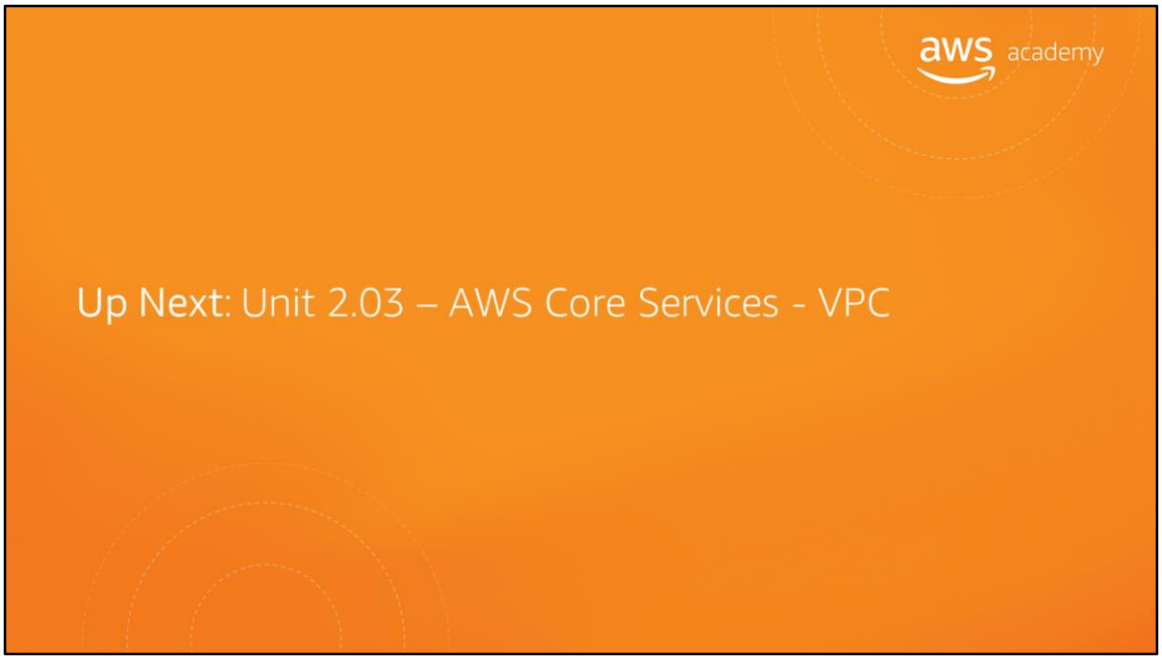
This video demonstration can be found in the learning management system.



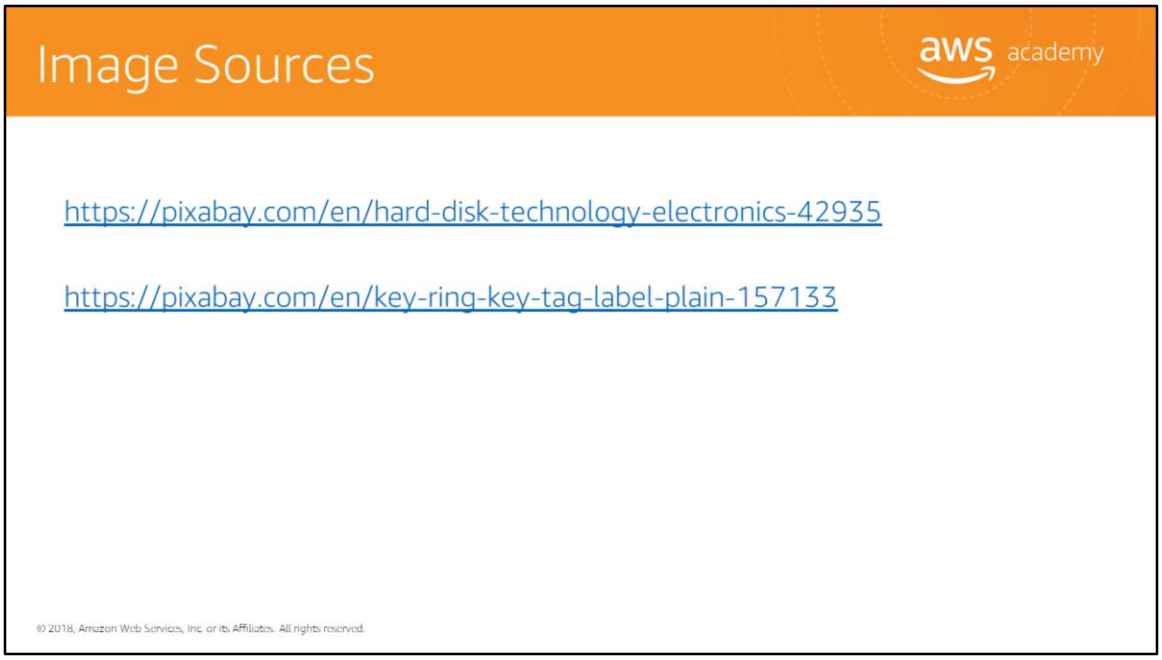
In review, we:

- Discussed storage services including Amazon EBS, Amazon S3, Amazon EFS, and Amazon Glacier
- Reviewed use cases for storage options
- Review storage pricing

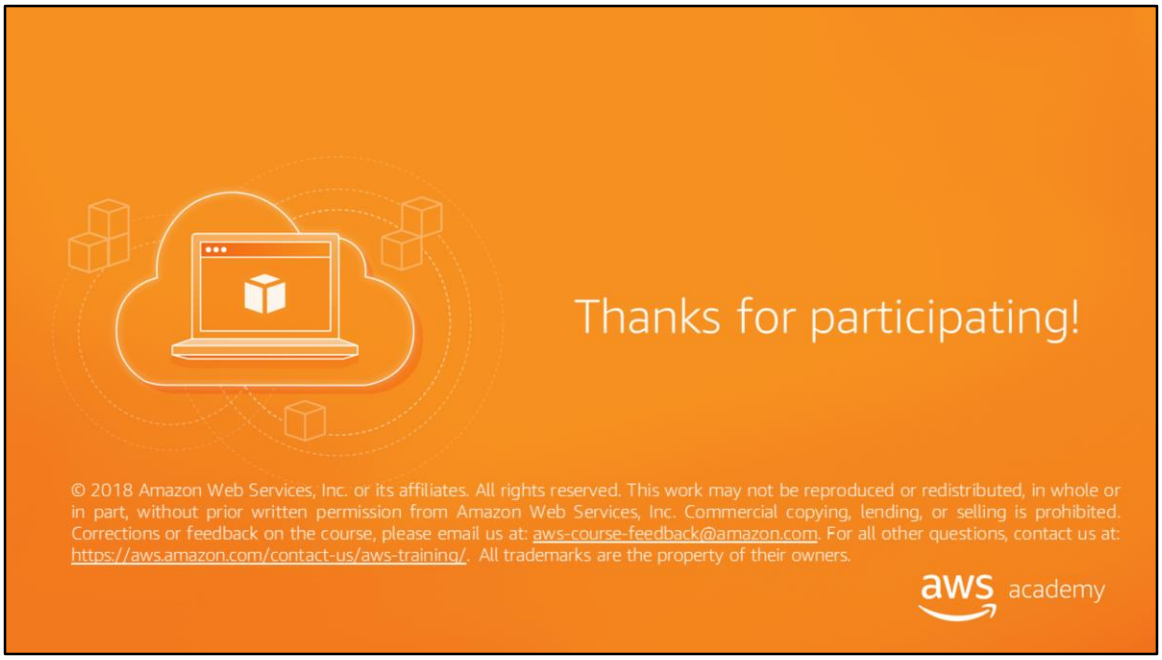
To finish this module, please complete the lab and the corresponding knowledge assessment.

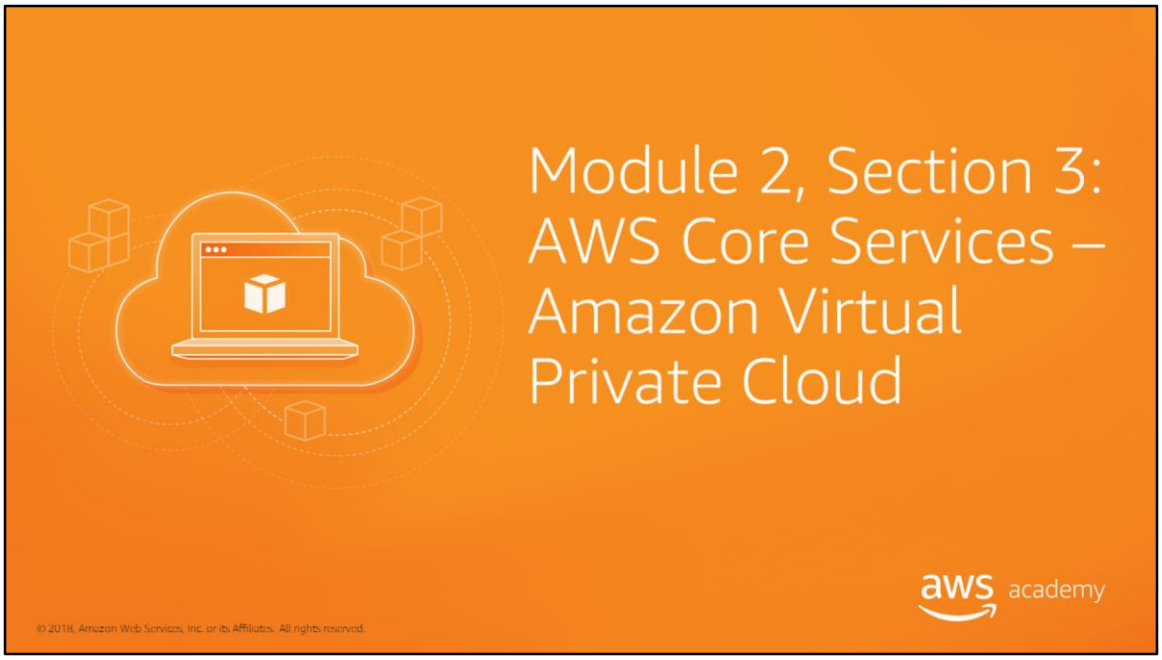


Now that we have a better understanding some of the storage services offered by AWS, in unit 2.03 we look at Amazon Virtual Private Cloud (**Amazon VPC**). It lets you provision a logically isolated section of the AWS Cloud where you can launch AWS resources in a virtual network that you define.

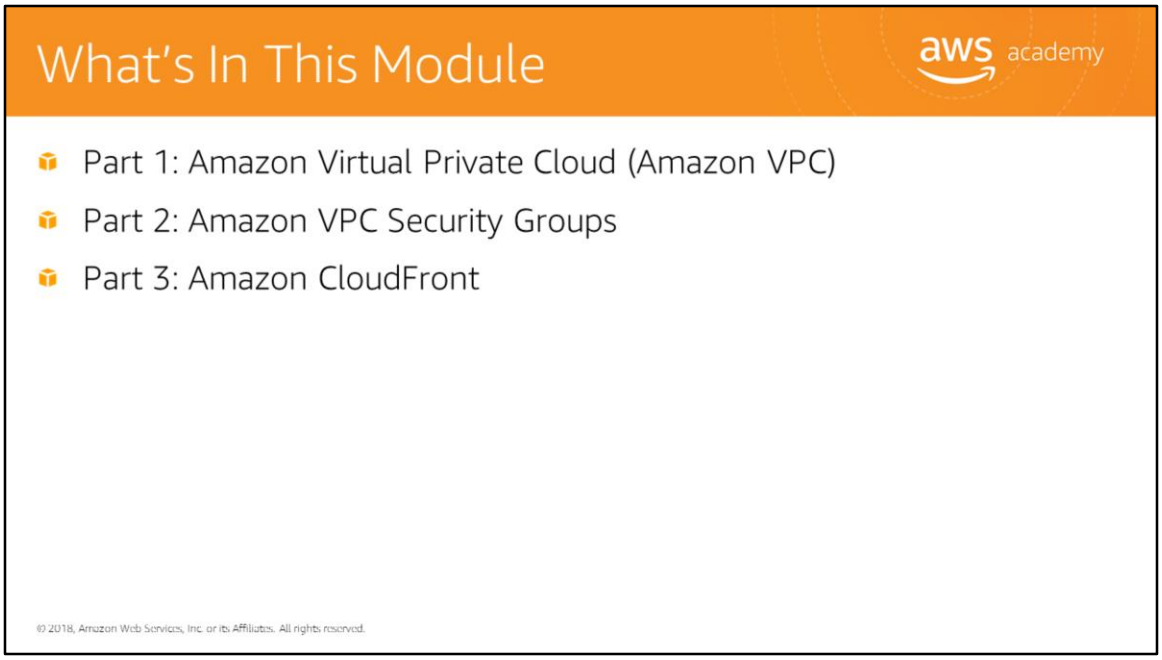


This slide contains attributions for any Creative Commons-licensed images used within this module.

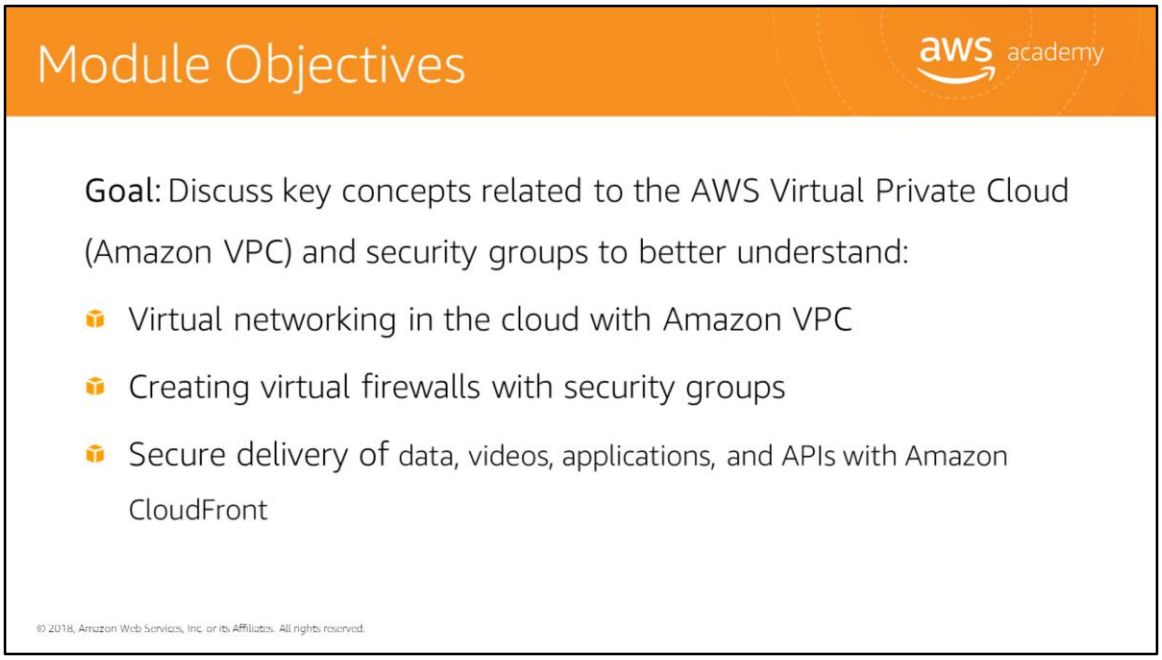




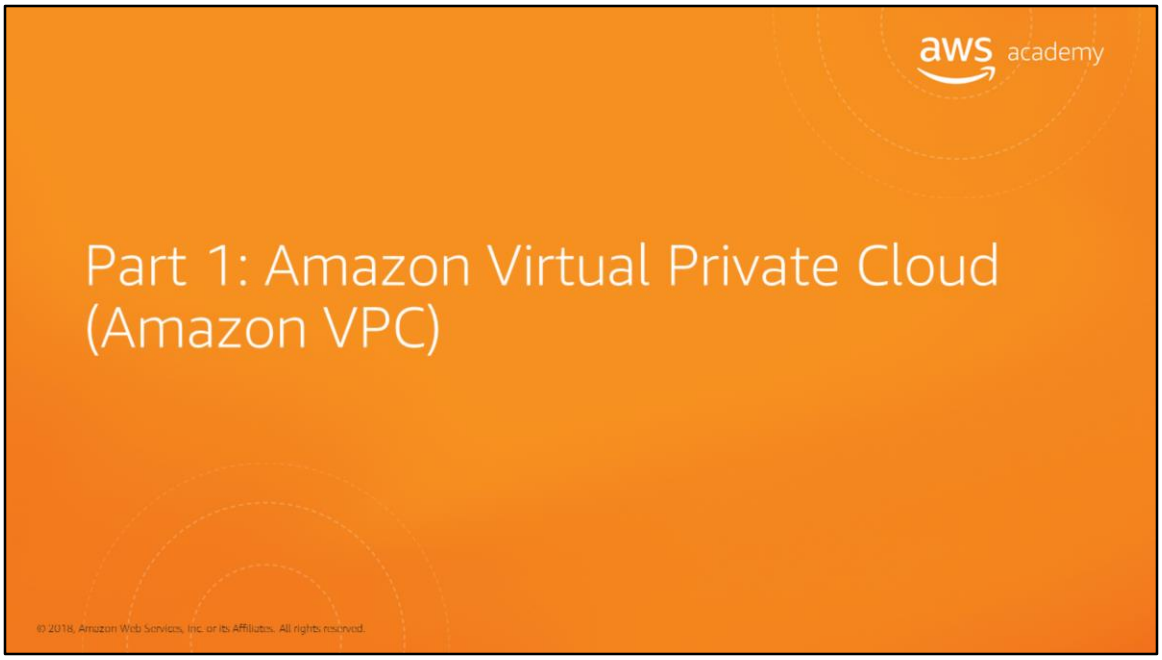
Welcome to Module 2, Section 3 – AWS Core Services – Amazon Virtual Private Cloud (Amazon VPC).



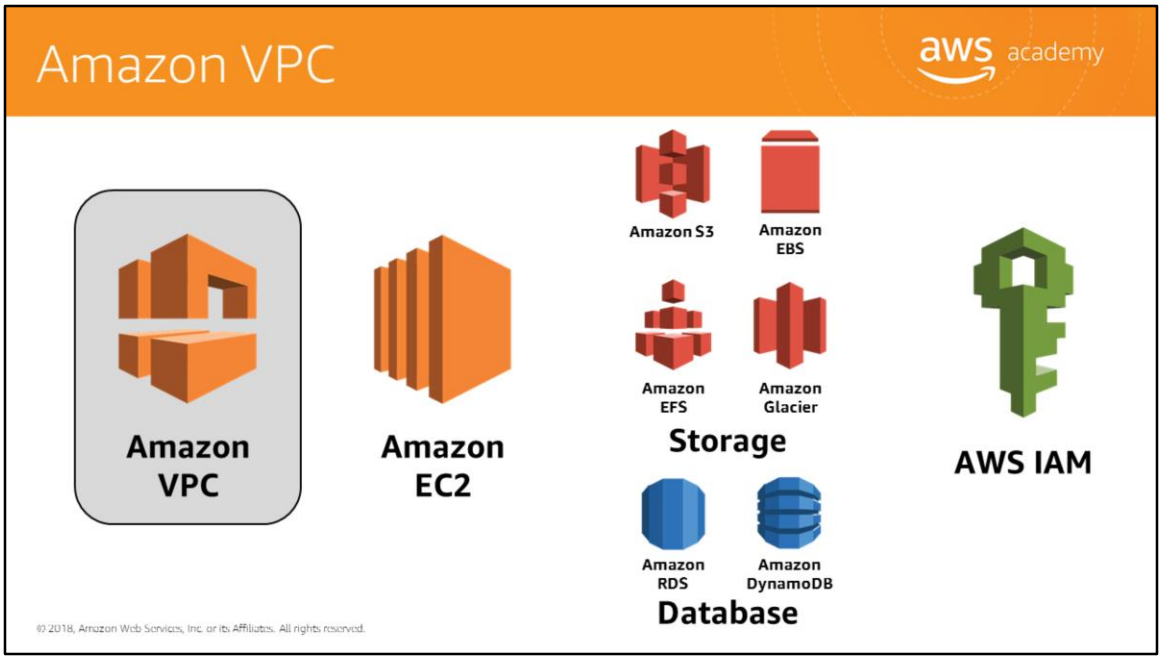
The Amazon Virtual Private Cloud (Amazon VPC) is a custom-defined network within the AWS Cloud. It enables you to design and implement an independent network that operates in the cloud. We will discuss Amazon VPC to help you understand your virtual network in the cloud where you will launch your cloud services. We also review Amazon CloudFront, a global content delivery network (CDN) service that securely delivers data, videos, applications, and APIs to your viewers with low latency and high transfer speeds.



The goal of this module understand virtual networking in the cloud, how to use security groups to add an additional layer of security to your Amazon VPC, and how to add secure content delivery to your solution.

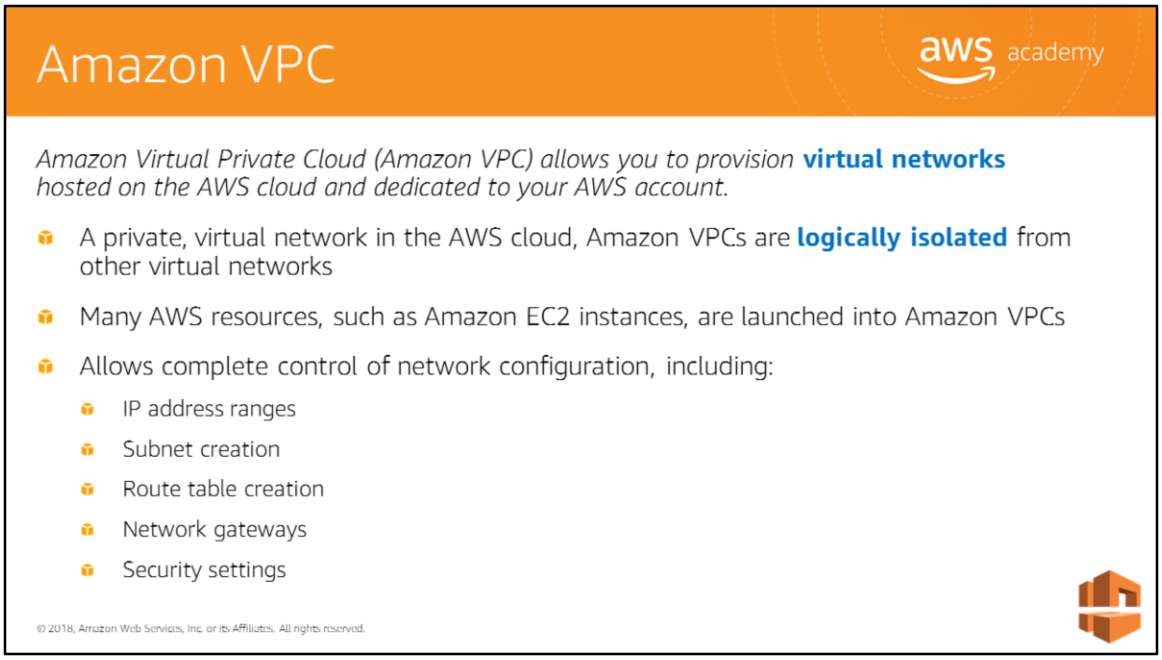


Let’s move on to our next service, Amazon Virtual Private Cloud (Amazon VPC).



The AWS cloud offers pay-as-you-go, on-demand compute as well as managed services, all accessible via the web. These compute resources and services must be accessible via normal IP protocols implemented with familiar network structures. Customers must adhere to networking best practices as well as meet regulatory and organizational requirements. Amazon VPC is the AWS service that will meet your networking requirements and enable you to build your own virtual private network in AWS.

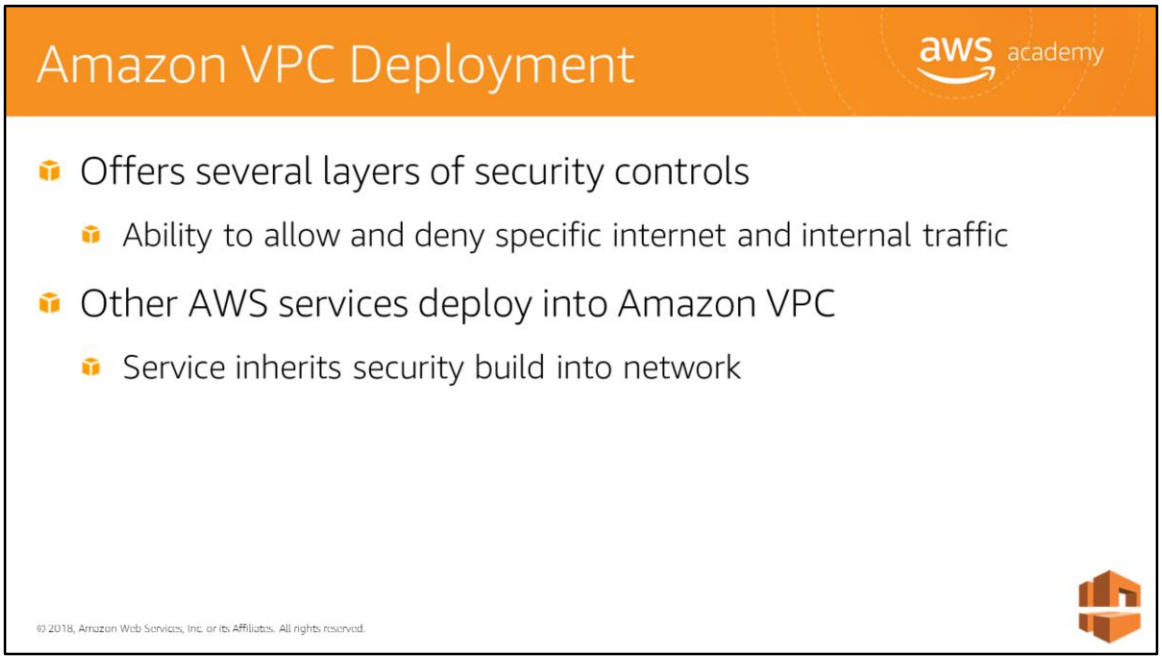
Let’s dive a little deeper into Amazon VPC.



Amazon VPC is your network environment in the cloud. It allows you to create a private network within the AWS cloud that uses many of the same concepts and constructs as an on-premises network, but as we shall see later, much of the complexity of setting up a network has been abstracted without sacrificing control, security, and usability.

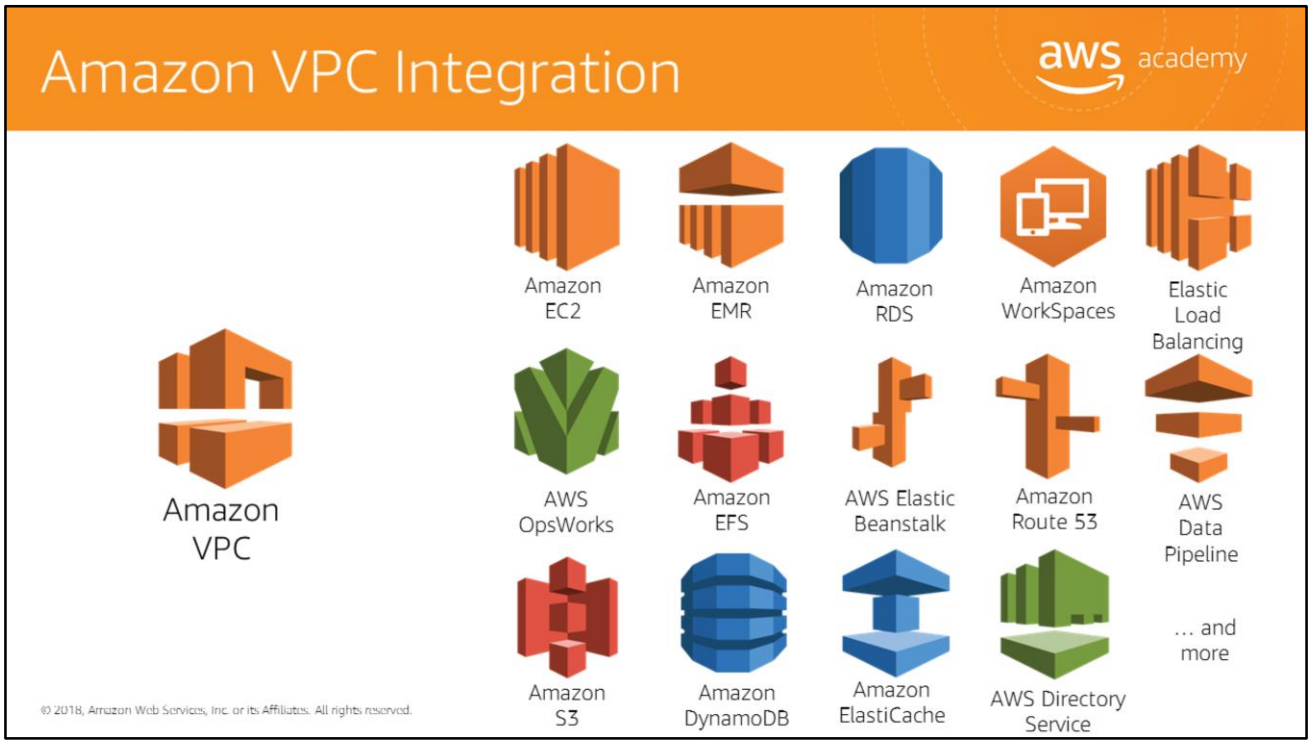
Amazon VPC is where you will launch many of your resources, and it's designed to provide greater control over the isolation of your environments and their resources from each other. Within a region, you can create multiple Amazon VPCs, and each Amazon VPC is logically isolated even if it shares its IP address space.

Amazon VPC also gives you complete control of the network configuration. Customers can define normal networking configuration items such as IP address ranges, subnet creation, route table creation, network gateways, and security settings. This allows you to control what you expose to the Internet and what you isolate within the Amazon VPC.

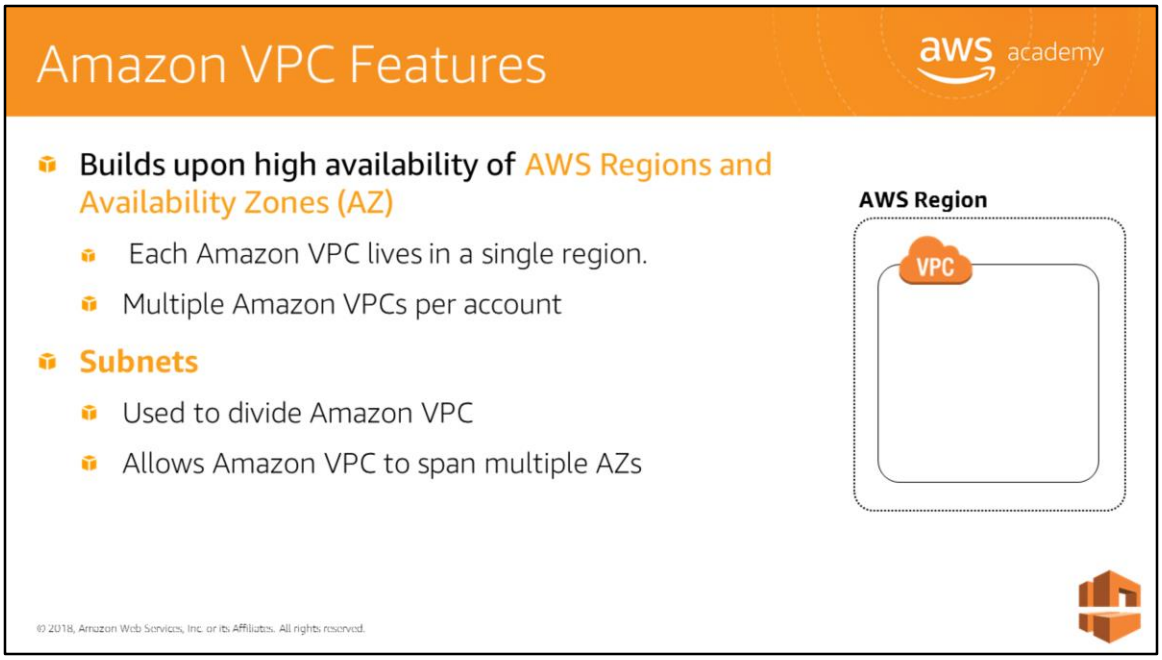


You can deploy your Amazon VPC in a way to layer security controls in the network. This includes isolating subnets, defining access control lists, and customizing routing rules. You have complete control to allow and deny both incoming and outgoing traffic.

Finally, there are numerous AWS services that deploy into your Amazon VPC that then inherit and take advantage of the security that you have built into your cloud network.



Amazon VPC is an AWS foundational service and integrates with numerous AWS services. For instance, Amazon EC2 instances are deployed into your Amazon VPC. Similarly, Amazon Relational Database Service (Amazon RDS) database instances deploy into your Amazon VPC, where the database is protected by the structure of the network just like your on-premises network. Understanding and implementing Amazon VPC will allow you to fully use other AWS services.

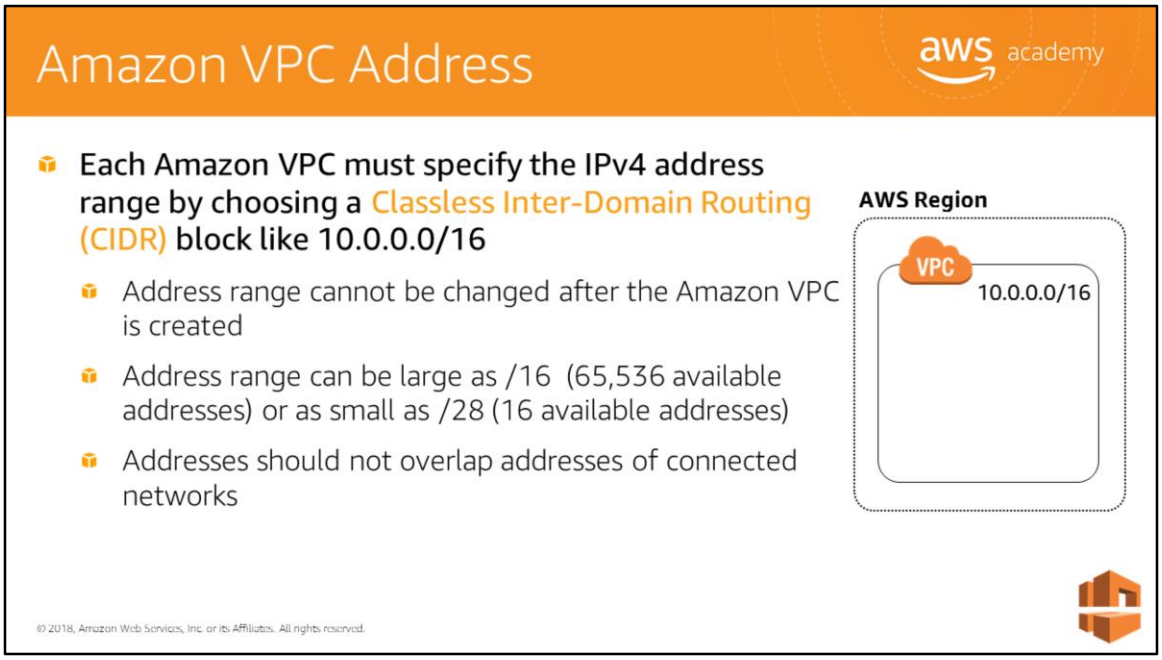


Let's take a look at the features of Amazon VPC. Amazon VPC builds upon the AWS global infrastructure of Regions and Availability Zones (AZ), and allows you to easily take advantage of the high availability provided by the AWS cloud. It also allows you to provision virtual networks hosted on the AWS cloud and dedicated to your AWS account. Amazon VPCs live within regions – they can exist only in a single region. (There are ways to connect Amazon VPCs in different regions to each other without going through the public Internet.) Each AWS account can create multiple Amazon VPCs that can be used to segregate environments.

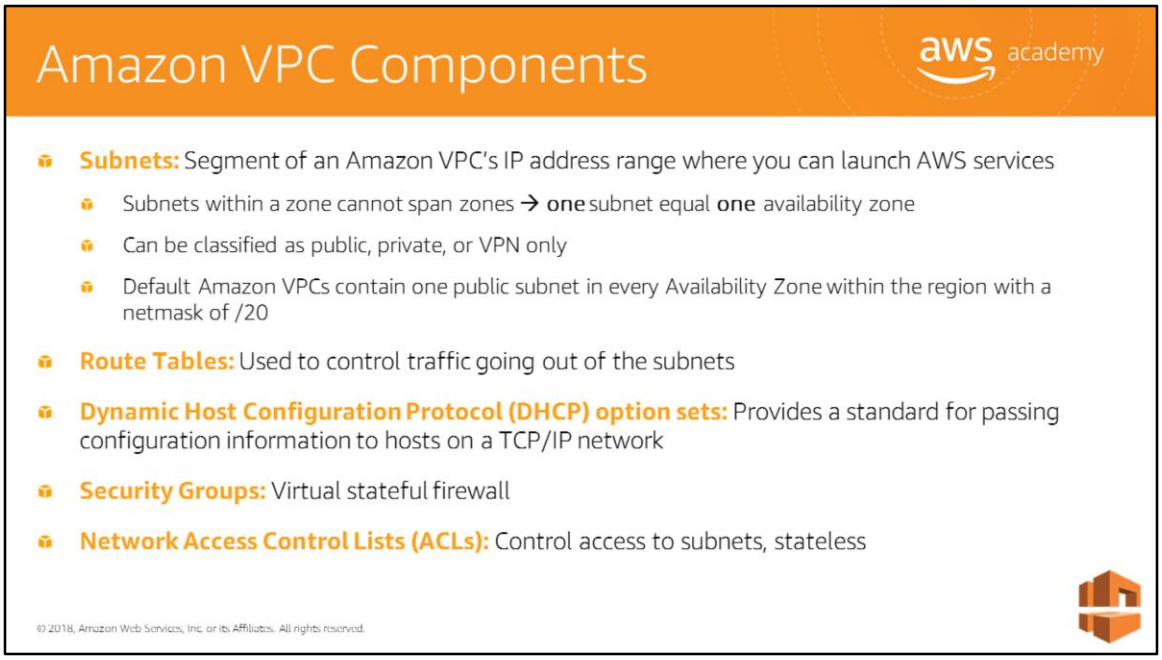
A Amazon VPC defines an IP address space that is then divided by subnets. These subnets are deployed within Availability Zones causing the Amazon VPC to span AZs. Amazon VPCs are logically isolated from other virtual networks. You can create many subnets in a Amazon VPC, though fewer is recommended to limit the complexity of the network topology, but this is totally up to you. You can configure route tables for your subnets to control the traffic between subnets and the Internet. By default, all subnets within a Amazon VPC can communicate with each other. It should be noted that while a Amazon VPC can span across multiple AZs, a subnet cannot.

Subnets are generally classified as public or private, with ***public*** having direct access to the Internet and ***private*** not having direct access to the Internet. For a subnet to be public, we need to attach an Internet gateway to the Amazon VPC and update the route table of the

public subnet to send non-local traffic to the Internet gateway. Amazon EC2 instances also need a public IP address to route to an Internet gateway.

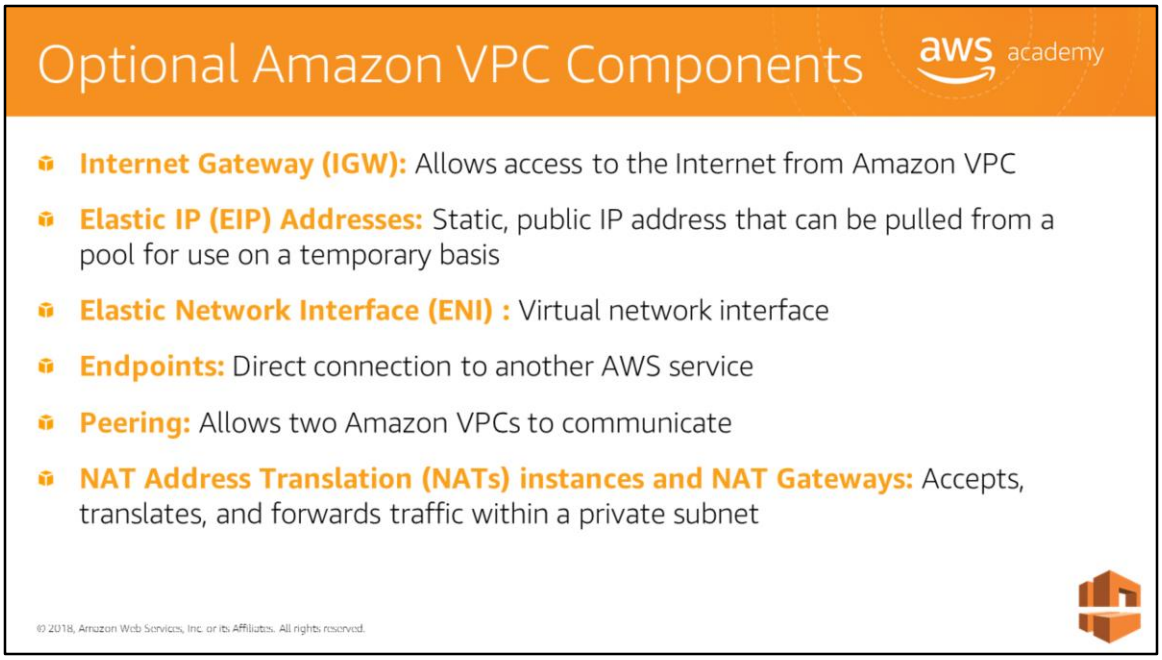


When you create an Amazon VPC, you must specify the IPv4 address range by choosing a Classless Inter-Domain Routing (CIDR) block, such as 10.0.0.0/16. The address range of the Amazon VPC cannot be changed after the Amazon VPC is created. An Amazon VPC address range may be as large as /16 (65,536 addresses available) or as small as /28 (16 addresses available) and should not overlap any addresses of other networks they are connected to.



You can use the following components to configure networking in your Amazon VPC:

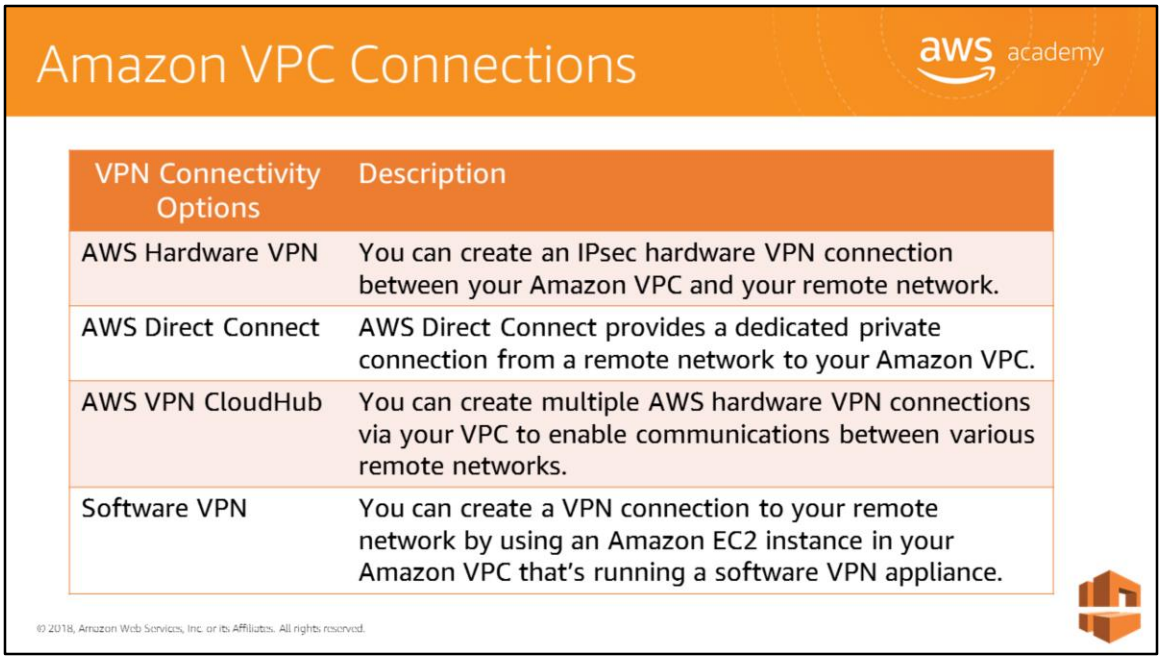
- A **subnet** is a segment of an Amazon IPC address range where you can launch AWS services.
    - CIDR blocks define subnets
    - AWS reserves the first four IP addresses and the last IP address of every subnet for internal networking purposes.
    - A public subnet is one in which an associated route table direct the subnet's traffic to the Amazon VPC's internet gateway. A private subnet is one in which the associated route table does not direct the subnet's traffic to the internet gateway. A VPN only subnet only direct traffic to the Amazon VPC's virtual private gateway.
- A **route table** contains a set of rules, called *routes*, that are used to determine where network traffic is directed. Each subnet in your Amazon VPC must be associated with a route table; the table controls the routing for the subnet. A subnet can only be associated with one route table at a time, but you can associate multiple subnets with the same route table. To learn more about route tables see https://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/VPC_Route_Tables.html.
- AWS automatically create and associates a **Dynamic Host Configuration Protocol (DHCP)** option set for your Amazon VPC upon creation and sets two options: domain-name-servers and domain-name.
- A **security group** is a virtual stateful firewall that controls inbound and outbound network traffic to AWS resources and Amazon EC2 instances. For more information about security groups see https://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/VPC_SecurityGroups.html
- A **network access control list (NACL)** is an optional layer of security for your Amazon VPC that acts as a firewall for controlling traffic in and out of one or more subnets. For more information about NACL see https://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/VPC_ACLs.html.



There are some optional Amazon VPC components:

- An **Internet Gateway (IGW)** is a horizontally scaled, redundant, and highly available Amazon VPC component that allows communication between instances in your Amazon VPC and the Internet. For more information about internet gateways, see https://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/VPC_Internet_Gateway.html.
- An **Elastic IP (EIP) Address** is a static IPv4 address designed for dynamic cloud computing. An Elastic IP address is associated with your AWS account. For more information see https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/elastic-ip-addresses-eip.html
- **Elastic Network Interface (ENI)** is a virtual network interface that you can attach to an instance in an Amazon VPC. For more information see https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-eni.html
- An Amazon VPC **endpoint** enables you to create a private connection between your Amazon VPC and another AWS service without requiring access over the Internet or through a NAT instance, VPN connection, or AWS Direct Connect. For more information see https://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/vpc-endpoints.html
- An Amazon VPC **peering** connection is a networking connection between two Amazon VPCs that enables instances in either Amazon VPC to communicate with each other as if they are within the same network. For more information see https://docs.aws.amazon.com/AmazonVPC/latest/PeeringGuide/Welcome.html
- **NAT Address Translation instances** is an Amazon Linux AMI designed to keep traffic from instances within a private subnet. A **NAT Gateway** is an Amazon managed resources designed to

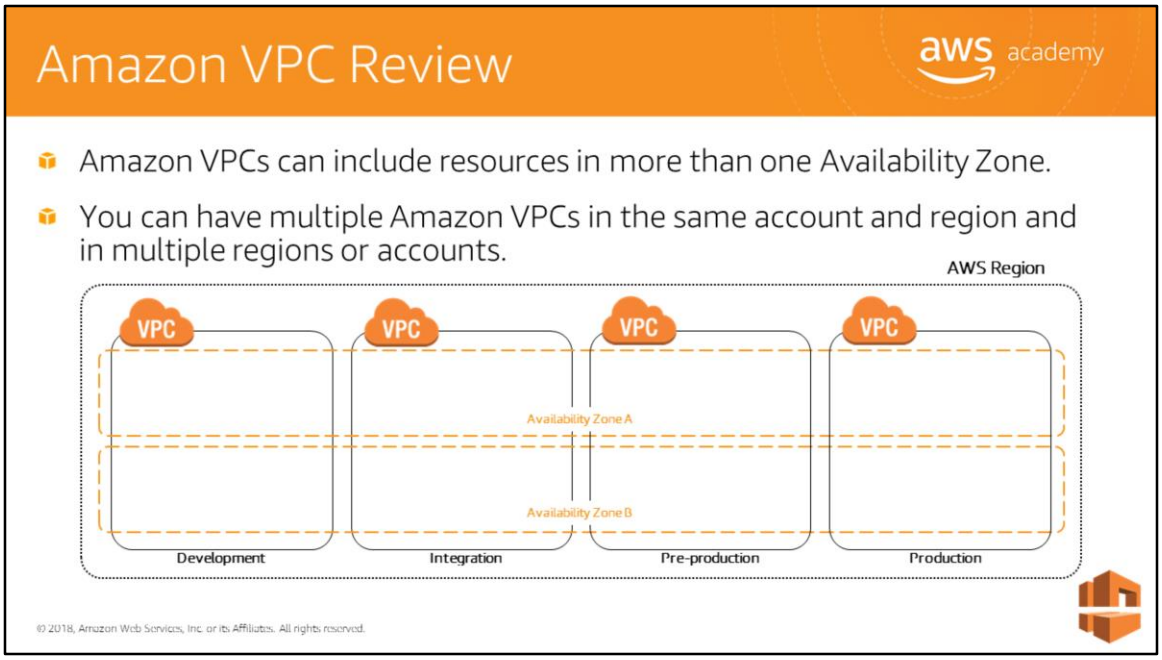
operate just like a NAT instance, but is simpler to manage and highly available within an AZ.



You can connect your Amazon VPC to remote networks using a VPN connection.

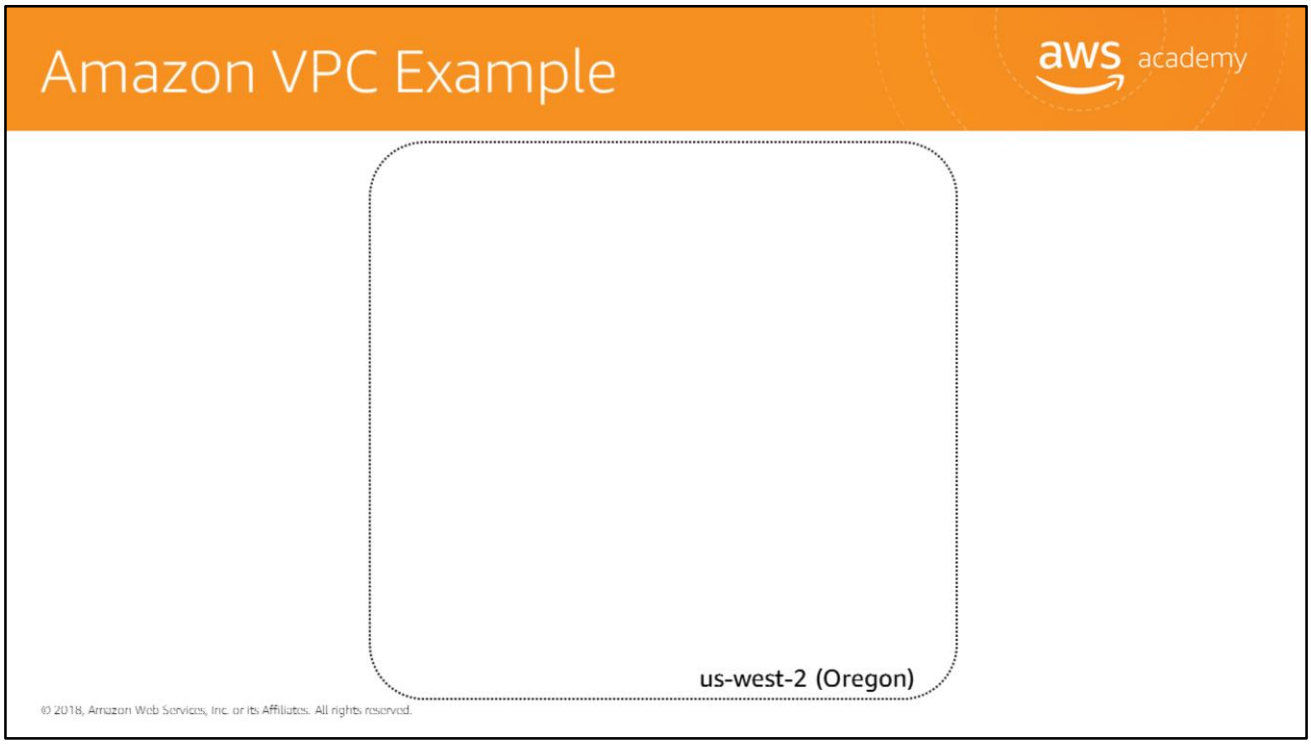
For more information, see:

- Amazon Virtual Private Cloud Connectivity Options whitepaper https://media.amazonwebservices.com/AWS_Amazon_VPC_Connectivity_Options.pdf
- http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/vpn-connections.html
- http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/VPN_CloudHub.html

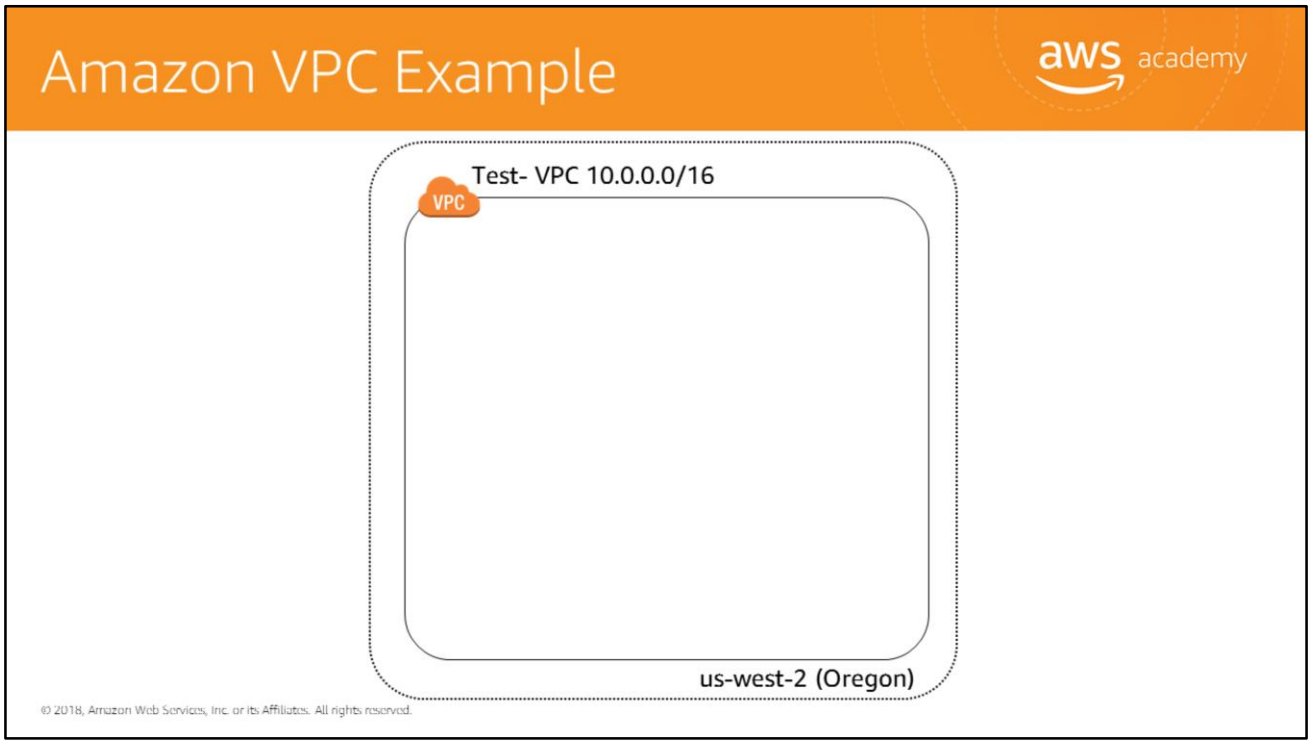


To review, Amazon VPC is your network environment in the cloud. It allows you to:

- Create a private network within the AWS cloud that uses many of the same concepts and constructs as an on-premises network.
- Include resources in more than one Availability Zone.
- Have multiple Amazon VPCs in each account or region and VPCs in as many regions as you'd like or in multiple accounts.
- Connect your Amazon VPC to remote networks using a VPN connection.



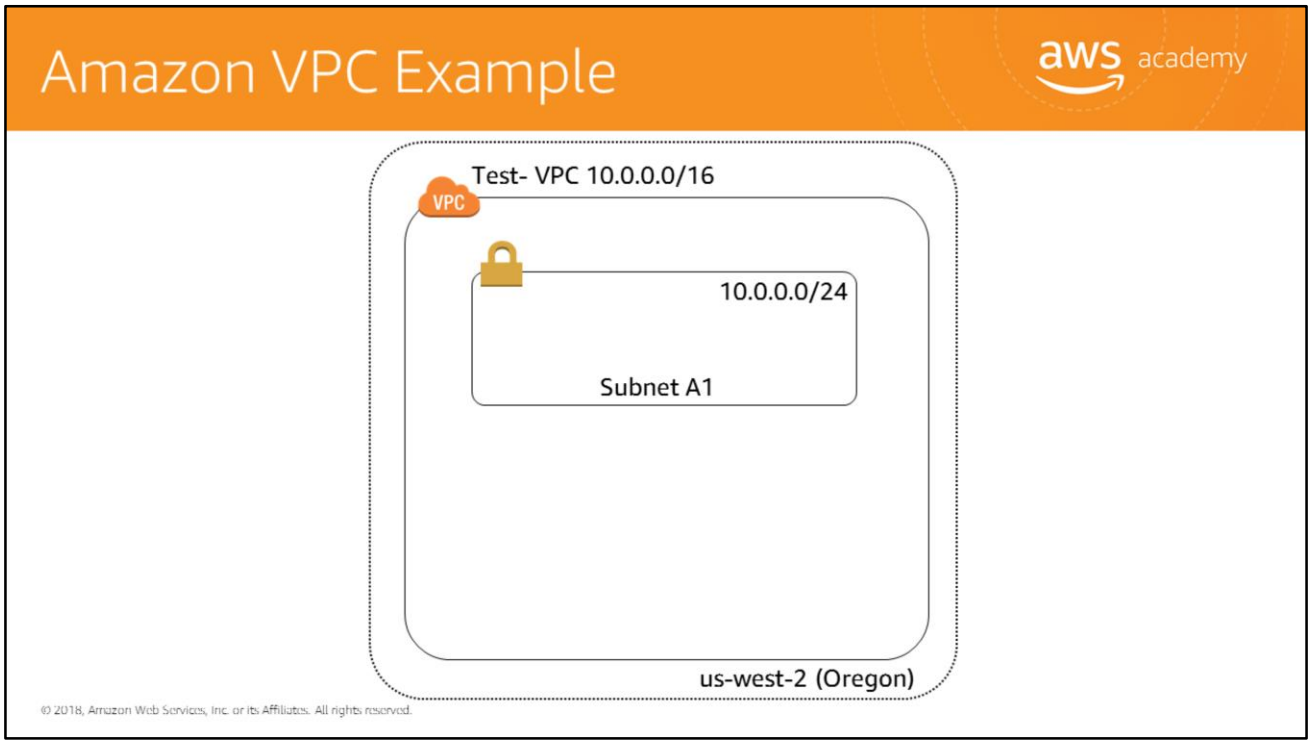
Let's design an example Amazon VPC that we can use to start deploying compute resources and AWS services. We'll create a network that supports high availability and uses multiple subnets. Since VPC are region based, we need to select a region. In this example, we've selected the Oregon region.



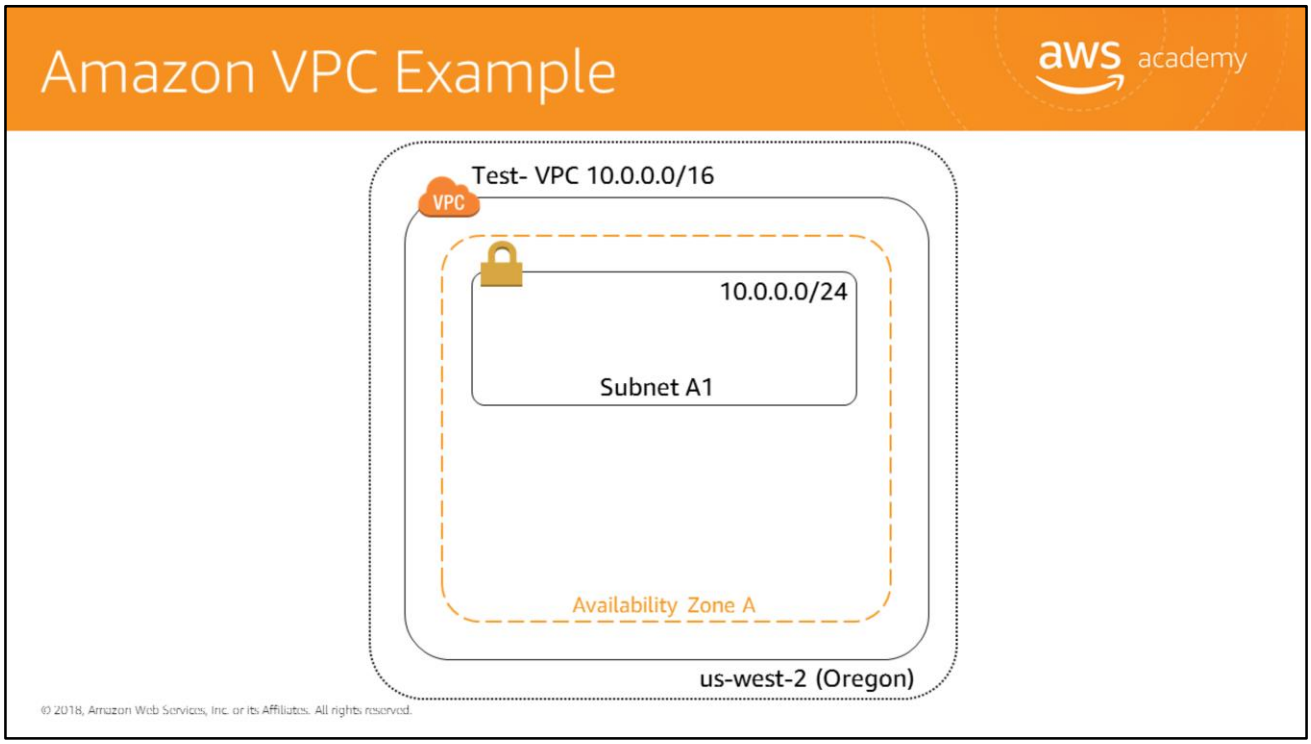
Next, we'll create the Amazon VPC and give it a name, *Test VPC,* and define the IP address space for the Amazon VPC. The 10.0.0.0/16 is the Classless Inter-Domain Routing (CIDR) format and means that there are over 65,000 IP addresses to use in the Amazon VPC.

CIDR (sometimes called *supernetting*) is a way to allow more flexible allocation of Internet Protocol (IP) addresses than was possible with the original system of IP address classes. A single IP address can be used to designate many unique IP addresses with CIDR. A CIDR IP address looks like a normal IP address, except that it ends with a slash followed by a number, called the IP network prefix. CIDR addresses reduce the size of routing tables and make more IP addresses available within organizations.

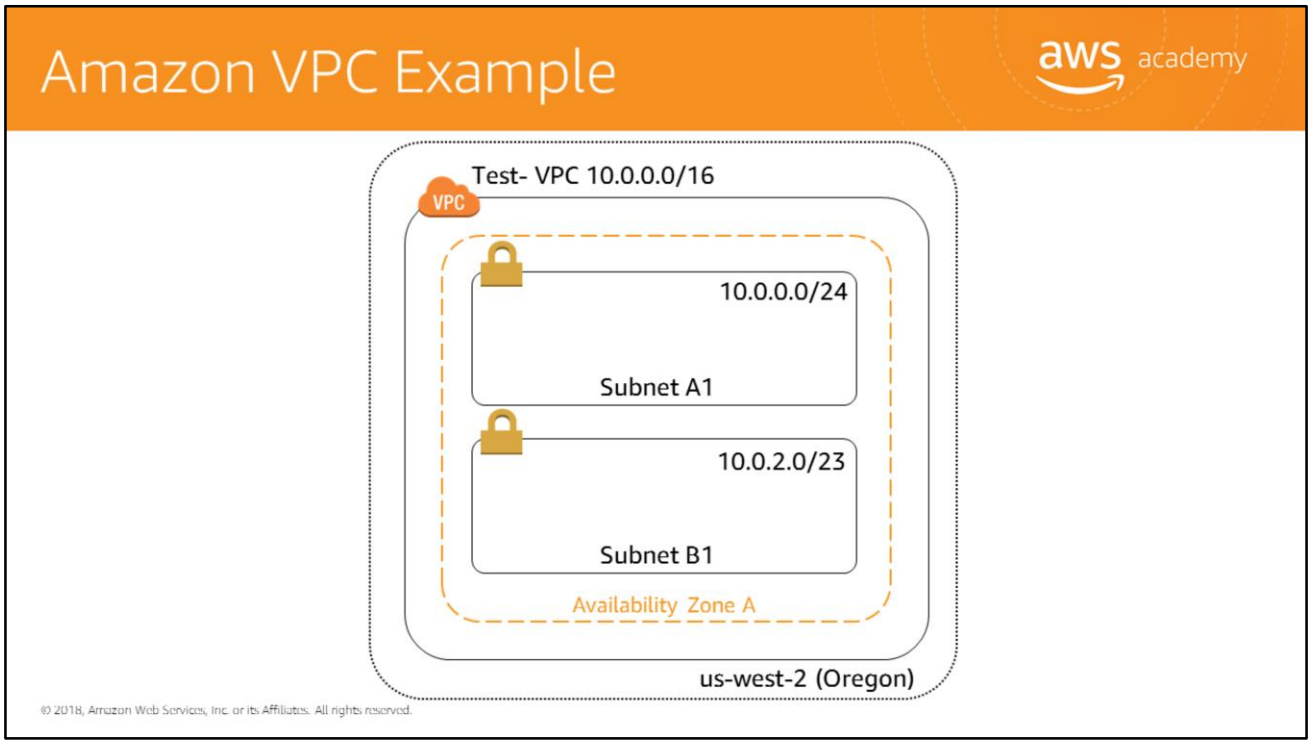
To illustrate, a CIDR network address looks like this: 192.30.250.00/18. The 192.30.250.0 is the network address itself and the “18” says that the first 18 bits are the network part of the address, leaving the last 14 bits for specific host addresses.



Next, we create a subnet named *Subnet A1* and assign an IP address space that contains 256 IP addresses.

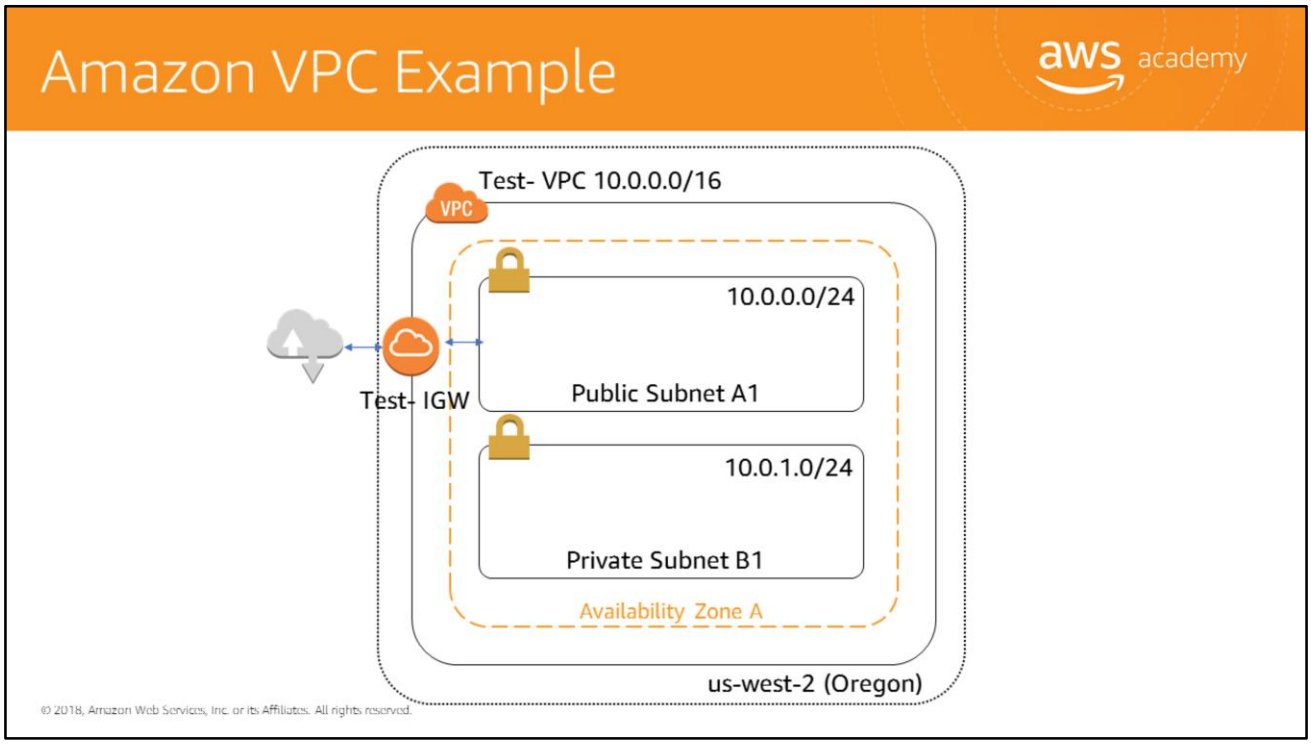


Also, we specify that this subnet will live in an Availability Zone A.



Finally, we create another sub-net called *Subnet B1* and assign an IP address space. This subnet contains 512 IP addresses.

Let's make a few more additions that will make Subnet A1 accessible via the Internet.



To accomplish this, add an internet gateway called *Test IGW*. Subnet A1 now becomes a public subnet where non-local traffic is routed through the Internet gateway. Subnet B1 will be our private subnet that is isolated from the Internet.

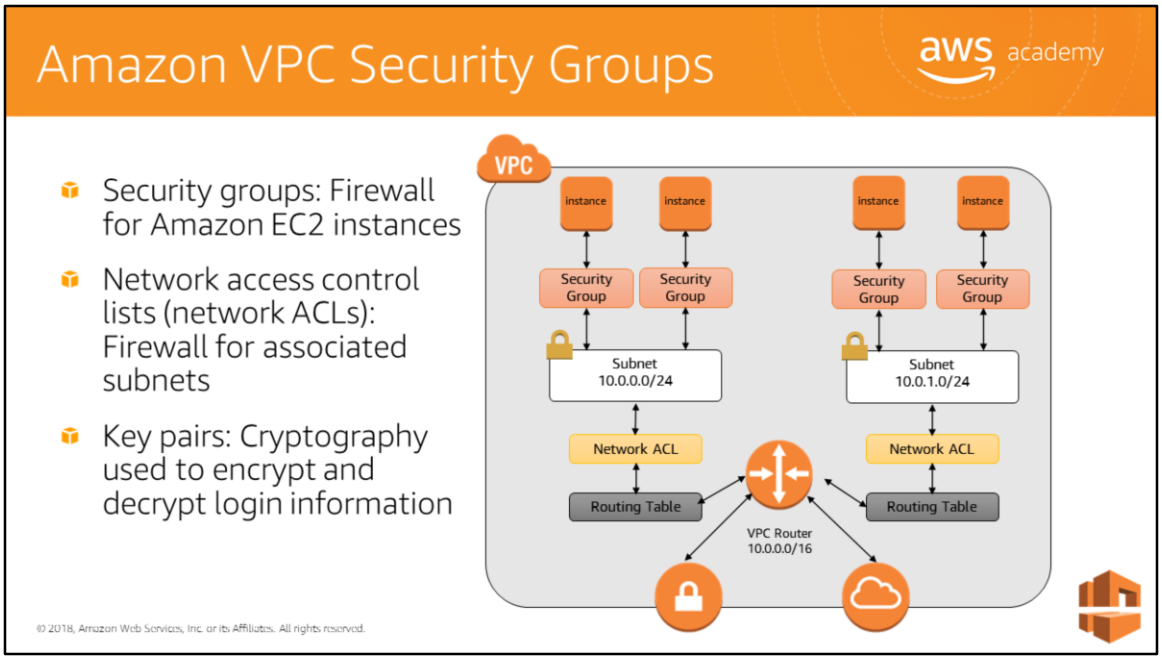
Security of the AWS Cloud is one of Amazon Web Services' highest priorities. This section reviews how AWS Security Groups can be utilized to improve your Amazon VPC security.

AWS VPC Security Groups

- Security groups act like a built-in firewall for your virtual servers.
- Security group rules determine who has access to instances.
- Security groups are stateful.

Let's take a look at security groups and how they help secure your data. At AWS, security groups will act like a built-in firewall for your virtual servers. With these security groups, you have full control on how accessible your instances are.

At the most basic level, it is just another method to filter traffic to your instances. It provides you control on what traffic to allow or deny. To determine who has access to your instances, you would configure a security group rule. Rules can vary from keeping the instance completely private, totally public, or somewhere in between.

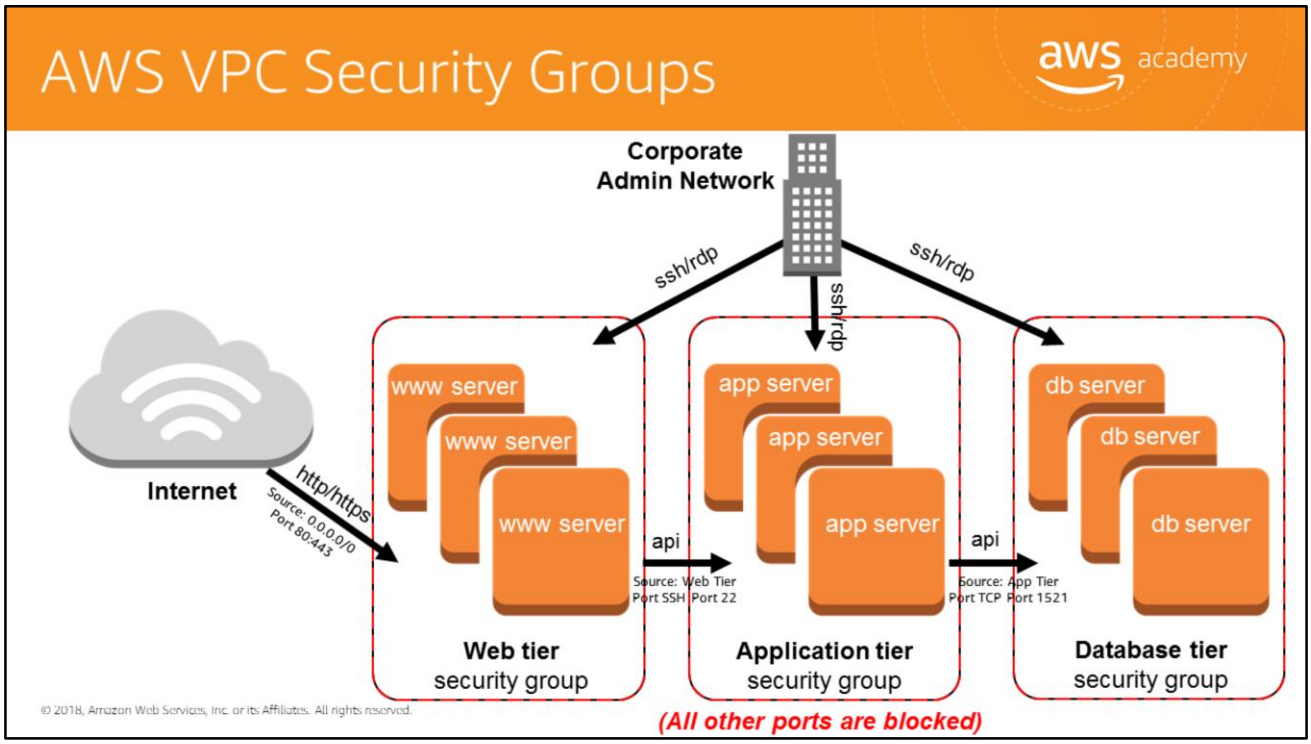


Amazon VPC provides various features that you can use to increase and monitor the security for your Amazon VPC:

- **Security groups** act as a firewall for associated Amazon EC2 instances, controlling both inbound and outbound traffic at the instance level.
- **Network access controls lists (network ACLs)** act as a firewall for associated subnets, controlling both inbound and outbound traffic at the subnet level.
- Amazon EC2 uses **public-key cryptography** to encrypt and decrypt login information. Public-key cryptography uses a public key to encrypt a piece of data, and the recipient uses the private key to decrypt the data. The private and public keys are known as a *key pair*. To log in to your instance, you must create a key pair, specific the name of the key pair when you launch the instance, and provide the private key when you connect to the instance. Linux instances have no password, and you use a key pair to log in using SSH. Windows instances require a key pair to obtain the administrator password to log in using RDP.

It should be noted that security groups are *stateful* while NACLs are *stateless*.

- **Stateful** means the computer keeps track of the state of interaction, usually by setting values in a storage field designated for that purpose.
- **Stateless** means no information is retained by either sender or receiver, and each interaction request has to be handled based entirely on information that comes with it.



Here is an example of a classic AWS multi-tier security group. In this architecture, you will notice that multiple different security group rules have been created to accommodate this multi-tiered web architecture.

If we start at the **web tier,** you will see that we have set up a rule to accept traffic from anywhere on the internet on port 80/443 by selecting the source 0.0.0.0/0.

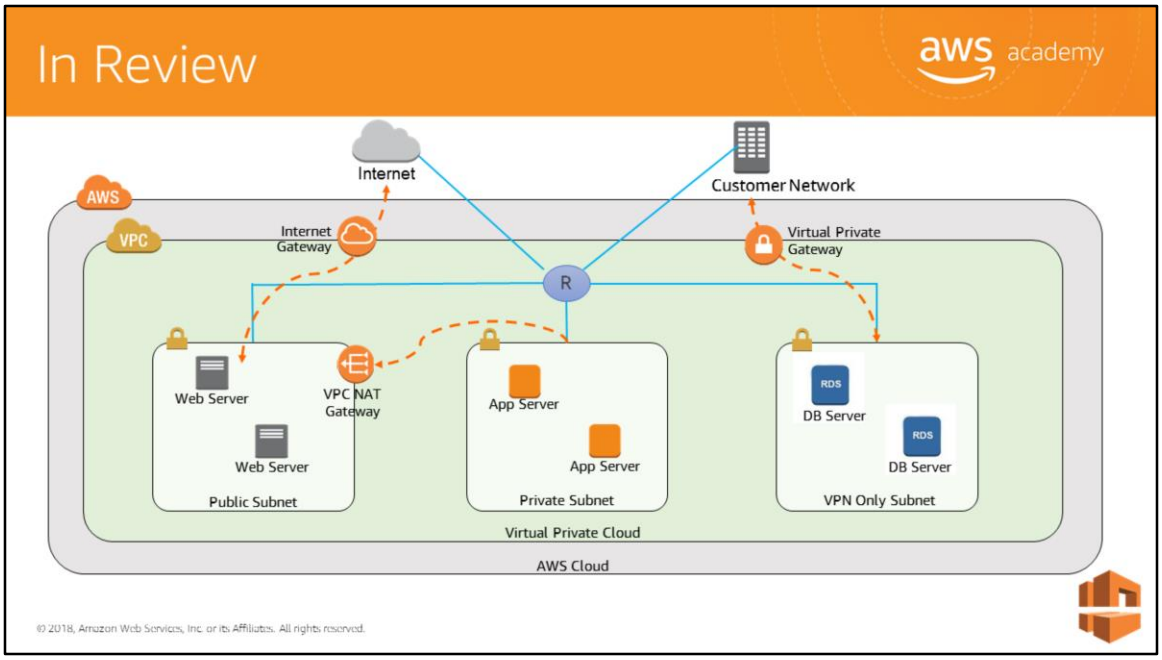
Next, moving to the **app tier**, there is a security group that only accepts traffic from the web tier, and similarly, the **database tier** can only accept traffic from the app tier.

Finally, you will notice that there has also been a rule created to allow **administration remotely** from the corporate network over SSH port 22.

To summarize what we have discussed about AWS Security Groups:

- AWS provides virtual firewalls that can control traffic for one or more instances, called *security groups*.
- Security groups are stateful.
- You can control accessibility to your instances by creating security group *rules*.
- These security groups can be managed on the AWS Management Console.

To learn more see https://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/VPC_SecurityGroups.html.

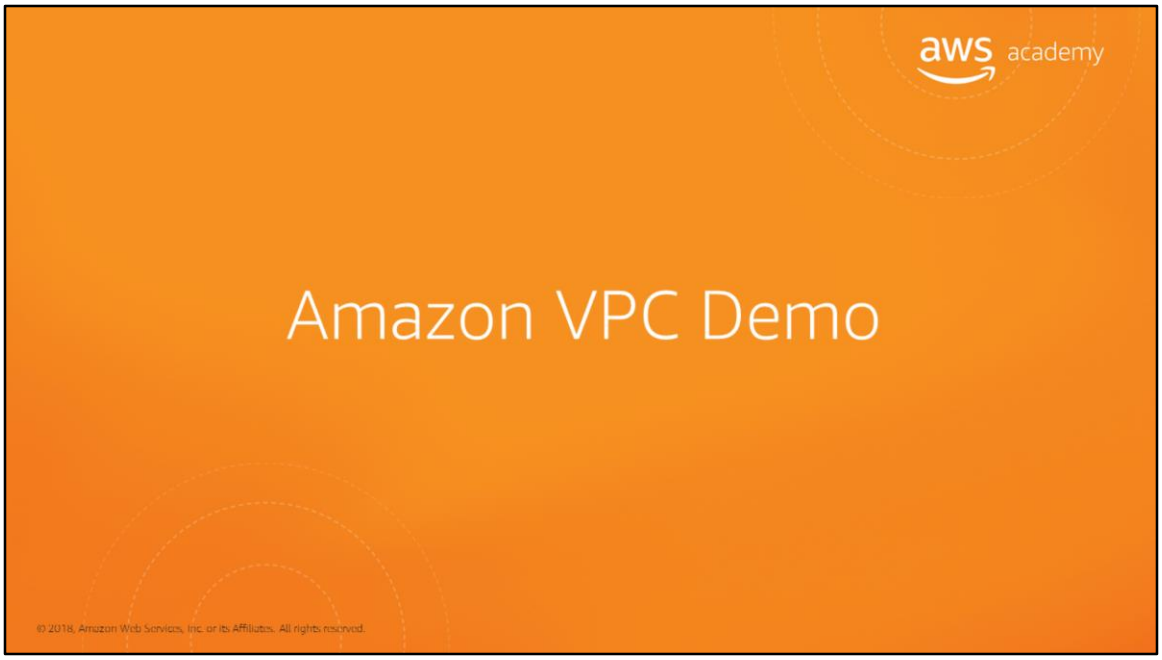


Let's summarize what we have covered so far. Amazon VPC allows you provision a logically isolated section of the AWS cloud where you can launch AWS resources in a virtual network that you define. With Amazon VPC:

- You have complete control over your virtual networking environment, including selection of your own IP address range, creation of subnets, configuration of route tables, network access control lists, and network gateways.
    - Each subnet must reside entirely within one Availability Zone and **cannot span zones.**
    - A **subnet** defines a range of IP addresses in your Amazon VPC.
    - You can launch AWS resources into a subnet that you select.
    - A **private subnet** should be used for resources that won't be accessible over the internet.
    - A **public subnet** should be used for resources that will be accessed over the internet.
- You can easily customize the network and configuration for your Amazon VPC instance. For example, you can create a public-facing subnet for your web servers that require access to the internet and place your back-end systems, such as databases or application servers, in a private-facing subnet with no internet access.
- You can create a hardware virtual private network (VPN) connection between your

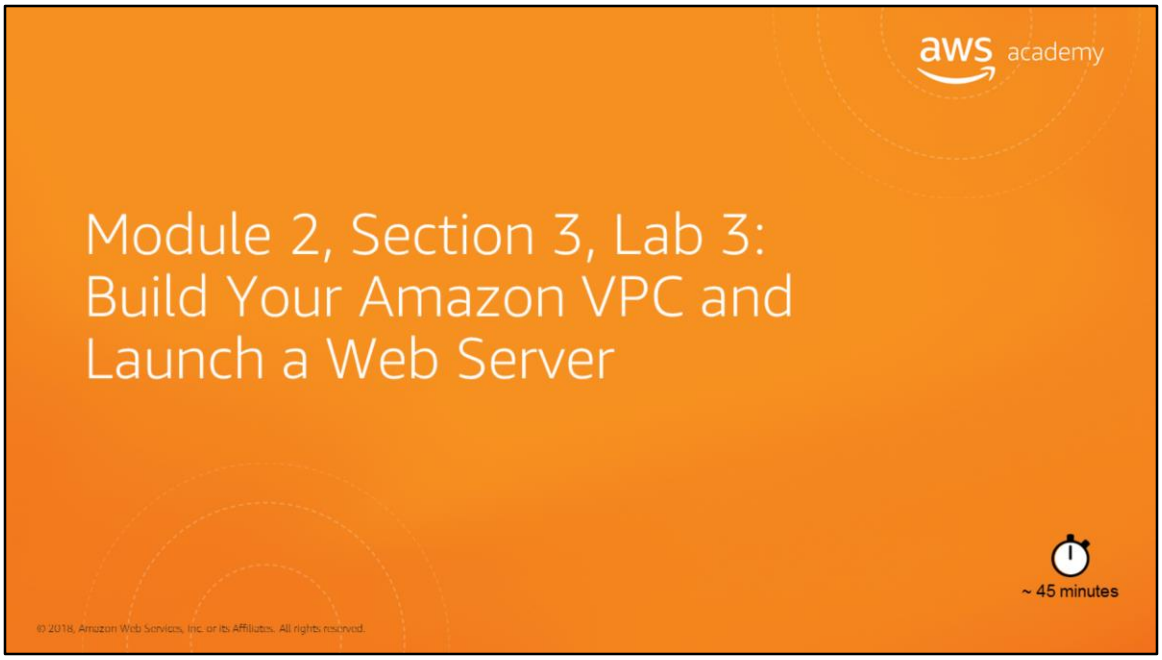
corporate data center and your Amazon VPC, which allows you to use the AWS cloud as an extension of your corporate data center.

- You can use multiple layers of security, including security groups and NACLs, to help control access to Amazon EC2 instances in each subnet.



Please review the Amazon VPC demonstration: M2_S1_CoreSVCS.mp4.

This video demonstration can be found in the learning management system.



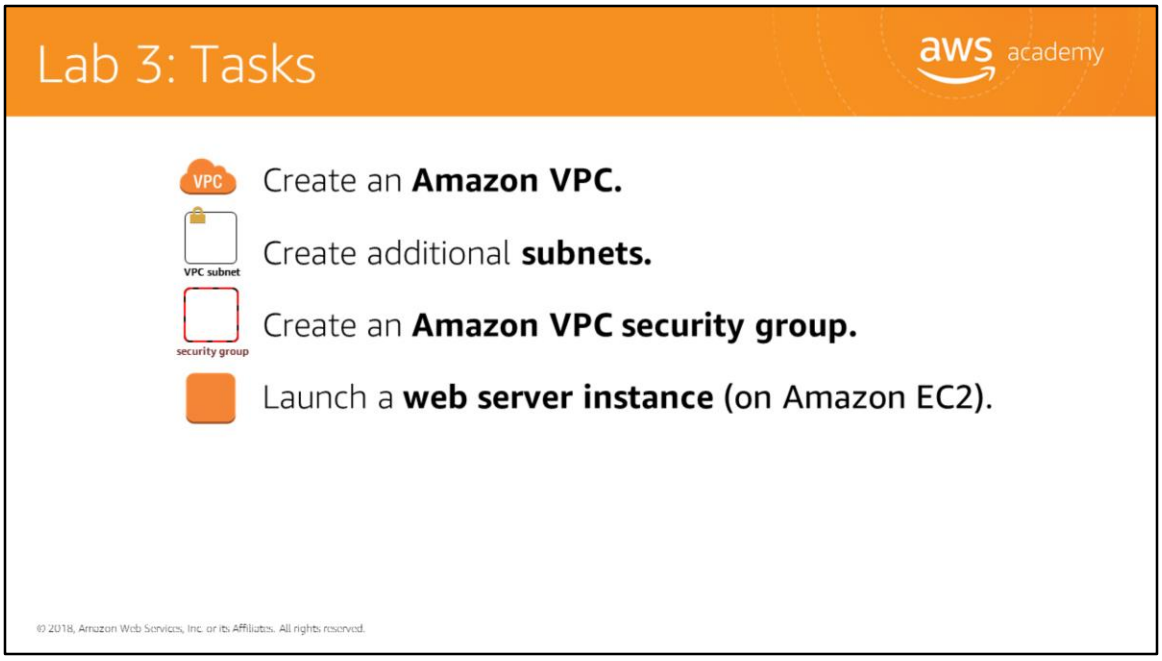
Amazon VPC enables you to launch AWS resources into a virtual network that you define. This virtual network closely resembles a traditional network that you operate in your own data center, with the benefits of using the scalable infrastructure of AWS. You can create a Amazon VPC that spans multiple Availability Zones. A *security group* acts as a virtual firewall that controls the traffic for one or more instances. When you launch an instance, you associate one or more security groups with the instance. You add rules to each security group that allow traffic to or from its associated instances.

An internet gateway is a Amazon VPC component that allows communication between instances in your Amazon VPC and the internet. A *route table* contains a set of rules, called *routes*, that are used to determine where network traffic is directed. Each subnet in a Amazon VPC must be associated with a route table; the route table controls routing for the subnet.

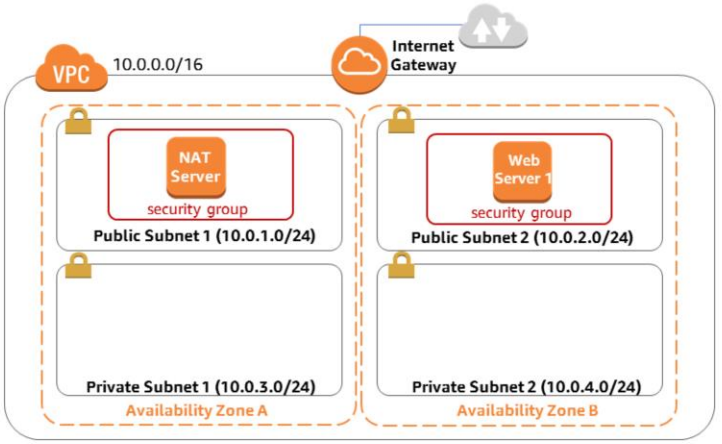
After completing this lab, you will be able to:
- Create an Amazon VPC
- Create subnets
- Configure a security group
- Launch an Amazon EC2 instance into an Amazon VPC

Duration: ~45 minutes

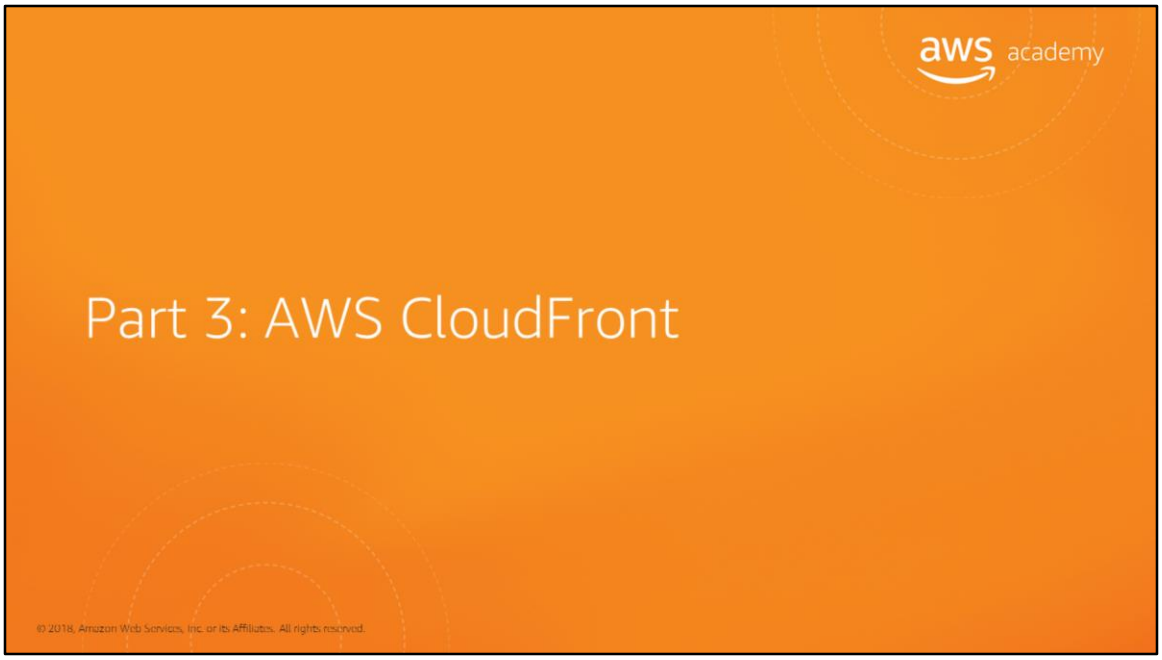
Lab 3: Tasks



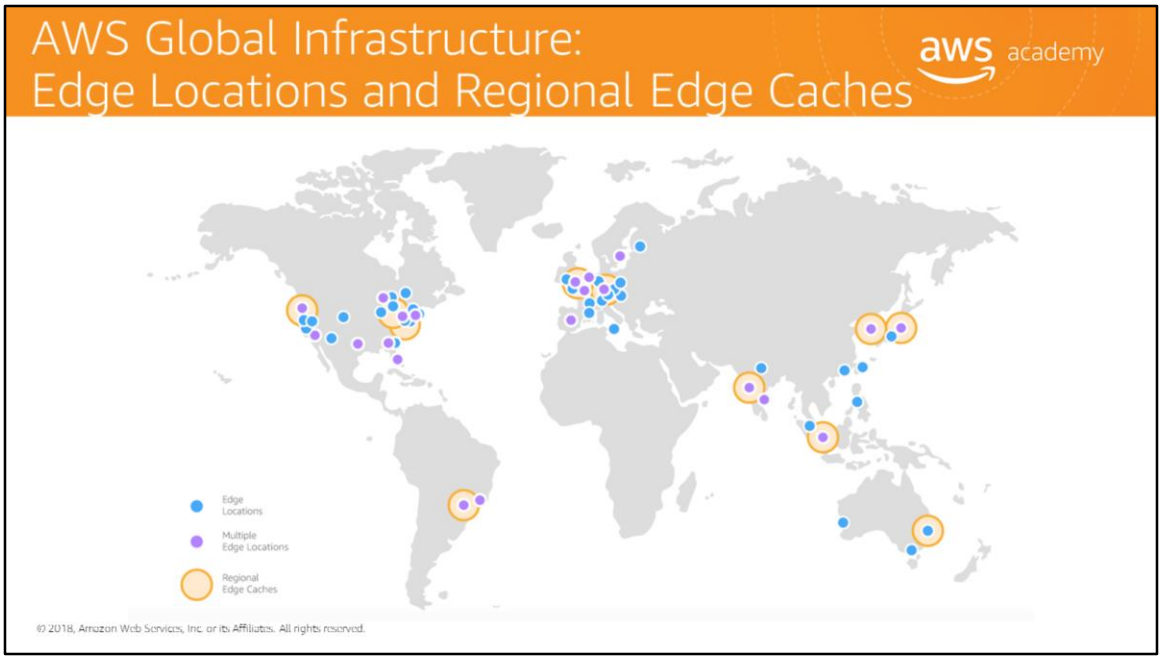
Lab 3: Final Product



~ 45 minutes

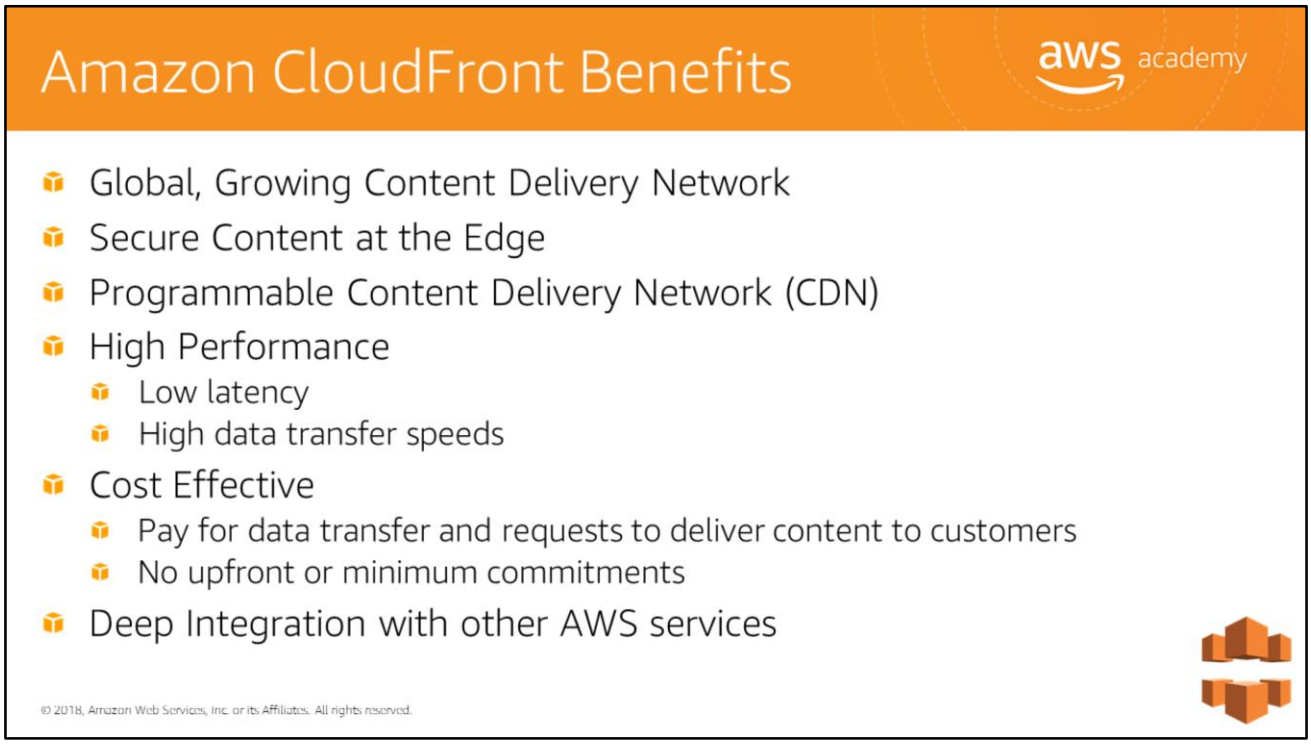


Amazon CloudFront allows you to scale out, save money and improved application performance. Amazon CloudFront is a global content delivery network (CDN) service that securely delivers data, videos, applications, and APIs to your viewers with low latency and high transfer speeds.



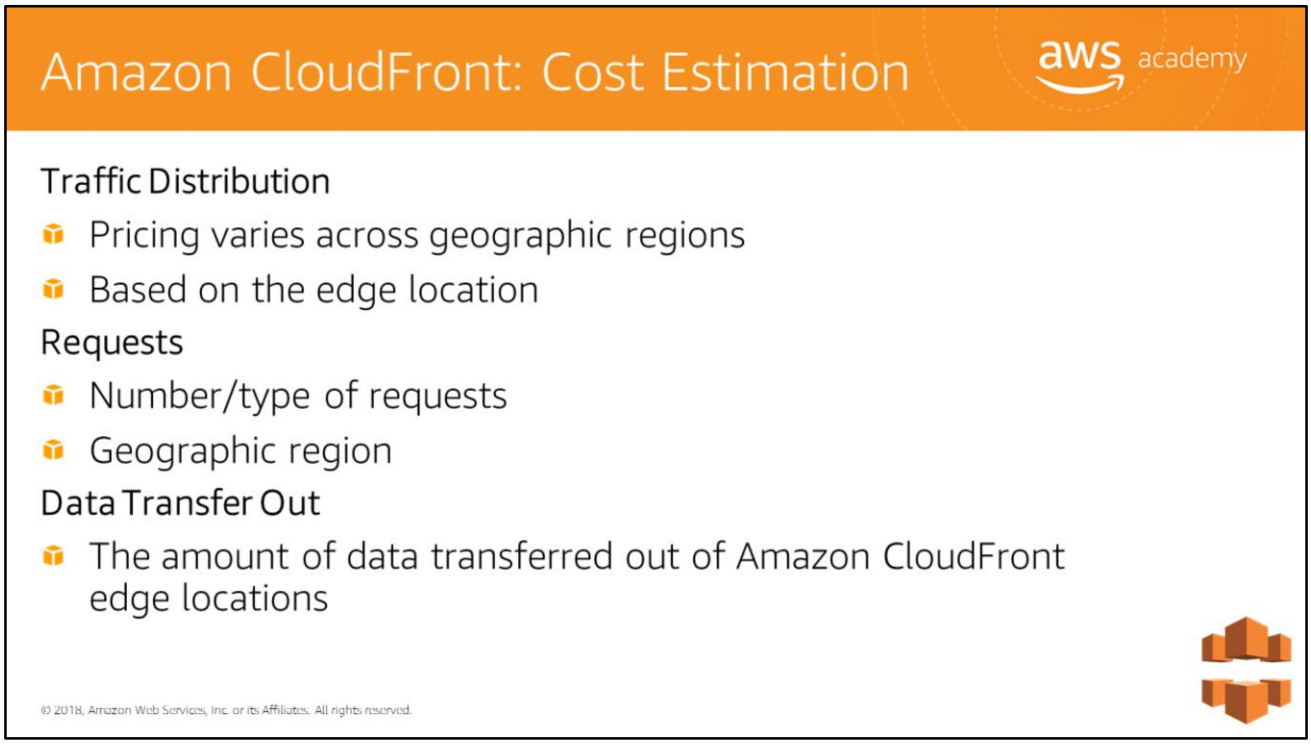
The earlier discussion about the AWS Global Infrastructure is related to this discussion about CloudFront. To deliver content to your users, Amazon CloudFront uses the global network of edge locations for content delivery.

By using CloudFront you can leverage multiple locations around the world to deliver your content allowing your users to interact with your application with lower latency.



Amazon CloudFront is a web service for content delivery or content delivery network (CDN). Amazon CloudFront provides the following benefits:

- A content delivery network built on the expanding global AWS infrastructure with a network of Edge locations to ensure that applications deliver high availability, scalability, and performance.
- A highly-secure Content Delivery Network (CDN) with both network and application level protection.
- It is programmable so you can run your code across AWS locations worldwide, allowing you to respond to your end users with the lowest latency.
- It is optimized for low latency and high data transfer speeds.
- It is cost effective because you pay only for the data transfer and requests used to deliver content to your customers. With CloudFront, there are no upfront payments or fixed platform fees, no long-term commitments, no premiums for dynamic content, and no requirements for professional services to get started. If you use AWS origins such as Amazon S3 or Elastic Load Balancing, you pay only for storage costs, not for any data transferred between these services and CloudFront.
- Deep integration with other Amazon Web Services to give you an easy way to distribute content to end users with low latency, high data transfer speeds, and no required minimum commitments.



When you begin to estimate the cost of Amazon CloudFront, you need to consider the following:

1. **Traffic Distribution** – Data transfer and request pricing vary across geographic regions, and pricing is based on the edge location through which your content is served.

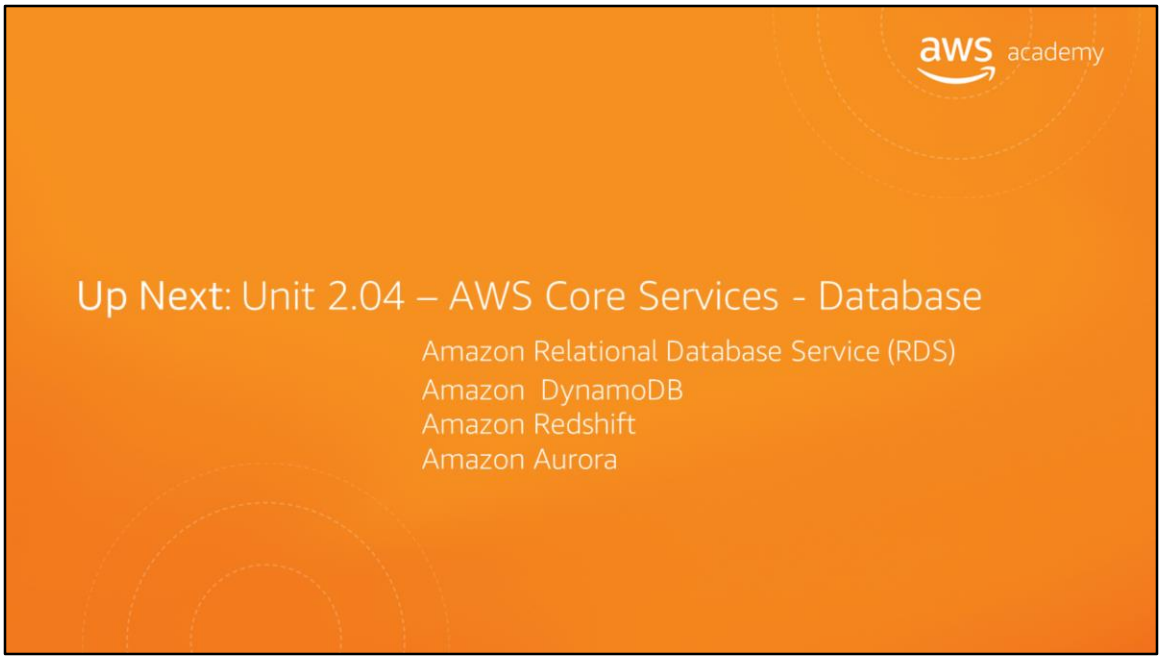
2. **Requests** – The number and type of requests (HTTP or HTTPS) made and the geographic region in which the requests are made.

3. **Data Transfer Out** – The amount of data transferred out of your Amazon CloudFront edge locations.

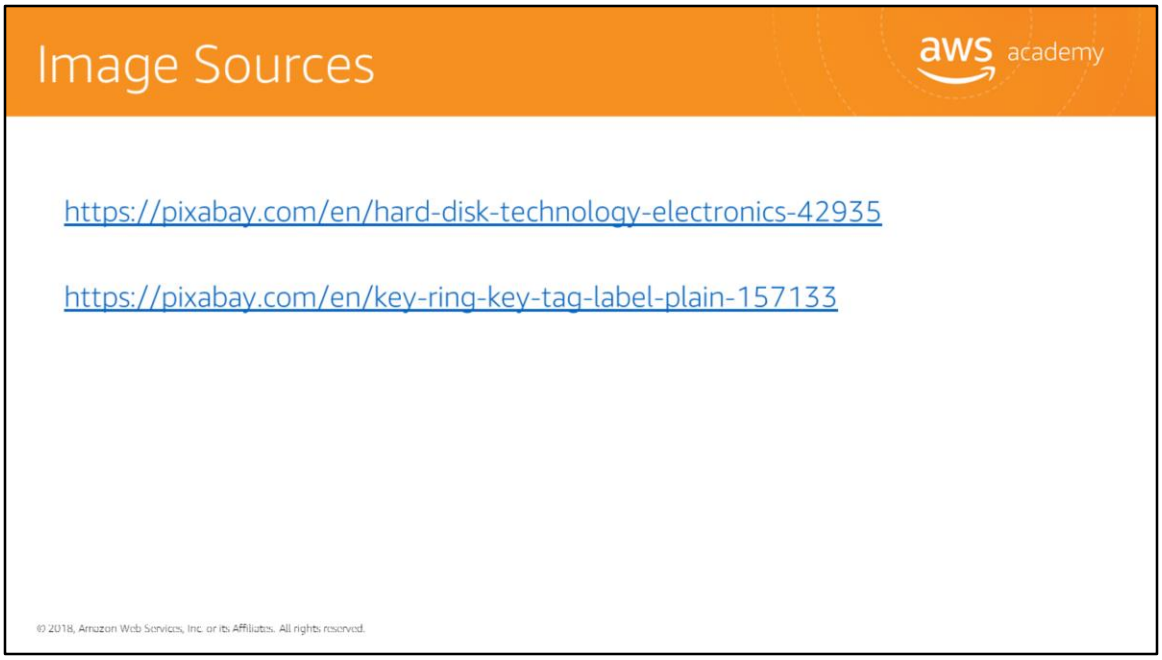
In review, we:

- Reviewed Amazon VPC including its required components and some optional components that are available.
- Discussed Amazon VPC security groups.
- Briefly introduced Amazon CloudFront.

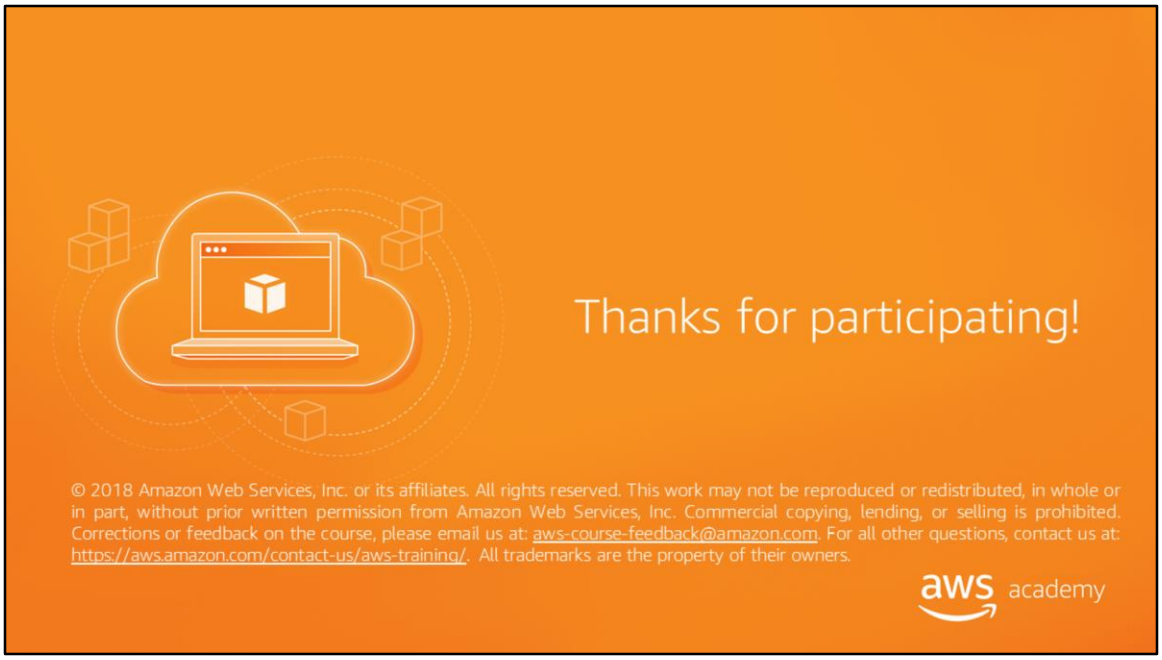
To finish this module, please complete the lab and the corresponding knowledge assessment.

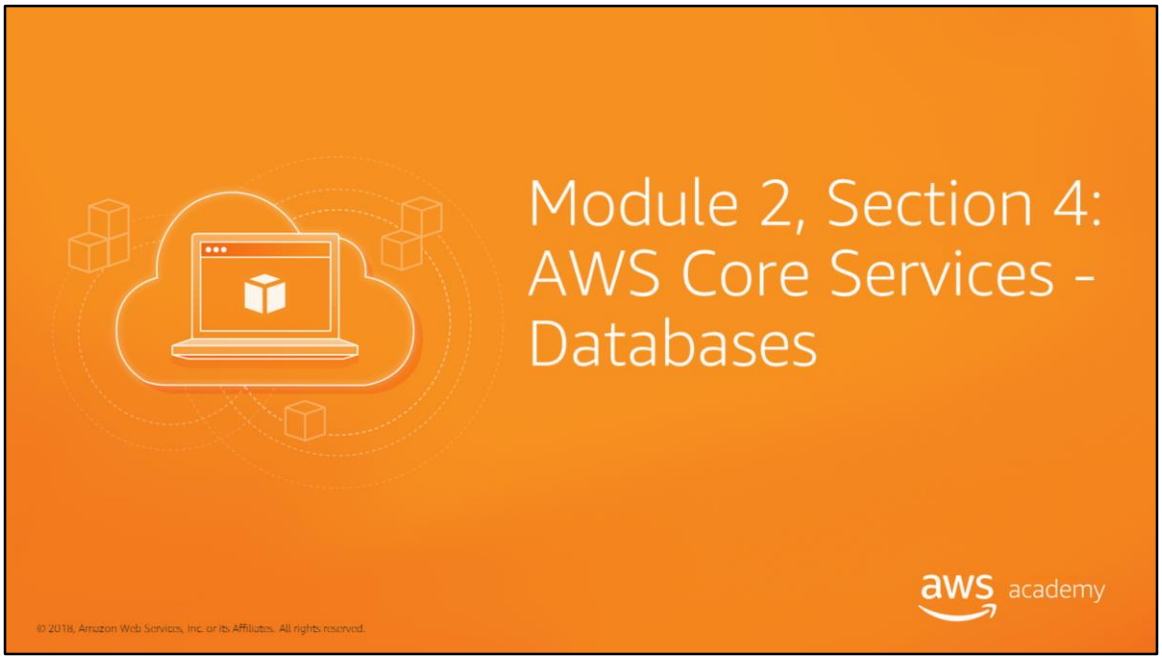


Now that we have a better understanding some of the storage services offered by AWS, in unit 2.04 we look at another AWS core service, database services.

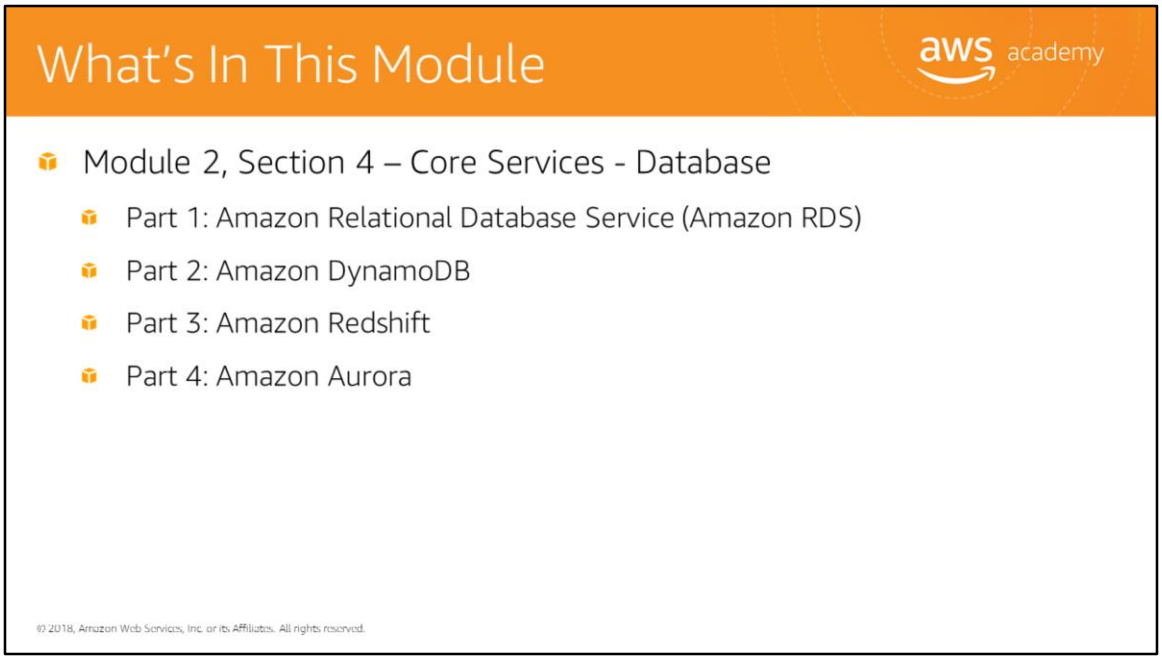


This slide contains attributions for any Creative Commons-licensed images used within this module.

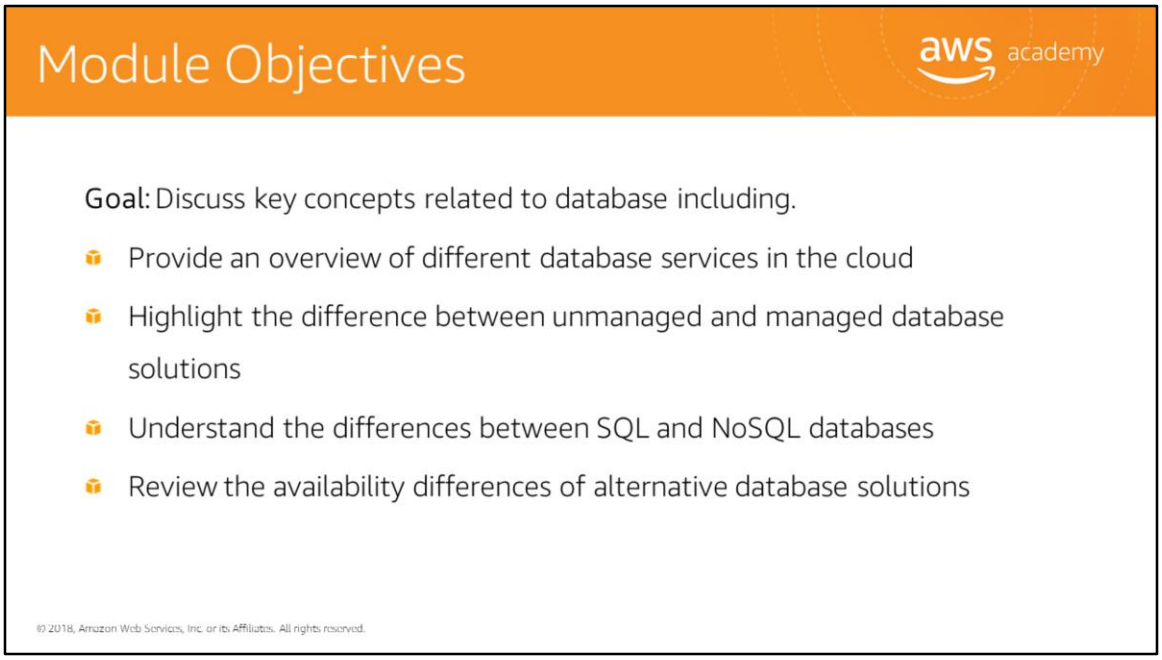




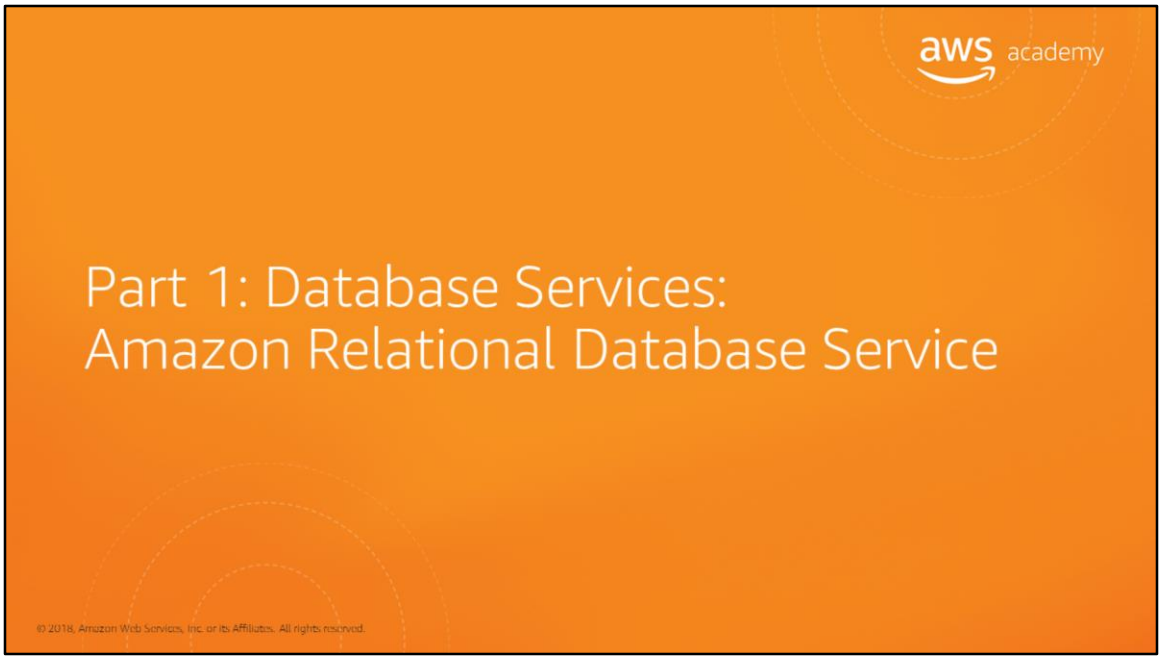
Welcome to Module 2, Section 4 – AWS Core Services - Database.



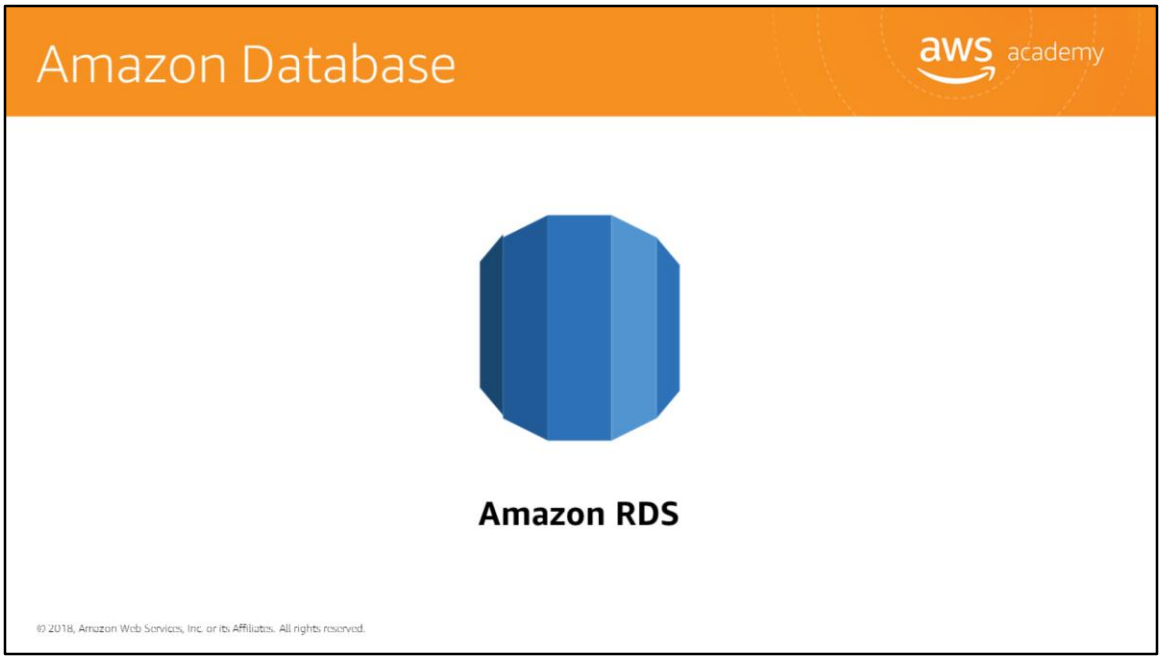
The business world is constantly changing and evolving. By accurately recording data, updating and tracking them on an efficient and regular basis, companies can leverage the immense potential from the insights obtained from their data. Database management systems are the crucial link for management of this data. Like other cloud services, cloud databases offer significant cost advantages over traditional database strategies.



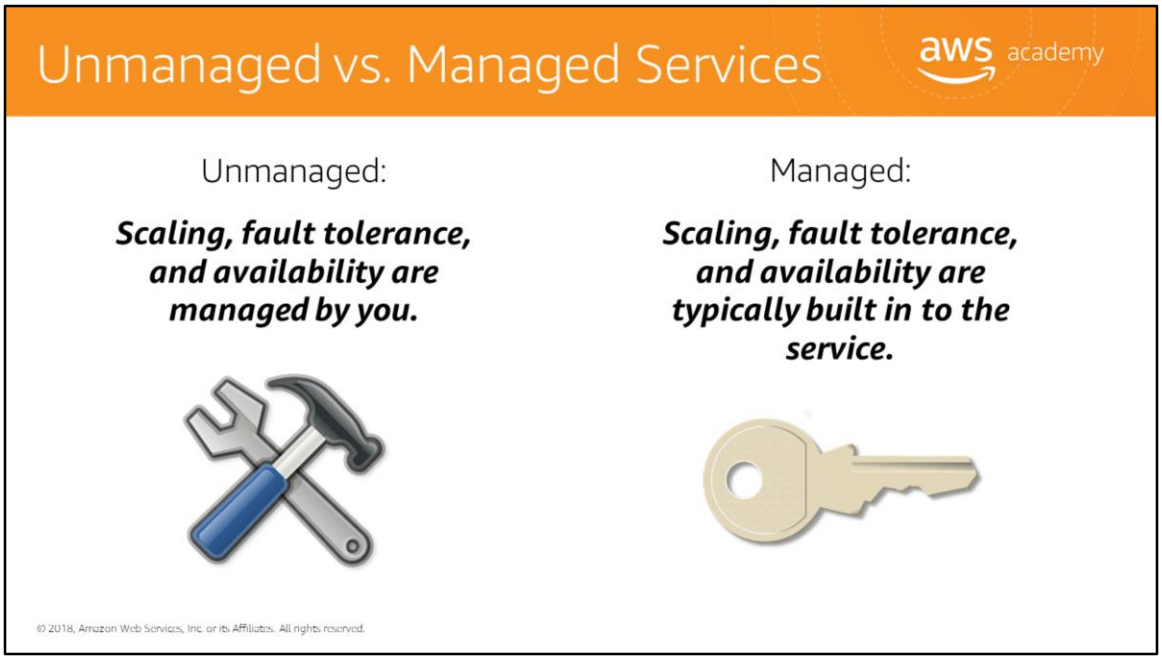
The goal of this module is to help you understand the database resources that are available to power your solution. We will also review the different service features that are available, so you can begin to understand how different choices impact things like solution availability.



Welcome to an introduction to the database services available on Amazon Web Services.



Let's start by taking a look at the differences between a managed and unmanaged service.



AWS solutions typically fall into one of two categories: unmanaged or managed.

Unmanaged services are typically provisioned in discrete portions as specified by you. Unmanaged services require the user to manage how the service responds to changes in load, errors, and situations where resources become unavailable. For instance, if you launch a web server on an Amazon EC2 instance, that web server will not scale to handle increased traffic load or replace unhealthy instances with healthy ones unless you specify it to use a scaling solution such as AWS Auto Scaling, because Amazon EC2 is an "unmanaged" solution. The benefit to using an unmanaged service is that you have more fine-tuned control over how your solution handles changes in load, errors, and situations where resources become unavailable.

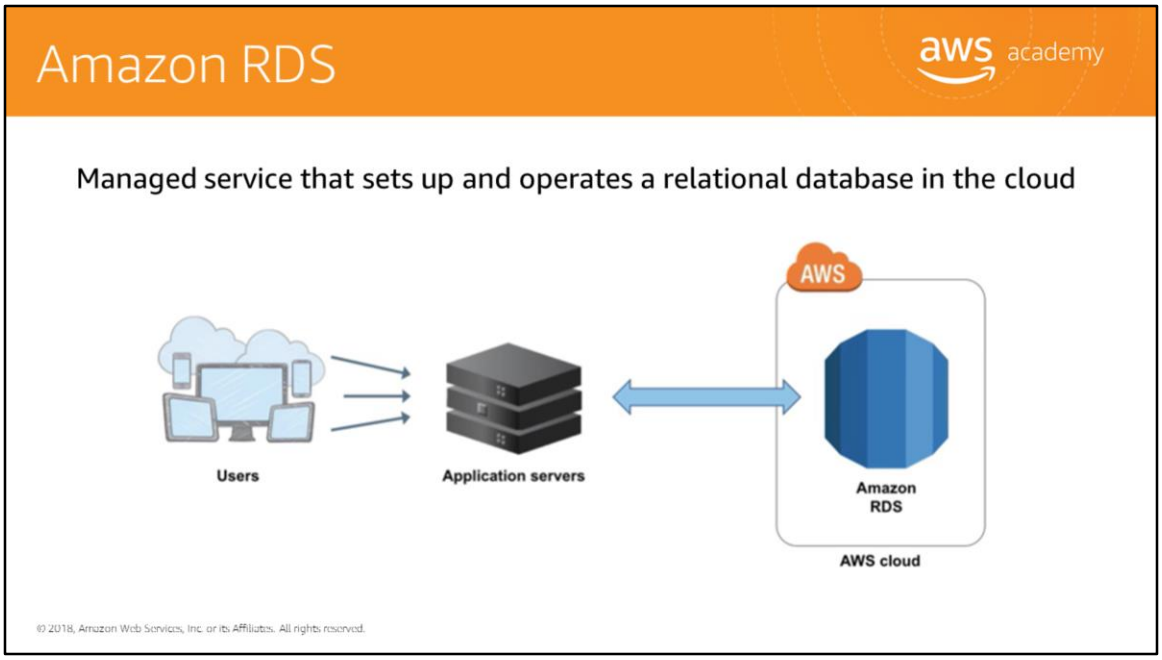
Managed services require the user to configure them (for example, creating an Amazon S3 bucket and setting permissions for it); however, managed services typically require far less configuration. For example, if you have a static website that you're hosting in a cloud-based storage solution such as Amazon S3 without a web server, those features (scaling, fault-tolerance, and availability) would be automatically handled internally by Amazon S3, because it is a managed solution.

Now, let’s look at the challenges of running an unmanaged, standalone relational database. Then we will see how Amazon RDS addresses these challenges.

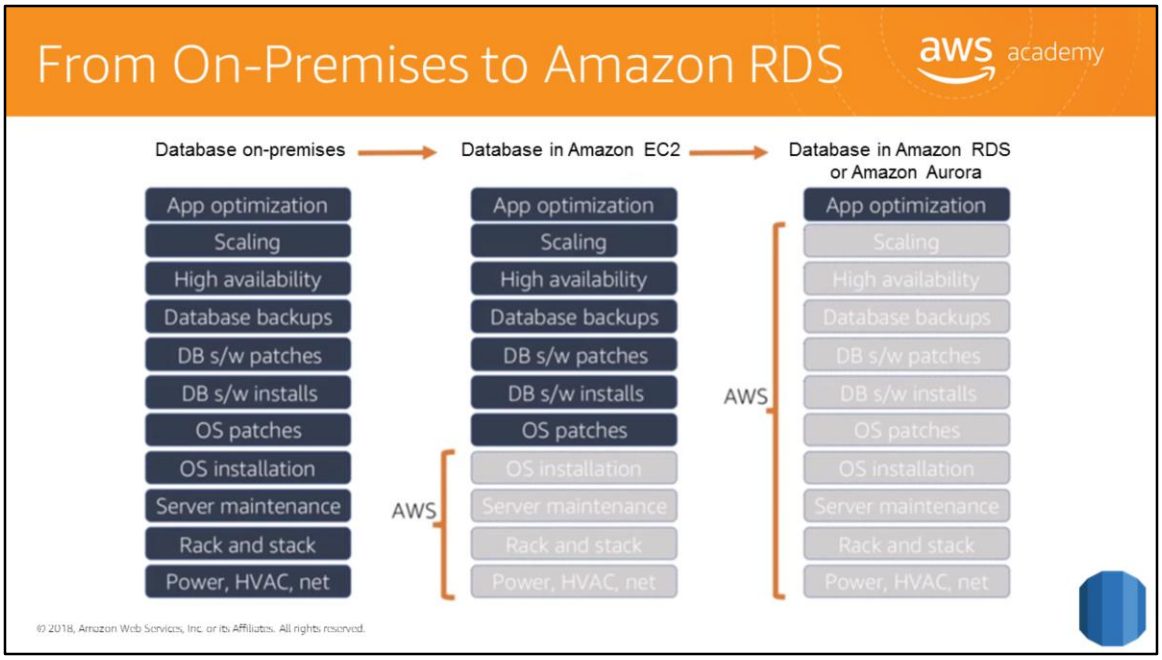
Challenges of Relational Databases

- Server maintenance and energy footprint
- Software installation and patches
- Database backups and high availability
- Limits on scalability
- Data security
- OS installation and patches

When running your own relational database, you are responsible for several administrative tasks, like server maintenance, software, installation and patching, backups, and ensuring high availability, scalability planning, data security, and OS installation and patching. All of these tasks take resources away from other items on your to-do list and require expertise in several areas.



To address those challenges, AWS provides a service that sets up, operates, and scales the relational database without any ongoing administration. Amazon RDS provides cost-efficient and resizable capacity while automating time-consuming administrative tasks. Amazon RDS frees you to focus on your application so you can give the applications the performance, high availability, security, and compatibility they need. With Amazon RDS, your primary focus becomes your data and optimizing your application.

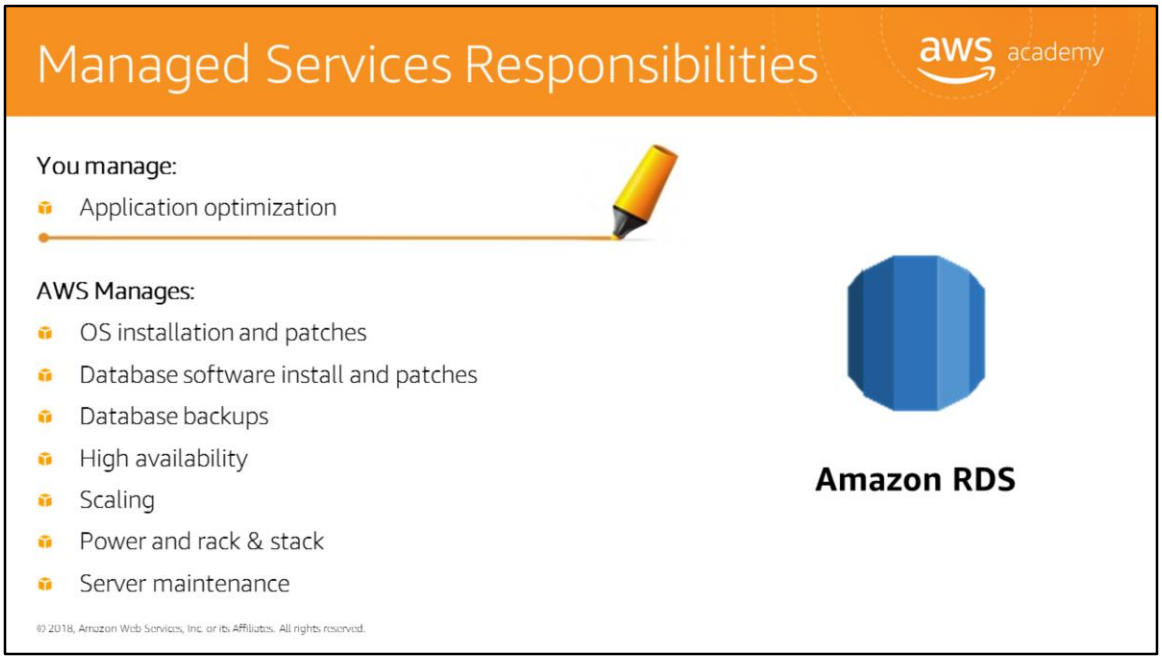


What do we mean by *managed services*? Let's take a look.

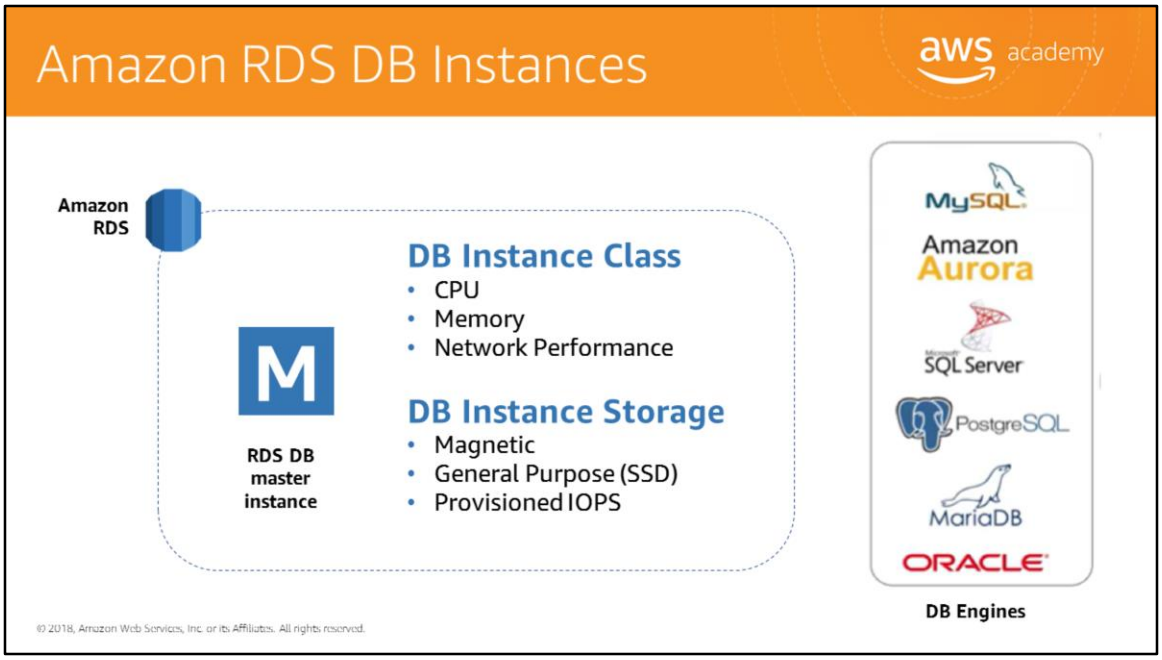
When your database is on-premises, the database administrator is responsible for everything from app and query optimization to standing up the hardware, patching the hardware, and setting up networking, power and HVAC.

If you move to a database running on an Amazon EC2 instance, you no longer have to manage the underlying hardware or handle data center operations. However, you're still responsible for patching the operating system and handling all software and backup operations.

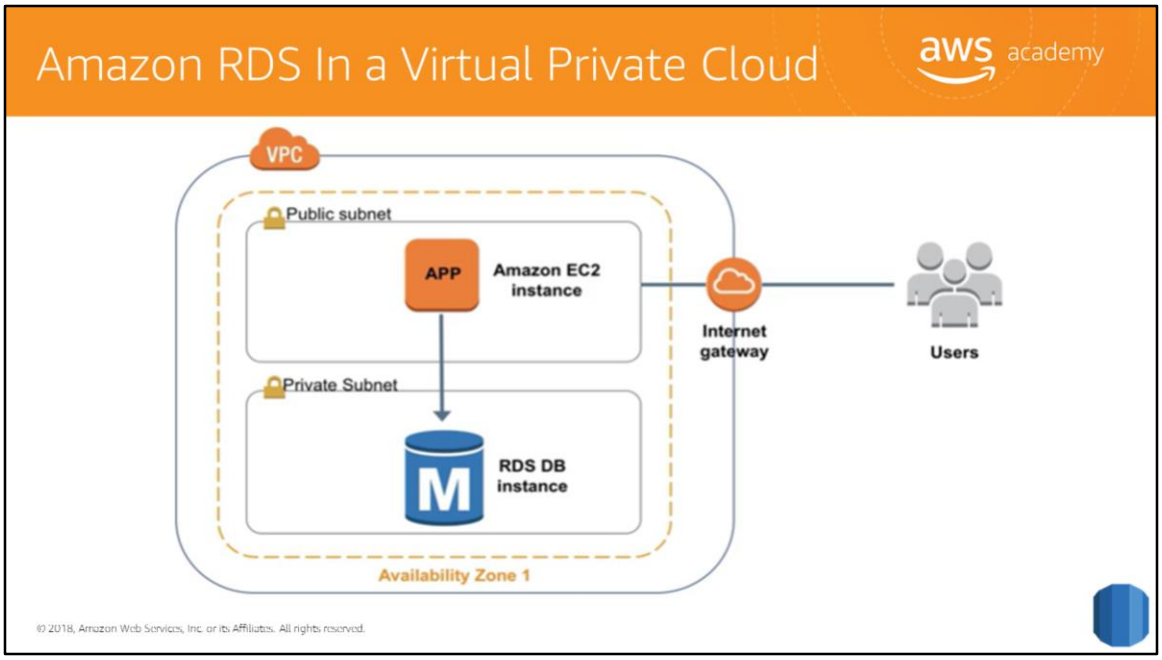
If you set up your database on Amazon RDS or Amazon Aurora, you free yourself from the administrative responsibilities. By moving to the cloud, you can automatically scale your database, enable high availability, manage backups and perform patching so that you can focus on what really matters most – optimizing your application.



Amazon RDS manages OS install and patching, database software installation and patching, automatic backups and high availability. Scaling resources, managing power and servers, and performing maintenance is also covered by AWS. Offloading these operations to the managed Amazon RDS service reduces your operational workload and the costs associated with your relational database. Now, let's go through a brief overview of the service and a few potential use cases.

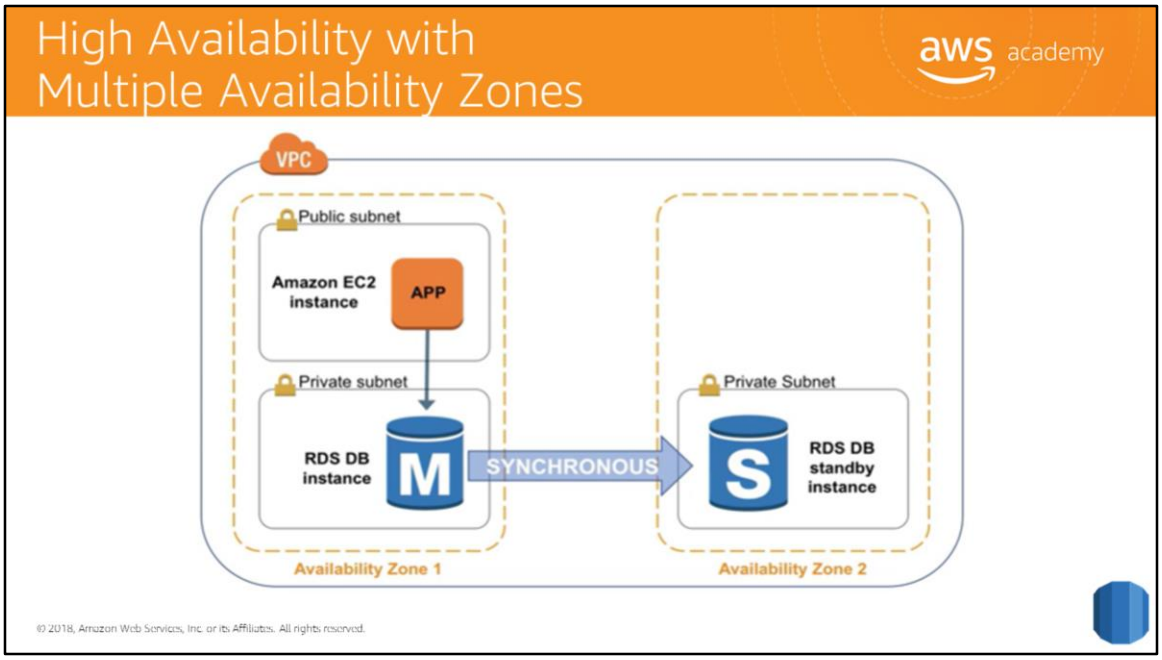


The basic building block of Amazon RDS is the database instance. A *database instance* is an isolated database environment that can contain multiple user-created databases and can be accessed by using the same tools and applications that you use with a standalone database instance. The resources found in a database instance are determined by its database instance class, and the type of storage is dictated by the type of disks. Database instances and storage differ in performance characteristics and price, allowing you to tailor your performance and cost to the needs of your database. When you choose to create a database instance, you first have to specify which database engine to run. Amazon RDS currently supports six databases: MySQL, Amazon Aurora, Microsoft Sequel Server, PostgreSQL, MariaDB, and Oracle.

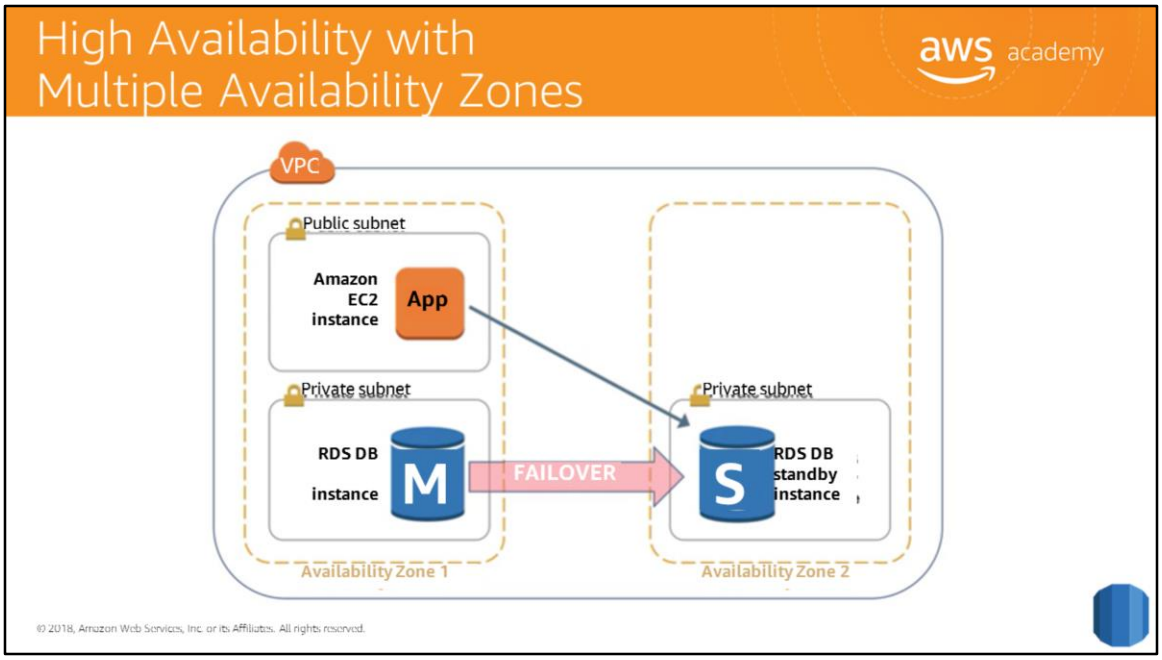


You can run an instance using Amazon Virtual Private Cloud (Amazon VPC). When you use an Amazon VPC, you have control over your virtual networking environment.

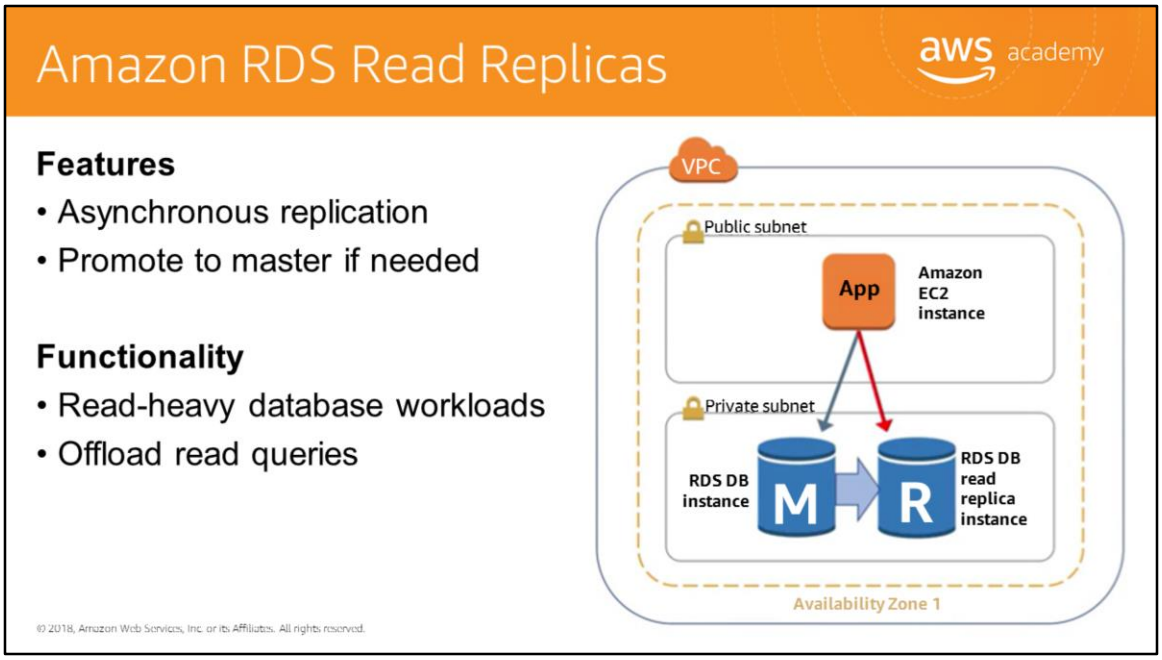
You can select your own IP address range, create subnets, and configure routing and access control lists. The basic functionality of Amazon RDS is the same whether or not it is running in an Amazon VPC. Usually the database instance is isolated in a private subnet and is only made directly accessible to indicated application instances. Subnets in an Amazon VPC are associated with a single Availability Zone, so when you select the subnet, you're also choosing the Availability Zone or physical location for your database instance.



One of the most powerful features of Amazon RDS is the ability to configure your database instance for high availability with a multi-AZ deployment. Once configured, Amazon RDS automatically generates a standby copy of the database instance in another Availability Zone within the same Amazon VPC. After seeding the database copy, transactions are synchronously replicated to the standby copy. Running a database instance with multiple Availability Zones can enhance availability during planned system maintenance and help protect your databases against database instance failure and Availability Zone disruption.

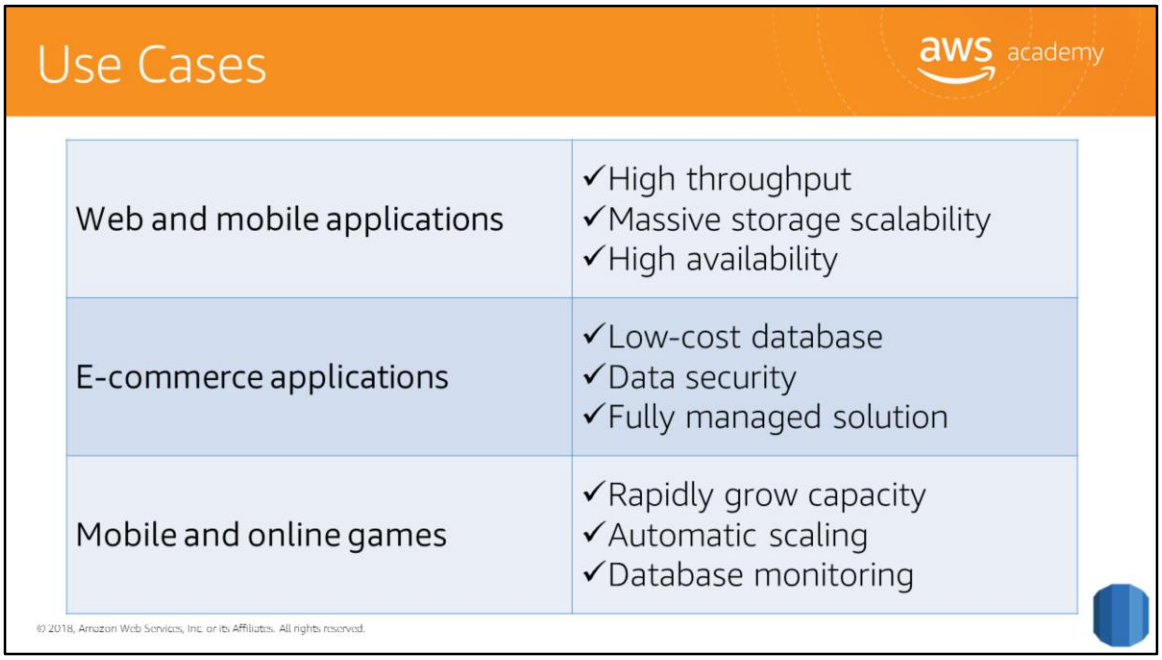


If the master database instance fails, Amazon RDS automatically brings the standby database instance online as the new master. Because of the synchronous replication, there should be no data loss. Because your applications reference the database by name using RDS DNS endpoint, you don't need to change anything in your application code to use the standby copy for failover.

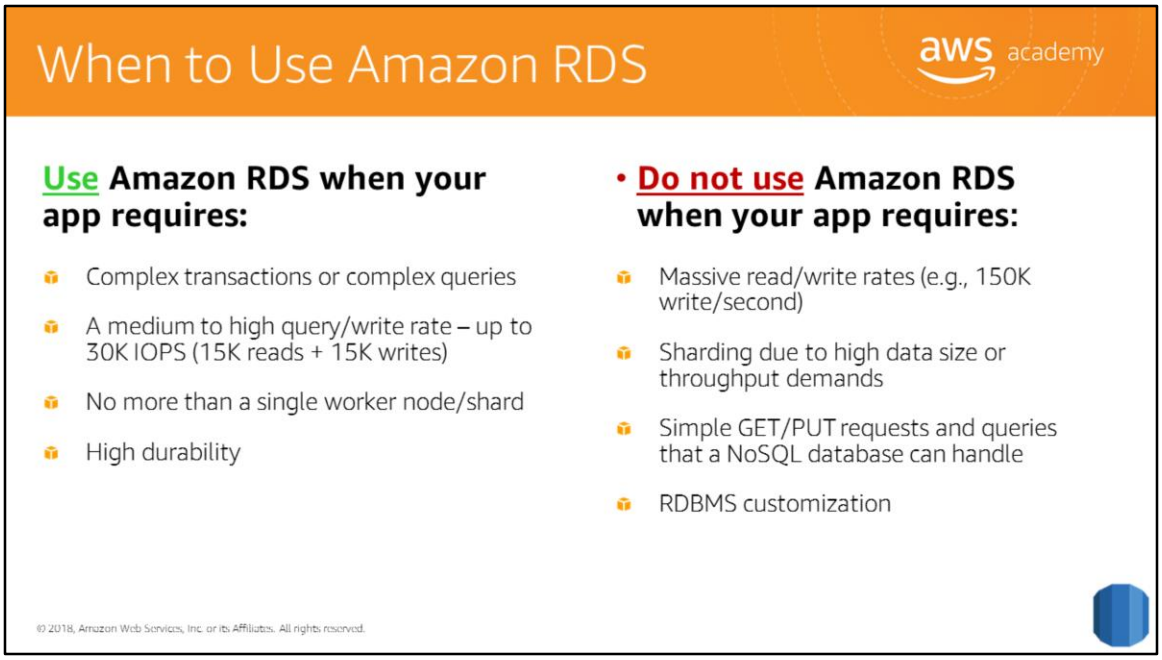


Amazon RDS also supports the creation of read replicas for MySQL, MariaDB, PostgreSQL, and Amazon Aurora. Updates made to the source database instance are asynchronously copied to the read replica instance. You can reduce the load on your source database instance by routing read queries from your applications to the read replica. Using read replicas, you can also scale out beyond the capacity constraints of a single database instance for read-heavy database workloads. Read replicas can also be promoted to become the master database instance, but due to the asynchronous replication, this requires manual action.

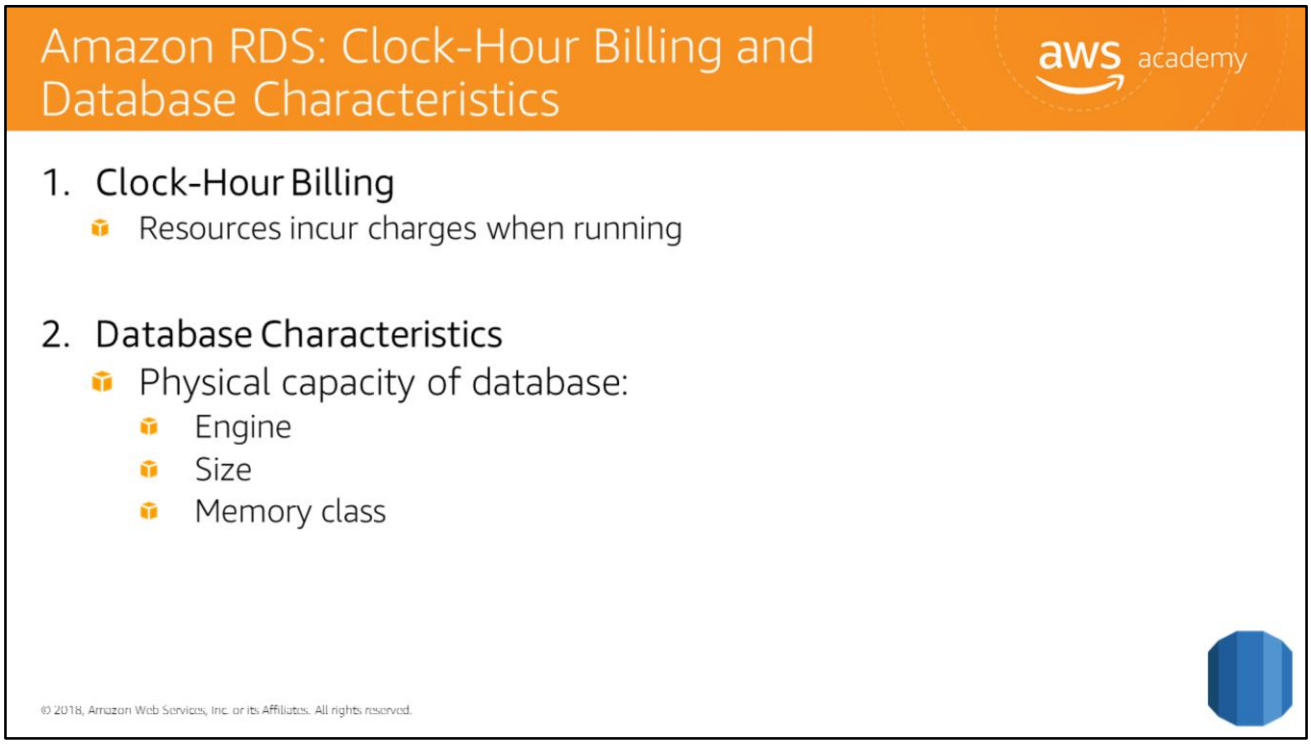
Read replicas can be created in a different region than the master database. This feature can help satisfy disaster recovery requirements or cut down on latency by directing reads to a read replica closer to the user.



Amazon RDS is ideal for web and mobile applications that need a database with high throughput, massive storage scalability, and high availability. Since Amazon RDS does not have any licensing constraints, it perfectly fits the variable usage pattern of these applications. When it comes to small and large e-commerce businesses, Amazon RDS provides a flexible, secured, and low-cost database solution for online sales and retailing. Mobile and online games require a database platform with high throughput and availability. Amazon RDS manages the database infrastructure, so game developers don't have to worry about provisioning, scaling, or monitoring database servers.



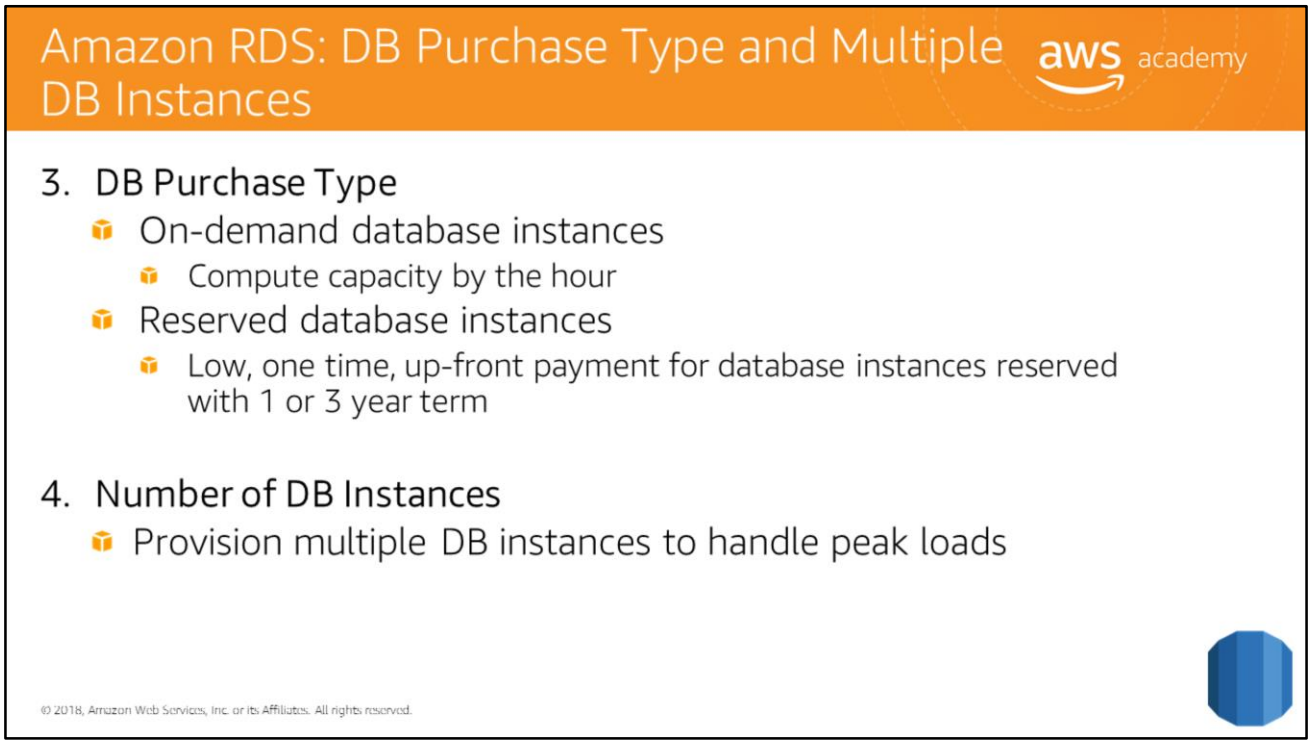
For circumstances where you should not use Amazon RDS, consider either using a NoSQL database solution such as DynamoDB or running your relational database engine on Amazon EC2 instances instead of Amazon RDS, which will provide you with more options for customizing your database.



When you begin to estimate the cost of Amazon RDS, you need to consider the following:

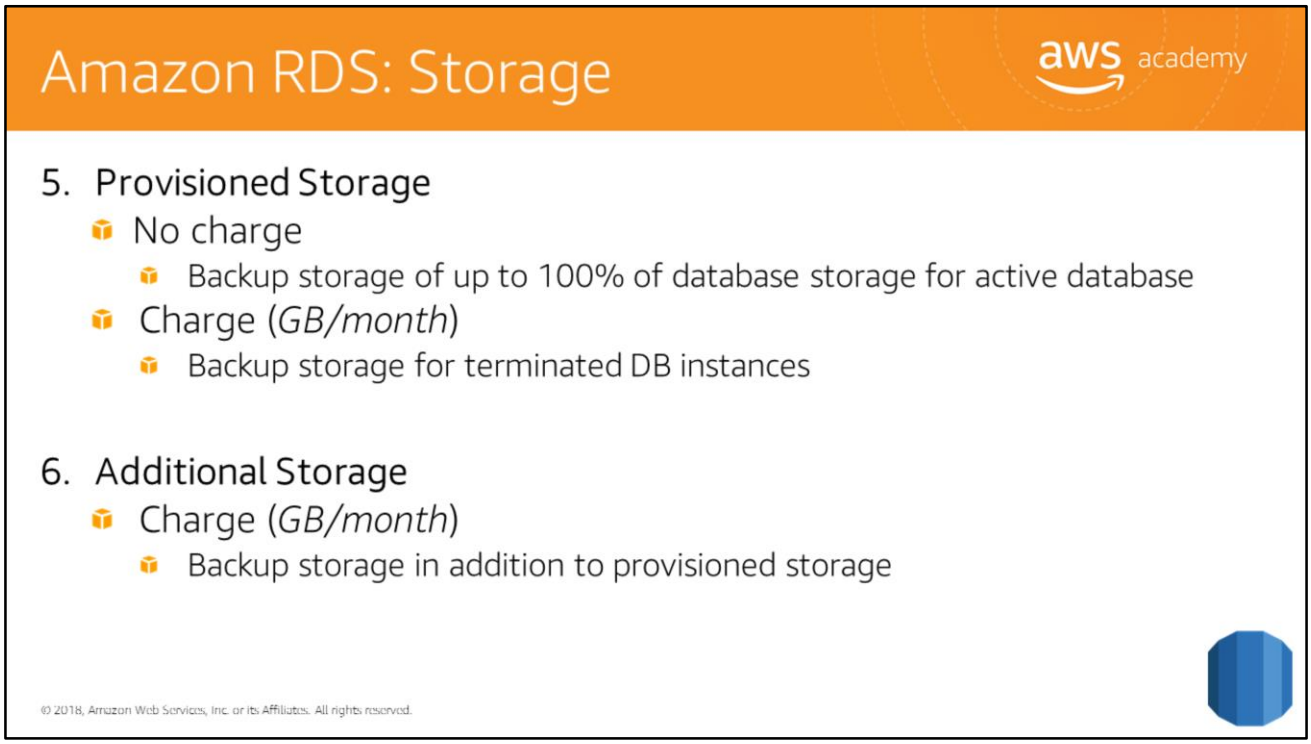
1. **Clock Hours of Server Time** – Resources incur charges when they are running. For example, from the time you launch a DB instance until you terminate the DB instance.

2. **Database Characteristics** – The physical capacity of the database you choose will affect how much you are charged. Database characteristics vary depending on the database engine, size, and memory class.



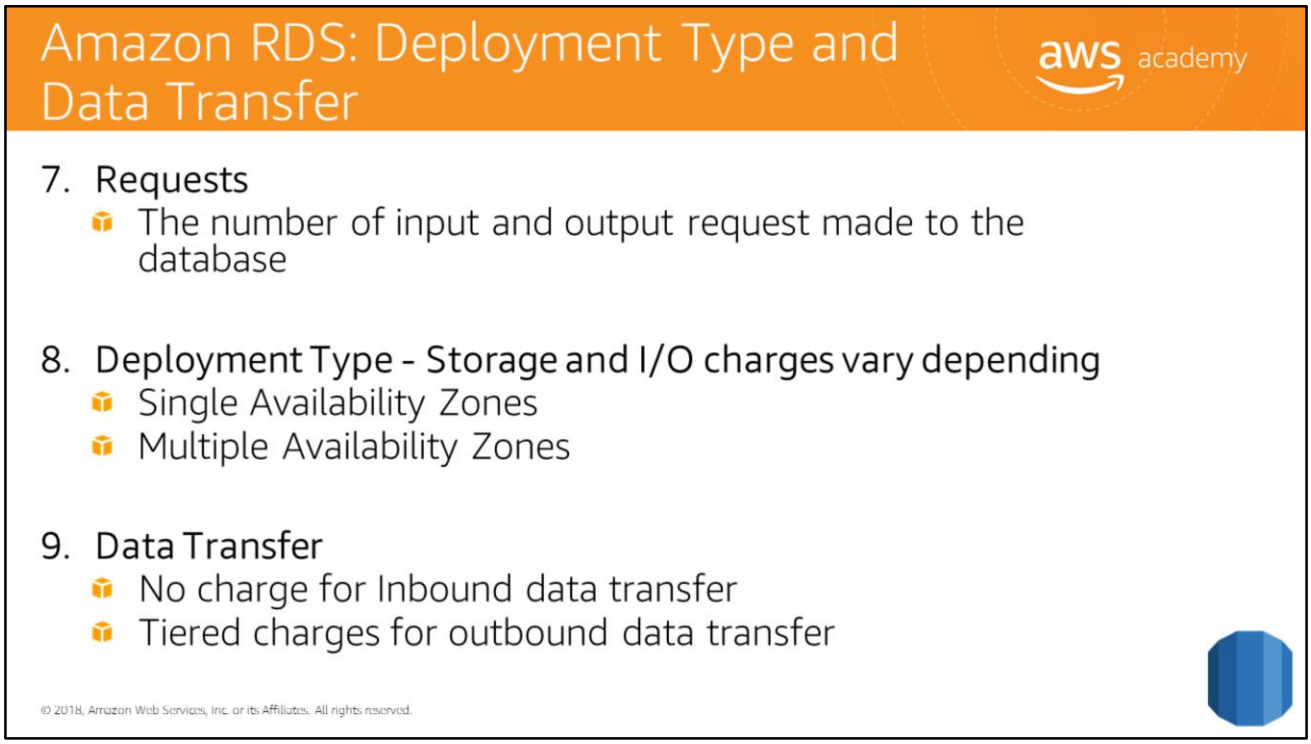
3. **Database Purchase Type** – When you use On-Demand DB Instances, you pay for compute capacity for each hour your DB Instance runs, with no required minimum commitments. With Reserved DB Instances, you can make a low, one-time, up-front payment for each DB Instance you wish to reserve for a 1-year or 3-year term.

4. **Number of Database Instances** – With Amazon RDS, you can provision multiple DB instances to handle peak loads.



5. **Provisioned Storage** – There is no additional charge for backup storage of up to 100% of your provisioned database storage for an active DB Instance. After the DB Instance is terminated, backup storage is billed per gigabyte per month

6. **Additional Storage** – The amount of backup storage in addition to the provisioned storage amount is billed per gigabyte per month.

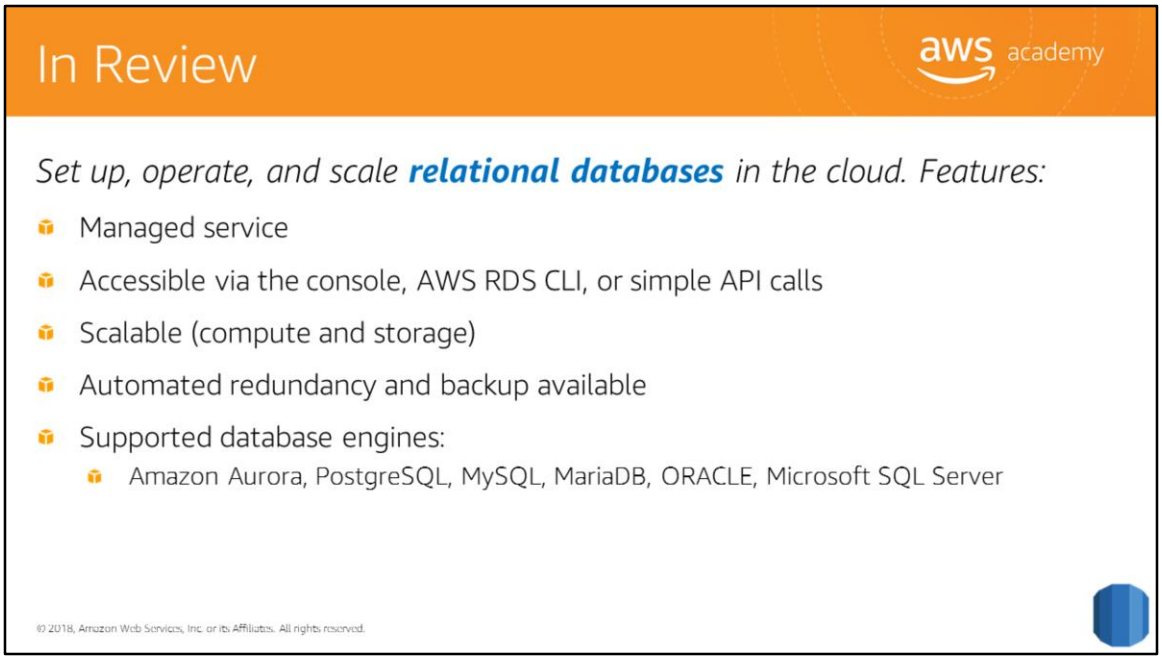


7. **Requests** – The number of input and output requests made to the database.

8. **Deployment Type** – You can deploy your DB instance to a single Availability Zone (analogous to a stand-alone data center) or multiple Availability Zones (analogous to secondary data center for enhanced availability and durability). Storage and I/O charges vary, depending on the number of Availability Zones you deploy to.

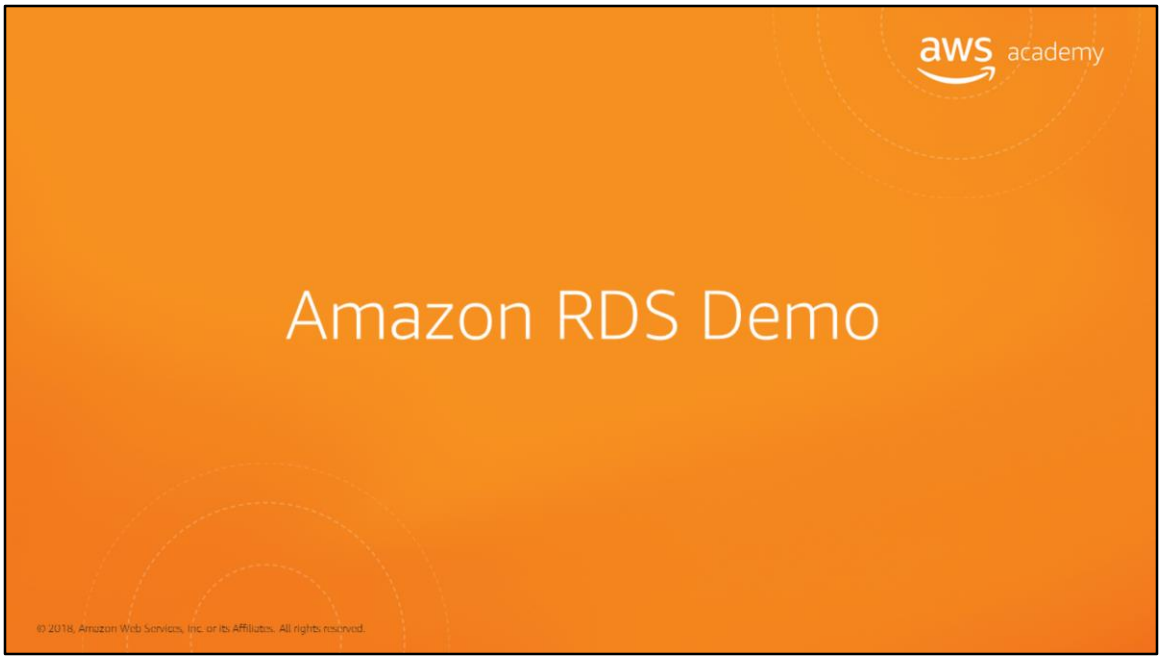
9. **Data Transfer** – Inbound data transfer is free, and outbound data transfer costs are tiered.

Depending on the needs for your application, it's possible to optimize your costs for Amazon RDS database instances by purchasing reserved Amazon RDS database instances. To purchase Reserved Instances, you make a low, one-time payment for each instance you want to reserve and in turn receive a significant discount on the hourly usage charge for that instance.



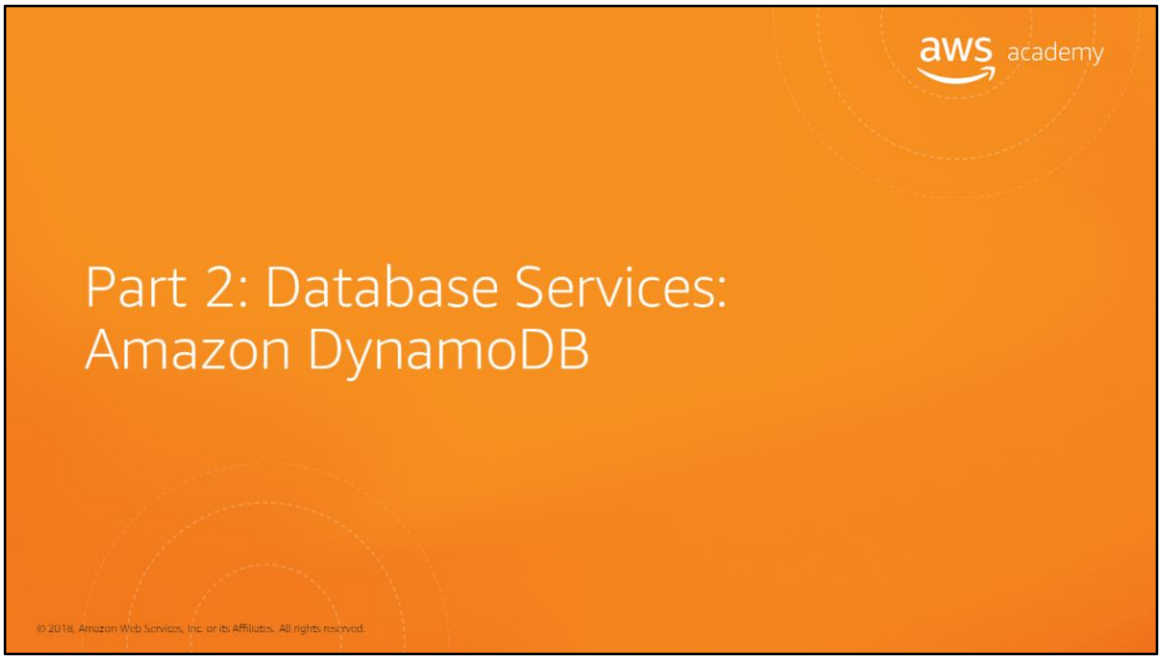
Amazon RDS is a web service that makes it easy to set up, operate, and scale a relational database in the cloud. It provides cost-efficient and resizable capacity while managing time-consuming database administration tasks, so you can focus on your applications and business.

Amazon RDS supports very demanding database applications. You can choose between two SSD-backed storage options: one optimized for high-performance OLTP applications, and the other for cost-effective general-purpose use. With Amazon RDS, you can scale your database's compute and storage resources with no downtime and use the console, the Amazon RDS CLI, or simple API calls to manage the service. Amazon RDS runs on the same highly reliable infrastructure used by other Amazon web services. It also lets you run your database instances and Amazon VPC, which provides you with control and security.

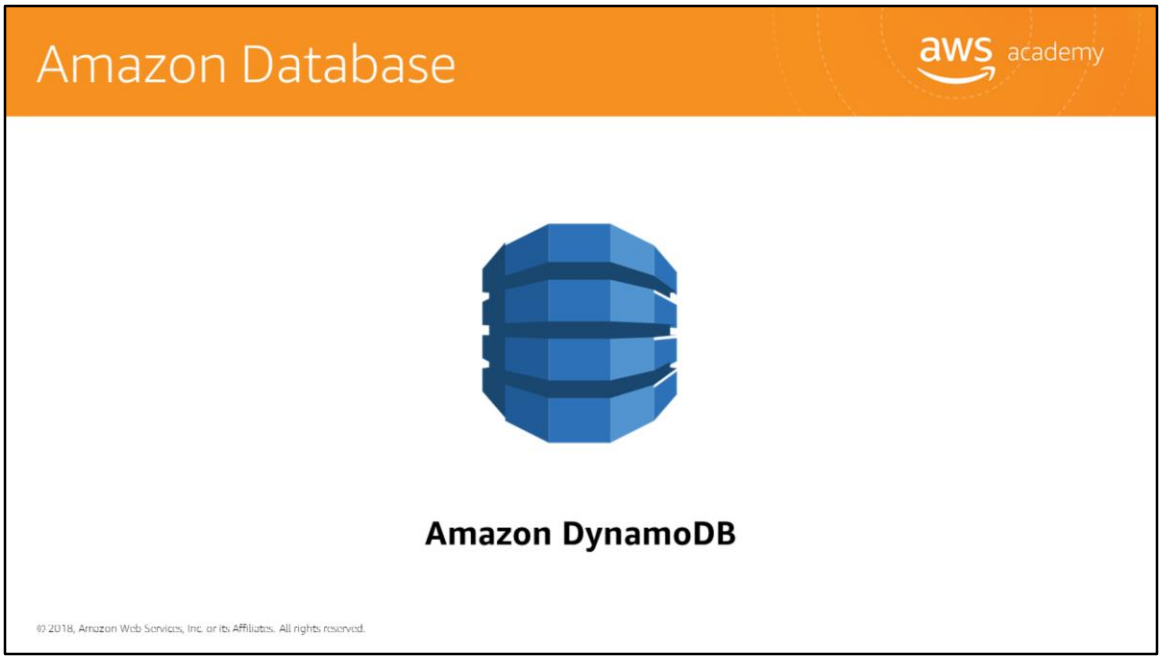


Please review the Amazon RDS demonstration: M2_S4_rds v2.0.mp4.

This video demonstration can be found in the learning management system.



Welcome to an introduction to Amazon DynamoDB.

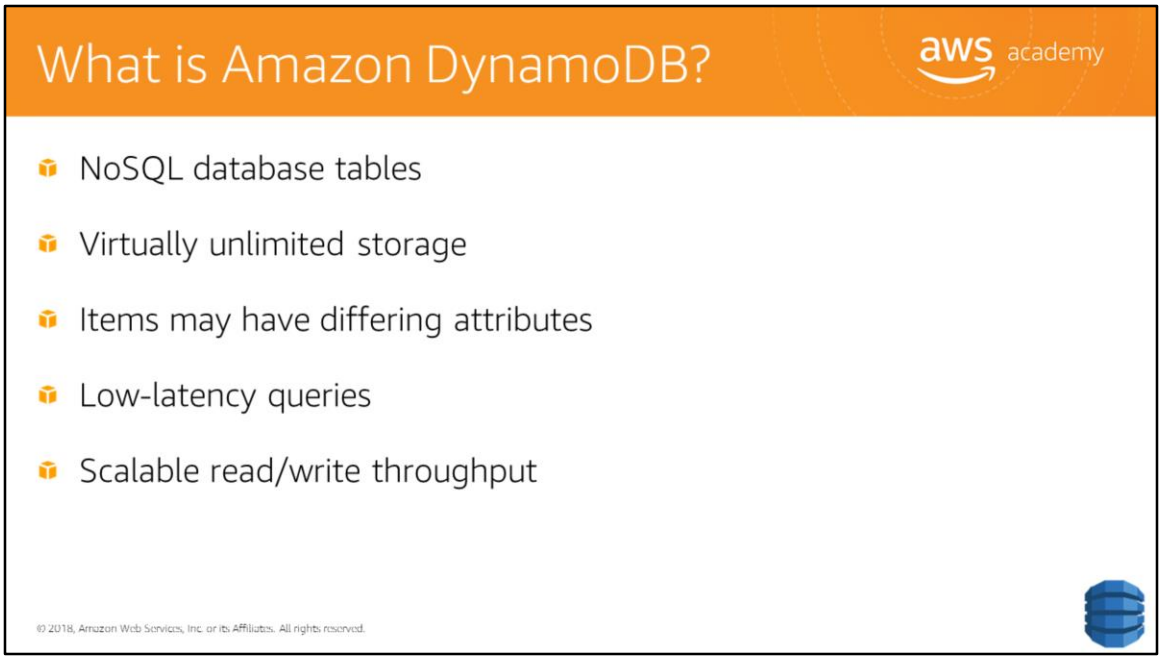


DynamoDB is a fast and flexible NoSQL database service for all applications that need consistent, single-digit millisecond latency at any scale.

With DynamoDB, we transition from relational databases to non-relational databases. Let's look at the differences:

- A **relational database** (RDB) works with structured data organized by tables, records and columns. RDBs establish a well-defined relationship between database tables. RDBs use Structured Query Language (SQL), which is a standard user application that provides an easy programming interface for database interaction. Relational databases do not scale out well horizontally, have problems working with semi-structured data, and normalized data require lots of joins.
- A **non-relational database** is any database that does not follow the relational model provided by traditional relational database management systems. Non-relational databases have grown in popularity because they were designed to overcome the limitations of relational databases in dealing with the demands of big data. Non-relational databases scale out horizontally and work with unstructured and semi-structured data.

Let's take a look at what DynamoDB has to offer.



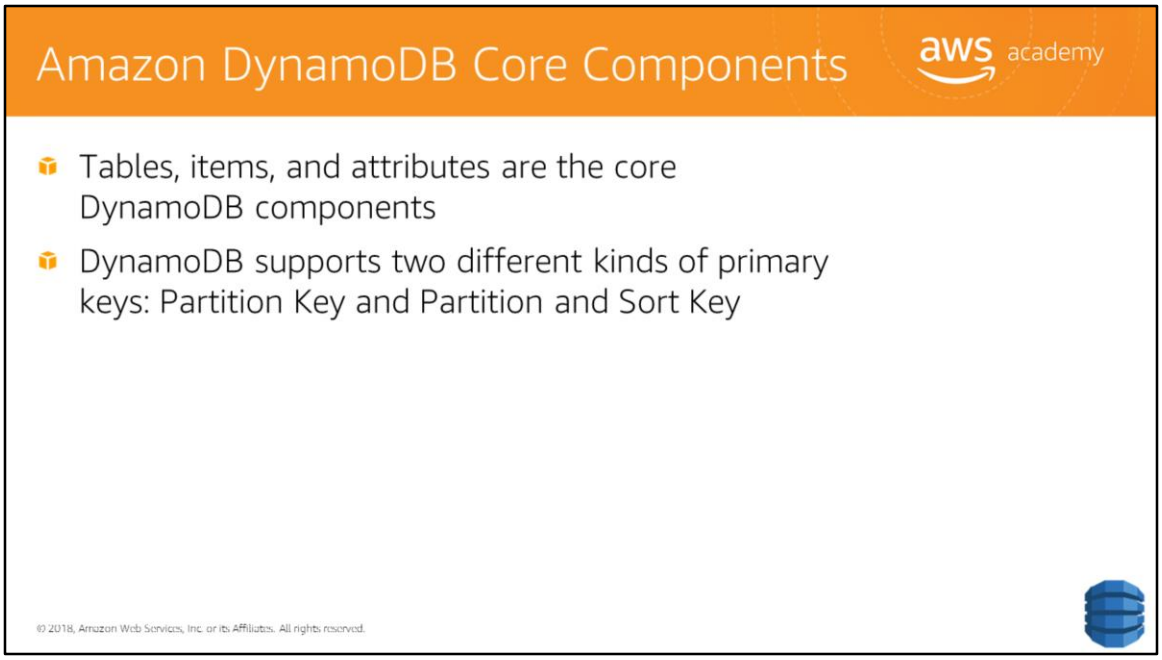
DynamoDB is a fully managed NoSQL database service. Amazon manages all of the underlying data infrastructure for this service and redundantly stores data across multiple facilities within a native US region as part of the fault-tolerant architecture. With DynamoDB, you can create tables and items. You can add items to a table. The surface automatically partitions your data and has table storage to meet the workload requirements. There is no practical limit on the number of items you can store in a table. For instance, some customers have production tables that contain billions of items.

One of the benefits of a NoSQL database is that items in the same table may have different attributes. This gives you the flexibility to add attributes as your application evolves. You can have newer format items stored side by side with older format items in the same table without needing to perform schema migrations.

As your application becomes more popular and as users continue to interact with it, your storage can grow with your application's needs. All of the data in DynamoDB is stored on solid-state drives and its simple query language allows for consistent low-latency query performance. In addition to scaling storage, DynamoDB also allows you to provision the amount of read or write throughput you need for your table. As the number of application users grow, DynamoDB tables can be scaled to handle the increased numbers of read and write requests with manual provisioning. Alternatively, you can enable automatic scaling so

that DynamoDB monitors the load on the table and automatically increases or decreases the provision throughput.

Some additional key differentiating features include global tables that enable you to automatically replicate across your choice of AWS regions, encryption at rest, and item TTL.



Tables, items, and attributes are the core DynamoDB components:
Table: a collection of data
Items: a group of attributes that is uniquely identifiable among all of the other items
Attributes: a fundamental data element, something that does not need to be broken down any further.

DynamoDB supports two different kinds of primary keys:
Partition Key: A simple primary key, composed of one attribute called the partition key
Partition Key and Sort Key: Also known as composite primary key which is composed of two attributes.

For more information on how DynamoDB works see https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/HowItWorks.CoreComponents.html#HowItWorks.CoreComponents.TablesItemsAttributes

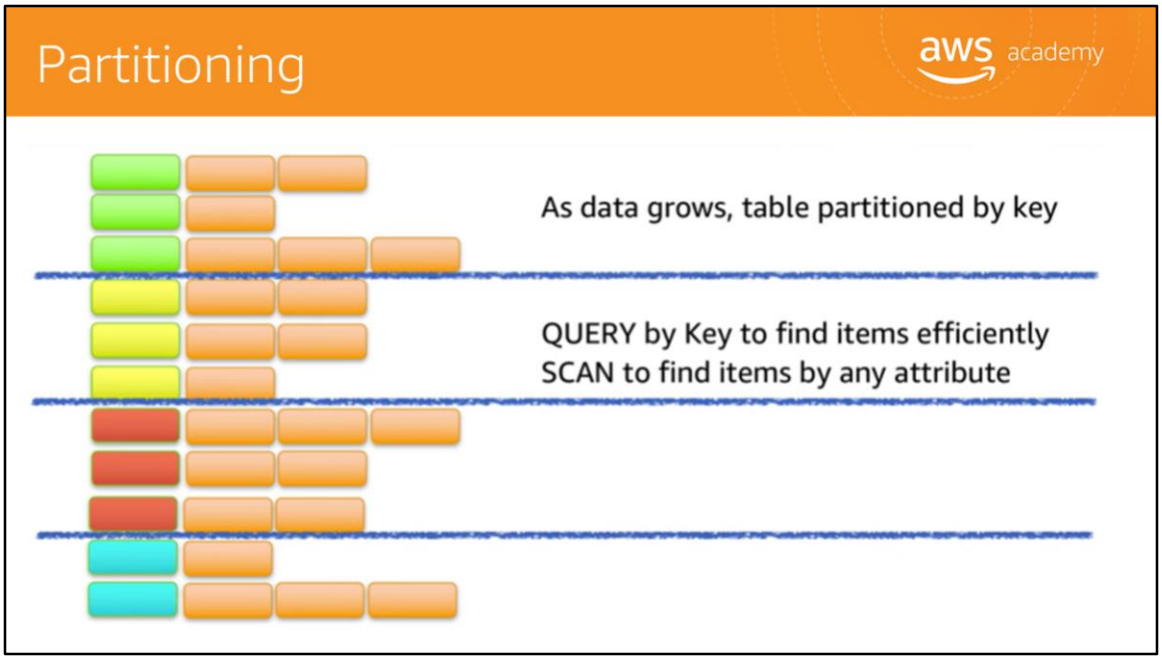
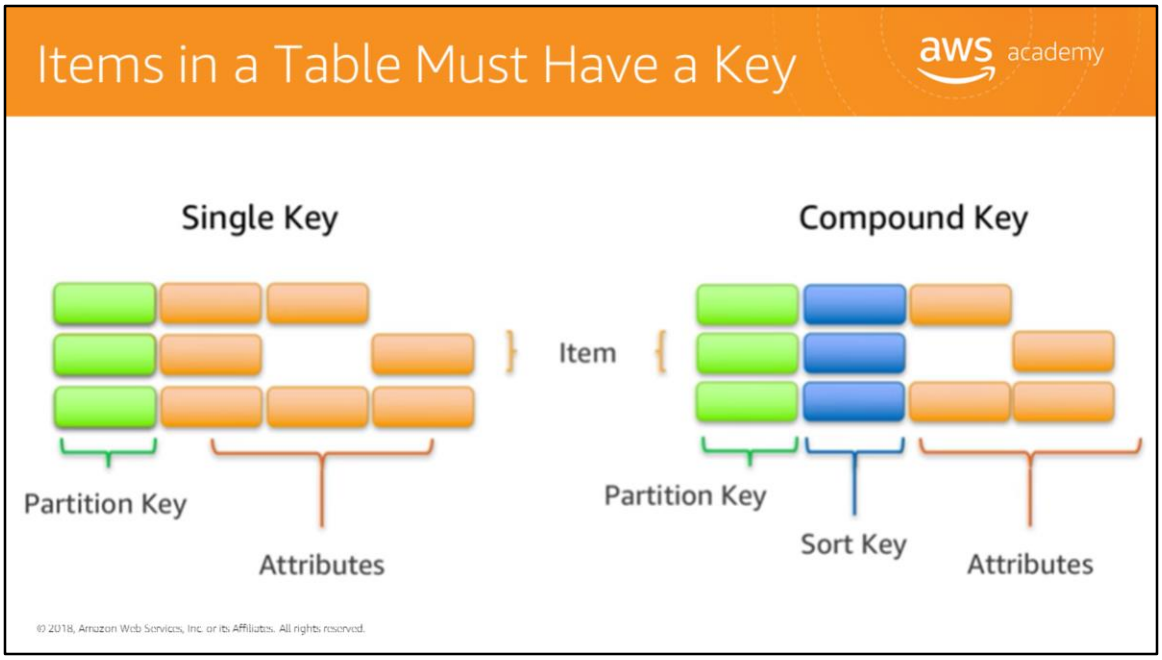


Table data is partitioned and indexed by primary key.

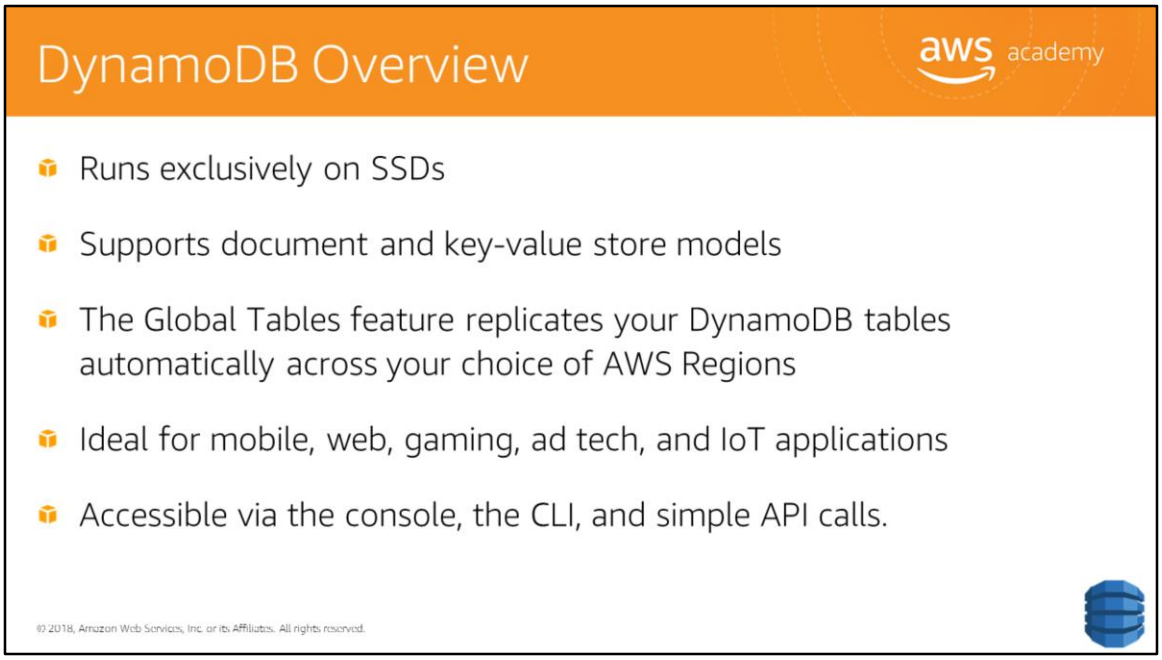
There are two different ways of retrieving data from a DynamoDB table.

In the first method, query operation takes advantage of the partitioning to effectively and locate items by using the primary key.

The second method is via a scan, which will allow you to locate items in the table by matching conditions on non-key attributes. The second method gives you flexibility to locate items by other attributes. However, the operation is less efficient, as DynamoDB will scan through all the items in the table to find the ones that match your criteria.

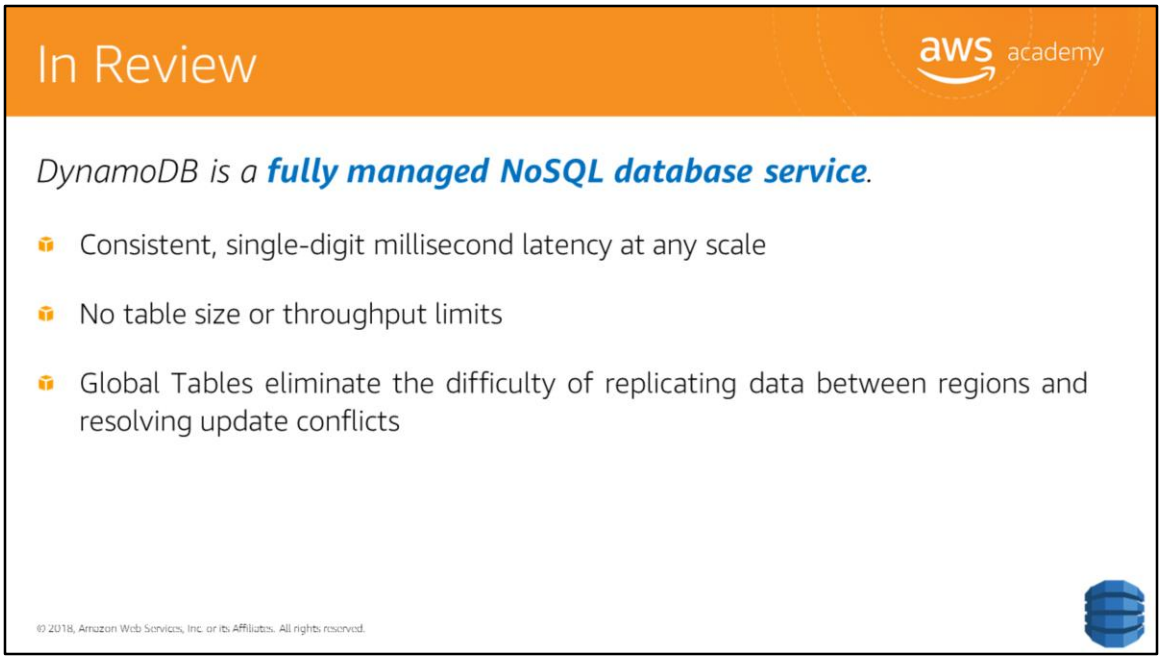


To take full advantage of query operations and Dynamo DB, it's important to think about the key you used to uniquely identify items in the DynamoDB table. You can set up a simple primary key based on a single attribute of the data values with a uniform distribution, such as the GUID or other random identifiers. For example, if you were to model a table with products, you could use some attributes such as the product ID. Alternatively, you can specify a compound key, which will be composed of a partition key and a secondary key. In this example, if I was to have a table with books, I might use the combination of author and title to uniquely identify table items. This could be useful if you expect to frequently look at books by author, since then you could use query.

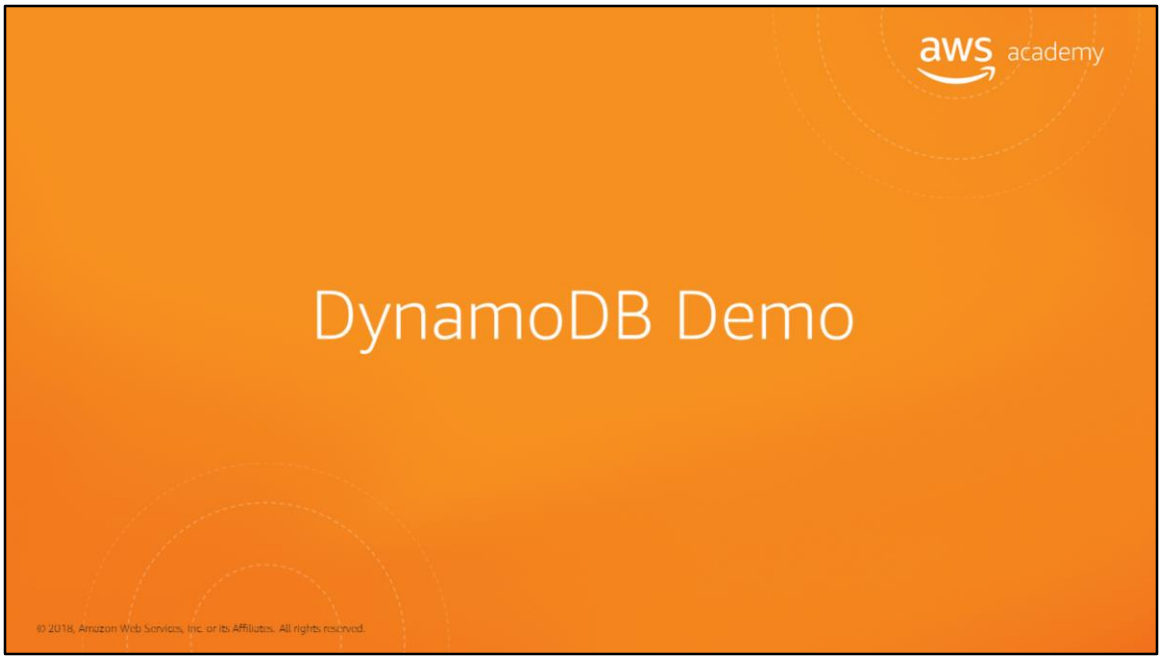


The ability to scale your tables in terms of both storage and provision throughput makes DynamoDB a good fit for structured data from the web, mobile, and IoT applications. For instance, you may have a large number of clients continuously generating data and making large numbers of requests per second. In this case, the throughput scaling of DynamoDB allows consistent performance for your clients. DynamoDB is also used in latency-sensitive applications. The predictable query performance, even in large tables, makes it useful for cases where variable latency could cause significant impact to the user experience or to business goals such as ad tech or gaming.

The DynamoDB Global Tables feature eliminates the difficult work of replicating data between regions and resolving update conflicts. It replicates your DynamoDB tables automatically across your choice of AWS Regions. Global Tables can help applications stay available and performant for business continuity.

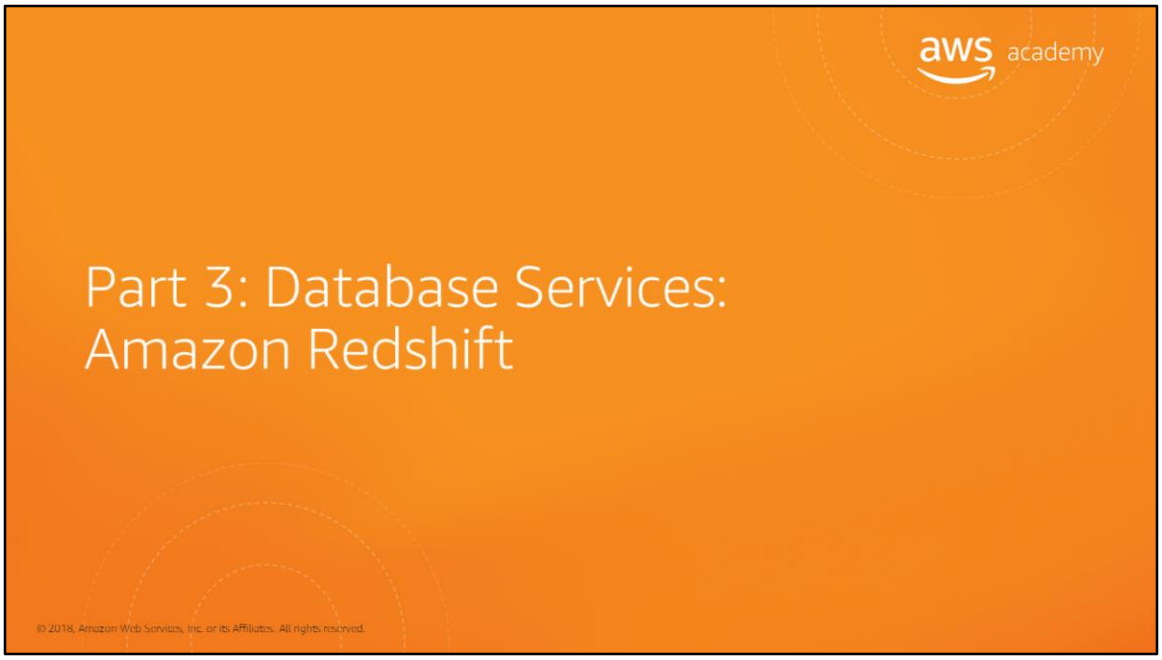


DynamoDB is a great solution for a database that must be highly performant but does not require complex operations on the data to make use of it. If you're running simple GET/PUT requests on your data, consider using DynamoDB instead of a relational database.

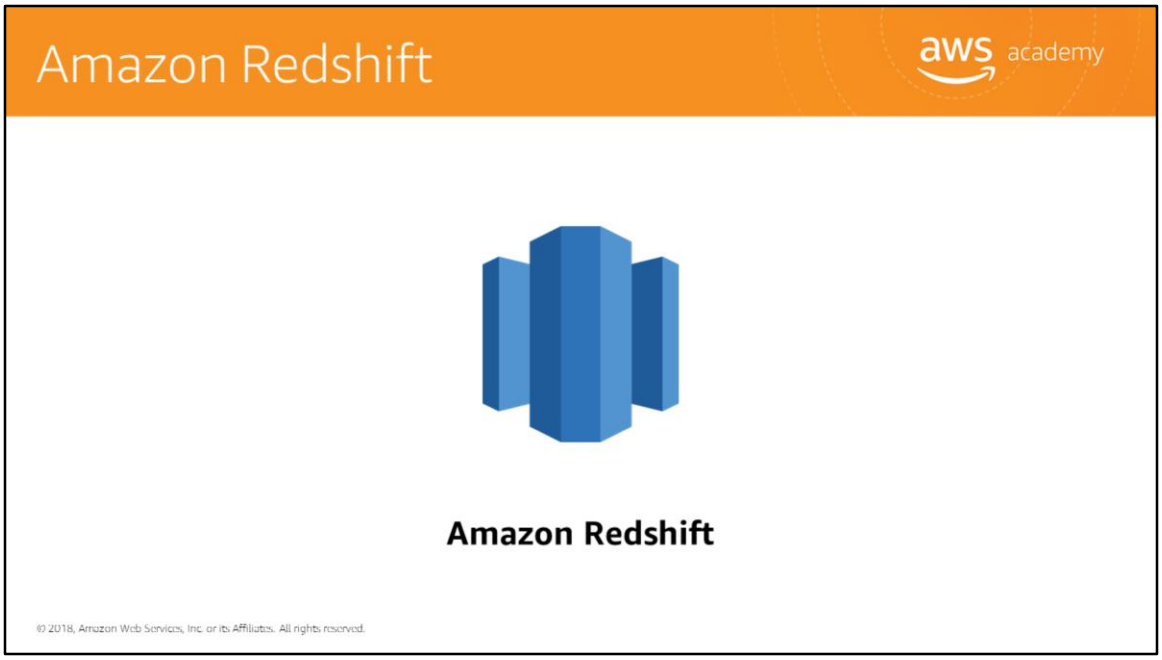


Please review the DynamoDB demonstration: M2_S4_dynamodb v2.0.mp4.

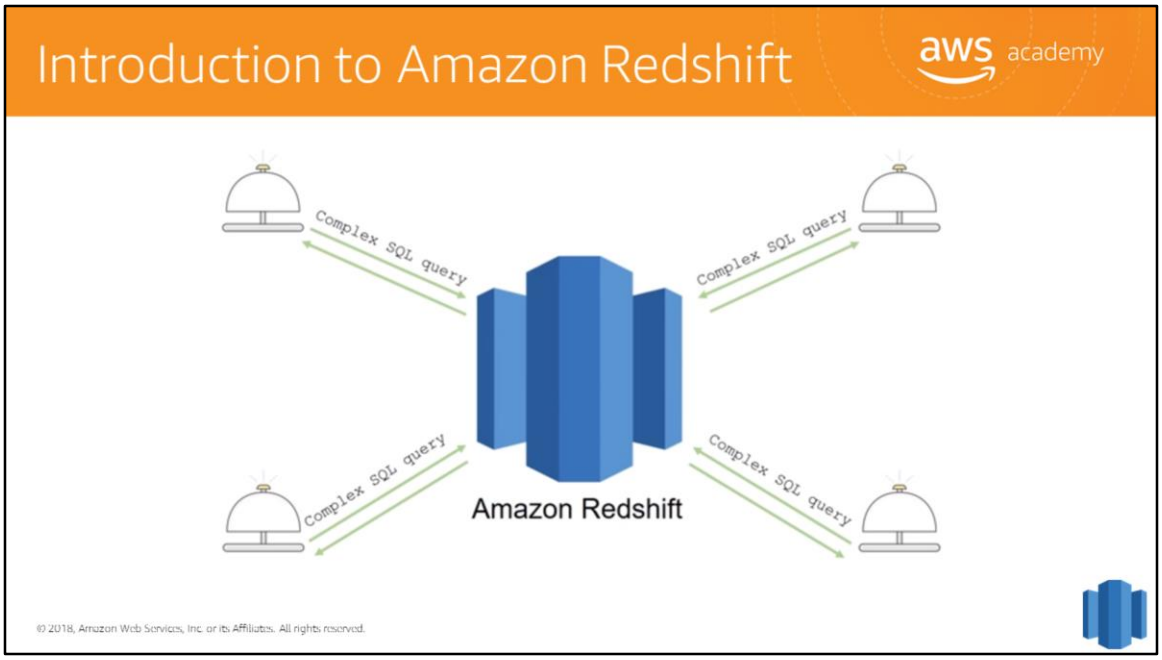
This video demonstration can be found in the learning management system.



Welcome to an introduction to Amazon Redshift.



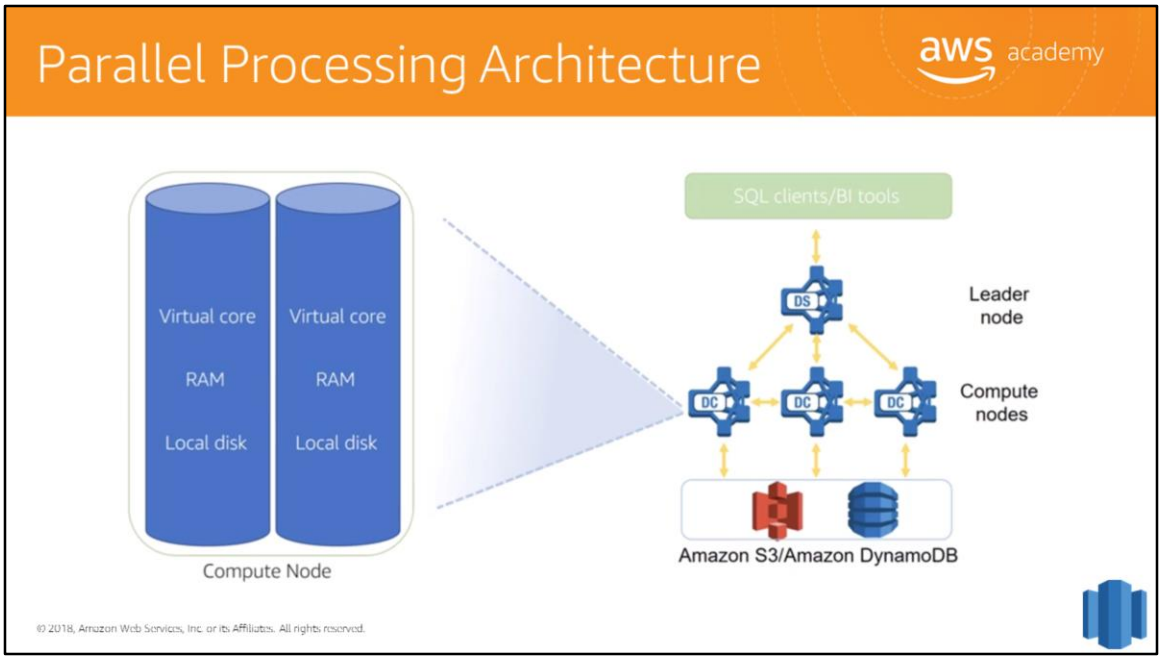
Amazon Redshift is a fast, fully managed data warehouse that makes it simple and cost-effective to analyze all your data using standard SQL and your existing business intelligence (BI) tools. Let's take a look at Amazon Redshift and its use for analytic applications.



Analytics are important for businesses today, but building a data warehouse is complex and expensive. Most data warehouses take months to set up and can cost millions of dollars in software and hardware costs—and that's only the set-up cost.

Amazon Redshift is a fast and powerful, fully-managed data warehouse that makes it simple and cost effective to set up, use, and scale. It allows you to run complex analytic queries against petabytes of structured data using sophisticated query optimization, columnar storage on high performance local disks, and massively parallel query execution. Most results come back in seconds.

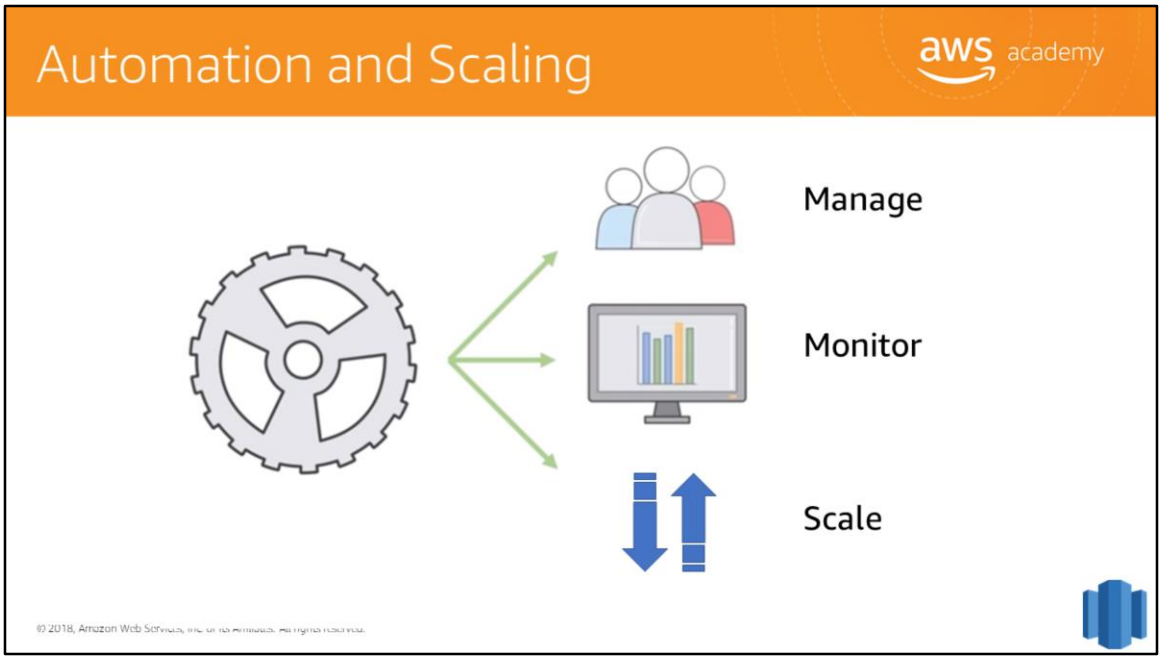
Now, let's review a slightly more detailed exploration of key Amazon Redshift features and some common use cases.



The leader node manages communications with client programs and all communication with compute nodes. It parses and develops execution plans to carry out database operations, in particular the series of steps necessary to obtain results for complex queries. The leader node compiles code for individual elements of the execution plan and assigns the code to individual compute nodes. The compute nodes execute the compiled code and send intermediate results back to the leader node for final aggregation.

As is true with nearly all AWS services, you only pay for what you use. You can get started for as little as 25 cents per hour and, at scale, Amazon Redshift delivers storage and processing for approximately $1,000 dollars per terabyte per year. That's about 1/10 the cost of traditional data warehouse solutions.

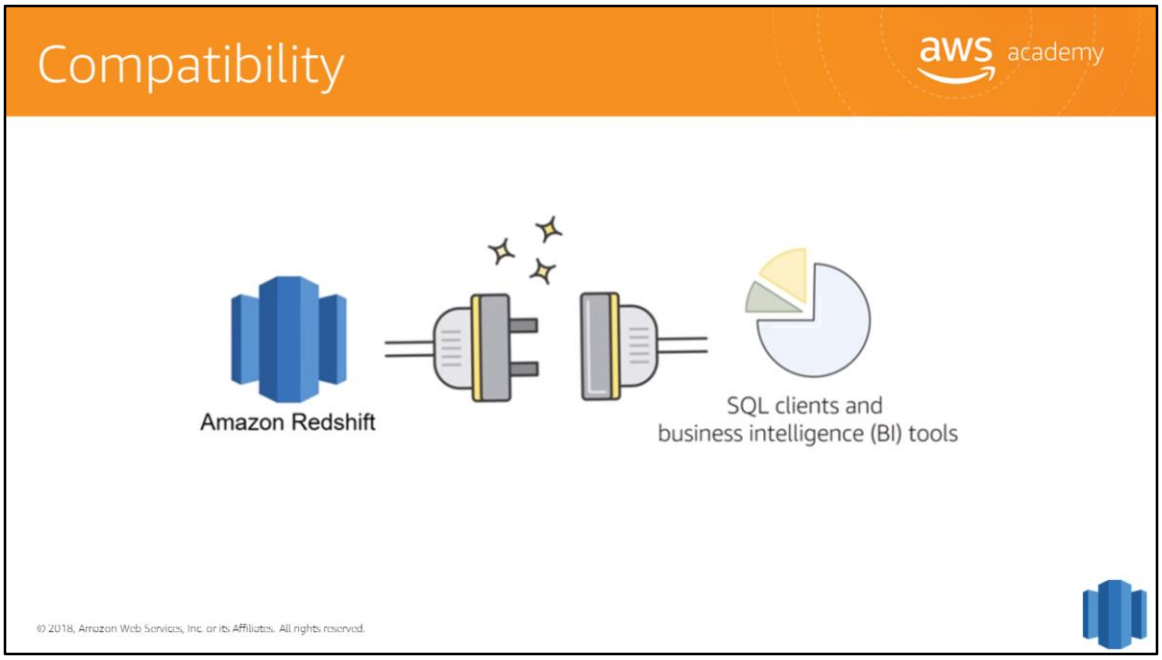
The Amazon Redshift Spectrum feature enables you to run queries against exabytes of data directly in Amazon S3.



It is quite simple to automate most of the common administrative tasks to manage, monitor, and scale your Amazon Redshift cluster, freeing you up to focus on your data and business.

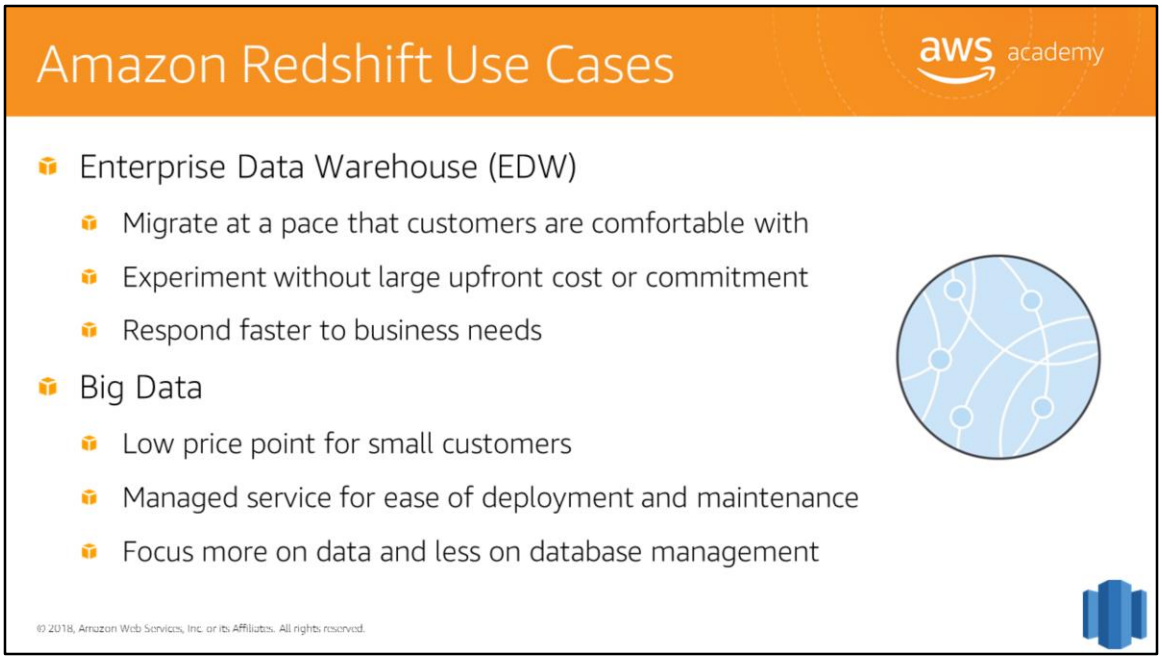
Scalability is intrinsic in Amazon Redshift. Your cluster can be scaled up and down as your needs changes with just a few clicks in the console.

As always at Amazon Web Services, security is our most important consideration. With Amazon Redshift, security is built-in, providing strong encryption of your data both at rest and in transit.



Finally, Amazon Redshift is already compatible with the tools you already know and use. Amazon Redshift supports standard SQL and provides high-performance JDBC and ODBC connectors, which allows you to use the SQL clients and BI tools of your choice.

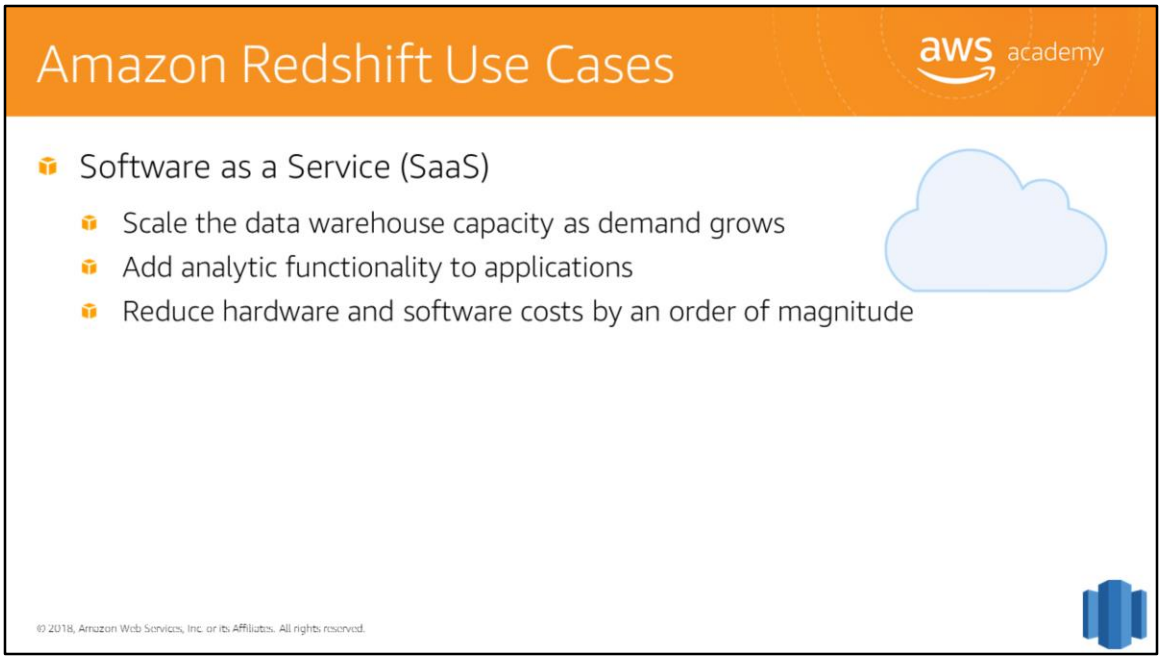
Let’s turn our attention to some common Amazon Redshift use cases.



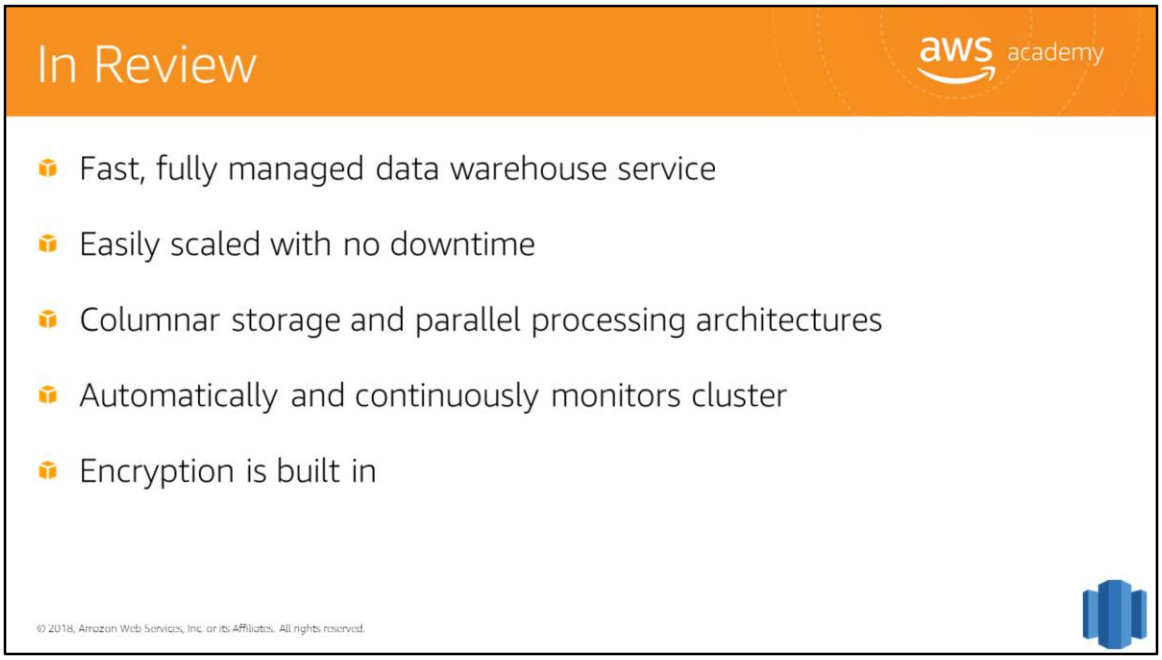
Many customers migrate their traditional enterprise data warehouses to Amazon Redshift with the primary goal of agility. Customers can start at whatever scale they want and experiment with their data without having to rely on complicated processes with their IT departments to procure and prepare their software.

Big data customers have one thing in common: massive amounts of data that stretch their existing systems to a breaking point. Smaller customers typically don't have the money to purchase the amount of hardware and expertise to run these systems. With Amazon Redshift, they can get up and running quickly with their data warehouse at a comparatively low price point.

As a managed service, Amazon Redshift takes care of many of the deployment and ongoing maintenance tasks that often require a database administrator. This frees them up to focus on querying and analysis of their data.



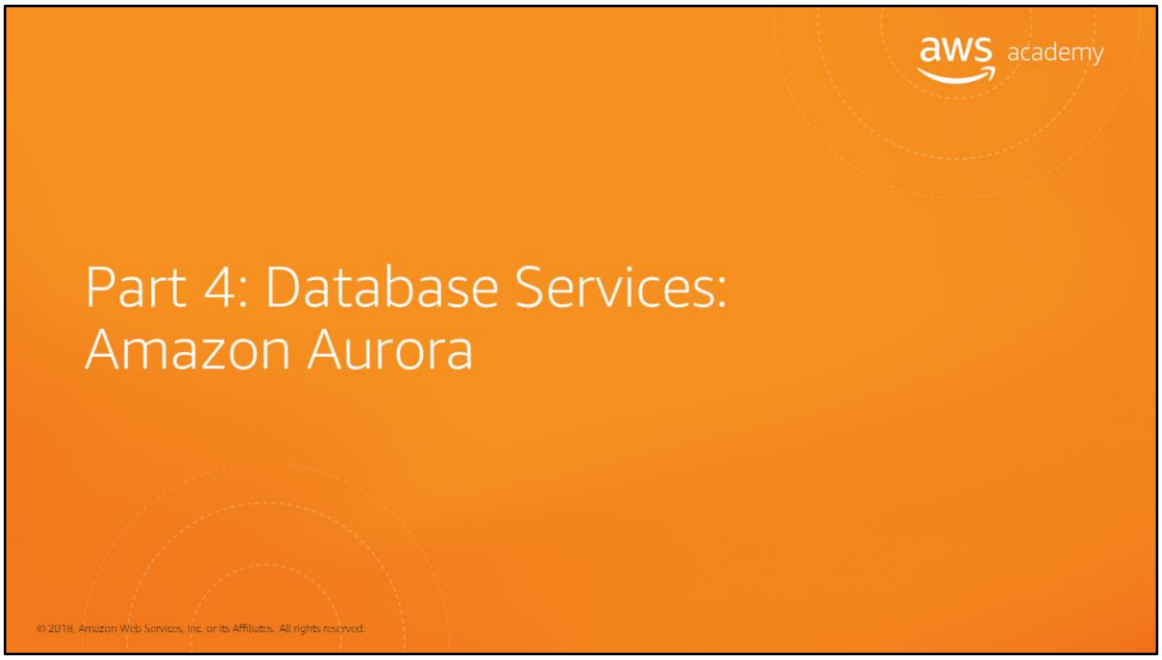
SaaS customers are drawn to the scalable, easy-to-manage platform provided by Amazon Redshift. Some use the platform to provide analytic capabilities to their applications. Some deploy a cluster per customer and use tagging to help simplify and manage their service level agreements and billing.

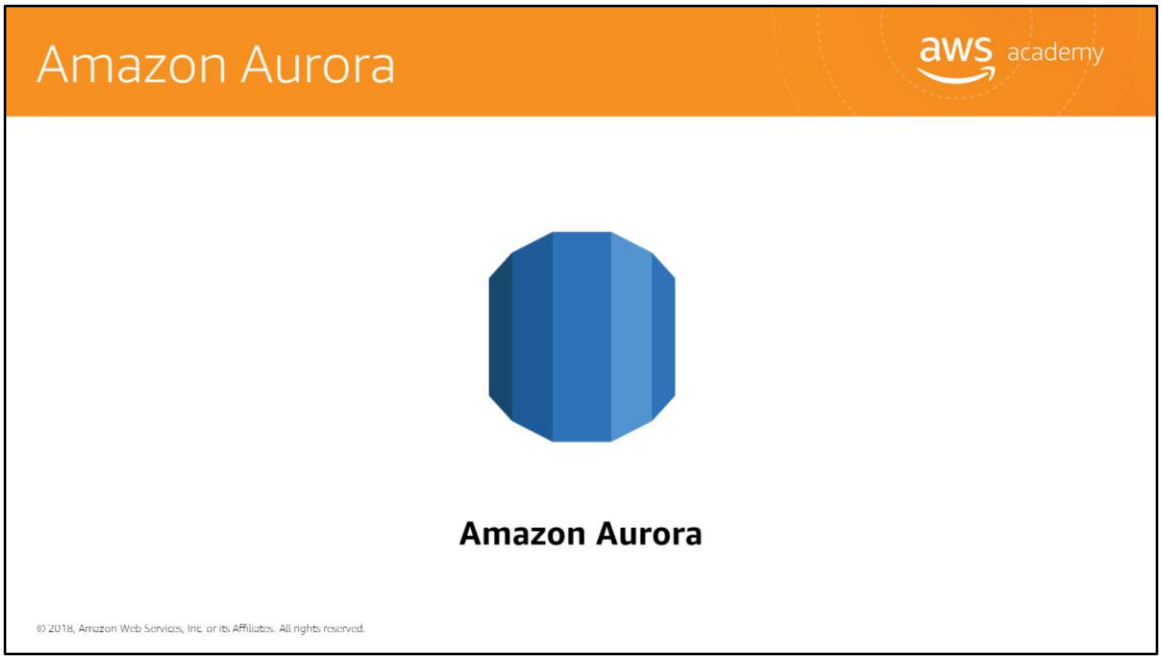


In summary, Amazon Redshift is a fast, fully managed data warehouse service. As a business grows, you can easily scale with no down time by just adding more nodes. Amazon Redshift automatically adds the nodes to your cluster and redistributes the data for maximum performance.

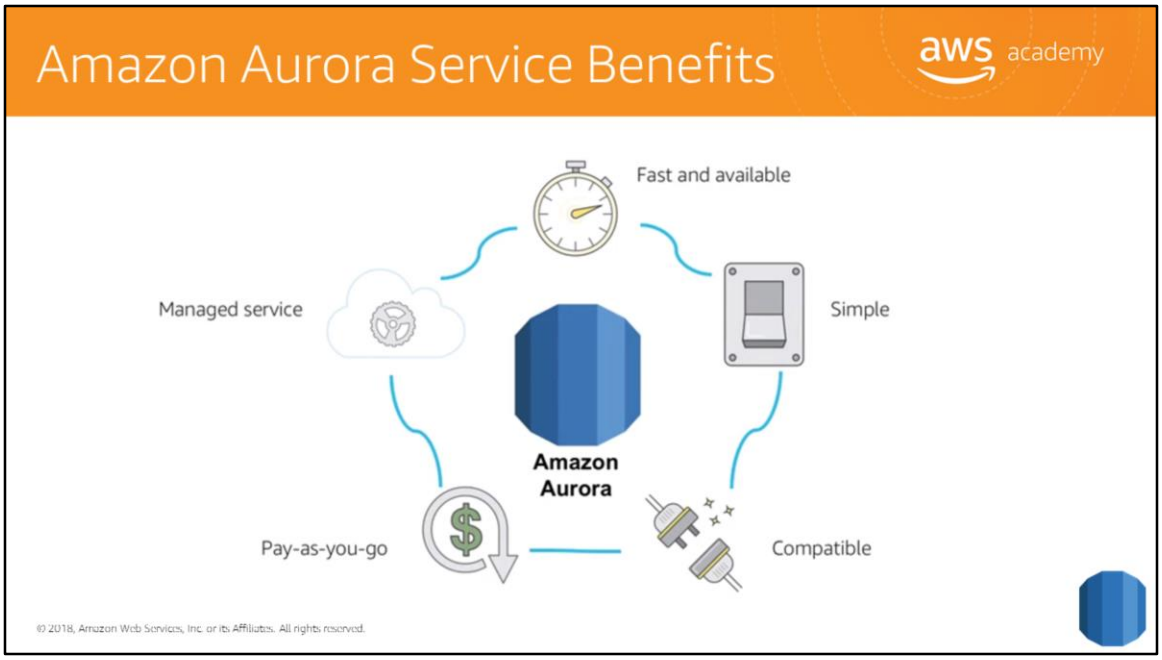
Amazon Redshift uses columnar storage and a massively parallel processing architecture to parallelize and distribute data and queries across multiple nodes to consistently deliver high performance. It also automatically monitors your cluster and backs up your data so you can easily restore if needed. Encryption is built in – you just have to turn it on.

For more information about Amazon Redshift, see https://aws.amazon.com/redshift/.



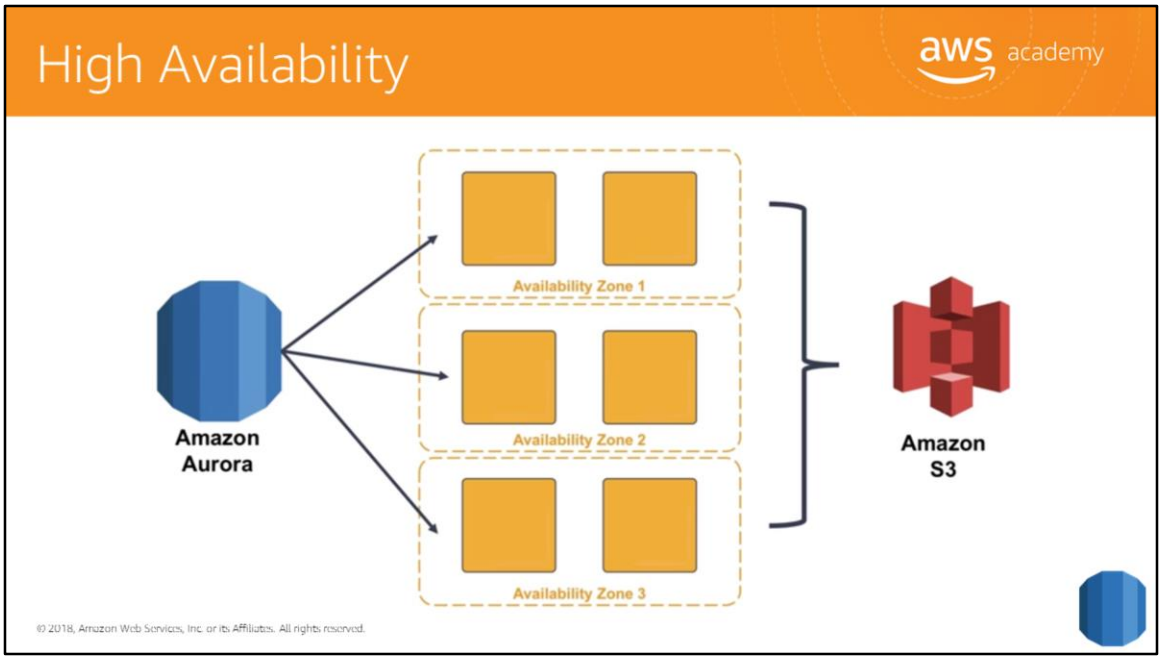


Amazon Aurora is a MySQL- and PostgreSQL-compatible relational database built for the cloud. It combines the performance and availability of high-end commercial databases with the simplicity and cost-effectiveness of open-source databases.



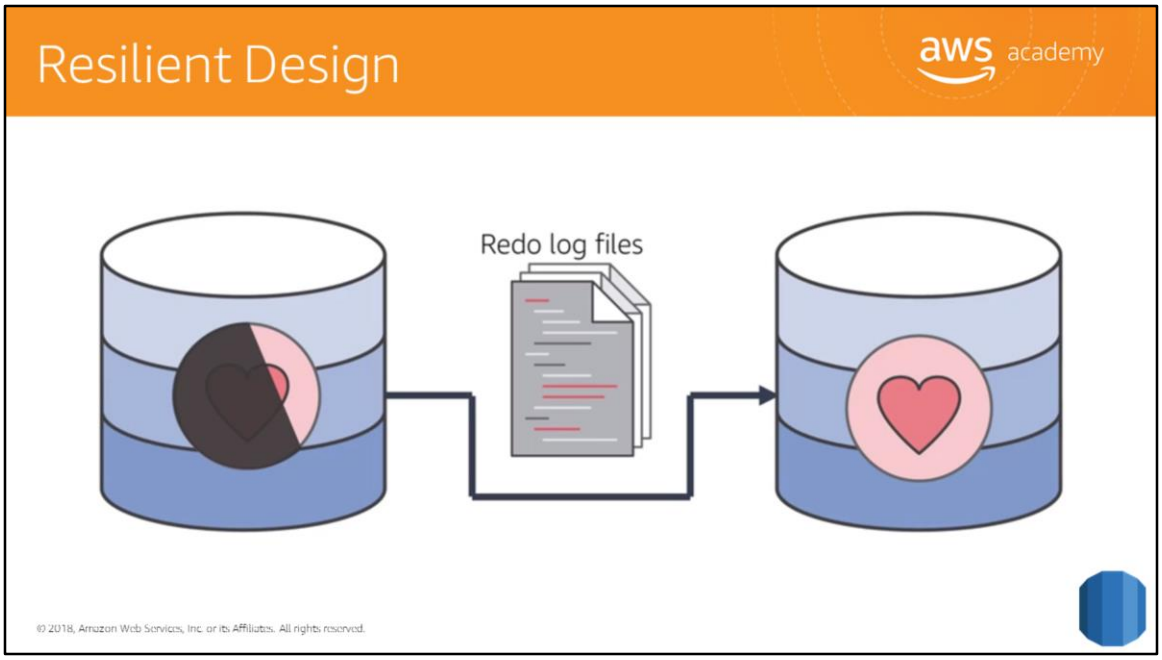
First, let's talk about some of the benefits of Amazon Aurora. It is highly available and offers approximately 5 times the performance of MySql. Amazon Aurora is simple to set up and uses SQL queries. It has drop-in compatibility with MySql 5.6 using the Inno DB storage engine.

Amazon Aurora is a pay-as-you-go service, which ensures that you only pay for the services and features you use. It's a managed service that integrates with features such as the database migration service and the schema conversion tool, which can help you move your data set into Amazon Aurora seamlessly.



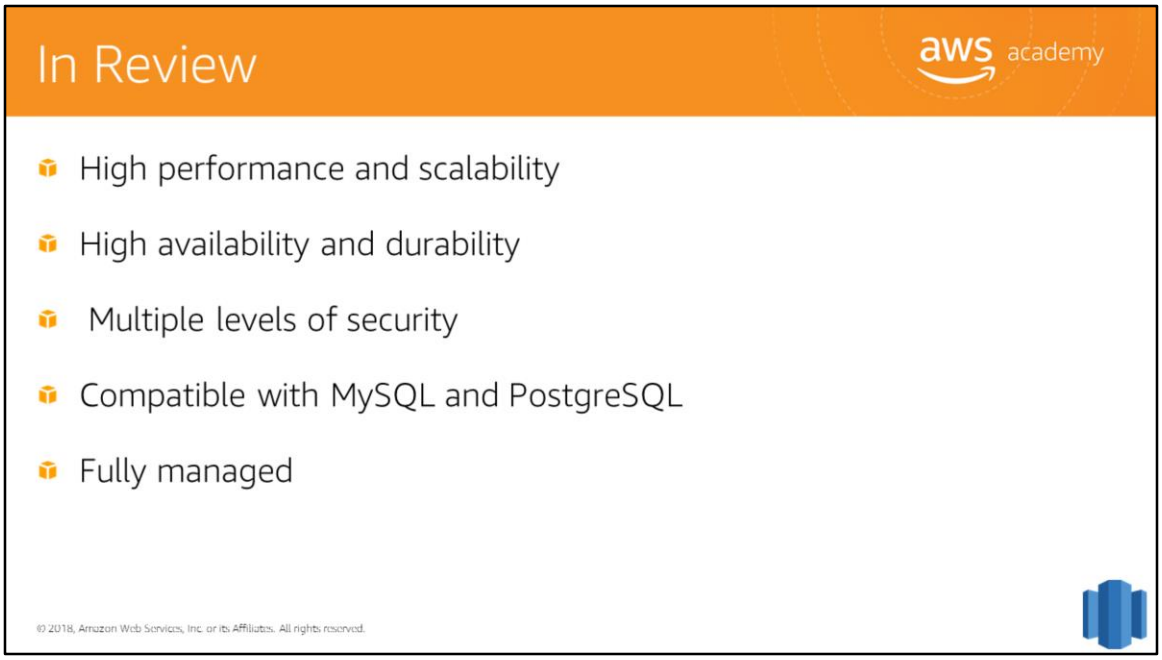
Why would you use Amazon Aurora over, for example, MySql with Amazon RDS? Most of that decision has to do with the high availability and resilient design that Amazon Aurora offers.

Amazon Aurora is highly available, storing six copies of your data across three Availability Zones with continuous backups to Amazon S3. Up to 15 read replicas can be used to help you ensure that your data is not lost. Additionally, Amazon Aurora is designed for instant crash recovery in the event that your primary database becomes unhealthy.



Unlike other databases, after a database crash, Amazon Aurora does not need to replay the redo log from the last database checkpoint. Instead, it performs this on every read operation. This reduces the restart time after a database crash to less than 60 seconds in most cases.

Amazon Aurora has moved the buffer cache out of the database process and makes it available immediately at restart time. This prevents you from having to throttle access until the cache is repopulated to avoid brownouts.



In summary, Amazon Aurora is a highly available, performant, and cost-effective managed relational database.

Amazon Aurora provides 5X the throughput of standard MySQL and 3X the throughput of standard PostgreSQL running on the same hardware. This performance is on par with commercial databases, at 1/10th the cost.

It also offers greater than 99.99% availability. It has fault-tolerant and self-healing storage built for the cloud that replicates six copies of your data across three Availability Zones while it continuously backs up your data to Amazon S3.

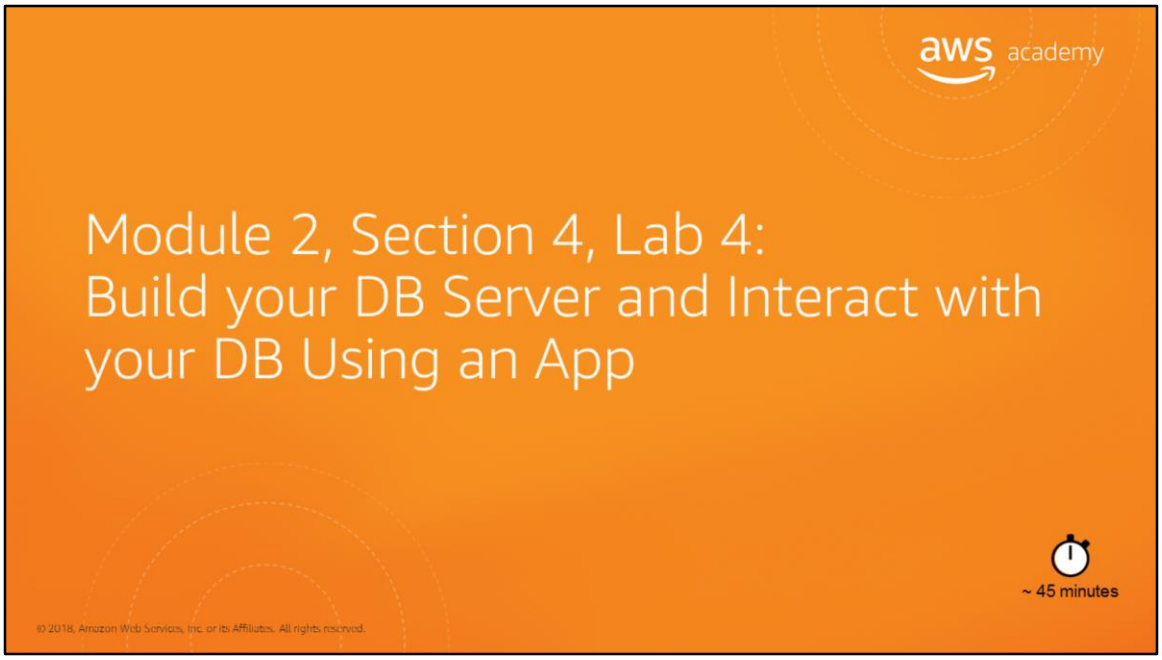
Multiple levels of security are available, including network isolation using Amazon VPC, encryption at rest using keys you create and control through AWS Key Management Service (KMS), and encryption of data in transit using SSL.

The Amazon Aurora database engine is fully compatible with existing MySQL and PostgreSQL open source databases, and adds compatibility for new releases regularly.

Finally, Amazon Aurora is fully managed by Amazon RDS. You no longer need to worry about database management tasks such as hardware provisioning, software patching, setup,

configuration, or backups.

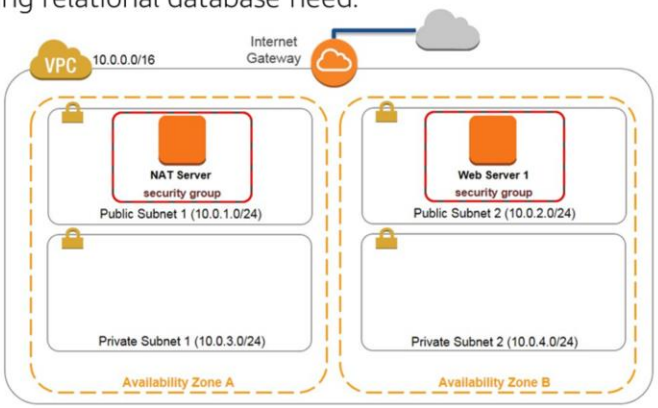
For more information about Amazon Aurora, see https://aws.amazon.com/rds/aurora/.



Lab 4 Scenario

aws academy

This lab is designed to show you how to leverage an AWS-managed database instance for solving relational database need.



Amazon RDS makes it easy to set up, operate, and scale a relational database in the cloud. It provides cost-efficient and resizable capacity while managing time-consuming database administration tasks, which allows you to focus on your applications and business. Amazon RDS provides you with six familiar database engines to choose from: Amazon Aurora, Oracle, Microsoft SQL Server, PostgreSQL, MYSQL, and MariaDB.

Amazon RDS multi-AZ deployments provide enhanced availability and durability for DB instances, making them a natural fit for production database workloads. When you provision a multi-AZ DB instance, Amazon RDS automatically creates a primary DB instance and synchronously replicates the data to a standby instance in a different Availability Zone.

After completing this lab, you will be able to:

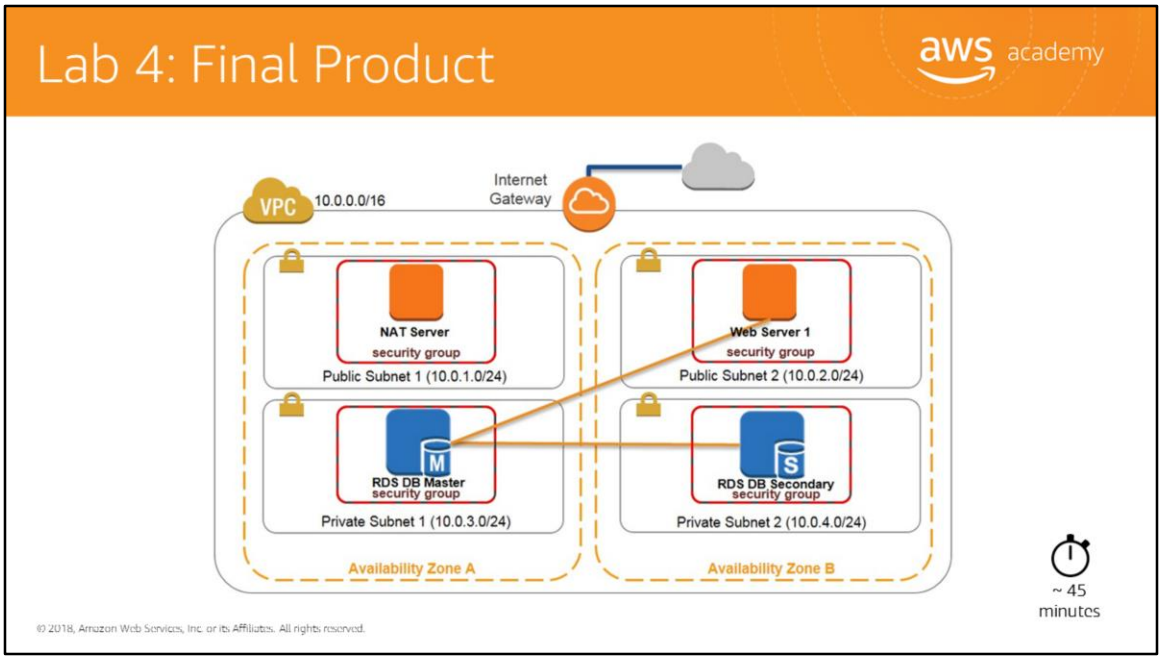
- Launch an Amazon RDS DB instance with high availability.
- Configure the DB instance to permit connections from your web server.
- Open a web application and interact with your database.

Lab 4: Tasks

Create a **VPC Security Group.**

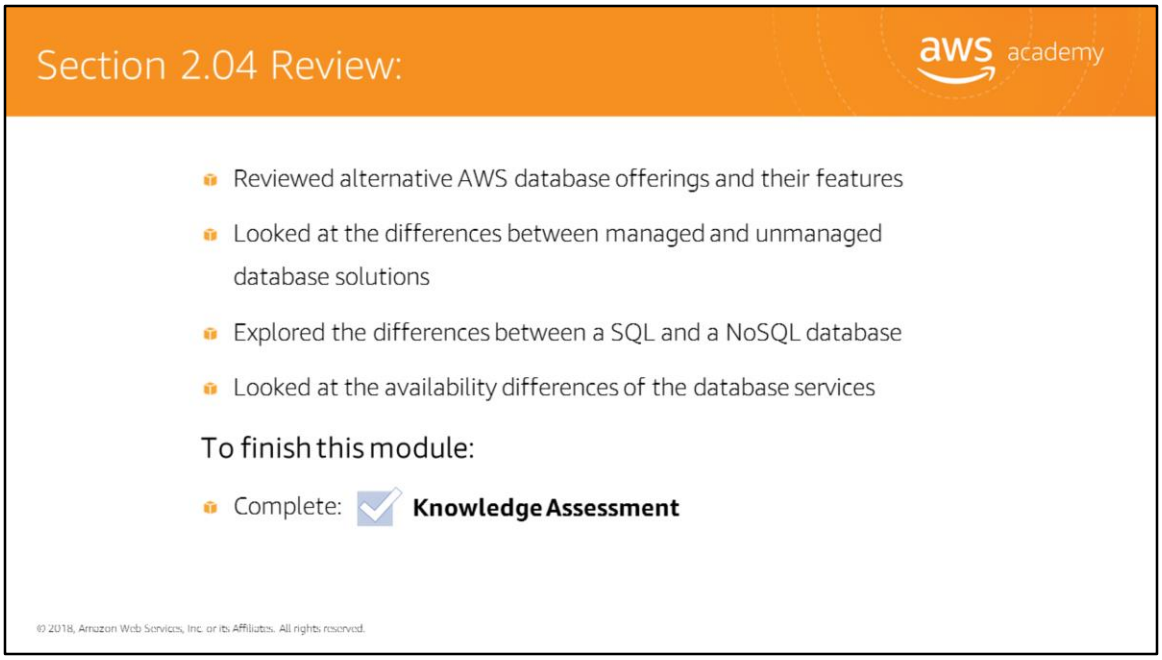
Create a **DB Subnet Group.**

Create an **Amazon RDS DB** instance and interact with your database.



In this lab, you:

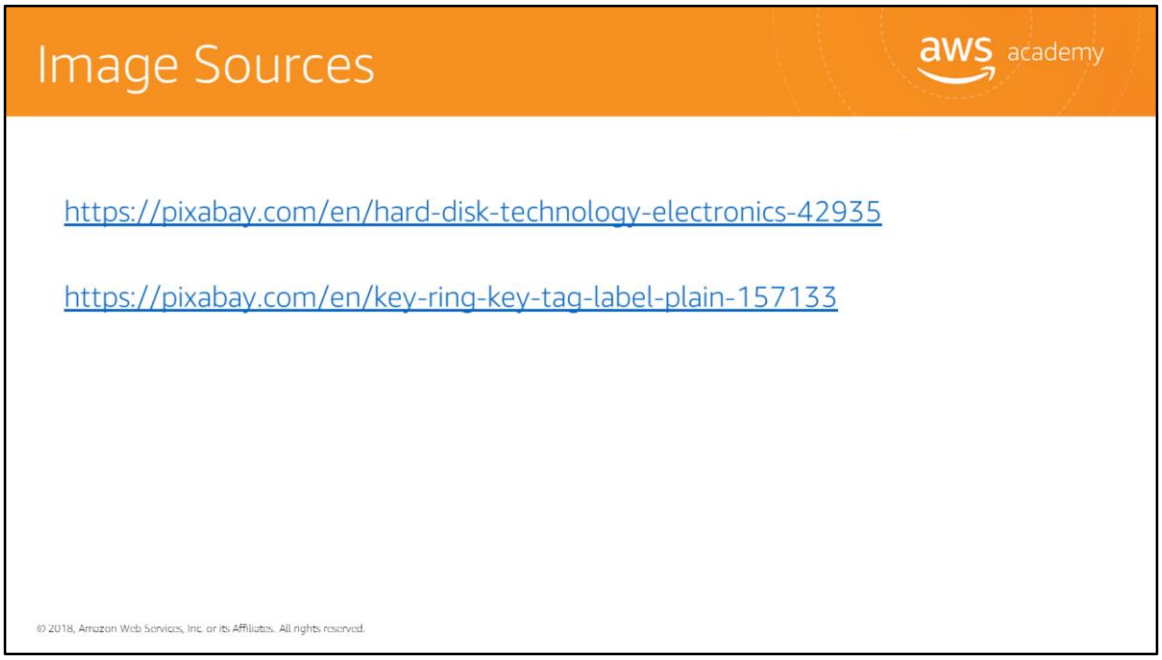
- Launched an Amazon RDS DB instance with high availability.
- Configured the DB instance to permit connections from your web server.
- Opened a web application and interacted with your database.



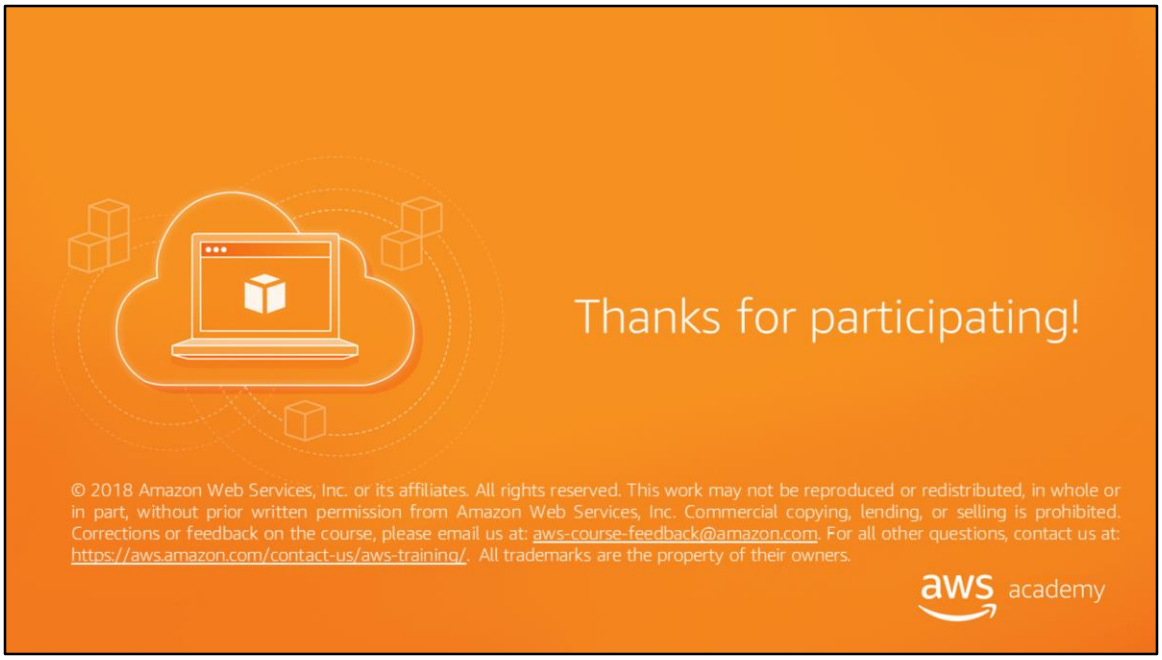
In review, we:

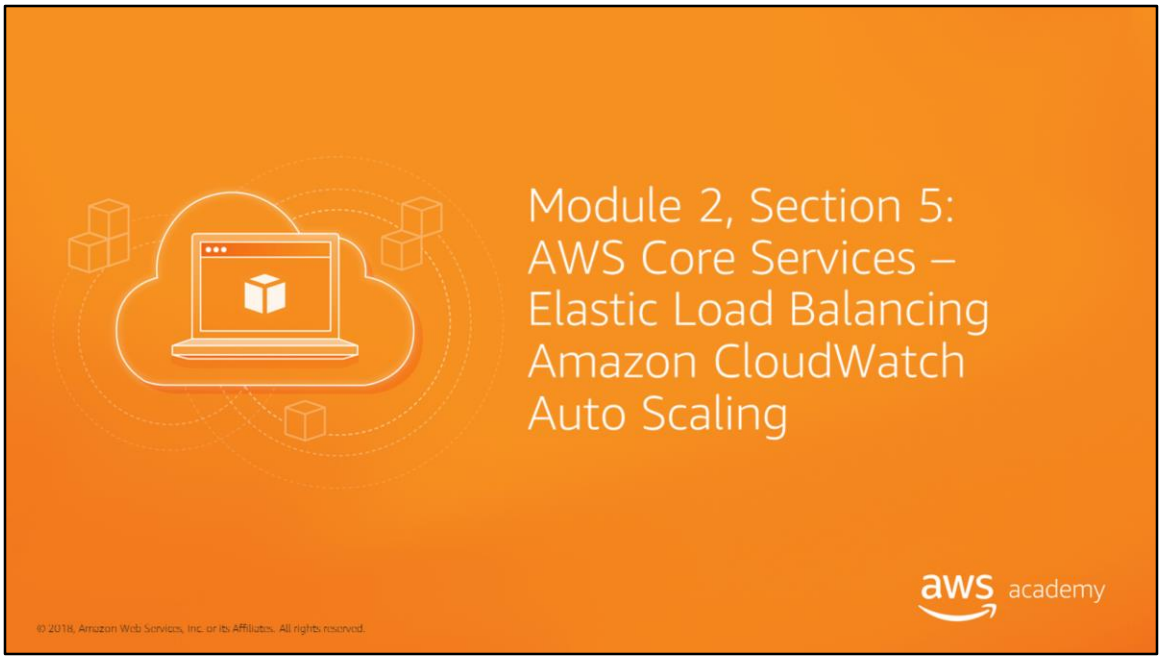
- Provided an overview of different AWS database services in the cloud
- Highlighted the difference between an unmanaged and a managed database solutions
- Explored the differences between a SQL and a NoSQL database
- Reviewed the availability differences of alternative database solutions

Now that we have a better understanding some of the database services offered by AWS, in Module 3 we look at another area that is of the utmost importance to AWS, security.

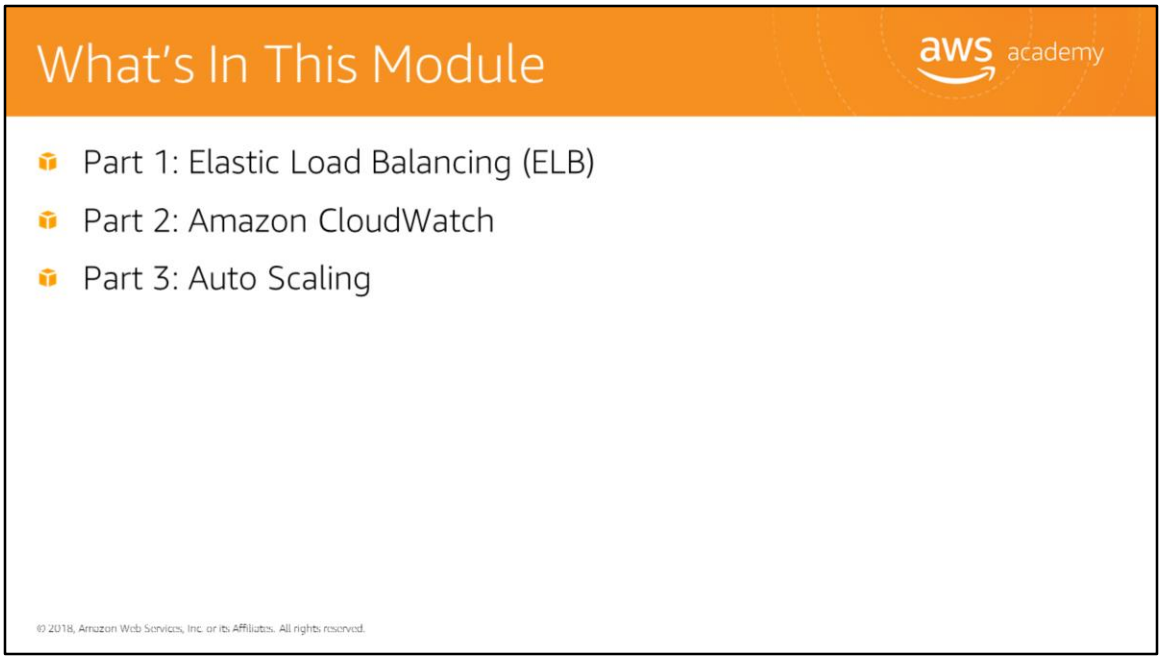


This slide contains attributions for any Creative Commons-licensed images used within this module.

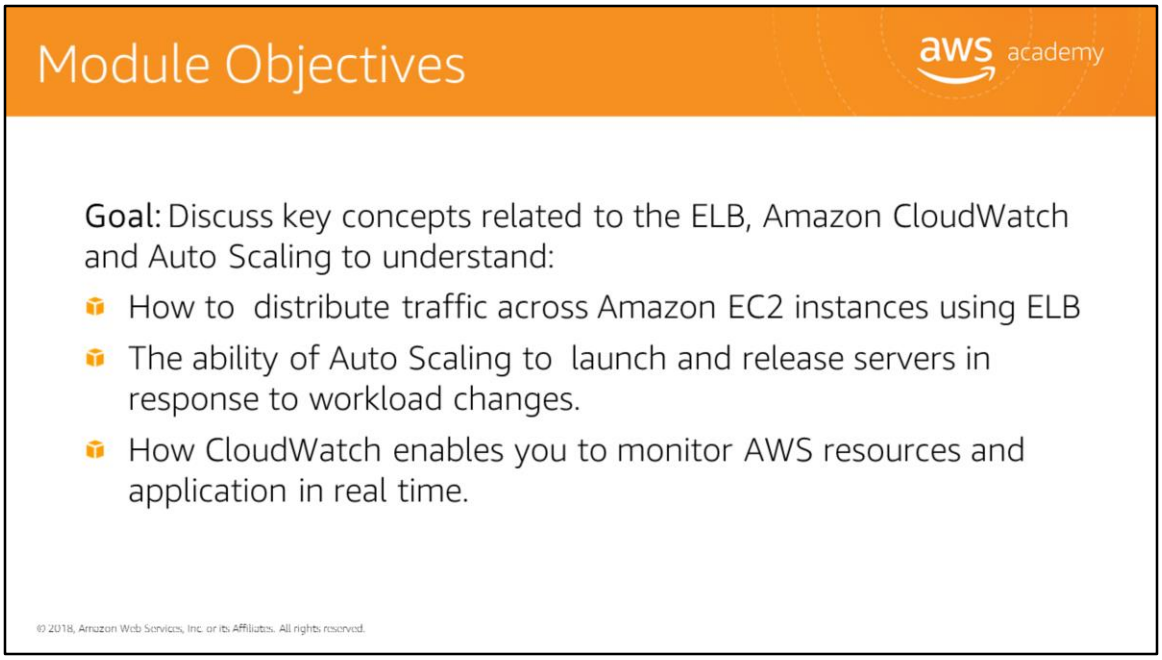




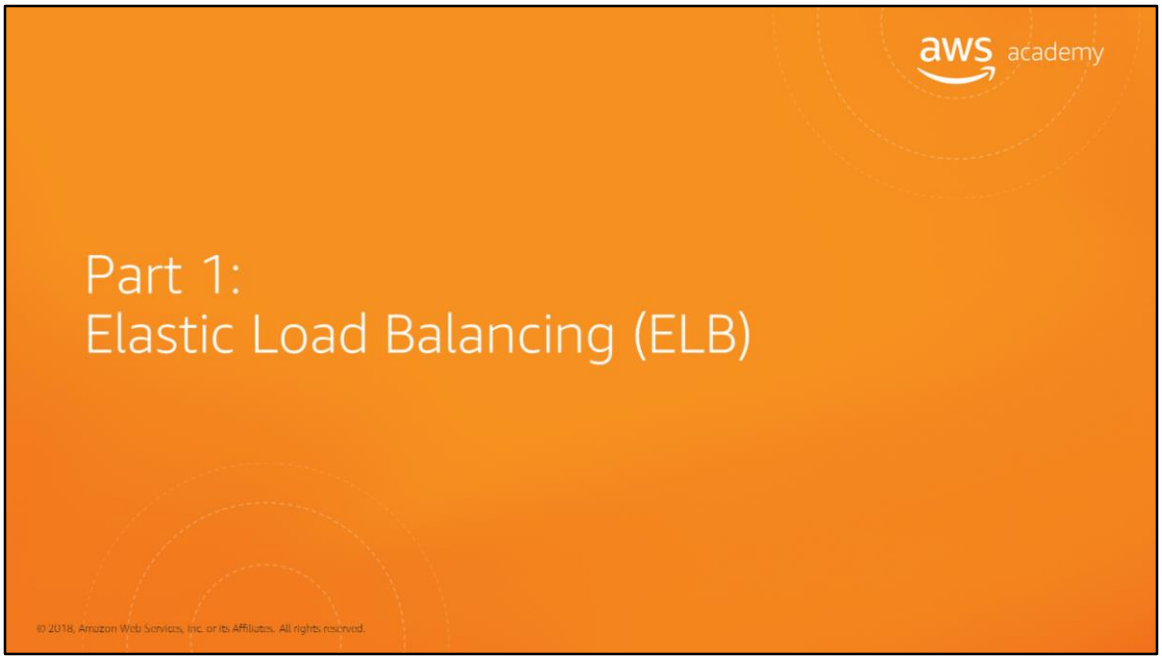
Welcome to Module 2, Section 5 – AWS Core Services – Elastic Load Balancing, Amazon CloudWatch, and Auto Scaling. In this module, we will learn how each of these services work both independently and together to help you deploy highly available and optimized workloads on AWS.



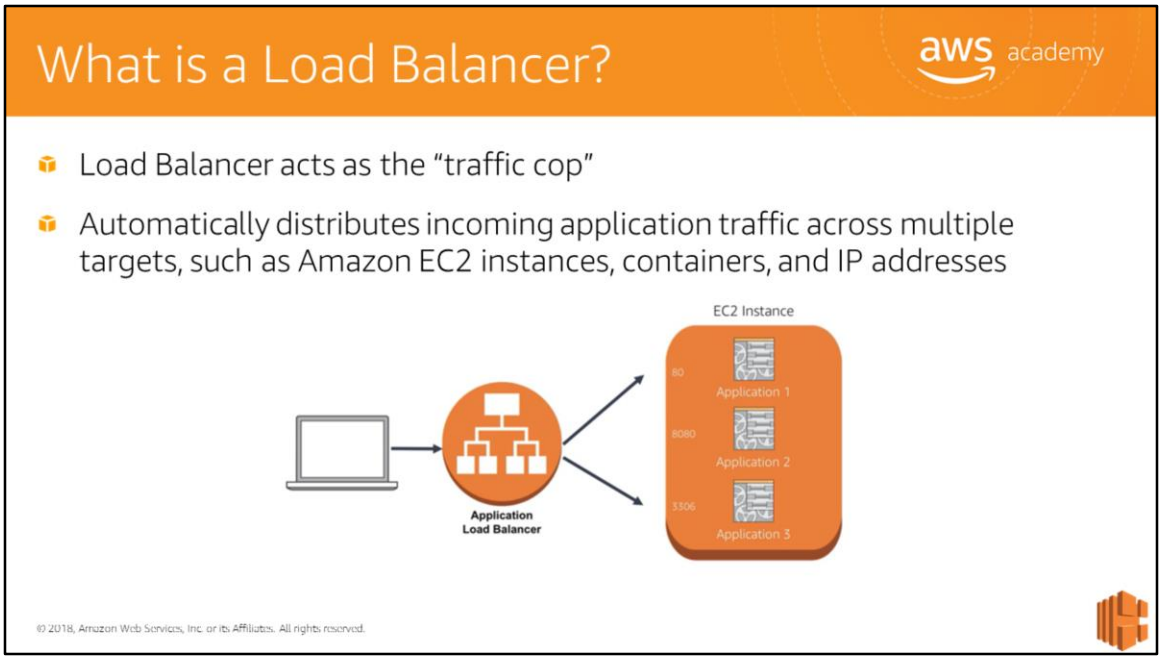
In this module we will review Elastic Load Balancing (ELB), Amazon CloudWatch and Auto Scaling to build robust and highly available architectures.



The goal of this module understand how traffic distribution of ELB and Auto Scaling can help you build robust and highly available architectures. We will also look at Amazon CloudWatch see how we can monitor AWS resources in real time.



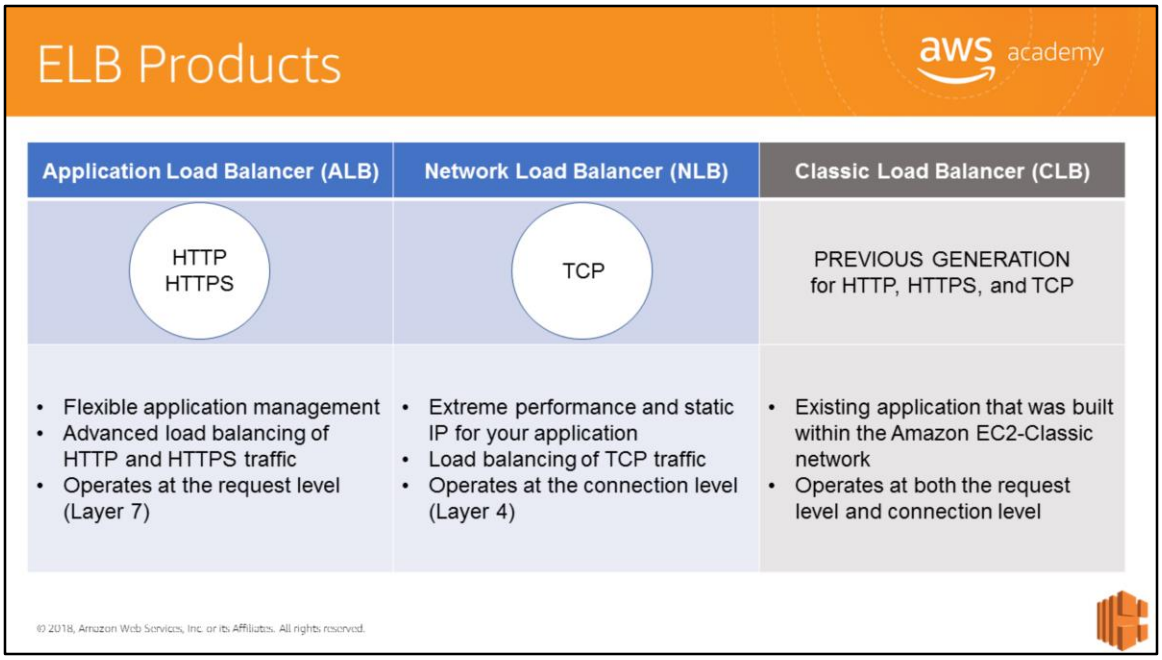
ELB automatically **distributes incoming application** traffic across multiple targets, such as Amazon EC2 instances, containers, and IP addresses.



Modern high-traffic websites must serve hundreds of thousands, if not millions, of concurrent requests from users or clients and return the correct text, images, video, or application data, all in a fast and reliable manner. To cost-effectively scale to meet these high volumes, modern computing best practice generally requires adding more servers.

A load balancer acts as the “traffic cop” sitting in front of your servers and routing client requests across all servers capable of fulfilling those requests in a manner that maximizes speed and capacity utilization and ensures that no one server is overworked, which could degrade performance. If a single server goes down, the load balancer redirects traffic to the remaining online servers. When a new server is added to the server group, the load balancer automatically starts to send requests to it.

Let’s start by looking at ELB and the three types of load balancers that all feature the high availability, automatic scaling, and robust security necessary to make your applications fault tolerant.



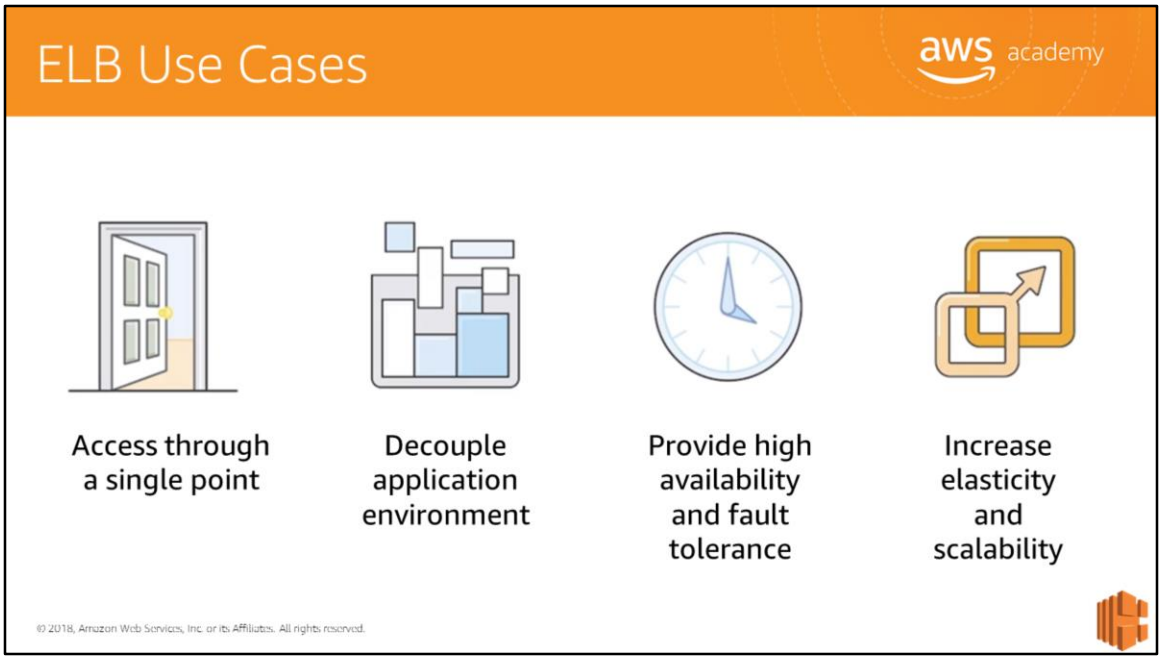
ELB offers three types of load balancers: Application Load Balancer, Network Load Balancer, and Classic Load Balancer.

First we have an application load balancer (ALB) that functions at the application level. It supports content-basing routing and applications that run in containers. It supports a pair of industry-standard protocols (websocket and http/2) and can provide additional visibility into the health of target instances and containers.

Next, we have network load balancers (NLB) which are designed to handle tens of millions of requests per second while maintaining high throughput at ultra low latency. NLB is ideal for load balancing TCP traffic and is capable of handling millions of requests per second while maintaining ultra-low latencies.

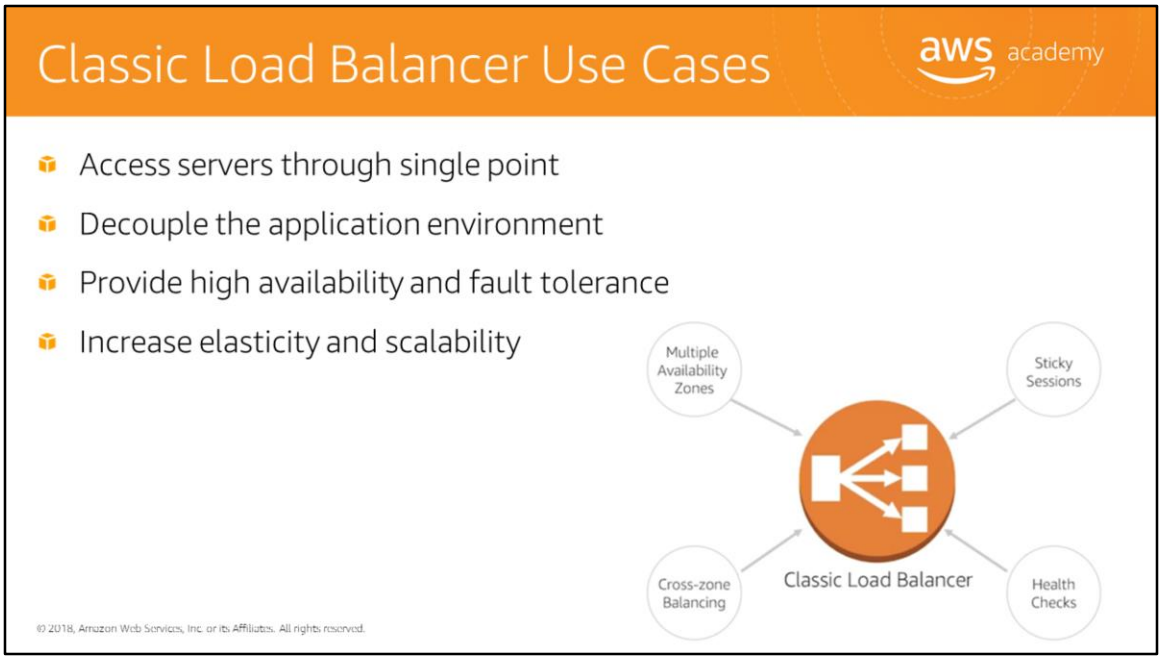
Finally, the classic load balancer (CLB) provides the basic load balancing across multiple Amazon EC2 instances and operates at both request and connection level. This ideal for applications that were built within the Amazon EC2-Classic network.

For more information, see https://aws.amazon.com/elasticloadbalancing/details/#compare.



There are many reasons to use a load balancer:

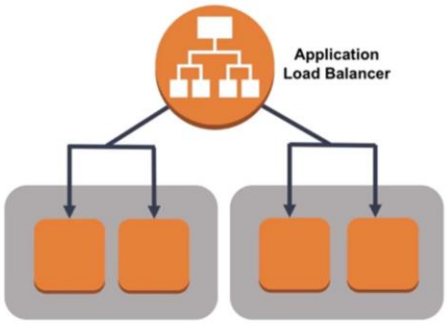
- To secure access to your web servers through a single exposed point of access
- To decouple your environment to using both public facing and internal load balancers
- To provide high availability and fault tolerance with the ability to distribute traffic across multiple Availability Zones
- To increase elasticity and scalability with minimal overhead



The Classic Load Balancer is a distributed software load balancing service that enables the use of many helpful features packaged into a managed solution.

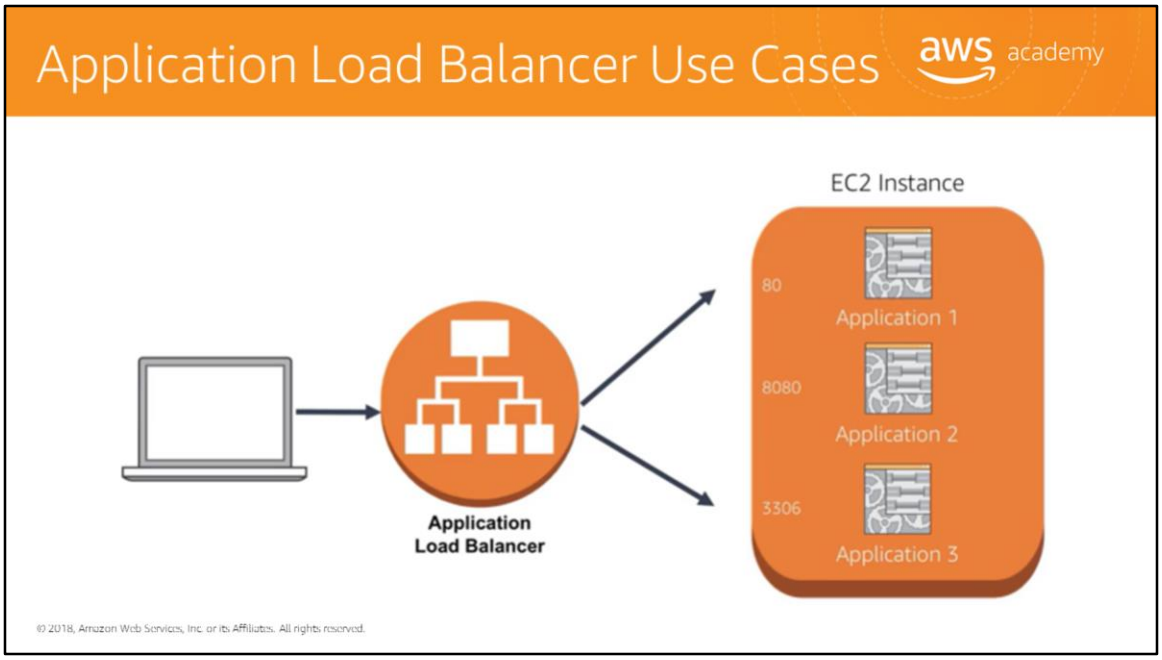
Some of the use cases for the Classic Load Balancer are securing access to your web servers through a single exposed point of access, decoupling your application environment, using both internet and internal load balancers, providing high availability and fault tolerance with the ability to distribute traffic across multiple Availability Zones, and increase elasticity and scalability with minimal overhead.

Application Load Balancer Features

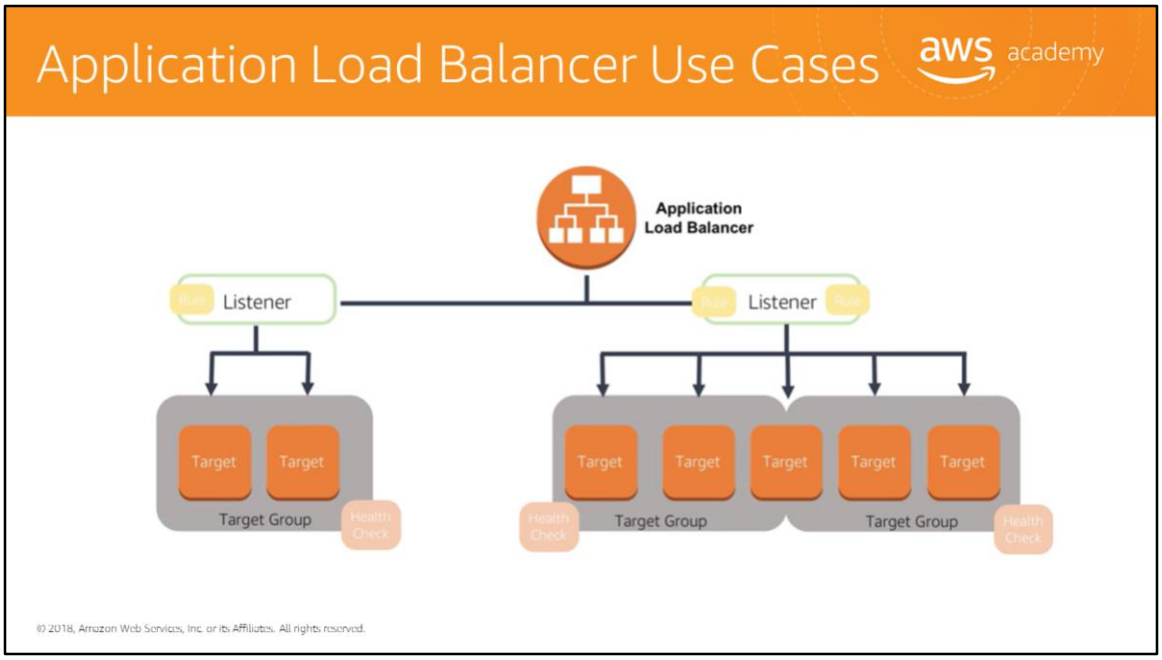


- Path- and host-based routing
- Native IPv6 Support
- Dynamic ports
- Additional supported request protocols
- Deletion protection and request Tracking
- Enhanced metrics and and access logs
- Targeted health checks

The Application Load Balancer offers most of the features offered by the Classic Load Balancer; however, it adds some important features and enhancements that make it applicable for unique use cases.



There are a number of scenarios in which you would use the Application Load Balancer. One is the ability to use containers to host your micro services and route to those applications from a single load balancer. Application Load Balancer allows you to route different requests to the same instance, but differ the path based on the port. If you have different containers listening on various ports, you can set up routing rules to distribute traffic to only the desired backend application.



Here, we can see how the Application Load Balancer routes and organizes backend targets. When configuring the listeners for the load balancer, you create *rules* in order to direct how the requests received by the load balancer will be routed to the backend targets. To register those targets to the load balancer and configure the health check the load balancer will use for the targets, you create target groups. As we see here, targets can also be members of multiple target groups.

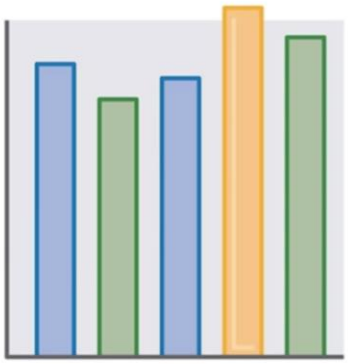
Network Load Balancer Use Cases

- Sudden and volatile traffic patterns
- Single static IP address per Availability Zone
- Ideal for applications that require extreme performance

NLB is optimized to handle sudden and volatile traffic patterns while using a single static IP address per Availability Zone.

Since it handles millions of requests per second while maintaining ultra-low latencies, it is ideal for the applications that require extreme performance.

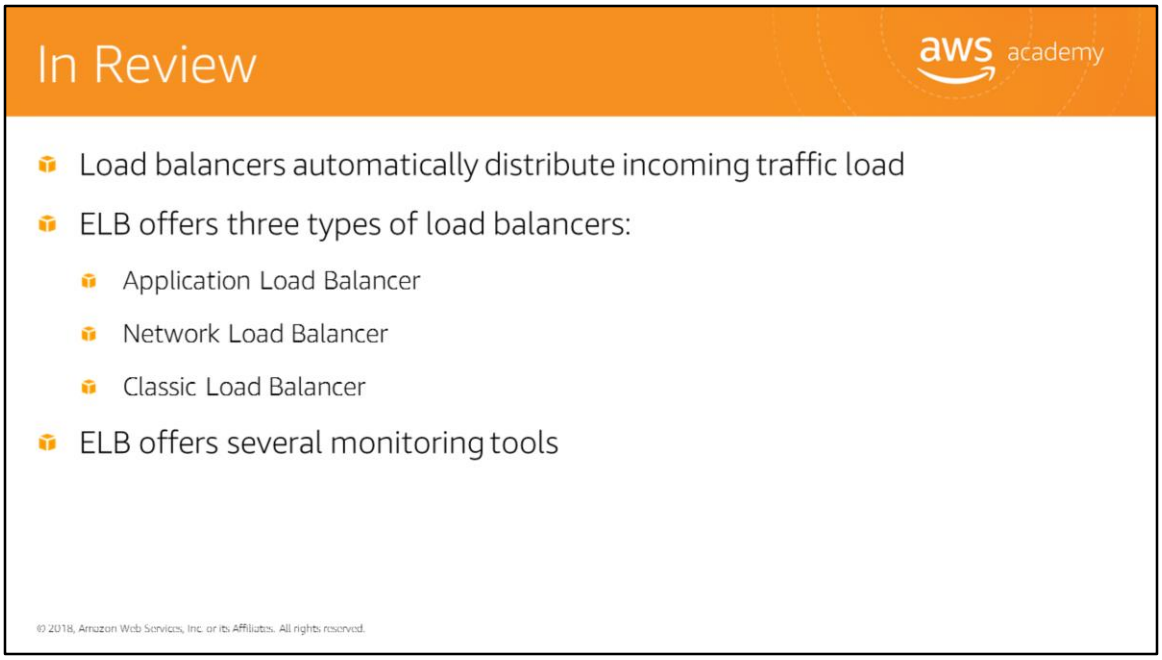
Load Balancer Monitoring



- View HTTP responses
- See number of healthy and unhealthy hosts
- Filter metrics based on Availability Zones or load balancer

ELB provides many metrics by default. These metrics allow you to see HTTP responses, the number of healthy and unhealthy hosts behind the load balancer, and you can filter these metrics based on the Availability Zone of the backend instances or based on the load balancer that you are using.

For health checks, the load balancer allows you to see the number of healthy and unhealthy Amazon EC2 hosts behind the load balancer. This is accomplished with a simple attempted connection request to the Amazon EC2 instance. To discover the availability of your Amazon EC2 instances, a load balancer periodically sends pings, attempts connections, or sends requests to test the Amazon EC2 instances. These tests are called *health checks*.



In review, load balancers are used to distribute traffic load. Load balancers are used to increase capacity (concurrent users) and reliability of applications.

There are three ELB products: Application Load Balancer, Network Load Balancer, and Classic Load Balancer.

Application Load Balancer is best suited for load balancing of HTTP and HTTPS traffic and provides advanced request routing targeted at the delivery of modern application architectures, including micro services and containers.

Network Load Balancer is best suited for load balancing of TCP traffic where extreme performance is required.

Classic Load Balancer provides basic load balancing across multiple Amazon EC2 instances and operates at both the request level and connection level.

You can use the monitoring tools offered by ELB to evaluate the health of your implementation. To learn more about ELB see https://aws.amazon.com/elasticloadbalancing/.

Before we get into the details of scaling, let's look at a tool that helps you leverage resources efficiently by providing insight into your resources.

Leveraging Information to Optimize

aws academy

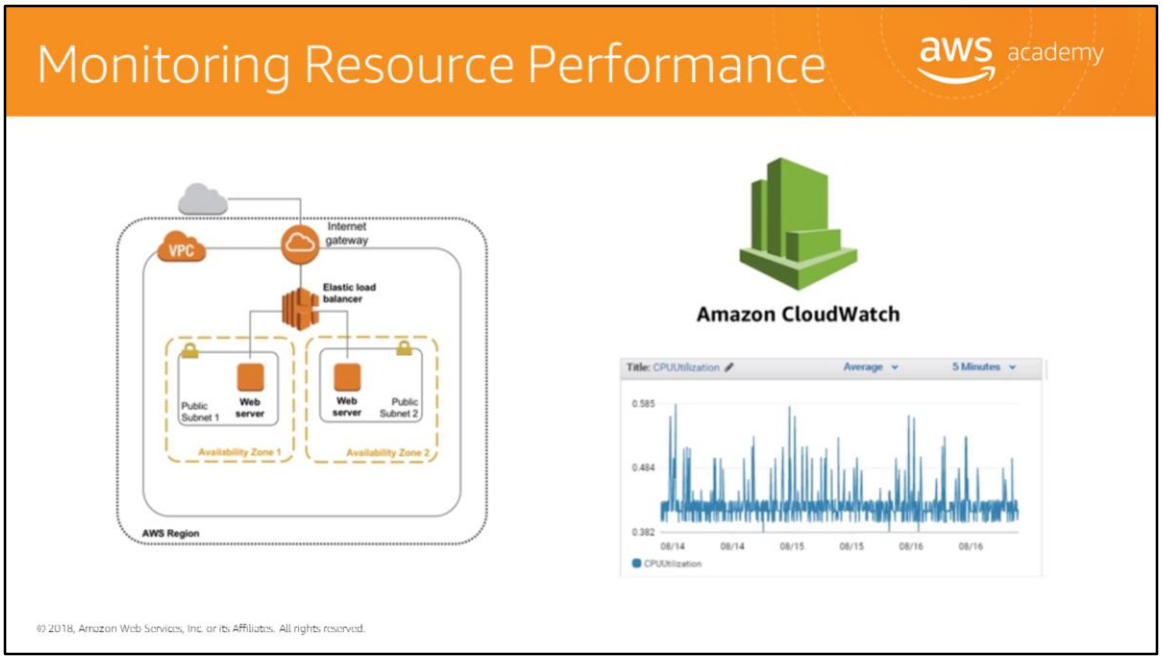
To leverage AWS in an **efficient** way, you need **insight** into your AWS resources:

- How do I know when I should launch more **Amazon EC2 instances?**
- Is my **application's performance** or **availability** being affected by a lack of sufficient capacity?
- How much of my infrastructure is actually **being used**?

How do you capture this information? Without any kind of instrumentation, you really are flying blind.

To leverage resources efficiently, you need insight into your resources. You need to understand:

- How do I know when I should launch more Amazon EC2 instances?
- Is my application's performance or availability being affected by a lack of sufficient capacity?
- How much of my infrastructure is actually being used?



You can capture this information with Amazon CloudWatch.

When you run your applications on Amazon EC2 instances, it is critical to monitor the performance of your workload using Amazon CloudWatch. While monitoring workload performance, you should ask yourself two critical questions:

1) How can I ensure that my workload has enough Amazon EC2 resources to meet fluctuating performance requirements?
2) How can I automate Amazon EC2 resource provisioning to occur on-demand?

CloudWatch helps with performance monitoring, but by itself will not add or remove Amazon EC2 instances. This is where Auto Scaling comes into the picture.

With Amazon EC2 Auto Scaling, you can maintain the health and availability of your fleet, also dynamically scale your Amazon EC2 instances to meet demands during spikes and lulls.

The primary function of Amazon CloudWatch is to enable you to track and monitor the performance and health of your resources and applications. You can also use CloudWatch to collect and monitor log files from Amazon EC2 instances, AWS CloudTrail, Amazon Route 53, and other sources.

Amazon CloudWatch is a distributed statistics gathering system. It collects and tracks your metrics from your applications. You also have the ability to create and use your own custom metrics.

CloudWatch has two different monitoring options:

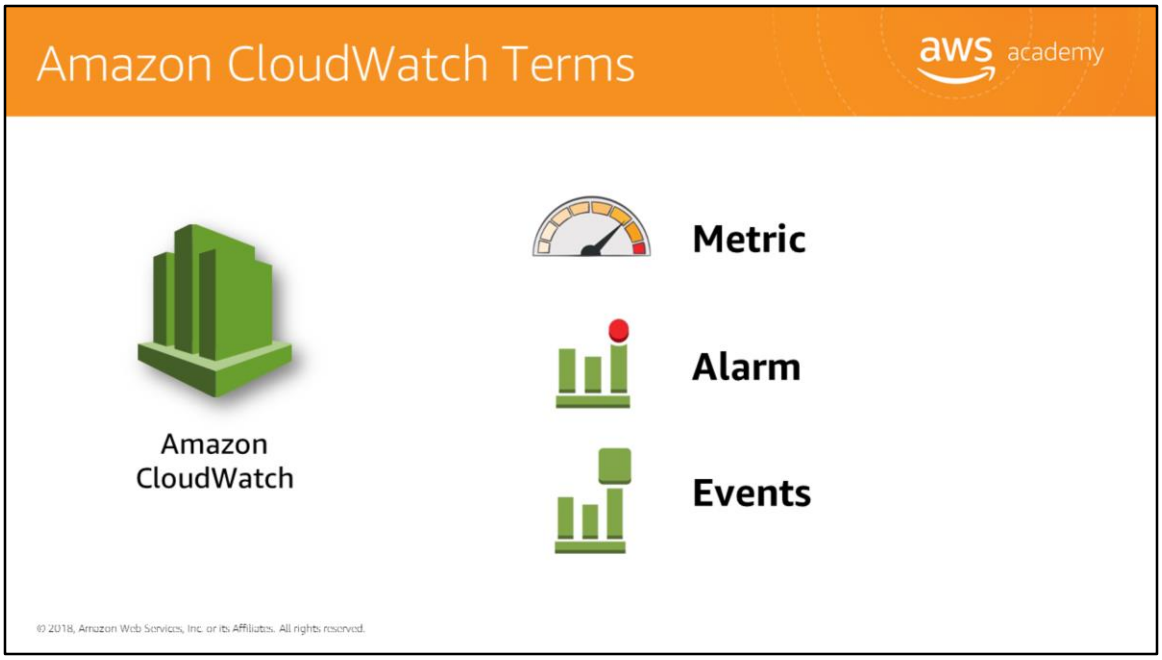
- **Basic Monitoring for Amazon EC2 instances**: Seven pre-selected metrics at five-minute frequency and three status check metrics at one-minute frequency, for no additional charge.
- **Detailed Monitoring for Amazon EC2 instances**: All metrics available to Basic Monitoring at one-minute frequency, for an additional charge. Instances with Detailed Monitoring enabled allows data aggregation by Amazon EC2 AMI ID and instance type.

CloudWatch retains metrics for 15 months, free of charge. CloudWatch Metrics supports the following three retention schedules:

- 1 minute datapoints are available for 15 days

- 5 minute datapoints are available for 63 days
- 1 hour datapoints are available for 455 days

You can learn more about CloudWatch at https://aws.amazon.com/cloudwatch/.

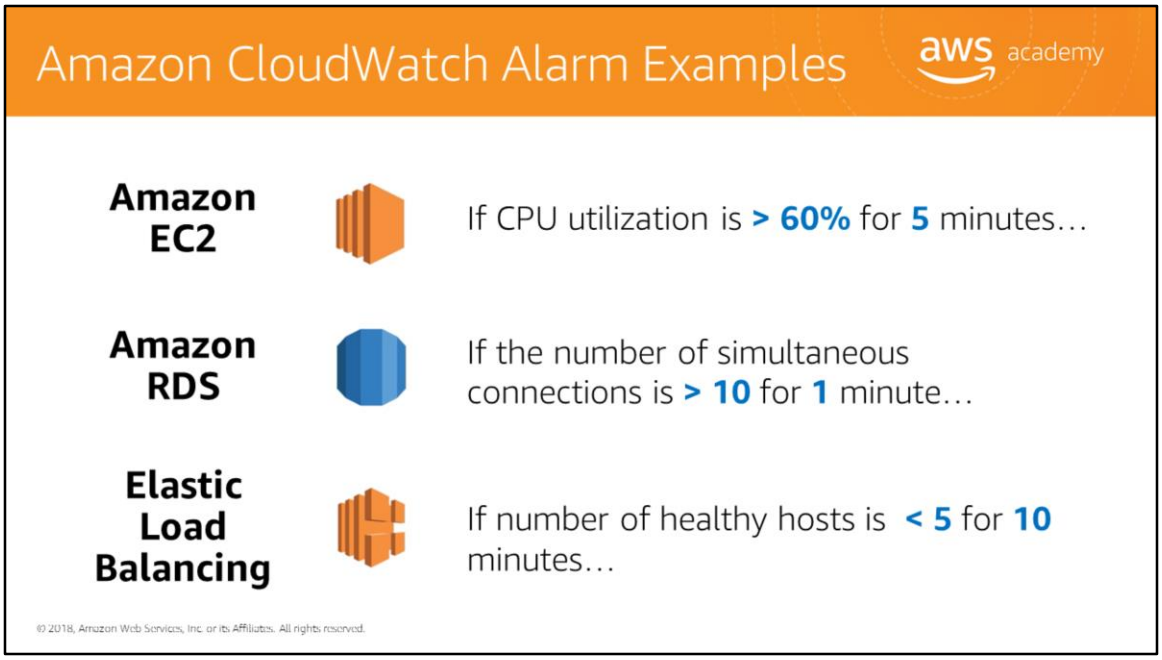


CloudWatch consists of three primary components: metrics, alarms, and events.

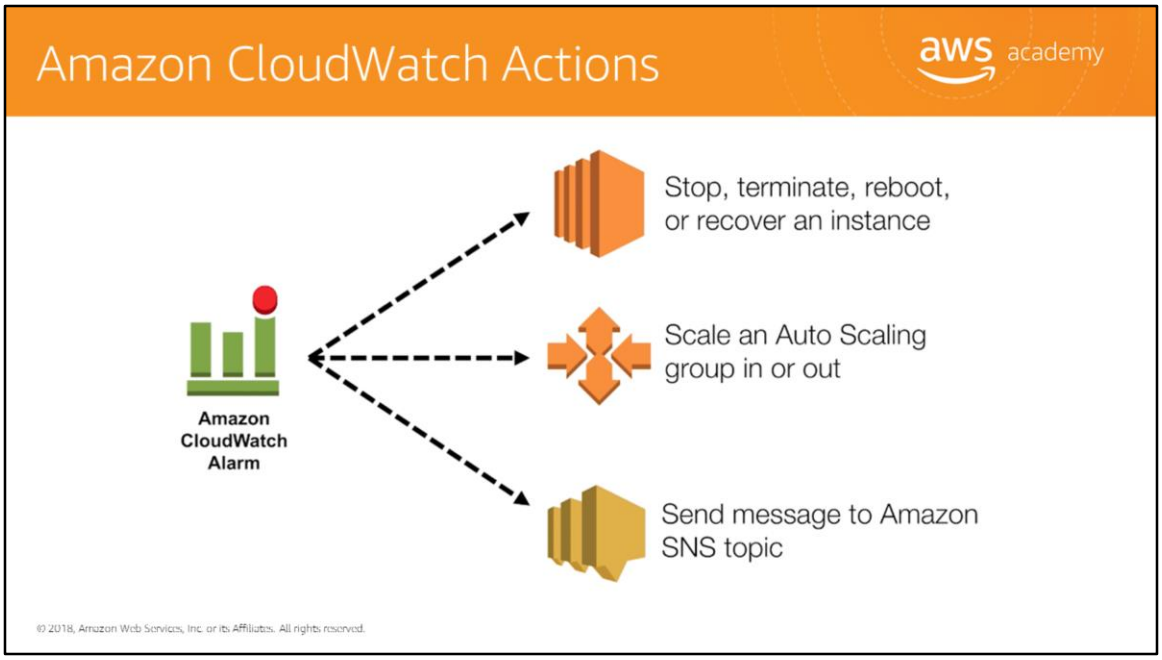
A CloudWatch **metric** is a specific data point from one of the resources or applications that you are monitoring. Many AWS resources submit metrics to CloudWatch automatically as part of the service. You can also capture your own metrics.

A CloudWatch **alarm** sends out a notification message when a tracked metric reaches a specified value for a specified period of time. The notification can be sent to an Amazon SMS topic which then push it to a mobile device or send an email or notify an auto scaling policy to take action.

A CloudWatch **event** can monitor AWS resources and deliver a near real-time stream of events that describe the changes in resources. That stream of resource changes can be sent to other AWS resources.



Here are some examples of CloudWatch alarms.



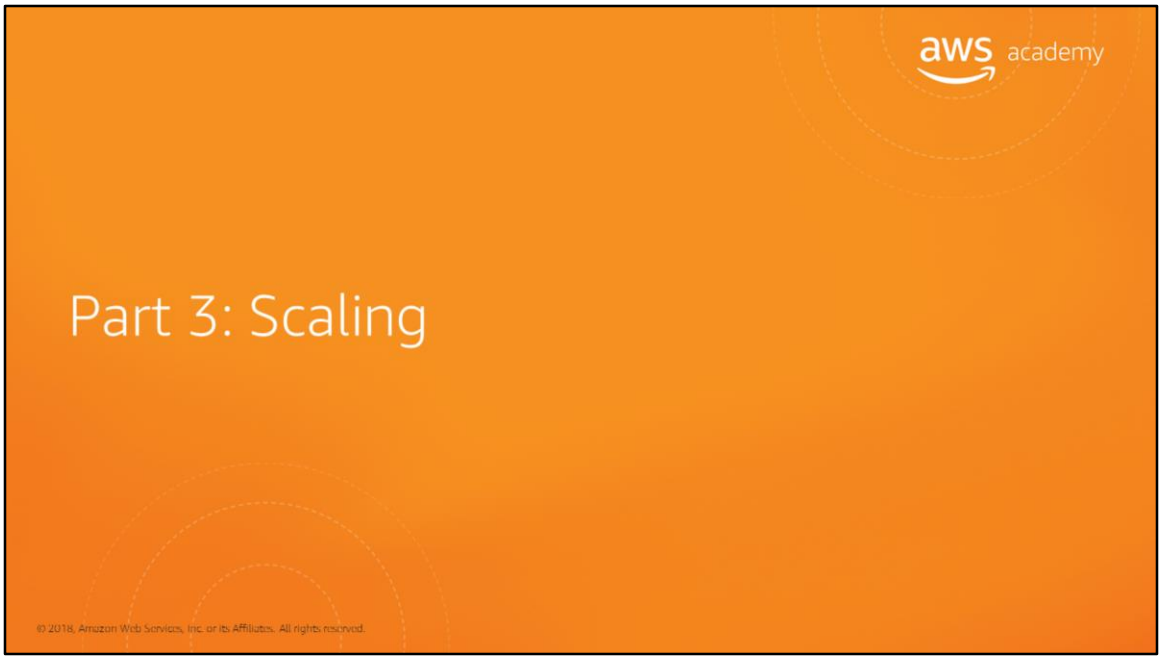
There are a number of actions that you can choose to take based on the CloudWatch alarms.

One of those actions is to automatically scale. Let’s learn what automatic scaling is and how it works.

In Review

- Amazon CloudWatch tracks and monitors the performance and health of your resources and applications
- It enables you to:
  - Track resource and application performance
  - Collect and monitor log files
  - Get notified when an alarm goes off
- CloudWatch consists of three primary components: metrics, alarms, and events.

Amazon CloudWatch



So, what exactly is scaling and why is it important?

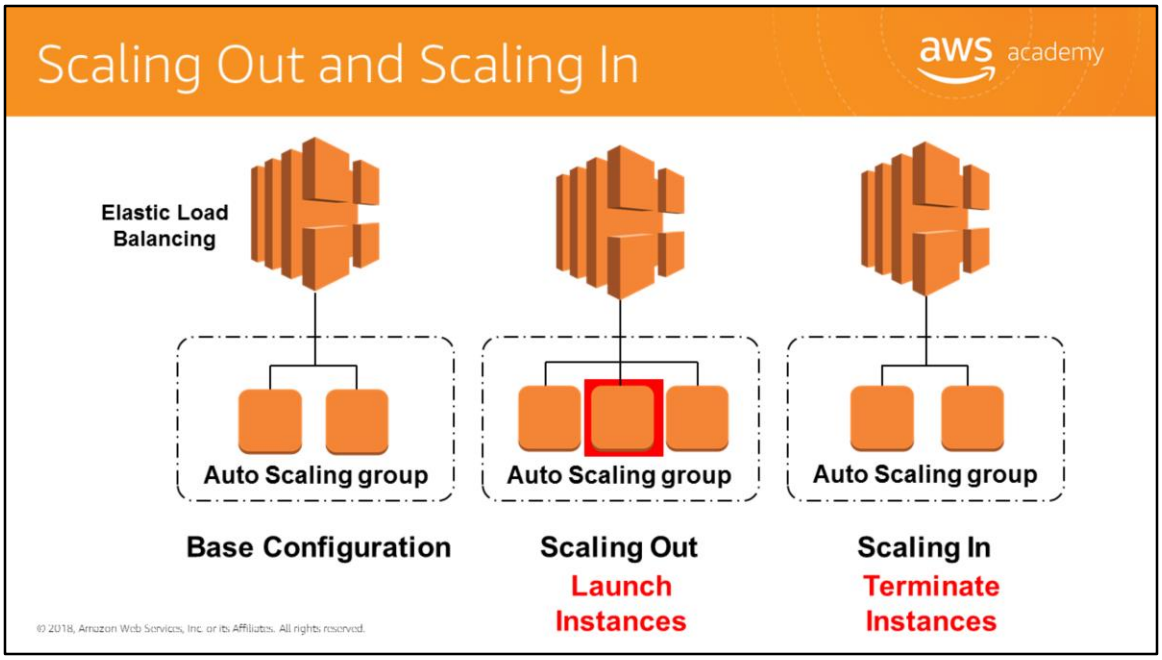
What is Scaling?

Scaling

Scaling provides a simple, powerful user interface that lets you build scaling plans for resources including:

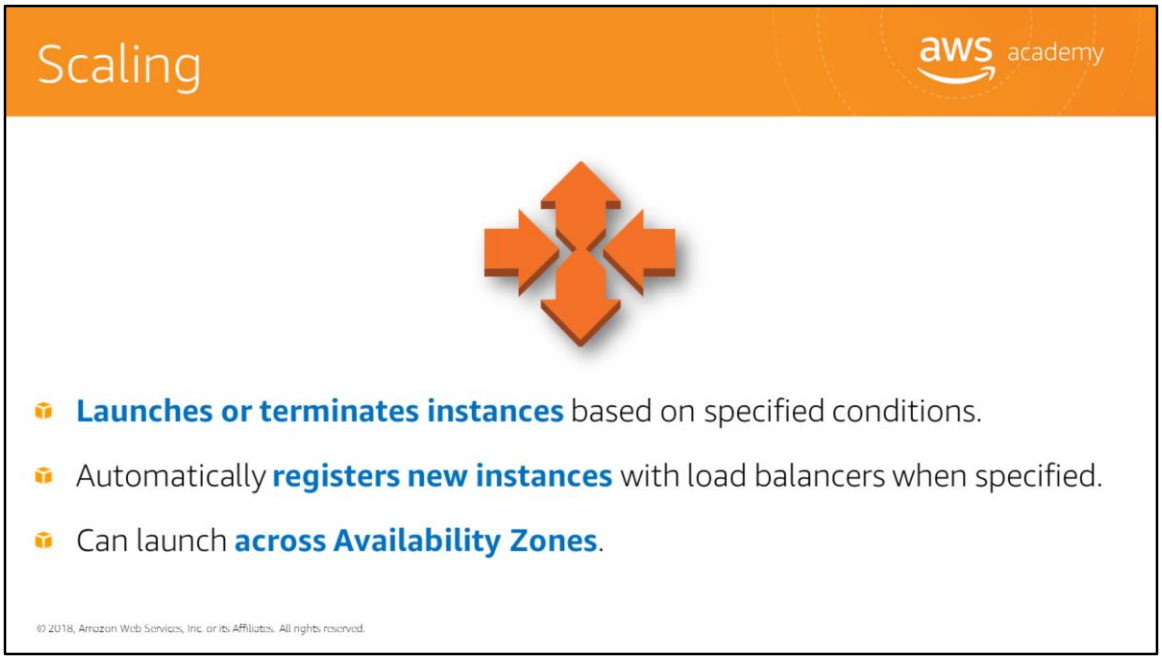
- Amazon EC2 instances and Spot Fleets
- Amazon DynamoDB tables
- Amazon DynamoDB tables and indexes
- Amazon Aurora Replicas

Scaling can be used with a number of AWS resources. With Amazon EC2, Auto Scaling helps you to verify that you have the correct number of Amazon EC2 instances available to handle the load for your application. Using Auto Scaling removes the guesswork of how many resources you need at a point in time to meet your workload requirements.



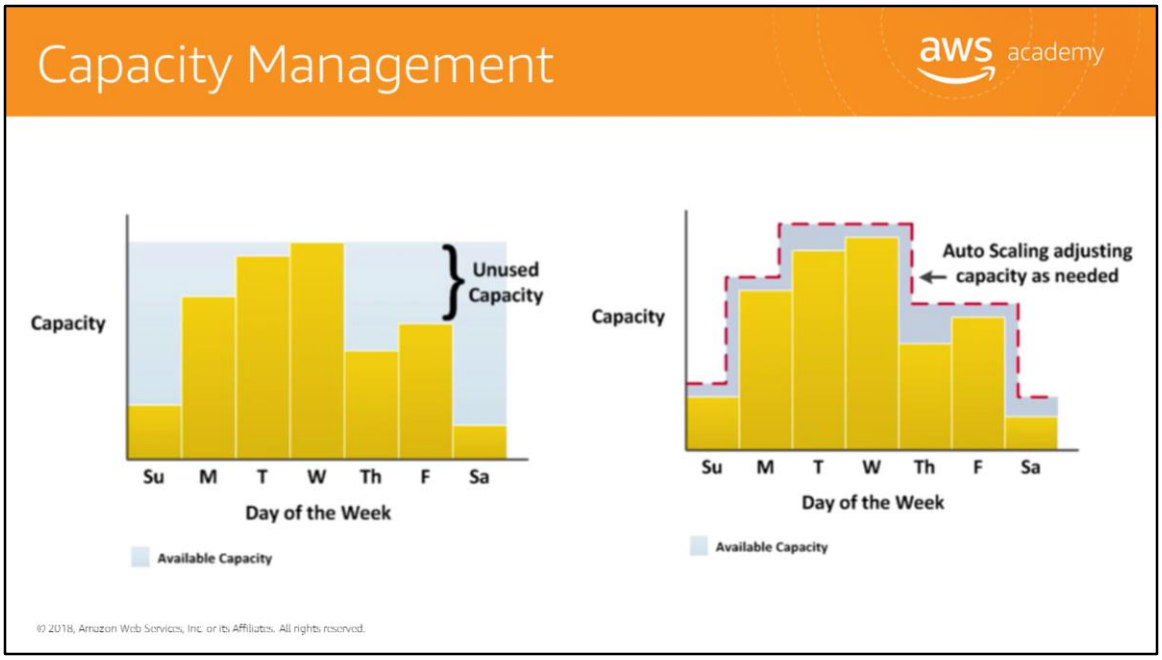
So what exactly do we mean by *scaling*? The first thing we have to do is define the concepts of scaling out and scaling in. If Scaling *adds more instances*, this is termed **scaling out**. When Scaling *terminates instances*, this is **scaling in**.

Scaling can automatically adjust the number of Amazon EC2 instances running in your workload based on either conditions that you defined (for example, CPU utilization over 80%) or on a schedule. Remember, you have control as to what initiates these events.



Scaling launches or terminates instances based on specified conditions. This service is designed to assist you in building a flexible system that can adjust and be modified depending on changes in customer demand.

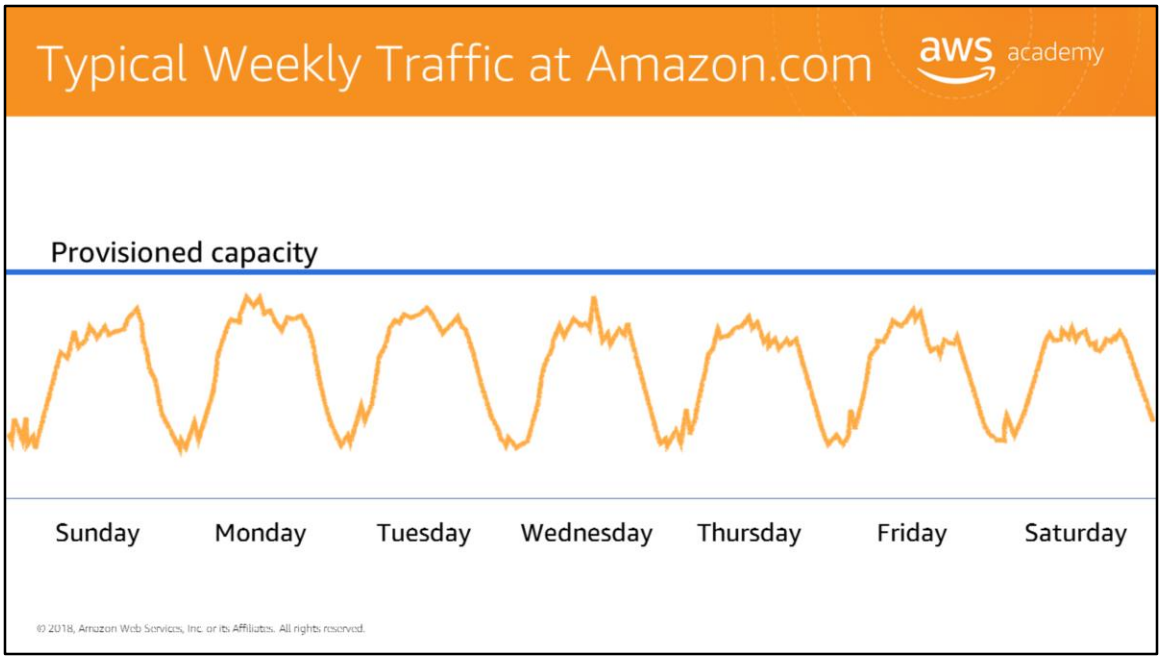
With automatic scaling, you can avoid limitations of being able to create new resources. Instead, you can create new resources on-demand or have scheduled provisioning. If you specify scaling policies, then Scaling can launch or terminate instances as demand on your application increases or decreases. Scaling integrates with ELB to enable you to attach one or more load balancers to an existing Automatic Scaling group. After you attach the load balancer, it automatically registers the instances in the group and distributes incoming traffic across the instances. When one Availability Zone becomes unhealthy or unavailable, a new new instance will be launched in an unaffected Availability Zone. When the unhealthy Availability Zone returns to a healthy state, automatic scaling automatically redistributes the application instances evenly across all of the Availability Zones for your group. Automatic scaling does this by attempting to launch new instances in the Availability Zone with the fewest instances. If the attempt fails, however, automatic scaling attempts to launch in other Availability Zones until it succeeds. This ensures that your applications and systems are always available no matter what the load is.



Let's look at an example workload. We will use CloudWatch to measure Amazon EC2 resource requirements over a standard week. Note that the resource requirements vary with Wednesday requiring the most capacity, and Saturday the least. We could go the route of allocating more than enough Amazon EC2 capacity to always be able to meet our highest demand time, in this case Wednesday. However, this means that we are running resources that will be underutilized most days of the week. This is an option, but our costs are not optimized.

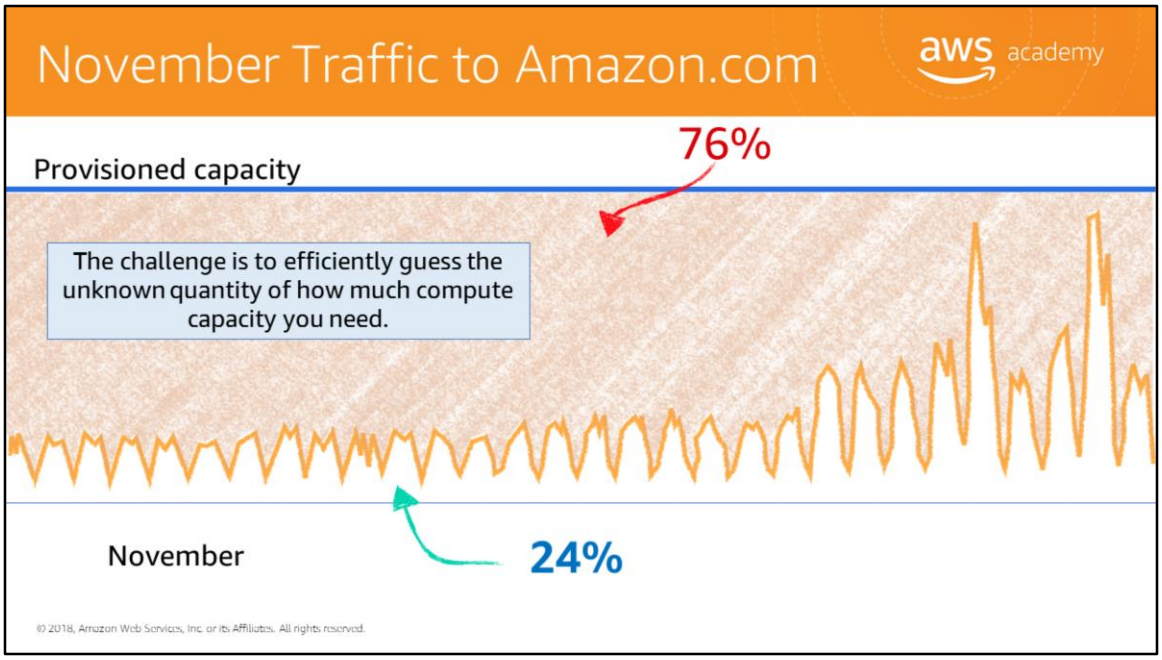
On the other end of the scale, we could allocate fewer Amazon EC2 instances, thus reducing costs. This means that we are under-capacity on certain days. If we don't solve our capacity problem, our application could underperform or potentially even time out for the user. Obviously, this is not a good thing.

Automatic scaling allows you to add or remove Amazon EC2 instances based on conditions that you specify. Automatic scaling is especially powerful in environments with fluctuating performance requirements. This allows you to maintain performance and minimize costs.

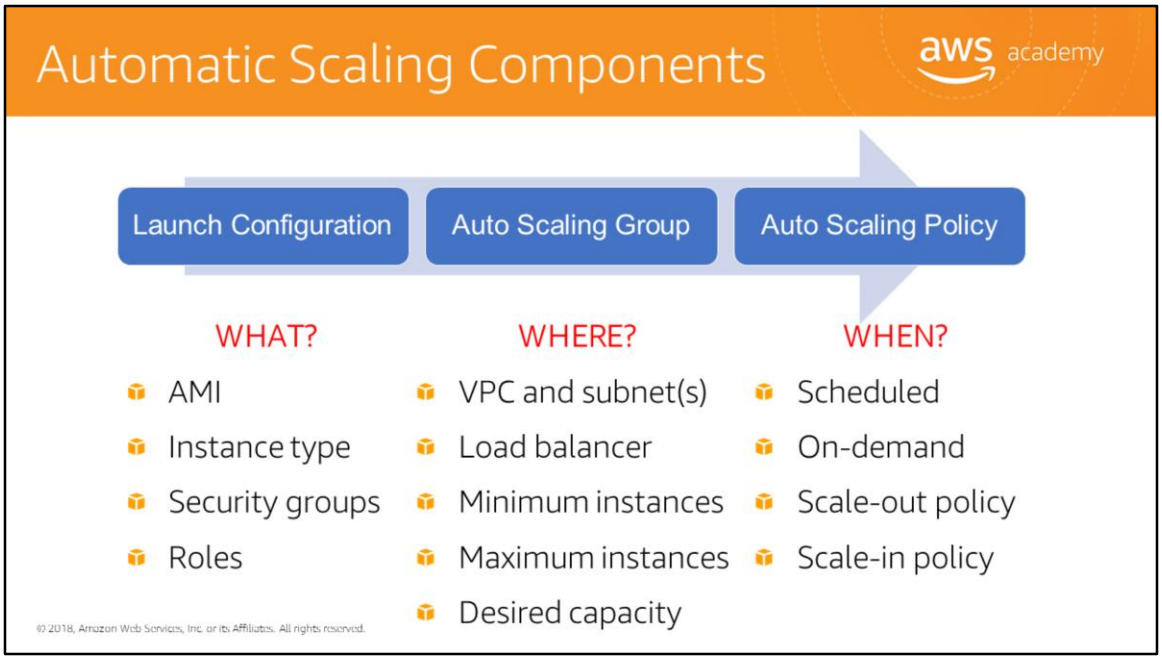


Let's look at another example.

Needless to say, the retail company Amazon.com is one of the largest AWS customers. Typically, the incoming traffic is very predictable. Before Amazon.com moved their infrastructure onto AWS, they had a traditional data center like many other companies. In order to support the peak load, your data center must provide enough hardware and software to support the capacity.



Amazon.com experiences a seasonal peak in November (Black Friday). Because of this peak in late November, they had to invest in enough resources to support this seasonal peak, knowing that this only occurs a certain time of the year. As the business grew, Amazon.com had to keep investing in additional hardware and software. At some point, they ran out of space so they had to add a new data center. The problem is that about 76% of the resources are idle for most of the year. But if you don't have enough compute capability to support the seasonal peak, the server can crash and your business can lose customer confidence.



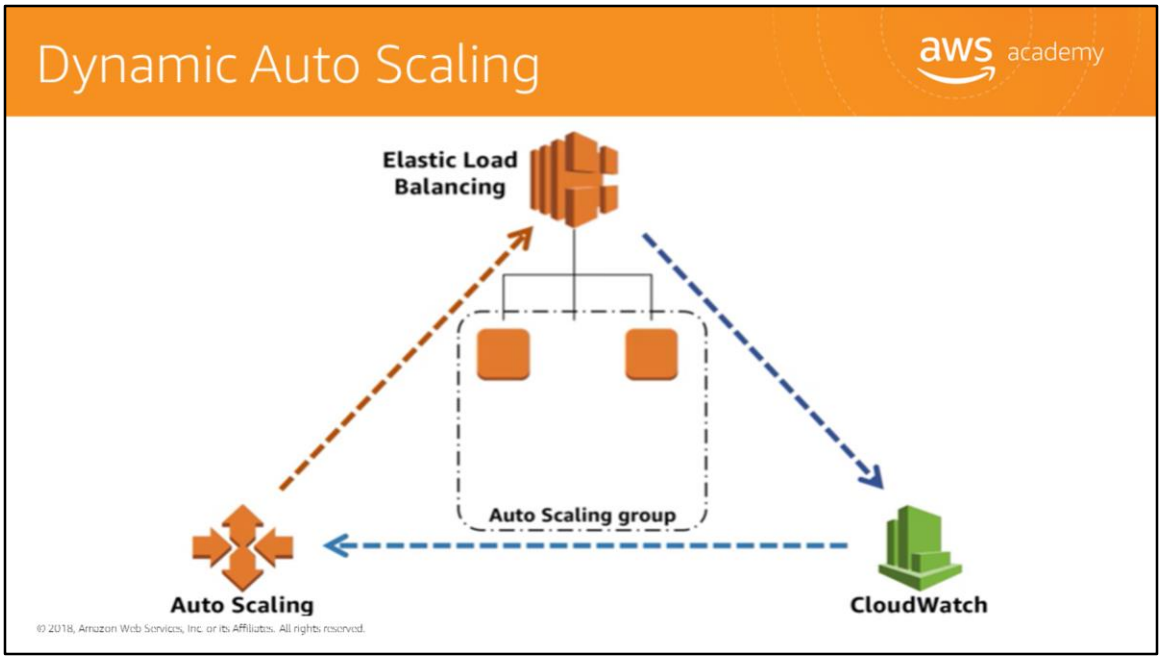
So how do you automatically scale? Three components are required for automatic scaling:
- First, create a launch configuration.
- Next, create an automatic scaling group.
- Then define at least one automatic scaling policy.

What is a launch configuration?
This is about defining **what** will be launched by automatic scaling. Think of all the things that you would specify when you launch an Amazon EC2 instance from the console, such as which Amazon Machine Image (AMI) to use, what instance type, security groups, or roles to apply to the instance.

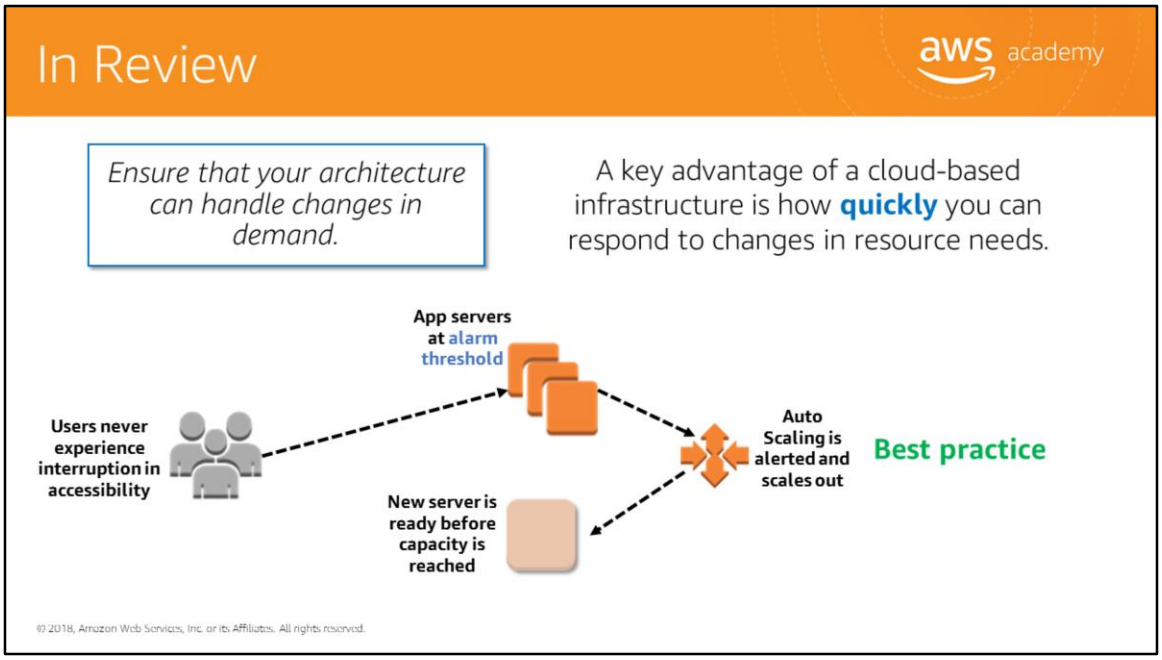
What is an automatic scaling group?
This is about defining **where** the deployment takes place and some boundaries for the deployment. This is where you define which VPC to deploy instances, in which load balancer to interact with. You also specify the boundaries for a group. If you set a minimum of two, if your server account goes below two, another instance will be launched to replace it. If you set the maximum to eight, you will never have more than eight instances in your group. The desired capacity is the number that you wish to start with.

What is an automatic scaling policy?

This is about specifying **when** to launch or terminate Amazon EC2 instances. You can schedule automatic scaling every Thursday at 3:00 p.m., as an example, or create conditions that define thresholds to trigger adding or removing instances. Condition-based policies make your scaling dynamic and able to meet fluctuating requirements. It is best practice to create at least one automatic scaling policy to specify when to scale out and at least one policy to specify when to scale in.



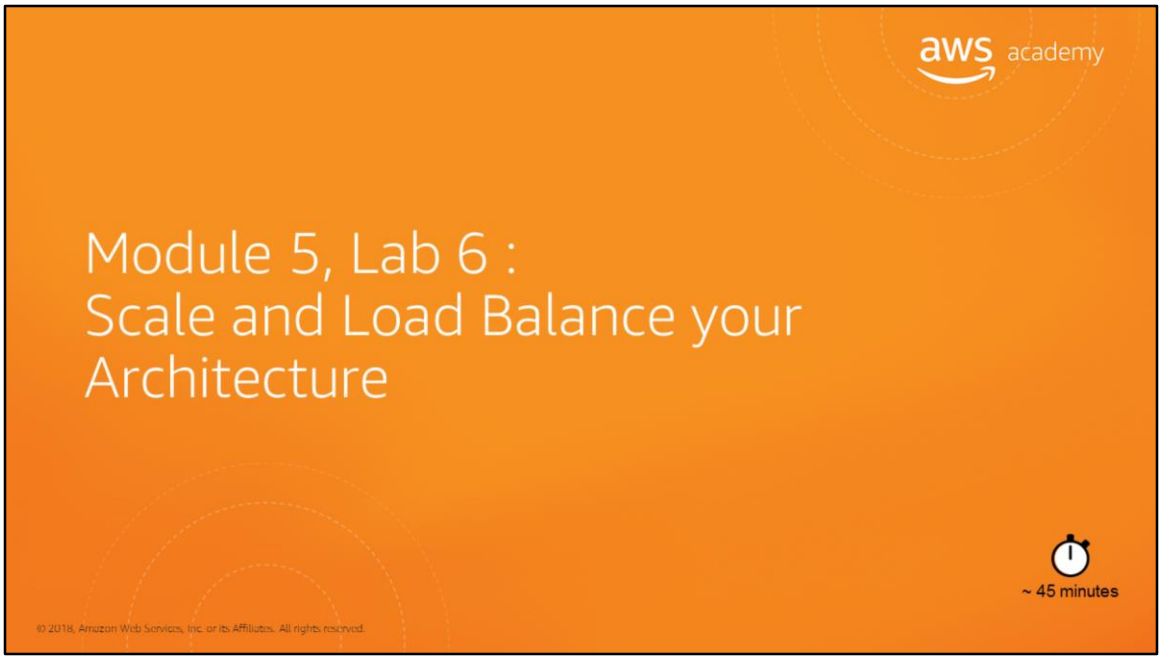
So, how does dynamic automatic scaling work? One common configuration is to create CloudWatch alarms based on performance information from your Amazon EC2 instances or a load balancer. When a performance threshold is breached, a CloudWatch alarm triggers an automatic scaling event which either scales out or scales in Amazon EC2 instances in the environment.



In review, it is a best practice to enable scalability.

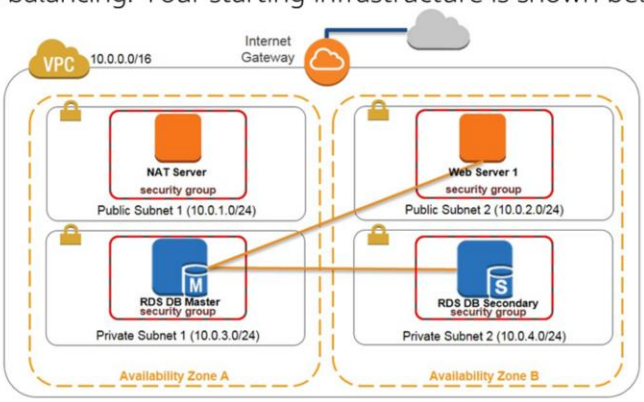
1. Using the best practice of enabling scalability will enable to anticipate need and have more capacity available before it's too late.
2. A monitoring solution (such as Amazon CloudWatch) detects that the total load across the fleet of servers has reached a specified threshold of load. This could be anything, such as "Stayed above 60% CPU utilization for longer than 5 minutes," or anything related to the use of resources. With CloudWatch, you can even design custom metrics based around your specific application, which can trigger scaling in whatever way you need. When that alarm is triggered, Amazon EC2 Auto Scaling immediately launches a new instance.
3. That instance is then ready before capacity is reached, providing a seamless experience for users, who never know that capacity was in danger of being reached.

Ideally, you should also design this system to scale down once demand drops off again, so that you're not running instances that are no longer needed.



Lab 6 Scenario

In this lab, you will start with an infrastructure that needs improvements in terms of scaling and load balancing. Your starting infrastructure is shown below:



ELB distributes incoming application traffic across multiple Amazon EC2 instances. It enables you to achieve fault tolerance in your application by seamlessly providing the required amount of load balancing capacity needed to route application traffic.

ELB offers two types of load balancers that both feature high availability, automatic scaling, and robust security. These are the Classic Load Balancer, which routes traffic based on either application- or network-level information, and the Application Load Balancer, which routes traffic based on advanced application-level information that includes the content of the request. The Classic Load Balancer is ideal for simple load balancing of traffic across multiple Amazon EC2 instances, and the Application Load Balancer is ideal for applications that need advanced routing capabilities, microservices, and container-based architectures. The Application Load Balancer offers you the ability to route traffic to multiple services or load balance across multiple ports on the same Amazon EC2 instance.

Automatic scaling helps you maintain application availability and allows you to scale your Amazon EC2 capacity out or in automatically according to conditions you define. You can use automatic scaling to help ensure that you are running your desired number of Amazon EC2 instances. Automatic scaling can also automatically increase the number of Amazon EC2 instances during demand spikes to maintain performance and decrease capacity during lulls to reduce costs. Automatic scaling is well suited to applications that have stable demand patterns or that experience hourly, daily, or weekly variability in usage.

After completing this lab, you will be able to:

- Create an Amazon Machine Image (AMI) from a running instance

- Create a load balancer.
- Create a launch configuration and an Auto Scaling Group.
- Automatically scale new instances within a private subnet.
- Create Amazon CloudWatch alarms and monitor performance of your infrastructure.

Duration: ~45 minutes

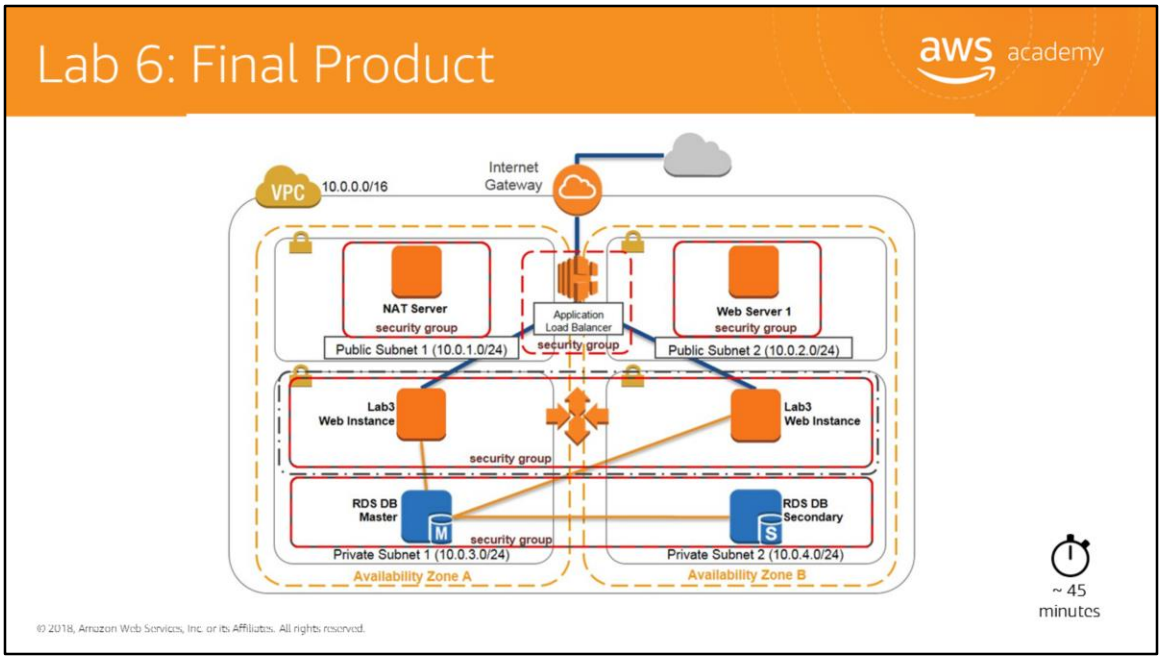
Lab 6: Tasks

Create an **Amazon Machine Image (AMI)** for Auto Scaling.

Create an **Application Load Balancer**.

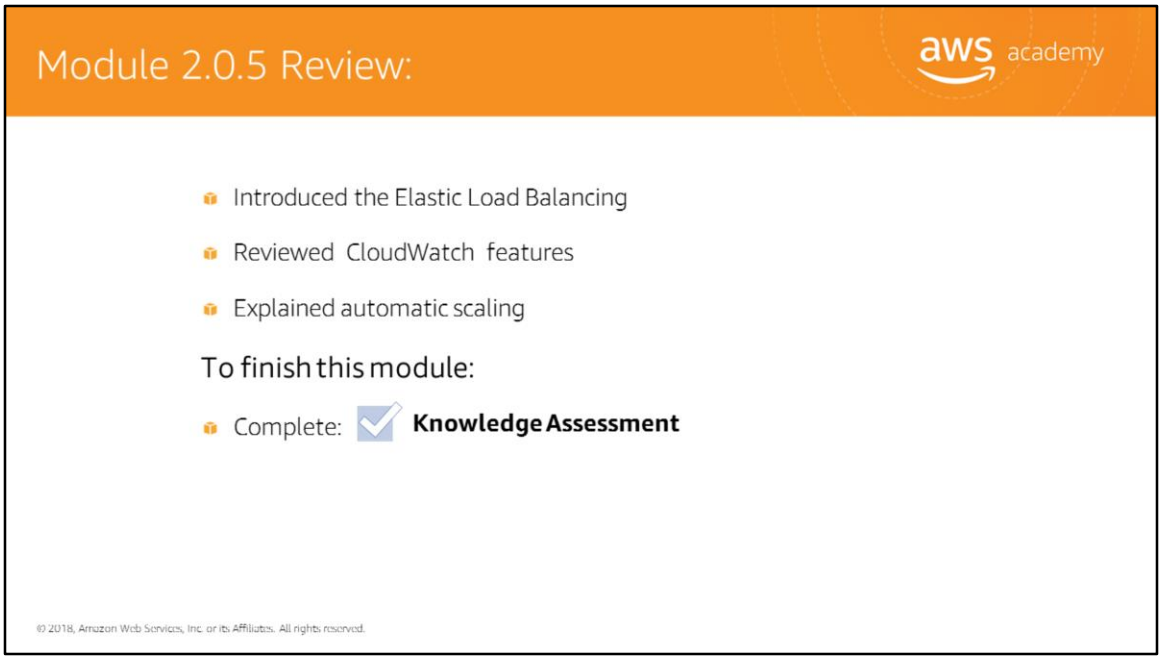
Create a **Launch Configuration** and put it in a security group.

Create and test **automatic scaling** to verify that it is working.



In this lab, you:

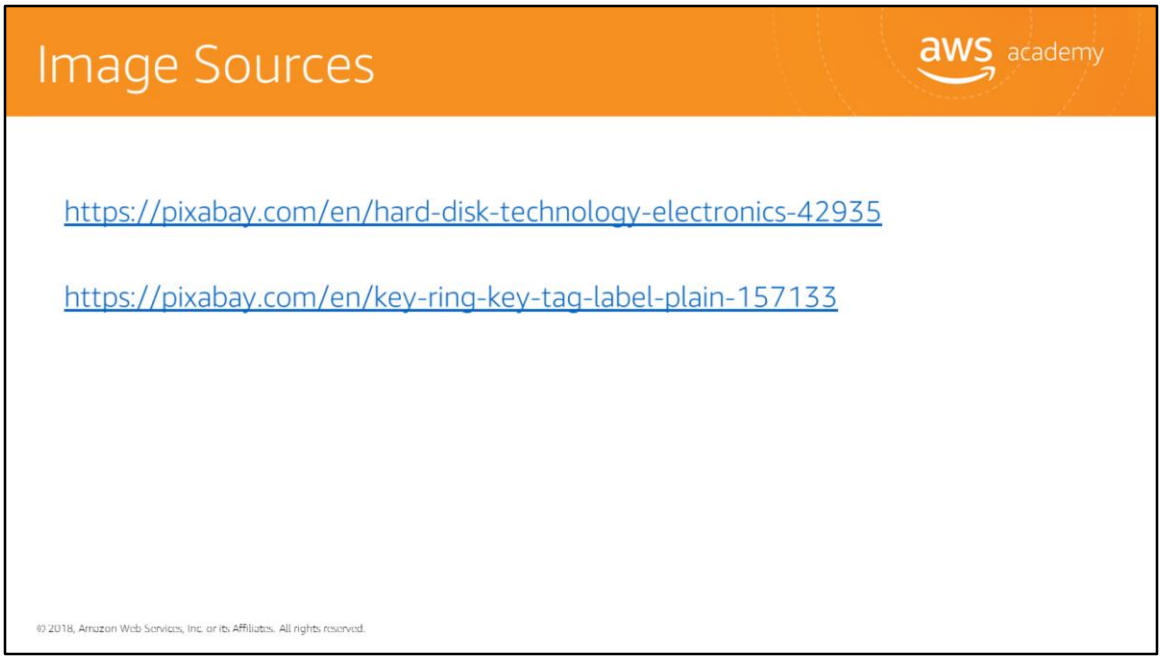
- Created an Amazon Machine Image (AMI) from a running instance
- Created a load balancer.
- Created a launch configuration and an Auto Scaling Group.
- Automatically scaled new instances within a private subnet.
- Created Amazon CloudWatch alarms and monitor performance of your infrastructure.



In review, we:
- Reviewed Elastic Load Balancing
- Briefly reviewed Amazon CloudFront and the valuable information it provides for monitoring your resources
- Explained automatic scaling

To finish this module, please complete the lab and the corresponding knowledge assessment.

In Module 4 we review important aspects of cloud security.



This slide contains attributions for any Creative Commons-licensed images used within this module.

Thanks for participating!

aws academy