Module-1 AWS Essential training

26 April 2024 09:49

Cloud computing is the on demand delivery of IT resources with primarily pay as you go pricing.

Cloud computing deployment models:

1. On premise
2. Cloud
3. Hybrid

1. On premises
    a. Earlier company and organization hosted and maintained hardware such as compute , storage and networking equipment in their own data canters
    b. They often allocated entire infrastructure departments to take care of their data canters which resulted in **costly operations** that made some workloads and **experimentation impossible** .
    c. For some and organisations , the **cost of maintaining a large physical presence was unsustainable .**

2. Cloud
    a. Cloud computing is the on-demand delivery of IT resources over the internet with primarily pay-as-you-go pricing.
    b. With cloud computing, **companies do not have to manage and maintain their own hardware and data centers**.
    c. Instead, **companies like Amazon Web Services (AWS) own and maintain data centers and provide virtual data center technologies and services to companies and users over the internet**.

3. Hybrid
    a. This type of deployment is a way to connect infrastructure and applications between cloud-based resources and existing resources that are not located in the cloud.
    b. The most common method of hybrid deployment between the cloud and existing on-premises infrastructure connects cloud resources to internal systems to extend and grow an organization's infrastructure into the cloud.

Advantages of cloud computing

1. Pas as you go:
2. Benefit from massive economies of scale: By using cloud computing, you can achieve a lower cost than you can get on your own. Because usage from hundreds of thousands of customers is aggregated in the cloud, AWS can achieve higher economies of scale, which translates into lower pay-as-you-go prices.
3. Stop guessing capacity: Stop guessing on your infrastructure capacity needs. When you make a capacity decision before deploying an application, you often end up either sitting on expensive idle resources or dealing with limited capacity. With cloud computing, these problems go away. You can access as much or as little capacity as you need, and scale up and down as required with only a few minutes notice.
4. Increased speed and agility: IT resources are only a click away, which means that you reduce the time to make resources available to developers from weeks to minutes. This results in a dramatic increase in agility for the organization, because the cost and time it takes to experiment and develop is significantly lower.
5. Realize cost savings: Companies can focus on projects that differentiate their business and remove the "undifferentiated heavy lifting", instead of maintaining data centers. With cloud computing, you can focus on your customers, rather than racking, stacking, and powering physical infrastructure.
6. Go global in minutes : Applications can be deployed in multiple Regions around the world with a few clicks. This means that you can provide lower latency and a better experience for your customers at a minimal cost.

Module-1 AWS Global Infrastructure

26 April 2024 10:52

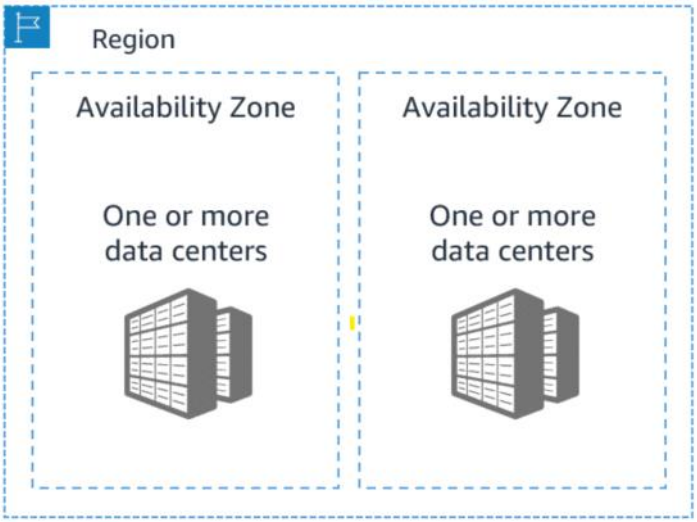
AWS Regions are independent from one another. Without explicit customer consent and authorization, data is not replicated from one Region to another. When you decide which AWS Region to host your applications and workloads, consider four main aspects: latency, price, service availability, and compliance.

1. **Latency** : If your application is sensitive to latency (the delay between a request for data and the response), choose a Region that is close to your user base. This helps prevent long wait times for your customers. Synchronous applications such as gaming, telephony, WebSockets, and Internet of Things (IoT) are significantly affected by high latency. Asynchronous workloads, such as ecommerce applications, can also suffer from user connectivity delays.
2. **Pricing**: Due to the local economy and the physical nature of operating data centers, prices vary from one Region to another. Internet connectivity, imported equipment costs, customs, real estate, and other factors impact a Region's pricing. Instead of charging a flat rate worldwide, AWS charges based on the financial factors specific to each Region.
3. **Data compliance**: Enterprise companies often must comply with regulations that require customer data to be stored in a specific geographic territory. If applicable, choose a Region that meets your compliance requirements.

Availability Zones

1. Inside every Region is a cluster of Availability Zones.
2. An Availability Zone consists of one or more data centers with redundant power, networking, and connectivity.
3. These data centers operate in discrete facilities in undisclosed locations.
4. They are connected using redundant high-speed and low-latency links.

Availability Zones also have code names. Because they are located inside Regions, they can be addressed by appending a letter to the end of the Region code name. Here are examples of Availability Zone codes:
us-east-1a is an Availability Zone in us-east-1 (N. Virginia Region).
sa-east-1b is an Availability Zone in sa-east-1 (São Paulo Region).

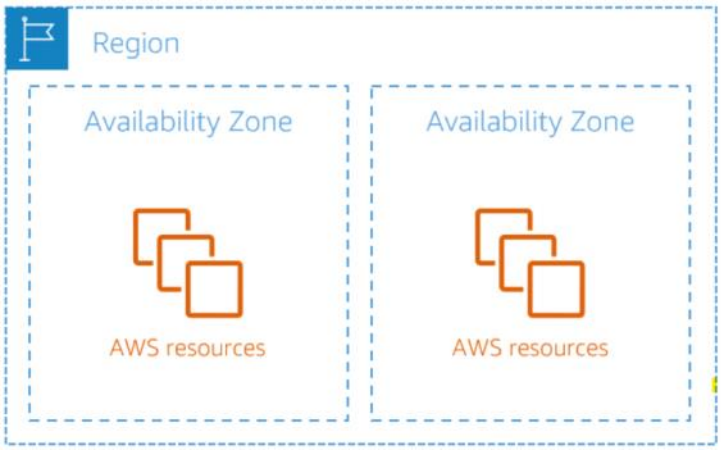


Scope of AWS services

1. Depending on the AWS service that you use, your resources are either deployed at the Availability Zone, Region, or Global level. Each service is different, so you must understand how the scope of a service might affect your application architecture.
2. When you operate a Region-scoped service, you only need to select the Region that you want to use. If you are not asked to specify an individual Availability Zone to deploy the service in, this is an indicator that the service operates on a Region-scope level. For Region-scoped services, AWS automatically performs actions to increase data durability and availability.
3. On the other hand, some services ask you to specify an Availability Zone. With these services, you are often responsible for increasing the data durability and high availability of these resources.

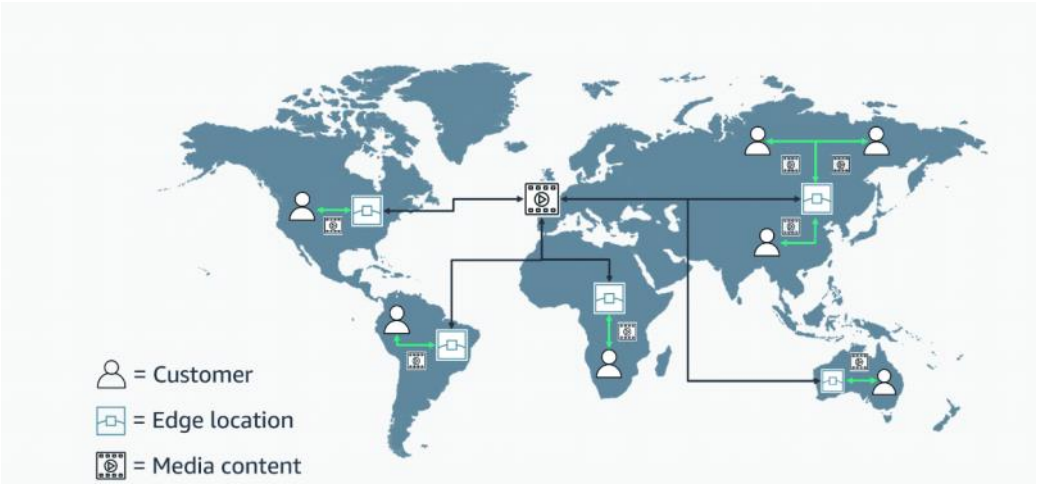
Maintaining resiliency

1. To keep your application available, you must maintain high availability and resiliency.
2. A well-known best practice for cloud architecture is to use Region-scoped, managed services. These services come with availability and resiliency built in. When that is not possible, make sure your workload is replicated across multiple Availability Zones.
3. At a minimum, you should use two Availability Zones. That way, if an Availability Zone fails, your application will have infrastructure up and running in a second Availability Zone to take over the traffic.



Edge locations

1. Edge locations are global locations where content is cached. For example, if your media content is in London and you want to share video files with your customers in Sydney, you could have the videos cached in an edge location closest to Sydney. This would make it possible for your customers to access the cached videos more quickly than accessing them from London. Currently, there are over 400+ edge locations globally.



1. **Amazon CloudFront** delivers your content through a worldwide network of edge locations. When a user requests content that is being served with CloudFront, the request is routed to the location that provides the lowest latency. So that content is delivered with the best possible performance.
2. CloudFront speeds up the distribution of your content by routing each user request through the AWS backbone network to the edge location that can best serve your content.

Module-1 Interacting with AWS

30 April 2024 16:24

Every action that you make in AWS is an API call that is authenticated and authorized.

When the infrastructure becomes virtual, the way that I work with that infrastructure has to change a bit. Instead of physically managing my infrastructure, now I logically manage it through the AWS Application Program Interface, or API. So now when I create, delete, or change any AWS resource, whether it's a virtual server or a storage system for employee photos, I use API calls to AWS to do that.

You can make these API calls in several ways, but the three main ways we're going to talk about in AWS are the

1. **AWS Management Console,**
2. **AWS Command Line Interface, and**
3. **AWS Software Development Kits, or SDKs.**

When people are first getting started with AWS, they typically use the AWS Management Console. This is a web-based method that you log into from your browser. The great thing about the console is that you can point and click. By simply clicking and following prompts, you can get started with some of these services without any previous knowledge of the service.

With the console, there's no need to worry about scripting or finding the proper syntax. When you log into the console, the landing page will show you services you've recently worked with but you can also choose to view all of the possible services organized into relevant categories such as compute, database storage, and more. If I change the Region to Paris, you're making requests to eu-west-3.console.aws.amazon.com or the Paris Region's web console. After you work with the console for a while, you may want to move away from the manual creation of resources.

For example, in the console, you have to go through multiple screens to set configurations to create a virtual machine. And if I wanted to create a second virtual machine I would need to go through that process all over again. While this is helpful, it also leaves room for human error. I could easily miss a checkbox or misspell something or even skip important settings by accident. So, when you get more familiar with AWS, or if you're working in a production environment that requires a degree of risk management, you should move to a tool that enables you to script or program these API calls. One of these tools is called the AWS Command Line Interface, or CLI. You can use this tool in a couple of ways. One is to download the tool and then use the terminal on your machine to create and configure AWS services. Another is to access the CLI through the use of AWS CloudShell, which can be done through the console. With both of these options, instead of having a GUI like the console to interact with, you run create commands using a defined AWS syntax.

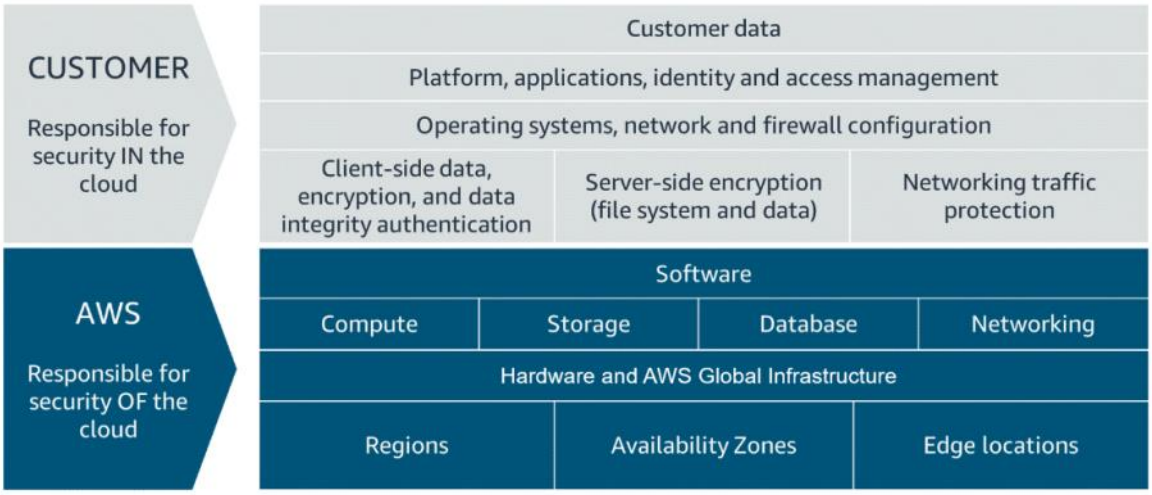
The other tool that allows you to interact with the AWS API programmatically is the AWS Software Development Kits, or SDKs. SDKs are created and maintained by AWS for the most popular programming languages such as Python, Java, Node.js, .NET, Ruby, and more. This comes in handy when you want to integrate your application source code with AWS services. For example, our employee directory application runs using Python and Flask. If I wanted to store all of the employee photos including pictures of employees in an AWS storage service, I could use the Python SDK to write code to interact with that AWS storage service. The ability of managing AWS services from a place where you can run source code with conditions, loops, arrays, lists, and other programming elements provides a lot of power and creativity. Alright, that wraps this video up. To recap, you have three main options to connect with AWS, the console, the CLI, and the SDKs. In this course we will mainly be using the console to interact with the services but feel free to challenge yourself by using the CLI if you're a bit more advanced.

Module-1 Security and the AWS Shared Responsibility Model

30 April 2024 16:54

Security and compliance are a shared responsibility between AWS and you.

When you work with the AWS Cloud, managing security and compliance is a shared responsibility between AWS and you. To depict this shared responsibility, AWS created the shared responsibility model. The distinction of responsibility is commonly referred to as security of the cloud as compared to security in the cloud.



AWS responsibility

AWS is responsible for security of the cloud. This means that AWS protects and secures the infrastructure that runs the services offered in the AWS Cloud. AWS is responsible for the following:

1. Protecting and securing AWS Regions, Availability Zones, and data centers, down to the physical security of the buildings
2. Managing the hardware, software, and networking components that run AWS services, such as the physical servers, host operating systems, virtualization layers, and AWS networking components

The level of responsibility that AWS has depends on the service. AWS classifies services into two categories. The following table provides information about each, including the AWS responsibility.

AWS responsibility

**Note: For users with screen readers, use table mode to read the table.

Category	Examples of AWS Services in the Category	AWS Responsibility
Infrastructure services	Compute services, such as Amazon Elastic Compute Cloud (Amazon EC2)	AWS manages the underlying infrastructure and foundation services.
Abstracted services	Services that require very little management from the customer, such as Amazon Simple Storage Service (Amazon S3)	AWS operates the infrastructure layer, operating system, and platforms, in addition to server-side encryption and data protection.

Customer responsibility

Customers are responsible for security in the cloud. When using any AWS service, the customer is responsible for properly configuring the service and their applications, in addition to ensuring that their data is secure.

The customers' level of responsibility depends on the AWS service. Some services require the customer to perform all the necessary security configuration and management tasks. Other more abstracted services require customers to only manage the data and control access to their resources. Using the two categories of AWS services, customers can determine their level of responsibility for each AWS service that they use.

Customer responsibility

**Note: For users with screen readers, use table mode to read the table.

Category	Examples of AWS Services in the Category	Customer Responsibility
Infrastructure services	Compute services, such as Amazon Elastic Compute Cloud (Amazon EC2)	Customers' control the operating system and application platform, in addition to encrypting, protecting, and managing customer data.
Abstracted services	Services that require very little management from the customer, such as Amazon Simple Storage Service (Amazon S3)	Customers' are responsible for customer data, encrypting the data, and protecting it through network firewalls and backups.

Module-1 Protecting the AWS Root User

15 May 2024 10:34

AWS root user

When you first create an AWS account, you begin with a single sign-in identity that has complete access to all AWS services and resources in the account. This identity is called the AWS root user and is accessed by signing in with the email address and password that were used to create the account.

AWS root user credentials

The AWS root user has two sets of credentials associated with it. One set of credentials is the email address and password that were used to create the account. This allows you to access the AWS Management Console. The second set of credentials is called access keys, which allow you to make programmatic requests from the AWS Command Line Interface (AWS CLI) or AWS API.

Access keys consist of two parts:

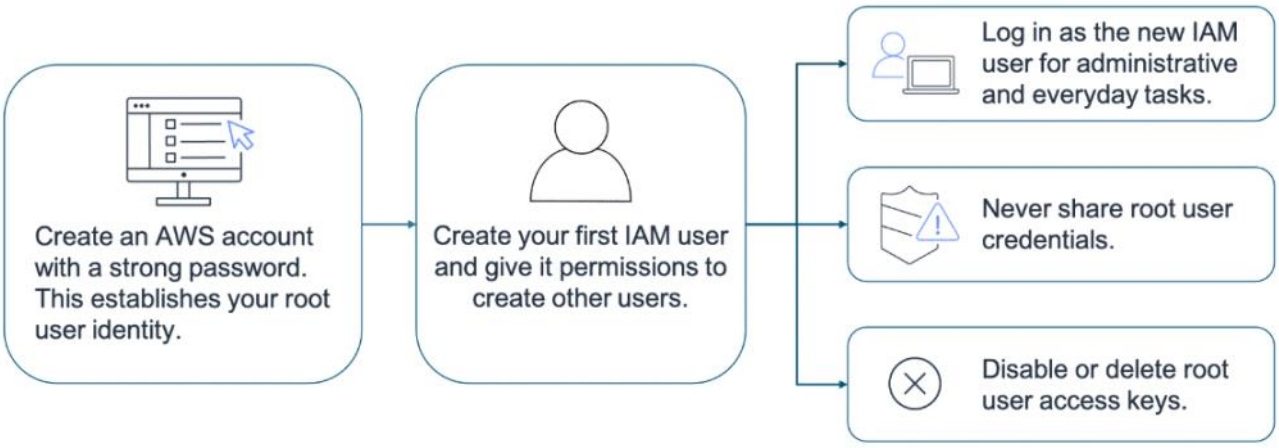
Access key ID: for example, A2lAl5EXAMPLE
Secret access key: for example, wJalrFE/KbEKxE

Similar to a user name and password combination, you need both the access key ID and secret access key to authenticate your requests through the AWS CLI or AWS API. Access keys should be managed with the same security as an email address and password.

AWS root user best practices

The root user has complete access to all AWS services and resources in your account, including your billing and personal information. Therefore, you should securely lock away the credentials associated with the root user and not use the root user for everyday tasks. Visit the links at the end of this lesson to learn more about when to use the AWS root user.

1. To ensure the safety of the root user, follow these best practices:
2. Choose a strong password for the root user.
3. Enable multi-factor authentication (MFA) for the root user.
4. Never share your root user password or access keys with anyone.
5. Disable or delete the access keys associated with the root user.
6. Create an Identity and Access Management (IAM) user for administrative tasks or everyday tasks.

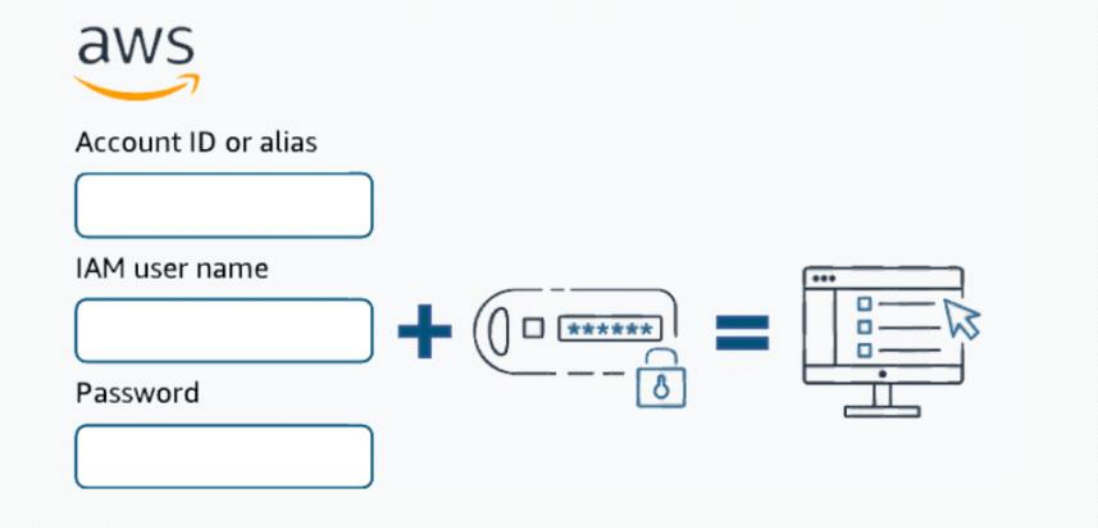


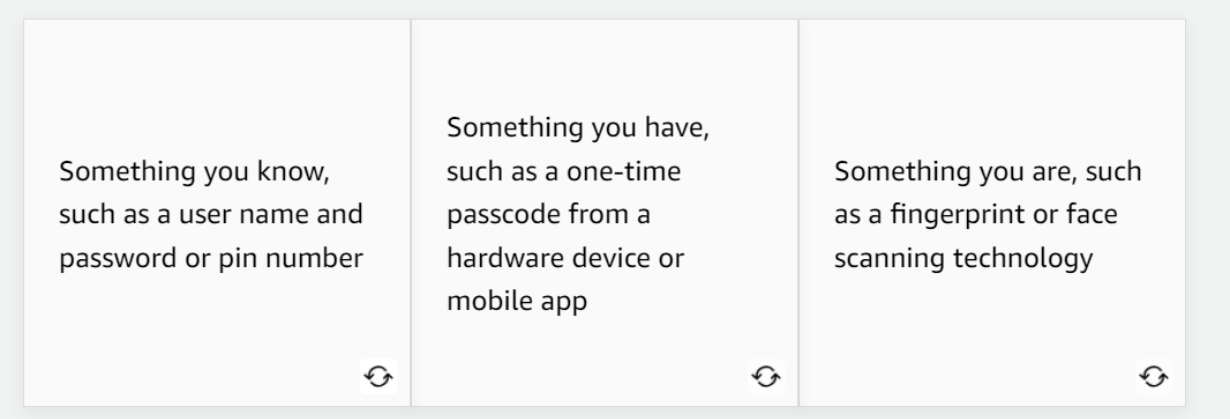
Multi-factor authentication

When you create an AWS account and first log in to the account, you use single-factor authentication. Single-factor authentication is the simplest and most common form of authentication. It only requires one authentication method. In this case, you use a user name and

password to authenticate as the AWS root user. Other forms of single-factor authentication include a security pin or a security token.

However, sometimes a user's password is easy to guess. For example, your coworker Bob's password, IloveCats222, might be easy for someone who knows Bob personally to guess, because it's a combination of information that is easy to remember and includes certain facts about Bob (Bob loves cats, and his birthday is February 22). If a bad actor guessed or cracked Bob's password through social engineering, bots, or scripts, Bob might lose control of his account. Unfortunately, this is a common scenario that users of any website often face. This is why using multi-factor authentication (MFA) is important in preventing unwanted account access.





With a combination of this information, systems can provide a layered approach to account access. So even if the first method of authentication, like Bob's password, is cracked by a malicious actor, the second method of authentication, such as a fingerprint, provides another level of security. This extra layer of security can help protect your most important accounts, which is why you should activate MFA on your AWS root user.

MFA on AWS

If you activate MFA on your root user, you must present a piece of identifying information from both the something you know category and the something you have category. The first piece of identifying information the user enters is an email and password combination. The second piece of information is a temporary numeric code provided by an MFA device.

Using MFA adds an additional layer of security because it requires users to use a supported MFA mechanism in addition to their regular sign-in credentials. Activating MFA on the AWS root user account is an AWS best practice.

Supported MFA devices

AWS supports a variety of MFA mechanisms, such as virtual MFA devices, hardware time-based one-time password (TOTP) tokens, and FIDO security keys. To learn more, take a look at the table below. For instructions on how to set up each method, see the Resources section.

Device	Description	Supported Devices
Virtual MFA	A software app that runs on a phone or other device that provides a one-time passcode. These applications can run on unsecured mobile devices, and because of that, they might not provide the same level of security as hardware or FIDO security keys.	Twilio Authy Authenticator, Duo Mobile, LastPass Authenticator, Microsoft Authenticator, Google Authenticator, Symantec VIP
Hardware TOTP token	A hardware device, generally a key fob or display card device, that generates a one-time, six-digit numeric code based on the time-based one-time password (TOTP) algorithm.	Key fob, display card
FIDO security keys	FIDO-certified hardware security keys are provided by third-party providers such as Yubico. You can plug your FIDO security key into a USB port on your computer and enable it using the instructions that follow.	FIDO Certified products

Module-1 AWS Identity and Access Management

15 May 2024 11:36

Authentication and authorization

When you configure access to any account, two terms come up frequently: authentication and authorization. Although these terms might seem basic, you must fully understand them to properly configure access management on AWS.

Authentication

When you create your AWS account, you use the combination of an email address and a password to verify your identity. If a user types in the correct email address and password, the system assumes the user is allowed to enter and grants them access. This is the process of authentication.

Authentication ensures that the user is who they say they are. User names and passwords are the most common types of authentication. But you might also work with other forms, such as token-based authentication or biometric data, like a fingerprint. Authentication simply answers the question, “Are you who you say you are?”

Authorization

After you’re authenticated and in your AWS account, you might be curious about what actions you can take. This is where authorization comes in. Authorization is the process of giving users permission to access AWS resources and services. Authorization determines whether a user can perform certain actions, such as read, edit, delete, or create resources. Authorization answers the question, “What actions can you perform?”

What is IAM?

AWS Identity and Access Management (IAM) is an AWS service that helps you manage access to your AWS account and resources. It also provides a centralized view of who and what are allowed inside your AWS account (authentication), and who and what have permissions to use and work with your AWS resources (authorization).

With IAM, you can share access to an AWS account and resources without sharing your set of access keys or password. You can also provide granular access to those working in your account, so people and services only have permissions to the resources that they need. For example, to provide a user of your AWS account with read-only access to a particular AWS service, you can granularly select which actions and which resources in that service that they can access.

IAM features

To help control access and manage identities in your AWS account, IAM offers many features to ensure security.

1. IAM is global and not specific to any one Region. You can see and use your IAM configurations from any Region in the AWS Management Console.
2. IAM is integrated with many AWS services by default.
3. You can grant other identities permission to administer and use resources in your AWS account without having to share your password and key.
4. IAM supports MFA. You can add MFA to your account and to individual users for extra security.
5. IAM supports identity federation, which allows users with passwords elsewhere—like your corporate network or internet identity provider—to get temporary access to your AWS account.

6. Any AWS customer can use IAM; the service is offered at no additional charge.

IAM user

An IAM user represents a person or service that interacts with AWS. You define the user in your AWS account. Any activity done by that user is billed to your account. When you create a user, that user can sign in to gain access to the AWS resources inside your account.

You can also add more users to your account as needed. For example, for your cat photo application, you could create individual users in your AWS account that correspond to the people who are working on your application. Each person should have their own login credentials to prevent sharing credentials between users.

IAM user credentials

An IAM user consists of a name and a set of credentials. When you create a user, you can provide them with the following types of access:

1. Access to the AWS Management Console
2. Programmatic access to the AWS CLI and AWS API

To access the console, provide the user with a user name and password. For programmatic access, AWS generates a set of access keys that can be used with the AWS CLI and AWS API. IAM user credentials are considered permanent, which means that they stay with the user until there's a forced rotation by admins.

When you create an IAM user, you can grant permissions directly at the user level. This can seem like a good idea if you have only one or a few users. However, as the number of users increases, keeping up with permissions can become more complicated. For example, if you have 3,000 users in your AWS account, administering access and getting a top-level view of who can perform what actions on which resources can be challenging.

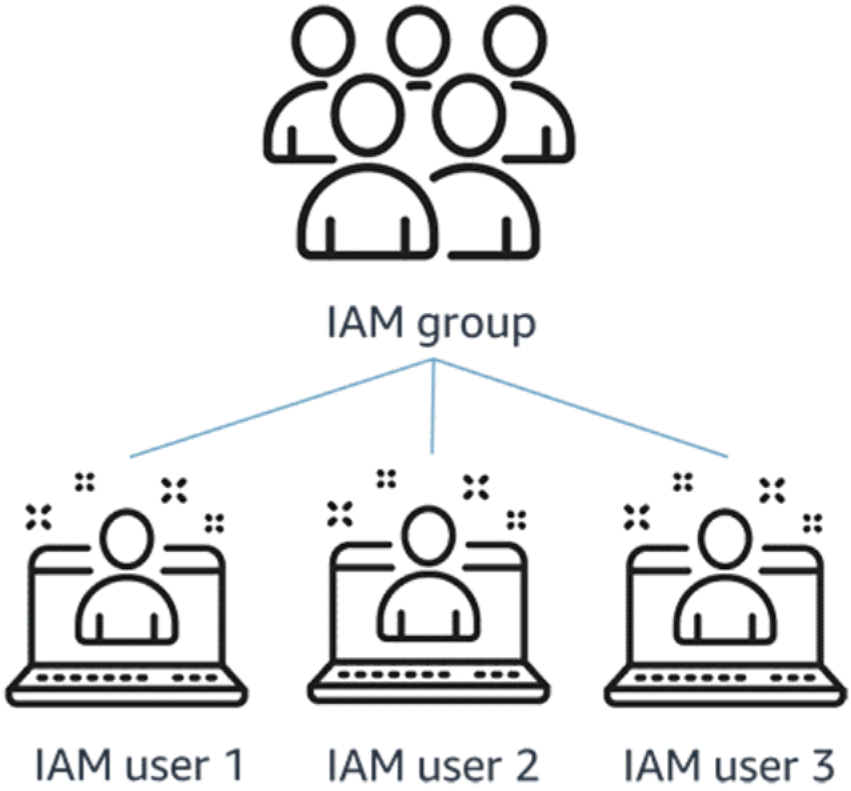
Fortunately, you can group IAM users and attach permissions at the group level.

IAM groups

An IAM group is a collection of users. All users in the group inherit the permissions assigned to the group. This makes it possible to give permissions to multiple users at once. It's a more convenient and scalable way of managing permissions for users in your AWS account. This is why using IAM groups is a best practice.

If you have an application that you're trying to build and you have multiple users in one account working on the application, you might organize the users by job function. For example, you might organize your IAM groups by developers, security, and admins. You could then place all your IAM users into their respective groups.

This provides a way to see who has what permissions in your organization. It also helps you scale when new people join, leave, and change roles in your organization.



Consider the following examples:

1. A new developer joins your AWS account to help with your application. You create a new user and add them to the developer group, without thinking about which permissions they need.
2. A developer changes jobs and becomes a security engineer. Instead of editing the user's permissions directly, you remove them from the old group and add them to the new group that already has the correct level of access.

Keep in mind the following features of groups:

- Groups can have many users.
- Users can belong to many groups.
- Groups cannot belong to groups.

The root user can perform all actions on all resources inside an AWS account by default. This is in contrast to creating new IAM users, new groups, or new roles. To allow an IAM identity to perform specific actions in AWS, such as implement resources, you must grant the IAM user the necessary permissions.

The way you grant permissions in IAM is by using IAM policies.

IAM policies

To manage access and provide permissions to AWS services and resources, you create IAM policies and attach them to an IAM identity. Whenever an IAM identity makes a request, AWS evaluates the policies associated with them. For example, if you have a developer inside the developers group who makes a request to an AWS service, AWS evaluates any policies attached to the developers group and any policies attached to the developer user to determine if the request should be allowed or denied.

IAM policy examples

Most policies are stored in AWS as JSON documents with several policy elements. The following example provides admin access through an IAM identity-based policy.

```
{
"Version": "2012-10-17",
"Statement": [{
"Effect": "Allow",
"Action": "*",
"Resource": "*"
}]
}
```

This policy has four major JSON elements: Version, Effect, Action, and Resource.

The Version element defines the version of the policy language. It specifies the language syntax rules that are needed by AWS to process a policy. To use all the available policy features, include "Version": "2012-10-17" before the "Statement" element in your policies.
The Effect element specifies whether the policy will allow or deny access. In this policy, the Effect is "Allow", which means you're providing access to a particular resource.
The Action element describes the type of action that should be allowed or denied. In the example policy, the action is "*". This is called a wildcard, and it is used to symbolize every action inside your AWS account.
The Resource element specifies the object or objects that the policy statement covers. In the policy example, the resource is the wildcard "*". This represents all resources inside your AWS console.
Putting this information together, you have a policy that allows you to perform all actions on all resources in your AWS account. This is what we refer to as an administrator policy.

The next example shows a more granular IAM policy.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DenyS3AccessOutsideMyBoundary",
      "Effect": "Deny",
      "Action": [
        "s3:*"
      ],
      "Resource": "*",
      "Condition": {
        "StringNotEquals": {
          "aws:ResourceAccount": [
            "222222222222"
          ]
        }
      }
    }
  ]
}
```

This policy uses a Deny effect to block access to Amazon S3 actions, unless the Amazon S3 resource that's being accessed is in account 222222222222. This ensures that any Amazon S3 principals are accessing only the resources that are inside of a trusted AWS account.

IAM best practices

- Lock down the AWS root user

–

The root user is an all-powerful and all-knowing identity in your AWS account. If a malicious user were to gain control of root-user credentials, they would be able to access every resource in your account, including personal and billing information. To lock down the root user, you can do the following:

Don’t share the credentials associated with the root user.
Consider deleting the root user access keys.
Activate MFA on the root account.

- Follow the principle of least privilege

–

Least privilege is a standard security principle that advises you to grant only the necessary permissions to do a particular job and nothing more. To implement least privilege for access control, start with the minimum set of permissions in an IAM policy and then grant additional permissions as necessary for a user, group, or role.

- Use IAM appropiatly

IAM is used to secure access to your AWS account and resources. It provides a way to create and manage users, groups, and roles to access resources in a single AWS account. IAM is not used for website authentication and authorization, such as providing users of a website with sign-in and sign-up functionality. IAM also does not support security controls for protecting operating systems and networks.

- Use IAM when possible

Maintaining roles is more efficient than maintaining users. When you assume a role, IAM dynamically provides temporary credentials that expire after a defined period of time, between 15 minutes and 36 hours. Users, on the other hand, have long-term credentials in the form of user name and password combinations or a set of access keys.

User access keys only expire when you or the account admin rotates the keys. User login credentials expire if you applied a password policy to your account that forces users to rotate their passwords.

- Consider using identity provider

If you decide to make your cat photo application into a business and begin to have more than a handful of people working on it, consider managing employee identity information through an identity provider (IdP). Using an IdP, whether it's with an AWS service such as AWS IAM Identity Center (successor to AWS Single Sign-On) or a third-party identity provider, provides a single source of truth for all identities in your organization.

You no longer have to create separate IAM users in AWS. You can instead use IAM roles to provide permissions to identities that are federated from your IdP. Being federated is a process that allows for the transfer of identity and authentication information across a set of networked systems.

For example, your employee Martha has access to multiple AWS accounts. Instead of creating and managing multiple IAM users named Martha in each of those AWS accounts, you could manage Martha in your company’s IdP. If Martha moves in the company or leaves the company, Martha can be updated in the IdP rather than in every AWS account in the company.

- Regularly review and remove unused users, roles, and other credentials

–

You might have IAM users, roles, permissions, policies, or credentials that you are no longer using in your account. IAM provides last accessed information to help you identify irrelevant credentials that you no longer need so that you can remove them. This helps you reduce the number of users, roles, permissions, policies, and credentials that you have to monitor.

Module-2 Compute as a Service

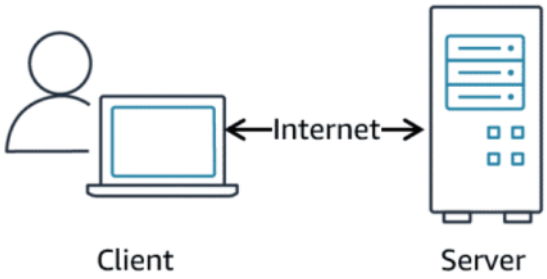
18 May 2024 11:47

At a fundamental level, three types of compute options are available:

1. virtual machines (VMs),
2. container services,
3. serverless.

Servers

The first building block that you need to host an application is a server. Servers can usually handle HTTP requests and send responses to clients following the client-server model. Although any API-based communication also falls under this model.



A client is a person or computer that sends a request. A server handling the requests is a computer, or collection of computers, connected to the internet serving websites to internet users. Servers power your application by providing CPU, memory, and networking capacity to process users' requests and transform them into responses. For context, common HTTP servers include the following:

Windows options, such as Internet Information Services (IIS)
Linux options, such as Apache HTTP Server, Nginx, and Apache Tomcat

To run an HTTP server on AWS, you must find a service that provides compute power in the AWS Management Console. You can view the complete list of AWS compute services when you log in to the console.

Choosing the right compute option

If you're responsible for setting up servers on AWS to run your infrastructure, you have many compute options. First, you need to know which compute service to use for each use case. At a fundamental level, three types of compute options are available: virtual machines (VMs), container services, and serverless.

In AWS, Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides secure and resizable compute capacity in the cloud. You can provision virtual servers called EC2 instances. Behind the scenes, AWS operates and manages the host machines and the hypervisor layer. AWS also installs the virtual machine operating system, called the guest operating system.

Module-2 Getting Started with Amazon EC2

18 May 2024 12:02

When architecting any application for high availability, consider using at least two EC2 instances in two separate Availability Zones.

Amazon EC2

Amazon EC2 is a web service that provides secure, resizable compute capacity in the cloud. With this service, you can provision virtual servers called EC2 instances.

With Amazon EC2, you can do the following:

1. Provision and launch one or more EC2 instances in minutes.
2. Stop or shut down EC2 instances when you finish running a workload.
3. Pay by the hour or second for each instance type (minimum of 60 seconds).

You can create and manage EC2 instances through the AWS Management Console, AWS CLI, AWS SDKs, automation tools, and infrastructure orchestration services.

To create an EC2 instance, you must define the following:

1. **Hardware specifications:** CPU, memory, network, and storage
2. **Logical configurations:** Networking location, firewall rules, authentication, and the operating system of your choice

Amazon Machine Image

When launching an EC2 instance, the first setting you configure is which operating system you want by selecting an Amazon Machine Image (AMI).

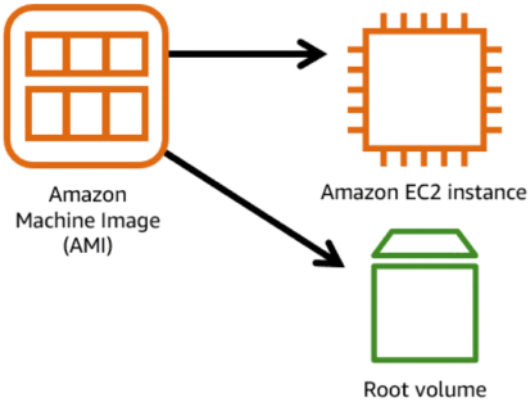
In the traditional infrastructure world, spinning up a server consists of installing an operating system from installation disks, drives, or wizards over the network. In the AWS Cloud, the operating system installation is not your responsibility. Instead, it's built into the AMI that you choose.

An AMI includes the operating system, storage mapping, architecture type, launch permissions, and any additional preinstalled software applications.

Relationship between AMIs and EC2 instances

EC2 instances are live instantiations (or versions) of what is defined in an AMI, as a cake is a live instantiation of a cake recipe. If you are familiar with software development, you can also see this kind of relationship between a class and an object. In this case, the AMI is how you model and define your instance. The EC2 instance is the entity you interact with, where you can install your web server and serve your content to users.

When you launch a new instance, AWS allocates a virtual machine that runs on a hypervisor. Then the AMI that you selected is copied to the root device volume, which contains the image that is used to boot the volume. In the end, you get a server that you can connect to and install packages and additional software on. In the example, you install a web server along with the properly configured source code of your employee directory application.



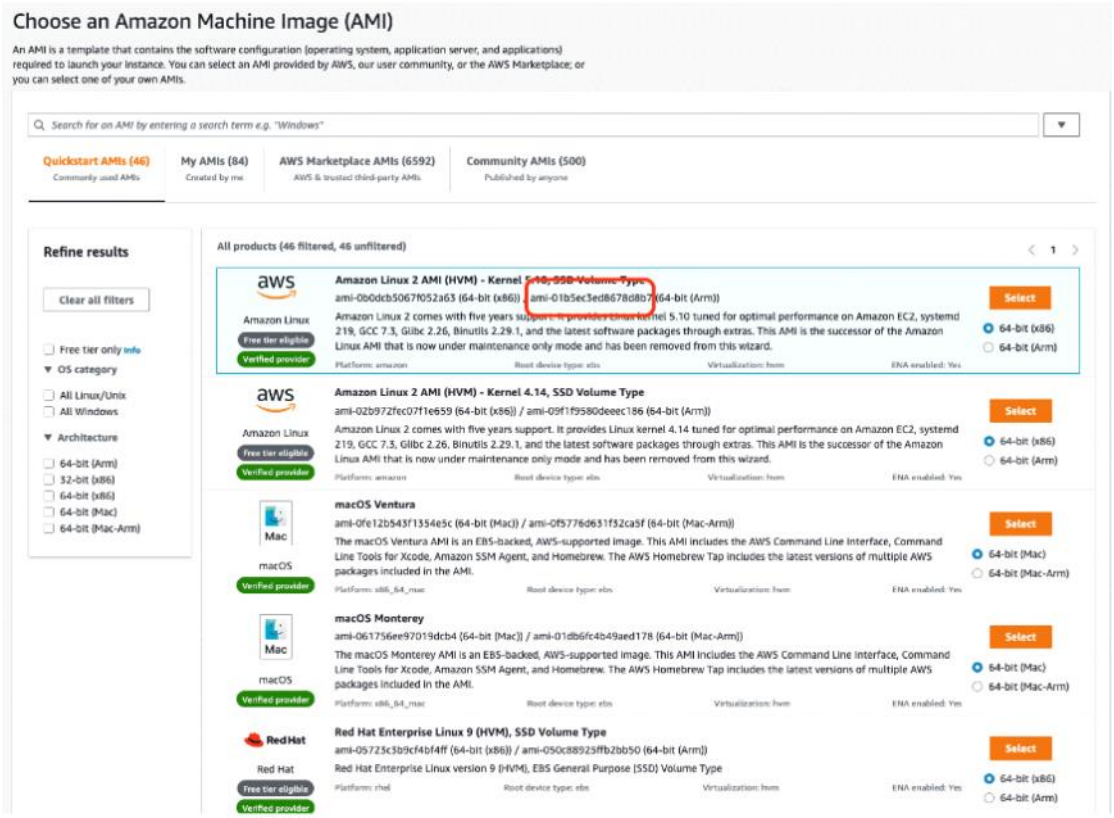
One advantage of using AMIs is that they are reusable. You might choose a Linux-based AMI and configure the HTTP server, application packages, and additional software that you need to run your application. If you want to create another EC2 instance with the same configurations, you could create and configure a new EC2 instance to match the first instance. Or you could create an AMI from your running instance and use the AMI to start a new instance. That way, your new instance would have the same configurations as your current instance because the configurations set in the AMIs are the same.

Finding AMIs

There are multiple ways to select an AMI. To learn more, expand each of the following five categories.

1. Quick Start AMIs are commonly used AMIs created by AWS that you can select to get started quickly.
2. AWS Marketplace AMIs provide popular open-source and commercial software from third-party vendors.
3. My AMIs are created from your EC2 instances.
4. Community AMIs are provided by the AWS user community.
5. Build your own custom image with EC2 Image Builder.

Each AMI in the AWS Management Console has an AMI ID, which is prefixed by ami-, followed by a random hash of numbers and letters. The IDs are unique to each AWS Region.



Configuring EC2

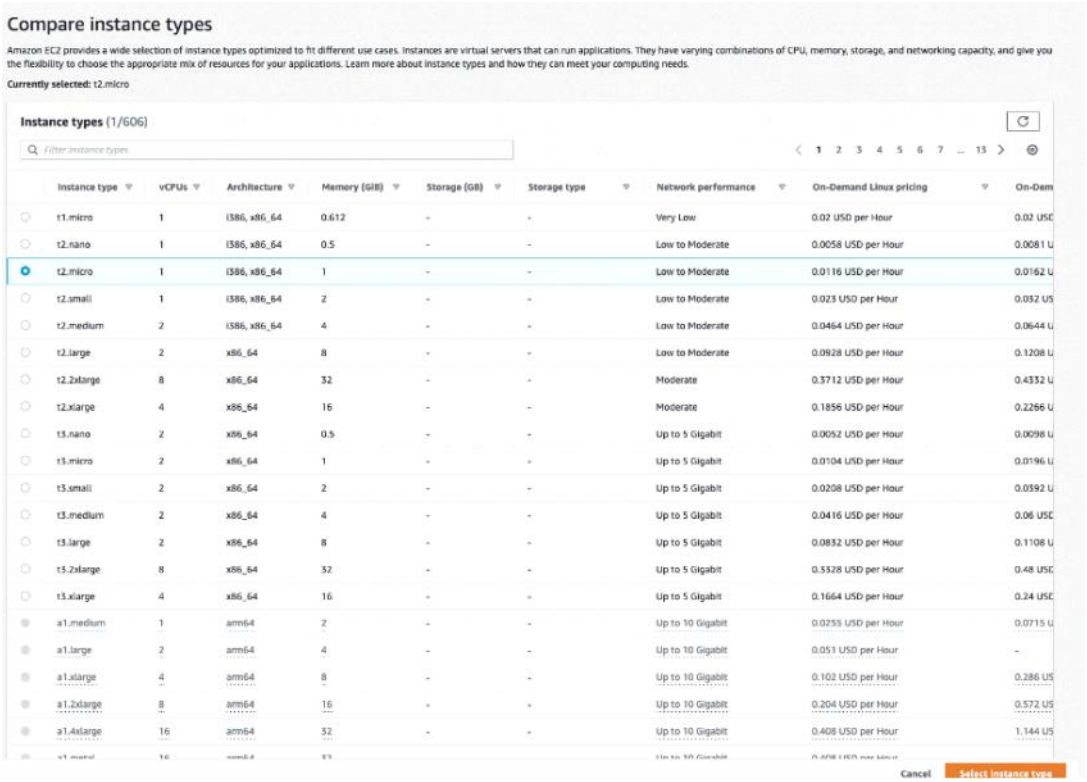
Now that you know how to select an operating system for your EC2 instance, you are ready to choose other configurations to create your EC2 instance, such as the instance type, network, and storage.

For an application like the employee directory application, you need instances with enough capacity to process customer requests. Your instance sizing will depend on both the demands of your application and the anticipated size of your user base.

Forecasting server capacity for an on-premises application requires difficult decisions involving significant upfront capital spending. In contrast, changes to the allocation of your cloud-based services can be made with a simple API call. Because of the AWS pay-as-you-go model, you can match your infrastructure capacity to your application's demand, instead of the other way around.

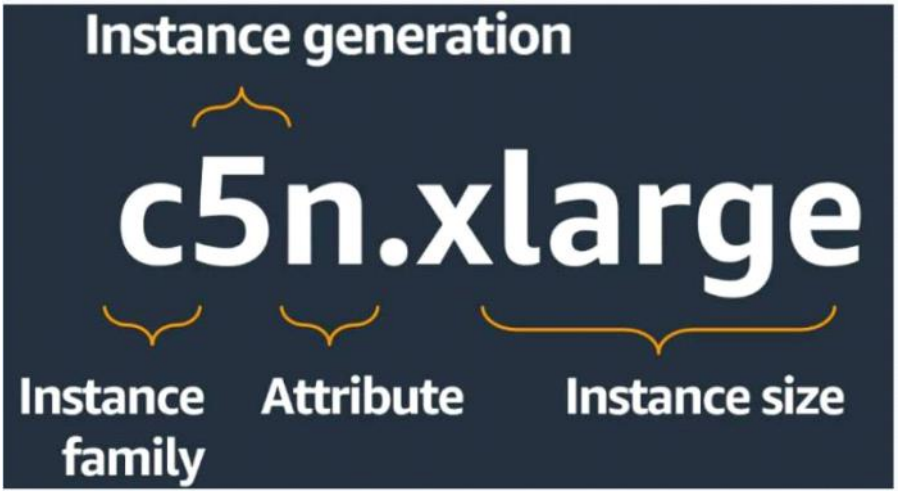
Amazon EC2 instance types

EC2 instances are a combination of virtual processors (vCPUs), memory, network, and, in some cases, instance storage and graphics processing units (GPUs). When you create an EC2 instance, you need to choose how much you need of each of these components.



AWS offers a variety of instances that differ based on performance. Some instances provide more capacity than others. To get an overview of the capacity details for a particular instance, you should look at the instance type. Instance types consist of a prefix identifying the type of workloads they're optimized for, followed by a size. For example, the instance type c5n.xlarge can be broken down as follows:

1. **First position** – The first position, c, indicates the instance family. This indicates that this instance belongs to the compute optimized family.
2. **Second position** – The second position, 5, indicates the generation of the instance. This instance belongs to the fifth generation of instances.
3. **Remaining letters before the period** – In this case, n indicates additional attributes, such as local NVMe storage.
4. **After the period** – After the period, xlarge indicates the instance size. In this example, it's xlarge.



Instance families

Each instance family is optimized to fit different use cases. The following table describes instance families and some typical workloads.

To learn more about EC2 instance families, expand the category below.

Instance Family	Description	Use case
General purpose	General purpose instances provide a balance of compute, memory, and networking resources, and can be used for a variety of workloads.	Ideal for applications that use these resources in equal proportions, such as web servers and code repositories
Compute optimized	Compute optimized instances are ideal for compute-bound applications that benefit from high-performance processors.	Well-suited for batch processing workloads, media transcoding, high performance web servers, high performance computing (HPC), scientific modeling, dedicated gaming servers and ad server engines, machine learning inference, and other compute intensive applications
Memory optimized	Memory optimized instances are designed to deliver fast performance for workloads that process large datasets in memory.	Memory-intensive applications, such as high-performance databases, distributed web-scale in-memory caches, mid-size in-memory databases, real-time big-data analytics, and other enterprise applications
Accelerated optimized	Accelerated computing instances use hardware accelerators or co-processors to perform functions such as floating-point number calculations, graphics processing, or data pattern matching more efficiently than is possible in software running on CPUs.	Machine learning, HPC, computational fluid dynamics, computational finance, seismic analysis, speech recognition, autonomous vehicles, and drug discovery
Storage optimized	Storage optimized instances are designed for workloads that require high sequential read and write access to large datasets on local storage. They are optimized to deliver tens of thousands of low-latency random I/O operations per second (IOPS) to applications that replicate their data across different instances.	NoSQL databases (Cassandra, MongoDB and Redis), in-memory databases, scale-out transactional databases, data warehousing, Elasticsearch, and analytics
HPC optimized	High performance computing (HPC) instances are purpose built to offer the best price performance for running HPC workloads at scale on AWS.	Ideal for applications that benefit from high-performance processors, such as large, complex simulations and deep learning workloads

EC2 instance locations

Unless otherwise specified, when you launch EC2 instances, they are placed in a default virtual private cloud (VPC). The default VPC is suitable for getting started quickly and launching public EC2 instances without having to create and configure your own VPC.

Any resource that you put inside the default VPC will be public and accessible by the internet, so you shouldn't place any customer data or private information in it.

When you get more comfortable with networking on AWS, you should change this default setting to choose your own custom VPCs and restrict access with additional routing and connectivity mechanisms.

Architecting for high availability

In the network, your instance resides in an Availability Zone of your choice. As you learned previously, AWS services that are scoped at the Availability Zone level must be architected with high availability in mind.

Although EC2 instances are typically reliable, two are better than one, and three are better than two. Specifying the instance size gives you an advantage when designing your architecture because you can use more smaller instances rather than a few larger ones.

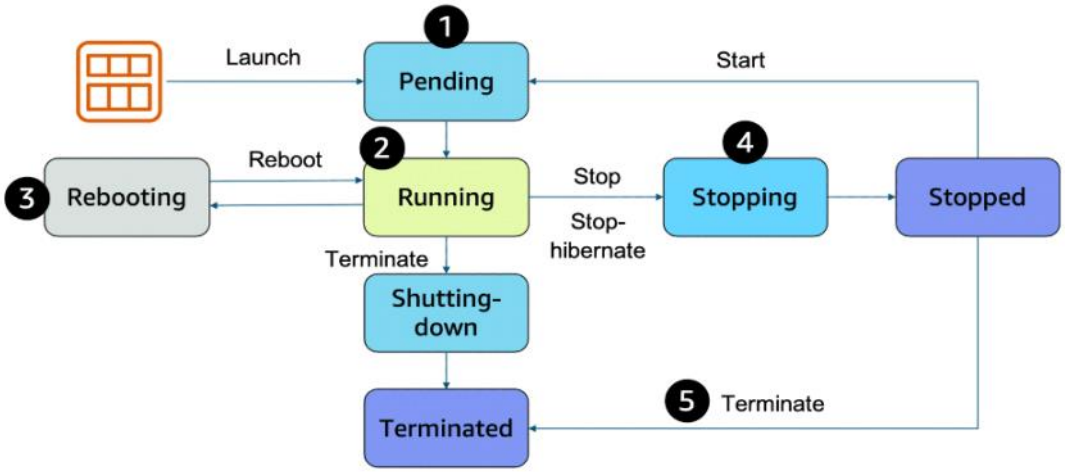
If your frontend only has a single instance and the instance fails, your application goes down. Alternatively, if your workload is distributed across 10 instances and one fails, you lose only 10 percent of your fleet, and your application availability is hardly affected.

When architecting any application for high availability, consider using at least two EC2 instances in two separate Availability Zones.

Module-2 Amazon EC2 Instance Lifecycle

18 May 2024 13:07

An EC2 instance transitions between different states from the moment you create it until its termination.



1. When you launch an instance, it enters the pending state. When an instance is pending, billing has not started. At this stage, the instance is preparing to enter the running state. Pending is where AWS performs all actions needed to set up an instance, such as copying the AMI content to the root device and allocating the necessary networking components.

2. When your instance is running, it's ready to use. This is also the stage where billing begins. As soon as an instance is running, you can take other actions on the instance, such as reboot, terminate, stop, and stop-hibernate.

3. When you reboot an instance, it’s different than performing a stop action and then a start action. Rebooting an instance is equivalent to rebooting an operating system. The instance keeps its public DNS name (IPv4) and private and public IPv4 addresses. An IPv6 address (if applicable) remains on the same host computer and maintains its public and private IP address, in addition to any data on its instance store volumes.

4. When you stop your instance, it enters the stopping and then stopped state. This is similar to when you shut down your laptop. You can stop and start an instance if it has an Amazon Elastic Block Store (Amazon EBS) volume as its root device. When you stop and start an instance, your instance can be placed on a new underlying physical server. Your instance retains its private IPv4 addresses and if your instance has an IPv6 address, it retains its IPv6 address. When you put the instance into stop-hibernate, the instance enters the stopped state, but saves the last information or content into memory, so that the start process is faster.

5. When you terminate an instance, the instance stores are erased, and you lose both the public IP address and private IP address of the machine. Termination of an instance means that you can no longer access the machine. As soon as the status of an instance changes to shutting down or terminated, you stop incurring charges for that instance.

Difference between stop and stop-hibernate

When you stop an instance, it enters the stopping state until it reaches the stopped state. AWS does not charge usage or data transfer fees for your instance after you stop it. But storage for any Amazon EBS volumes is still charged. While your instance is in the stopped state, you can modify some attributes, like the instance type. When you stop your instance, the data from the instance memory (RAM) is lost.

When you stop-hibernate an instance, Amazon EC2 signals the operating system to perform hibernation (suspend-to-disk), which saves the contents from the instance memory (RAM) to the EBS root volume. You can hibernate an instance only if hibernation is turned on and the instance meets the hibernation prerequisites.

Pricing

One of the ways to reduce costs with Amazon EC2 is to choose the right pricing option for the way that your applications run. AWS offers a variety of pricing options to address different workload scenarios.

- **On-Demand Instances**

With On-Demand Instances, you pay for compute capacity per hour or per second, depending on which instances that you run. There are no long-term commitments or upfront payments required. Billing begins whenever the instance is running, and billing stops when the instance is in a stopped or terminated state. You can increase or decrease your compute capacity to meet the demands of your application and only pay the specified hourly rates for the instance that you use.

On-Demand Instances are recommended for the following use cases:

1. Users who prefer the low cost and flexibility of Amazon EC2 without upfront payment or long-term commitments.
2. Applications with short-term, spiky, or unpredictable workloads that cannot be interrupted.
3. Applications being developed or tested on Amazon EC2 for the first time.

- **Spot Instances**

For applications that have flexible start and end times, Amazon EC2 offers the Spot Instances option. With Amazon EC2 Spot Instances, you can request spare Amazon EC2 computing capacity for up to 90 percent off the On-Demand price. Spot Instances are recommended for the following use cases:

1. Applications that have flexible start and end times
2. Applications that are only feasible at very low compute prices
3. Users with fault-tolerant or stateless workloads

With Spot Instances, you set a limit on how much you want to pay for the instance hour. This is compared against the current Spot price that AWS determines. Spot Instance prices adjust gradually based on long-term trends in supply and demand for Spot Instance capacity. If the amount that you pay is more than the current Spot price and there is capacity, you will receive an instance.

- **Savings Plans**

Savings Plans are a flexible pricing model that offers low usage prices for a 1-year or 3-year term commitment to a consistent amount of usage. Savings Plans apply to Amazon EC2, AWS Lambda, and AWS Fargate usage and provide up to 72 percent savings on AWS compute usage.

For workloads that have predictable and consistent usage, Savings Plans can provide significant savings compared to On-Demand Instances. Savings Plans are recommended for the following use cases:

1. Workloads with a consistent and steady-state usage .
2. Customers who want to use different instance types and compute solutions across different locations .
3. Customers who can make monetary commitment to use Amazon EC2 over a 1-year or 3-year term.

- **Reserved Instances**

For applications with steady state usage that might require reserved capacity, Amazon EC2 offers the Reserved Instances option. With this option, you save up to 75 percent compared to On-Demand Instance pricing. You can choose between three payment options: All Upfront, Partial Upfront, or No Upfront. You can select either a 1-year or 3-year term for each of these options.

With Reserved Instances, you can choose the type that best fits your applications needs.

1. **Standard Reserved Instances**: These provide the most significant discount (up to 72 percent off On-Demand pricing) and are best suited for steady-state usage.

2. **Convertible Reserved Instances:** These provide a discount (up to 54 percent off On-Demand pricing) and the capability to change the attributes of the Reserved Instance if the exchange results in the creation of Reserved Instances of equal or greater value. Like Standard Reserved Instances, Convertible Reserved Instances are best suited for steady-state usage.
3. **Scheduled Reserved Instances:** These are available to launch within the time windows that you reserve. With this option, you can match your capacity reservation to a predictable recurring schedule that only requires a fraction of a day, a week, or a month.

- **Dedicated Hosts**

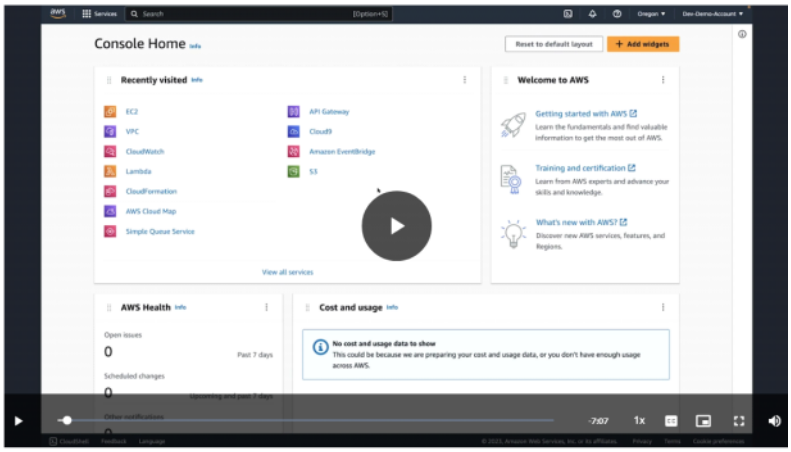
A Dedicated Host is a physical Amazon EC2 server that is dedicated for your use. Dedicated Hosts can help you reduce costs because you can use your existing server-bound software licenses, such as Windows Server, SQL Server, and Oracle licenses. And they can also help you meet compliance requirements. Amazon EC2 Dedicated Host is also integrated with AWS License Manager, a service that helps you manage your software licenses, including Microsoft Windows Server and Microsoft SQL Server licenses.

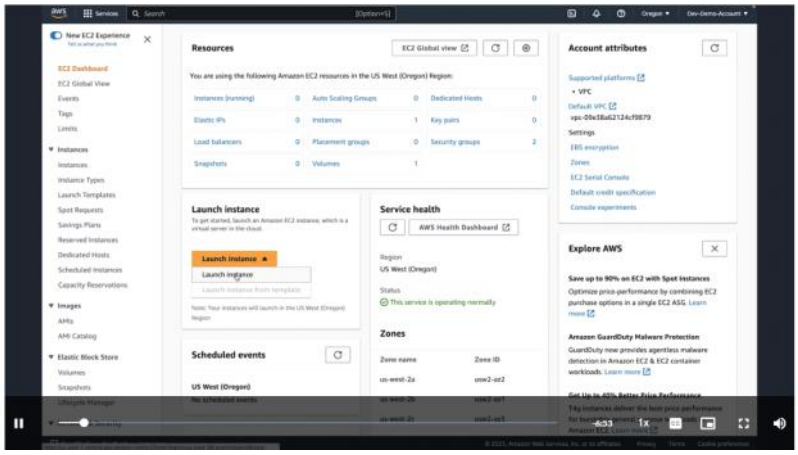
- Dedicated Hosts can be purchased on demand (hourly).
- Dedicated Hosts can be purchased as a Reservation for up to 70 percent off the On-Demand price.

Module-2 Demonstration: Launching the Employee Directory Application on Amazon EC2

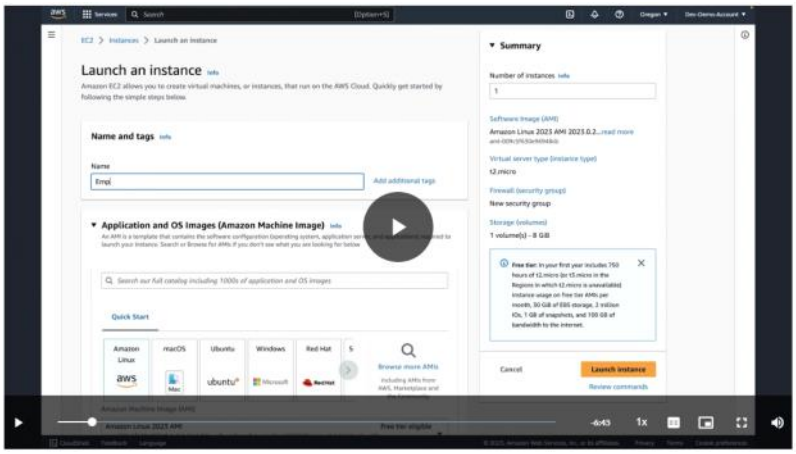
20 May 2024 09:45

we are going to launch the employee directory application on an Amazon EC2 instance using the default VPC. So, you can see I'm already in the AWS console and I'm going to navigate to the EC2 dashboard. From here, I want to launch a new instance. So I'm going to select launch instance and then select launch instance again. And now this brings us to a page where we can provide the configurations for our EC2 instance. I'm gonna give this a name, employee directory app and then we can scroll down and we can view what AMI we want to choose.

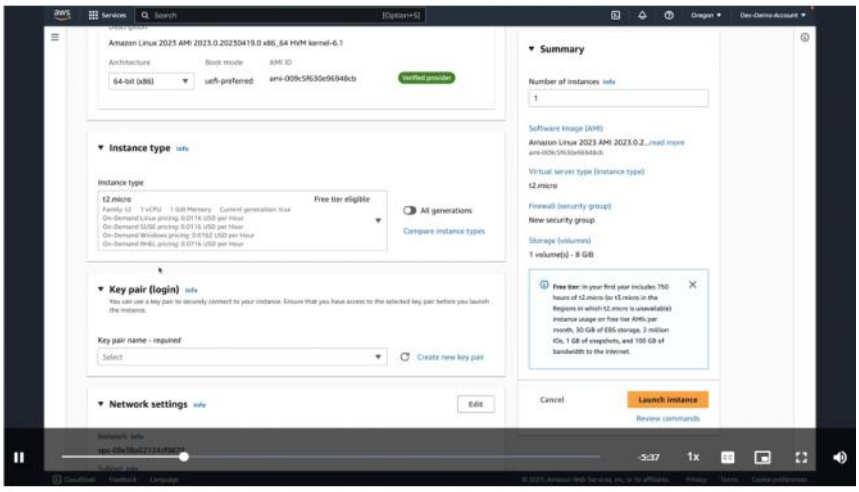


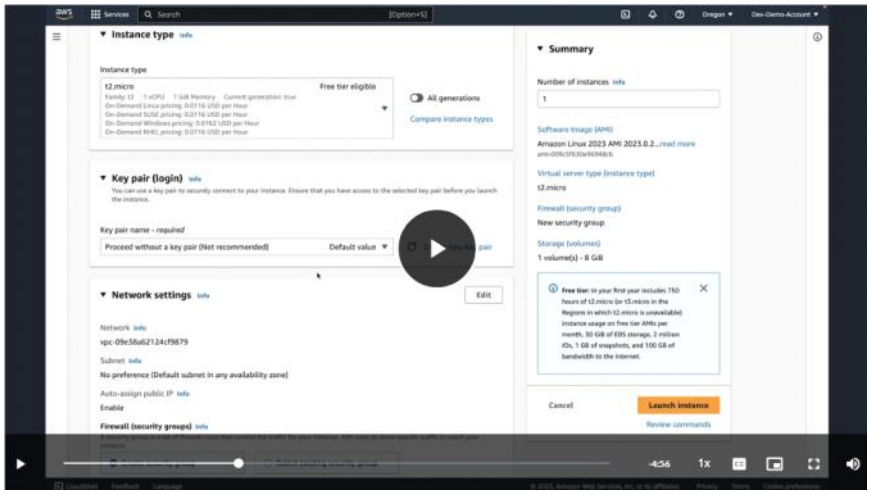


The AMI is the Amazon machine image, and this is a template that contains the software configuration for your instance on boot like the operating system, any sort of application server or any applications that you wanna have pre-installed when you launch your instance. We're gonna be using the Amazon Linux 2023 AMI for this, though you could browse the AWS marketplace to get access to the different types of AMIs that vendors offer. So, if you're using some sort of third party software that you want to launch on an EC2 instance, they might have an AMI out there that you could use to make your life a little bit simpler. You also can create and manage your own AMIs and share them internally within your organization to say you can only launch an EC2 instance using one of our approved AMIs that might have some specific security or compliant sort of packages pre-installed. That's another option.



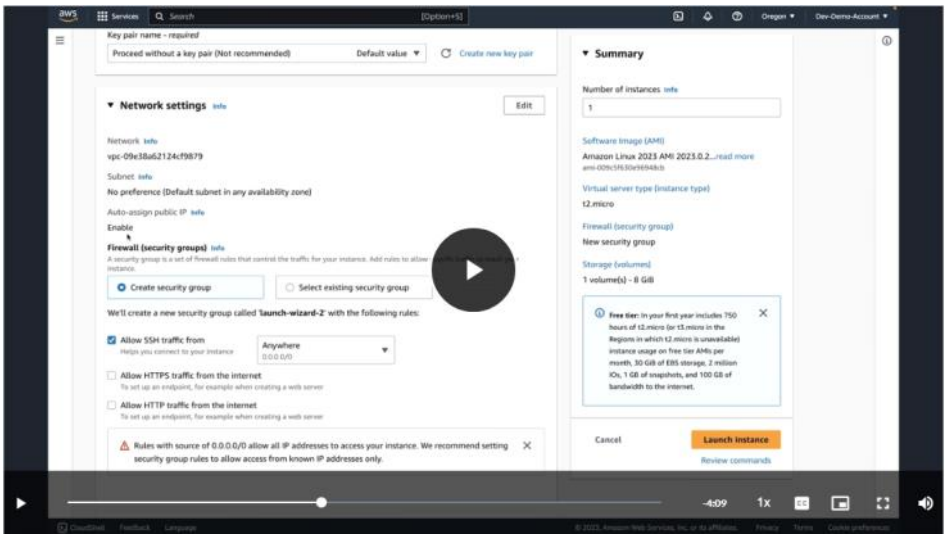
So leaving the Amazon Linux 2023 AMI selected I'm going to scroll down and then we can select our instance type. The instance type will determine how much CPU, how much memory, what types of hardware is available to you on this instance. Some instances come with a GPU, for example. In this case, we are gonna use the free-tier eligible EC2 instance type, T2 micro which is the default that's selected. So we'll leave that the way it is. Then we can scroll down and select whether we want to use a key pair. The key pair is going to allow you to have access to this instance doing something like SSH. I'm going to select proceed without a key pair because if I need to connect to this instance, I'm gonna use the connect button in the AWS console where I don't need to have a key pair to connect to this instance to view things like logs.



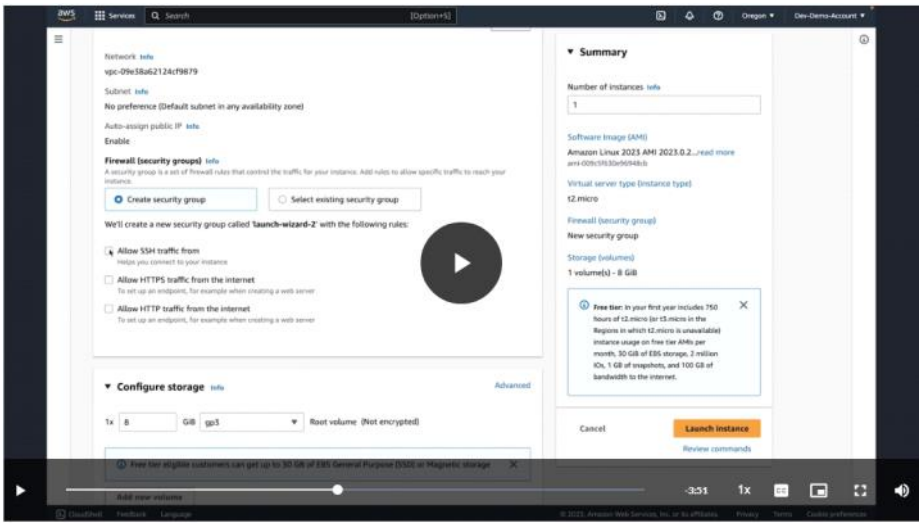


Then scrolling down some more we then can configure our network settings. This is using a default VPC and a default subnet. This default VPC is going to have subnets that have internet access, so

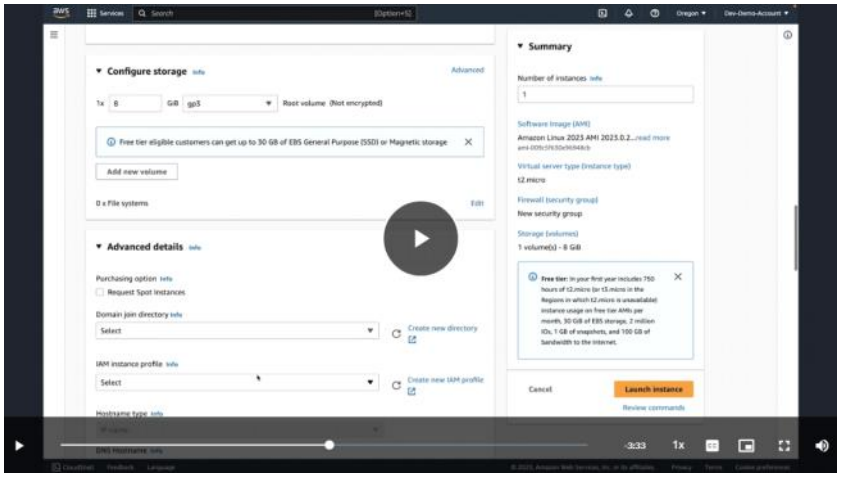
meaning that there is an internet gateway attached to this VPC that allows internet traffic to flow into this default VPC. You'll learn more about VPCs coming up, but just understand that this default VPC does allow public internet access which is good for our use case, but might not be great for other use cases where you would more than likely put your instance behind something like a load balancer and then you would use private subnets. But we wanna be able to access this instance directly. So with that in mind, we're going to use the default VPC and we're also going to leave this auto-assigned public IP to enable and this will allow our EC2 incidents to have a publicly reachable IP address once it's been launched.



Then scrolling down a little bit more we have to configure our firewall. This is going to be our security group. Security groups are instance level firewalls. We want to disable traffic from SSH because we're not going to need that. Then we're going to allow HTTPS and allow HTTP. In our application, we're gonna be using HTTP directly.

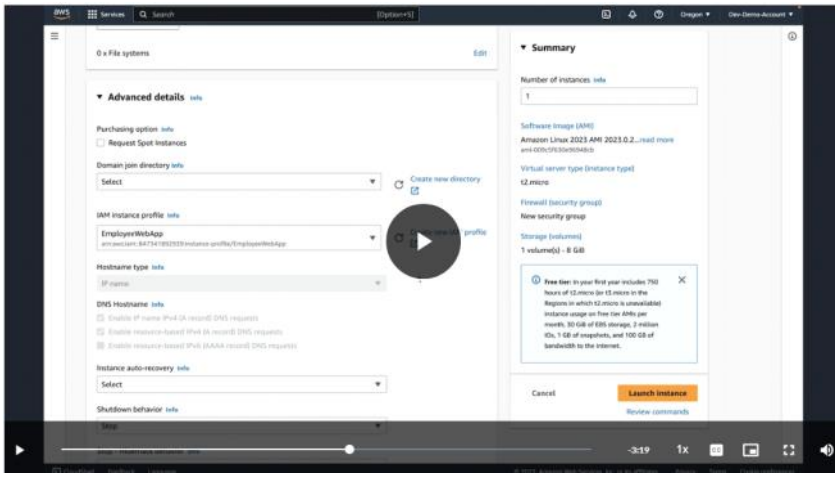


Then scrolling down, we can configure some storage. We're gonna leave our root volume here, but we're not going to add any more EBS volumes.



Then we can expand the advanced details section and then we need to select an IAM instance profile.

In this case, I'll be selecting the employee web app role and this role has permissions attached to it to allow whoever is using the role to make calls to S3 or to DynamoDB.

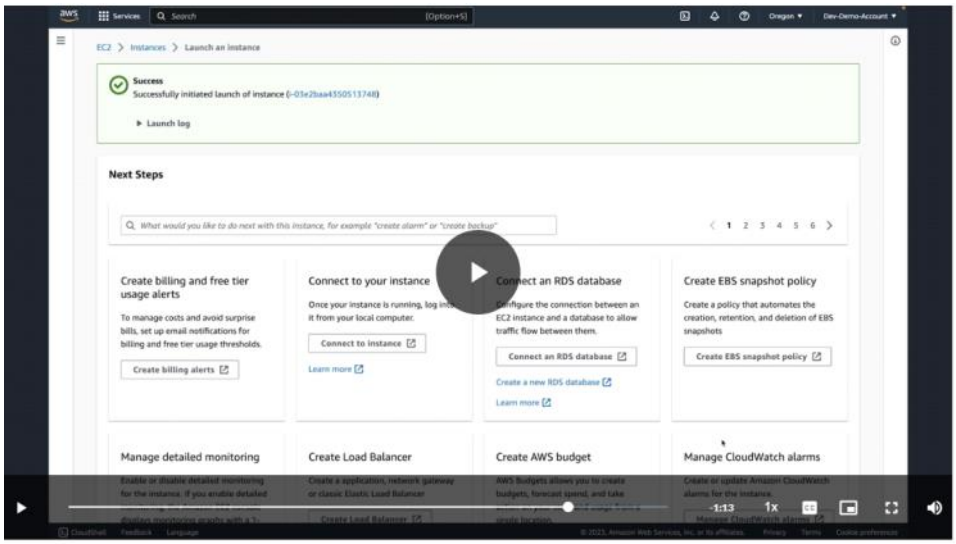


We know that our application has some code that's running that's going to be making API calls to S3 and DynamoDB. We need to allow our AWS SDK to actually get access to these credentials, and the way that we do that is through this IAM instance profile. So we're associating this role with this instance profile that will allow our code to get access to these temporary credentials.

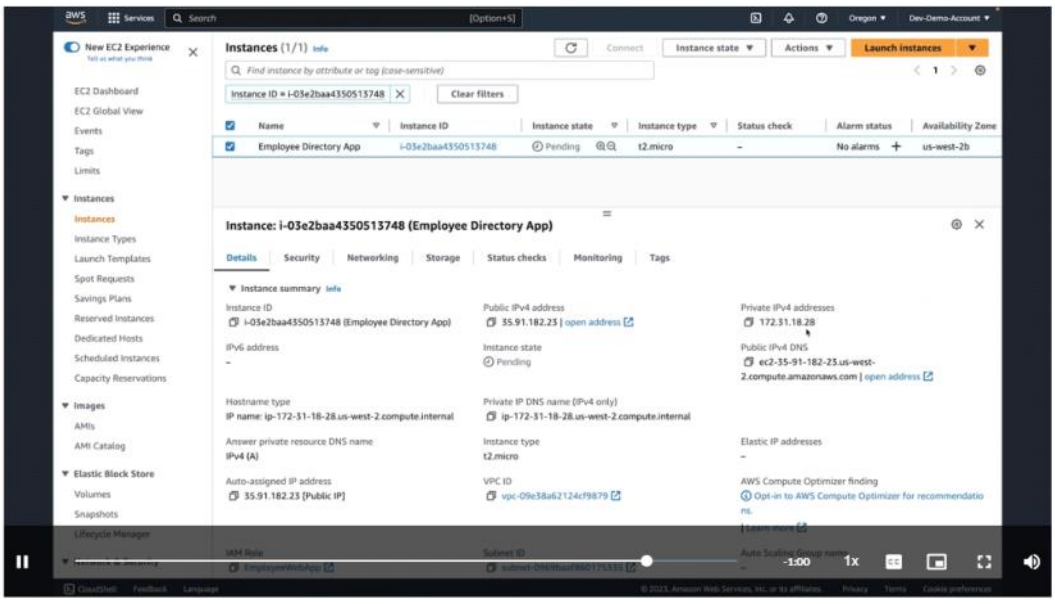
Then scrolling down some more we will come to the user data section, and this is where you can provide a bash script that will run on launch. So, this script here is what's gonna be used to download and install our application. Our application was written in Python using Flask as the web server, and this is going to be for our employee directory application. So on the first line, we are just declaring that this is a Bash script. Then we are using wget to download a zip file from S3. So we hosted the zip file in an S3 bucket for this to be downloaded. Then we're unzipping that, and then we're changing directories into that new directory. Then we're using Yum to install Python three and PIP. Then we're using PIP to install any dependencies for our Flask application, which will all be listed in a file called requirements.txt, which got downloaded from this zip file. Then we're installing a package called stress. This stress package is going to be used to simulate a CPU spike on this instance, which is going to allow us to simulate autoscaling whenever we use Amazon EC2 autoscaling in a future lesson. Then we have three different environment variables. One for our photos bucket, we're gonna be using S3 to store our photos. We don't have that value yet, so we'll just leave this place holder there. Then we have the default region we're operating out of Oregon, and then we have our Dynamo mode on and this means that we want our application to use DynamoDB and then we're running this application and hosting it on port 80. Now we can select launch instance, and this will begin to go build out the different things that we need like our security

group and we can see that our instance has launched.

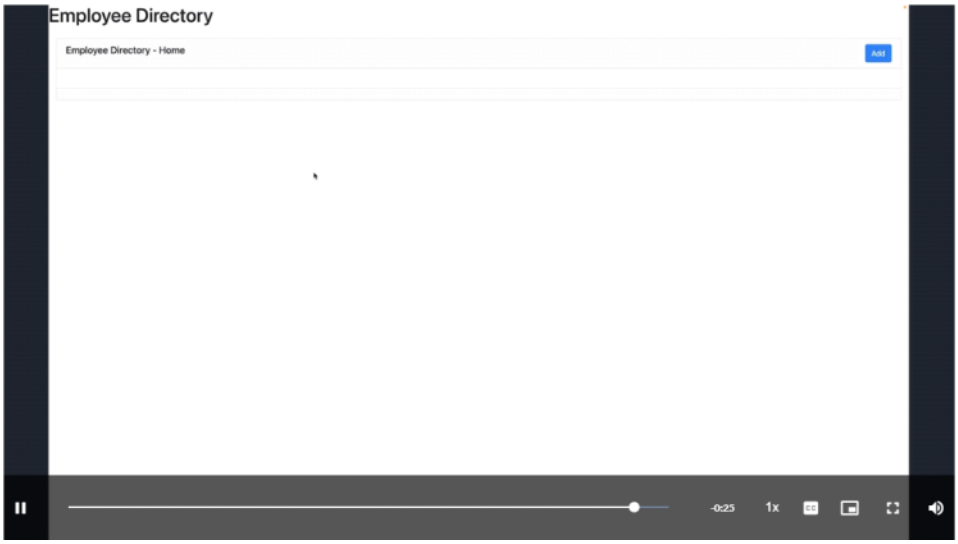
```
#!/bin/bash -ex
wget https://aws-tc-largeobjects.s3-us-west-2.amazonaws.com/DEV-AWS-MO-GCNv2/FlaskApp.zip
unzip FlaskApp.zip
cd FlaskApp/
yum -y install python3 mysql
pip3 install -r requirements.txt
amazon-linux-extras install epel
yum -y install stress
export PHOTOS_BUCKET=${SUB_PHOTOS_BUCKET}
export AWS_DEFAULT_REGION=<INSERT REGION HERE>
export DYNAMO_MODE=on
FLASK_APP=application.py /usr/local/bin/flask run --host=0.0.0.0 --port=80
```



Now if I select this instance ID, that brings us to a page where we can then select our instance and view some information about it. We can see things like our public IP address our private IP address, and our public DNS name. But we can't access this directly yet. We can see that our instant state is running. But if I refresh this, you can see that we have some status checks that are still initializing. I wanna wait until both of these status checks have passed. So we'll give this a few minutes and then we'll come back once it's ready. Okay, we're back and we can see that our instant state is running and this status checks have passed. Now, if I select this instance again we can then open up our IP address in a new tab.



All right, and we can see that our employee directory application has been opened in a new tab and we have an empty employee directory. This is exactly what we would expect at this point as we don't have a place to store our photos or the employee information with our S3 bucket or DynamoDB table.



Module-2 Container Services

20 May 2024 11:22

No one-size-fits-all compute service exists because it depends on your needs. The key is to understand what each option offers.

AWS offers a broad spectrum of compute offerings that give you the flexibility to choose the right tool for the job. As mentioned earlier, the three main categories of compute are virtual machines (VMs), containers, and serverless. No one-size-fits-all compute service exists because it depends on your needs.

The key is to understand what each option offers. Then you can build an appropriate cloud architecture for your use case. In this section, you will learn about containers and how to run them.

Containers can host a variety of different workloads, including web applications, lift and shift migrations, distributed applications, and streamlining of development, test, and production environments.

Containers

Although containers are often referred to as a new technology, the idea started in the 1970s with certain UNIX kernels (the central core of the operating system) having the ability to separate their processes through isolation. At the time, this was configured manually, making operations complex.

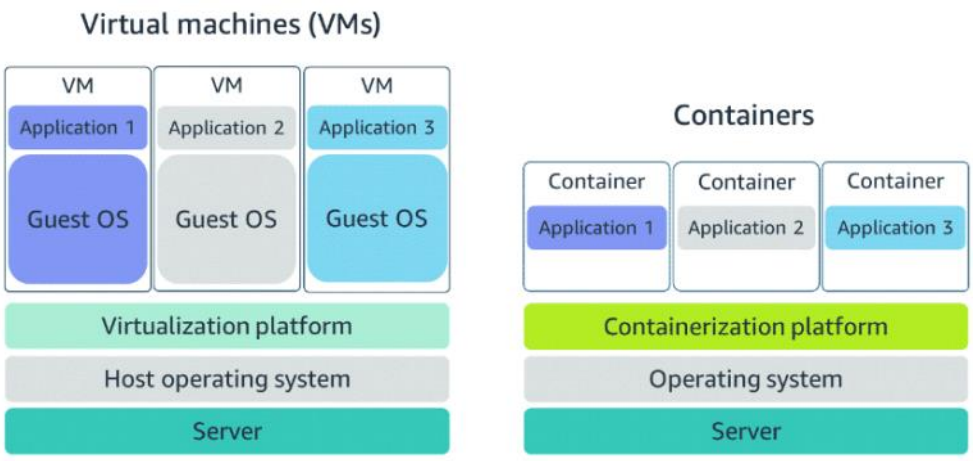
With the evolution of the open-source software community, containers evolved. Today, containers are used as a solution to problems of traditional compute, including the issue of getting software to run reliably when it moves from one compute environment to another.

Containers

A container is a standardized unit that packages your code and its dependencies. This package is designed to run reliably on any platform, because the container creates its own independent environment. With containers, workloads can be carried from one place to another, such as from development to production or from on-premises environments to the cloud.

An example of a containerization platform is Docker. Docker is a popular container runtime that simplifies the management of the entire operating system stack required for container isolation, including networking and storage. Docker helps customers create, package, deploy, and run containers.

Difference between VMs and containers



Compared to virtual machines (VMs), containers share the same operating system and kernel as the host that they are deployed on.

Containers share the same operating system and kernel as the host that they exist on. But virtual machines contain their own operating system. Each virtual machine must maintain a copy of an operating system, which results in a degree of wasted resources.

A container is more lightweight. Containers spin up quicker, almost instantly. This difference in startup time becomes instrumental when designing applications that must scale quickly during I/O bursts.

Containers can provide speed, but virtual machines offer the full strength of an operating system and more resources, like package installation, dedicated kernel, and more.

Orchestrating containers

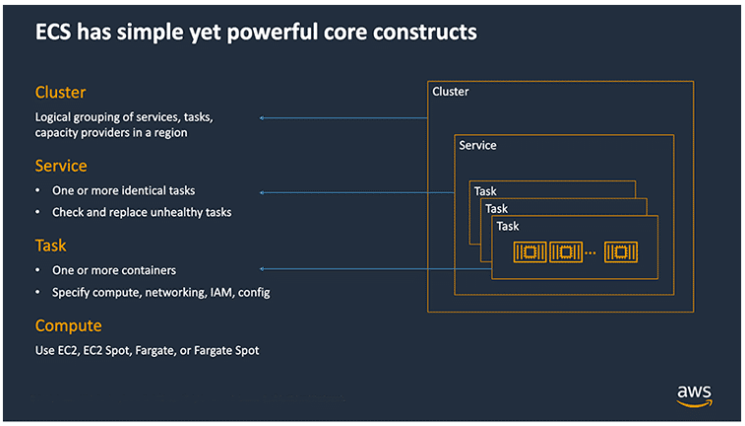
In AWS, containers can run on EC2 instances. For example, you might have a large instance and run a few containers on that instance. Although running one instance is uncomplicated to manage, it lacks high availability and scalability. Most companies and organizations run many containers on many EC2 instances across several Availability Zones.

If you're trying to manage your compute at a large scale, you should consider the following:

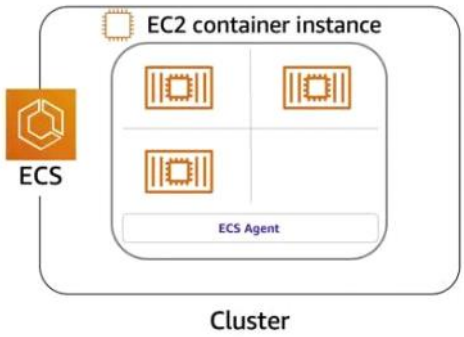
1. How to place your containers on your instances
2. What happens if your container fails
3. What happens if your instance fails
4. How to monitor deployments of your containers

This coordination is handled by a container orchestration service. AWS offers two container orchestration services:
1. Amazon Elastic Container Service (Amazon ECS)
2. Amazon Elastic Kubernetes Service (Amazon EKS).

Managing containers with Amazon ECS



If you choose to have more control by running and managing your containers on a cluster of Amazon EC2 instances, you will also need to install the Amazon ECS container agent on your EC2 instances. Note that an EC2 instance with the container agent installed is often called a container instance. This container agent is open source and responsible for communicating to the Amazon ECS service about cluster management details. You can run the agent on both Linux and Windows AMIs.



When the Amazon ECS container instances are up and running, you can perform actions that include, but are not limited to, the following:

1. Launching and stopping containers
2. Getting cluster state
3. Scaling in and out
4. Scheduling the placement of containers across your cluster
5. Assigning permissions
6. Meeting availability requirements
7. To prepare your application to run on Amazon ECS, you create a task definition. The task definition is a text file, in JSON format, that describes one or more containers. A task definition is similar to a blueprint that describes the resources that you need to run a container, such as CPU, memory, ports, images, storage, and networking information.

Here is a simple task definition that you can use for your corporate directory application. In this example, this runs on the Nginx web server.

```
{
"family": "webserver",
"containerDefinitions": [ {
"name": "web",
"image": "nginx",
"memory": "100",
"cpu": "99"
} ],
"requiresCompatibilities": [ "FARGATE" ],
"networkMode": "awsvpc",
"memory": "512",
"cpu": "256"
}
```

Using Kubernetes with Amazon EKS

Kubernetes is a portable, extensible, open-source platform for managing containerized workloads and services. By bringing software development and operations together by design, Kubernetes created a rapidly growing ecosystem that is very popular and well established in the market.

If you already use Kubernetes, you can use Amazon EKS to orchestrate the workloads in the AWS Cloud. Amazon EKS is a managed service that you can use to run Kubernetes on AWS without needing to install, operate, and maintain your own Kubernetes control plane or nodes. Amazon EKS is conceptually similar to Amazon ECS, but with the following differences:

1. In Amazon ECS, the machine that runs the containers is an EC2 instance that has an ECS agent installed and configured to run and manage your containers. This instance is called a container instance. In Amazon EKS, the machine that runs the containers is called a worker node or Kubernetes node.
2. An ECS container is called a task. An EKS container is called a pod.
3. Amazon ECS runs on AWS native technology. Amazon EKS runs on Kubernetes.

If you have containers running on Kubernetes and want an advanced orchestration solution that can provide simplicity, high availability, and fine-grained control over your infrastructure, Amazon EKS could be the tool for you.

Module-2 Introduction to Serverless

20 May 2024 12:24

Spend time on the things that differentiate your application, rather than spending time on ensuring availability, scaling, and managing servers.

Removing the undifferentiated heavy lifting

If you run your code on Amazon EC2, AWS is responsible for the physical hardware. You are still responsible for the logical controls, such as the guest operating system, security and patching, networking, security, and scaling.

As covered in the previous Container Services lesson, you choose to have more control by running and managing your containers on Amazon ECS and Amazon EKS. By doing so, AWS is still responsible for more of the container management, such as deploying containers across EC2 instances and managing the container cluster. However, when running Amazon ECS and Amazon EKS on Amazon EC2, you are still responsible for maintaining the underlying EC2 instances.

Is there a way to remove some of this undifferentiated heavy lifting? Yes! If you want to deploy your workloads and applications without having to manage any EC2 instances, you can do that on AWS with serverless compute.

Go serverless

With serverless compute, you can spend time on the things that differentiate your application, rather than spending time on ensuring availability, scaling, and managing servers. Every definition of serverless mentions the following four aspects:

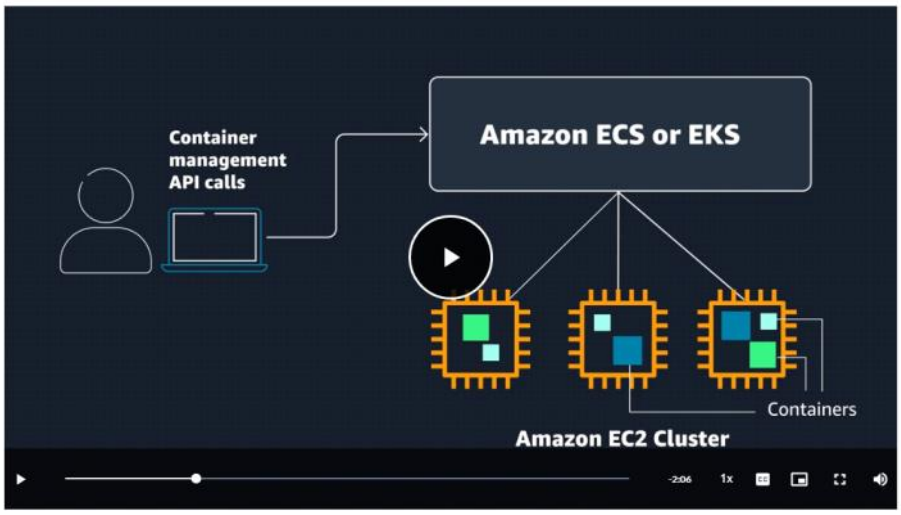
1. There are no servers to provision or manage.
2. It scales with usage.
3. You never pay for idle resources.
4. Availability and fault tolerance are built in.
5. AWS has developed serverless services for all three layers of the application stack. We will cover two services, AWS Fargate and AWS Lambda, in the following lessons.

Module-2 Serverless with AWS Fargate

20 May 2024 12:31

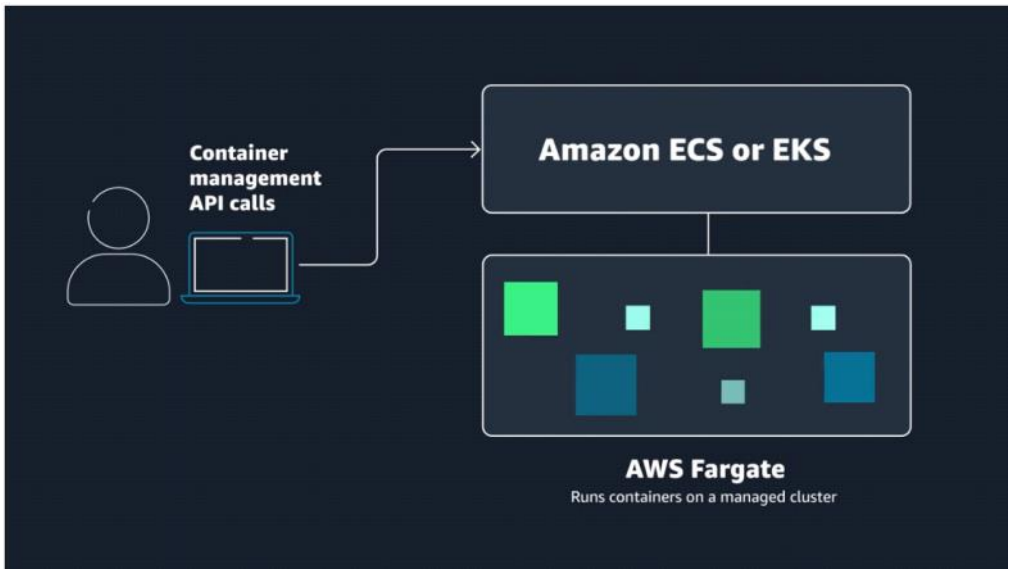
AWS Fargate scales and manages the infrastructure, so developers can work on what they do best, application development.

Let's continue our conversation about containers. To recap, ECS or EKS is the container orchestrator. This is the tool that is managing the container's lifecycle. Then you need the compute platform. This is where the containers actually run. Previously, you learned that ECS and EKS run containers on clusters of EC2 instances. And in that case, you were using EC2 as the compute platform for your containers, and you also have tight control over those instances.



Now, EC2 is certainly not serverless.

So thinking about serverless compute for containers, I want to introduce to you a service named AWS Fargate. AWS Fargate is a serverless compute platform for containers that you can use with either ECS or EKS. With AWS Fargate as the compute platform, you run your containers on a managed serverless platform. The scaling and fault tolerance is built in, and you don't need to worry about the underlying operating system or environment.



Instead, you define your application content, networking, storage, and scaling requirements. There is no provisioning, patching, cluster capacity management, or infrastructure management required.

1. To break that down a bit, the way you work with Fargate is you first build your container images and push them into a repository like Amazon Elastic Container Registry, for example. Which is a place where you can store container images for them to be pooled and deployed from. So you have your container images in your repo.
2. And then you define memory and compute resources for your task if you're using ECS or your pod if you're using EKS. Then you run your containers.

3. And at the end, you only pay for the amount of vCPU, memory, and storage resources consumed by your containerized applications.

Fargate does support Spot and Compute Savings Plan pricing options just like with Amazon EC2 instances. So there is some flexibility on how you plan to run containers on Fargate. So you can still get a good amount of control for your container's deployment, but without needing to worry about the provisioning, patching, and managing the underlying operating systems or instances. And no need to scale the infrastructure in and out to meet demand like you do with EC2. As far as use cases go, AWS Fargate can be used for all of the common container use cases, including microservice architecture applications, batch processing, machine learning applications, and migrating on-premises applications to the cloud.

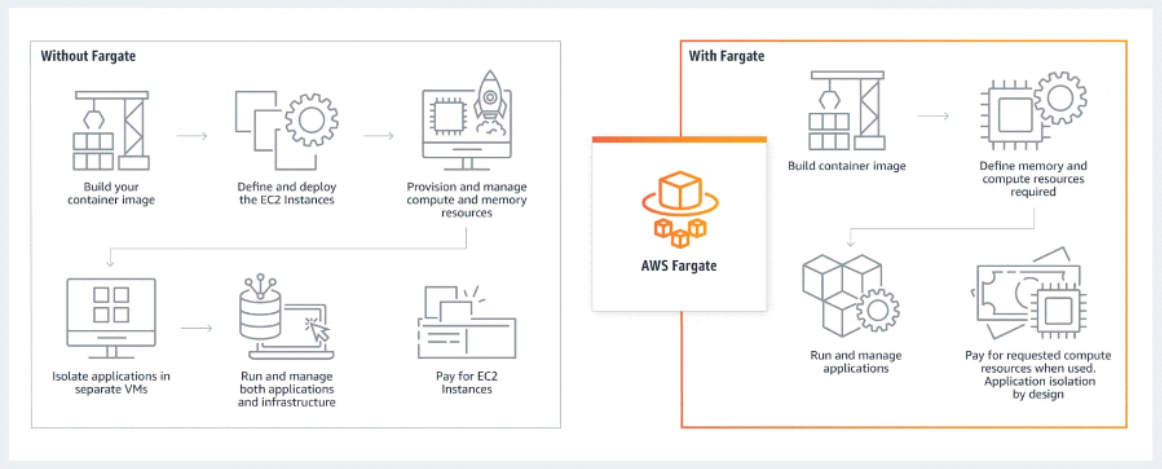
Exploring serverless containers with AWS Fargate

Fargate abstracts the EC2 instance so that you're not required to manage the underlying compute infrastructure. However, with Fargate, you can use all the same Amazon ECS concepts, APIs, and AWS integrations. It natively integrates with IAM and Amazon Virtual Private Cloud (Amazon VPC). With native integration with Amazon VPC, you can launch Fargate containers inside your network and control connectivity to your applications.

AWS Fargate is a purpose-built serverless compute engine for containers. AWS Fargate scales and manages the infrastructure, so developers can work on what they do best, application development. It achieves this by allocating the right amount of compute. This eliminates the need to choose and manage EC2 instances, cluster capacity, and scaling. Fargate supports both Amazon ECS and Amazon EKS architecture and provides workload isolation and improved security by design.

AWS Fargate

Serverless compute for containers



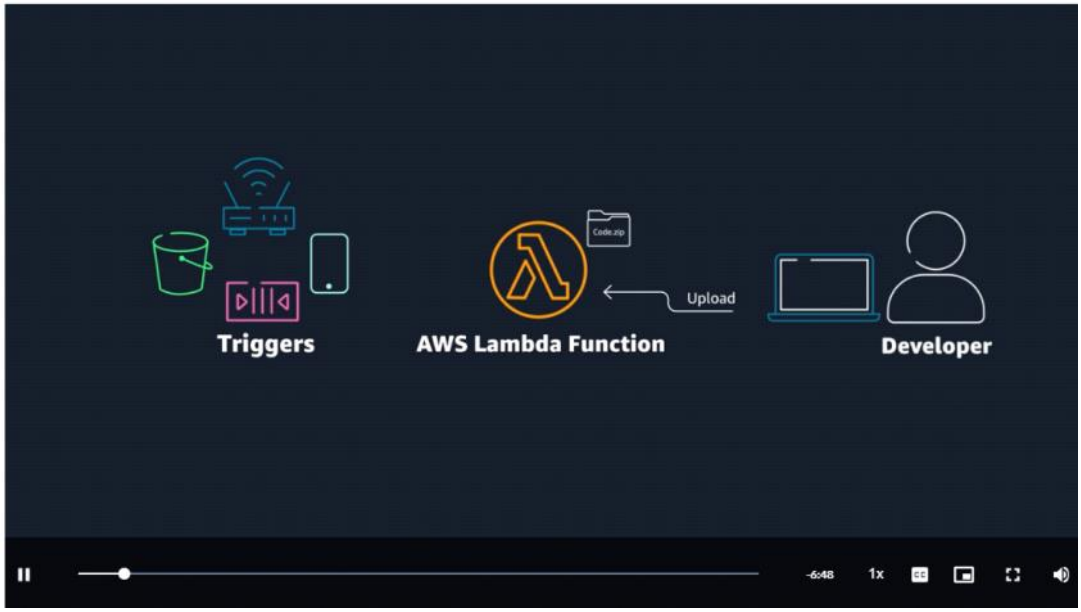
aws

Module-2 Serverless with AWS Lambda

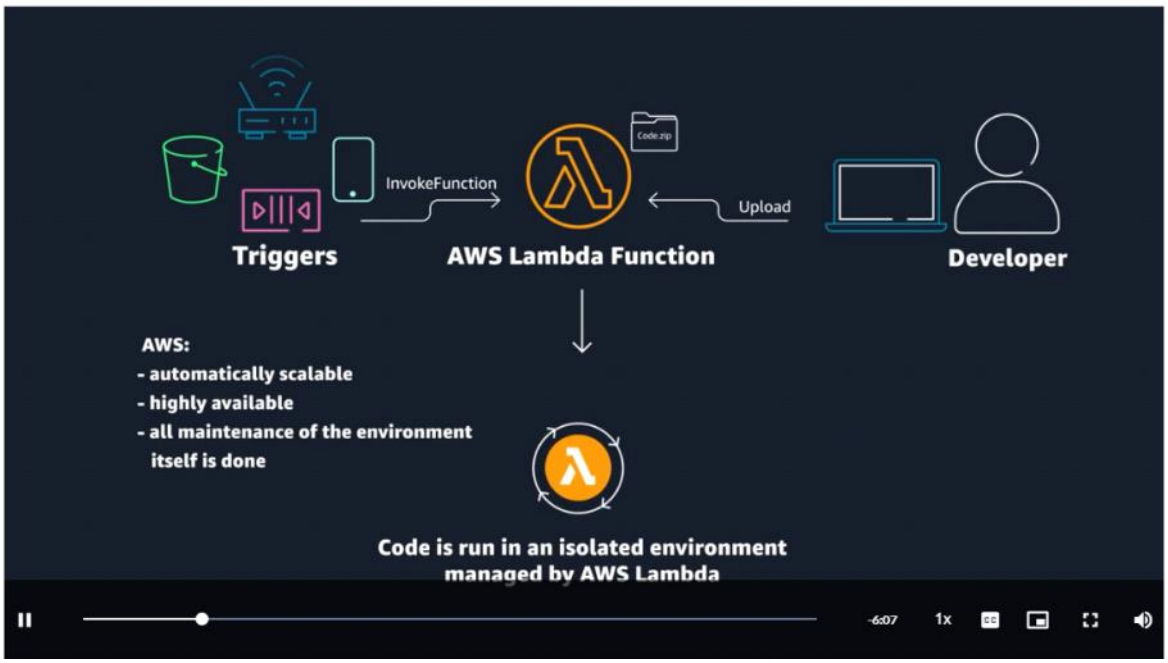
20 May 2024 12:47

With AWS Lambda, there are no servers to manage. You get continuous scaling with subsecond metering and consistent performance.

AWS Lambda is one of the serverless compute options you have available to you on AWS. Lambda allows you to package and upload your code to the Lambda service, creating what is called a Lambda function. Once you create a Lambda function, it isn't always running all of the time. Instead, Lambda functions run in response to triggers. You can configure a trigger for when you want your function to run. And from there, the Lambda service waits for the trigger or polls for an event to trigger the function, depending on what the trigger is that you choose.

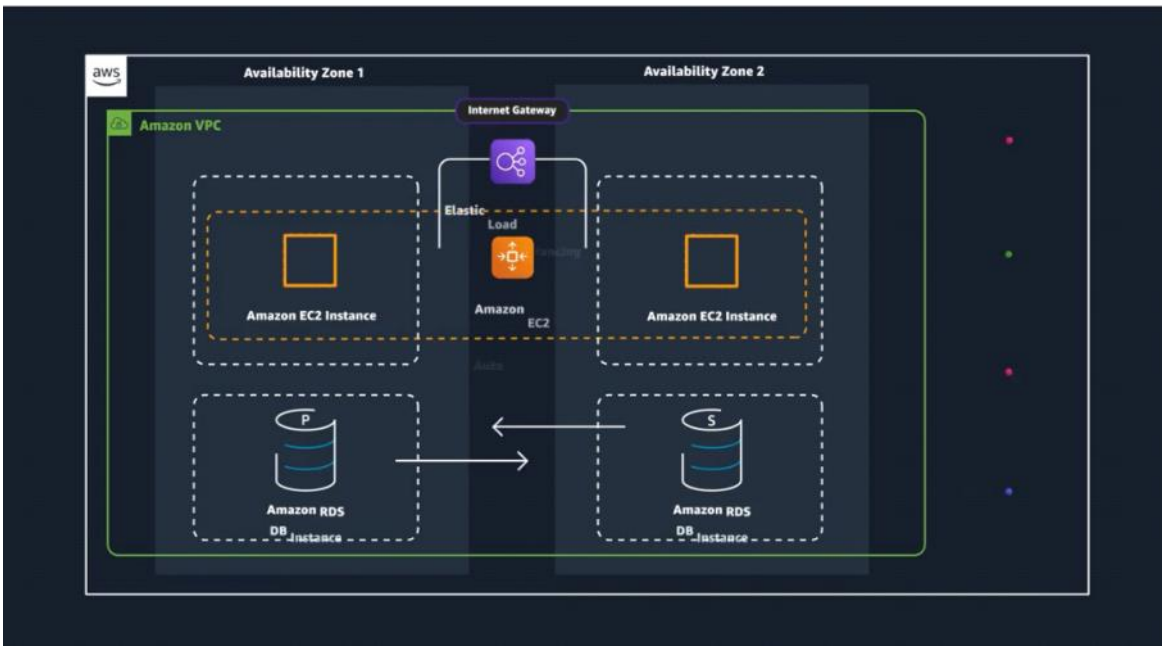


There's a long list of available triggers for AWS Lambda, and new ones are being supported all of the time, so I won't run through them all now. However, a couple of common examples of triggers for Lambda functions are an HTTP request, an upload of a file to the storage service Amazon S3, events originating from other AWS services, or even in-app activity from mobile devices. When the trigger is detected, the code is automatically run in a managed environment, an environment that you do not need to worry too much about because it is automatically scalable, highly available, and all of the maintenance of the environment itself is done by AWS. You do, however, get to choose the language your Lambda function is coded in, the amount of memory and CPU your function is allocated in the environment, permissions, dependencies, and many other aspects of how the function runs.

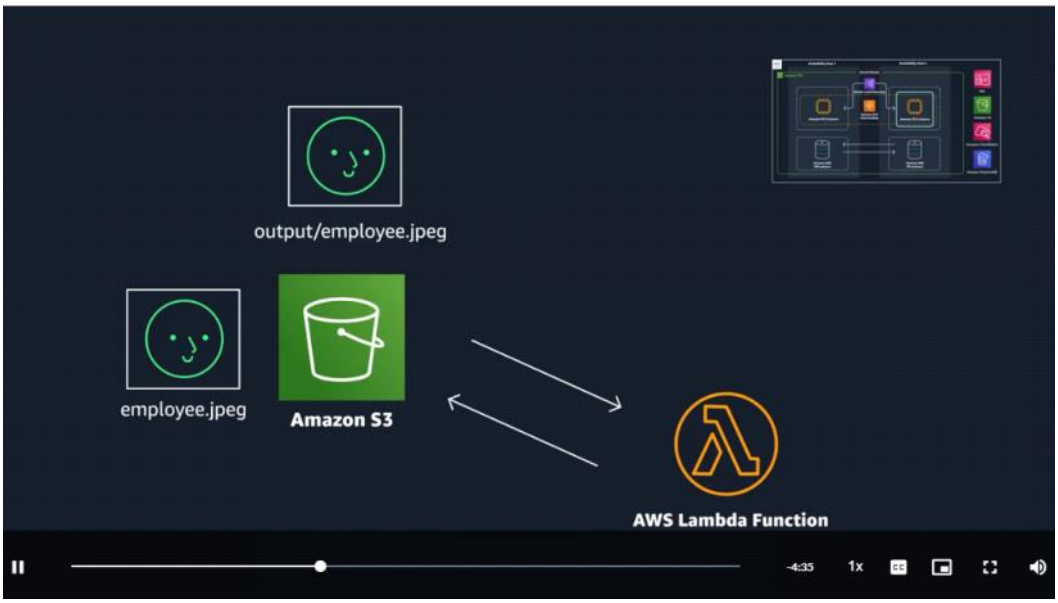


If you have 1 or 1,000 incoming triggers, AWS Lambda will scale your function to meet demand, each in its own isolated environment. Lambda is currently designed to run code that has a runtime of under 15 minutes. So this isn't for long-running processes like deep learning or batch jobs. You wouldn't host something like a WordPress site on AWS Lambda. It's more suited for quick processing like a web backend handling request, or a backend report processing service, or microservice hosting. One of the best things about Lambda is that you aren't billed for code that isn't running. You only get billed for the resources that you use, rounded up to the nearest 1-millisecond interval.

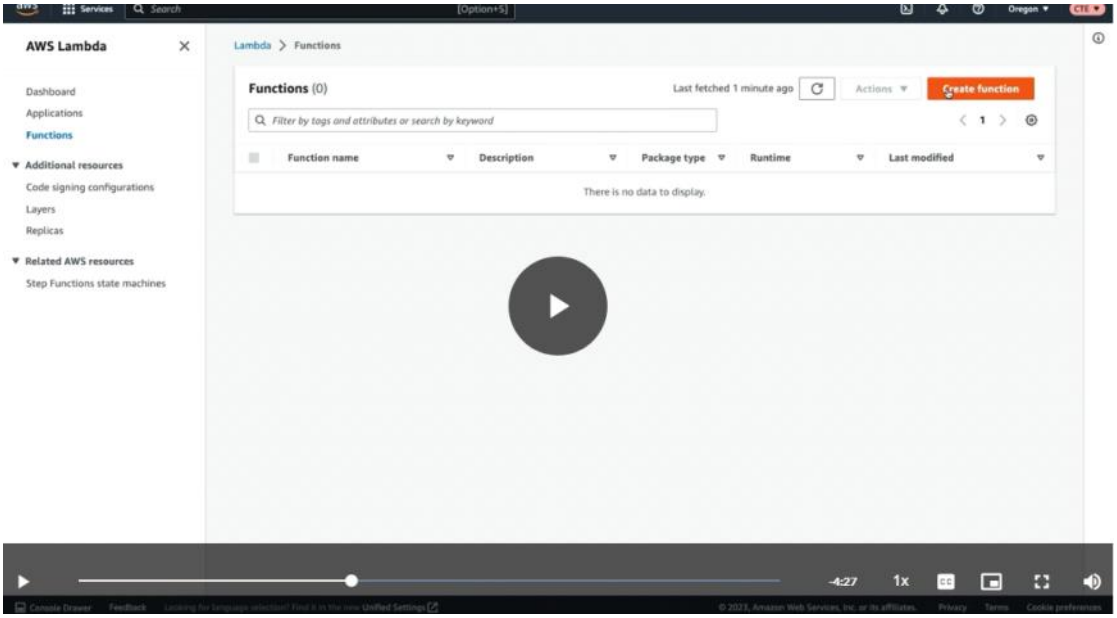
To understand Lambda more thoroughly, let's run through a quick demo. In this demo, I will create a Lambda function that resizes images uploaded into the employee directory to be uniform thumbnail size. It makes sense to create this sort of logic with a Lambda function. You don't need an application to be running 24/7 on EC2 to resize photos uploaded to the application. You really only need to run the resize logic when a new photo is uploaded.
So here's our diagram for the app.



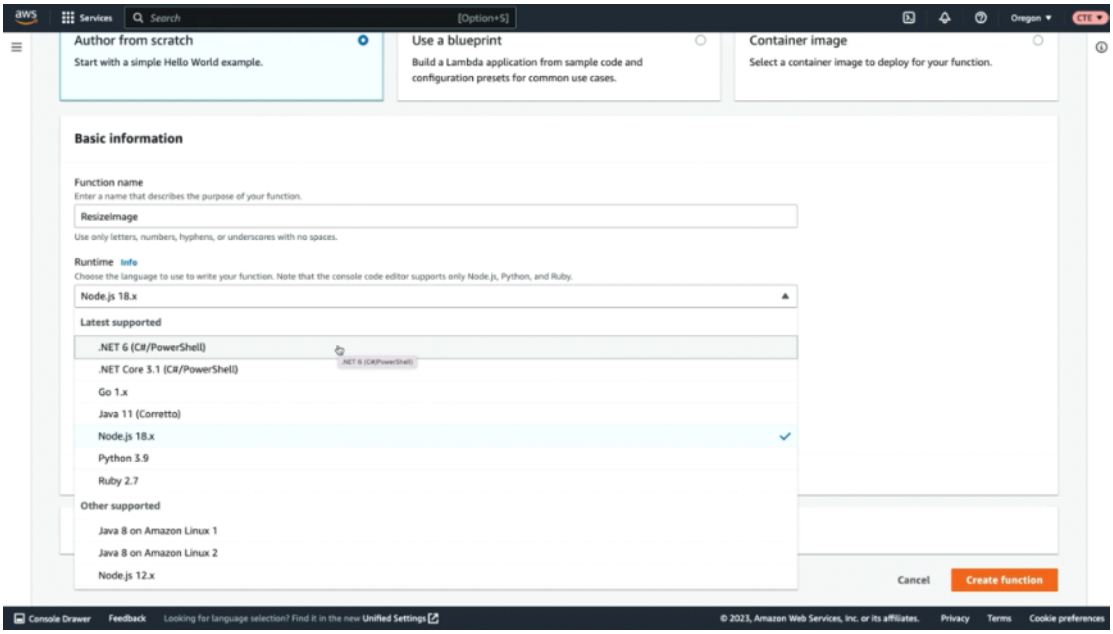
What I want to do here is when a new photo is uploaded to S3, it triggers a Lambda function that then resizes the image and uploads it into that same S3 bucket, but to a different location than where the original image is stored. All right, let's go ahead and build this out.



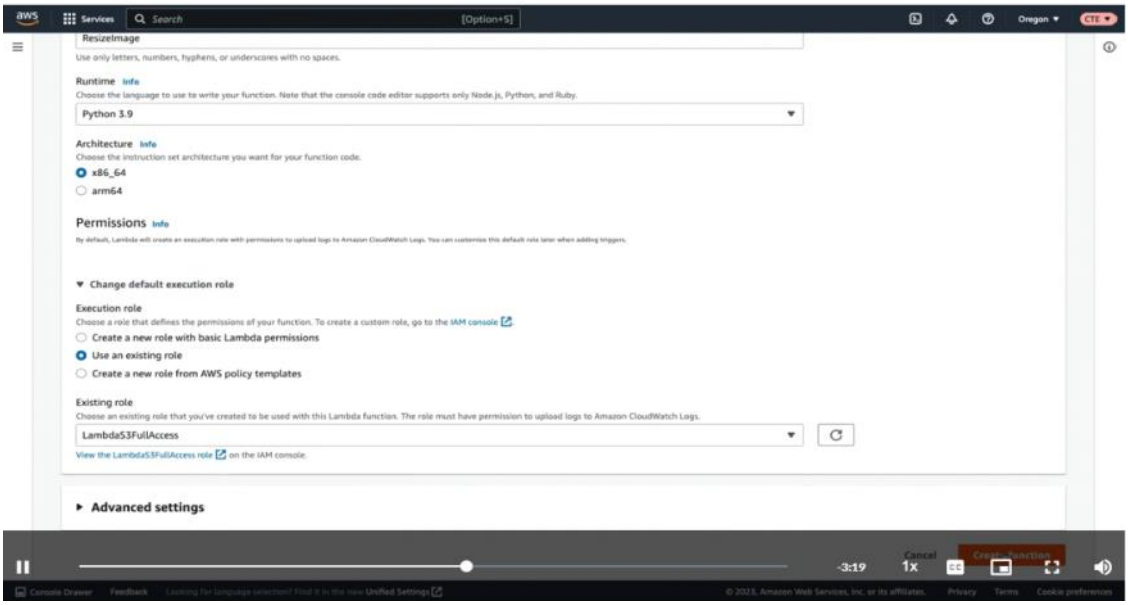
You can see that I'm already in the AWS Lambda console, and I'm going to first click Create function.



Next, we get to select what type of function we want to build. Do we want to author a function from scratch? Do we want to use a blueprint which will give you some sample code and configuration presets? Or you can use a container image with AWS Lambda, as Lambda does support running containers. We are going to author a function from scratch. For the function name, we will call this ResizeImage, and then we get to select the runtime. The runtime is the language that your Lambda function is written in. Because with Lambda, it runs in the Lambda environment, but it's still your code. So I'm going to go ahead and select Python 3.9 for this, as we wrote this function in Python.

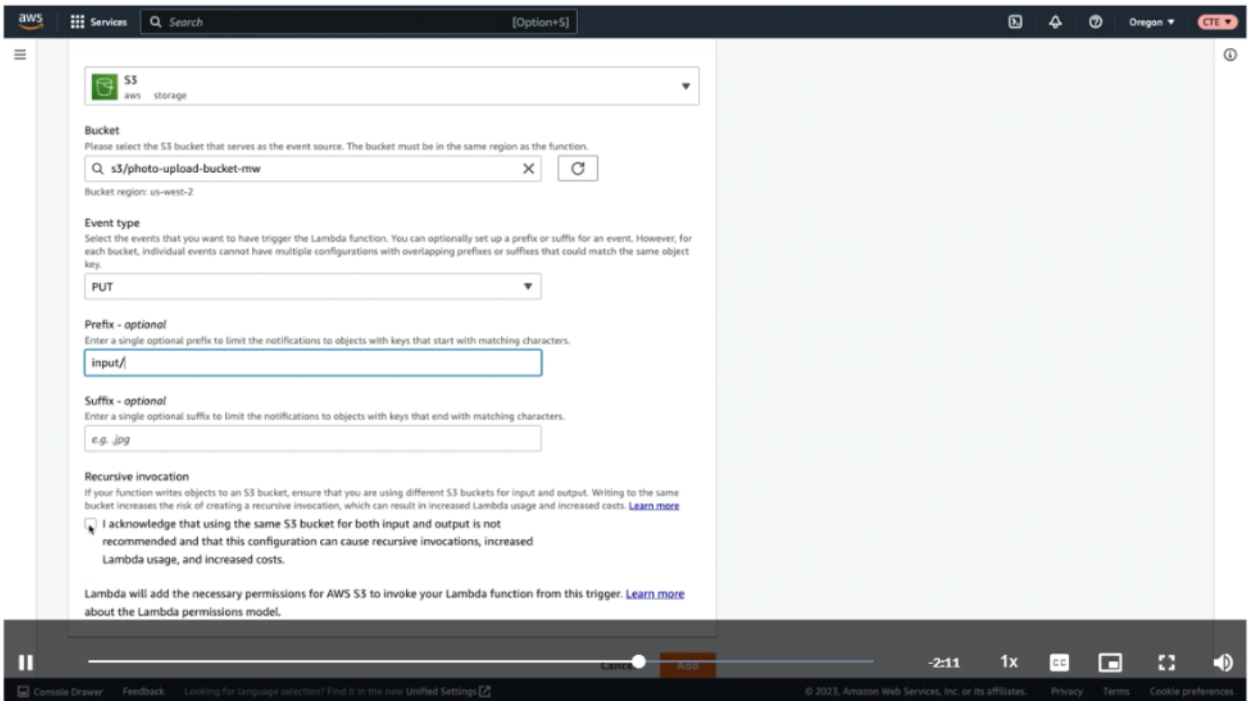


And then we have to select the permissions. And the permissions are going to be delegated to the function through IAM roles. So I'm going to click Use an existing role. And then if I expand this dropdown, we can select LambdaS3FullAccess, which is an IAM role that I've already created in this account that will give this function permissions to read and write from S3. Now we can scroll down and select Create function.

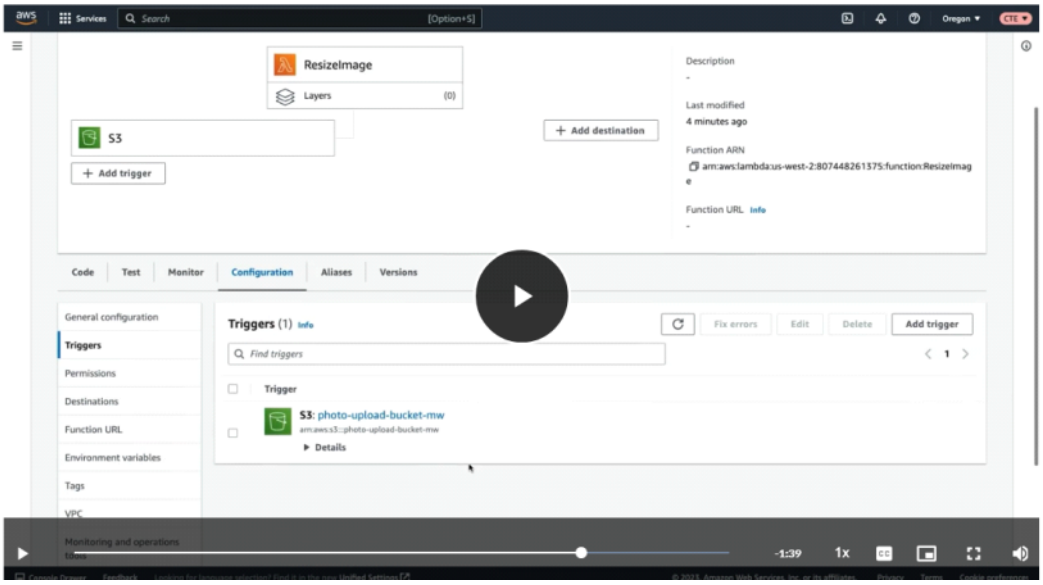


Now we can add a trigger for this Lambda function, and I will select Add trigger. And here we get to configure the trigger source. And if we expand this dropdown, you can see a list of AWS services that can act as triggers, like API Gateway, an Application Load Balancer, DynamoDB. We're going to scroll down and select S3 for our trigger. And then we need to select which S3 bucket will be the event source. I already have an S3 bucket created in this account called photo-upload-bucket-mw. We will select that one. And then if we scroll down, I'm going to select the event type. I want to select only a PUT, so I want PUT operations to trigger this AWS Lambda function. Then we can provide a prefix. With this prefix, what we're essentially doing is we're going to say: I only want to trigger this Lambda function if a PUT occurs in a specific location inside of this bucket. We're going to upload photos to an input prefix and that will then trigger the Lambda function. We then need to acknowledge that if we're using the same S3 bucket for both input and output, that it could cause recursive invocations. So the way that we're getting around that is by supplying this input prefix. The event will be triggered when an image is uploaded to this prefix, but then the output will actually be uploaded to a different location. We will have an output prefix where the image will be uploaded to. So we're going to

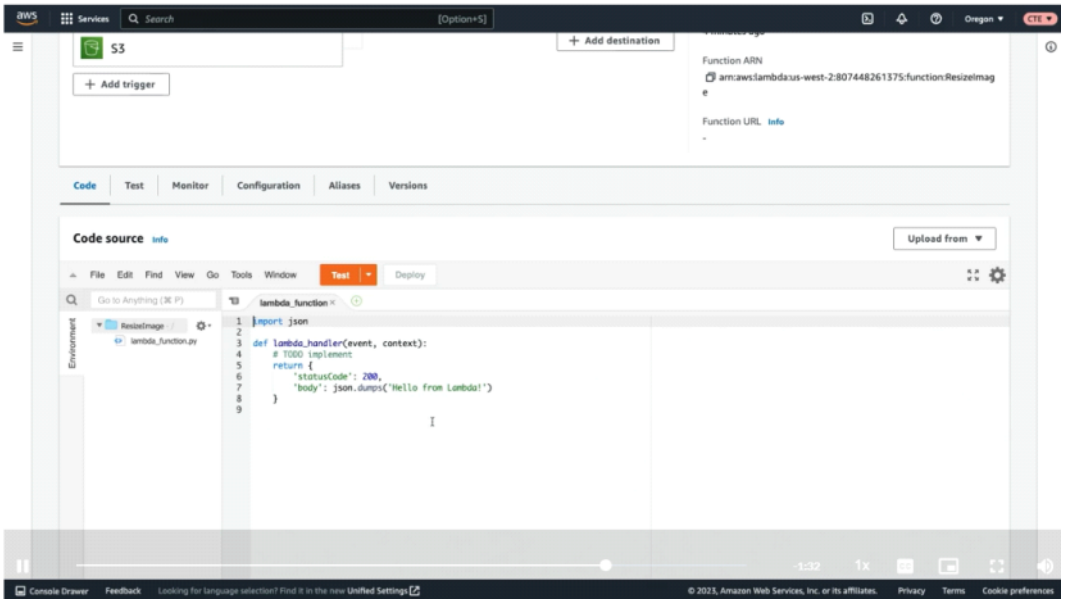
go ahead and acknowledge by checking this checkbox and then click Add.



Now back on our Function overview page, what we need to do next is actually upload the code for this.

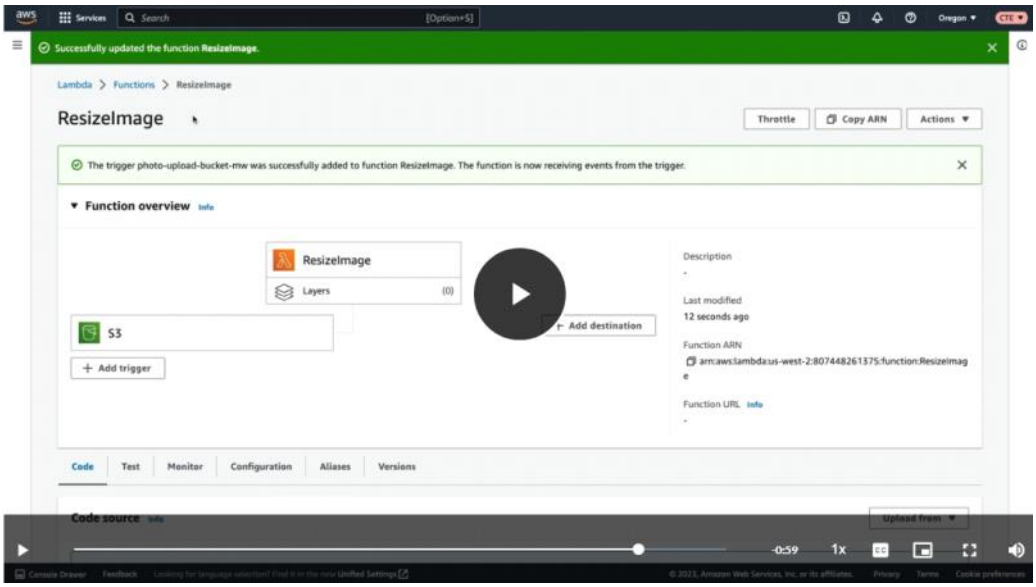


So if we scroll down, we can select the Code tab. And then you can see that there's a stubbed-out Lambda function here that essentially just says, "Hello world! Hello from Lambda!"

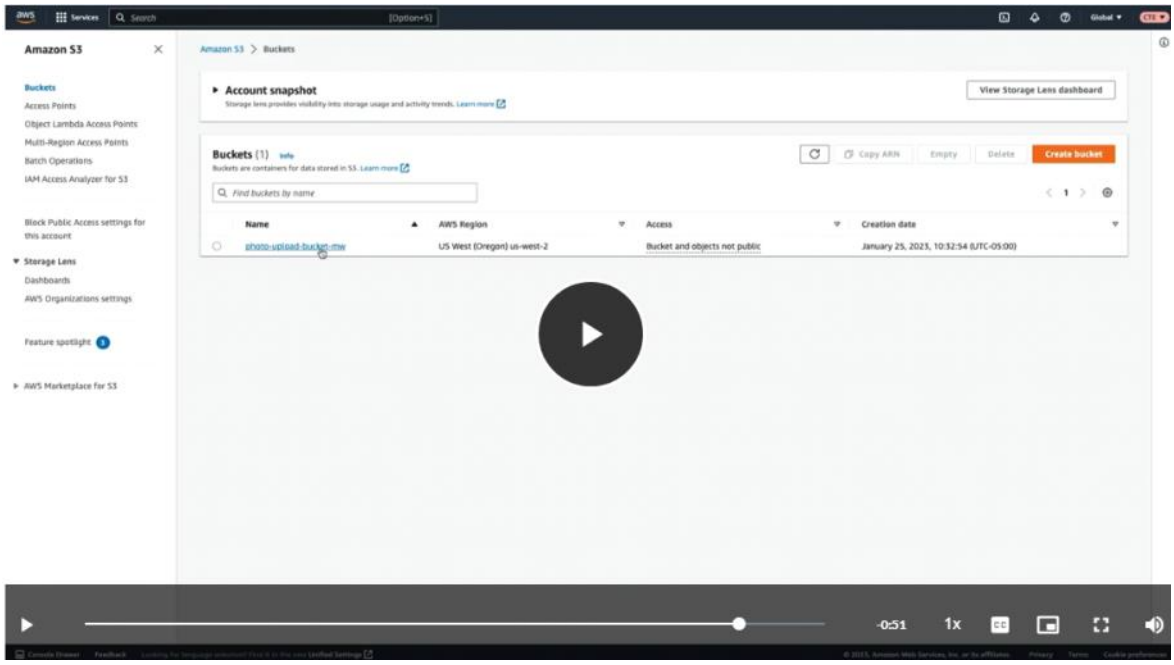


And what we need to do is upload our code. So I'm going to click Upload from, and then select zip file. Click Upload, and then I will select this ResizeImage.zip, click Open, and then click Save.

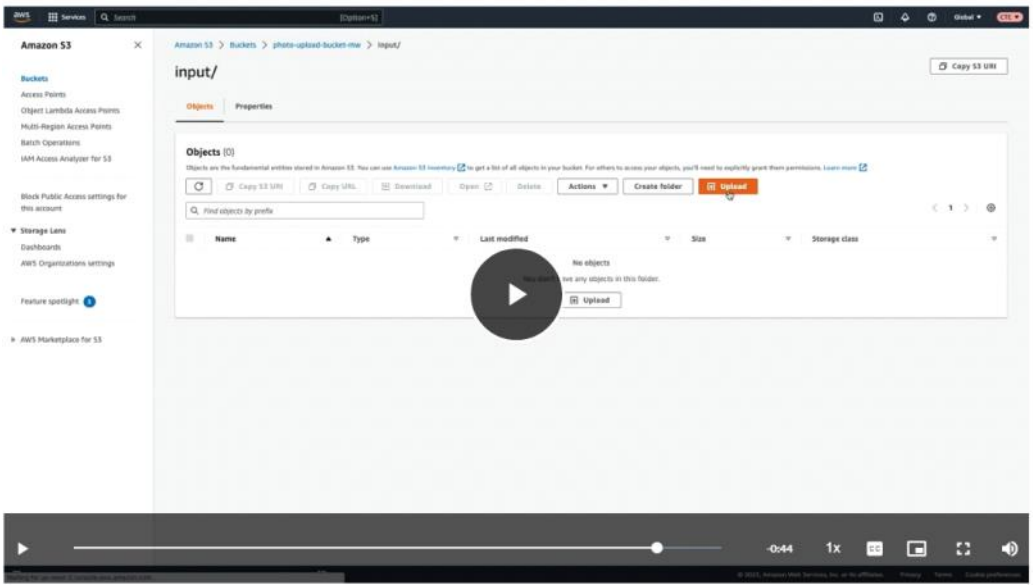
All right, our function has been successfully created. We have our trigger defined, and we have our code uploaded.

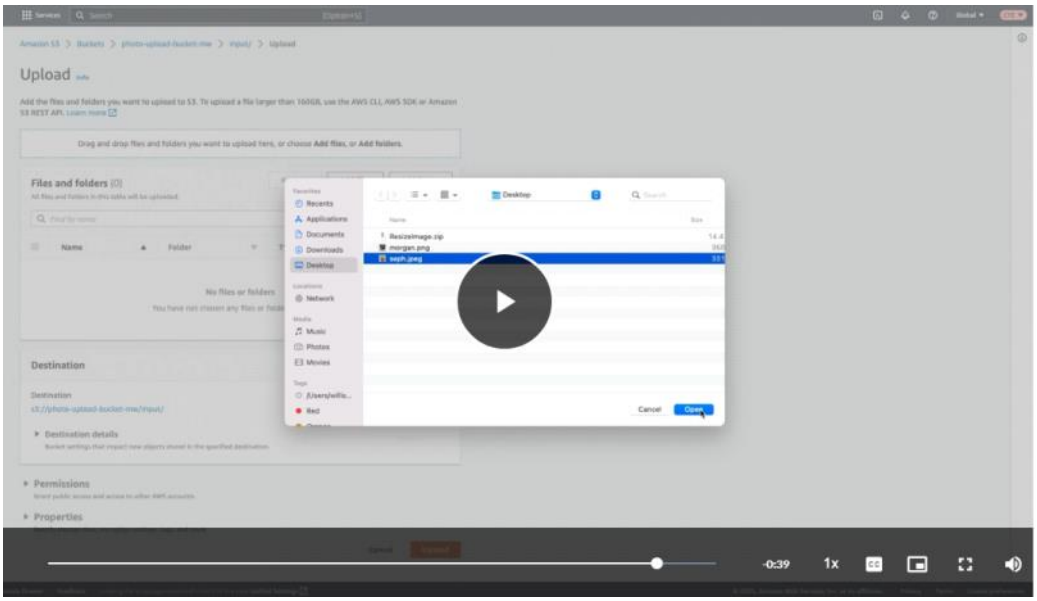


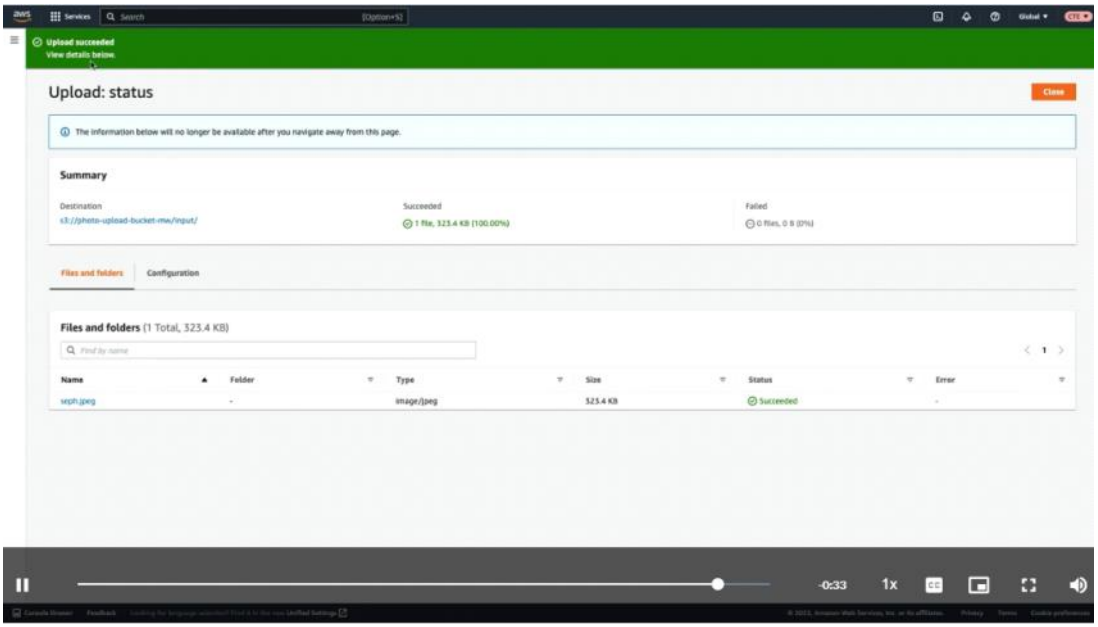
What I want to do now is test this out by actually uploading an image to S3 and then viewing the output. So in the search bar, I will type in S3 and then select S3.



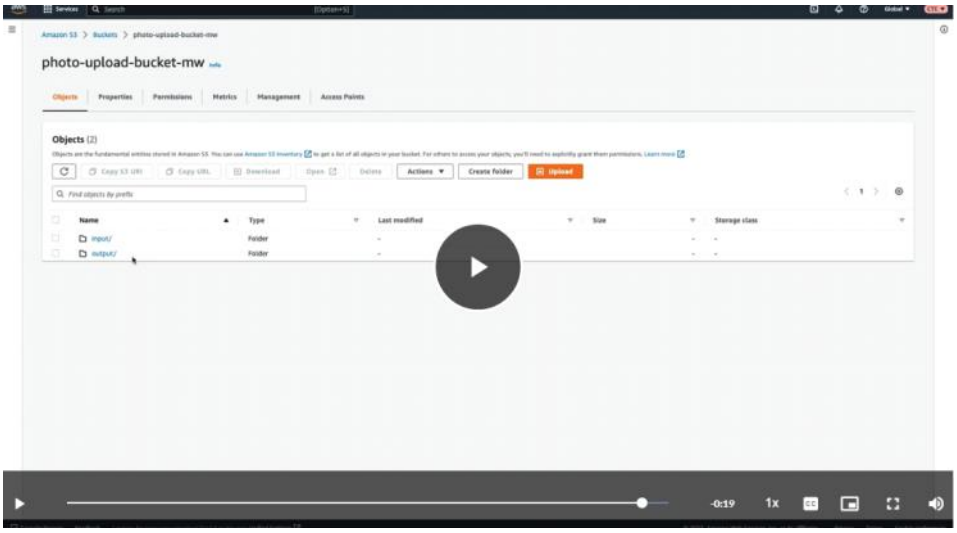
Then from here we will select the bucket, and then we will go into the input folder and then select Upload. Click Add files, and then I will upload Seph's image, and select Open, and then Upload.

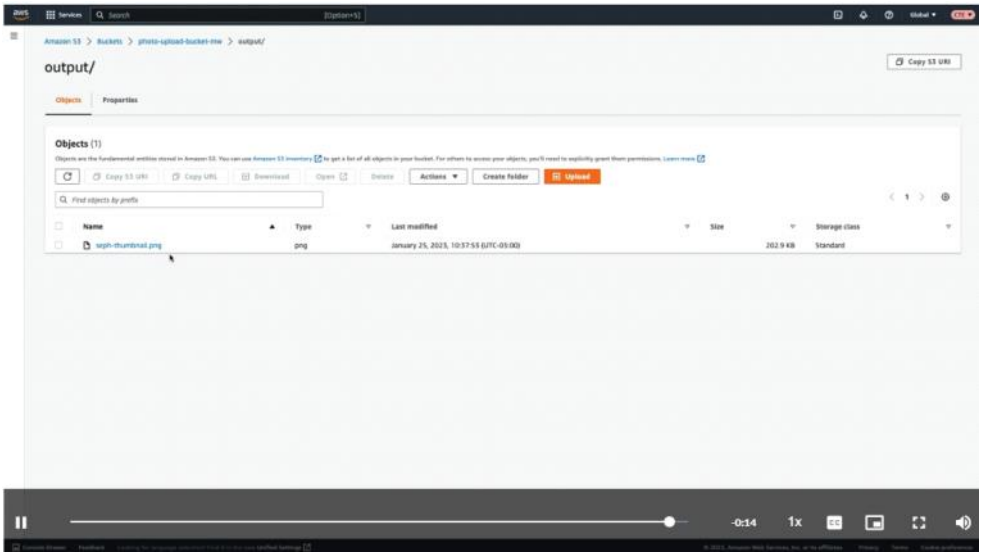






We can see that that upload was successful. So now if we go check, we should see the output with the thumbnail version of that image.





So now going back into our bucket, if we go up one level, we can now see that the output prefix has been created, so I

will select that. And we can see that the thumbnail image was created. And that's it, that's how you create an AWS Lambda function. You could host the entire employee directory application's backend on Lambda with some refactoring, but I'm going to go ahead and save that for a later conversation.

Module-3 Introduction to Networking

21 May 2024 14:57

Networking is how you connect computers around the world and allow them to communicate with one another.

Networking defined

Networking is how you connect computers around the world and allow them to communicate with one another. In this course, you've already seen a few examples of networking. One is the AWS Global Infrastructure. AWS has built a network of resources using data centers, Availability Zones, and Regions.

Networking basics

One way to think about networking is to think about sending a letter. When you send a letter, you provide the following three elements:

1. The letter, inside the envelope
2. The address of the sender in the from section
3. The address of the recipient in the to section

Each address must contain specific information:

1. Name of sender or recipient
2. Street
3. City
4. State or province
5. Zip, area, or postal code
6. Country

You need all parts of an address to ensure that your letter gets to its destination. Without the correct address, postal workers cannot properly deliver the letter. In the digital world, computers handle the delivery of messages in a similar way. This is called routing.

IP addresses

To properly route your messages to a location, you need an address. Just like each home has a mailing address, each computer has an IP address. However, instead of using the combination of street, city, state, zip code, and country, the IP address uses a combination of bits, 0s and 1s.

Here is an example of a 32-bit address in binary format:

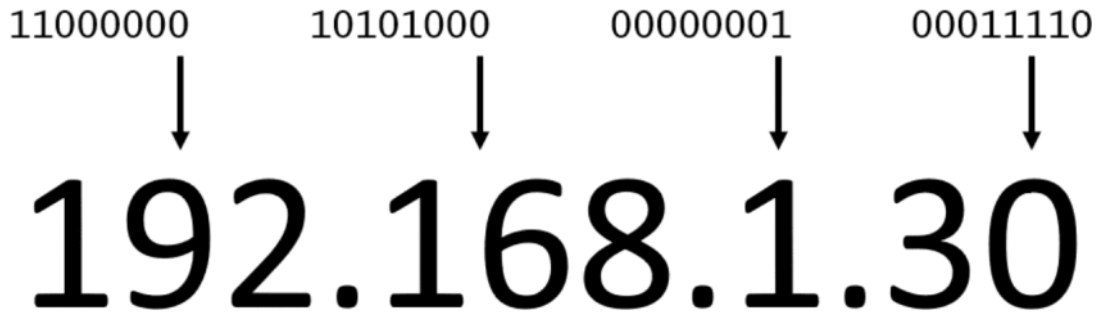
11000000 10101000 00000001 00011110

A 32-bit address written in binary format.

IPv4 notation

Typically, you don't see an IP address in its binary format. Instead, it's converted into decimal format and noted as an IPv4 address.

In the following diagram, the 32 bits are grouped into groups of 8 bits, also called octets. Each of these groups is converted into decimal format separated by a period.



An IPv4 address, for example, 192.168.1.30 is converted from four groups of eight bits.

In the end, this is what is called an IPv4 address. This is important to know when trying to communicate to a single computer. But remember, you're working with a network. This is where Classless Inter-Domain Routing (CIDR) comes in.

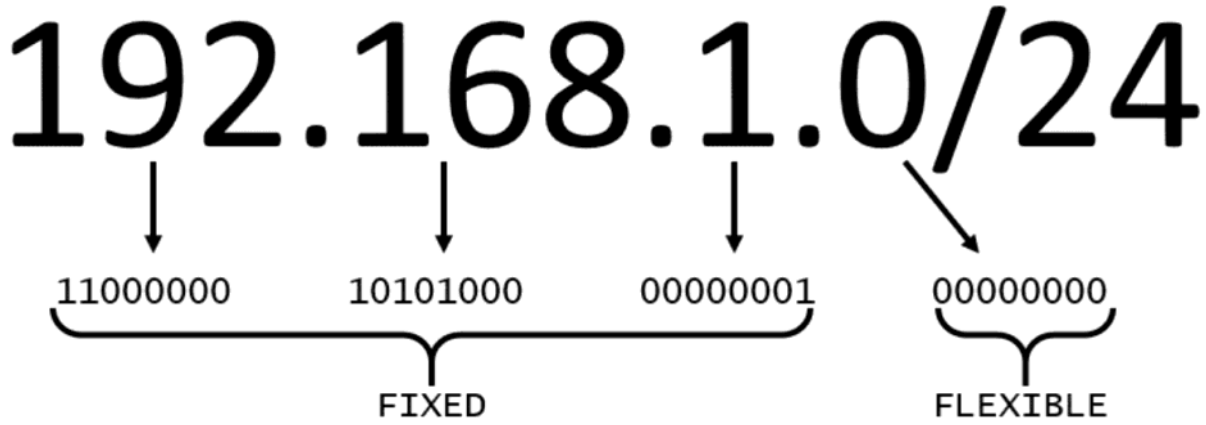
CIDR notation

192.168.1.30 is a single IP address. If you want to express IP addresses between the range of 192.168.1.0 and 192.168.1.255, how can you do that?

One way is to use CIDR notation. CIDR notation is a compressed way of representing a range of IP addresses. Specifying a range determines how many IP addresses are available to you.

CIDR notation is shown here.

It begins with a starting IP address and is separated by a forward slash (the / character) followed by a number. The number at the end specifies how many of the bits of the IP address are fixed. In this example, the first 24 bits of the IP address are fixed. The rest (the last 8 bits) are flexible.



2 total bits subtracted by 24 fixed bits leaves 8 flexible bits. Each of these flexible bits can be either 0 or 1, because they are binary. That means that you have two choices for each of the 8 bits, providing 256 IP addresses in that IP range.

The higher the number after the /, the smaller the number of IP addresses in your network. For example, a range of 192.168.1.0/24 is smaller than 192.168.1.0/16.

When working with networks in the AWS Cloud, you choose your network size by using CIDR notation. In AWS, the smallest IP range you can have is /28, which provides 16 IP addresses. The largest IP range you can have is a /16, which provides 65,536 IP addresses.

Module-3 Amazon VPC

21 May 2024 15:31

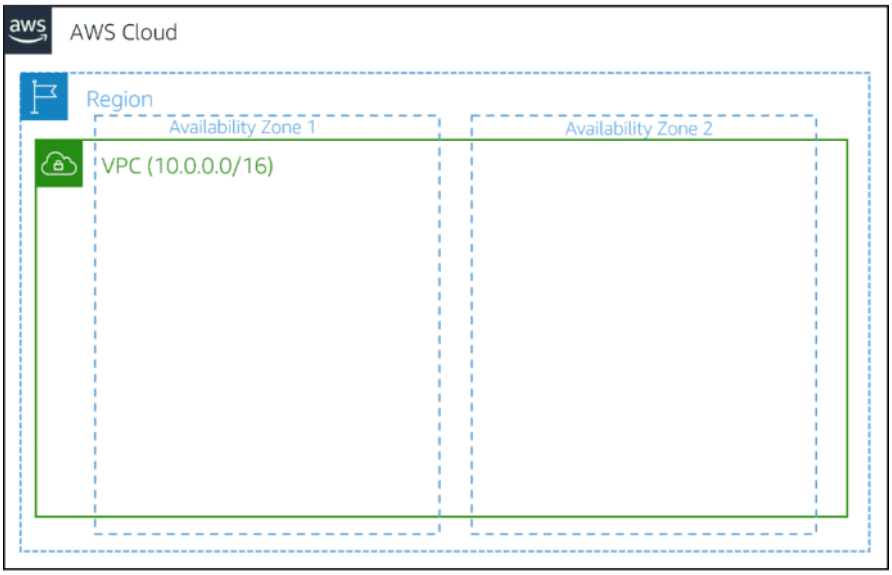
To maintain redundancy and fault tolerance, create at least two subnets configured in two Availability Zones.

Amazon VPC

A virtual private cloud (VPC) is an isolated network that you create in the AWS Cloud, similar to a traditional network in a data center. When you create an Amazon VPC, you must choose three main factors:

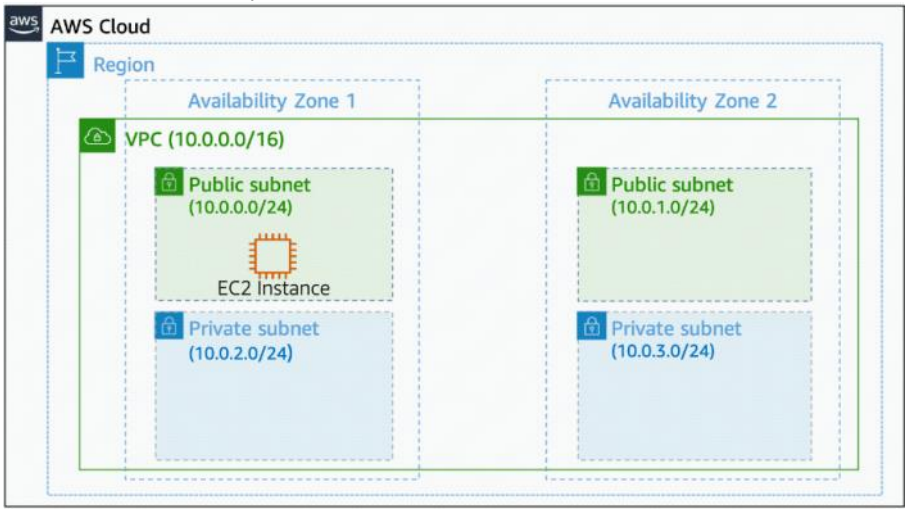
1. Name of the VPC
2. Region where the VPC will live – A VPC spans all the Availability Zones within the selected Region.
3. IP range for the VPC in CIDR notation – This determines the size of your network. Each VPC can have up to five CIDRs: one primary and four secondaries for IPv4. Each of these ranges can be between /28 (in CIDR notation) and /16 in size.

Using this information, AWS will provision a network and IP addresses for that network.



Creating a subnet

After you create your VPC, you must create subnets inside the network. Think of subnets as smaller networks inside your base network, or virtual local area networks (VLANs) in a traditional, on-premises network. In an on-premises network, the typical use case for subnets is to isolate or optimize network traffic. In AWS, subnets are used to provide high availability and connectivity options for your resources. Use a public subnet for resources that must be connected to the internet and a private subnet for resources that won't be connected to the internet.



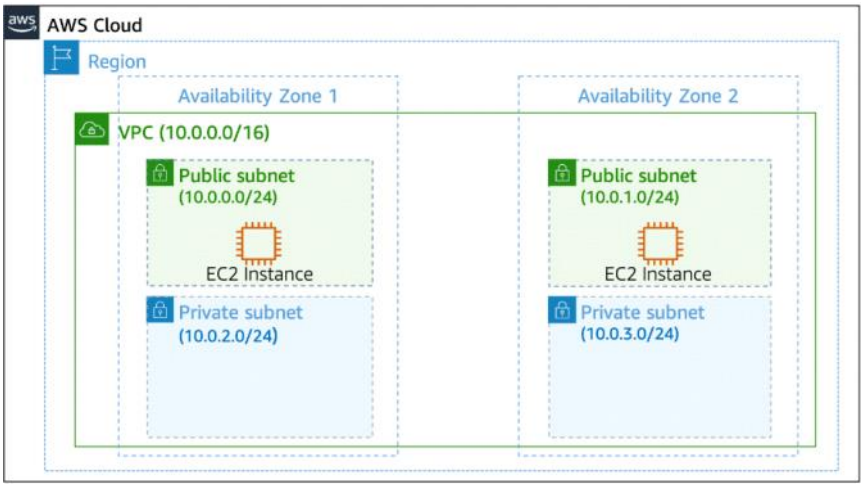
When you create a subnet, you must specify the following:

1. VPC that you want your subnet to live in—in this case: VPC (10.0.0.0/16)
2. Availability Zone that you want your subnet to live in—in this case: Availability Zone 1
3. IPv4 CIDR block for your subnet, which must be a subset of the VPC CIDR block—in this case: 10.0.0.0/24
4. When you launch an EC2 instance, you launch it inside a subnet, which will be located inside the Availability Zone that you choose.

High availability with a VPC

When you create your subnets, keep high availability in mind. To maintain redundancy and fault tolerance, create at least two subnets configured in two Availability Zones.

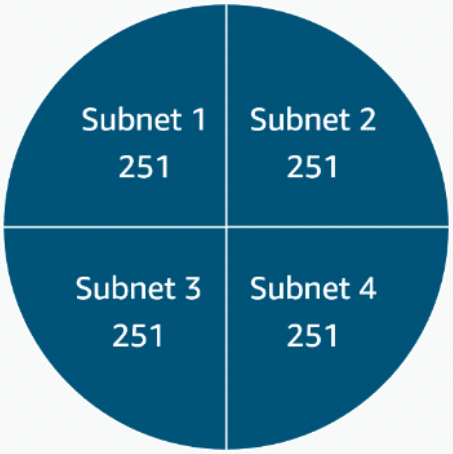
As you learned earlier, remember that "everything fails all of the time." With the example network, if one of the Availability Zones fails, you will still have your resources available in another Availability Zone as backup.



Reserved IPs

For AWS to configure your VPC appropriately, AWS reserves five IP addresses in each subnet. These IP addresses are used for routing, Domain Name System (DNS), and network management.

For example, consider a VPC with the IP range 10.0.0.0/22. The VPC includes 1,024 total IP addresses. This is then divided into four equal-sized subnets, each with a /24 IP range with 256 IP addresses. Out of each of those IP ranges, there are only 251 IP addresses that can be used because AWS reserves five.



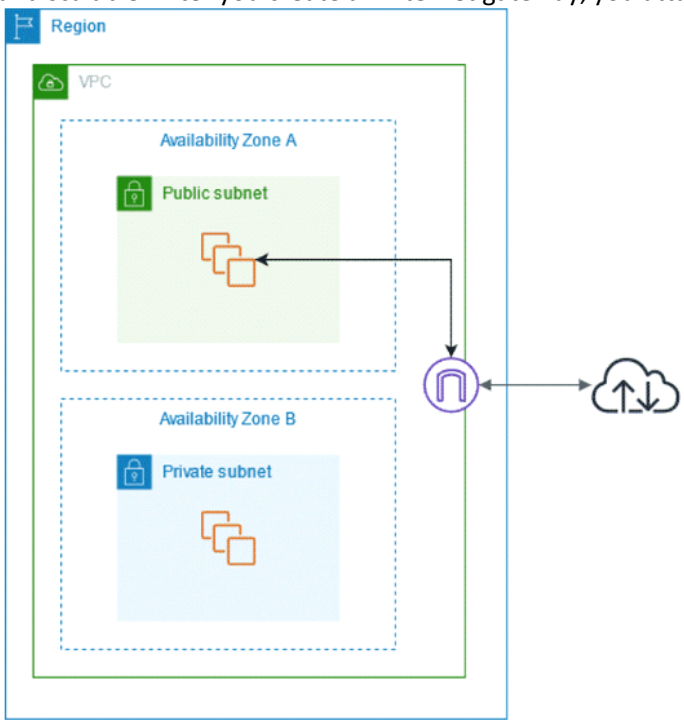
IP address	Reserved for:
10.0.0.0	Network address
10.0.0.1	VPC local router
10.0.0.2	DNS server
10.0.0.3	Future use
10.0.2.255	Network broadcast address

The five reserved IP addresses can impact how you design your network. A common starting place for those who are new to the cloud is to create a VPC with an IP range of /16 and create subnets with an IP range of /24. This provides a large amount of IP addresses to work with at both the VPC and subnet levels.

Gateways

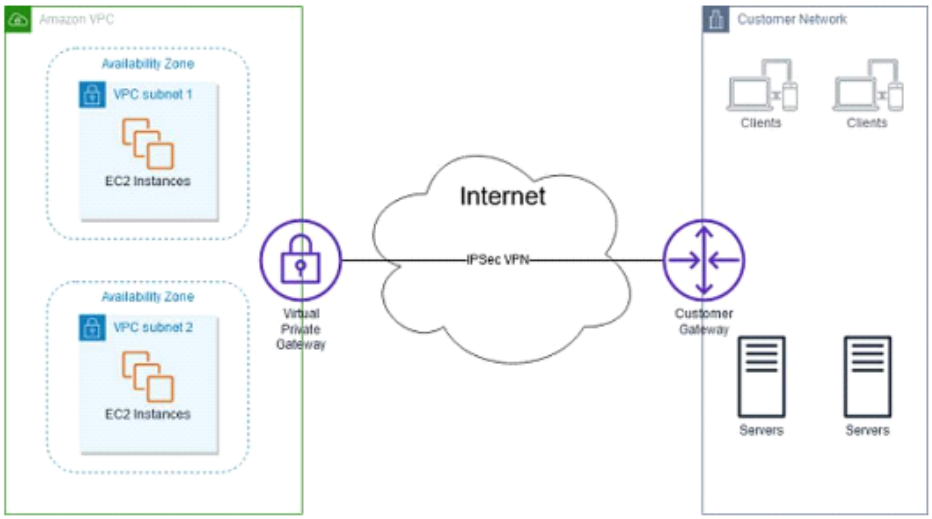
Internet gateway

To activate internet connectivity for your VPC, you must create an internet gateway. Think of the gateway as similar to a modem. Just as a modem connects your computer to the internet, the internet gateway connects your VPC to the internet. Unlike your modem at home, which sometimes goes down or offline, an internet gateway is highly available and scalable. After you create an internet gateway, you attach it to your VPC.



Virtual private gateway

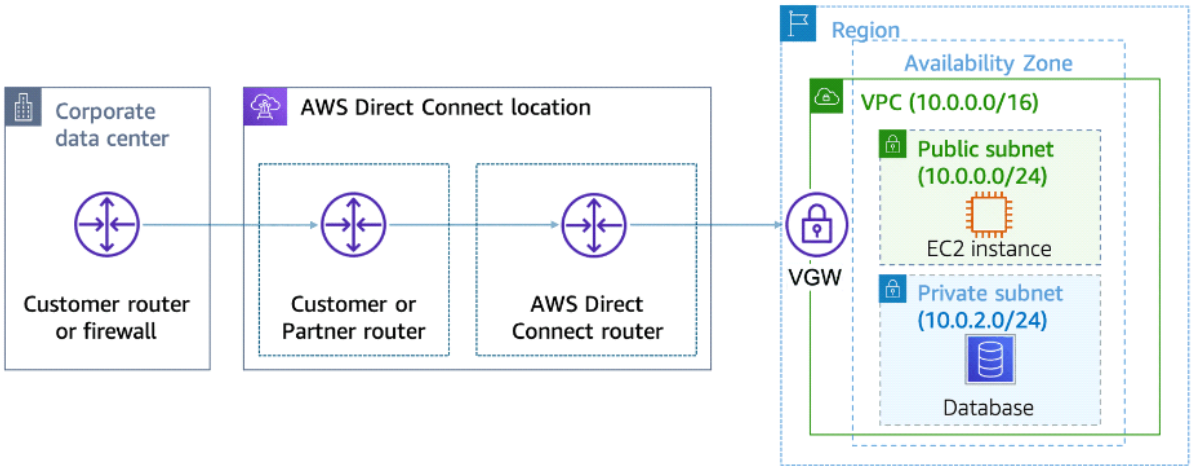
A virtual private gateway connects your VPC to another private network. When you create and attach a virtual private gateway to a VPC, the gateway acts as anchor on the AWS side of the connection. On the other side of the connection, you will need to connect a customer gateway to the other private network. A customer gateway device is a physical device or software application on your side of the connection. When you have both gateways, you can then establish an encrypted virtual private network (VPN) connection between the two sides.



AWS Direct Connect

To establish a secure physical connection between your on-premises data center and your Amazon VPC, you can use AWS Direct Connect. With AWS Direct Connect, your internal network is linked to an AWS Direct Connect location over

a standard Ethernet fiber-optic cable. This connection allows you to create virtual interfaces directly to public AWS services or to your VPC.



Module-3 Amazon VPC Routing

21 May 2024 16:51

A route table contains a set of rules, called routes, that determine where network traffic from your subnet or gateway is directed.

Main route table

When you create a VPC, AWS creates a route table called the main route table. A route table contains a set of rules, called routes, that are used to determine where network traffic is directed. AWS assumes that when you create a new VPC with subnets, you want traffic to flow between them. Therefore, the default configuration of the main route table is to allow traffic between all subnets in the local network.

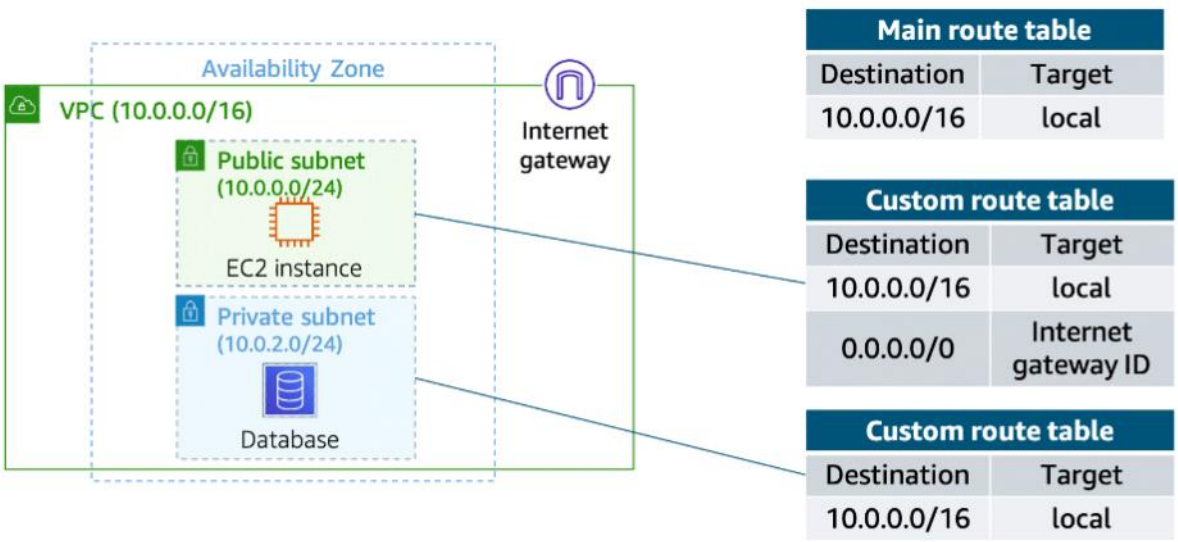
The following rules apply to the main route table:

1. You cannot delete the main route table.
2. You cannot set a gateway route table as the main route table.
3. You can replace the main route table with a custom subnet route table.
4. You can add, remove, and modify routes in the main route table.
5. You can explicitly associate a subnet with the main route table, even if it's already implicitly associated.

Custom route tables

The main route table is used implicitly by subnets that do not have an explicit route table association. However, you might want to provide different routes on a per-subnet basis for traffic to access resources outside of the VPC. For example, your application might consist of a frontend and a database. You can create separate subnets for the resources and provide different routes for each of them.

If you associate a subnet with a custom route table, the subnet will use it instead of the main route table. Each custom route table that you create will have the local route already inside it, allowing communication to flow between all resources and subnets inside the VPC. You can protect your VPC by explicitly associating each new subnet with a custom route table and leaving the main route table in its original default state.



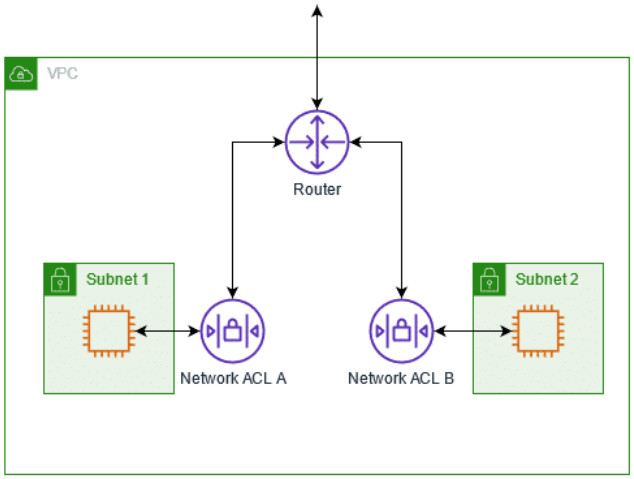
Module-3 Amazon VPC Security

21 May 2024 16:59

Cloud security at AWS is the highest priority. You benefit from a data center and network architecture that is built to meet the requirements of the most security-sensitive organizations.

Secure subnets with network access control lists

A network ACL allows or denies specific inbound or outbound traffic at the subnet level.
Think of a network access control list (network ACL) as a virtual firewall at the subnet level. A network ACL lets you control what kind of traffic is allowed to enter or leave your subnet. You can configure this by setting up rules that define what you want to filter. Here is an example of a default ACL for a VPC that supports IPv4.



Expand the following block for an example of a default network ACL.

Default network ACL

**Note: For users with screen readers, use table mode to read the table.

Inbound					
Rule #	Type	Protocol	Port Range	Source	Allow or Deny
100	All IPv4 traffic	All	All	0.0.0.0/0	ALLOW
*	All IPv4 traffic	All	All	0.0.0.0/0	DENY

Outbound					
Rule #	Type	Protocol	Port Range	Destination	Allow or Deny
100	All IPv4 traffic	All	All	0.0.0.0/0	ALLOW
*	All IPv4 traffic	All	All	0.0.0.0/0	DENY

The default network ACL shown in the preceding table, allows all traffic in and out of the subnet. To allow data to flow freely to the subnet, this is a good starting place.

However, you might want to restrict data at the subnet level. For example, if you have a web application, you might restrict your network to allow HTTPS traffic and Remote Desktop Protocol (RDP) traffic to your web servers.

Expand the following block or an example of a custom network ACL.

Custom network ACL –

**Note: For users with screen readers, use table mode to read the table.

Inbound					
Rule #	**Source IP**	**Protocol**	**Port**	**Allow or Deny**	**Comments**
100	All IPv4 traffic	TCP	443	ALLOW	Allows inbound HTTPS traffic from anywhere
130	192.0.2.0/24	TCP	3389	ALLOW	Allows inbound RDP traffic to the web servers from your home network's public IP address range (over the internet gateway)
*	All IPv4 traffic	All	All	DENY	Denies all inbound traffic not already handled by a preceding rule (not modifiable)

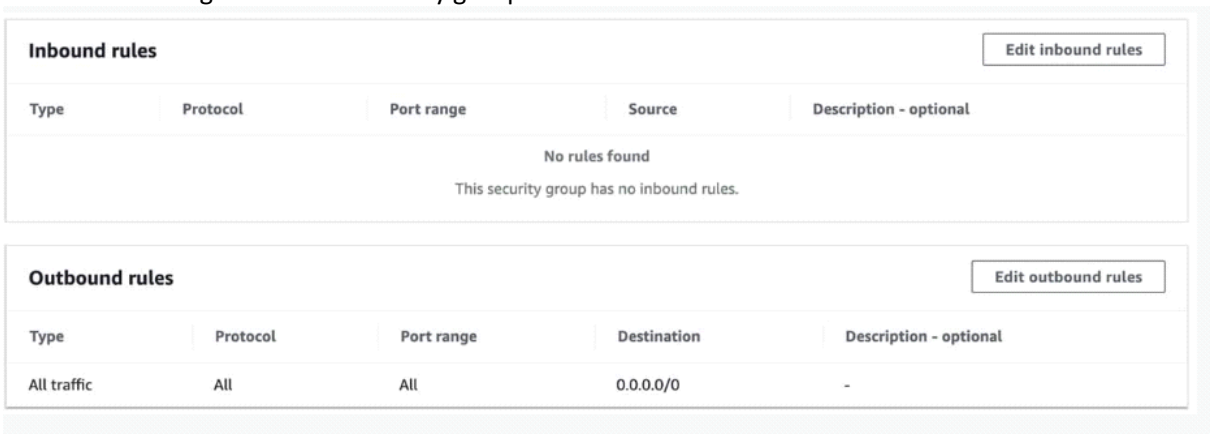
Outbound					
Rule #	**Destination IP**	**Protocol**	**Port**	**Allow or Deny**	**Comments**
120	0.0.0.0/0	TCP	1025-65535	ALLOW	Allows outbound responses to clients on the internet (serving people visiting the web servers in the subnet)
*	0.0.0.0/0	All	All	DENY	Denies all outbound traffic not already handled by a preceding rule (not modifiable)

Notice that in the custom network ACL in the preceding example, you allow inbound 443 and outbound range 1025–65535. That's because HTTPS uses port 443 to initiate a connection and will respond to an ephemeral port. Network ACLs are considered stateless, so you need to include both the inbound and outbound ports used for the protocol. If you don't include the outbound range, your server would respond but the traffic would never leave the subnet.

Because network ACLs are configured by default to allow incoming and outgoing traffic, you don't need to change their initial settings unless you need additional security layers.

Secure EC2 instances with security groups

The next layer of security is for your EC2 instances. Here, you can create a virtual firewall called a security group. The default configuration of a security group blocks all inbound traffic and allows all outbound traffic.



By default, a security group only allows outbound traffic. To allow inbound traffic, you must create inbound rules.

You might be wondering, “Wouldn’t this block all EC2 instances from receiving the response of any customer requests?” Well, security groups are stateful. That means that they will remember if a connection is originally initiated by the EC2 instance or from the outside, and temporarily allow traffic to respond without modifying the inbound rules.

If you want your EC2 instance to accept traffic from the internet, you must open up inbound ports. If you have a web server, you might need to accept HTTP and HTTPS requests to allow that type of traffic into your security group. You can create an inbound rule that will allow port 80 (HTTP) and port 443 (HTTPS), as shown.

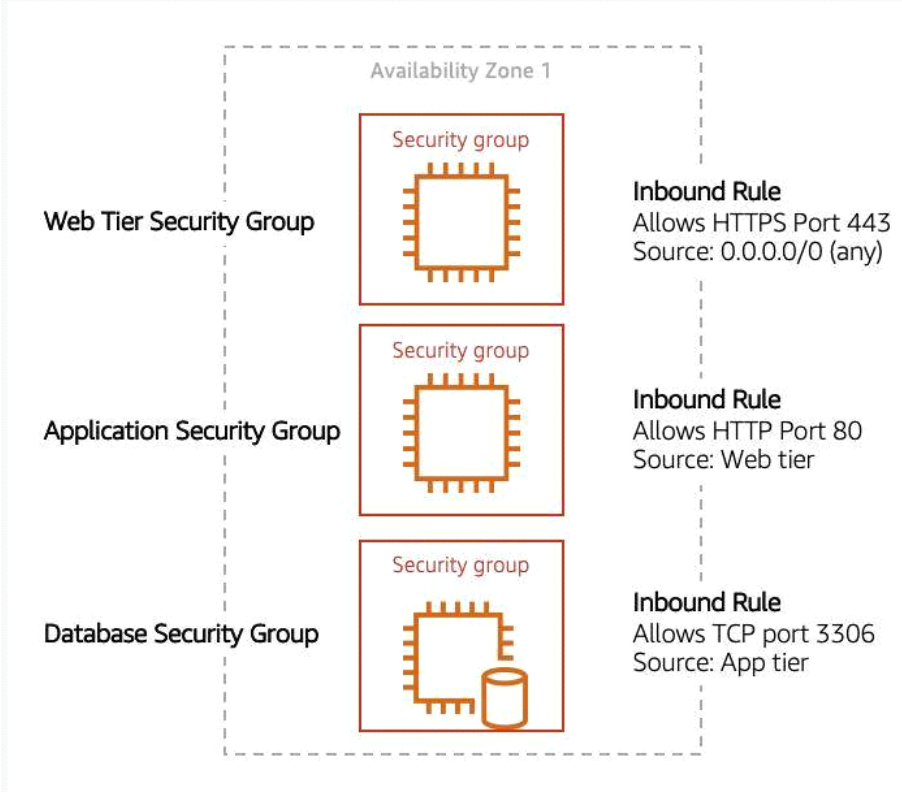
Expand the block below for security group inbound rules.

Security group inbound rules –

**Note: For users with screen readers, use table mode to read the table.

Inbound rules			
Type	**Protocol**	**Port Range**	**Source**
HTTP (80)	TCP (6)	80	0.0.0.0/0
HTTP (80)	TCP (6)	80	::/0
HTTPS (443)	TCP (6)	443	0.0.0.0/0
HTTPS (443)	TCP (6)	443	::/0

You learned earlier that subnets can be used to segregate traffic between computers in your network. Security groups can be used in the same way. A common design pattern is to organize resources into different groups and create security groups for each to control network communication between them.

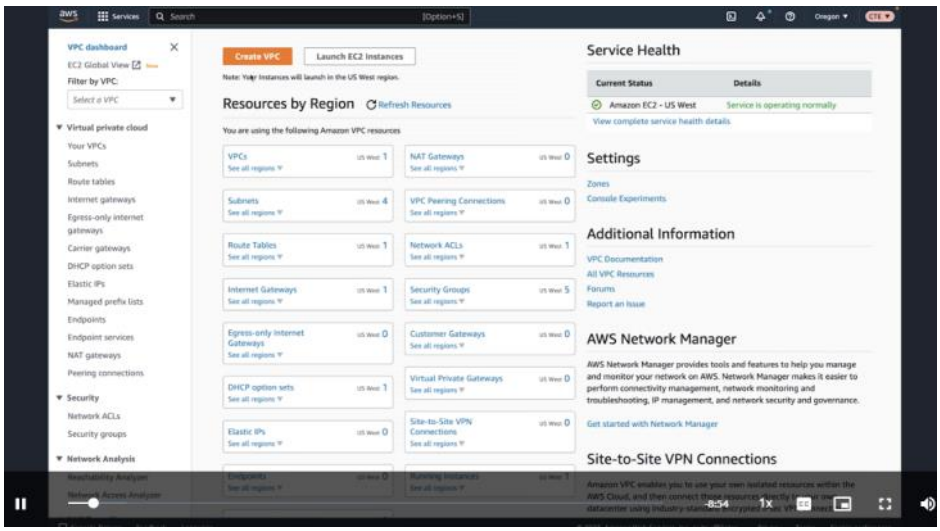


This example defines three tiers and isolates each tier with defined security group rules. In this case, internet traffic to the web tier is allowed over HTTPS. Web tier to application tier traffic is allowed over HTTP, and application tier to database tier traffic is allowed over MySQL. This is different from traditional on-premises environments, in which you isolate groups of resources with a VLAN configuration. In AWS, security groups allow you to achieve the same isolation without tying the security groups to your network.

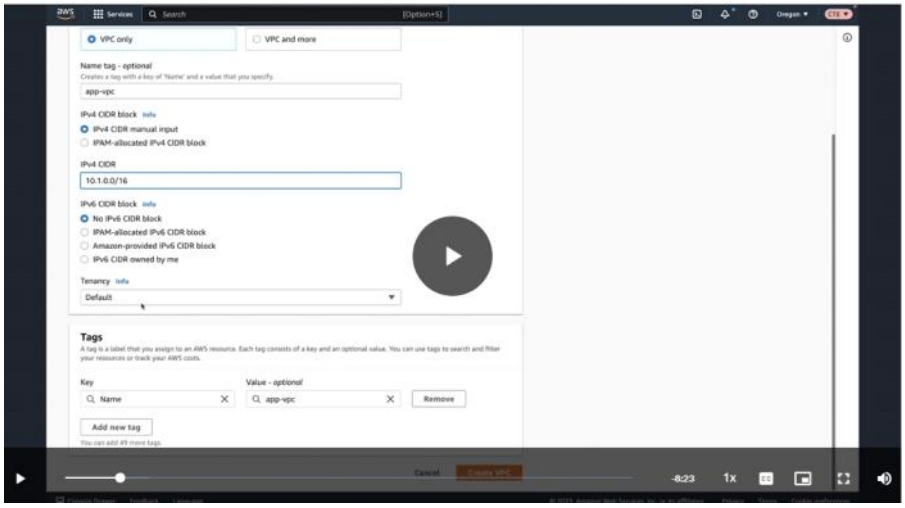
Module-3 Demonstration: Relaunching the Employee Directory Application in Amazon EC2

22 May 2024 14:49

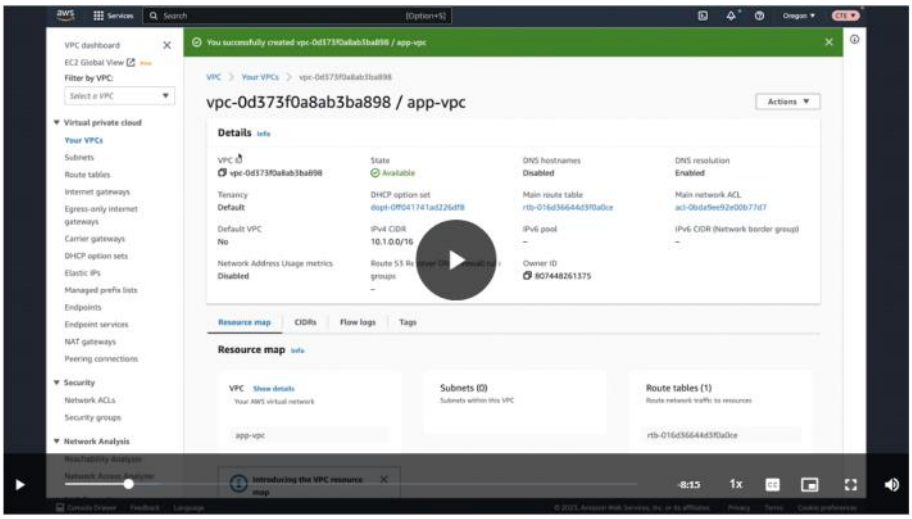
we are going to create a VPC, create four subnets, create a route table for the public subnets, and then we will attach an internet gateway to the VPC and then relaunch the employee directory application in the new VPC.



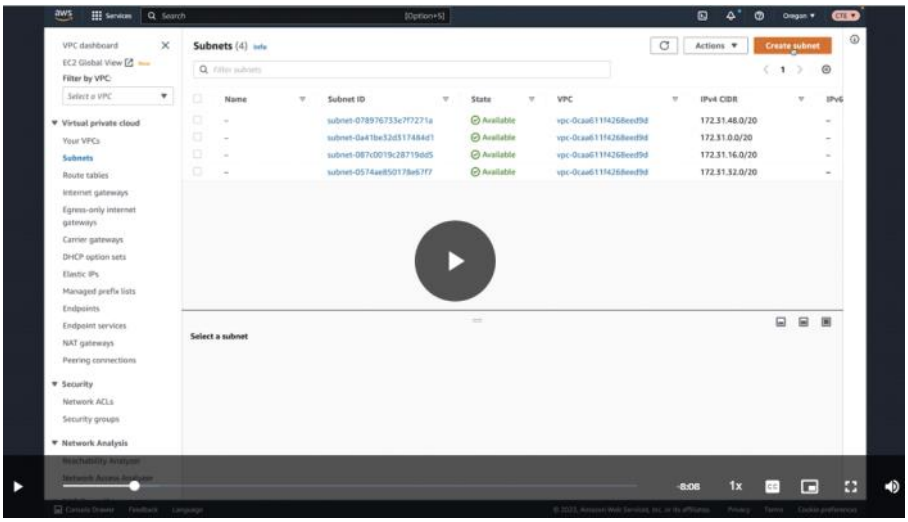
So you can first see that I am in the VPC dashboard here, and I'm going to click on Create VPC. Now I want to select, this is a wizard that's going to help you create your VPCs, your subnets, things like that. I'm going to create just the VPC for now. So I'm going to select VPC only, and then I'm going to give this VPC a name. Let's call it app-vpc. And then I want to enter in the CIDR range for this VPC. So to do that, I'm going to give the CIDR range to be 10.1.0.0/16. And then I'm going to scroll down and click Create VPC.



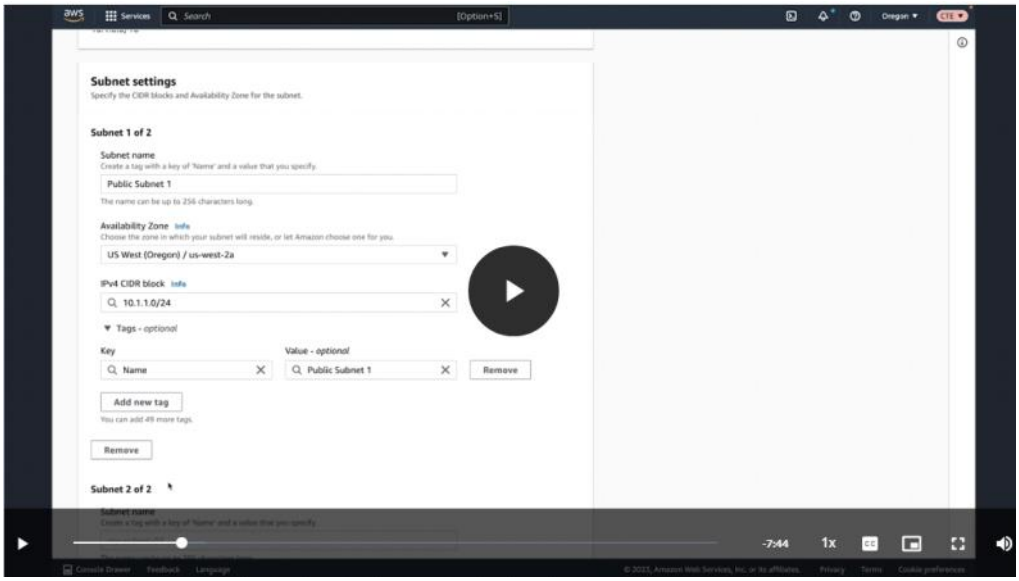
Now we have our VPC.

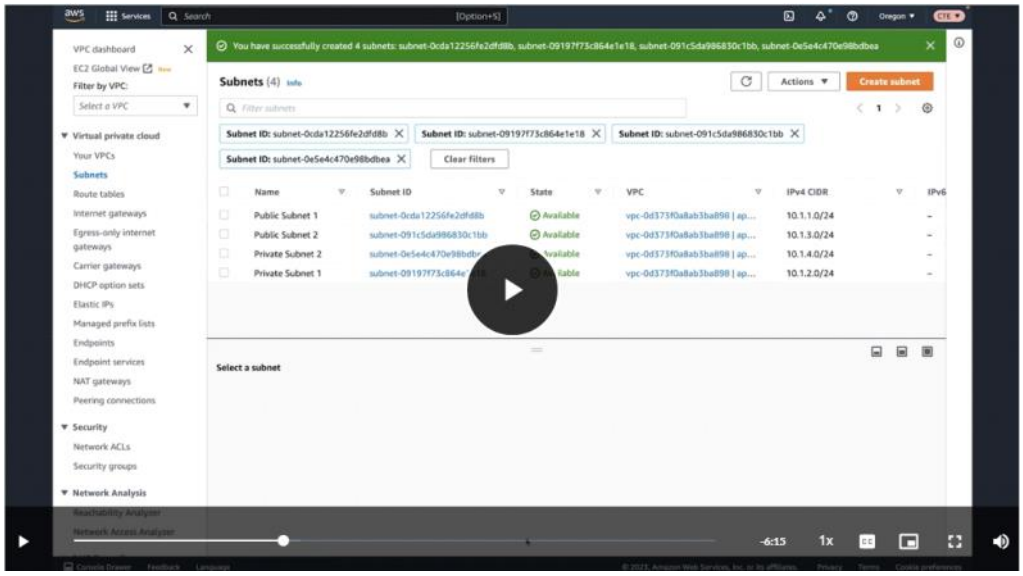


The next thing that I want to do is to create the subnets. So now I'm going to select the subnets on the left-hand navigation and then I'm going to click Create subnet. And then we will select a VPC for this.



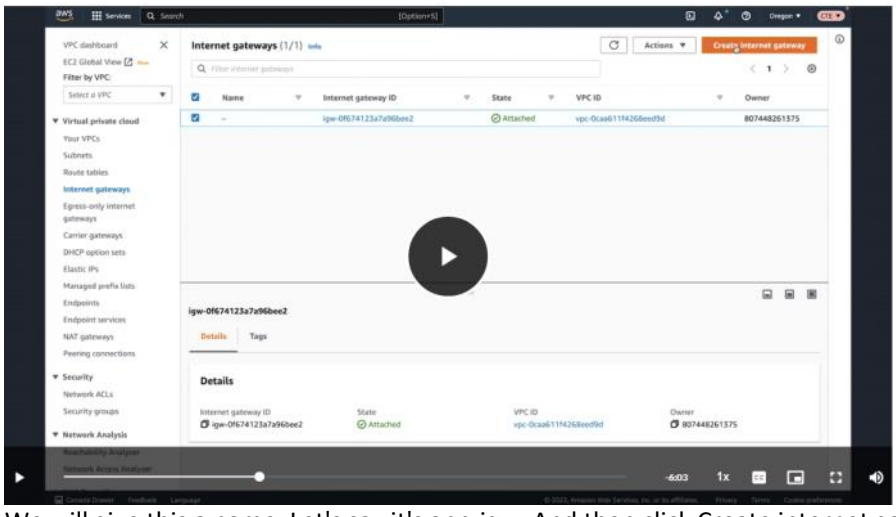
And we will select the VPC we just created, app-vpc. And now for our first subnet, we will give this a name, Public Subnet 1. And then we will select the Availability Zone, which will be us-west-2a. And then we will give this a CIDR range, which is going to be 10.1.1.0/24. And then we can go ahead and add our next subnet. So we're going to have a public and a private subnet in each Availability Zone. So we will create our private one next. So Private Subnet 1, Availability Zone, same one, us-west-2a. And then providing the CIDR range, we will say this is 10.1.2.0/24. And now we can add our subnets into the other Availability Zone. So this one is going to be Public Subnet 2. And let's say this time, we want to put it in us-west-2b. And then for the subnet range we can say 10.1.3.0/24. And then finally we will create one more subnet, which will be our Private Subnet 2. And then selecting the Availability Zone we will say is us-west-2b, and we will give this the CIDR range of 10.1.4.0/24. So you can see that we have, if I scroll back up to the top, you can see we have four different subnets being created with non-overlapping CIDR ranges. All of those CIDR ranges being a subset of the CIDR range that we created for our VPC. So we have 10.1.1.0, 10.1.2.0, 10.1.3.0, and 10.1.4.0. All right, now we can go ahead and click Create subnet and that will create all four.





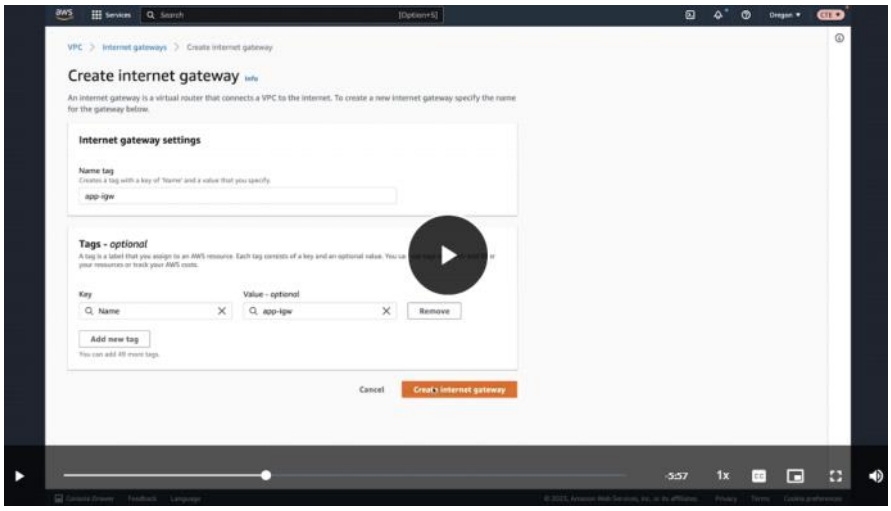
All right, so now we have four subnets, Public Subnet 1 and 2, and Private Subnet 1 and 2.

Next what I want to do is create an internet gateway. So I will select Internet gateways in the left-hand navigation and then click Create internet gateway.

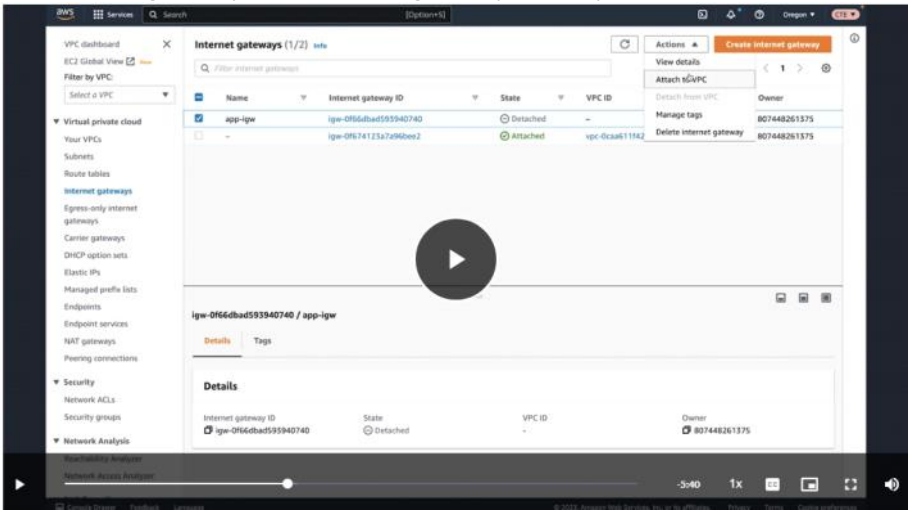


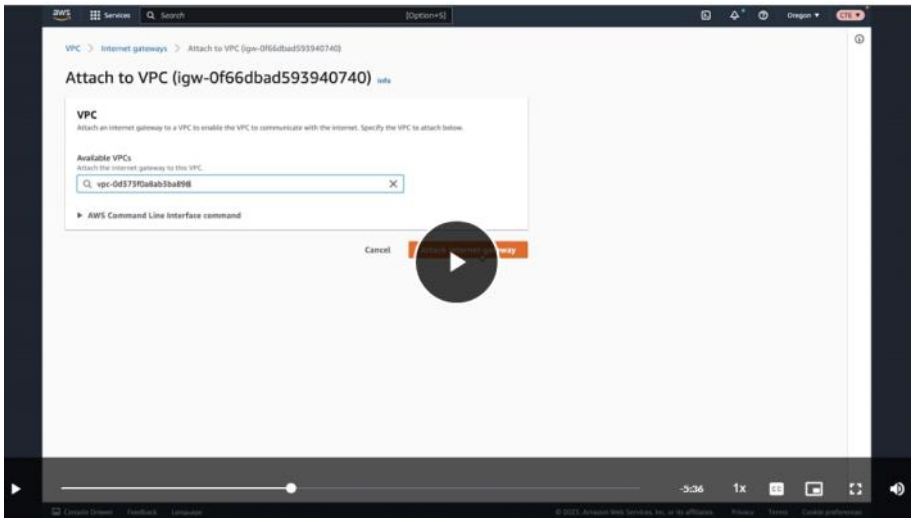
We will give this a name. Let's say it's app-igw. And then click Create internet gateway. Then we can go back to the internet gateways page where we can view all of our internet gateways. We currently have one internet gateway attached to our

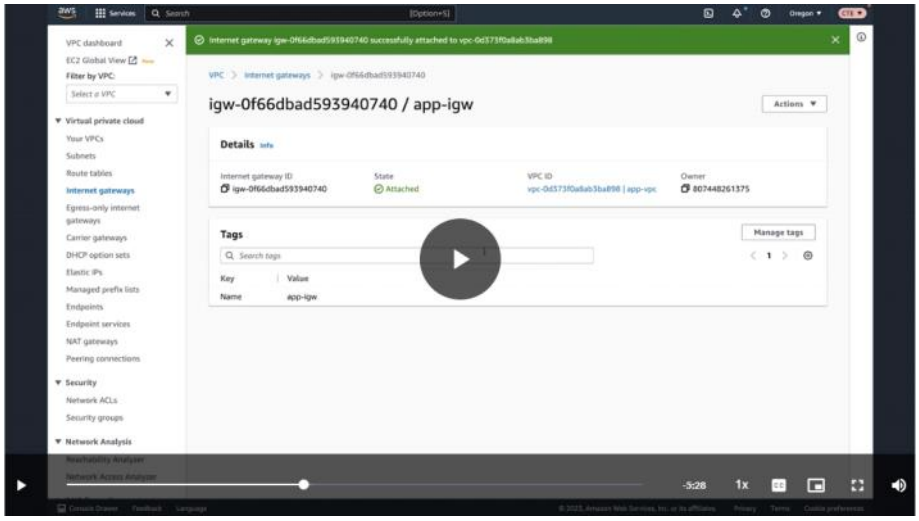
default VPC. What we want to do is select the new internet gateway that we just created, and we want to attach this to a VPC. And then we will select our app-vpc and then click Attach internet gateway.



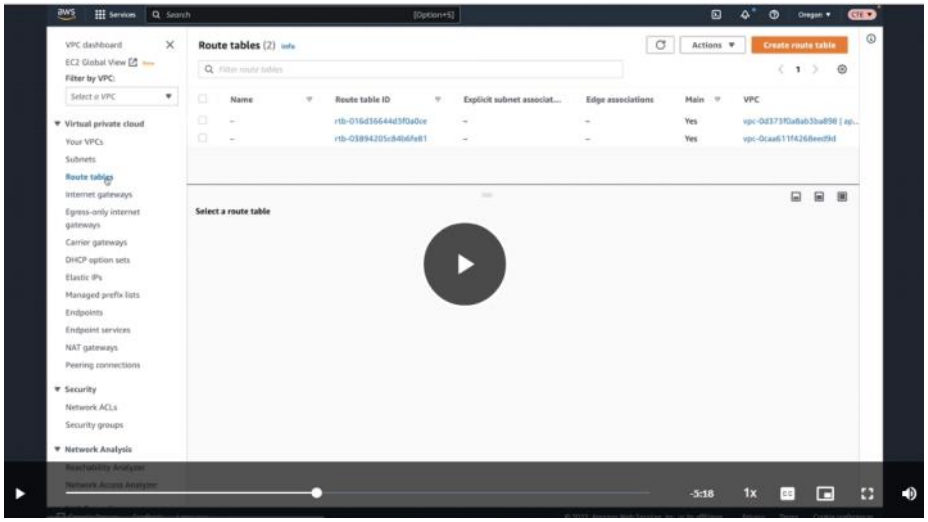
For internet gateways, one internet gateway can only ever be attached to one VPC. So it's a one-to-one relationship there.



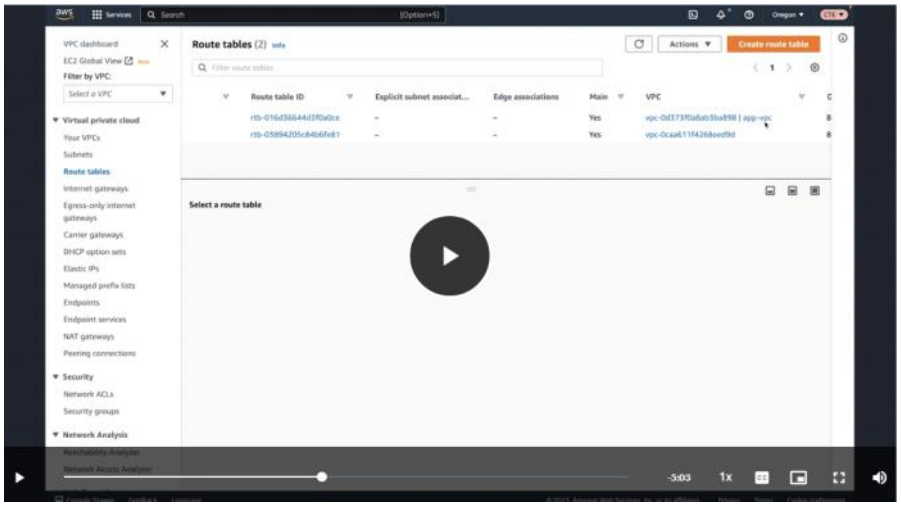




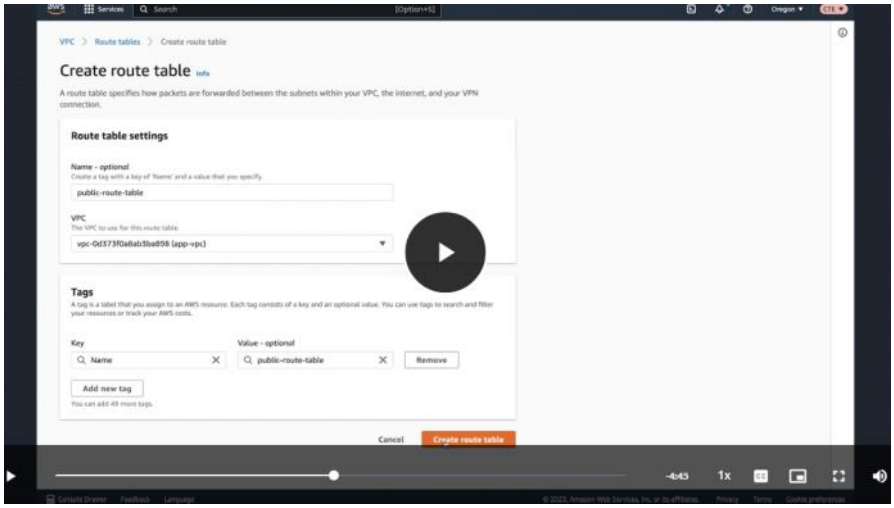
All right, next what we need to do is configure our route tables. So to do that I'm going to click Route tables in the left-hand navigation.

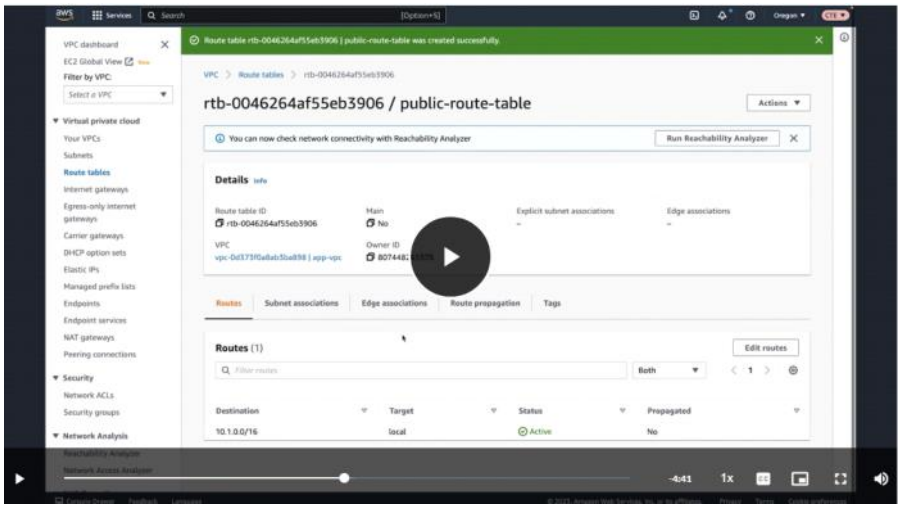


And you can see that we have two route tables already. We have the main route tables for both of our VPCs, our default VPC, and our app-vpc, which we can scroll to the right and expand this and read that VPC there for our app-vpc.

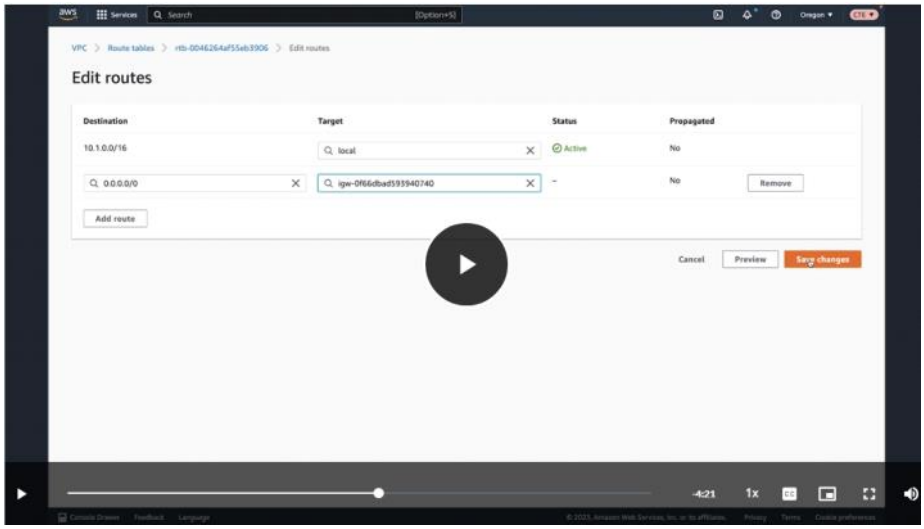


So now scrolling back over, I'm going to click Create route table and then we will give this route table a name. We will say public-route-table. And then I want to associate this with our app-vpc and then click Create route table.

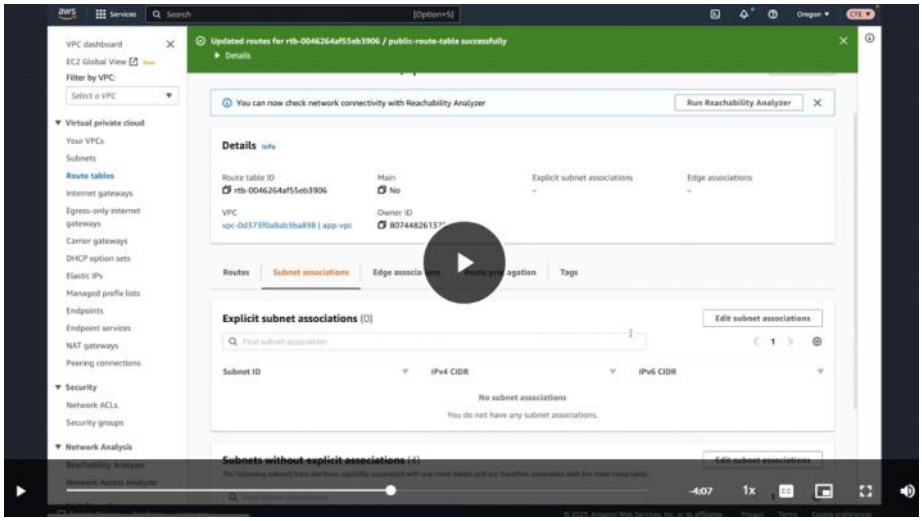




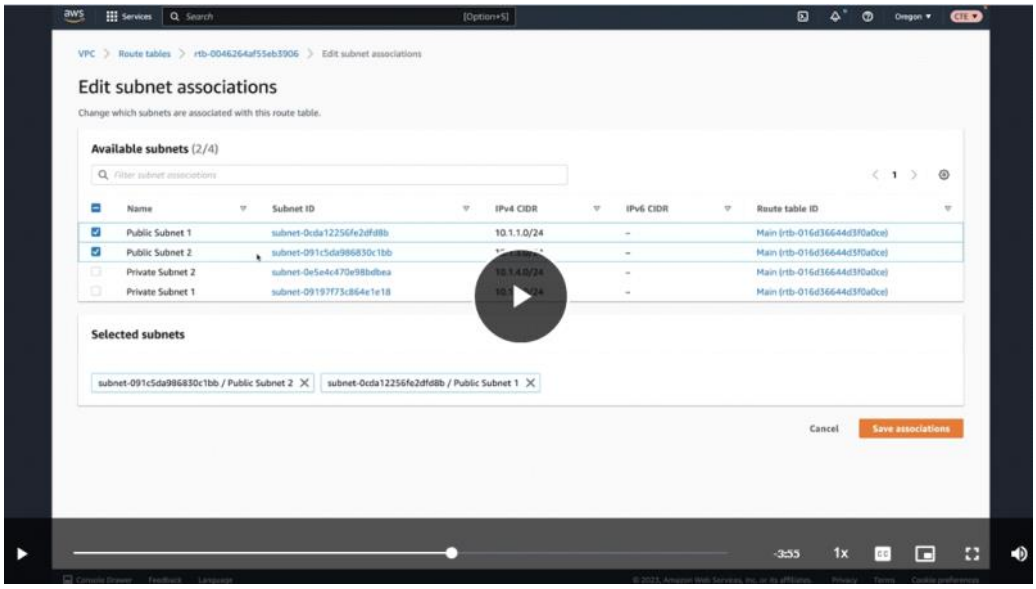
So now we have a route table that's been created but we want to now add a route that will allow any subnet that has this route table associated with it. We want to add a route that will allow traffic from the internet, 0.0.0.0/0. To where? The internet gateway. And the internet gateway we want to choose the one that we just created and then we will click Save changes.



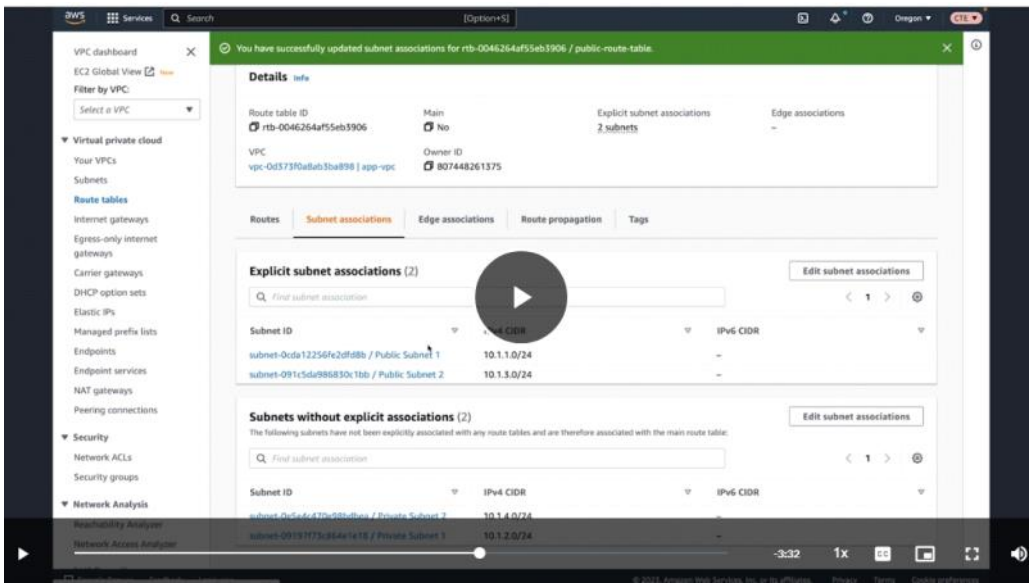
Now, scrolling back down, we are not done yet. Now what we have to do is associate these subnets with this route table. So I will click Subnet associations and then scrolling down, we can see we currently have this associated with no subnets.



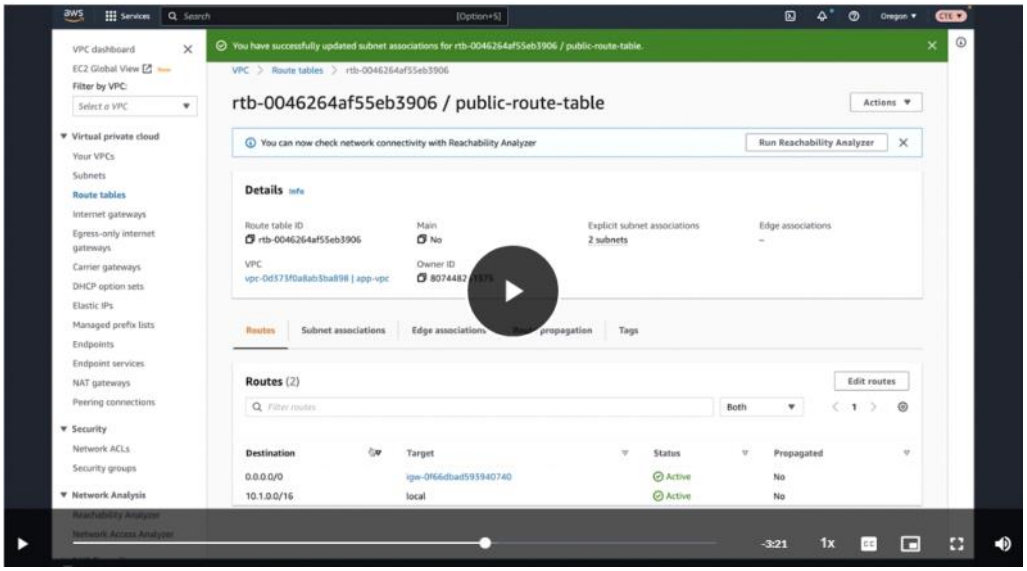
I'm going to click Edit subnet associations and then I'm going to select our first two public subnets only. So there's nothing inherently about a subnet that makes it public or private. The only thing that makes it public or private is whether or not it has a route table association that includes a route from that subnet to the internet gateway. All right, so we will go ahead and click Save associations here.



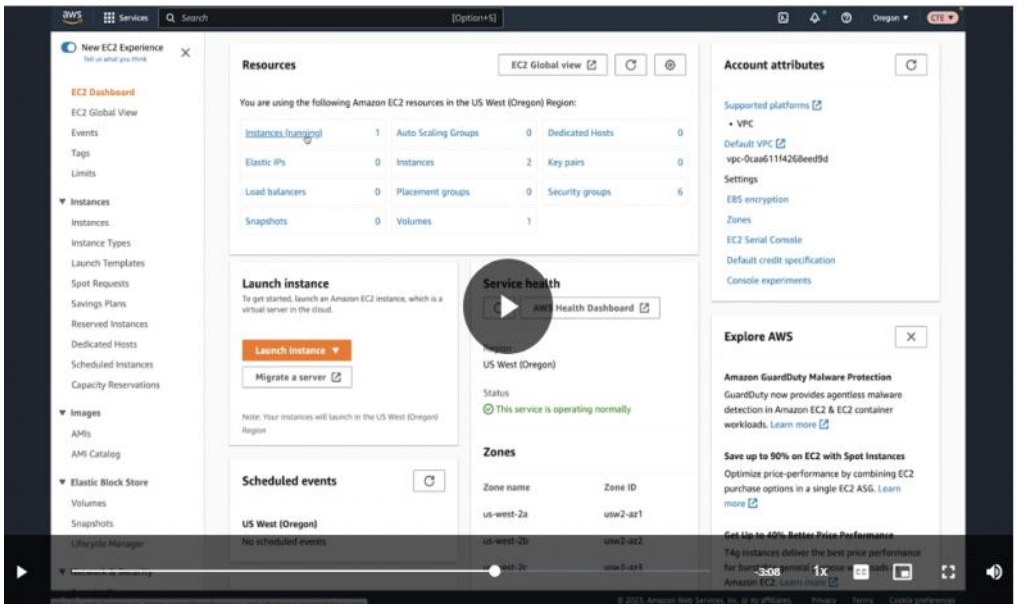
And then we can scroll back down, click on the Subnet associations tab, and we can now see that we have our two public subnets associated with this route table.



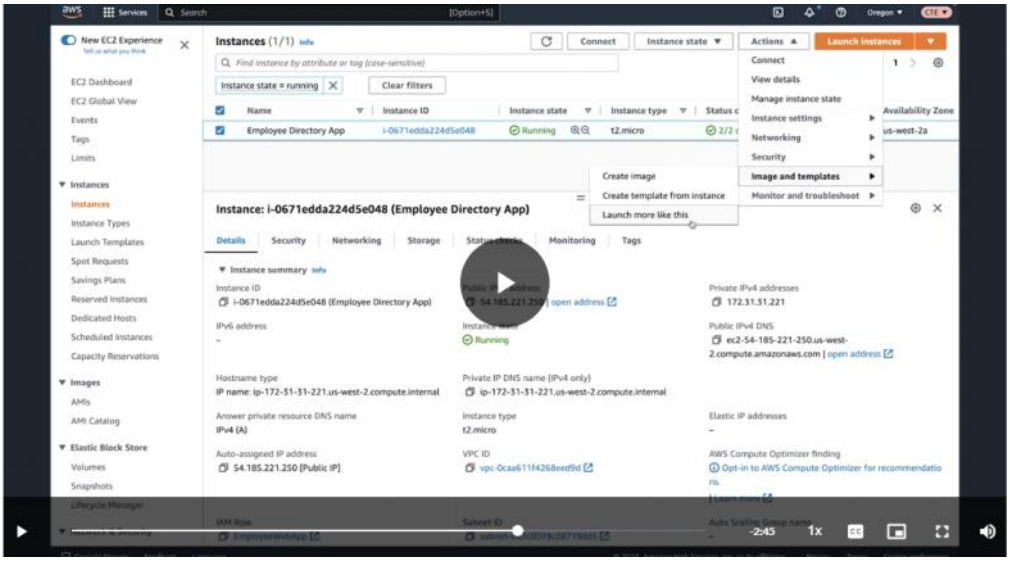
And then again, reviewing the routes, clicking back on the Routes tab. We can see we have our local route and we have our route to the internet through the internet gateway.



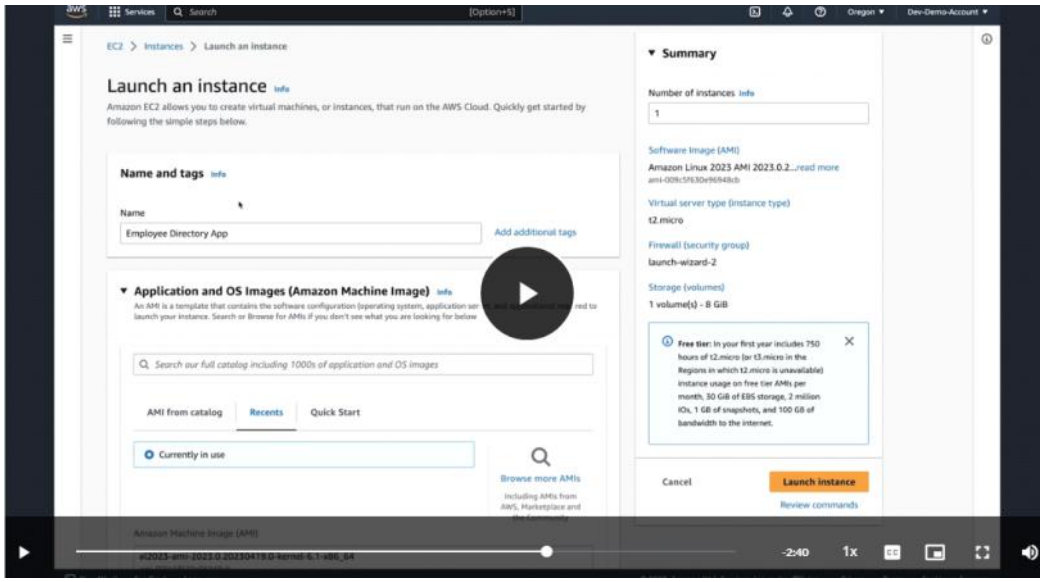
So now the last step here is let's go ahead and relaunch our employee directory application. So I'm going to navigate over to the EC2 console. And then from here I'm going to click on the Instances link, which we can see that we have our employee directory application still running.



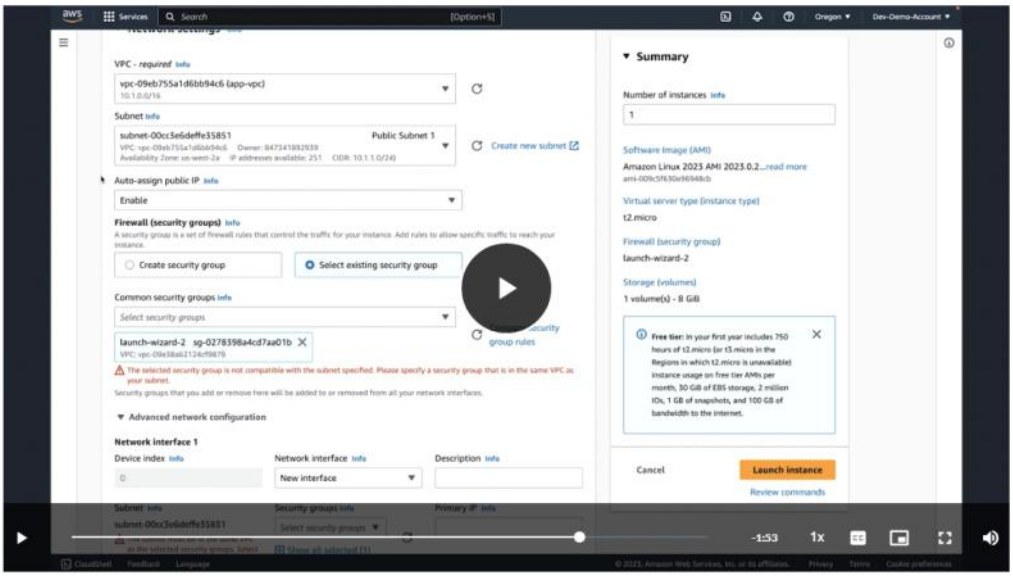
And if I select this, I can scroll down and I can see where this is running, which is our default VPC. So what I'm going to do is I'm going to keep this checked and then I'm going to select Actions and then Image and templates. And then I want to select Launch more like this.



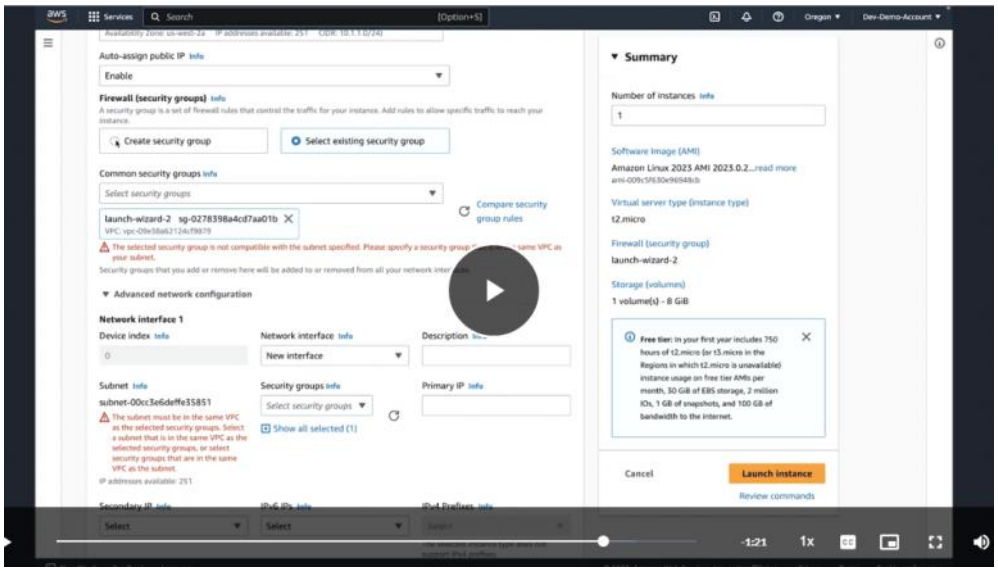
And what this does is it brings you to a page that has a lot of the configurations for that original instance prepopulated over here, so you don't have to go through and reselect all of the configurations.



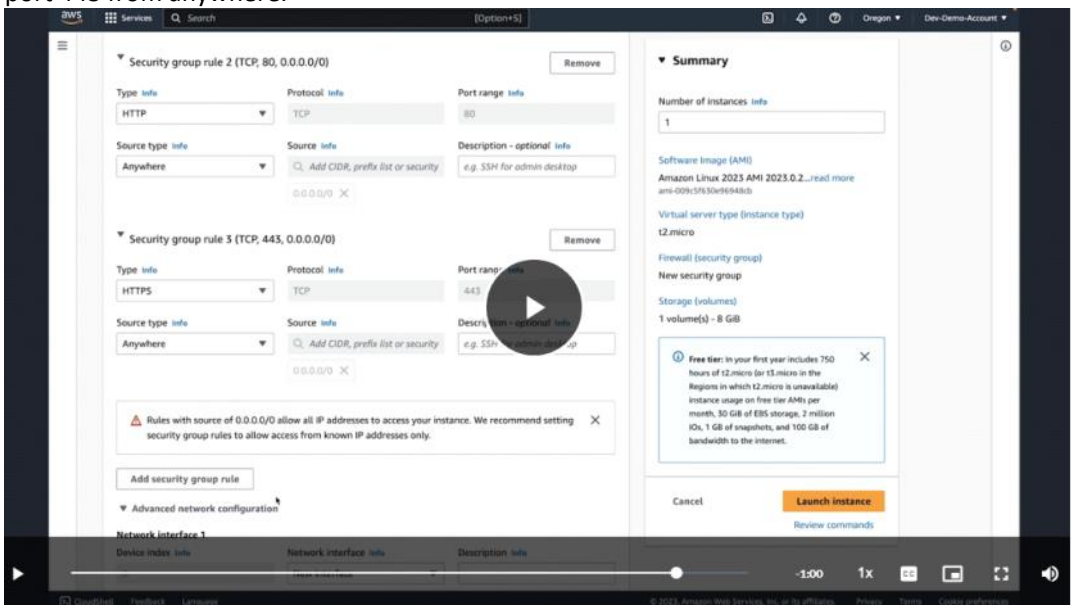
So we can see that we have Employee Directory App prepopulated. I'm going to just go ahead and call this Employee Directory App 2. We can see we have the Linux 2 AMI selected here and we also have our t2.micro selected. For the key pair, we do have to select that we want to proceed without a key pair again. And then here under Network settings this is where we're going to make most of our changes. We're going to select the new app-vpc that we just created, and then we're going to select what public subnet we want to launch into, Public Subnet 1 or Public Subnet 2. We're going to go ahead and select Public Subnet 1. And then we're also ensuring that this auto-assigned public IP is set to Enable, which it is.



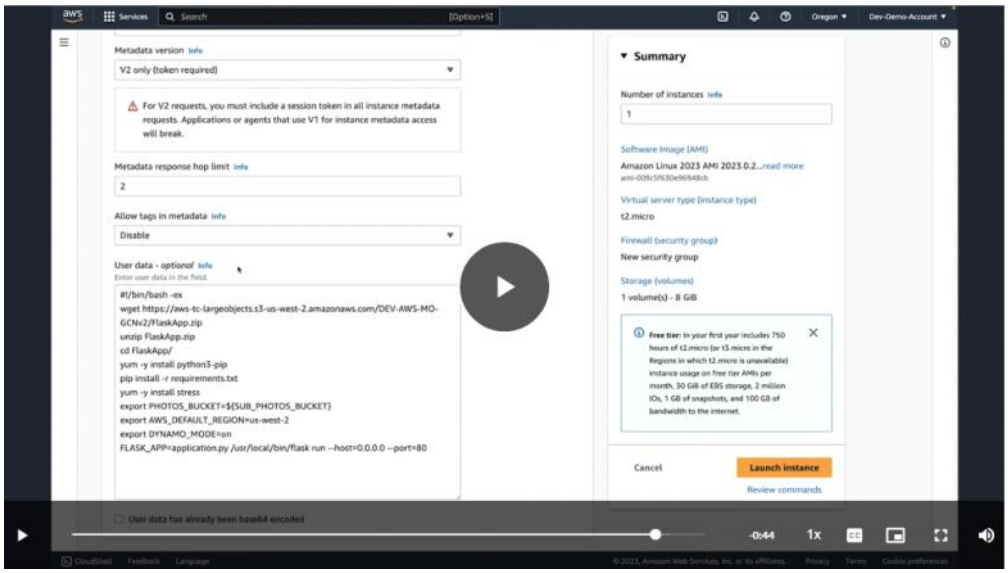
Now scrolling down, you can see that we can select our security group. Currently we have the security group that was created previously associated with this instance and it's giving us an error saying that we can't use the security group with this subnet.



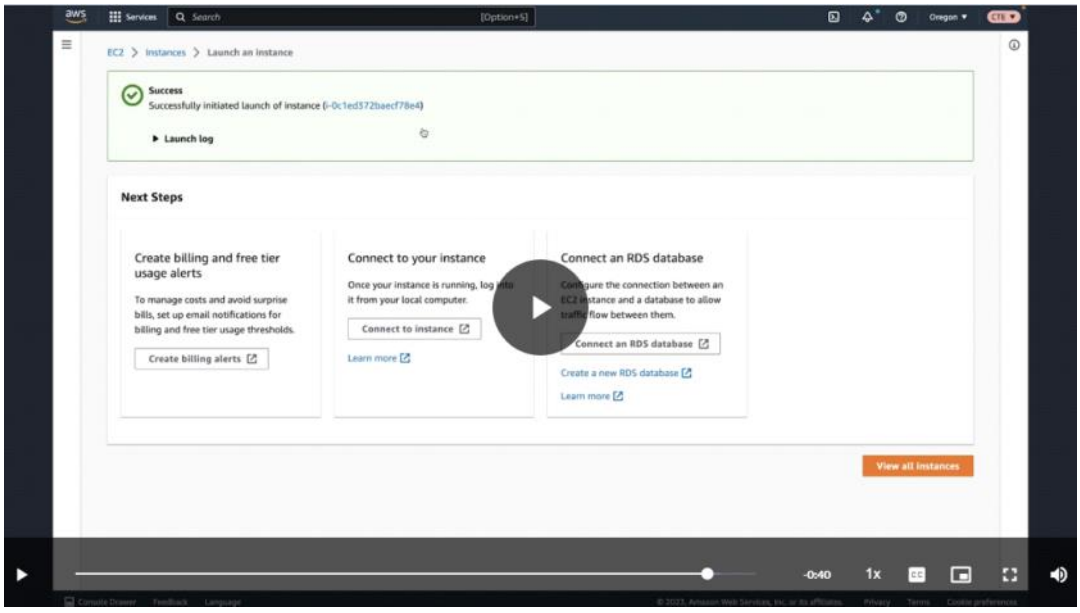
That's because the security group is tied to the VPC. So we need to create a new security group that will be associated with this new VPC, not the default VPC. So what we want to do here is click Create security group and then we will leave the default for the name and we will do the same thing that we did when we created our original security group, which is we want to allow both HTTP traffic on port 80 from the internet and adding a second rule, we want to allow HTTPS traffic on port 443 from anywhere.

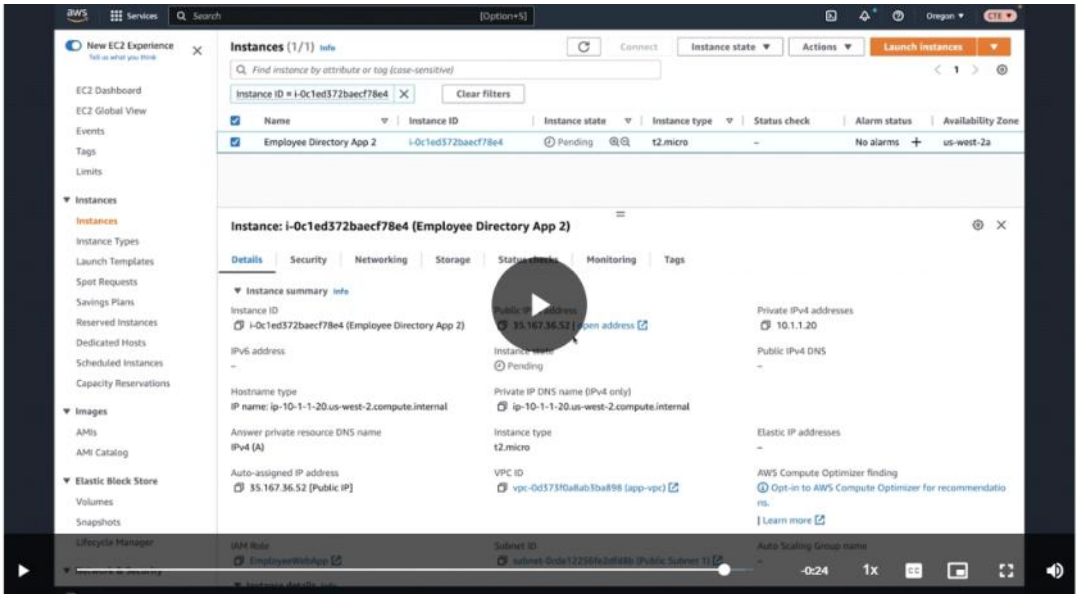


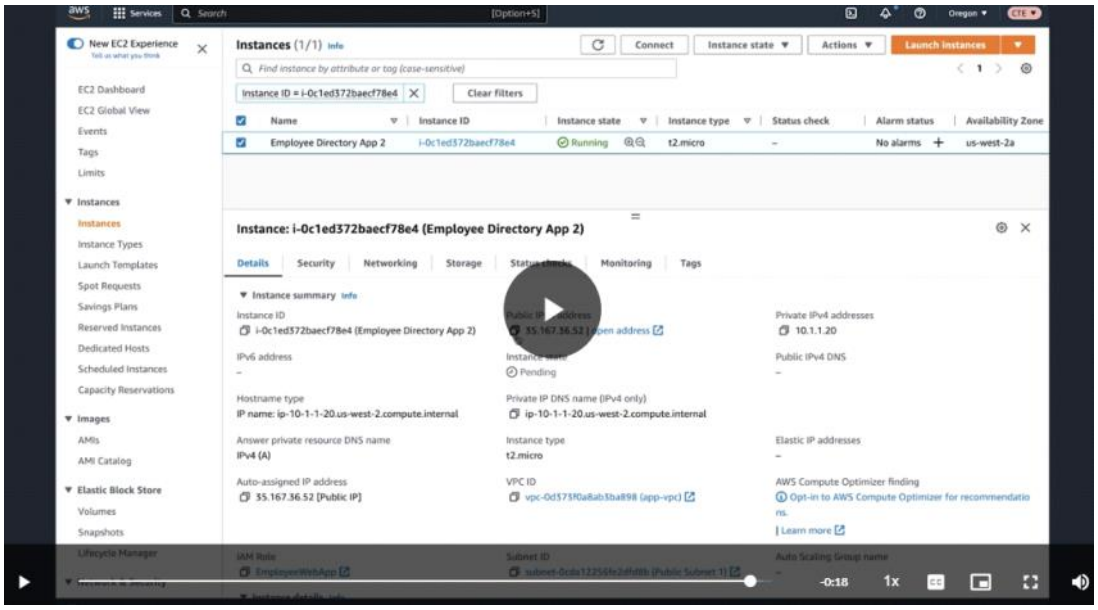
Now scrolling down, we can scroll down to the Advanced details and expand this here. We can see that the role has been prepopulated, so that's great. And then if we scroll down some more, let's take a look at that user data. We can see that this also is prepopulated.

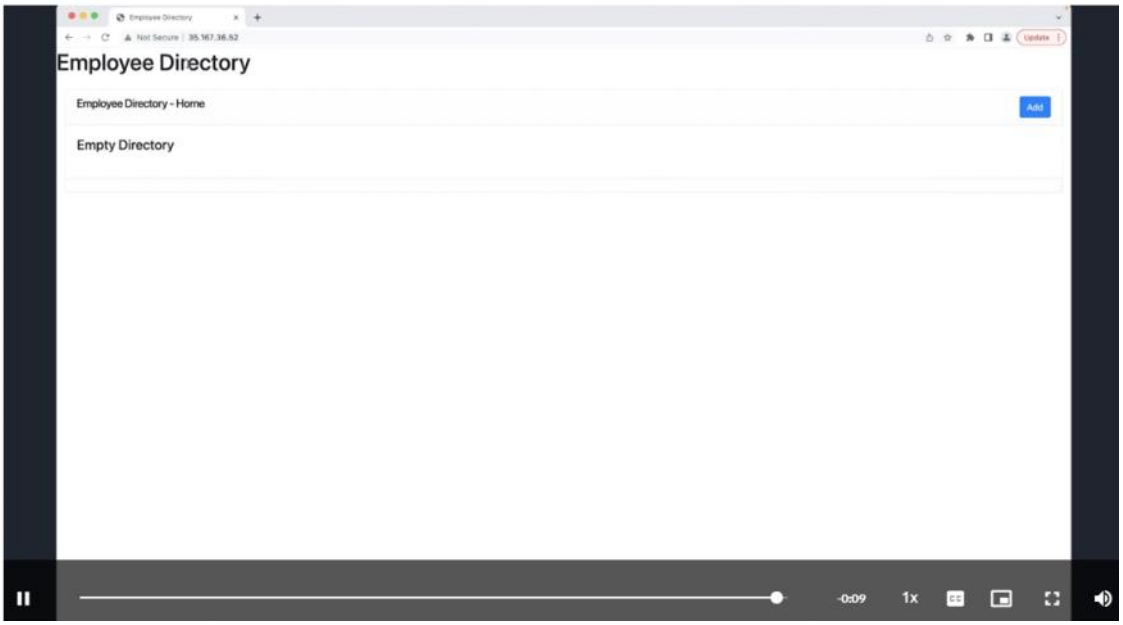


So now we can click Launch instance. We can click on the instance ID, see that this is in the pending state. And so now we will wait a few minutes and come back and try to access it through this public IP address. And if we can access it, that means that all of our network configurations were configured correctly. Okay, so now we are back and we can see that the instance is in the running state. If we copy this IP address and then I'm going to paste it into a new tab off screen, drag this tab over, we can see that we can now access the Employee Directory application at the IP address of the new instance that was launched into our new app-vpc.




Instances (1/1) Info
Instance: i-0c1ed372baecf78e4 (Employee Directory App 2)
Details
Security
Networking
Storage
Monitoring
Tags
Instance summary Info
Employee Directory App 2
i-0c1ed372baecf78e4
Pending
t2.micro
No alarms
us-west-2a
-0:24
1x


Instances (1/1) Info
Instance: i-0c1ed372baecf78e4 (Employee Directory App 2)
Details
Security
Networking
Storage
Monitoring
Tags
Instance summary Info
Employee Directory App 2
i-0c1ed372baecf78e4
Running
t2.micro
No alarms
us-west-2a
-0:18
1x

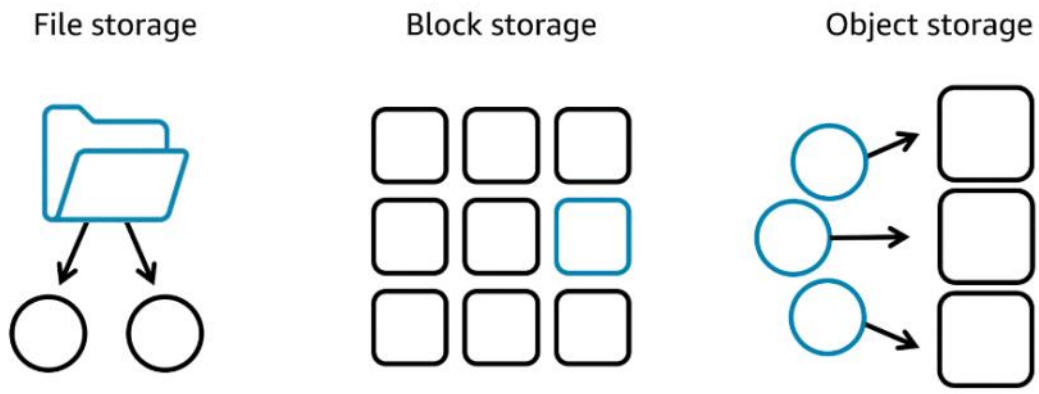

Employee Directory
Employee Directory - Home
Add
Empty Directory
-0:09
1x

Module-4 **Storage Types**

22 May 2024 16:06

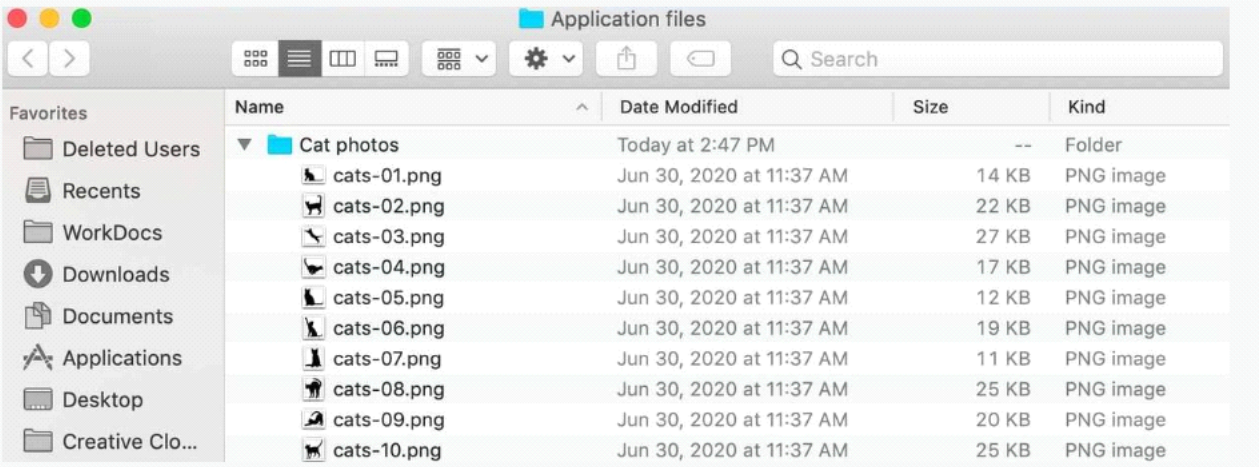
With cloud computing, you can create, delete, and modify storage solutions within a matter of minutes.

AWS storage services are grouped into three categories: file storage, block storage, and object storage. In file storage, data is stored as files in a hierarchy. In block storage, data is stored in fixed-size blocks. And in object storage, data is stored as objects in buckets.



File storage

You might be familiar with file storage if you have interacted with file storage systems like Windows File Explorer or Finder on macOS. Files are organized in a tree-like hierarchy that consist of folders and subfolders. For example, if you have hundreds of cat photos on your laptop, you might want to create a folder called Cat photos, and place the images inside that folder to organize them. Because you know that these images will be used in an application, you might want to place the Cat photos folder inside another folder called Application files.



Each file has metadata such as file name, file size, and the date the file was created. The file also has a path, for example, computer/Application_files/Cat_photos/cats-03.png. When you need to retrieve a file, your system can use the path to find it in the file hierarchy.

File storage is ideal when you require centralized access to files that must be easily shared and managed by multiple host computers. Typically, this storage is mounted onto multiple hosts, and requires file locking and integration with existing file system communication protocols.

Use cases for file storage

Web serving

Cloud file storage solutions follow common file-level protocols, file naming conventions, and permissions that developers are familiar with. Therefore, file storage can be integrated into web applications

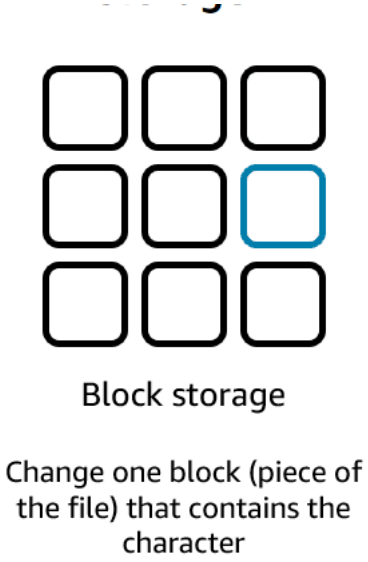
Analytics
Many analytics workloads interact with data through a file interface and rely on features such as file lock or writing to portions of a file. Cloud-based file storage supports common file-level protocols and has the ability to scale capacity and performance. Therefore, file storage can be conveniently integrated into analytics workflows.

Media and entertainment
Many businesses use a hybrid cloud deployment and need standardized access using file system protocols (NFS or SMB) or concurrent protocol access. Cloud file storage follows existing file system semantics. Therefore, storage of rich media content for processing and collaboration can be integrated for content production, digital supply chains, media streaming, broadcast playout, analytics, and archive.

Home directories
Businesses wanting to take advantage of the scalability and cost benefits of the cloud are extending access to home directories for many of their users. Cloud file storage systems adhere to common file-level protocols and standard permissions models. Therefore, customers can lift and shift applications that need this capability to the cloud.

Block storage

File storage treats files as a singular unit, but block storage splits files into fixed-size chunks of data called blocks that have their own addresses. Each block is an individual piece of data storage. Because each block is addressable, blocks can be retrieved efficiently. Think of block storage as a more direct route to access the data.



When data is requested, the addresses are used by the storage system to organize the blocks in the correct order to form a complete file to present back to the requestor. Besides the address, no additional metadata is associated with each block.

Changing one character in a 1-GB file with block storage

If you want to change one character in a file, you just change the block, or the piece of the file, that contains the character. This ease of access is why block storage solutions are fast and use less bandwidth.

Use cases for block storage

Because block storage is optimized for low-latency operations, it is a preferred storage choice for high-performance enterprise workloads and transactional, mission-critical, and I/O-intensive applications.

Transactional workloads

Organizations that process time-sensitive and mission-critical transactions store such workloads into a low-latency, high-capacity, and fault-tolerant database. Block storage allows developers to set up a robust, scalable, and highly efficient transactional database. Because each block is a self-contained unit, the database performs optimally, even when the stored data grows.

Containers

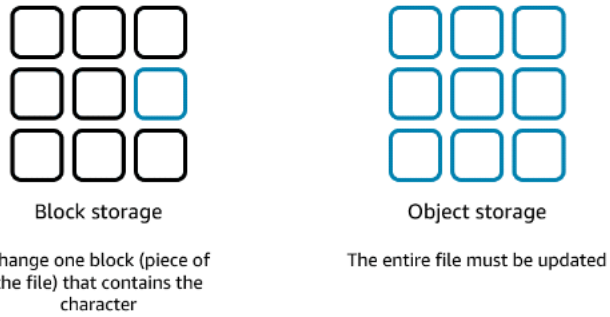
Developers use block storage to store containerized applications on the cloud. Containers are software packages that contain the application and its resource files for deployment in any computing environment. Like containers, block storage is equally flexible, scalable, and efficient. With block storage, developers can migrate the containers seamlessly between servers, locations, and operating environments.

Virtual machines

Block storage supports popular virtual machine (VM) hypervisors. Users can install the operating system, file system, and other computing resources on a block storage volume. They do so by formatting the block storage volume and turning it into a VM file system. So they can readily increase or decrease the virtual drive size and transfer the virtualized storage from one host to another.

Object storage

In object storage, files are stored as objects. Objects, much like files, are treated as a single, distinct unit of data when stored. However, unlike file storage, these objects are stored in a bucket using a flat structure, meaning there are no folders, directories, or complex hierarchies. Each object contains a unique identifier. This identifier, along with any additional metadata, is bundled with the data and stored.



Changing one character in a 1-GB file with object storage

Changing just one character in an object is more difficult than with block storage. When you want to change one character in an object, the entire object must be updated.

Use cases for object storage

With object storage, you can store almost any type of data, and there is no limit to the number of objects stored, which makes it readily scalable. Object storage is generally useful when storing large or unstructured data sets.

To learn more, expand each of the following three categories.

Data archiving

Cloud object storage is excellent for long-term data retention. You can cost-effectively archive large amounts of rich media content and retain mandated regulatory data for extended periods of time. You can also use cloud object storage to replace on-premises tape and disk archive infrastructure. This storage solution provides enhanced data durability, immediate retrieval times, better security and compliance, and greater data accessibility.

Backup and recovery

You can configure object storage systems to replicate content so that if a physical device fails, duplicate object storage devices become available. This ensures that your systems and applications continue to run without interruption. You can also replicate data across multiple data centers and geographical regions.

Rich media

With object storage, you can accelerate applications and reduce the cost of storing rich media files such as videos, digital images, and music. By using storage classes and replication features, you can create cost-effective, globally replicated architecture to deliver media to distributed users.

Relating back to traditional storage systems

If you have worked with on-premises storage, you might already be familiar with block, file, and object storage. Consider the following technologies and how they relate to systems that you might have seen before:

Block storage in the cloud is analogous to direct-attached storage (DAS) or a storage area network (SAN).
File storage systems are often supported with a network-attached storage (NAS) server.

Adding storage in a traditional data center is a rigid process—the storage solutions must be purchased, installed, and configured. With cloud computing, the process is more flexible. You can create, delete, and modify storage solutions within a matter of minutes.

Block storage and object storage are two distinct types of data storage technologies used in computing environments, each with its own characteristics, use cases, and benefits.

Block Storage

Characteristics:

1. Data Structure: Data is stored in fixed-sized blocks. Each block is identified by a unique address and can be managed independently.
2. File Systems: Typically requires a file system (e.g., NTFS, ext4) to manage these blocks.
3. Performance: Generally offers high performance and low latency, making it suitable for applications that require fast I/O operations.
4. Use Cases: Ideal for databases, virtual machines, and high-performance applications where fast read/write operations are crucial.
5. Management: Requires more administrative overhead for managing the storage, ensuring data integrity, and optimizing performance.

Examples:

- SAN (Storage Area Network)
- DAS (Direct Attached Storage)
- Cloud block storage services (e.g., Amazon EBS, Google Persistent Disk)

Object Storage

Characteristics:

6. Data Structure: Data is stored as objects, which include the data itself, metadata, and a unique identifier. Objects are stored in a flat address space.
7. Scalability: Designed for massive scalability, capable of handling large amounts of unstructured data.
8. Performance: Optimized for large-scale, sequential data access rather than random I/O operations. May have higher latency compared to block storage.
9. Use Cases: Suitable for storing large amounts of unstructured data such as backups, media files, logs, and big data. It is also ideal for cloud-native applications.

10. Management: Easier to manage in terms of data durability, replication, and availability. The storage system manages metadata and data integrity.
    Examples:

- Amazon S3 (Simple Storage Service)
- Google Cloud Storage
- Microsoft Azure Blob Storage

Comparison Summary

Feature	Block Storage	Object Storage
Data Structure	Blocks with fixed sizes	Objects with metadata and unique identifiers
Performance	Low latency, high performance	High latency, optimized for large-scale access
Scalability	Limited by file system and architecture	Massively scalable
Use Cases	Databases, virtual machines, high-performance apps	Backups, media files, big data, cloud apps
Management	Requires file system and administrative overhead	Easier to manage with built-in durability
Examples	SAN, DAS, Amazon EBS	Amazon S3, Google Cloud Storage

Both block and object storage have their place in modern IT environments, and the choice between them depends on the specific needs of the application and workload. Block storage is preferred for performance-critical applications, while object storage is ideal for handling vast amounts of unstructured data with ease of management and scalability.

From <https://chatgpt.com/c/576dc0b7-ff1a-4995-9de1-0611782e058c>

Module-4 File Storage with Amazon EFS and Amazon FSx

22 May 2024 17:13

Use a range of cloud storage solutions to readily move to managed file storage without changes to your applications or workflows.

Amazon Elastic File System (Amazon EFS)

Amazon Elastic File System (Amazon EFS) is a set-and-forget file system that automatically grows and shrinks as you add and remove files. There is no need for provisioning or managing storage capacity and performance. Amazon EFS can be used with AWS compute services and on-premises resources. You can connect tens, hundreds, and even thousands of compute instances to an Amazon EFS file system at the same time, and Amazon EFS can provide consistent performance to each compute instance.

With the Amazon EFS simple web interface, you can create and configure file systems quickly without any minimum fee or setup cost. You pay only for the storage used and you can choose from a range of storage classes designed to fit your use case.

Standard storage classes	One zone storage classes
EFS Standard and EFS Standard-Infrequent Access (Standard-IA) offer Multi-AZ resilience and the highest levels of durability and availability.	EFS One Zone and EFS One Zone-Infrequent Access (EFS One Zone-IA) provide additional savings by saving your data in a single availability zone.

Amazon FSx

Amazon FSx is a fully managed service that offers reliability, security, scalability, and a broad set of capabilities that make it convenient and cost effective to launch, run, and scale high-performance file systems in the cloud. With Amazon FSx, you can choose between four widely used file systems: Lustre, NetApp ONTAP, OpenZFS, and Windows File Server. You can choose based on your familiarity with a file system or based on your workload requirements for feature sets, performance profiles, and data management capabilities.

Amazon EFS (Elastic File System) and Amazon FSx are both managed file storage services provided by AWS, but they cater to different use cases and have distinct characteristics.

Amazon EFS (Elastic File System)

Characteristics:
Data Structure: Provides a shared, scalable, and fully managed NFS (Network File System) for use with AWS Cloud services and on-premises resources.
Scalability: Automatically scales to accommodate petabytes of data, growing and shrinking as files are added or removed.
Performance Modes: Offers two performance modes: General Purpose (for latency-sensitive use cases) and Max I/O (for highly parallelized applications).
Use Cases: Ideal for web serving, content management, home directories, and data-intensive applications that require scalable file storage.

Management: Managed service, so AWS handles the infrastructure, providing high availability and durability.
Access: Multiple instances can access the file system concurrently, making it suitable for workloads that require shared access.
Examples:

- Web serving and content management
- Media processing workflows
- Big data and analytics workloads

Amazon FSx

Amazon FSx provides managed file systems that are optimized for specific use cases. There are different FSx offerings, each tailored for a different type of file system:

1. Amazon FSx for Windows File Server:

- Characteristics: Fully managed native Microsoft Windows file system that provides Windows-native compatibility.
- Use Cases: Ideal for Windows-based applications such as SharePoint, SQL Server, and custom .NET applications.
- Features: Supports SMB (Server Message Block) protocol, Active Directory integration, and Windows ACLs (Access Control Lists).

2. Amazon FSx for Lustre:

- Characteristics: High-performance file system for fast processing of workloads like machine learning, high-performance computing (HPC), and media processing.
- Use Cases: Suitable for applications that require sub-millisecond latencies and high-throughput performance.
- Features: Integration with Amazon S3, providing fast processing of data sets stored in S3.

3. Amazon FSx for NetApp ONTAP:

- Characteristics: Provides the features and capabilities of NetApp's ONTAP file system.
- Use Cases: Suitable for complex file system needs such as data protection, multiprotocol access, and advanced data management.
- Features: Supports NFS, SMB, and iSCSI protocols, as well as data deduplication, compression, and tiering.

Comparison Summary

Feature	Amazon EFS	Amazon FSx (Windows, Lustre, NetApp ONTAP)
Data Structure	NFS-based shared file system	Windows File System, Lustre File System, or NetApp ONTAP
Scalability	Automatically scales with usage	Scales according to file system type and configuration
Performance Modes	General Purpose, Max I/O	High-performance modes (varies by FSx type)
Use Cases	Web serving, content management, home directories	Windows-based apps, HPC, ML, media processing, complex file systems
Management	Fully managed by AWS	Fully managed by AWS
Access	Concurrent access from multiple instances	Varies by FSx type (e.g., SMB for Windows, NFS for Lustre)
Examples	Web servers, media workflows, big data	Windows applications, HPC, data management, multiprotocol access

Choosing Between EFS and FSx

- Choose Amazon EFS if you need a scalable, shared file storage system that supports NFS and can be accessed concurrently by multiple instances. It is well-suited for general-purpose workloads and applications that require scalable and managed file storage.
- Choose Amazon FSx if you need a specialized file system:
- FSx for Windows File Server for Windows-based applications requiring SMB protocol

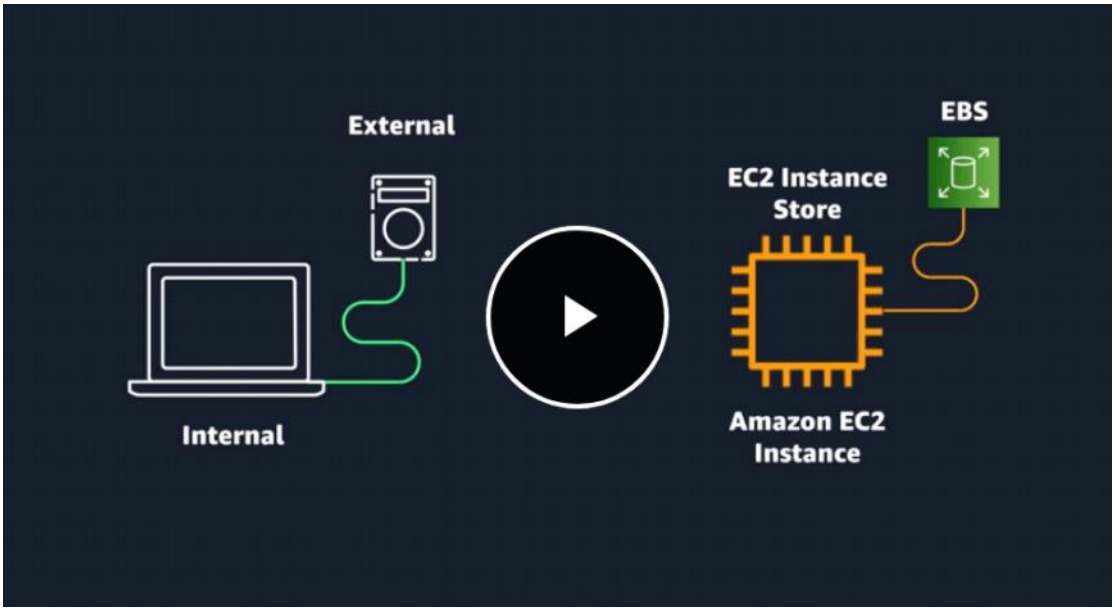
support and Windows compatibility.

- FSx for Lustre for high-performance computing applications requiring fast data processing and integration with S3.
- FSx for NetApp ONTAP for advanced data management capabilities and multiprotocol access.
  Each service is designed to meet specific needs and optimize performance for different types of workloads.

Module-4 Block Storage with Amazon EC2 Instance Store and Amazon EBS

22 May 2024 17:23

When you launch an EC2 instance, you're going to need some kind of block storage to go with it. This block storage can be used as a boot volume for your operating system or a separate data volume. For example, think about your laptop. With a laptop, you store your data in drives, and those drives are either built in internally to your laptop or connected externally. EC2 instances have the same options as far as block storage goes. The internal storage is called instance store and the external connected storage is called Amazon Elastic Block Store, or Amazon EBS. Let's talk about instance store first.



Instance store is a form of directly attached storage, which means the underlying physical server has at least one storage unit directly attached to it. This direct attachment is also the main advantage of using this form of storage. Because it's so close to the physical server, it can be very fast and respond very quickly.

But while it can be very fast, there is also one big downside. With instance store being directly attached to an EC2 instance, its lifecycle is tied to that of the instance. That means if you stop or terminate an instance, all data in the instance store is gone. It can no longer be used or accessed. Naturally, there are many use cases where you want the ability to keep data, even if you shut an EC2 instance down.

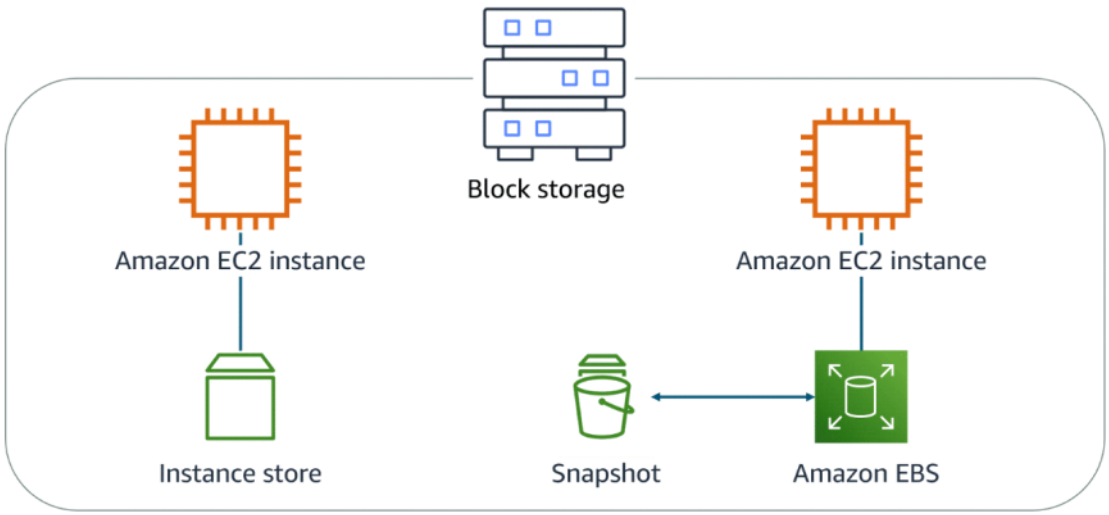
This is where EBS volumes come in. These volumes, as the name implies, are drives of a user configured size that are separate from an EC2 instance. The drives are simply network attached storage for your instances. You can think of it as similar to how you might attach an external drive to your laptop. You can attach multiple EBS volumes to one EC2 instance, and then you can configure how to use that storage on the OS of the EC2 instance. When I connect that EBS volume to my instance, my instance now has a direct communication line to the data in that volume. Nobody else can directly talk to that volume so that it maintains secure communication. You need an EC2 instance to access data on an EBS volume. If I decided I want to use that EBS volume with a different instance, that's no problem. We can stop the instance, detach the volume, and then attach it to another instance in the same AZ. Much like you can unplug your drive from a laptop, and plug it into another one. Or depending on the instance type and EBS volume we're using, we may be able to attach it to multiple instances at the same time, which is called EBS Multi-Attach. And perhaps the most important similarity is that an EBS volume is separate from your instance. Just like an external drive is separate from your laptop. That means if an accident happens, and the instance goes down, you still have your data on your EBS volume. This is what we refer to as persistent storage. You can stop or terminate your instance, and your EBS volume can still exist with your data on it.

EBS is often the right storage type for workloads that require persistence of data. However, the question typically comes down to which EBS volume type do I use? That's right. There are many different types of volumes, but they've divided into two main volume types: SSD backed volumes and HDD backed volumes. In the readings, you will learn more about these two options. The last thing we will need to talk about here is backing up data. Things fail, errors happen, so you need to back up your data, even in AWS. The way you backup EBS volumes is by taking what we call snapshots. EBS snapshots are incremental backups that are stored redundantly. The idea here is that if something goes wrong, you can create new volumes from your snapshots and restore your data to a safe state.

Amazon EC2 instance store

Amazon Elastic Compute Cloud (Amazon EC2) instance store provides temporary block-level storage for an instance. This storage is located on disks that are physically attached to the host computer. This ties the lifecycle of the data to the lifecycle of the EC2 instance. If you delete the instance, the instance store is also deleted. Because of this, instance store is considered ephemeral storage. Read more about it in the Amazon EC2 documentation found in the resources section at the end of this lesson.

Instance store is ideal if you host applications that replicate data to other EC2 instances, such as Hadoop clusters. For these cluster-based workloads, having the speed of locally attached volumes and the resiliency of replicated data helps you achieve data distribution at high performance. It's also ideal for temporary storage of information that changes frequently, such as buffers, caches, scratch data, and other temporary content.



Amazon EBS

As the name implies, Amazon Elastic Block Store (Amazon EBS) is block-level storage that you can attach to an Amazon EC2 instance. You can compare this to how you much attach an external drive to your laptop. This attachable storage is called an EBS volume. EBS volumes act similarly to external drives in more than one way.

Detachable: You can detach an EBS volume from one EC2 instance and attach it to another EC2 instance in the same Availability Zone to access the data on it.

Distinct: The external drive is separate from the computer. That means that if an accident occurs and the computer goes down, you still have your data on your external drive. The same is true for EBS volumes.

Size-limited: You're limited to the size of the external drive, because it has a fixed limit to how scalable it can be. For example, you might have a 2 TB external drive, which means you can only have 2 TB of content on it. This also relates to Amazon EBS, because a volume also has a max limitation of how much content you can store on it.

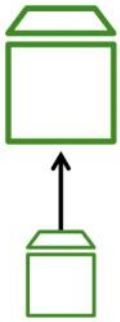
1-to-1 connection: Most EBS volumes can only be connected with one computer at a time. Most EBS volumes have a one-to-one relationship with EC2 instances, so they cannot be shared by or attached to multiple instances at one time.

Scaling Amazon EBS volumes

You can scale EBS volumes in two ways. To learn about a category, choose the appropriate tab.

INCREASE VOLUME SIZE | ATTACH MULTIPLE VOLUMES

Increase the volume size only if it doesn't increase above the maximum size limit. Depending on the volume selected, Amazon EBS currently supports a maximum volume size of 64 tebibytes (TiB). For example, if you provision a 5-TiB io2 Block Express volume, you can choose to increase the size of your volume until you get to 64 TiB.



INCREASE VOLUME SIZE | ATTACH MULTIPLE VOLUMES

Attach multiple volumes to a single EC2 instance. Amazon EC2 has a one-to-many relationship with EBS volumes. You can add these additional volumes during or after EC2 instance creation to provide more storage capacity for your hosts.

Amazon EBS use cases

Amazon EBS is useful when you must retrieve data quickly and have data persist long term. Volumes are commonly used in the following scenarios.

To learn more, expand each of the following four categories.

Operating systems
Boot and root volumes can be used to store an operating system. The root device for an instance launched from an Amazon Machine Image (AMI) is typically an EBS volume. These are commonly referred to as EBS-backed AMIs.

Databases
As a storage layer for databases running on Amazon EC2 that will scale with your performance needs and provide consistent and low-latency performance.

Enterprise applications
Amazon EBS provides high availability and high durability block storage to run business-critical applications.

Big data analytics engines
Amazon EBS offers data persistence, dynamic performance adjustments, and the ability to detach and reattach volumes, so you can resize clusters for big data analytics.

EBS volume types

EBS volumes are organized into two main categories: solid-state drives (SSDs) and hard-disk drives (HDDs). SSDs are used for transactional workloads with frequent read/write operations with small I/O size. HDDs are used for large streaming workloads that need high throughput performance. AWS offers two types of each.

The following two tables can help you decide which EBS volume is the right option for your workload.

To learn more, expand the following two categories.

SSD volumes

**Note: For users with screen readers, use table mode to read the table.

	General Purpose SSD volumes		Provisioned IOPS SSD volumes		
Volume type	gp3	gp2	io2 Block Express	io2	io1
Description	Provides a balance of price and performance for a wide variety of transactional workloads		Provides high-performance SSD designed for latency-sensitive transactional workloads		
Volume size	1 GiB–16 TiB		4 GiB–64 TiB	4GiB–16 TiB	
Max IOPS per volume	16,000		256,000	64,000	
Max throughput per volume	1,000 MiB/s	250 MiB/s	4,000 MiB/s	1,000 MiB/s	
Amazon EBS Multi-attach	Not supported		Supported		

HDD volumes

**Note: For users with screen readers, use table mode to read the table.

	Throughput Optimized HDD volumes	Cold HDD volumes
Volume type	st1	sc1
Description	A low-cost HDD designed for frequently accessed, throughput-intensive workloads	The lowest cost HDD designed for less frequently accessed workloads
Volume size	125 GiB–16 TiB	
Max IOPS per volume	500	250
Max throughput per volume	500 MiB/s	250 MiB/s
Amazon EBS Multi-attach	Not supported	

Use cases of EBS

1. **High Performance**

- Consistent and Low Latency: EBS delivers consistent, low-latency performance required to run a wide variety of workloads.
- Performance Levels: Offers multiple volume types designed to meet the performance needs of various workloads, from small development and test environments to mission-critical applications.

2. **Scalability**

- Elastic Storage: Allows you to increase or decrease your storage capacity within minutes without disrupting your applications.
- Volume Types: Provides a range of volume types that allow you to balance price and performance for your workload needs. These include General Purpose SSD (gp3, gp2), Provisioned IOPS SSD (io2, io1), Throughput Optimized HDD (st1), and Cold HDD (sc1).

3. **Durability and Availability**

- High Availability: Automatically replicates within its Availability Zone to protect against hardware failure, offering high availability.
- Data Durability: Designed for 99.999% availability and data durability, ensuring your data is safe and secure.

4. **Snapshots and Backup**

- Snapshots: Supports incremental snapshots to Amazon S3 for backup and disaster recovery. Snapshots capture only the blocks that have changed since your last snapshot, reducing time and storage costs.
- Automated Backups: You can set up automated snapshots and backups to ensure data protection and simplify recovery processes.

5. **Security**

- Data Encryption: Provides seamless encryption of data at rest and data in transit, using AWS Key Management Service (KMS) for key management.
- Access Control: Integrates with AWS Identity and Access Management (IAM) to control access to EBS volumes and snapshots, ensuring secure access.

6. **Flexibility**

- Volume Modifications: Allows you to modify volume type, IOPS, or size without downtime or performance degradation, providing flexibility to adapt to changing application requirements.
- Attach/Detach Volumes: You can attach and detach EBS volumes to and from EC2 instances, enabling dynamic scaling of storage as needed.

7. **Cost-Effectiveness**

- Optimized Pricing: Offers cost-effective storage solutions with a range of volume types, allowing you to choose the right balance of cost and performance.
- Pay-as-You-Go: Charges based on the provisioned storage capacity, making it a cost-effective solution for varying workloads and storage requirements.

8. **Integration with AWS Services**

- Seamless Integration: Integrates seamlessly with other AWS services, such as Amazon EC2, AWS Lambda, and Amazon RDS, enhancing the overall functionality and flexibility of your applications.
- Automation and Management: Supports automation and management tools like AWS CloudFormation, AWS Auto Scaling, and AWS CloudWatch for monitoring and managing your storage resources efficiently.

9. **Reliability**

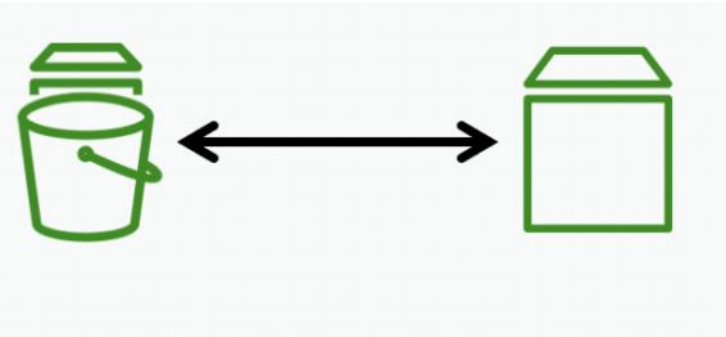
- Service Level Agreement (SLA): Offers a strong SLA that ensures service reliability and uptime, giving you confidence in the stability and performance of your storage infrastructure.

10. **Data Transfer Management**

- Data Transfer Acceleration: Utilizes Amazon EC2's enhanced networking capabilities for faster data transfer and reduced latency, optimizing performance for high-traffic applications.

Amazon EBS snapshots

Errors happen. One error is not backing up data and then inevitably losing it. To prevent this from happening to you, always back up your data, even in AWS. Because your EBS volumes consist of the data from your EC2 instance, you should make backups of these volumes, called snapshots.



EBS snapshots are incremental backups that only save the blocks on the volume that have changed after your most recent snapshot. For example, if you have 10 GB of data on a volume and only 2 GB of data have been modified since your last snapshot, only the 2 GB that have been changed are written to Amazon S3.

When you take a snapshot of any of your EBS volumes, the backups are stored redundantly in multiple Availability Zones using Amazon S3. This aspect of storing the backup in Amazon S3 is handled by AWS, so you won't need to interact with Amazon S3 to work with your EBS snapshots. You manage them in the Amazon EBS console, which is part of the Amazon EC2 console.

EBS snapshots can be used to create multiple new volumes, whether they're in the same Availability Zone or a different one. When you create a new volume from a snapshot, it's an exact copy of the original volume at the time the snapshot was taken.

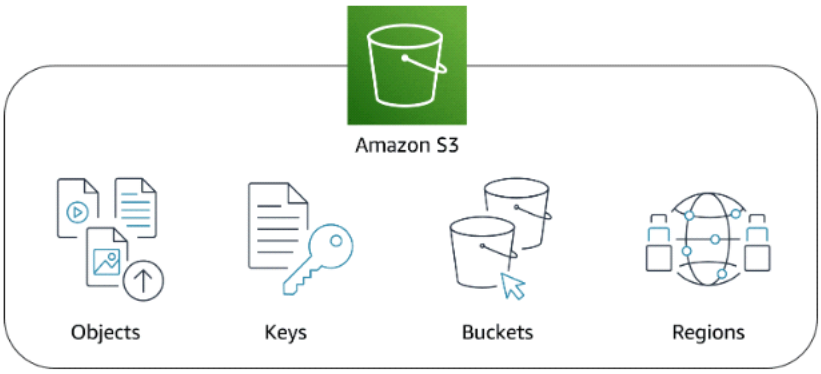
Module-4 **Object Storage with Amazon S3**

24 May 2024 15:52

Object storage is built for the cloud and delivers virtually unlimited scalability, high durability, and cost effectiveness.

Amazon S3

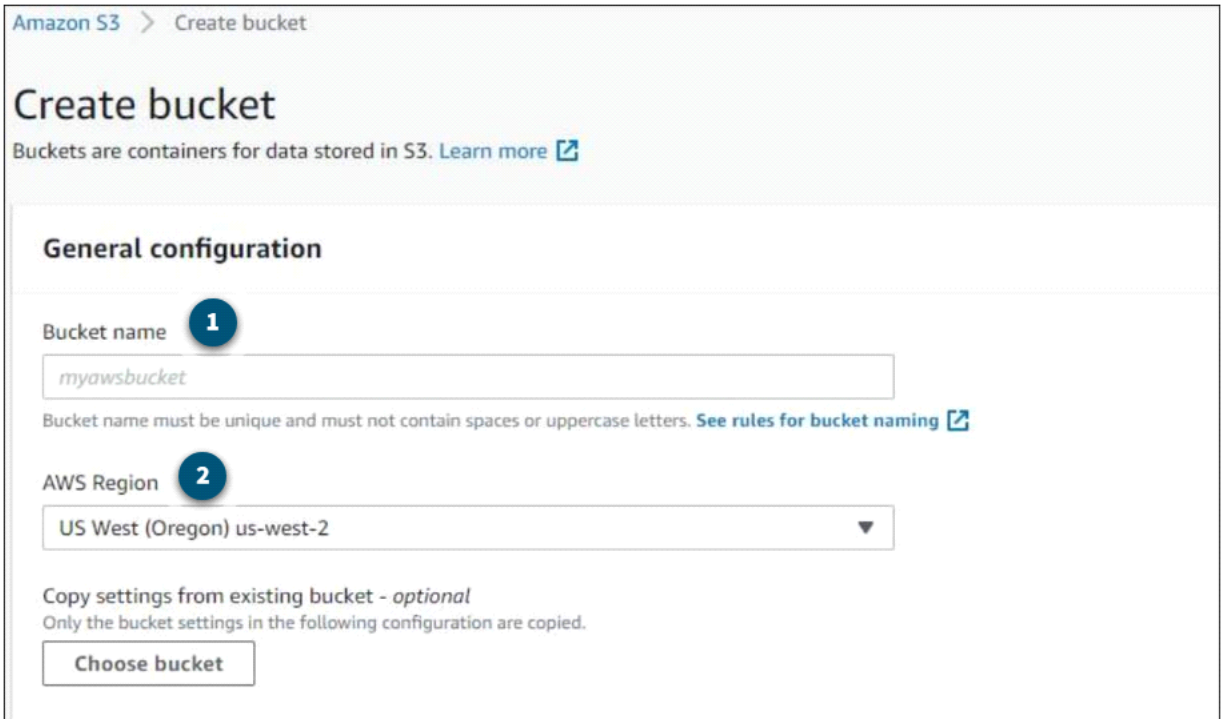
Unlike Amazon EBS, Amazon Simple Storage Service (Amazon S3) is a standalone storage solution that isn't tied to compute. With Amazon S3, you can retrieve your data from anywhere on the web. If you have used an online storage service to back up the data from your local machine, you most likely have used a service similar to Amazon S3. The big difference between those online storage services and Amazon S3 is the storage type.



Amazon S3 is an object storage service. Object storage stores data in a flat structure. An object is a file combined with metadata. You can store as many of these objects as you want. All the characteristics of object storage are also characteristics of Amazon S3.

In Amazon S3, you store your objects in containers called buckets. You can't upload an object, not even a single photo, to Amazon S3 without creating a bucket first. When you store an object in a bucket, the combination of a bucket name, key, and version ID uniquely identifies the object.

When you create a bucket, you specify, at the very minimum, two details: the bucket name and the AWS Region that you want the bucket to reside in.



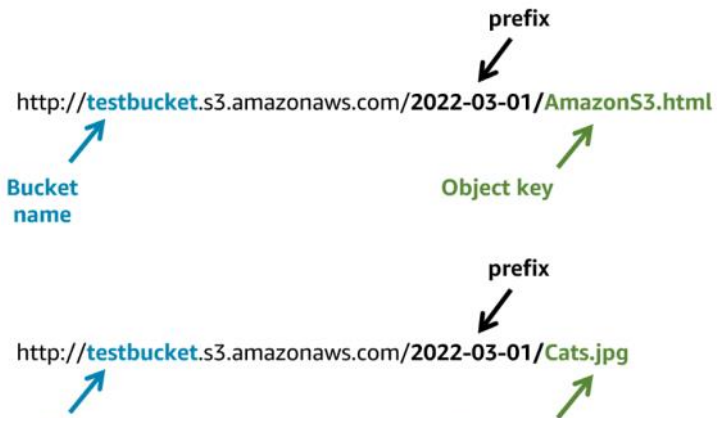
Amazon S3 bucket names

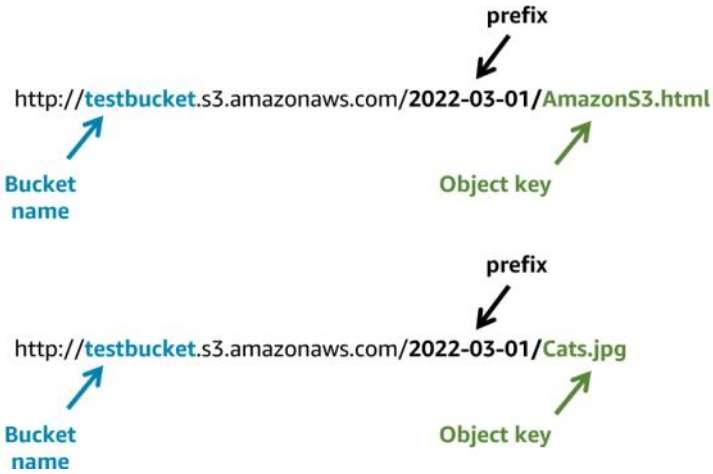
Amazon S3 supports global buckets. Therefore, each bucket name must be unique across all AWS accounts in all AWS Regions within a partition. A partition is a grouping of Regions, of which AWS currently has three: Standard Regions, China Regions, and AWS GovCloud (US). When naming a bucket, choose a name that is relevant to you or your business. For example, you should avoid using AWS or Amazon in your bucket name.

The following are some examples of the rules that apply for naming buckets in Amazon S3. For a full list of rules, see the link in the resources section.

- Bucket names must be between 3 (min) and 63 (max) characters long.
- Bucket names can consist only of lowercase letters, numbers, dots (.), and hyphens (-).
- Bucket names must begin and end with a letter or number.
- Buckets must not be formatted as an IP address.

A bucket name cannot be used by another AWS account in the same partition until the bucket is deleted.
If your application automatically creates buckets, choose a bucket naming scheme that is unlikely to cause naming conflicts and will choose a different bucket name, should one not be available.





Amazon S3 use cases

Amazon S3 is a widely used storage service, with far more use cases than could fit on one screen. To learn more, expand each of the following six categories.

Backup and storage
Amazon S3 is a natural place to back up files because it is highly redundant. As mentioned in the last lesson, AWS stores your EBS snapshots in Amazon S3 to take advantage of its high availability.

Media hosting
Because you can store unlimited objects, and each individual object can be up to 5 TB, Amazon S3 is an ideal location to host video, photo, and music uploads.

Software delivery
You can use Amazon S3 to host your software applications that customers can download.

Data lakes
Amazon S3 is an optimal foundation for a data lake because of its virtually unlimited scalability. You can increase storage from gigabytes to petabytes of content, paying only for what you use.

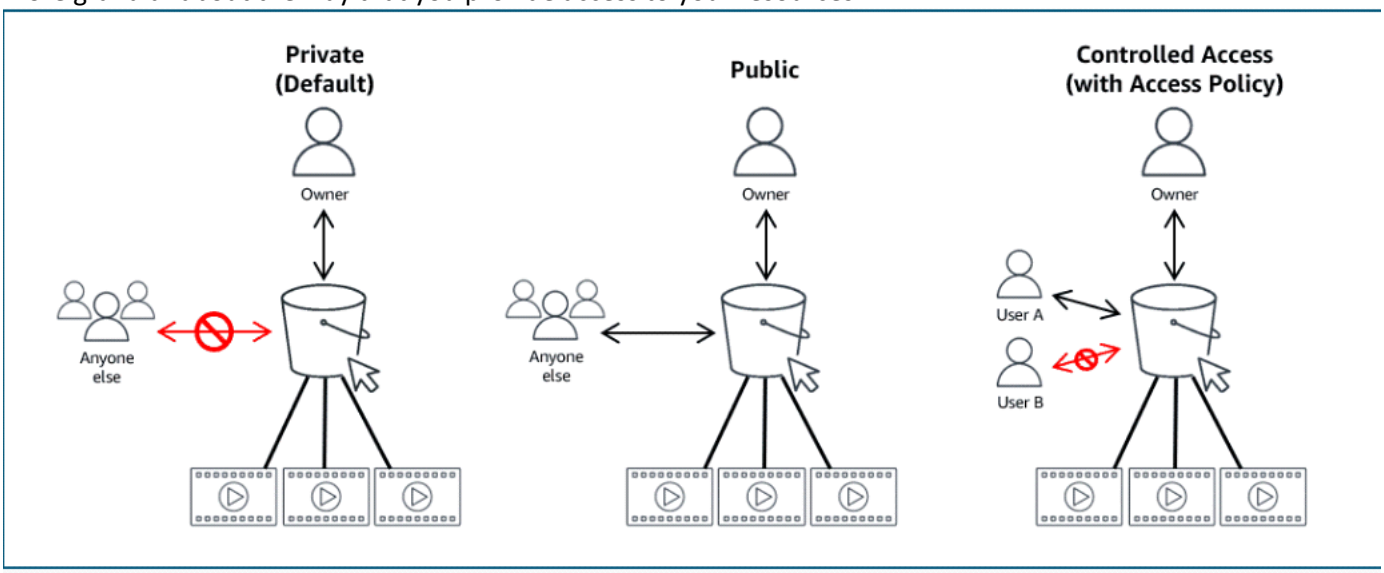
Static websites
You can configure your S3 bucket to host a static website of HTML, CSS, and client-side scripts.

Static content
Because of the limitless scaling, the support for large files, and the fact that you can access any object over the web at any time, Amazon S3 is the perfect place to store static content.

Security in Amazon S3

Everything in Amazon S3 is private by default. This means that all Amazon S3 resources, such as buckets and objects, can only be viewed by the user or AWS account that created that resource. Amazon S3 resources are all private and protected to begin with.

If you decide that you want everyone on the internet to see your photos, you can choose to make your buckets and objects public. A public resource means that everyone on the internet can see it. Most of the time, you don't want your permissions to be all or nothing. Typically, you want to be more granular about the way that you provide access to your resources.



To be more specific about who can do what with your Amazon S3 resources, Amazon S3 provides several security management features: IAM policies, S3 bucket policies, and encryption to develop and implement your own security policies.

Amazon S3 and IAM policies

Previously, you learned about creating and using AWS Identity and Access Management (IAM) policies. Now you can apply that knowledge to Amazon S3. When IAM policies are attached to your resources (buckets and objects) or IAM users, groups, and roles, the policies define which actions they can perform. Access policies that you attach to your resources are referred to as resource-based policies and access policies attached to users in your account are called user policies.

You should use IAM policies for private buckets in the following two scenarios:

1. You have many buckets with different permission requirements. Instead of defining many different S3 bucket policies, you can use IAM policies.
2. You want all policies to be in a centralized location. By using IAM policies, you can manage all policy information in one location.

Amazon S3 bucket policies

Like IAM policies, S3 bucket policies are defined in a JSON format. Unlike IAM policies, which are attached to resources and users, S3 bucket policies can only be attached to S3 buckets. The policy that is placed on the bucket applies to every object in that bucket. S3 bucket policies specify what actions are allowed or denied on the bucket.

You should use S3 bucket policies in the following scenarios:

1. You need a simple way to do cross-account access to Amazon S3, without using IAM roles.
2. Your IAM policies bump up against the defined size limit. S3 bucket policies have a larger size limit.
3. For examples of bucket policies, see the Bucket Policy Examples(opens in a new tab) link here or in the resources section.

Amazon S3 encryption

Amazon S3 reinforces encryption in transit (as it travels to and from Amazon S3) and at rest. To protect data, Amazon S3 automatically encrypts all objects on upload and applies server-side encryption with S3-managed keys as the base level of encryption for every bucket in Amazon S3 at no additional cost.

Amazon S3 storage classes

When you upload an object to Amazon S3 and you don't specify the storage class, you upload it to the default storage class, often referred to as standard storage. In previous lessons, you learned about the default Amazon S3 standard storage class.

Amazon S3 storage classes let you change your storage tier when your data characteristics change. For example, if you are accessing your old photos infrequently, you might want to change the storage class for the photos to save costs.

Storage Class	Description
S3 Standard	This is considered general-purpose storage for cloud applications, dynamic websites, content distribution, mobile and gaming applications, and big data analytics.
S3 Intelligent-Tiering	This tier is useful if your data has unknown or changing access patters. S3 Intelligent-Tiering stores objects in three tiers: a frequent access tier, an infrequent access tier, and an archive instance access tier. Amazon S3 monitors access patterns of your data and automatically moves your data to the most cost-effective storage tier based on frequency of access.
S3 Standard-Infrequent Access (S3 Standard-IA)	This tier is for data that is accessed less frequently but requires rapid access when needed. S3 Standard-IA offers the high durability, high throughput, and low latency of S3 Standard, with a low per-GB storage price and per-GB retrieval fee. This storage tier is ideal if you want to store long-term backups, disaster recovery files, and so on.
S3 One Zone-Infrequent Access (S3 One Zone-IA)	Unlike other S3 storage classes that store data in a minimum of three Availability Zones, S3 One Zone-IA stores data in a single Availability Zone, which makes it less expensive than S3 Standard-IA. S3 One Zone-IA is ideal for customers who want a lower-cost option for infrequently accessed data, but do not require the availability and resilience of S3 Standard or S3 Standard-IA. It's a good choice for storing secondary backup copies of on-premises data or easily recreatable data.
S3 Glacier Instant Retrieval	Use S3 Glacier Instant Retrieval for archiving data that is rarely accessed and requires millisecond retrieval. Data stored in this storage class offers a cost savings of up to 68 percent compared to the S3 Standard-IA storage class, with the same latency and throughput performance.
S3 Glacier Flexible Retrieval	S3 Glacier Flexible Retrieval offers low-cost storage for archived data that is accessed 1–2 times per year. With S3 Glacier Flexible Retrieval, your data can be accessed in as little as 1–5 minutes using an

	expedited retrieval. You can also request free bulk retrievals in up to 5–12 hours. It is an ideal solution for backup, disaster recovery, offsite data storage needs, and for when some data occasionally must be retrieved in minutes.
S3 Glacier Deep Archive	S3 Glacier Deep Archive is the lowest-cost Amazon S3 storage class. It supports long-term retention and digital preservation for data that might be accessed once or twice a year. Data stored in the S3 Glacier Deep Archive storage class has a default retrieval time of 12 hours. It is designed for customers that retain data sets for 7–10 years or longer, to meet regulatory compliance requirements. Examples include those in highly regulated industries, such as the financial services, healthcare, and public sectors.
S3 on Outposts	Amazon S3 on Outposts delivers object storage to your on-premises AWS Outposts environment using S3 API's and features. For workloads that require satisfying local data residency requirements or need to keep data close to on premises applications for performance reasons, the S3 Outposts storage class is the ideal option.

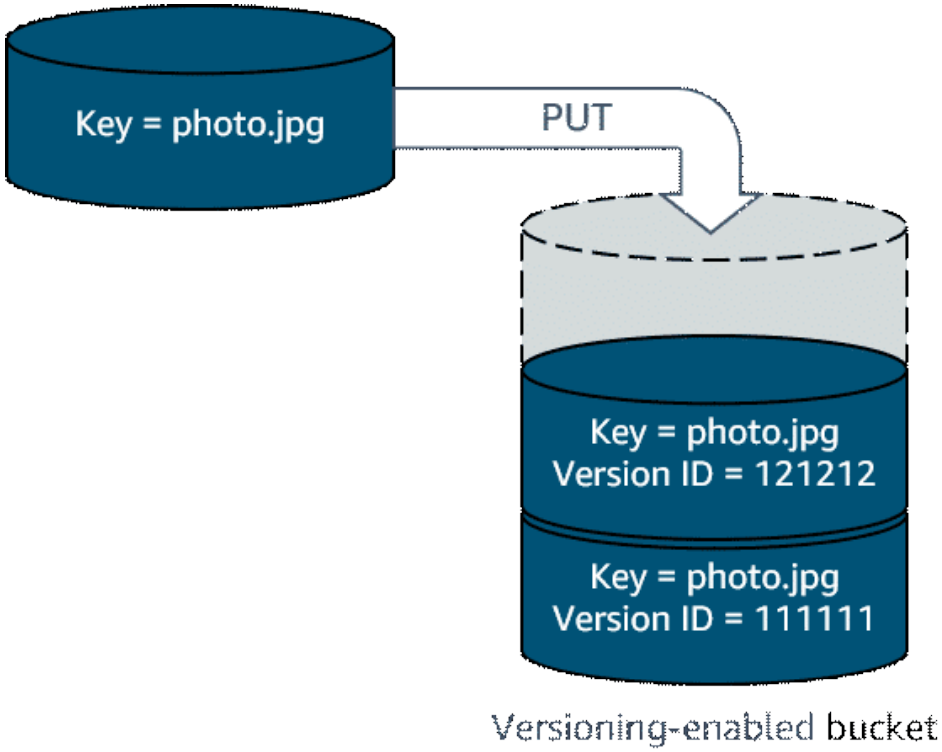
Amazon S3 versioning

As described earlier, Amazon S3 identifies objects in part by using the object name. For example, when you upload an employee photo to Amazon S3, you might name the object *employee.jpg* and store it in a bucket called employees. Without Amazon S3 versioning, every time you upload an object called employee.jpg to the employees bucket, it will overwrite the original object.

This can be an issue for several reasons, including the following:

- **Common names:** The employee.jpg object name is a common name for an employee photo object. You or someone else who has access to the bucket might not have intended to overwrite it; but once it's overwritten, the original object can't be accessed.
- **Version preservation:** You might want to preserve different versions of employee.jpg. Without versioning, if you wanted to create a new version of employee.jpg, you would need to upload the object and choose a different name for it. Having several objects all with slight differences in naming variations can cause confusion and clutter in S3 buckets.

To counteract these issues, you can use Amazon S3 versioning. Versioning keeps multiple versions of a single object in the same bucket. This preserves old versions of an object without using different names, which helps with object recovery from accidental deletions, accidental overwrites, or application failures.



If you enable versioning for a bucket, Amazon S3 automatically generates a unique version ID for the object. In one bucket, for example, you can have two objects with the same key but different version IDs, such as employeephoto.jpg (version 111111) and employeephoto.jpg (version 121212).

By using versioning-enabled buckets, you can recover objects from accidental deletion or overwrite. The following are examples:

- Deleting an object does not remove the object permanently. Instead, Amazon S3 puts a marker on the object that shows that you tried to delete it. If you want to restore the object, you can remove the marker and the object is reinstated.
- If you overwrite an object, it results in a new object version in the bucket. You still have access to previous versions of the object.

Versioning states

Buckets can be in one of three states. The versioning state applies to all objects in the bucket. Storage costs are incurred for all objects in your bucket, including all versions. To reduce your Amazon S3 bill, you might want to delete previous versions of your objects when they are no longer needed.
To learn more, expand each of the following three categories.

Unversioned (default)
No new and existing objects in the bucket have a version.

Versioning-enabled
Versioning is enabled for all objects in the bucket. After you version-enable a bucket, it can never return to an unversioned state. However, you can suspend versioning on that bucket.

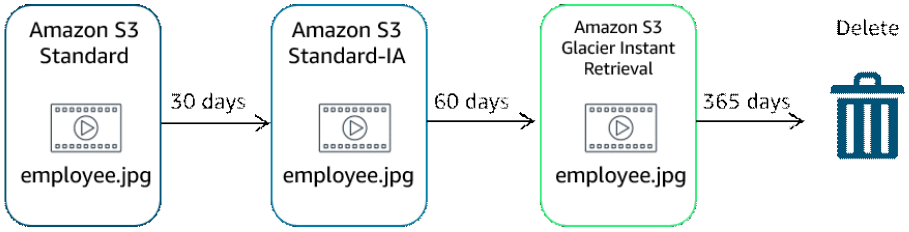
Versioning-suspended
Versioning is suspended for new objects. All new objects in the bucket will not have a version. However, all existing objects keep their object versions.

Managing your storage lifecycle

If you keep manually changing your objects, such as your employee photos, from storage tier to storage tier, you might want to automate the process by configuring their Amazon S3 lifecycle. When you define a lifecycle configuration for an object or group of objects, you can choose to automate between two types of actions: transition and expiration.

- **Transition actions** define when objects should transition to another storage class.
- **Expiration actions** define when objects expire and should be permanently deleted.

For example, you might transition objects to S3 Standard-IA storage class 30 days after you create them. Or you might archive objects to the S3 Glacier Deep Archive storage class 1 year after creating them.



The following use cases are good candidates for the use of lifecycle configuration rules:

- **Periodic logs:** If you upload periodic logs to a bucket, your application might need them for a week or a month. After that, you might want to delete them.
- **Data that changes in access frequency:** Some documents are frequently accessed for a limited period of time. After that, they are infrequently accessed. At some point, you might not need real-time access to them. But your organization or regulations might require you to archive them for a specific period. After that, you can delete them.

Module-5 Introduction to Databases on AWS

27 May 2024 11:30

A high-performing database is crucial to any organization. Databases support the internal operations of companies and store interactions with customers and suppliers.

Relational database management system

With a relational database management system (RDBMS), you can create, update, and administer a relational database. Some common examples of RDBMSs include the following:

- MySQL
- PostgresQL
- Oracle
- Microsoft SQL Server
- Amazon Aurora

You communicate with an RDBMS by using structured query language (SQL) queries, similar to the following example:
SELECT * FROM table_name.

This query selects all the data from a particular table. However, the power of SQL queries is in creating more complex queries that pull data from several tables to identify patterns and answers to business problems. For example, querying the sales table and the books table together to see sales in relation to an author's books. Querying tables together to better understand their relationships is made possible by a "join".

Relational database benefits

To learn more about the benefits of using relational databases, flip each of the following flashcards by choosing them.

- With SQL, you can **join** multiple tables so you can better understand relationships between your data.
- You can store data in one table and reference it from other tables instead of saving the same data in different places.
- Because relational databases have been a popular choice since the 1970s, technical professionals often have familiarity and experience with them.
- Relational databases ensure that your data has high integrity and adheres to the atomicity, consistency, isolation, and durability (ACID) principle.

Relational database use cases

Much of the world runs on relational databases. In fact, they're at the core of many mission-critical applications, some of which you might use in your day-to-day life. To learn some common use cases for relational databases, expand each of the following two categories.

Applications that have a fixed schema

–

These are applications that have a fixed schema and don't change often. An example is a lift-and-shift application that lifts an app from on-premises and shifts it to the cloud, with little or no modifications.
Applications that need persistent storage

–

These are applications that need persistent storage and follow the ACID principle, such as:

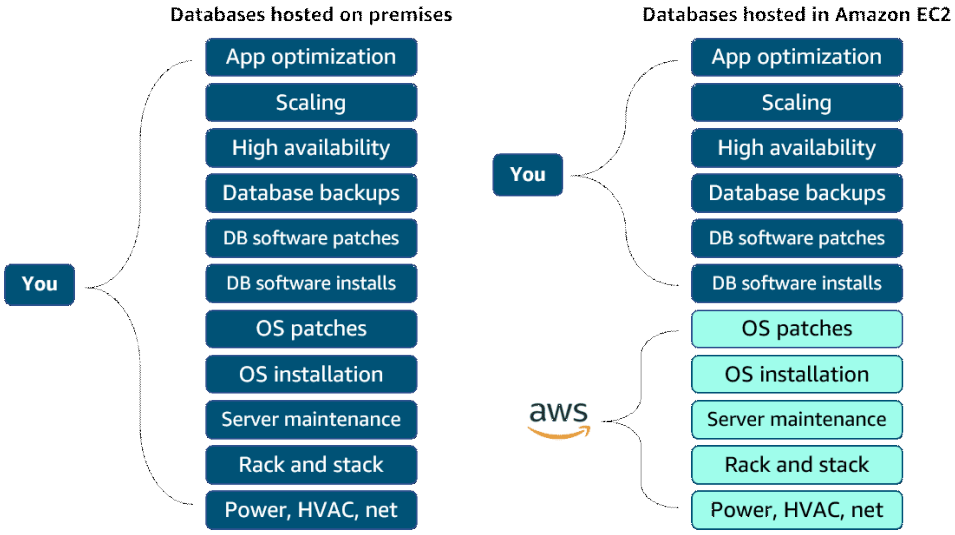
- Enterprise resource planning (ERP) applications
- Customer relationship management (CRM) applications
- Commerce and financial applications

Unmanaged databases

If you operate a relational database on premises, you are responsible for all aspects of operation. This includes data center security and electricity, host machines management, database management, query optimization, and customer data management. You are responsible for absolutely everything, which means you have control over absolutely everything.

Now, suppose you want to shift some of the work to AWS by running your relational database on Amazon Elastic Compute Cloud (Amazon EC2). If you host a database on Amazon EC2, AWS implements and maintains the physical infrastructure and hardware and installs the EC2 instance operating system (OS). However, you are still responsible for managing the EC2 instance, managing the database on that host, optimizing queries, and managing customer data.

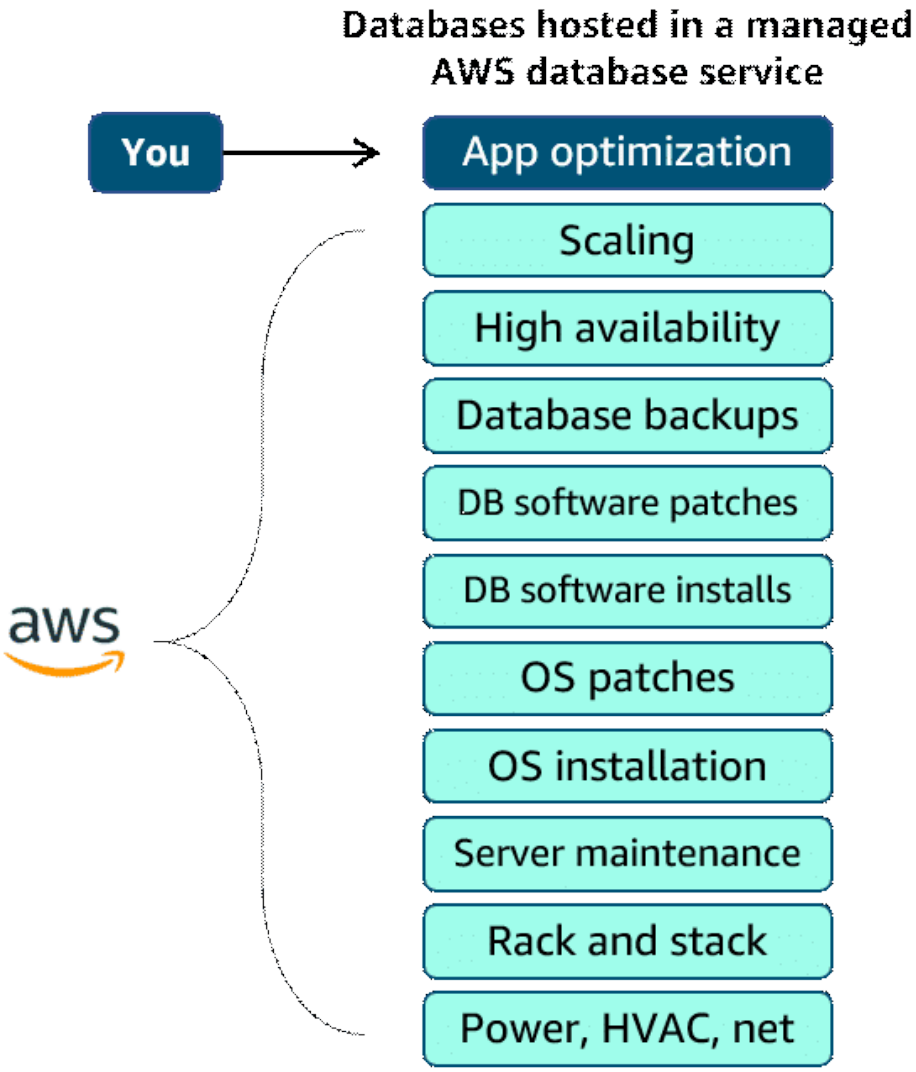
This is called an unmanaged database option. In this option, AWS is responsible for and has control over the hardware and underlying infrastructure. You are responsible for and have control over management of the host and database.



You are responsible for everything in a database hosted on-premises. AWS takes on more of that responsibility in databases hosted in Amazon EC2.

Managed databases

To shift more of the work to AWS, you can use a managed database service. These services provide the setup of both the EC2 instance and the database, and they provide systems for high availability, scalability, patching, and backups. However, in this model, you're still responsible for database tuning, query optimization, and ensuring that your customer data is secure. This option provides the ultimate convenience but the least amount of control compared to the two previous options.


Databases hosted in a managed
AWS database service
You
App optimization
Scaling
High availability
Database backups
DB software patches
DB software installs
OS patches
OS installation
Server maintenance
Rack and stack
Power, HVAC, net
aws

Module-5 Amazon RDS

27 May 2024 12:08

Amazon RDS overview

Amazon Relational Database Service (Amazon RDS)

Amazon RDS is a managed database service customers can use to create and manage relational databases in the cloud without the operational burden of traditional database management. For example, imagine you sell healthcare equipment, and your goal is to be the number-one seller on the West Coast of the United States. Building a database doesn't directly help you achieve that goal. However, having a database is a necessary component to achieving that goal.

With Amazon RDS, you can offload some of the unrelated work of creating and managing a database. You can focus on the tasks that differentiate your application, instead of focusing on infrastructure-related tasks, like provisioning, patching, scaling, and restoring.

Amazon RDS supports most of the popular RDBMSs, ranging from commercial options to open-source options and even a specific AWS option. Supported Amazon RDS engines include the following:

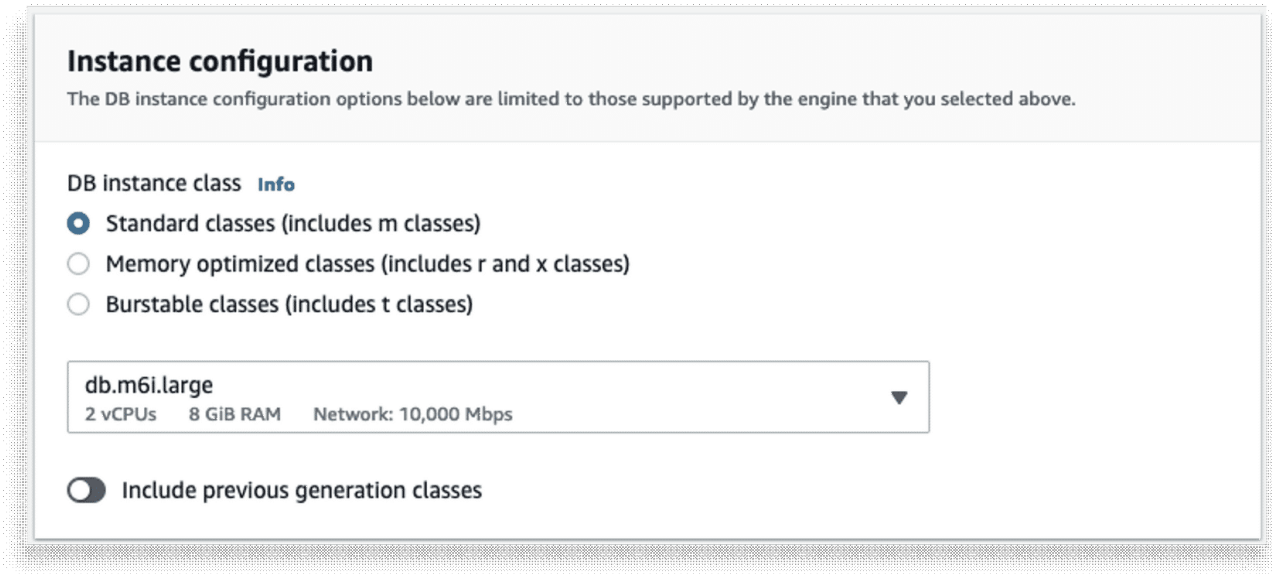
- **Commercial:** Oracle, SQL Server
- **Open source:** MySQL, PostgreSQL, MariaDB
- **Cloud native:** Aurora

Database instances

Just like the databases you build and manage yourself, Amazon RDS is built from compute and storage. The compute portion is called the database (DB) instance, which runs the DB engine. Depending on the engine selected, the instance will have different supported features and configurations. A DB instance can contain multiple databases with the same engine, and each DB can contain multiple tables.

Underneath the DB instance is an EC2 instance. However, this instance is managed through the Amazon RDS console instead of the Amazon EC2 console. When you create your DB instance, you choose the instance type and size. The DB instance class you choose affects how much processing power and memory it has.
To learn more about the various instance classes, choose each of the four

numbered markers.



Storage on Amazon RDS

The storage portion of DB instances for Amazon RDS use Amazon Elastic Block Store (Amazon EBS) volumes for database and log storage. This includes MySQL, MariaDB, PostgreSQL, Oracle, and SQL Server.
When using Aurora, data is stored in cluster volumes, which are single, virtual volumes that use solid-state drives (SSDs). A cluster volume contains copies of your data across three Availability Zones in a single AWS Region. For nonpersistent, temporary files, Aurora uses local storage.
Amazon RDS provides three storage types: General Purpose SSD (also called gp2 and gp3), Provisioned IOPS SSD (also called io1), and Magnetic (also called standard). They differ in performance characteristics and price, which means you can tailor your storage performance and cost to the needs of your database workload.
To learn more about the different storage types, choose each of the three numbered markers.

Amazon RDS in an Amazon Virtual Private Cloud

When you create a DB instance, you select the Amazon Virtual Private Cloud (Amazon VPC) your databases will live in. Then, you select the subnets that will be designated for your DB. This is called a DB subnet group, and it has at least two Availability Zones in its Region. The subnets in a DB subnet group should be private, so they don’t have a route to the internet gateway. This ensures that your DB instance, and the data inside it, can be reached only by the application backend.

Access to the DB instance can be restricted further by using network access control lists (network ACLs) and security groups. With these firewalls, you can control, at a granular level, the type of traffic you want to provide access into your database.
Using these controls provides layers of security for your infrastructure. It reinforces that only the backend instances have access to the database.

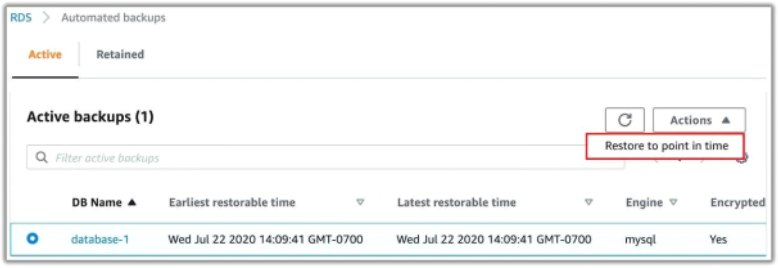
Backup data

You don’t want to lose your data. To take regular backups of your Amazon RDS instance, you can use automated backups or manual snapshots. To learn about a category, choose the appropriate tab.

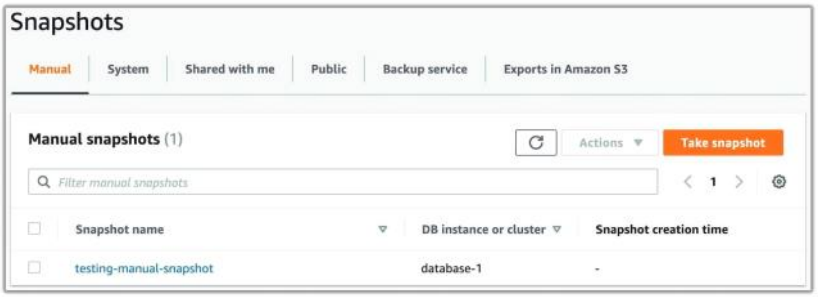
AUTOMATED BACKUPSMANUAL SNAPSHOTS
Automated backups are turned on by default. This backs up your entire DB instance (not just individual databases on the instance) and your transaction logs. When you create your DB instance, you set a backup window that is the period of time that automatic backups occur. Typically, you want to set the window during a time when your database experiences little activity because it can cause increased latency and downtime.

Retaining backups: Automated backups are retained between 0 and 35 days. You might ask yourself, “Why set automated backups for 0 days?” The 0 days setting stops automated backups from happening. If you set it to 0, it will also delete all existing automated backups. This is not ideal. The benefit of automated backups that you can do point-in-time recovery.

Point-in-time recovery: This creates a new DB instance using data restored from a specific point in time. This restoration method provides more granularity by restoring the full backup and rolling back transactions up to the specified time range.



MANUAL SNAPSHOTS
If you want to keep your automated backups longer than 35 days, use manual snapshots. Manual snapshots are similar to taking Amazon EBS snapshots, except you manage them in the Amazon RDS console. These are backups that you can initiate at any time. They exist until you delete them. For example, to meet a compliance requirement that mandates you to keep database backups for a year, you need to use manual snapshots. If you restore data from a manual snapshot, it creates a new DB instance using the data from the snapshot.



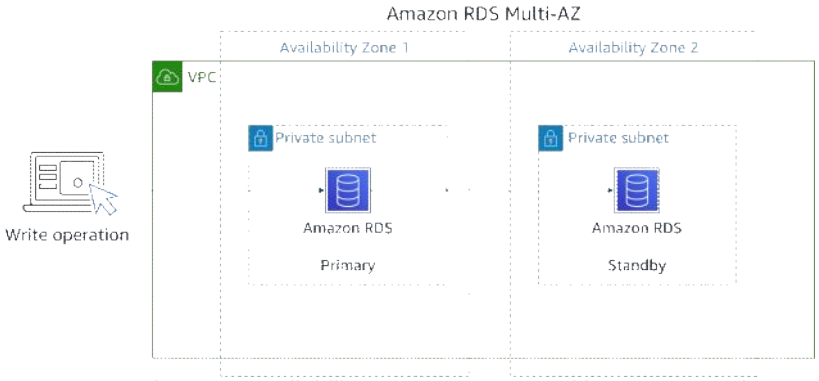
Choosing a backup option

It is advisable to deploy both backup options. Automated backups are beneficial for point-in-time recovery. With manual snapshots, you can retain backups for longer than 35 days.

Redundancy with Amazon RDS Multi-AZ

In an Amazon RDS Multi-AZ deployment, Amazon RDS creates a redundant copy of your database in another Availability Zone. You end up with two copies of your database—a primary copy in a subnet in one Availability Zone and a standby copy in a subnet in a second Availability Zone.

The primary copy of your database provides access to your data so that applications can query and display the information. The data in the primary copy is synchronously replicated to the standby copy. The standby copy is not considered an active database, and it does not get queried by applications.



To improve availability, Amazon RDS Multi-AZ ensures that you have two copies of your database running and that one of them is in the primary role. If an availability issue arises, such as the primary database loses connectivity, Amazon RDS initiates an automatic failover.

When you create a DB instance, a Domain Name System (DNS) name is provided. AWS uses that DNS name to fail over to the standby database. In an automatic failover, the standby database is promoted to the primary role, and queries are redirected to the new primary database.

To help ensure that you don't lose Multi-AZ configuration, there are two ways you can create a new standby database. They are as follows:

- Demote the previous primary to standby if it's still up and running.
- Stand up a new standby DB instance.

The reason you can select multiple subnets for an Amazon RDS database is because of the Multi-AZ configuration. You will want to ensure that you have subnets in different Availability Zones for your primary and standby copies.

Amazon RDS security

When it comes to security in Amazon RDS, you have control over managing access to your Amazon RDS resources, such as your databases on a DB instance. How you manage access will depend on the tasks you or other users need to perform in Amazon RDS. Network ACLs and security groups help users dictate the flow of traffic. If you want to restrict the actions and resources others can access, you can use AWS Identity and Access Management (IAM) policies.

27 May 2024 14:02

Purpose-built databases for all application needs

We covered Amazon RDS and relational databases in the previous lesson, and for a long time, relational databases were the default option. They were widely used in nearly all applications. A relational database is like a multi-tool. It can do many things, but it is not perfectly suited to any one particular task. It might not always be the best choice for your business needs.

The one-size-fits-all approach of using a relational database for everything no longer works. Over the past few decades, there has been a shift in the database landscape, and this shift has led to the rise of purpose-built databases. Developers can consider the needs of their application and choose a database that will fit those needs.

AWS offers a broad and deep portfolio of purpose-built databases that support diverse data models. Customers can use them to build data-driven, highly scalable, distributed applications. You can pick the best database to solve a specific problem and break away from restrictive commercial databases. You can focus on building applications that meet the needs of your organization.

Amazon DynamoDB

DynamoDB is a fully managed NoSQL database that provides fast, consistent performance at any scale. It has a flexible billing model, tight integration with infrastructure as code (IaC), and a hands-off operational model. DynamoDB has become the database of choice for two categories of applications: high-scale applications and serverless applications. Although DynamoDB is the database of choice for high-scale and serverless applications, it can work for nearly all online transaction processing (OLTP) application workloads. We will explore DynamoDB more in the next lesson.

Amazon ElastiCache

ElastiCache is a fully managed, in-memory caching solution. It provides support for two open-source, in-memory cache engines: Redis and Memcached. You aren't responsible for instance failovers, backups and restores, or software upgrades.

Amazon MemoryDB for Redis

MemoryDB is a Redis-compatible, durable, in-memory database service that delivers ultra-fast performance. With MemoryDB, you can achieve microsecond read latency, single-digit millisecond write latency, high throughput, and Multi-AZ durability for modern applications, like those built with microservices architectures. You can use MemoryDB as a fully managed, primary database to build high-performance applications. You do not need to separately manage a cache, durable database, or the required underlying infrastructure.

Amazon DocumentDB (with MongoDB compatibility)

Amazon DocumentDB is a fully managed document database from AWS. A document database is a type of NoSQL database you can use to store and query rich documents in your application. These types of databases work well for the following use cases: content management systems, profile management, and web and mobile applications. Amazon DocumentDB has API compatibility with MongoDB. This means you can use popular open-source libraries to interact with Amazon DocumentDB, or you can migrate existing databases to Amazon DocumentDB with minimal hassle.

Amazon Keyspaces (for Apache Cassandra)

Amazon Keyspaces is a scalable, highly available, and managed Apache Cassandra compatible database service. Apache Cassandra is a popular option for high-scale applications that need top-tier performance. Amazon Keyspaces is a good option for high-volume applications with straightforward access patterns. With Amazon Keyspaces, you can run your Cassandra workloads on AWS using the same Cassandra Query Language (CQL) code, Apache 2.0 licensed drivers, and tools that you use today.

Amazon Neptune

Neptune is a fully managed graph database offered by AWS. A graph database is a good choice for highly connected data with a rich variety of relationships. Companies often use graph databases for recommendation engines, fraud detection, and knowledge graphs.

Amazon Timestream

Timestream is a fast, scalable, and serverless time series database service for Internet of Things (IoT) and operational applications. It makes it easy to store and analyze trillions of events per day up to 1,000 times faster and for as little as one-tenth of the cost of relational databases. Time series data is a sequence of data points recorded over a time interval. It is used for measuring events that change over time, such as stock prices over time or temperature measurements over time.

Amazon Quantum Ledger Database (Amazon QLDB)

With traditional databases, you can overwrite or delete data, so developers use techniques, such as audit tables and audit trails to help track data lineage. These approaches can be difficult to scale and put the burden of ensuring that all data is recorded on the application developer. Amazon QLDB is a purpose-built ledger database that provides a complete and cryptographically verifiable history of all changes made to your application data.

Module-5 AWS DynamoDB

29 May 2024 10:10

DynamoDB overview

DynamoDB is a fully managed NoSQL database service that provides fast and predictable performance with seamless scalability. With DynamoDB, you can offload the administrative burdens of operating and scaling a distributed database. You don't need to worry about hardware provisioning, setup and configuration, replication, software patching, or cluster scaling.

Amazon DynamoDB

With DynamoDB, you can do the following:

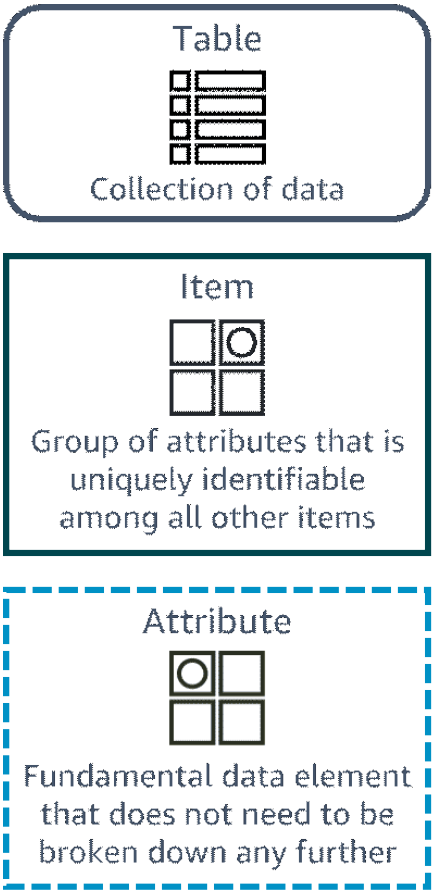
- Create database tables that can store and retrieve any amount of data and serve any level of request traffic.
- Scale up or scale down your tables' throughput capacity without downtime or performance degradation.
- Monitor resource usage and performance metrics using the AWS Management Console.

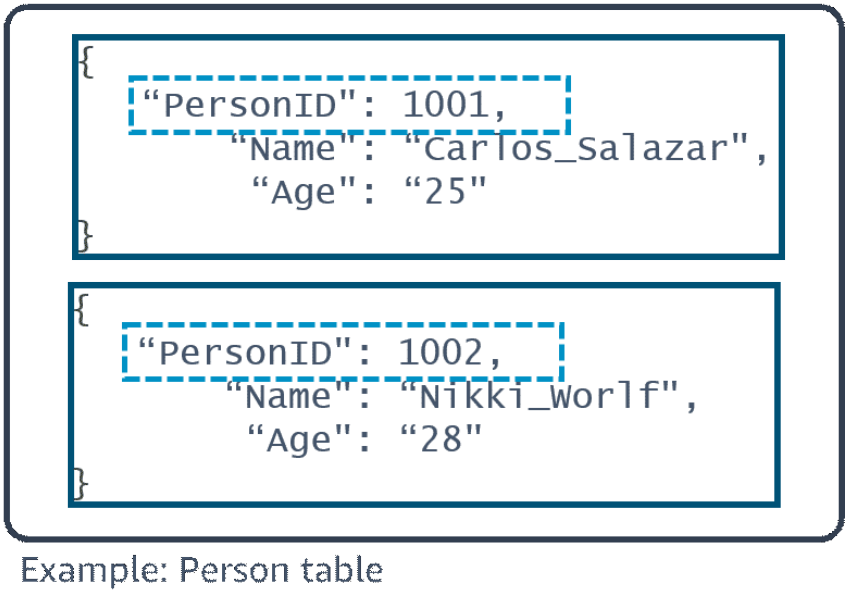
DynamoDB automatically spreads the data and traffic for your tables over a sufficient number of servers to handle your throughput and storage requirements. It does this while maintaining consistent, fast performance. All your data is stored on SSDs and is automatically replicated across multiple Availability Zones in a Region, providing built-in high availability and data durability.

DynamoDB core components

In DynamoDB, tables, items, and attributes are the core components that you work with. A table is a collection of items, and each item is a collection of attributes. DynamoDB uses primary keys to uniquely identify each item in a table and secondary indexes to provide more querying flexibility.

To learn more, choose each of the three numbered markers.





Example: Person table

1.
2.
3.

DynamoDB use cases

DynamoDB is a fully managed service that handles the operations work. You can offload the administrative burdens of operating and scaling distributed databases to AWS.
You might want to consider using DynamoDB in the following circumstances:

- You are experiencing scalability problems with other traditional database systems.
- You are actively engaged in developing an application or service.
- You are working with an OLTP workload.
- You care deploying a mission-critical application that must be highly available at all times without manual intervention.
- You require a high level of data durability, regardless of your backup-and-restore strategy.

DynamoDB is used in a wide range of workloads because of its simplicity, from low-scale operations to ultrahigh-scale operations, such as those demanded by Amazon.com.
To learn more about potential use cases, expand each of the following four categories.

Develop software applications

–

Build internet-scale applications supporting user-content metadata and caches that require high concurrency and connections for millions of users and millions of requests per second.

Create media metadata stores

–

Scale throughput and concurrency for analysis of media and entertainment

workloads, such as real-time video streaming and interactive content. Deliver lower latency with multi-Region replication across Regions.

Scale gaming platforms

–

Focus on driving innovation with no operational overhead. Build out your game platform with player data, session history, and leaderboards for millions of concurrent users.

Deliver seamless retail experiences

–

Use design patterns for deploying shopping carts, workflow engines, inventory tracking, and customer profiles. DynamoDB supports high-traffic, extreme-scaled events and can handle millions of queries per second.

DynamoDB security

DynamoDB provides a number of security features to consider as you develop and implement your own security policies. They include the following:

- bullet
  DynamoDB provides a highly durable storage infrastructure designed for mission-critical and primary data storage. Data is redundantly stored on multiple devices across multiple facilities in a DynamoDB Region.
- bullet
  All user data stored in DynamoDB is fully encrypted at rest. DynamoDB encryption at rest provides enhanced security by encrypting all your data at rest using encryption keys stored in AWS Key Management Service (AWS KMS).
- bullet
  IAM administrators control who can be authenticated and authorized to use DynamoDB resources. You can use IAM to manage access permissions and implement security policies.
- bullet
  As a managed service, DynamoDB is protected by the AWS global network security procedures.

Module-5 Choosing the Right Database Service

29 May 2024 10:27

AWS database services

As we learned in the previous lessons, AWS has a variety of database options for different use cases. The following table provides a quick look at the AWS database portfolio.

AWS Service(s)	Database Type	Use Cases
Amazon RDS, Aurora, Amazon Redshift	Relational	Traditional applications, ERP, CRM, ecommerce
DynamoDB	Key-value	High-traffic web applications, ecommerce systems, gaming applications
Amazon ElastiCache for Memcached, Amazon ElastiCache for Redis	In-memory	Caching, session management, gaming leaderboards, geospatial applications
Amazon DocumentDB	Document	Content management, catalogs, user profiles
Amazon Keyspaces	Wide column	High-scale industrial applications for equipment maintenance, fleet management, route optimization
Neptune	Graph	Fraud detection, social networking, recommendation engines
Timestream	Time series	IoT applications, Development Operations (DevOps), industrial telemetry
Amazon QLDB	Ledger	Systems of record, supply chain, registrations, banking transactions

Breaking up applications and databases

As the industry changes, applications and databases change too. Today, with larger applications, you no longer see just one database supporting them. Instead, applications are broken into smaller services, each with its own purpose-built database supporting it. This shift removes the idea of a one-size-fits-all database and replaces it with a complimentary database strategy. You can give each database the appropriate functionality, performance, and scale the workload requires.

Module-6 Monitoring

29 May 2024 11:34

Monitoring gives you insights into your applications that help you detect, investigate, and remediate problems faster.

Purpose of monitoring

When operating a website like the employee directory application on AWS, you might have questions like the following:

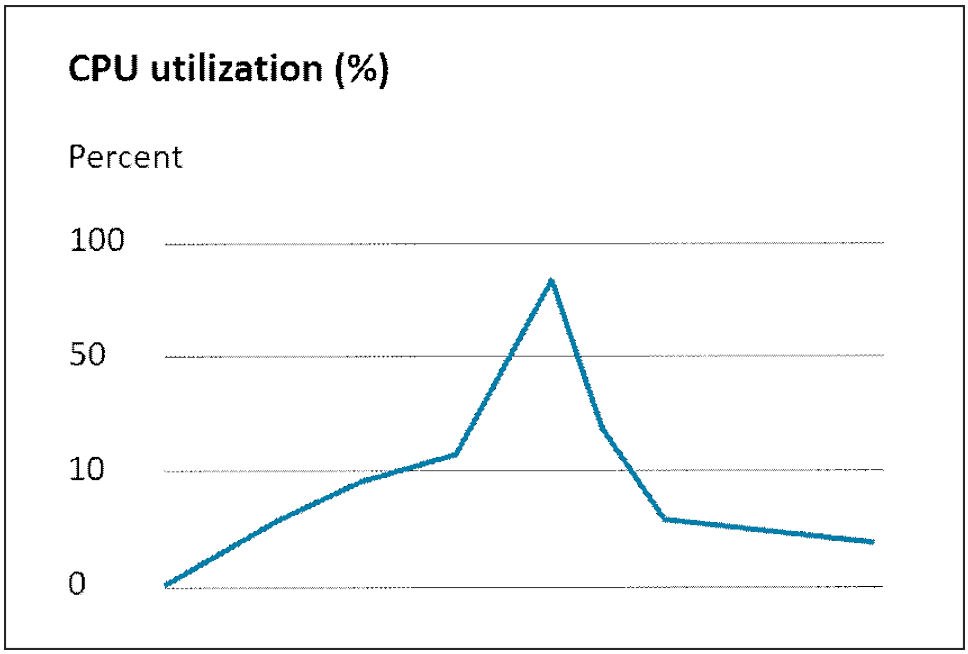
- How many people are visiting my site day to day?
- How can I track the number of visitors over time?
- How will I know if the website is having performance or availability issues?
- What happens if my Amazon Elastic Compute Cloud (Amazon EC2) instance runs out of capacity?
- Will I be alerted if my website goes down?

You need a way to collect and analyze data about the operational health and usage of your resources. The act of collecting, analyzing, and using data to make decisions or answer questions about your IT resources and systems is called monitoring.

Monitoring provides a near real-time pulse on your system and helps answer the previous questions. You can use the data you collect to watch for operational issues caused by events like overuse of resources, application flaws, resource misconfiguration, or security-related events. Think of the data collected through monitoring as outputs of the system, or metrics.

Use metrics to solve problems

The AWS resources that host your solutions create various forms of data that you might be interested in collecting. Each individual data point that a resource creates is a metric. Metrics that are collected and analyzed over time become statistics, such as average CPU utilization over time showing a spike.



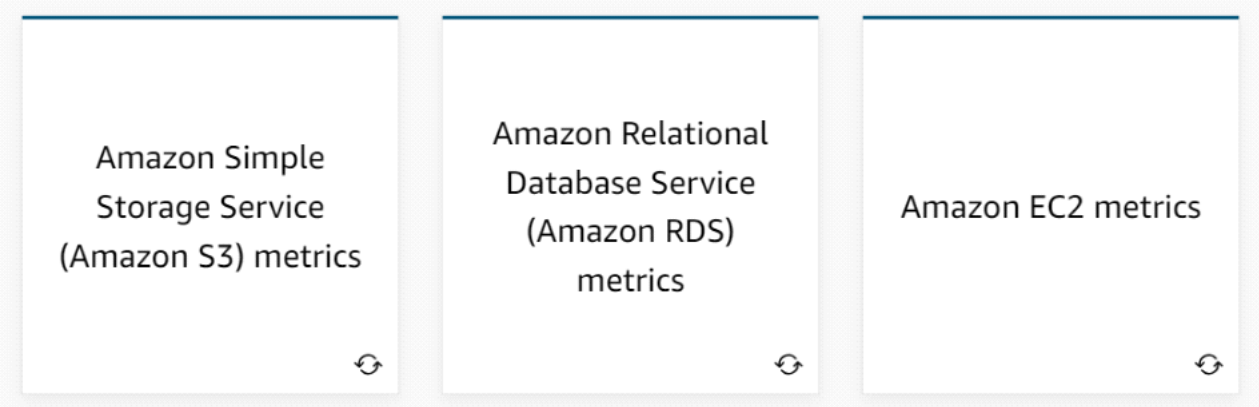
One way to evaluate the health of an EC2 instance is through CPU utilization. Generally speaking, if an EC2 instance has a high CPU utilization, it can mean a flood of requests. Or it can reflect a process that has encountered an error and is consuming too much of the CPU. When analyzing CPU utilization, take a process that exceeds a specific threshold for an unusual length of time. Use that abnormal event as a cue to either manually or automatically resolve the issue through actions

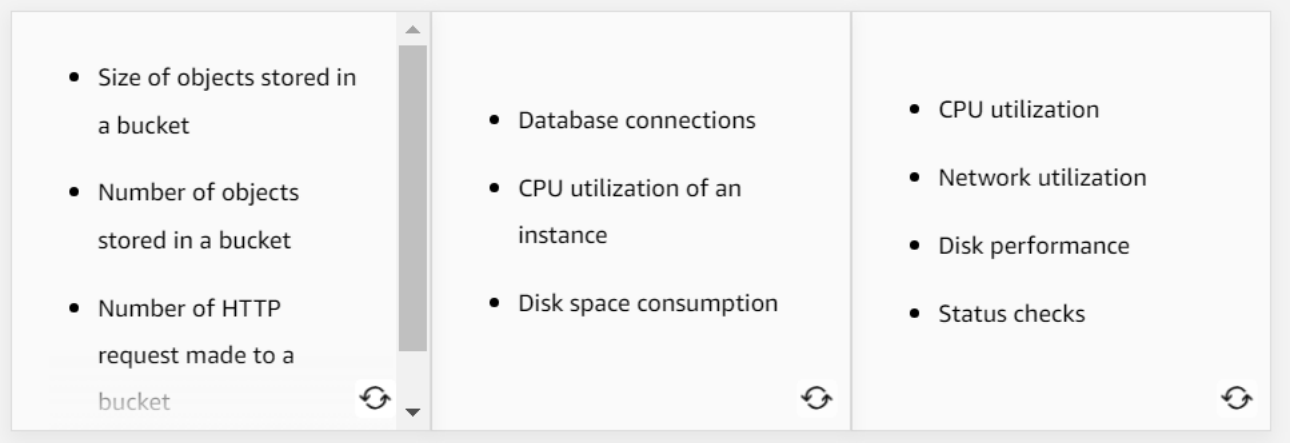
like scaling the instance.

CPU utilization is one example of a metric. Other examples of metrics that EC2 instances have are network utilization, disk performance, memory utilization, and the logs created by the applications running on top of Amazon EC2.

Types of metrics

Different resources in AWS create different types of metrics. To see examples of metrics associated with different resources, flip each of the following flashcards by choosing them.





This is not a complete list of metrics for any of the services mentioned, but you can see how different resources create different metrics. You might be interested in a wide variety of metrics depending on your resources, goals, and questions.

Monitoring benefits

Monitoring gives you visibility into your resources, but the question now is, "Why is that important?" This section describes some of the benefits of monitoring.
To learn more, expand each of the following five categories.

Respond proactively

–

Respond to operational issues proactively before your end users are aware of them. Waiting for end users to let you know when your application is experiencing an outage is a bad practice. Through monitoring, you can keep tabs on metrics like error response rate and request latency. Over time, the metrics help signal when an outage is

going to occur. You can automatically or manually perform actions to prevent the outage from happening and fix the problem before your end users are aware of it.

Improve performance and reliability

_

Monitoring can improve the performance and reliability of your resources. Monitoring the various resources that comprise your application provides you with a full picture of how your solution behaves as a system. Monitoring, if done well, can illuminate bottlenecks and inefficient architectures. This helps you drive performance and improve reliability.

Recognize security threats and events

_

By monitoring, you can recognize security threats and events. When you monitor resources, events, and systems over time, you create what is called a baseline. A baseline defines normal activity. Using a baseline, you can spot anomalies like unusual traffic spikes or unusual IP addresses accessing your resources. When an anomaly occurs, an alert can be sent out or an action can be taken to investigate the event.

Make data-driven decisions

_

Monitoring helps you make data-driven decisions for your business. Monitoring keeps an eye on IT operational health and drives business decisions. For example, suppose you launched a new feature for your cat photo app and now you want to know if it’s being used. You can collect application-level metrics and view the number of users who use the new feature. With your findings, you can decide whether to invest more time into improving the new feature.

Create cost-effective solutions

_

Through monitoring, you can create more cost-effective solutions. You can view resources that are underused and rightsize your resources to your usage. This helps you optimize cost and make sure you aren’t spending more money than necessary.

Module-6 Amazon CloudWatch

29 May 2024 11:41

Amazon CloudWatch is a monitoring and observability service that collects your resource data and provides actionable insights into your applications.

Visibility using CloudWatch

AWS resources create data that you can monitor through metrics, logs, network traffic, events, and more. This data comes from components that are distributed in nature. This can lead to difficulty in collecting the data you need if you don't have a centralized place to review it all. AWS has taken care of centralizing the data collection for you with a service called CloudWatch.

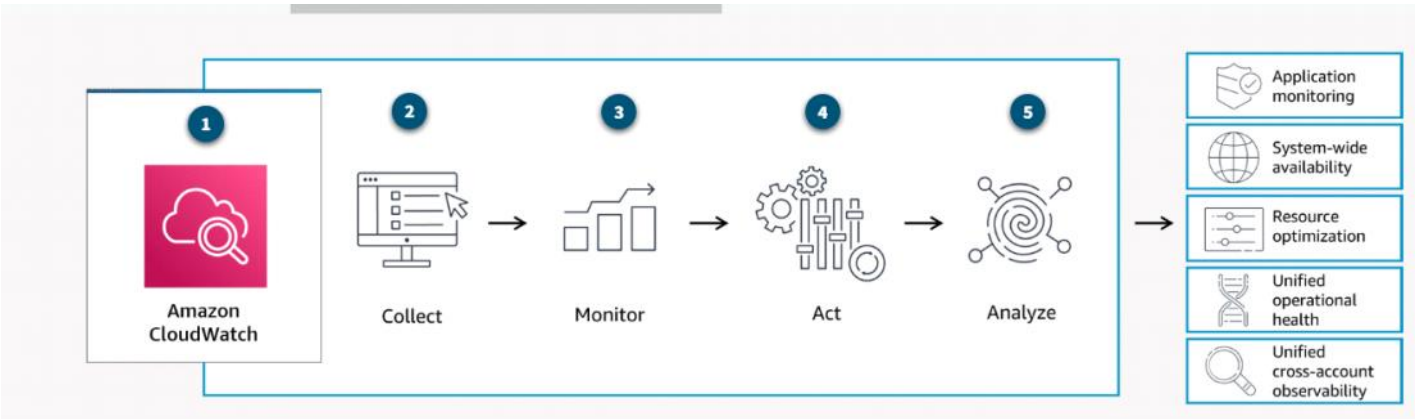
Amazon CloudWatch
CloudWatch is a monitoring and observability service that collects your resource data and provides actionable insights into your applications. With CloudWatch, you can respond to system-wide performance changes, optimize resource usage, and get a unified view of operational health.

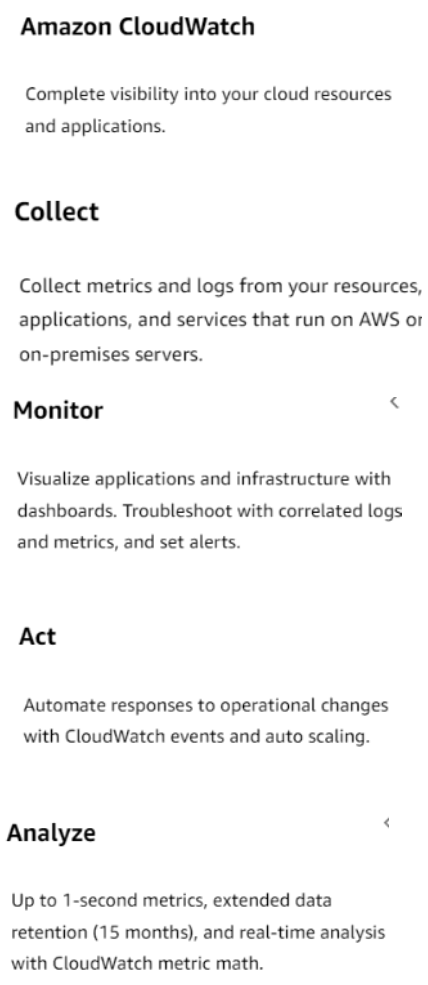
You can use CloudWatch to do the following:

- bullet
  Detect anomalous behavior in your environments.
- bullet
  Set alarms to alert you when something is not right.
- bullet
  Visualize logs and metrics with the AWS Management Console.
- bullet
  Take automated actions like scaling.
- bullet
  Troubleshoot issues.
- bullet
  Discover insights to keep your applications healthy.

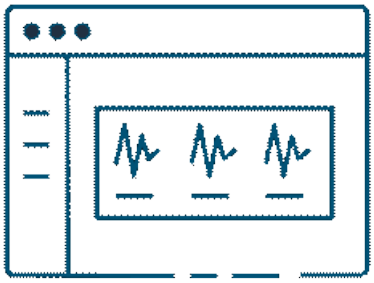
How CloudWatch works

With CloudWatch, all you need to get started is an AWS account. It is a managed service that you can use for monitoring without managing the underlying infrastructure.
To learn more, choose each numbered marker.





The employee directory application is built with various AWS services working together as building blocks. Monitoring the individual services independently can be challenging. Fortunately, CloudWatch acts as a centralized place where metrics are gathered and analyzed.



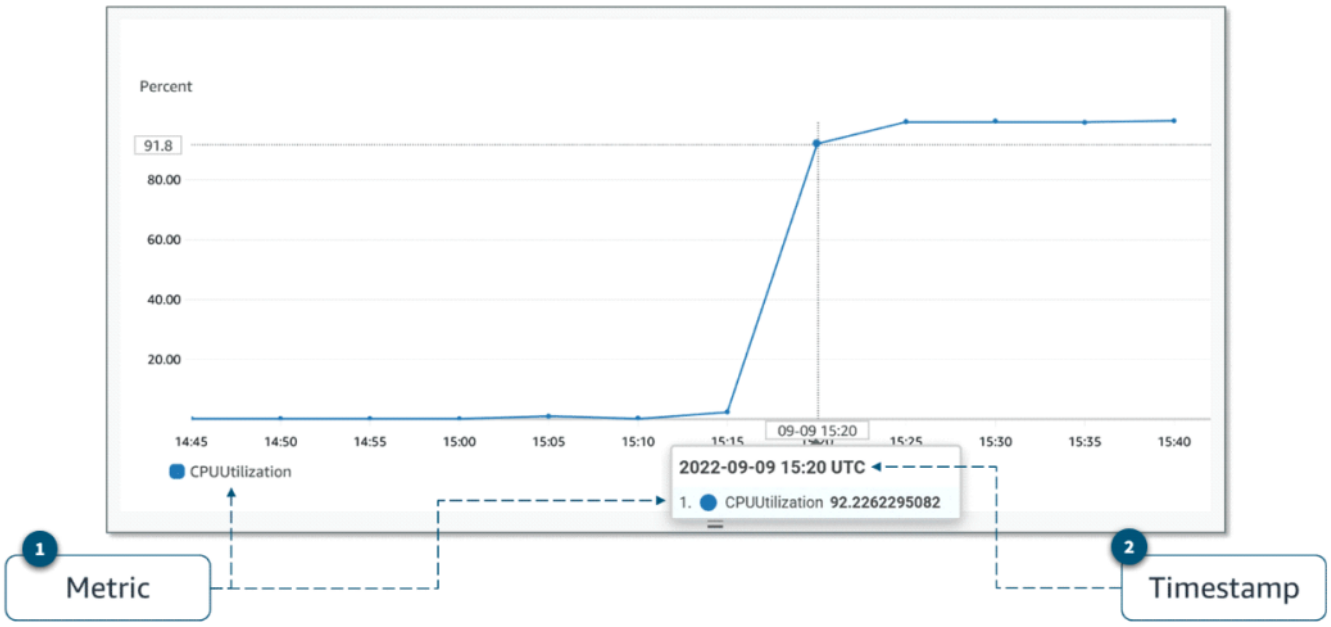
Many AWS services automatically send metrics to CloudWatch for free at a rate of 1 data point per metric per 5-minute interval. This is called basic monitoring, and it gives you visibility into your systems without any extra cost. For many applications, basic monitoring is adequate.

For applications running on EC2 instances, you can get more granularity by posting metrics every minute instead of every 5-minutes using a feature like detailed monitoring. Detailed monitoring incurs a fee. For more information about pricing, see "Amazon CloudWatch Pricing" in the Resources section at the end of this lesson.

CloudWatch concepts

Metrics are the fundamental concept in CloudWatch. A metric represents a time-ordered set of data points that are published to CloudWatch. Think of a metric as a variable to monitor and the data points as representing the values of

that variable over time. Every metric data point must be associated with a timestamp.



Metric

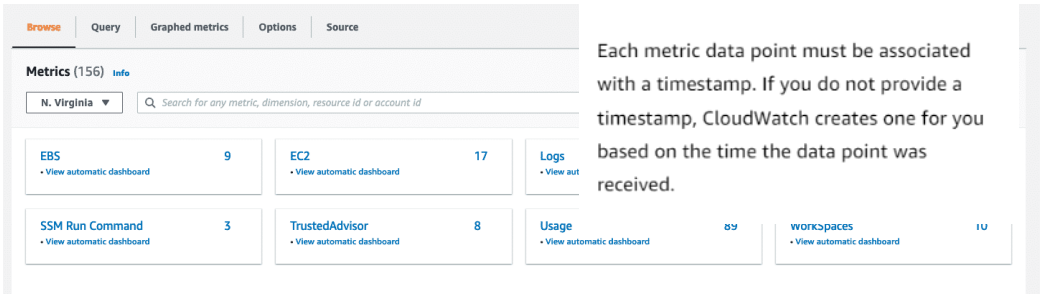
Metrics are data about the performance of your systems.

For example, the CPU usage of a particular EC2 instance is one metric provided by Amazon EC2.

AWS services that send data to CloudWatch attach dimensions to each metric. A dimension is a name and value pair that is part of the metric's identity. You can use dimensions to filter the results that CloudWatch returns. For example, many Amazon EC2 metrics publish **InstanceId** as a dime as the value for that dimension.

Timestamp

Each metric data point must be associated with a timestamp. If you do not provide a timestamp, CloudWatch creates one for you based on the time the data point was received.



By default, many AWS services provide metrics at no charge for resources such as EC2 instances, Amazon Elastic Block Store (Amazon EBS) volumes, and Amazon RDS database (DB) instances. For a charge, you can activate features such as detailed monitoring or publishing your own application metrics on resources such as your EC2 instances.

Custom metrics

Suppose you have an application, and you want to record the number of page views your website gets. How would you record this metric with CloudWatch? First, it's an application-level metric. That means it's not something the EC2 instance would post to CloudWatch by default. This is where custom metrics come in. With custom metrics, you can publish your own metrics to CloudWatch.

If you want to gain more granular visibility, you can use high-resolution custom metrics, which make it possible for you to collect custom metrics down to a 1-second resolution. This means you can send 1 data point per second per custom metric. Some examples of custom metrics include the following:

- Webpage load times
- Request error rates
- Number of processes or threads on your instance

- Amount of work performed by your application

CloudWatch dashboards

Once you provision your AWS resources and they are sending metrics to CloudWatch, you can visualize and review that data using CloudWatch dashboards. Dashboards are customizable home pages you can configure for data visualization for one or more metrics through widgets, such as a graph or text.

You can build many custom dashboards, each one focusing on a distinct view of your environment. You can even pull data from different AWS Regions into a single dashboard to create a global view of your architecture. The following screenshot an example of a dashboard with metrics from Amazon EC2 and Amazon EBS.

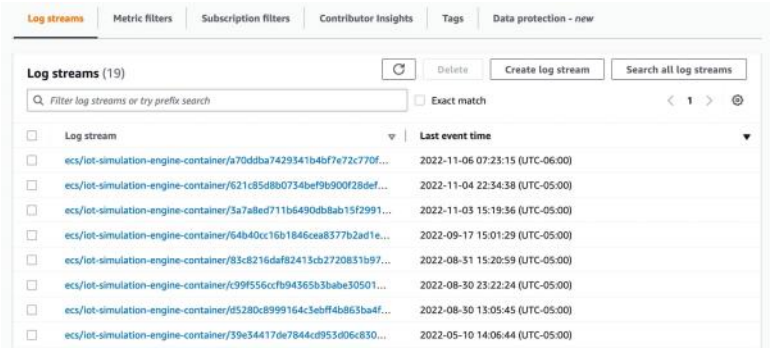
CloudWatch aggregates statistics according to the period of time that you specify when creating your graph or requesting your metrics. You can also choose whether your metric widgets display live data. Live data is data published within the last minute that has not been fully aggregated.

You are not bound to using CloudWatch exclusively for all your visualization needs. You can use external or custom tools to ingest and analyze CloudWatch metrics using the GetMetricData API.

As far as security is concerned, with AWS Identity and Access Management (IAM) policies, you control who has access to view or manage your CloudWatch dashboards.

Amazon CloudWatch Logs

CloudWatch Logs is centralized place for logs to be stored and analyzed. With this service, you can monitor, store, and access your log files from applications running on EC2 instances, AWS Lambda functions, and other sources.

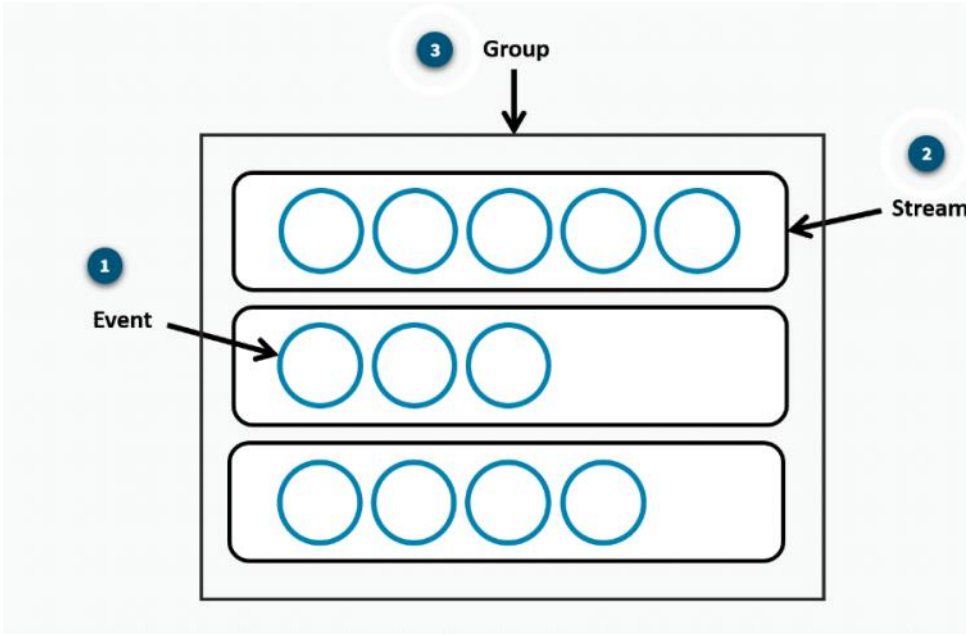


With CloudWatch Logs, you can query and filter your log data. For example, suppose you're looking into an application logic error for your application. You know that when this error occurs, it will log the stack trace. Because you know it logs the error, you query your logs in CloudWatch Logs to find the stack trace. You also set up metric filters on logs, which turn log data into numerical CloudWatch metrics that you can graph and use on your dashboards.

Some services, like Lambda, are set up to send log data to CloudWatch Logs with minimal effort. With Lambda, all you need to do is give the Lambda function the correct IAM permissions to post logs to CloudWatch Logs. Other services require more configuration. For example, to send your application logs from an EC2 instance into CloudWatch Logs, you need to install and configure the CloudWatch Logs agent on the EC2 instance. With the CloudWatch Logs agent, EC2 instances can automatically send log data to CloudWatch Logs.

CloudWatch Logs terminology

Log data sent to CloudWatch Logs can come from different sources, so it's important you understand how they're organized.
To learn more about logs terminology, choose each of the three numbered markers.



Log event

A log event is a record of activity recorded by the application or resource being monitored. It has a timestamp and an event message.

Log stream

Log events are grouped into log streams, which are sequences of log events that all belong to the same resource being monitored.

For example, logs for an EC2 instance are grouped together into a log stream that you can filter or query for insights.

Log group

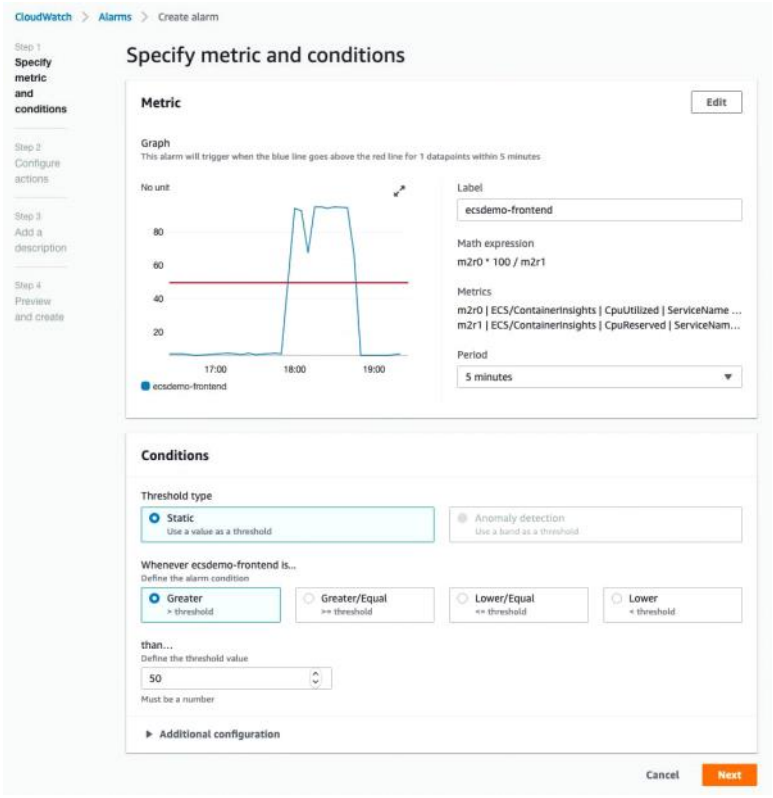
A log group is composed of log streams that all share the same retention and permissions settings.

For example, suppose you have multiple EC2 instances hosting your application and you send application log data to CloudWatch Logs. You can group the log streams from each instance into one log group.

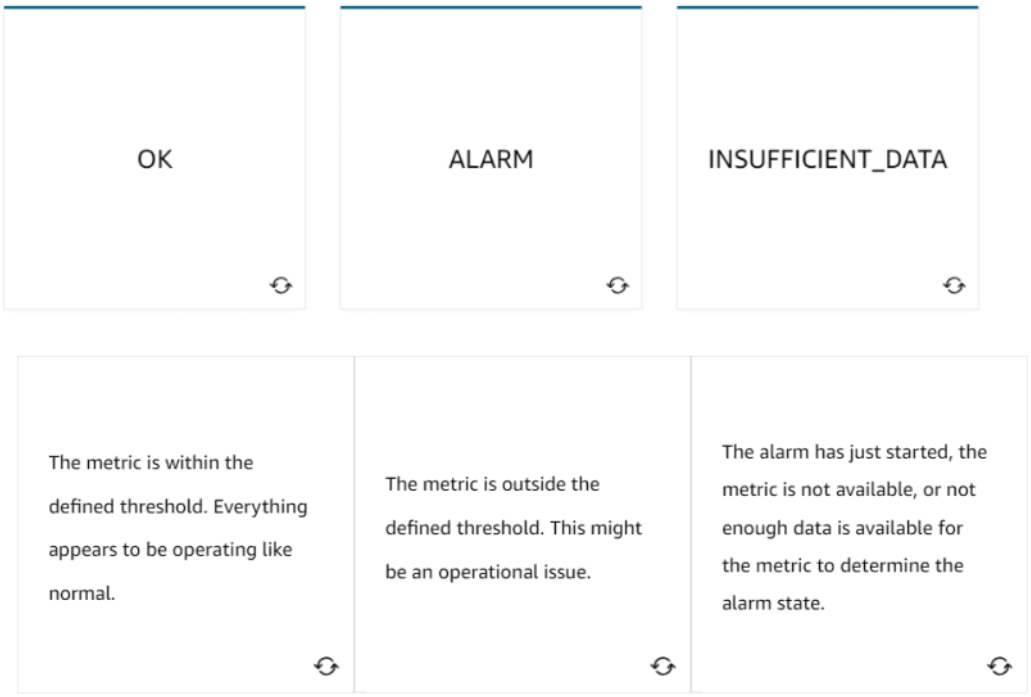
CloudWatch alarms

You can create CloudWatch alarms to automatically initiate actions based on sustained state changes of your metrics. You configure when alarms are invoked and the action that is performed.

First, you must decide which metric you want to set up an alarm for, and then you define the threshold that will invoke the alarm. Next, you define the threshold's time period. For example, suppose you want to set up an alarm for an EC2 instance to invoke when the CPU utilization goes over a threshold of 80 percent. You also must specify the time period the CPU utilization is over the threshold.
You don’t want to invoke an alarm based on short, temporary spikes in the CPU. You only want to invoke an alarm if the CPU is elevated for a sustained amount of time. For example, if CPU utilization exceeds 80 percent for 5 minutes or longer, there might be a resource issue. To set up an alarm you need to choose the metric, threshold, and time period.



An alarm can be invoked when it transitions from one state to another. After an alarm is invoked, it can initiate an action. Actions can be an Amazon EC2 action, an automatic scaling action, or a notification sent to Amazon Simple Notification Service (Amazon SNS).
To learn more about the three possible states of an alarm, flip each of the following flashcards by choosing them.



Prevent and troubleshoot issues with CloudWatch alarms

CloudWatch Logs uses metric filters to turn the log data into metrics that you can graph or set an alarm on. The following timeline indicates the order of the steps to complete when setting up an alarm. It also provides an example using our employee directory application.

- **Set up a metric filter**
  For the employee directory application, suppose you set up a metric filter for HTTP 500 error response codes.
- **Define an alarm**

Then, you define which metric alarm state should be invoked based on the threshold. With this example, the alarm state is invoked if HTTP 500 error responses are sustained for a specified period of time.

- **Define an action**

  Next, you define an action that you want to take place when the alarm is invoked. Here, it makes sense to send an email or text alert to you so you can start troubleshooting the website. Hopefully, you can fix it before it becomes a bigger issue.

  After the alarm is set up, you know that if the error happens again, you will be notified promptly.

  You can set up different alarms for different reasons to help you prevent or troubleshoot operational issues. In the scenario just described, the alarm invokes an Amazon SNS notification that goes to a person who looks into the issue manually. Another option is to have alarms invoke actions that automatically remediate technical issues. For example, you can set up an alarm to invoke an EC2 instance to reboot or scale services up or down. You can even set up an alarm to invoke an Amazon SNS notification that invokes a Lambda function. The Lambda function then calls any AWS API to manage your resources and troubleshoot operational issues. By using AWS services together like this, you can respond to events more quickly.

Module-6 Solution Optimization

29 May 2024 12:05

Design your system to have no single point of failure by using automated monitoring, failure detection, and failover mechanisms.

Availability

The availability of a system is typically expressed as a percentage of uptime in a given year or as a number of nines. In the following table is a list of availability percentages based on the downtime per year and its notation in nines.

Availability (%)	Downtime (per year)
90% (one nine of availability)	36.53 days
99% (two nines of availability)	3.65 days
99.9% (three nines of availability)	8.77 hours
99.95% (three and a half nines of availability)	4.38 hours
99.99% (four nines of availability)	52.60 minutes
99.995% (four and a half nines of availability)	26.30 minutes
99.999% (five nines of availability)	5.26 minutes

To increase availability, you need redundancy. This typically means more infrastructure—more data centers, more servers, more databases, and more replication of data. You can imagine that adding more of this infrastructure means a higher cost. Customers want the application to always be available, but you need to draw a line where adding redundancy is no longer viable in terms of revenue.

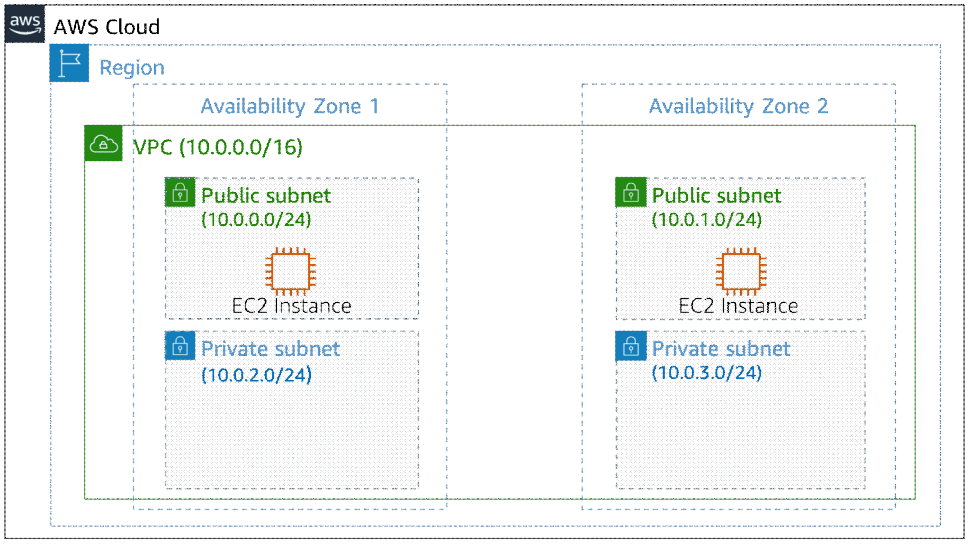
Why improve application availability?

In the current application, one EC2 instance hosts the application. The photos are served from Amazon S3, and the structured data is stored in Amazon DynamoDB. That single EC2 instance is a single point of failure for the application.

Even if the database and Amazon S3 are highly available, customers have no way to connect if the single instance becomes unavailable. One way to solve this single point of failure issue is to add one more server in a second Availability Zone.

Adding a second Availability Zone

The physical location of a server is important. In addition to potential software issues at the operating system (OS) or application level, you must also consider hardware issues. They might be in the physical server, the rack, the data center, or even the Availability Zone hosting the virtual machine. To remedy the physical location issue, you can deploy a second EC2 instance in a second Availability Zone. This second instance might also solve issues with the OS and the application.



However, when there is more than one instance, it brings new challenges, such as the following:

- bullet
  Replication process – The first challenge with multiple EC2 instances is that you need to create a process to replicate the configuration files, software patches, and application across instances. The best method is to automate where you can.
- bullet
  Customer redirection – The second challenge is how to notify the clients—the computers sending requests to your server—about the different servers. You can use various tools here. The most common is using a Domain Name System (DNS) where the client uses one record that points to the IP address of all available servers. However, this method isn't always used because of propagation — the time frame it takes for DNS changes to be updated across the Internet.

  Another option is to use a load balancer, which takes care of health checks and distributing the load across each server. Situated between the client and the server, a load balancer avoids propagation time issues. You will learn more about load balancers in the next section.
- bullet
  Types of high availability – The last challenge to address when there is more than one server is the type of availability you need: active-passive or active-active.

To learn more about types of high availability, expand each of the following two categories.

Active-passive systems

–

With an active-passive system, only one of the two instances is available at a time. One advantage of this method is that for stateful applications (where data about the client's session is stored on the server), there won't be any issues. This is because the customers are always sent to the server where their session is stored.

Active-active systems

–

A disadvantage of an active-passive system is scalability. This is where an active-active system shines. With both servers available, the second server can take some load for the application, and the entire system can

take more load. However, if the application is stateful, there would be an issue if the customer’s session isn’t available on both servers. Stateless applications work better for active-active systems.

Module-6 Traffic Routing with Elastic Load Balancing

29 May 2024 12:12

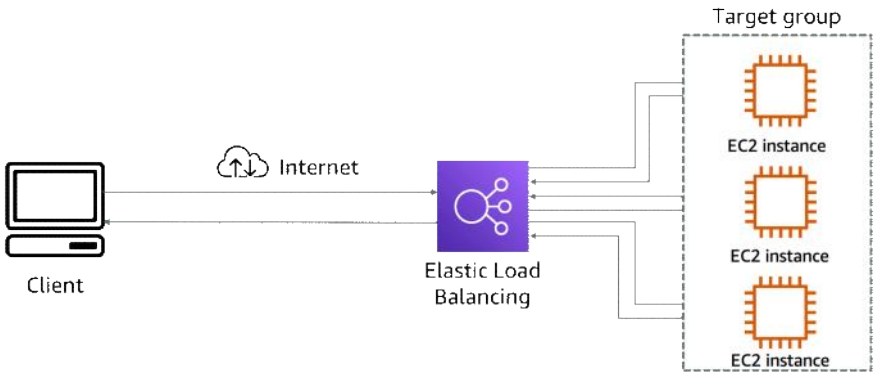
The Elastic Load Balancing (ELB) service can distribute incoming application traffic across EC2 instances, containers, IP addresses, and Lambda functions.

Load balancers

Load balancing refers to the process of distributing tasks across a set of resources. In the case of the Employee Directory application, the resources are EC2 instances that host the application, and the tasks are the requests being sent. You can use a load balancer to distribute the requests across all the servers hosting the application.

To do this, the load balancer needs to take all the traffic and redirect it to the backend servers based on an algorithm. The most popular algorithm is round robin, which sends the traffic to each server one after the other.
A typical request for an application starts from a client's browser. The request is sent to a load balancer. Then, it's sent to one of the EC2 instances that hosts the application. The return traffic goes back through the load balancer and back to the client's browser.

Although it is possible to install your own software load balancing solution on EC2 instances, AWS provides the ELB service for you.



ELB is directly in the path of the traffic.

ELB features

The ELB service provides a major advantage over using your own solution to do load balancing. Mainly, you don't need to manage or operate ELB. It can distribute incoming application traffic across EC2

instances, containers, IP addresses, and Lambda functions. Other key features include the following:

- bullet
  Hybrid mode – Because ELB can load balance to IP addresses, it can work in a hybrid mode, which means it also load balances to on-premises servers.
- bullet
  High availability – ELB is highly available. The only option you must ensure is that the load balancer's targets are deployed across multiple Availability Zones.
- bullet
  Scalability – In terms of scalability, ELB automatically scales to meet the demand of the incoming traffic. It handles the incoming traffic and sends it to your backend application.

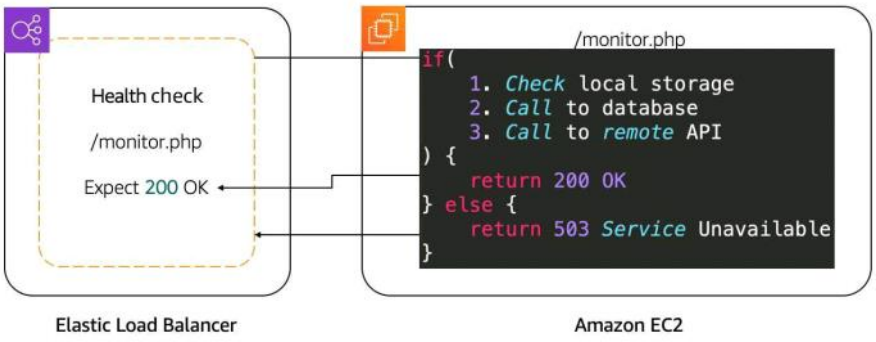
Health checks

Monitoring is an important part of load balancers because they should route traffic to only healthy EC2 instances. That's why ELB supports two types of health checks as follows:

- bullet
  Establishing a connection to a backend EC2 instance using TCP and marking the instance as available if the connection is successful.
- bullet
  Making an HTTP or HTTPS request to a webpage that you specify and validating that an HTTP response code is returned.

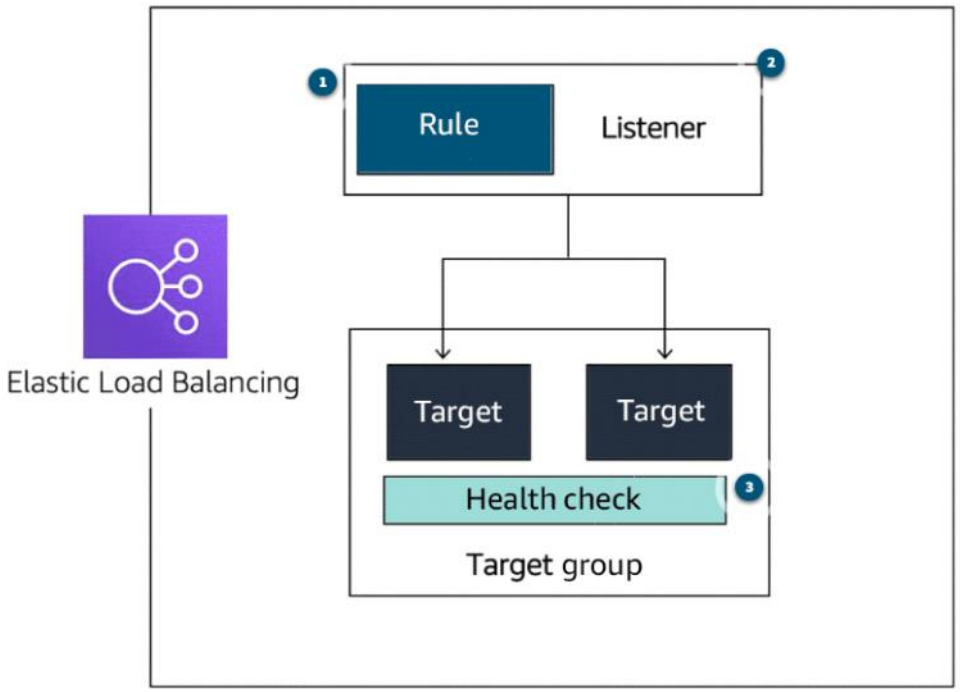
Taking time to define an appropriate health check is critical. Only verifying that the port of an application is open doesn't mean that the application is working. It also doesn't mean that making a call to the home page of an application is the right way either.

For example, the Employee Directory application depends on a database and Amazon S3. The health check should validate all the elements. One way to do that is to create a monitoring webpage, such as **/monitor**. It will make a call to the database to ensure that it can connect, get data, and make a call to Amazon S3. Then, you point the health check on the load balancer to the **/monitor** page.



After determining the availability of a new EC2 instance, the load balancer starts sending traffic to it. If ELB determines that an EC2 instance is no longer working, it stops sending traffic to it and informs Amazon EC2 Auto Scaling. It is the responsibility of Amazon EC2 Auto Scaling to remove that instance from the group and replace it with a new EC2 instance. Traffic is only sent to the new instance if it passes the health check.

If Amazon EC2 Auto Scaling has a scaling policy that calls for a scale down action, it informs ELB that the EC2 instance will be terminated. ELB can prevent Amazon EC2 Auto Scaling from terminating an EC2 instance until all connections to the instance end. It also prevents any new connections. This feature is called connection draining. We will learn more about Amazon EC2 Auto Scaling in the next lesson.



Rule

To associate a target group to a listener, you must use a rule. Rules are made up of two conditions. The first condition is the source IP address of the client. The second condition decides which target group to send the traffic to.

Listener

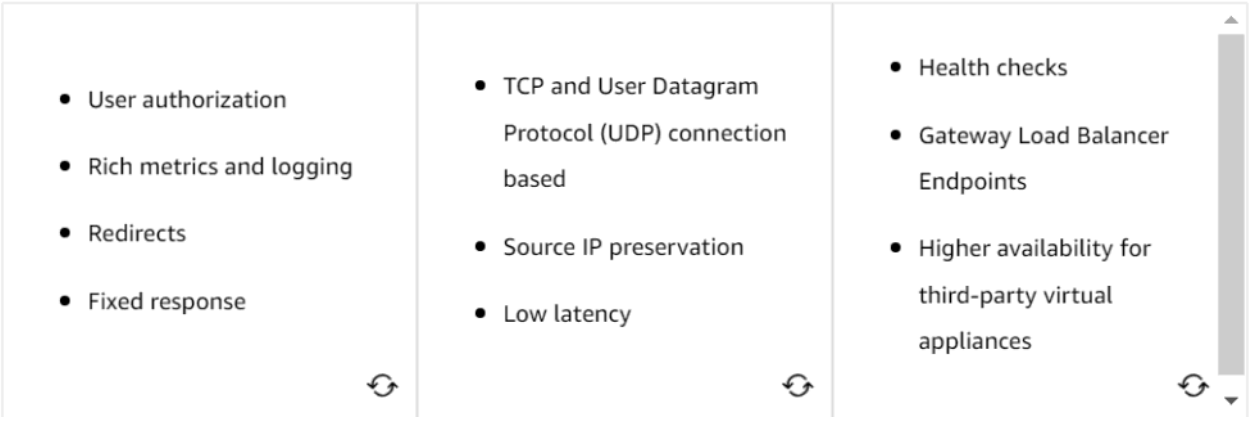
The client connects to the listener. This is often called client side. To define a listener, a port must be provided in addition to the protocol, depending on the load balancer type. There can be many listeners for a single load balancer.

Target group

The backend servers, or server side, are defined in one or more target groups. This is where you define the type of backend you want to direct traffic to, such as EC2 instances, Lambda functions, or IP addresses. Also, a health check must be defined for each target group.

Types of load balancers

We will cover three types of load balancers: Application Load Balancer (ALB), Network Load Balancer (NLB), and Gateway Load Balancer (GLB).
To learn more about each type of load balancer, flip each of the following flashcards by choosing them.



Application Load Balancer

For our Employee Directory application, we are using an Application Load Balancer. An Application Load Balancer functions at Layer 7 of the Open Systems Interconnection (OSI) model. It is ideal for load balancing HTTP and HTTPS traffic. After the load balancer receives a request, it evaluates the listener rules in priority order to determine which rule to apply. It then routes traffic to targets based on the request content.

To learn more about some of the primary features of an Application Load Balancer, expand each of the following six categories.

Routes traffic based on request data

–

An Application Load Balancer makes routing decisions based on the HTTP and HTTPS protocol. For example, the ALB could use the URL path (/upload) and host, HTTP headers and method, or the source IP address of the client. This facilitates granular routing to target groups.

Sends responses directly to the client

–

An Application Load Balancer can reply directly to the client with a fixed response, such as a custom HTML page. It can also send a redirect to the client. This is useful when you must redirect to a specific website or redirect a request from HTTP to HTTPS. It removes that work from your backend servers.

Uses TLS offloading

–

An Application Load Balancer understands HTTPS traffic. To pass HTTPS traffic through an

Application Load Balancer, an SSL certificate is provided one of the following ways:

- Importing a certificate by way of IAM or ACM services
- Creating a certificate for free using ACM

This ensures that the traffic between the client and Application Load Balancer is encrypted.

Authenticates users

–

An Application Load Balancer can authenticate users before they can pass through the load balancer. The Application Load Balancer uses the OpenID Connect (OIDC) protocol and integrates with other AWS services to support protocols, such as the following:

- SAML
- Lightweight Directory Access Protocol (LDAP)
- Microsoft Active Directory
- Others

Secures traffic

–

To prevent traffic from reaching the load balancer, you configure a security group to specify the supported IP address ranges.

Supports sticky sessions

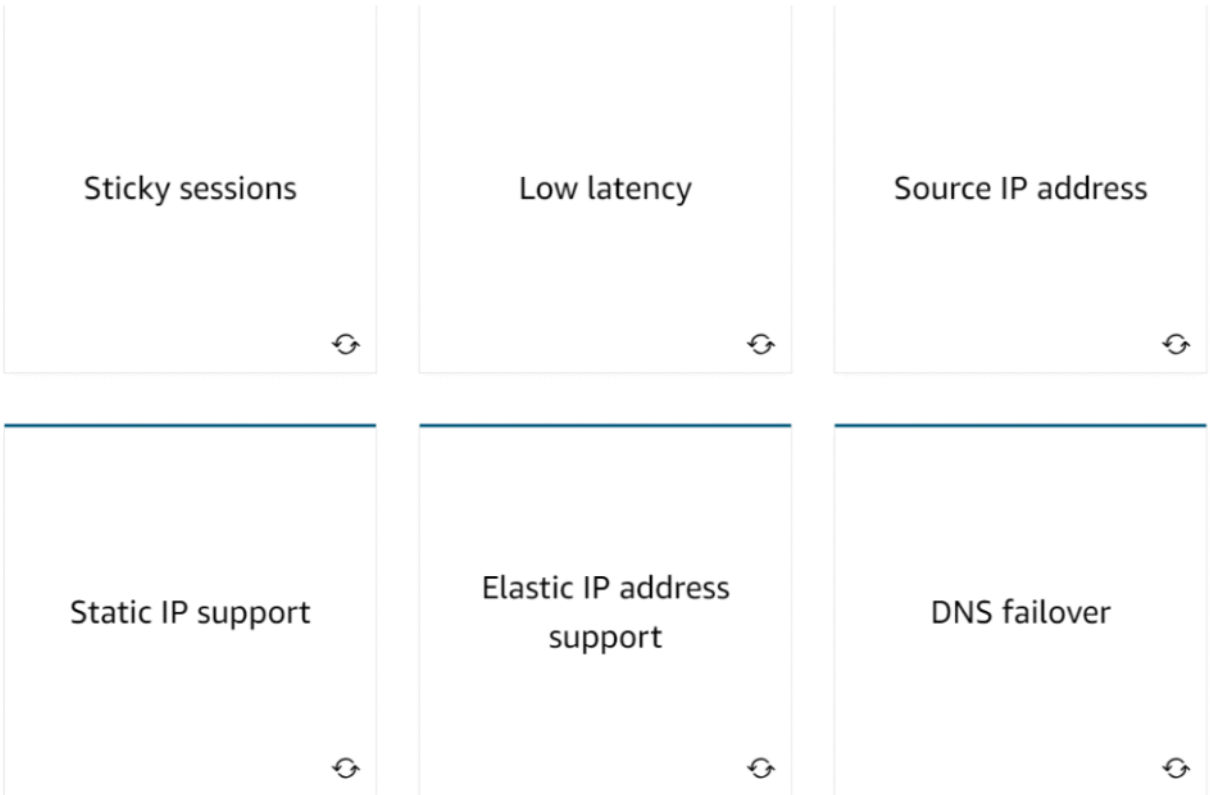
–

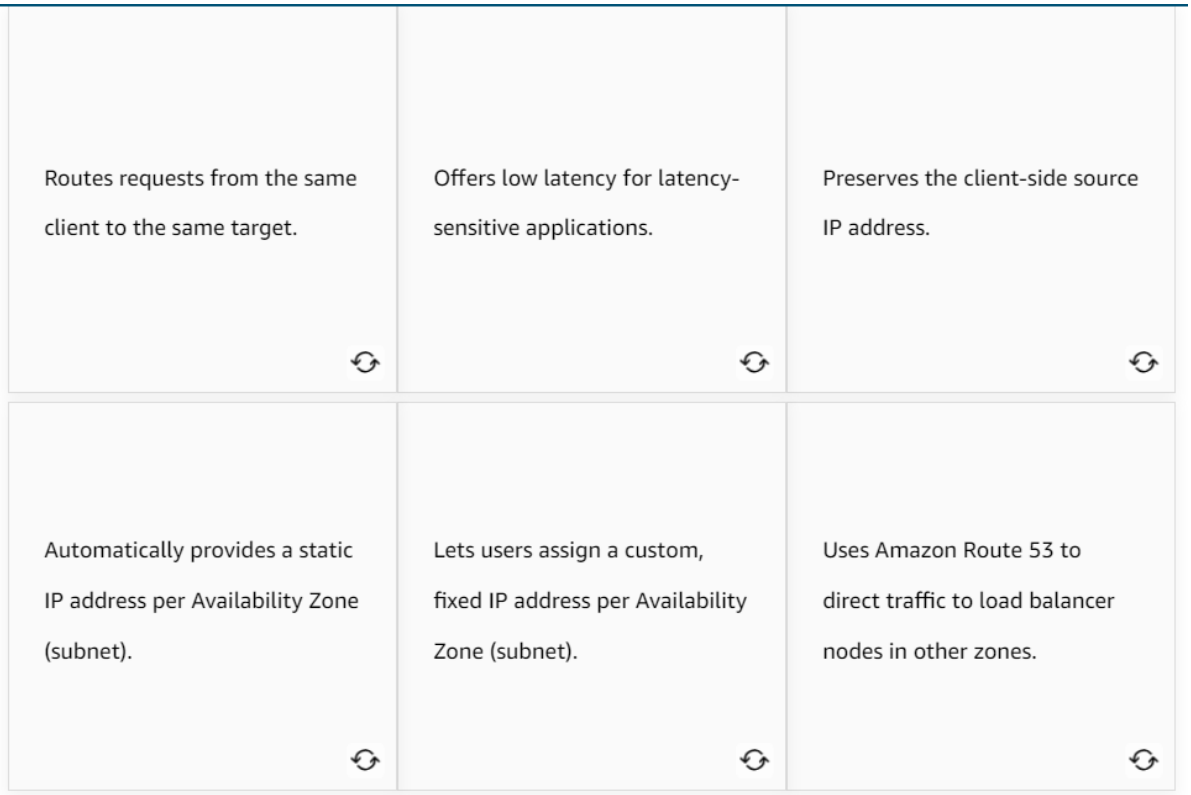
If requests must be sent to the same backend server because the application is stateful, use the sticky session feature. This feature uses an HTTP cookie to remember which server to send the traffic to across connections.

Network Load Balancer

A Network Load Balancer is ideal for load balancing TCP and UDP traffic. It functions at Layer 4 of the OSI model, routing connections from a target in the target group based on IP protocol data.

To learn more about some of the primary features of Network Load Balancers, flip each of the following flashcards by choosing them.

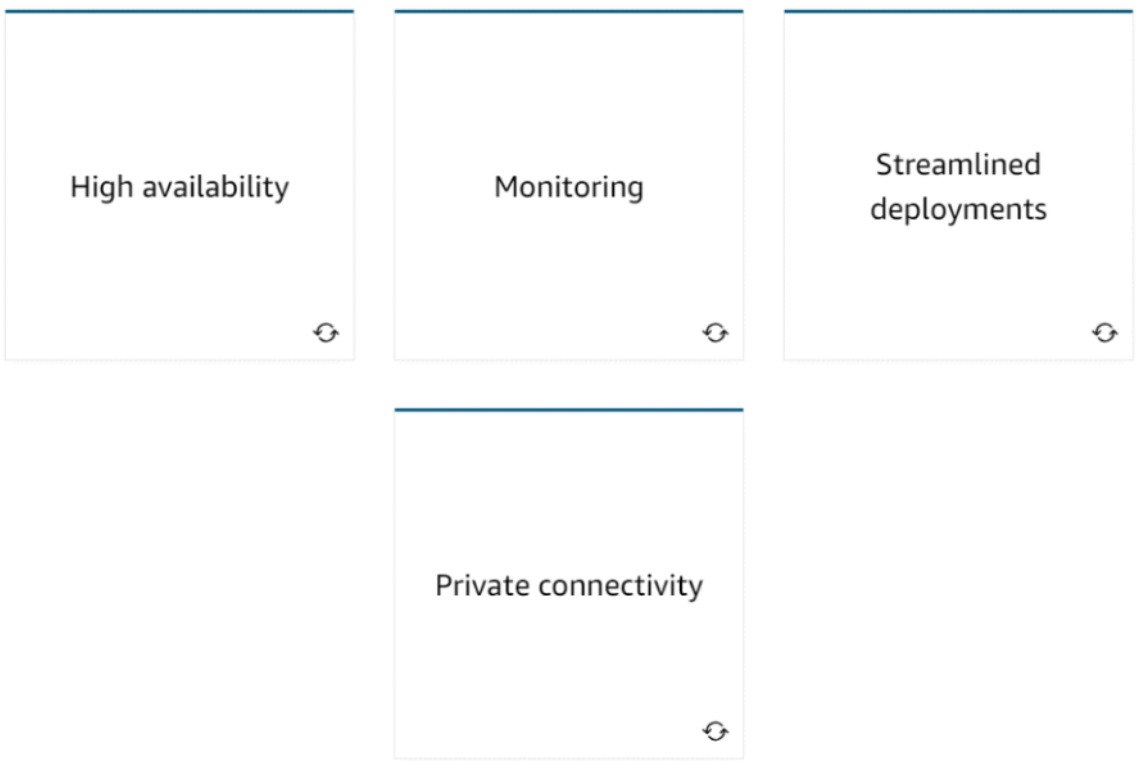


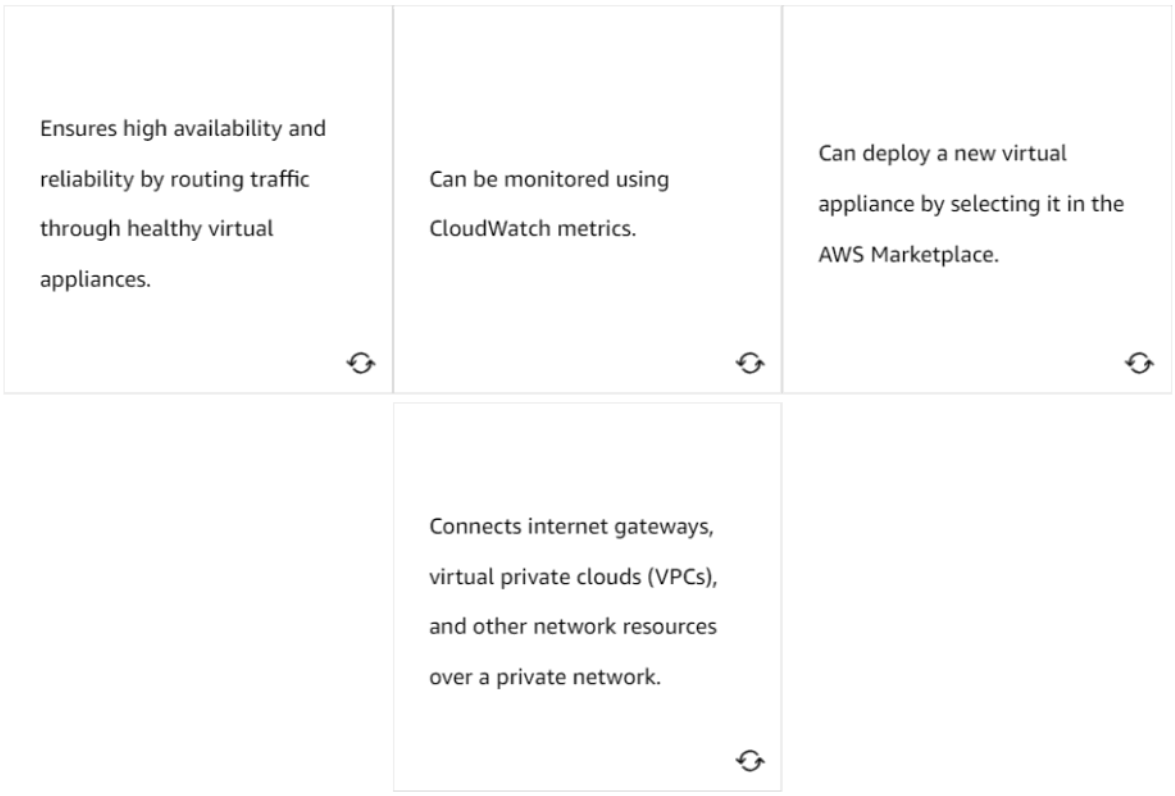


Gateway Load Balancer

A Gateway Load Balancer helps you to deploy, scale, and manage your third-party appliances, such as firewalls, intrusion detection and prevention systems, and deep packet inspection systems. It provides a gateway for distributing traffic across multiple virtual appliances while scaling them up and down based on demand.

To learn more about some of the primary features of Gateway Load Balancers, flip each of the following flashcards by choosing them.





Selecting between ELB types

You can select between the ELB service types by determining which feature is required for your application. The following table presents a list of some of the major features of load balancers. For a complete list, see "Elastic Load Balancing features" in the Resources section at the end of this lesson.

Feature	ALB	NLB	GLB
Load Balancer Type	Layer 7	Layer 4	Layer 3 gateway and Layer 4 load balancing
Target Type	IP, instance, Lambda	IP, instance, ALB	IP, instance
Protocol Listeners	HTTP, HTTPS	TCP, UDP, TLS	IP
Static IP and Elastic IP Address		Yes	
Preserve Source IP Address	Yes	Yes	Yes
Fixed Response	Yes		
User Authentication	Yes		

Module-6 Amazon EC2 Auto Scaling

29 May 2024 12:37

Amazon EC2 Auto Scaling helps you maintain application availability. You can automatically add or remove EC2 instances using scaling policies that you define.

Capacity issues

You can improve availability and reachability by adding one more server. However, the entire system can again become unavailable if there is a capacity issue. This section looks at load issues for both active-passive systems and active-active systems. These issues are addressed through scaling.
To learn more, choose each tab.

VERTICAL SCALINGHORIZONTAL SCALING

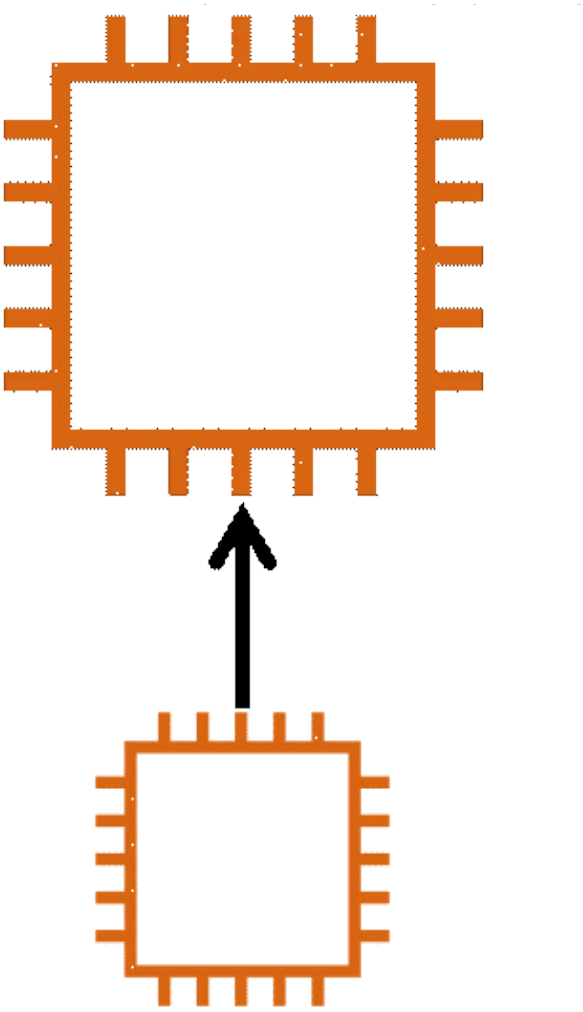
Increase the instance size. If too many requests are sent to a single active-passive system, the active server will become unavailable and, hopefully, fail over to the passive server. But this doesn't solve anything.

With active-passive systems, you need vertical scaling. This means increasing the size of the server. With EC2 instances, you select either a larger type or a different instance type. This can be done only while the instance is in a stopped state. In this scenario, the following steps occur:

1. **Stop the passive instance.** This doesn't impact the application because it's not taking any traffic.
2. **Change the instance size or type,** and then start the instance again.
3. **Shift the traffic** to the passive instance, turning it active.
4. **Stop, change the size, and start** the previous active instance because both instances should match.

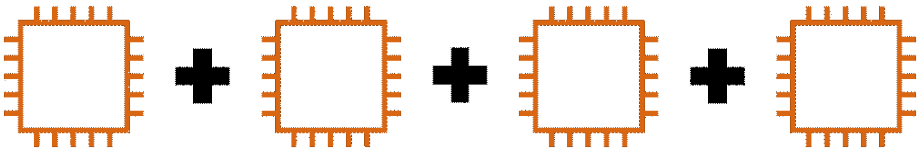
When the number of requests reduces, you must do the same operation. Even though there aren't that many steps involved, it's actually a lot of manual work. Another disadvantage is that a server can only scale vertically up to a certain limit. When that limit is reached, the only option is to create another active-passive system and split the requests and functionalities across them. This can require massive application rewriting.

This is where the active-active system can help. When there are too many requests, you can scale this system horizontally by adding more servers.



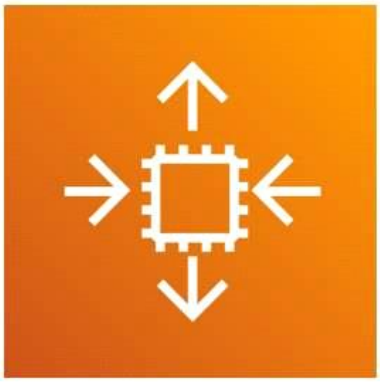
HORIZONTAL SCALING
Add additional instances. As mentioned, for the application to work in an active-active system, it's already created as stateless, not storing any client sessions on the server. This means that having two or four servers wouldn't require any application changes. It would only be a matter of creating more instances when required and shutting them down when traffic decreases. The Amazon EC2 Auto Scaling service can take care of that task by automatically creating and removing EC2 instances based on metrics from Amazon CloudWatch. We will learn more about this service in this lesson.

You can see that there are many more advantages to using an active-active system in comparison with an active-passive system. Modifying your application to become stateless provides scalability.



Traditional scaling compared to auto scaling

With a traditional approach to scaling, you buy and provision enough servers to handle traffic at its peak. However, this means at nighttime, for example, you might have more capacity than traffic, which means you're wasting money. Turning off your servers at night or at times when the traffic is lower only saves on electricity.

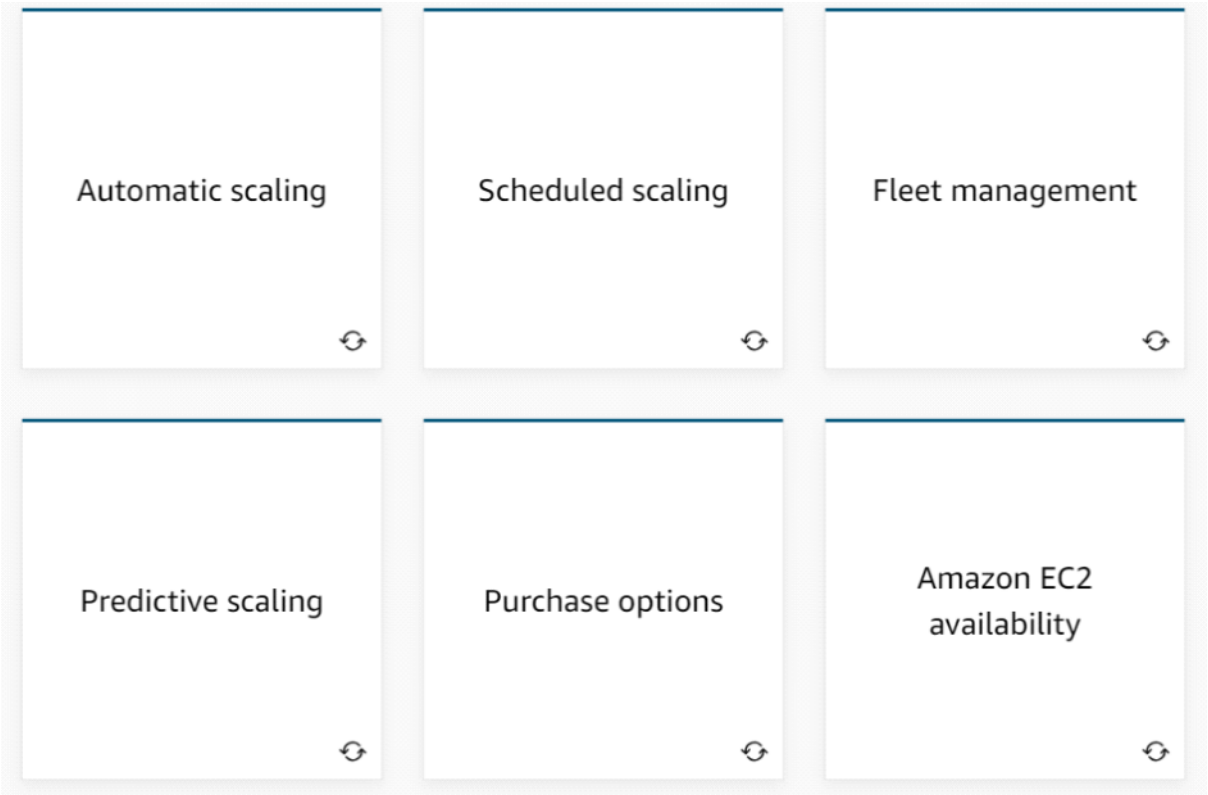


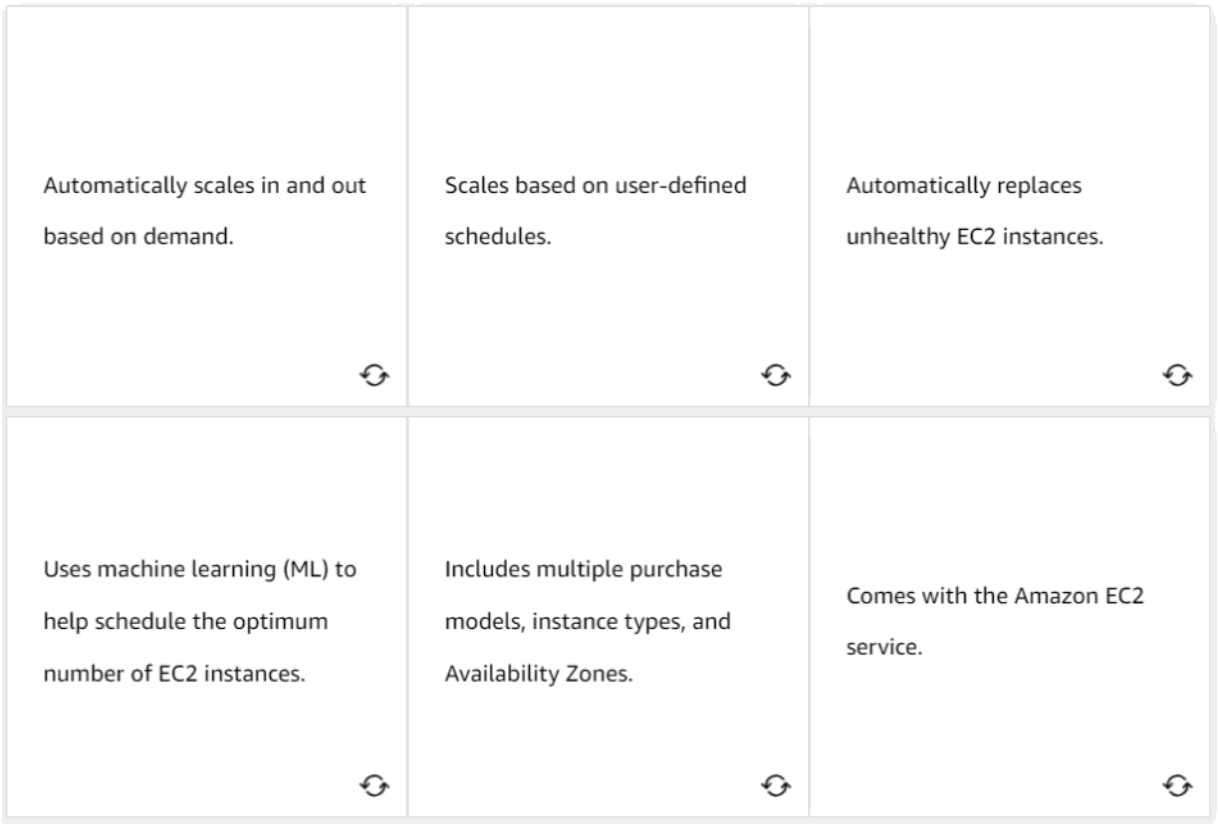
The cloud works differently with a pay-as-you-go model. You must turn off the unused services, especially EC2 instances you pay for on-demand. You can manually add and remove servers at a predicted time. But with unusual spikes in traffic, this solution leads to a waste of resources with over-provisioning or a loss of customers because of under-provisioning.

The need here is for a tool that automatically adds and removes EC2 instances according to conditions you define. That's exactly what the Amazon EC2 Auto Scaling service does.

Amazon EC2 Auto Scaling features

The Amazon EC2 Auto Scaling service adds and removes capacity to keep a steady and predictable performance at the lowest possible cost. By adjusting the capacity to exactly what your application uses, you only pay for what your application needs. This means Amazon EC2 Auto Scaling helps scale your infrastructure and ensure high availability.
To learn more about auto scaling features, flip each of the following flashcards by choosing them.





ELB with Amazon EC2 Auto Scaling

Additionally, the ELB service integrates seamlessly with Amazon EC2 Auto Scaling. As soon as a new EC2 instance is added to or removed from the Amazon EC2 Auto Scaling group, ELB is notified. However, before ELB can send traffic to a new EC2 instance, it needs to validate that the application running on the EC2 instance is available.

This validation is done by way of the ELB health checks feature you learned about in the previous lesson.

Configure Amazon EC2 Auto Scaling components

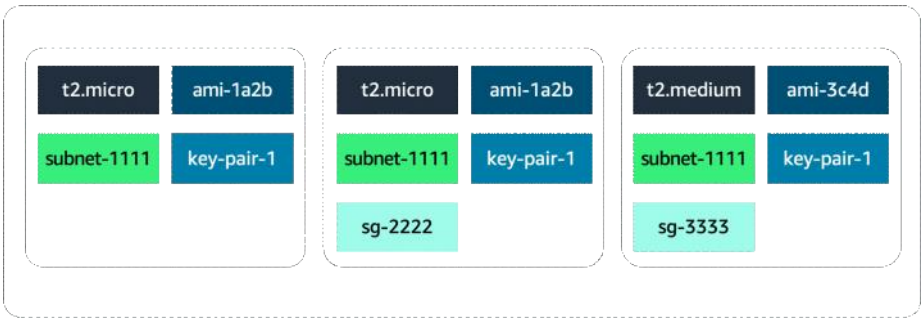
There are three main components of Amazon EC2 Auto Scaling. Each of these components addresses one main question as follows:

- bullet
  Launch template or configuration: Which resources should be automatically scaled?
- bullet
  Amazon EC2 Auto Scaling groups: Where should the resources be deployed?
- bullet
  Scaling policies: When should the resources be added or removed?

Launch templates and configurations

Multiple parameters are required to create EC2 instances—Amazon Machine Image (AMI) ID, instance type, security group, additional Amazon EBS volumes, and more. All this information is also required by Amazon EC2 Auto Scaling to create the EC2 instance on your behalf when there is a need to scale. This information is stored in a launch template.

You can use a launch template to manually launch an EC2 instance or for use with Amazon EC2 Auto Scaling. It also supports versioning, which can be used for quickly rolling back if there's an issue or a need to specify a default version of the template. This way, while iterating on a new version, other users can continue launching EC2 instances using the default version until you make the necessary changes.



A launch template specifies instance configuration information, such as the ID of the AMI, instance type, and security groups. You can have multiple versions of a launch template with a subset of the full parameters.

You can create a launch template in one of three ways as follows:

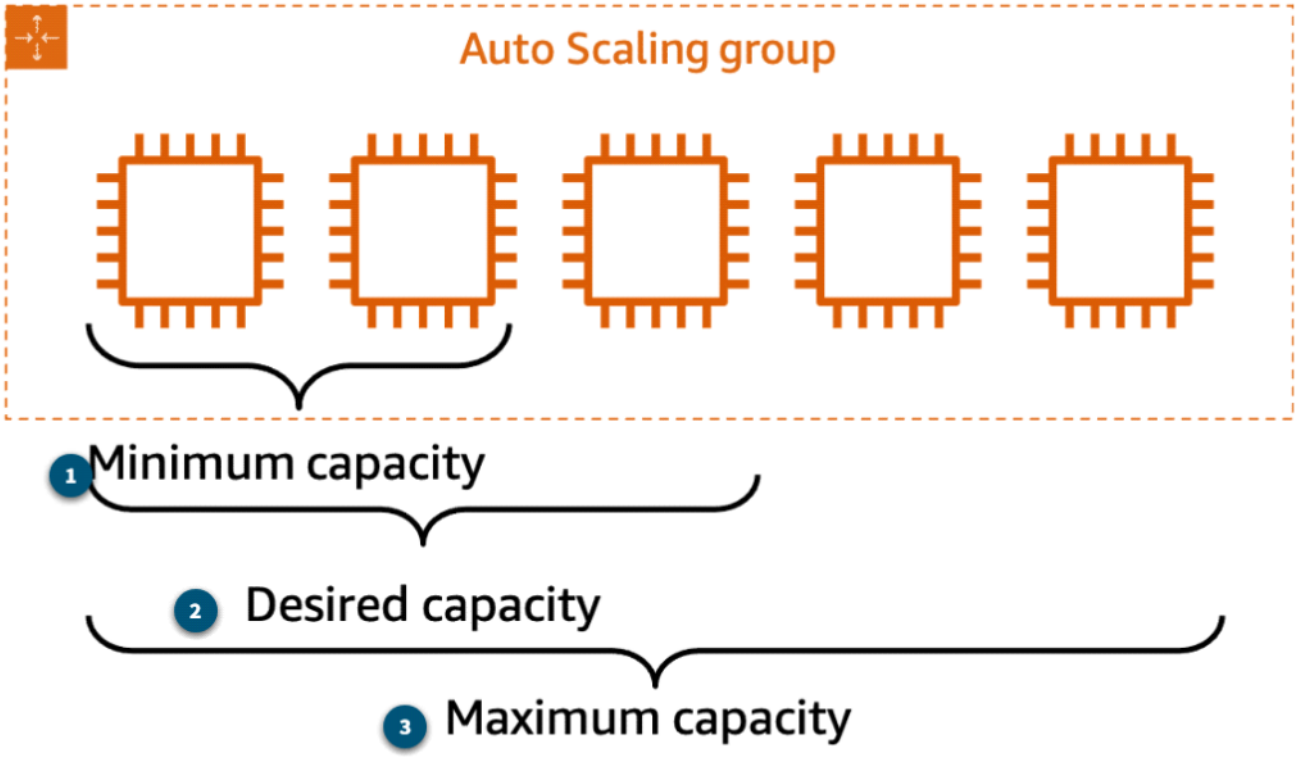
- bullet
  Use an existing EC2 instance. All the settings are already defined.
- bullet
  Create one from an already existing template or a previous version of a launch template.
- bullet
  Create a template from scratch. These parameters will need to be defined: AMI ID, instance type, key pair, security group, storage, and resource tags.

Another way to define what Amazon EC2 Auto Scaling needs to scale is by using a launch configuration. It's similar to the launch template, but you cannot use a previously created launch configuration as a template. You also cannot create a template from an already existing EC2 instance. For these reasons, and to ensure that you get the latest features from Amazon EC2, AWS recommends you use a launch template instead of a launch configuration.

Amazon EC2 Auto Scaling groups

The next component Amazon EC2 Auto Scaling needs is an Amazon EC2 Auto Scaling group. An Auto Scaling group helps you define where Amazon EC2 Auto Scaling deploys your resources. This is where you specify the Amazon Virtual Private Cloud (Amazon VPC) and subnets the EC2 instance should be launched in. Amazon EC2 Auto Scaling takes care of creating the EC2 instances across the subnets, so select at least two subnets that are across different Availability Zones.

With Auto Scaling groups, you can specify the type of purchase for the EC2 instances. You can use On-Demand Instances or Spot Instances. You can also use a combination of the two, which means you can take advantage of Spot Instances with minimal administrative overhead.
To specify how many instances Amazon EC2 Auto Scaling should launch, you have three capacity settings to configure for the group size.



Scaling policies

By default, an Auto Scaling group will be kept to its initial desired capacity. While it's possible to manually change the desired capacity, you can also use scaling policies.

In the Monitoring lesson, you learned about CloudWatch metrics and alarms. You use metrics to keep information about different attributes of your EC2 instance, such as the CPU percentage. You use alarms to specify an action when a threshold is reached. Metrics and alarms are what scaling policies use to know when to act. For example, you can set up an alarm that states when the CPU utilization is above 70 percent across the entire fleet of EC2 instances. It will then invoke a scaling policy to add an EC2 instance.

Three types of scaling policies are available: simple, step, and target tracking scaling. To learn about a category, choose the appropriate tab.

SIMPLE SCALING POLICY

With a simple scaling policy, you can do exactly what's described in this module. You use a CloudWatch alarm and specify what to do when it is invoked. This can include adding or removing a number of EC2 instances or specifying a number of instances to set the desired capacity to. You can specify a percentage of the group instead of using a number of EC2 instances, which makes the group grow or shrink more quickly.

After the scaling policy is invoked, it enters a cooldown period before taking any other action. This is important because it takes time for the EC2 instances to start, and the CloudWatch alarm might still be invoked while the EC2 instance is booting. For example, you might decide to add an EC2 instance if the CPU utilization across all instances is above 65 percent. You don't want to add more instances until that new EC2 instance is accepting traffic. However, what if the CPU utilization is now above 85 percent across the Auto Scaling group?

Adding one instance might not be the right move. Instead, you might want to add another step in your scaling policy. Unfortunately, a simple scaling policy can't help with that. This is where a step scaling policy helps.

STEP SCALING POLICY

Step scaling policies respond to additional alarms even when a scaling activity or health check replacement is in progress.

Similar to the previous example, you might decide to add two more instances when CPU utilization is at 85 percent and four more instances when it's at 95 percent.

Deciding when to add and remove instances based on CloudWatch alarms might seem like a difficult task. This is why the third type of scaling policy exists—target tracking.

TARGET TRACKING SCALING POLICY

If your application scales based on average CPU utilization, average network utilization (in or out), or request count, then this scaling policy type is the one to use. All you need to provide is the target value to track, and it automatically creates the required CloudWatch alarms.