Prepared by:

Shaik Moulali ,

AWS Cloud & DevOps Engineer,

moula.cloud5@gmail.com.

Amazon Web Services (AWS) Zero To Hero

Amazon Virtual Private Cloud:

Amazon VPC is your network environment in the cloud. With Amazon VPC, you can launch AWS resources into a virtual network that you have defined. VPCs deploy into one of the AWS Regions and can host resources from any Availability Zone within its Region. You can use both IPv4 and IPv6 in your VPC for secure and easy access to resources and applications. A VPC is a virtual network dedicated to your AWS account.

Subnets

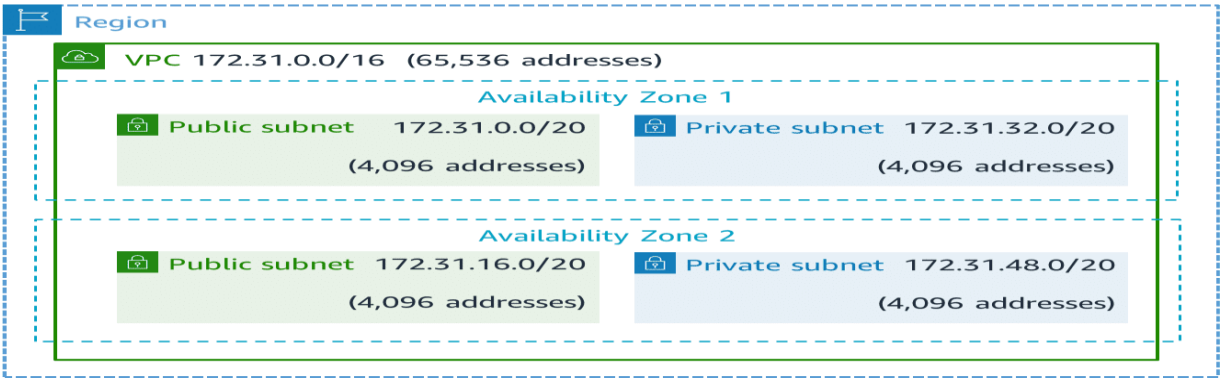
A subnet is a range of IP addresses in your VPC. You can launch AWS resources into a specified subnet. Use a public subnet for resources that must be connected to the internet and a private subnet for resources that won't be connected to the internet. A subnet resides within one Availability Zone. AWS reserves the first four IP addresses and the last IP address in each subnet CIDR block. Consider larger subnets over smaller ones (/24 and larger). You are less likely to waste or run out of IPs if you distribute your workload into larger subnets.

Example

In this example, the VPC CIDR block 172.31.0.0/16 allows a total of 65,536 IP addresses. AWS reserves five IP addresses per subnet so the actual usable addresses are 65,531.

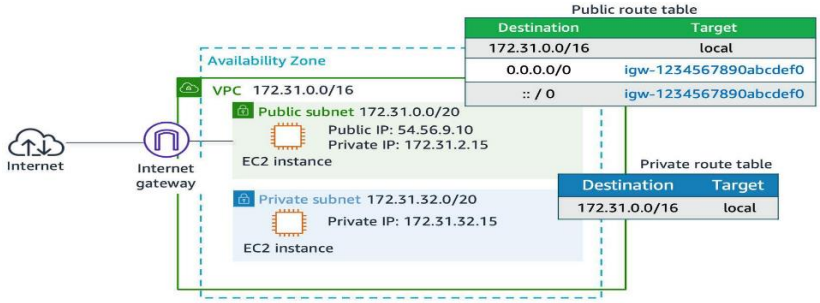
These are divided between its four subnets as follows:

- Public subnet 1: 172.31.0.0/20 allows 4,096 addresses between 172.31.0.1 - 172.31.15.254
- Public subnet 2: 172.31.16.0/20 allows 4,096 addresses between 172.31.16.1 - 172.31.31.254
- Private subnet 1: 172.31.32.0/20 allows 4,096 addresses between 172.31.32.1 - 172.31.47.254
- Private subnet 2: 172.31.48.0/20 allows 4,096 addresses between 172.31.48.1 - 172.31.63.254



Route tables:

A route table contains a set of rules (routes) that are used to determine where network traffic is directed. When you create a VPC, it automatically has a main route table. Initially, the main route table (and every route table in a VPC) contains only a single route: This is a local route that permits communication for all the resources within the VPC. You can't modify the local route in a route table. Whenever you launch an instance in the VPC, the local route automatically covers that instance. You can create additional custom route tables for your VPC.



Public subnets:

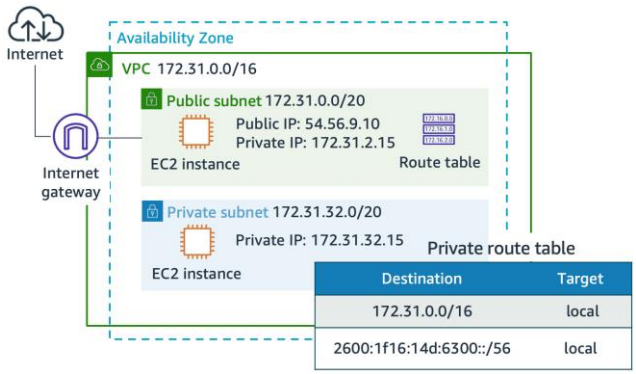
A public subnet is associated with a route table that has a route to an internet gateway. It lets you reach resources inside that subnet from the public internet by assigning public IP addresses. Your public subnet configuration acts as a two-way door—allowing traffic to flow either direction, invited or not invited.

Public subnets use the following:

- Internet gateways allow communication between resources in your VPC and the internet.
- Route tables are set of rules that the VPC uses to route network traffic. In a public subnet, it includes a route to the internet gateway.
- Public IP addresses can be reached from the internet.
- Private IP addresses are only reachable on the network.

Private subnets:

Private subnets allow indirect access to the internet. Traffic stays within your private network. A private IP address assigned to an EC2 instance will never change unless you manually assign a new IP address on the network interface of the EC2 instance. While you can put web-tier instances into a public subnet, we recommend that you put web-tier instances inside private subnets behind a load balancer placed in a public subnet. Elastic Load Balancing (ELB) is discussed later in this course.



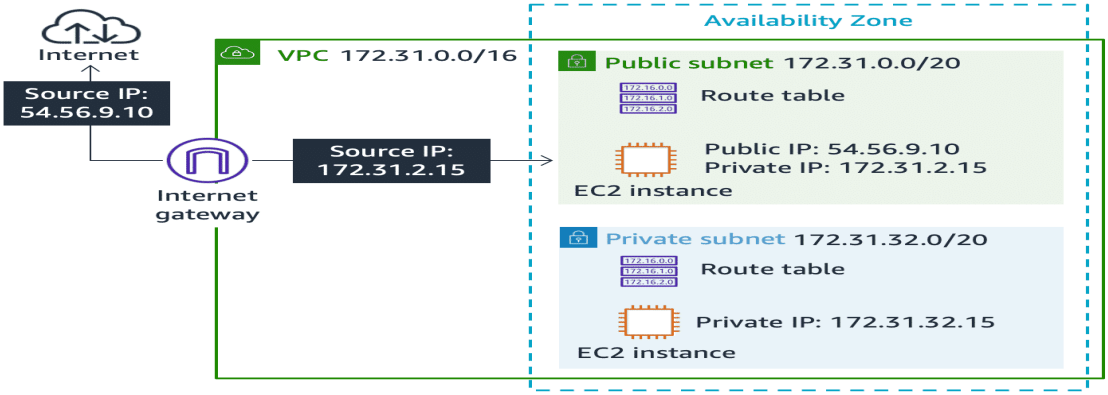
Internet Gateways:

An internet gateway is a horizontally scaled, redundant, and highly available VPC component that permits communication between instances in your VPC and the internet. It imposes no availability risks or bandwidth constraints on your network traffic. An internet gateway supports IPv4 and IPv6 traffic.

An internet gateway serves two purposes:

1. Provides a target in your route table for internet-routable traffic.
2. Protects IP addresses on your network by performing network address translation (NAT).

An internet gateway performs network address translation (NAT) by mapping public and private IP addresses. In this example, the internet gateway translates the source IP address of a request from the private IP address used on the network (172.31.2.15), to the public IP address (54.56.9.10). The recipient directs its response to the public IP address. The internet gateway receives the response and translates the public IP address to the matching private IP address. The VPC routes the response to the requester.



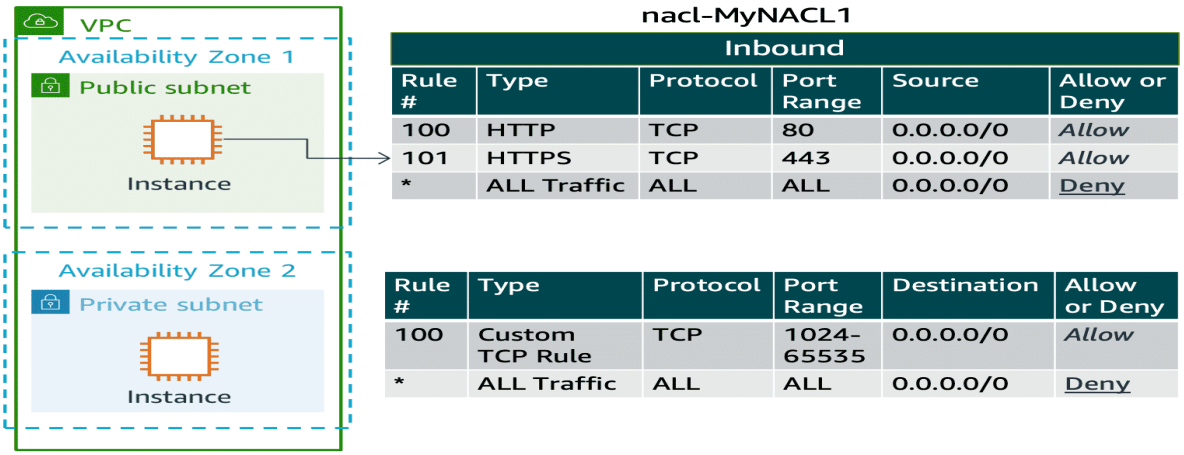
Network Access Control Lists (NACLs)

A network ACL is an optional layer of security for your VPC that acts as a firewall for controlling traffic in and out of one or more subnets. Every VPC automatically comes with a default network ACL. It allows all inbound and outbound IPv4 traffic.

Network ACLs are **stateless**, which means that responses to allowed inbound traffic are subject to the rules for outbound traffic (and the other way around).

Inbound and outbound IPv4 traffic

- A network ACL contains a numbered list of rules. We evaluate the rules in order, starting with the lowest numbered rule, to determine whether traffic is allowed in or out of any subnet associated with the network ACL.
- Every VPC automatically comes with a modifiable default network ACL. By default, it allows all inbound and outbound IPv4 traffic. You can create a custom network ACL and associate it with a subnet. By default, custom network ACLs deny all inbound and outbound traffic, until you add rules.
- Each network ACL includes a rule whose rule number is an asterisk. This rule ensures that if a packet doesn't match any of the other numbered rules, it is denied. You can't modify or remove this rule.



nacl-MyNACL1

Inbound

Rule #	Type	Protocol	Port Range	Source	Allow or Deny
100	HTTP	TCP	80	0.0.0.0/0	*Allow*
101	HTTPS	TCP	443	0.0.0.0/0	*Allow*
*	ALL Traffic	ALL	ALL	0.0.0.0/0	Deny

Rule #	Type	Protocol	Port Range	Destination	Allow or Deny
100	Custom TCP Rule	TCP	1024-65535	0.0.0.0/0	*Allow*
*	ALL Traffic	ALL	ALL	0.0.0.0/0	Deny

Network ACL rules

Each network ACL includes a rule whose rule number is an asterisk. This rule ensures that if a packet doesn't match any of the other numbered rules, it is denied. You can't modify or remove this rule. Components of a network ACL rule include:

- Rule number – Rules are evaluated starting with the lowest numbered rule. As soon as a rule matches traffic, it is applied regardless of any higher-numbered rule that might contradict it.
- Type – The type of traffic; for example, Secure Shell, or SSH. You can also specify all traffic or a custom range.
- Protocol – You can specify any protocol that has a standard protocol number.
- Port range – The listening port or port range for the traffic. For example, 80 for HTTP traffic.
- Source – For inbound rules only, the source of the traffic (CIDR range).
- Destination – For outbound rules only, the destination for the traffic (CIDR range).
- Allow or Deny – Whether to allow or deny the specified traffic.

Network ACLs are stateless, which means that responses to allowed inbound traffic are subject to the rules for outbound traffic (and vice versa).

Security Groups

A security group acts as a virtual firewall for your instance to control inbound and outbound traffic. Security groups act at the network interface level, it secure EC2 Instances not the subnet level, and they support **Allow** rules only.

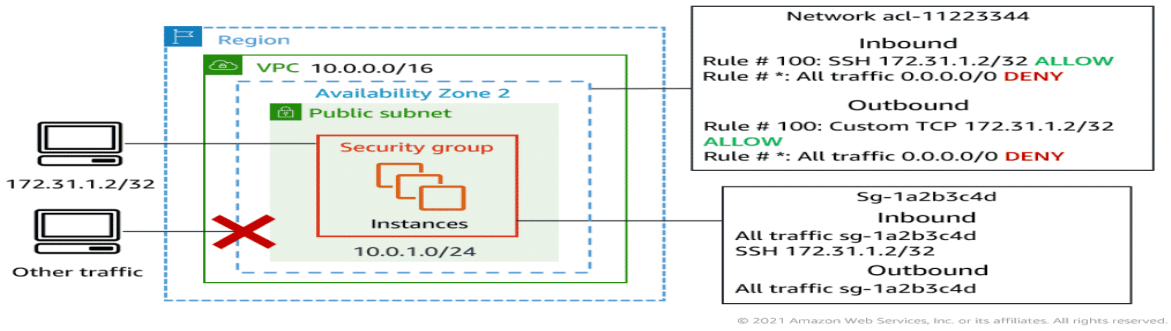
Comparing security groups and network ACLs

A security group acts as a firewall for associated EC2 instances, controlling both inbound and outbound traffic at the instance level. Network ACLs act as a firewall for associated subnets, controlling both inbound and outbound traffic at the subnet level. A network ACL denies communication by default. Network ACL rule order matters.

Security Group	Network ACL
Security groups act as a firewall for associated EC2 instances, and are associated with elastic network interface implemented by a hypervisor	Network ACLs act as a firewall for associated subnets
Controls inbound and outbound traffic at instance level	Controls inbound and outbound traffic at subnet level
Supports Allow rules only	Supports both Allow and Deny rules
A stateful firewall	A stateless firewall
Needs to be manually assigned to instances	Automatically applied when instances are added to subnet

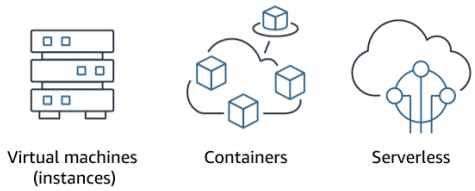
Network ACL Use Case

The network ACL controls access to instances in a subnet and acts as a backup layer of defense. Network ACL rules apply to all instances in the subnet.



In the example to the left, instances in your subnet can communicate with each other and are accessible from a trusted remote computer. The remote computer might be a computer in your local network or an instance in a different subnet or VPC. You use it to connect to your instances to perform administrative tasks.

Compute as a Service:
At a fundamental level, three types of compute options are available: virtual machines (VMs), container services, and serverless.



When designing your architecture on AWS, you'll need compute to run your servers in the cloud. AWS offers multiple compute options. First, you need to know which compute service to use for each use case.

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.

Beneath the surface, some AWS compute services use Amazon EC2 or use virtualization concepts. You should understand this service before advancing to container services and serverless compute.

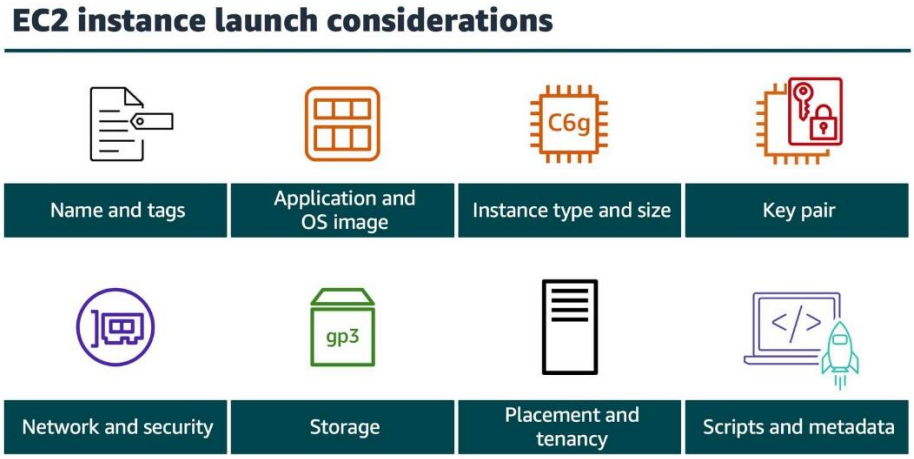
Amazon Elastic Compute Cloud (Amazon EC2)

Amazon EC2 is the service you use to create and run virtual machines. Your virtual machines in the AWS Cloud are called EC2 instances.

Amazon EC2 is just like your traditional on-premises server, but it is available in the cloud. It can support workloads such as web hosting, applications, databases, authentication services, and anything else a server can support.

On AWS, servers, databases, storage, and higher-level application components can be instantiated within seconds. You can treat these as temporary and disposable resources, free from the constraints of a fixed IT infrastructure. Elastic cloud computing redefines the way you approach change management, testing, reliability, and capacity planning.

There are many options you need to consider before you start working with EC2 instances



Amazon Machine Image (AMI)

An **Amazon Machine Image (AMI)** provides the information required to launch an instance, which is a virtual server in the cloud. You can launch multiple instances from a single AMI when you need

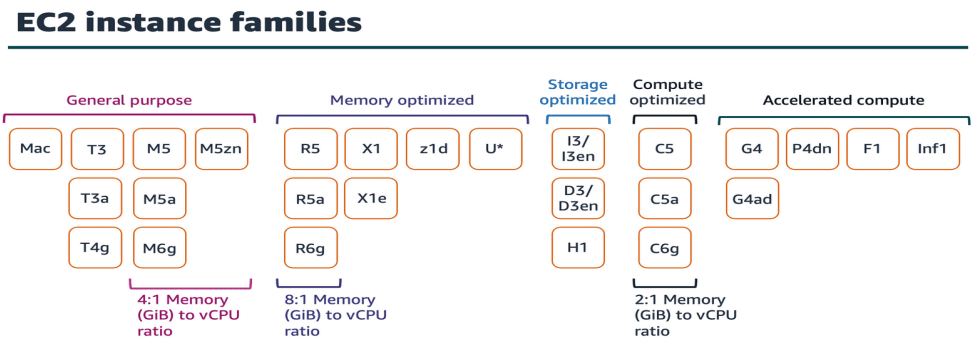
multiple instances with the same configuration. You can use different AMIs to launch instances when you need instances with different configurations. Specify a source AMI when you launch an instance.

An AMI includes the following:

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

To create a custom AMI, launch an instance and customize it to meet your requirements. Then, save that configuration as a custom AMI. Instances launched from this custom AMI will use all of your customizations. These custom AMIs can also be published for internal, private, or external public use.

EC2 Instance Families



Pick the optimal instance family for the type of workload you plan to deploy. Doing this saves time and cost, and reduces the need to resize later. Some instance types are only available in certain Regions.

General purpose

- Balance of compute, memory and networking
- Diverse workloads
- Web applications

Compute optimized

- Compute-bound applications
- High-performance processors
- Media transcoding
- Scientific modelling
- Machine learning

Memory optimized

- Fast delivery of large datasets in memory

- Database servers
- Web caches
- Data analytics

Accelerated computing

- High-graphics processing
- GPU bound
- Machine learning
- High performance computing (HPC)
- Autonomous vehicles

Storage optimized

- High sequential read/write
- Large datasets
- NoSQL databases
- Amazon OpenSearch Service

High Performance Computing (HPC) optimized

- Purpose-built
- Compute-intensive high performance computing (HPC) workloads
- Workloads at scale

Amazon EC2 keypair

A key pair, consisting of a private key and a public key, is a set of security credentials that you use to prove your identity when connecting to an instance. Amazon EC2 stores the public key and you store the private key. You use the private key instead of a password to securely access your instances. Anyone who possesses your private keys can connect to your instances, so it's important that you store your private keys in a secure place.

Tenancy

AWS compute tenancy refers to how EC2 instances are distributed across the underlying physical hardware, with three main options - shared tenancy, dedicated instance, or dedicated host. The choice of tenancy depends on your specific requirements around compliance, licensing, and cost optimization.

- By default, EC2 instances have shared tenancy, meaning multiple AWS accounts might share the same physical hardware.
- Dedicated Instances are EC2 instances that are physically isolated at the host hardware level from instances that aren't dedicated and from instances that belong to other AWS accounts.

- When you launch instances on a Dedicated Host, the instances run on a physical server with EC2 instance capacity fully dedicated to your use. You are provided an isolated server with configurations that you can control.
- With Dedicated Hosts, you have the option to allow AWS to automatically select a server to place your instance. Or you can manually select a dedicated server to place your instance.

Container Services

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

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.

Managing containers with Amazon ECS

Amazon ECS is an end-to-end container orchestration service that helps you spin up new containers. With Amazon ECS, your containers are defined in a task definition that you use to run an individual task or a task within a service. You have the option to run your tasks and services on a serverless infrastructure that's managed by another AWS service called AWS Fargate. Alternatively, for more control over your infrastructure, you can run your tasks and services on a cluster of EC2 instances that you manage.

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. An ECS-optimized AMI is also available with the agent pre-installed.

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

- Launching and stopping containers.
- Getting cluster state.

- Scaling in and out.
- Scheduling the placement of containers across your cluster.
- Assigning permissions.
- Meeting availability requirements.

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:

- 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.
- An ECS container is called a task. An EKS container is called a pod.
- 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.

Serverless

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.

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:

- There are no servers to provision or manage.
- It scales with usage.
- You never pay for idle resources.
- Availability and fault tolerance are built in.

AWS Fargate

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 Lambda

If you want to deploy your workloads and applications without having to manage any EC2 instances or containers, you can use Lambda.

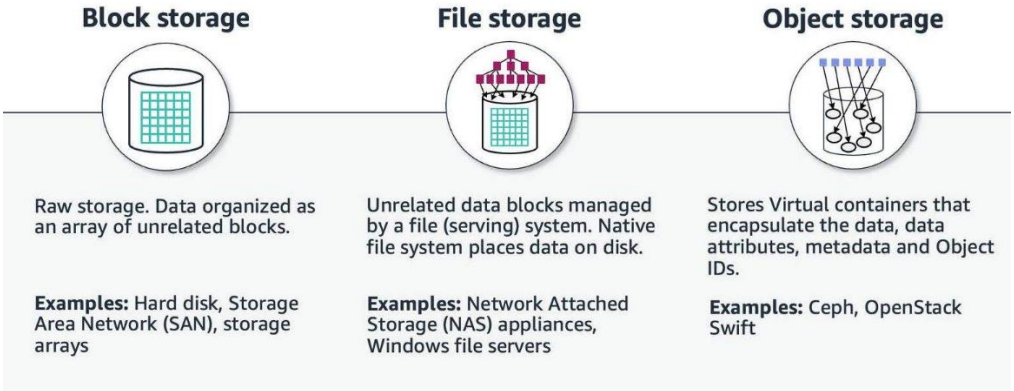
With Lambda, you can run code without provisioning or managing servers. You can run code for virtually any type of application or backend service. This includes data processing, real-time stream processing, machine learning, WebSockets, IoT backends, mobile backends, and web applications like your employee directory application!

Lambda runs your code on a high availability compute infrastructure and requires no administration from the user. You upload your source code in one of the languages that Lambda supports, and Lambda takes care of everything required to run and scale your code with high availability. There are no servers to manage. You get continuous scaling with subsecond metering and consistent performance.

Cloud Storage Overview

Block, object, and file storage represent three distinct approaches to data storage, each suited for different use cases.

To learn more on each storage type, choose each of the numbered markers.



Block Storage

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.

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.

SSD volumes

	General Purpose SSD volumes		**Provisioned IOPS SSD volumes**	
Volume type	gp3	gp2	io2 Block Express	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

	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	

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.

BS 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.

File Storage

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 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 use cases

Amazon S3 is a widely used storage service, with far more use cases than could fit on one screen.

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.

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 four tiers: a frequent access tier, an infrequent access tier, an archive instance access tier, and an archive 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 Express One Zone	Optimized for your most frequently accessed data and low-latency applications within a single Availability Zone, often used with S3 Directory Buckets for high-performance workloads.
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 re-creatable 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.

Database Services

AWS database services are purpose-built databases supporting a variety of workloads, including relational, key-value, document, graph, time-series and so on. Choosing the right database for your workload will give you best results in terms of high availability, low latency and extreme performance.

Relational and Non-relational Databases

For decades, the predominant data model that was used for application development was the relational data model used by relational databases such as Oracle, IBM DB2,Microsoft SQL Server, MySQL, and PostgreSQL. It wasn't until the mid to late 2000s that other data models began to gain significant adoption and usage. To differentiate and categorize these new classes of databases and data models, the term NoSQL was coined. Often the term NoSQL is used interchangeably with non-relational.

Relational databases

A relational database is a collection of data items with predefined relationships between them. These items are organized as a set of tables with columns and rows. Each column in a table holds a certain kind of data and a field stores the actual value of an attribute. The rows in the table represent a collection of related values of one object or entity. This data can be accessed in many different ways without reorganizing the database tables themselves.

Non-relational or NoSQL databases

NoSQL databases are purpose built for specific data models and have flexible schemas for building modern applications. NoSQL is a term used to describe non-relational database systems that are highly available, scalable, and optimized for high performance. NoSQL databases use alternate models for data management, such as key-value pairs or document storage.

Though there are many types of databases with varying features, this table shows some of the differences between SQL (relational) and NoSQL (non-relational) databases.

Relational Database	NoSQL Database

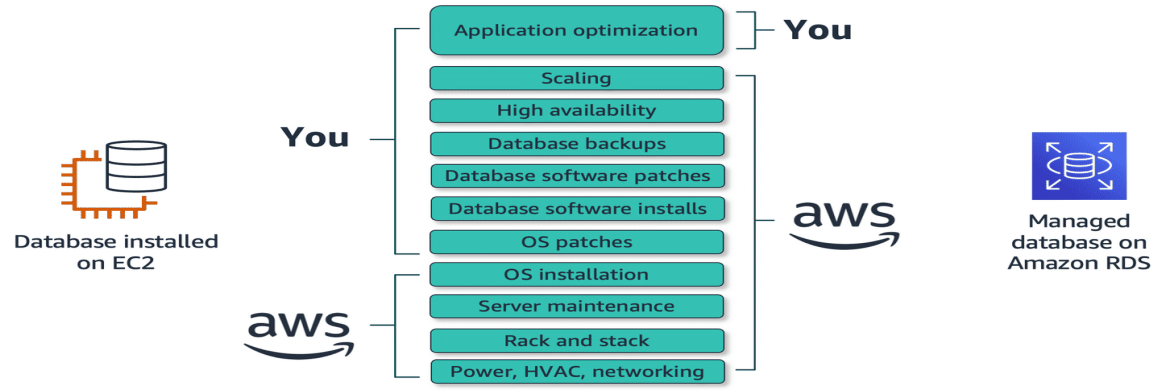
You require strict schema rules and data quality enforcement.	You need your database to scale horizontally.
Your database doesn't need extreme read/write capacity.	Your data does not lend itself well to traditional schemas.
If you have a relational data set that does not require extreme performance, a relational database management system can be the best, lowest effort solution.	Your read/write rates exceed those that can be economically supported through a traditional SQL database.

Managed and Unmanaged Services

When building resources in the cloud, you want to consider the level of control you need and the resources you have to manage that resource.

For example, when running a database in the cloud, you can install a database on an Amazon Elastic Compute Cloud (Amazon EC2) instance or you can choose a managed database option such as Amazon RDS.

To learn more, select each numbered marker.



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.

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.

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.

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.

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.

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.

Prepared by:

Shaik Moulali ,

AWS Cloud & DevOps Engineer,

Moula.cloud5@gmail.com.