**AWS Technical Essentials**

**Module 1. Introduction to AWS**

What is AWS?: Cloud computing is the on-demand delivery of IT resources with primarily pay-as-you-go pricing.

Cloud computing deployment models: On-premises, Cloud, Hybrid

Six advantages of cloud computing: pay-as-you-go, Benefit from massive economies of scale, Stop guessing capacity, Increase speed and agility, Realize cost savings, Go global in minute

AWS Global Infrastructure: In AWS, this physical infrastructure makes up the AWS Global Infrastructure, in the form of Regions and Availability Zones.

Choosing the right AWS Region: latency, service availability, data compliance

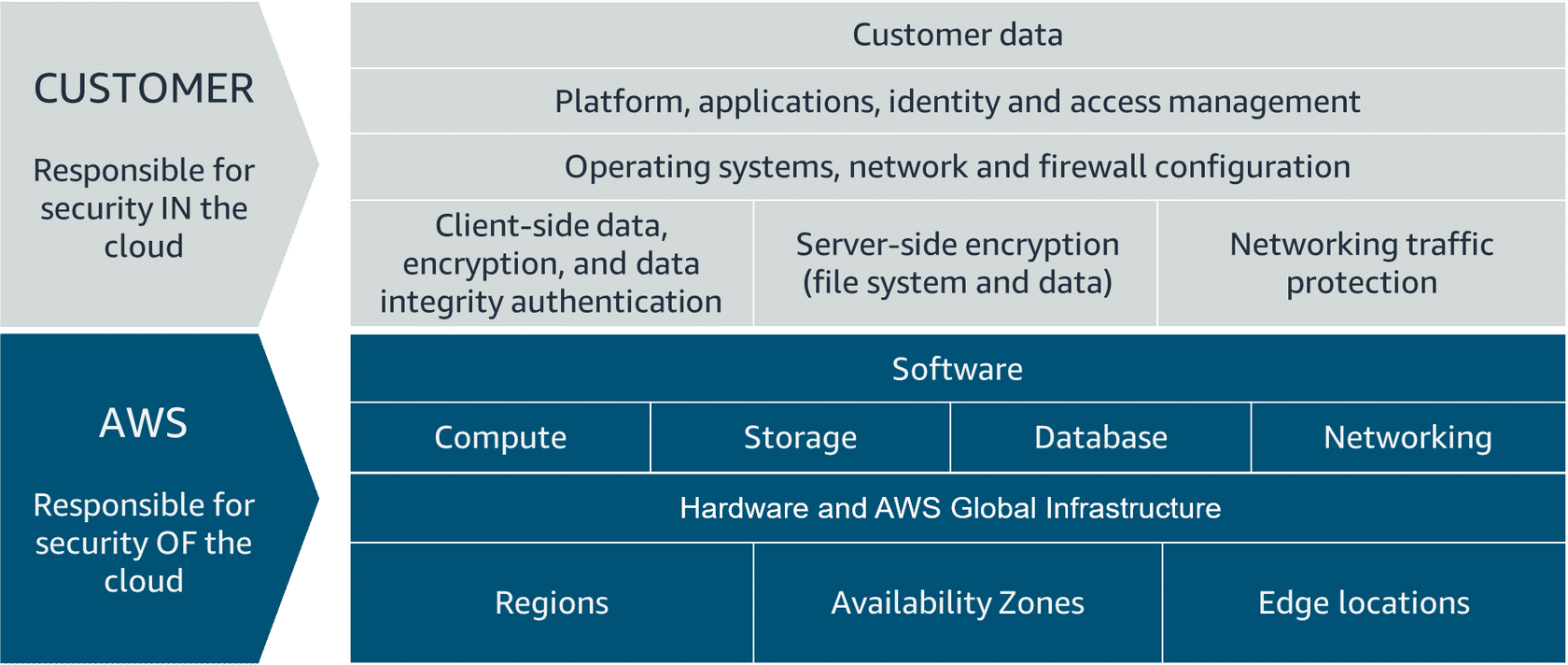
Scope of AWS services: Depending on the AWS service that you use, your resources are either deployed at the Availability Zone, Region, or Global level.

Maintaining resiliency: 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 are global locations where content is cached.

Interacting with AWS: Every action that you make in AWS is an API call that is authenticated and authorized. In AWS, you can make API calls to services and resources through the AWS Management Console, AWS Command Line Interface (AWS CLI), or AWS SDKs.

Security and the AWS Shared Responsibility Model: Security and compliance are a shared responsibility between AWS and you.



Protecting the 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.

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 and the Secret access key.

A screenshot of a computer screen

Description automatically generated

Multi-factor authentication: When you create an AWS account and first log in to the account, you use single-factor authentication. It only requires one authentication method. In this case, you use a user name and password to authenticate as the AWS root user. MFA requires two or more authentication methods to verify an identity.

AWS Identity and Access Management (IAM): Authentication answers the question, "Are you who you say you are?" Authorization answers the question, "What actions can you perform?". AWS Identity and Access Management (IAM) is an AWS service that helps you manage access to your AWS account and resources.

IAM features: Global, Integrated with AWS services, Shared access, MFA, Identity federation, Free to use.

IAM user: represents a person or service that interacts with AWS. You define the user in your AWS account.

IAM user credentials: An IAM user consists of a name and a set of credentials.

IAM group: is a collection of users. All users in the group inherit the permissions assigned to the group.

IAM policies: To manage access and provide permissions to AWS services and resources, you create IAM policies and attach them to an IAM identity.

IAM policy Example: provides admin access through an IAM identity-based policy.

{

"Version": "2012-10-17",

"Statement": [{

"Effect": "Allow",

"Action": "\*",

"Resource": "\*"

}]

}

IAM roles: is an identity that can be assumed by someone or something who needs temporary access to AWS credentials.

IAM best practices: Lock down the AWS root user, Follow the principle of least privilege, Use IAM appropriately, Use IAM roles when possible, Consider using an identity provider, Regularly review and remove unused users, roles, and other credentials

(Video Demo) Implementing Security with IAM: how to create a role and how to create a user

(Video Demo) Hosting the Employee Directory Application on AWS: how to create a EC2 instance in the VPC default network.

**Module 2. AWS Compute**

Compute as a Service:



Servers: power your application by providing CPU, memory, and networking capacity to process users’ requests and transform them into responses. HTTP servers include: Windows options, such as Internet Information Services (IIS), or Linux options, such as Apache HTTP Server, Nginx, and Apache Tomcat.

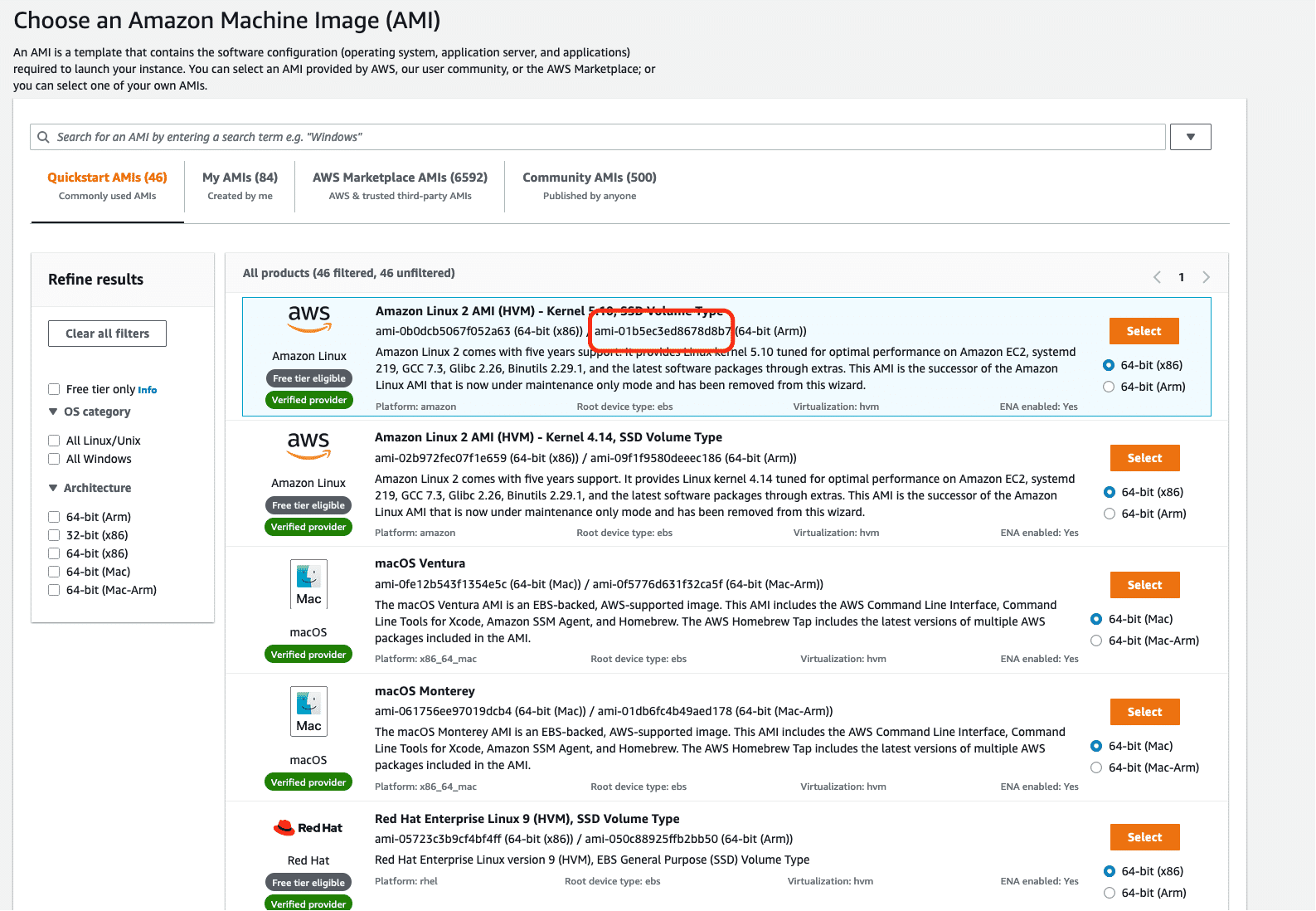
Getting started with EC2: When architecting any application for high availability, consider using at least two EC2 instances in two separate Availability Zones.

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.

Amazon Machine Image (AMI): 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.

Finding AMIs: Quick Start AMIs, AWS Marketplace AMIs, My AMIs, Community AMIs, Custom image.



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.

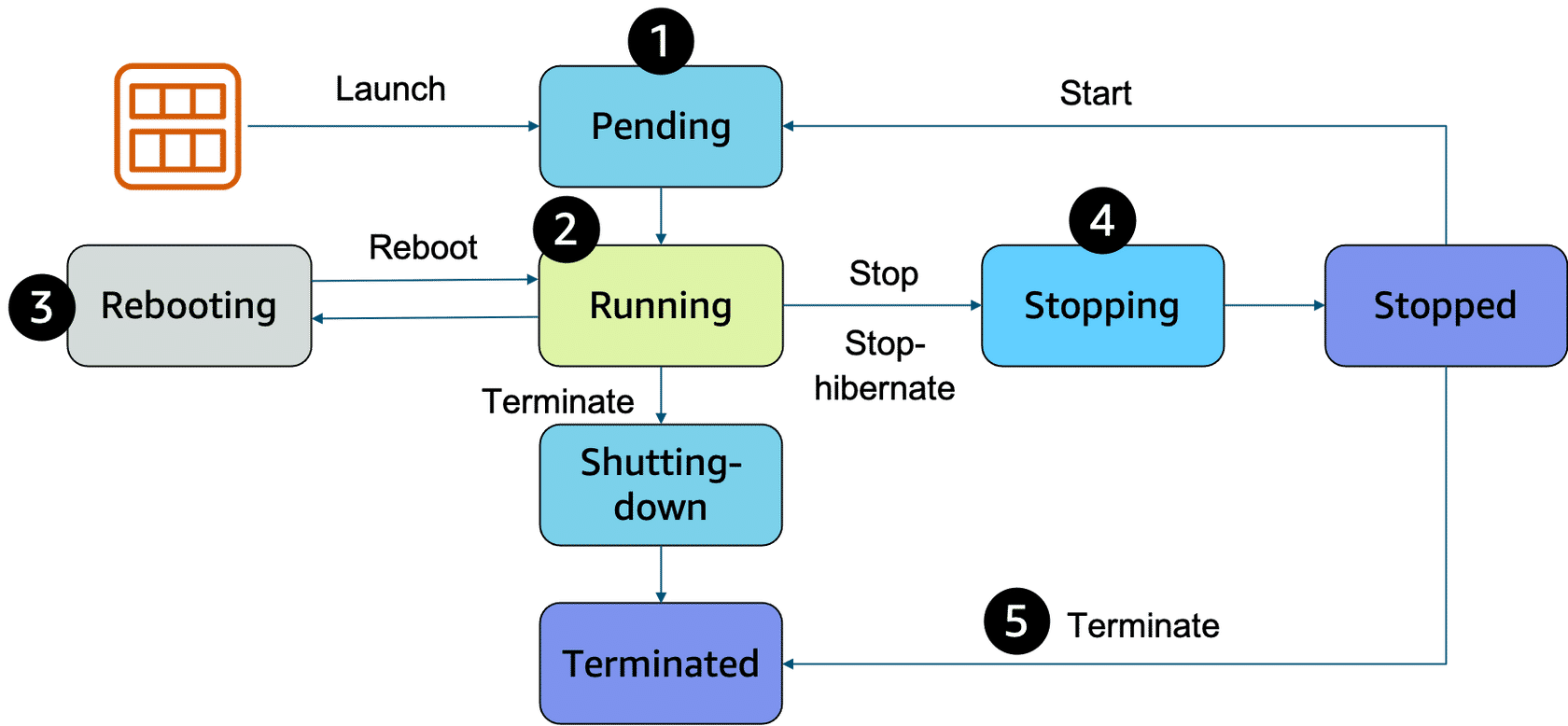
Amazon EC2 instance types: EC2 instances are a combination of virtual processors (vCPUs), memory, network, and, in some cases, instance storage and graphics processing units (GPUs). When you create an EC2 instance, you need to choose how much you need of each of these components.

Instance families: Each instance family is optimized to fit different use cases (General purpose, Compute optimized, Memory optimized, Accelerated computing, Storage optimized, HPC optimized)

EC2 instance locations: Unless otherwise specified, when you launch EC2 instances, they are placed in a default virtual private cloud (VPC).

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

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



Difference between stop and stop-hibernate:

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, spot instances, saving plans, reserved instances, dedicated hosts)

(Video Demo): launching the Employee Directory Application on Amazon EC2

Amazon Linux 2023 user data script:

#!/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-pip

pip install -r requirements.txt

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

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. An example of a containerization platform is Docker.

Difference between VMs and containers:

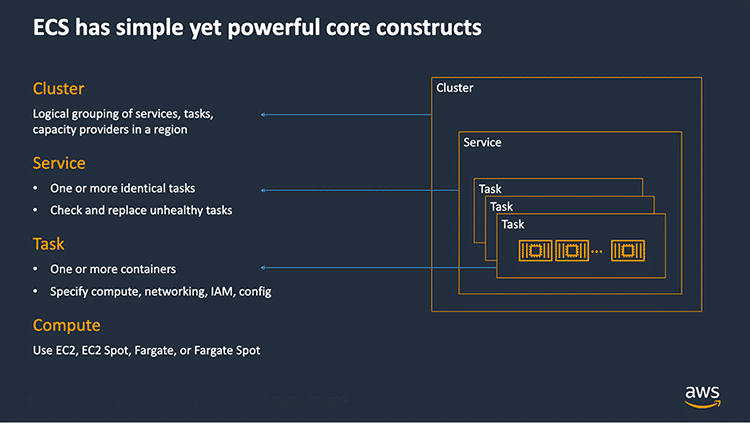
A screenshot of a diagram

Description automatically generated

Containers share the same operating system and kernel as the host that they exist on.

Orchestrating containers: AWS offers two container orchestration services: Amazon Elastic Container Service (Amazon ECS) and Amazon Elastic Kubernetes Service (Amazon EKS).

Managing containers with Amazon ECS: 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. ou have the option to run your tasks and services on a serverless infrastructure that's managed by another AWS service called AWS Fargate. 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.

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.

Introduction to Serverless: 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.

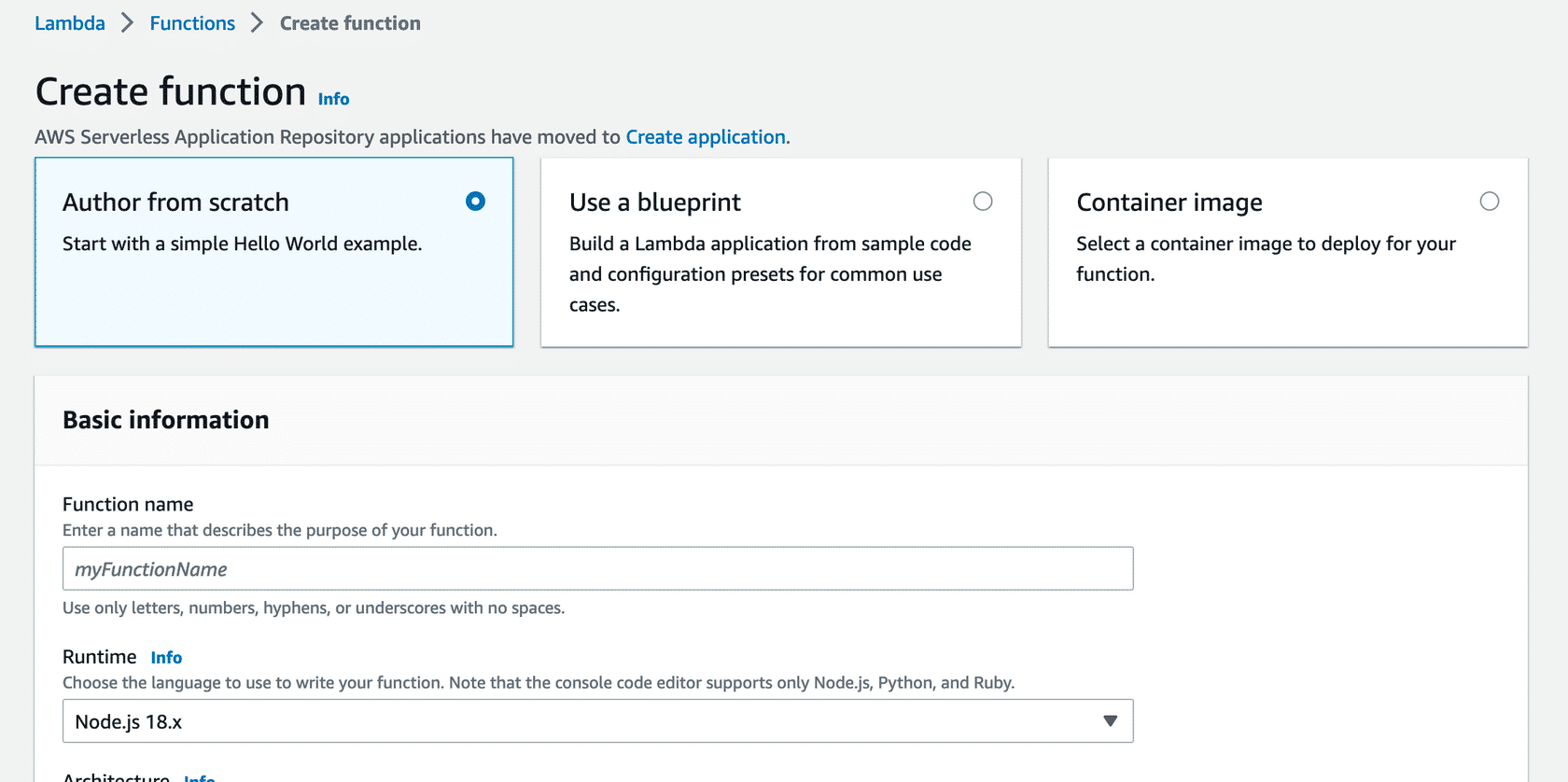
Serverless includes four aspects: There are no servers to provision or manage, It scales with usage, You never pay for idle resources, Availability and fault tolerance are built in.

Serverless with AWS Fargate: AWS Fargate is a purpose-built serverless compute engine for containers. Fargate abstracts the EC2 instance so that you’re not required to manage the underlying compute infrastructure.

Serverless with AWS Lambda: Running code on 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.

How Lambda works: Lambda concepts

- **Function** is a resource that you can invoke to run your code in Lambda



-**Triggers** describe when a Lambda function should run.

- **Event** is a JSON-formatted document that contains data for a Lambda function to process.

- **Application Environment** provides a secure and isolated runtime environment for your Lambda function. An application environment manages the processes and resources that are required to run the function.

- **Deployment package**:

**A .zip file archive** – This contains your function code and its dependencies. Lambda provides the operating system and runtime for your function.

**A container image** – You add your function code and dependencies to the image. You must also include the operating system and a Lambda runtime.

**Runtime** – You can use built-in runtimes, such as Python, Node.js, Ruby, Go, Java, or .NET Core.

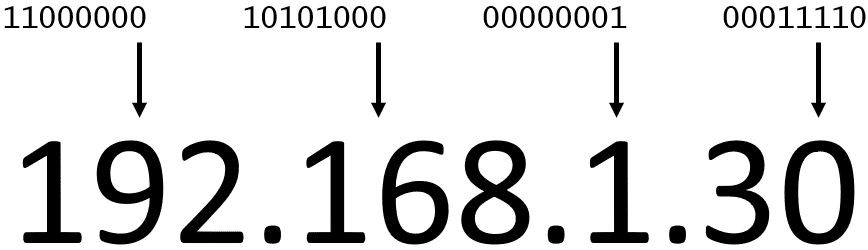
**Lambda function handler** is the method in your function code that processes events.

**Module 3. AWS Networking**

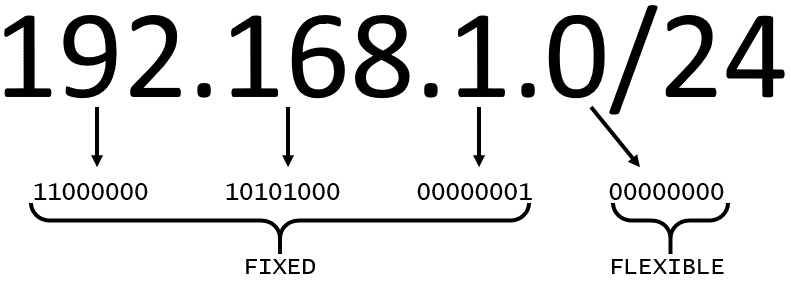
Networking: is how you connect computers around the world and allow them to communicate with one another. One example is the AWS Global Infrastructure. AWS has built a network of resources using data centers, Availability Zones, and Regions.

In the digital world, computers handle the delivery of messages in a similar way as sending a letter. This is called **routing**.

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.



Classless Inter-Domain Routing (**CIDR**): is a compressed way of representing a **range of IP addresses**



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.

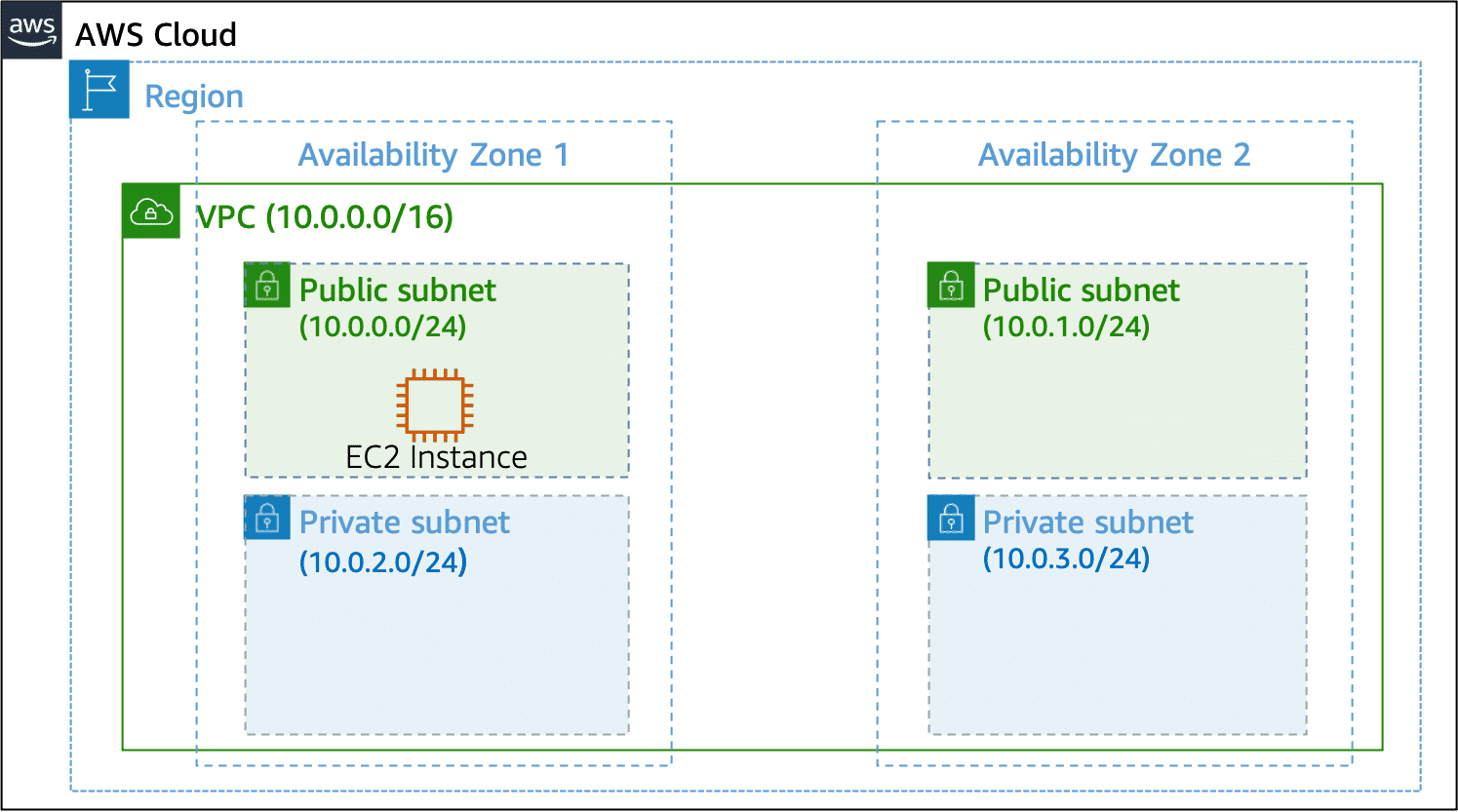
In AWS, the smallest IP range you can have is /28, which provides 16 IP addresses. The largest IP range you can have is a /16, which provides 65,536 IP addresses.

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: Name of the VPC, Region where the VPC will live – A VPC spans all the Availability Zones within the selected Region, IP range for the VPC in CIDR notation.

A screenshot of a computer

Description automatically generated

Subnets: as smaller networks inside your base network. 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.



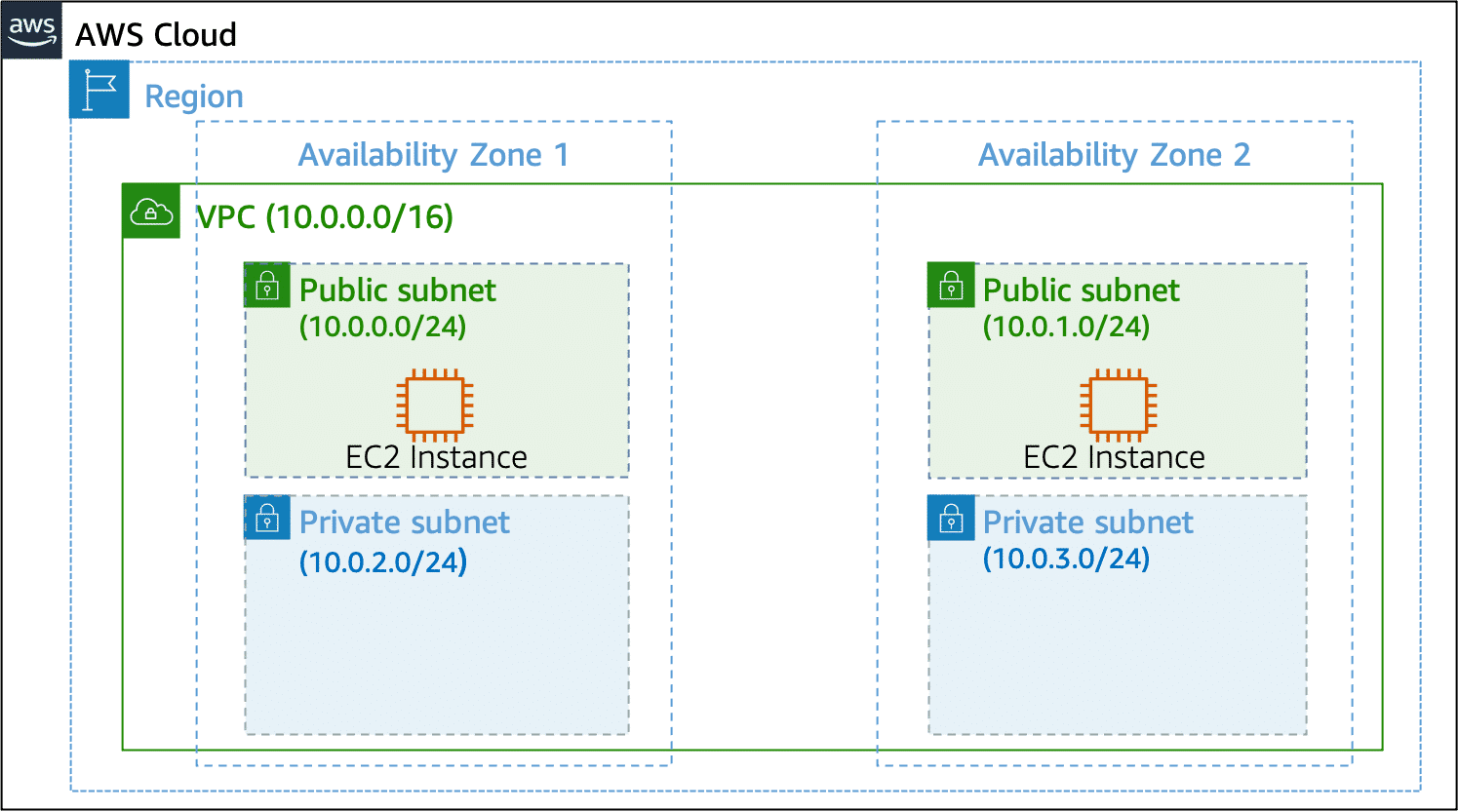
When you create a subnet, you must specify the following:

**VPC** that you want your subnet to live in—in this case: VPC (10.0.0.0/16)

**Availability Zone** that you want your subnet to live in—in this case: Availability Zone 1

**IPv4 CIDR** block for your subnet, which must be a subset of the VPC CIDR block—in this case: 10.0.0.0/24

High availability with a VPC:

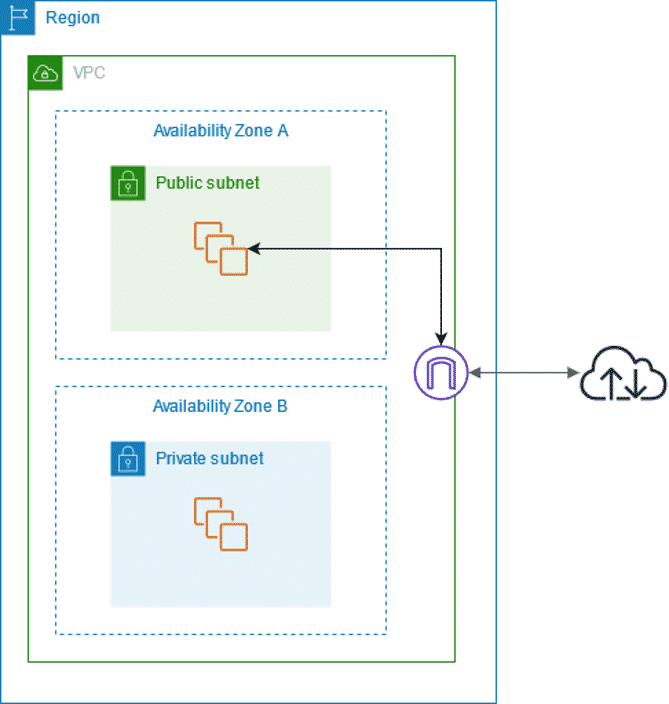


Reserved IPs:

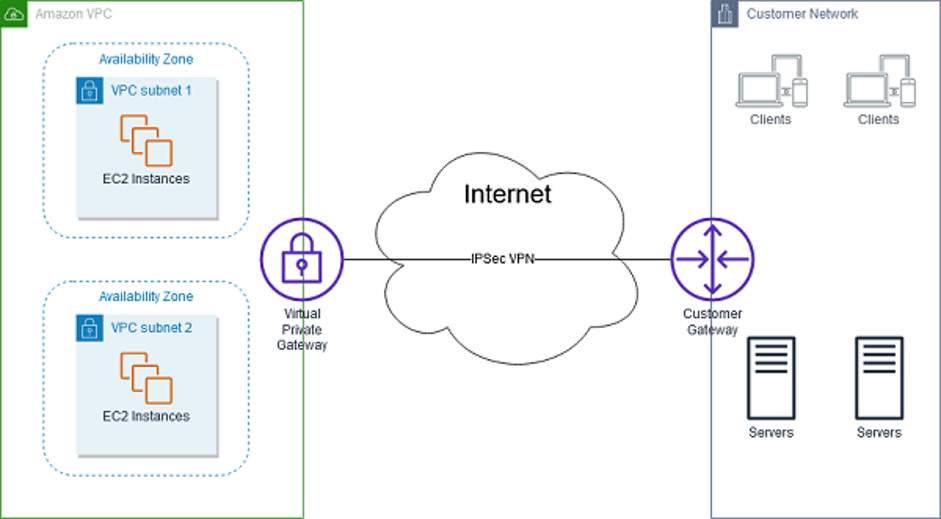
A table with text on it

Description automatically generated

Internet gateway:

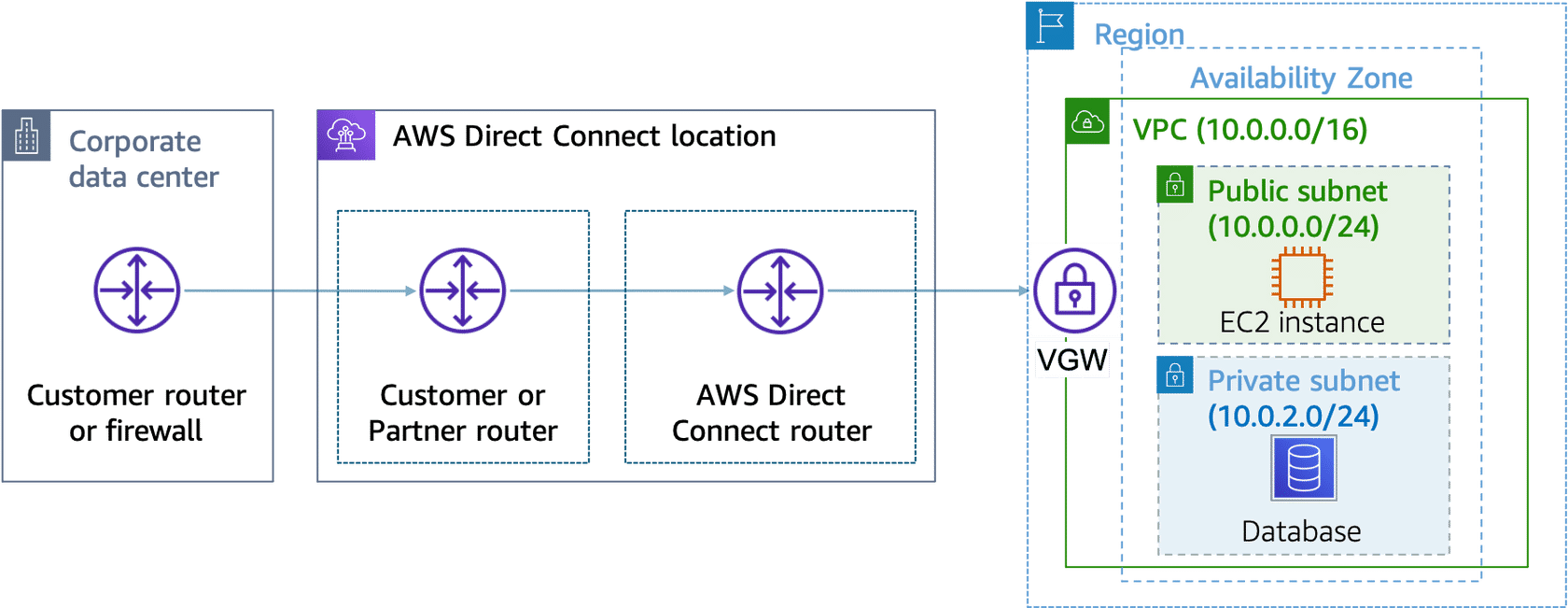


Virtual private gateway:



AWS Direct connect:

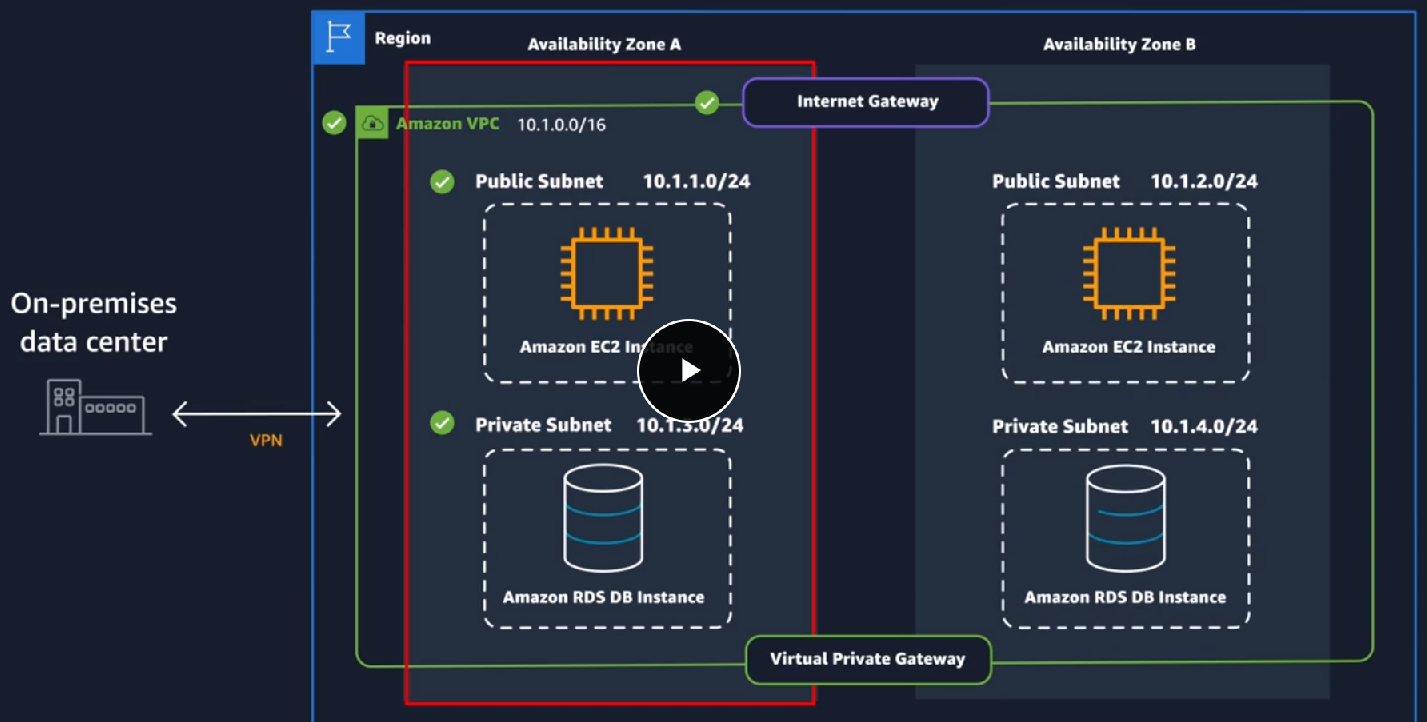
To establish a secure **physical** **connection** between your on-premises data center and your Amazon VPC, you can use AWS Direct Connect.



This is an example of VPC with four subnets

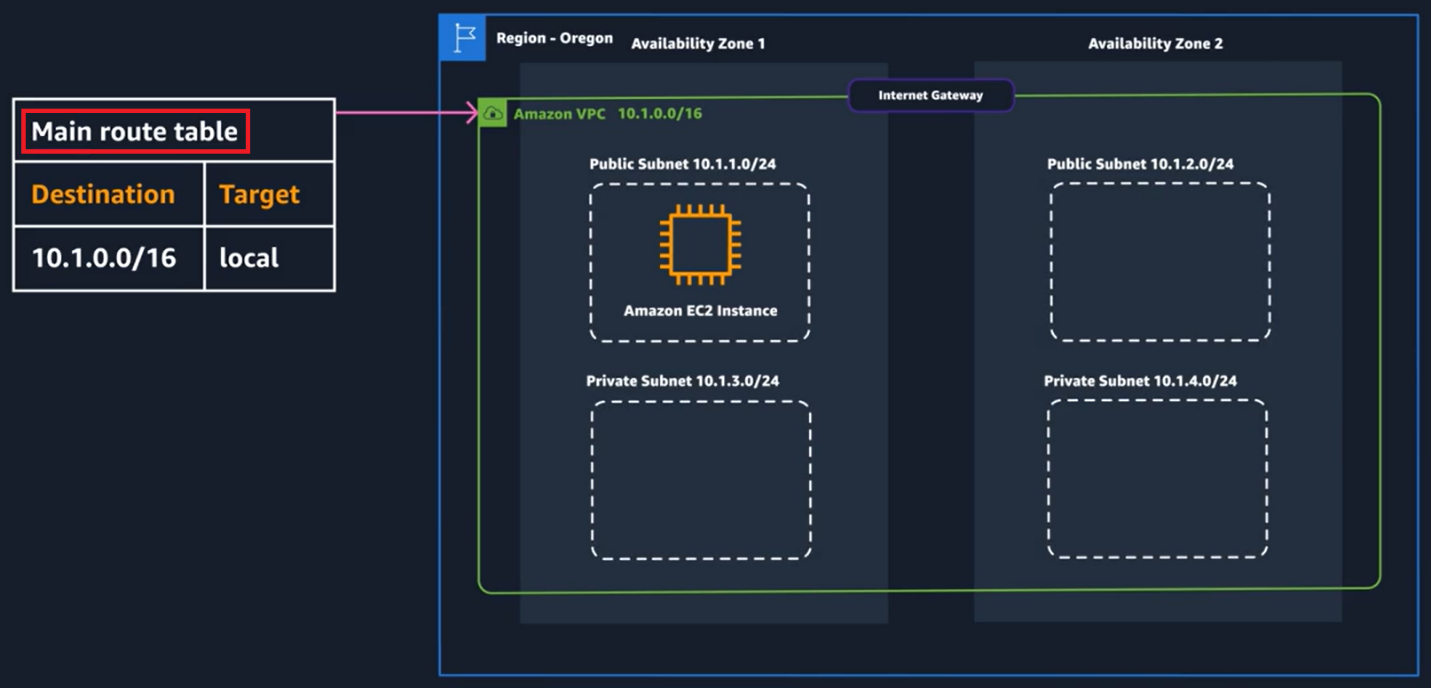
We attached a Internet Gateway to the VPC to allow internet access to Public Subnets

We attached a Virtual Private Gateway to access from the VPC to an On-premises data center

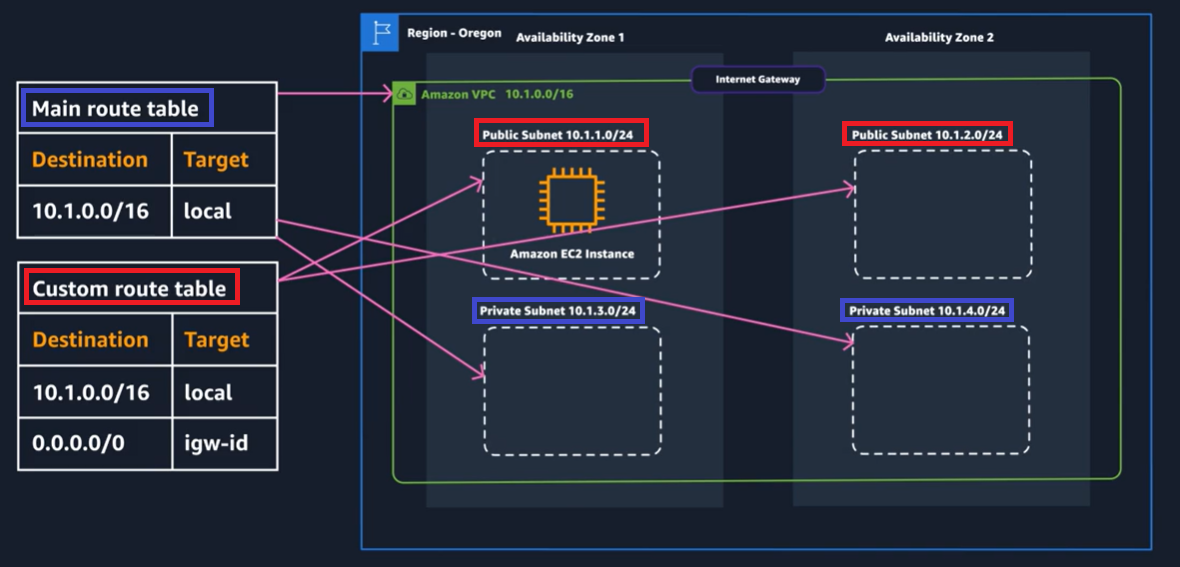


VPC Routing: A route table contains a set of rules, called routes, that determine where network traffic from your subnet or gateway is directed.

Main Route Table:



Custom Route Table:



Amazon VPC Security:

Think of a network **Access Control List** (network **ACL**) as a virtual firewall at the subnet level.

A diagram of a network connection

Description automatically generated

The **default network ACL** shown in the preceding table, allows all traffic in and out of the subnet.

**Custom network ACL**

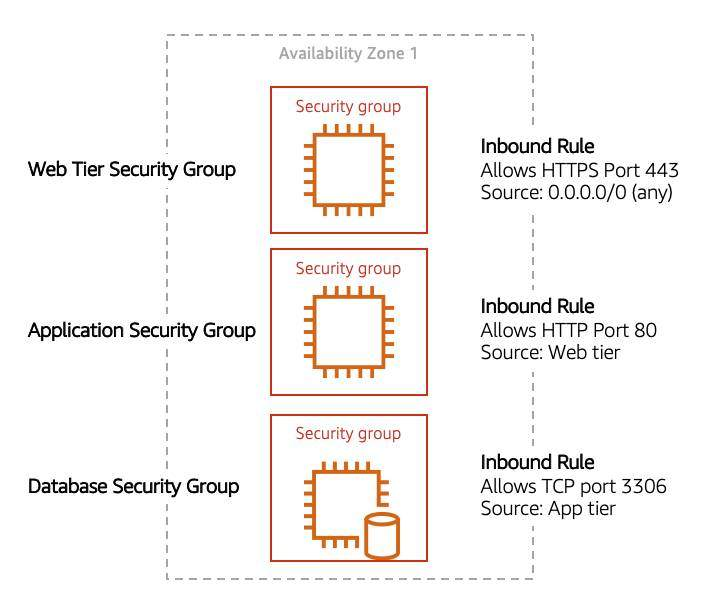
A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Secure EC2 instances with security groups: 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**.



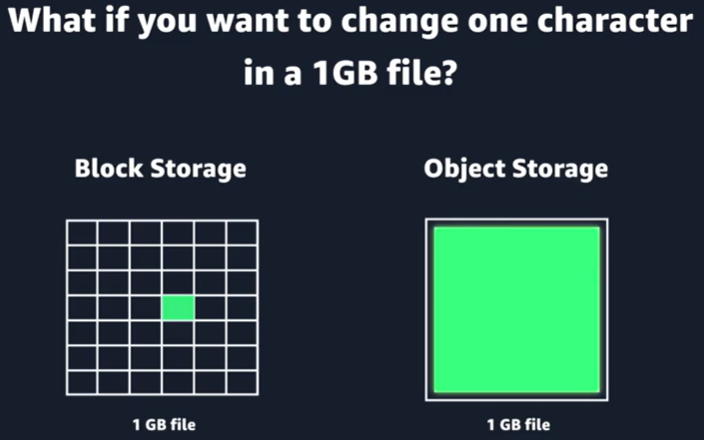
**Module 4. AWS Storage**

Storage Types: AWS storage services are grouped into three categories: file storage, block storage, and object storage

n 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.

A screenshot of a block storage

Description automatically generated



File Storage with Amazon **EFS** and Amazon **FSx**:

A green square with white outline of a folder and arrows

Description automatically generated

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.

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.

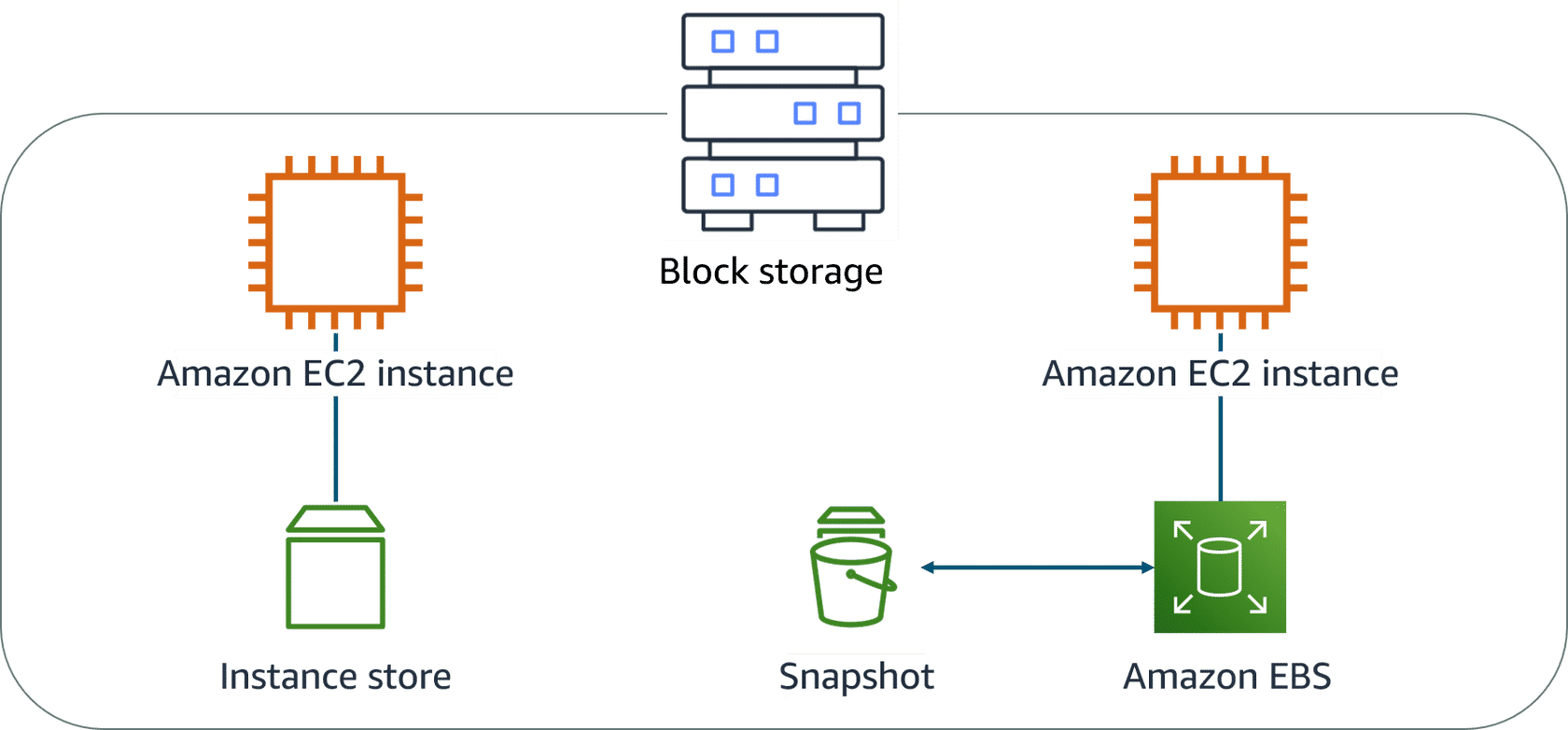
A close-up of a blue and white box

Description automatically generated

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. File systems: **Lustre**, NetApp **ONTAP**, **OpenZFS**, and **Windows File Server**.

**Block Storage** with Amazon EC2 Instance Store and Amazon **EBS**:

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.



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.

**EBS volumes** act similarly to external drives: Detachable, Distinct, Size-limited, 1-to-1 connection.

You can **scale EBS volumes** in two ways: **Increase the volume size**, **Attach multiple volumes** to a single EC2 instance.

EBS volumes are organized into two main categories: solid-state drives (**SSDs**) and hard-disk drives (**HDDs**).

**Amazon EBS use cases**: operating systems, databases, enterprise applications, big data analytics engines.

**Amazon EBS benefits**: Flexibility, Backups, High availability, Data persistence, Data encryption.

**Amazon EBS snapshots**: 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.

A black arrow with a white background

Description automatically generated

**Object Storage** with Amazon **S3**: Object storage is built for the cloud and delivers virtually unlimited scalability, high durability, and cost effectiveness

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.

Amazon S3 concepts: In Amazon S3, you store your objects in containers called buckets. When you store an object in a bucket, the combination of a bucket name, key, and version ID uniquely identifies the object.

A close-up of a logo

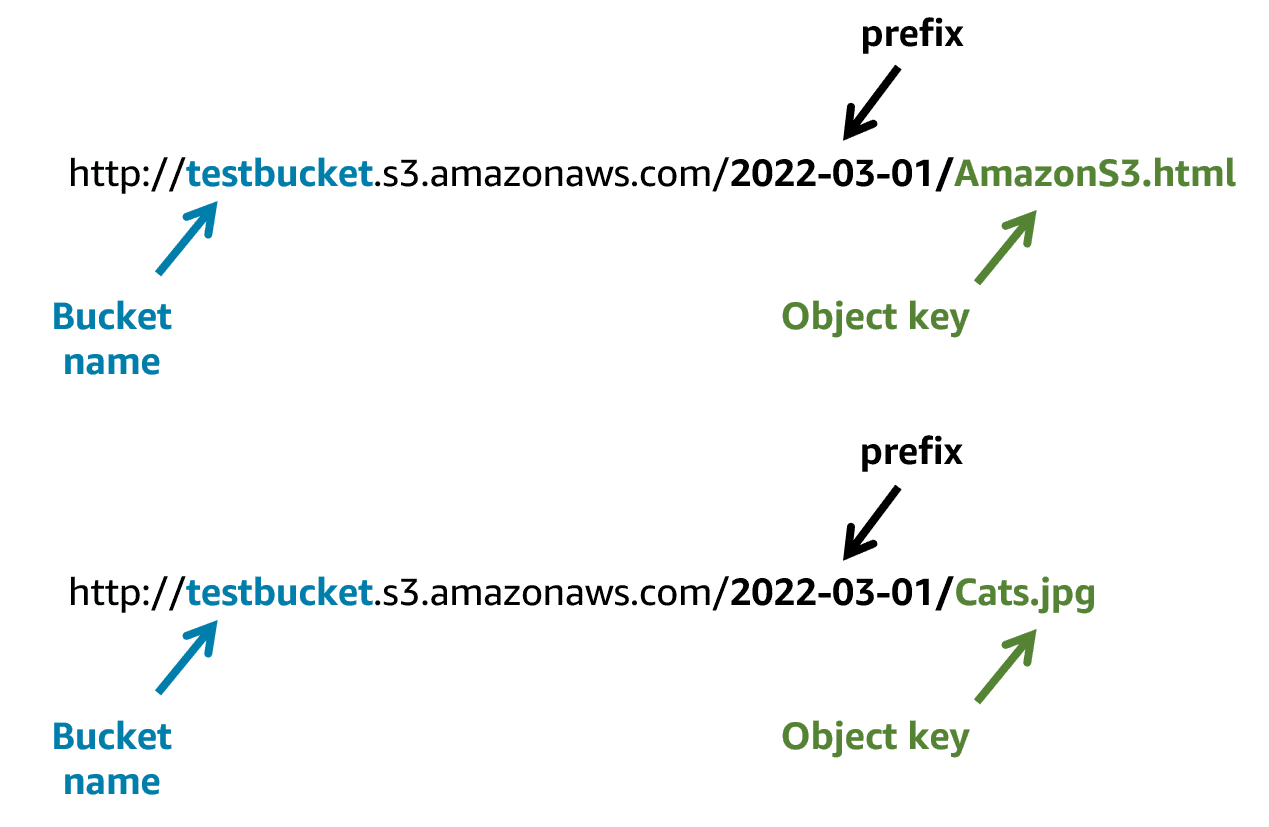
Description automatically generated

Amazon S3 console does support the concept of folders

Amazon S3 bucket names: Therefore, each bucket name must be unique across all AWS accounts in all AWS Regions within a partition.

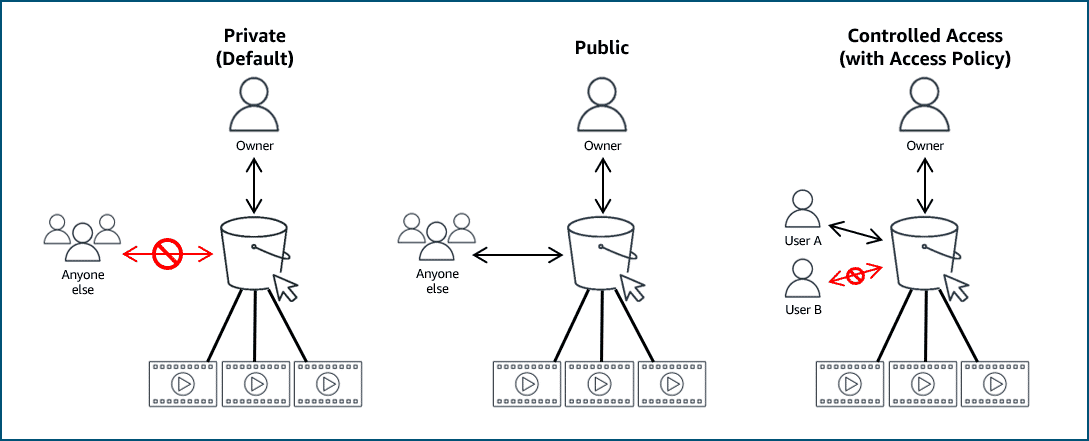
A **partition** is a grouping of Regions, of which AWS currently has three: Standard Regions, China Regions, and AWS GovCloud (US).

Object key names: The object key (key name) uniquely identifies the object in an Amazon S3 bucket.



Amazon S3 use cases: backup and storage, media hosting, software delivery, data lakes, static websites, static content.

Security in Amazon S3: Everything in Amazon S3 is private by default.



Amazon S3 and IAM policies: 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.

Amazon S3 bucket policies: 3 bucket policies are defined in a JSON format. 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.

<https://docs.aws.amazon.com/en_us/AmazonS3/latest/userguide/example-bucket-policies.html>

Amazon S3 encryption:

Amazon S3 storage classes:

Amazon S3 versioning:

Managing your storage lifecycle:

Choosing the Right Storage Service:

(Video Demo): Creating an Amazon S3 Bucket:

**Module 5. Databases on AWS**

**Module 6. Monitoring, load balancing and scaling**