

SECTION 1

Let's Get Started!

What is Cloud Computing and AWS?





What is Cloud Computing?

Non-Cloud Services



Email Server



File Server



Customer Relationship
Management (CRM)

Cloud Services



Gmail



Dropbox



Salesforce.com

You don't **own** or **manage** the infrastructure on which the service runs

Cloud services are offered on a **subscription / consumption** model

The service **scales automatically** as demand changes

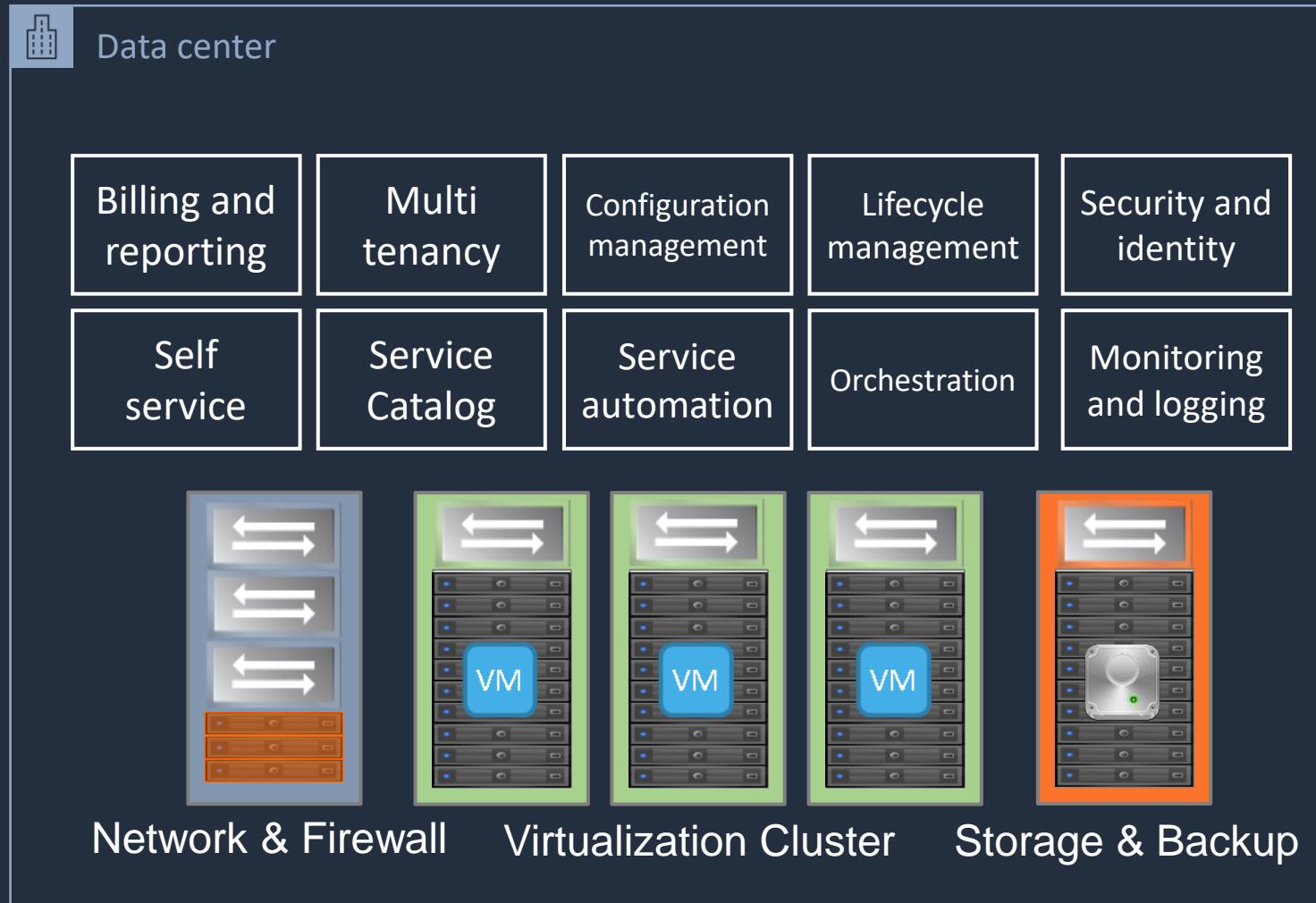


Key Characteristics of Cloud

Name	Description
On-demand, self-service	A user can consume cloud resources, as needed, automatically, and without human interaction
Broad network access	Capabilities are available over the network using standard mechanisms. Can be the Internet or a Wide Area Network (WAN)
Resource pooling	The providers resources are pooled and serve multiple consumers using a multi-tenant model
Rapid elasticity	Capabilities can scale “elastically” based on demand
Measured service	Resource usage is monitored and metered



Private Cloud



Costs:

- Data center
- Security
- Power
- Connectivity
- Physical hardware
- Software licensing
- Maintenance contracts
- Staff



Public Cloud

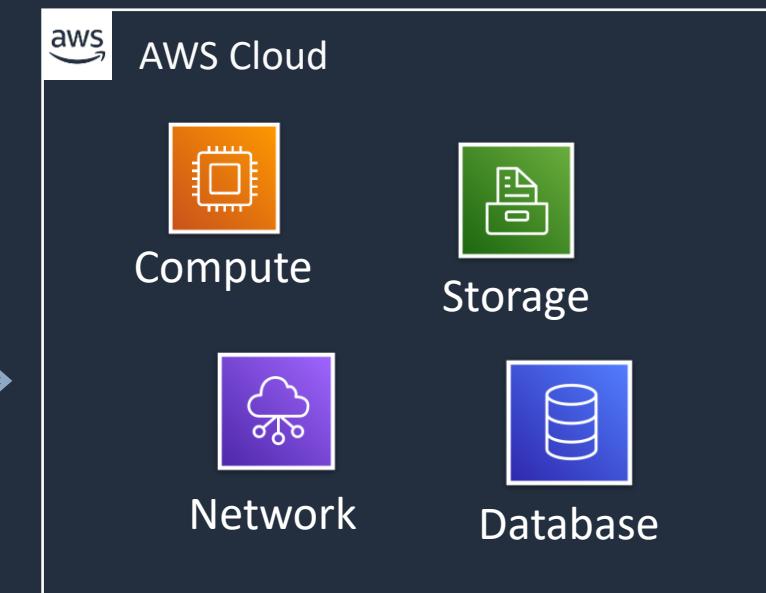
Examples include AWS, Microsoft Azure, Google Cloud Platform

Benefits:

- Variable expense, instead of capital expense
- Economies of scale
- Massive elasticity



Public Cloud



Connected using either the
Internet or a **private link**



Cloud Service Models

Private Cloud – Managed by you

Infrastructure as a Service (IaaS) – Managed virtual machines

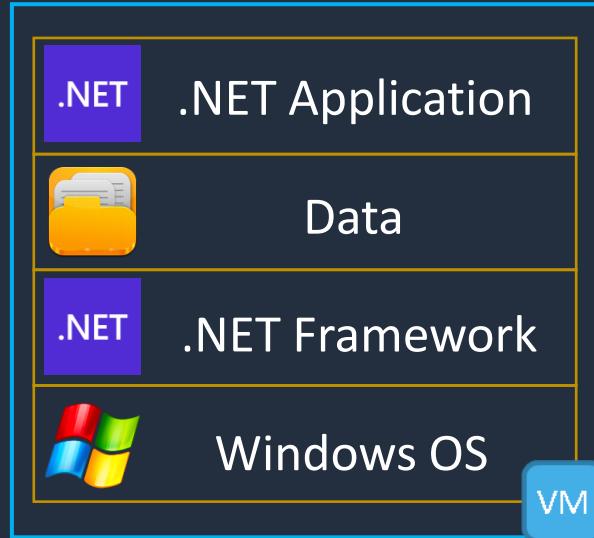
Platform as a Service (PaaS) – Just bring your code

Software as a Service (SaaS) – Everything is managed for you



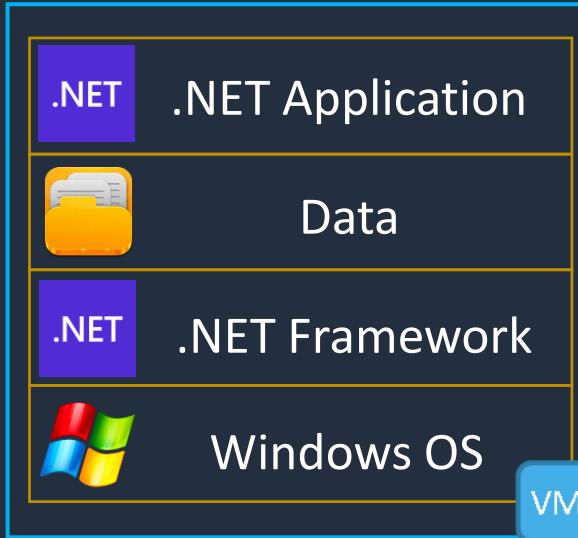
Cloud Service Models: Comparison

Private Cloud



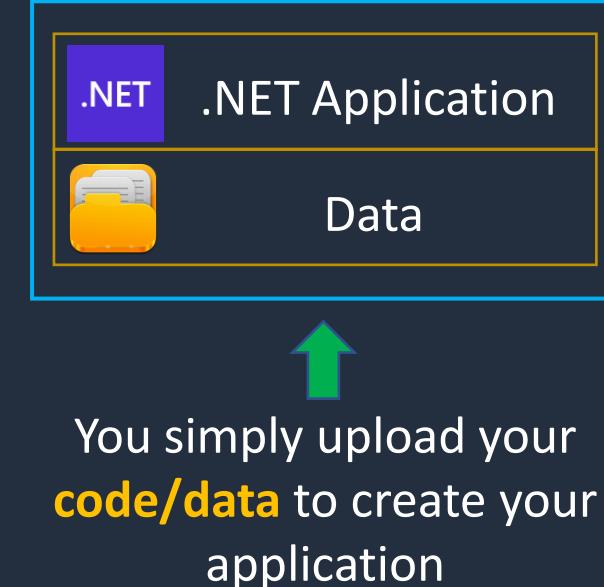
You manage **everything** -
greater responsibility +
greater control

IaaS

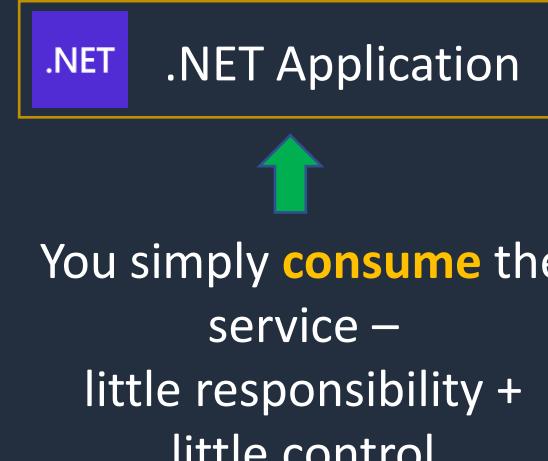


You manage from the
virtual server upwards

PaaS

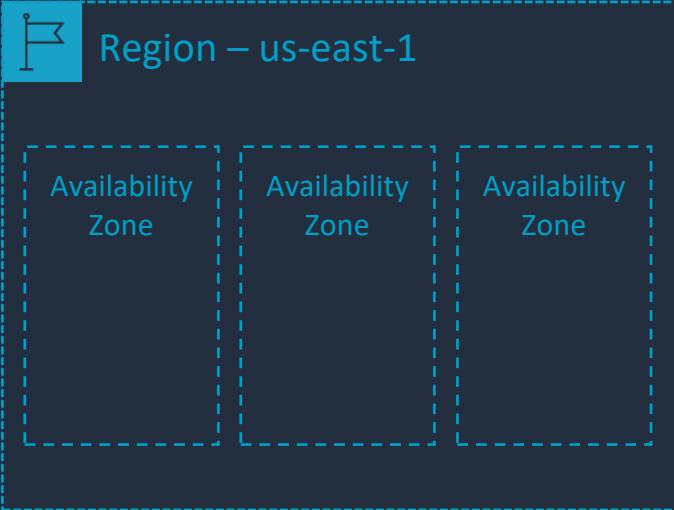


SaaS



- **Amazon Web Services (AWS):** A Hyperscale public cloud provider with over 200 services
- **Market Leadership and Growth:** AWS is the leading cloud service provider with the largest market share
- **Global Infrastructure:** AWS infrastructure is spread globally across many Regions and availability zones
- **Innovation and Expansion:** AWS is known for its rapid pace of innovation, introducing new services and features regularly

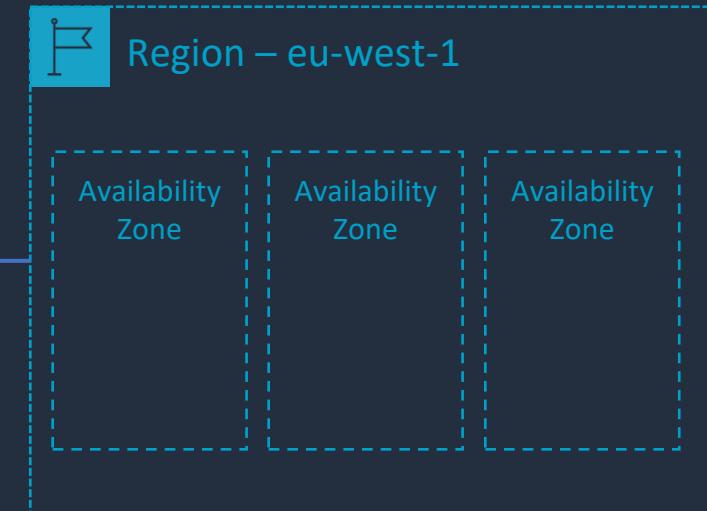
AWS Global Infrastructure



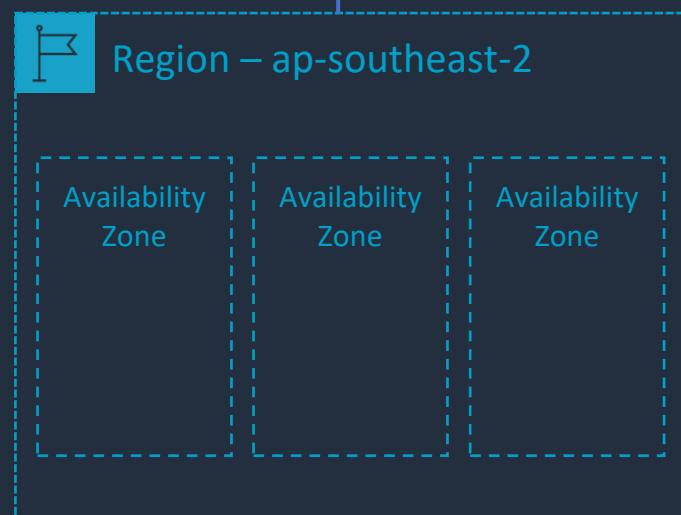
A **Region** is a separate physical location in the world



There are many **Regions** around the world



An Availability Zone is composed of **one** or **more** data centers



Each region consists of multiple **Availability Zones**

aws Managing AWS: Management Console



AWS Management Console

Console Home [Info](#)

Recently visited [Info](#)

No recently visited services

Explore one of these commonly visited AWS services.

[EC2](#) [S3](#) [RDS](#) [Lambda](#)

[View all services](#)

Welcome to AWS

Getting started with AWS [Info](#)

Training and certification [Info](#)

What's new with AWS? [Info](#)

AWS Health

Open issues 0 Past 7 days

Scheduled changes 1 Upcoming and past 7 days

Other notifications 0 Past 7 days

Cost and usage

Current month costs \$0.59

Forecasted month end costs \$0.71 Down 39% from last month

Last month costs \$1.17

Average month costs \$1.66

Cost (USD)

Month (Year)

Route 53 Tax Key Management Service

© 2024, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

View all services All AWS services organized by category on a page [X](#)

A

Activate for Startups AWS Activate provides resources to help startups build and grow on AWS.

AWS Amplify AWS Amplify is a complete platform—frameworks & tools and app services—for developing, building, testing, and running mobile and web apps

API Gateway Build, Deploy and Manage APIs

AWS App Mesh Easily monitor and control microservices

AWS App Runner Build and run production web applications at scale

AWS AppConfig Use feature flags, operational flags, and other runtime configuration to make changes quickly and safely on production

AWS AppFabric Connecting SaaS applications for better productivity and security.

Amazon AppFlow Amazon AppFlow integrates apps and automates data flows without code.

Application Composer Visually design and build modern applications quickly

Application Discovery Service Discover on-premises application inventory and dependencies

AWS Application Migration Service AWS Application Migration Service (MGN) automates lift-and-shift migration.

Recently visited

Favorites

All services

Analytics

Application Integration

Blockchain

Business Applications

Cloud Financial Management

Compute

Containers

Customer Enablement

Database

Developer Tools

End User Computing

Front-end Web & Mobile

Game Development

Internet of Things

Machine Learning

Management & Governance

Media Services

Migration & Transfer

Networking & Content Delivery

Quantum Technologies

Robotics



Managing AWS: Command Line



AWS Command Line Interface (AWS CLI)



```
aws ec2 run-instances --image-id ami-xxxxxxxx --count 1 --instance-type t2.micro
```

This command launches a **virtual server**
(EC2 instance) on Amazon EC2



```
aws s3 ls s3://mys3databucket
```

This command lists the contents of a
storage container (bucket) on **Amazon S3**



Software Development Kit (SDK)

```
import boto3
from botocore.exceptions import NoCredentialsError

def upload_file_to_s3(file_name, bucket_name, object_name=None):
    """
    Upload a file to an S3 bucket
    """
    if object_name is None:
        object_name = file_name

    # Upload the file
    s3_client = boto3.client('s3')
    try:
        response = s3_client.upload_file(file_name, bucket_name, object_name)
        print(f"File {file_name} uploaded to {bucket_name}/{object_name}")
    except FileNotFoundError:
        print(f"The file {file_name} was not found.")
    except NoCredentialsError:
        print("Credentials not available")

def main():
    file_name = 'docs/s-log91293.txt'
    bucket_name = 's3-storage-b1-3'
    object_name = 's-log-91293.txt' # Optional

    upload_file_to_s3(file_name, bucket_name, object_name)

if __name__ == "__main__":
    main()
```

The screenshot shows a dark-themed integrated development environment (IDE) window. On the left is a vertical toolbar with various icons: magnifying glass, gear, file/folder, terminal, test tube, hexagon, ship, and AWS logo. The main area displays a Python script titled 'Untitled-1'. The script imports 'boto3' and handles exceptions for 'NoCredentialsError'. It defines a function 'upload_file_to_s3' that takes 'file_name', 'bucket_name', and 'object_name' parameters. Inside the function, it prints a message and uses 's3_client.upload_file' to upload the file. It also handles 'FileNotFoundError' and 'NoCredentialsError'. Finally, it defines a 'main' function that sets 'file_name' to 'docs/s-log91293.txt', 'bucket_name' to 's3-storage-b1-3', and 'object_name' to 's-log-91293.txt'. It then calls the 'upload_file_to_s3' function. At the bottom, there are status indicators for tabs (0), AWS Builder ID (0), CodeWhisperer, encoding (UTF-8), line endings (LF), Python 3.9.6 64-bit, Go Live, and a bell icon.

A developer writes code in an **integrated development environment (IDE)**

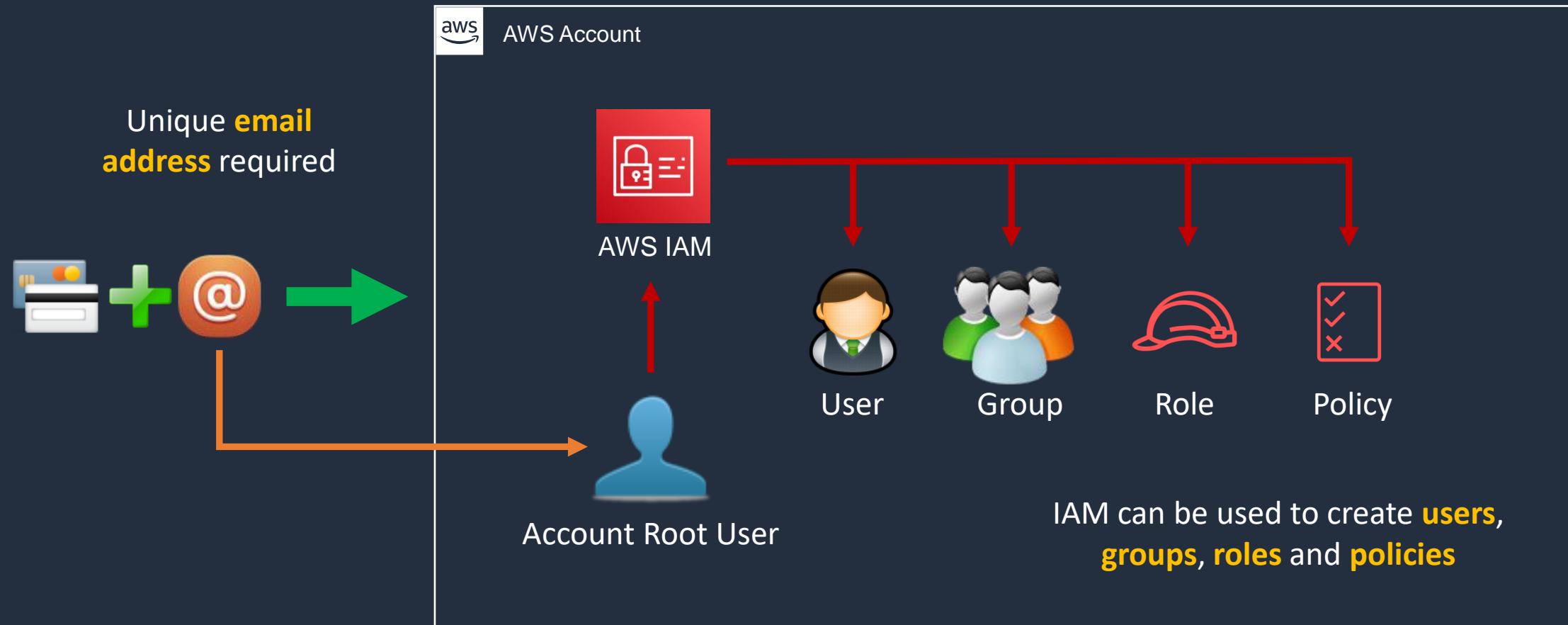
The code leverages an **AWS Software Development Kit (SDK)** to work with cloud services

AWS Account Overview



AWS Account Overview

It's an IAM best practice to create **individual users**
and to avoid using the **Root** account

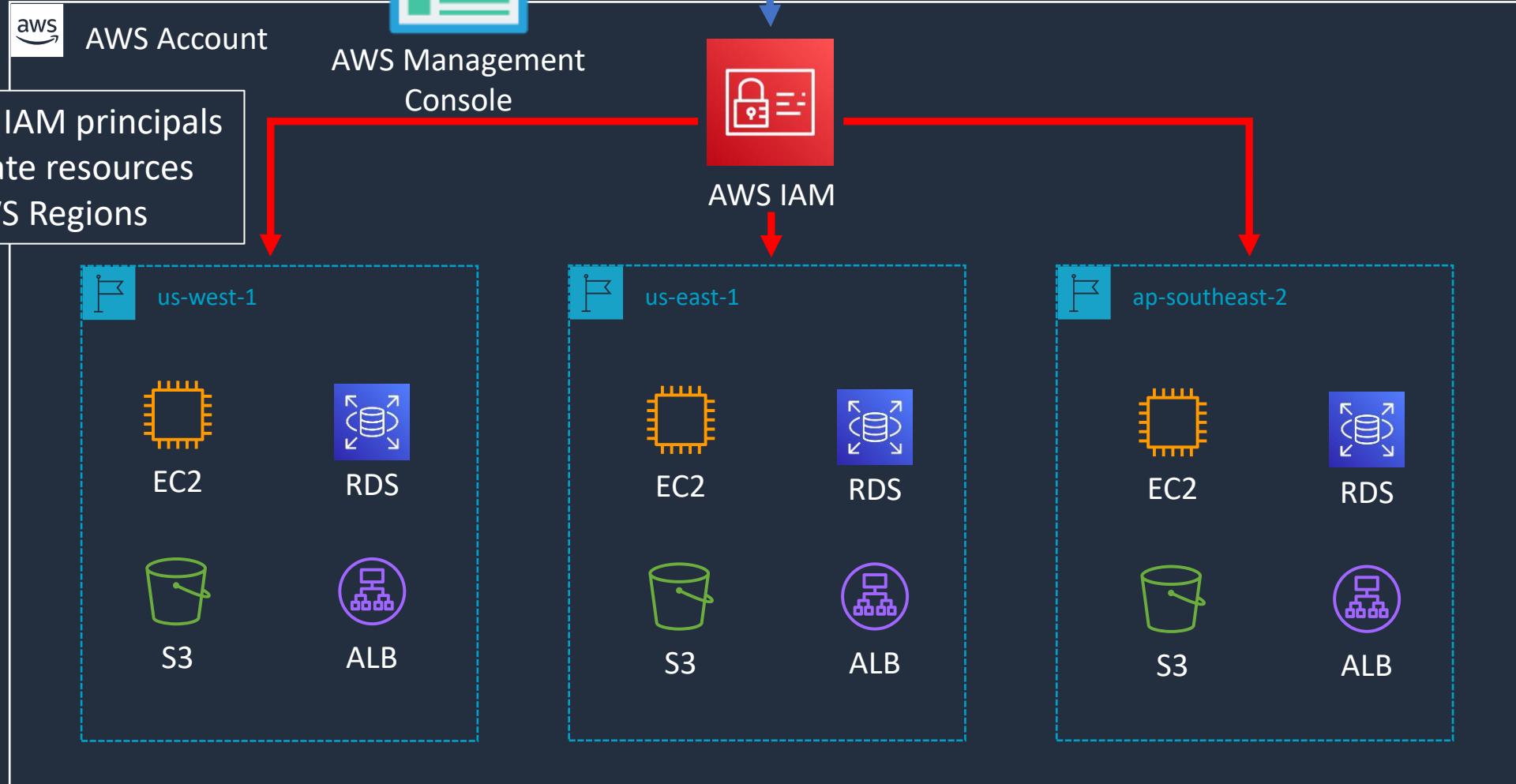


The Root user has **full control**
over the account



AWS Account Overview

Authentication: IAM principals authenticate to IAM using the console, API, or CLI



All AWS **identities** and **resources** are created
within the AWS account

Create your AWS Free Tier Account



aws What you need...



Credit card for setting up the account and paying any bills



Unique email address for this account

john@gmail.com



Check if you can use a **dynamic alias** with an existing email address



john+ACCOUNT-ALIAS-1@gmail.com

john+ACCOUNT-ALIAS-2@gmail.com



AWS account name / alias



Phone to receive an **SMS** verification code

Configure Account and Create a Budget



Account Configuration

- Configure **Account Alias**
- Enable access to billing for **IAM users**
- Update **billing preferences**
- Create a **budget and alarm**

Install Tools and AWS CLI





Install Tools and Configure AWS CLI

- ✓ Download the code (*last lesson of this section*)
- ✓ Install Visual Studio Code
- ✓ Install the AWS CLI
- ✓ Access AWS CloudShell

Command Line Practice



SECTION 2

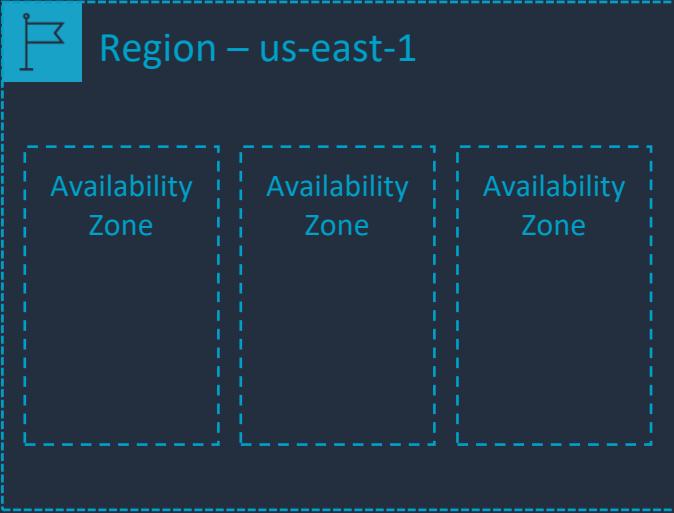
Cloud Computing and AWS

The AWS Global Infrastructure





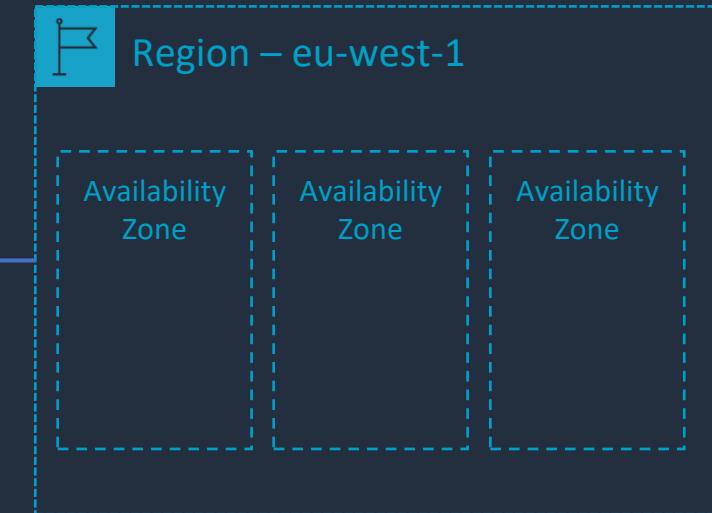
AWS Global Infrastructure



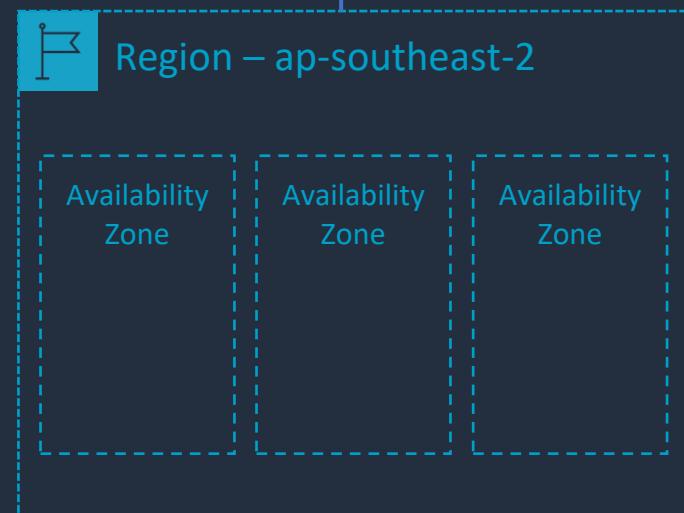
A **Region** is a separate physical location in the world



There are many **Regions** around the world



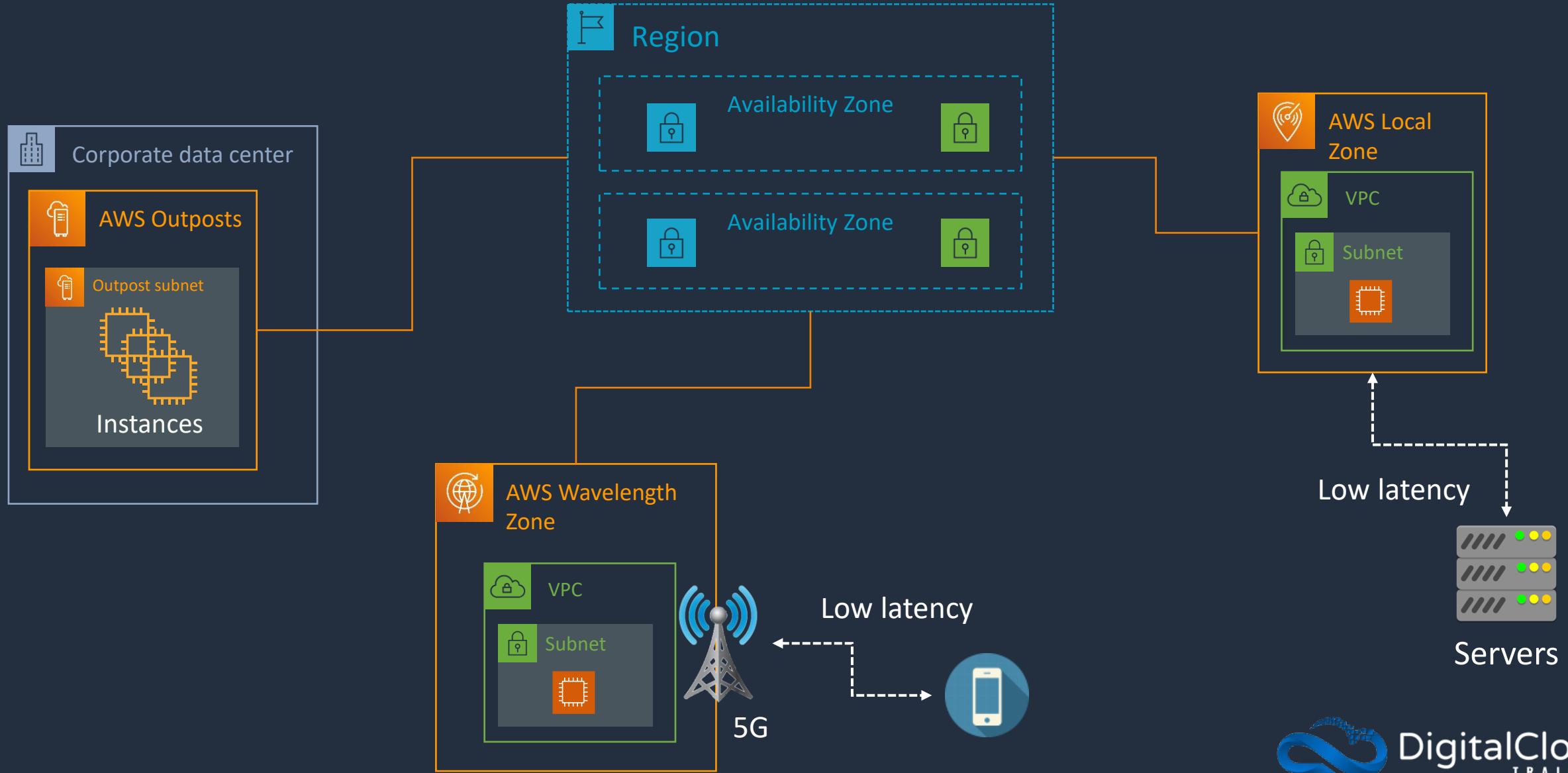
An Availability Zone is composed of **one** or **more** data centers



Each region consists of multiple **Availability Zones**

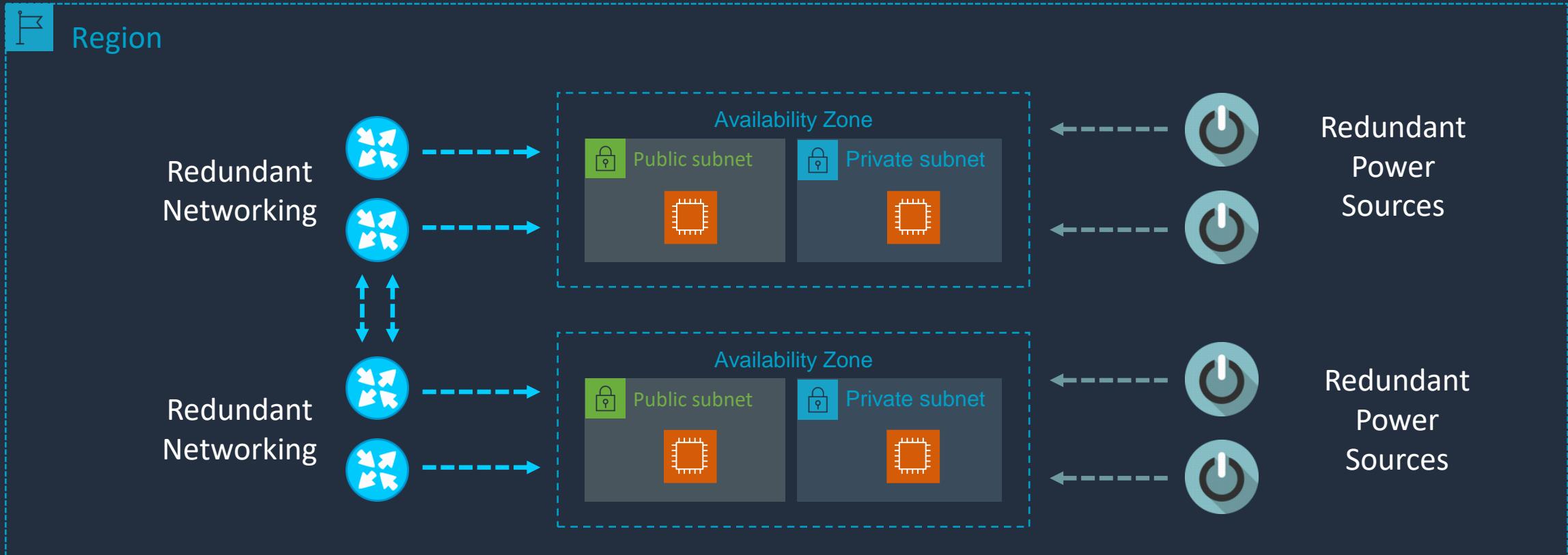


AWS Global Infrastructure



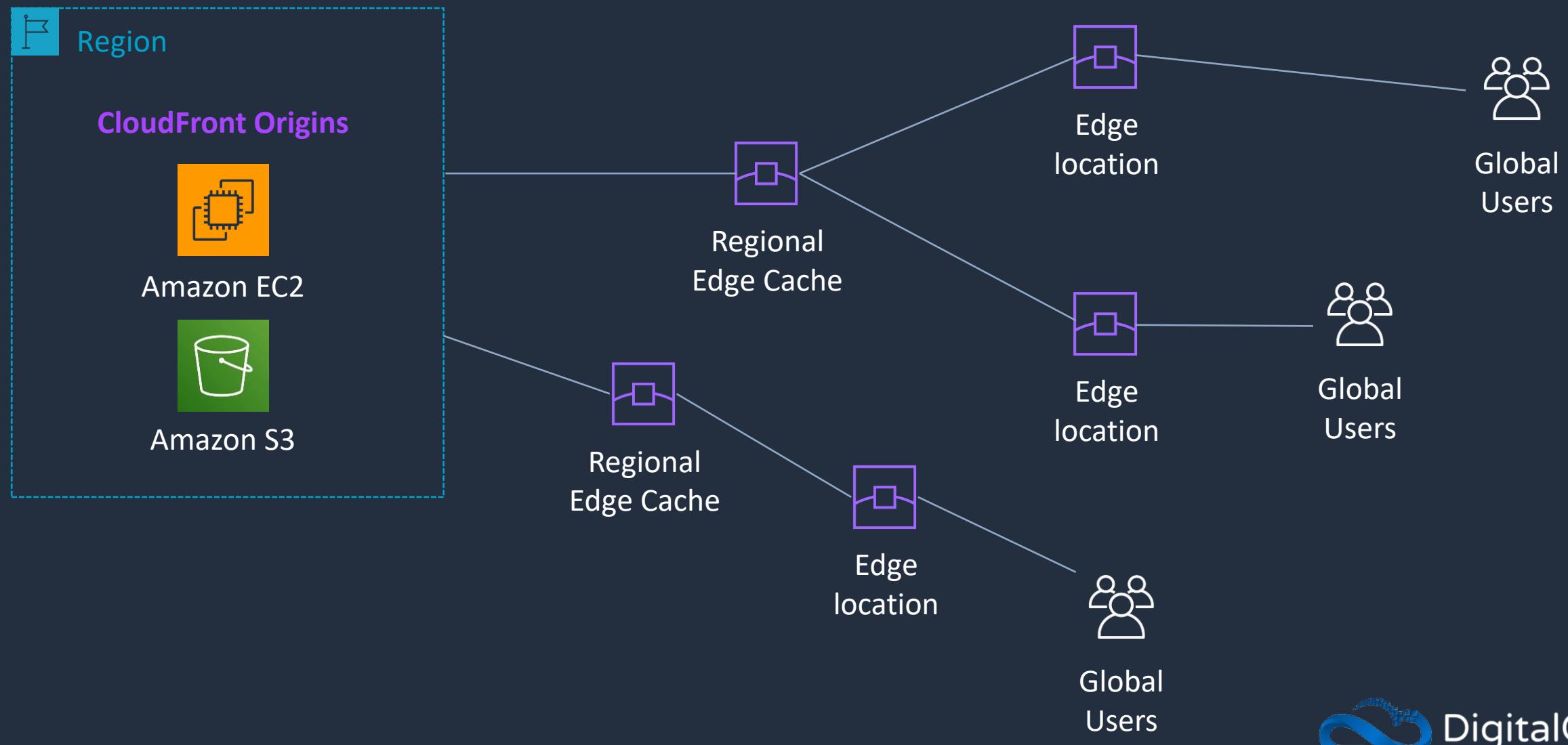


AWS Global Infrastructure



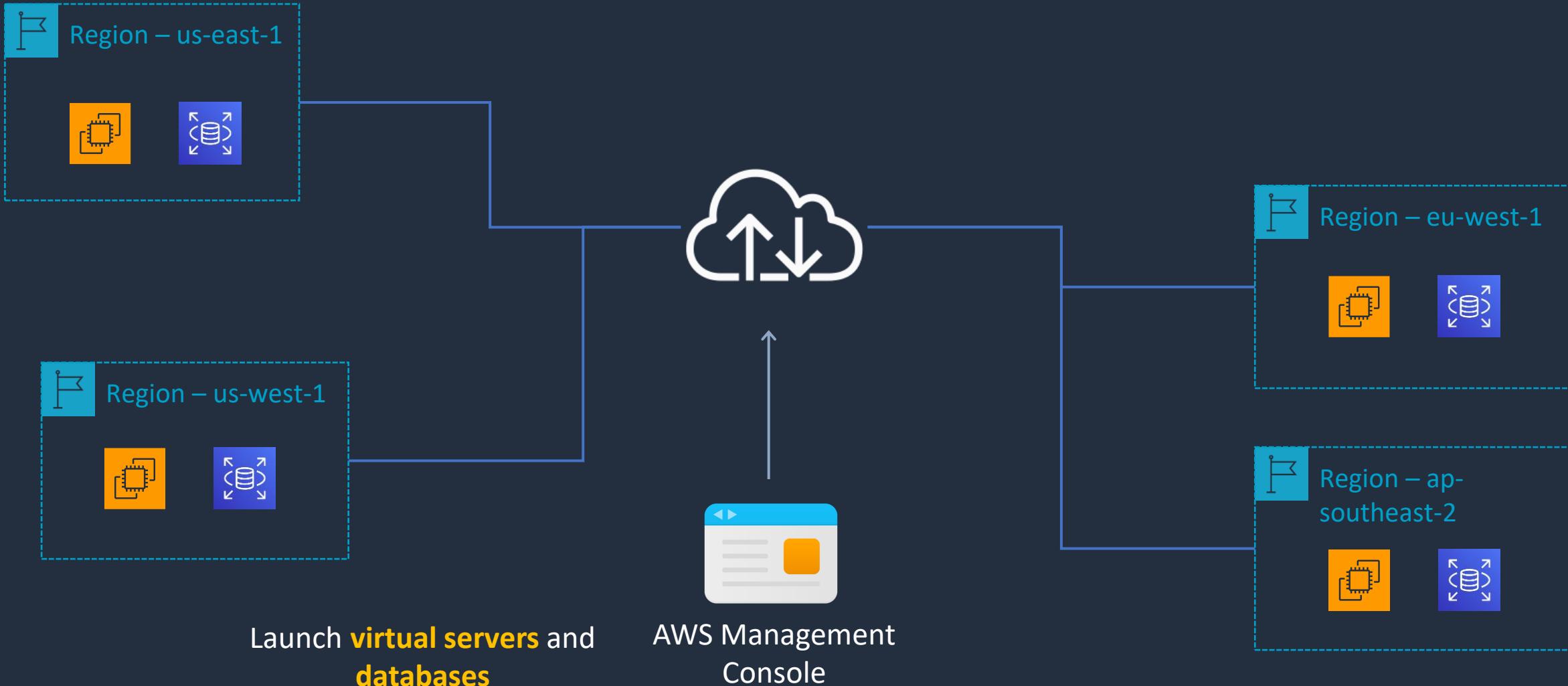


Amazon CloudFront





Deploying Services Globally

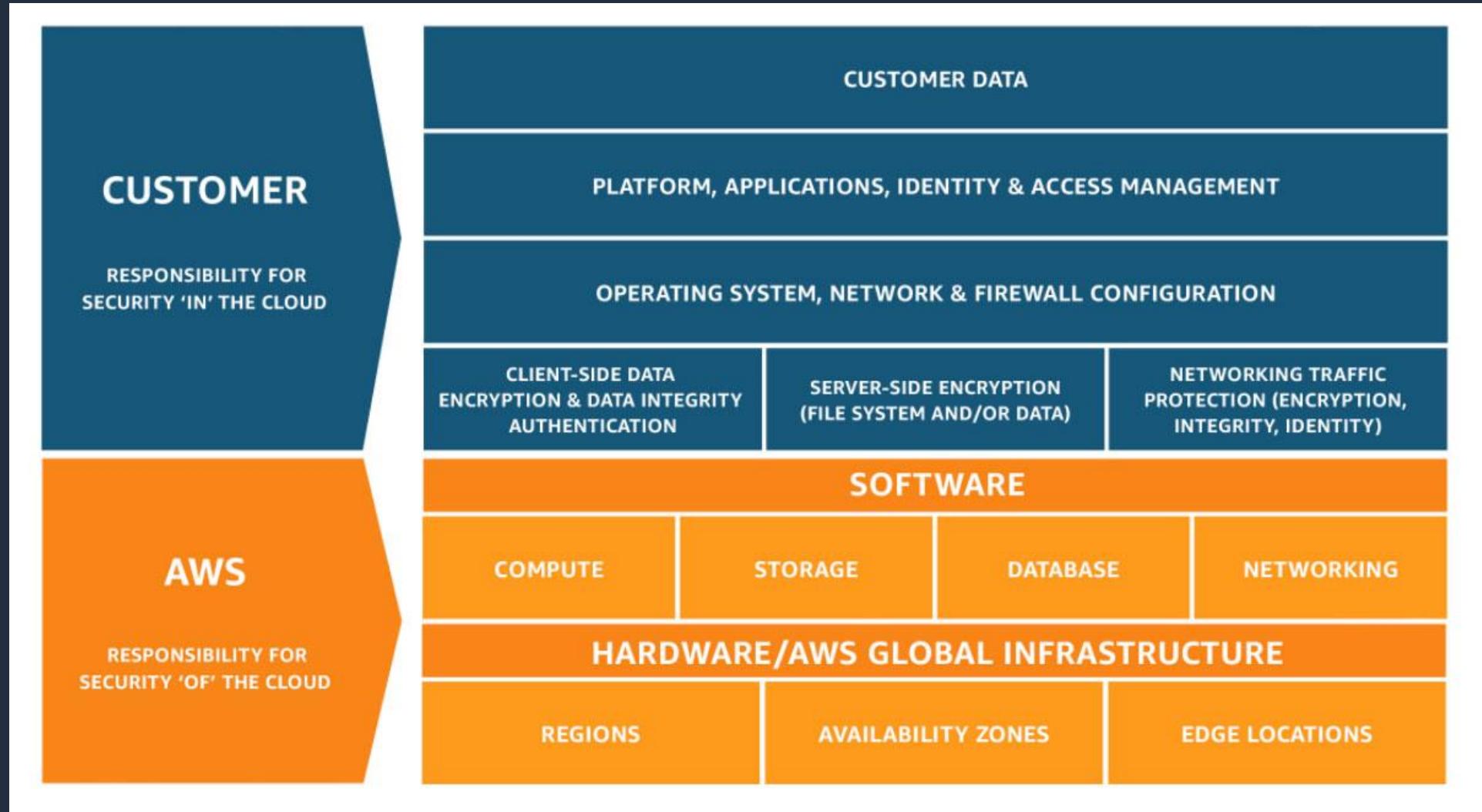


The AWS Shared Responsibility Model





The AWS Shared Responsibility Model





The AWS Shared Responsibility Model

CUSTOMER RESPONSIBILITY



Bucket with objects



Role



Multi-Factor Authentication



Security Group



Patch management



Staff training



Data encryption



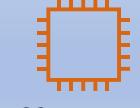
IAM User



Network ACL



SSL encryption



EC2 Instance



Auto Scaling



Elastic load balancer

AWS RESPONSIBILITY



Data center security



Network router



Network switch



Storage



Database Server

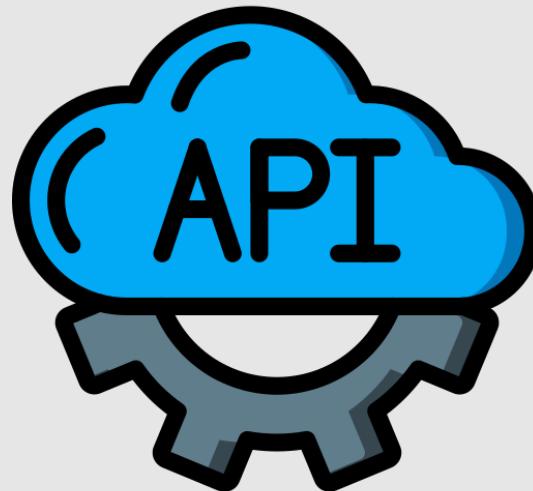


Server



Disk drive

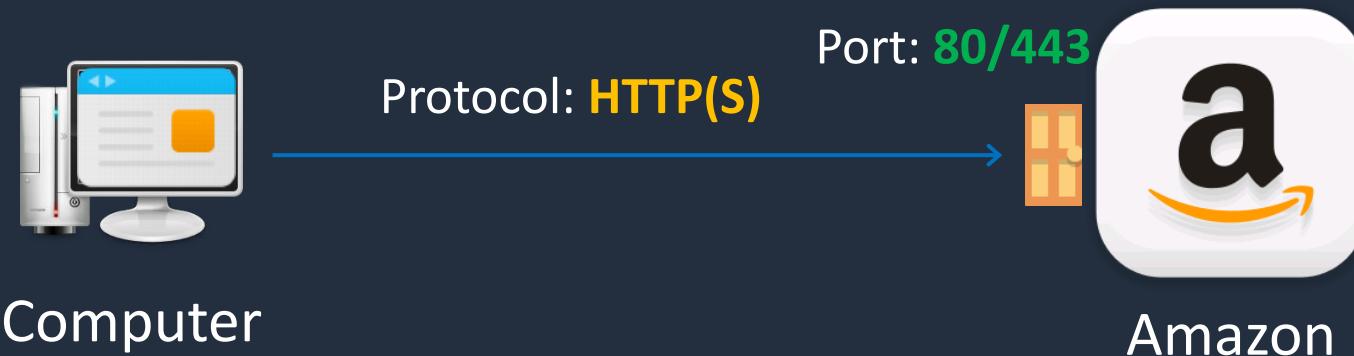
Application Programming Interfaces (APIs)





Ports and Protocols

A **protocol** is a language used for communication over a network



A **port** is like an open door behind which a specific service is waiting for connections



HTTP Methods

- **GET:** The GET method retrieves information
- **POST:** The POST method is used to submit data
- **PUT:** The PUT method replaces data
- **DELETE:** The DELETE method deletes data



HTTP Request

Browser issues an **HTTP GET** request to the web server



Computer



Amazon

The web server returns
the website **content**



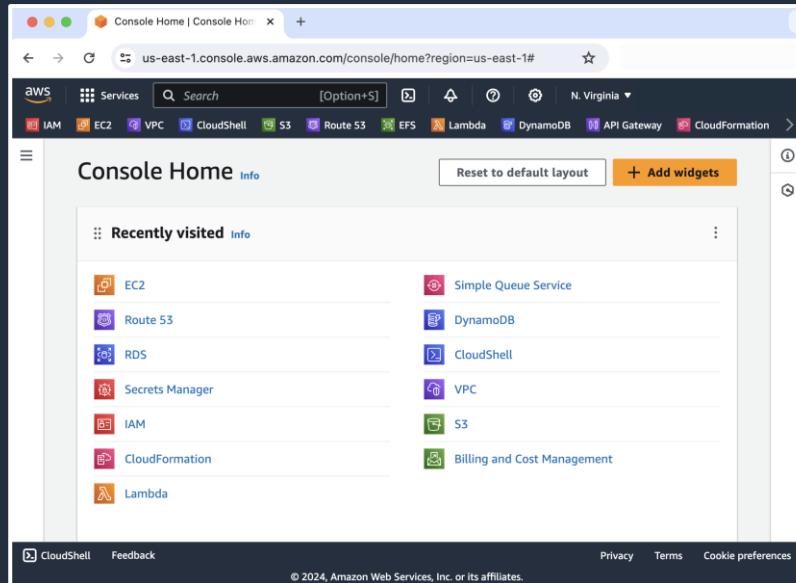
What are APIs?

- **Stands for** Application Programming Interface
- **APIs are** a set of rules and protocols that allow computer programs talk to each other
- **APIs assist with** sharing information and data between these programs
- **APIs leverage** standard protocols such as HTTP



AWS Example

Create a new **user account**



Browser sends **HTTP requests** to the **API**

AWS creates the new **user account**



The **API** defines specific **actions** you can request

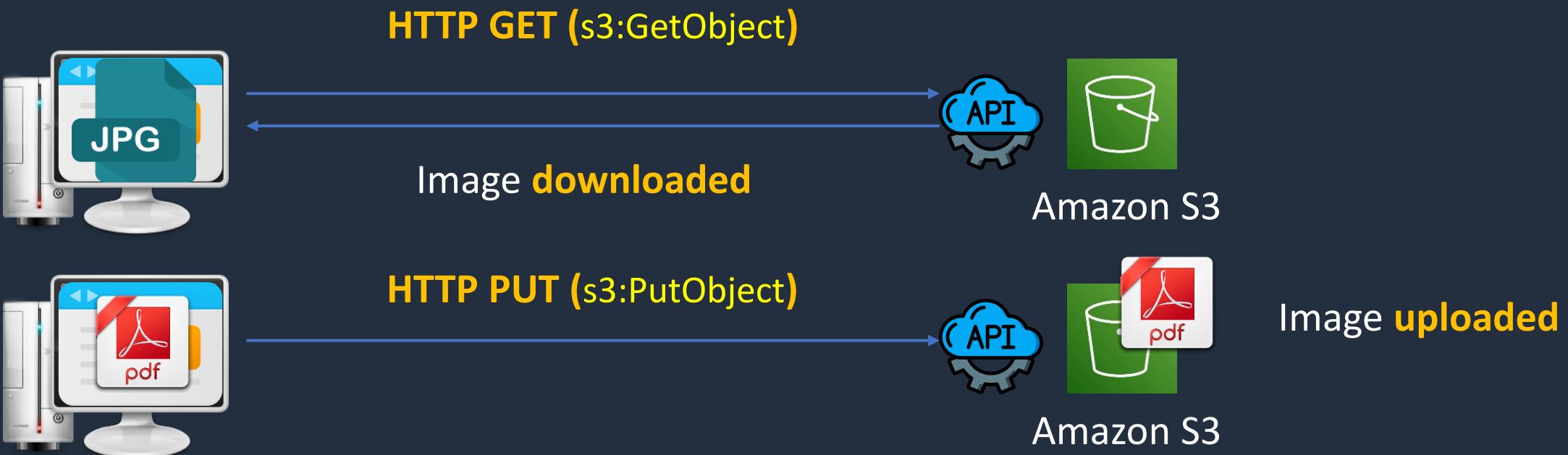
Every **action** you take on AWS is an **API call**



API Actions (AWS examples)

Amazon S3 (Simple Storage Service):

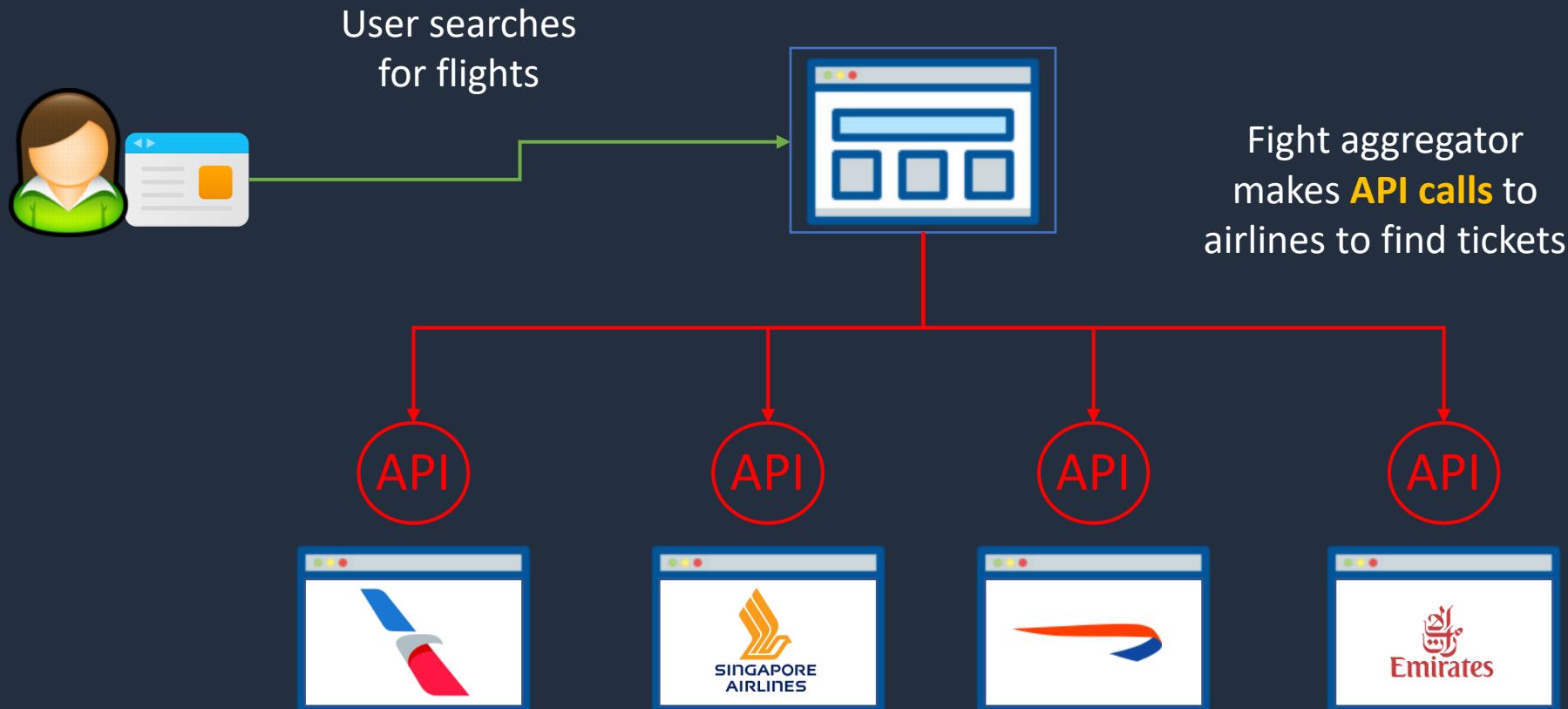
- **GetObject:** Retrieves objects from Amazon S3
- **PutObject:** Adds an object to a bucket





Flight Aggregator Example

Flight aggregator such as
Momondo or Skyscanner



AWS Pricing Fundamentals





AWS Pricing Fundamentals

Compute



Amount of resources such as CPU and RAM and **duration**

Storage



Quantity of **data stored / allocated**

Outbound Data Transfer



Quantity of data that is **transferred out** from all services



AWS Pricing Fundamentals

Pay-as-you-go

- Easily adapt to changing business needs
- Improved responsiveness to change
- Adapt based on needs, not forecasts



AWS Pricing Fundamentals

Save when you reserve

- Invest in reserved capacity (e.g. RDS and EC2)
- Save up to 75% compared to on-demand (pay-as-you-go)
- The more you pay upfront the greater the discount



AWS Pricing Fundamentals

Pay less by using more

- Pay less using volume-based discounts
- Tiered pricing means the more you use the lower the unit pricing

The 6 Advantages of Cloud Computing





The 6 Advantages of Cloud Computing

1. Trade capital expense for variable expense

CAPEX

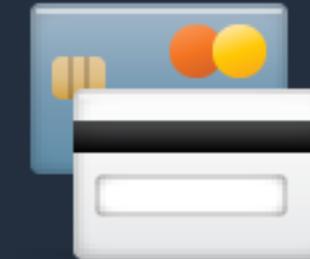


Purchase servers



Tax deductible over depreciation lifetime

OPEX



Pay as you go



Tax deductible in same year



The 6 Advantages of Cloud Computing

2. Benefit from massive economies of scale





The 6 Advantages of Cloud Computing

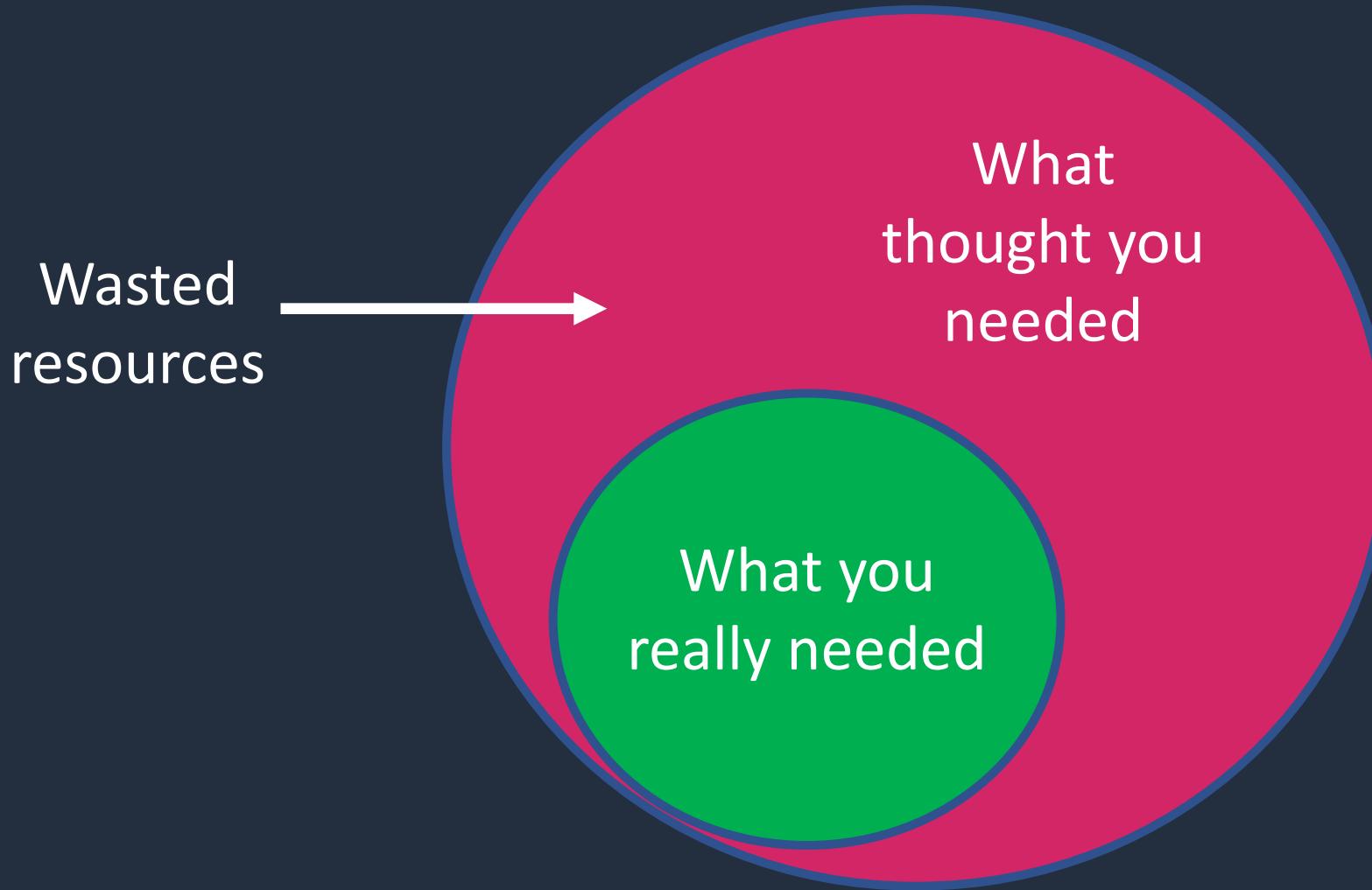
2. Benefit from massive economies of scale

- Aggregated usage across hundreds of thousands of customers = lower variable costs for customers



The 6 Advantages of Cloud Computing

3. Stop guessing capacity





The 6 Advantages of Cloud Computing

4. Increase speed and agility



Speed = deploy resources easily and quickly

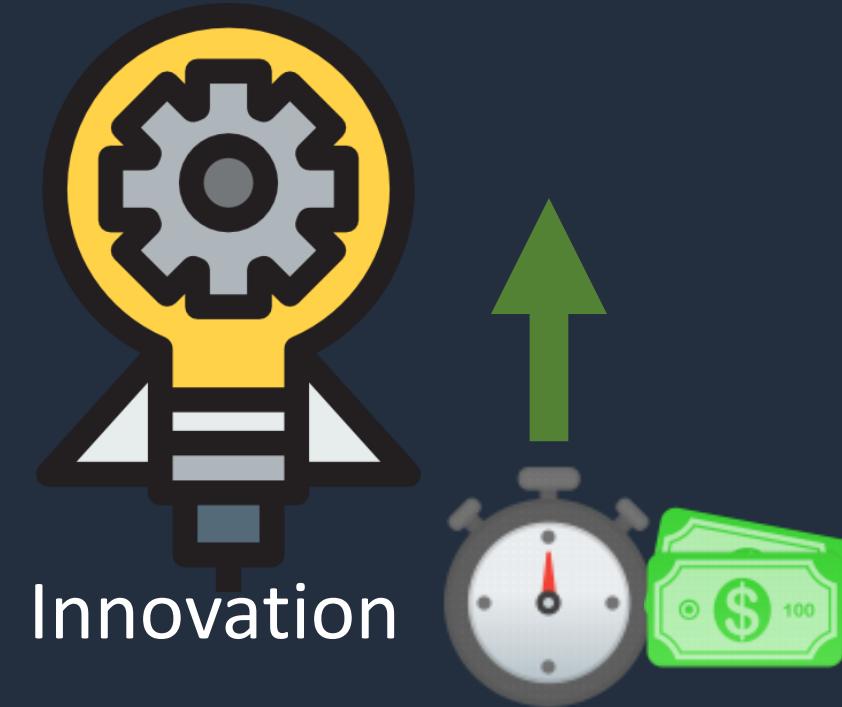


Agility = react to change ; speed to market



The 6 Advantages of Cloud Computing

5. Stop spending money running and maintaining data centers





The 6 Advantages of Cloud Computing

6. Go global in minutes



SECTION 3

AWS Authentication and Access Control

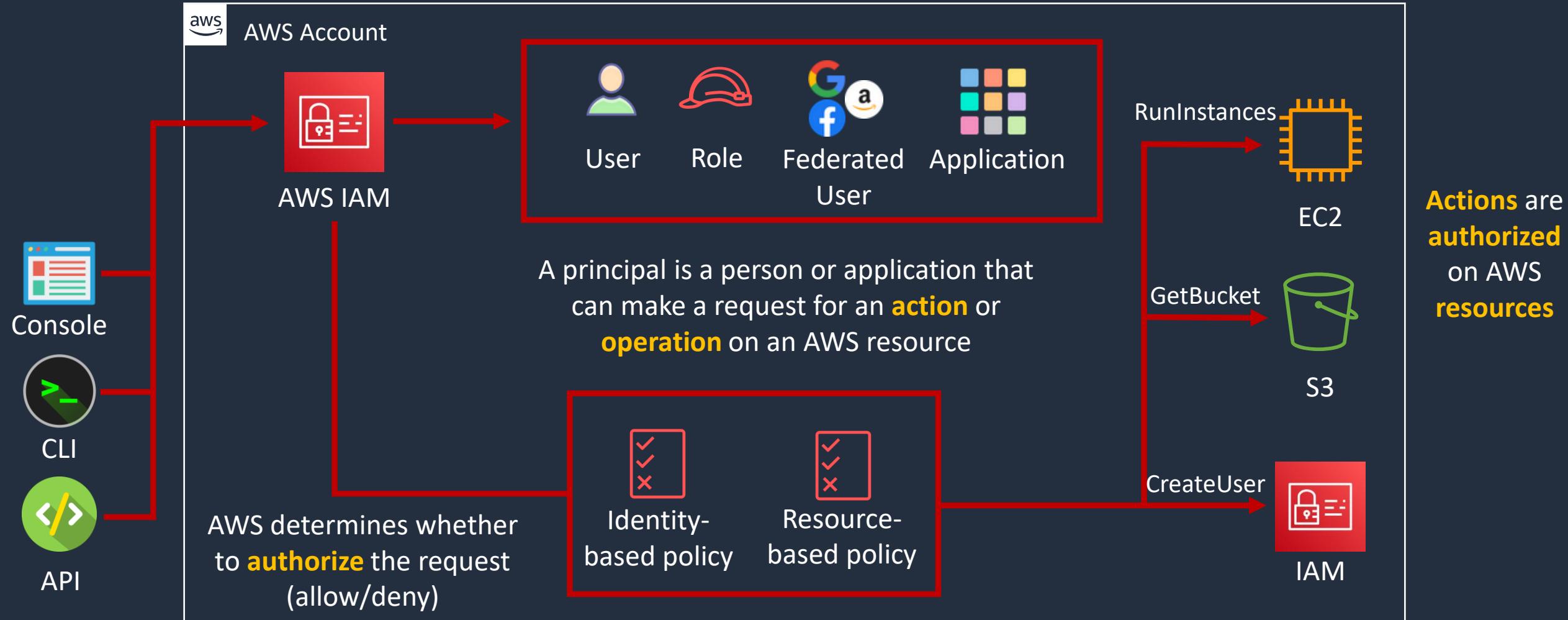
AWS Identity and Access Management (IAM)





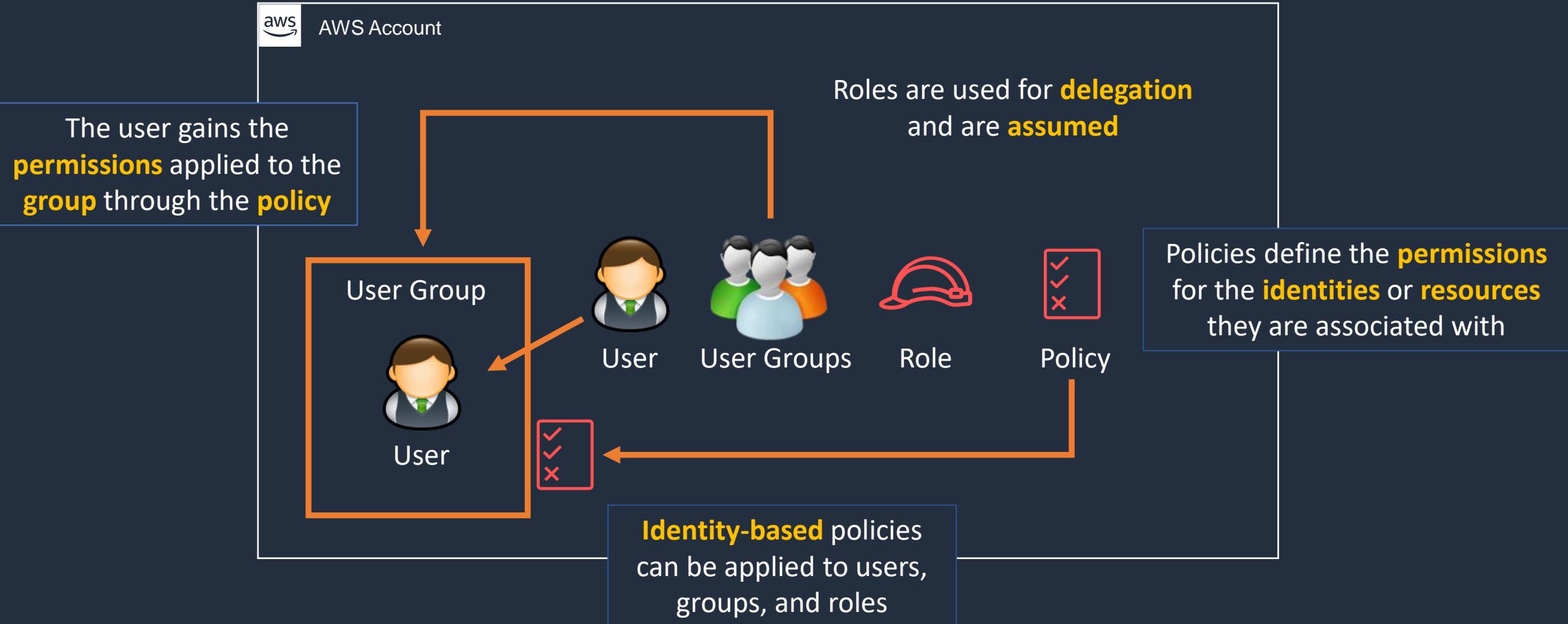
AWS Identity and Access Management (IAM)

IAM Principals must be **authenticated** to send requests (with a few exceptions)



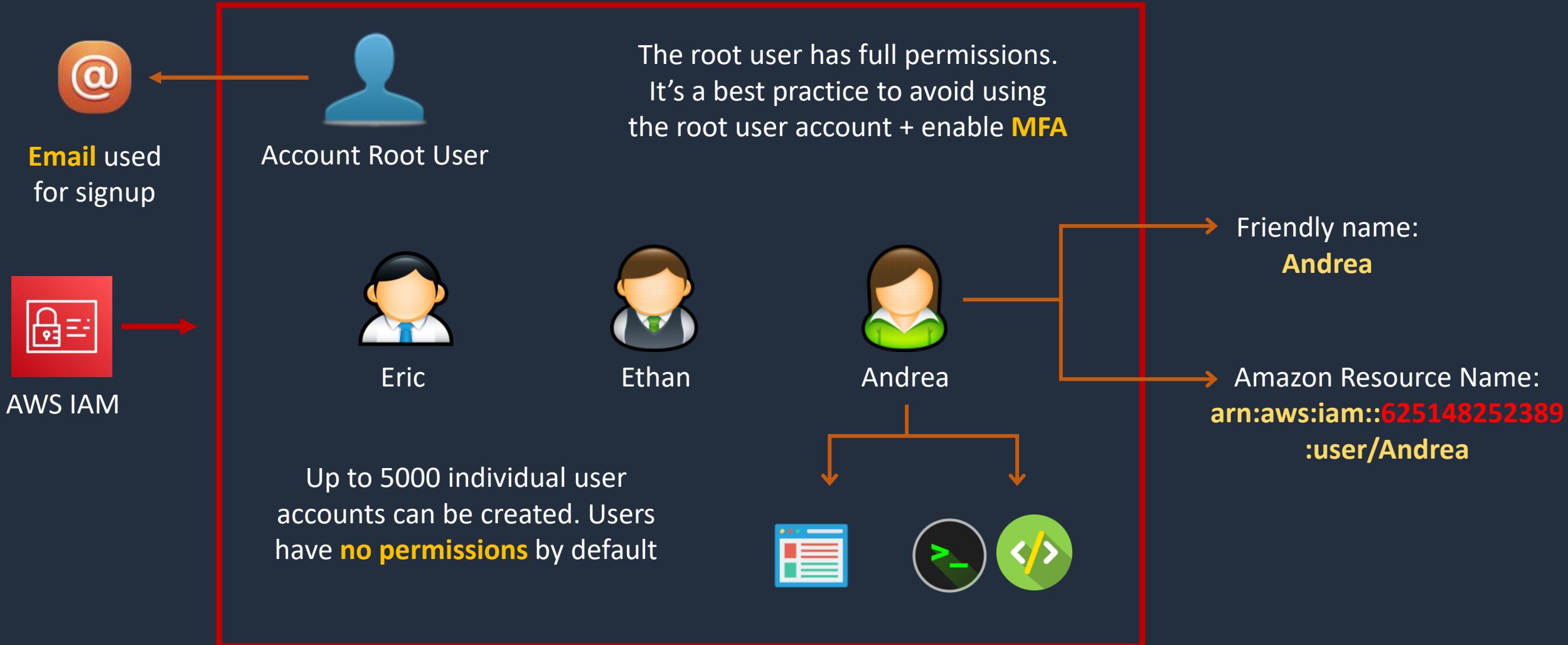


Users, User Groups, Roles and Policies



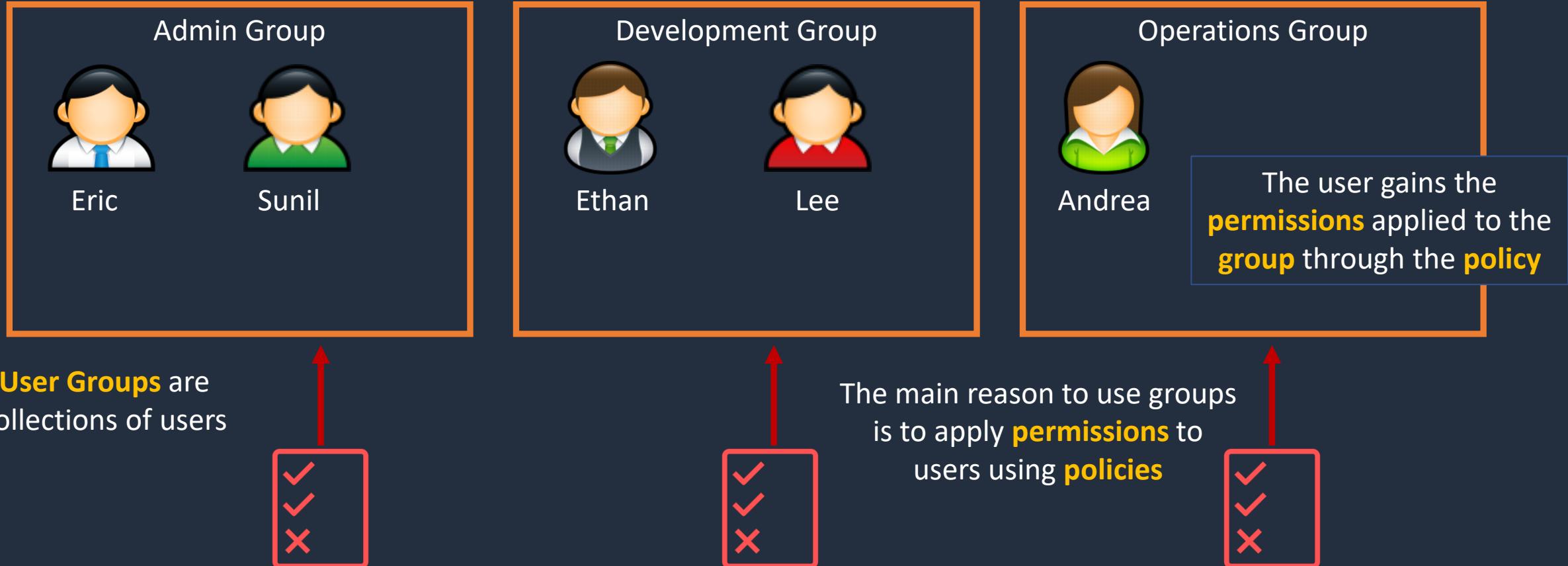


IAM Users



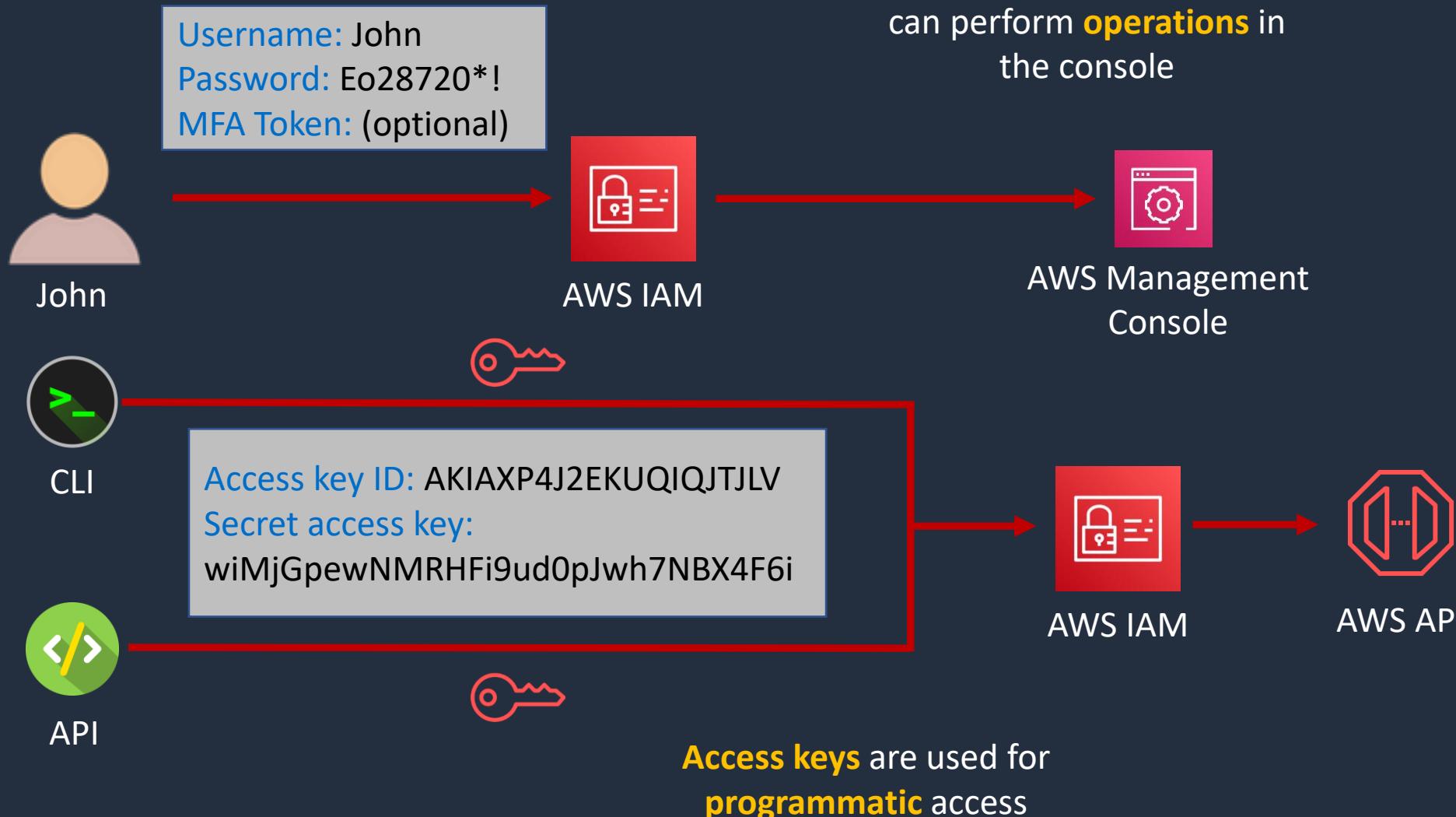


IAM User Groups





IAM Authentication Methods





Root User vs IAM User

User	Login Details	Permissions
 Root User	 Email address	 Full - Unrestricted
 IAM User	Friendly name: John + AWS account ID or Alias	 IAM Permissions Policy

Creating IAM Users and Groups



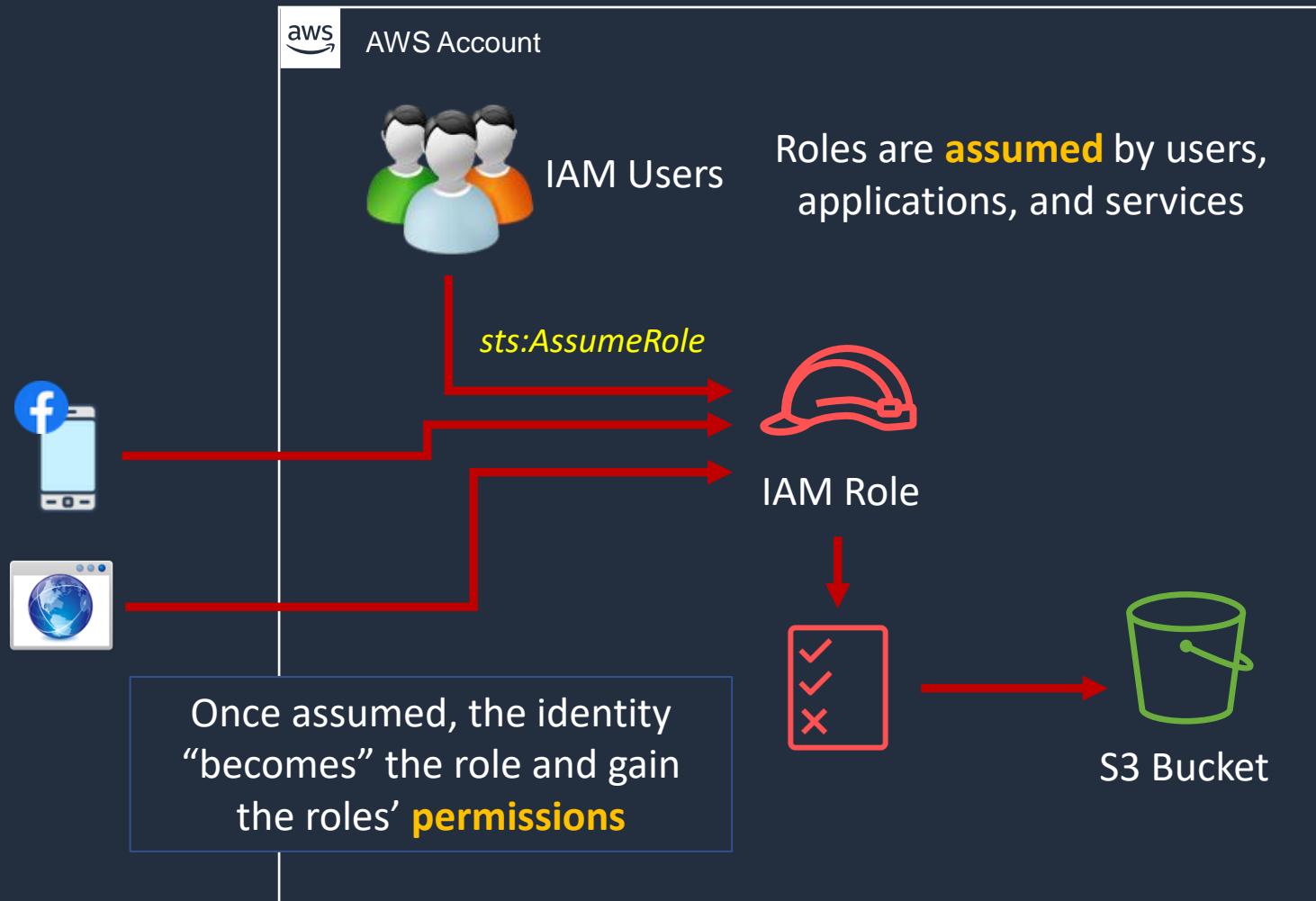
IAM Roles and Policies





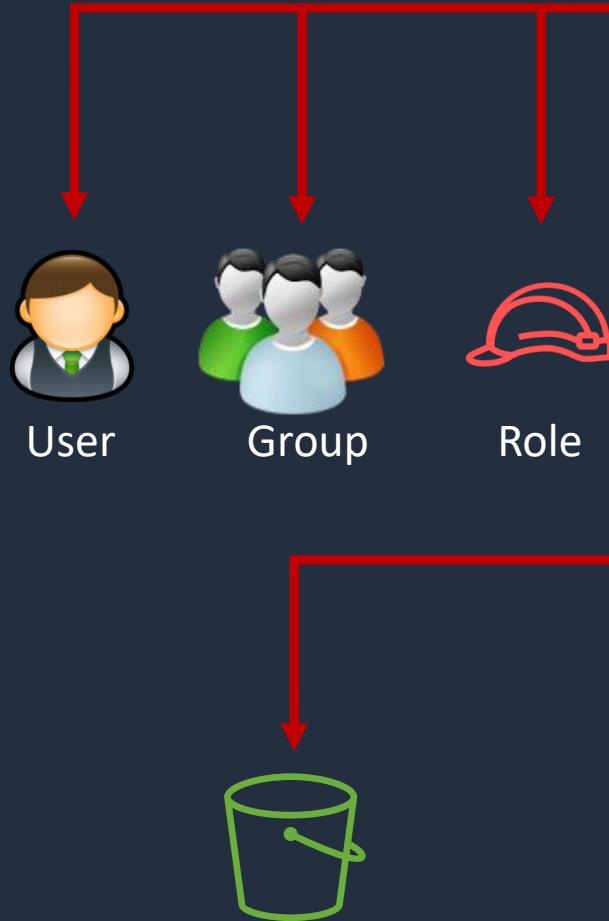
IAM Roles

An **IAM role** is an IAM **identity** that has specific **permissions**





IAM Policies



Policies are **documents** that define **permissions** and are written in JSON



AdministratorAccess

Identity-based policies can be applied to users, groups, and roles



Bucket Policy



S3 Bucket

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

All permissions are **implicitly denied** by default

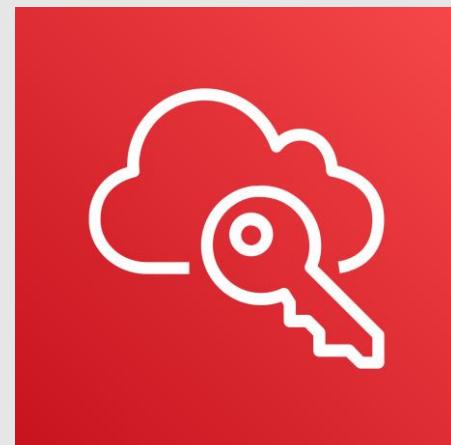
```
{  
  "Version": "2012-10-17",  
  "Id": "Policy1561964929358",  
  "Statement": [  
    {  
      "Sid": "Stmt1561964454052",  
      "Effect": "Allow",  
      "Principal": {  
        "AWS": "arn:aws:iam::515148227241:user/Paul"  
      },  
      "Action": "s3:*",  
      "Resource": "arn:aws:s3:::dctcompany",  
      "Condition": {  
        "StringLike": {  
          "s3:prefix": "Confidential/*"  
        }  
      }  
    }  
  ]  
}
```

Resource-based policies apply to **resources** such as S3 buckets

Switching IAM Roles



IAM Identity Center

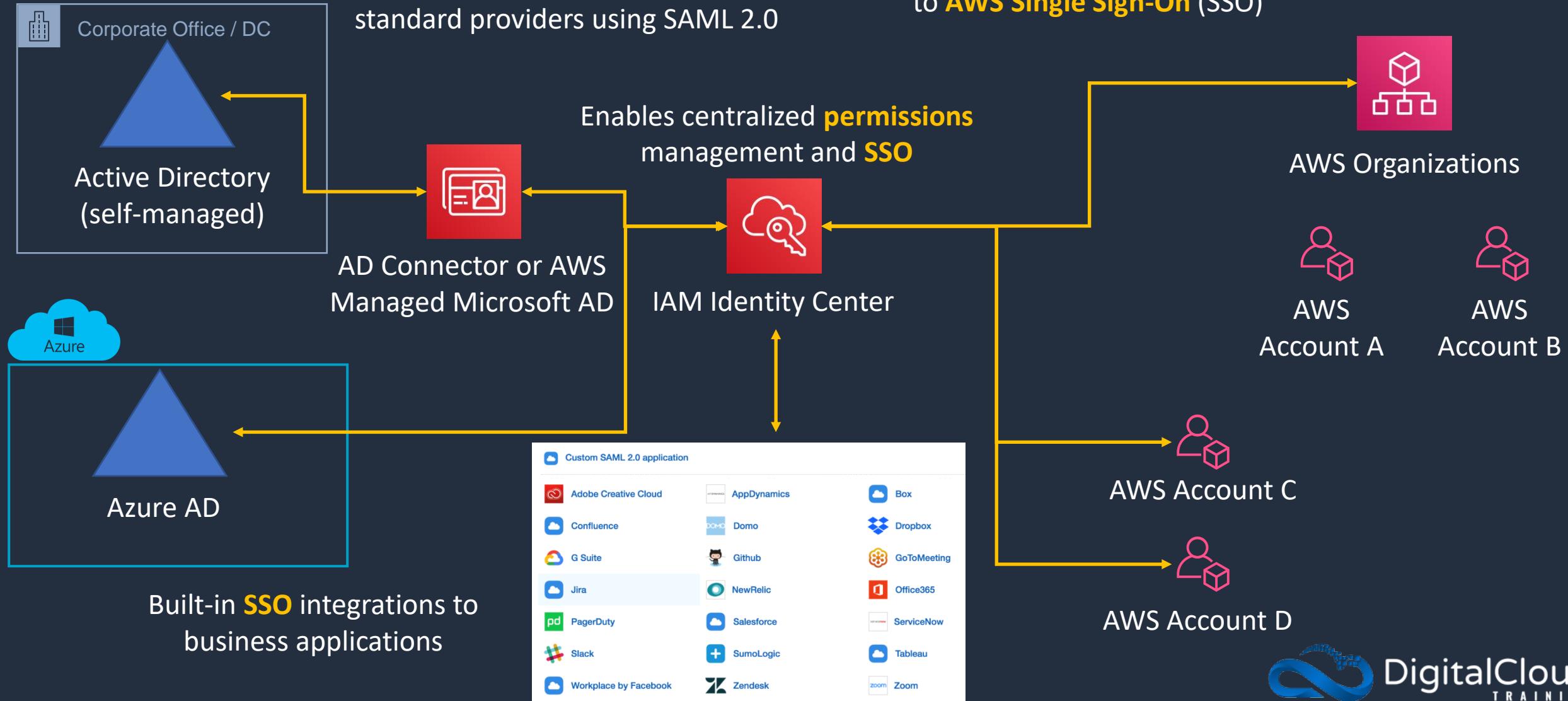




IAM Identity Center

Identity sources can be Identity Center directory, Active Directory and standard providers using SAML 2.0

IAM Identity Center is the successor to **AWS Single Sign-On (SSO)**



Built-in **SSO** integrations to business applications



IAM vs IAM Identity Center

Feature/Use Case	AWS IAM	IAM Identity Center
Primary Purpose	Manage access to AWS services and resources	Centralize identity management and provide single sign-on access to AWS accounts and business applications
Identity Federation	Supports federation with external IdPs using SAML and OpenID Connect	Built-in federation with external IdPs, streamlined for ease of setup and management
Multi-Account Access	Requires more complex setup for cross-account access	Simplifies granting users access to multiple AWS accounts and applications with a single login
Integration with Business Applications	Limited to AWS services; requires custom setup for non-AWS applications	Provides SSO access to commonly used business applications (like Salesforce, Office 365) in addition to AWS accounts
Single Sign-On (SSO) Capability	Enables SSO through federation, but setup is complex and manual	Offers a more user-friendly and simplified SSO setup for both AWS and non-AWS applications



IAM vs IAM Identity Center – Use Cases

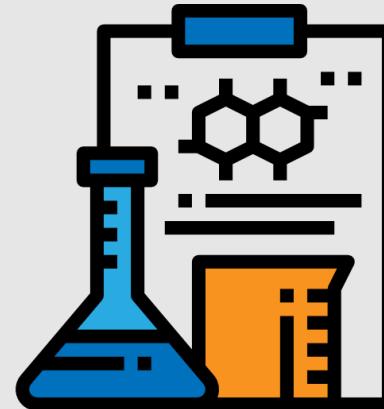
AWS IAM

- Managing AWS resources and permissions
- Creating and managing IAM users and roles within AWS
- Federating with external IdPs for SSO to AWS services
- Programmatic Access Management
- Fine-grained access control

IAM Identity Center

- Providing SSO access to AWS and non-AWS applications
- Centralizing identity management across multiple AWS accounts
- Integrating with external directory services
- Streamlining Access to Business Applications
- Enhancing User Experience with a User Portal

IAM Identity Center in Action



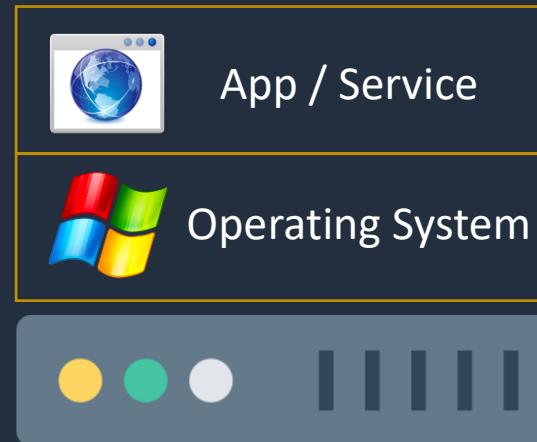
Server Virtualization



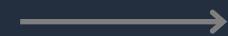


Server Without Virtualization

Application / Service



Operating System



Hardware



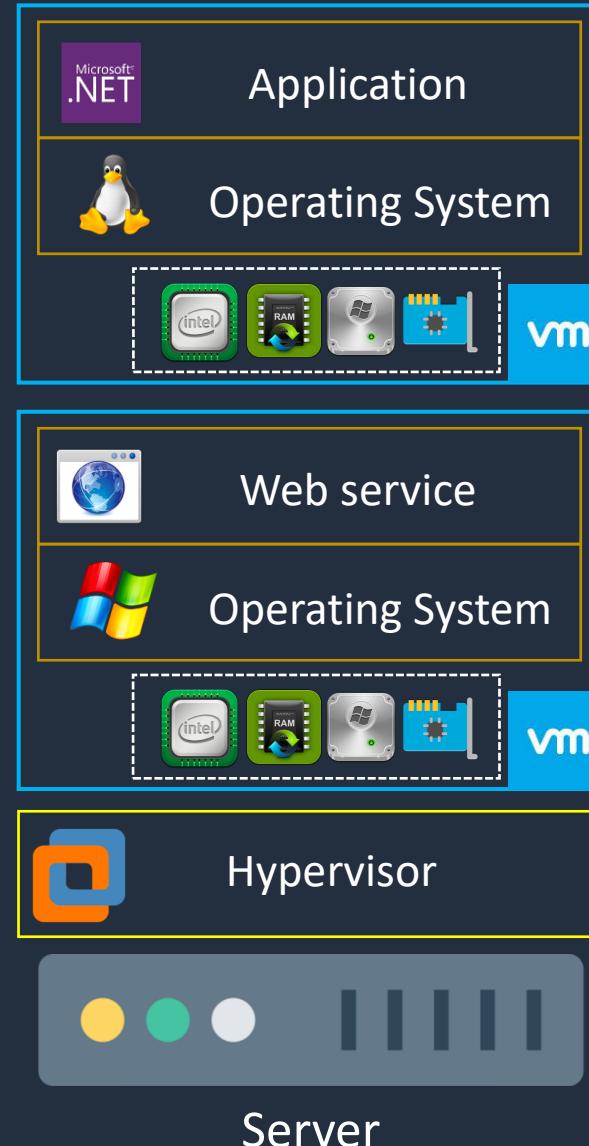
Limitations:

- OS is tied to hardware
- Underutilized resources
- Higher costs
- Scalability constraints
- Longer deployment times

Server With Virtualization

This is known as a **virtual server** or **virtual machine**

The **hypervisor** creates a layer of abstraction

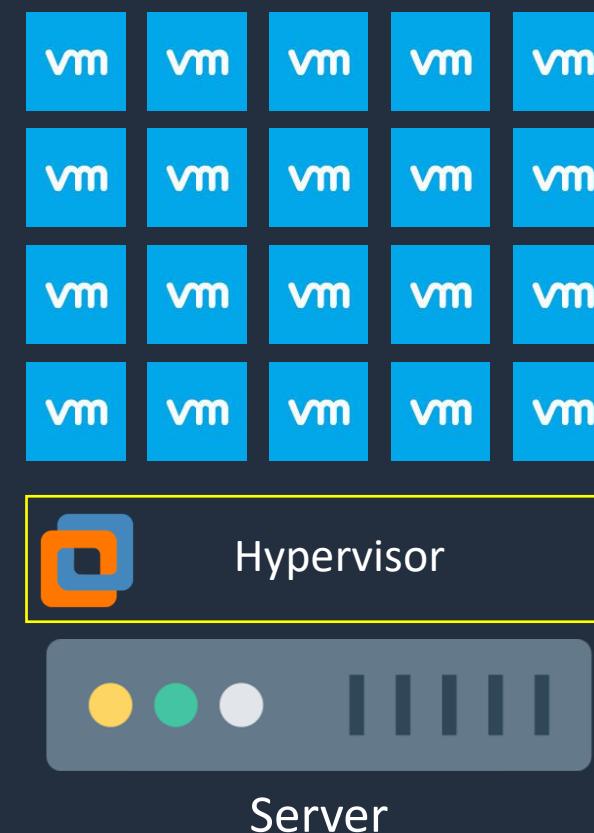


Many VMs can run on the same **physical hardware**

The hypervisor allocates physical resources to the VM



Server With Virtualization

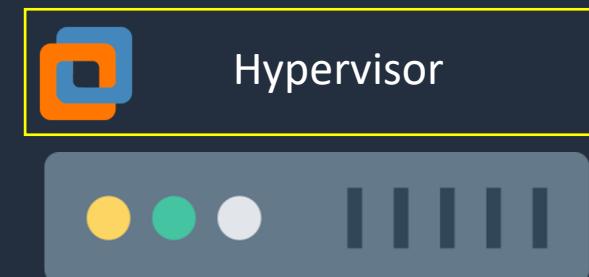




Server Virtualization: Portability



Server

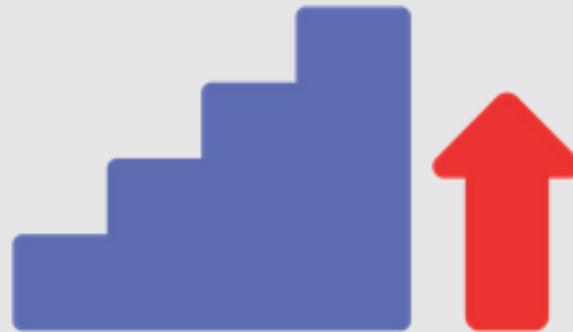


Server

Benefits:

- VM is portable
- Better resource utilization
- Lower costs
- Improved scalability
- Quick deployment times

Scaling Up vs Scaling Out





Stateful vs Stateless Applications

Stateless

No “state” is recorded about the user's session



Person checks a weather website

Stateful

Amazon stores information about **activity**

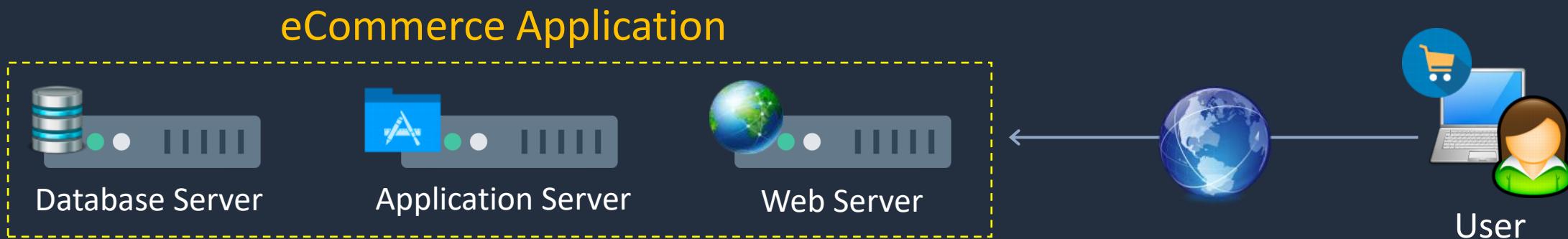


Person browses / purchases on Amazon



Stateful vs Stateless Applications

No data is stored on the web server, it is **stateless**



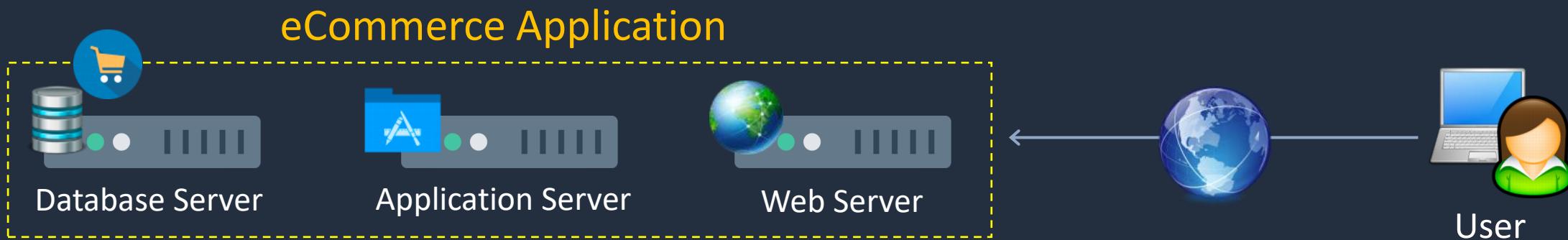
When the user purchases, the application layer processes the order and records the data in the database. This is **stateful**

The cart items are stored in cookies on the computer



Stateful vs Stateless Applications

No data is stored on the web server, it is **stateless**



When the user purchases, the application layer processes the order and records the data in the database. This is **stateful**

The cart items are stored in cookies on the computer



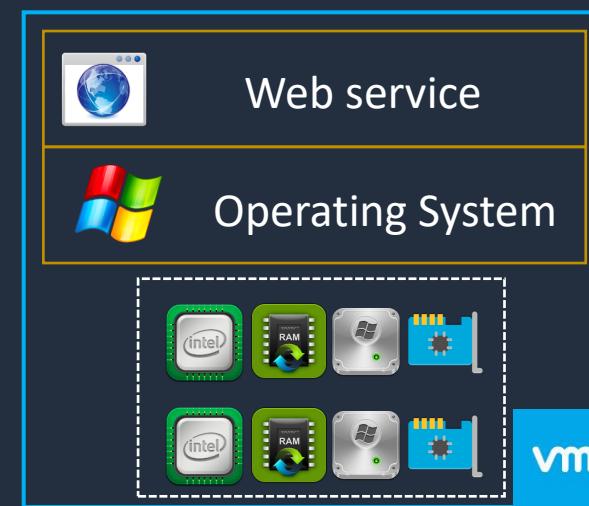
Scalability and Elasticity: Scaling Up





Scalability and Elasticity: Scaling Up

Scaling up means adding resources to the server



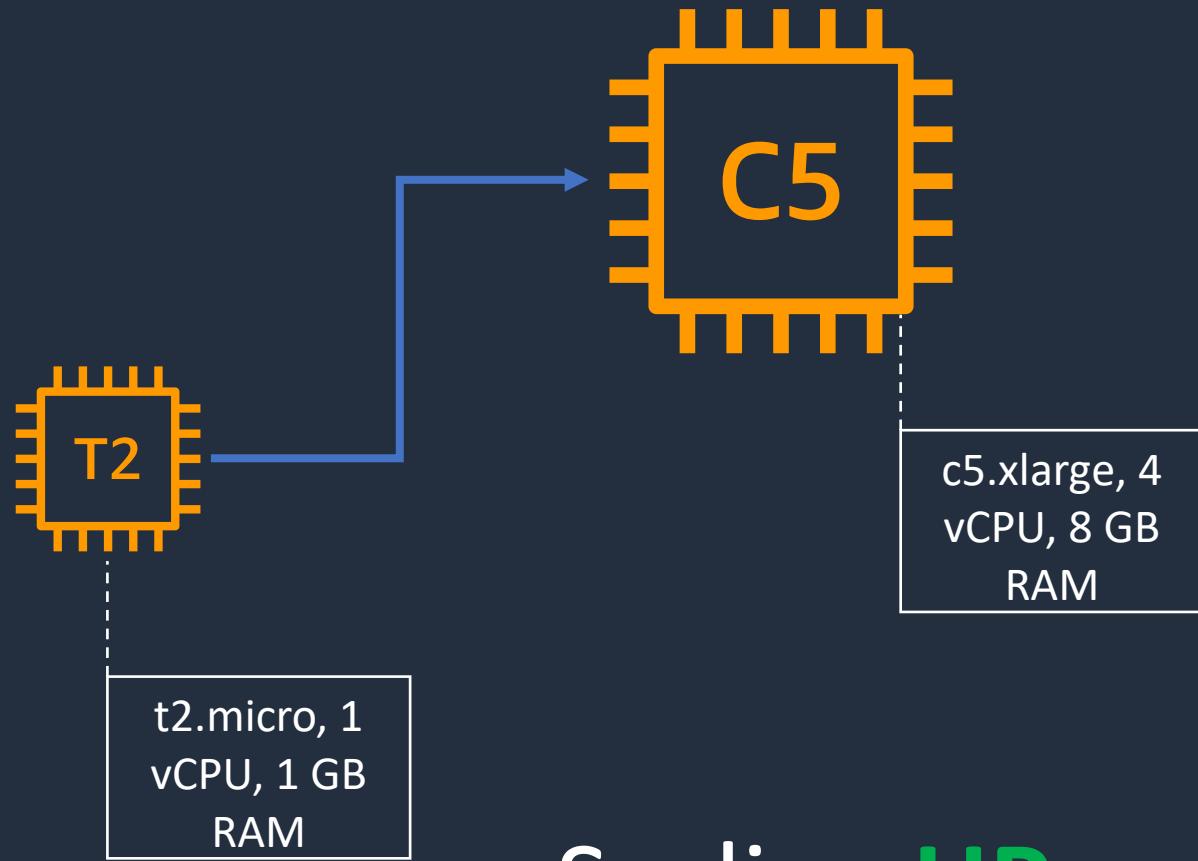


Scalability and Elasticity: Scaling Out

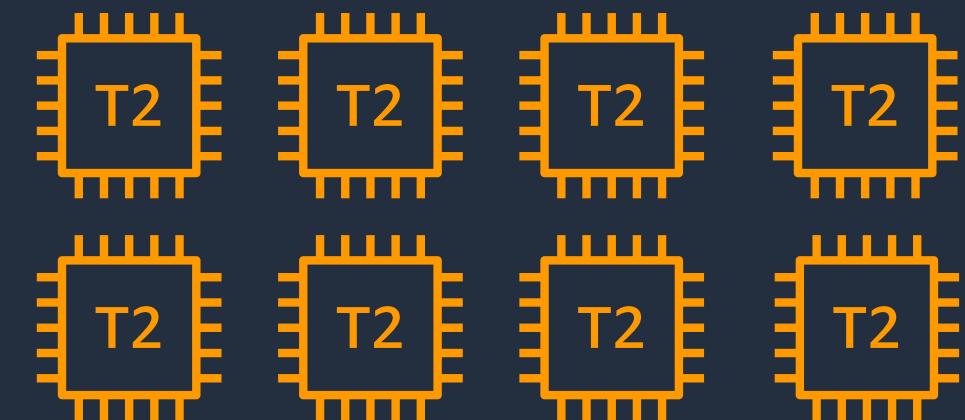




Scaling Up vs Out



Scaling **UP**



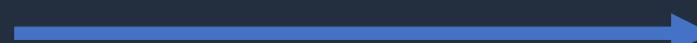
Scaling **OUT**



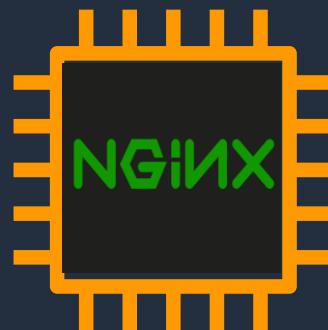
Which scaling model should be used?



EC2 with MySQL DB



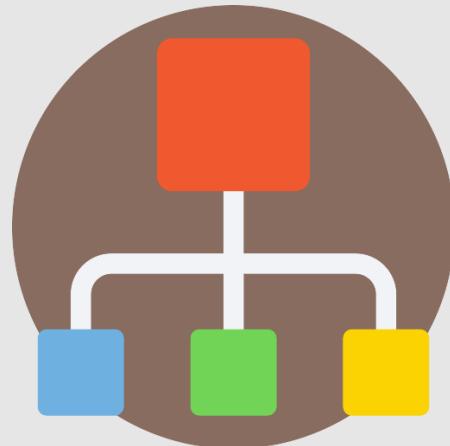
Scale UP



EC2 with **Static** Website

Scale OUT

High Availability and Fault Tolerance





High Availability vs Fault Tolerance

High Availability

- Minimal service interruption
- Designed with no single point of failure (redundancy)
- Uptime measured in %, e.g. 99.99%
- Synchronous or asynchronous replication
- Lower cost compared to FT
- Services that create HA:
 - Elastic Load Balancing
 - EC2 Auto Scaling
 - Amazon Route 53

Fault Tolerance

- No service interruption
- Specialized hardware with instantaneous failover
- No downtime
- Synchronous replication
- Higher cost compared to HA
- Examples of FT:
 - Fault tolerant NICs
 - Disk mirroring (RAID 1)
 - Synchronous DB replication
 - Redundant power



Fault Tolerance

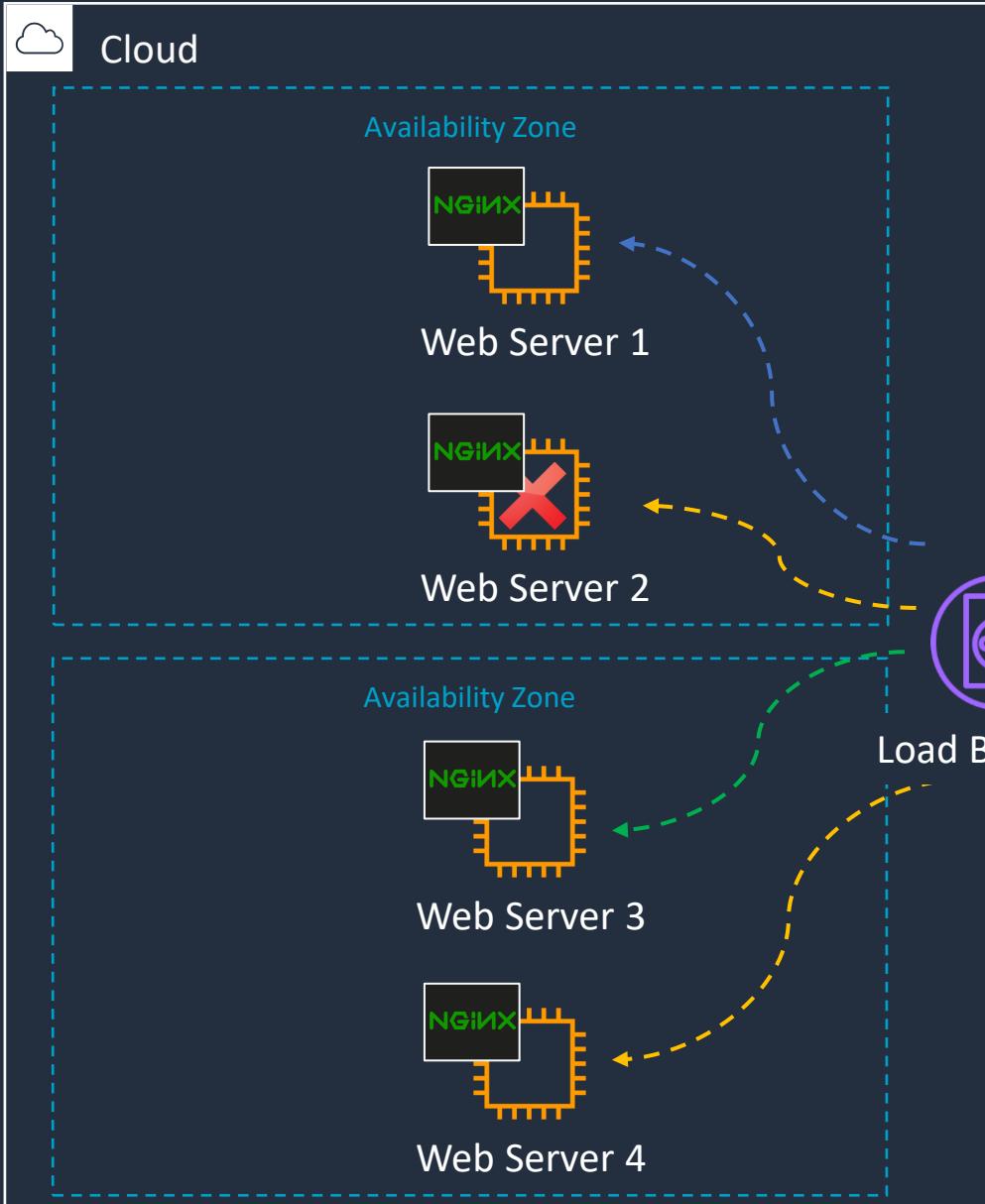
Redundant components allow the system to continue to operate



The system may fail if there is no built-in redundancy



High Availability and Fault Tolerance

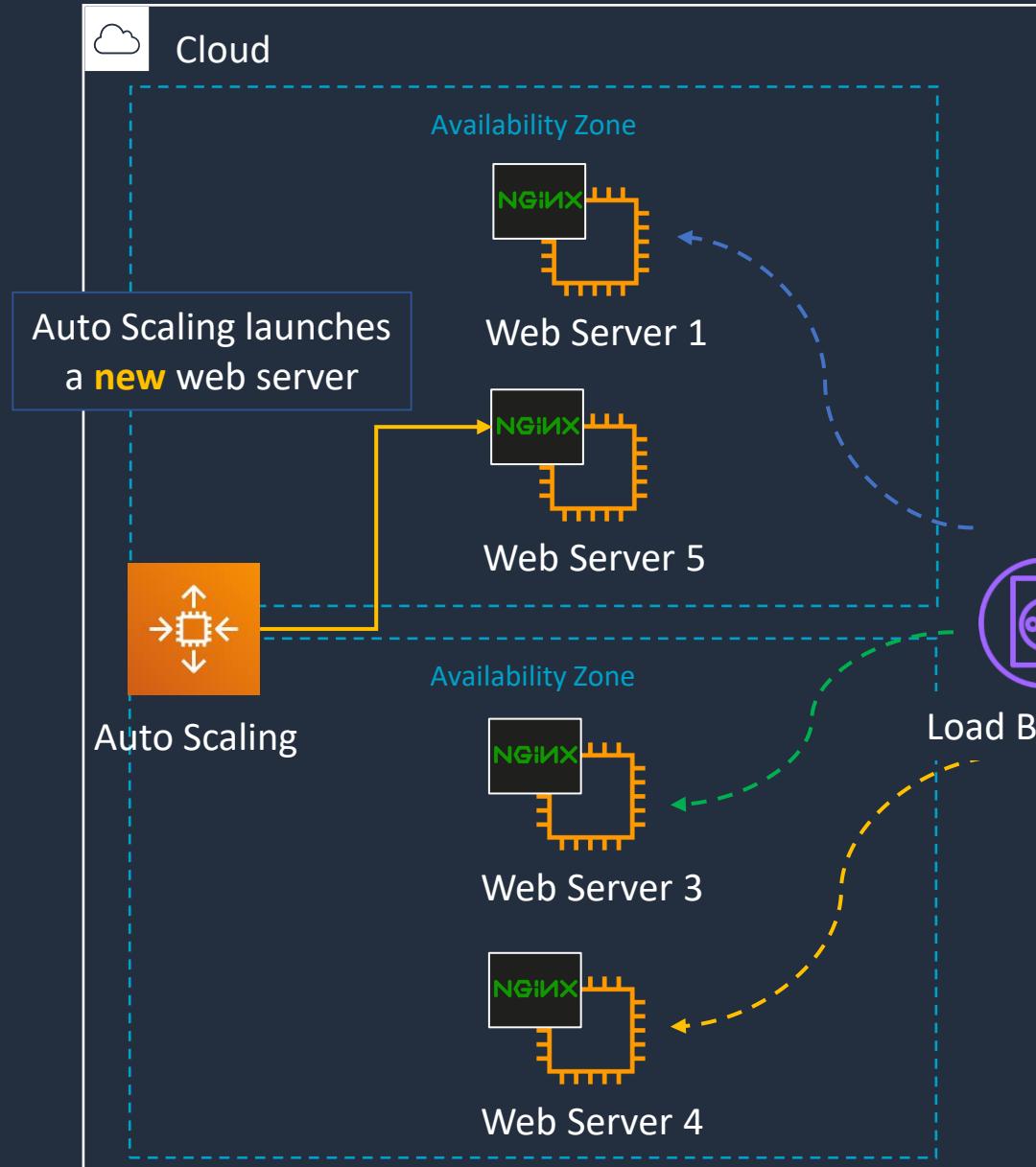


Think of an **availability zone** as a separate data center





High Availability and Fault Tolerance

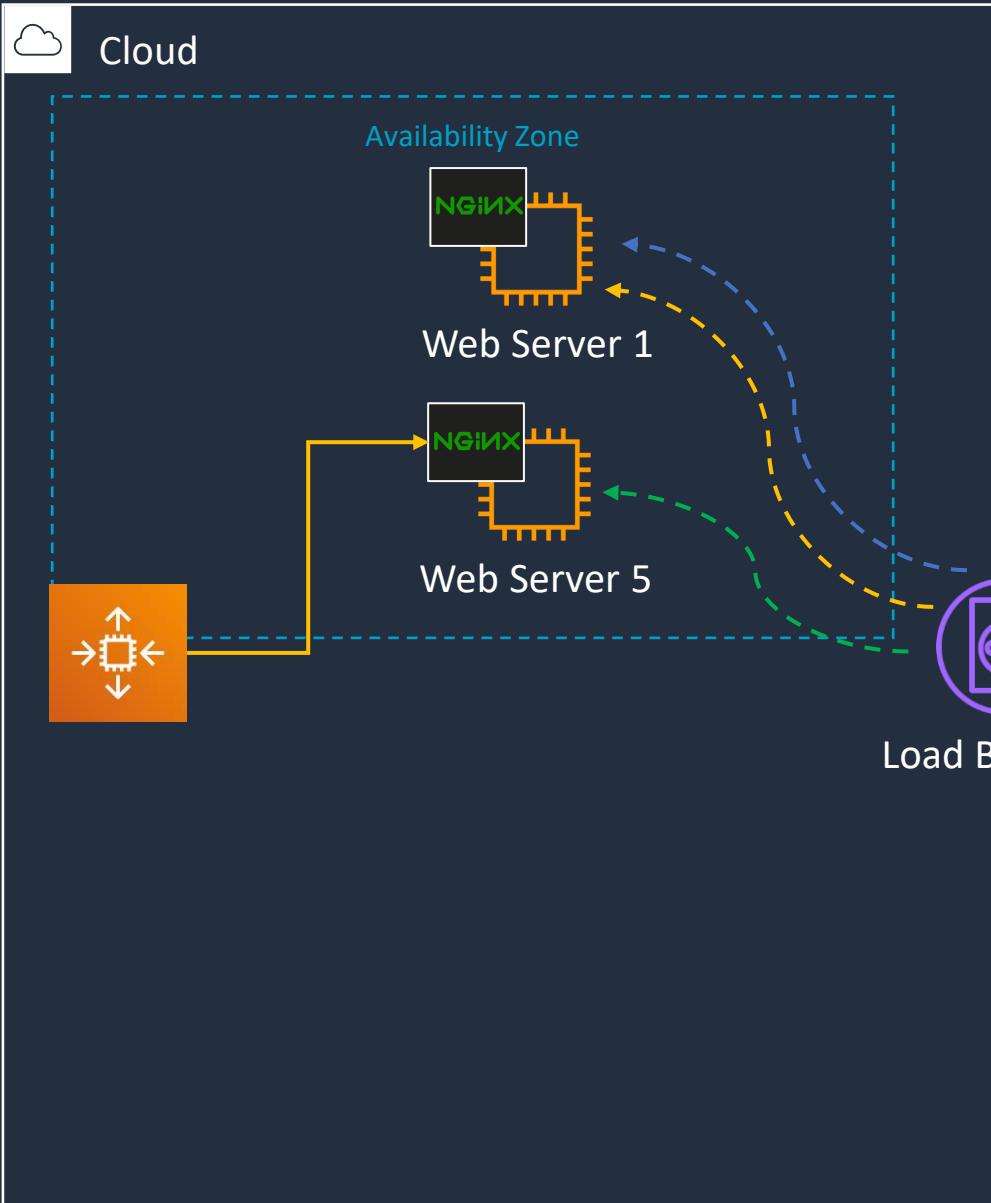


Think of an **availability zone** as a separate data center





High Availability and Fault Tolerance



Think of an **availability zone** as
a separate data center



User 1



User 2



User 3



Durability and Availability

Durability

Durability is protection against:

- Data loss
- Data corruption
- S3 offers 11 9s durability (99.99999999)

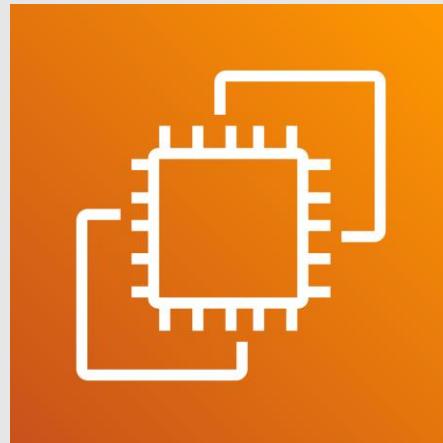
If you store 10 million objects in S3, then you can expect to lose one object every 10,000 years!

Availability

Availability is a measurement of:

- The amount of time the data is available to access
- Expressed as a percent of time per year
- E.g. 99.99%

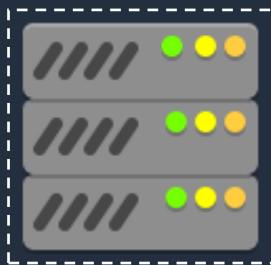
Amazon EC2 Overview



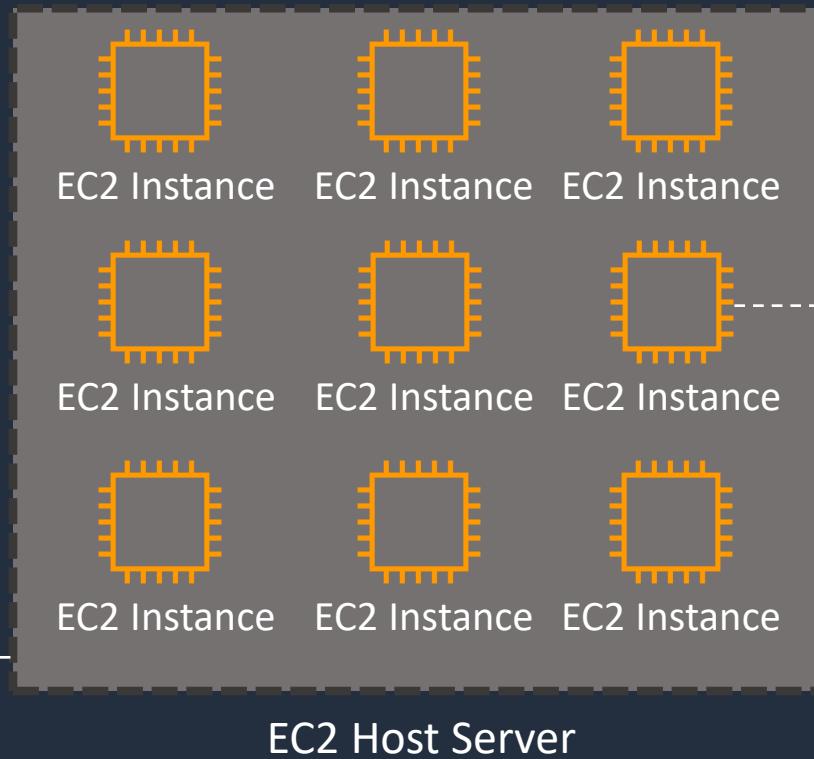


Amazon Elastic Compute Cloud (EC2)

EC2 hosts are
managed by AWS



EC2 instances run Windows,
Linux, or MacOS



An **EC2 instance** is
a virtual server



A selection of **instance types**
come with varying combinations
of CPU, memory, storage and
networking

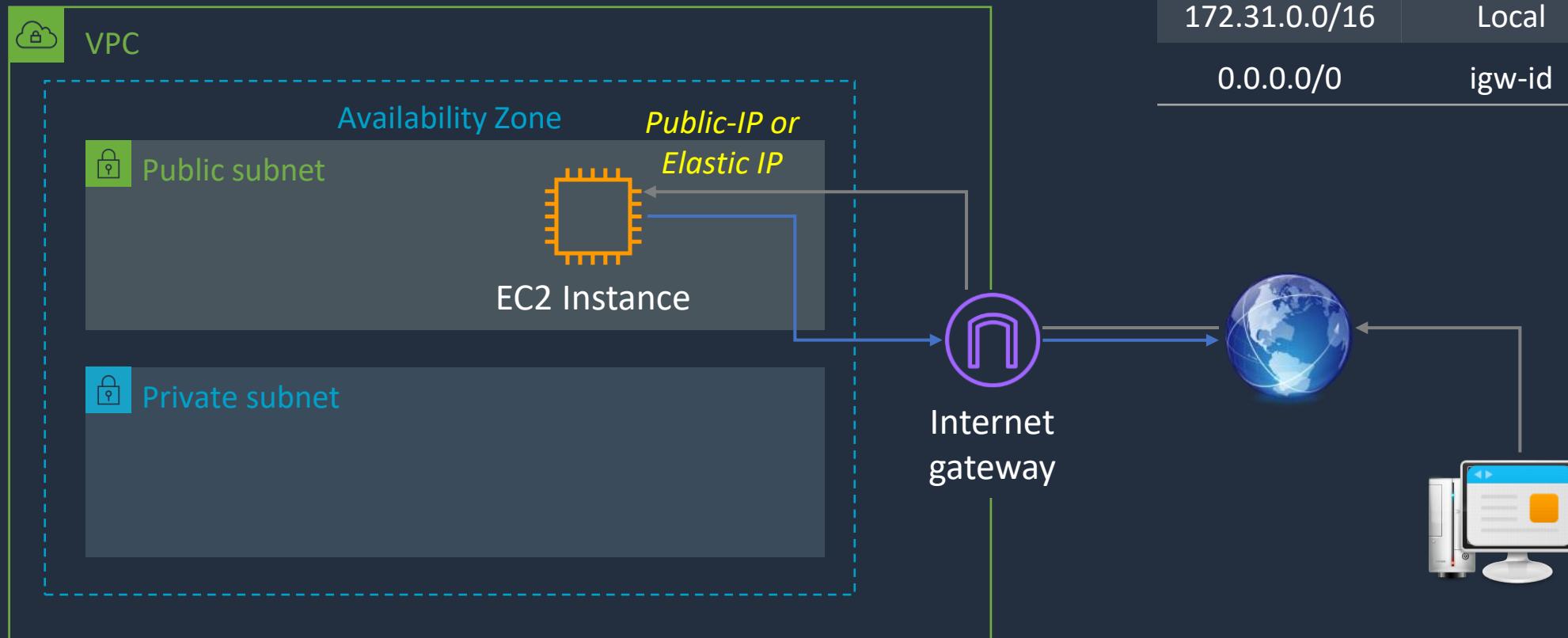


Public, Private, and Elastic IP addresses

Type	Description
Public IP address	<p>Lost when the instance is stopped</p> <p>Used in Public Subnets</p> <p>No charge</p> <p>Associated with a private IP address on the instance</p> <p>Cannot be moved between instances</p>
Private IP address	<p>Retained when the instance is stopped</p> <p>Used in Public and Private Subnets</p>
Elastic IP address	<p>Static Public IP address</p> <p>You are charged if not used</p> <p>Associated with a private IP address on the instance</p> <p>Can be moved between instances and Elastic Network Adapters</p>



Public Subnets

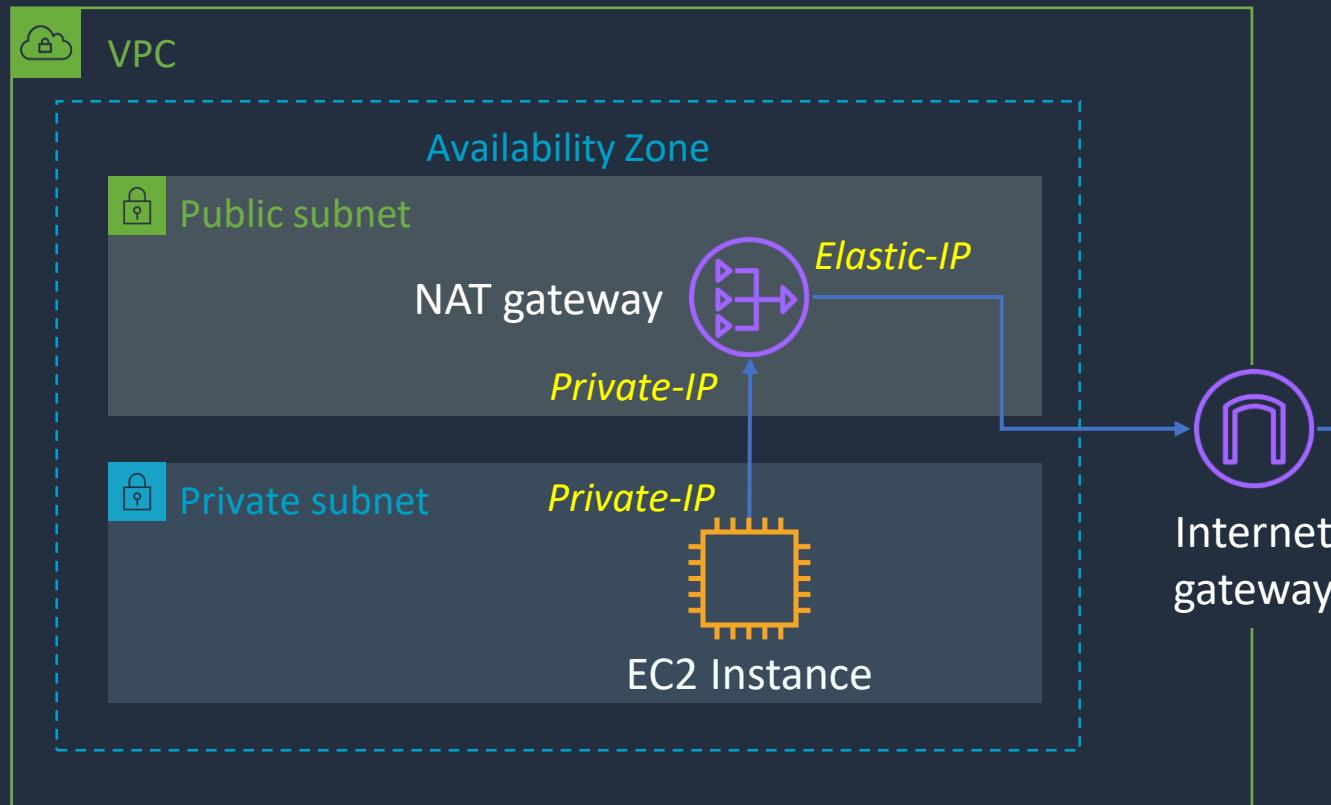


Public Subnet Route Table

Destination	Target
172.31.0.0/16	Local
0.0.0.0/0	igw-id



Public Subnets



Public Subnet Route Table

Destination	Target
172.31.0.0/16	Local
0.0.0.0/0	igw-id

Private Subnet Route Table

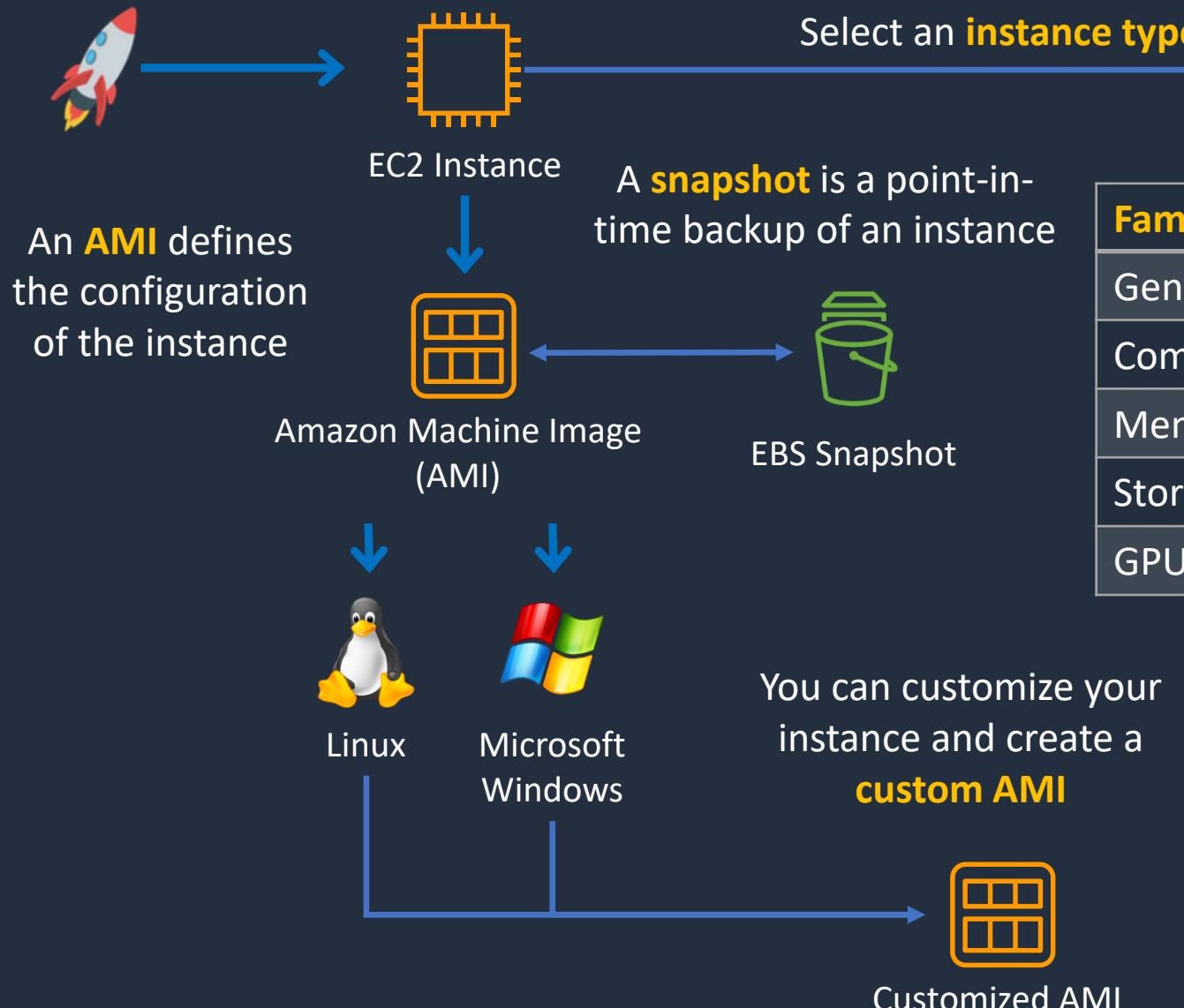
Destination	Target
172.31.0.0/16	Local
0.0.0.0/0	nat-gateway-id

Launching Amazon EC2 Instances





Launching an EC2 Instance



Connecting to Amazon EC2

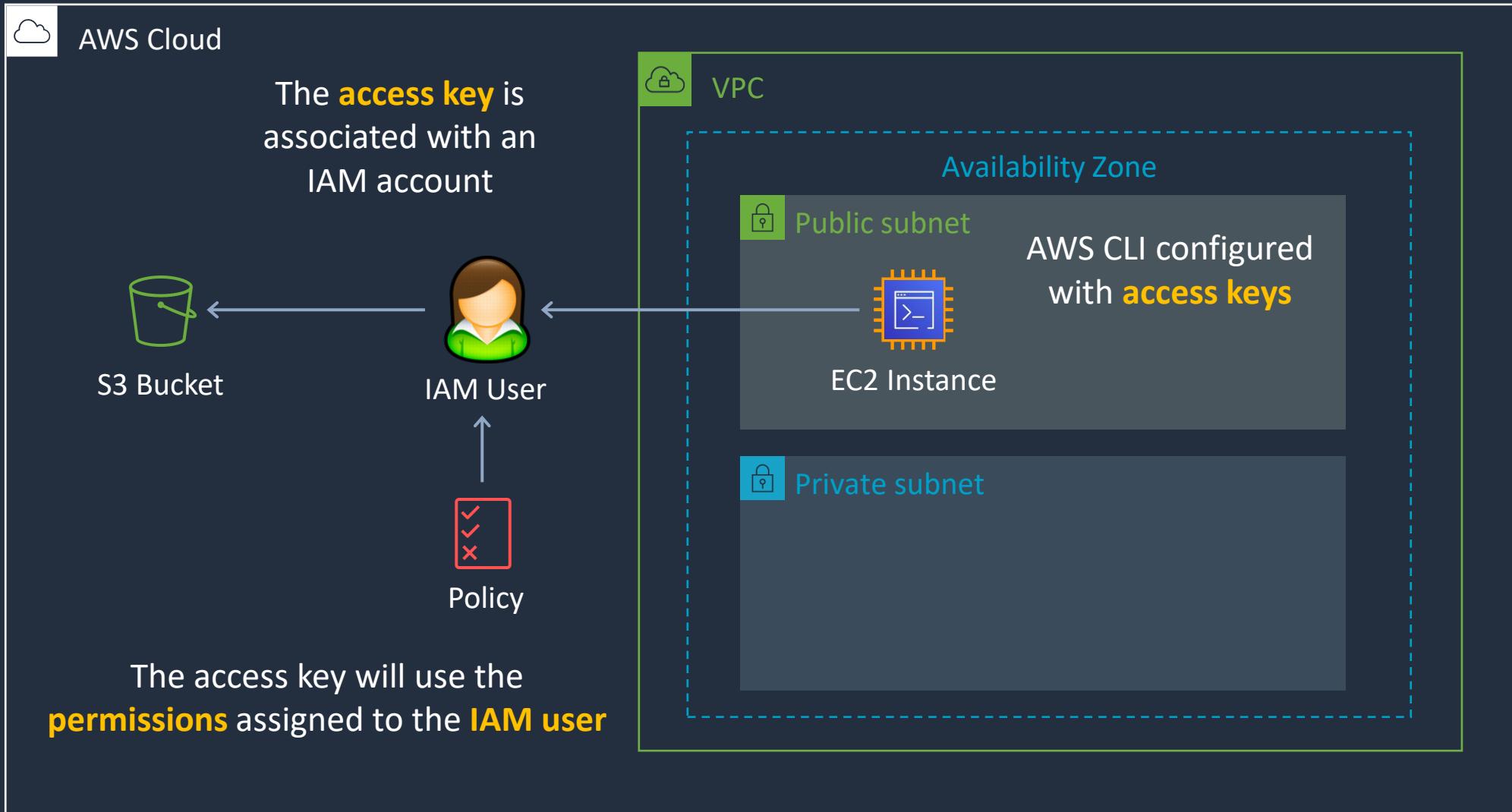


Access Keys and IAM Roles with EC2



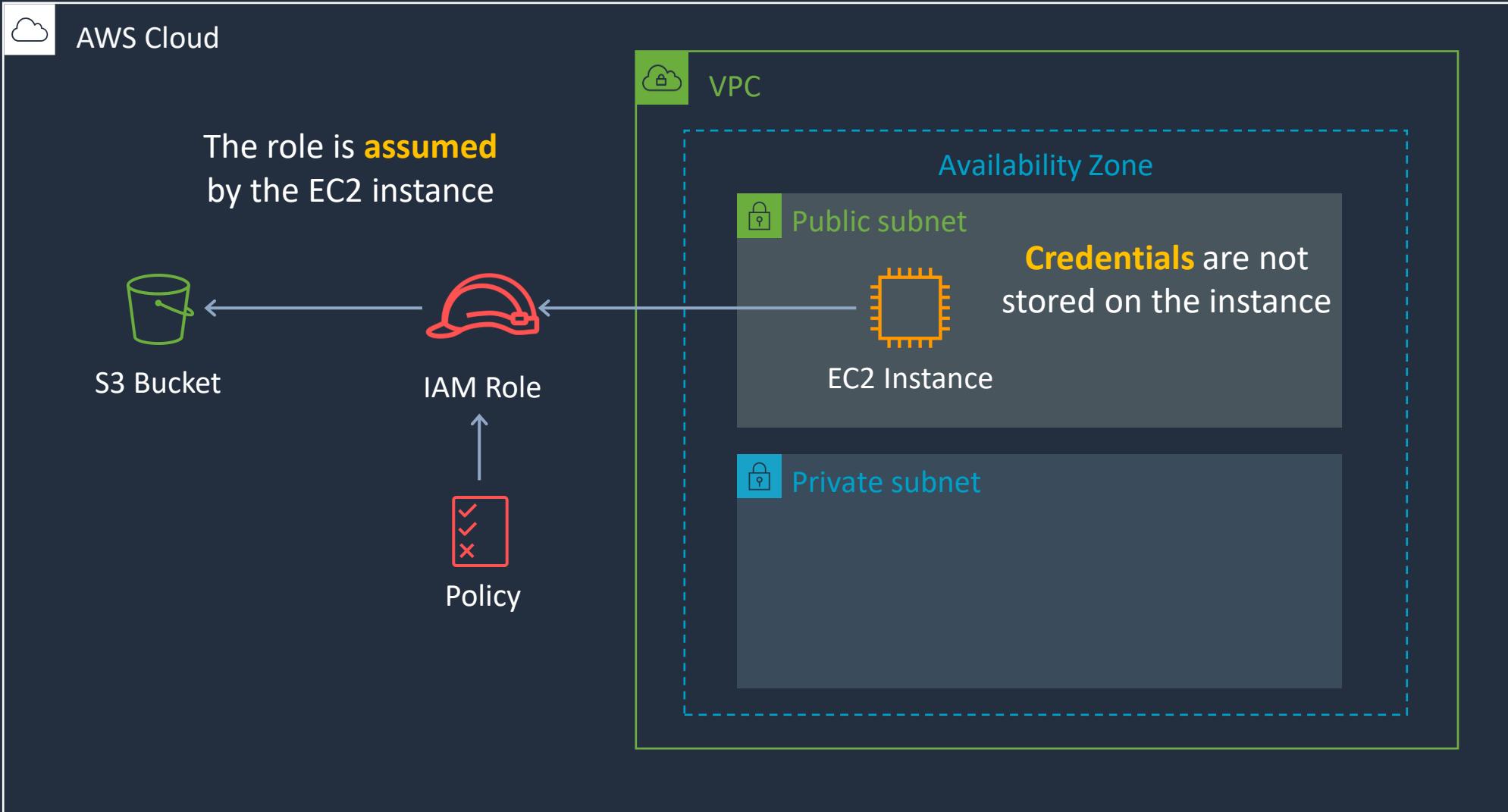


Using Access Keys with Amazon EC2





Using Access Keys with Amazon EC2



Practice with Access Keys and IAM Roles



Create a Website with User Data





Amazon EC2 User Data

The code is run when the instance starts for the **first time**

AWS Management Console

Batch and **PowerShell** scripts can be run on Windows

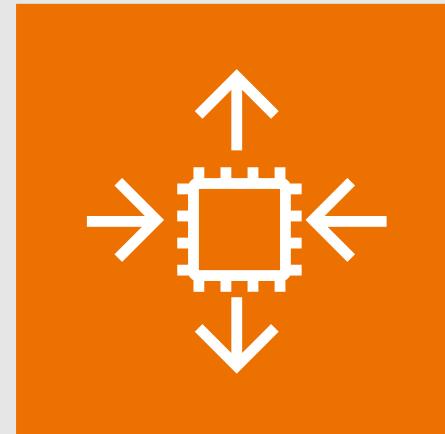
```
#!/bin/bash
yum update -y
yum install -y httpd
systemctl start httpd
systemctl enable httpd
```

Limited to
16 KB



EC2 Instance with a **web service** is launched

Amazon EC2 Auto Scaling





Amazon EC2 Auto Scaling

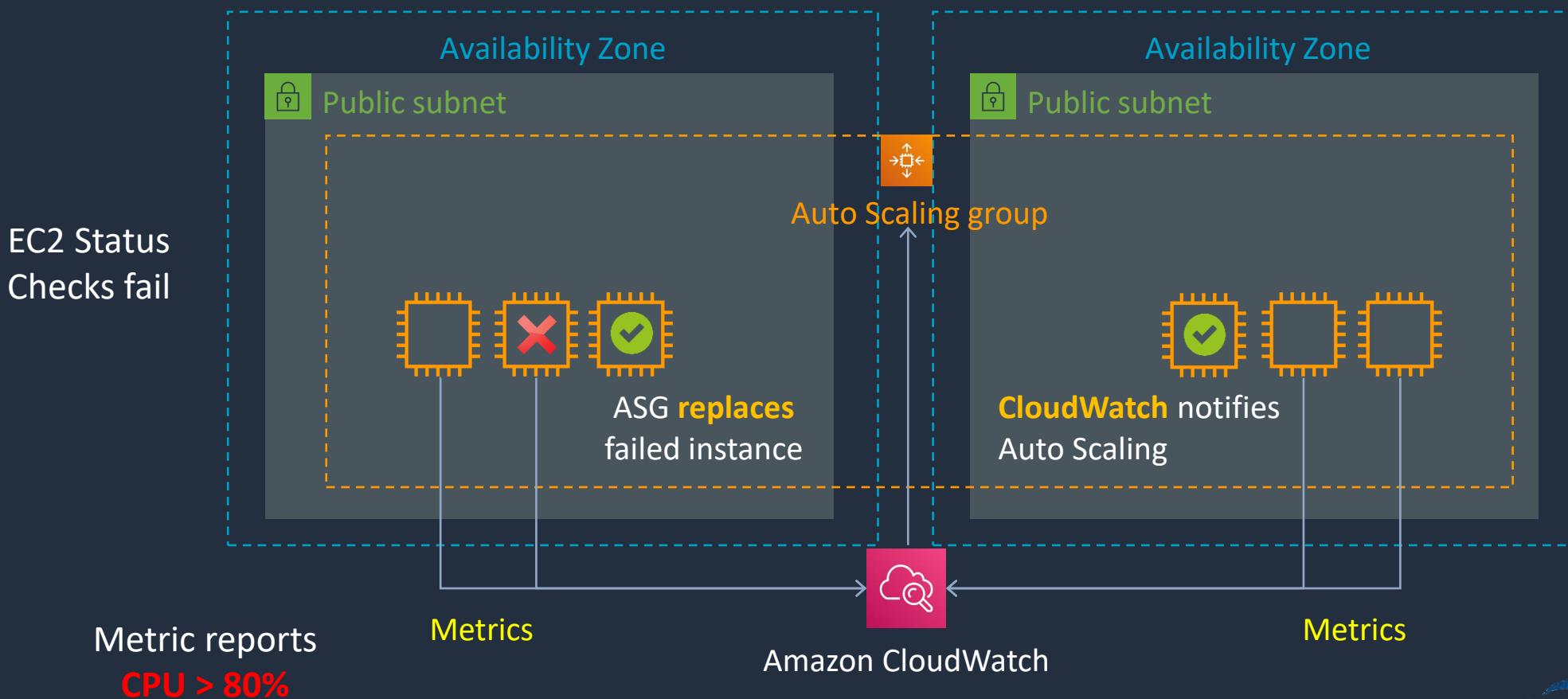
- Automatically **launches** and **terminates** instances
- Maintain **availability** and **scale** capacity
- Works with EC2, ECS, and EKS
- Integrates with many AWS services, including:
 - CloudWatch for monitoring and scaling
 - Elastic Load Balancing for distributing connections
 - EC2 Spot Instances for cost optimization
 - Amazon VPC for deploying instances across AZs



Amazon EC2 Auto Scaling

1. Automatic scaling
2. Maintaining availability

Auto Scaling launches
an extra instance





Amazon EC2 Auto Scaling

- Scaling is horizontal (scales out)
- Provides **elasticity** and **scalability**
- Responds to EC2 status checks and CloudWatch metrics
- Can scale based on demand (performance) or on a schedule
- Scaling policies define how to respond to changes in demand



Configuration of an Auto Scaling Group

A **Launch Template**

specifies the EC2 instance configuration



Launch Template

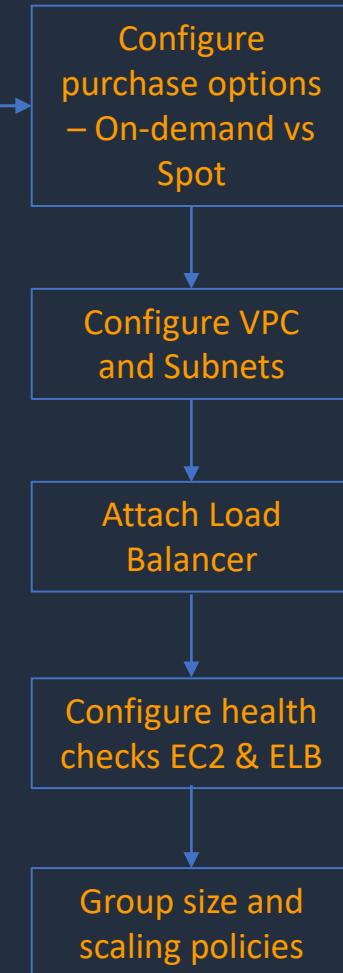
- AMI and instance type
- EBS volumes
- Security groups
- Key pair
- IAM instance profile
- User data
- Shutdown behavior
- Termination protection
- Placement group name
- Capacity reservation
- Tenancy
- Purchasing option (e.g. Spot)



Launch Config

- AMI and instance type
- EBS volumes
- Security groups
- Key pair
- Purchasing option (e.g. Spot)
- IAM instance profile
- User data

Launch Configurations are replaced by launch templates and have fewer features





Amazon EC2 Auto Scaling

- **Health checks**
 - EC2 = EC2 status checks
 - ELB = Uses the ELB health checks **in addition** to EC2 status checks
- **Health check grace period**
 - How long to wait before checking the health status of the instance
 - Auto Scaling does not act on health checks until grace period expires



Amazon EC2 Auto Scaling

Types of Auto Scaling:

- **Manual** – make changes to ASG size manually
- **Dynamic** – automatically scales based on demand
- **Predictive** – uses Machine Learning to predict
- **Scheduled** – scales based on a schedule

Create an Auto Scaling Group



Amazon Elastic Load Balancing





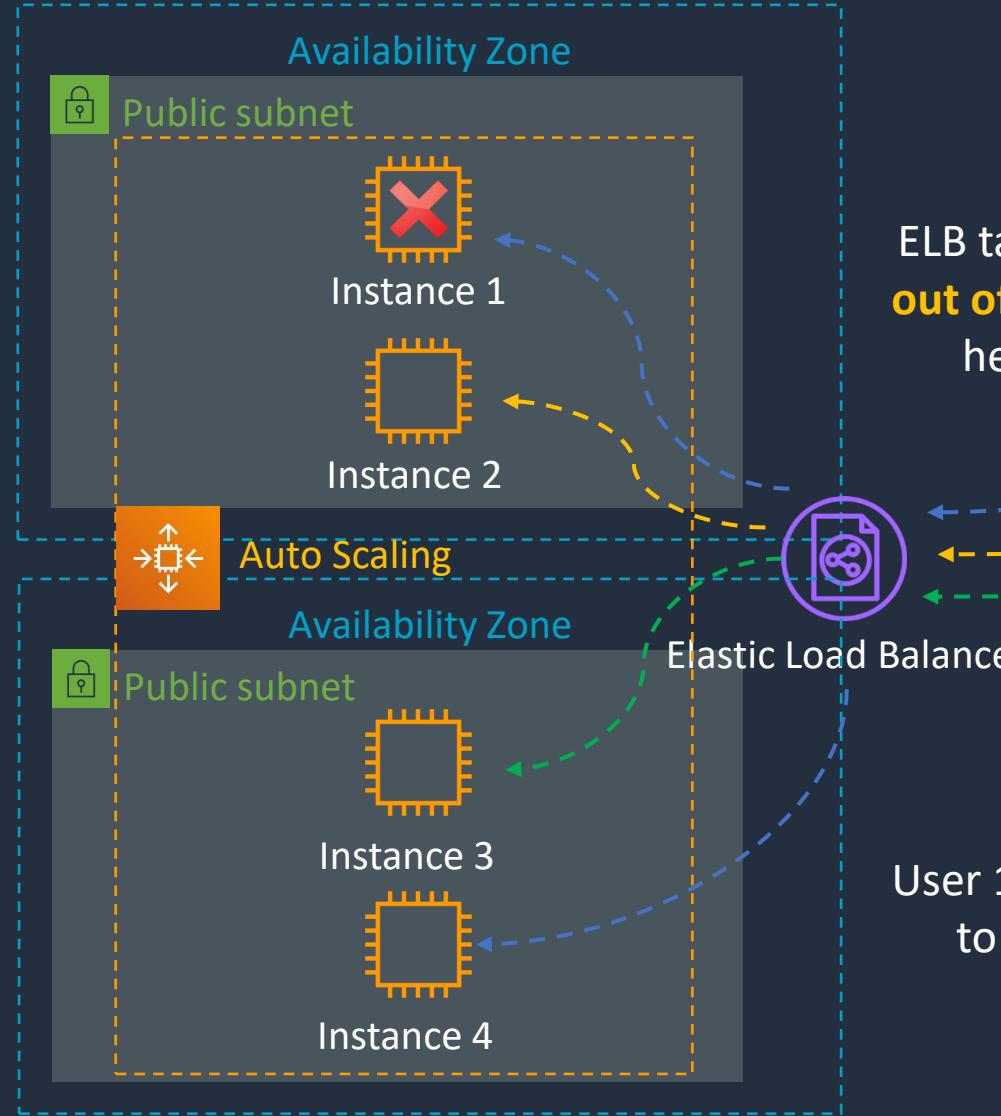
Amazon Elastic Load Balancing

- Provides high availability and fault tolerance
- Targets include:
 - Amazon EC2 instances
 - Amazon ECS containers
 - IP addresses
 - Lambda functions
 - Other load balancers



Amazon Elastic Load Balancing

EC2 Auto Scaling
terminates
instance 1



ELB takes instance 1
out of service (failed
health check)

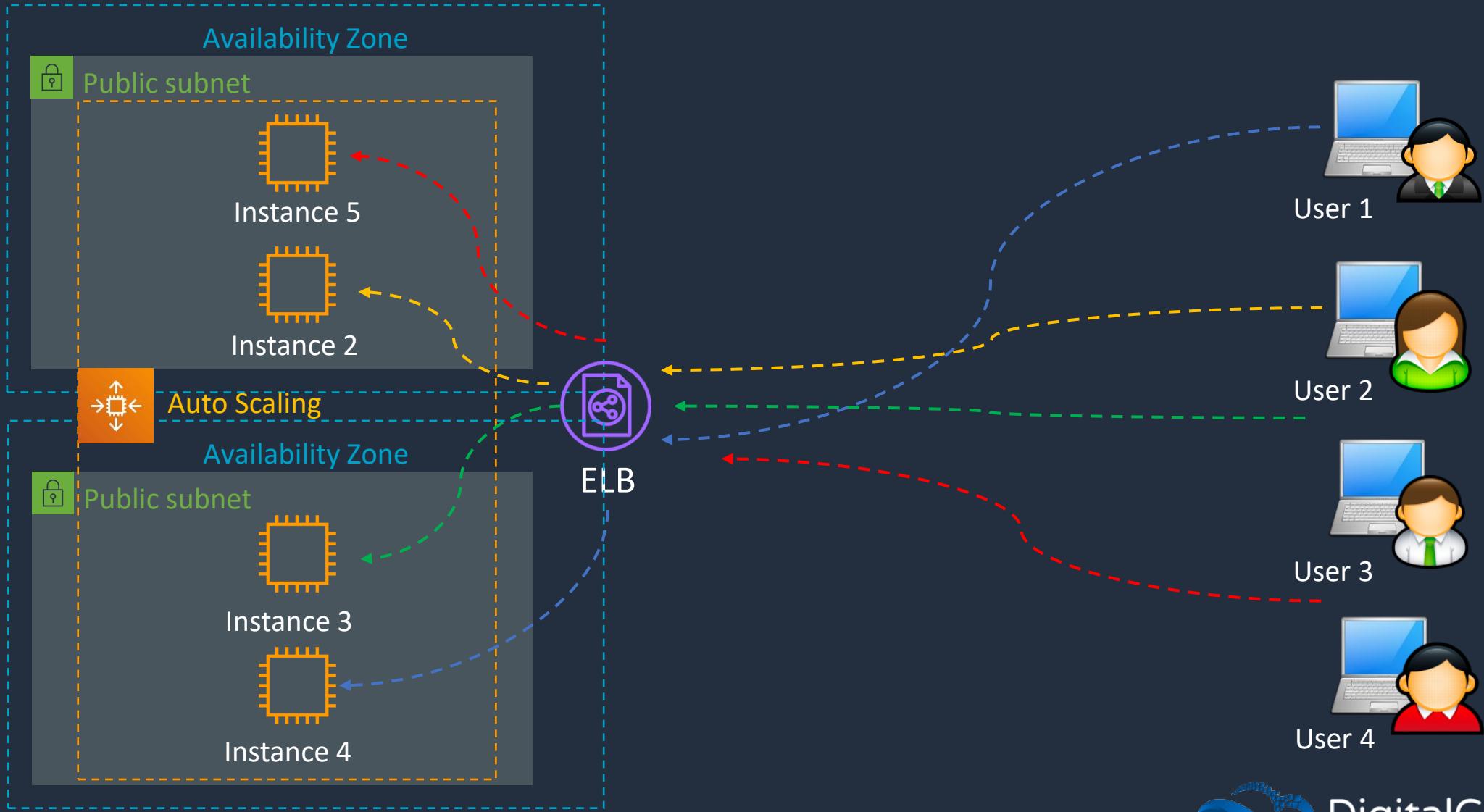
User 1 is connected
to **instance 4**





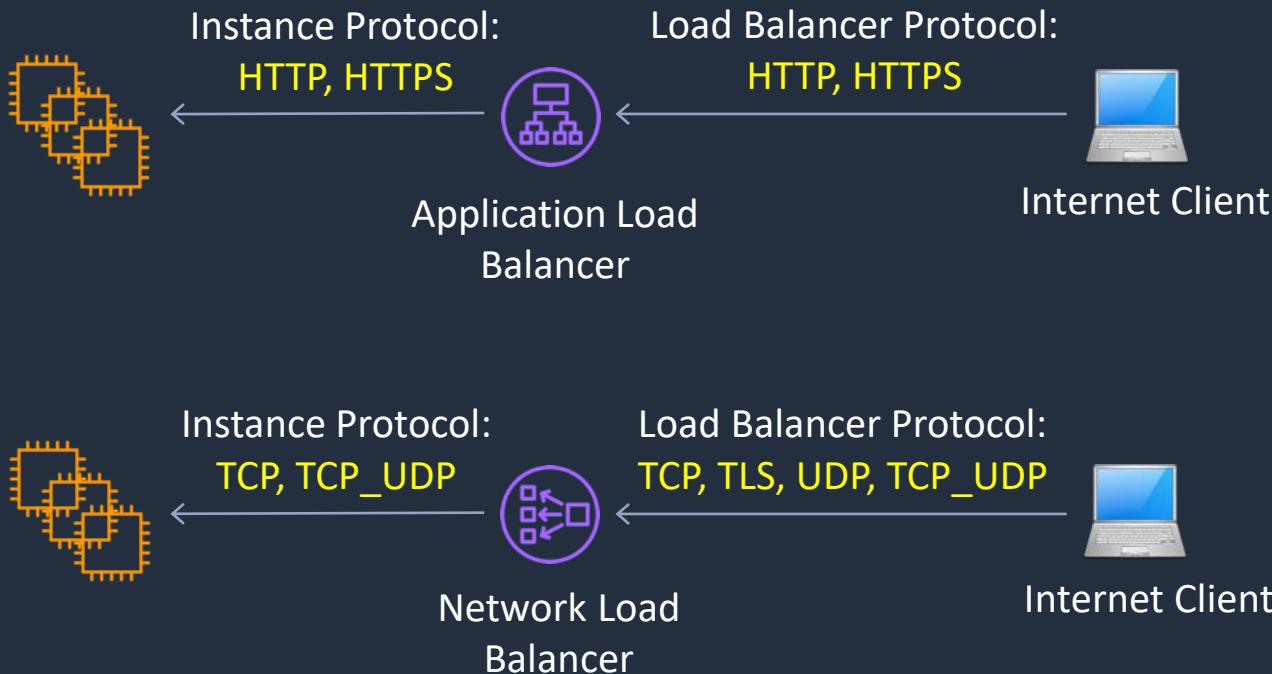
Amazon Elastic Load Balancing

EC2 Auto Scaling
launches
instance 5





Types of Elastic Load Balancer (ELB)



Application Load Balancer

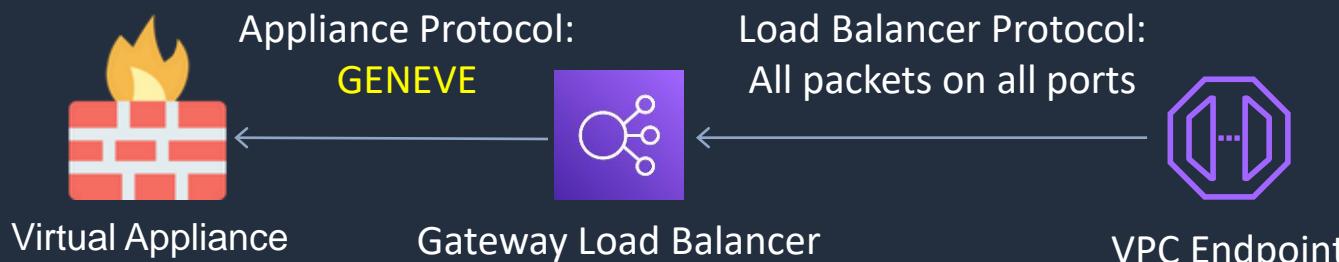
- Operates at the request level
- Routes based on the content of the request (L7)
- Supports path-based routing, host-based routing, query string parameter-based routing, and source IP address-based routing
- Supports instances, IP addresses, Lambda functions and containers as targets

Network Load Balancer

- Operates at the connection level
- Routes connections based on IP protocol data (L4)
- Offers ultra high performance, low latency and TLS offloading at scale
- Can have a static IP / Elastic IP
- Supports UDP and static IP addresses as targets



Types of Elastic Load Balancer (ELB)



Gateway Load Balancer

- Used in front of virtual appliances such as firewalls, IDS/IPS, and deep packet inspection systems
- Operates at Layer 3 – listens for all packets on all ports
- Forwards traffic to the TG specified in the listener rules
- Exchanges traffic with appliances using the GENEVE protocol on port 6081



ELB Use Cases

Application Load Balancer

- Web applications with L7 routing (HTTP/HTTPS)
- Microservices architectures (e.g. Docker containers)
- Lambda targets

Network Load Balancer

- TCP and UDP based applications
- Ultra-low latency
- Static IP addresses
- VPC endpoint services



ELB Use Cases

Gateway Load Balancer

- Deploy, scale and manage 3rd party virtual network appliances
- Centralized inspection and monitoring
- Firewalls, intrusion detection and prevention systems, and deep packet inspection systems

Create an Application Load Balancer



Create a Scaling Policy



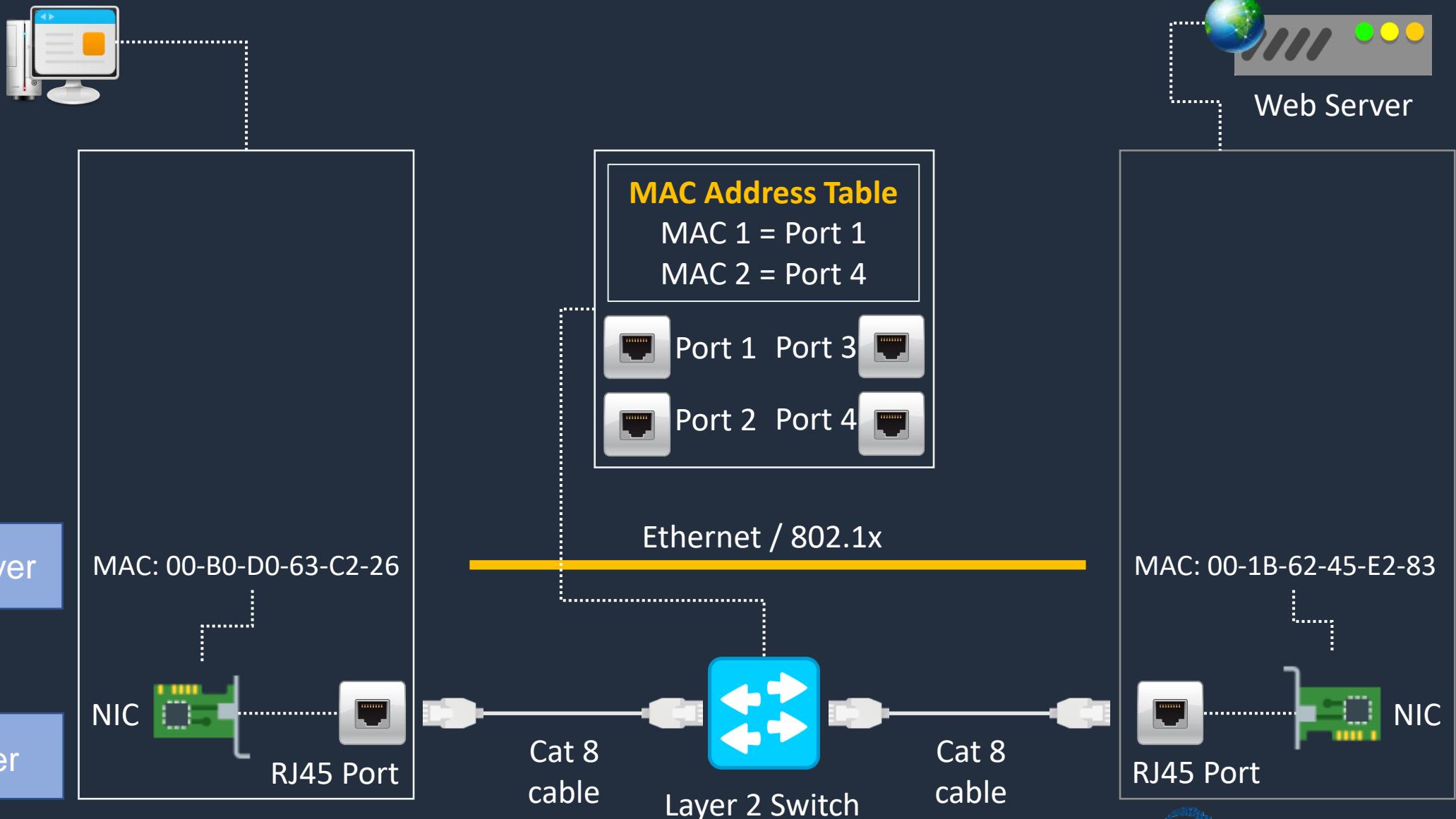
SECTION 5

Amazon Virtual Private Cloud (VPC)

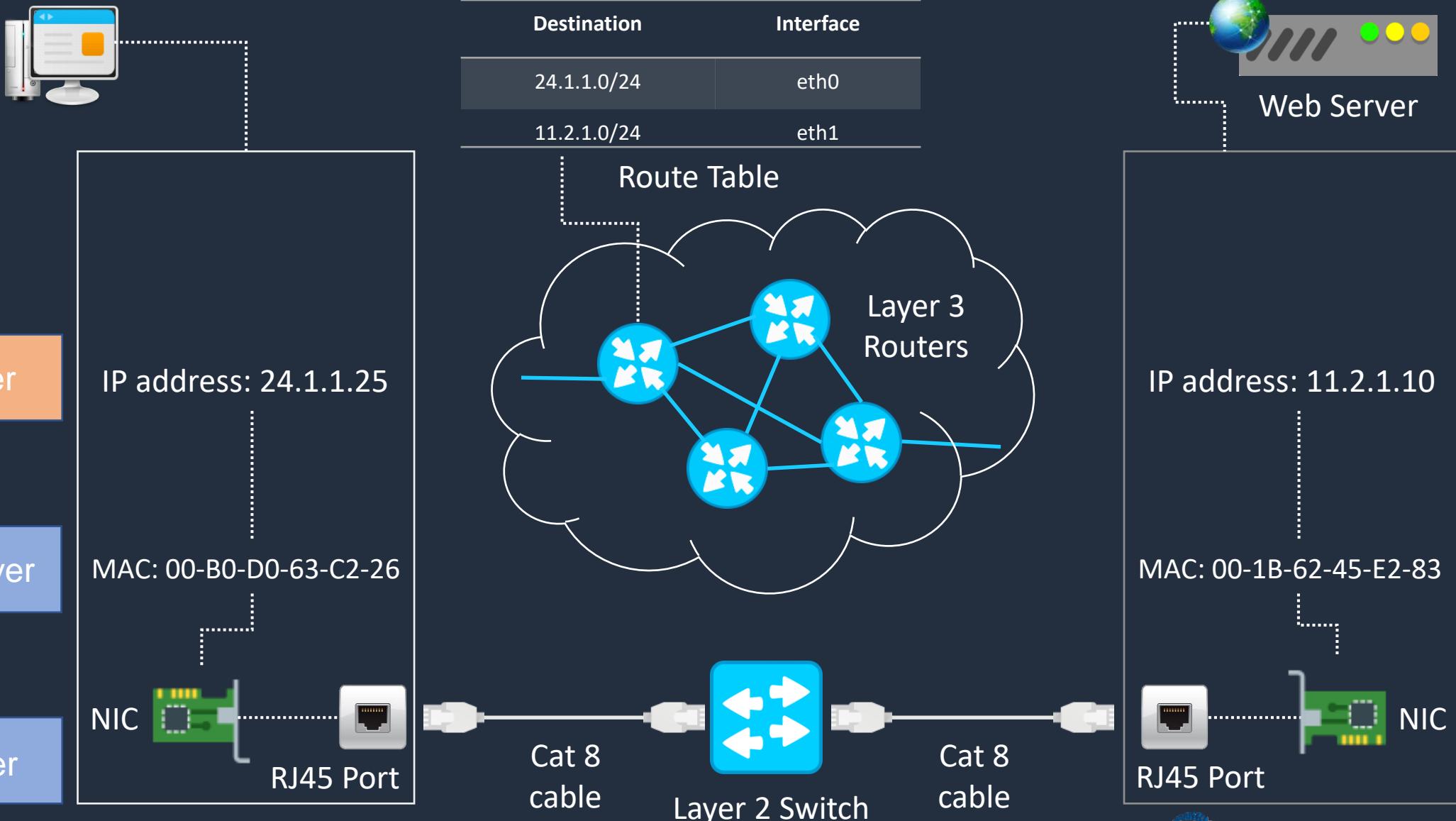
The OSI Model



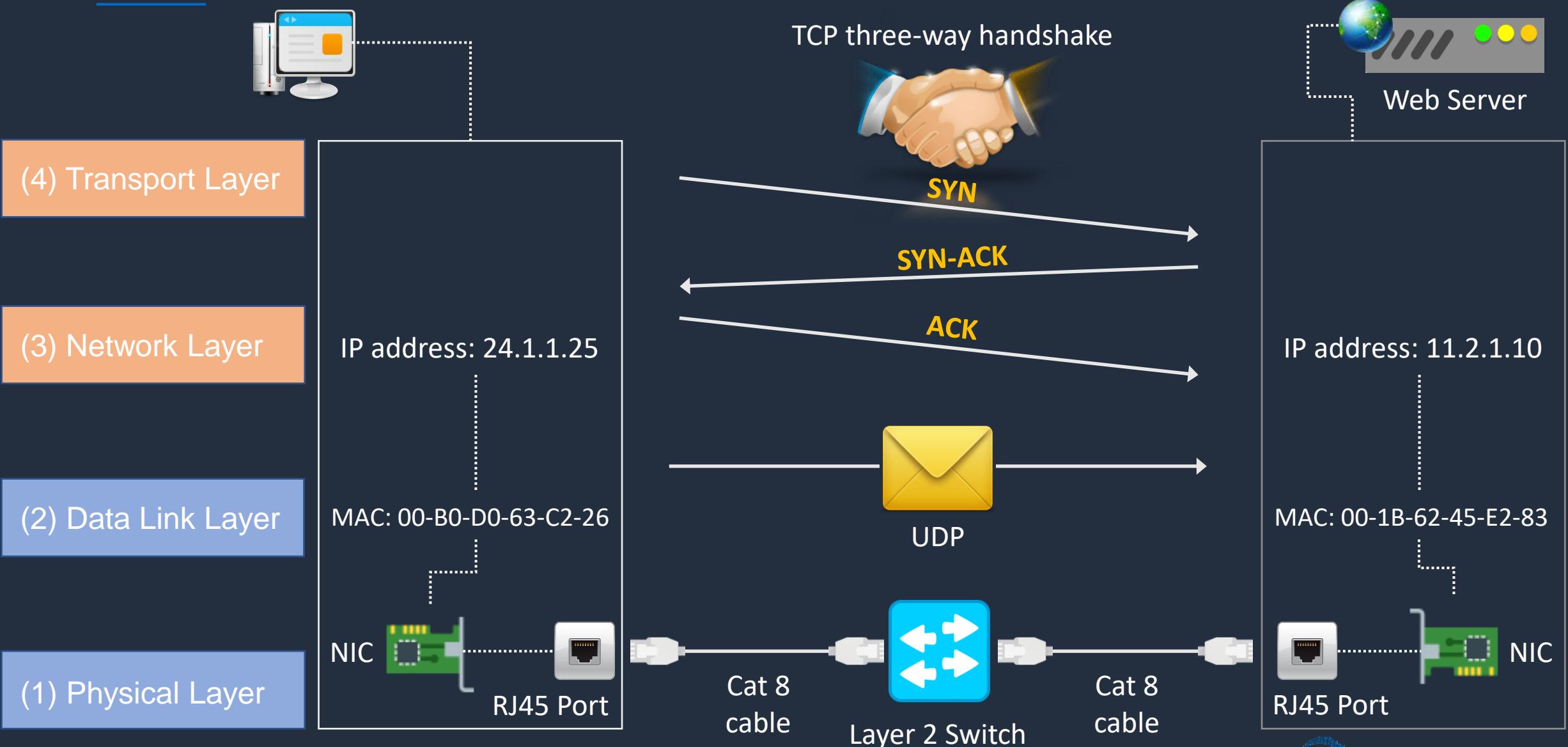
The Open Systems Interconnection (OSI) Model



The Open Systems Interconnection (OSI) Model



The Open Systems Interconnection (OSI) Model



The Open Systems Interconnection (OSI) Model

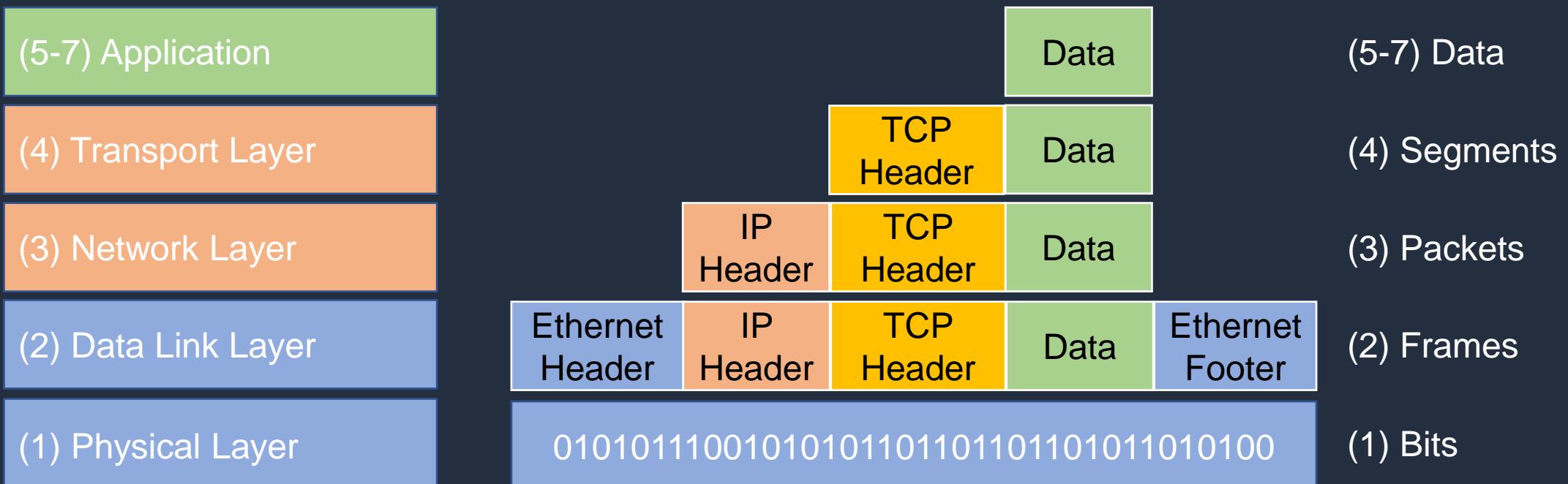
All People Seem To Need Data Processing

(7) Application Layer	Data	HTTP, FTP, SSH, DNS	 	Network services to end user applications
(6) Presentation Layer	Data	SSL, FTP, SSH, HTML	 	Data representation and encryption
(5) Session Layer	Data	TCP, RPC		Interhost communication
(4) Transport Layer	Segments	TCP, UDP, TLS		End-to-end connections and reliability
(3) Network Layer	Packets	IP, IPSec, ICMP	  	Path determination (e.g. routing)
(2) Data Link Layer	Frames	Ethernet, 802.11	 	Physical addressing (e.g. MAC and LLC)
(1) Physical Layer	Bits	Coax, Fiber, Wireless	 	Media, signal, binary transmission

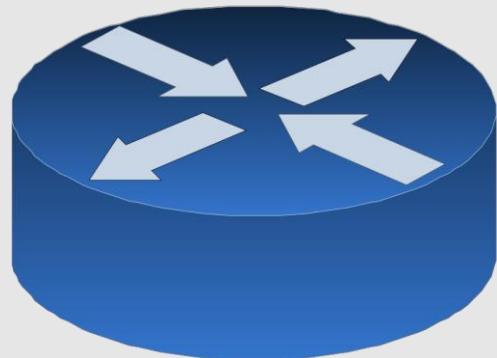


TCP Encapsulation

OSI Layer



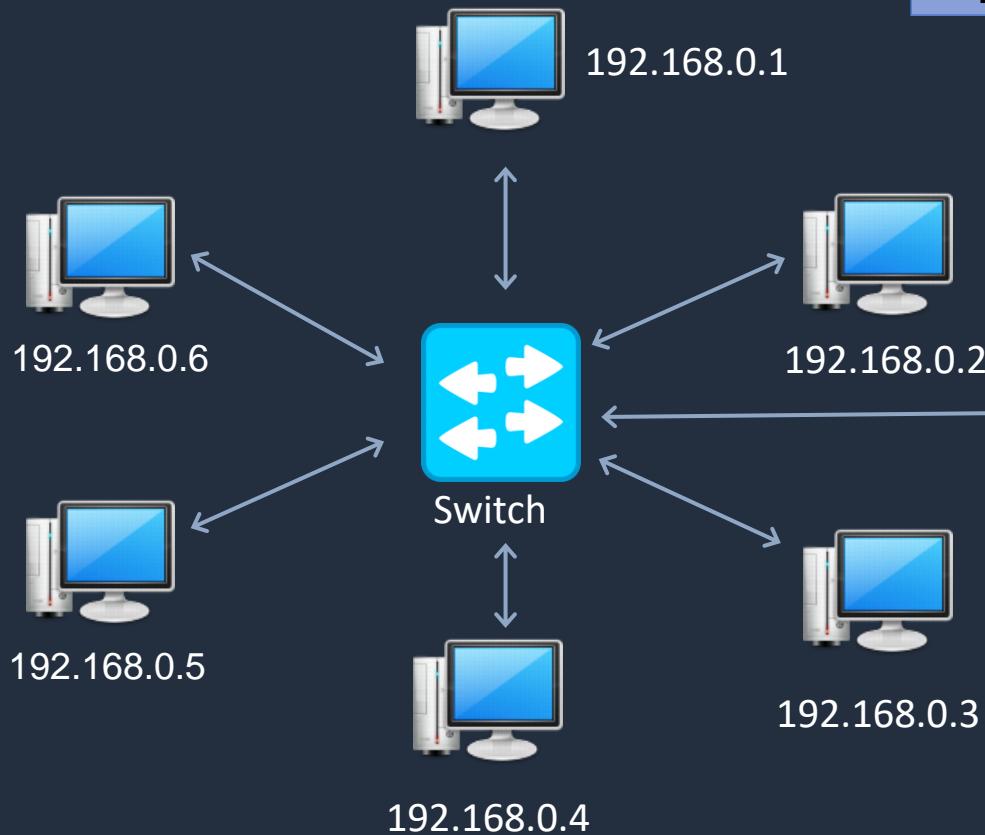
Routers, Switches, and Firewalls





Routers and Switches

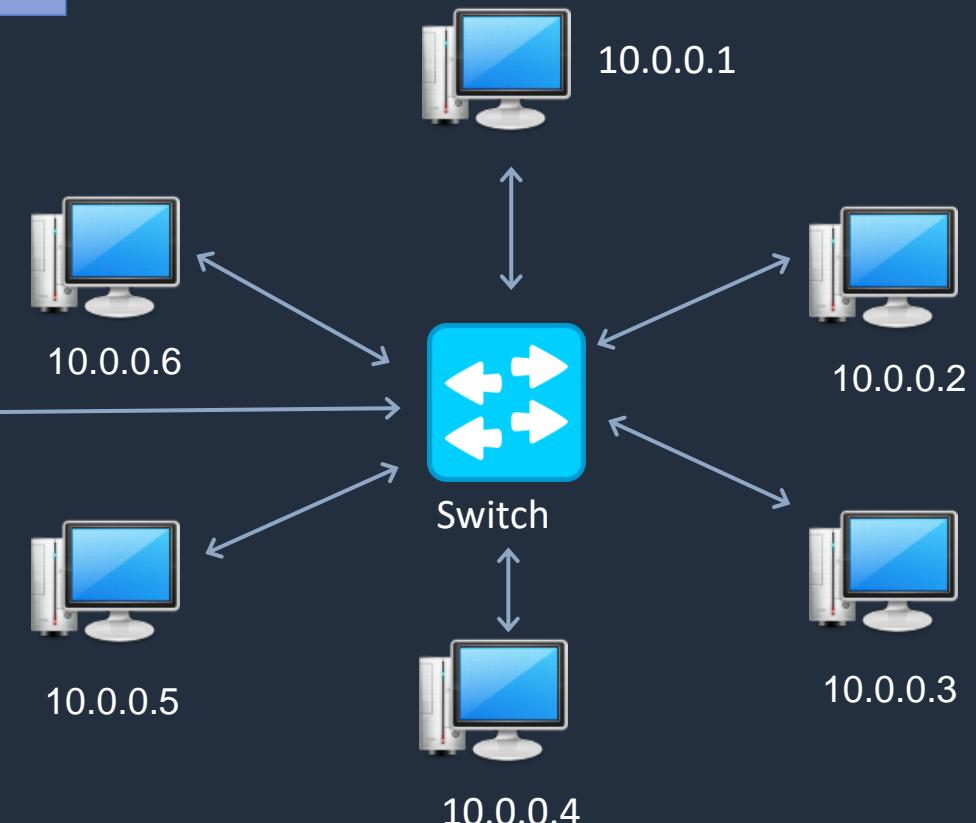
IP Subnet A: 192.168.0.0/24



Destination	Interface
192.168.0.0/24	<code>eth0</code>
10.0.0.0/24	<code>eth1</code>

Route Table

IP Subnet B: 10.0.0.0/24





Firewalls

POLICY	PROTOCOL	PORT	DESTINATION	SOURCE
ALLOW	HTTP	80	INTERNAL	ANY
ALLOW	HTTPS	443	INTERNAL	ANY
DENY	ANY	ANY	INTERNAL	ANY

Firewall Rules

IP Subnet A



Database Server



Application Server

IP Subnet B



Web Server



Firewall



Firewall



Firewall



The Internet



Database Server



Application Server



DigitalCloud
TRAINING

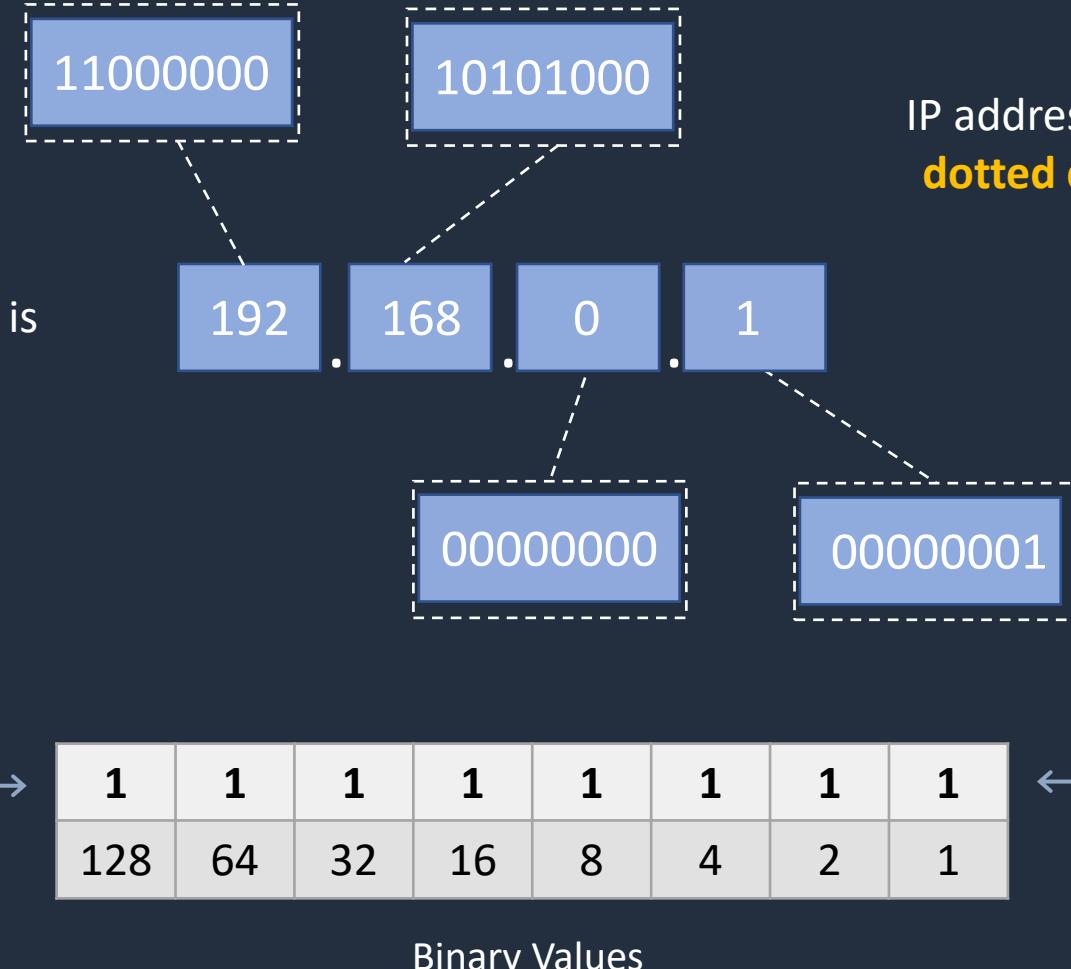
IP Addressing





Structure of an IPv4 Address

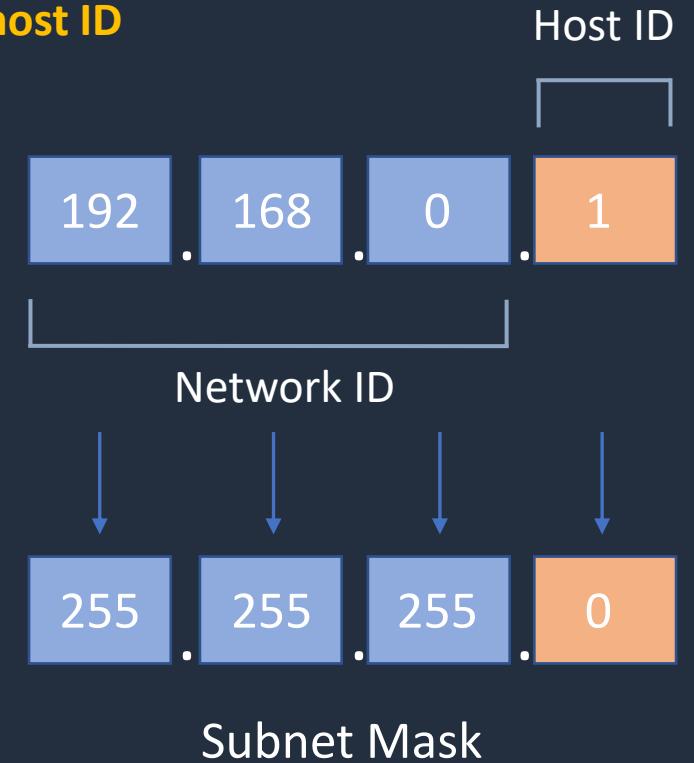
Each part of the address is a **binary octet**





Networks and Hosts

An IPv4 address has a **network** and **host ID**



The **subnet mask** is used to define the **network** and **host ID**



Networks and Hosts

Network

192	168	0	0
-----	-----	---	---

Subnet Mask

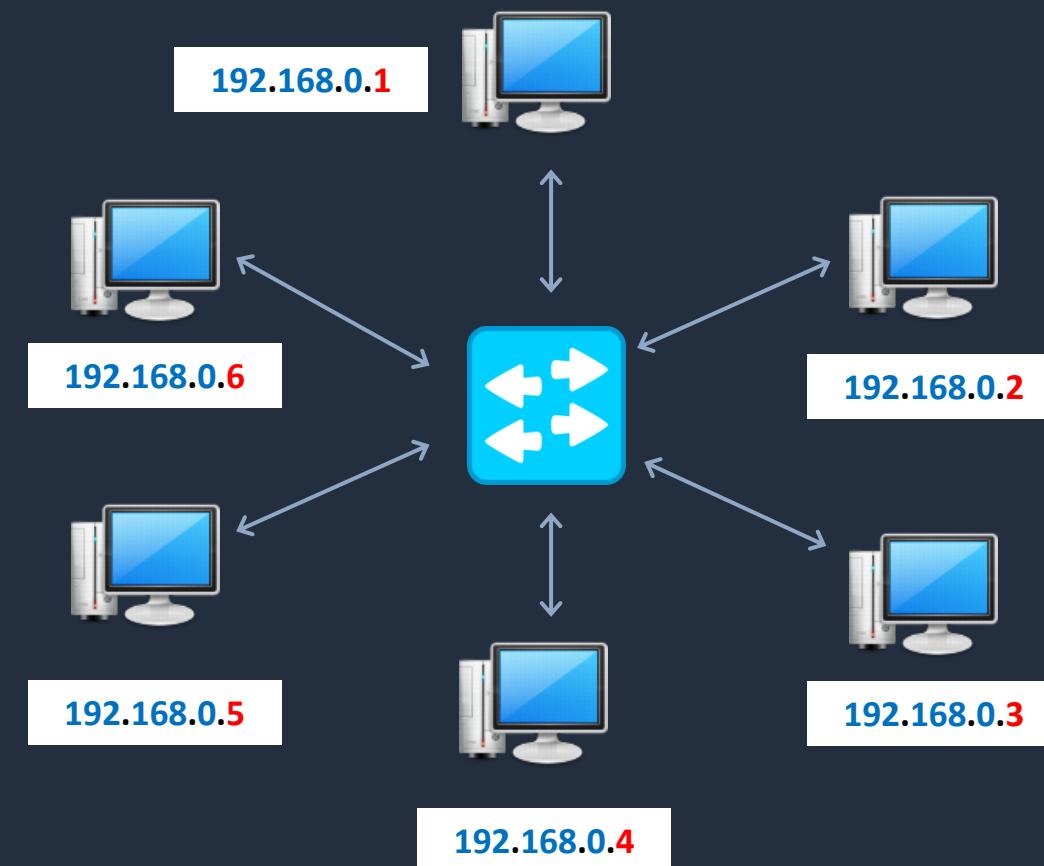
255	255	255	0
-----	-----	-----	---

24 bits

192.168.0.0/24

A **network** and **subnet mask** can also be written in this format

All computers share the same **network ID** and have a unique **host ID**





Private IP Address Ranges



These addresses are reserved
for **private use** according to
IETF RFC-1918

Amazon Virtual Private Cloud (VPC)

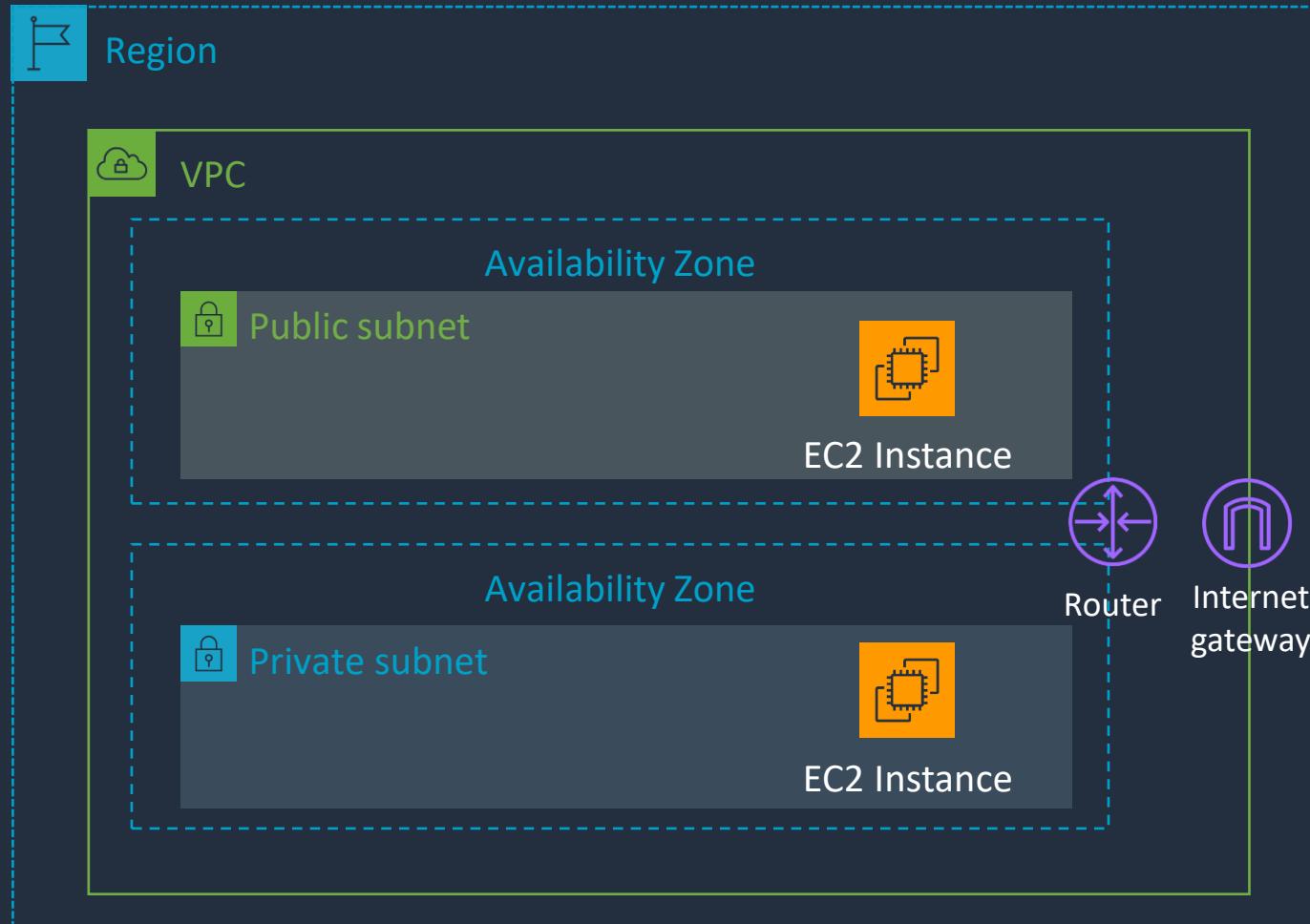




Amazon VPC

A **VPC** is a logically isolated portion of the AWS cloud within a region

Subnets are created within **AZs**



You can launch **EC2 instances** into your VPC subnets

Main Route Table

Destination	Target
10.0.0.0/16	Local
0.0.0.0/0	igw-id

The **route table** is used to configure the VPC router

An **Internet Gateway** is used to connect to the Internet



Amazon VPC

Each **VPC** has a different block of IP addresses

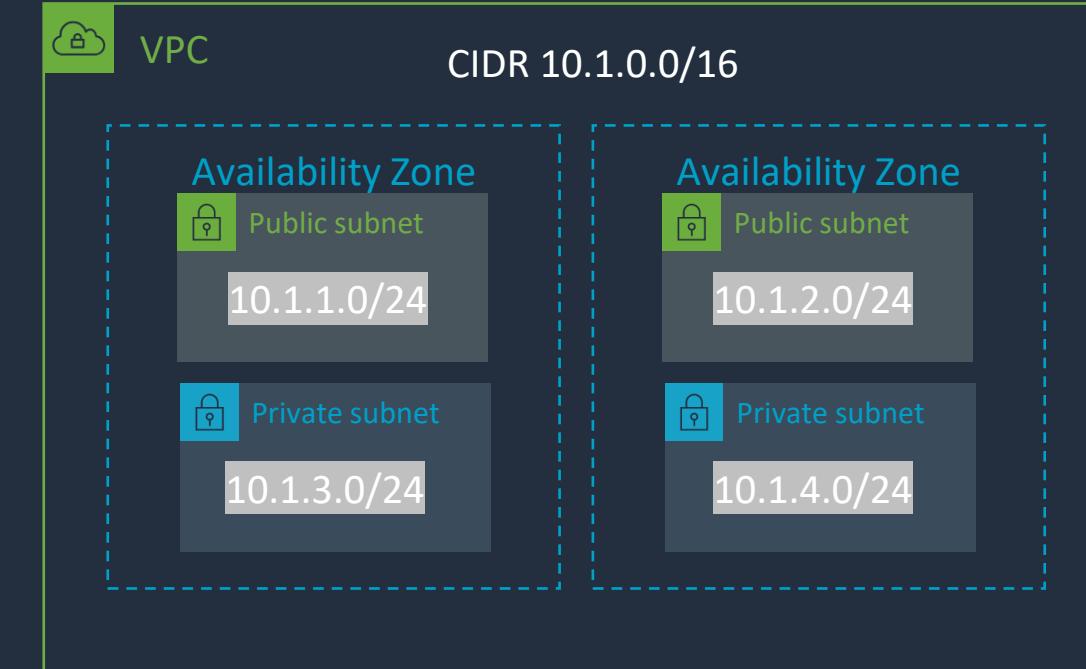
CIDR stands for Classless Interdomain Routing



Region



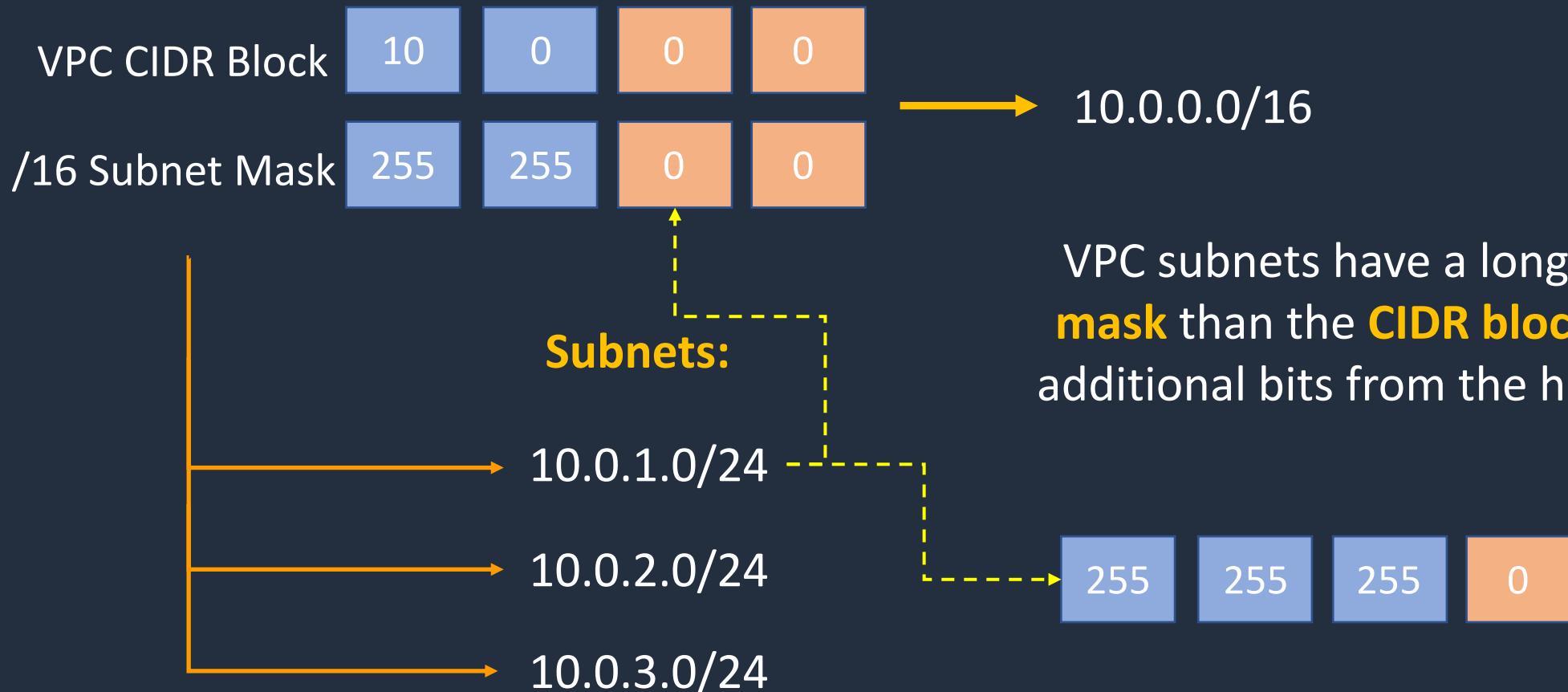
Each subnet has a block of **IP addresses** from the CIDR block



You can create **multiple VPCs** within each region



VPC CIDR Blocks and Subnets

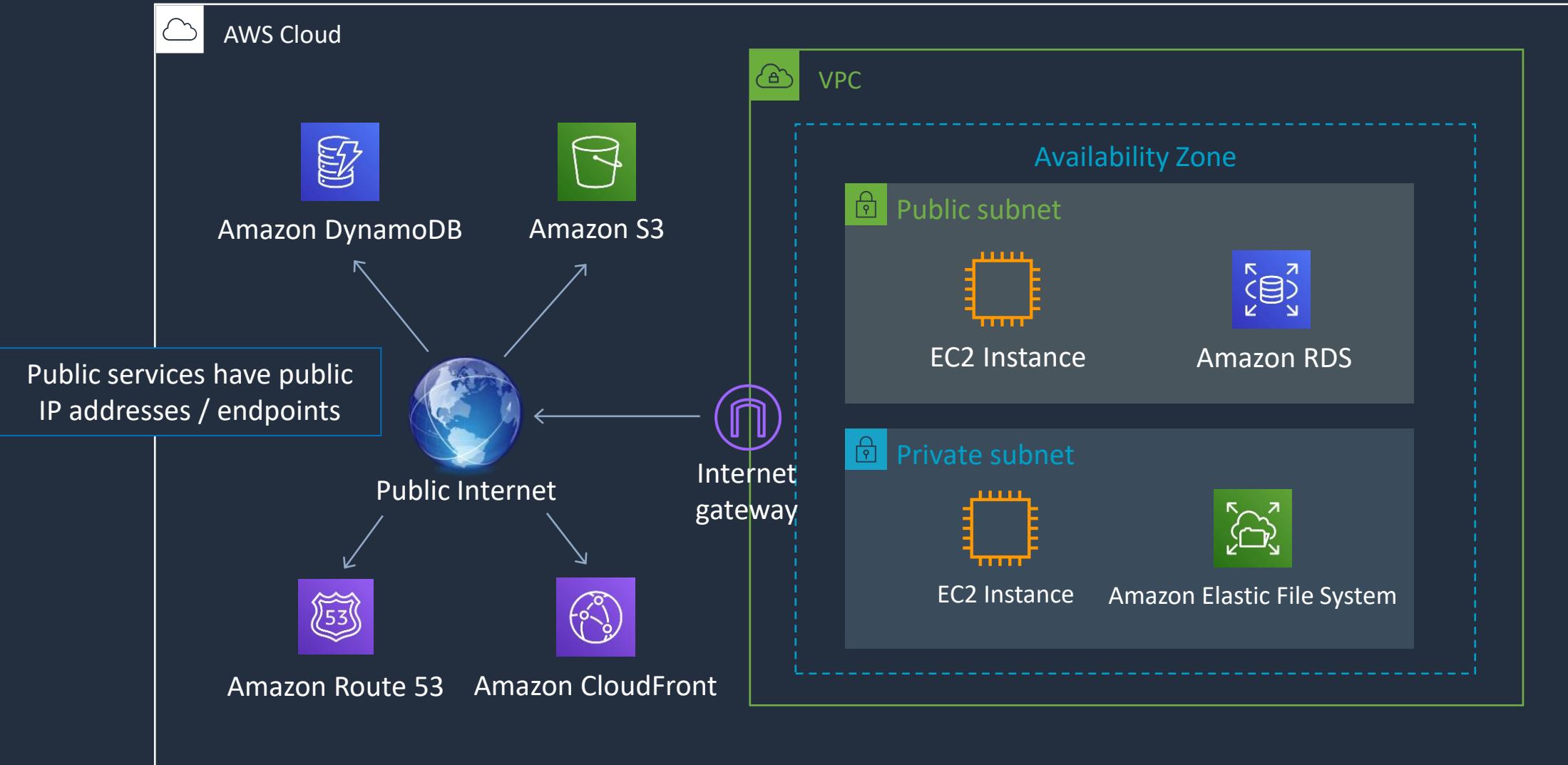


VPC subnets have a longer **subnet mask** than the **CIDR block** by using additional bits from the host portion



AWS Public and Private Services

Private services can have public IP addresses but exist within the VPC

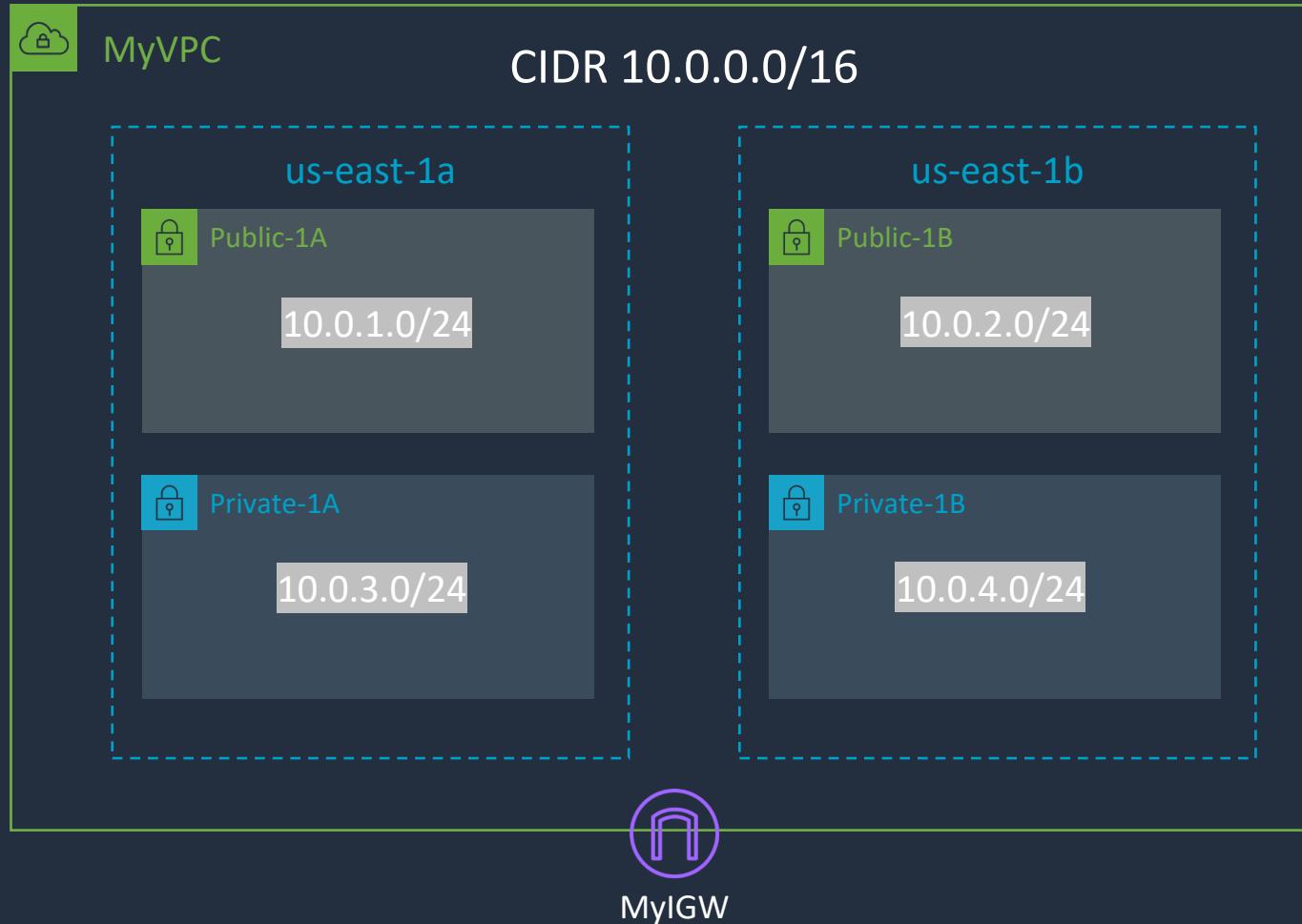


Create a Custom VPC





Custom VPC



Main Route Table

Destination	Target
10.0.0.0/16	Local
0.0.0.0/0	igw-id

Private-RT Route Table

Destination	Target
10.0.0.0/16	Local

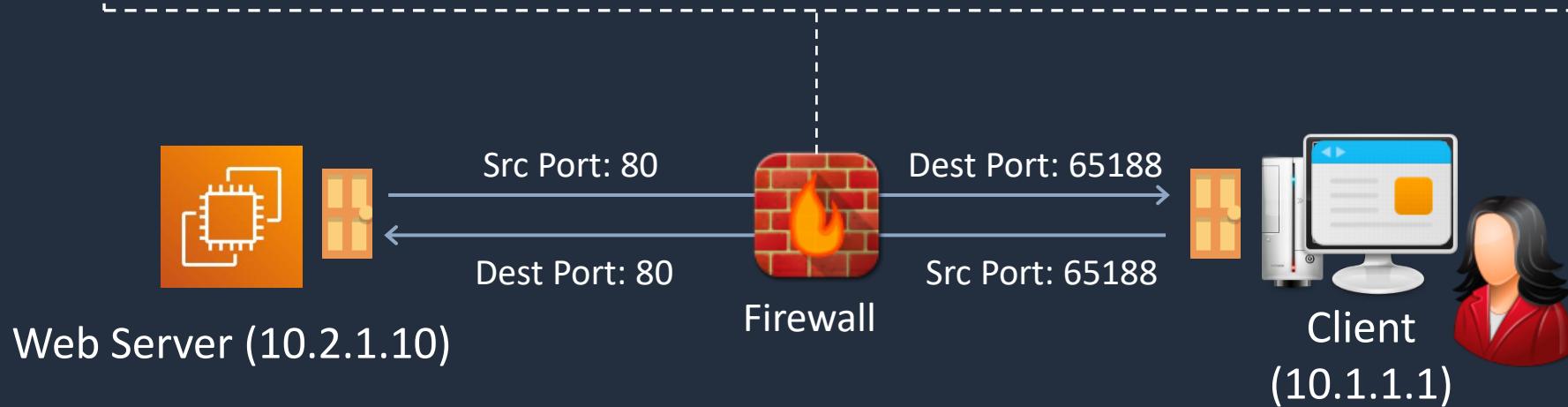
Security Groups and Network ACLs





Stateful vs Stateless Firewalls

PROTOCOL	SOURCE IP	DESTINATION IP	SOURCE PORT	DESTINATION PORT
HTTP	10.1.1.1	10.2.1.10	65188	80
HTTP	10.2.1.10	10.1.1.1	80	65188

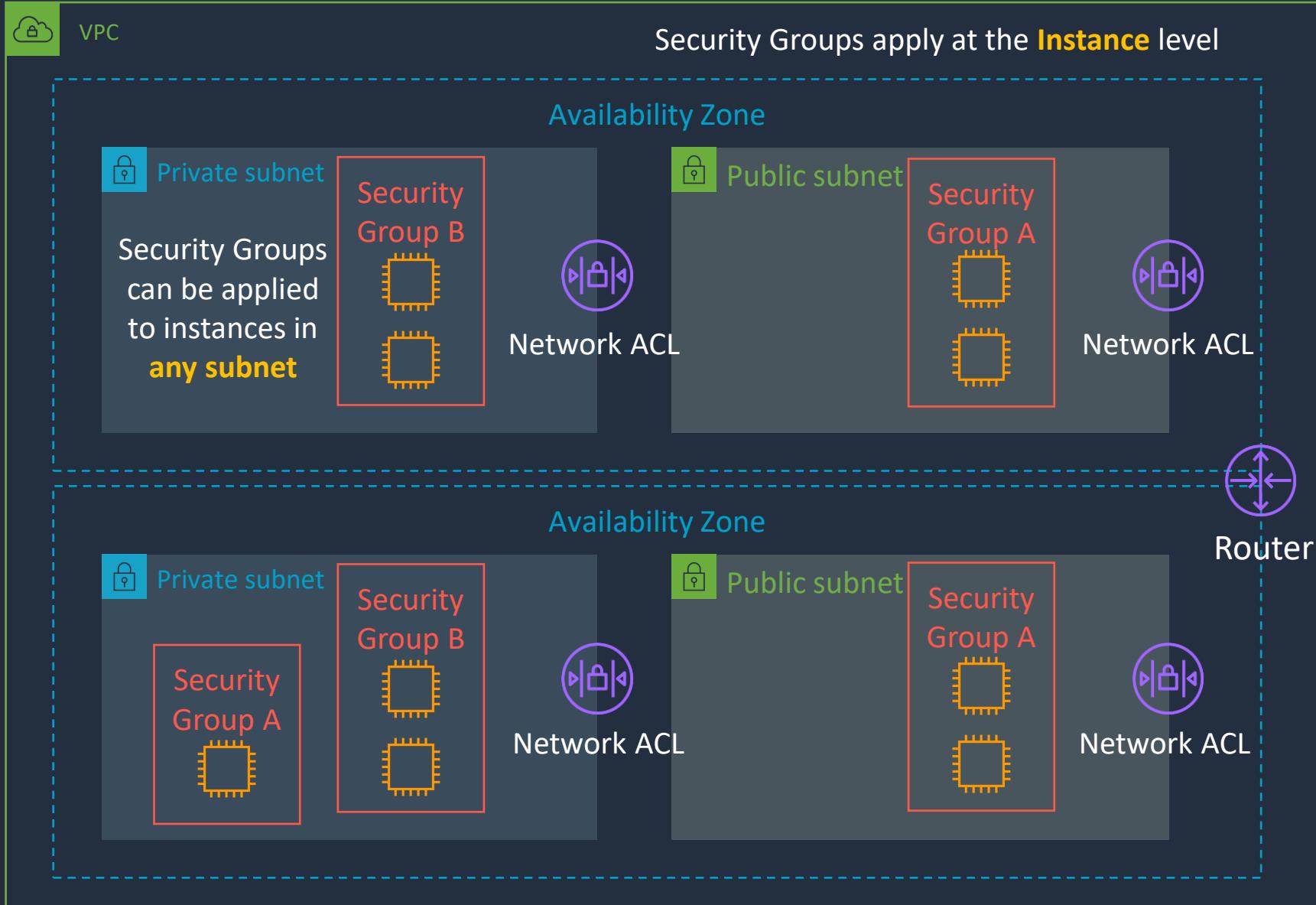


A **stateful** firewall
allows the return
traffic automatically

A **stateless** firewall
checks for an allow
rule for **both**
connections



Security Groups and Network ACLs





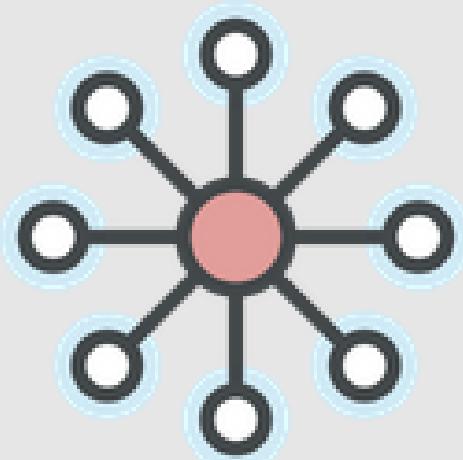
Security Groups & Network Access Control Lists (NACLs)

Security Group	Network ACL
Operates at the instance (interface) level	Operates at the subnet level
Supports allow rules only	Supports allow and deny rules
Stateful	Stateless
Evaluates all rules	Processes rules in order
Applies to an instance only if associated with a group	Automatically applies to all instances in the subnets its associated with

Using Security Groups and NACLs



Cloud Computing Deployment Models





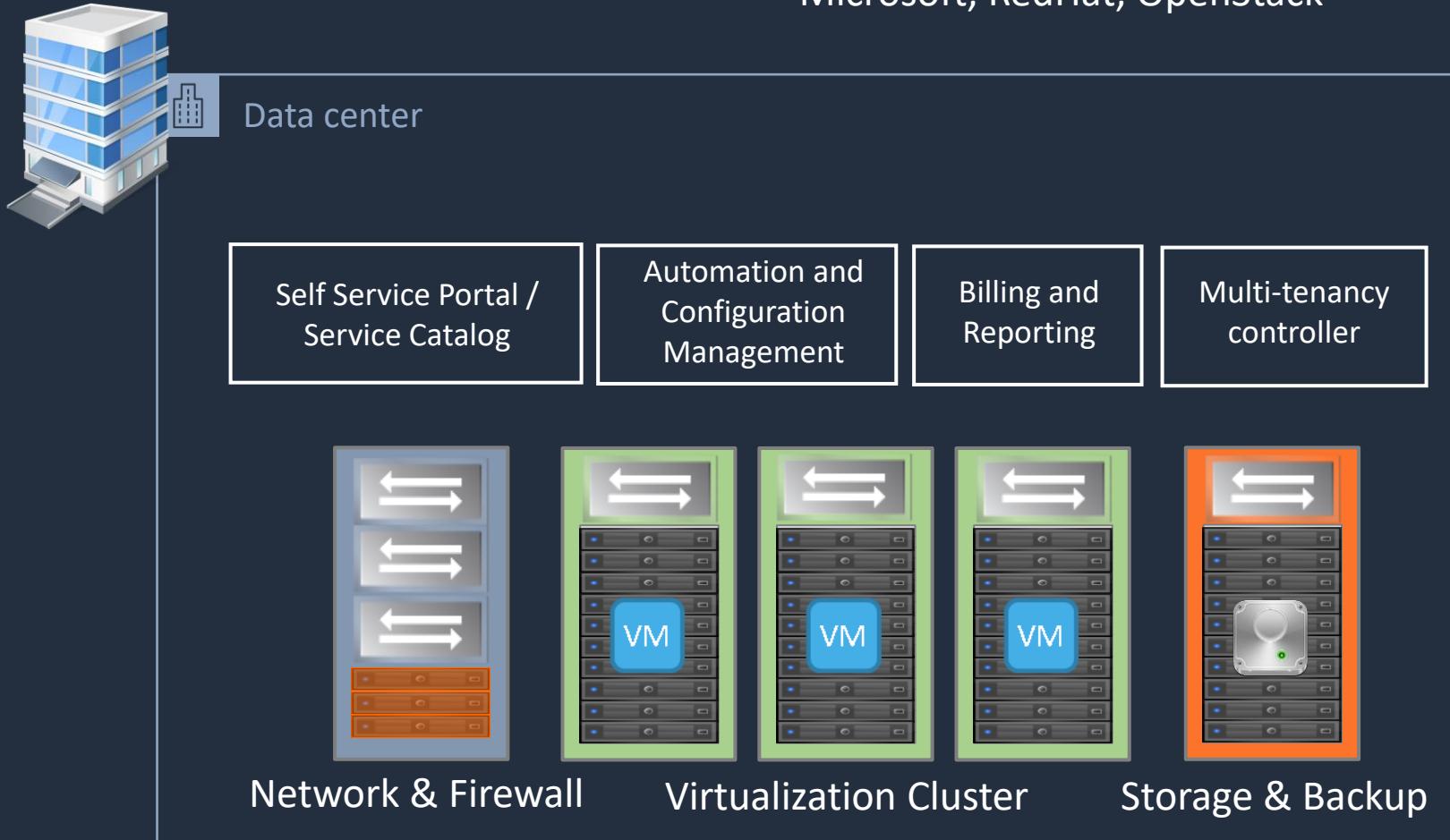
Cloud Computing Deployment Models

Name	Description	Examples
Private Cloud	An enterprise deploys their own infrastructure and applications into their own data center	VMware, Microsoft, RedHat, OpenStack
Public Cloud	The IT services that you consume are hosted and delivered from a third-party and accessed over the Internet	AWS, Microsoft Azure, Google Cloud Platform
Hybrid Cloud	A combination of on-premises, private cloud, and public cloud services are consumed	
Multicloud	Usage of two or more public clouds at a time, and possibly multiple private clouds	



Private Cloud

Examples are VMware,
Microsoft, RedHat, OpenStack



Cloud management
software layer

Benefits:

- Complete control of the entire stack
- Security – in a few cases, organizations may need to keep all or some of their applications and data in house

You **build** and **manage** the
cloud deployment



Cloud Deployment Models - Public Cloud

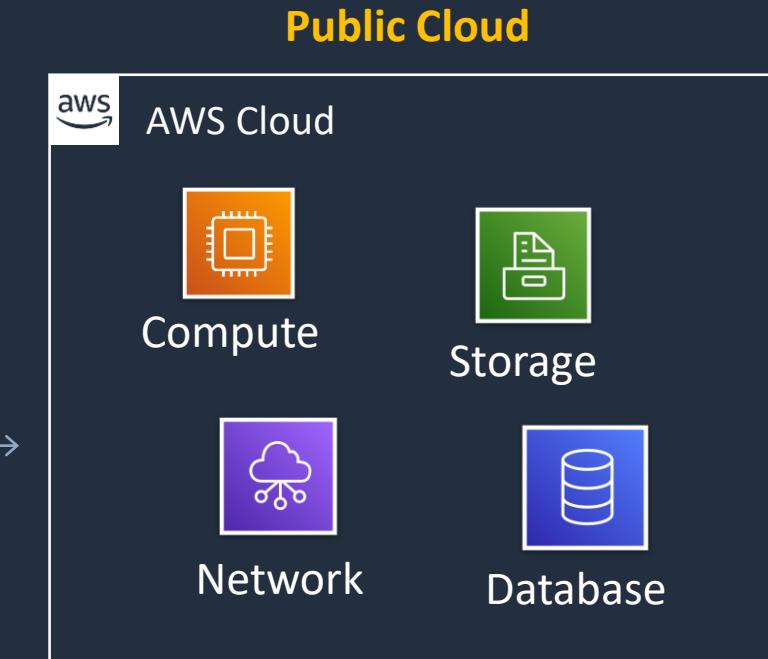
Examples are AWS, Microsoft Azure, Google Cloud Platform

Benefits:

- Variable expense, instead of capital expense
- Economies of scale
- Massive elasticity



Connected using either the
Internet or a **private link**

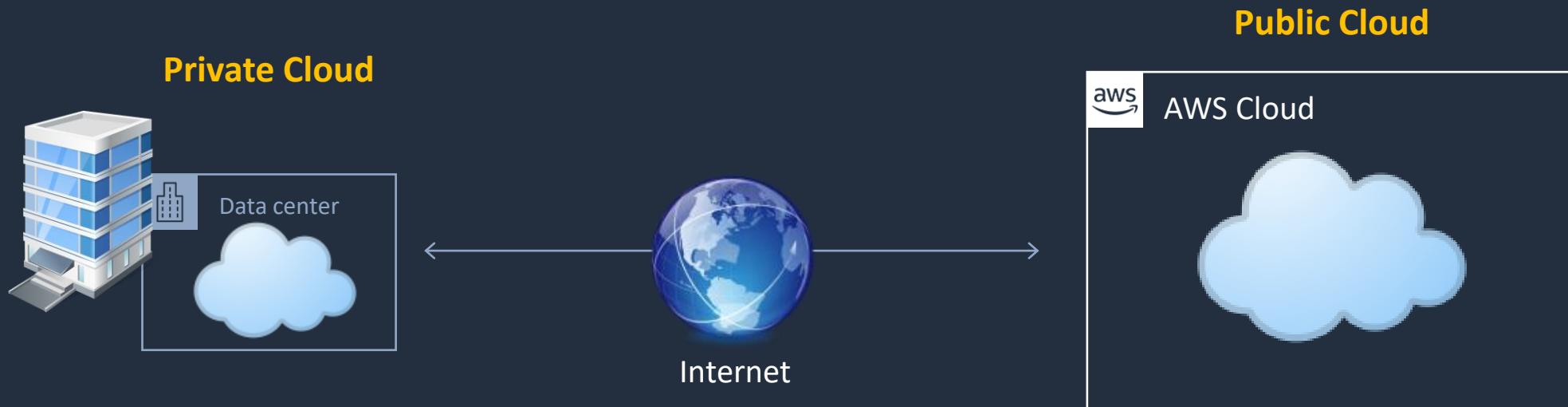




Cloud Deployment Models - Hybrid Cloud

Benefits:

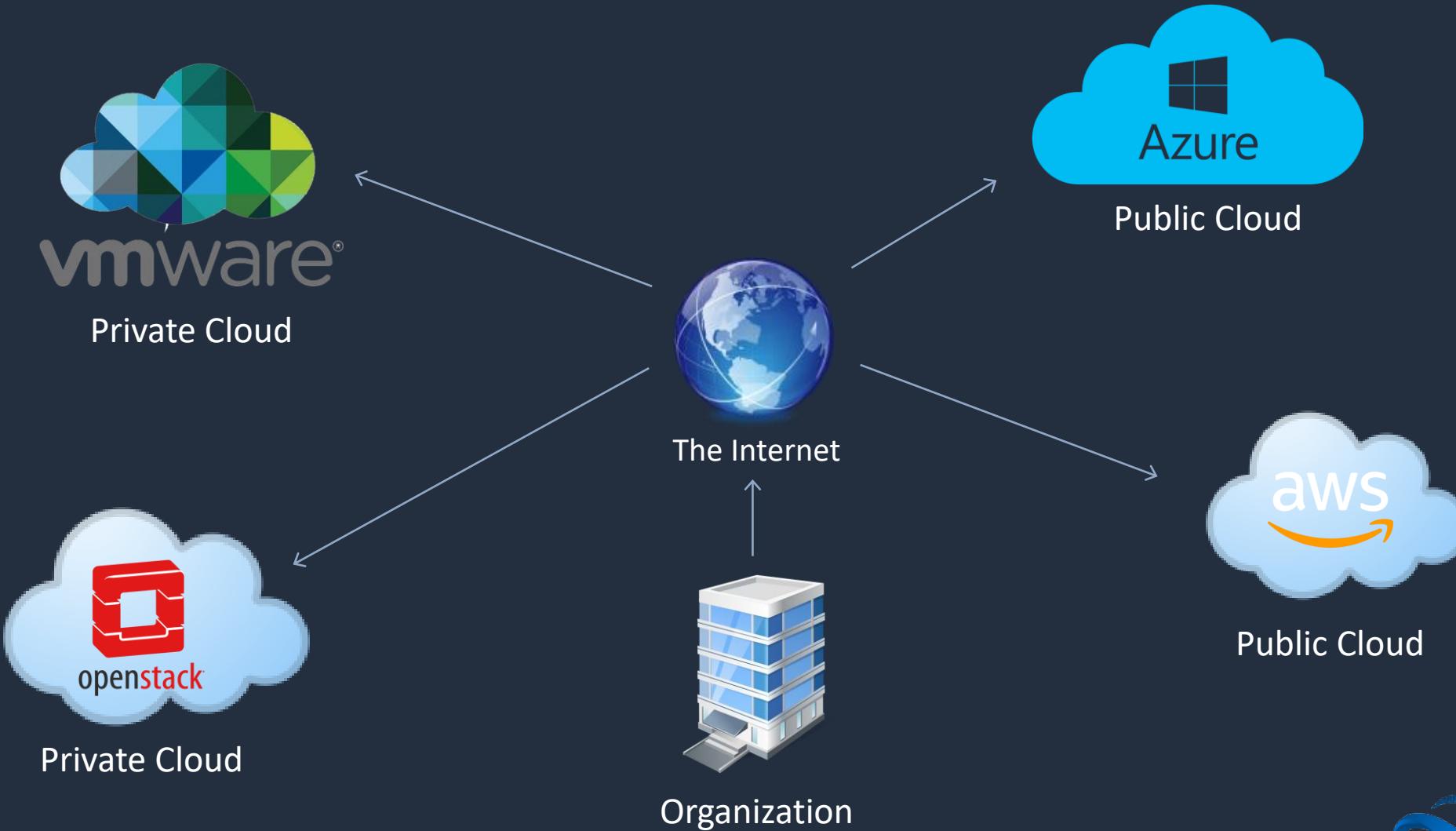
- Allows companies to keep the critical applications and sensitive data in a traditional data center environment or private cloud
- Take advantage of public cloud resources like SaaS, for the latest applications, and IaaS, for elastic virtual resources
- Facilitates portability of data, apps and services and more choices for deployment models



Connected using either the
Internet or a **private** link



Cloud Deployment Models - Multicloud



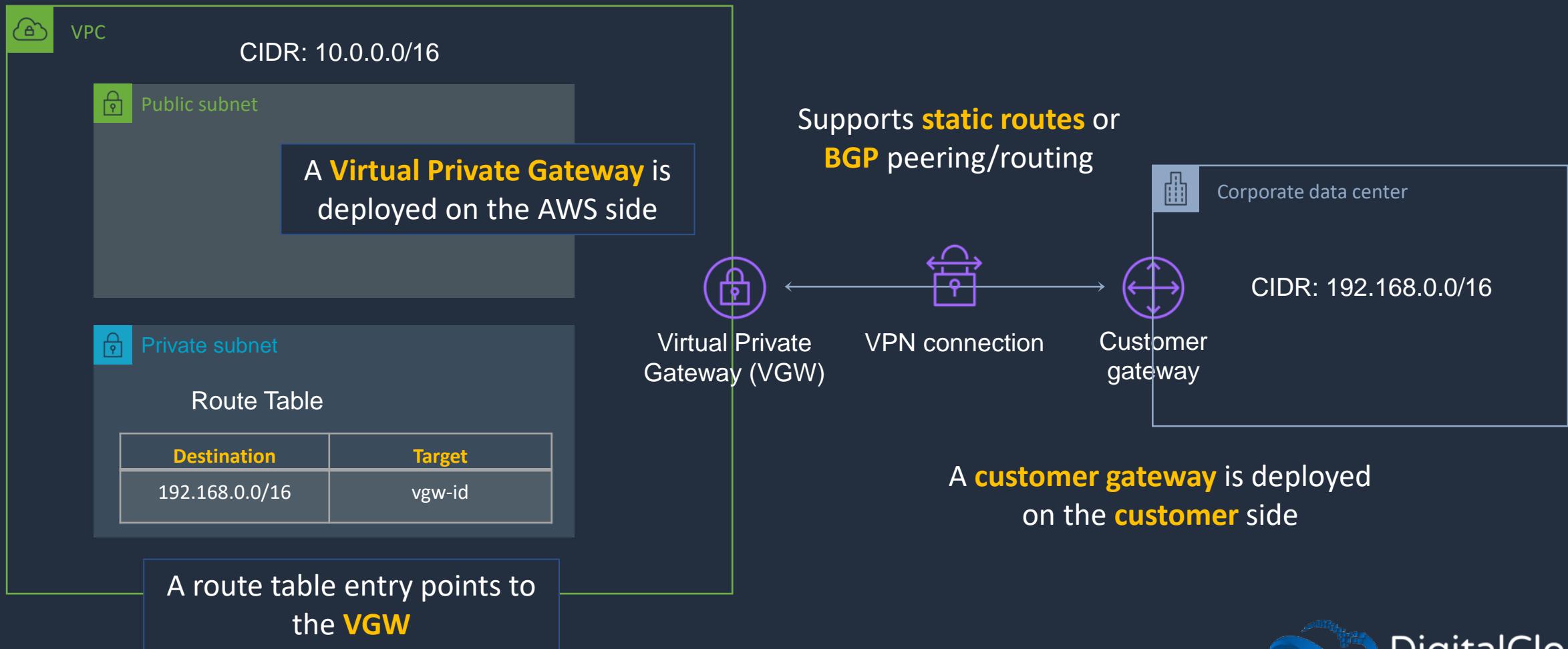
AWS VPN and AWS Direct Connect





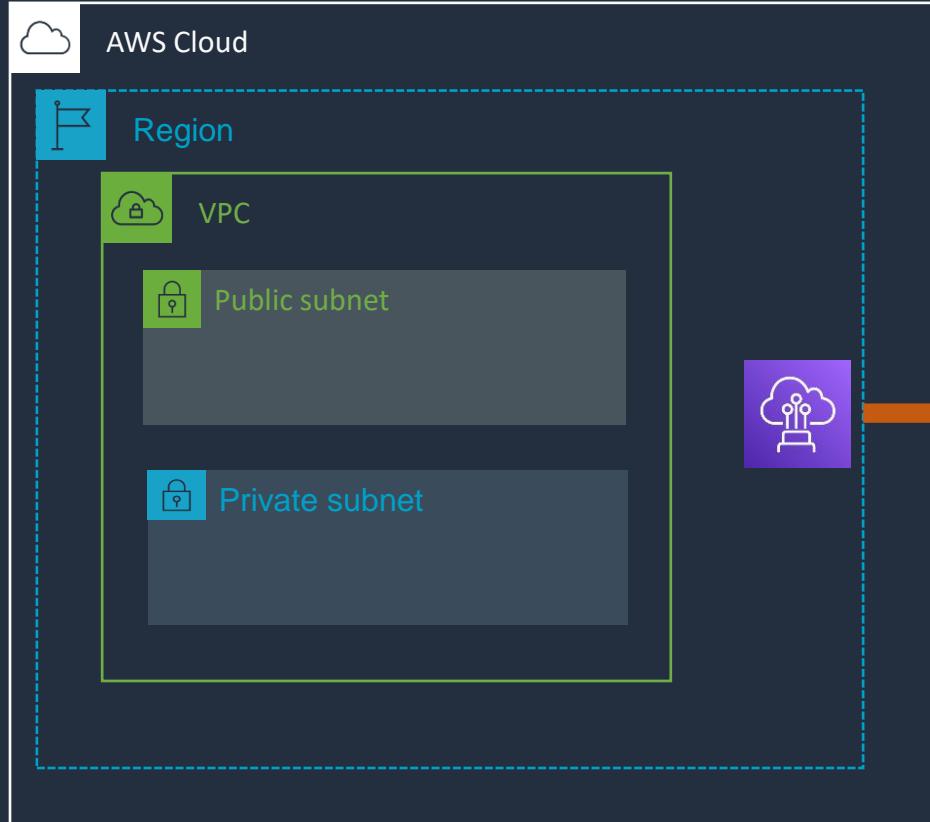
AWS Site-to-Site VPN

AWS VPN is a managed IPSec VPN

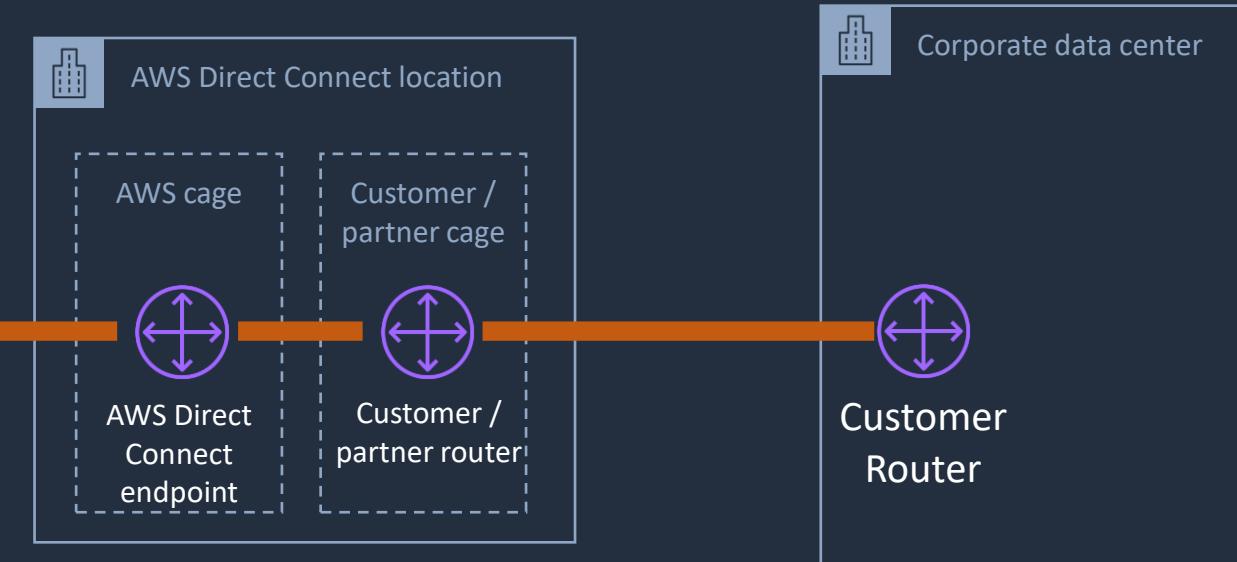


AWS Direct Connect (DX)

DX is a physical fibre connection to AWS running at **1Gbps** to **100Gbps**



A **cross-connect** between the AWS DX router and the customer/partner DX router



A **DX port** (1000-Base-LX or 10GBASE-LR) must be allocated in a **DX location**

The **customer router** is connected to the DX router in the DX location



AWS Direct Connect Benefits

- **Enhanced security** – uses private connectivity between AWS and your data center / office
- **Consistent network performance** – increased speed/latency & bandwidth/throughput
- **Lower cost** – can reduce costs for organizations that transfer large volumes of data

SECTION 6

AWS Storage Services

Block vs File vs Object Storage





Block Storage



Hard Disk Drive (HDD)

- Also known as magnetic drives
- Older technology
- Much slower than SSD
- Much cheaper than SSD



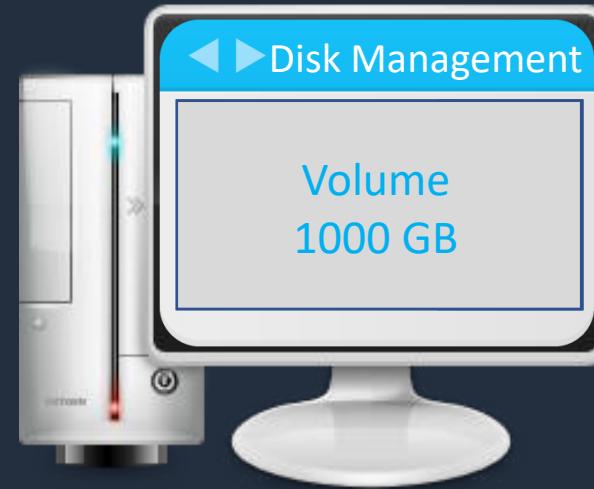
Solid State Drive (SSD)

- Uses flash memory
- Newer technology
- MUCH faster than HDD
- More expensive than HDD



Block Storage

Hard drives are
block-based
storage systems



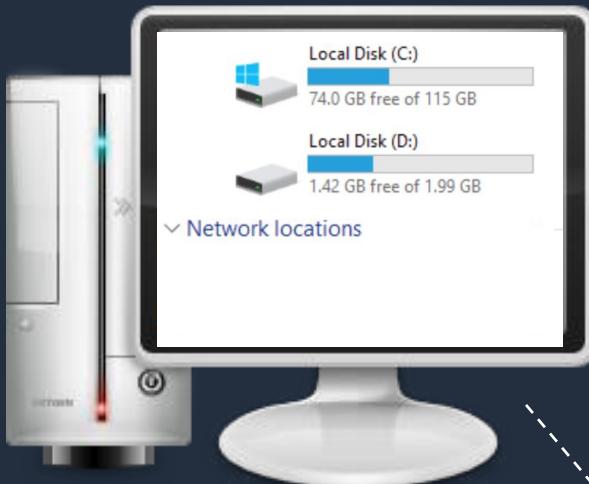
Hard drives are block-based storage systems

The Operating System
(OS) sees a volume. A
volume can be
partitioned and
formatted

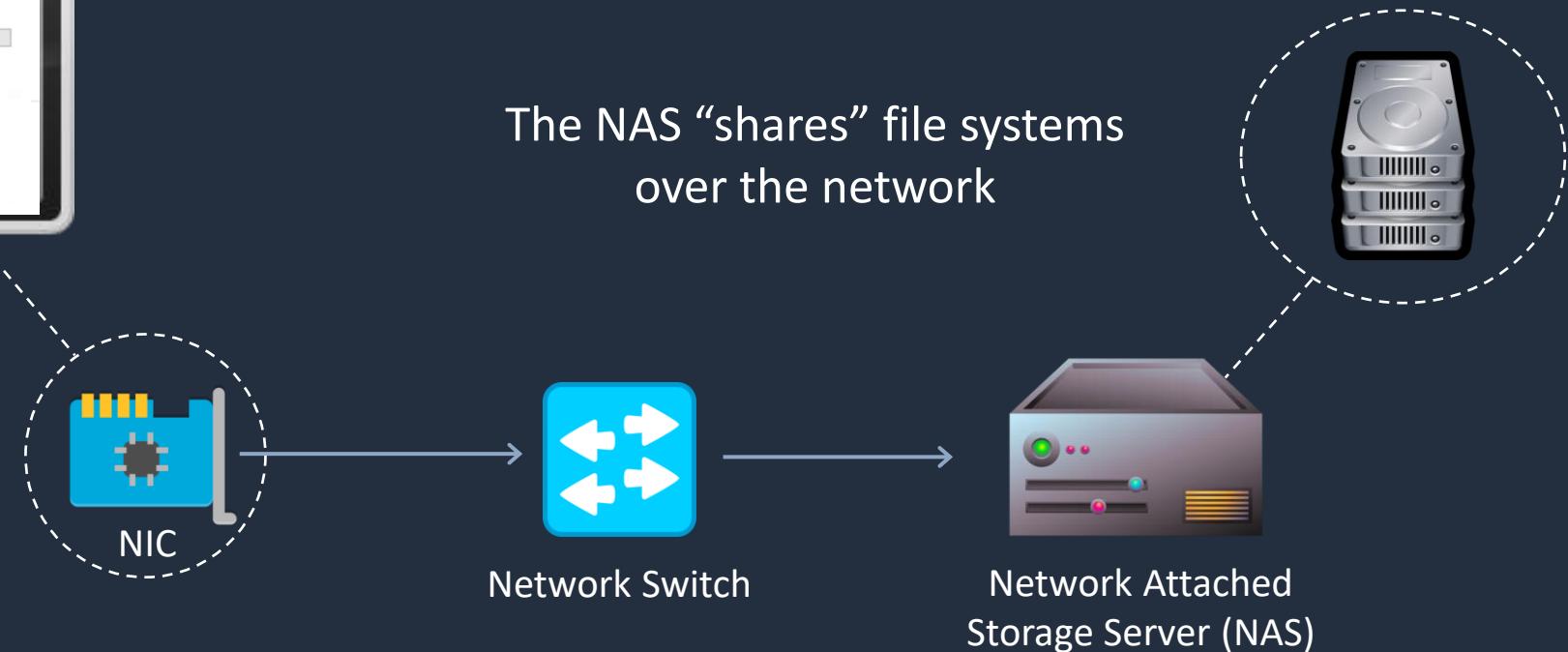


File Storage

The Operating System (OS) sees a **file system** that is mapped to a local drive letter



The NAS “shares” file systems over the network



Object Storage Systems

User uploads **objects** using a web browser



The **HTTP protocol** is used with a **REST API** (e.g. GET, PUT, POST, SELECT, DELETE)

There is no hierarchy of objects in the container



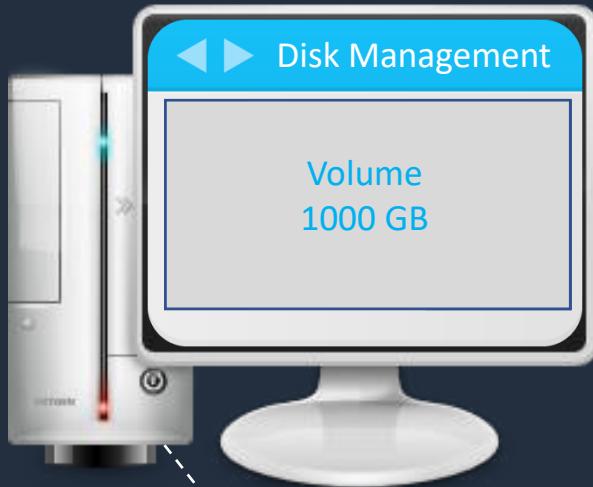
Object Storage Container

Objects can be files, videos, images etc.

Block vs File vs Object Storage

The OS sees **volumes** that can be partitioned and formatted

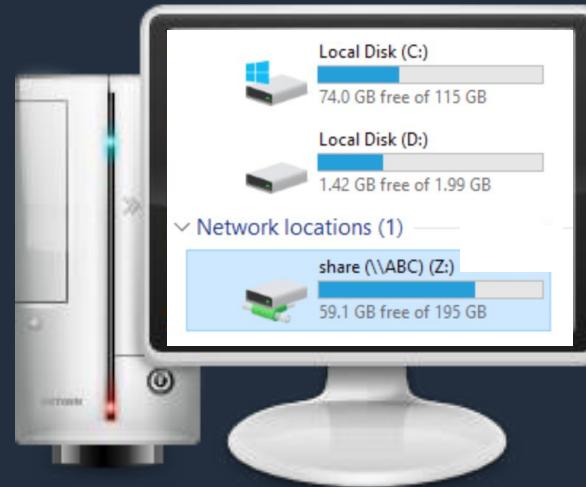
Block Storage



The OS reads/writes at the **block level**. Disks can be internal, or network attached

A filesystem can be shared by many users

File Storage



A filesystem is “**mounted**” to the OS using a network share

Massively scalable and low cost

Object Storage

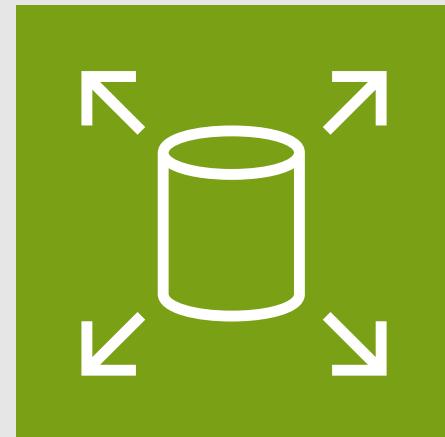


Object Storage Container

There is **no hierarchy** of objects in the container

Cannot be mounted but can be accessed programmatically using the **REST API**

Amazon EBS and Instance Stores





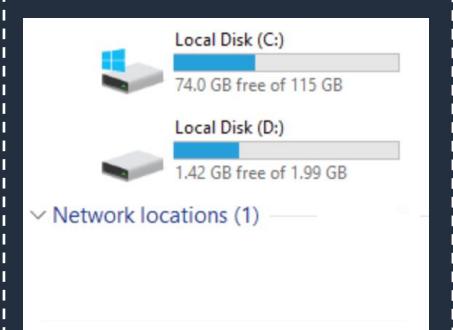
Amazon Elastic Block Store (EBS)

EBS volumes exist within an **Availability Zone**

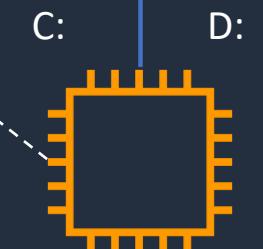
Availability Zone



The volume is automatically replicated **within the AZ**



The volume is **attached** over a network

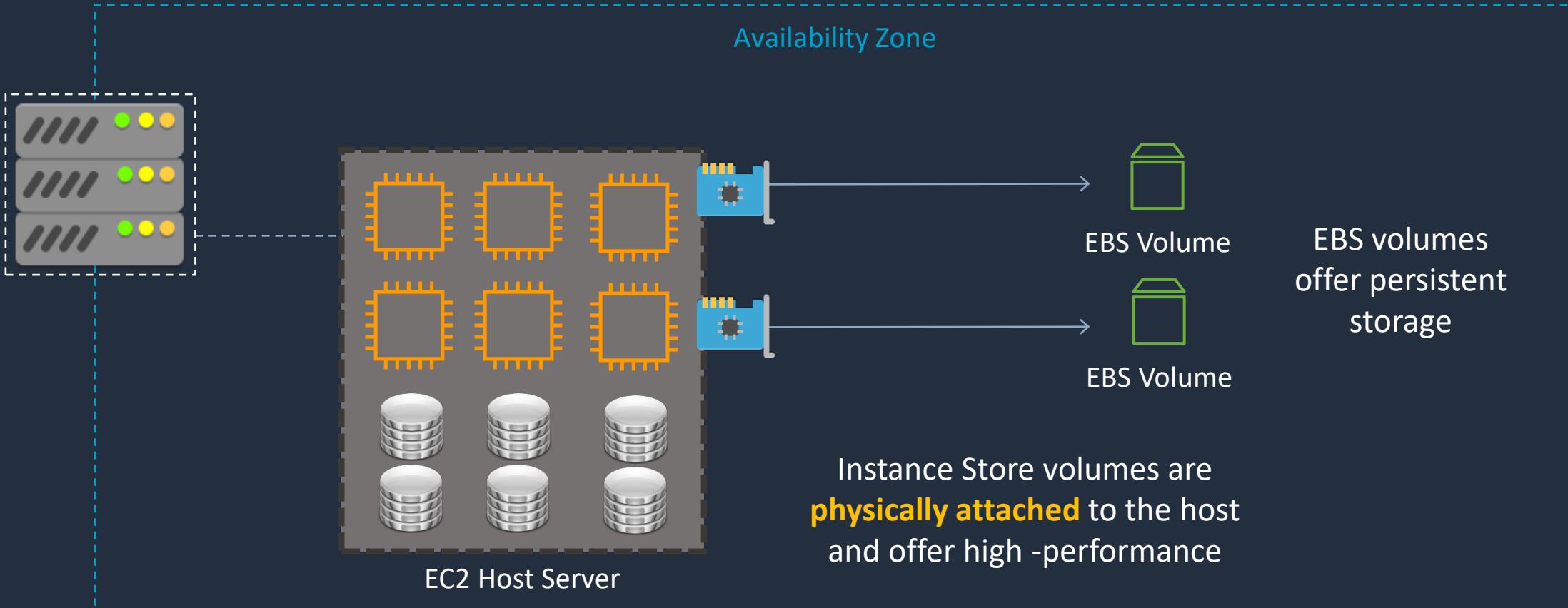


EC2 Instance



Amazon EBS vs Instance Store

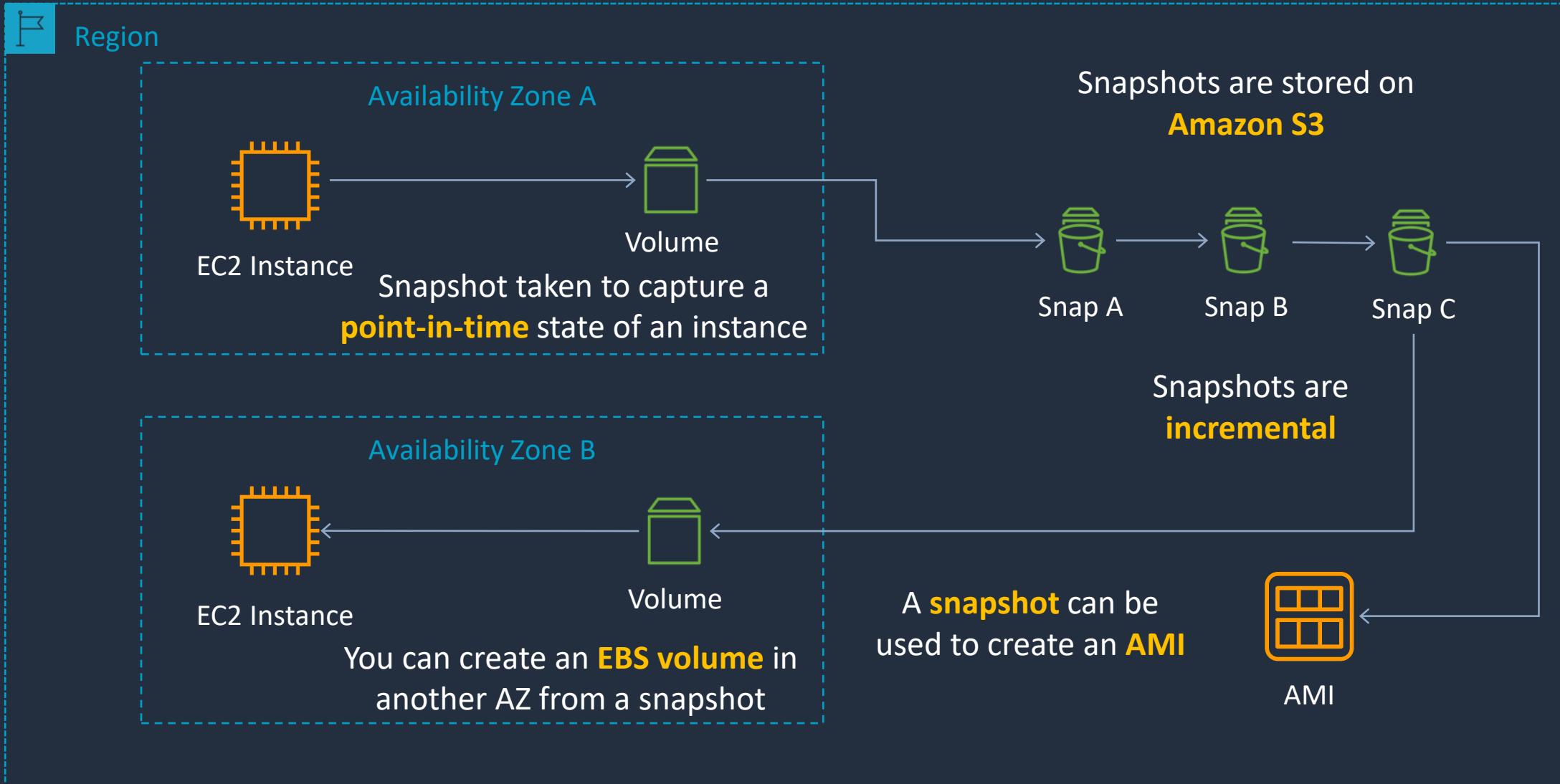
EBS volumes are **attached** over the **network**



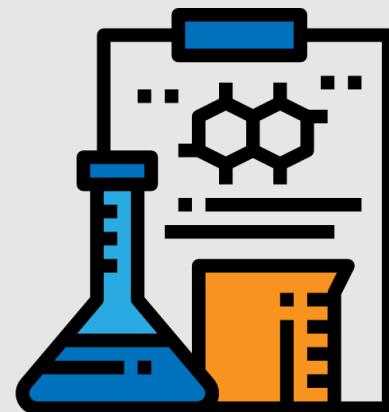
Instance Store volumes are **ephemeral** (non-persistent)



Amazon EBS Snapshots



Create and Attach an EBS Volume



EBS Snapshots and AMIs



Amazon Elastic File System (EFS)

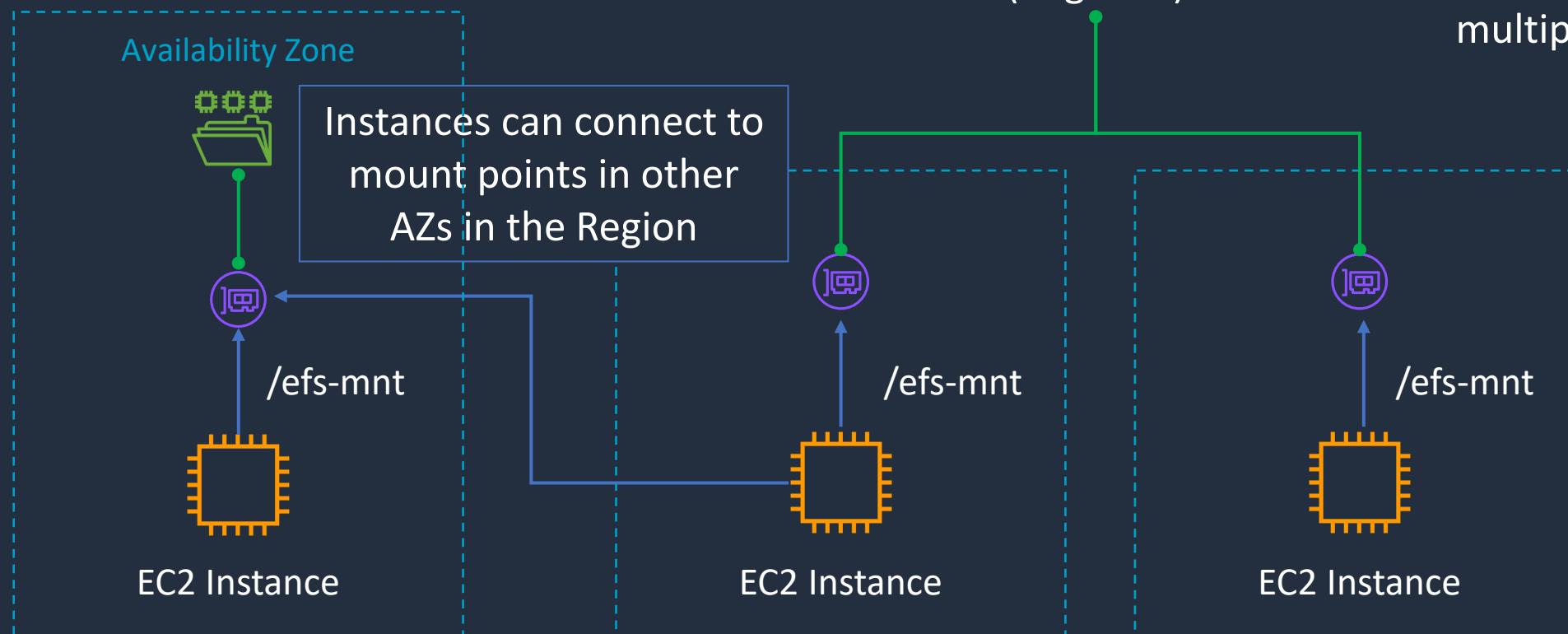




Amazon Elastic File System (EFS)

Note: EFS supports **Linux** only

One Zone file systems have mount targets in a single AZ



EFS File system
(Regional)

Regional file systems
have mount targets in
multiple AZs

The connection protocol is **NFS**

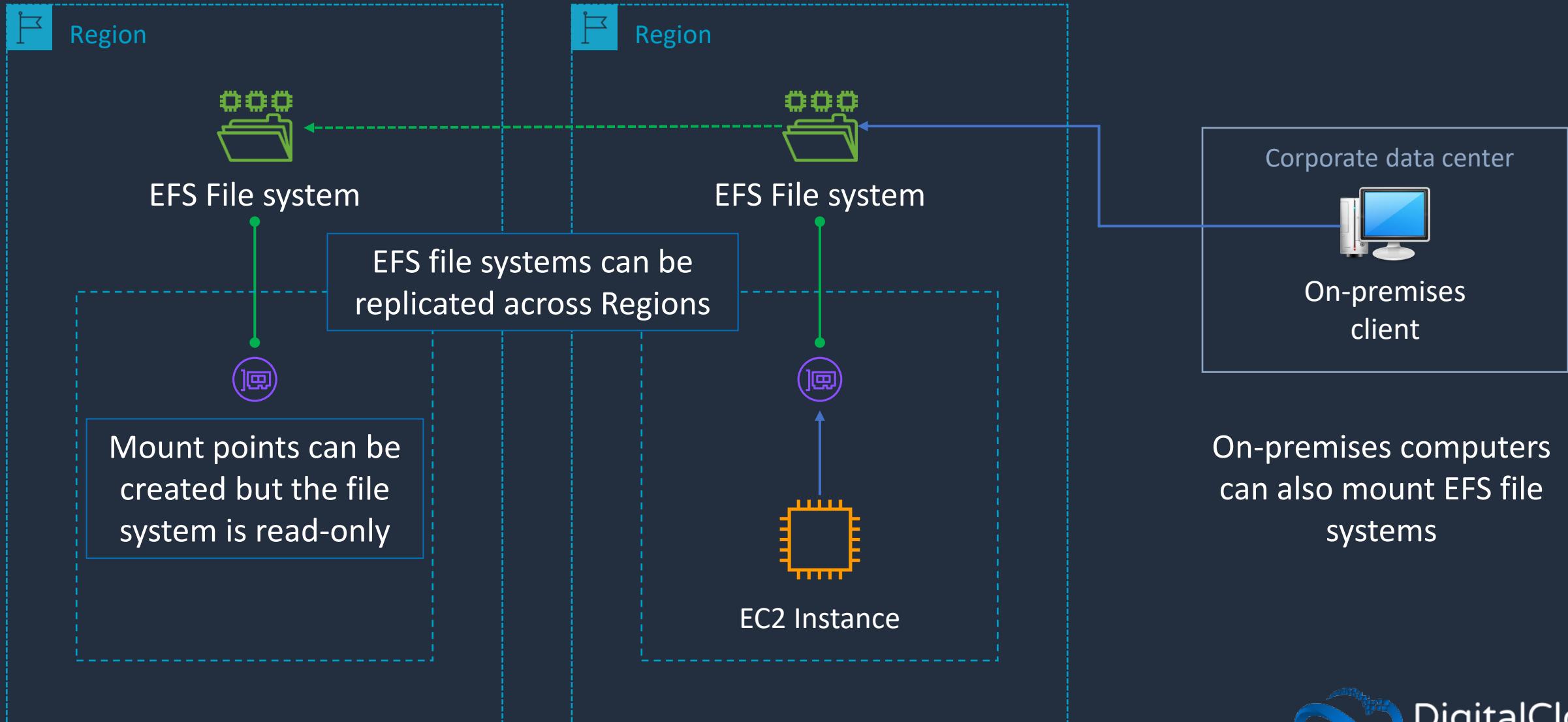


Amazon Elastic File System (EFS)

- **Data consistency** – write operations for Regional file systems are durably stored across Availability Zones
- **File locking** – NFS client applications can use NFS v4 file locking for read and write operations on EFS files
- **Storage classes** – there are three options:
 - **EFS Standard** – uses SSDs for low latency performance
 - **EFS Infrequent Access (IA)** – cost effective option
 - **EFS Archive** – even cheaper for less active data (archival)
- **Durability** – all storage classes offer 11 9s of durability



Amazon Elastic File System (EFS)





Amazon Elastic File System (EFS)

- **EFS Replication** – data is replicated across Regions for disaster recovery purposes with RPO/RTO in the minutes
- **Automatic Backup** – EFS integrates with AWS Backup for automatic file system backups
- **Performance options** – there are two options:
 - **Provisioned throughput** – Specify a level of throughput that the file system can drive independent of the file system's size
 - **Bursting throughput** – Throughput scales with the amount of storage and supports bursting to higher levels

Create an Amazon EFS File System



Amazon Simple Storage Service (S3)





Amazon Simple Storage Service (S3)



S3 Bucket



A **bucket** is a container for objects

An **object** is a file you upload

You can store millions of **objects** in a **bucket**



Accessing objects in a bucket:

`https://bucket.s3.aws-region.amazonaws.com/key`
`https://s3.aws-region.amazonaws.com/bucket/key`

The **HTTP protocol** is used with a **REST API** (e.g. GET, PUT, POST, SELECT, DELETE)



Amazon Simple Storage Service (S3)



Bucket

A **bucket** is a container for objects

<http://bucket.s3.aws-region.amazonaws.com>

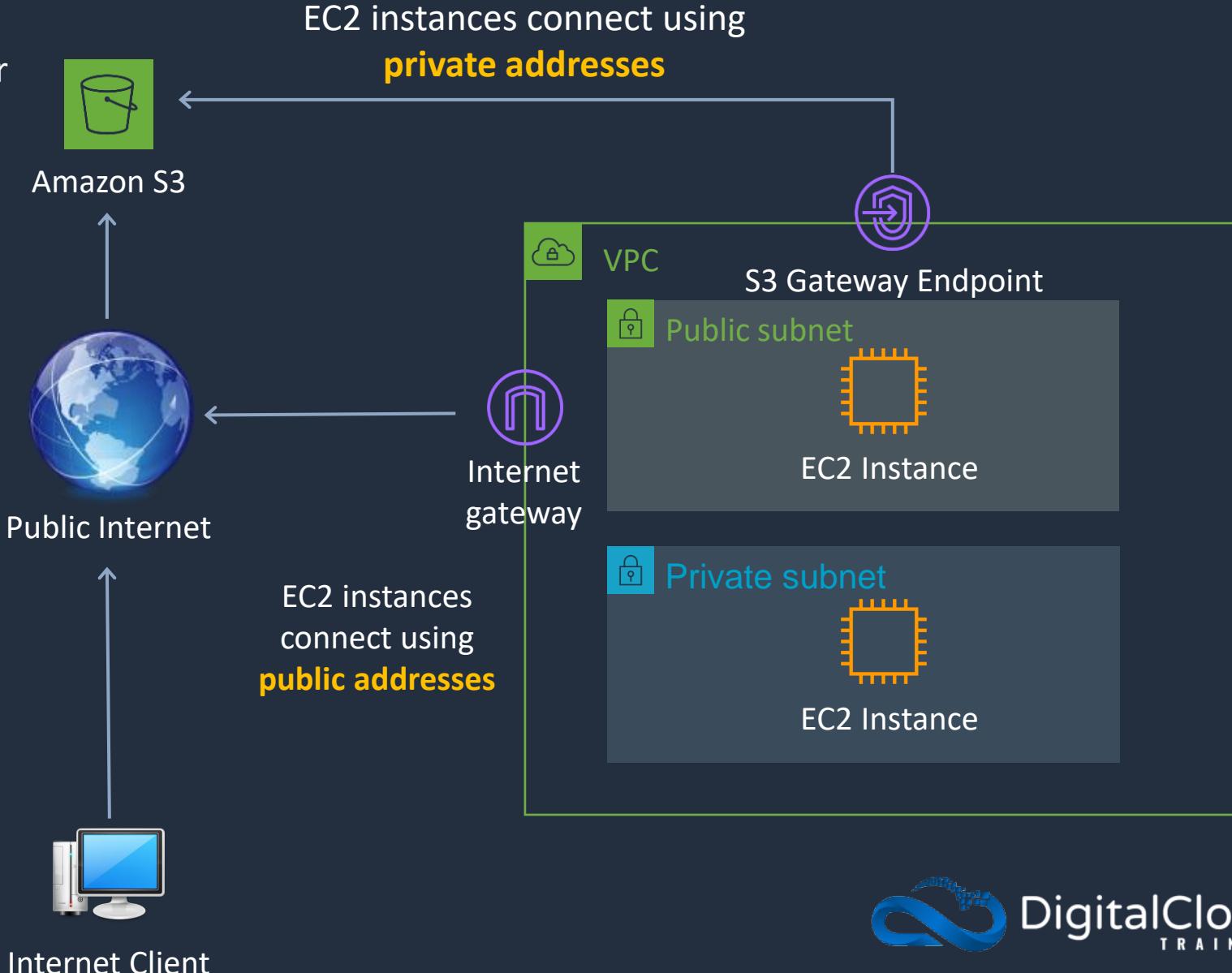
<http://s3.aws-region.amazonaws.com/bucket>



Object

An object consists of:

- Key (name of objects)
- Version ID
- Value (actual data)
- Metadata
- Subresources
- Access control information





File Storage vs Object Storage

File Share



- Data stored in directories
- A hierarchy of directories can be formed
- File systems are mounted to an operating system
- Function like local storage
- Network connection is maintained
- Example is Amazon EFS

Object Store



`http://bucket.s3.aws-region.amazonaws.com`

- Data stored in buckets
- Flat namespace (no hierarchy)
- Hierarchy can be mimicked with prefixes
- Accessed by **REST API** and cannot be mounted
- Network connection is completed after each request
- Example is Amazon S3

Working with S3 Buckets and Objects



Create an Amazon S3 Static Website



SECTION 7

AWS Database Services

Database Types and Use Cases





Relational vs Non-Relational

Key differences are how data are *managed* and how data are *stored*

Relational	Non-Relational
Organized by tables, rows and columns	Varied data storage models
Rigid schema (SQL)	Flexible schema (NoSQL) – data stored in key-value pairs, columns, documents or graphs
Rules enforced within database	Rules can be defined in application code (outside database)
Typically scaled vertically	Scales horizontally
Supports complex queries and joins	Unstructured, simple language that supports any kind of schema
Amazon RDS, Oracle, MySQL, IBM DB2, PostgreSQL	Amazon DynamoDB, MongoDB, Redis, Neo4j



Relational Databases

EmployeeID	FirstName	LastName	JobRole	Location
00001	Paul	Peterson	Senior Developer	Atlanta
00002	Kaleigh	Annette	Assistant Manager	Miami
00003	Carl	Wood	Sales Support	New York
00004	Vinni	Jones	Customer Service	Dallas
00005	Stefanie	Howard	IT Architect	Los Angeles

SQL is used for defining the structure of the database and its elements

SQL provides the tools for inserting, updating, deleting, and querying data within the database table

Example **Structured Query Language** (SQL) query:

```
SELECT FirstName  
FROM employees  
WHERE Location = Atlanta
```



Non-Relational Databases

NoSQL databases can be **key/value** and **document** stores

This is an example of a **key/value** store

Primary Key		Attributes				
Partition Key	Sort Key	sku	category	size	color	weight
john@example.com	1583975308	SKU-S523	T-Shirt	Small	Red	Light
chris@example.com	1583975613	SKU-J091	Pen		Blue	
chris@example.com	1583975449	SKU-A234	Mug			
sarah@example.com	1583976311	SKU-R873	Chair			
jenny@example.com	1583976323	SKU-I019	Plate	30		

There is no rigid schema so attributes can be missing or have different data types



Graph Databases

Graph databases like Amazon Neptune are designed to store, manage, and navigate relationships in data

Graph databases use:

- **Nodes** to represent entities
- **Edges** to represent relationships
- **Properties** to store information about nodes and edges





Operational vs Analytical

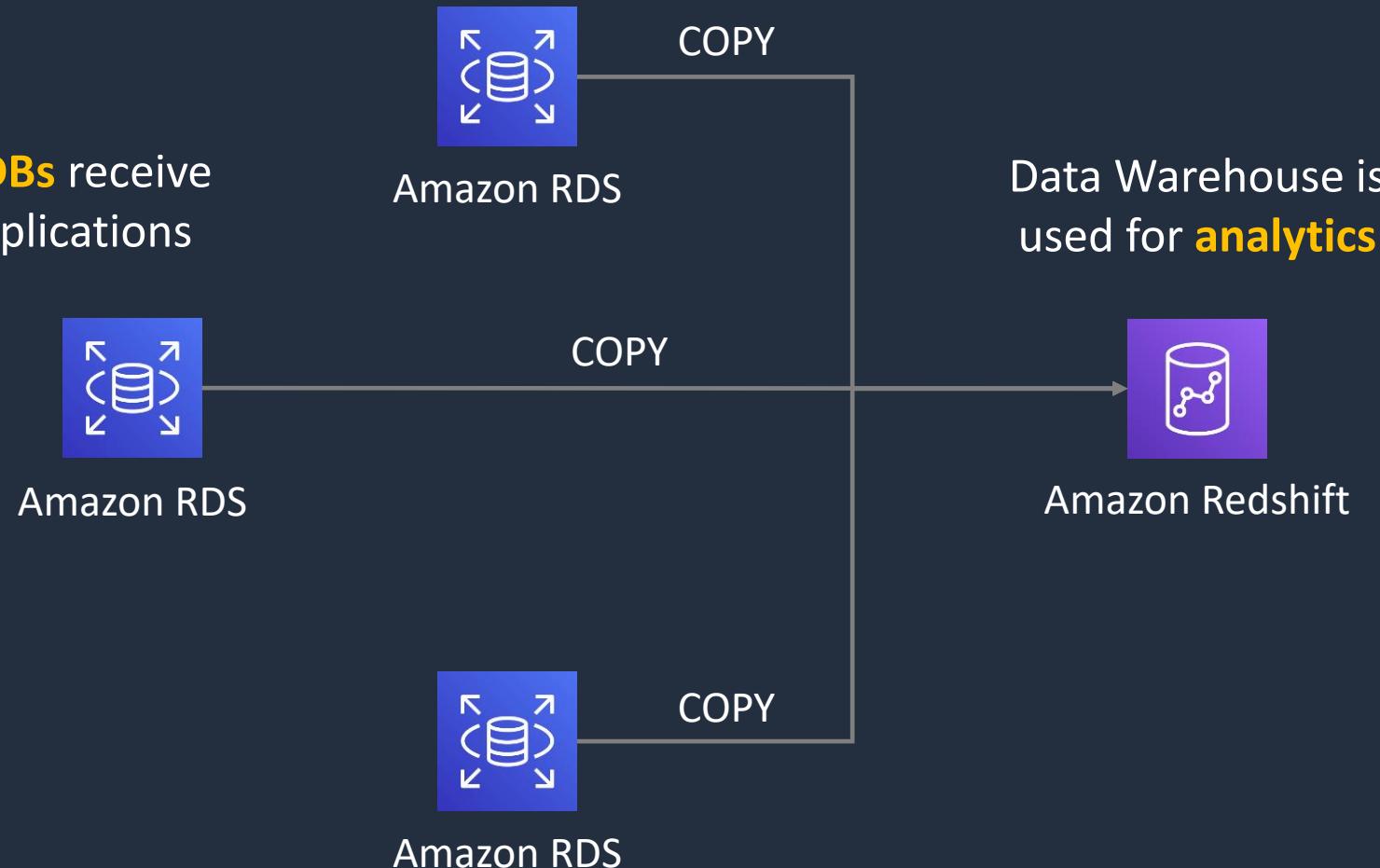
Key differences are **use cases** and how the database is **optimized**

Operational / transactional	Analytical
Online Transaction Processing (OLTP)	Online Analytics Processing (OLAP) – the source data comes from OLTP DBs
Production DBs that process transactions. E.g. adding customer records, checking stock availability (INSERT, UPDATE, DELETE)	Data warehouse. Typically, separated from the customer facing DBs. Data is extracted for decision making
Short transactions and simple queries	Long transactions and complex queries
Relational examples: Amazon RDS, Oracle, IBM DB2, MySQL	Relational examples: Amazon RedShift, Teradata, HP Vertica
Non-relational examples: MongoDB, Cassandra, Neo4j, HBase	Non-relational examples: Amazon EMR, MapReduce



Operational vs Analytical

Operational DBs receive data from applications





AWS Databases

Data Store	Use Case
Database on EC2	<ul style="list-style-type: none">• Need full control over instance and database• Third-party database engine (not available in RDS)
Amazon RDS	<ul style="list-style-type: none">• Need traditional relational database• e.g. Oracle, PostgreSQL, Microsoft SQL, MariaDB, MySQL• Data is well-formed and structured
Amazon DynamoDB	<ul style="list-style-type: none">• NoSQL database• In-memory performance• High I/O needs• Dynamic scaling
Amazon RedShift	<ul style="list-style-type: none">• Data warehouse for large volumes of aggregated data
Amazon ElastiCache	<ul style="list-style-type: none">• Fast temporary storage for small amounts of data• In-memory database

Amazon Relational Database Service (RDS)

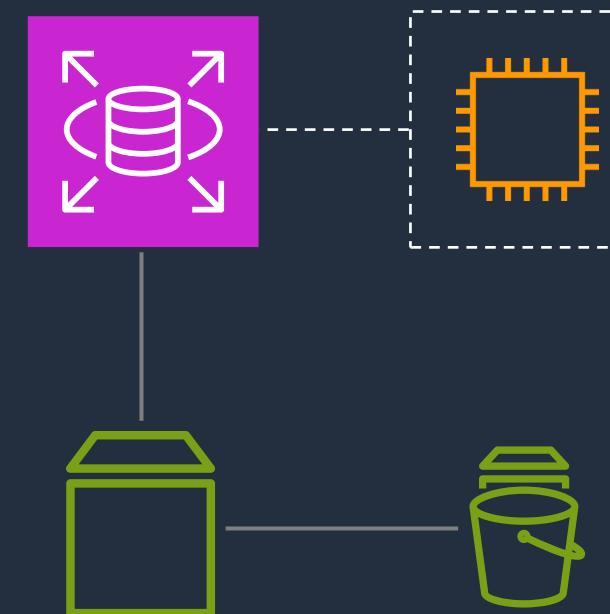




Amazon RDS

- Managed **relational** database service
- Used for online transaction processing (OLTP) use cases
- Runs on Amazon EC2 instances

A DB instance can contain
multiple user-created databases



You must choose the
DB instance type

RDS uses **Amazon EBS** volumes
for storage

Backups can be taken using
EBS snapshots



Amazon RDS

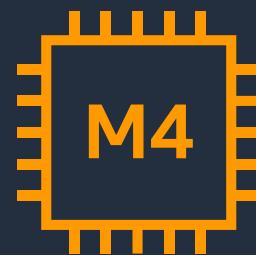
Amazon RDS offers a choice of database engines:

- **Amazon Aurora** – (MySQL and PostgreSQL compatible)
- **MySQL** – One of the most popular open-source relational database management systems
- **PostgreSQL** – An advanced open-source relational database that supports both SQL (relational) and JSON (non-relational) querying
- **Oracle** – Offers support for Oracle Database under two licensing models: "License Included" and "Bring Your Own License (BYOL)"
- **Microsoft SQL Server** – Supports several editions of Microsoft SQL Server
- **MariaDB** – A community-developed fork of MySQL intended to remain free under the GNU GPL



Amazon RDS – Scaling up (vertically)

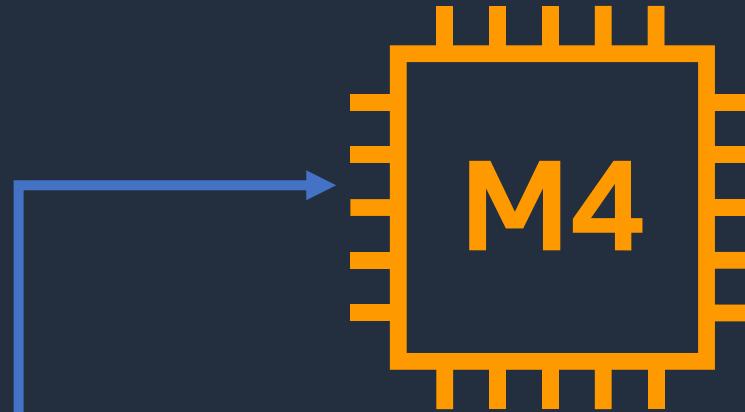
RDS scales up by changing
the **instance type**



M4 instance

db.m4.large 2 vCPUs,
8 GiB RAM

The database must shut
down and restart



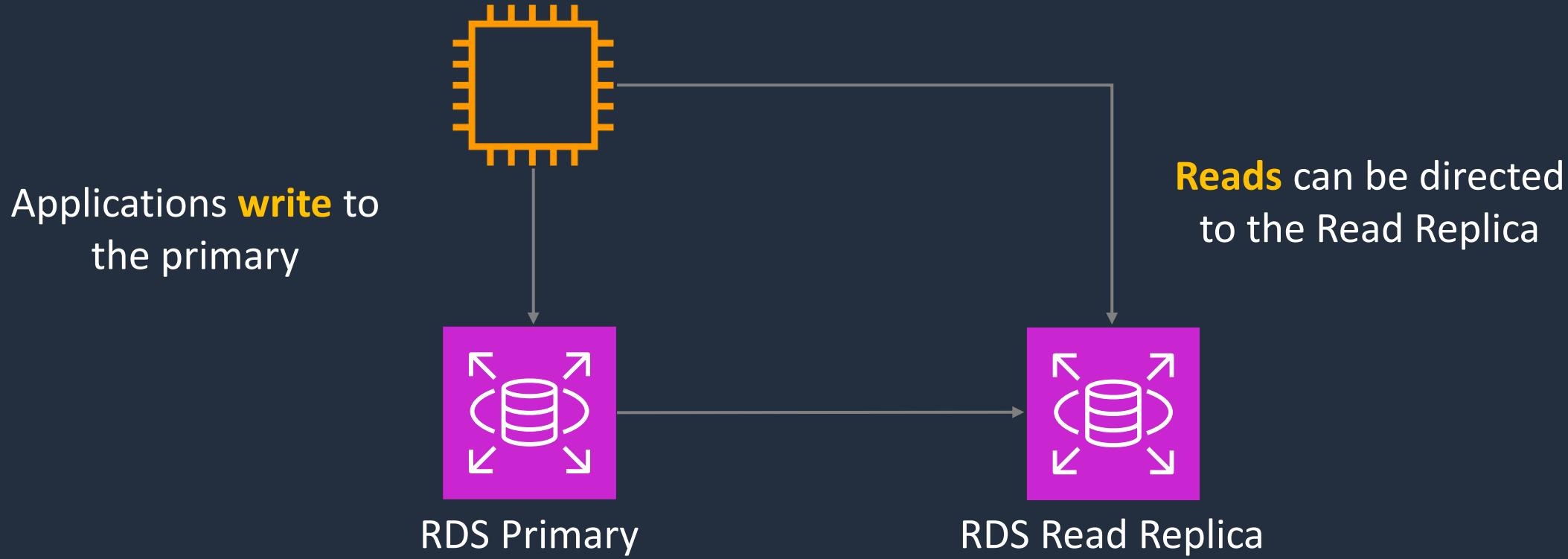
M4 Instance

db.m4.2xlarge 4
vCPUs, 32 GiB RAM

Scaling up is required for
better **write** performance



Amazon RDS – Scaling out (horizontally)

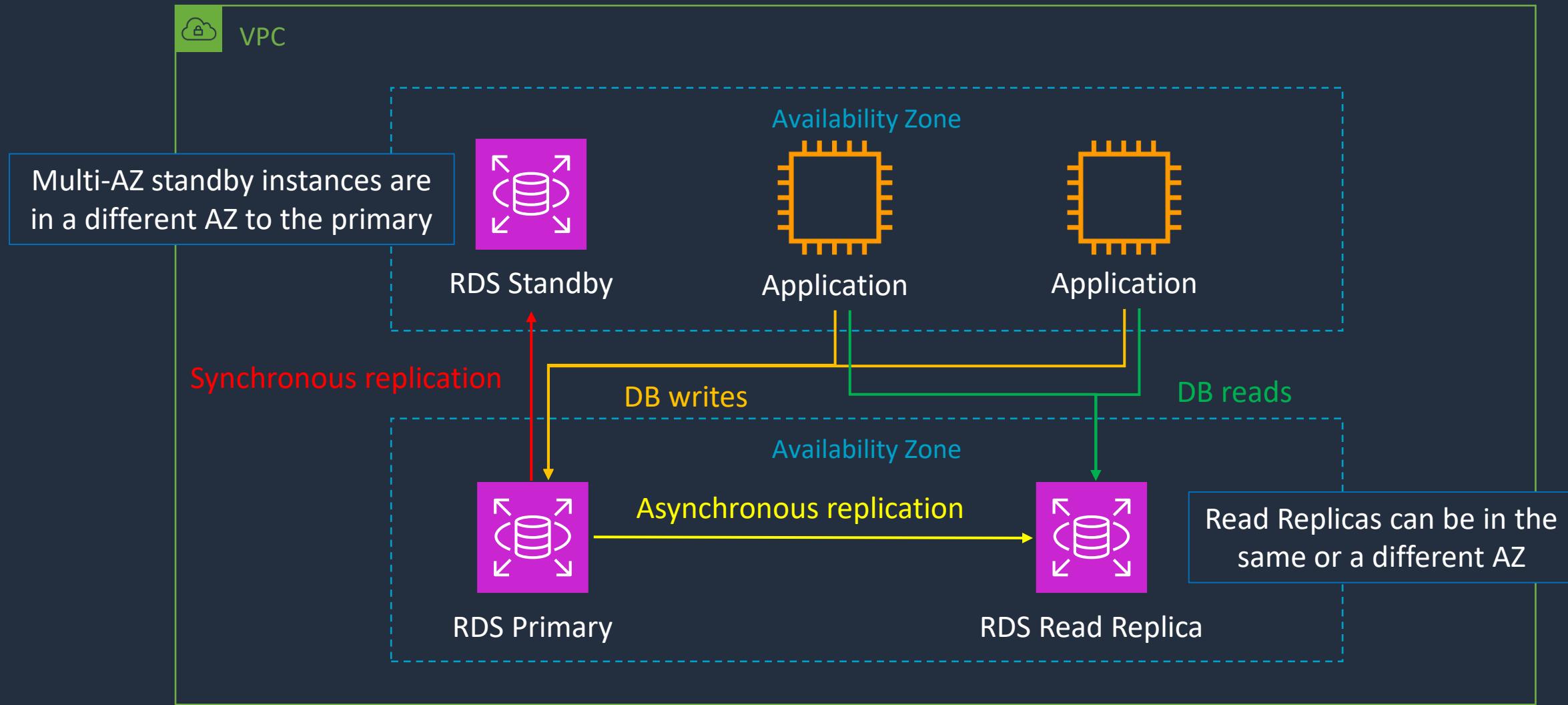


RDS scales **reads** horizontally with Read Replicas



Amazon RDS Multi AZ

Multi-AZ deployments enable automatic **disaster recovery (DR)**



Create an Amazon RDS Database



Amazon DynamoDB





Amazon DynamoDB

- DynamoDB is a fully managed NoSQL database service
- DynamoDB is a fully serverless service
- Key/value store and document store
- Low latency access to data (milliseconds)
- Offers push button scaling with no downtime



Data is stored in **partitions** which are replicated across multiple AZs in a Region



DynamoDB Features

DynamoDB Feature	Benefit
Serverless	Fully managed, fault tolerant, service
Highly available	99.99% availability SLA – 99.999% for Global Tables!
NoSQL type of database	Flexible schema, good for when data is not well structured or unpredictable
Horizontal scaling	Seamless scalability to any scale with push button scaling or Auto Scaling
DynamoDB Streams	Captures a time-ordered sequence of item-level modifications in a DynamoDB table
Transaction options	Strongly consistent or eventually consistent reads, support for ACID transactions
Backup	Point-in-time recovery down to the second in last 35 days; On-demand backup and restore
Global Tables	Fully managed multi-region, multi-master solution



DynamoDB Core Components

The basic DynamoDB components are:



Tables



Items



Attributes

userid	orderid	book	price	date
user001	1000092	ISBN100..	9.99	2020.04..
user002	1000102	ISBN100..	24.99	2020.03..
user003	1000168	ISBN2X0..	12.50	2020.04..



Amazon DynamoDB Pricing

Amazon DynamoDB pricing is primarily based on:

- **Provisioned Throughput:** You can choose provisioned capacity, paying for the amount of reads and writes per second that you allocate for your table
- **On-Demand Capacity:** You pay for the read and write requests your application performs on your tables without managing capacity planning
- **Storage Costs:** You pay for the data storage your tables consume
- **Additional features:** Costs apply for using Global Tables, DynamoDB Streams, backup and restore, and data transfer

Create an Amazon DynamoDB Table



SECTION 8

Automation and DevOps on AWS

Infrastructure as Code with AWS CloudFormation





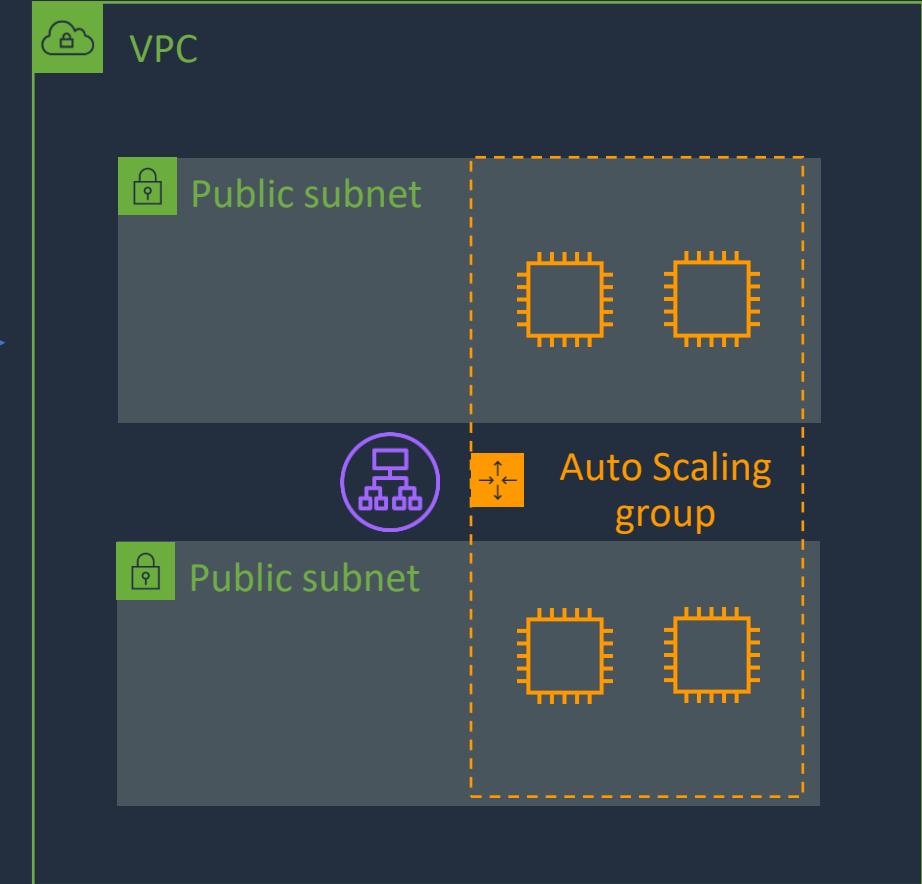
AWS CloudFormation

Infrastructure patterns are defined in a **template** file using **code**



CloudFormation **builds** your infrastructure according to the **template**

```
1 "AWSTemplateFormatVersion": "2010-09-09",
2
3 "Description": "AWS CloudFormation Sample Template WordPress_Multi_AZ: WordPress is web
4
5 "Parameters": {
6   "VpcId": {
7     "Type": "AWS::EC2::VPC::Id",
8     "Description": "VpcId of your existing Virtual Private Cloud (VPC)",
9     "ConstraintDescription": "must be the VPC Id of an existing Virtual Private Cloud."
10 },
11
12 "Subnets": {
13   "Type": "List<AWS::EC2::Subnet::Id>",
14   "Description": "The list of SubnetIds in your Virtual Private Cloud (VPC)",
15   "ConstraintDescription": "must be a list of at least two existing subnets associated
16 },
```





AWS CloudFormation

Component	Description
Templates	The JSON or YAML text file that contains the instructions for building out the AWS environment
Stacks	The entire environment described by the template and created, updated, and deleted as a single unit
StackSets	AWS CloudFormation StackSets extends the functionality of stacks by enabling you to create, update, or delete stacks across multiple accounts and regions with a single operation
Change Sets	A summary of proposed changes to your stack that will allow you to see how those changes might impact your existing resources before implementing them

Creating and Updating Stacks



Deploy a VPC Using CloudFormation



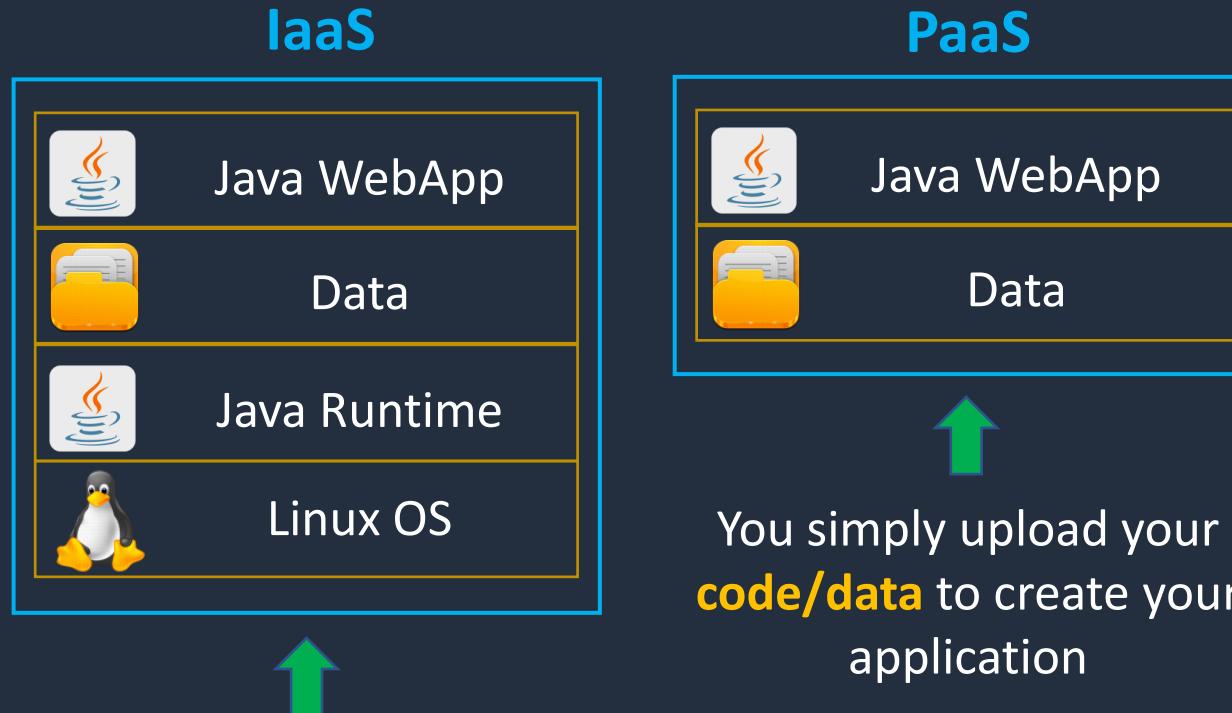
Platform as a Service with AWS Elastic Beanstalk





Cloud Service Models: Comparison

Example is
Amazon EC2



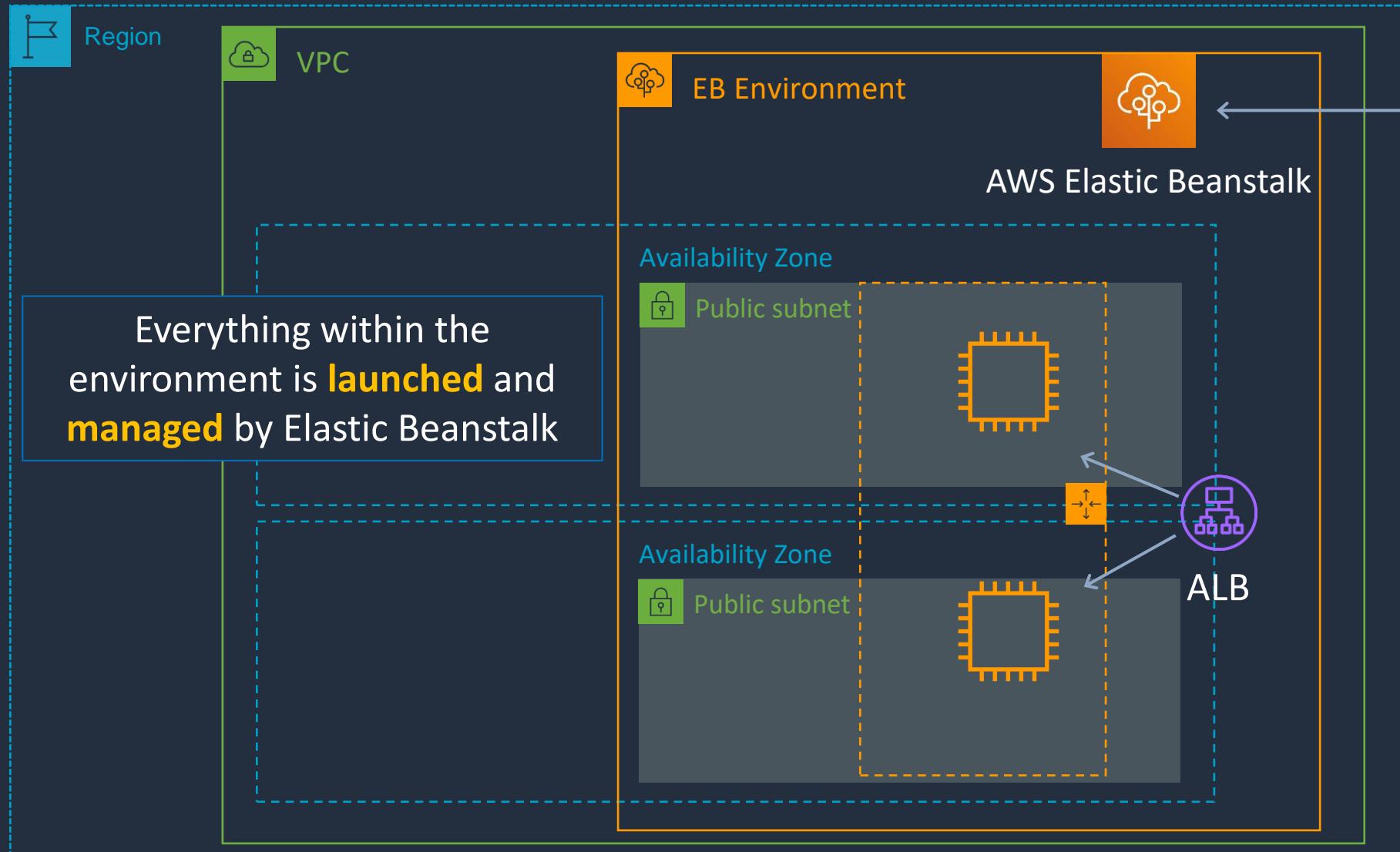
You manage from the
virtual server upwards

Example is **AWS Elastic Beanstalk**

You simply upload your
code/data to create your
application



AWS Elastic Beanstalk





AWS Elastic Beanstalk

- Supports many application platforms including:
 - Java, .NET, Node.js, PHP, Ruby, Python, Go, and Docker
- Uses core AWS services including EC2, ECS, Auto Scaling, and Elastic Load Balancing
- Elastic Beanstalk provides a UI to monitor and manage the health of applications
- Managed platform updates deploy the latest versions of software and patches



AWS Elastic Beanstalk

There are several **layers**

Applications:

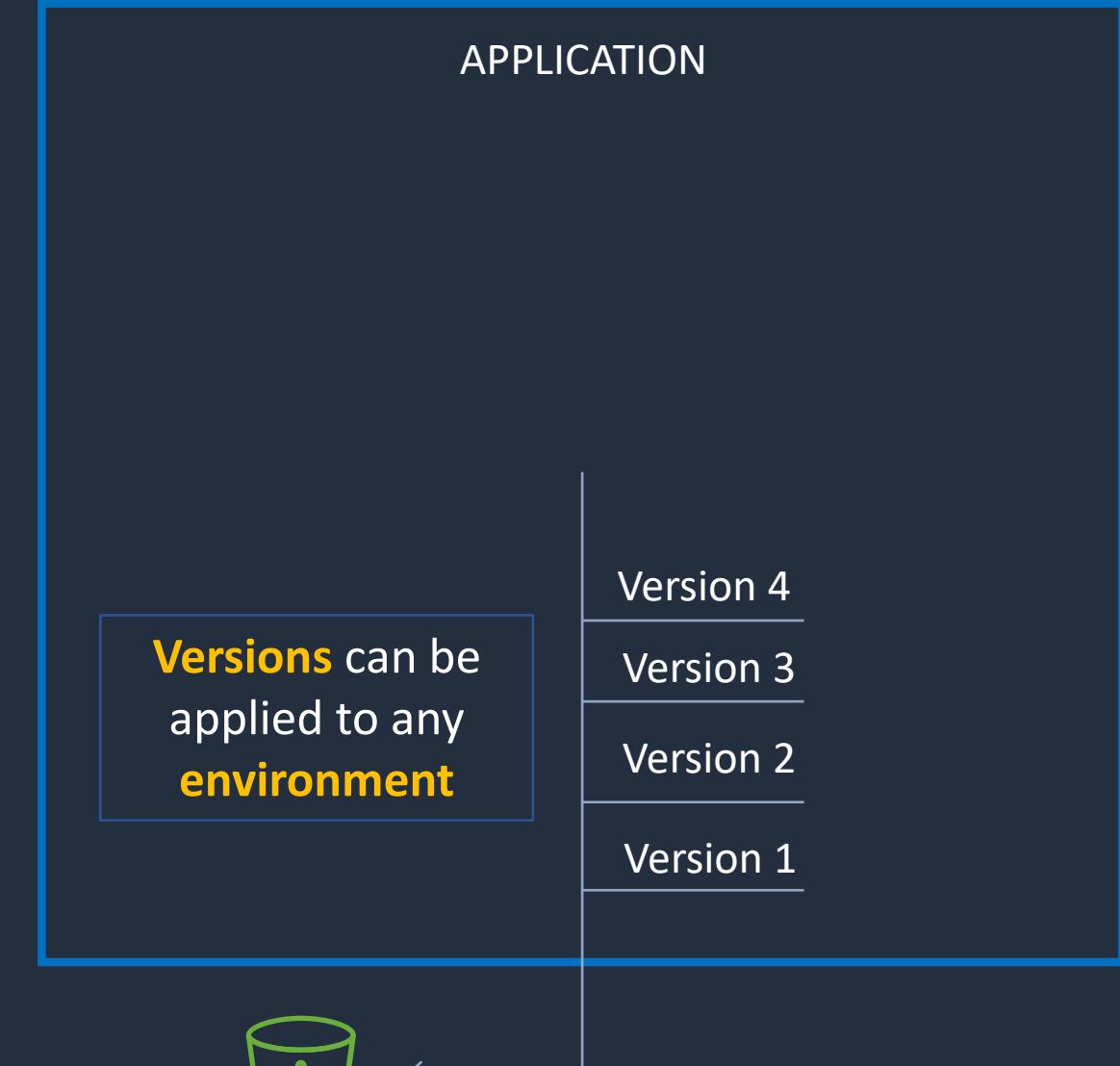
- Contain environments, environment configurations, and application versions
- You can have multiple application versions held within an application

APPLICATION



Application version

- A specific reference to a section of deployable code
- The application version will point typically to an Amazon S3 bucket containing the code



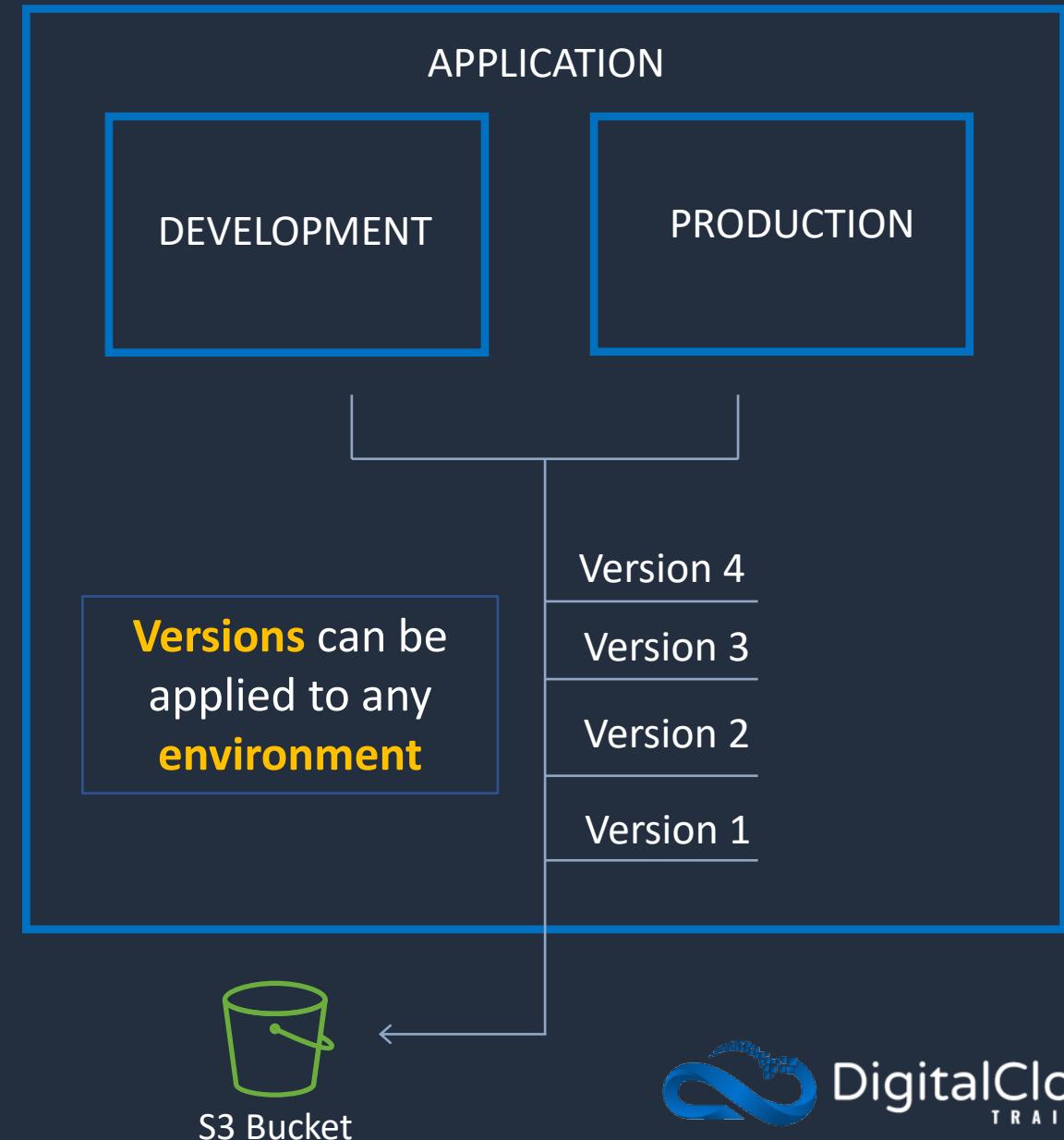
S3 Bucket



AWS Elastic Beanstalk

Environments:

- An application version that has been deployed on AWS resources
- The resources are configured and provisioned by AWS Elastic Beanstalk
- The environment is comprised of all the resources created by Elastic Beanstalk and not just an EC2 instance with your uploaded code



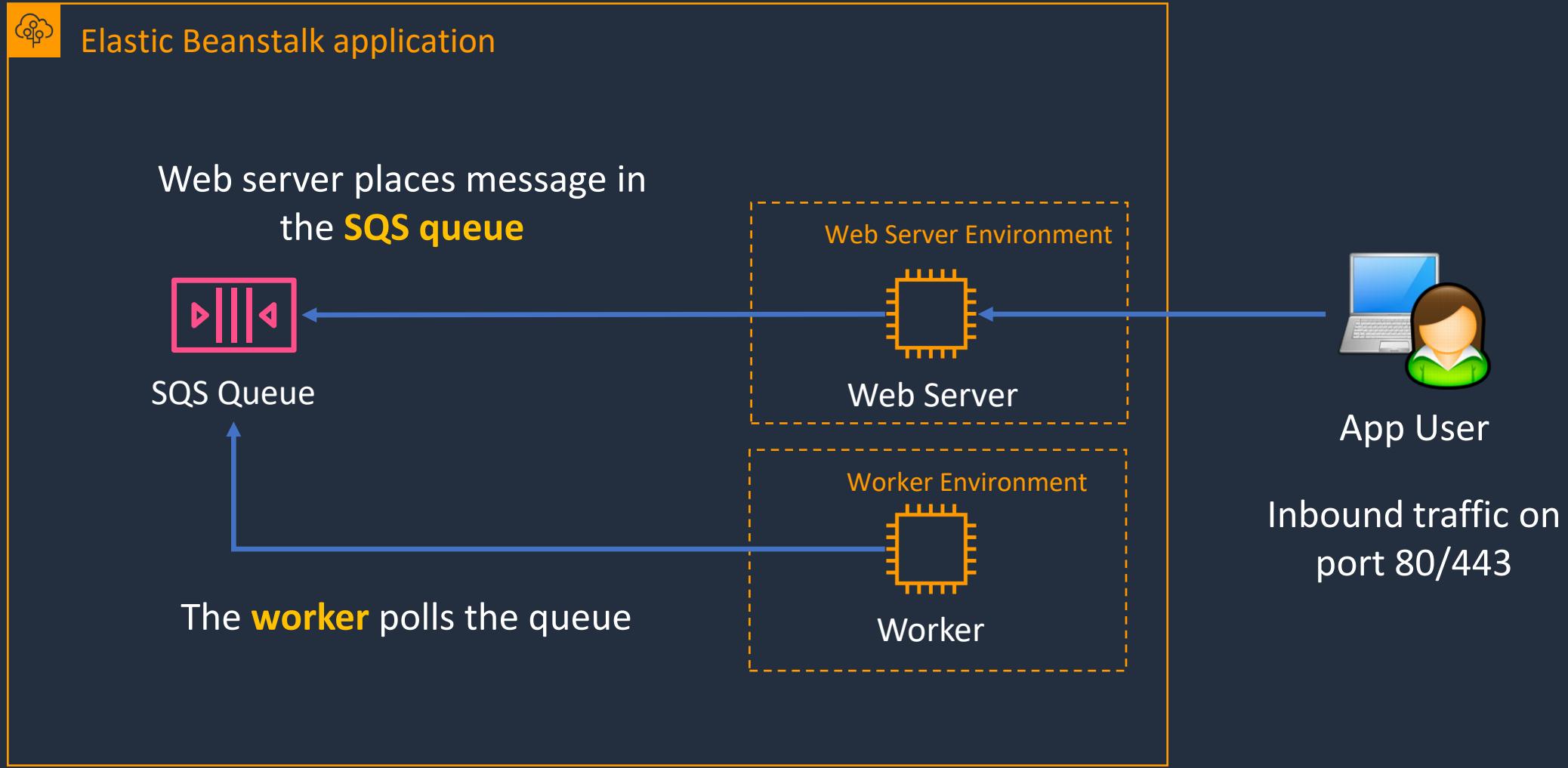


Web Servers and Workers

- **Web servers** are standard applications that listen for and then process HTTP requests, typically over port 80
- **Workers** are specialized applications that have a background processing task that listens for messages on an Amazon SQS queue
- **Workers** should be used for long-running tasks



AWS Elastic Beanstalk

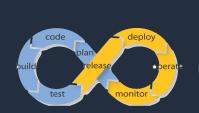


Create an Elastic Beanstalk Application



Continuous Integration and Continuous Delivery (CI/CD)

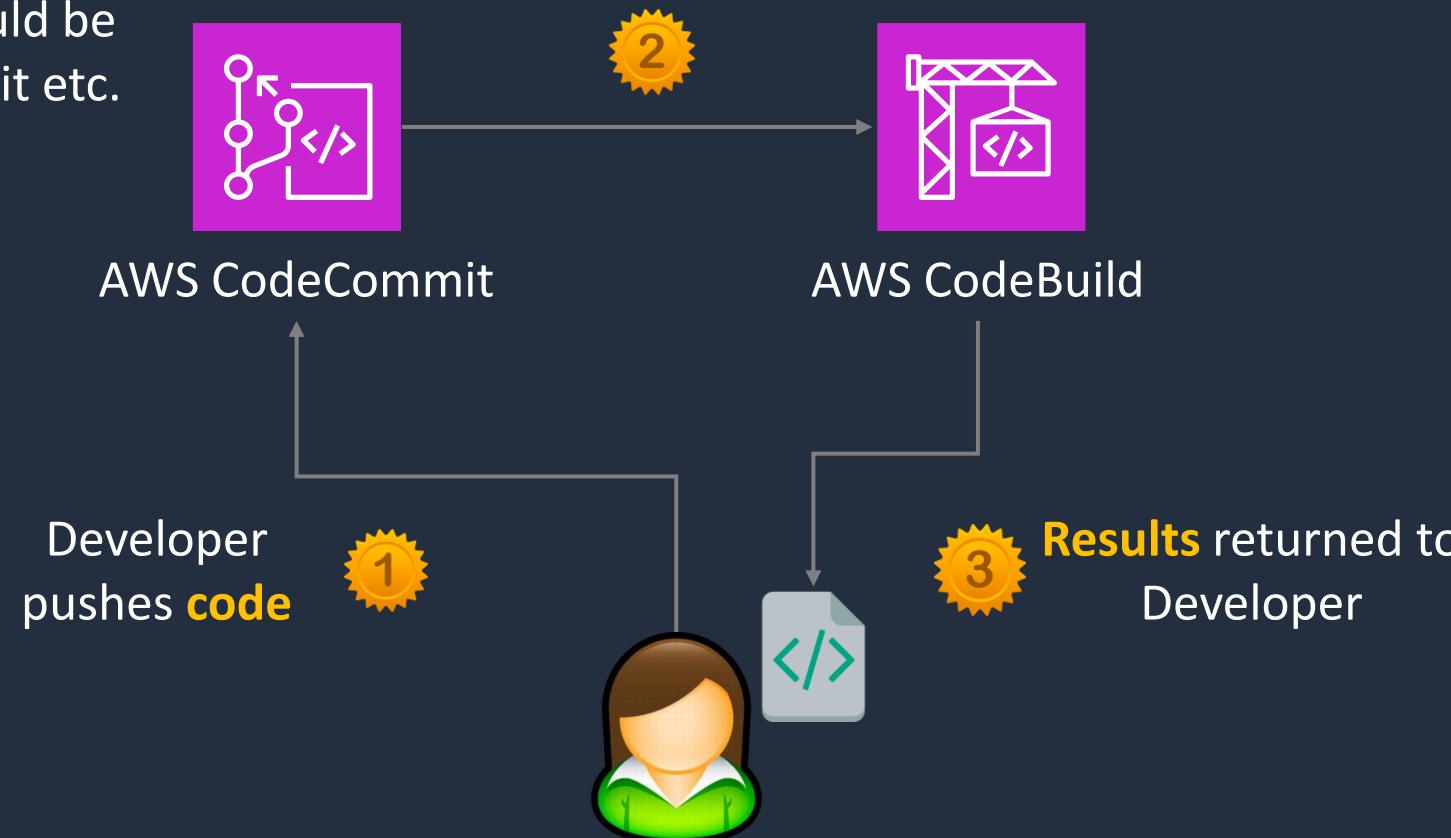




Continuous Integration

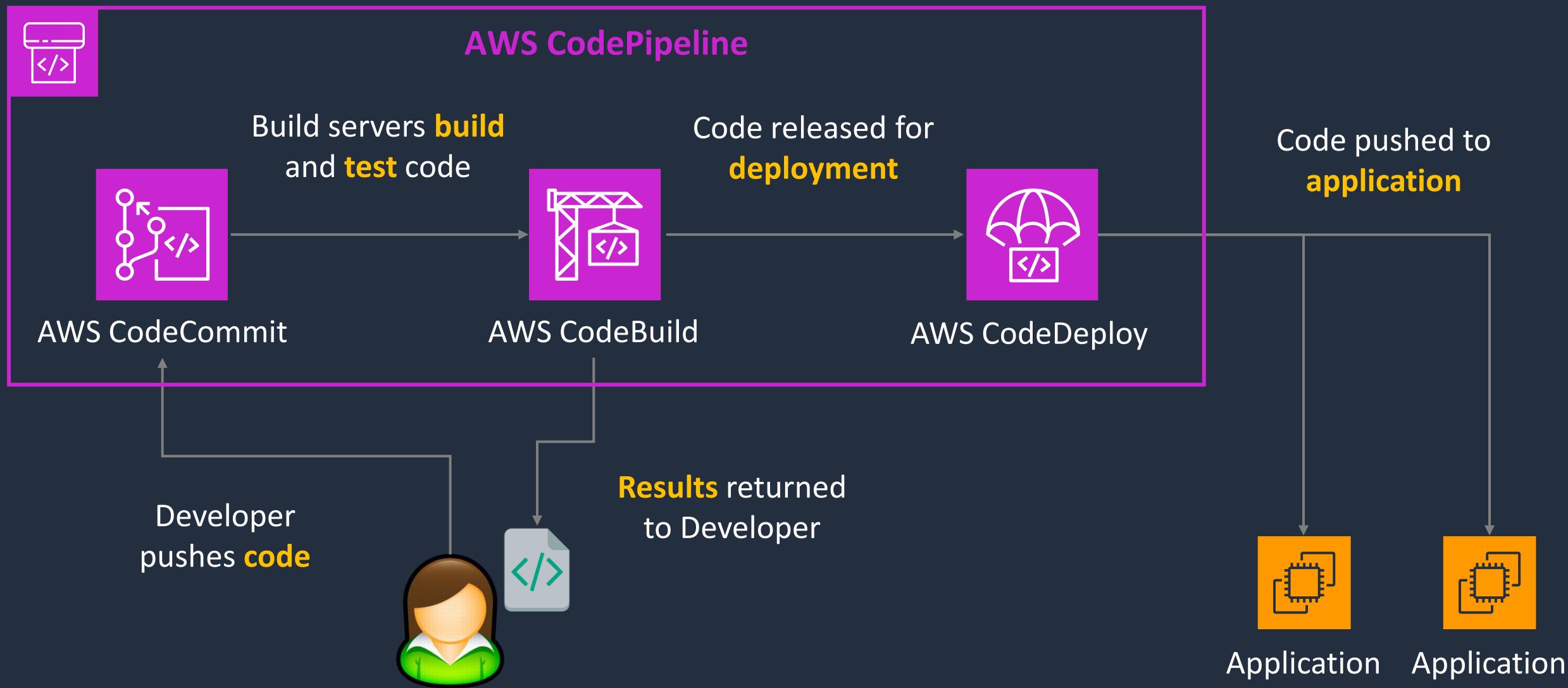
Code repository could be GitHub, CodeCommit etc.

Build servers **build** and **test** code





Continuous Integration & Continuous Delivery

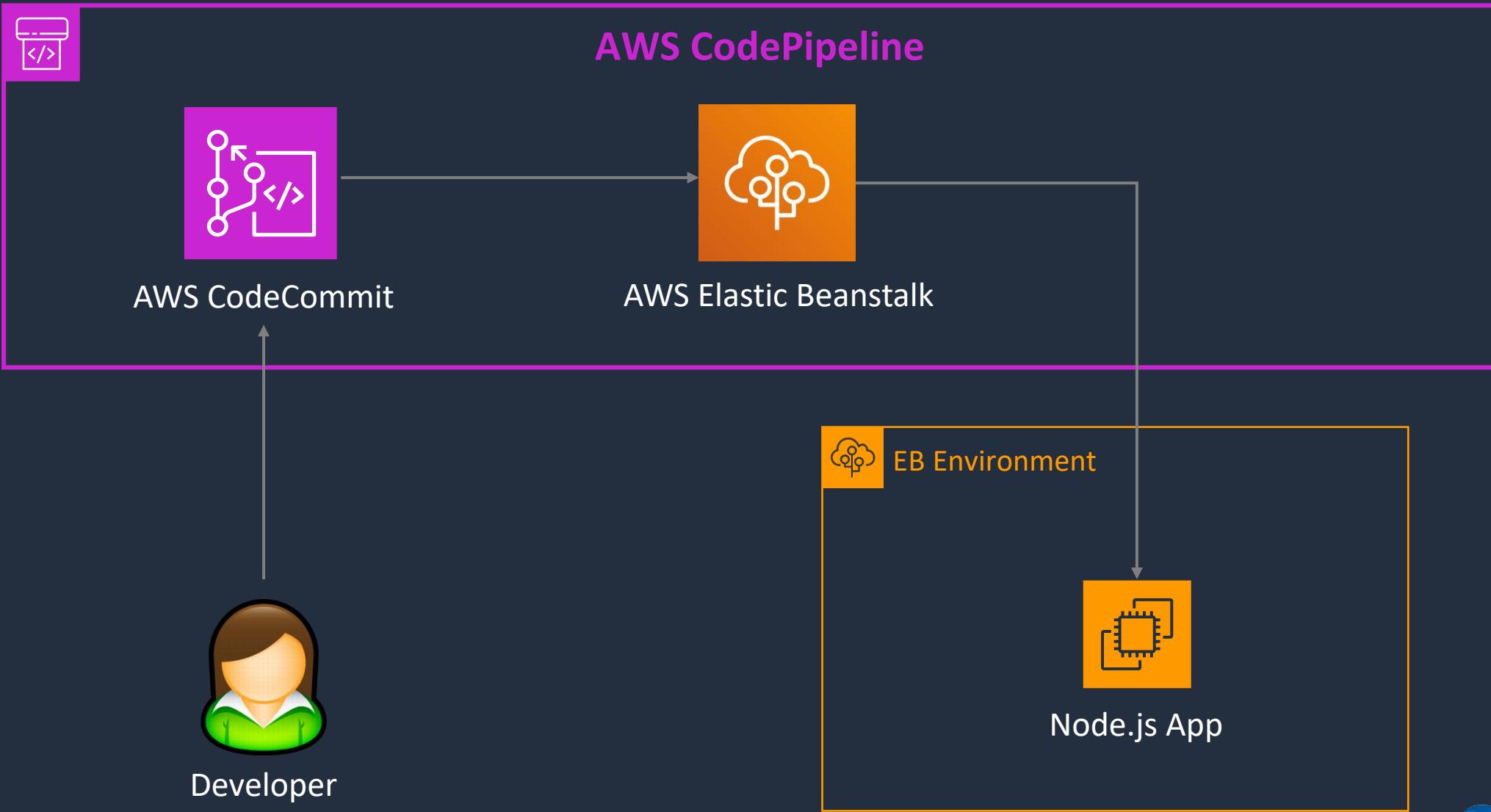


AWS CodePipeline with AWS Elastic Beanstalk





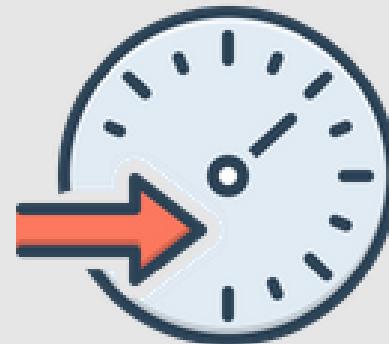
AWS CodePipeline with Elastic Beanstalk



SECTION 9

DNS, Caching, and Performance Optimization

Bandwidth and Latency





Bandwidth and Latency

Bandwidth is the **rate** of data transfer for a **fixed period of time** measured in **Gbps**



Bandwidth can be considered the **width** of the communication band

Latency is the amount of time it takes to send data from one point to another measured in **microseconds** or **milliseconds**

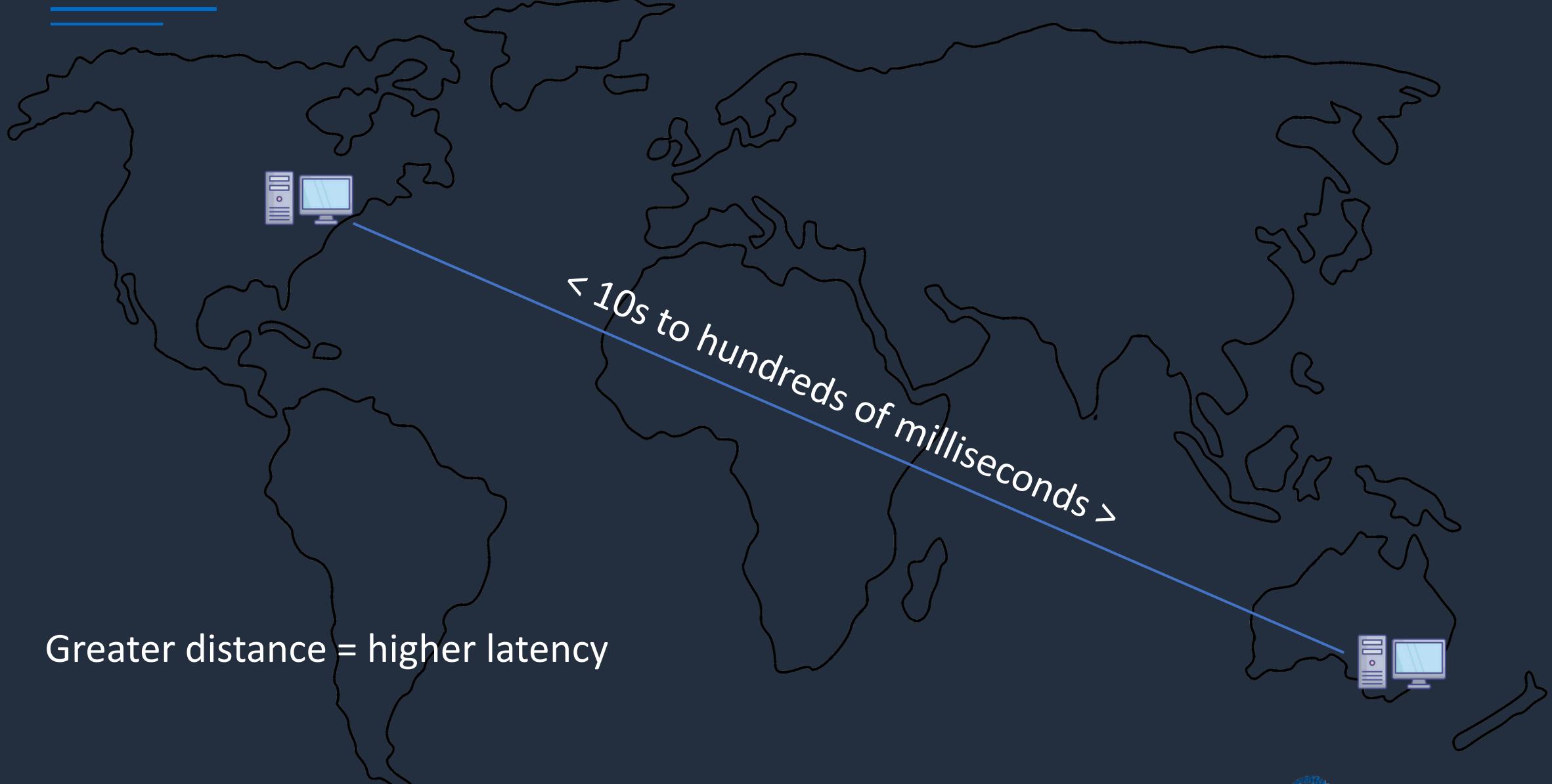


Bandwidth and Latency





Bandwidth and Latency





Bandwidth and Latency

Factors that contribute to network latency include:

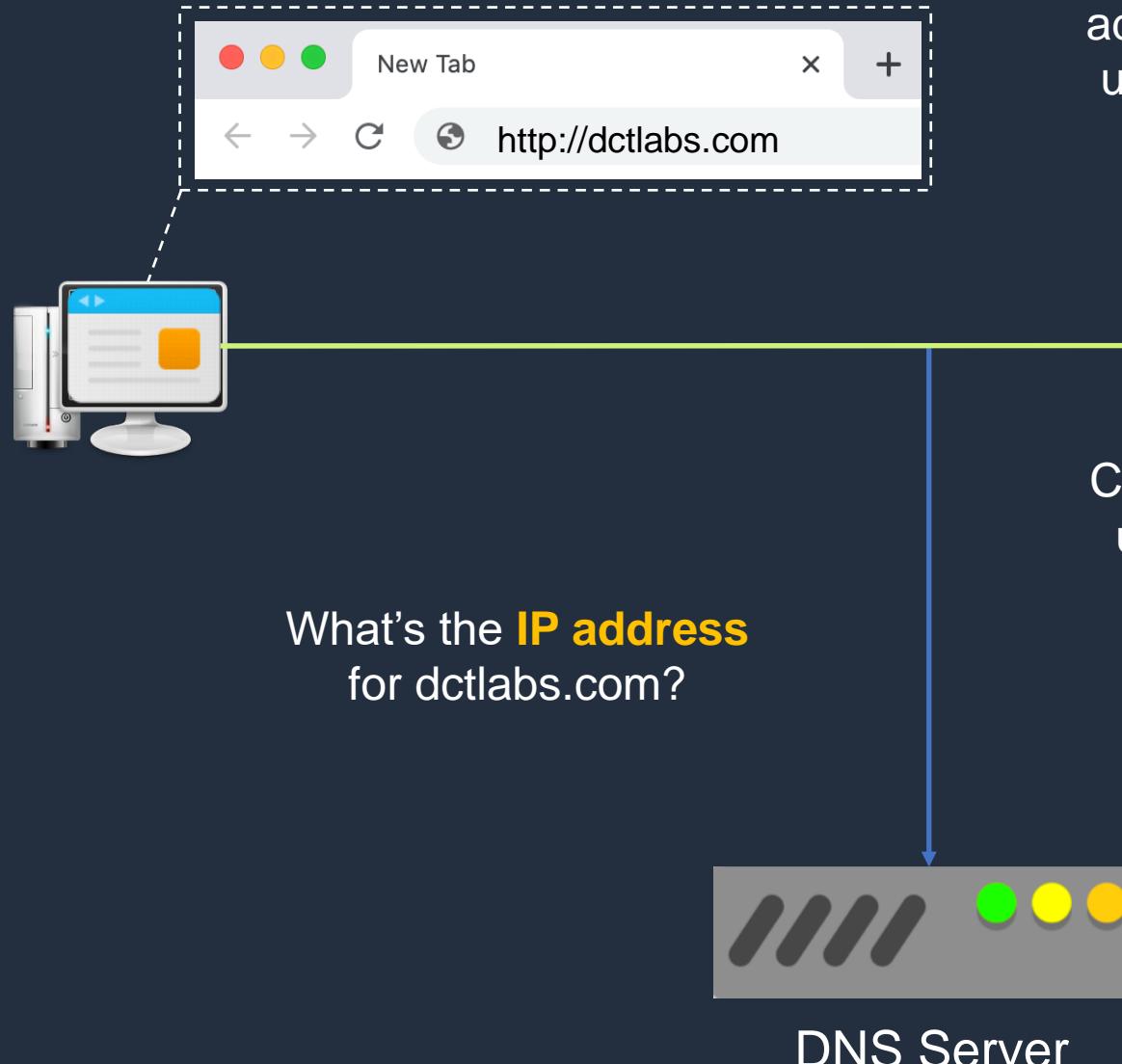
- **Propagation delay** – the time it takes for the signal to travel from source to destination (primarily determined by distance)
- **Transmission Delay** – the time required to push all the packet's bits into the wire, which depends on the packet size and the data rate of the link (bandwidth)
- **Queueing Delay** – the time a packet spends waiting in a queue until it can be processed or transmitted
- **Processing Delay** – the time routers or switches take to process the packet

DNS and Amazon Route 53





The Domain Name System



IP addresses are the addresses computers use to communicate

52.134.26.12



Web Server

Connection is made using **IP address**

Name	Type	Value
dctlabs.com	A	52.134.26.12
www.dctlabs.com	A	52.134.26.12

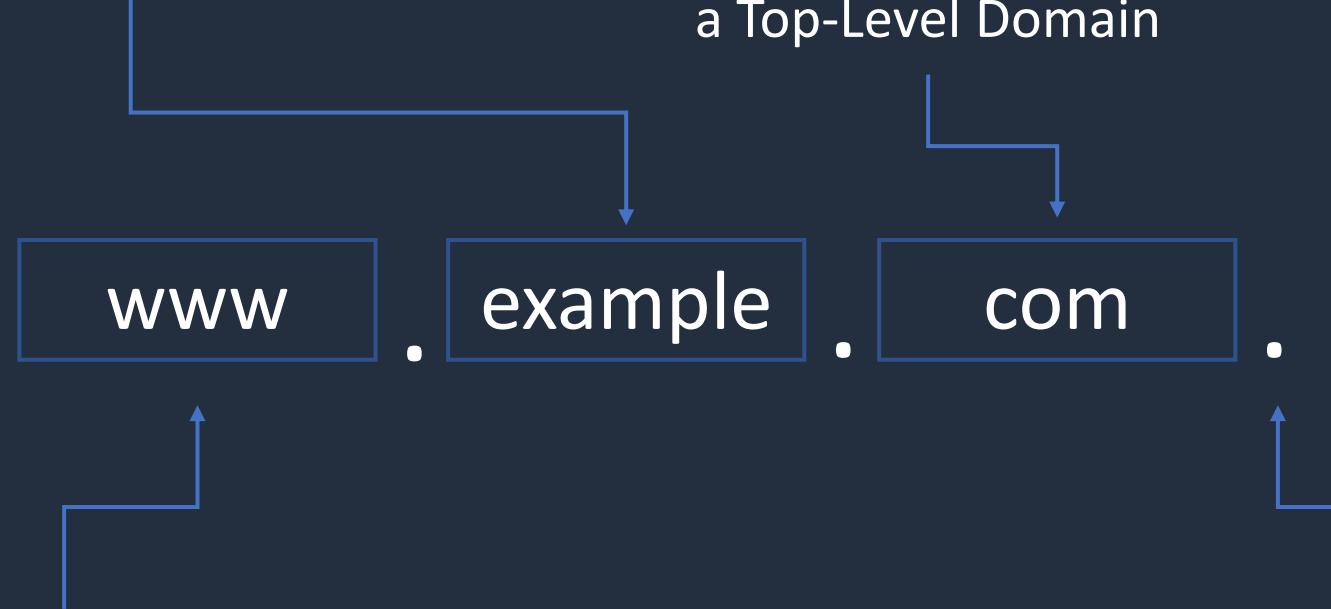
DNS Zone File



Fully Qualified Domain Names (FQDNs)

Example is a subdomain

com is an example of
a Top-Level Domain



www is a hostname within
the example subdomain

The **root domain** is represented by a
“.” And is not usually visible in a DNS
name



Subdomains

support is a subdomain
of amazon.com

A **subdomain** is subdivision of a domain name for organizing a set of related resources or services



mail is a subdomain
of google.com



DNS Zones and Records

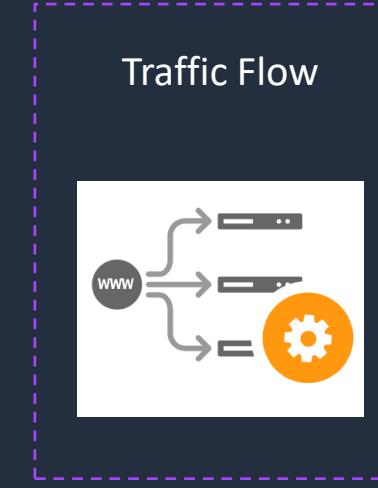
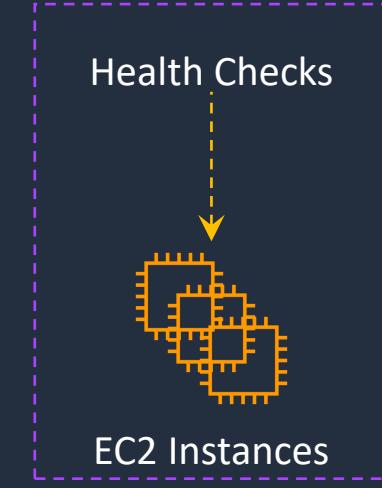
Record Type	Description
A	Maps a domain name to an IP address (e.g. dctlabs.com to 52.23.21.43)
CNAME	Maps a domain name to another domain name (e.g. mail.dctlabs.com to mailserver1.net)
MX	Returns the mail servers for a domain name
TXT	Associates text with a domain name (used for verification, authorization etc.)
SRV	Maps a domain name to a specific service or protocol (e.g. a Kerberos server)
NS	Specifies the authoritative DNS servers for a particular domain
SOA	Start of Authority record stores important information about the domain



Amazon Route 53

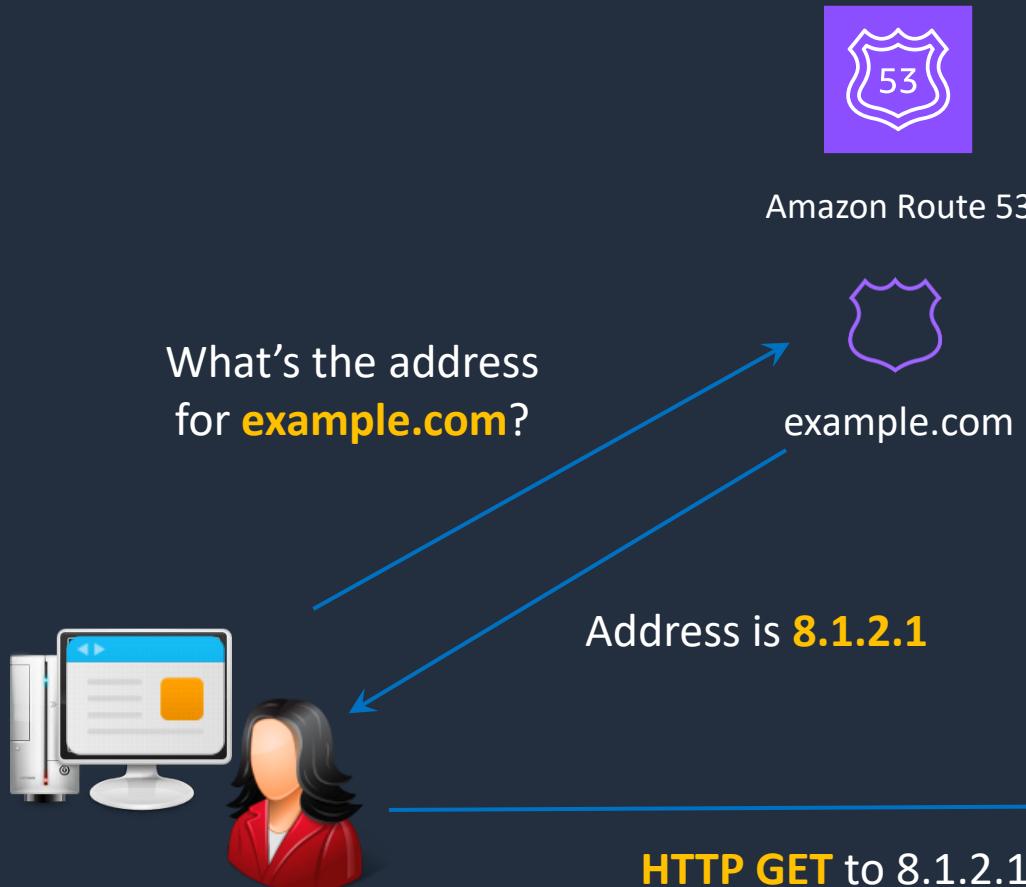


Amazon Route 53

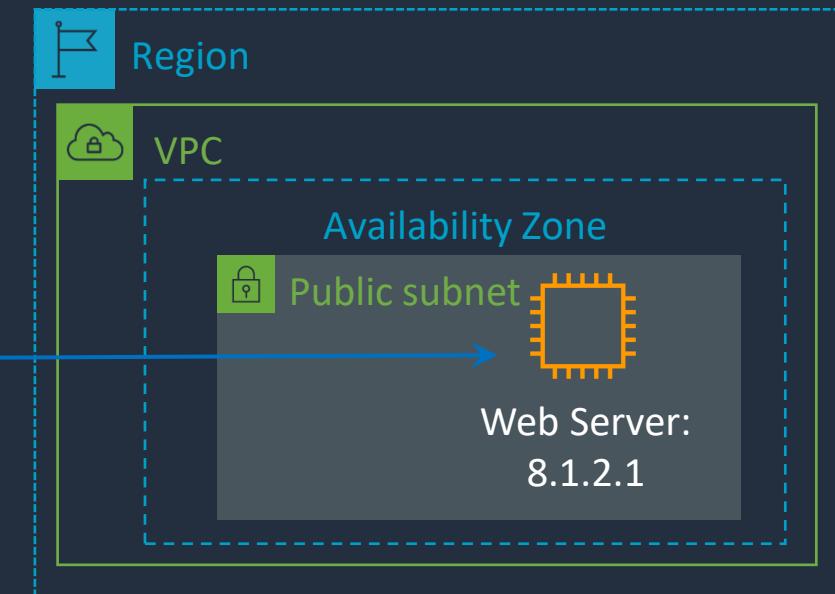




DNS Resolution with Route 53



A **hosted zone** represents a set of records belonging to a domain



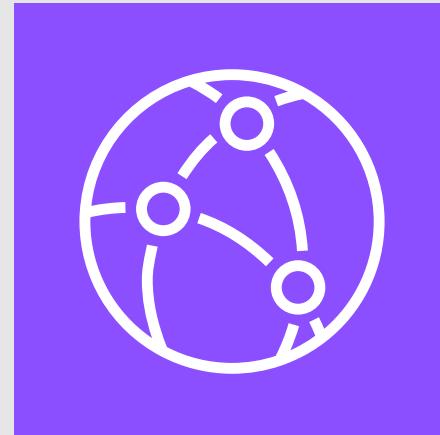
Route 53 Routing Policies

Routing Policy	What it does
Simple	Simple DNS response providing the IP address associated with a name
Failover	If primary is down (based on health checks), routes to secondary destination
Geolocation	Uses geographic location client is in (e.g. Europe) to route to the closest region
Geoproximity	Routes to the closest region within a geographic area
Latency	Directs based on the lowest latency route to resources
Multivalue answer	Returns several IP addresses and functions as a basic load balancer
Weighted	Uses the relative weights assigned to resources
IP Based	Route based on the originating IP address of the traffic

Register a Domain using Route 53



Amazon CloudFront





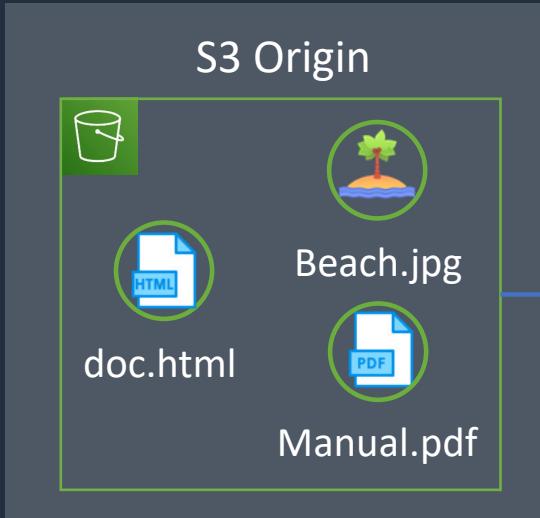
Content Delivery Network (CDN)



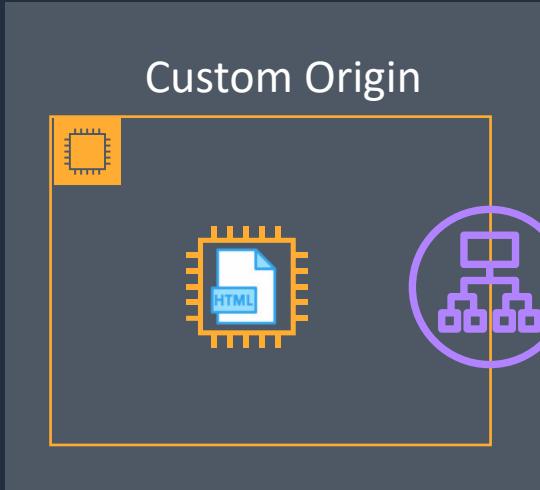


Amazon CloudFront

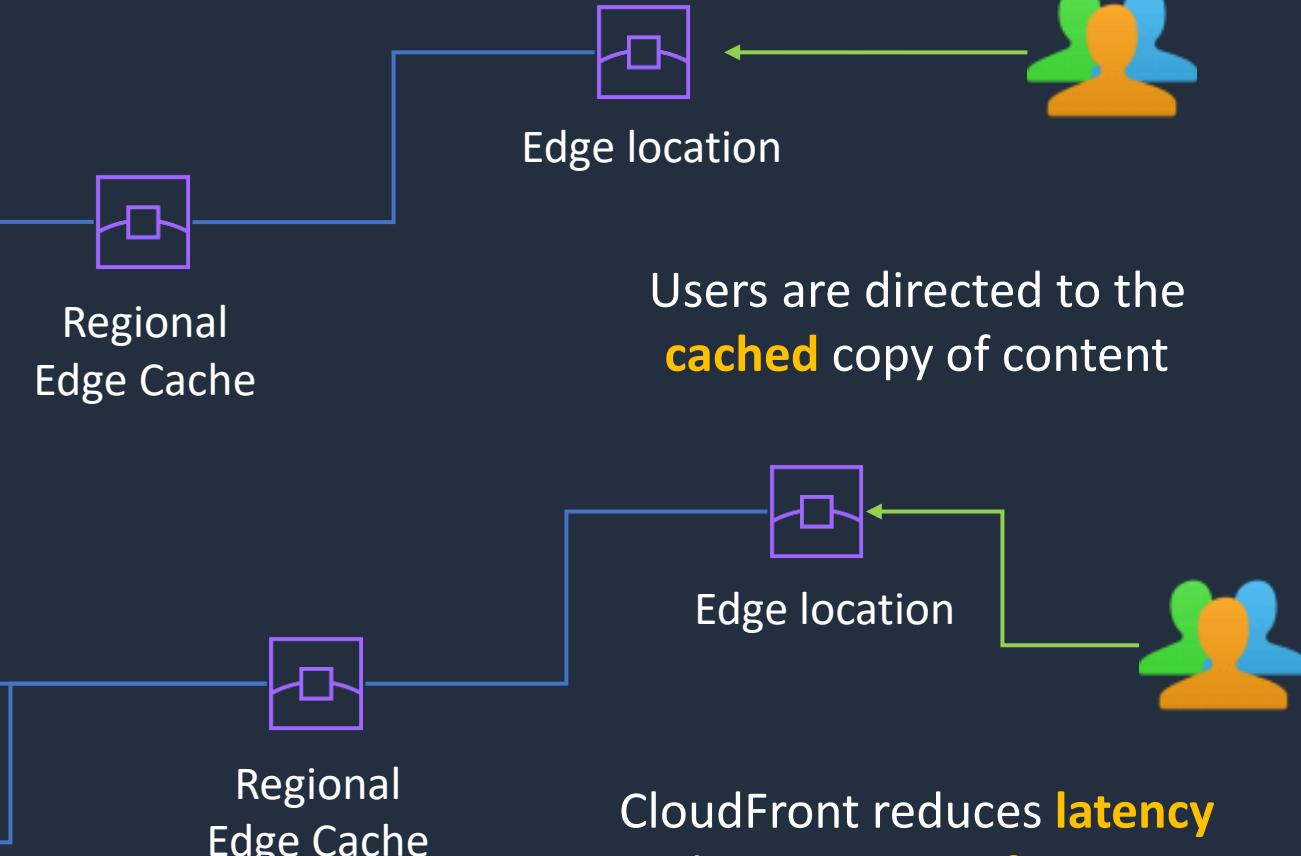
CloudFront Distribution



CloudFront Distribution



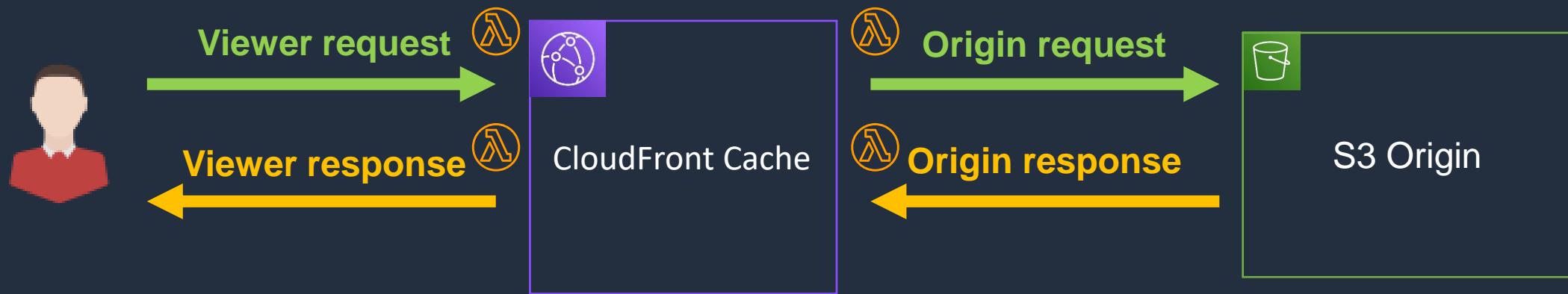
Content from the **origins** gets
cached around the world





Amazon CloudFront

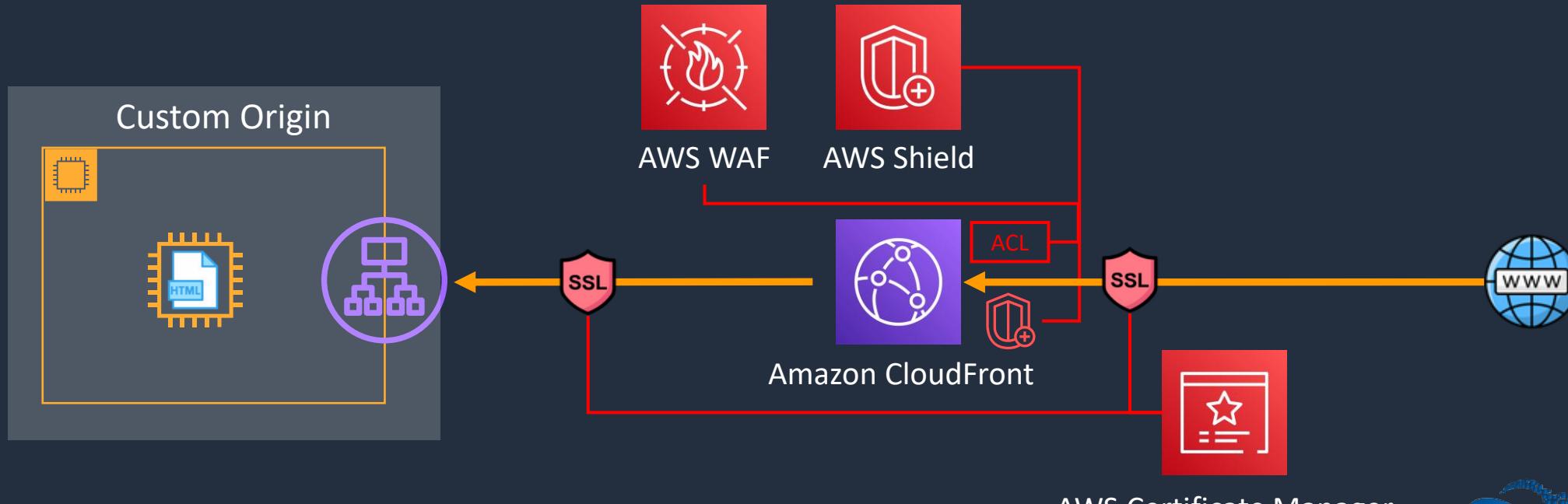
- CloudFront utilizes the AWS Global Network for low latency, high performance connectivity
- Delivers static and dynamic content and optimizes delivery based on content type
- Supports live streaming and video on demand (VOD)
- Lambda@Edge enables processing data with Lambda functions closer to users





Amazon CloudFront

- Uses HTTPS and integrates with AWS ACM for managing SSL/TLS certificates
- Integrates with AWS Shield and AWS WAF for additional security protection
- Content can also be protected with features including signed cookies, signed URLs, and origin access identity (OAI)



Create a Secure CloudFront Distribution

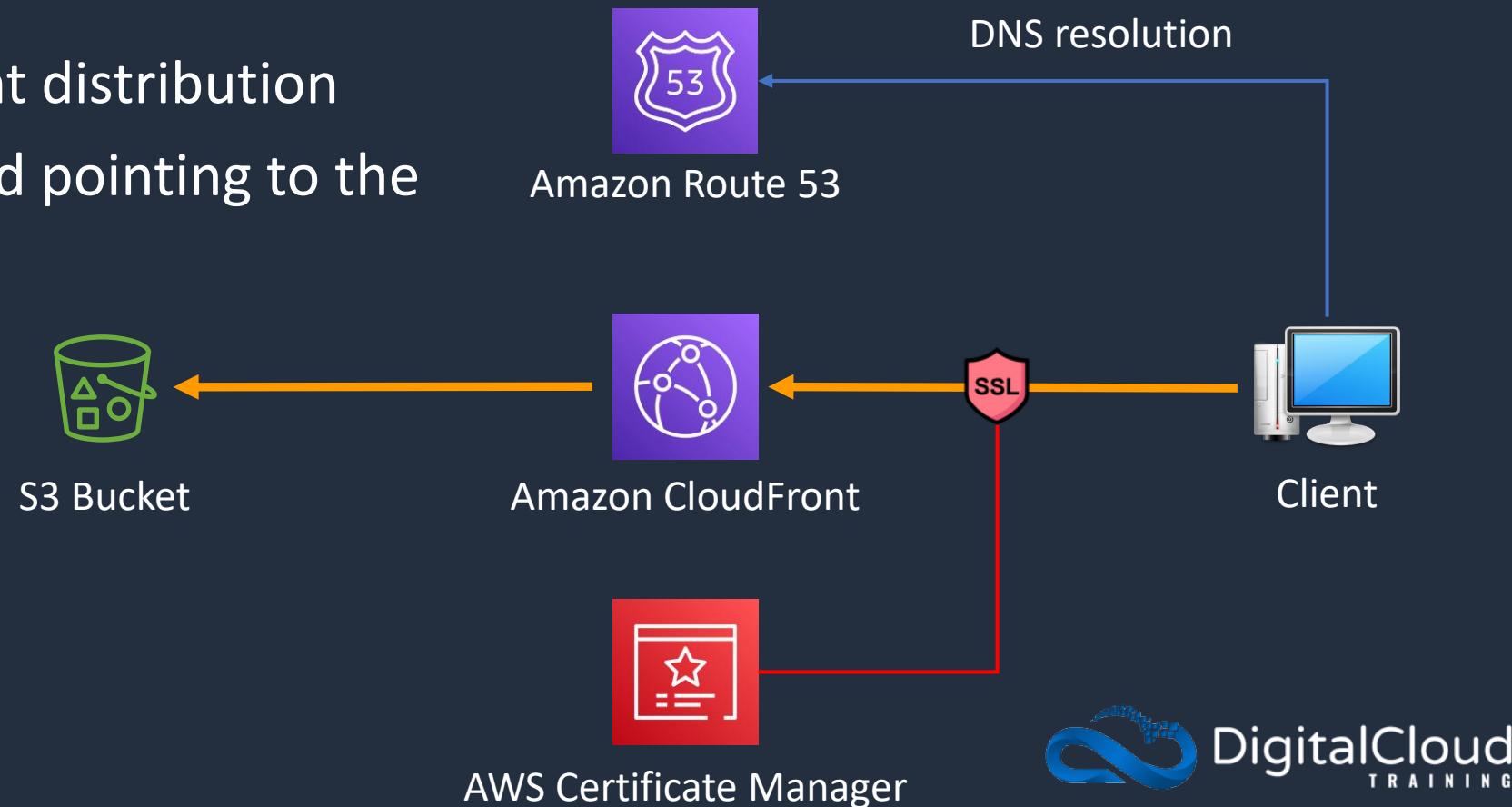




Amazon CloudFront Distribution

We will create:

- An Amazon S3 bucket
- An AWS Certificate Manager SSL/TLS certificate
- An Amazon CloudFront distribution
- A Route 53 alias record pointing to the distribution



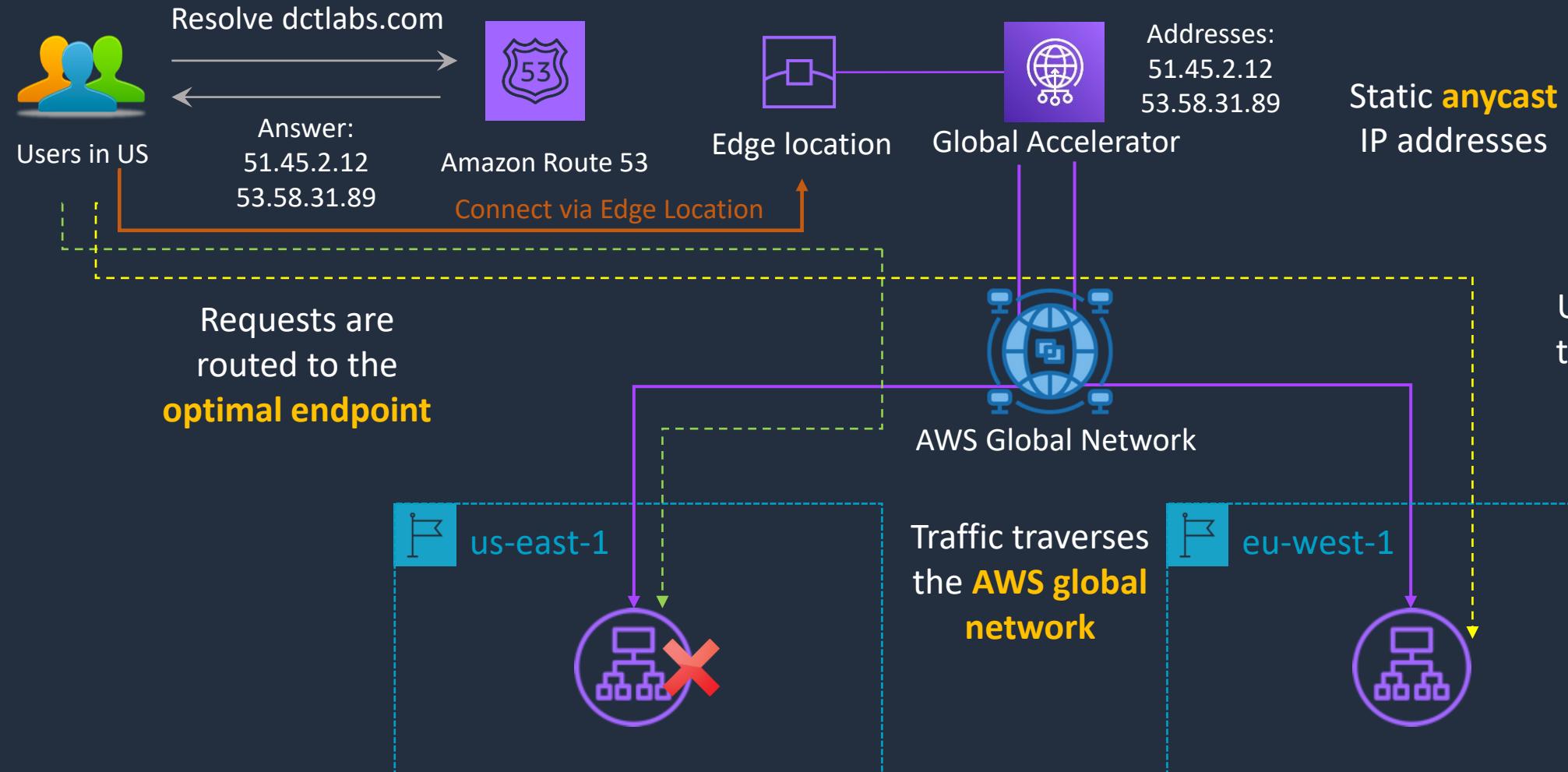
AWS Global Accelerator



AWS Global Accelerator



User traffic ingresses using the closest **Edge Location**





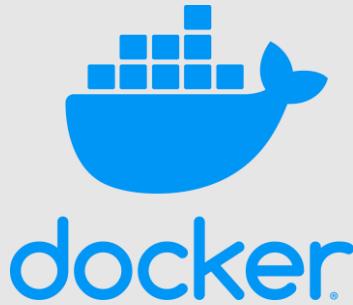
AWS Global Accelerator

- **Network Layer:** Operates at the network layer (Layer 4 of the OSI model)
- **IP Address:** Provides static IP addresses as a fixed entry point to your applications
- **Performance:** Improves performance by leveraging the AWS global network backbone, reducing internet latency and jitter
- **Health Checks:** Performs health checks and automatically reroutes traffic to healthy endpoints
- **Application Protocols:** Supports TCP and UDP traffic, making it suitable for a wide range of applications, including those requiring non-HTTP protocols
- **Use Cases:** Ideal for non-HTTP use cases such as gaming (UDP traffic), IoT, VoIP, or for services where having a static IP address is beneficial

SECTION 10

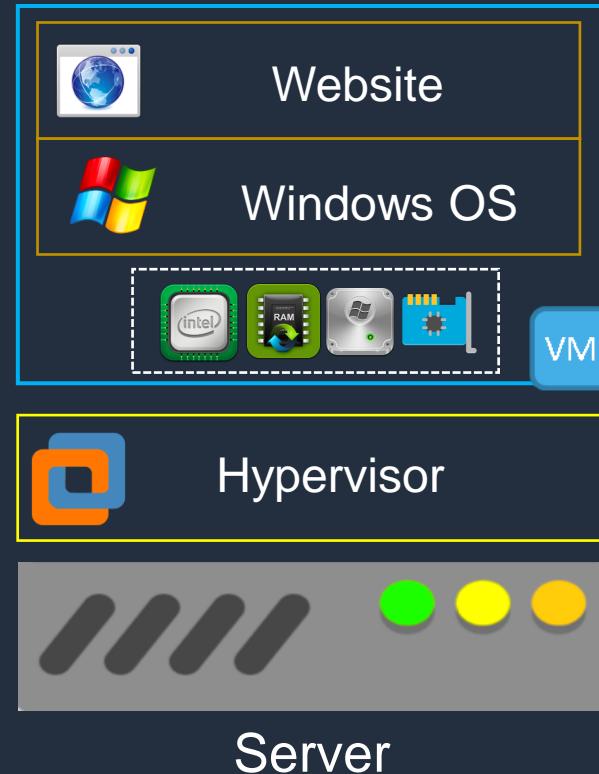
Containers and Serverless Computing

Docker Containers and Microservices



Server Virtualization vs Containers

Every VM-instance needs an **operating system** which uses significant resources





Docker Containers

A **container** includes all the code, settings, and dependencies for running the application

Containers **start up** very **quickly**



Containers are very resource **efficient**

Each container is **isolated** from other containers



Docker Containers

- Docker utilizes containerization to package an application and its dependencies into a single **container image**
- Docker provides **Docker Hub**, a cloud-based registry service for sharing container images and automating workflows
- Containers are lightweight because they share the host system's kernel
- Docker is ideal for **microservices** architectures and building **cloud-native** applications



Cloud-Native Applications

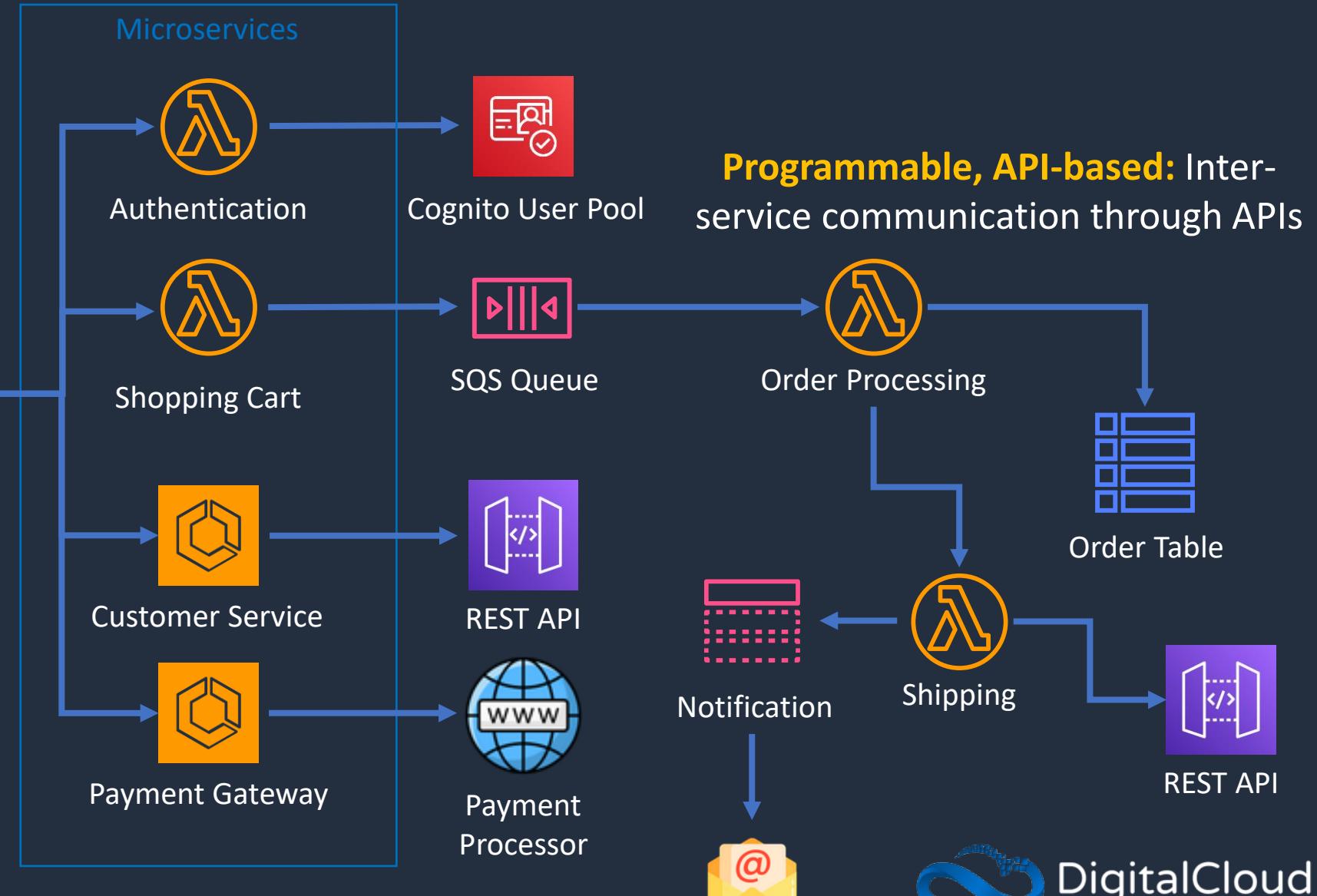
Microservices architecture:

Applications are structured as a collection of **loosely coupled**, independently deployable services, each running its own process



Containers and Functions:

Code runs in Docker containers and Lambda functions for isolation, elasticity, and cost-efficiency





Microservices: Attributes and Benefits

Microservices Attribute	Microservices Benefit
Use of Application Programming Interfaces (APIs)	Easier integrations between application components; assists with loose coupling
Independently deployable blocks of code	Can be scaled and maintained independently
Business-oriented architecture	Development organized around business capabilities; teams may be cross-functional and services may be reused
Flexible use of technologies	Each microservice can be written using different technologies (e.g. programming languages)
Speed and agility	Fast to deploy and update. Easy to include high availability and fault tolerance for each microservice

Amazon Elastic Container Service (ECS)





Amazon ECS

ECS **Services** are used to maintain a **desired count** of tasks

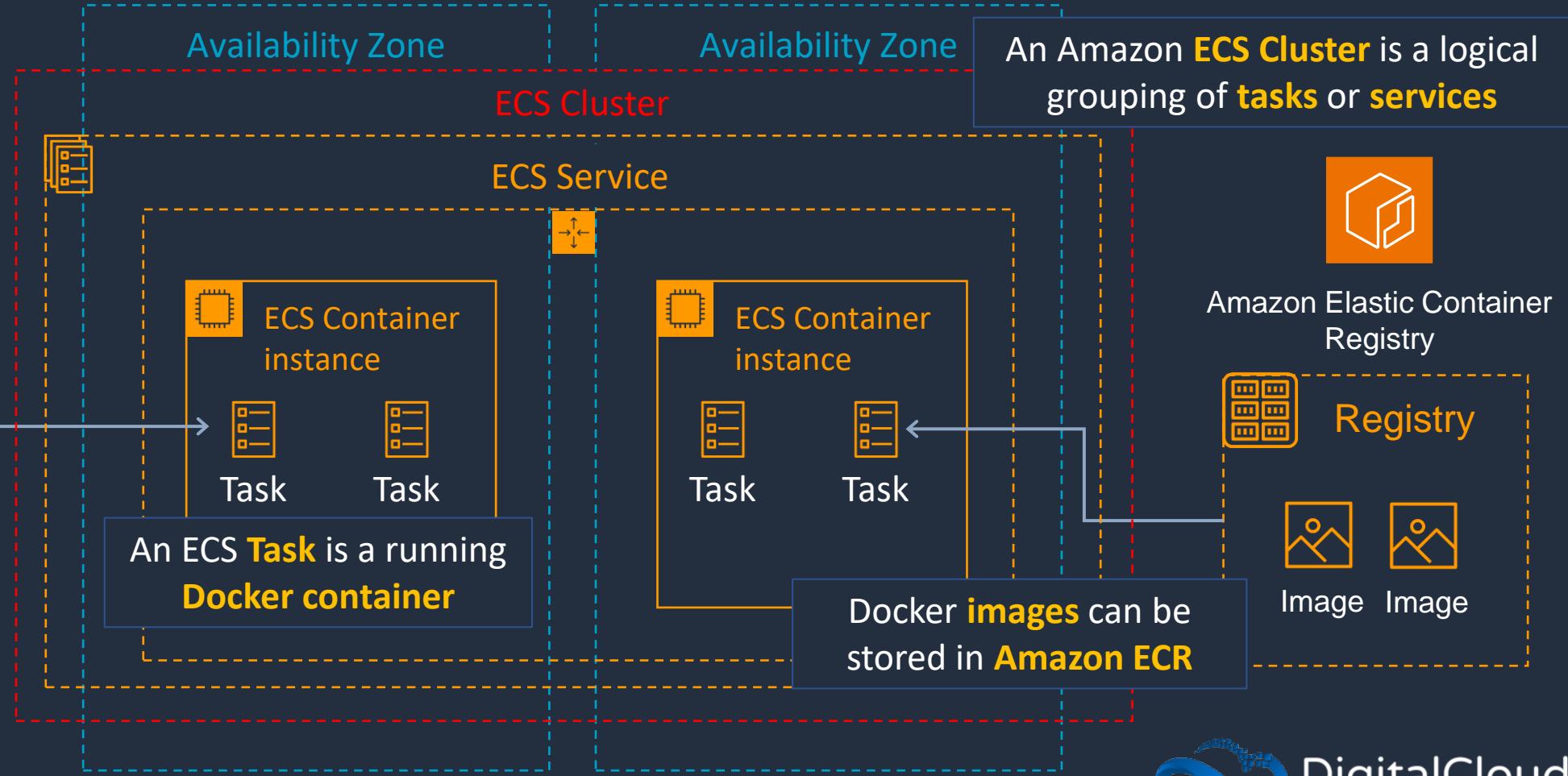
An ECS **Task** is created from a **Task Definition**

Task Definition

```
{
  "containerDefinitions": [
    {
      "name": "wordpress",
      "links": [
        "mysql"
      ],
      "image": "wordpress",
      "essential": true,
      "portMappings": [
        {
          "containerPort": 80,
          "hostPort": 80
        }
      ],
      "memory": 500,
      "cpu": 10
    }
  ]
}
```



Amazon Elastic Container Service





Amazon ECS Key Features

- **Serverless with AWS Fargate** – managed for you and fully scalable
- **Fully managed container orchestration** – control plane is managed for you
- **Docker support** – run and manage Docker containers with integration into the Docker Compose CLI
- **Windows container support** – ECS supports management of Windows containers
- **Elastic Load Balancing integration** – distribute traffic across containers using ALB or NLB
- **Amazon ECS Anywhere** – enables the use of Amazon ECS control plane to manage on-premises implementations



Amazon ECS Components

Elastic Container Service (ECS)	Description
Cluster	Logical grouping of tasks or services
Container instance	EC2 instance running the the ECS agent
Task Definition	Blueprint that describes how a docker container should launch
Task	A running container using settings in a task definition
Image	A Docker image referenced in the task definition
Service	Defines long running tasks – can control task count with Auto Scaling and attach an ELB



Amazon ECS Images

- Containers are created from a read-only template called an **image** which has the instructions for creating a Docker container
- Images are built from a **Dockerfile**
- Only Docker containers are supported on ECS
- Images are stored in a registry such as DockerHub or Amazon Elastic Container Registry (ECR)
- ECR is a managed AWS Docker registry service that is secure, scalable and reliable
- ECR supports private Docker repositories with resource-based permissions using AWS IAM in order to access repositories and images
- You can use the Docker CLI to push, pull and manage images



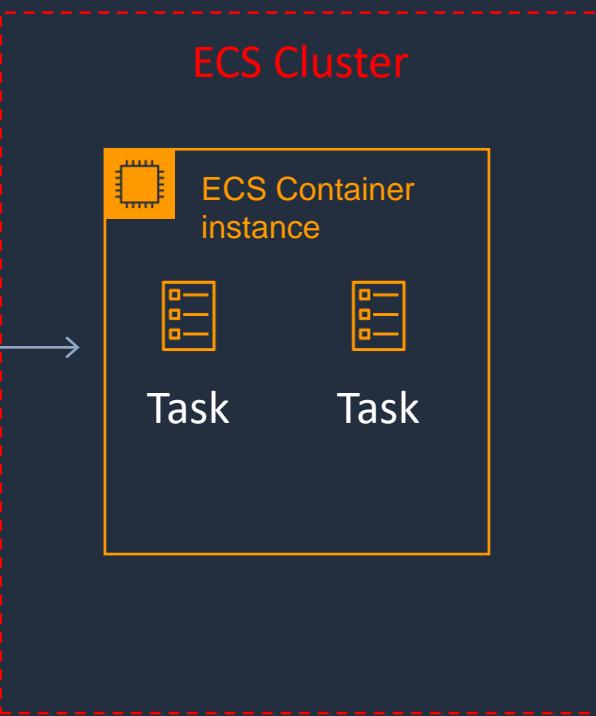


Amazon ECS Tasks and Task Definitions

- A task definition is required to run Docker containers in Amazon ECS
- A task definition is a text file in JSON format that describes one or more containers, up to a maximum of 10
- Task definitions use Docker images to launch containers

Task Definition

```
{  
  "containerDefinitions": [  
    {  
      "name": "wordpress",  
      "links": [  
        "mysql"  
      ],  
      "image": "wordpress",  
      "essential": true,  
      "portMappings": [  
        {  
          "containerPort": 80,  
          "hostPort": 80  
        }  
      ],  
      "memory": 500,  
      "cpu": 10  
    }  
  ]  
}
```

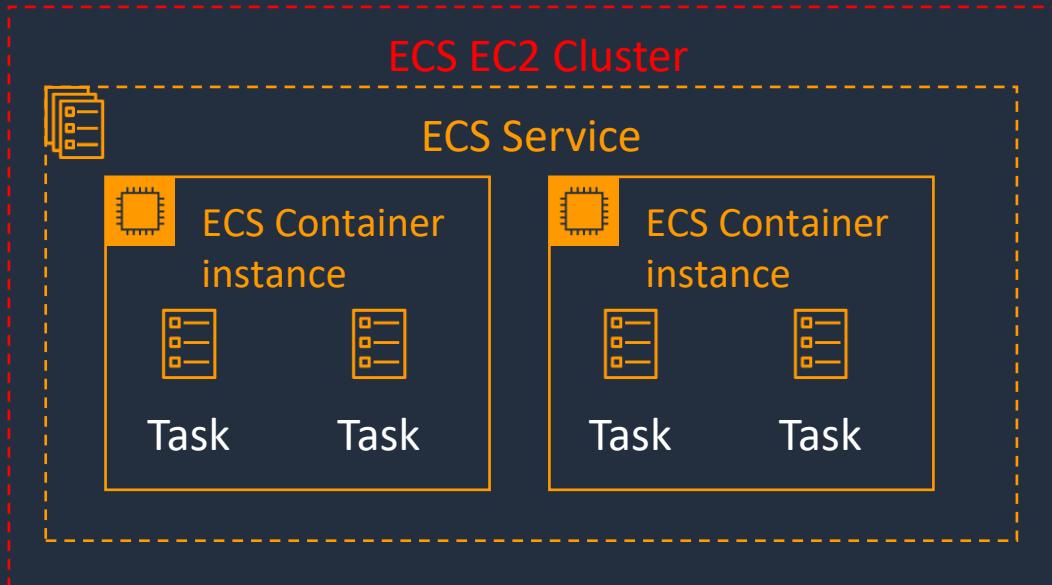




Launch Types – EC2 and Fargate



Registry:
ECR, Docker Hub, Self-hosted

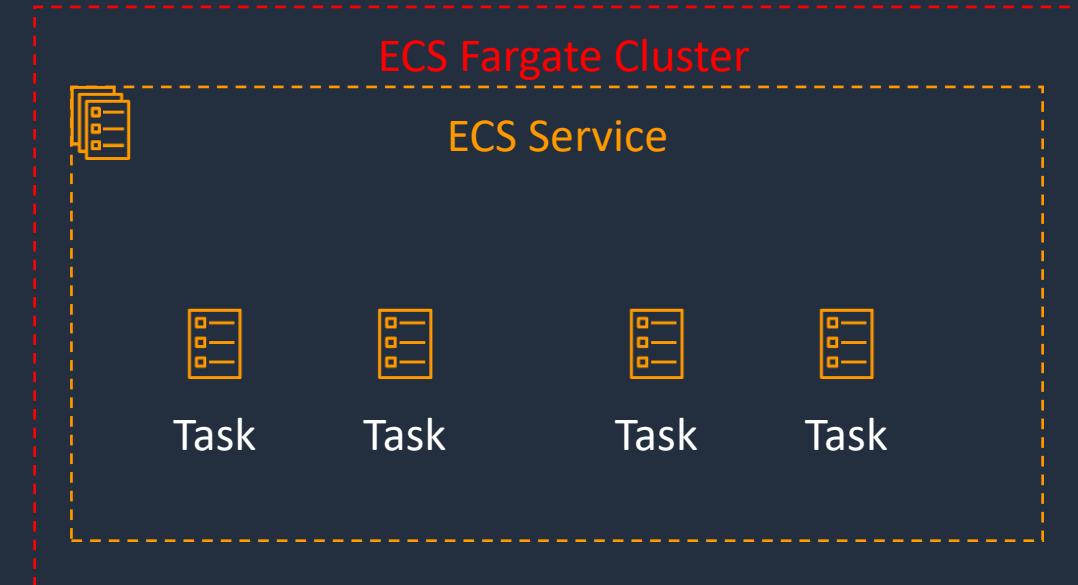


EC2 Launch Type:

- You explicitly provision EC2 instances
- You're responsible for managing EC2 instances
- Charged per running EC2 instance
- EFS, FSx, and EBS integration
- You handle cluster optimization
- More granular control over infrastructure



Registry:
ECR, Docker Hub

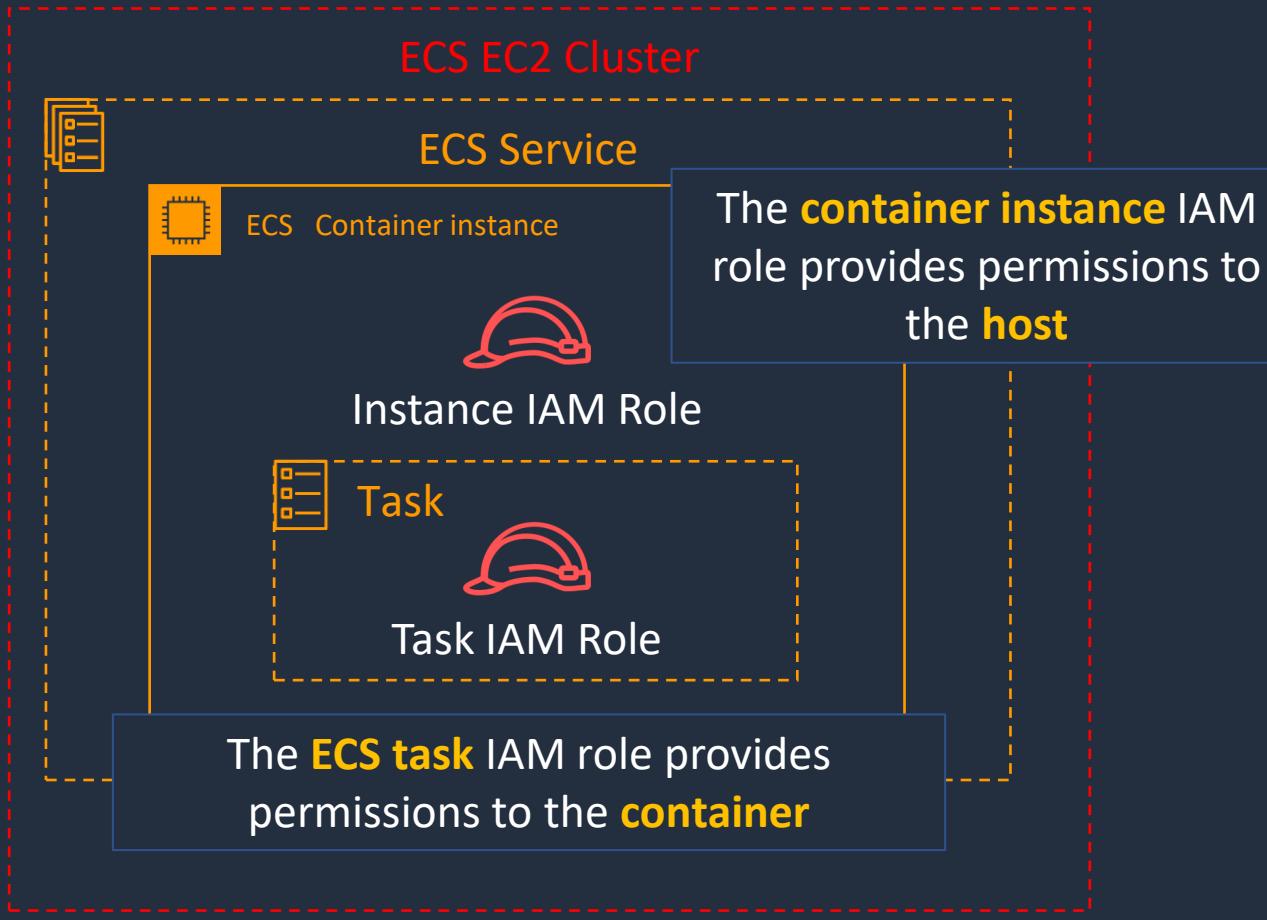


Fargate Launch Type:

- Fargate automatically provisions resources
- Fargate provisions and manages compute
- Charged for running tasks
- EFS integration only
- Fargate handles cluster optimization
- Limited control



ECS and IAM Roles



NOTE: With the Fargate launch type the container instance role is replaced with the **Task Execution Role**

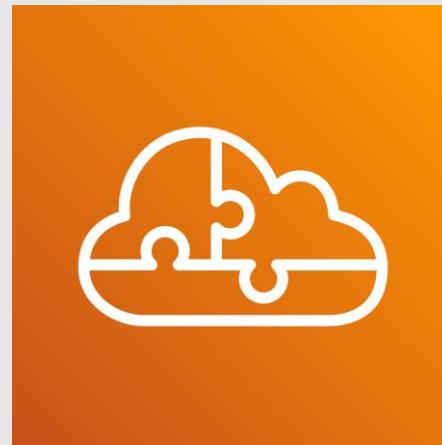
AmazonEC2ContainerServiceforEC2Role

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Action": [  
        "ec2:DescribeTags",  
        "ecs>CreateCluster",  
        "ecs>DeregisterContainerInstance",  
        "ecs>DiscoverPollEndpoint",  
        "ecs>Poll",  
        "ecs>RegisterContainerInstance",  
        "ecs>StartTelemetrySession",  
        "ecs>UpdateContainerInstancesState",  
        "ecs>Submit*",  
        "ecr>GetAuthorizationToken",  
        "ecr>BatchCheckLayerAvailability",  
        "ecr>GetDownloadUrlForLayer",  
        "ecr>BatchGetImage",  
        "logs>CreateLogStream",  
        "logs>PutLogEvents"  
      ],  
      "Resource": "*"  
    }  
  ]  
}
```

Launch Docker Containers on AWS Fargate



Serverless Services and Event-Driven Architecture



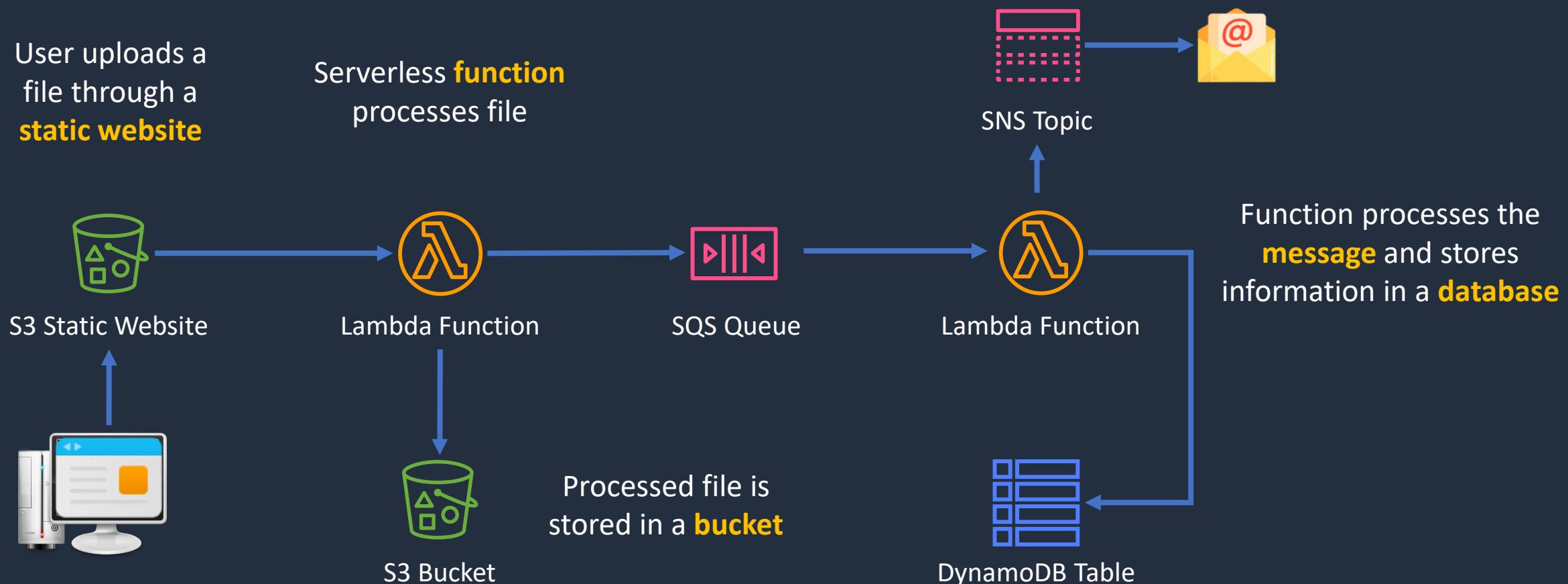


Serverless Services

- With serverless there are **no instances** to manage
- You don't need to provision hardware
- There is no management of operating systems or software
- Capacity provisioning and patching is handled automatically
- Provides automatic scaling and high availability
- Can be very cheap!



Serverless Services and Event-Driven Architecture



AWS Lambda





AWS Lambda

Pricing is based on memory assigned and the duration of function execution

Functions are invoked based on **events** and the code is then executed

```
import json

def lambda_handler(event, context):
    # Print the event to CloudWatch Logs
    print("Received event: " + json.dumps(event))

    # Return a response
    return {
        'statusCode': 200,
        'body': json.dumps('Event logged successfully!')
    }
```

Code is executed





AWS Lambda

- **Languages** – Lambda natively supports Java, Go, PowerShell, Node.js, C#, Python, and Ruby code
- **Execution Role (IAM Role)** – this role grants the function permissions to access AWS services and resources
- **Monitoring and Logging** – integrates with Amazon CloudWatch
- **Memory and Timeout** – you can specify the amount of memory allocated to a function and the maximum execution time
- **Note:** The maximum execution time is 15 minutes



Lambda Function Invocation

- Lambda functions run in response to events from various AWS services or direct invocation from the AWS SDKs or API
- Functions can be invoked synchronously or asynchronously:
 - With **synchronous** invocation, applications wait for the function to process the event and return a response
 - With **asynchronous** invocation, Lambda queues the event for processing and returns a response immediately
- Lambda scales horizontally by running multiple instances of a function in parallel, up to the concurrency limit





Connecting Lambda to an Amazon VPC

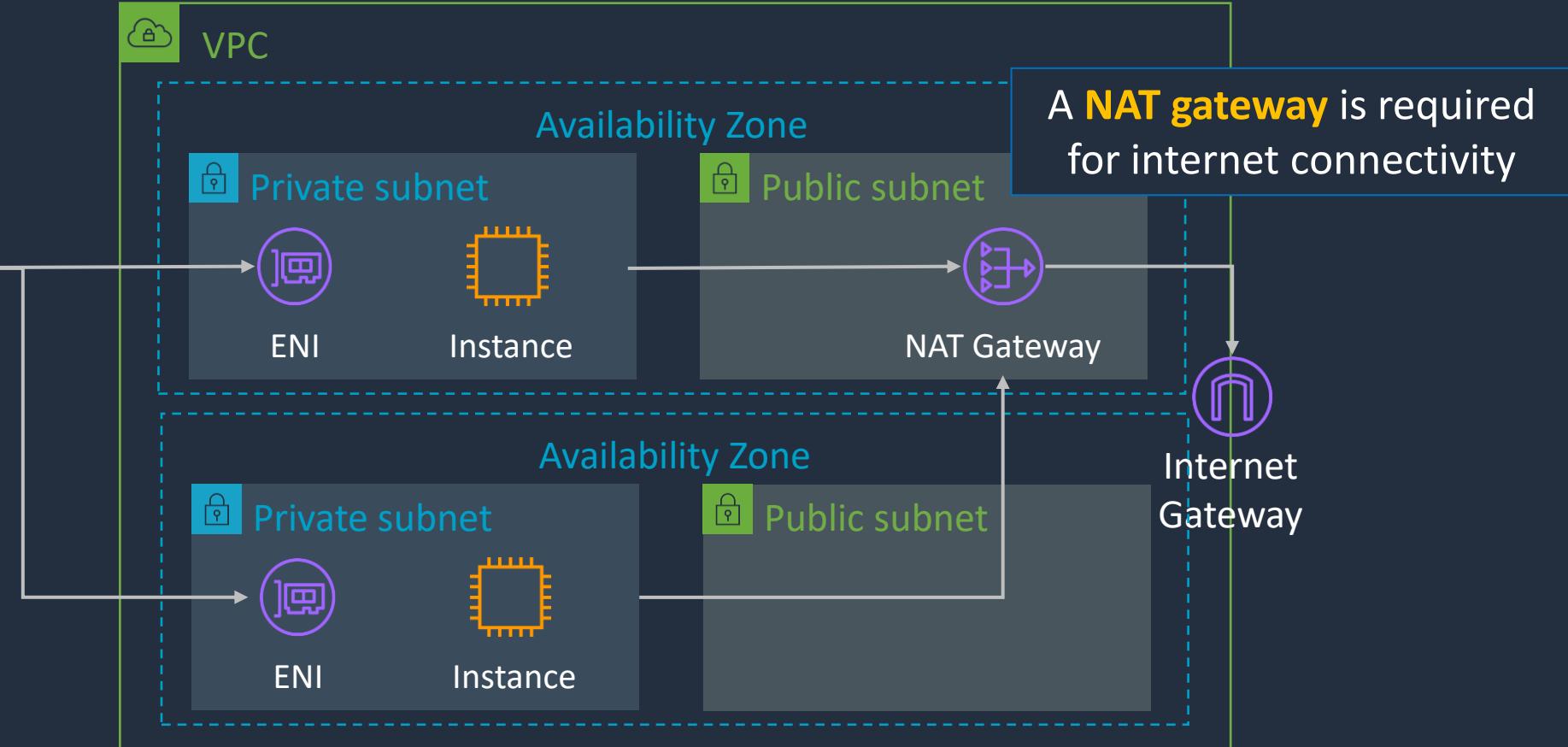
Must select the VPC, subnets,
and security group

Lambda functions are
Regional and do not have
VPC access by default



Lambda Function

Lambda functions can be
connected to a VPC



The function **execution role** must have
permissions create the ENIs

Create an AWS Lambda Function

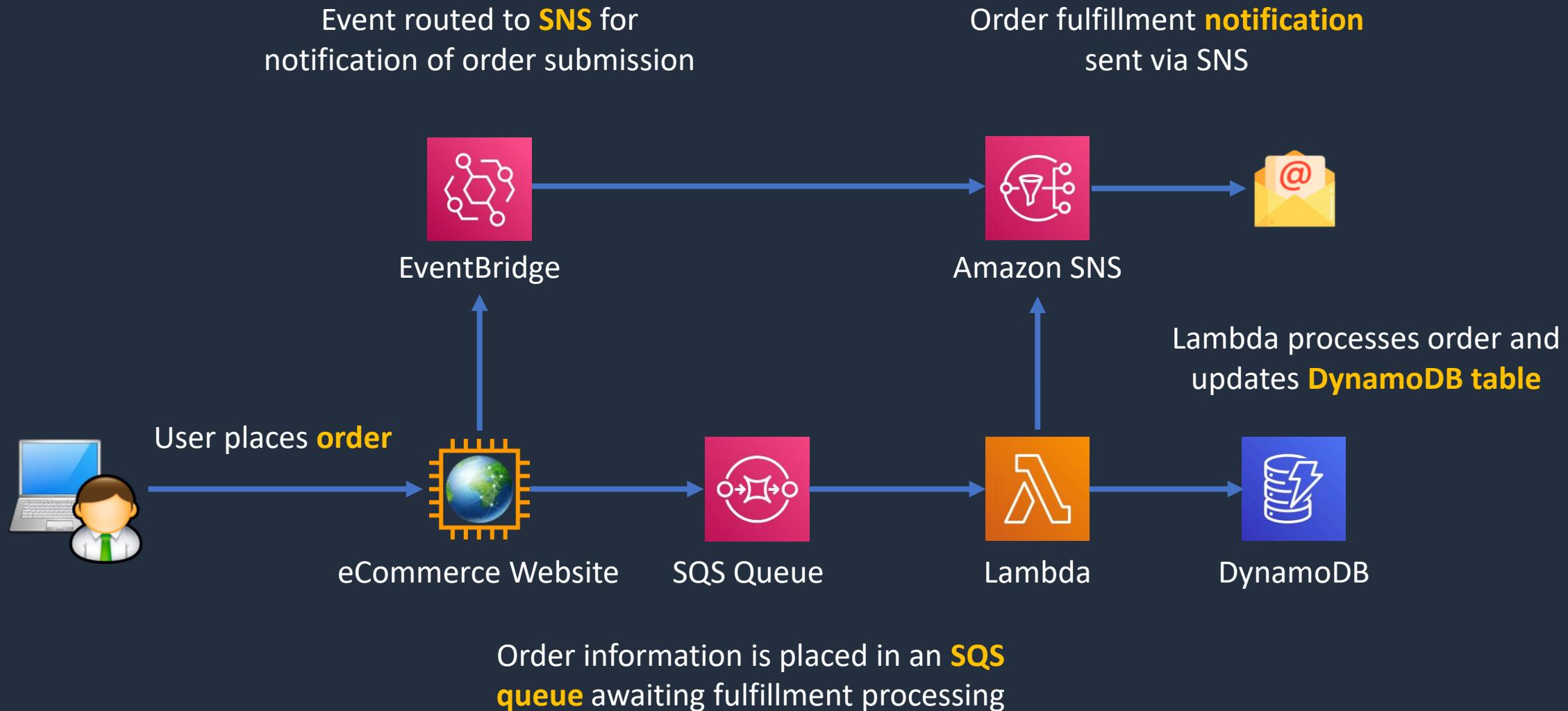


Application Integration Services



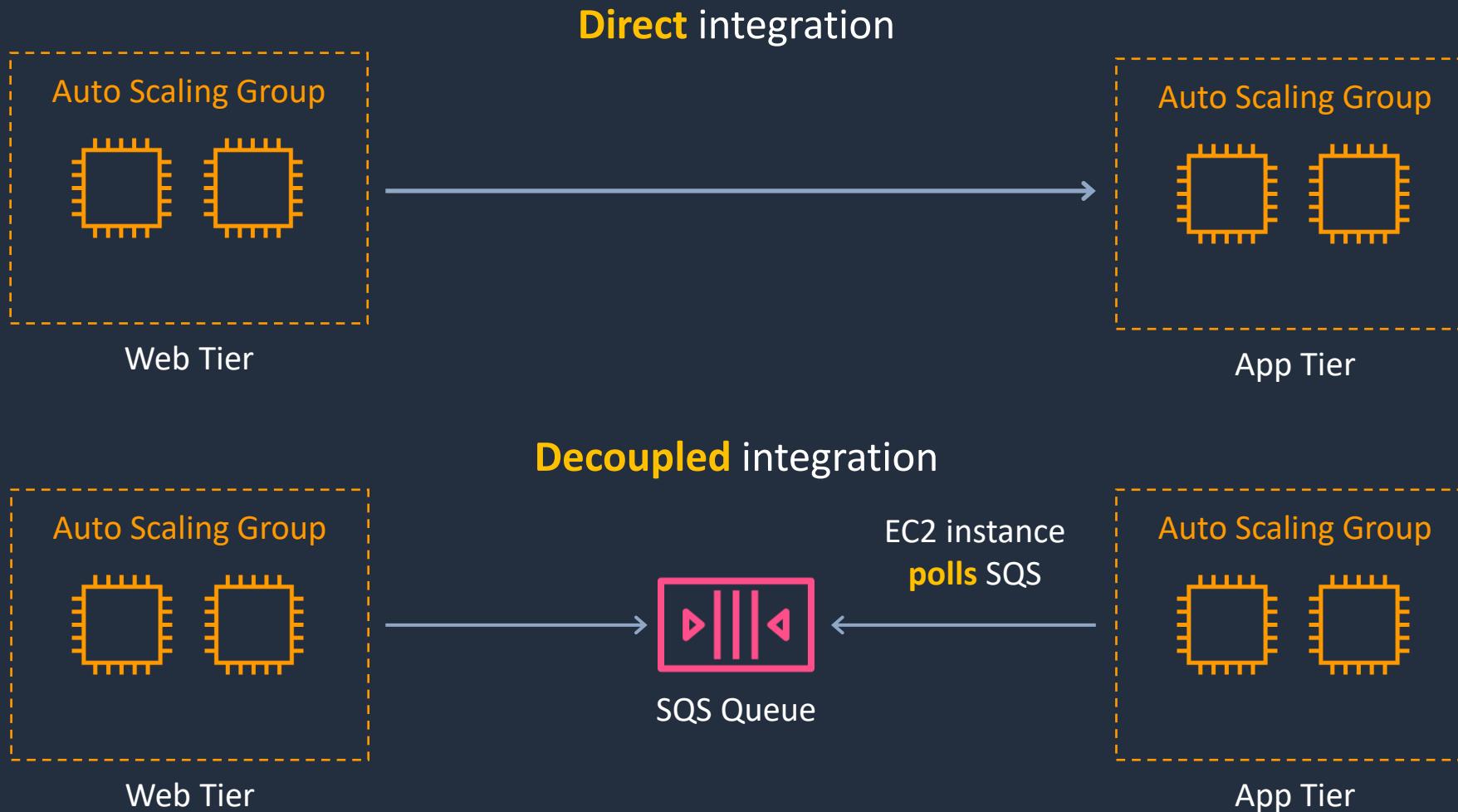


Event-Driven Architecture



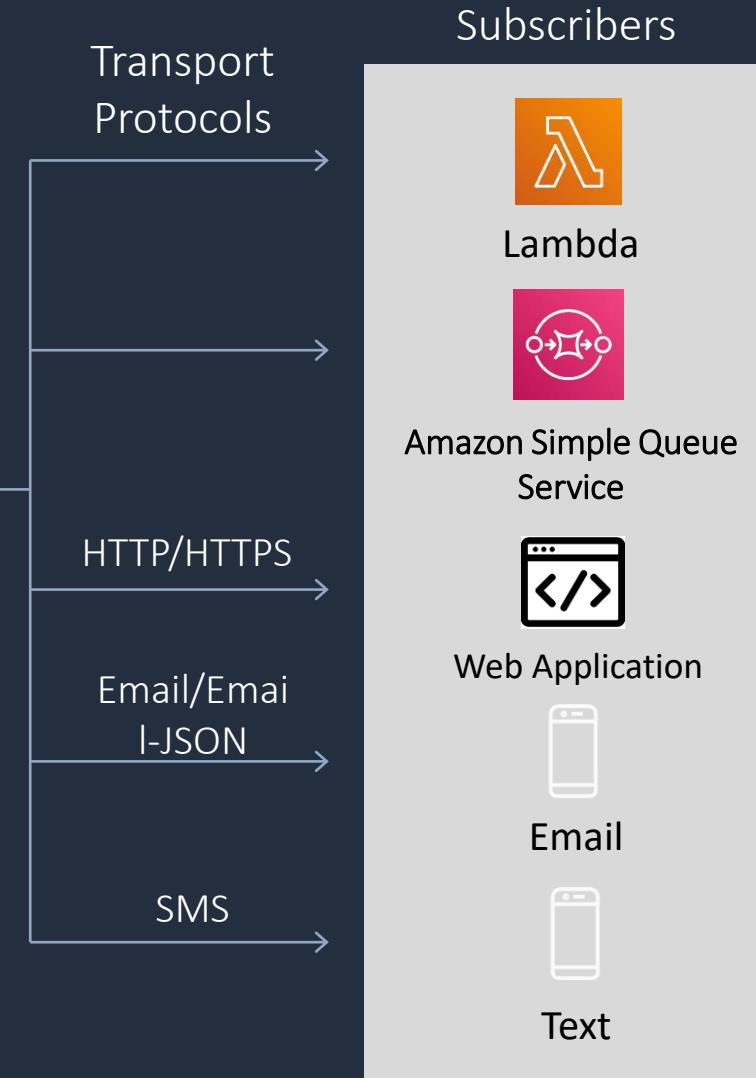
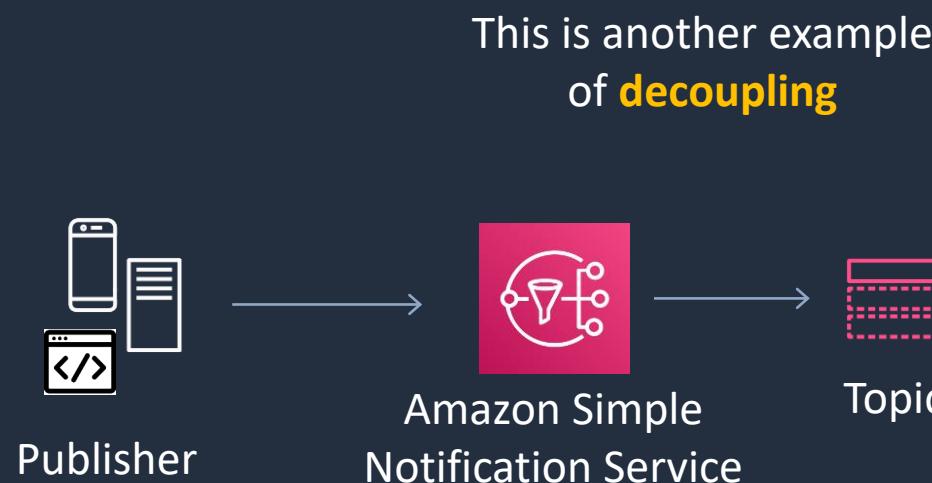


Amazon Simple Queue Service (SQS)





Amazon Simple Notification Service (SNS)

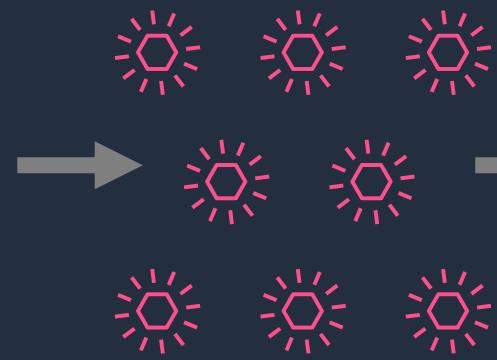
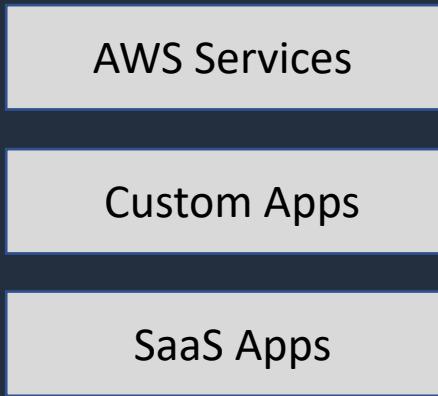




Amazon EventBridge

Amazon EventBridge is a serverless event bus that connects
loosely coupled application components together

Event Sources



Events

EventBridge
event bus



Rules



Targets



Application Integration Services Comparison

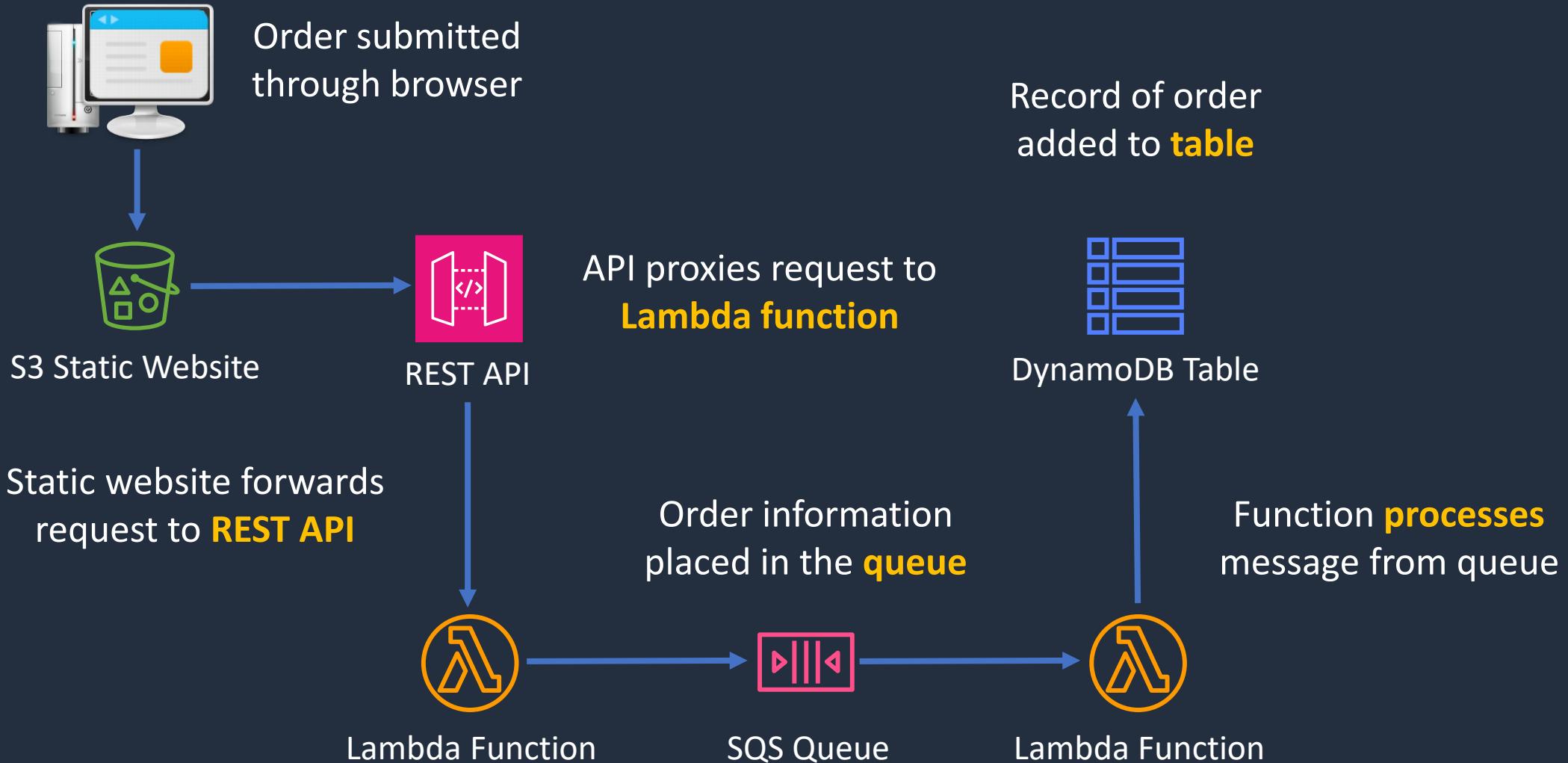
Service	What it does	Example use cases
Simple Queue Service	Messaging queue; store and forward patterns	Building distributed / decoupled applications
Simple Notification Service	Set up, operate, and send notifications from the cloud	Send email notification when CloudWatch alarm is triggered
Step Functions	Coordination of AWS services with visual workflow	Order processing workflows
Amazon MQ	Message broker service for Apache Active MQ and RabbitMQ	Need a message queue that supports industry standard APIs and protocols
Amazon EventBridge	Serverless event bus for connecting applications and AWS services	Create event driven applications

Serverless Application with REST API – Part 1





Serverless Application with REST API

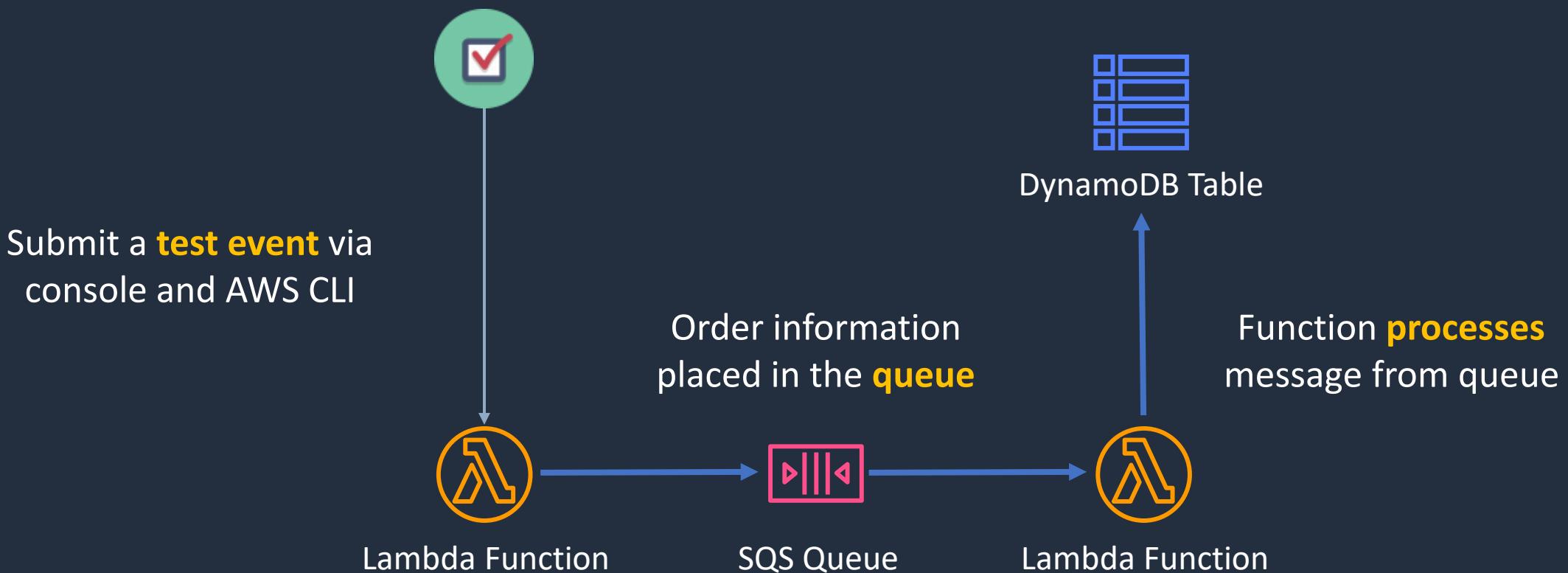




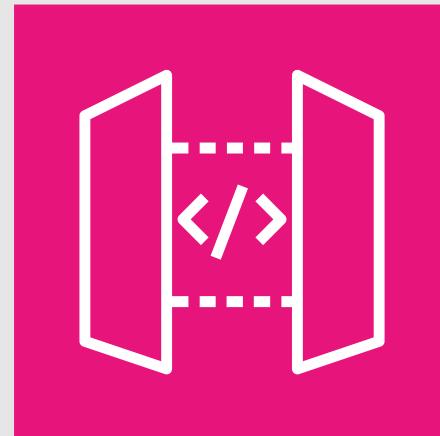
Serverless Application with REST API

```
{  
  "body": "{\"productName\":\"Test Product\", \"quantity\":3}"}  
}
```

Record of order
added to **table**

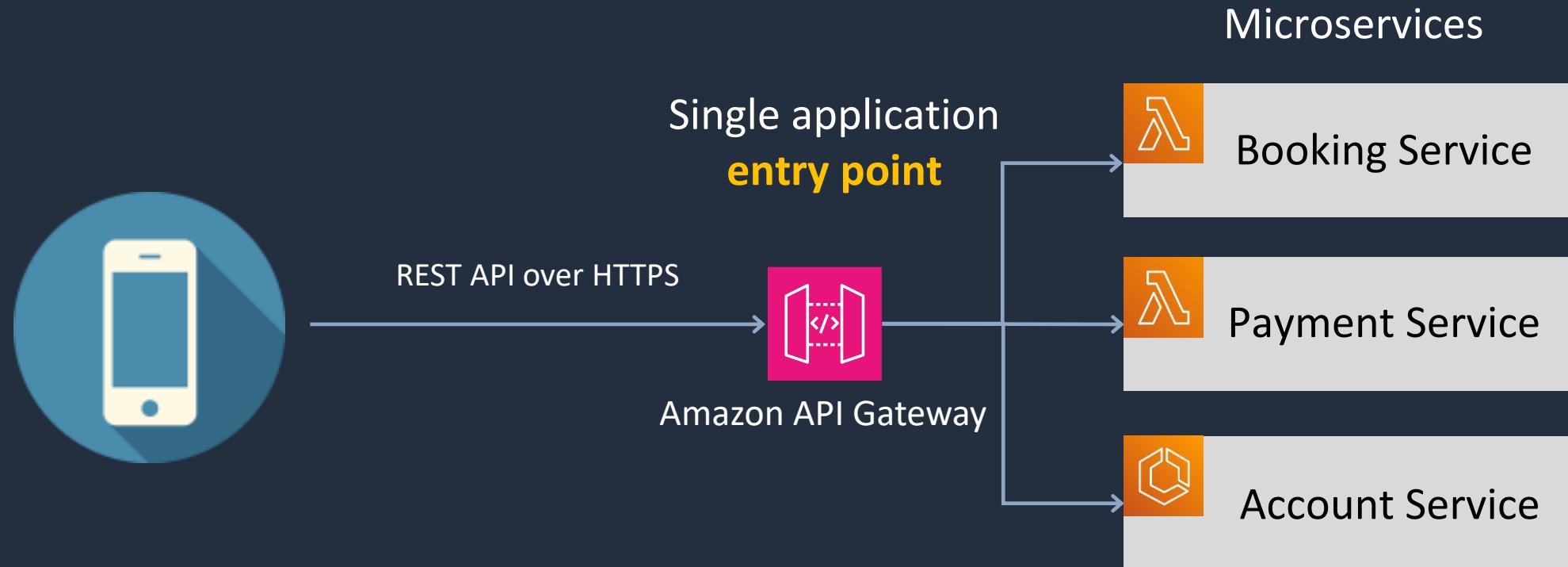


Amazon API Gateway





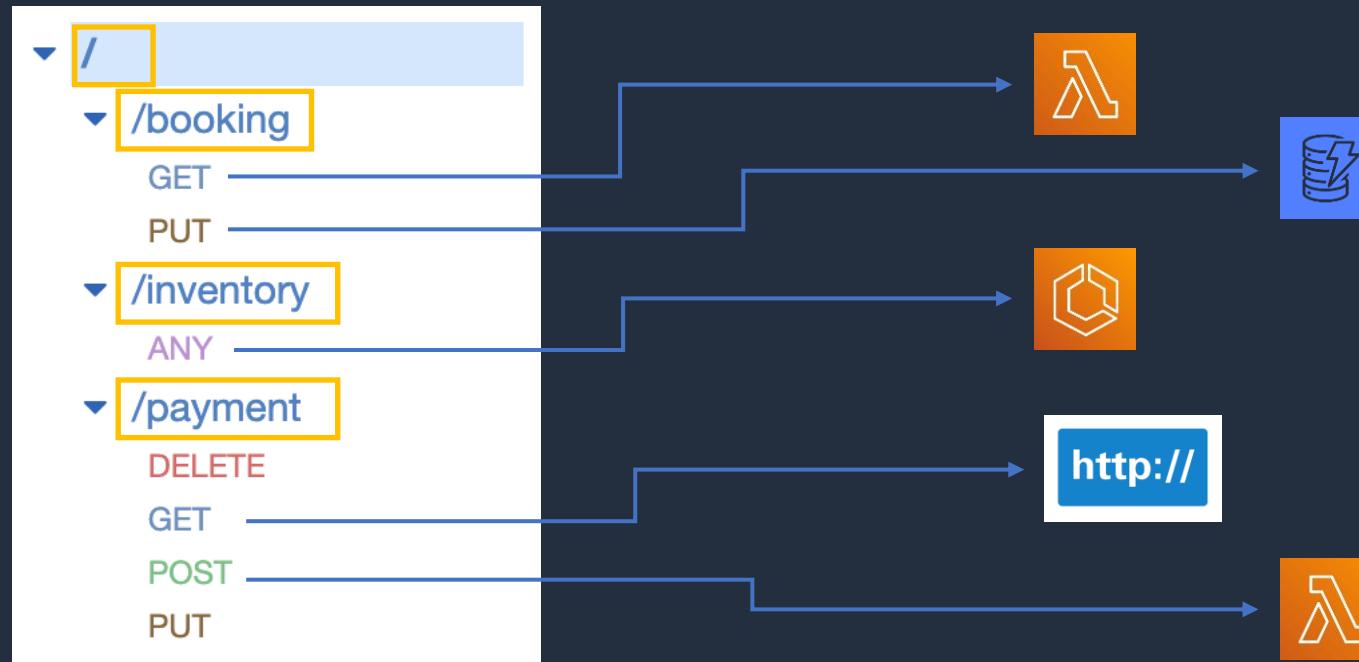
REST API with Amazon API Gateway





REST API with Amazon API Gateway

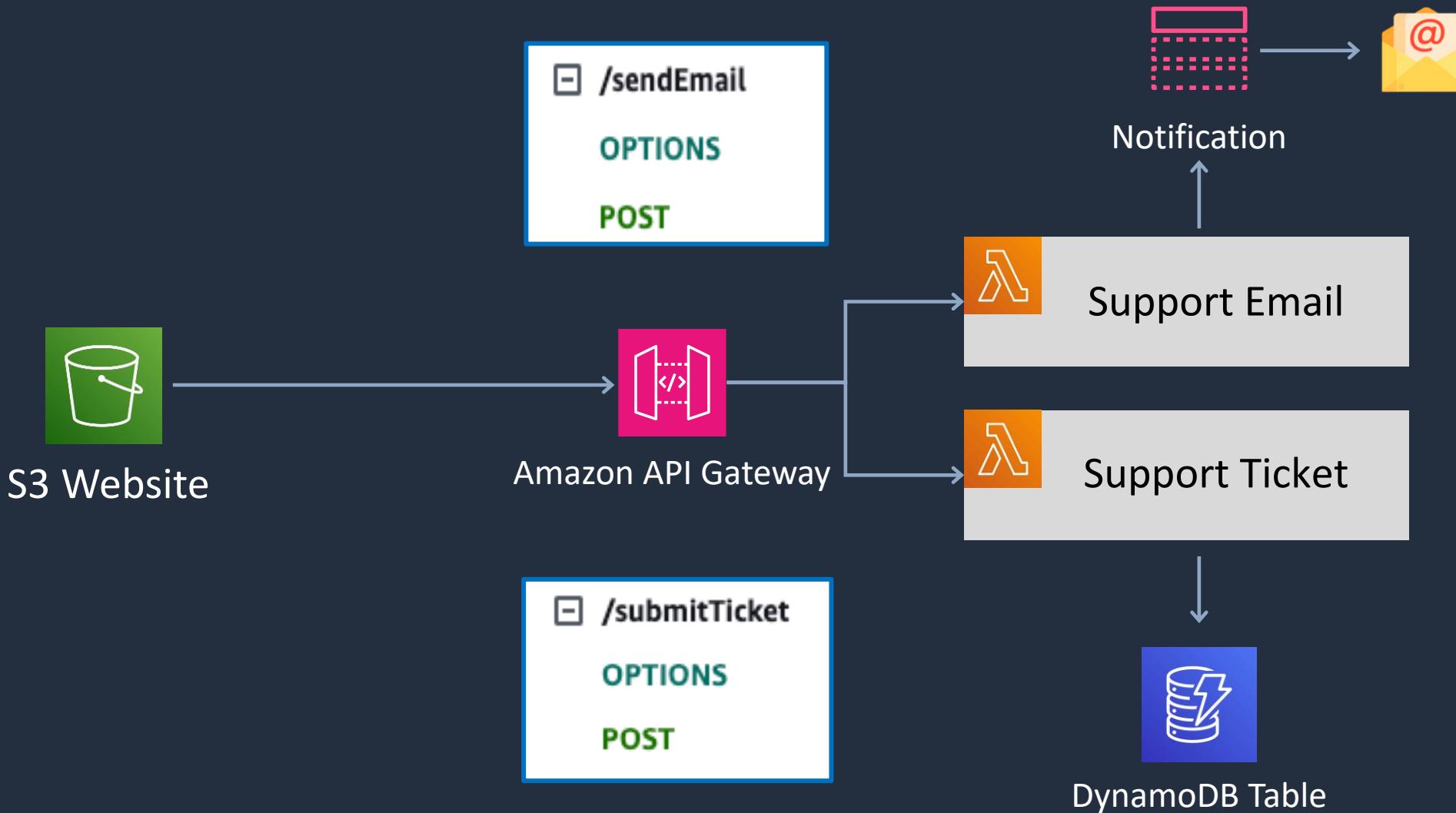
A **resource** represents a path in your API



Methods are created within resources and represent client-facing interfaces by which the client calls the API to access back-end resources



REST API with Amazon API Gateway



Serverless Application with REST API – Part 2

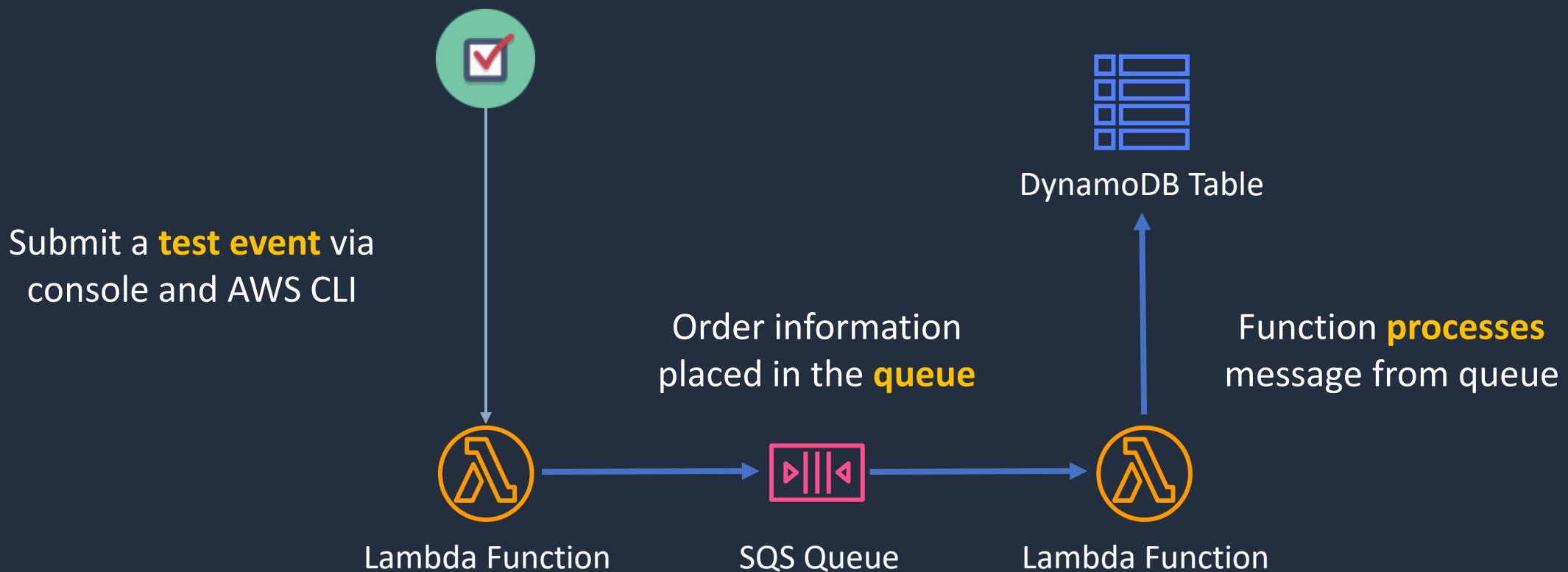




Serverless Application with REST API

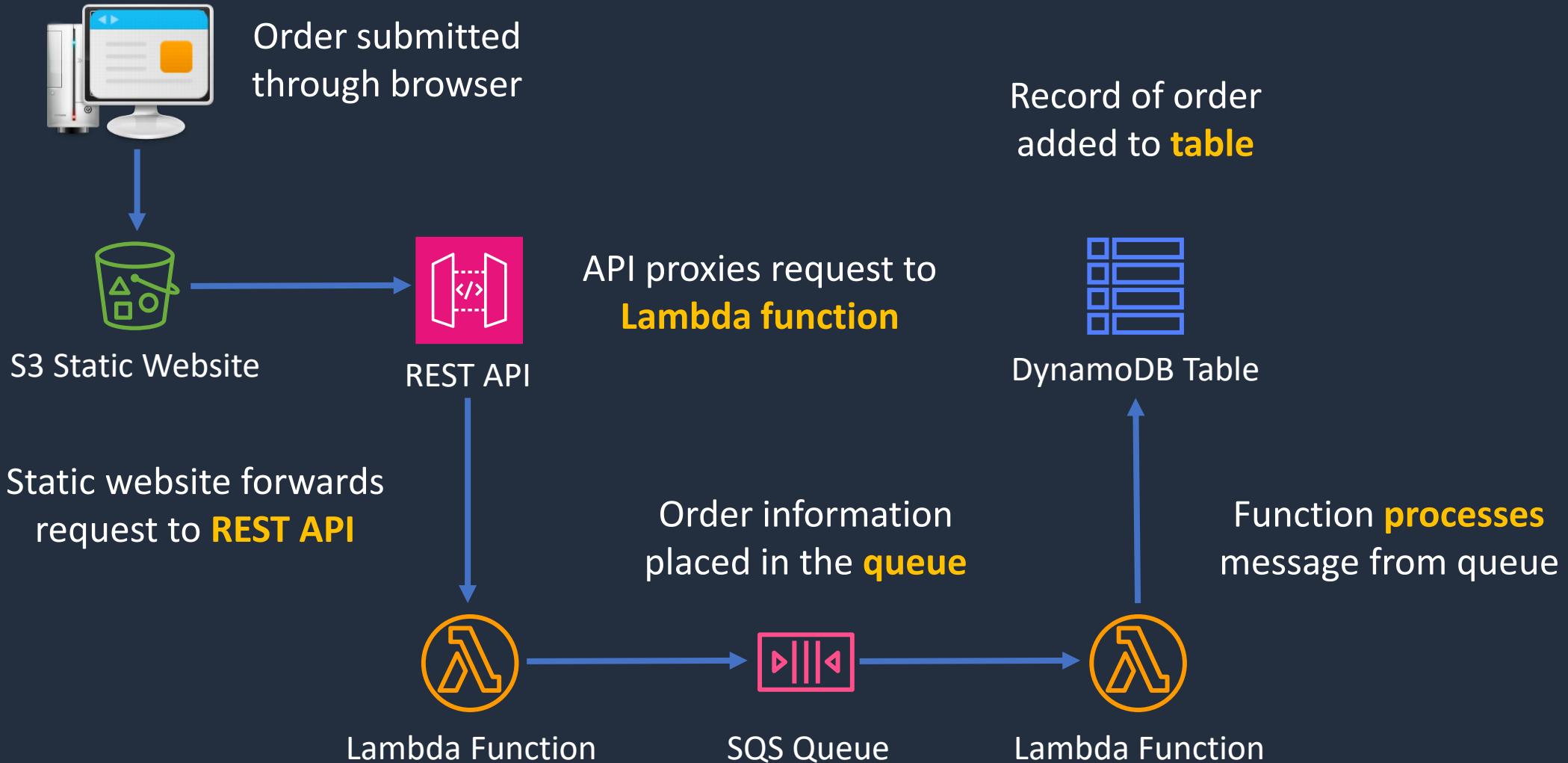
```
{  
  "body": "{\"productName\":\"Test Product\", \"quantity\":3}"}  
}
```

Record of order
added to **table**





Serverless Application with REST API



Amazon EventBridge





Amazon EventBridge

Amazon EventBridge is a serverless **event bus** for building **event-driven** applications

Event Sources

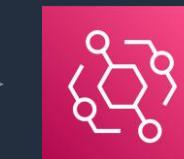
AWS Services

Custom Apps

SaaS Apps

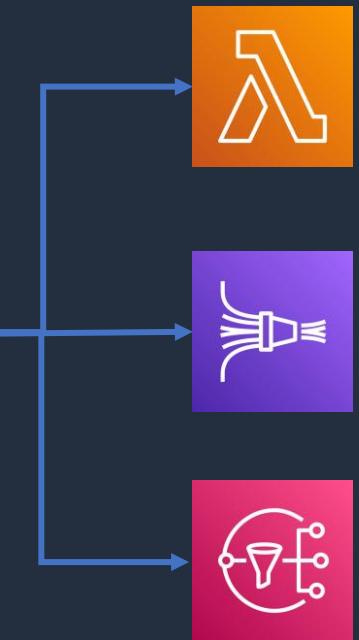


Events



EventBridge
event bus

Rules



Targets

EventBridge ingests, filters, transforms, and delivers events to build **loosely-coupled** applications

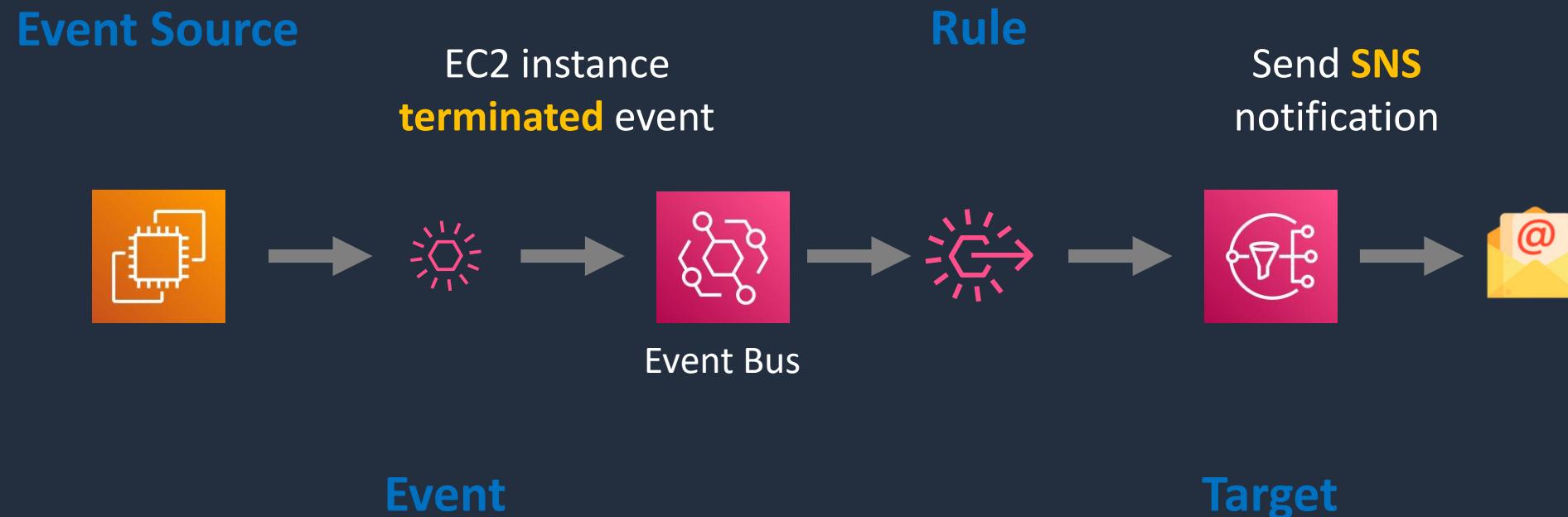


Amazon EventBridge

- **Event Sources** – supports events from AWS services, SaaS applications, and custom applications
- **Default and Custom Event Buses** – Offers a default bus for AWS events and supports creating custom buses for personal or third-party app events
- **Event Filtering and Routing** – Events can be filtered and routed to different targets based on content-based filtering rules
- **Scalability and Reliability** – scales automatically with the number of events, handling millions of events per second



Amazon EventBridge with Amazon EC2



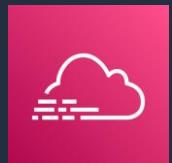


Amazon EventBridge with AWS CloudTrail

Event Source

S3:PutBucketPolicy

API used



Rule

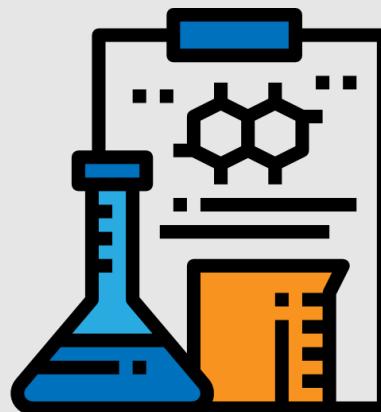
Secure S3 bucket

Event

Target

```
{  
  "source": ["aws.cloudtrail"],  
  "detail-type": ["AWS API Call via CloudTrail"],  
  "detail": {  
    "eventSource": ["cloudtrail.amazonaws.com"],  
    "eventName": ["s3:PutBucketPolicy"]  
  }  
}
```

Create Event Bus and Rule



SECTION 11

Machine Learning and AI

Using Generative AI as a Learner





ChatGPT Overview

- ChatGPT is an AI language model developed by OpenAI that can understand and generate human-like text
- It can converse, answer questions, and provide information on a wide range of topics
- **Plans:**
 - **Free:** access to GPT-3.5 model only
 - **Plus:** \$20/month with access to GPT-4 model (MUCH better)
 - **Team:** \$30/month, minimum 2 users, team data excluded from training
- Can install plugins for a variety of functions
- Can create, browse, create, and use GPTs (Plus/Team only)



Benefits:

- Immediate assistance with questions
- Context aware so you can have a conversation
- Extremely broad training data

Concerns:

- Check when the model was trained, facts could be out of date
- Sometimes it's wrong and sometimes it "hallucinates"



Prompt Engineering Tips

- **Clarity and Specificity:** The prompt should be clear and specific about what is being asked or the kind of response that is desired
- **Contextual Information:** Including relevant context within the prompt can significantly improve the quality of the response. This might involve providing background information, setting the scene, or explaining the purpose of the query
- **Instructional Design:** Sometimes, prompts are crafted to instruct the AI in a specific way of responding. For example, ask the AI to answer in the form of a list, a detailed explanation, a brief summary, or even in a particular style or tone
- **Iterative Refinement:** Prompt engineering often involves an iterative process, where prompts are continually refined based on the responses they elicit



Code Generation

- **Be Specific and Detailed:** Clearly describe the functionality you want to implement. Include specific details about the AWS services involved
- **Define the Scope:** Indicate the scope of the code you need. Do you want a snippet, a function, a full script, or an entire application architecture?
- **Mention the Programming Language and Tools:** Specify the programming language (e.g., Python, JavaScript, Java) and any specific libraries or SDKs (like Boto3 for Python, AWS SDK for JavaScript)
- **State AWS Services Clearly:** Clearly mention which AWS services are involved (e.g., Amazon S3, AWS Lambda, Amazon DynamoDB)
- **Security and Best Practices:** If you have specific security practices or best practices you need to adhere to, include these in your prompt
- **Error Handling and Logging:** Indicate if you need error handling and logging mechanisms in your code
- **Testing and Validation:** Always test and validate the generated code!



Use Cases for Learners

1. Use ChatGPT to get answers to questions
2. Use ChatGPT to come up with project ideas
3. Get ChatGPT to create the code and instructions

Creating Projects and Code with ChatGPT



AWS Machine Learning and AI Services





AWS Rekognition

Identify objects



Perform facial analysis



Celebrity recognition





AWS Rekognition in Event-Driven Architecture



Rekognition publishes result status to an SNS Topic



AWS Rekognition

- Add image and video analysis to your applications
- Identify objects, people, text, scenes, and activities in images and videos
- Processes videos stored in an Amazon S3 bucket
- Publish completion status to Amazon SNS Topic



Amazon Transcribe

- Add speech to text capabilities to applications
- Recorded speech can be converted to text before it can be used in applications
- Uses a deep learning process called automatic speech recognition (ASR) to convert speech to text quickly and accurately



Amazon Translate

- Neural machine translation service that delivers fast, high-quality, and affordable language translation
- Uses deep learning models to deliver more accurate and more natural sounding translation
- Localize content such as websites and applications for your diverse users



Amazon Comprehend

- Natural-language processing (NLP) service
- Uses machine learning to uncover information in unstructured data
- Can identify critical elements in data, including references to language, people, and places, and the text files can be categorized by relevant topics
- In real time, you can automatically and accurately detect customer sentiment in your content



Amazon Lex

- Conversational AI for Chatbots
- Build conversational interfaces into any application using voice and text
- Build bots to increase contact center productivity, automate simple tasks, and drive operational efficiencies across the enterprise



Amazon DevOps Guru

- Cloud operations service for improving **application operational performance and availability**
- Detect behaviors that deviate from normal operating patterns
- Benefits:
 - Automatically detect operational issues
 - Resolve issues with ML-powered insights
 - Elastically scale operational analytics
 - Uses ML to reduce alarm noise



Amazon CodeGuru Security

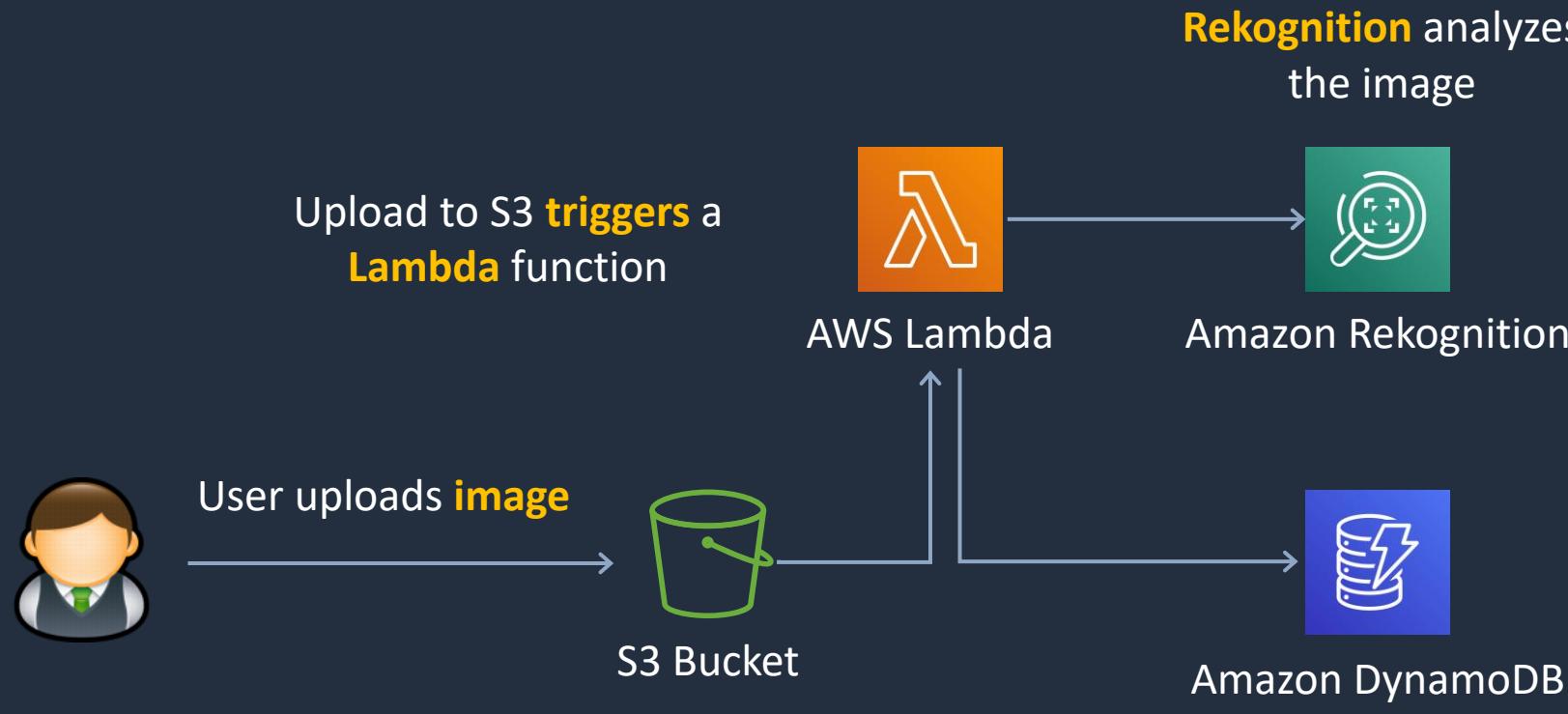
- Detect, track, and fix code security vulnerabilities anywhere in the development cycle using ML and automated reasoning
- Integrates with IDEs and CI/CD tools
- Automated bug tracking
- Assisted remediation through suggested code fixes
- Offers performance optimization recommendations
- Detects anomalies in application profiles

Process and Analyze Images





Process and Analyze Images



The **Lambda function** receives the results and creates an item in **DynamoDB**

SECTION 12

Get Certified on AWS

Get Certified on AWS



Why work in cloud computing?

1. Job demand
2. Globally relevant skills
3. Rewarding career paths
4. Great salaries

Why get AWS certified?

1. Demonstrate skills to employers
2. Differentiate yourself
3. Gain knowledge
4. Develop practical skills



AWS Certification

Professional



Specialty



Associate



Foundational



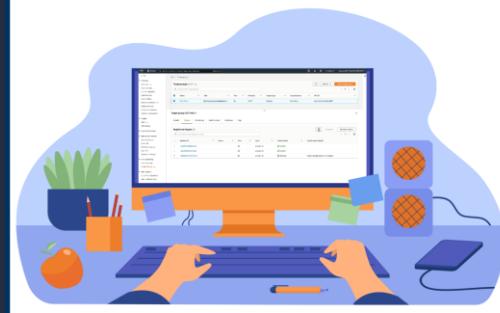
<https://aws.amazon.com/certification/exams/>

ON-DEMAND TRAINING COURSES



Prepare for your next AWS certification with video courses & practice exams

HANDS-ON CHALLENGE LABS



Build hands-on cloud skills in a secure sandbox environment

LIVE AWS BOOTCAMPS



Get ready for your next cloud job with real-world projects in a virtual classroom