

# SECTION 1

Let's Get Started!

# The SAA-C02 Exam





# The SAA-C02 Exam

---

---

**Level:** Associate

**Length:** 130 minutes

**Format:** 65 questions

**Cost:** \$150 USD

**Delivery Method:** Testing center or online

**Scoring:**

- Scaled score between 100 – 1000
- Minimum passing score of 720



# The SAA-C02 Exam

---

## Recommended knowledge:

- One year of hands-on experience with AWS technology
- Experience deploying, managing, and operating workloads on AWS
- Familiarity with using both the AWS Management Console and the AWS Command Line Interface (CLI)
- Understanding of the AWS Well-Architected Framework, AWS networking, security services, and the AWS global infrastructure
- Ability to identify which AWS services meet a given technical requirement and to define technical requirements for an AWS-based application



## Question format:

- Multiple-choice: Has one correct response and three incorrect responses
- Multiple-response: Has two or more correct responses out of five or more options



# The SAA-C02 Exam

---

---

## Domain 1: Design Resilient Architectures

- 1.1 Design a multi-tier architecture solution
- 1.2 Design highly available and/or fault-tolerant architectures
- 1.3 Design decoupling mechanisms using AWS services
- 1.4 Choose appropriate resilient storage

## Domain 2: Design High-Performing Architectures

- 2.1 Identify elastic and scalable compute solutions for a workload
- 2.2 Select high-performing and scalable storage solutions for a workload
- 2.3 Select high-performing networking solutions for a workload
- 2.4 Choose high-performing database solutions for a workload



# The SAA-C02 Exam

---

---

## Domain 3: Design Secure Applications and Architectures

- 3.1 Design secure access to AWS resources
- 3.2 Design secure application tiers
- 3.3 Select appropriate data security options

## Domain 4: Design Cost-Optimized Architectures

- 4.1 Identify cost-effective storage solutions
- 4.2 Identify cost-effective compute and database services
- 4.3 Design cost-optimized network architectures



# The SAA-C02 Exam

---

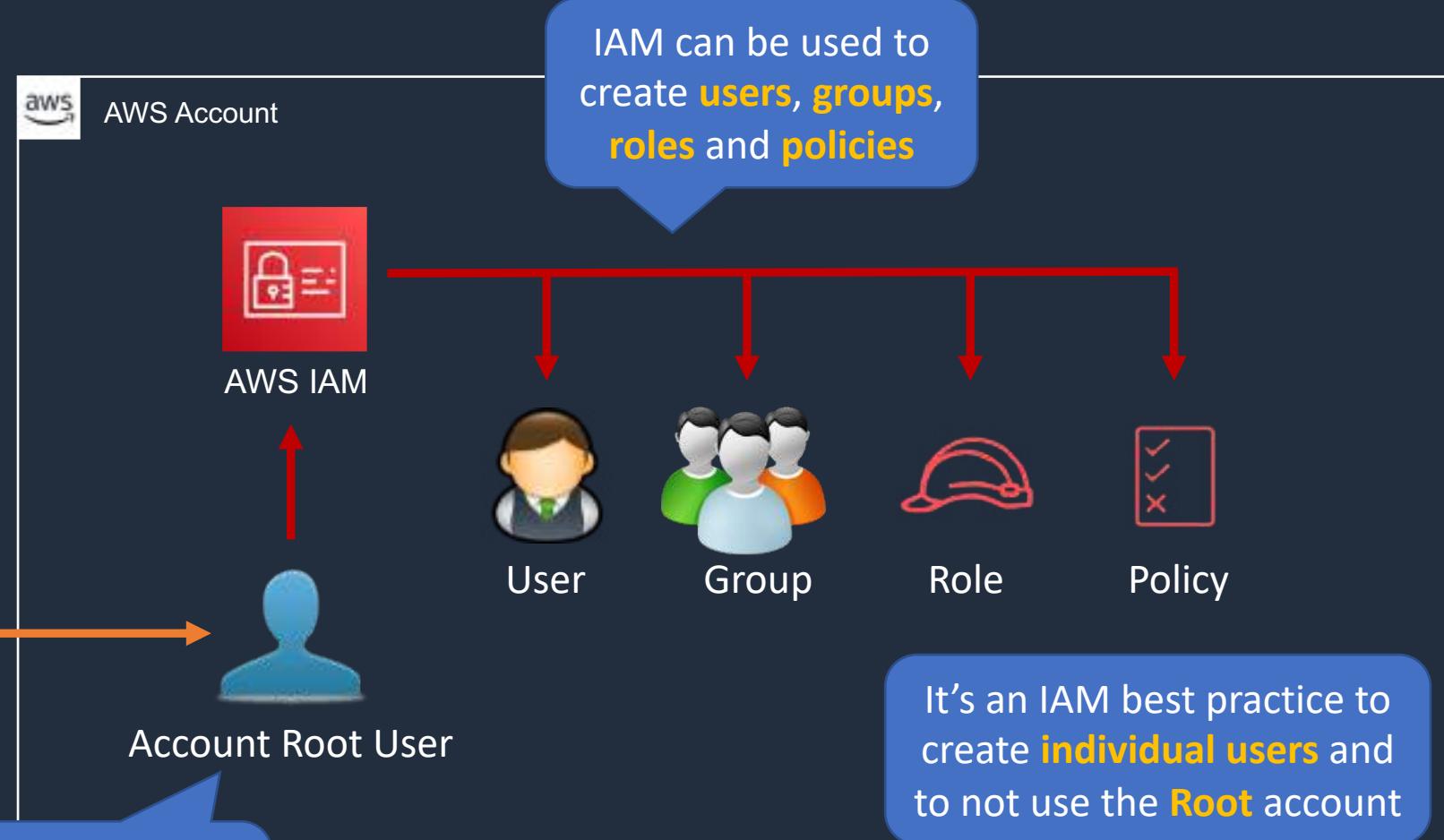
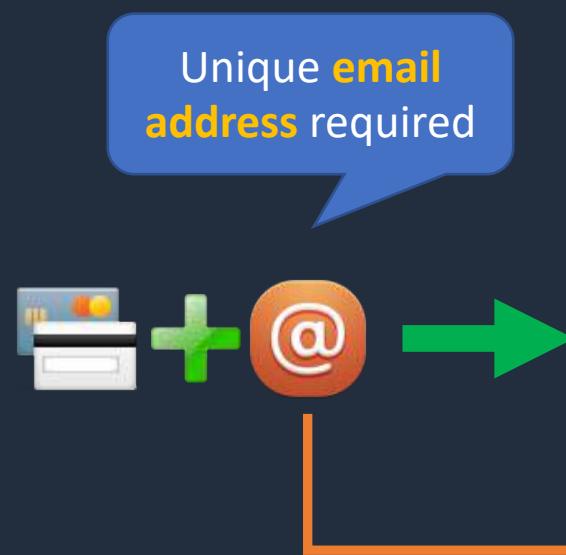
---

Domain	% of Exam
Domain 1: Design Resilient Architectures	30%
Domain 2: Design High-Performing Architectures	28%
Domain 3: Design Secure Applications and Architectures	24%
Domain 4: Design Cost-Optimized Architectures	18%
<b>TOTAL</b>	<b>100%</b>

# AWS Account Overview



# AWS Account Overview



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



# AWS Account Overview



AWS Account



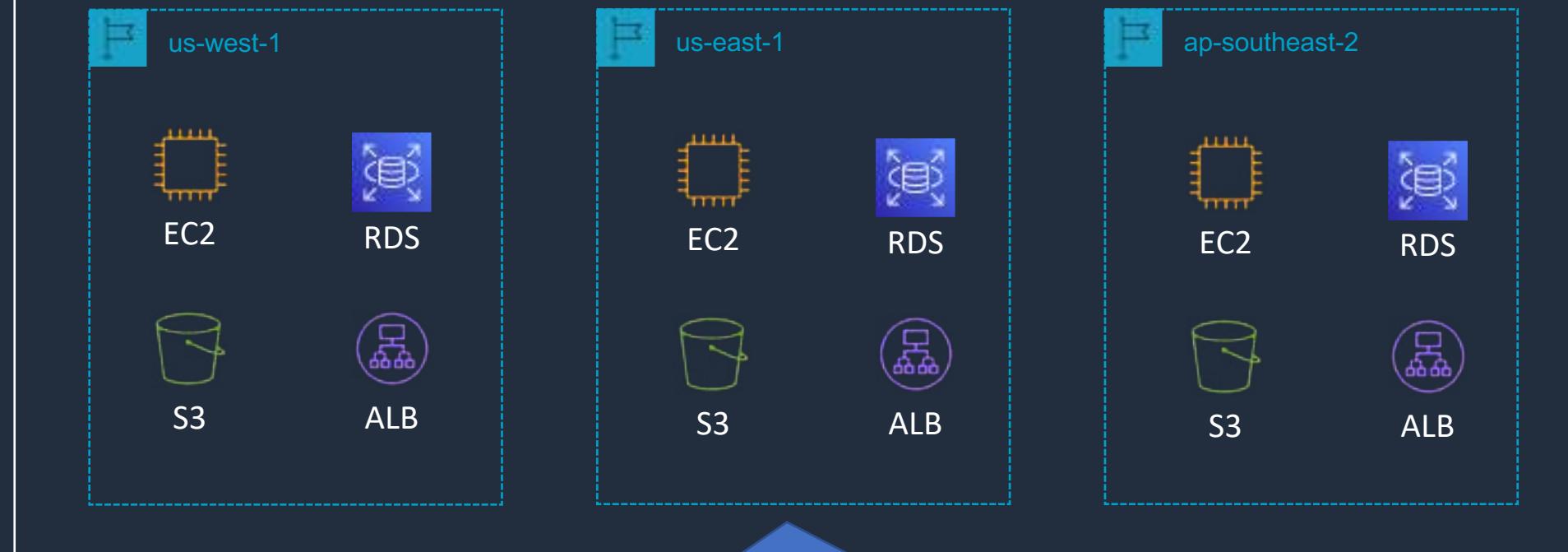
AWS Management  
Console

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

**Authorization:** IAM principals can then create resources across AWS Regions



AWS IAM



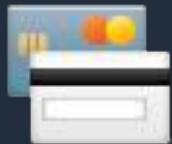
All AWS **identities** and **resources** are created within the AWS account

# Create Your AWS Free Tier Account





# What you need...



Credit card for setting up the account and paying any bills



Unique email address for this account

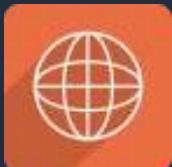
john@gmail.com



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 Setup Billing Alarm





# Account Configuration

- Configure **Account Alias**
- Enable access to billing for **IAM users**
- Update **billing preferences**
- Create a **billing alarm**
- Confirm **SNS subscription**

# Install Tools (AWS CLI, VS Code, CloudShell)





# Install Tools

---

- Install the **AWS Command Line Interface (CLI)**
- Install **Visual Studio Code**
- Launch **AWS CloudShell**

# SECTION 2

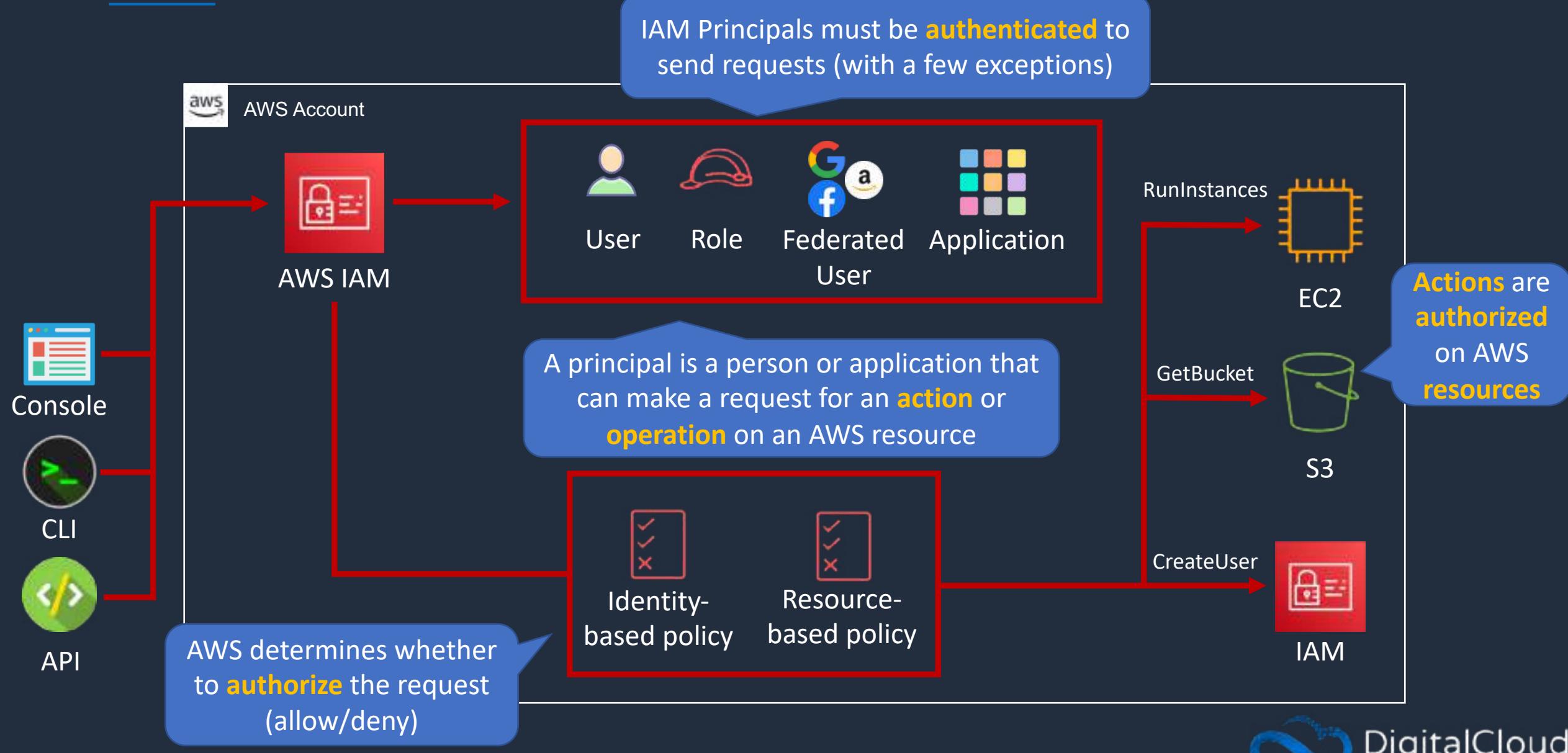
## AWS Identity and Access Management (IAM)

# AWS IAM Overview





# AWS Identity and Access Management (IAM)

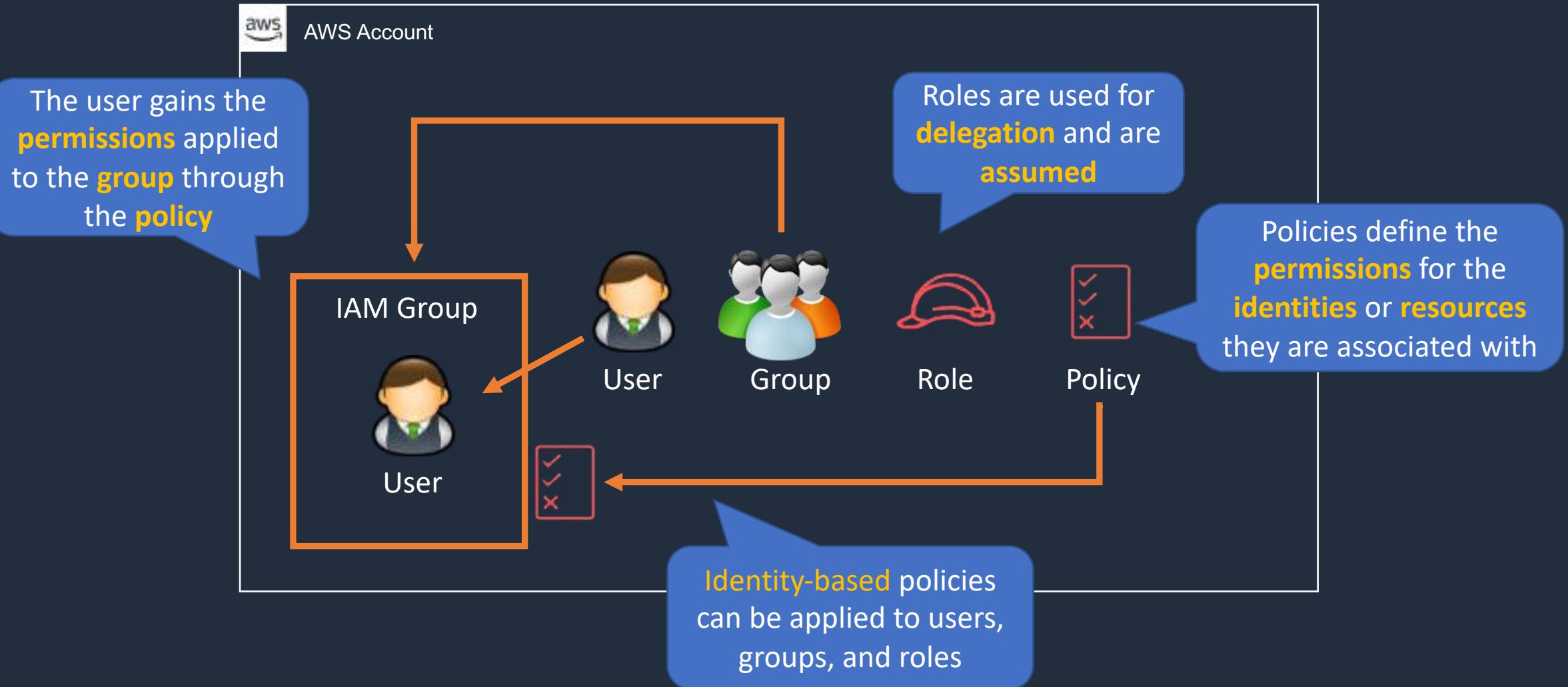


# IAM Users, Groups, Roles, and Policies



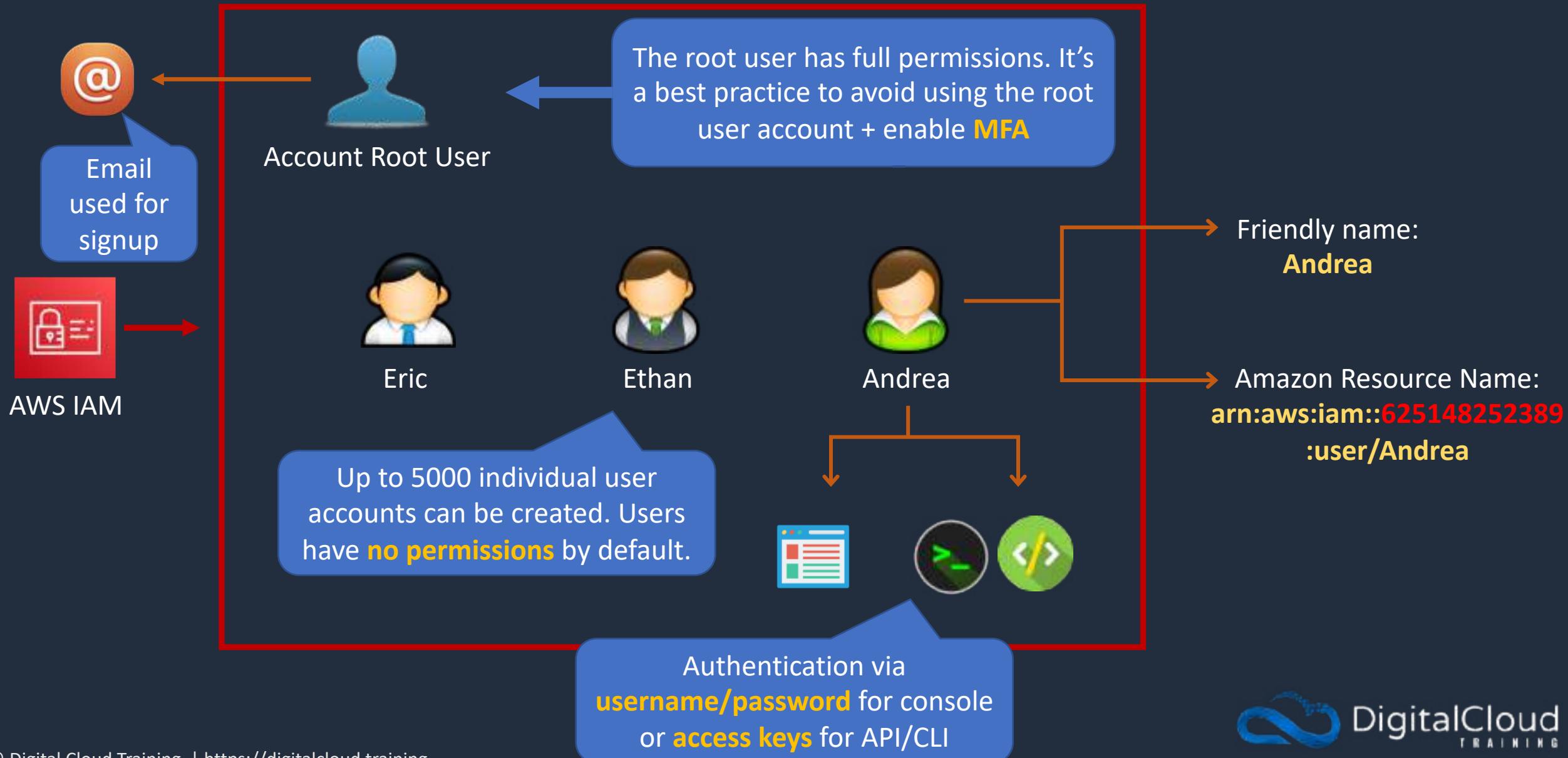


# Users, Groups, Roles and Policies



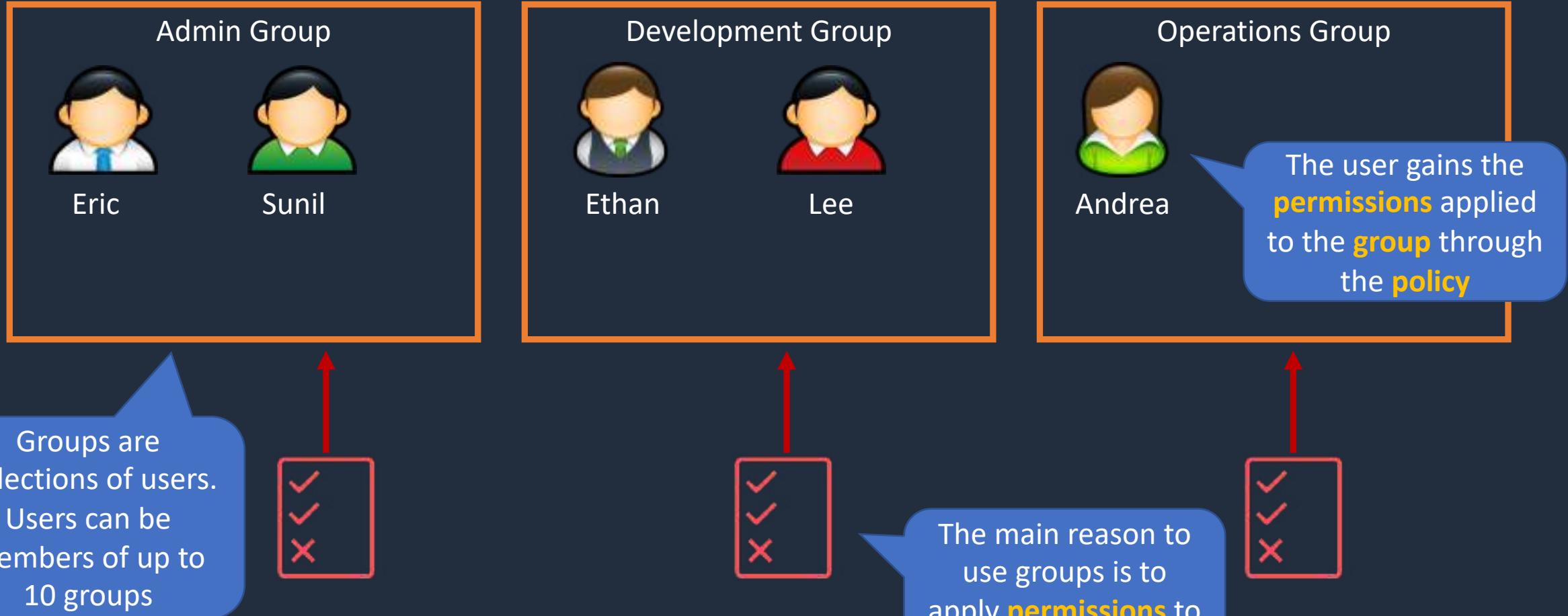


# IAM Users





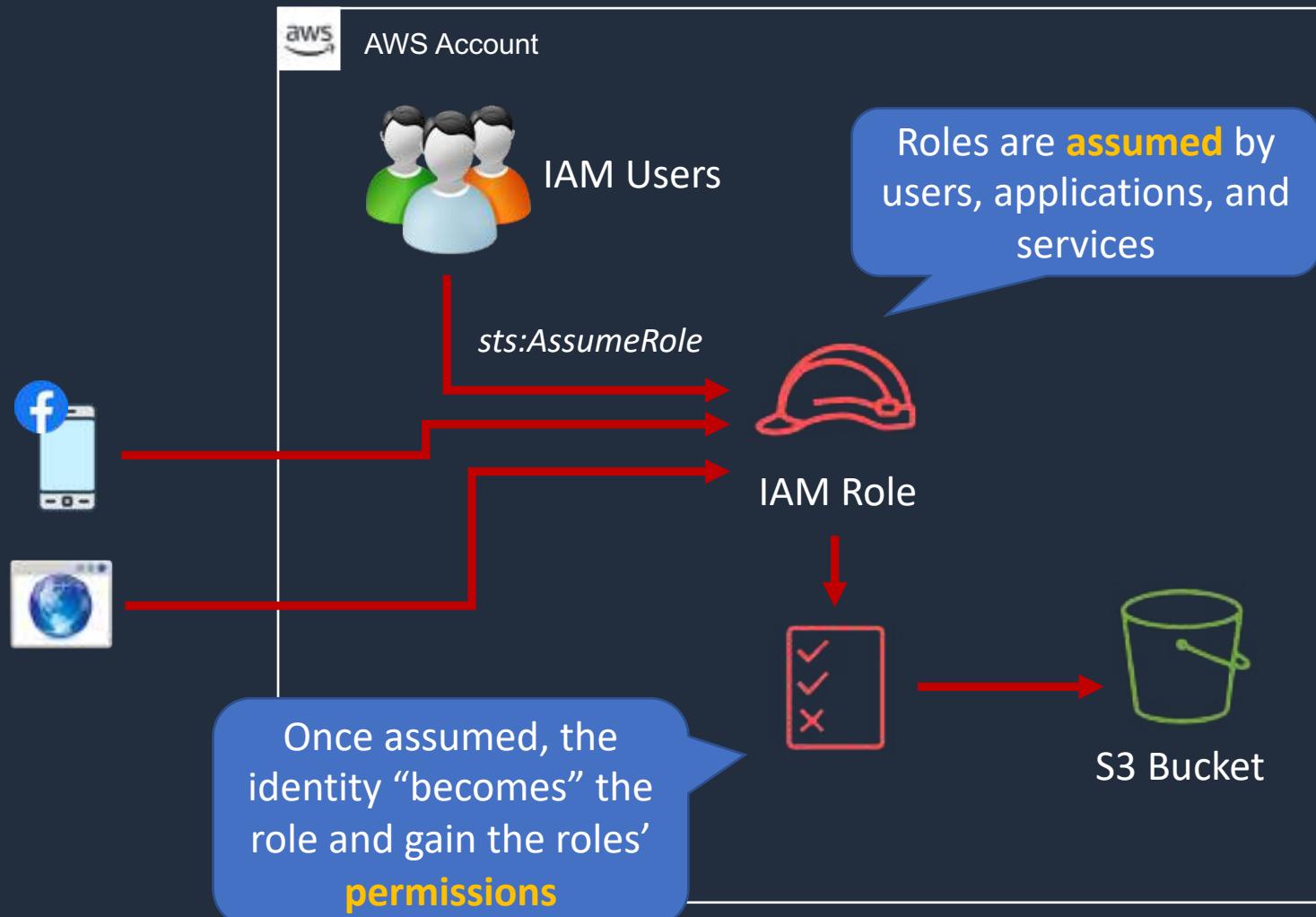
# IAM Groups





# IAM Roles

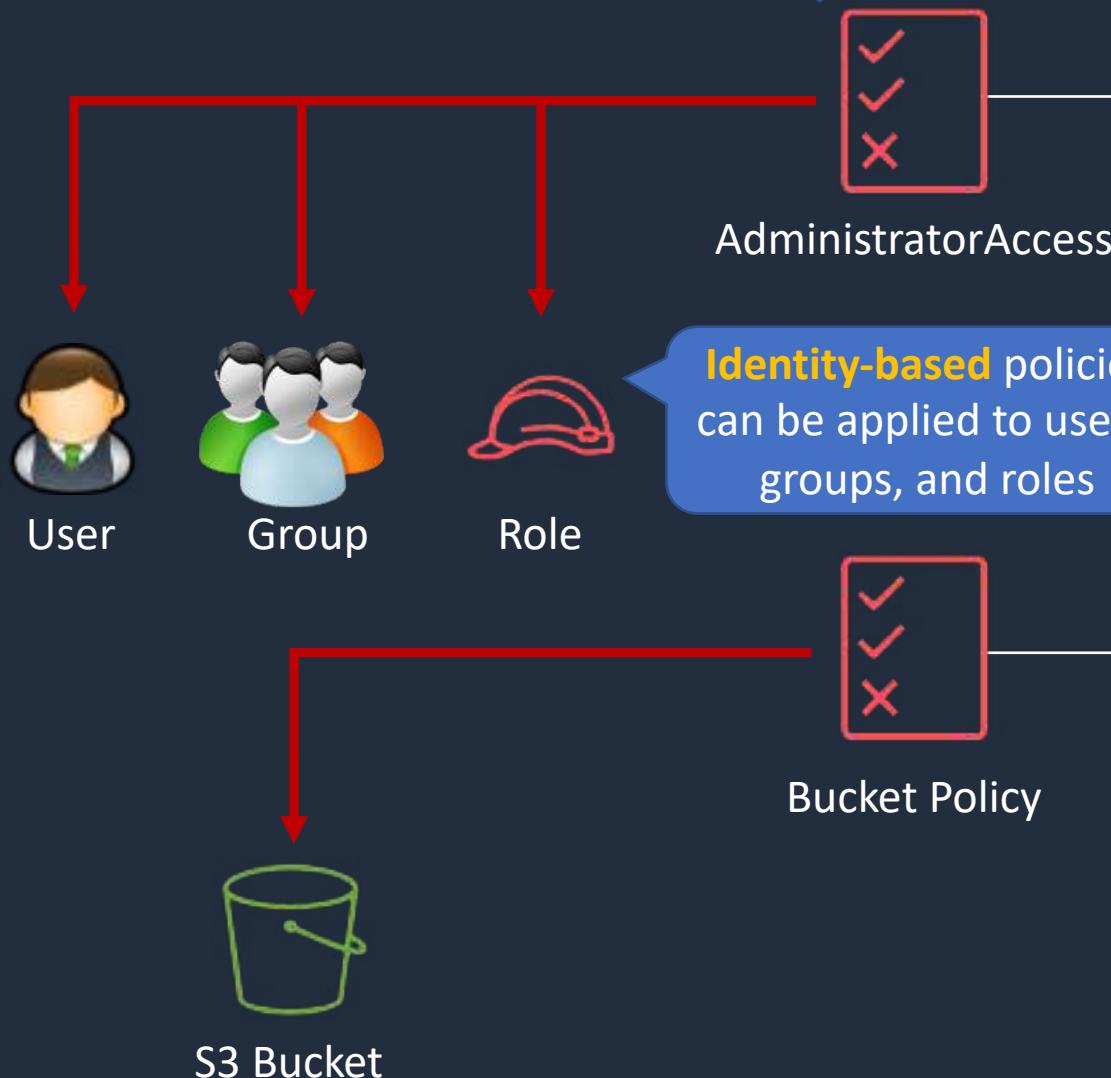
An **IAM role** is an IAM **identity** that that has specific **permissions**





# IAM Policies

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



```
{  
  "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": "  
        ARN:aws:iam::515148227241:user/Paul",  
      "Action": "s3:*",  
      "Resource": "arn:aws:s3:::detcompany",  
      "Condition": {  
        "StringLike": {  
          "s3:prefix": "Confidential/*"  
        }  
      }  
    }  
  ]  
}
```

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

# Create IAM User Account





# Root User vs IAM User

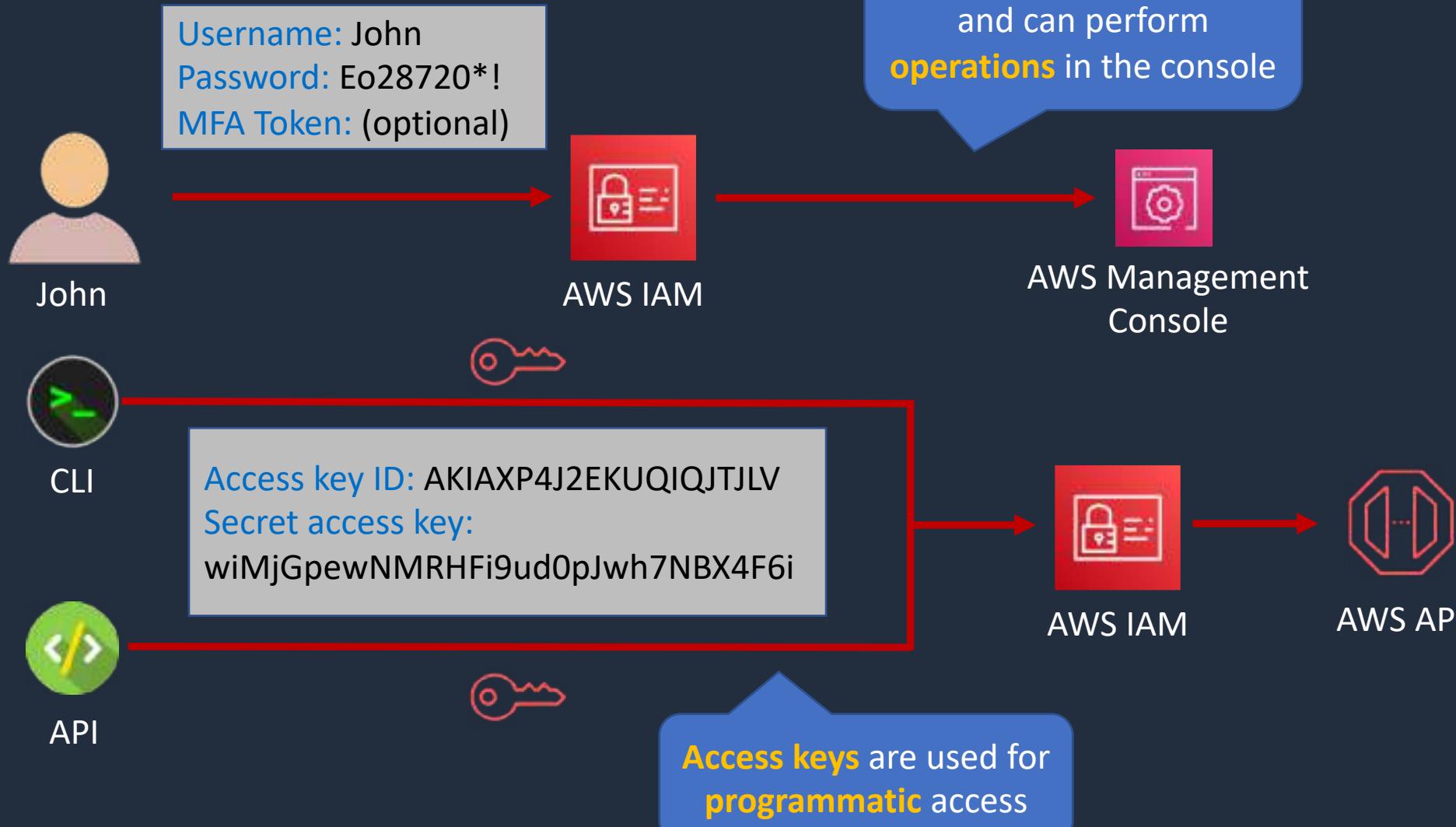
User	Login Details	Permissions
Root User	Email address	Full - Unrestricted
IAM User	Friendly name: <b>John</b> + AWS account ID or Alias	IAM Permissions Policy

# IAM Authentication and MFA





# IAM Authentication Methods





# Multi-Factor Authentication

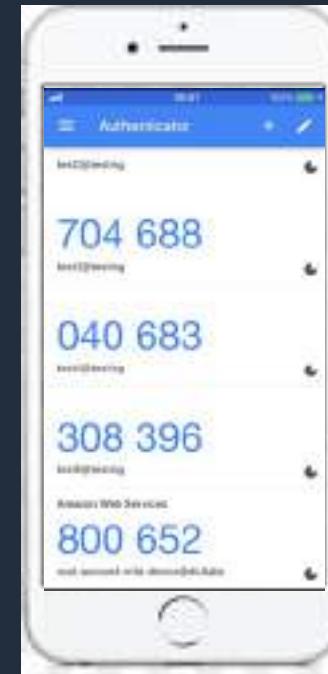
---

Something you **know**:

EJPx!\*21p9%

Password

Something you **have**:



Something you **are**:





# Multi-Factor Authentication

---

Something you **know**:



IAM User

EJPx!\*21p9%

Password

Something you **have**:

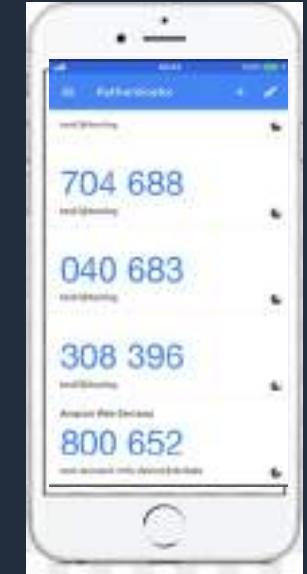


Virtual MFA



Physical MFA

e.g. Google Authenticator on  
your smart phone

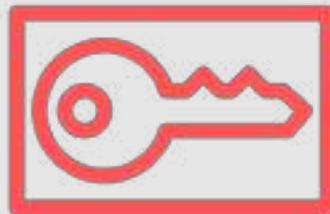


Physical tokens can  
be purchased from  
**third parties**

# Enable Multi-Factor Authentication (MFA)

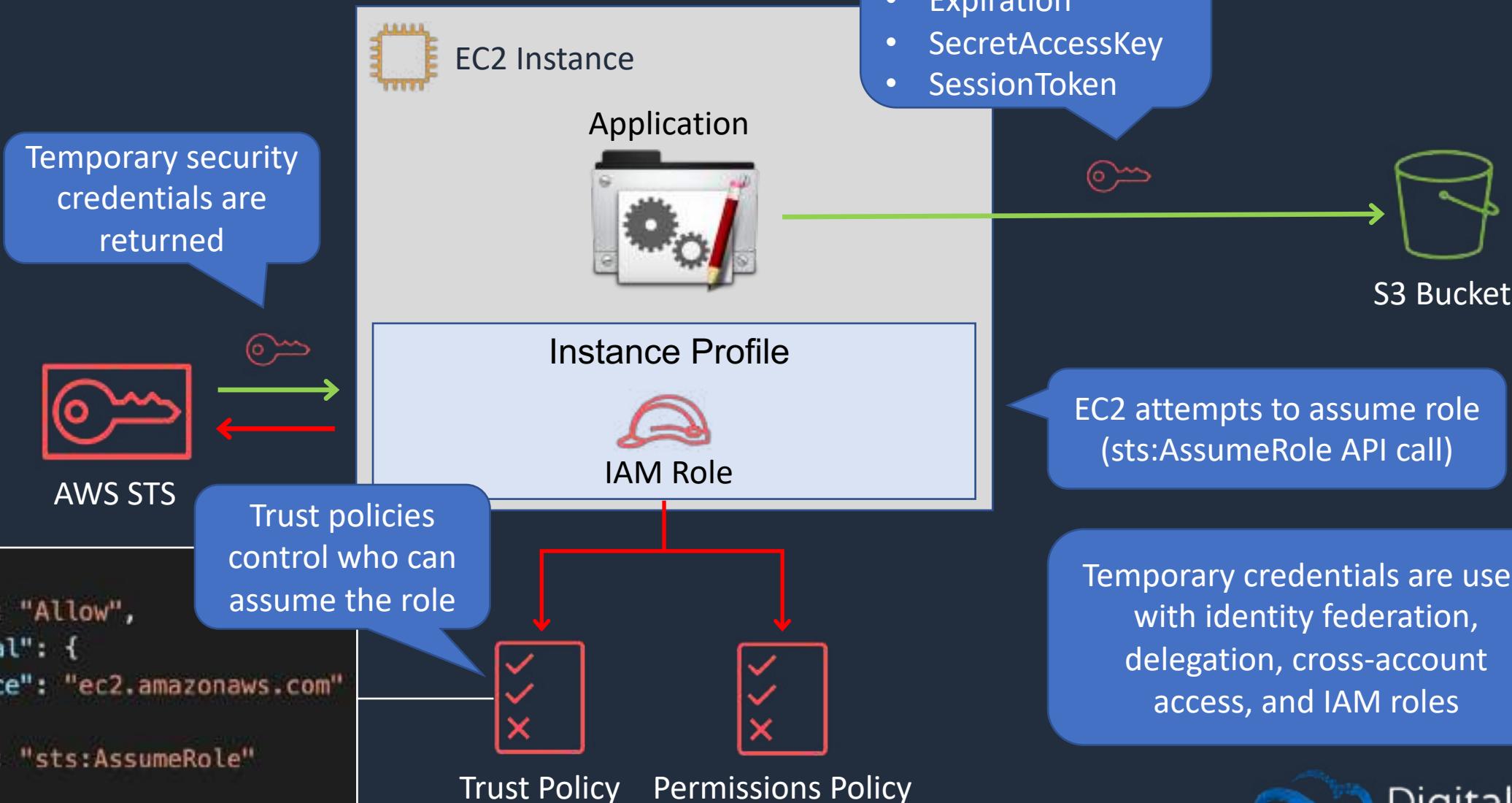


# AWS Security Token Service (STS)





# AWS Security Token Service (STS)



# IAM Password Policy



# Identity-Based Policies and Resource-Based Policies





# Identity-Based IAM Policies

Identity-based policies are JSON permissions policy documents that control what actions an identity can perform, on which resources, and under what conditions

Managed policy. Either AWS managed or customer managed

AWS managed are created and managed by AWS; customer managed are created and managed by you



Managed policies are standalone policies that can be attached to multiple users, groups, or roles



User



Inline policy



Group



Role



Inline policies have a 1-1 relationship with the user, group, or role



# Resource-Based Policies

---

Resource-based policies are JSON policy documents that you attach to a resource such as an Amazon S3 bucket



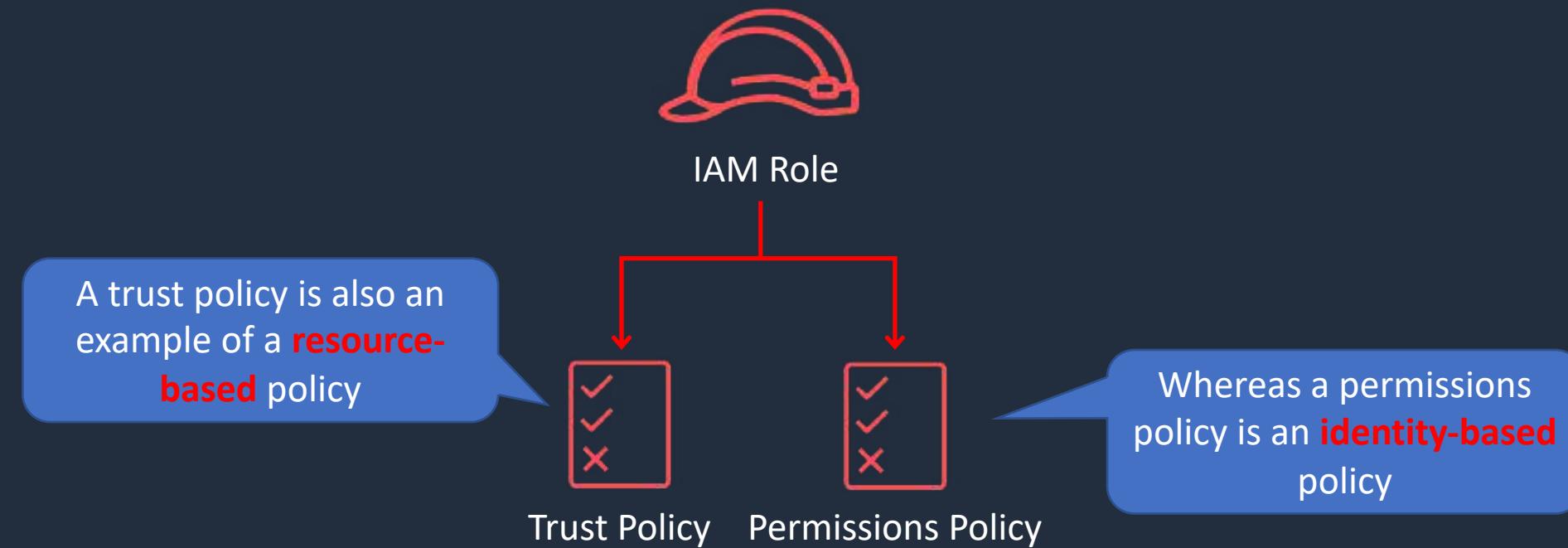
Resource-based policies grant the specified **principal** (Paul) **permission** to perform specific **actions** on the **resource**

```
{  
  "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"  
    }  
  ]  
}
```



# Resource-Based Policies

---

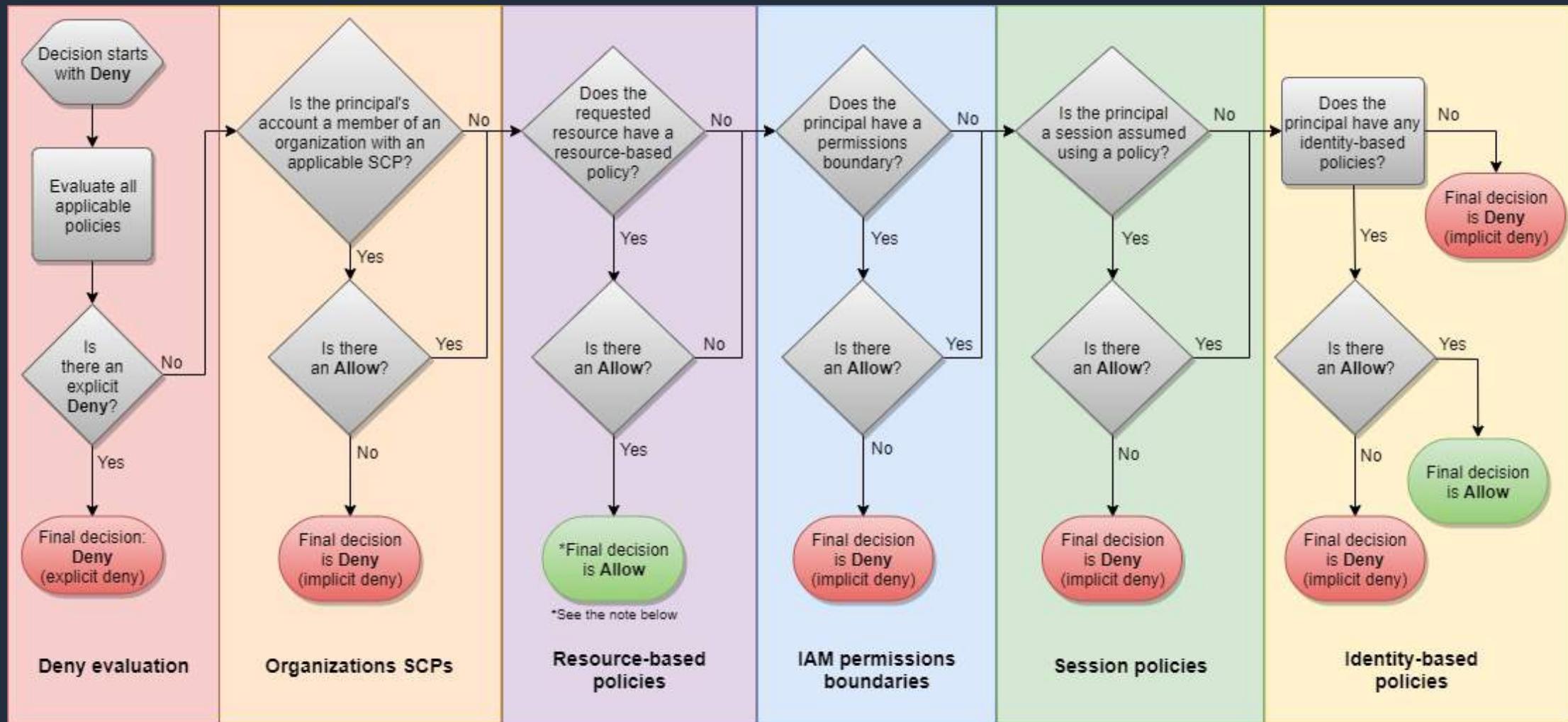


# IAM Policy Evaluation Logic





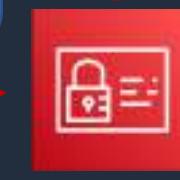
# Evaluation Logic





# Steps for Authorizing Requests to AWS

1. Authentication – AWS authenticates the principal that makes the request



AWS IAM



Identity-based policy

3. Evaluating all policies within the account



Resource-based policy

s3:GetObject



S3 Bucket

4. Determining whether a request is allowed or denied

Console



AWS IAM

CLI



API



- Request context:
- **Actions** – the actions or operations the principal wants to perform
  - **Resources** – The AWS resource object upon which actions are performed
  - **Principal** – The user, role, federated user, or application that sent the request
  - **Environment data** – Information about the IP address, user agent, SSL status, or time of day
  - **Resource data** – Data related to the resource that is being requested

2. Processing the request context

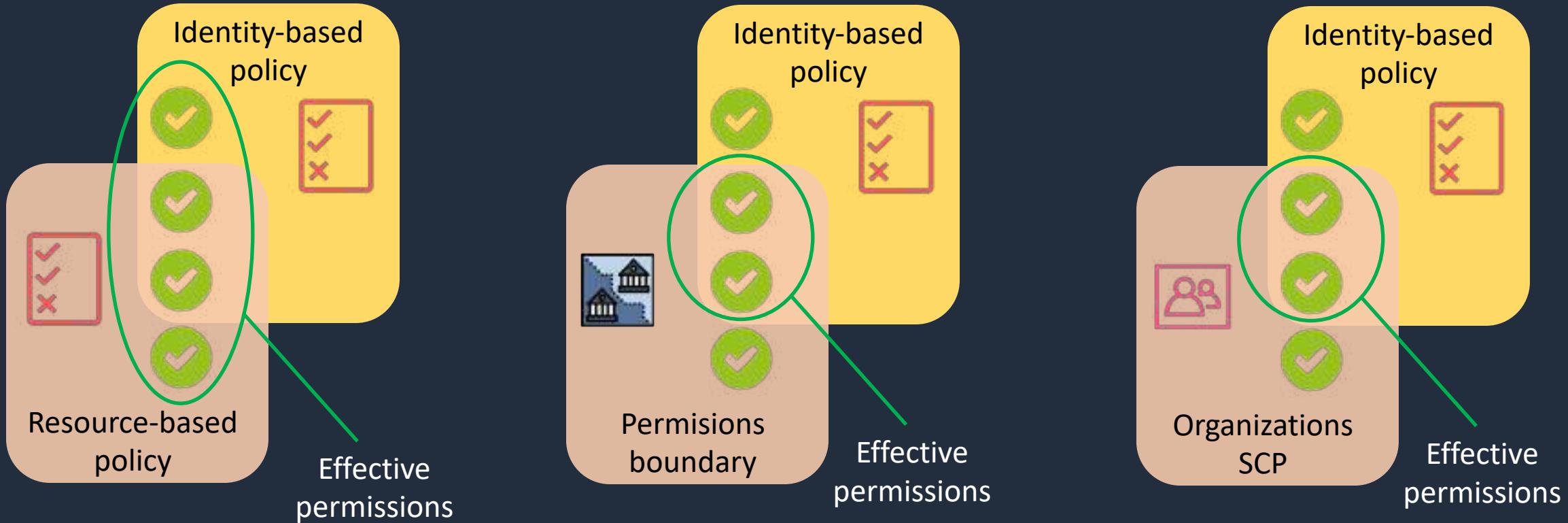


# Types of Policy

- **Identity-based policies** – attached to users, groups, or roles
- **Resource-based policies** – attached to a resource; define permissions for a principal accessing the resource
- **IAM permissions boundaries** – set the maximum permissions an identity-based policy can grant an IAM entity
- **AWS Organizations service control policies (SCP)** – specify the maximum permissions for an organization or OU
- **Session policies** – used with AssumeRole\* API actions



# Evaluating Policies within an AWS Account





# Determination Rules

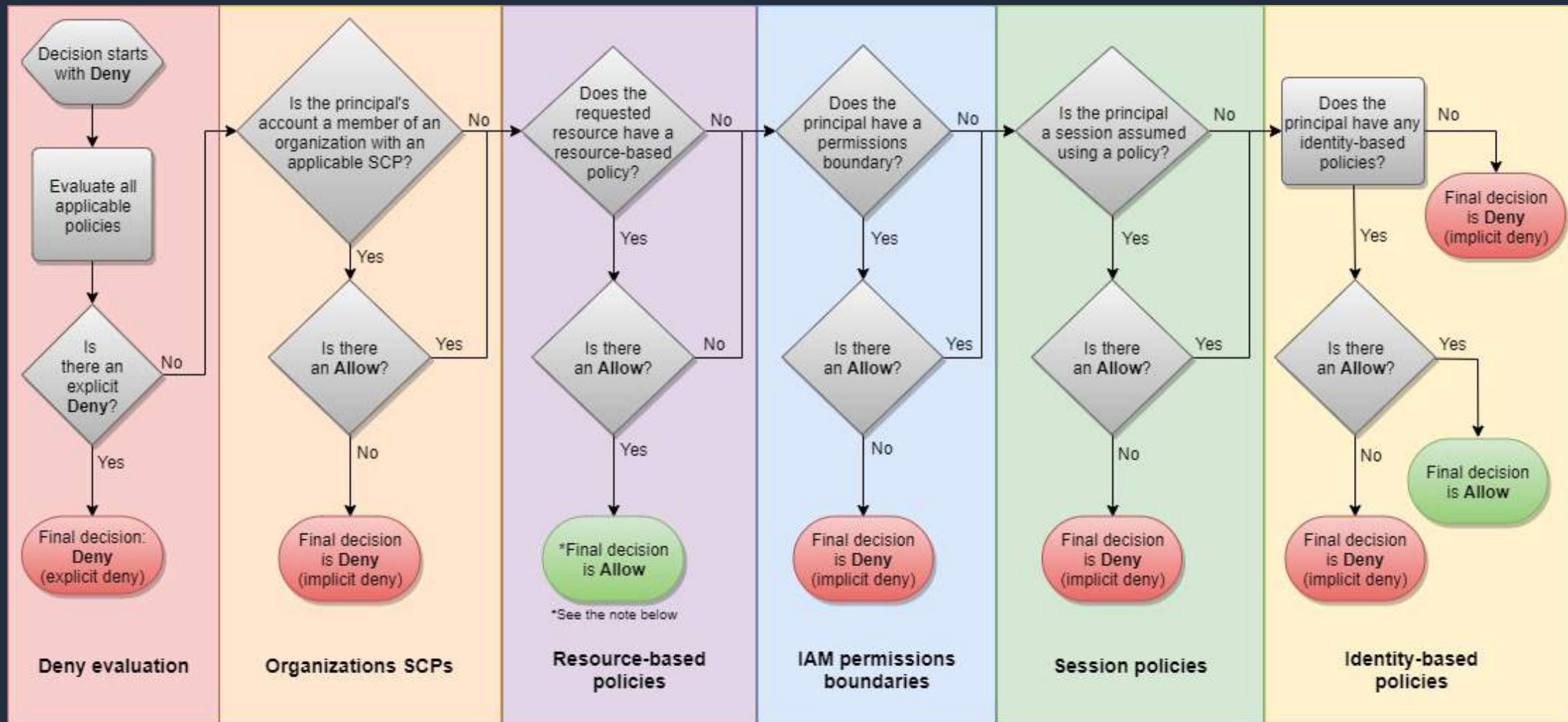
---

---

1. By default, all requests are implicitly denied (though the root user has full access)
2. An explicit allow in an identity-based or resource-based policy overrides this default
3. If a permissions boundary, Organizations SCP, or session policy is present, it might override the allow with an implicit deny
4. An explicit deny in any policy overrides any allows



# Evaluation Logic



# IAM Policy Structure





# IAM Policy Structure

An IAM policy is a JSON document that consists of one or more statements

The **Action** element is the specific API action for which you are granting or denying permission

```
{  
  "Statement": [  
    {  
      "Effect": "effect",  
      "Action": "action",  
      "Resource": "arn",  
      "Condition": {  
        "condition": {  
          "key": "value"  
        }  
      }  
    }  
  ]  
}
```

The **Effect** element can be Allow or Deny

The **Resource** element specifies the resource that's affected by the action

The **Condition** element is optional and can be used to control when your policy is in effect



# IAM Policy Example 1

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

The AdministratorAccess policy uses wildcards (\*) to allow all actions on all resources



# IAM Policy Example 2

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": ["ec2:TerminateInstances"],  
            "Resource": ["*"]  
        },  
        {  
            "Effect": "Deny",  
            "Action": ["ec2:TerminateInstances"],  
            "Condition": {  
                "NotIpAddress": {  
                    "aws:SourceIp": [  
                        "192.0.2.0/24",  
                        "203.0.113.0/24"  
                    ]  
                }  
            },  
            "Resource": ["*"]  
        }  
    ]  
}
```

The specific API action is defined

The effect is to deny the API action if the IP address is not in the specified range



# IAM Policy Example 3

```
{  
    "Version": "2012-10-17",  
    "Id": "ExamplePolicy01",  
    "Statement": [  
        {  
            "Sid": "ExampleStatement01",  
            "Effect": "Allow",  
            "Principal": {  
                "AWS": "*"  
            },  
            "Action": [  
                "elasticfilesystem:ClientRootAccess",  
                "elasticfilesystem:ClientMount",  
                "elasticfilesystem:ClientWrite"  
            ],  
            "Condition": {  
                "Bool": {  
                    "aws:SecureTransport": "true"  
                }  
            }  
        }  
    ]  
}
```

You can tell this is a resource-based policy as it has a principal element defined

The policy grants read and write access to an EFS file systems to all IAM principals ("AWS ": "\*")

Additionally, the policy condition element requires that SSL/TLS encryption is used



# IAM Policy Example 4

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Action": ["s3>ListBucket"],  
            "Effect": "Allow",  
            "Resource": ["arn:aws:s3:::mybucket"],  
            "Condition": {"StringLike": {"s3:prefix": ["${aws:username}/*"]}}  
        },  
        {  
            "Action": [  
                "s3:GetObject",  
                "s3:PutObject"  
            ],  
            "Effect": "Allow",  
            "Resource": ["arn:aws:s3:::mybucket/${aws:username}/*"]  
        }  
    ]  
}
```

A variable is used for the s3:prefix that is replaced with the user's friendly name

The actions are allowed only within the user's folder within the bucket

# IAM Policy Simulator



# IAM Best Practices





# AWS IAM Best Practices

---

---

- Lock away your AWS account root user access keys
- Create individual IAM users
- Use groups to assign permissions to IAM users
- Grant least privilege
- Get started using permissions with AWS managed policies
- Use customer managed policies instead of inline policies
- Use access levels to review IAM permissions
- Configure a strong password policy for your users
- Enable MFA



# AWS IAM Best Practices

---

---

- Use roles for applications that run on Amazon EC2 instances
- Use roles to delegate permissions
- Do not share access keys
- Rotate credentials regularly
- Remove unnecessary credentials
- Use policy conditions for extra security
- Monitor activity in your AWS account

# Architecture Patterns – AWS IAM





# Architecture Patterns – AWS IAM

## Requirement

A select group of users only should be allowed to change their IAM passwords

An Amazon EC2 instance must be delegated with permissions to an Amazon DynamoDB table

A company has created their first AWS account. They need to assign permissions to users based on job function

## Solution

Create a group for the users and apply a permissions policy that grants the iam:ChangePassword API permission

Create a role and assign a permissions policy to the role that grants access to the database service

Use AWS managed policies that are aligned with common job functions



# Architecture Patterns – AWS IAM

## Requirement

A solutions architect needs to restrict access to an AWS service based on the source IP address of the requester

## Solution

Create an IAM permissions policy and use the Condition element to control access based on source IP address

A developer needs to make programmatic API calls from the AWS CLI

Instruct the developer to create a set of access keys and use those for programmatic access

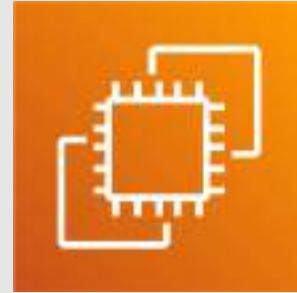
A group of users require full access to all Amazon EC2 API actions

Create a permissions policy that uses a wildcard for the Action element relating to EC2 (ec2:\*)

# SECTION 3

## Amazon Elastic Compute Cloud (EC2)

# Amazon Elastic Compute Cloud (EC2)



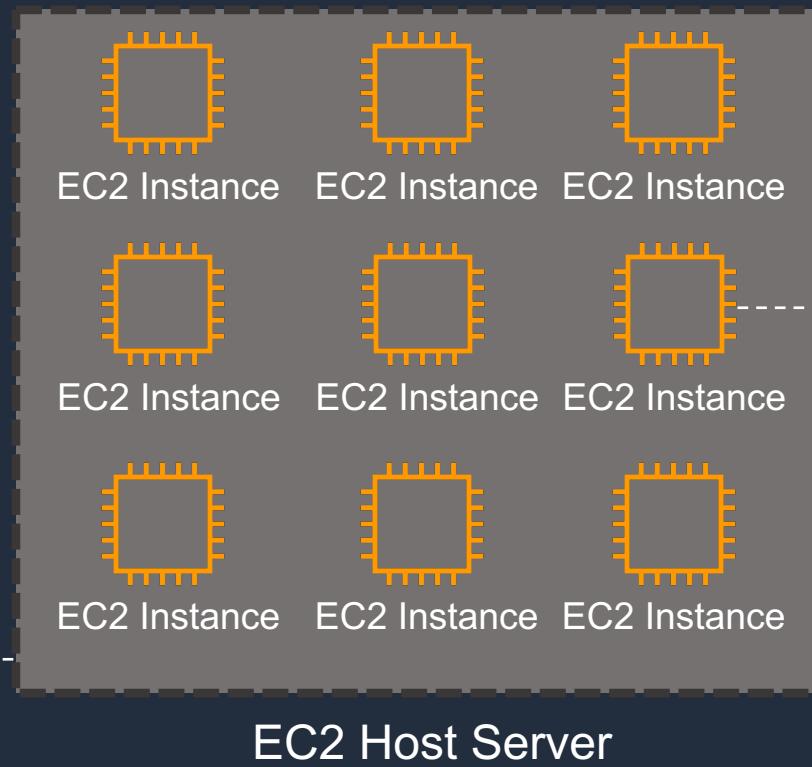


# Amazon EC2

EC2 instances run  
Windows, Linux, or  
MacOS

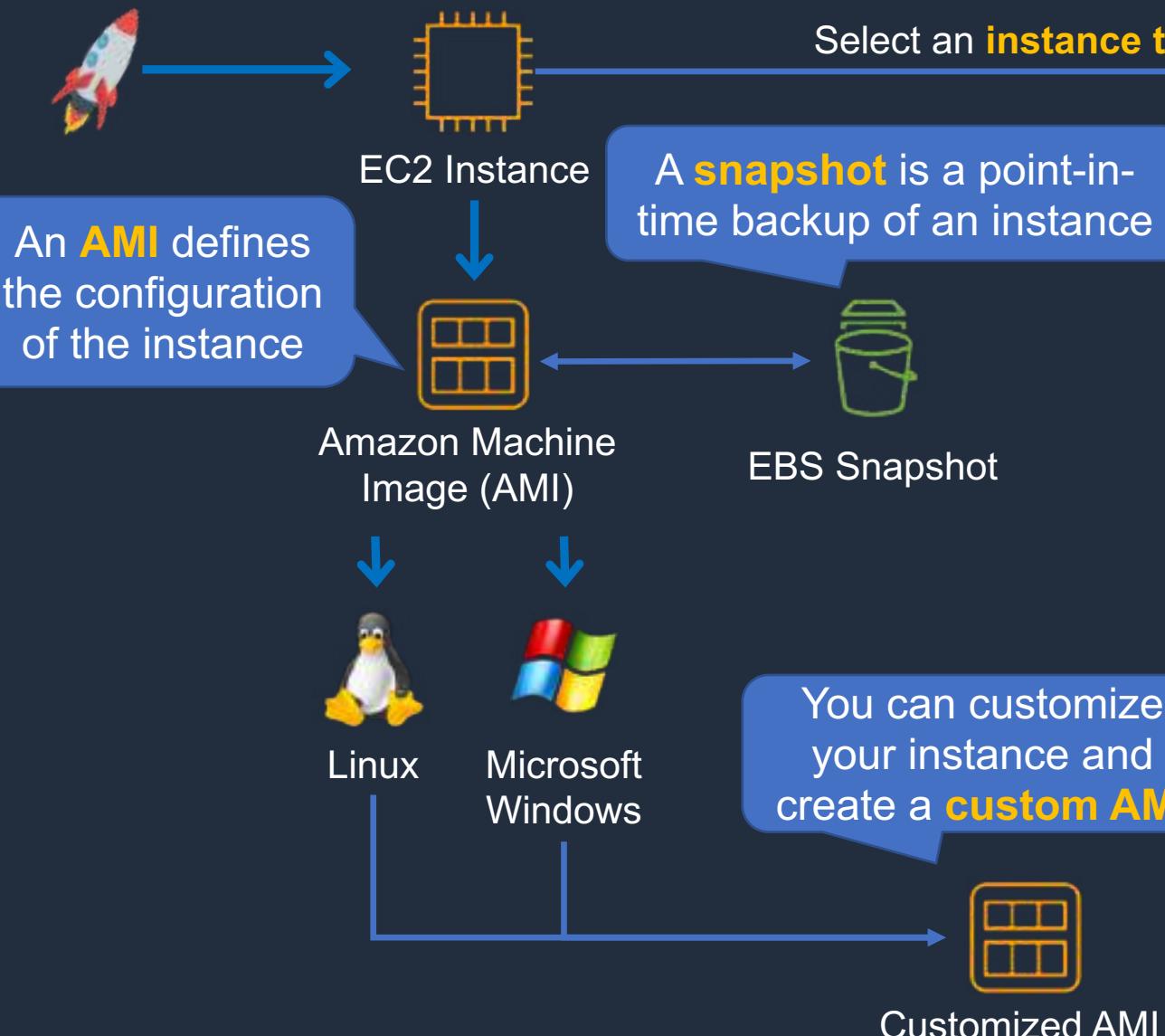
An **EC2 instance**  
is a virtual server

EC2 hosts are  
**managed by AWS**

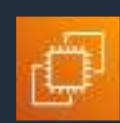


A selection of **instance types**  
come with varying combinations  
of CPU, memory, storage and  
networking

# Launching an EC2 Instance



The **instance type** defines the hardware profile (and cost)



# Benefits of Amazon EC2

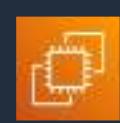
---

---

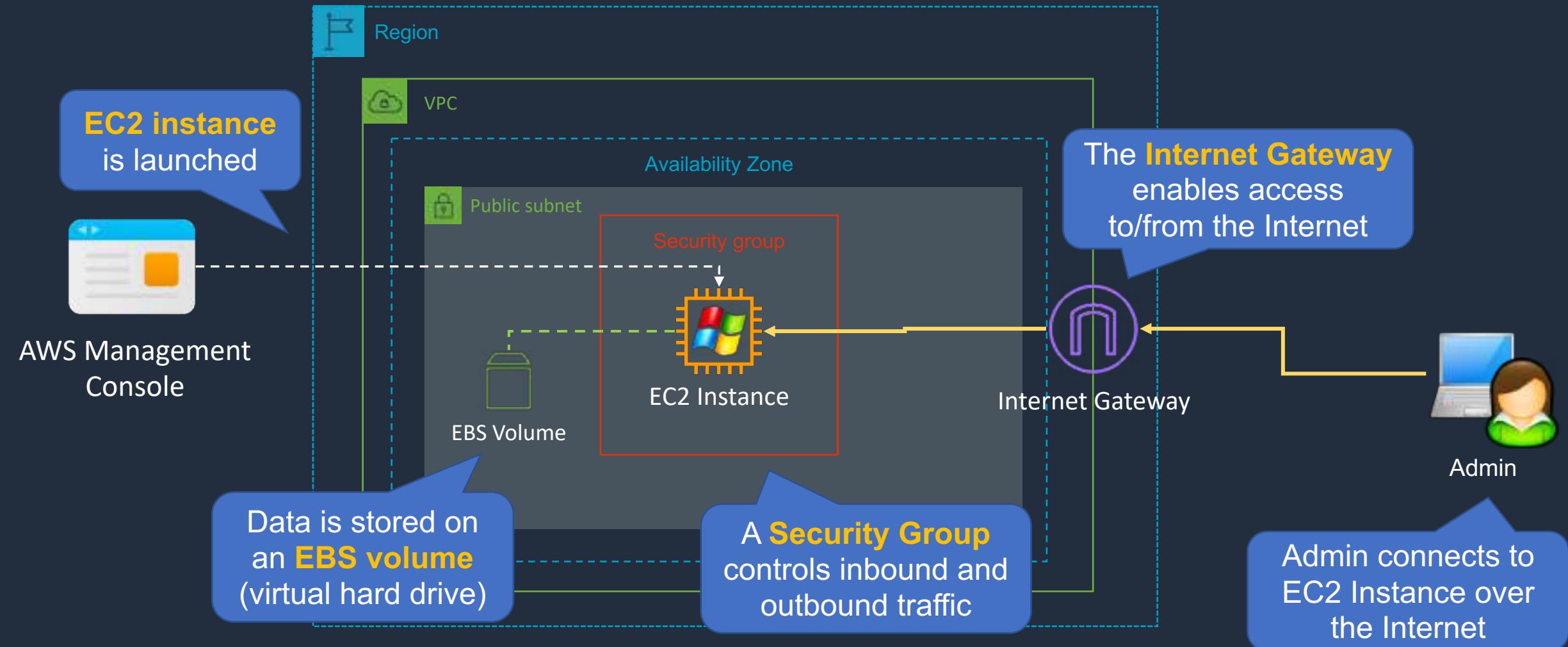
- **Elastic computing** – easily launch hundreds to thousands of EC2 instances within minutes
- **Complete control** – you control the EC2 instances with full root/administrative access
- **Flexible** – Choice of instance types, operating systems, and software packages
- **Reliable** – EC2 offers very high levels of availability and instances can be rapidly commissioned and replaced
- **Secure** – Fully integrated with Amazon VPC and security features
- **Inexpensive** – Low cost, pay for what you use

# Launch EC2 Instances (Windows + Linux)





# Amazon EC2 Instance in a Public Subnet



# EC2 Instance Connect and SSH



# RDP to Windows Instance



# Amazon EC2 User Data and Metadata





# 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

User data i  As text  As file  Input is already base64 encoded

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

Limited to  
**16 KB**



EC2 Instance

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



# Amazon EC2 Metadata

- Instance metadata is data about your EC2 instance
- Instance metadata is available at <http://169.254.169.254/latest/meta-data>
- Examples:



```
[ec2-user@ip-172-31-42-248 ~]$ curl http://169.254.169.254/latest/meta-data
ami-id
ami-launch-index
ami-manifest-path
block-device-mapping/
events/
hibernation/
hostname
identity-credentials/
instance-action
instance-id
instance-life-cycle
instance-type
local-hostname
local-ipv4
```



# Amazon EC2 Metadata

- Examples ctd.:

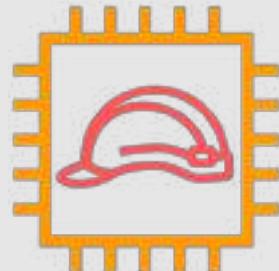
```
[ec2-user@ip-172-31-42-248 ~]$ curl http://169.254.169.254/latest/meta-data/local-ipv4  
172.31.42.248[ec2-user@ip-172-31-42-248 ~]$
```

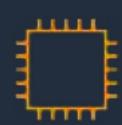
```
[ec2-user@ip-172-31-42-248 ~]$ curl http://169.254.169.254/latest/meta-data/public-ipv4  
3.26.54.18[ec2-user@ip-172-31-42-248 ~]$
```

# [HOL] Launch Instance with User Data and Metadata

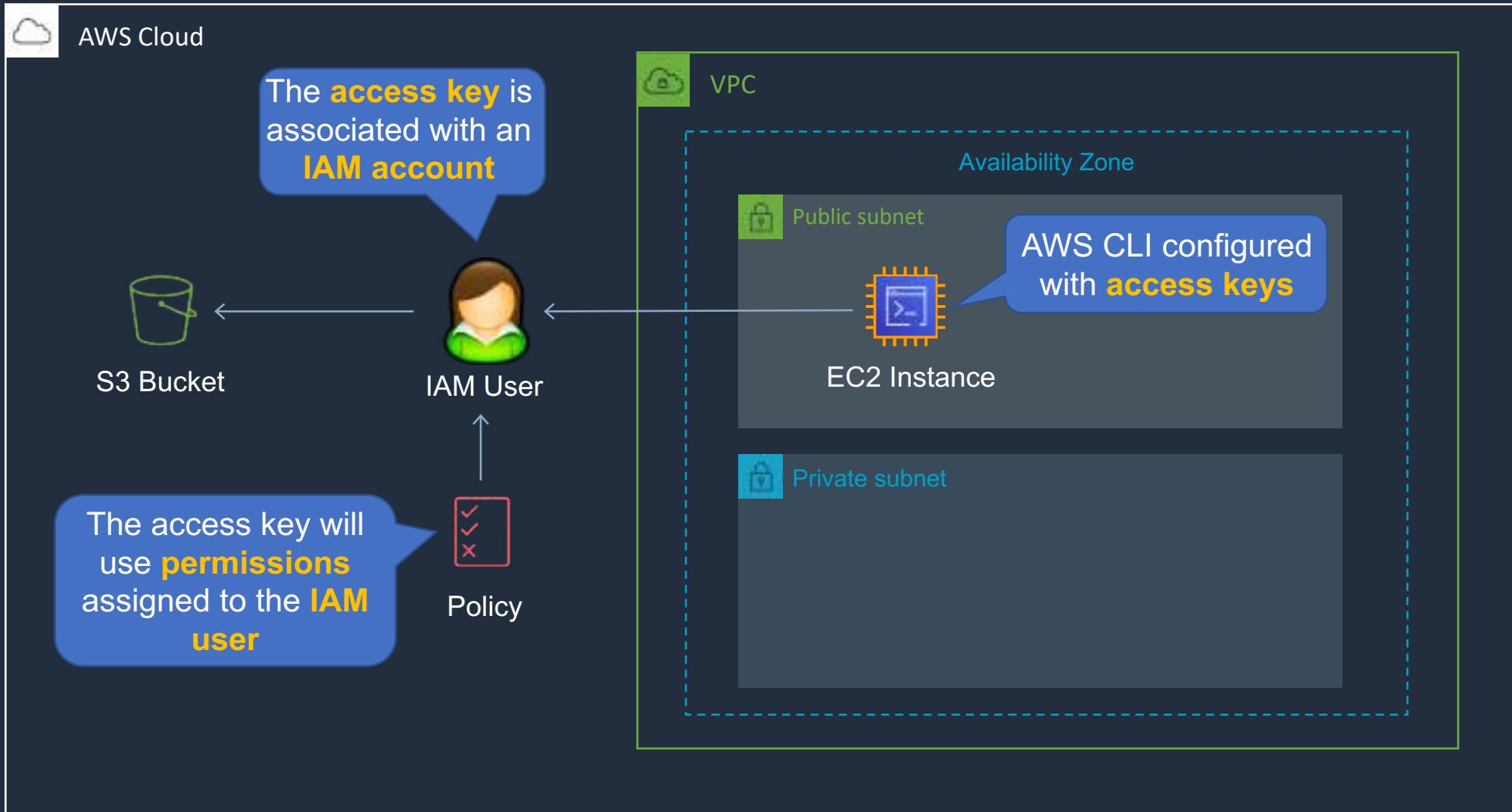


# Accessing Services – Access Keys and IAM Roles



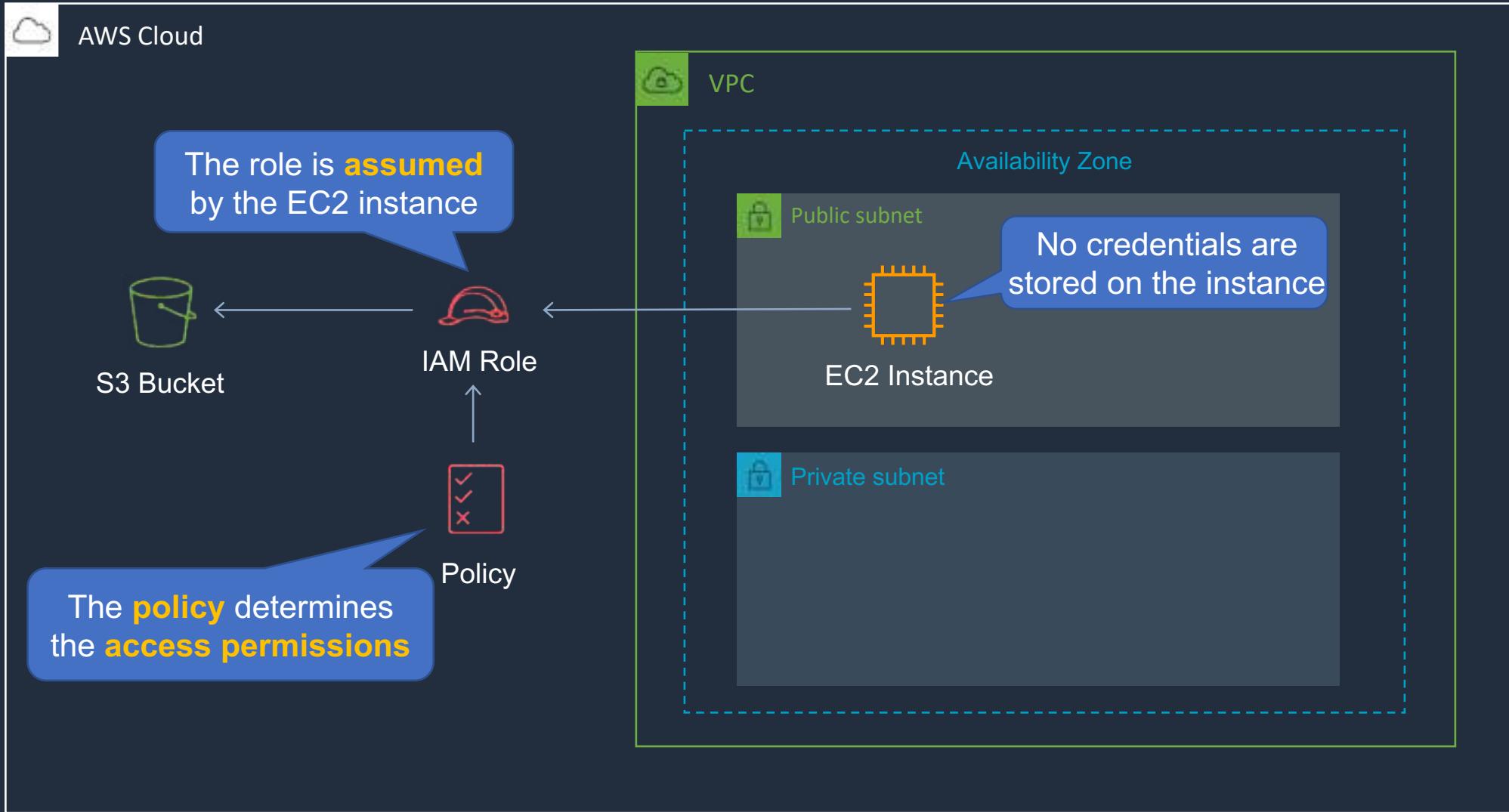


# Access Keys





# Amazon EC2 Instance Profiles (IAM Roles for EC2)



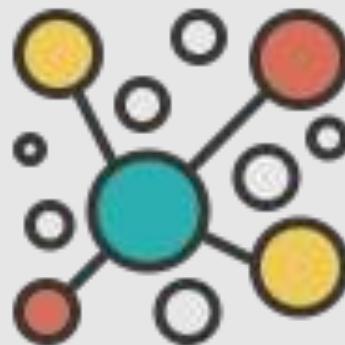
# Access Keys and IAM Roles



# Status Checks and Monitoring



# EC2 Placement Groups





# EC2 Placement Groups

---

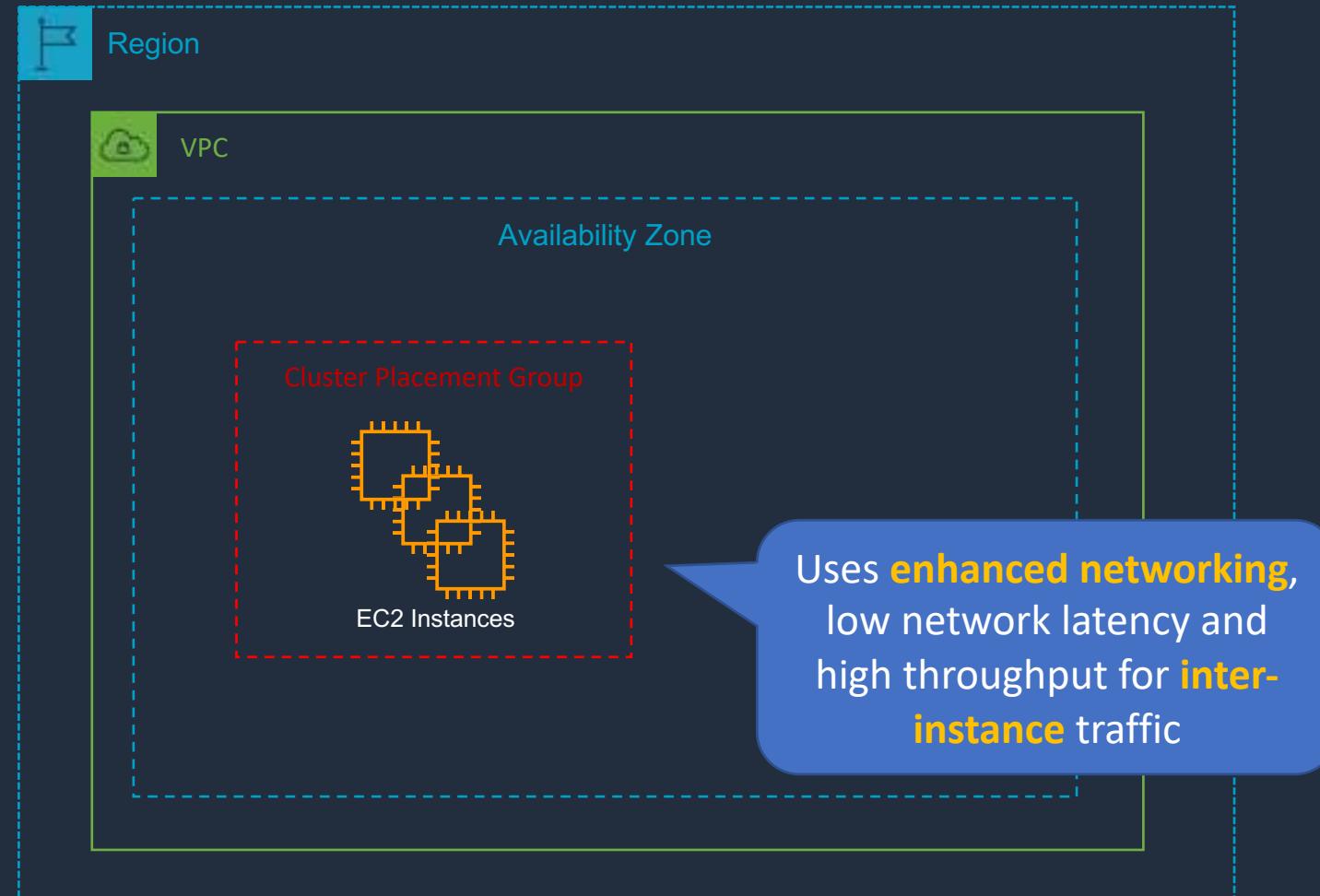
---

- **Cluster** – packs instances close together inside an Availability Zone. This strategy enables workloads to achieve the **low-latency** network performance necessary for **tightly-coupled** node-to-node communication that is typical of **HPC applications**
- **Partition** – spreads your instances across logical partitions such that groups of instances in one partition **do not share the underlying hardware** with groups of instances in different partitions. This strategy is typically used by large **distributed and replicated workloads**, such as Hadoop, Cassandra, and Kafka
- **Spread** – strictly places a small group of instances across **distinct underlying hardware** to reduce correlated failures



# Cluster Placement Group

---





# Partition Placement Group



Region



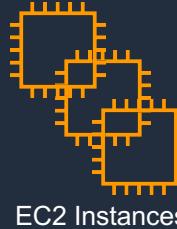
VPC

Availability Zone

Each partition is located on  
a **separate AWS rack**

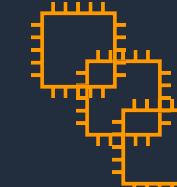
Availability Zone

Partition 1



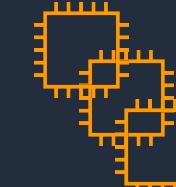
EC2 Instances

Partition 2



EC2 Instances

Partition 3

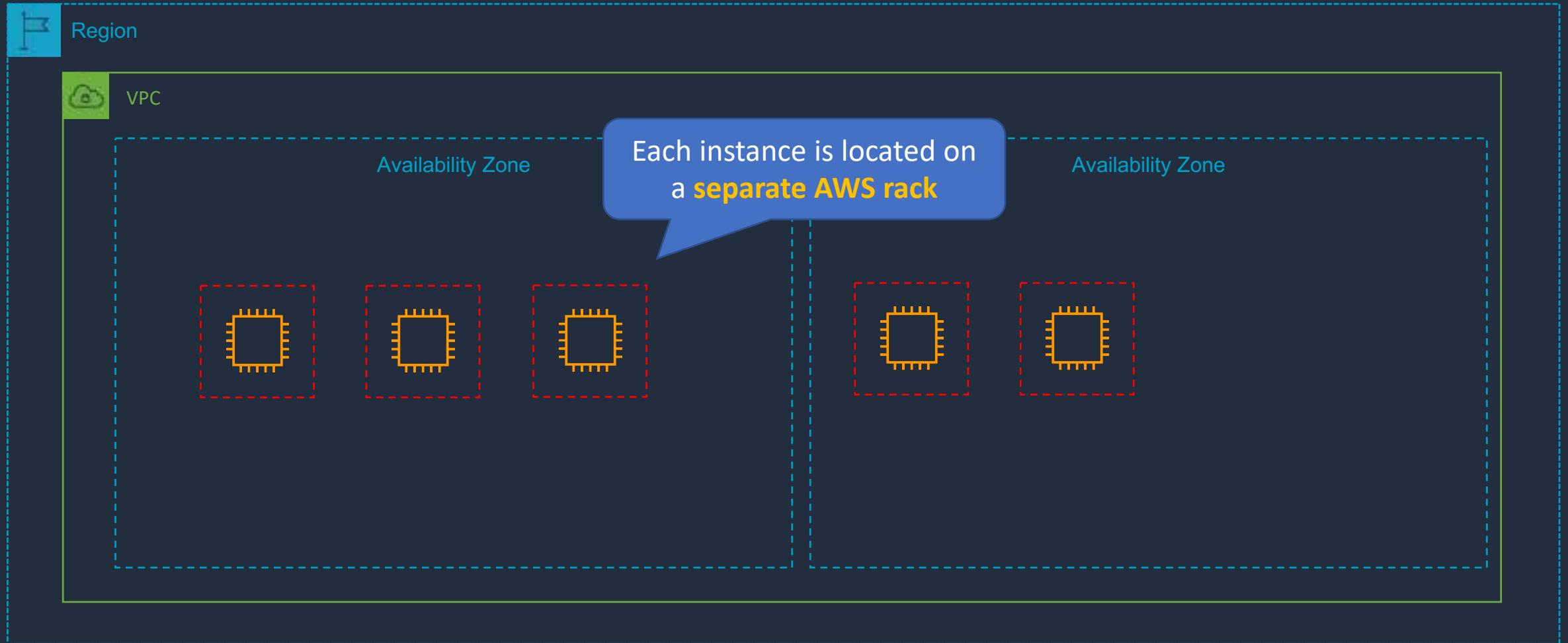


EC2 Instances

Partitions can be  
in **multiple AZs**  
(up to 7 per AZ)



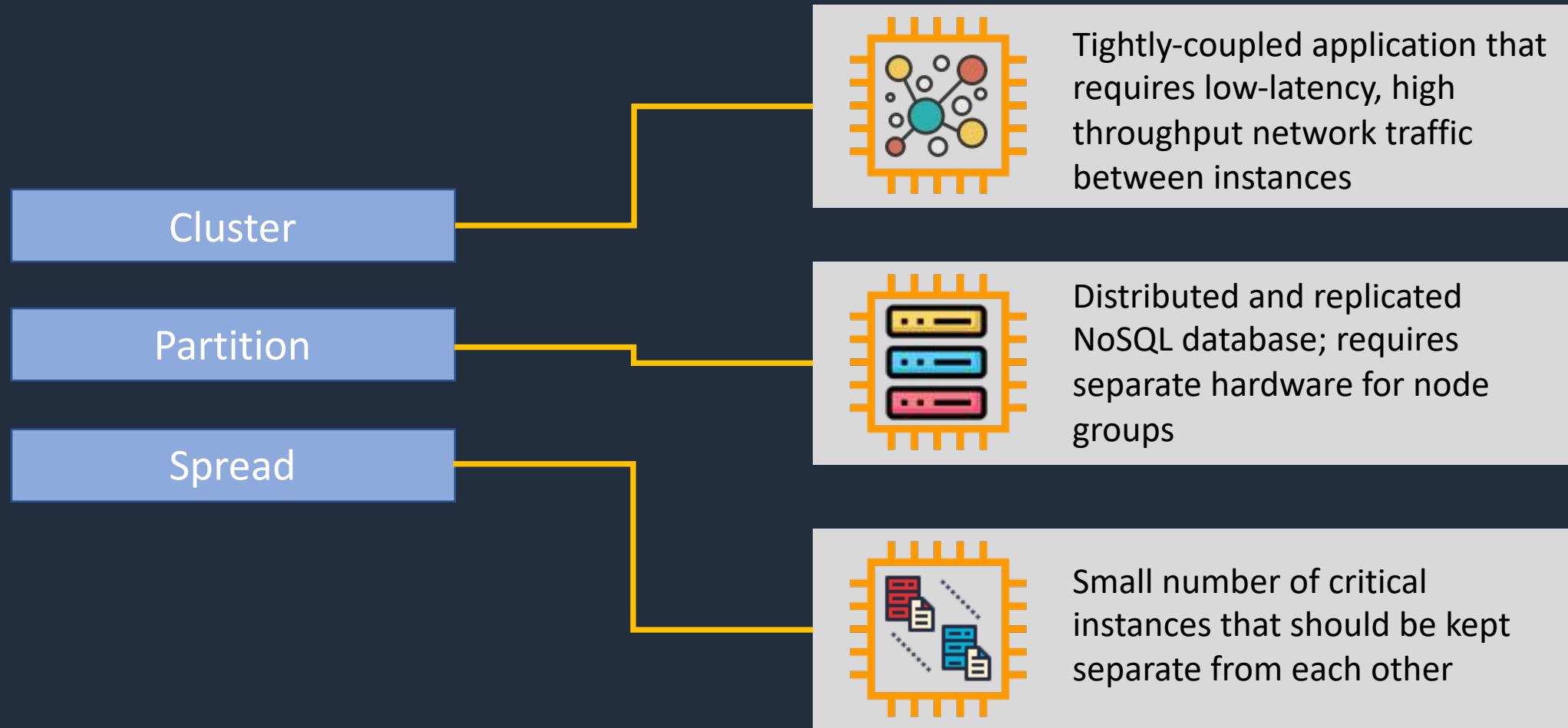
# Spread Placement Group





# EC2 Placement Group Use Cases

---

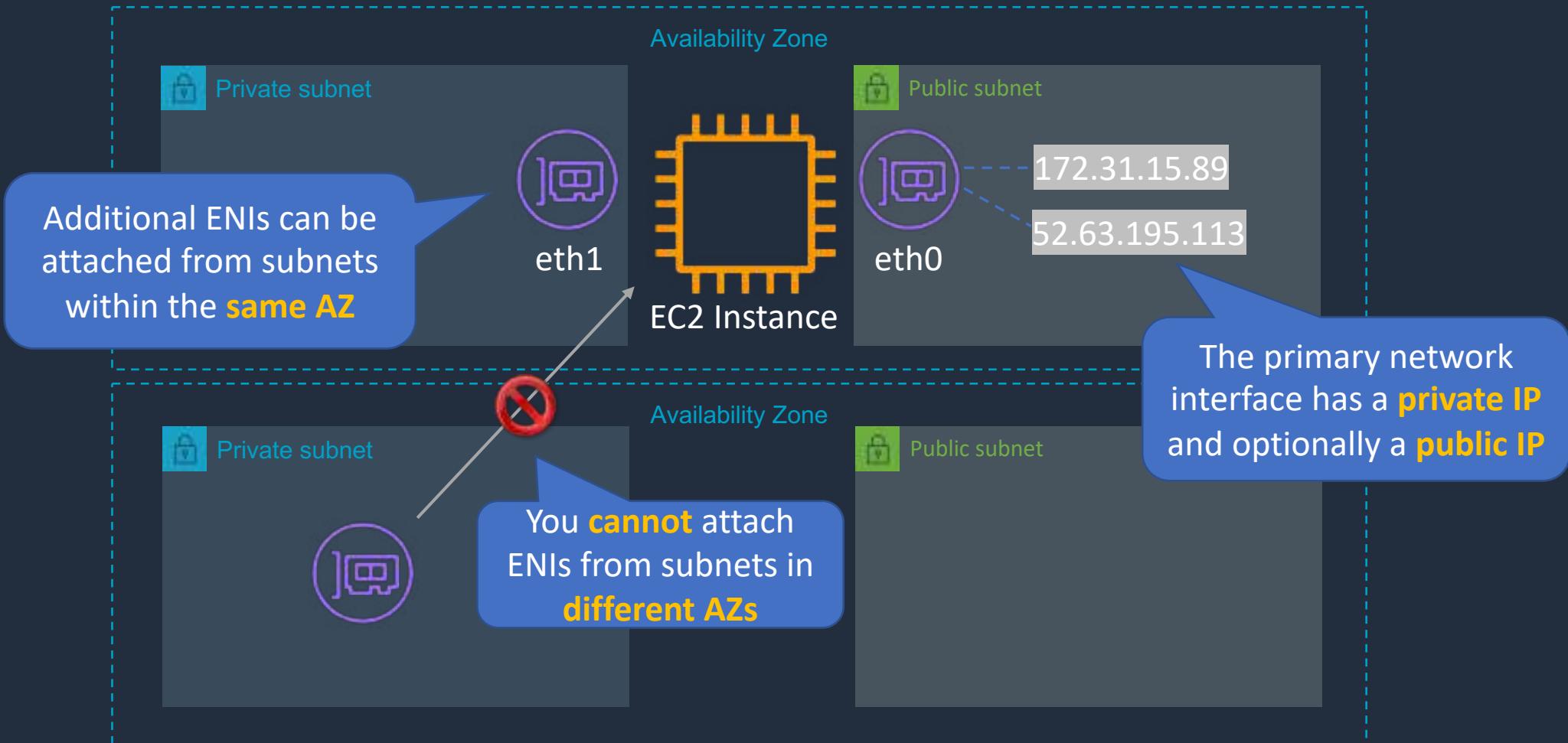


# Network Interfaces (ENI, ENA, EFA)





# Network Interfaces (ENI, ENA, EFA)





# Network Interfaces (ENI, ENA, EFA)

---



Elastic network  
interface

- Basic adapter type for when you don't have any high-performance requirements
- Can use with all instance types



Elastic network  
adapter

- Enhanced networking performance
- Higher bandwidth and lower inter-instance latency
- Must choose supported instance type



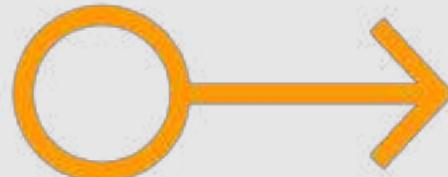
Elastic Fabric  
Adapter

- Use with High Performance Computing and MPI and ML use cases
- Tightly coupled applications
- Can use with all instance types

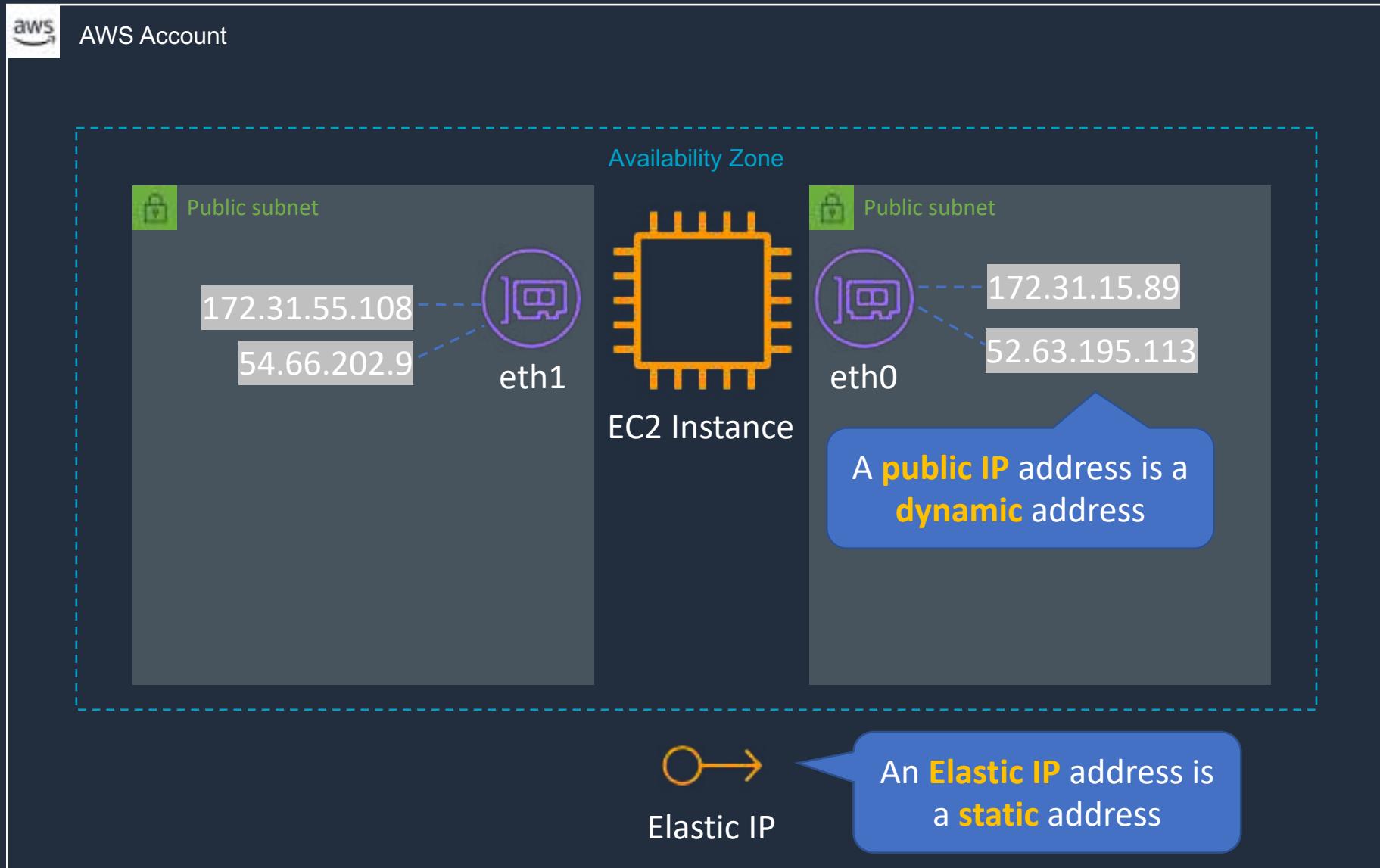
# Working with ENIs



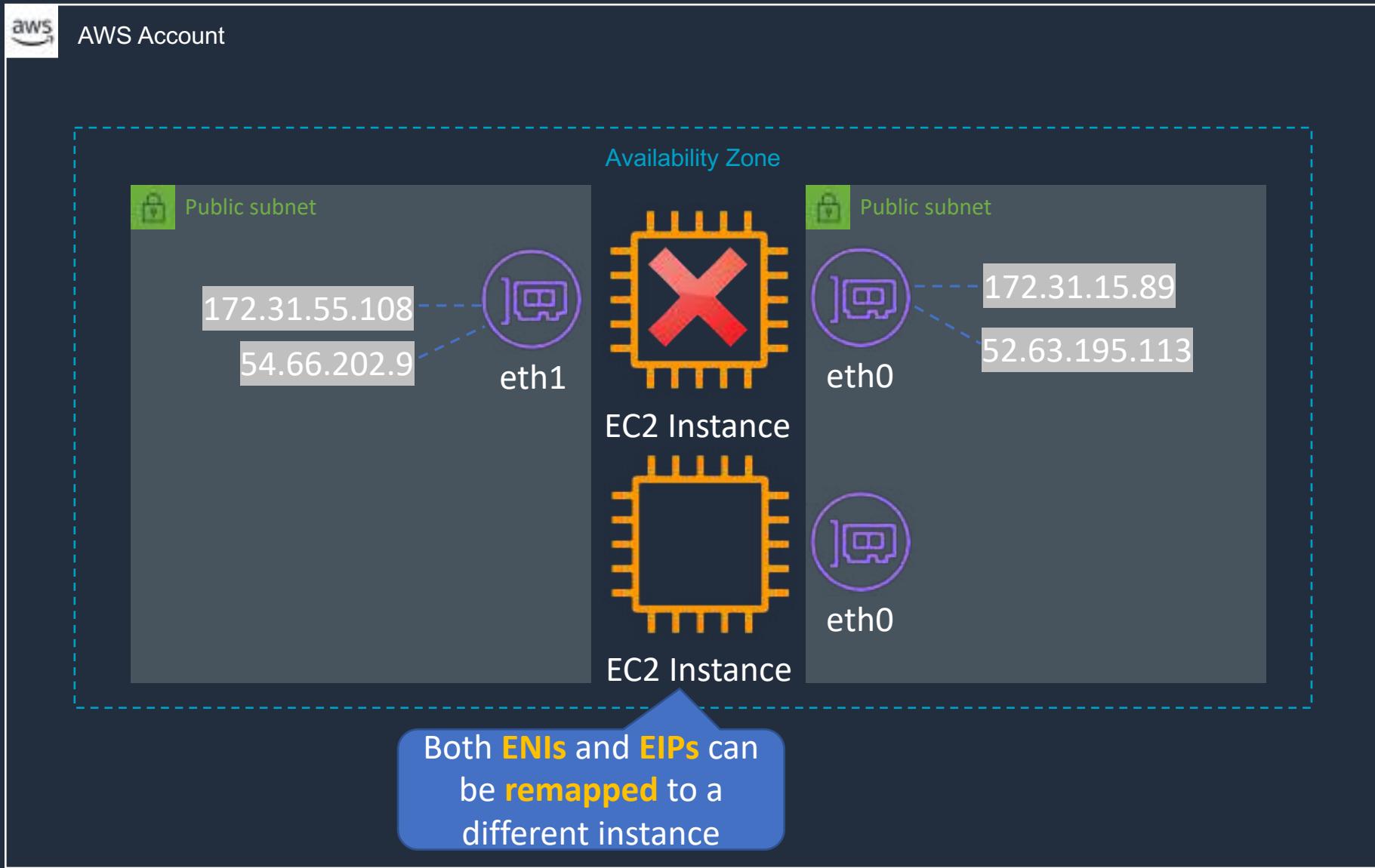
# Public, Private and Elastic IP Addresses



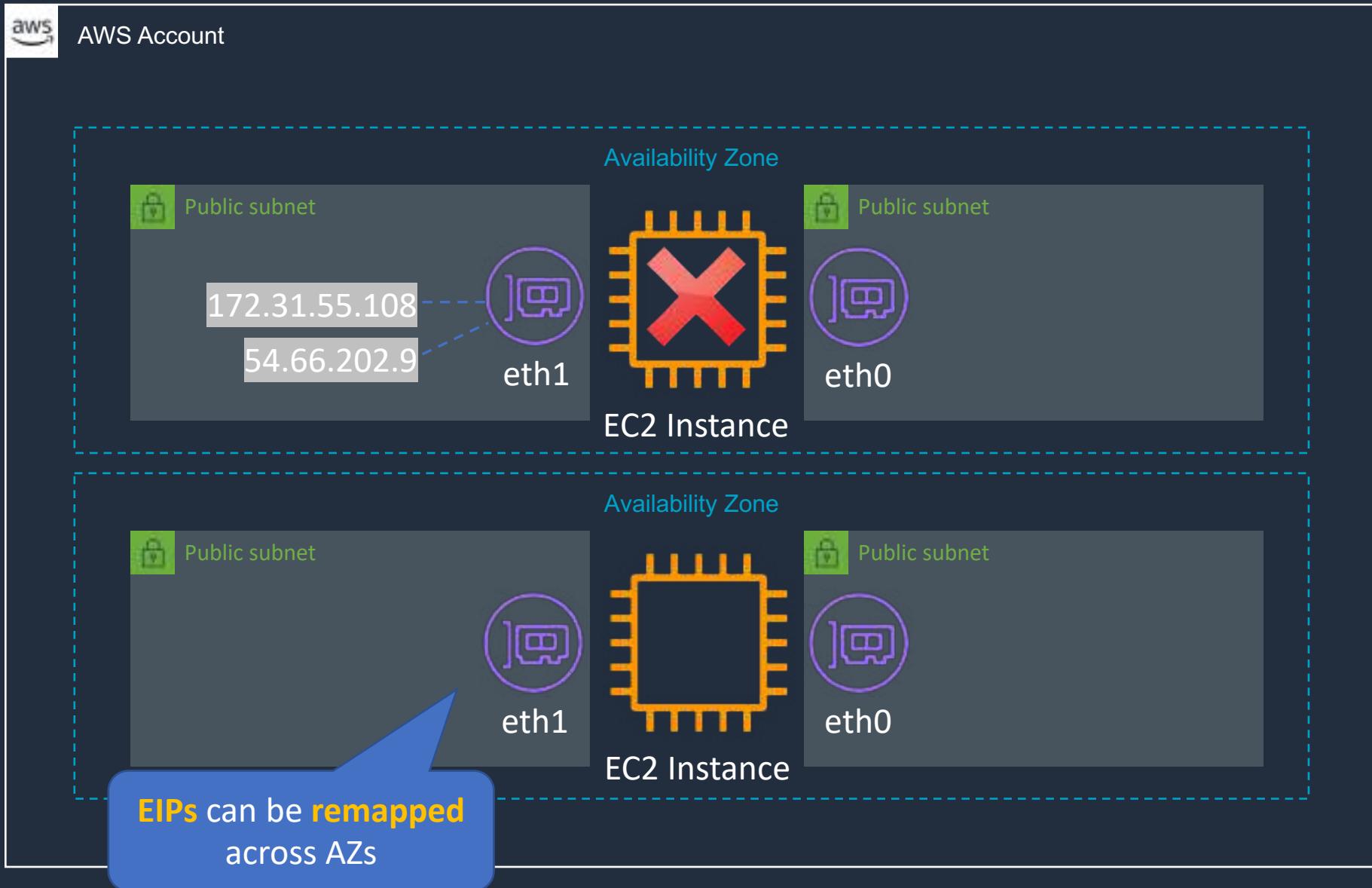
# → Public, Private and Elastic IP Addresses



# → Public, Private and Elastic IP Addresses



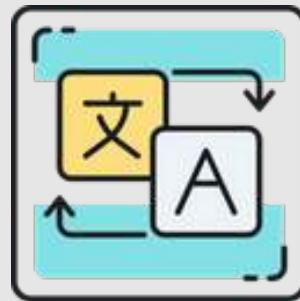
# → Public, Private and Elastic IP Addresses



# → Public, Private and Elastic IP addresses

Name	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>

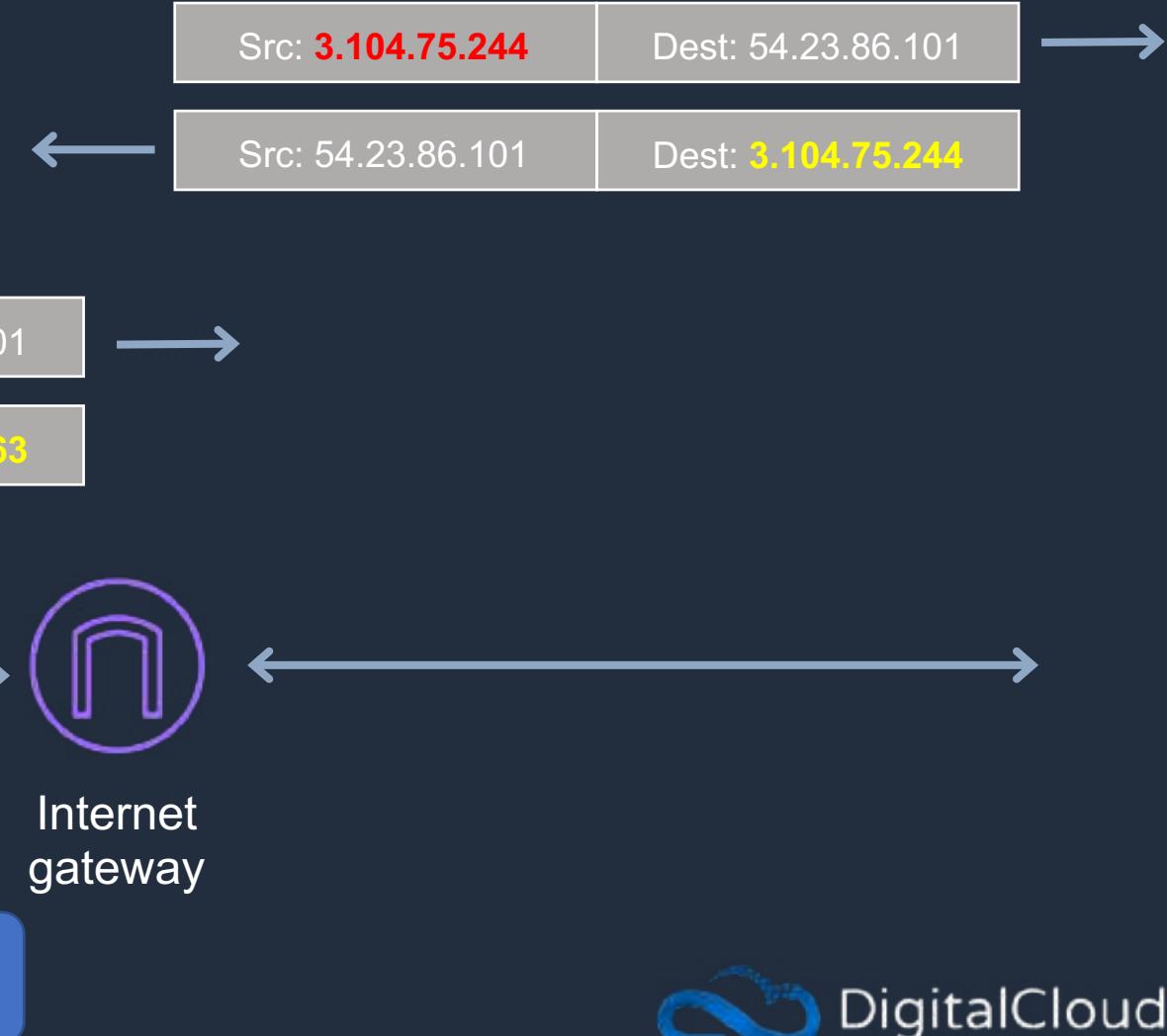
# NAT for Public Addresses





# NAT for Public Addresses

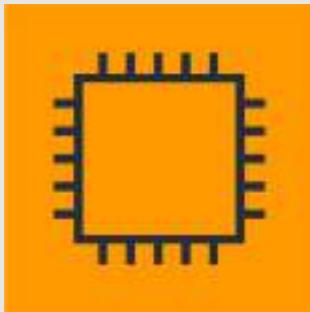
```
[ec2-user@ip-172-31-32-63 ~]$ ip addr show eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9001 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 06:06:dh:ad:56:28 brd ff:ff:ff:ff:ff:ff
    inet 172.31.32.63/20 brd 172.31.47.255 scope global dynamic eth0
        valid_lft 3330sec preferred_lft 3330sec
    inet6 fe80::406:dbff:fead:5628/64 scope link
        valid_lft forever preferred_lft forever
```



# Working with EC2 IP addresses

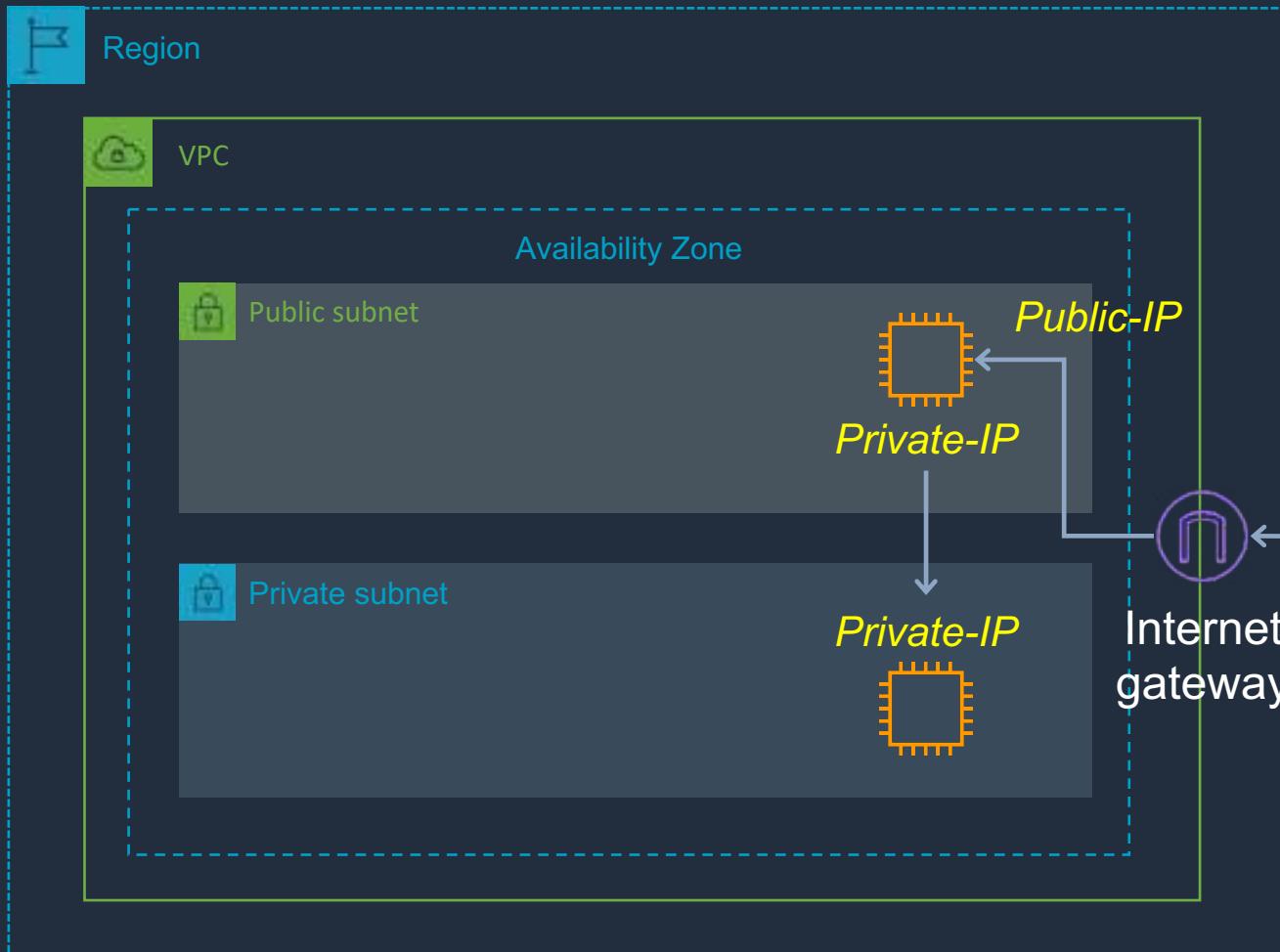


# Private Subnets and Bastion Hosts





# Private Subnets and Bastion Hosts



Public Subnet Route Table

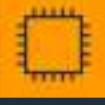
Destination	Target
172.31.0.0/16	Local
0.0.0.0/0	igw-id

Private Subnet Route Table

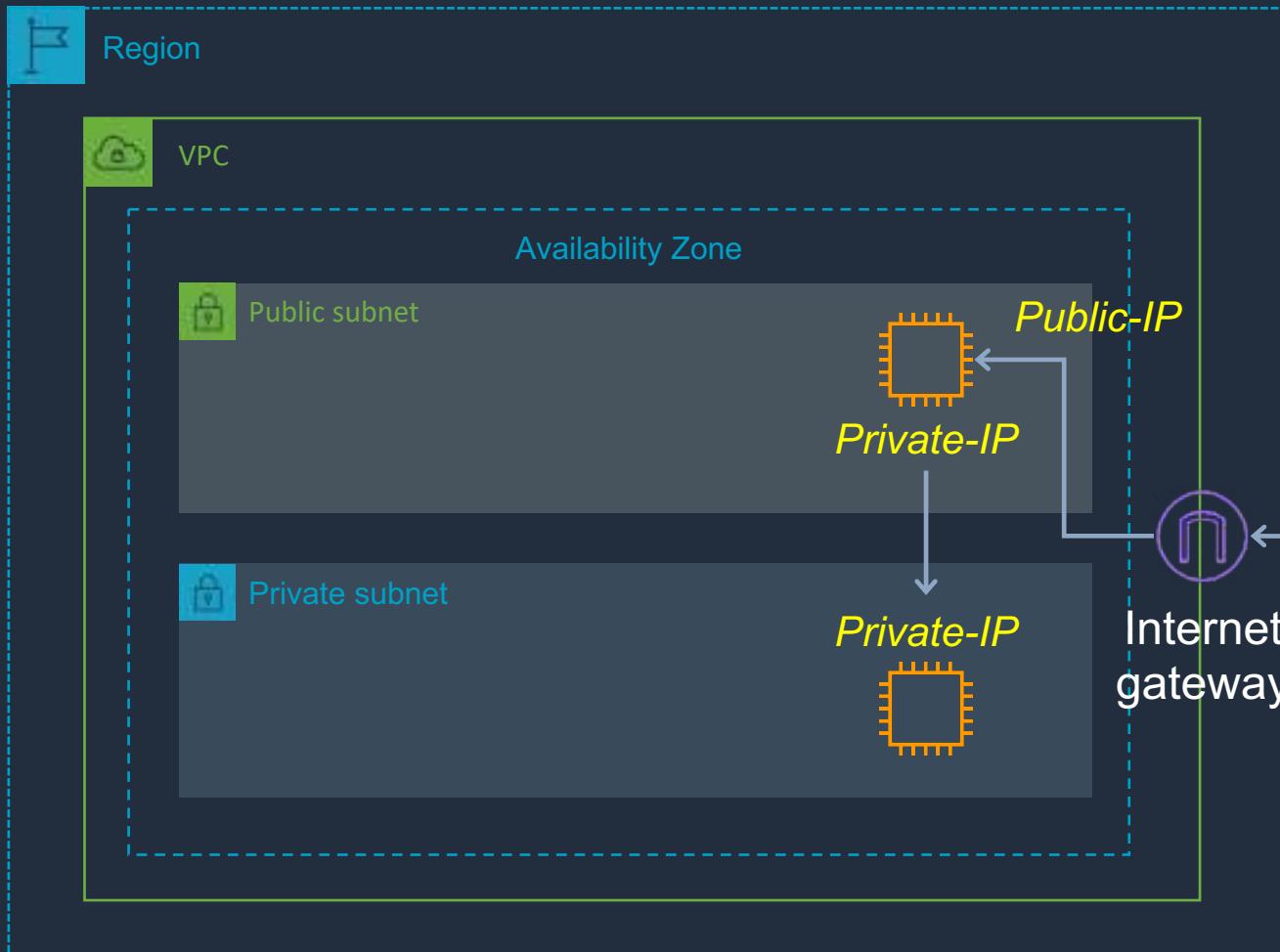
Destination	Target
172.31.0.0/16	Local

# Private Subnets and Bastion Hosts





# Private Subnets and Bastion Hosts



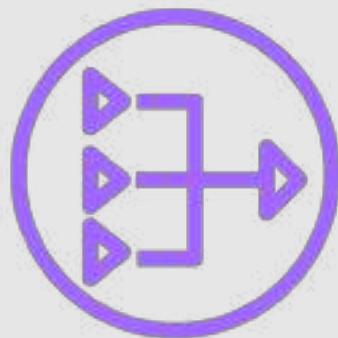
Public Subnet Route Table

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

Private Subnet Route Table

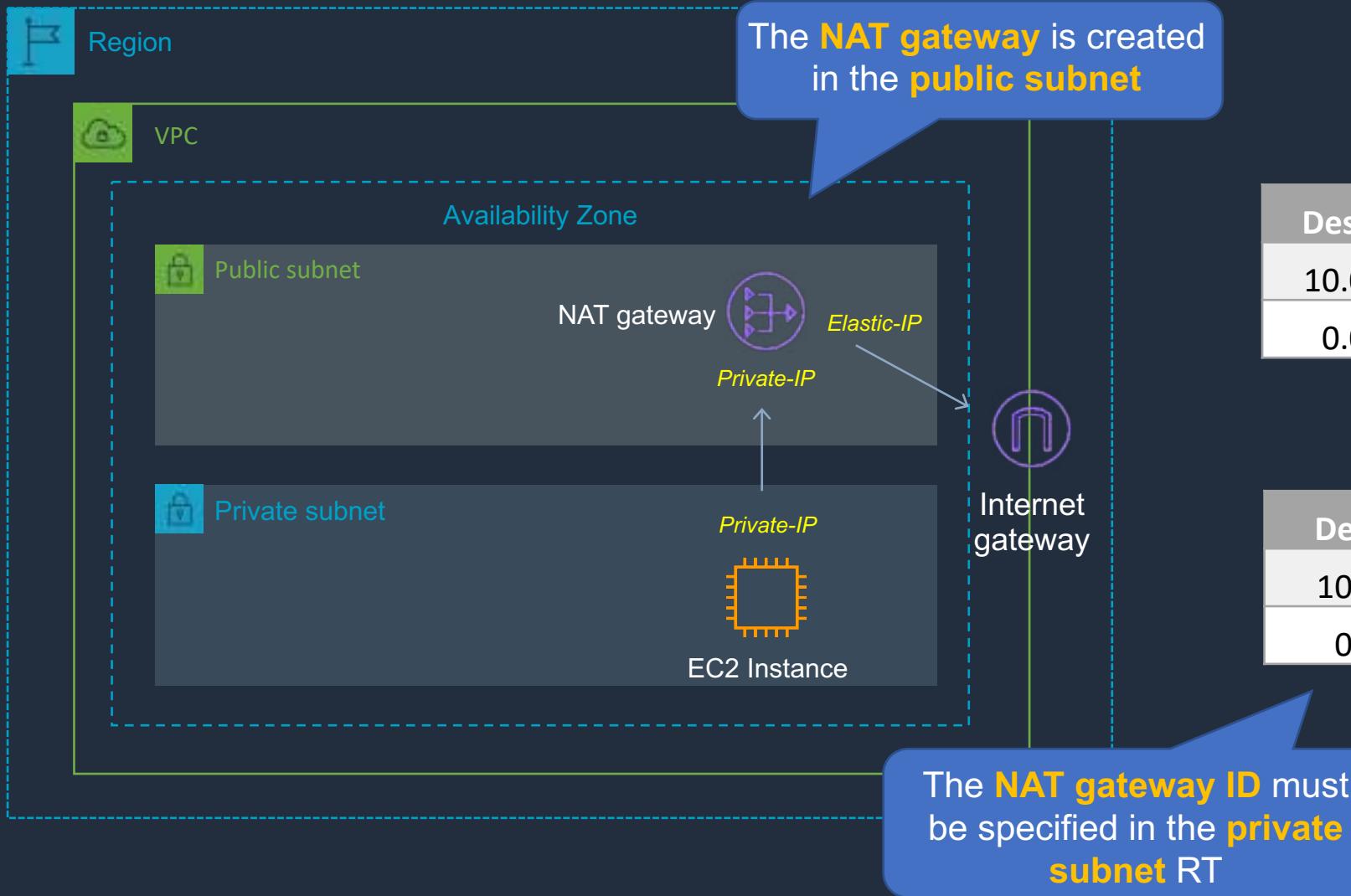
Destination	Target
172.31.0.0/16	Local

# NAT Gateways and NAT Instances Overview



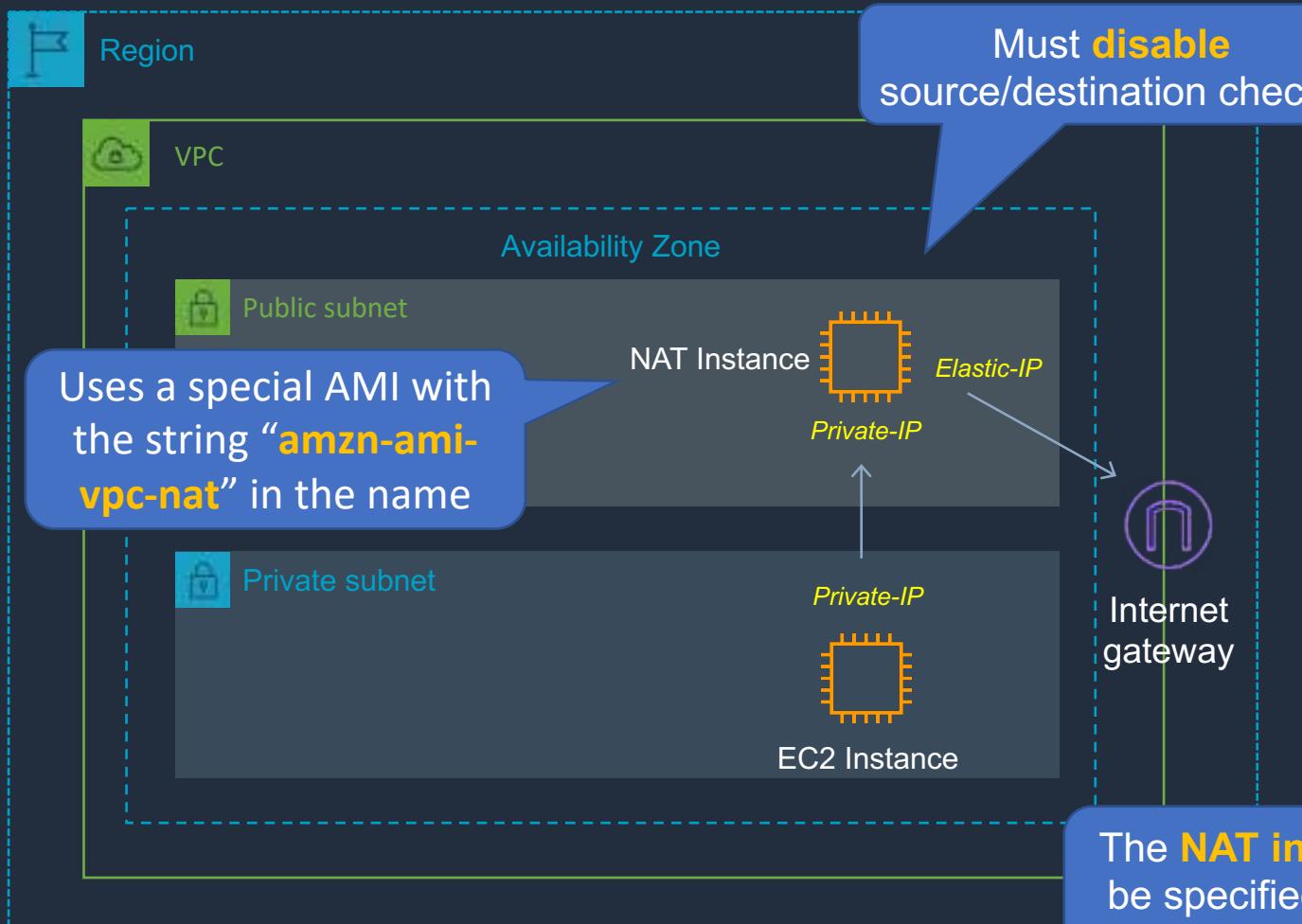


# NAT Gateways





# NAT Instances



Main Route Table

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

Private Route Table

Destination	Target
10.0.0.0/16	Local
0.0.0.0/0	nat-instance-id



# NAT Instance vs NAT Gateway

---

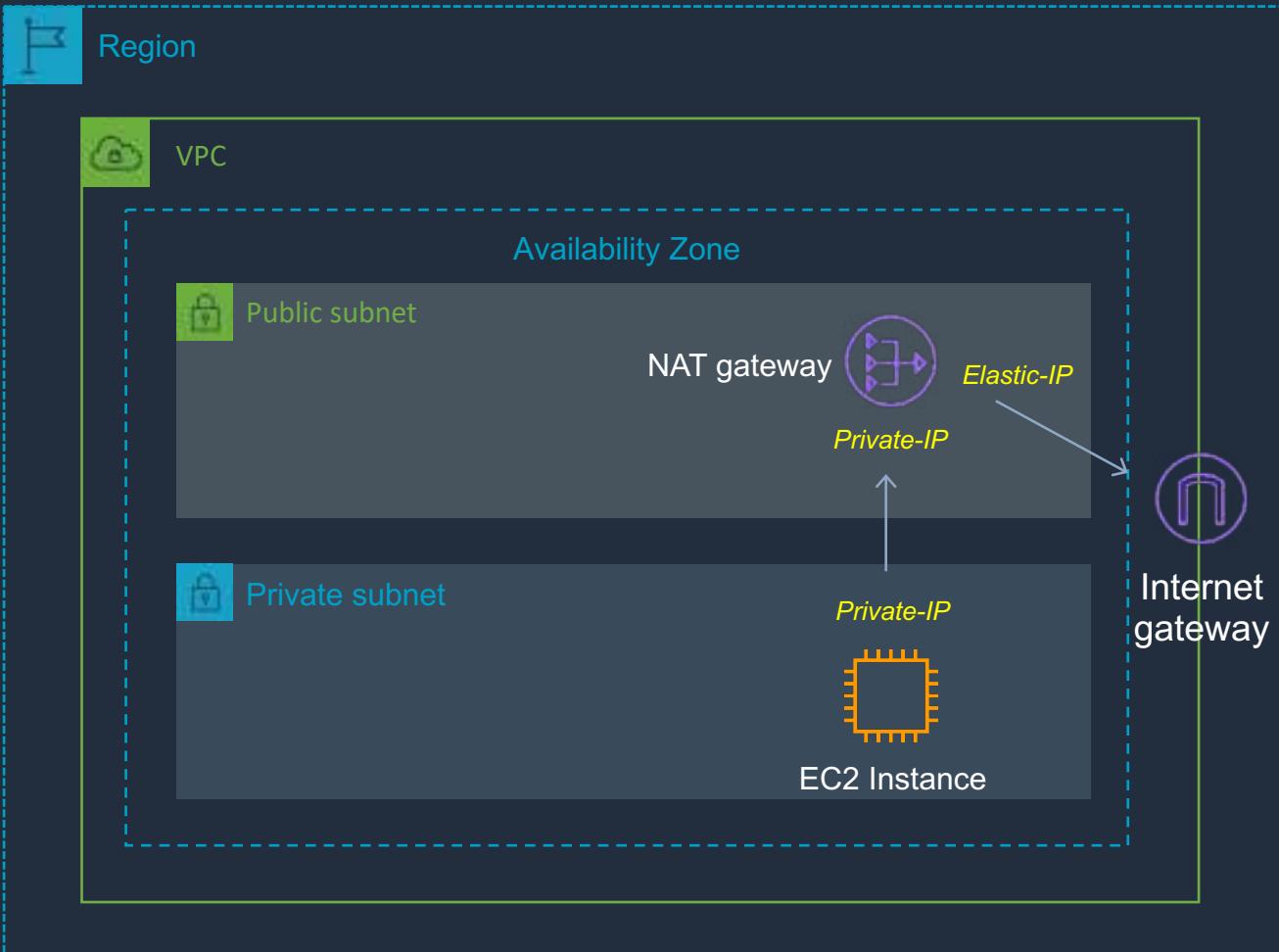
NAT Instance	NAT Gateway
Managed by you (e.g. software updates)	Managed by AWS
Scale up (instance type) manually and use enhanced networking	Elastic scalability up to 45 Gbps
No high availability – scripted/auto-scaled HA possible using multiple NATs in multiple subnets	Provides automatic high availability within an AZ and can be placed in multiple AZs
Need to assign Security Group	No Security Groups
Can use as a bastion host	Cannot access through SSH
Use an Elastic IP address or a public IP address with a NAT instance	Choose the Elastic IP address to associate with a NAT gateway at creation
Can implement port forwarding through manual customisation	Does not support port forwarding

# Private Subnet with NAT Gateway





# NAT Gateways



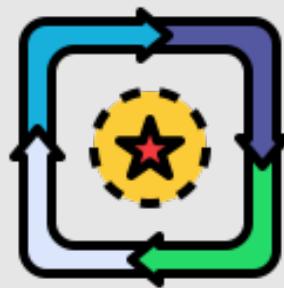
Main Route Table

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

Private Route Table

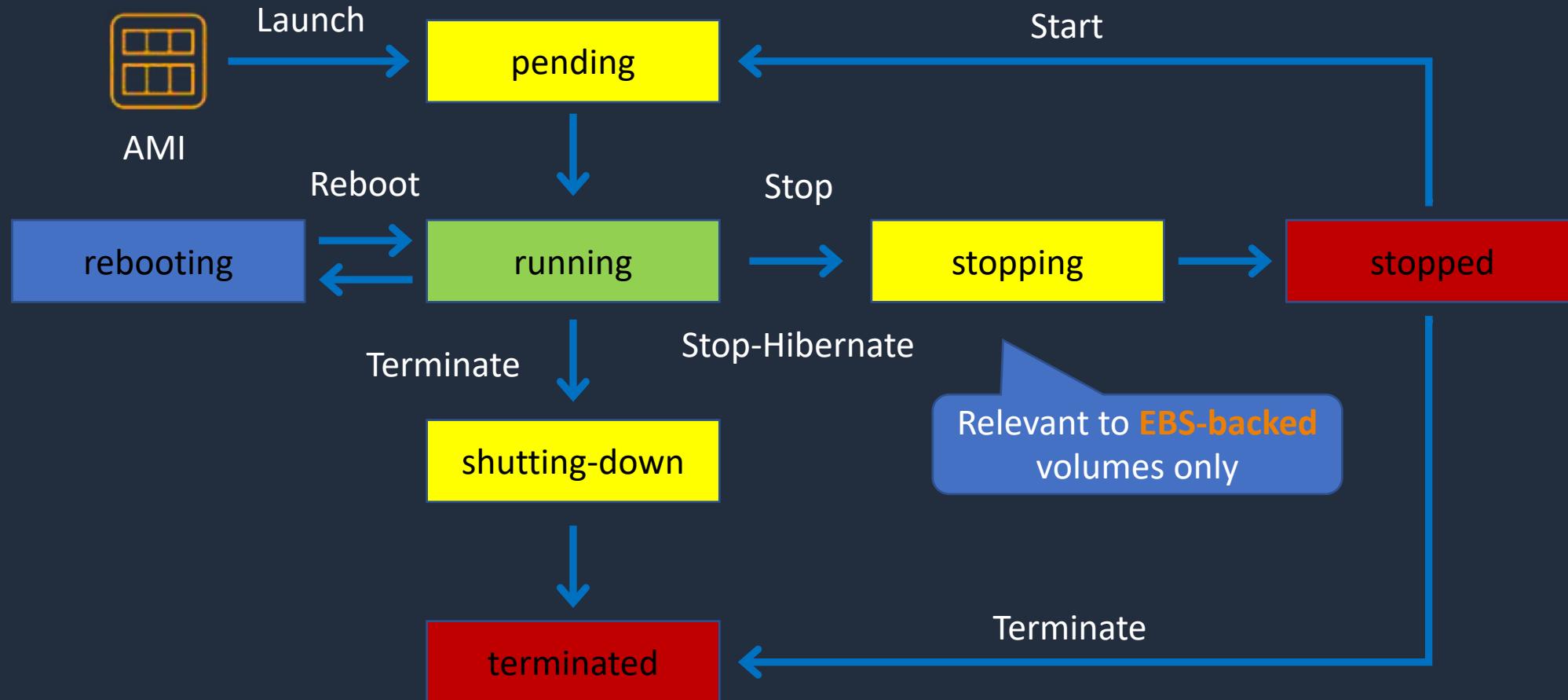
Destination	Target
172.31.0.0/16	Local
0.0.0.0/0	nat-gateway-id

# EC2 Instance Lifecycle





# EC2 Instance Lifecycle





# EC2 Instance Lifecycle

---

## Stopping EC2 instances

- EBS backed instances only
- No charge for stopped instances
- EBS volumes remain attached (chargeable)
- Data in RAM is lost
- Instance is migrated to a different host
- Private IPv4 addresses and IPv6 addresses retained; public IPv4 addresses released
- Associated Elastic IPs retained



# EC2 Instance Lifecycle

---

## Hibernating EC2 instances

- Applies to on-demand or reserved Linux instances
- Contents of RAM saved to EBS volume
- Must be enabled for hibernation when launched
- Specific prerequisites apply
- When started (after hibernation):
  - The EBS root volume is restored to its previous state
  - The RAM contents are reloaded
  - The processes that were previously running on the instance are resumed
  - Previously attached data volumes are reattached and the instance retains its instance ID



# EC2 Instance Lifecycle

---

## Rebooting EC2 instances

- Equivalent to an OS reboot
- DNS name and all IPv4 and IPv6 addresses retained
- Does not affect billing

## Retiring EC2 instances

- Instances may be retired if AWS detects irreparable failure of the underlying hardware that hosts the instance
- When an instance reaches its scheduled retirement date, it is stopped or terminated by AWS



# EC2 Instance Lifecycle

---

## Terminating EC2 instances

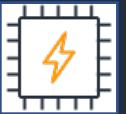
- Means deleting the EC2 instance
- Cannot recover a terminated instance
- By default root EBS volumes are deleted

## Recovering EC2 instances

- CloudWatch can be used to monitor system status checks and recover instance if needed
- Applies if the instance becomes impaired due to underlying hardware / platform issues
- Recovered instance is identical to original instance

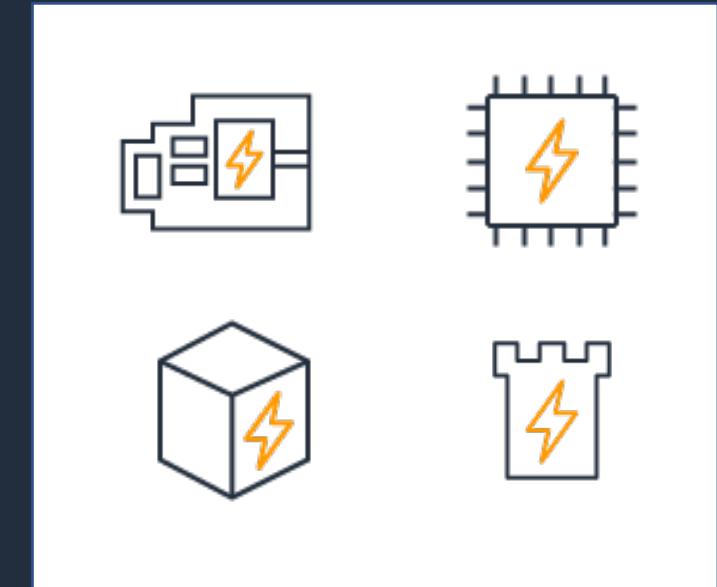
# Nitro Instances and Nitro Enclaves

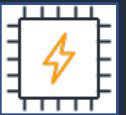




# AWS Nitro System

- Nitro is the underlying platform for the next generation of EC2 instances
- Support for many virtualized and bare metal instance types
- Breaks functions into specialized hardware with a Nitro Hypervisor
- Specialized hardware includes:
  - Nitro cards for VPC
  - Nitro cards for EBS
  - Nitro for Instance Storage
  - Nitro card controller
  - Nitro security chip
  - Nitro hypervisor
  - Nitro Enclaves





# AWS Nitro System

- Improves performance, security and innovation:
  - Performance close to bare metal for virtualized instances
  - Elastic Network Adapter and Elastic Fabric Adapter
  - More bare metal instance types
  - Higher network performance (e.g. 100 Gbps)
  - High Performance Computing (HPC) optimizations
  - Dense storage instances (e.g. 60 TB)



# AWS Nitro Enclaves

- Isolated compute environments
- Runs on isolated and hardened virtual machines
- No persistent storage, interactive access, or external networking
- Uses cryptographic attestation to ensure only authorized code is running
- Integrates with AWS Key Management Service (KMS)
- Protect and securely process highly sensitive data:
  - Personally identifiable information (PII)
  - Healthcare data
  - Financial data
  - Intellectual Property data

# Amazon EC2 Pricing Options





# Amazon EC2 Pricing Options

## On-Demand

Standard rate - no discount; no commitments; dev/test, short-term, or unpredictable workloads

## Spot Instances

Bid for unused capacity; up to 90% discount; can be terminated at any time; workloads with flexible start and end times

## Dedicated Hosts

Physical server dedicated for your use; Socket/core visibility, host affinity; pay per host; workloads with server-bound software licenses

## Reserved

1 or 3-year commitment; up to 75% discount; steady-state, predictable workloads and reserved capacity

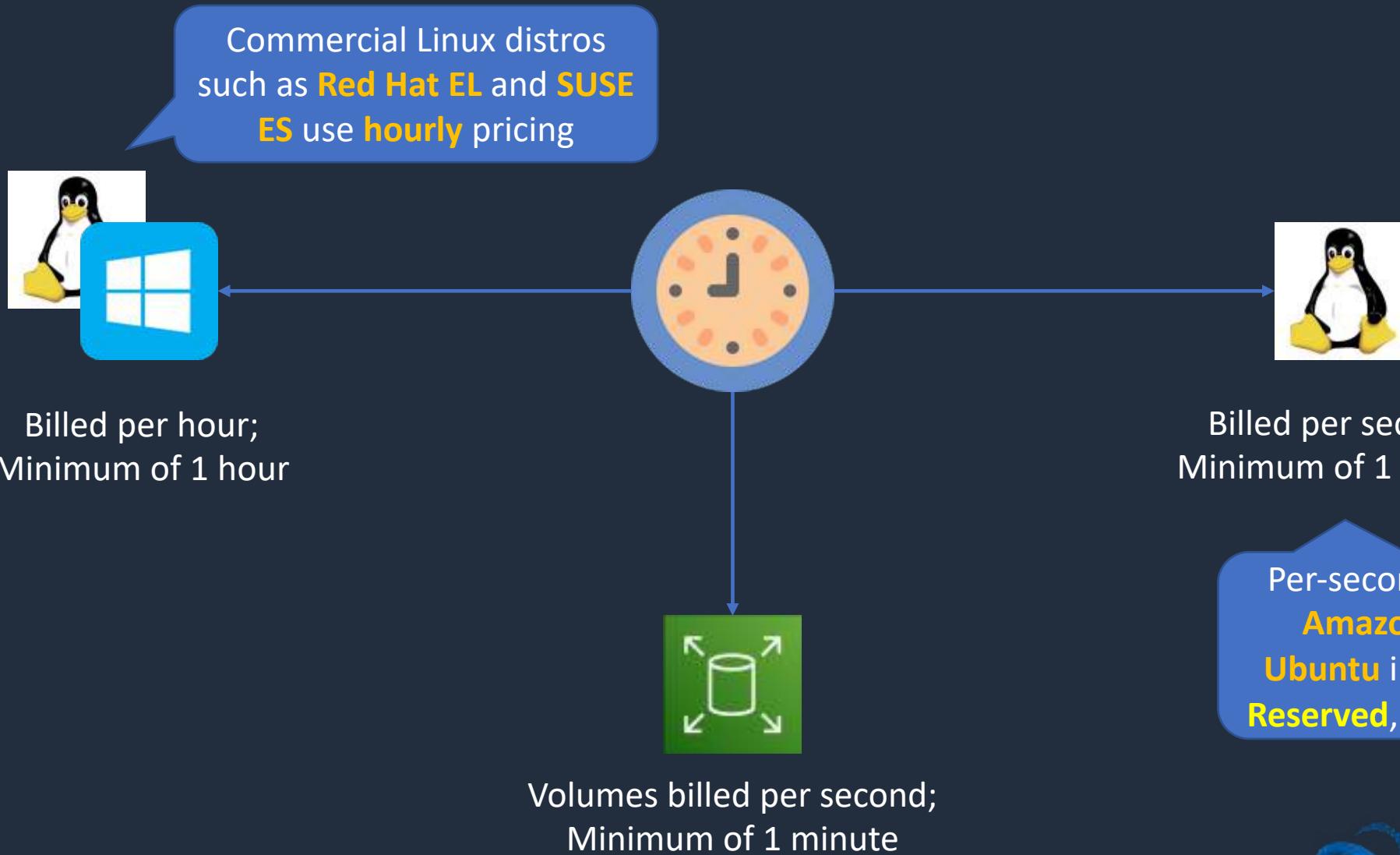
## Dedicated Instances

Physical isolation at the host hardware level from instances belonging to other customers; pay per instance

## Savings Plans

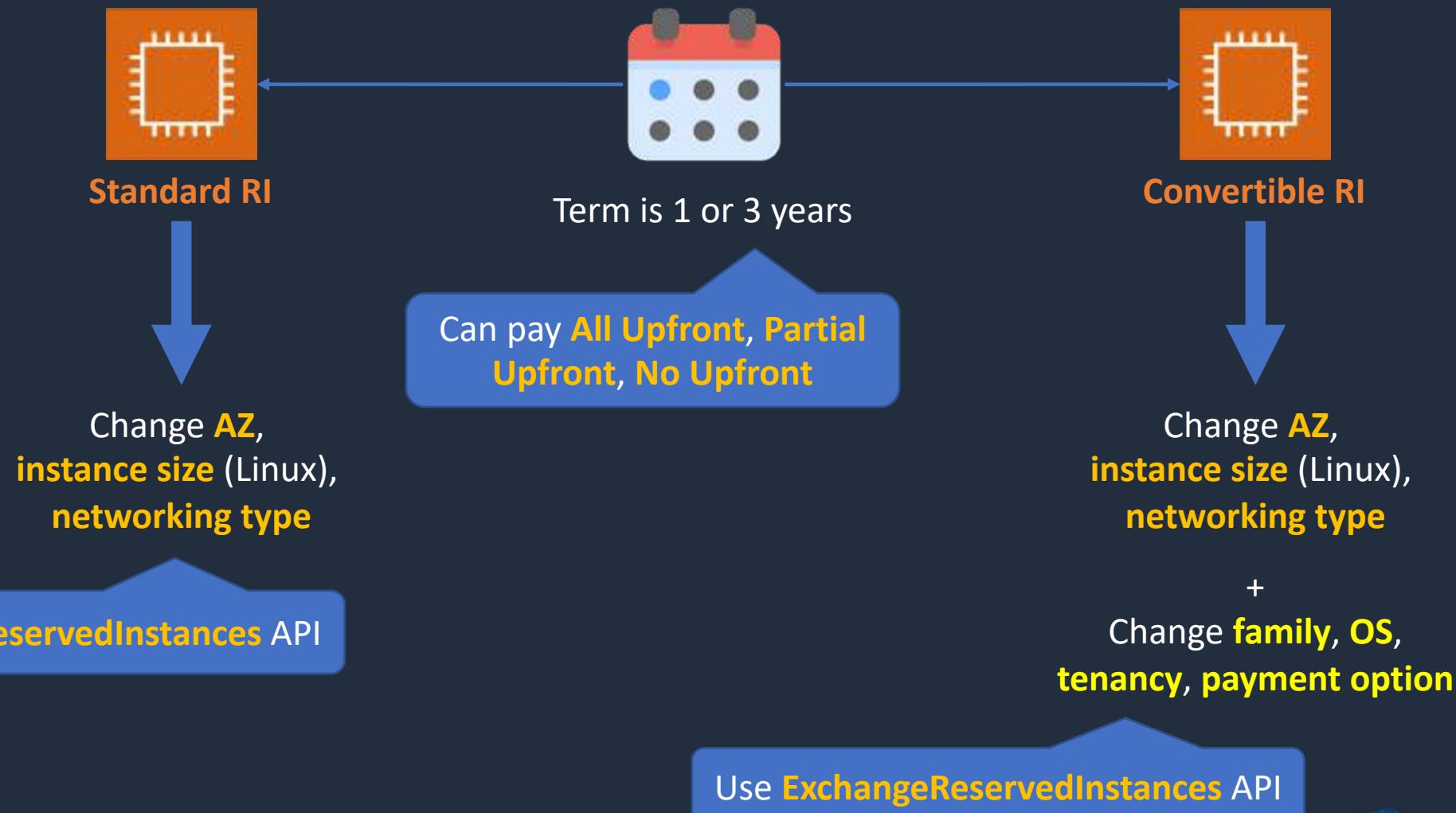
Commitment to a consistent amount of usage (EC2 + Fargate + Lambda); Pay by \$/hour; 1 or 3-year commitment

# \$ Amazon EC2 Billing





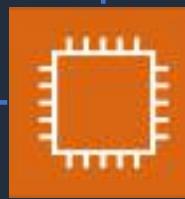
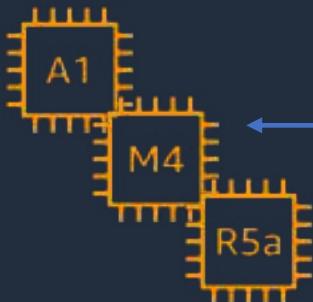
# Amazon EC2 Reserved Instances (RIs)





# Amazon EC2 Reserved Instances (RIs)

Tenancy: **Default** or **Dedicated**



When the **attributes** of a used instance  
match the **attributes** of an RI the  
discount is applied

Reserves capacity  
in specified AZ

Availability Zone



Region

Does **not** reserve  
capacity; discount  
applies to all AZs



# Amazon EC2 Reserved Instances (RIs)



Scheduled RI

- Match capacity reservation to **recurring schedule**
- Minimum **1200 hours** per year
- Example: Reporting app that runs 6 hours a day 4 days a week = 1248 hours per year



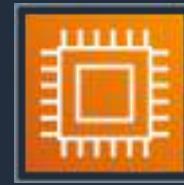
#### Important

We do not have any capacity for purchasing Scheduled Reserved Instances or any plans to make it available in the future. To reserve capacity, use [On-Demand Capacity Reservations](#) instead. For discounted rates, use [Savings Plans](#).

This message started showing recently but exam may **not** reflect this yet

# \$ AWS Savings Plans

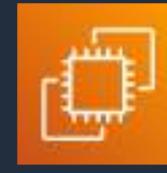
---



## Compute Savings Plan



1 or 3-year; hourly commitment to usage of **Fargate**, **Lambda**, and **EC2**; Any Region, family, size, tenancy, and OS



## EC2 Savings Plan



1 or 3-year; hourly commitment to usage of **EC2** within a **selected Region** and **Instance Family**; Any size, tenancy and OS



# Amazon EC2 Spot Instances



**Bid** for unused capacity at up to 90% discount



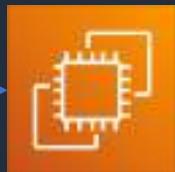
**2-minute warning** if AWS need to reclaim capacity – available via **instance metadata** and **CloudWatch Events**



**Spot Instance:** One or more EC2 instances



**Spot Fleet:** launches and maintains the number of Spot / On-Demand instances to meet specified target capacity



**EC2 Fleet:** launches and maintains specified number of Spot / On-Demand / Reserved instances in a **single API call**

Can define separate OD/Spot **capacity targets, bids, instance types, and AZs**



# Spot Block



Requirement:  
Uninterrupted for  
1-6 hours

Pricing is **30% - 45%** less  
than On-Demand

Solution: **Spot Block**

```
$ aws ec2 request-spot-instances \
--block-duration-minutes 360 \
--instance-count 5 \
--spot-price "0.25" ...
```



# Dedicated Instances and Dedicated Hosts

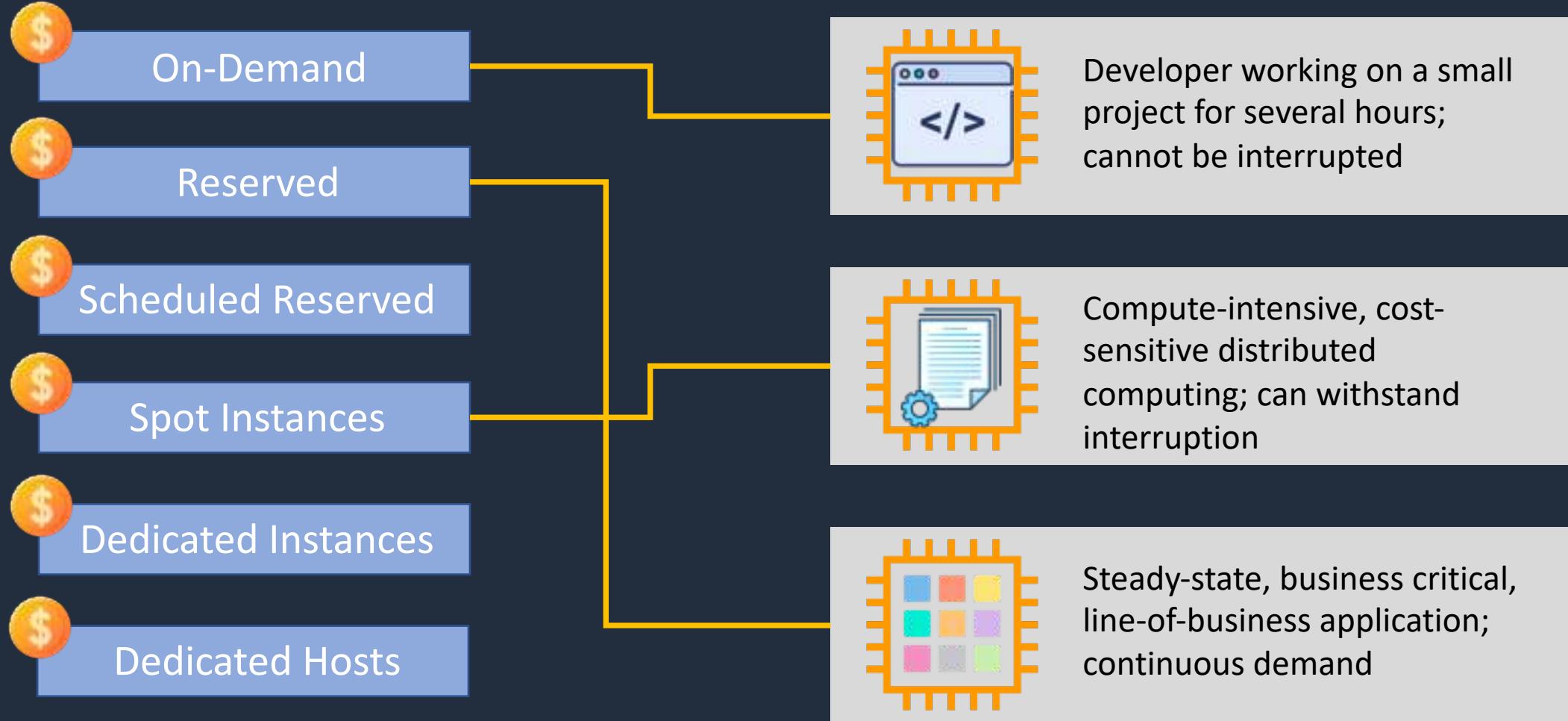
Characteristic	Dedicated Instances	Dedicated Hosts
Enables the use of dedicated physical servers	X	X
Per instance billing (subject to a \$2 per region fee)	X	
Per host billing		X
Visibility of sockets, cores, host ID		X
Affinity between a host and instance		X
Targeted instance placement		X
Automatic instance placement	X	X
Add capacity using an allocation request		X

# Amazon EC2 Pricing Use Cases





# Amazon EC2 Pricing Use Cases





# Amazon EC2 Pricing Use Cases



On-Demand



Reserved



Scheduled Reserved



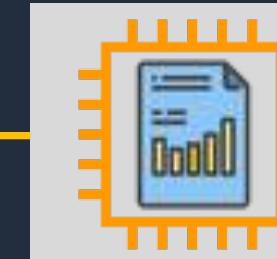
Spot Instances



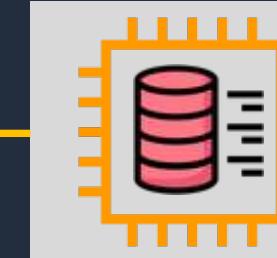
Dedicated Instances



Dedicated Hosts



Reporting application, runs for 6 hours a day, 4 days per week

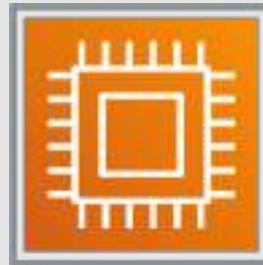


Database with per-socket licensing



Security-sensitive application, requires dedicated hardware; per-instance billing

# Architecture Patterns – Amazon EC2





# Architecture Patterns – Amazon EC2

---

## Requirement

Company needs to run a short batch script to configure Amazon EC2 Linux instances after they are launched

A tightly coupled High Performance Computing (HPC) workload requires low-latency between nodes and optimum network performance

LoB application receives weekly bursts of traffic and must scale for short periods – need the most cost-effective solution

## Solution

Add the bash script to the user data of the EC2 instances

Launch EC2 instances in a single AZ in a cluster placement group and use an Elastic Fabric Adapter (EFA)

Use reserved instances for minimum required workload and then use Spot instances for the bursts in traffic



# Architecture Patterns - Amazon EC2

---

---

## Requirement

A single instance application uses a static public IP address. In the event of failure, the address must be remapped to a failover instance

A fleet of Amazon EC2 instances run in private subnets across multiple AZs. Company needs a redundant path to the internet

A team of engineers must administer EC2 instances in private subnets from remote locations using SSH

## Solution

Attach an Elastic IP address to the EC2 instance. Remap the EIP in the event of failure

Deploy NAT Gateways into multiple AZs and update route tables

Deploy a bastion host in a public subnet and instruct the engineers to use the bastion host to “jump” to the instances in private subnets



# Architecture Patterns - Amazon EC2

---

---

## Requirement

An application uses several EC2 instances. Architect must eliminate the risk of correlated hardware failures

## Solution

Launch the instances in a spread placement group across distinct underlying hardware

Application requires enhanced networking capabilities

Choose an instance type that supports enhanced networking and ensure the ENA module is installed and ENA support is enabled

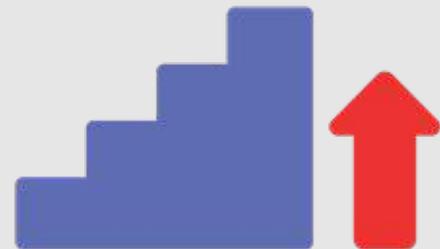
Instance needs close to bare metal performance, EFA, and high performance networking

Use an AWS Nitro instance type

# SECTION 4

## Elastic Load Balancing, and Auto Scaling

# Elasticity: Scaling Up vs Out





# Scaling Up (vertical scaling)

---





# Scaling Up (vertical scaling)

Scaling up means  
**adding** resources  
to the instance



Limitation is that you  
have a **single point of  
failure** (SPOF)



# Scaling Out (horizontal scaling)

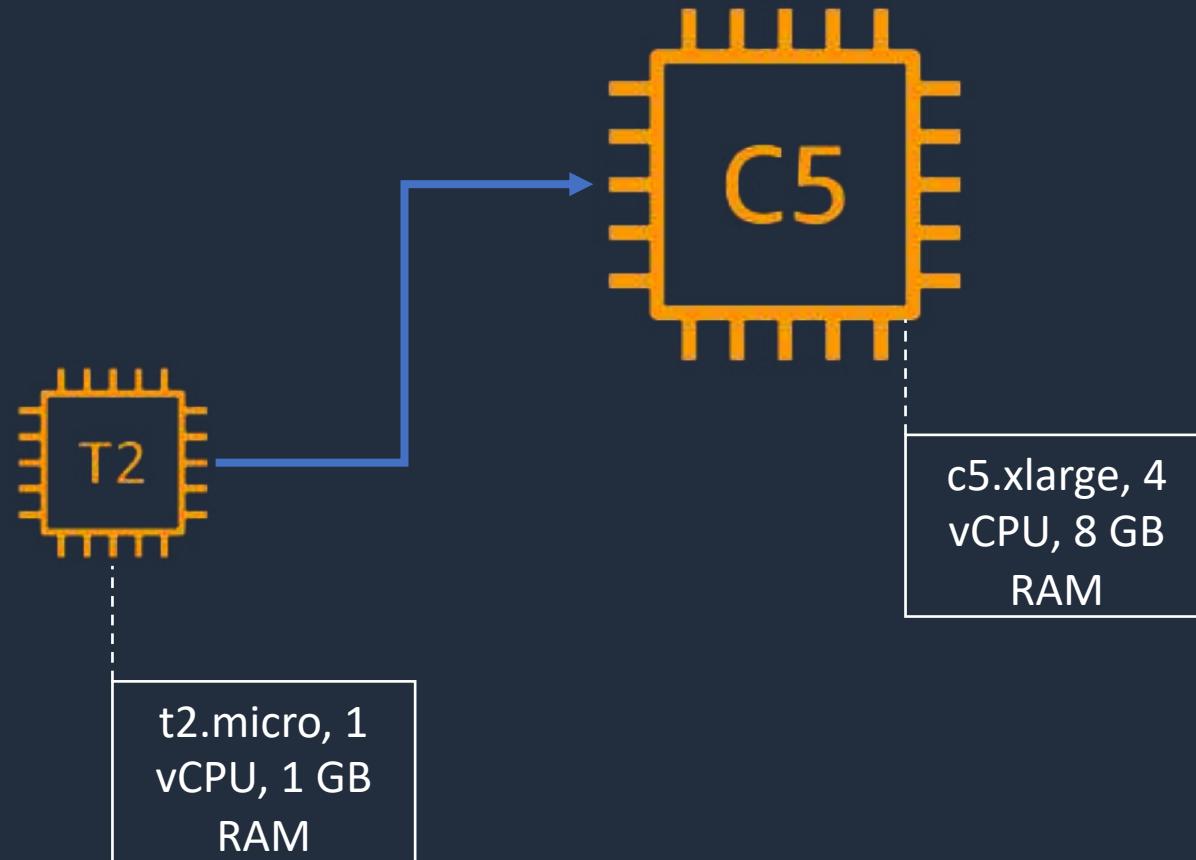
Scaling out provides greater **resiliency**



Scaling out can be used to add almost unlimited capacity



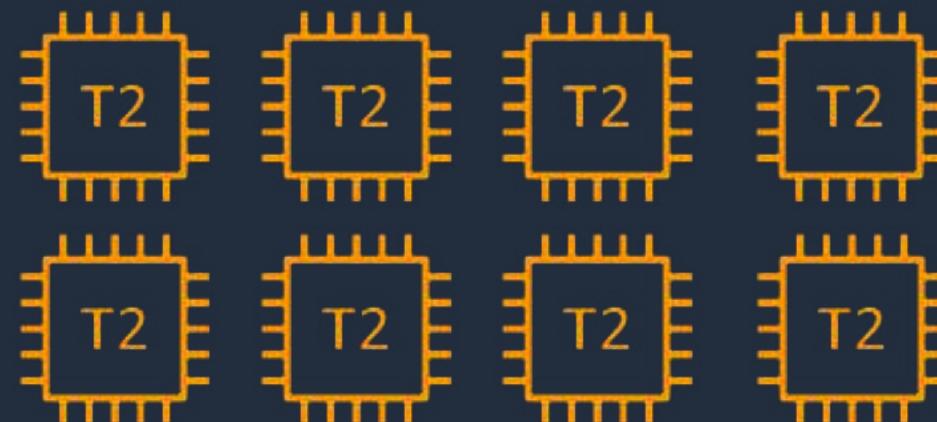
# Scaling Up (vertical scaling)



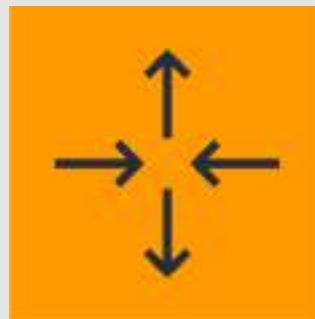


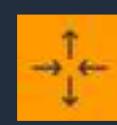
# Scaling Out (horizontal scaling)

---

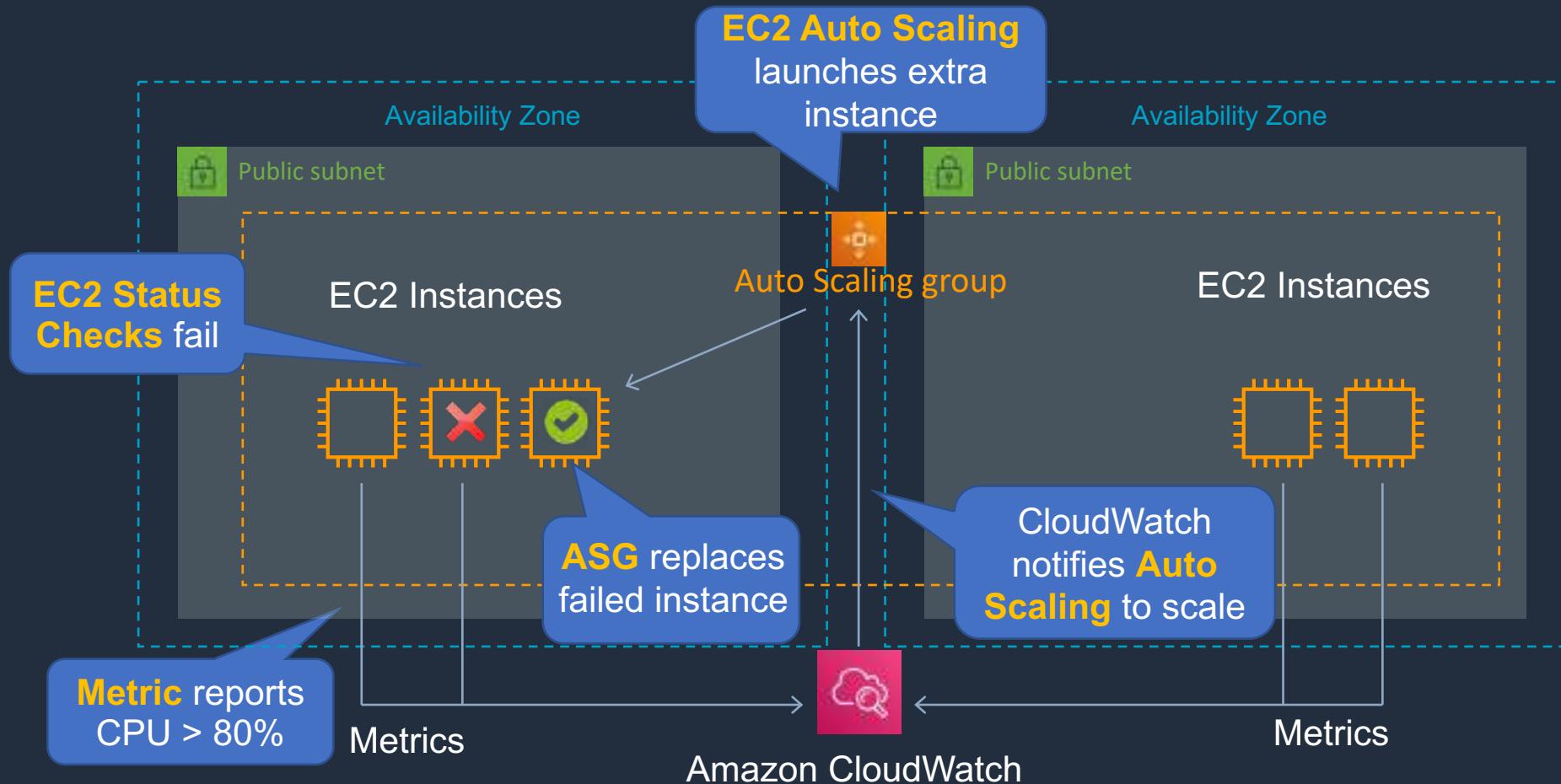


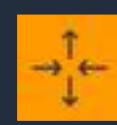
# Amazon EC2 Auto Scaling





# Amazon EC2 Auto Scaling

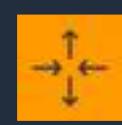




# Amazon EC2 Auto Scaling

---

- EC2 Auto Scaling **launches** and **terminates** instances dynamically
- 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
- Auto Scaling groups define collections of EC2 instances that are scaled and managed together



# Configuration of an Auto Scaling Group

A **Launch Template** specifies the EC2 instance configuration



Launch Template

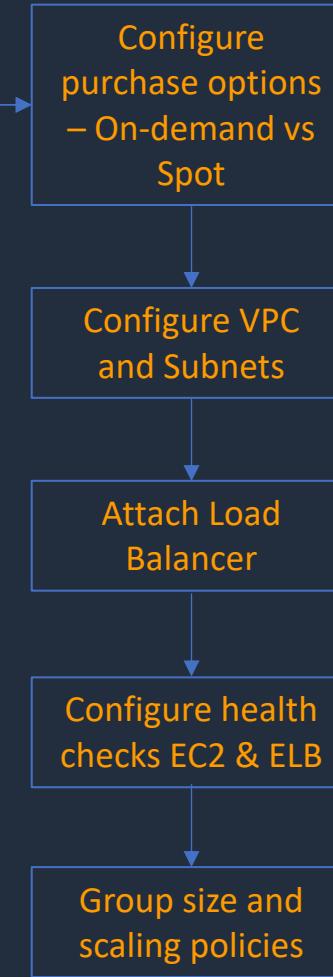
- AMI and instance type
- EBS volumes
- Security groups
- Key pair
- IAM instance profile
- User data
- Shutdown behavior
- Termination protection
- Placement group name
- Capacity reservation
- Tenancy
- Purchasing option (e.g. Spot)

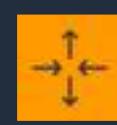


Launch Config

- 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

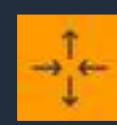




# Amazon EC2 Auto Scaling

---

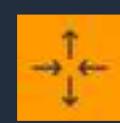
- Health checks
  - EC2 = EC2 status checks
  - ELB = Uses the ELB health checks **in addition** to EC2 status checks
- Health check grace period
  - How long to wait before checking the health status of the instance
  - Auto Scaling does not act on health checks until grace period expires



# Auto Scaling - Monitoring

---

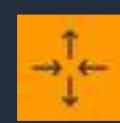
- **Group metrics (ASG)**
  - Data points about the Auto Scaling group
  - 1-minute granularity
  - No charge
  - Must be enabled
- **Basic monitoring (Instances)**
  - 5-minute granularity
  - No Charge
- **Detailed monitoring (Instances)**
  - 1-minute granularity
  - Charges apply



# Additional Scaling Settings

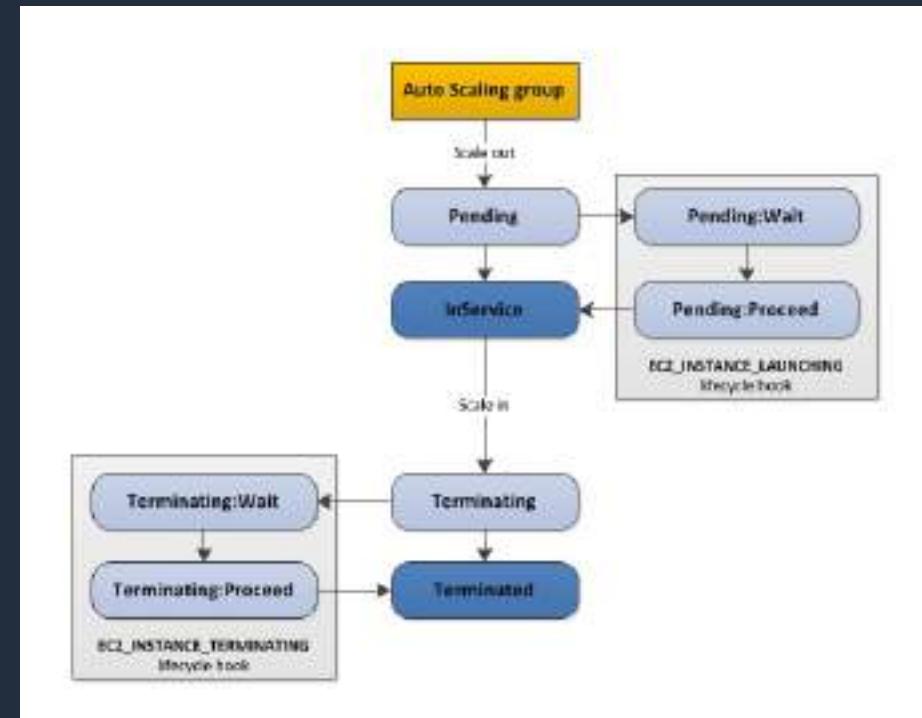
---

- **Cooldowns** – Used with simple scaling policy to prevent Auto Scaling from launching or terminating before effects of previous activities are visible. Default value is 300 seconds (5 minutes)
- **Termination Policy** – Controls which instances to terminate first when a scale-in event occurs
- **Termination Protection** – Prevents Auto Scaling from terminating protected instances
- **Standby State** – Used to put an instance in the **InService** state into the **Standby** state, update or troubleshoot the instance



# Additional Scaling Settings

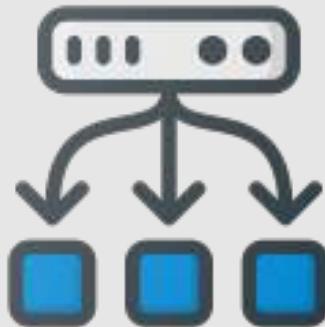
- **Lifecycle Hooks** – Used to perform custom actions by pausing instances as the ASG launches or terminates them
- Use case:
  - Run a script to download and install software after launching
  - Pause an instance to process data before a scale-in (termination)



# Create an Auto Scaling Group

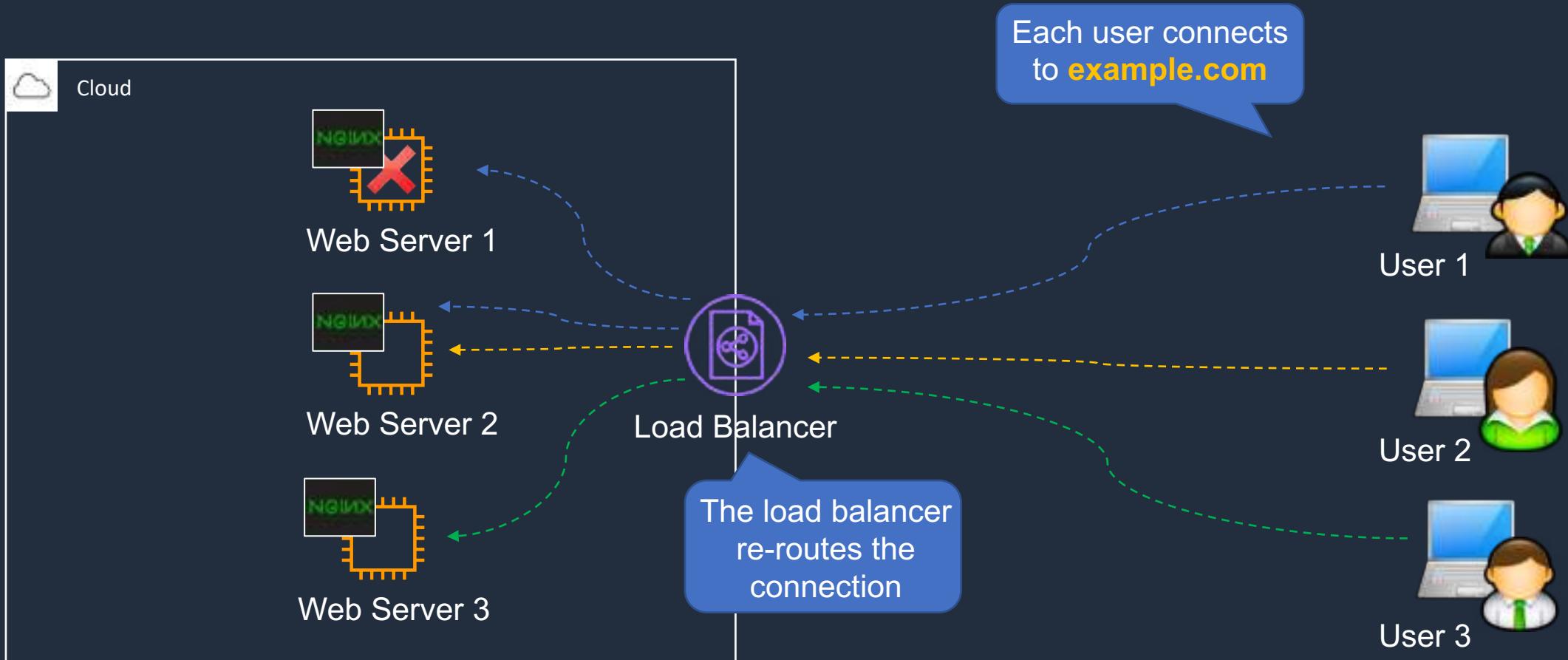


# Load Balancing and High Availability





# Load Balancing and High Availability





# Fault Tolerance

---

**Redundant** components allow the system to continue to operate

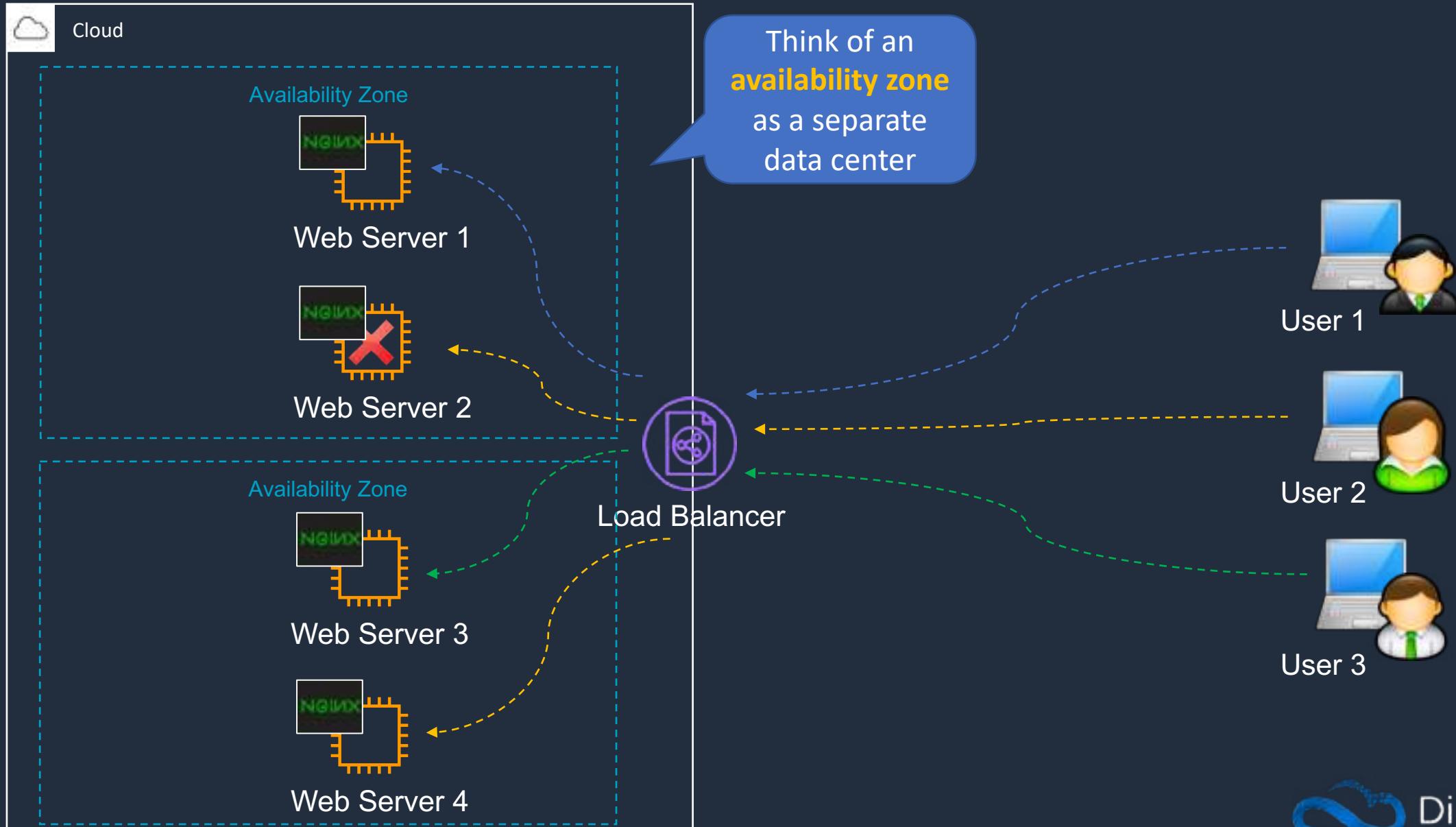


Network Card

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

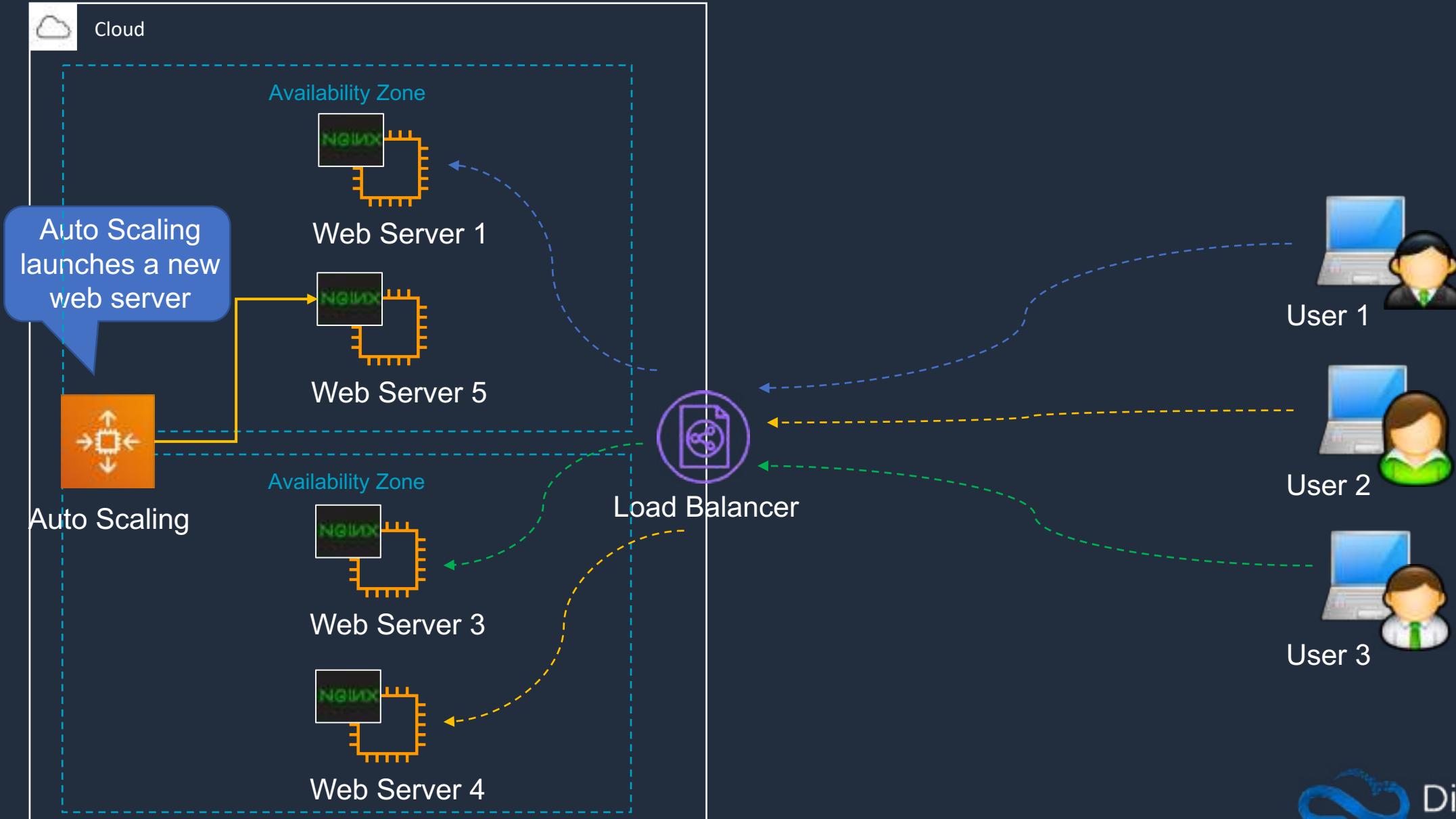


# High Availability and Fault Tolerance



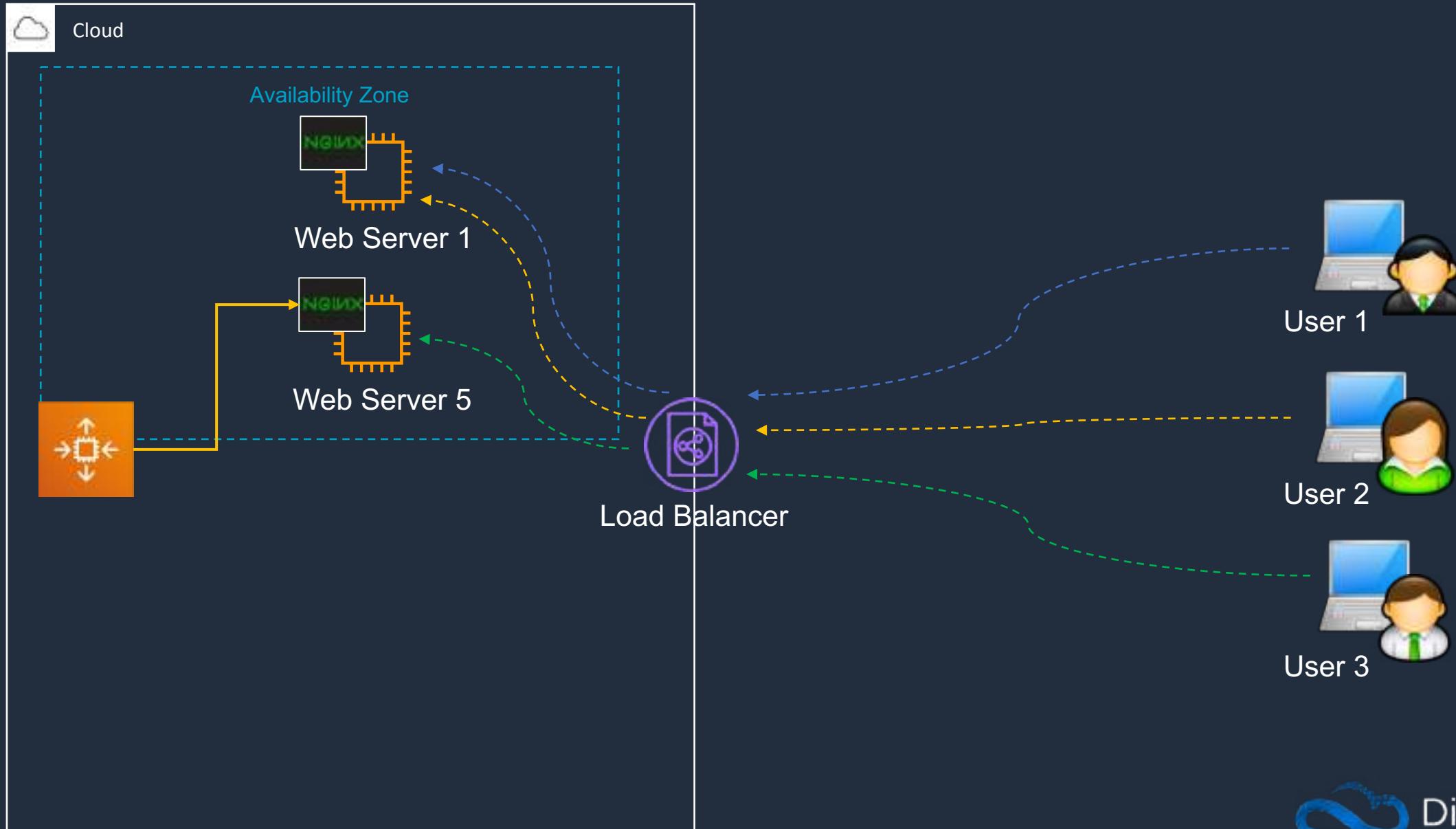


# High Availability and Fault Tolerance





# High Availability and Fault Tolerance

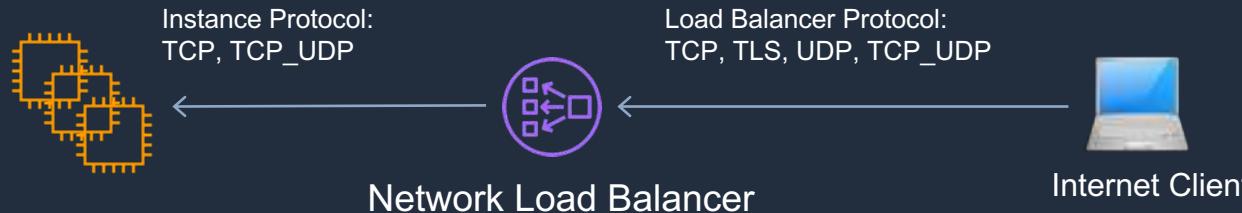


# Types of Elastic Load Balancer (ELB)





# Types of Elastic Load Balancer (ELB)



## Application Load Balancer

- Operates at the request level
- Routes based on the content of the request (layer 7)
- 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 (layer 4)
- 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)

Old and **shouldn't** be  
in the exam anymore



## Classic Load Balancer

- Old generation; not recommended for new applications
- Performs routing at Layer 4 and Layer 7
- Use for existing applications running in EC2-Classic



## 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

New and **starting** to  
appear in the exam



# ELB Features

Feature	ALB	NLB
OSI Layer	7	4
Target Type	IP, Instance, Lambda, ECS	IP, Instance
Protocol Listeners	HTTP, HTTPS, gRPC	TCP, UDP, TLS
PrivateLink support	No	(TCP, TLS)
Static IP address	No	Yes
HTTP header based routing	Yes	No
Source IP preservation	x-forwarded-for	Native
SSL termination	Load Balancer	Load Balancer or target



# 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



## Gateway Load Balancer

- Load balance virtual appliances such as:
  - Intrusion detection systems (IDS)
  - Intrusion prevention systems (IPS)
  - Next generation firewalls (NGFW)
  - Web application firewalls (WAF)
  - Distributed denial of service protection systems (DDoS)
- Integrate with Auto Scaling groups for elasticity
- Apply network monitoring and logging for analytics

# Routing with ALB and NLB





# Application Load Balancer (ALB)

Application Load  
Balancer (ALB)

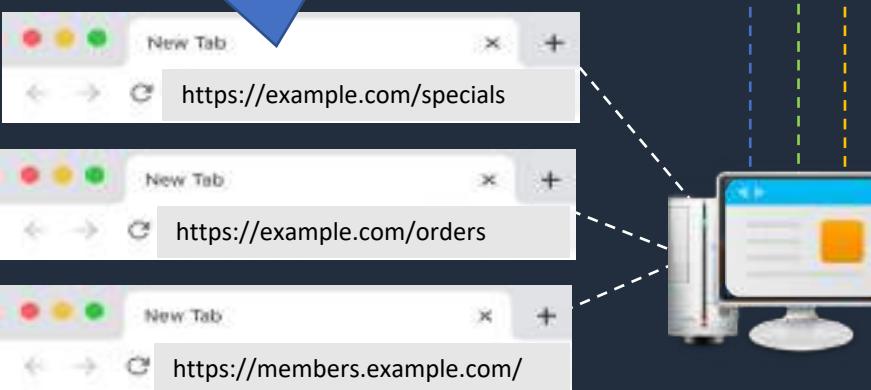
Requests can also be routed based  
on the **host** field in the **HTTP header**

A **rule** is  
configured on  
the **listener** –  
ALBs listen on  
**HTTP/HTTPS**

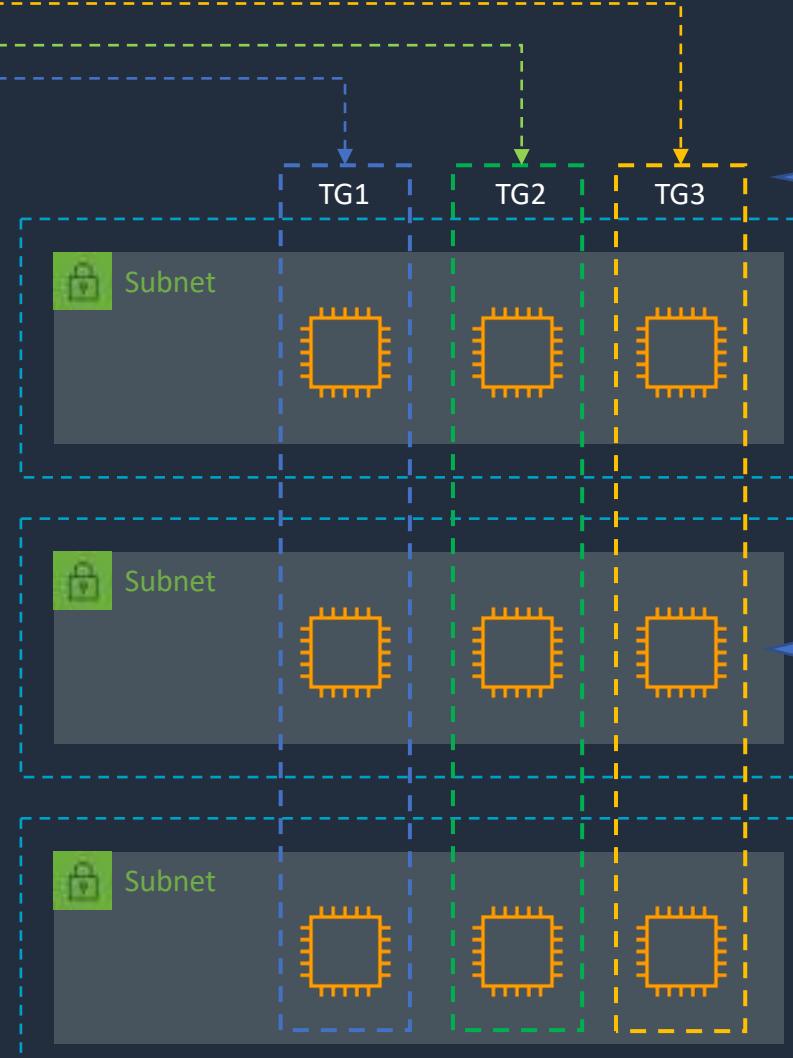


Requests can be  
routed based on  
the **path** in the **URL**

**Path-based**  
routing



**Host-based** routing

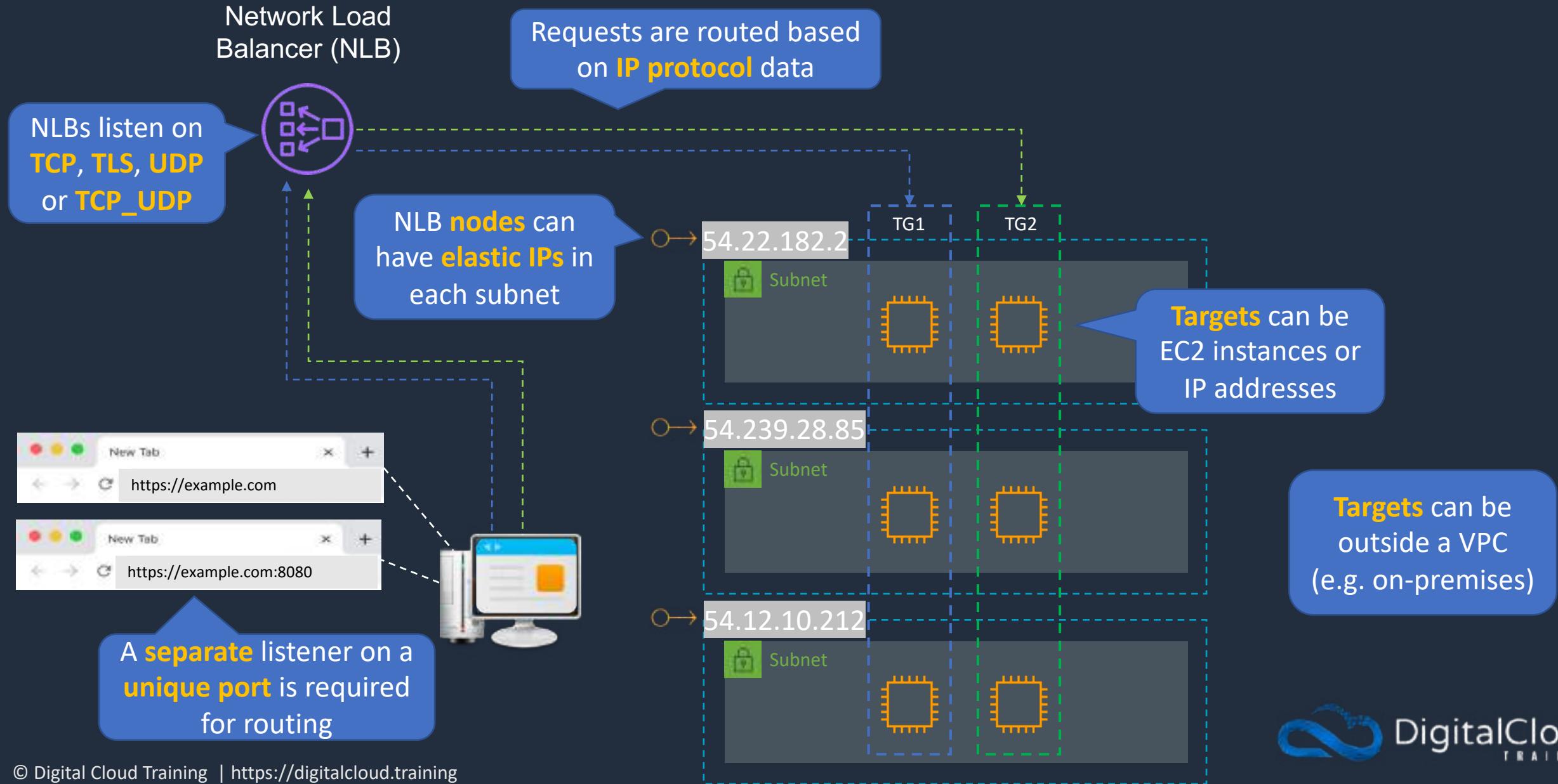


**Target groups** are used  
to route requests to  
registered targets

**Targets** can be EC2  
instances, IP addresses,  
Lambda functions or  
containers



# Network Load Balancer (NLB)



# Create Target Groups



# Network Load Balancer (NLB)



# Application Load Balancer (ALB)



# ALB - Query String Routing



# Amazon EC2 Scaling Policies

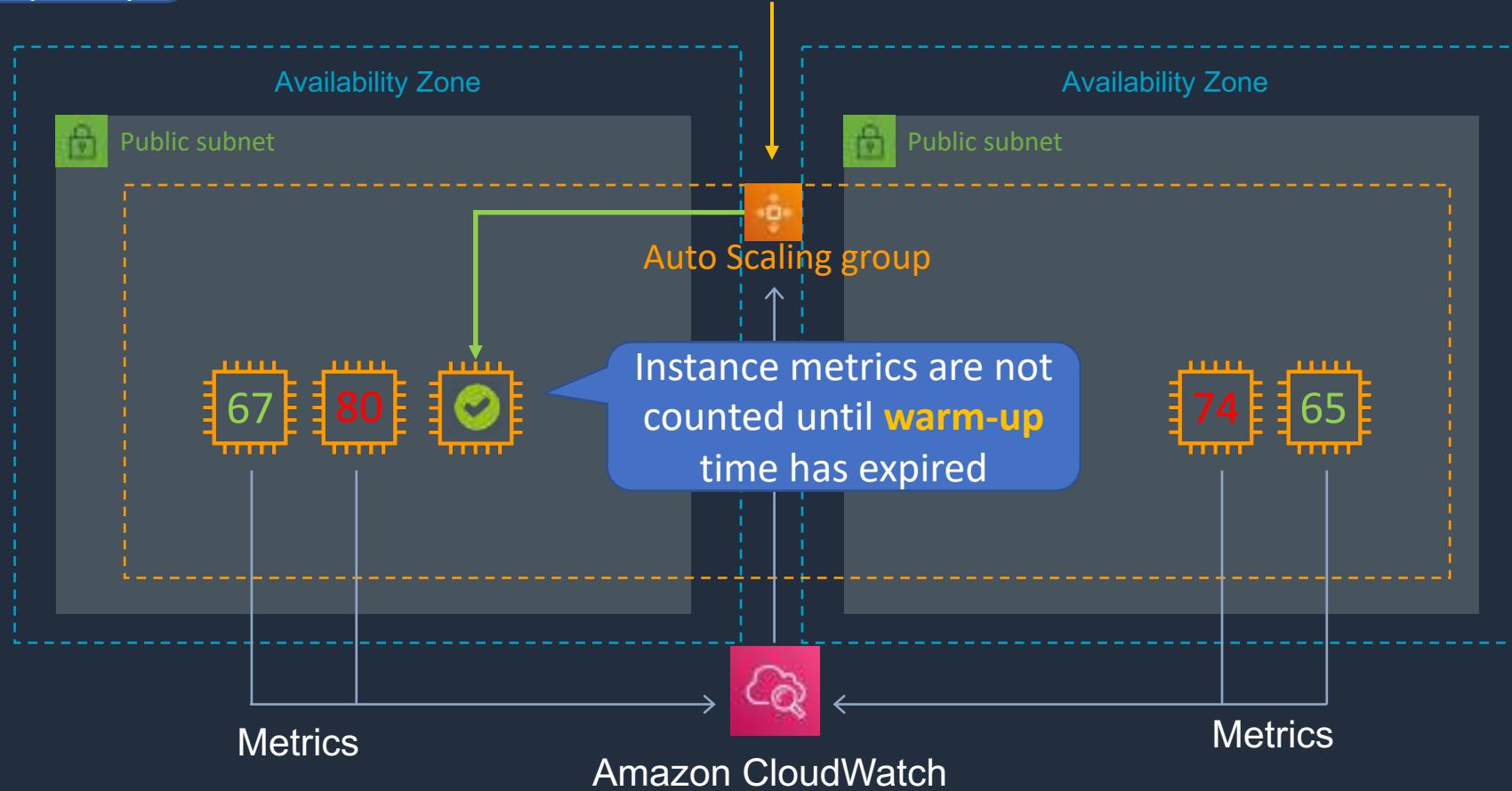




# Dynamic Scaling – Target Tracking

AWS recommend scaling on metrics with a **1-minute** frequency

**ASGAverageCPUUtilization = 60%**

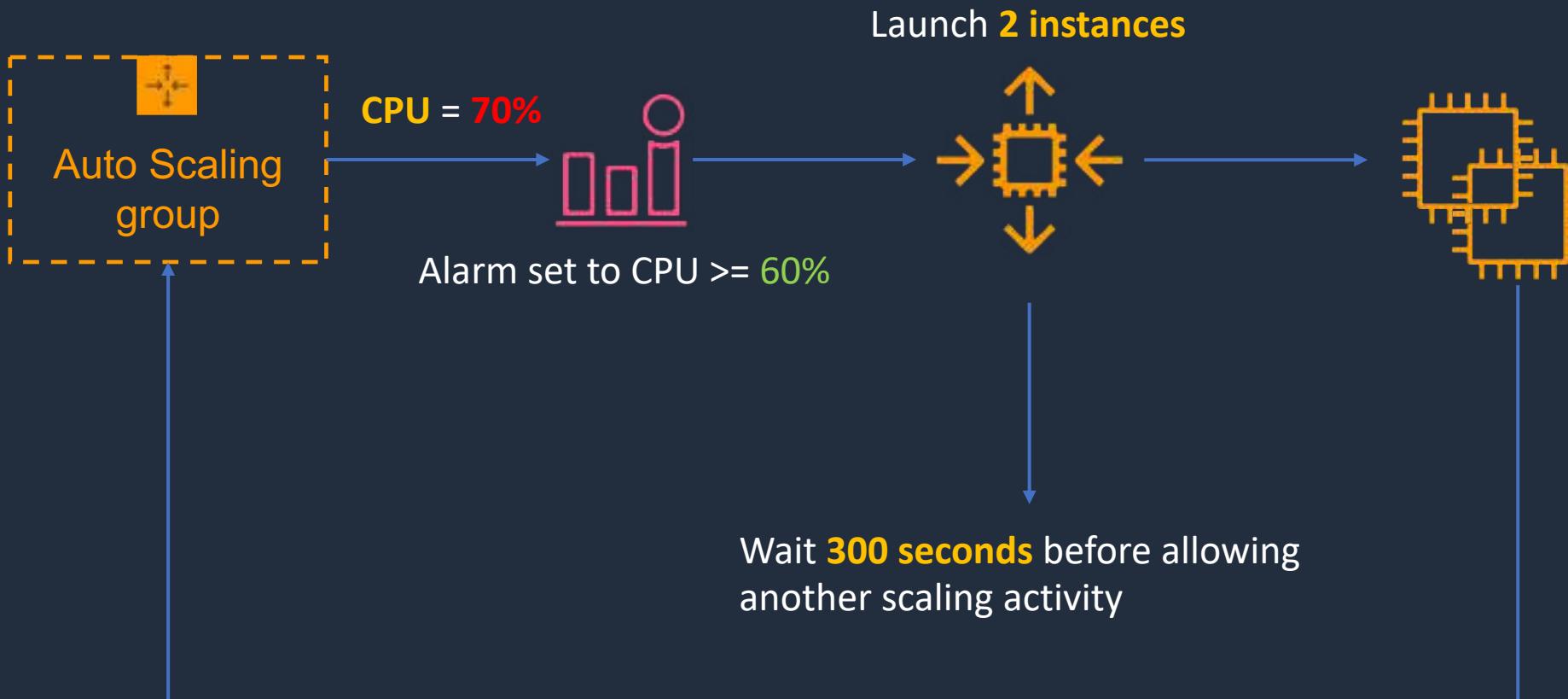


**Average CPU = 71.5%**



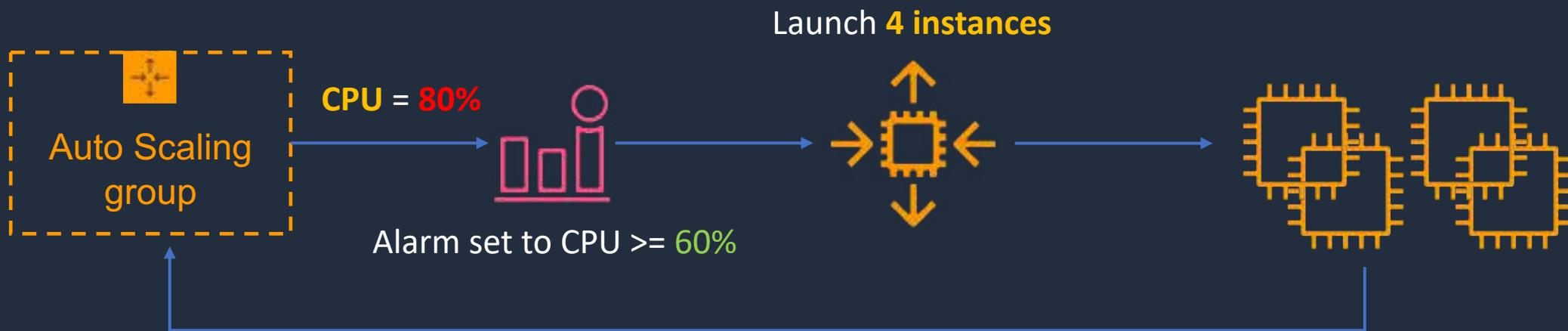
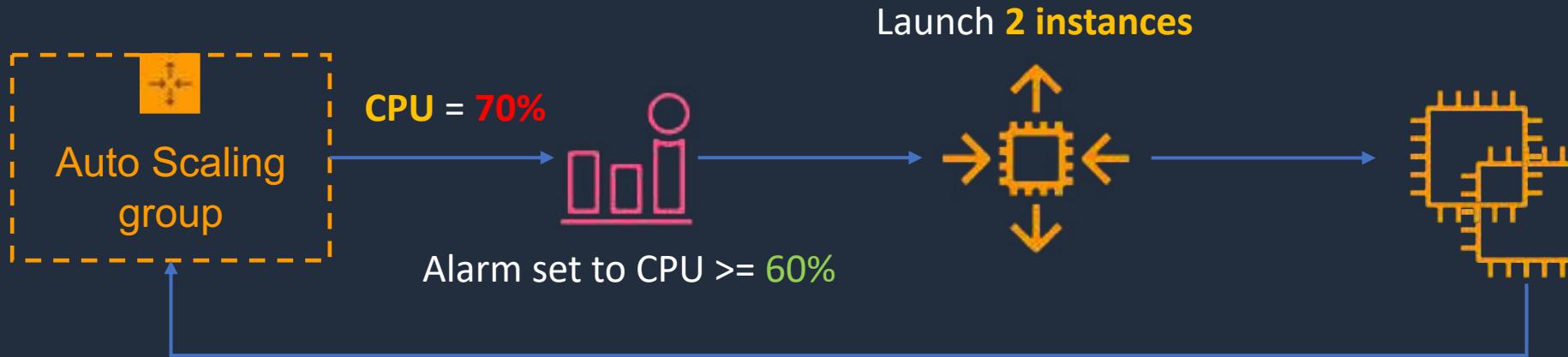
# Dynamic Scaling – Simple Scaling

---



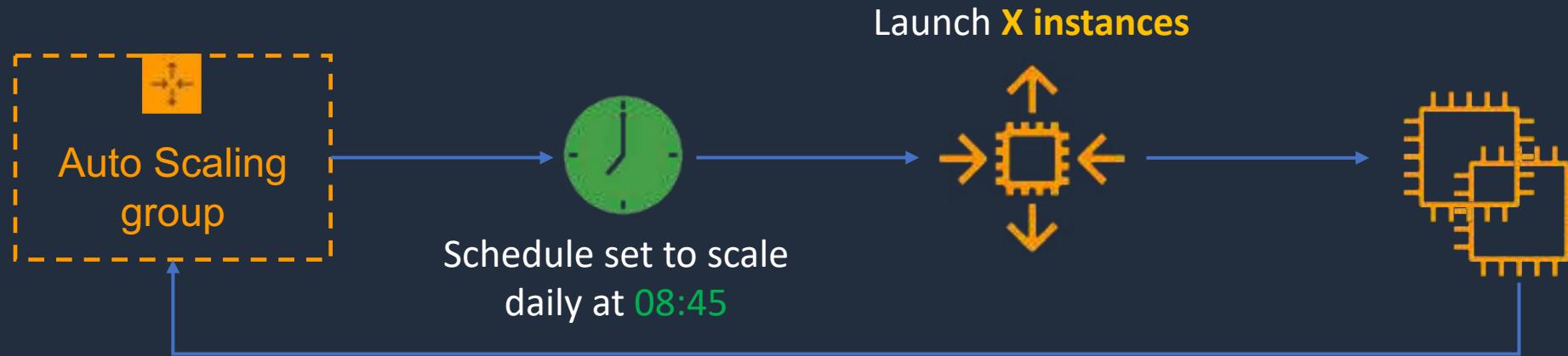


# Dynamic Scaling – Step Scaling





# Scheduled Scaling



Attempts to maintain **desired** count ->

The **minimum** instances running at any time ->

The **maximum** instances that can run ->

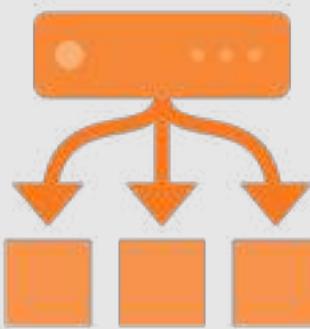
A screenshot of the AWS Auto Scaling Scheduled Action configuration interface. It shows the following settings:

- Desired capacity: 15
- Min: 10
- Max: 25
- Recurrence: Every day (Cron: 45 8 \* \* \*)
- Start time: 2021/03/01 08:45 (Specify the start time in UTC)

# Elastically Scale the Application



# Cross-Zone Load Balancing





# Cross-Zone Load Balancing

---

When cross-zone load balancing is enabled:

- Each load balancer node distributes traffic across the registered targets in all enabled Availability Zones

When cross-zone load balancing is disabled:

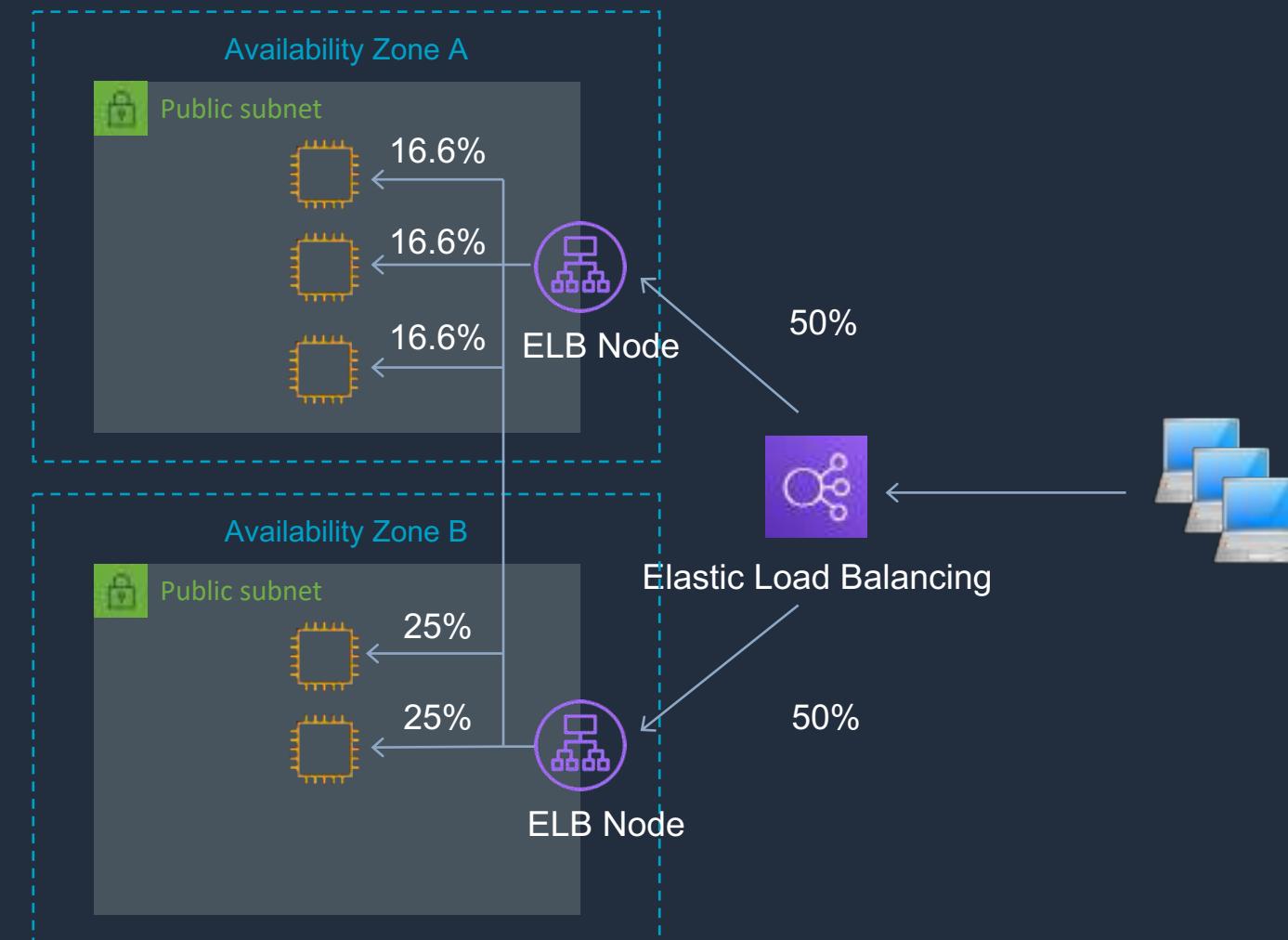
- Each load balancer node distributes traffic only across the registered targets in its Availability Zone
- With Application Load Balancers, cross-zone load balancing is **always enabled**
- With Network Load Balancers and Gateway Load Balancers, cross-zone load balancing is **disabled by default**



# Cross-Zone Load Balancing - Disabled

If cross-zone load balancing is disabled:

- Each of the three targets in Availability Zone A receives 16.6% of the traffic
- Each of the two targets in Availability Zone B receives 25% of the traffic

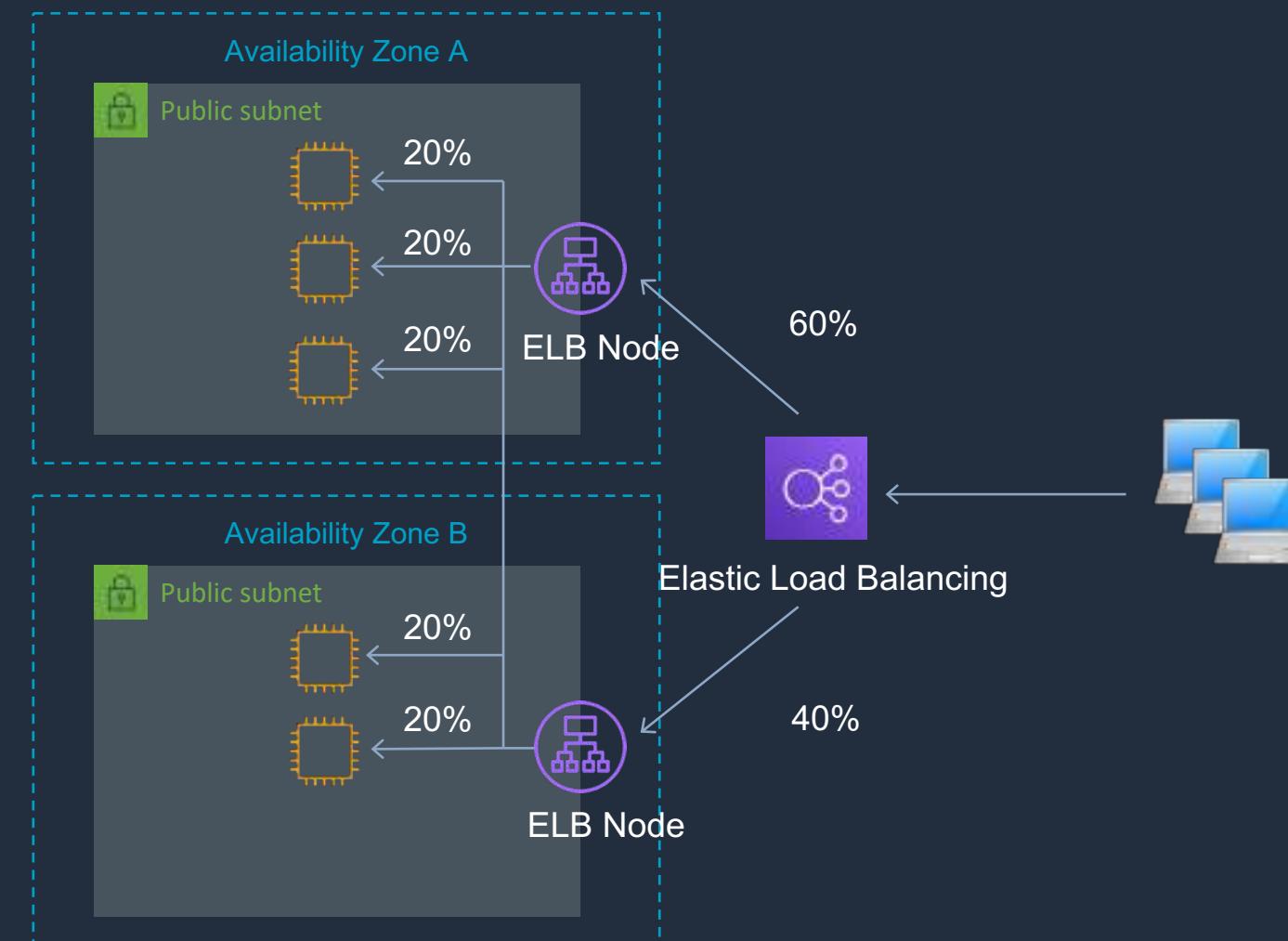




# Cross-Zone Load Balancing - Enabled

If cross-zone load balancing is enabled:

- Each of the five targets receives 20% of the traffic

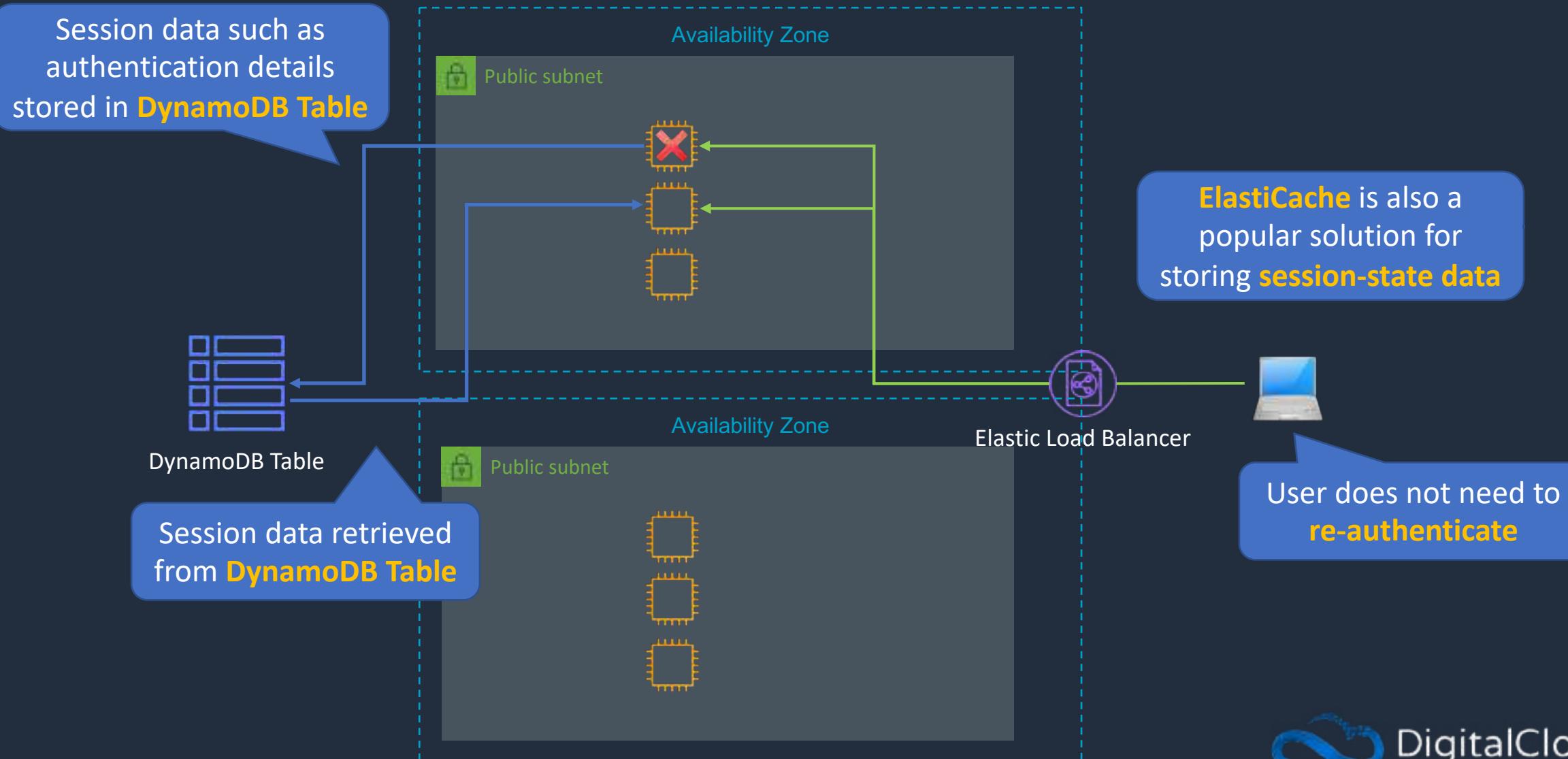


# Session State and Session Stickiness



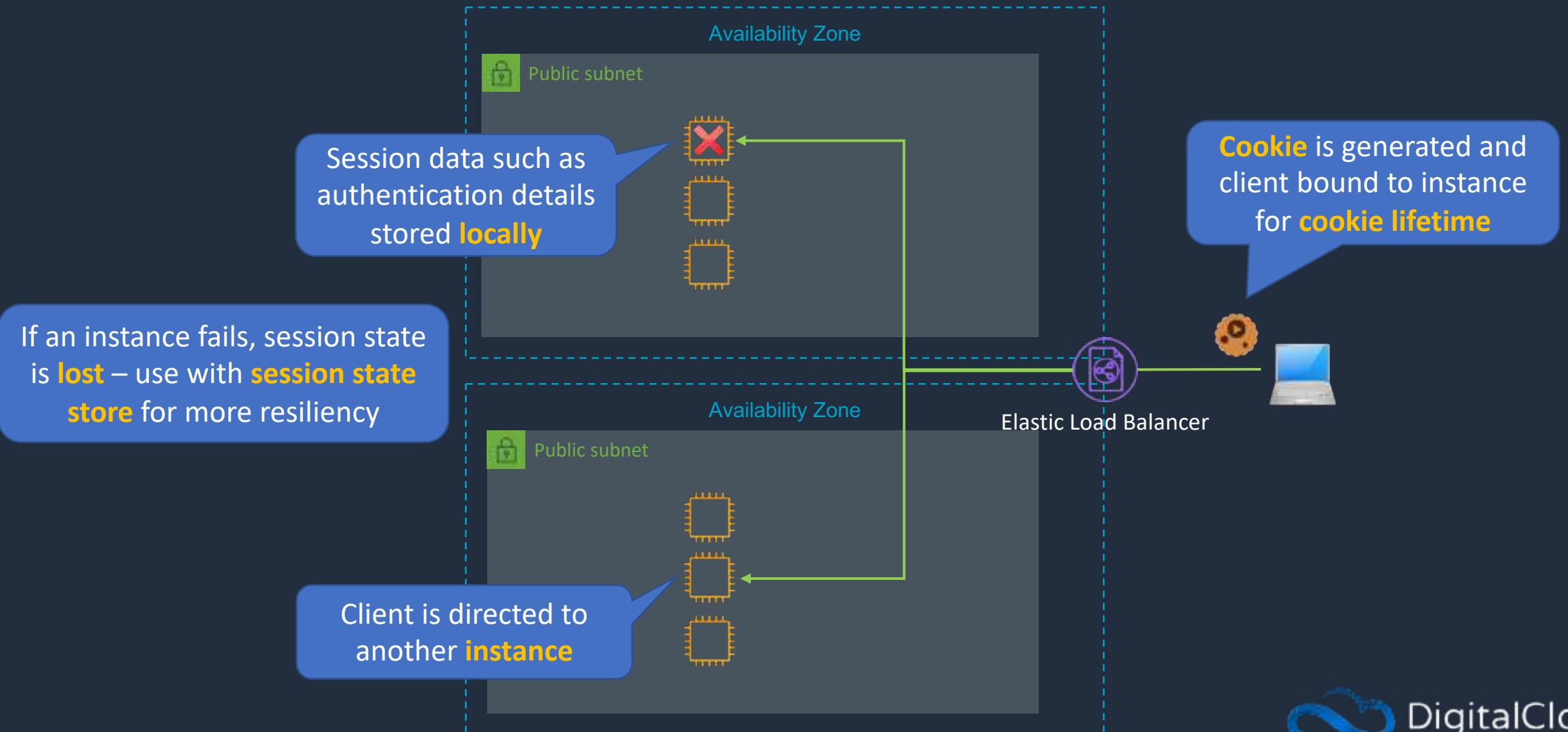


# Storing Session State





# Sticky Sessions



# Enable Sticky Sessions



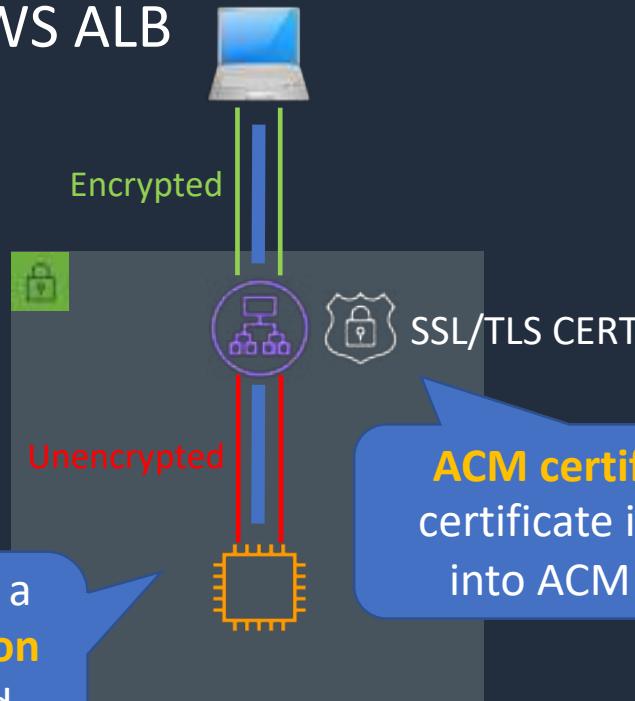
# Secure Listeners for ELB





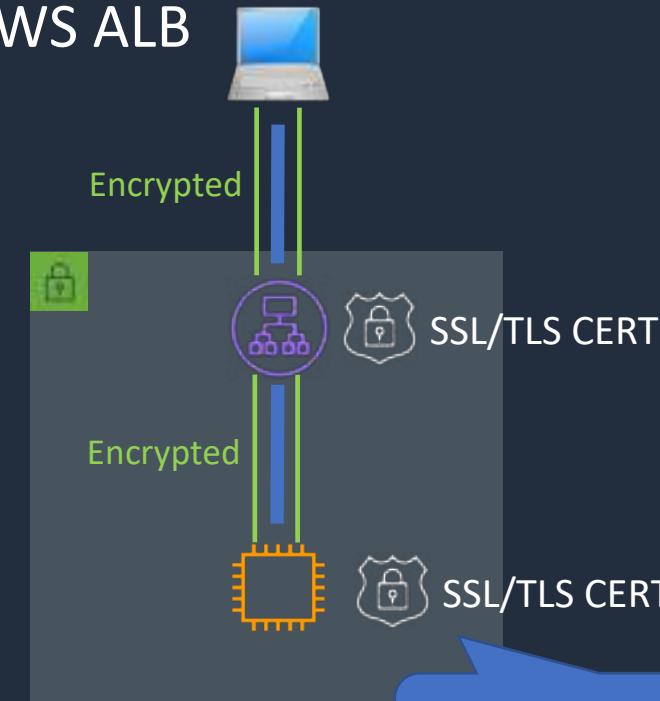
# SSL/TLS Termination

AWS ALB



With a **L7 ELB** a  
**new connection**  
is established  
with the instance

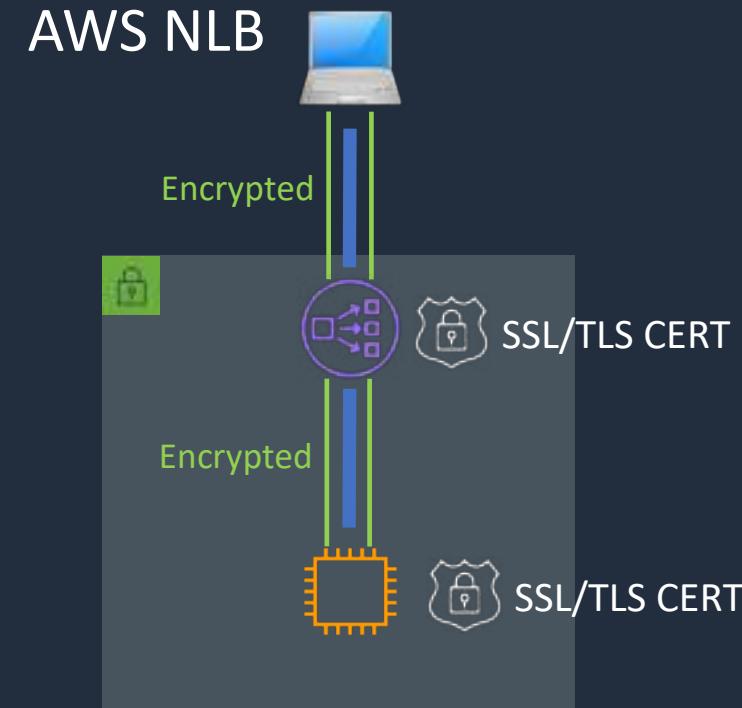
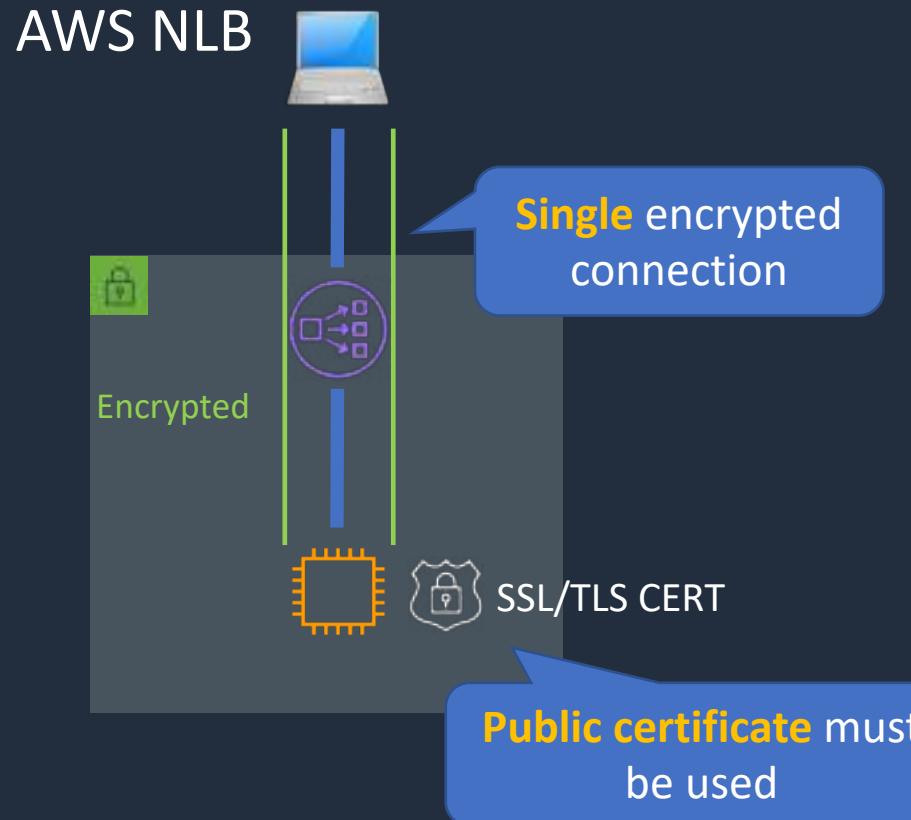
AWS ALB



**Self-signed certificate**  
can be used



# SSL/TLS Termination



# Create a Secure Listener



# Clean-up Resources



# Architecture Patterns – Auto Scaling and ELB





# Architecture Patterns - Auto Scaling and ELB

---

---

## Requirement

High availability and elastic scalability for web servers

Low-latency connections over UDP to a pool of instances running a gaming application

Clients need to whitelist static IP addresses for a highly available load balanced application in an AWS Region

## Solution

Use Amazon EC2 Auto Scaling and an Application Load Balancer across multiple AZs

Use a Network Load Balancer with a UDP listener

Use an NLB and create static IP addresses in each AZ



# Architecture Patterns - Auto Scaling and ELB

---

---

## Requirement

Application on EC2 in an Auto Scaling group requires disaster recovery across Regions

Application on EC2 must scale in larger increments if a big increase in traffic occurs, compared to small increases in traffic

Need to scale EC2 instances behind an ALB based on the number of requests completed by each instance

## Solution

Create an ASG in a second Region with the capacity set to 0. Take snapshots and copy them across Regions (Lambda or DLM)

Use Auto Scaling with a Step Scaling policy and configure a larger capacity increase

Configure a target tracking policy using the ALBRequestCountPerTarget metric



# Architecture Patterns - Auto Scaling and ELB

---

---

## Requirement

Application runs on EC2 behind an ALB. Once authenticated users should not need to reauthenticate if an instance fails

## Solution

Use session state store such as DynamoDB or ElastiCache

Company is deploying an IDS/IPS system using virtual appliances and needs to scale horizontally

Deploy a Gateway Load Balancer in front of the virtual appliances

# SECTION 5

## AWS Organizations

# AWS Organizations





# AWS Organizations

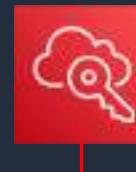
---

---

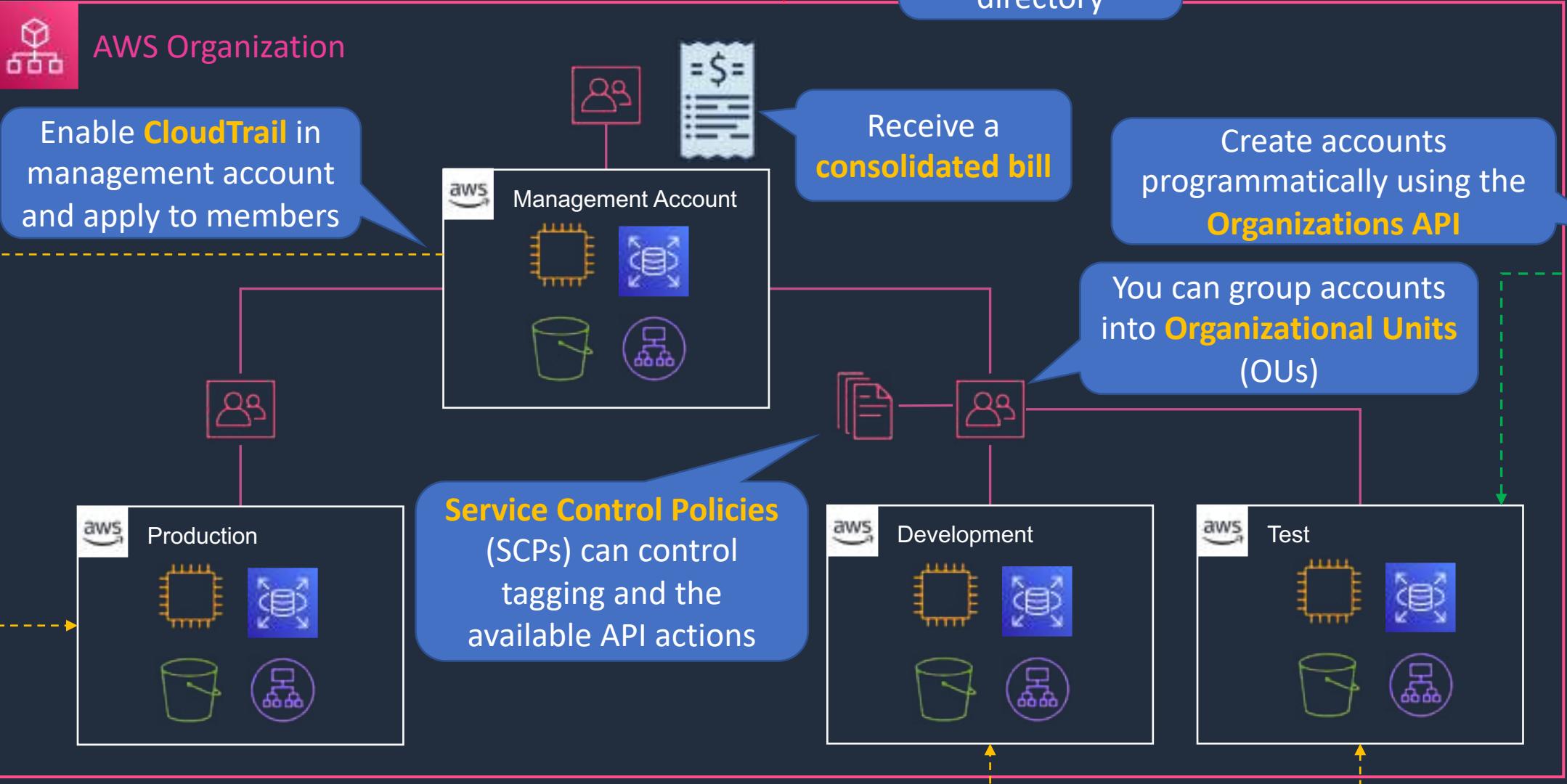
- AWS organizations allows you to consolidate multiple AWS accounts into an organization that you create and centrally manage
- Available in two feature sets:
  - **Consolidated Billing**
  - **All features**
- Includes root accounts and organizational units
- Policies are applied to root accounts or OUs
- Consolidated billing includes:
  - **Paying Account** – independent and cannot access resources of other accounts
  - **Linked Accounts** – all linked accounts are independent



# AWS Organizations



Enable AWS SSO  
using on-prem  
directory

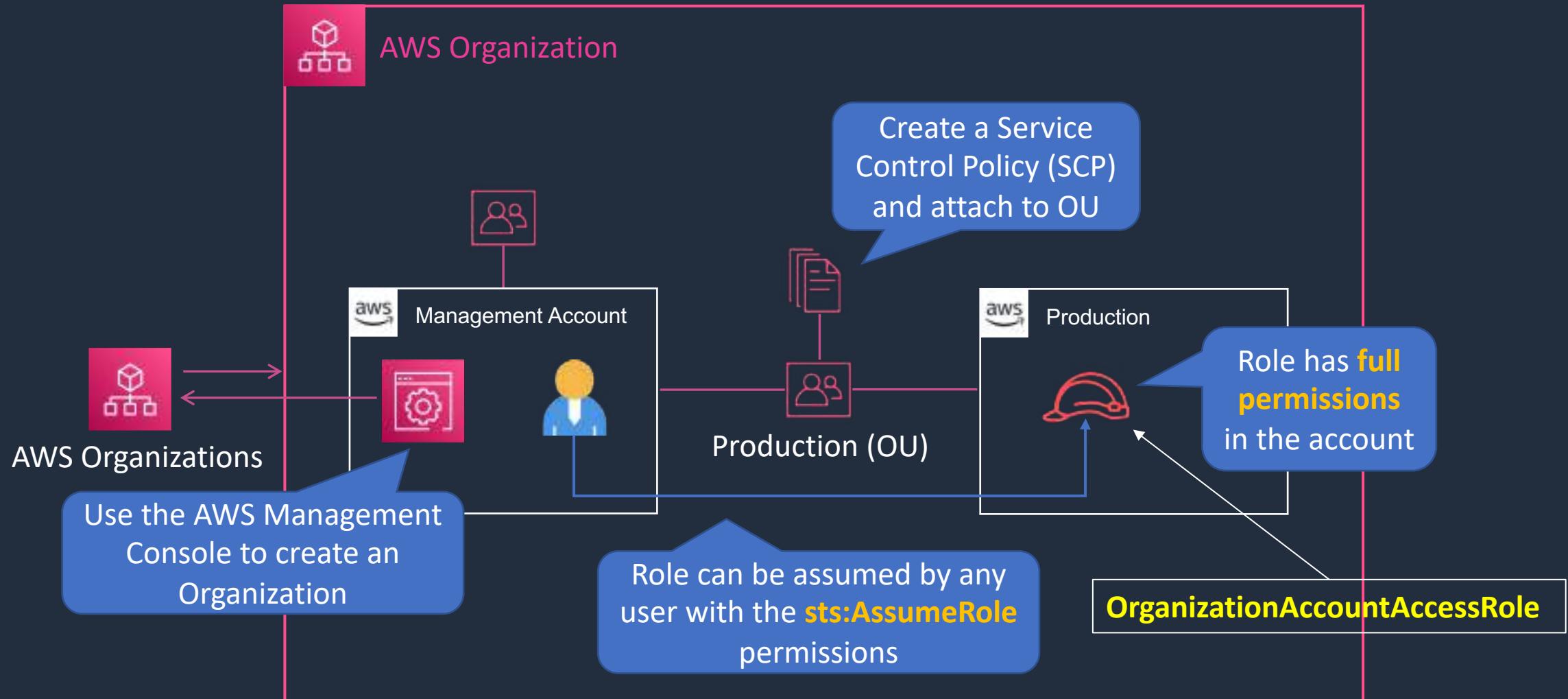


# Create AWS Organization and Add Account





# Account Configuration

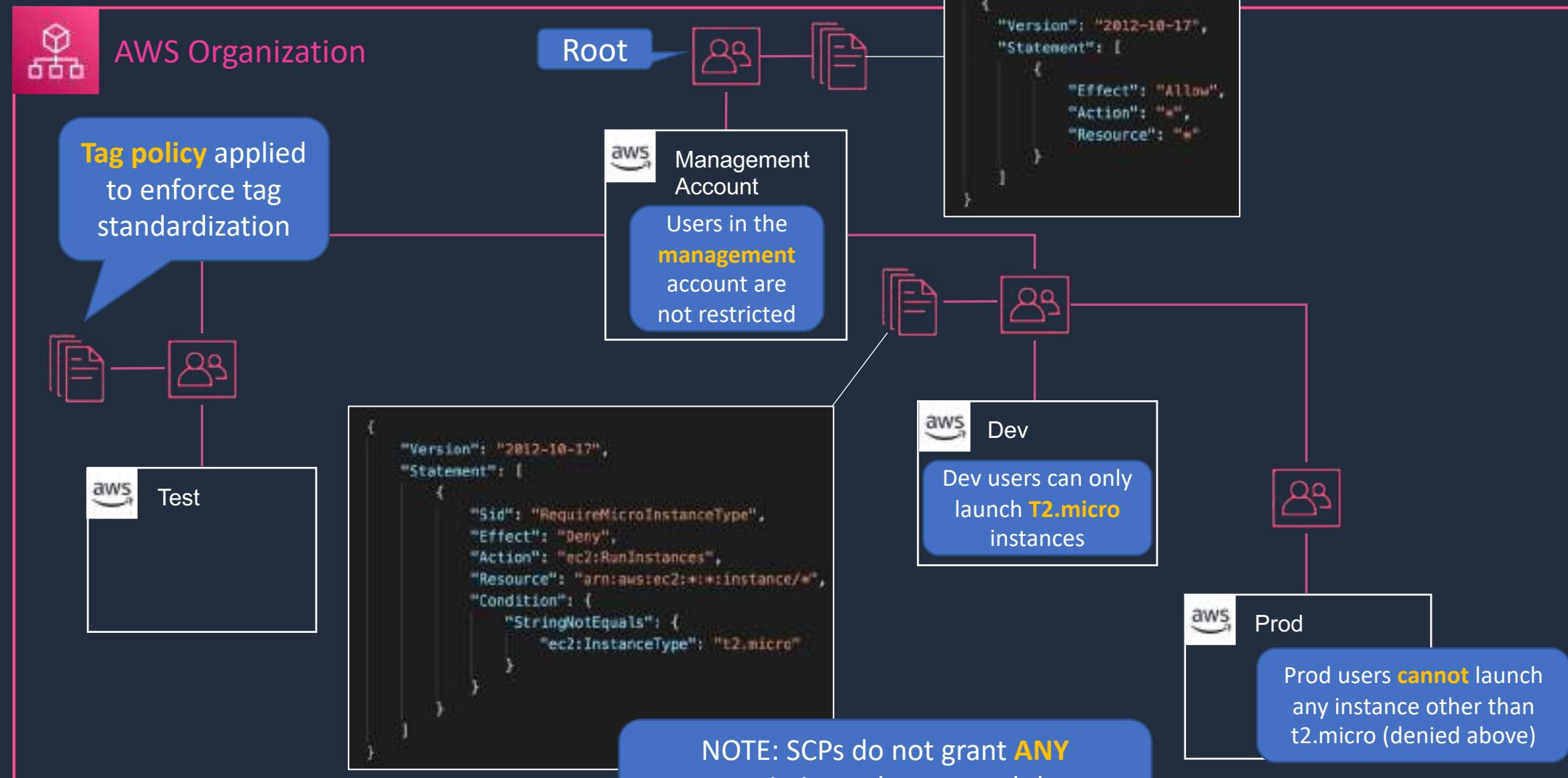


# Service Control Policies (SCPs)



# Service Control Policies

SCPs control the maximum available permissions



# Create Service Control Policy (SCP)



# Architecture Patterns – AWS Organizations





# Architecture Patterns – AWS Organizations

---

## Requirement

A company needs a method of quickly creating AWS accounts programmatically

## Solution

Use the Organizations API to create the accounts programmatically

Users in a member account in AWS Organizations should be restricted from making changes in IAM

User a Service Control Policy (SCP) to deny access to IAM actions

An AWS account must be moved between Organizations

Migrate the account using the AWS Organizations console



# Architecture Patterns – AWS Organizations

---

## Requirement

A solutions architect created a new account through the Organizations console and needs to login to launch resources

Multiple member accounts in AWS Organizations require the same permissions to be restricted using SCPs

The developers in a company each have their own AWS accounts for testing. The security team wish to enable central governance

## Solution

The architect should switch roles to access the new account

Create an OU and add the member accounts then attach the SCP to the OU

Create an AWS Organization and send an invite to each developer's AWS account to join the Organization

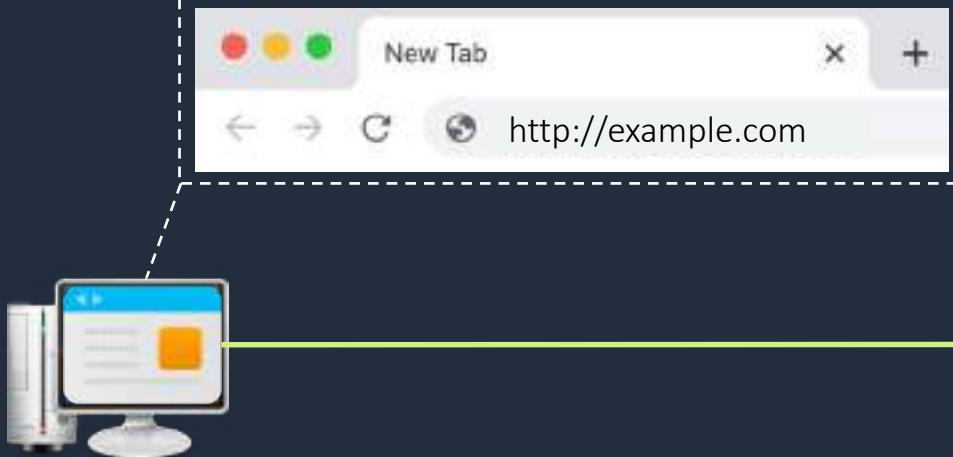
# SECTION 6

## Amazon Virtual Private Cloud (VPC)

# IPv4 Addressing Primer



# What is an IPv4 address?



IP addresses are the addresses computers use to communicate

52.134.26.12



Web Server

Connection is made using IP address

What's the IP address for example.com?



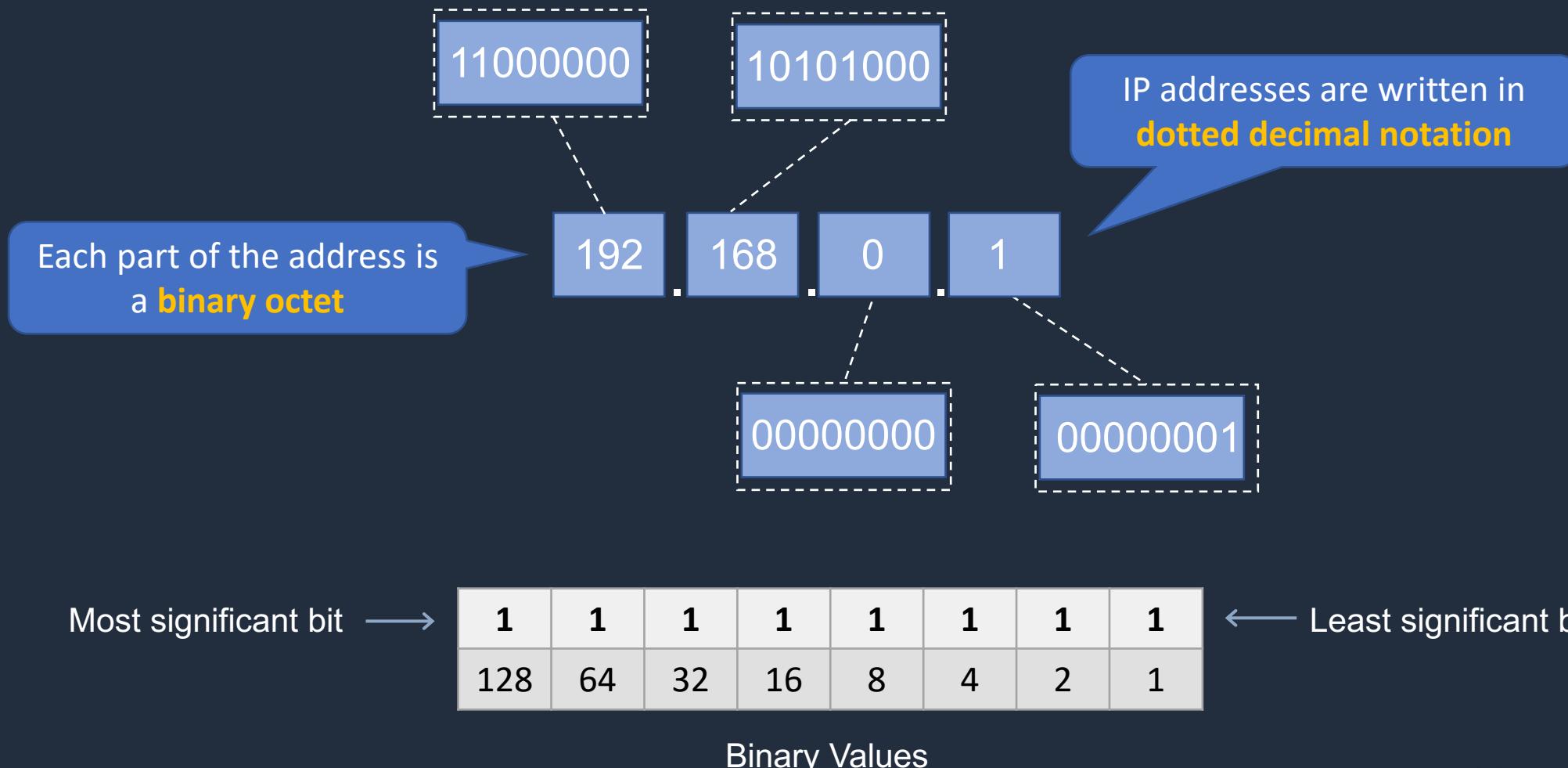
DNS Server

Name	Type	Value
example.com	A	52.134.26.12
test.com	A	137.10.47.51

DNS Zone File



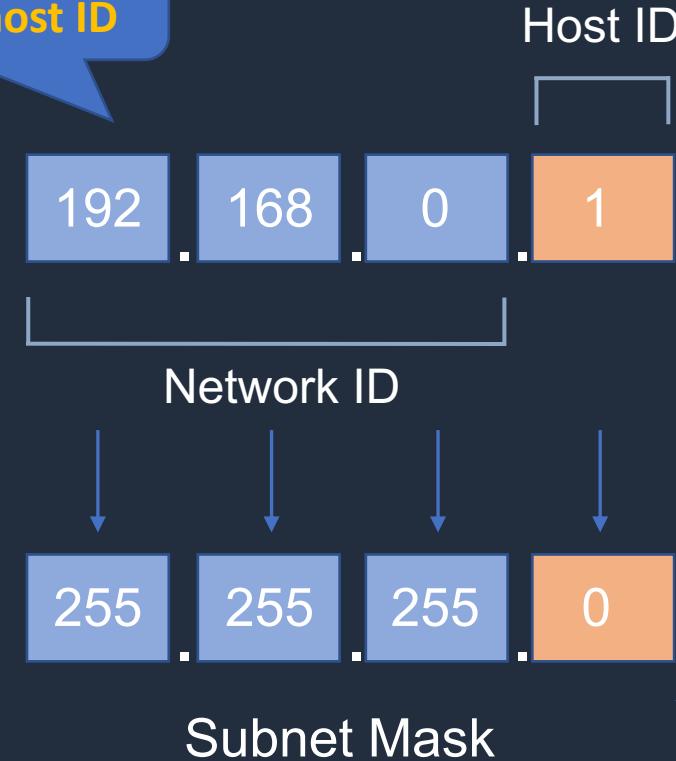
# Structure of an IPv4 Address





# 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
-----	-----	---	---

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

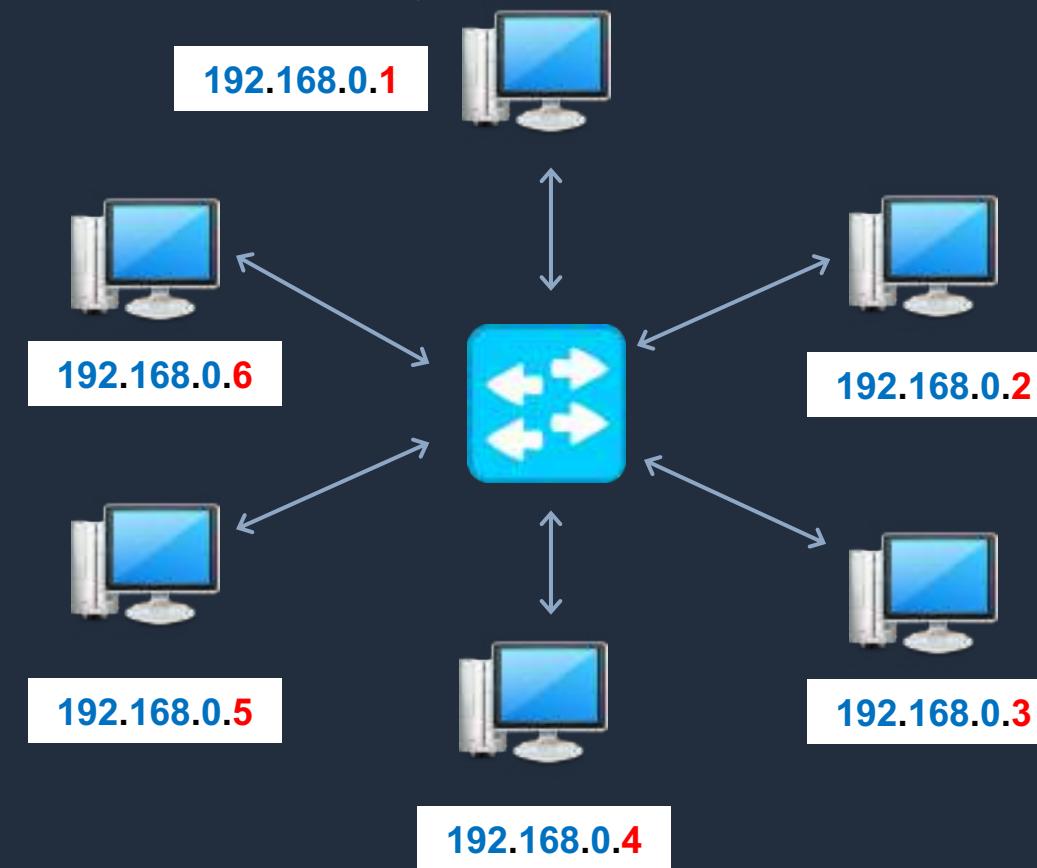
Subnet Mask    

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

24 bits

192.168.0.0/24

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

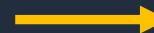




# Classes of IPv4 Address

Class A

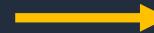
10	0	0	0
255	0	0	0



First assignable address = 10.0.0.1  
Last assignable address = 10.255.255.254  
Total networks = 126  
Usable addresses per network = 16,777,214

Class B

172	16	0	0
255	255	0	0



First assignable address = 172.16.0.1  
Last assignable address = 172.16.255.254  
Total networks = 16,382  
Usable addresses per network = 65,534

Class C

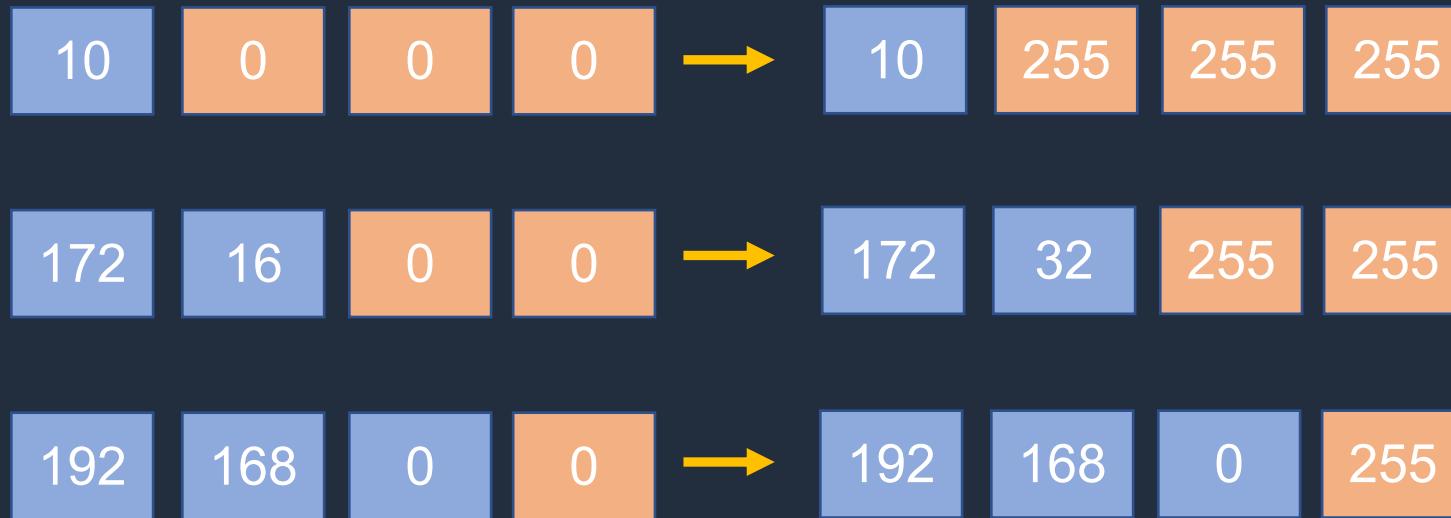
192	168	0	0
255	255	255	0



First assignable address = 192.168.0.1  
Last assignable address = 192.168.0.254  
Total networks = 2,097,150  
Usable addresses per network = 254



# Private IP Address Ranges



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



# Classless Interdomain Routing (CIDR)

Network      

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

First Address	Last Address
192.168.0.1	192.168.0.254

/24 Subnet Mask      

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

8 host bits = 254 addresses

/16 Subnet Mask      

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

16 host bits = 65,534 addresses

First Address	Last Address
192.168.0.1	192.168.255.254

/20 Subnet Mask      

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

12 host bits = 4094 addresses

First Address	Last Address
192.168.0.1	192.168.15.254

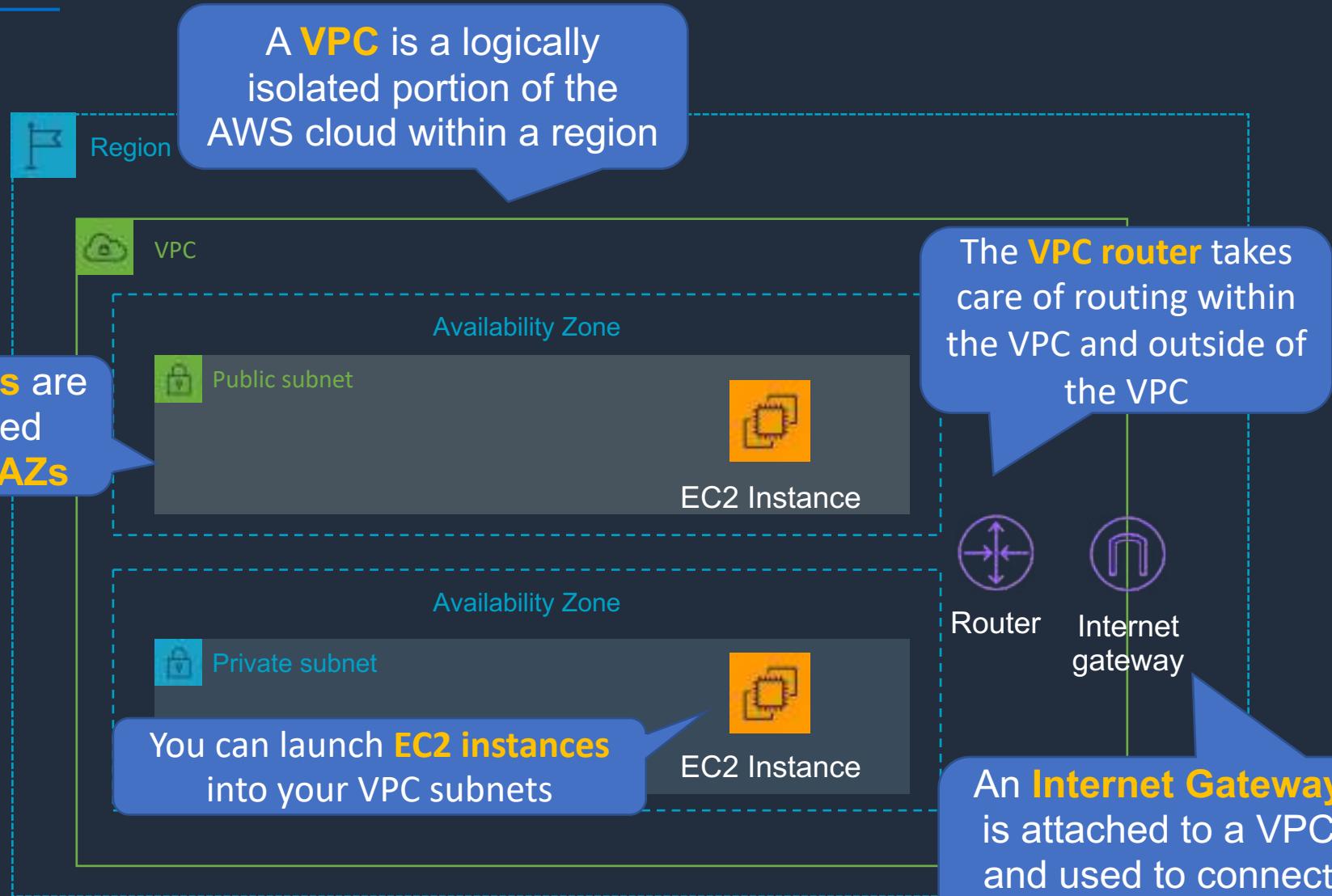
Classless Interdomain  
Routing (CIDR) uses  
**variable length  
subnets masks (VLSM)**

# Amazon VPC Overview





# Amazon Virtual Private Cloud (VPC)





# Amazon VPC



Region

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

**CIDR** stands for Classless Interdomain Routing



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

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



# Amazon VPC Components

VPC Component	What it is
Virtual Private Cloud (VPC)	A logically isolated virtual network in the AWS cloud
Subnet	A segment of a VPC's IP address range where you can place groups of isolated resources
Internet Gateway/Egress-only Internet Gateway	The Amazon VPC side of a connection to the public Internet for IPv4/IPv6
Router	Routers interconnect subnets and direct traffic between Internet gateways, virtual private gateways, NAT gateways, and subnets
Peering Connection	Direct connection between two VPCs
VPC Endpoints	Private connection to public AWS services
NAT Instance	Enables Internet access for EC2 instances in private subnets managed by you
NAT Gateway	Enables Internet access for EC2 instances in private subnets (managed by AWS)
Virtual Private Gateway	The Amazon VPC side of a Virtual Private Network (VPN) connection
Customer Gateway	Customer side of a VPN connection
AWS Direct Connect	High speed, high bandwidth, private network connection from customer to aws
Security Group	Instance-level firewall
Network ACL	Subnet-level firewall



# Amazon VPC Core Knowledge

---

- A virtual private cloud (VPC) is a virtual network dedicated to your AWS account
- Analogous to having your own data center inside AWS
- It is logically isolated from other virtual networks in the AWS Cloud
- Provides complete control over the virtual networking environment including selection of IP ranges, creation of subnets, and configuration of route tables and gateways
- You can launch your AWS resources, such as Amazon EC2 instances, into your VPC



# Amazon VPC Core Knowledge

---

---

- When you create a VPC, you must specify a range of IPv4 addresses for the VPC in the form of a Classless Inter-Domain Routing (CIDR) block; for example, 10.0.0.0/16
- A VPC spans all the Availability Zones in the region
- You have full control over who has access to the AWS resources inside your VPC
- By default you can create up to 5 VPCs per region
- A default VPC is created in each region with a subnet in each AZ

# Defining VPC CIDR Blocks





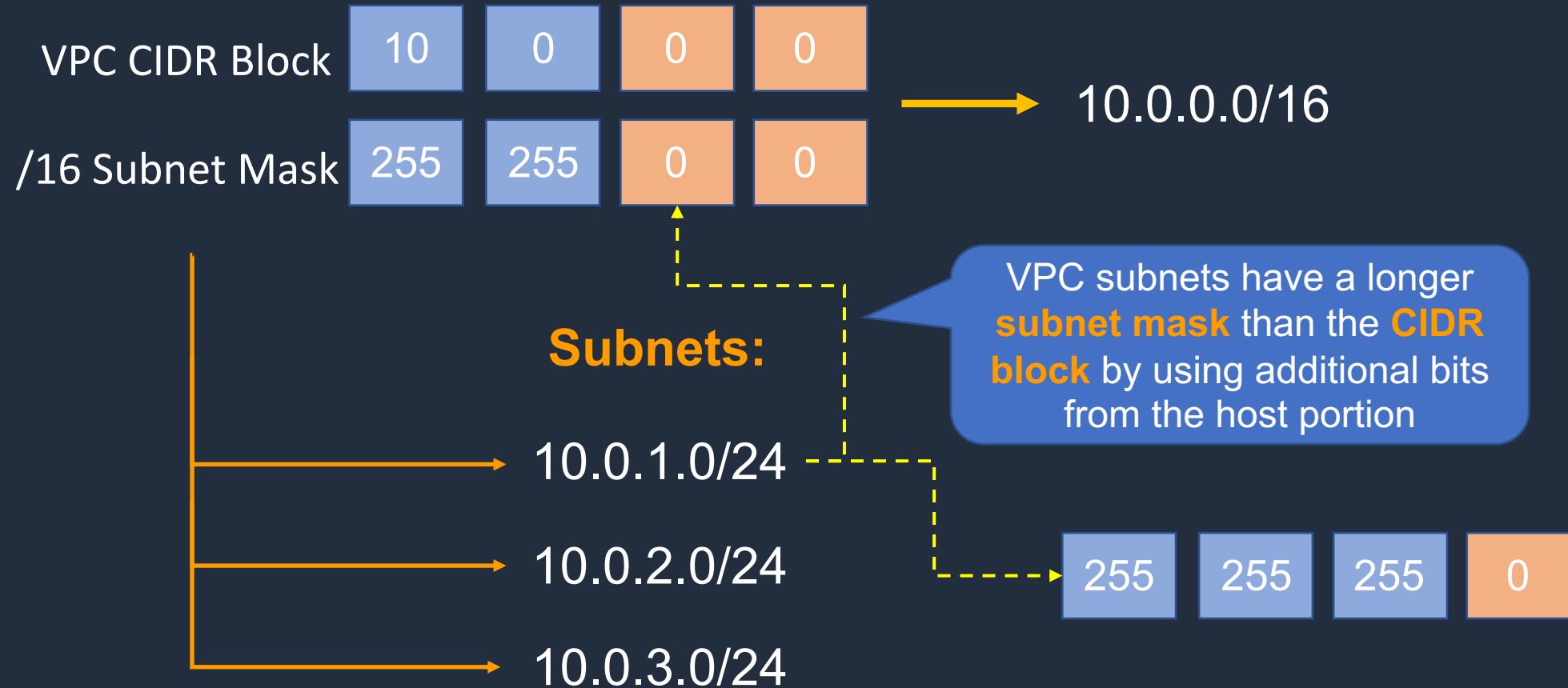
# Rules and Guidelines

- CIDR block size can be between /16 and /28
- The CIDR block must not overlap with any existing CIDR block that's associated with the VPC
- You cannot increase or decrease the size of an existing CIDR block
- The first four and last IP address are not available for use
- AWS recommend you use CIDR blocks from the RFC 1918 ranges:

RFC 1918 Range	Example CIDR Block
10.0.0.0 - 10.255.255.255 (10/8 prefix)	Your VPC must be /16 or smaller, for example, 10.0.0.0/16
172.16.0.0 - 172.31.255.255 (172.16/12 prefix)	Your VPC must be /16 or smaller, for example, 172.31.0.0/16
192.168.0.0 - 192.168.255.255 (192.168/16 prefix)	Your VPC can be smaller, for example 192.168.0.0/20



# VPC CIDR Blocks and Subnets





# Additional Considerations

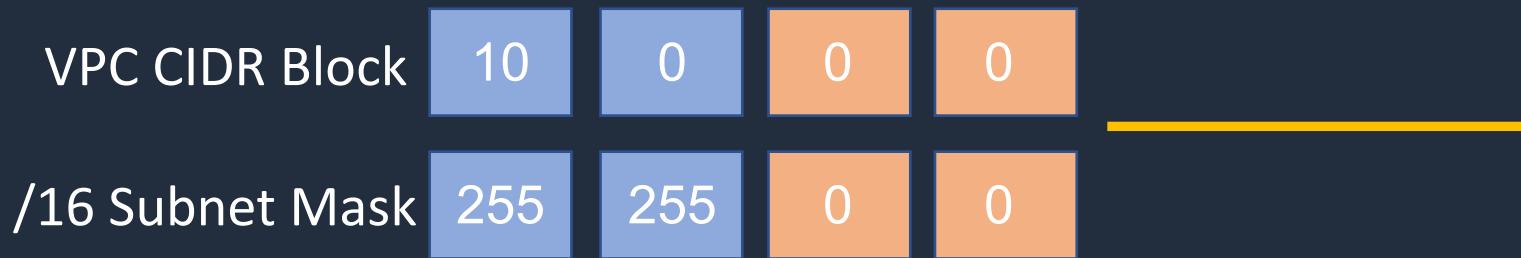
---

---

- Ensure you have enough networks and hosts
- **Bigger CIDR blocks** are typically better (more flexibility)
- **Smaller subnets** are OK for most use cases
- Consider deploying application tiers per subnet
- Split your HA resources across subnets in different AZs
- VPC Peering requires non-overlapping CIDR blocks
  - This is across all VPCs in all Regions / accounts you want to connect
- **Avoid overlapping CIDR blocks** as much as possible!



# Example VPC CIDR Block and Subnets



Subnet Name	IPv4 CIDR block	Availability Zone	Route Table	Auto-assign Public IP v4
private-1a	10.0.0.0/24	us-east-1a	Private-RT	No
private-1b	10.0.1.0/24	us-east-1b	Private-RT	No
private-1c	10.0.2.0/24	us-east-1c	Private-RT	No
public-1a	10.0.3.0/24	us-east-1a	MAIN	Yes
public-1b	10.0.4.0/24	us-east-1b	MAIN	Yes
public-1c	10.0.5.0/24	us-east-1c	MAIN	Yes

# VPC Wizard

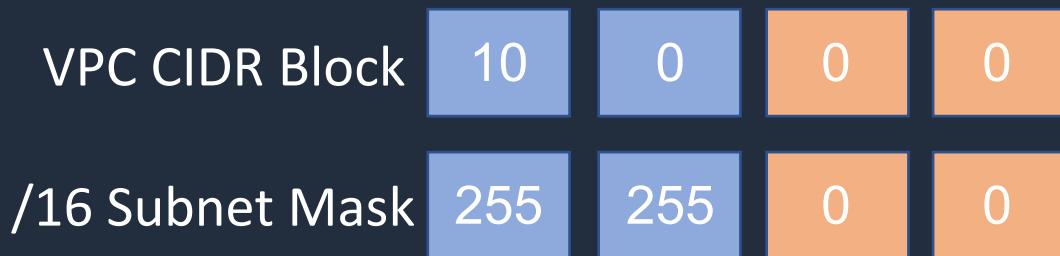


# Create a Custom VPC with Subnets





# VPC CIDR Block and Subnets



Subnet Name	IPv4 CIDR block	Availability Zone	Route Table	Auto-assign Public IPv4
private-1a	10.0.3.0/24	us-east-1a	Private-RT	No
private-1b	10.0.4.0/24	us-east-1b	Private-RT	No
public-1a	10.0.1.0/24	us-east-1a	MAIN	Yes
public-1b	10.0.2.0/24	us-east-1b	MAIN	Yes

Has a route to an  
**Internet Gateway**

Automatically assign  
**IPv4 Public  
addresses**

# Launch Instances and Test VPC

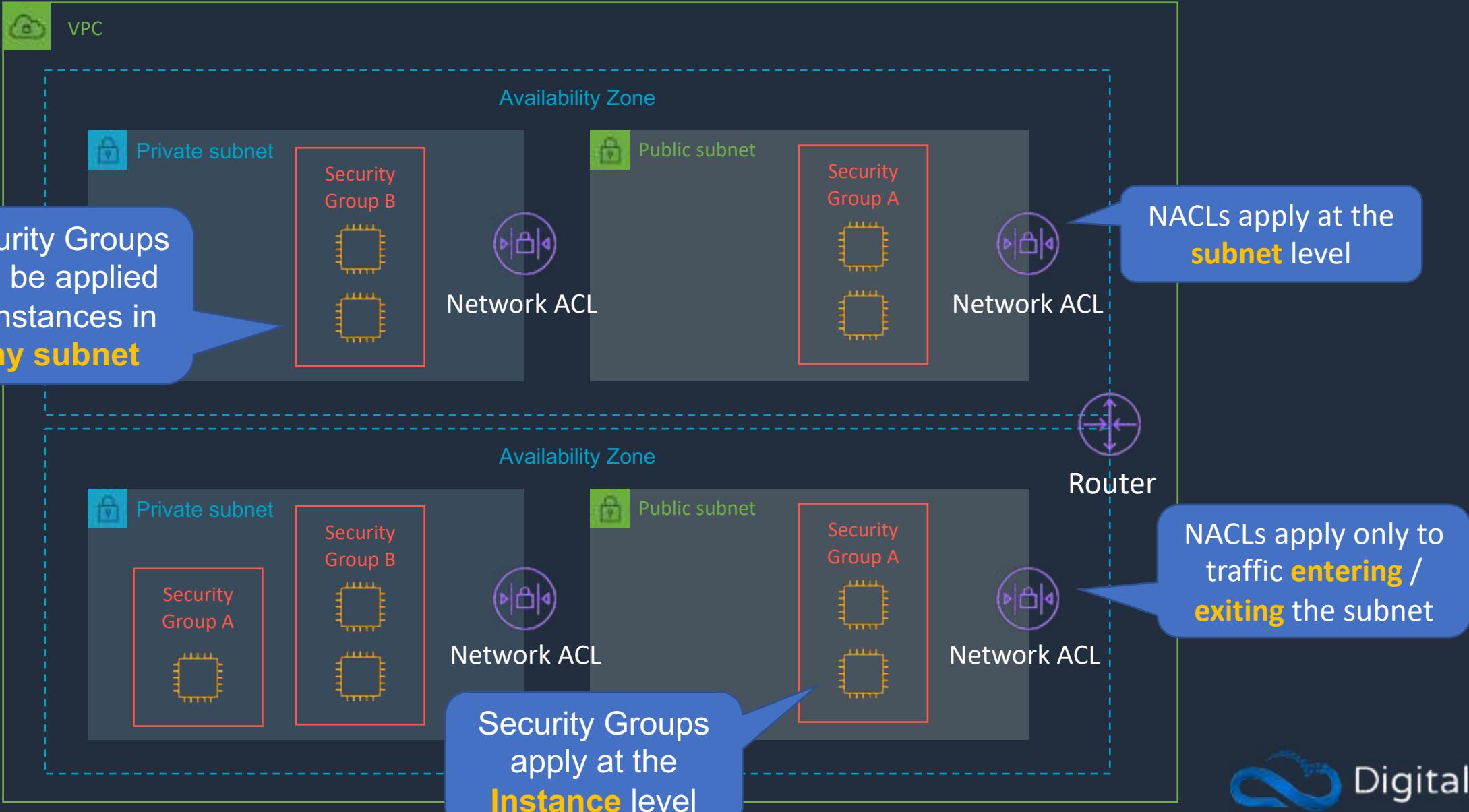


# Security Groups and Network ACLs

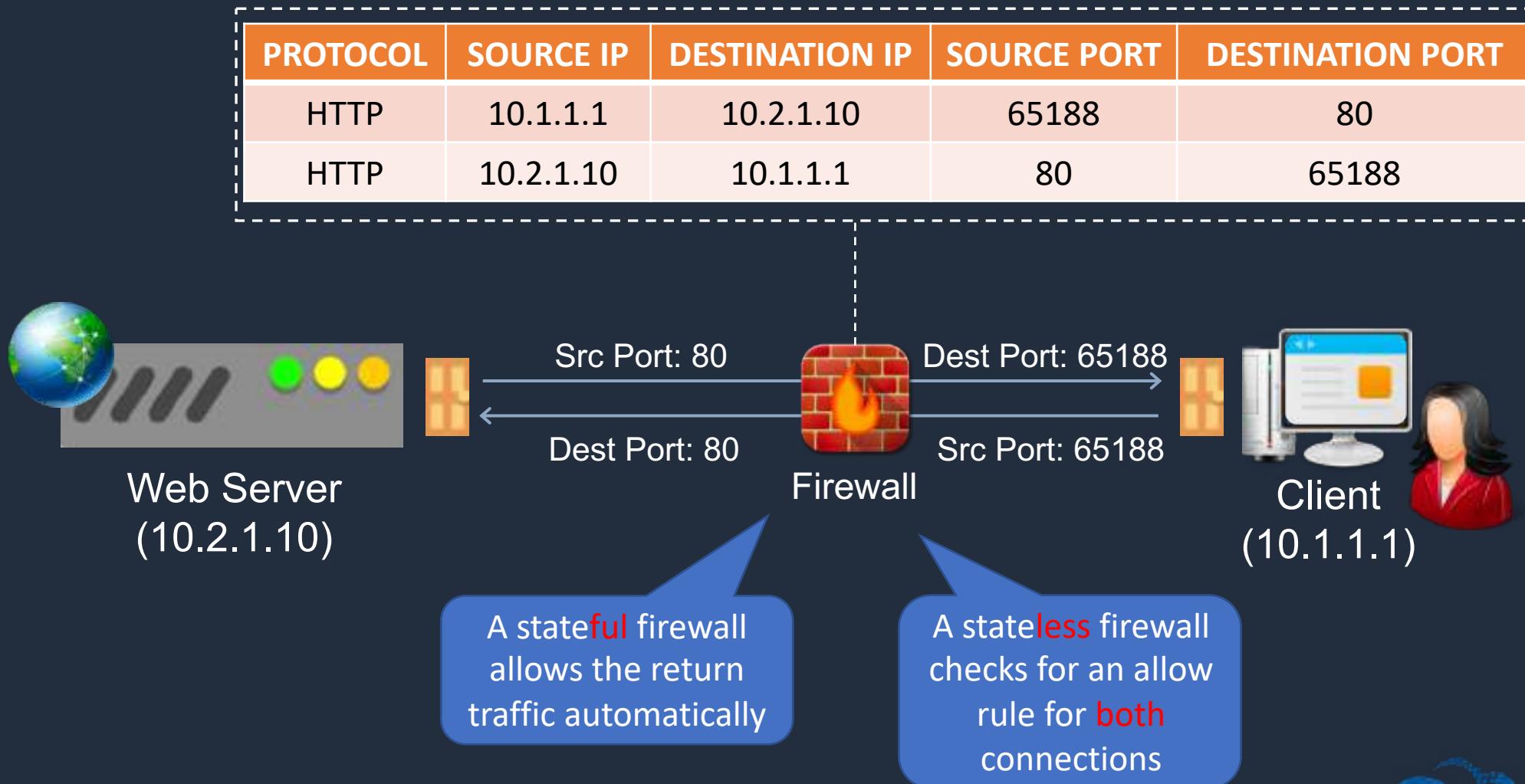




# Security Groups and Network ACLs



# Stateful vs Stateless Firewalls





# Security Group Rules

Security groups support  
**allow** rules only

## Inbound rules

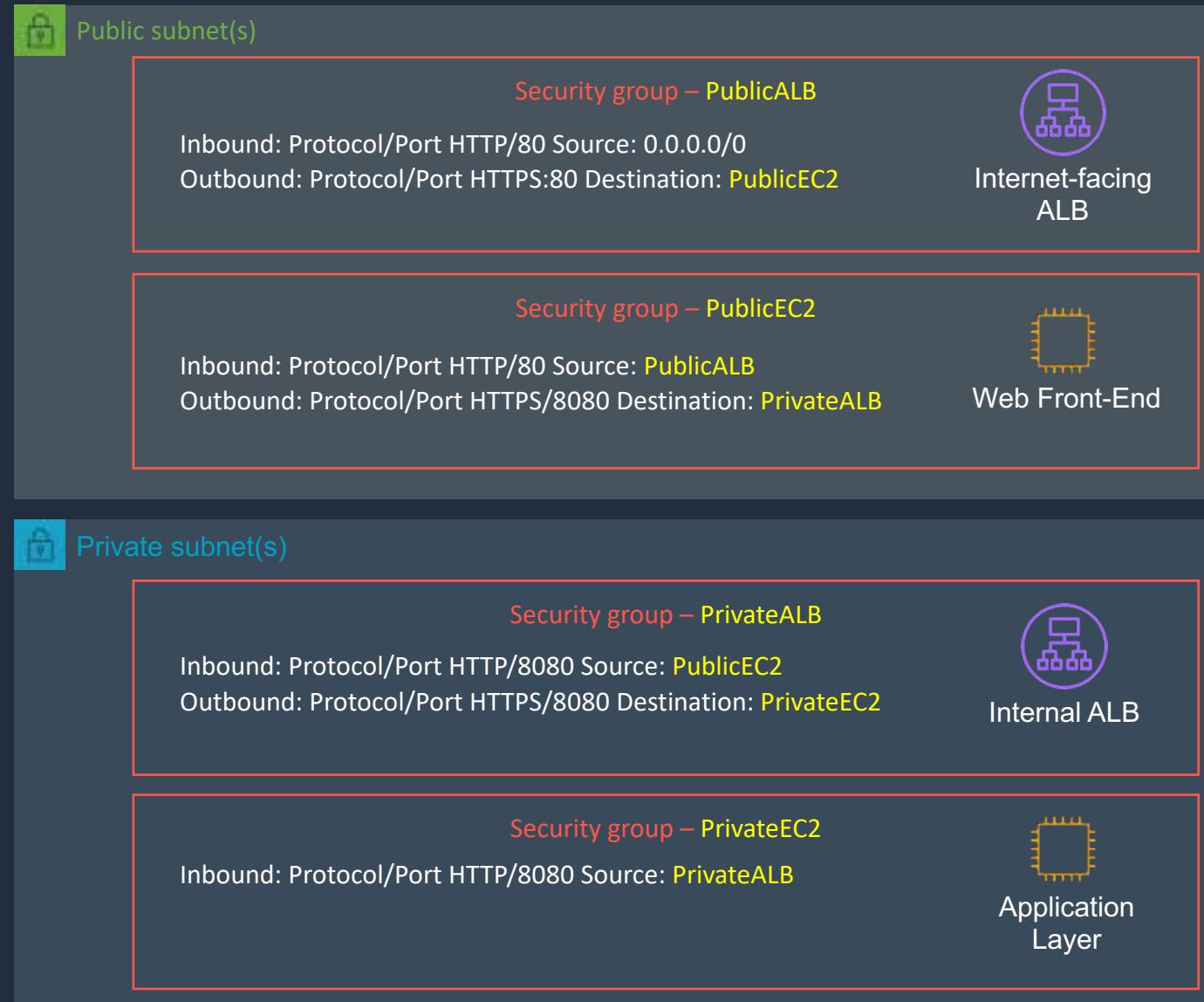
Type	Protocol	Port range	Source
SSH	TCP	22	0.0.0.0/0
RDP	TCP	3389	0.0.0.0/0
RDP	TCP	3389	::/0
HTTPS	TCP	443	0.0.0.0/0
HTTPS	TCP	443	::/0
All ICMP - IPv4	ICMP	All	0.0.0.0/0

Separate rules  
are defined for  
outbound traffic

A source can be an **IP  
address or security  
group ID**



# Security Groups Best Practice





# Network ACLs

## Inbound Rules

Rule #	Type	Protocol	Port Range	Source	Allow / Deny
100	ALL Traffic	ALL	ALL	0.0.0.0/0	ALLOW
101	ALL Traffic	ALL	ALL	::/0	ALLOW
*	ALL Traffic	ALL	ALL	0.0.0.0/0	DENY
*	ALL Traffic	ALL	ALL	::/0	DENY

## Outbound Rules

Rule #	Type	Protocol	Port Range	Destination	
100	ALL Traffic	ALL	ALL	0.0.0.0/0	ALLOW
101	ALL Traffic	ALL	ALL	::/0	ALLOW
*	ALL Traffic	ALL	ALL	0.0.0.0/0	DENY
*	ALL Traffic	ALL	ALL	::/0	DENY

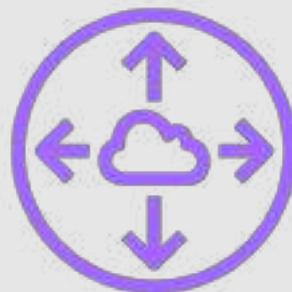
Rules are processed  
in order

NACLs have an  
explicit deny

# Configure Security Groups and NACLs

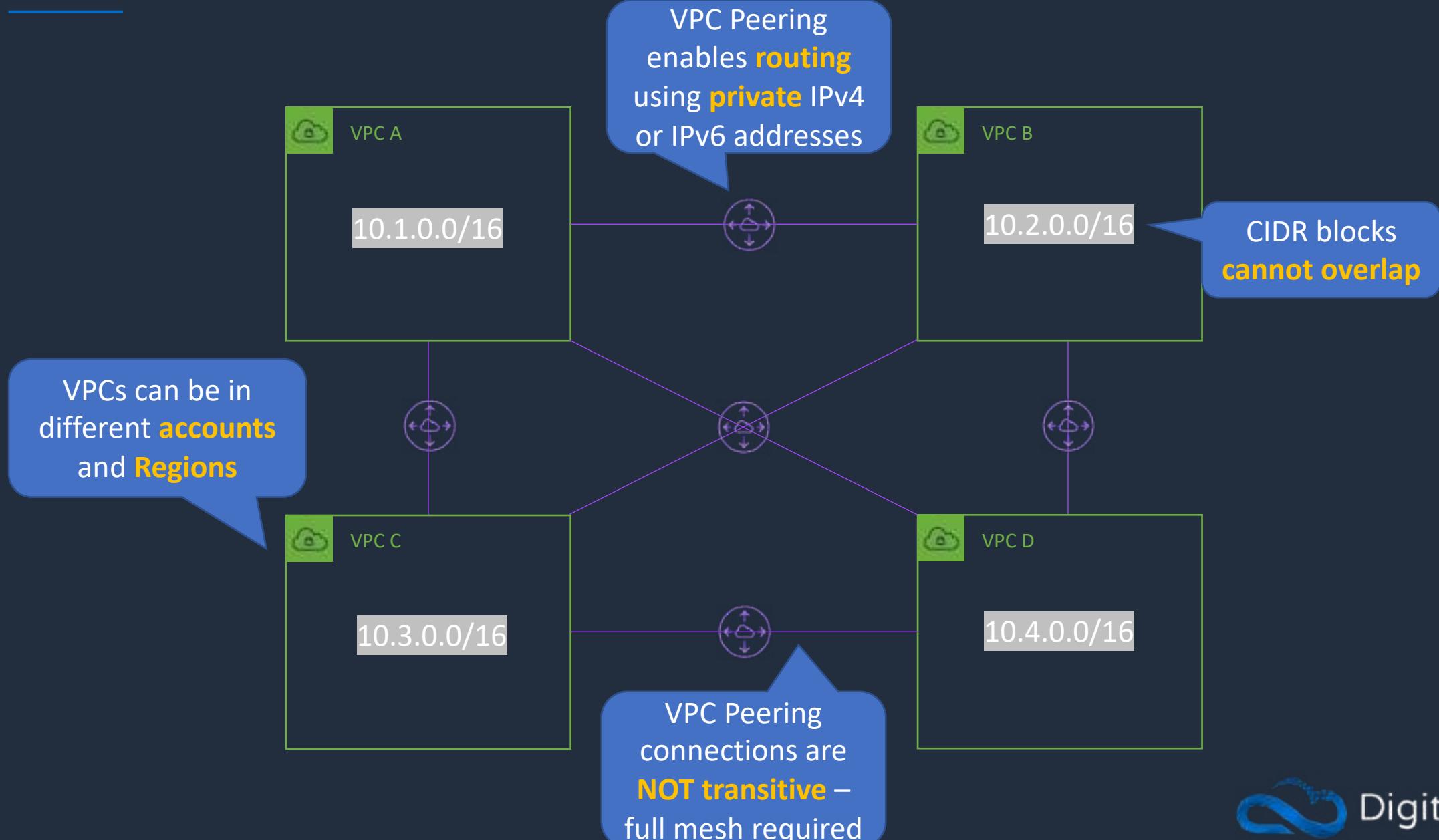


# VPC Peering





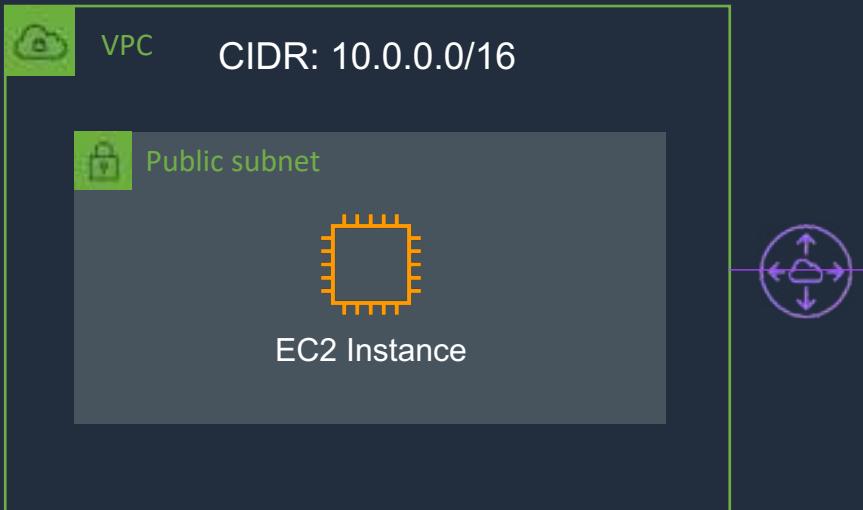
# VPC Peering



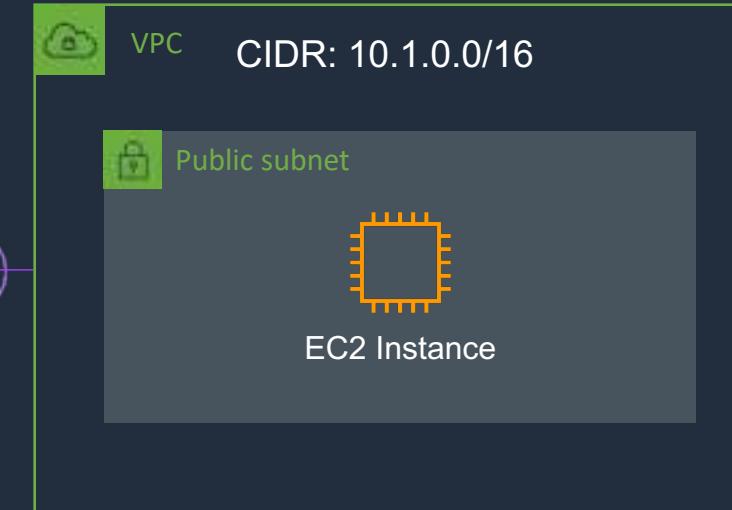


# VPC Peering

Management Account



Production Account



Security group (VPCPEER-MGMT)

Protocol	Port	Source
ICMP	All	10.1.0.0/16
TCP	22	0.0.0.0/0

Security group (VPCPEER-PROD)

Protocol	Port	Source
ICMP	All	10.0.0.0/16
TCP	22	0.0.0.0/0

Route Table

Destination	Target
10.1.0.0/16	peering-id

Route Table

Destination	Target
10.0.0.0/16	peering-id

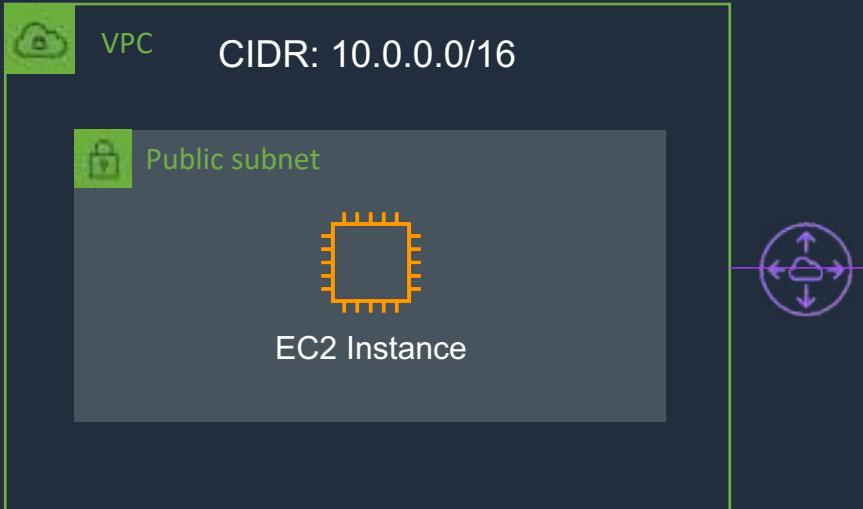
# Configure VPC Peering



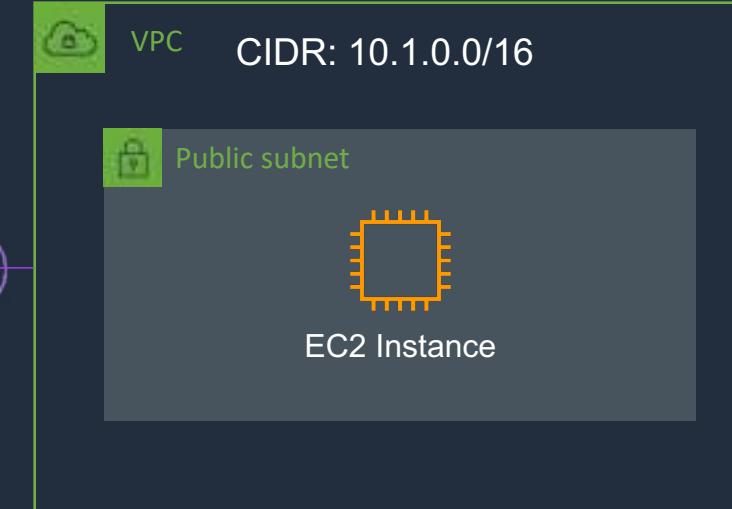


# VPC Peering

Management Account



Production Account



Security group (VPCPEER-MGMT)

Protocol	Port	Source
ICMP	All	10.1.0.0/16
TCP	22	0.0.0.0/0

Security group (VPCPEER-PROD)

Protocol	Port	Source
ICMP	All	10.0.0.0/16
TCP	22	0.0.0.0/0

Route Table

Destination	Target
10.1.0.0/16	peering-id

Route Table

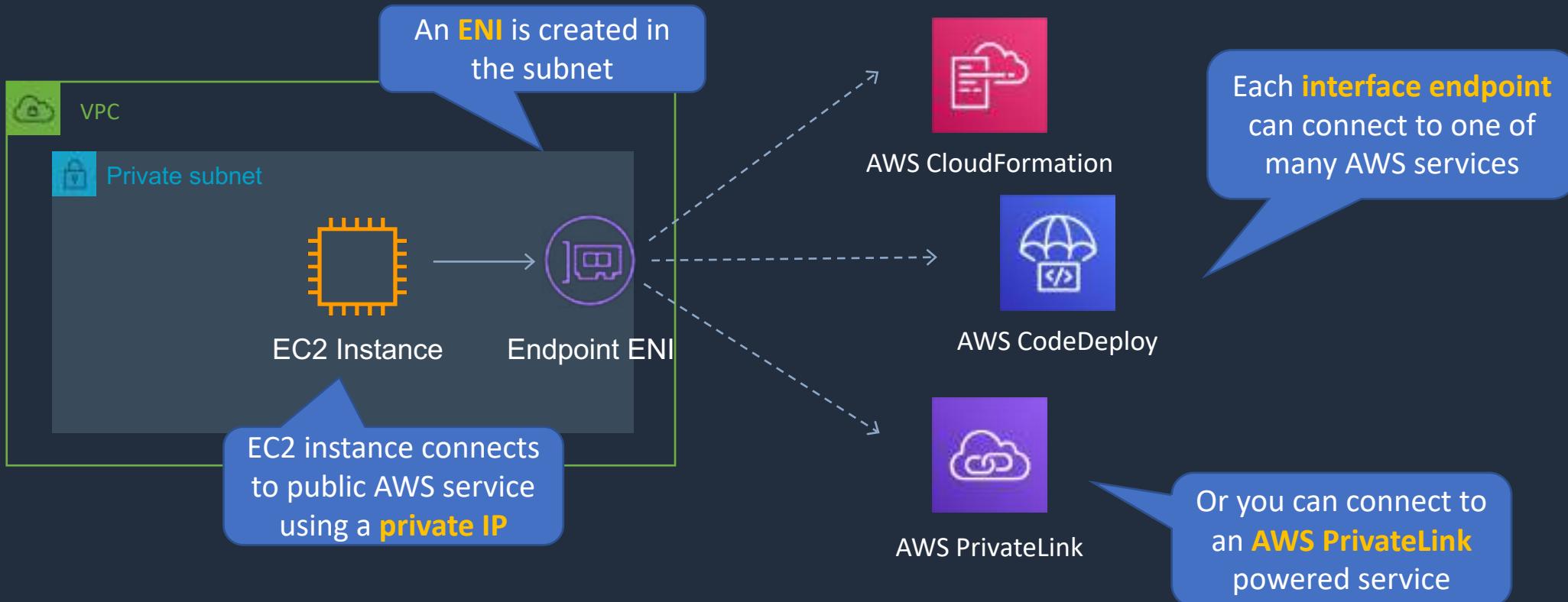
Destination	Target
10.0.0.0/16	peering-id

# VPC Endpoints



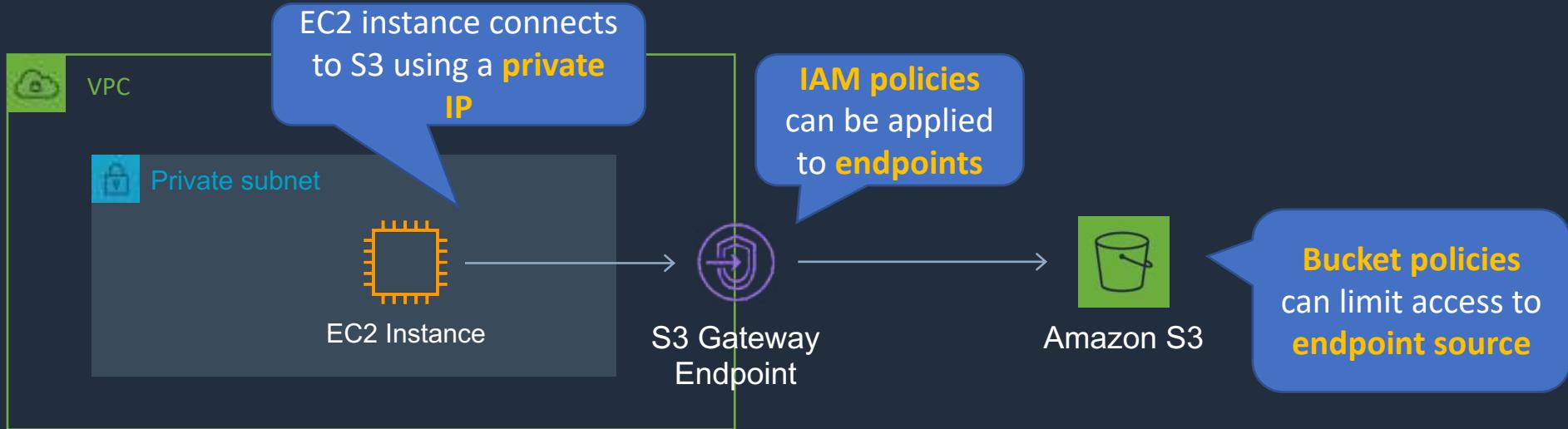


# VPC Interface Endpoints





# VPC Gateway Endpoints



Route Table

Destination	Target
<code>pl-6ca54005 (com.amazonaws.ap-southeast-2.s3, 54.231.248.0/22, 54.231.252.0/24, 52.95.128.0/21)</code>	<code>vpce-ID</code>

A **route table** entry is required with the prefix list for S3 and the **gateway ID**



# VPC Endpoints

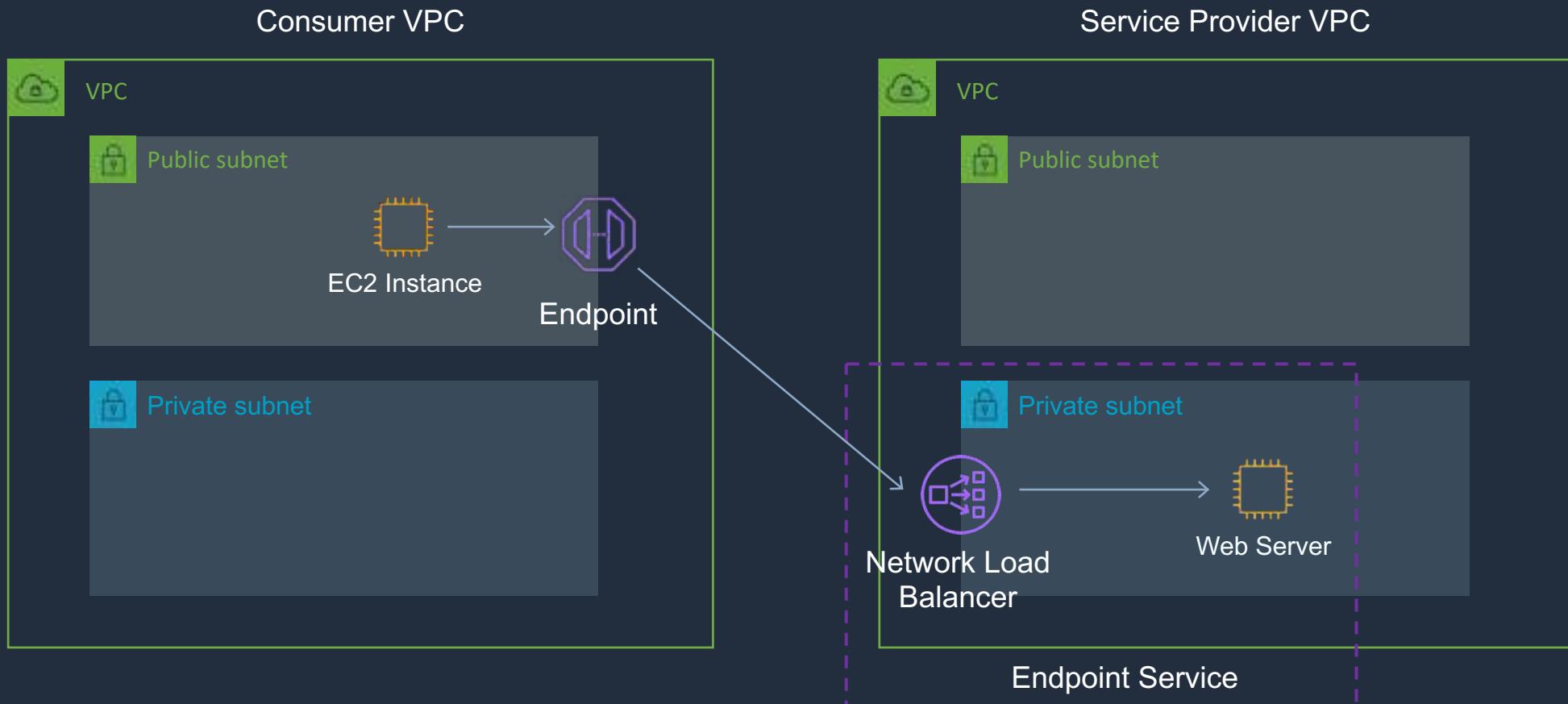
---

	Interface Endpoint	Gateway Endpoint
What	Elastic Network Interface with a Private IP	A gateway that is a target for a specific route
How	Uses DNS entries to redirect traffic	Uses prefix lists in the route table to redirect traffic
Which services	API Gateway, CloudFormation, CloudWatch etc.	Amazon S3, DynamoDB
Security	Security Groups	VPC Endpoint Policies



# Service Provider Model

---

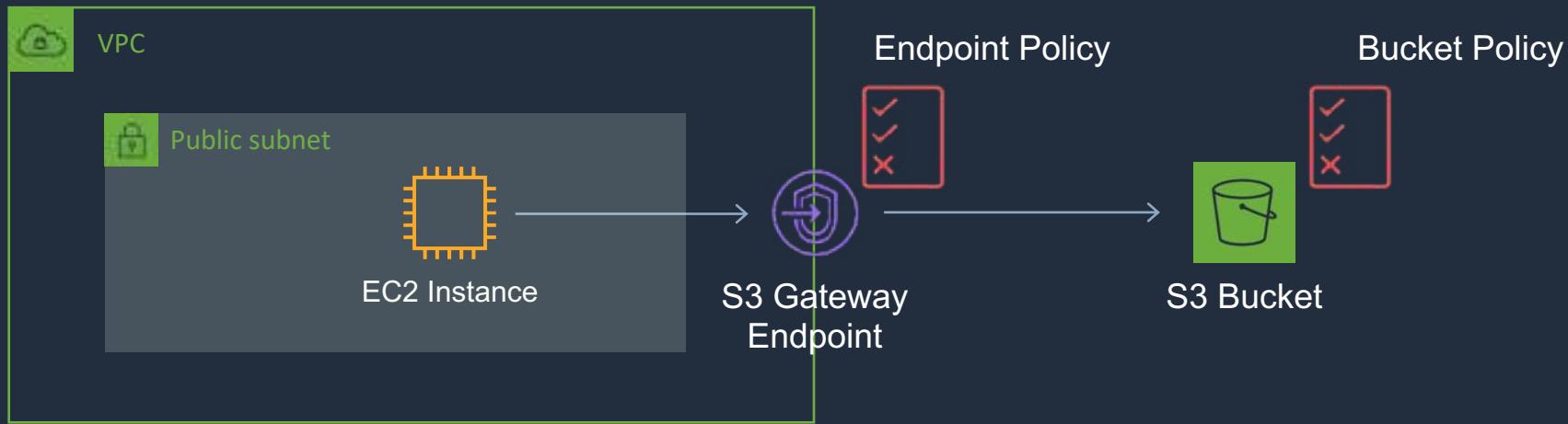


# Create VPC Endpoint





# VPC Gateway Endpoints



Route Table

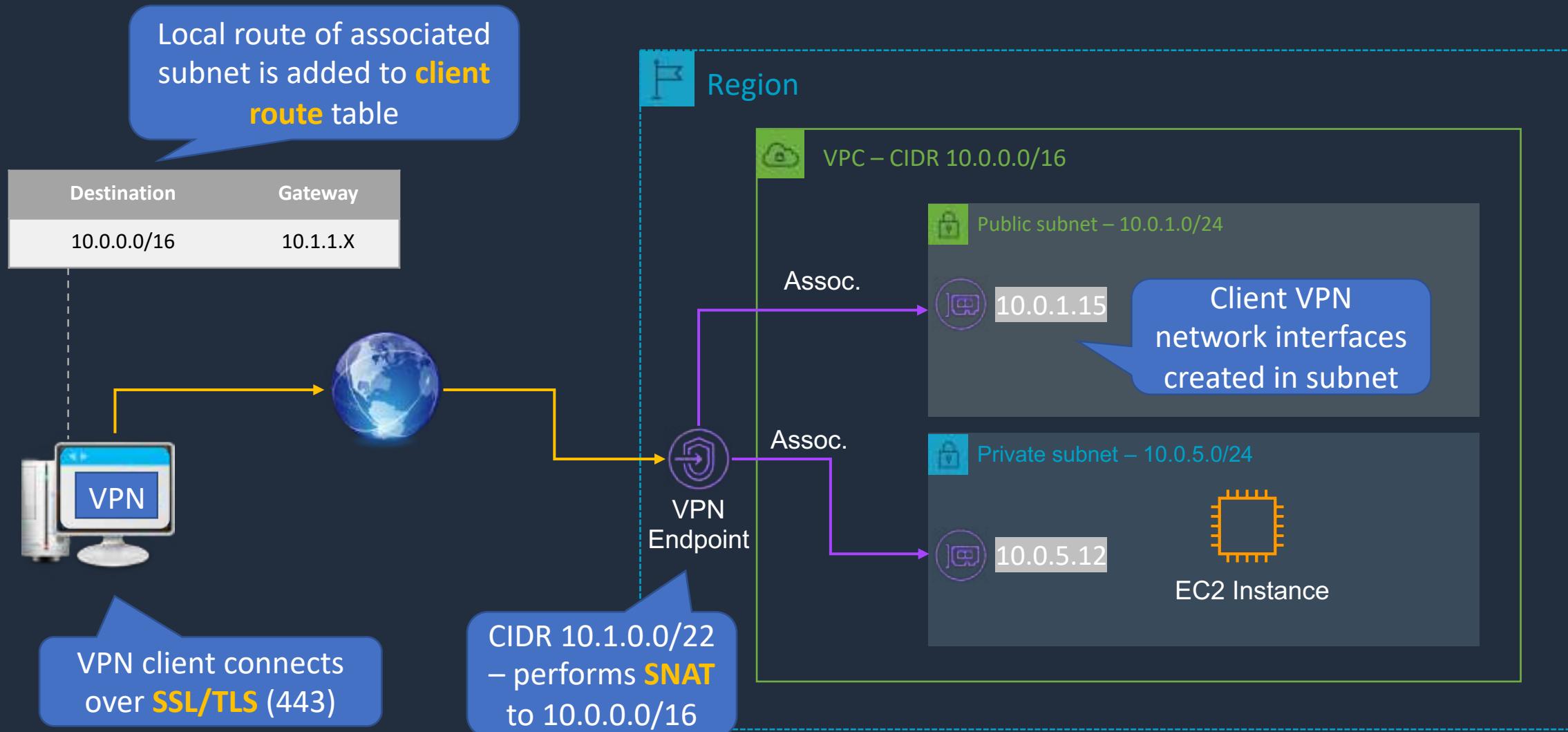
Destination	Target
<code>pl-6ca54005 (com.amazonaws.ap-southeast-2.s3, 54.231.248.0/22, 54.231.252.0/24, 52.95.128.0/21)</code>	<code>vpce-ID</code>

# AWS Client VPN

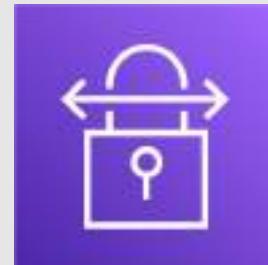




# AWS Client VPN

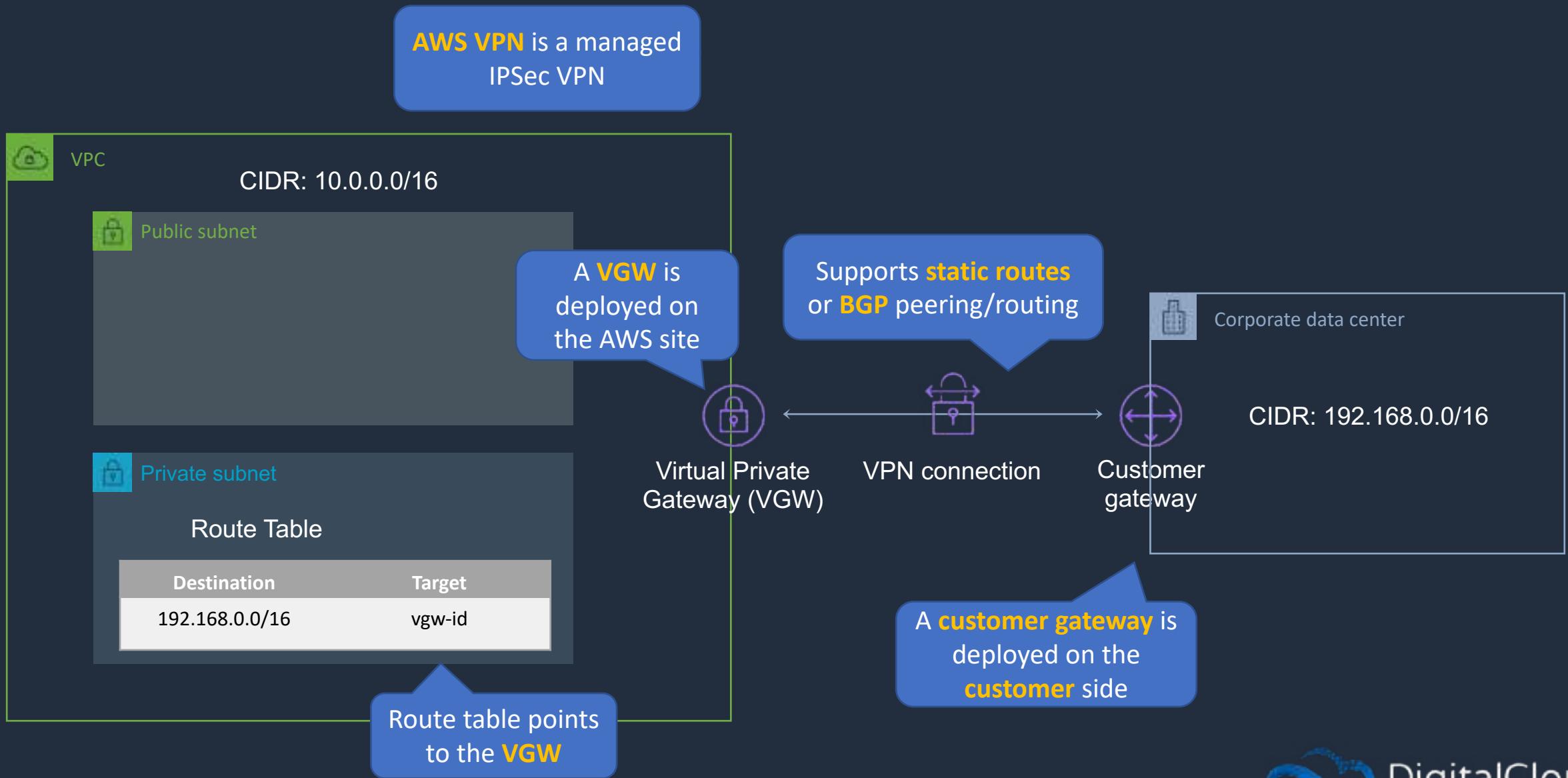


# AWS Site-to-Site VPN





# AWS Site-to-Site VPN

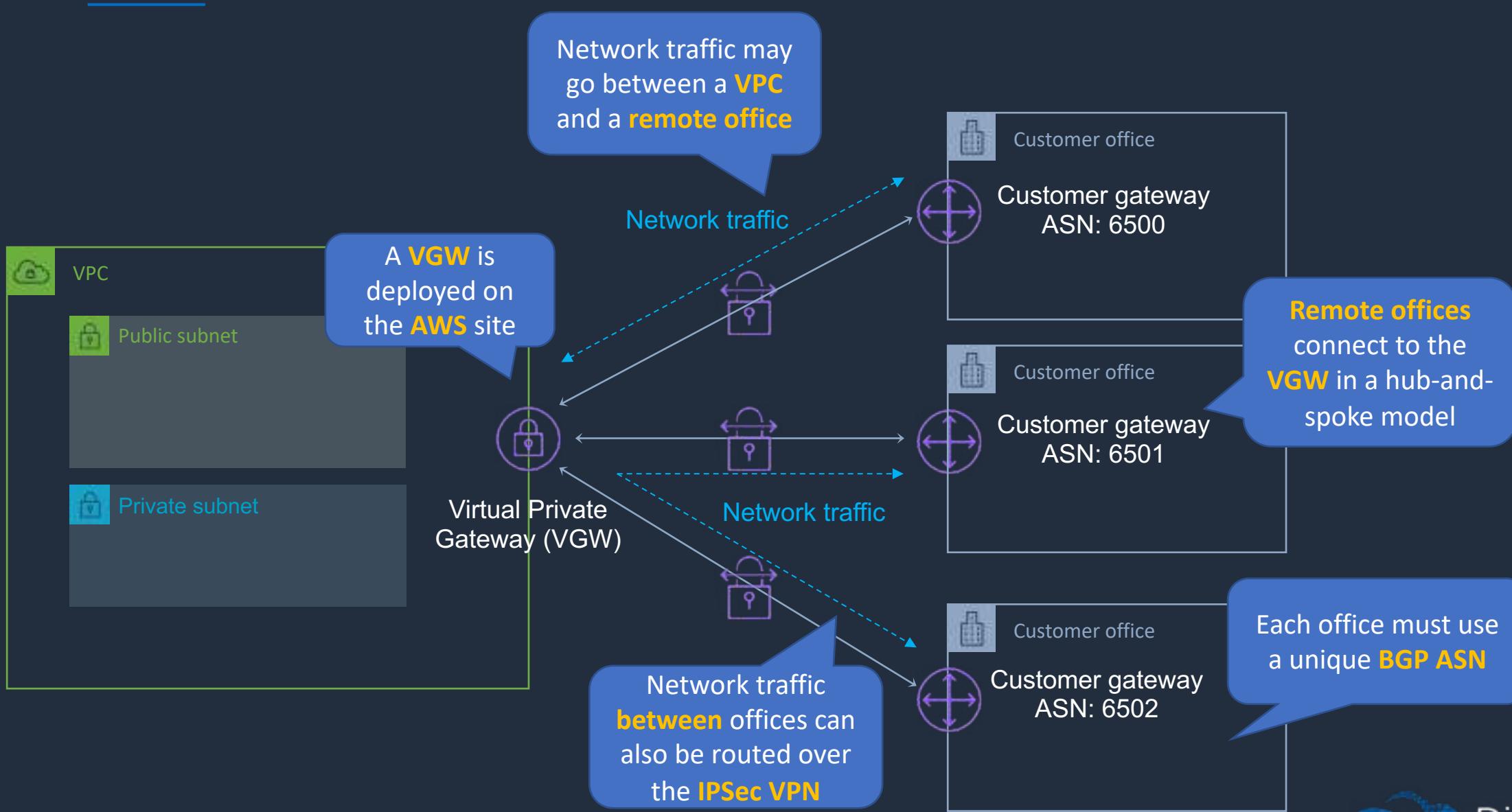


# AWS VPN CloudHub





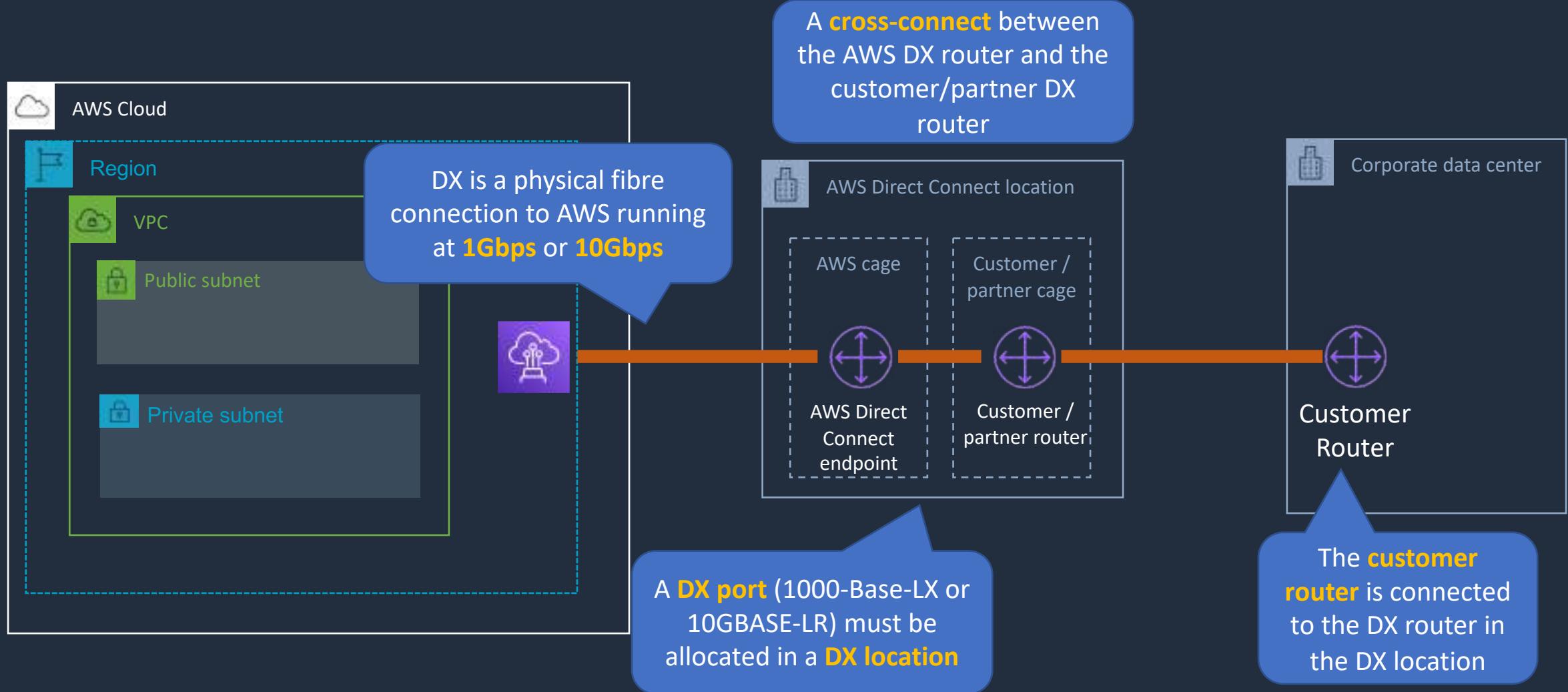
# AWS VPN CloudHub



# AWS Direct Connect (DX)



# AWS Direct Connect (DX)





# AWS Direct Connect Benefits

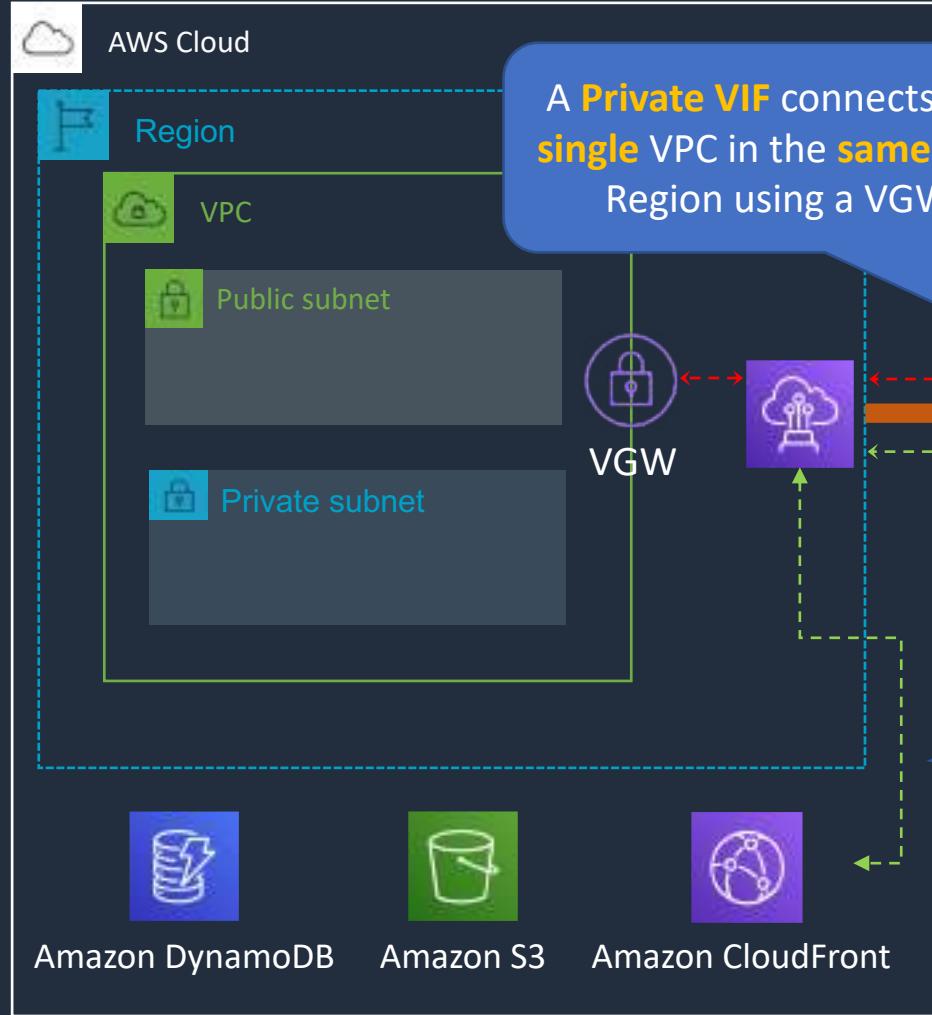
---

---

- **Private** connectivity between AWS and your data center / office
- Consistent network experience – increased **speed/latency** & **bandwidth/throughput**
- Lower costs for organizations that transfer **large** volumes of data



# AWS Direct Connect (DX)



A **VIF** is a virtual interface (802.1Q VLAN) and a **BGP** session

A **Private VIF** connects to a **single** VPC in the **same** AWS Region using a VGW

Private VIF

Public VIF

AWS Direct Connect location

AWS cage

Customer / partner cage

AWS Direct Connect endpoint

Customer / partner router

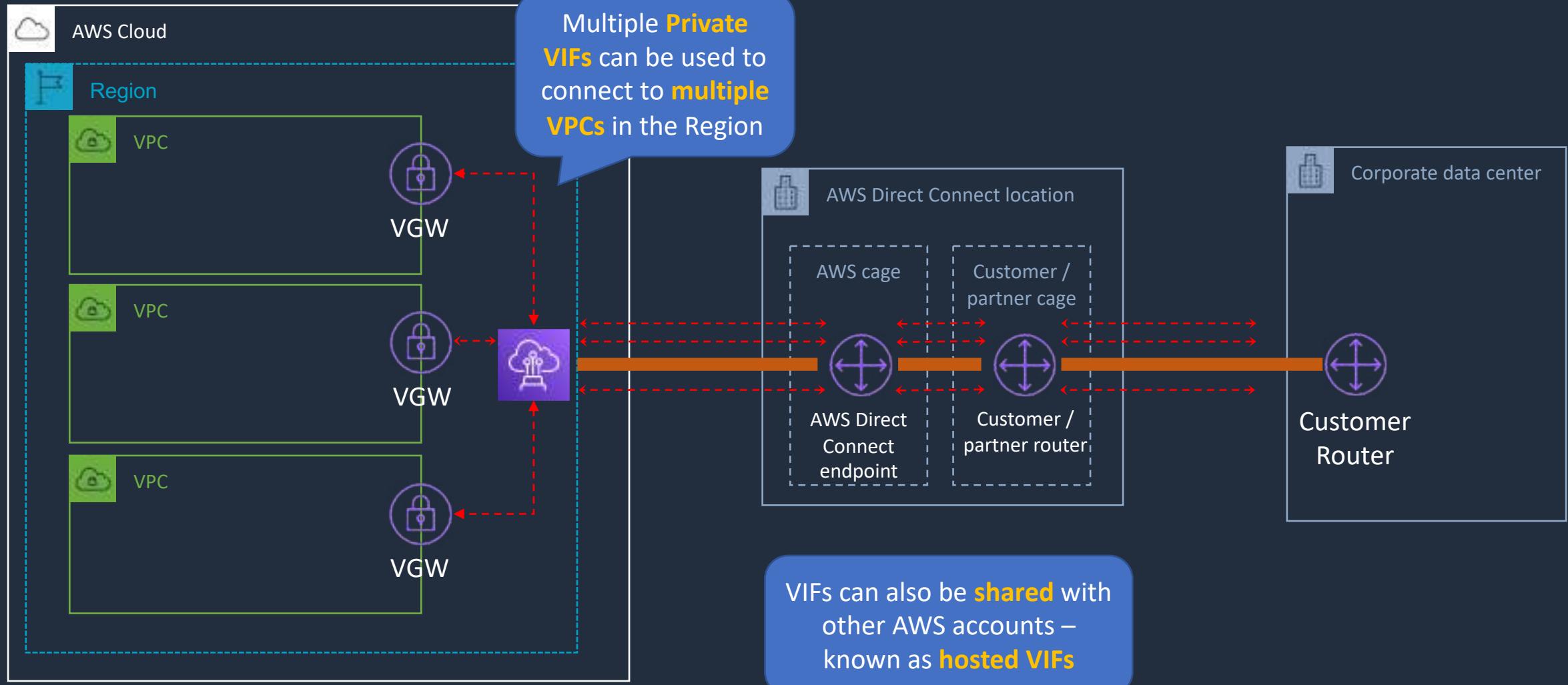
Corporate data center

Customer Router

A **Public VIF** can be used to connect to AWS **Public services** in any Region (but **not** the Internet)



# AWS Direct Connect (DX)



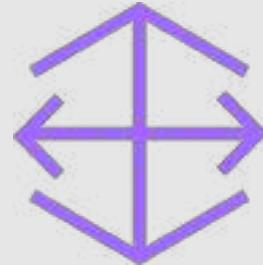


# AWS Direct Connect (DX)

---

- Speeds from 50Mbps to 500Mbps can also be accessed via an APN partner
- 100 Gbps is now featuring in select locations
- DX Connections are **NOT** encrypted!
- Use an **IPSec S2S VPN** connection over a VIF to add encryption in transit

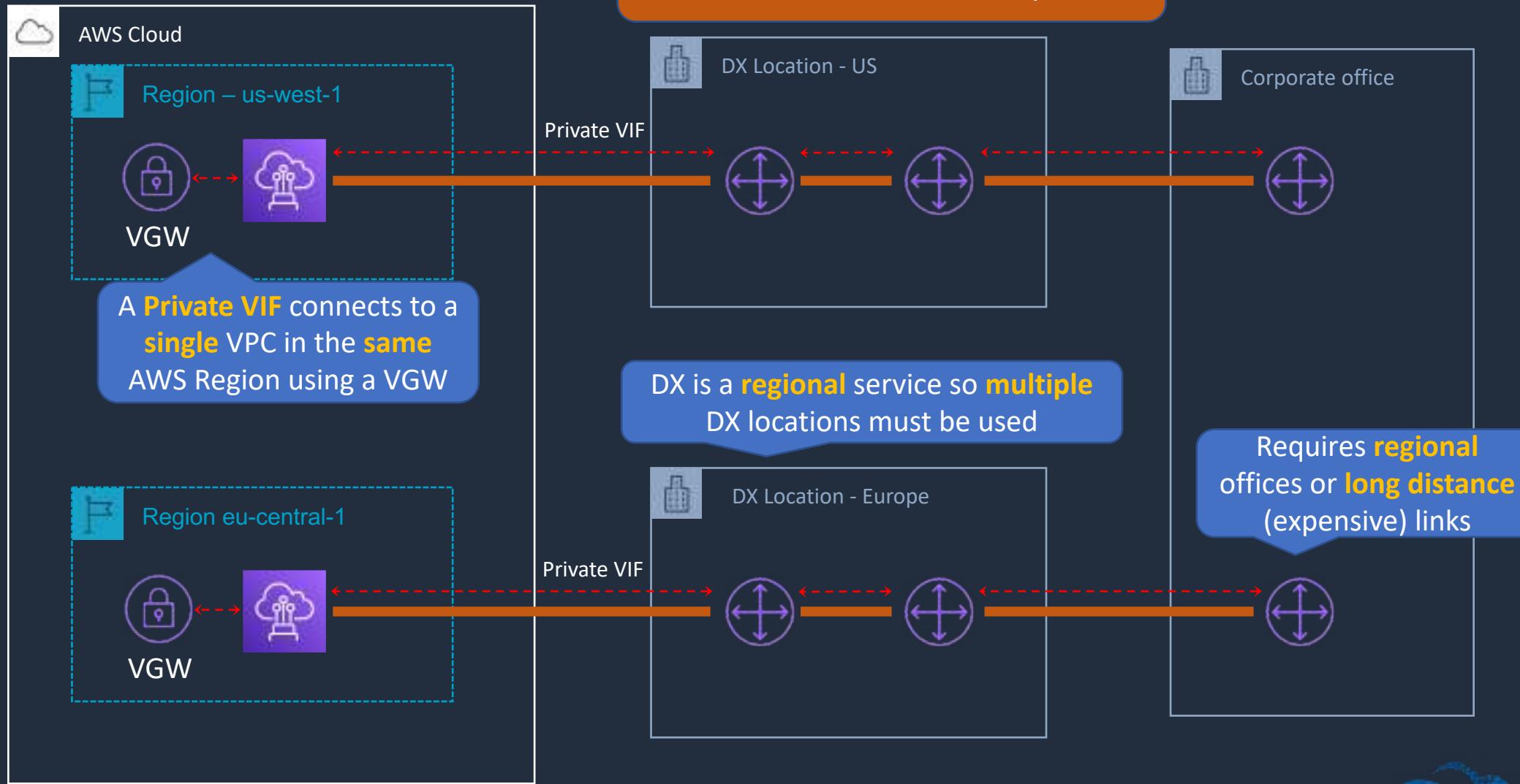
# AWS Direct Connect Gateway





# Direct Connect - Multiple Regions

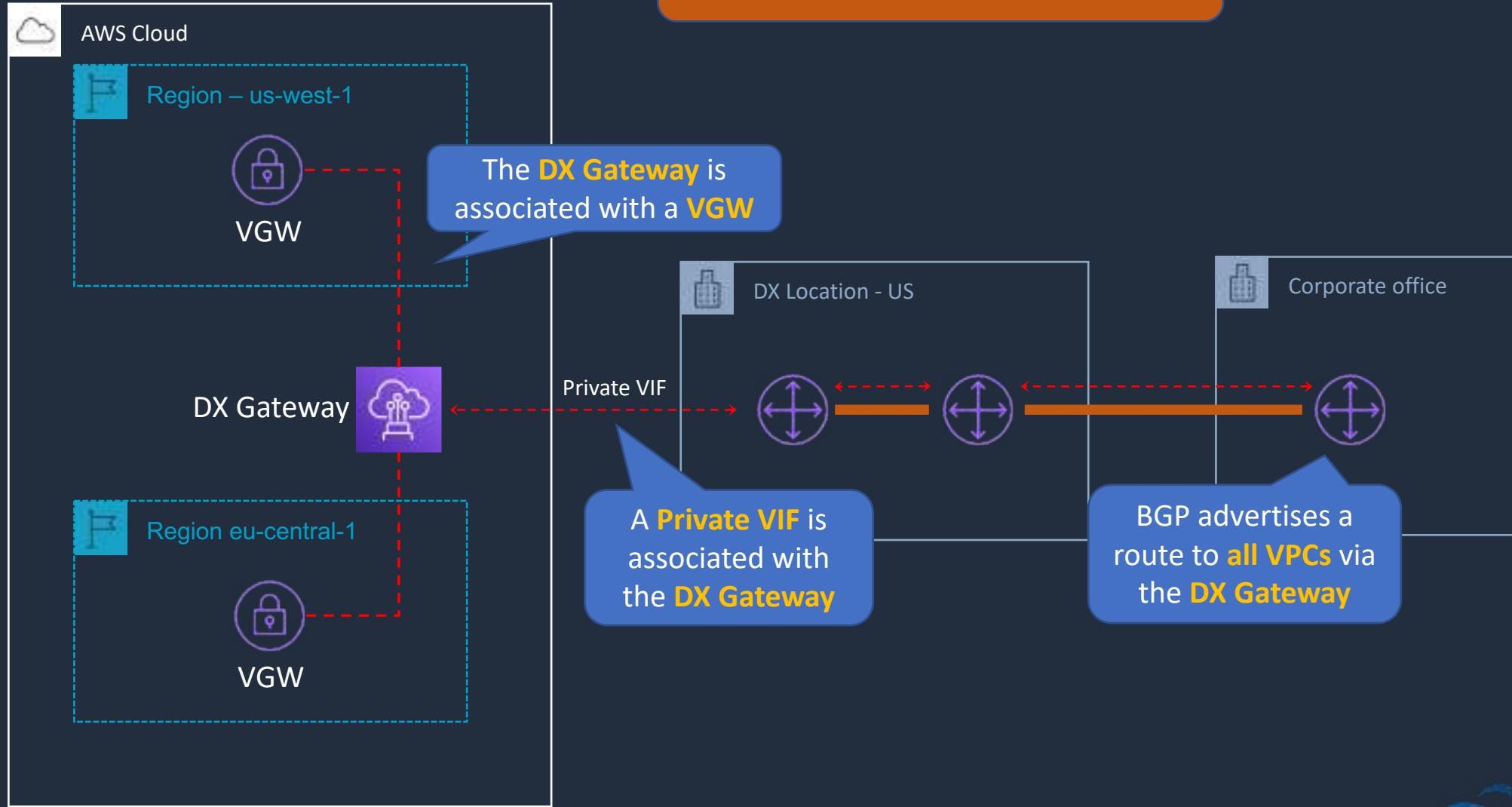
Example architecture **without** AWS Direct Connect Gateway





# Direct Connect - Multiple Regions

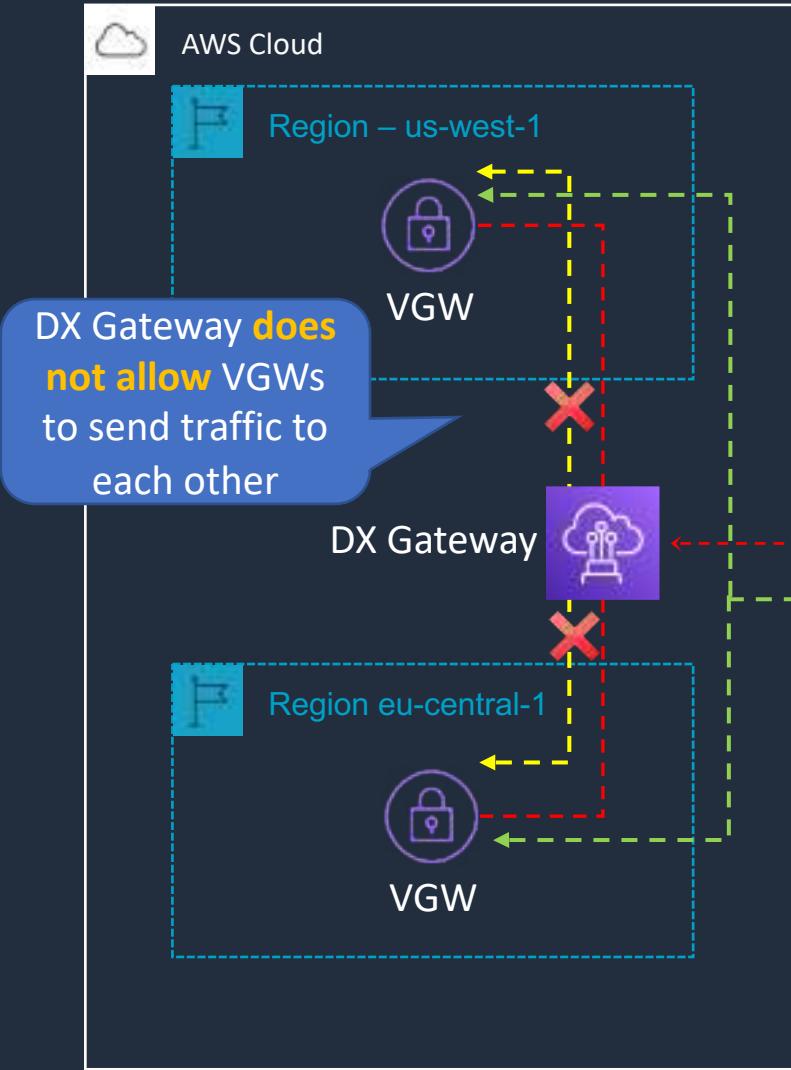
Example architecture **with** AWS Direct Connect Gateway





# Direct Connect - Multiple Regions

Example architecture **with** AWS Direct Connect Gateway

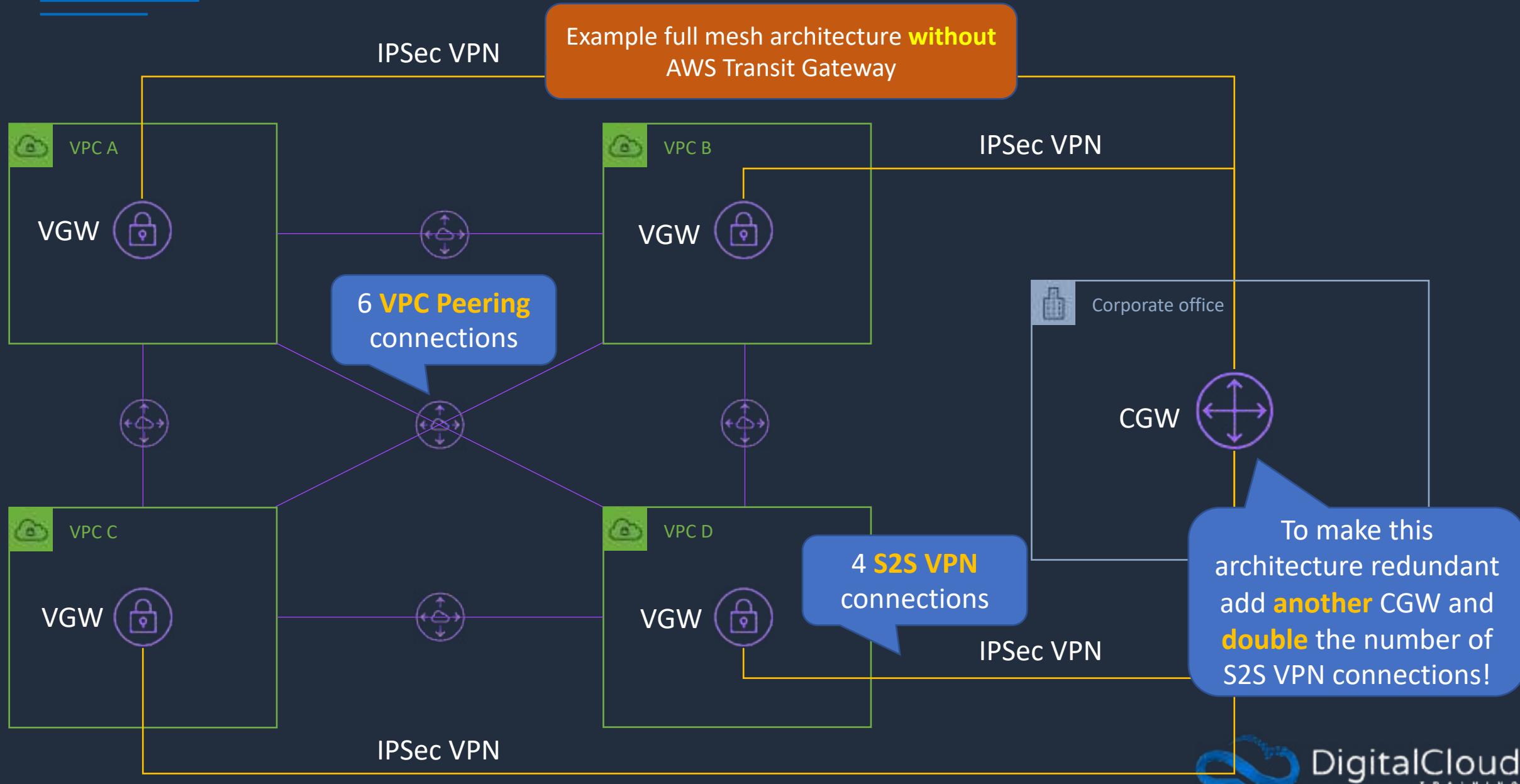


# AWS Transit Gateway



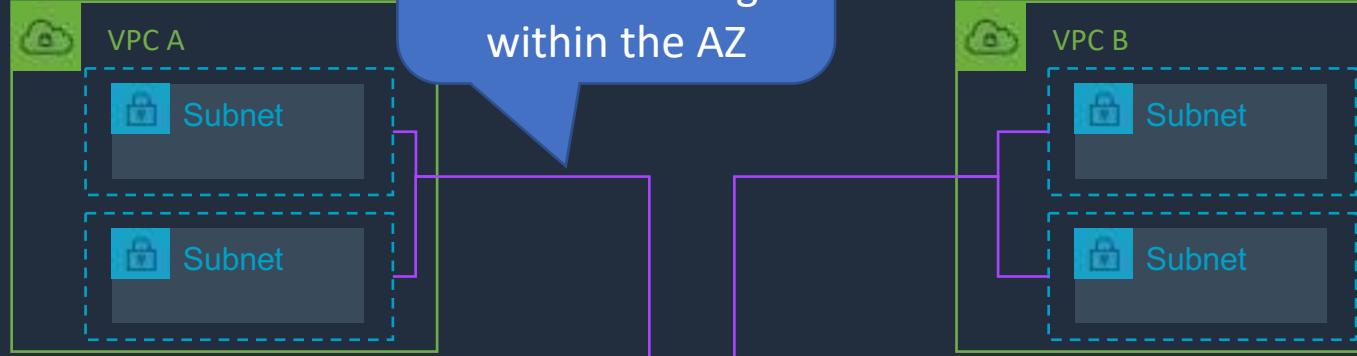


# AWS Transit Gateway



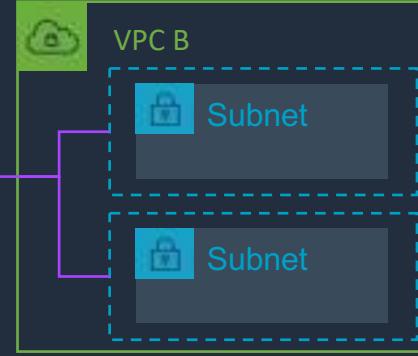


# AWS Transit Gateway

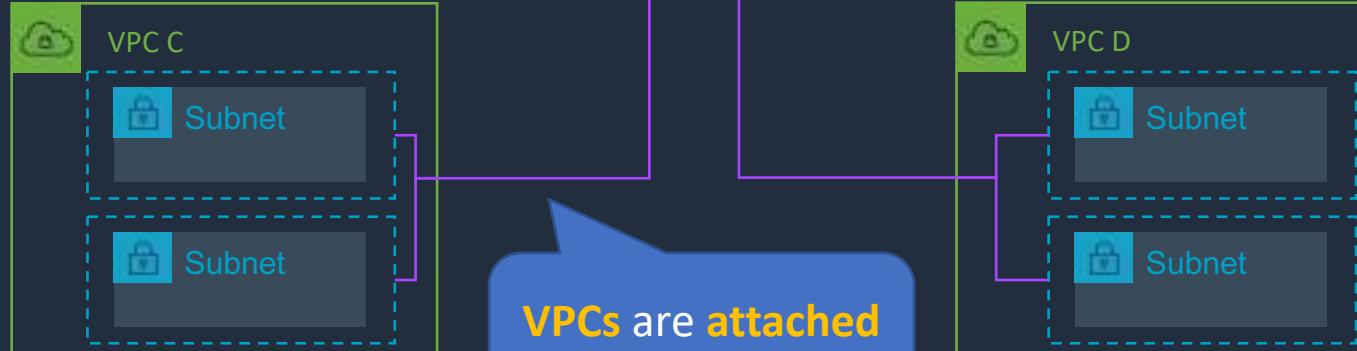


Specify **one subnet** from **each AZ** to enable routing within the AZ

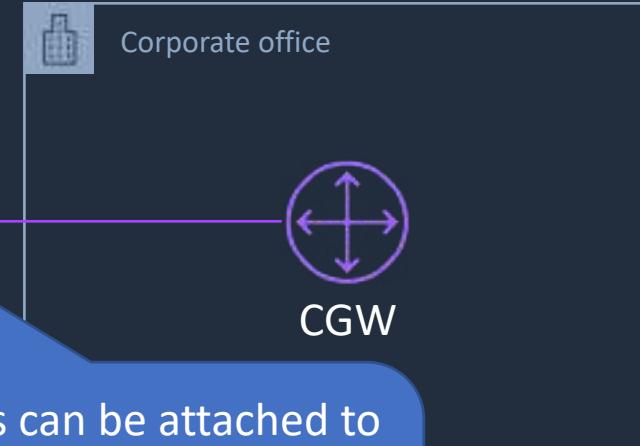
Example full mesh architecture **with** AWS Transit Gateway



**Transit Gateway** is a network transit hub that interconnects **VPCs** and **on-premises** networks



VPCs are **attached** to Transit Gateway

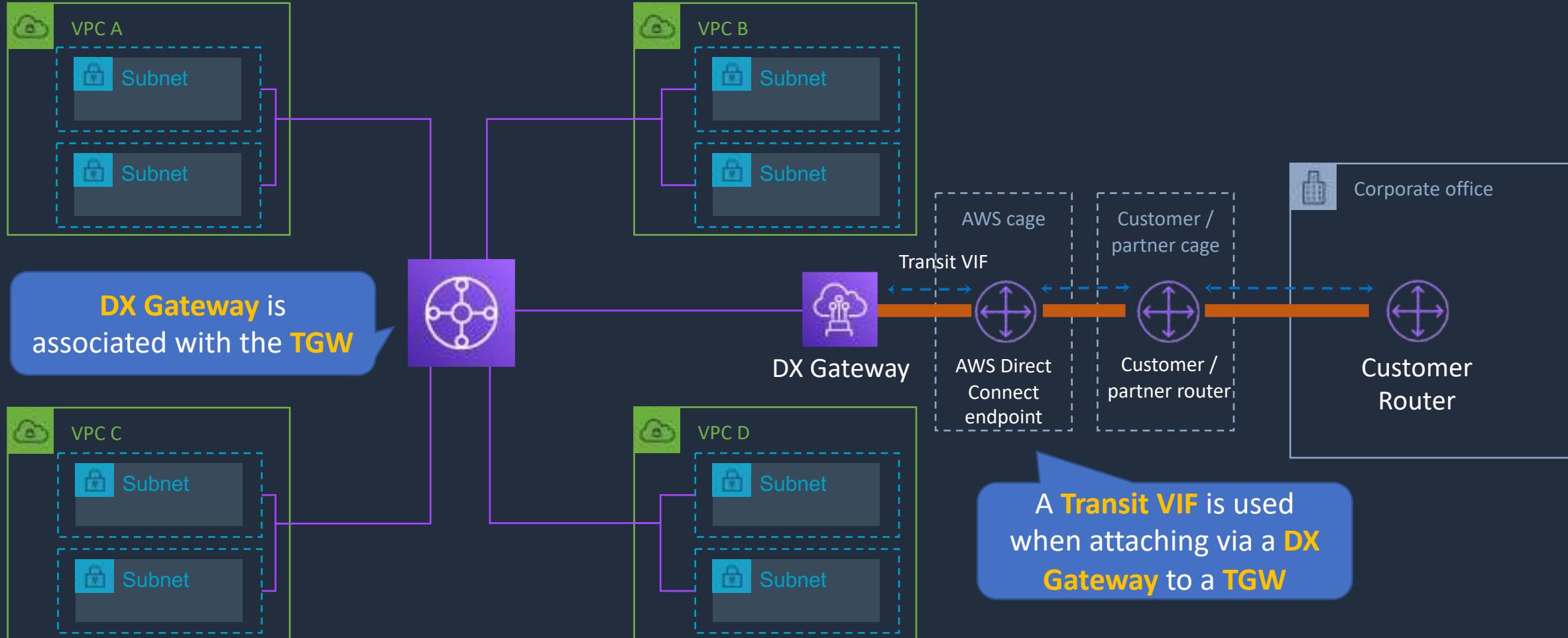


TGWs can be attached to **VPNs, Direct Connect Gateways, 3<sup>rd</sup> party appliances** and **TGWs** in other Regions/accounts



# AWS TGW + DX Gateway

This architecture supports **full transitive** routing between **on-premises**, **TGW** and **VPCs**



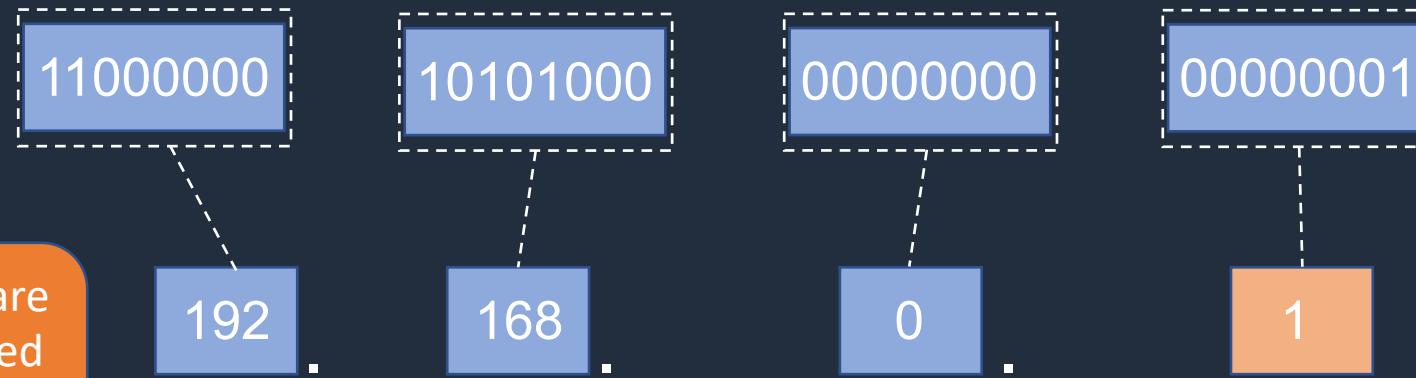
# Using IPv6 in a VPC





# Using IPv6 in a VPC

An IPv4 address is **32 bits** long



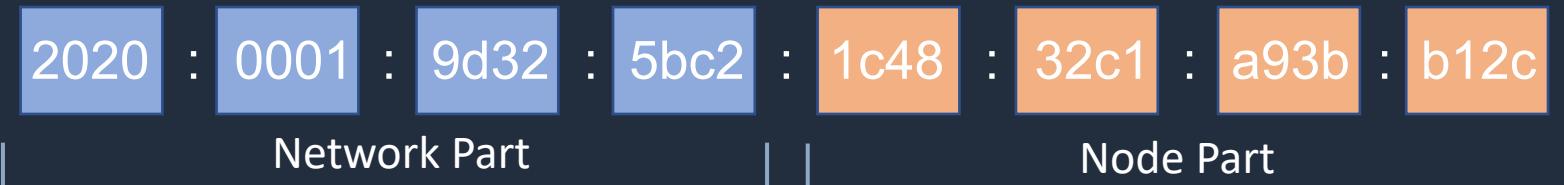
Public **IPv4** addresses are close to being exhausted and **NAT** must be used extensively

IPv4 provides approximately **4.3 billion** addresses



# Using IPv6 in a VPC

An IPv6 address is **128 bits** long



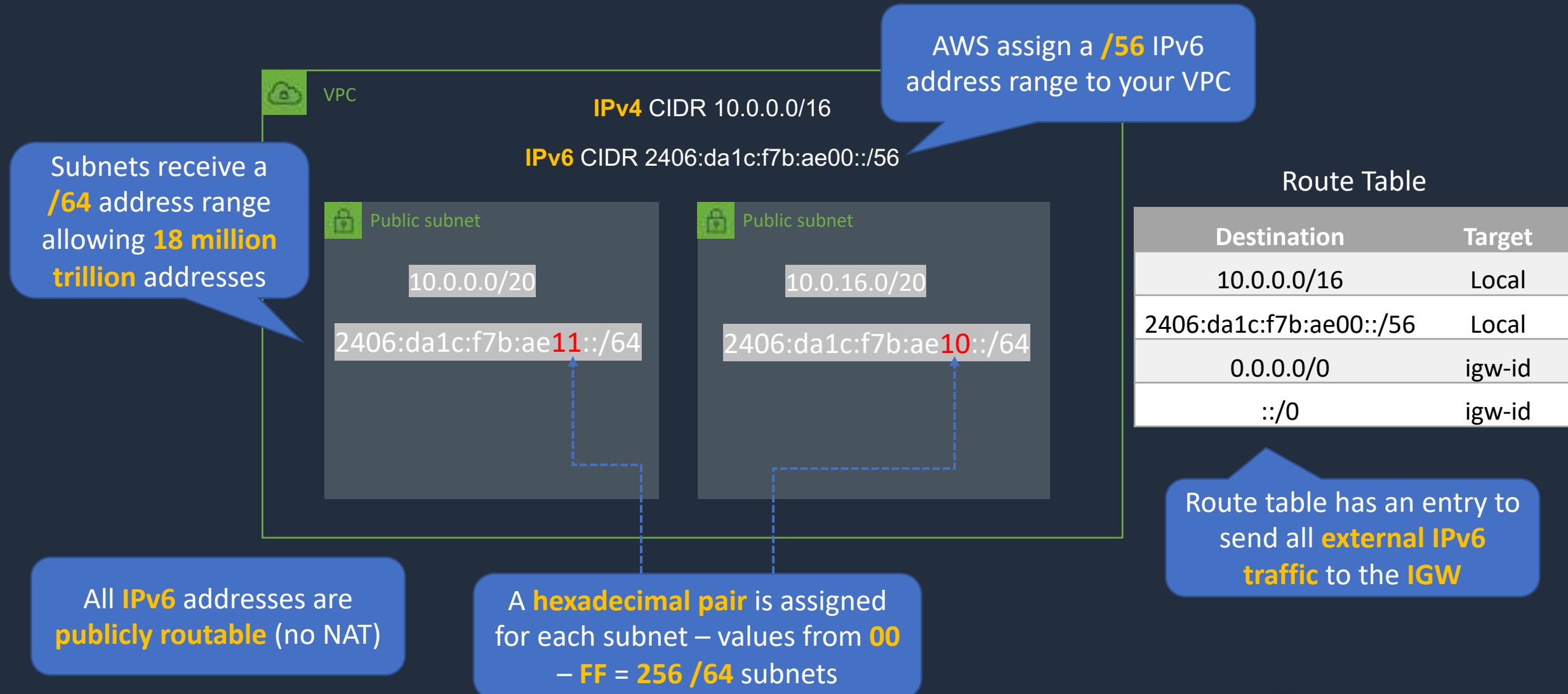
An **IPv6** addresses use **hexadecimal** whereas **IPv4** addresses use **dotted decimal**

That's enough to assign more than **100 IPv6 addresses** to **every atom** on earth!!!

IPv6 provides **340,282,366,920,938,463,463,374,607,431,768,211,456** addresses



# Using IPv6 in a VPC





# Using IPv6 in a VPC



All **IPv6** addresses are **publicly routable** (no NAT)

Route Table	
Destination	Target
172.31.0.0/16	Local
0.0.0.0/0	igw-id
::/0	eo-igw-id

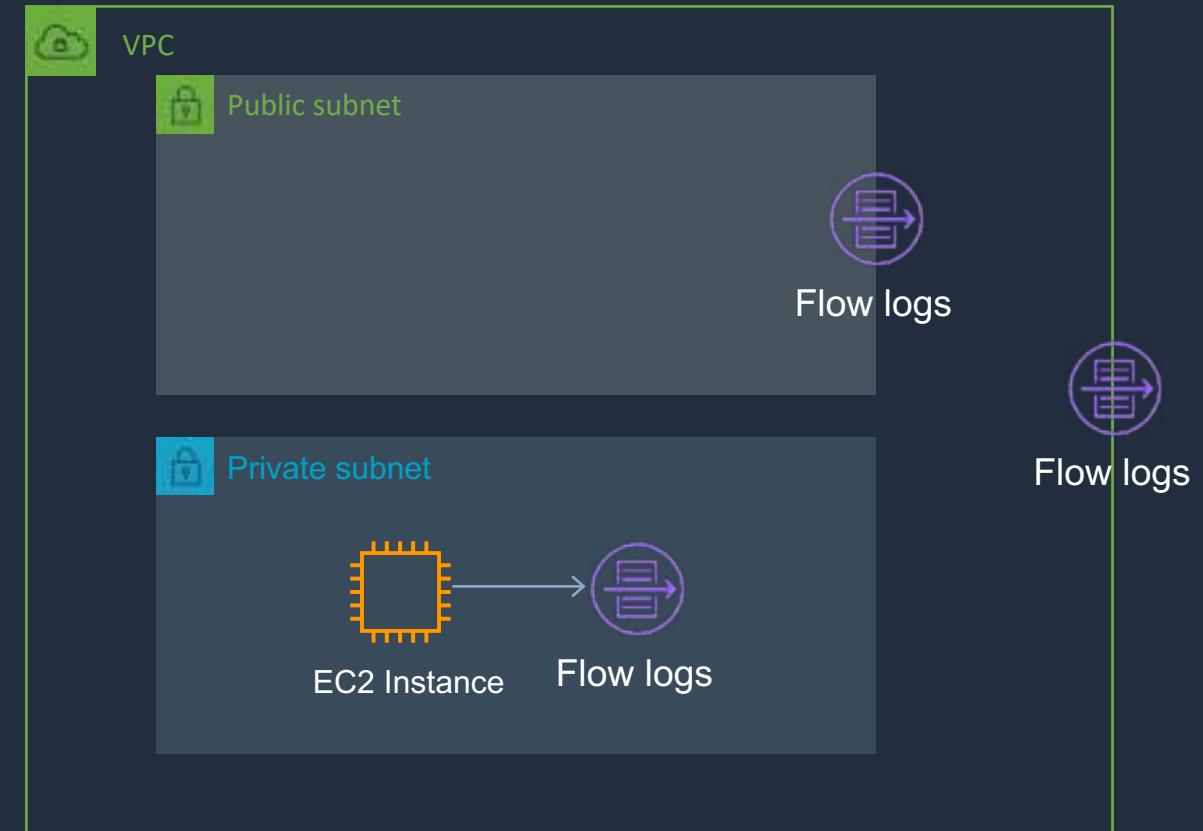
# VPC Flow Logs





# VPC Flow Logs

- Flow Logs capture information about the IP traffic going to and from network interfaces in a VPC
- Flow log data is stored using Amazon CloudWatch Logs or S3
- Flow logs can be created at the following levels:
  - VPC
  - Subnet
  - Network interface



# Architecture Patterns – Amazon VPC





# Architecture Patterns – Amazon VPC

## Requirement

An Amazon S3 bucket must only allow access from EC2 instances in a private subnet using private IPs

Malicious traffic is reaching some EC2 instances in a public subnet from a few identified public IP addresses

A company wants to connect their on-premises data center to AWS and requires consistent performance and encryption

## Solution

Create a VPC endpoint and configure a bucket policy that restricts access to the VPC endpoint ID using the Condition element

Use a Network ACL to deny access based on the source IP addresses

Create an AWS Direct Connect connection and run an AWS S2S VPN over the DX connection to enable encryption



# Architecture Patterns – Amazon VPC

---

## Requirement

A company requires private connectivity between VPCs in different Regions will full redundancy

## Solution

Create VPC peering connections between the VPCs

Several remote office locations should be connected to an Amazon VPC and to each other over the internet with full encryption

Create VPG and attach multiple remote locations in a hub and spoke topology using AWS CloudHub

Microservices app requires instance-level firewall with different rules per application component

Create separate security groups for each application component and configure the appropriate rules



# Architecture Patterns – Amazon VPC

---

## Requirement

A company is using IPv6 addresses with Amazon EC2 and needs to enable outbound internet connectivity

## Solution

Configure an egress-only Internet Gateway

Subnet must be configured that allows internet connectivity using IPv4 with auto-address assignment

Attach an internet gateway to the VPC and update the route table for the new subnet. Enable the auto-assign public IPv4 setting for the subnet

An on-premises data center needs to establish S2S VPN connections to several VPCs in a full mesh architecture

Deploy an AWS Transit Gateway and attach the VPN connection from on-premises and each VPC

# SECTION 7

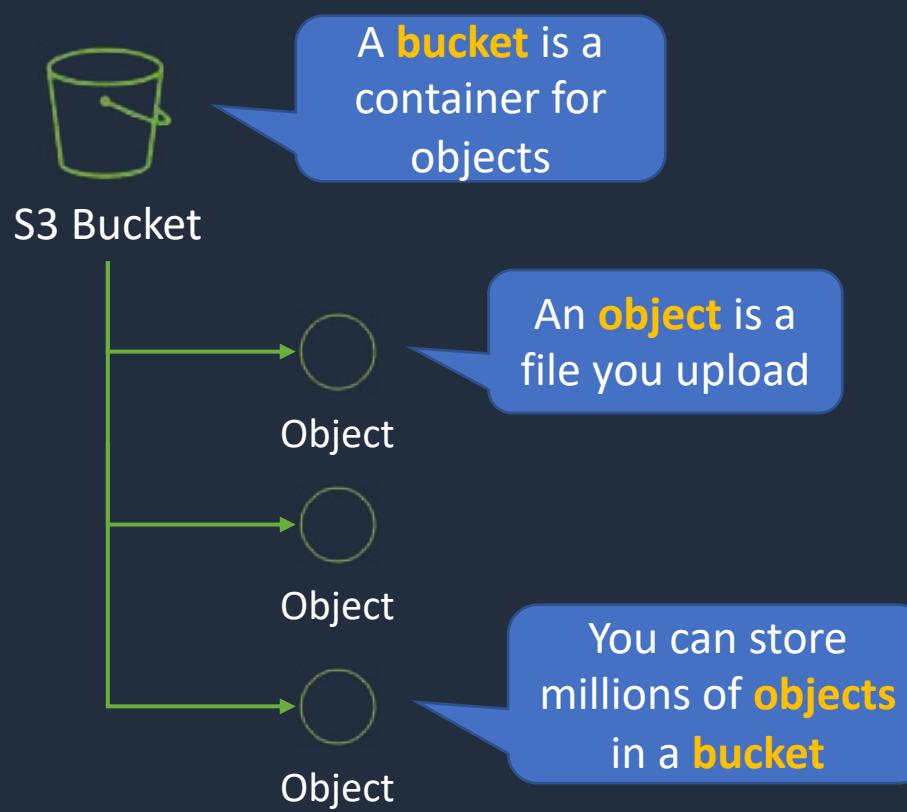
## Amazon Simple Storage Service (S3)

# Amazon S3 Overview





# Amazon Simple Storage Service (S3)



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)

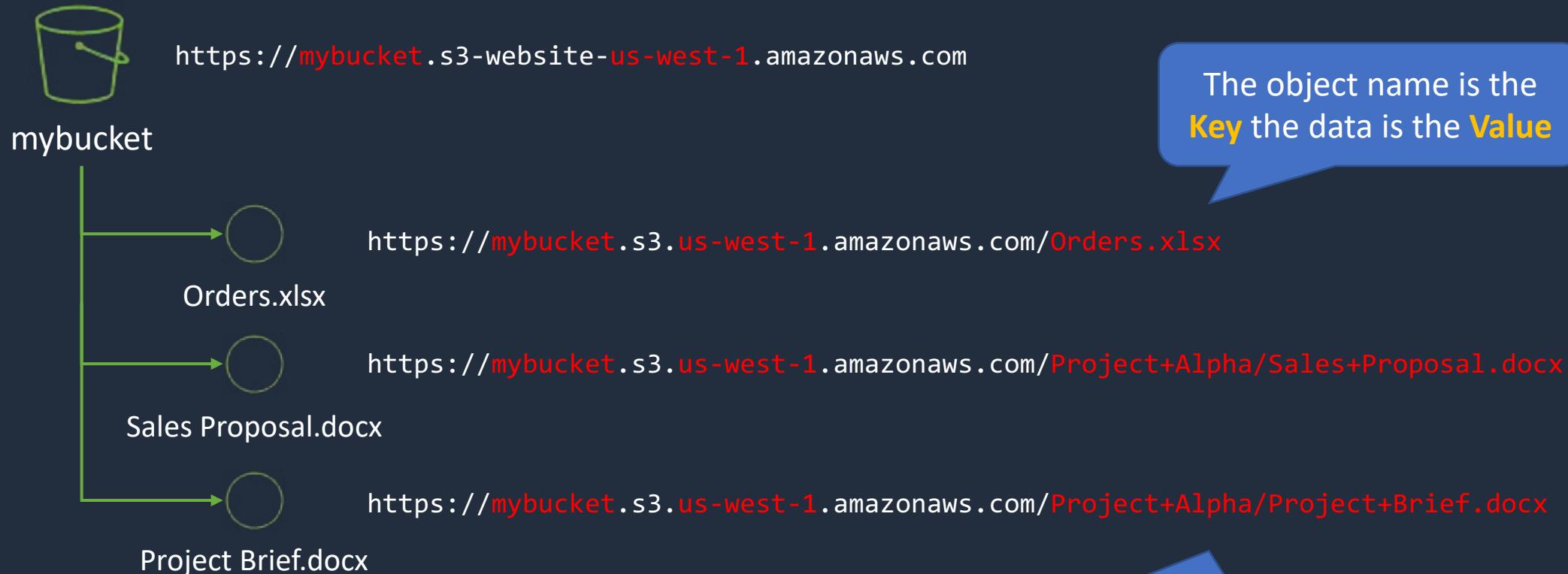
---

---

- You can store any type of file in S3
- Files can be anywhere from 0 bytes to 5 TB
- There is unlimited storage available
- S3 is a universal namespace so **bucket names** must be **unique globally**
- However, you create your buckets within a **REGION**
- It is a best practice to create buckets in regions that are physically closest to your users to reduce latency
- There is no hierarchy for objects within a bucket
- Delivers strong read-after-write consistency



# Buckets, Folders, and Objects





# Buckets, Folders, and Objects

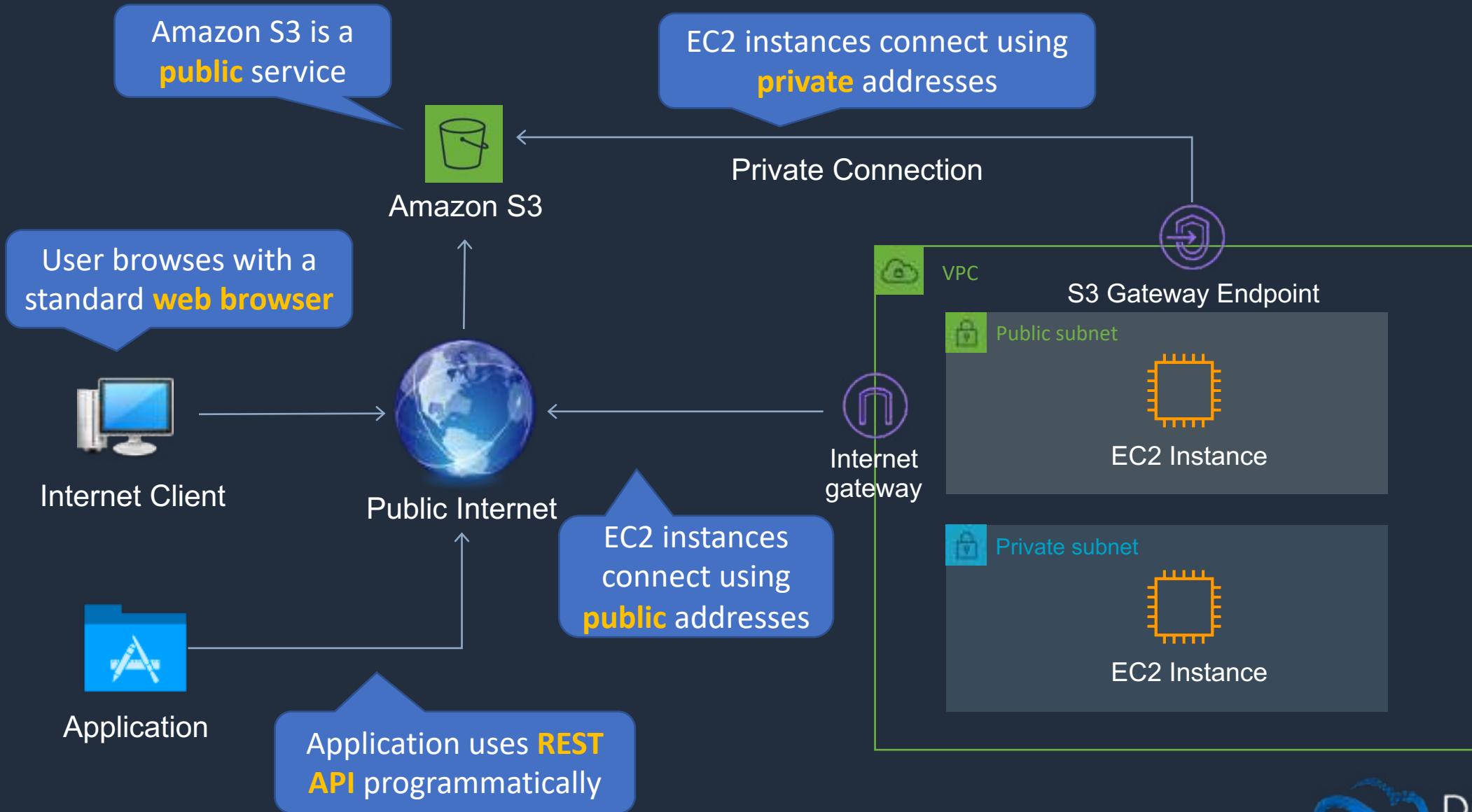
---

---

- Folders **can** be created within folders
- Buckets **cannot** be created within other buckets
- An objects consists of:
  - Key (the name of the object)
  - Version ID
  - Value (actual data)
  - Metadata
  - Subresources
  - Access control information



# Accessing Amazon S3



# Amazon S3 Storage Classes



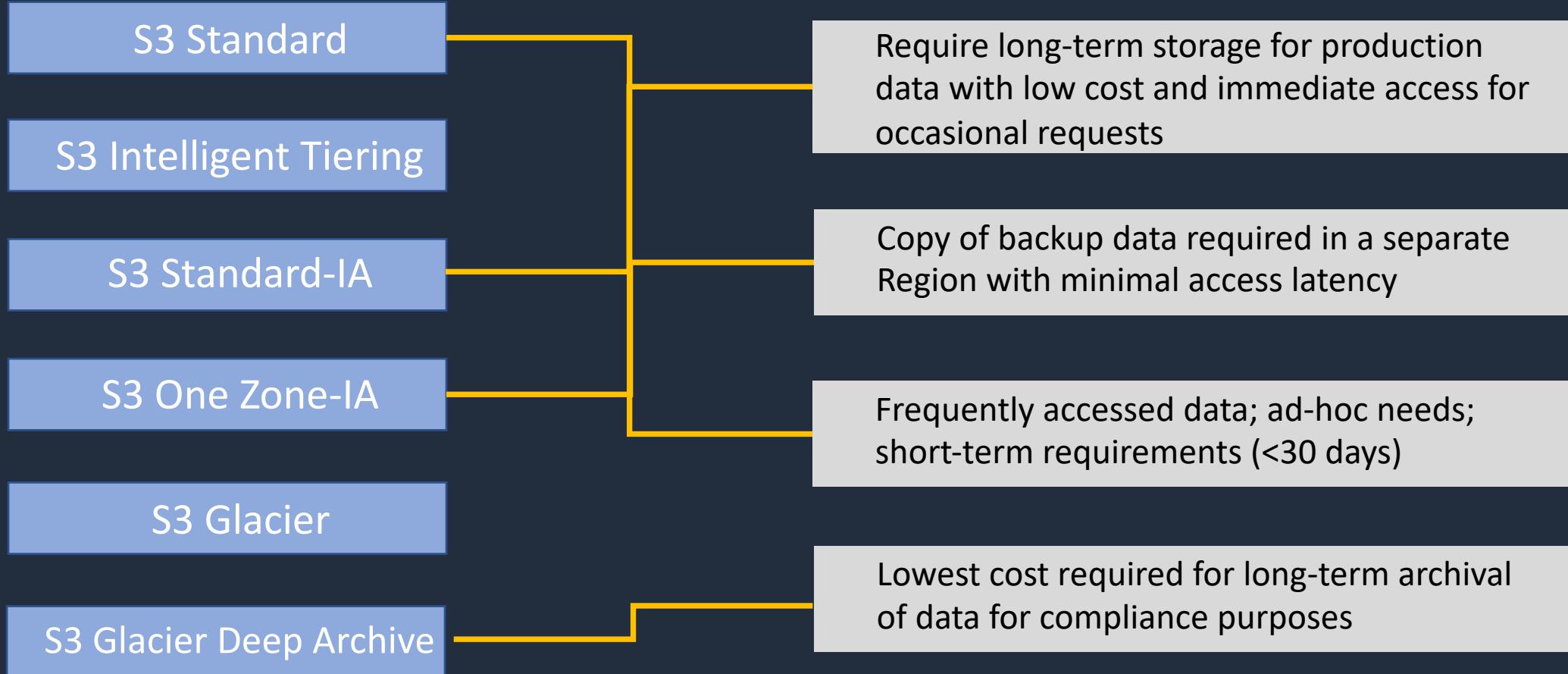


# S3 Storage Classes

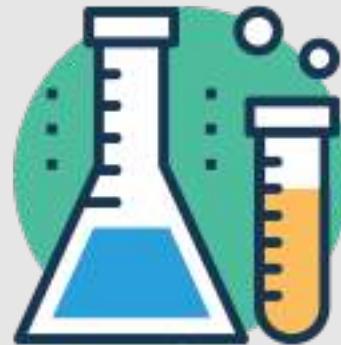
	S3 Standard	S3 Intelligent Tiering	S3 Standard-IA	S3 One Zone-IA	S3 Glacier	S3 Glacier Deep Archive
<b>Designed for durability</b>	99.99999999%	99.99999999%	99.99999999%	99.99999999%	99.99999999%	99.99999999%
<b>Designed for availability</b>	99.99%	99.9%	99.9%	99.5%	99.99%	99.99%
<b>Availability SLA</b>	99.9%	99%	99%	99%	99.9%	99.9%
<b>Availability Zones</b>	≥3	≥3	≥3	1	≥3	≥3
<b>Minimum capacity charge per object</b>	N/A	N/A	128KB	128KB	40KB	40KB
<b>Minimum storage duration charge</b>	N/A	30 days	30 days	30 days	90 days	180 days
<b>Retrieval fee</b>	N/A	N/A	Per GB retrieved	Per GB retrieved	Per GB retrieved	Per GB retrieved
<b>First byte latency</b>	milliseconds	milliseconds	milliseconds	milliseconds	select minutes or hours	select hours
<b>Storage type</b>	Object	Object	Object	Object	Object	Object
<b>Lifecycle transitions</b>	Yes	Yes	Yes	Yes	Yes	Yes



# S3 Storage Class Use Cases



# Create Amazon S3 Bucket



# IAM Policies, Bucket Policies and ACLs





# IAM Policies

- IAM Policies are identity-based policies
- Specify what actions are allowed on what AWS resources

IAM Policies are attached to IAM **users**, **groups**, or **roles**



IAM Policies are written in JSON using the AWS access policy language

The Principal element is not required in the policy



User



Group



Role



# Bucket Policies

- Bucket Policies are resource-based policies
- Can only be attached to Amazon S3 buckets
- Also use the AWS access policy language



```
{  
    "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"  
        }  
    ]  
}
```



# S3 Access Control Lists (ACLs)

---

- Legacy access control mechanism that predates IAM
- AWS generally recommends using **S3 bucket policies** or **IAM policies** rather than ACLs
- Can be attached to a **bucket** or directly to an **object**
- Limited options for grantees and permissions



# When to use each access control mechanism

---

---

- **Use IAM policies if:**

- You need to control access to AWS services other than S3
- You have numerous S3 buckets each with different permissions requirements (IAM policies will be easier to manage)
- You prefer to keep access control policies in the IAM environment

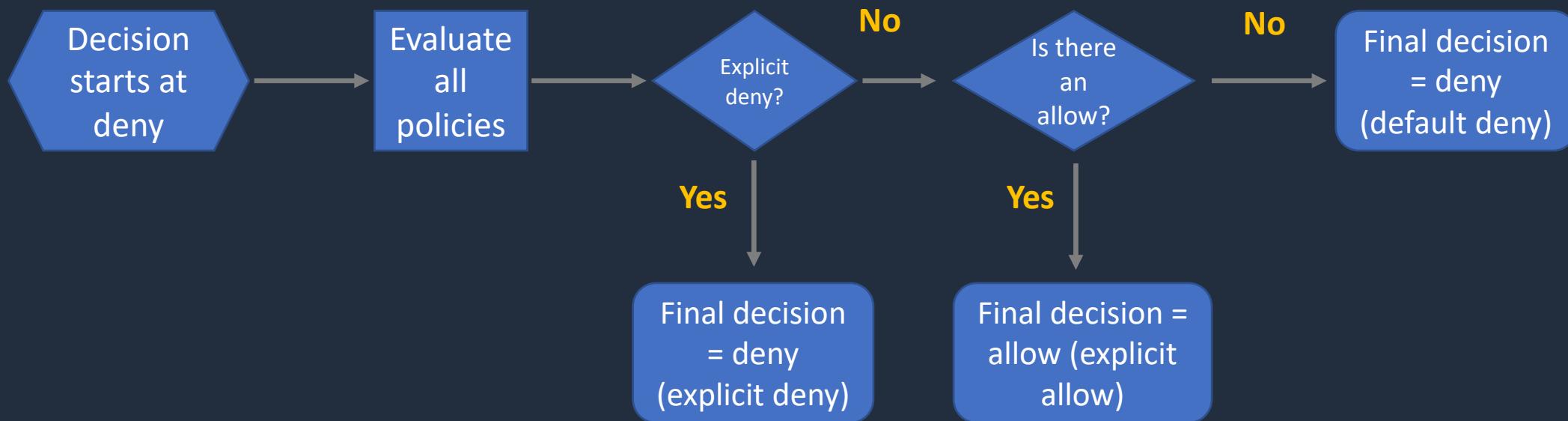
- **Use S3 bucket policies if:**

- You want a simple way to grant cross-account access to your S3 environment, without using IAM roles
- Your IAM policies are reaching the size limits
- You prefer to keep access control policies in the S3 environment



# Authorization Process

---



# Access Control Lists (ACLs)



# Bucket and User Policy Practice



# S3 Versioning, Replication and Lifecycle Rules





# S3 Versioning

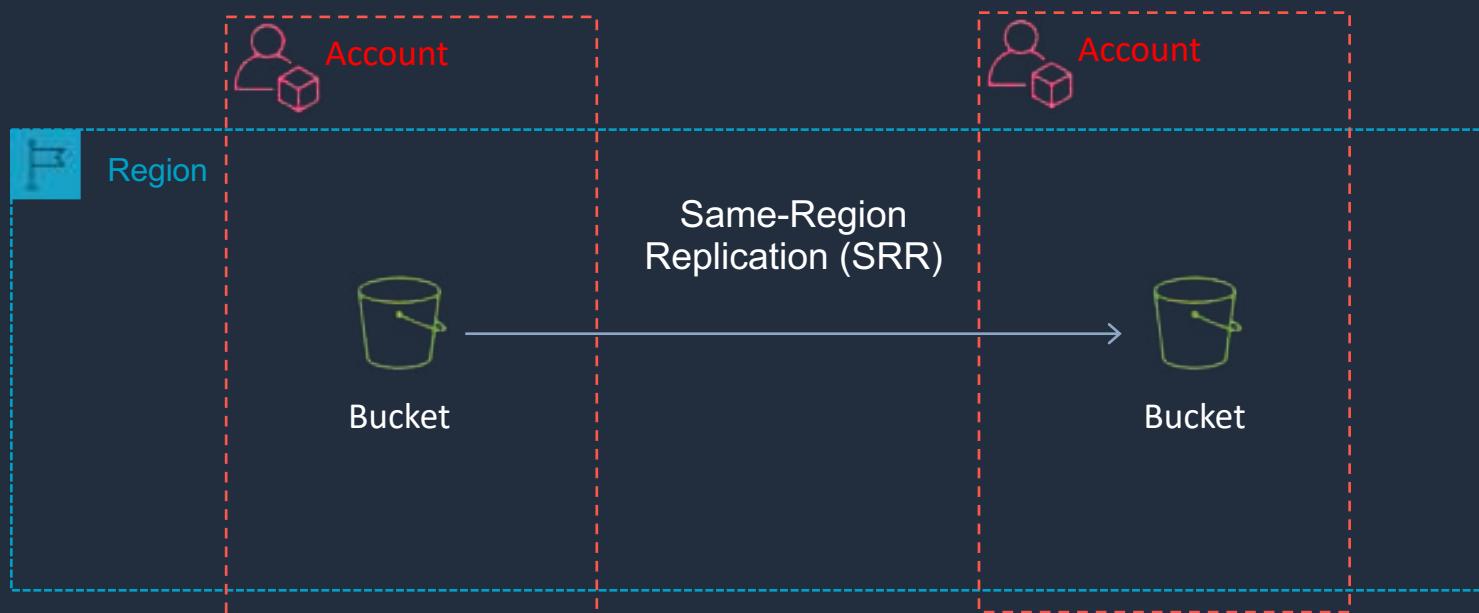
---

- Versioning is a means of keeping multiple variants of an object in the same bucket
- Use versioning to preserve, retrieve, and restore every version of every object stored in your Amazon S3 bucket
- Versioning-enabled buckets enable you to recover objects from **accidental deletion** or **overwrite**



# S3 Replication

## Cross-Region Replication (CRR)



Buckets must have  
**versioning** enabled



# S3 Lifecycle Management

---

There are two types of actions:

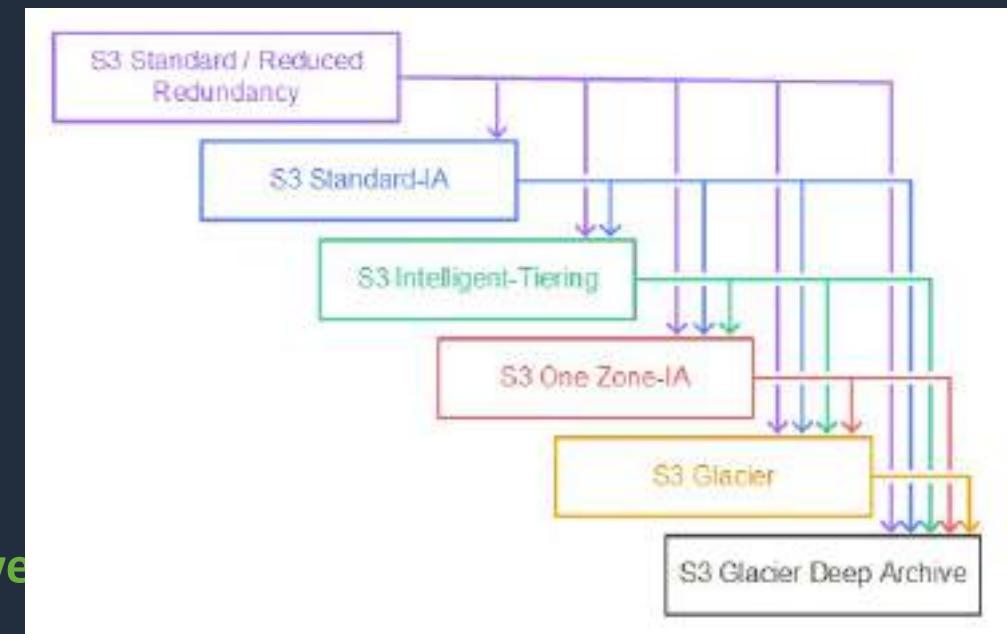
- **Transition actions** - Define when objects transition to another storage class
- **Expiration actions** - Define when objects expire (deleted by S3)



# S3 LM: Supported Transitions

You can transition from the following:

- The **S3 Standard** storage class to any other storage class
- Any storage class to the **S3 Glacier** or **S3 Glacier Deep Archive** storage classes
- The **S3 Standard-IA** storage class to the **S3 Intelligent-Tiering** or **S3 One Zone-IA** storage classes
- The **S3 Intelligent-Tiering** storage class to the **S3 One Zone-IA** storage class
- The **S3 Glacier** storage class to the **S3 Glacier Deep Archive** storage class

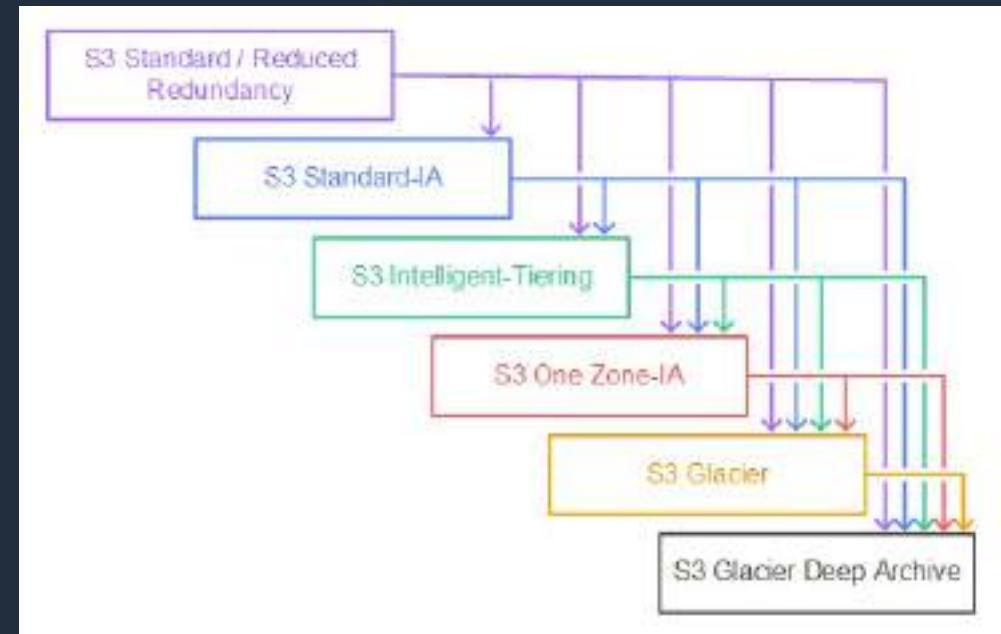




# S3 LM: Unsupported Transitions

You can't transition from the following:

- Any storage class to the **S3 Standard** storage class
- Any storage class to the **Reduced Redundancy** storage class
- The **S3 Intelligent-Tiering** storage class to the **S3 Standard-IA** storage class
- The **S3 One Zone-IA** storage class to the **S3 Standard-IA** or **S3 Intelligent-Tiering** storage classes



# Versioning and Replication



# Lifecycle Rules

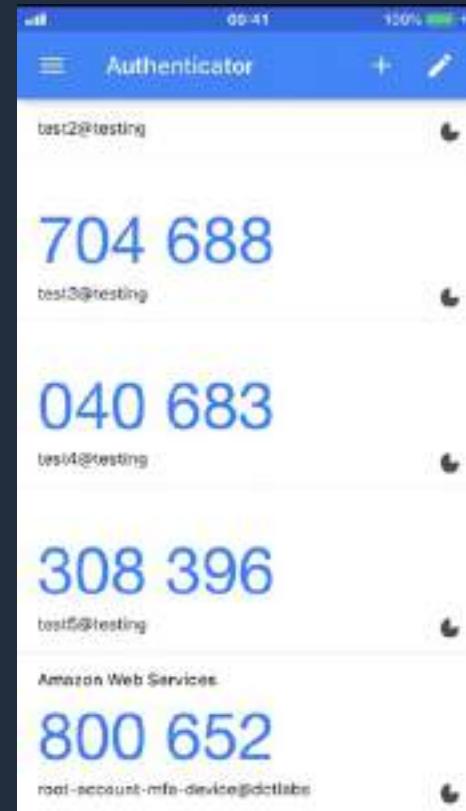


# MFA with Amazon S3



# S3 Multi-Factor Authentication Delete (MFA Delete)

- Adds MFA requirement for bucket owners to the following operations:
  - Changing the versioning state of a bucket
  - Permanently deleting an object version
- The **x-amz-mfa** request header must be included in the above requests
- The second factor is a token generated by a hardware device or software program
- Requires **versioning** to be enabled on the bucket



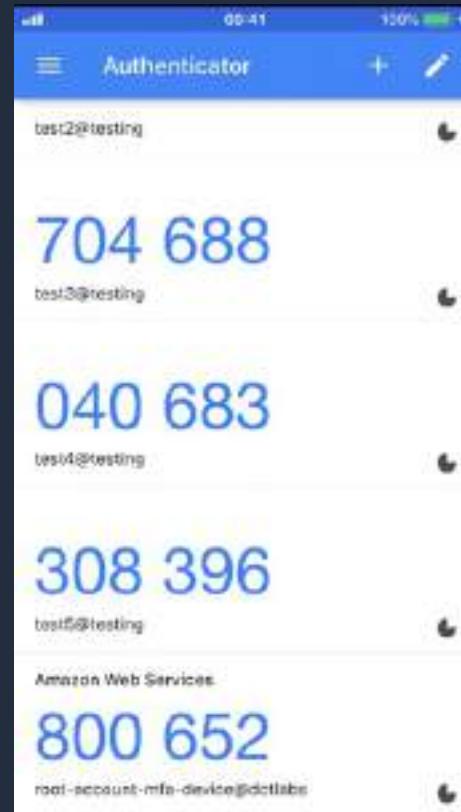
# S3 Multi-Factor Authentication Delete (MFA Delete)

- **Versioning** can be enabled by:

- Bucket owners (root account)
- AWS account that created the bucket
- Authorized IAM users

- **MFA delete** can be enabled by:

- Bucket owner (root account)



# MFA-Protected API Access

- Used to enforce another authentication factor (MFA code) when accessing AWS resources (not just S3)
- Enforced using the **aws:MultiFactorAuthAge** key in a bucket policy:

```
{  
    "Version": "2012-10-17",  
    "Id": "123",  
    "Statement": [  
        {  
            "Sid": "",  
            "Effect": "Deny",  
            "Principal": "*",  
            "Action": "s3:*",  
            "Resource": "arn:aws:s3:::examplebucket/securedocuments/*",  
            "Condition": { "Null": { "aws:MultiFactorAuthAge": true } }  
        }  
    ]  
}
```

Denies any API operation  
that is not authenticated  
using MFA

# S3 Encryption





# S3 Encryption

## Server-side encryption with S3 managed keys (SSE-S3)



- S3 managed keys
- Unique object keys
- Master key
- AES 256



Encryption /  
decryption



## Server-side encryption with AWS KMS managed keys (SSE-KMS)



- KMS managed keys
- Customer master keys
- CMK can be customer generated



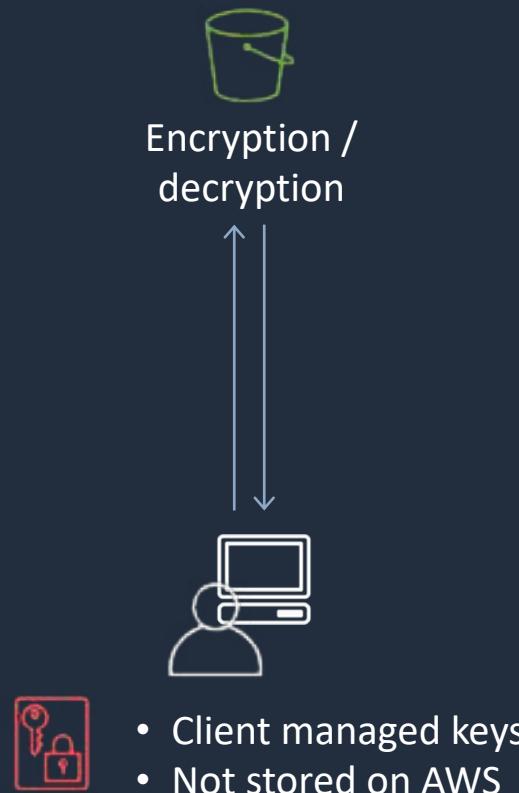
Encryption /  
decryption



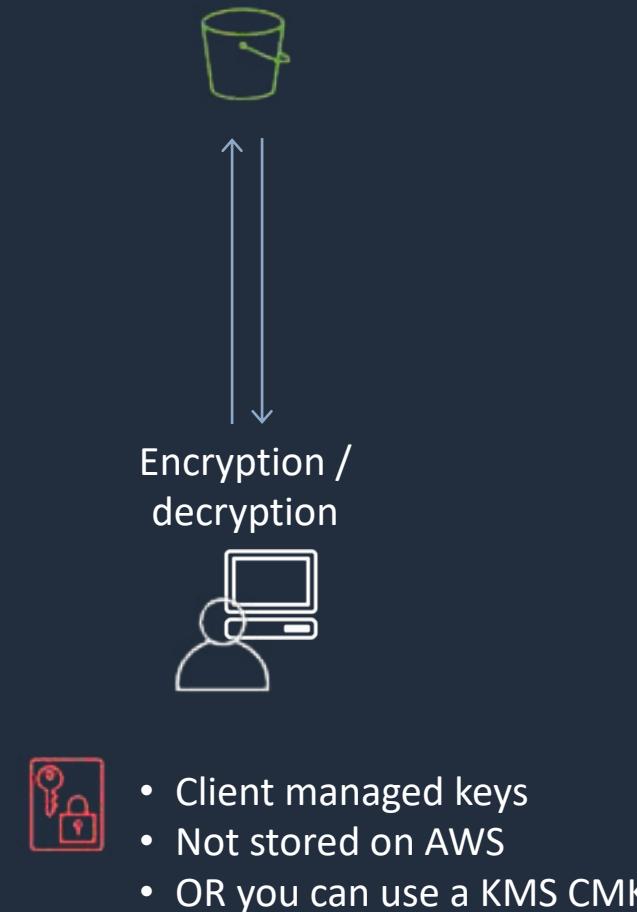


# S3 Encryption

## Server-side encryption with client provided keys (SSE-C)



## Client-side encryption





# S3 Default Encryption

---

- Amazon S3 default encryption provides a way to set the default encryption behavior for an S3 bucket
- You can set default encryption on a bucket so that all new objects are encrypted when they are stored in the bucket
- The objects are encrypted using server-side encryption
- Amazon S3 encrypts objects before saving them to disk and decrypts them when the objects are downloaded
- There is no change to the encryption of objects that existed in the bucket before default encryption was enabled



# Prevent uploads of unencrypted objects

```
{  
    "Version": "2012-10-17",  
    "Id": "PutObjPolicy",  
    "Statement": [  
        {  
            "Sid": "DenyIncorrectEncryptionHeader",  
            "Effect": "Deny",  
            "Principal": "*",  
            "Action": "s3:PutObject",  
            "Resource": "arn:aws:s3:::<bucket_name>/*",  
            "Condition": {  
                "StringNotEquals": {  
                    "s3:x-amz-server-side-encryption": "AES256"  
                }  
            }  
        },  
        {  
            "Sid": "DenyUnEncryptedObjectUploads",  
            "Effect": "Deny",  
            "Principal": "*",  
            "Action": "s3:PutObject",  
            "Resource": "arn:aws:s3:::<bucket_name>/*",  
            "Condition": {  
                "Null": {  
                    "s3:x-amz-server-side-encryption": true  
                }  
            }  
        }  
    ]  
}
```

Enforces encryption using SSE-S3

For SSE-KMS use "aws:kms"

## Example PUT request

```
PUT /example-object HTTP/1.1  
Host: myBucket.s3.amazonaws.com  
Date: Wed, 8 Jun 2016 17:50:00 GMT  
Authorization: authorization string  
Content-Type: text/plain  
Content-Length: 11434  
x-amz-meta-author: Janet  
Expect: 100-continue  
x-amz-server-side-encryption: AES256  
[11434 bytes of object data]
```

# Enforce Encryption



# S3 Event Notifications





# S3 Event Notifications

- Sends notifications when events happen in buckets
- Destinations include:
  - Amazon Simple Notification Service (SNS) topics
  - Amazon Simple Queue Service (SQS) queues
  - AWS Lambda



# S3 Presigned URLs





# S3 Presigned URLs

AWS S3 CLI command to generate a presigned URL



```
aws s3 presign s3://dct-data-bucket/cool_image.jpeg
```



```
https://dct-data-bucket.s3.ap-southeast-2.amazonaws.com/cool_image.jpeg?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIA3KSVPHP6MAHNW5YH%2F20200909%2Fap-southeast-2%2Fs3%2Faws4_request&X-Amz-Date=20200909T053538Z&X-Amz-Expires=3600&X-Amz-SignedHeaders=host&X-Amz-Signature=8b74653beee371da07a73dfdb4ff6883742383afa528aec5c95c326c97764db
```

This is the response; the URL expires after 1 hour

# Multipart Upload & Transfer Acceleration





# S3 Multipart Upload

---

- Multipart upload uploads objects in parts independently, in parallel and in any order
- Performed using the S3 Multipart upload API
- It is recommended for objects of 100 MB or larger
- Can be used for objects from 5 MB up to 5 TB
- Must be used for objects larger than 5 GB



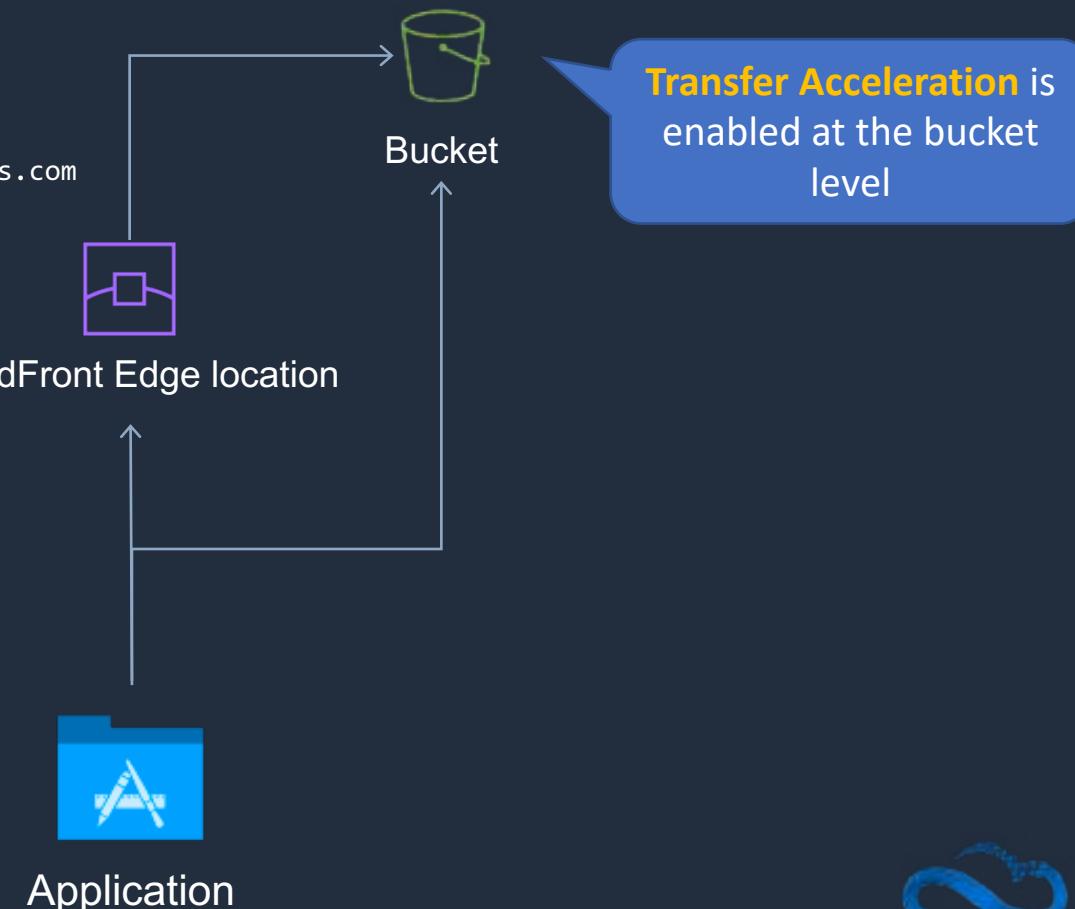
# S3 Transfer Acceleration

---

- Uses CloudFront edge locations to improve performance of transfers from client to S3 bucket

`http://bucketname.s3-accelerate.amazonaws.com`  
`http://bucketname.s3-accelerate.dualstack.amazonaws.com`

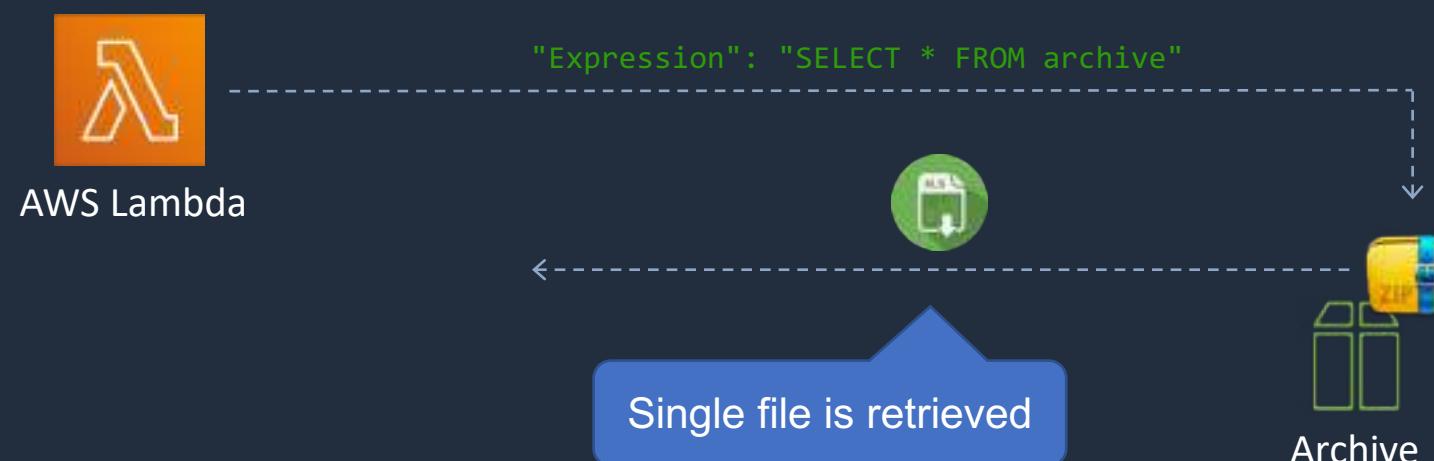
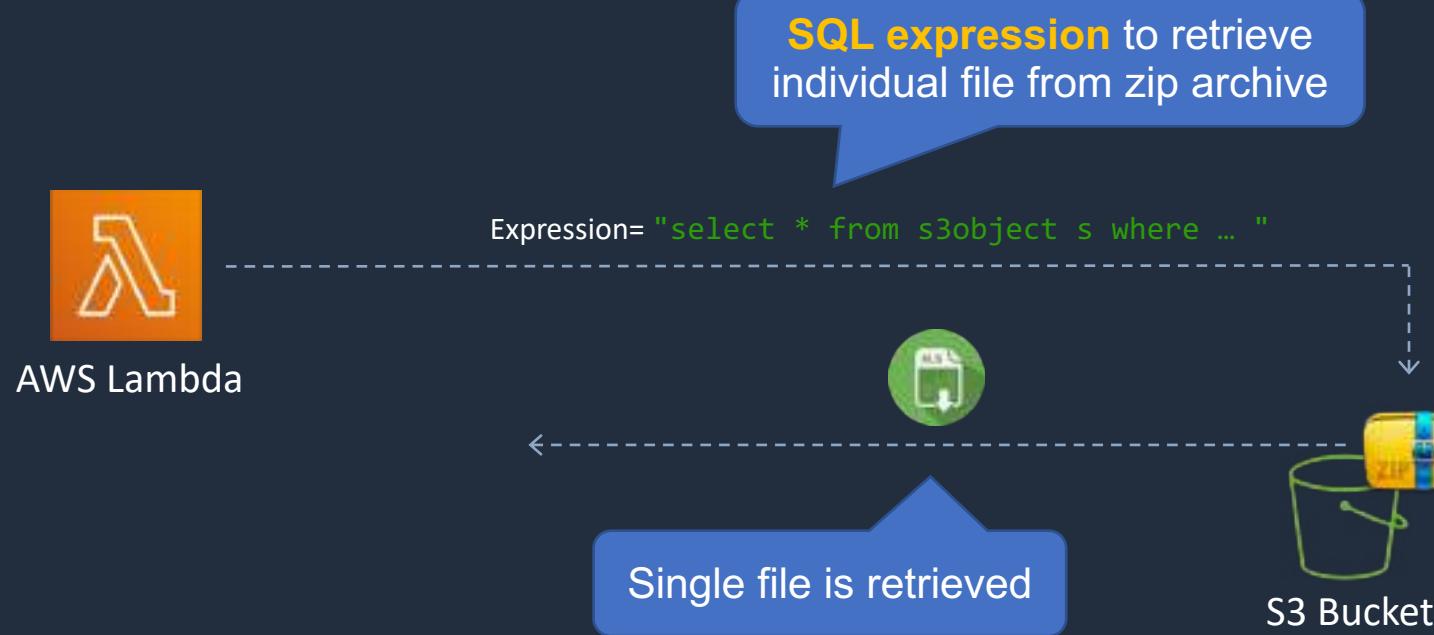
AWS only charges if  
there's a performance  
improvement



# S3 Select and Glacier Select



# S3 Select and Glacier Select



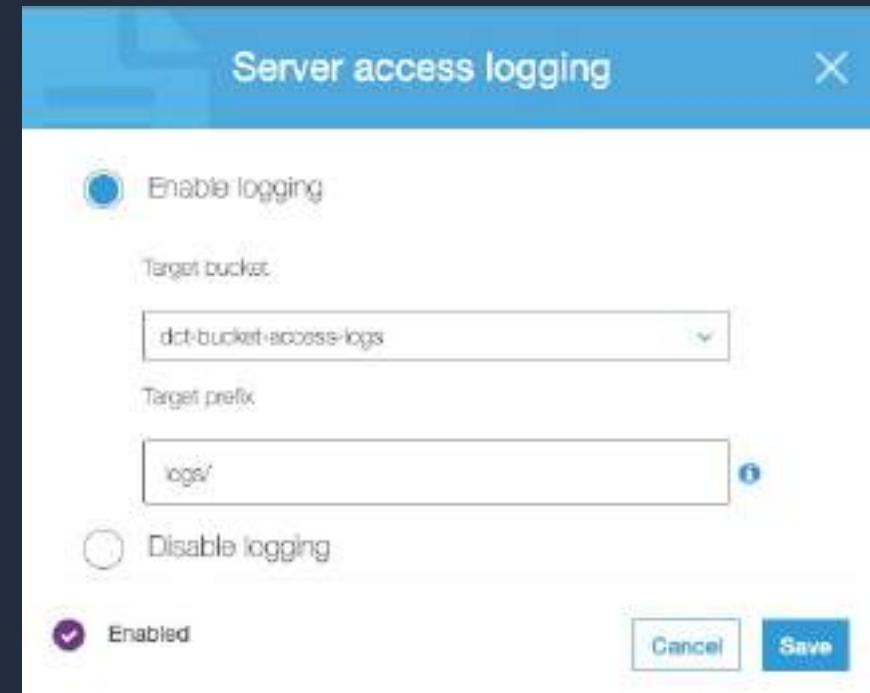
# Server Access Logging





# Server Access Logging

- Provides detailed records for the requests that are made to a bucket
- Details include the requester, bucket name, request time, request action, response status, and error code (if applicable)
- Disabled by default
- Only pay for the storage space used
- Must configure a separate bucket as the destination (can specify a prefix)
- Must grant write permissions to the Amazon S3 Log Delivery group on destination bucket



# S3 Static Website



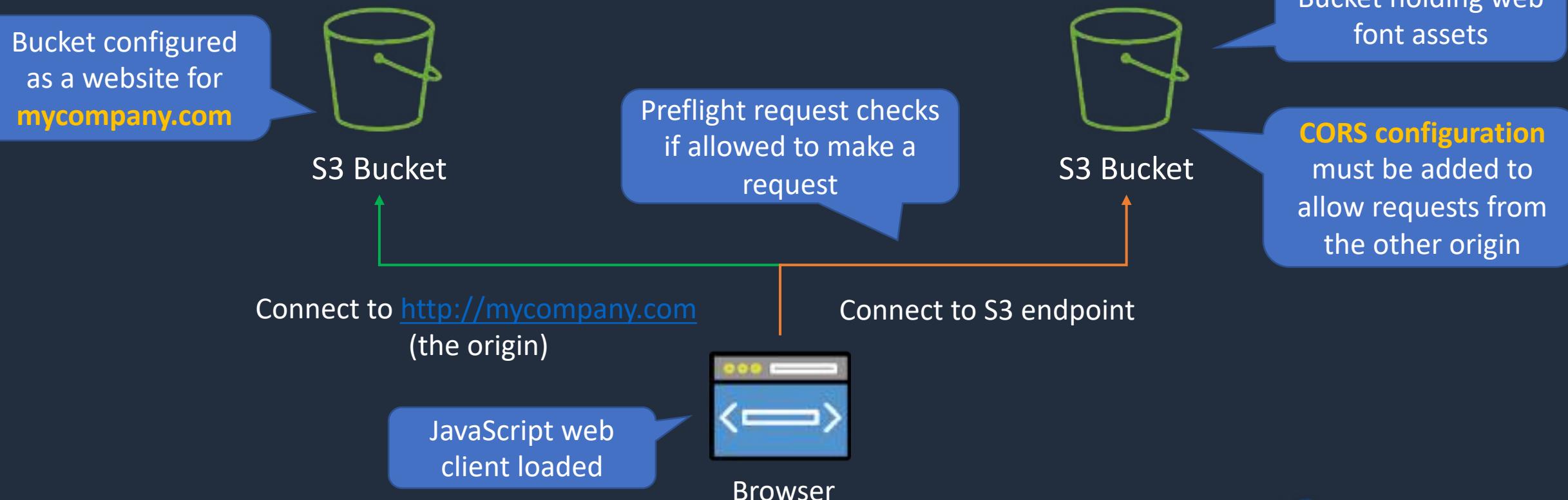
# Cross-Origin Resource Sharing (CORS)





# CORS with Amazon S3

- Allows requests from an origin to another origin
- Origin is defined by DNS name, protocol, and port





# CORS with Amazon S3

---

- Enabled through setting:
  - Access-Control-Allow-Origin
  - Access-Control-Allow-Methods
  - Access-Control-Allow-Headers
- These settings are defined using rules
- Rules are added using JSON files in S3



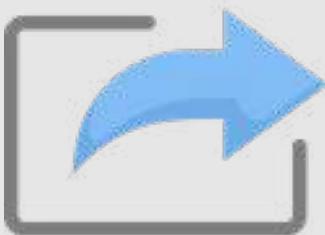
# Example CORS Rule

---

---

```
[  
  {  
    "AllowedHeaders": [  
      "*"  
    ],  
    "AllowedMethods": [  
      "PUT",  
      "POST",  
      "DELETE"  
    ],  
    "AllowedOrigins": [  
      "http://www.mycompany.com"  
    ],  
    "ExposeHeaders": []  
  }  
]
```

# Cross Account Access





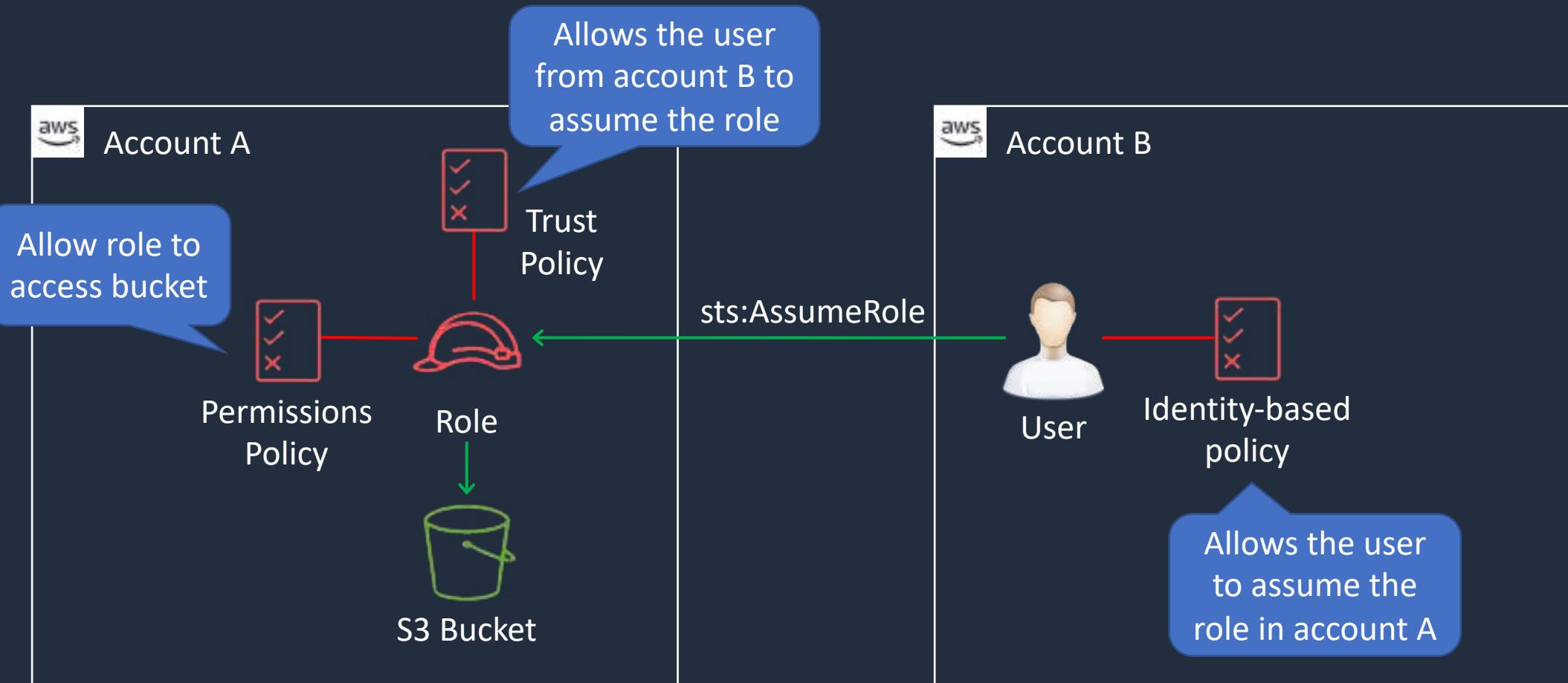
# Cross Account Access Methods

---

- Resource-based policies and IAM policies for **programmatic-only** access to S3 bucket objects
- Resource-based ACL and IAM policies for **programmatic-only** access to S3 bucket objects
- Cross-account IAM roles for **programmatic and console access** to S3 bucket objects



# Cross Account Access

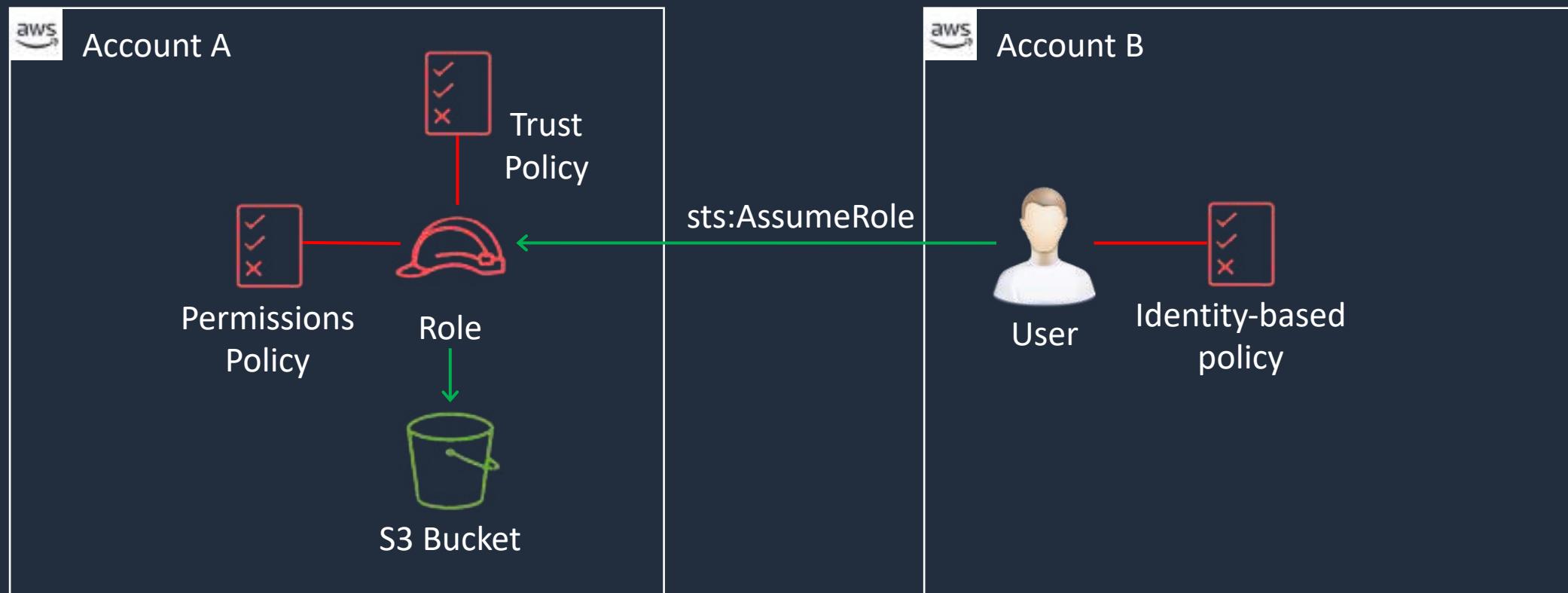


# Configure Cross Account Access





# Cross Account Access



# Architecture Patterns – Amazon S3





# Architecture Patterns – Amazon S3

## Requirement

Company is concerned about accidental deletion of Amazon S3 objects

Data stored in S3 is frequently accessed for 30 days then is rarely accessed but must be immediately retrievable

A backup of S3 objects within a specific folder in a bucket must be replicated to another Region

## Solution

Enable S3 versioning

Use a lifecycle policy to transition objects from S3 standard to S3 Standard-IA after 30 days

Configure cross-region replication and specify the folder name as a prefix



# Architecture Patterns – Amazon S3

## Requirement

Previous versions of objects in a versioning-enabled S3 bucket must be stored long term at the lowest cost

A company wishes to manage all encryption of S3 objects through their application with their own encryption keys

Unencrypted objects in an Amazon S3 bucket must be encrypted

## Solution

Create a lifecycle rule that transitions previous versions to S3 Glacier Deep Archive

Use client-side encryption with client managed keys

Re-upload the objects and specify the encryption an encryption key



## Requirement

An administrator requires a notification when objects are deleted from an Amazon S3 bucket

## Solution

Configure an event notification that uses the SNS service

A group of customers without AWS credentials must be granted time-limited access to a software update that is stored in an Amazon S3 bucket

Generate a presigned URL

Solutions architects require both programmatic and console access across AWS accounts

Configure cross-account access using IAM roles

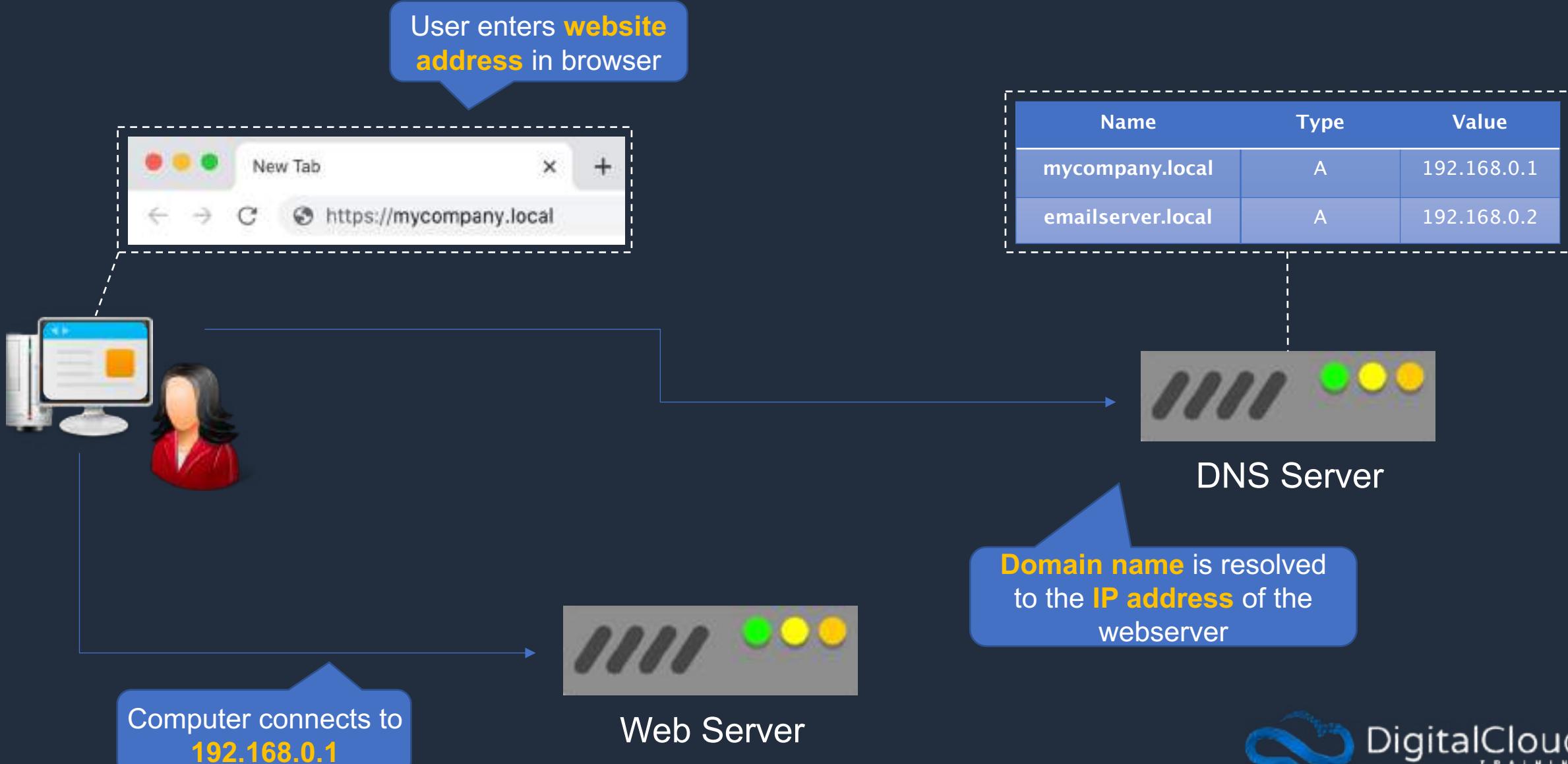
# SECTION 8

## DNS, Caching, and Performance Optimization

# DNS and Amazon Route 53

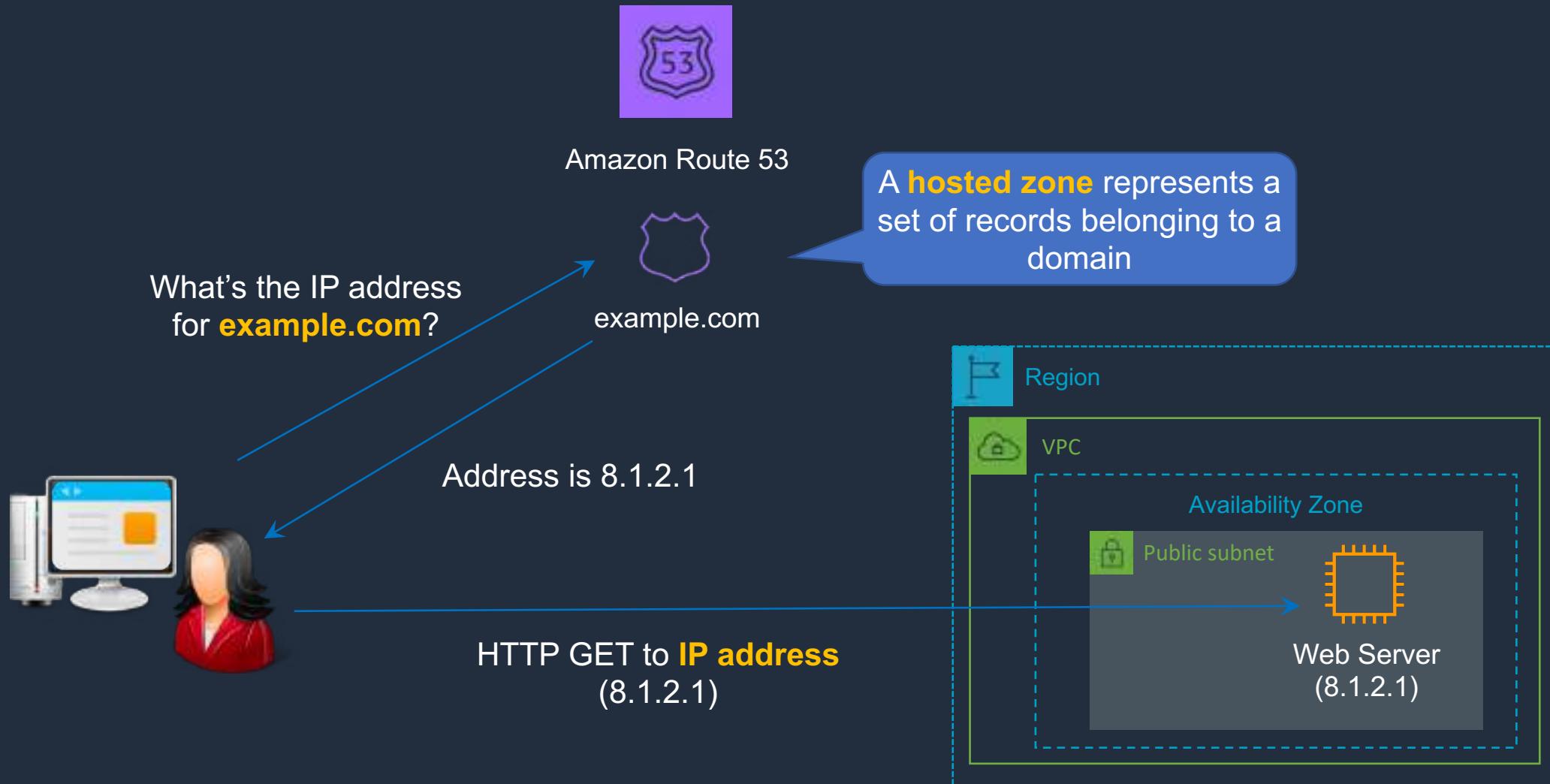


# The Domain Name System (DNS)





# Amazon Route 53





# Amazon Route 53 Routing Policies

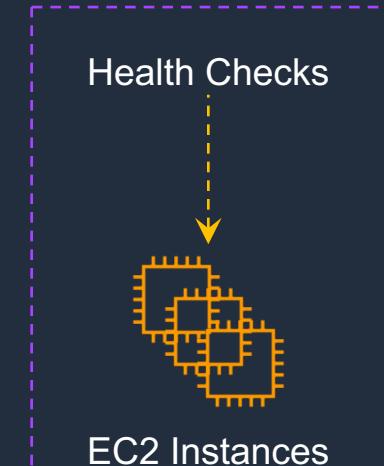
Routing Policy	What it does
<b>Simple</b>	Simple DNS response providing the IP address associated with a name
<b>Failover</b>	If primary is down (based on health checks), routes to secondary destination
<b>Geolocation</b>	Uses geographic location you're in (e.g. Europe) to route you to the closest region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



# Amazon Route Features



Amazon Route 53



# Register Domain with Route 53 (Optional)



# Amazon Route 53 Hosted Zones





# Amazon Route 53 Hosted Zones

Name	Type	Value	TTL
example.com	A	8.1.2.1	60
dev.example.com	A	8.1.2.2	60



Amazon Route 53

This is an example of a  
**public hosted zone**

What's the address for  
example.com?

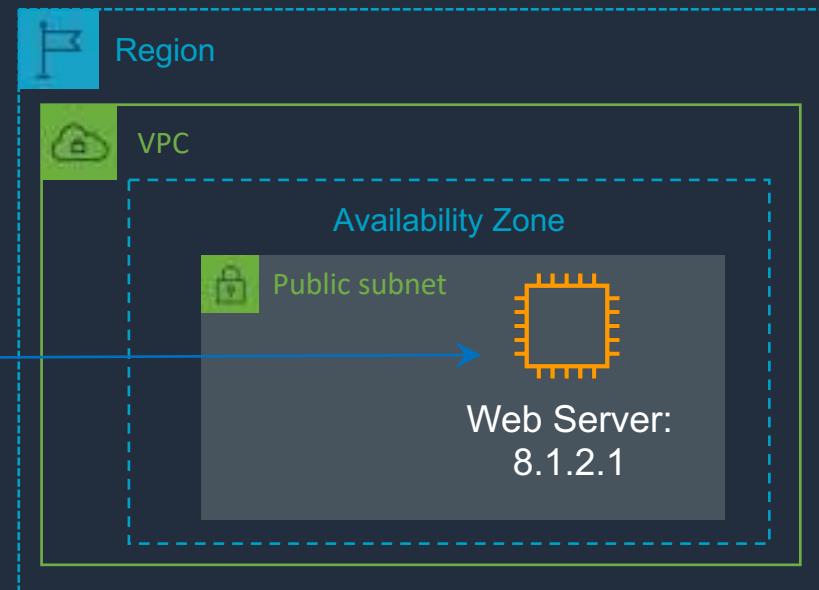
example.com

Address is 8.1.2.1



HTTP GET to 8.1.2.1

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





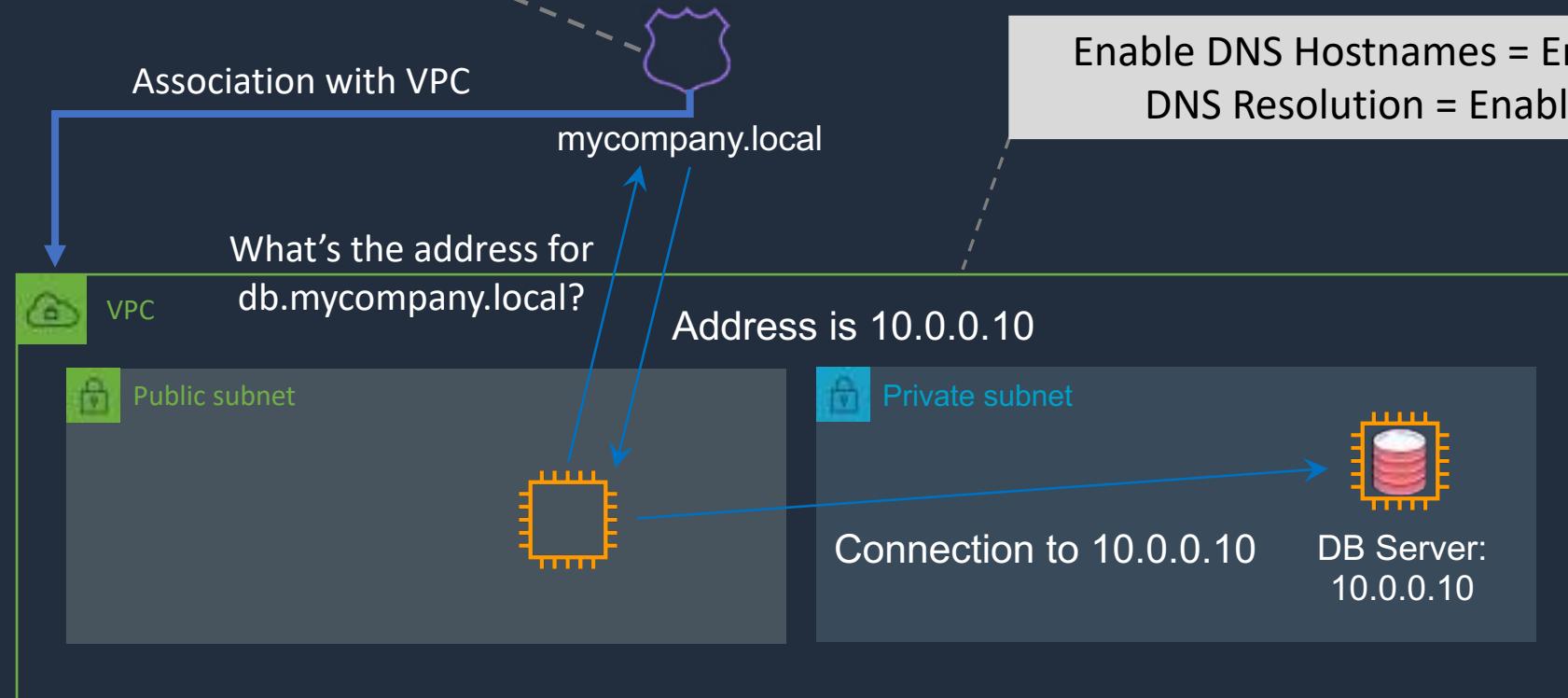
# Amazon Route 53 Hosted Zones

Name	Type	Value	TTL
db.mycompany.local	A	10.0.0.10	60
app.mycompany.local	A	10.0.0.11	60



Amazon Route 53

This is an example of a  
**private hosted zone**





# Migration to/from Route 53

---

- You can migrate from **another DNS provider** and can import records
- You can migrate a hosted zone to **another AWS account**
- You can migrate from Route 53 to **another registrar**
- You can also associate a Route 53 hosted zone with a **VPC in another account**
  - Authorize association with VPC in the second account.
  - Create an association in the second account

# Route 53 Routing Policies





# Amazon Route 53 Routing Policies

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



# Amazon Route 53 – Simple Routing Policy

Name	Type	Value	TTL
simple.dctlabs.com	A	1.1.1.1	60
		2.2.2.2	
simple2.dctlabs.com	A	3.3.3.3	60



Amazon Route 53





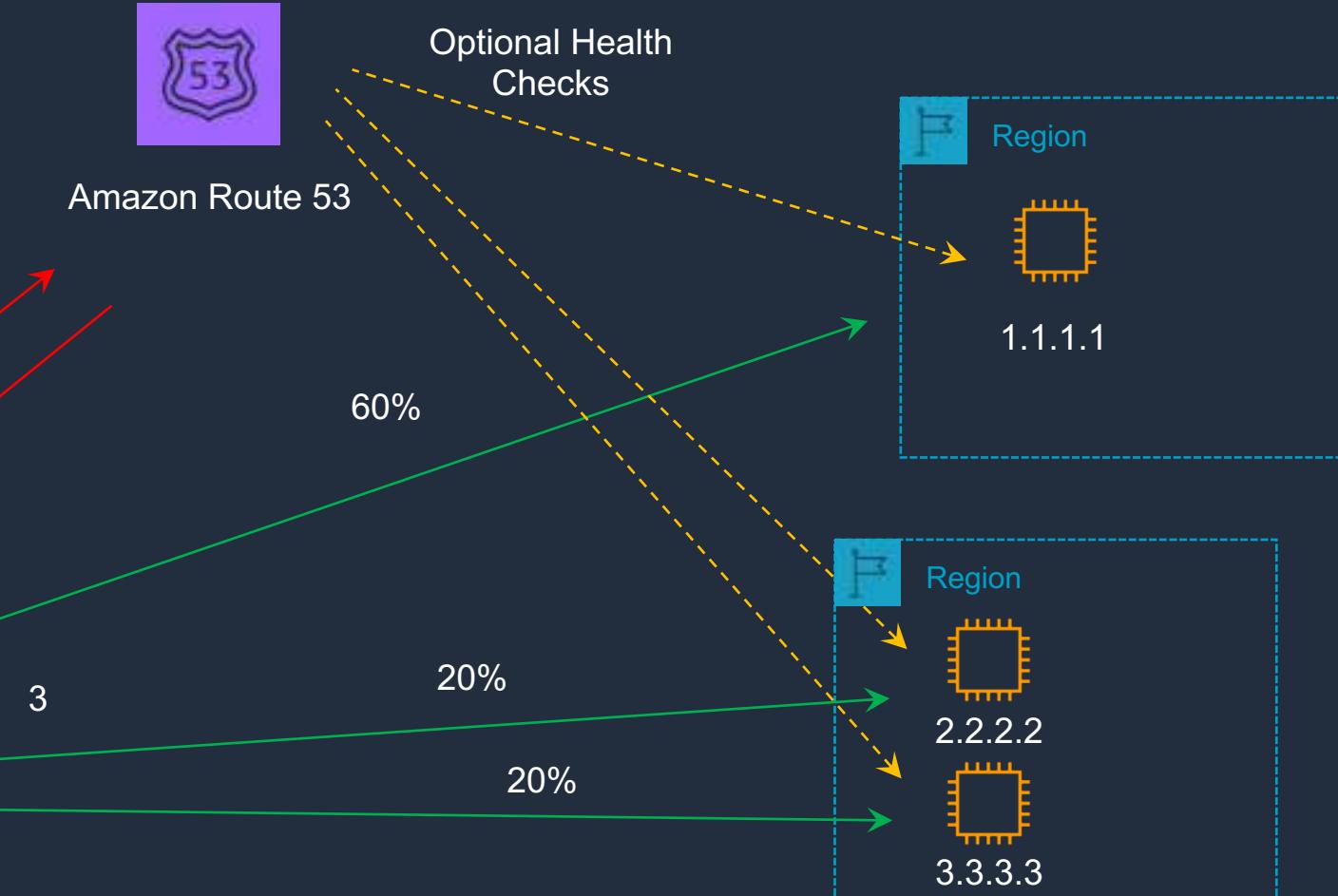
# Amazon Route 53 – Weighted Routing Policy

Name	Type	Value	Health	Weight
weighted.dctlabs.com	A	1.1.1.1	ID	60
weighted.dctlabs.com	A	2.2.2.2	ID	20
weighted.dctlabs.com	A	3.3.3.3	ID	20

Simplified values - actually uses an integer between 0 and 255

?

DNS query





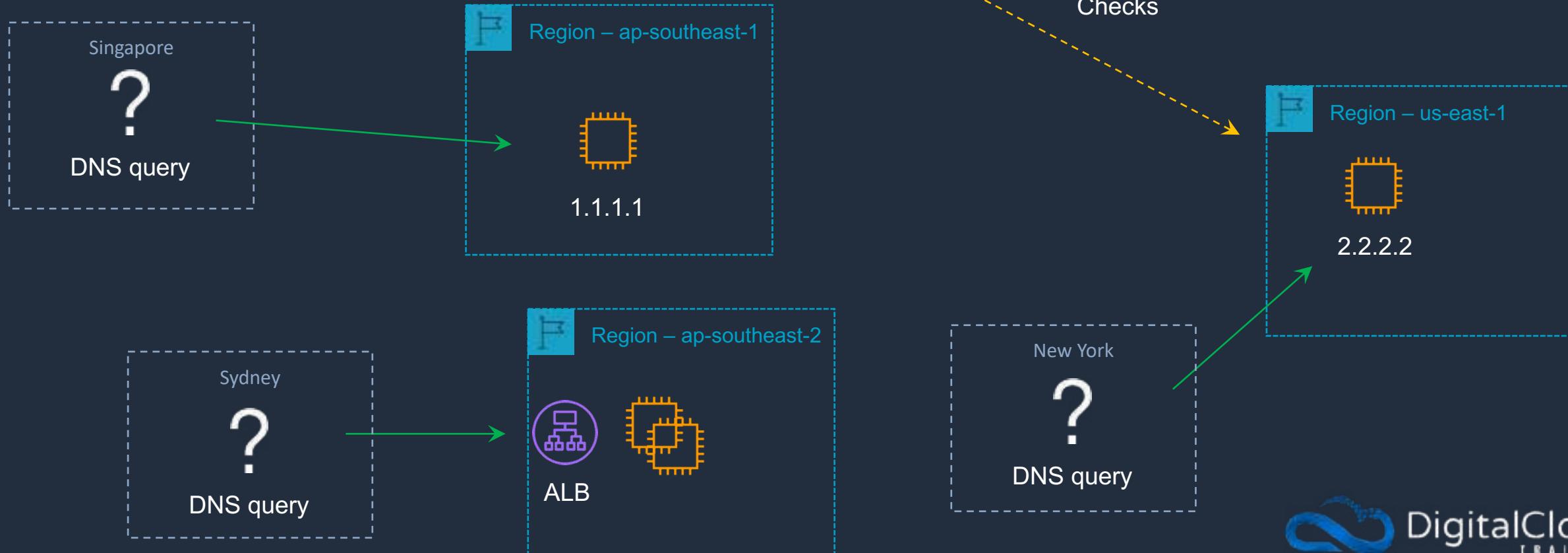
# Amazon Route 53 – Latency Routing Policy

Name	Type	Value	Health	Region
latency.dctlabs.com	A	1.1.1.1	ID	ap-southeast-1
latency.dctlabs.com	A	2.2.2.2	ID	us-east-1
latency.dctlabs.com	A	alb-id	ID	ap-southeast-2



Amazon Route 53

Optional Health Checks





# Amazon Route 53 – Failover Routing Policy

Name	Type	Value	Health	Record Type
failover.dctlabs.com	A	1.1.1.1	ID	Primary
failover.dctlabs.com	A	alb-id		Secondary



Amazon Route 53

Health check is  
**required** on Primary



DNS query

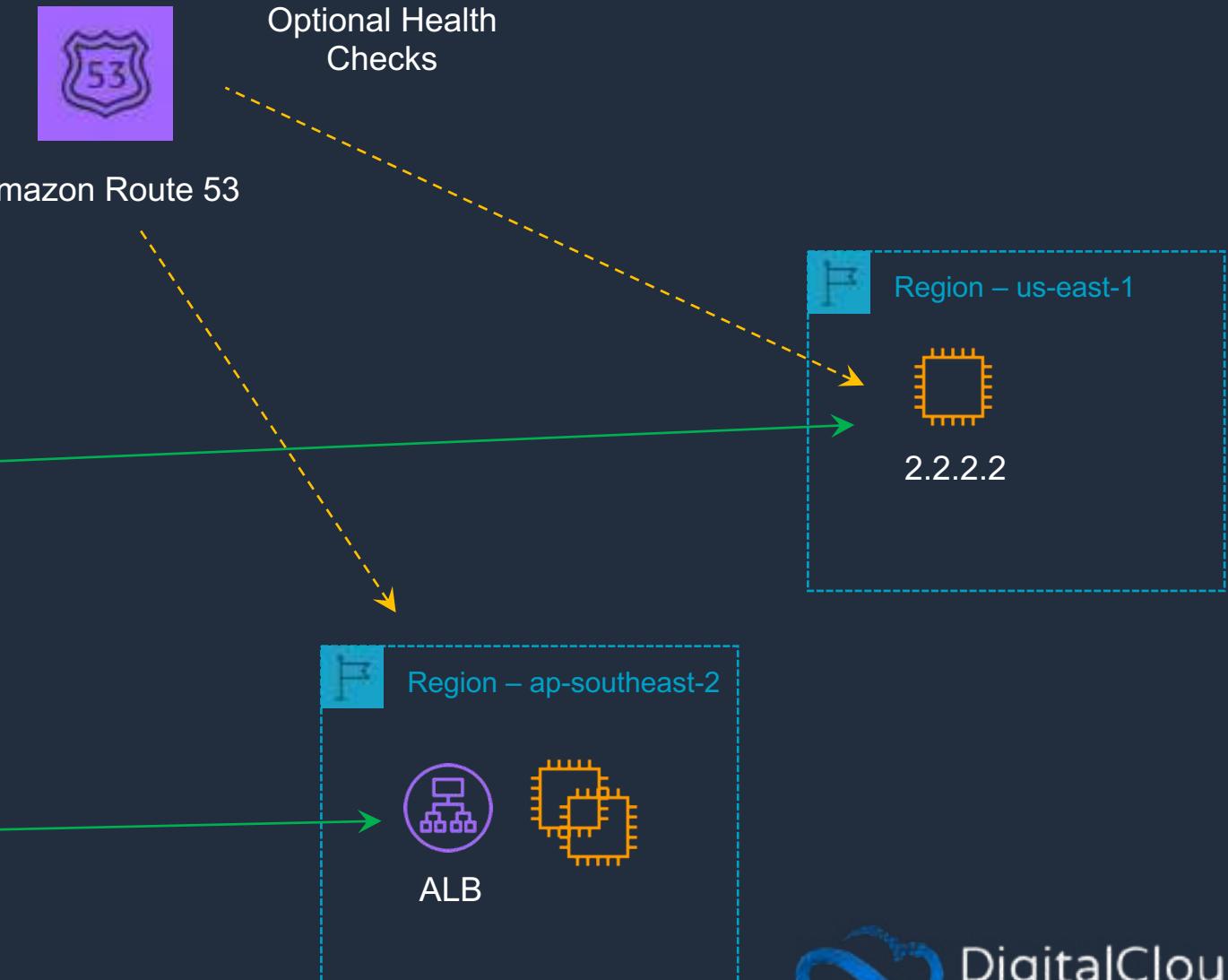


ap-southeast-2 is the  
**secondary** Region



# Amazon Route 53 – Geolocation Routing Policy

Name	Type	Value	Health	Geolocation
geolocation.dctlabs.com	A	1.1.1.1	ID	Singapore
geolocation.dctlabs.com	A	2.2.2.2	ID	Default
geolocation.dctlabs.com	A	alb-id	ID	Oceania





# Amazon Route 53 – Multivalue Routing Policy

Name	Type	Value	Health	Multi Value
multivalue.dctlabs.com	A	1.1.1.1	ID	Yes
multivalue.dctlabs.com	A	2.2.2.2	ID	Yes
multivalue.dctlabs.com	A	3.3.3.3	ID	Yes



Amazon Route 53

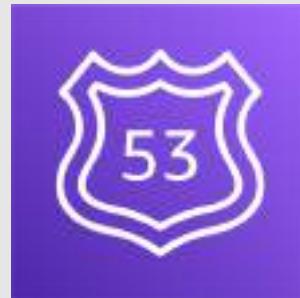
Health Checks:  
returns healthy  
records only



# Using Route 53 Routing Policies

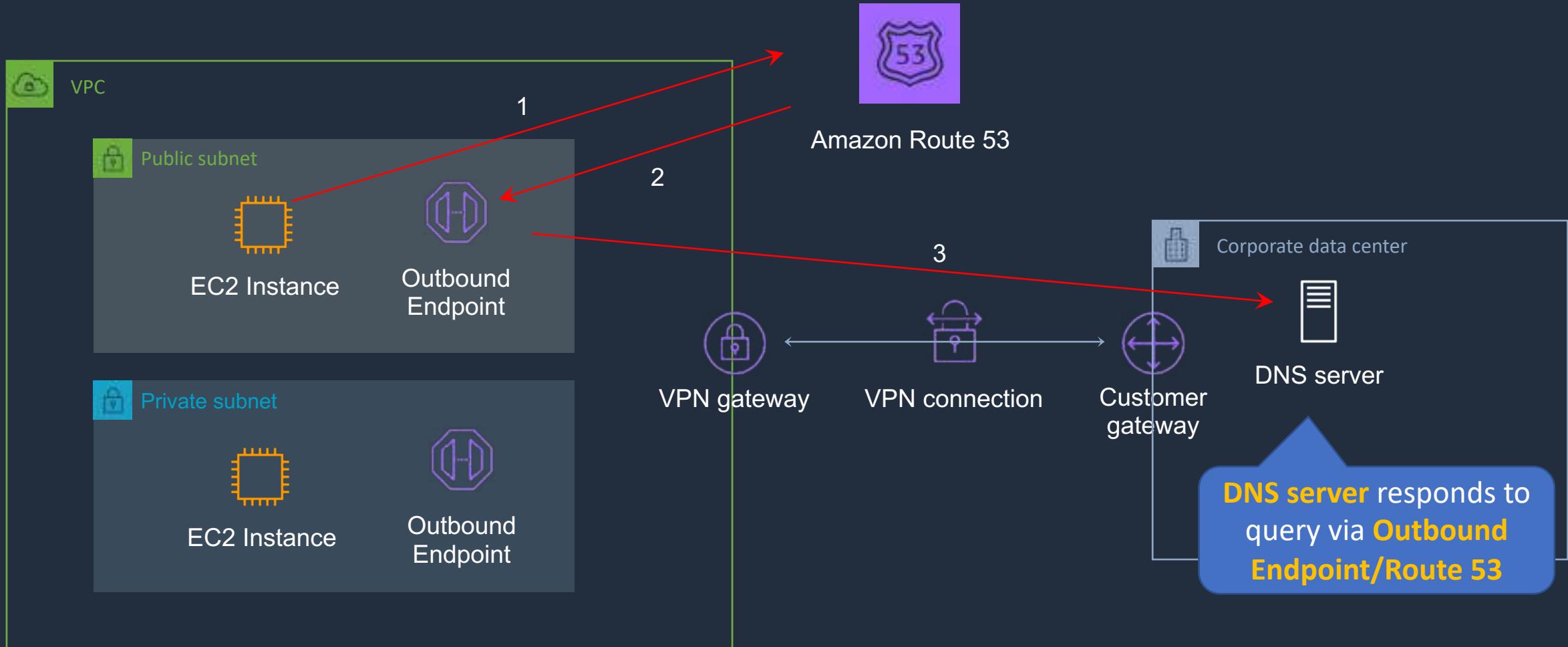


# Amazon Route 53 Resolver



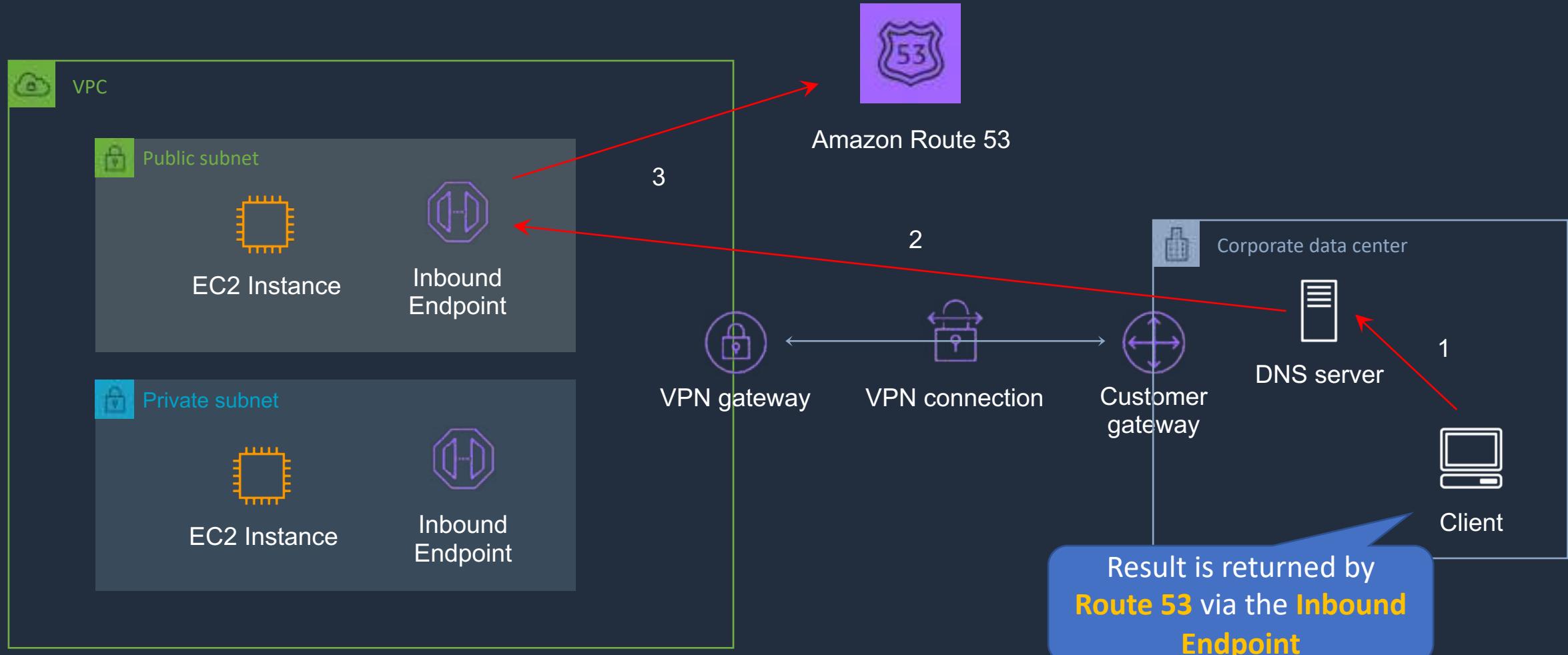


# Route 53 Resolver – Outbound Endpoints





# Route 53 Resolver – Inbound Endpoints

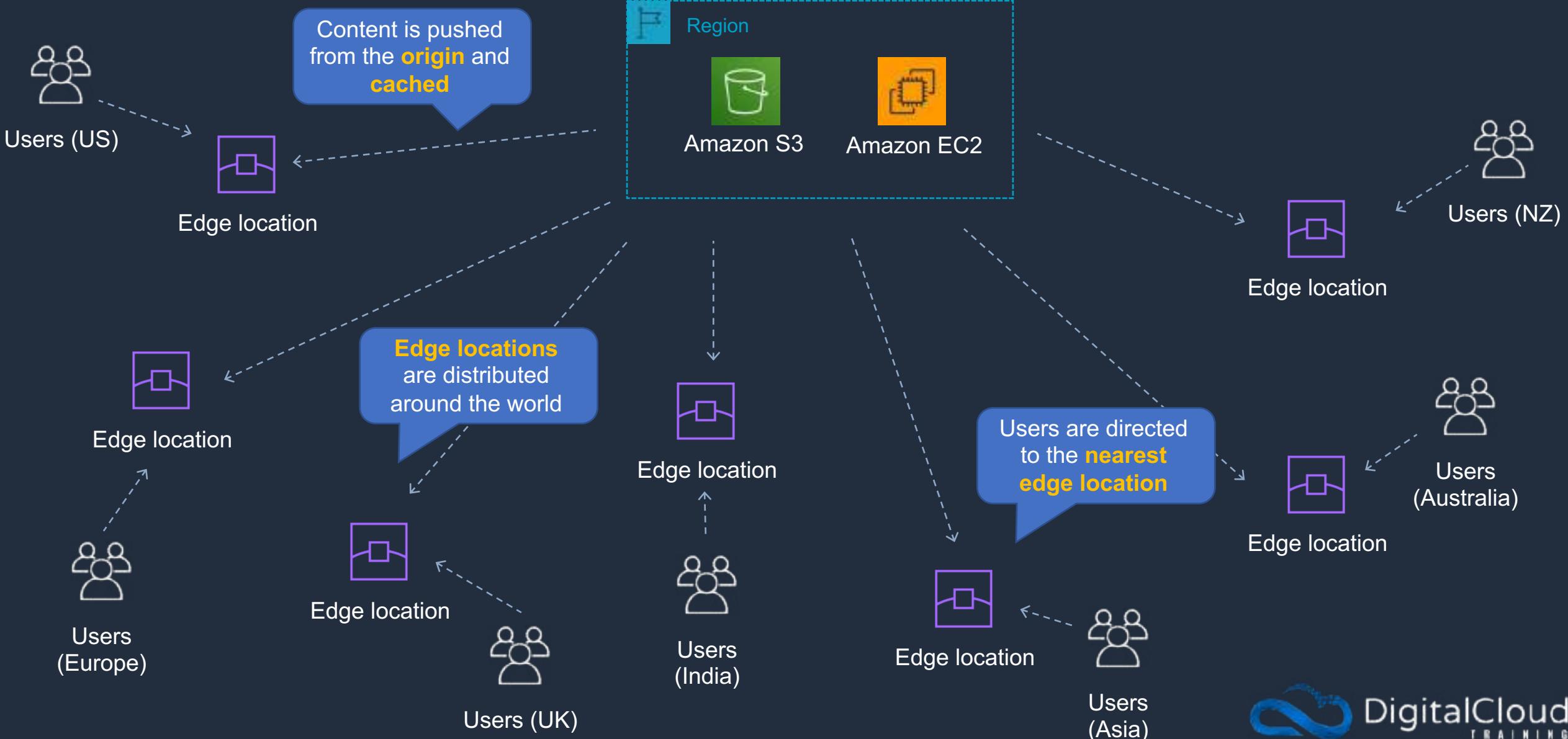


# Amazon CloudFront Origins and Distributions





# Amazon CloudFront



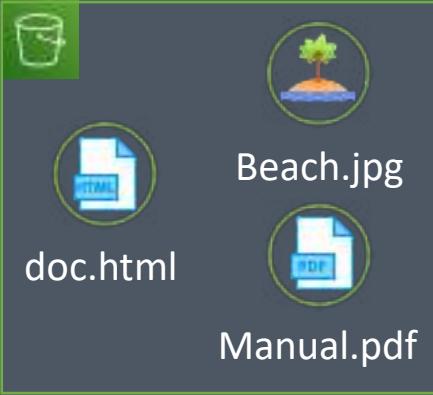


# CloudFront Origins and Distributions

## CloudFront Distribution

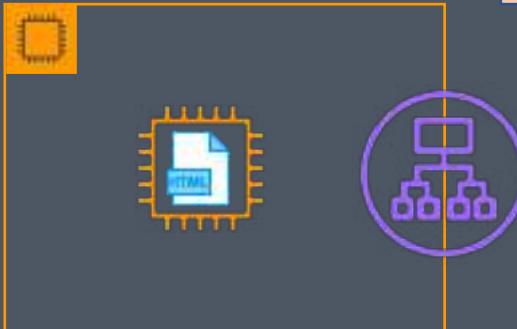
Name: **d1schtd9zdwr1.cloudfront.net**

### S3 Origin



**S3 static websites**  
can also be origins

### Custom Origin



**RTMP distributions** were discontinued  
so only **web distributions** are currently  
available

### CloudFront Web Distribution:

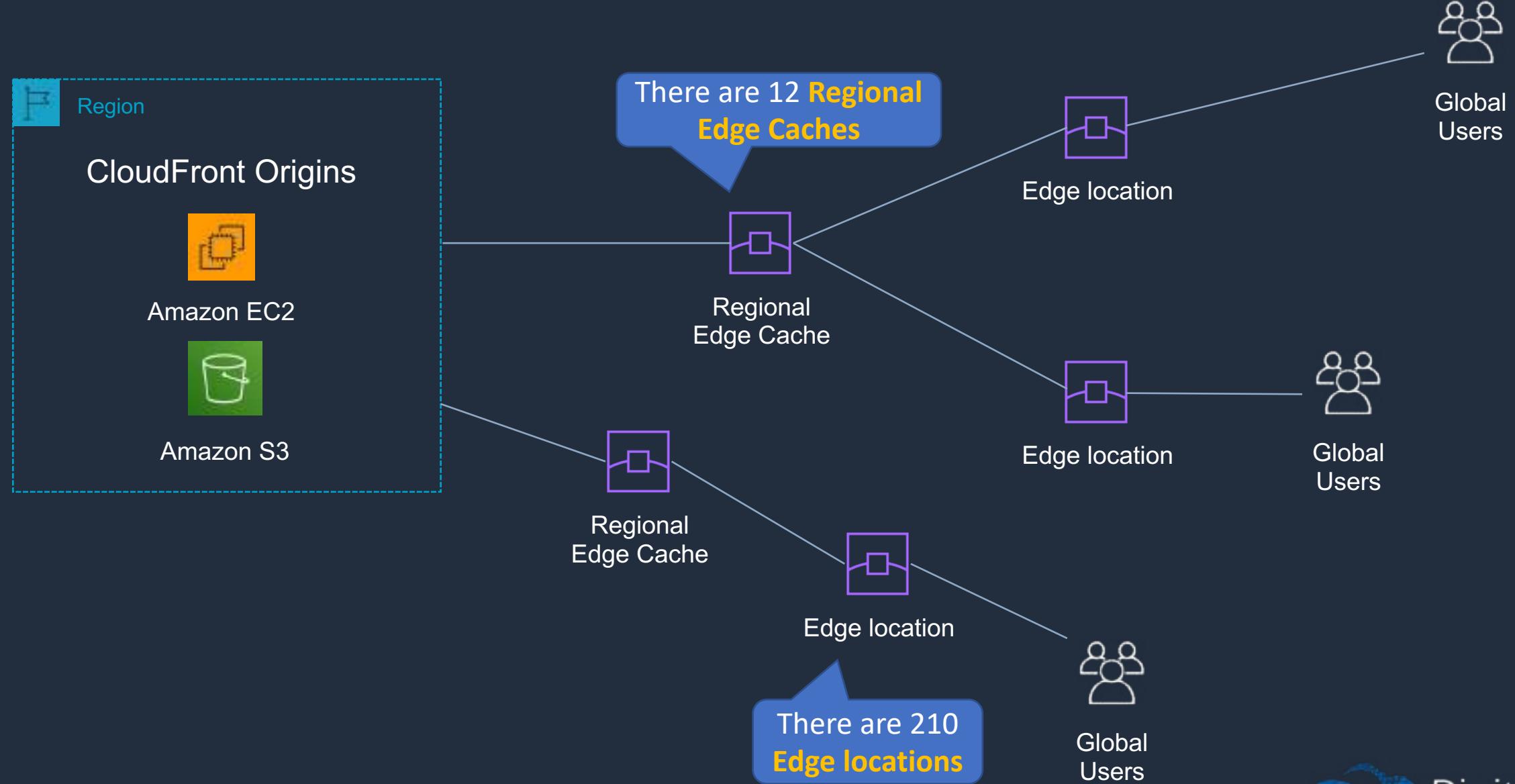
- Speed up distribution of static and dynamic content, for example, .html, .css, .php, and graphics files
- Distribute media files using HTTP or HTTPS
- Add, update, or delete objects, and submit data from web forms
- Use live streaming to stream an event in real time

# Amazon CloudFront Caching and Behaviors



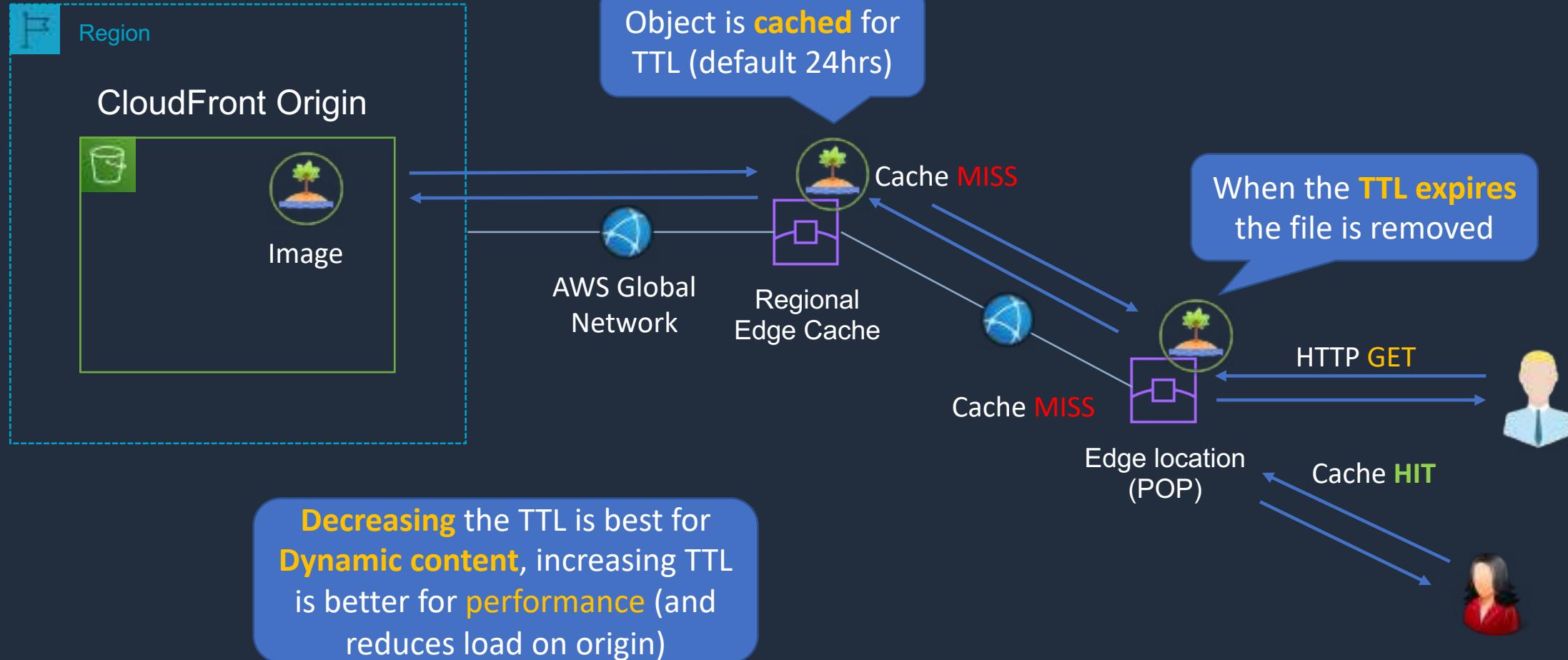


# Amazon CloudFront Caching





# Amazon CloudFront Caching





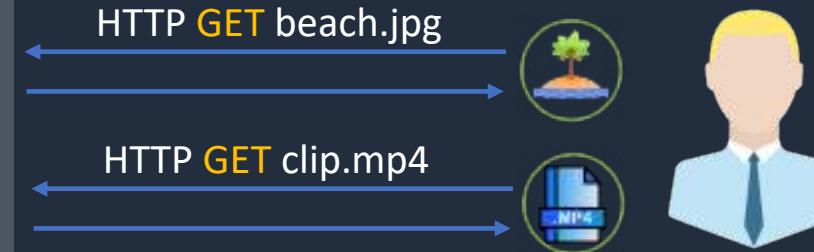
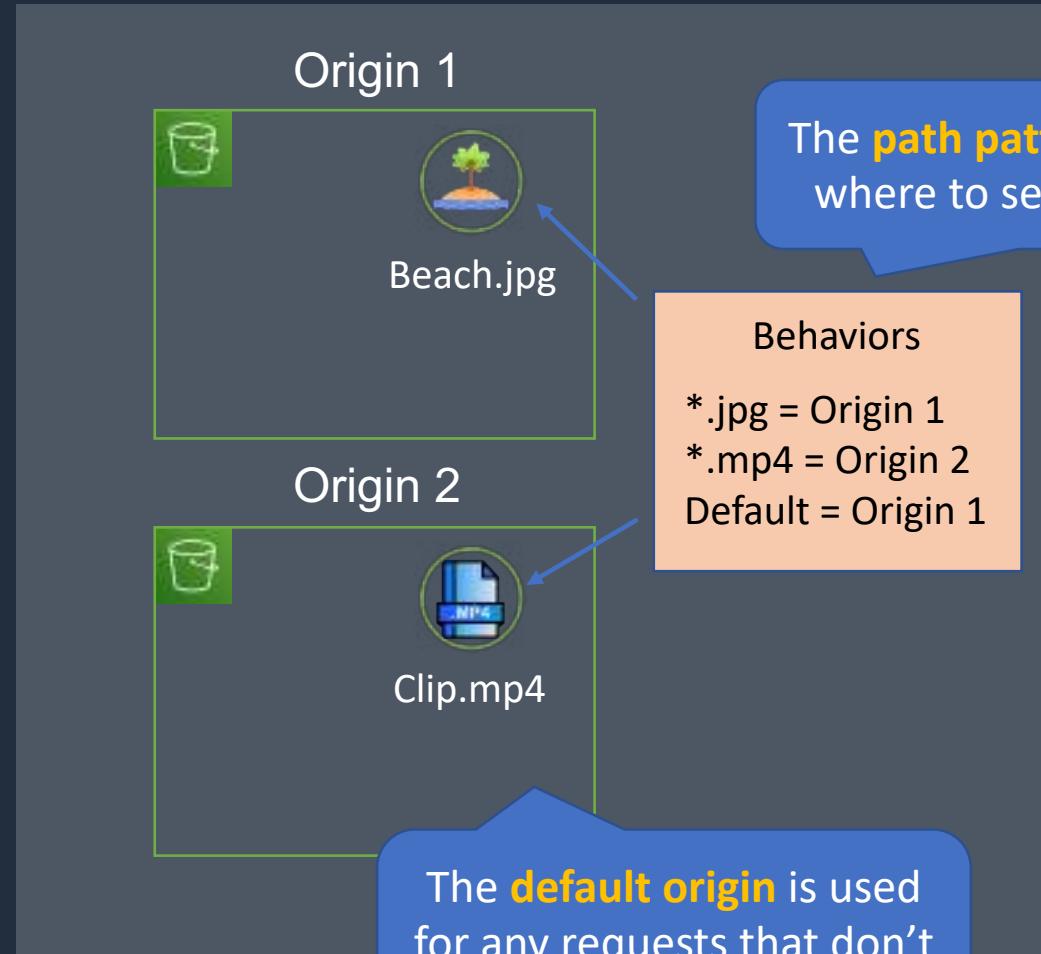
# Amazon CloudFront Caching

---

- You can define a maximum **Time To Live** (TTL) and a default TTL
- TTL is defined at the **behavior** level
- This can be used to define different TTLs for different file types (e.g. png vs jpg)
- After expiration, CloudFront checks the origin for any new requests (check the file is the latest version)
- Headers can be used to control the cache:
  - **Cache-Control max-age=(seconds)** - specify how long before CloudFront gets the object again from the origin server
  - **Expires** – specify an expiration date and time

# CloudFront Path Patterns

## CloudFront Distribution



Precedence	Path Pattern	Origin or Origin Group	Viewer Protocol Policy	Cache Policy Name
0	*.jpg	Origin 1	HTTP and HTTPS	Managed-CachingOptimized
1	*.mp4	Origin 2	HTTP and HTTPS	Managed-CachingOptimized
2	Default (*)	Origin 1	HTTP and HTTPS	Managed-CachingOptimized



# Caching Based on Request Headers

---

- You can configure CloudFront to forward **headers** in the **viewer request** to the origin
- CloudFront can then cache multiple versions of an object based on the values in one or more request headers
- Controlled in a behavior to do one of the following:
  - Forward all headers to your origin (objects are **not cached**)
  - Forward a whitelist of headers that you specify
  - Forward only the default headers (doesn't cache objects based on values in request headers)

# CloudFront Signed URLs and OAI

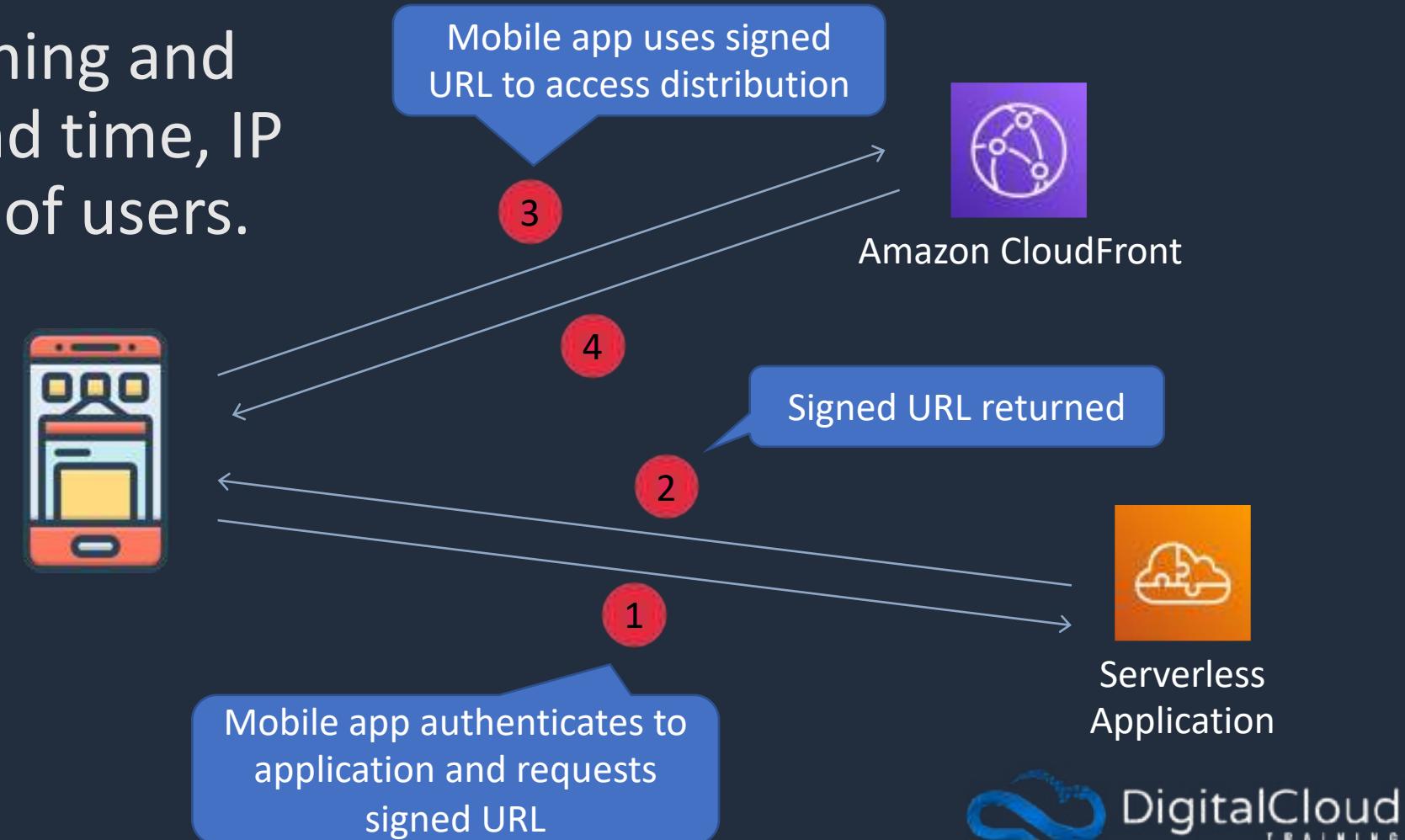




# CloudFront Signed URLs

- Signed URLs provide more control over access to content.
- Can specify beginning and expiration date and time, IP addresses/ranges of users.

Signed URLs should be used for **individual files** and clients that don't support **cookies**.





# CloudFront Signed Cookies

---

- Similar to Signed URLs
- Use signed cookies when you don't want to change URLs
- Can also be used when you want to provide access to **multiple restricted files** (Signed URLs are for individual files)



# CloudFront Origin Access Identity (OAI)



# Cache and Behavior Settings



# CloudFront SSL/TLS and SNI





# CloudFront SSL/TLS



## Viewer Protocol

Certificate can be **ACM** or a trusted **third-party CA**



SSL

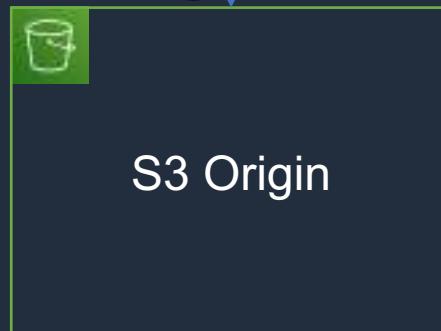
For CloudFront certificate must be issued in **us-east-1**



AWS Certificate Manager

Default CF **domain name** can be changed using **CNAMES**

S3 has its **own** certificate (can't be changed)



S3 Origin

SSL

## Origin Protocol

Origin certificates must be **public certificates**



Custom Origin

Certificate can be **ACM** (ALB) or **third-party** (EC2)



# CloudFront Server Name Indication (SNI)



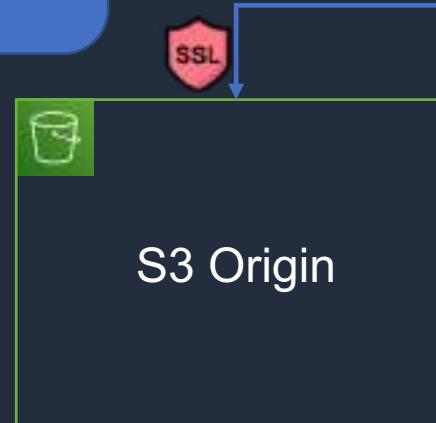
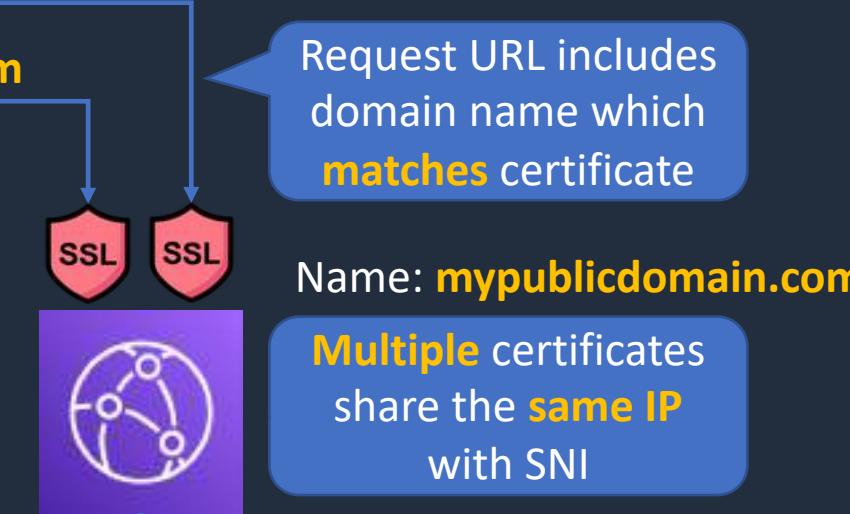
HTTP GET: <https://mypublicdomain.com>



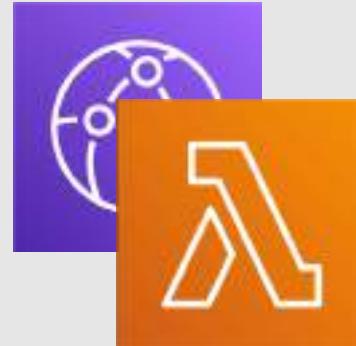
HTTP GET: <https://myotherdomain.com>

Note: SNI works with  
browsers/clients released  
**after 2010** – otherwise  
need **dedicated IP**

Name: [myotherdomain.com](https://myotherdomain.com)



# Lambda@Edge





# Lambda@Edge

- Run Node.js and Python Lambda functions to customize the content CloudFront delivers
- Executes functions closer to the viewer
- Can be run at the following points
  - After CloudFront receives a request from a viewer (viewer request)
  - Before CloudFront forwards the request to the origin (origin request)
  - After CloudFront receives the response from the origin (origin response)
  - Before CloudFront forwards the response to the viewer (viewer response)



# AWS Global Accelerator



# AWS Global Accelerator



Resolve dctlabs.com  
Users in US



Answer:  
51.45.2.12  
53.58.31.89



Edge location



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

Addresses:  
51.45.2.12  
53.58.31.89

Static **anycast**  
IP addresses

Amazon Route 53

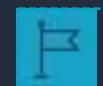
Connect via Edge Location

Requests are  
routed to the  
**optimal endpoint**

AWS Global Network

Traffic  
traverses the  
**AWS global  
network**

Users are  
**redirected** to  
another **endpoint**



us-east-1



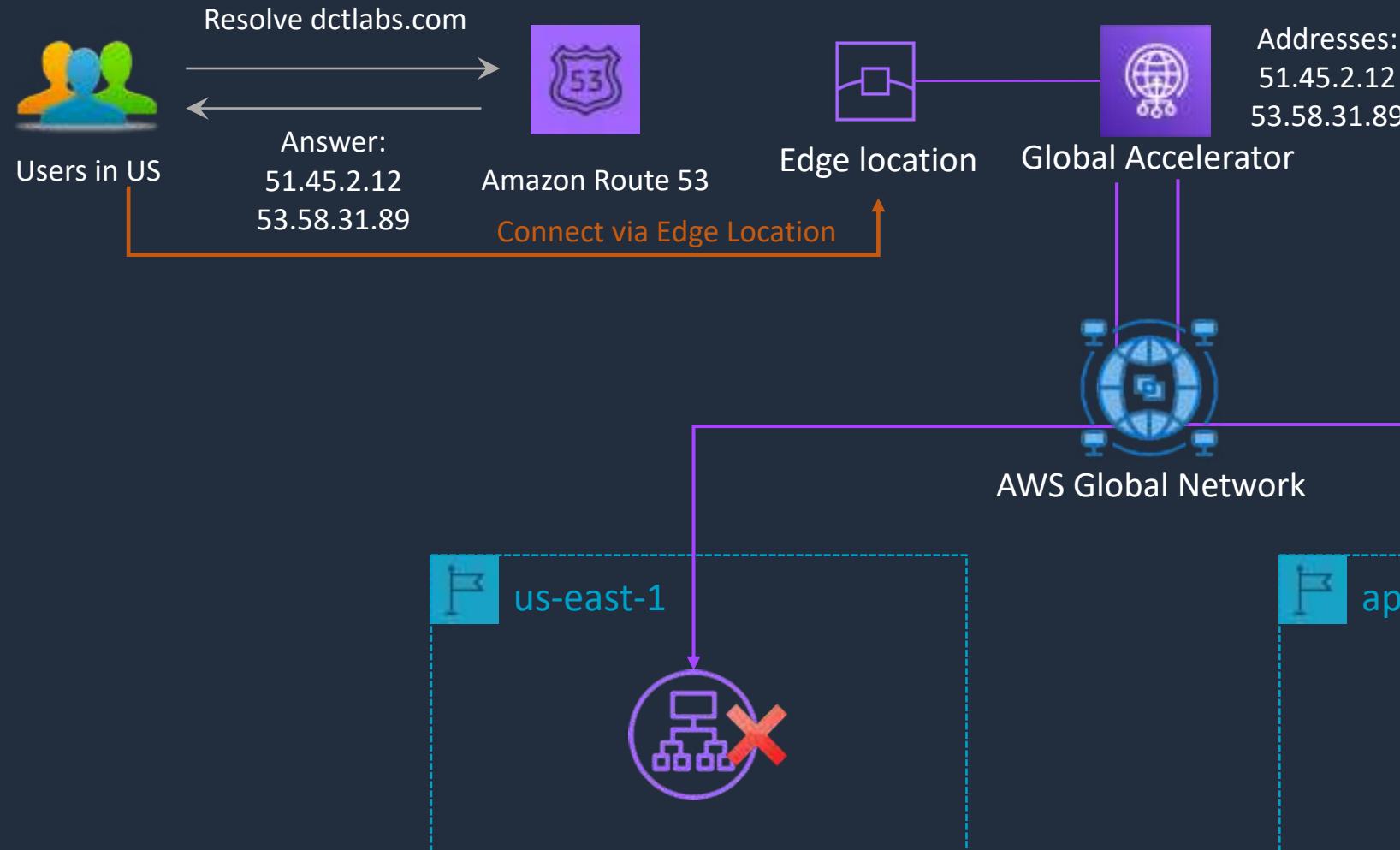
ap-southeast-2

# Create a Global Accelerator





# AWS Global Accelerator



# Architecture Patterns – DNS, Caching and Performance Optimization





## Requirement

An Elastic Load Balancer must be resolvable using a company's public domain name. A Route 53 hosted zone exists

A website runs across two AWS Regions. All traffic goes to one Region and should be redirected only if the website is unavailable

Websites run in several countries and distribution rights require restricting access to content based on the geographic source of the connection

## Solution

Create an Alias record that maps the domain name to the ELB

Create a failover routing policy in AWS Route 53 and configure health checks on the primary

Use AWS Route 53 geolocation routing and restrict distribution based on geographic location



## Requirement

A CloudFront distribution has multiple S3 origins. Requests should be served from different origins based on file type being requested

Content is accessed using an application and CloudFront distribution. Need to control access to multiple files on the distribution

Application runs behind an Application Load Balancer in multiple Regions. Need to intelligently route traffic based on latency and availability

## Solution

Modify the CloudFront behavior and configure a path pattern

Configure signed cookies and update the application

Create an AWS Global Accelerator and add the ALBs

# SECTION 9

## Block and File Storage

# Block vs File vs Object Storage





# Hard Drives

Hard drives are **block-based** storage systems

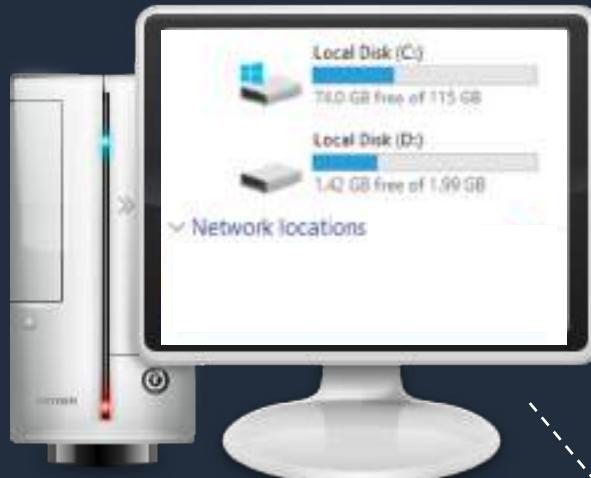


The Operating System (OS) can be used to create **volumes**. A volume can be partitioned and formatted

Hard drives are block-based storage systems



# Network Attached Storage



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

The NAS “shares” **filesystems** over the network



Network Switch



Network Attached Storage Server (NAS)



NAS devices are file-based storage systems



# 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, File, and Object Storage

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

Block Storage

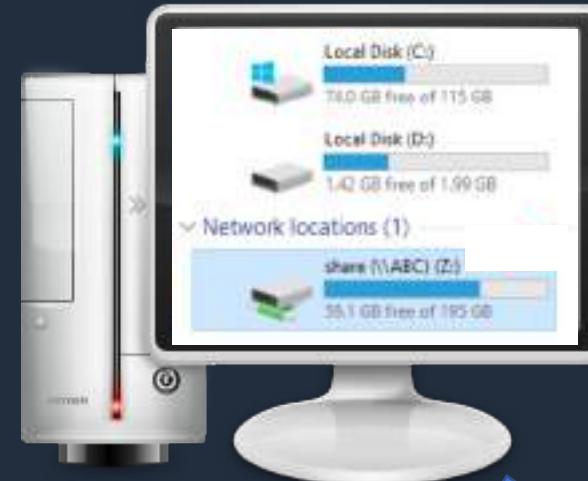


Disk Management

Volume  
1000 GB

A **filesystem** can be shared by many users/computers

File Storage



Massively scalable, low cost

Object Storage



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

Uses a **REST API**

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

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



# AWS Storage Services

---

## Block Storage



Amazon Elastic Block  
Store

## File Storage



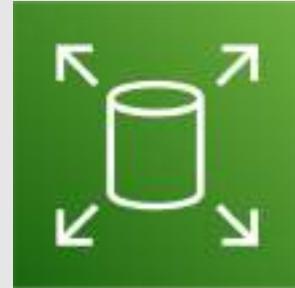
Amazon Elastic  
File System

## Object Storage



Amazon Simple  
Storage Service (S3)

# Amazon EBS Deployment and Volume Types

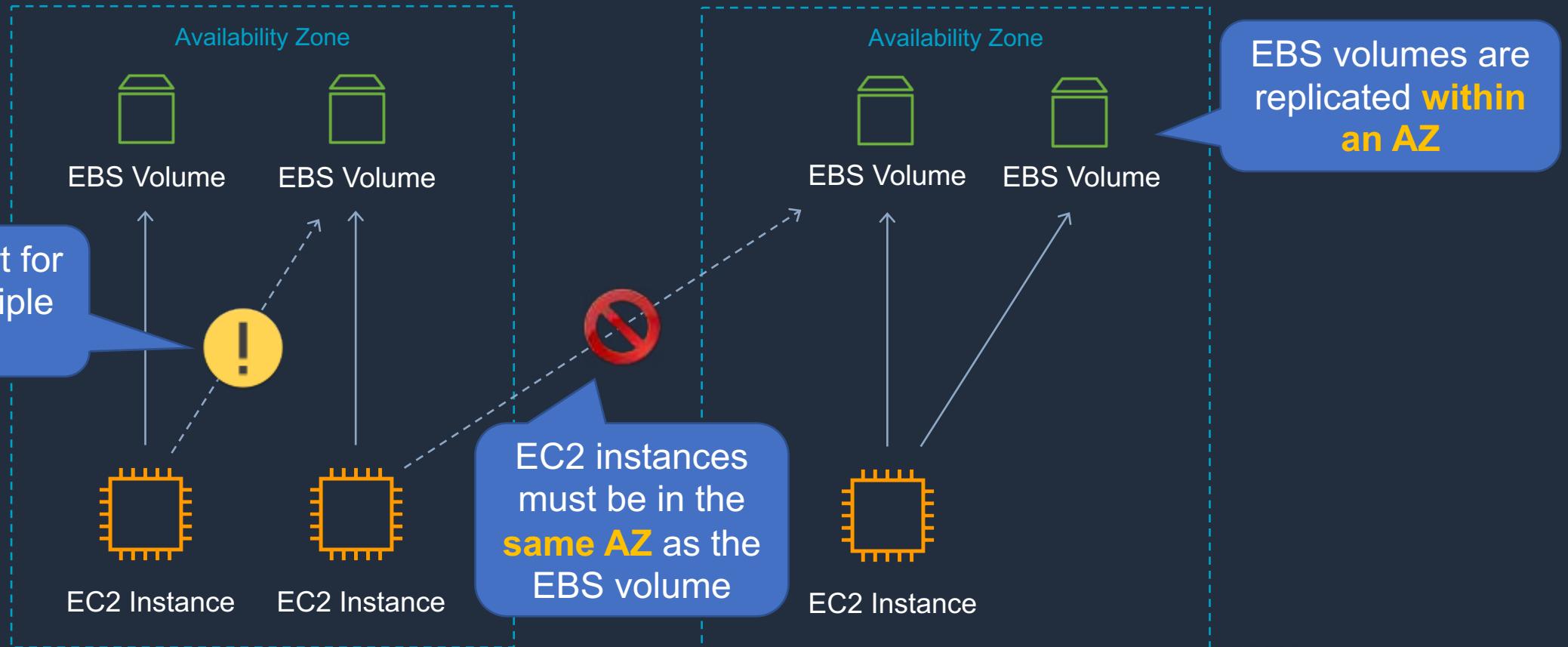




# Amazon EBS Deployment



## Amazon Elastic Block Store (EBS)





# Amazon EBS Multi-Attach



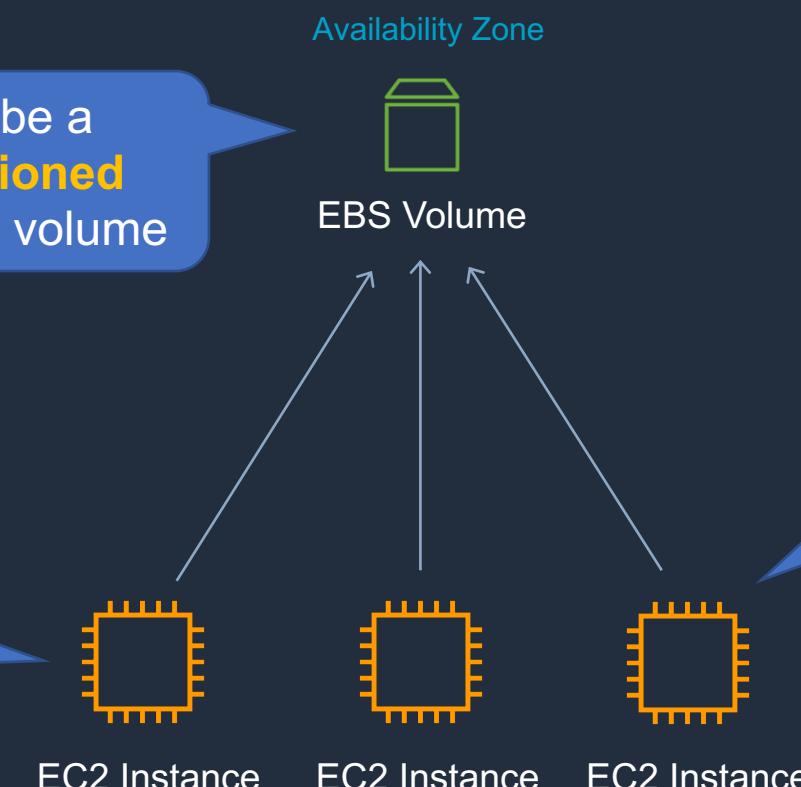
May not be on the exam yet

Available for **Nitro system-based** EC2 instances

Must be a **Provisioned IOPS** io1 volume

Must be within a **single AZ**

Up to **16 instances** can be attached to a single volume





# Amazon EBS SSD-Backed Volumes

New and **may not** be on the exam yet

New and **may not** be on the exam yet

	General Purpose SSD		Provisioned IOPS SSD		
Volume type	gp3	gp2	io2 Block Express ‡	io2	io1
Durability	99.8% - 99.9% durability (0.1% - 0.2% annual failure rate)	99.8% - 99.9% durability (0.1% - 0.2% annual failure rate)	99.999% durability (0.001% annual failure rate)	99.8% - 99.9% durability (0.1% - 0.2% annual failure rate)	99.8% - 99.9% durability (0.1% - 0.2% annual failure rate)
Use cases	<ul style="list-style-type: none"><li>Low-latency interactive apps</li><li>Development and test environments</li></ul>		Workloads that require sub-millisecond latency, and sustained IOPS performance or more than 64,000 IOPS or 1,000 MiB/s of throughput	<ul style="list-style-type: none"><li>Workloads that require sustained IOPS performance or more than 16,000 IOPS</li><li>I/O-intensive database workloads</li></ul>	
Volume size	1 GiB - 16 TiB		4 GiB - 64 TiB	4 GiB - 16 TiB	
Max IOPS per volume (16 KiB I/O)	16,000		256,000	64,000 †	
Max throughput per volume	1,000 MiB/s	250 MiB/s *	4,000 MiB/s	1,000 MiB/s †	
Amazon EBS Multi-attach	Not supported		Not supported	Supported	
Boot volume	Supported				



# Amazon EBS HDD-Backed Volumes

	Throughput Optimized HDD	Cold HDD
<b>Volume type</b>	st1	sc1
<b>Durability</b>	99.8% - 99.9% durability (0.1% - 0.2% annual failure rate)	99.8% - 99.9% durability (0.1% - 0.2% annual failure rate)
<b>Use cases</b>	<ul style="list-style-type: none"><li>• Big data</li><li>• Data warehouses</li><li>• Log processing</li></ul>	<ul style="list-style-type: none"><li>• Throughput-oriented storage for data that is infrequently accessed</li><li>• Scenarios where the lowest storage cost is important</li></ul>
<b>Volume size</b>	125 GiB - 16 TiB	125 GiB - 16 TiB
<b>Max IOPS per volume (1 MiB I/O)</b>	500	250
<b>Max throughput per volume</b>	500 MiB/s	250 MiB/s
<b>Amazon EBS Multi-attach</b>	Not supported	Not supported
<b>Boot volume</b>	Not supported	Not supported

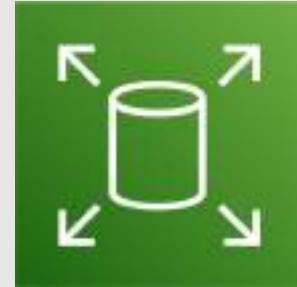


# Amazon EBS

---

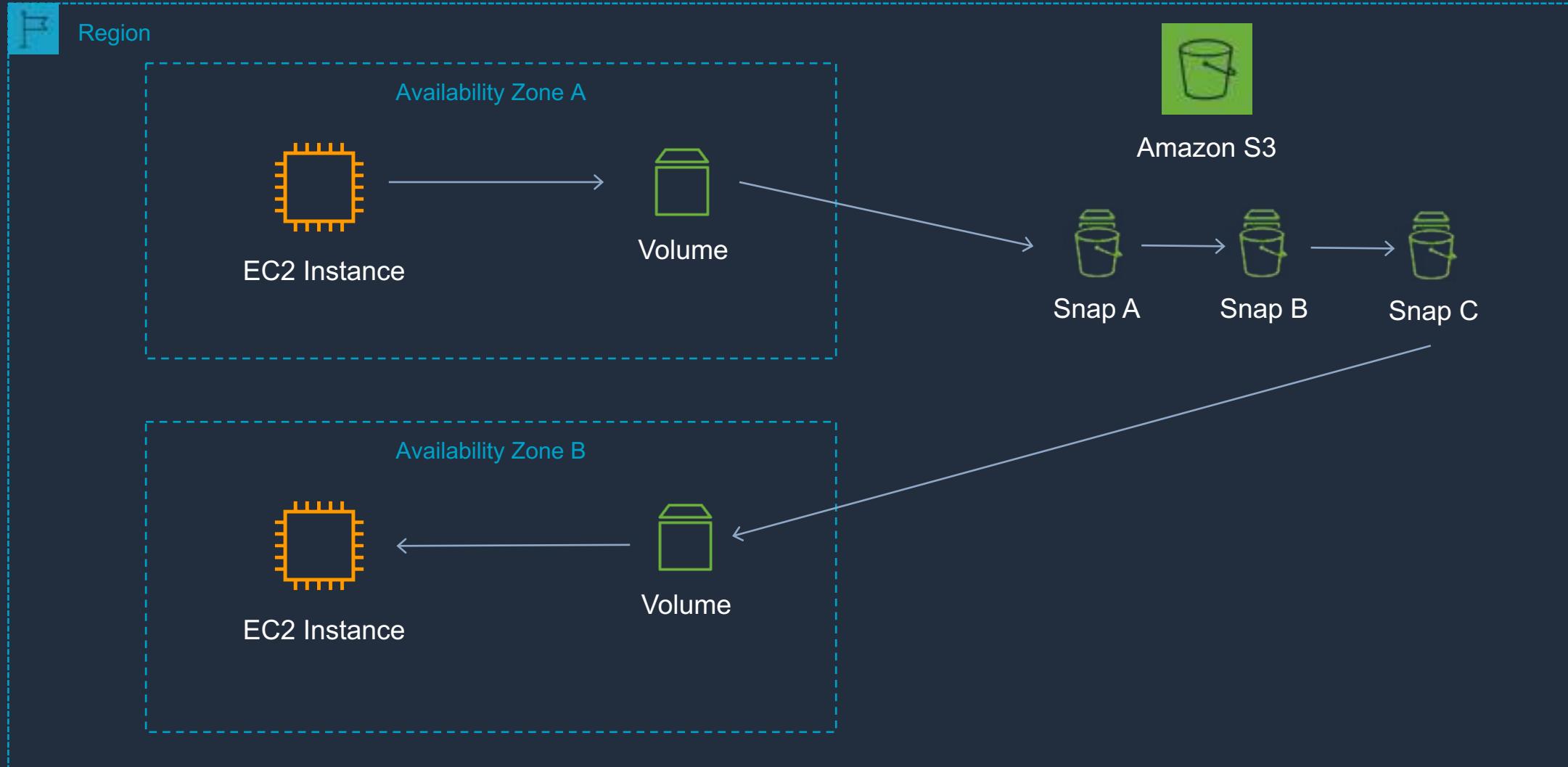
- EBS volume data persists **independently** of the life of the instance
- EBS volumes do not need to be attached to an instance
- You can attach multiple EBS volumes to an instance
- You can use multi-attach to attach a volume to multiple instances but with some constraints
- EBS volumes must be in the **same AZ** as the instances they are attached to
- Root EBS volumes **are deleted** on termination by default
- Extra non-boot volumes **are not deleted** on termination by default

# Amazon EBS Copying, Sharing and Encryption



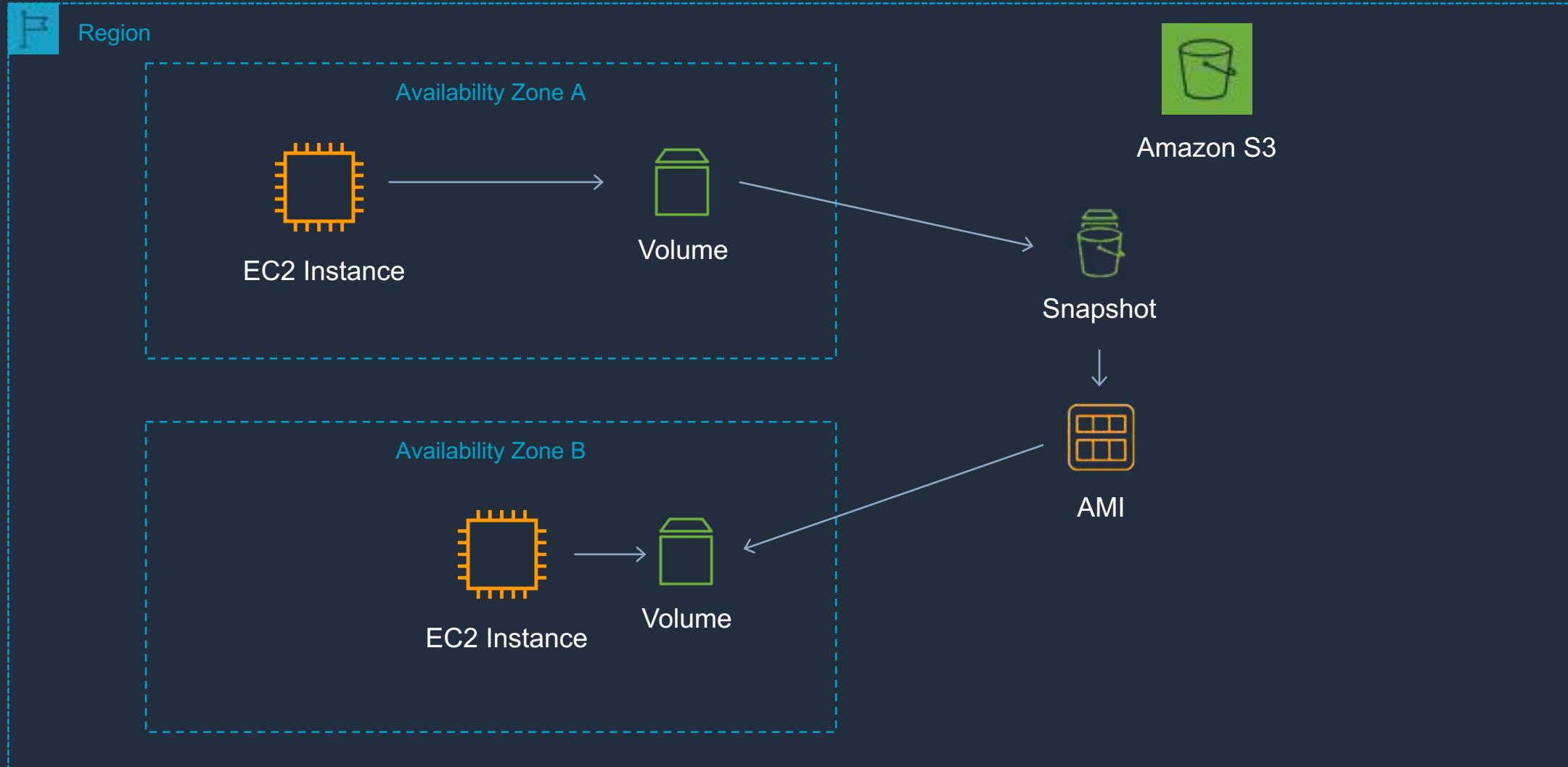


# Amazon EBS Copying, Sharing and Encryption





# Take Snapshot, Create AMI, Launch New Instance





# Copying and Sharing AMIs and Snapshots



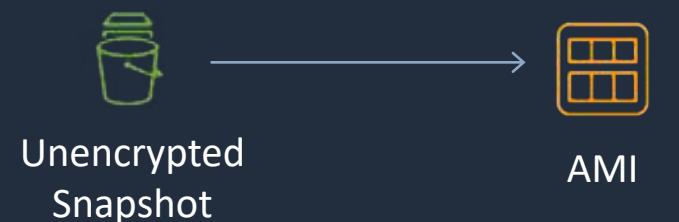
- Encryption state retained
- Same region



- Can be encrypted
- Can change regions



- Can be encrypted
- Can change AZ



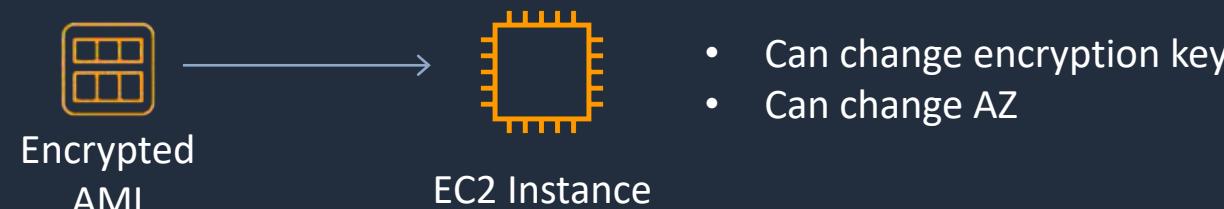
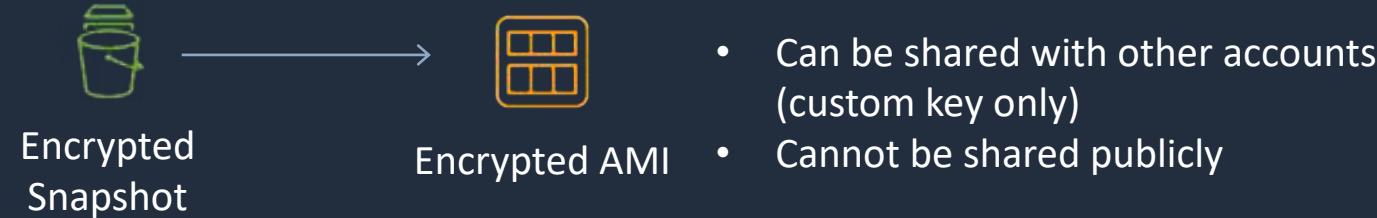
- Cannot be encrypted
- Can be shared with other accounts
- Can be shared publicly



- Can change encryption key
- Can change regions



# Copying and Sharing AMIs and Snapshots

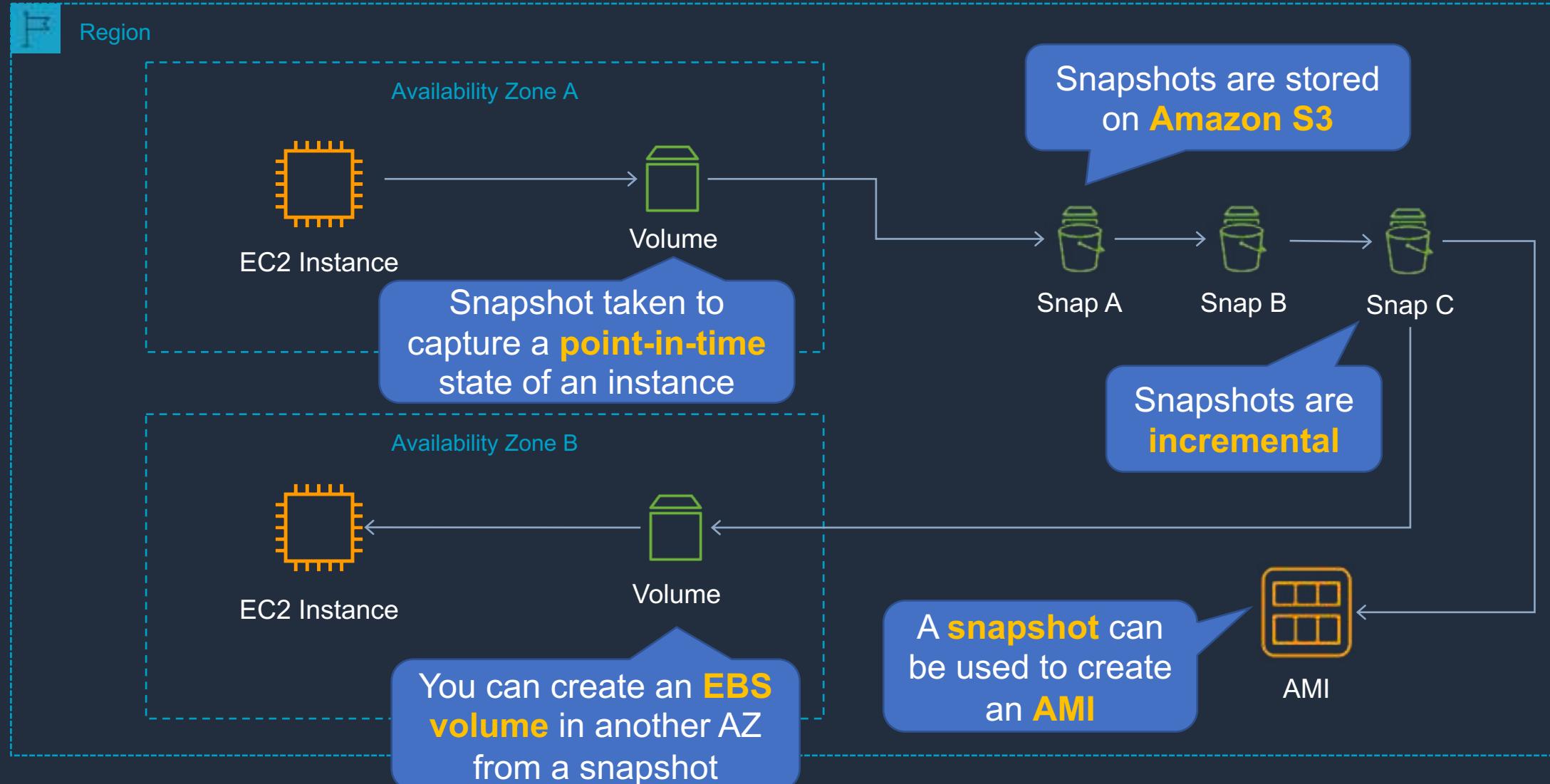


# Amazon EBS Snapshots and DLM





# Amazon EBS Snapshots





# Amazon Data Lifecycle Manager (DLM)

---

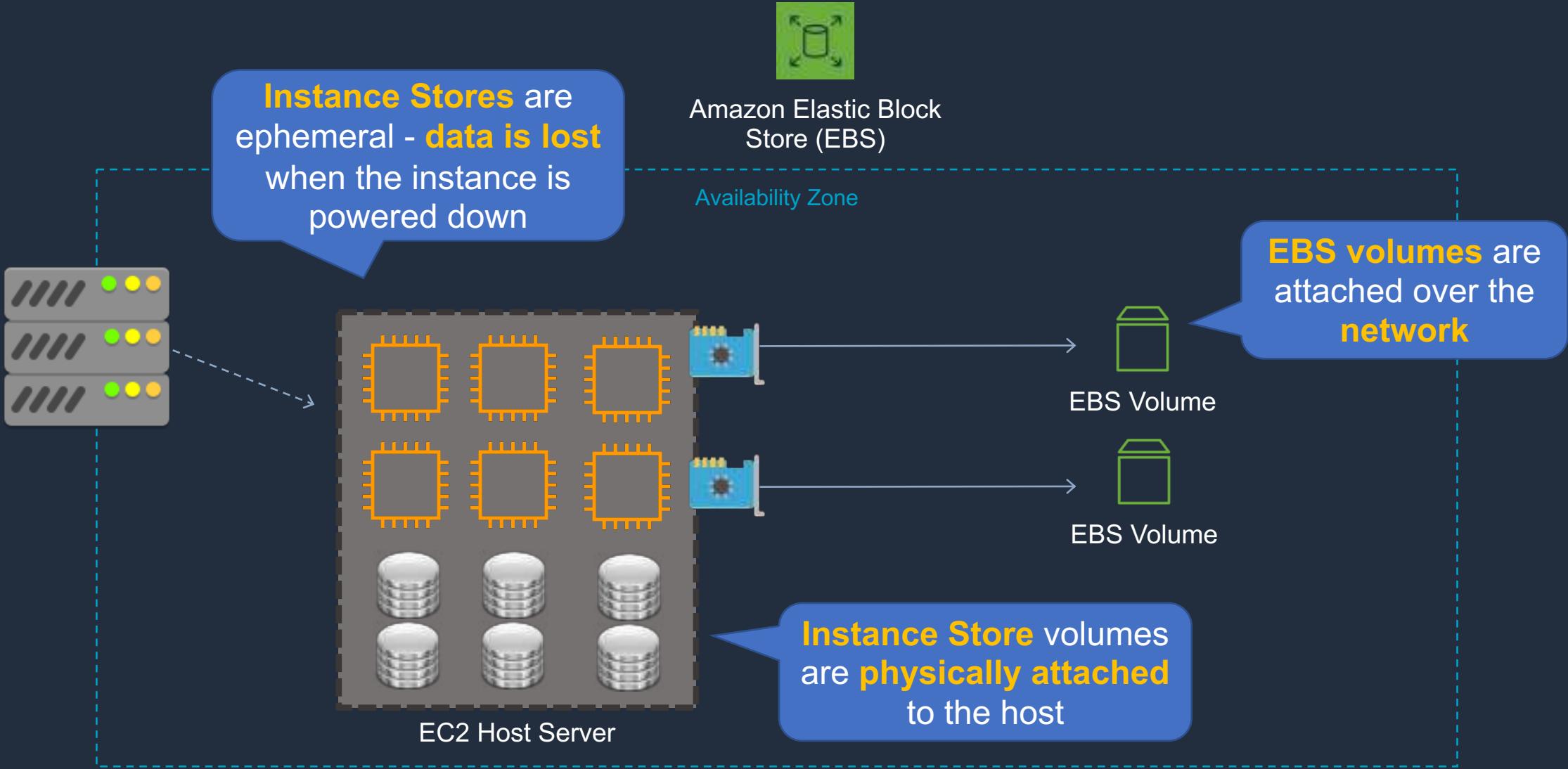
- DLM automates the creation, retention, and deletion of EBS snapshots and EBS-backed AMIs
- DLM helps with the following:
  - Protects valuable data by enforcing a regular backup schedule
  - Create standardized AMIs that can be refreshed at regular intervals
  - Retain backups as required by auditors or internal compliance
  - Reduce storage costs by deleting outdated backups
  - Create disaster recovery backup policies that back up data to isolated accounts

# EC2 Instance Store Volumes





# EBS vs instance store





# EBS vs instance store

---

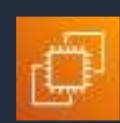
- Instance store volumes are high performance local disks that are physically attached to the host computer on which an EC2 instance runs
- Instance stores are ephemeral which means the data is lost when powered off (non-persistent)
- Instance stores are ideal for temporary storage of information that changes frequently, such as buffers, caches, or scratch data
- Instance store volume root devices are created from AMI templates stored on S3
- Instance store volumes cannot be detached/reattached

# Creating and Attaching EBS Volumes



# Using Amazon Machine Images (AMIs)





# Amazon Machine Images (AMIs)

---

---

- An **Amazon Machine Image** (AMI) provides the information required to launch an instance
- An AMI includes the following:
  - One or more EBS snapshots, or, for instance-store-backed AMIs, a template for the root volume of the instance (for example, an operating system, an application server, and applications)
  - Launch permissions that control which AWS accounts can use the AMI to launch instances
  - A block device mapping that specifies the volumes to attach to the instance when it's launched
- AMIs come in three main categories:
  - **Community AMIs** - free to use, generally you just select the operating system you want
  - **AWS Marketplace AMIs** - pay to use, generally come packaged with additional, licensed software
  - **My AMIs** - AMIs that you create yourself

# EC2 Image Builder



# Migrate EBS Volume between AZs



# Working with EBS Snapshots and DLM



# Using RAID with EBS





# Using RAID with EBS

---

---

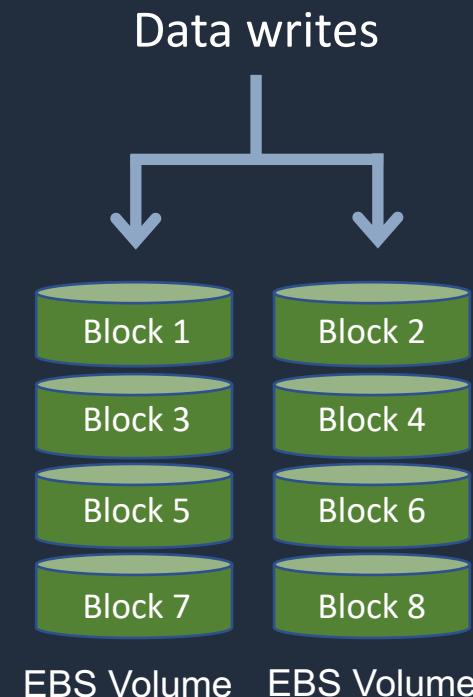
- RAID stands for Redundant Array of Independent disks
- Not provided by AWS, you must configure through your operating system
- RAID 0 and RAID 1 are potential options on EBS
- RAID 5 and RAID 6 are not recommended by AWS



# Using RAID with EBS

---

- RAID 0 is used for striping data across disks (performance):
  - Use 2 or more disks
  - If one disk fails, the entire RAID set fails

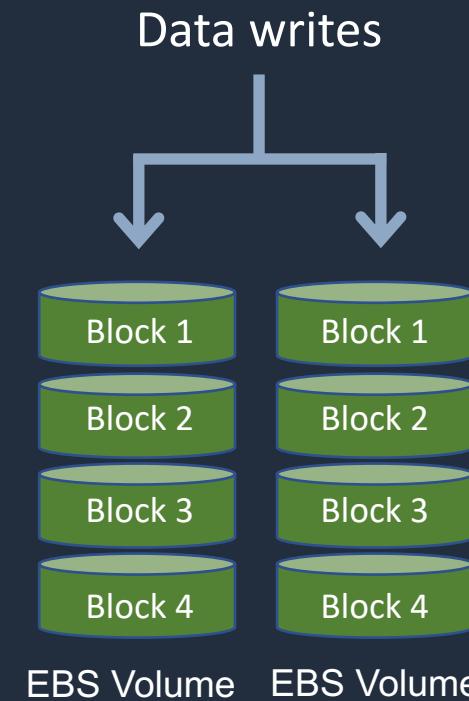




# Using RAID with EBS

---

- RAID 1 is used for mirroring data across disks (redundancy / fault tolerance):
  - If one disk fails, the other disk is still working
  - Data gets sent to 2 EBS volumes at the same time

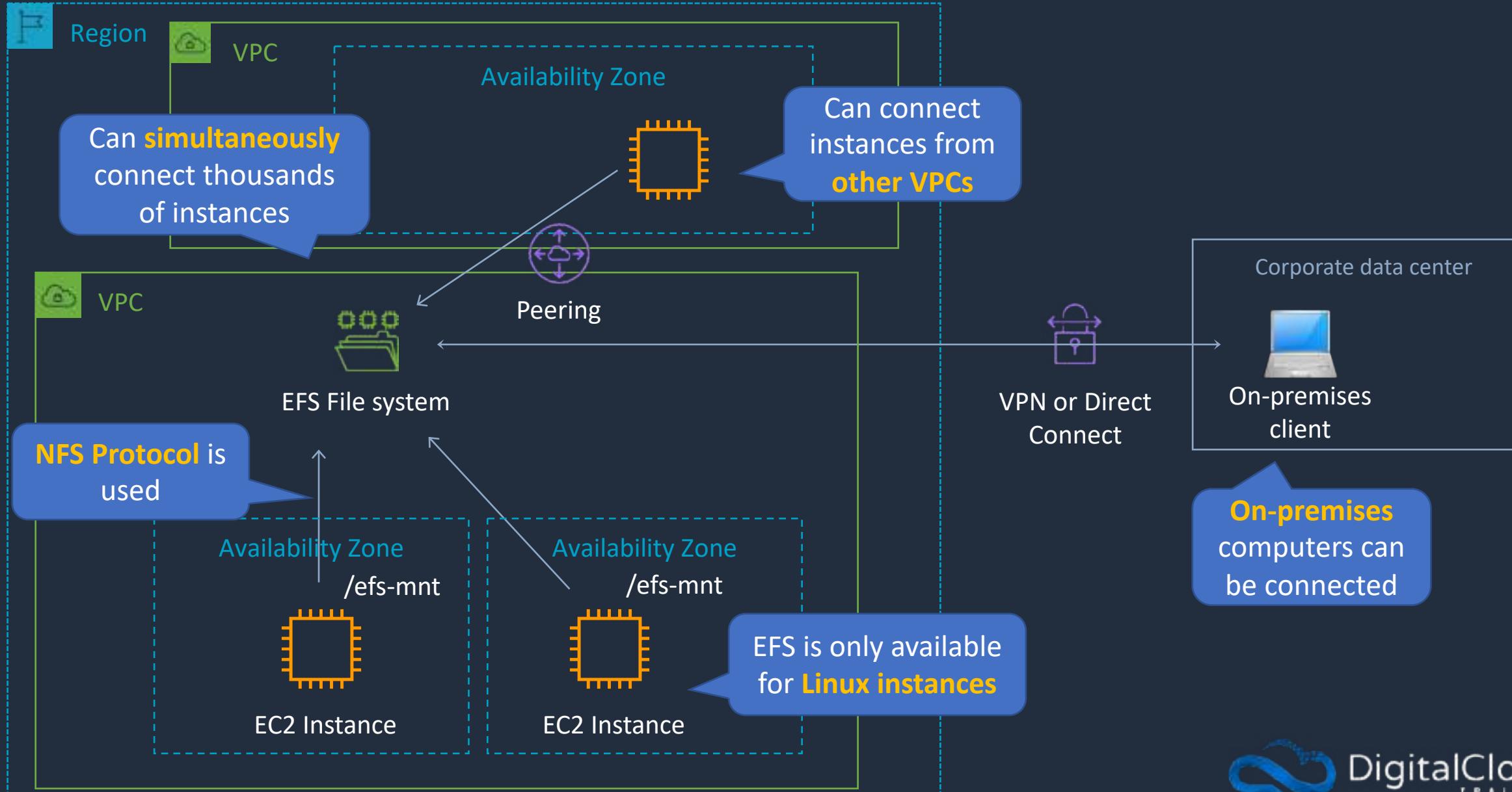


# Amazon Elastic File System (EFS)



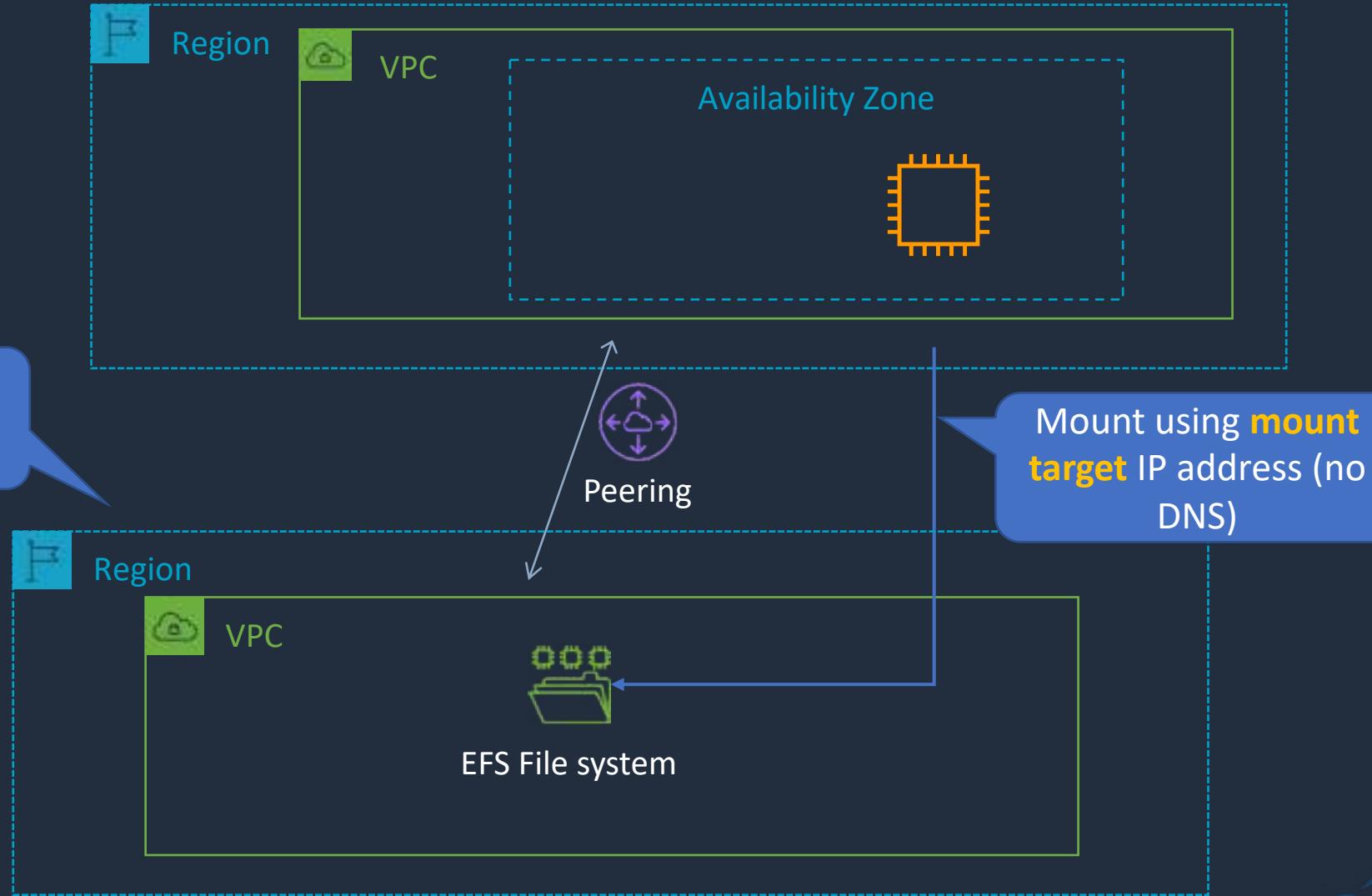


# Amazon Elastic File System (EFS) Overview





# Amazon Elastic File System (EFS) Overview



# Create EFS Filesystem



# Amazon FSx

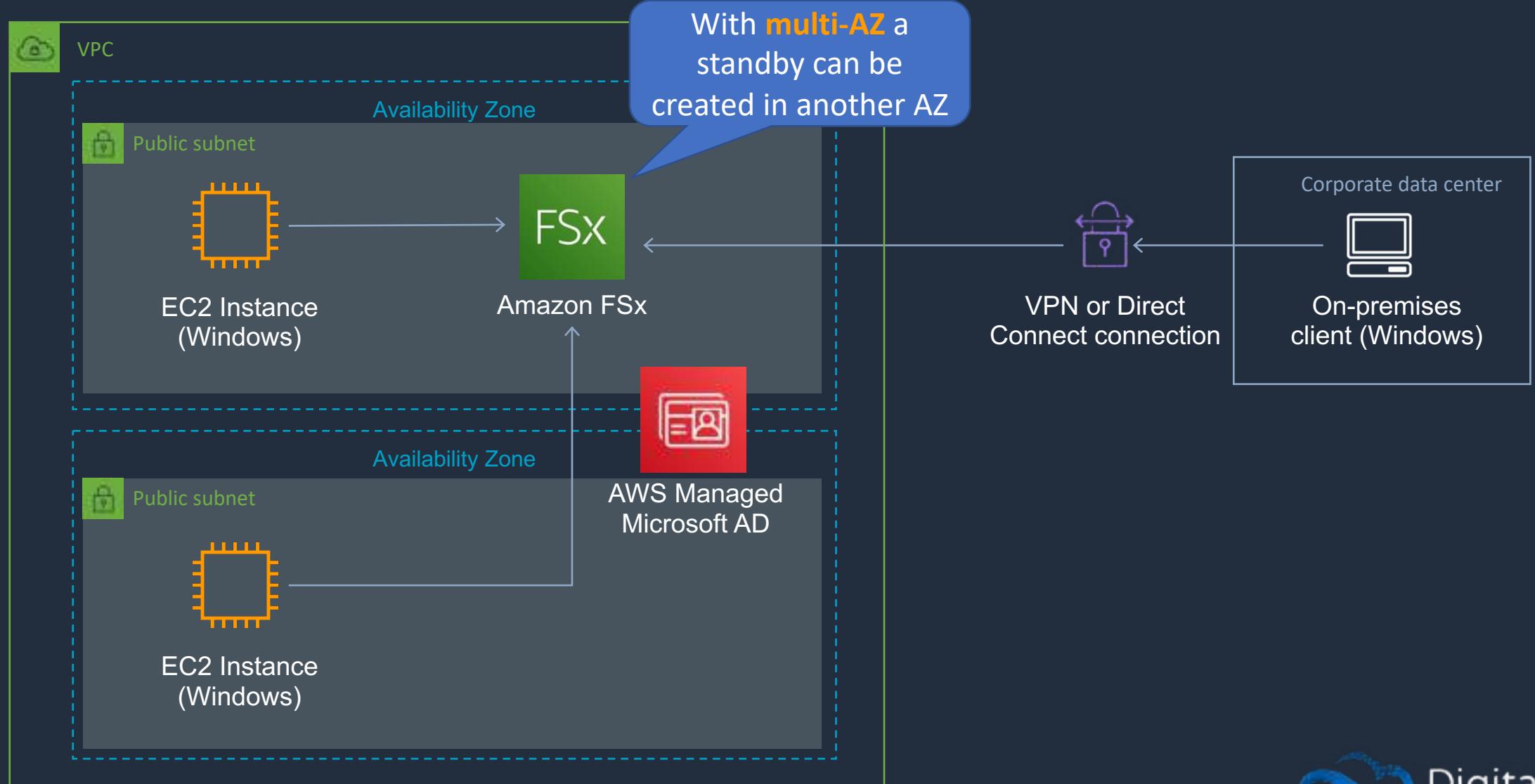


- Amazon FSx provides fully managed third-party file systems
- Amazon FSx provides you with two file systems to choose from:
  - **Amazon FSx for Windows File Server** for Windows-based applications
  - **Amazon FSx for Lustre** for compute-intensive workloads

# Amazon FSx for Windows File Server

- Provides a fully managed native Microsoft Windows file system
- Full support for the SMB protocol, Windows NTFS, and Microsoft Active Directory (AD) integration
- Supports Windows-native file system features:
  - Access Control Lists (ACLs), shadow copies, and user quotas.
  - NTFS file systems that can be accessed from up to thousands of compute instances using the SMB protocol
- **High availability:** replicates data within an Availability Zone (AZ)
- **Multi-AZ:** file systems include an active and standby file server in separate AZs

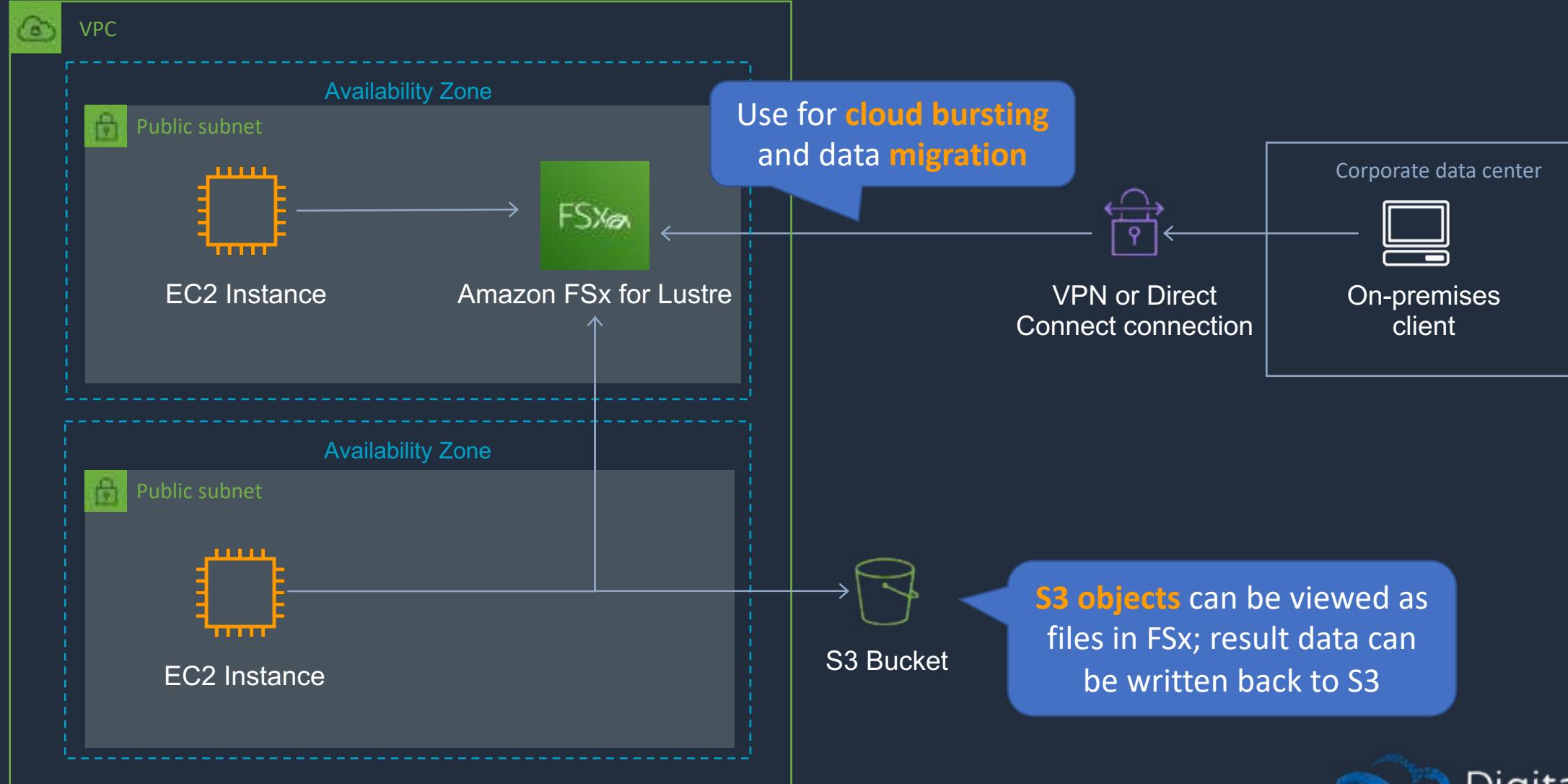
# Amazon FSx for Windows File Server



# Amazon FSx for Lustre

- High-performance file system optimized for fast processing of workloads such as:
  - Machine learning
  - High performance computing (HPC)
  - Video processing
  - Financial modeling
  - Electronic design automation (EDA)
- Works natively with S3, letting you transparently access your S3 objects as files
- Your S3 objects are presented as files in your file system, and you can write your results back to S3
- Provides a POSIX-compliant file system interface

# Amazon FSx for Lustre

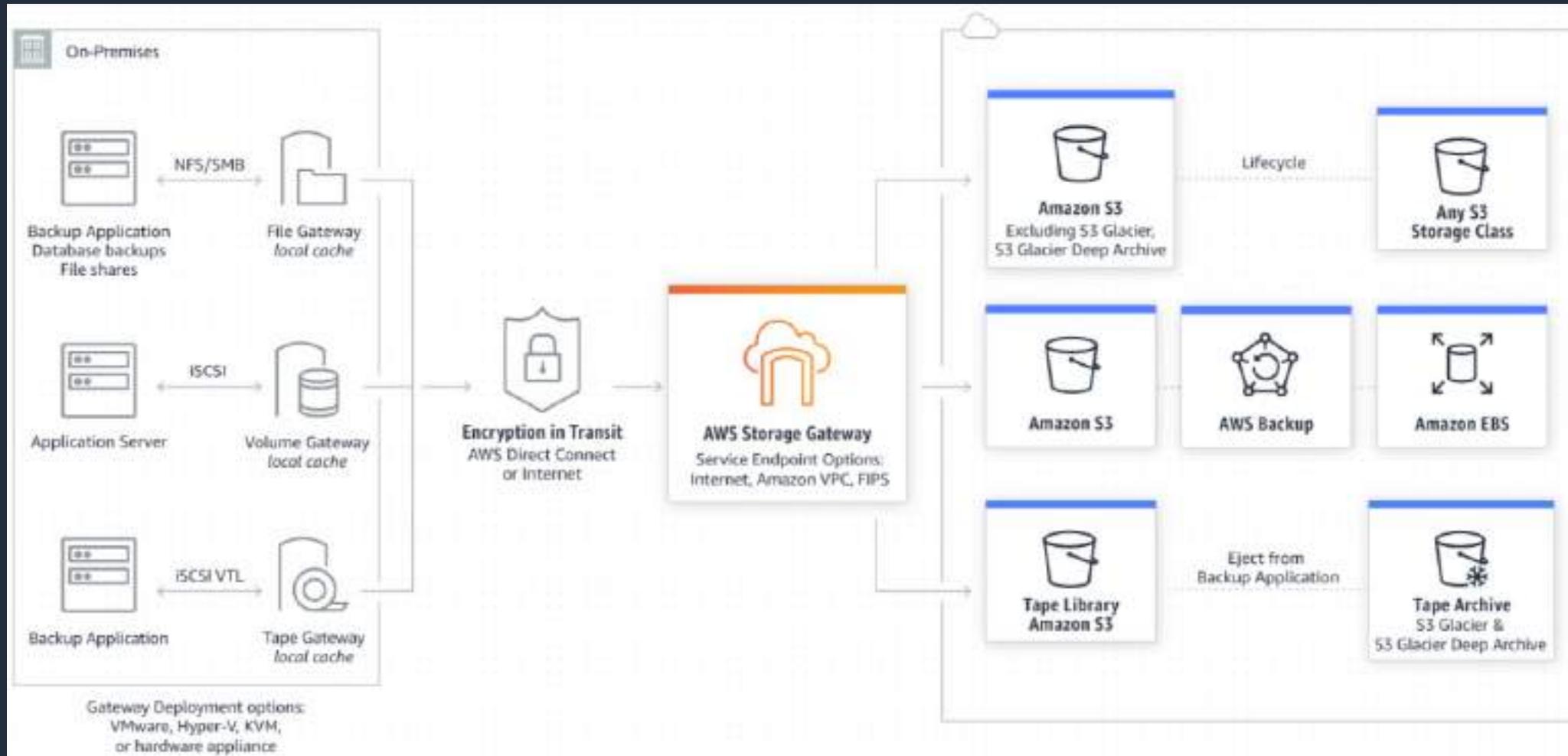


# AWS Storage Gateway



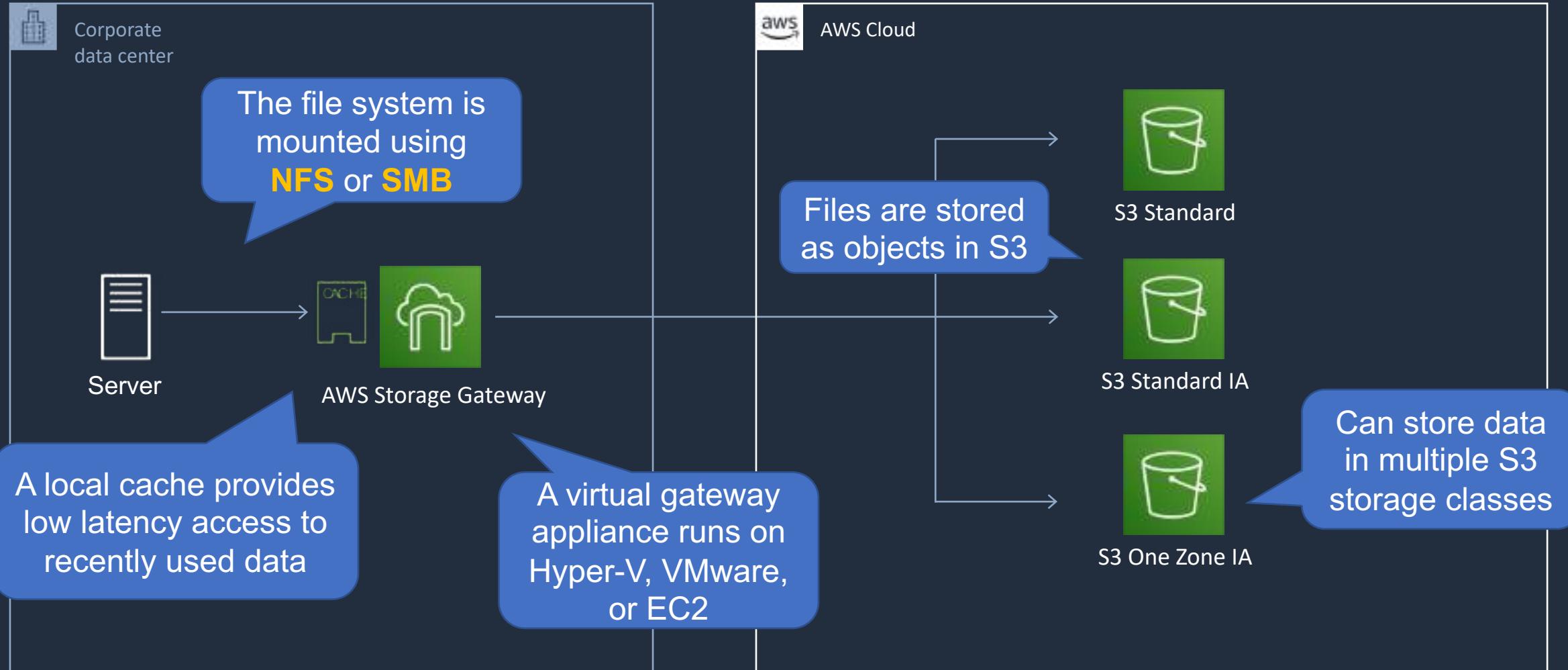


# AWS Storage Gateway





# AWS Storage Gateway – File Gateway





# AWS Storage Gateway – File Gateway

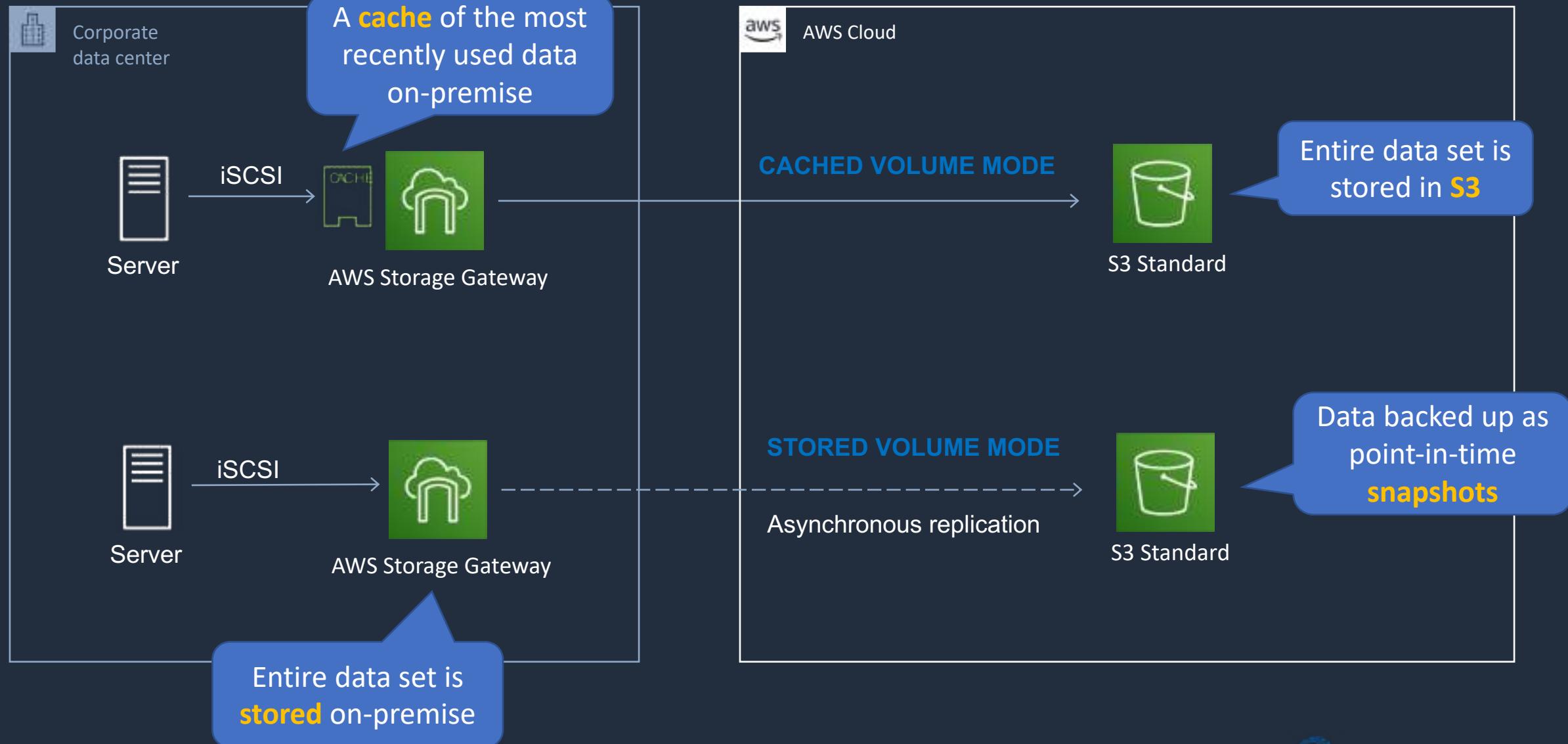
---

---

- File gateway provides a virtual **on-premises file server**
- Store and retrieve files as objects in Amazon S3
- Use with on-premises applications, and EC2-based applications that need file storage in S3 for object-based workloads
- File gateway offers **SMB** or **NFS**-based access to data in Amazon S3 with local caching



# AWS Storage Gateway - Volume Gateway





# AWS Storage Gateway - Volume Gateway

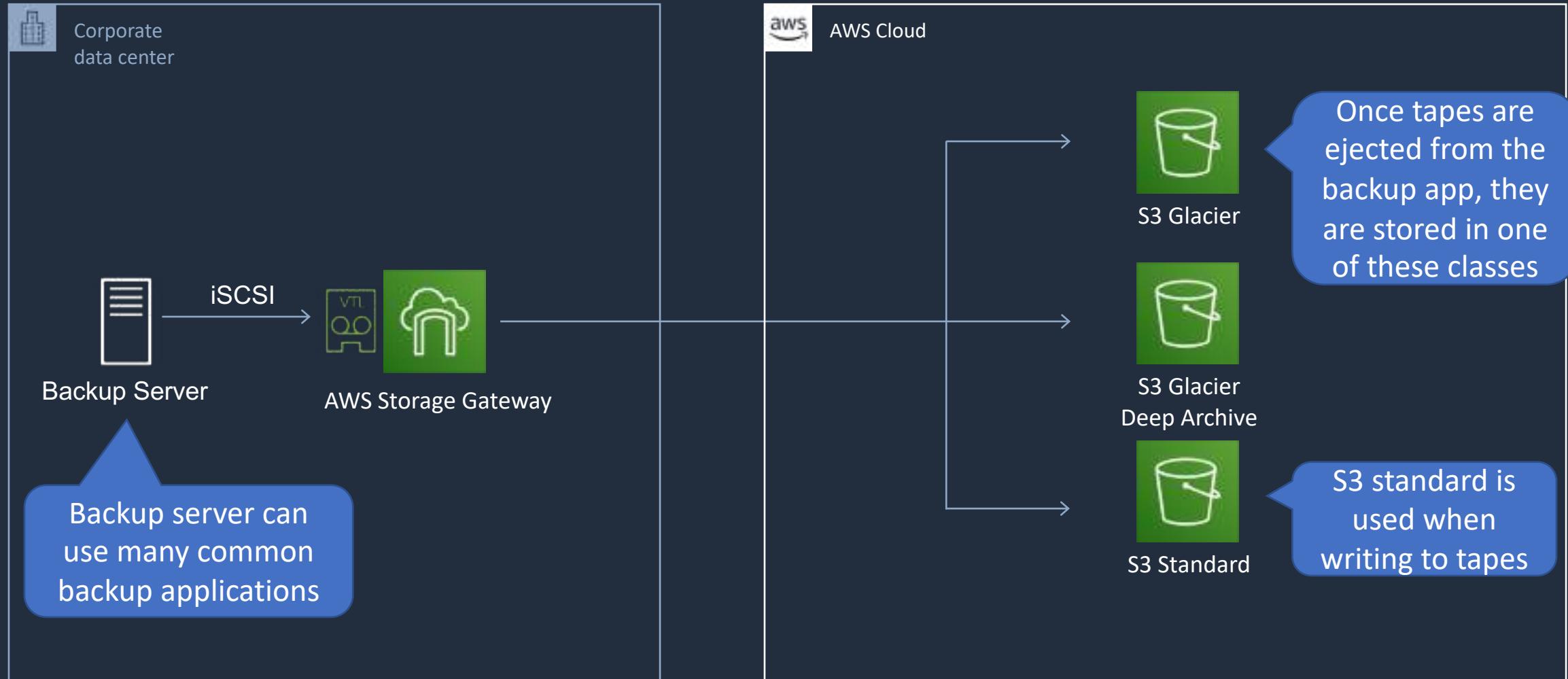
---

---

- The volume gateway supports block-based volumes
- Block storage – iSCSI protocol
- **Cached Volume mode** – the entire dataset is stored on S3 and a cache of the most frequently accessed data is cached on-site
- **Stored Volume mode** – the entire dataset is stored on-site and is asynchronously backed up to S3 (EBS point-in-time snapshots).  
Solutions are incremental and compressed



# AWS Storage Gateway - Tape Gateway





# AWS Storage Gateway - Tape Gateway

---

---

- Used for backup with popular backup software
- Each gateway is preconfigured with a media changer and tape drives. Supported by NetBackup, Backup Exec, Veeam etc.
- When creating virtual tapes, you select one of the following sizes: 100 GB, 200 GB, 400 GB, 800 GB, 1.5 TB, and 2.5 TB
- A tape gateway can have up to 1,500 virtual tapes with a maximum aggregate capacity of 1 PB
- All data transferred between the gateway and AWS storage is encrypted using SSL
- All data stored by tape gateway in S3 is encrypted server-side with Amazon S3-Managed Encryption Keys (SSE-S3)

# AWS Backup



# Architecture Patterns – Block and File Storage





# Architecture Patterns – Block and File Storage

---

---

## Requirement

Simple method of backing up Amazon EBS volumes is required that is fully automated

## Solution

Use Data Lifecycle Manager to create a backup schedule

A distributed application has many nodes that each hold a copy of data that is synchronized between them. Need the best performance

Use instance stores for storing the data with the best performance

Application must startup quickly when launched by ASG but requires app dependencies and code to be installed

Create an AMI that includes the application dependencies and code



# Architecture Patterns – Block and File Storage

## Requirement

Many Linux instances must be attached to a shared filesystem that scales elastically

Company requires a managed file system that uses the NTFS file system

On-premises servers must be able to attach a block storage system locally. Data should be backed up to S3 as snapshots

## Solution

Create an Amazon EFS file system and mount from each instance

Use Amazon FSx for Windows File Server

Deploy an AWS Storage Gateway volume gateway in stored volume mode



# Architecture Patterns – Block and File Storage

---

---

## Requirement

An Amazon EBS volume must be moved between Regions

## Solution

Take a snapshot and copy the snapshot between Regions

Root EBS volumes for a critical application must not be deleted on termination

Modify the delete on termination attribute when launching the EC2 instances

On-premises servers use NFS to attach a file system. The file system should be replaced with an AWS service that uses Amazon S3 with a local cache

Deploy an AWS Storage Gateway file gateway

# SECTION 10

## Docker Containers and ECS

# Docker Containers and Microservices





# Server Virtualization vs Containers

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



Server



Server



# Docker Containers

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

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



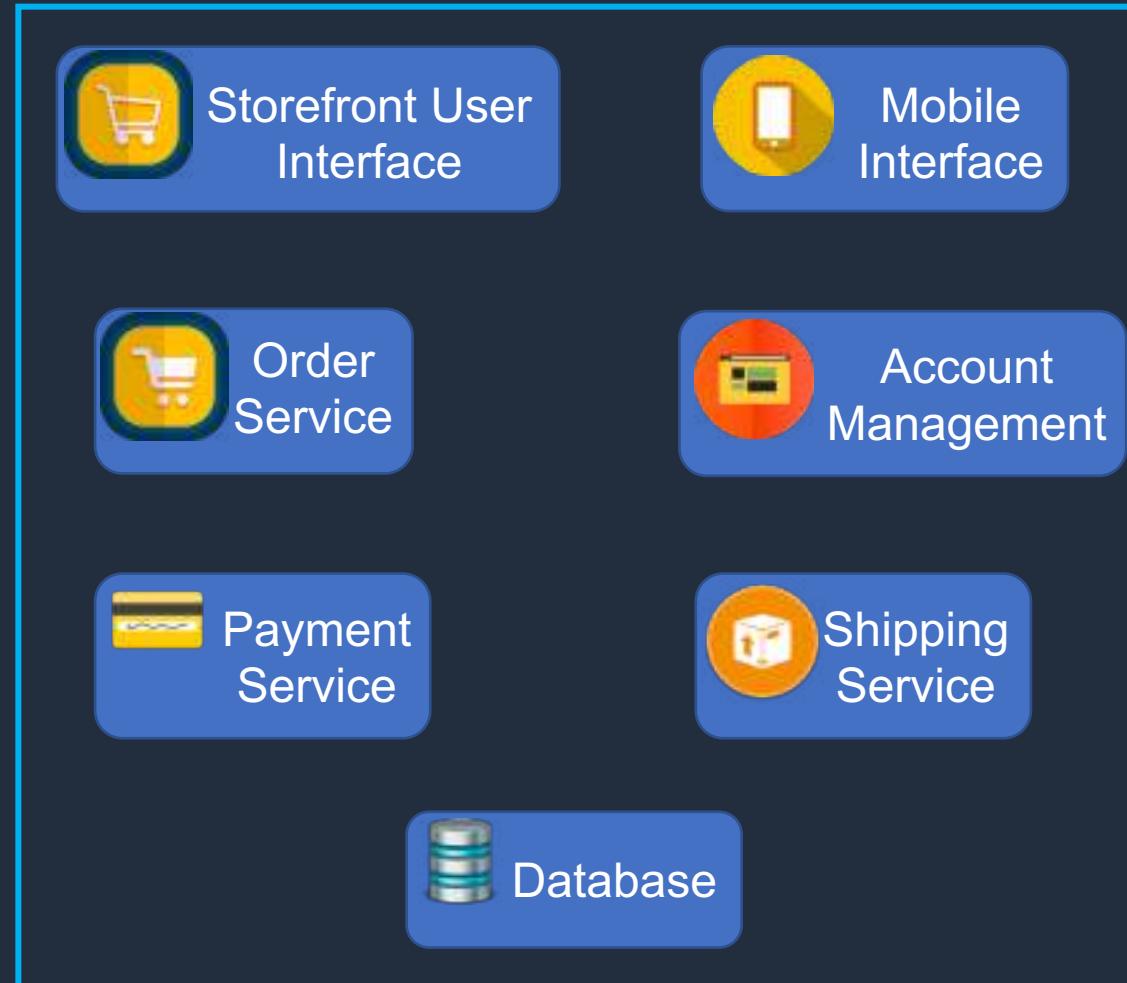
Containers are very resource **efficient**

Each container is **isolated** from other containers

# Monolithic Application

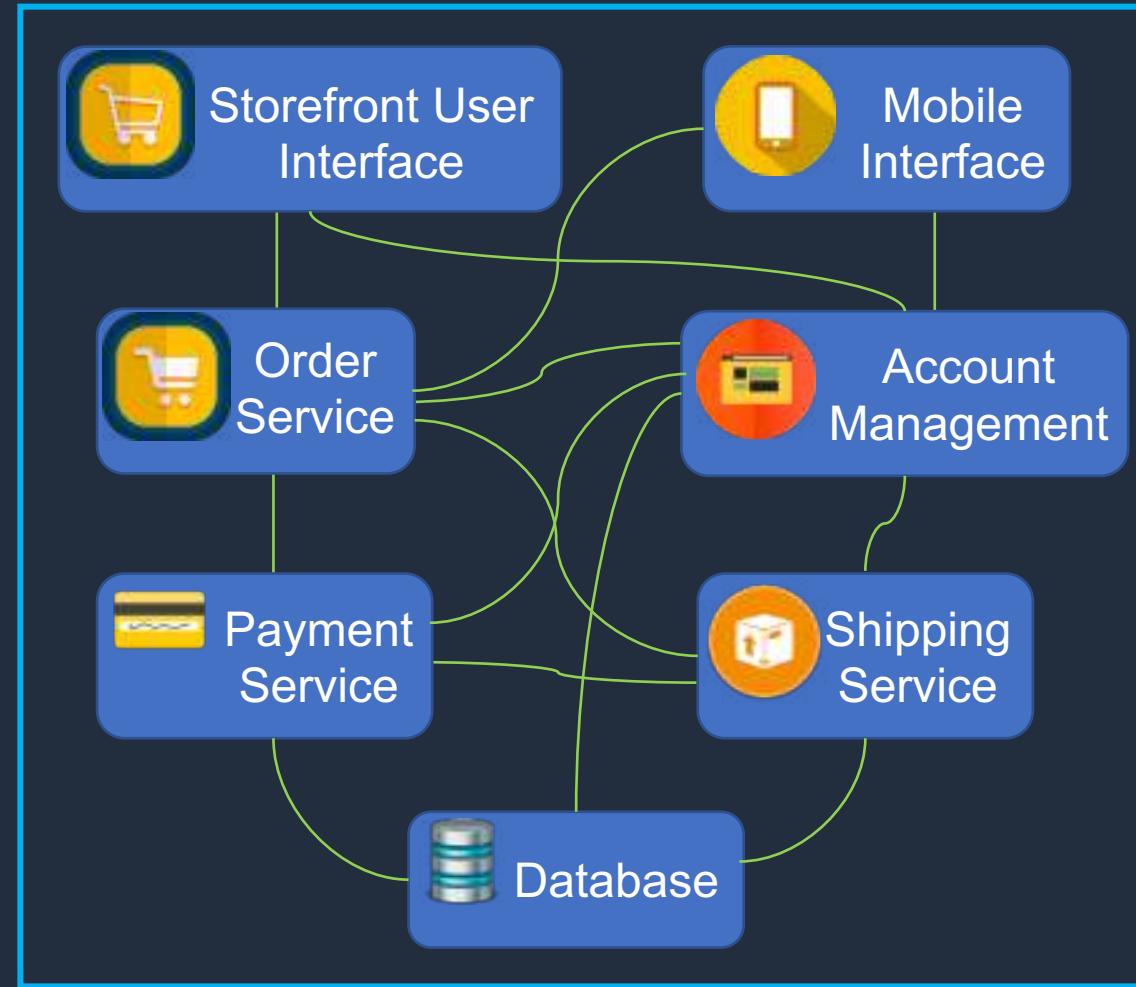
---

---



# Monolithic Application

Updates to, or failures of, any single component can take down the whole application



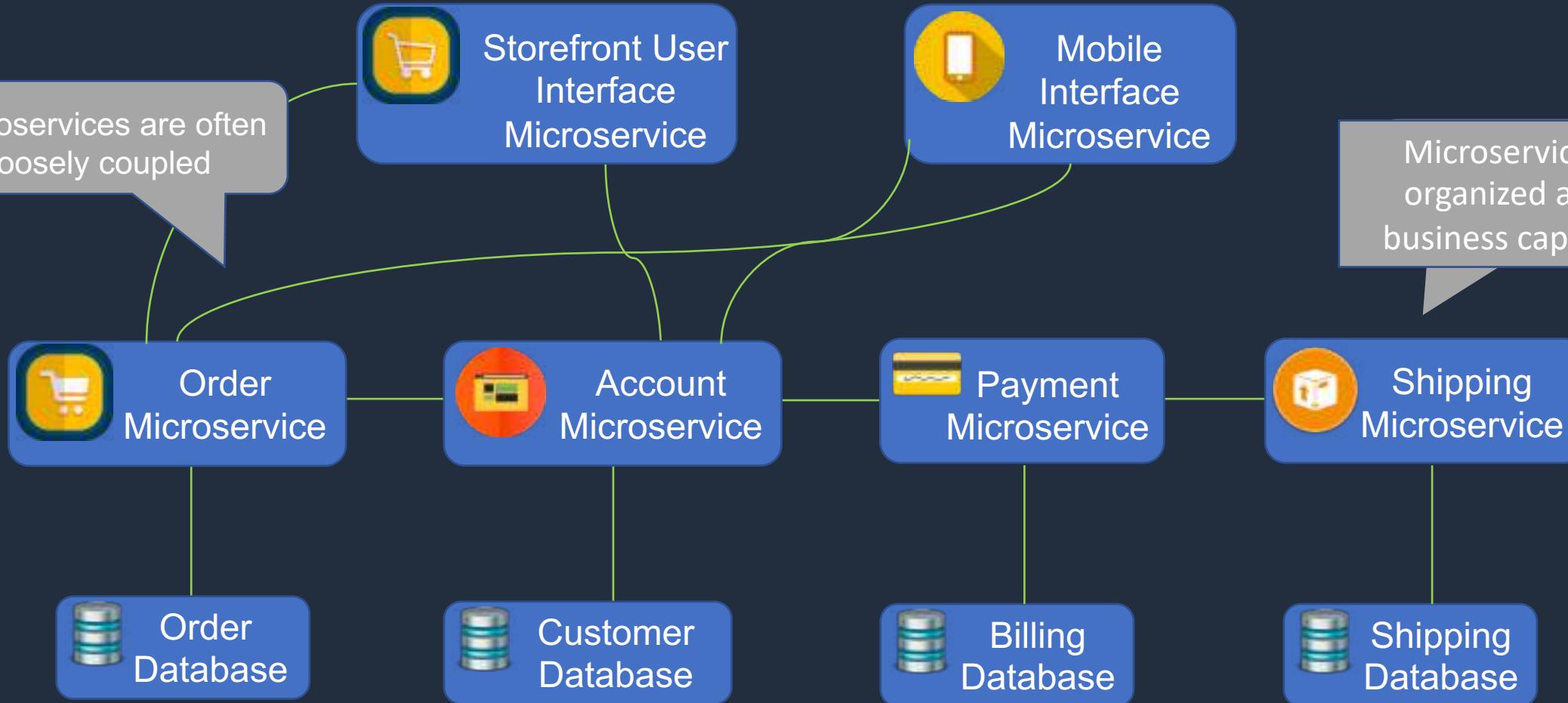
The user interface, business logic, and data access layer are combined on a single platform



# Microservices Application

A microservice is an independently deployable unit of code

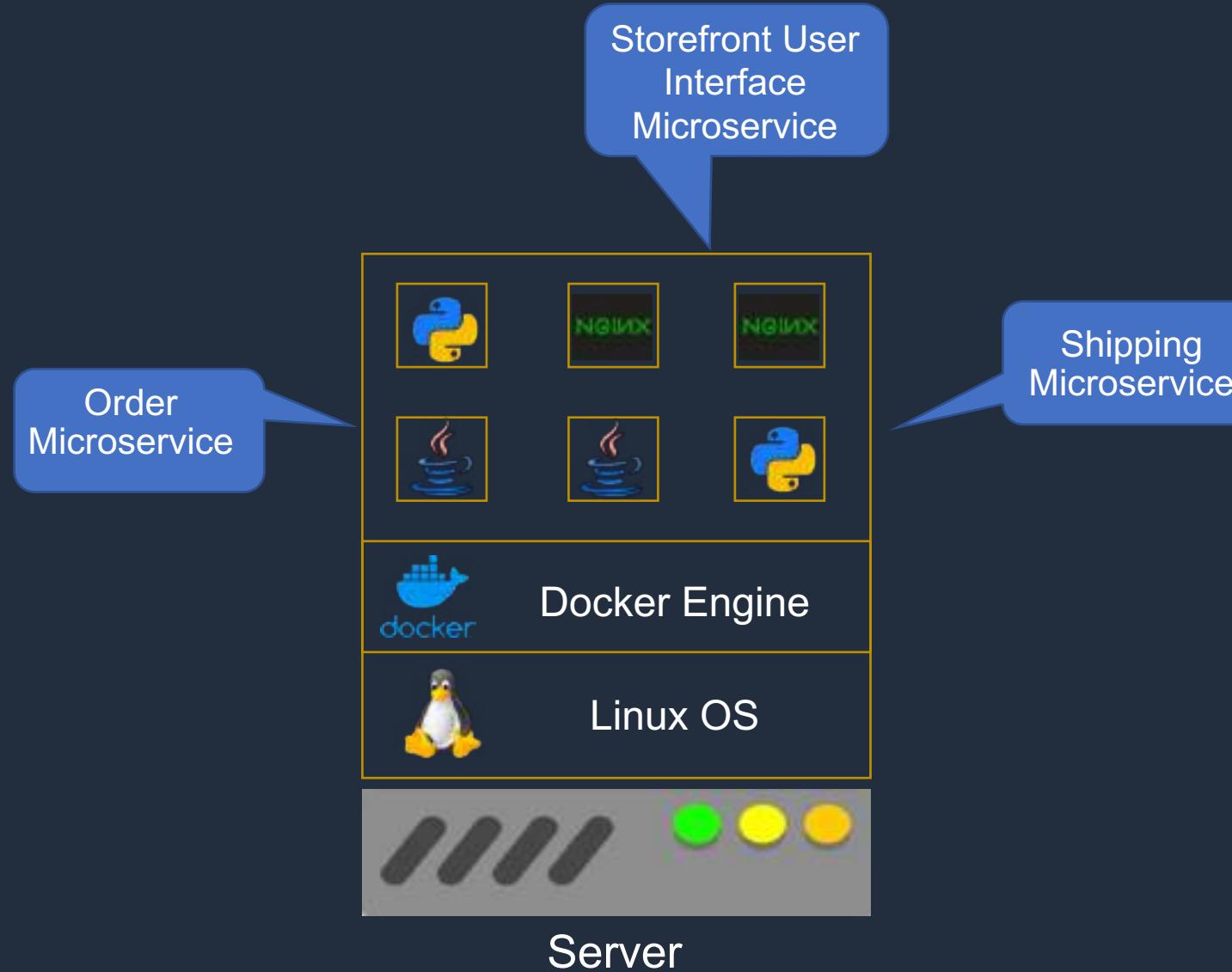
Microservices are often loosely coupled



Microservices are organized around business capabilities



# Microservices Application





# Microservices Application

Microservices can also be  
**spread** across hosts

**Many instances** of each microservice  
can run on each host



# Amazon Elastic Container Service (ECS)





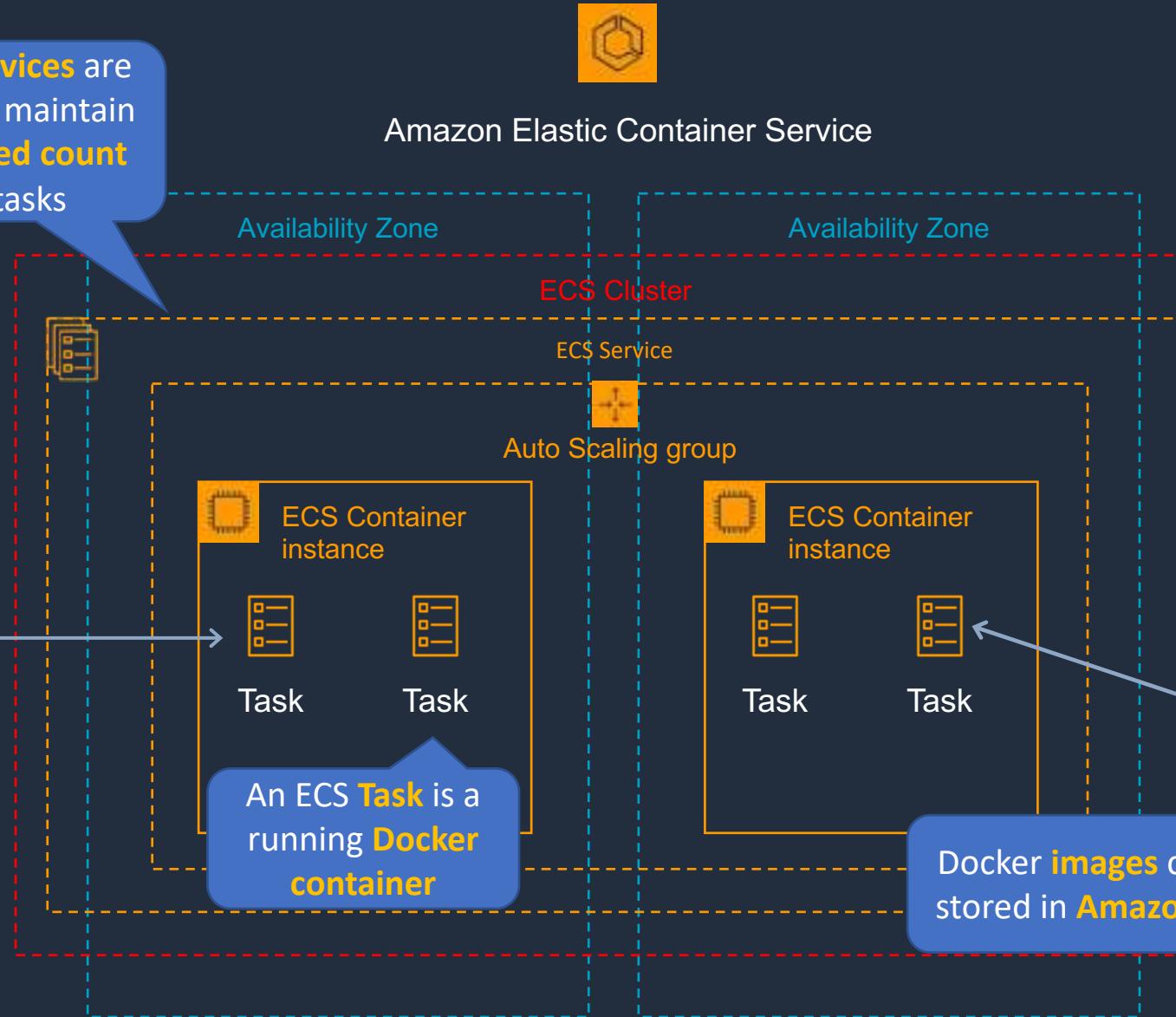
# Amazon ECS

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
    }
  ]
}
```

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

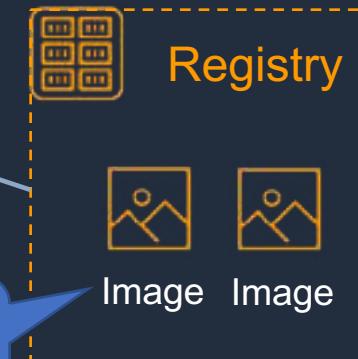


Amazon Elastic Container Service

An Amazon **ECS Cluster** is a logical grouping of **tasks** or **services**



Amazon Elastic Container Registry





# 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 (NEW)** – enables the use of Amazon ECS control plane to manage on-premises implementations



# Amazon ECS Components

---

---

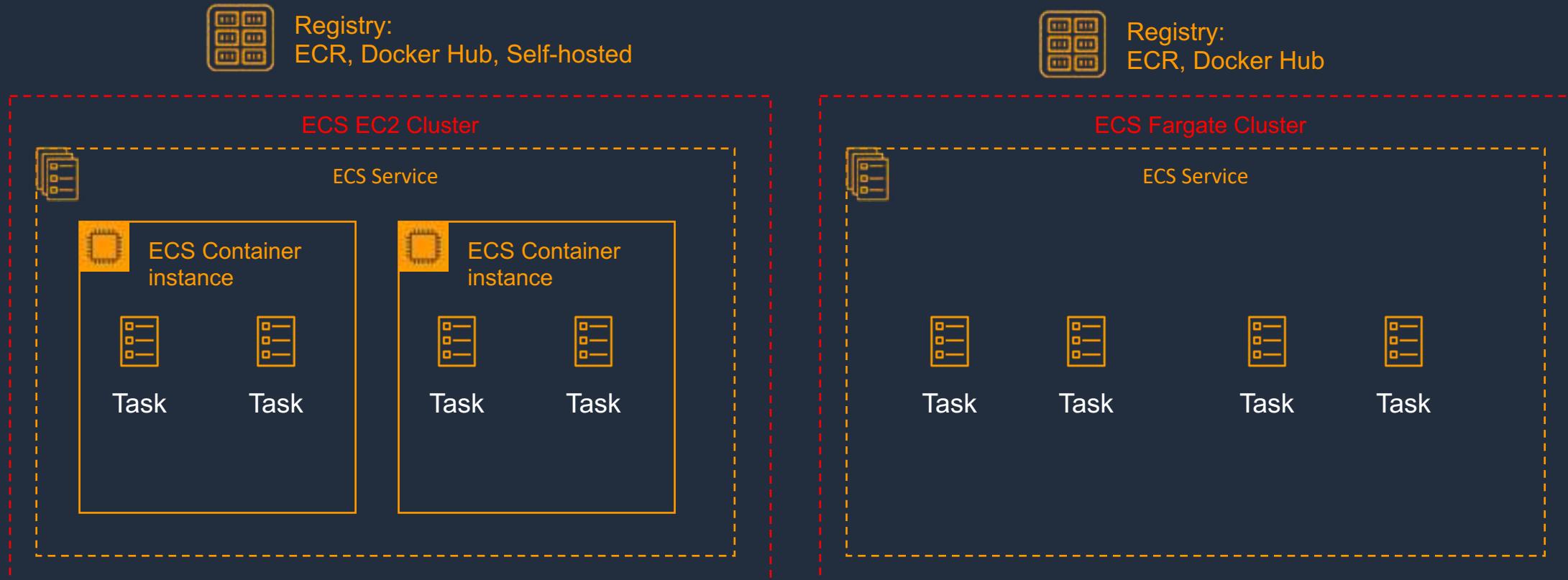
Elastic Container Service (ECS)	Description
Cluster	Logical grouping of EC2 instances
Container instance	EC2 instance running the the ECS agent
Task Definition	Blueprint that describes how a docker container should launch
Task	A running container using settings in a Task Definition
Service	Defines long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

# Amazon ECS Launch Types





# Launch Types – EC2 and Fargate



## EC2 Launch Type

- You explicitly provision EC2 instances
- You're responsible for managing EC2 instances
- Charged per running EC2 instance
- Docker volumes, EFS, and FSx for Windows File Server
- You handle cluster optimization
- More granular control over infrastructure

## Fargate Launch Type

- Fargate automatically provisions resources
- Fargate provisions and manages compute
- Charged for running tasks
- EFS integration
- Fargate handles cluster optimization
- Limited control, infrastructure is automated

# Launch Task on AWS Fargate

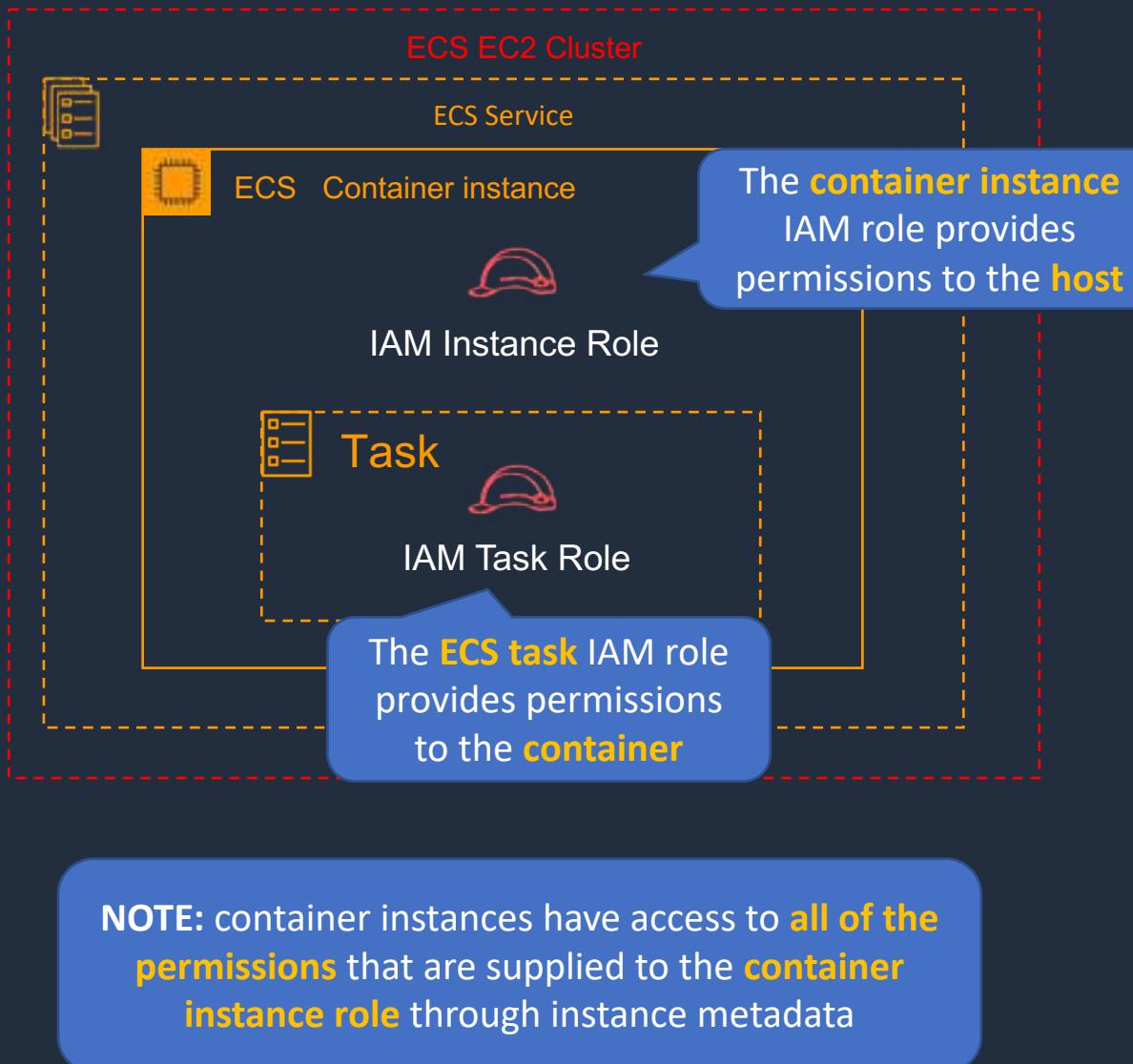


# Amazon ECS and IAM Roles





# ECS and IAM Roles



AmazonEC2ContainerServiceforEC2Role

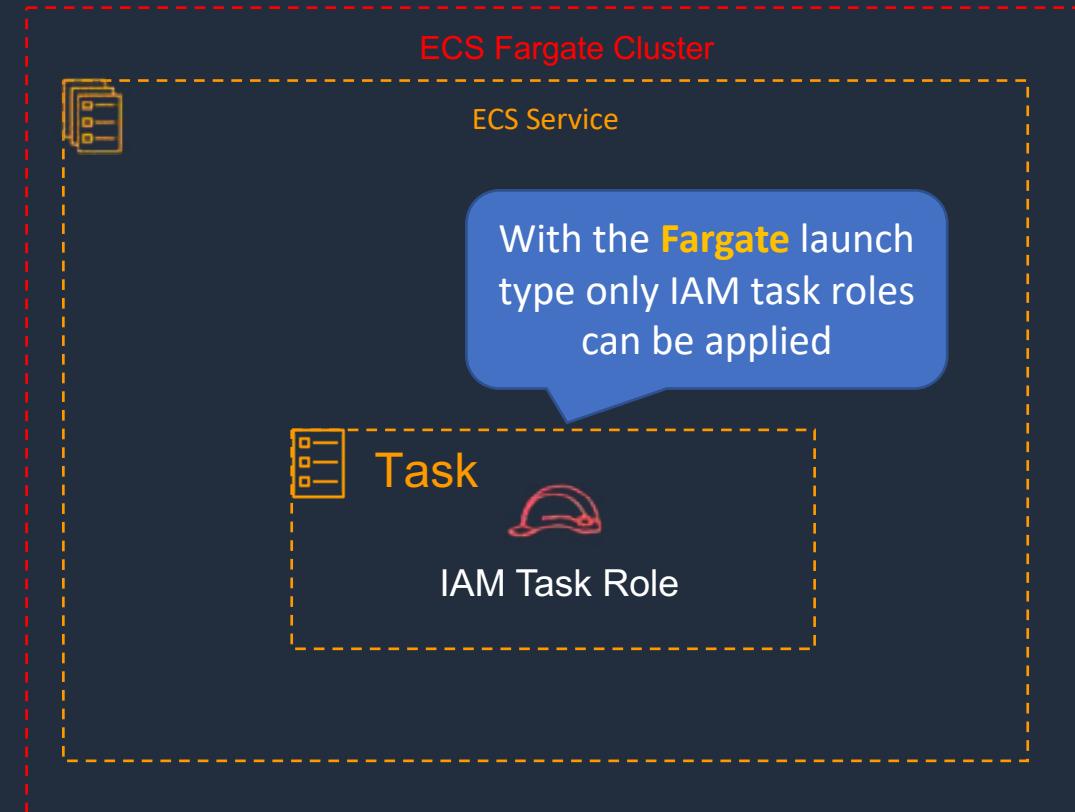
```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Action": [  
        "ec2:DescribeTags",  
        "ecs>CreateCluster",  
        "ecs>DeregisterContainerInstance",  
        "ecs>DiscoverPollEndpoint",  
        "ecs>Poll",  
        "ecs>RegisterContainerInstance",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



# ECS and IAM Roles

---

---



# Scaling Amazon ECS





# Auto Scaling for ECS

---

Two types of scaling:

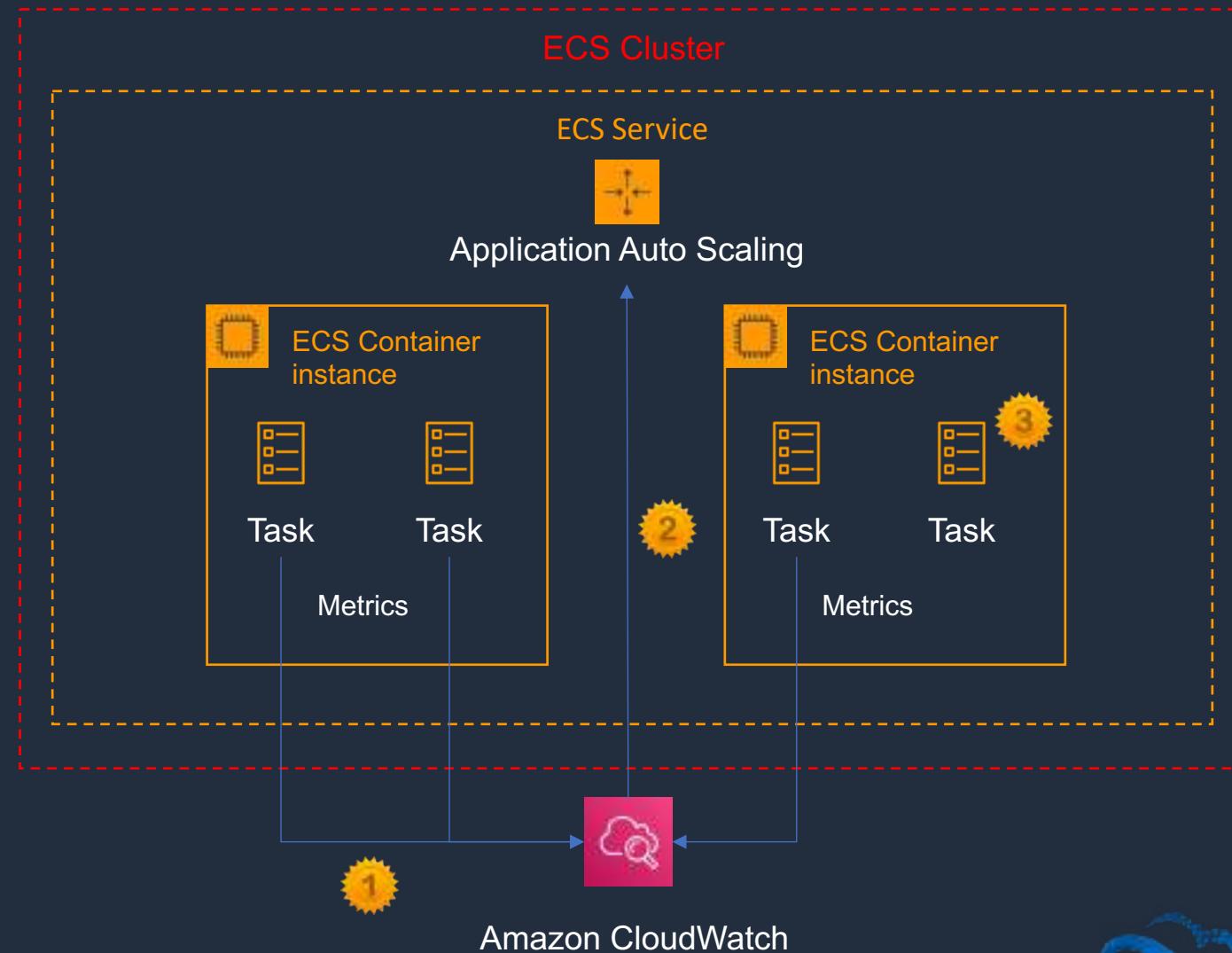
1. Service auto scaling
2. Cluster auto scaling

- **Service auto scaling** automatically adjusts the desired task count up or down using the Application Auto Scaling service
- **Service auto scaling** supports target tracking, step, and scheduled scaling policies
- **Cluster auto scaling** uses a Capacity Provider to scale the number of EC2 cluster instances using EC2 Auto Scaling



# Service Auto Scaling

- 1. Metric reports CPU > 80%
- 2. CloudWatch notifies Application Auto Scaling
- 3. ECS launches additional task





# Service Auto Scaling

---

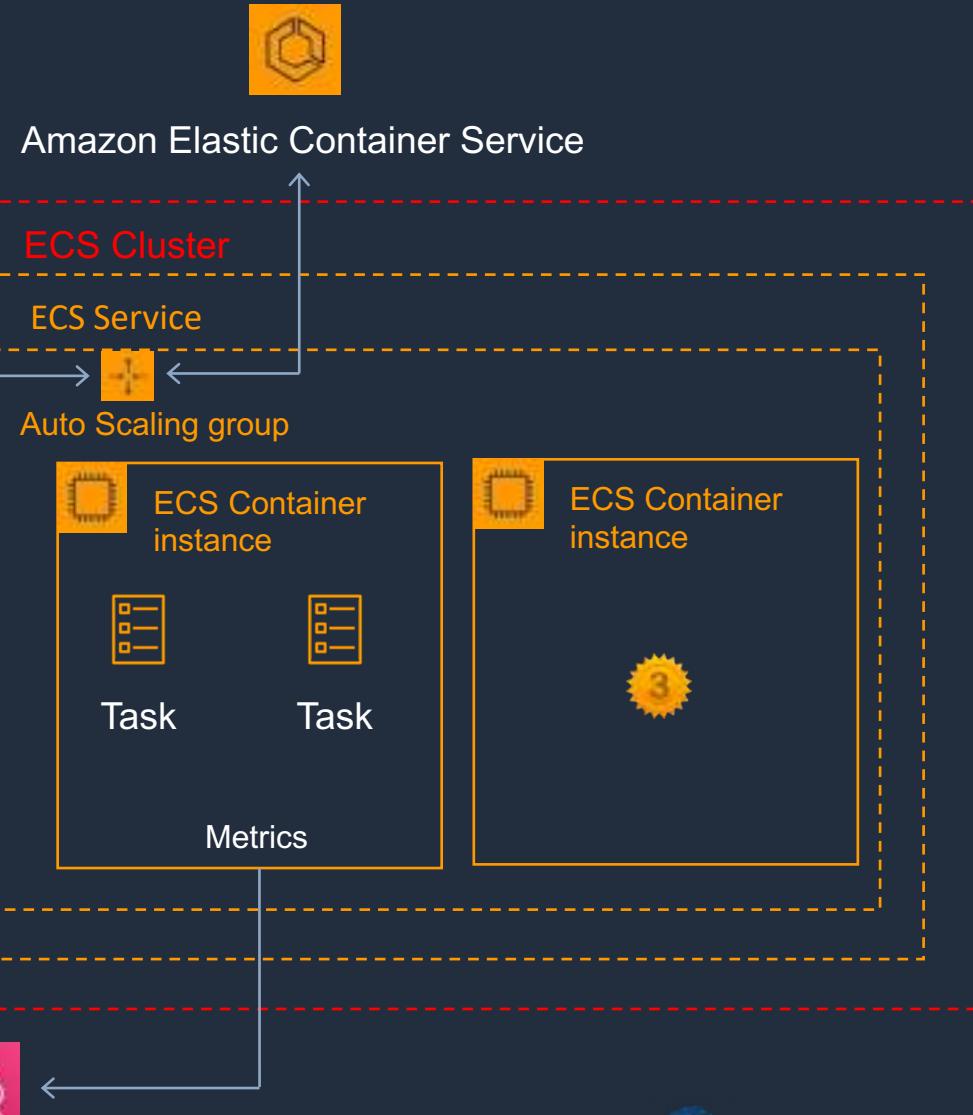
- Amazon ECS Service Auto Scaling supports the following types of scaling policies:
  - **Target Tracking Scaling Policies**—Increase or decrease the number of tasks that your service runs based on a target value for a specific CloudWatch metric
  - **Step Scaling Policies**—Increase or decrease the number of tasks that your service runs in response to CloudWatch alarms. Step scaling is based on a set of scaling adjustments, known as step adjustments, which vary based on the size of the alarm breach
  - **Scheduled Scaling**—Increase or decrease the number of tasks that your service runs based on the date and time



# Cluster Auto Scaling

- 1. Metric reports target capacity > 80%
- 2. CloudWatch notifies ASG
- 3. AWS launches additional container instance

ASG is linked to ECS using a **Capacity Provider**





# Cluster Auto Scaling

---

---

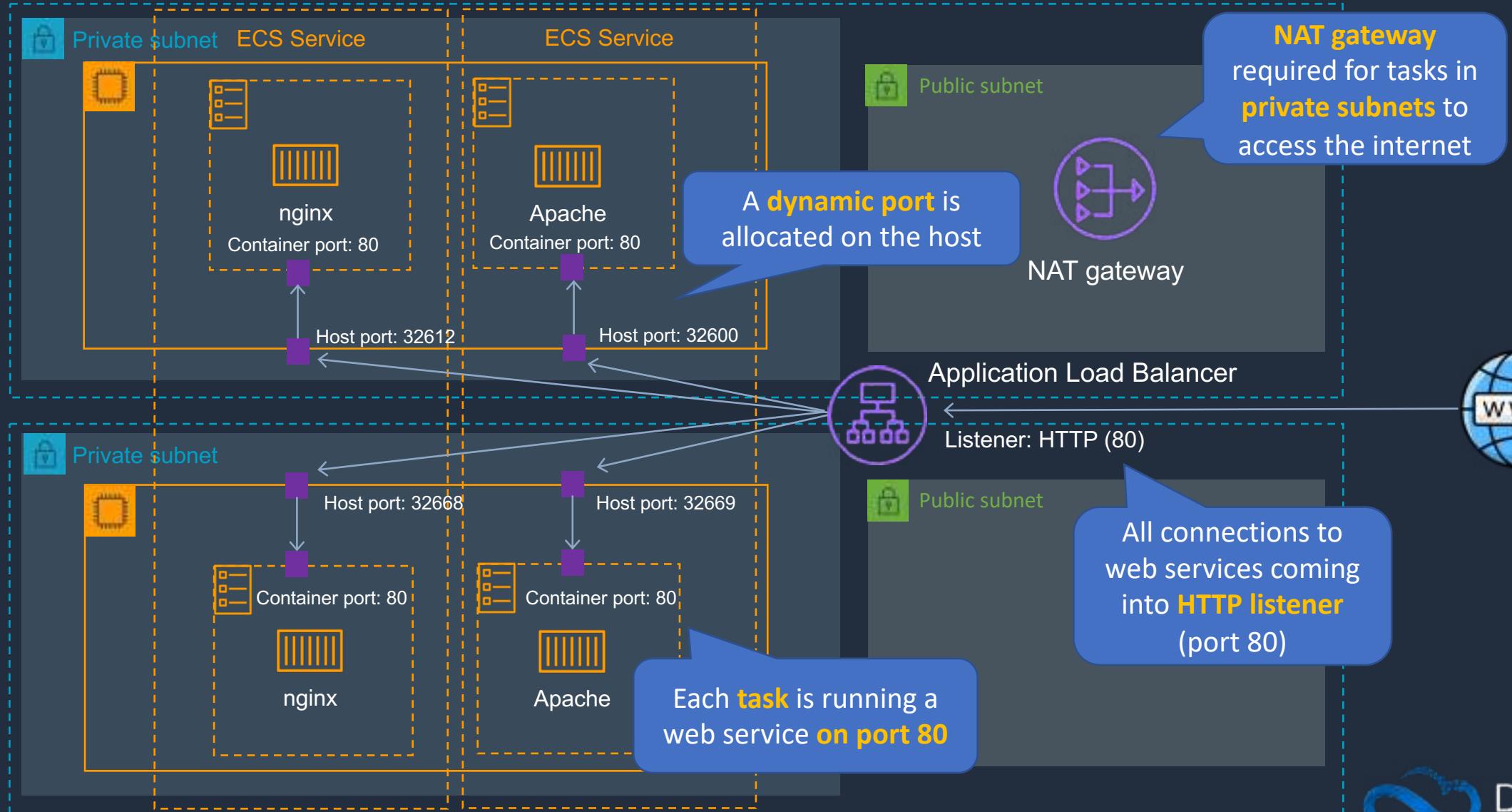
- Uses an ECS resource type called a **Capacity Provider**
- A Capacity Provider can be associated with an EC2 **Auto Scaling Group** (ASG)
- ASG can automatically scale using:
  - **Managed scaling** - with an automatically-created scaling policy on your ASG
  - **Managed instance termination protection** - which enables container-aware termination of instances in the ASG when scale-in happens

# Amazon ECS with ALB





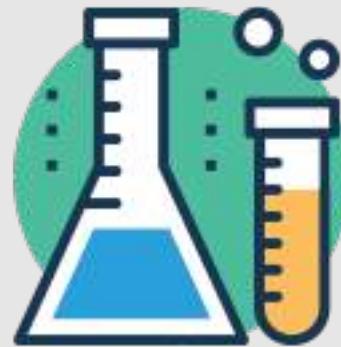
# Amazon ECS with ALB



# Deploy ECS Cluster (EC2 Launch Type)



# Use ALB with Fargate Cluster

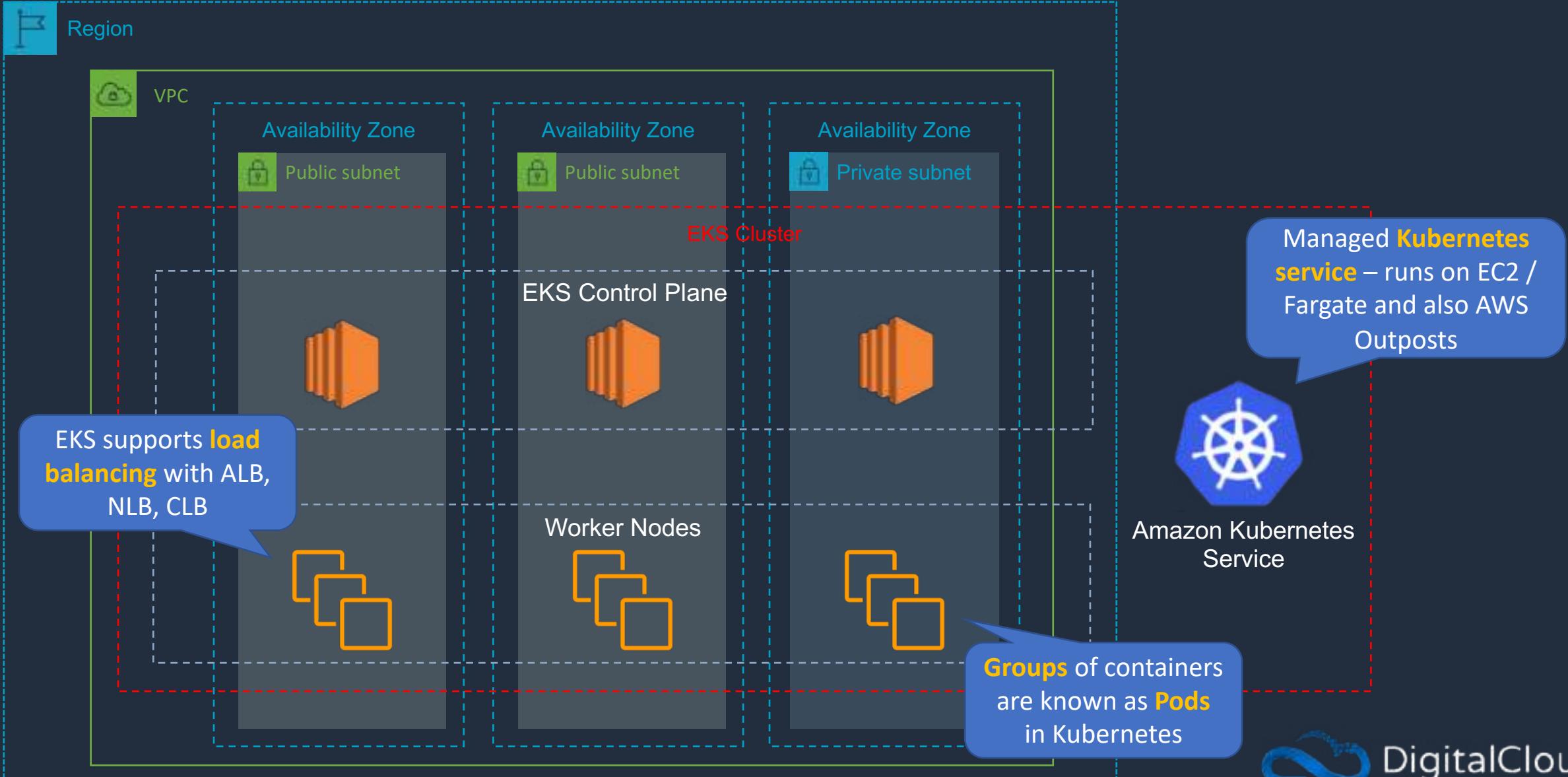


# Amazon Elastic Kubernetes Service (EKS)





# Amazon Elastic Kubernetes Service (EKS)





# Amazon EKS Use Cases

---

- Use when you need to **standardize** container orchestration across multiple environments using a **managed Kubernetes** implementation
- **Hybrid Deployment** - manage Kubernetes clusters and applications across hybrid environments (AWS + On-premises)
- **Batch Processing** - run sequential or parallel batch workloads on your EKS cluster using the Kubernetes Jobs API. Plan, schedule and execute batch workloads
- **Machine Learning** - use Kubeflow with EKS to model your machine learning workflows and efficiently run distributed training jobs using the latest EC2 GPU-powered instances, including Inferentia
- **Web Applications** - build web applications that automatically scale up and down and run in a highly available configuration across multiple Availability Zones

# Architecture Patterns – Amazon ECS





# Architecture Patterns – Amazon ECS

---

---

## Requirement

Application will be deployed on Amazon ECS and must scale based on memory

Application will run on Amazon ECS tasks across multiple hosts and needs access to an Amazon S3 bucket

Company requires standard Docker container automation and management service to be used across multiple environments

## Solution

Use service auto-scaling and use the memory utilization

Use a task execution IAM role to provide permissions to S3 bucket.

Deploy Amazon EKS



# Architecture Patterns – Amazon ECS

## Requirement

Company plans to deploy Docker containers on AWS at the lowest cost

Company plans to migrate Docker containers to AWS and does not want to manage operating systems

Multiple microservices applications running on Amazon ECS. Need to route based on information in the HTTP header

## Solution

Use Amazon ECS with a cluster of Spot instances and enable Spot instance draining

Migrate to Amazon ECS using the Fargate launch type

Deploy an Application Load Balancer in front of ECS and use query string parameter based routing



# Architecture Patterns – Amazon ECS

---

---

## Requirement

Containerized app runs on Amazon EKS. Need to collect and centrally view metrics and logs including EKS namespaces and EKS services

## Solution

Configure CloudWatch Container Insights and view data in CloudWatch console

# SECTION 11

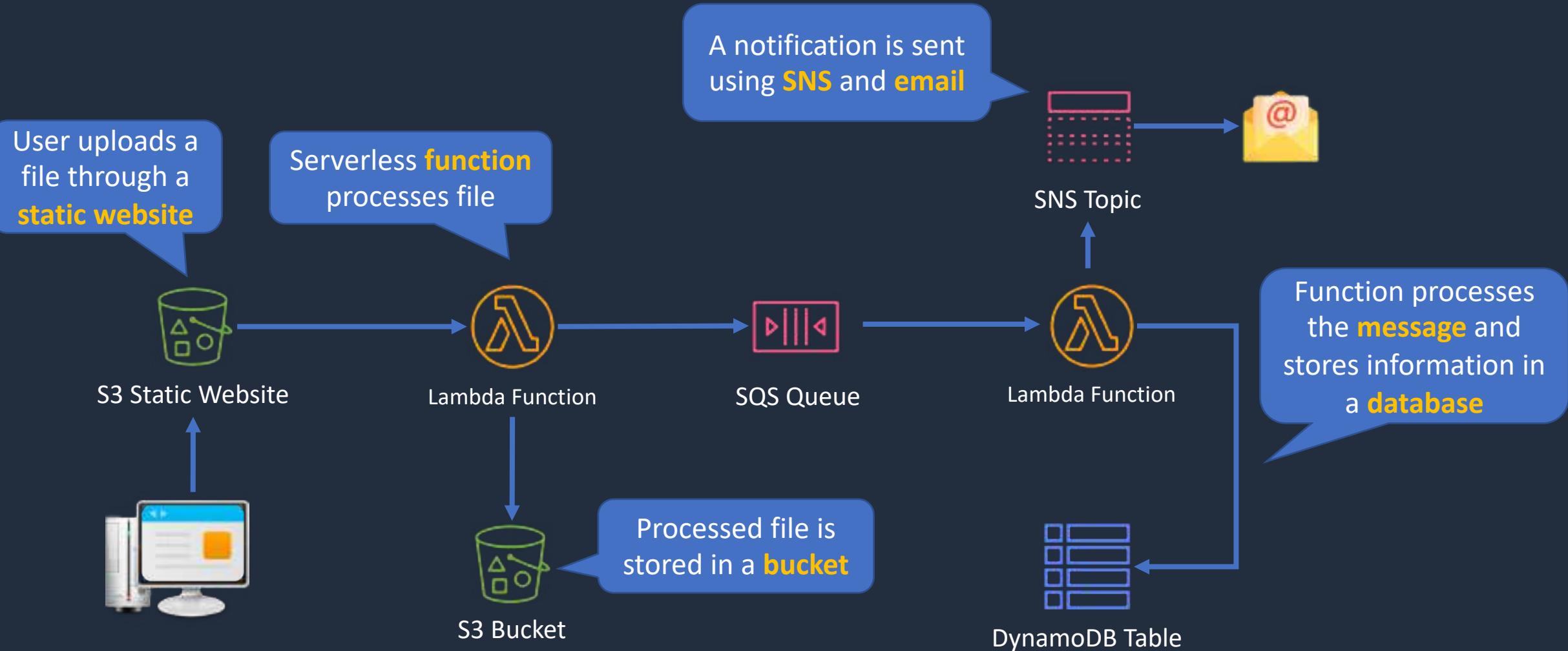
## Serverless Applications

# Serverless Services and Event-Driven Architecture





# Serverless Services and Event-Driven Architecture





# Serverless Services

---

---

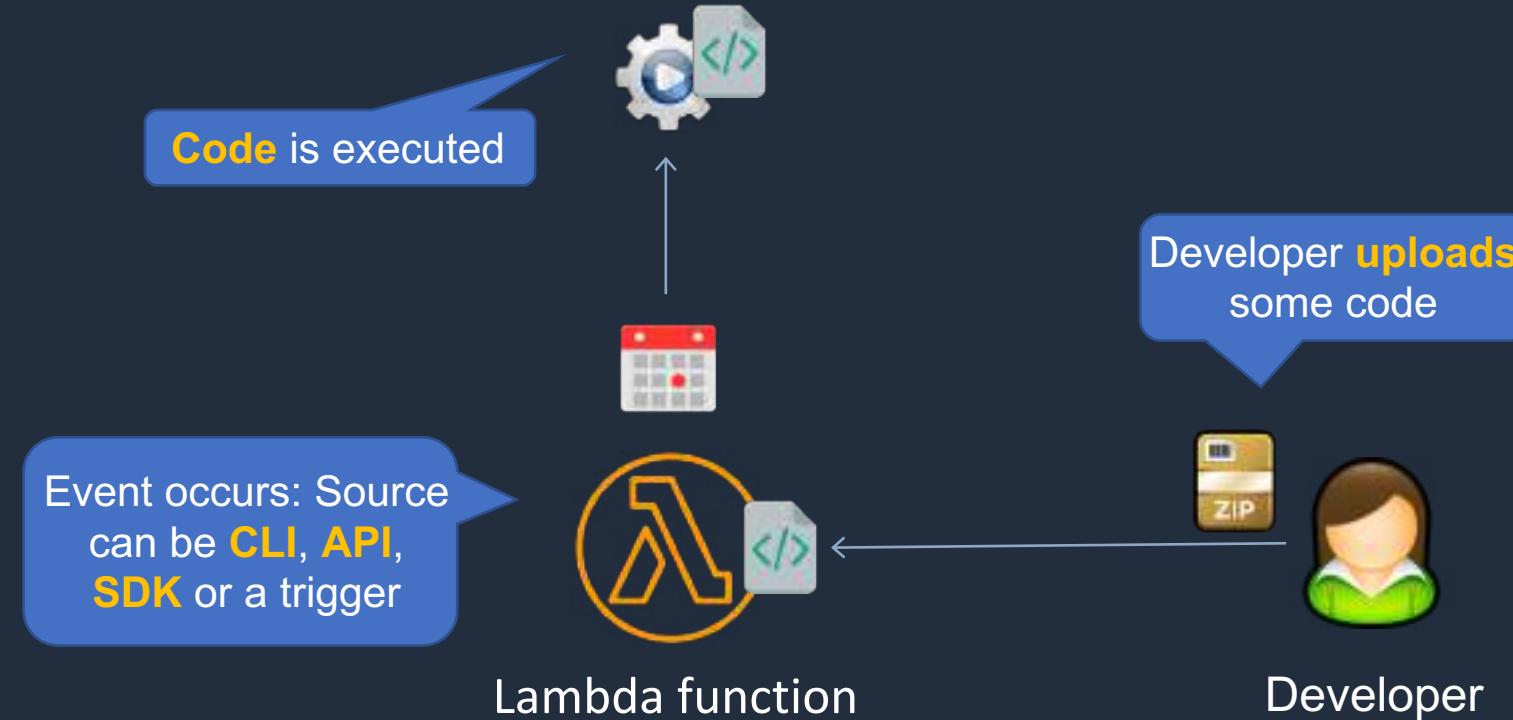
- With serverless there are **no instances** to manage
- You don't need to provision hardware
- There is no management of operating systems or software
- Capacity provisioning and patching is handled automatically
- Provides automatic scaling and high availability
- Can be very cheap!

# AWS Lambda





# AWS Lambda





# AWS Lambda

---

---

- AWS Lambda executes code only when needed and scales automatically
- You pay only for the compute time you consume (you pay nothing when your code is not running)
- Benefits of AWS Lambda:
  - No servers to manage
  - Continuous scaling
  - Millisecond billing
  - Integrates with almost all other AWS services



# AWS Lambda

---

- Primary use cases for AWS Lambda:
  - Data processing
  - Real-time file processing
  - Real-time stream processing
  - Build serverless backends for web, mobile, IOT, and 3rd party API requests



# Lambda Function Invocations

---

## Synchronous:

- CLI, SDK, API Gateway
- Result returned immediately
- Error handling happens client side (retries, exponential backoff etc.)

## Asynchronous:

- S3, SNS, CloudWatch Events etc.
- Lambda retries up to 3 times
- Processing must be idempotent (due to retries)

## Event source mapping:

- SQS, Kinesis Data Streams, DynamoDB Streams
- Lambda does the polling (polls the source)
- Records are processed in order (except for SQS standard)

SQS can also trigger  
Lambda

# Lambda Function Concurrency



## Burst **concurrency** quotas:

- 3000 – US West (Oregon), US East (N. Virginia), Europe (Ireland)
- 1000 – Asia Pacific (Tokyo), Europe (Frankfurt), US East (Ohio)
- 500 – Other Regions

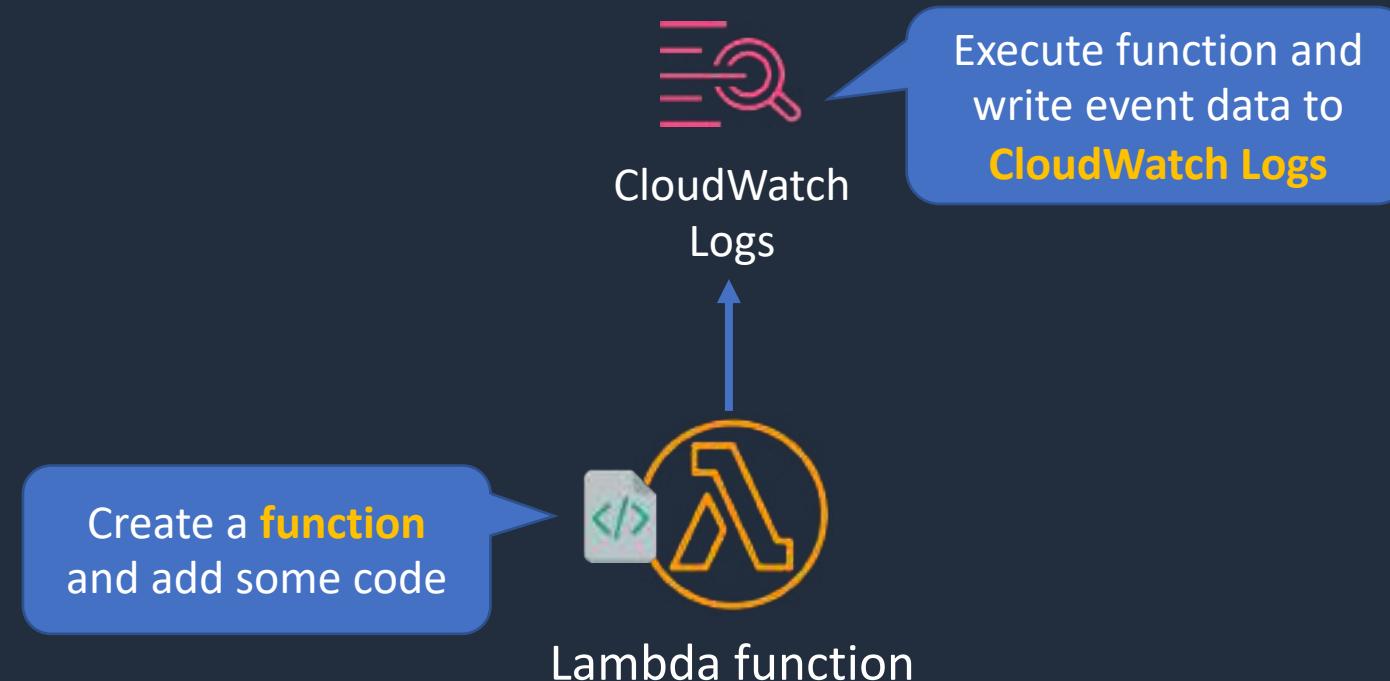
If the concurrency **limit** is **exceeded** throttling occurs with error "**Rate exceeded**" and 429 "**TooManyRequestsException**"

# Create Simple Lambda Function





# Create a Simple Lambda Function



# Application Integration Services Overview





# Application Integration Services Overview

Service	What it does	Example use cases
<b>Simple Queue Service</b>	<b>Messaging queue; store and forward patterns</b>	<b>Building distributed / decoupled applications</b>
<b>Simple Notification Service</b>	<b>Set up, operate, and send notifications from the cloud</b>	<b>Send email notification when CloudWatch alarm is triggered</b>
<b>Step Functions</b>	<b>Out-of-the-box coordination of AWS service components with visual workflow</b>	<b>Order processing workflow</b>
<b>Simple Workflow Service</b>	<b>Need to support external processes or specialized execution logic</b>	<b>Human-enabled workflows like an order fulfilment system or for procedural requests</b>  <b>Note:</b> AWS recommends that for new applications customers consider Step Functions instead of SWF
<b>Amazon MQ</b>	<b>Message broker service for Apache Active MQ and RabbitMQ</b>	<b>Need a message queue that supports industry standard APIs and protocols; migrate queues to AWS</b>
<b>Amazon Kinesis</b>	<b>Collect, process, and analyze streaming data.</b>	<b>Collect data from IoT devices for later processing</b>



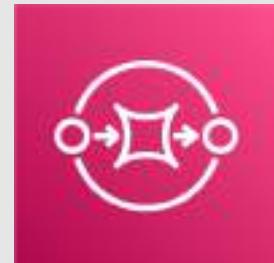
# Kinesis vs SQS vs SNS

---

---

Amazon Kinesis	Amazon SQS	Amazon SNS
<b>Consumers pull data</b>	<b>Consumers pull data</b>	<b>Push data to many subscribers</b>
<b>As many consumers as you need</b>	<b>Data is deleted after being consumed</b>	<b>Publisher / subscriber model</b>
<b>Routes related records to same record processor</b>	<b>Can have as many workers (consumers) as you need</b>	<b>Integrates with SQS for fan-out architecture pattern</b>
<b>Multiple applications can access stream concurrently</b>	<b>No ordering guarantee (except with FIFO queues)</b>	<b>Up to 10,000,000 subscribers</b>
<b>Ordering at the shard level</b>	<b>Provides messaging semantics</b>	<b>Up to 100,000 topics</b>
<b>Can consume records in correct order at later time</b>	<b>Individual message delay</b>	<b>Data is not persisted</b>
<b>Must provision throughput</b>	<b>Dynamically scales</b>	<b>No need to provision throughput</b>

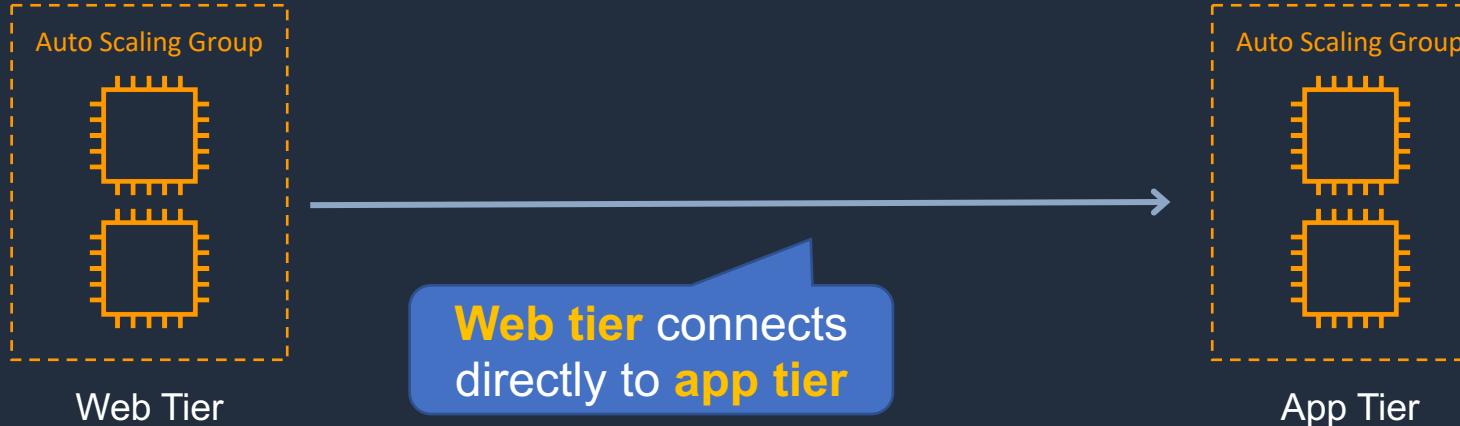
# Amazon SQS



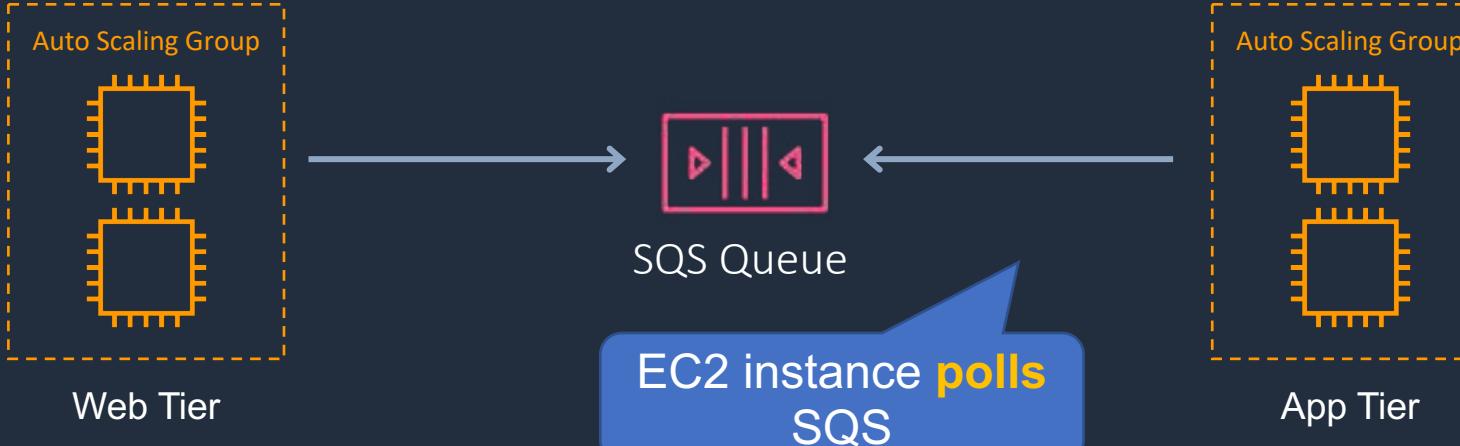


# Decoupling with SQS Queues

## Direct integration



## Decoupled integration





# SQS Queue Types

---

---



**Best-effort ordering**



**First-in, First-out  
Delivery**



# SQS Queue Types

Standard Queue	FIFO Queue
<p><b>Unlimited Throughput:</b> Standard queues support a nearly unlimited number of transactions per second (TPS) per API action.</p>	<p><b>High Throughput:</b> FIFO queues support up to 300 messages per second (300 send, receive, or delete operations per second). When you batch 10 messages per operation (maximum), FIFO queues can support up to 3,000 messages per second</p>
<p><b>Best-Effort Ordering:</b> Occasionally, messages might be delivered in an order different from which they were sent</p>	<p><b>First-In-First-out Delivery:</b> The order in which messages are sent and received is strictly preserved</p>
<p><b>At-Least-Once Delivery:</b> A message is delivered at least once, but occasionally more than one copy of a message is delivered</p>	<p><b>Exactly-Once Processing:</b> A message is delivered once and remains available until a consumer processes and deletes it. Duplicates are not introduced into the queue</p>



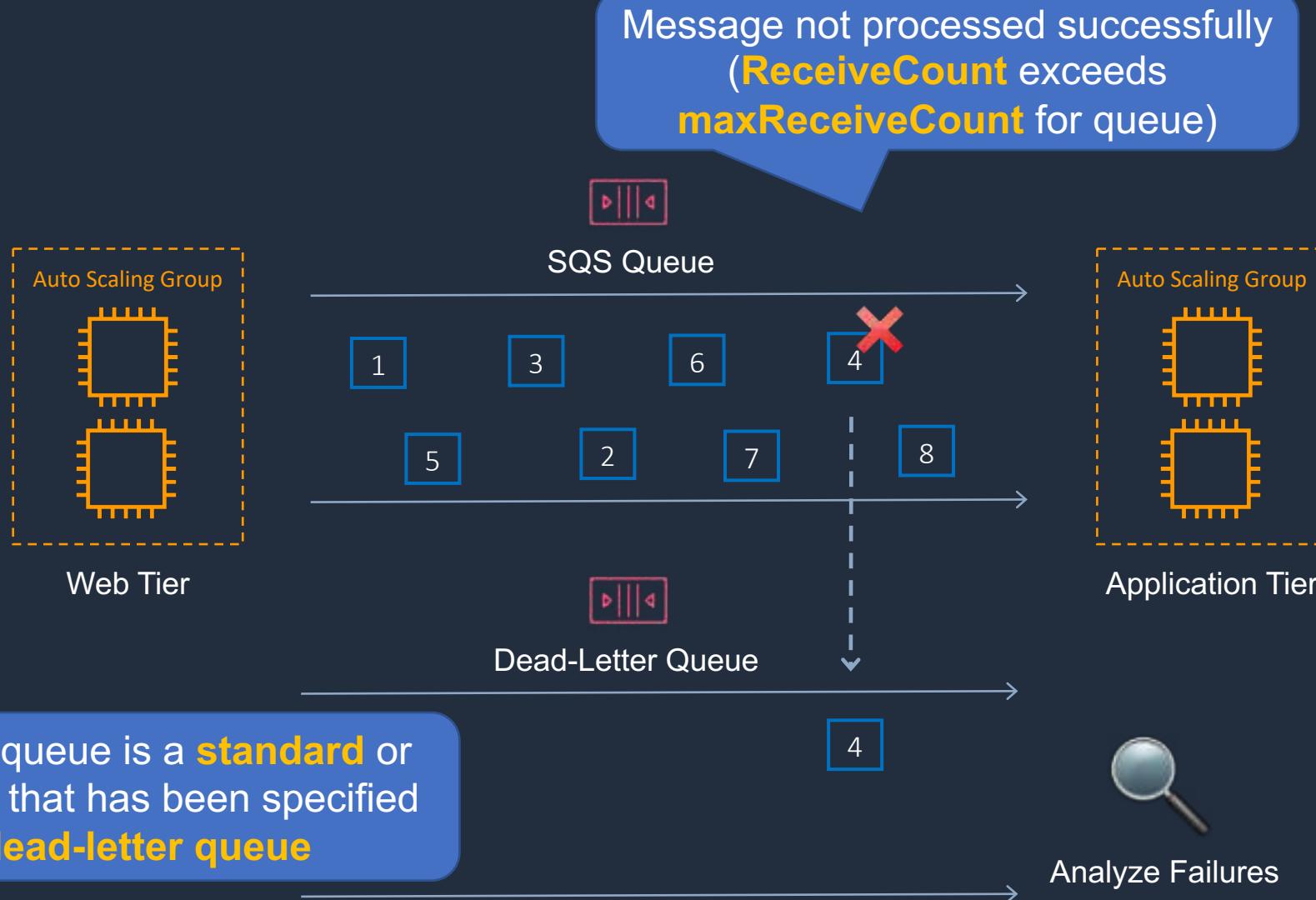
# SQS Queue Types

---

- FIFO queues require the **Message Group ID** and **Message Deduplication ID** parameters to be added to messages
- **Message Group ID:**
  - The tag that specifies that a message belongs to a specific message group Messages that belong to the same message group are guaranteed to be processed in a FIFO manner
- **Message Deduplication ID:**
  - The token used for deduplication of messages within the deduplication interval



# SQS – Dead Letter Queue

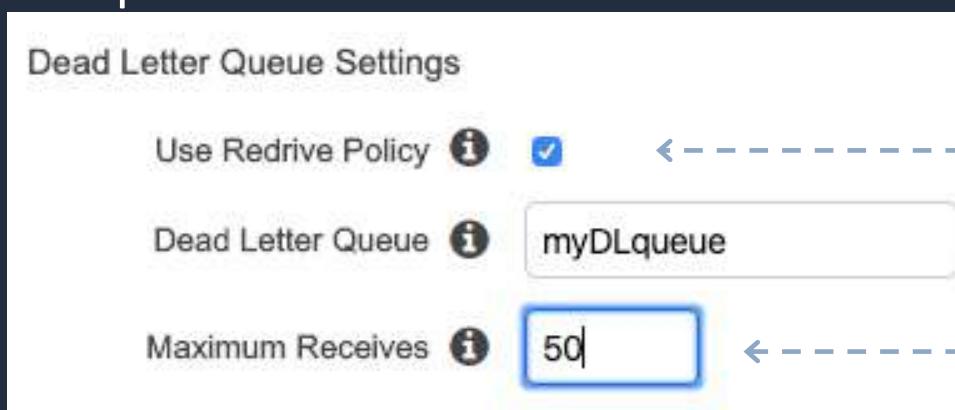




# SQS – Dead Letter Queue

---

- The main task of a dead-letter queue is handling message failure
- A dead-letter queue lets you set aside and isolate messages that can't be processed correctly to determine why their processing didn't succeed
- It is not a queue type, it is a **standard** or **FIFO** queue that has been specified as a dead-letter queue in the configuration of **another** standard or FIFO queue



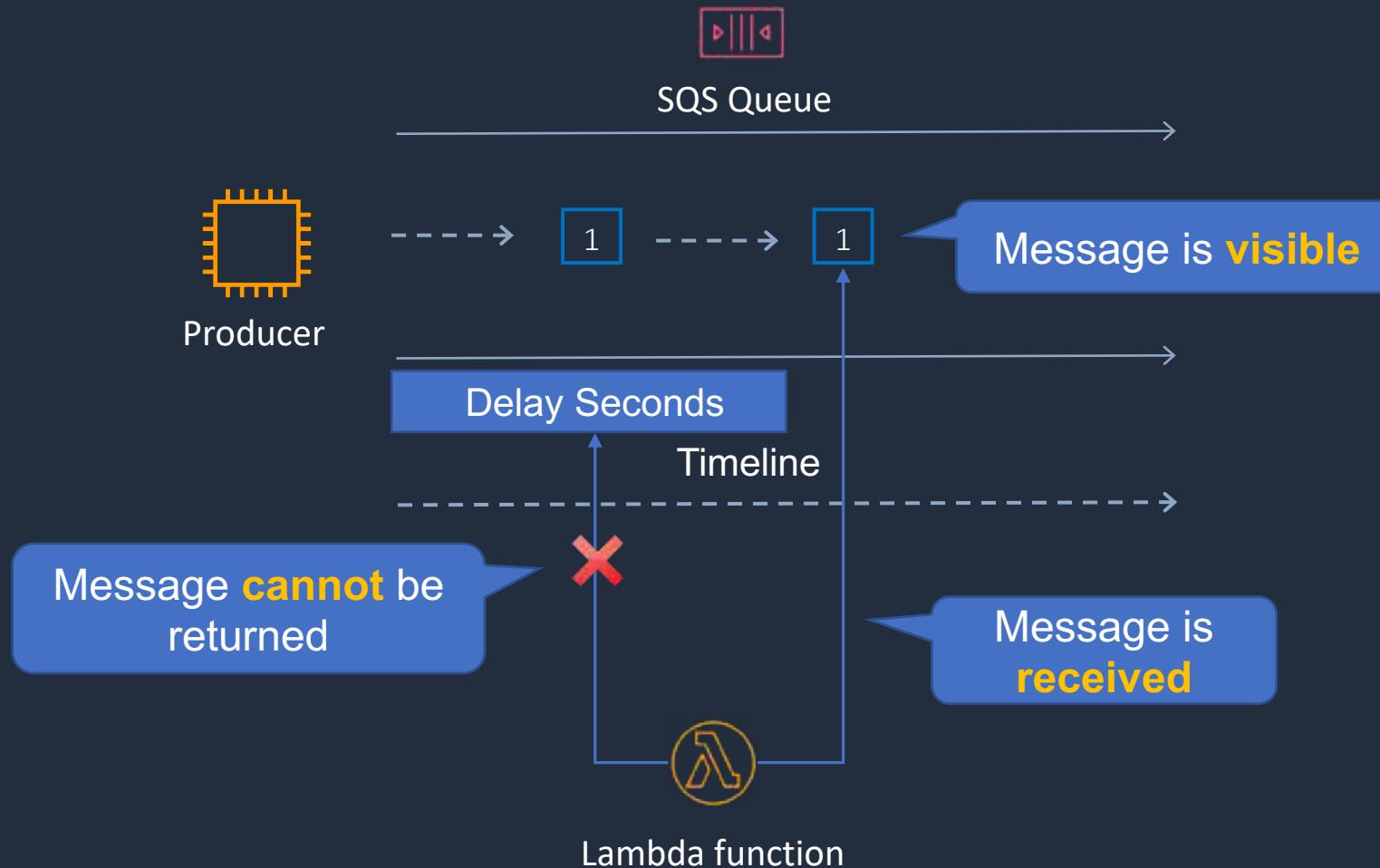
Enable Redrive Policy

Specify the queue to use as a dead-letter queue

Specify the maximum receives before a message is sent to the dead-letter queue



# SQS – Delay Queue



Delivery delay [Info](#)

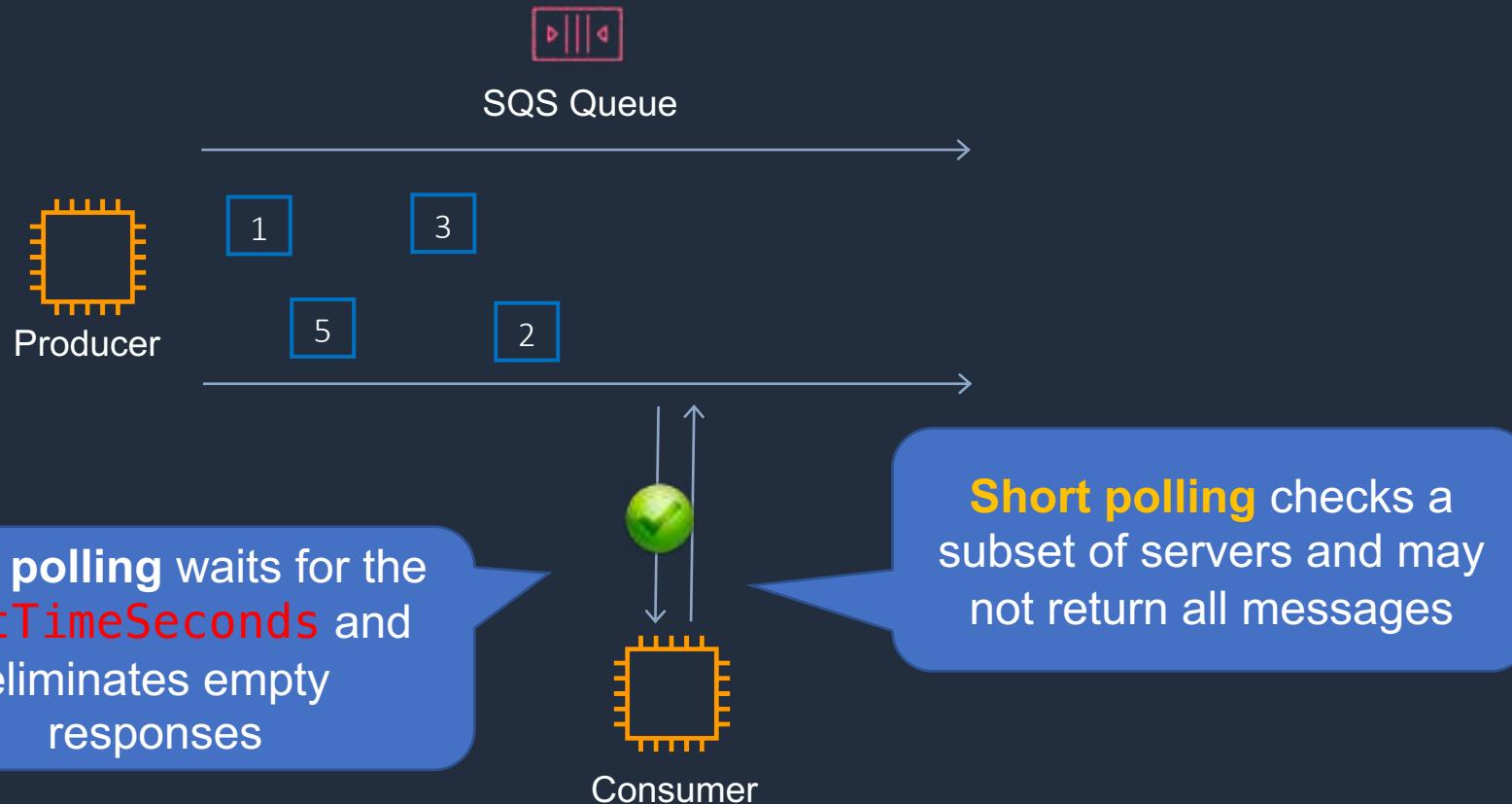
Default 0s, max 15mins

0  ▾



# SQS Long Polling vs Short Polling

---

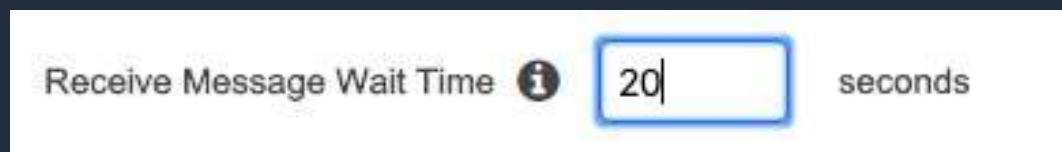




# SQS Long Polling vs Short Polling

---

- SQS **Long polling** is a way to retrieve messages from SQS queues – waits for messages to arrive
- SQS **Short polling** returns immediately (even if the message queue is empty)
- SQS Long polling can lower costs
- SQS Long polling can be enabled at the queue level or at the API level using **WaitTimeSeconds**
- SQS Long polling is in effect when the Receive Message Wait Time is a value greater than 0 seconds and up to 20 seconds



The maximum amount of time that a long polling receive call will wait for a message to become available before returning an empty response.

# Amazon SNS

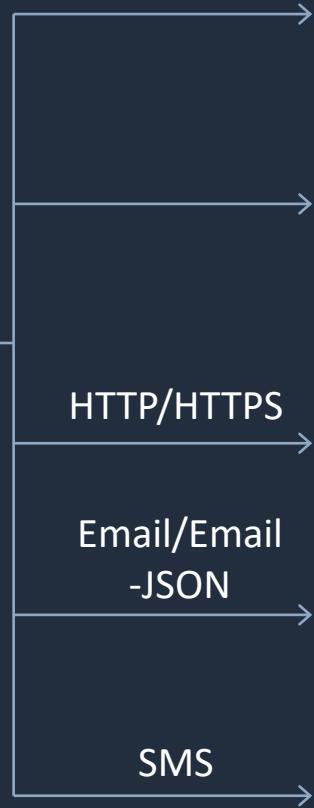


# Amazon SNS Notifications



Event **producer** sends one message to one SNS **topic**

Transport Protocols



Subscribers



Lambda



Amazon Simple Queue Service



Web Application



Email



Text

Many subscribers listen for notifications

Each **subscriber** will get all the messages



# Amazon SNS

---

---

- Amazon SNS is a highly available, durable, secure, fully managed pub/sub messaging service
- Amazon SNS provides topics for high-throughput, push-based, many-to-many messaging
- Publisher systems can fan out messages to a large number of subscriber endpoints:
- Endpoints include:
  - Amazon SQS queues
  - AWS Lambda functions
  - HTTP/S webhooks
  - Mobile push
  - SMS
  - Email



# Amazon SNS

---

---

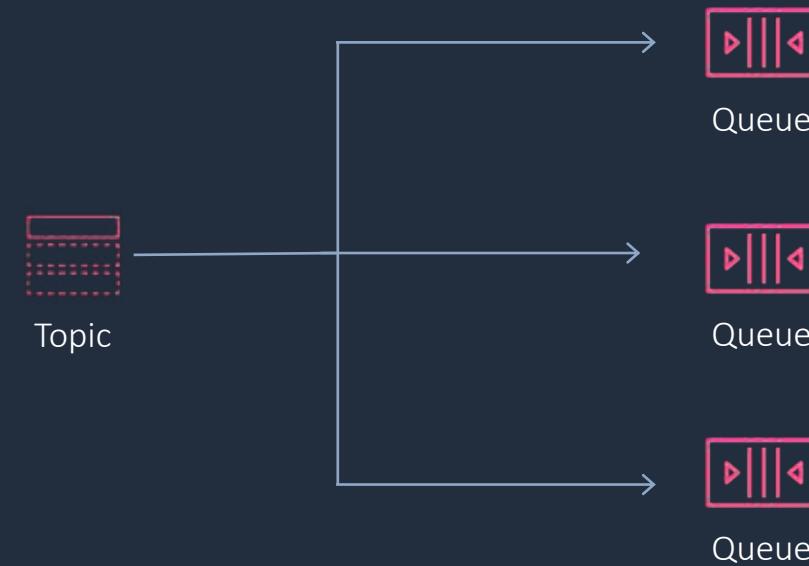
- Multiple recipients can be grouped using Topics
- A topic is an “access point” for allowing recipients to dynamically subscribe for identical copies of the same notification
- One topic can support deliveries to multiple endpoint types
- Simple APIs and easy integration with applications
- Flexible message delivery over multiple transport protocols



# Amazon SNS + Amazon SQS Fan-Out

---

- You can subscribe one or more Amazon SQS queues to an Amazon SNS topic
- Amazon SQS manages the subscription and any necessary permissions
- When you publish a message to a topic, Amazon SNS sends the message to every subscribed queue



# Simple Event-Driven App



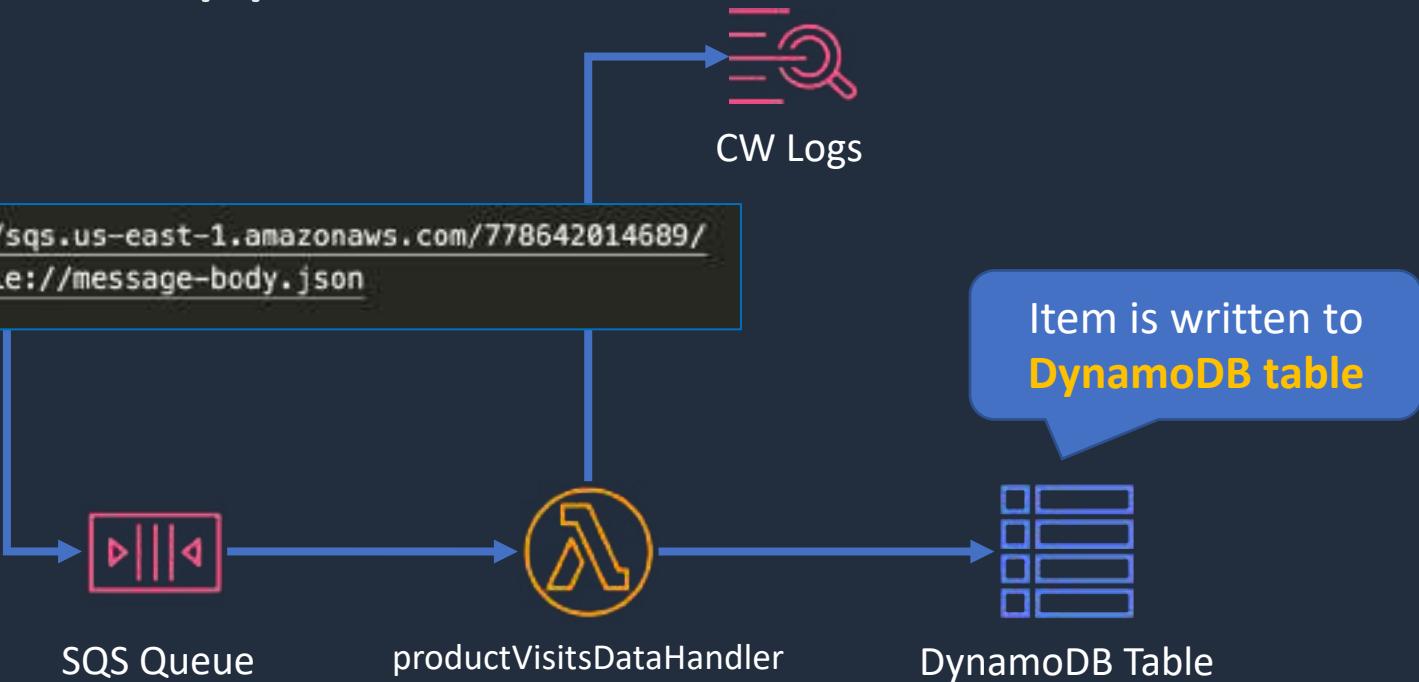


# Simple Event-Driven App



```
aws sqs send-message --queue-url https://sqs.us-east-1.amazonaws.com/778642014689/  
ProductVisitsDataQueue --message-body file://message-body.json
```

Manually add message to queue with **AWS CLI**



JSON

```
{  
    "ProductId": "c96b49bb-c378-4a15-b2e3-842a9850b23d",  
    "ProductName": "Gloves",  
    "Category": "Accessories",  
    "PricePerUnit": "10",  
    "customerId": "be44af0a-74f9-438e-a3ac-e3e21d84259f",  
    "customerName": "John Doe",  
    "TimeOfVisit": "2021-02-21T16:23:42.389Z"  
}
```

SQS triggers Lambda

This is the **message body**

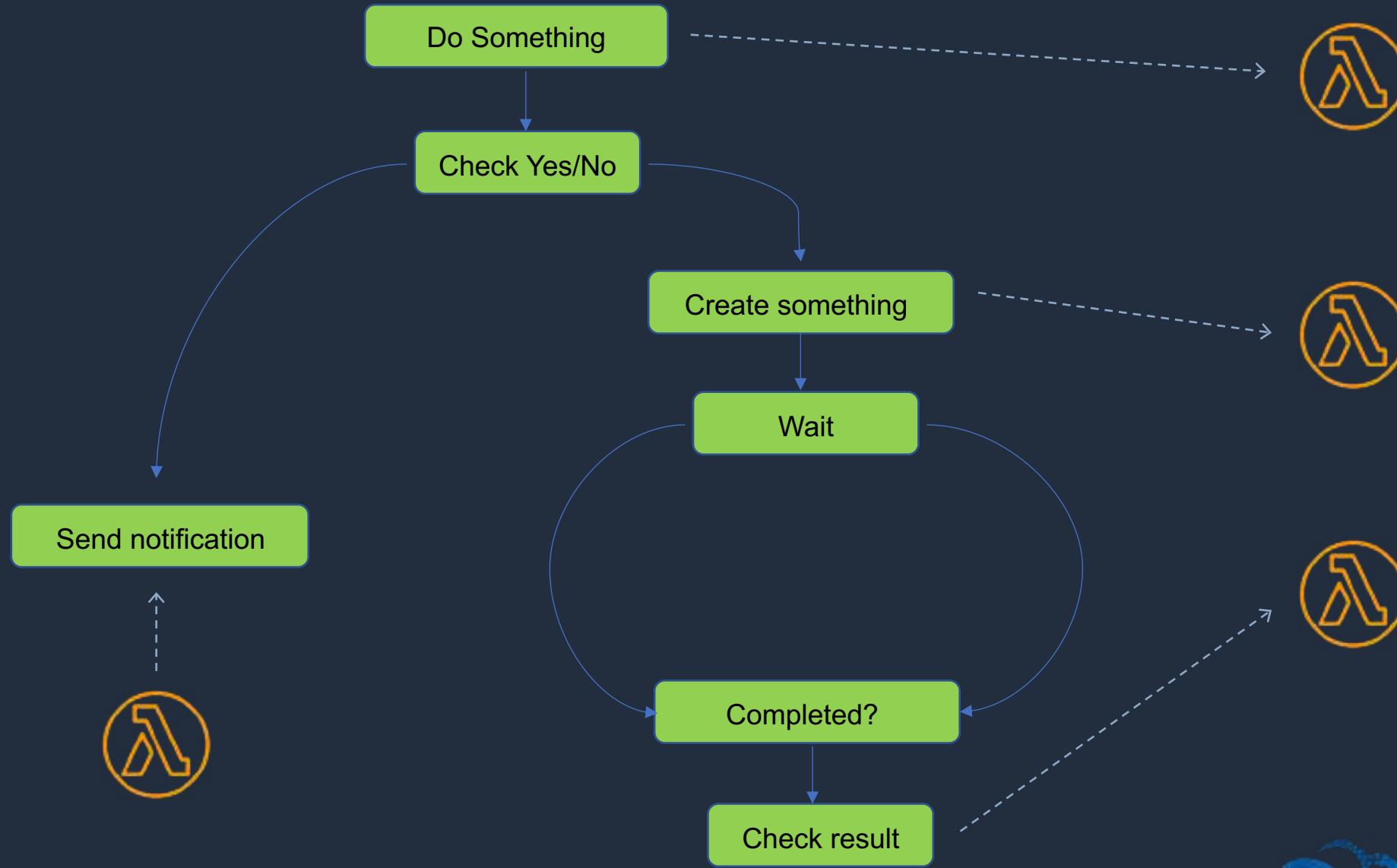
# AWS Step Functions





# AWS Step Functions

---





# AWS Step Functions

---

---

- AWS Step Functions is used to build distributed applications as a series of steps in a visual workflow.
- You can quickly build and run state machines to execute the steps of your application

## How it works:

1. Define the steps of your workflow in the **JSON-based Amazon States Language**.  
The visual console automatically graphs each step in the order of execution
2. Start an execution to visualize and verify the steps of your application are operating as intended. The console highlights the real-time status of each step and provides a detailed history of every execution
3. AWS Step Functions **operates and scales** the steps of your **application** and **underlying compute** for you to help ensure your application executes reliably under increasing demand

# Create a State Machine



# Amazon EventBridge

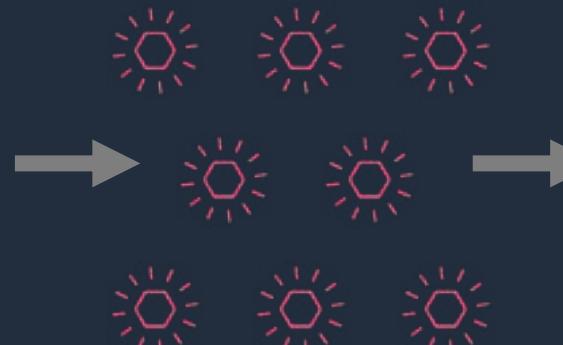
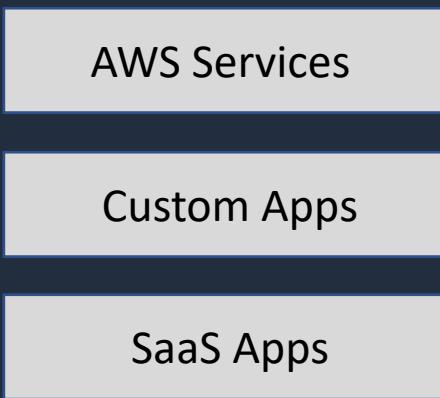




# Amazon EventBridge

EventBridge used to be known as **CloudWatch Events**

## Event Sources



## Events

EventBridge  
event bus

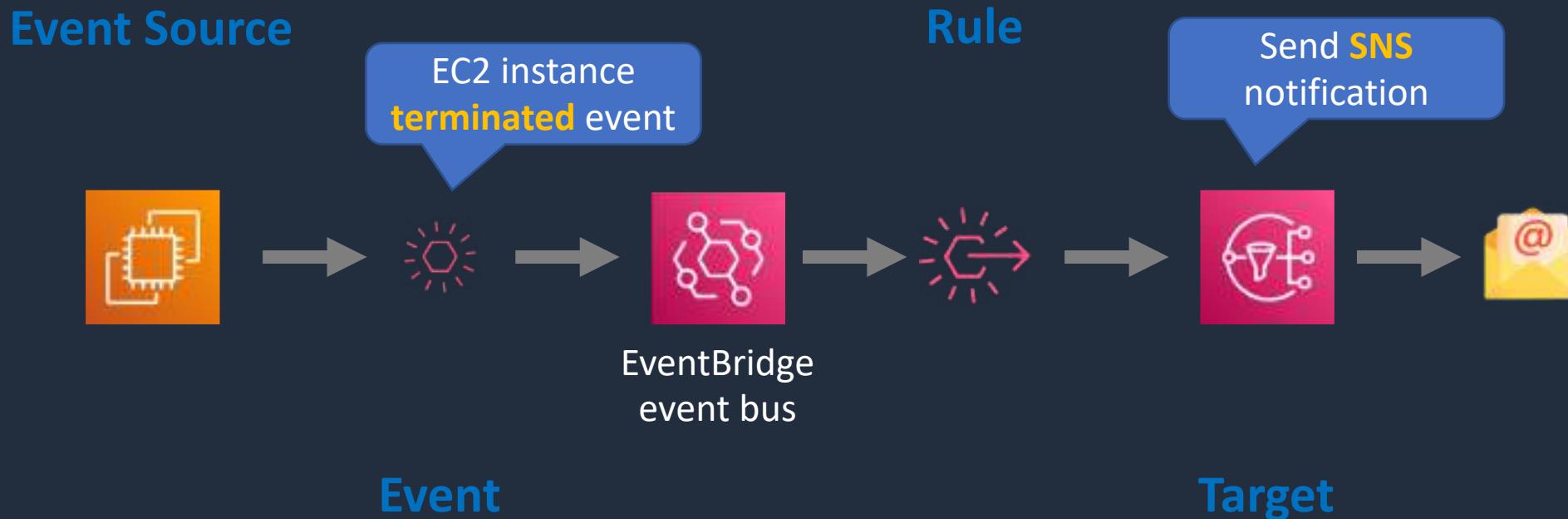
## Rules



## Targets



# Amazon EventBridge Example 1





# Amazon EventBridge Example 1

Event matching pattern  
You can use pre-defined pattern provided by a service or create a custom pattern.

Pre-defined pattern by service  
 Custom pattern

Service provider  
AWS services or custom/partner services

AWS

Service name  
The name of partner service selected as the event source

EC2

Event type  
The type of events as the source of the matching pattern

EC2 Instance State-change Notification

Any state  
 Specific state(s)

terminated X

Any instance  
 Specific Instance Id(s)

I-1234567890abcdef0

Remove

Add

Event pattern

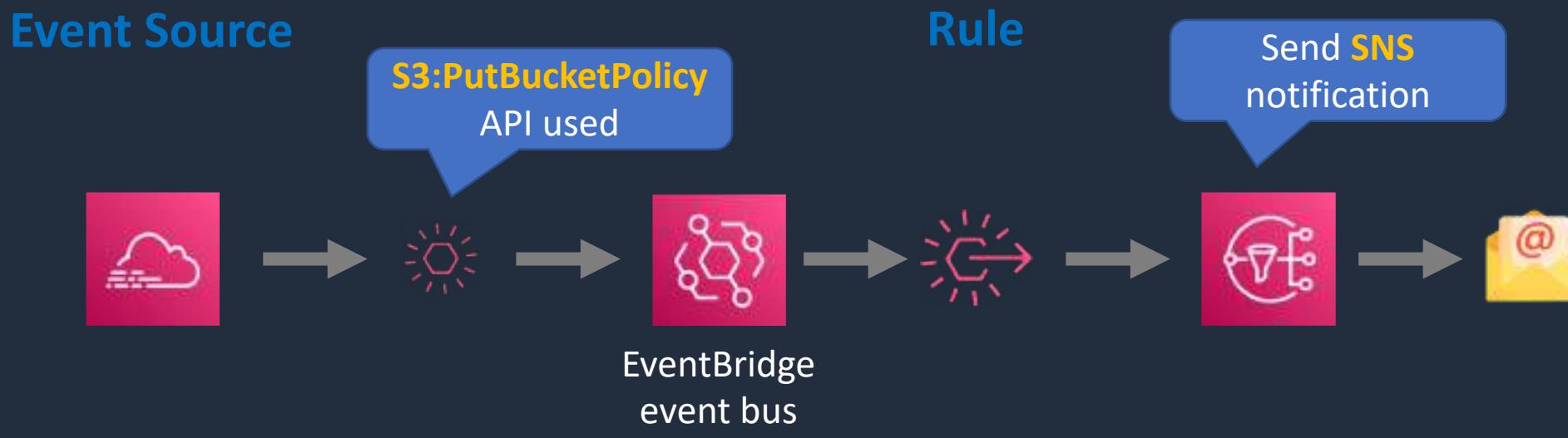
Copy Edit

```
1 {
2   "source": ["aws.ec2"],
3   "detail-type": ["EC2 Instance State-change Notificati
4   "detail": [
5     "state": ["terminated"],
6     "instance-id": ["i-1234567890abcdef0"]
7   ]
8 }
```

```
{
  "version": "0",
  "id": "6a7e8feb-b491-4cf7-a9f1-bf3703467718",
  "detail-type": "EC2 Instance State-change Notification",
  "source": "aws.ec2",
  "account": "111122223333",
  "time": "2017-12-22T18:43:48Z",
  "region": "us-west-1",
  "resources": [
    "arn:aws:ec2:us-west-1:123456789012:instance/i-1234567890abcdef0"
  ],
  "detail": {
    "instance-id": "i-1234567890abcdef0",
    "state": "terminated"
  }
}
```



# Amazon EventBridge Example 2



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

# Create Event Bus and Rule





# Amazon EventBridge Example 1

Event matching pattern  
You can use pre-defined pattern provided by a service or create a custom pattern.

Pre-defined pattern by service  
 Custom pattern

Service provider  
AWS services or custom/partner services

AWS

Service name  
The name of partner service selected as the event source

EC2

Event type  
The type of events as the source of the matching pattern

EC2 Instance State-change Notification

Any state  
 Specific state(s)

terminated X

Any instance  
 Specific Instance Id(s)

I-1234567890abcdef0

Remove

Add

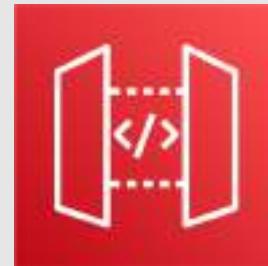
Event pattern

Copy Edit

```
1 {
2   "source": ["aws.ec2"],
3   "detail-type": ["EC2 Instance State-change Notificati
4   "detail": [
5     "state": ["terminated"],
6     "instance-id": ["i-1234567890abcdef0"]
7   ]
8 }
```

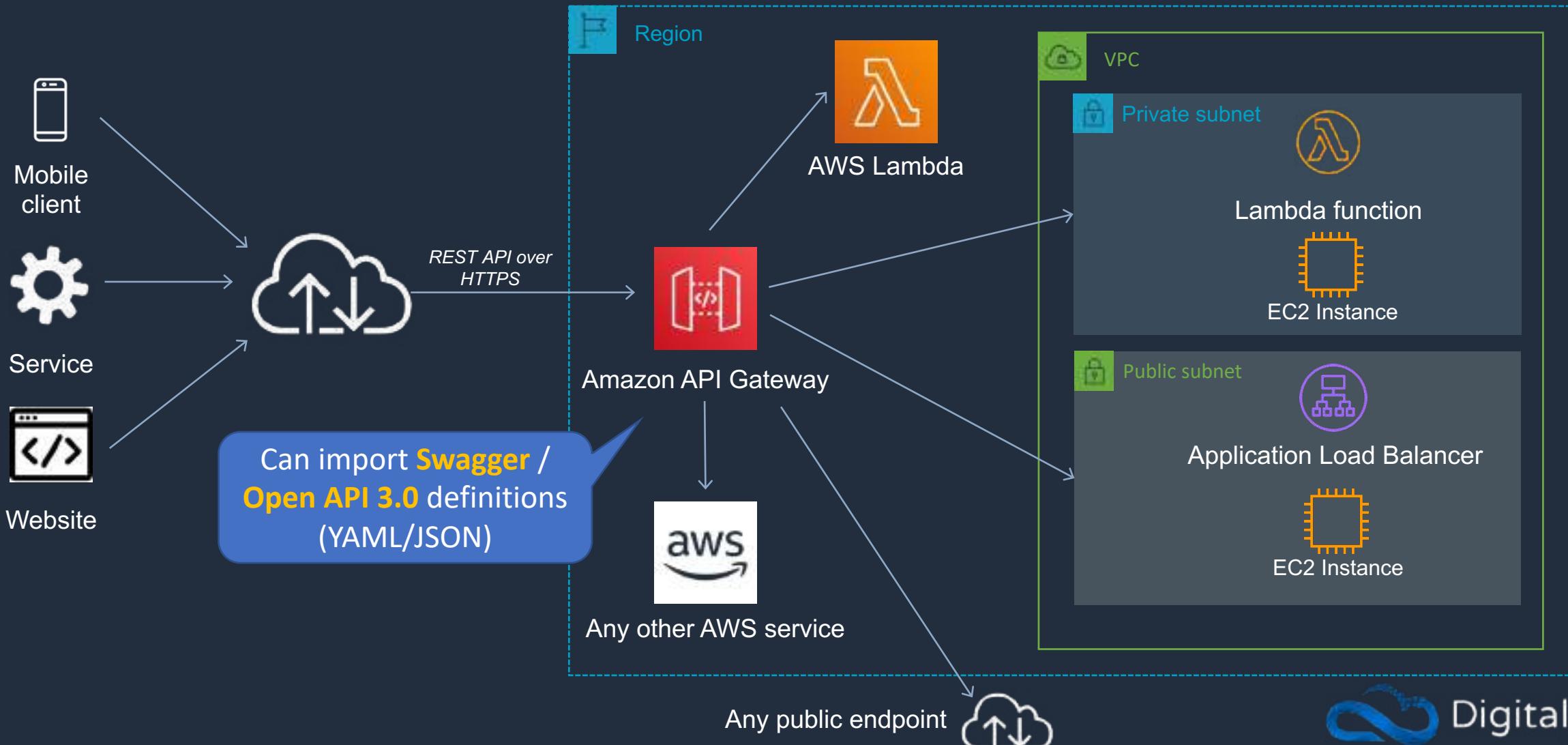
```
{
  "version": "0",
  "id": "6a7e8feb-b491-4cf7-a9f1-bf3703467718",
  "detail-type": "EC2 Instance State-change Notification",
  "source": "aws.ec2",
  "account": "111122223333",
  "time": "2017-12-22T18:43:48Z",
  "region": "us-west-1",
  "resources": [
    "arn:aws:ec2:us-west-1:123456789012:instance/i-1234567890abcdef0"
  ],
  "detail": {
    "instance-id": "i-1234567890abcdef0",
    "state": "terminated"
  }
}
```

# Amazon API Gateway





# Amazon API Gateway Overview





# Amazon API Gateway Deployment Types

## Edge-optimized endpoint



## Regional endpoint



## Private endpoint



### Key benefits:

- Reduced latency for requests from around the world

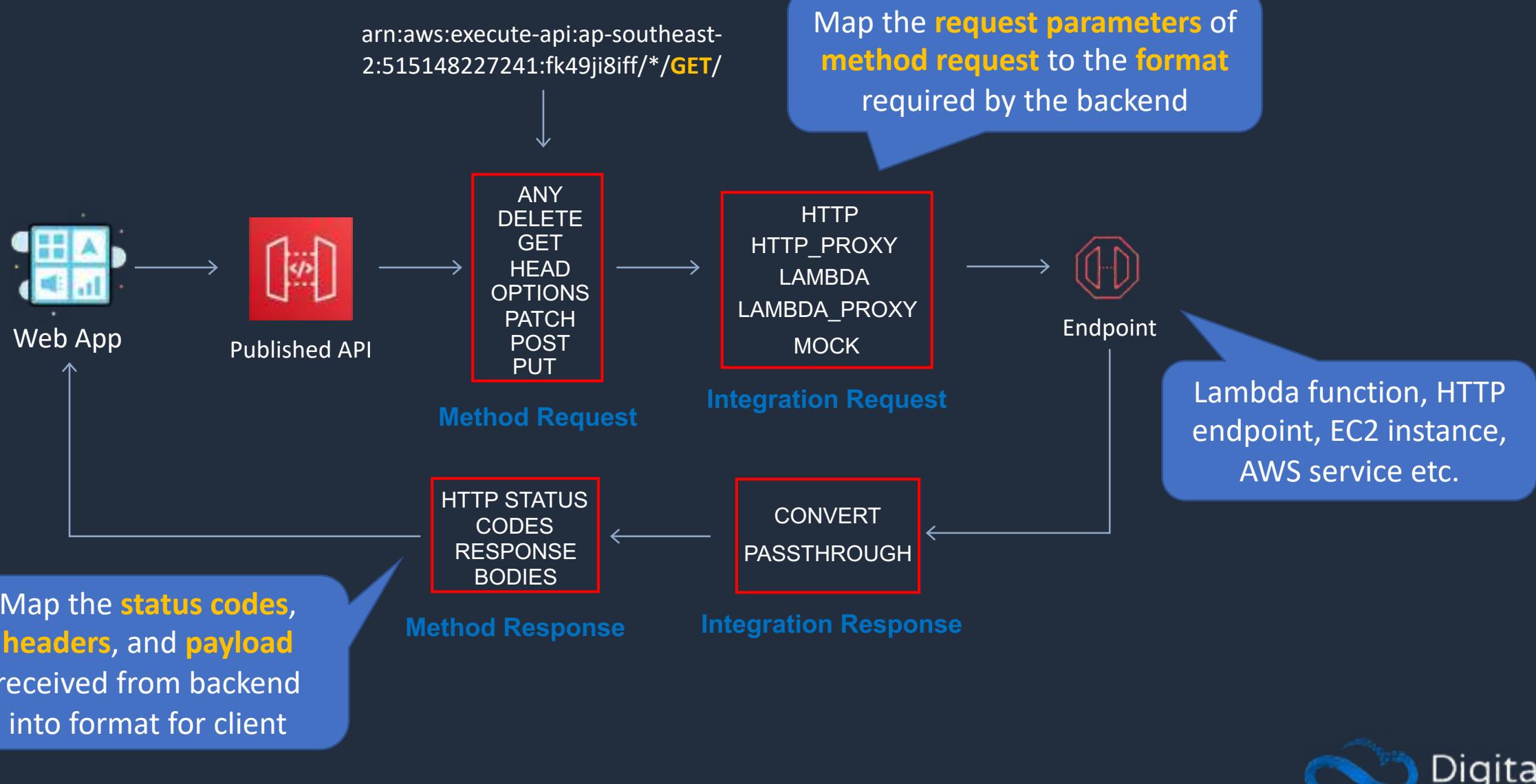
### Key benefits:

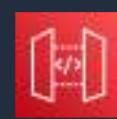
- Reduced latency for requests that originate in the same region
- Can also configure your own CDN and protect with WAF

### Key benefits:

- Securely expose your REST APIs only to other services within your VPC or connect via Direct Connect

# Structure of a REST API





# API Gateway Integrations

---

For a **Lambda function** you can have:

- Lambda proxy integration
- Lambda custom integration

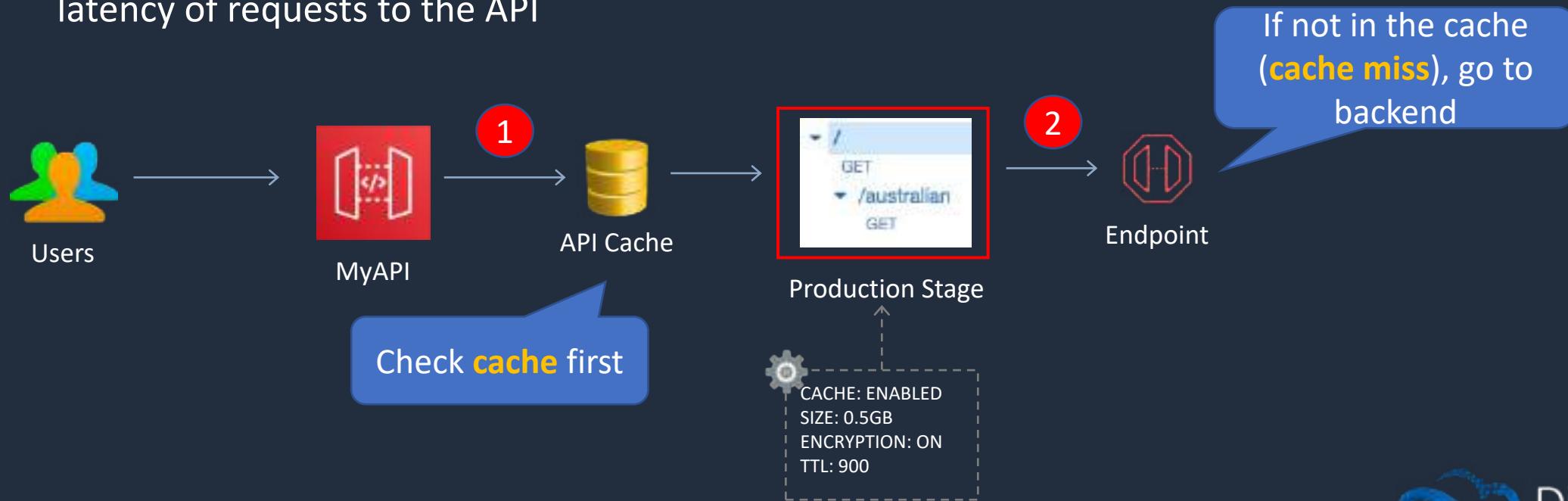
For an **HTTP endpoint** you can have:

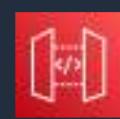
- HTTP proxy integration
- HTTP custom integration
- For an **AWS service action** you have the AWS integration of the non-proxy type only



# API Gateway - Caching

- You can add caching to API calls by provisioning an Amazon API Gateway cache and specifying its size in gigabytes
- Caching allows you to cache the endpoint's response
- Caching can reduce number of calls to the backend and improve latency of requests to the API

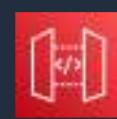




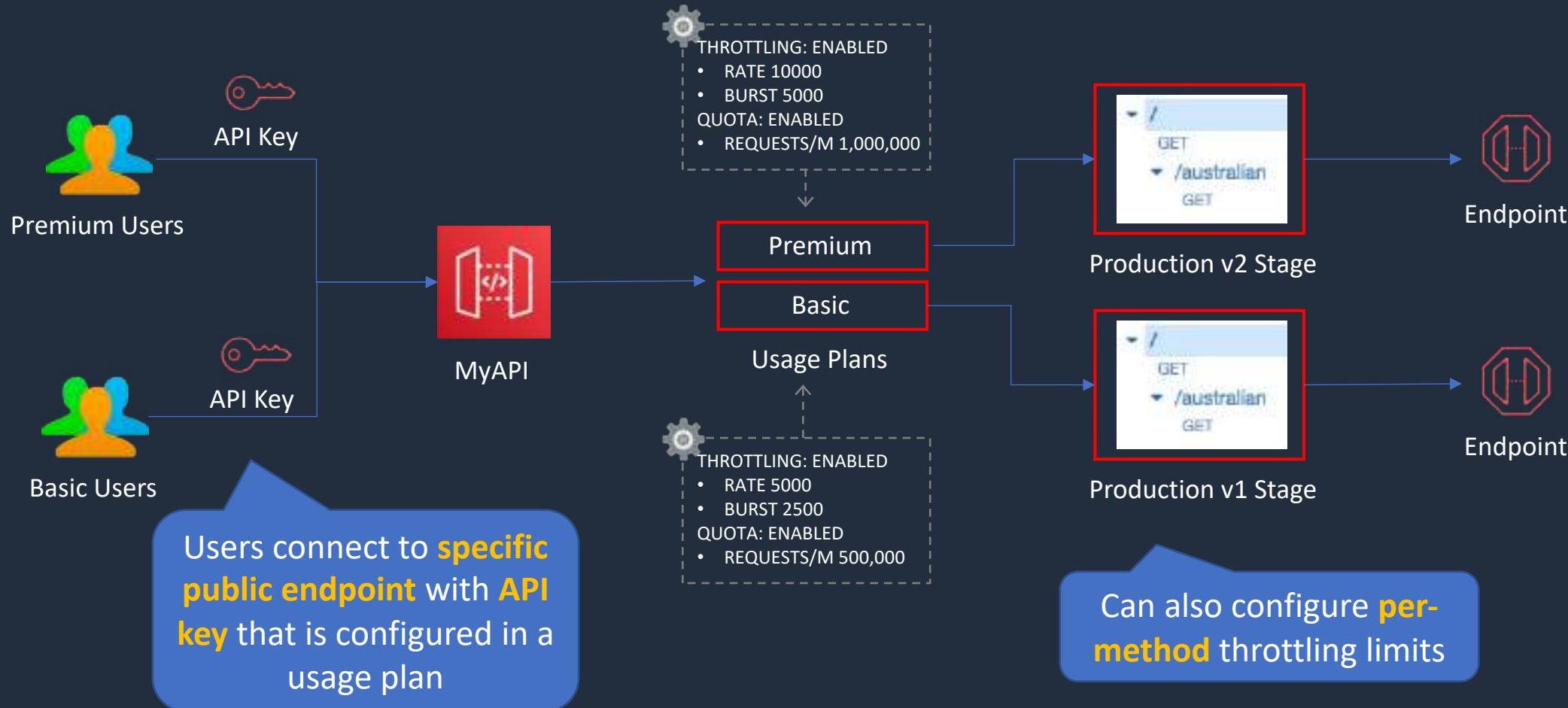
# API Gateway - Throttling

---

- API Gateway sets a limit on a steady-state rate and a burst of request submissions against all APIs in your account
- Limits:
  - By default API Gateway limits the steady-state request rate to 10,000 requests per second
  - The maximum concurrent requests is 5,000 requests across all APIs within an AWS account
  - If you go over 10,000 requests per second or 5,000 concurrent requests you will receive a **429 Too Many Requests** error response
- Upon catching such exceptions, the client can resubmit the failed requests in a way that is rate limiting, while complying with the API Gateway throttling limits



# API Gateway – Usage Plans and API Keys

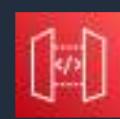


# Simple REST API

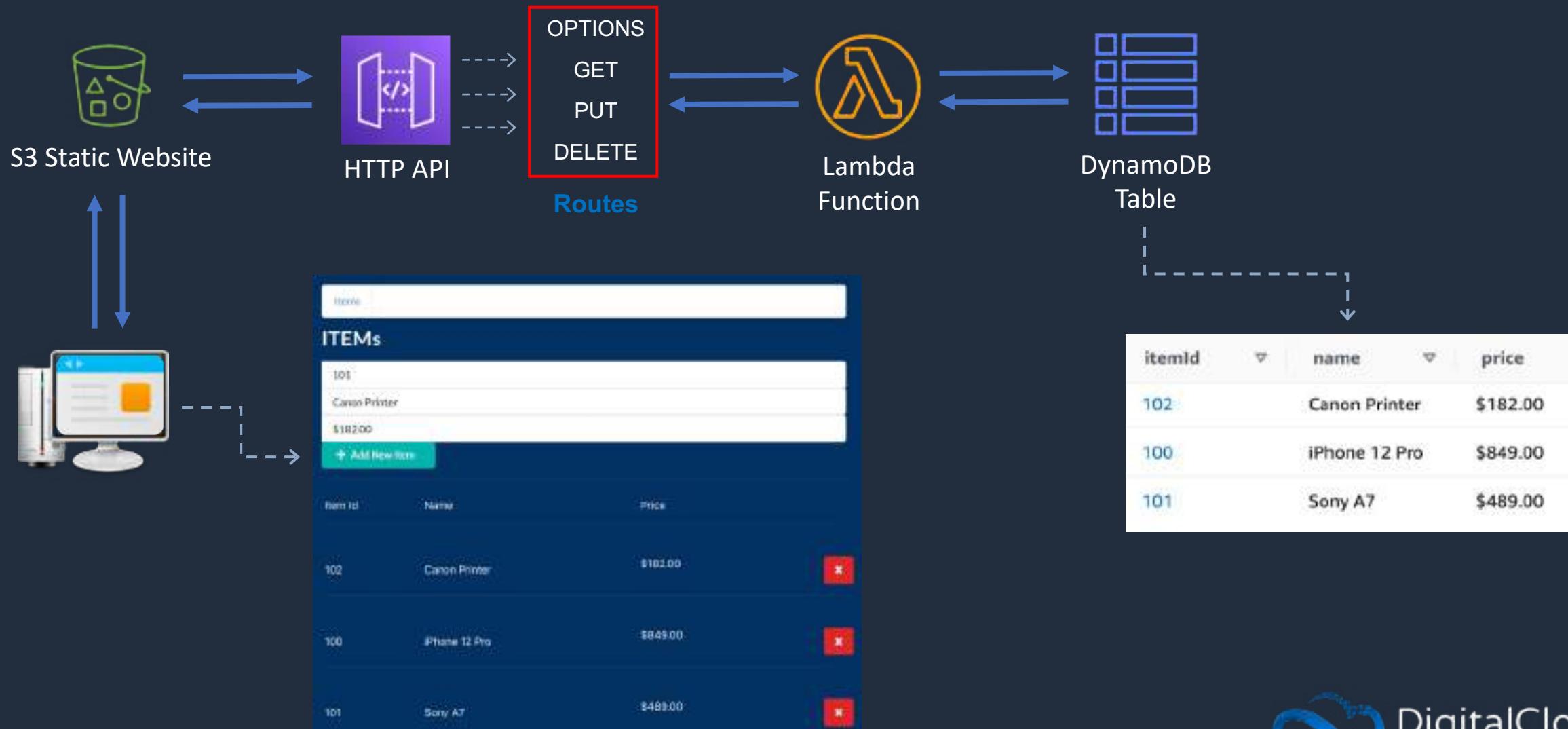


# Web App with HTTP API





# Web App with HTTP API



# Architecture Patterns - Serverless





# Architecture Patterns - Serverless

## Requirement

Application includes EC2 and RDS.  
Spikes in traffic causing writes to be dropped by RDS

## Solution

Decouple EC2 and RDS database with an SQS queue; use Lambda to process records in the queue

Web app includes a web tier and processing tier. Must be decoupled and processing tier should dynamically scale based on number of jobs

Decouple the web tier and processing tier with SQS and scale with Auto Scaling based on the queue length

Lambda function execution time has increased significantly as the payload size increased.

Optimize execution time by increasing memory available to the function which will proportionally increase CPU



# Architecture Patterns - Serverless

## Requirement

Statistical data is stored in RDS and will be accessed using a REST API. Demand will range from no traffic to sudden bursts of traffic and is unpredictable

New application processes customer orders and consists of multiple decoupled tiers. Orders must be processed in the order they are received

App uses API Gateway and Lambda. During busy periods, many requests fail multiple times before succeeding. No errors reported in Lambda

## Solution

Create a REST API using Amazon API Gateway and integrate with an AWS Lambda function for connecting to RDS

Implement an Amazon SQS FIFO queue to preserve the record order

Throttle limit could be configured with a value that is too low. Increase the throttle limit



# Architecture Patterns - Serverless

---

## Requirement

EC2 instance processes images using JavaScript code and stores results in S3. Load is highly variable. Need a more cost-effective solution

App uses API Gateway regional REST API. Just gone global and performance has suffered

Legacy application uses many batch scripts that process data and pass on to next script. Complex and difficult to maintain

## Solution

Replace EC2 with AWS Lambda function

Convert API to an edge-optimized API to optimize for the global user base

Migrate scripts to AWS Lambda functions and use AWS Step Functions to coordinate components



# Architecture Patterns - Serverless

---

---

## Requirement

Objects uploaded to an S3 bucket must be processed by AWS Lambda

## Solution

Create an event source notification to notify Lambda function to process new objects

# SECTION 12

## Databases and Analytics

# Types of Database





# 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 Database

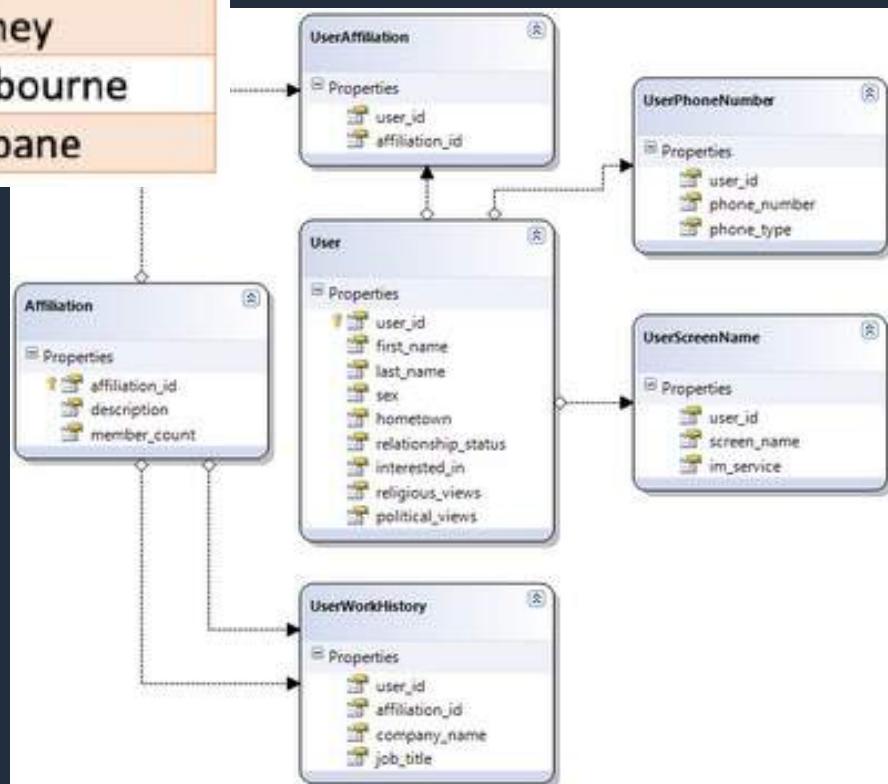
EmployeeID	FirstName	LastName	JobRole	Location
00001	Paul	Peterson	Senior Developer	Sydney
00002	Kaleigh	Annette	Assistant Manager	Brisbane
00003	Carl	Wood	Sales Support	Sydney
00004	Vinni	Jones	Customer Services	Melbourne
00005	Stefanie	Howard	IT Architect	Brisbane

Relational Database

Structured Query Language (SQL) query:

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

Relational Database – Multiple Tables



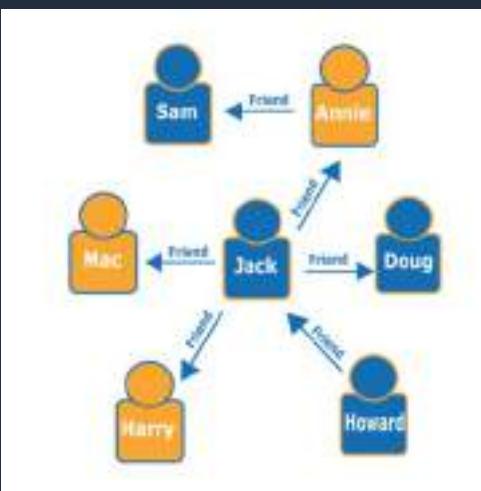


# Types of Non-Relational DB (NoSQL)

Key-value – e.g. Amazon DynamoDB



## Graph – e.g. Amazon Neptune



## Document – e.g. MongoDB



# Operational vs Analytical

---

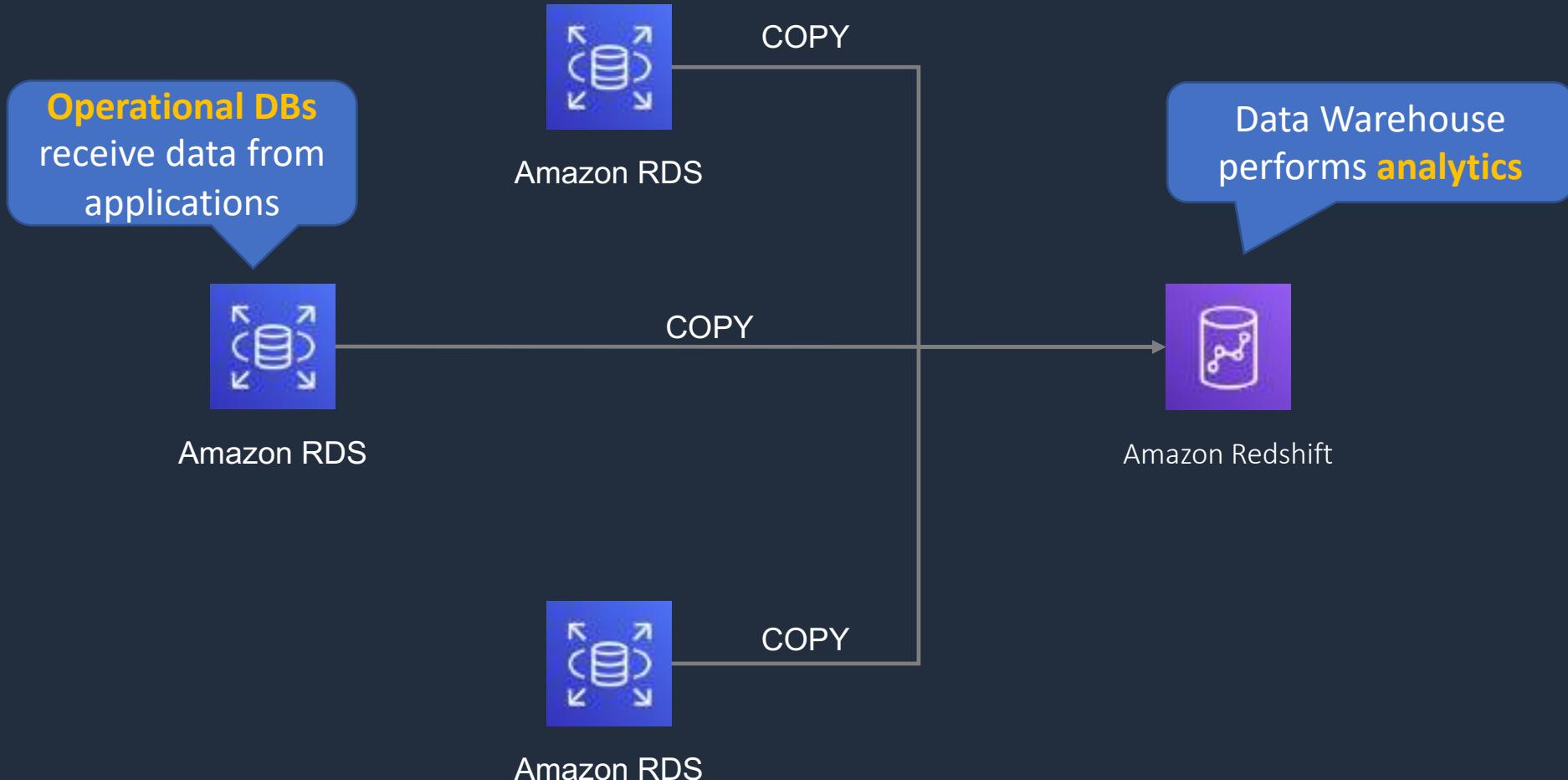
Key differences are **use cases** and how the database is **optimized**

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

---





# AWS Databases

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

# Amazon Relational Database Service (RDS)





# Amazon RDS

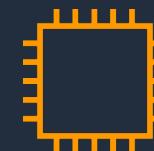
---

RDS is a **managed**, relational database



Amazon RDS

RDS runs on **EC2 instances**, so you must choose an **instance type**



EC2

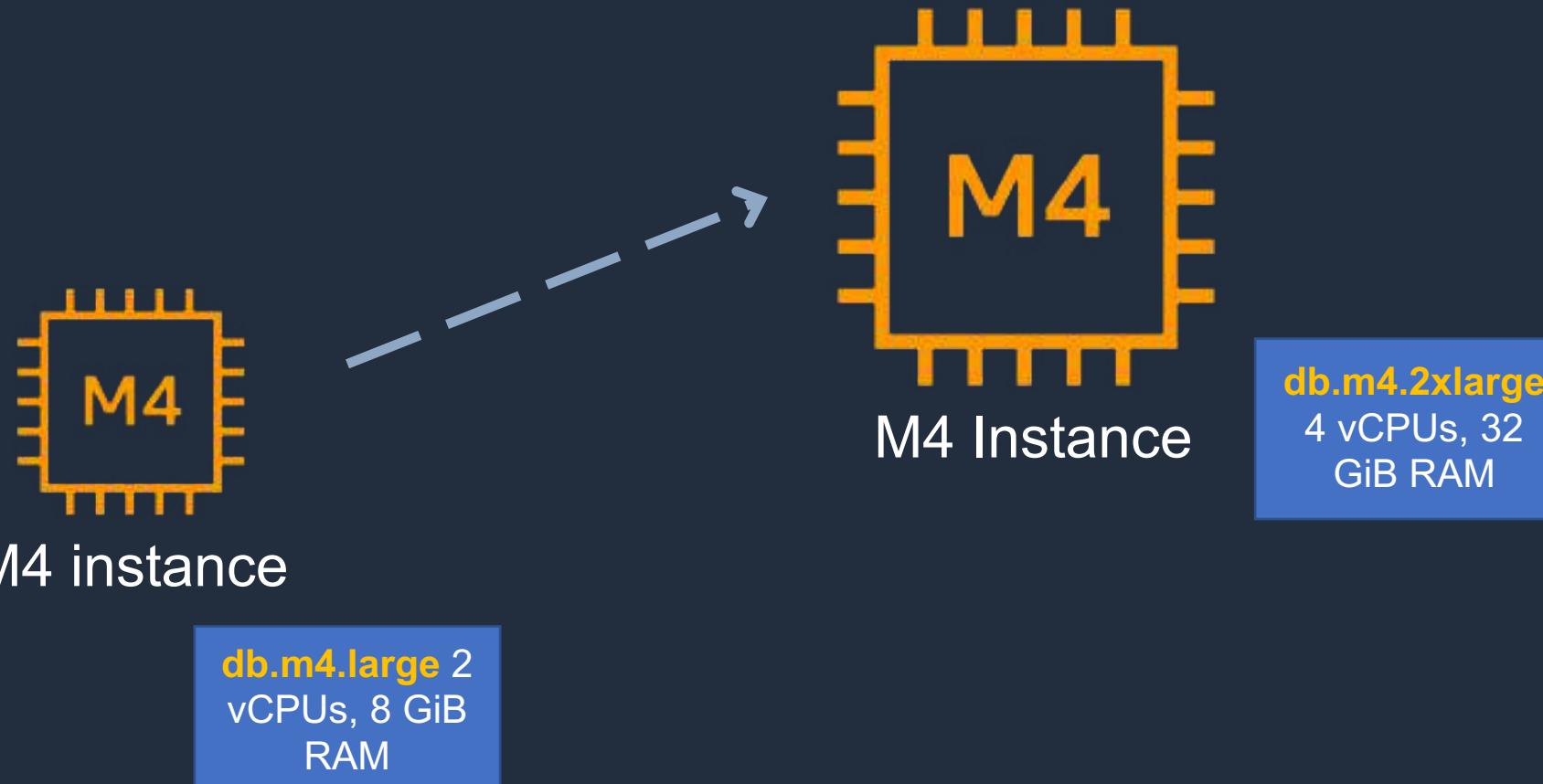
RDS supports the following database engines:

- Amazon Aurora
- MySQL
- MariaDB
- Oracle
- Microsoft SQL Server
- PostgreSQL



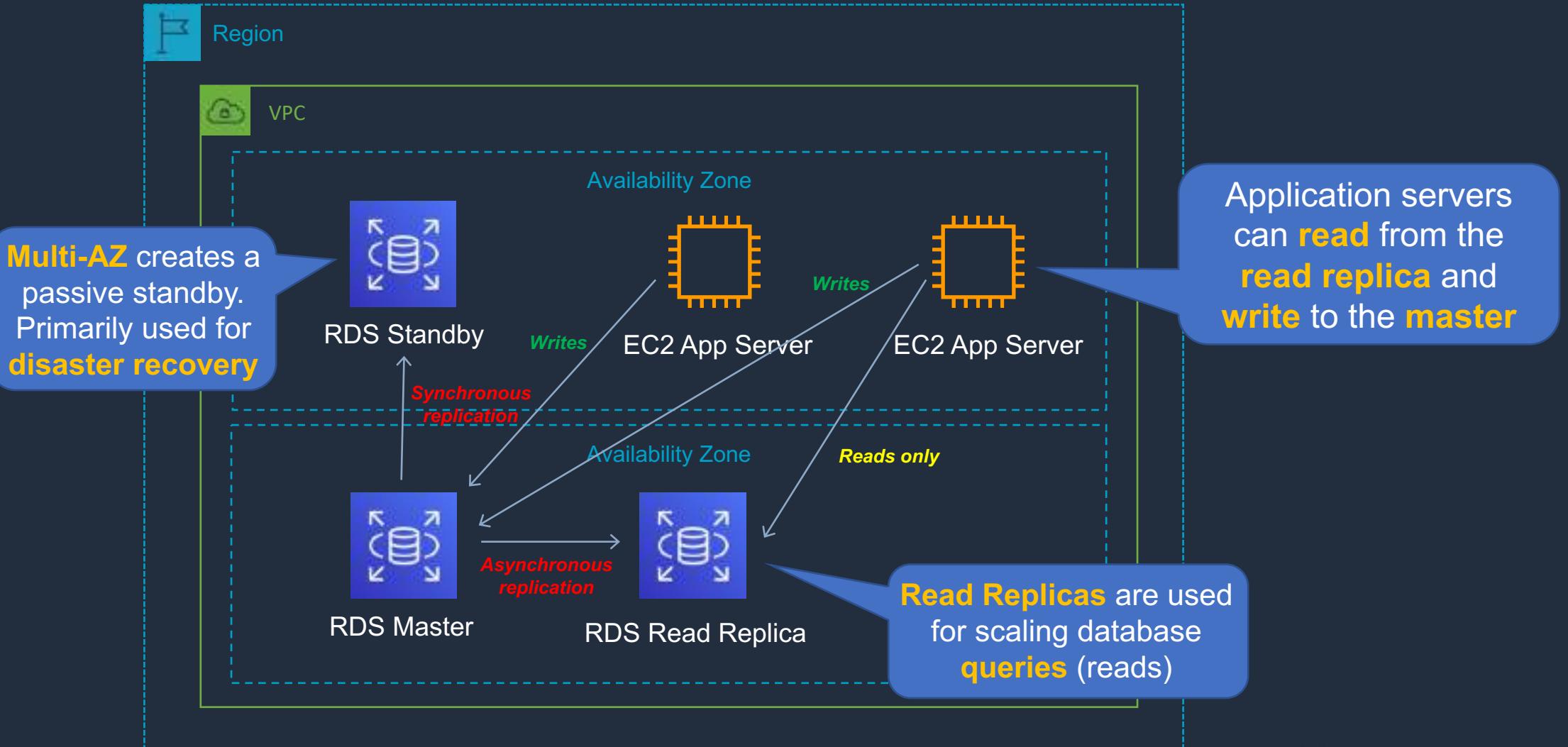
# Amazon RDS Scaling Up (vertically)

---





# Disaster Recovery (DR) and Scaling Out (Horizontally)





# Amazon RDS

---

---

- RDS uses EC2 instances, so you must choose an instance family/type
- Relational databases are known as Structured Query Language (SQL) databases
- RDS is an Online Transaction Processing (OLTP) type of database
- Easy to setup, highly available, fault tolerant, and scalable
- Common use cases include online stores and banking systems
- You can encrypt your Amazon RDS instances and snapshots at rest by enabling the encryption option for your Amazon RDS DB instance (during creation)
- Encryption uses AWS Key Management Service (KMS)



# Amazon RDS

---

- Amazon RDS supports the following database engines:
  - SQL Server
  - Oracle
  - MySQL Server
  - PostgreSQL
  - Aurora
  - MariaDB
- Scales up by increasing instance size (compute and storage)
- Read replicas option for read heavy workloads (scales out for reads/queries only)
- Disaster recovery with Multi-AZ option

# Amazon RDS Backup and Recovery





# Amazon RDS Automated Backups

Backup window [Info](#)

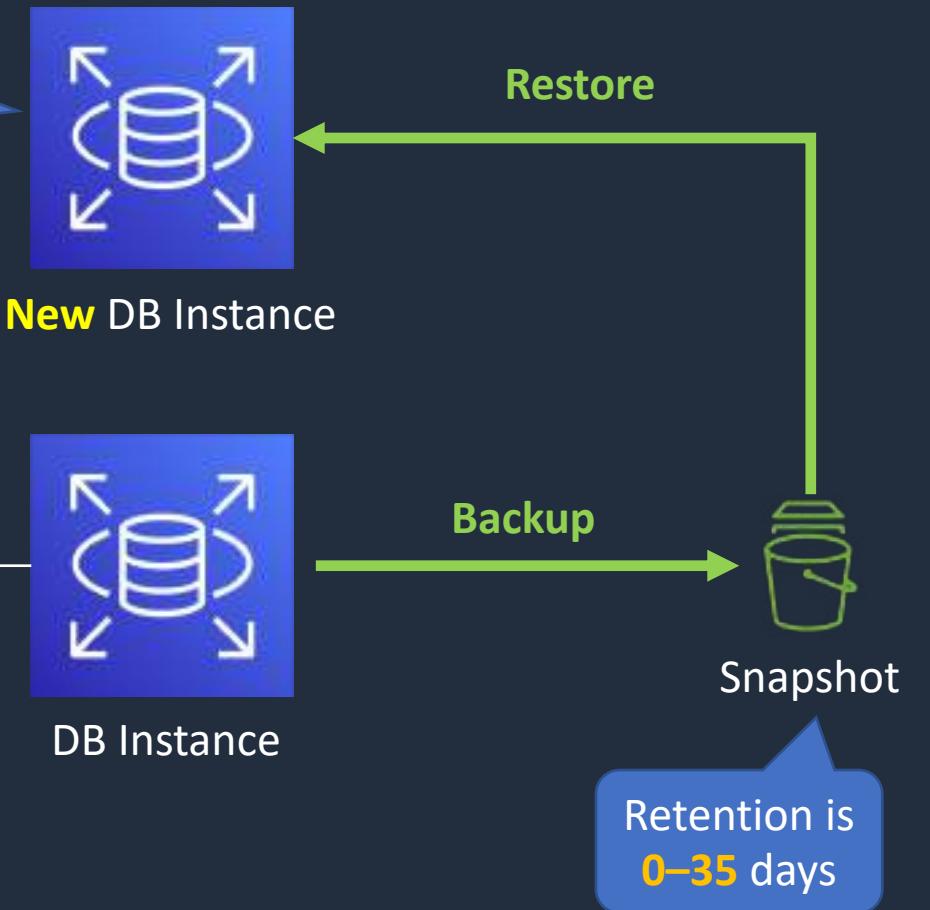
Select the period you want automated backups of the database to be created by Amazon RDS.

Select window

No preference

Start time  
00 : 00 UTC

Duration  
0.5 hours





# Amazon RDS Manual Backups (Snapshot)

---

- Backs up the entire DB instance, not just individual databases
- For single-AZ DB instances there is a brief suspension of I/O
- For Multi-AZ SQL Server, I/O activity is briefly suspended on primary
- For Multi-AZ MariaDB, MySQL, Oracle and PostgreSQL the snapshot is taken from the standby
- Snapshots do not expire (no retention period)



# Amazon RDS Maintenance Windows

- Operating system and DB patching can require taking the database offline
- These tasks take place during a maintenance window
- By default a weekly maintenance window is configured
- You can choose your own maintenance window

Maintenance window [Info](#)

Select the period you want pending modifications or maintenance applied to the database by Amazon RDS.

Select window

No preference

Start day	Start time	Duration
Monday	00 : 00 UTC	0.5 hours

# Create Amazon RDS Database



# Read Replicas and Multi-AZ

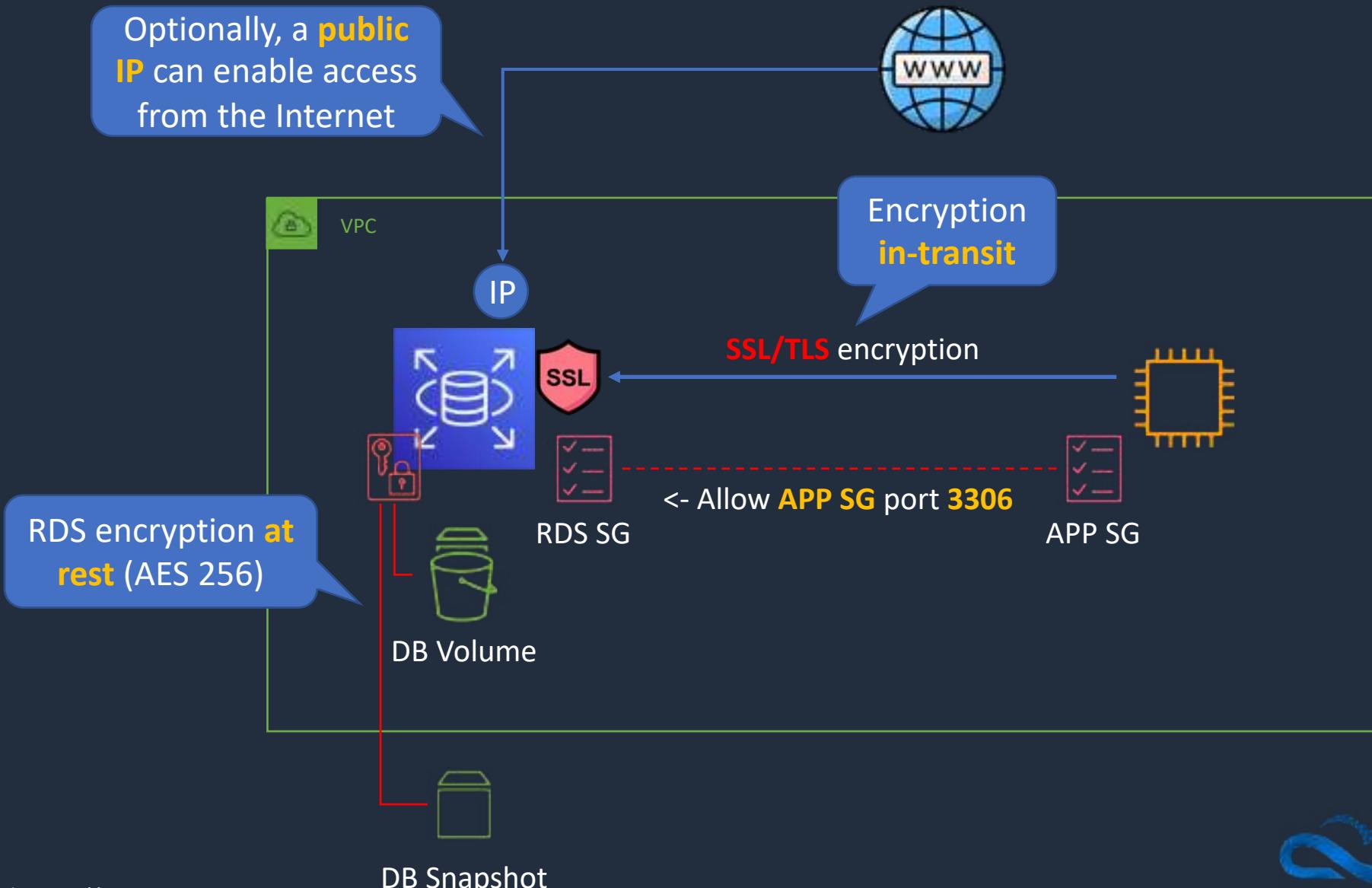


# Amazon RDS Security





# Amazon RDS Security





# Amazon RDS Security

---

---

- Encryption **at rest** can be enabled – includes DB storage, backups, read replicas and snapshots
- You can only enable encryption for an Amazon RDS DB instance when you create it, not after the DB instance is created
- DB instances that are encrypted can't be modified to disable encryption
- Uses AES 256 encryption and encryption is transparent with minimal performance impact
- RDS for Oracle and SQL Server is also supported using Transparent Data Encryption (TDE) (may have performance impact)
- AWS KMS is used for managing encryption keys



# Amazon RDS Security

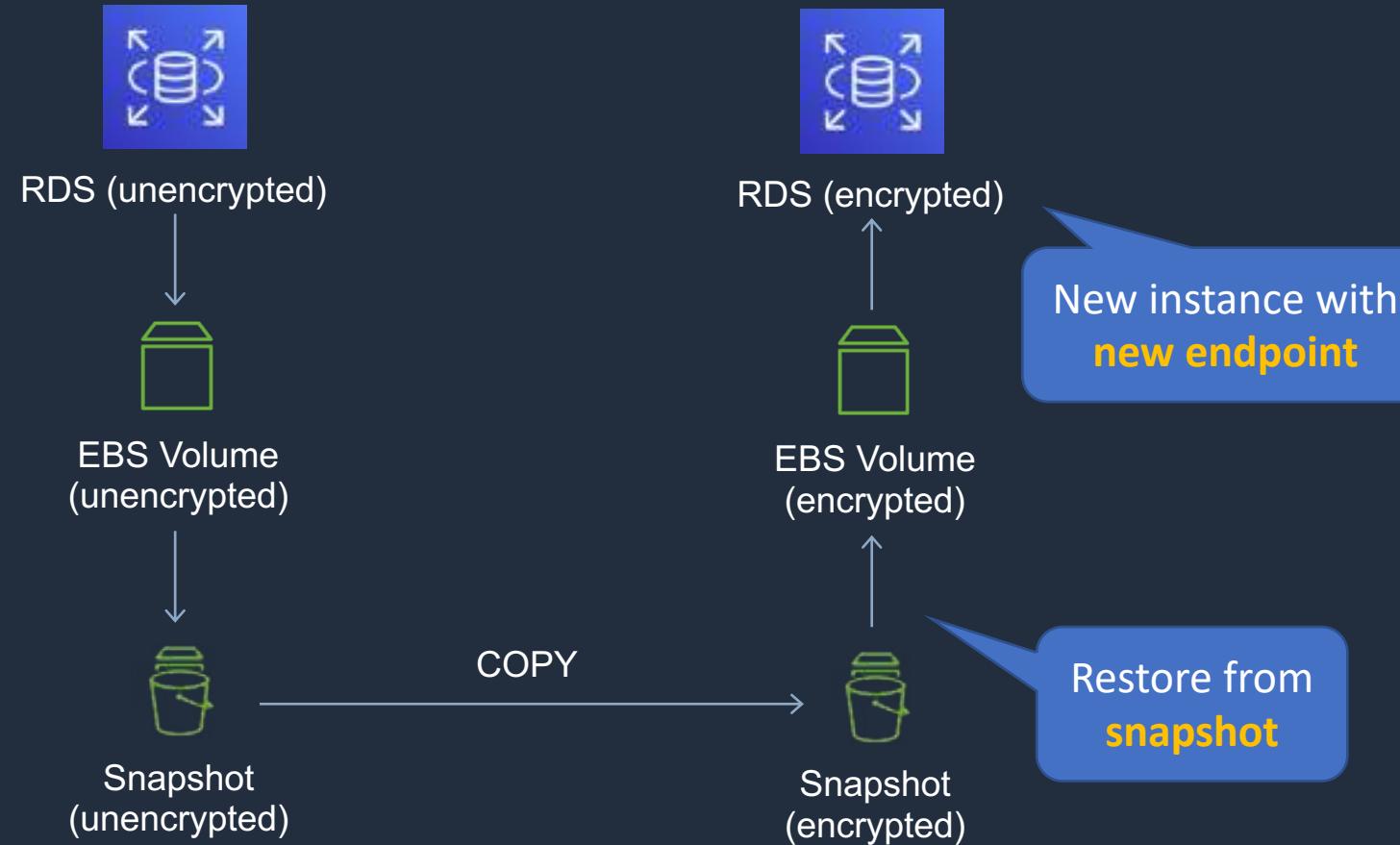
---

- You can't have:
  - An **encrypted** read replica of an **unencrypted** DB instance
  - An **unencrypted** read replica of an **encrypted** DB instance
- Read replicas of encrypted primary instances are encrypted
- The same KMS key is used if in the same Region as the primary
- If the read replica is in a different Region, a different KMS key is used
- You can't restore an unencrypted backup or snapshot to an encrypted DB instance



# Amazon RDS Security

---



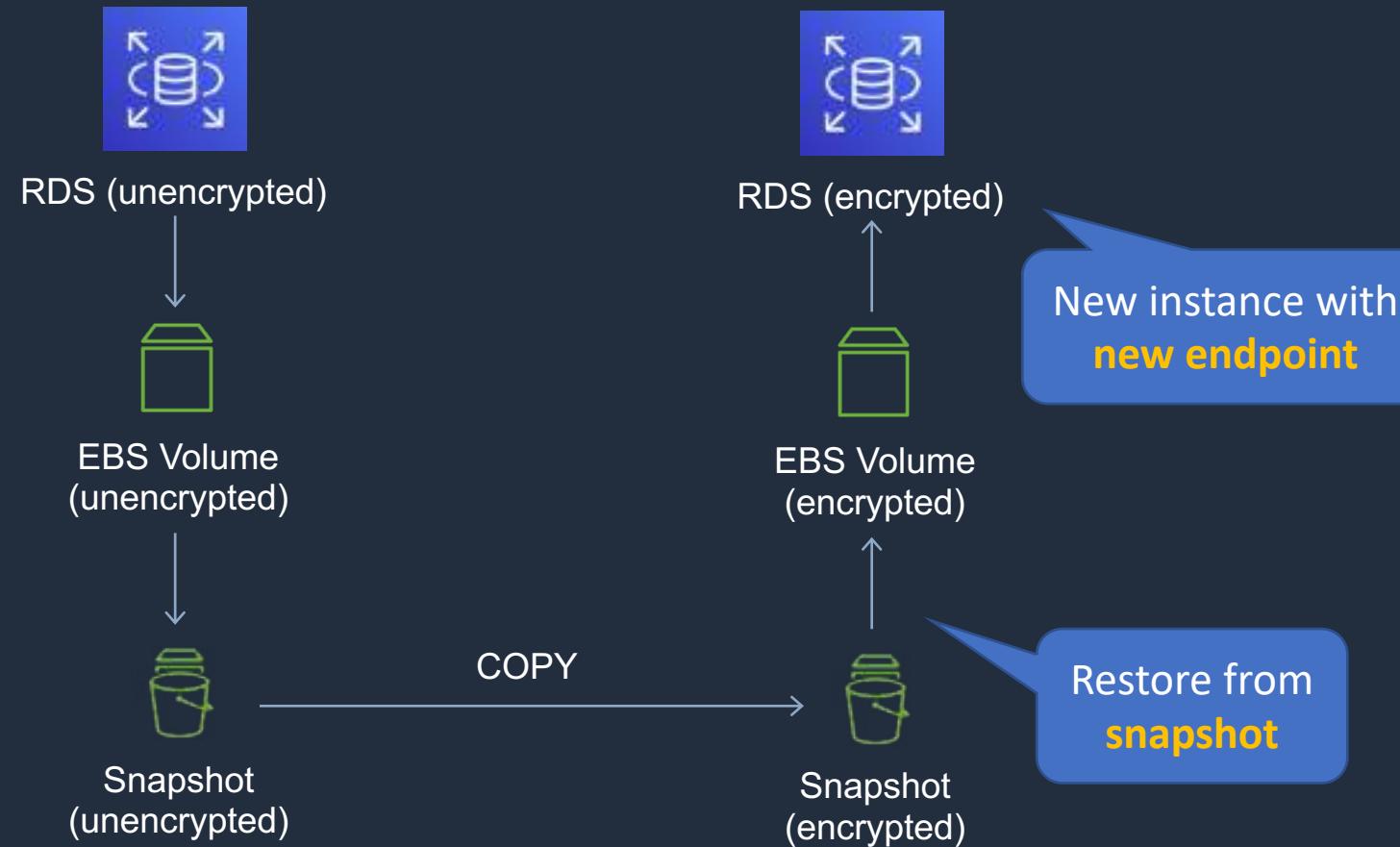
# Create Encrypted Copy of RDS Database





# Amazon RDS Security

---



# Amazon Aurora





# Amazon Aurora

---

- Amazon Aurora is an AWS database offering in the RDS family
- Amazon Aurora is a MySQL and PostgreSQL-compatible relational database built for the cloud
- Amazon Aurora is up to five times faster than standard MySQL databases and three times faster than standard PostgreSQL databases
- Amazon Aurora features a distributed, fault-tolerant, self-healing storage system that auto-scales up to 128TB per database instance

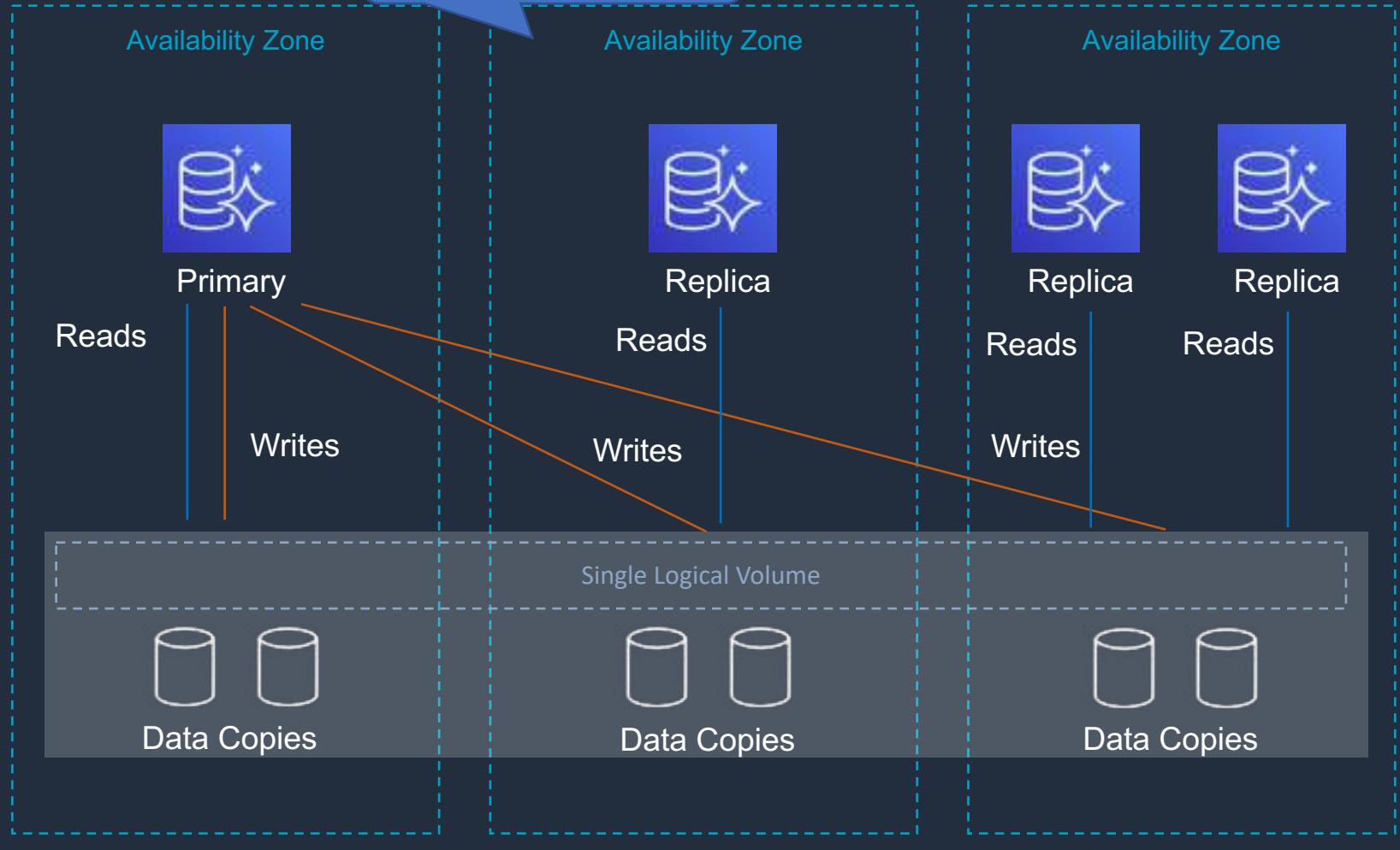


# Amazon Aurora



Region

Aurora Replicas are  
within a region



## Aurora Fault Tolerance

- Fault tolerance across 3 AZs
- Single logical volume
- Aurora Replicas scale-out read requests
- Can **promote** Aurora Replica to be a new primary or create new primary
- Can use **Auto Scaling** to add replicas



# Amazon Aurora Key Features

Aurora Feature	Benefit
<b>High performance and scalability</b>	Offers high performance, self-healing storage that scales up to 128TB, point-in-time recovery and continuous backup to S3
<b>DB compatibility</b>	Compatible with existing MySQL and PostgreSQL open source databases
<b>Aurora Replicas</b>	In-region read scaling and failover target – up to 15 (can use Auto Scaling)
<b>MySQL Read Replicas</b>	Cross-region cluster with read scaling and failover target – up to 5 (each can have up to 15 Aurora Replicas)
<b>Global Database</b>	Cross-region cluster with read scaling (fast replication / low latency reads). Can remove secondary and promote
<b>Multi-Master</b>	Scales out writes within a region. In preview currently and will not appear on the exam
<b>Serverless</b>	On-demand, autoscaling configuration for Amazon Aurora - does not support read replicas or public IPs (can only access through VPC or Direct Connect - not VPN)



# Amazon Aurora Replicas

Feature	Aurora Replica	MySQL Replica
<b>Number of replicas</b>	Up to 15	Up to 5
<b>Replication type</b>	Asynchronous (milliseconds)	Asynchronous (seconds)
<b>Performance impact on primary</b>	Low	High
<b>Replica location</b>	In-region	Cross-region
<b>Act as failover target</b>	Yes (no data loss)	Yes (potentially minutes of data loss)
<b>Automated failover</b>	Yes	No
<b>Support for user-defined replication delay</b>	No	Yes
<b>Support for different data or schema vs. primary</b>	No	Yes

# Amazon Aurora Deployment Options



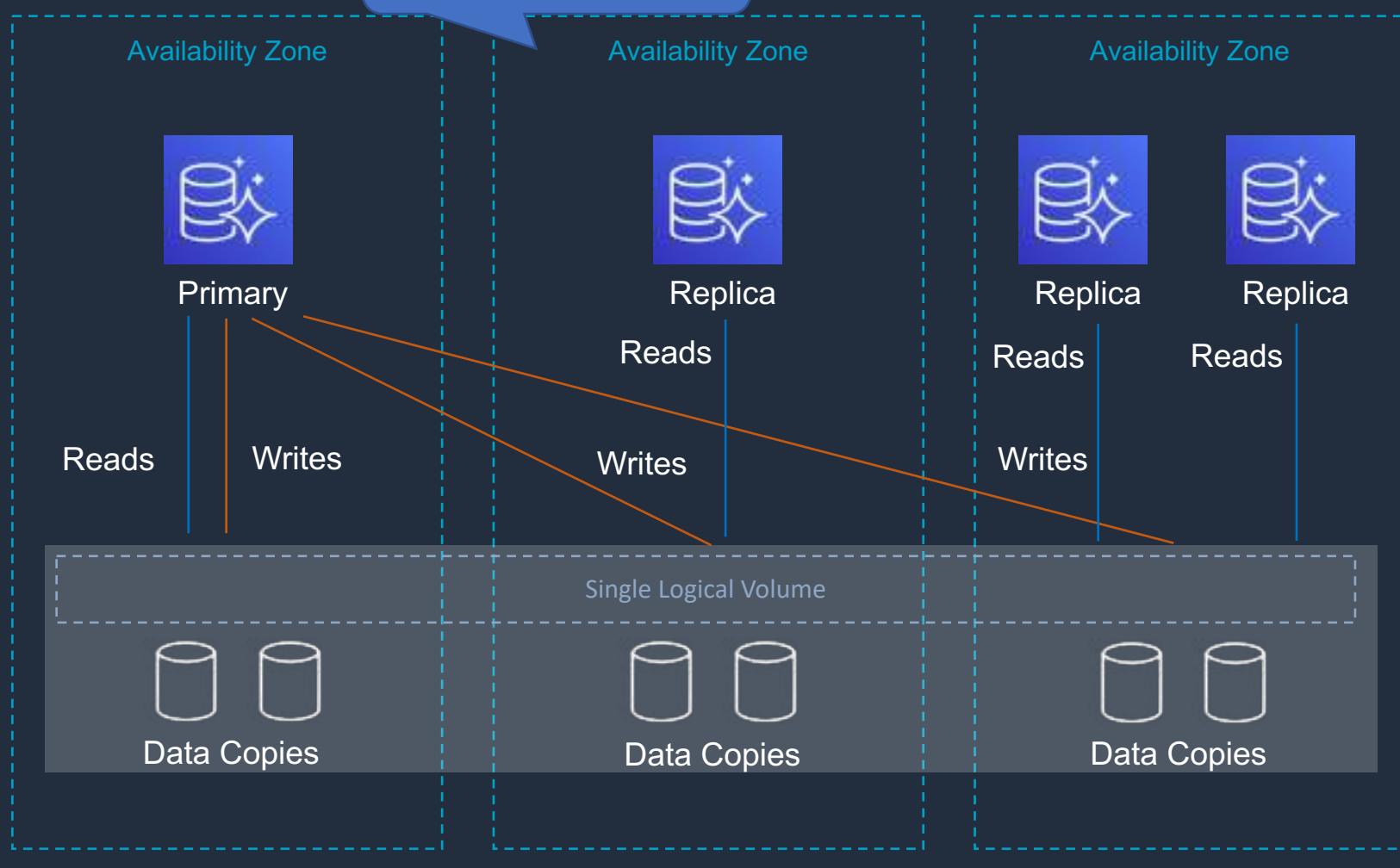


# Aurora Fault Tolerance and Aurora Replicas



Region

Aurora Replicas are  
within a region

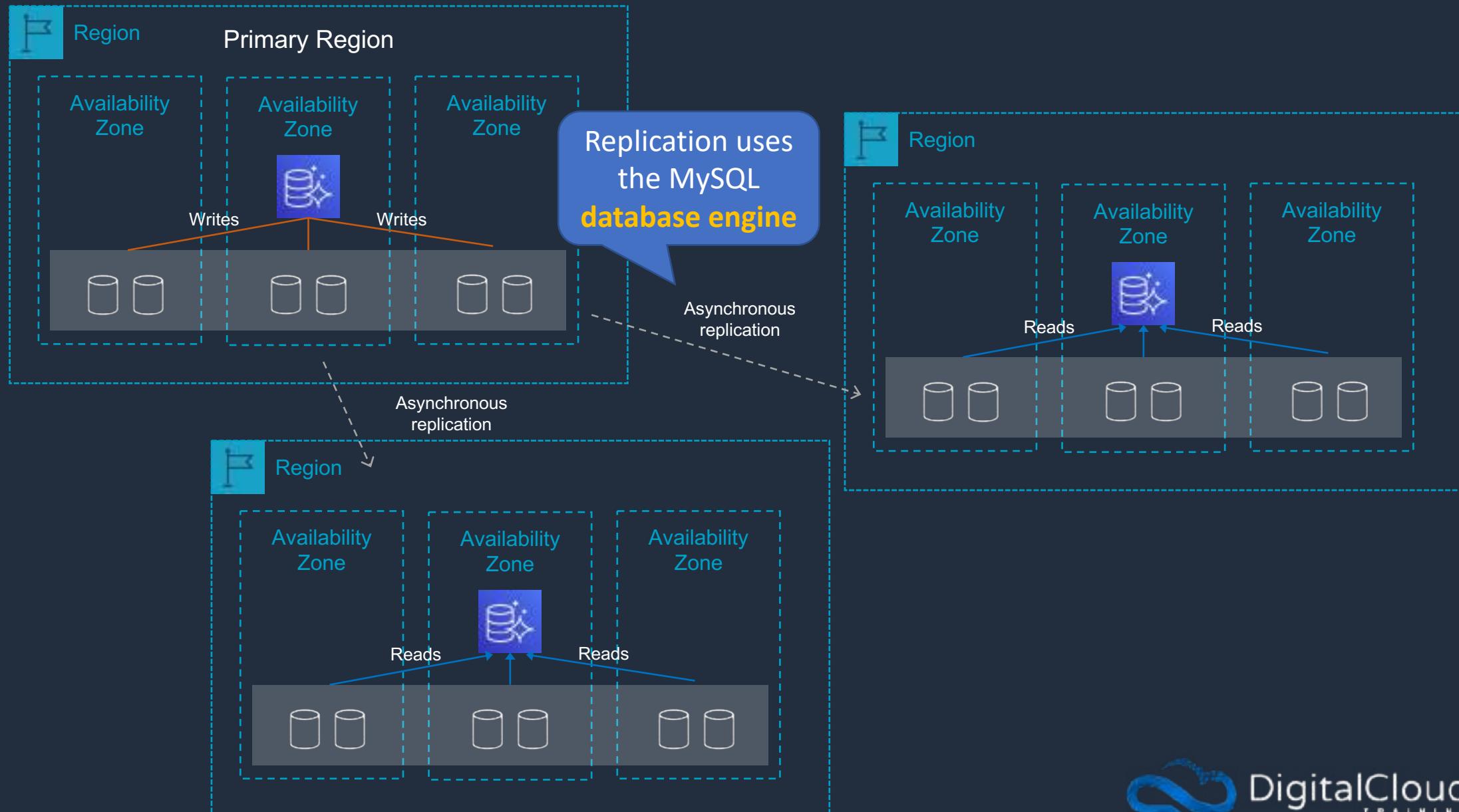


## Aurora Fault Tolerance

- Fault tolerance across 3 AZs
- Single logical volume
- Aurora Replicas scale-out read requests
- Up to 15 Aurora Replicas with **sub-10ms** replica lag
- Aurora Replicas are independent endpoints
- Can **promote** Aurora Replica to be a new primary or create new primary
- Set priority (tiers) on Aurora Replicas to control order of promotion
- Can use **Auto Scaling** to add replicas

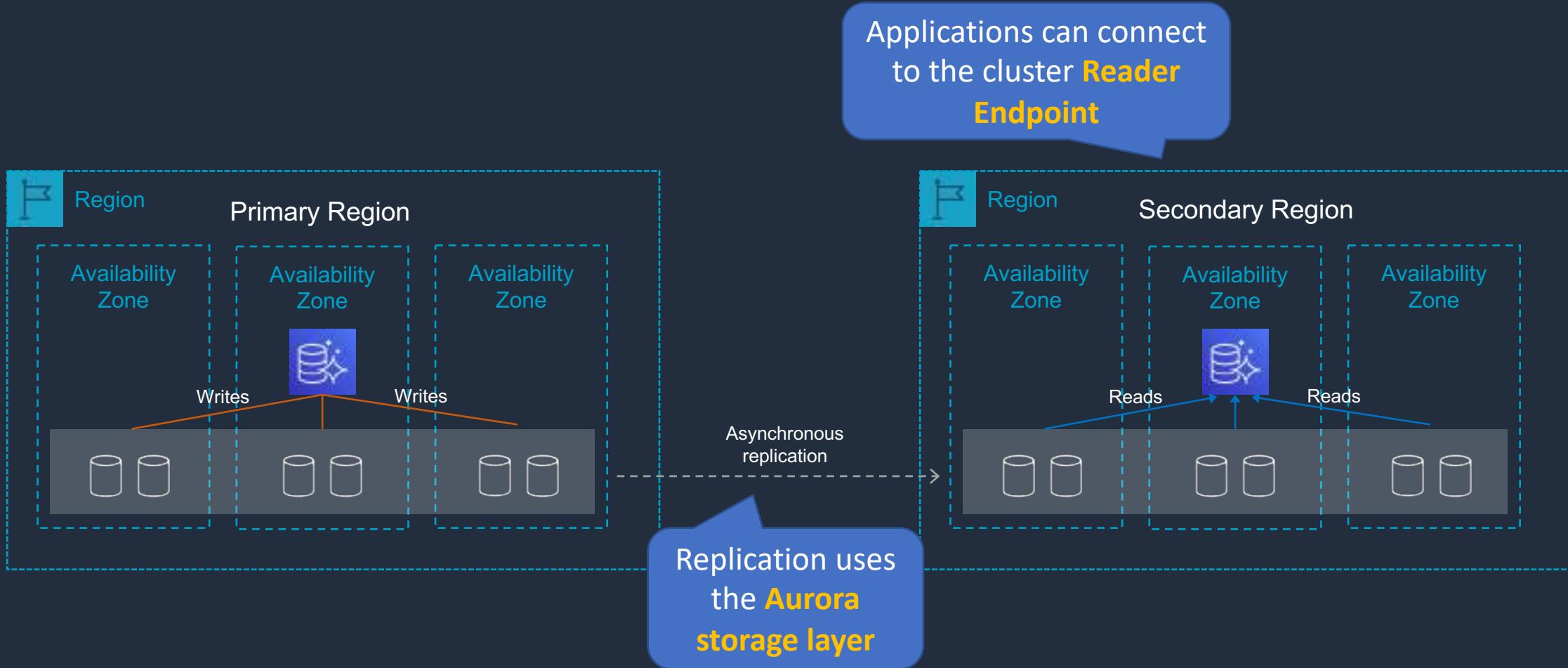


# Cross-Region Replica with Aurora MySQL





# Aurora Global Database

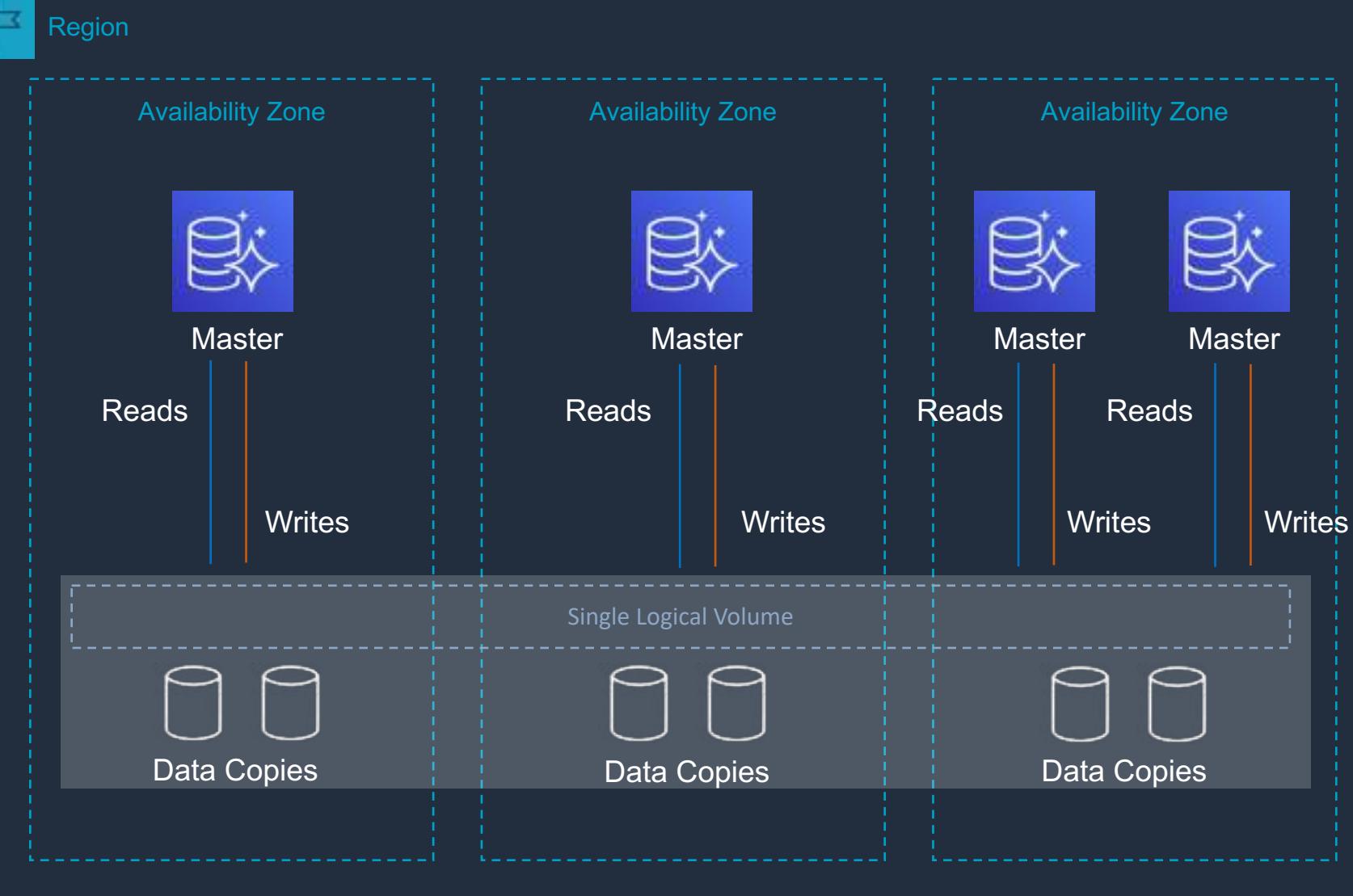




# Aurora Multi-Master



Region

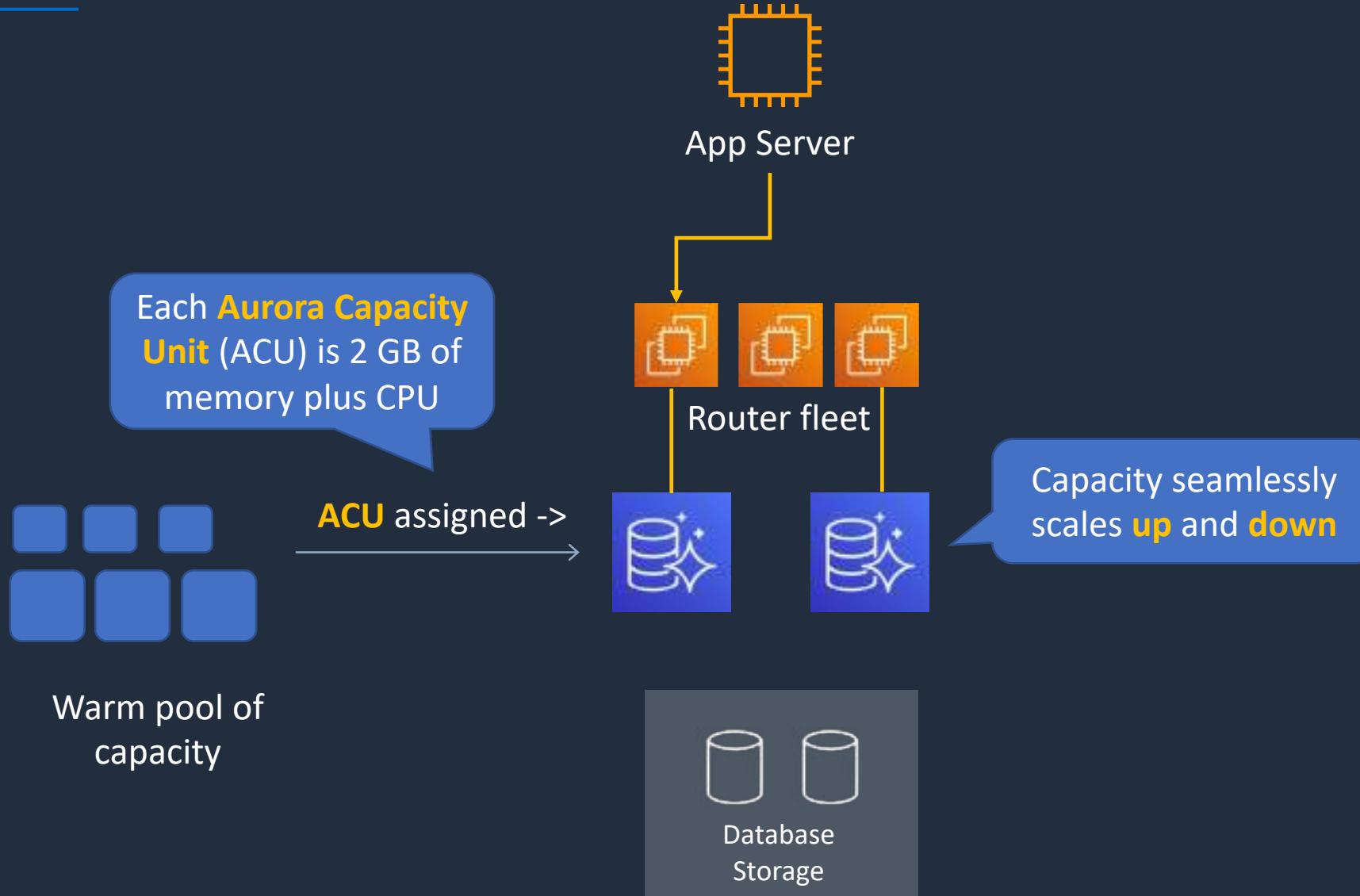


## Aurora Multi-Master

- All nodes allow reads/writes
- Available for MySQL only
- Up to four read/write nodes
- Single Region only
- Cannot have cross-Region replicas
- Can work with active-active and active-passive workloads
- Can restart read/write DB instance without impacting other instances



# Aurora Serverless





# Aurora Serverless Use Cases

---

---

- Infrequently used applications
- New applications
- Variable workloads
- Unpredictable workloads
- Development and test databases
- Multi-tenant applications

# Amazon RDS Anti-Patterns and Alternatives





# When NOT to use Amazon RDS (anti-patterns)

---

---

- Anytime you need a **DB type** other than:
  - MySQL
  - MariaDB
  - SQL Server
  - Oracle
  - PostgreSQL
- You need **root access** to the OS (e.g. install software such as management tools)



# When NOT to use Amazon RDS (anti-patterns)

---

---

Requirement	More Suitable Service
Lots of large binary objects (BLOBs)	S3
Automated Scalability	DynamoDB
Name/Value Data Structure	DynamoDB
Data is not well structured or unpredictable	DynamoDB
Other database platforms like IBM DB2 or SAP HANA	EC2
Complete control over the database	EC2



# Database on Amazon EC2

---

---

- You can run any database you like with **full control** and **ultimate flexibility**
- You must **manage everything** like backups, redundancy, patching and scaling



# Amazon ElastiCache

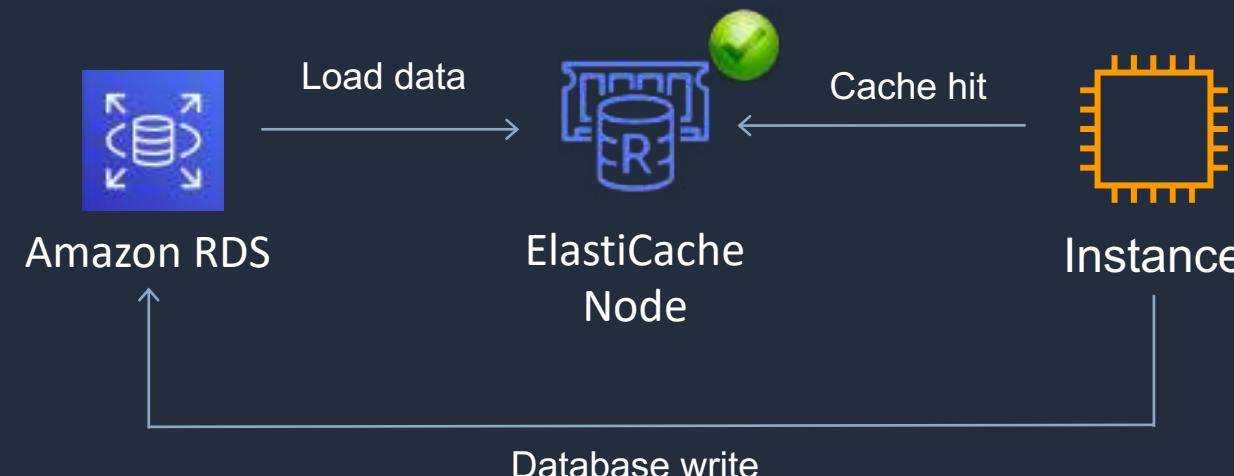




# Amazon ElastiCache

---

- Fully managed implementations **Redis** and **Memcached**
- ElastiCache is a **key/value** store
- In-memory database offering high performance and low latency
- Can be put in front of databases such as RDS and DynamoDB
- ElastiCache nodes run on Amazon EC2 instances, so you must choose an instance family/type





# Amazon ElastiCache

Feature	Memcached	Redis (cluster mode disabled)	Redis (cluster mode enabled)
<b>Data persistence</b>	No	Yes	Yes
<b>Data types</b>	Simple	Complex	Complex
<b>Data partitioning</b>	Yes	No	Yes
<b>Encryption</b>	No	Yes	Yes
<b>High availability (replication)</b>	No	Yes	Yes
<b>Multi-AZ</b>	Yes, place nodes in multiple AZs. No failover or replication	Yes, with auto-failover. Uses read replicas (0-5 per shard)	Yes, with auto-failover. Uses read replicas (0-5 per shard)
<b>Scaling</b>	Up (node type); out (add nodes)	Up (node type); out (add replica)	Up (node type); out (add shards)
<b>Multithreaded</b>	Yes	No	No
<b>Backup and restore</b>	No (and no snapshots)	Yes, automatic and manual snapshots	Yes, automatic and manual snapshots



# Amazon ElastiCache Use Cases

---

---

- Data that is relatively **static** and **frequently accessed**
- Applications that are tolerant of stale data
- Data is slow and expensive to get compared to cache retrieval
- Require push-button scalability for memory, writes and reads
- Often used for storing session state



# Amazon ElastiCache Examples

Use Case	Benefit
Web session store	In cases with load-balanced web servers, store web session information in Redis so if a server is lost, the session info is not lost, and another web server can pick it up
Database caching	Use Memcached in front of AWS RDS to cache popular queries to offload work from RDS and return results faster to users
Leaderboards	Use Redis to provide a live leaderboard for millions of users of your mobile app
Streaming data dashboards	Provide a landing spot for streaming sensor data on the factory floor, providing live real-time dashboard displays

# Scaling ElastiCache





# Amazon ElastiCache - Scalability

---

## Memcached

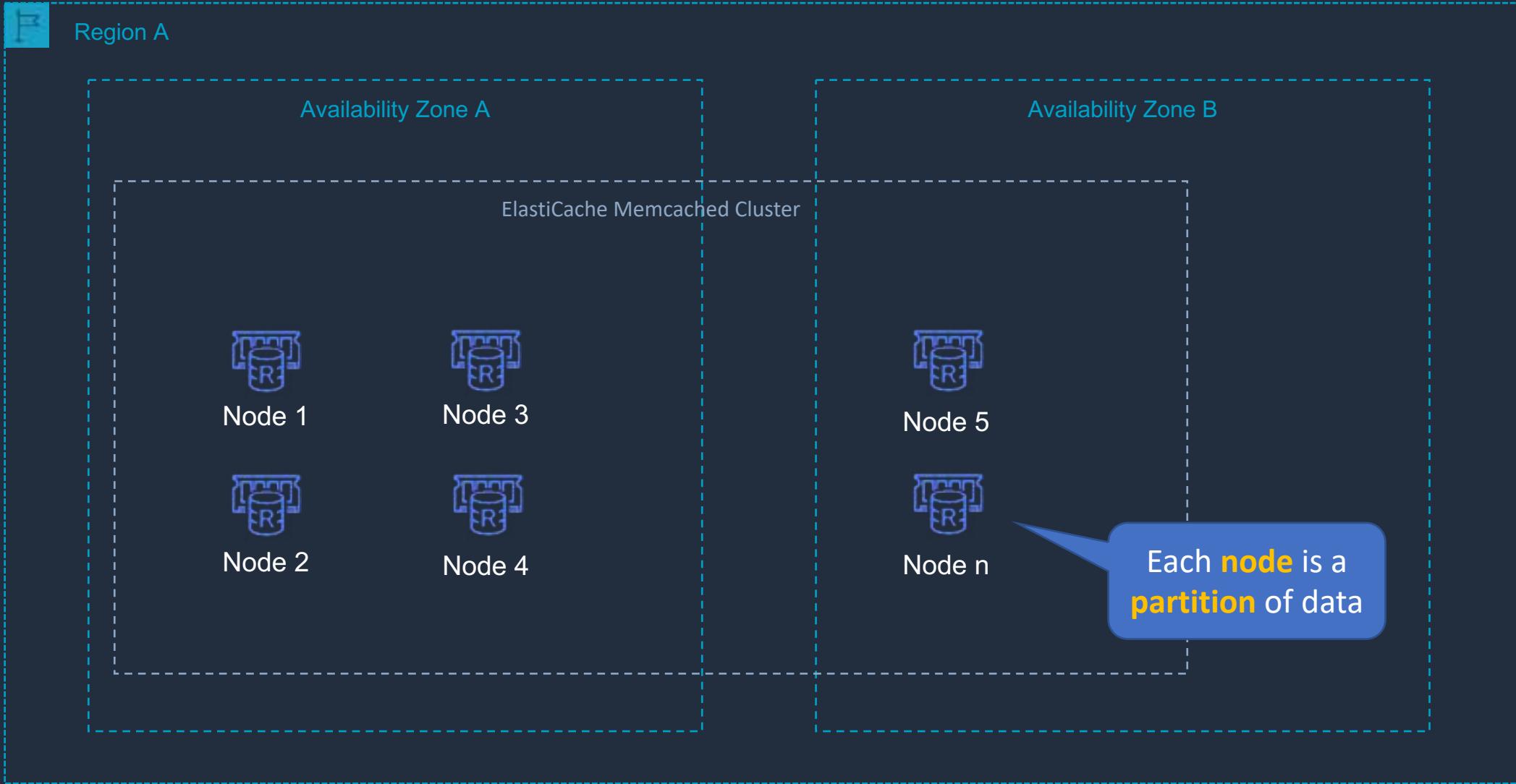
- Add nodes to a cluster
- Scale vertically (node type) – must create a **new cluster** manually

## Redis

- Cluster mode **disabled**:
  - Add replica or change node type – creates a new cluster and migrates data
- Cluster mode **enabled**:
  - Online resharding to add or remove shards; vertical scaling to change node type
  - Offline resharding to add or remove shards change node type or upgrade engine (more flexible than online)

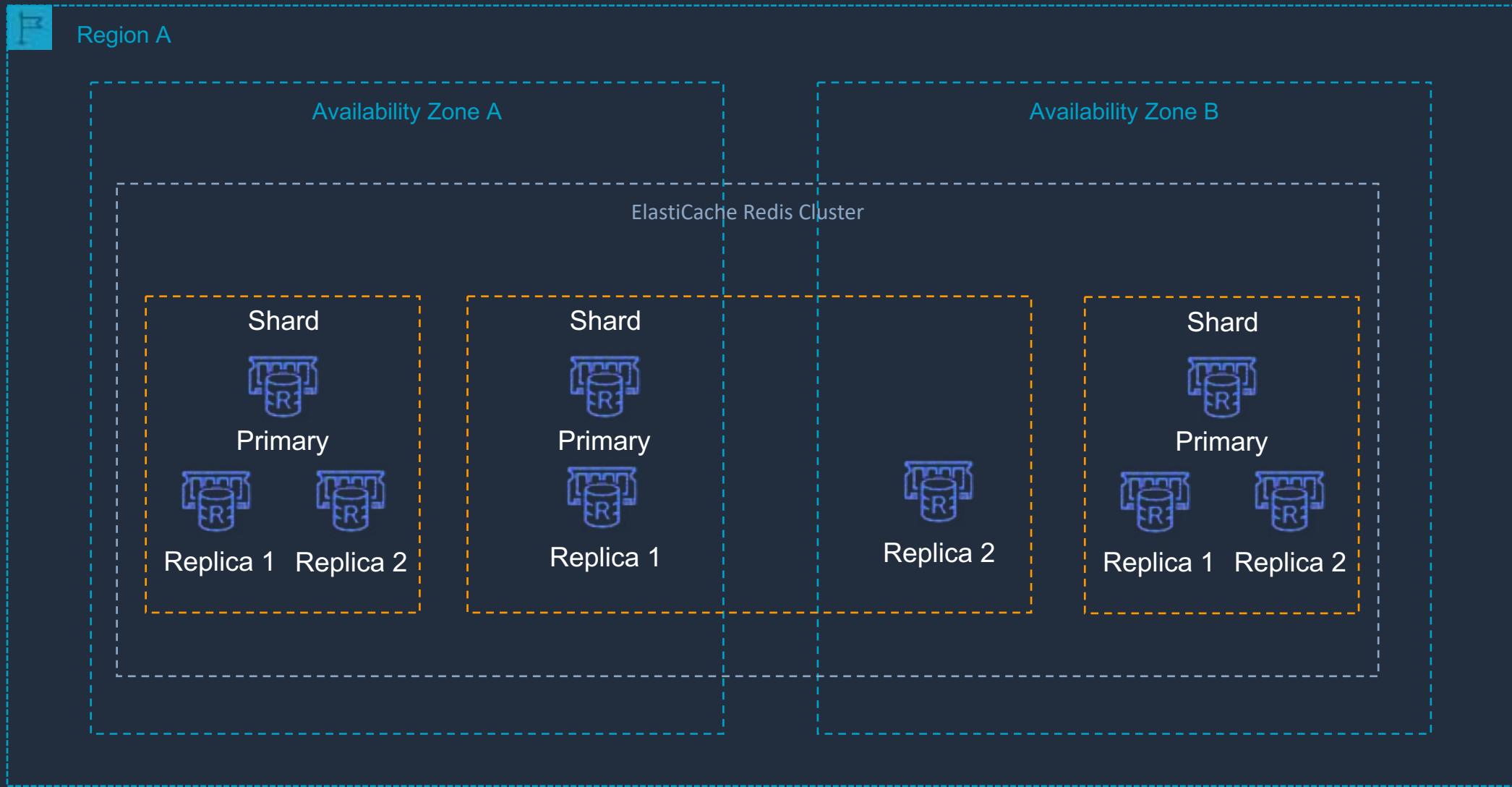


# Amazon ElastiCache Memcached



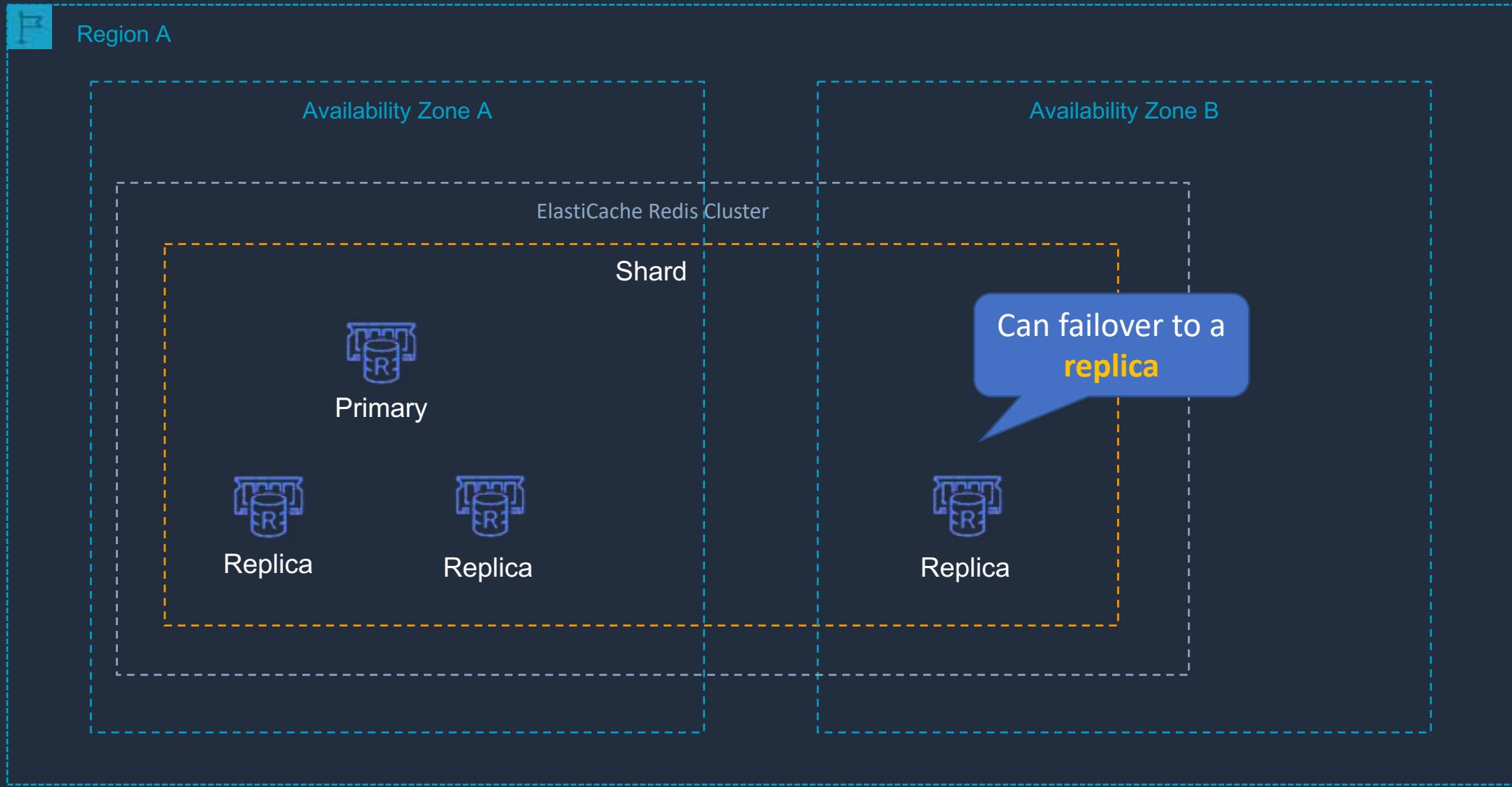


# Amazon ElastiCache Redis (Cluster mode enabled)





# Amazon ElastiCache Redis (Cluster mode disabled)



# Create ElastiCache Cluster



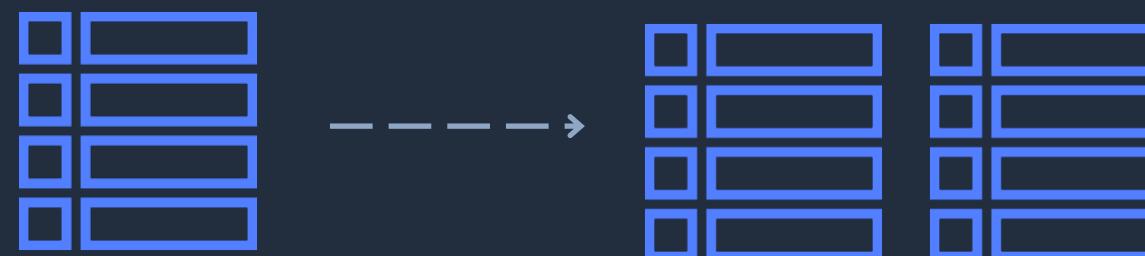
# Amazon DynamoDB





# Amazon DynamoDB

- Fully managed NoSQL database service
- Key/value store and document store
- It is a non-relational, key-value type of database
- Fully serverless service
- Push button scaling



DynamoDB Table



# Amazon DynamoDB

- DynamoDB is made up of:

- 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..



# DynamoDB Time to Live (TTL)

- TTL lets you define when items in a table expire so that they can be automatically deleted from the database
- With TTL enabled on a table, you can set a timestamp for deletion on a per-item basis
- No extra cost and does not use WCU / RCU
- Helps reduce storage and manage the table size over time



# Amazon DynamoDB

Primary Key		Attributes				
Partition Key	Sort Key	sku	category	size	colour	weight
clientid	created	SKU-S523	T-Shirt	Small	Red	Light
john@example.com	1583975308	SKU-J091	Pen		Blue	
chris@example.com	1583975613	SKU-A234	Mug			
chris@example.com	1583975449	SKU-R873	Chair			4011
sarah@example.com	1583976311	SKU-I019	Plate	30		
jenny@example.com	1583976323					

This is known as a  
**composite key** as it has a  
**partition key + sort key**

Attributes

Useful when the data structure is **unpredictable**



# Amazon DynamoDB

DynamoDB Feature	Benefit
Serverless	Fully managed, fault tolerant, service
Highly available	99.99% availability SLA – 99.999% for Global Tables!
NoSQL type of database with Name / Value structure	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 and durably stores the information for up to 24 hours. Often used with Lambda and the Kinesis Client Library (KCL)
DynamoDB Accelerator (DAX)	Fully managed in-memory cache for DynamoDB that increases performance (microsecond latency)
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

# Create DynamoDB Table

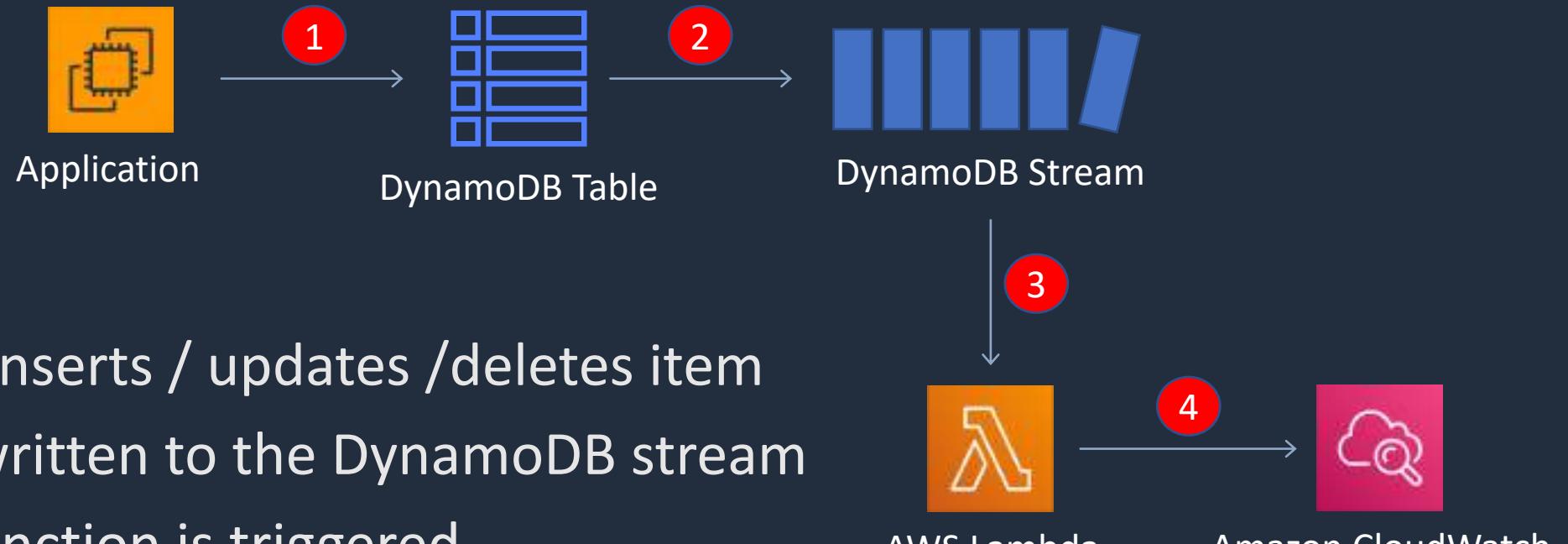


# DynamoDB Streams





# DynamoDB Streams



1. Application inserts / updates / deletes item
2. A record is written to the DynamoDB stream
3. A Lambda function is triggered
4. The Lambda function writes to CloudWatch Logs



# DynamoDB Streams

- Captures a **time-ordered** sequence of **item-level** modifications in any DynamoDB table and stores this information in a log for up to **24 hours**
- Can configure the information that is written to the stream:
  - **KEYS\_ONLY** — Only the key attributes of the modified item
  - **NEW\_IMAGE** — The entire item, as it appears after it was modified
  - **OLD\_IMAGE** — The entire item, as it appeared before it was modified
  - **NEW\_AND\_OLD\_IMAGES** — Both the new and the old images of the item

# DynamoDB Accelerator (DAX)



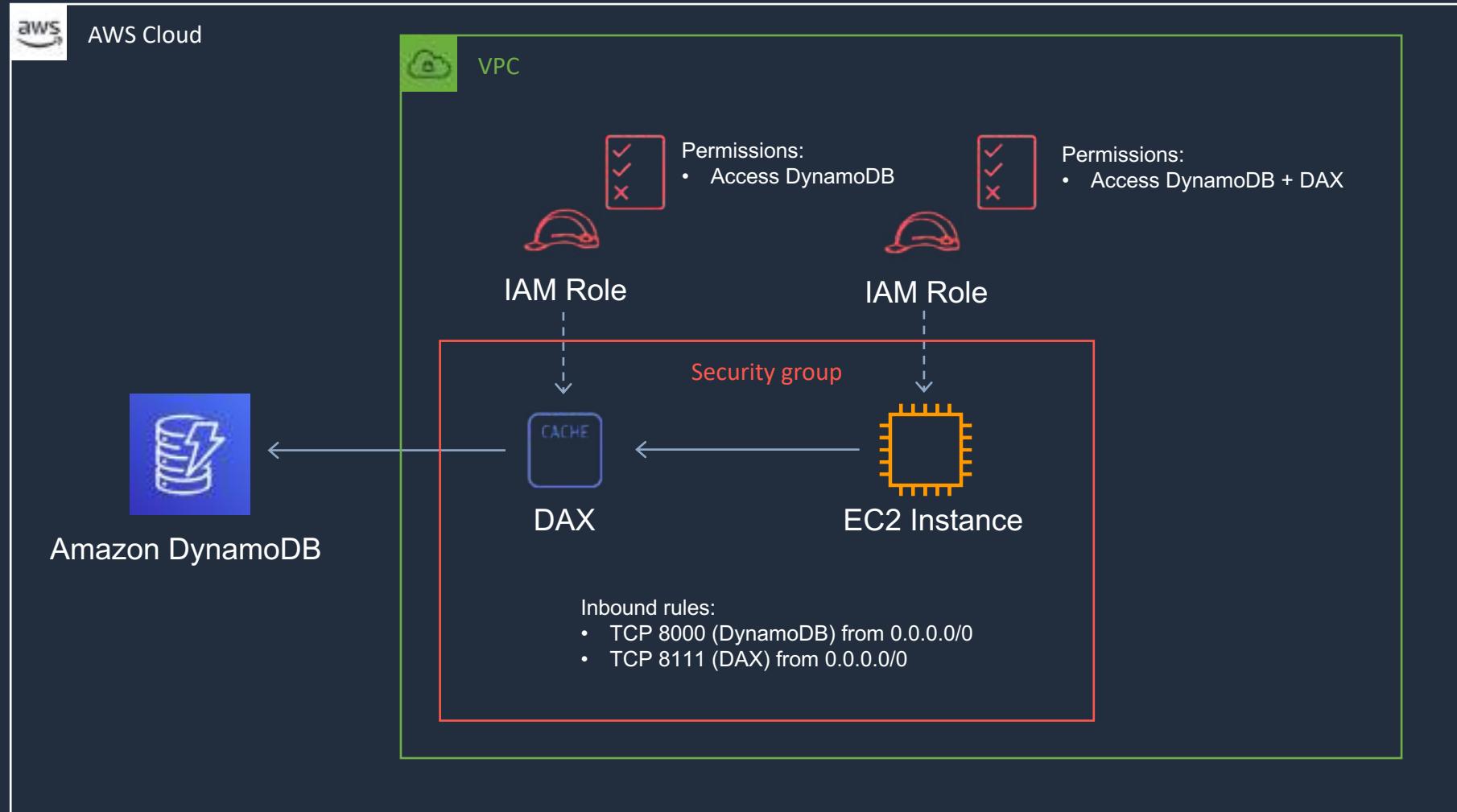


# DynamoDB Accelerator (DAX)

- DAX is a fully managed, highly available, **in-memory** cache for DynamoDB
- Improves performance from milliseconds to microseconds
- Can be a read-through cache and a write-through cache
- Used to improve **READ** and **WRITE** performance
- You do not need to modify application logic, since DAX is compatible with existing DynamoDB API calls



# DynamoDB Accelerator (DAX)





# DAX vs ElastiCache

---

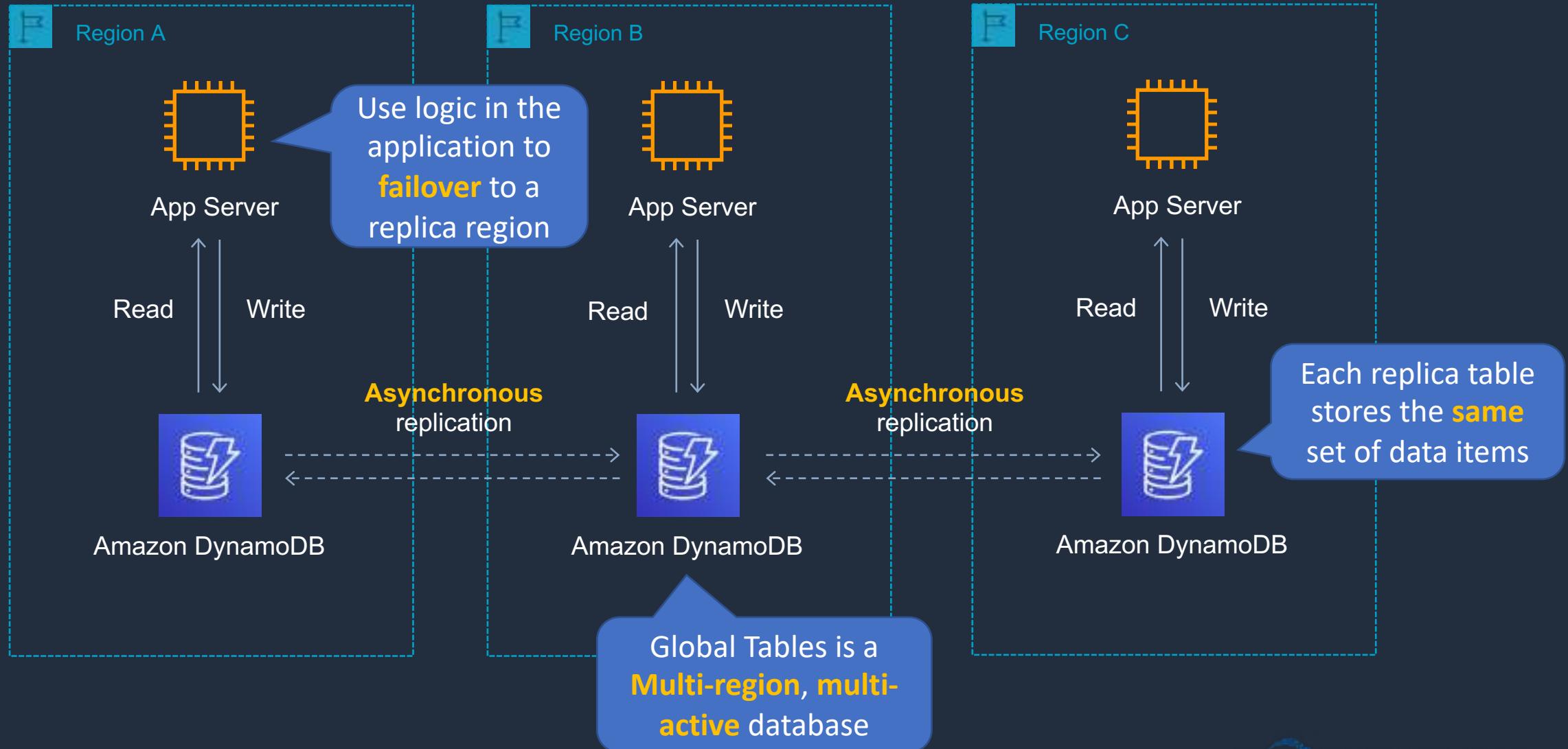
- DAX is **optimized** for DynamoDB
- With ElastiCache you have more **management overhead** (e.g. invalidation)
- With ElastiCache you need to **modify application code** to point to cache
- ElastiCache supports more datastores

# DynamoDB Global Tables





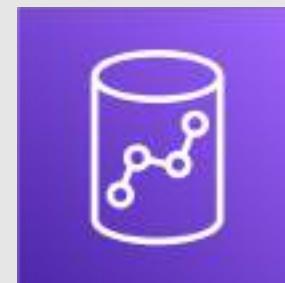
# DynamoDB Global Tables



# Create DynamoDB Global Table



# Amazon RedShift





# Amazon RedShift

---

---

- Amazon Redshift is a fast, fully managed data warehouse
- Analyze data using standard SQL and existing Business Intelligence (BI) tools
- RedShift is a SQL based data warehouse used for analytics applications
- RedShift is a relational database that is used for Online Analytics Processing (OLAP) use cases
- RedShift uses Amazon EC2 instances, so you must choose an instance family/type
- RedShift always keeps three copies of your data
- RedShift provides continuous/incremental backups



# OLTP vs OLAP (refresher)

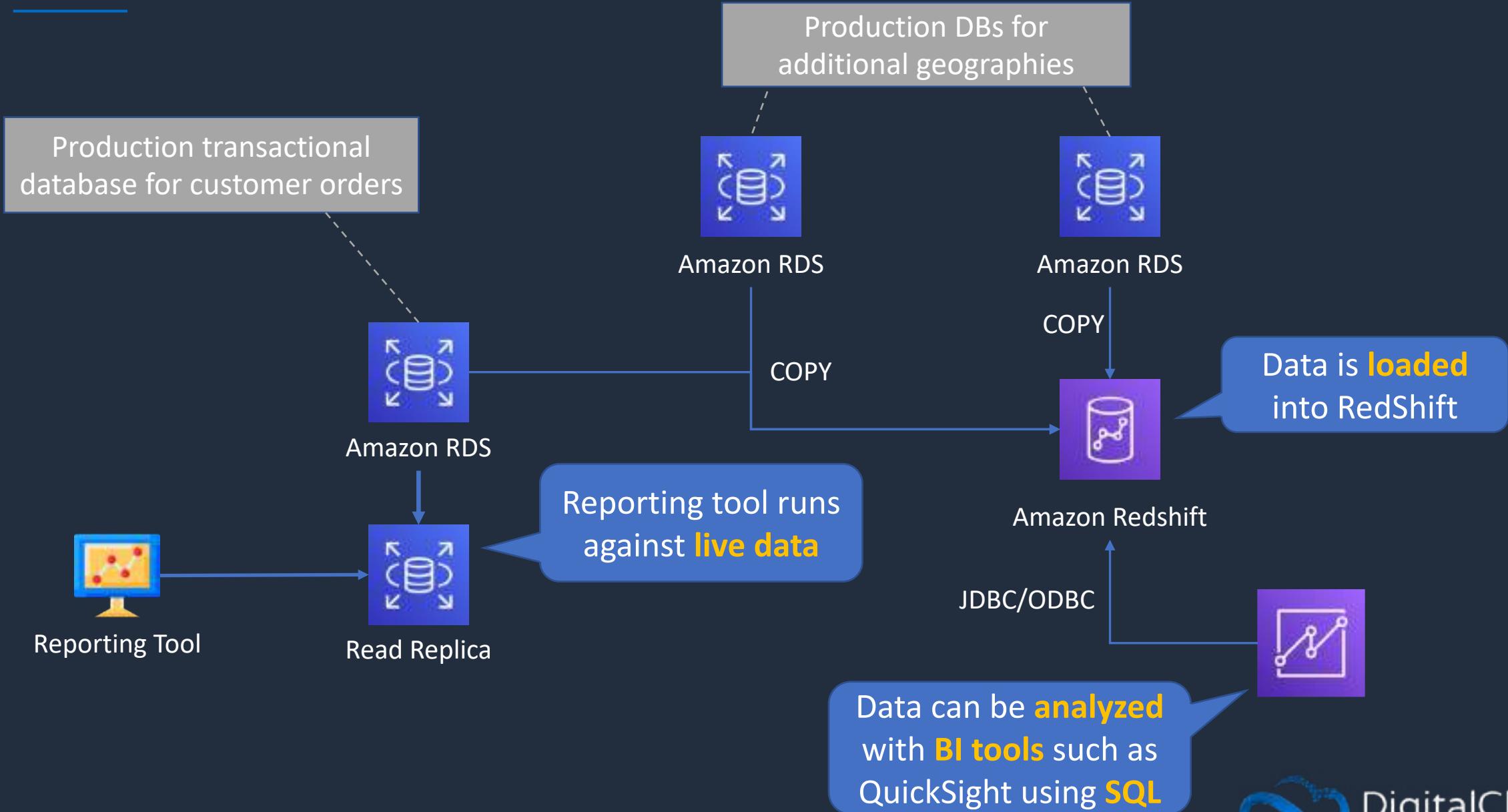
---

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
Examples: Amazon RDS, DynamoDB	Examples: Amazon RedShift, Amazon EMR



# Reporting and Analytics Use Cases





# Redshift Data Sources



**RedShift Spectrum** can  
run SQL queries on data  
directly in **S3**



# RedShift Use Cases

- Perform **complex queries** on massive collections of **structured** and **semi-structured** data and get fast performance
- Frequently accessed data that needs a consistent, highly structured format
- Use **Spectrum** for direct access of **S3 objects** in a data lake
- Managed data warehouse solution with:
  - Automated provisioning, configuration and patching
  - Data durability with continuous backup to S3
  - Scales with simple API calls
  - Exabyte scale query capability

# Amazon Elastic Map Reduce (EMR)





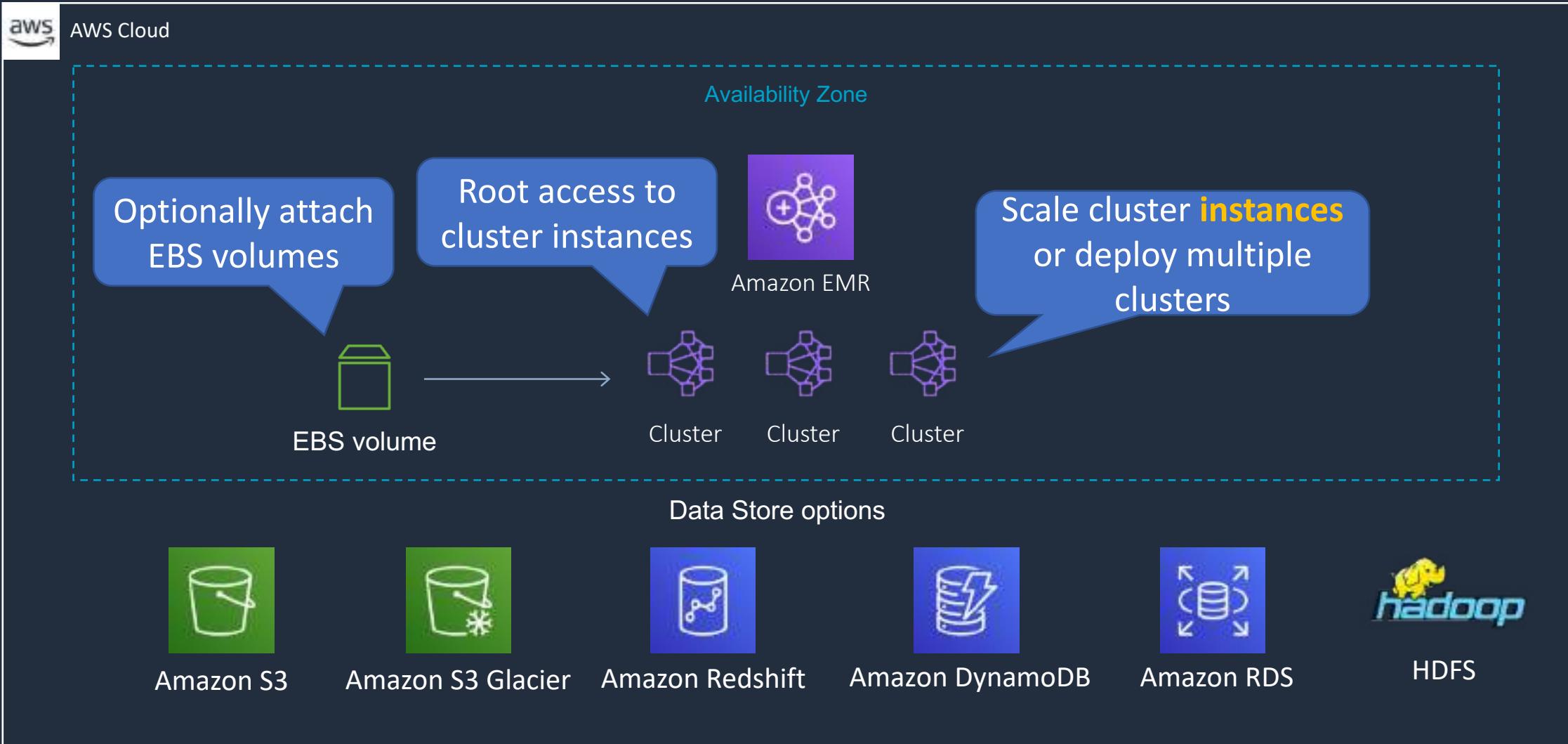
# Amazon EMR

---

- Managed cluster platform that simplifies running big data frameworks including **Apache Hadoop** and **Apache Spark**
- Used for processing data for analytics and business intelligence
- Can also be used for transforming and moving large amounts of data
- Performs extract, transform, and load (ETL) functions



# Amazon EMR

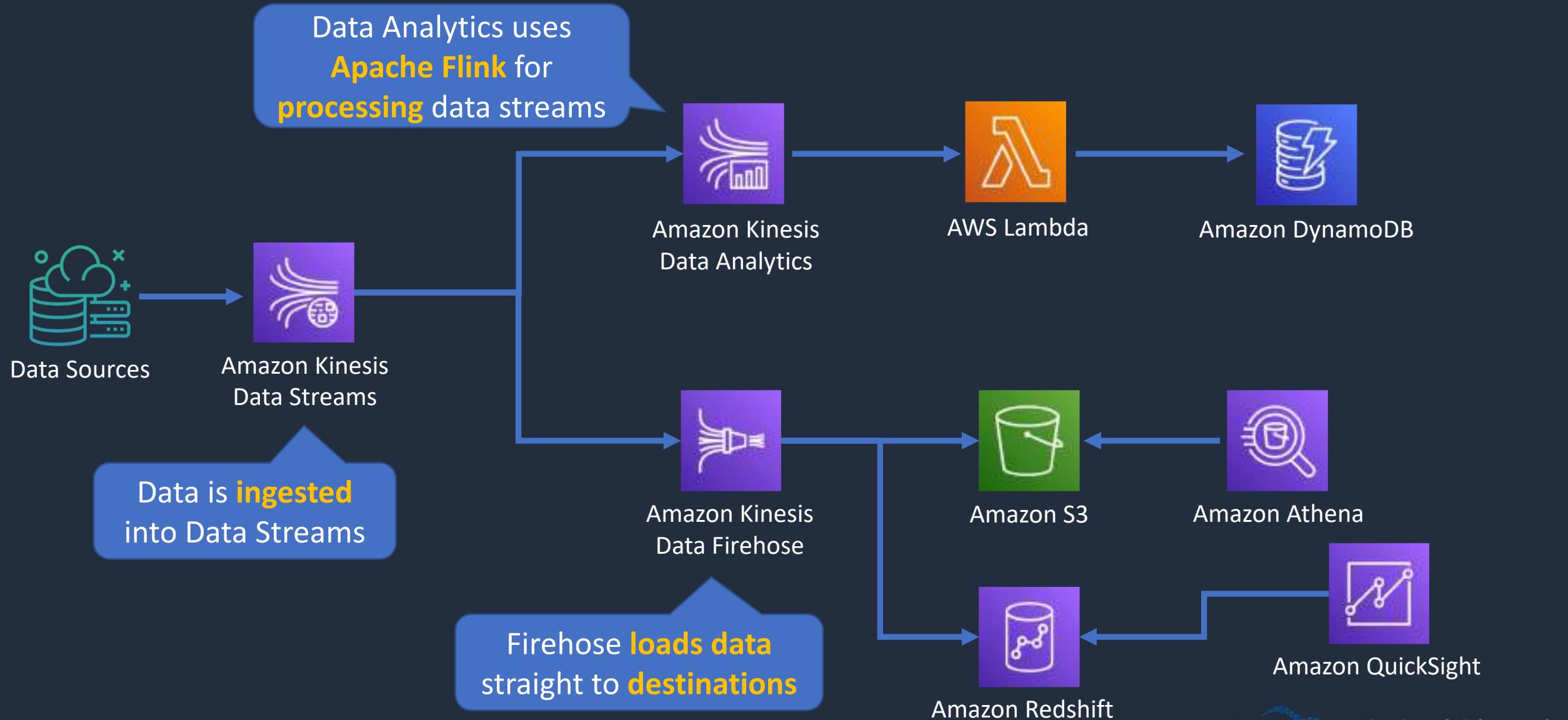


# Amazon Kinesis Core Knowledge





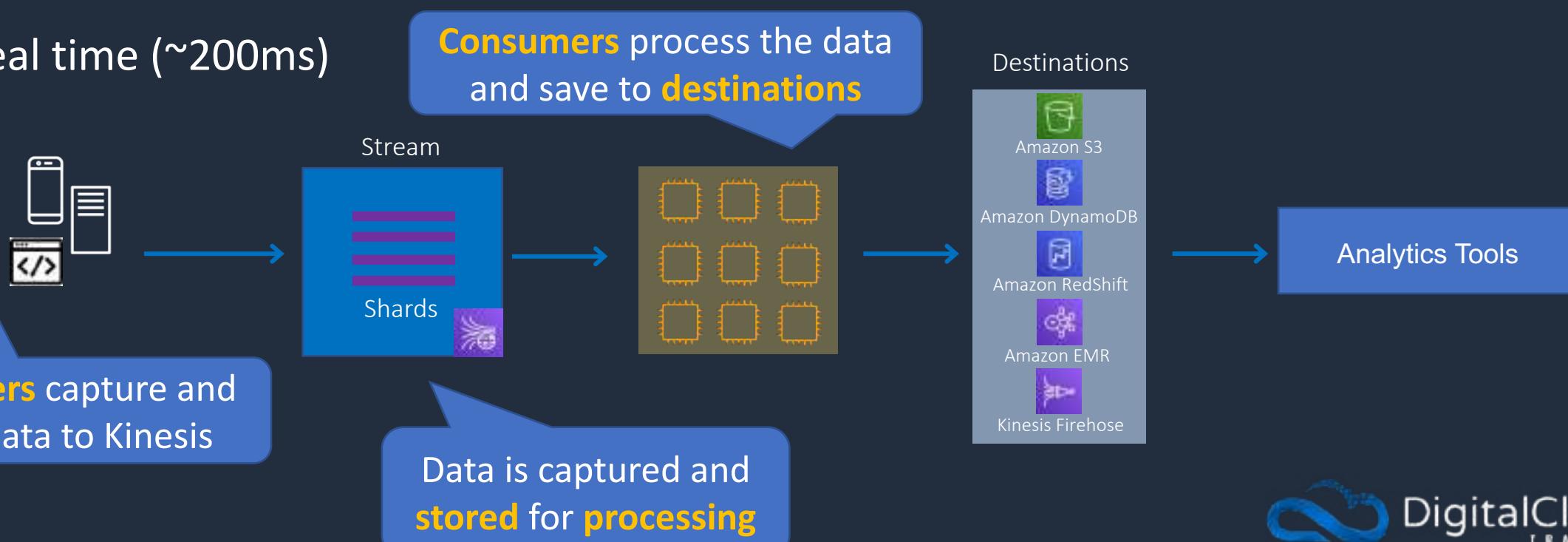
# Amazon Kinesis Services





# Amazon Kinesis Data Streams

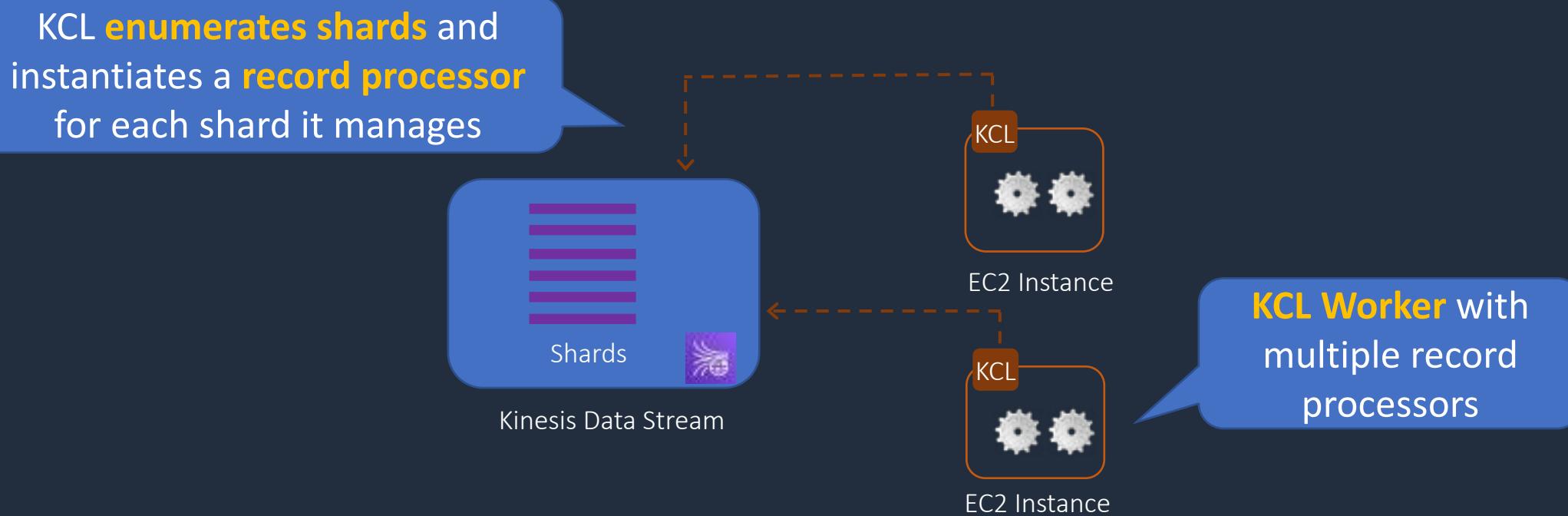
- Producers send data to Kinesis, data is stored in Shards for 24 hours (by default, up to 7 days)
- Consumers then take the data and process it - data can then be saved into another AWS service
- Real time (~200ms)





# Kinesis Client Library (KCL)

- The Kinesis Client Library (KCL) helps you consume and process data from a Kinesis data stream

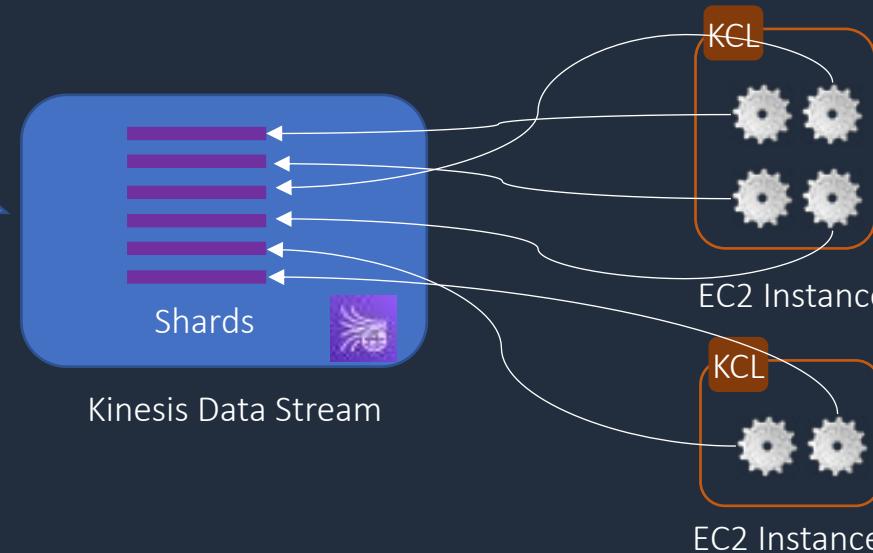




# Kinesis Client Library (KCL)

- Each shard is processed by exactly one KCL worker and has exactly one corresponding record processor
- One worker can process any number of shards, so it's fine if the number of shards exceeds the number of instances

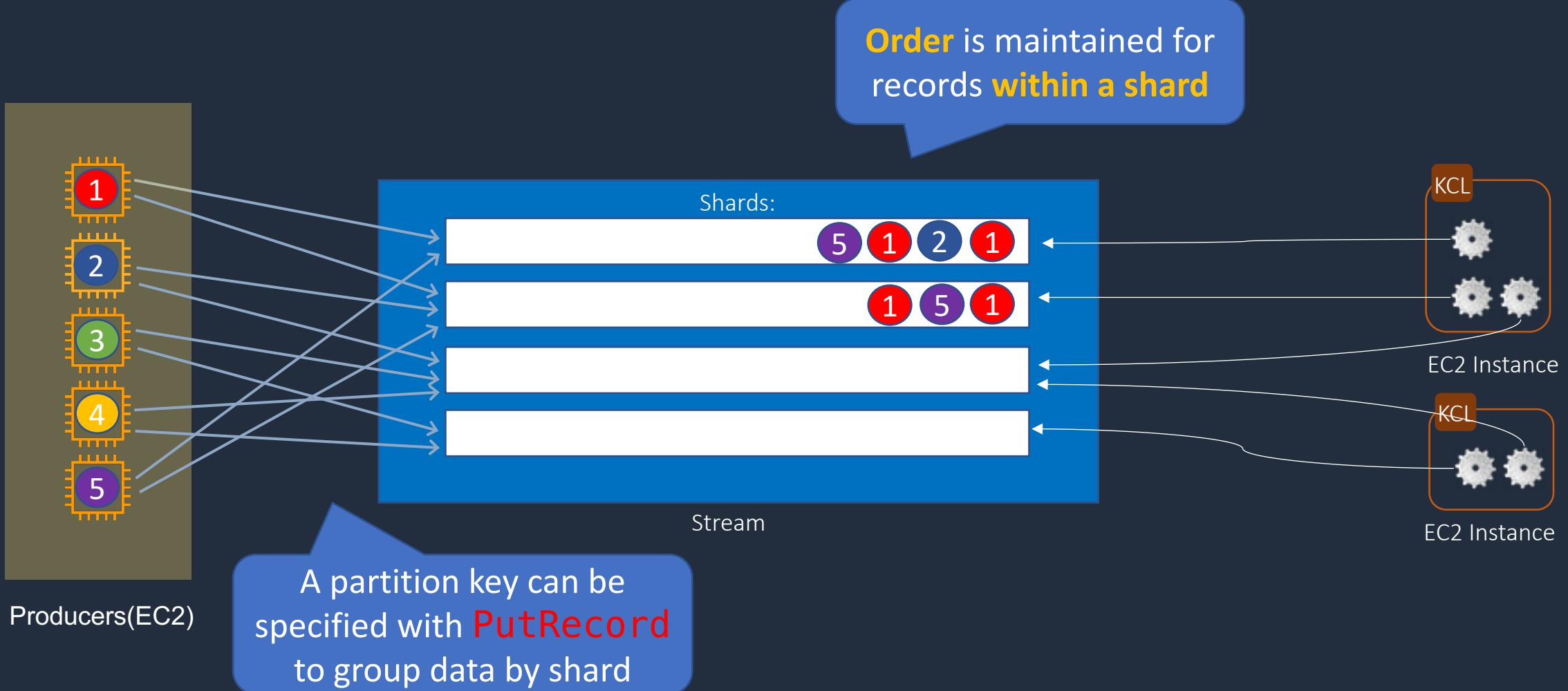
Each shard is **processed** by exactly **one** KCL worker



A record processor maps to exactly one shard



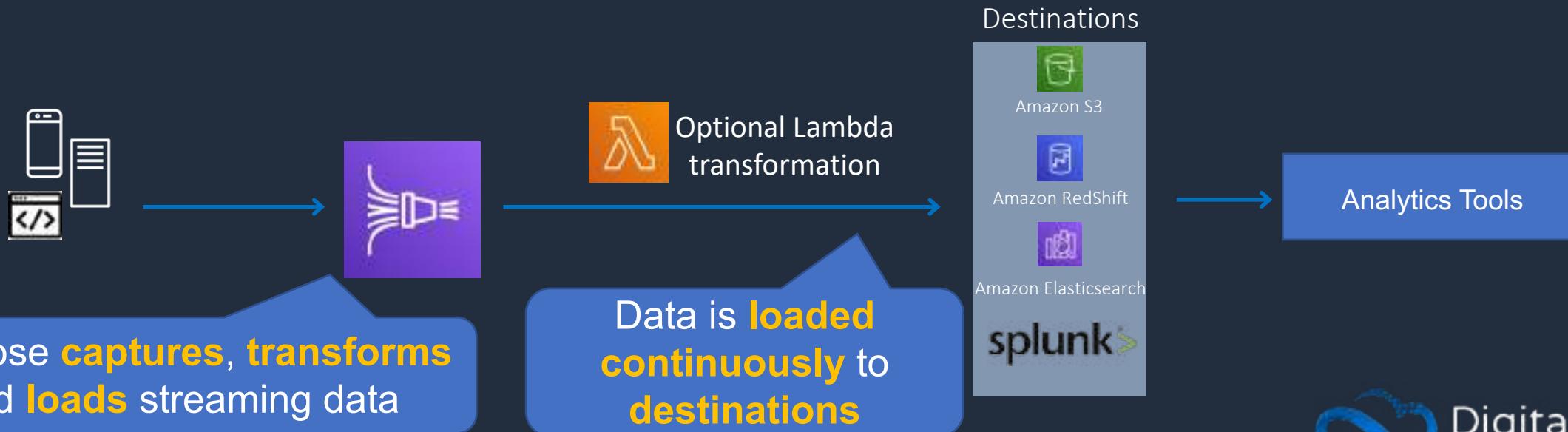
# Amazon Kinesis Data Streams





# Kinesis Data Firehose

- Producers send data to Firehose
- There are no Shards, completely automated (scalability is elastic)
- Firehose data is sent to another AWS service for storing, data can be optionally processed/transformed using AWS Lambda
- Near real-time delivery (~60 seconds latency)





# Kinesis Data Firehose

---

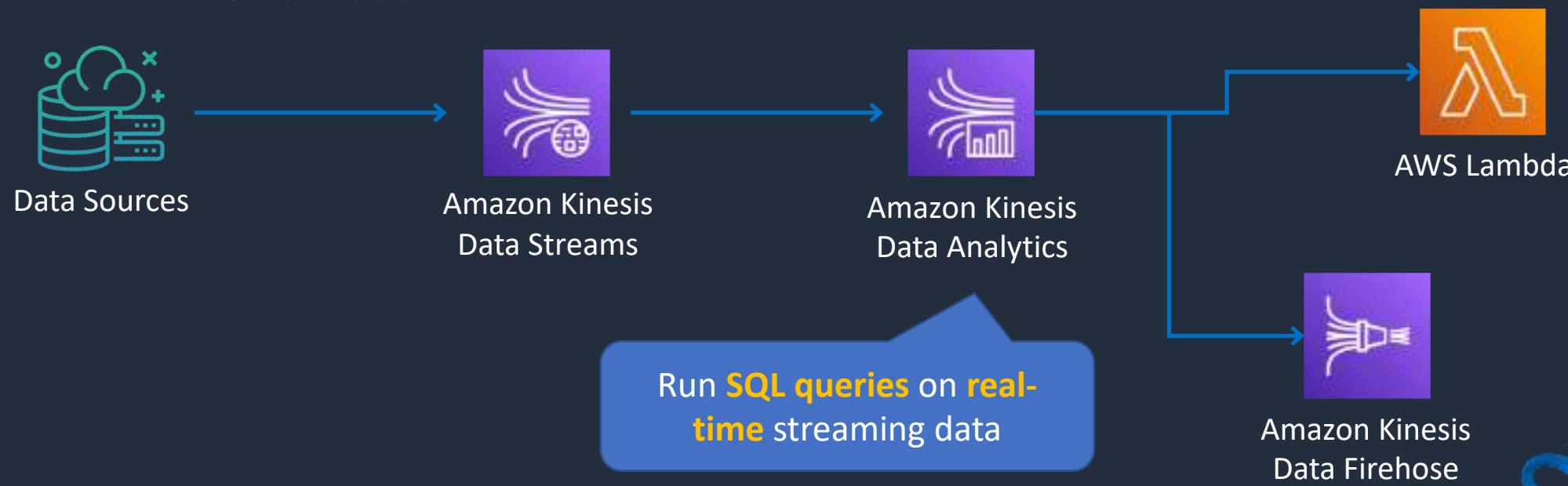
Kinesis Data Firehose destinations:

- RedShift (via an intermediate S3 bucket)
- Elasticsearch
- Amazon S3
- Splunk
- Datadog
- MongoDB
- New Relic
- HTTP Endpoint



# Kinesis Data Analytics

- Provides real-time SQL processing for streaming data
- Provides analytics for data coming in from Kinesis Data Streams and Kinesis Data Firehose
- Destinations can be Kinesis Data Streams, Kinesis Data Firehose, or AWS Lambda



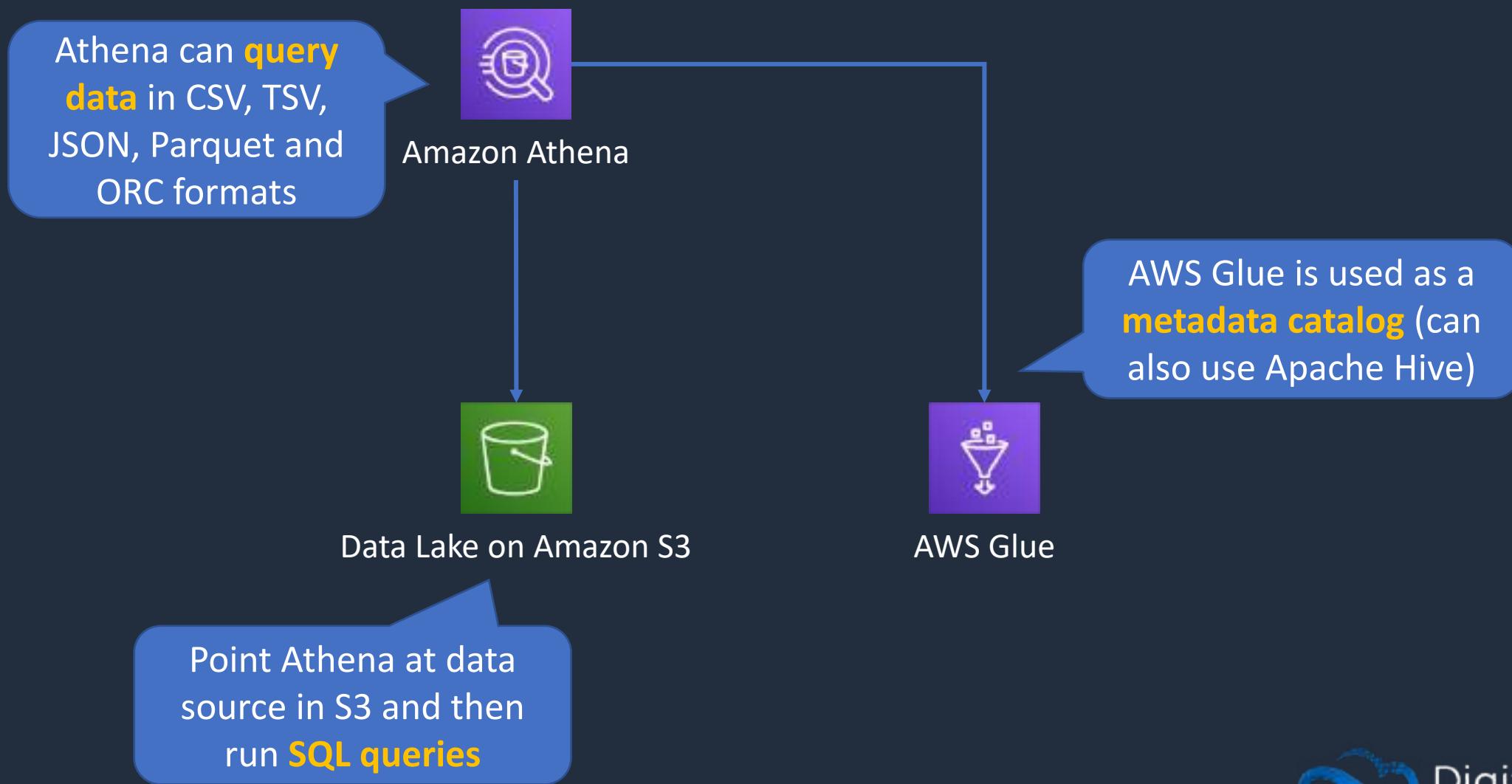
Run **SQL queries** on **real-time** streaming data

# Amazon Athena and AWS Glue



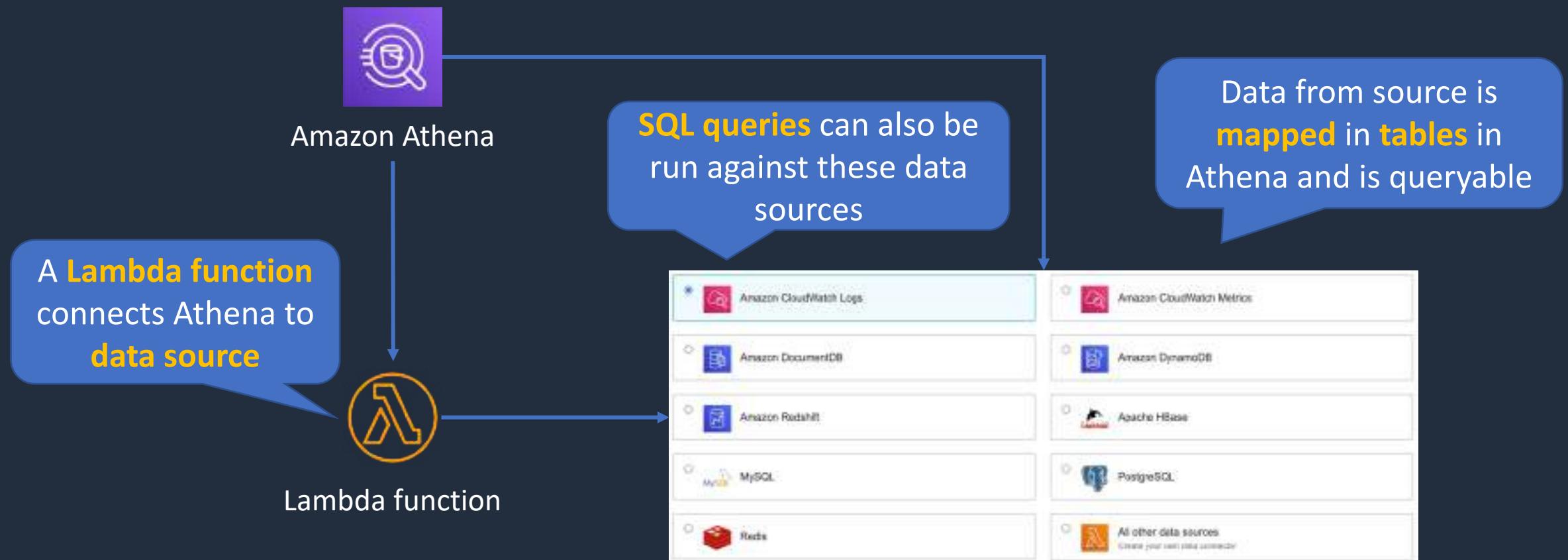


# Amazon Athena and AWS Glue





# Amazon Athena and AWS Glue





# Amazon Athena

---

- Athena queries data in S3 using SQL
- Can be connected to other data sources with Lambda
- Data can be in CSV, TSV, JSON, Parquet and ORC formats
- Uses a managed Data Catalog (AWS Glue) to store information and schemas about the databases and tables



# Optimizing Athena for Performance

---

- **Partition your data**
- **Bucket your data** – bucket the data within a single partition
- **Use Compression** – AWS recommend using either Apache Parquet or Apache ORC
- **Optimize file sizes**
- **Optimize columnar data store generation** – Apache Parquet and Apache ORC are popular columnar data stores
- **Optimize ORDER BY and Optimize GROUP BY**
- **Use approximate functions**
- **Only include the columns that you need**



# AWS Glue

---

- Fully managed extract, transform and load (ETL) service
- Used for preparing data for analytics
- AWS Glue runs the ETL jobs on a fully managed, scale-out Apache Spark environment
- AWS Glue discovers data and stores the associated metadata (e.g. table definition and schema) in the AWS Glue Data Catalog
- Works with data lakes (e.g. data on S3), data warehouses (including RedShift), and data stores (including RDS or EC2 databases)



# AWS Glue

---

---

- You can use a **crawler** to populate the AWS Glue Data Catalog with tables
- A crawler can crawl multiple data stores in a single run
- Upon completion, the crawler creates or updates one or more tables in your Data Catalog.
- ETL jobs that you define in AWS Glue use the Data Catalog tables as sources and targets

# Query S3 ALB Access Logs with Athena



# Architecture Patterns – Databases and Analytics





# Architecture Patterns – Databases & Analytics

## Requirement

Relational database running on MySQL must be migrated to AWS and must be highly available

## Solution

Use Amazon RDS MySQL and configure a Multi-AZ standby node for HA

Amazon RDS DB has high query traffic that is causing performance degradation

Create a Read Replica and configure the application to use the reader endpoint for database queries

Amazon RDS DB is approaching its storage capacity limits and/or is suffering from high write latency

Scale up the DB instance to an instance type that has more storage / CPU



# Architecture Patterns – Databases & Analytics

## Requirement

Amazon RDS database is unencrypted and a cross-Region read replica must be created with encryption

## Solution

Encrypt a snapshot of the main DB and create a new encrypted DB instance from the encrypted snapshot. Create a encrypted cross-Region read replica

Amazon Aurora DB deployed and requires a cross-Region replica

Deploy an Aurora MySQL Replica in the second Region

Amazon Aurora DB deployed and requires a read replica in the same Region with minimal synchronization latency

Deploy an Aurora Replica in the Region in a different Availability Zone



# Architecture Patterns – Databases & Analytics

---

---

## Requirement

Aurora deployed and app in another Region requires read-only access with low latency – synchronization latency must also be minimized

Application and DB migrated to Aurora and requires the ability to write to the DB across multiple nodes

Application requires a session-state data store that provides low-latency

## Solution

Use Aurora Global Database and configure the app in the second Region to use the reader endpoint

Use Aurora Multi-Master for an in-Region multi-master database

Use either Amazon ElastiCache or DynamoDB



# Architecture Patterns – Databases & Analytics

## Requirement

Multi-threaded in-memory datastore required for unstructured data

In-memory datastore required that offers microsecond performance for unstructured data

In-memory datastore required that supports data persistence and high availability

## Solution

Use Amazon ElastiCache Memcached

Use Amazon DynamoDB DAX (DAX)

Use Amazon ElastiCache Redis



# Architecture Patterns – Databases & Analytics

---

---

## Requirement

Serverless database required that supports No-SQL key-value store workload

## Solution

Use Amazon DynamoDB

Serverless database required that supports MySQL or PostgreSQL

Use Amazon Aurora Serverless

Relational database required for a workload with an unknown usage pattern (usage expected to be low and variable)

Use Amazon Aurora Serverless



# Architecture Patterns – Databases & Analytics

## Requirement

Application requires a key-value database that can be written to from multiple AWS Regions

Athena is being used to analyze a large volume of data based on date ranges. Performance must be optimized

Lambda is processing streaming data from API Gateway and is generating a `TooManyRequestsException` as volume increases

## Solution

Use DynamoDB Global Tables

Store data using Apache Hive partitioning with a key based on the data. Use the Apache Parquet and ORC storage formats

Stream the data into a Kinesis Data Stream from API Gateway and process in batches



# Architecture Patterns – Databases & Analytics

---

---

## Requirement

Lambda function is processing streaming data that must be analyzed with SQL

## Solution

Load data into a Kinesis Data Stream and then analyze with Kinesis Data Analytics

Security logs generated by AWS WAF must be sent to a third-party auditing application

Send logs to Kinesis Data Firehose and configure the auditing application using an HTTP endpoint

Real-time streaming data must be stored for future analysis

Ingest data into a Kinesis Data Stream and then use Firehose to load to a data store for later analysis



# Architecture Patterns – Databases & Analytics

---

---

## Requirement

Company runs several production databases and must run complex queries across consolidated data set for business forecasting

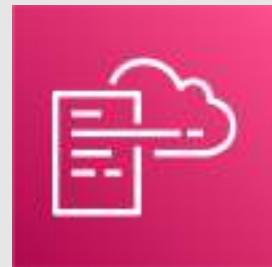
## Solution

Load the data from the OLTP databases into a RedShift data warehouse for OLAP

# SECTION 13

## Deployment and Management

# AWS CloudFormation



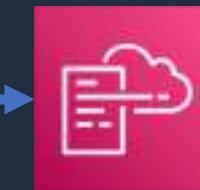


# AWS CloudFormation

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

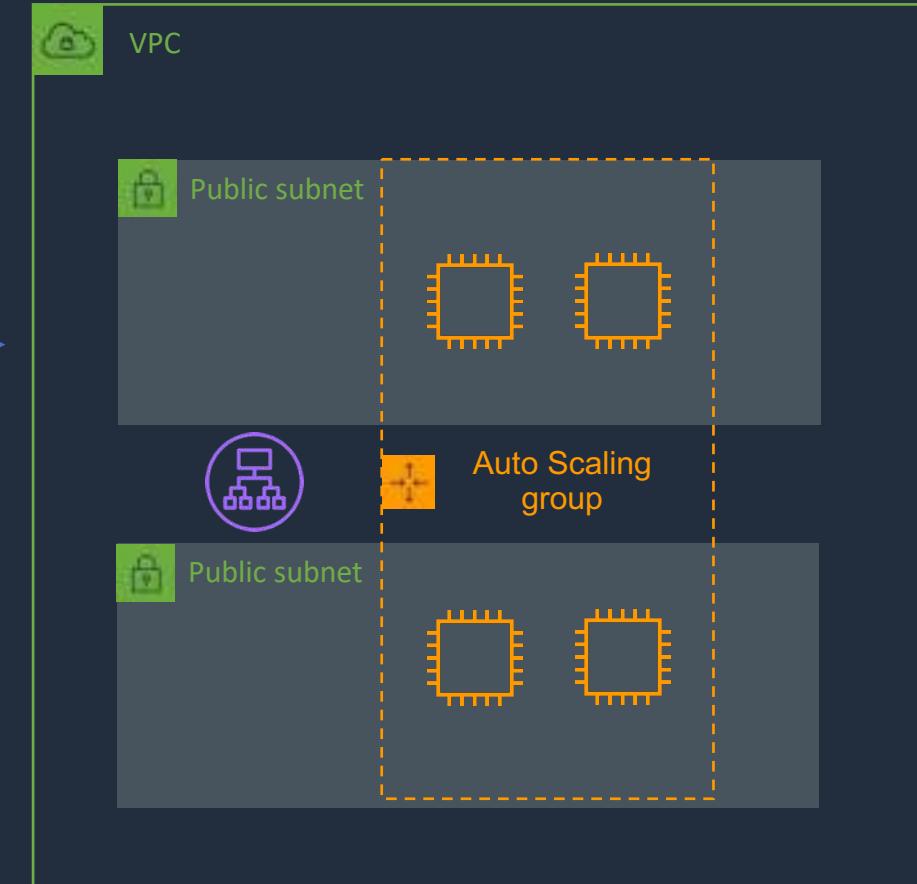


CloudFormation **builds** your infrastructure according to the **template**



AWS CloudFormation

```
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     },
17 }
```





# AWS CloudFormation - Benefits

---

---

- Infrastructure is provisioned consistently, with fewer mistakes (human error)
- Less time and effort than configuring resources manually
- You can use version control and peer review for your CloudFormation templates
- Free to use (you're only charged for the resources provisioned)
- Can be used to manage updates and dependencies
- Can be used to rollback and delete the entire stack as well



# 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

# Deploy Stack and Change Set



# Complex CloudFormation Template

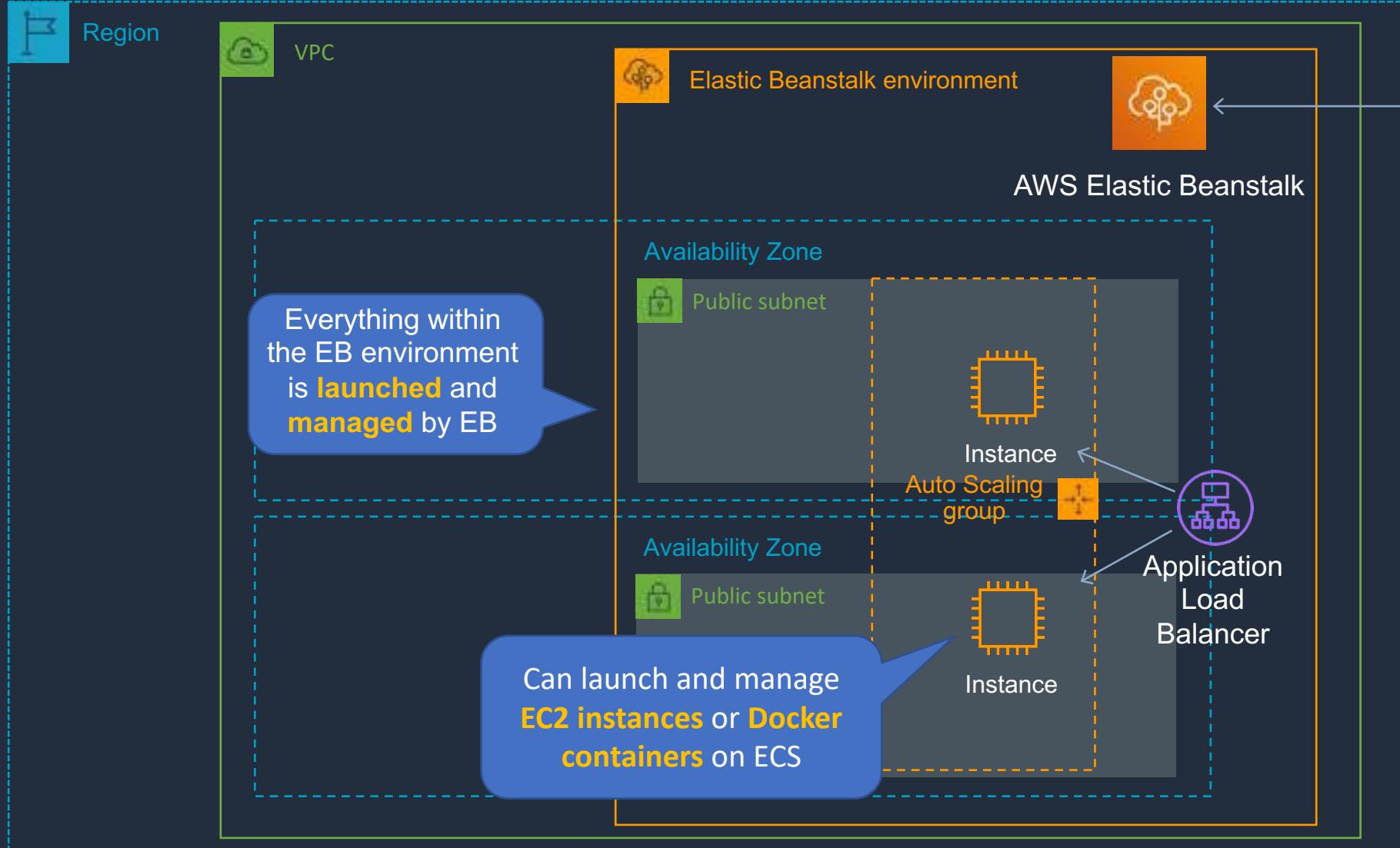


# AWS Elastic Beanstalk





# AWS Elastic Beanstalk



Developer Client



Upload source code in ZIP file



# AWS Elastic Beanstalk

---

---

- Supports Java, .NET, PHP, Node.js, Python, Ruby, Go, and Docker web applications
- Supports the following languages and development stacks:
  - Apache Tomcat for Java applications
  - Apache HTTP Server for PHP applications
  - Apache HTTP Server for Python applications
  - Nginx or Apache HTTP Server for Node.js applications
  - Passenger or Puma for Ruby applications
  - Microsoft IIS 7.5, 8.0, and 8.5 for .NET applications
  - Java SE
  - Docker
  - Go



# 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



# AWS Elastic Beanstalk

## 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

APPLICATION

**Versions** can be applied to any **environment**

- Version 4
- Version 3
- Version 2
- Version 1



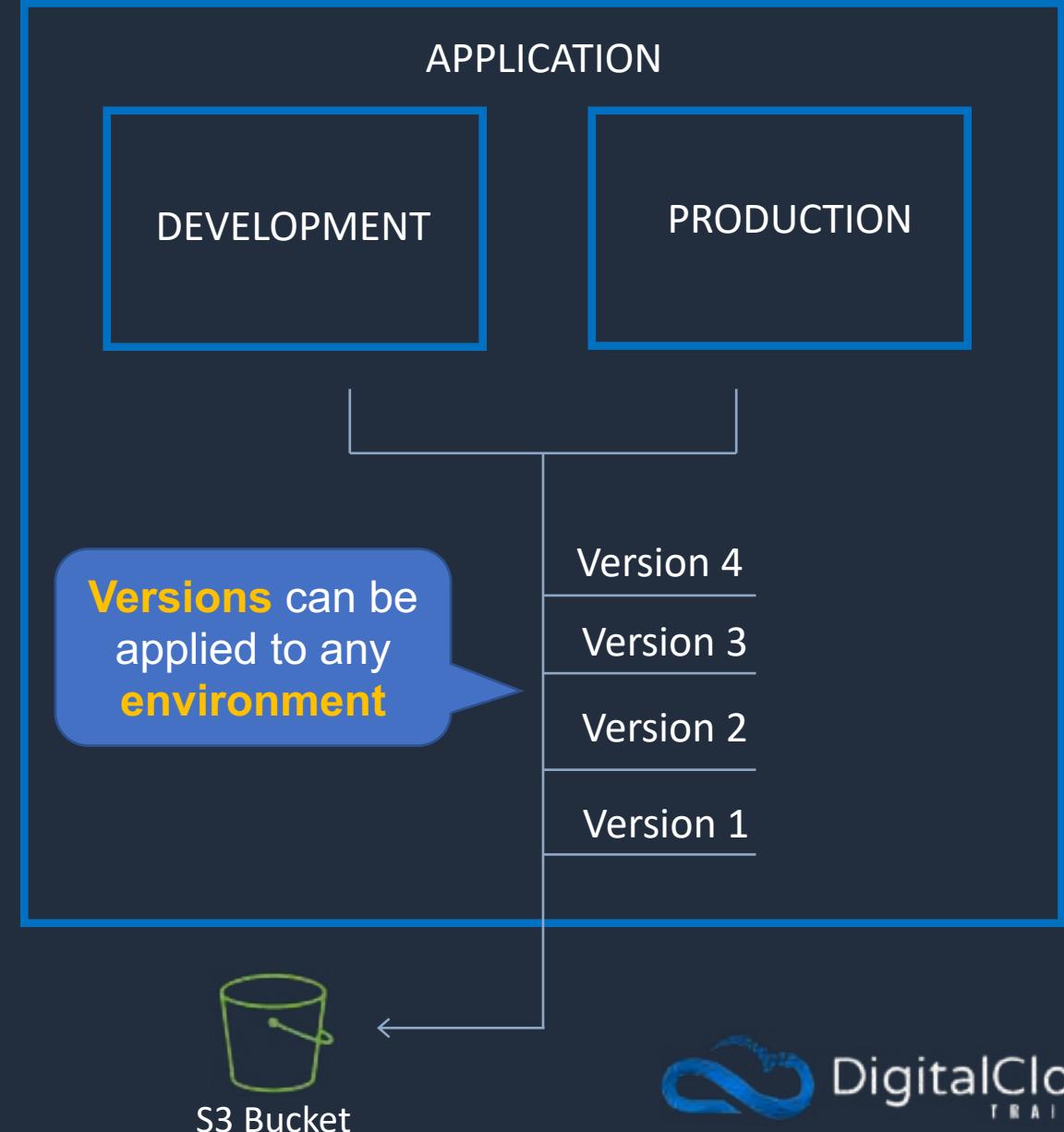
S3 Bucket



# 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

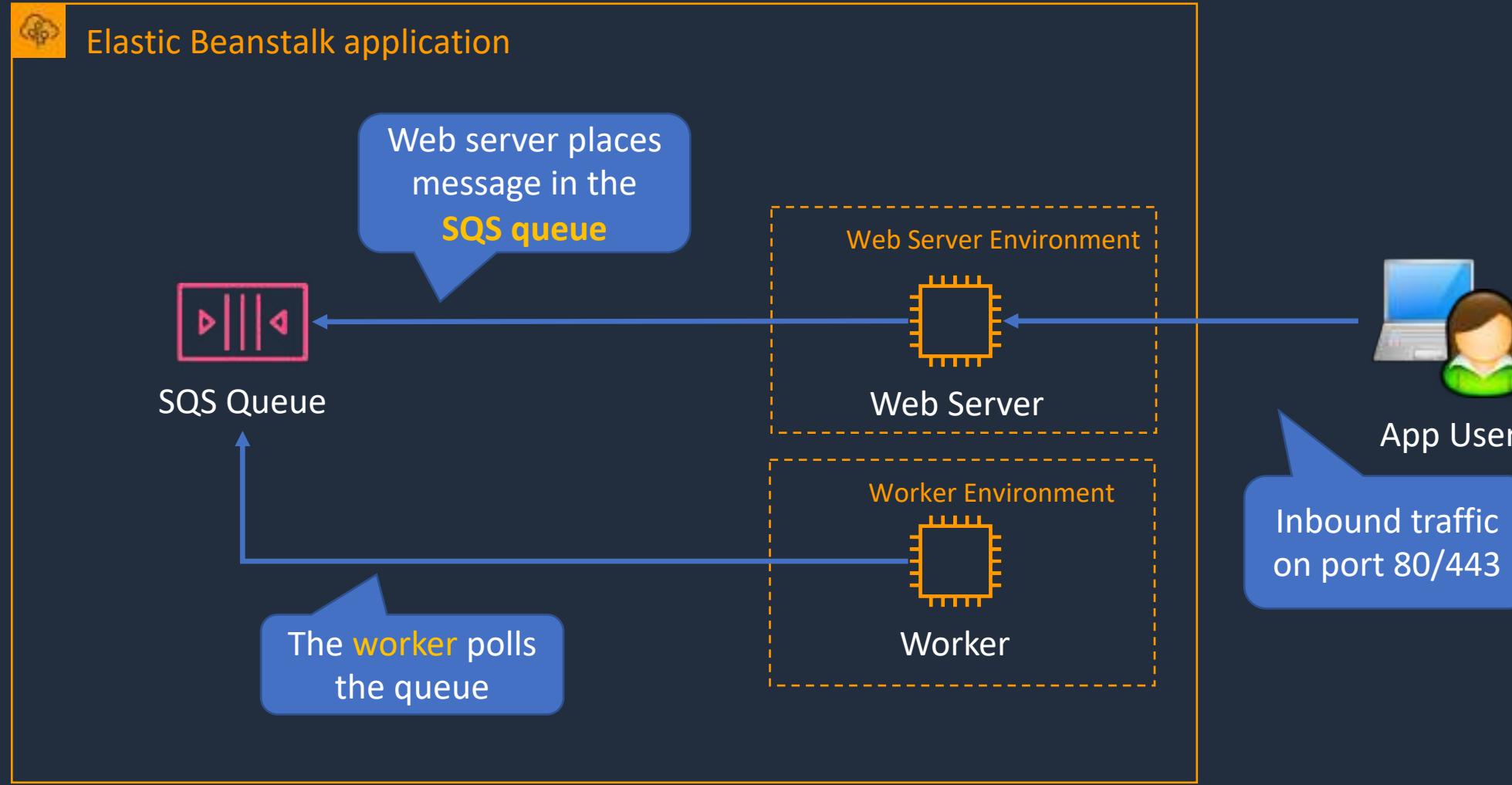
---

---

- **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



# Deploy Elastic Beanstalk Web Server and Worker



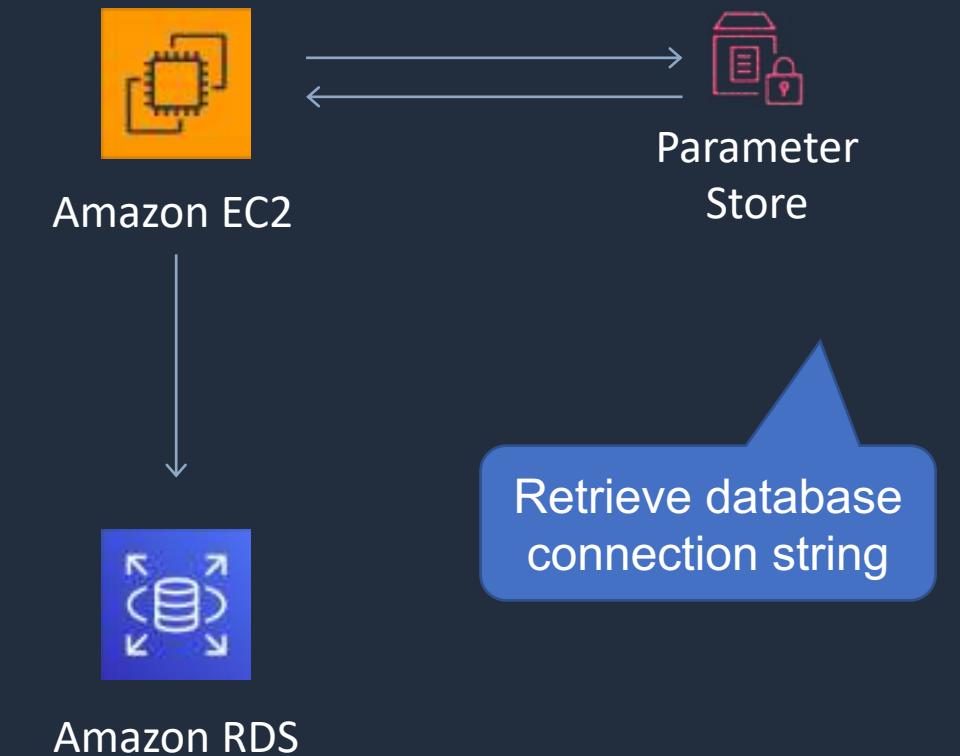
# AWS Systems Manager Parameter Store





# AWS SSM Parameter Store

- Parameter Store provides secure, hierarchical storage for configuration data and secrets
- Highly scalable, available, and durable
- Store data such as passwords, database strings, and license codes as parameter values
- Store values as plaintext (unencrypted data) or ciphertext (encrypted data)
- Reference values by using the unique name that you specified when you created the parameter
- No native rotation of keys (difference with AWS Secrets Manager which does it automatically)



# AWS Config





# AWS Config

---

---

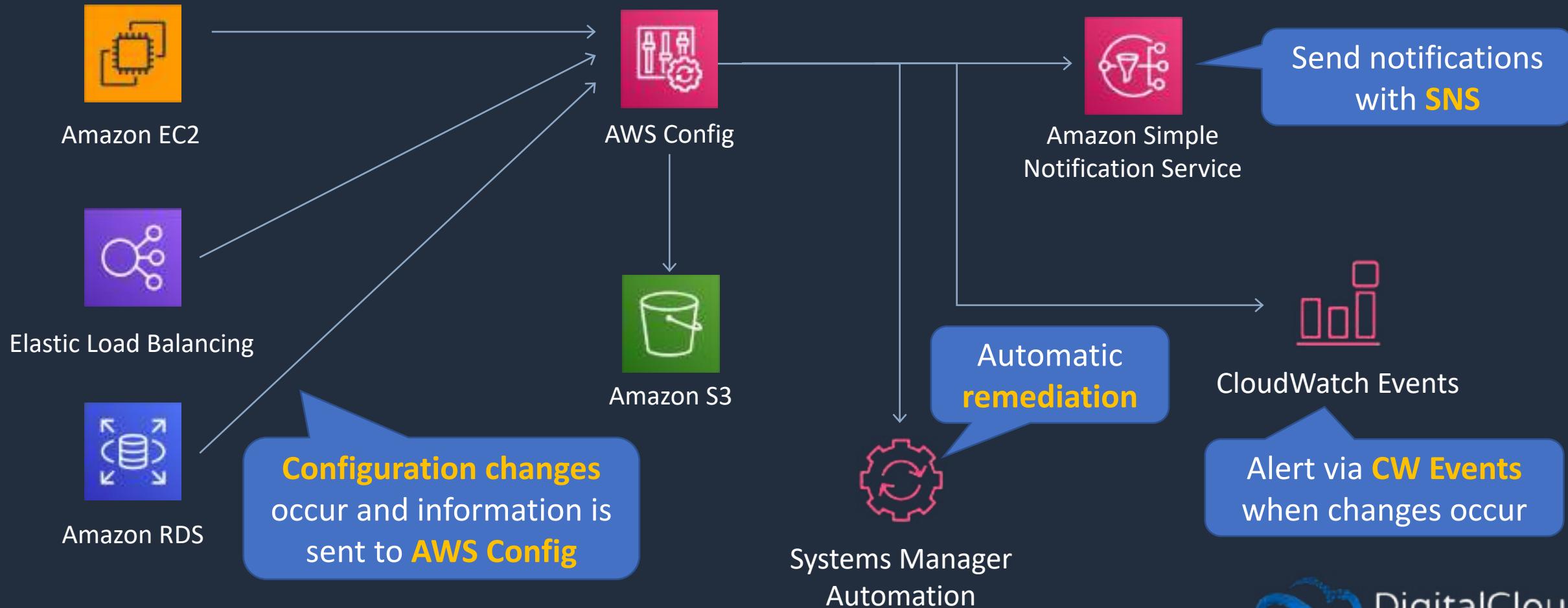
- Evaluate your AWS resource configurations for desired settings
- Get a snapshot of the current configurations of resources that are associated with your AWS account
- Retrieve configurations of resources that exist in your account
- Retrieve historical configurations of one or more resources
- Receive a notification whenever a resource is created, modified, or deleted
- View relationships between resources



# AWS Config

AWS Config evaluates the **configuration** against desired configurations

Example Services:

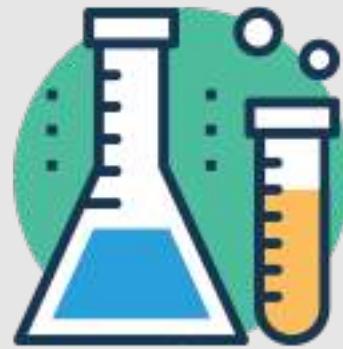




# AWS Config

Example Rule	Description
s3-bucket-server-side-encryption-enabled	Checks that your Amazon S3 bucket either has S3 default encryption enabled or that the S3 bucket policy explicitly denies put-object requests without server side encryption
restricted-ssh	Checks whether security groups that are in use disallow unrestricted incoming SSH traffic
rds-instance-public-access-check	Checks whether the Amazon Relational Database Service (RDS) instances are not publicly accessible
cloudtrail-enabled	Checks whether AWS CloudTrail is enabled in your AWS account

# AWS Config Rule with Remediation



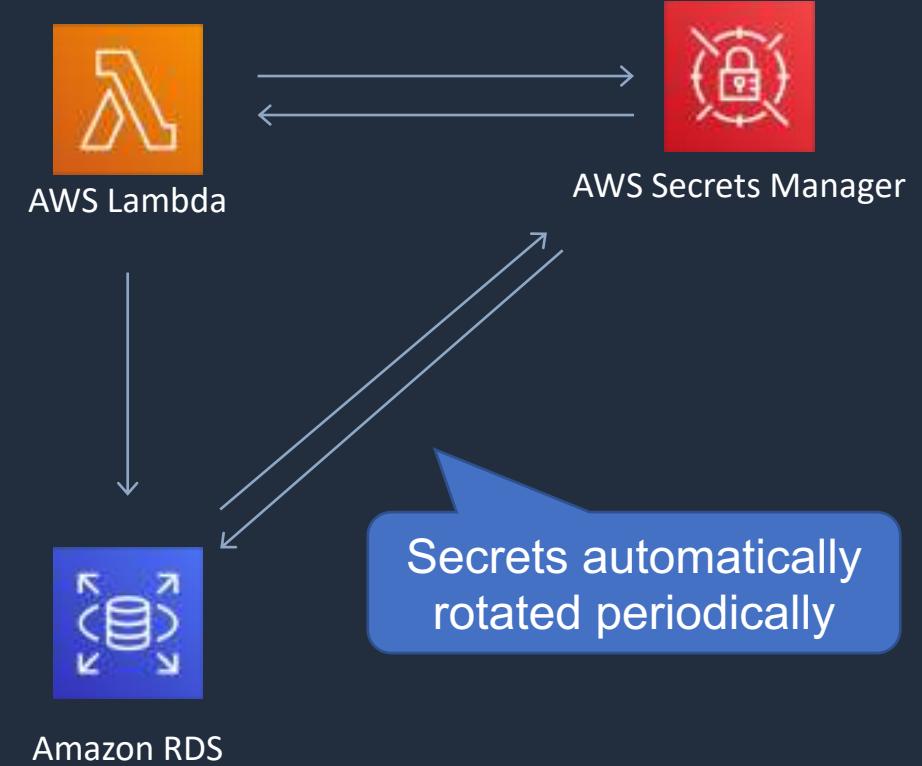
# AWS Secrets Manager





# AWS Secrets Manager

- Stores and rotate secrets safely without the need for code deployments
- Secrets Manager offers automatic rotation of credentials (built-in) for:
  - Amazon RDS (MySQL, PostgreSQL, and Amazon Aurora)
  - Amazon Redshift
  - Amazon DocumentDB
- For other services you can write your own AWS Lambda function for automatic rotation





# AWS Secrets Manager vs SSM Parameter Store

---

	Secrets Manager	SSM Parameter Store
<b>Automatic Key Rotation</b>	Yes, built-in for some services, use Lambda for others	No native key rotation; can use custom Lambda
<b>Key/Value Type</b>	String or Binary (encrypted)	String, StringList, SecureString (encrypted)
<b>Hierarchical Keys</b>	No	Yes
<b>Price</b>	Charges apply per secret	Free for standard, charges for advanced

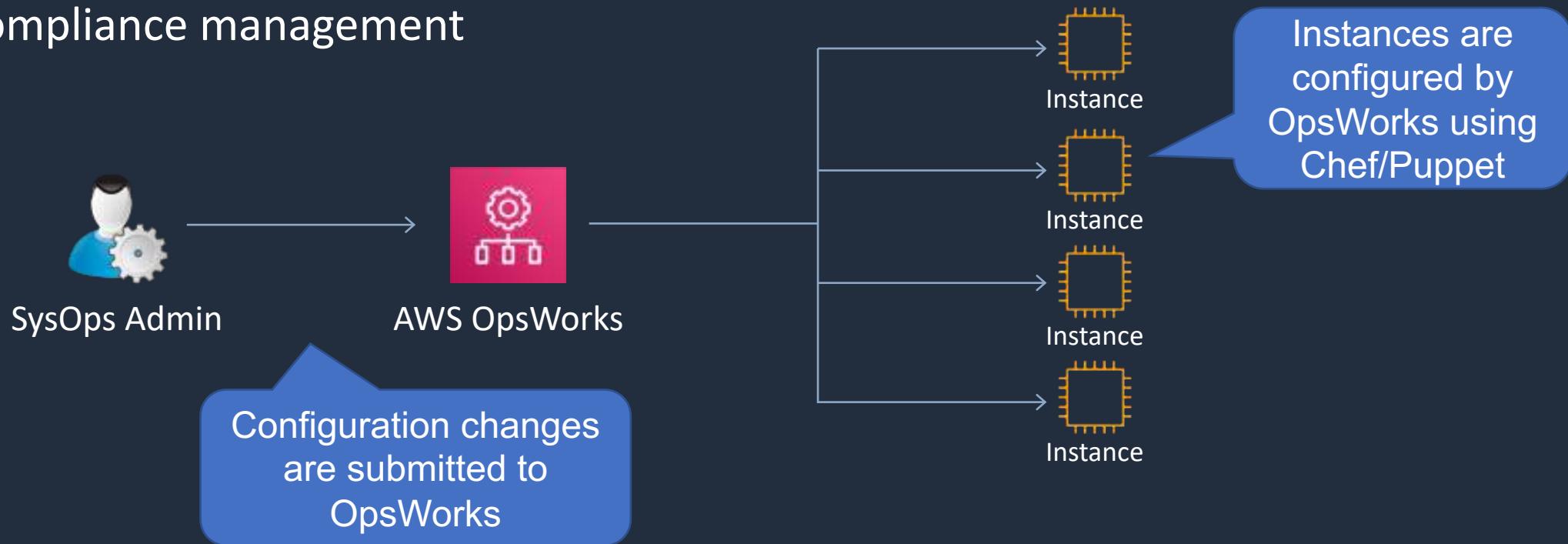
# AWS OpsWorks





# AWS OpsWorks

- AWS OpsWorks is a configuration management service that provides managed instances of Chef and Puppet
- Updates include patching, updating, backup, configuration and compliance management



# AWS Resource Access Manager (RAM)





- Shares resources:
  - Across AWS accounts
  - Within AWS Organizations or OUs
  - IAM roles and IAM users
- Resource shares are created with:
  - The AWS RAM Console
  - AWS RAM APIs
  - AWS CLI
  - AWS SDKs



# AWS RAM

---

RAM can be used to share:

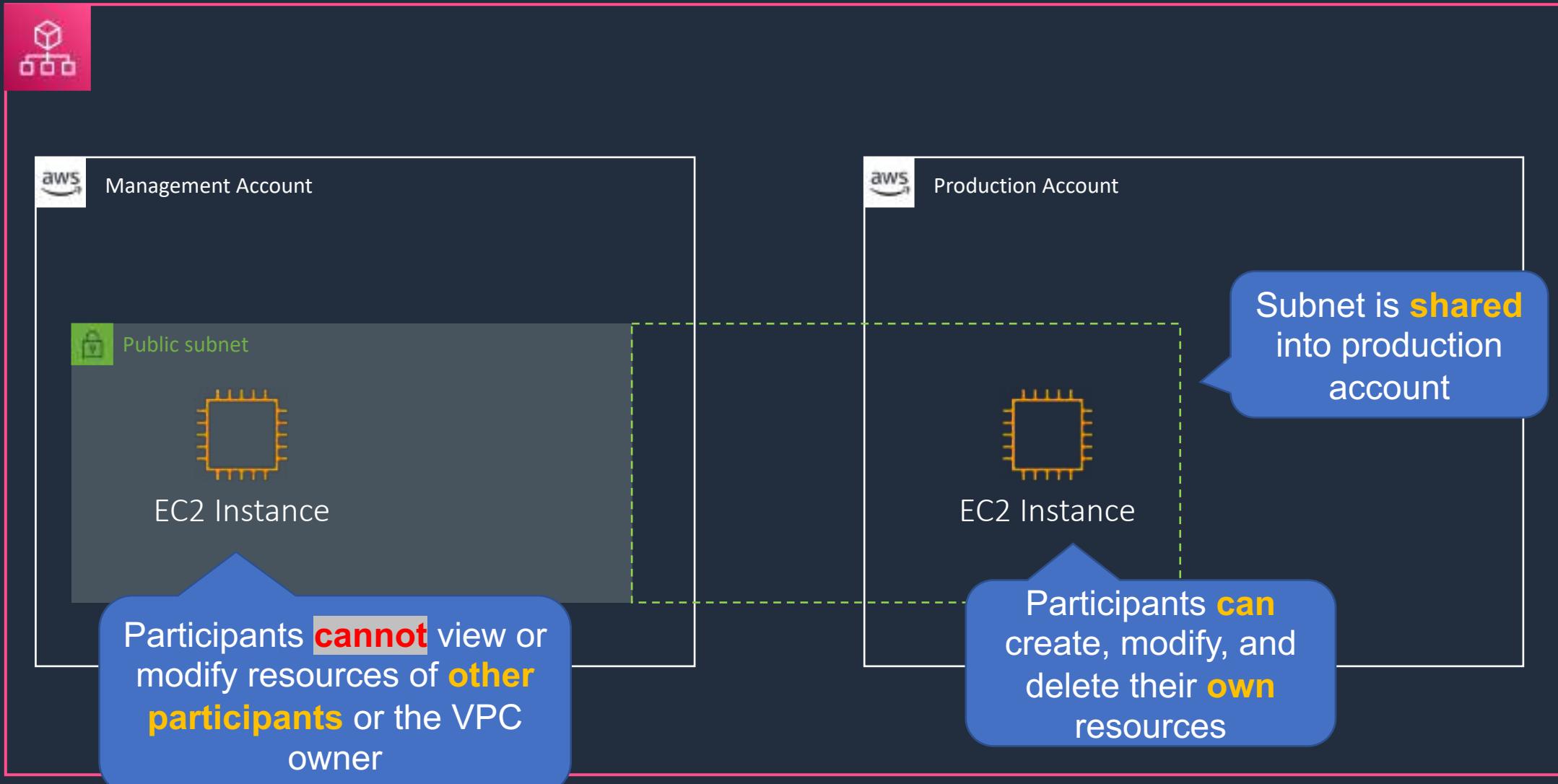
- AWS App Mesh
- Amazon Aurora
- AWS Certificate Manager Private Certificate Authority
- AWS CodeBuild
- Amazon EC2
- EC2 Image Builder
- AWS Glue
- AWS License Manager
- AWS Network Firewall
- AWS Outposts
- Amazon S3 on Outposts
- AWS Resource Groups
- Amazon Route 53
- AWS Systems Manager Incident Manager
- Amazon VPC

# Share a Subnet Across Accounts





# AWS RAM



# Architecture Patterns – Deployment and Management





# Architecture Patterns – Deployment and Management

## Requirement

Application must authenticate to Amazon Aurora and need to securely store credentials. Automatic credential rotation is required on a monthly basis

## Solution

Use AWS Secrets Manager to store the credentials. Update the app to retrieve credentials from Secrets Manager; enable automatic monthly rotation

Company currently uses Chef cookbooks to manage infrastructure and is moving to the cloud. Need to minimize migration complexity

Use AWS OpsWorks for Chef Automate

Need a managed environment for running a simple web application. App processes incoming data which can take several minutes per task

Use an Elastic Beanstalk environment with a web server for the app front-end and a decoupled worker tier for the long running process



# Architecture Patterns – Deployment and Management

## Requirement

Systems integrator deploys standardized Amazon VPC configuration for many customers. Need to increase efficiency and reduce errors

Company has experienced issues rolling out updates to a CloudFormation Stack and needs to preview changes before the next update

Manager wishes to monitor and enforce configuration compliance for AWS resources including S3 buckets and security groups

## Solution

Create an AWS CloudFormation template based on the standard config and use CloudFormation to deploy to new customers

Preview the change by creating a change set with the updated template before updating the stack

Use AWS Config to create rules to monitor compliance and use auto remediation to enforce compliance

# SECTION 14

## Monitoring, Logging, and Auditing

# Amazon CloudWatch Overview





# Amazon CloudWatch

**CloudWatch** is used for performance monitoring, alarms, log collection and automated actions

## Use cases / benefits include:

- Collect performance metrics from AWS and on-premises systems
- Automate responses to operational changes
- Improve operational performance and resource optimization
- Derive actionable insights from logs
- Get operational visibility and insight



# Amazon CloudWatch

---

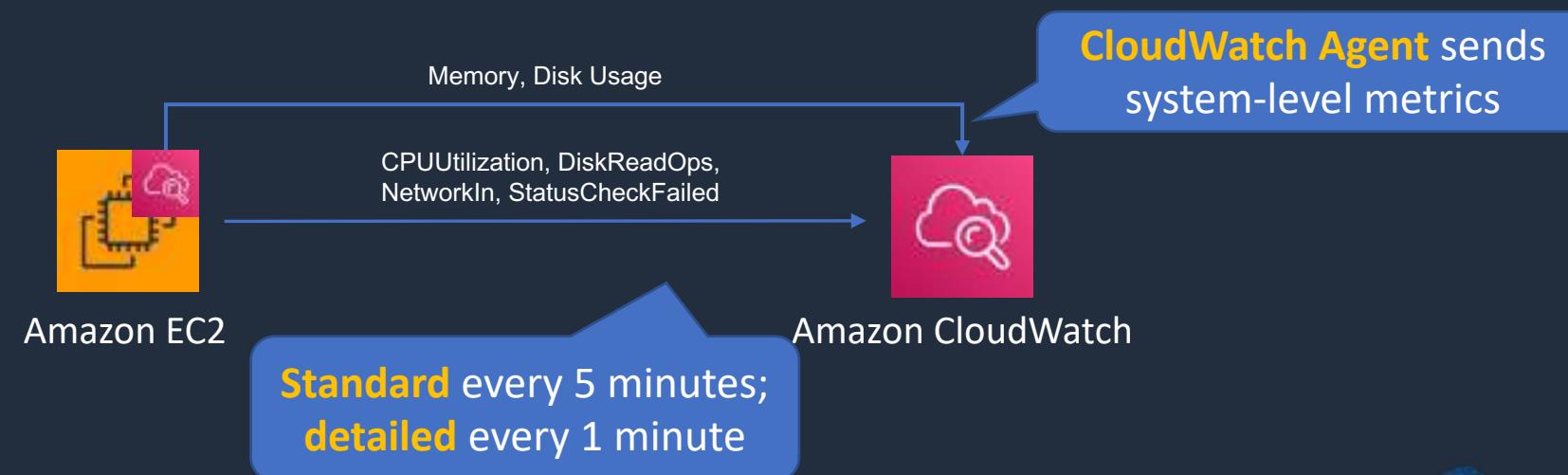
## CloudWatch Core Features:

- **CloudWatch Metrics** – services send time-ordered data points to CloudWatch
- **CloudWatch Alarms** – monitor metrics and initiate actions
- **CloudWatch Logs** – centralized collection of system and application logs
- **CloudWatch Events** – stream of system events describing changes to AWS resources and can trigger actions



# Amazon CloudWatch Metrics

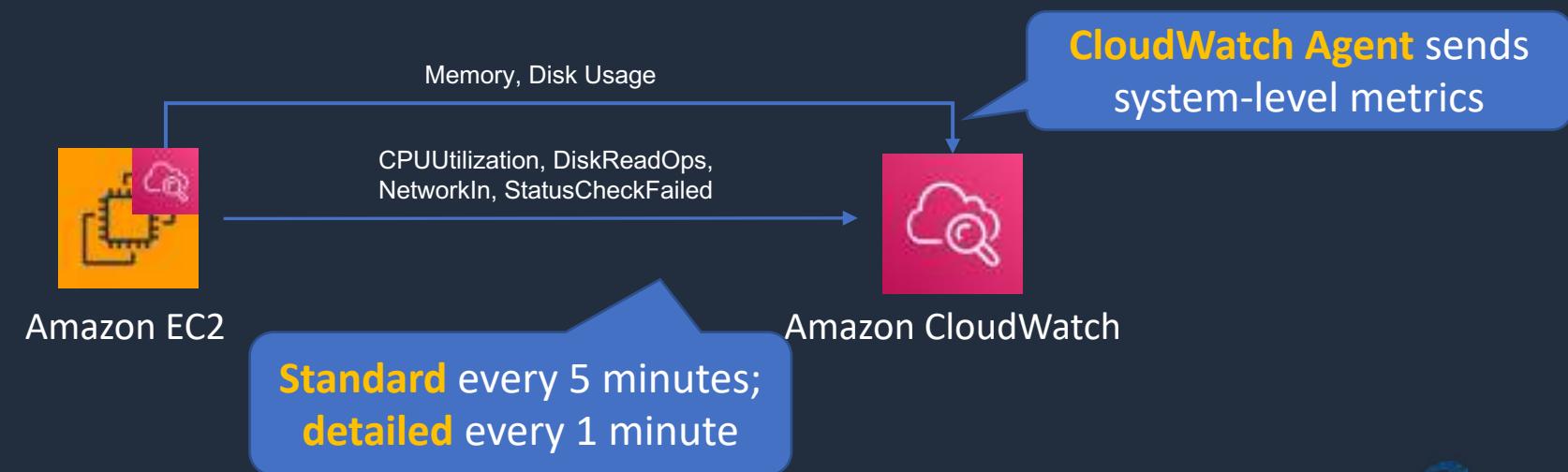
- Metrics are sent to CloudWatch for many AWS services
- EC2 metrics are sent every **5 minutes** by default (free)
- Detailed EC2 monitoring sends every **1 minute** (chargeable)
- Unified CloudWatch Agent sends system-level metrics for EC2 and on-premises servers
- System-level metrics include memory and disk usage





# Amazon CloudWatch Metrics

- Can publish custom metrics using CLI or API
- Custom metrics are one of the following resolutions:
  - **Standard resolution** – data having a one-minute granularity
  - **High resolution** – data at a granularity of one second
- AWS metrics are standard resolution by default

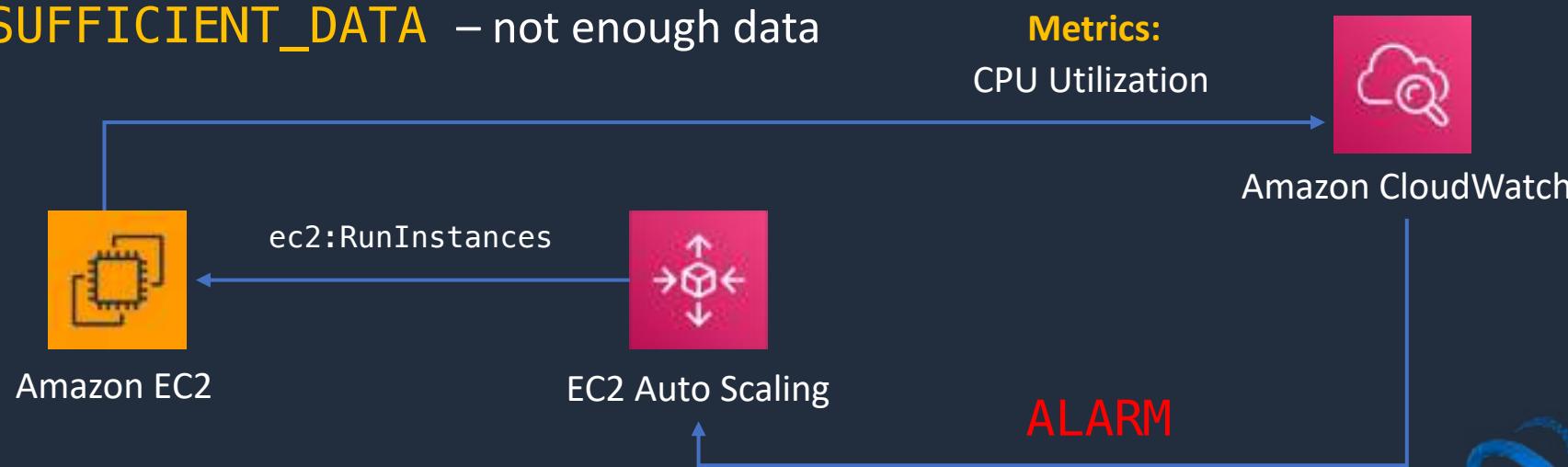




# Amazon CloudWatch Alarms

Two types of alarms

- **Metric alarm** – performs one or more actions based on a single metric
- **Composite alarm** – uses a rule expression and takes into account multiple alarms
- Metric alarm states:
  - **OK** – Metric is within a threshold
  - **ALARM** – Metric is outside a threshold
  - **INSUFFICIENT\_DATA** – not enough data



# Create a Custom Metric



# Create a CloudWatch Alarm



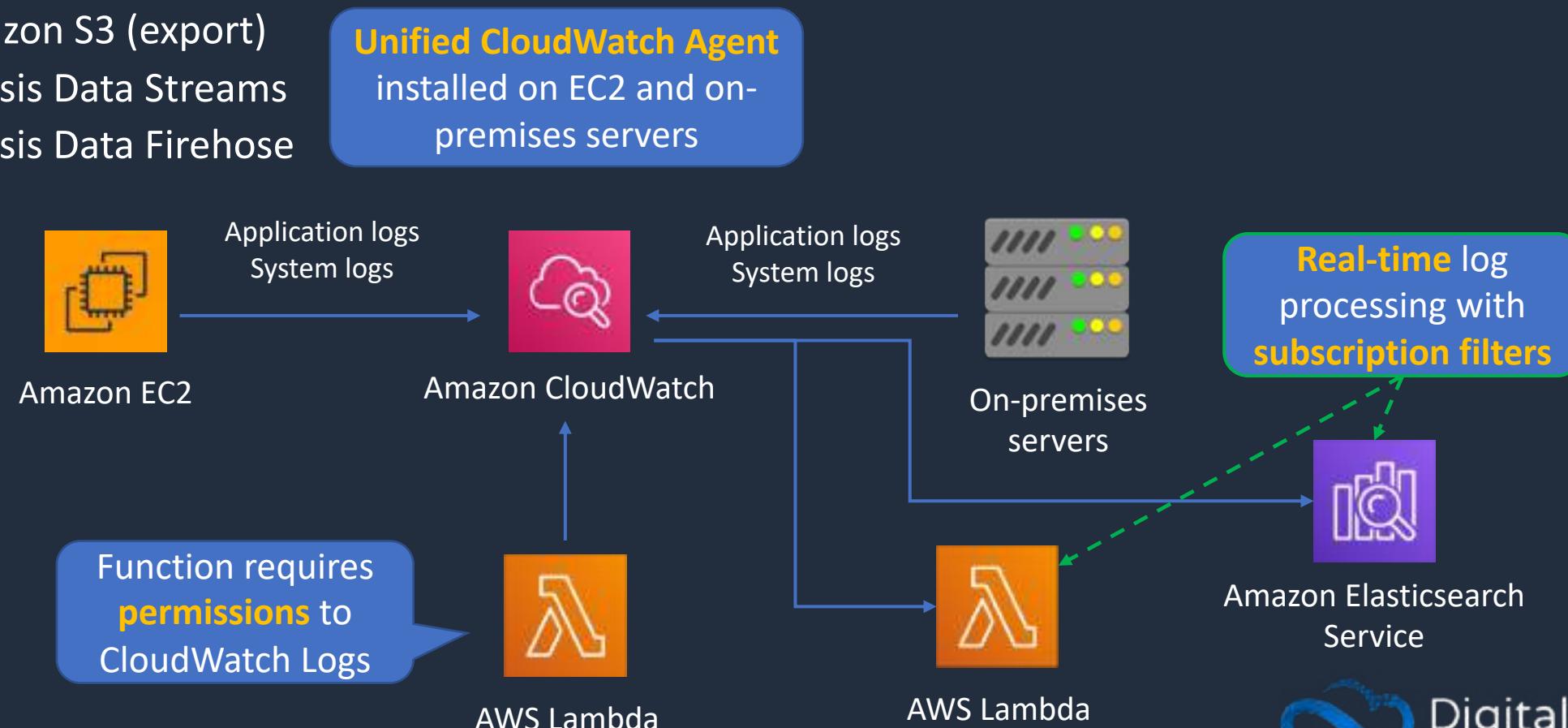
# Amazon CloudWatch Logs



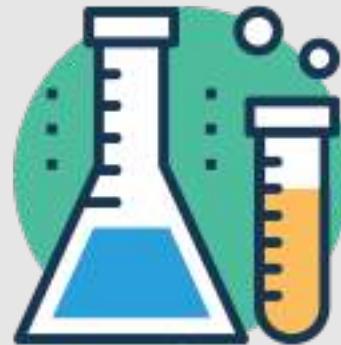


# Amazon CloudWatch Logs

- Gather application and system logs in CloudWatch
- Defined expiration policies and KMS encryption
- Send to:
  - Amazon S3 (export)
  - Kinesis Data Streams
  - Kinesis Data Firehose



# Review CloudWatch Logs



# The Unified CloudWatch Agent





# The Unified CloudWatch Agent

The unified CloudWatch agent enables you to do the following:

- Collect internal system-level metrics from Amazon **EC2 instances** across operating systems
- Collect system-level metrics from **on-premises servers**
- Retrieve custom metrics from your applications or services using the StatsD and collectd protocols
- Collect logs from Amazon EC2 instances and on-premises servers (Windows / Linux)

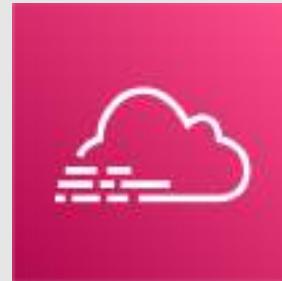


# The Unified CloudWatch Agent

---

- Agent must be installed on the server
- Can be installed on:
  - Amazon EC2 instances
  - On-premises servers
  - Linux, Windows Server, or macOS

# AWS CloudTrail





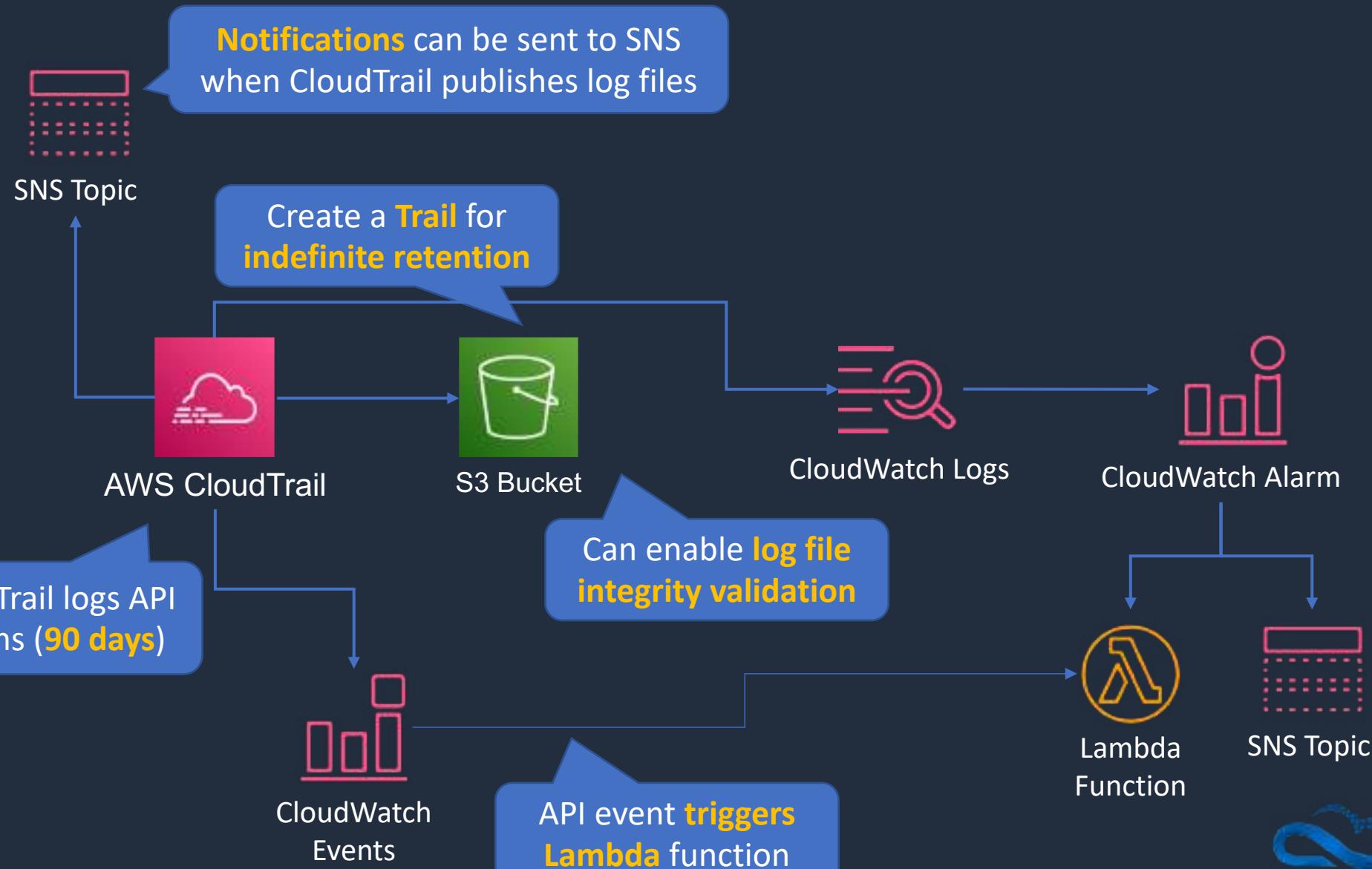
# AWS CloudTrail

---

- CloudTrail logs **API activity** for auditing
- By default, management events are logged and retained for 90 days
- A **CloudTrail Trail** logs any events to S3 for indefinite retention
- Trail can be within Region or all Regions
- CloudWatch Events can triggered based on API calls in CloudTrail
- Events can be streamed to CloudWatch Logs



# AWS CloudTrail





# CloudTrail – Types of Events

---

- **Management events** provide information about management operations that are performed on resources in your AWS account
- **Data events** provide information about the resource operations performed on or in a resource
- **Insights events** identify and respond to unusual activity associated with write API calls by continuously analyzing CloudTrail management events

# Create a CloudTrail Trail

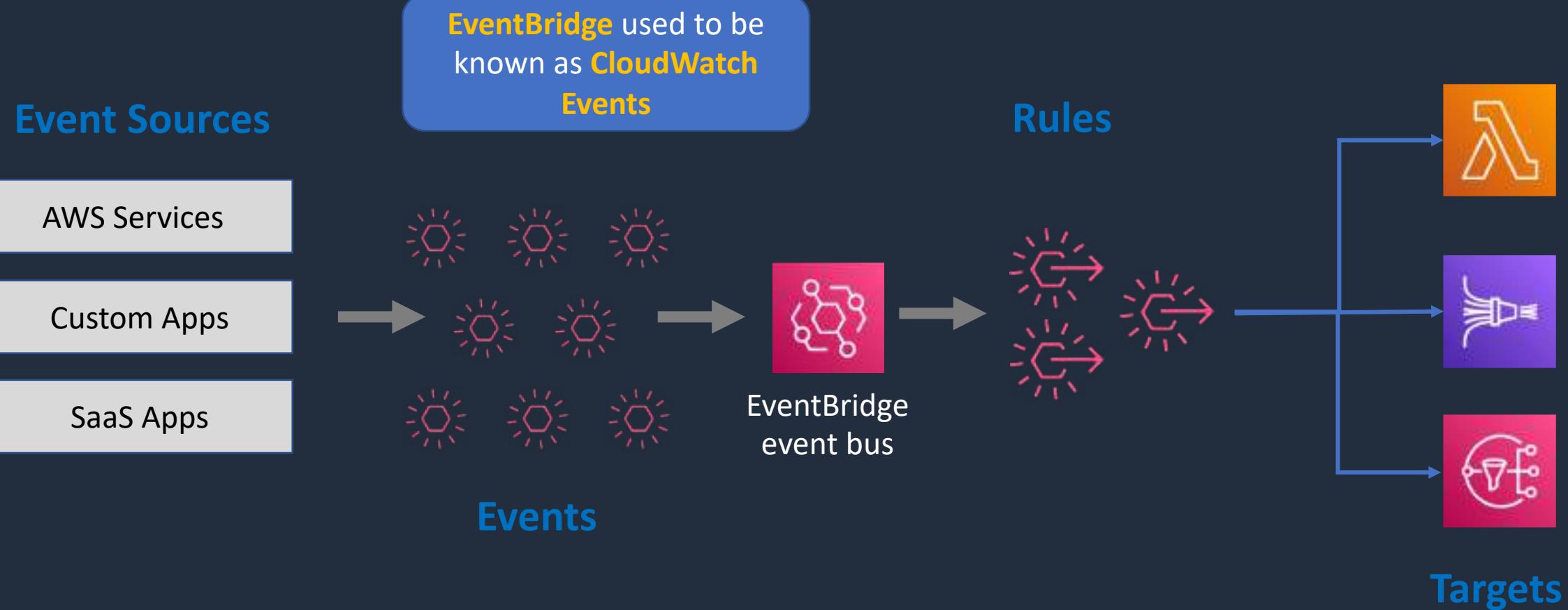


# Amazon EventBridge (Refresher)





# Amazon CloudWatch Events / EventBridge

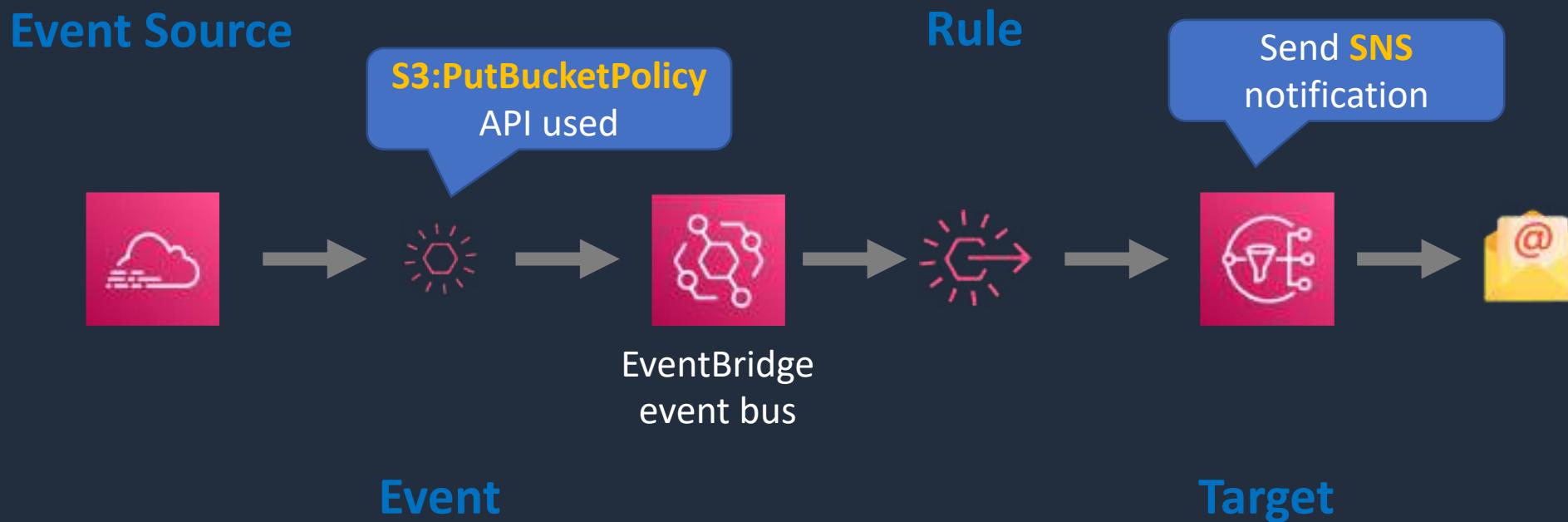


# Create EventBridge rule for CloudTrail API calls





# Amazon EventBridge Example 2



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

# Architecture Patterns - Monitoring, Logging and Auditing





# Architecture Patterns – Monitoring, Logging and Auditing

---

## Requirement

Need to stream logs from Amazon EC2 instances in an Auto Scaling Group

Need to collect metrics from EC2 instances with a 1 second granularity

The application logs from on-premises servers must be processed by AWS Lambda in real time

## Solution

Install the unified CloudWatch Agent and collect log files in Amazon CloudWatch

Create a custom metric with high resolution

Install the unified CloudWatch Agent on the servers and use a subscription filter in CloudWatch to connect to a Lambda function



# Architecture Patterns – Monitoring, Logging and Auditing

---

## Requirement

CloudWatch Logs entries must be transformed with Lambda and then loaded into Amazon S3

Access auditing must be enabled, and records must be stored for a minimum of 5 years. Any attempts to modify the log files must be identified

API activity must be captured from all accounts in an Organization. Admins in member accounts must not be able to modify or delete the trail

## Solution

Configure a Kinesis Firehose destination, transform with Lambda and then load into an S3 bucket

Create a trail in CloudTrail that stores the data in an S3 bucket and enable log file integrity validation

Create an Organization trail in AWS CloudTrail that applies to the management account and all member accounts



# Architecture Patterns – Monitoring, Logging and Auditing

---

---

## Requirement

Company requires API events that involve the root user account to generate a notification

## Solution

Create a CloudTrail trail and an EventBridge rule that looks for API events that involve root and configure an SNS notification

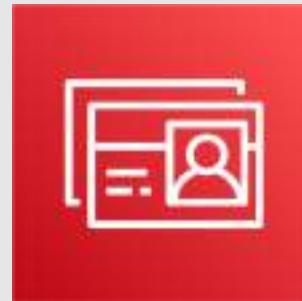
For compliance reasons all S3 buckets must have encryption enabled and any non-compliant buckets must be auto remediated

Use AWS Config to check the encryption status of the buckets and use auto remediation to enable encryption as required

# SECTION 15

## Security in the Cloud

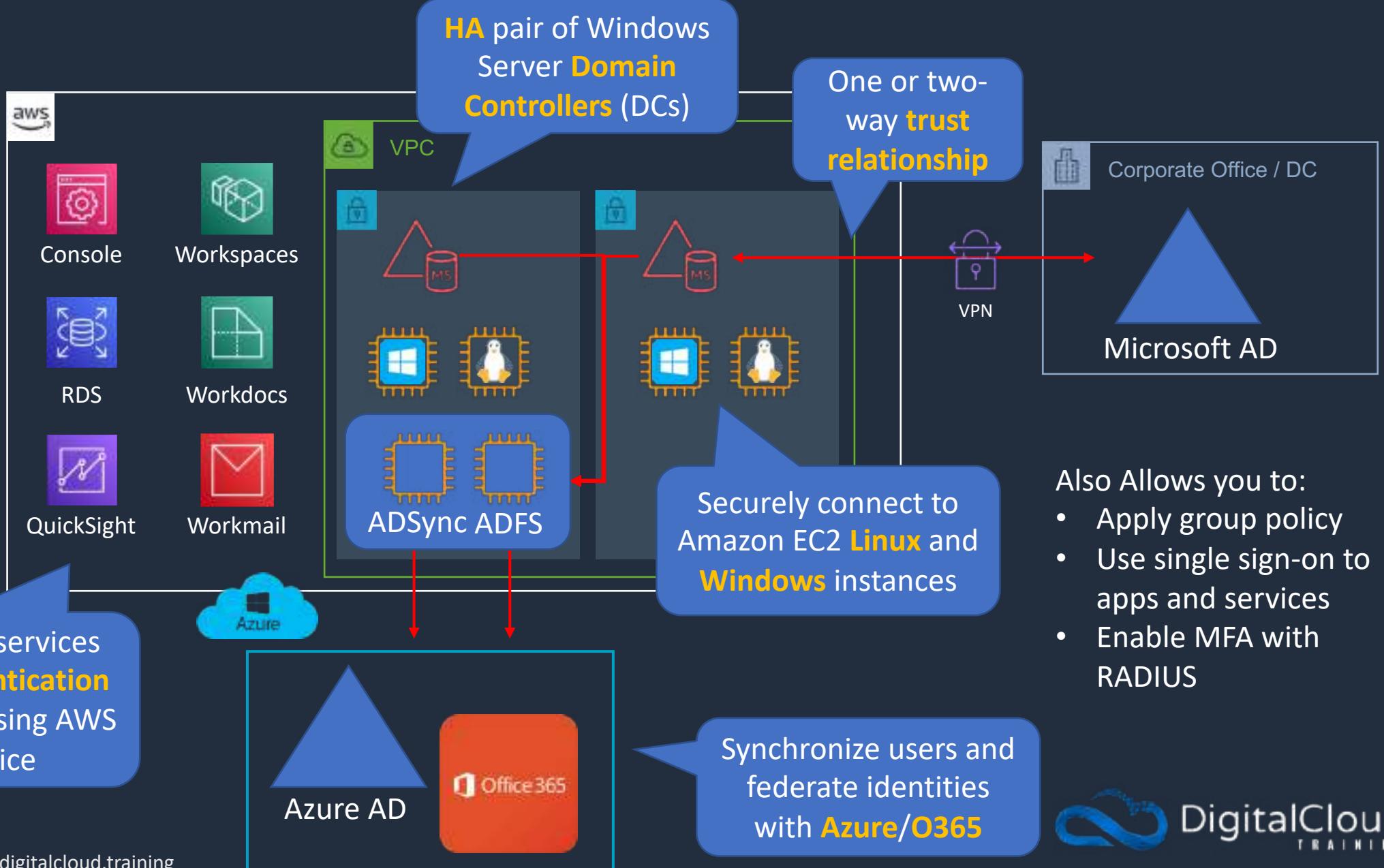
# AWS Directory Service





# AWS Managed Microsoft AD

Managed implementation of **Microsoft Active Directory** running on Windows Server



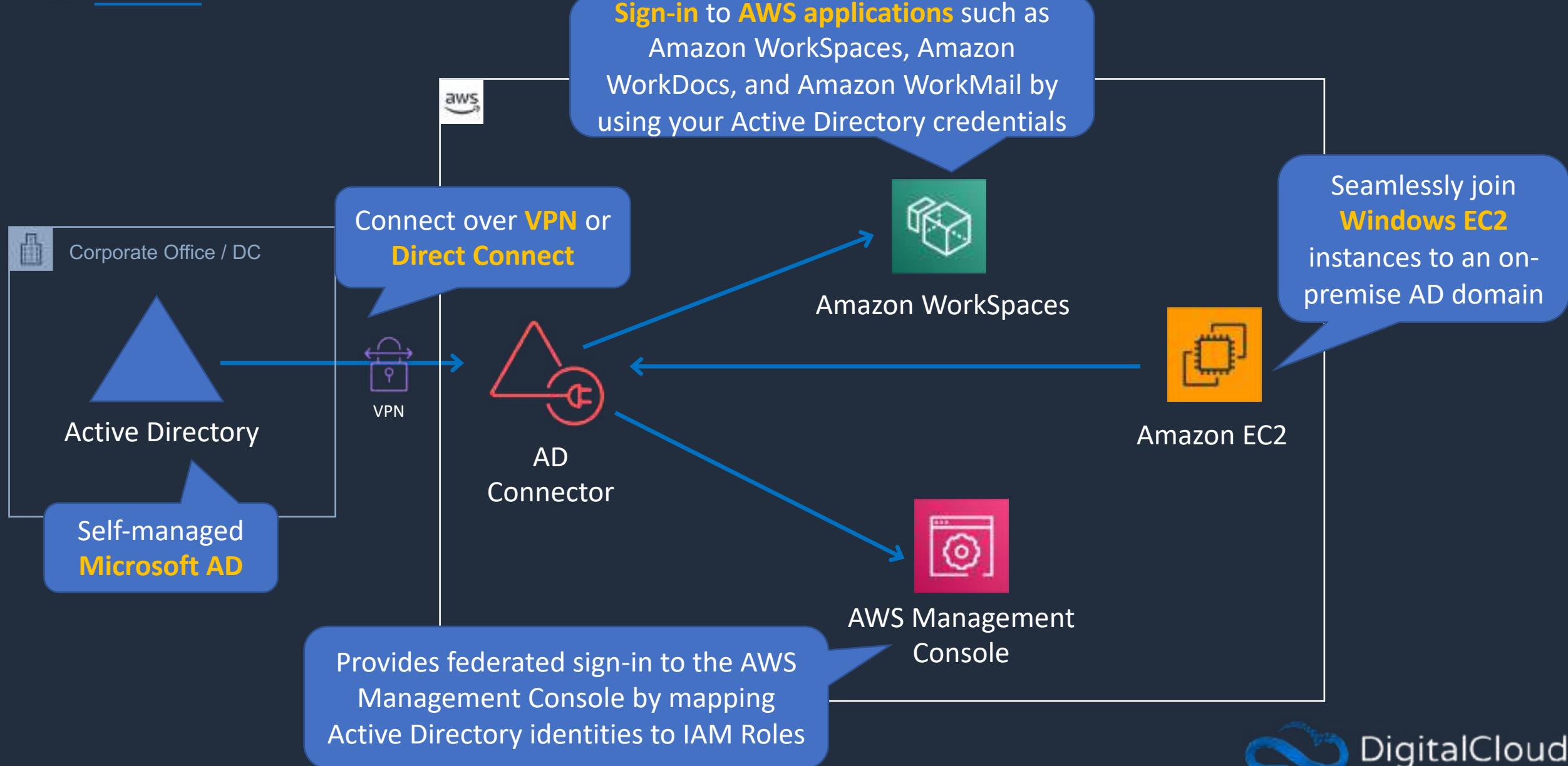


# AWS Managed Microsoft AD

- Fully managed AWS service
- Best choice if you have more than 5000 users and/or need a trust relationship set up
- Can perform schema extensions
- Can setup trust relationships to with on-premises Active Directories:
  - On-premise users and groups can access resources in either domain using SSO
  - Requires a VPN or Direct Connect connection
- Can be used as a standalone AD in the AWS cloud



# AD Connector





# AD Connector

---

- Redirects directory requests to your on-premise Active Directory
- Best choice when you want to use an existing Active Directory with AWS services
- AD Connector comes in two sizes:
  - Small – designed for organizations up to 500 users
  - Large – designed for organizations up to 5000 users
- Requires a VPN or Direct Connect connection
- Join EC2 instances to your on-premise AD through AD Connector
- Login to the AWS Management Console using your on-premise AD DCs for authentication

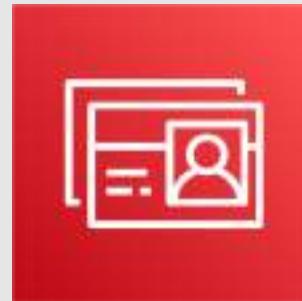


# Simple AD

---

- Inexpensive Active Directory-compatible service with common directory features
- Standalone, fully managed, directory on the AWS cloud
- Simple AD is generally the least expensive option
- Best choice for less than 5000 users and don't need advanced AD features
- Features include:
  - Manage user accounts / groups
  - Apply group policies
  - Kerberos-based SSO
  - Supports joining Linux or Windows based EC2 instances

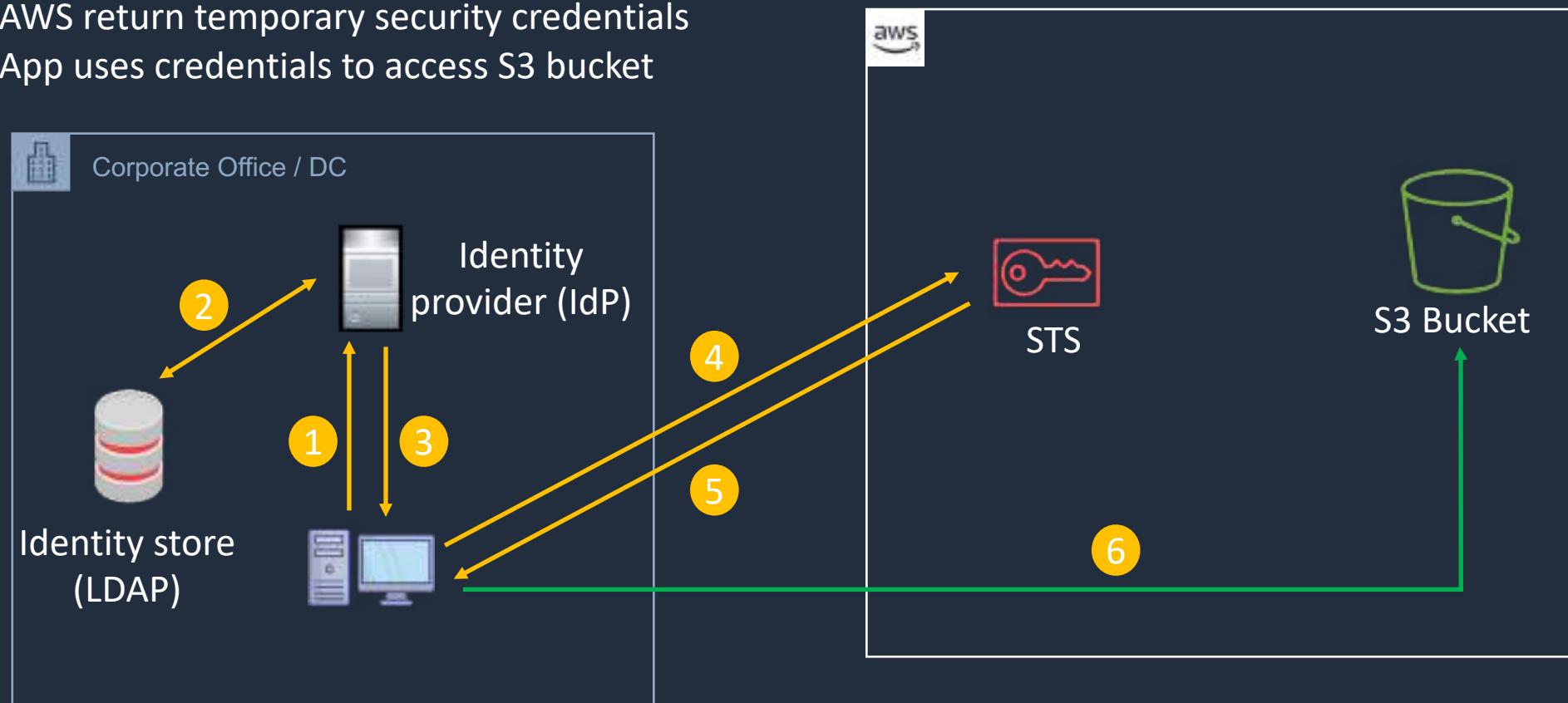
# Identity Federation





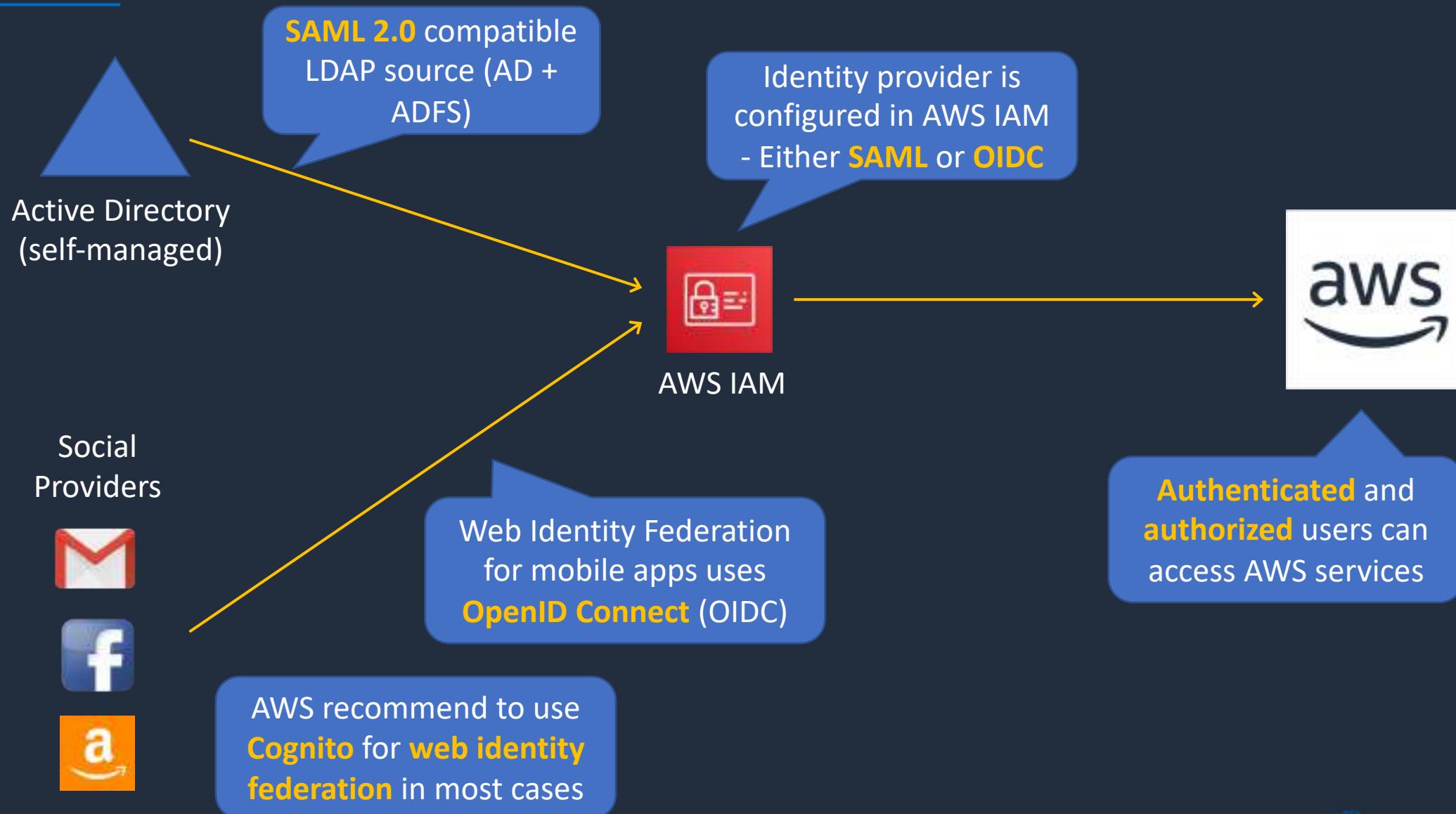
# Identity Federation

1. Client application attempts to authenticate using IdP
2. IdP authenticates the user
3. IdP sends client SAML assertion
4. App calls `sts:AssumeRoleWithSAML`
5. AWS return temporary security credentials
6. App uses credentials to access S3 bucket





# IAM Federation

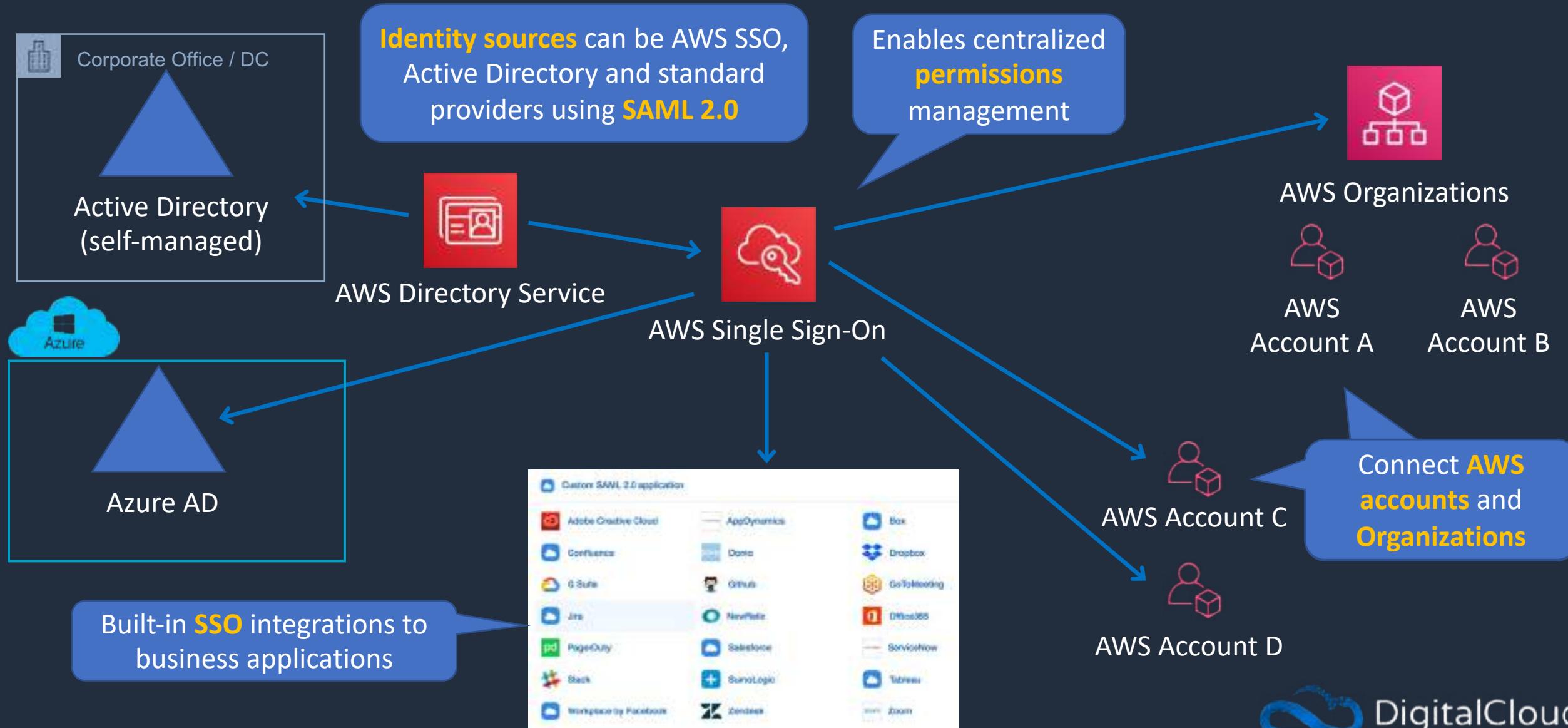


# AWS Single Sign-on (SSO)





# AWS Single Sign-on (SSO)



# Amazon Cognito

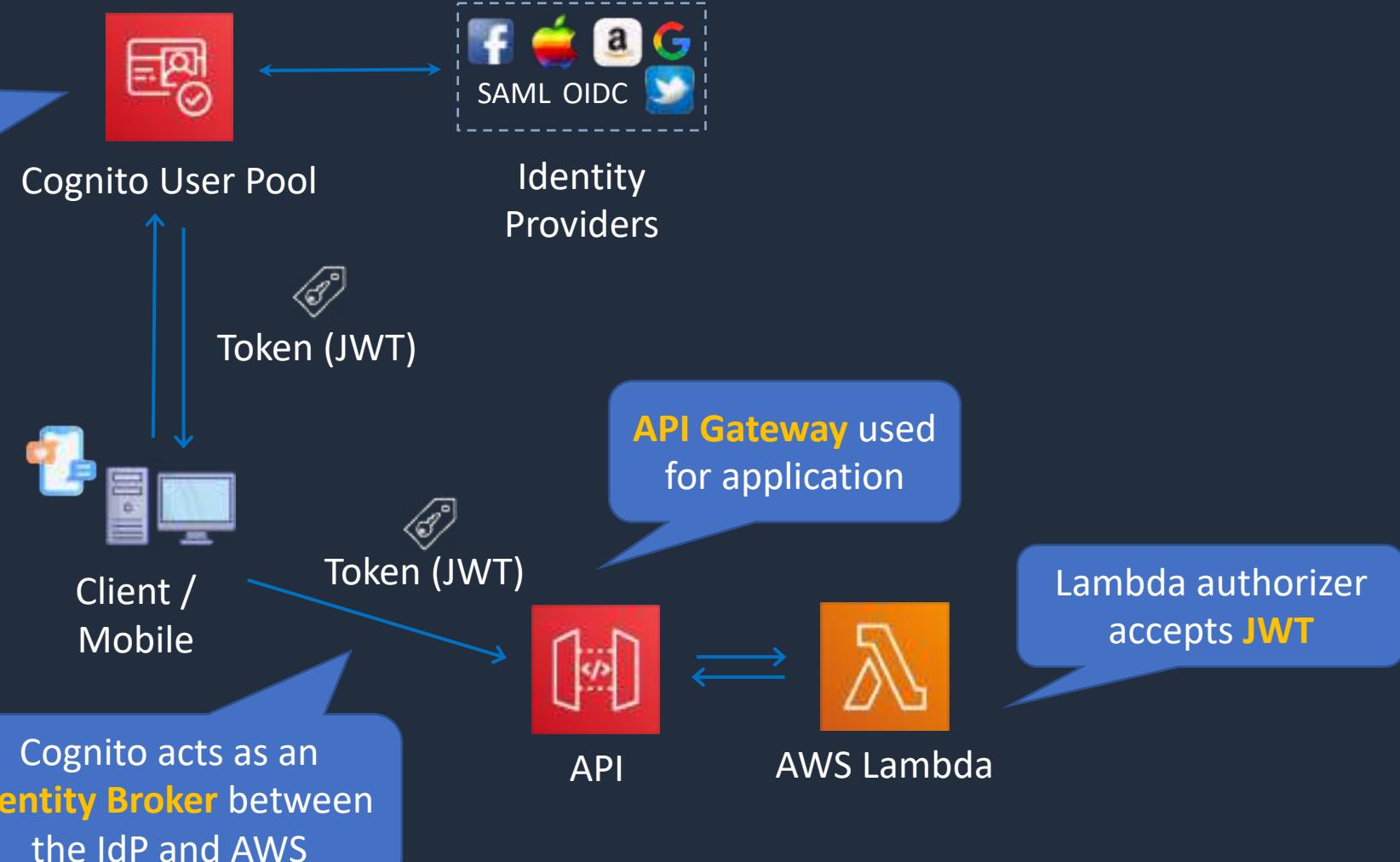




# Cognito User Pools

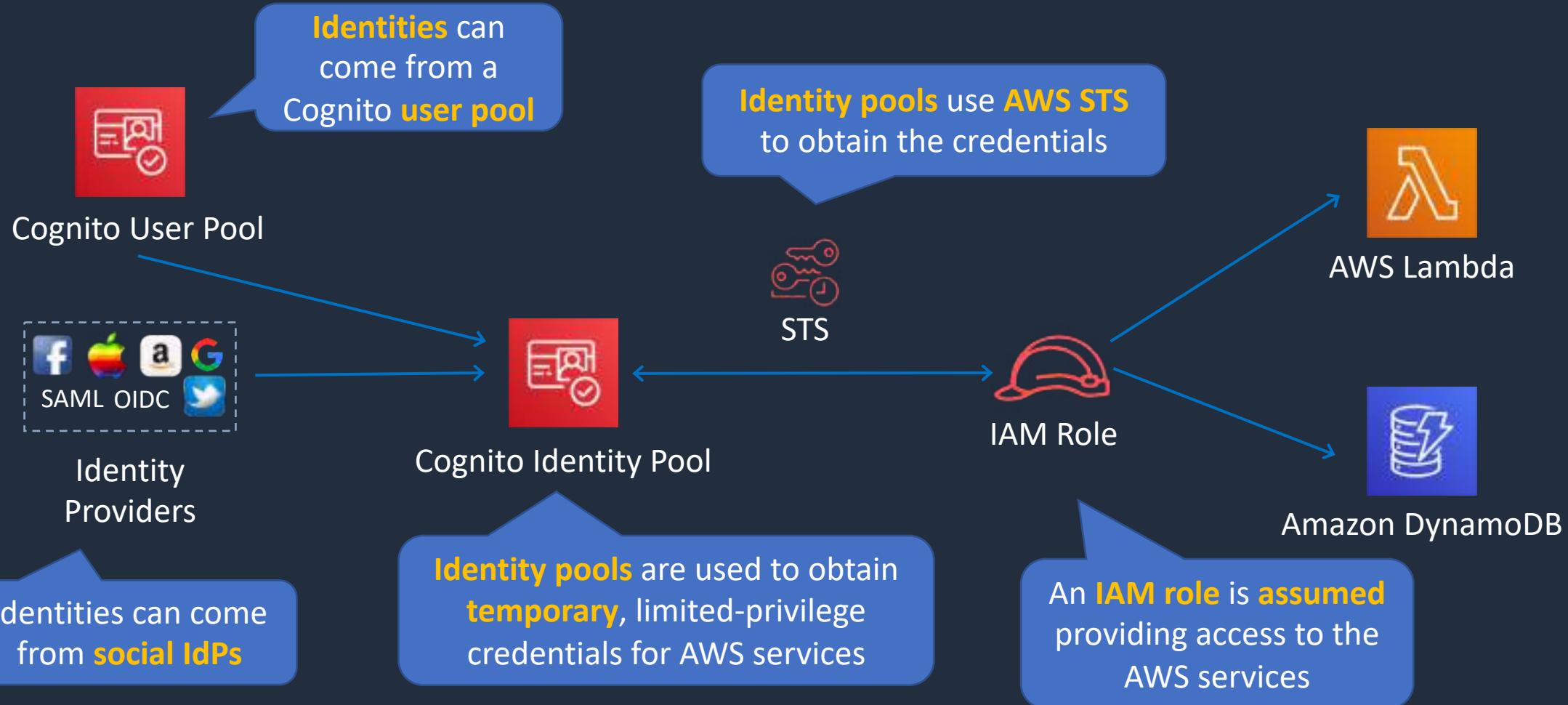
A **User Pool** is a directory for managing **sign-in** and **sign-up** for mobile applications

Users can also sign in using **social IdPs**



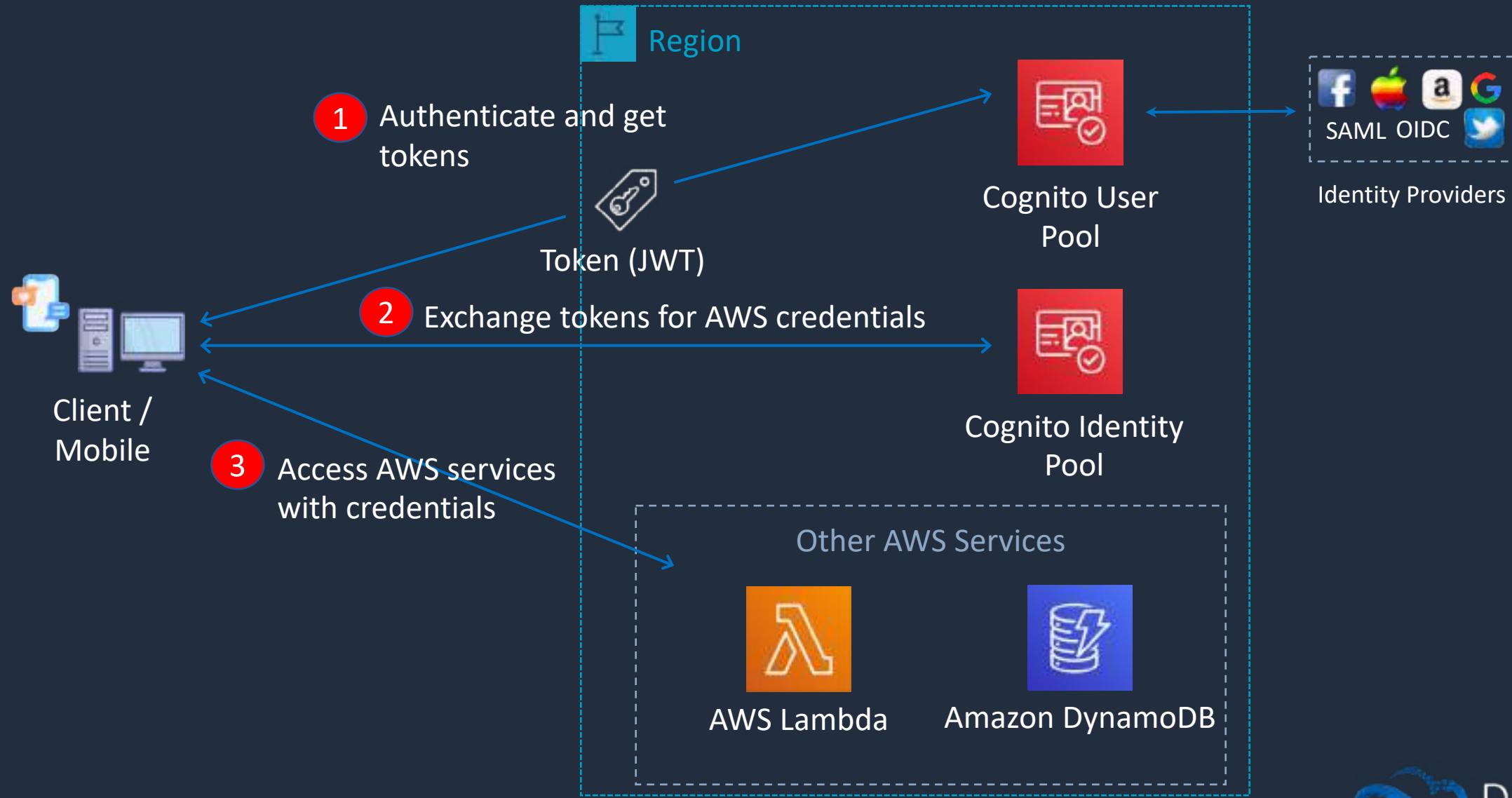


# Cognito Identity Pool





# User Pools + Identity Pools



# Encryption Primer





# Encryption In Transit vs At Rest



User

## Encryption In Transit

HTTPS Connection

Data is protected by  
**SSL/TLS** in transit



ALB

## Encryption At Rest

Amazon S3 **encrypts** the object as it is **written** to the bucket it



Unencrypted  
Object



Data encryption key



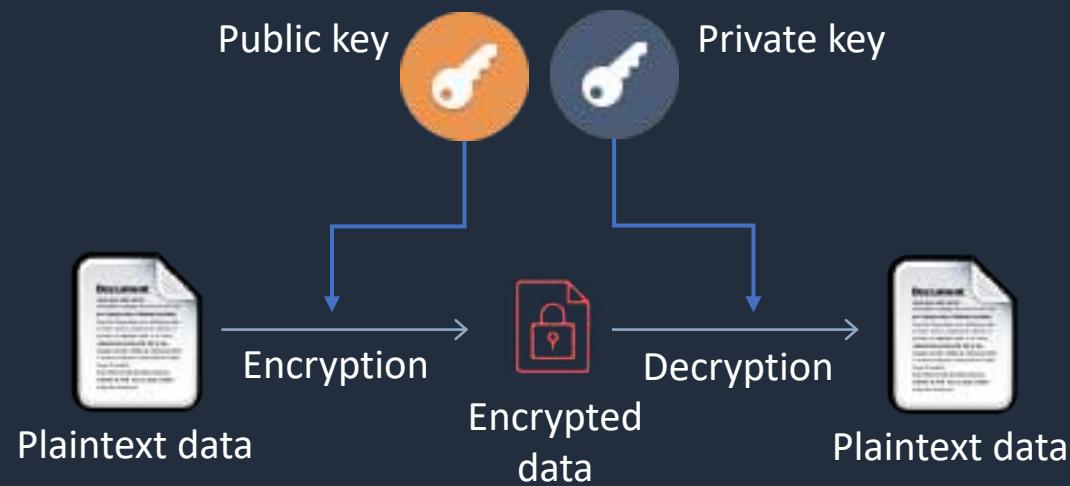
Encryption process



Encrypted  
bucket

# Asymmetric Encryption

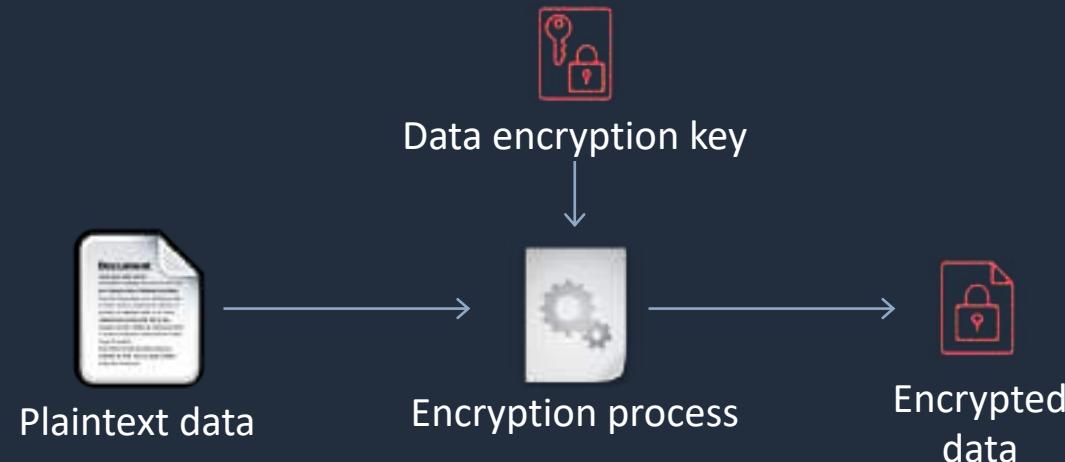
- Asymmetric encryption is also known as public key cryptography
- Messages encrypted with the public key can only be decrypted with the private key
- Messages encrypted with the private key can be decrypted with the public key
- Examples include SSL/TLS and SSH



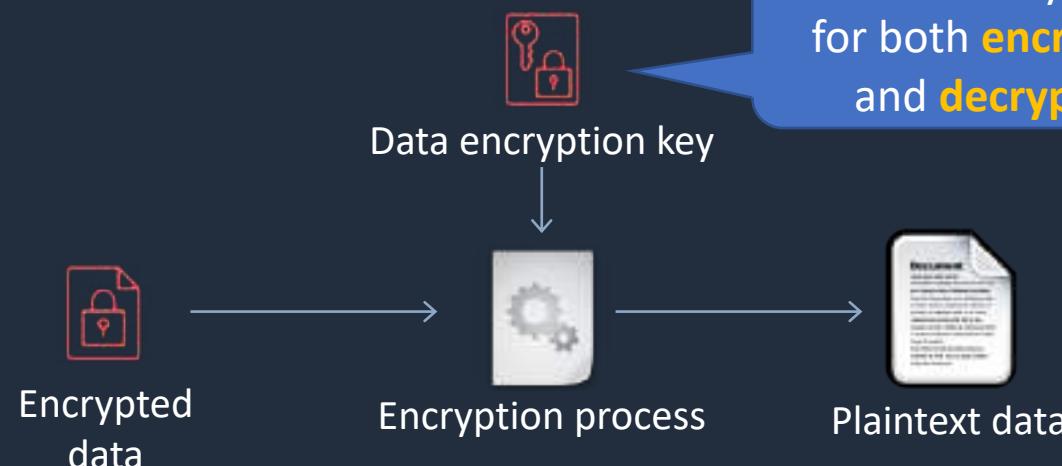


# Symmetric Encryption

## Encryption



## Decryption



The same key is used  
for both **encryption**  
and **decryption**

# AWS Key Management Service (KMS)





# Customer Master Keys (CMKs)

---

- Customer master keys are the primary resources in AWS KMS
- The CMK also contains the key material used to encrypt and decrypt data
- CMKs are created in AWS KMS
- Symmetric CMKs and the private keys of asymmetric CMKs never leave AWS KMS unencrypted
- By default, AWS KMS creates the key material for a CMK
- Can also import your own key material
- A CMK can encrypt data up to 4KB in size
- A CMK can generate, encrypt and decrypt Data Encryption Keys
- Data Encryption Keys can be used for encrypting large amounts of data



# AWS Managed CMKs

---

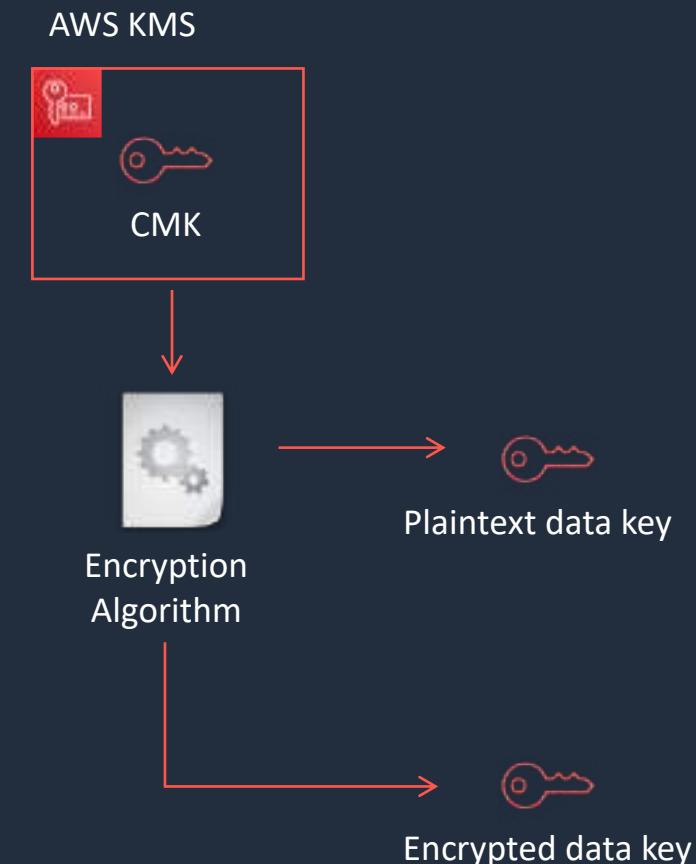
---

- Created, managed, and used on your behalf by an AWS service that is integrated with AWS KMS
- You cannot manage these CMKs, rotate them, or change their key policies
- You also cannot use AWS managed CMKs in cryptographic operations directly; the service that creates them uses them on your behalf



# Data Encryption Keys

- Data keys are encryption keys that you can use to encrypt data, including large amounts of data and other data encryption keys
- You can use AWS KMS customer master keys (CMKs) to generate, encrypt, and decrypt data keys
- AWS KMS does not store, manage, or track your data keys, or perform cryptographic operations with data keys
- You must use and manage data keys outside of AWS KMS





# Customer Master Keys (CMKs)

Type of CMK	Can view	Can manage	Used only for my AWS account	Automatic rotation
<b>Customer managed CMK</b>	Yes	Yes	Yes	Optional. Every 365 days
<b>AWS managed CMK</b>	Yes	No	Yes	Required. Every 1095 days
<b>AWS owned CMK</b>	No	No	No	Varies

# Create Encryption Key



# AWS CloudHSM



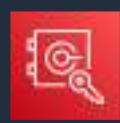


# AWS CloudHSM

---

---

- AWS CloudHSM is a cloud-based hardware security module (HSM)
- Generate and use your own encryption keys on the AWS Cloud
- CloudHSM runs in your Amazon VPC
- Uses FIPS 140-2 level 3 validated HSMs
- Managed service and automatically scales
- Retain control of your encryption keys - you control access (and AWS has no visibility of your encryption keys)



# AWS CloudHSM Use Cases

---

---

- Offload SSL/TLS processing from web servers
- Protect private keys for an issuing certificate authority (CA)
- Store the master key for Oracle DB Transparent Data Encryption
- Custom key store for AWS KMS – retain control of the HSM that protects the master keys



# AWS CloudHSM vs KMS

---

---

	CloudHSM	AWS KMS
<b>Tenancy</b>	Single-tenant HSM	Multi-tenant AWS service
<b>Availability</b>	Customer-managed durability and available	Highly available and durable key storage and management
<b>Root of Trust</b>	Customer managed root of trust	AWS managed root of trust
<b>FIPS 140-2</b>	Level 3	Level 2 / Level 3
<b>3<sup>rd</sup> Party Support</b>	Broad 3 <sup>rd</sup> Party Support	Broad AWS service support

# AWS Certificate Manager (ACM)





# AWS Certificate Manager (ACM)

---

---

- Create, store and renew SSL/TLS X.509 certificates
- Single domains, multiple domain names and wildcards
- Integrates with several AWS services including:
  - **Elastic Load Balancing**
  - **Amazon CloudFront**
  - **AWS Elastic Beanstalk**
  - **AWS Nitro Enclaves**
  - **AWS CloudFormation**



# AWS Certificate Manager (ACM)

---

---

- **Public certificates** are signed by the AWS public Certificate Authority
- You can also create a Private CA with ACM
- Can then issue private certificates
- You can also import certificates from third-party issuers

# SSL/TLS Certificate in ACM



# AWS Web Application Firewall (WAF)





# AWS WAF

---

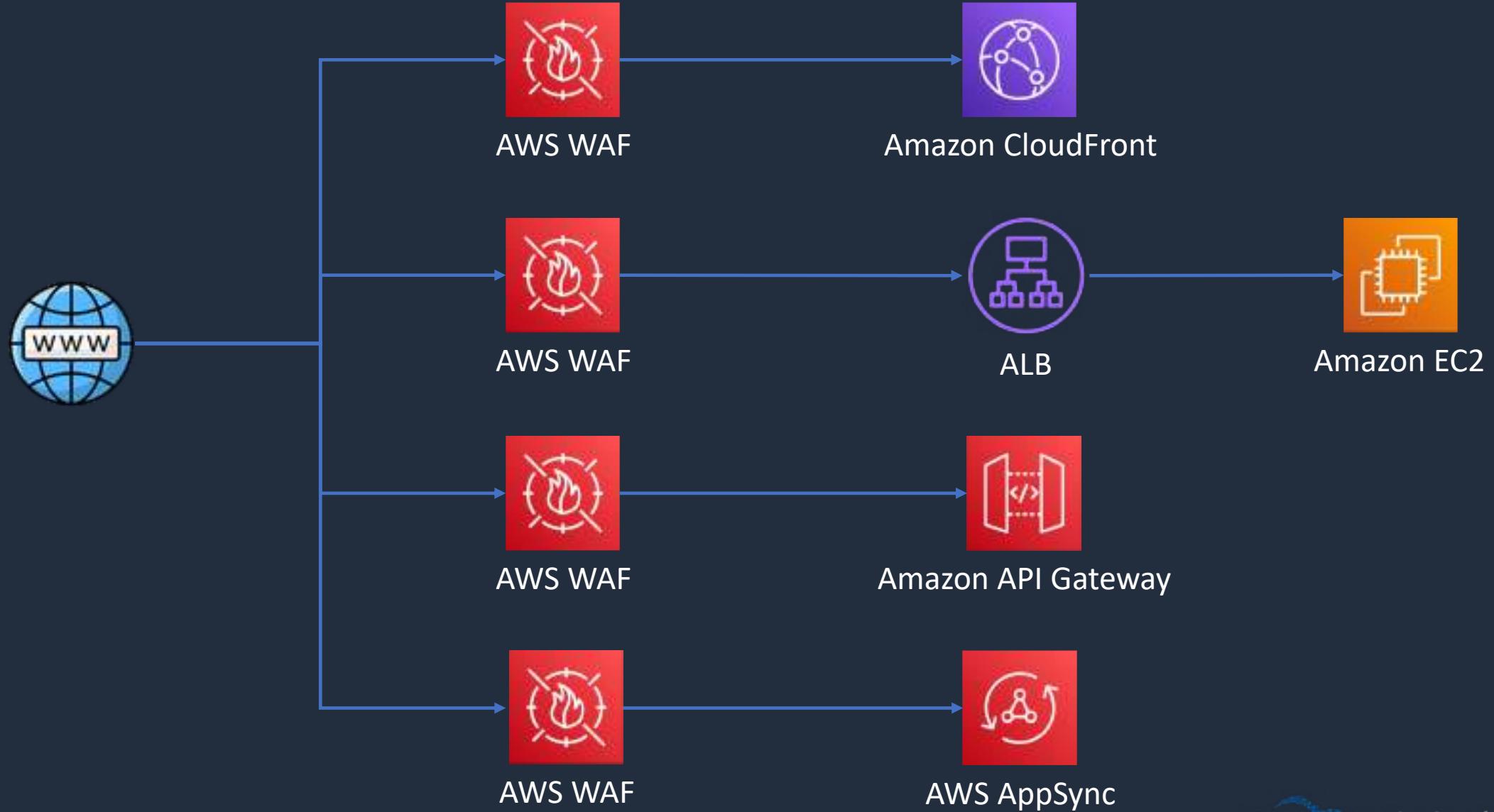
---

- AWS WAF is a web application firewall
- WAF lets you create rules to filter web traffic based on conditions that include IP addresses, HTTP headers and body, or custom URIs
- WAF makes it easy to create rules that block common web exploits like SQL injection and cross site scripting



# AWS WAF

---





# AWS WAF

- **Web ACLs** – You use a web access control list (ACL) to protect a set of AWS resources
- **Rules** – Each rule contains a statement that defines the inspection criteria, and an action to take if a web request meets the criteria
- **Rule groups** – You can use rules individually or in reusable rule groups





# AWS WAF

---

---

- **IP Sets** - An IP set provides a collection of IP addresses and IP address ranges that you want to use together in a rule statement
- **Regex pattern set** - A regex pattern set provides a collection of regular expressions that you want to use together in a rule statement



# AWS WAF

---

---

A **rule action** tells AWS WAF what to do with a web request when it **matches** the criteria defined in the rule:

- **Count** – AWS WAF counts the request but doesn't determine whether to allow it or block it. With this action, AWS WAF continues processing the remaining rules in the web ACL
- **Allow** – AWS WAF allows the request to be forwarded to the AWS resource for processing and response
- **Block** – AWS WAF blocks the request and the AWS resource responds with an HTTP 403 (Forbidden) status code



**Match** statements compare the web request or its origin against conditions that you provide

Match Statement	Description
Geographic match	Inspects the request's country of origin
IP set match	Inspects the request against a set of IP addresses and address ranges
Regex pattern set	Compares regex patterns against a specified request component
Size constraint	Checks size constraints against a specified request component
SQLi attack	Inspects for malicious SQL code in a specified request component
String match	Compares a string to a specified request component
XSS scripting attack	Inspects for cross-site scripting attacks in a specified request component

# AWS Shield





# AWS Shield

---

---

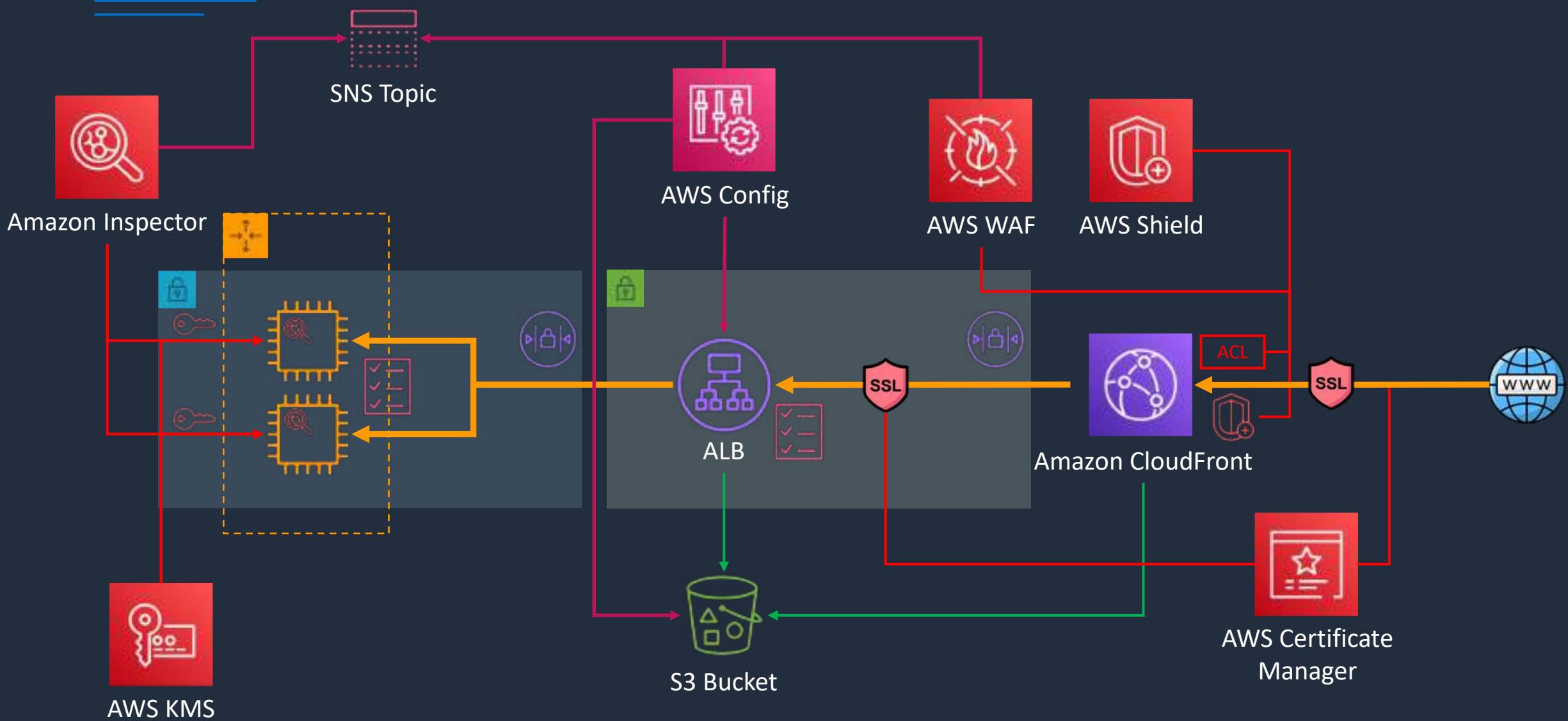
- AWS Shield is a managed Distributed Denial of Service (DDoS) protection service
- Safeguards web application running on AWS with always-on detection and automatic inline mitigations
- Helps to minimize application downtime and latency
- Two tiers –
  - **Standard** – no cost
  - **Advanced** - \$3k USD per month and 1 year commitment
- Integrated with Amazon CloudFront (standard included by default)

# Defense In-Depth





# Secure Multi-Tier Architecture



# Architecture Patterns - Security





# Architecture Patterns – Security

---

---

## Requirement

Need to enable custom domain name and encryption in transit for an application running behind an Application Load Balancer

Website running on EC2 instances behind and ALB must be protected against well known web exploits

Need to block access to an application running on an ALB from connections originating in a specific list of countries

## Solution

Use AWS Route 53 to create an Alias record to the ALB's DNS name and attach an SSL/TLS certificate issued by Amazon Certificate Manager (ACM)

Create a Web ACL in AWS WAF to protect against web exploits and attach to the ALB

Create a Web ACL in AWS WAF with a geographic match and block traffic that matches the list of countries



## Requirement

Company needs to encrypt large volumes of data using a CMK in AWS KMS

## Solution

Create a data encryption key using the CMK to encrypt large volumes of data

Mobile app requires authorized access to AWS services. Users authenticate using social IdPs and a preconfigured Web UI is required for logging in

Create an Amazon Cognito User Pool that leverages the social IdPs and an Identity Pool for gaining temporary credentials for AWS

Company has an on-premises Microsoft AD and AWS DX connection. Requires joining EC2 instances to on-premises domain

Configure an AD Connector that uses the on-premises AD

# SECTION 16

## Migration and Transfer Services

# AWS Migration Tools Overview



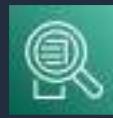


# AWS Migration Tools



Region

**Collect data**  
about servers in  
on-premises DC



AWS Application  
Discovery Service



AWS Migration Hub

**Monitor migrations**  
that use AWS or  
partner tools



Amazon S3



Amazon RDS



EC2 Instances



EFS File system



AWS Server Migration  
Service



AWS Database Migration  
Service



AWS DataSync



VPN, Direct  
Connect or Internet



Corporate data center



Servers



Database



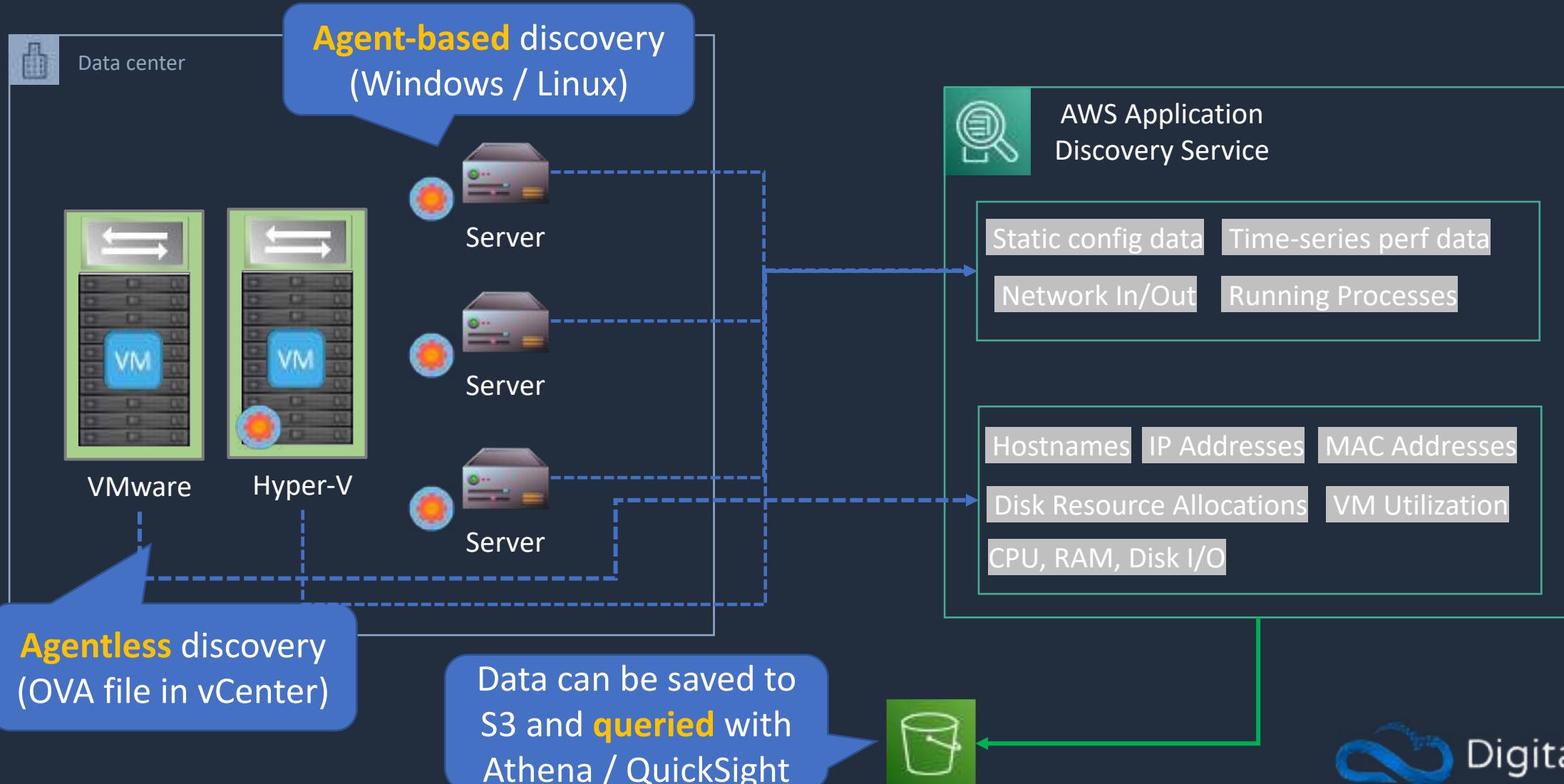
NAS / File  
Server

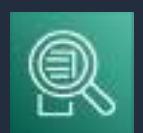
# AWS Application Discovery Service





# AWS Application Discovery Service





# AWS Application Discovery Service

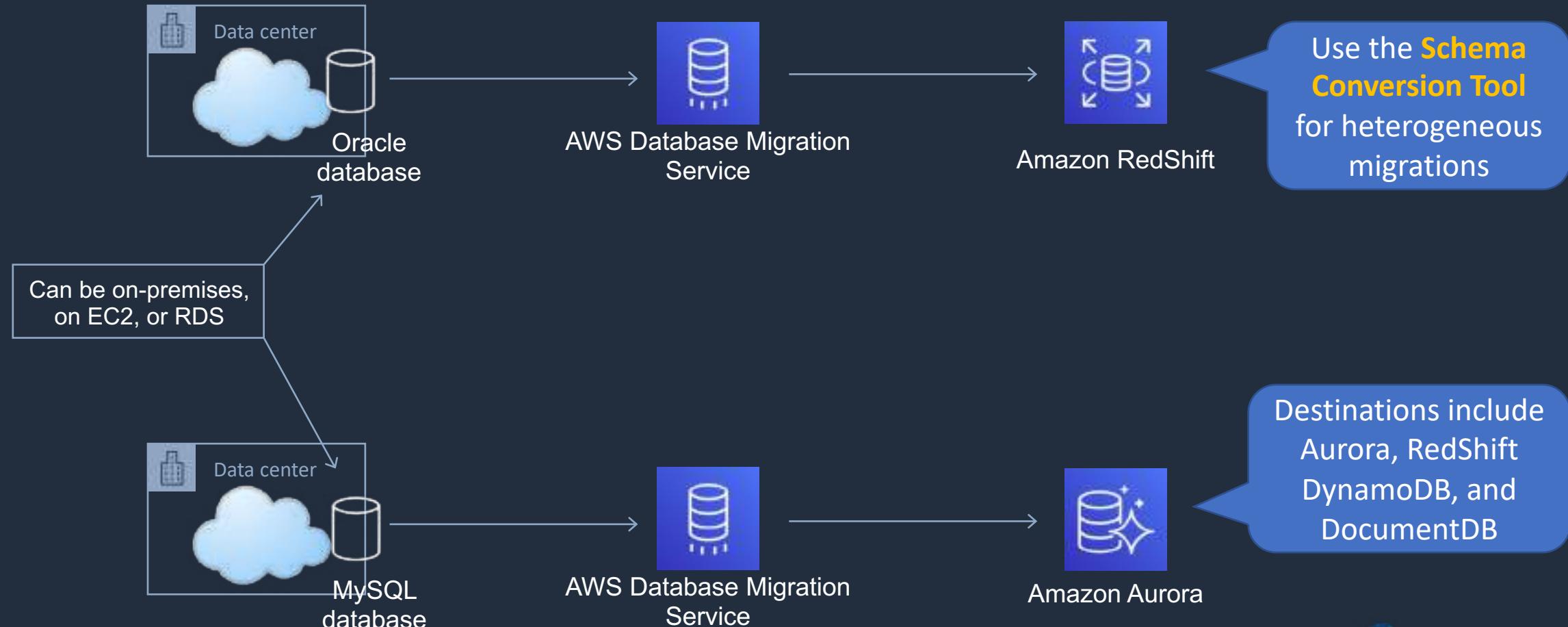
	Discovery Connector	Discovery Agent
<b>Supported Servers</b>	VMware	VMware, Physical
<b>Deployment</b>	Per vCenter	Per Server
<b>Collected data</b>	Static configuration data  VM utilization metrics	Static configuration data  Time Series Performance info (export)  Network Inbound/outbound (export)  Running processes (export)
<b>Supported OS</b>	Any OS running in VMware vCenter (v5.5, v6, & v6.5)	Amazon Linux, Ubuntu, RHEL, CentOS, SUSE, Windows Server

# AWS Database Migration Service (DMS)





# AWS Database Migration Service (DMS)





# AWS DMS Use Cases

- **Cloud to Cloud** – EC2 to RDS, RDS to RDS, RDS to Aurora
- **On-Premises to Cloud**
- **Homogeneous migrations** – Oracle to Oracle, MySQL to RDS MySQL, Microsoft SQL to RDS for SQL Server
- **Heterogeneous migrations** – Oracle to Aurora, Oracle to PostgreSQL, Microsoft SQL to RDS MySQL (must convert schema first with the **Schema Conversion Tool (SCT)**)



# AWS DMS Use Cases

---

---

- **Development and Test** – use the cloud for dev/test workloads
- **Database consolidation** – consolidate multiple source DBs to a single target DB
- **Continuous Data Replication** – use for DR, dev/test, single source multi-target or multi-source single target

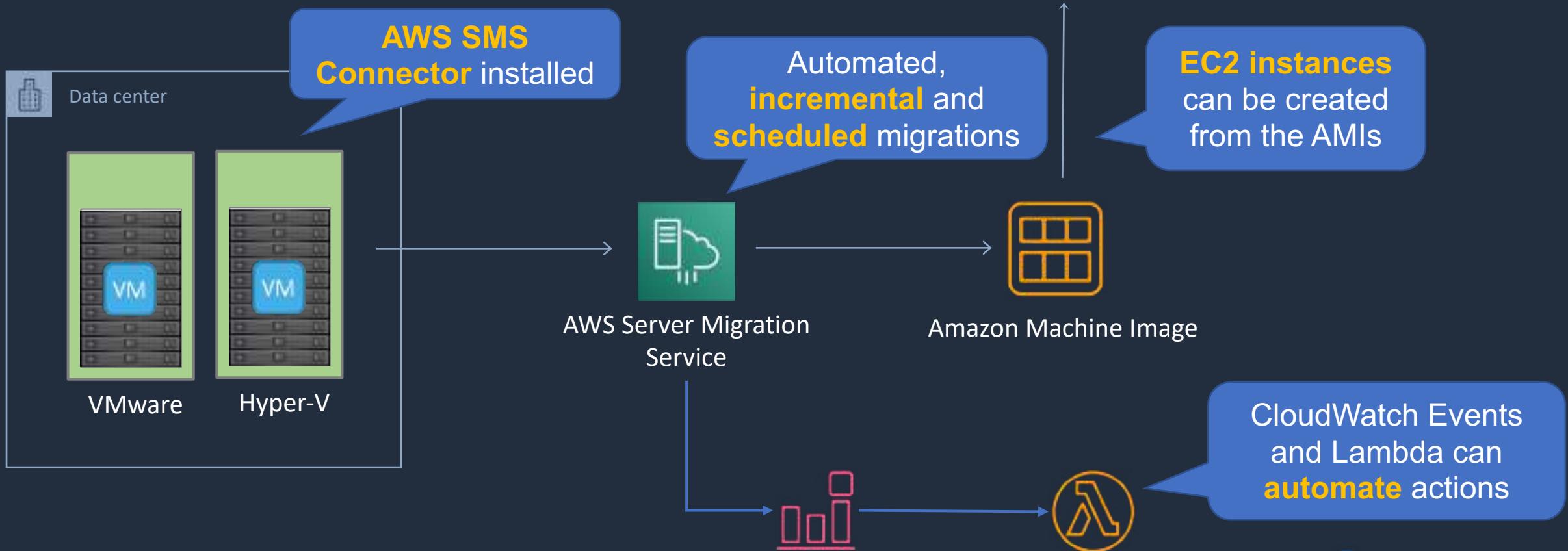
# AWS Server Migration Service (SMS)





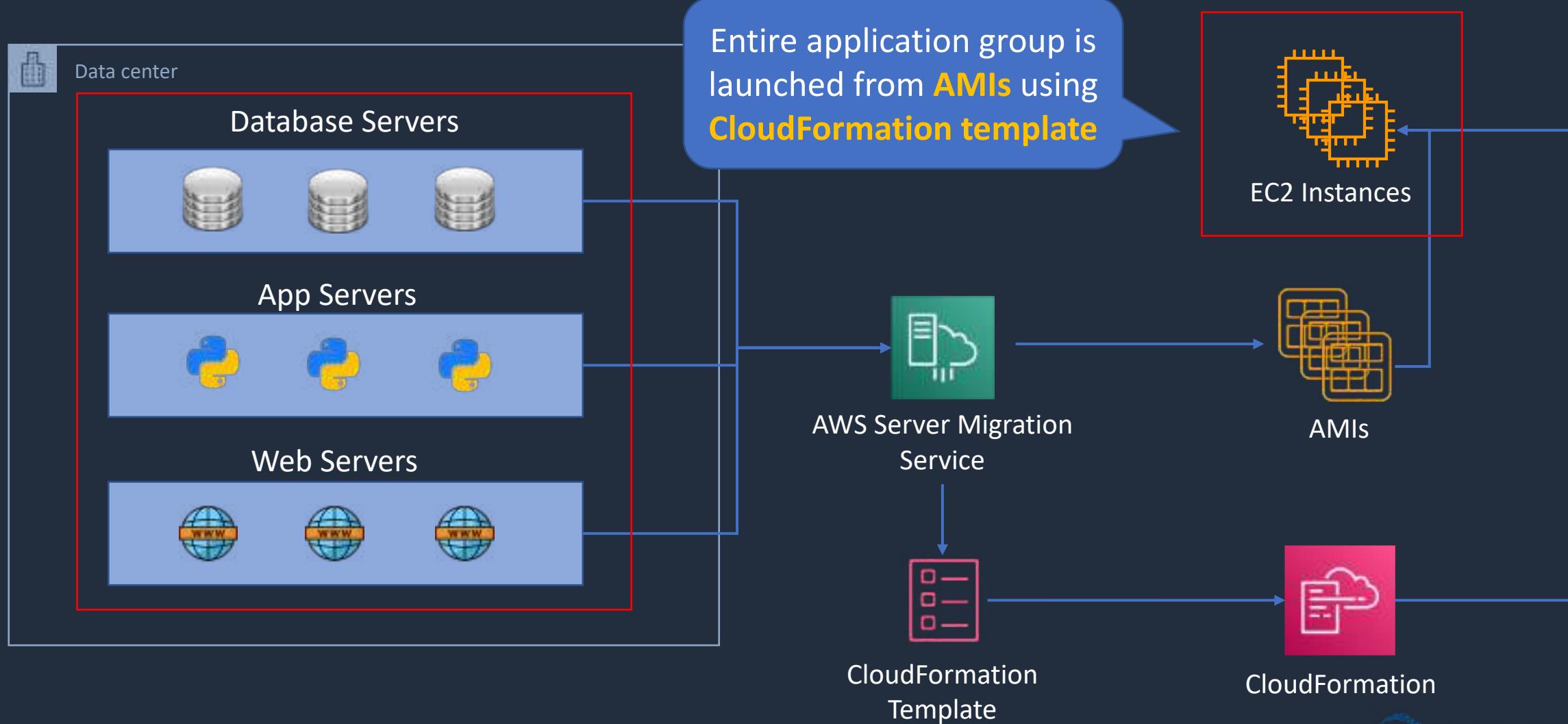
# AWS Server Migration Service (SMS)

- AWS SMS migrates VMware vSphere, Microsoft Hyper-V/SCVMM, and Azure virtual machines to Amazon EC2





# AWS SMS – Application Migration

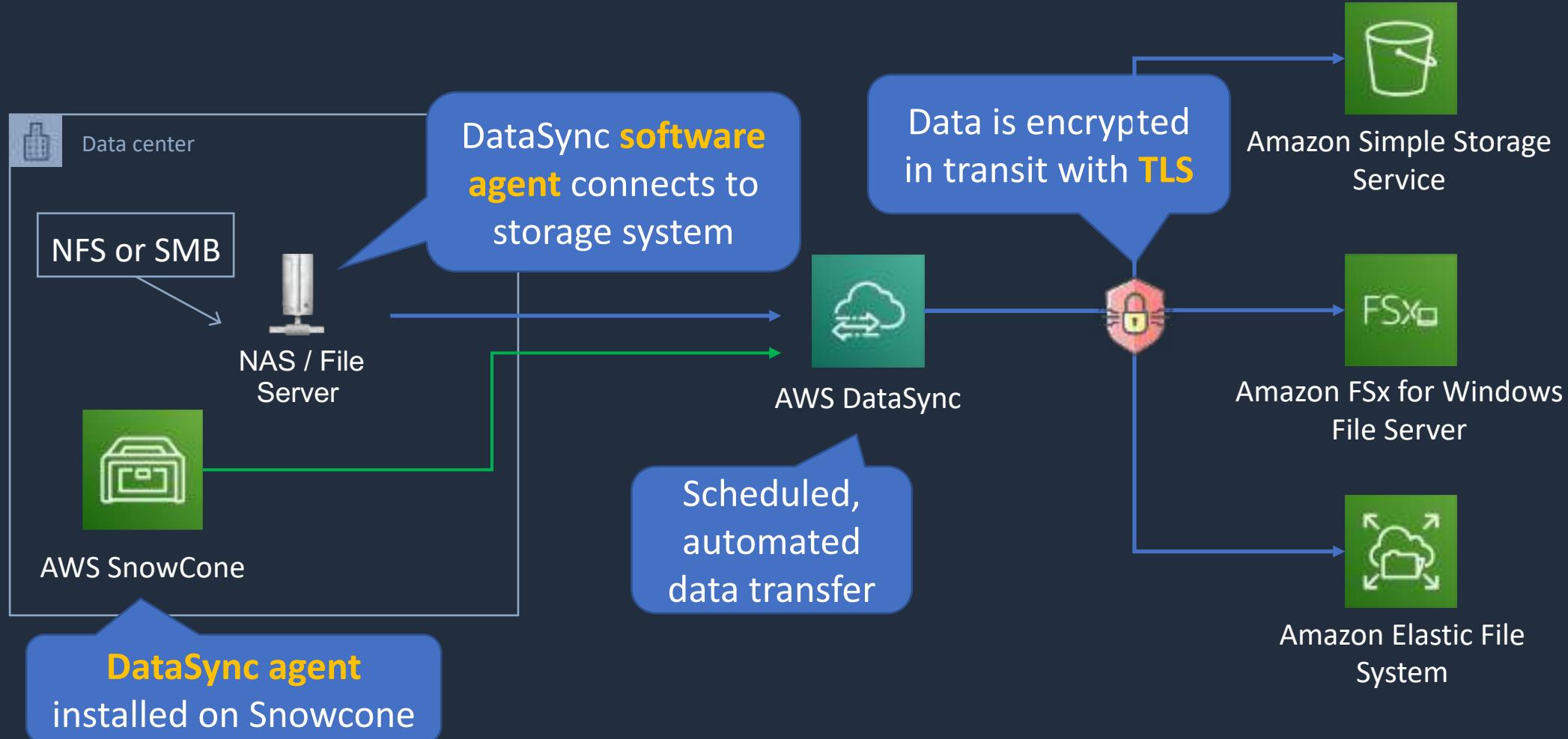


# AWS DataSync

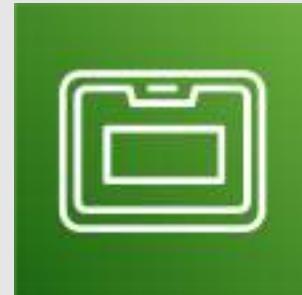




# AWS DataSync



# AWS Snow Family





# AWS Snowball Family

- **AWS Snowball and Snowmobile** are used for migrating large volumes of data to AWS
- **Snowball Edge Compute Optimized**
  - Provides block and object storage and optional GPU
  - Use for data collection, machine learning and processing, and storage in environments with intermittent connectivity (edge use cases)
- **Snowball Edge Storage Optimized**
  - Provides block storage and Amazon S3-compatible object storage
  - Use for local storage and large-scale data transfer
- **Snowcone**
  - Small device used for edge computing, storage and data transfer
  - Can transfer data offline or online with AWS DataSync agent





# AWS Snowball Family

---

- Uses a secure storage device for physical transportation
- Snowball Client is software that is installed on a local computer and is used to identify, compress, encrypt, and transfer data
- Uses 256-bit encryption (managed with the AWS KMS) and tamper-resistant enclosures with TPM
- Snowball (80TB) (50TB ) “petabyte scale”
- Snowball Edge (100TB) “petabyte scale”
- Snowmobile – “exabyte scale” with up to 100PB per Snowmobile





# AWS Snowball Family

---

Ways to optimize the performance of Snowball transfers:

1. Use the latest Mac or Linux Snowball client
2. Batch small files together
3. Perform multiple copy operations at one time
4. Copy from multiple workstations
5. Transfer directories, not files





# AWS Snowball Use Cases

---

- **Cloud data migration** – migrate data to the cloud
- **Content distribution** – send data to clients or customers
- **Tactical Edge Computing** – collect data and compute
- **Machine learning** – run ML directly on the device
- **Manufacturing** – data collection and analysis in the factory
- **Remote locations with simple data** – pre-processing, tagging, compression etc.

# Architecture Patterns – Migration and Transfer





# Architecture Patterns – Migration and Transfer

## Requirement

Company is migrating Linux and Windows VMs in VMware to the cloud. Need to determine performance requirements for right-sizing

Company has a mixture of VMware VMs and physical servers to migrate to AWS. Need to identify dependencies for grouping applications for migration

Need to migrate an Oracle data warehouse to AWS

## Solution

Install the Application Discovery Service discovery connector in VMware vCenter to gather data

Install the Application Discovery Service discovery connector in VMware vCenter and the discovery agent on physical servers

Migrate using AWS DMS and SCT to a RedShift data warehouse



# Architecture Patterns – Migration and Transfer

## Requirement

Snowball Edge used to transfer millions of small files using a shell script.  
Transfer times are very slow

## Solution

Perform multiple copy operations at one time by running each command from a separate terminal, in separate instances of the Snowball client

Need to minimize downtime for servers that must be migrated to AWS

Use AWS SMS and perform a final synchronization before cutting over in a short outage window

Need to migrate 50TB of data and company only has a 1Gbps internet link. Requirement is urgent

Use AWS Snowball to transfer the data