

# SECTION 1

## Introduction

# SECTION 2

## AWS Accounts and Organizations

# Create Management AWS Account



# What you need...



Credit card for setting up the account and paying any bills



Unique email address for this account

john@example.com



john+dctmanagement@example.com

john+dctprod@example.com



AWS account name – mine will be **DCT-MANAGEMENT**



Phone to receive an **SMS** verification code



Check if you can use an alias with an existing email address (e.g dynamic aliases in Gmail / O365)

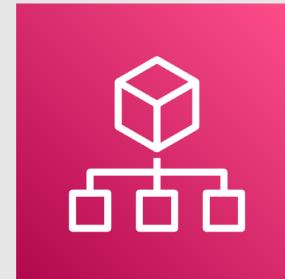
# Configure Account and Setup Billing Alarms



# Install Tools (AWS CLI and VS Code)



# AWS Organizations

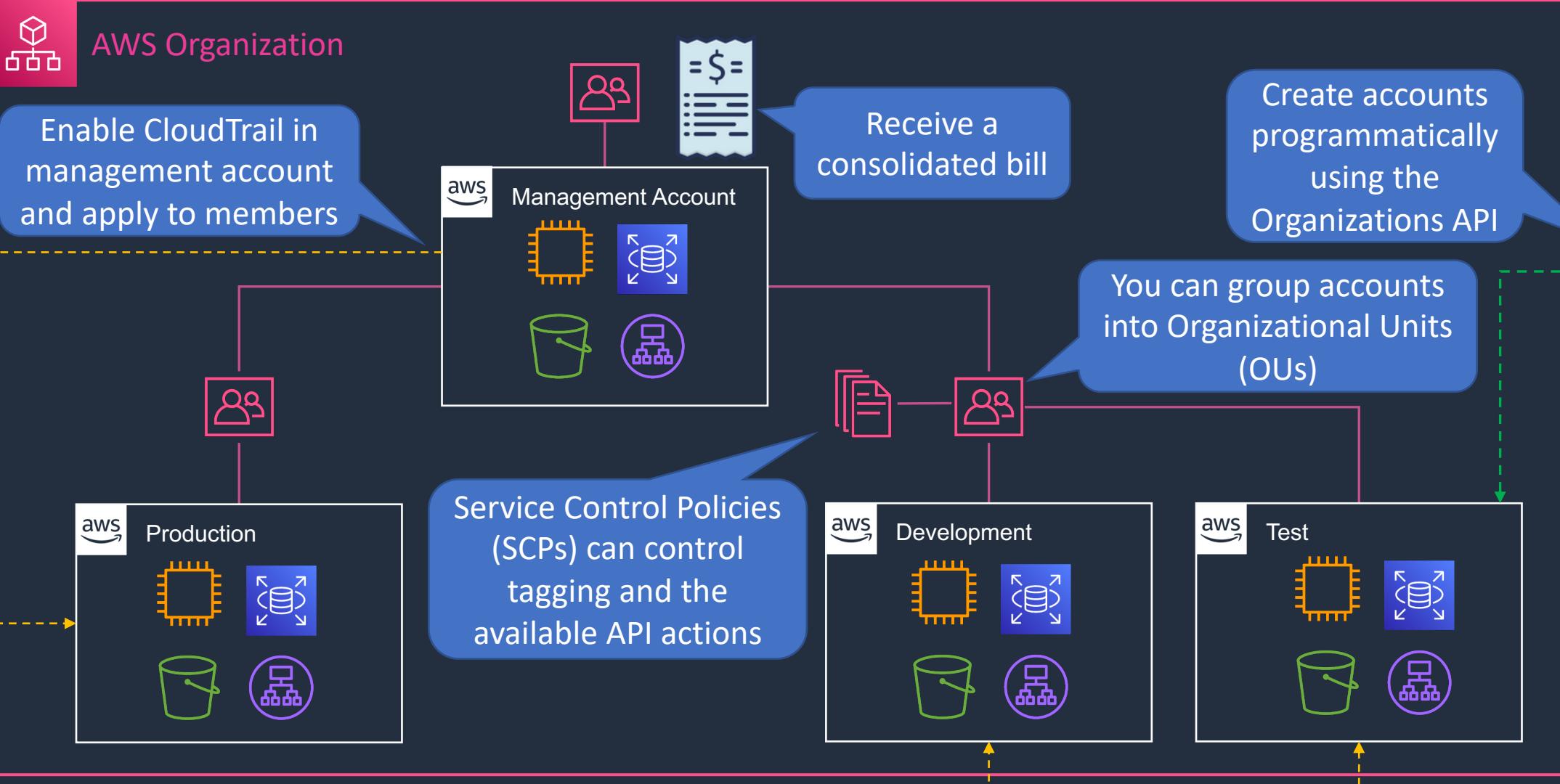




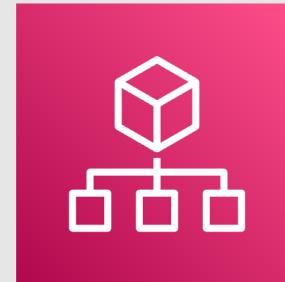
# AWS Organizations



Enable AWS SSO using  
on-prem directory

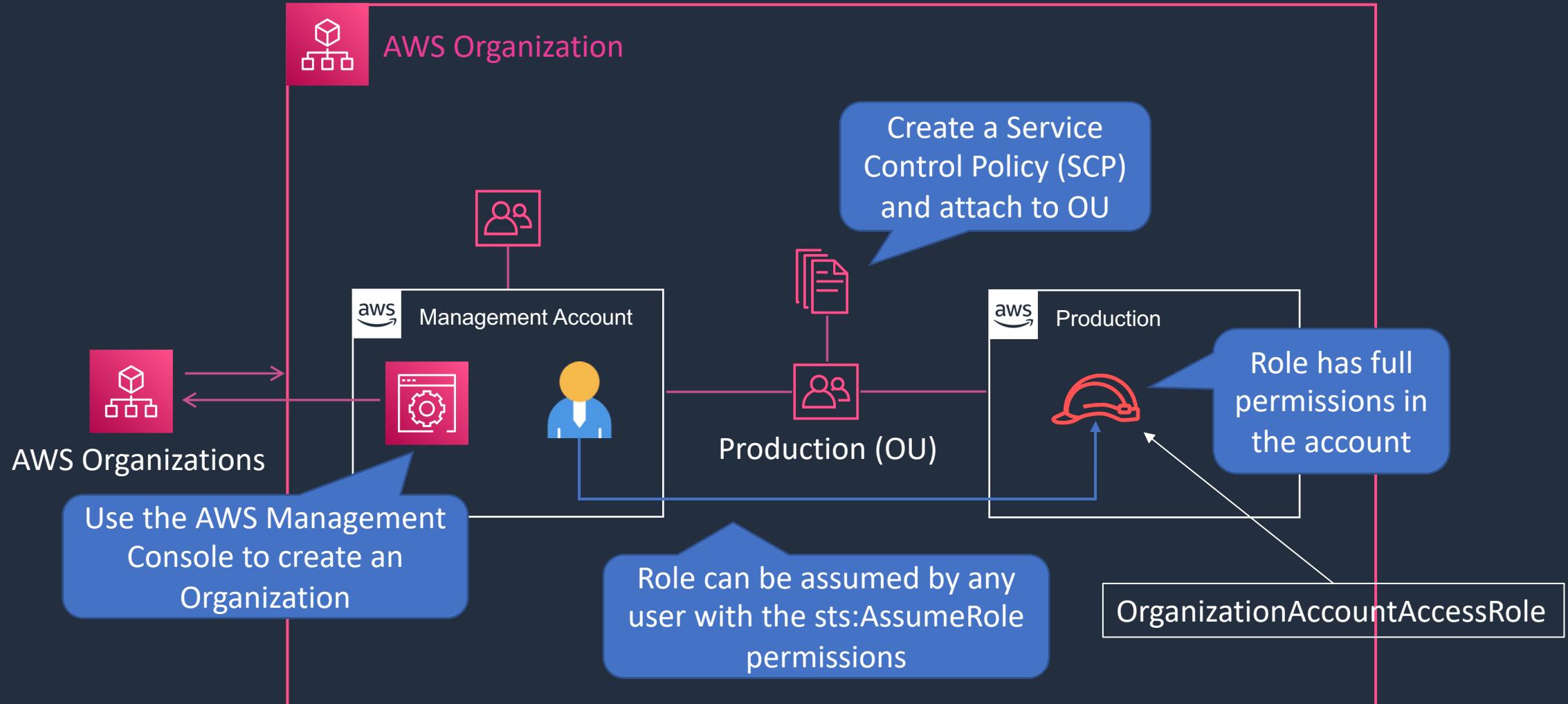


# Account Configuration





# Account Configuration



# Create AWS Organization and Add Account

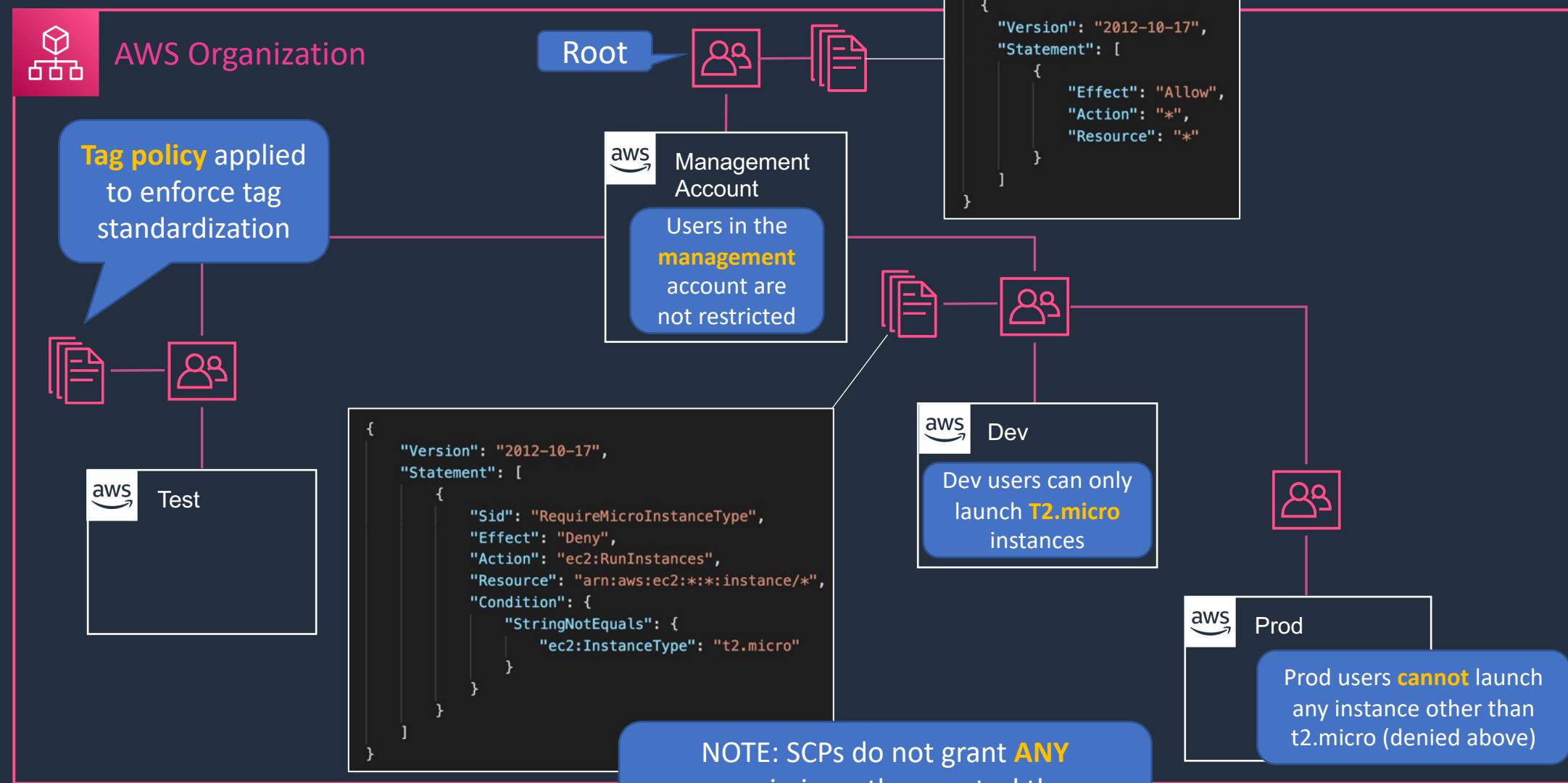


# Service Control Policies (SCPs)



# Service Control Policies

SCPs control the maximum available permissions



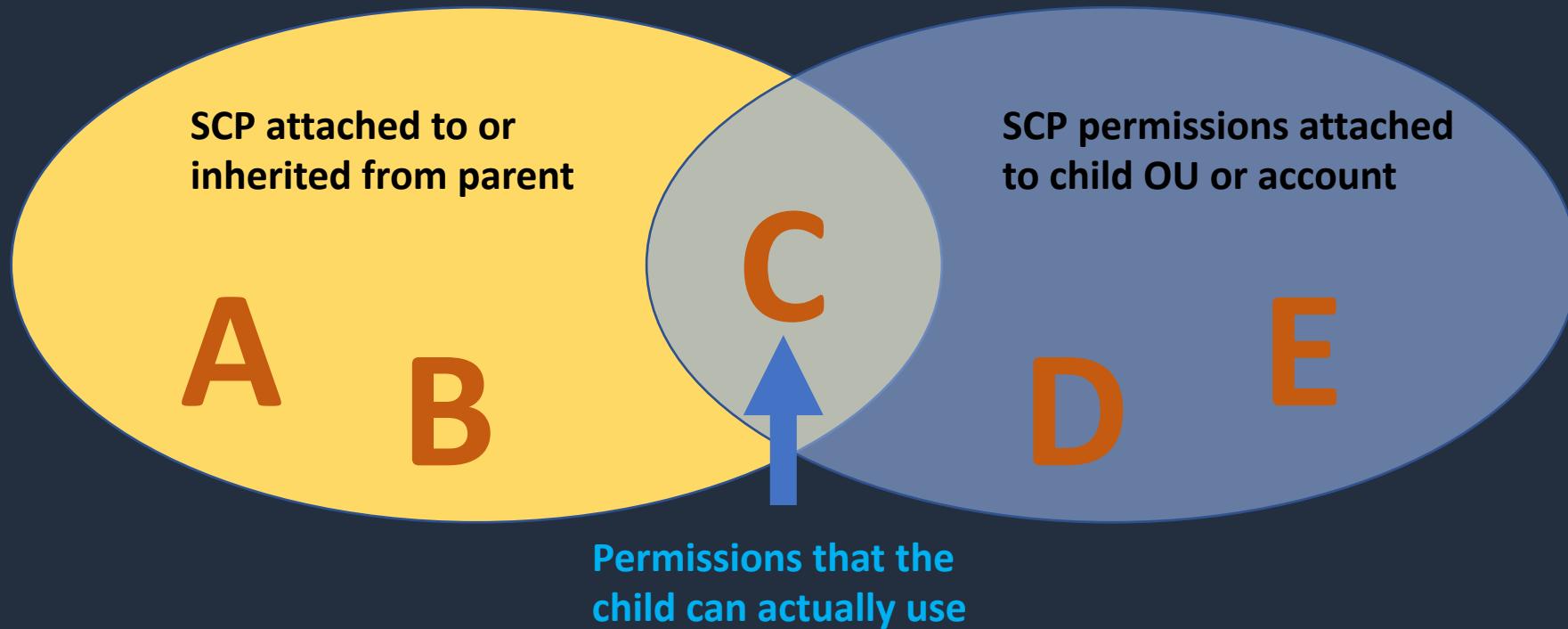
# SCP Strategies and Inheritance





# SCP Strategies and Inheritance

---





# SCP Strategies and Inheritance

---

## Deny List Strategy

- The **FullAWSAccess** SCP is attached to every OU and account
- Explicitly allows all permissions to flow down from the root
- Can explicitly override with a deny in an SCP
- This is the default setup

Note: An **explicit deny** overrides any kind of **allow**

## Allow List Strategy

- The **FullAWSAccess** SCP is **removed** from every OU and account
- To allow a permission, SCPs with allow statements must be added to the account and every OU above it including root
- Every SCP in the hierarchy must explicitly allow the APIs you want to use

Note: An **explicit allow** overrides an **implicit deny**

# Create Development Account



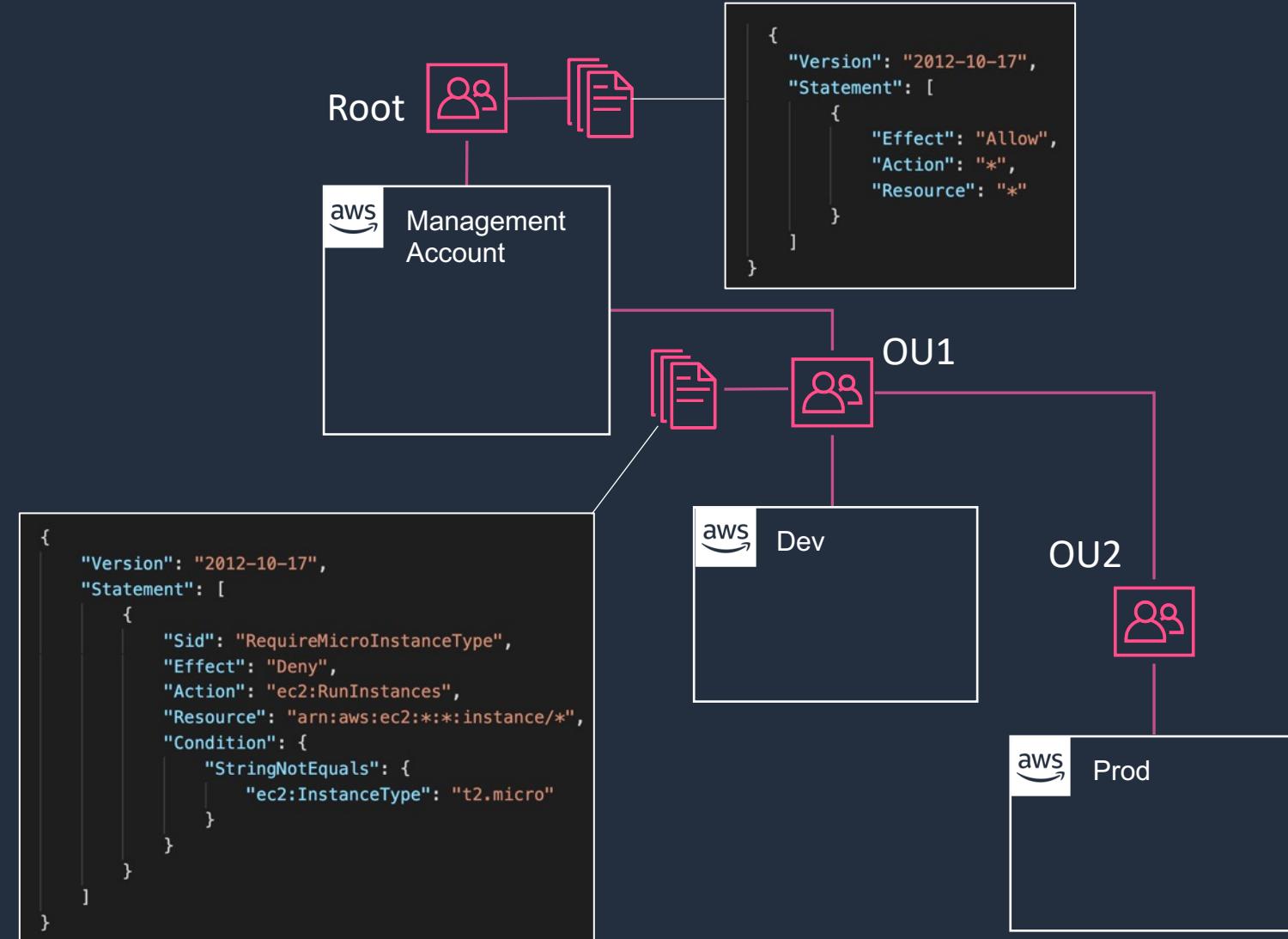
# Test SCP Inheritance



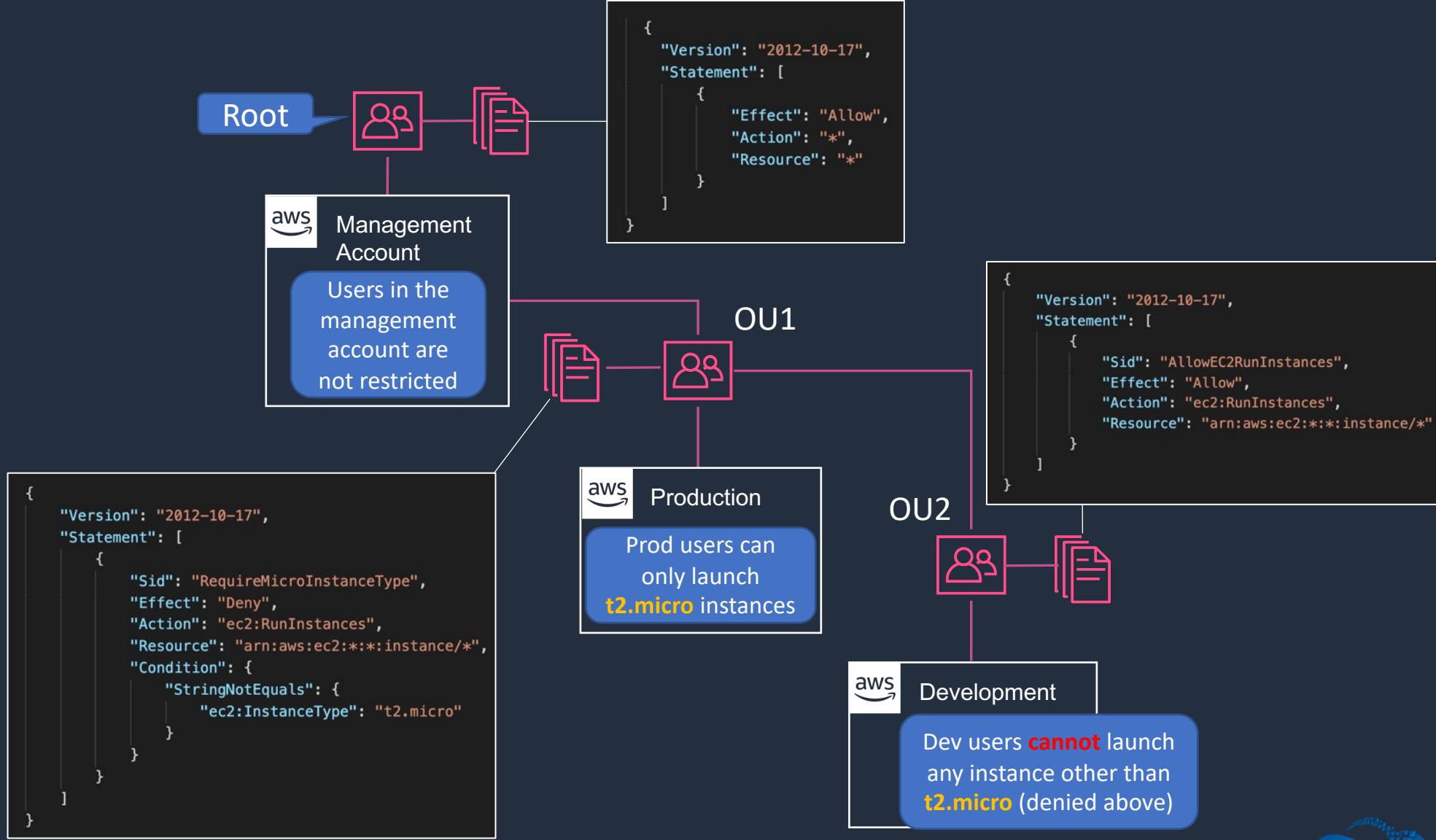


# Service Control Policies

---



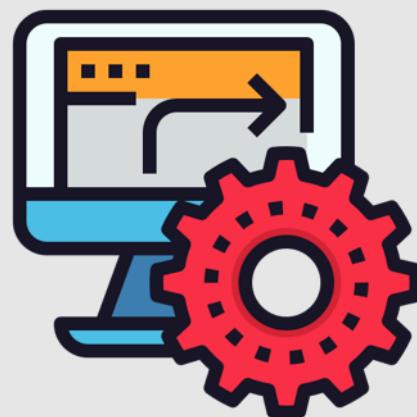
# Service Control Policies



# SECTION 3

## Identity Management and Permissions

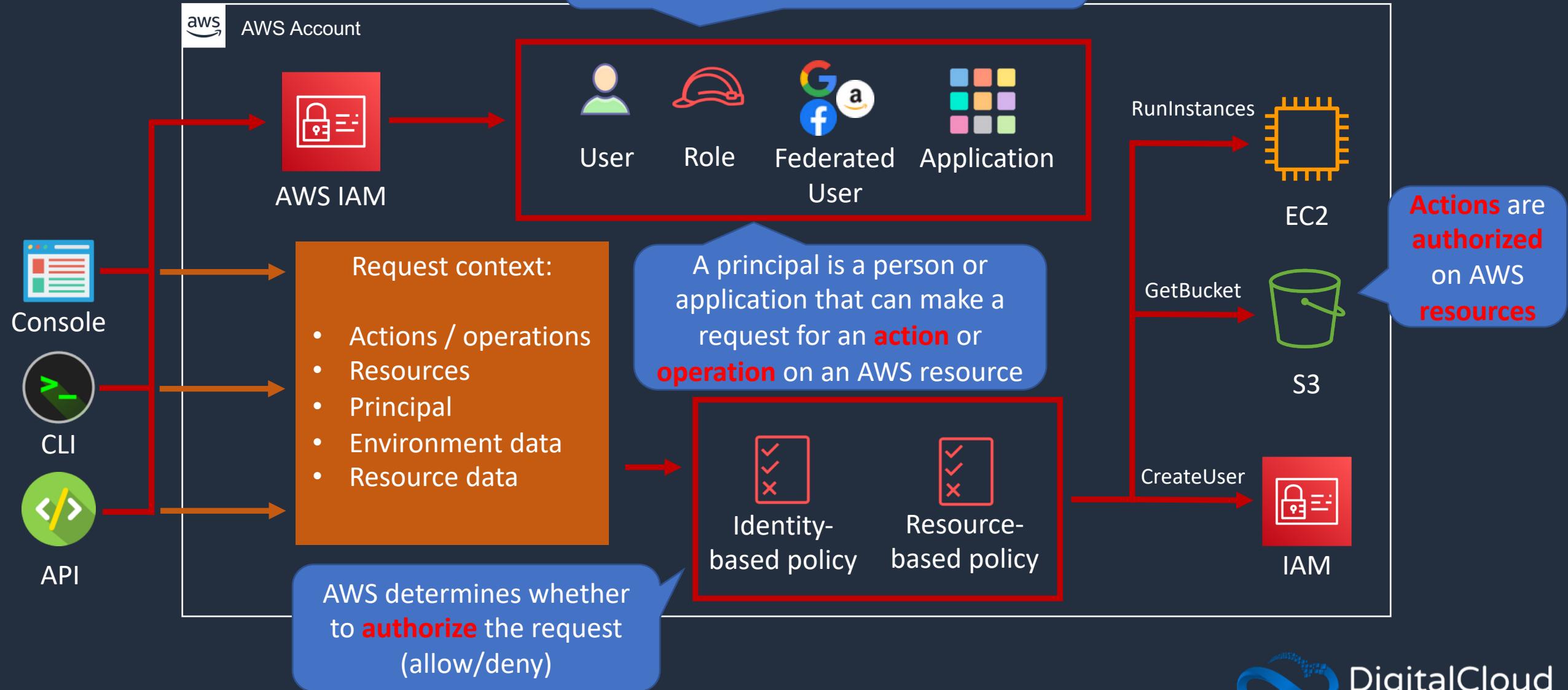
# How IAM Works



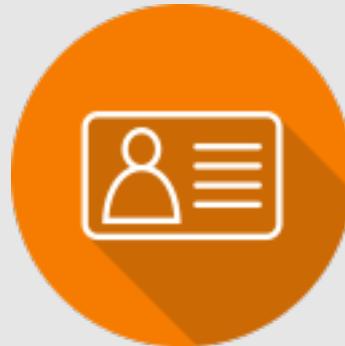


# How IAM Works

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

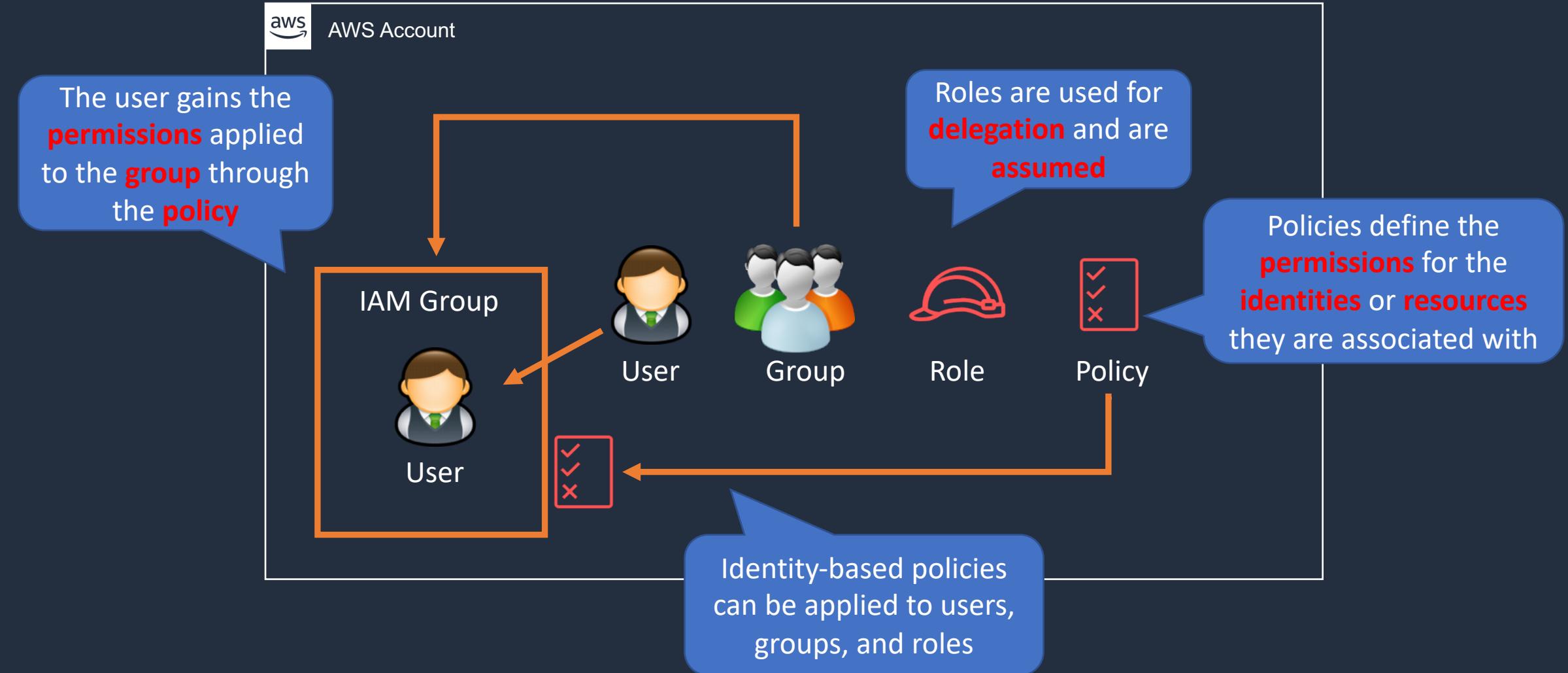


# Overview of Users, Groups, Roles and Policies



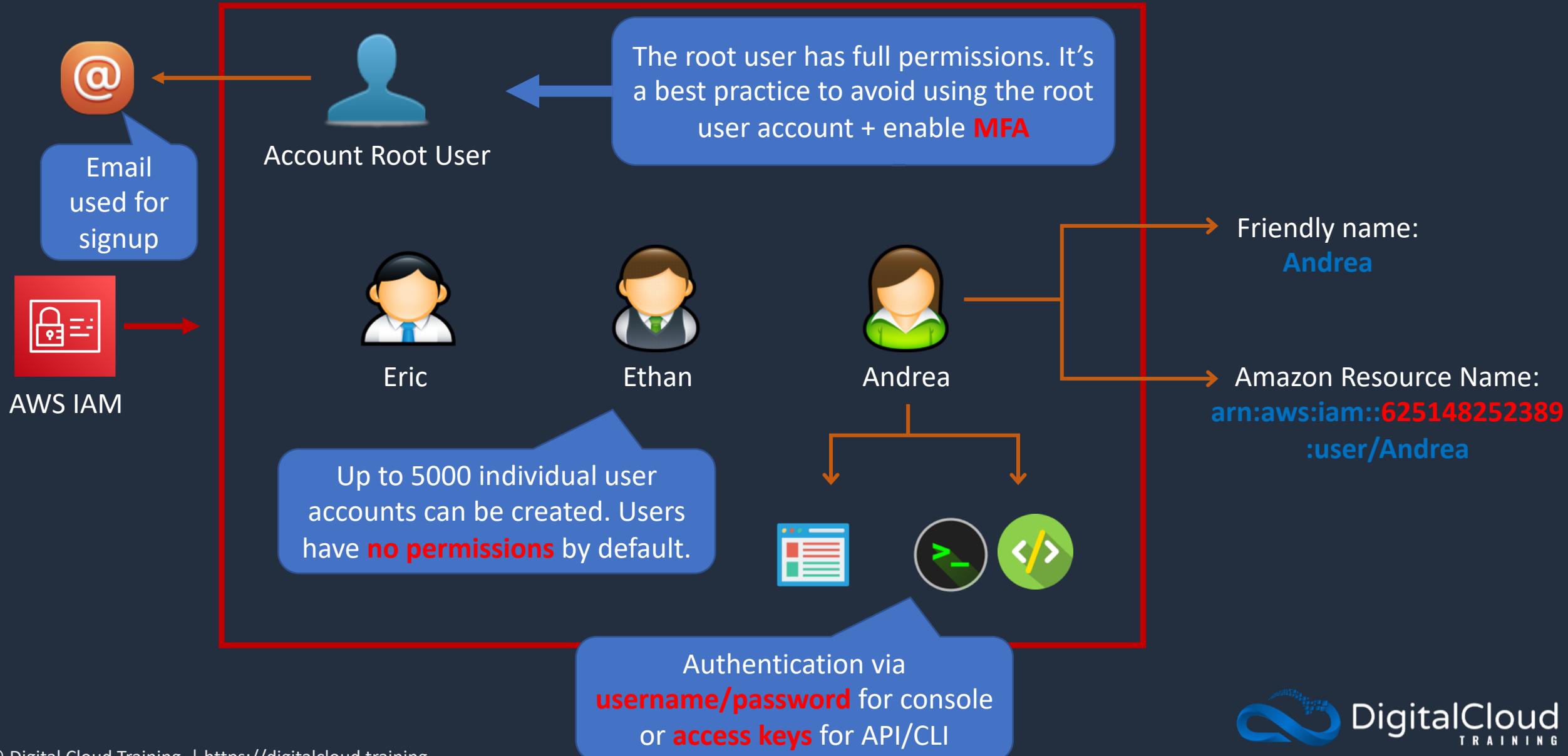


# Users, Groups, Roles and Policies



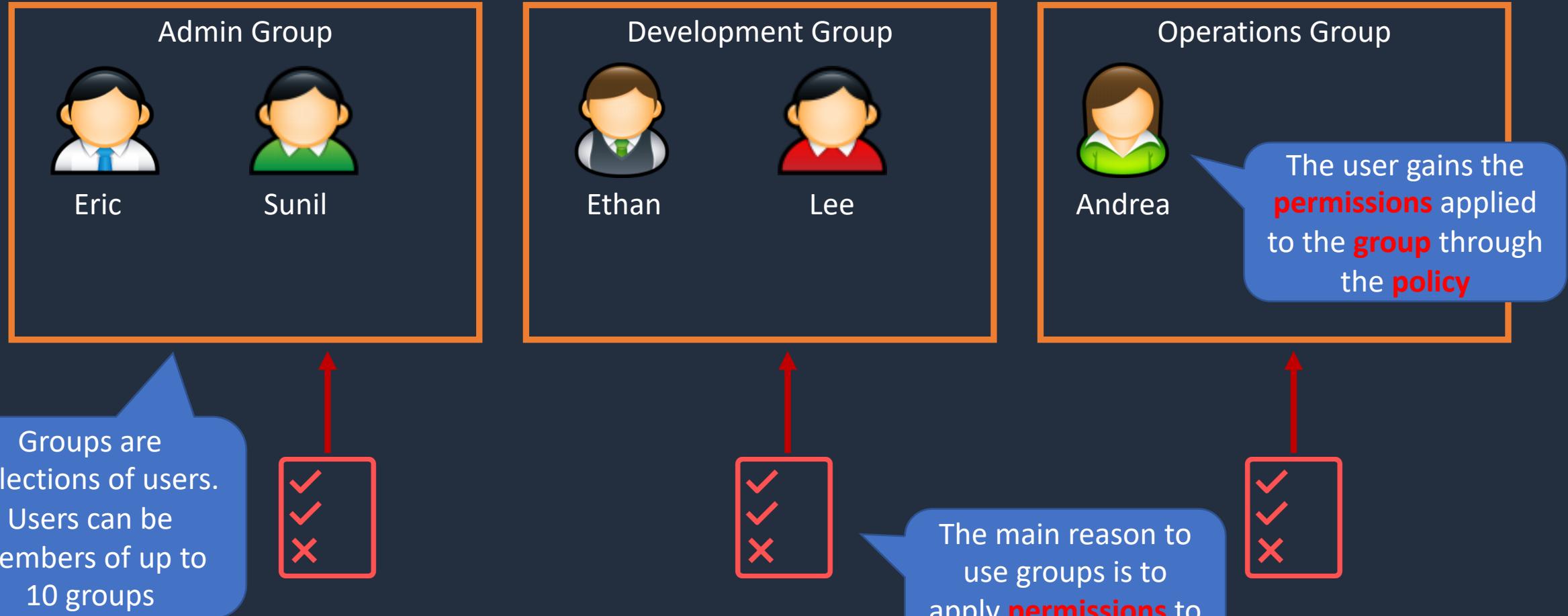


# IAM Users





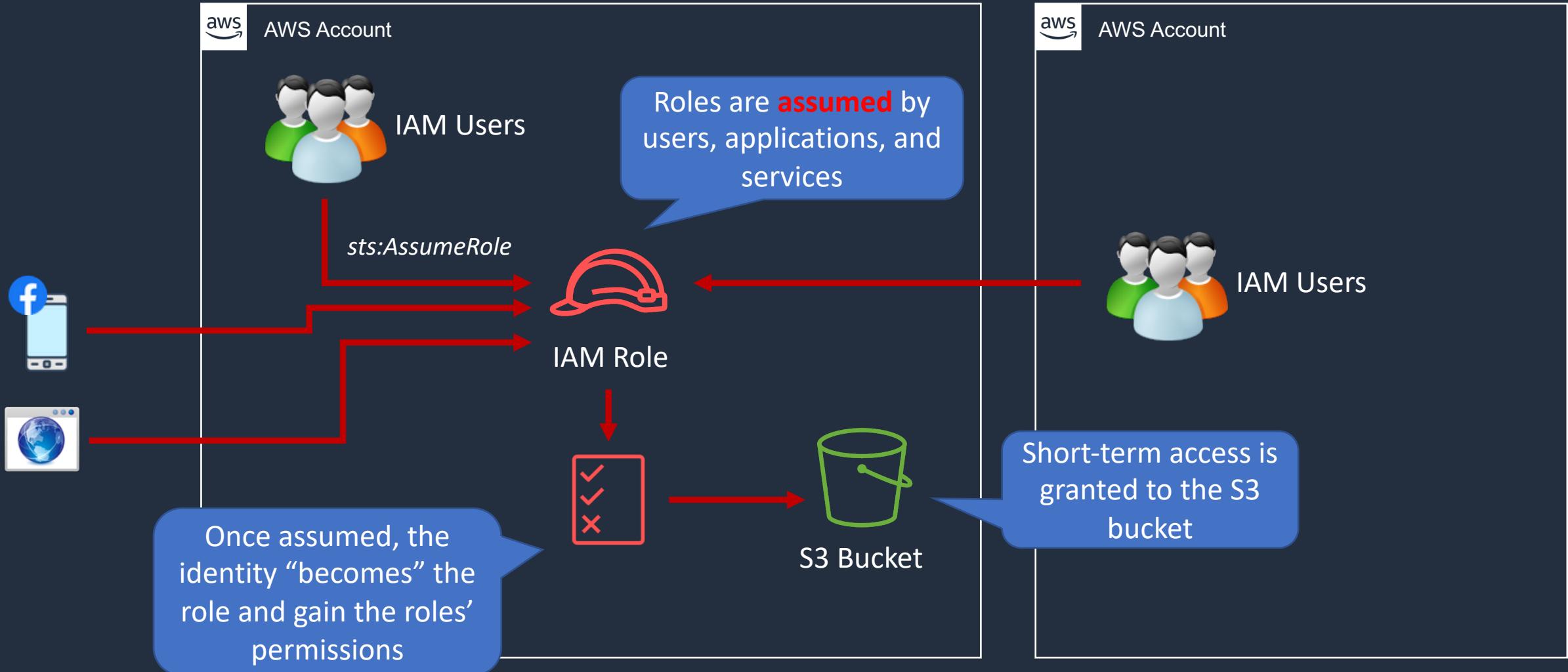
# IAM Groups





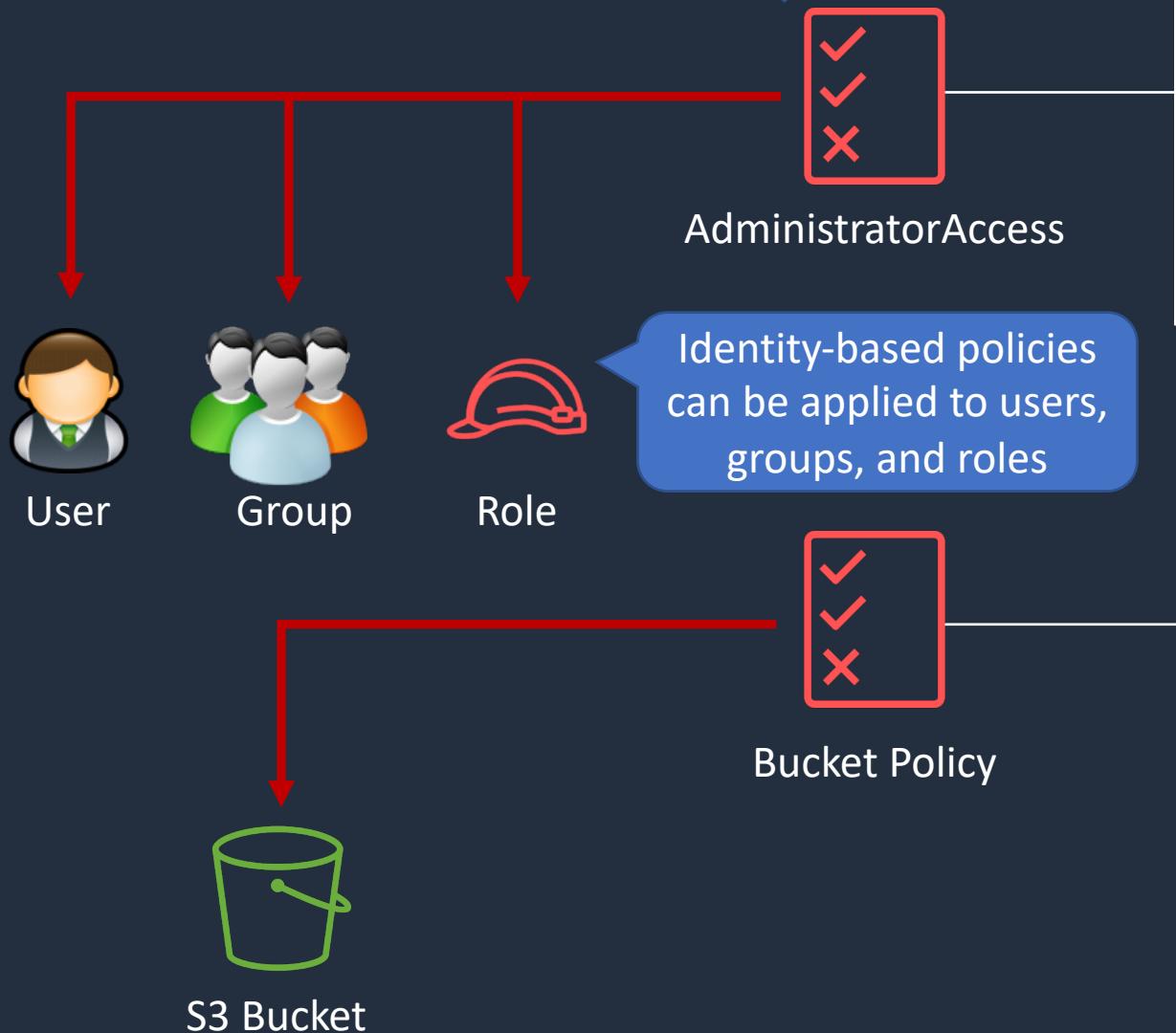
# IAM Roles

An IAM role is an IAM identity that has specific permissions





# IAM Policies



Policies are documents that define permissions and are written in JSON

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

All permissions are implicitly denied by default

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

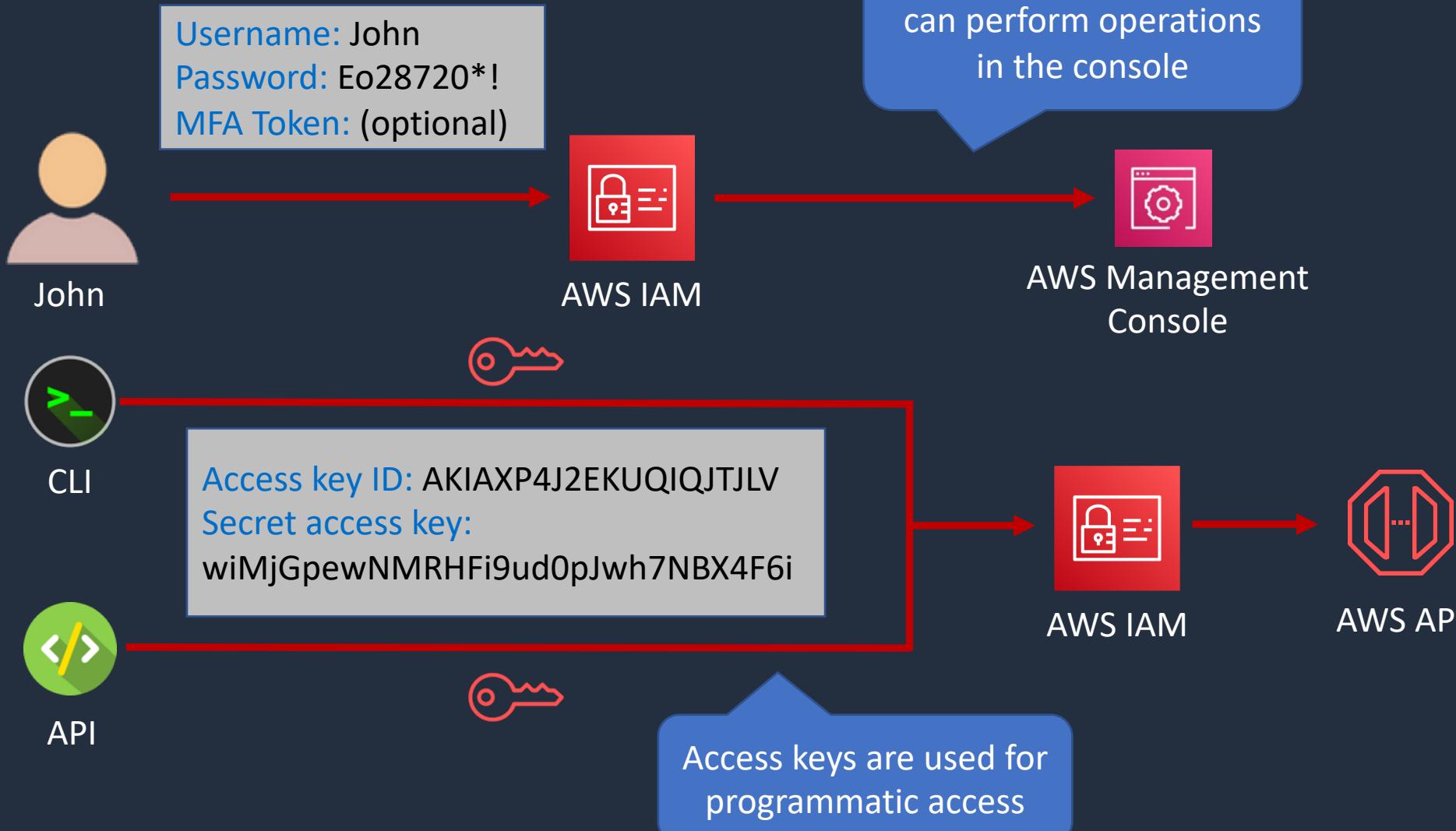
Resource-based policies apply to resources such as S3 buckets or DynamoDB tables

# IAM Authentication Methods



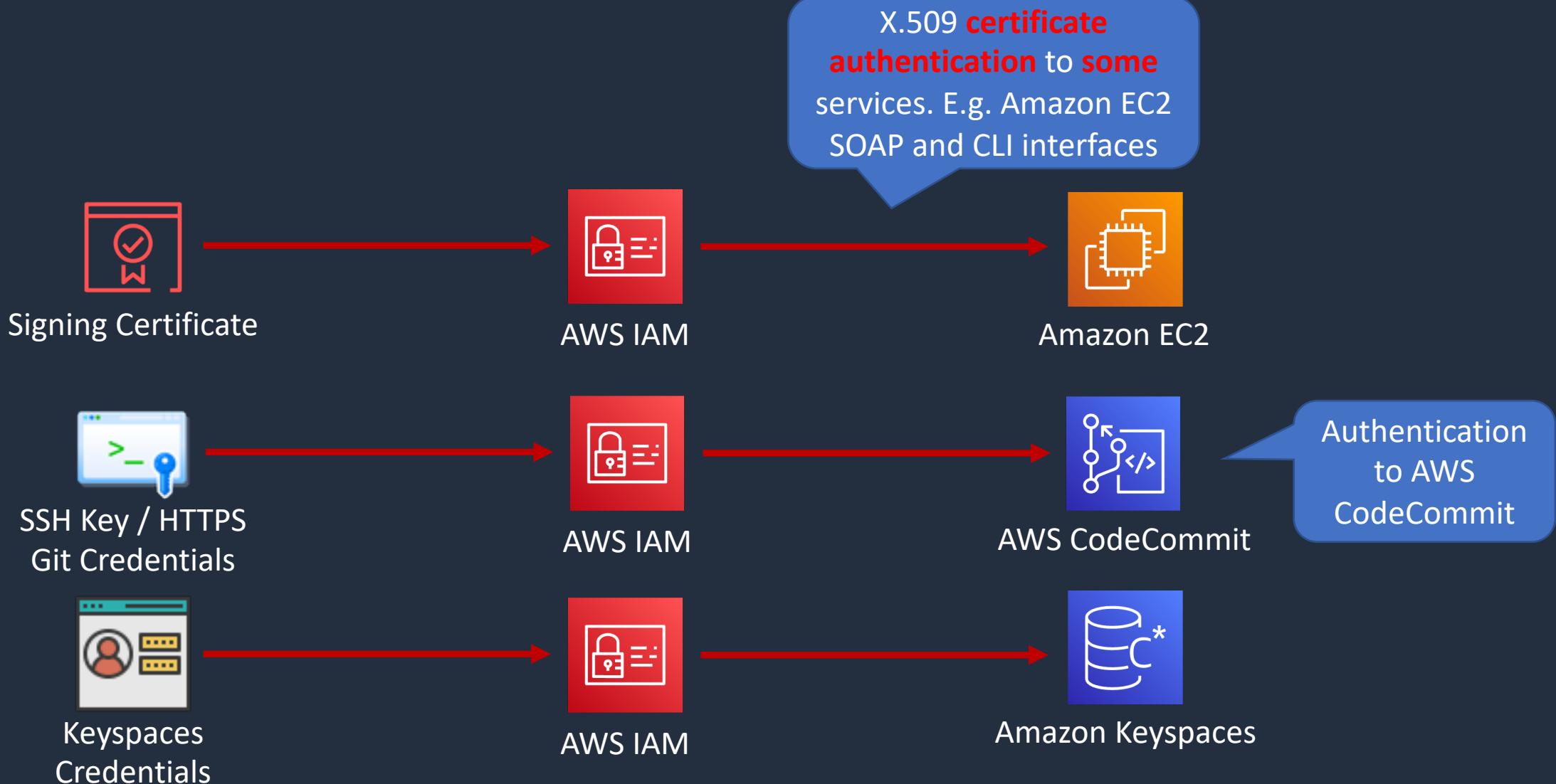


# IAM Authentication Methods





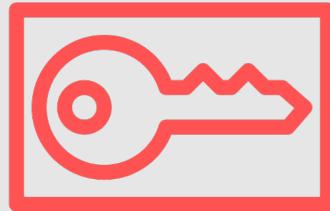
# IAM Authentication Methods



# Create User, Group, and Configure CLI

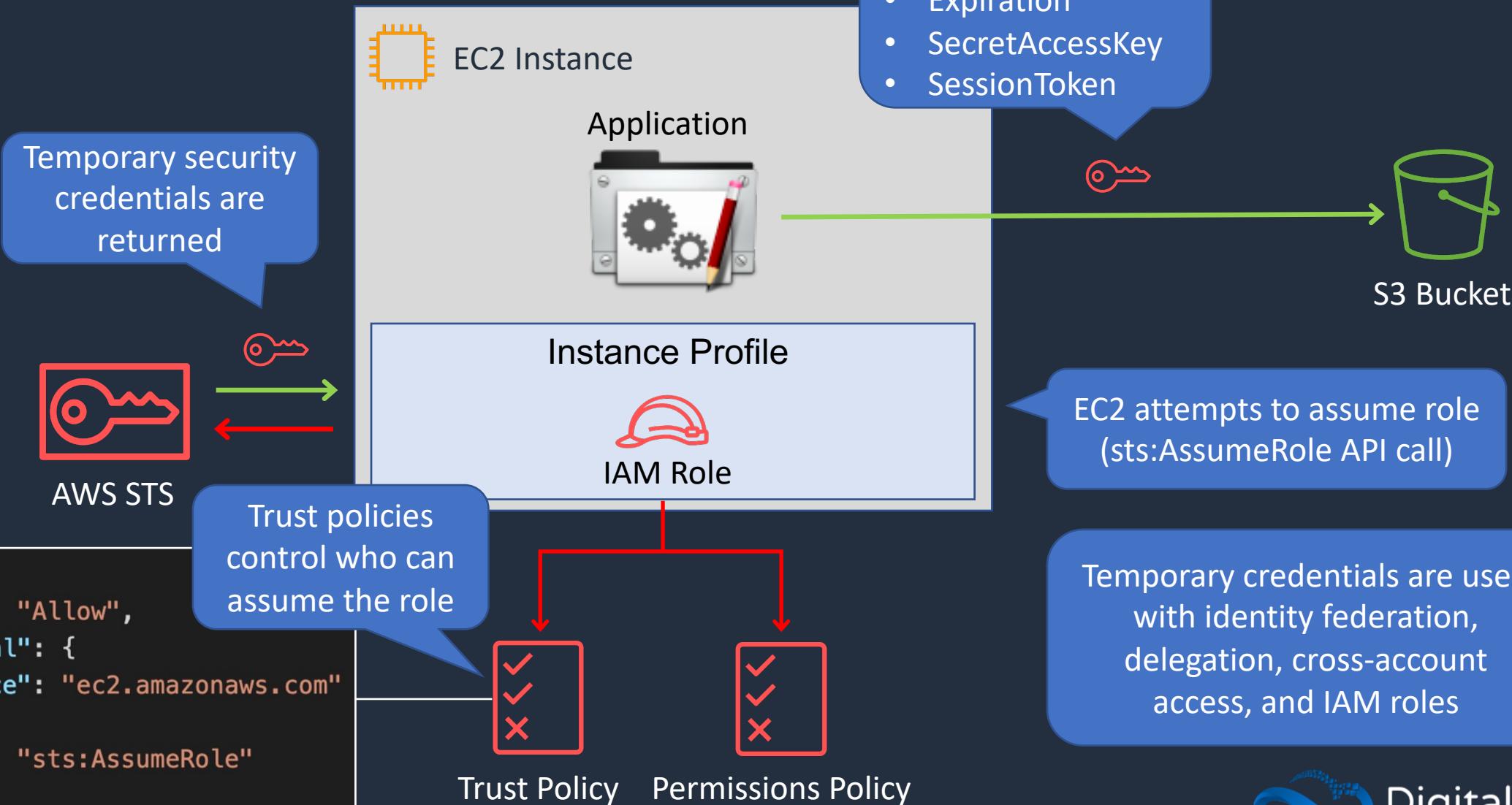


# AWS Security Token Service (STS)





# AWS Security Token Service (STS)



# Multi-Factor Authentication (MFA)



# Multi-Factor Authentication

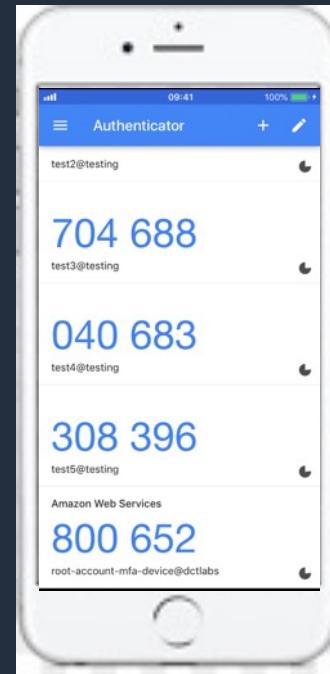
---

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**:



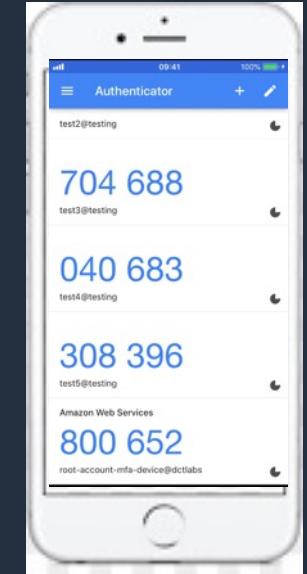
Virtual MFA



Physical MFA



e.g. Google Authenticator on  
your smart phone



# Secure the AWS Account



# 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



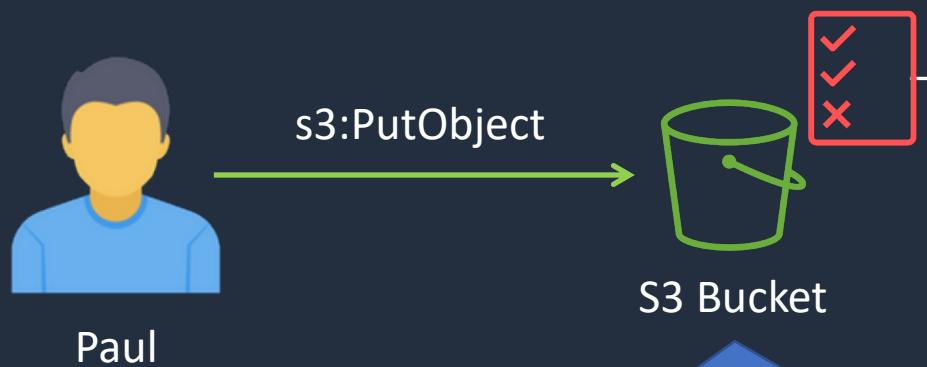
Inline policies have a 1-1 relationship with the user, group, or role



# Resource-Based Policies

---

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



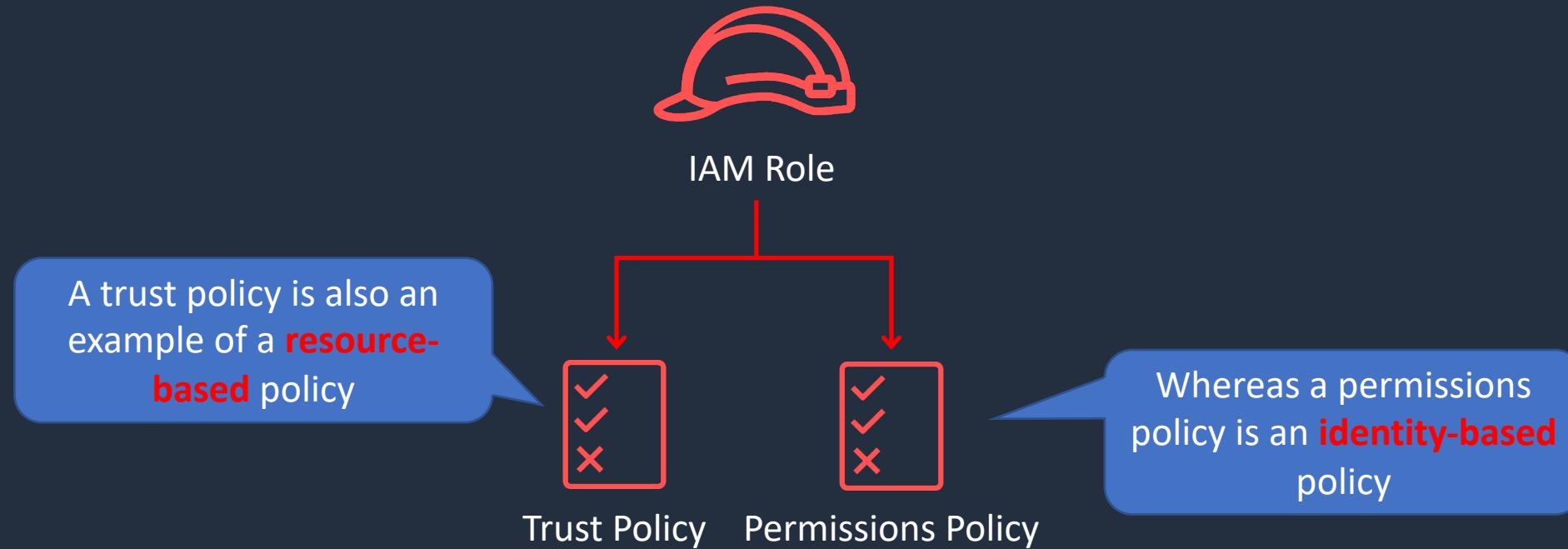
Resource-based policies grant the specified **principal** (Paul) **permission** to perform specific **actions** on the **resource**

```
{  
  "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

---

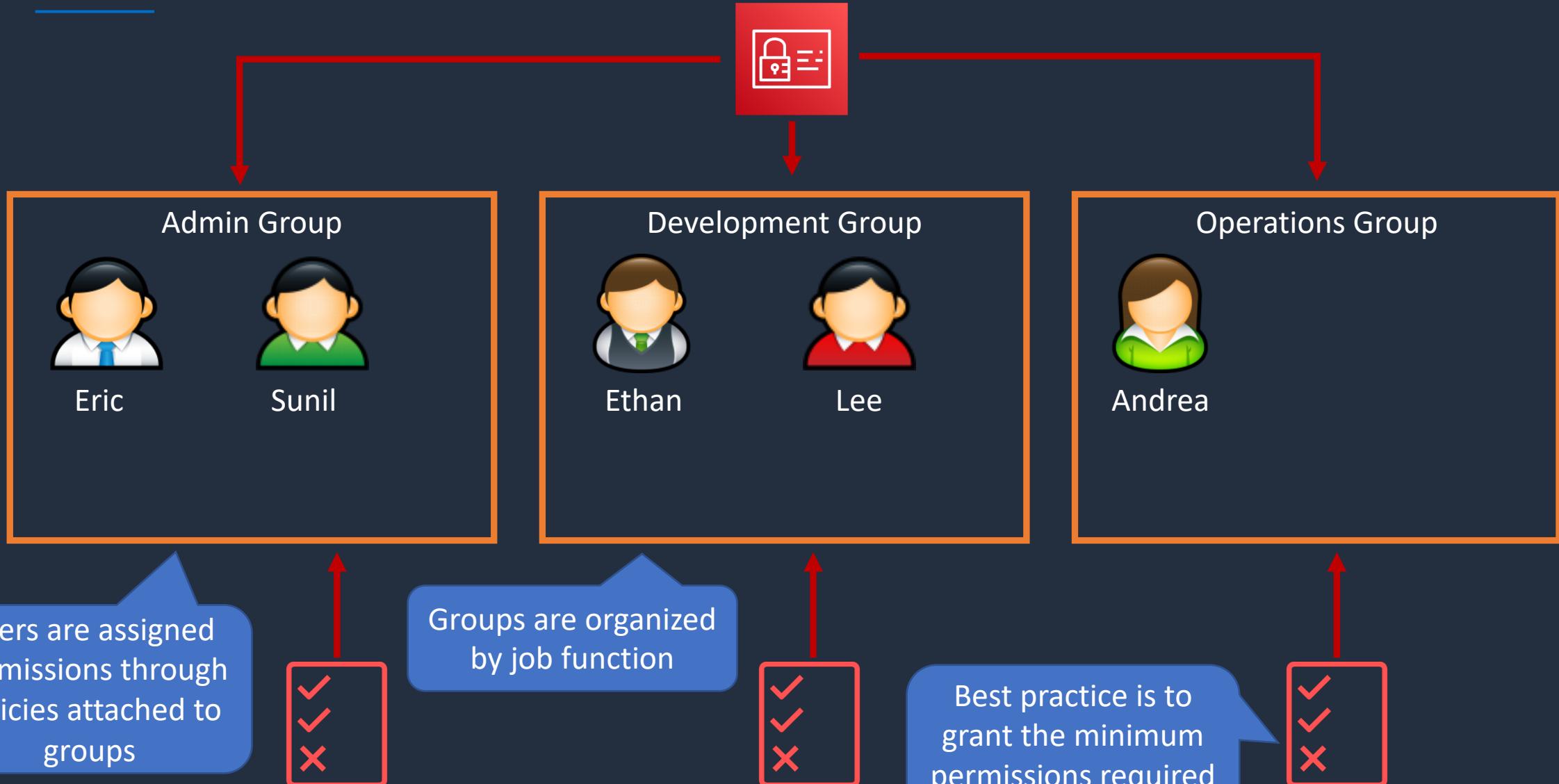


# Access Control Methods - RBAC & ABAC





# Role-Based Access Control (RBAC)



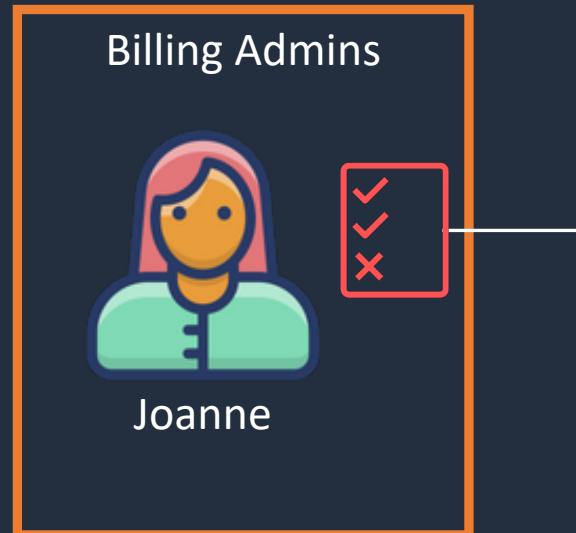


# Role-Based Access Control (RBAC)

## Job function policies:

- Administrator
- Billing
- Database administrator
- Data scientist
- Developer power user
- Network administrator
- Security auditor
- Support user
- System administrator
- View-only user

The Billing managed policy is attached to the group

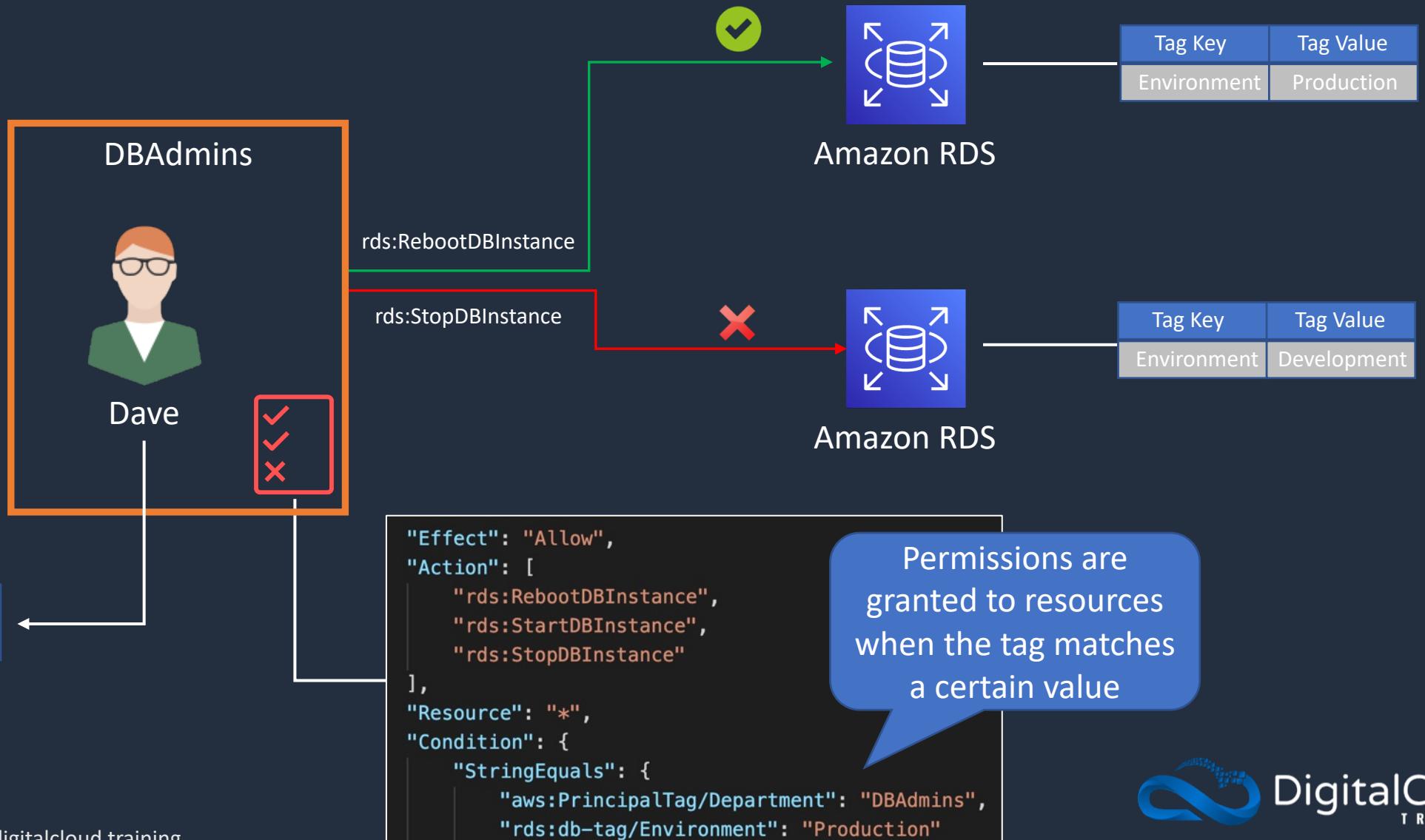


AWS managed policies for job functions are designed to closely align to common job functions in the IT industry

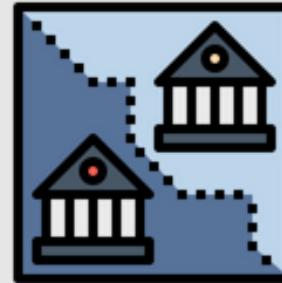
```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "aws-portal:*Billing",  
                "aws-portal:*Usage",  
                "aws-portal:*PaymentMethods",  
                "budgets:ViewBudget",  
                "budgets:ModifyBudget",  
                "ce:UpdatePreferences",  
                "ce>CreateReport",  
                "ce:UpdateReport",  
                "ce>DeleteReport",  
                "ce>CreateNotificationSubscription",  
                "ce:UpdateNotificationSubscription",  
                "ce>DeleteNotificationSubscription",  
                "cur:DescribeReportDefinitions",  
                "cur:PutReportDefinition",  
                "cur:ModifyReportDefinition",  
                "cur>DeleteReportDefinition",  
                "purchase-orders:*PurchaseOrders"  
            ],  
            "Resource": "*"  
        }  
    ]  
}
```



# Attribute-Based Access Control (ABAC)

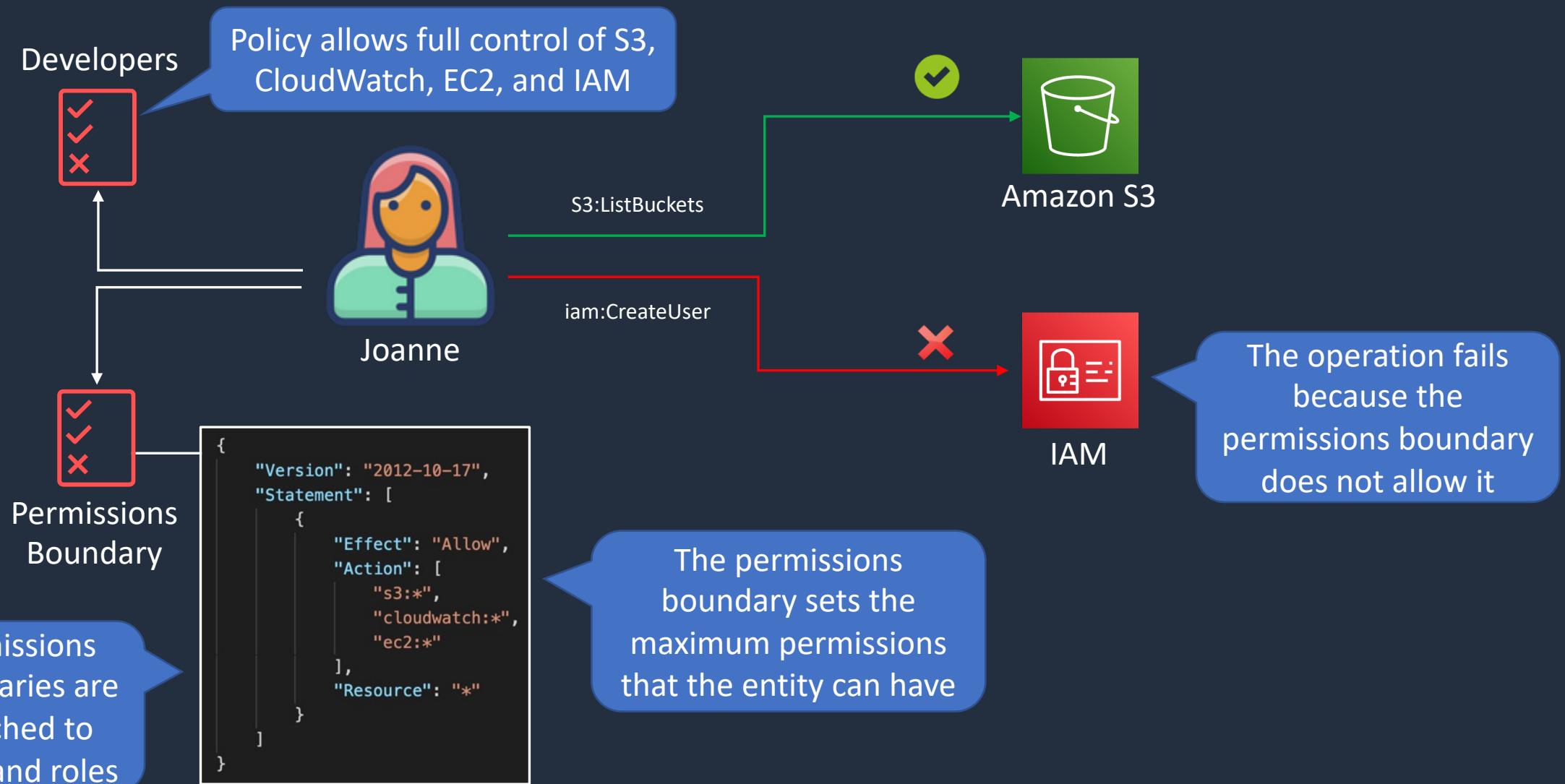


# Permissions Boundaries





# Permissions Boundaries





# Privilege Escalation

IAMFullAccess



Lindsay

Lindsay is assigned permissions to AWS IAM only and cannot launch AWS resources

iam:CreateUser



IAM

Lindsay applies the AdministratorAccess policy to the X-User account

AdministratorAccess



X-User

Lindsay mines bitcoins



AWS Batch



Lindsay is now able to login with the X-User account and gain full privileges to the AWS account



# Preventing Privilege Escalation

IAMFullAccess



Lindsay is assigned permissions to AWS IAM only and cannot launch AWS resources

iam:CreateUser



IAM

Permissions Boundary

The permissions boundary ensures that users created by Lindsay have the same or fewer permissions

Lindsay applies the AdministratorAccess policy to the X-User account

AdministratorAccess



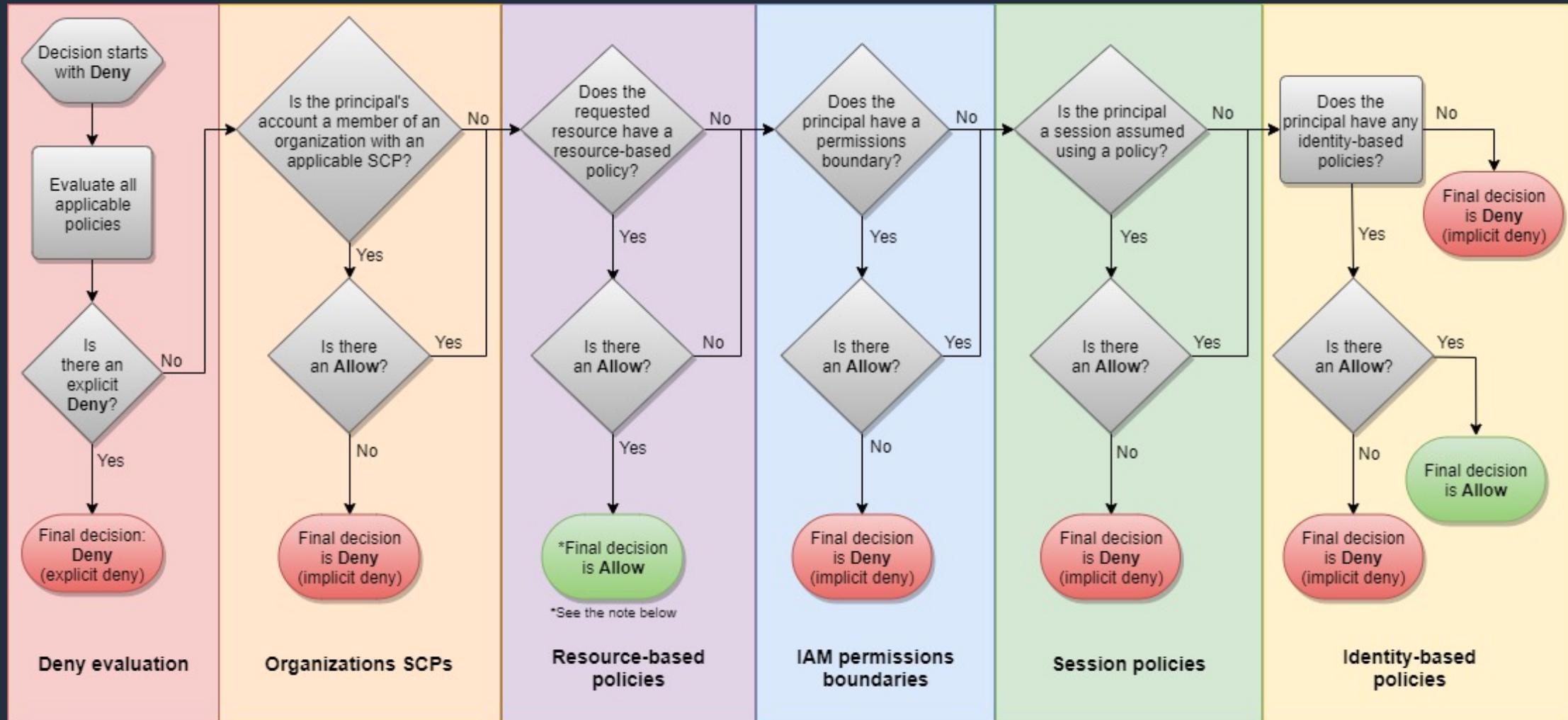
Lindsay does not have more privileges when logging in as X-User and cannot launch AWS resources

# IAM Policy Evaluation Logic





# Evaluation Logic





# Steps for Authorizing Requests to AWS

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



AWS IAM



User



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

- 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

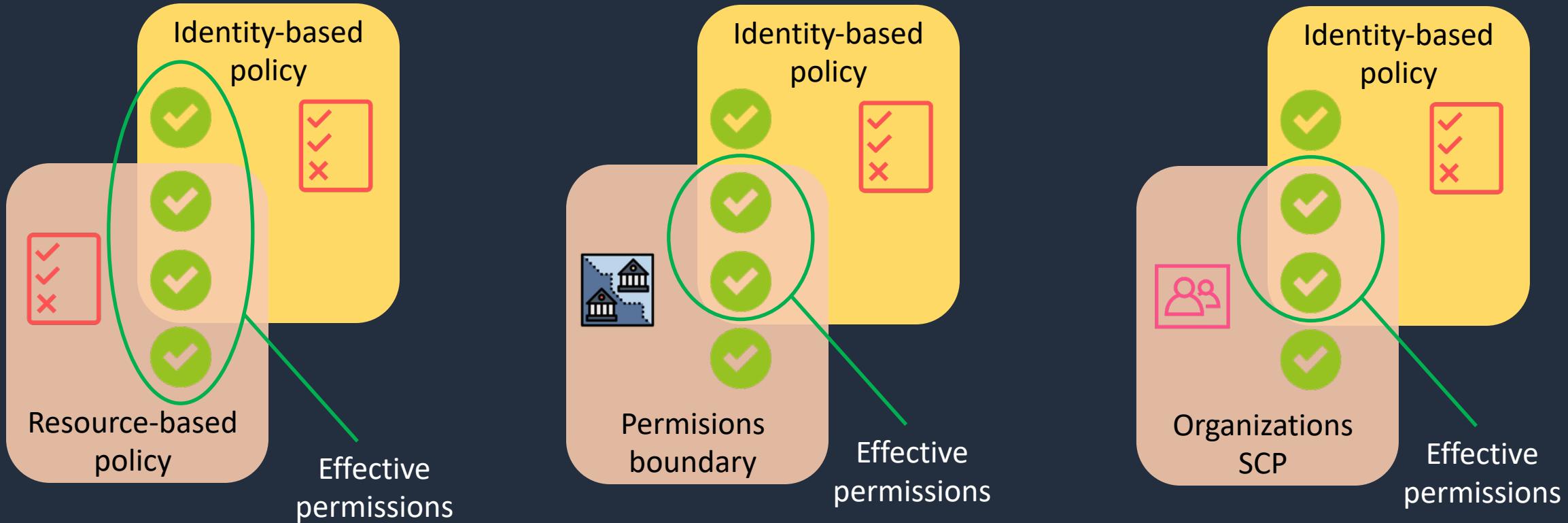


# Types of Policy

- **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

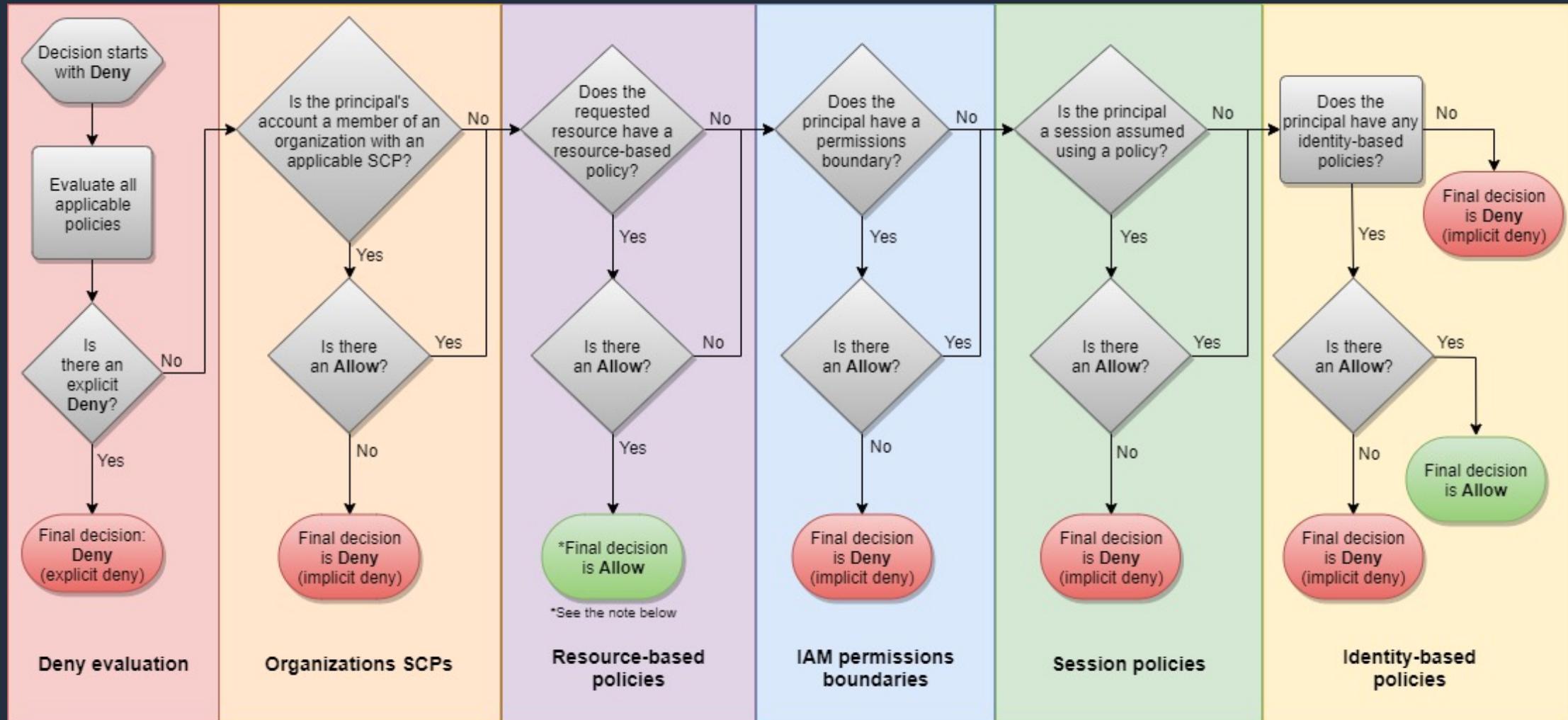
---

---

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

{

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

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

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

# Using Role-Based Access Control (RBAC)



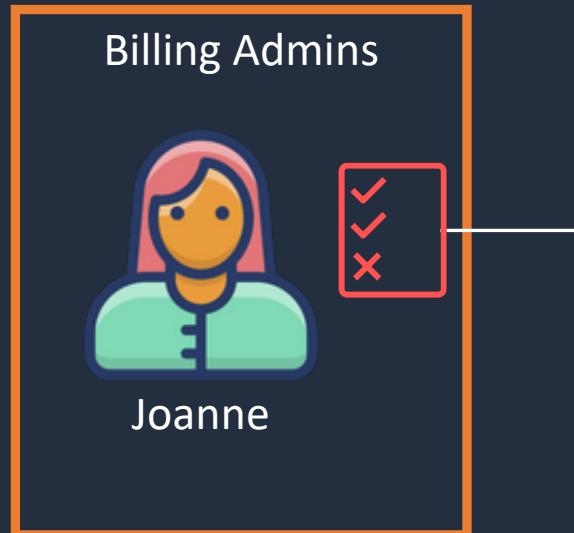


# Role-Based Access Control (RBAC)

## Job function policies:

- Administrator
- Billing
- Database administrator
- Data scientist
- Developer power user
- Network administrator
- Security auditor
- Support user
- System administrator
- View-only user

The Billing managed policy is attached to the group



AWS managed policies for job functions are designed to closely align to common job functions in the IT industry

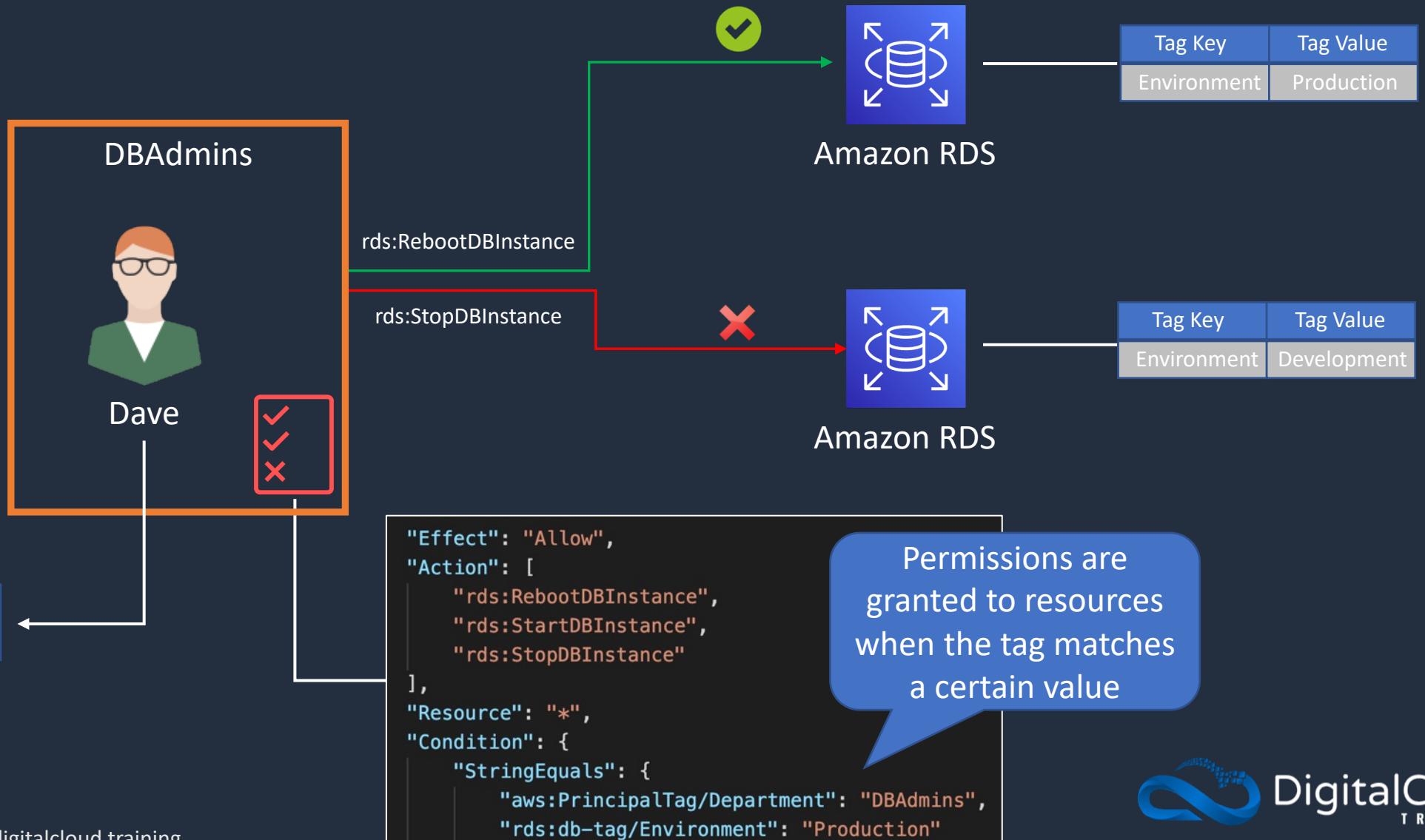
```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Action": [  
        "aws-portal:*Billing",  
        "aws-portal:*Usage",  
        "aws-portal:*PaymentMethods",  
        "budgets:ViewBudget",  
        "budgets:ModifyBudget",  
        "ce:UpdatePreferences",  
        "ce>CreateReport",  
        "ce:UpdateReport",  
        "ce>DeleteReport",  
        "ce>CreateNotificationSubscription",  
        "ce:UpdateNotificationSubscription",  
        "ce>DeleteNotificationSubscription",  
        "cur:DescribeReportDefinitions",  
        "cur:PutReportDefinition",  
        "cur:ModifyReportDefinition",  
        "cur>DeleteReportDefinition",  
        "purchase-orders:*PurchaseOrders"  
      ],  
      "Resource": "*"  
    }  
  ]  
}
```

# Using Attribute-Based Access Control (ABAC)





# Attribute-Based Access Control (ABAC)



# Apply Permissions Boundary





# Permissions Boundary Hands-On Practice

---

\*\*\* Use the **PermissionsBoundary.json** file  
from the course download \*\*\*

The policy will enforce the following:

- IAM principals can't alter the permissions boundary to allow their own permissions to access restricted services
- IAM principals must attach the permissions boundary to any IAM principals they create
- IAM admins can't create IAM principals with more privileges than they already have
- The IAM principals created by IAM admins can't create IAM principals with more permissions than IAM admins



# Privilege Escalation

IAMFullAccess



Lindsay

Lindsay is assigned permissions to AWS IAM only and cannot launch AWS resources

iam:CreateUser



IAM

Lindsay applies the AdministratorAccess policy to the X-User account

AdministratorAccess



X-User

Lindsay mines bitcoins



AWS Batch



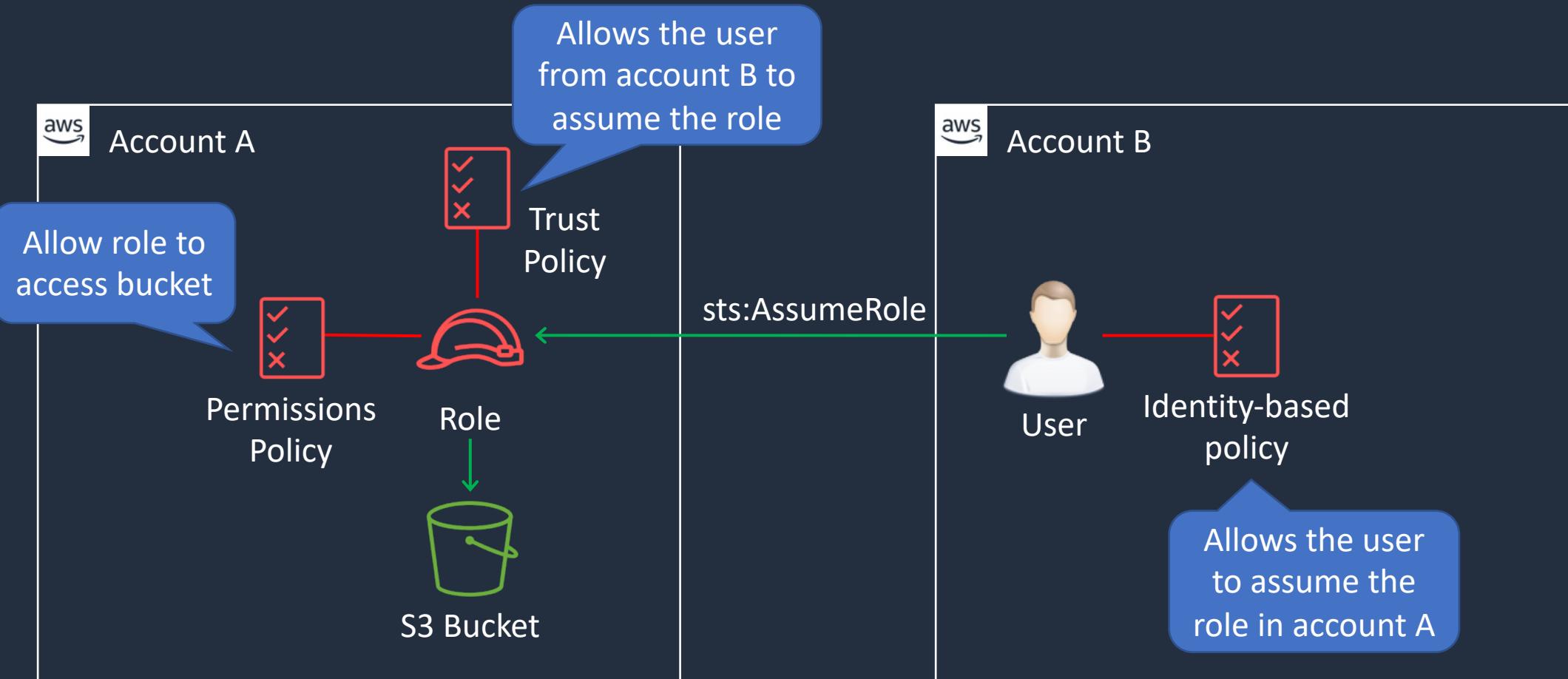
Lindsay is now able to login with the X-User account and gain full privileges to the AWS account

# Use Cases for IAM Roles





# Use Case: Cross Account Access

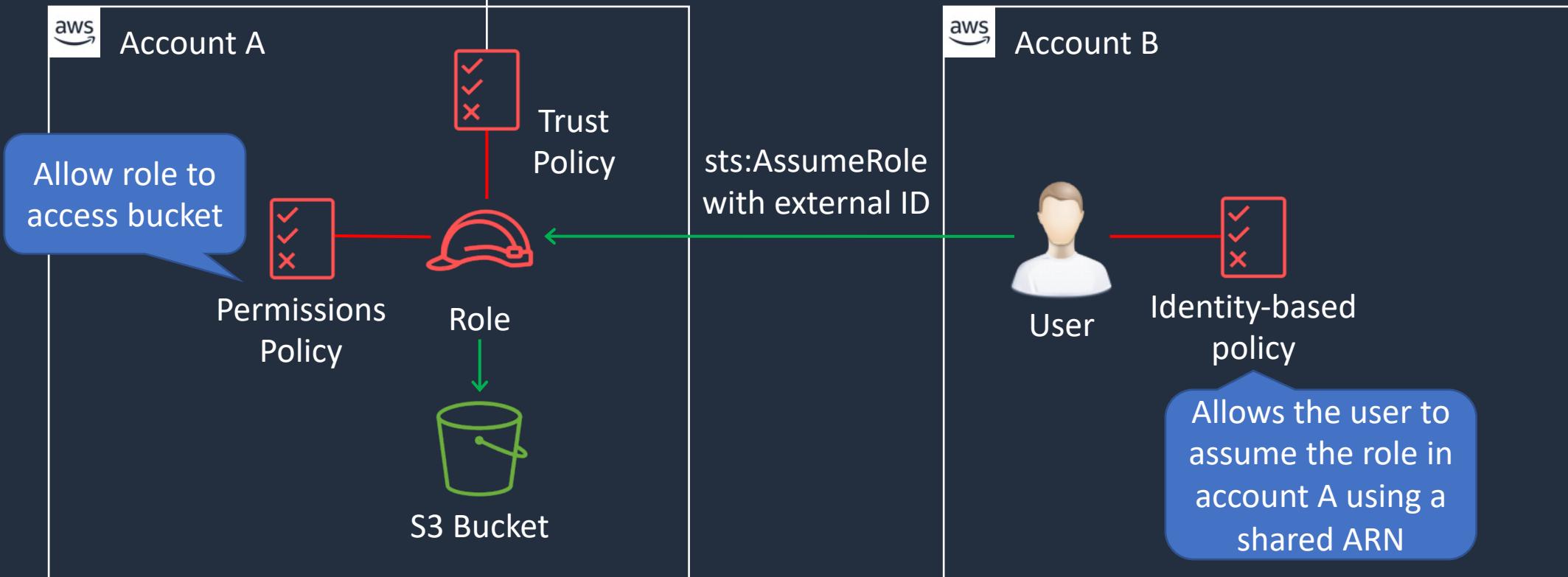




# Use Case: Cross Account Access (3<sup>rd</sup> Party)

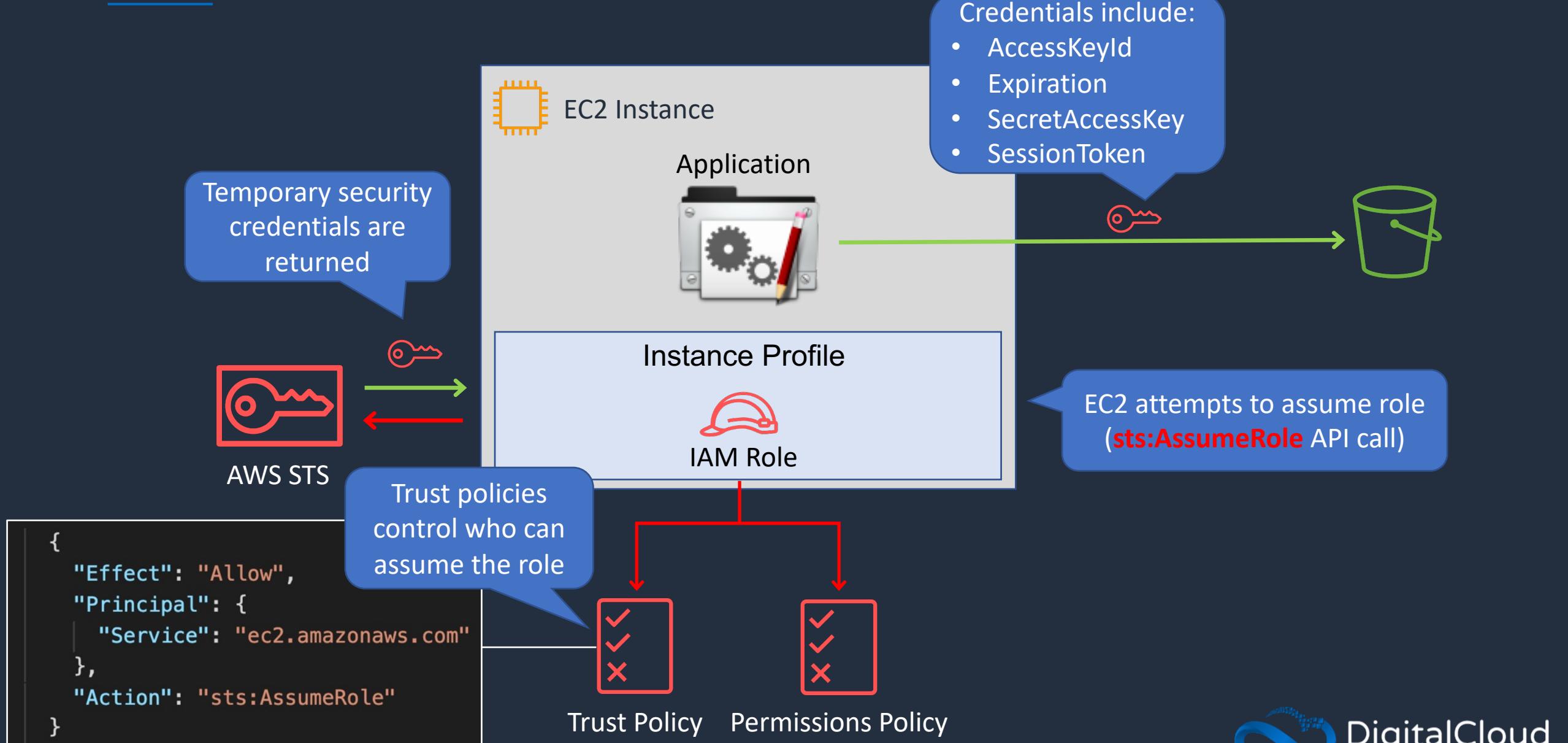
The trust policy condition requires the external ID

```
"Statement": {  
    "Effect": "Allow",  
    "Action": "sts:AssumeRole",  
    "Principal": {"AWS": "3rd party AWS Account ID"},  
    "Condition": {"StringEquals": {"sts:ExternalId": "12345"}}}
```





# Use Case: Delegation to AWS Services

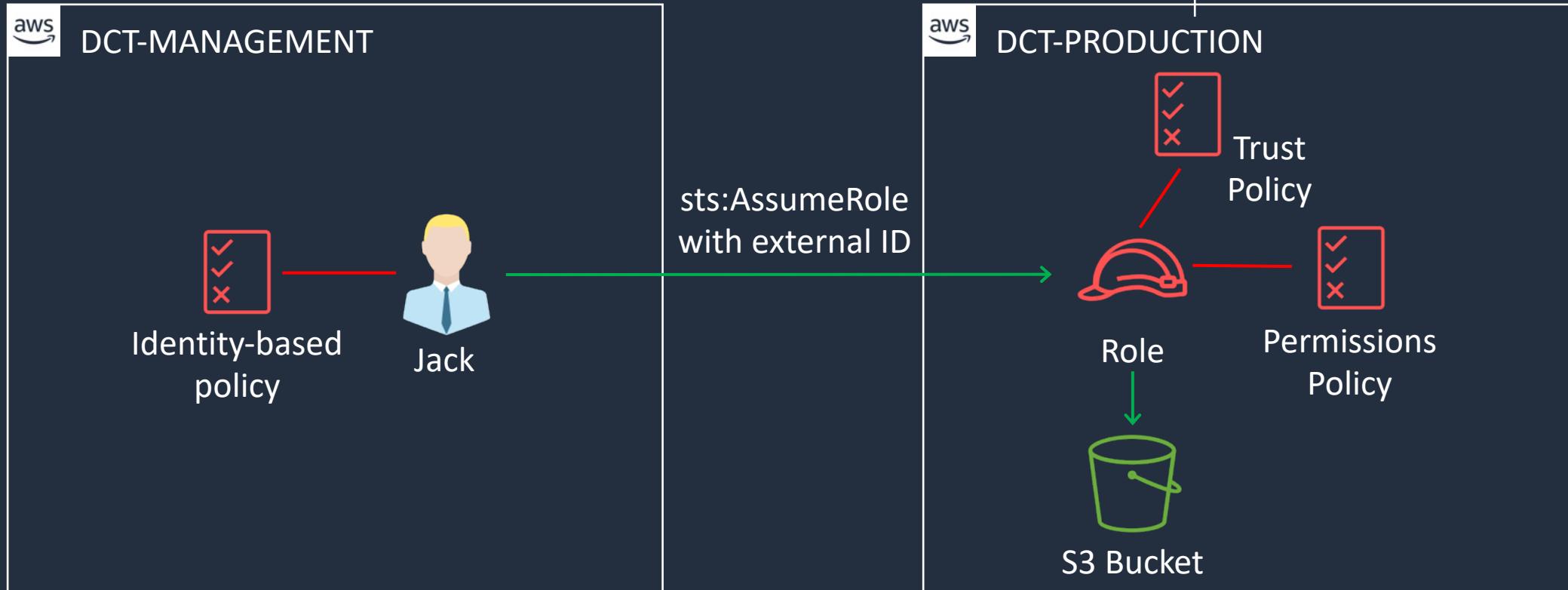


# Cross Account Access to Amazon S3





# Cross Account Access (3<sup>rd</sup> Party) Hands-On

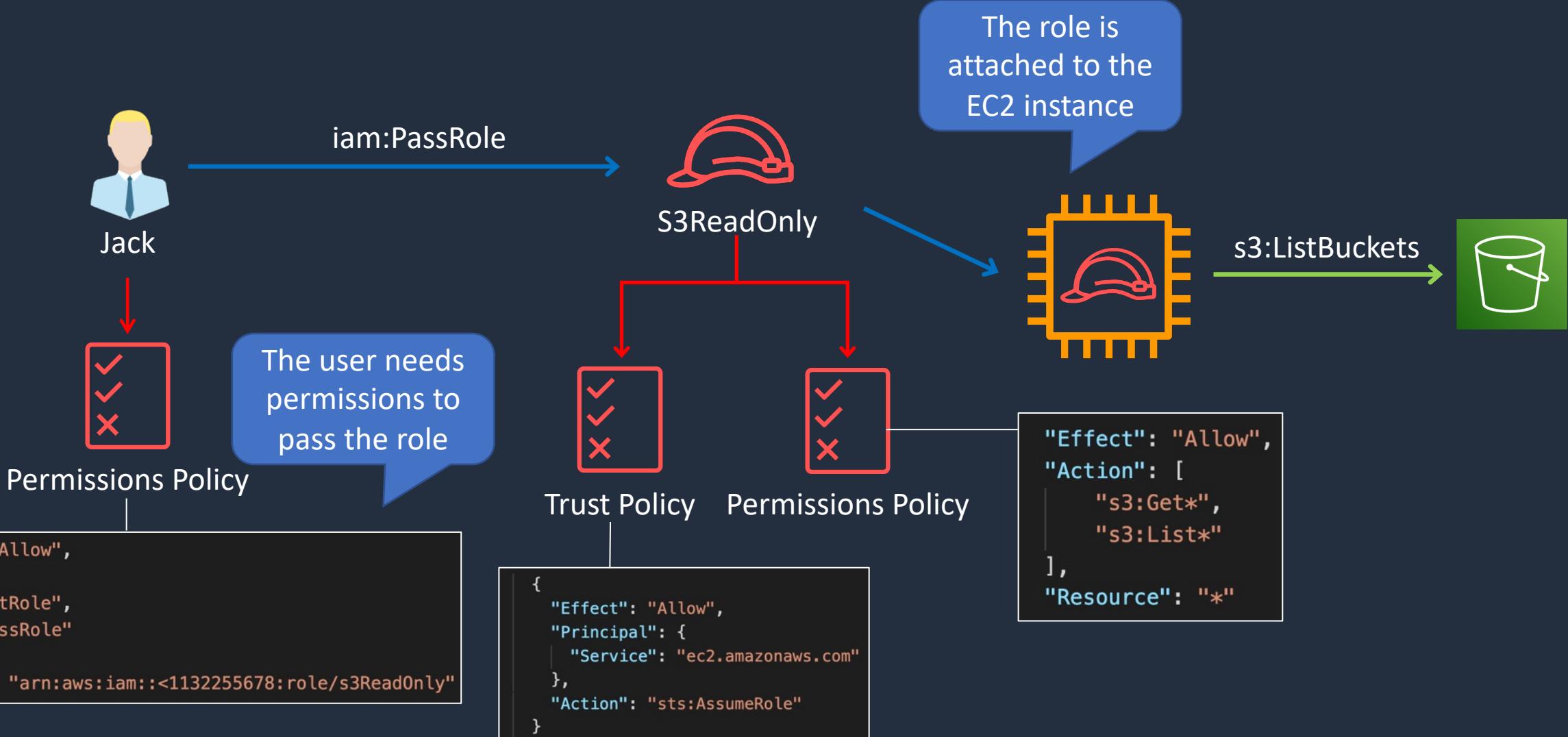


# Amazon EC2 Instance Profile





# Attach Role to EC2 Instance



# AWS 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

# SECTION 4

## AWS Directory Services and Federation

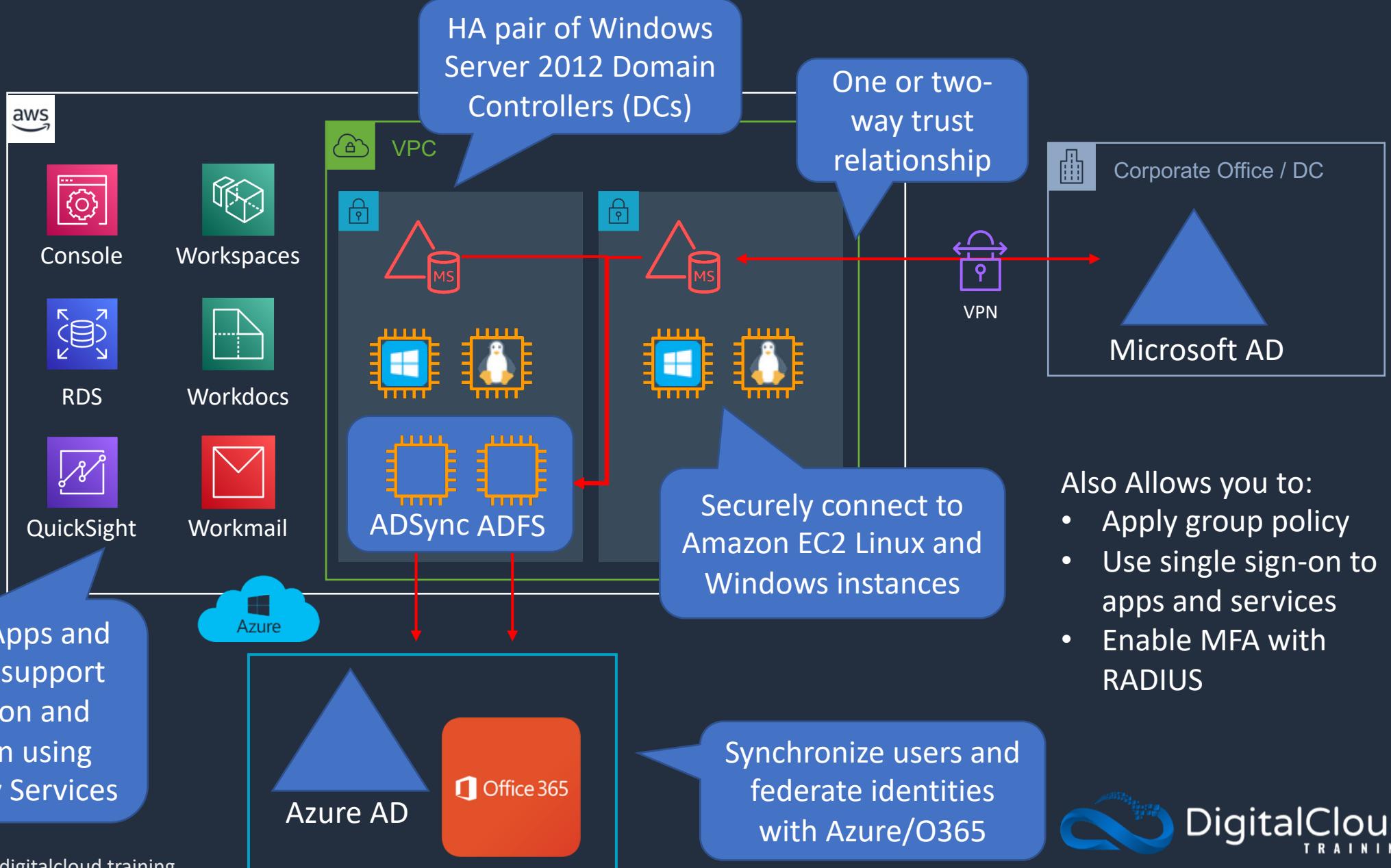
# AWS Directory Services



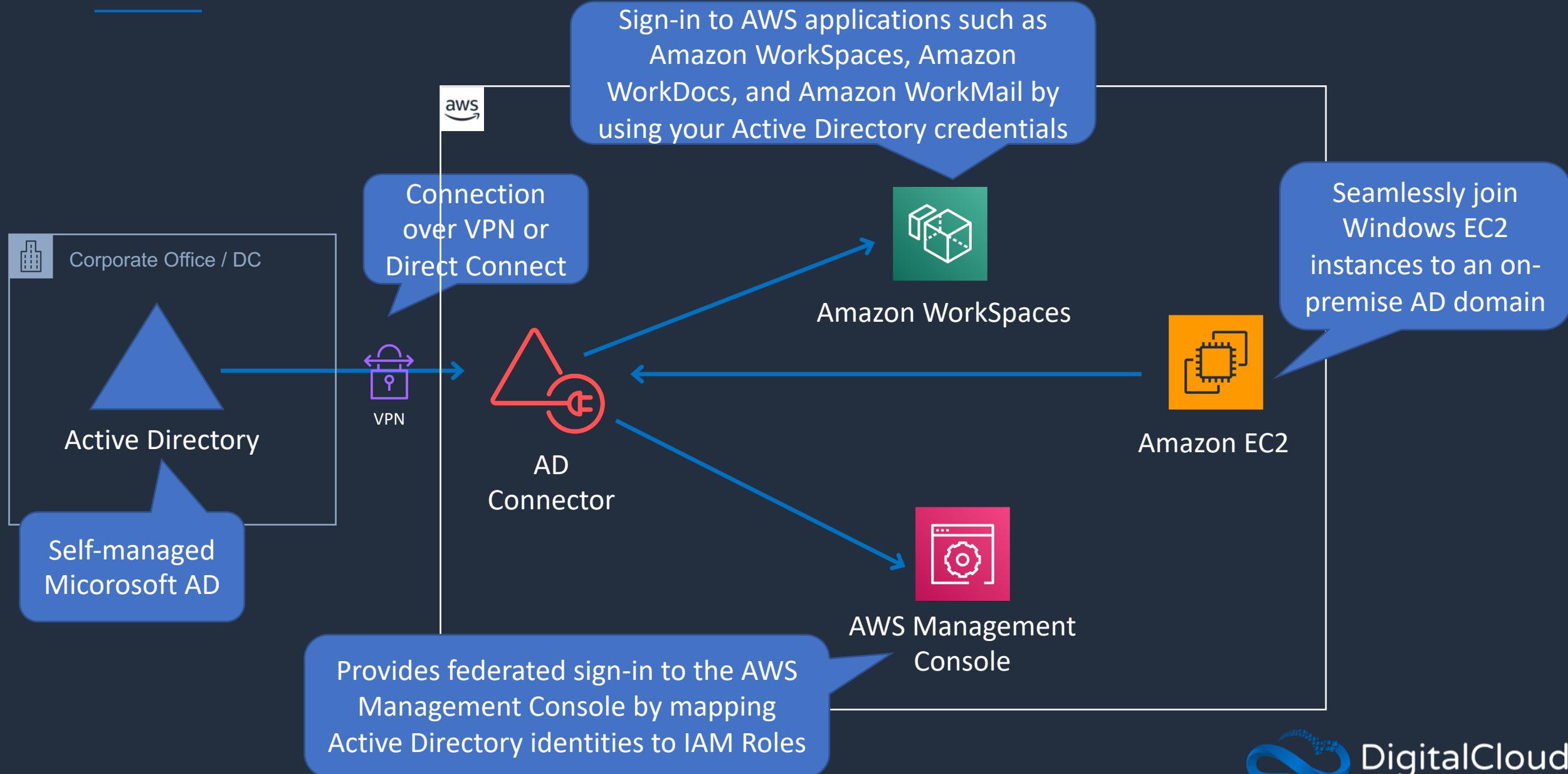
# AWS Managed Microsoft AD



Managed implementation of Microsoft Active Directory running on Windows Server 2012 R2



# AD Connector

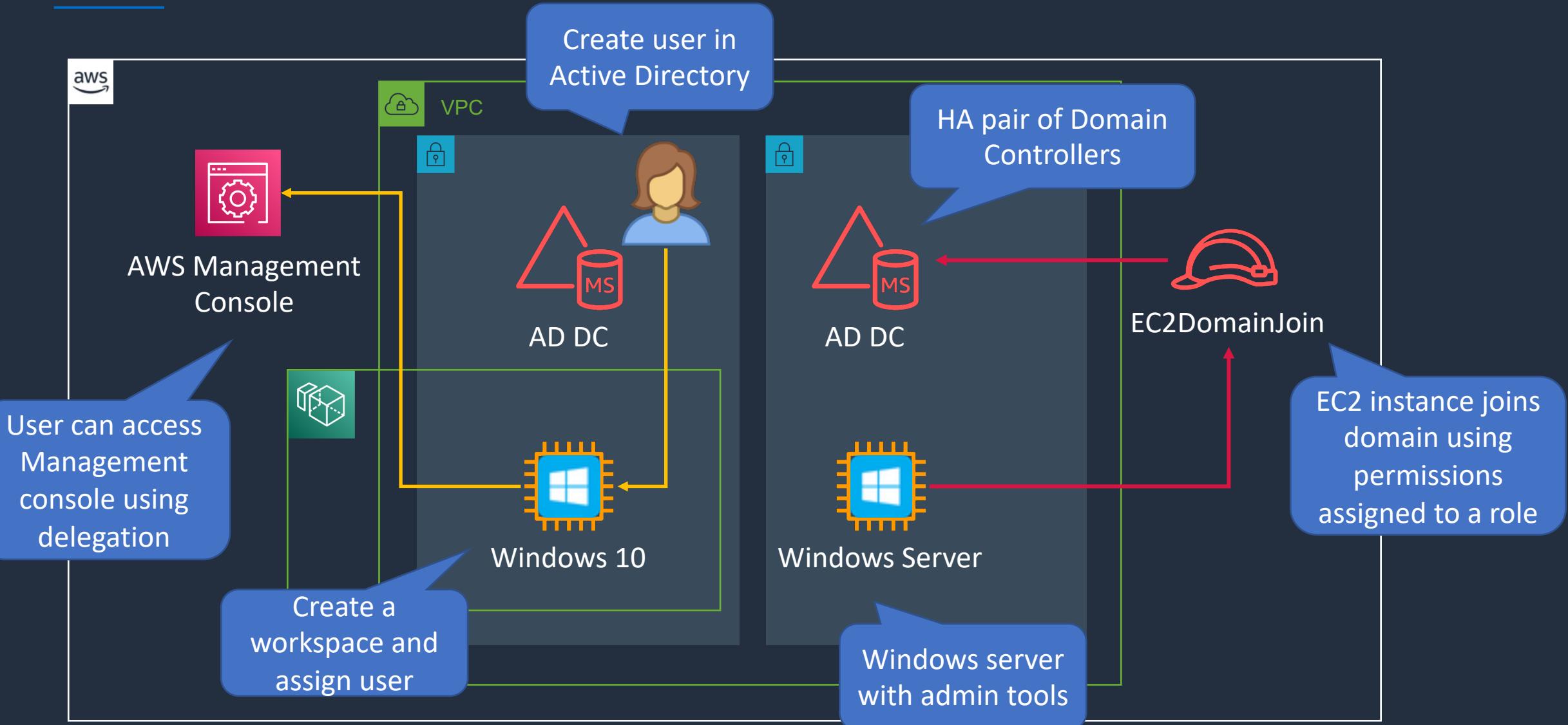


# Create AWS Managed Microsoft AD





# AWS Managed Microsoft AD



# Identity Federation





# Identity Federation Services



## AWS Identity & Access Management

- Can use separate SAML 2.0 or OIDC IdPs for each account
- Enables access control using federated user attributes
- User attributes can be cost center, job role etc.



## AWS Single Sign-On

- Central management for federated access
- Attach multiple AWS accounts and business applications
- Identities can be in AWS SSO
- Works with many IdPs (e.g. Active Directory)
- Permissions assigned based on group membership in IdP



## Amazon Cognito

- Federation support for web and mobile applications
- Provides sign-in and sign-up
- Supports sign-in with social IdPs such as Apple, Facebook, Google, and Amazon
- Supports IdPs using SAML 2.0

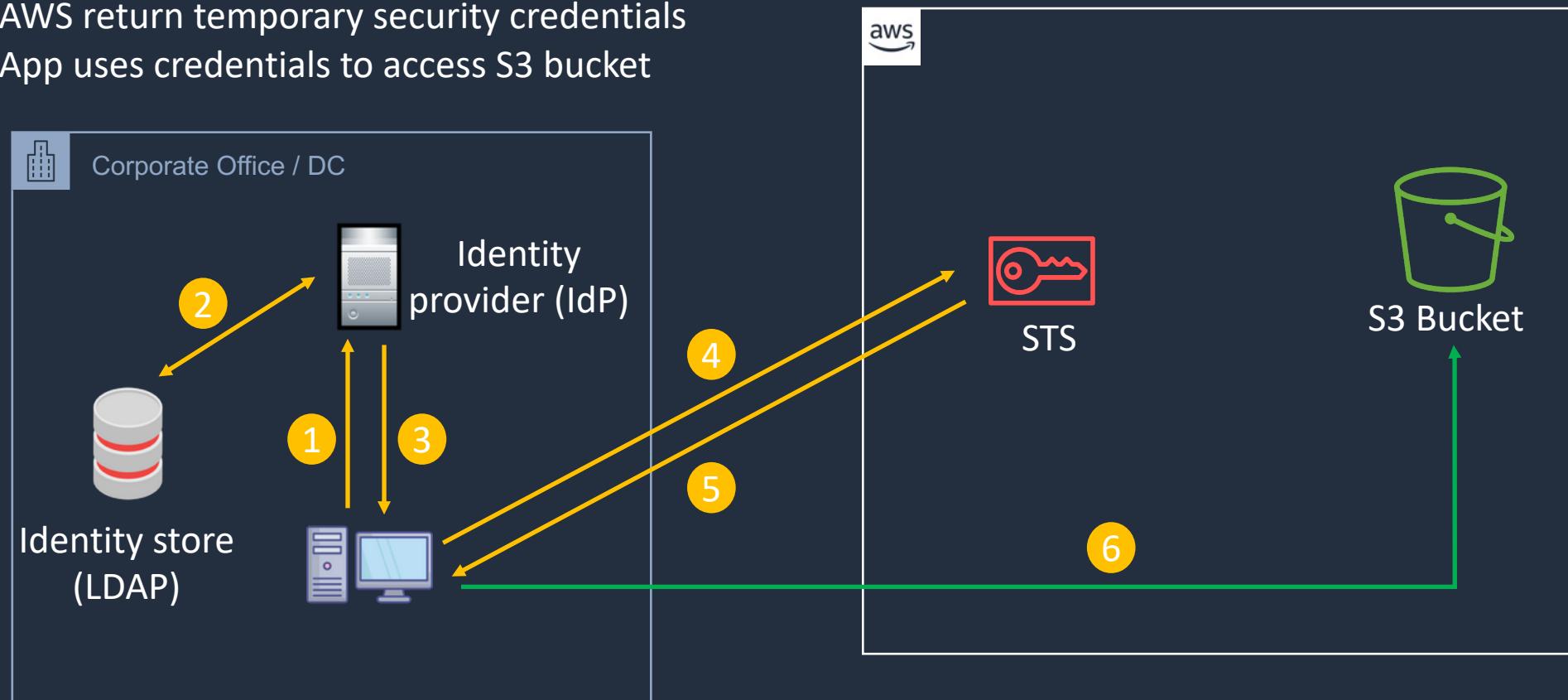
# IAM Identity Federation





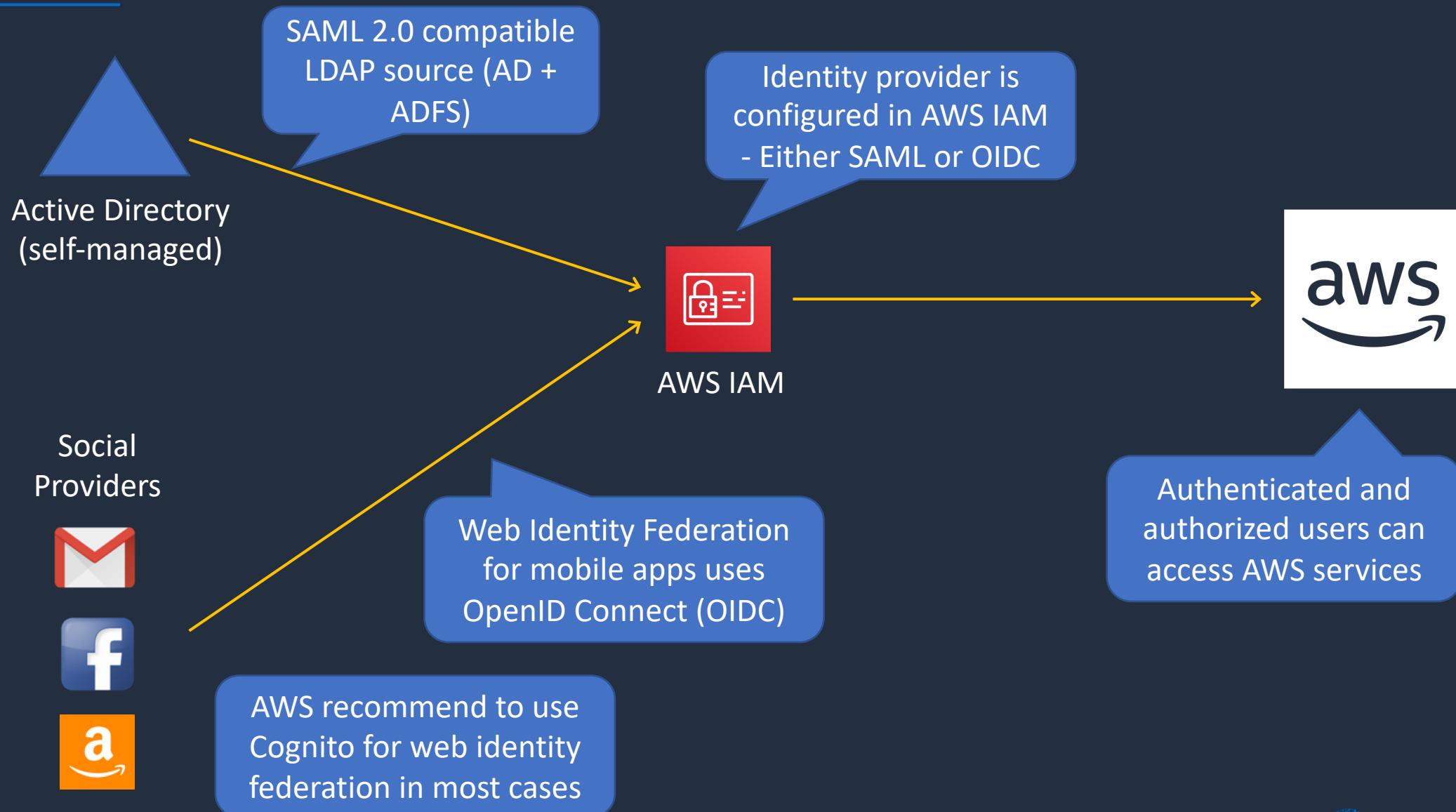
# Identity Federation

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





# Identity Provider Implementation

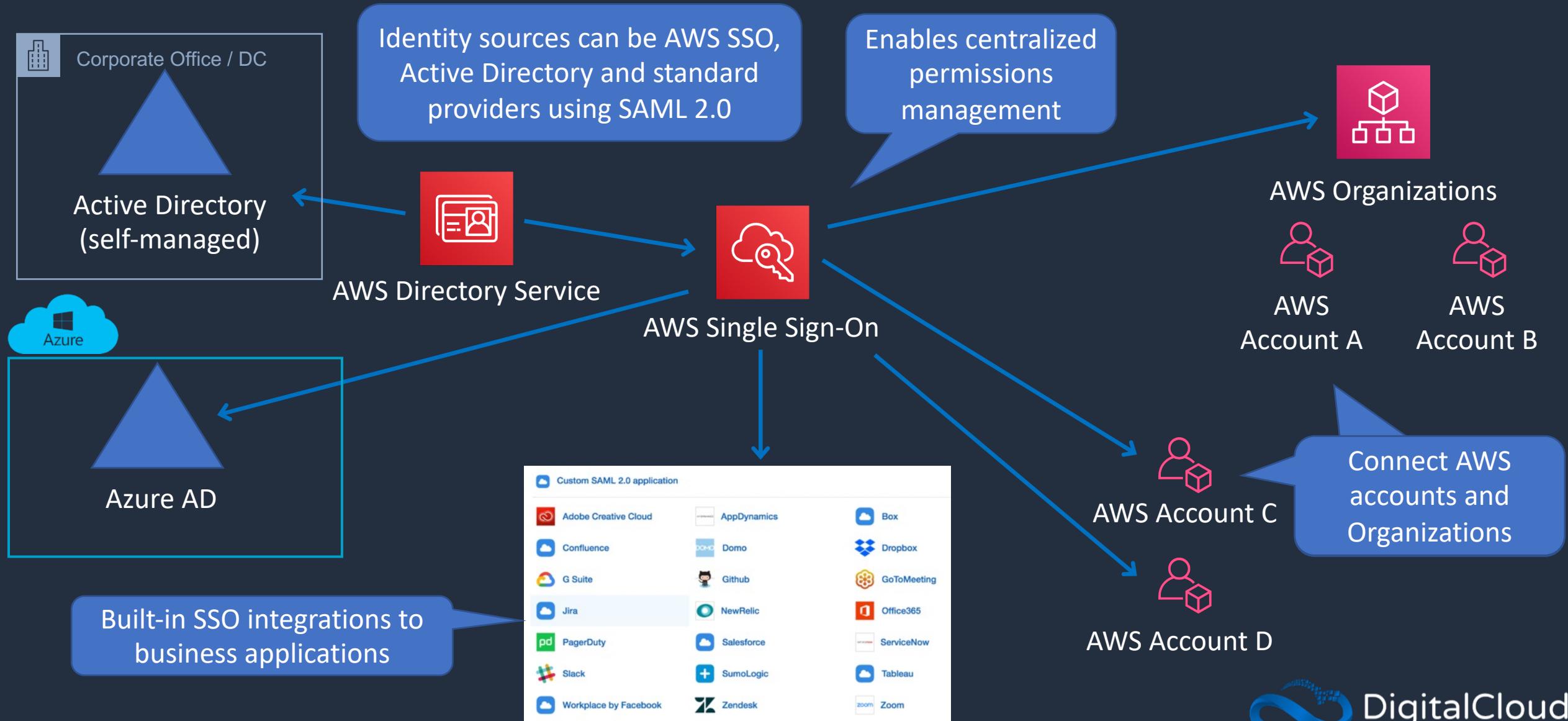


# AWS Single Sign-on (SSO)





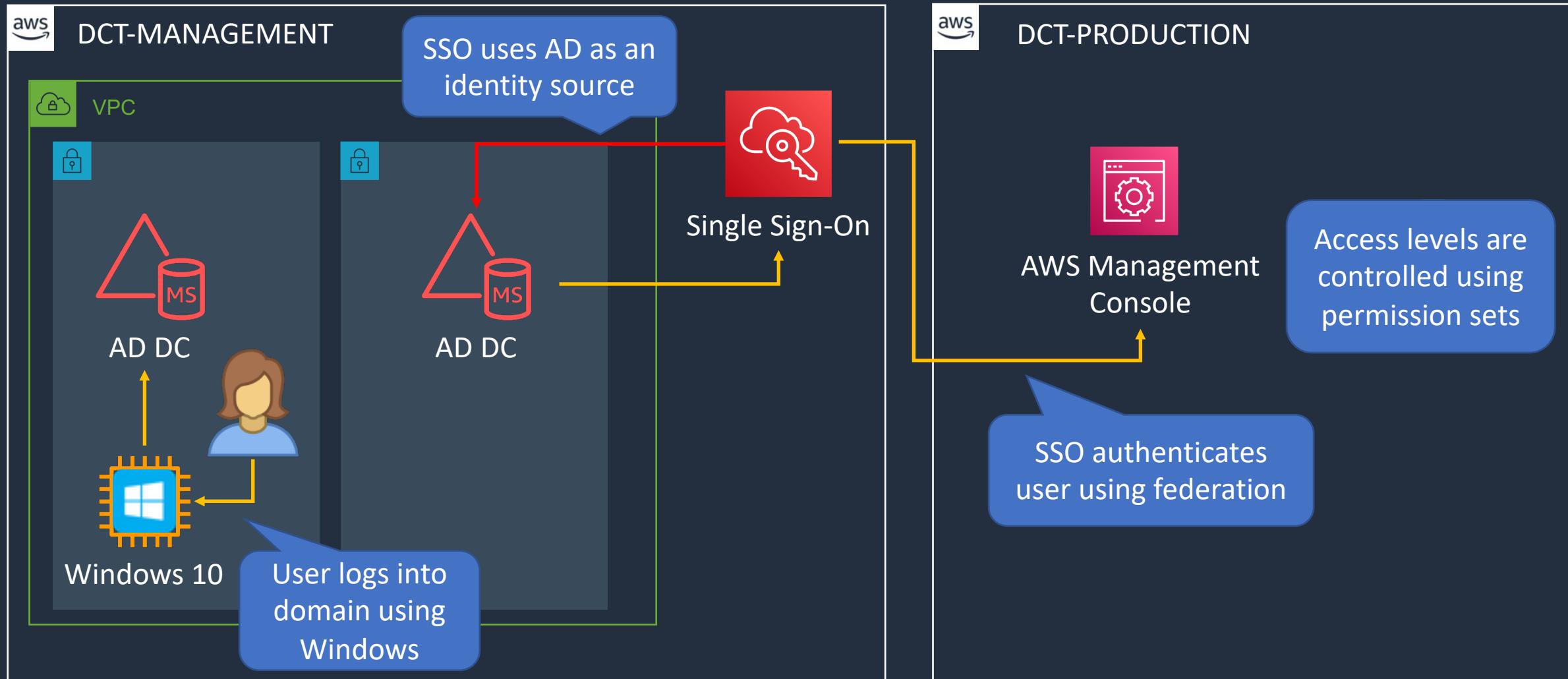
# AWS Single Sign-on (SSO)



# Configure AWS SSO with AWS Managed AD



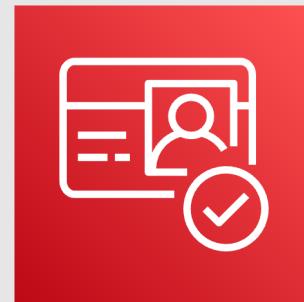
# AWS Managed Microsoft AD



# Cleanup the Hands-On Lab



# Amazon Cognito

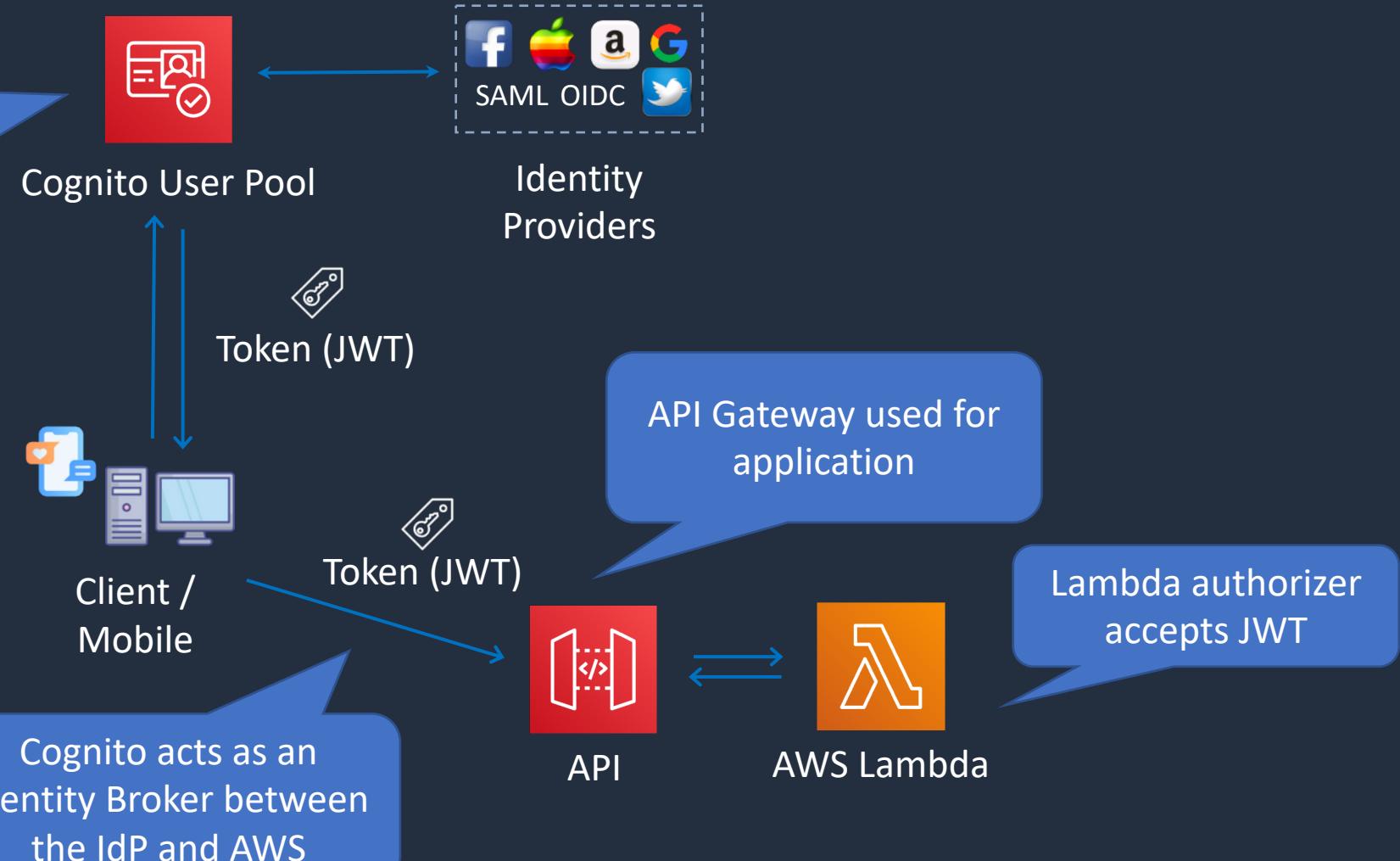




# Cognito User Pools

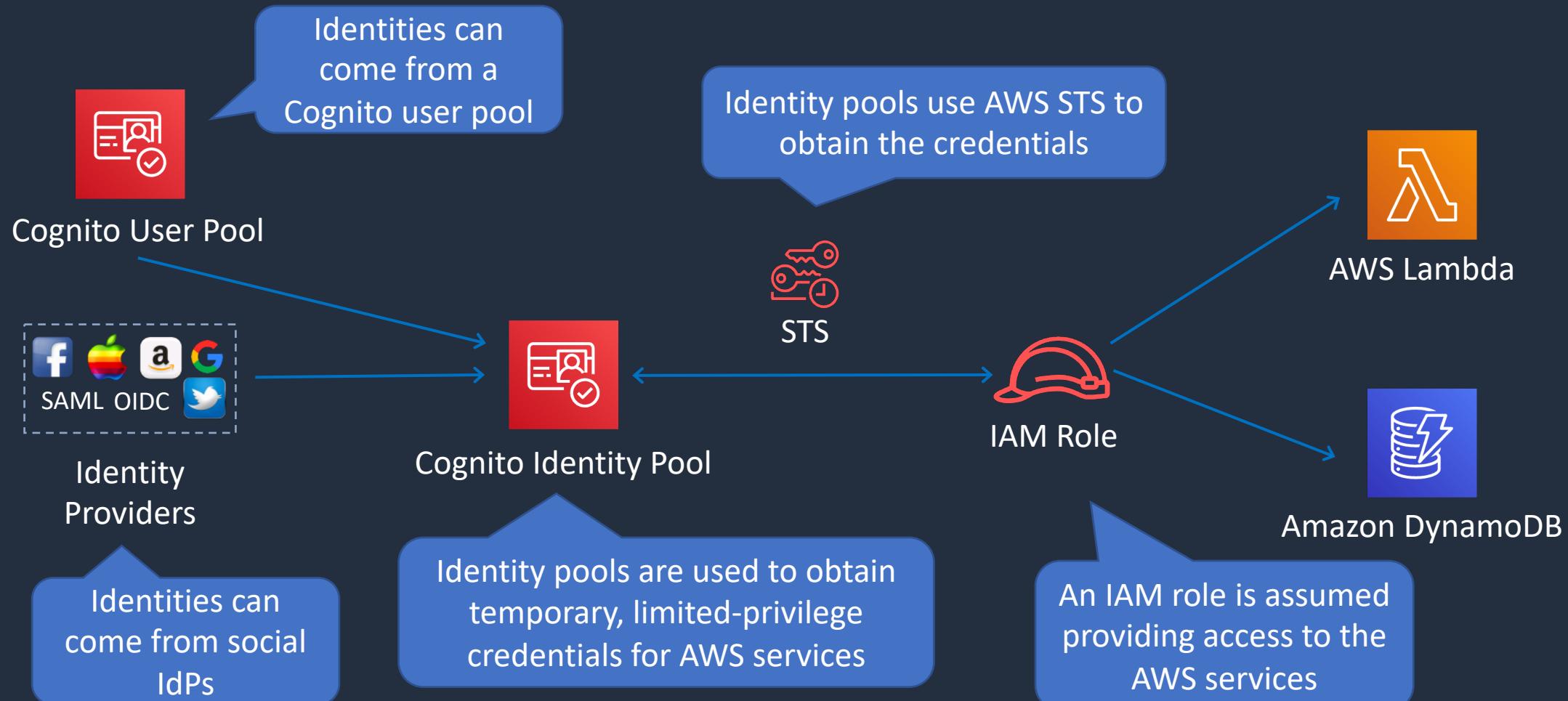
A User Pool is a directory for managing sign-in and sign-up for mobile applications

Users can also sign in using social IdPs



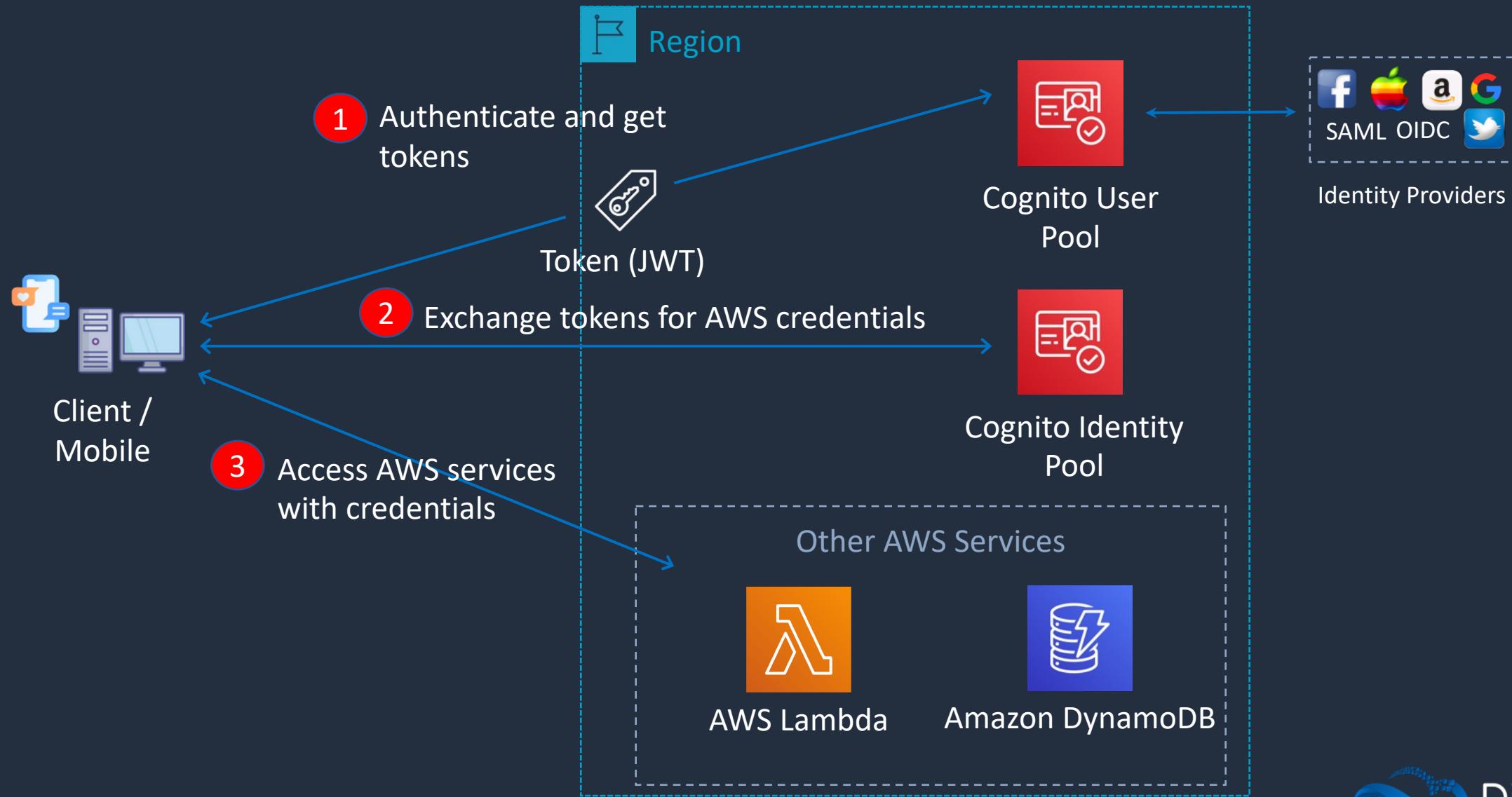


# Cognito Identity Pool





# User Pools + Identity Pools



# SECTION 5

## Advanced VPC

# Defining VPC CIDR Blocks





# Defining VPC CIDR Blocks

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 = 256 addresses

/16 Subnet Mask      

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

16 host bits = 65536 addresses

First Address	Last Address
192.168.0.1	192.168.255.254

/20 Subnet Mask      

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

12 host bits = 4096 addresses

First Address	Last Address
192.168.0.1	192.168.15.254

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



# 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:

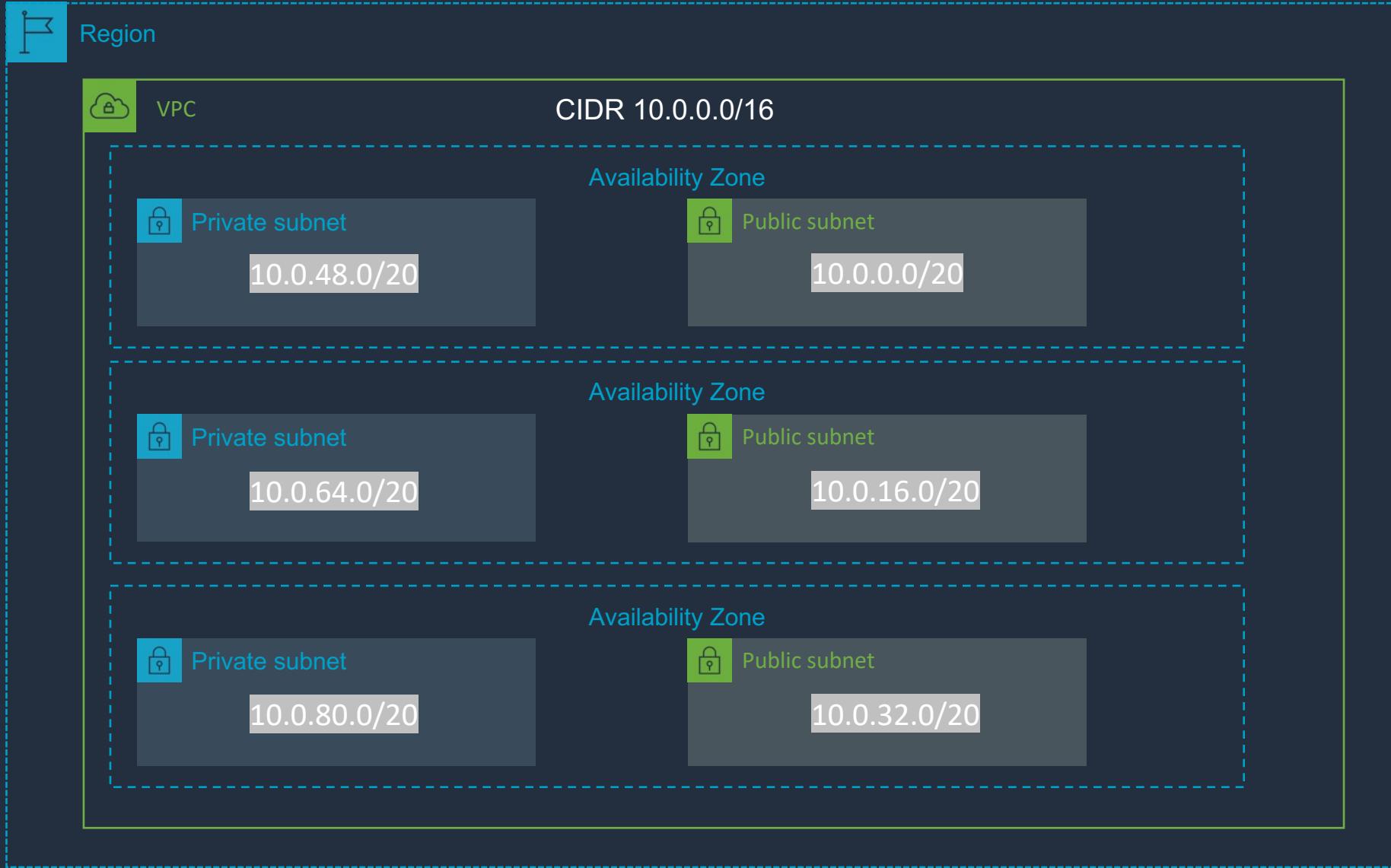
RFC 1918 Range	Example CIDR Block
10.0.0.0 - 10.255.255.255 (10/8 prefix)	Your VPC must be /16 or smaller, for example, 10.0.0.0/16
172.16.0.0 - 172.31.255.255 (172.16/12 prefix)	Your VPC must be /16 or smaller, for example, 172.31.0.0/16
192.168.0.0 - 192.168.255.255 (192.168/16 prefix)	Your VPC can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

# Create a Custom VPC with Subnets





# Create a Custom VPC



# VPC Routing Deep Dive

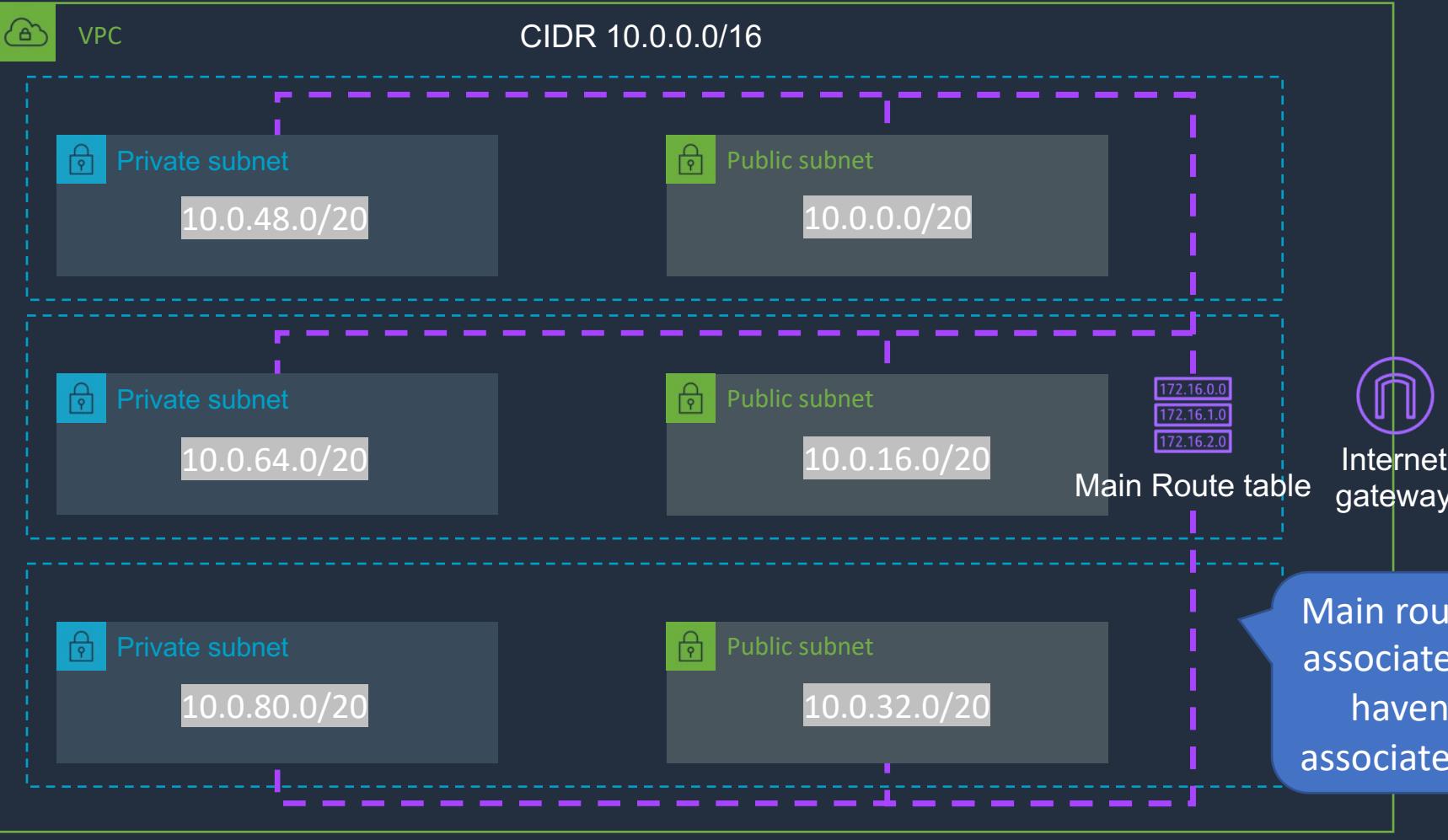
172.16.0.0  
172.16.1.0  
172.16.2.0



# VPC Routing Deep Dive



Region



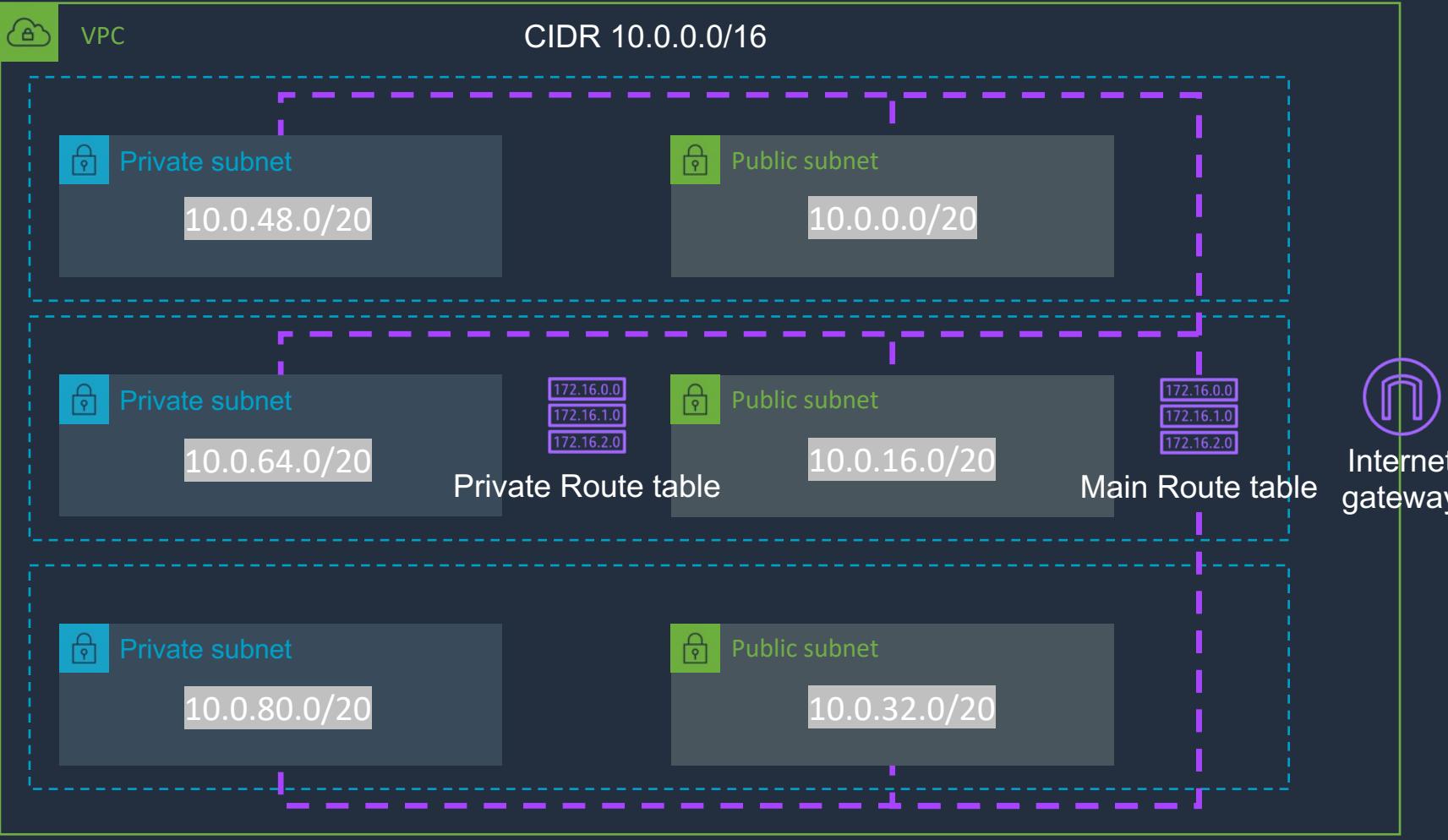
Main Route Table

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

# VPC Routing Deep Dive



Region



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

# VPC Routing Deep Dive



Region



VPC

CIDR 10.0.0.0/16

Each **subnet** can only  
be associated with  
**one** route table

Private subnet

10.0.48.0/20

Public subnet

10.0.0.0/20

Private subnet

10.0.64.0/20

Public subnet

10.0.16.0/20

Private Route table

172.16.0.0  
172.16.1.0  
172.16.2.0

Main Route table

172.16.0.0  
172.16.1.0  
172.16.2.0



Private subnet

10.0.80.0/20

Public subnet

10.0.32.0/20

Subnets are **explicitly**  
associated the private  
route table

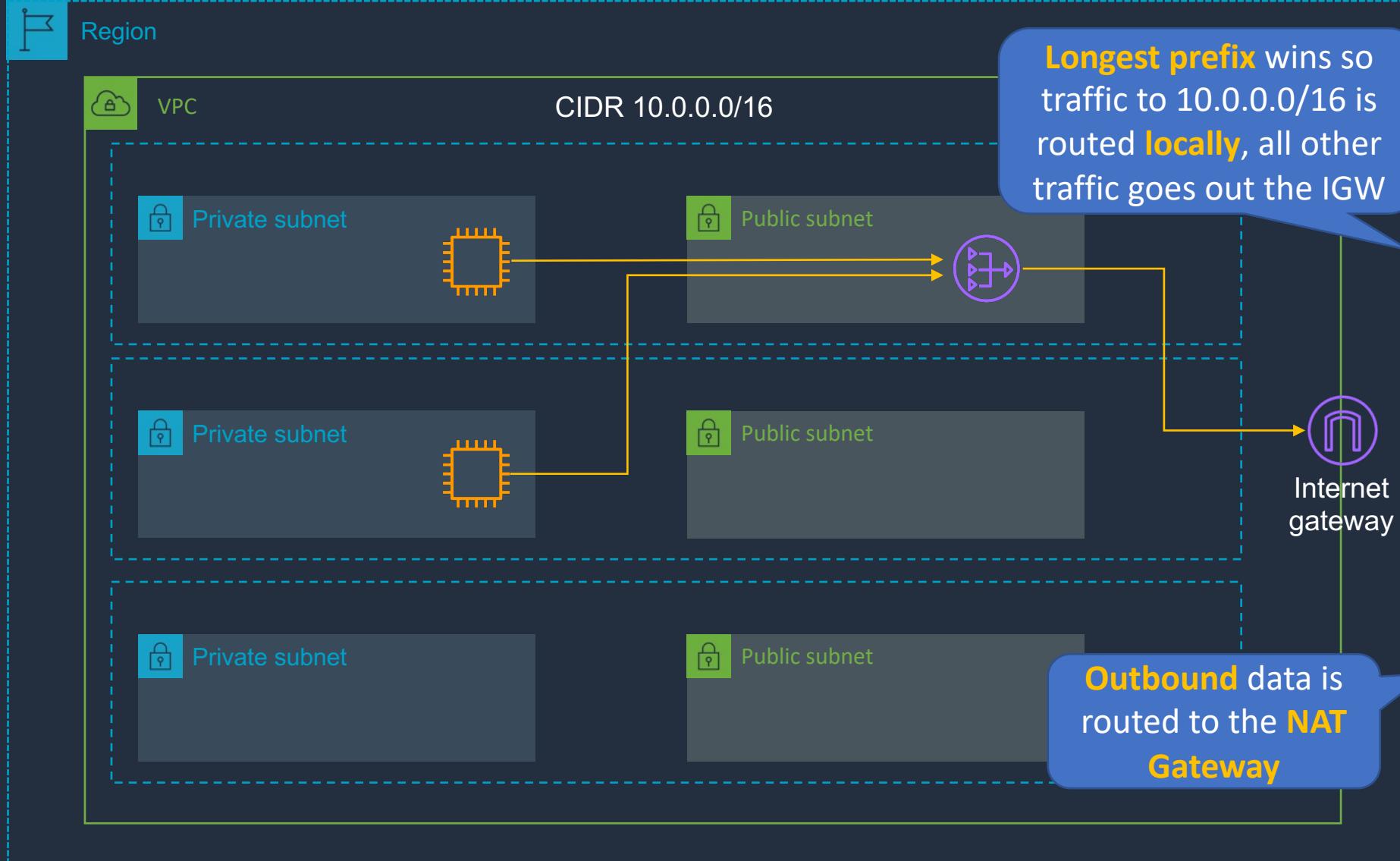
Main Route Table

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

Private Route Table

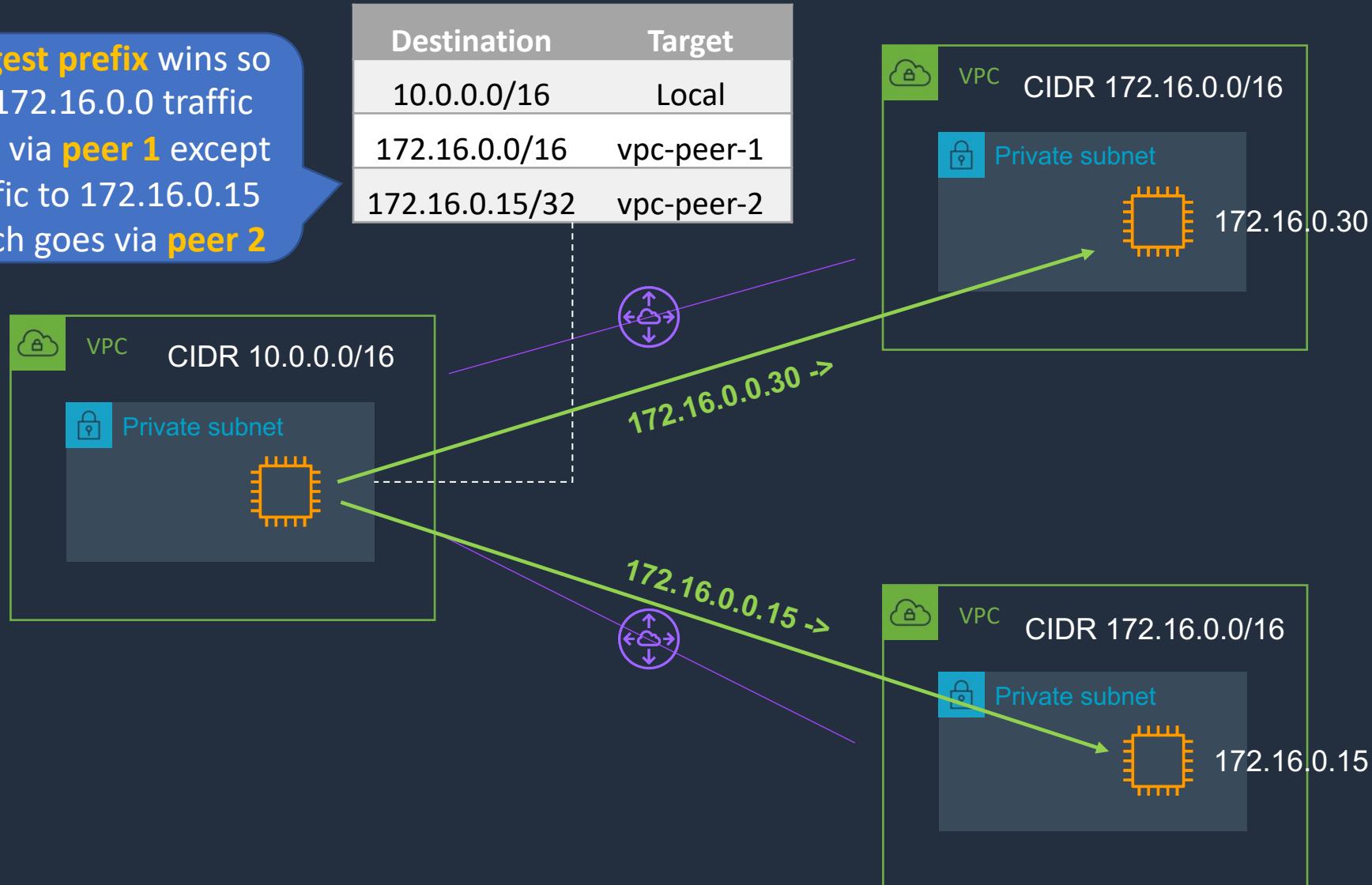
Destination	Target
10.0.0.0/16	Local

# VPC Routing Deep Dive



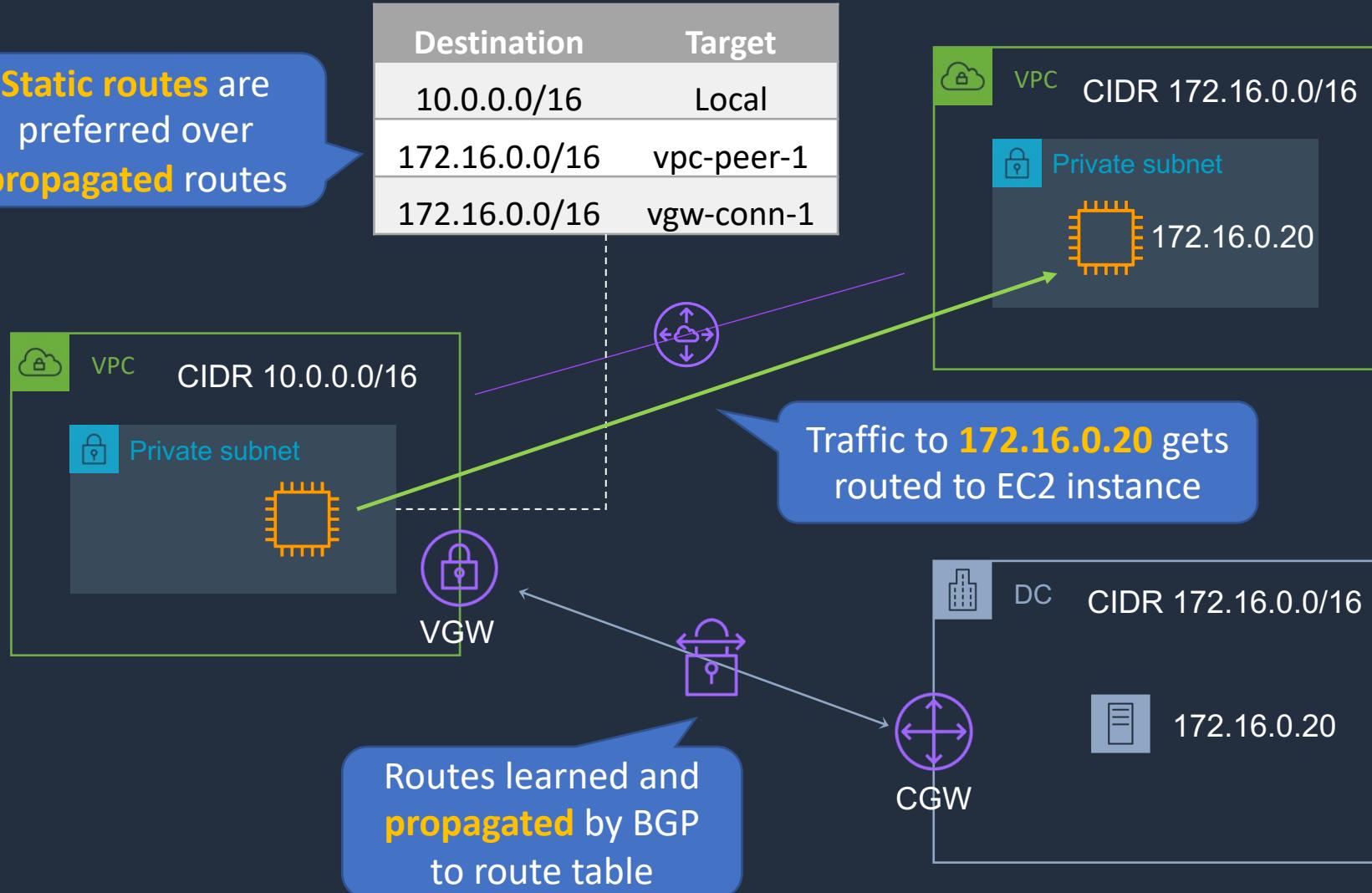
# VPC Routing Deep Dive

Longest prefix wins so all 172.16.0.0 traffic goes via **peer 1** except traffic to 172.16.0.15 which goes via **peer 2**



# VPC Routing Deep Dive

**Static routes** are preferred over **propagated** routes

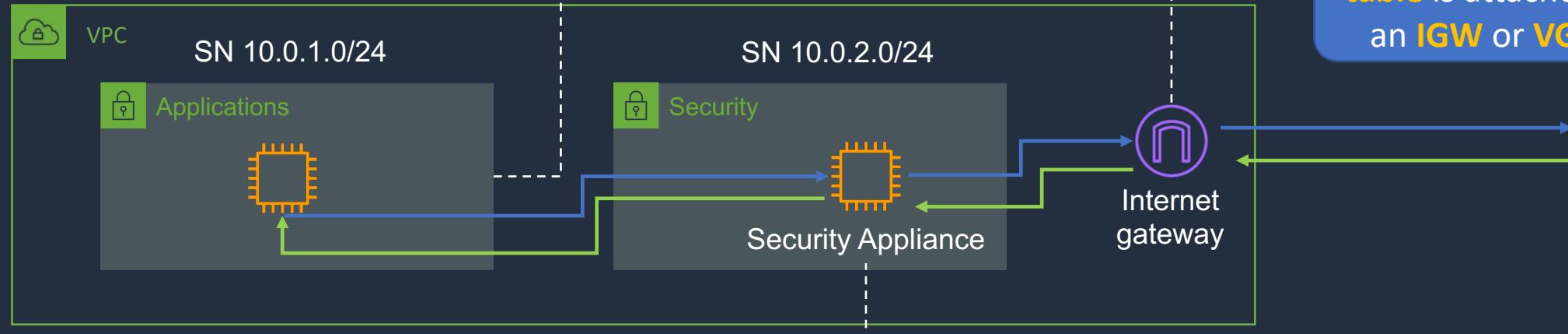


# Gateway Route Tables

0.0.0.0/0 points to the ENI ID of the **security appliance**

Destination	Target
10.0.0.0/16	Local
0.0.0.0/0	eni-id-sec

Destination	Target
10.0.0.0/16	Local
10.0.1.0/24	eni-id-sec



All **outbound** traffic forwarded to **IGW**

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

A **Gateway route table** is attached to an **IGW** or **VGW**

# IPv4 and IPv6 Routing

IPv6 traffic within the VPC is routed locally

IPv4 traffic within the VPC is routed locally

Traffic that doesn't match a more specific route goes via the IGW

IPv6 traffic that doesn't match a more specific route goes via the EIGW

IPv4 traffic for 172.31.0.0/16 network goes via a peering connection

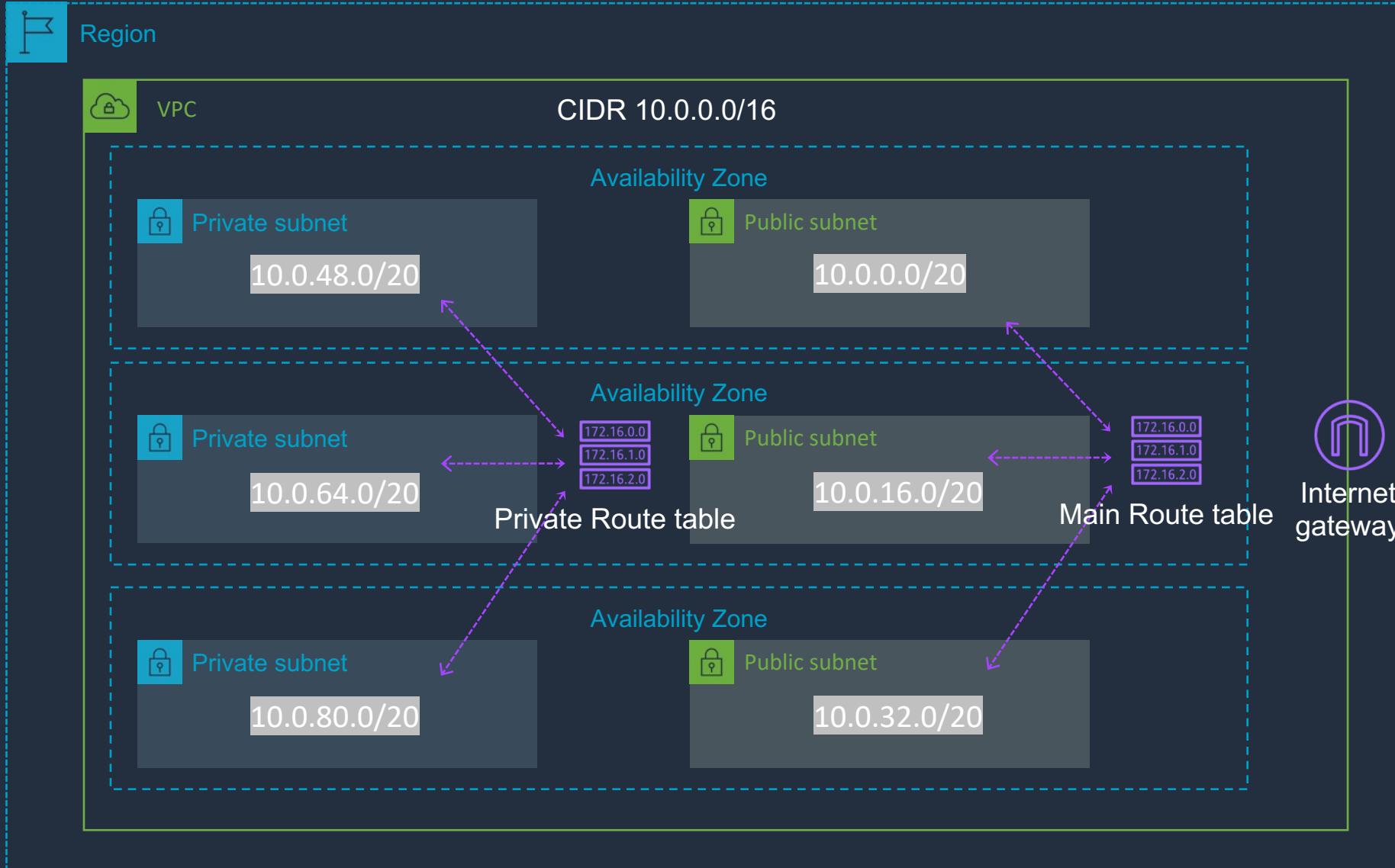
Destination	Target
10.0.0.0/16	Local
2001:db8:1234:1a00::/56	Local
172.31.0.0/16	pcx-11223344556677889
0.0.0.0/0	igw-12345678901234567
::/0	eigw-aabbccddeee1122334

# Configure Routing





# Create a Custom VPC

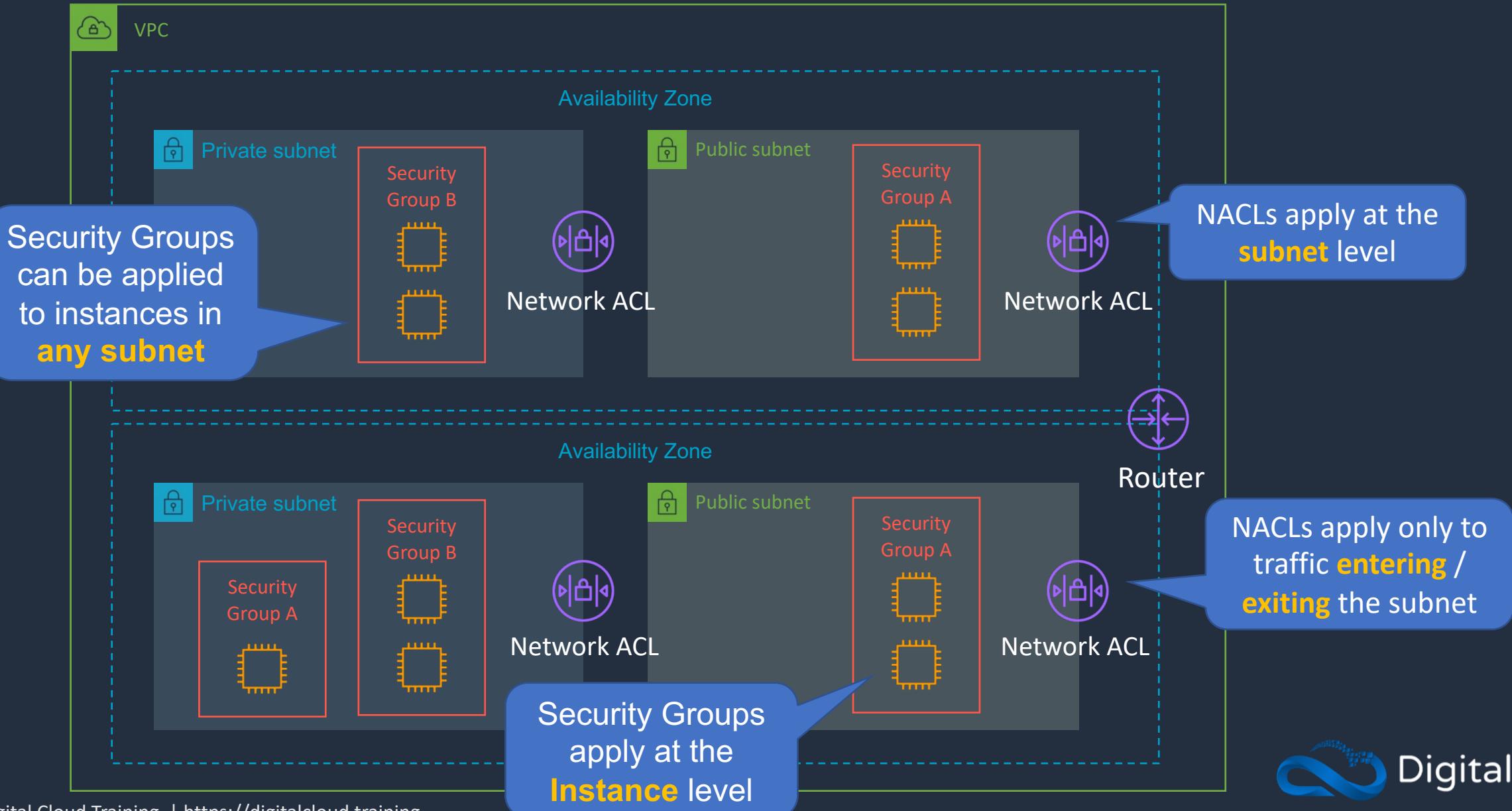


# Security Groups and Network ACLs

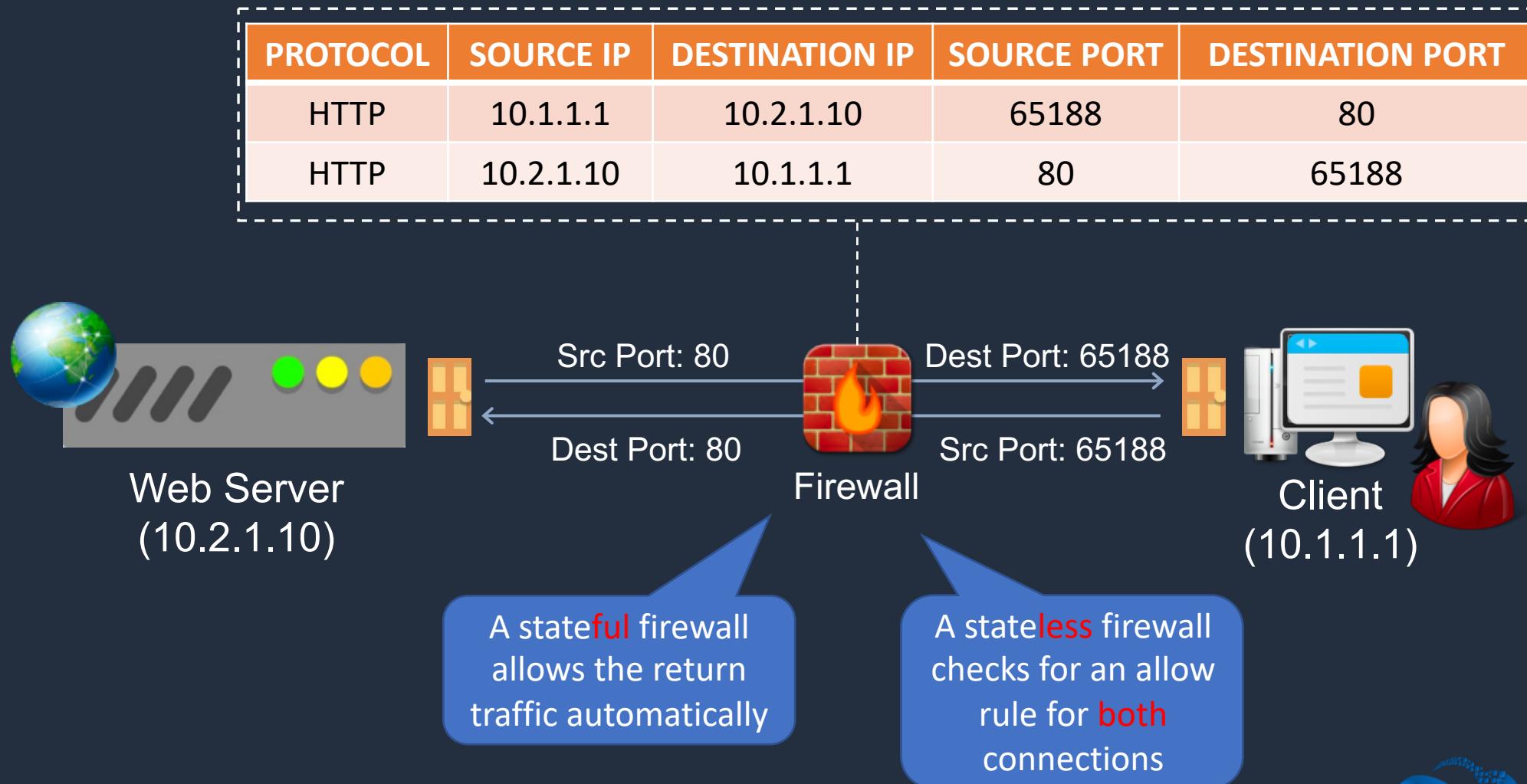




# Security Groups and Network ACLs



# 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

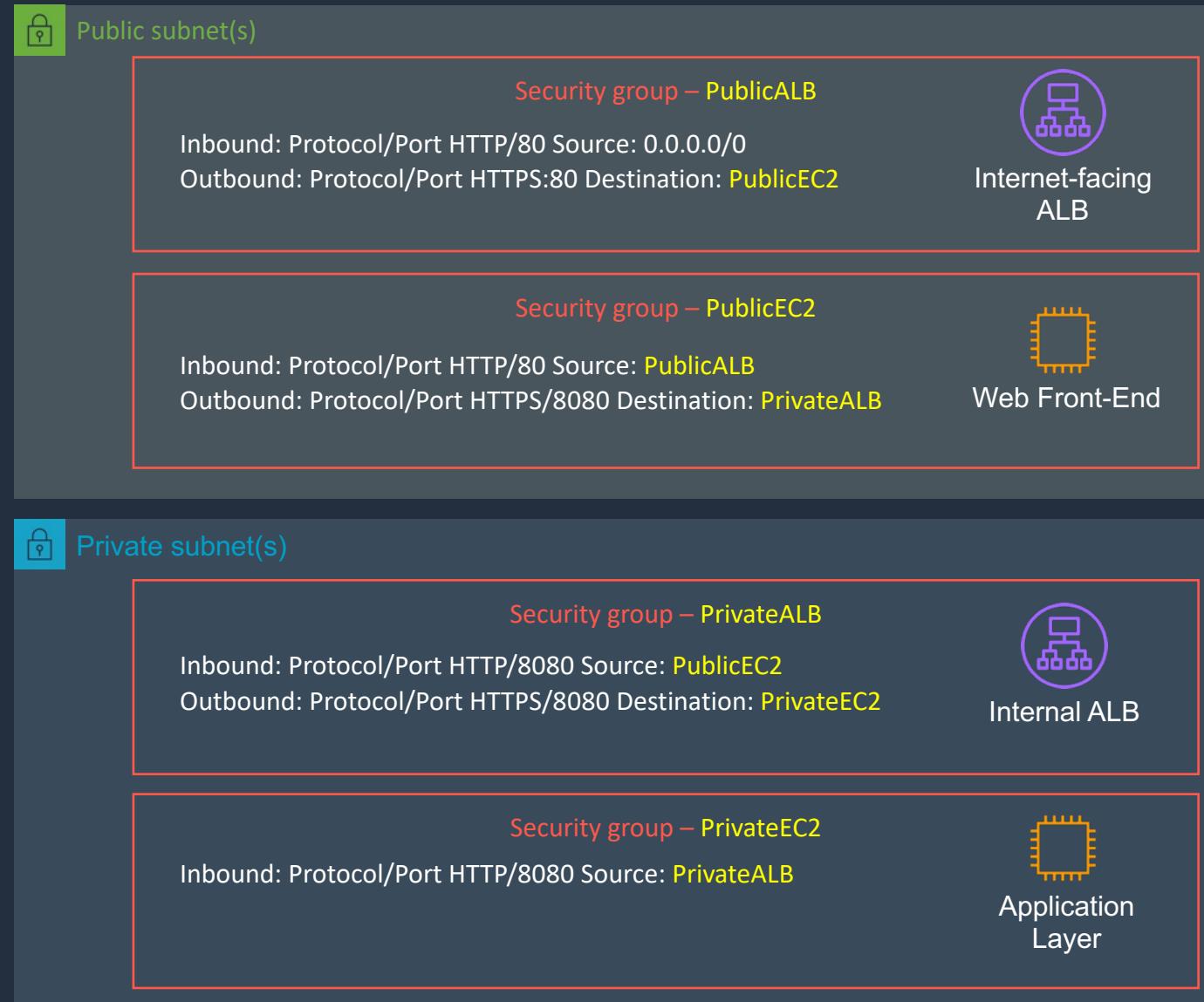
A source can be an **IP  
address or security  
group ID**



# 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

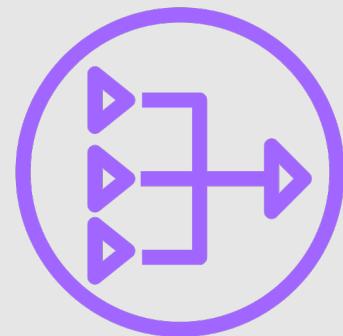
NACLs have an explicit deny

Rules are processed in order

# Setup Security Groups and NACLs

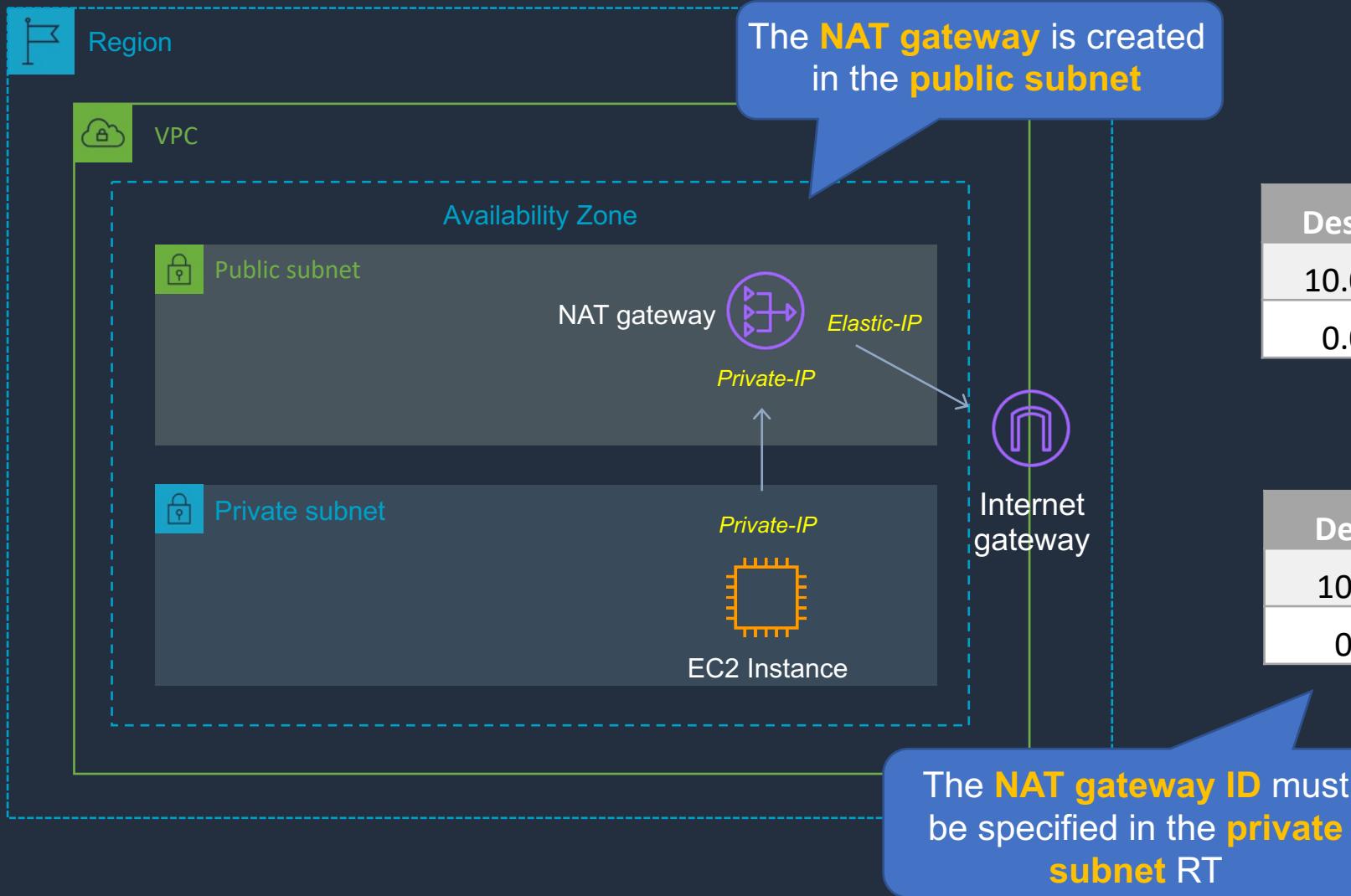


# NAT Gateways and NAT Instances



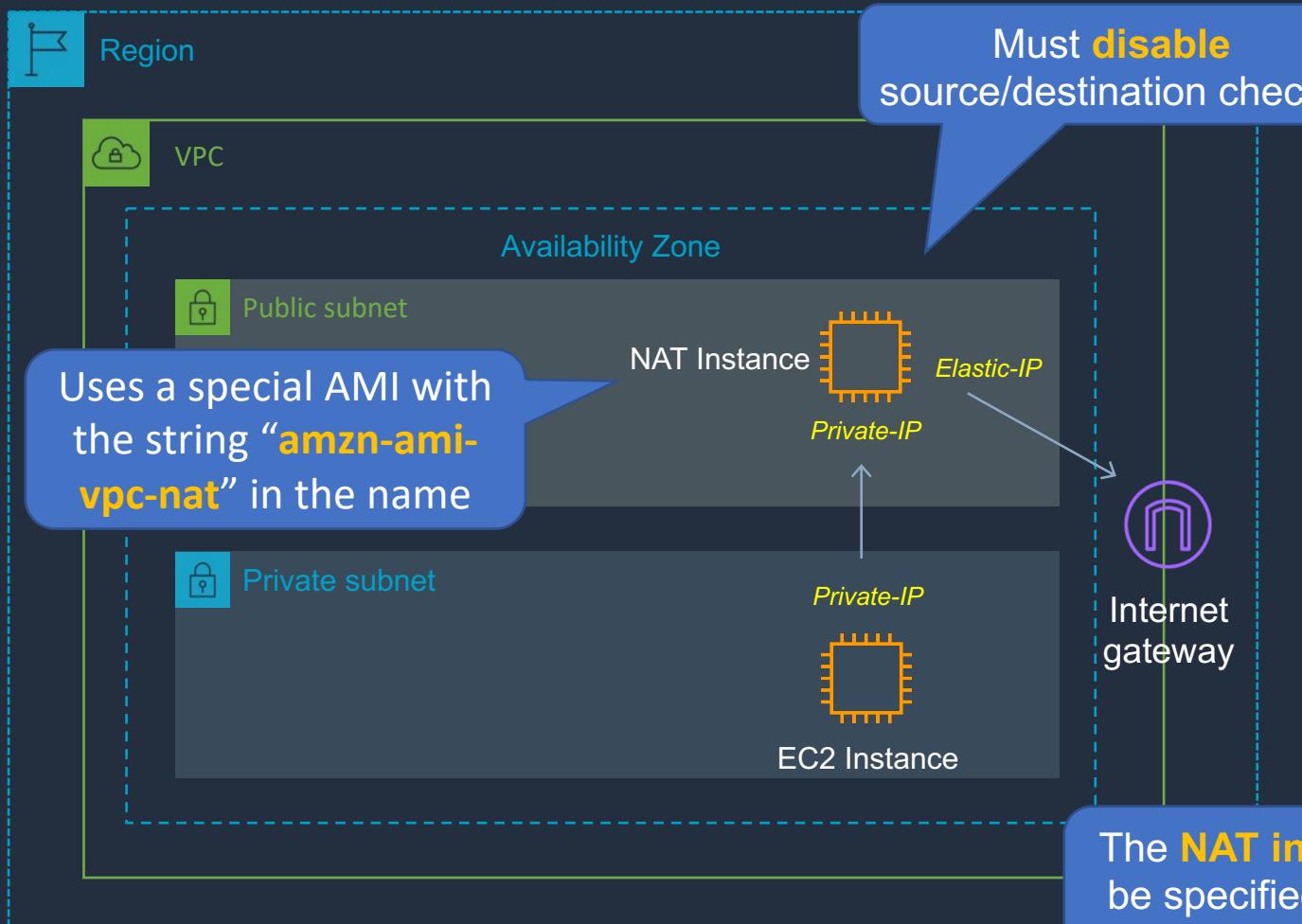


# NAT Gateways





# 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

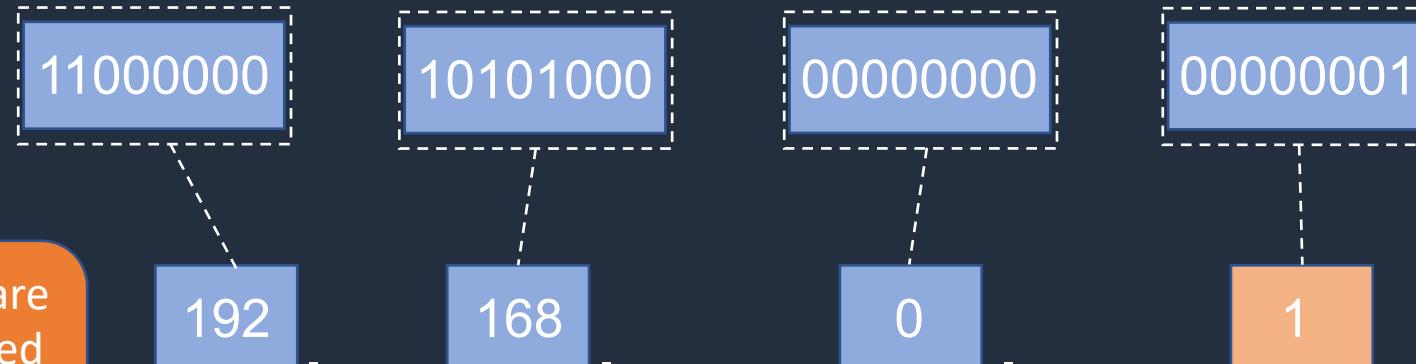
# Using IPv6 in a VPC





# 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

2020 : 0001 : 9d32 : 5bc2 : 1c48 : 32c1 : a93b : b12c

Network Part

Node Part

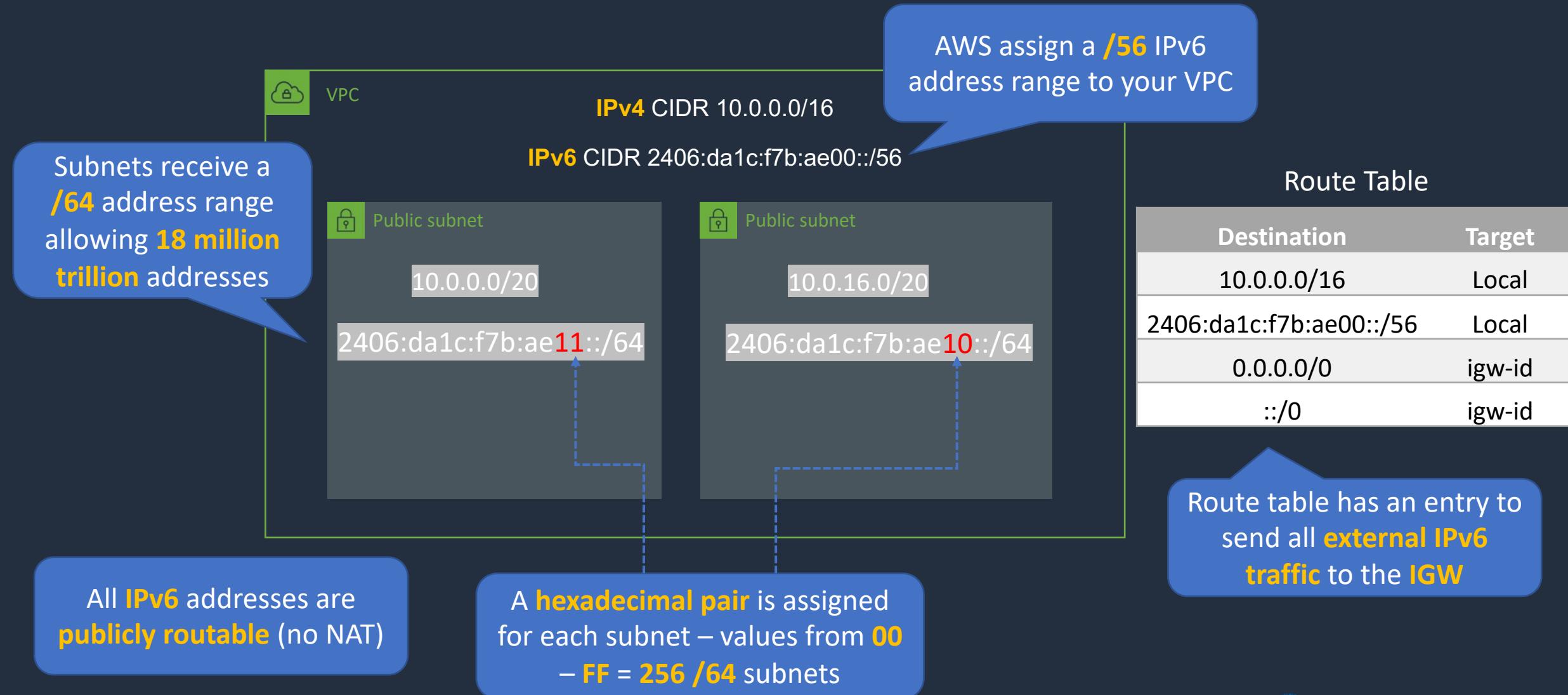
An **IPv6** addresses use **hexadecimal** whereas **IPv4** addresses use **dotted decimal**

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

# Configure IPv6



# Additional Settings

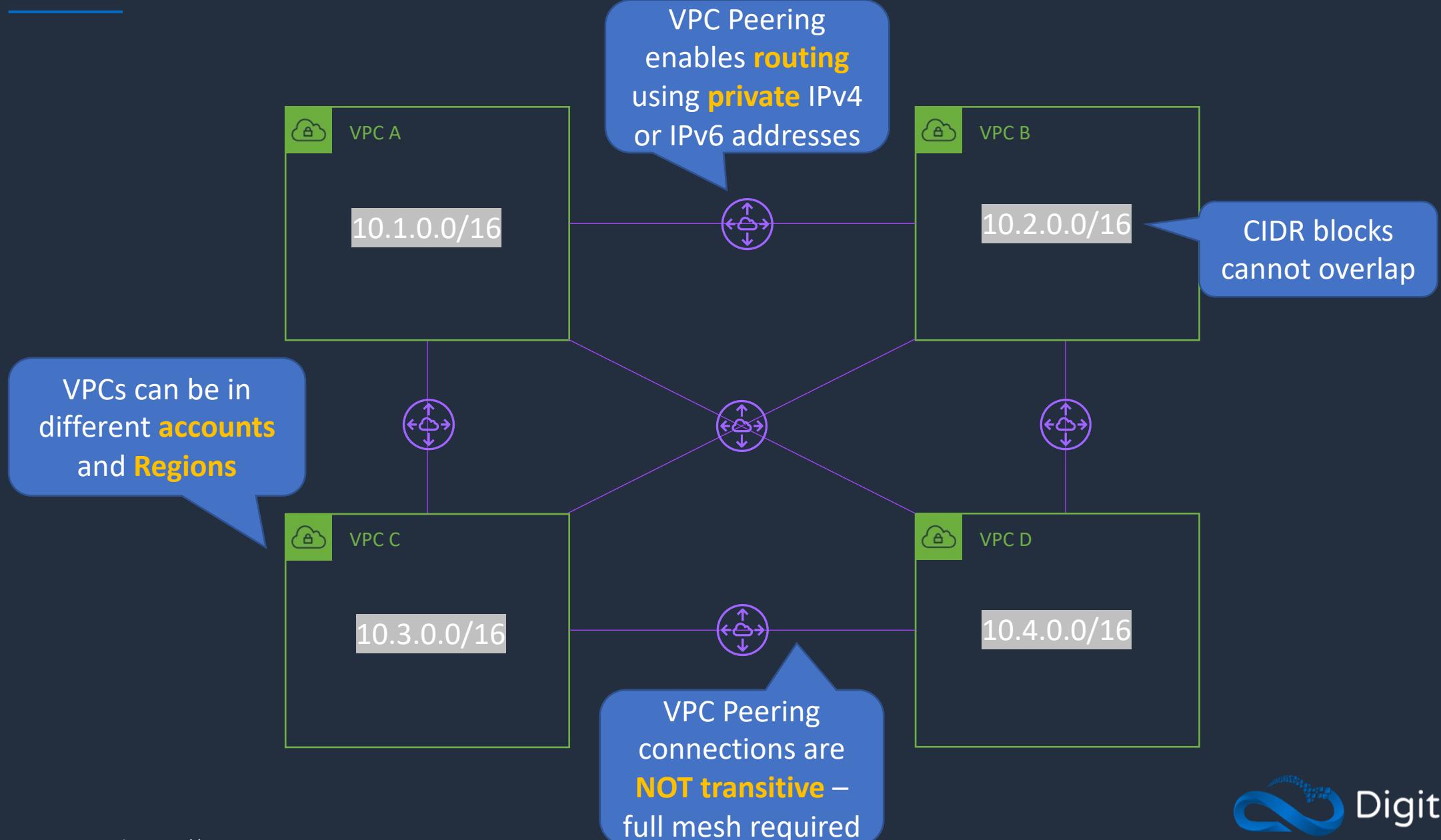


# VPC Peering





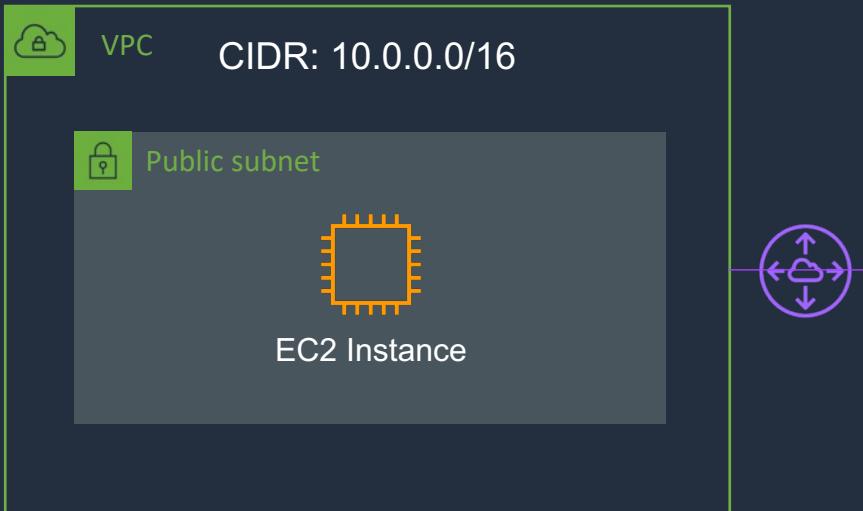
# VPC Peering



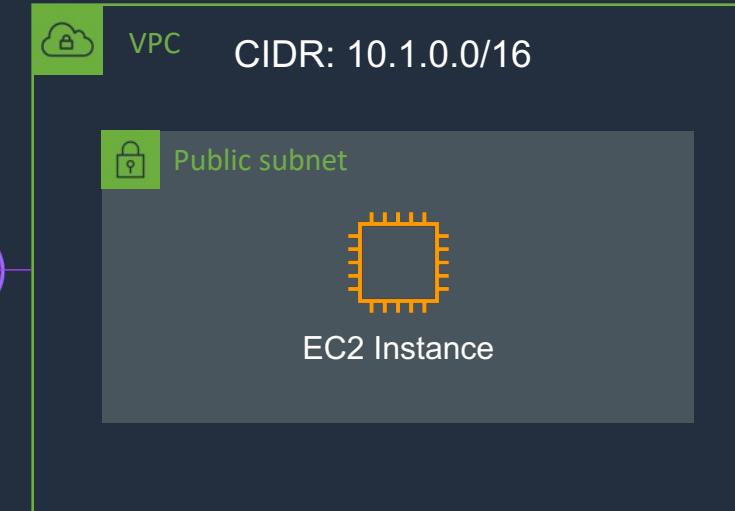


# VPC Peering

Region 1



Region 2



Security group (Region1-SG)

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

Security group (Region2-SG)

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

# Setup VPC in Second Account



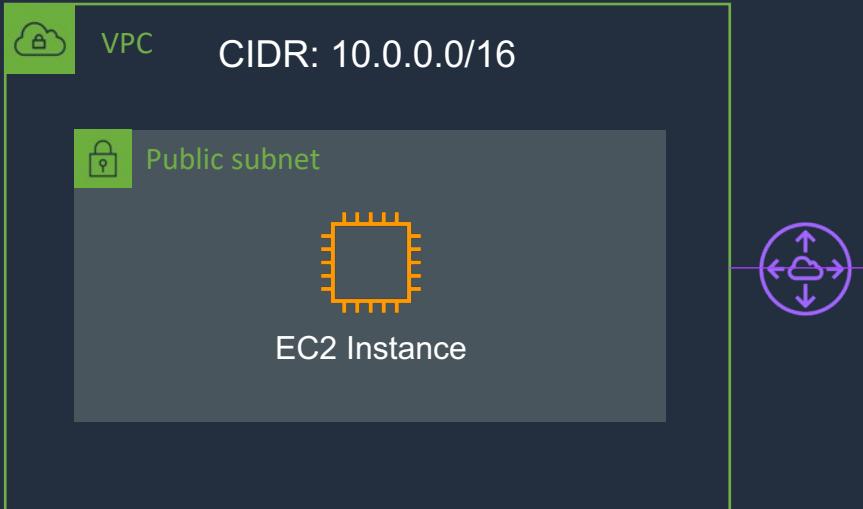
# Create VPC Peering Connection



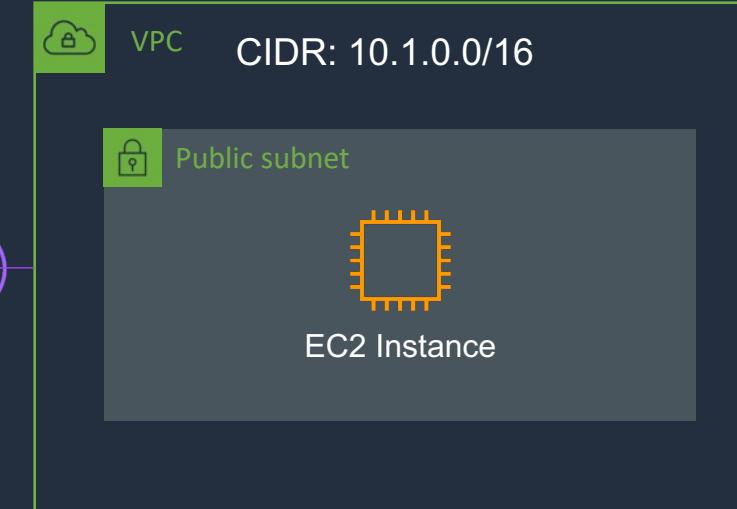


# Create VPC Peering Connection

Region 1 / Account 1



Region 2 / Account 2



Security group (Region1-SG)

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

Security group (Region2-SG)

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

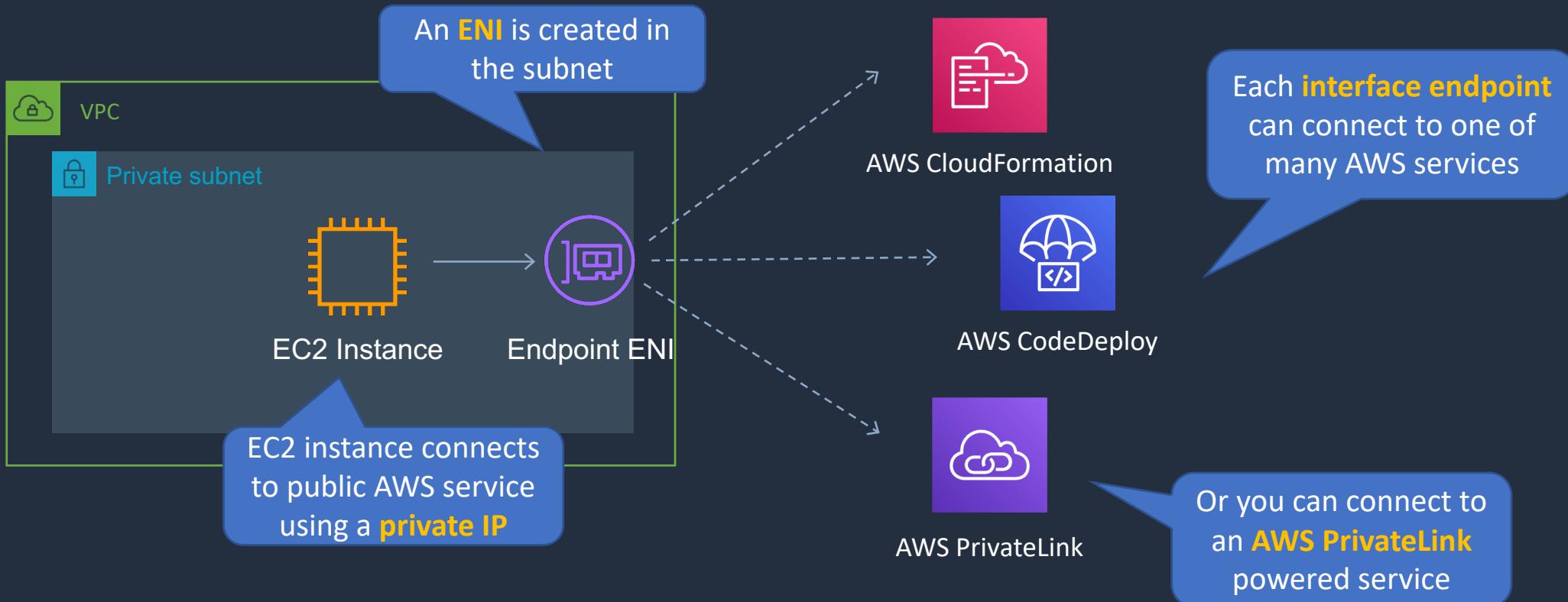
Destination	Target
10.0.0.0/16	peering-id

# VPC Endpoints



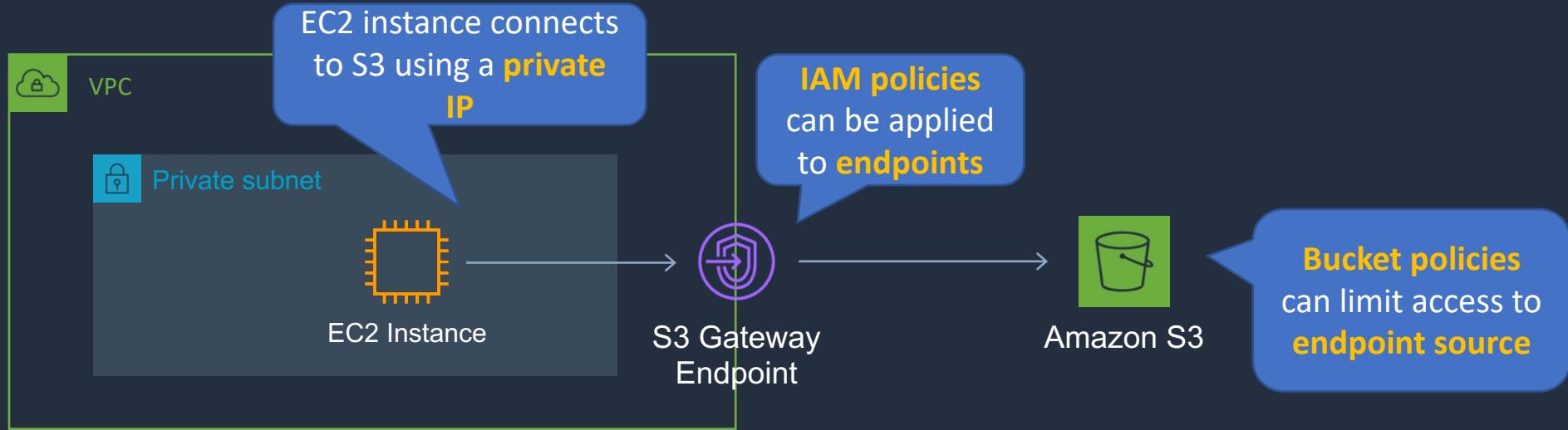


# VPC Interface Endpoints





# 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**



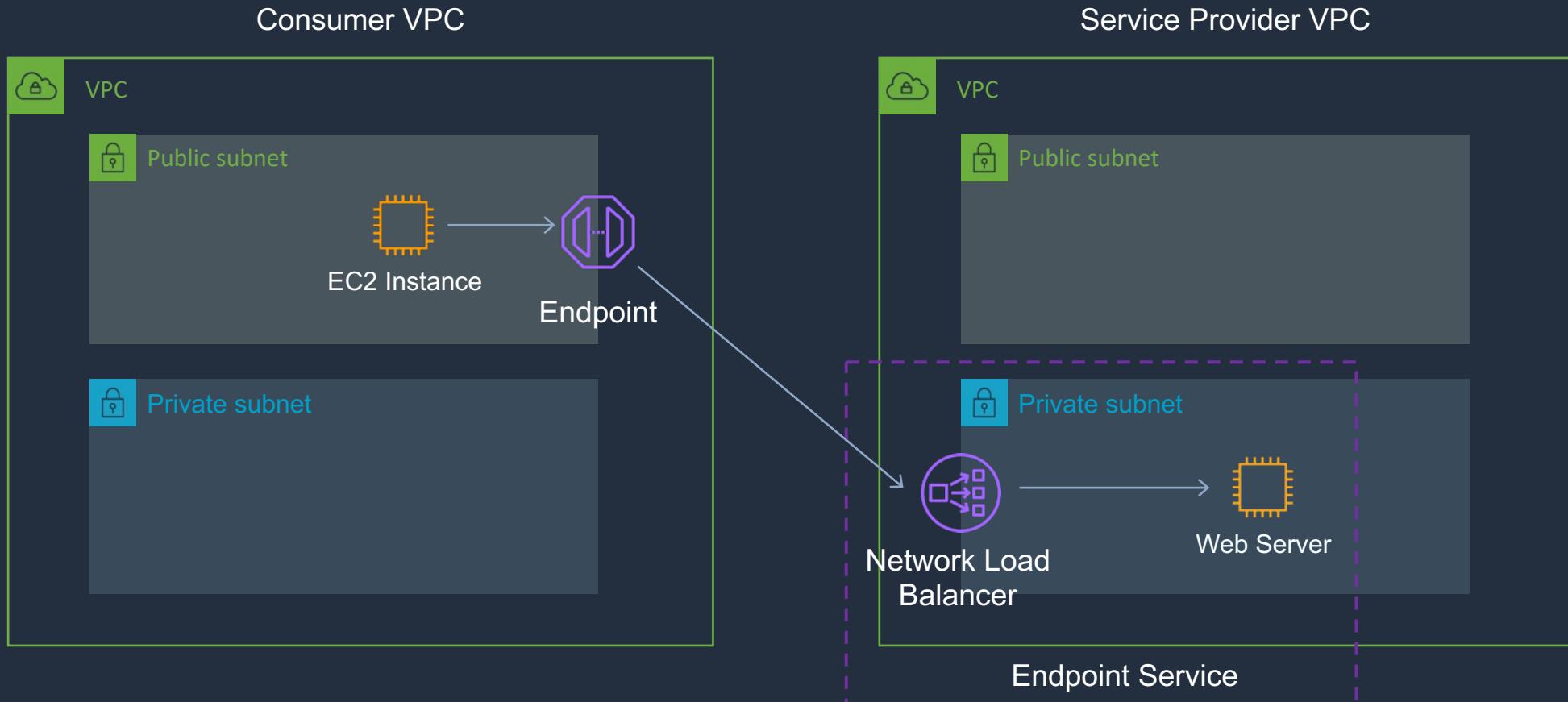
# VPC Endpoints

---

	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

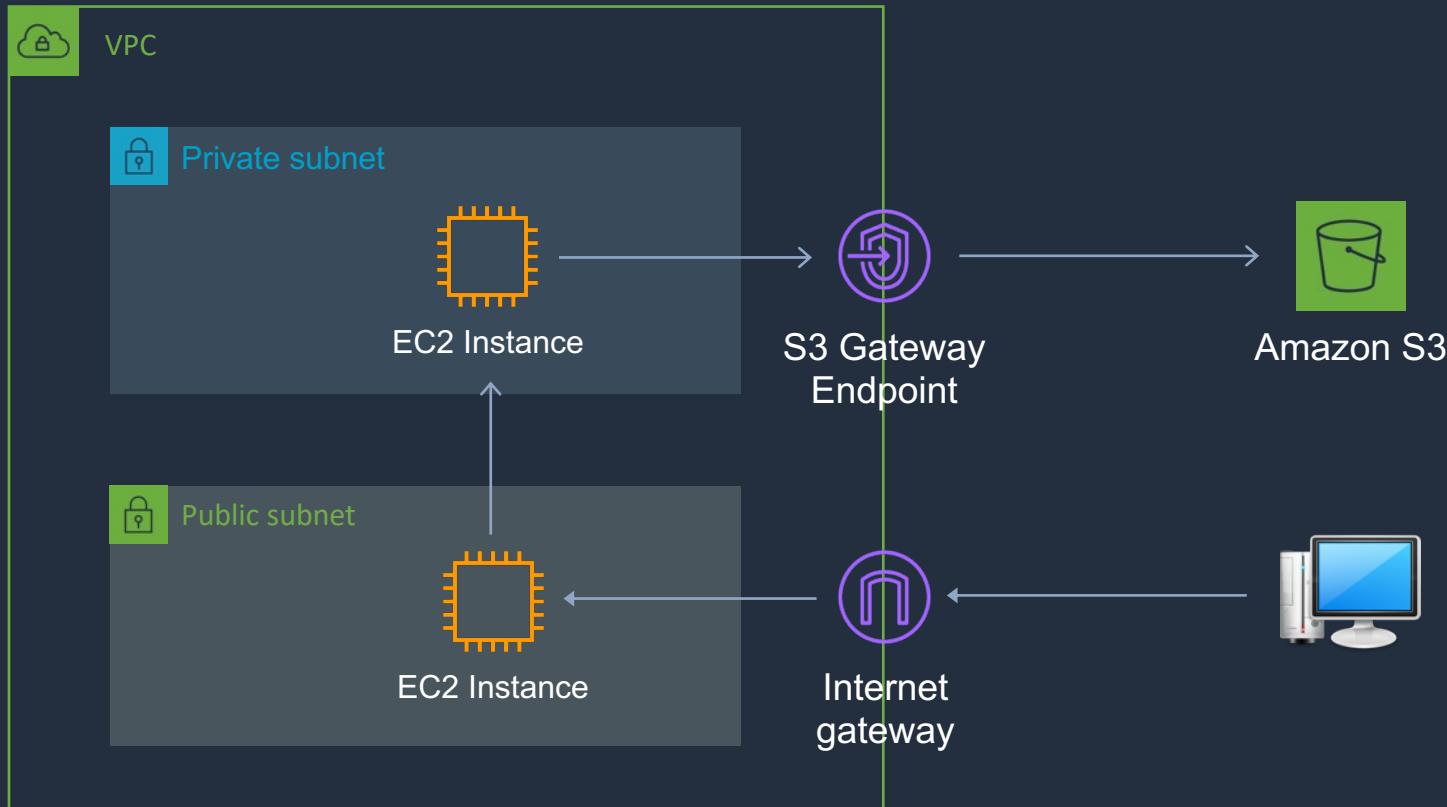


# Create VPC Endpoint





# VPC Gateway Endpoints



Private Subnet 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>

# SECTION 6

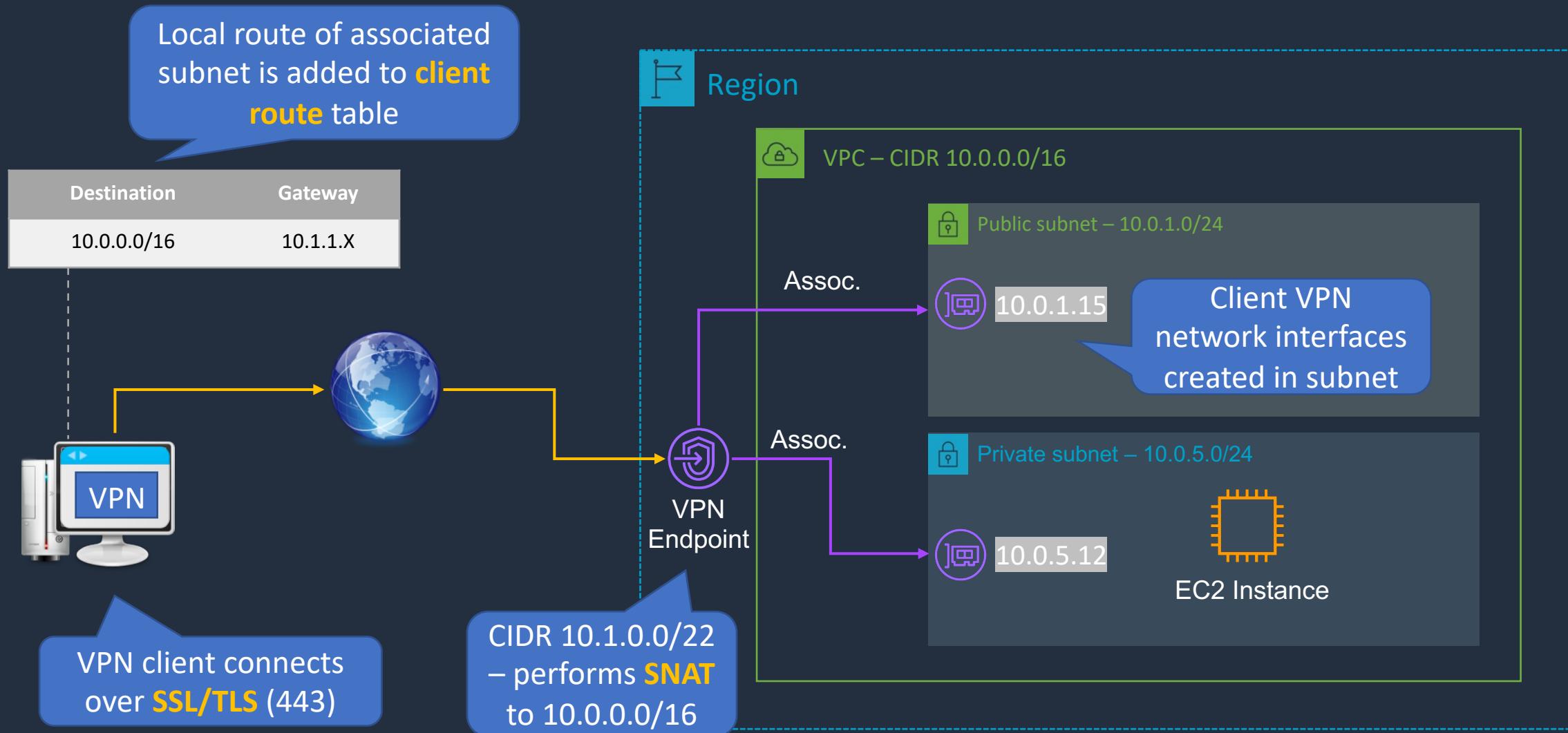
## Hybrid Connectivity

# AWS Client VPN





# AWS Client VPN

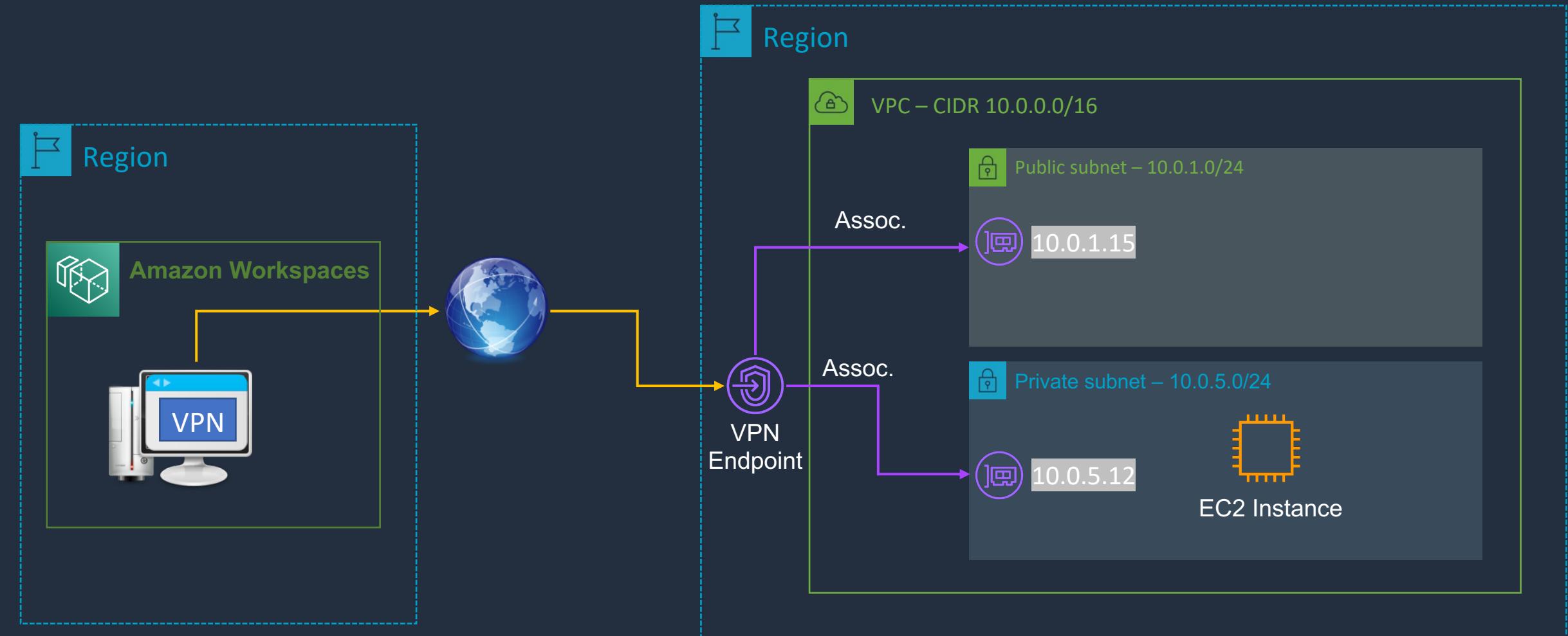


# Deploy AWS Client VPN

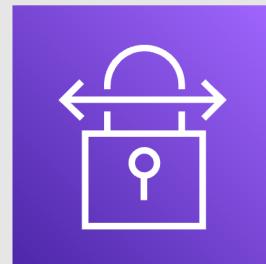




# AWS Client VPN – Hands-On

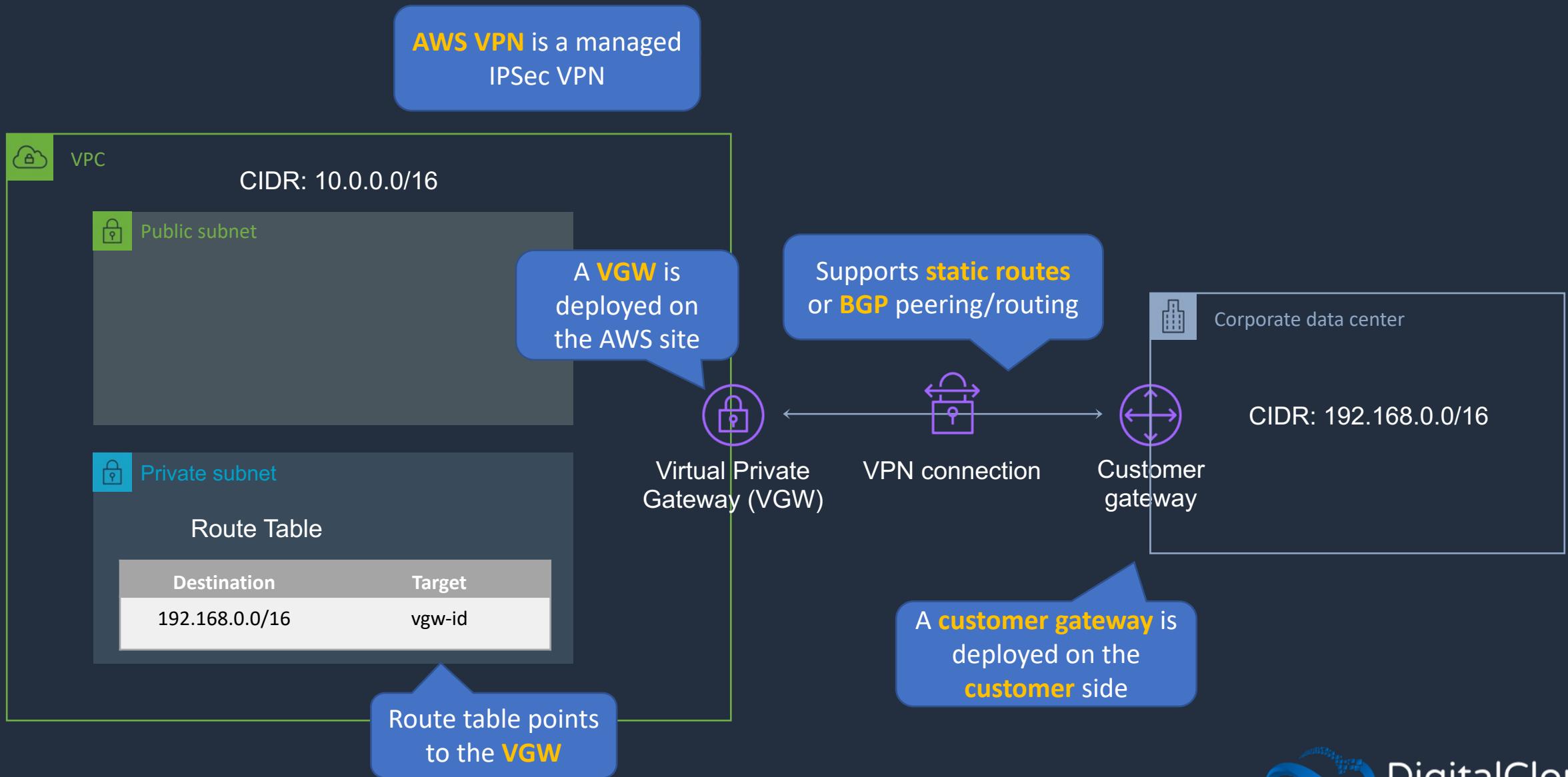


# AWS Site-to-Site VPN





# AWS Site-to-Site VPN

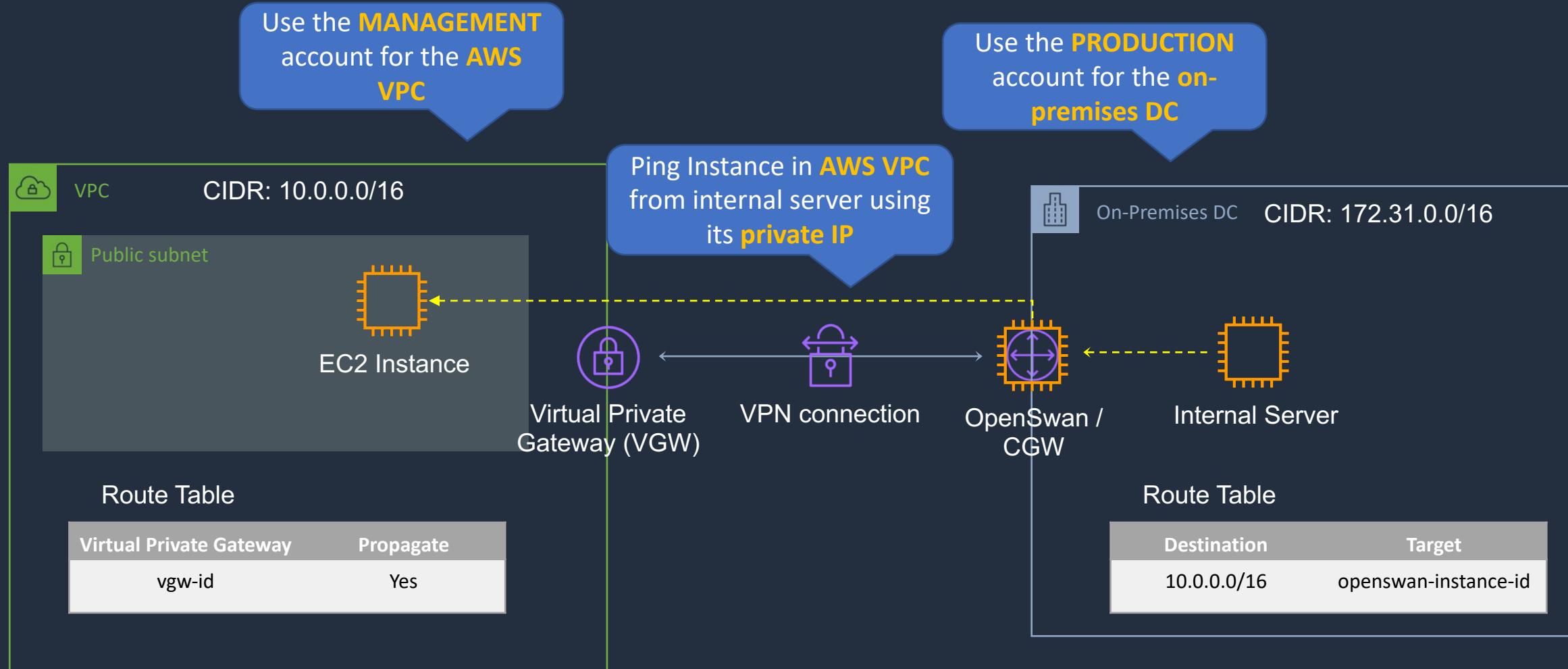


# Deploy AWS Site-to-Site VPN





# AWS Site-to-Site VPN – Hands-On

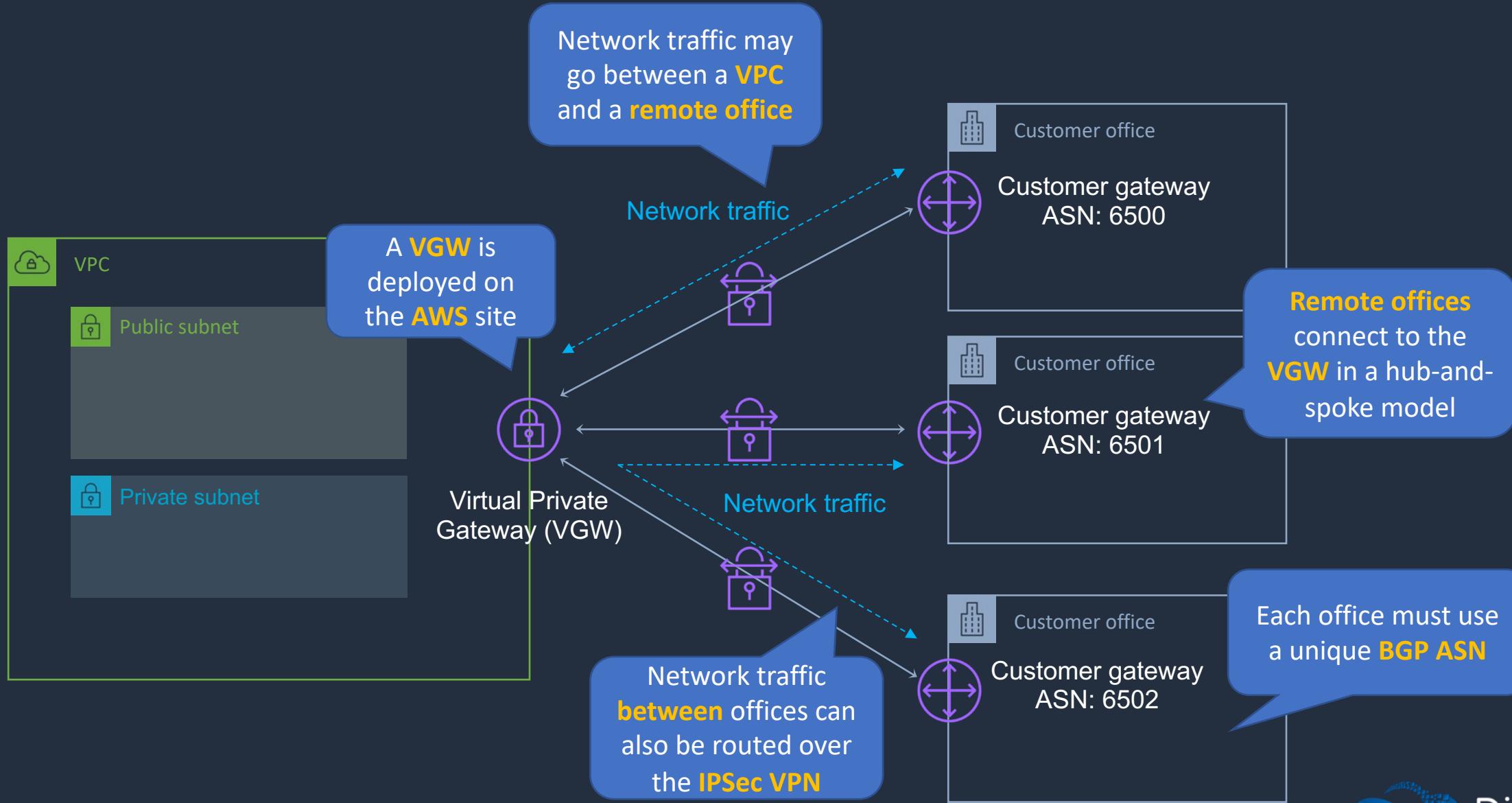


# AWS VPN CloudHub





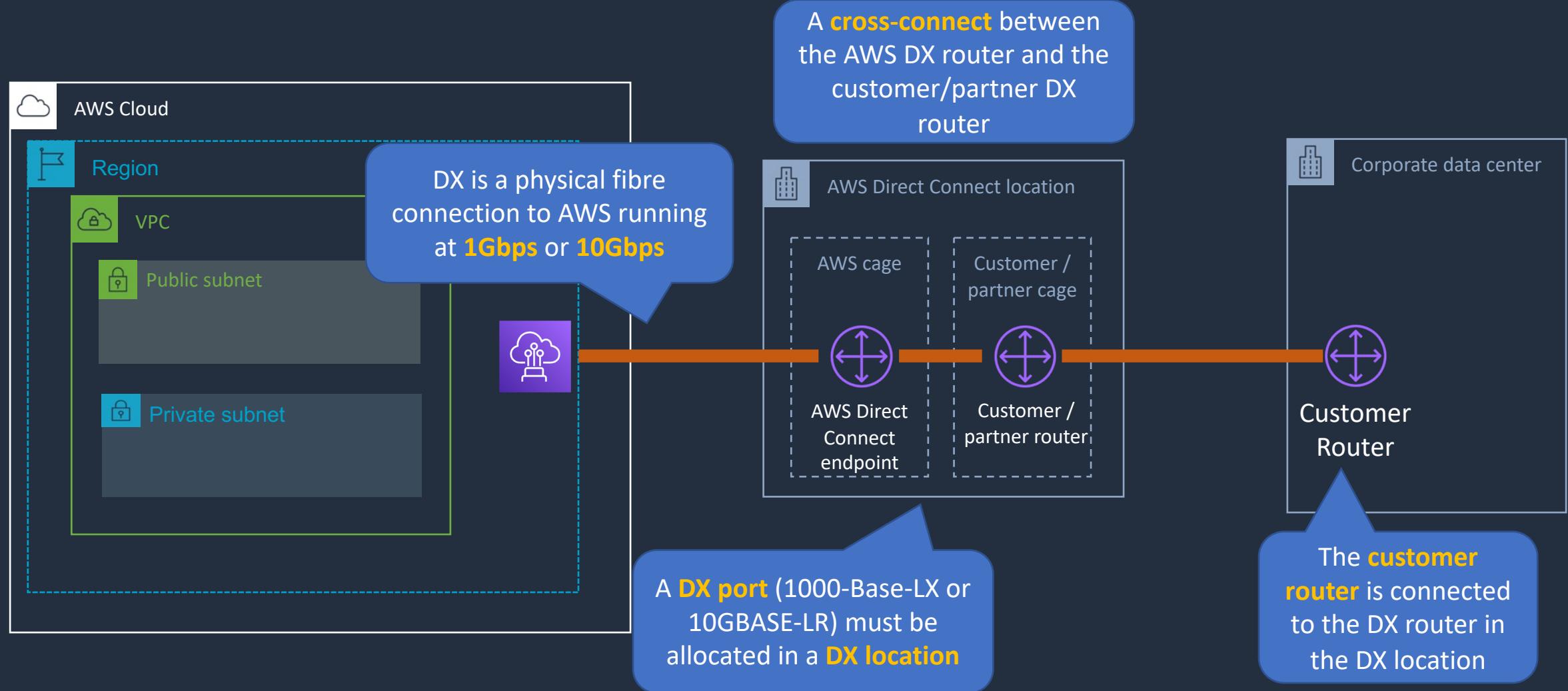
# AWS VPN CloudHub



# AWS Direct Connect (DX)



# AWS Direct Connect (DX)





# AWS Direct Connect Benefits

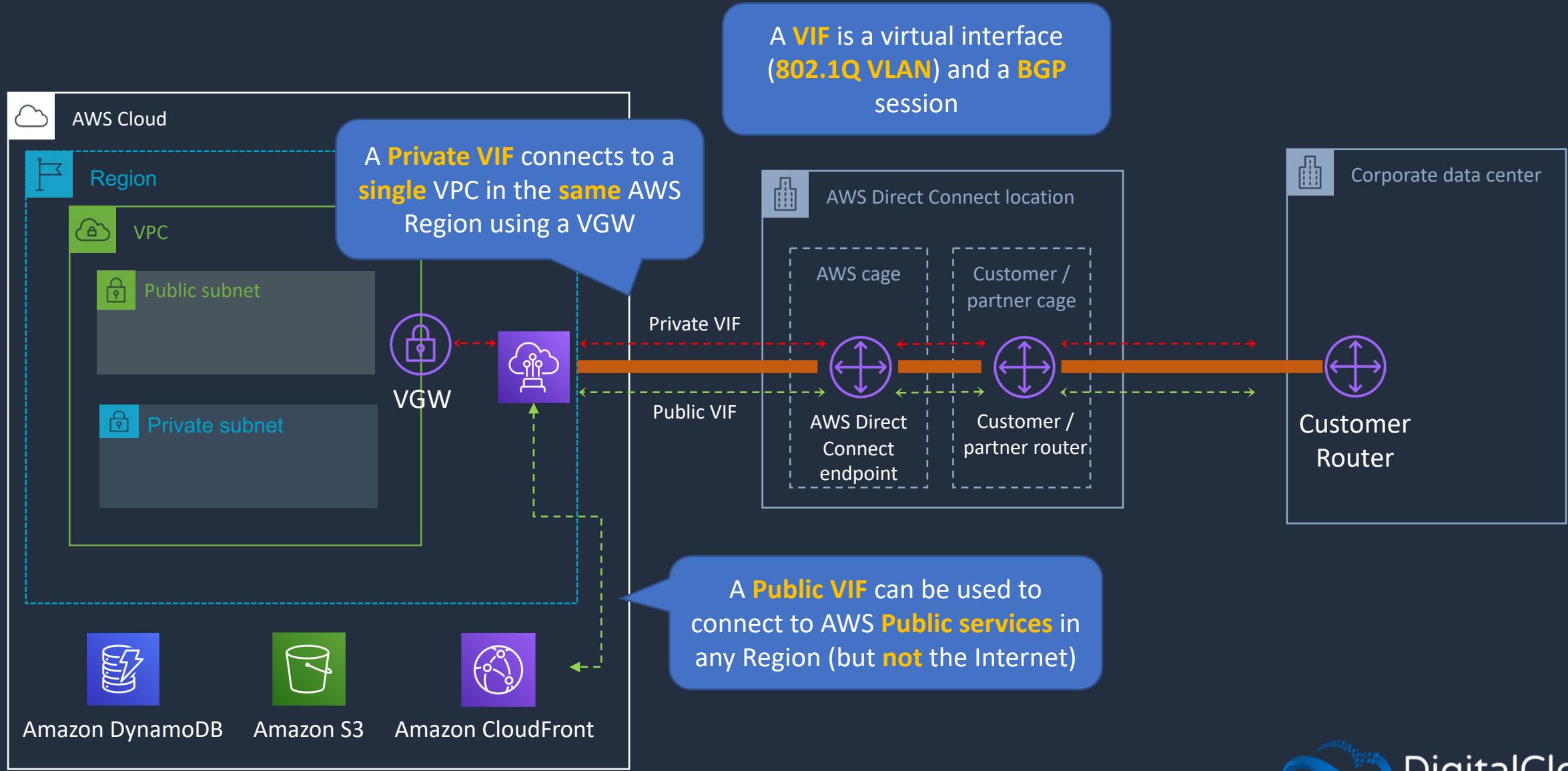
---

---

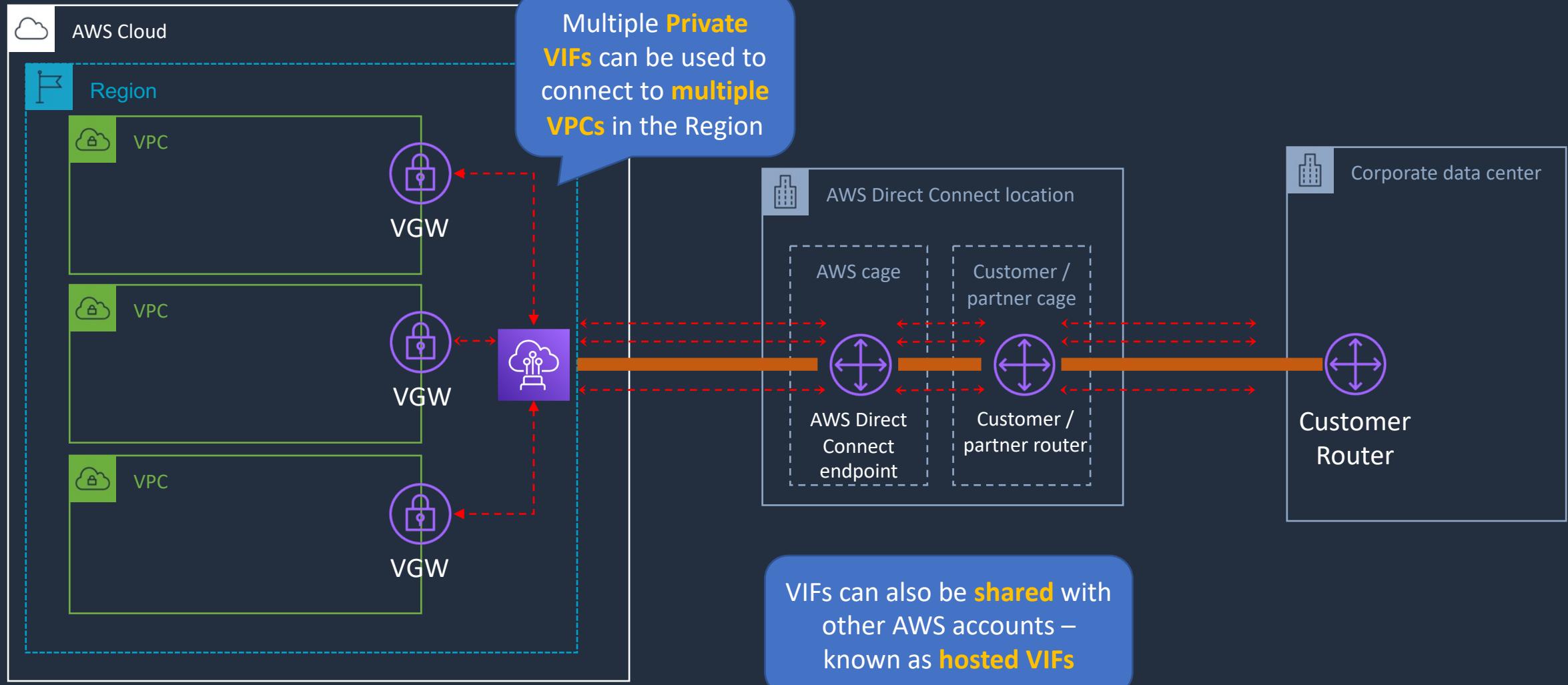
- **Private** connectivity between AWS and your data center / office
- Consistent network experience – increased **speed/latency** & **bandwidth/throughput**
- Lower costs for organizations that transfer **large** volumes of data



# AWS Direct Connect (DX)



# AWS Direct Connect (DX)





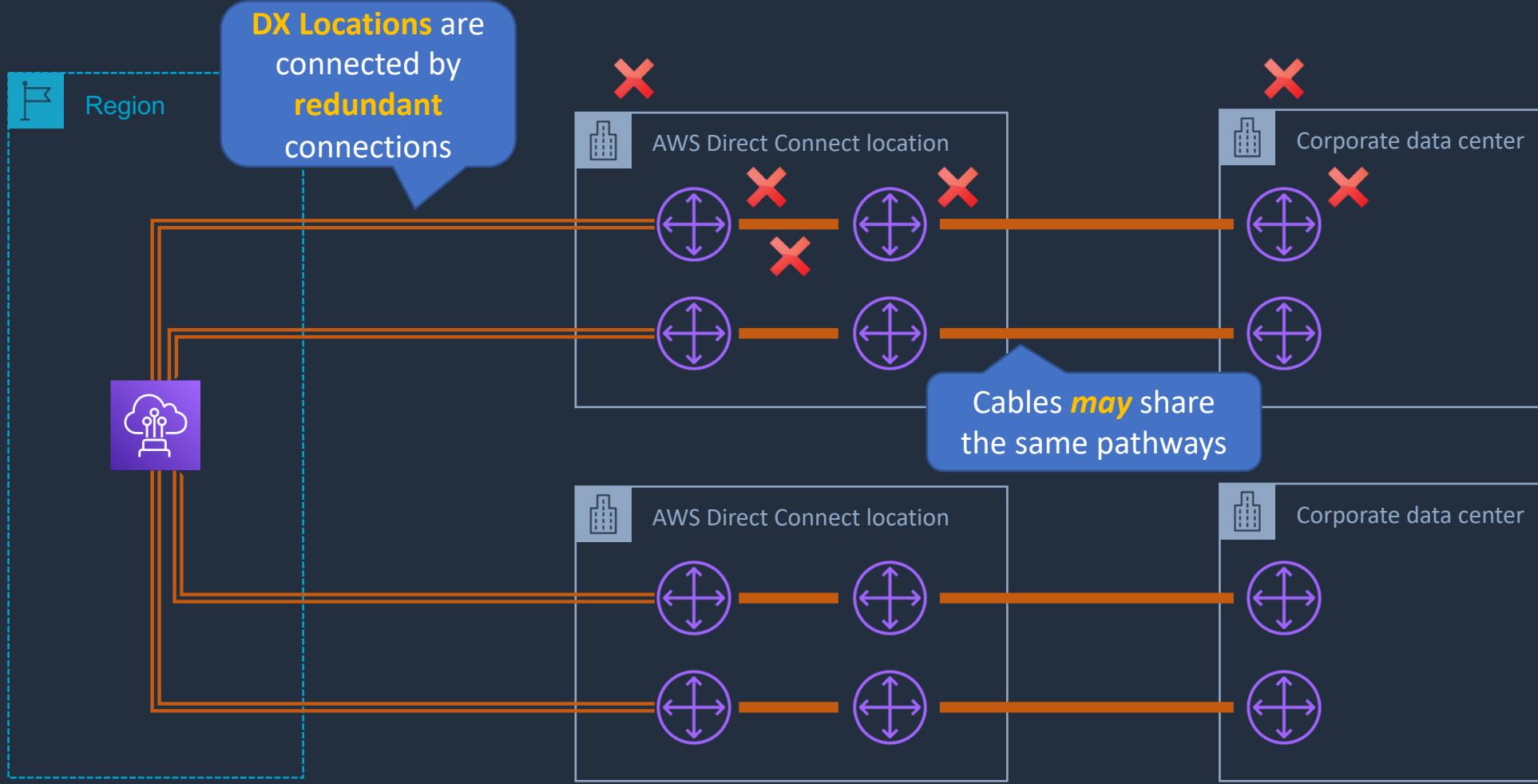
# AWS Direct Connect (DX)

---

- Speeds from 50Mbps to 500Mbps can also be accessed via an APN partner (uses **hosted VIFs** or **hosted connections**):
  - A **hosted VIF** is a single VIF that is shared with other customers (shared bandwidth)
  - A **hosted connection** is a DX connection with a single VIF dedicated to you
- DX Connections are **NOT** encrypted!
- Use an **IPSec S2S VPN** connection over a VIF to add encryption in transit
- Link aggregation groups (**LAGs**) can be used to combine multiple **physical** connections into a single **logical** connection using **LACP** – provides improved **speed**



# DX - Native High Availability

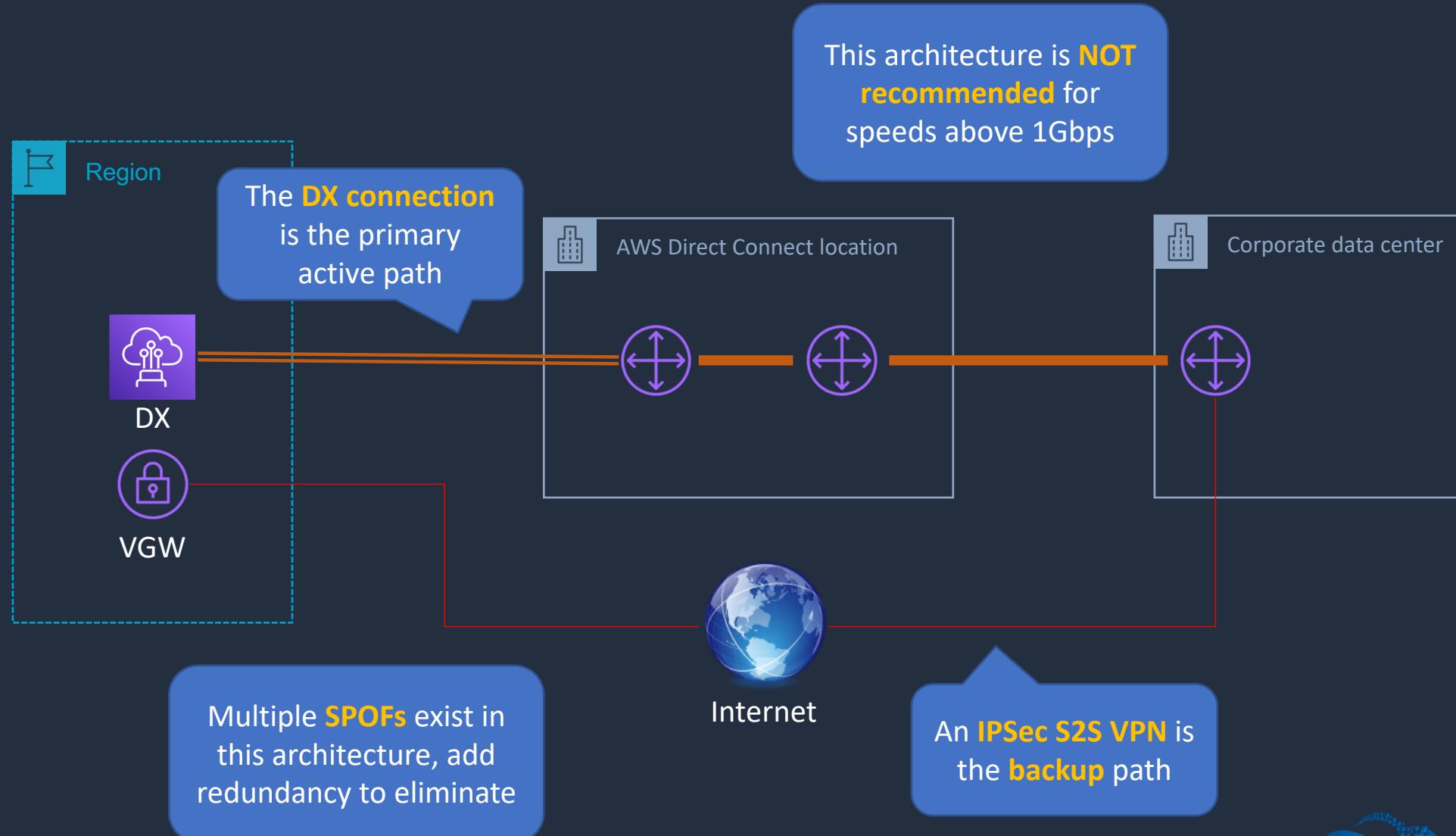


✗ = Single Point of Failure

Multiple DX Locations exist in metropolitan areas where AWS has Regions



# Direct Connect + IPSec S2S VPN



# Create Direct Connect Connection



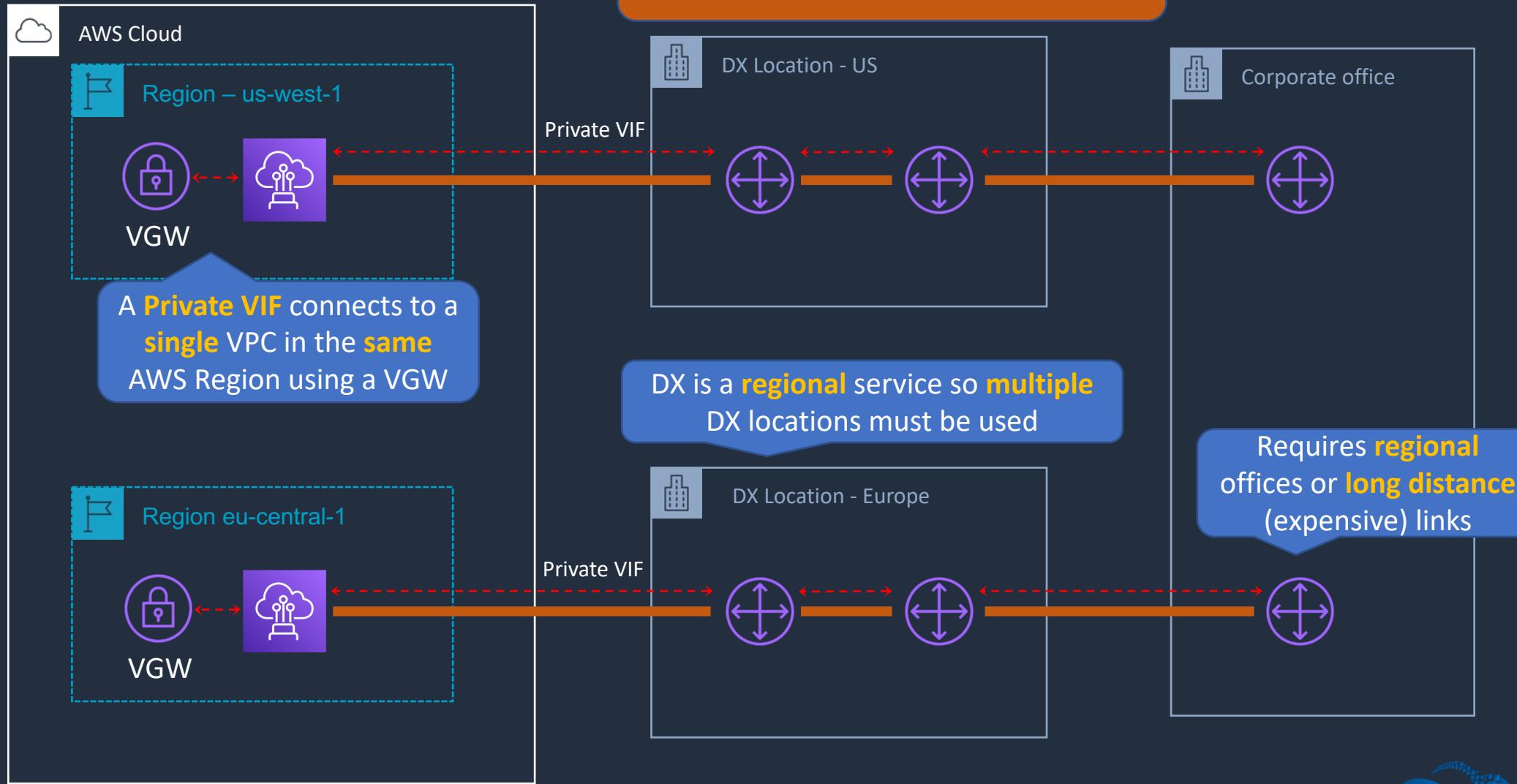
# AWS Direct Connect Gateway





# Direct Connect - Multiple Regions

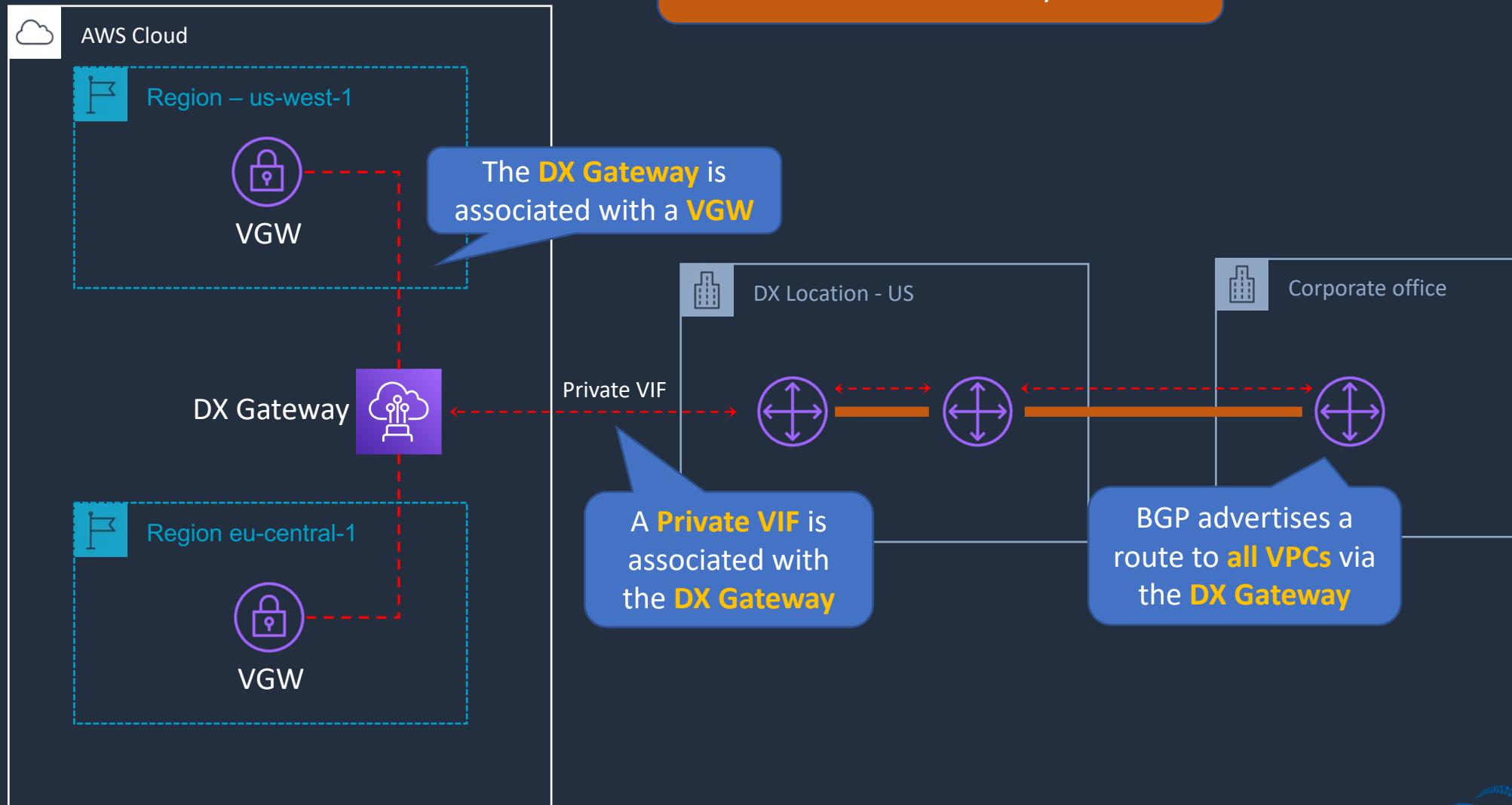
Example architecture **without** AWS Direct Connect Gateway





# Direct Connect - Multiple Regions

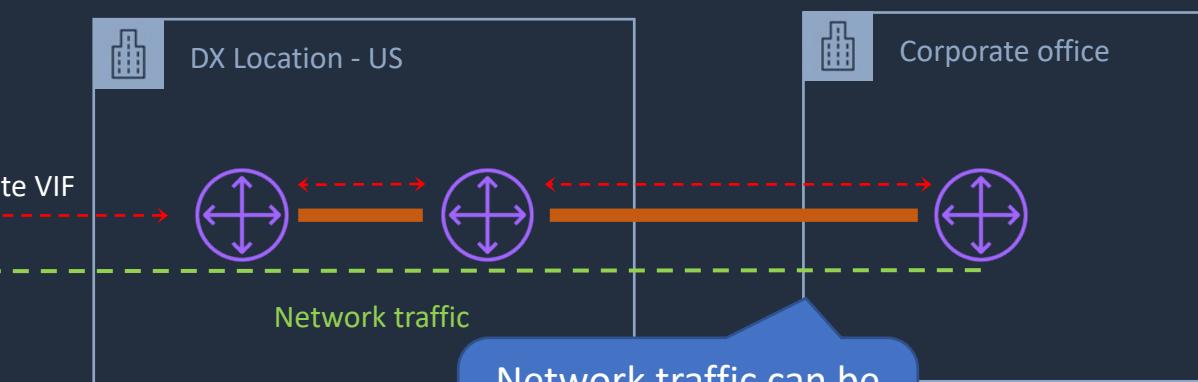
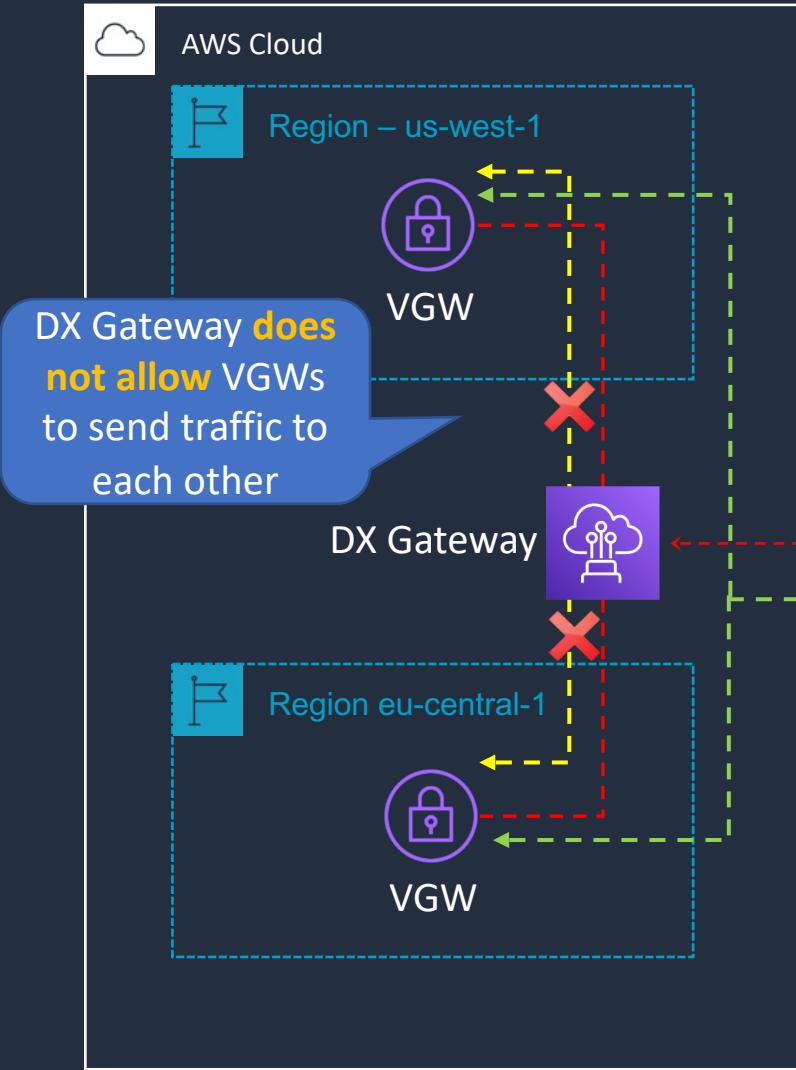
Example architecture **with** AWS Direct Connect Gateway



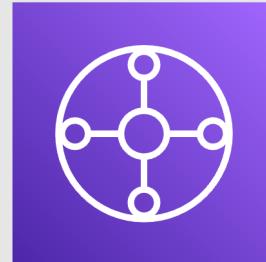


# Direct Connect - Multiple Regions

Example architecture **with** AWS Direct Connect Gateway

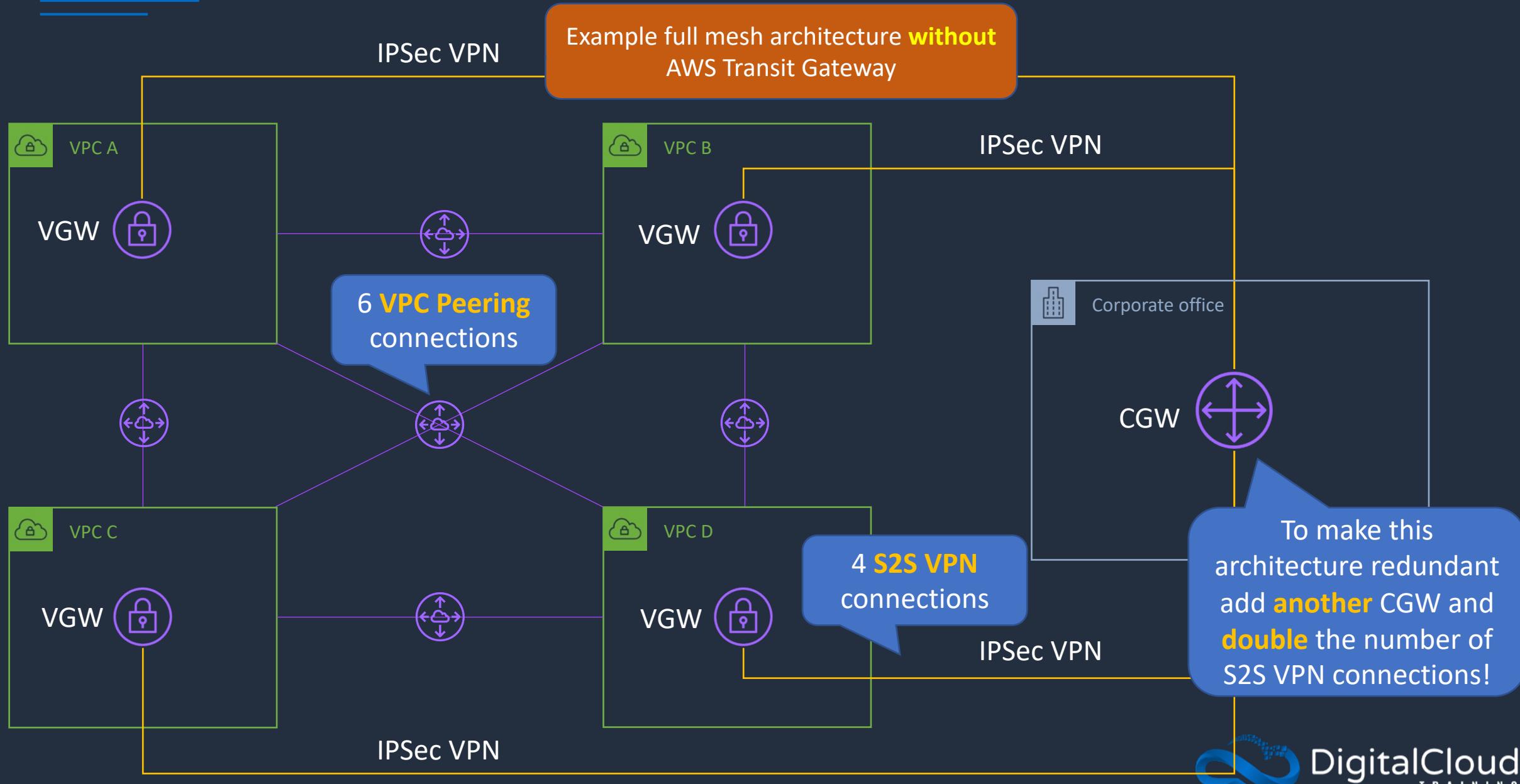


# AWS Transit Gateway



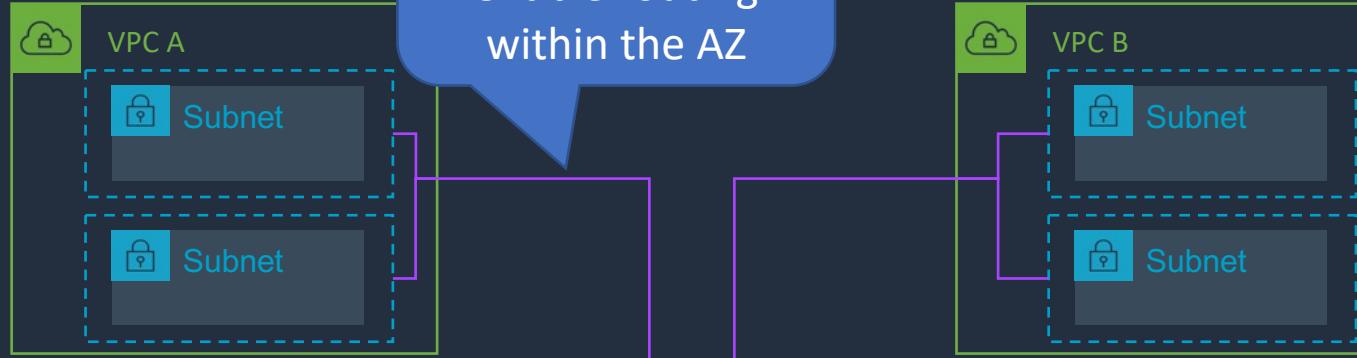


# AWS Transit Gateway



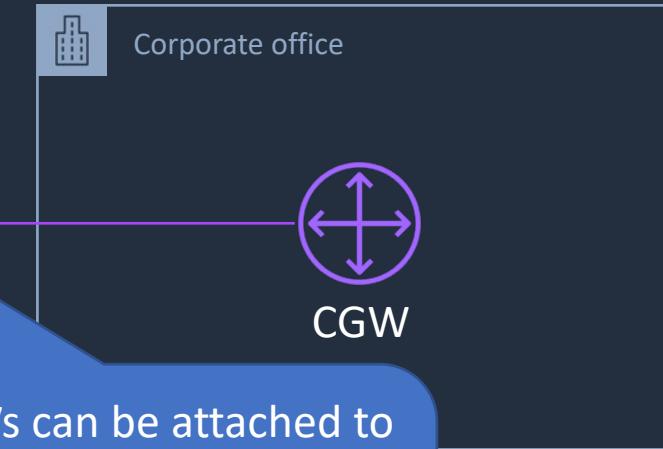
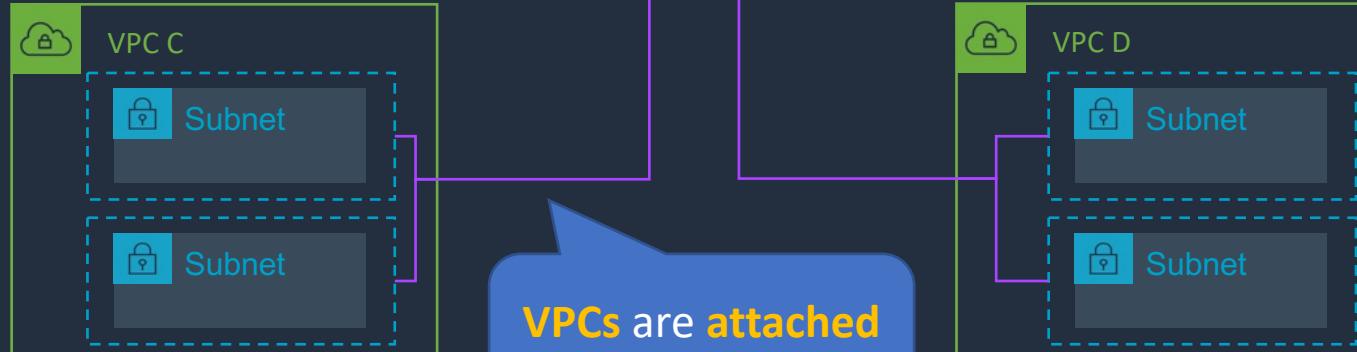


# AWS Transit Gateway



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

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

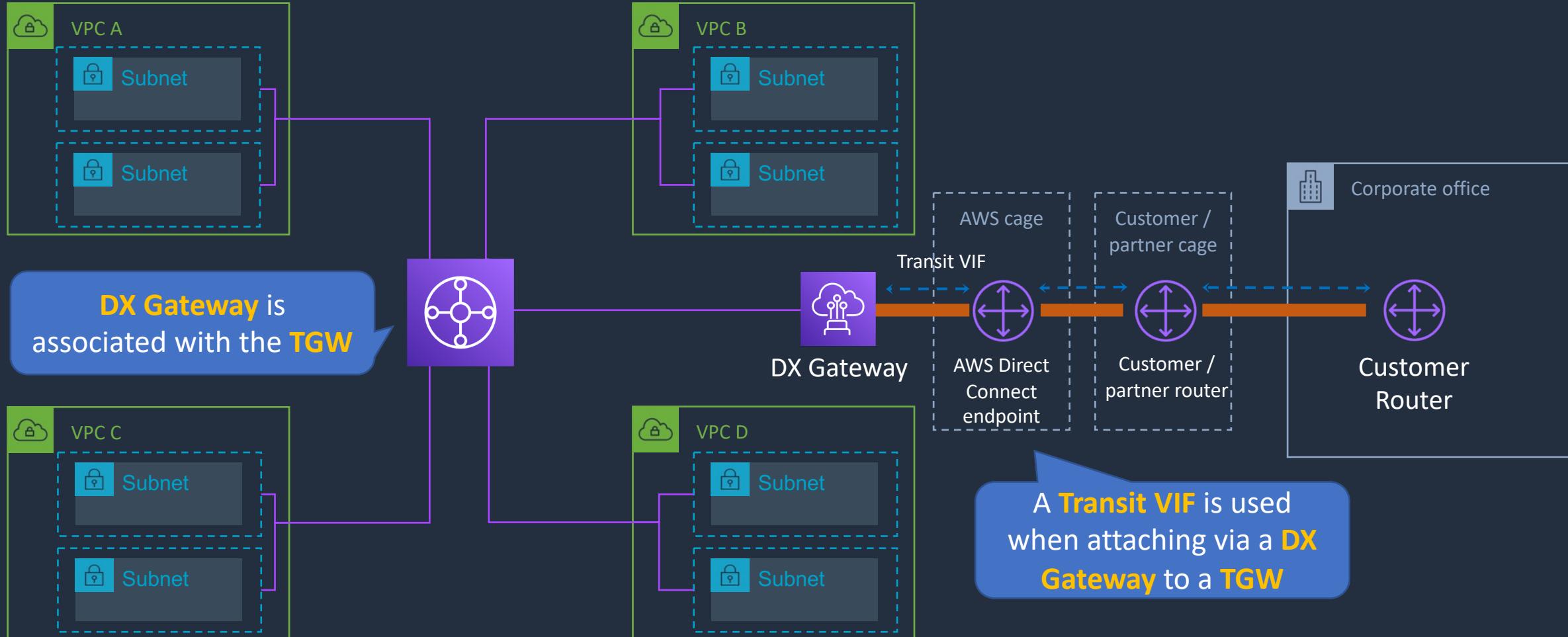


TGWs can be attached to **VPNs, Direct Connect Gateways, 3<sup>rd</sup> party appliances** and **TGWs** in other Regions/accounts



# AWS TGW + DX Gateway

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



# SECTION 7

## Compute, Auto Scaling, and Load Balancing

# 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

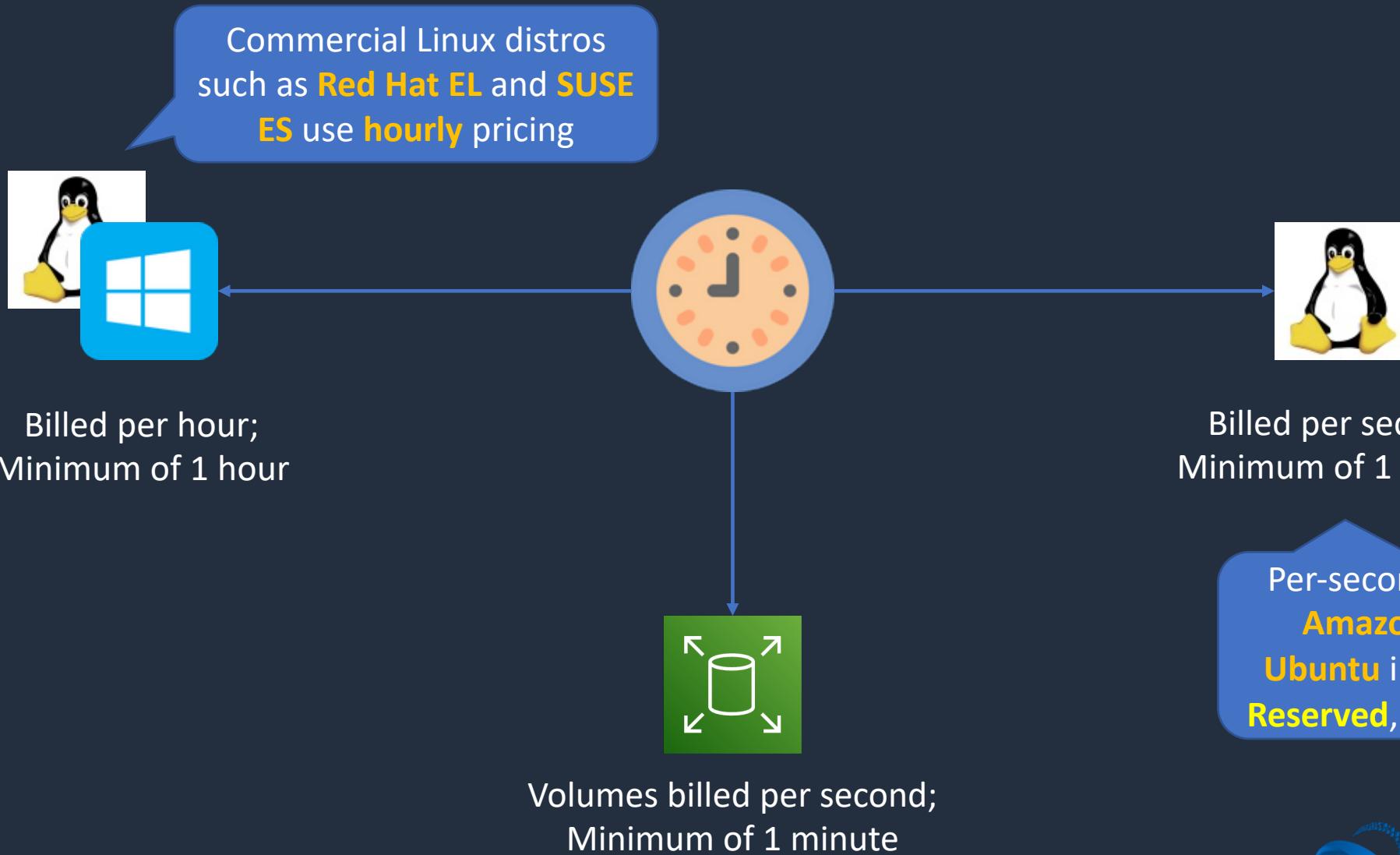
## Dedicated Instances

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

## Savings Plans

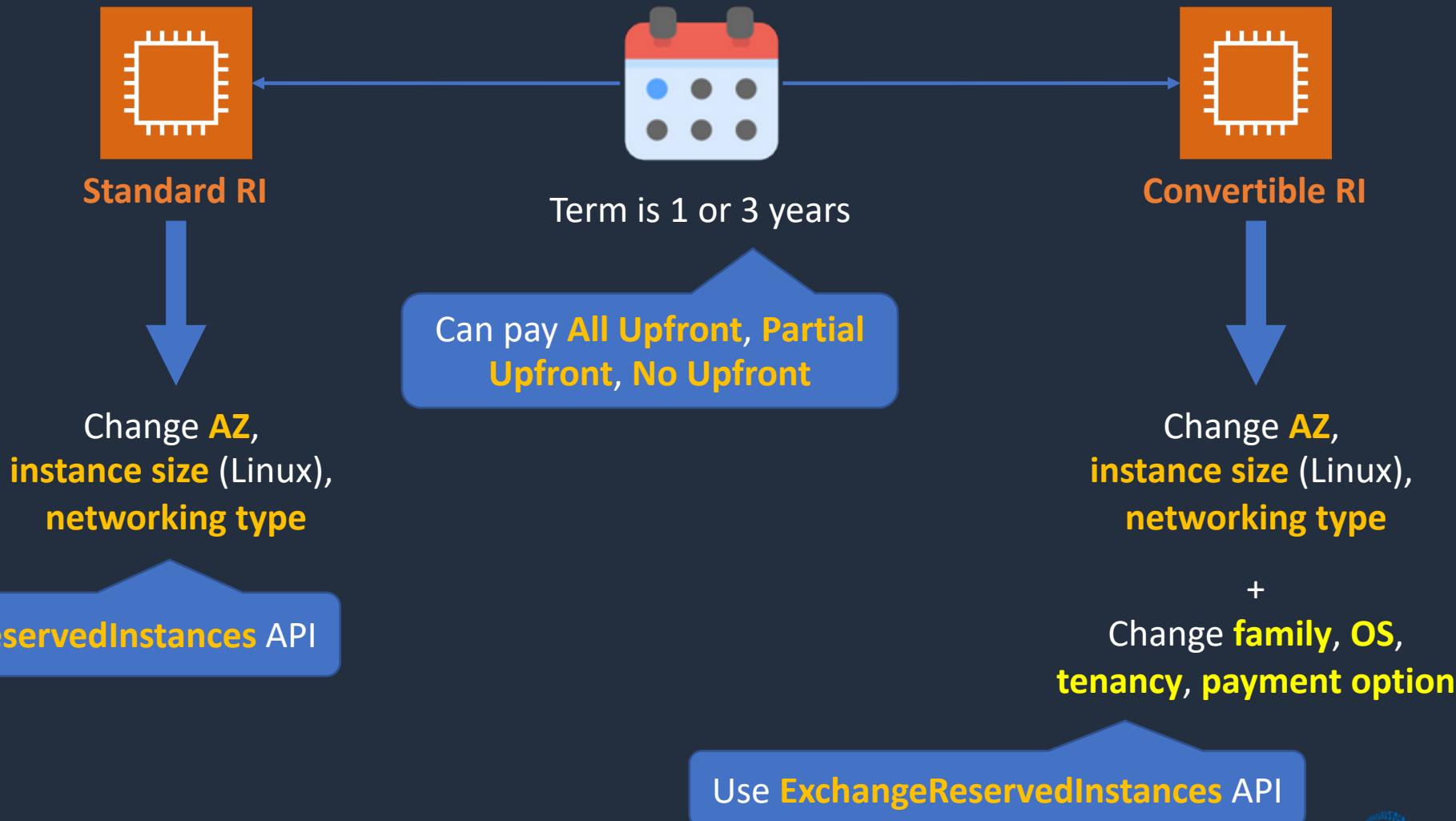
Commitment to a consistent amount of usage (EC2 + Fargate + Lambda); Pay by \$/hour; 1 or 3-year commitment

# \$ Amazon EC2 Billing





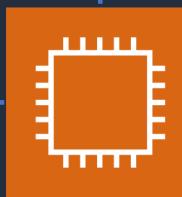
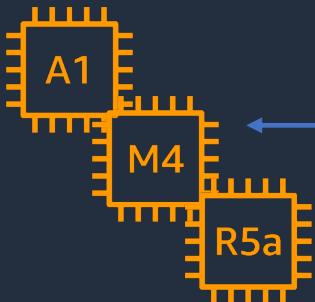
# Amazon EC2 Reserved Instances (RIs)





# 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

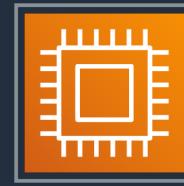


**Important**  
We do not have any capacity for purchasing Scheduled Reserved Instances or any plans to make it available in the future. To reserve capacity, use [On-Demand Capacity Reservations](#) instead. For discounted rates, use [Savings Plans](#).

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

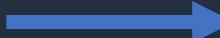


**EC2 Fleet:** launches and maintains specified number of Spot / On-Demand / Reserved instances in a **single API call**

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

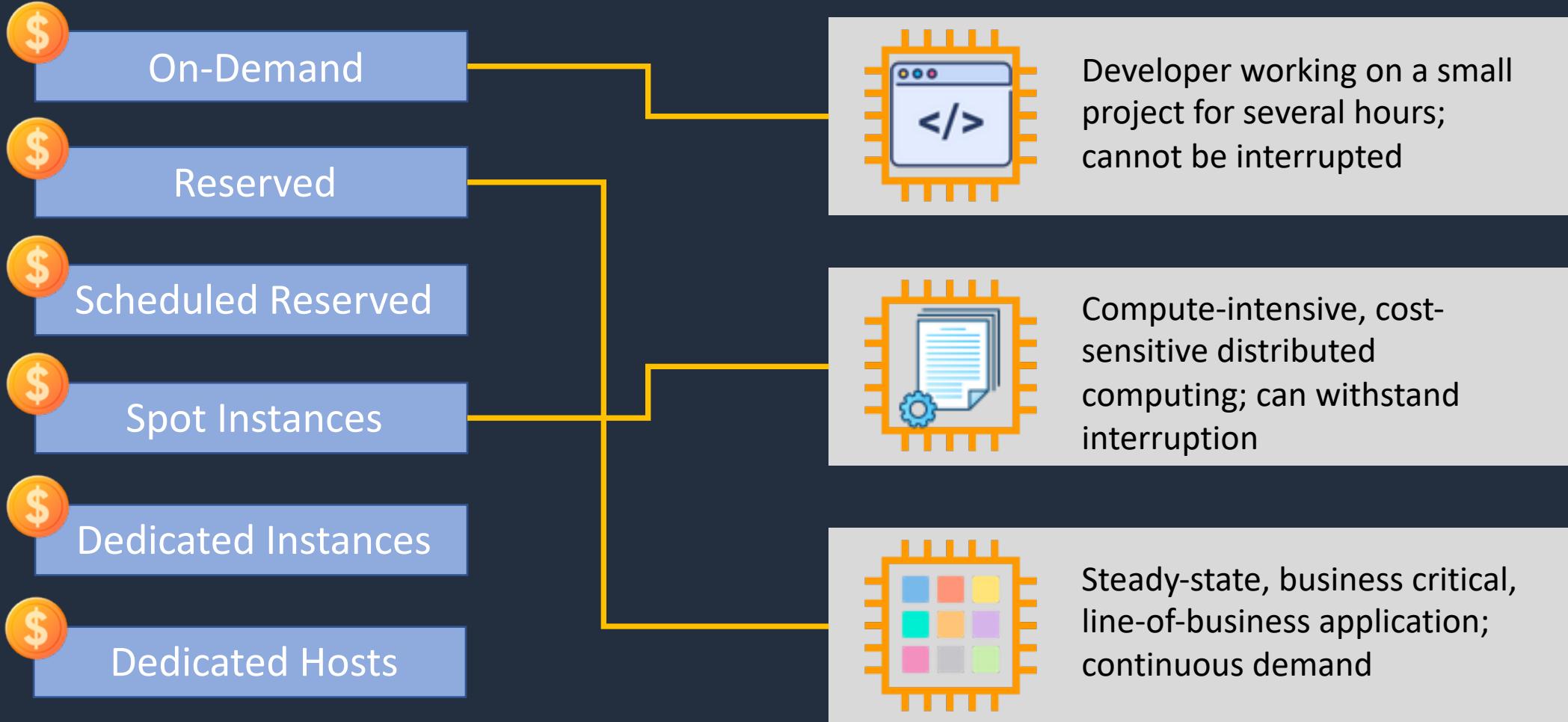
Characteristic	Dedicated Instances	Dedicated Hosts
Enables the use of dedicated physical servers	X	X
Per instance billing (subject to a \$2 per region fee)	X	
Per host billing		X
Visibility of sockets, cores, host ID		X
Affinity between a host and instance		X
Targeted instance placement		X
Automatic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

# Amazon EC2 Pricing Use Cases



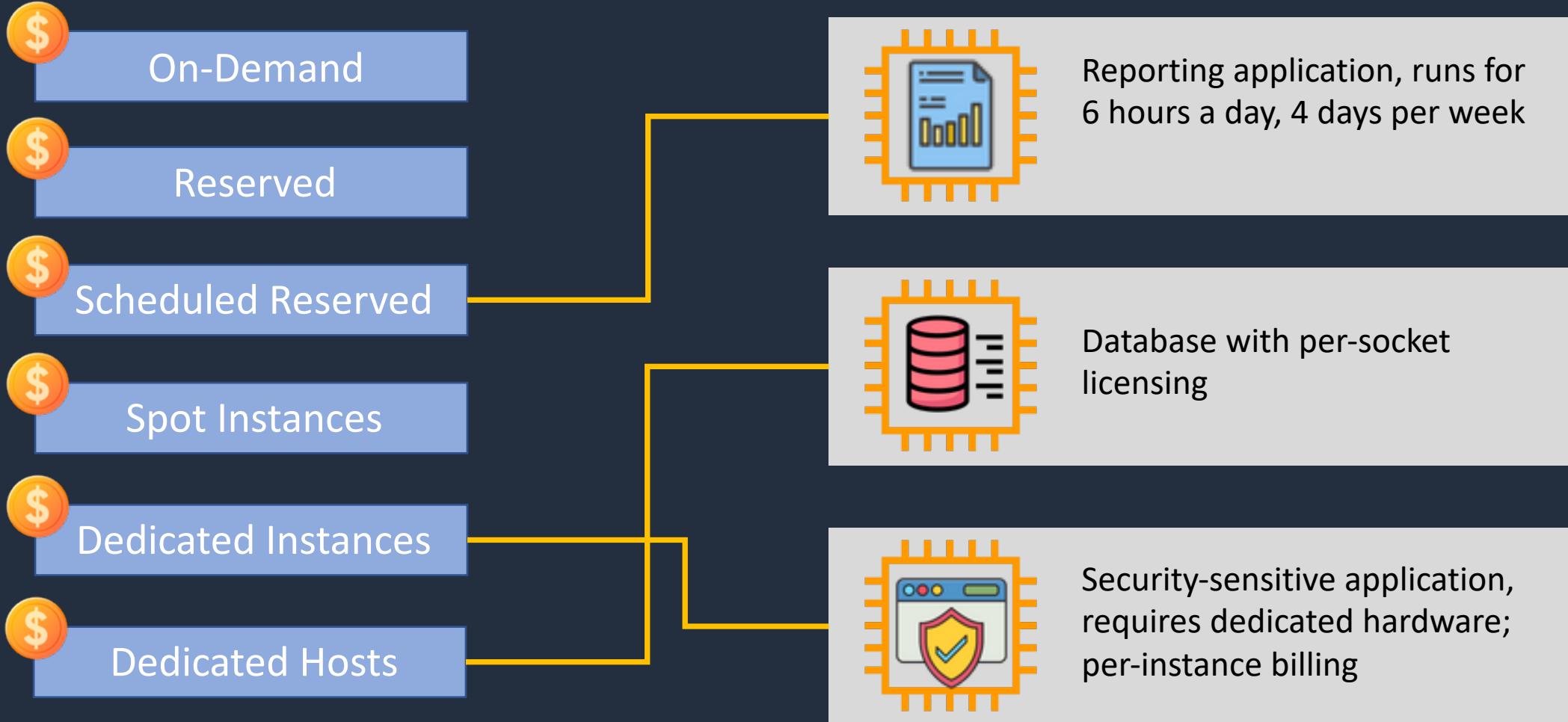


# Amazon EC2 Pricing Use Cases





# Amazon EC2 Pricing Use Cases



# Bootstrapping AMIs





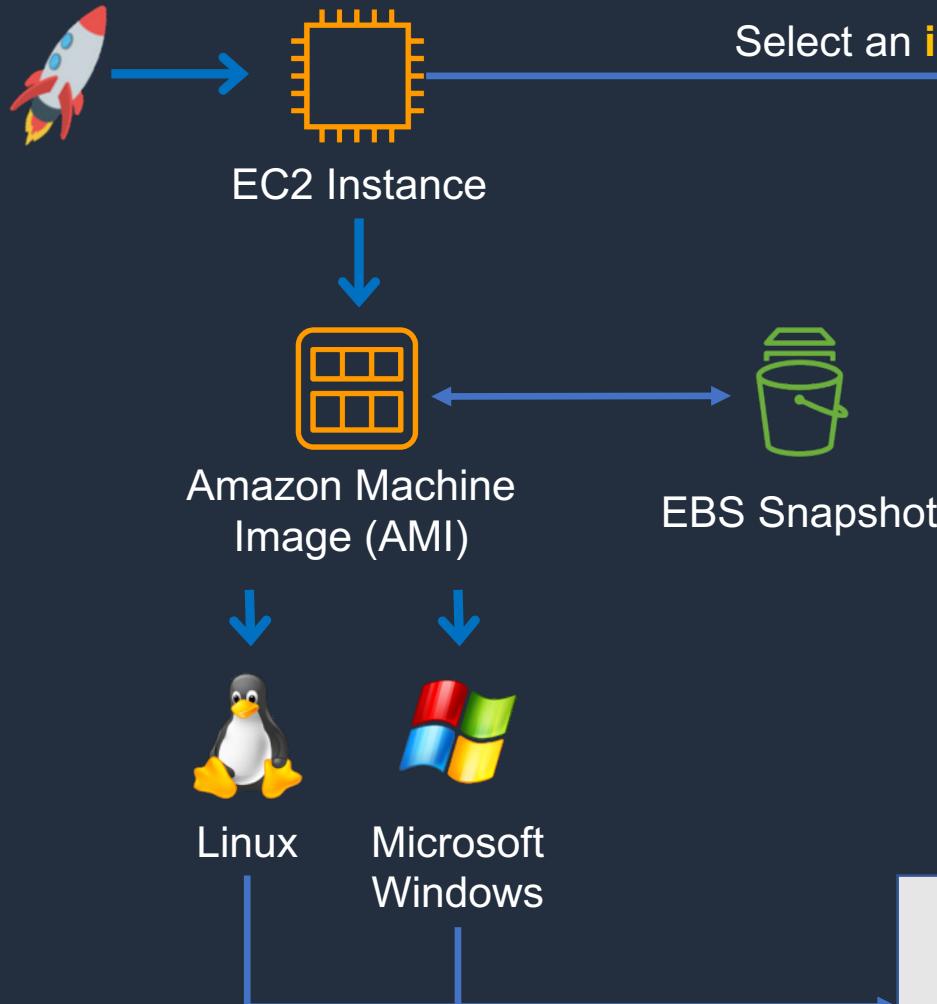
# AMIs and Bootstrapping

---

How can we launch our EC2 instances and install dependencies, applications, software and security updates, and configure customizations **quickly**?



# Customized AMIs



Select an **instance type**

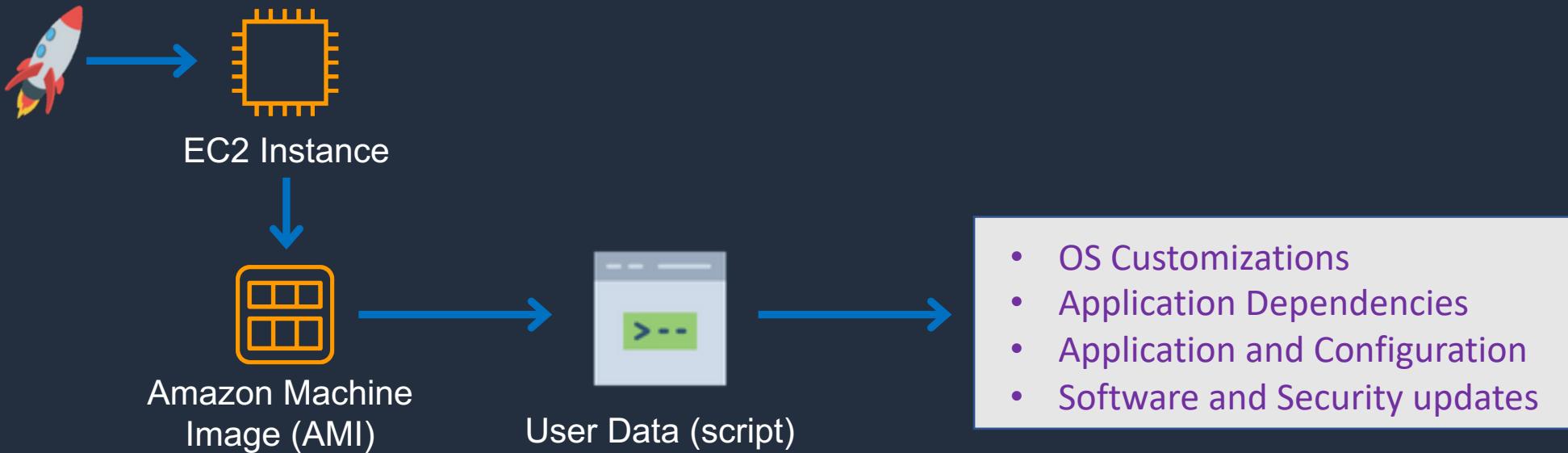
Family	Type	vCPUs	Memory (GiB)
General purpose	t2.micro	1	1
Compute optimized	c5n.large	2	5.25
Memory optimized	r5ad.large	2	16
Storage optimized	d2.xlarge	4	30.5
GPU instances	g2.2xlarge	8	15

- OS Customizations
- Application Dependencies
- Application and Configuration
- Software and Security updates





# User Data



A combined approach would use a  
**customized AMI + User Data**

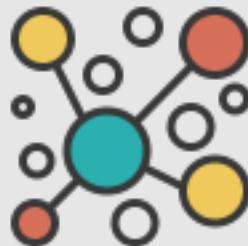


Use **automation** and **configuration** tools



- AWS CloudFormation
- AWS OpsWorks
- AWS Systems Manager
- AWS CodePipeline, CodeDeploy etc.
- Chef and Puppet
- Jenkins

# EC2 Placement Group Use Cases





# EC2 Placement Groups

---

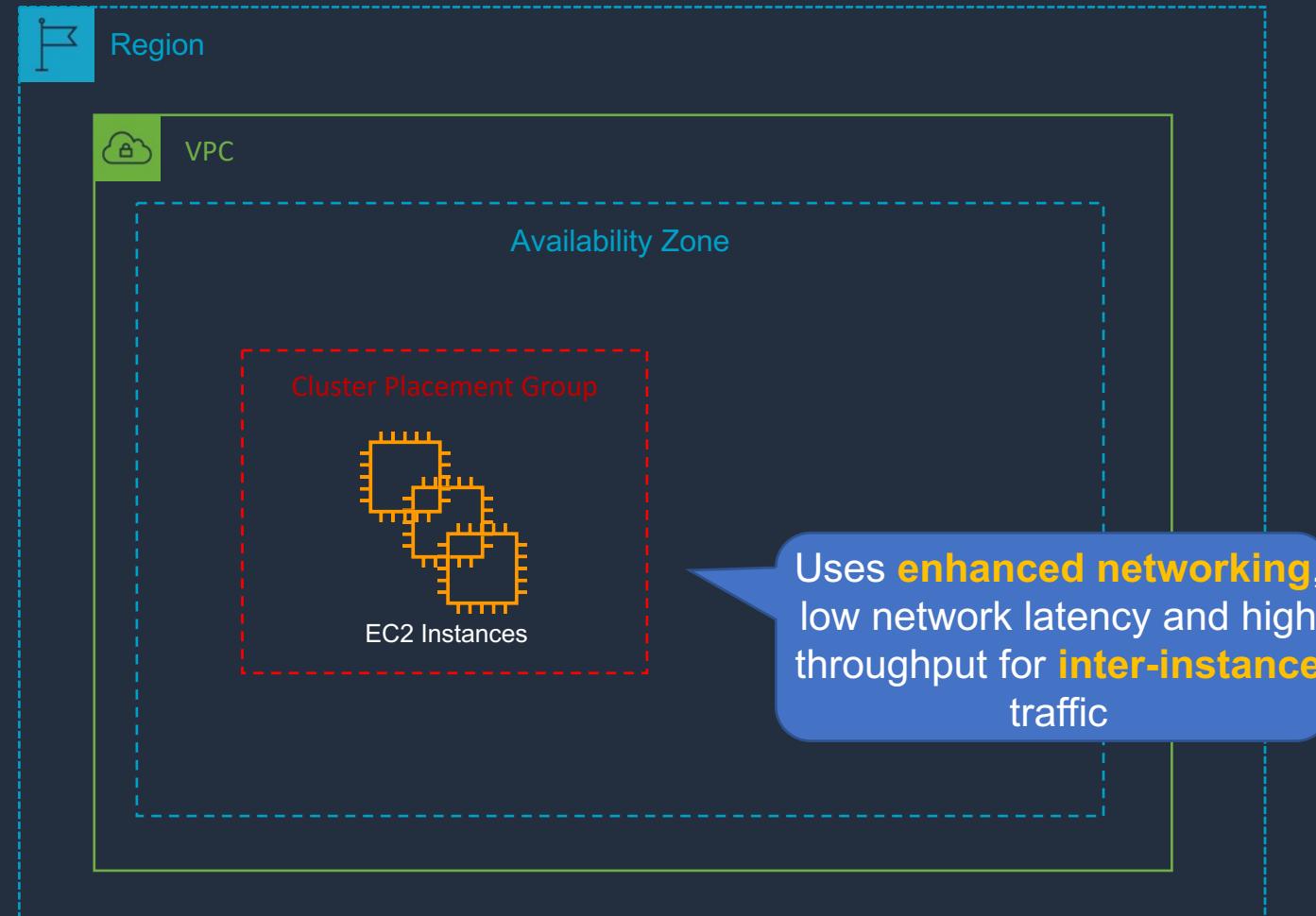
---

- **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**

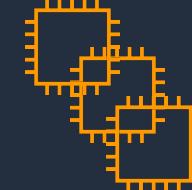
Availability Zone

Partition 1



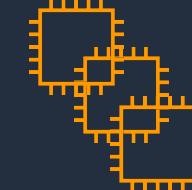
EC2 Instances

Partition 2



EC2 Instances

Partition 3

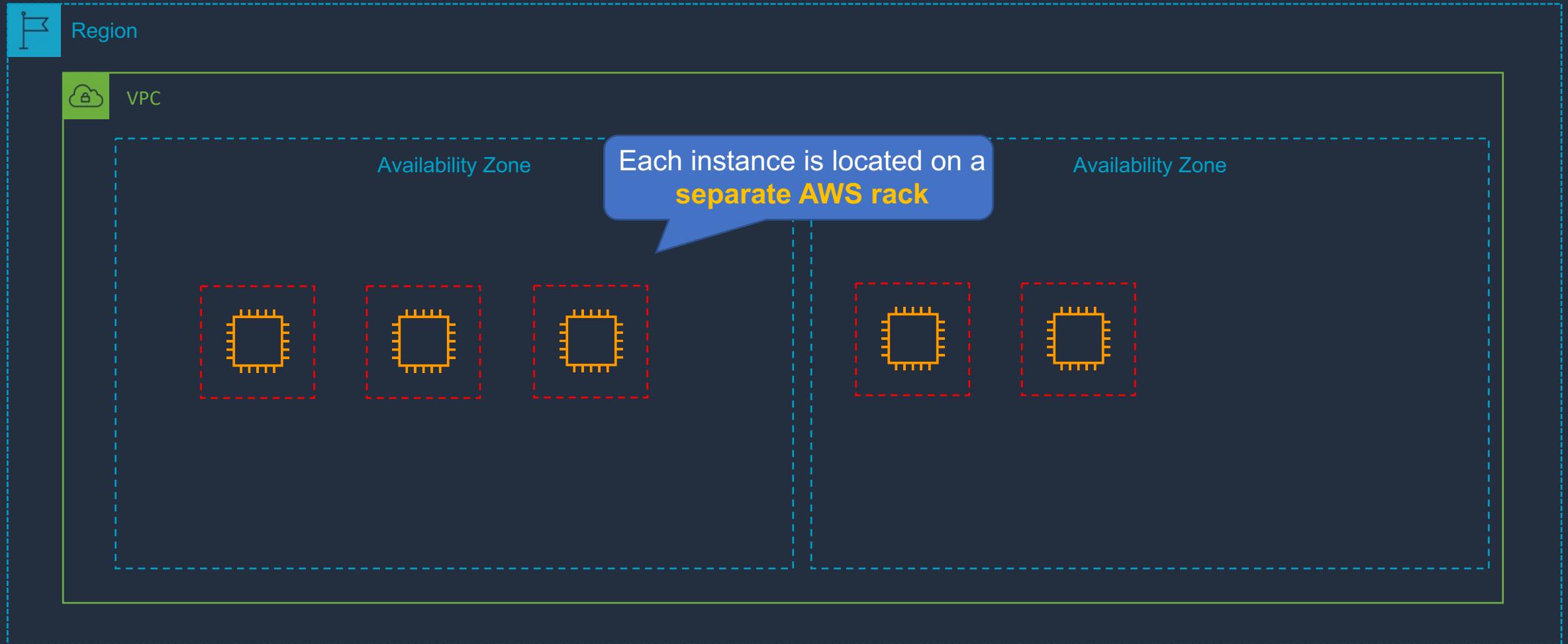


EC2 Instances

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

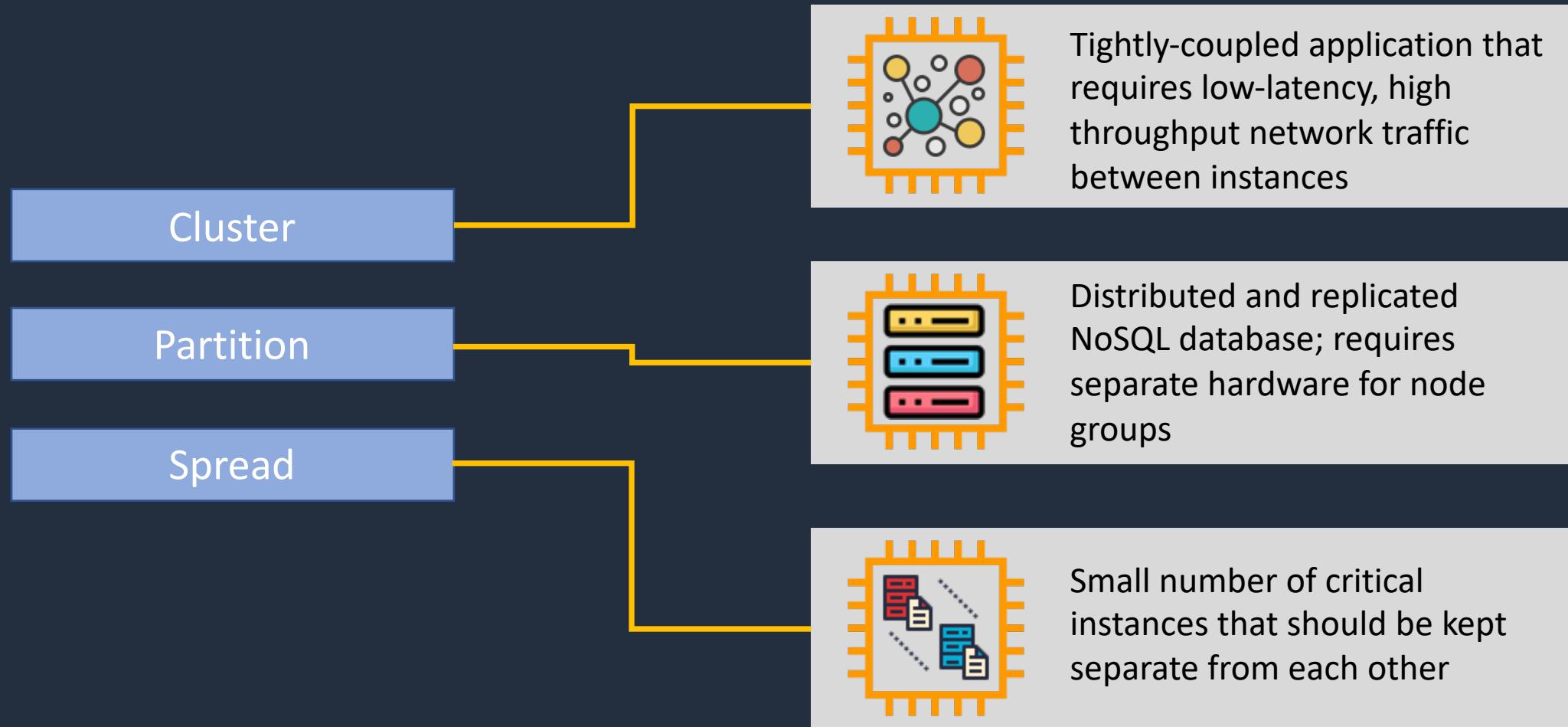


# Spread Placement Group





# EC2 Placement Group Use Cases

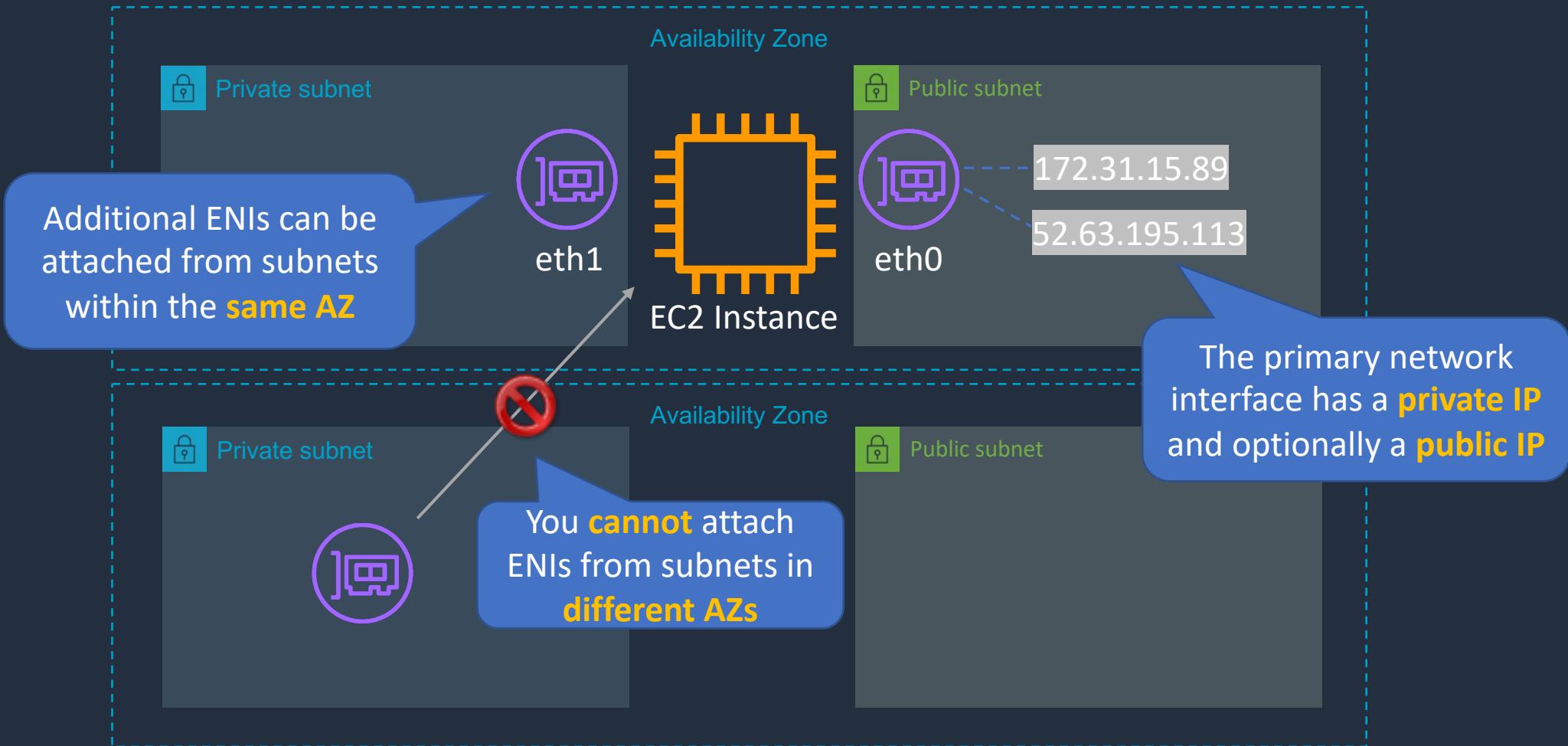


# Network Interfaces (ENI, ENA, EFA)





# 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

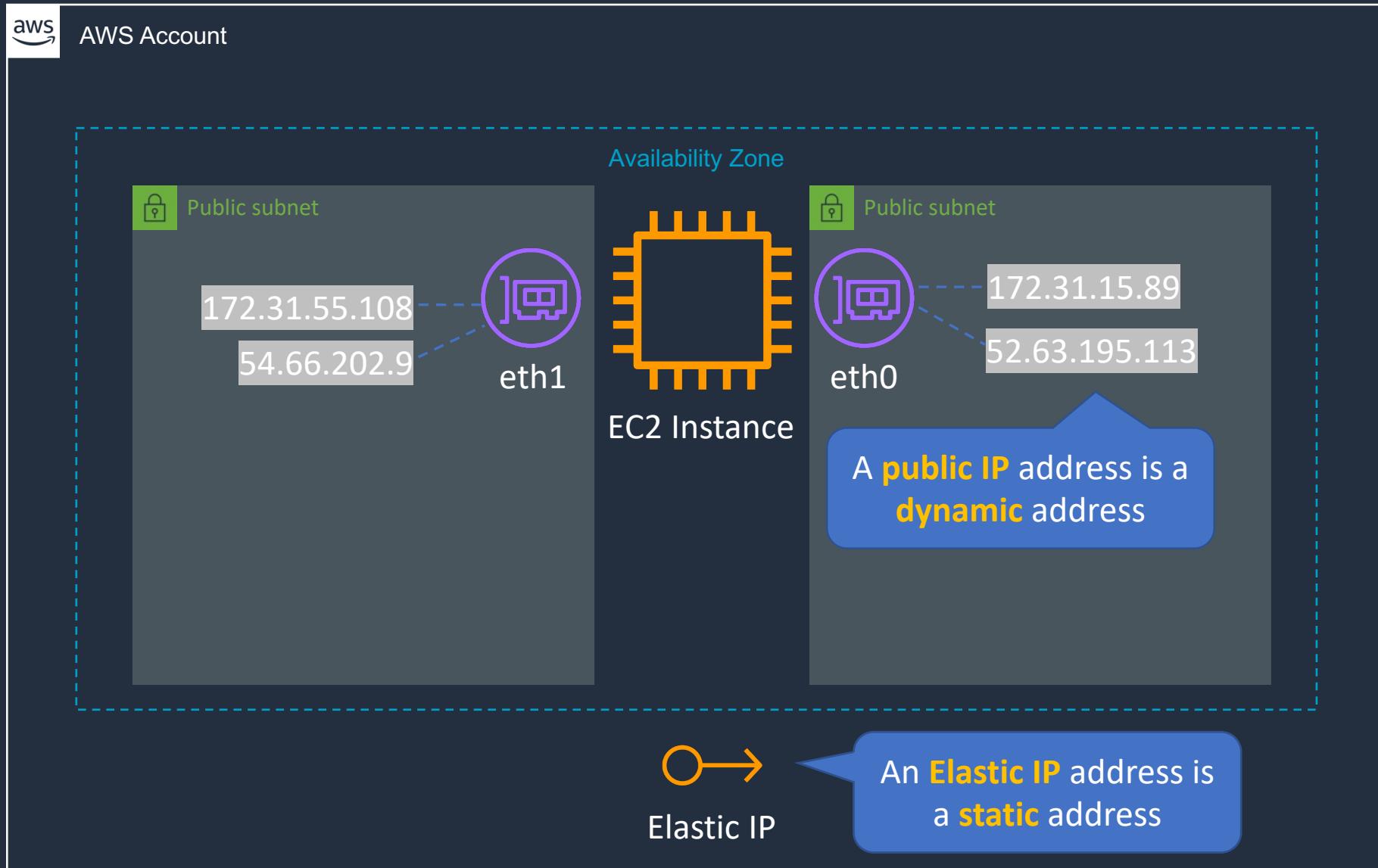
# Working with ENIs



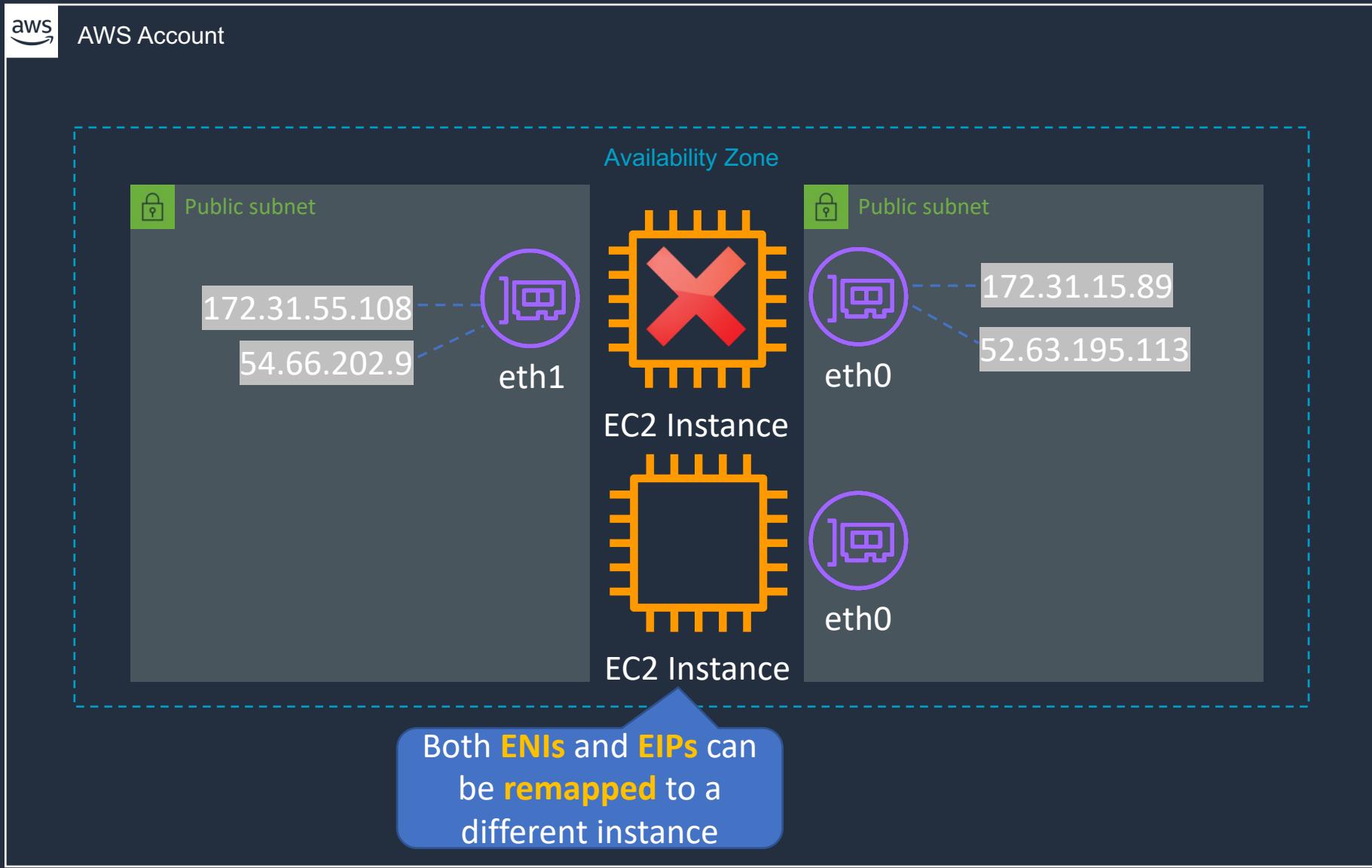
# Public, Private and Elastic IP Addresses



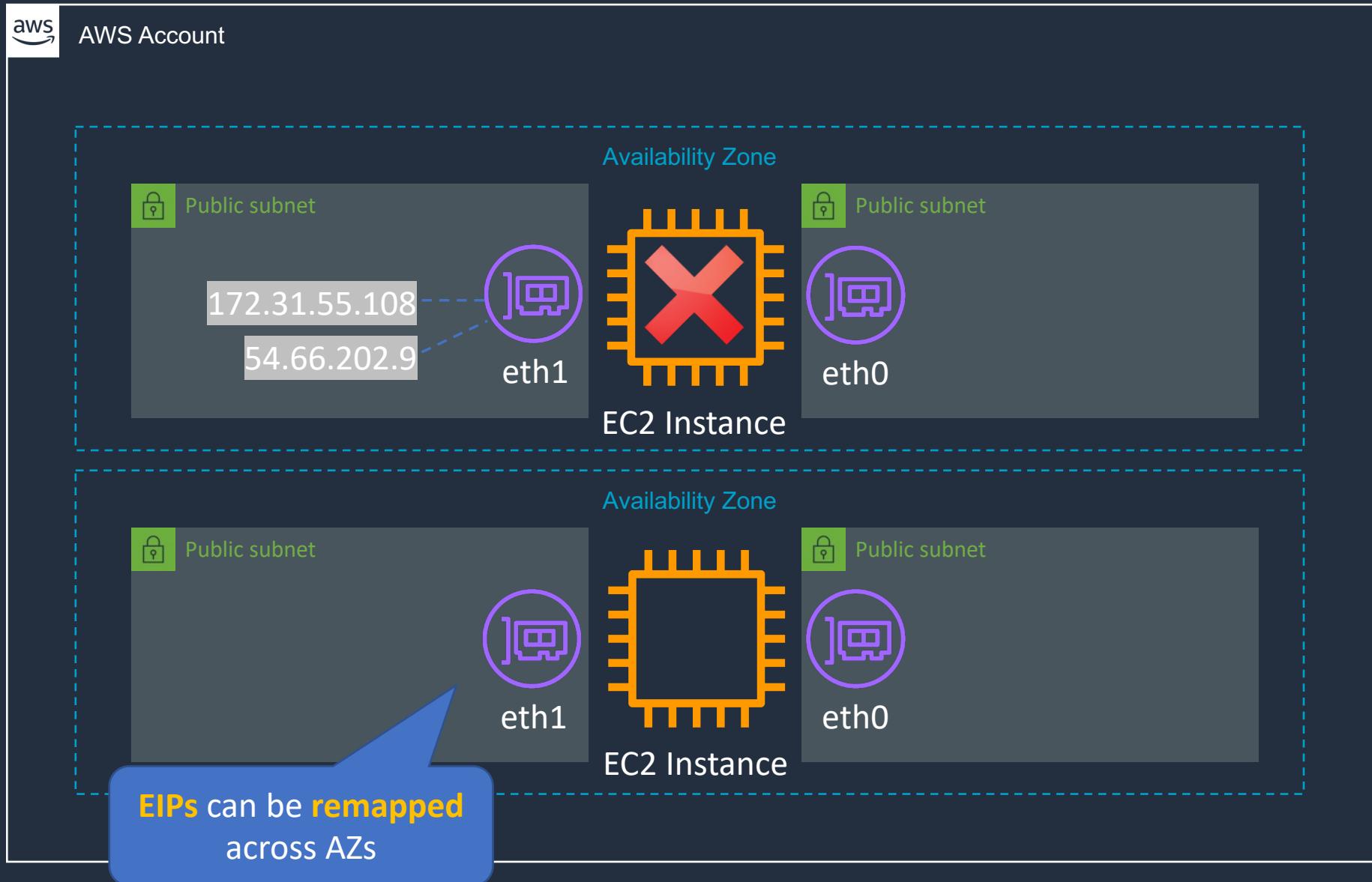
# → Public, Private and Elastic IP Addresses



# → Public, Private and Elastic IP Addresses



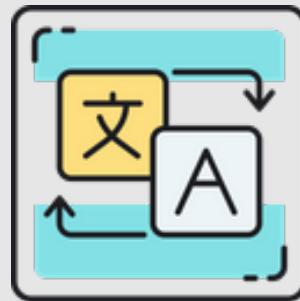
# → Public, Private and Elastic IP Addresses



# → 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>

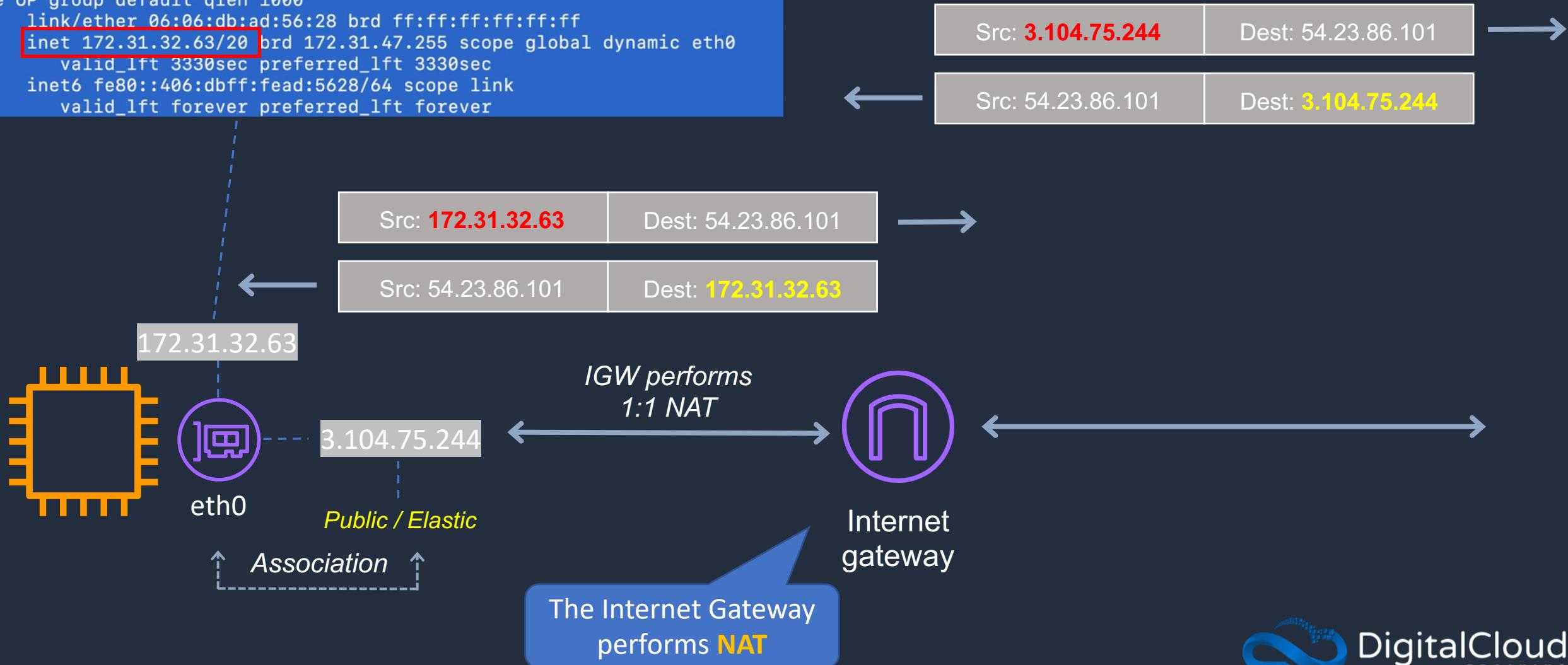
# NAT for Public Addresses





# NAT for Public Addresses

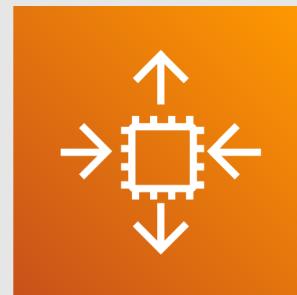
```
[ec2-user@ip-172-31-32-63 ~]$ ip addr show eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9001 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 06:06:db:ad:56:28 brd ff:ff:ff:ff:ff:ff
    inet 172.31.32.63/20 brd 172.31.47.255 scope global dynamic eth0
        valid_lft 3330sec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



# Working with EC2 IP addresses

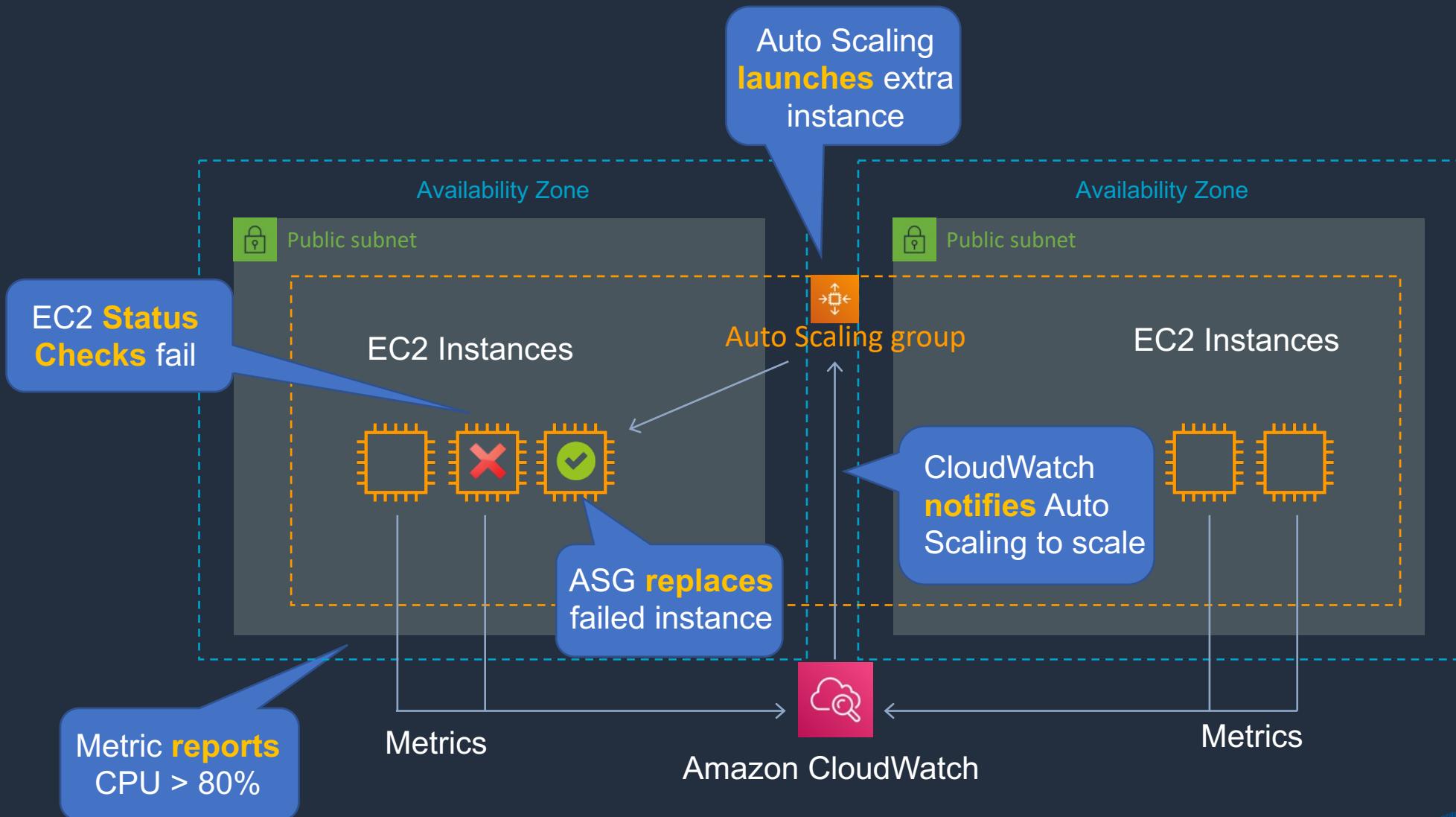


# Advanced Auto Scaling



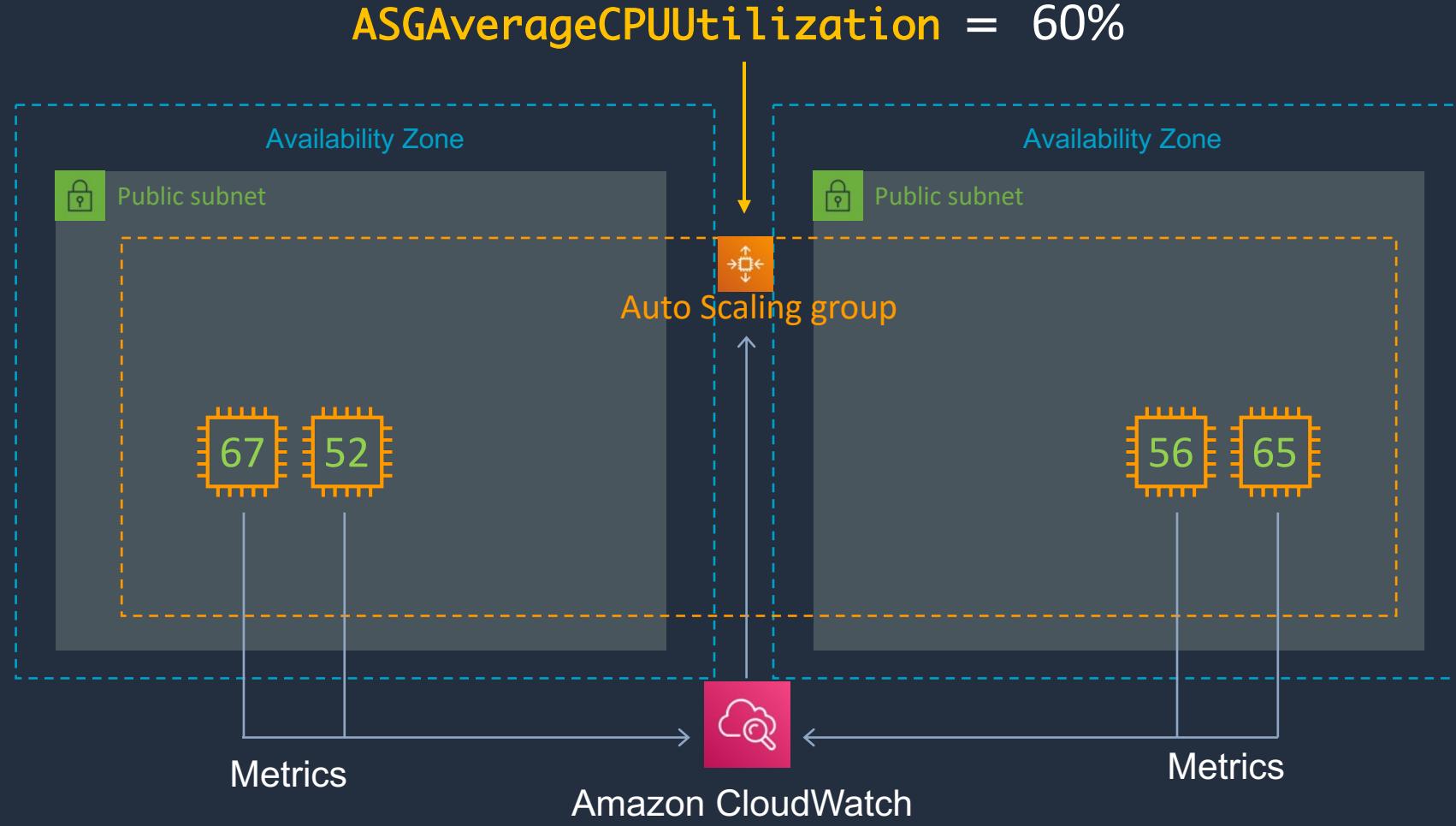


# Refresher: Auto Scaling Basics





# Dynamic Scaling – Target Tracking



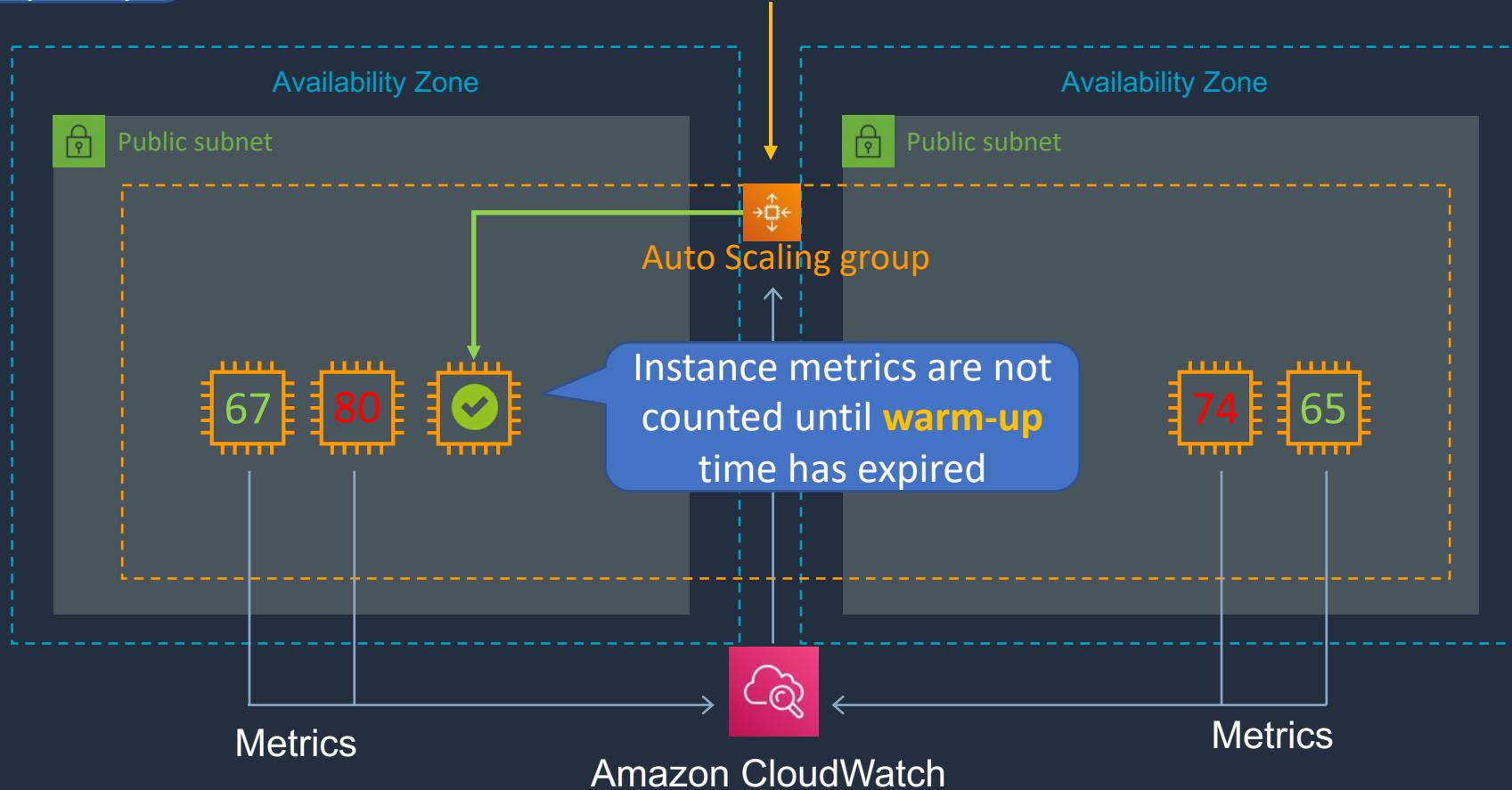
**Average CPU = 60%**



# Dynamic Scaling – Target Tracking

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

**ASGAverageCPUUtilization = 60%**



**Average CPU = 71.5%**



# Dynamic Scaling – Target Tracking

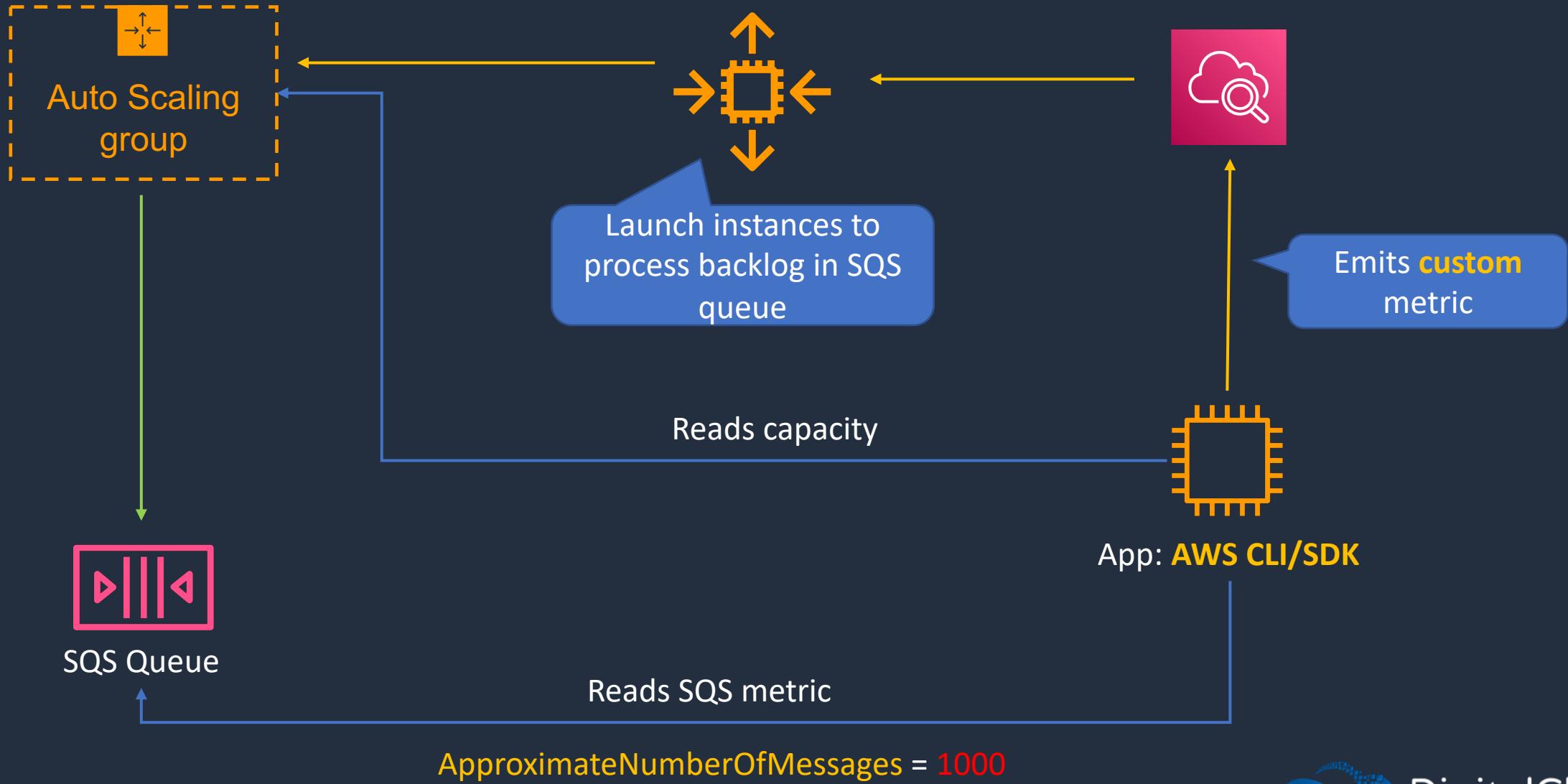
---

## Metrics:

- **ASGAverageCPUUtilization**—Average CPU utilization of the Auto Scaling group
- **ASGAverageNetworkIn**—Average number of bytes received on all network interfaces by the Auto Scaling group
- **ASGAverageNetworkOut**—Average number of bytes sent out on all network interfaces by the Auto Scaling group
- **ALBRequestCountPerTarget**—Number of requests completed per target in an Application Load Balancer target group

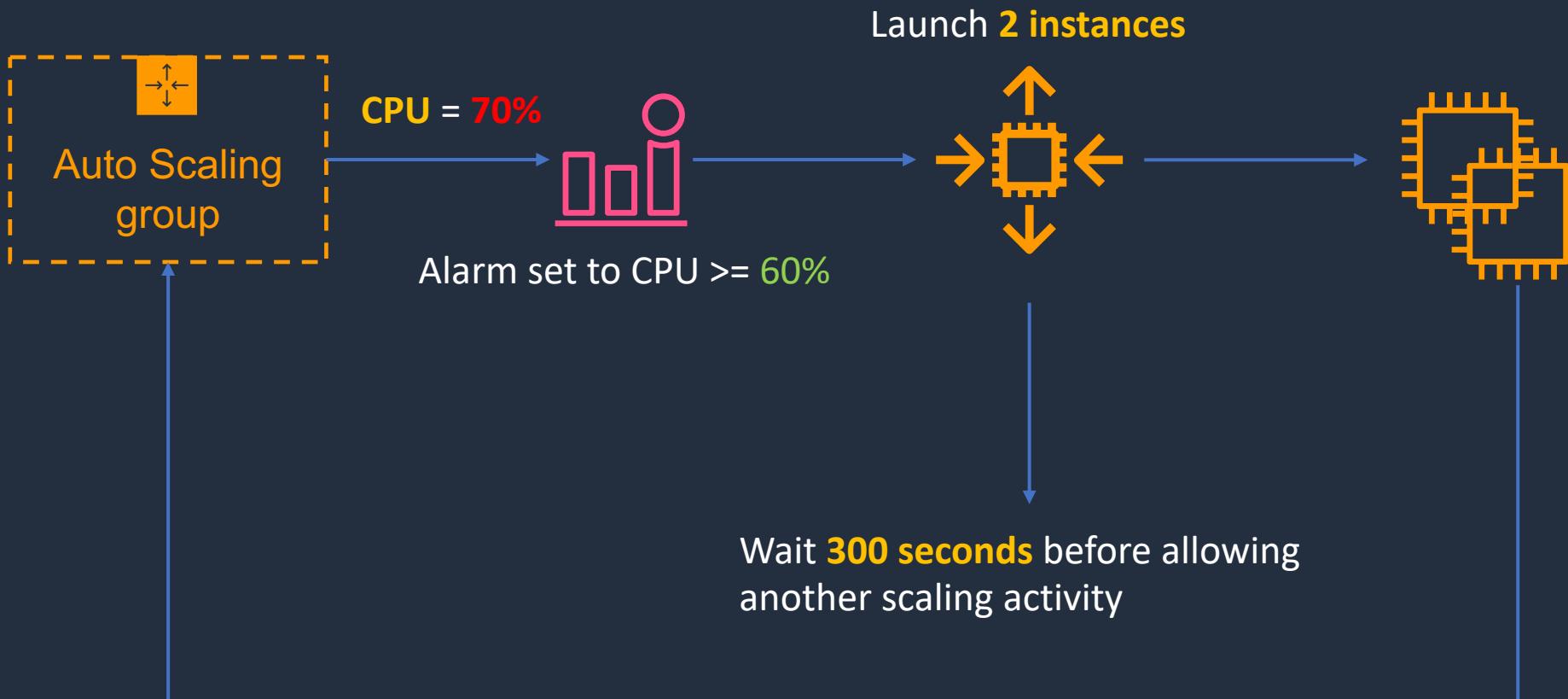


# Dynamic Scaling – Target Tracking with SQS



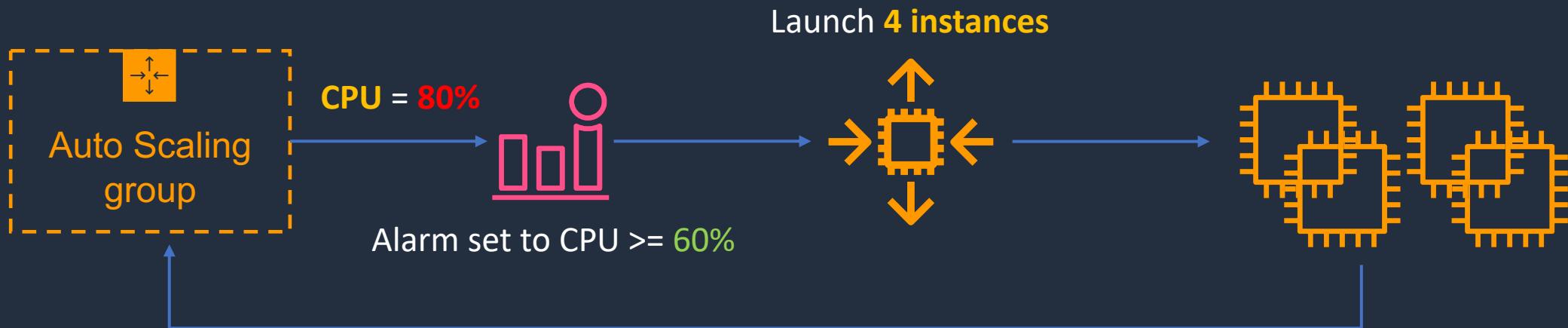
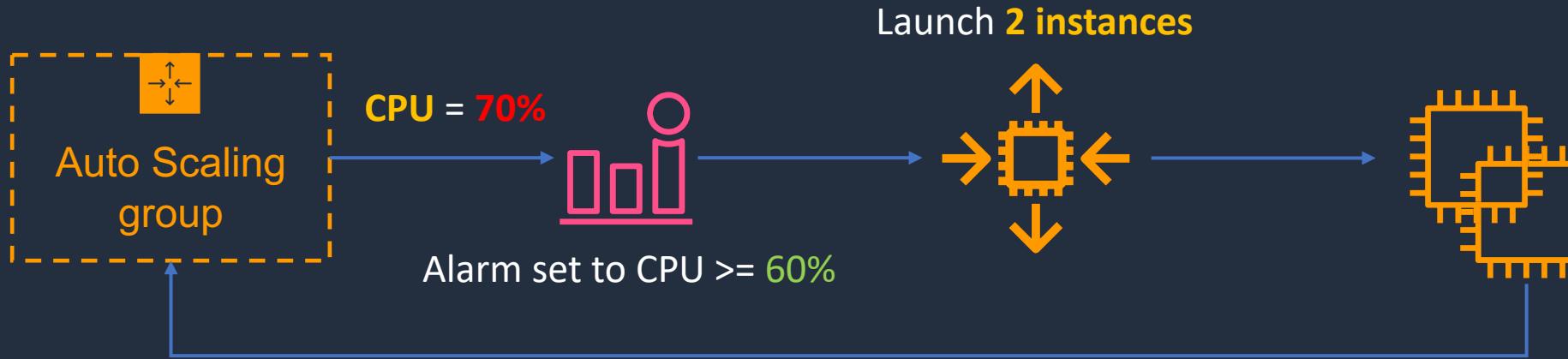


# Dynamic Scaling – Simple Scaling



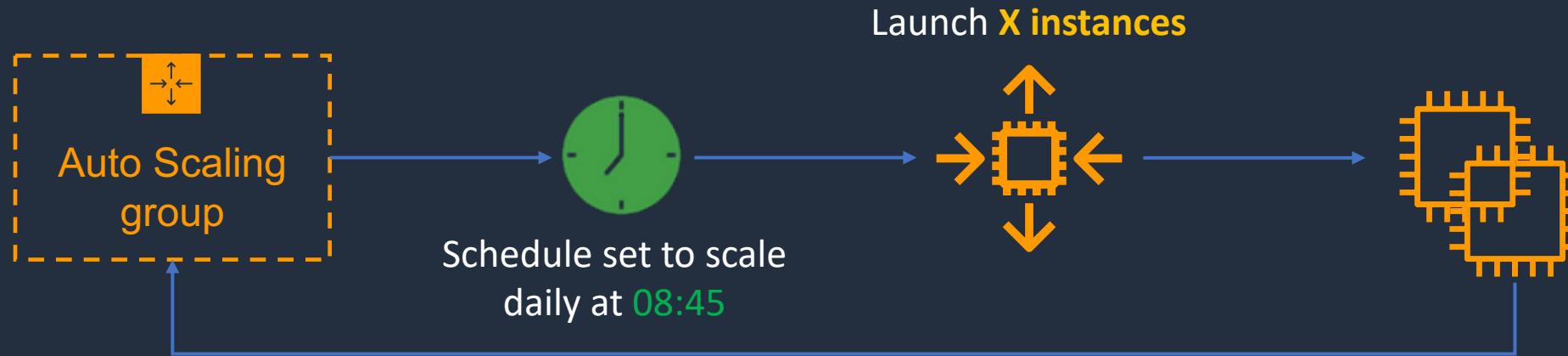


# Dynamic Scaling – Step Scaling





# Scheduled Scaling



Attempts to maintain **desired** count ->

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

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

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



# Scaling Processes

---

- **Launch** – Adds a new EC2 instance to an Auto Scaling group
- **Terminate** – Removes an EC2 instance from the group
- **AddToLoadBalancer** – Adds instances to an attached ELB or TG
- **AlarmNotification** – Accepts notifications from CloudWatch alarms that are associated with the group's scaling policies
- **AZRebalance** – Balances the number of EC2 instances in the group evenly across all of the specified Availability Zones
- **HealthCheck** – Checks the health of the instances and marks an instance as unhealthy if Amazon EC2 or Elastic Load Balancing tells Amazon EC2 Auto Scaling that the instance is unhealthy
- **ReplaceUnhealthy** – Terminates instances that are marked as unhealthy and then creates new instances to replace them
- **ScheduledActions** – Performs scheduled scaling actions



# Additional Scaling Settings

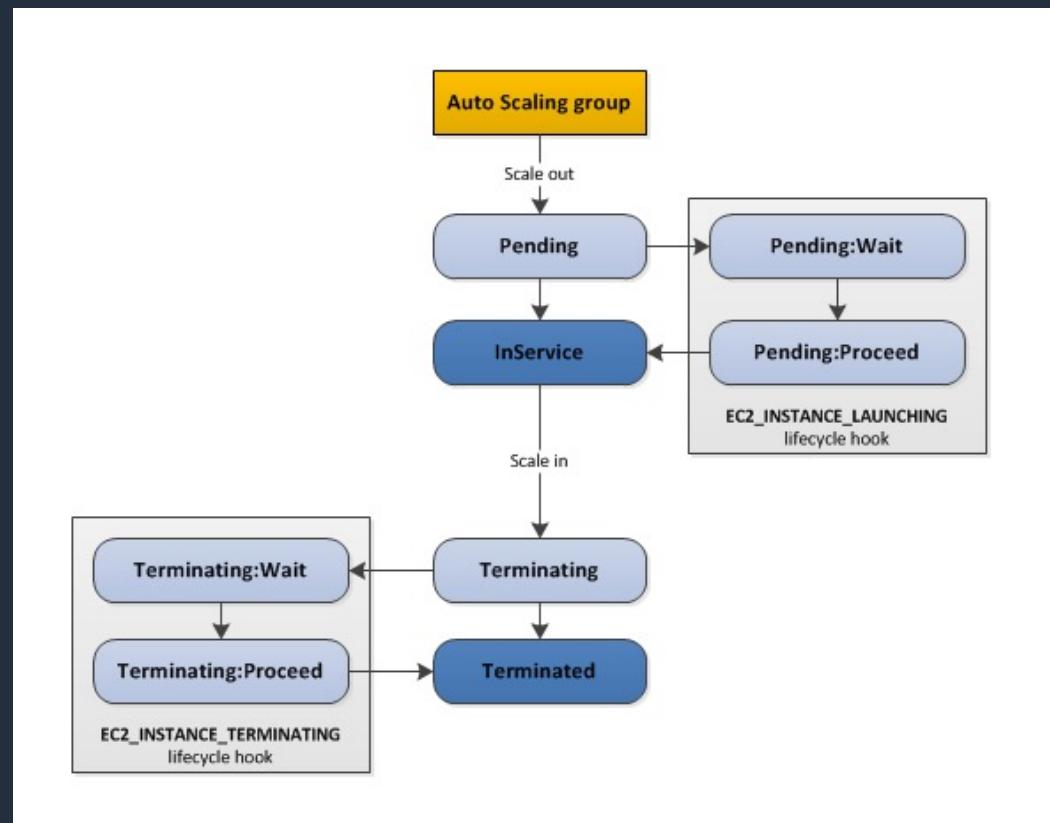
---

- **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

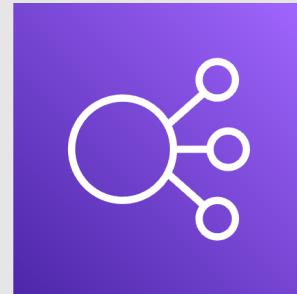
- **Lifecycle Hooks** – Used to perform custom actions by pausing instances as the ASG launches or terminates them.



# Advanced Auto Scaling

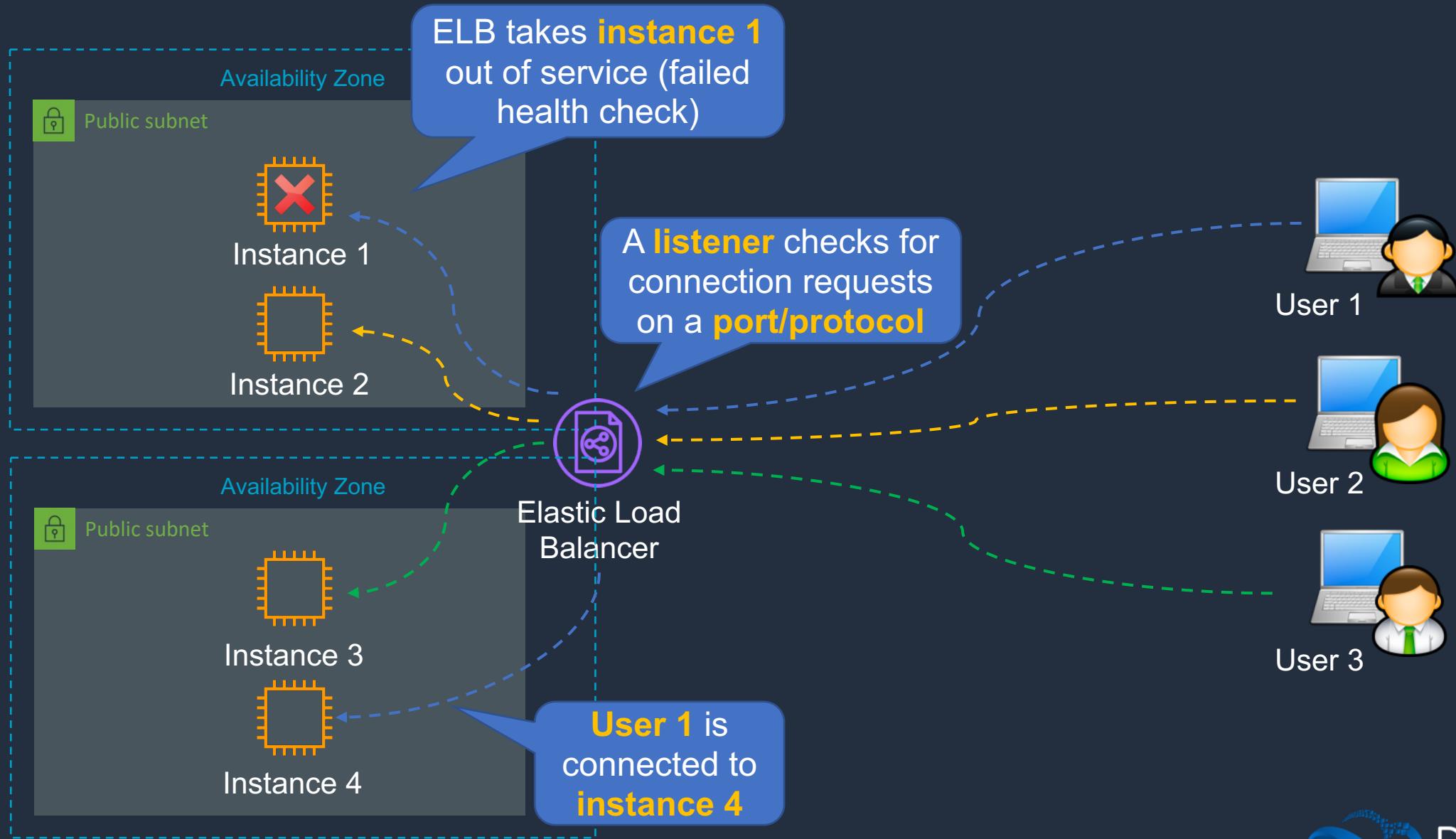


# Types of Elastic Load Balancer (ELB)



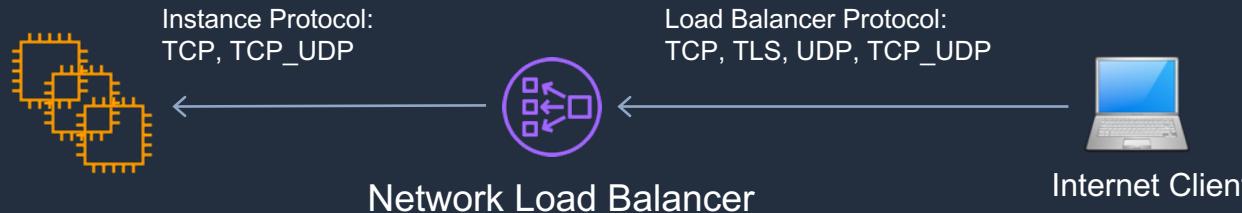


# Refresher: ELB Basics





# 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

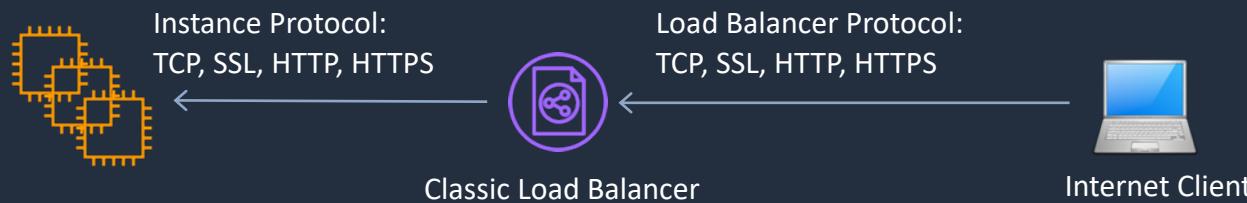
## Network Load Balancer

- 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



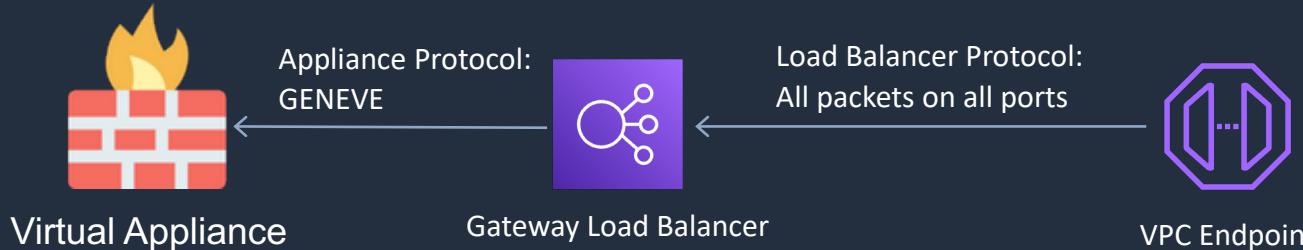
# Types of Elastic Load Balancer (ELB)

Old and **shouldn't** be  
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

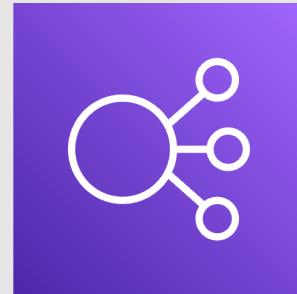


## Gateway Load Balancer

- 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 **not** yet  
on the exam

# Routing with ALB and NLB





# Application Load Balancer (ALB)

Application Load  
Balancer (ALB)

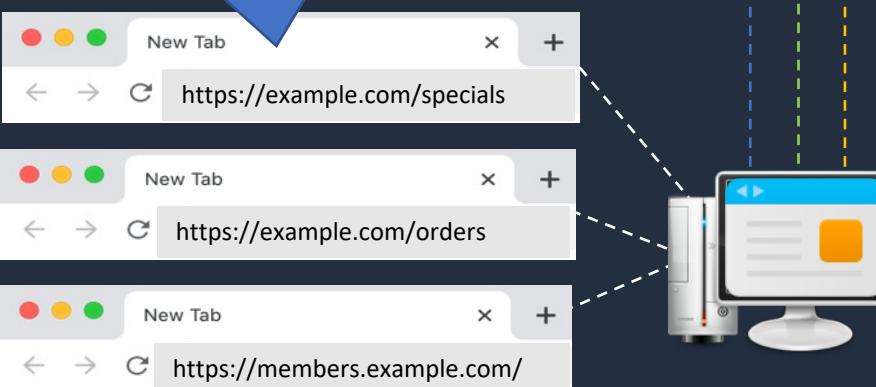
Requests can also be routed based  
on the **host** field in the **HTTP header**

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

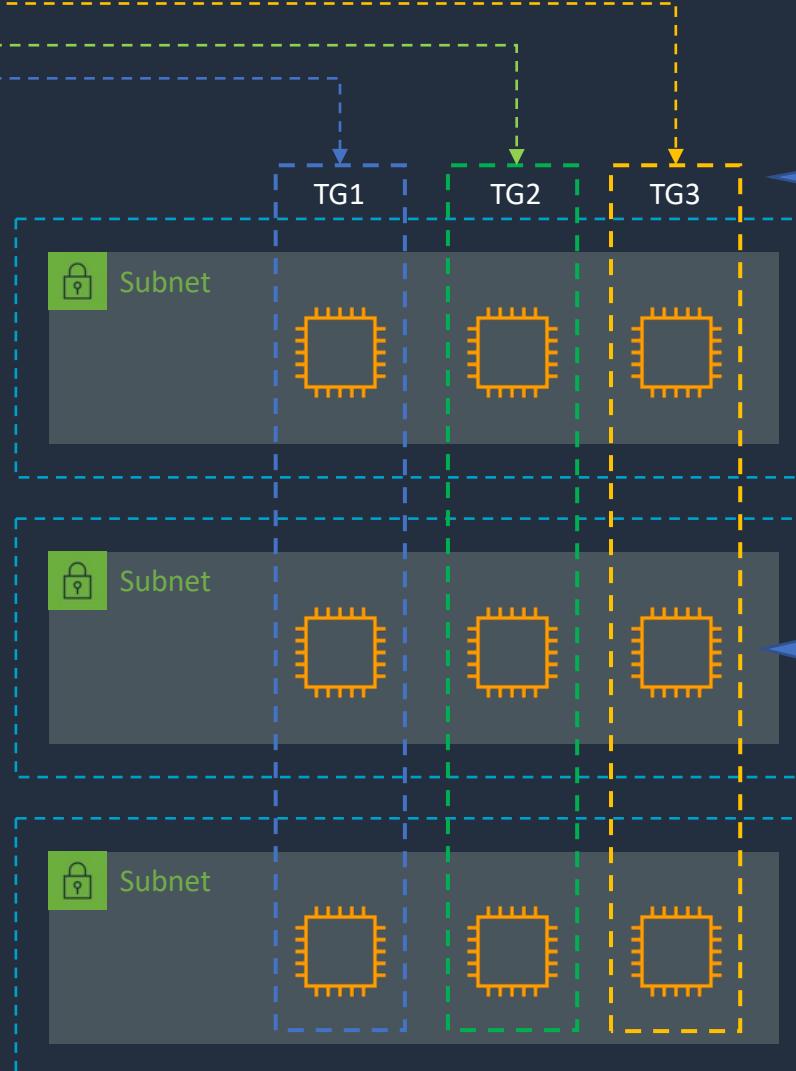


Requests can be  
routed based on  
the **path** in the **URL**

**Path-based**  
routing



**Host-based routing**

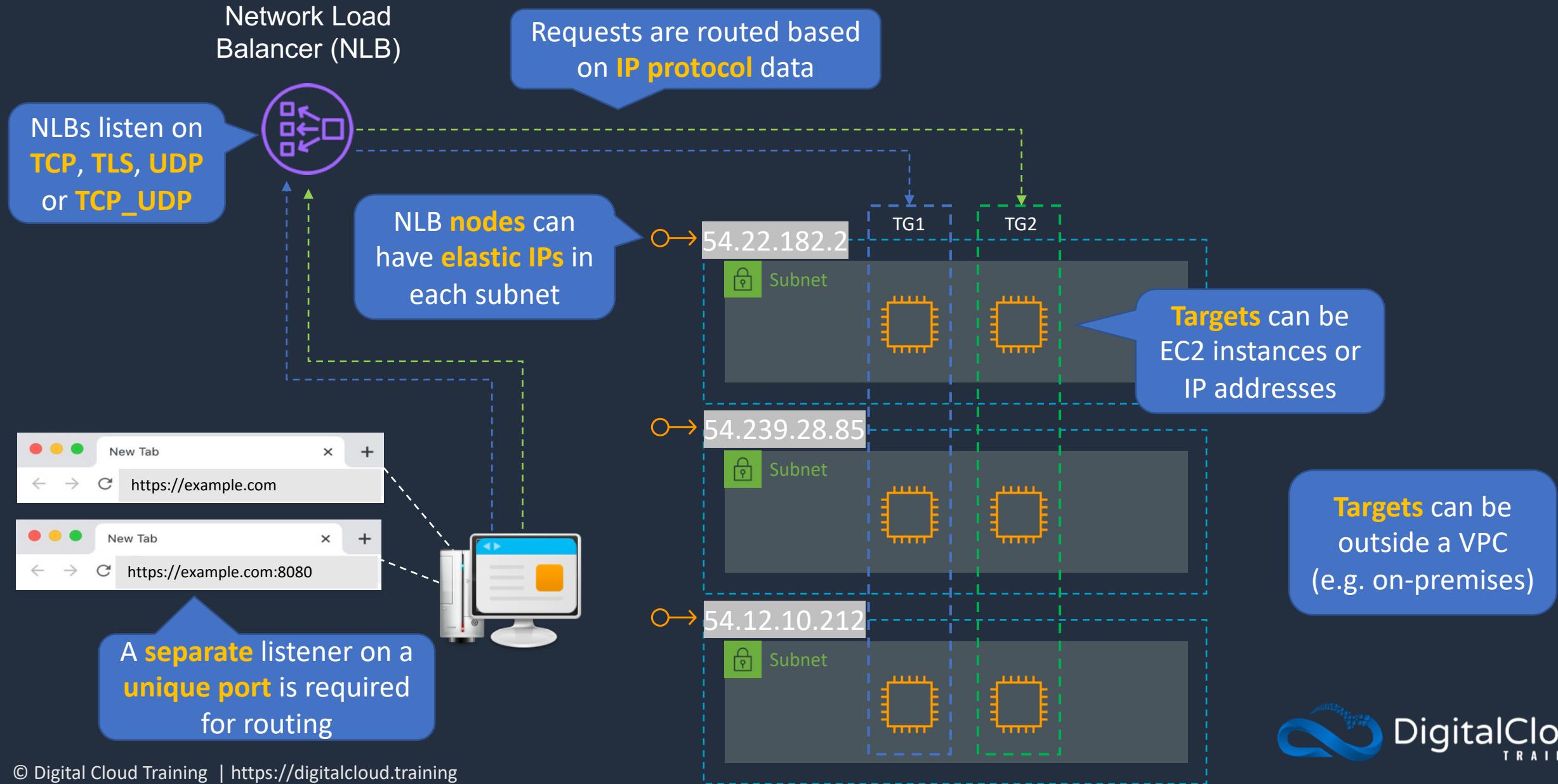


**Target groups** are used  
to route requests to  
registered targets

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



# Network Load Balancer (NLB)

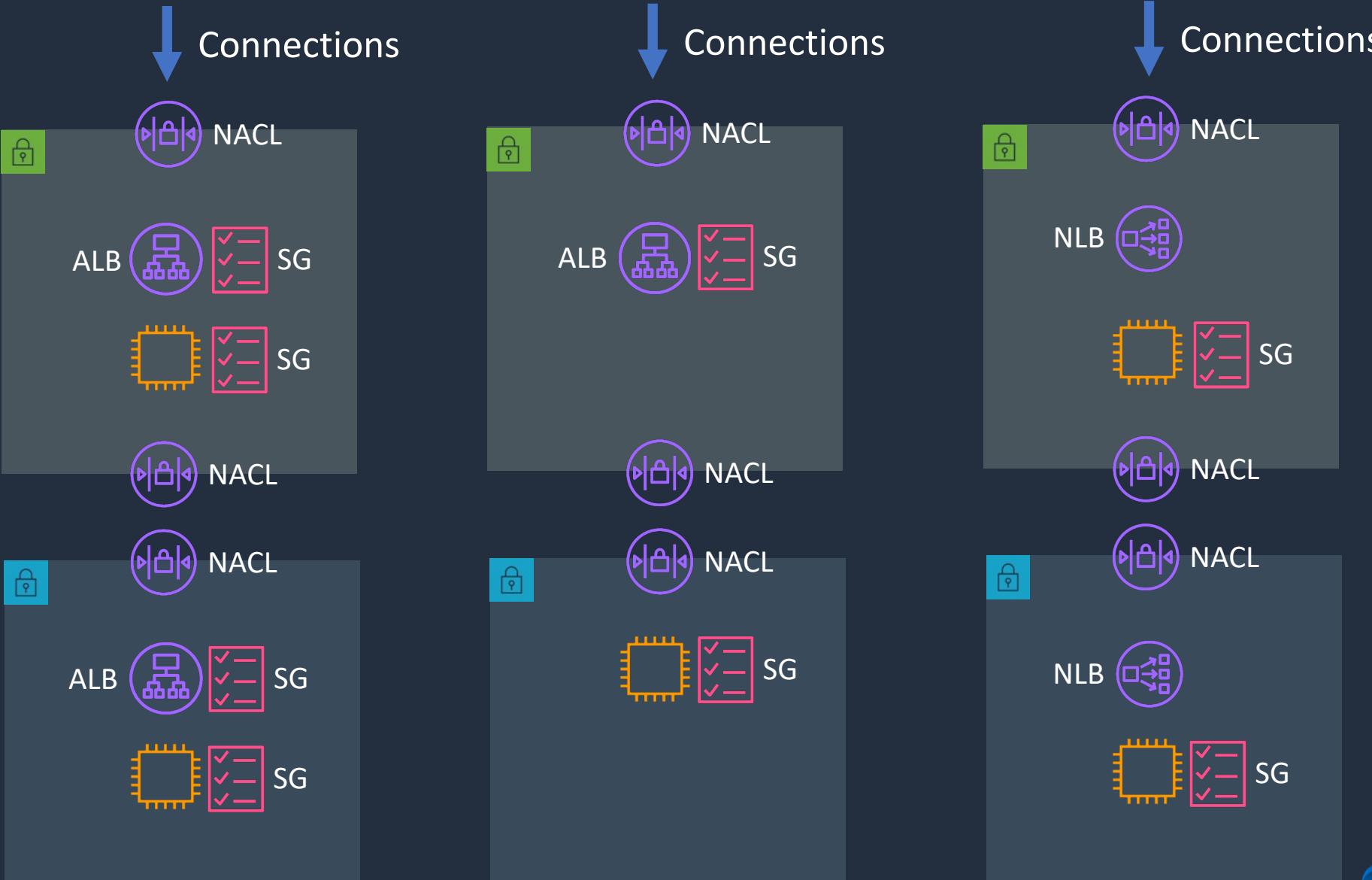


# ALB and NLB Access Control and SSL/TLS





# Access Control with ALB and NLB





# What's the Source IP Address the App sees?

Note: X-forwarded-for can be used with ALB to capture client IP

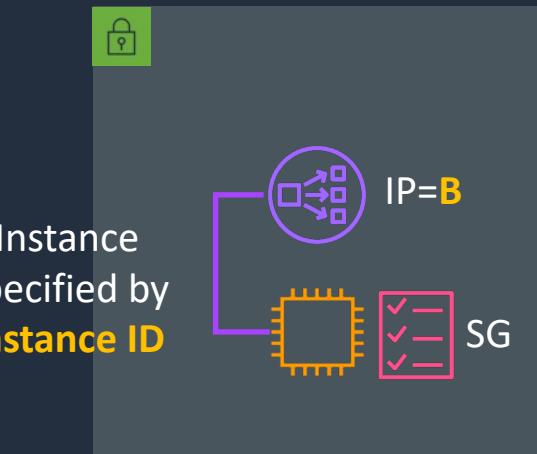
AWS ALB



CLB and ALB use  
private IP of their ENIs  
as source address

Source	Protocol	Port
IP=B	TCP	80

AWS NLB



AWS NLB



Applicable to TCP and TLS – for UDP and TCP\_UDP should be IP=A

When using an NLB with a VPC  
Endpoint or AWS GA source IPs are  
private IPs of NLB nodes

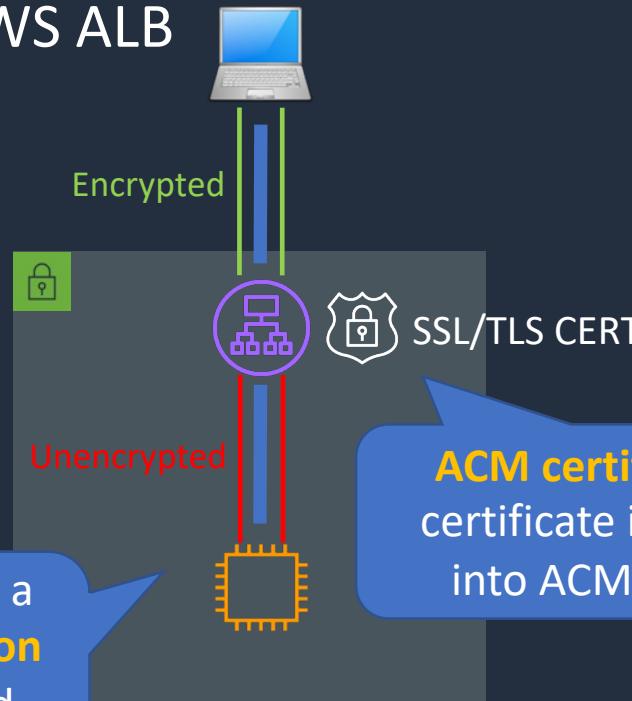
Source	Protocol	Port
IP=A	TCP	80

Source	Protocol	Port
IP=B	TCP	80



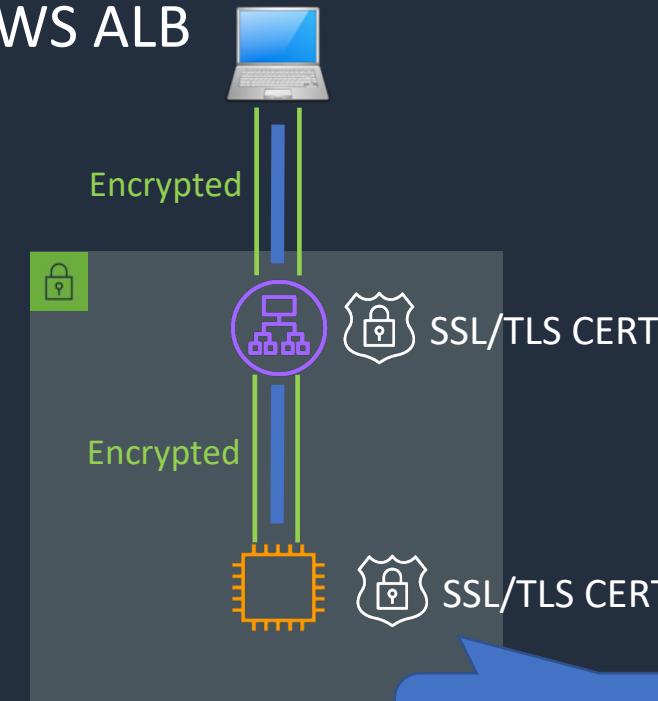
# SSL/TLS Termination

AWS ALB



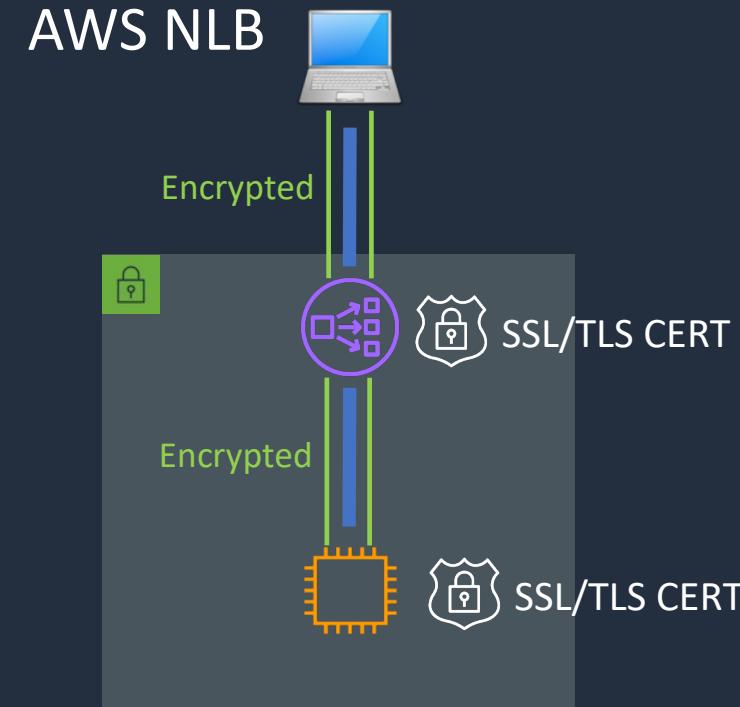
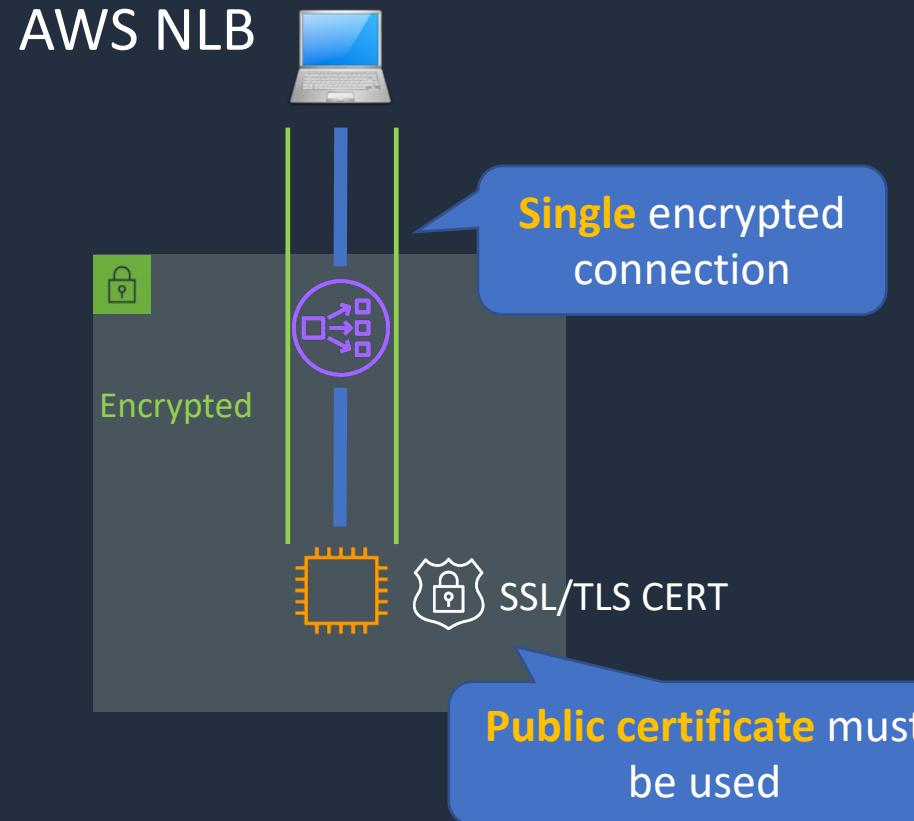
With a **L7 ELB** a  
**new connection**  
is established  
with the instance

AWS ALB





# SSL/TLS Termination



# Register Domain with Route 53

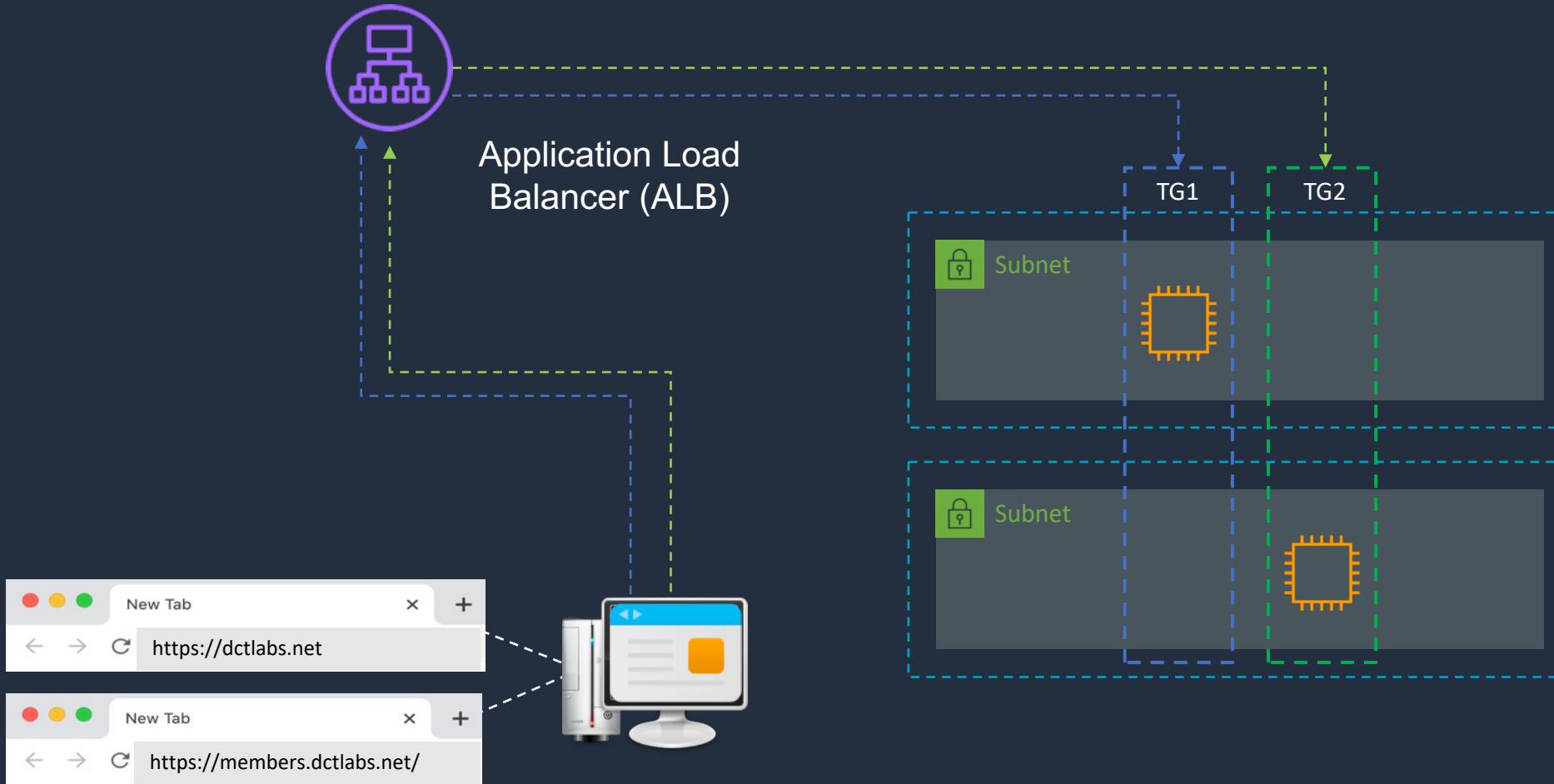


# Request Routing with ALB





# Application Load Balancer (ALB)

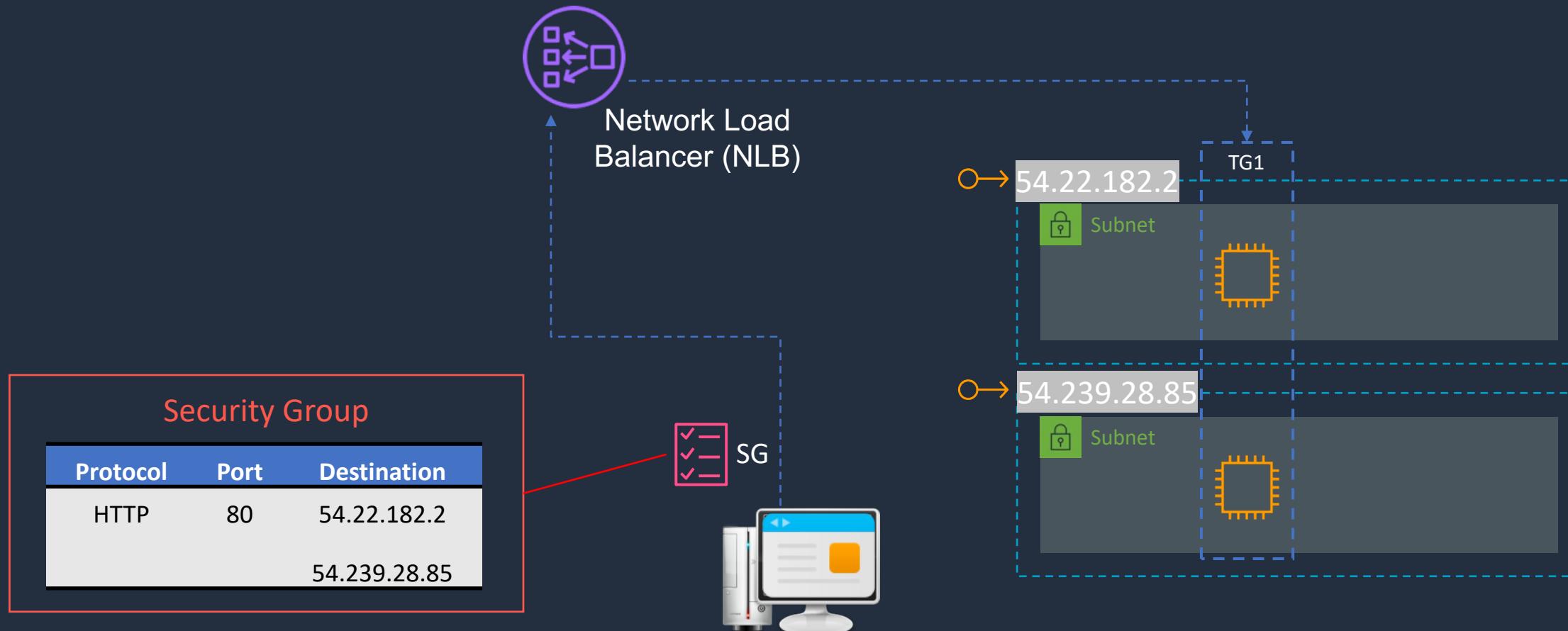


# NLB Static IPs and Whitelisting

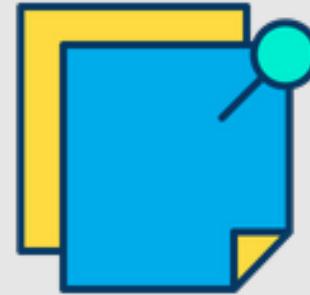




# Network Load Balancer (NLB)

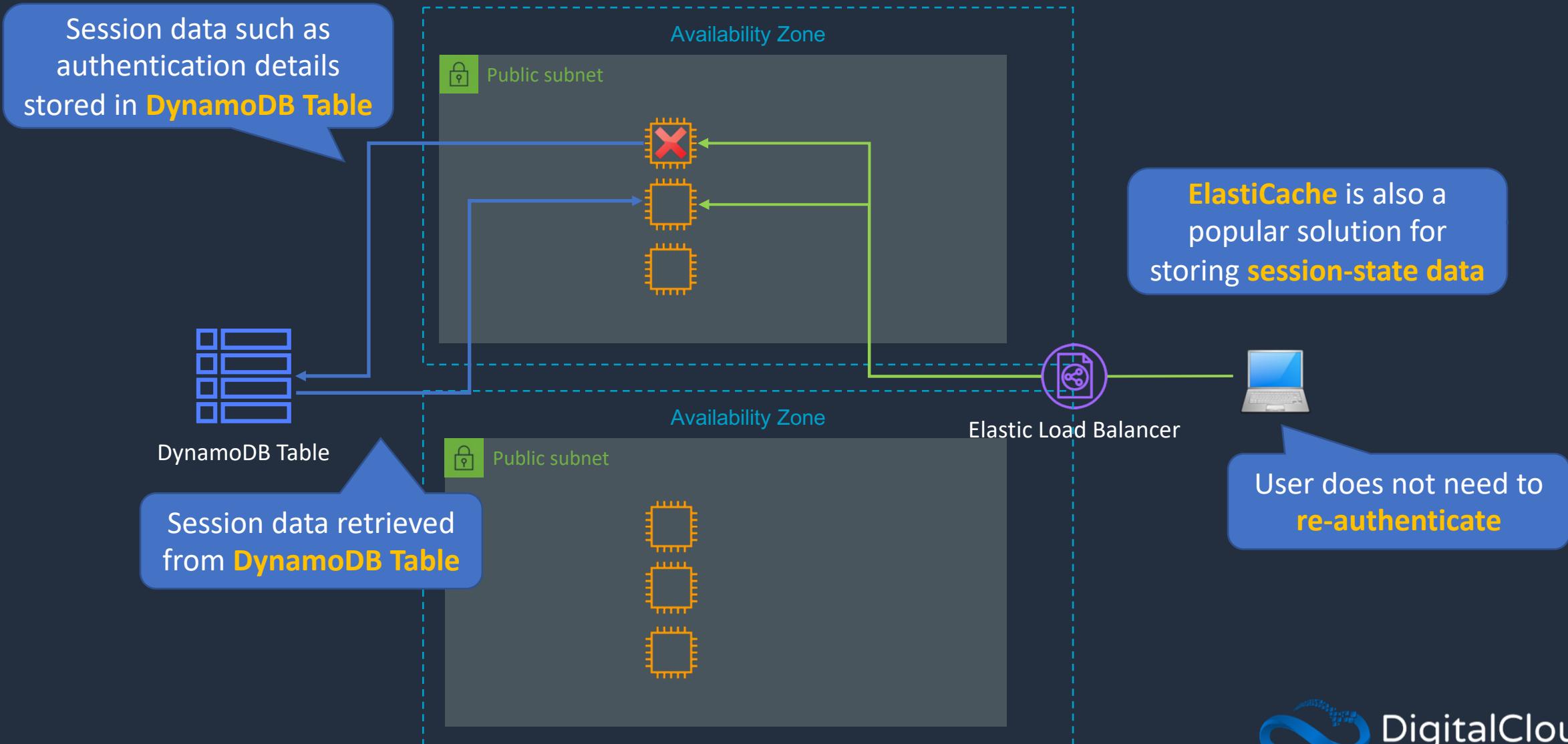


# Session State and Session Stickiness



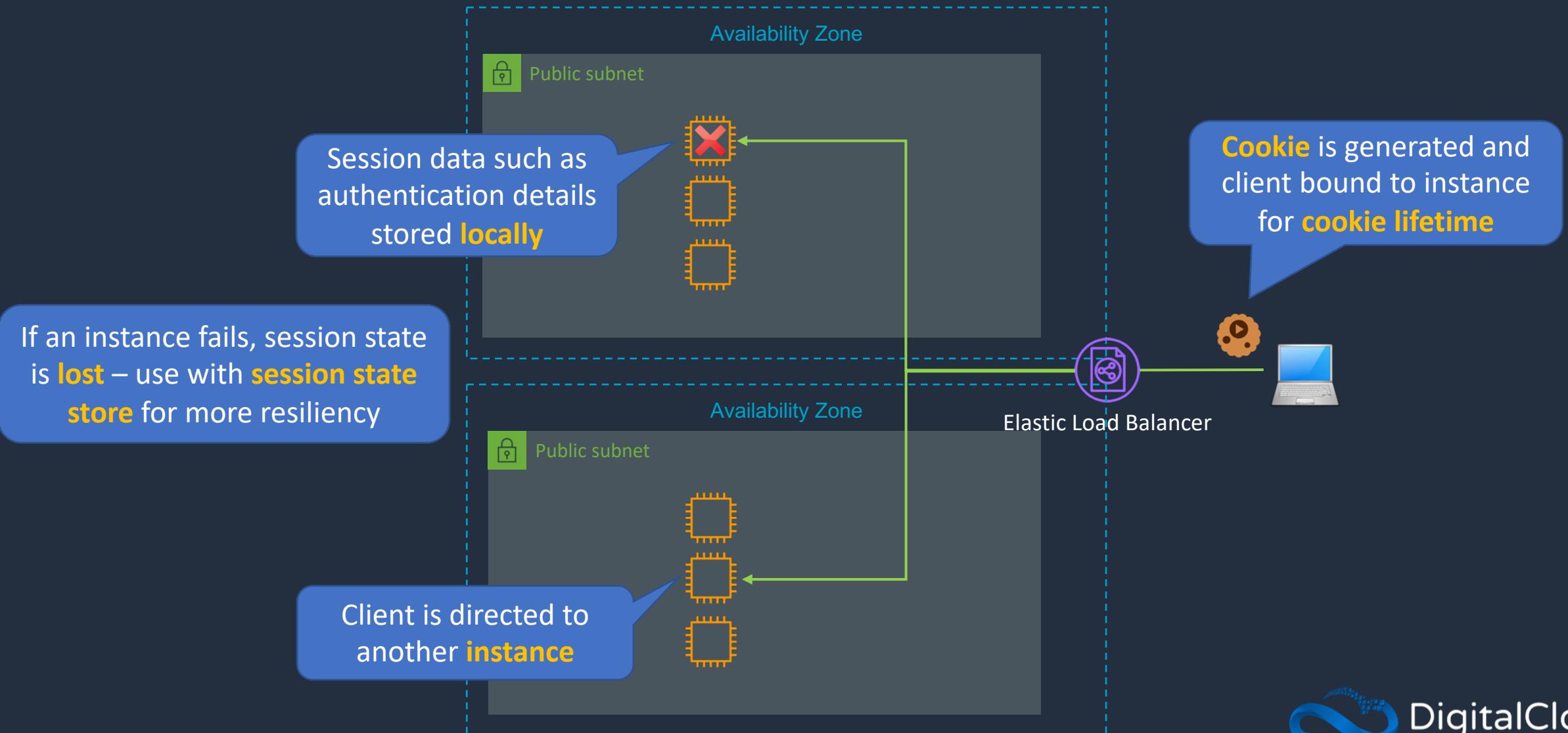


# Storing Session State





# Sticky Sessions



# AWS Batch





# AWS Batch



Launch a **Batch Job**



Job **Definition**

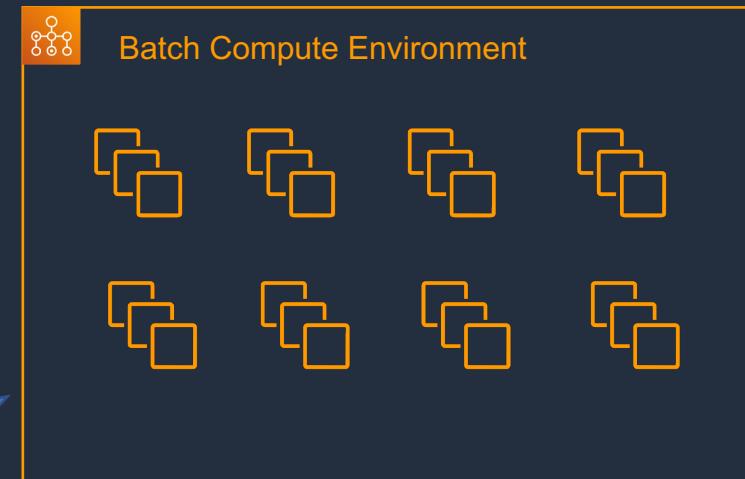


Job **Queue**

A job is submitted to a **queue** until **scheduled** onto a compute environment

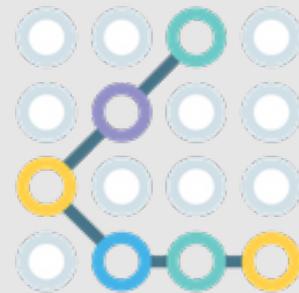
A job is a unit of work such as a **shell script**, **executable** or **Docker container image**

Batch **launches**, **manages**, and **terminates** resources as required (EC2 and ECS/Fargate)



Managed or **unmanaged** resources used to run the job

# Architecture Patterns - Compute





# Architecture Patterns - Compute

## Requirement

High availability and elastic scalability for web servers

## Solution

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

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

Use a Network Load Balancer with a UDP listener

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

Use an NLB and create static IP addresses in each AZ



# Architecture Patterns - Compute

## 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 - Compute

## Requirement

Need to run a large batch computing job at the lowest cost. Must be managed. Nodes can pick up where others left off in case of interruption

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

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

## Solution

Use a managed AWS Batch job and use EC2 Spot 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 - Compute

---

---

## Requirement

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

## Solution

Create an AMI that includes the application dependencies and code

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

Enable Sticky session for the target group or alternatively use a session store such as DynamoDB

# SECTION 8

## AWS Storage Services

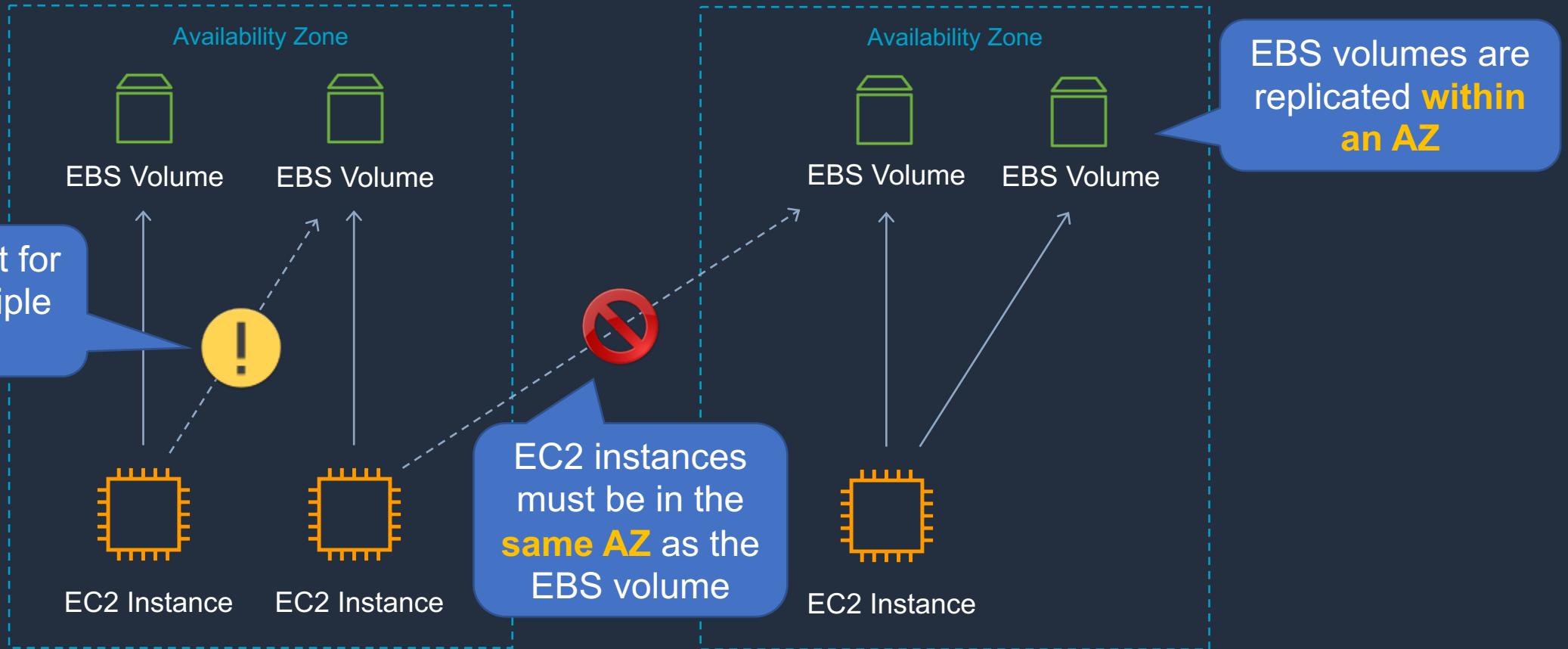
# Amazon EBS Deployment and Volume Types



# Amazon EBS Deployment



## Amazon Elastic Block Store (EBS)





# Amazon EBS Multi-Attach



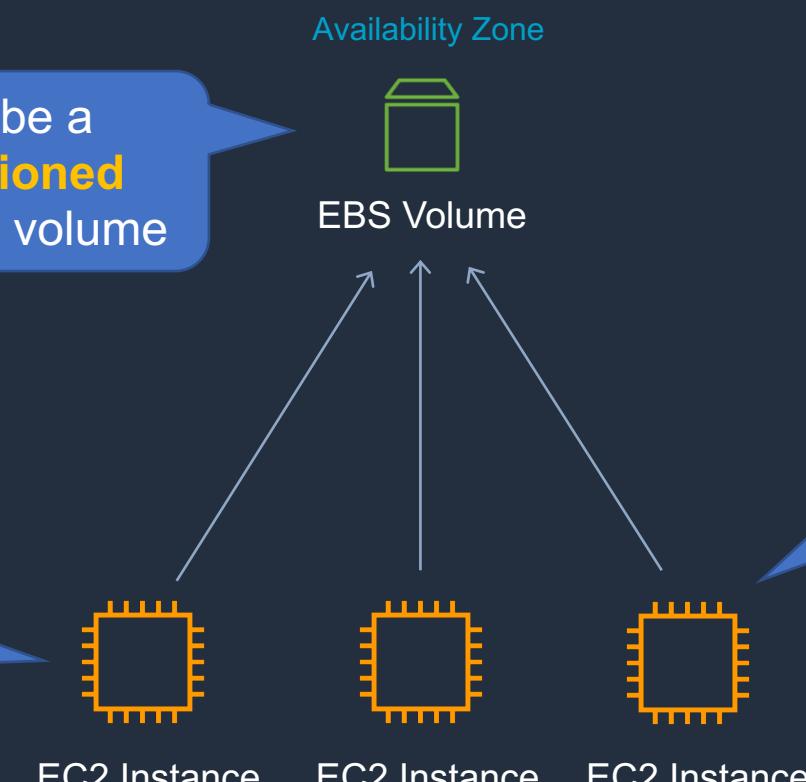
May not be on the exam yet

Must be within a **single AZ**

Must be a **Provisioned IOPS io1 volume**

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

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





# Amazon EBS SSD-Backed Volumes

New and **not** on  
the exam yet

New and **not** on  
the exam yet

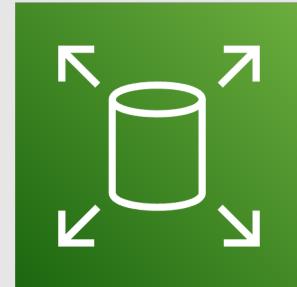
	General Purpose SSD		Provisioned IOPS SSD		
Volume type	gp3	gp2	io2 Block Express ‡	io2	io1
Durability	99.8% - 99.9% durability (0.1% - 0.2% annual failure rate)	99.8% - 99.9% durability (0.1% - 0.2% annual failure rate)	99.999% durability (0.001% annual failure rate)		99.8% - 99.9% durability (0.1% - 0.2% annual failure rate)
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

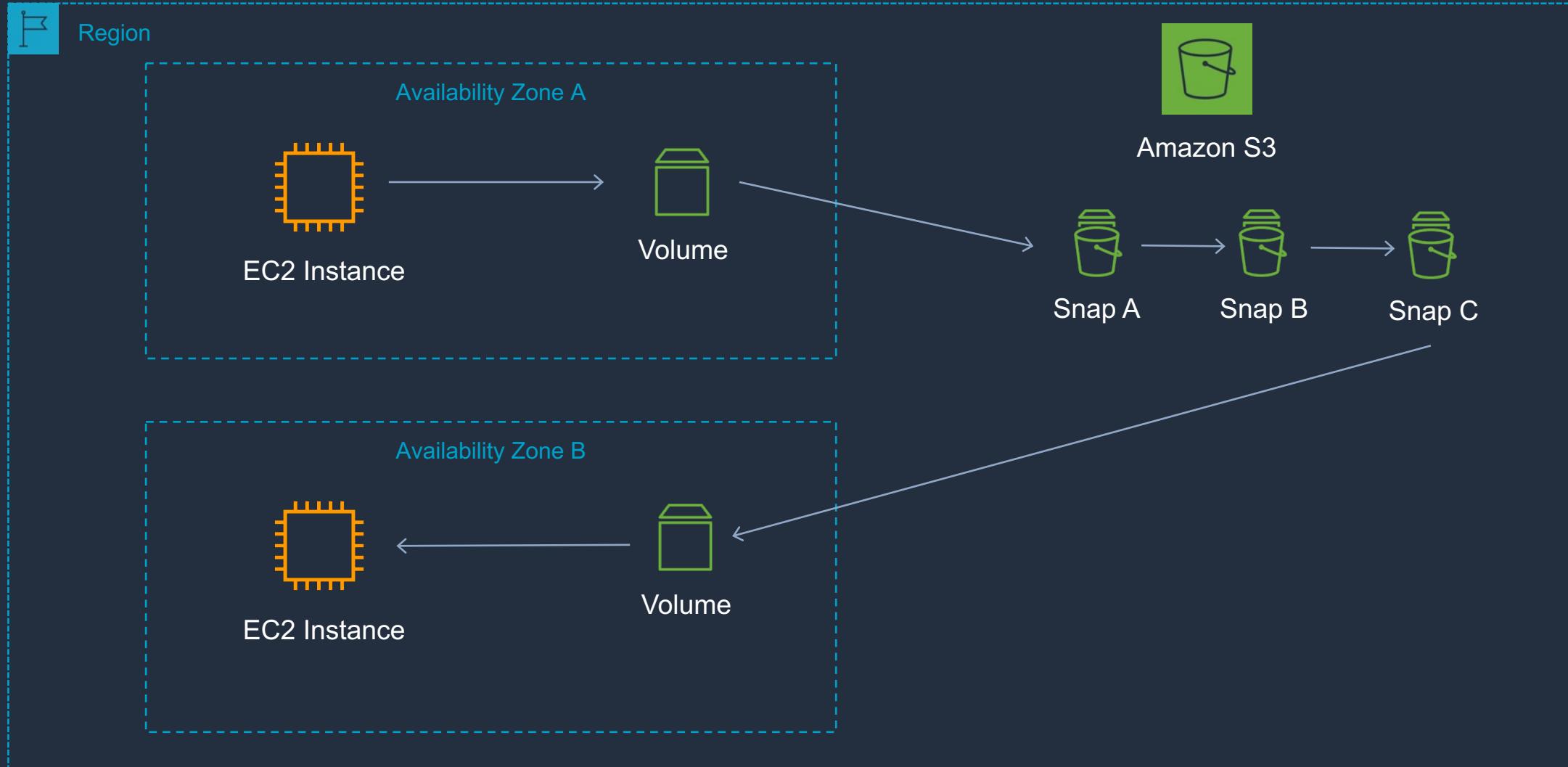
	Throughput Optimized HDD	Cold HDD
<b>Volume type</b>	st1	sc1
<b>Durability</b>	99.8% - 99.9% durability (0.1% - 0.2% annual failure rate)	99.8% - 99.9% durability (0.1% - 0.2% annual failure rate)
<b>Use cases</b>	<ul style="list-style-type: none"><li>• Big data</li><li>• Data warehouses</li><li>• Log 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 Copying, Sharing and Encryption



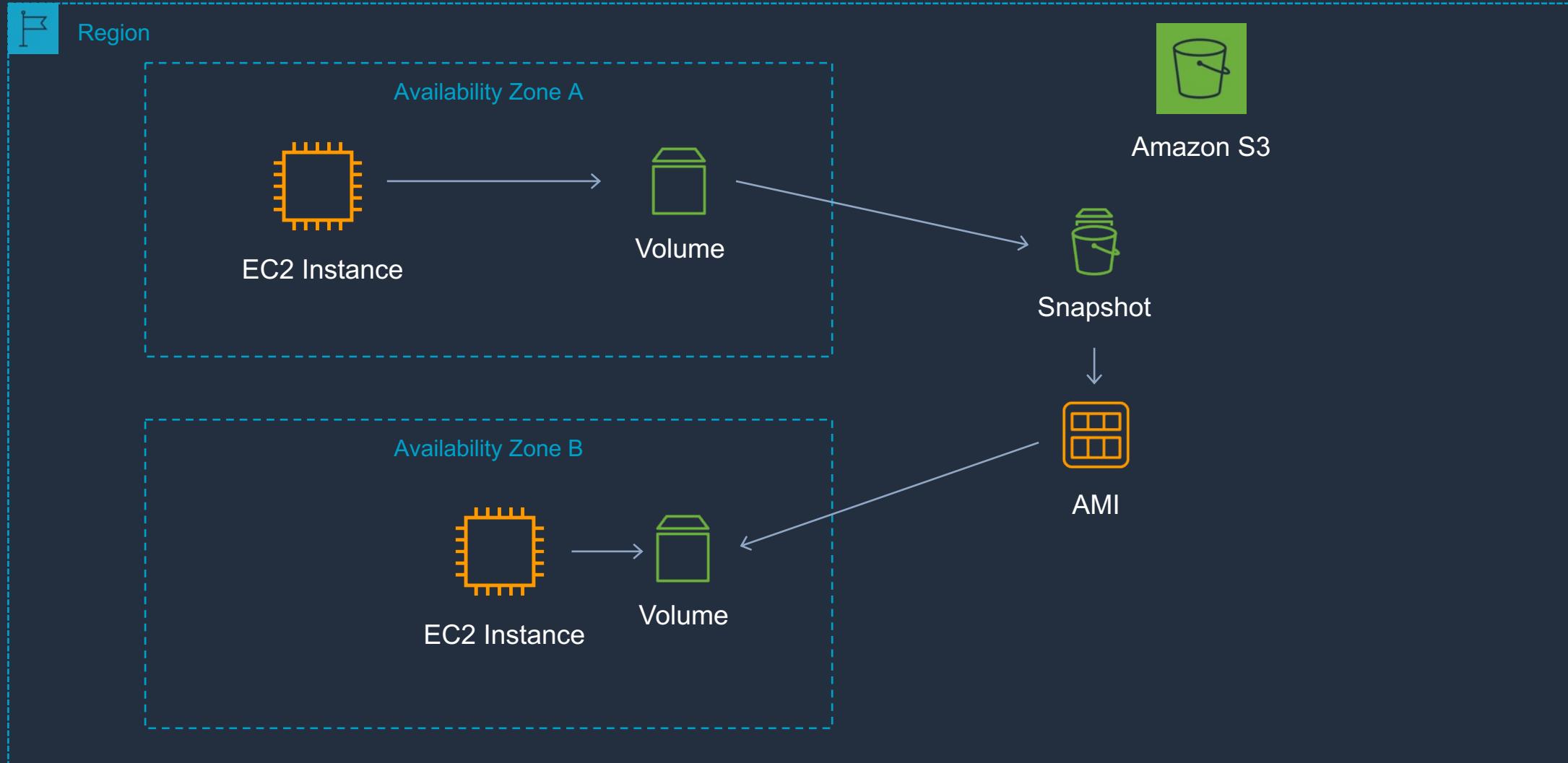


# Amazon EBS Copying, Sharing and Encryption



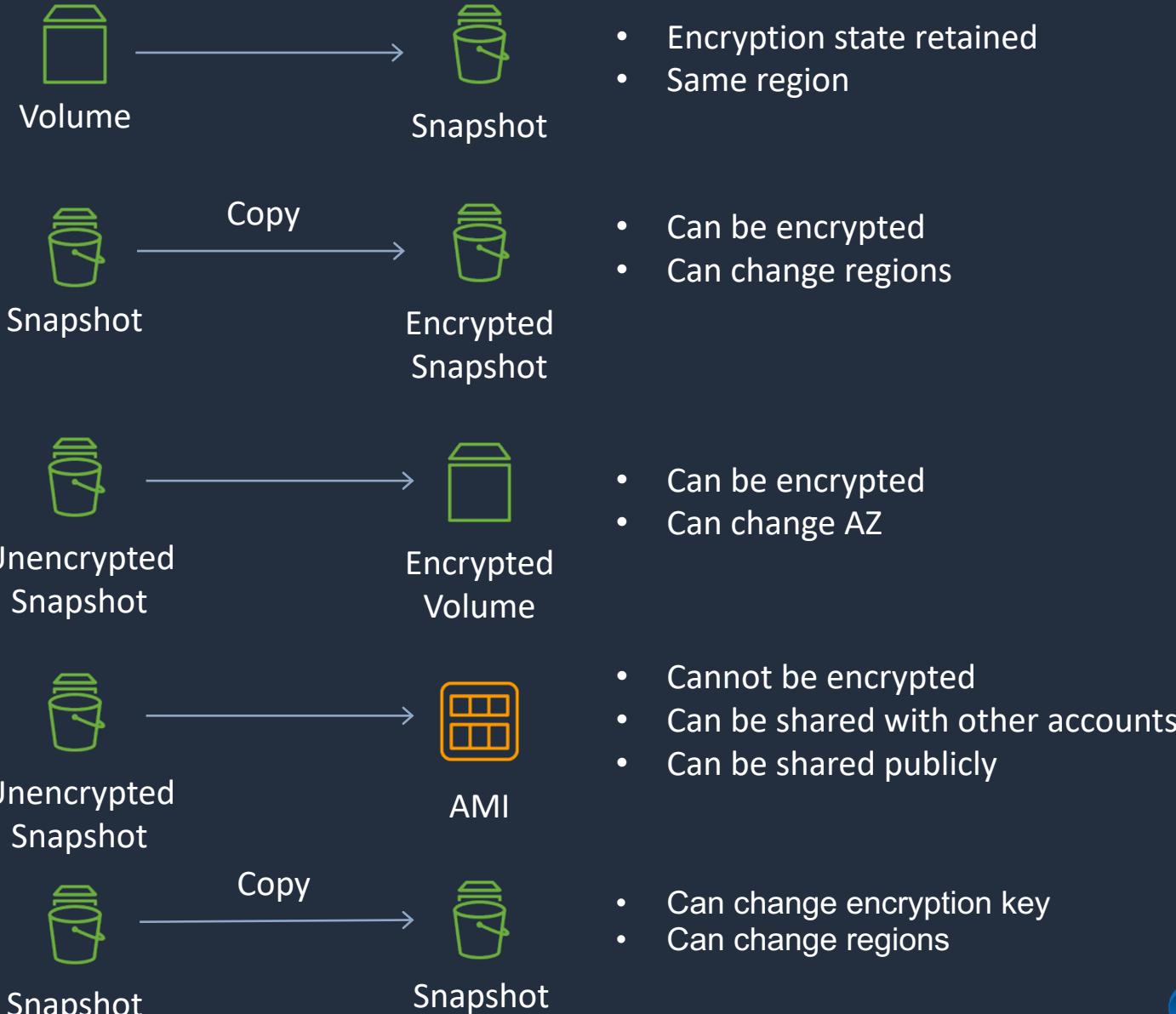


# Take Snapshot, Create AMI, Launch New Instance



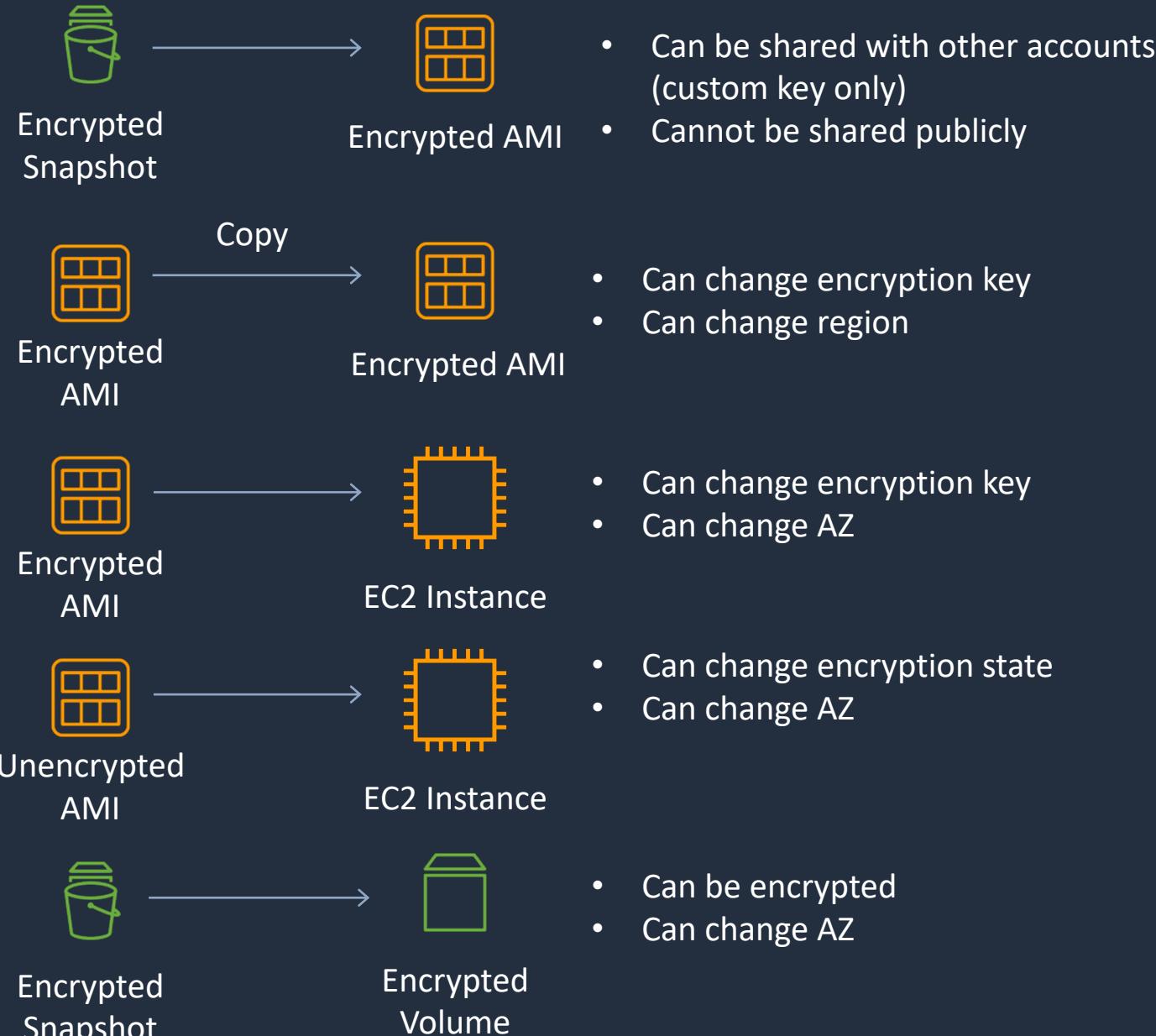


# Copying and Sharing AMIs and Snapshots

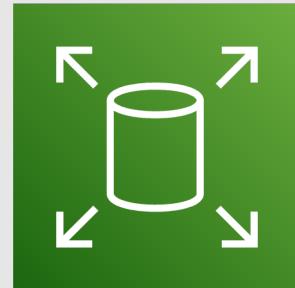




# Copying and Sharing AMIs and Snapshots

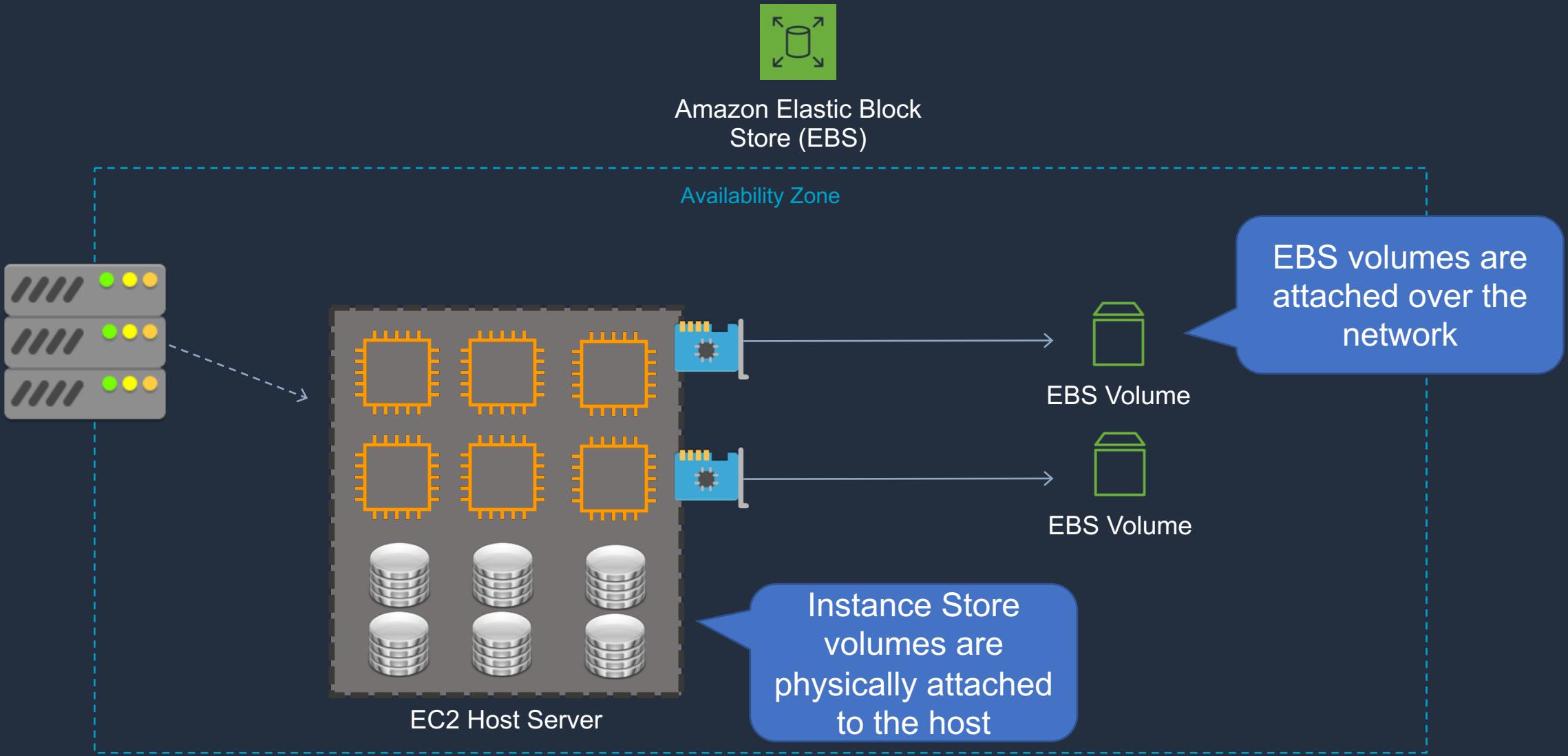


# EBS vs instance store





# EBS vs instance store





# EBS vs instance store

---

---

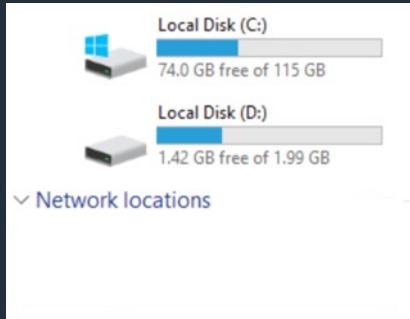
- Instance store volumes are high performance local disks that are physically attached to the host computer on which an EC2 instance runs
- Instance stores are ephemeral which means the data is lost when powered off (non-persistent)
- Instance stores are ideal for temporary storage of information that 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

# Amazon EFS Refresher





# Network Attached Storage



File Management

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

The **NAS** “shares”  
file systems over  
the network



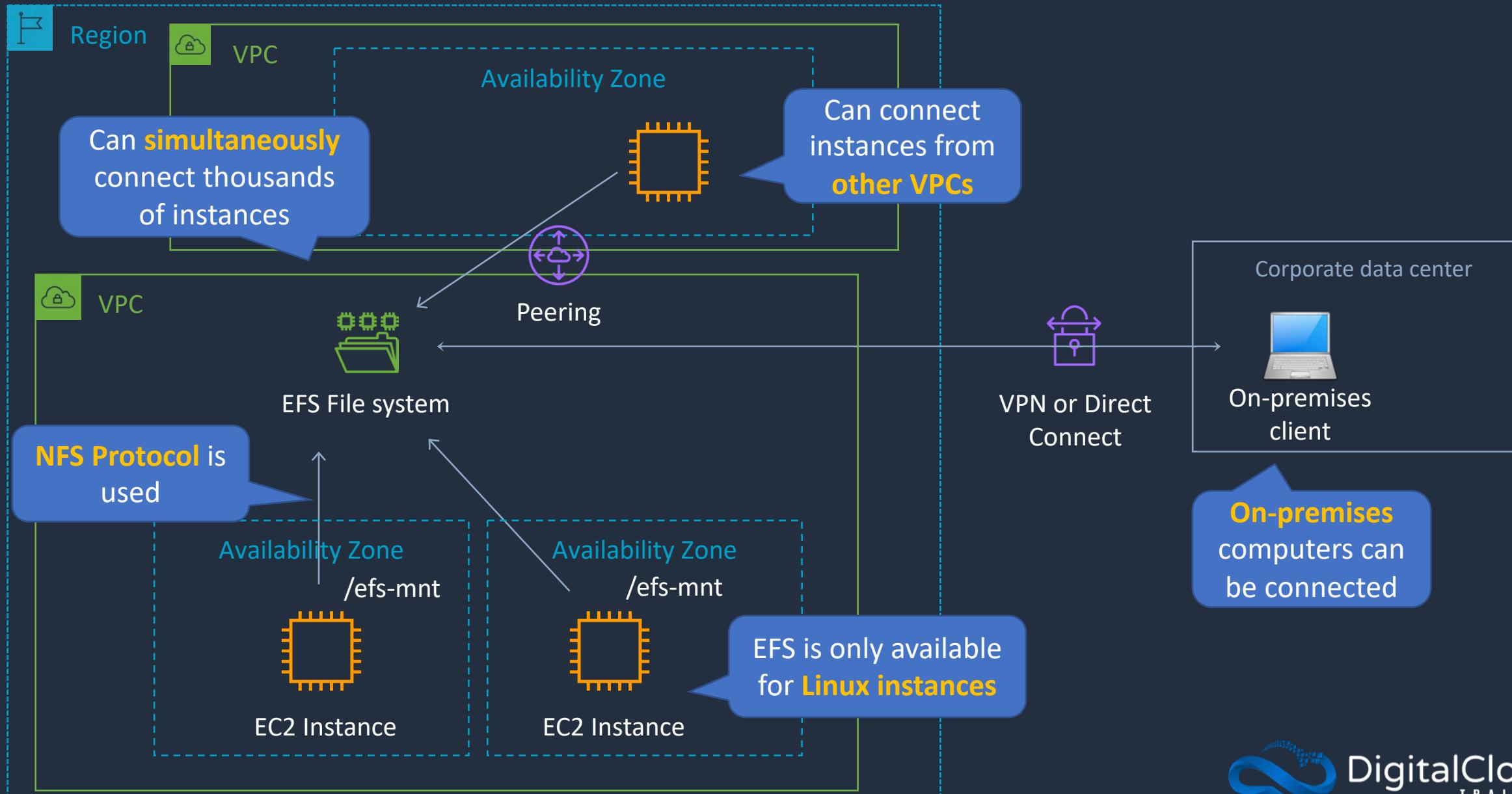
Network Attached  
Storage Server (NAS)



NAS devices are file-based storage systems

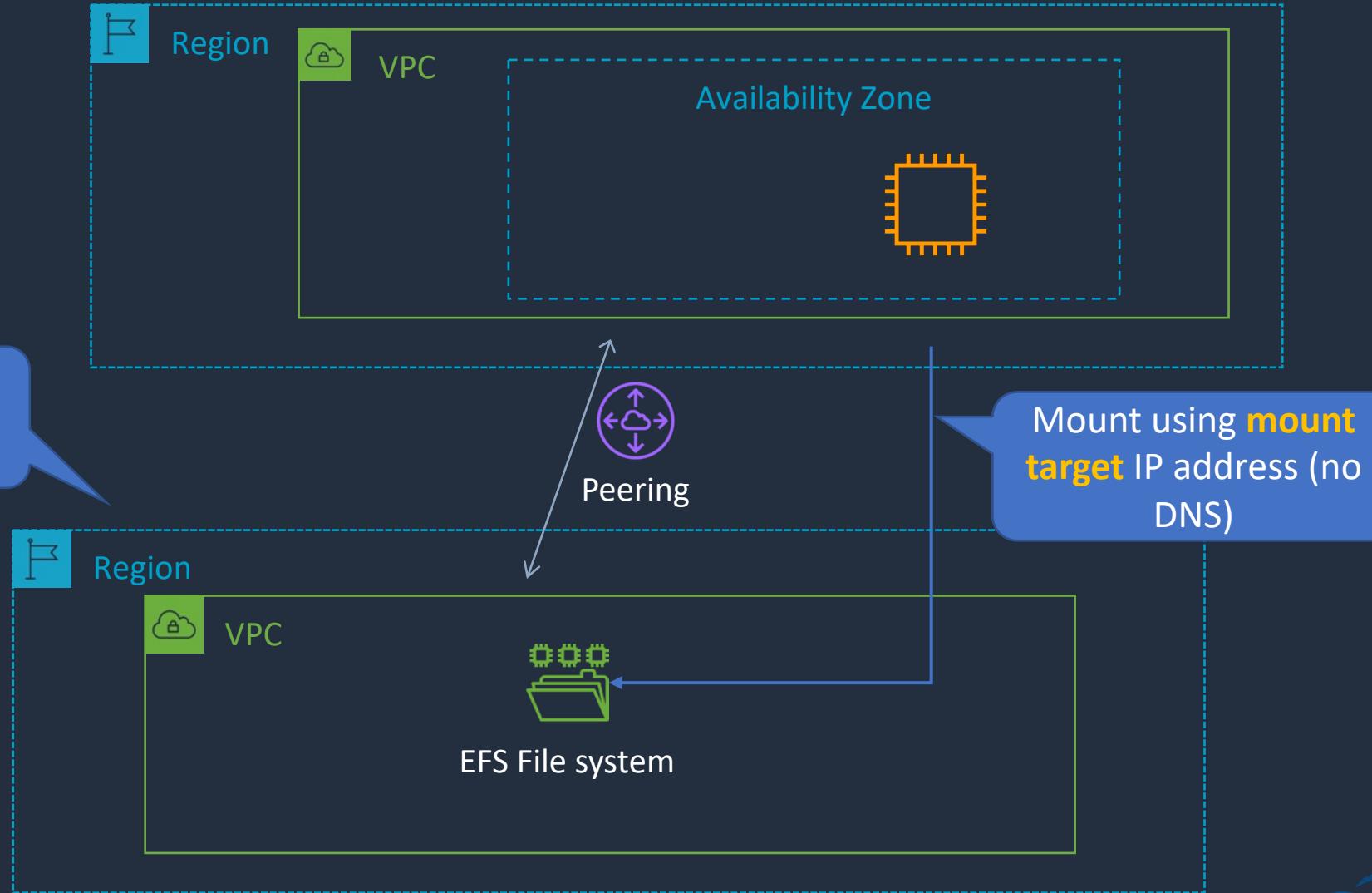


# Amazon Elastic File System (EFS) Overview





# Amazon Elastic File System (EFS) Overview



# Create and Mount EFS File System



# Amazon S3 Refresher





# Amazon Simple Storage Service (S3)



Bucket

A **bucket** is a container for objects

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

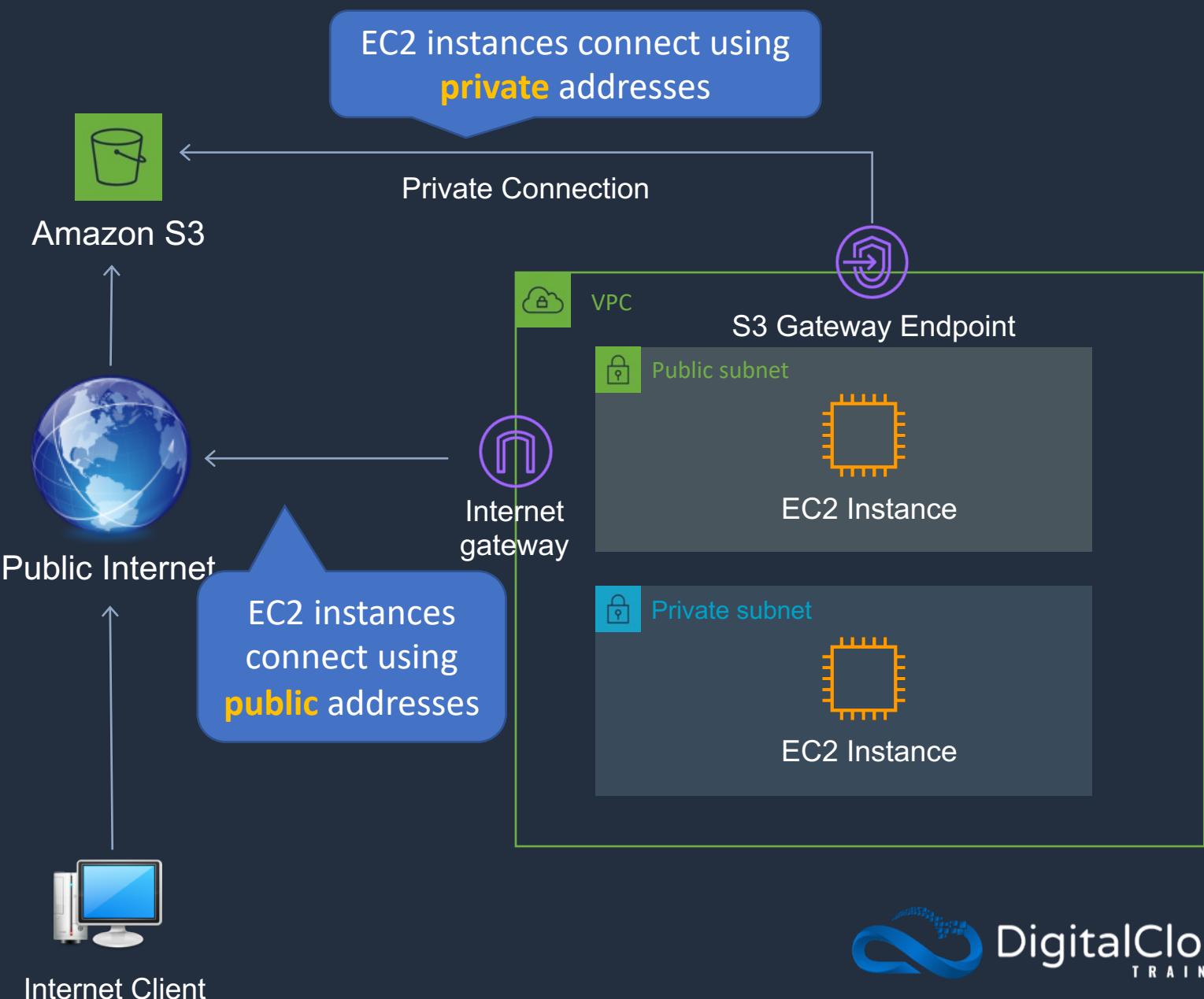
<http://s3.aws-region.amazonaws.com/bucket>



Object

An object consists of:

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





# Block, File, and Object Storage



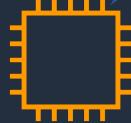
Amazon Elastic Block  
Store (EBS)

Availability Zone



HDD/SSD

Volume



EC2 Instance

/dev/xvdf  
or C:



Amazon Elastic File  
System



File system

Corporate data center

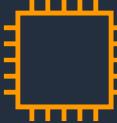


On-premises client

Uses the **NFS Protocol**

Linux only

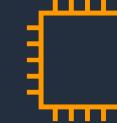
Availability Zone



EC2 Instance

/efs-mnt

Availability Zone



EC2 Instance

/efs-mnt



Amazon S3

<http://s3.amazonaws.com/bucket/object>

REST API: GET, PUT,  
POST, SELECT, DELETE



Object



Internet Client

# 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

# Amazon S3 Lifecycle Policies





# S3 Lifecycle Management

---

There are two types of actions:

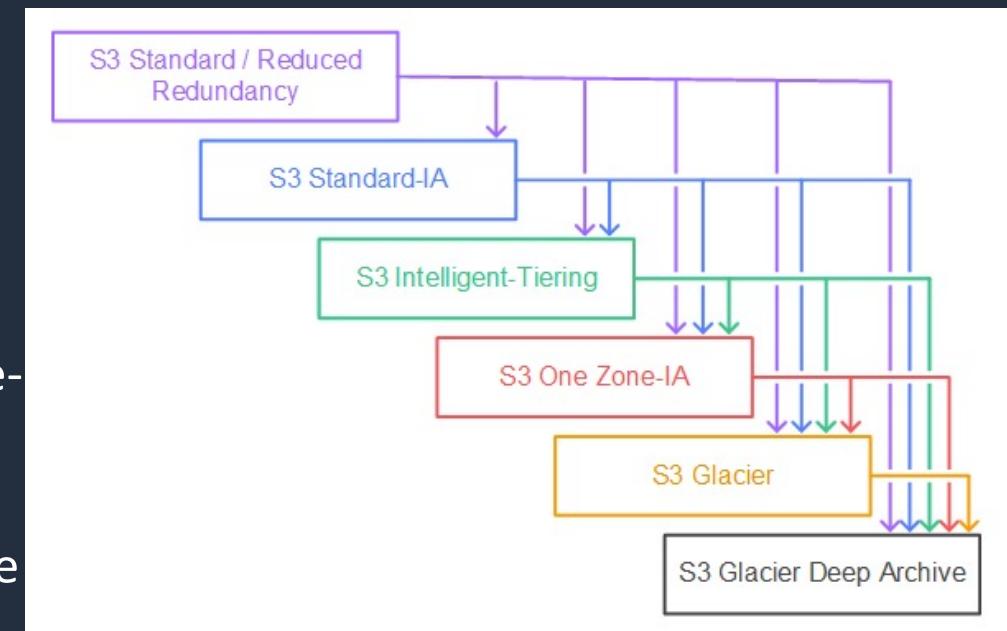
- **Transition actions** - Define when objects transition to another storage class
- **Expiration actions** - Define when objects expire (deleted by S3)



# 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

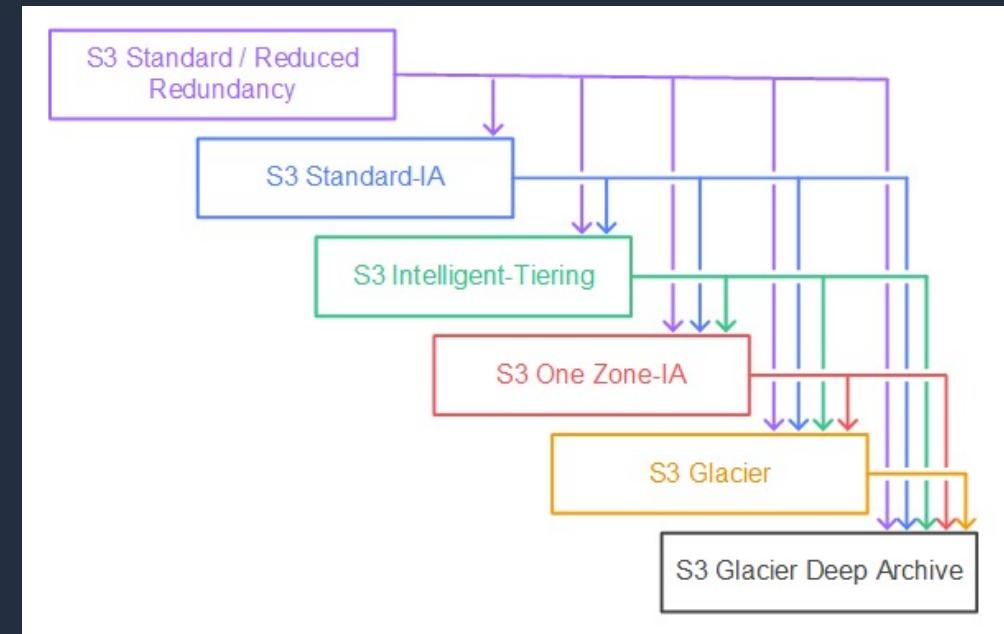




# S3 LM: Unsupported Transitions

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





# S3 Lifecycle Management

---

- Can create a lifecycle policy through the console or CLI/API
- When configured using the CLI/API an XML or JSON file must be supplied
- API actions to create/update/delete lifecycle policies:
  - **PutBucketLifecycleConfiguration** - Creates a new lifecycle configuration for the bucket or replaces an existing lifecycle configuration
  - **GetBucketLifecycleConfiguration** - Returns the lifecycle configuration information set on the bucket
  - **DeleteBucketLifecycle** - Deletes the lifecycle configuration from the specified bucket



# Example S3 Lifecycle Policy (XML)

```
<LifecycleConfiguration>
  <Rule>
    <ID>ExampleRule</ID>
    <Filter>
      <Prefix>documents/</Prefix>
    </Filter>
    <Status>Enabled</Status>
    <Transition>
      <Days>365</Days>
      <StorageClass>GLACIER</StorageClass>
    </Transition>
    <Expiration>
      <Days>3650</Days>
    </Expiration>
  </Rule>
</LifecycleConfiguration>
```

# Create Lifecycle Policies



# S3 Versioning and Replication





# S3 Versioning

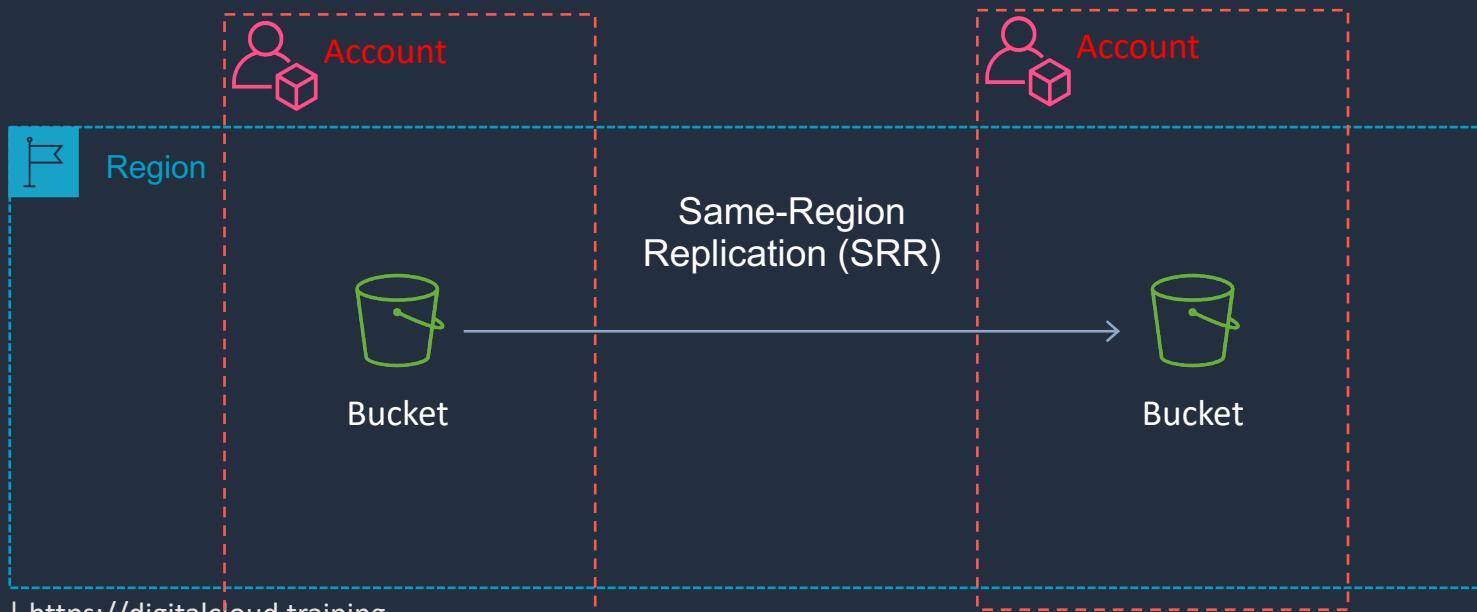
---

- 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 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



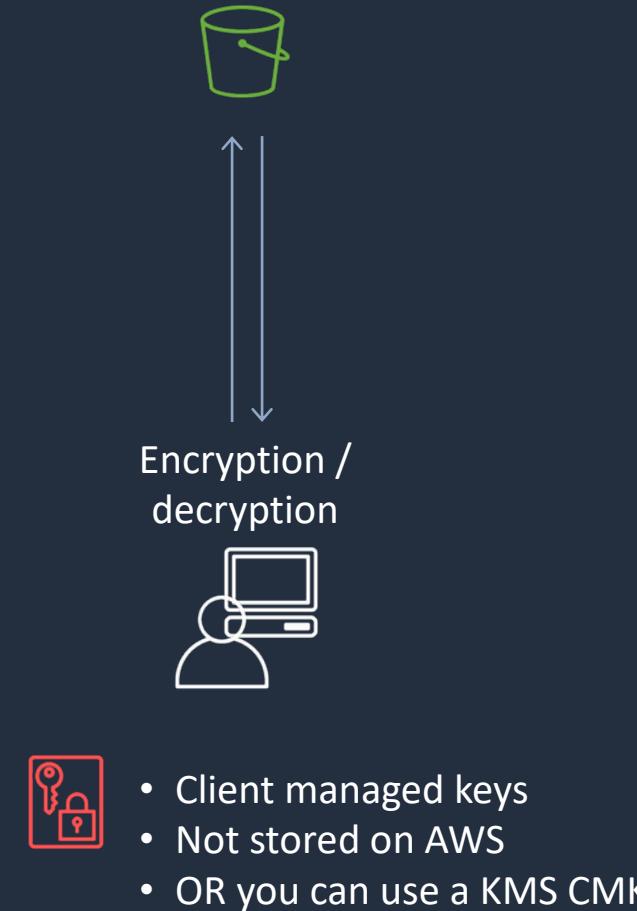


# S3 Encryption

## 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]
```

# Test S3 Encryption



# 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

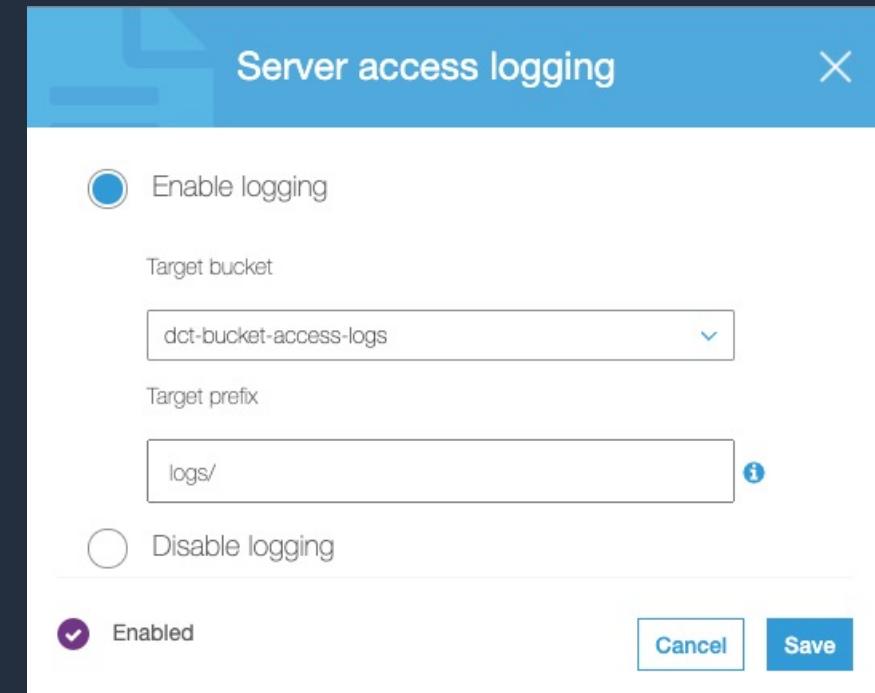
# Server Access Logging





# 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 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

Events

Add notification Delete Edit

Name	Events	Filter	Type
New event			

Name i  
Notify for Uploads

Events i  
 PUT       All object delete events  
 POST       Restore initiated  
 COPY       Restore completed  
 Multipart upload completed       Replication time missed threshold  
 All object create events       Replication time completed after threshold  
 Object in RRS lost       Replication time not tracked  
 Permanently deleted       Replication failed  
 Delete marker created

Prefix i  
e.g. images/

Suffix i  
e.g. .jpg

Send to i  
SNS Topic

SNS  
NotifyMe

# AWS Storage Gateway



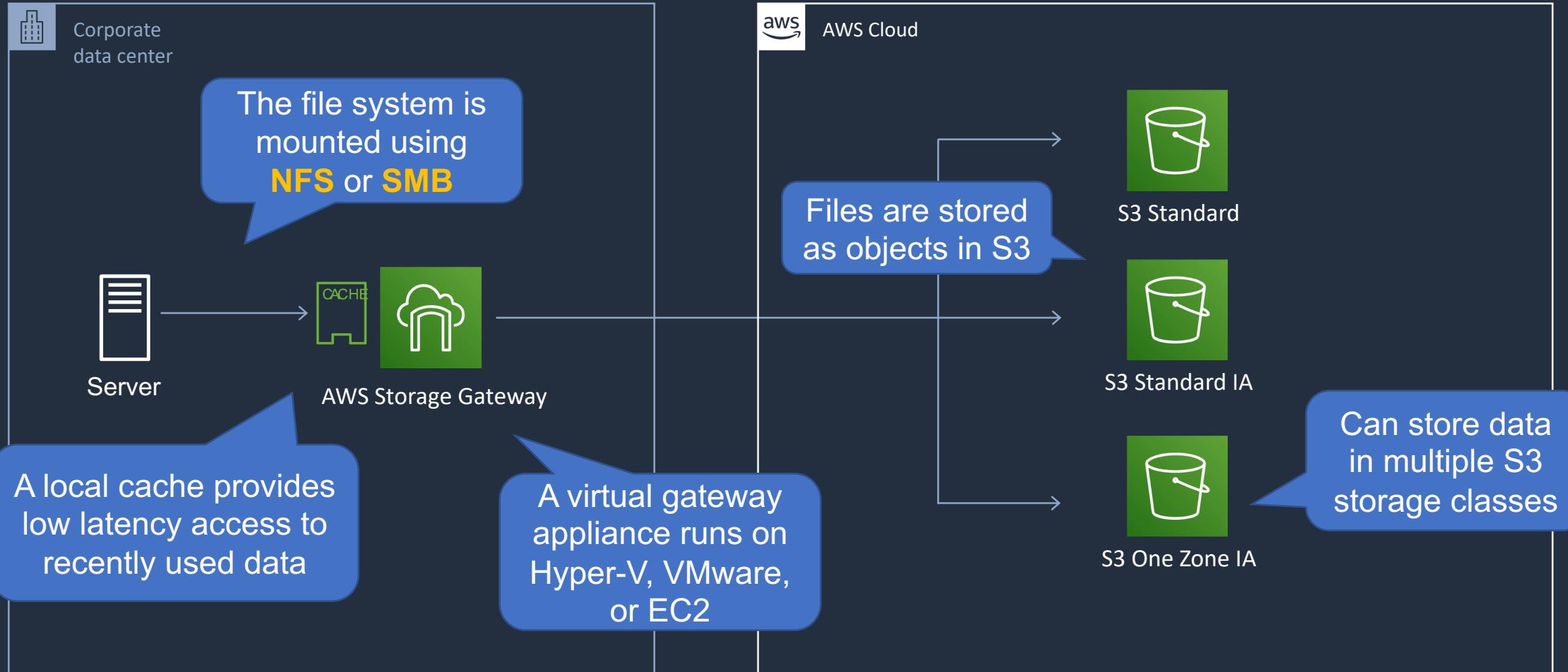


# AWS Storage Gateway





# AWS Storage Gateway – File Gateway





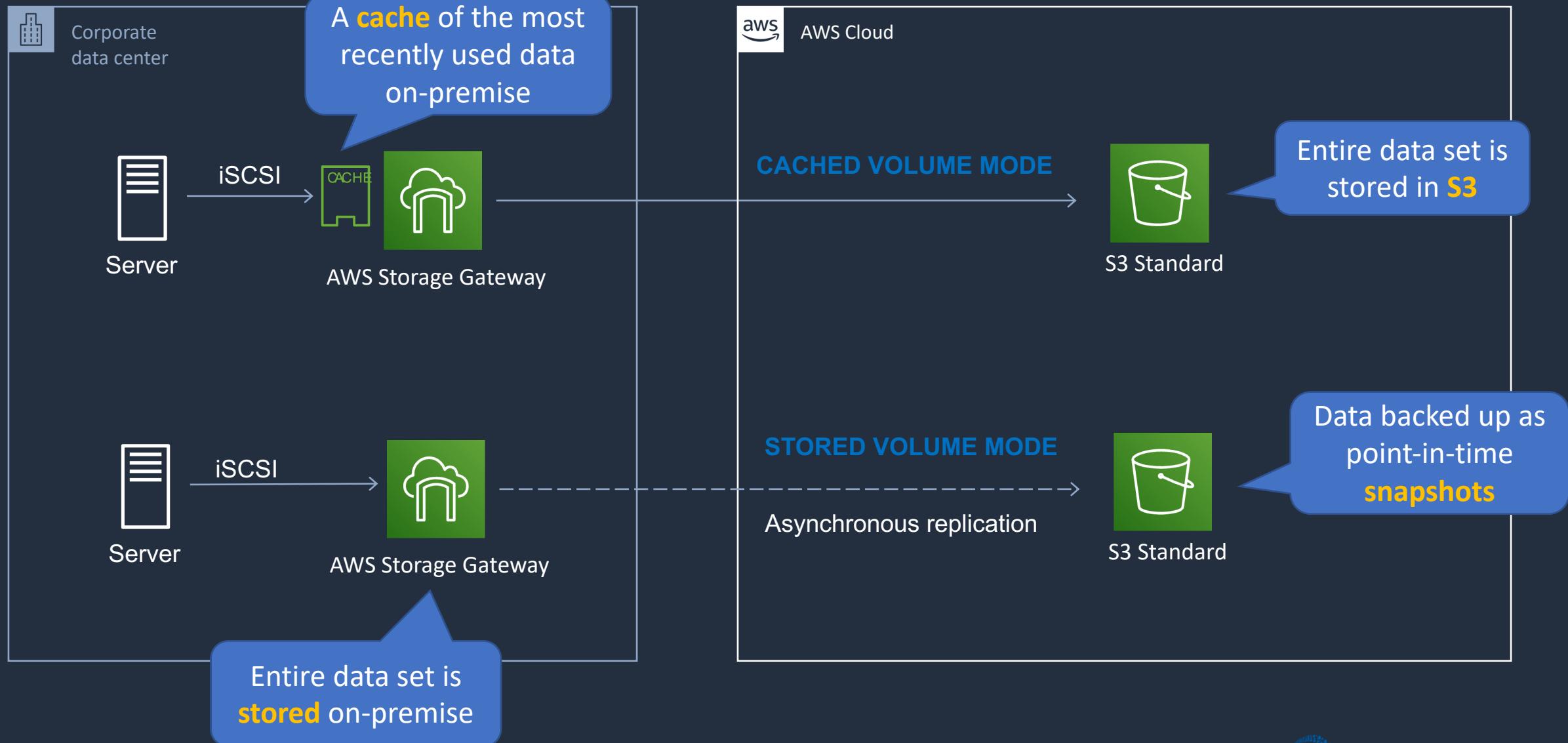
# AWS Storage Gateway – File Gateway

---

- 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





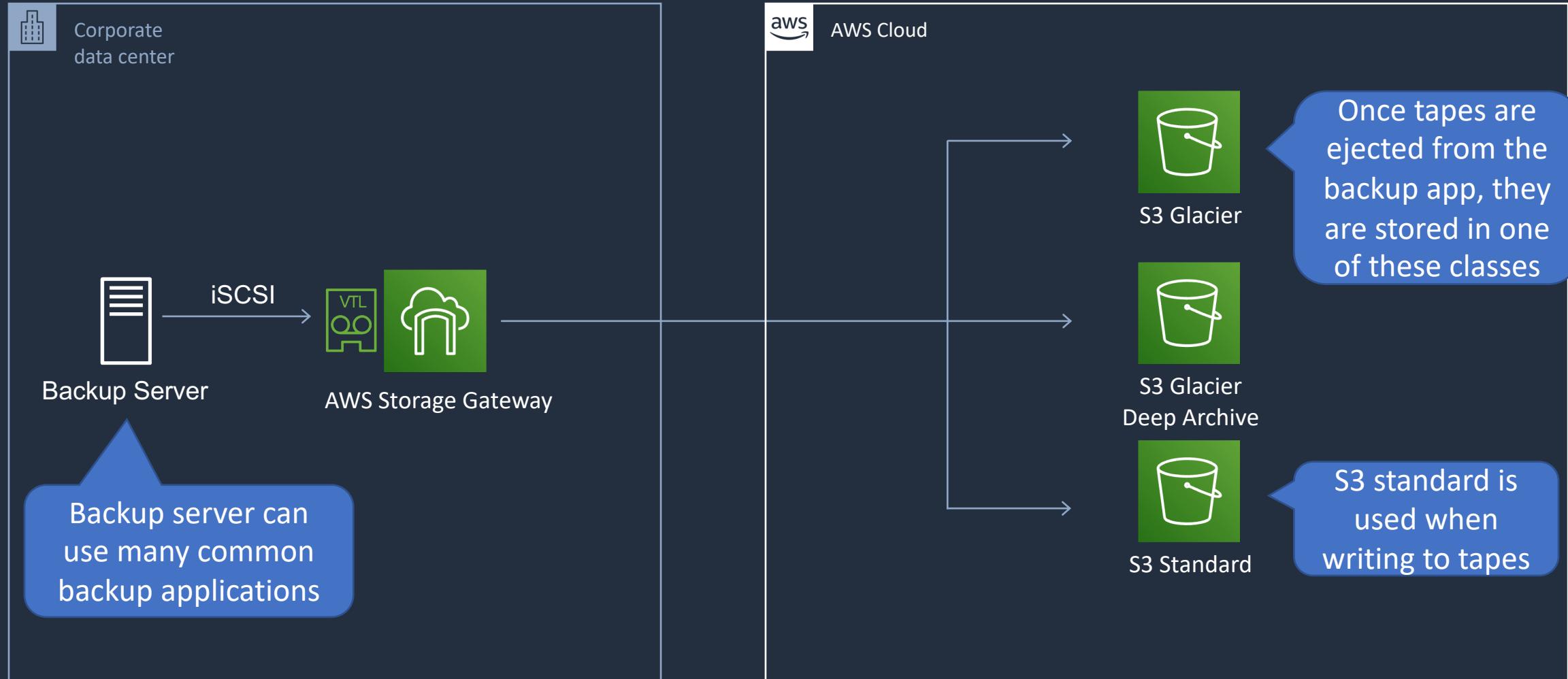
# AWS Storage Gateway - Volume Gateway

---

- 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)

# SECTION 9

## DNS, Caching, and Performance Optimization

# 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?

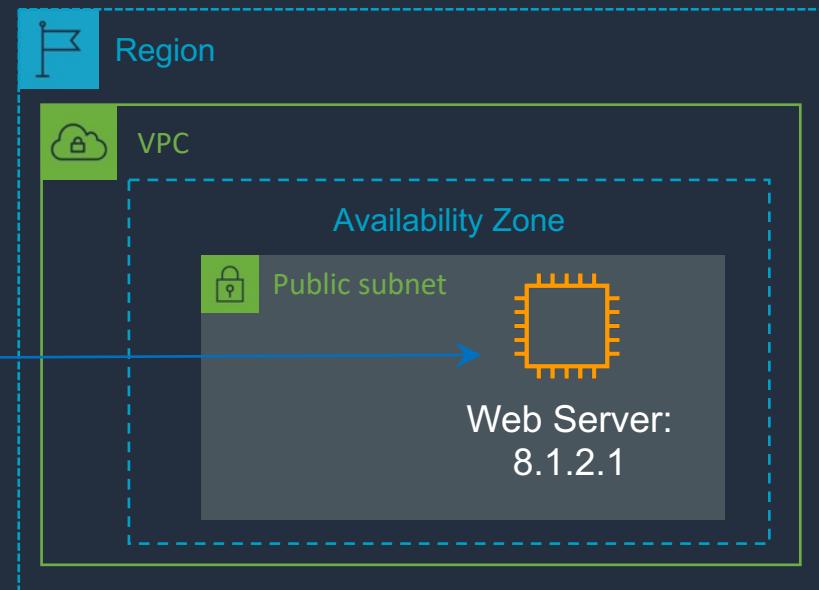
example.com

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



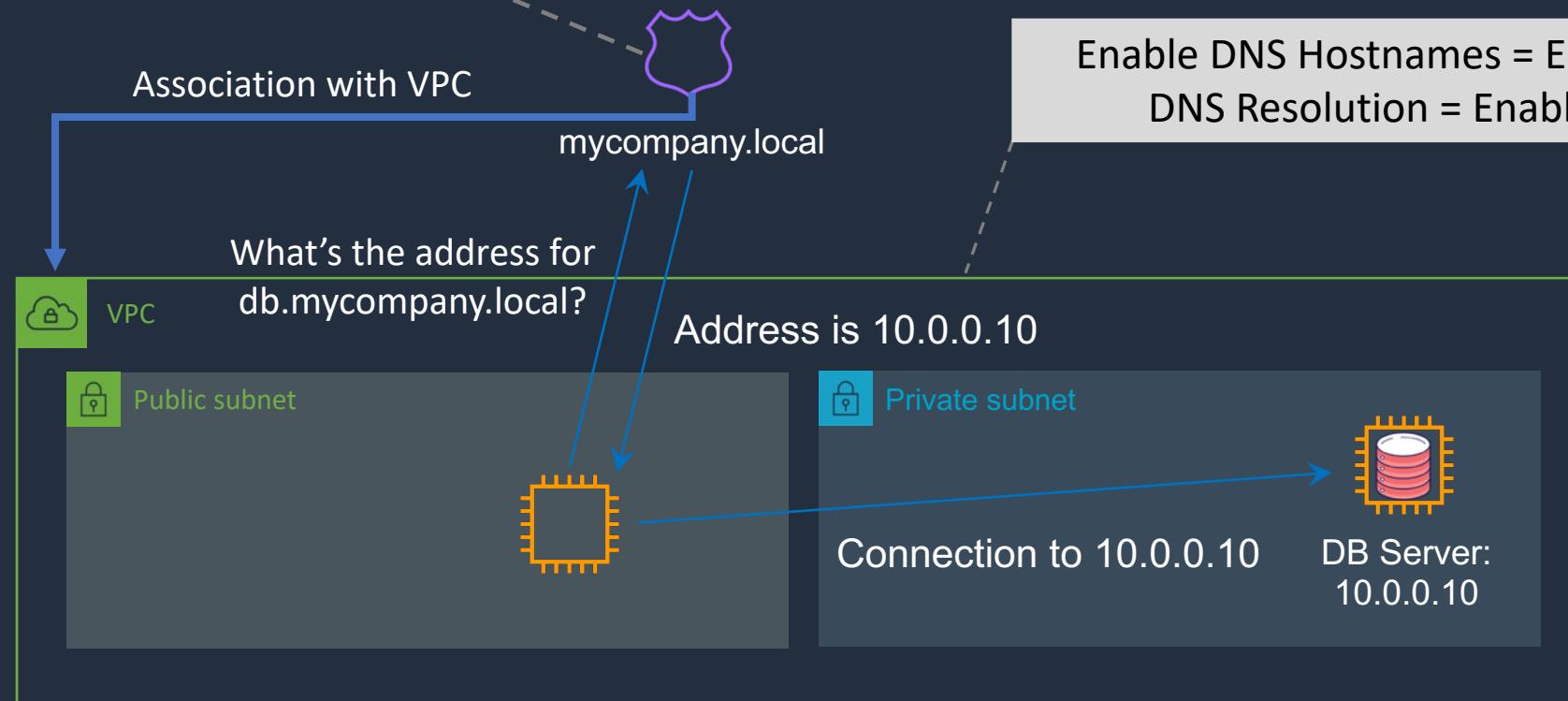
# Amazon Route 53 Hosted Zones

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



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

Amazon Route 53



# 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





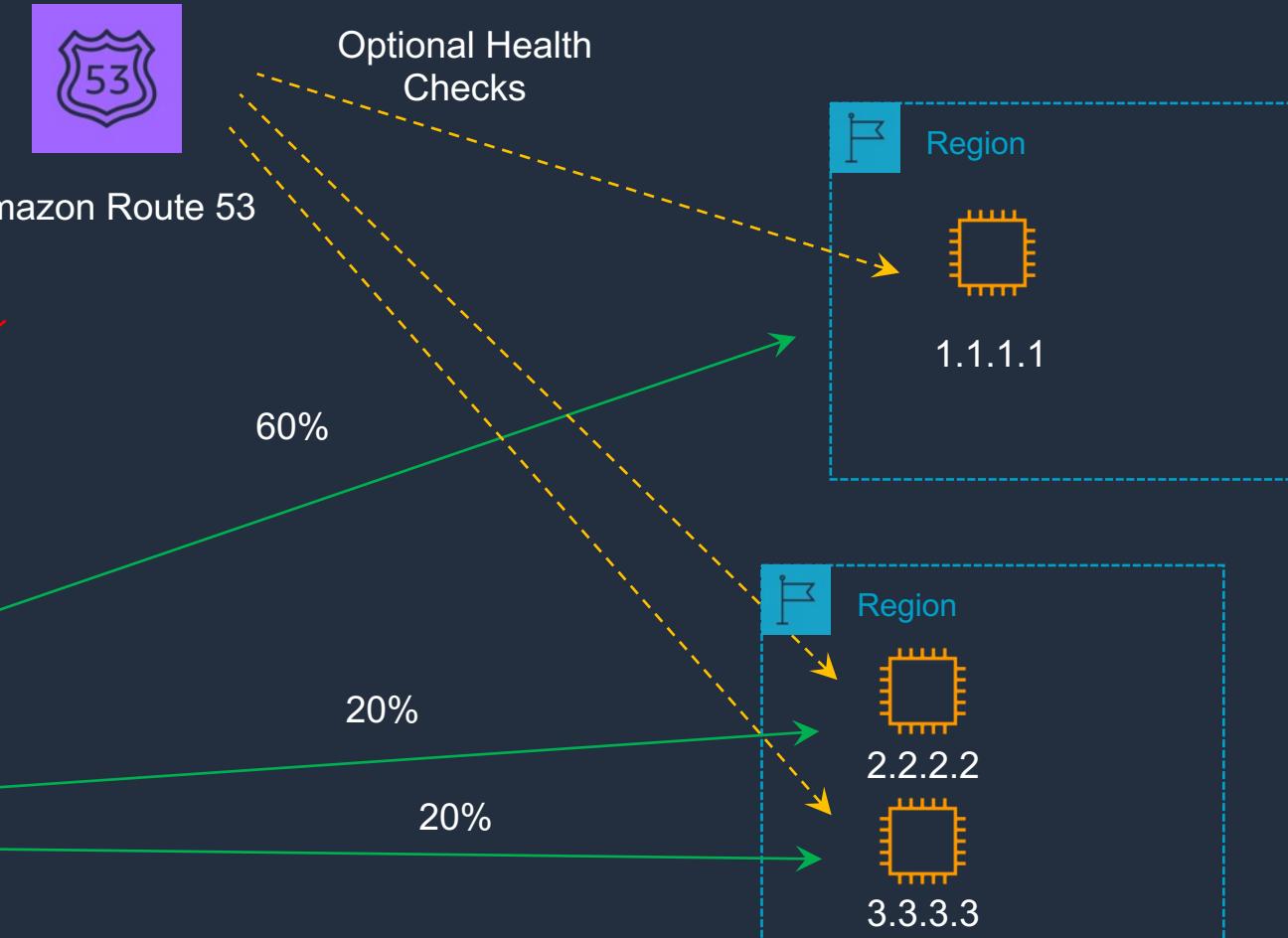
# Amazon Route 53 – Weighted Routing Policy

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





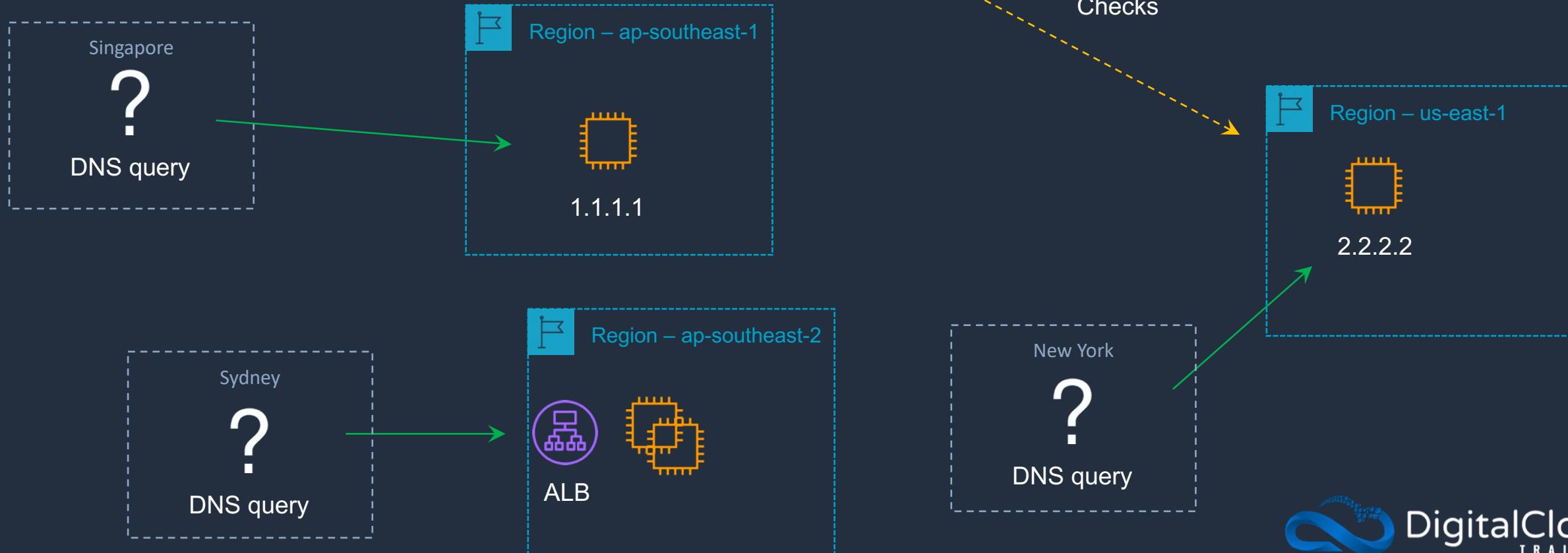
# Amazon Route 53 – Latency Routing Policy

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

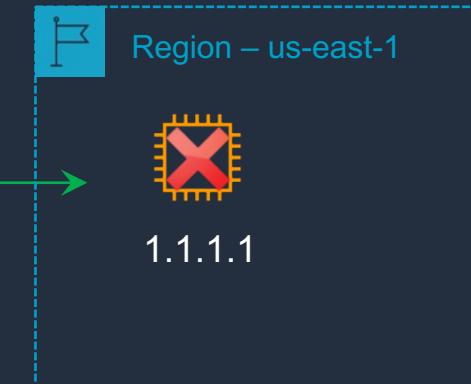


Amazon Route 53

Health check is  
**required** on Primary

?

DNS query



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



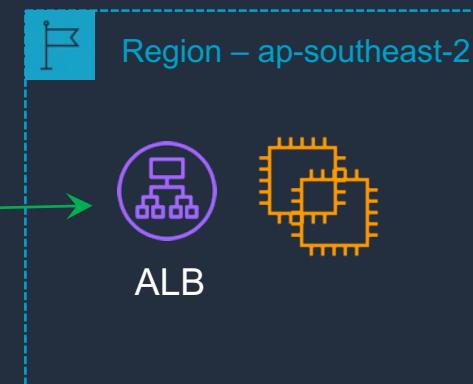
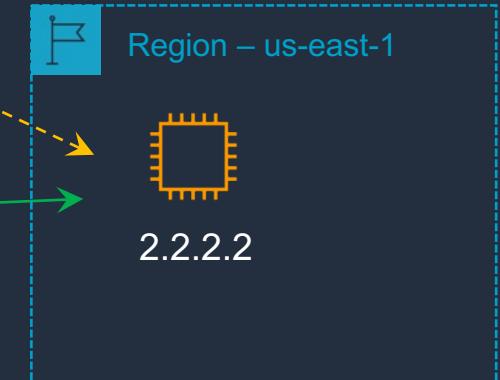
# Amazon Route 53 – Geolocation Routing Policy

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



Optional Health Checks

Amazon Route 53



# 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



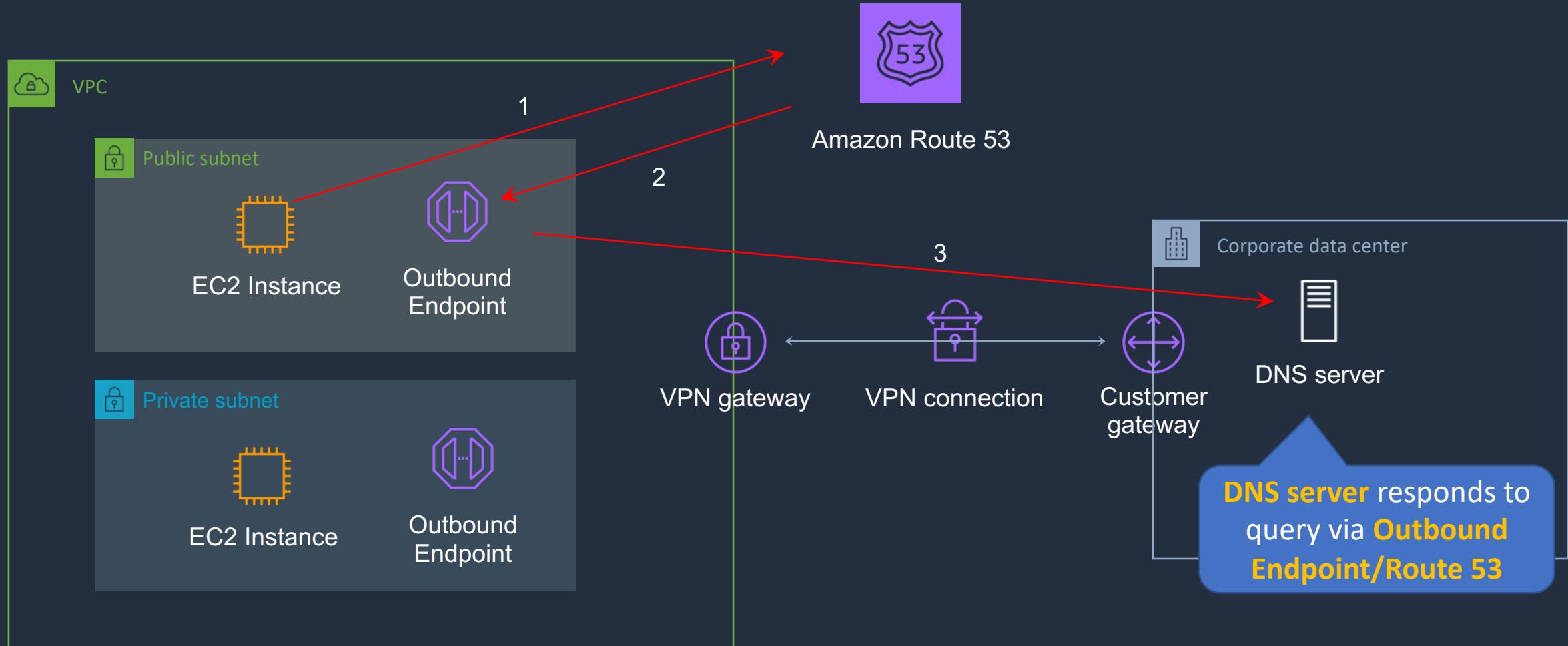
# Test Route 53 Routing Policies



# Amazon Route 53 Resolver

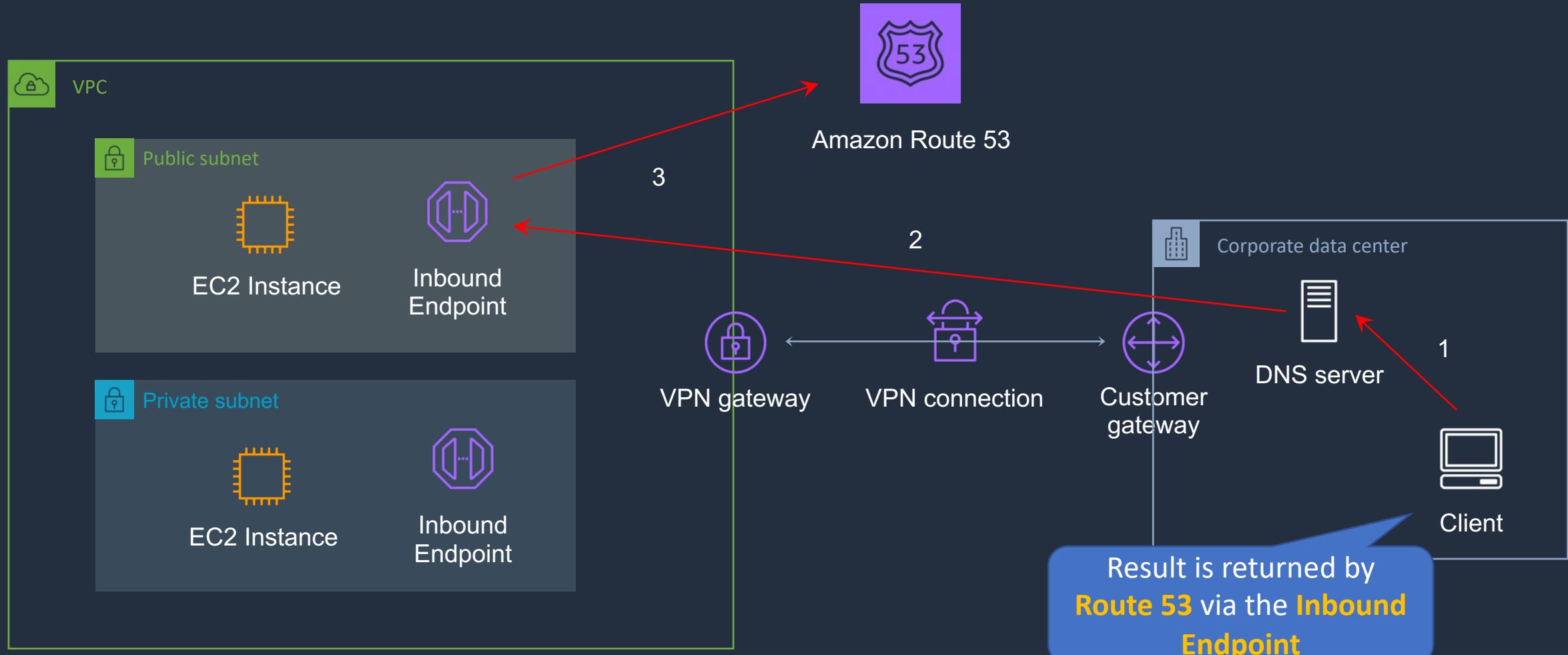


# Route 53 Resolver – Outbound Endpoints





# Route 53 Resolver – Inbound Endpoints

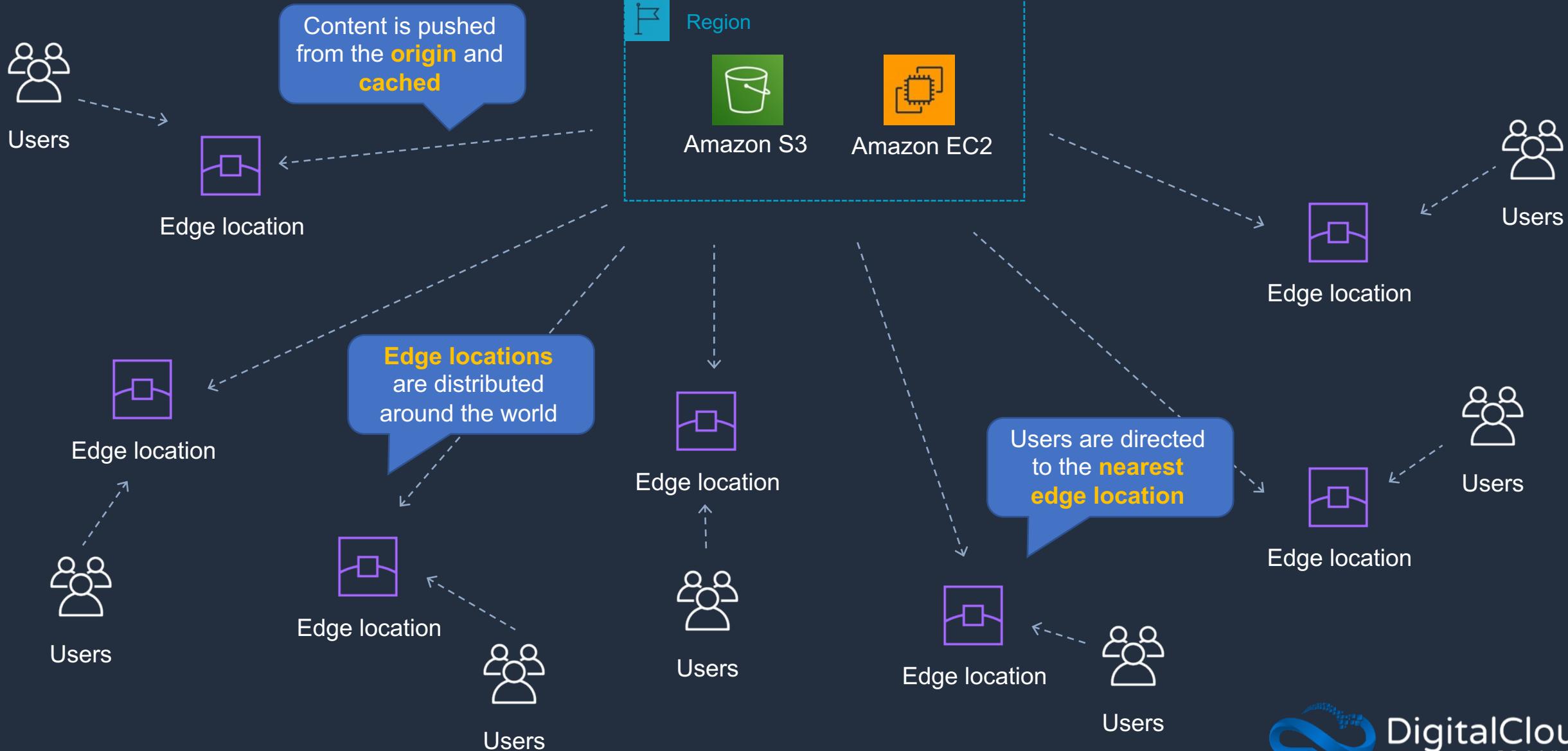


# Amazon CloudFront Origins and Distributions





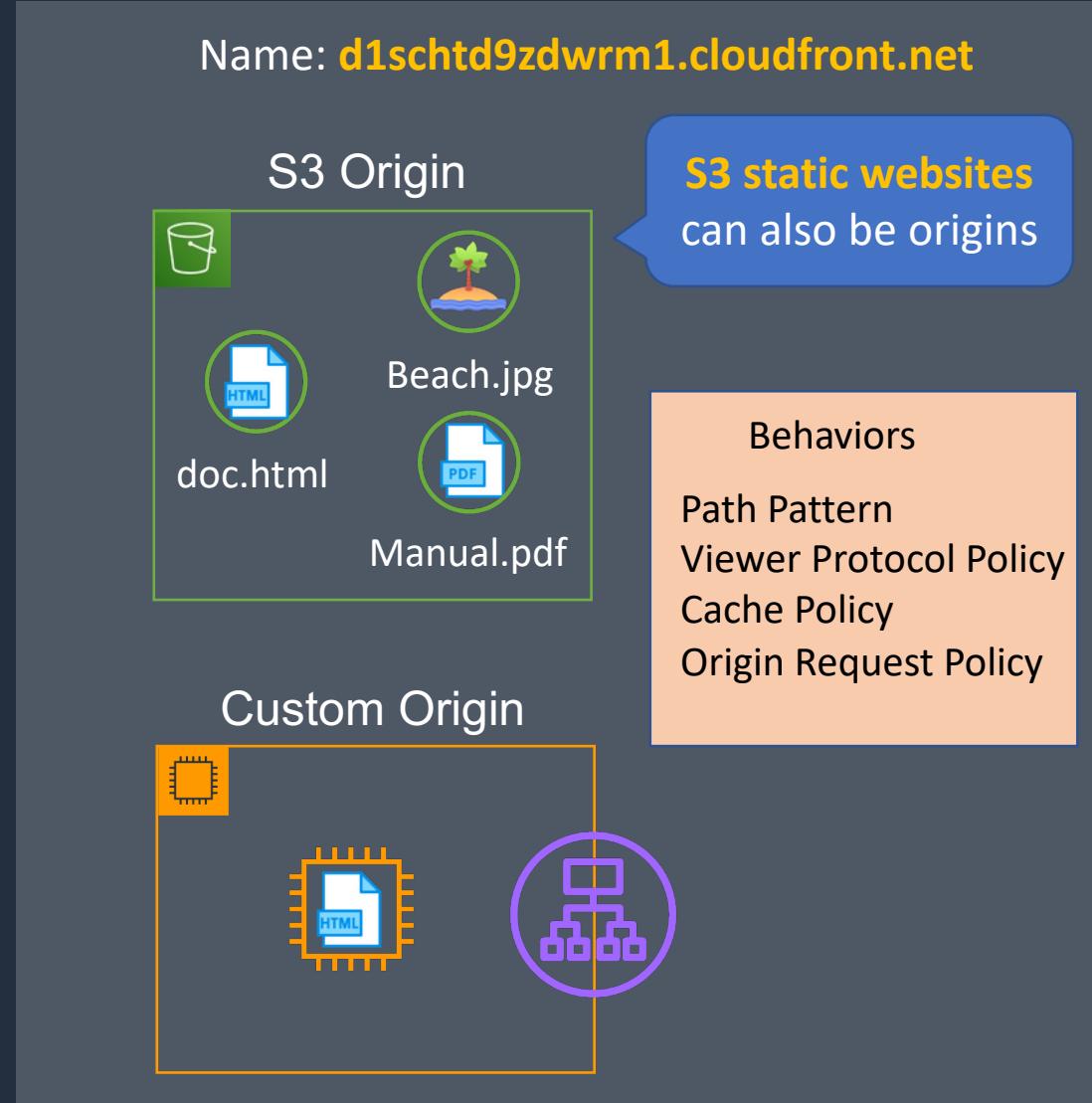
# Amazon CloudFront





# CloudFront Origins and Distributions

## CloudFront Distribution



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

### CloudFront Web Distribution:

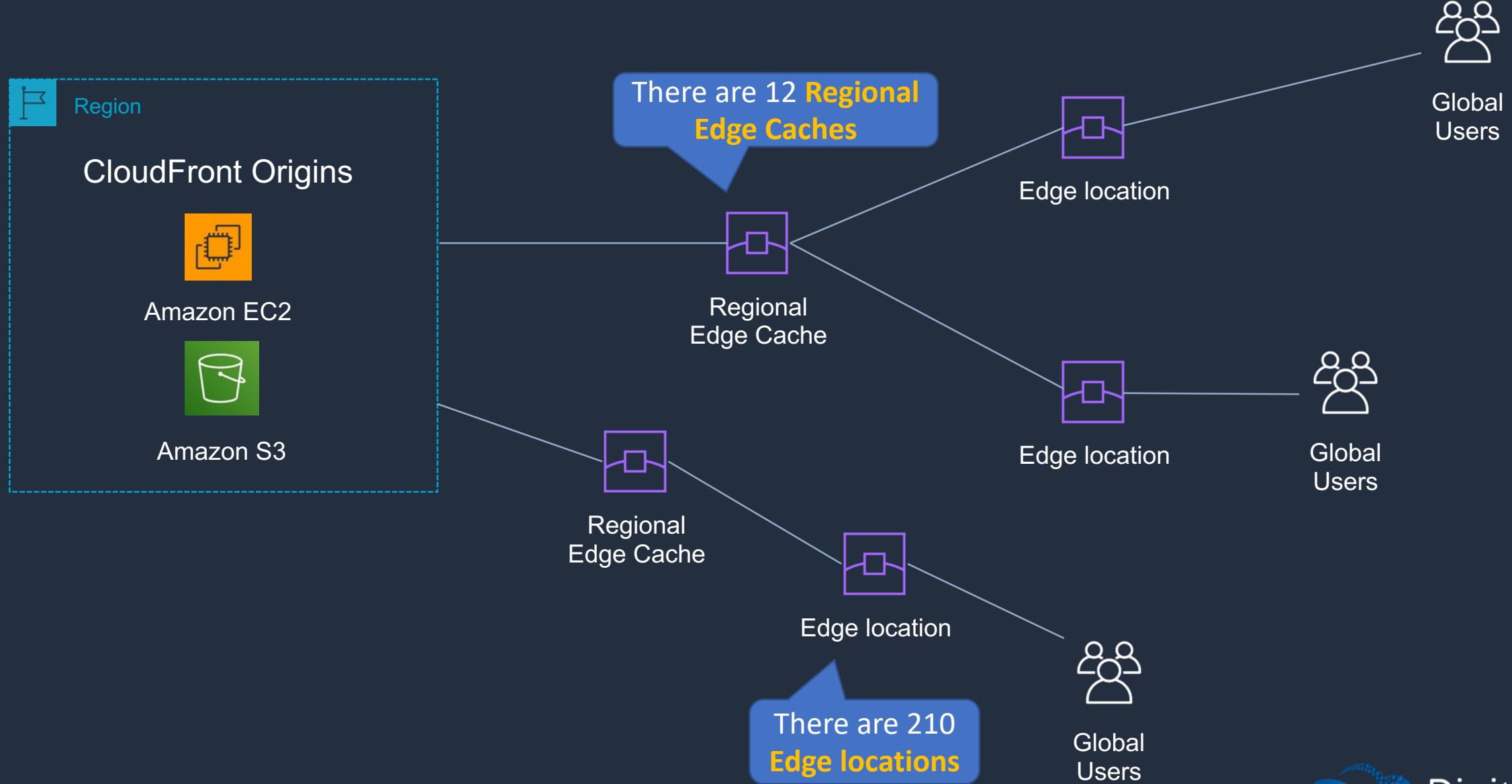
- Speed up distribution of static and dynamic content, for example, .html, .css, .php, and graphics files.
- Distribute media files using HTTP or HTTPS.
- Add, update, or delete objects, and submit data from web forms.
- Use live streaming to stream an event in real time.

# Amazon CloudFront Caching and Behaviors



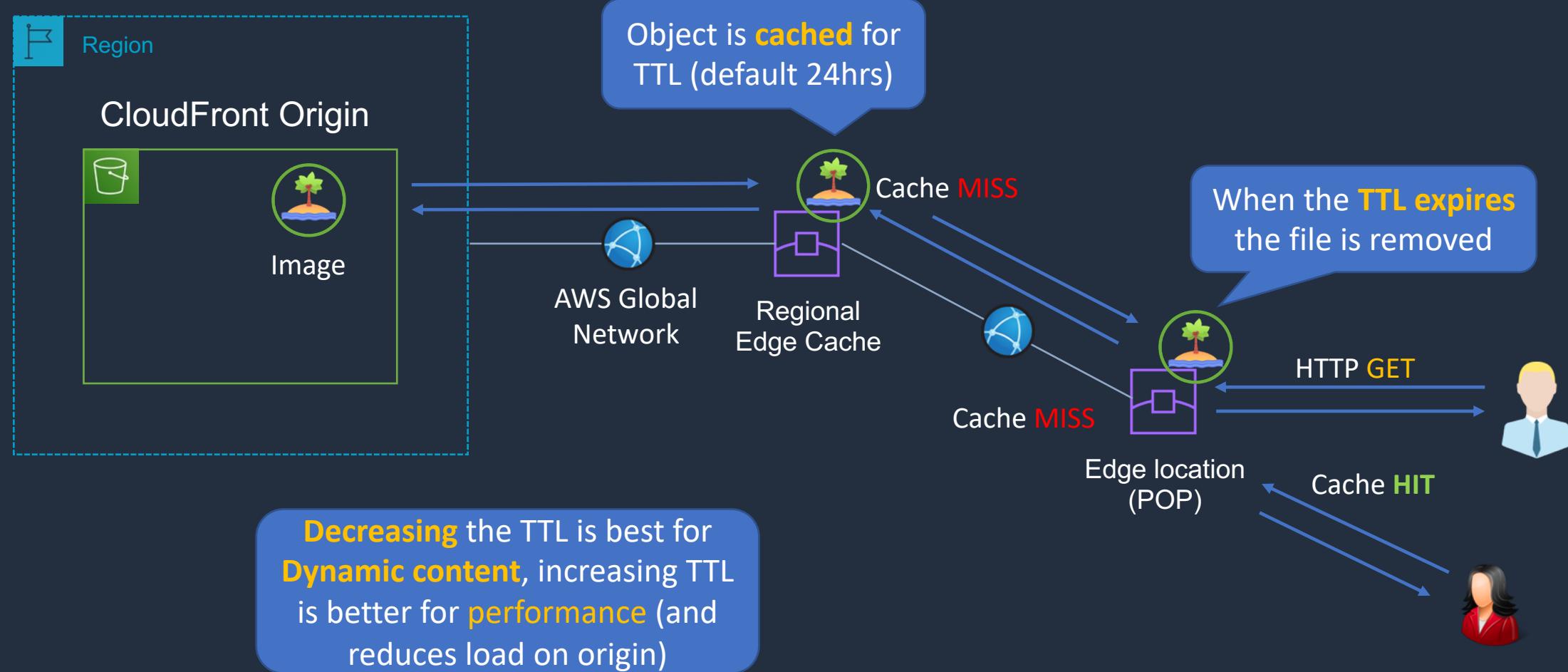


# Amazon CloudFront Caching





# Amazon CloudFront Caching



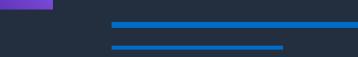


# Amazon CloudFront Caching

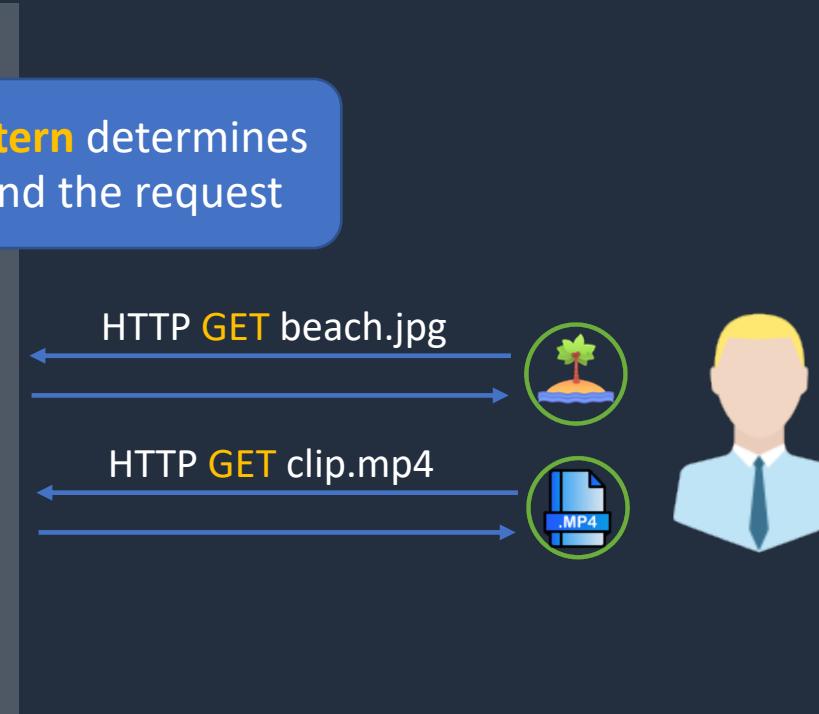
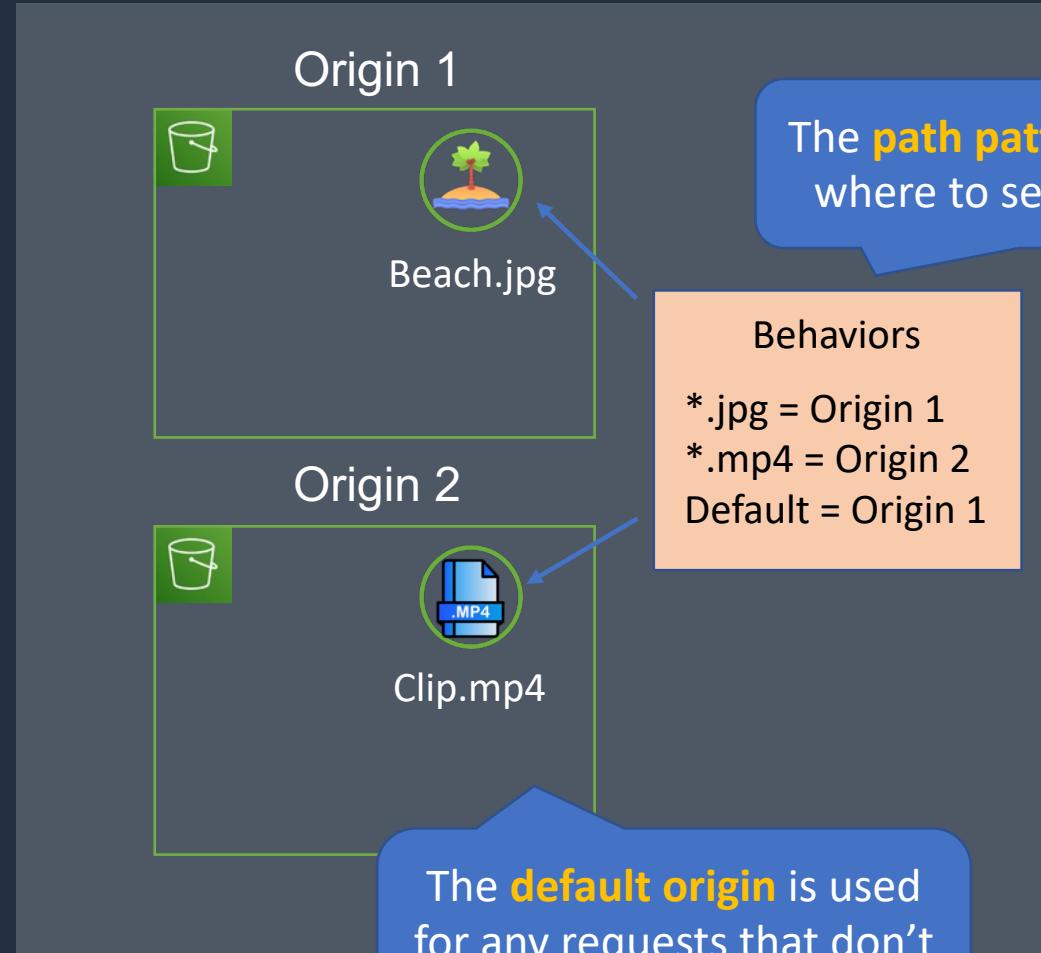
---

- 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

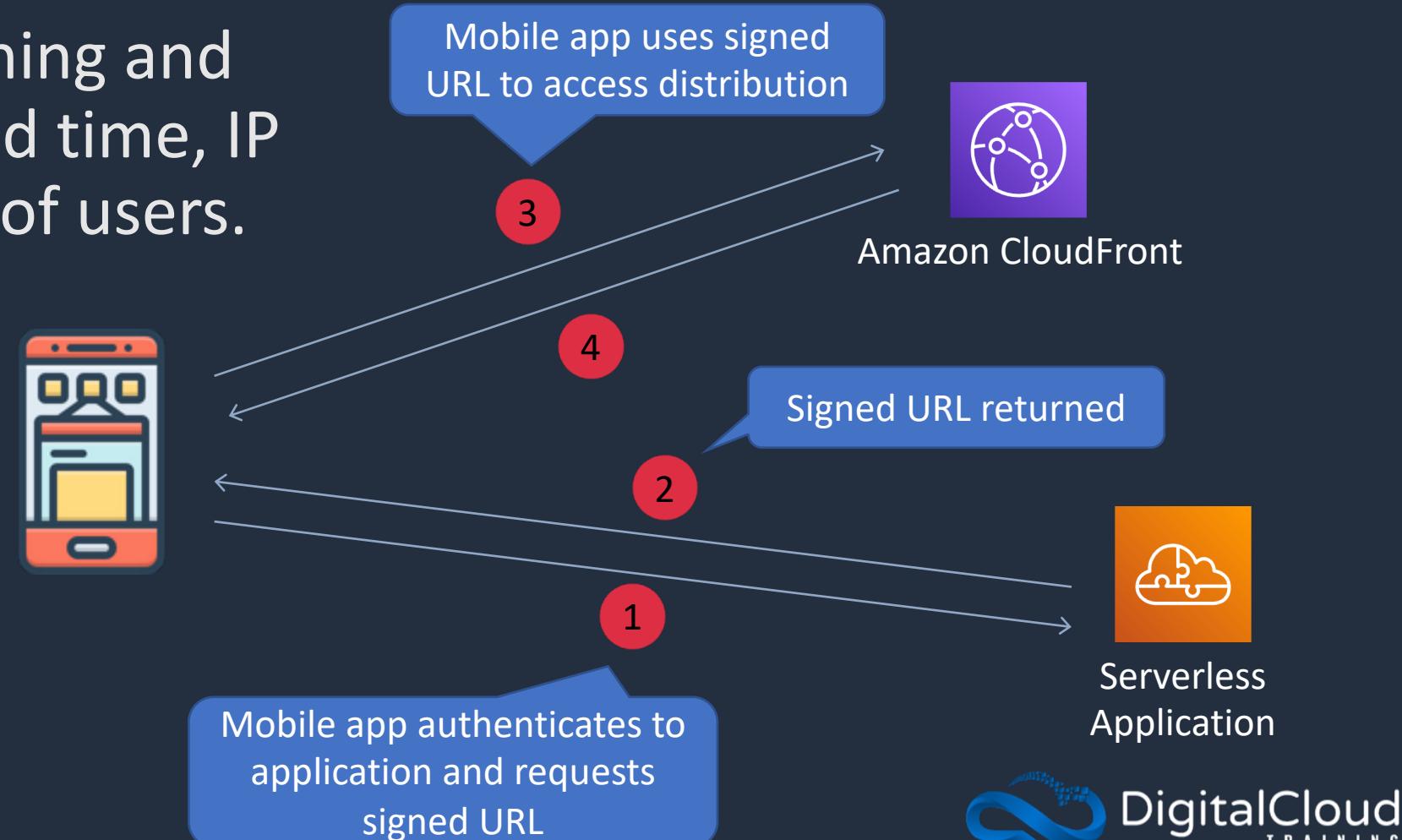




# CloudFront Signed URLs

- 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**.





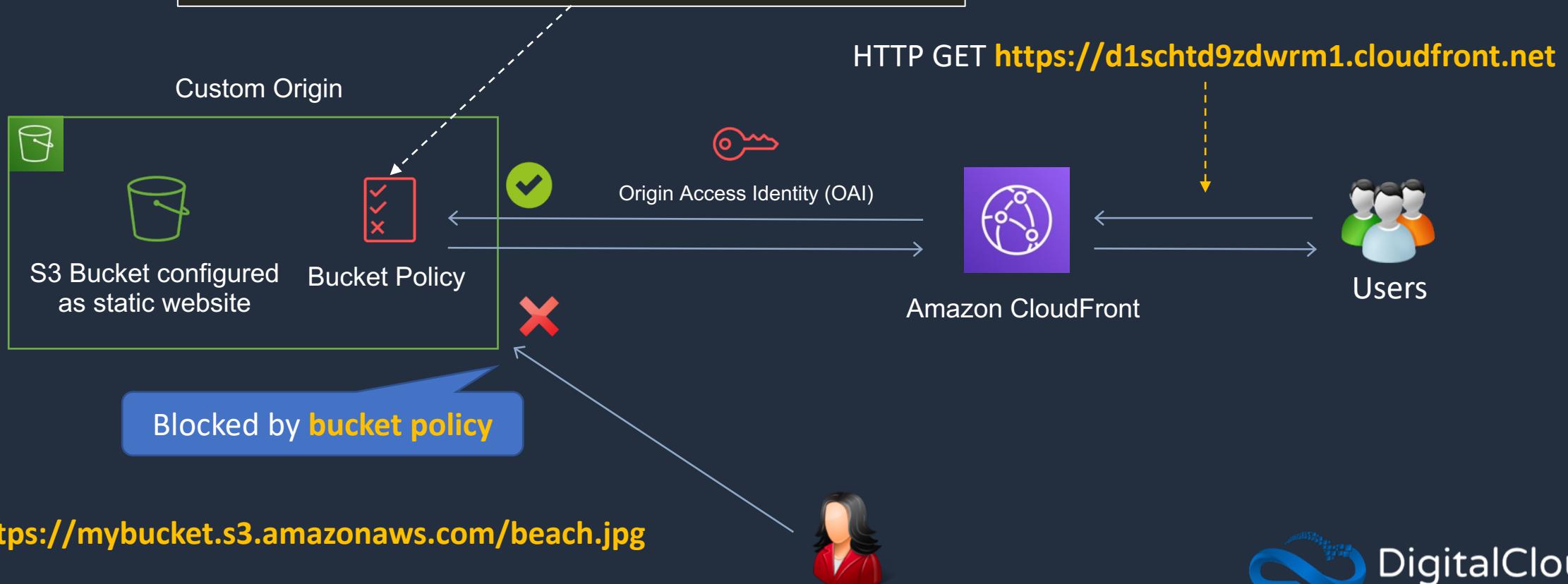
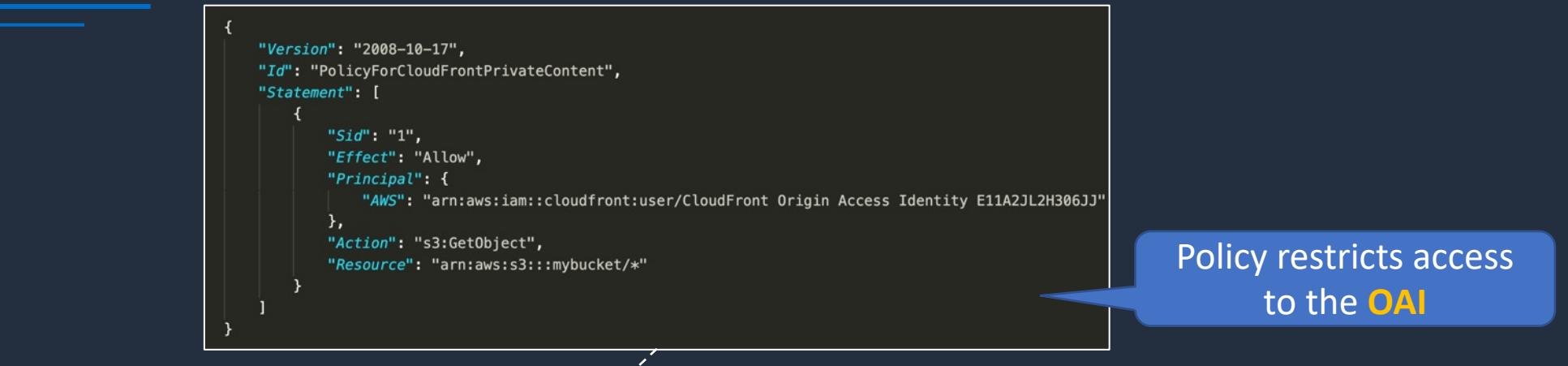
# CloudFront Signed Cookies

---

- 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





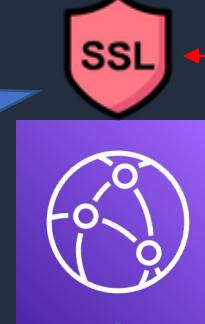
# CloudFront SSL/TLS



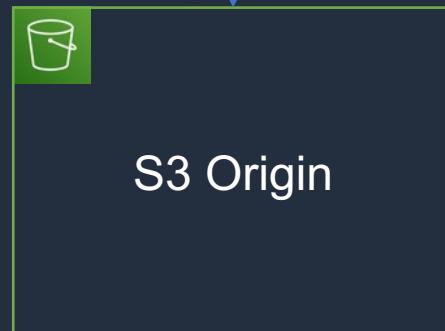
- Viewer Protocol Policy
- HTTP and HTTPS
  - Redirect HTTP to HTTPS
  - HTTPS Only

## Viewer Protocol

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



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



## Origin Protocol

Origin certificates must be **public certificates**



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



AWS Certificate Manager

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**

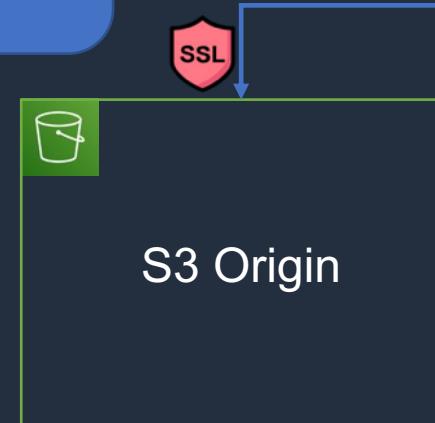
Name: [myotherdomain.com](https://myotherdomain.com)



Request URL includes  
domain name which  
**matches** certificate

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

**Multiple** certificates  
share the **same IP**  
with SNI



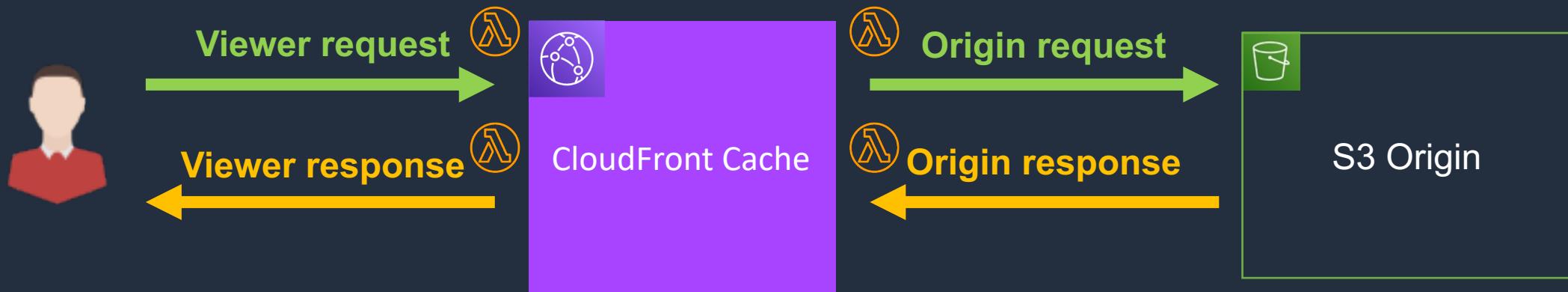
# Lambda@Edge





# Lambda@Edge

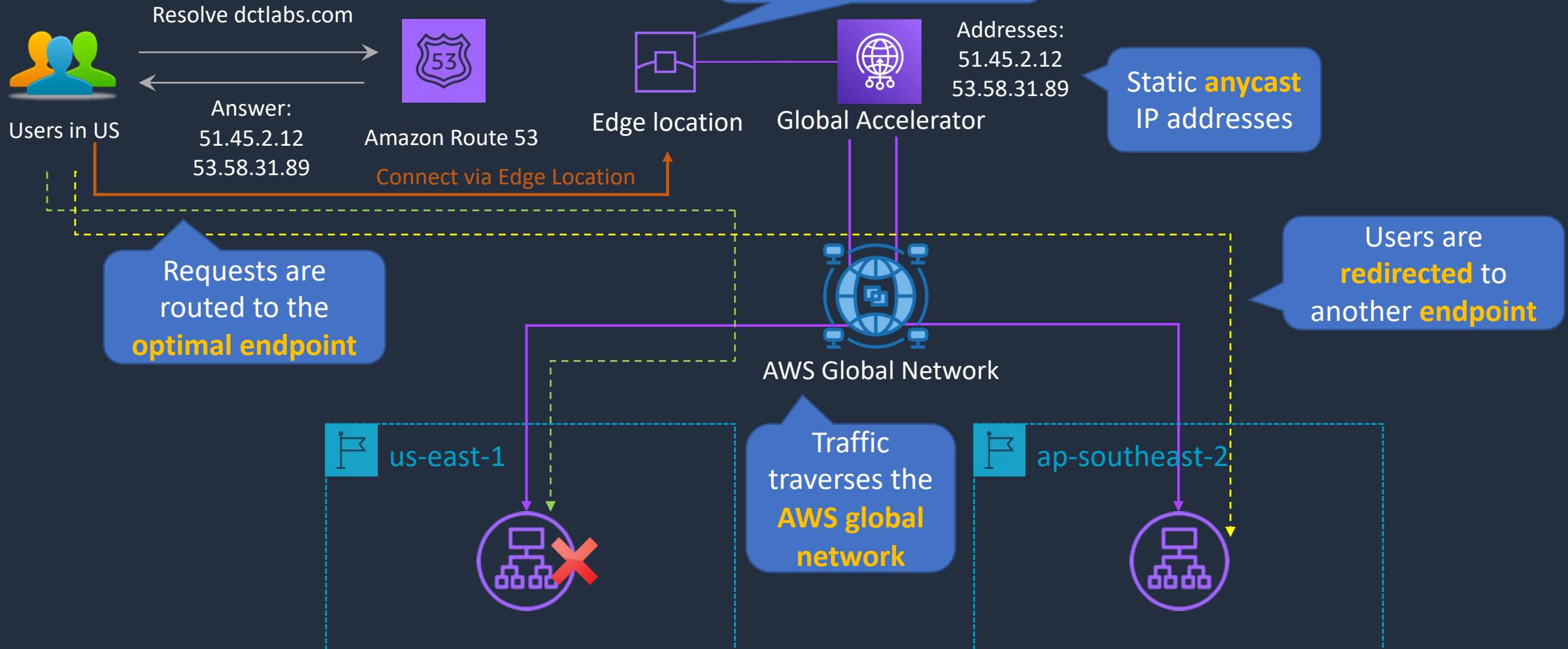
- Run Node.js and Python Lambda functions to customize the content CloudFront delivers
- Executes functions closer to the viewer
- Can be run at the following points
  - After CloudFront receives a request from a viewer (viewer request)
  - Before CloudFront forwards the request to the origin (origin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



# AWS Global Accelerator



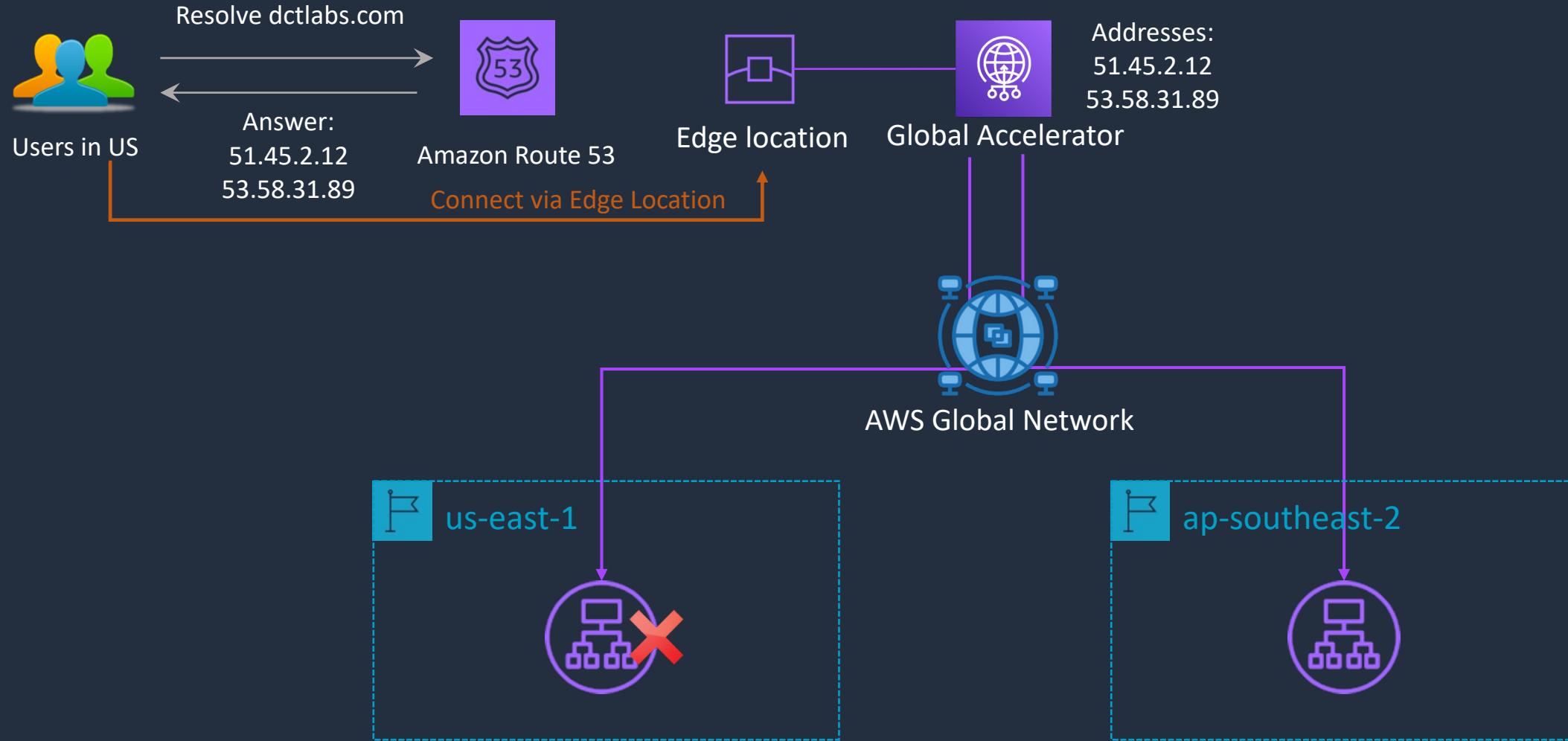
# AWS Global Accelerator



# Create a Global Accelerator



# AWS Global Accelerator



# SECTION 10

## AWS Database Services

# Amazon RDS Scaling and Deployment





# Amazon RDS Overview

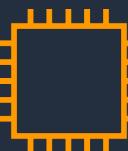
---

RDS is a **managed**, relational database



Amazon RDS

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



EC2

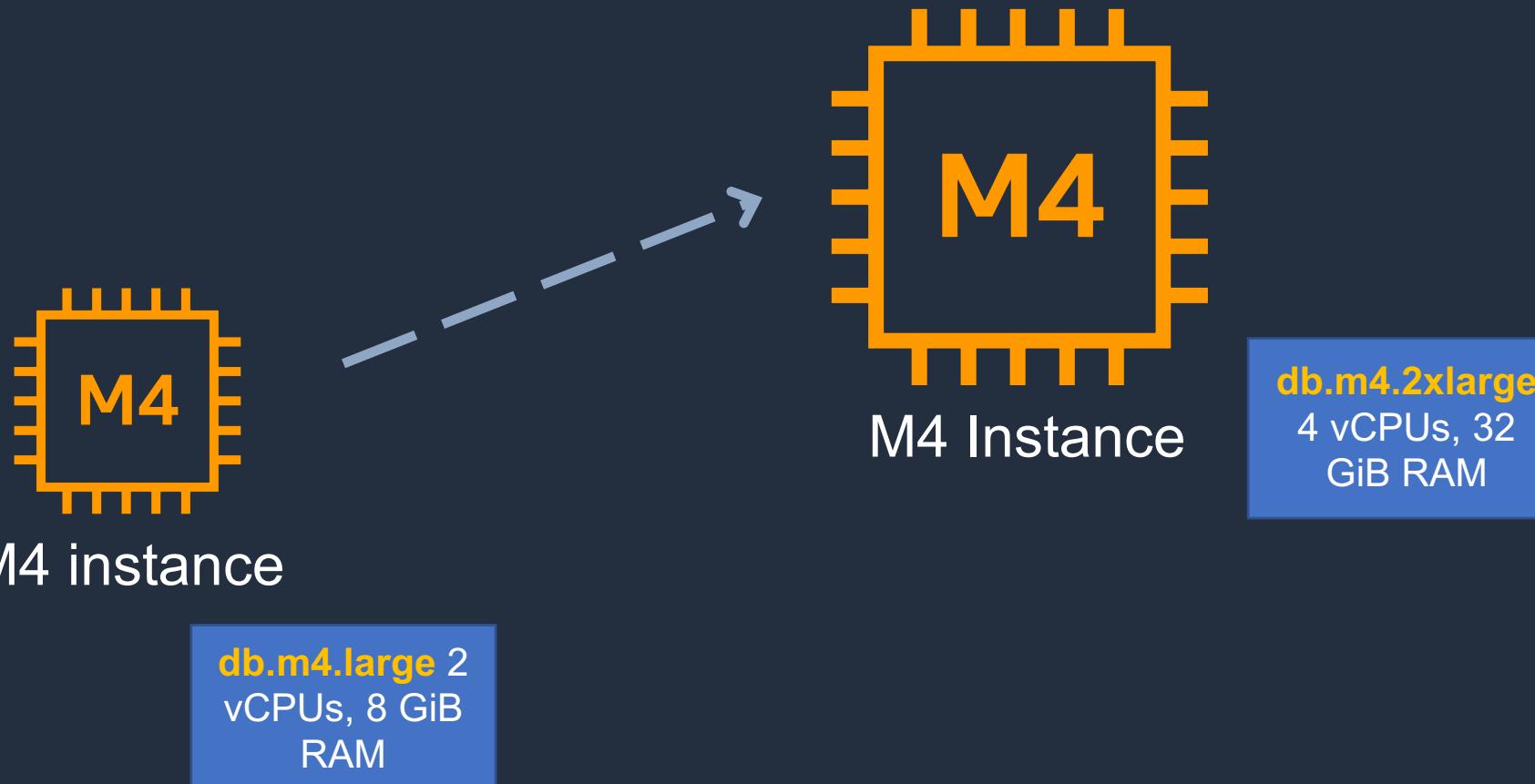
RDS supports the following database engines:

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



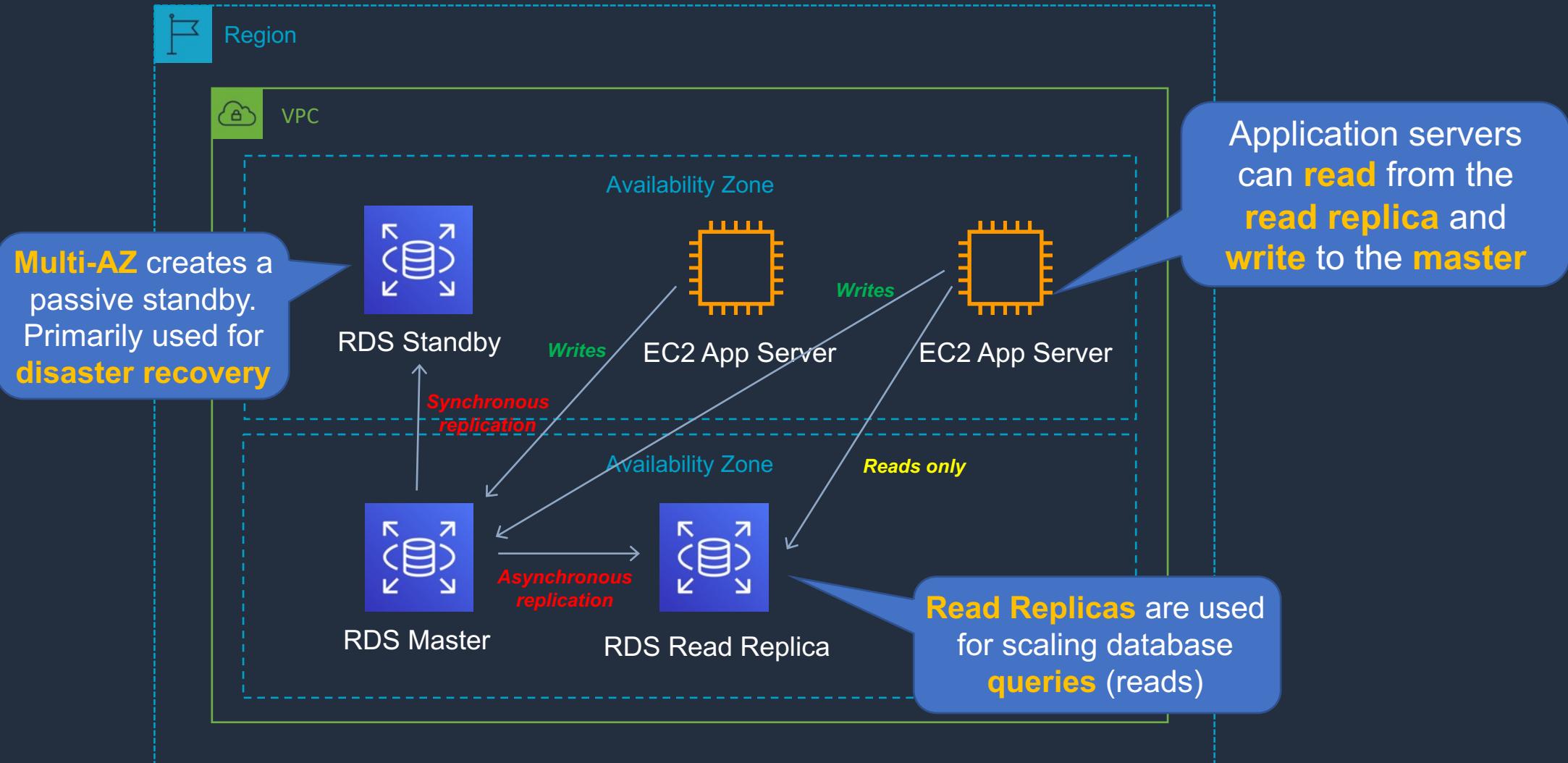
# Amazon RDS Scaling Up (vertically)

---





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





# Amazon RDS Multi-AZ and Read Replicas

---

Multi-AZ Deployments	Read Replicas
Synchronous replication – highly durable	Asynchronous replication – highly scalable
Only database engine on primary instance is active	All read replicas are accessible and can be used for read scaling
Automated backups are taken from standby	No backups configured by default
Always span two Availability Zones within a single Region	Can be within an Availability Zone, Cross-AZ, or Cross-Region
Database engine version upgrades happen on primary	Database engine version upgrade is independent from source instance
Automatic failover to standby when a problem is detected	Can be manually promoted to a standalone database instance

# Amazon RDS Backup and Recovery





# Amazon RDS Automated Backups

Backup window [Info](#)

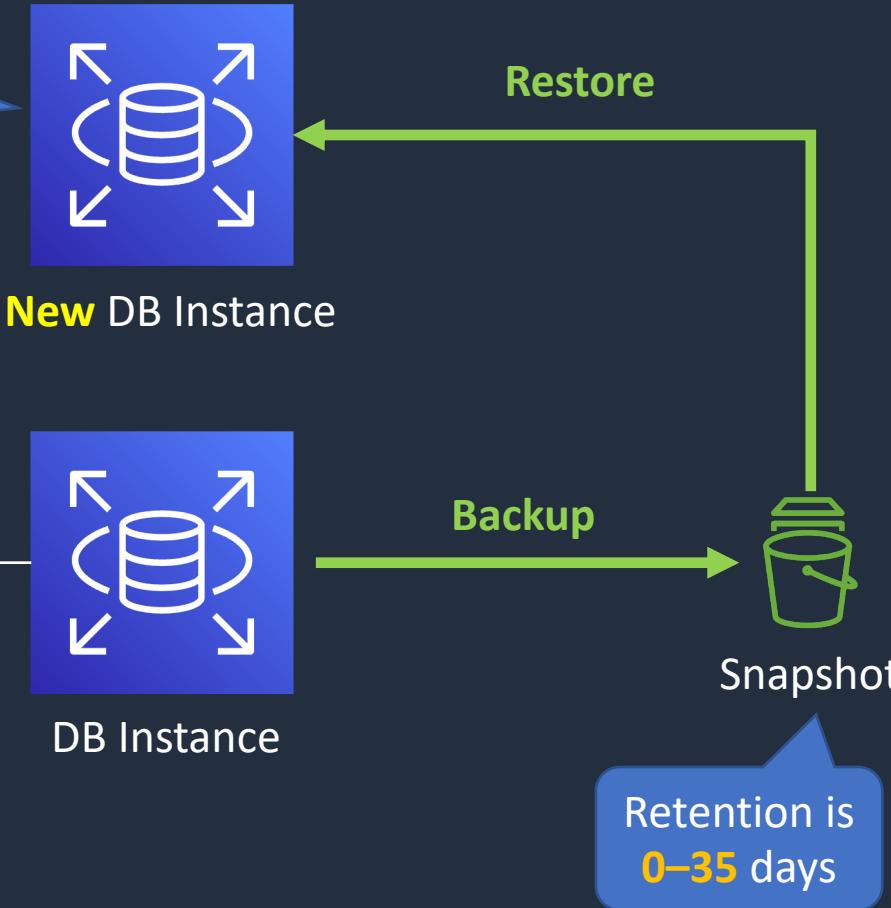
Select the period you want automated backups of the database to be created by Amazon RDS.

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

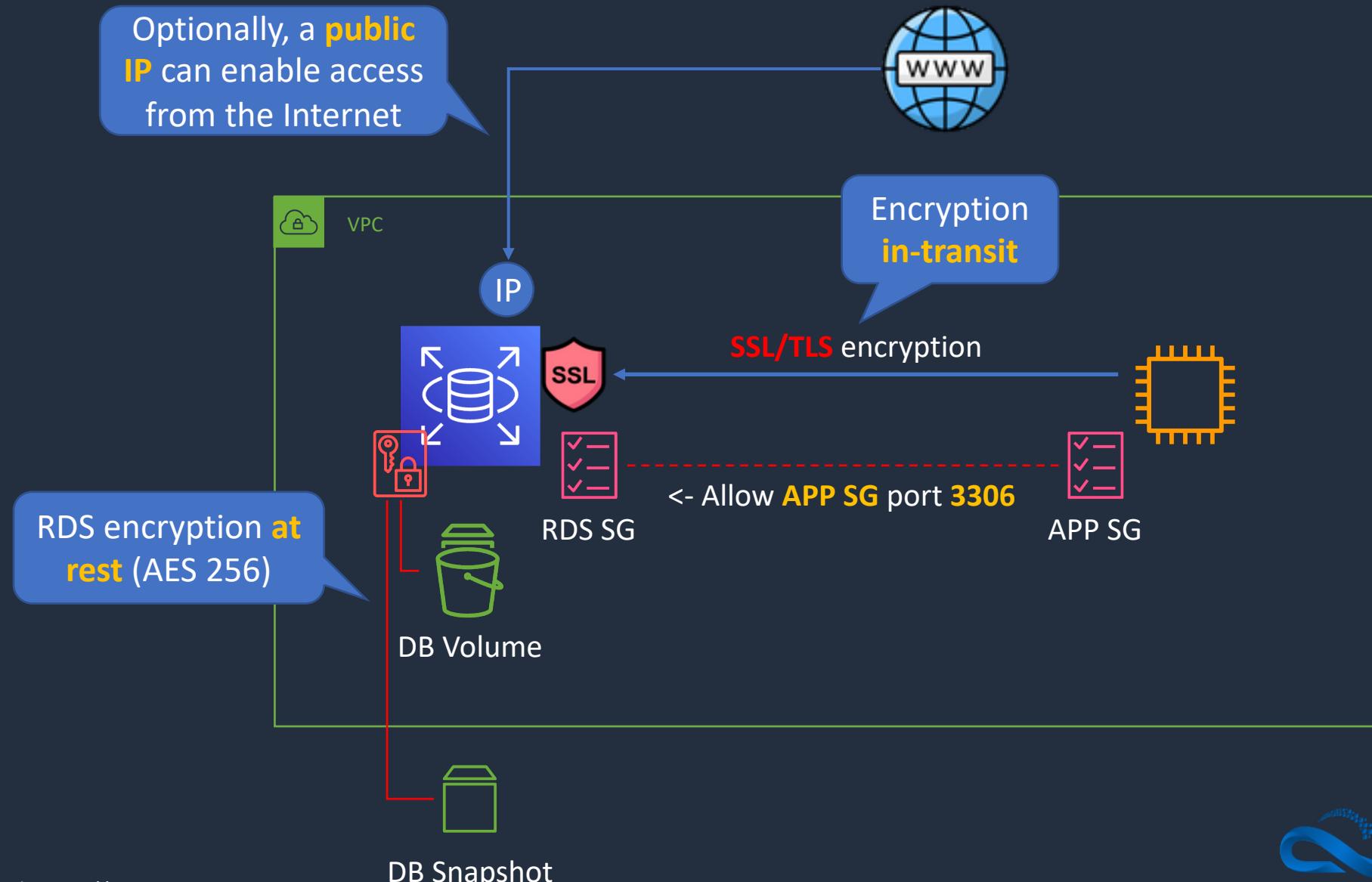
Start day	Start time	Duration
Monday ▾	00 ▾ : 00 ▾ UTC	0.5 ▾ hours

# 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

---

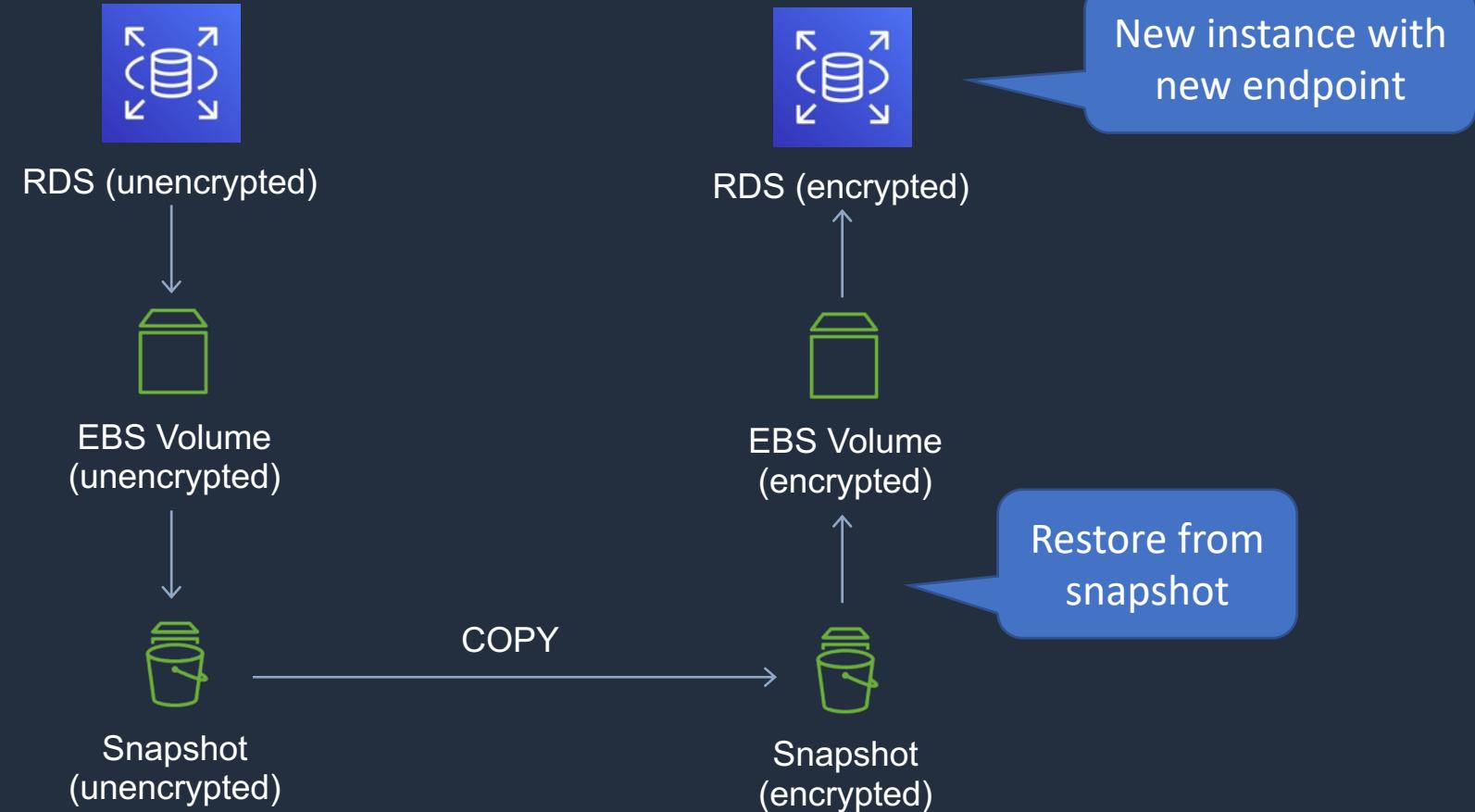
- You can't have an encrypted read replica of an unencrypted DB instance or an unencrypted read replica of an encrypted DB instance
- Read replicas of encrypted master instances are encrypted
- The same key is used if in the same Region as the master
- If the read replica is in a different Region, a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



# Amazon RDS Security



Region



# Amazon Aurora Core Knowledge





# Amazon Aurora Key Features

Aurora Feature	Benefit
<b>High performance and scalability</b>	Offers high performance, self-healing storage that scales up to 64TB, 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



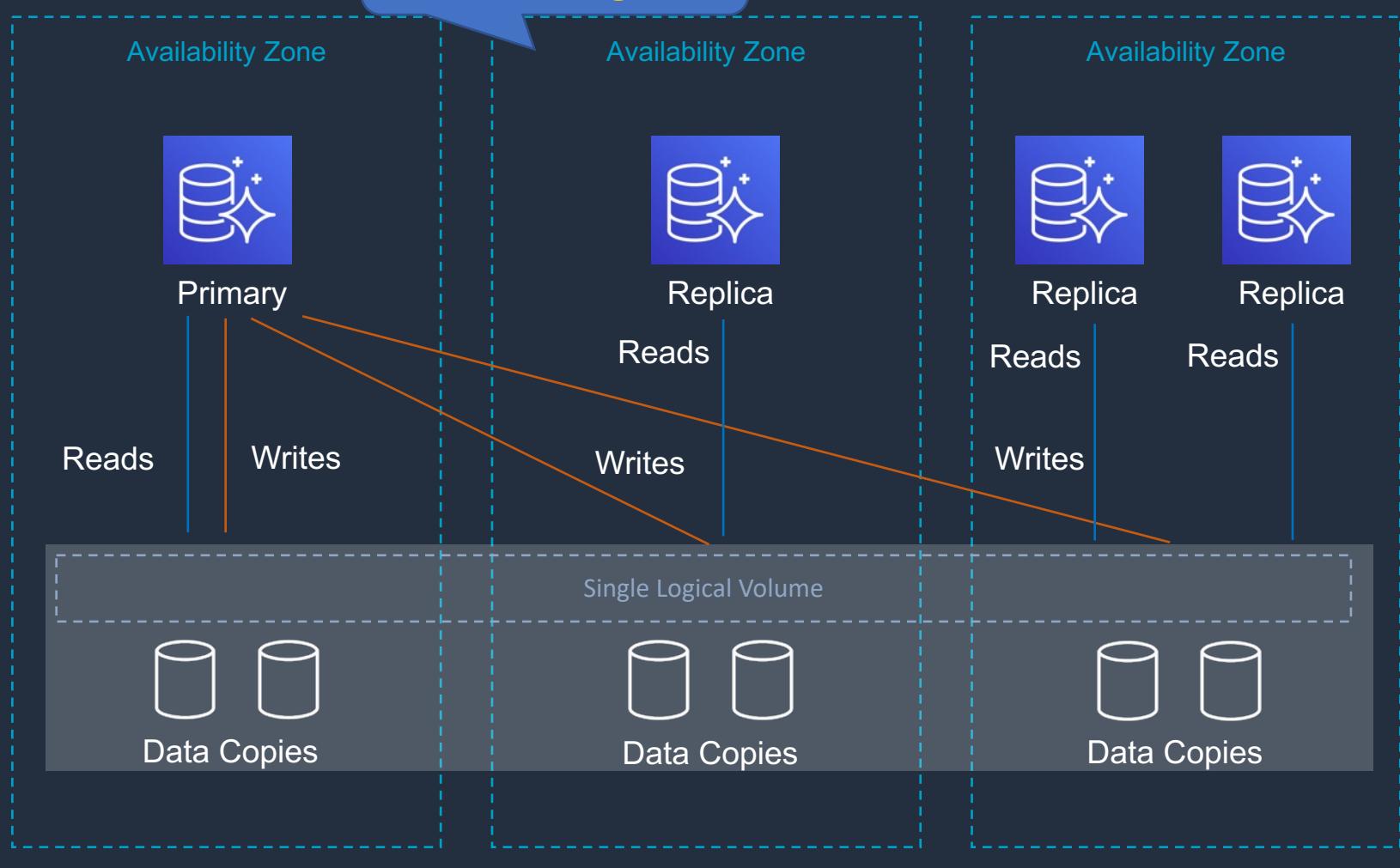


# Aurora Fault Tolerance and Aurora Replicas



Region

Aurora Replicas are  
within a region

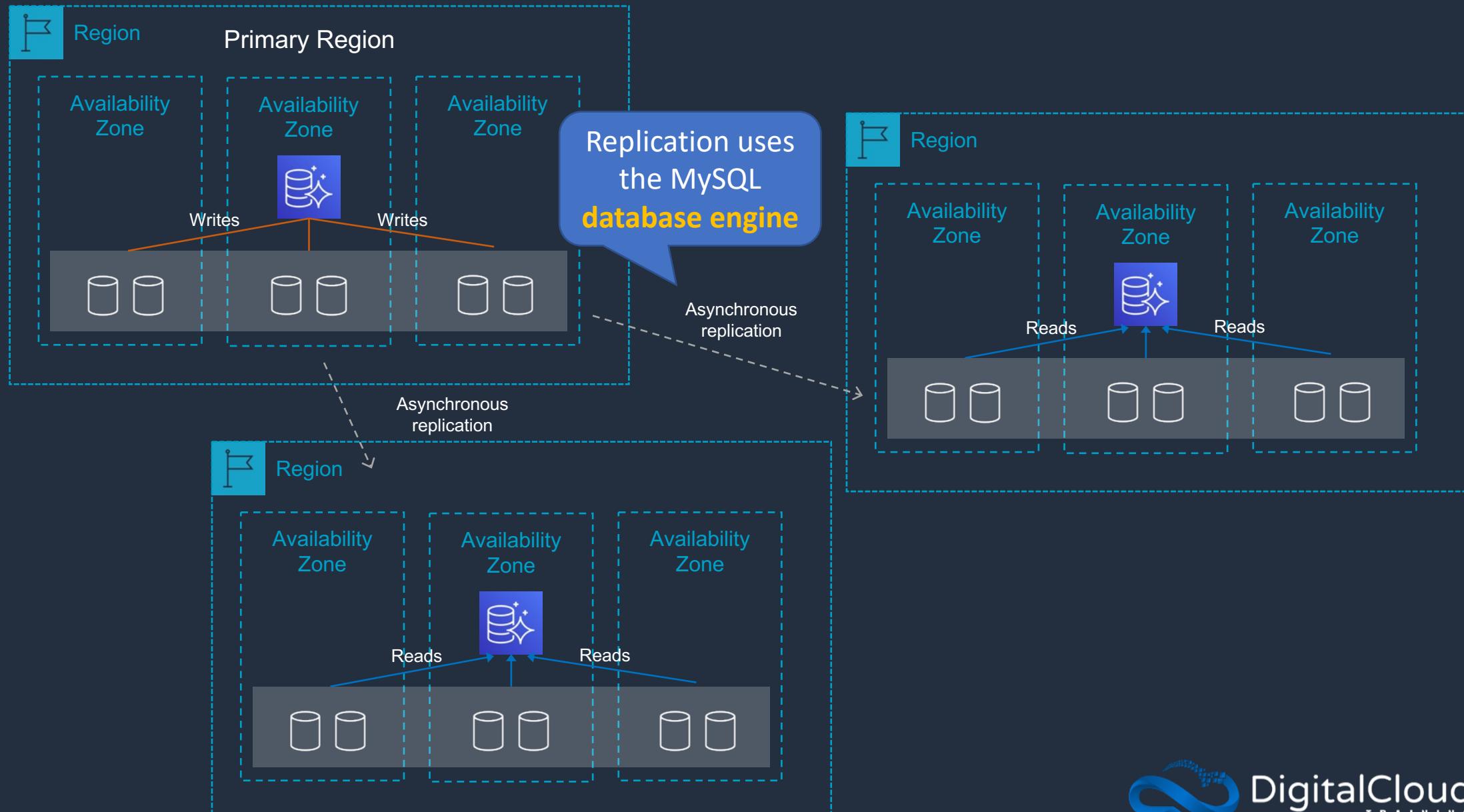


## Aurora Fault Tolerance

- 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

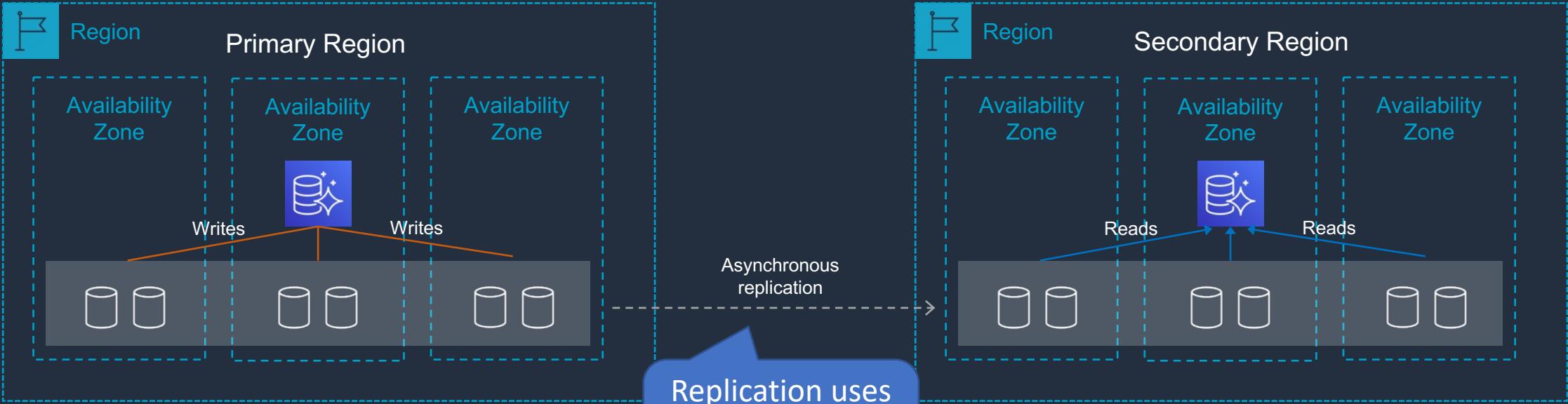
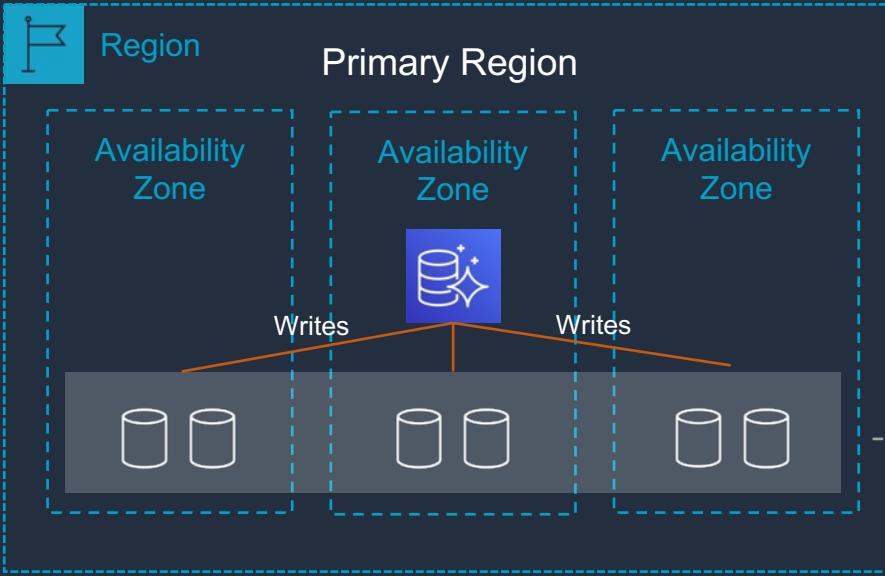


# Cross-Region Replica with Aurora MySQL



# Aurora Global Database

Region 1



Applications can connect  
to the cluster **Reader**  
**Endpoint**

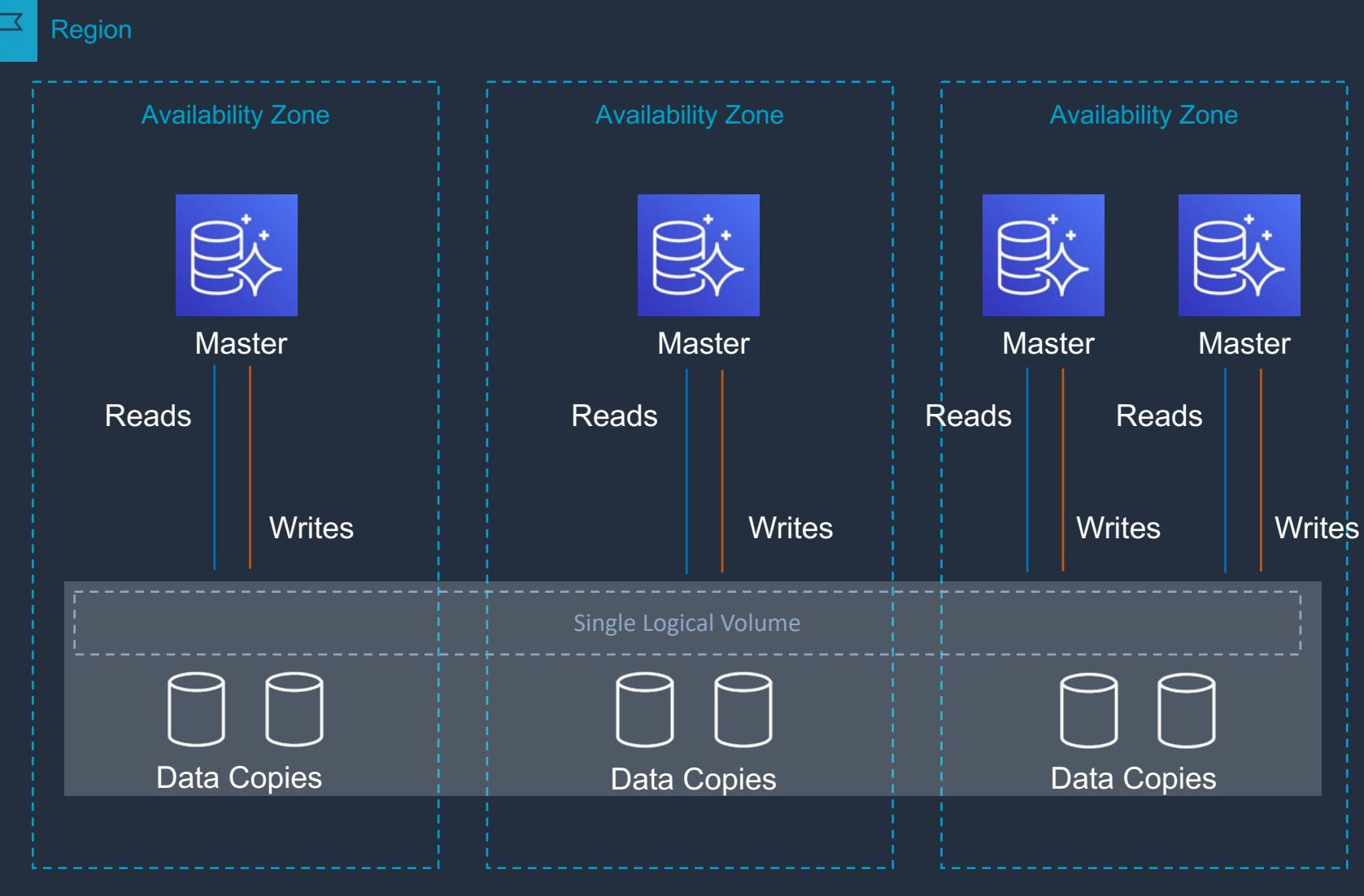
Replication uses  
the **Aurora**  
**storage layer**



# Aurora Multi-Master



Region

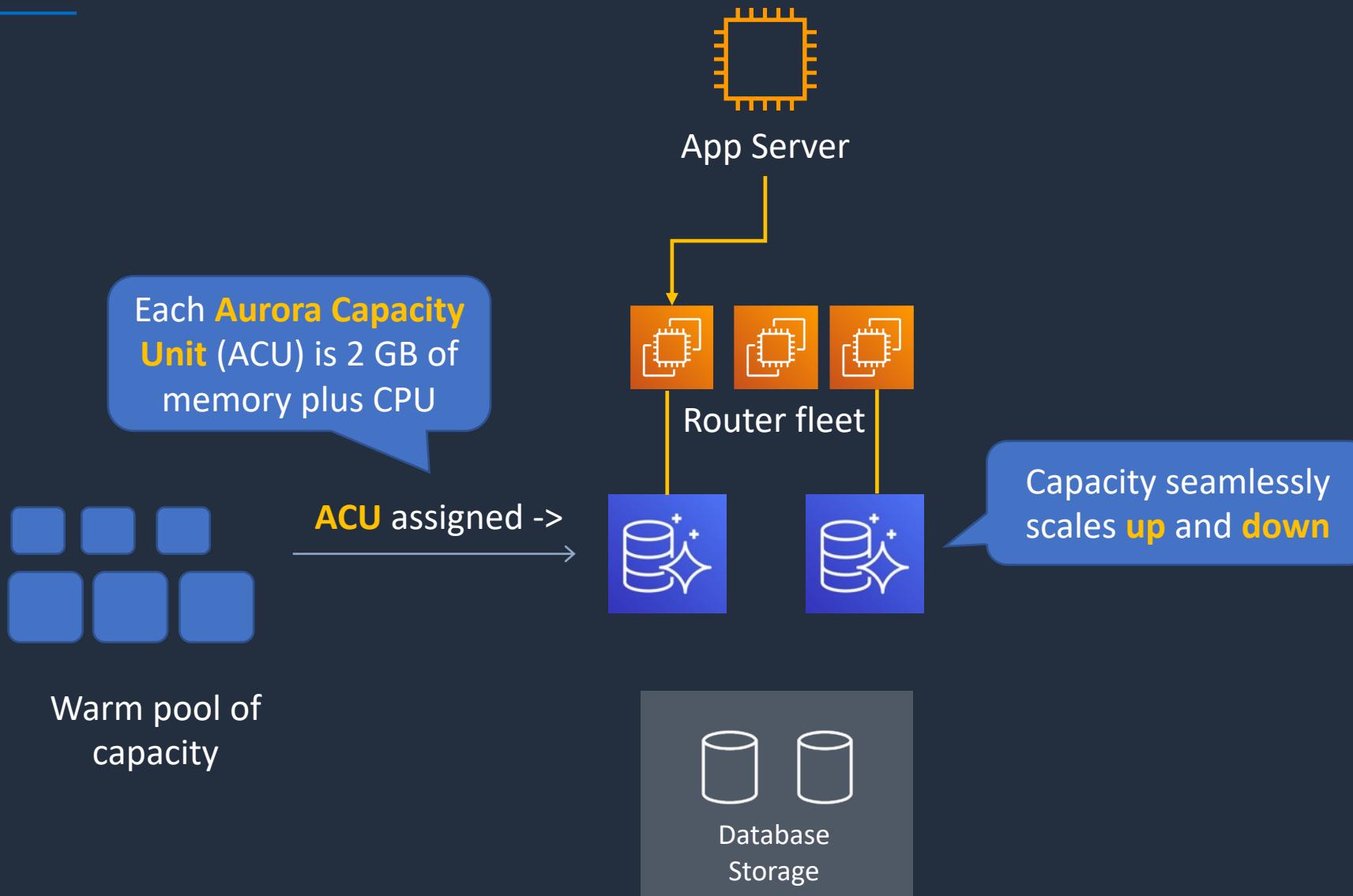


## Aurora Multi-Master

- 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

---

---

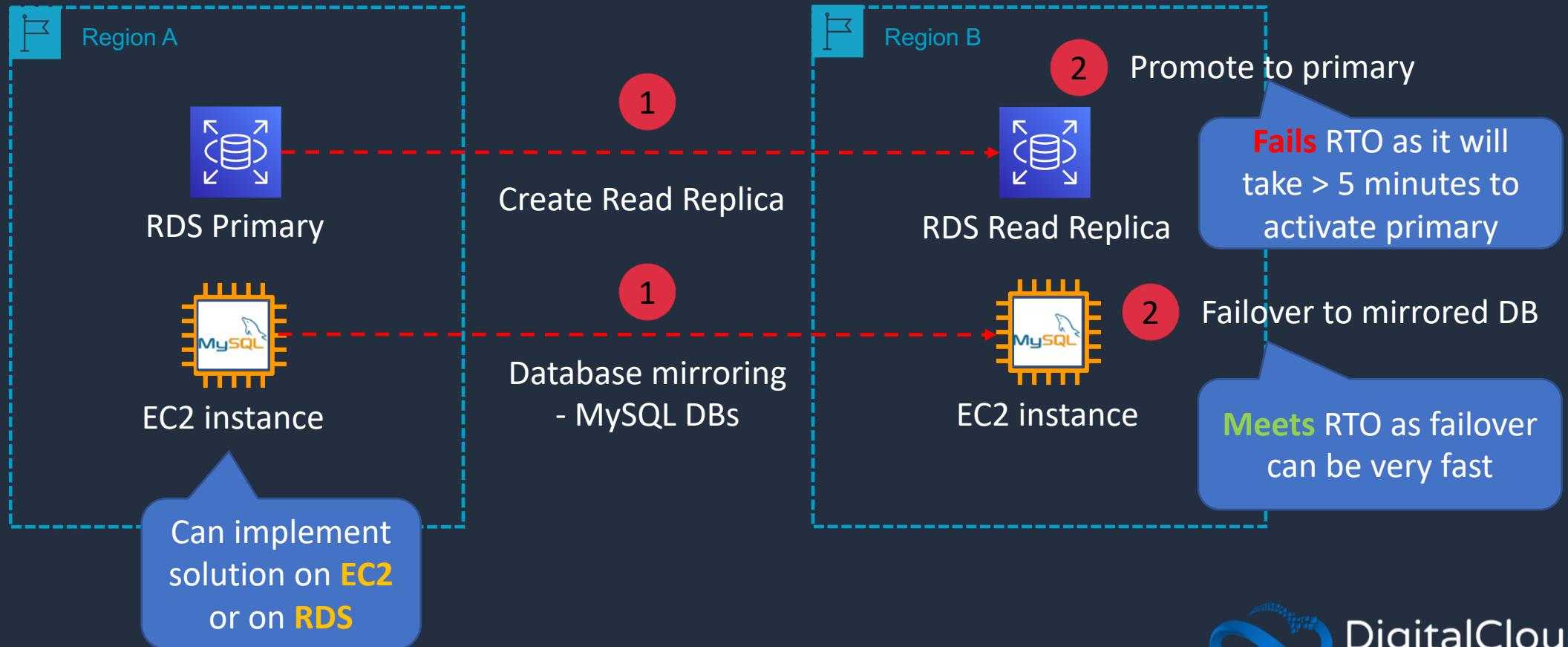
- You can run any database you like with **full control** and **ultimate flexibility**
- You must **manage everything** like backups, redundancy, patching and scaling





# Alternatives to Amazon RDS Replication

Requirement: DR for MySQL DB  
**RPO** = 10 minutes, **RTO** = 5 minutes



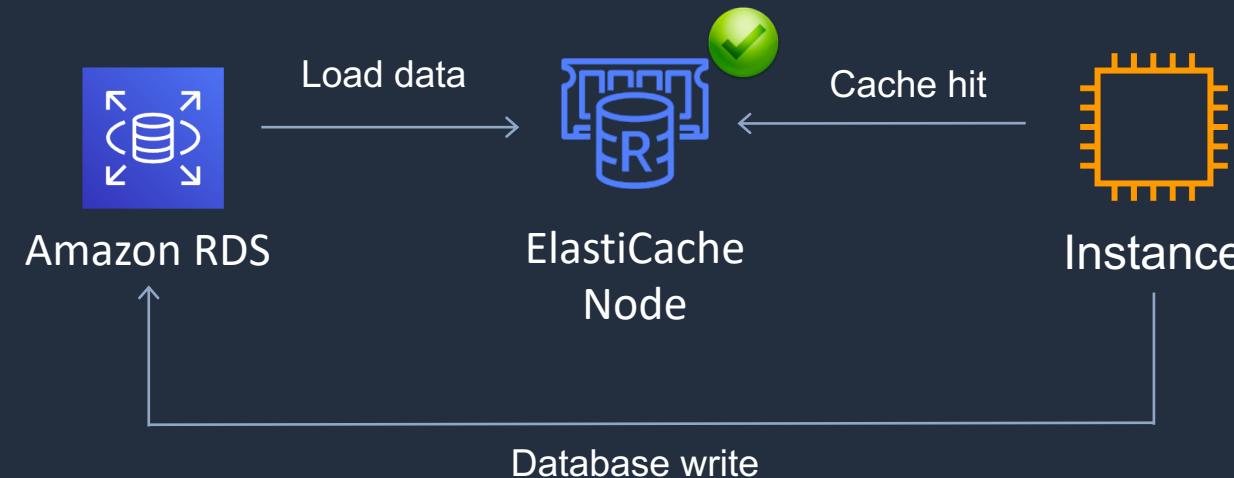
# Amazon ElastiCache Core Knowledge





# Amazon ElastiCache Core Knowledge

- 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





# Amazon ElastiCache Core Knowledge

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

# Scaling ElastiCache





# Amazon ElastiCache - Scalability

## Memcached

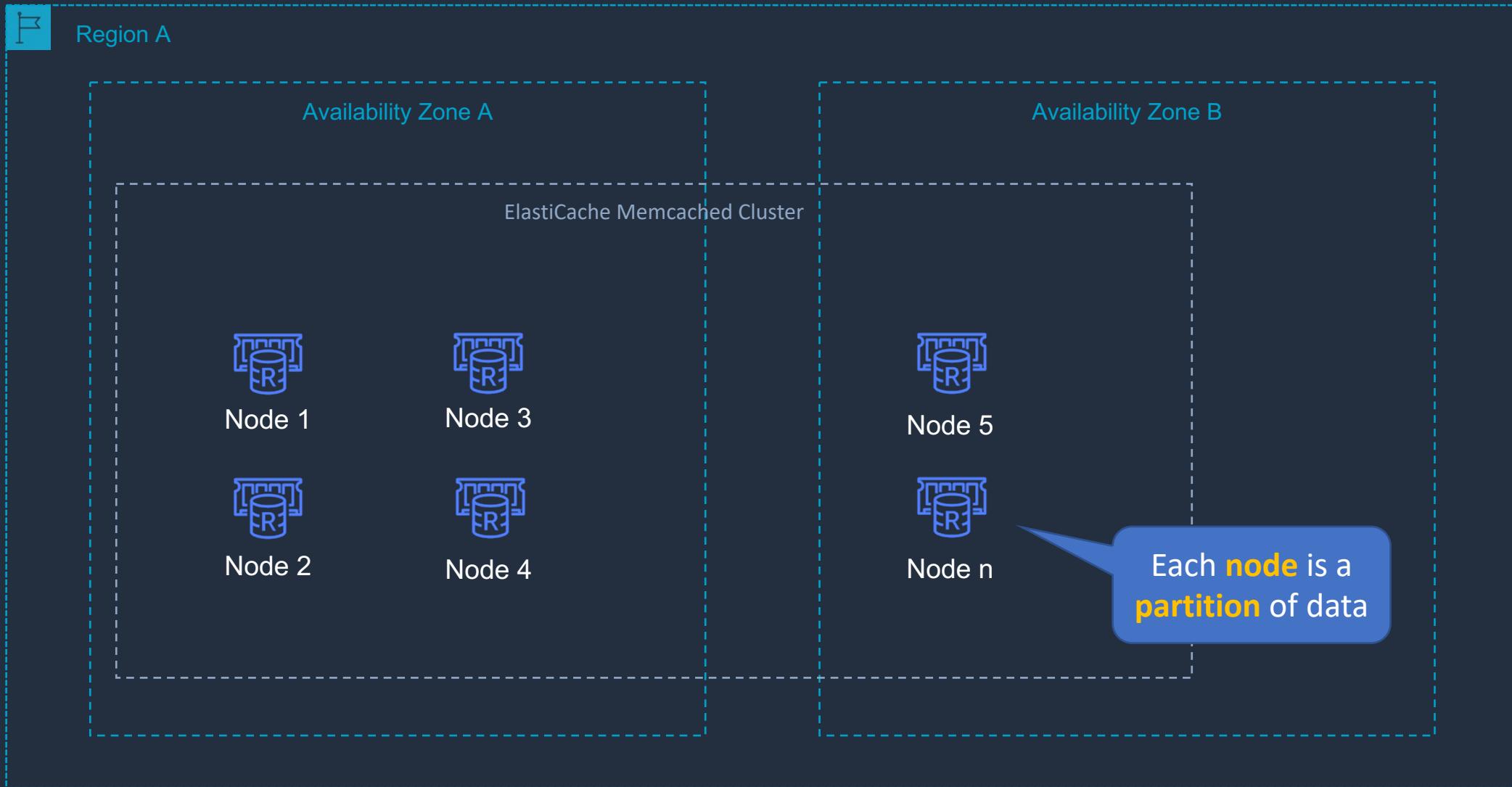
- Add nodes to a cluster
- Scale vertically (node type) – must create a **new cluster** manually

## 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)

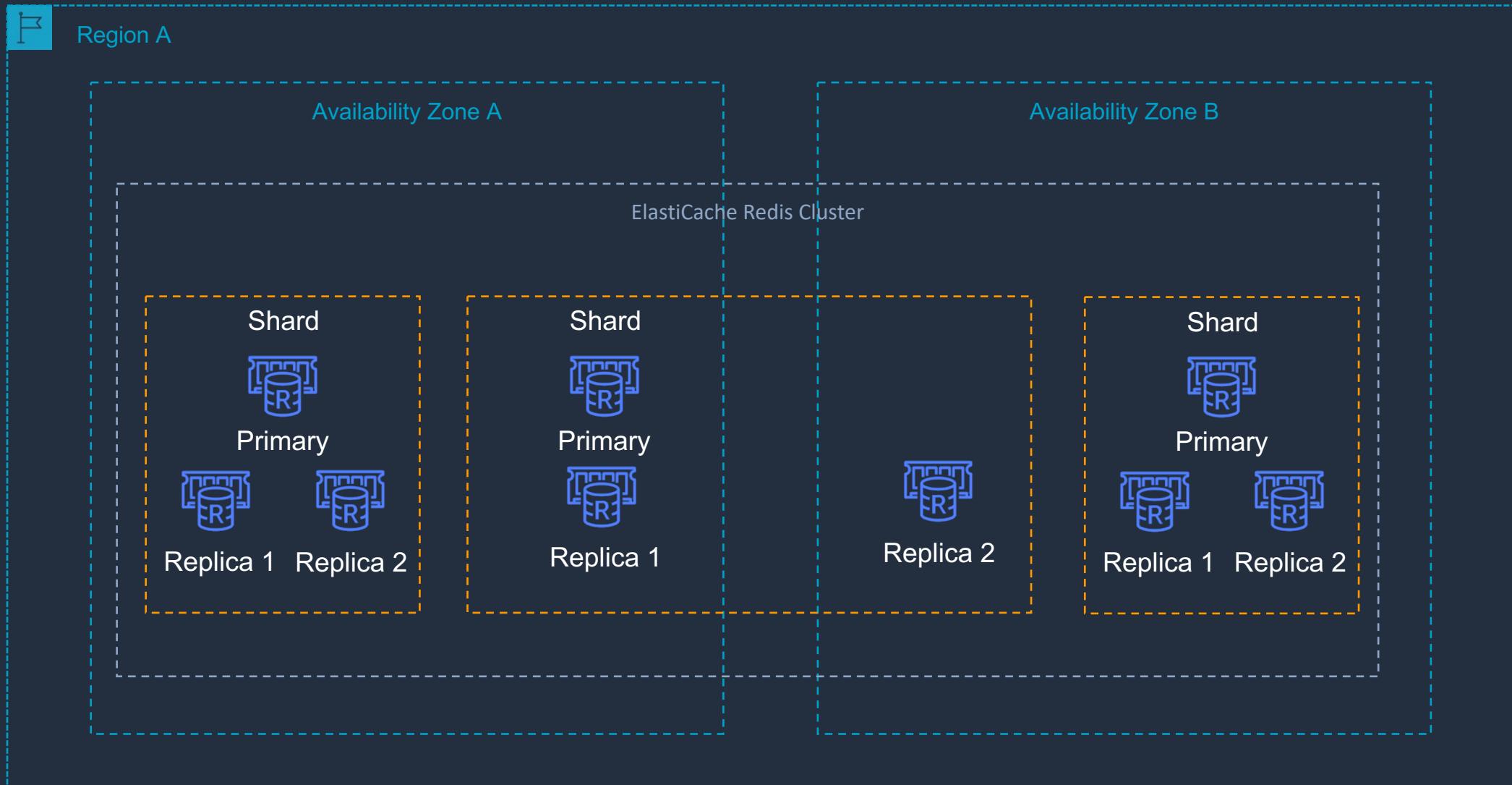


# Amazon ElastiCache Memcached



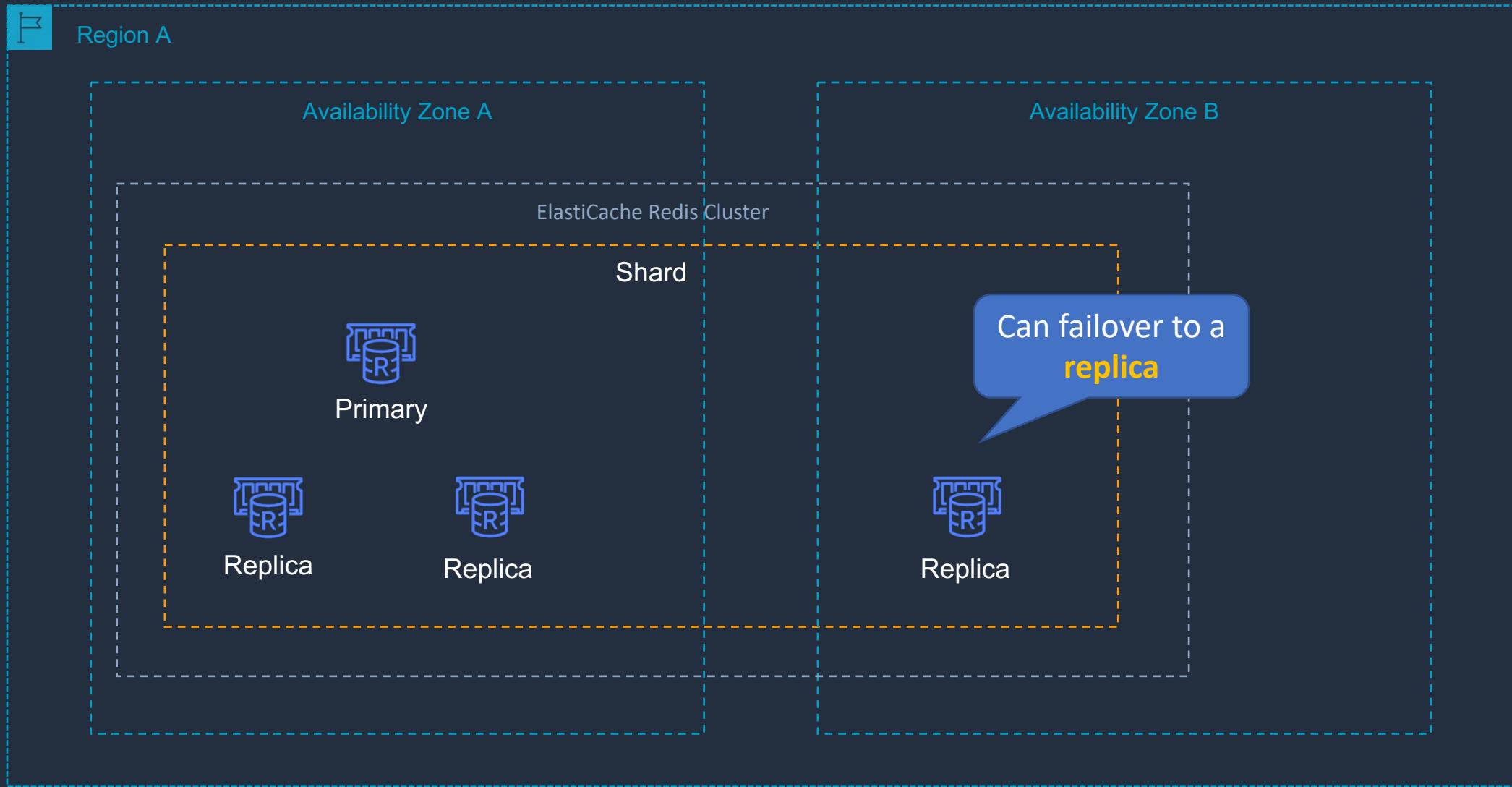


# Amazon ElastiCache Redis (Cluster mode enabled)





# Amazon ElastiCache Redis (Cluster mode disabled)



# Amazon DynamoDB Core Knowledge





# Amazon DynamoDB Core Knowledge

---

- 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 Core Knowledge

---

➤ 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 Core Knowledge

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**

Useful when the data  
structure is **unpredictable**



# Amazon DynamoDB Core Knowledge

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

# DynamoDB Capacity Modes and RCUs/WCUs





# On-Demand and Provisioned Capacity

---

## On-Demand

- Uses **Pay-per-request** pricing for reads and writes

## Provisioned

- Specify number of reads and writes per second (RCU/WCU)
- Can use **AWS Application Auto Scaling** to automatically adjust in response to demand

Note: You can switch between modes once per day



# Read Capacity Units (RCUs)

Read capacity unit (RCU):

- Each API call to read data from your table is a read request
- Read requests can be strongly consistent, eventually consistent, or transactional
- For items up to 4 KB in size, one RCU equals:
  - One strongly consistent read request per second
  - Two eventually consistent read requests per second
  - 0.5 transactional read requests per second
- Items larger than 4 KB require additional RCUs





# Write Capacity Units (WCUs)

---

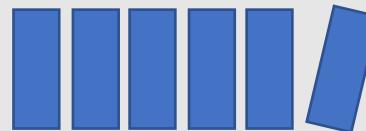
Write capacity unit (WCU):

- Each API call to write data to your table is a write request
- For items up to 1 KB in size, one WCU can perform:
  - One standard write request per second
  - 0.5 transactional writes requests (one transactional write requires two WCUs)
- Items larger than 1 KB require additional WCUs

# Create DynamoDB Table

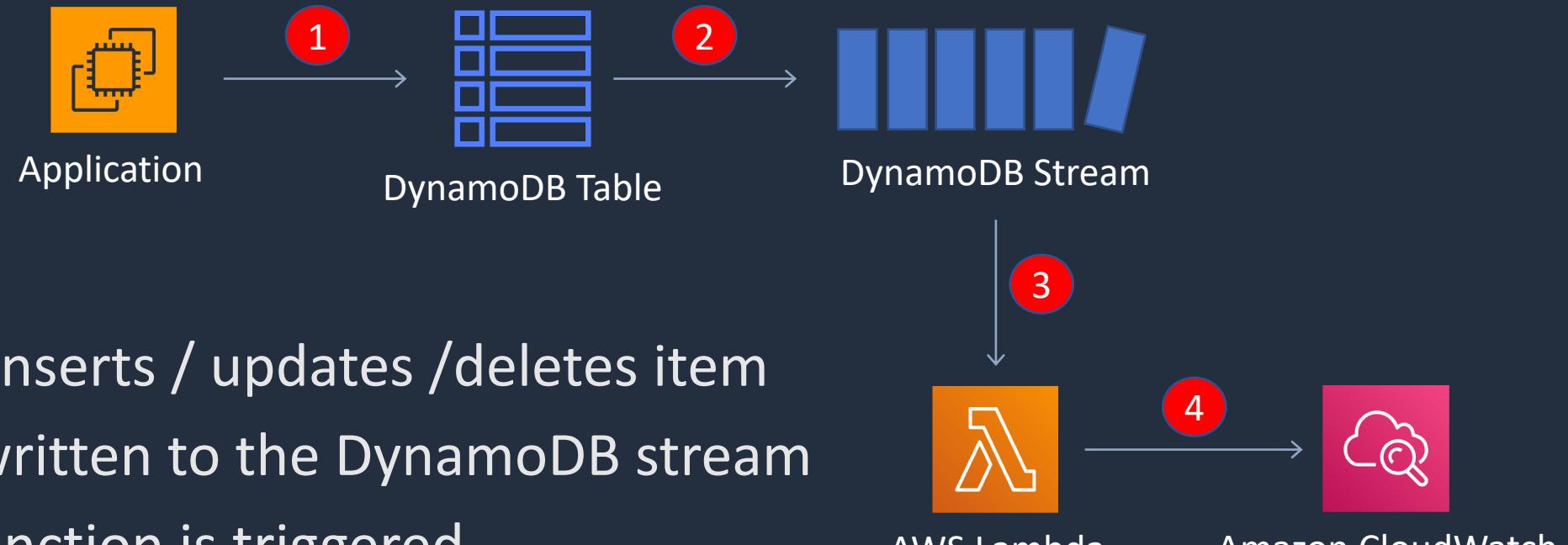


# DynamoDB Streams





# DynamoDB Streams





# DynamoDB Streams

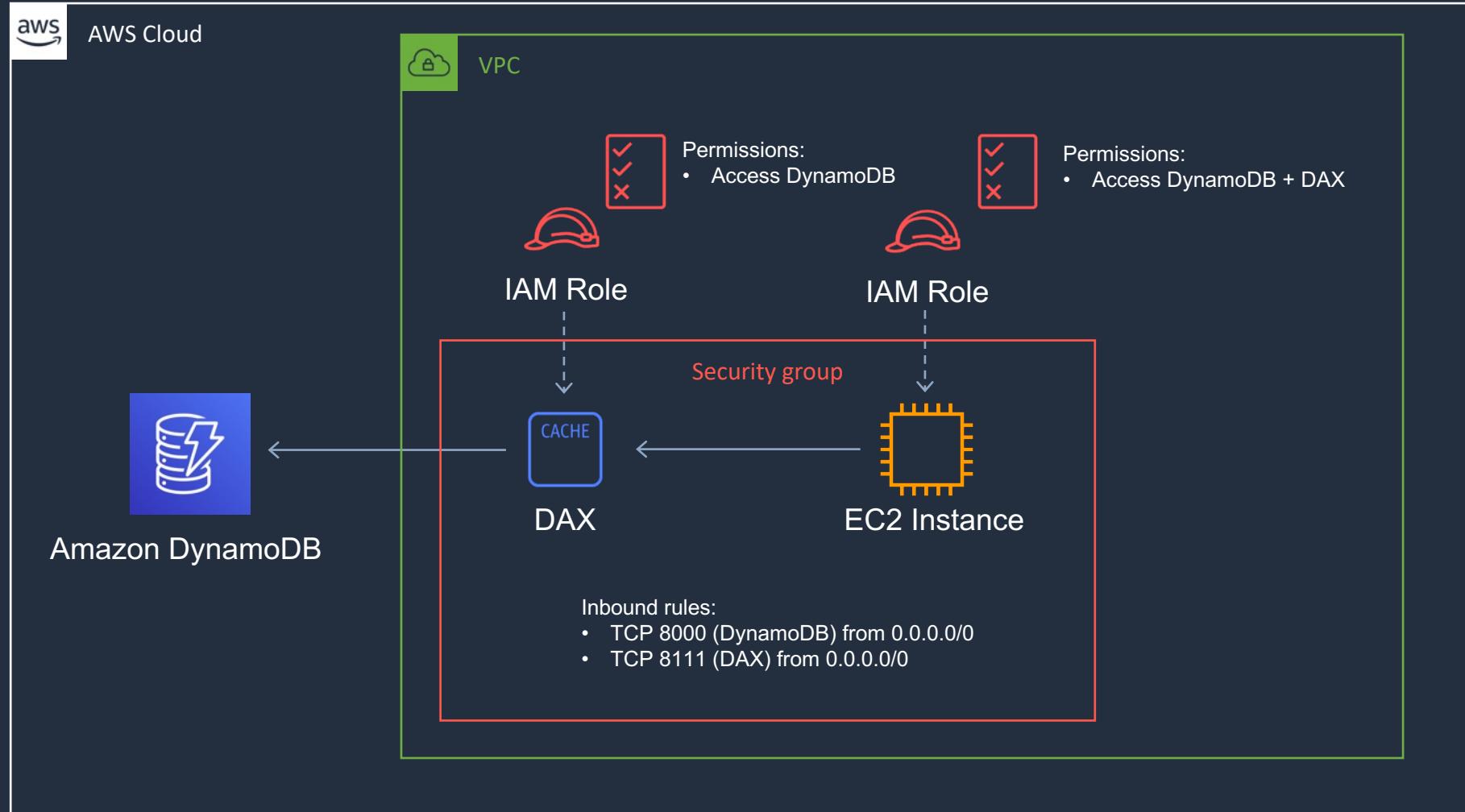
---

- 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)





# DynamoDB Accelerator (DAX)

---

- DAX is a fully managed, highly available, **in-memory** cache for DynamoDB
- Improves performance from milliseconds to microseconds
- DAX is used to improve **READ** performance (not writes)
- You do not need to modify application logic, since DAX is compatible with existing DynamoDB API calls



# DAX vs ElastiCache

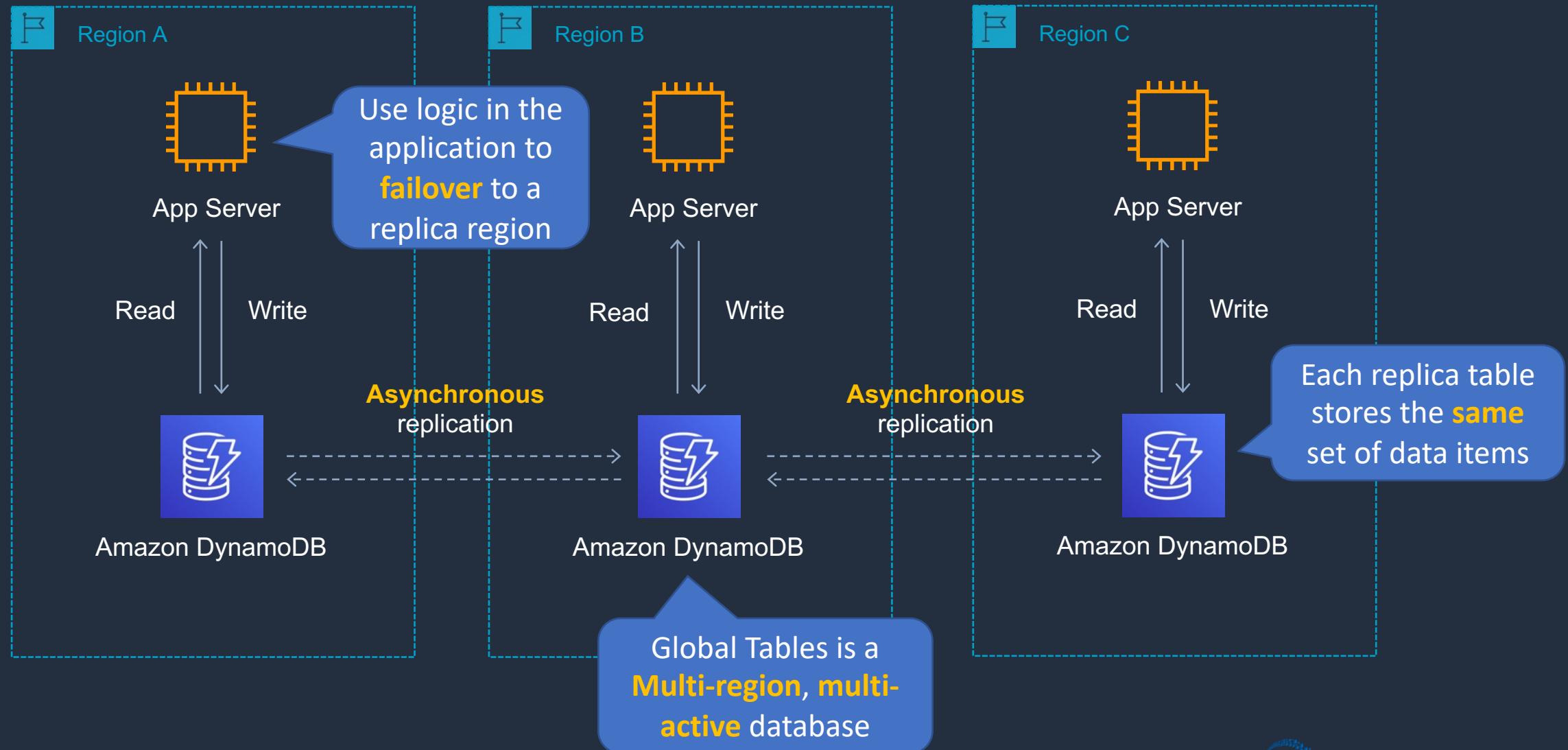
- DAX is **optimized** for DynamoDB
- DAX does not support lazy loading (uses write-through caching)
- With ElastiCache you have more **management overhead** (e.g. invalidation)
- With ElastiCache you need to **modify application code** to point to cache
- ElastiCache supports more datastores

# DynamoDB Global Tables





# DynamoDB Global Tables



# DynamoDB Global Tables



# Architecture Patterns - AWS Databases





# Architecture Patterns - AWS Databases

## Requirement

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

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

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

## Solution

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

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

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



# Architecture Patterns - AWS Databases

## 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 - AWS Databases

---

---

## 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 - AWS Databases

---

---

## Requirement

Multi-threaded in-memory datastore required for unstructured data

## Solution

Use Amazon ElastiCache Memcached

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

Use Amazon DynamoDB DAX (DAX)

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

Use Amazon ElastiCache Redis



# Architecture Patterns - AWS Databases

## 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 - AWS Databases

---

---

## Requirement

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

## Solution

Use DynamoDB Global Tables

# SECTION 11

## Serverless Applications

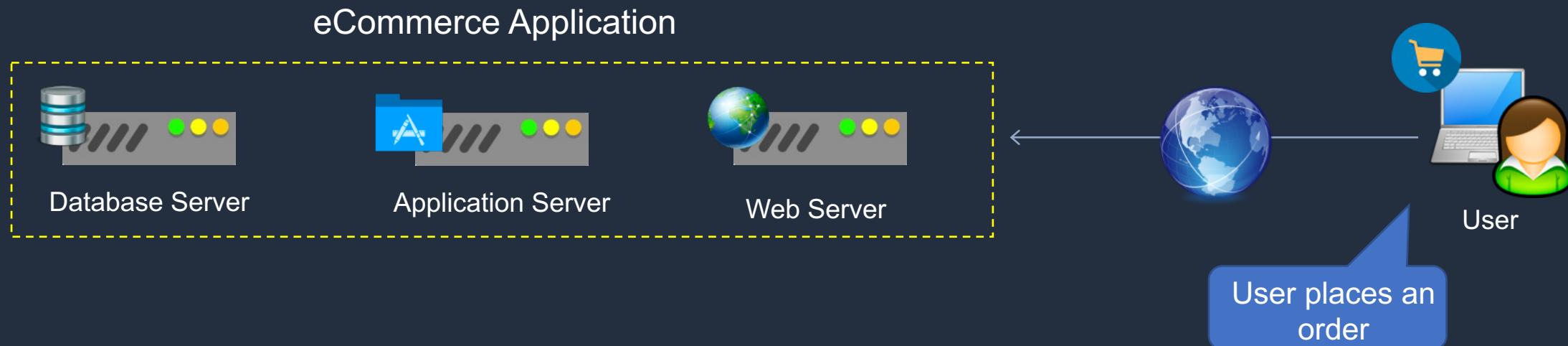
# Event-Driven Architectures





# Event-Driven Architectures

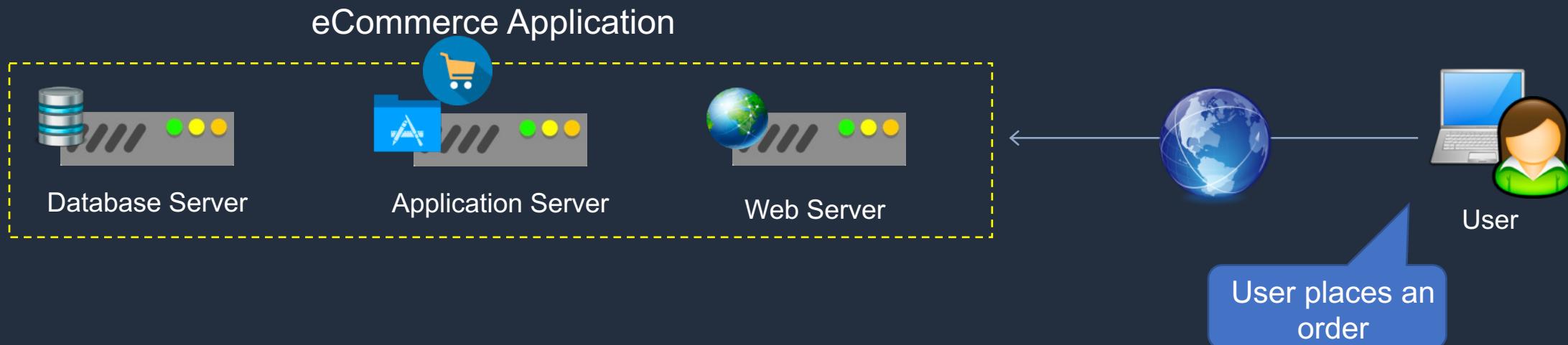
---





# Event-Driven Architectures

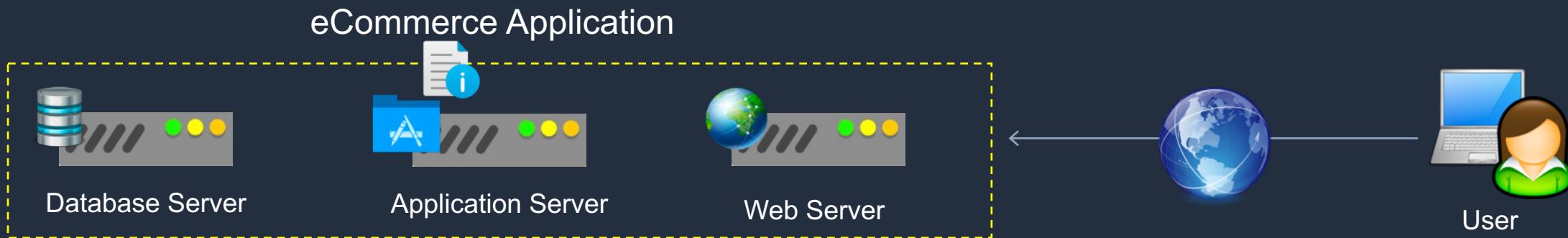
---





# Event-Driven Architectures

---

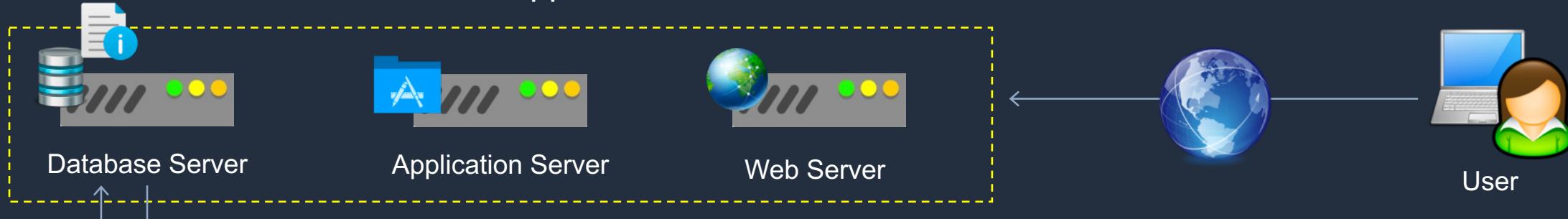


# Event-Driven Architectures



A record of the order is **written** to the **database**

eCommerce Application



Database Server

Application Server

Web Server

User

The event publisher **notifies** an event processor



Event Processor

The event processor **processes** the record from the database



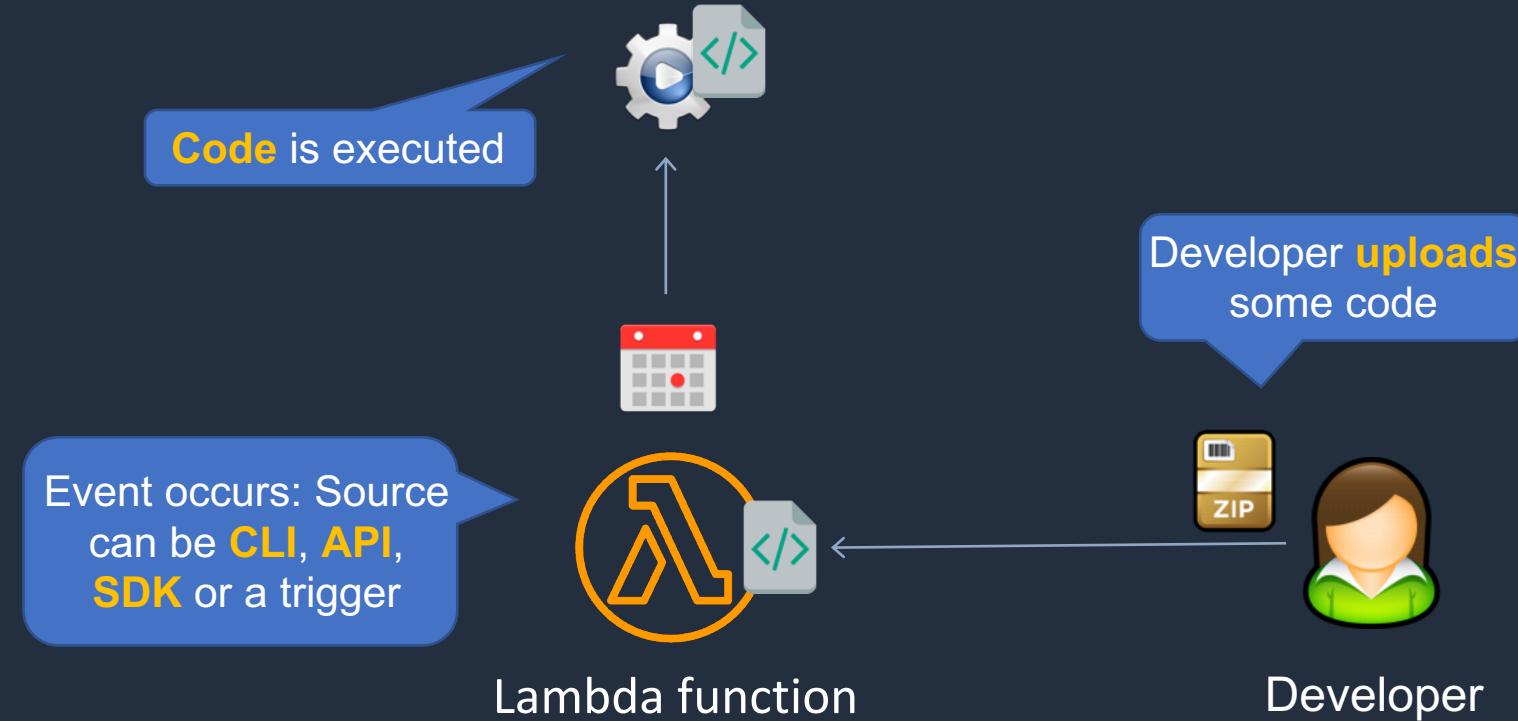
Database Server

The processed data is **written** to a separate **database**

# AWS Lambda Invocations and Concurrency



# AWS Lambda Invocations





# 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



Additional functions are initialized up to the **burst or account limit**

## 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**”



# Lambda Function Concurrency

- Throttling can result in the error: “Rate exceeded” and 429 “TooManyRequestsException”
- If the above error occurs, verify if you see throttling messages in Amazon CloudWatch Logs but no corresponding data points in the Lambda Throttles metrics
- If there are no Lambda Throttles metrics, the throttling is happening on API calls in your Lambda function code
- For asynchronous invocations Lambda retries up to 3 times then goes to a Dead Letter Queue
- DLQ can be SNS topic or SQS queue



# Lambda Function Concurrency

Methods to resolve throttling include:

- Configure reserved concurrency
- Use exponential backoff in your application code

Concurrency metrics:

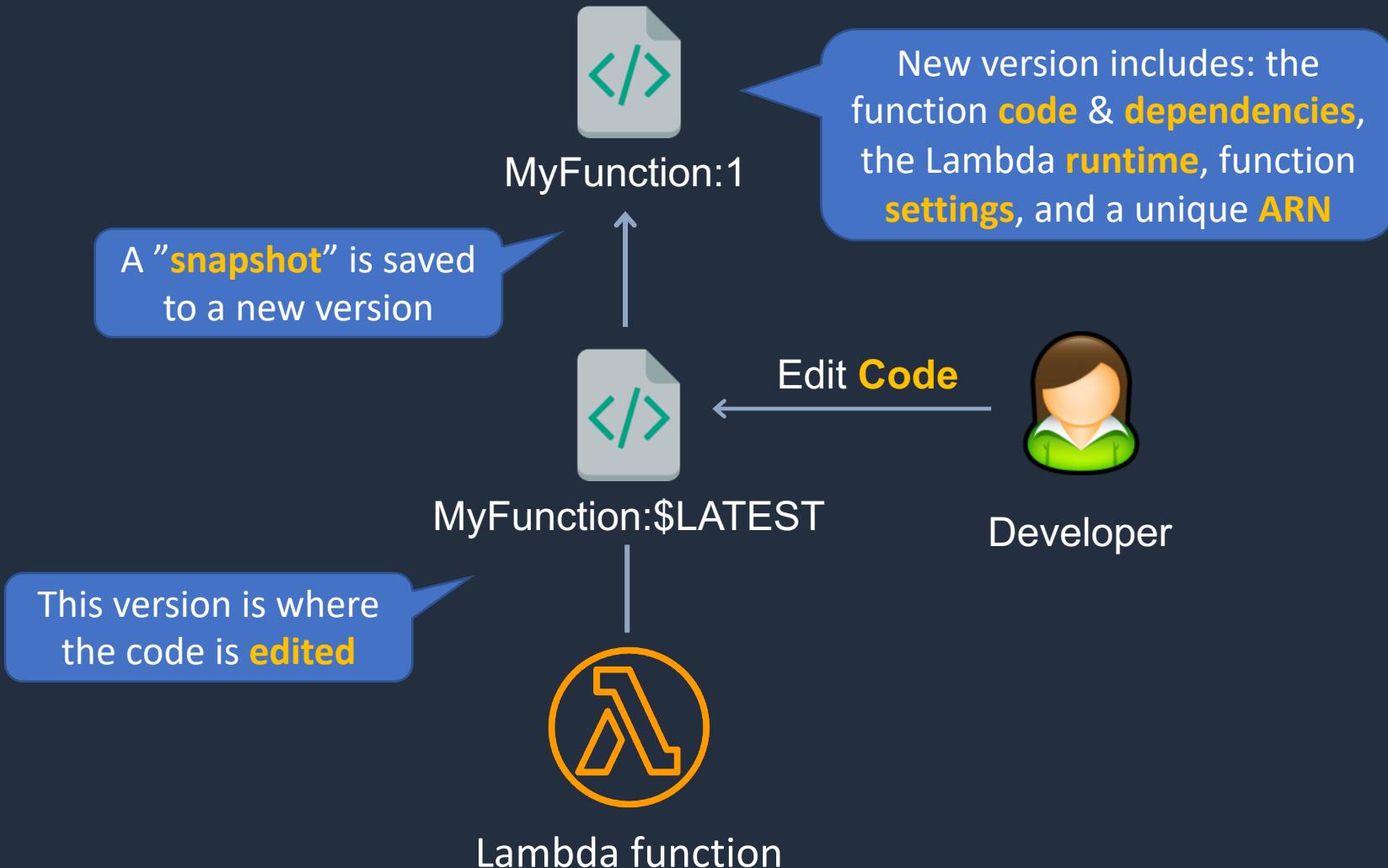
- ConcurrentExecutions
- UnreservedConcurrentExecutions
- ProvisionedConcurrentExecutions
- ProvisionedConcurrencyInvocations
- ProvisionedConcurrencySpilloverInvocations
- ProvisionedConcurrencyUtilization

# Lambda Versions and Aliases





# Lambda Versions





# Lambda Versions

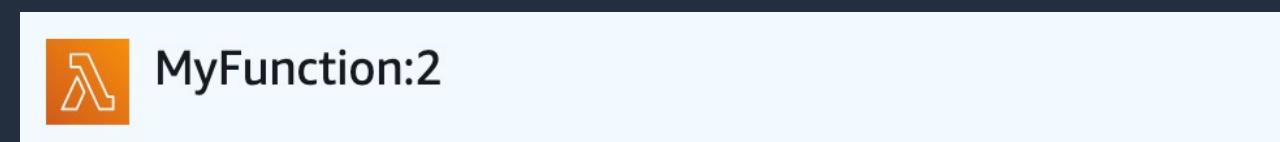
- You work on \$LATEST which is the latest version of the code - this is mutable (changeable)



- When you're ready to publish a Lambda function you create a version (these are numbered)



- Numbered versions are assigned a number starting with 1 and subsequent versions are incremented by 1



- Versions are immutable (code cannot be edited)



# Lambda Versions

- Each version has its own ARN

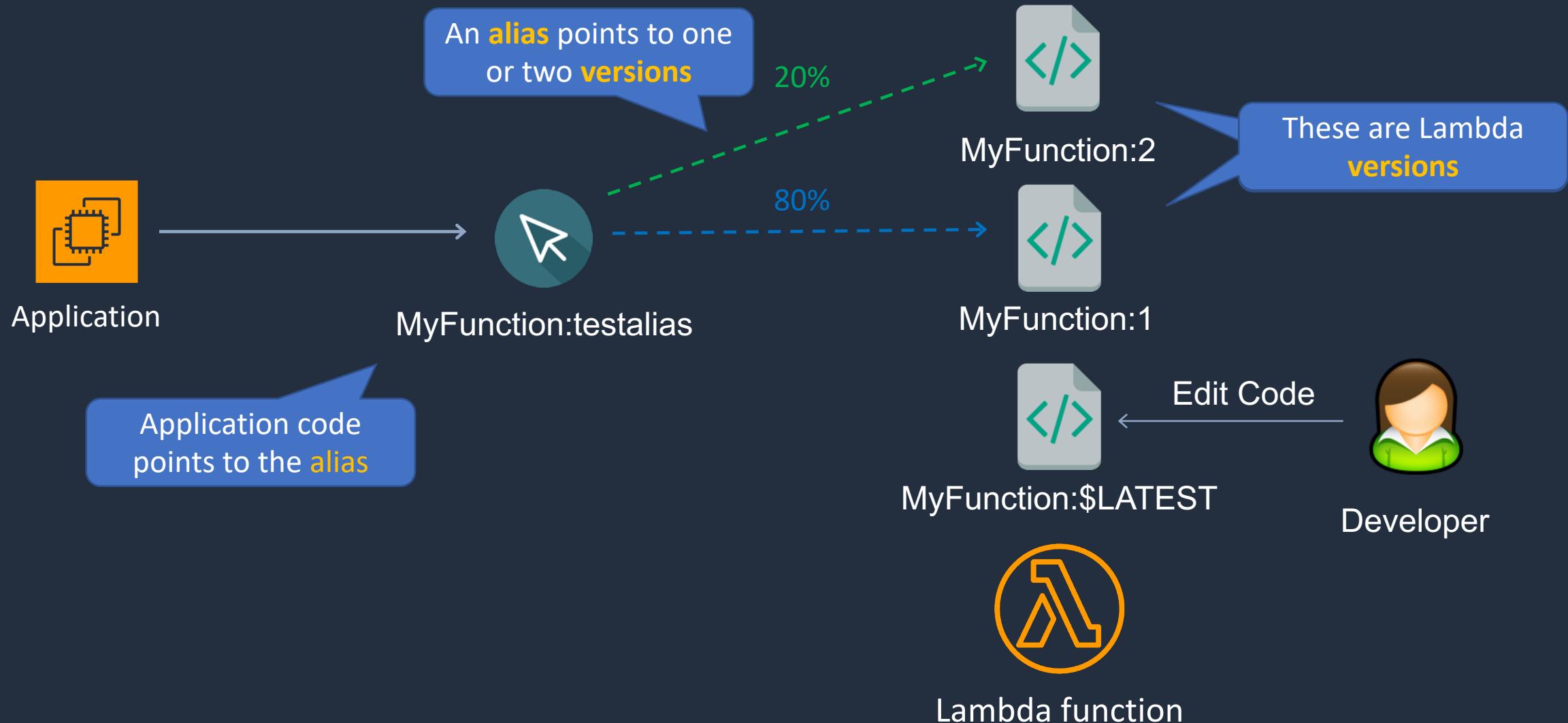
**ARN - arn:aws:lambda:ap-southeast-2:515148227241:function:MyFunction:\$LATEST**

**ARN - arn:aws:lambda:ap-southeast-2:515148227241:function:MyFunction:1**

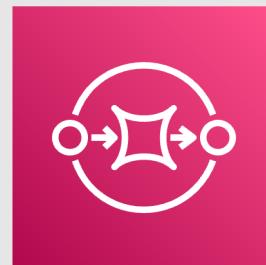
- Because different versions have unique ARNs this allows you to effectively manage them for different environments like Production, Staging or Development
- A qualified ARN has a version suffix
- An unqualified ARN does not have a version suffix
- You cannot create an alias from an unqualified ARN



# Lambda Aliases



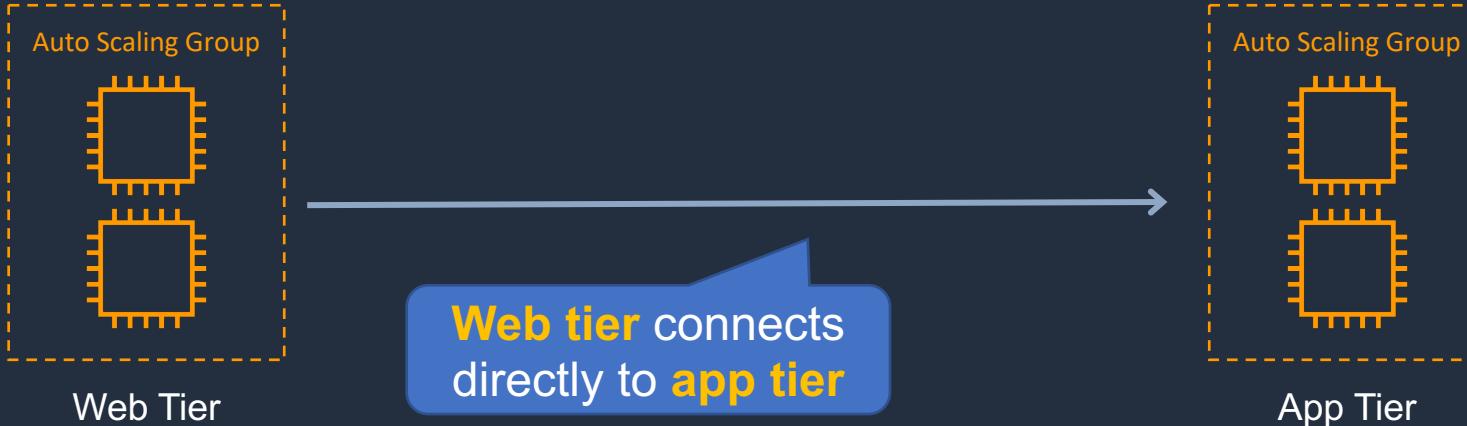
# Advanced Amazon SQS





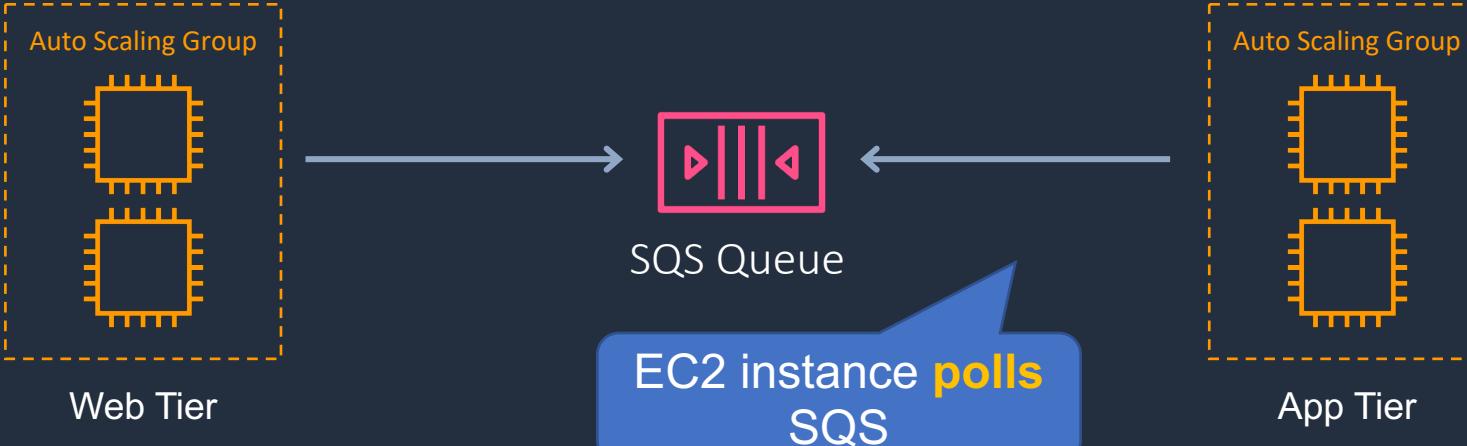
# Decoupling with SQS Queues

## Direct integration



**App tier** must keep up with workload or **failure** will occur

## Decoupled integration

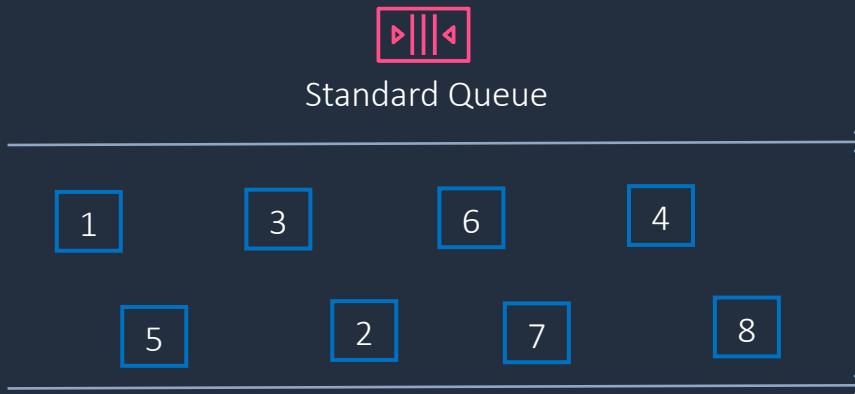




# SQS Queue Types

---

---



**Best-effort** ordering



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



# SQS Queue Types

Standard Queue	FIFO Queue
<b>Unlimited Throughput:</b> Standard queues support a nearly unlimited number of transactions per second (TPS) per API action.	<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
<b>Best-Effort Ordering:</b> Occasionally, messages might be delivered in an order different from which they were sent	<b>First-In-First-out Delivery:</b> The order in which messages are sent and received is strictly preserved
<b>At-Least-Once Delivery:</b> A message is delivered at least once, but occasionally more than one copy of a message is delivered	<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

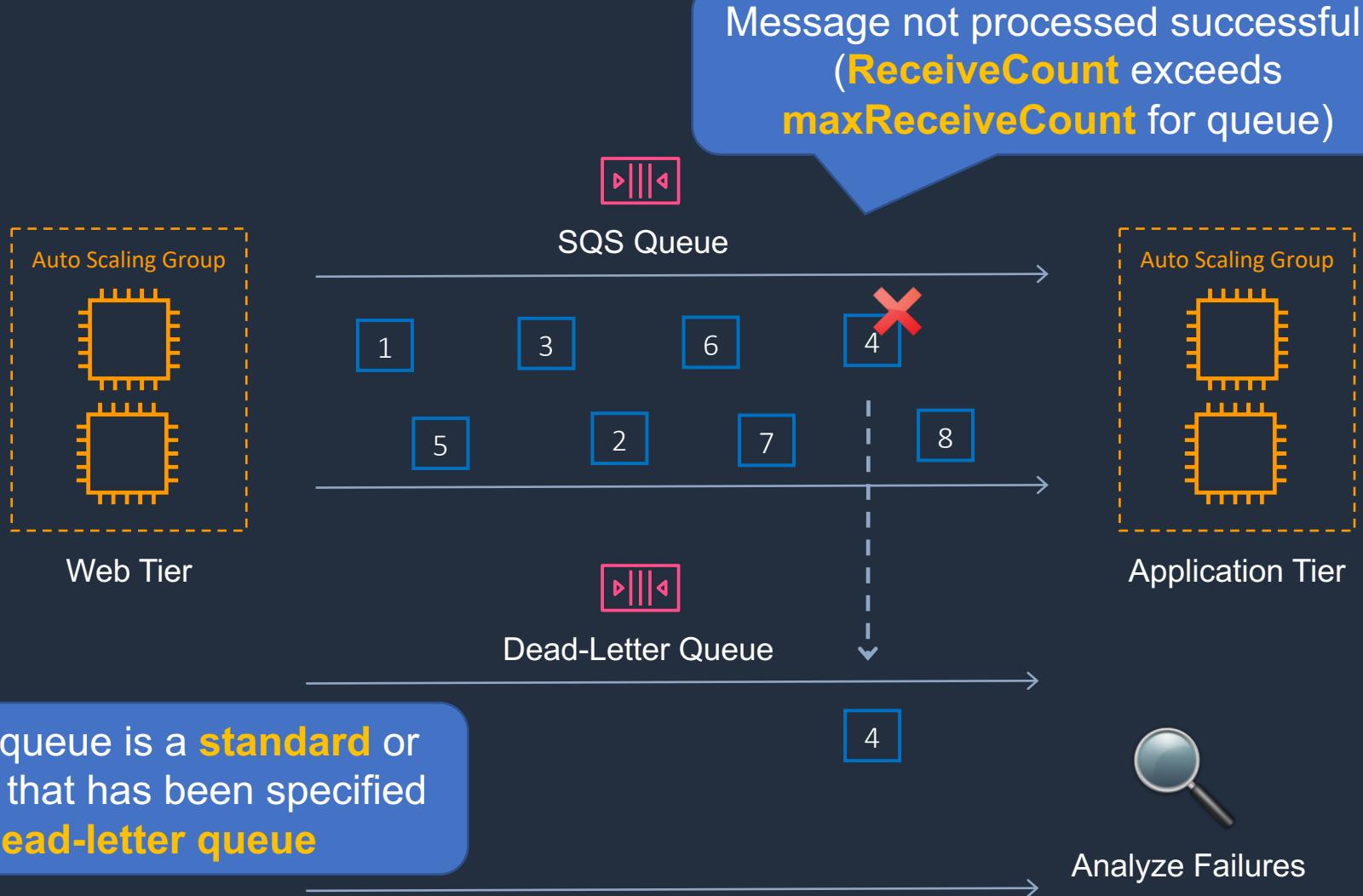


# SQS Queue Types

---

- 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

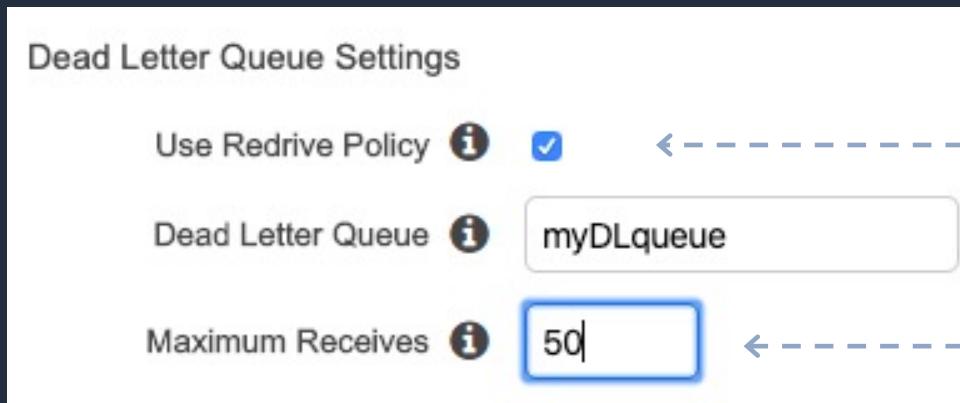




# SQS – Dead Letter Queue

---

- 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



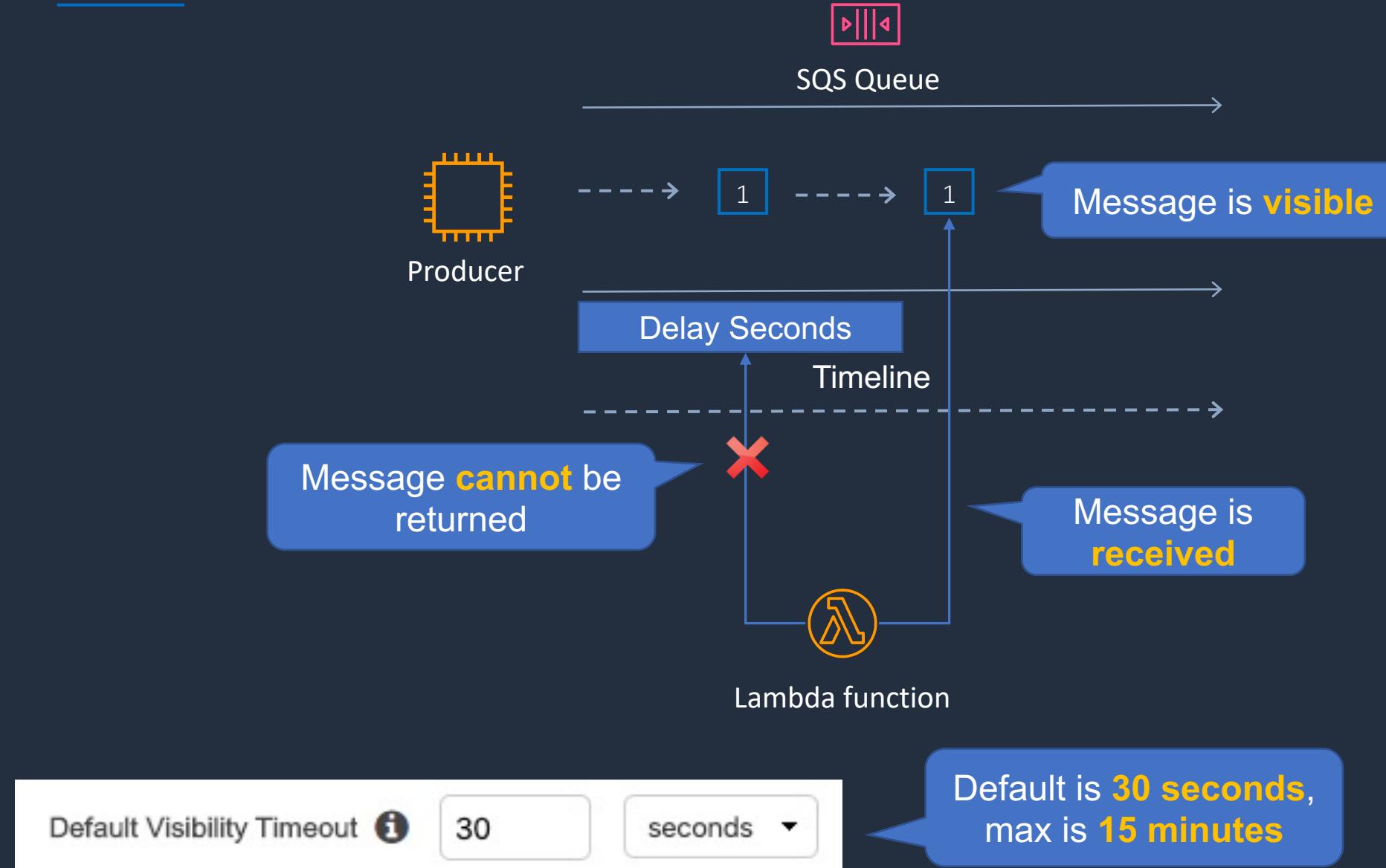
Enable Redrive Policy

Specify the queue to use as a dead-letter queue

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

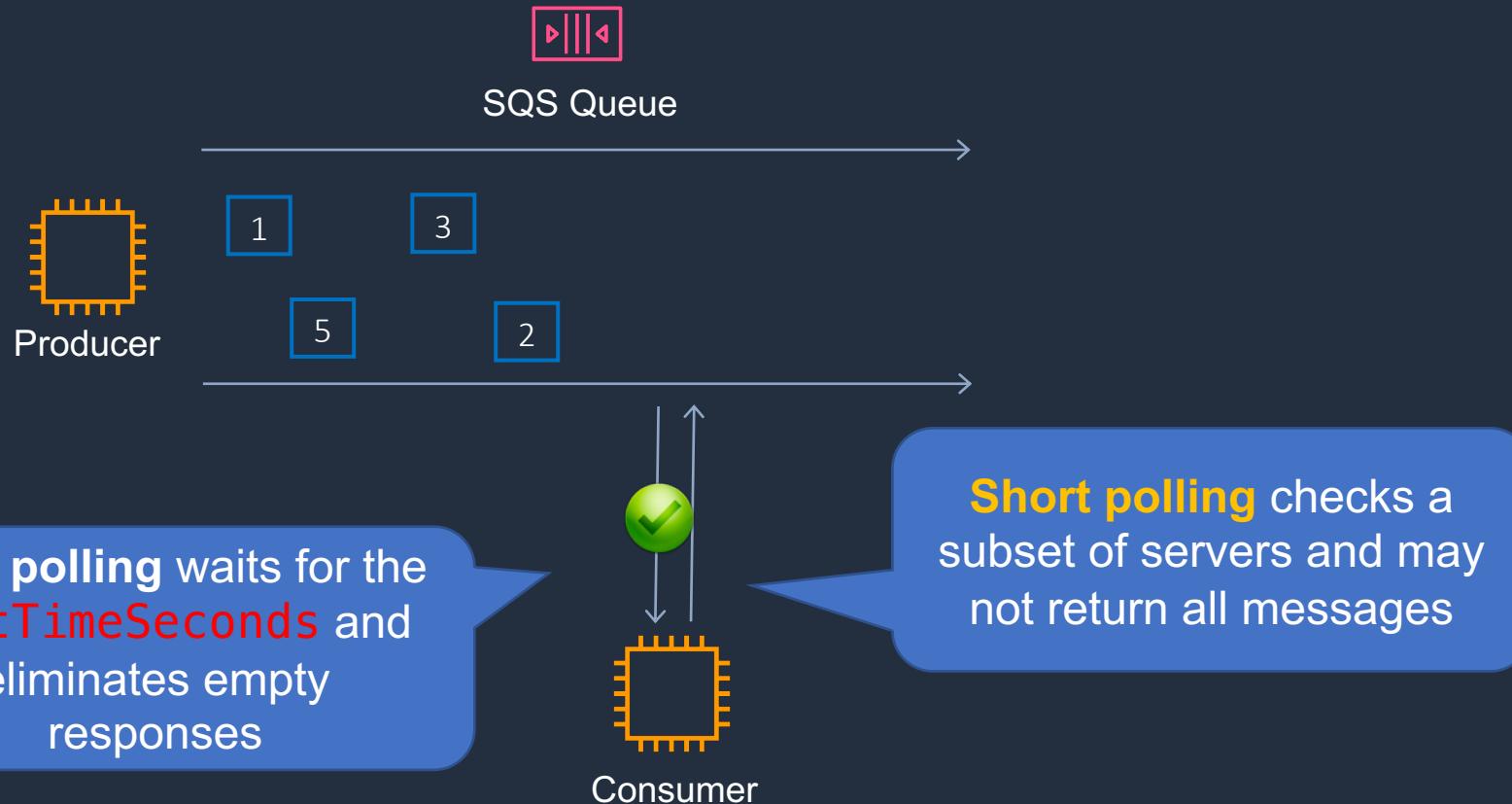


# SQS – Delay Queue





# SQS Long Polling vs Short Polling





# SQS Long Polling vs Short Polling

---

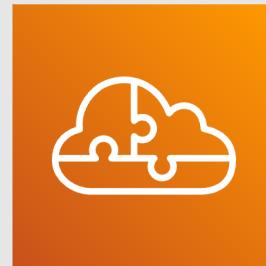
- SQS **Long polling** is a way to retrieve messages from SQS queues – waits for messages to arrive
- SQS **Short polling** returns immediately (even if the message queue is empty)
- SQS Long polling can lower costs
- SQS Long polling can be enabled at the queue level or at the API level using **WaitTimeSeconds**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

Receive Message Wait Time 20 seconds

←-----

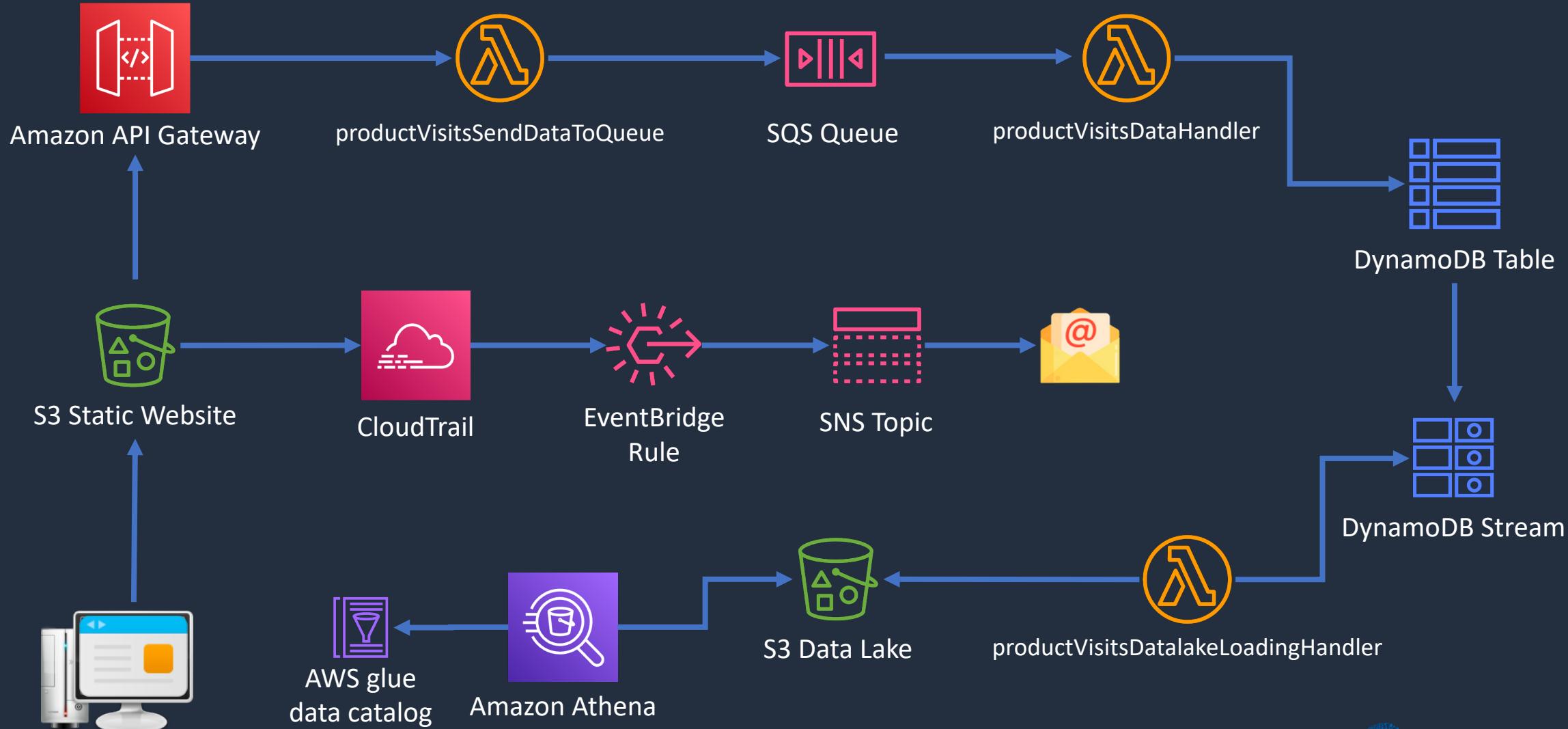
The maximum amount of time that a long polling receive call will wait for a message to become available before returning an empty response.

# Serverless App Architecture for HOL





# Serverless App Architecture



# Build a Serverless App – Part 1



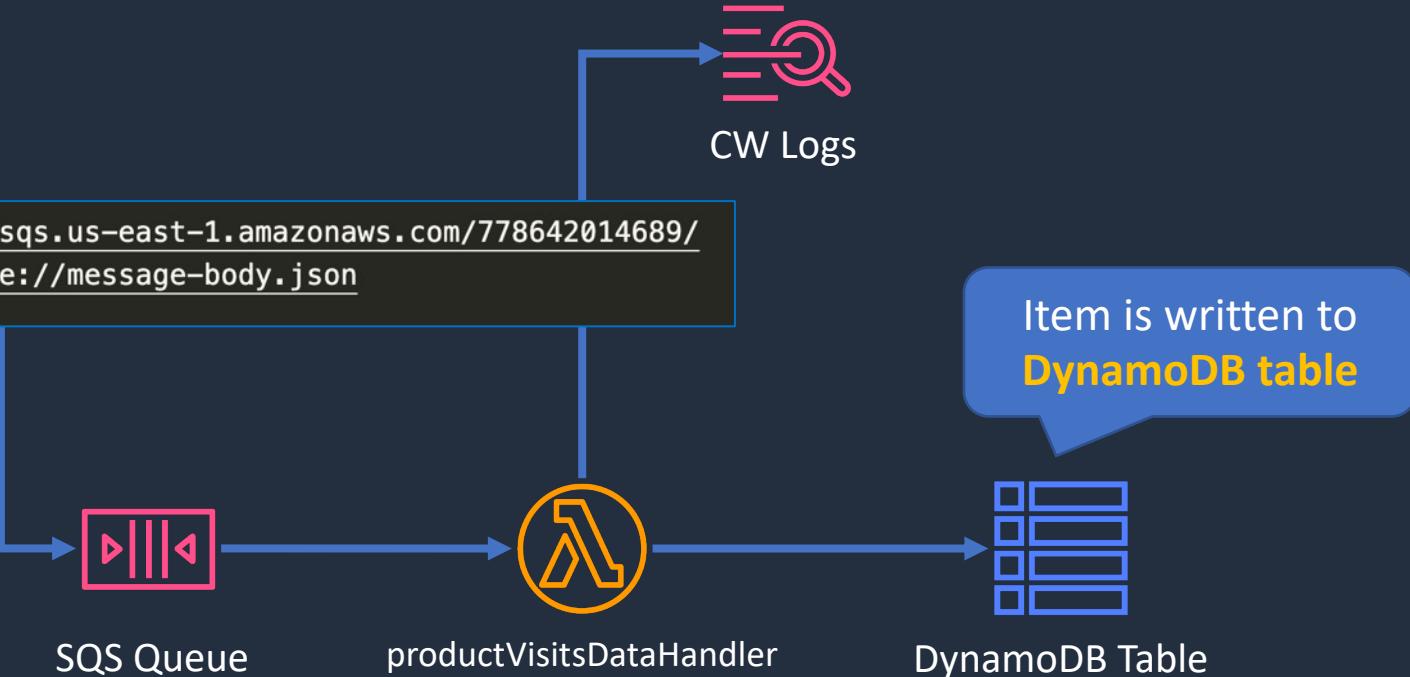


# Build a Serverless App - Part 1



```
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**

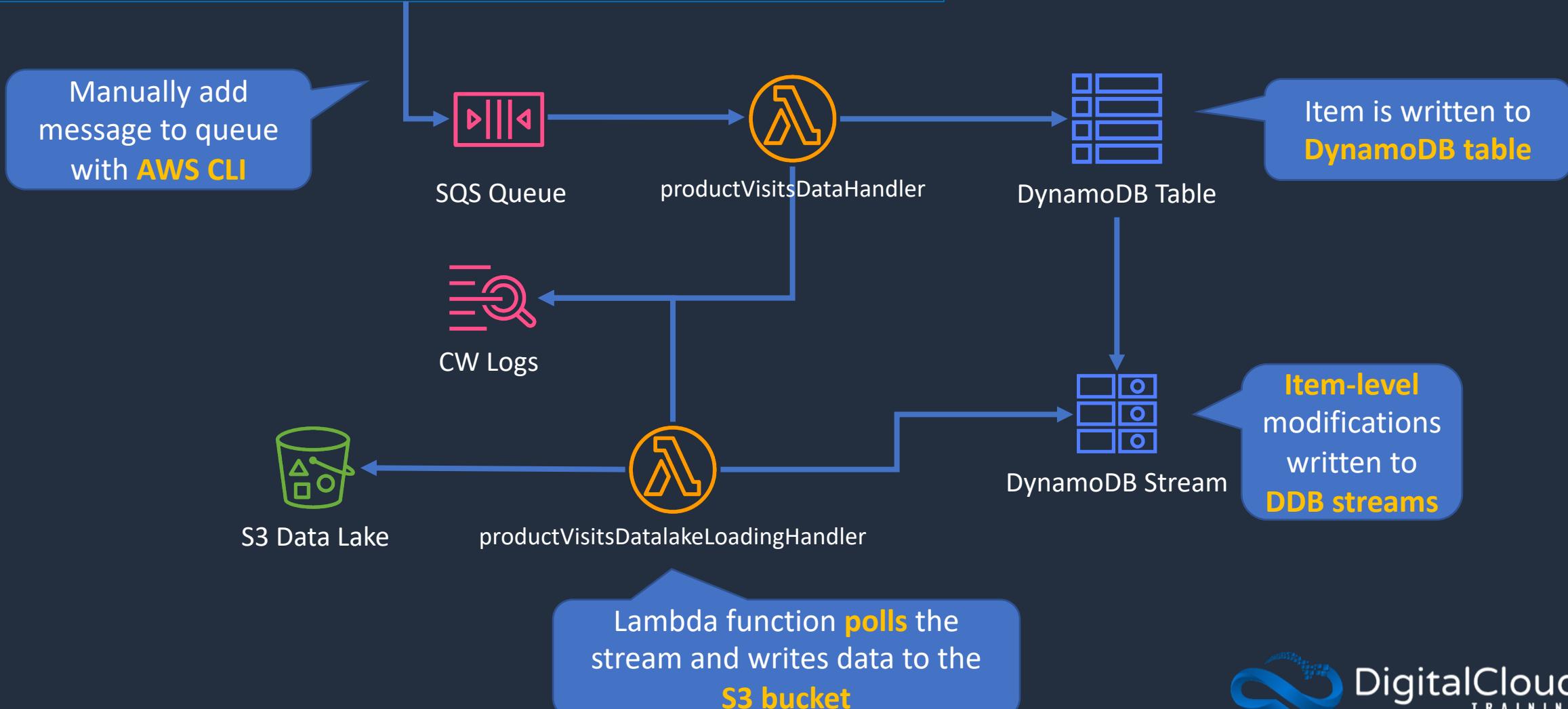
# Build a Serverless App – Part 2





# Build a Serverless App - Part 2

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



# Application Integration Services Comparison



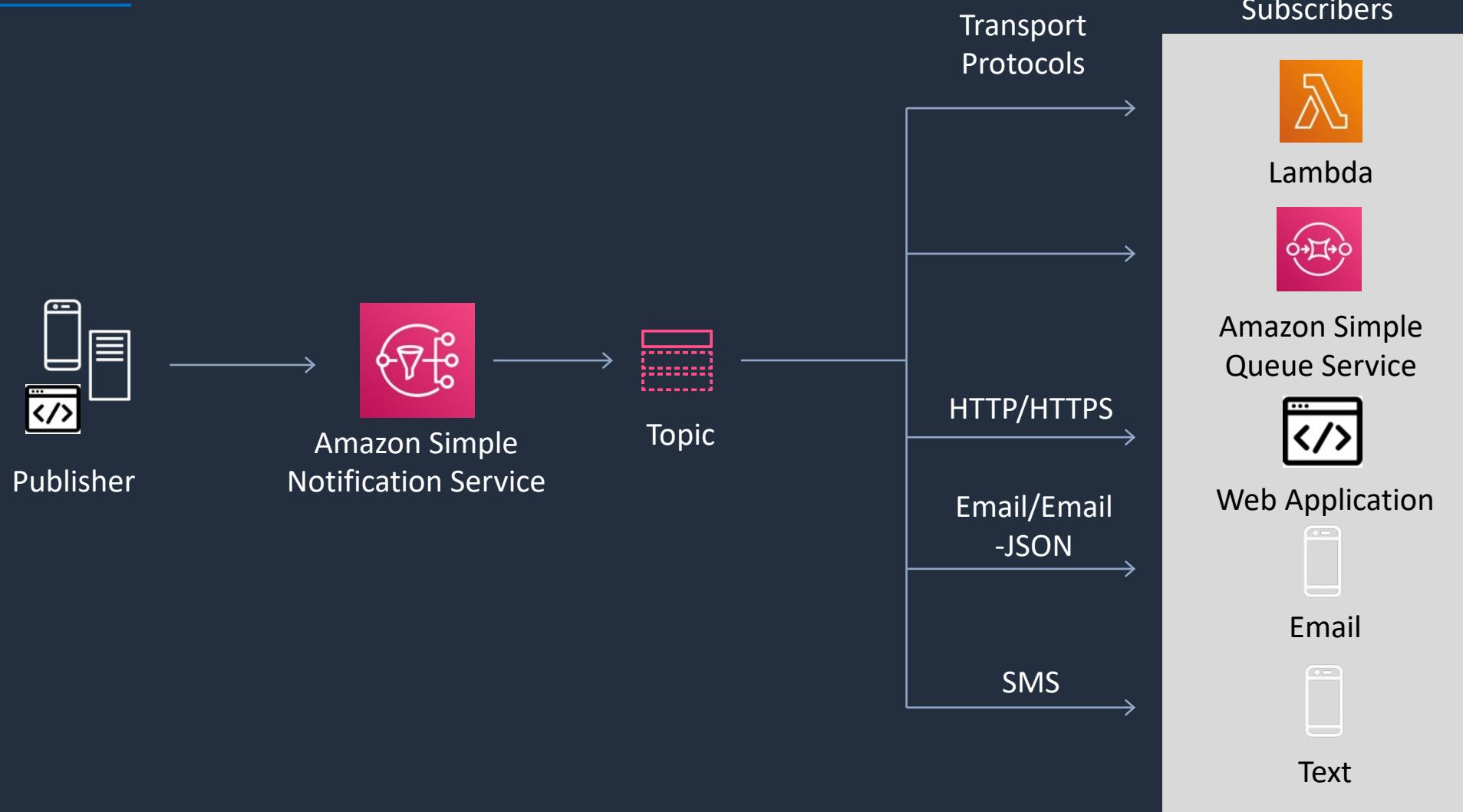


# Application Integration Services Comparison

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

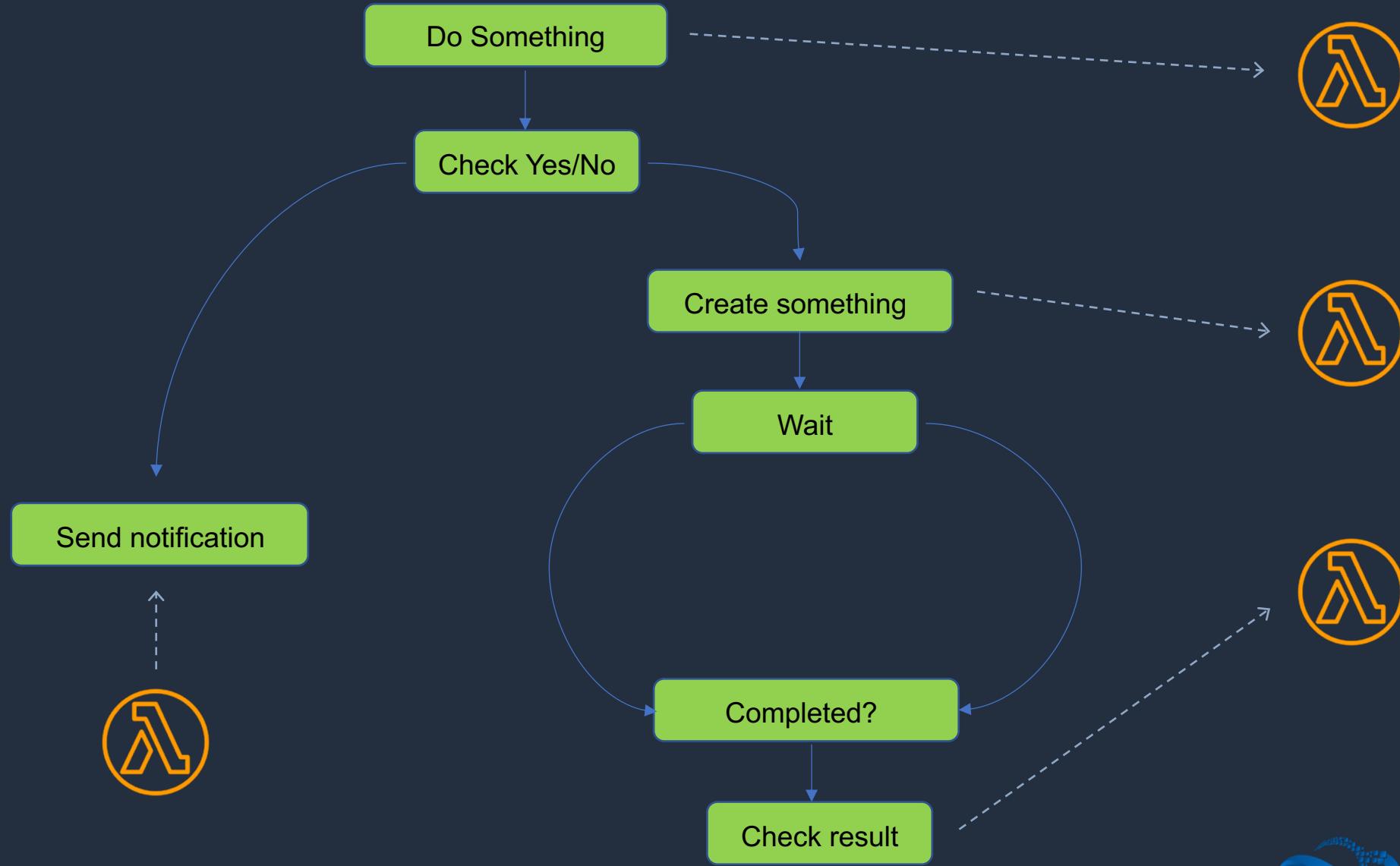


# Amazon SNS Notifications





# 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



# 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 EventBridge





# Amazon EventBridge

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

## Event Sources

AWS Services

Custom Apps

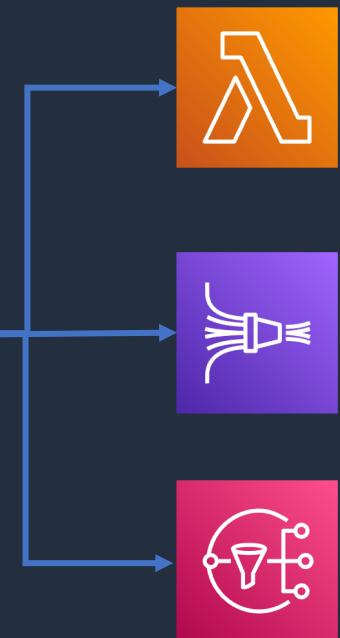
SaaS Apps



Events

EventBridge  
event bus

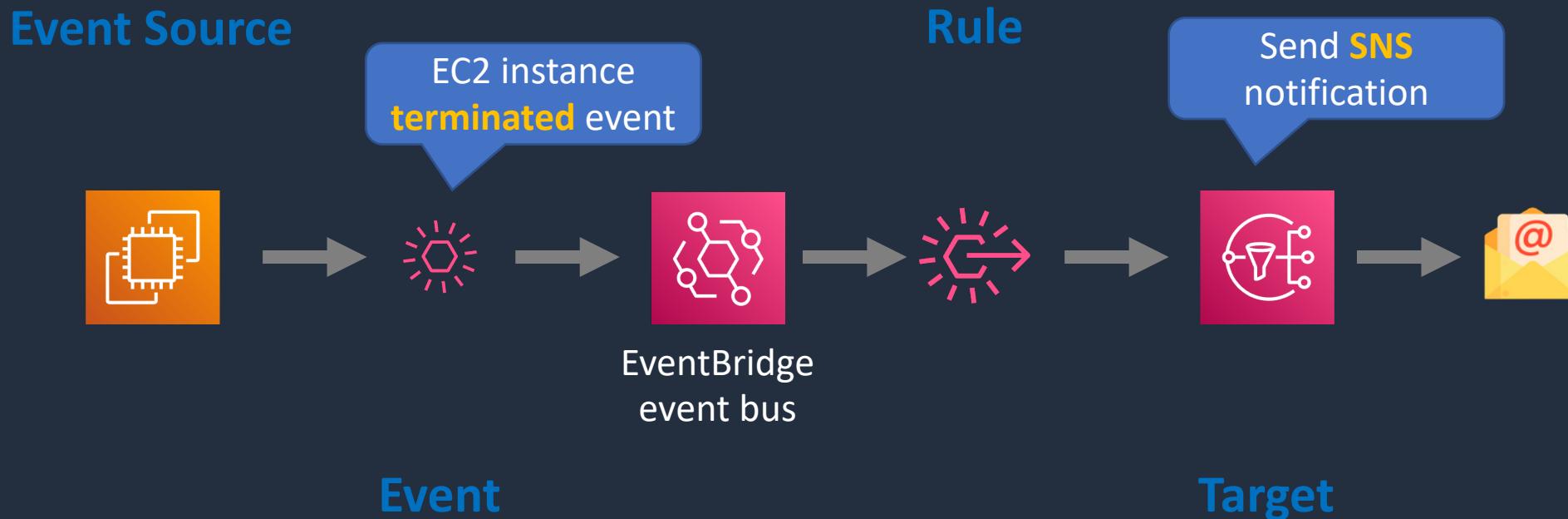
## Rules



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**

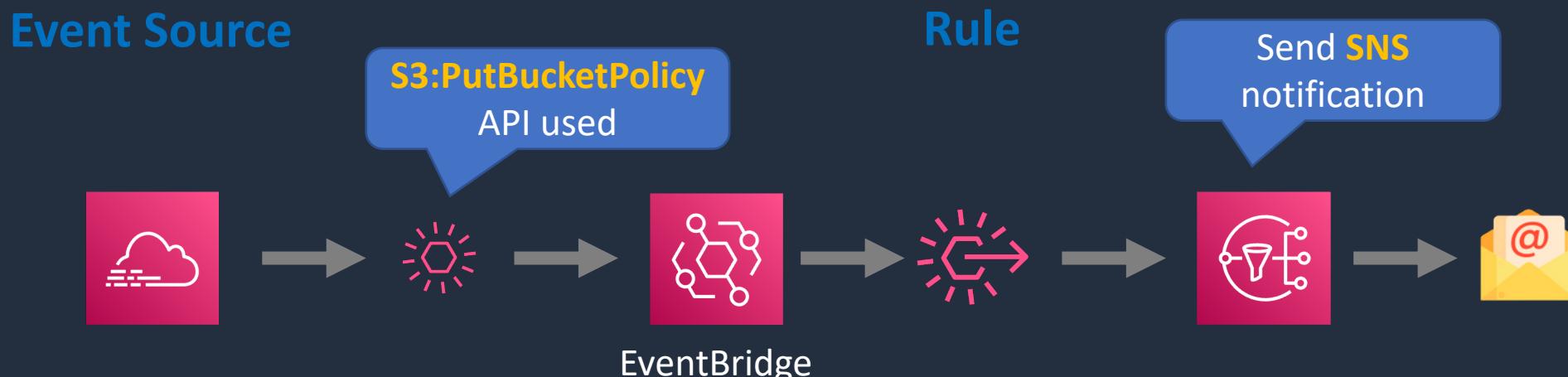
Copy Edit

```
1 {
2     "source": ["aws.ec2"],
3     "detail-type": ["EC2 Instance State-change N
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

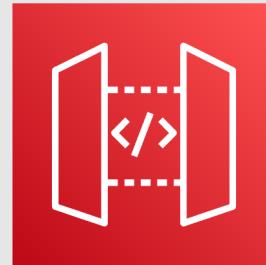


Event

Target

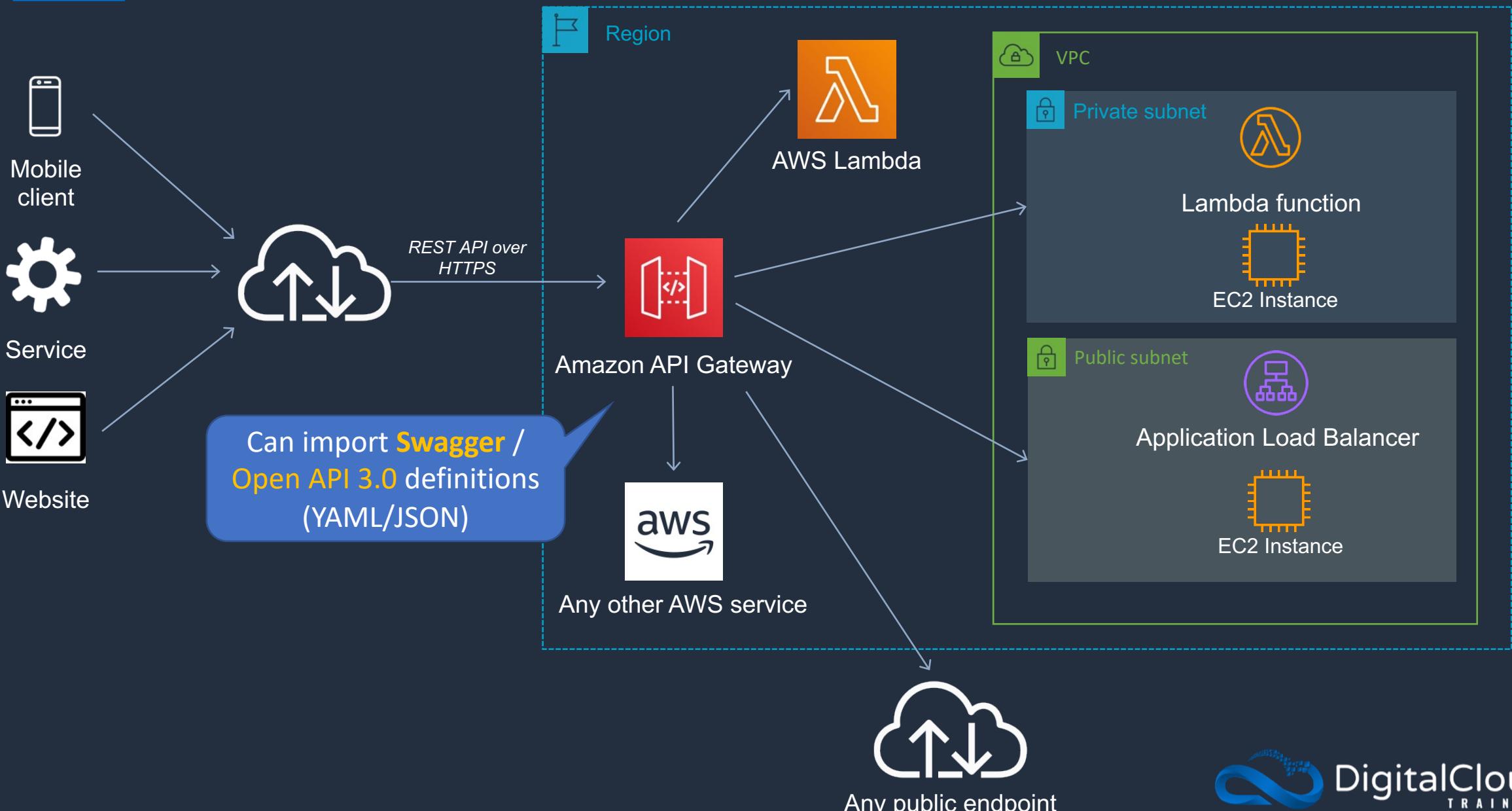
```
{  
    "source": ["aws.cloudtrail"],  
    "detail-type": ["AWS API Call via CloudTrail"],  
    "detail": {  
        "eventSource": ["clouptrail.amazonaws.com"],  
        "eventName": ["s3:PutBucketPolicy"]  
    }  
}
```

# Amazon API Gateway Core Knowledge





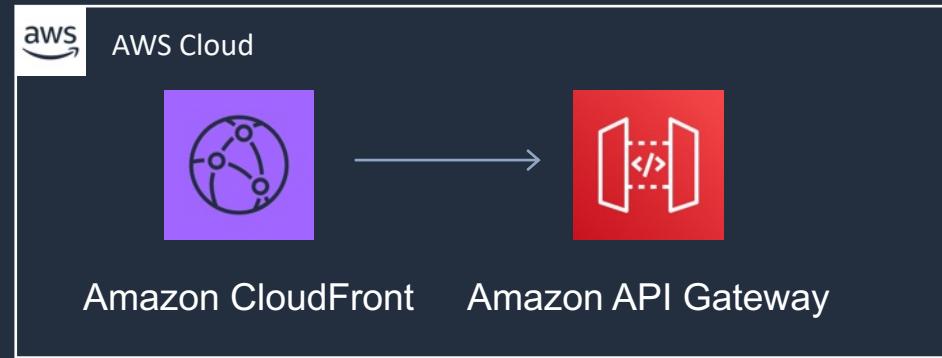
# Amazon API Gateway Overview





# Amazon API Gateway Deployment Types

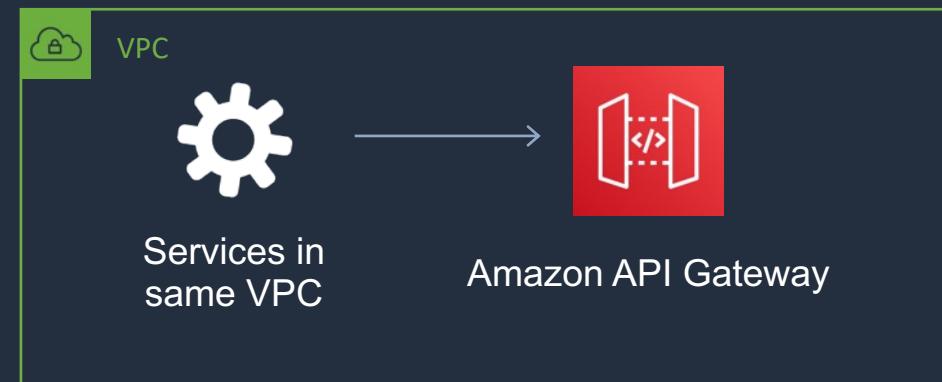
## Edge-optimized endpoint



## Regional endpoint



## Private endpoint



### Key benefits:

- Reduced latency for requests from around the world

### Key benefits:

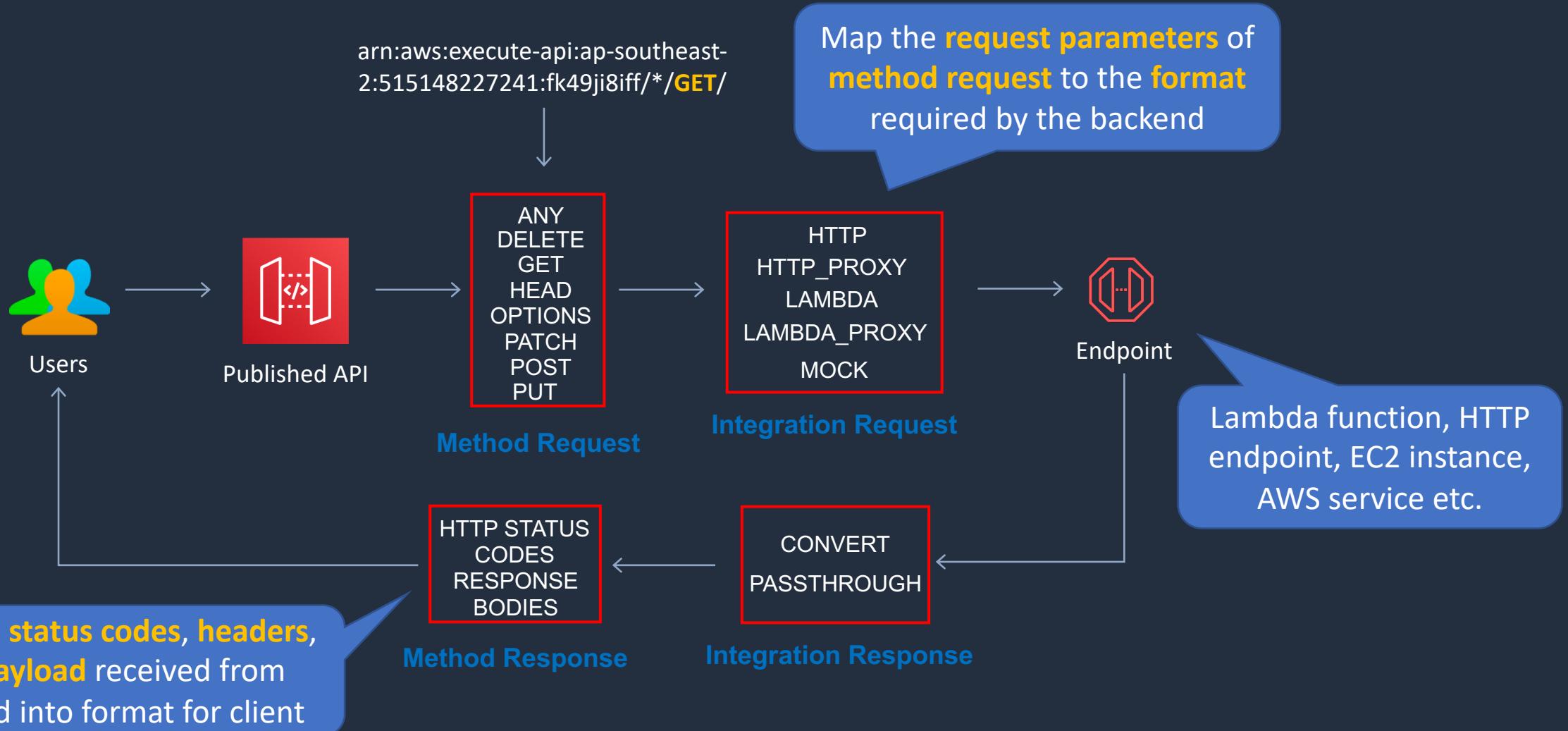
- Reduced latency for requests that originate in the same region
- Can also configure your own CDN and protect with WAF

### Key benefits:

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



# Amazon API Gateway – Structure of an API





# API Gateway Integrations

---

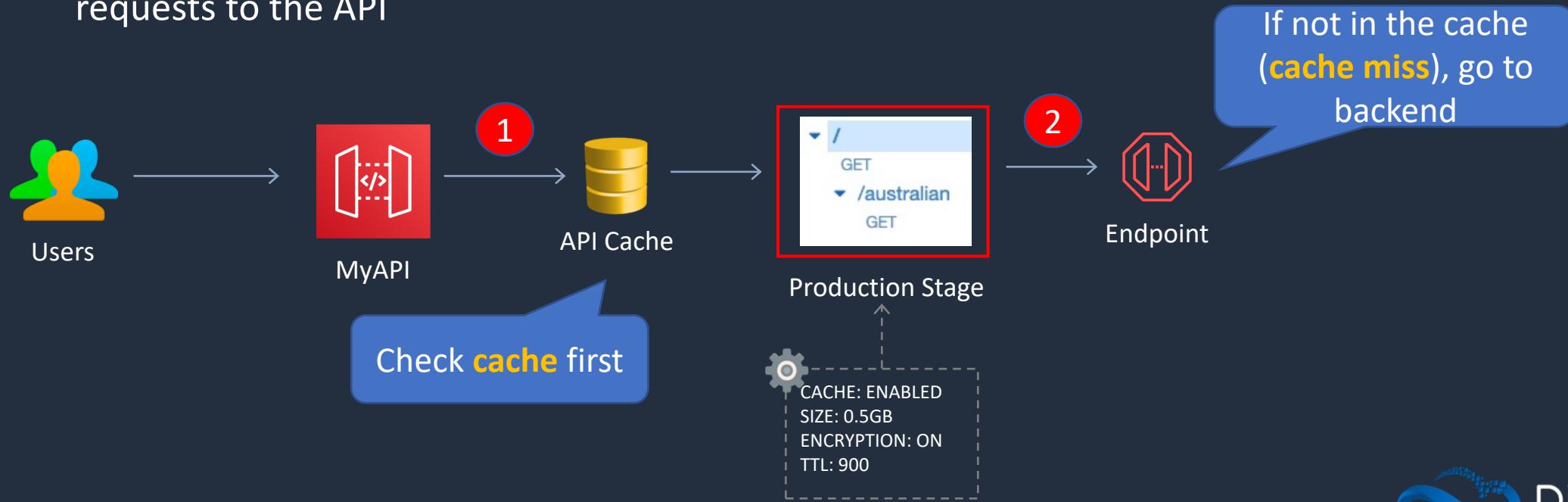
- For a Lambda function, you can have the Lambda proxy integration, or the Lambda custom integration
- For an HTTP endpoint, you can have the HTTP proxy integration or the HTTP custom integration
- For an AWS service action, you have the AWS integration of the non-proxy type only



# 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





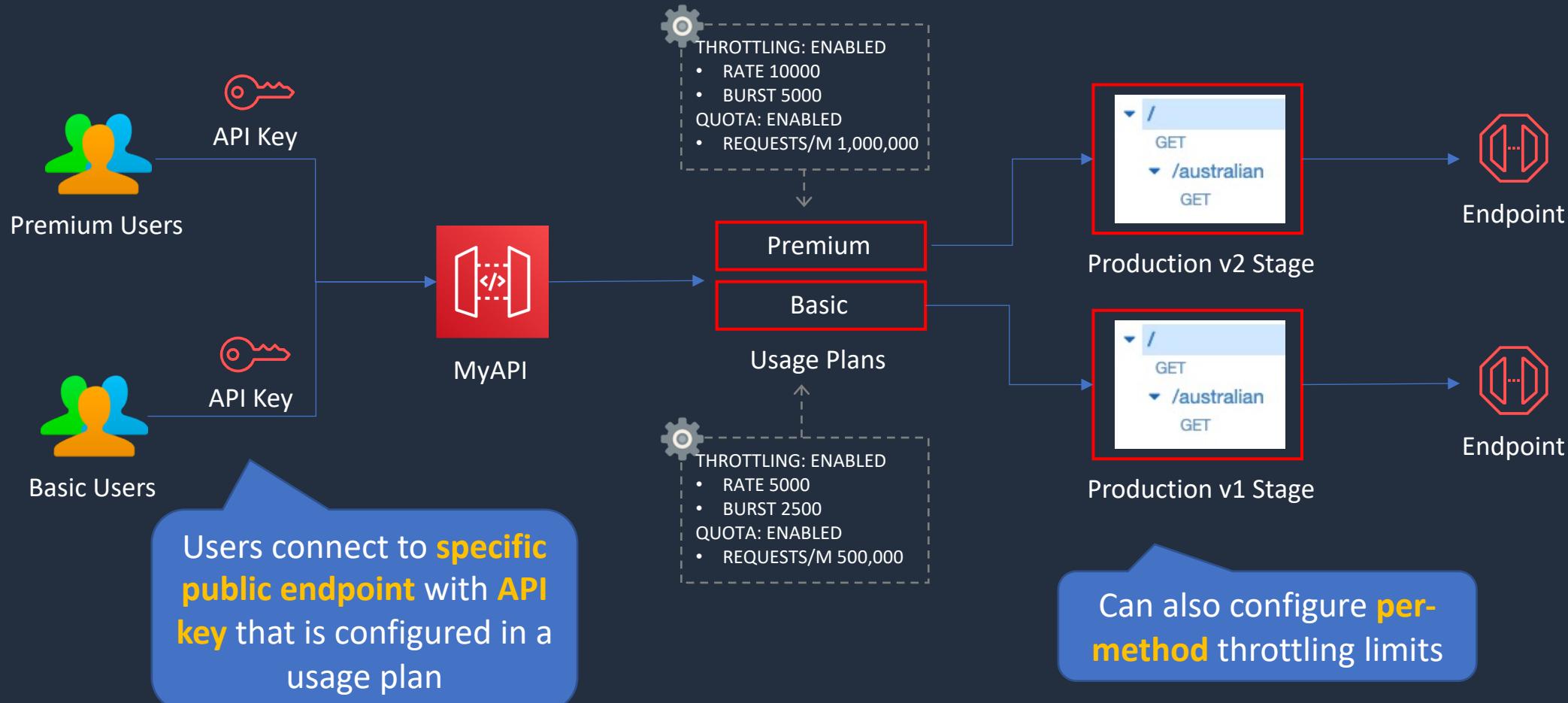
# API Gateway - Throttling

---

- 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

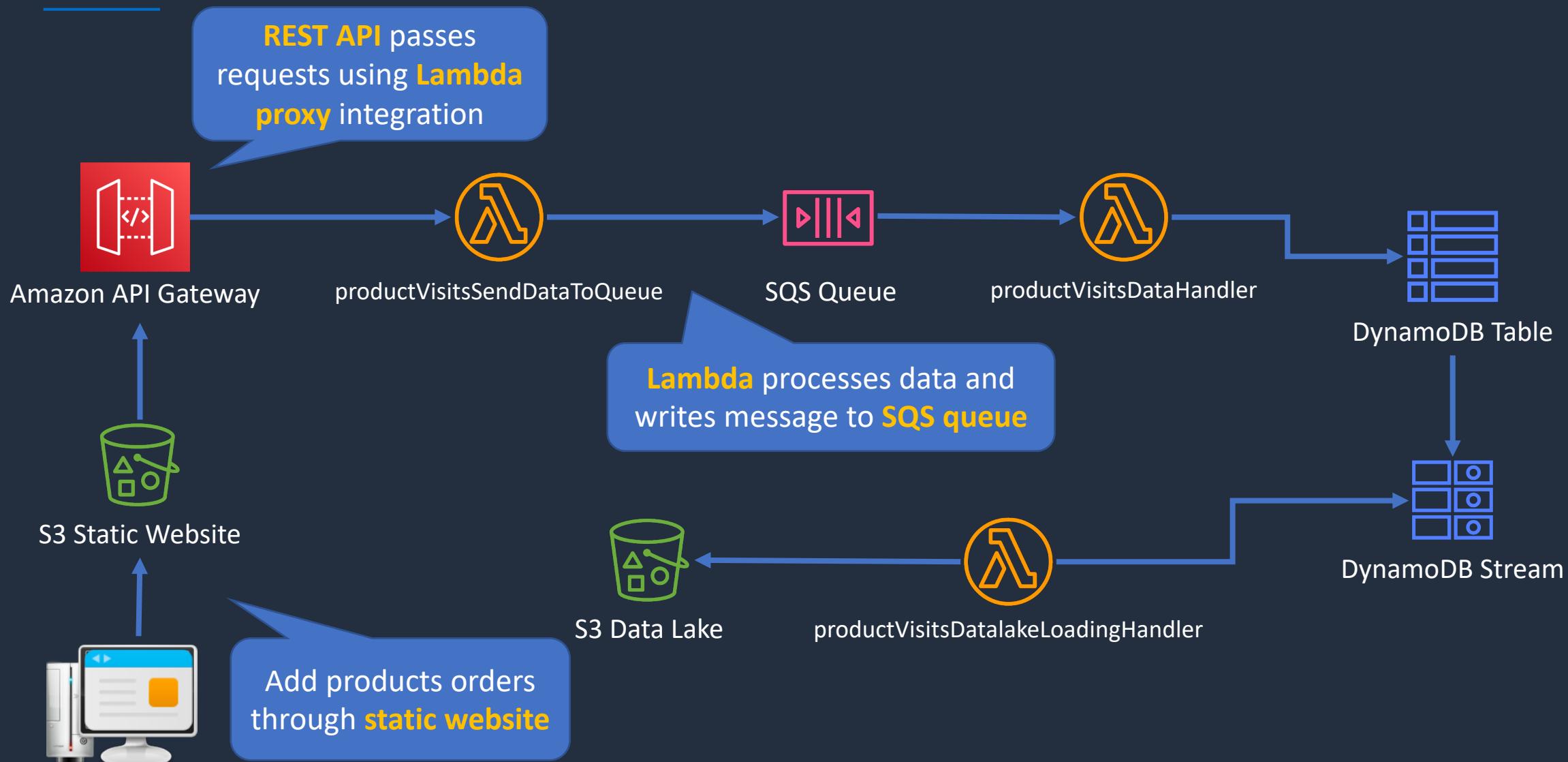


# Build a Serverless App – Part 3





# Build a Serverless App - Part 3

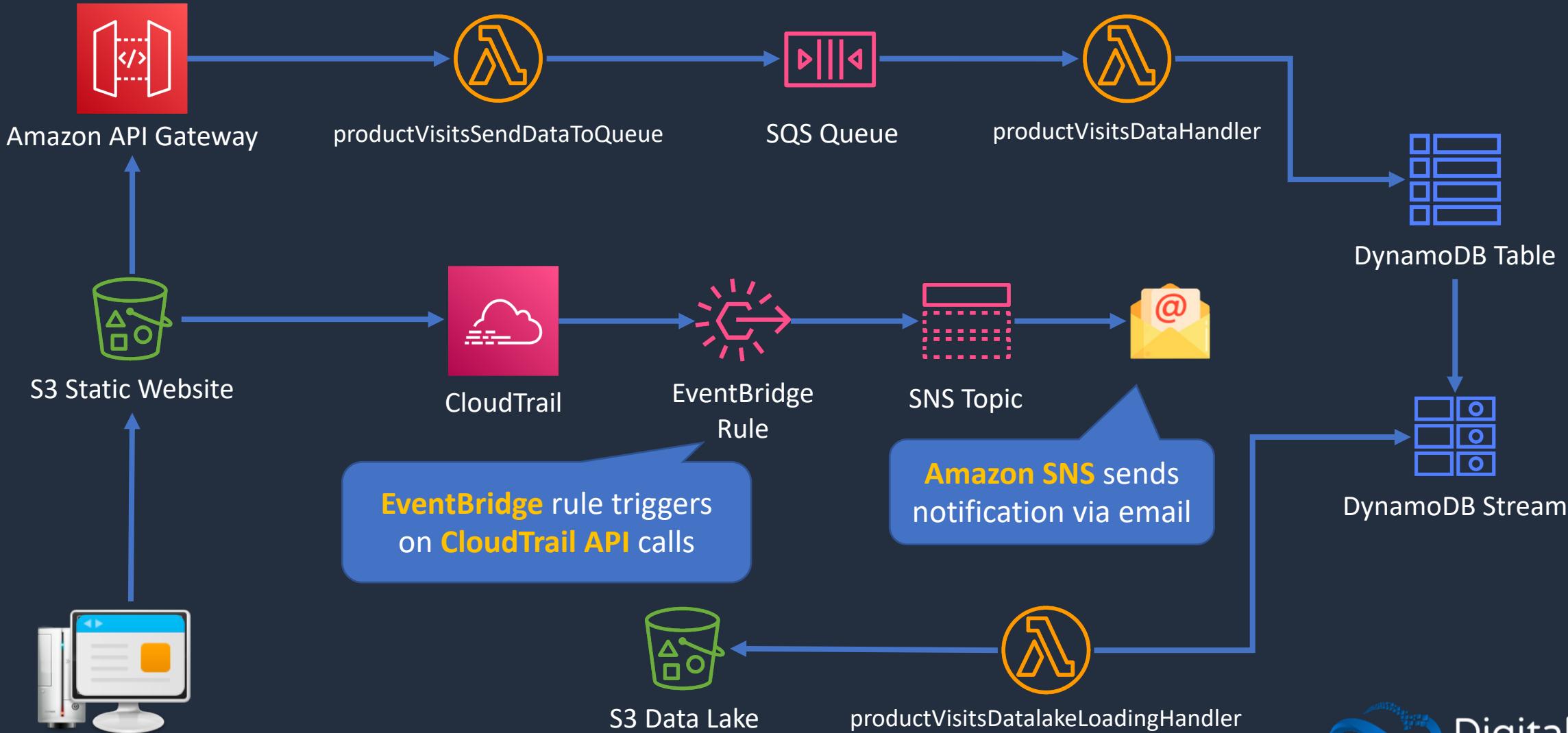


# Build a Serverless App – Part 4

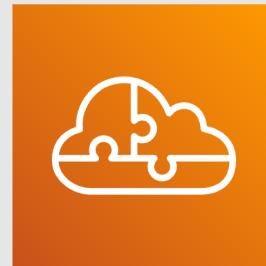




# Build a Serverless App - Part 4



# 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

Migrate decoupled on-premises web app. Users upload files and processing tier processes and stores in NFS file system. Should scale dynamically

Migrate to EC2 instances, SQS and EFS. Use Auto Scaling and scale the processing tier based on the SQS queue length

Lambda function execution time has increased significantly as the number of records in the data to process has increased

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



# Architecture Patterns - Serverless

## Requirement

API Gateway forwards streaming data to AWS Lambda to process and TooManyRequestsException is experienced

Migrating app with highly variable load to AWS. Must be decoupled and orders must be processed in the order they are received

Company needs to process large volumes of media files with Lambda which takes +2hrs. Need to optimize time and automate the whole process

## Solution

Send the data to a Kinesis Data Stream and then configure Lambda to process in batches

Implement an Amazon SQS FIFO queue to preserve the record order

Configure a Lambda function to write jobs to queue. Configure queue as input to Step Functions which will coordinate multiple functions to process in parallel



# Architecture Patterns - Serverless

## Requirement

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

Company requires API events that involve the root user account to be captured in a third-party ticketing system

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

## Solution

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

Create a CloudTrail trail and an EventBridge rule that looks for API events that involve root, put events on SQS queue; process queue with Lambda

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



# Architecture Patterns - Serverless

## Requirement

Lambda processes objects created in bucket. Large volumes of objects can be uploaded. Must ensure function does not affect other critical functions

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

Solutions Architect needs to update Lambda function code using canary strategy; traffic should be routed based on weights

## Solution

Configure reserved concurrency to set the maximum limit for the function. Monitor critical functions' CloudWatch alarms for the Throttles Lambda metric

Replace EC2 with AWS Lambda function

Create an Alias for the Lambda function and configure weights to Lambda versions



# Architecture Patterns - Serverless

## Requirement

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

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

Need to ensure only authorized IAM users can access REST API on API Gateway

## Solution

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

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

Set authorization to AWS\_IAM for API Gateway method. Grant execute-api:Invoke permissions in IAM policy

# SECTION 12

## Docker Containers and PaaS

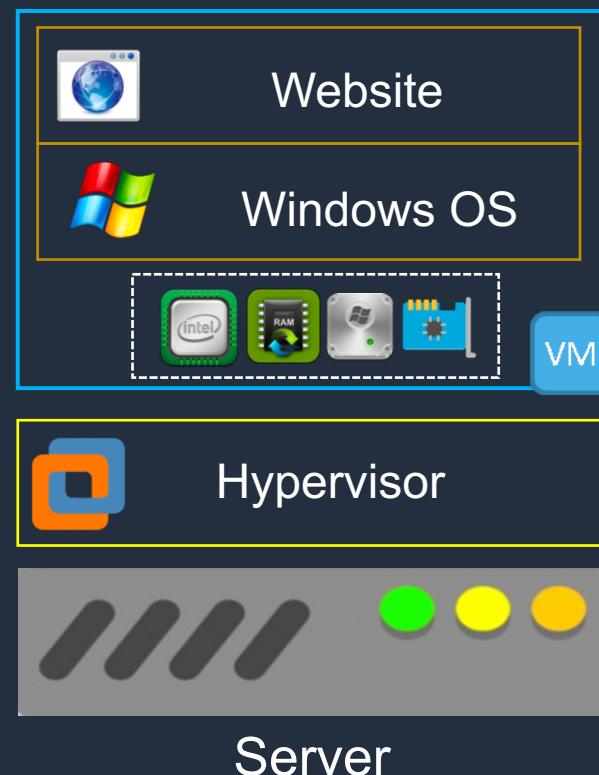
# Docker and Microservices Architectures





# Docker Containers

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





# Docker Containers

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

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



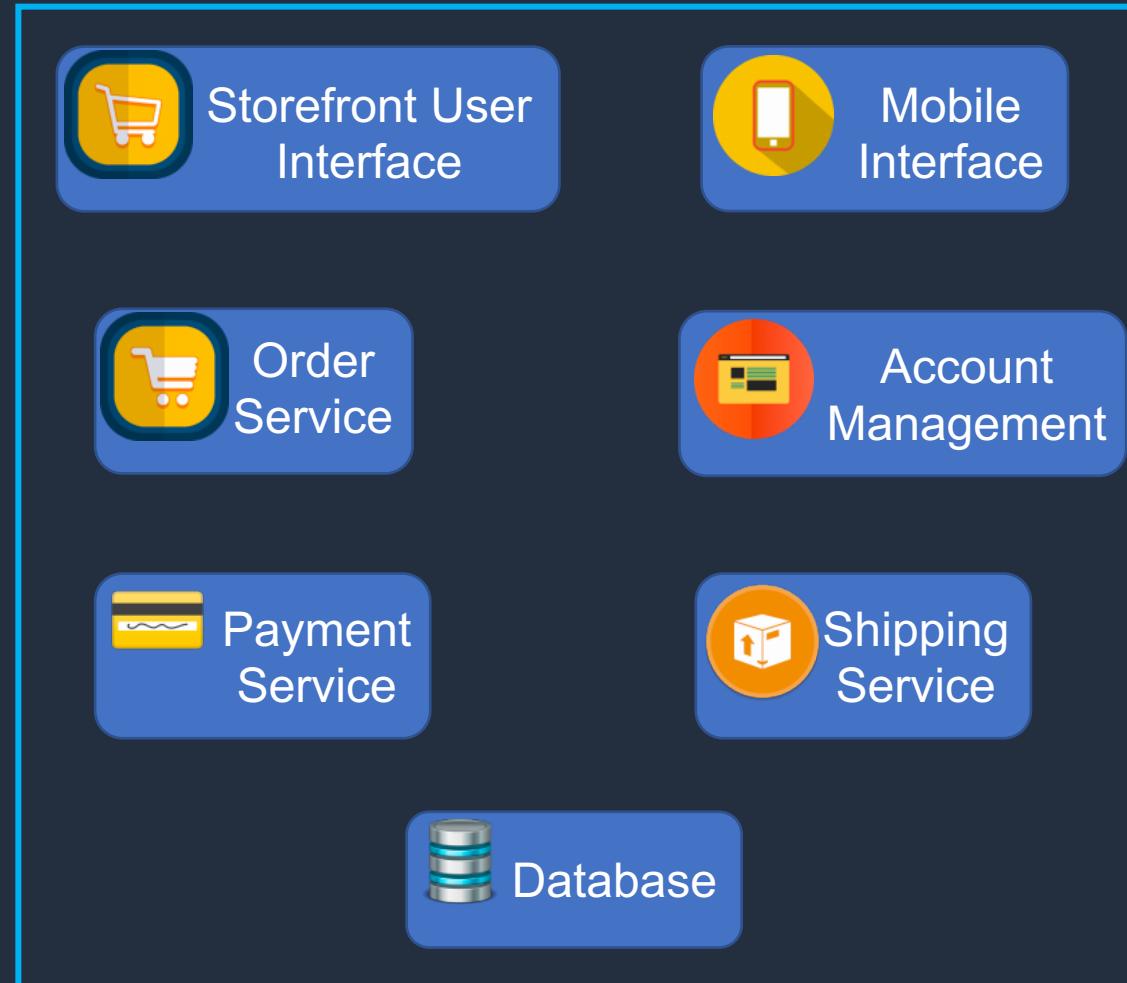
Containers are very resource **efficient**

Each container is **isolated** from other containers



# Monolithic Application

---





# Monolithic Application

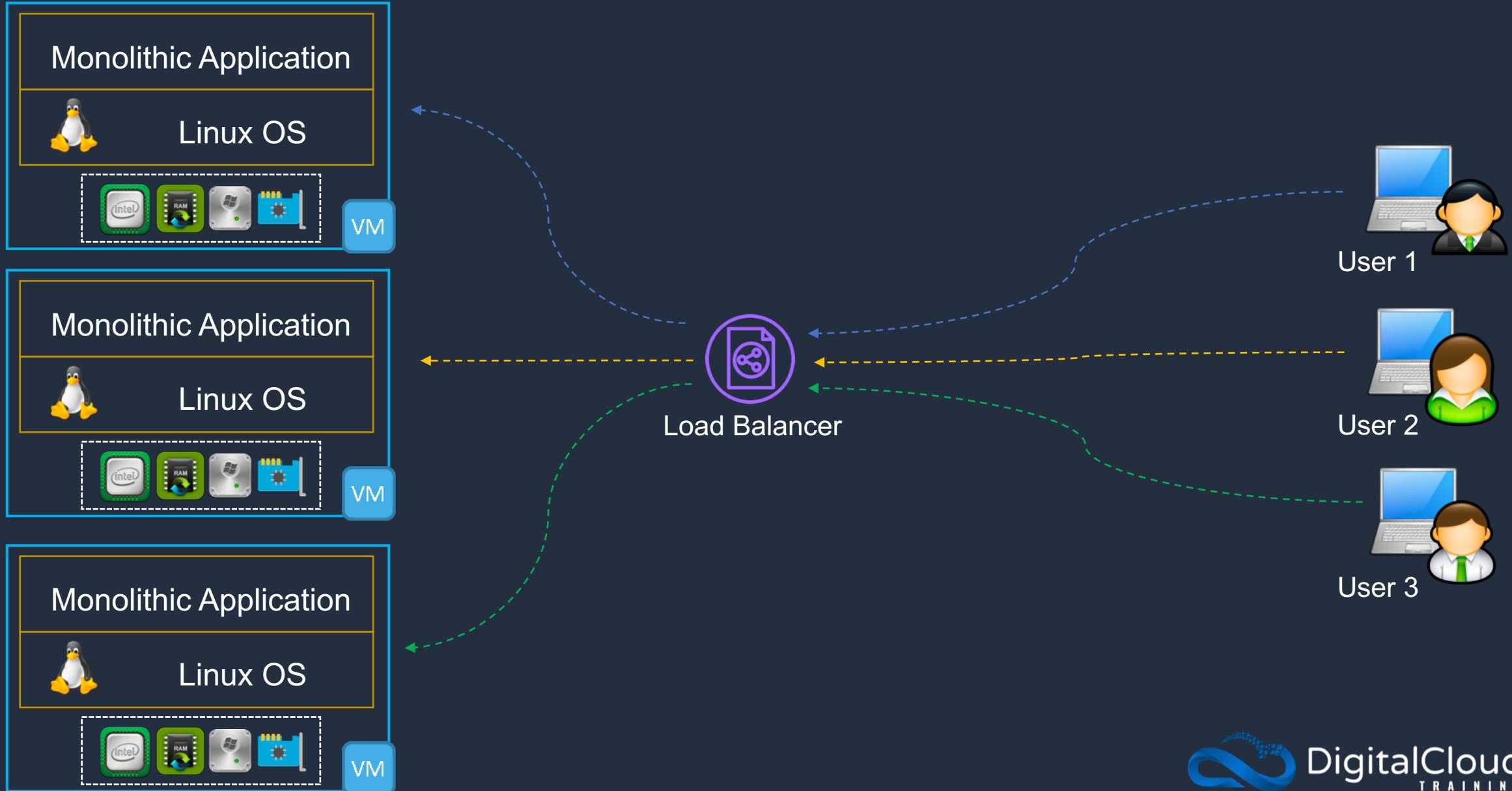
Updates to, or failures of, any single component can take down the whole application



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



# Monolithic Application





# Monolithic Application

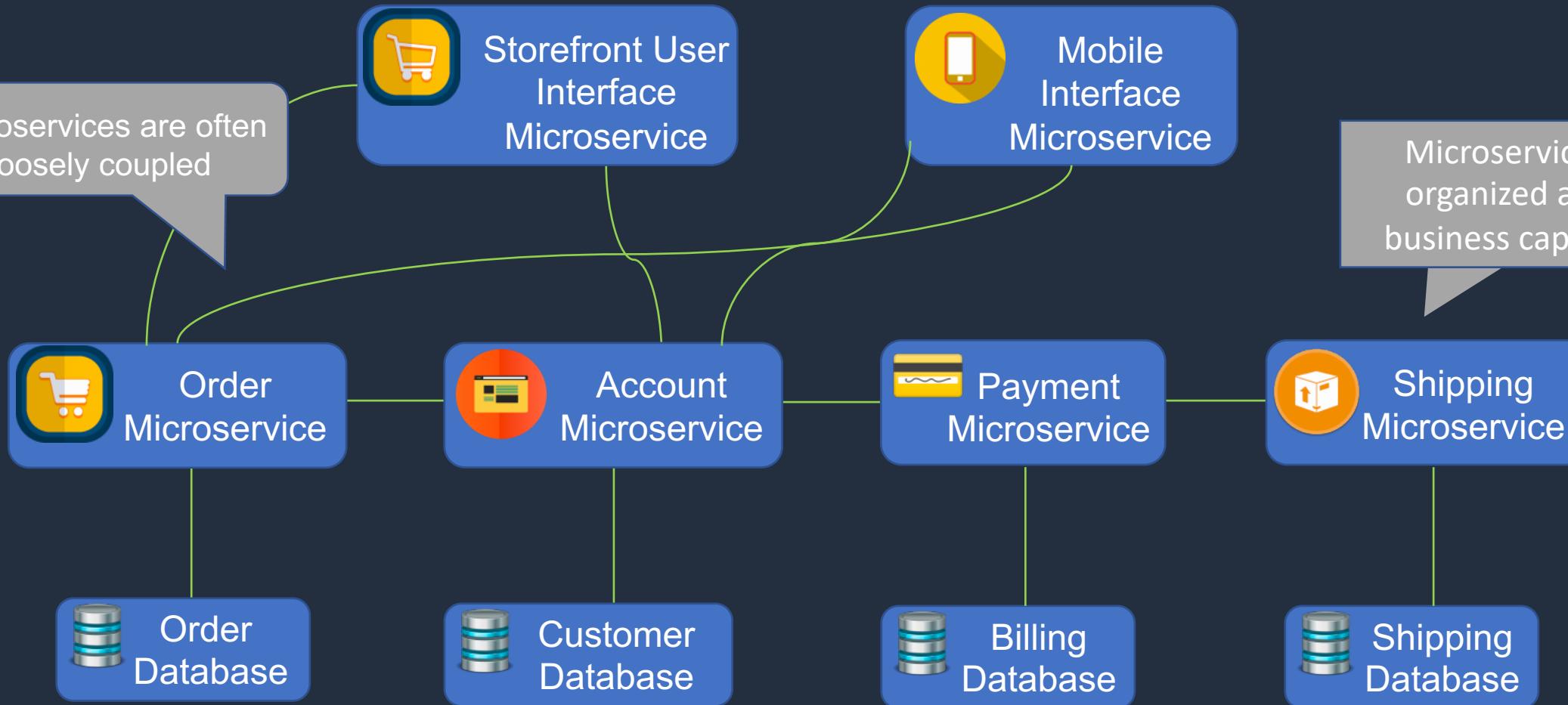




# Microservices Architecture

A microservice is an independently deployable unit of code

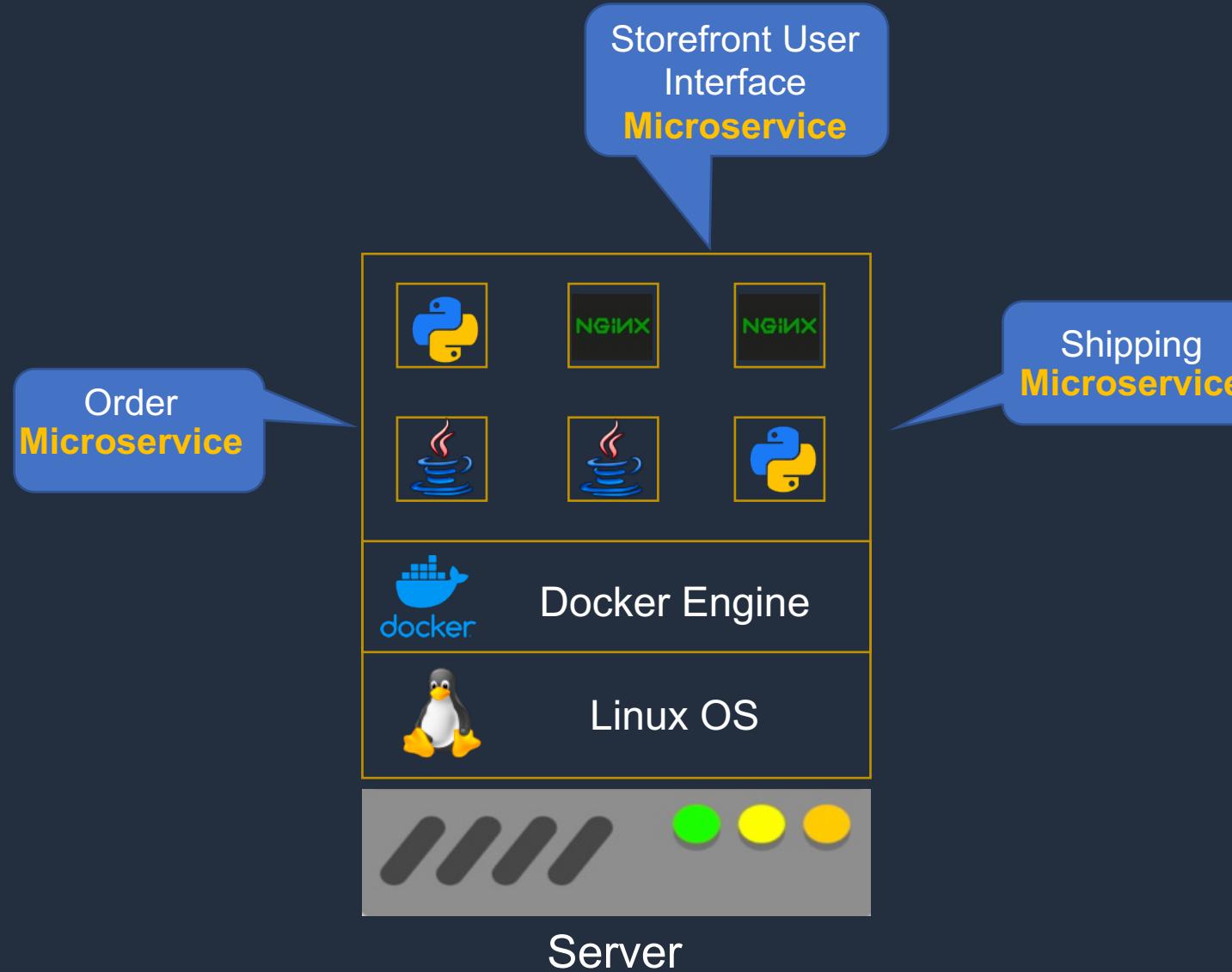
Microservices are often loosely coupled



Microservices are organized around business capabilities



# Microservices with Docker Containers

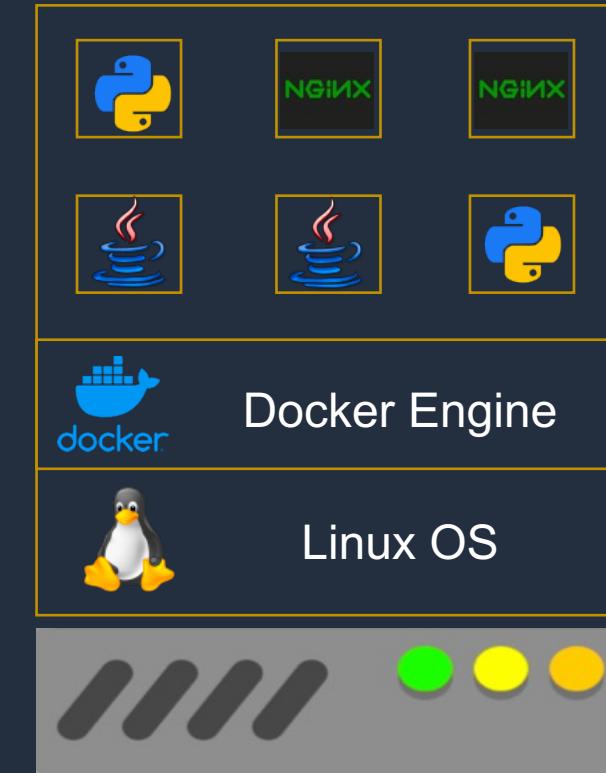




# Microservices with Docker Containers

**Microservices** can also be spread **across hosts**

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





# Microservices: Attributes and Benefits

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

# Amazon ECS Core Knowledge





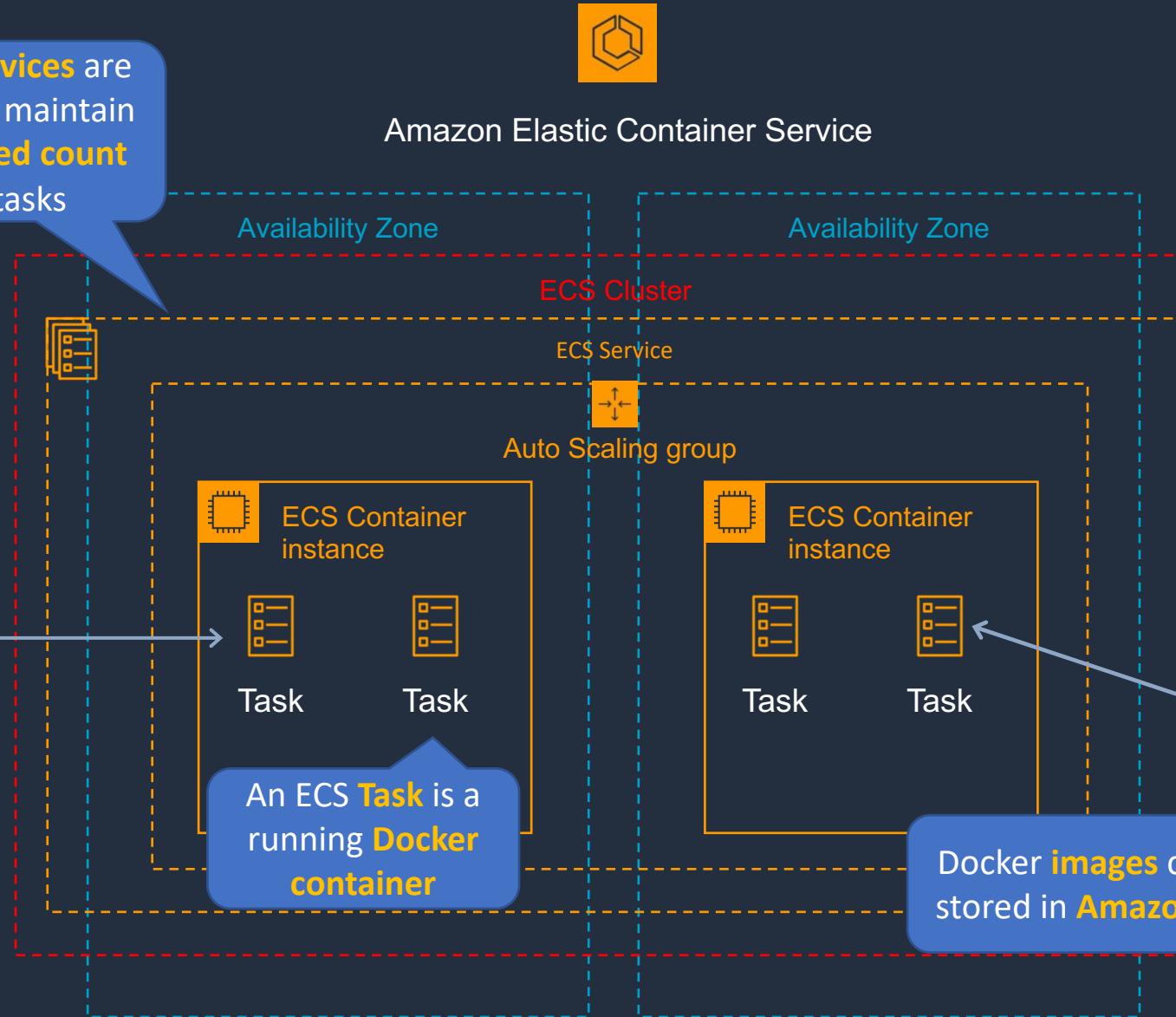
# Amazon ECS

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

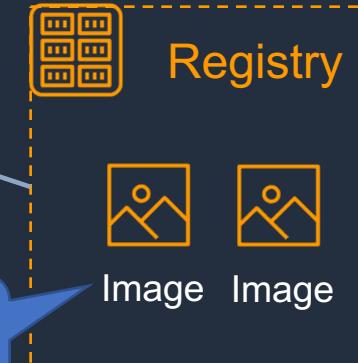


## Amazon Elastic Container Service

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



Amazon Elastic Container Registry





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



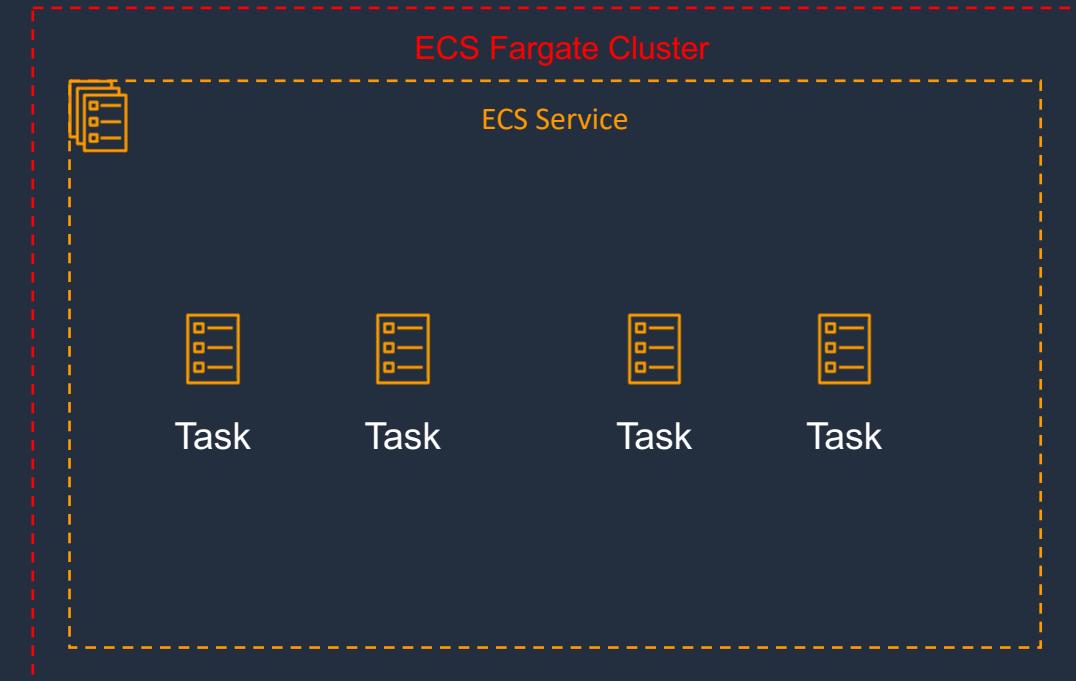
# Launch Types – EC2 and Fargate



Registry:  
ECR, Docker Hub, Self-hosted



Registry:  
ECR, Docker Hub



## EC2 Launch Type

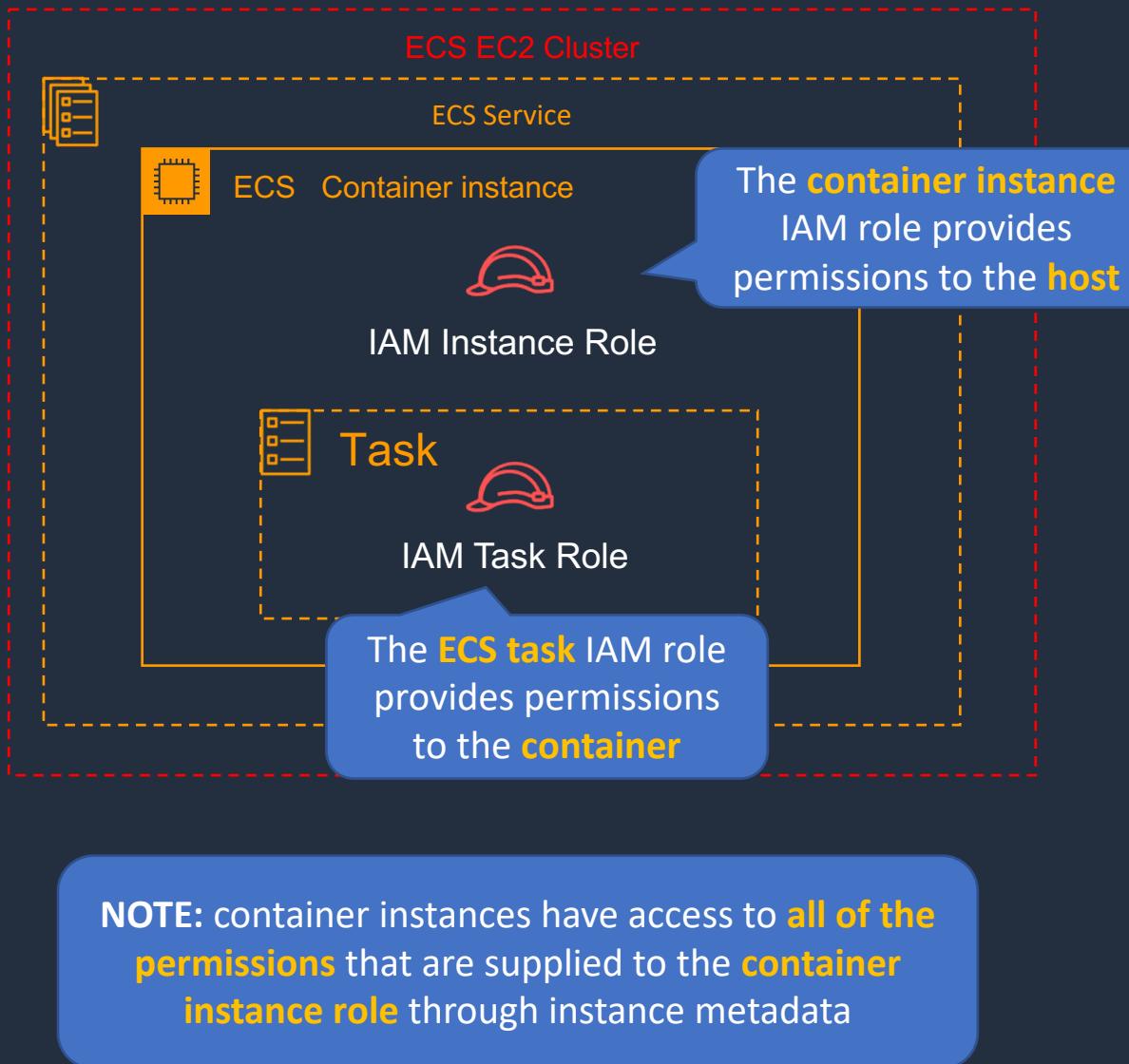
- You explicitly provision EC2 instances
- You're responsible for managing EC2 instances
- Charged per running EC2 instance
- EFS and EBS integration
- You handle cluster optimization
- More granular control over infrastructure

## Fargate Launch Type

- Fargate automatically provisions resources
- Fargate provisions and manages compute
- Charged for running tasks
- No EFS and EBS integration
- Fargate handles cluster optimization
- Limited control, infrastructure is automated



# ECS and IAM Roles



## AmazonEC2ContainerServiceforEC2Role

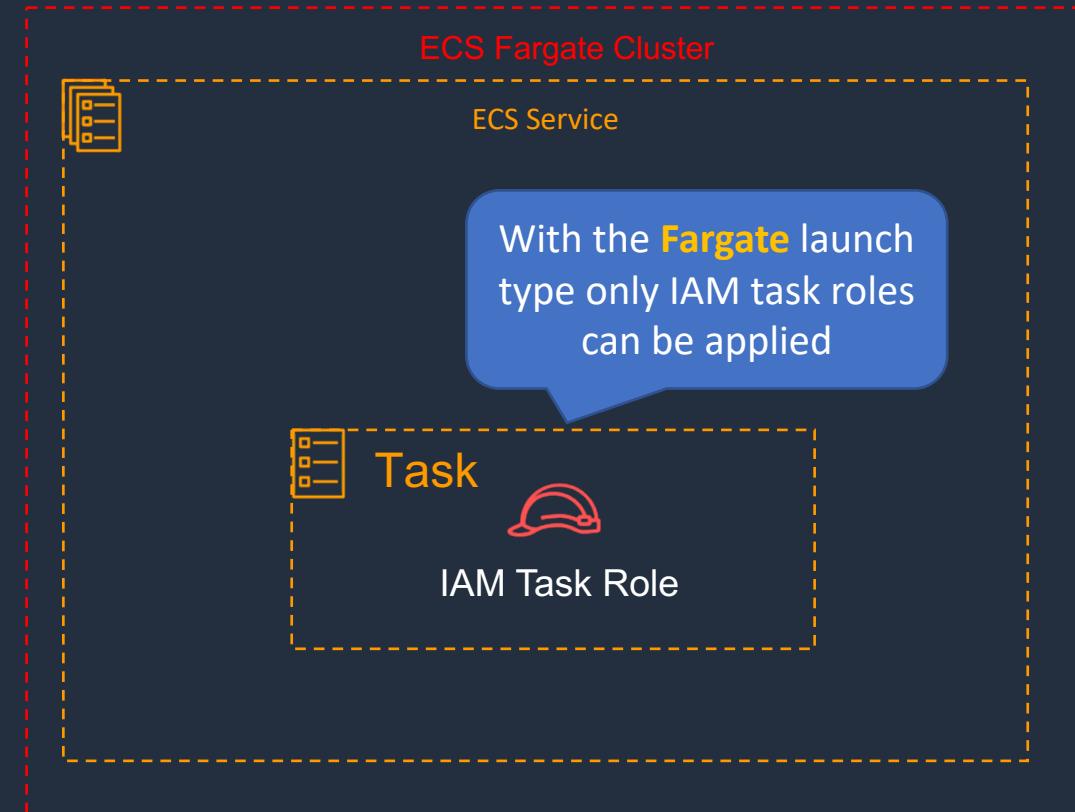
```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Action": [  
        "ec2:DescribeTags",  
        "ecs>CreateCluster",  
        "ecs>DeregisterContainerInstance",  
        "ecs>DiscoverPollEndpoint",  
        "ecs>Poll",  
        "ecs>RegisterContainerInstance",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



# ECS and IAM Roles

---

---





# ECS Networking modes

---

- **awsvpc** — The task is allocated its own elastic network interface (ENI) and a primary private IPv4 address. This gives the task the same networking properties as Amazon EC2 instances
- **bridge** — The task utilizes Docker's built-in virtual network which runs inside each Amazon EC2 instance hosting the task
- **host** — The task bypasses Docker's built-in virtual network and maps container ports directly to the ENI of the Amazon EC2 instance hosting the task
- **none** — The task has no external network connectivity



# ECS Spot Instance & Draining

---

- Can run ECS instances using Spot
- ECS **Spot Instance draining** can be enabled on the instance. ECS receives a Spot Instance interruption notice and places the instance in **DRAINING** status
- When a container instance is set to **DRAINING**, Amazon ECS prevents new tasks from being scheduled for placement on the container instance

# Scaling Amazon ECS





# Auto Scaling for ECS

---

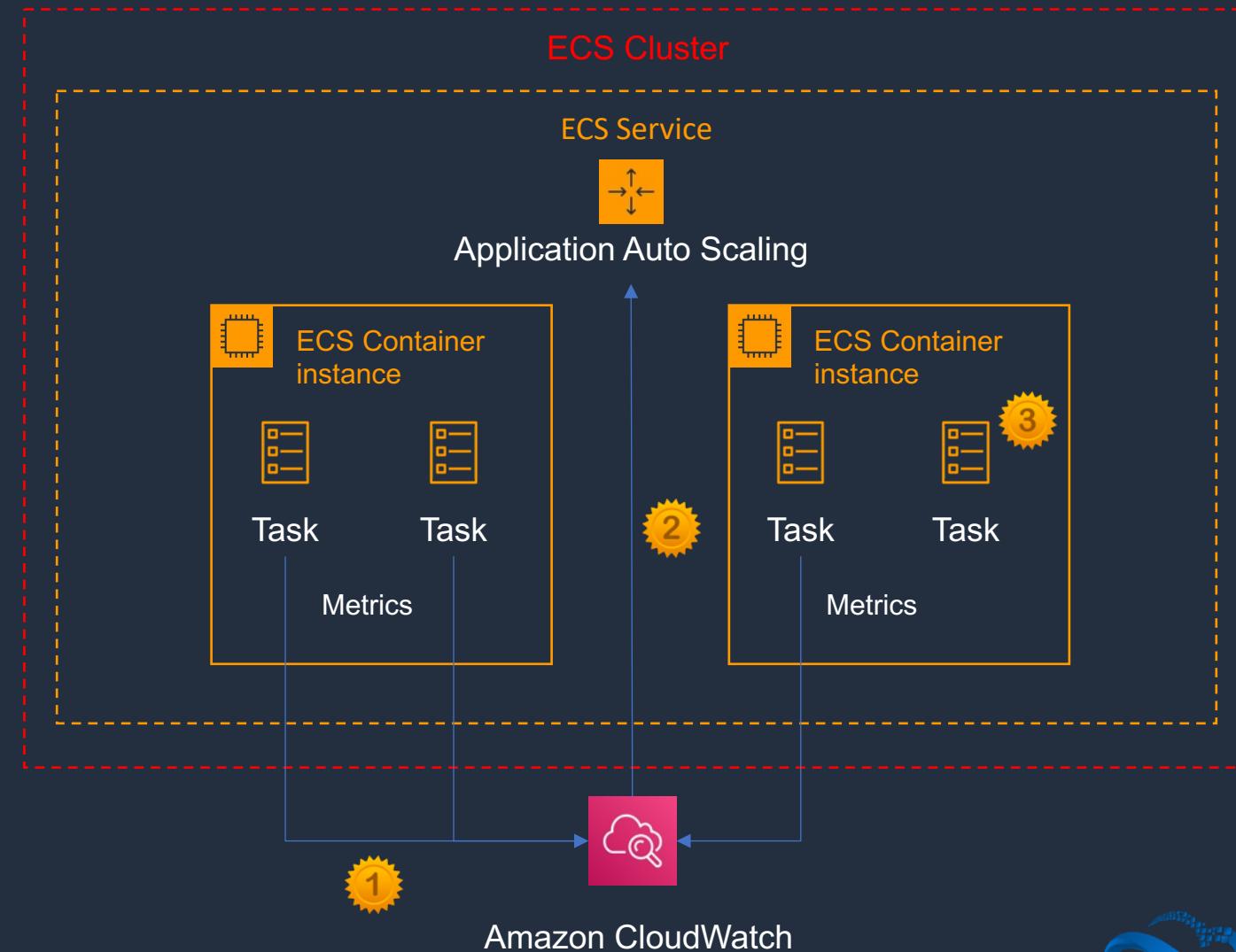
---

- Two types of scaling: service auto scaling and 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

---

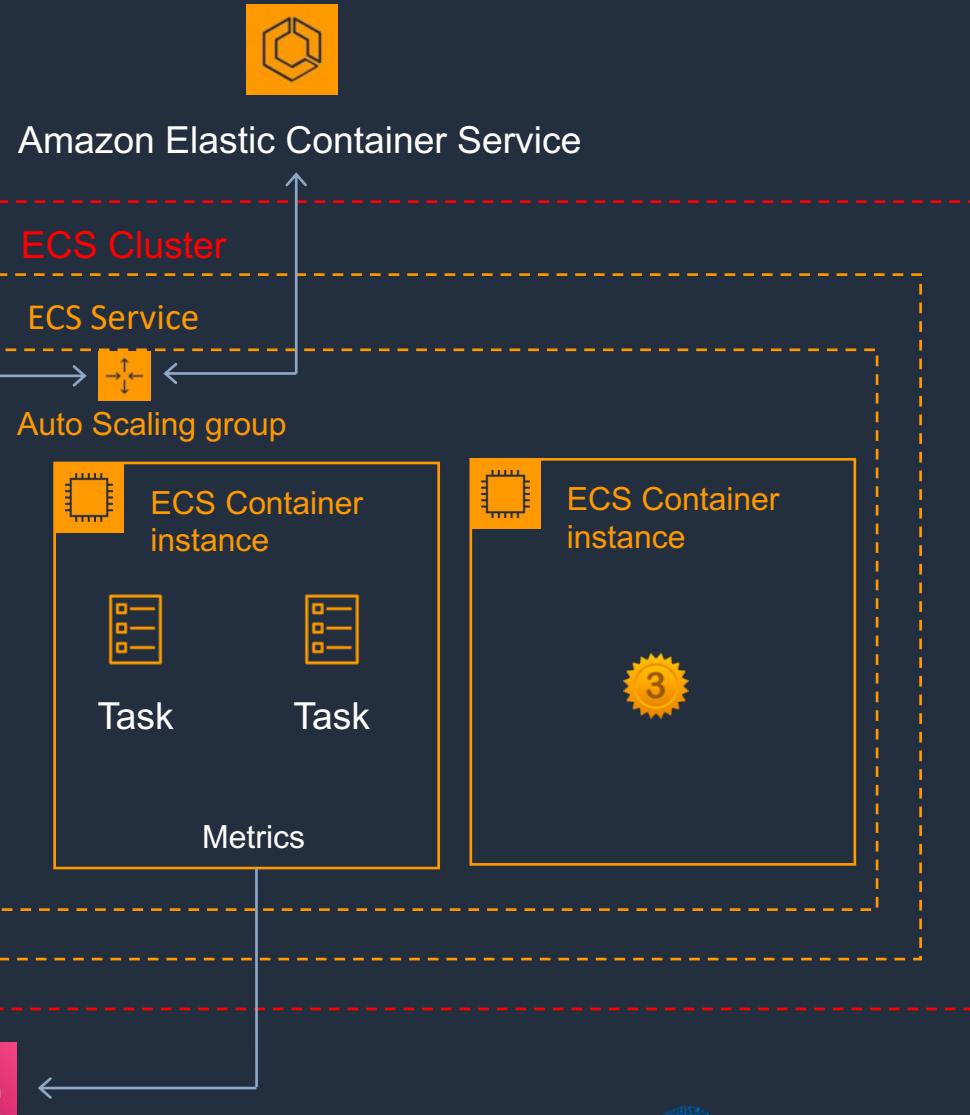
- Amazon ECS service can optionally be configured to use **Service Auto Scaling** to adjust the desired **task count** up or down automatically
- Service Auto Scaling leverages the Application Auto Scaling service to provide this functionality.
- Amazon ECS Service Auto Scaling supports the following 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**



A **Capacity provider reservation** metric measures the total percentage of cluster resources needed by all ECS workloads in the cluster



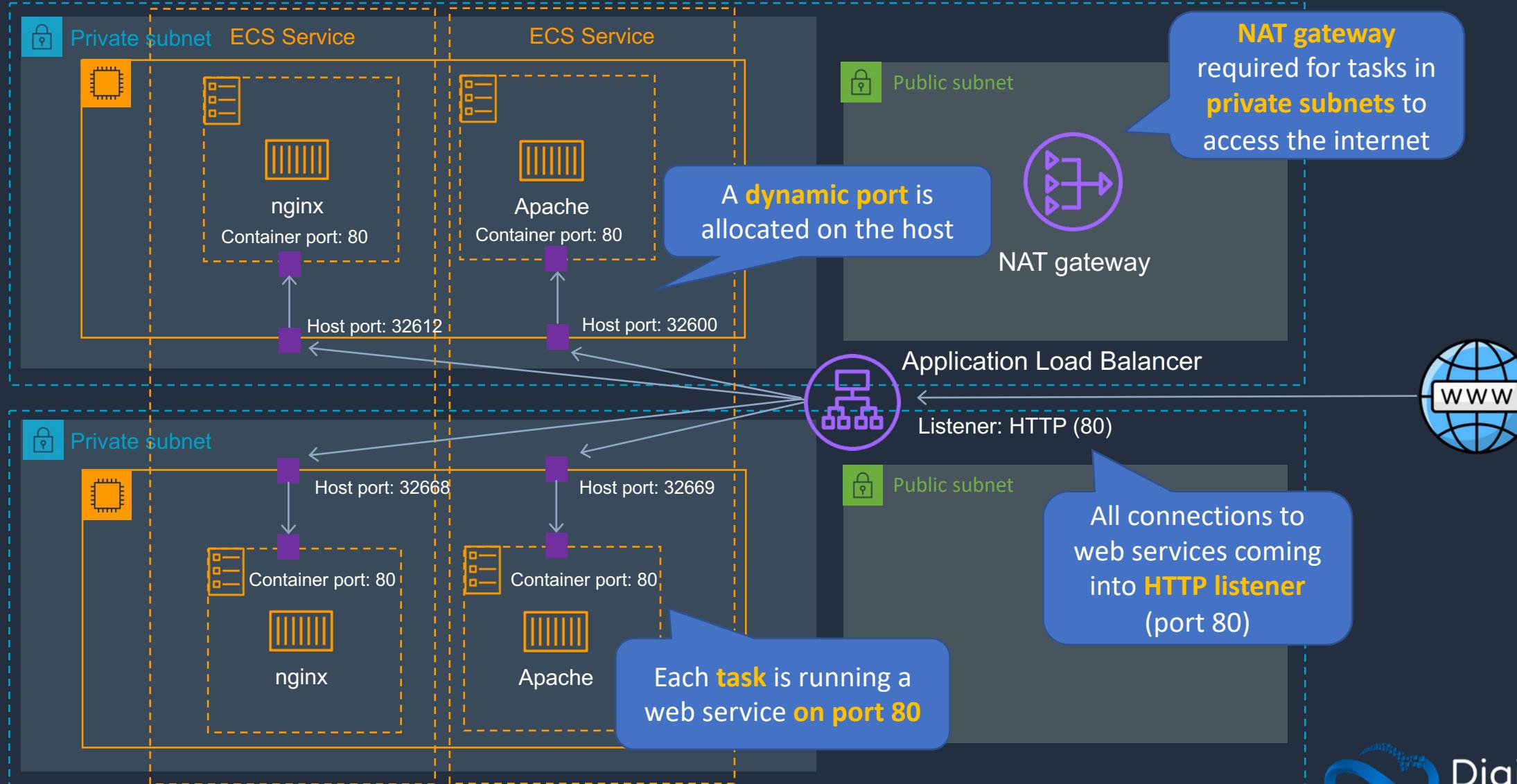
# Cluster Auto Scaling

- Uses an ECS resource type called a **Capacity Provider**
- A Capacity Provider can be associated with an EC2 **Auto Scaling Group** (ASG)
- When you associate an ECS Capacity Provider with an ASG and add the Capacity Provider to an ECS cluster, the cluster can now scale your ASG automatically by using two new features of ECS:
  - **Managed scaling**, with an automatically-created scaling policy on your ASG, and a new scaling metric (Capacity Provider Reservation) that the scaling policy uses; and
  - **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



# [HOL] Create and Scale ECS Tasks

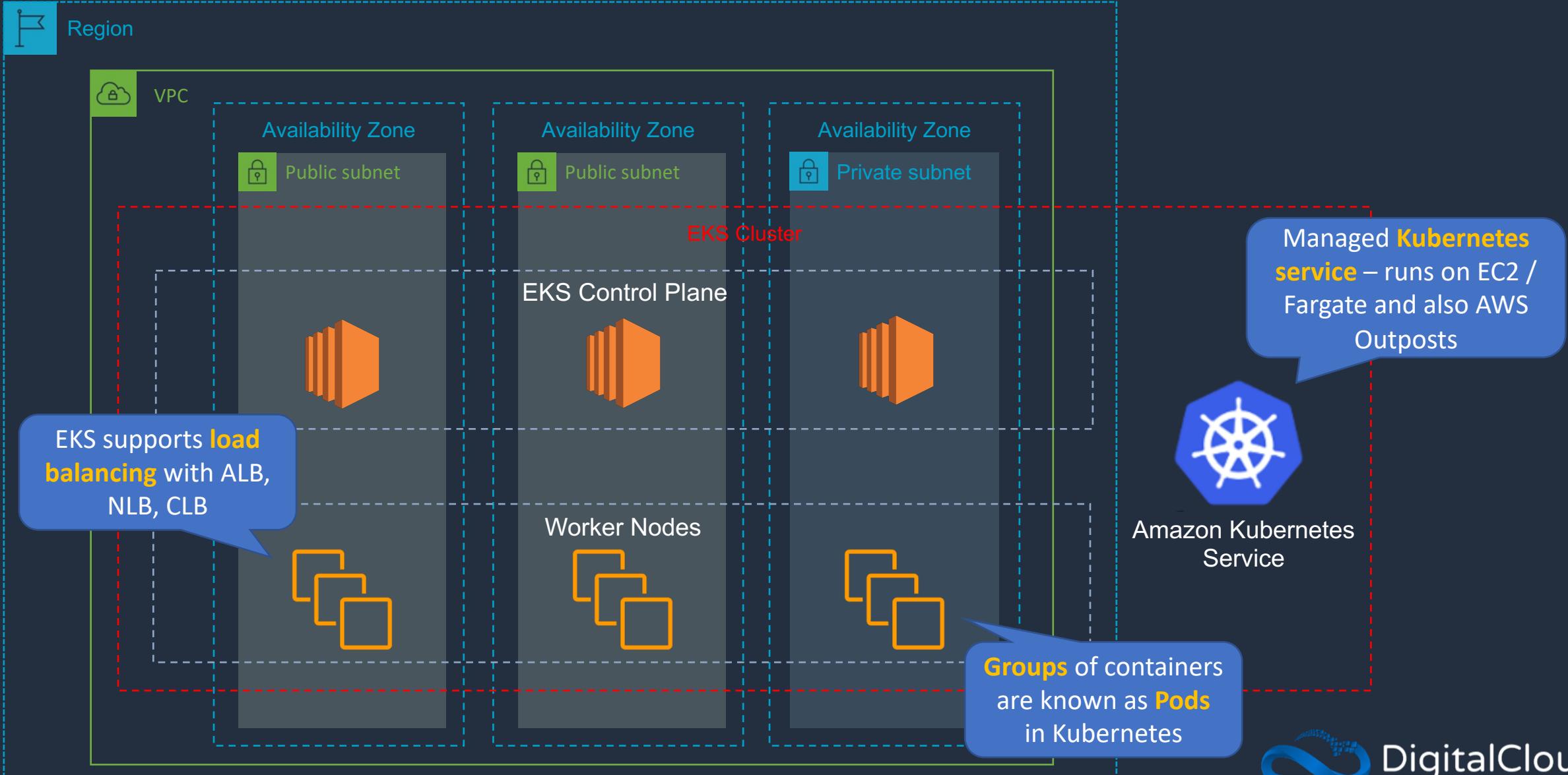


# Amazon Elastic Kubernetes Service (EKS)





# Amazon Elastic Kubernetes Service (EKS)





# Amazon EKS Use Cases

---

---

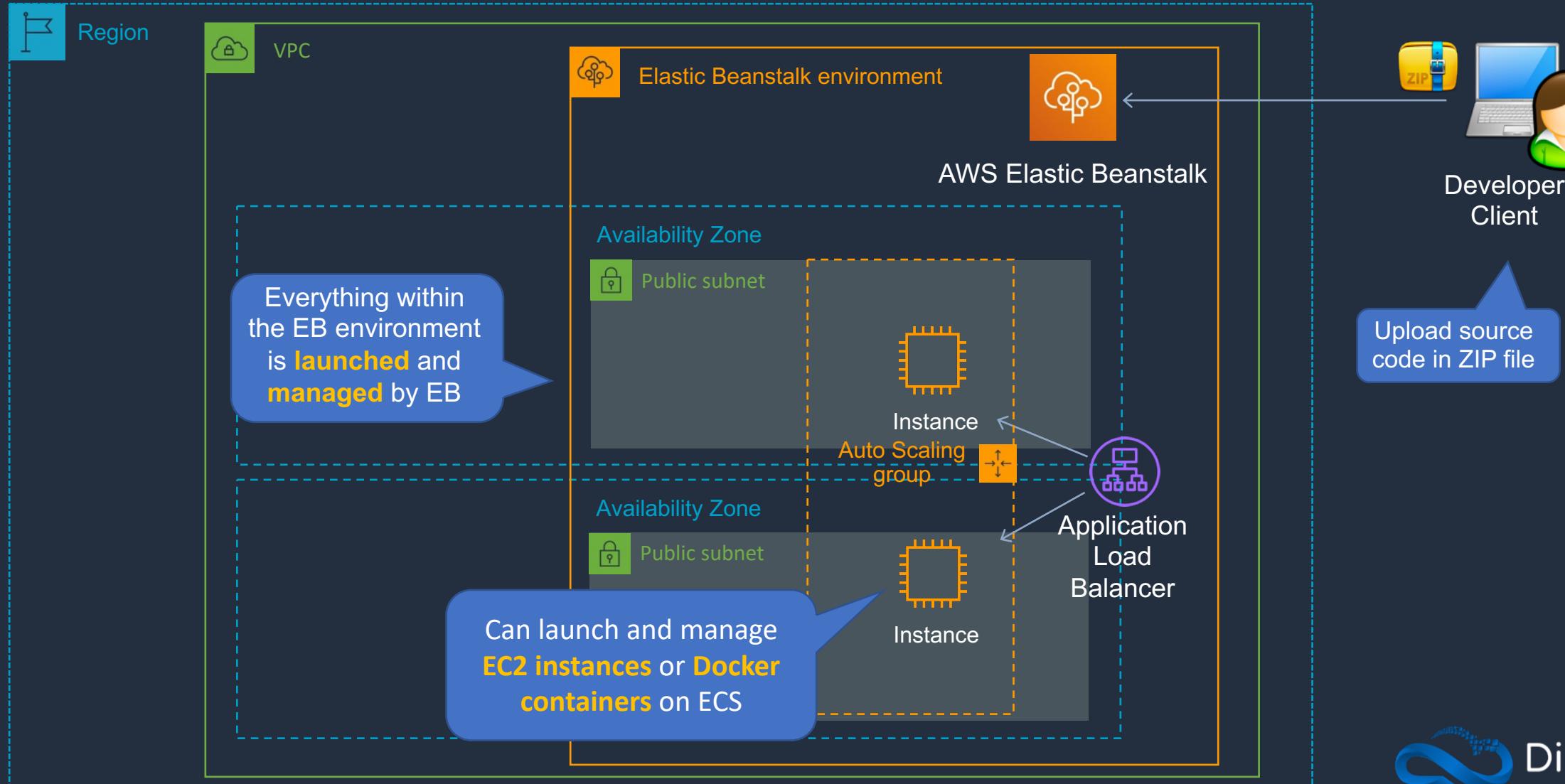
- Use when you need to **standardize** container orchestration across multiple environments using a **managed Kubernetes** implementation
- **Hybrid Deployment** - manage Kubernetes clusters and applications across hybrid environments (AWS + On-premises)
- **Batch Processing** - run sequential or parallel 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

# AWS Elastic Beanstalk Core Knowledge





# AWS Elastic Beanstalk





# 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



# 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



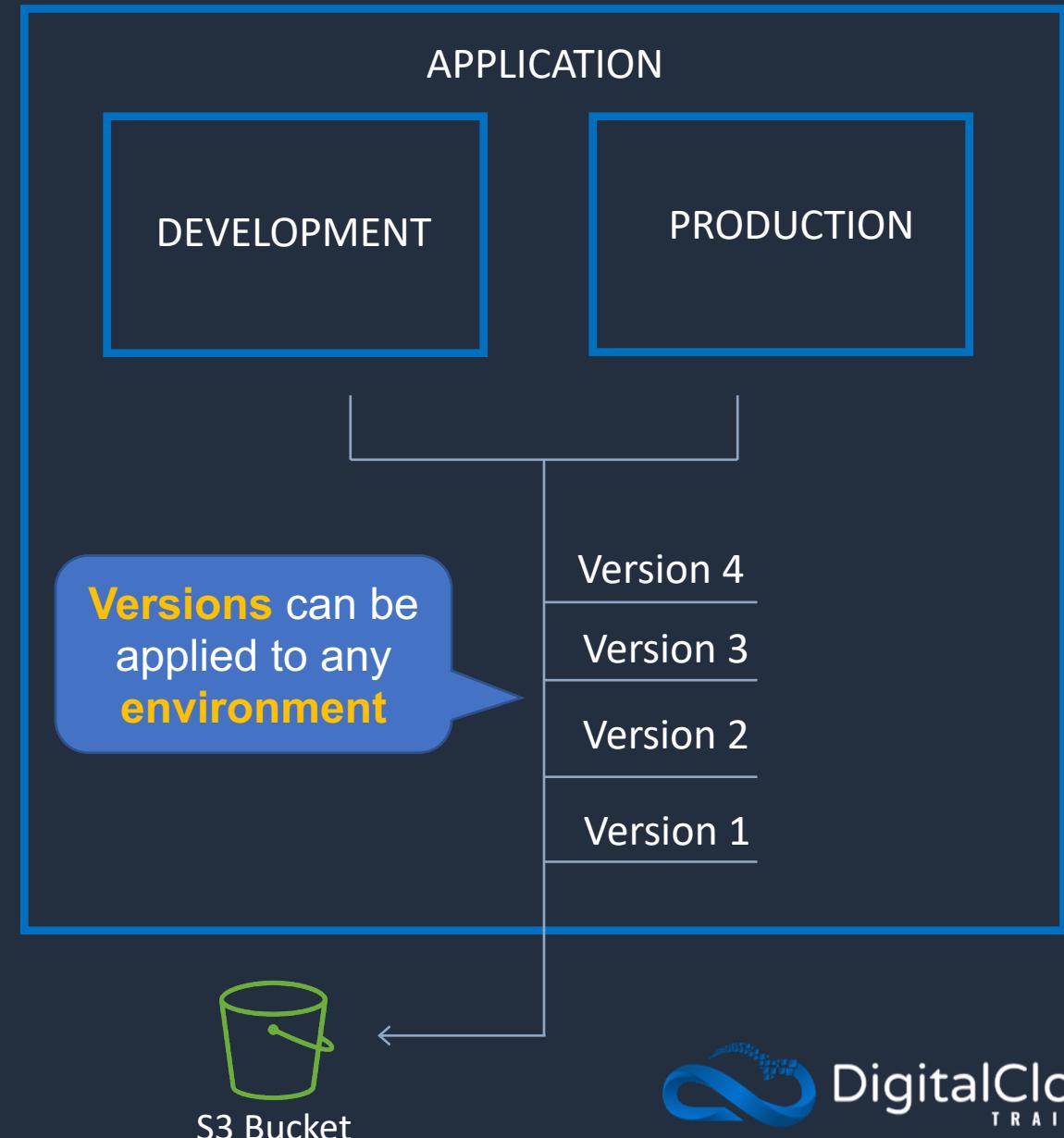
S3 Bucket



# 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





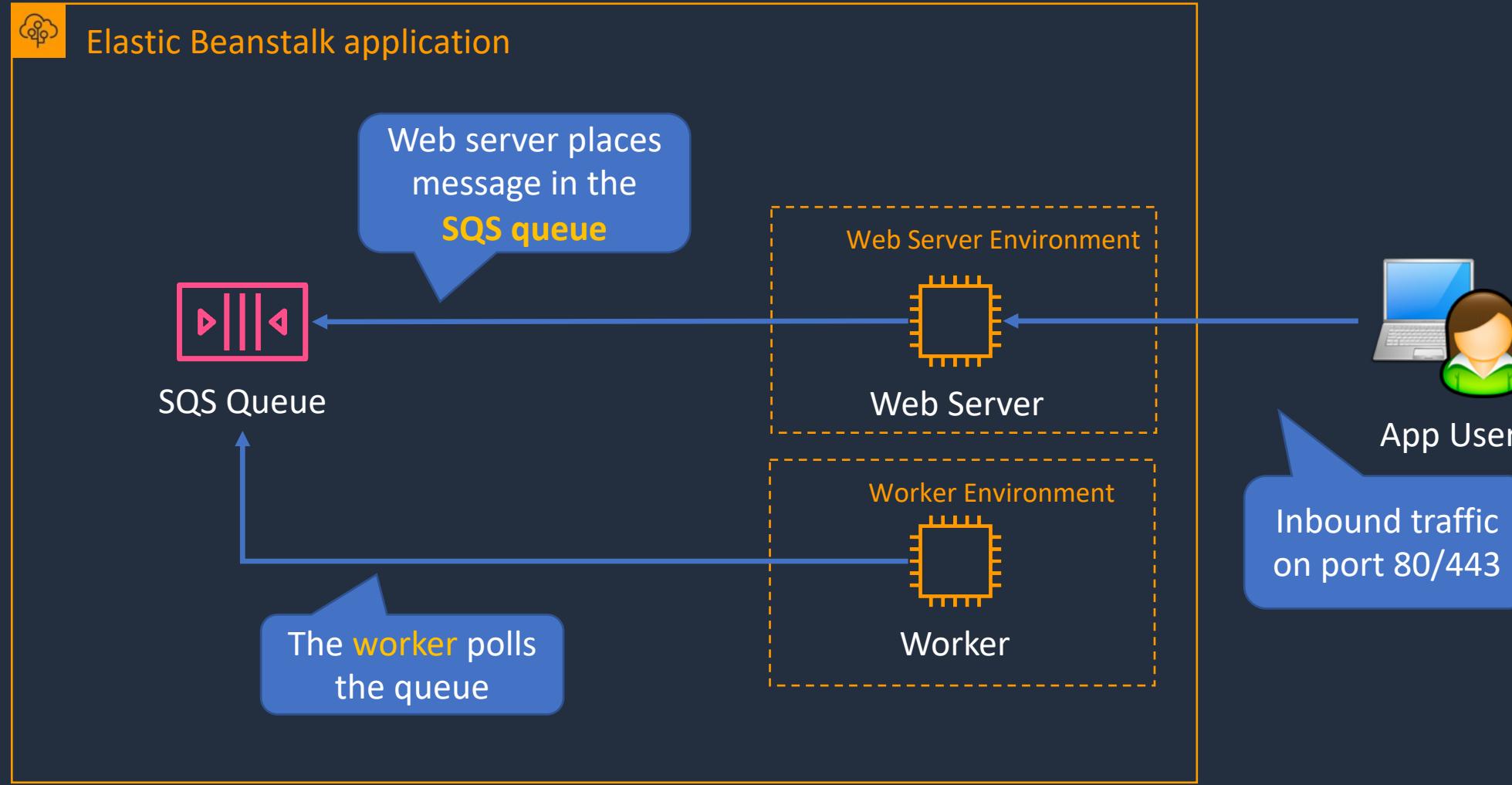
# Web Servers and Workers

---

- **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



# Updating Elastic Beanstalk Applications





# Elastic Beanstalk Deployment Policies

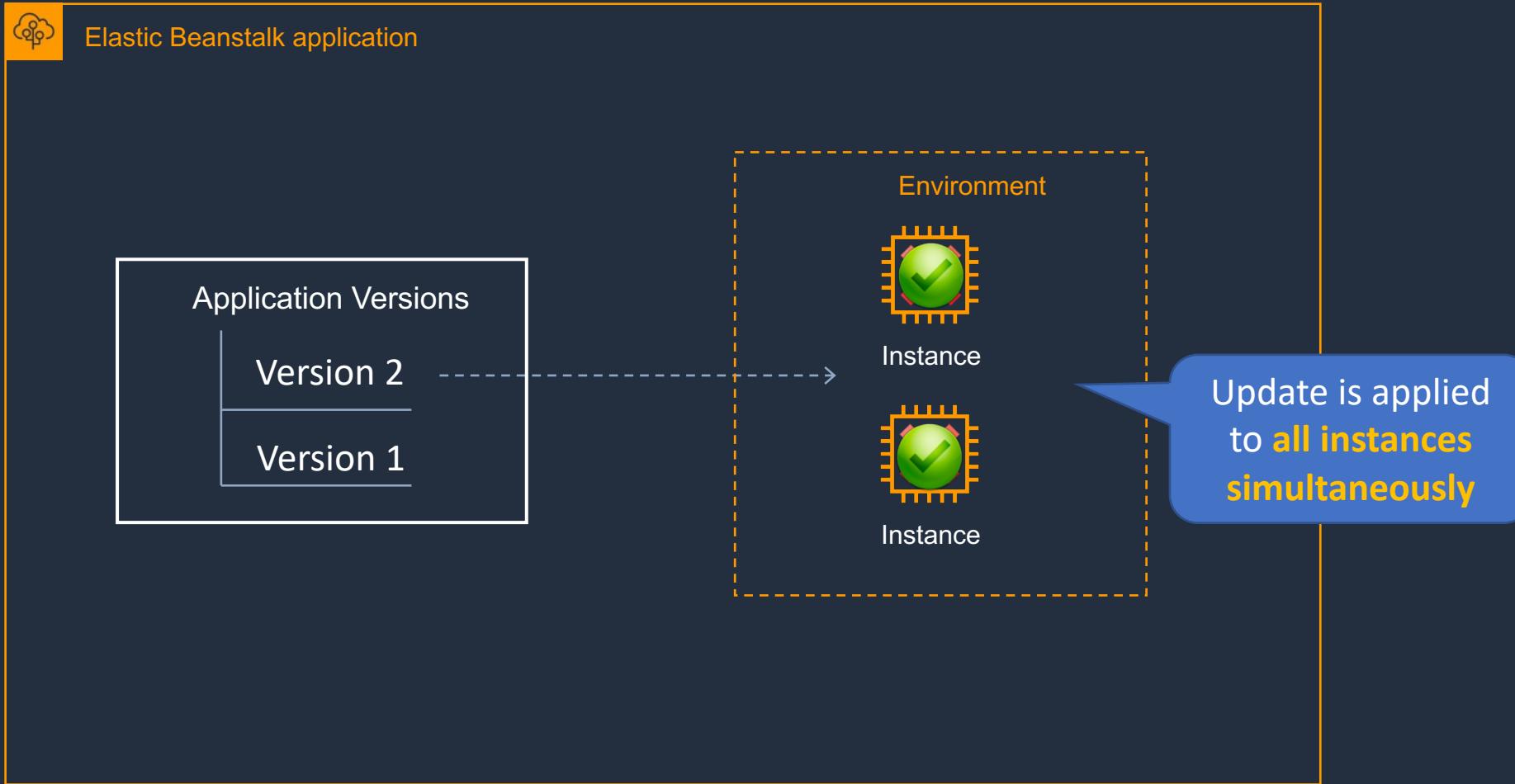
---

---

- **All at once** - Deploys the new version to all instances simultaneously
- **Rolling** - Update a batch of instances, and then move onto the next batch once the first batch is healthy
- **Rolling with additional batch** - Like Rolling but launches new instances in a batch ensuring that there is full availability
- **Immutable** - Launches new instances in a new ASG and deploys the version update to these instances before swapping traffic to these instances once healthy
- **Blue/green** - Create a new "stage" environment and deploy updates there



# All at once update





# All at once update

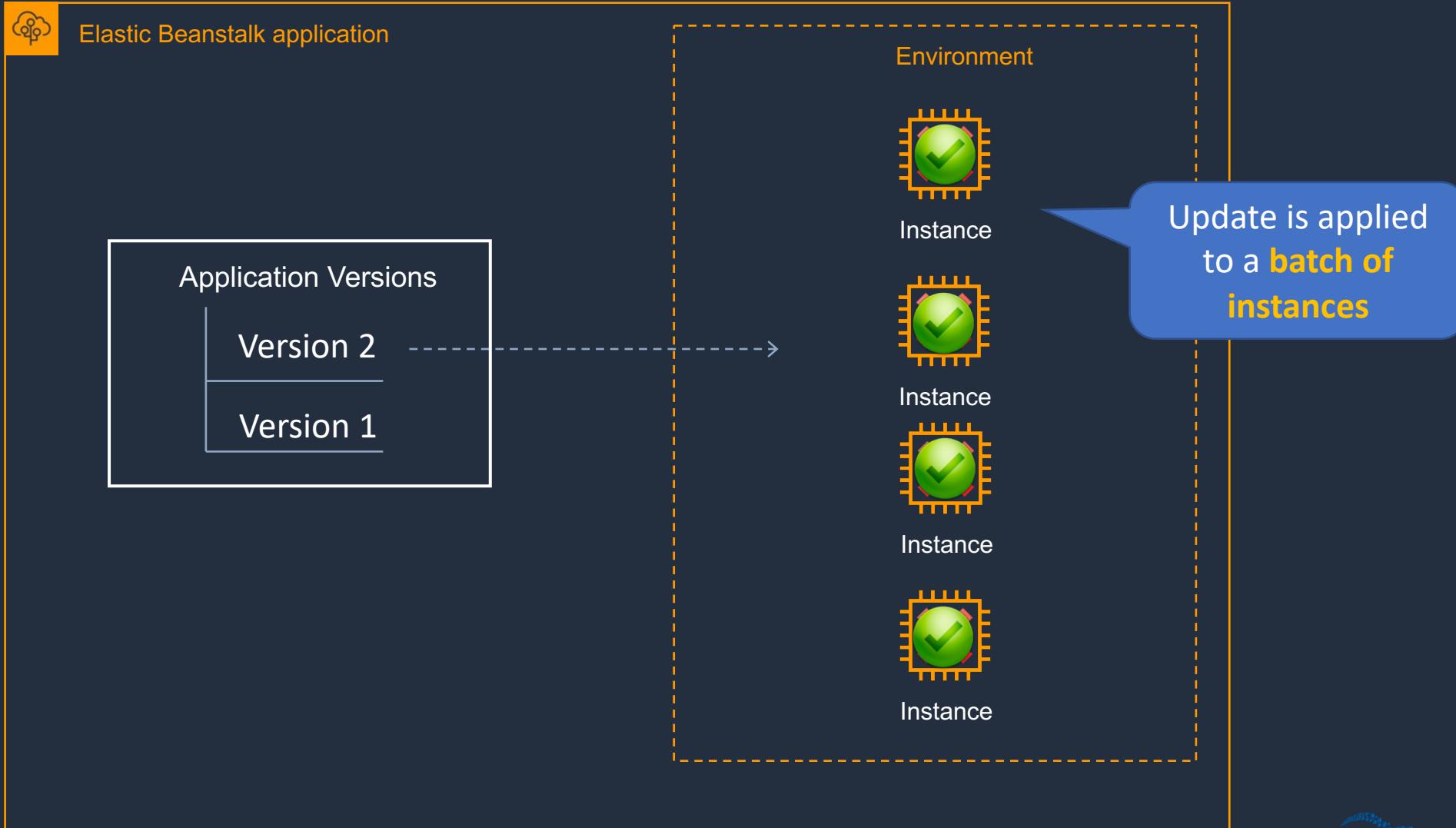
---

---

- Deploys the new version to all instances simultaneously
- All of your instances are out of service while the deployment takes place
- Fastest deployment
- Good for quick iterations in development environment
- You will experience an outage while the deployment is taking place - not ideal for mission-critical systems
- If the update fails, you need to roll back the changes by re-deploying the original version to all of your instances
- No additional cost



# Rolling Update





# Rolling Update

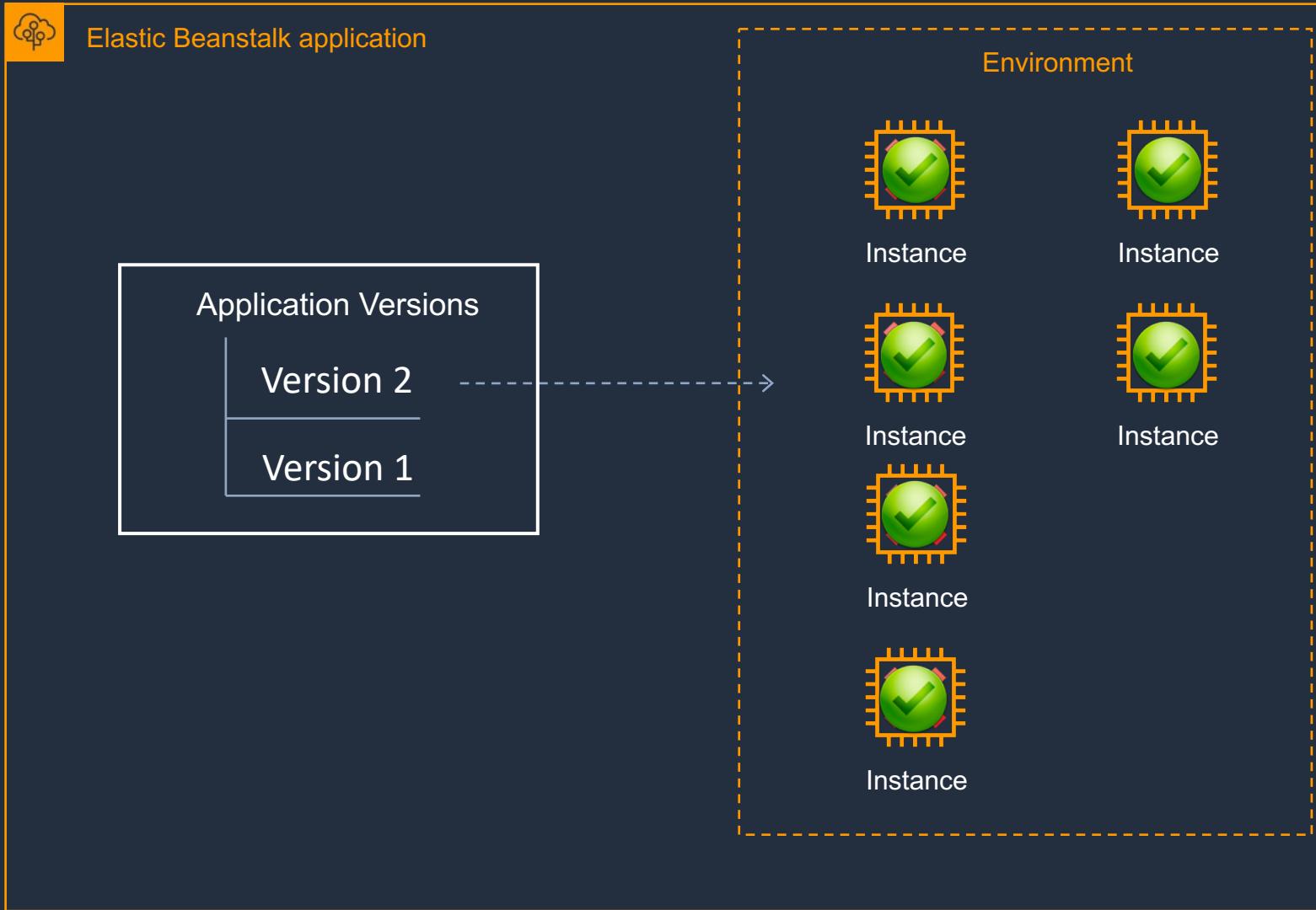
---

---

- Update a few instances at a time (batch), and then move onto the next batch once the first batch is healthy (downtime for 1 batch at a time)
- Application is running both versions simultaneously
- Each batch of instances is taken out of service while the deployment takes place
- Your environment capacity will be reduced by the number of instances in a batch while the deployment takes place
- Not ideal for performance-sensitive systems
- If the update fails, you need to perform an additional rolling update to roll back the changes
- No additional cost
- Long deployment time



# Rolling with Additional Batch Update





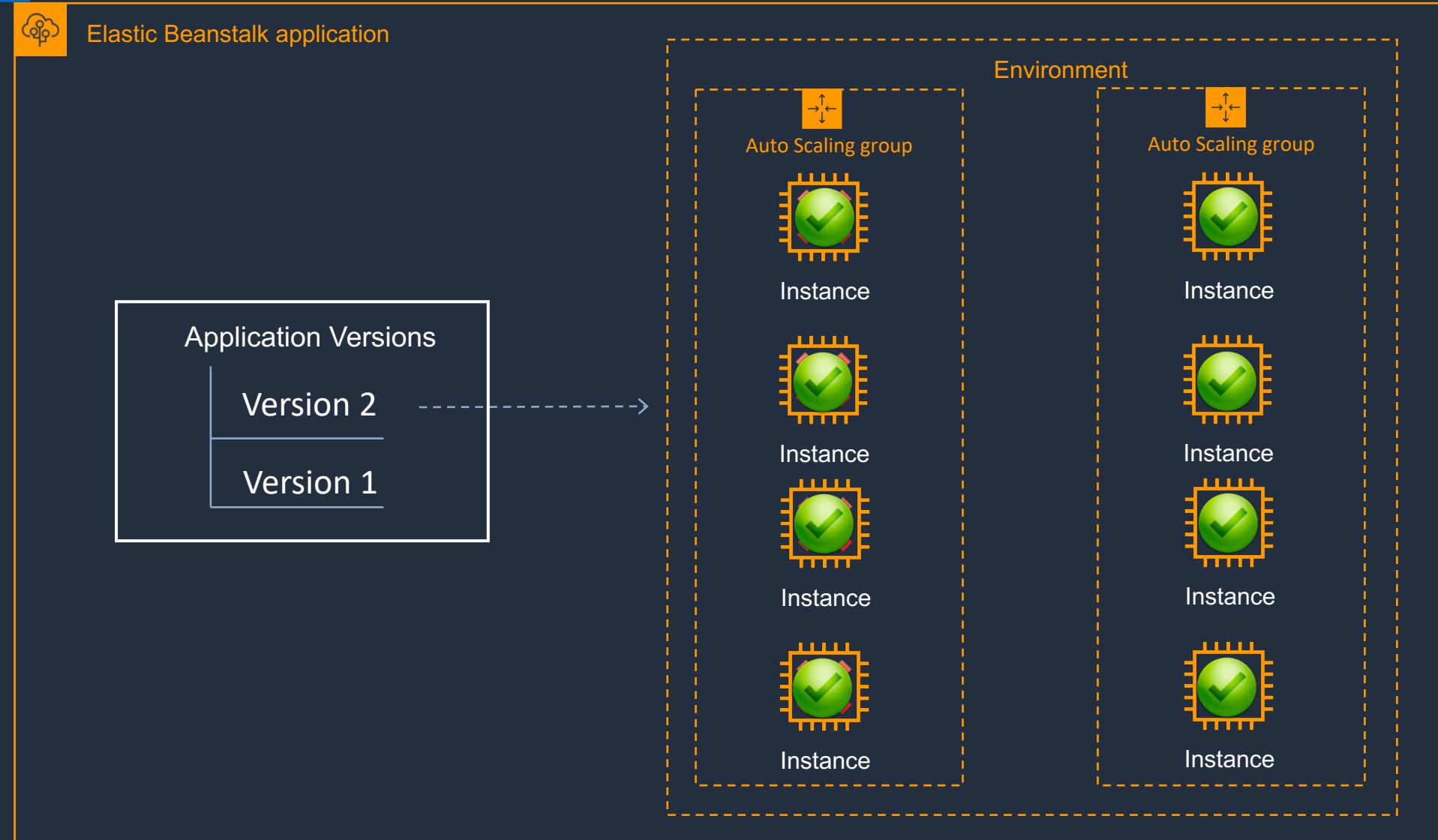
# Rolling with Additional Batch Update

---

- Like Rolling but launches new instances in a batch ensuring that there is full availability
- Application is running at capacity
- Can set the batch size
- Application is running both versions simultaneously
- Small additional cost
- Additional batch is removed at the end of the deployment
- Longer deployment
- Good for production environments



# Immutable Update





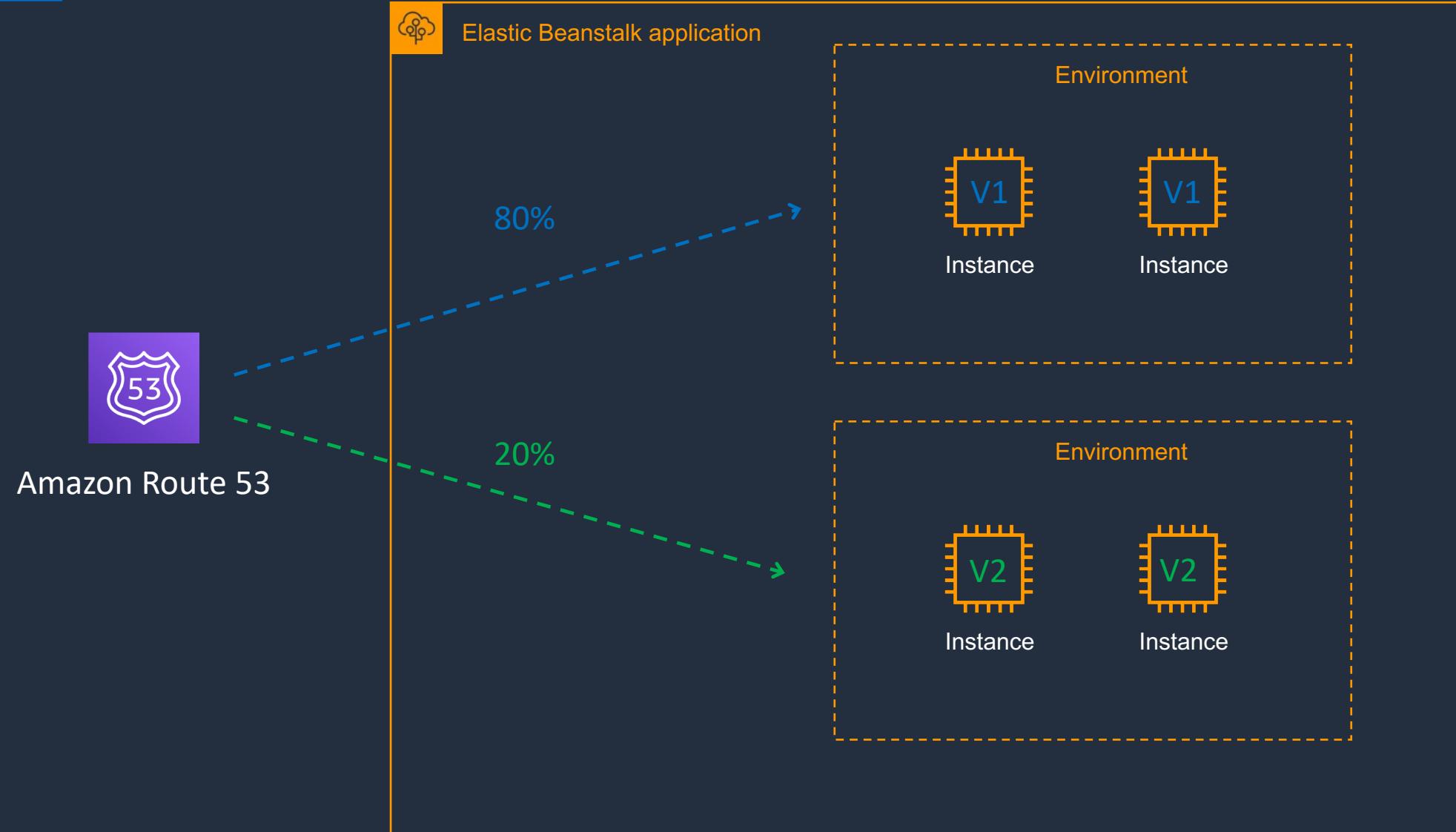
# Immutable Update

---

- Launches new instances in a new ASG and deploys the version update to these instances before swapping traffic to these instances once healthy
- Zero downtime
- New code is deployed to new instances using an ASG
- High cost as double the number of instances running during updates
- Longest deployment
- Quick rollback in case of failures
- Great for production environments



# Blue/Green Update





# Blue/Green Update

---

- This is not a feature within Elastic Beanstalk
- You create a new "staging" environment and deploy updates there
- The new environment (green) can be validated independently, and you can roll back if there are issues
- Route 53 can be setup using weighted policies to redirect a percentage of traffic to the staging environment
- Using Elastic Beanstalk, you can "swap URLs" when done with the environment test
- Zero downtime

# Deploy Elastic Beanstalk Apps



# Architecture Patterns – Containers and PaaS





# Architecture Patterns – Containers and PaaS

---

---

## Requirement

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

## Solution

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

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

Migrate to Amazon ECS using the Fargate launch type

Fargate task is launched in a private subnet and fails with error “CannotPullContainer”

Disable auto-assignment of public IP addresses and configure a NAT gateway



# Architecture Patterns – Containers and PaaS

---

---

## Requirement

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

## Solution

Use service auto-scaling and use the memory utilization

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

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

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

Deploy Amazon EKS



# Architecture Patterns – Containers and PaaS

---

---

## Requirement

Company needs to move many simple web apps running on PHP, Java, and Ruby to AWS. Utilization is very low

Business critical application running on Elastic Beanstalk must be updated. Require zero downtime and quick and complete rollback

A development application running on Elastic Beanstalk needs a cost-effective and quick update. Downtime is acceptable

## Solution

Deploy to single-instance Elastic Beanstalk environments

Update using an immutable update with a new ASG and swap traffic

Use an all-at-once update



## Requirement

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

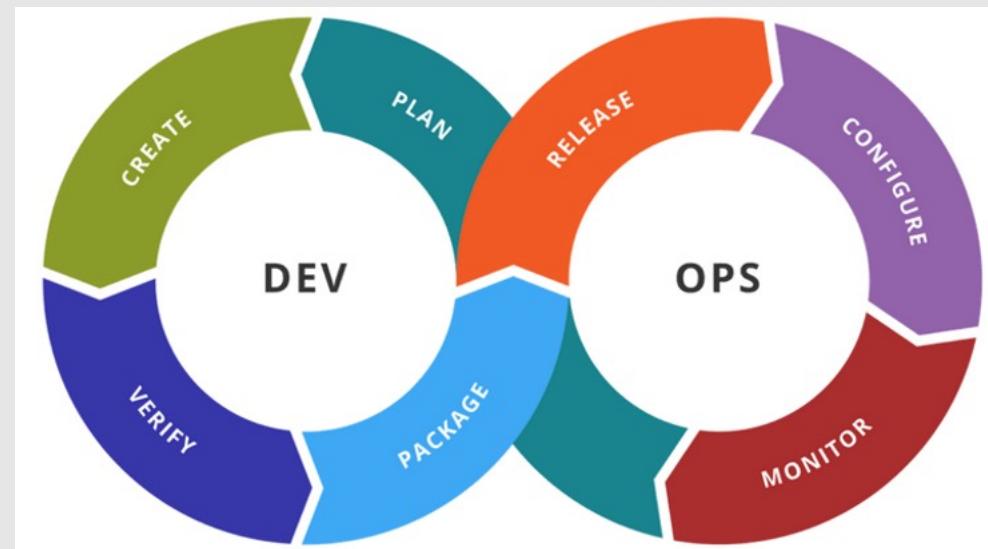
## Solution

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

# SECTION 13

## Deployment and Management

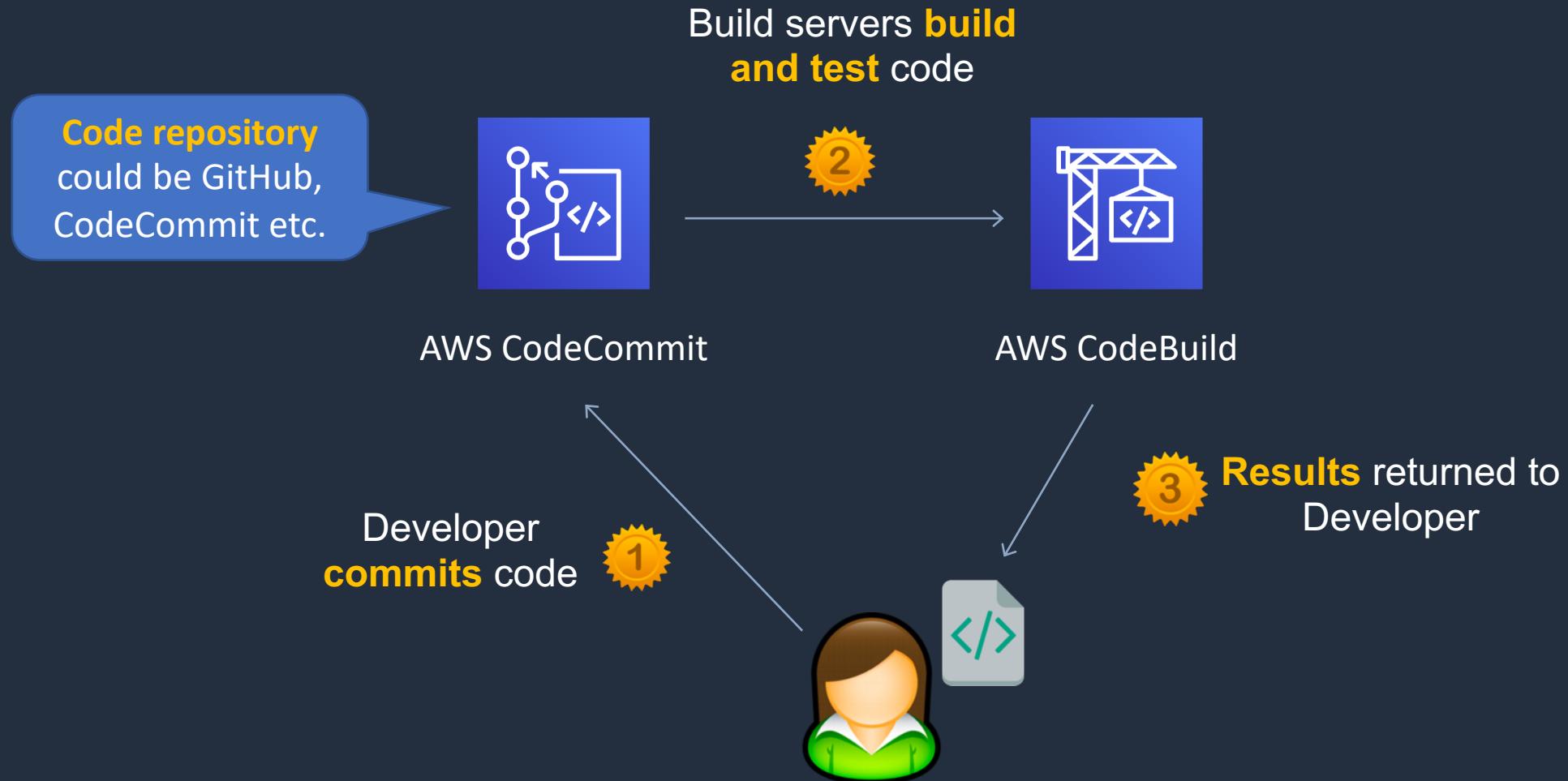
# CI/CD Overview





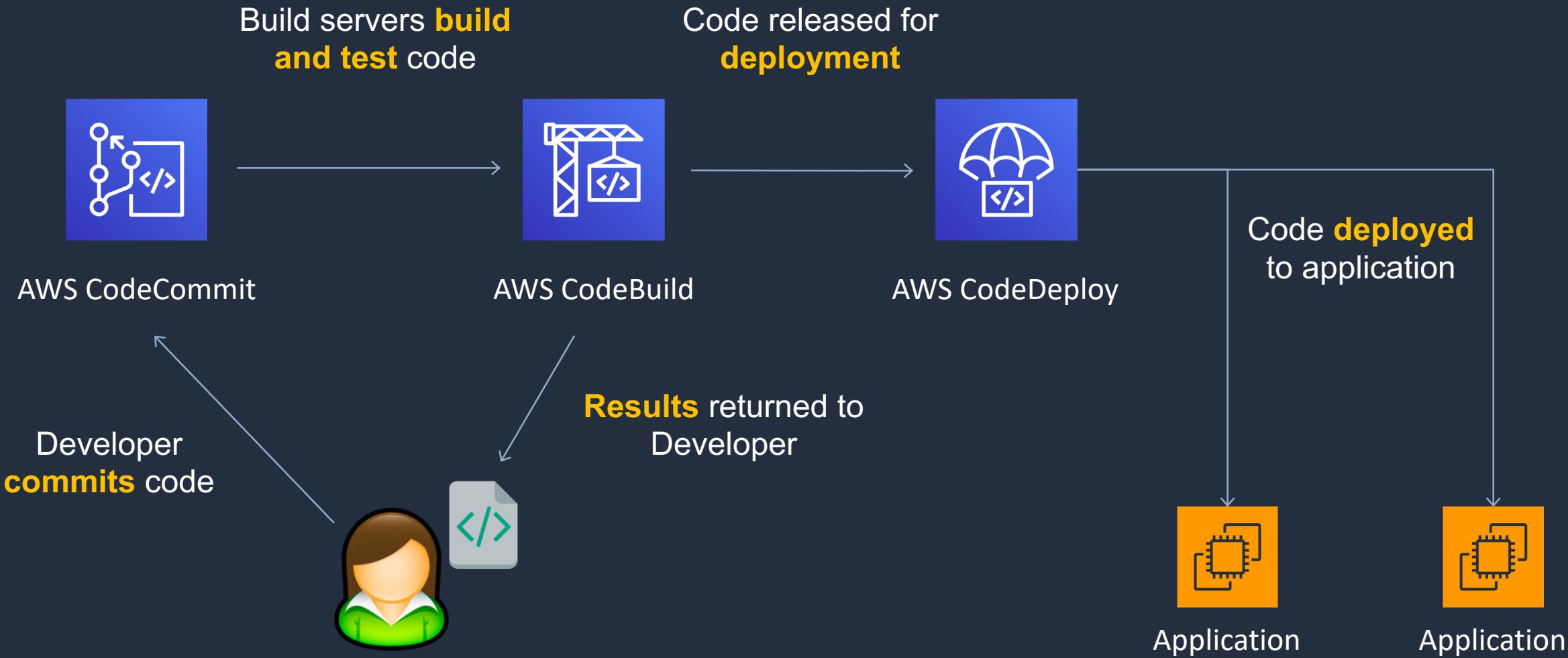
# Continuous Integration

---





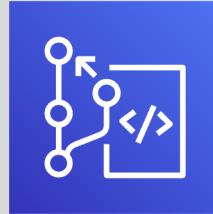
# Continuous Integration and Continuous Delivery





# Continuous Integration and Continuous Delivery

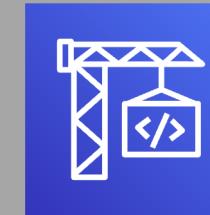
CODE



AWS CodeCommit

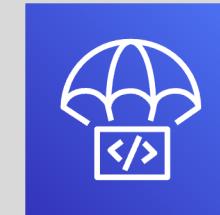
BUILD & TEST

AWS CodePipeline



AWS CodeBuild

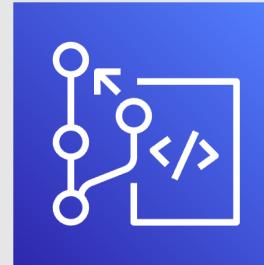
DEPLOY



AWS CodeDeploy



# AWS CodeCommit and CodePipeline

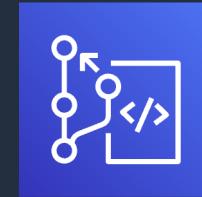




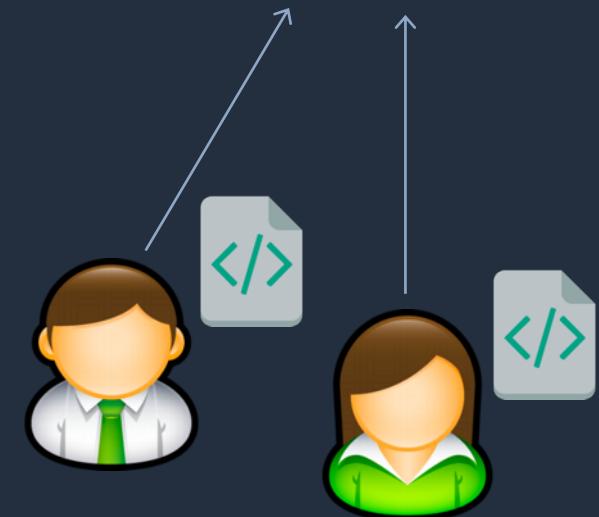
# AWS CodeCommit

---

- AWS CodeCommit is a fully-managed **source control** service that hosts secure Git-based repositories
- Git is an Open-Source distributed source control system:
  - Centralized repository for all of your code, binaries, images, and libraries
  - Tracks and manages code changes
  - Maintains version history
  - Manages updates from multiple sources
  - Enables collaboration



AWS CodeCommit

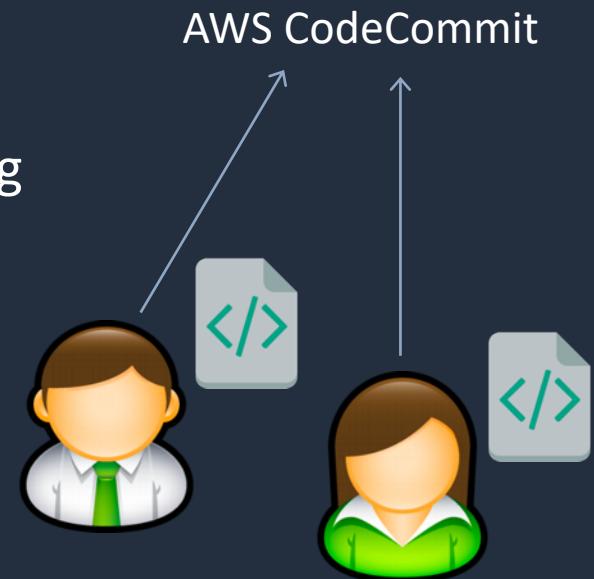
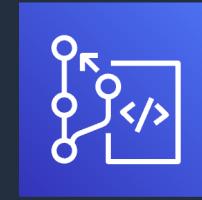




# AWS CodeCommit

---

- CodeCommit repositories are private
- CodeCommit scales seamlessly
- CodeCommit is integrated with Jenkins, CodeBuild and other CI tools
- You can transfer your files to and from AWS CodeCommit using HTTPS or SSH
- Repositories are automatically encrypted at rest through AWS Key Management Service (AWS KMS) using customer-specific keys





# AWS CodeCommit

---

---

- You need to configure your Git client to communicate with CodeCommit repositories
- IAM supports CodeCommit with three types of credentials:
  - **Git credentials** - an IAM-generated user name and password pair you can use to communicate with CodeCommit repositories over HTTPS
  - **SSH keys** - a locally generated public-private key pair that you can associate with your IAM user to communicate with CodeCommit repositories over SSH
  - **AWS access keys** - which you can use with the credential helper included with the AWS CLI to communicate with CodeCommit repositories over HTTPS



# AWS CodePipeline

---

---

- Fully managed continuous delivery service that helps you automate your release pipelines for fast and reliable application and infrastructure updates
- Automates the build, test, and deploy phases of your release process every time there is a code change, based on the release model you define
- CodePipeline provides tooling integrations for many AWS and third-party software at each stage of the pipeline including:
  - **Source stage** – S3, CodeCommit, Github, ECR, Bitbucket Cloud (beta).
  - **Build** – CodeBuild, Jenkins.
  - **Deploy stage** – CloudFormation, CodeDeploy, ECS, Elastic Beanstalk, AWS Service Catalog, S3.



# AWS CodePipeline

---

---

## Pipelines

- A workflow that describes how software changes go through the release process

## Artifacts

- Files or changes that will be worked on by the actions and stages in the pipeline
- Each pipeline stage can create "artifacts"
- Artifacts are passed, stored in Amazon S3 and then passed on to the next stage

## Stages

- Pipelines are broken up into stages
- Each stage can have sequential actions and or parallel actions
- Stage examples would be build, test, deploy, load test etc.
- Manual approval can be defined at any stage



# AWS CodePipeline

---

## Actions

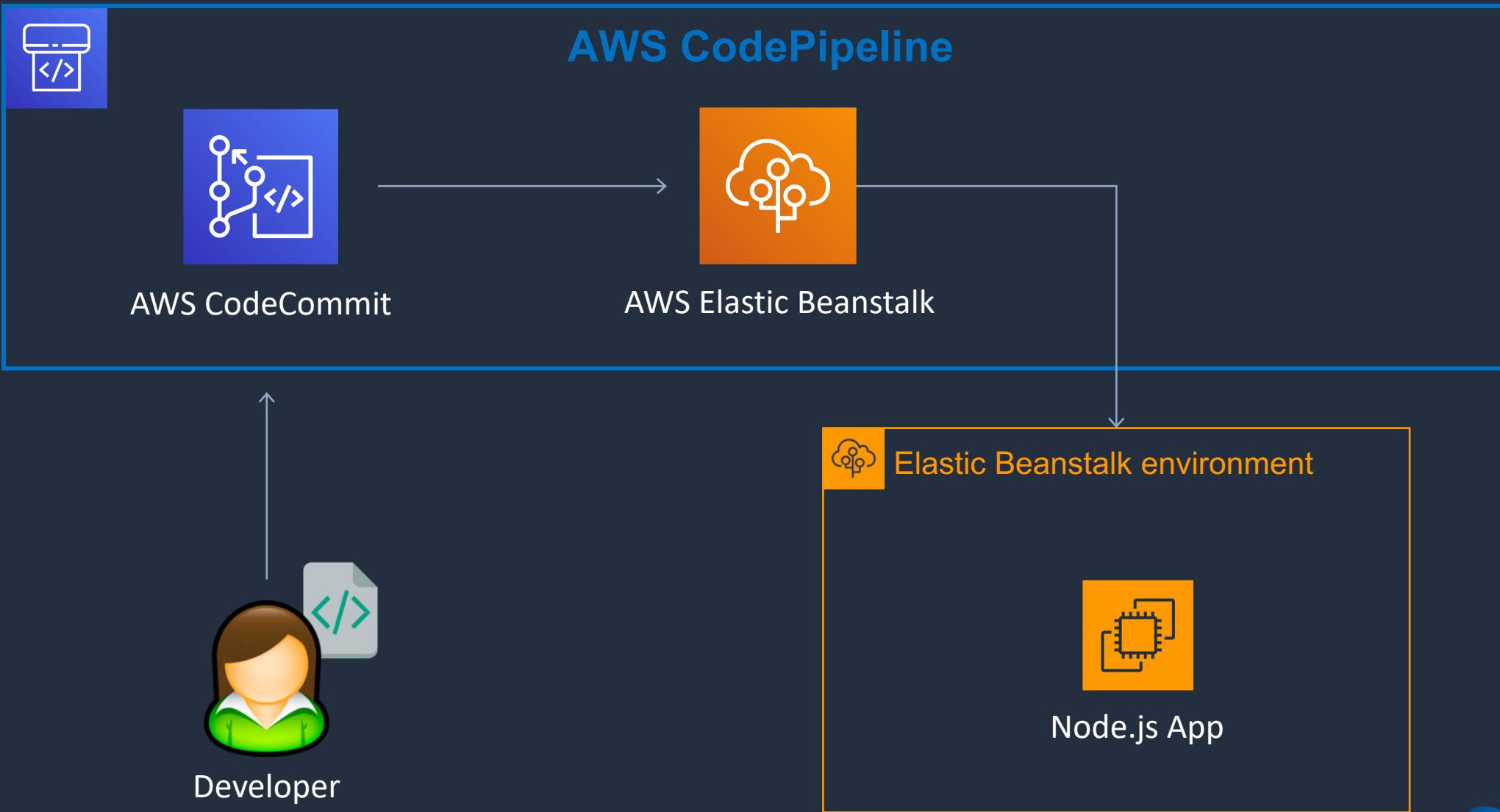
- Stages contain at least one action
- Actions affect artifacts and will have artifacts as either an input, and output, or both

## Transitions

- The progression from one stage to another inside of a pipeline



# CodePipeline with Elastic Beanstalk

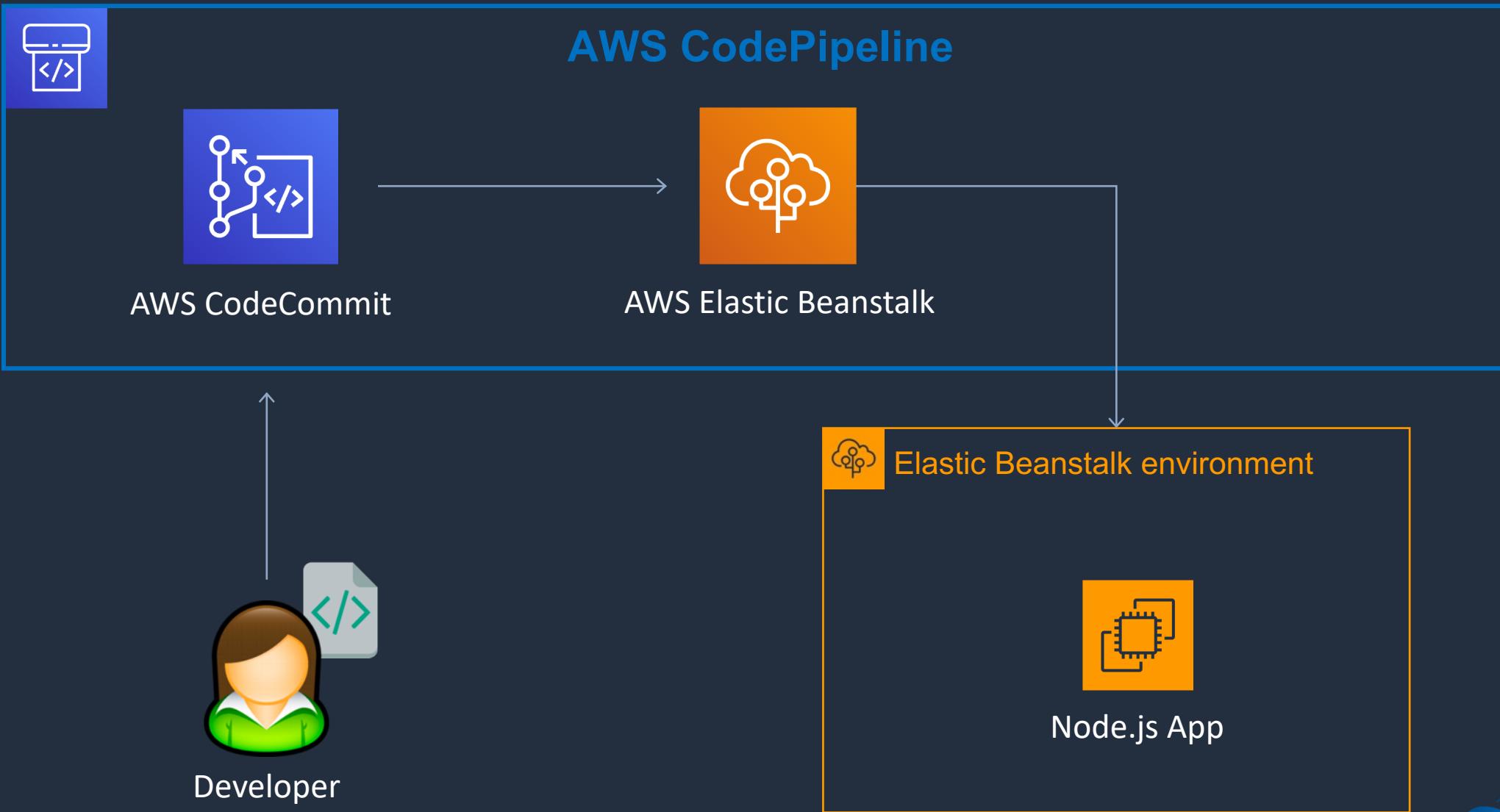


# Deploy App using CodePipeline

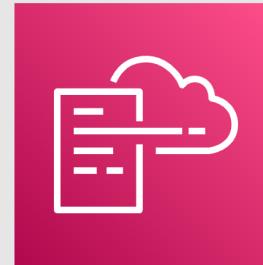




# CodePipeline with Elastic Beanstalk



# AWS CloudFormation Core Knowledge





# AWS CloudFormation

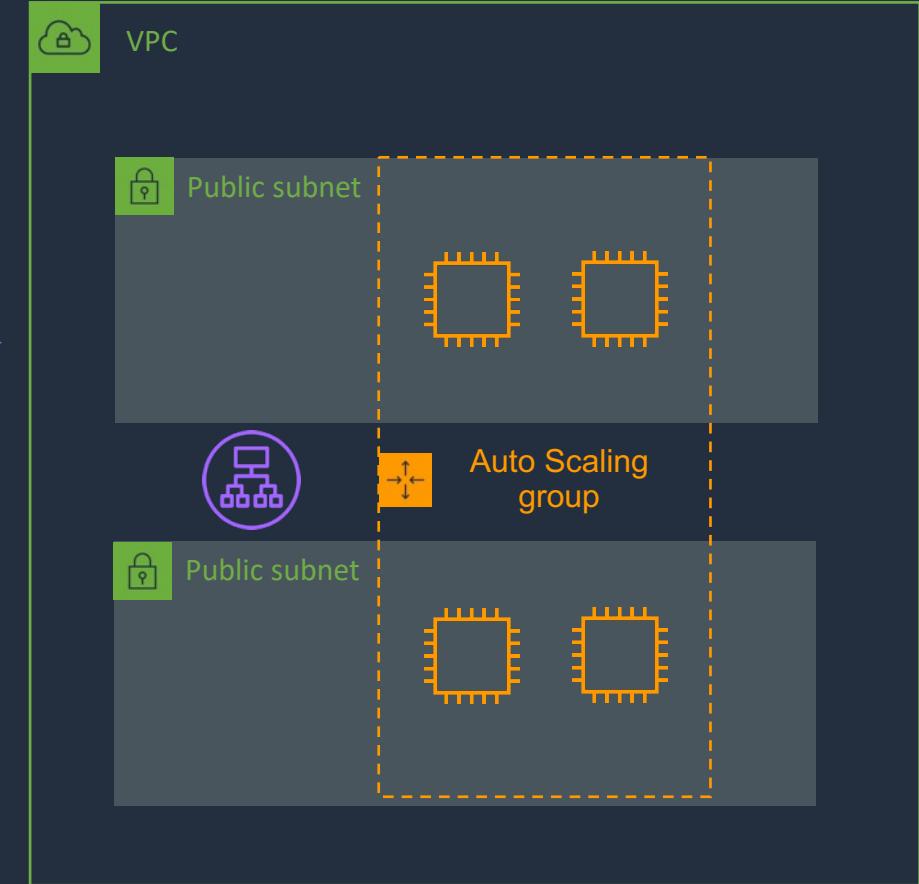
Infrastructure patterns are defined in a **template** file using **code**



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 },
```





# 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



# AWS CloudFormation - Templates

---

---

- A template is a YAML or JSON template used to describe the end-state of the infrastructure you are either provisioning or changing
- After creating the template, you upload it to CloudFormation directly or using Amazon S3
- CloudFormation reads the template and makes the API calls on your behalf.
- The resulting resources are called a "Stack"
- Logical IDs are used to reference resources within the template
- Physical IDs identify resources outside of AWS CloudFormation templates, but only after the resources have been created



# AWS CloudFormation - Stacks

---

- Deployed resources based on templates
- Create, update and delete stacks using templates
- Deployed through the Management Console, CLI or APIs
- Stack creation errors:
  - Automatic rollback on error is enabled by default
  - You will be charged for resources provisioned even if there is an error



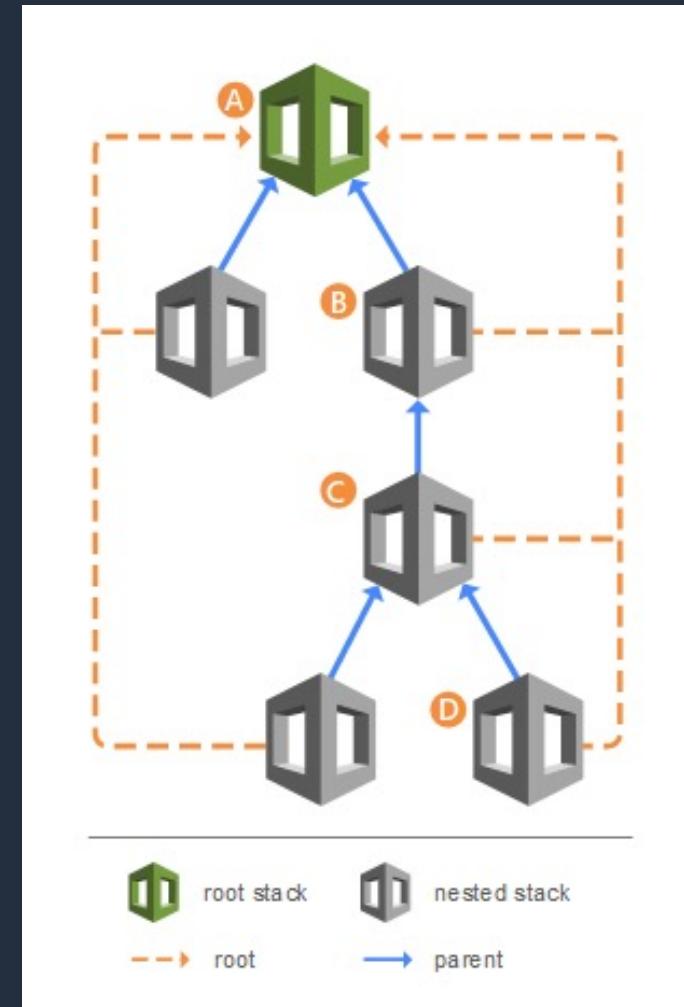
# AWS CloudFormation - Stacks

- 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
- Using an administrator account, you define and manage an AWS CloudFormation template, and use the template as the basis for provisioning stacks into selected target accounts across specified regions
- An administrator account is the AWS account in which you create stack sets
- A stack set is managed by signing in to the AWS administrator account in which it was created
- A target account is the account into which you create, update, or delete one or more stacks in your stack set



# AWS CloudFormation - NestedStacks

- Nested stacks allow re-use of CloudFormation code for common use cases
- For example standard configuration for a load balancer, web server, application server etc.
- Instead of copying out the code each time, create a standard template for each common use case and reference from within your CloudFormation template





# AWS CloudFormation - Change Sets

- AWS CloudFormation provides two methods for updating stacks:  
direct update or creating and executing change sets
- When you directly update a stack, you submit changes and AWS CloudFormation immediately deploys them
- Use direct updates when you want to quickly deploy your updates
- With change sets, you can preview the changes AWS CloudFormation will make to your stack, and then decide whether to apply those changes

# Deploy Stack and Change Set



# AWS Service Catalog





# AWS Service Catalog

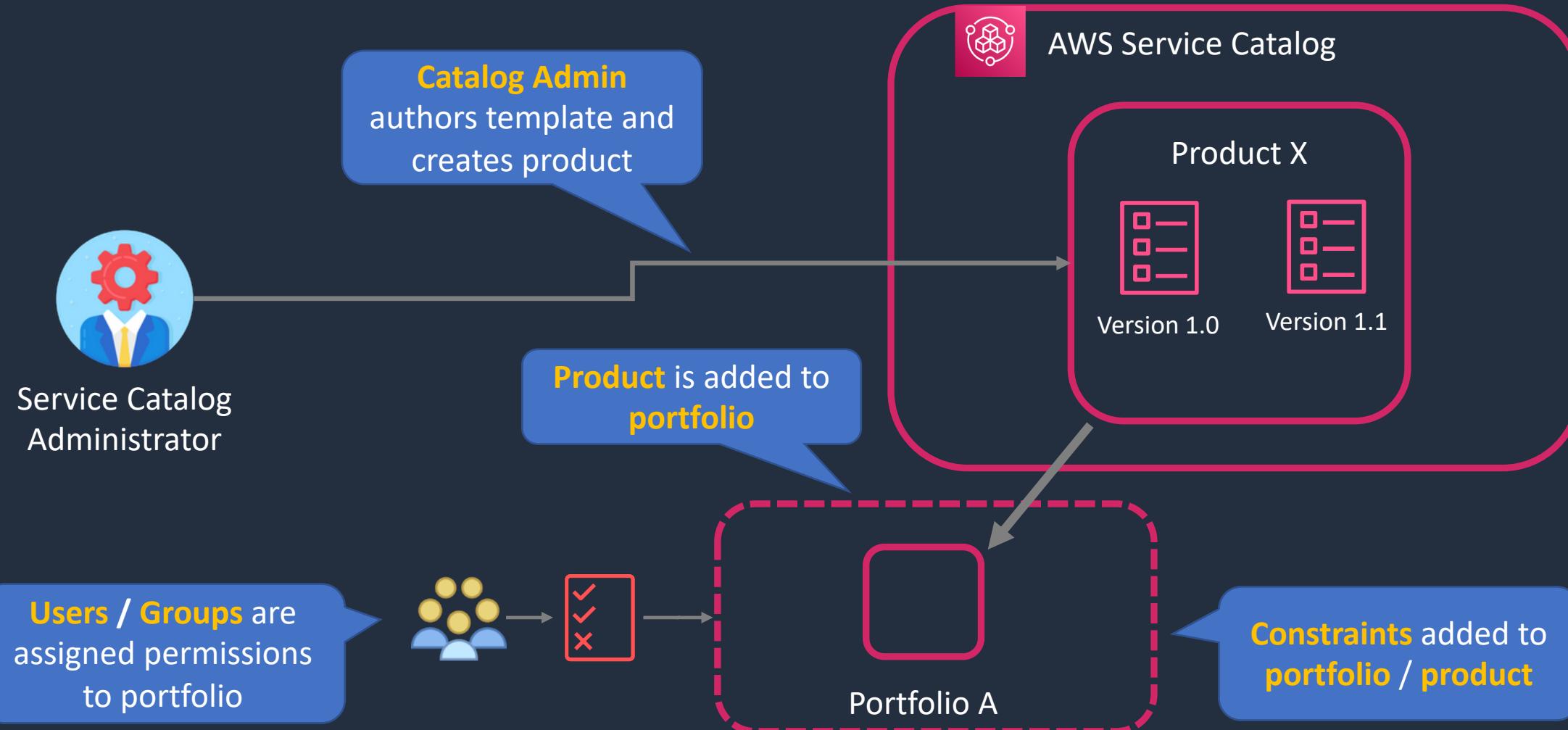
---

---

- AWS Service Catalog allows organizations to create and manage **catalogs of IT services** that are approved for use on AWS
- IT services can include everything from virtual machine images, servers, software, and databases to complete multi-tier application architectures
- AWS Service Catalog allows you to **centrally manage** commonly deployed IT services
- Enables users to quickly deploy only the approved IT services they need



# AWS Service Catalog





# AWS Service Catalog – Users

---

---

## Catalog administrators

- Manage a catalog of products, organizing them into portfolios and granting access to end users

## End users

- Receive AWS credentials use the AWS Management Console to launch products to which they have been granted access



# AWS Service Catalog – Constraints

---

---

- Constraints control the ways that specific AWS resources can be deployed for a product.
- You can use them to apply limits to products for governance or cost control.
- **Launch constraints** - specify a role for a product in a portfolio. The role is used to provision the resources at launch, so you can restrict user permissions without impacting users' ability to provision products from the catalog
- **Notification constraints** - enable you to get notifications about stack events using an Amazon SNS topic
- **Template constraints** - restrict the configuration parameters that are available for the user when launching the product



# AWS Service Catalog – Sharing

---

- You can share portfolios across accounts in AWS Organizations
- Either share a reference to the catalog or deploy a copy of the catalog
- With copies you must redeploy any updates
- CloudFormation StackSets can be used to deploy a catalog to multiple accounts at the same time
- Check reference link for more information on the behavior

# Deploy Product using Service Catalog



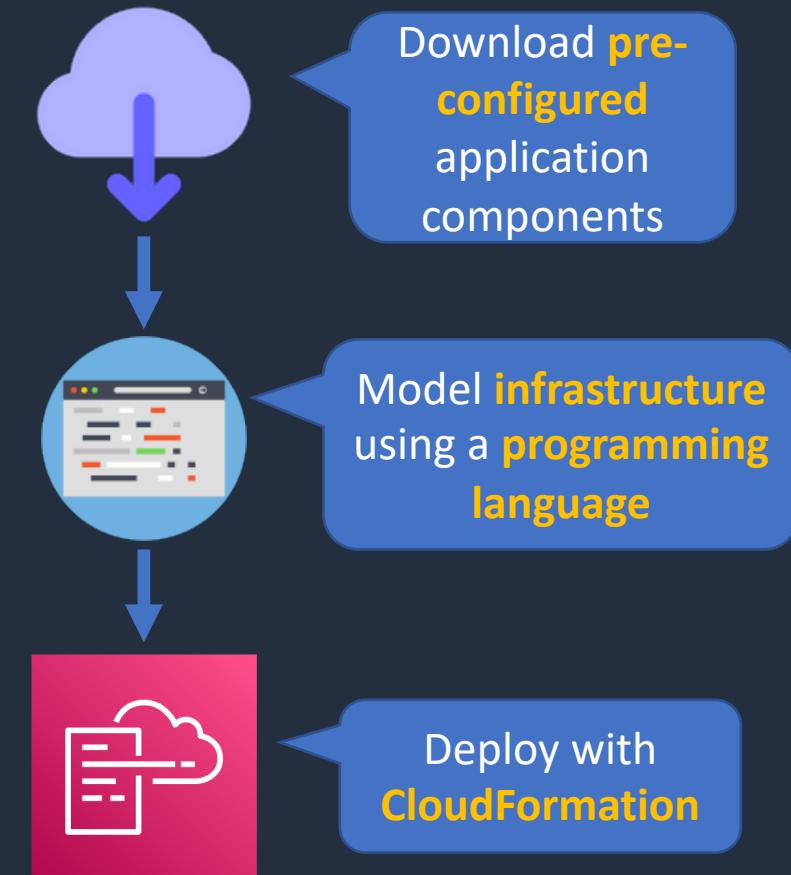
# AWS Cloud Development Kit





# AWS Cloud Development Kit

- Open source software development framework to define your cloud application resources using **familiar programming languages**
- Preconfigures cloud resources with proven defaults using **constructs**
- Provisions your resources using **AWS CloudFormation**
- Enables you to model application infrastructure using TypeScript, Python, Java, and .NET
- Use existing IDE, testing tools, and workflow patterns



# AWS Serverless Application Model (SAM)





# AWS SAM

- Provides a **shorthand syntax** to express functions, APIs, databases, and event source mappings
- Can be used to create Lambda functions, API endpoints, DynamoDB tables, and other resources

```
AWSTemplateFormatVersion: '2010-09-09'  
Transform: AWS::Serverless-2016-10-31  
Resources:  
  HelloWorldFunction:  
    Type: AWS::Serverless::Function  
    Properties:  
      CodeUri: hello-world/  
      Handler: app.lambdaHandler  
      Runtime: nodejs12.x
```

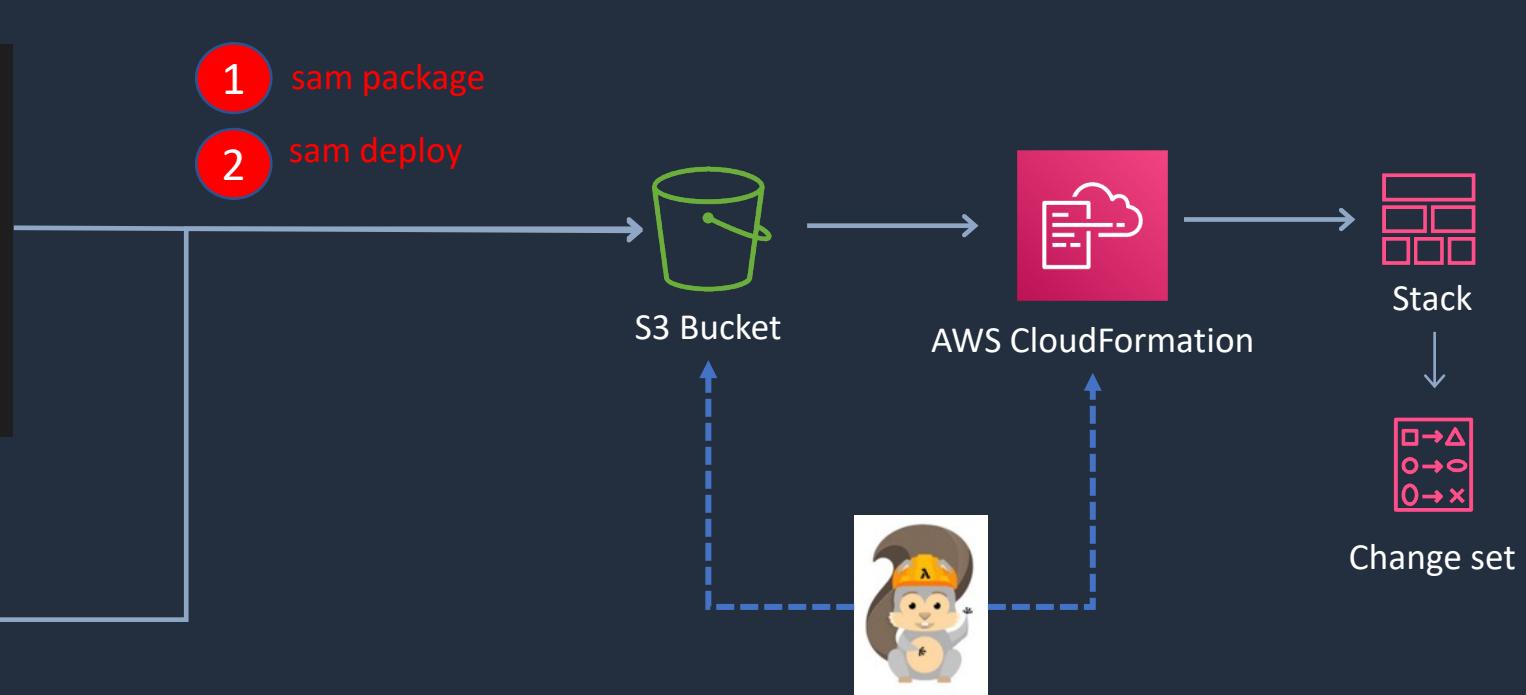
SAM Template (YAML)



Application Code

Commands:

- 1 sam package
- 2 sam deploy



SAM uploads source files to bucket  
and initiates CloudFormation



Change set



# AWS SAM

---

---

- A SAM template file is a **YAML** configuration that represents the architecture of a serverless application
- You use the template to declare all of the AWS resources that comprise your serverless application in one place
- AWS SAM templates are an extension of **AWS CloudFormation** templates, so any resource that you can declare in an AWS CloudFormation template can also be declared in an AWS SAM template



# AWS SAM

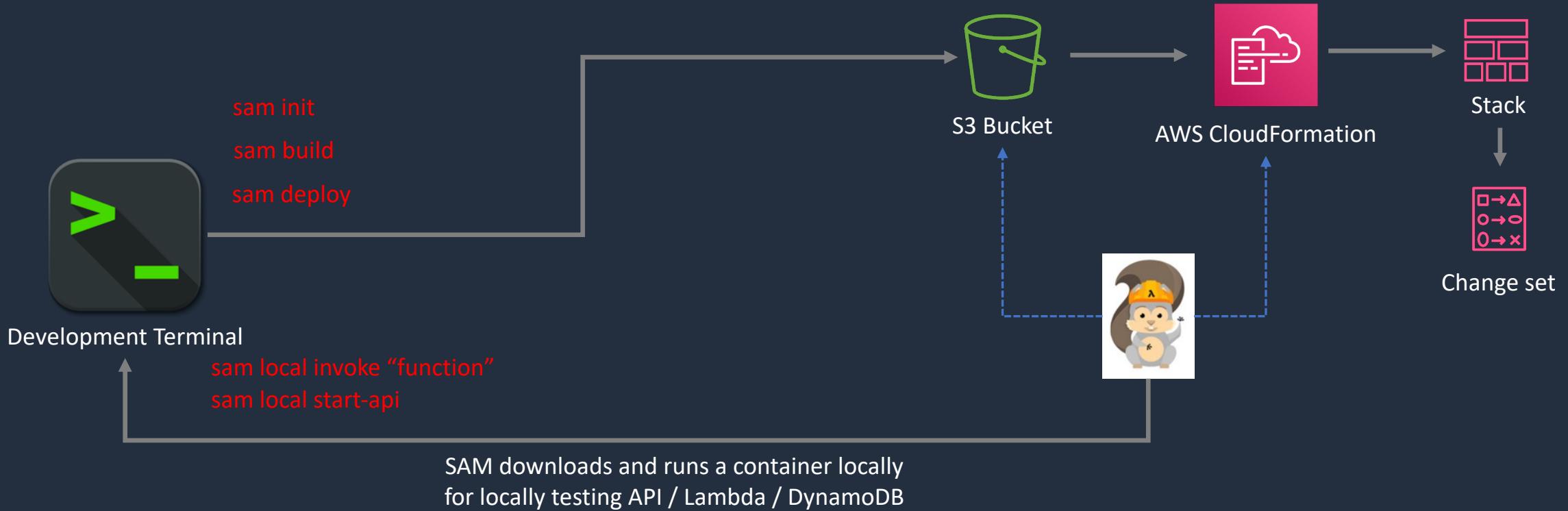
---

---

- The “Transform” header indicates it’s a SAM template:
  - Transform: 'AWS::Serverless-2016-10-31'
- There are several resources types:
  - AWS::Serverless::Function (AWS Lambda)
  - AWS::Serverless::Api (API Gateway)
  - AWS::Serverless::SimpleTable (DynamoDB)
  - AWS::Serverless::Application (AWS Serverless Application Repository)
  - AWS::Serverless::HttpApi (API Gateway HTTP API)
  - AWS::Serverless::LayerVersion (Lambda layers)



# AWS SAM



# AWS Systems Manager





# AWS Systems Manager

---

---

- Manages many AWS resources including Amazon EC2, Amazon S3, Amazon RDS etc.
- Systems Manager Components:
  - Automation
  - Run Command
  - Inventory
  - Patch Manager
  - Session Manager
  - Parameter Store



# AWS Systems Manager Automation

**Documents** define the actions to perform (YAML or JSON)

Automates **IT operations** and **management** tasks across AWS resources



Documents



Automation



Amazon RDS

```
description: Creates an RDS Snapshot for an RDS instance.
schemaVersion: '0.3'
assumeRole: "{{AutomationAssumeRole}}"
parameters:
  DBInstanceIdentifier:
    type: String
    description: (Required) The DBInstanceId ID of the RDS Instance.
  DBSnapshotIdentifier:
    type: String
    description: (Optional) The DBSnapshotIdentifier ID.
    default: ''
  InstanceTags:
    type: String
    default: ''
    description: (Optional) Tags to create for instance.
  SnapshotTags:
    type: String
    default: ''
    description: (Optional) Tags to create for snapshot
```

This automation, takes a snapshot of an RDS DB instance



# AWS Systems Manager Run Command

Document types include  
**command, automation, package** etc.

Runs commands on  
managed EC2 instances

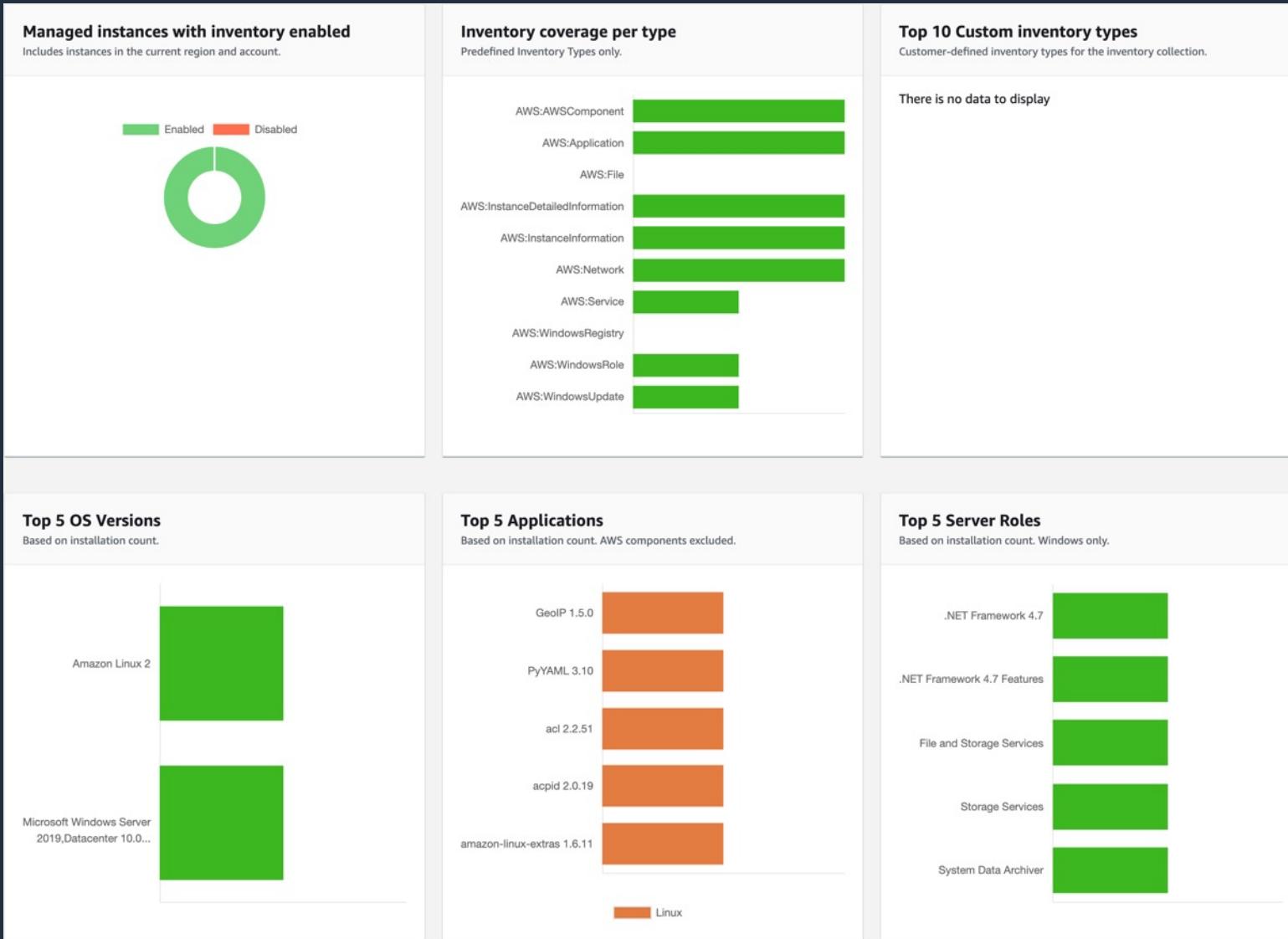


```
{  
  "schemaVersion": "1.2",  
  "description": "List missing Microsoft Windows updates.",  
  "parameters": {  
    "UpdateLevel": {  
      "type": "String",  
      "default": "Important",  
      "description": "Important: .....",  
      "allowedValues": [  
        "None",  
        "All",  
        "Important",  
        "Optional"  
      ]  
    }  
  },  
  "maxConcurrency": 100,  
  "maxErrors": 0  
}
```

This command checks  
for missing updates



# AWS Systems Manager Inventory



Inventory



# AWS Systems Manager Patch Manager

- Helps you select and deploy operating system and software patches automatically across large groups of Amazon EC2 or on-premises instances
- Patch baselines:
  - Set rules to auto-approve select categories of patches to be installed
  - Specify a list of patches that override these rules and are automatically approved or rejected
- You can also schedule maintenance windows for your patches so that they are only applied during predefined times
- Systems Manager helps ensure that your software is up-to-date and meets your compliance policies



Patch Manager



# AWS Systems Manager Compliance

- AWS Systems Manager lets you scan your managed instances for patch compliance and configuration inconsistencies
- You can collect and aggregate data from multiple AWS accounts and Regions, and then drill down into specific resources that aren't compliant
- By default, AWS Systems Manager displays data about patching and associations
- You can also customize the service and create your own compliance types based on your requirements (must use the AWS CLI, AWS Tools for Windows PowerShell, or the SDKs)



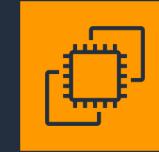
# AWS Systems Manager Session Manager

- Secure remote management of your instances at scale without logging into your servers
- Replaces the need for bastion hosts, SSH, or remote PowerShell
- Integrates with IAM for granular permissions
- All actions taken with Systems Manager are recorded by AWS CloudTrail
- Can store session logs in an S3 and output to CloudWatch Logs
- Requires IAM permissions for EC2 instance to access SSM, S3, and CloudWatch Logs

Doesn't require port 22,5985/5986



No need for bastion hosts



Amazon EC2  
(Linux)

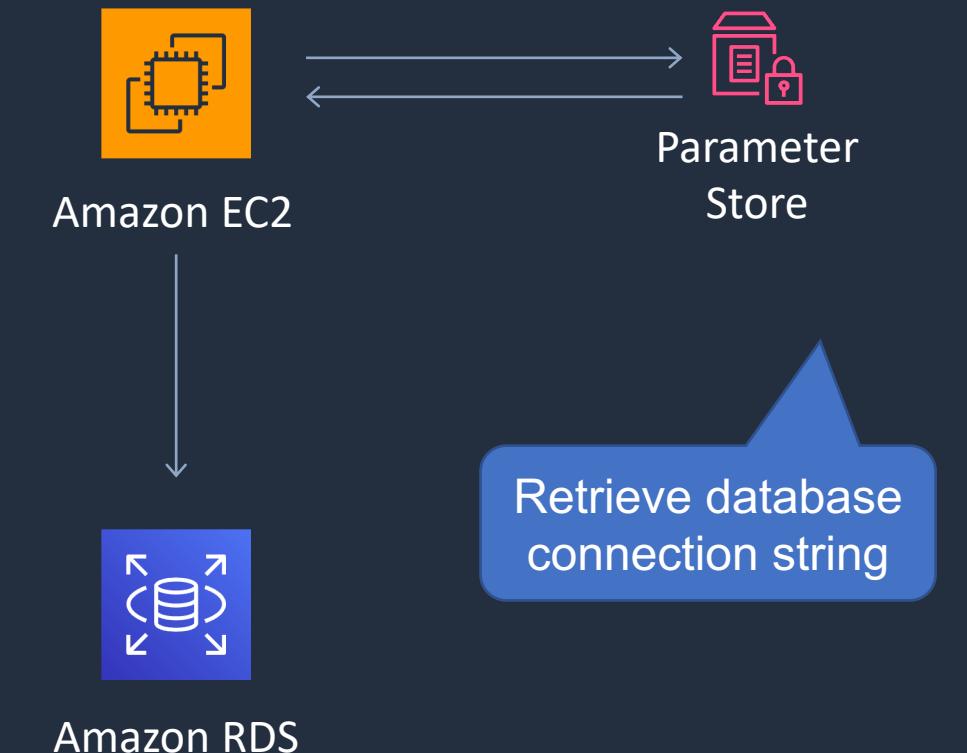


Amazon EC2  
(Windows)



# AWS Systems Manager Parameter Store

- Parameter Store provides secure, hierarchical storage for configuration data management and secrets management
- 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)



# Launch EC2 Managed Instances



# Systems Manager Automation





# AWS Systems Manager Automation

**Documents** define the actions to perform (YAML or JSON)

Automates **IT operations** and **management** tasks across AWS resources



Documents



Automation



Amazon RDS

```
description: Creates an RDS Snapshot for an RDS instance.
schemaVersion: '0.3'
assumeRole: "{{AutomationAssumeRole}}"
parameters:
  DBInstanceIdentifier:
    type: String
    description: (Required) The DBInstanceId ID of the RDS Instance.
  DBSnapshotIdentifier:
    type: String
    description: (Optional) The DBSnapshotIdentifier ID.
    default: ''
  InstanceTags:
    type: String
    default: ''
    description: (Optional) Tags to create for instance.
  SnapshotTags:
    type: String
    default: ''
    description: (Optional) Tags to create for snapshot
```

This automation, takes a snapshot of an RDS DB instance

# Systems Manager Run Command and Patch Manager





# AWS Systems Manager Run Command

Document types include  
**command, automation, package** etc.

Runs commands on  
managed EC2 instances



```
{  
  "schemaVersion": "1.2",  
  "description": "List missing Microsoft Windows updates.",  
  "parameters": {  
    "UpdateLevel": {  
      "type": "String",  
      "default": "Important",  
      "description": "Important: .....",  
      "allowedValues": [  
        "None",  
        "All",  
        "Important",  
        "Optional"  
      ]  
    }  
  }  
}
```

This command checks  
for missing updates

# Systems Manager Configuration Compliance



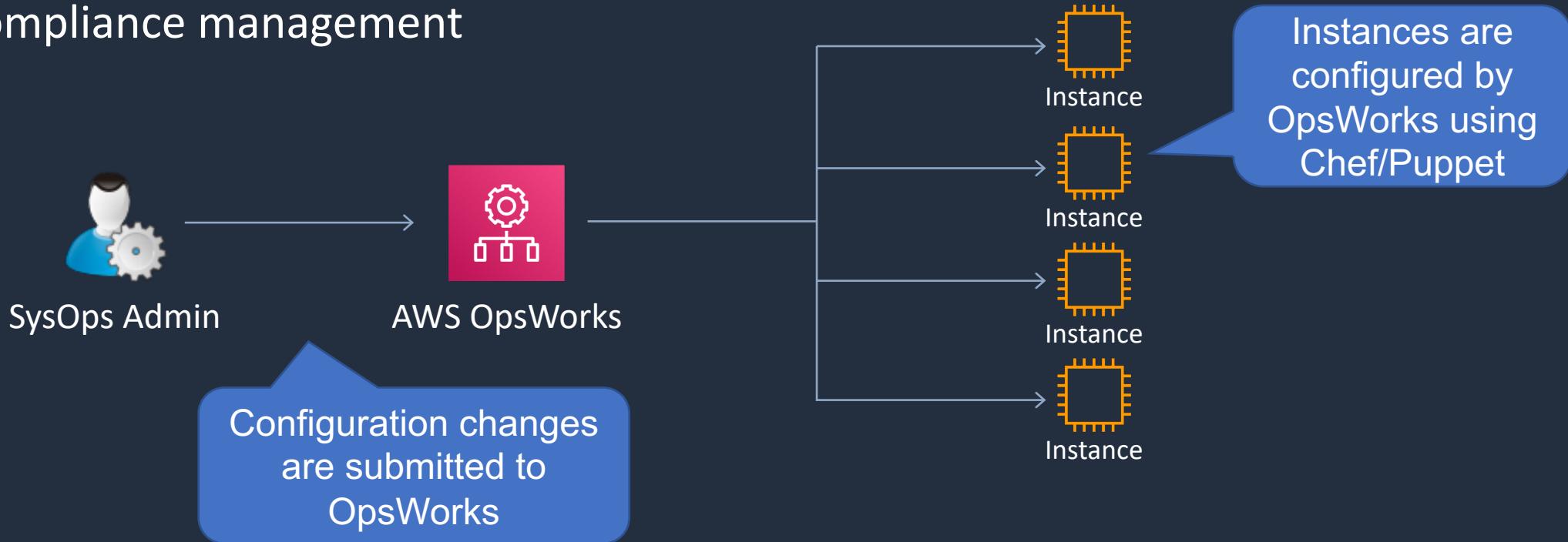
# AWS OpsWorks





# 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)





# AWS RAM

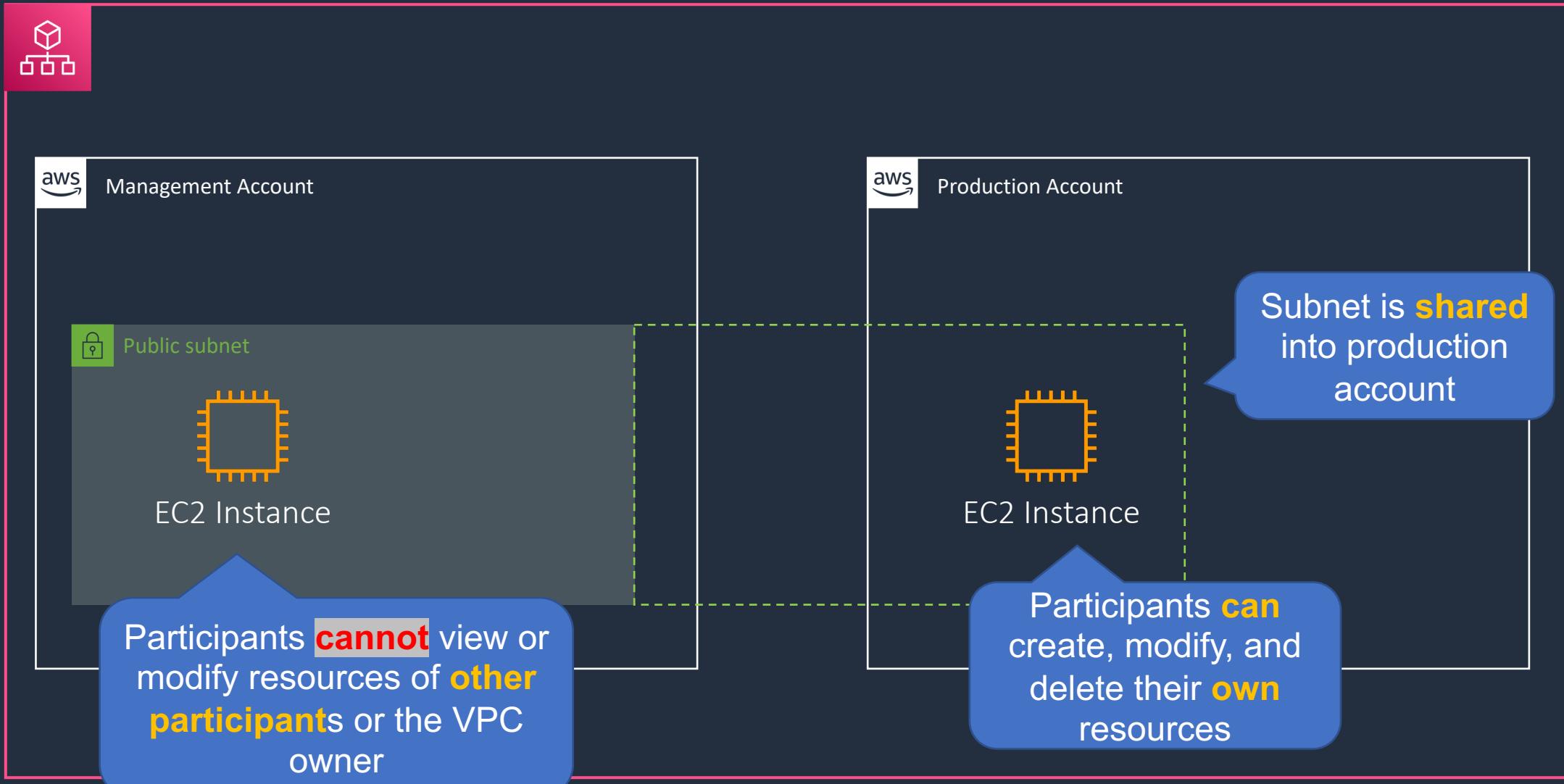
---

---

- Shares resources with accounts or organizations
- RAM can be used to share:
  - AWS Transit Gateways
  - Subnets
  - AWS License Manager configurations
  - Amazon Route 53 Resolver rules



# AWS RAM



# Share a Subnet Across Accounts



# Architecture Patterns – Deployment and Management





# Architecture Patterns – Deployment and Management

## Requirement

Global company needs to centrally manage creation of infrastructure services to accounts in AWS Organizations

Company is concerned about malicious attacks on RDP and SSH ports for remote access to EC2

Development team require method of deploying applications using CloudFormation. Developers typically use JavaScript and TypeScript

## Solution

Define infrastructure in CloudFormation templates, create Service Catalog products and portfolios in central account and share using Organizations

Deploy Systems Manager agent and use Session Manager

Define resources in JavaScript and TypeScript and use the AWS Cloud Development Kit (CDK) to create CloudFormation templates



# Architecture Patterns – Deployment and Management

## Requirement

Need to automate the process of updating an application when code is updated

## Solution

Create a CodePipeline that sources code from CodeCommit and use CodeBuild and CodeDeploy

Need to safely deploy updates to EC2 through a CodePipeline. Resources defined in CF templates and app code stored in S3

Use CodeBuild to automate testing, use CF changes sets to evaluate changes and CodeDeploy to deploy using a blue/green deployment pattern

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

# SECTION 14

## 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



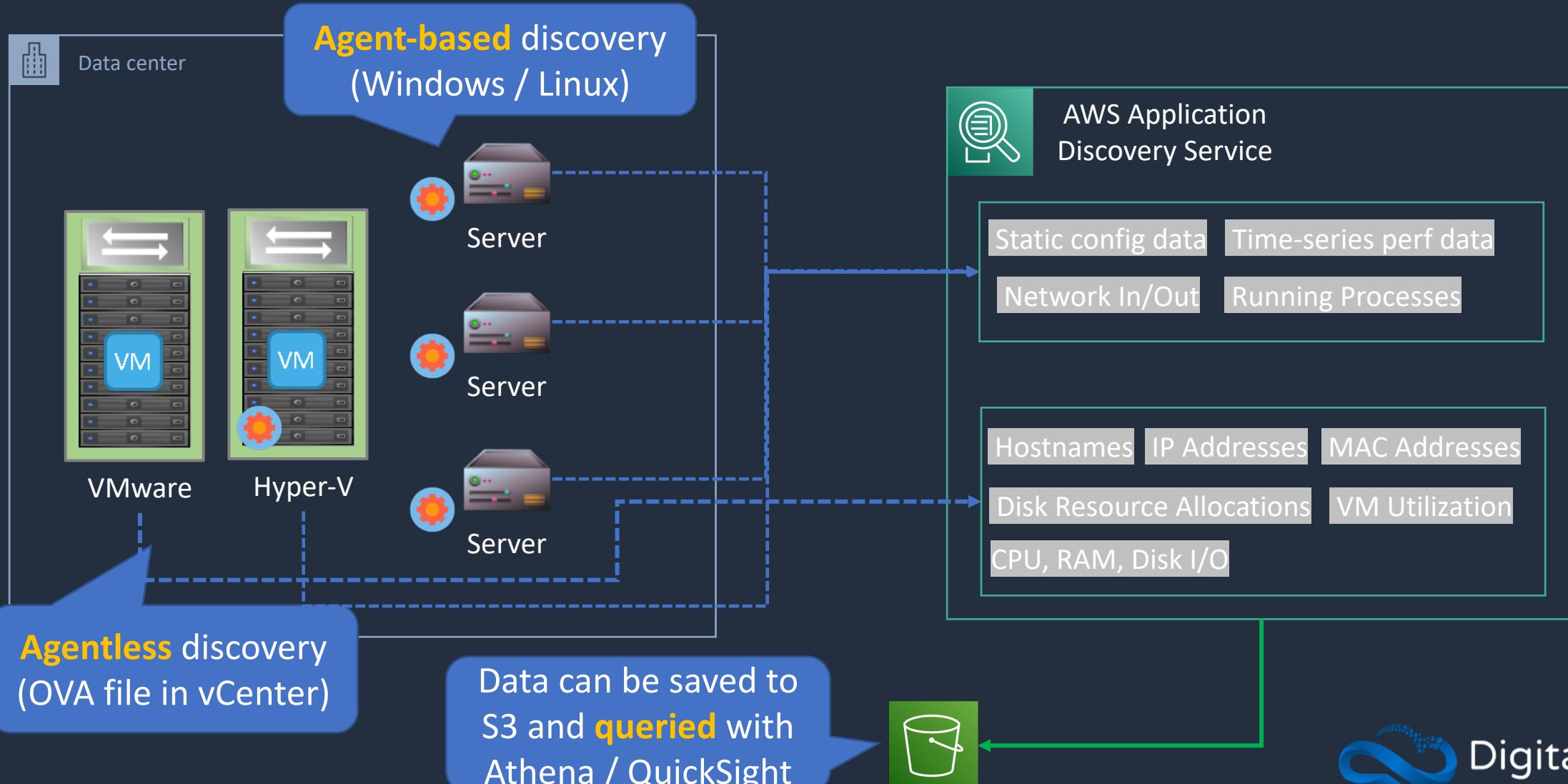
NAS / File  
Server

# AWS Application Discovery Service





# AWS Application Discovery Service

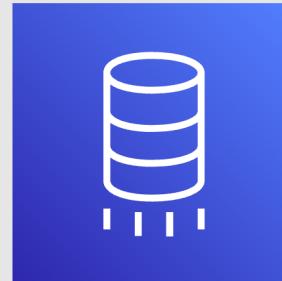




# AWS Application Discovery Service

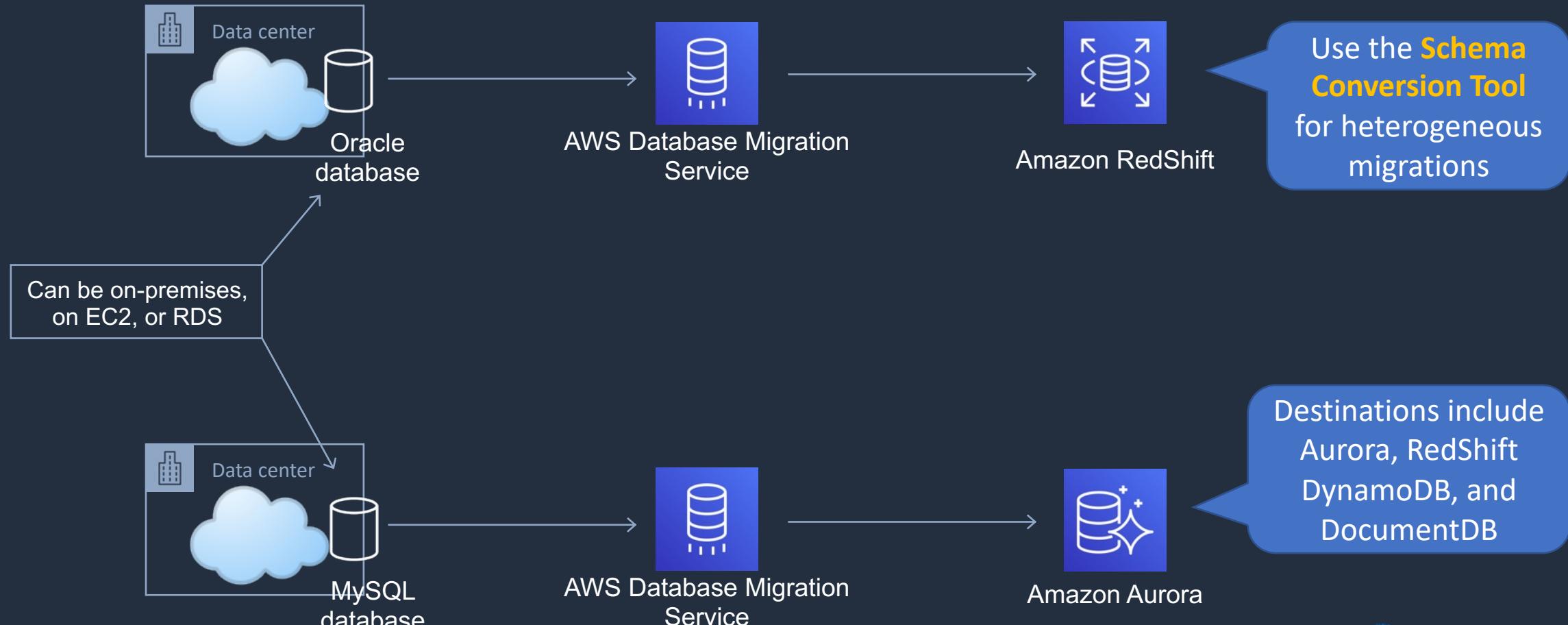
	Discovery Connector	Discovery Agent
<b>Supported Servers</b>	VMware	VMware, Physical
<b>Deployment</b>	Per vCenter	Per Server
<b>Collected data</b>	Static configuration data  VM utilization metrics	Static configuration data  Time Series Performance info (export)  Network Inbound/outbound (export)  Running 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

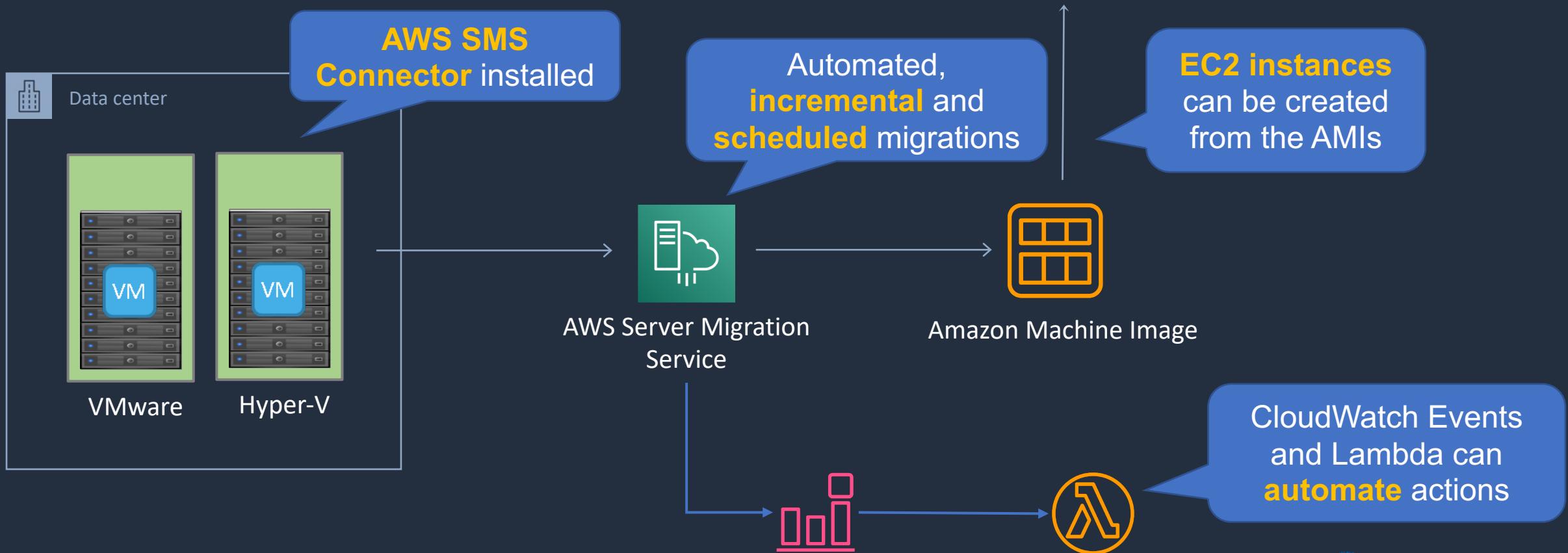
# AWS Server Migration Service (SMS)





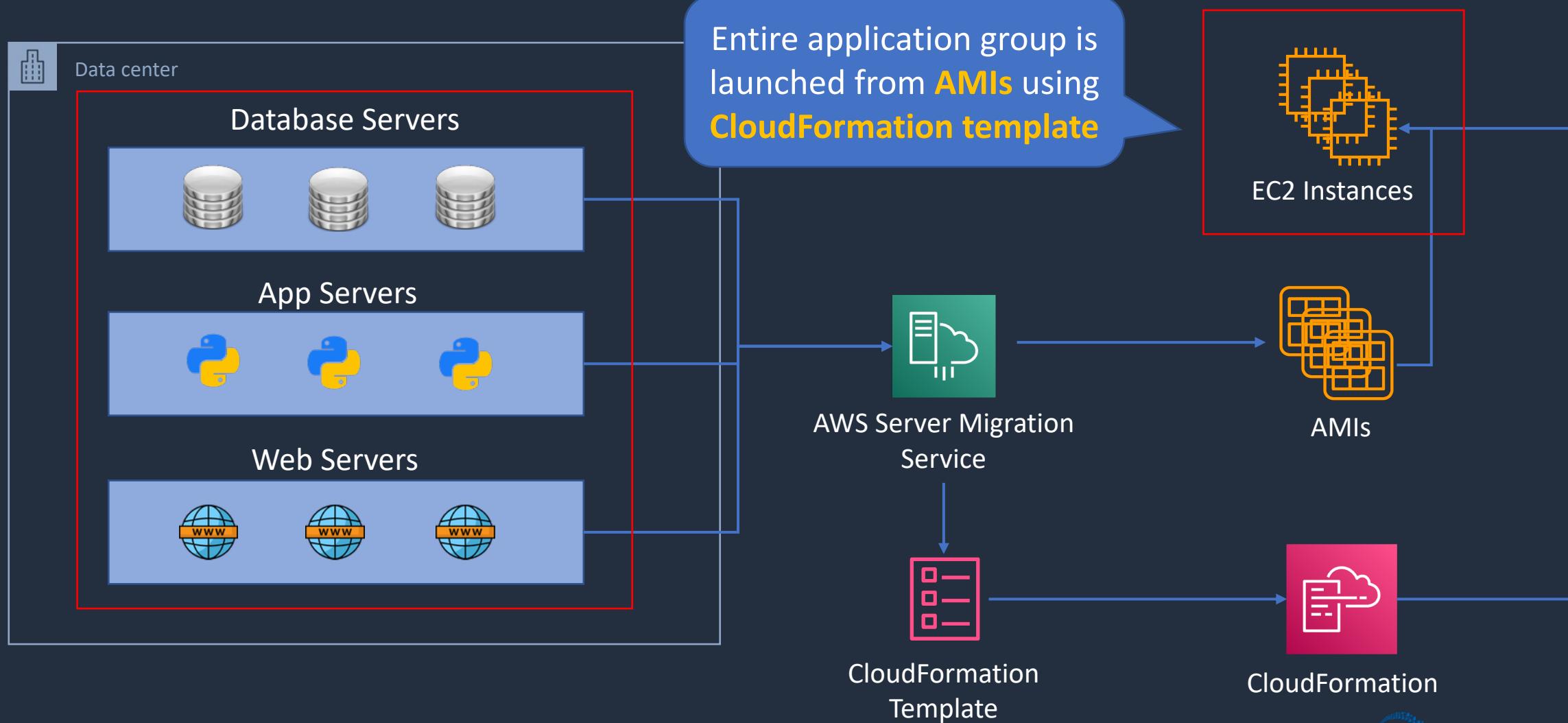
# AWS Server Migration Service (SMS)

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

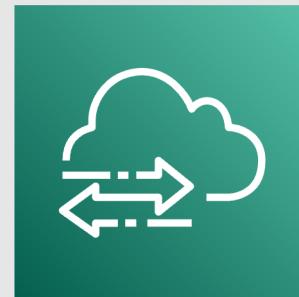




# AWS SMS – Application Migration

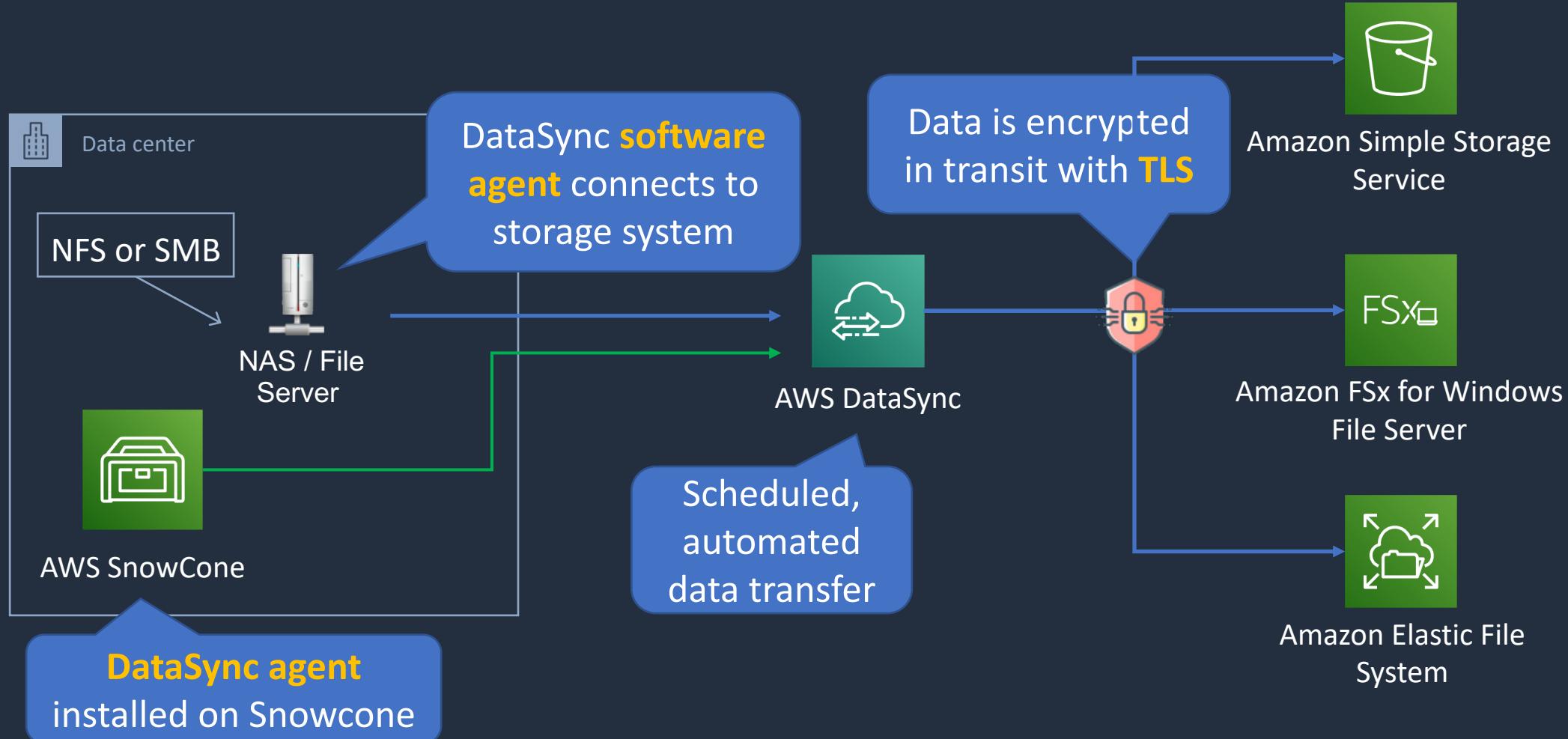


# AWS DataSync





# 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

## Solution

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

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

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

Need to migrate an Oracle data warehouse to AWS

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

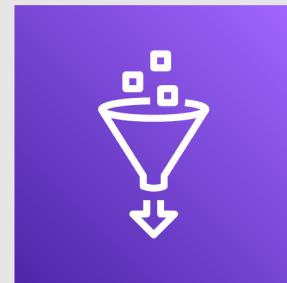
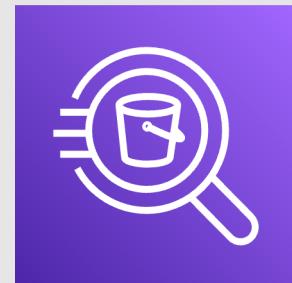
Need to migrate 50TB of data and company only has a 1Gbps internet link. Requirement is urgent

Use AWS Snowball to transfer the data

# SECTION 15

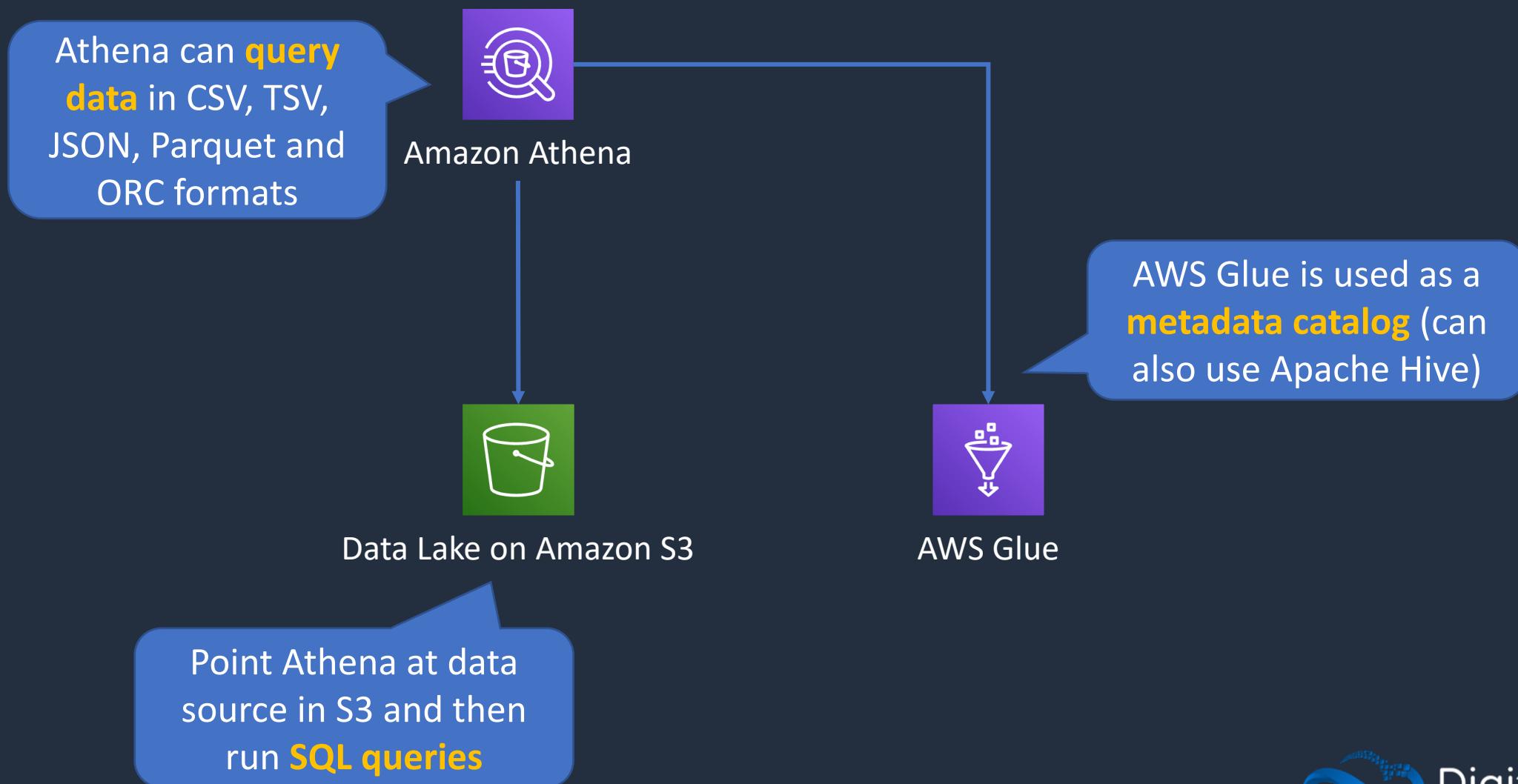
## Analytics Services

# Amazon Athena and AWS Glue



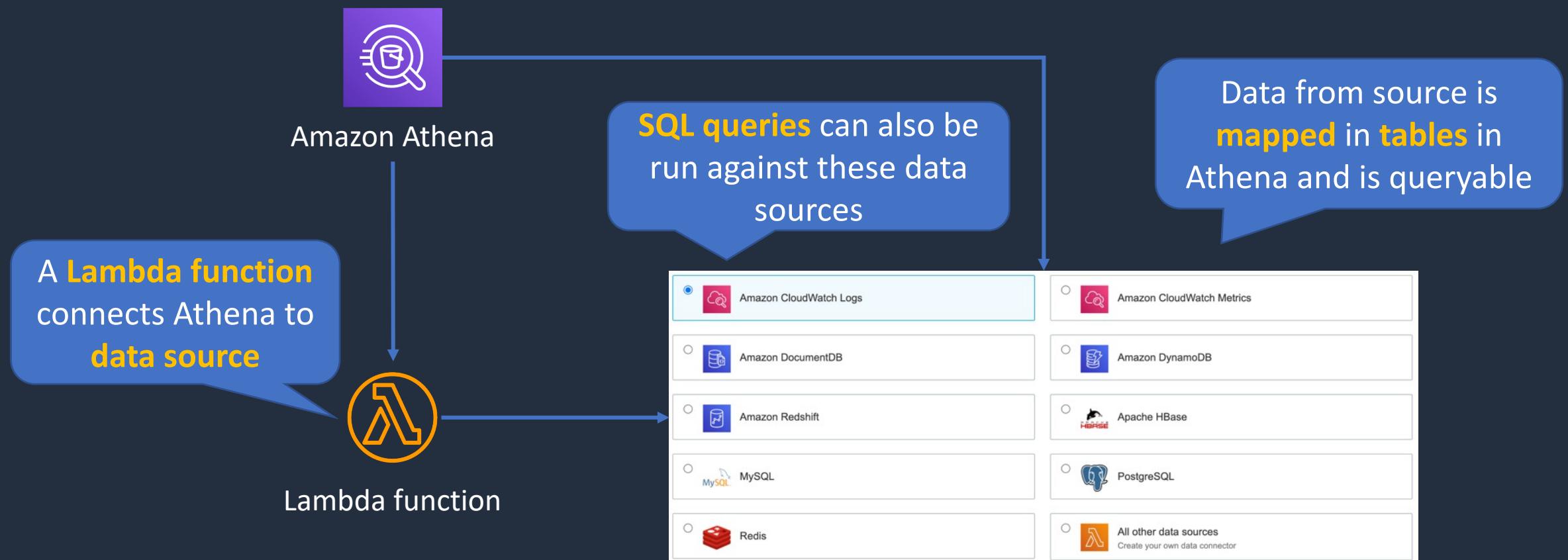


# Amazon Athena and AWS Glue





# 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

---

---

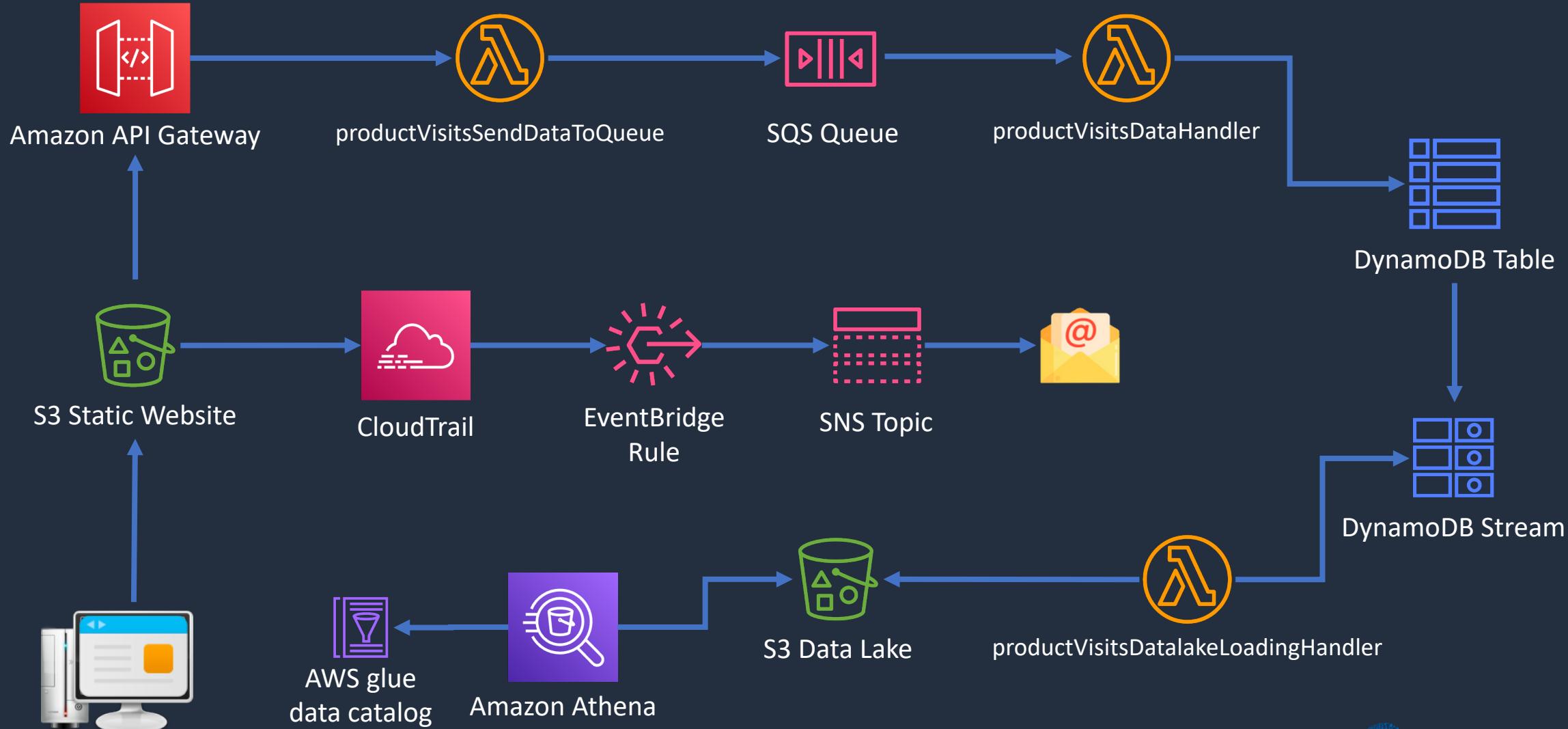
- You can use a **crawler** to populate the AWS Glue Data Catalog with tables
- A crawler can crawl multiple data stores in a single run
- Upon completion, the crawler creates or updates one or more tables in your Data Catalog.
- ETL jobs that you define in AWS Glue use the Data Catalog tables as sources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

# Build a Serverless App – Part 5





# Serverless App Architecture



# Redshift and OLAP Use Cases





# OLTP vs OLAP

---

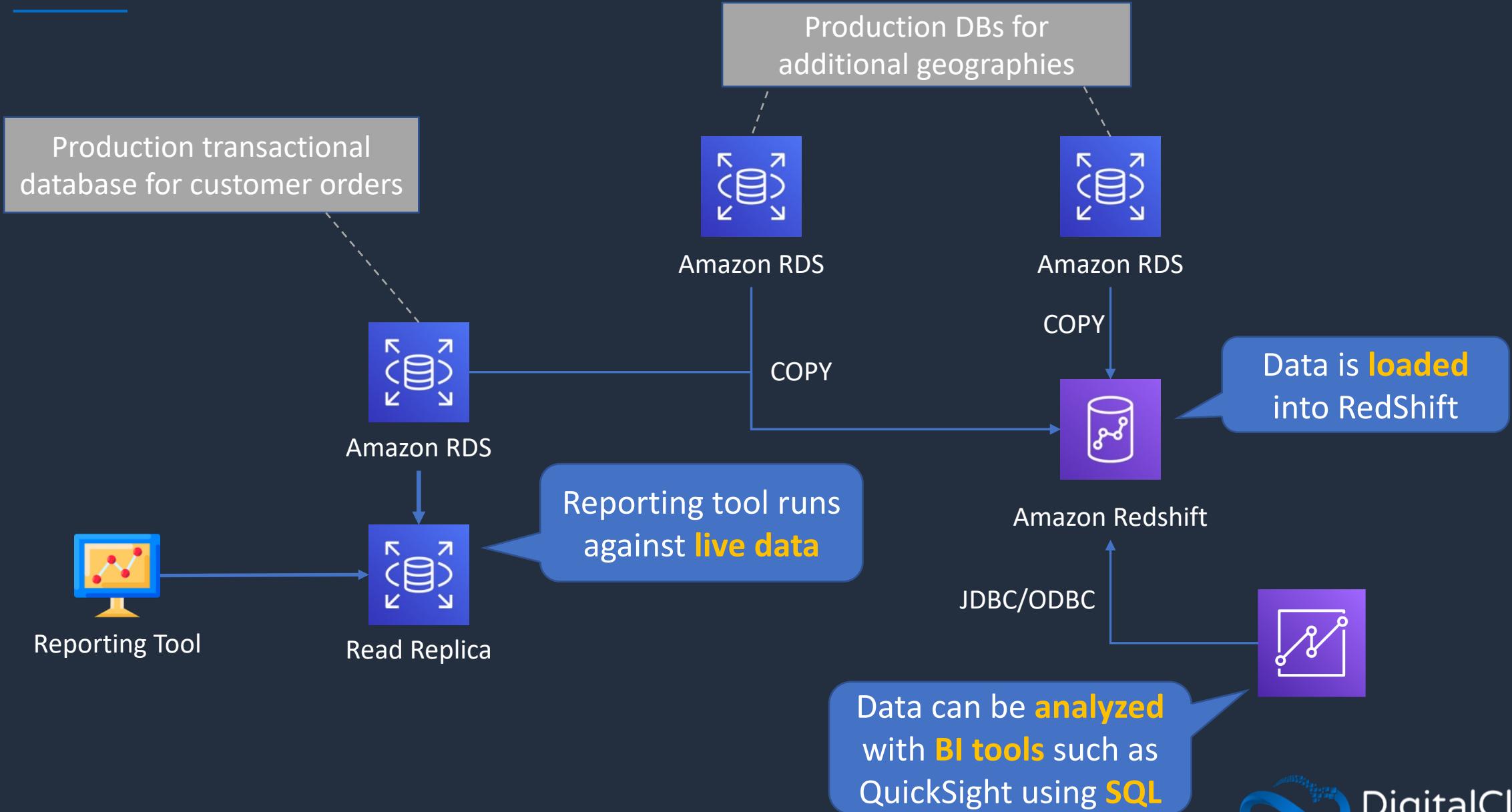
---

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

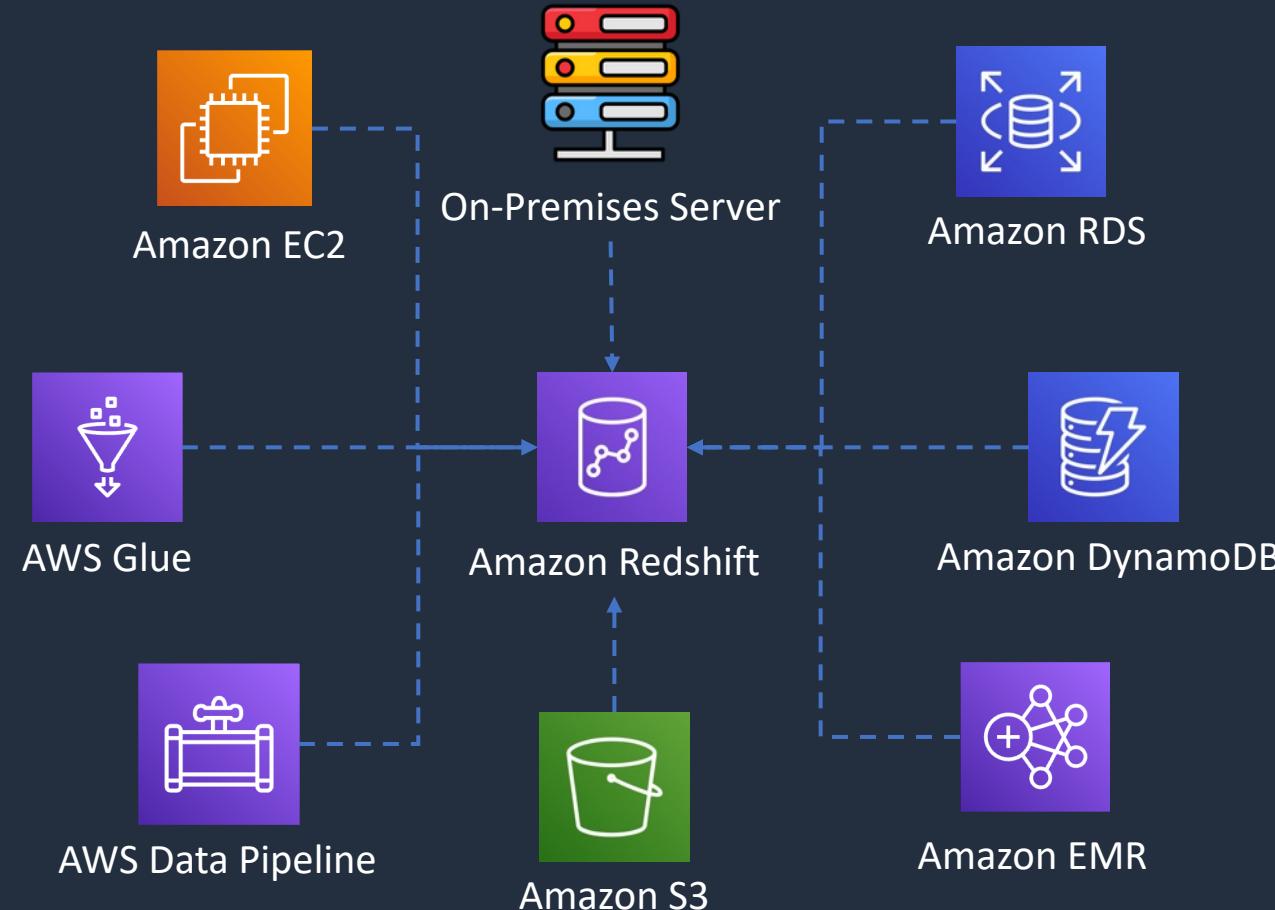


# Redshift and OLAP Use Cases





# 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 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 Integrations

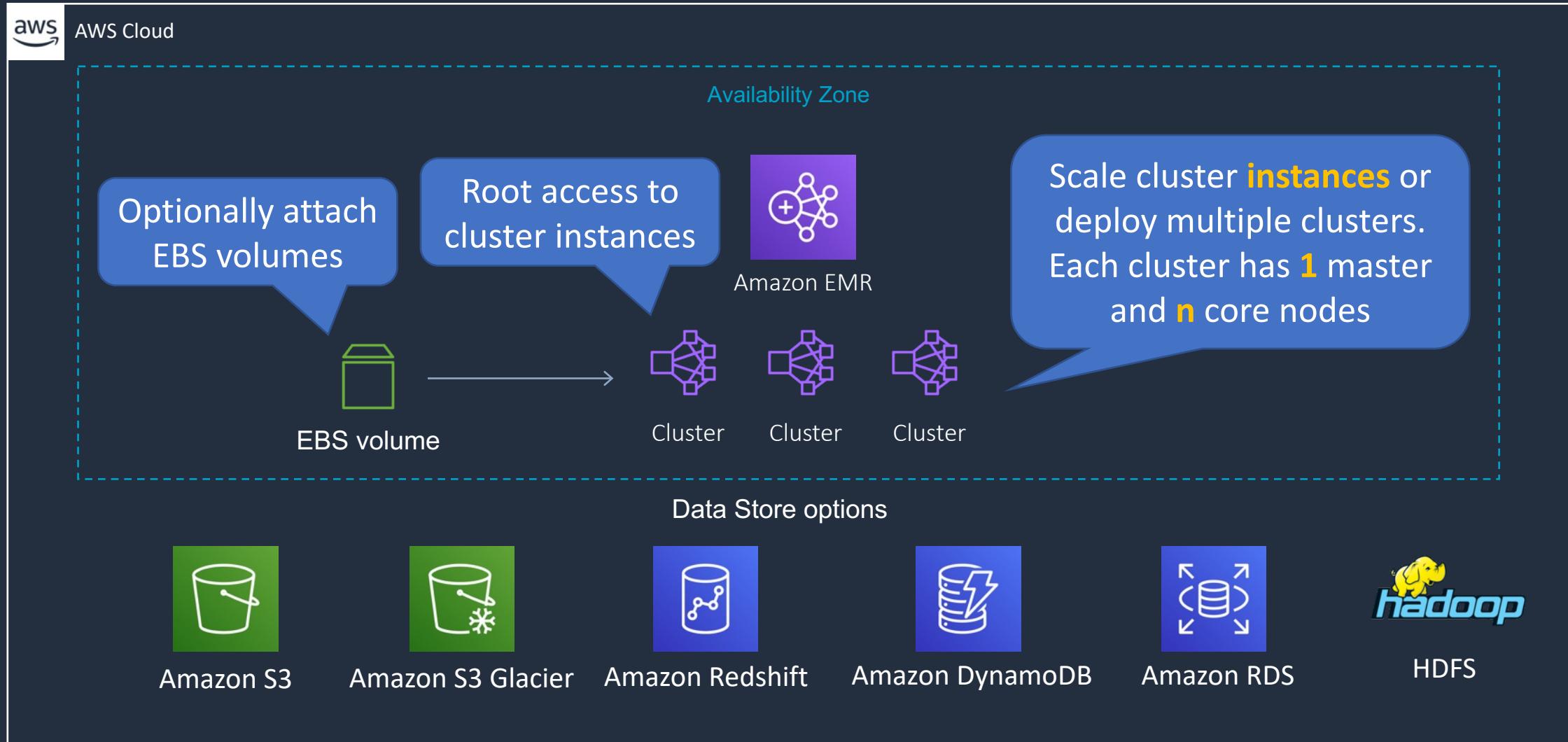
---

---

- **Amazon EC2** – the instances that comprise the nodes in the cluster
- **Amazon VPC** – configure the virtual network in which you launch your instances
- **Amazon S3** – store input and output data
- **Amazon CloudWatch** – monitor cluster performance and configure alarms
- **AWS IAM** – configure permissions
- **AWS CloudTrail** – audit requests made to the service
- **AWS Data Pipeline** – schedule and start your clusters
- **AWS Lake Formation** – discover, catalog, and secure data in an Amazon S3 data lake

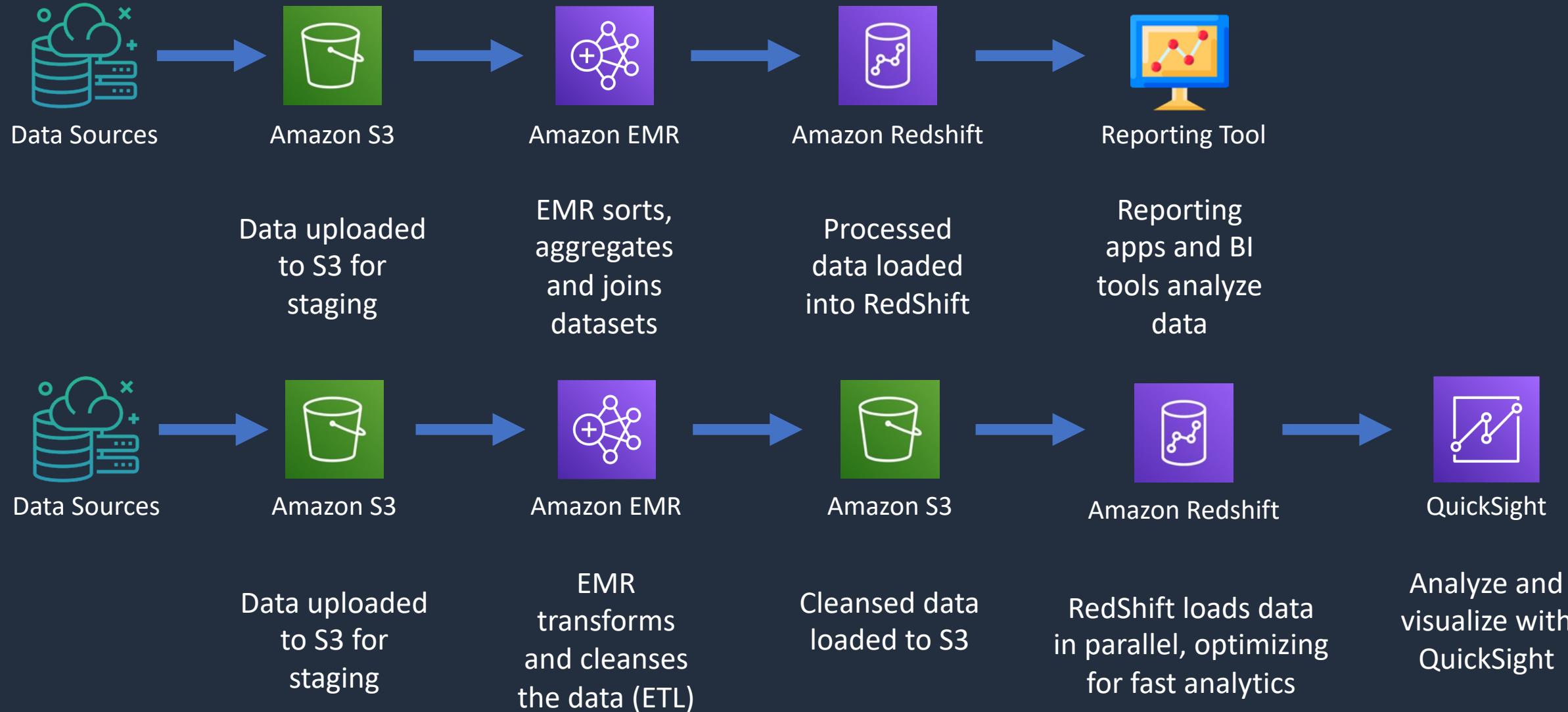


# Amazon EMR Architecture





# Amazon EMR Use Cases

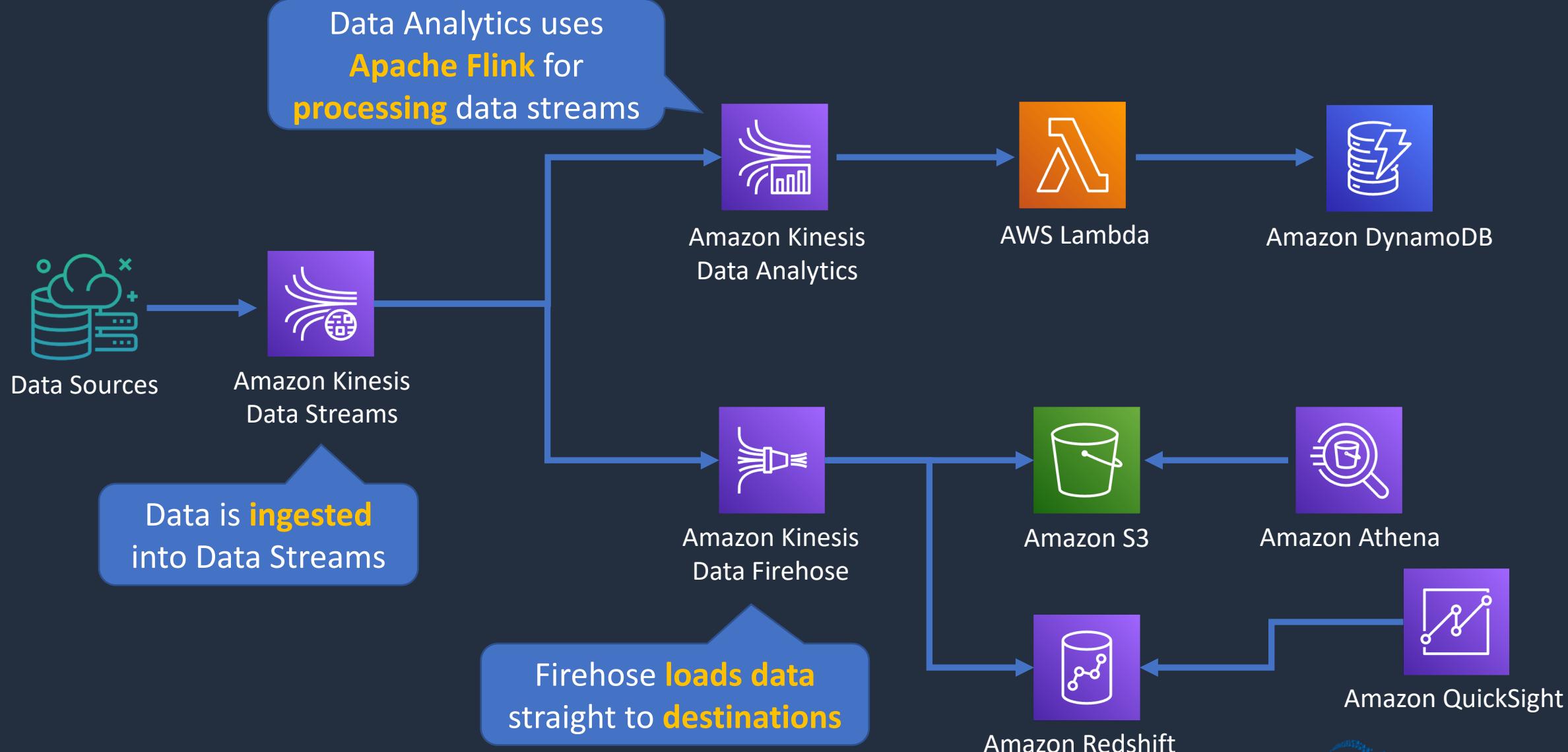


# Amazon Kinesis Core Knowledge





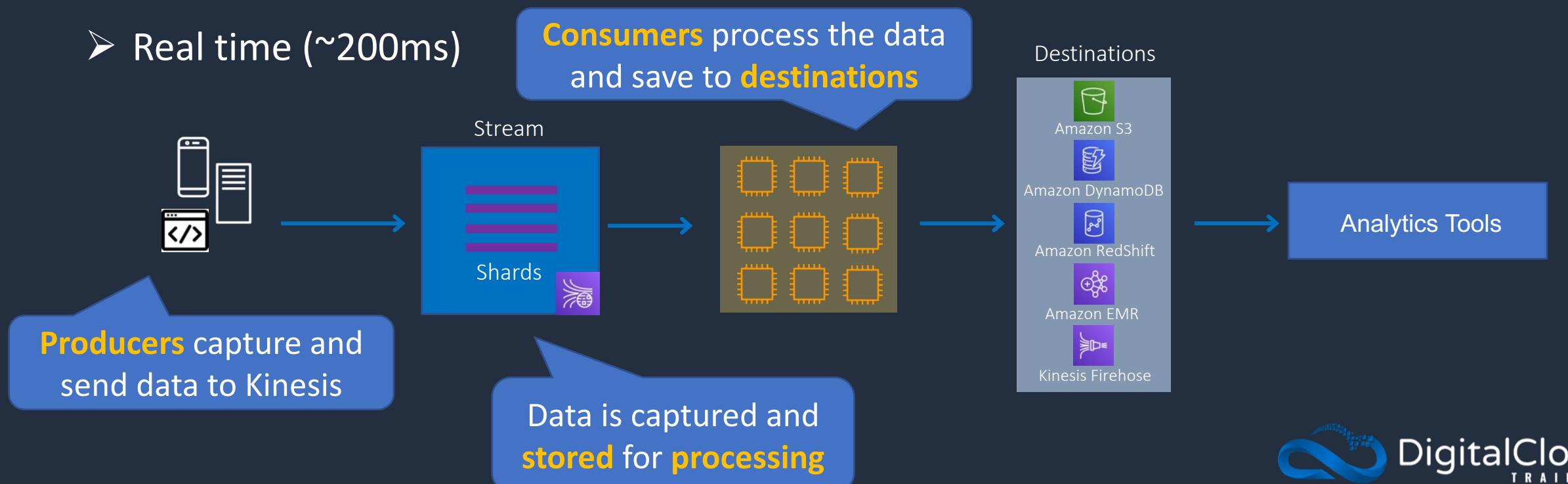
# Amazon Kinesis Services





# Amazon Kinesis Data Streams

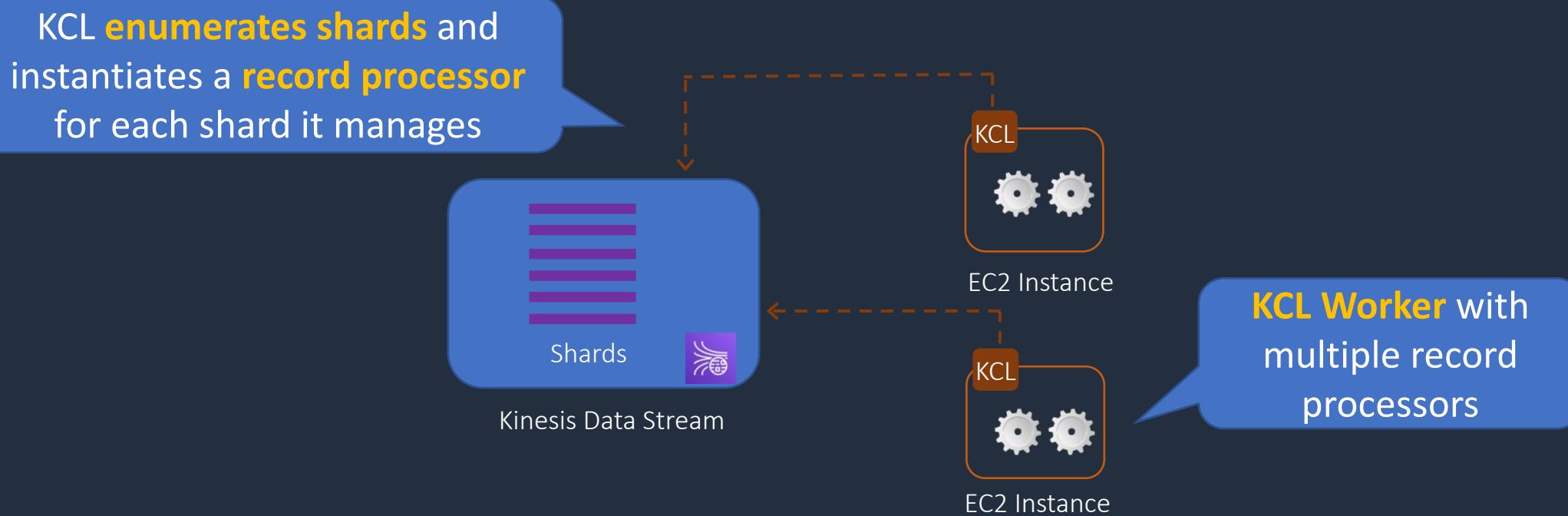
- Producers send data to Kinesis, data is stored in Shards for 24 hours (by default, up to 7 days)
- Consumers then take the data and process it - data can then be saved into another AWS service
- Real time (~200ms)





# Kinesis Client Library (KCL)

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



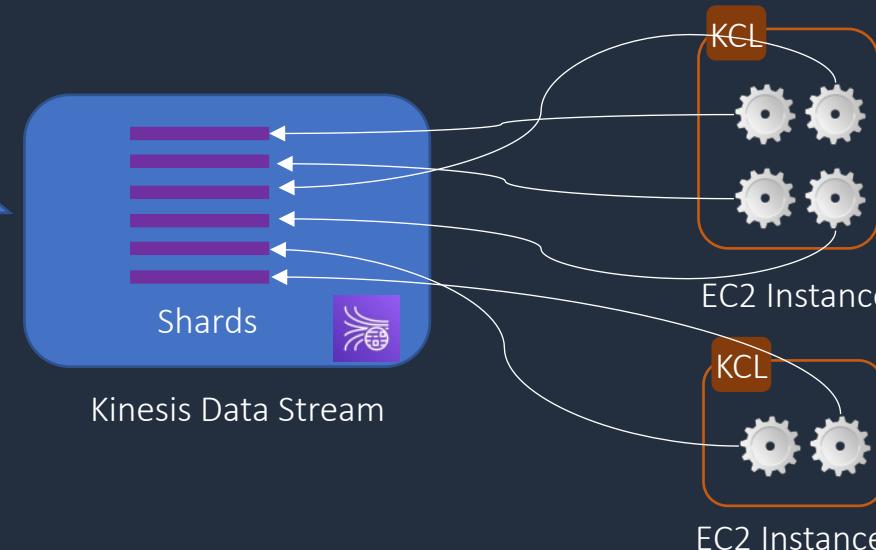


# Kinesis Client Library (KCL)

---

- 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

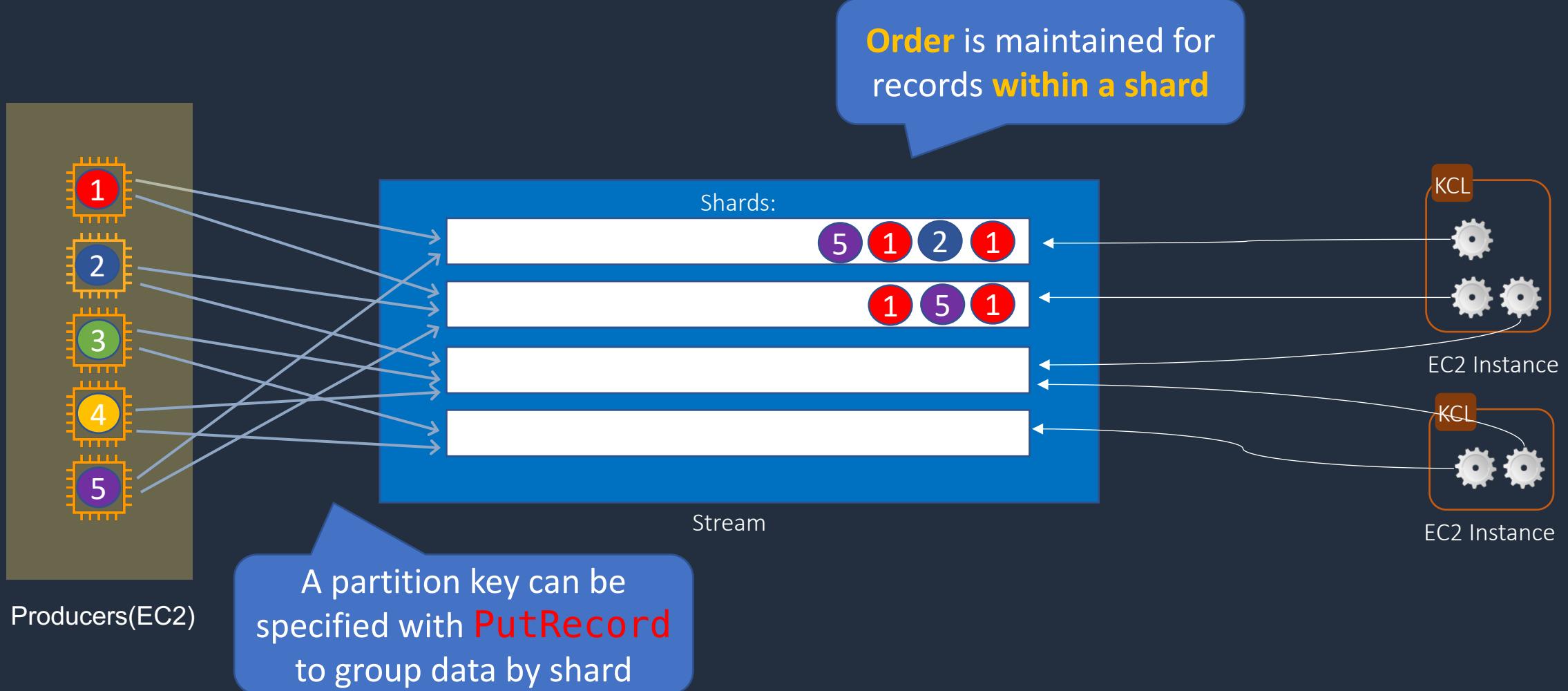
Each shard is **processed** by exactly **one** KCL worker



A record processor maps to exactly one shard



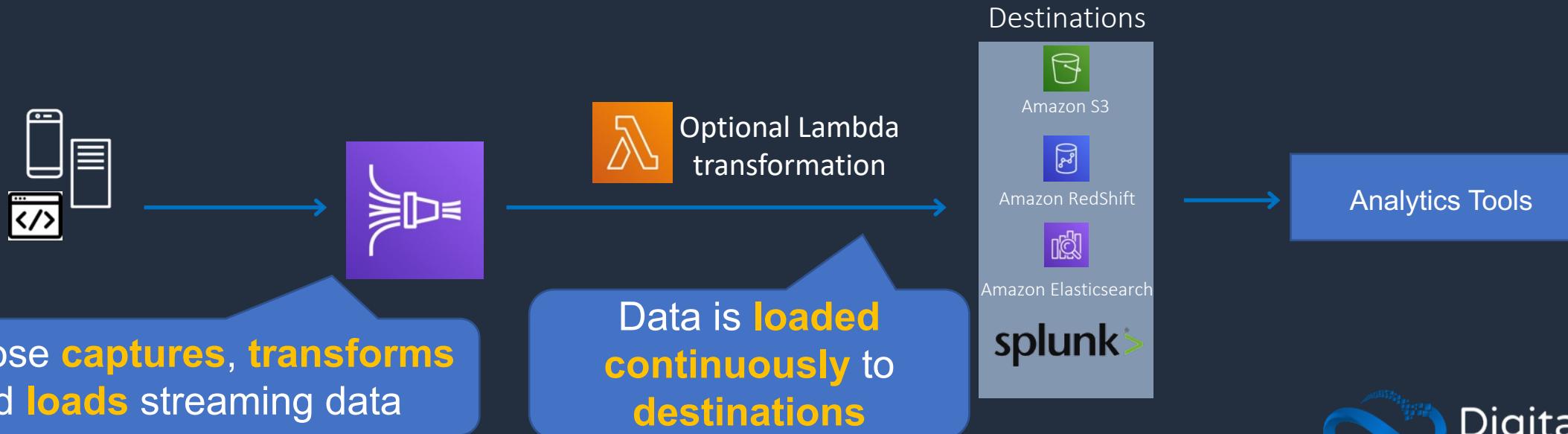
# Amazon Kinesis Data Streams





# 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:

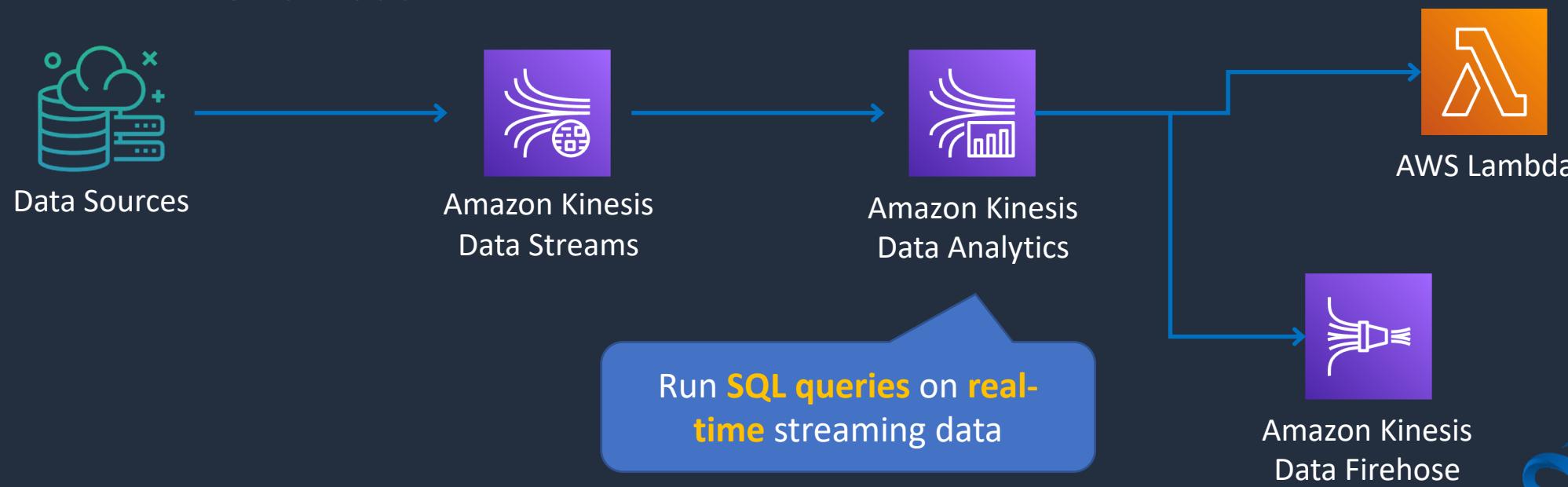
- RedShift (via an intermediate S3 bucket)
- Elasticsearch
- Amazon S3
- Splunk
- Datadog
- MongoDB
- New Relic
- HTTP Endpoint



# 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



# Architecture Patterns - Analytics





# Architecture Patterns – Analytics

## Requirement

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

## Solution

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

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

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

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

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



# Architecture Patterns – Analytics

## Requirement

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

## Solution

Send logs to Kinesis Data Firehose and configure the auditing application using a 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

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

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

# SECTION 16

## Monitoring, Logging, and Auditing

# Amazon CloudWatch Features and Use Cases





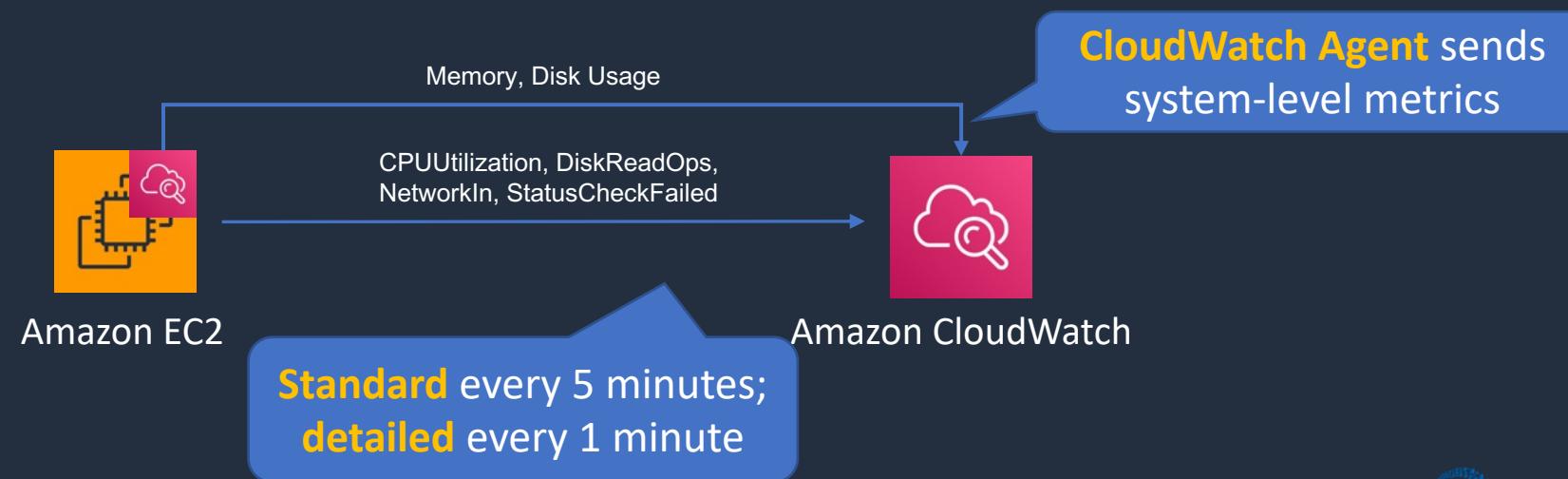
# Amazon CloudWatch

- **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

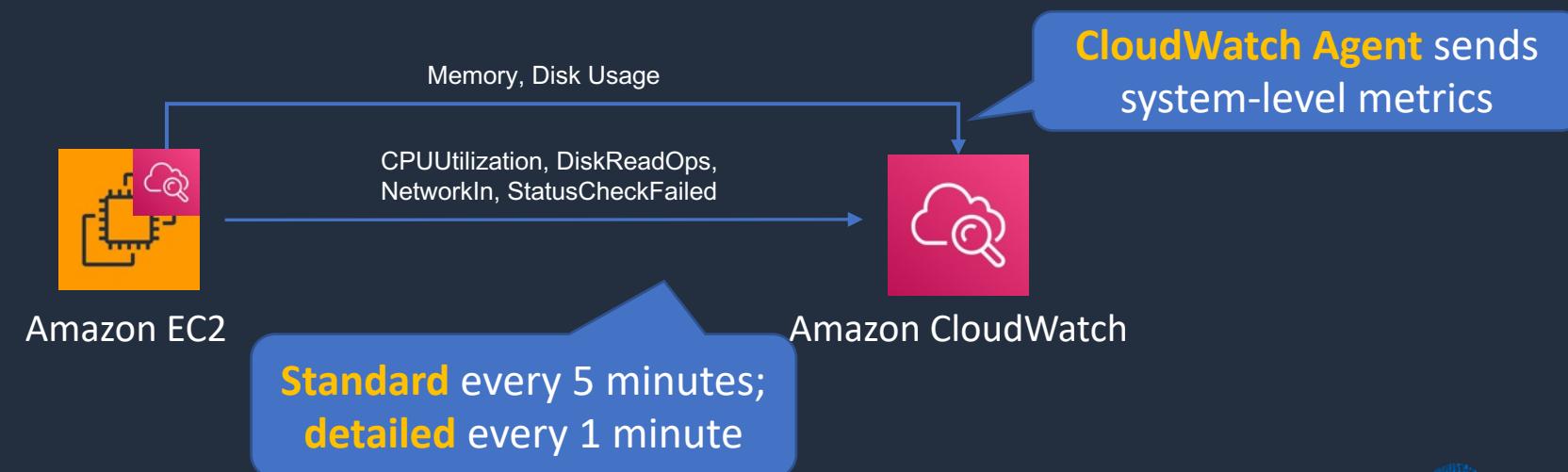
- Metrics are sent to CloudWatch for many AWS services
- EC2 metrics are sent every **5 minutes** by default (free)
- Detailed EC2 monitoring sends every **1 minute** (chargeable)
- Unified CloudWatch Agent sends system-level metrics for EC2 and on-premises servers
- System-level metrics include memory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

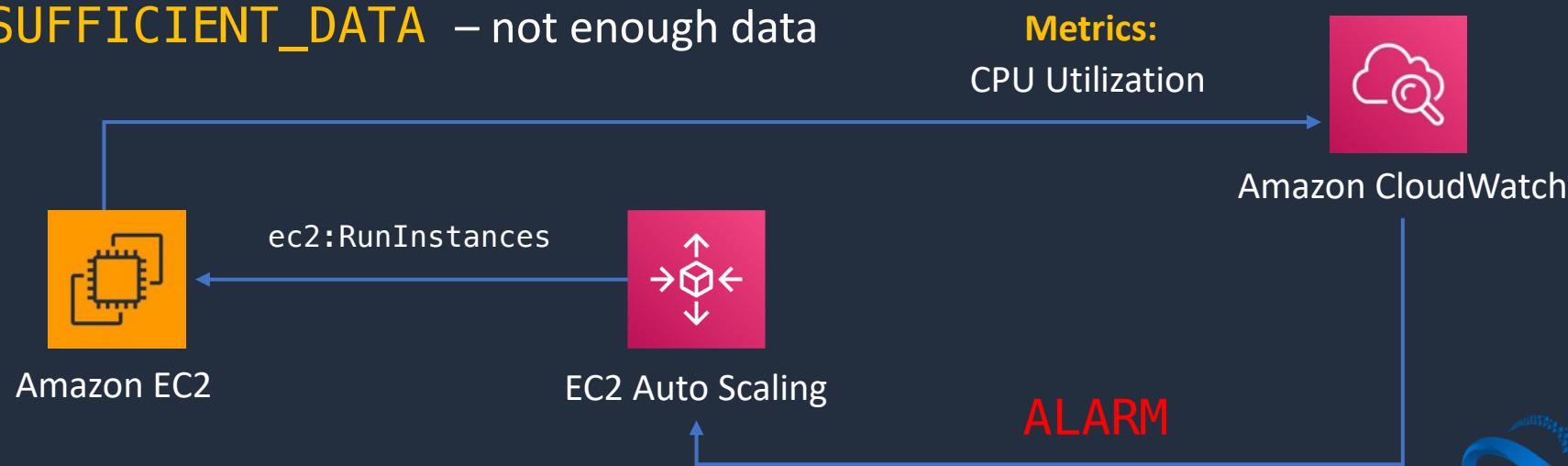




# Amazon CloudWatch Alarms

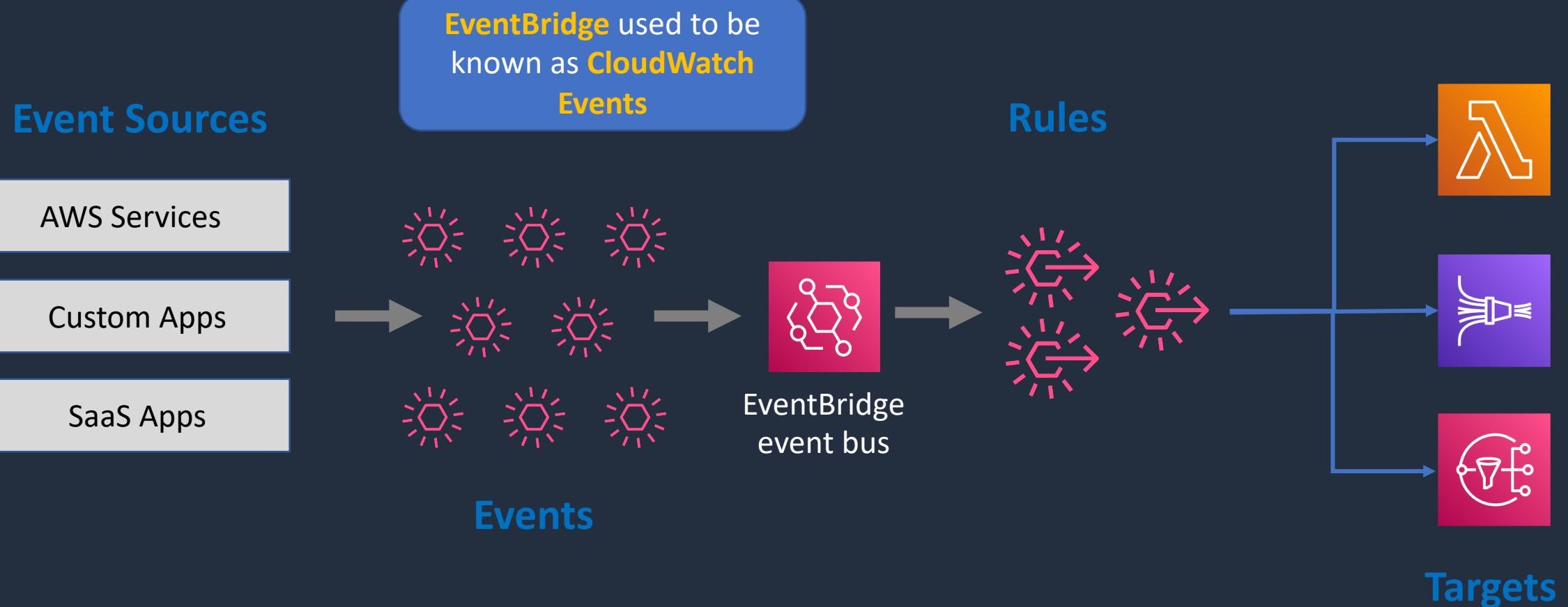
Two types of alarms

- **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





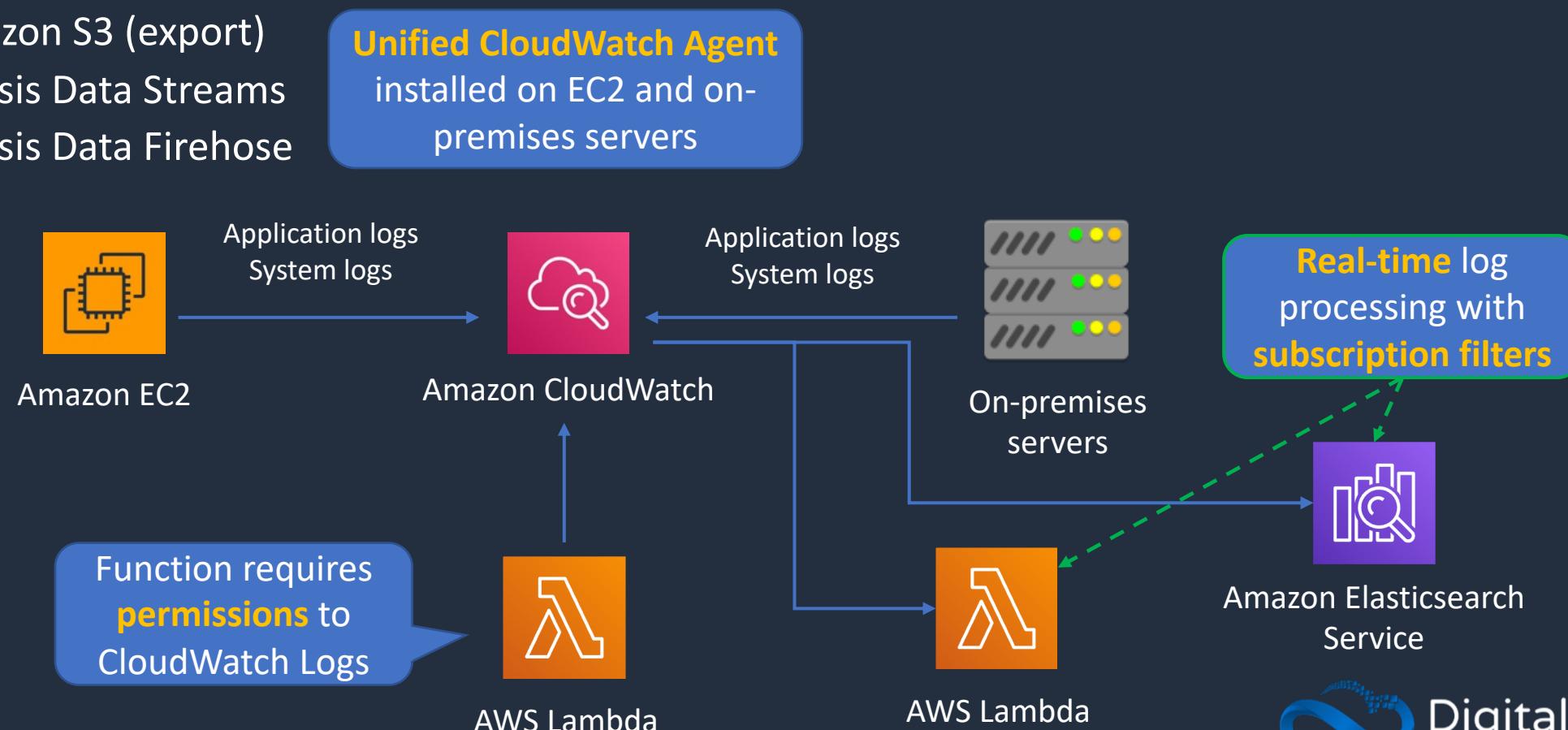
# Amazon CloudWatch Events / EventBridge





# Amazon CloudWatch Logs

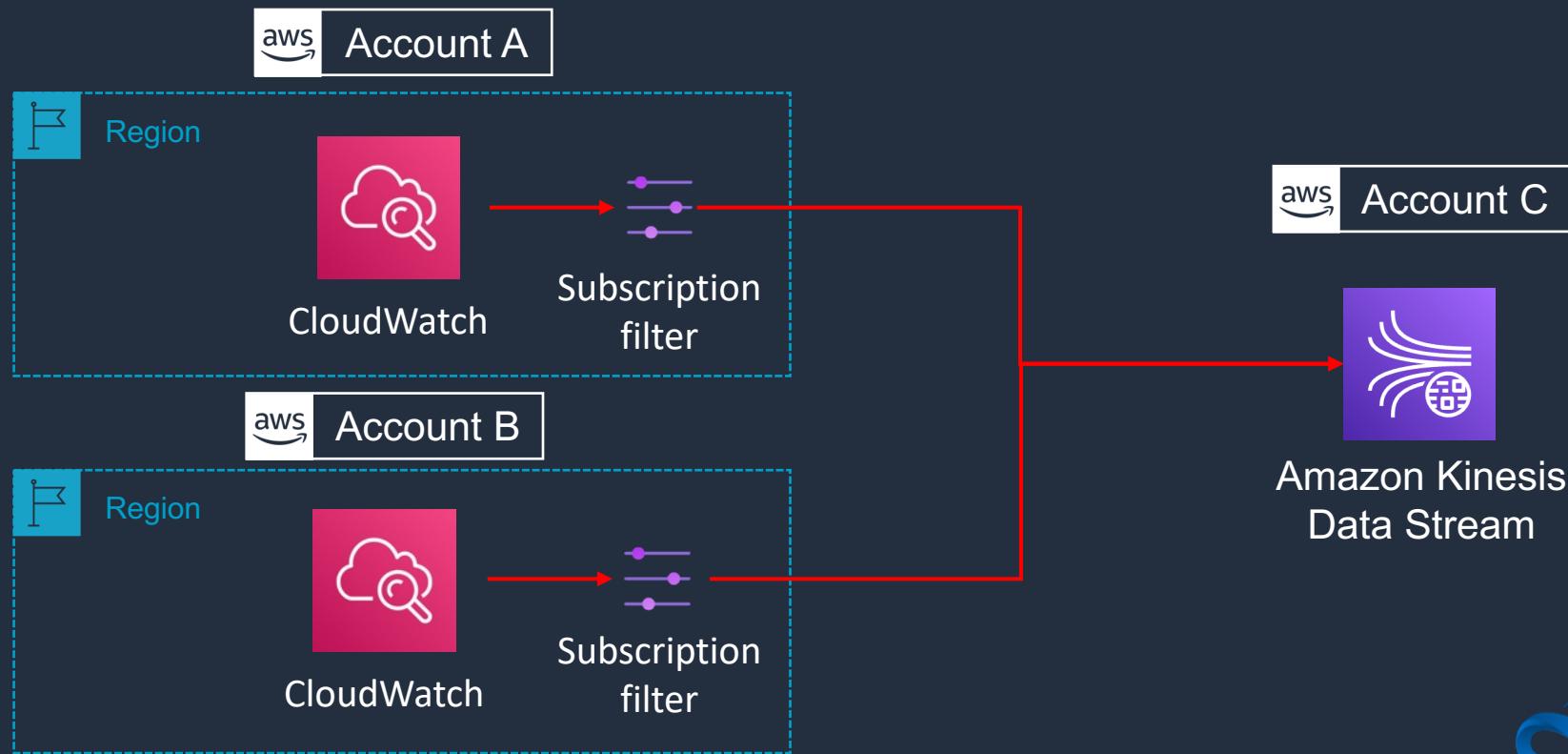
- Gather application and system logs in CloudWatch
- Defined expiration policies and KMS encryption
- Send to:
  - Amazon S3 (export)
  - Kinesis Data Streams
  - Kinesis Data Firehose





# Cross-Account Log Data Sharing

- Share CloudWatch Logs across accounts
- Kinesis Data Streams is the only supported destination
- **Log data sender** – sends log data to the recipient
- **Log data recipient** – sends data to a Kinesis Data stream

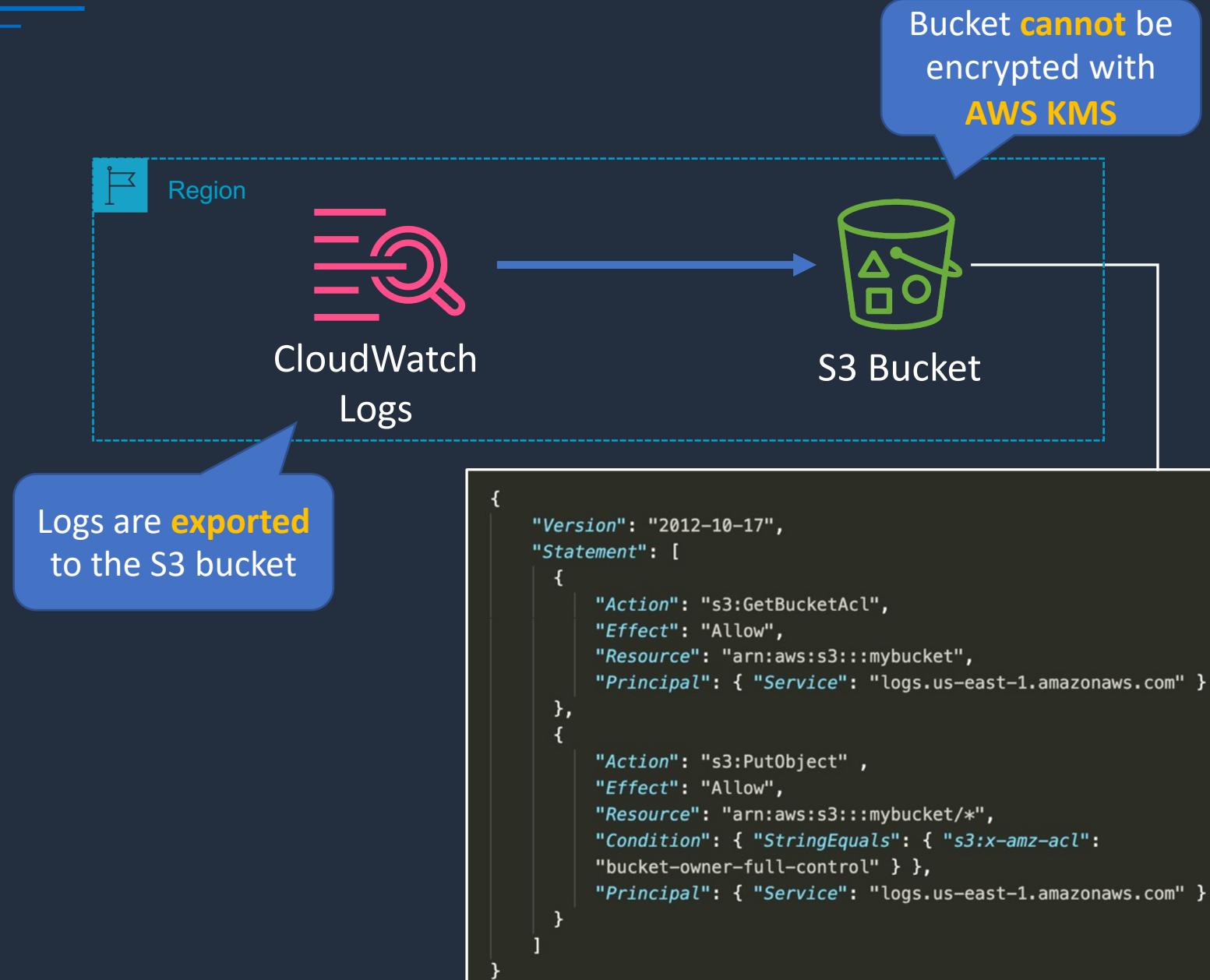


# Export CloudWatch Logs to S3





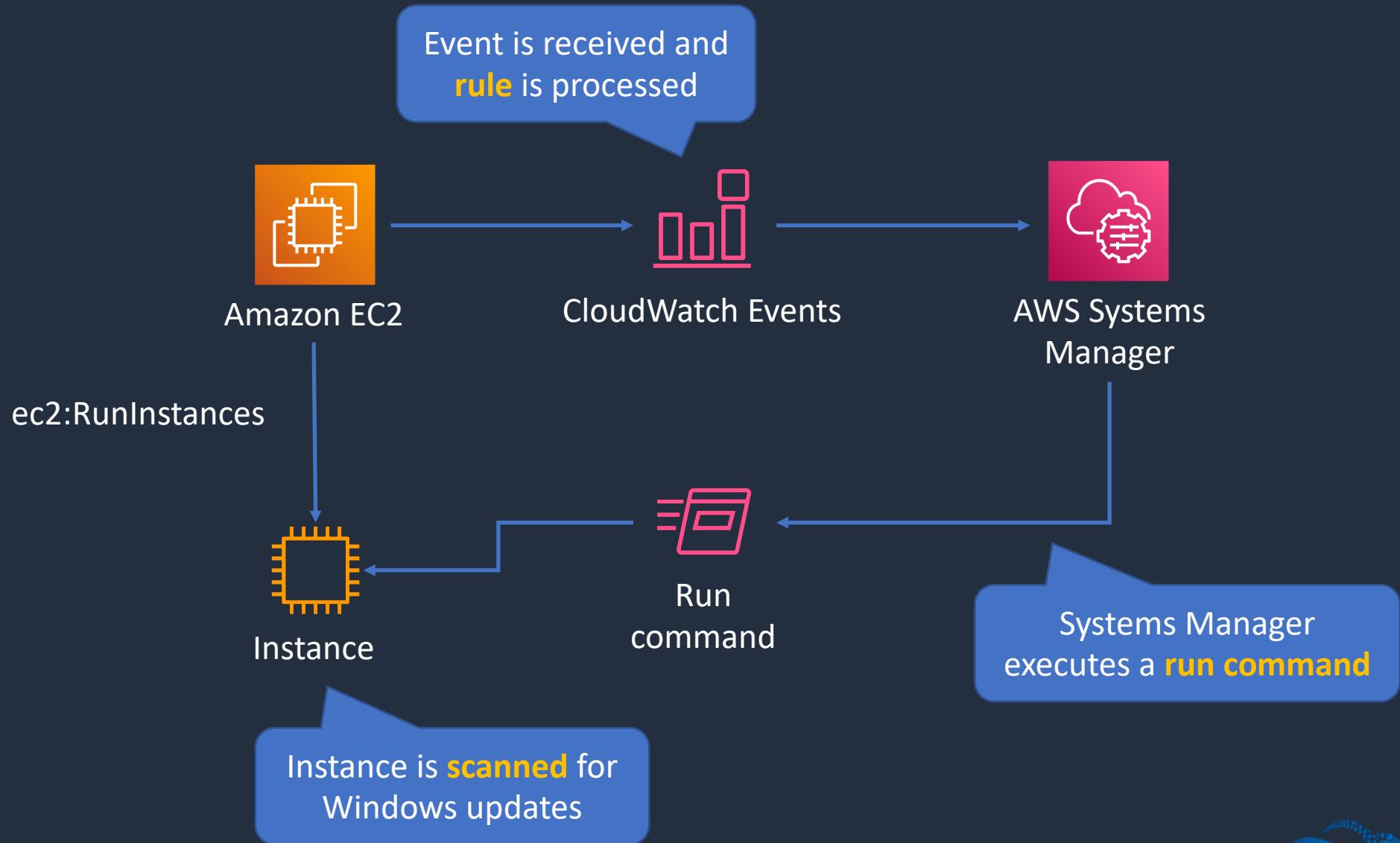
# Export CloudWatch Logs to S3



# Trigger SSM on Instance Launch



# Trigger SSM on Instance Launch

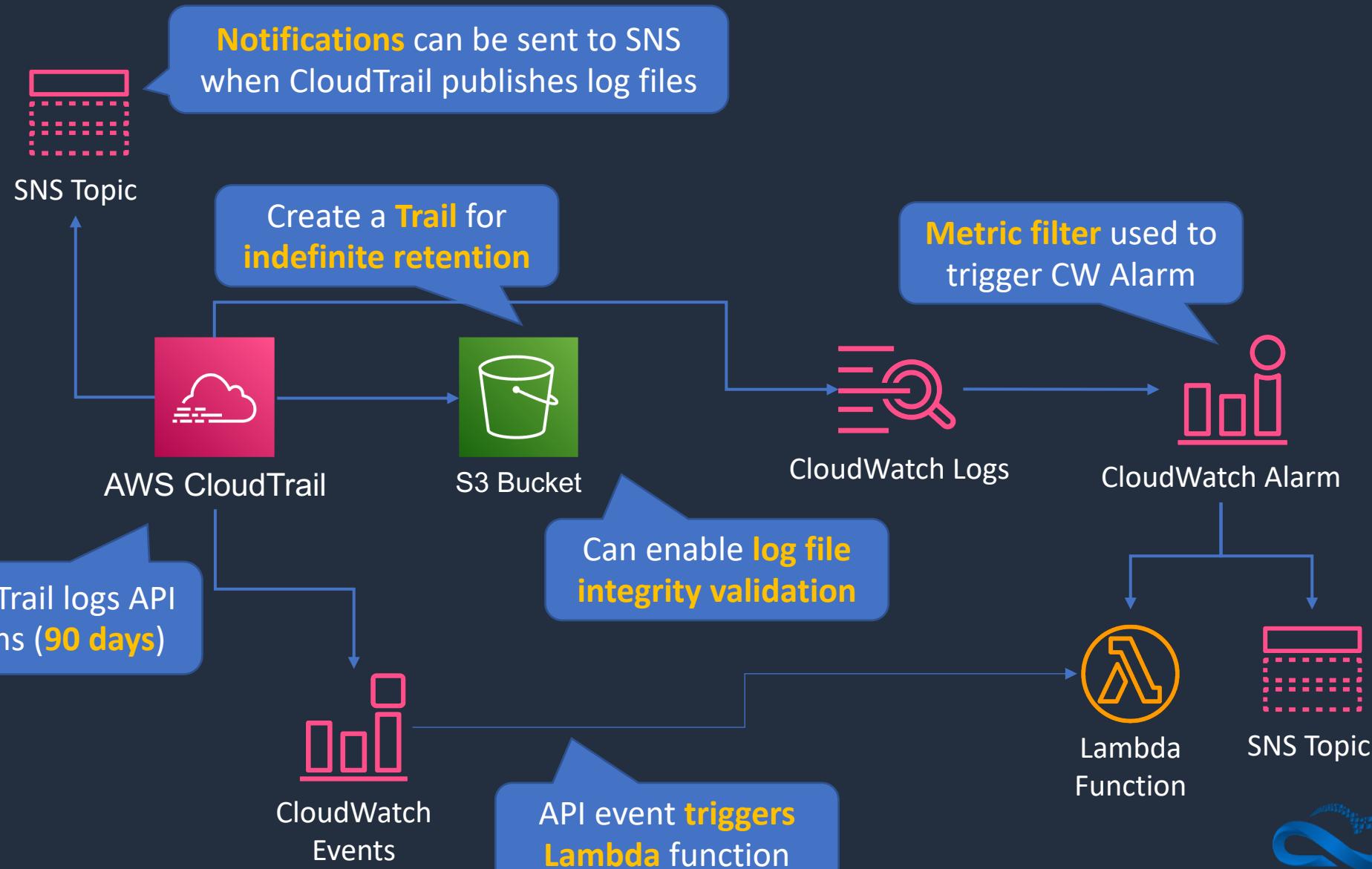


# AWS CloudTrail Use Cases





# 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



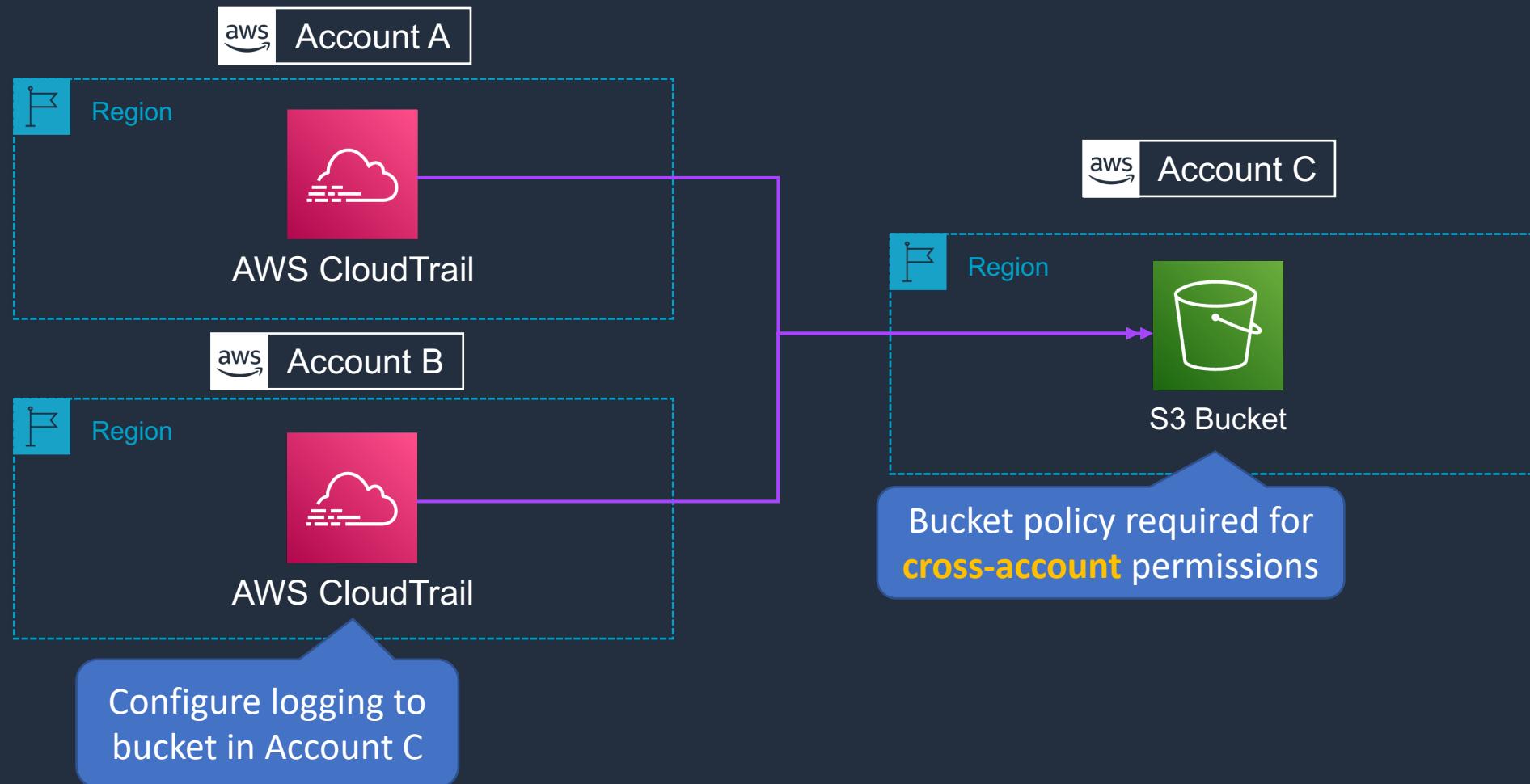
# CloudTrail – Management and Data Events

---

- **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



# CloudTrail – Multi Account and Region

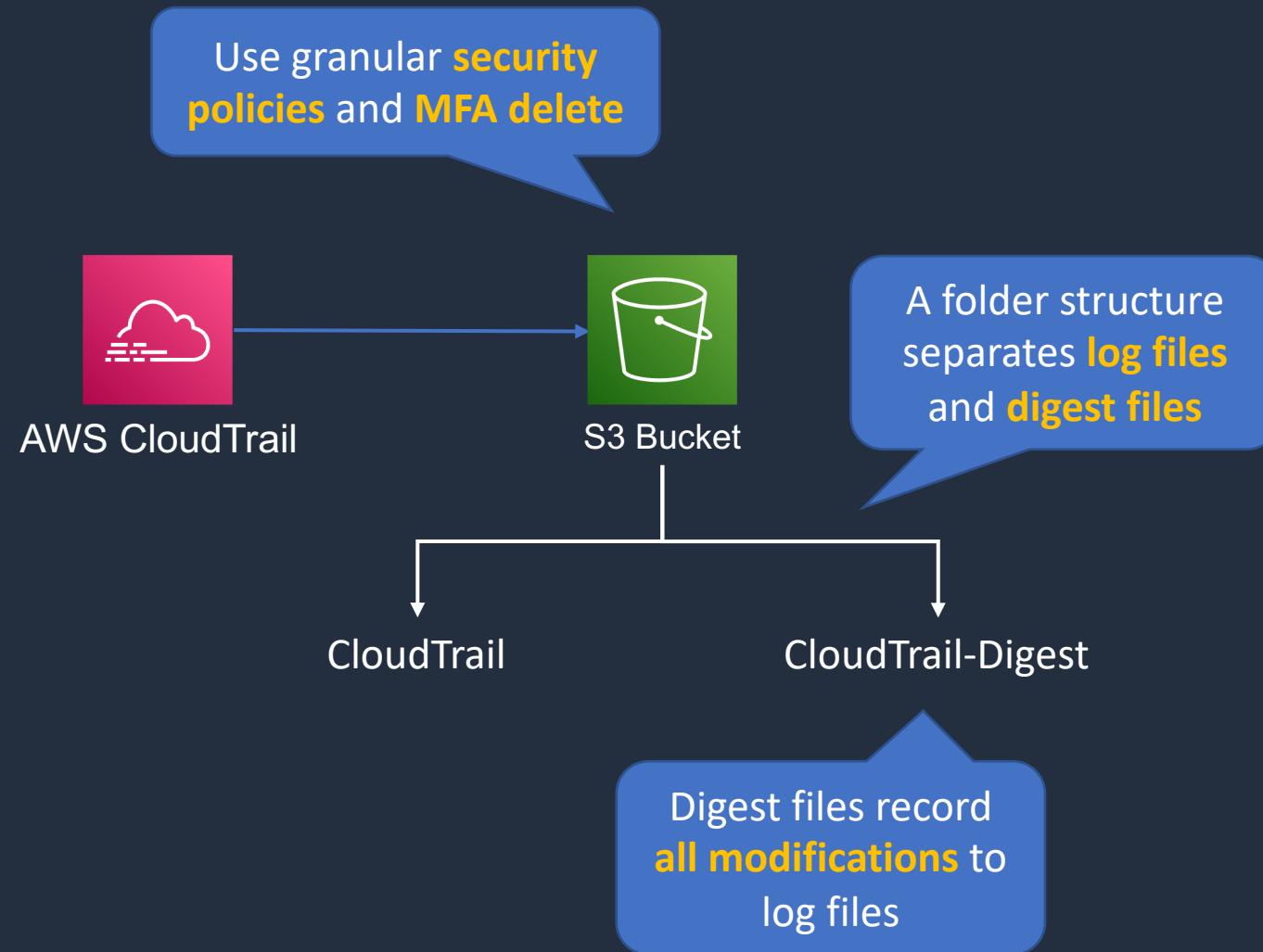


# Enable CloudTrail Log File Validation

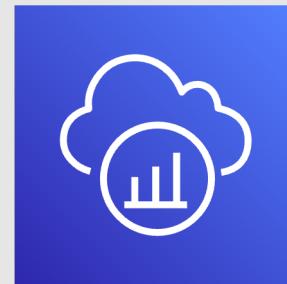




# Enable CloudTrail Log File Validation



# AWS X-Ray

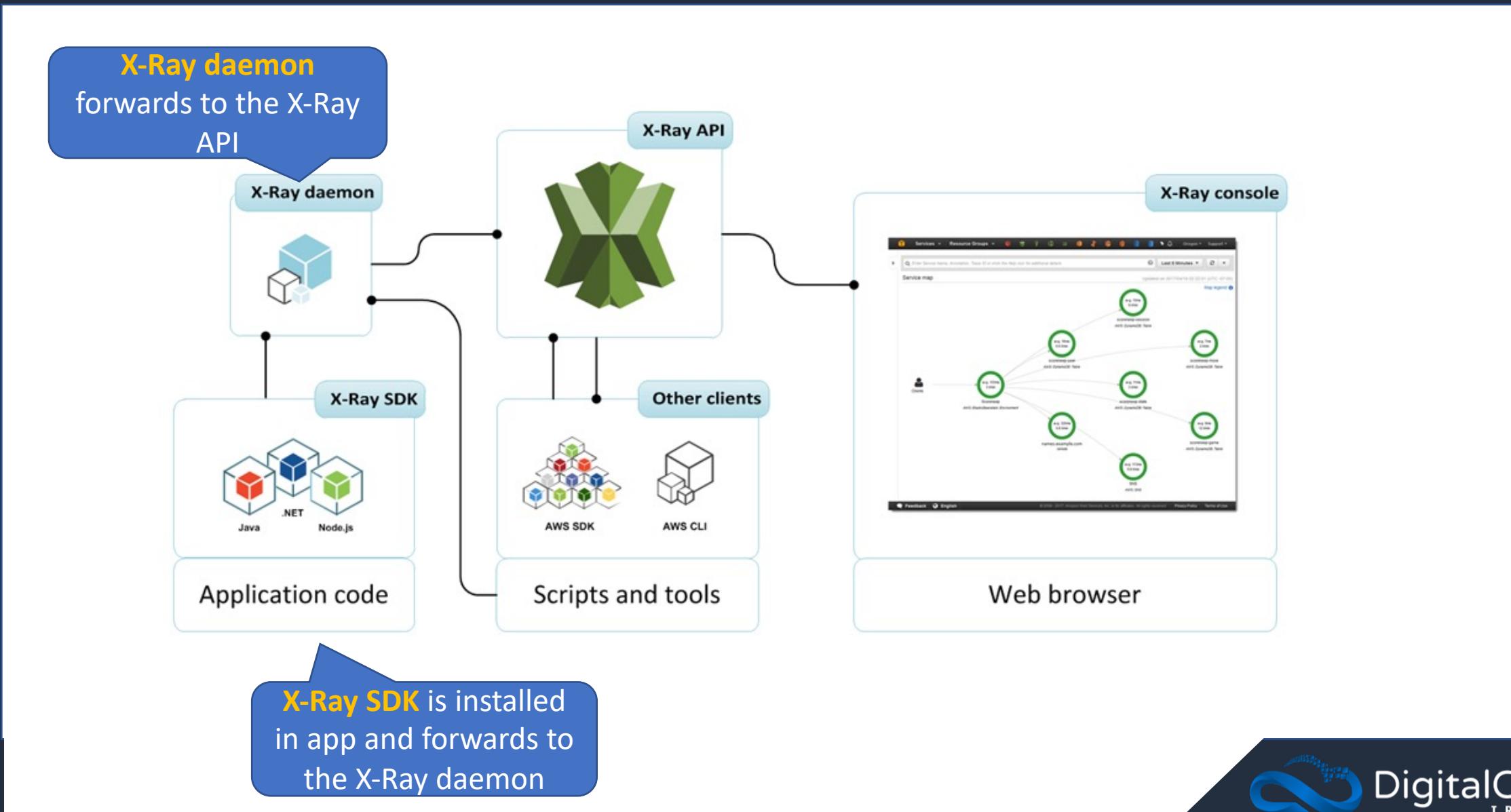




- Analyze and debug production, distributed applications, such as those built using a microservices architecture
- AWS X-Ray supports applications running on:
  - Amazon EC2
  - Amazon ECS
  - AWS Lambda
  - AWS Elastic Beanstalk
- X-Ray SDK must be integrated with application along with X-Ray agent



# AWS X-Ray





- The **X-Ray SDK** is installed in your application and forwards to the X-Ray daemon which forwards to the X-Ray API.
- You can then visualize what is happening in the X-Ray console.
- The X-Ray SDK provides:
  - **Interceptors** to add your code to trace incoming HTTP requests.
  - **Client handlers** to instrument AWS SDK client that your application uses to call other AWS services.
  - **An HTTP client** to use to instrument calls to other internal and external HTTP web services.

# 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

CloudWatch Logs entries must be analyzed and stored centrally in a security account

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

## Solution

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

Use cross-account sharing and configure a Kinesis Data Stream in the security account to collect the log files then use Lambda to analyze and store

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



# Architecture Patterns – Monitoring, Logging and Auditing

---

---

## Requirement

API activity must be captured from multiple accounts and stored in a central security account

## Solution

Use CloudTrail in each account to record API activity and use cross-account access to a security account to store the log files in a central S3 bucket

Need to trace and debug application with distributed components

Use AWS X-Ray to trace and debug the application

# SECTION 17

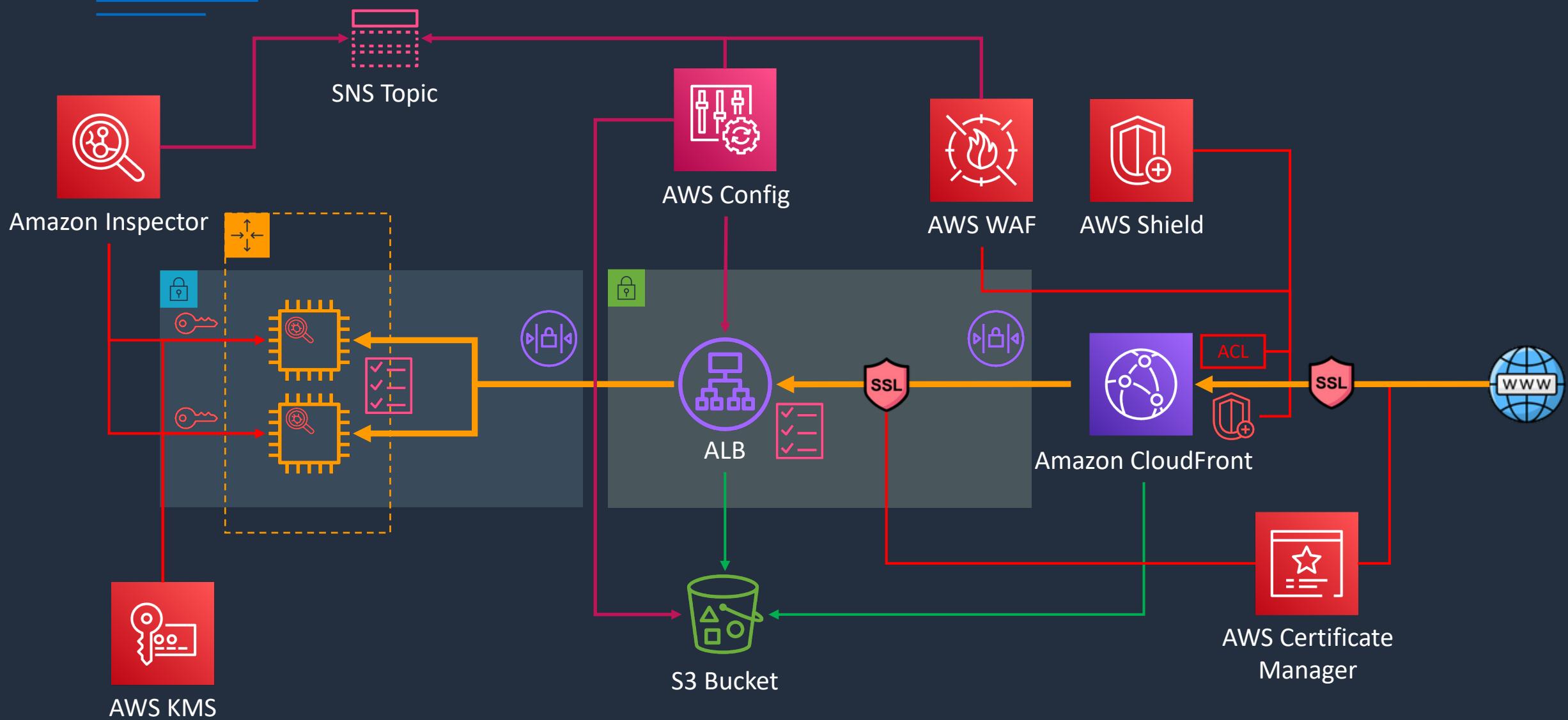
## Security Services

# Secure Multi-Tier Architecture for HOL





# Build a Secure Multi-Tier Architecture



# AWS Certificate Manager (ACM)





# 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

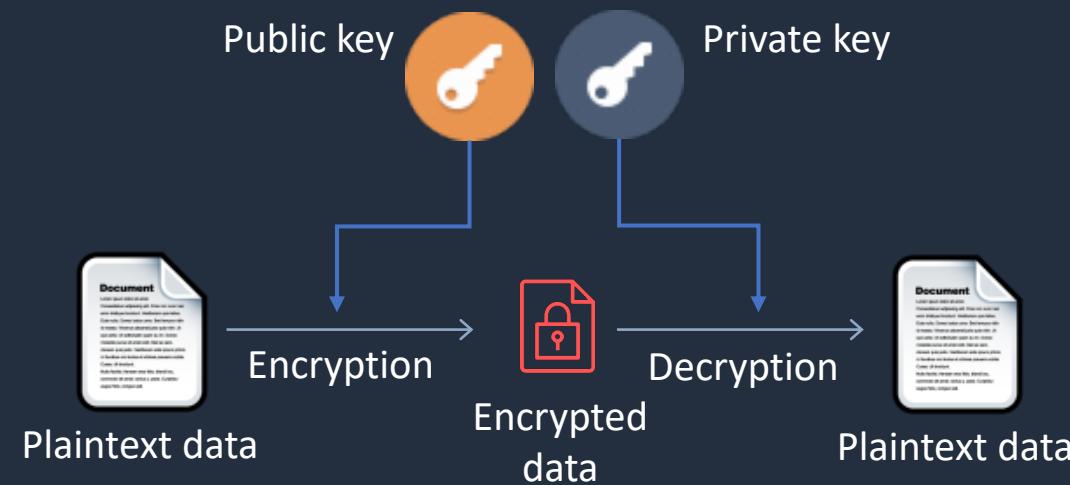


Encrypted bucket



# 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





# 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

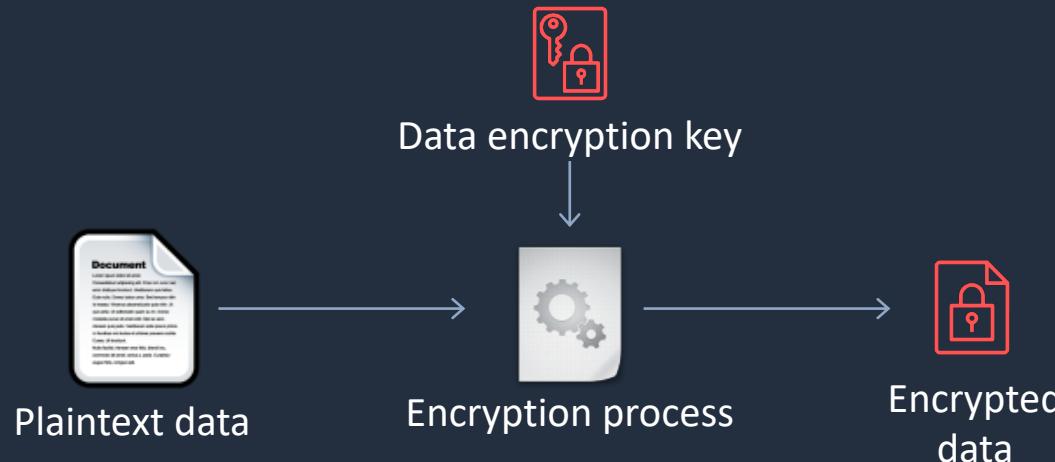
# AWS Key Management Service (KMS)



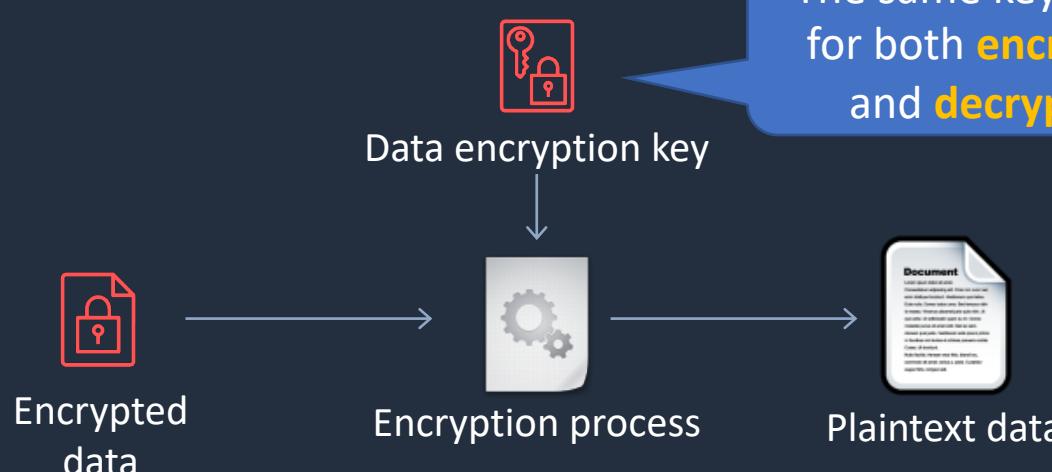


# Symmetric Encryption

## Encryption



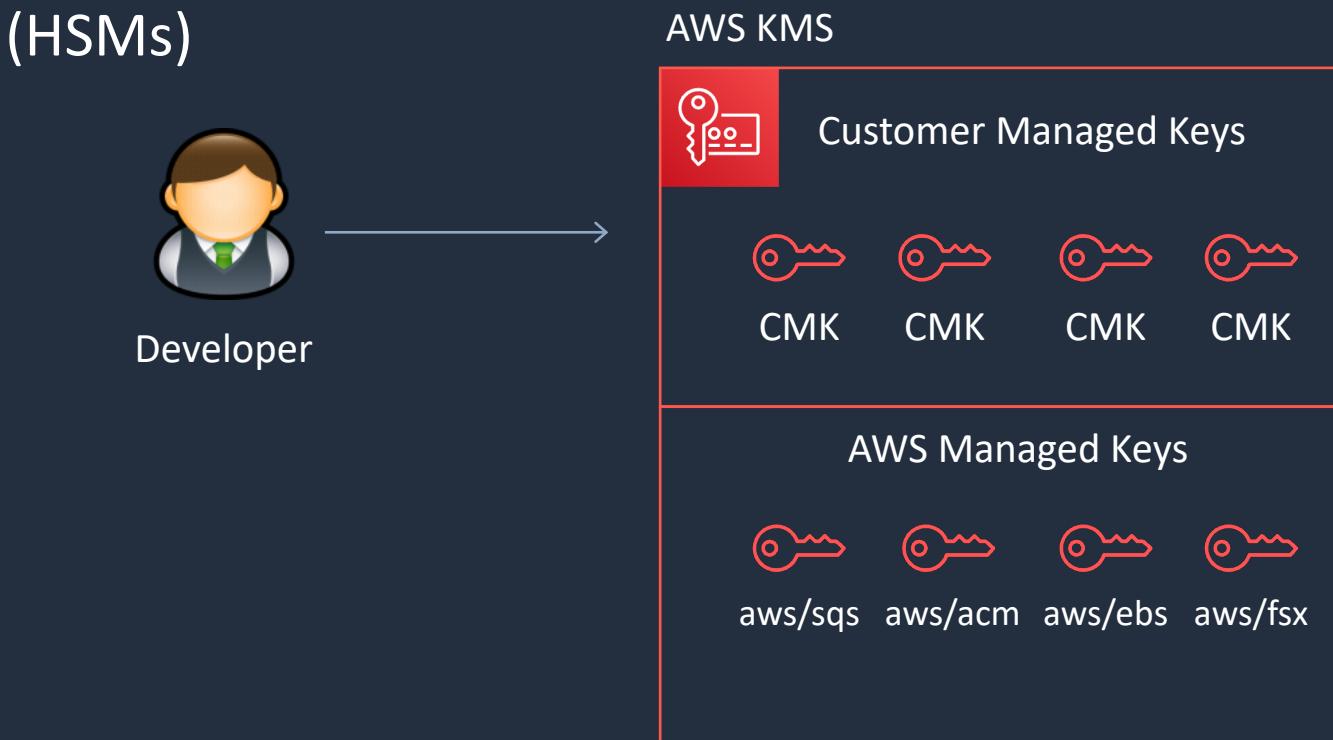
## Decryption





# AWS Key Management Service (KMS)

- Create and manage **symmetric** and **asymmetric** encryption keys
- The **customer master keys** (CMKs) are protected by hardware security modules (HSMs)





# 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 (DEKs)



# AWS Managed CMKs

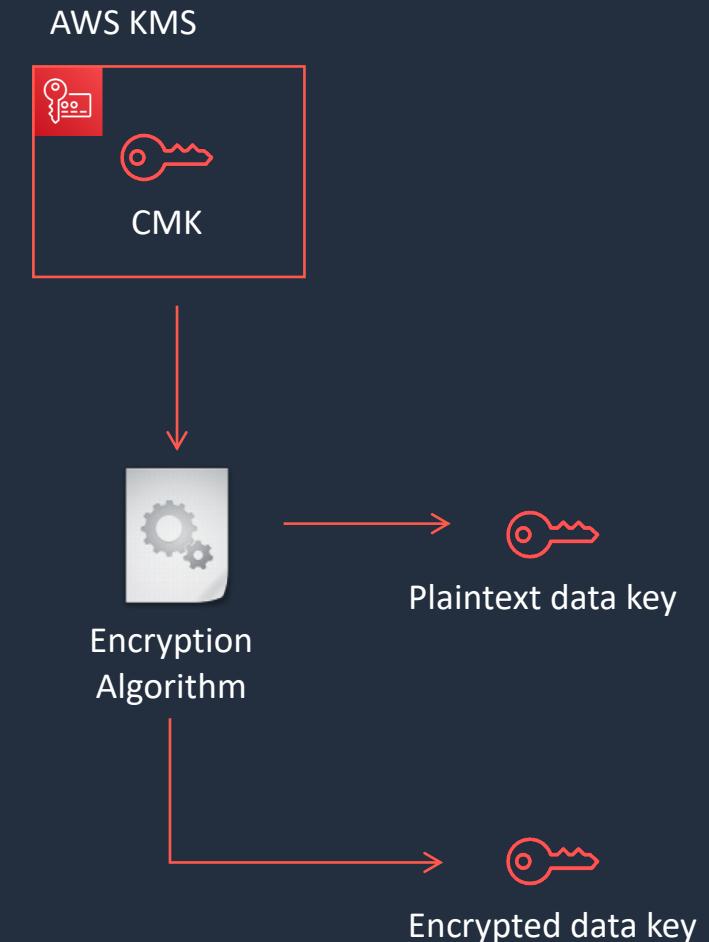
---

- 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

# 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
- Manage your own encryption keys using FIPS 140-2 Level 3 validated HSMs
- CloudHSM runs in your VPC

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



# AWS CloudHSM Benefits

---

---

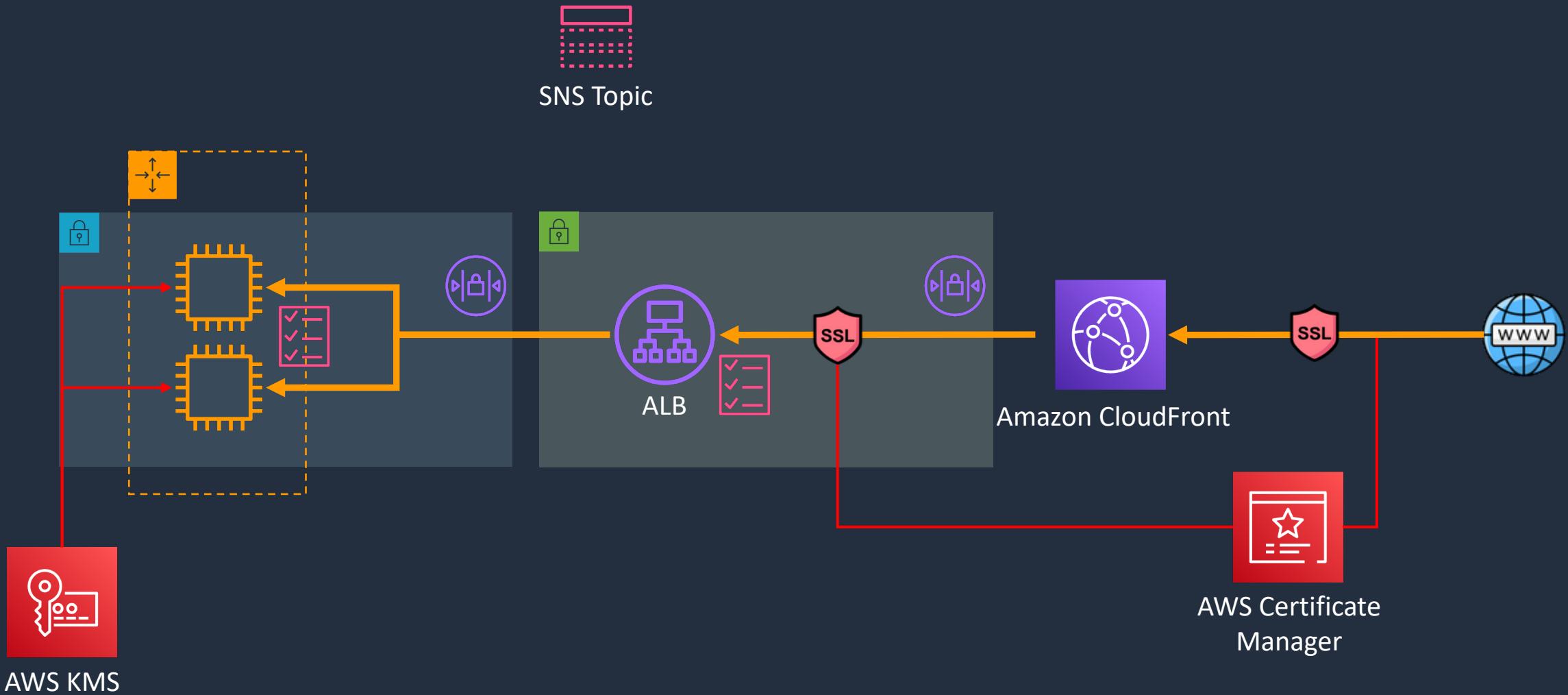
- FIPS 140-2 level 3 validated HSMs
- You can configure AWS Key Management Service (KMS) to use your AWS CloudHSM cluster as a custom key store rather than the default KMS key store
- Managed service and automatically scales
- Retain control of your encryption keys - you control access (and AWS has no visibility of your encryption keys)

# Build a Secure Multi-Tier Architecture - Part 1





# Build a Secure Multi-Tier Architecture



# Build a Secure Multi-Tier Architecture - Part 2

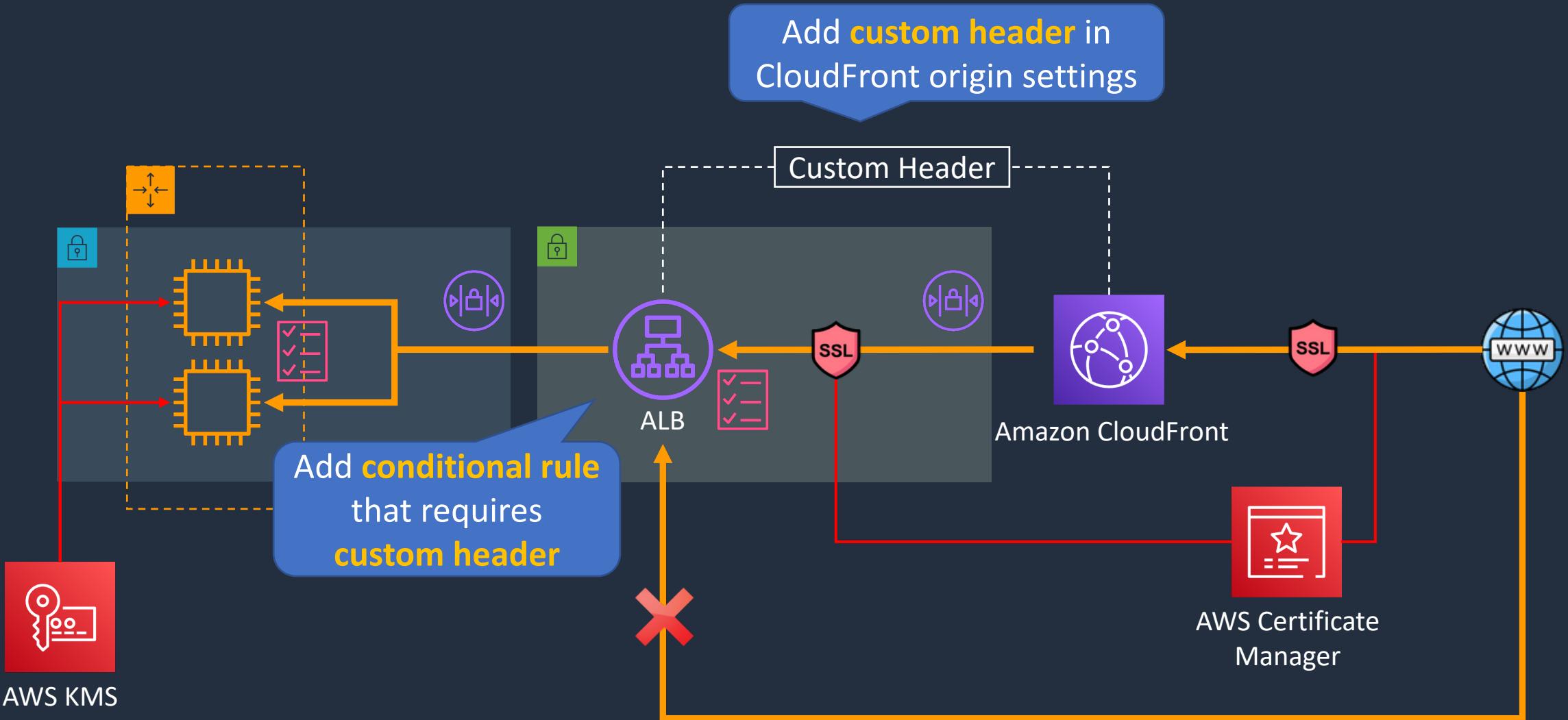


# Build a Secure Multi-Tier Architecture - Part 3





# Build a Secure Multi-Tier Architecture



# Amazon Macie





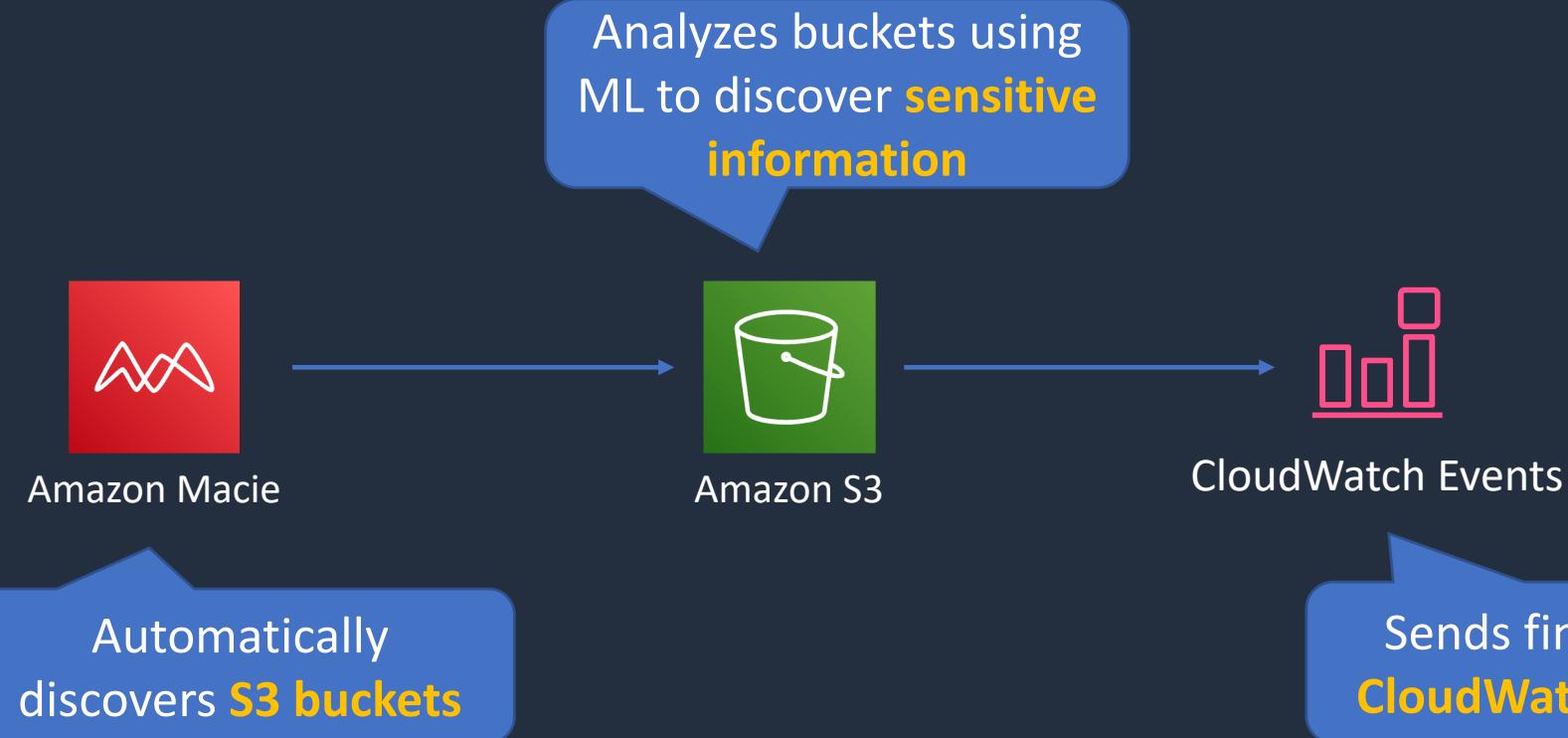
# Amazon Macie

---

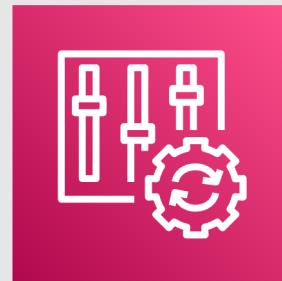
- Macie is a fully managed data security and data privacy service
- Uses machine learning and pattern matching to discover, monitor, and help you protect your sensitive data on Amazon S3
- Macie enables security compliance and preventive security as follows:
  - Identify a variety of data types, including PII, Protected Health Information (PHI), regulatory documents, API keys, and secret keys
  - Identify changes to policy and access control lists
  - Continuously monitor the security posture of Amazon S3
  - Generate security findings that you can view using the Macie console, AWS Security Hub, or Amazon EventBridge
  - Manage multiple AWS accounts using AWS Organizations



# Amazon Macie



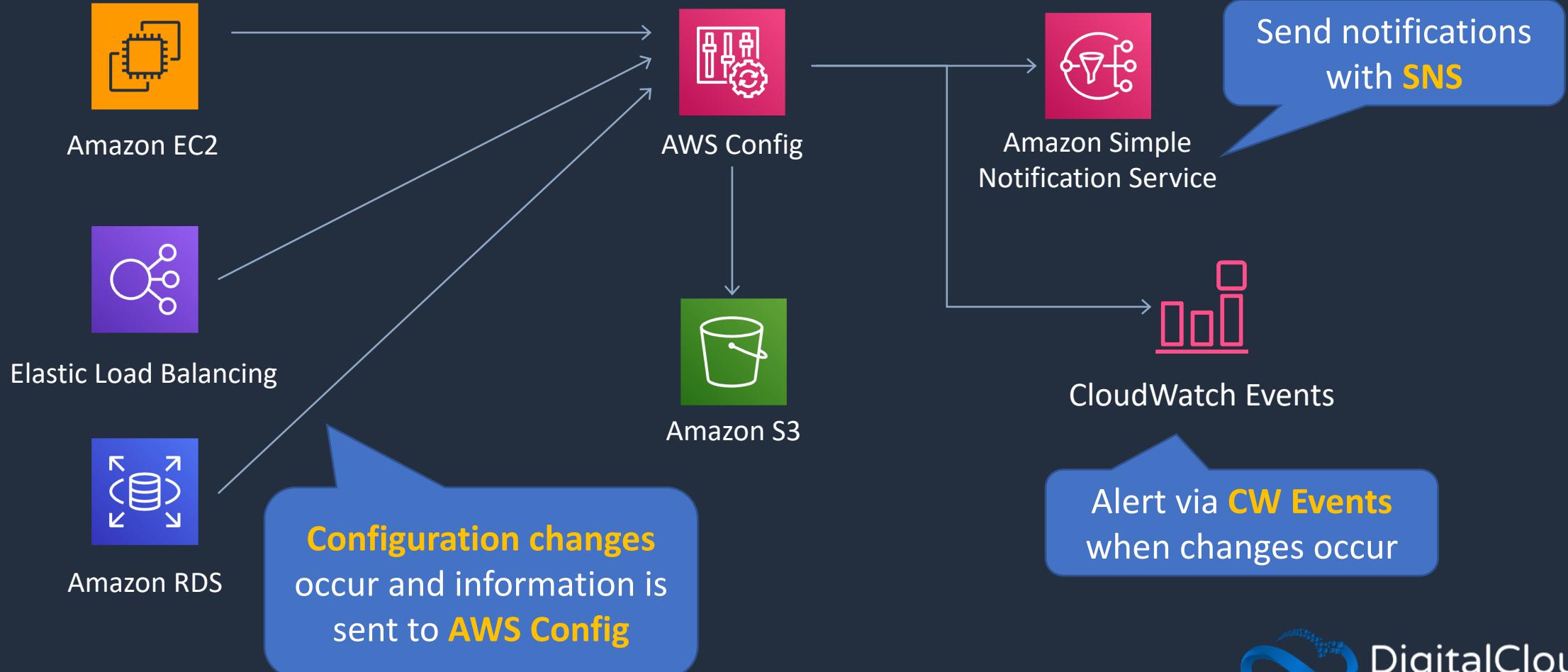
# AWS Config





# AWS Config

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

# Amazon Inspector





# Amazon Inspector

---

---

- Runs assessments that check for security exposures and vulnerabilities in EC2 instances
- Can be configured to run on a schedule
- Agent must be installed on EC2 for host assessments
- Network assessments do not require an agent



## Network Assessments

- Assessments: Network configuration analysis to check for ports reachable from outside the VPC
- If the Inspector Agent is installed on your EC2 instances, the assessment also finds processes reachable on port
- Price based on the number of instance assessments



## Host Assessments

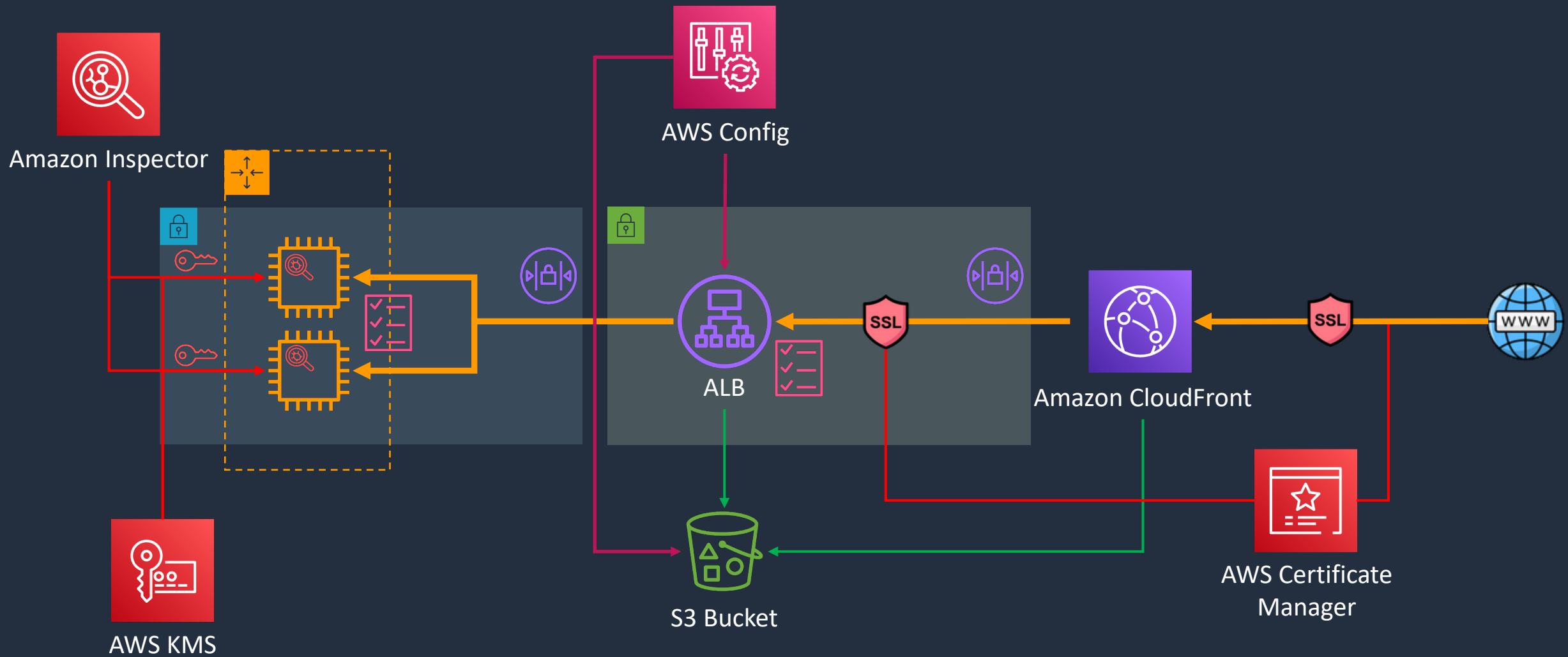
- Assessments: Vulnerable software (CVE), host hardening (CIS benchmarks), and security best practices
- Requires an agent (auto-install with SSM Run Command)
- Price based on the number of instance assessments

# Build a Secure Multi-Tier Architecture - Part 4





# Build a Secure Multi-Tier Architecture



# AWS Web Application Firewall (WAF)





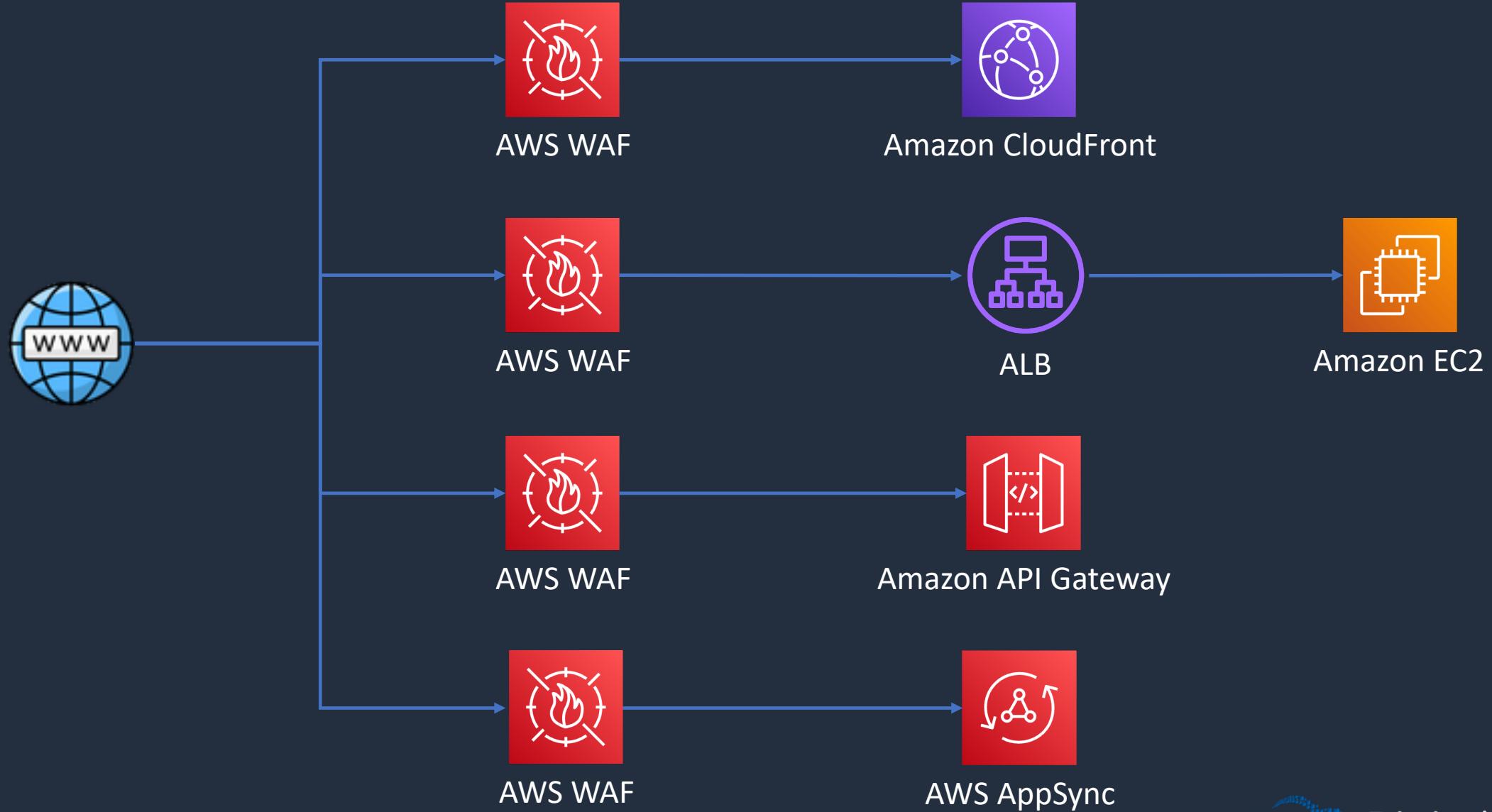
# AWS WAF

---

- 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





- **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

---

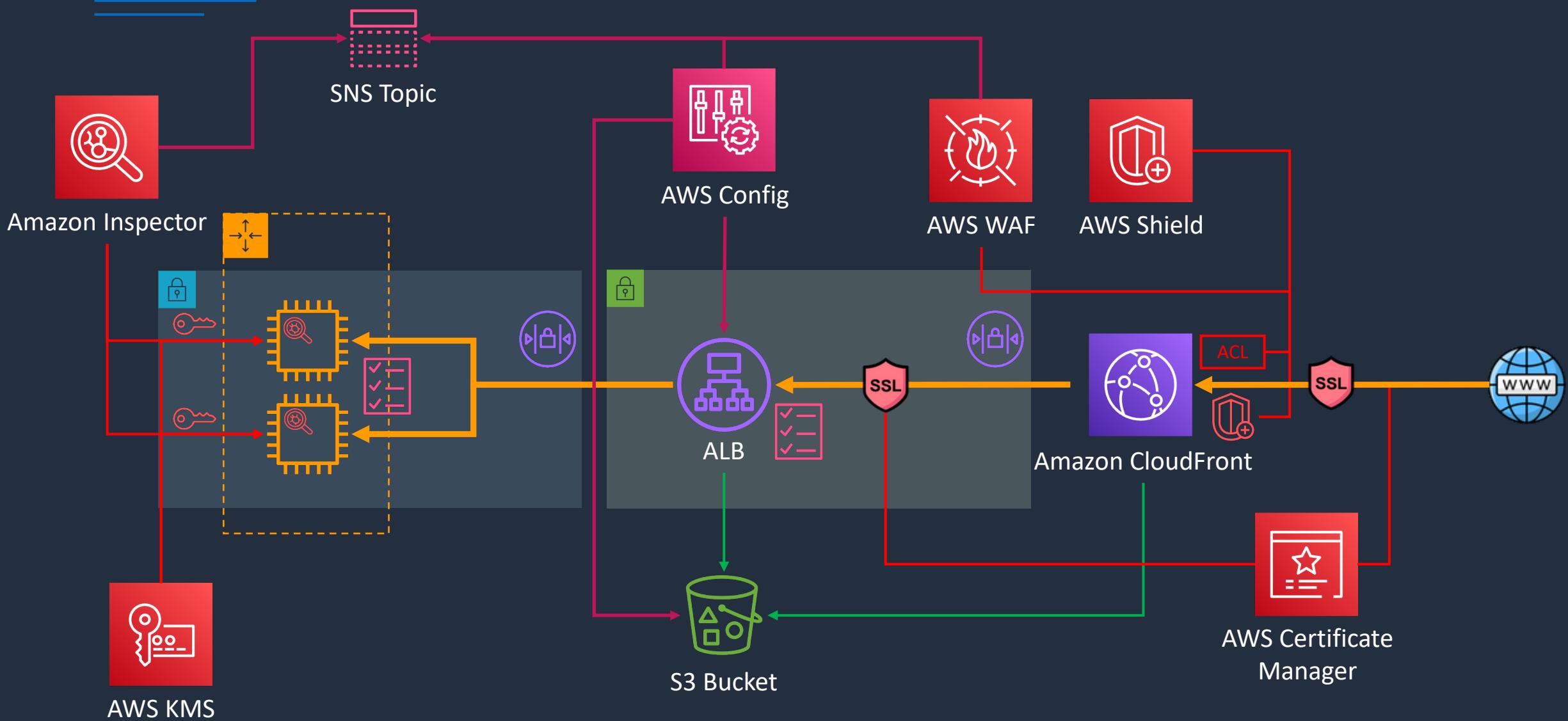
- AWS Shield is a managed Distributed Denial of Service (DDoS) protection service
- Safeguards web application running on AWS with always-on detection and automatic inline mitigations
- Helps to minimize application downtime and latency
- Two tiers –
  - **Standard** – no cost
  - **Advanced** - \$3k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

# Build a Secure Multi-Tier Architecture - Part 5





# Build a Secure Multi-Tier Architecture



# AWS GuardDuty





# AWS GuardDuty

---

- Intelligent threat detection service
- Detects account compromise, instance compromise, malicious reconnaissance, and bucket compromise
- Continuous monitoring for events across:
  - **AWS CloudTrail Management Events**
  - **AWS CloudTrail S3 Data Events**
  - **Amazon VPC Flow Logs**
  - **DNS Logs**

# 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

Company records customer information in CSV files in an Amazon S3 bucket and must not store PII data

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

## 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)

Use Amazon Macie to scan the S3 bucket for any PII data

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



## Requirement

EC2 instances must be checked against CIS benchmarks every 7 days

Website running on EC2 instances behind an 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

Install the Amazon Inspector agent and configure a host assessment every 7 days

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

# SECTION 18

## Cost Management

# Pricing for AWS Services





# The Fundamentals of AWS Pricing

---

## Compute



Amount of resources such as CPU and RAM and duration

## Storage



Quantity of data stored

## Outbound Data Transfer



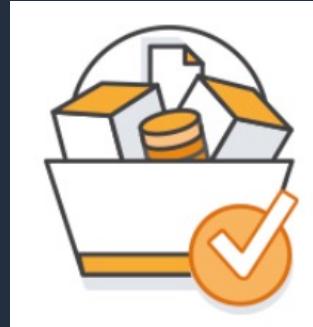
Quantity of data that is transferred out from all services



# How you pay for AWS

---

## Pay-as-you-go



No commitment; pure pay-as-you-go with no contracts or termination fees

## Save when you reserve



Reserve capacity for 1 or 3 years for a substantial discount (up to 75%)

## Pay less by using more



Some pricing is tiered, if you use more you pay less per unit



# What you need to know for the exam

---

---

## You don't need to know:

- Exact prices /rates, e.g. price per GB on S3

## You need to know how to answer questions such as these:

- Which is more expensive: ElastiCache or an RDS Read Replica?
- What's the most cost-effective block storage for 3000 IOPS?
- Which data transfer is charged vs not charged – e.g. inbound, outbound, inter-region, inter-AZ,

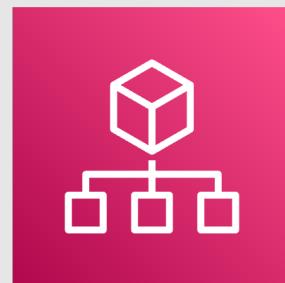


# Tools and Research

---

- **AWS Pricing Calculator:** <https://calculator.aws/#/>
- **Pricing pages for AWS services** e.g.:
  - <https://aws.amazon.com/ec2/pricing/>
  - <https://aws.amazon.com/rds/pricing/>
  - <https://aws.amazon.com/dynamodb/pricing/>

# Consolidated Billing





# Consolidated Billing

- Feature of **AWS Organizations**
- Benefits:
  - **One bill** – You get one bill for multiple accounts
  - **Easy tracking** – You can track the charges across multiple accounts and download the combined cost and usage data
  - **Combined usage** – You can combine the usage across all accounts in the organization to share the volume pricing discounts, Reserved Instance discounts, and Savings Plans.
  - **No extra fee** – Consolidated billing is offered at no additional cost



# Consolidated Billing

Tier Description	Price Per GB	Price Per TB
First 1 TB/month	\$0.10	\$100.00
Next 49 TB/month	\$0.08	\$80.00
Next 450 TB/month	\$0.06	\$60.00

## Usage within Organization:

Account A (master) usage: 2 TB

Account B usage: 80 TB

Account C usage: 120 TB

Total: 202 TB

## Calculation:

First 1 TB = \$100.00

Next 49 TB = \$3,920.00

Next 157 TB = \$9,120.00

Total cost = \$13,140.00

# AWS Budgets





# AWS Budgets

All budgets (1)	Cost budgets (1)	Usage budgets (0)	Reservation budgets (0)	Savings Plans budgets (0)		
Budget name	Type	Current	Budgeted	Forecasted	Current vs. budgeted	Forecasted vs. budgeted
MyBudget	Cost	\$13.20	\$100.00	\$129.47	<div style="width: 13.2%; background-color: #0070C0;"></div> 13.2%	<div style="width: 129.47%; background-color: #E74C3C;"></div> 129.47%

**Set Custom Budgets** - set custom usage and reservation budgets

**Configure Alerts** – receive alerts when you exceed or are forecast to exceed your alert thresholds

**Integrated with other AWS services** – Includes Cost Explorer Chatbot, and Service Catalog

# Cost Allocation Tags

