

# AWS Certified Security Specialty

By Stéphane Maarek



COURSE →



EXTRA PRACTICE EXAMS →

# Disclaimer: These slides are copyrighted and strictly for personal use only

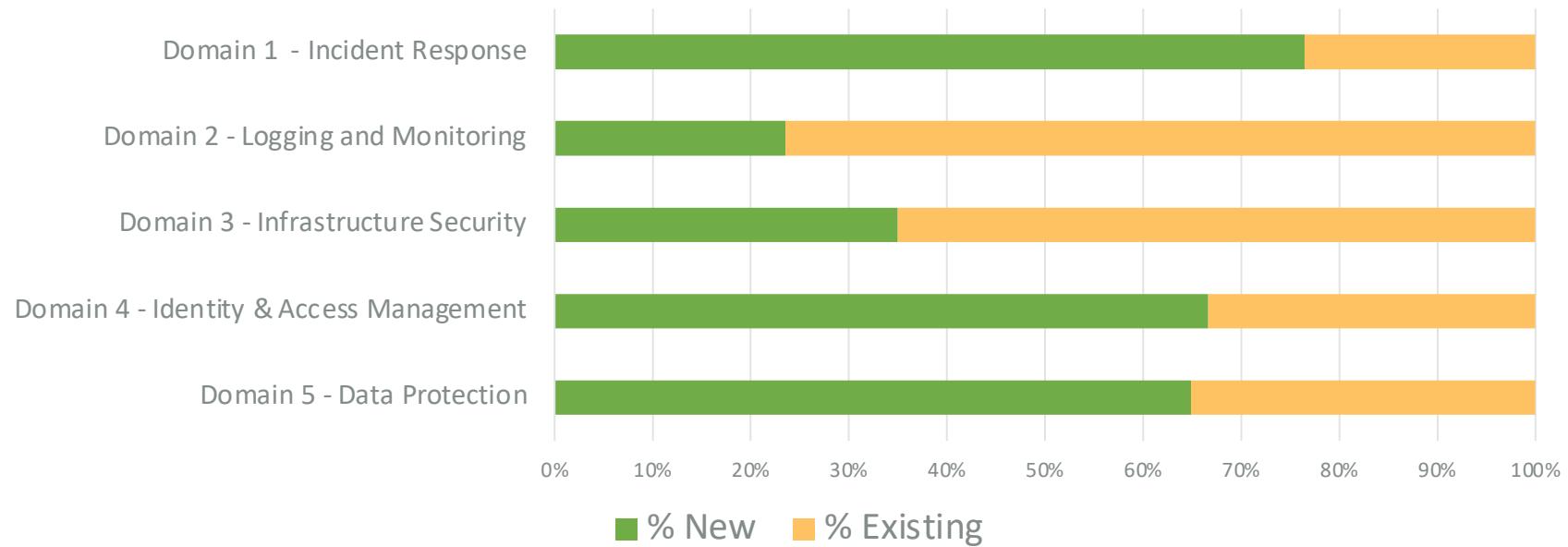
- This document is reserved for people enrolled into the [AWS Certified Cloud Practitioner Course](#)
- Please do not share this document, it is intended for personal use and exam preparation only, thank you.
- Best of luck for the exam and happy learning!

# AWS Certified Security Specialty Course

## SCS-C01

# Course Structure

- One section per domain + extra section for additional knowledge
- Some lectures are taken from other courses (basic knowledge)
  - Example: [SAA] CloudTrail (values are : [CCP], [SAA], [DVA], [SOA])



# About me

- I'm Stephane!
- 10x AWS Certified (so far!)
- Worked with AWS many years; I built websites, apps, streaming platforms
- Veteran Instructor on AWS Certifications
- You can find me on
  - LinkedIn: <https://www.linkedin.com/in/stephanemaarek>
  - Instagram: <https://www.instagram.com/stephanemaarek/>
  - Twitter: <https://twitter.com/stephanemaarek>
  - GitHub: <https://github.com/simplesteph>
  - Medium: <https://medium.com/@stephane.maarek>



4.7 Instructor Rating  
 554,165 Reviews  
 1,819,442 Students  
 43 Courses

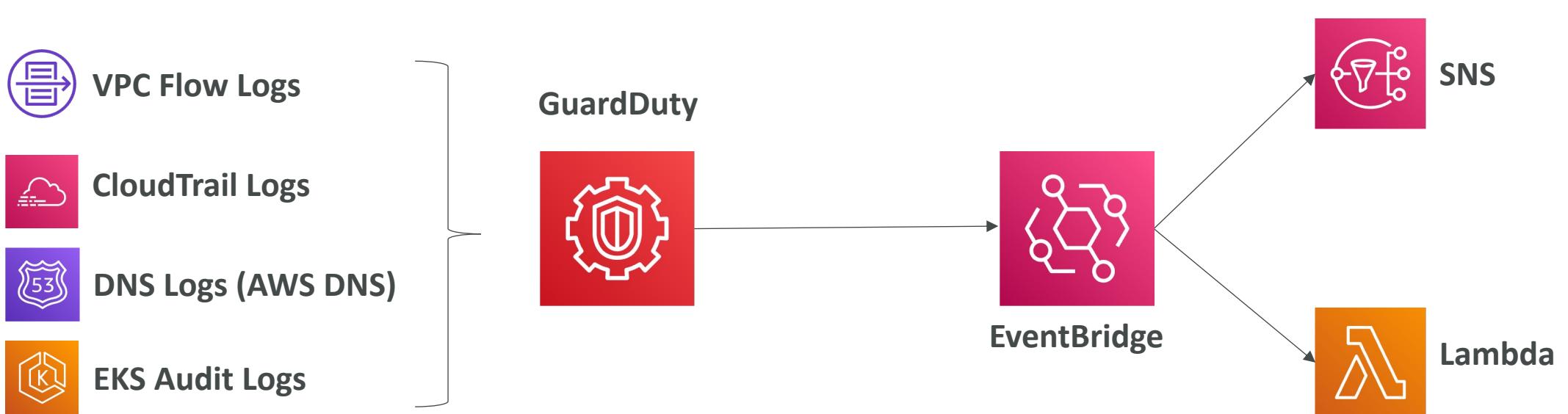
# Domain I – Incident Response



# Amazon GuardDuty

- Intelligent Threat discovery to protect your AWS Account
- Uses Machine Learning algorithms, anomaly detection, 3<sup>rd</sup> party data
- One click to enable (30 days trial), no need to install software
- Input data includes:
  - **CloudTrail Events Logs** – unusual API calls, unauthorized deployments
    - CloudTrail Management Events – create VPC subnet, create trail, ...
    - CloudTrail S3 Data Events – get object, list objects, delete object, ...
  - **VPC Flow Logs** – unusual internal traffic, unusual IP address
  - **DNS Logs** – compromised EC2 instances sending encoded data within DNS queries
  - **Kubernetes Audit Logs** – suspicious activities and potential EKS cluster compromises
- Can setup **EventBridge rules** to be notified in case of findings
- EventBridge rules can target AWS Lambda or SNS
- **Can protect against CryptoCurrency attacks** (has a dedicated “finding” for it)

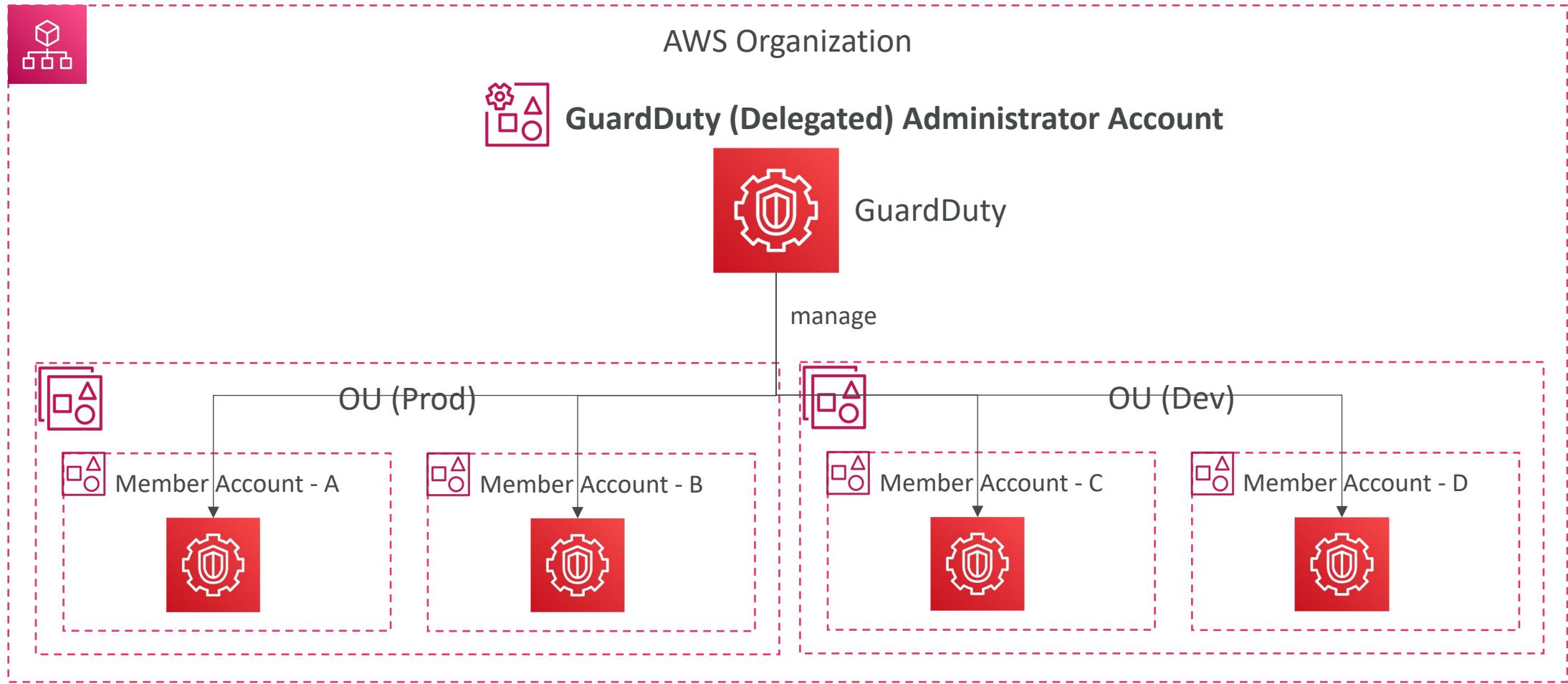
# Amazon GuardDuty



# Amazon GuardDuty – Multi-Account Strategy

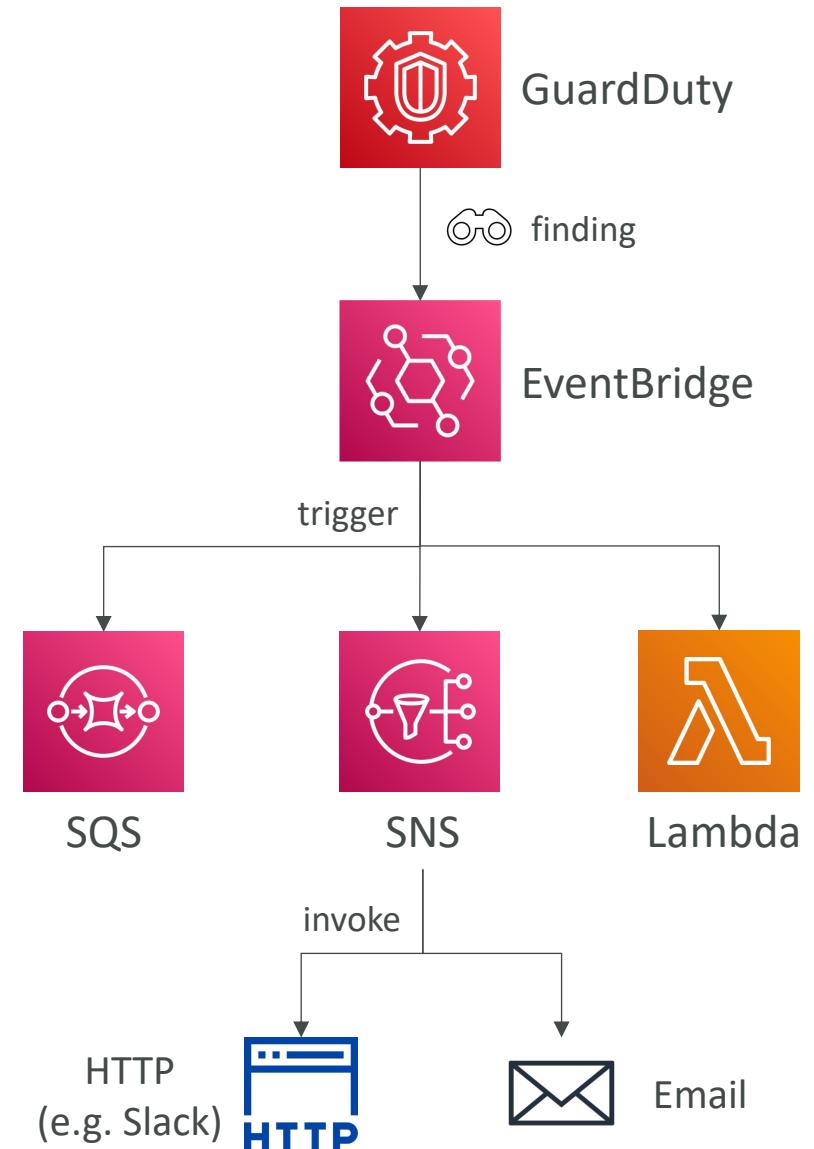
- You can manage multiple accounts in GuardDuty
- Associate the Member accounts with the Administrator account
  - Through an AWS Organization
  - Sending invitation through GuardDuty
- Administrator account can:
  - Add and remove member accounts
  - Manage GuardDuty within the associated member accounts
  - Manage findings, suppression rules, trusted IP lists, threat lists
- In an AWS Organization, you can specify a member account as the Organization's **delegated administrator for GuardDuty**

# Amazon GuardDuty – Multi-Account Strategy

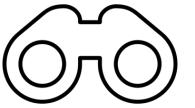


# Amazon GuardDuty – Findings Automated Response

- Findings are potential security issue for malicious events happening in your AWS account
- Automate response to security issues revealed by GuardDuty Findings using EventBridge
- Send alerts to SNS (email, Lambda, Slack, Chime, ...)
- Events are published to both the administrator account and the member account that it is originated from



# Amazon GuardDuty – Findings

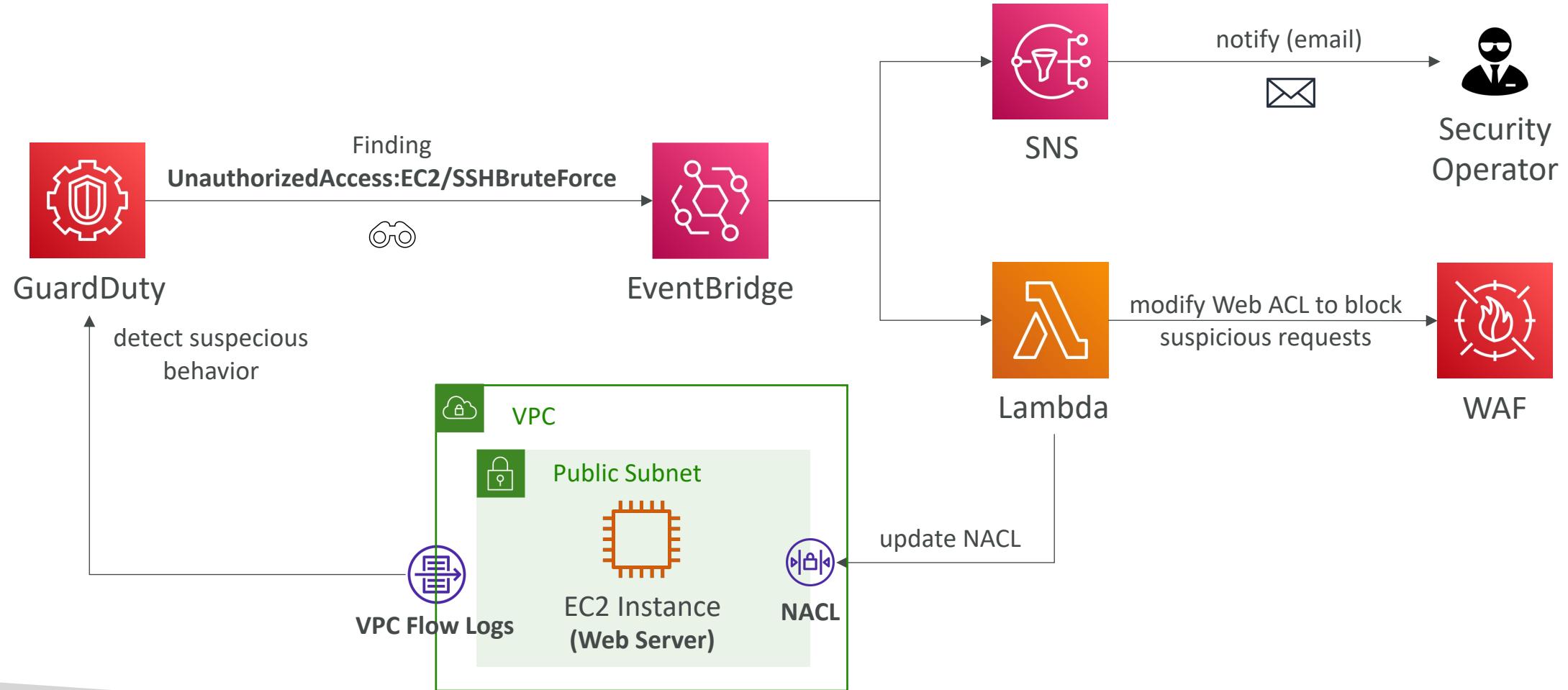


- GuardDuty pulls independent streams of data directly from CloudTrail logs (management events, data events), VPC Flow Logs or EKS logs
- Each finding has a severity value between 0.1 to 8+ (High, Medium, Low)
- Finding naming convention  
*ThreatPurpose:ResourceTypeAffected/ThreatFamilyName.DetectionMechanism!Artifact*
  - ThreatPurpose – primary purpose of the threat (e.g., Backdoor, CryptoCurrency)
  - ResourceTypeAffected – which AWS resource is the target (e.g., EC2, S3)
  - ThreatFamilyName – describes the potential malicious activity (e.g., NetworkPortUnusual)
  - DetectionMechanism – method GuardDuty detecting the finding (e.g., Tcp, Udp)
  - Artifact – describes a resource that is used in the malicious activity (e.g., DNS)
- You can generate sample findings in GuardDuty to test your automations

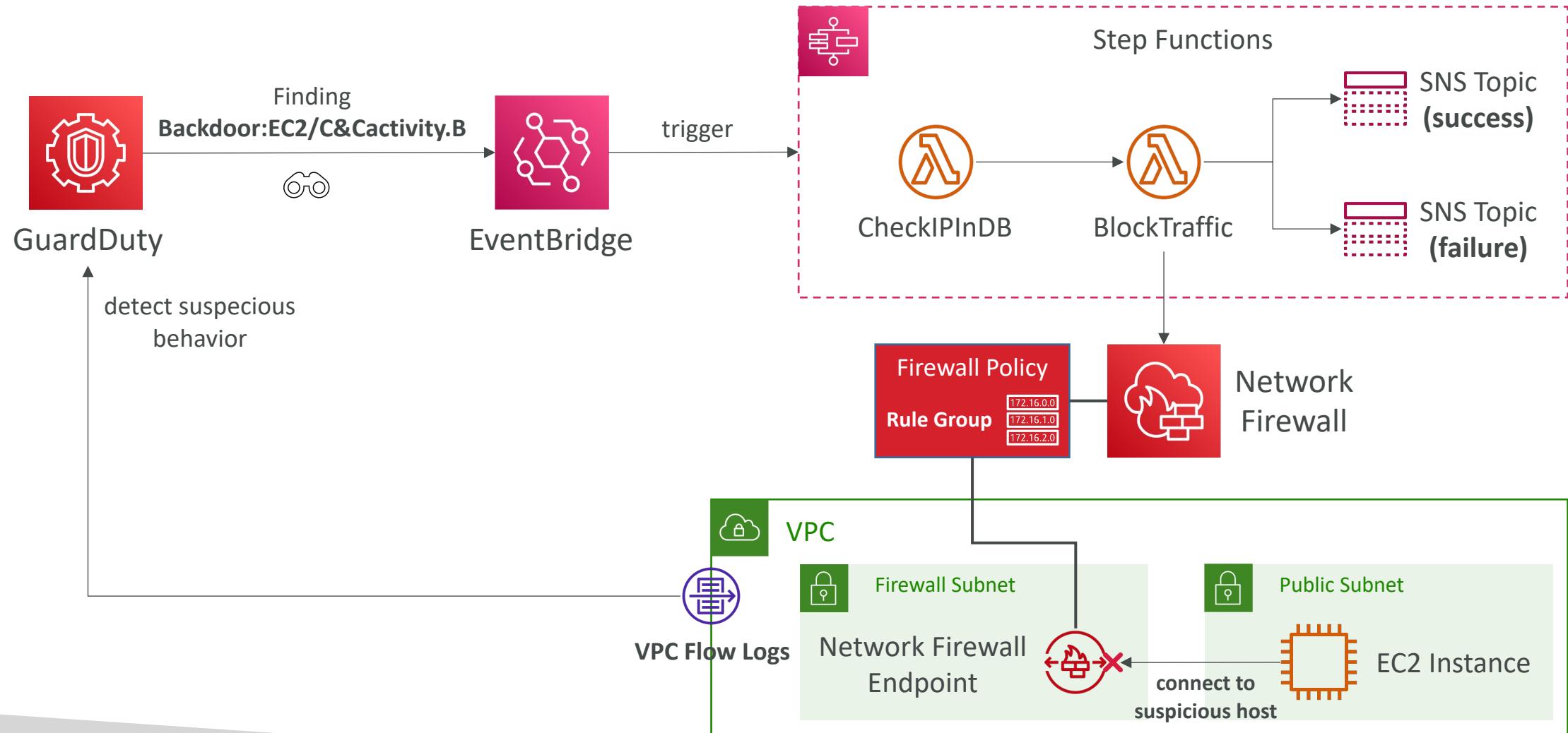
# Amazon GuardDuty – Findings Types

- EC2 Finding Types
  - UnauthorizedAccess:EC2/SSHBruteForce, CryptoCurrency:EC2/BitcoinToo.B!DNS
- IAM Finding Types
  - Stealth:IAMUser/CloudTrailLoggingDisabled, Policy:IAMUser/RootCredentialUsage
- Kubernetes Audit Logs Finding Types
  - CredentialAccess:Kubernetes/MaliciousIPCaller
- Malware Protection Finding Types
  - Execution:EC2/SuspiciousFile, Execution:ECS/ SuspiciousFile
- RDS Protection Finding Types
  - CredentialAccess:RDS/AnomalousBehavior:SuccessfulLogin
- S3 Finding Types
  - Policy:S3/AccountBlockPublicAccessDisabled, PenTest:S3/KaliLinux

# Amazon GuardDuty – Architectures

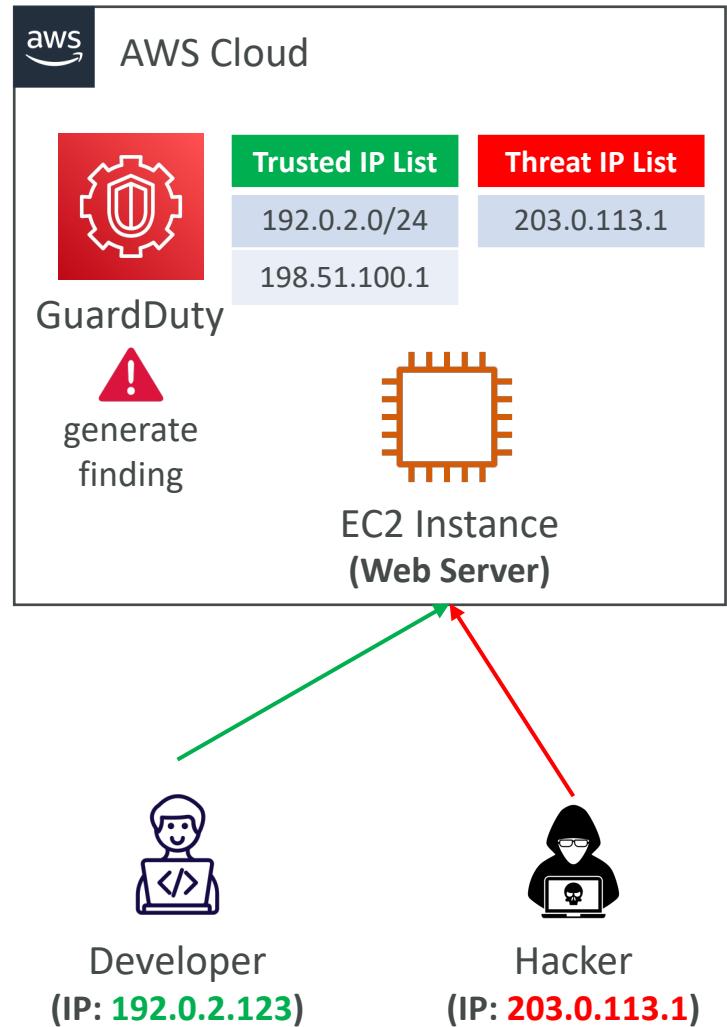


# Amazon GuardDuty – Architectures



# Amazon GuardDuty – Trusted and Threat IP Lists

- Works only for public IP addresses
- **Trusted IP List**
  - List of IP addresses and CIDR ranges that you trust
  - GuardDuty doesn't generate findings for these trusted lists
- **Threat IP List**
  - List of known malicious IP addresses and CIDR ranges
  - GuardDuty generates findings based on these threat lists
  - Can be supplied by 3rd party threat intelligence or created custom for you
- In a multi-account GuardDuty setup, only the GuardDuty administrator account can manage those lists

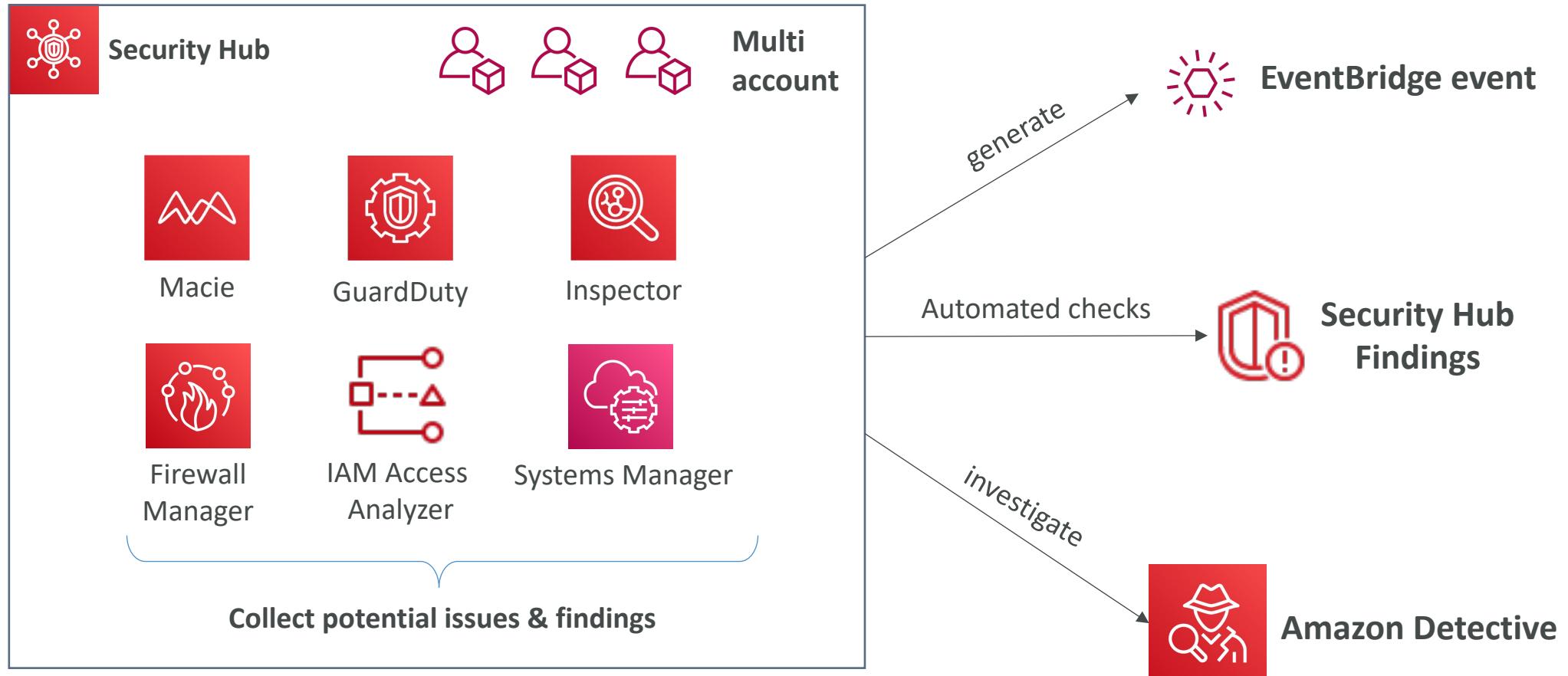




# AWS Security Hub

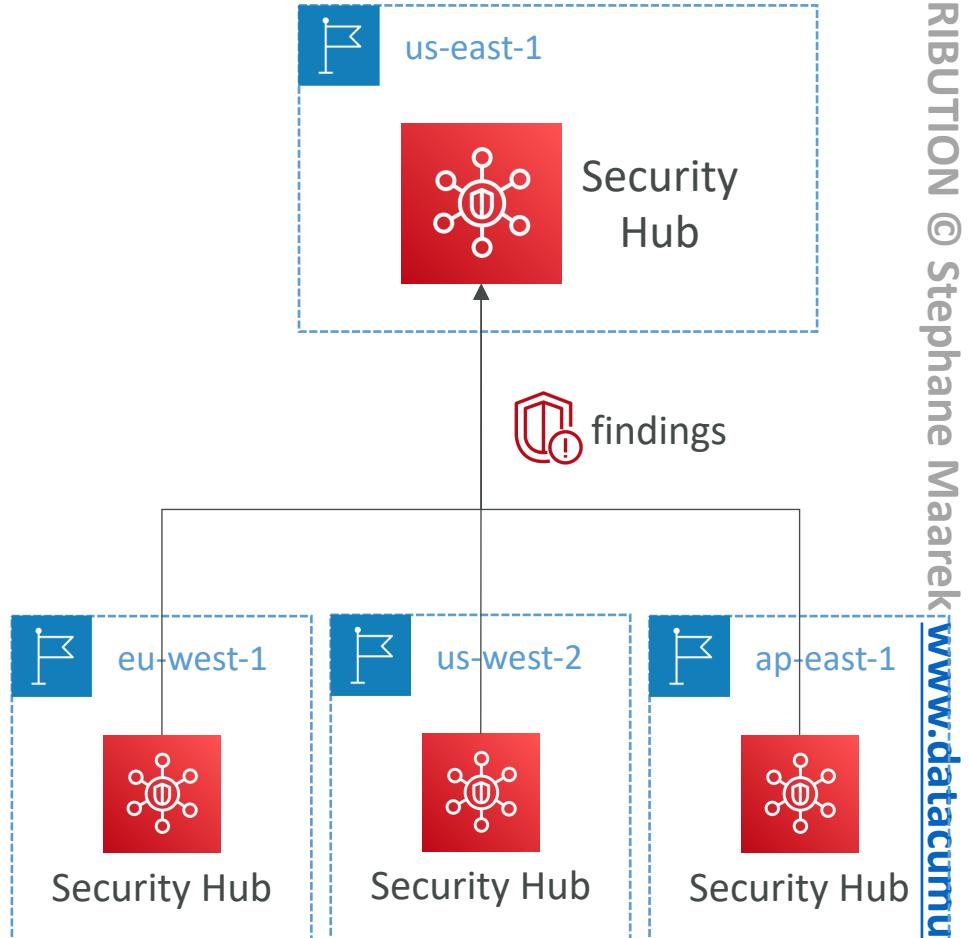
- Central security tool to manage security across several AWS accounts and automate security checks
- Integrated dashboards showing current security and compliance status to quickly take actions
- Automatically aggregates alerts in predefined or personal findings formats from various AWS services & AWS partner tools:
  - GuardDuty
  - Inspector
  - Macie
  - IAM Access Analyzer
  - AWS Systems Manager
  - AWS Firewall Manager
  - AWS Partner Network Solutions
- Must first enable the AWS Config Service

# AWS Security Hub



# Security Hub – Main Features

- **Cross-Region Aggregation** – aggregate findings, insights, and security scores from multiple Regions to a single aggregation Region
- **AWS Organizations Integration**
  - Manage all accounts in the Organization
  - Security Hub automatically detects new accounts
  - By default, Organization management account is the Security Hub administrator
  - Ability to have a designated Security Hub administrator from member accounts
- **AWS Config must be enabled**
  - Security Hub uses AWS Config to perform its security checks
  - Must be enabled on all accounts (Security Hub does NOT manage AWS Config)



# Security Hub – Security Standards

- Security Hub generates findings and continuous checks against the rules in a set of supported security standards
- Security Hub supports the following standards: CIS AWS Foundations, PCI DSS, AWS Foundational Security Best Practices...
- Ability to enable/disable a security standard

**Security standards**

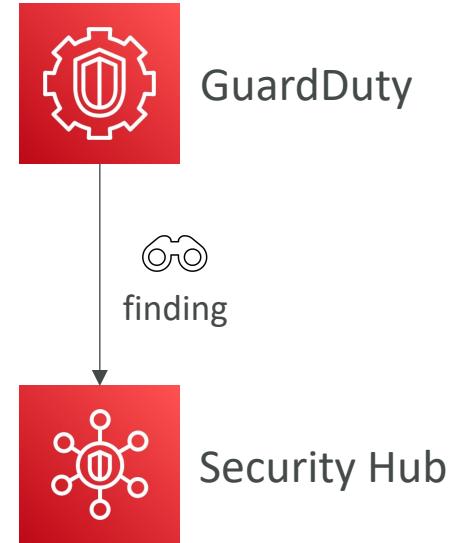
Enabling AWS Security Hub grants it permissions to conduct security checks. **Service Linked Roles (SLRs)** with the following services are used to conduct security checks: Amazon CloudWatch, Amazon SNS, AWS Config, and AWS CloudTrail.

<input checked="" type="checkbox"/> Enable AWS Foundational Security Best Practices v1.0.0
<input checked="" type="checkbox"/> Enable CIS AWS Foundations Benchmark v1.2.0
<input type="checkbox"/> Enable CIS AWS Foundations Benchmark v1.4.0
<input type="checkbox"/> Enable PCI DSS v3.2.1

# Amazon GuardDuty – Integrations with Security Hub

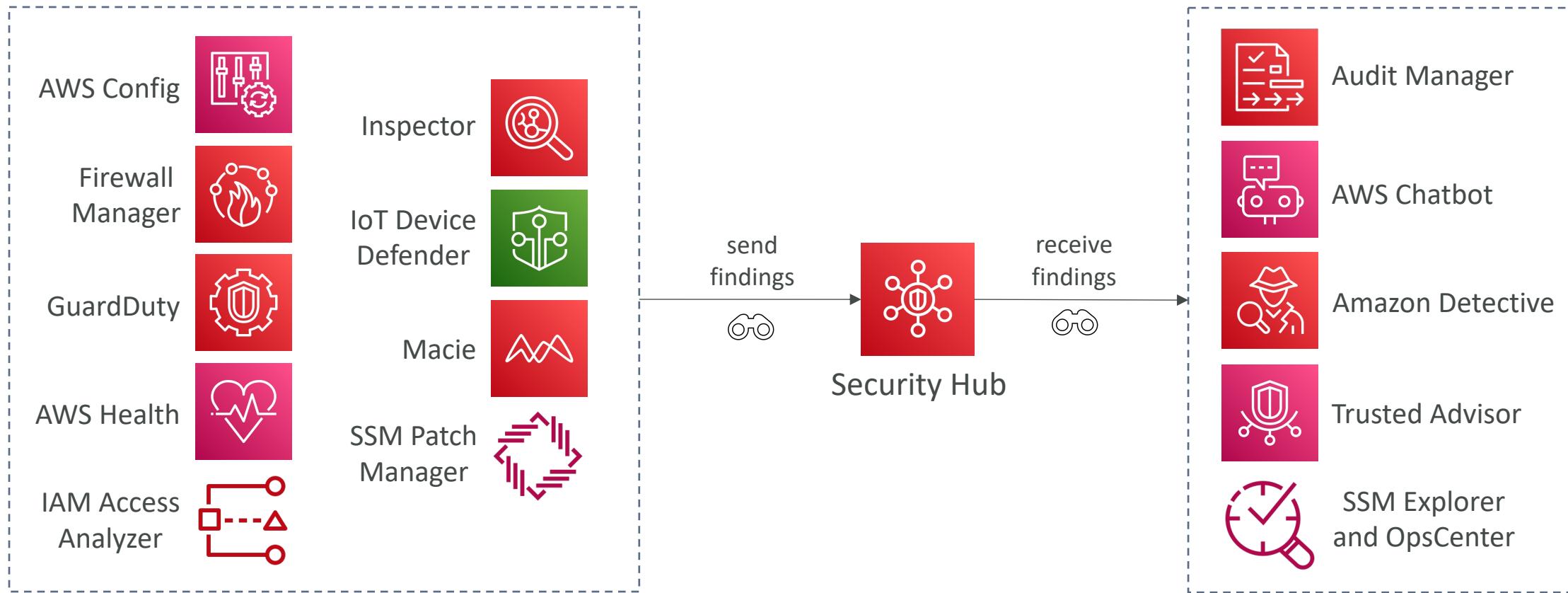
- When enabled, GuardDuty sends findings it generates to Security Hub
- Findings are sent in **AWS Security Finding Format (ASFF)**
- Findings are usually sent within 5 minutes
- Archiving a GuardDuty finding will NOT update the finding in Security Hub

GuardDuty Finding Type
UnauthorizedAccess:EC2/SSHBruteForce
Stealth:IAMUser/CloudTrailLoggingDisabled
Execution:EC2/SuspiciousFile



ASFF Finding Type
TTPs/Initial Access/UnauthorizedAccess:EC2-SSHBruteForce
TTPs/Defense Evasion/Stealth:IAMUser-CloudTrailLoggingDisabled
TTPs/Execution/Execution:EC2-SuspiciousFile

# Security Hub – Services Integration



# Security Hub – 3rd Party Integration

- Security Hub integrates with multiple 3<sup>rd</sup> partner products
- Send Findings to Security Hub



- Receive Findings from Security Hub



- Update Findings in Security Hub



**3CORESEC**

**3CORESec: 3CORESec NTA**

Description  
3CORESec Network Traffic Analyses and Network Intrusion Detection System.

Type of integration  
Sends findings to Security Hub

Categories  
Network Intrusion Detection Systems (IDS), Network Forensics

How to receive findings from this integration

1. Purchase a subscription to this product
2. Follow the integration's configuration instructions: [Configure](#)
3. Choose **Accept findings**

Status  
 Not accepting findings Accept findings



# Security Hub – Findings

- Security Hub consumes findings using **AWS Security Finding Format (ASFF)** format
- Security Hub automatically updates and deletes findings
- Findings past 90 days are automatically deleted
- Filter by Region, Integration, Security Standard, Insights

Company	Product ▾	Title ▾	Resource	Compliance Status	Updated at ▾
AWS	Security Hub	IAM.3 IAM users' access keys should be rotated every 90 days or less	IAM User <a href="#">stephane</a>	<span>✖ FAILED</span>	a minute ago
AWS	Security Hub	1.4 Ensure access keys are rotated every 90 days or less	IAM User <a href="#">stephane</a>	<span>✖ FAILED</span>	2 minutes ago
AWS	Security Hub	IAM.7 Password policies for IAM users should have strong configurations	Account <a href="#">387124123361</a>	<span>✖ FAILED</span>	2 minutes ago

**IAM.3 IAM users' access keys should be rotated every 90 days or less** X

Finding ID: [arn:aws:securityhub:eu-west-1:387124123361:subscription/aws-foundational-security-best-practices/v/1.0.0/IAM.3/finding/1ca2a163-03b6-455a-8b5c-cd422d5a991e](#)

■ MEDIUM

This AWS control checks whether the active access keys are rotated within 90 days.

Rule(s) ▾ ▼

**Workflow status**

<span style="color: green;">New</span>	▲
New	▼
Notified	⊕
Suppressed	⊕
Resolved	⊕

**Compliance status**

<span style="color: red;">✖ FAILED</span>	⊕
---	---

**RECORD STATE**

**ACTIVE**

Set by the finding provider

---

**Severity (original)**

40 ⊕

---

**Created at**

2022-12-20T08:48:12.067Z ⊕

---

**Updated at**

2022-12-20T08:48:12.067Z ⊕

---

**Product name**

Security Hub ⊕

# Security Hub – Insights

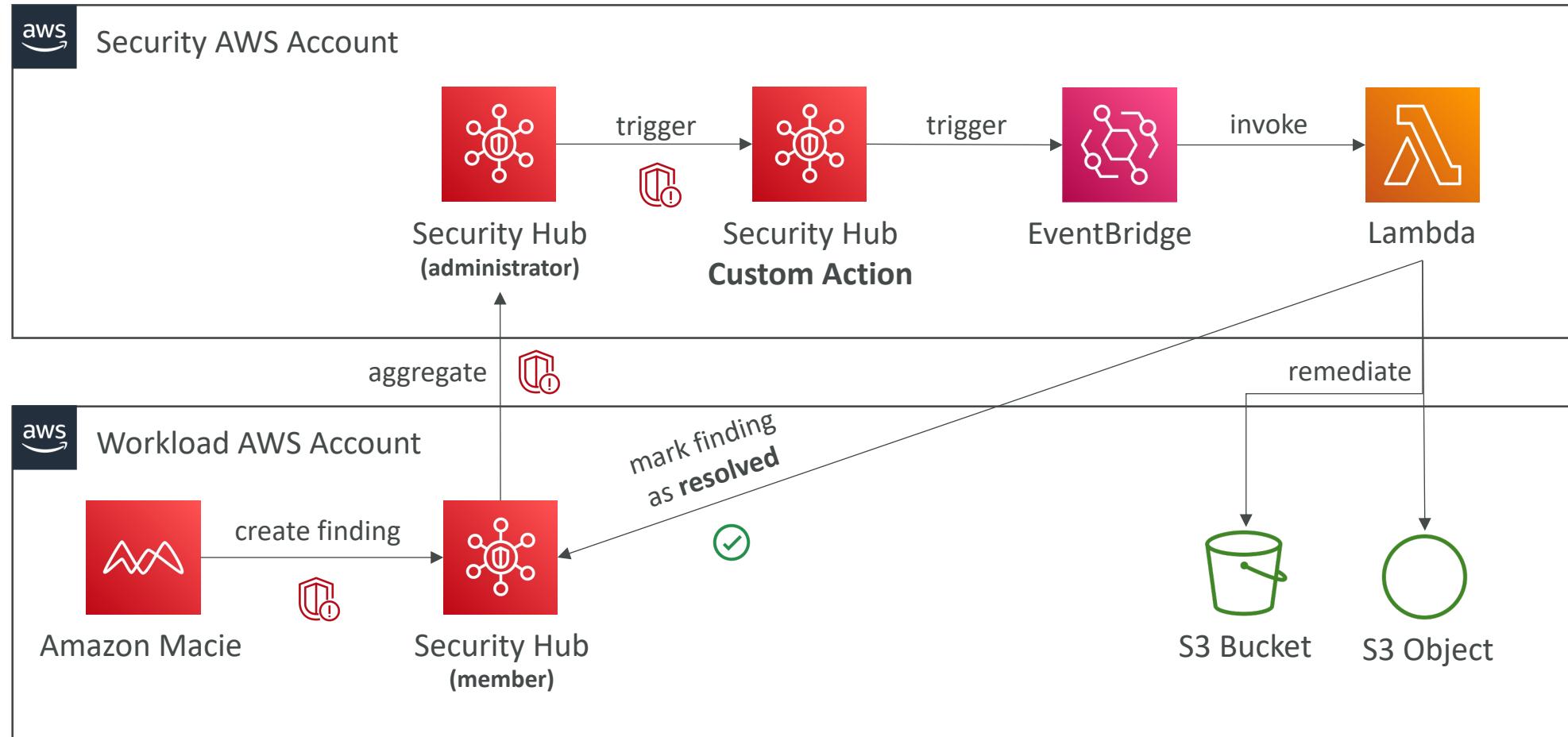
- Collection of related findings that identifies a security area that requires attention and intervention
- Example: Insight points out EC2 instances that are subject of findings that detect poor security practices
- Brings findings from across finding providers
- Each Insight defined by a Group By statement and optional Filters
- **Built-In Managed Insights** – return results only if you enabled related product integration or security standard (can not edit or delete)
- **Custom Insights** – Track issues specific to your environment
  - Example: Custom Insight to track critical findings affecting member accounts

# Security Hub – Custom Actions

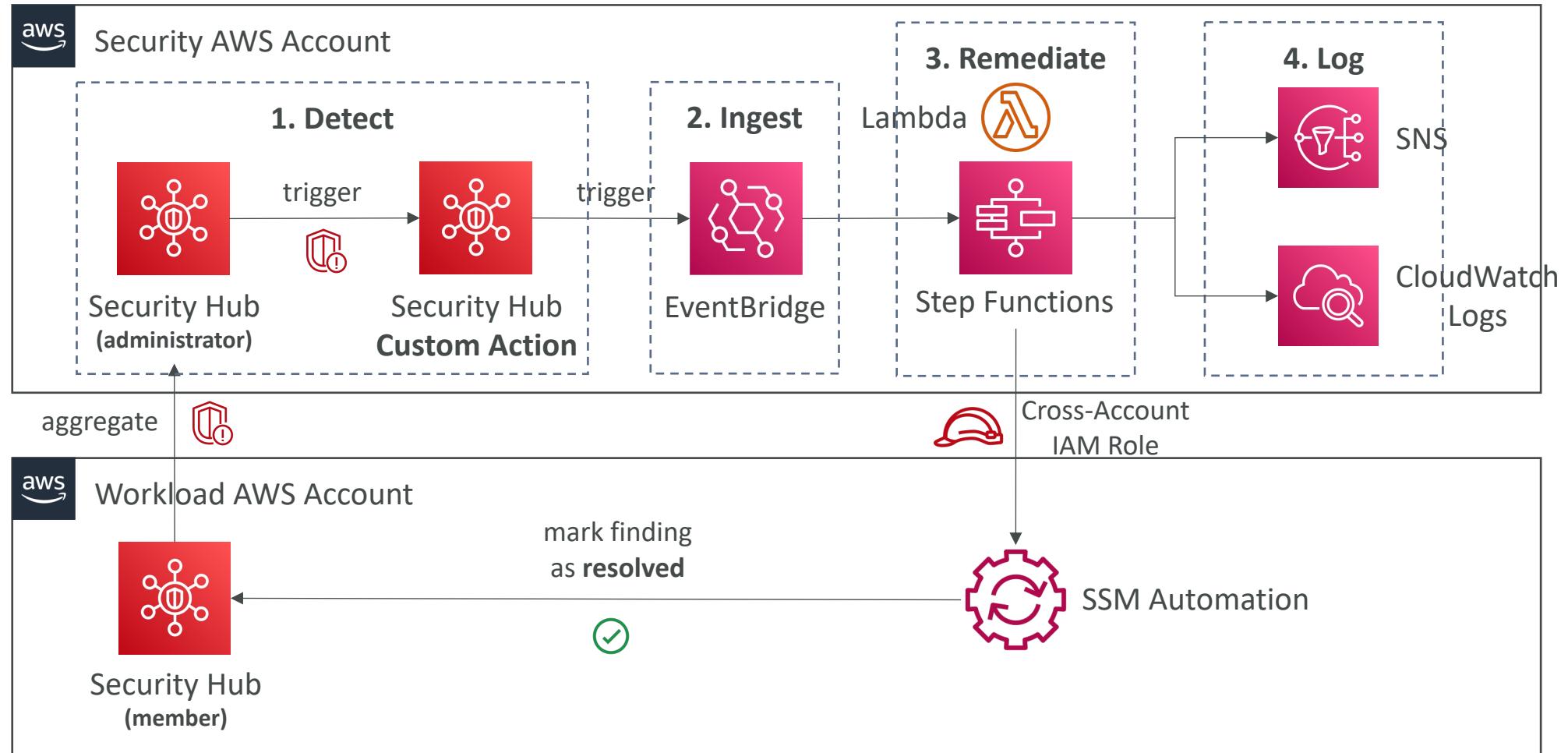
- Helps you automate Security Hub with EventBridge
- Allows you to create actions for response and remediation to selected findings within the Security Hub Console
- EventBridge event type is **Security Hub Findings - Custom Action**

Custom actions			<a href="#">Delete</a>	<a href="#">Create custom action</a>
Action name	Description	Custom action ARN		
<a href="#">Foobar</a>	test	arn:aws:securityhub:eu-west-1:387124123361:action/custom/test	<a href="#">Update</a>	

# Security Hub – Custom Actions – Architecture



# Security Hub – Custom Actions – Architecture



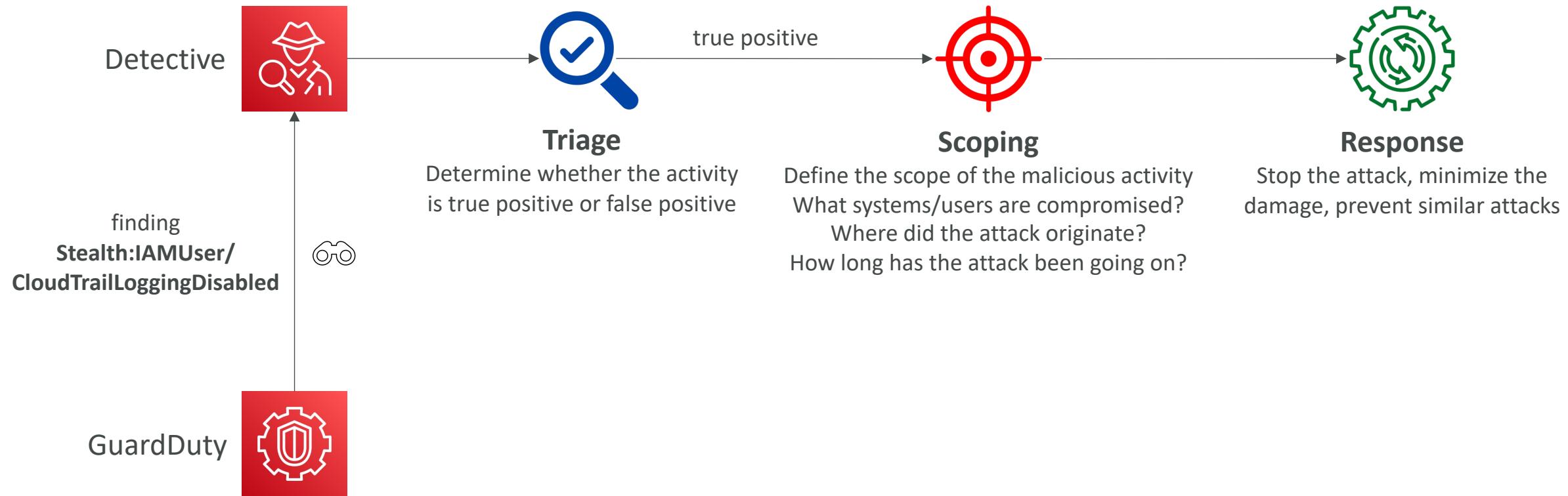
# Amazon Detective



- GuardDuty, Macie, and Security Hub are used to identify potential security issues, or findings
- Sometimes security findings require deeper analysis to isolate the root cause and take action – it's a complex process
- Amazon Detective analyzes, investigates, and quickly identifies the root cause of security issues or suspicious activities using ML and graphs
- Automatically collects and processes events from VPC Flow Logs, CloudTrail, and GuardDuty to create a unified view
- Produces visualizations with details and context to get to the root cause

# Amazon Detective – Architectures

## Stealth:IAMUser/CloudTrailLoggingDisabled





# Penetration Testing on AWS Cloud

- AWS customers are welcome to carry out security assessments or penetration tests against their AWS infrastructure **without prior approval for 8 services:**
  - Amazon EC2 instances, NAT Gateways, and Elastic Load Balancers
  - Amazon RDS
  - Amazon CloudFront
  - Amazon Aurora
  - Amazon API Gateways
  - AWS Lambda and Lambda Edge functions
  - Amazon Lightsail resources
  - Amazon Elastic Beanstalk environments
- List can increase over time (you won't be tested on that at the exam)

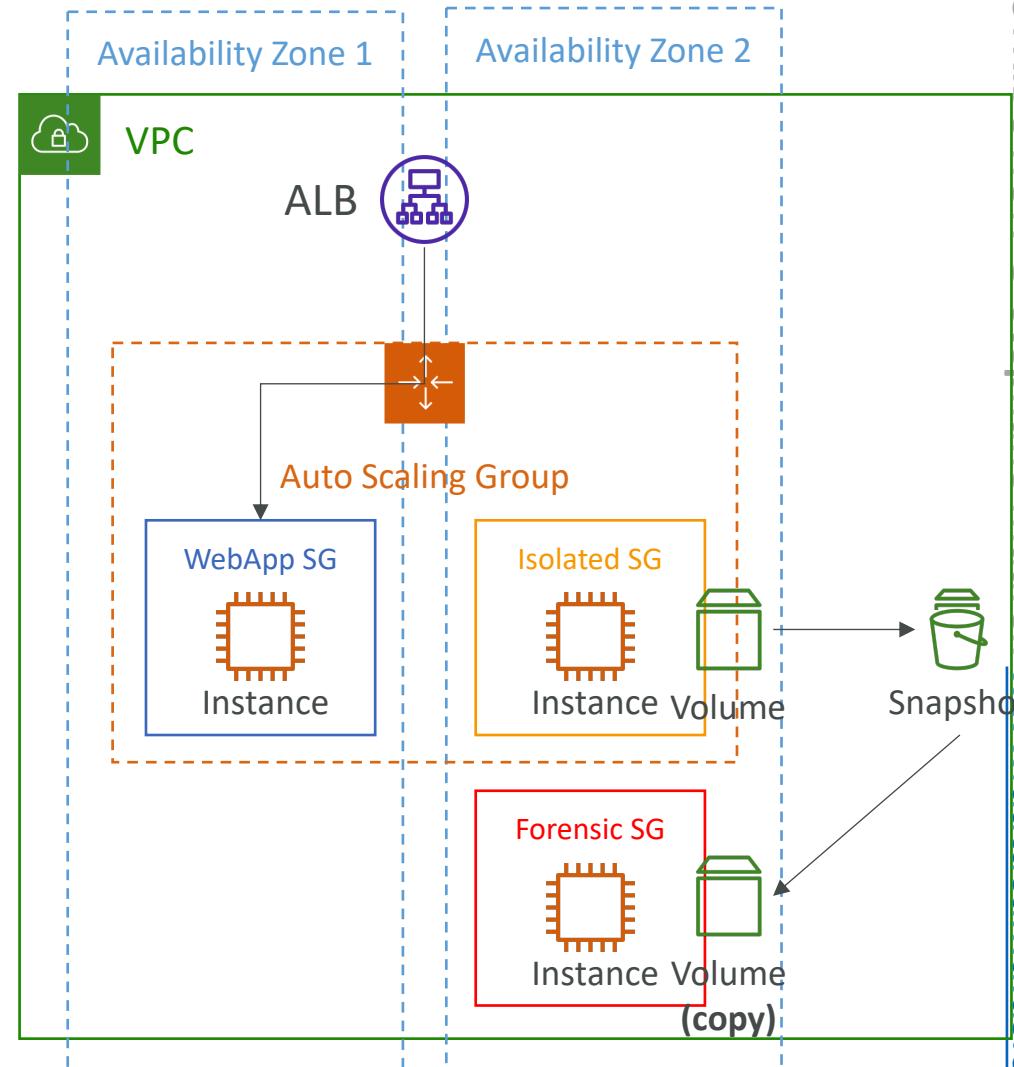
# Penetration Testing on your AWS Cloud



- Prohibited Activities
  - DNS zone walking via Amazon Route 53 Hosted Zones
  - Denial of Service (DoS), Distributed Denial of Service (DDoS), Simulated DoS, Simulated DDoS
  - Port flooding
  - Protocol flooding
  - Request flooding (login request flooding, API request flooding)
- For any other simulated events, contact [aws-security-simulated-event@amazon.com](mailto:aws-security-simulated-event@amazon.com)
- Read more: <https://aws.amazon.com/security/penetration-testing/>

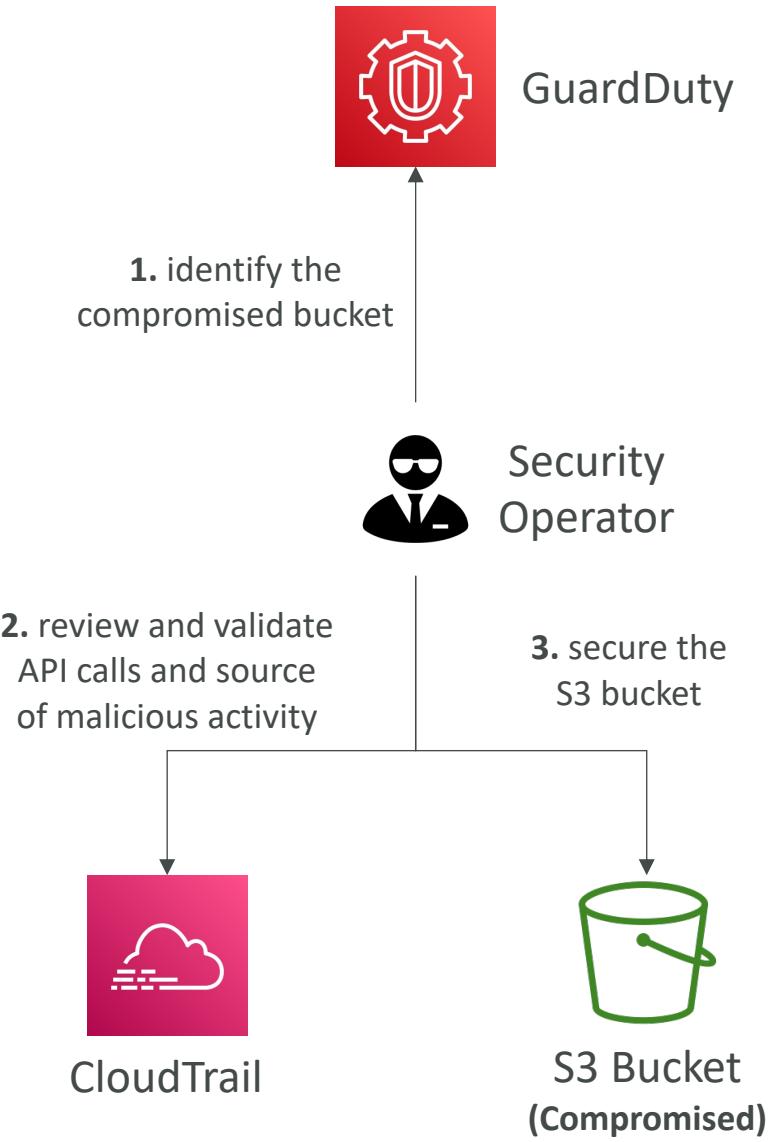
# Compromised EC2 Instance

- Steps to address compromised instances:
  - Capture the instance's metadata
  - Enable Termination Protection
  - Isolate the instance (replace instance's SG – no outbound traffic authorized)
  - Detach the instance from any ASG (Suspend processes)
  - Deregister the instance from any ELB
  - Snapshot the EBS volumes (deep analysis)
  - Tag the EC2 instance (e.g., investigation ticket)
- Offline investigation: shutdown instance
- Online investigation (e.g., snapshot memory or capture network traffic)
- Automate the isolation process: Lambda
- Automate memory capture: SSM Run Command



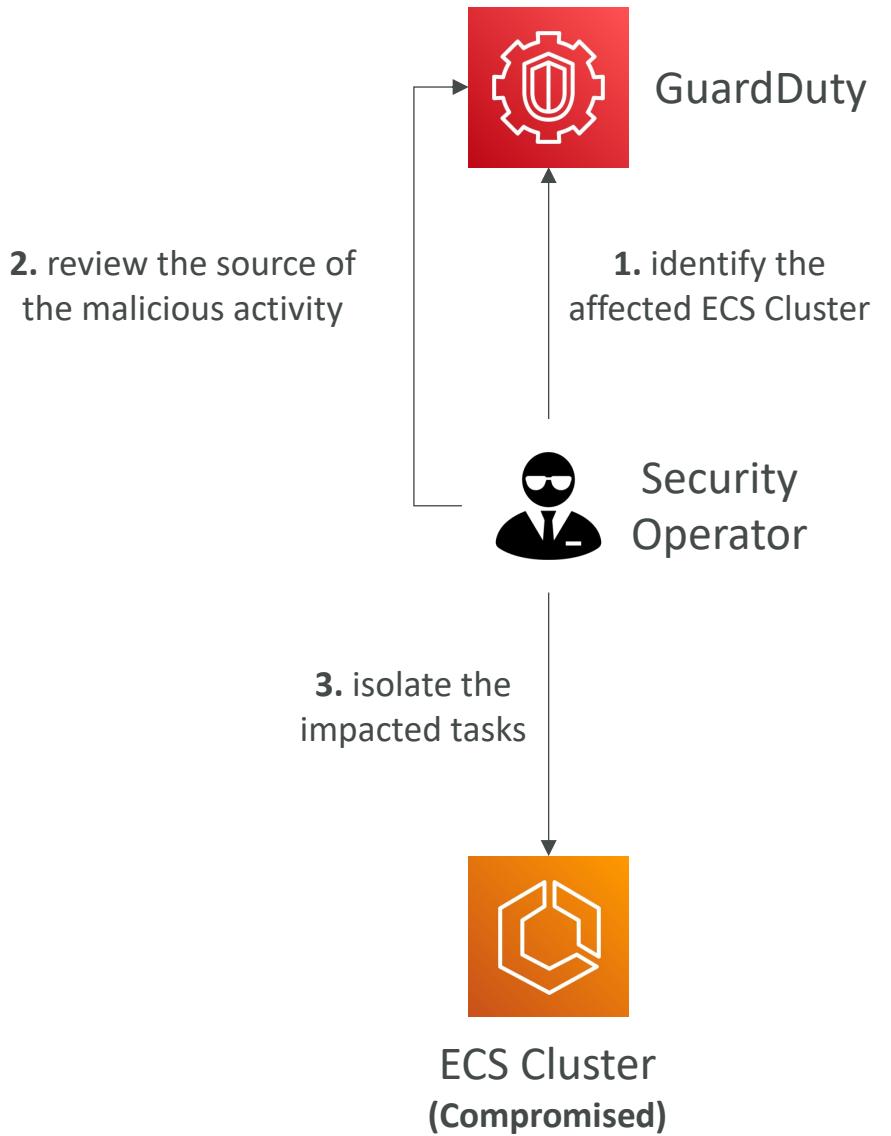
# Compromised S3 Bucket

- Identify the compromised S3 bucket using **GuardDuty**
- Identify the source of the malicious activity (e.g., IAM user, role) and the API calls using **CloudTrail** or **Amazon Detective**
- Identify whether the source was authorized to make those API calls
- Secure your S3 bucket, recommended settings:
  - S3 Block Public Access Settings, S3 Bucket Policies and User Policies, VPC Endpoints for S3, S3 Pre-signed URLs, S3 Access Points, S3 ACLs



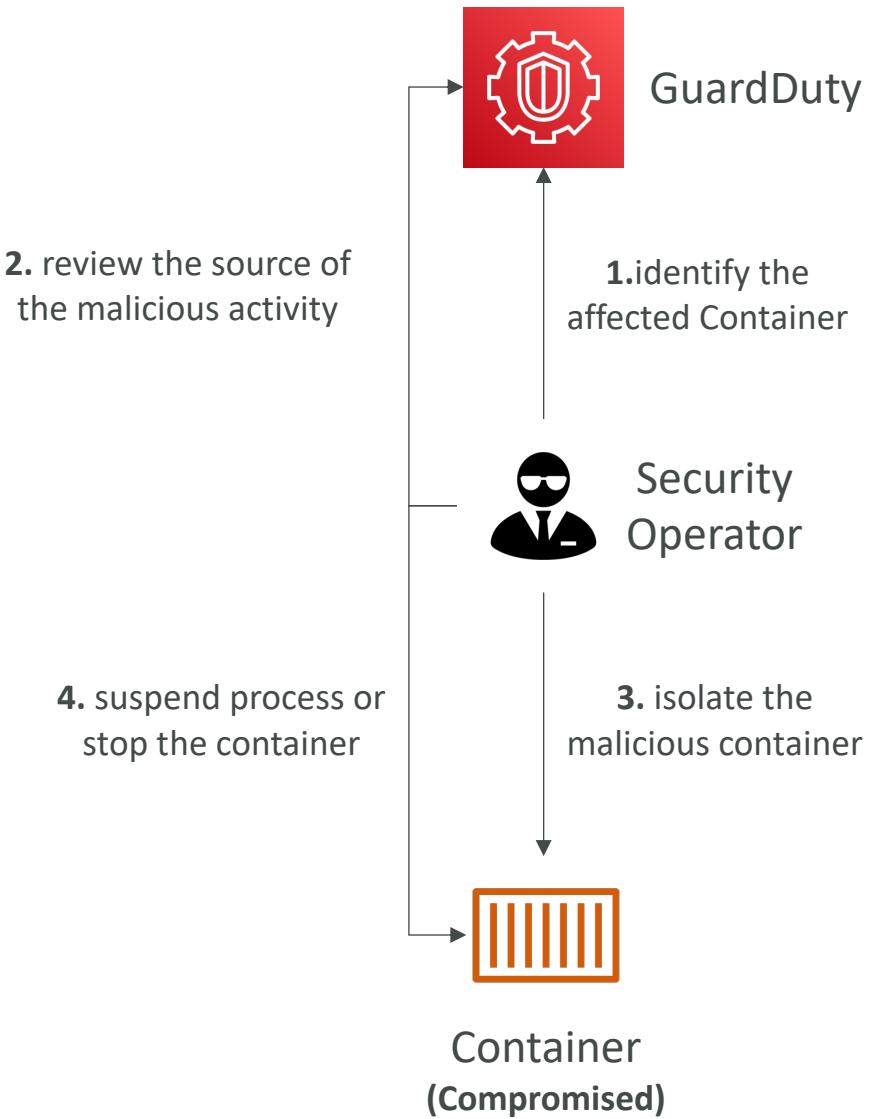
# Compromised ECS Cluster

- Identify the affected ECS Cluster using **GuardDuty**
- Identify the source of the malicious activity (e.g., container image, tasks)
- Isolate the impacted tasks (deny all ingress/egress traffic to the task using security groups)
- Evaluate the presence of malicious activity (e.g., malware)



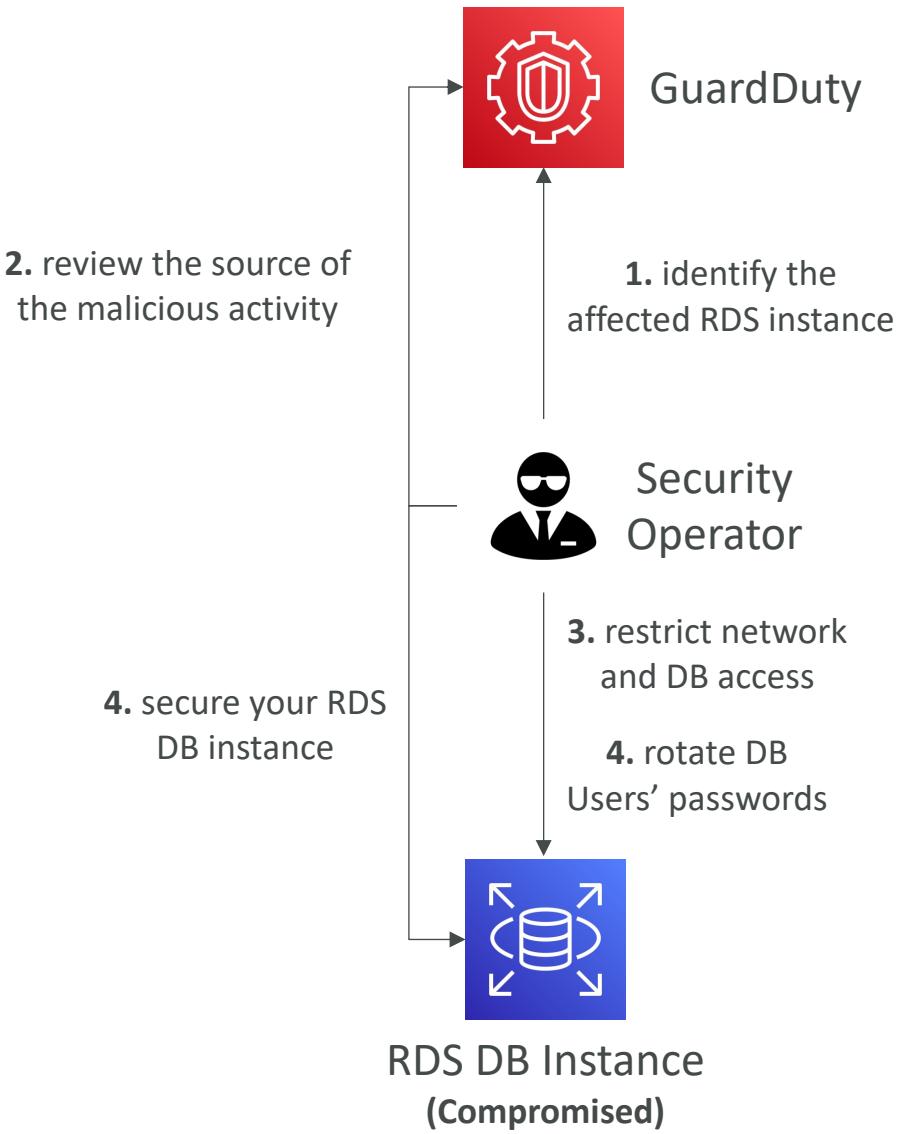
# Compromised Standalone Container

- Identify the malicious container using GuardDuty
- Isolate the malicious container (deny all ingress/egress traffic to the container)
- Suspend all process in the container (pause the container)
- Or stop the container and look at EBS Snapshots retained by GuardDuty (snapshots retention feature)
- Evaluate the presence of malicious activity (e.g., malware)



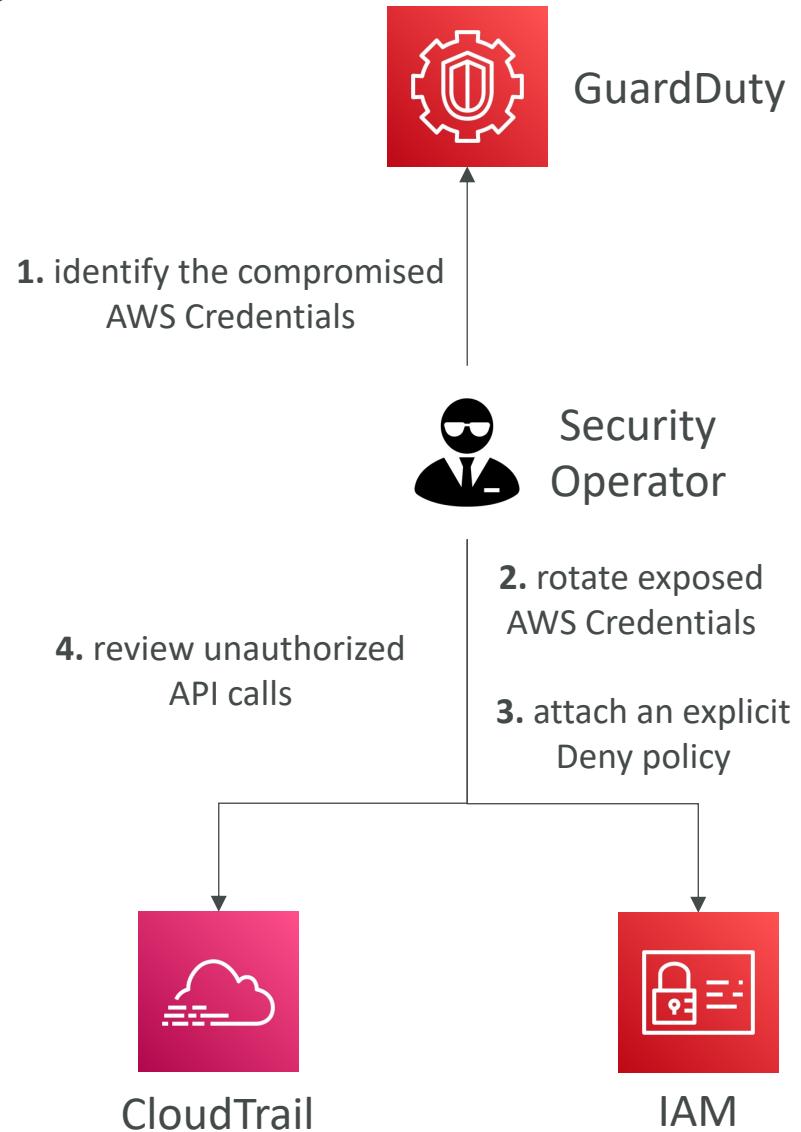
# Compromised RDS Database Instance

- Identify the affected DB instance and DB user using **GuardDuty**
- If it is NOT legitimate behavior:
  - Restrict network access (SGs & NALCs)
  - Restrict the DB access for the suspected DB user
- Rotate the suspected DB users' passwords
- Review DB Audit Logs to identify leaked data
- Secure your RDS DB instance, recommended settings:
  - Use Secrets Manager to rotate the DB password
  - Use IAM DB Authentication to manage DB users' access without passwords



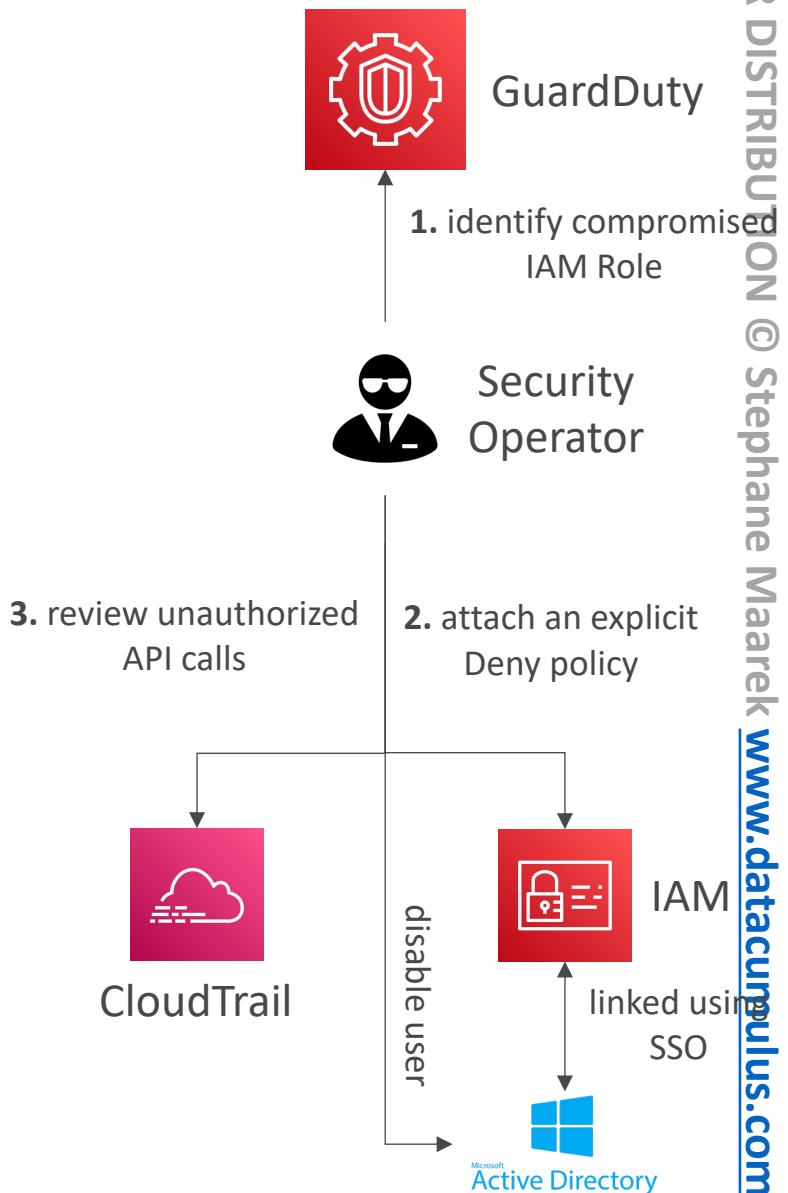
# Compromised AWS Credentials

- Identify the affected IAM user using **GuardDuty**
- Rotate the exposed AWS Credentials
- Invalidate temporary credentials by **attaching an explicit Deny policy** to the affected IAM user with an STS date condition (see IAM section)
- Check **CloudTrail logs** for other unauthorized activity
- Review your AWS resources (e.g., delete unauthorized resources)
- Verify your AWS account information



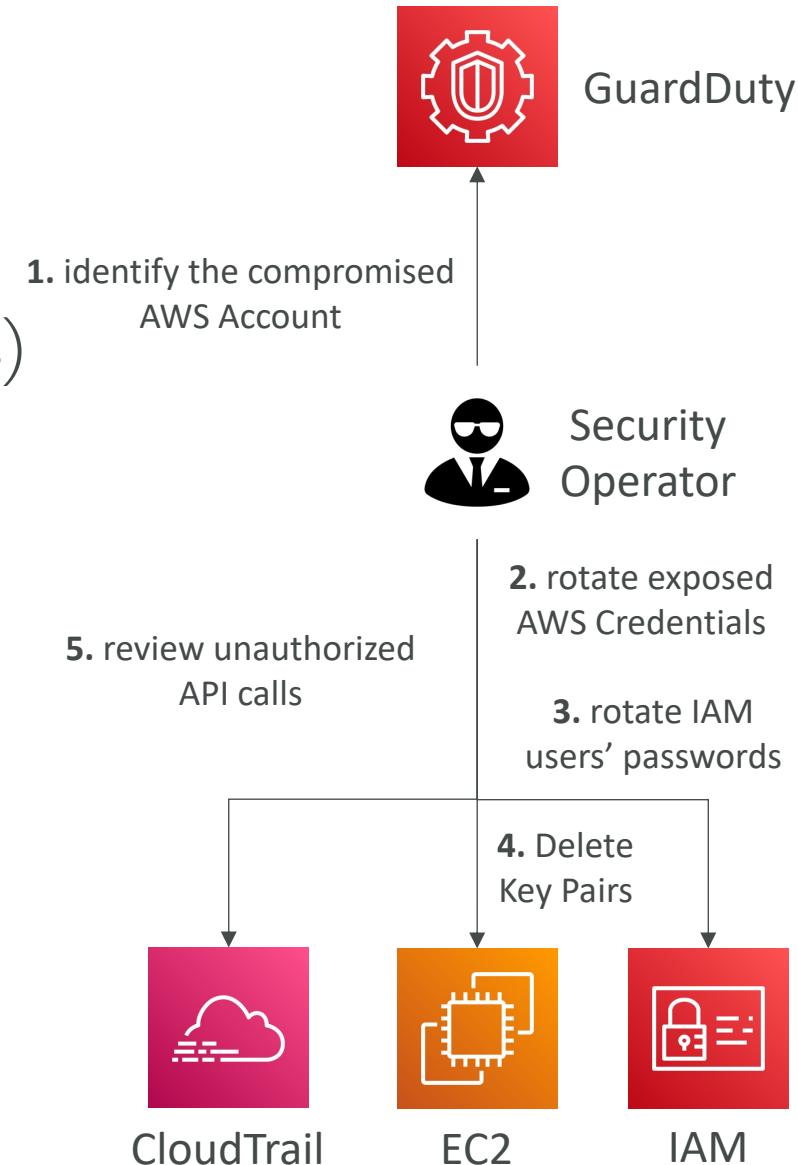
# Compromised IAM Role

- Invalidate temporary credentials by **attaching an explicit Deny policy** to the affected IAM user with an STS date condition (see IAM section)
- Revoke access for the identity to the linked AD if any
- Check **CloudTrail logs** for other unauthorized activity
- Review your AWS resources (e.g., delete unauthorized resources)
- Verify your AWS account information

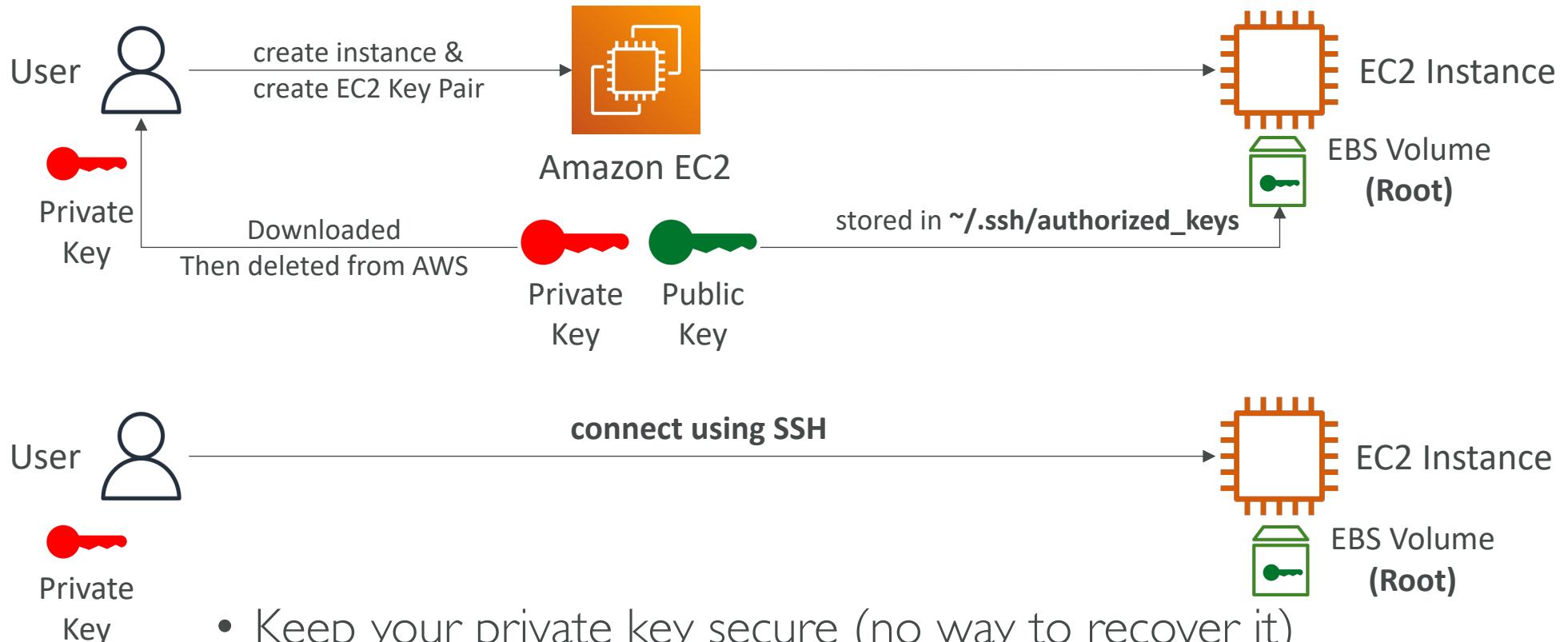


# Compromised Account

- Rotate and delete exposed AWS Access Keys
- Rotate and delete any unauthorized IAM user credentials (rotate existing IAM users' passwords)
- Rotate and delete all EC2 Key Pairs
- Check CloudTrail logs for other unauthorized activity
- Review your AWS resources (e.g., delete unauthorized resources)
- Verify your AWS account information



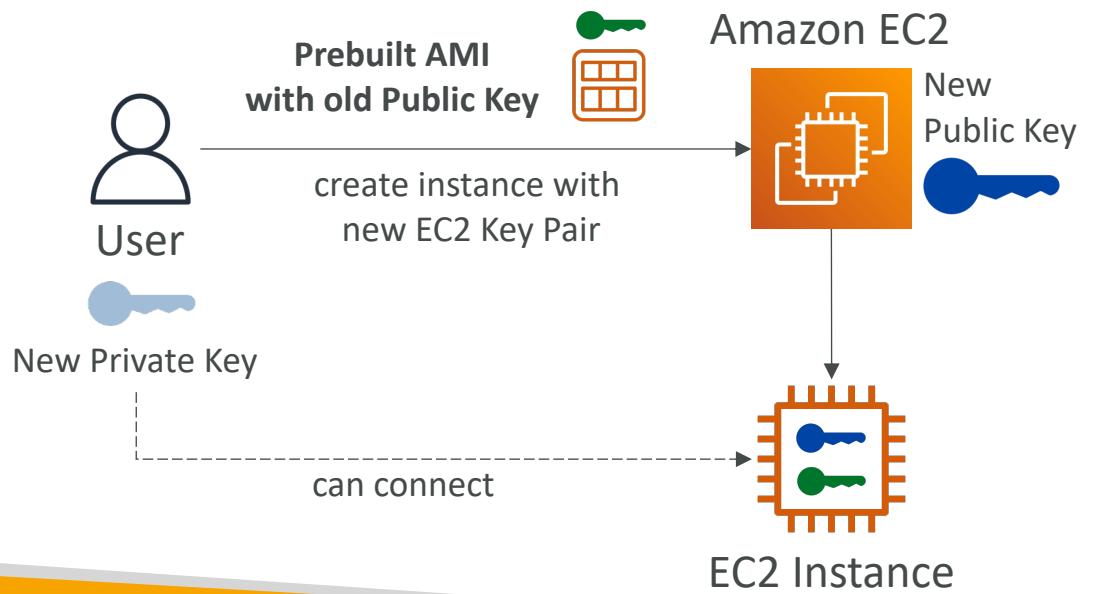
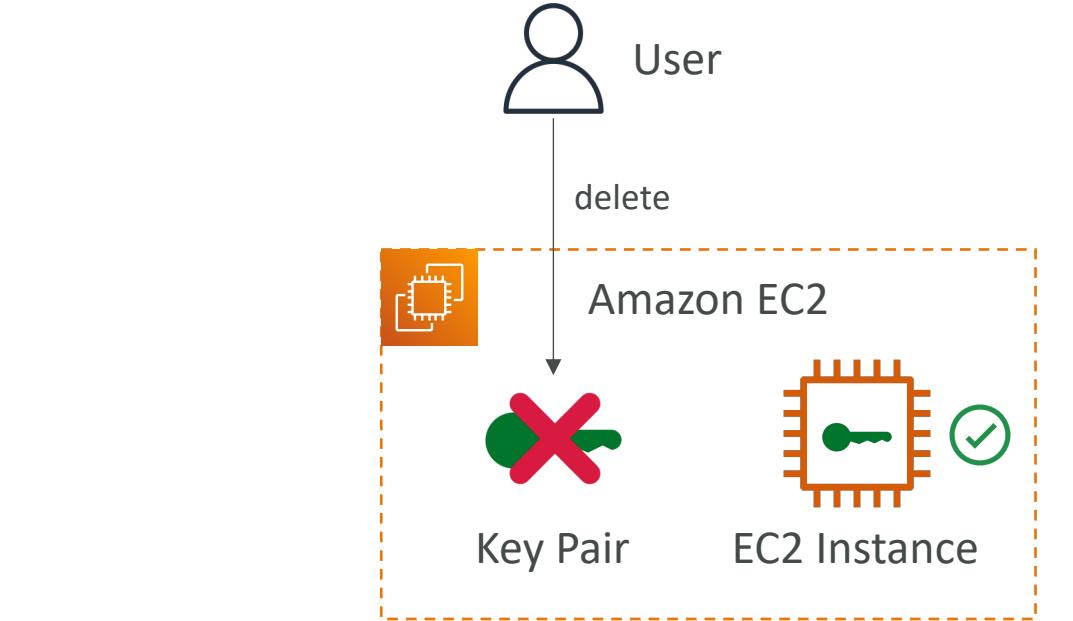
# EC2 Key Pairs - Explained



- Keep your private key secure (no way to recover it)
- You can create Key Pairs outside of AWS and upload them
- Both ED25519 and 2048-bit SSH-2 RSA keys are supported

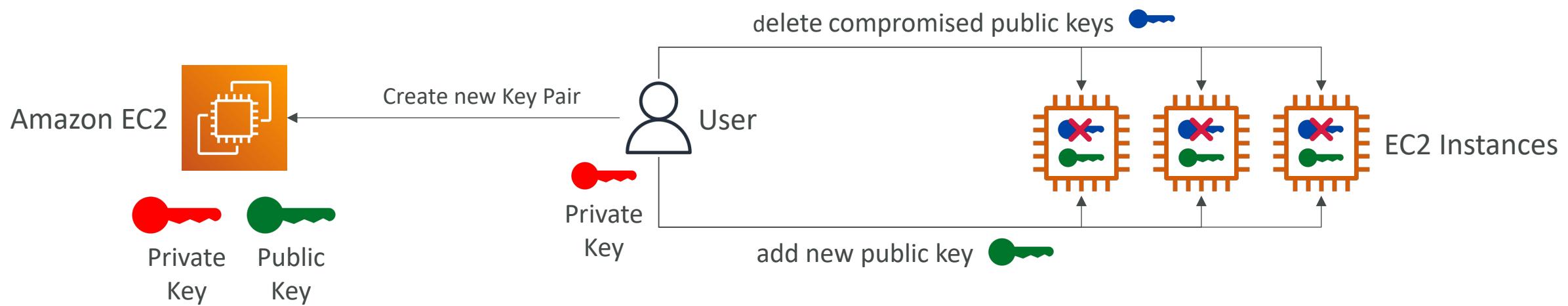
# EC2 Key Pairs – Notes

- Key Pairs don't get deleted from EC2 instance's root volumes when the Key Pair removed from the EC2 Console
- Launching an EC2 instance with prebuilt AMI, the old Key Pair will exist alongside with the new Key Pair



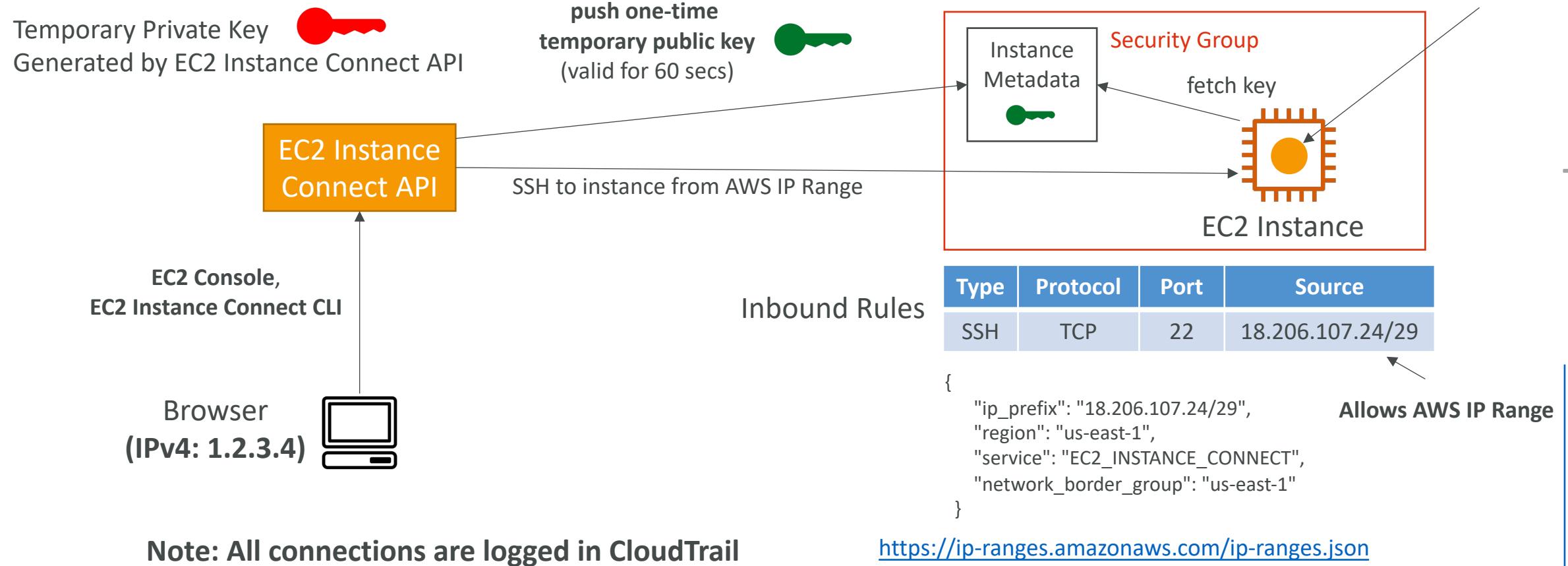
# Remediating Exposed EC2 Key Pairs

- Remove all the public keys in `~/.ssh/authorized_keys` file on EC2 instances
- Create a new Key Pair and add its public key to the `~/.ssh/authorized_keys` file on all EC2 instances
- Note: Use SSM Run Command to automate the process add/delete public keys on EC2 instances



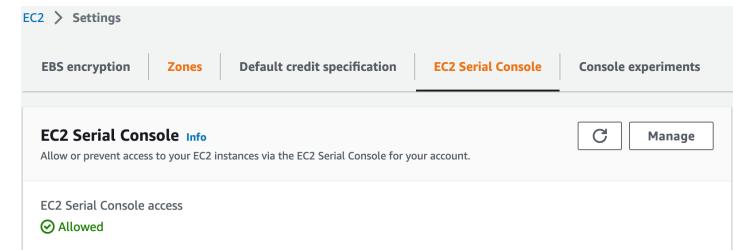
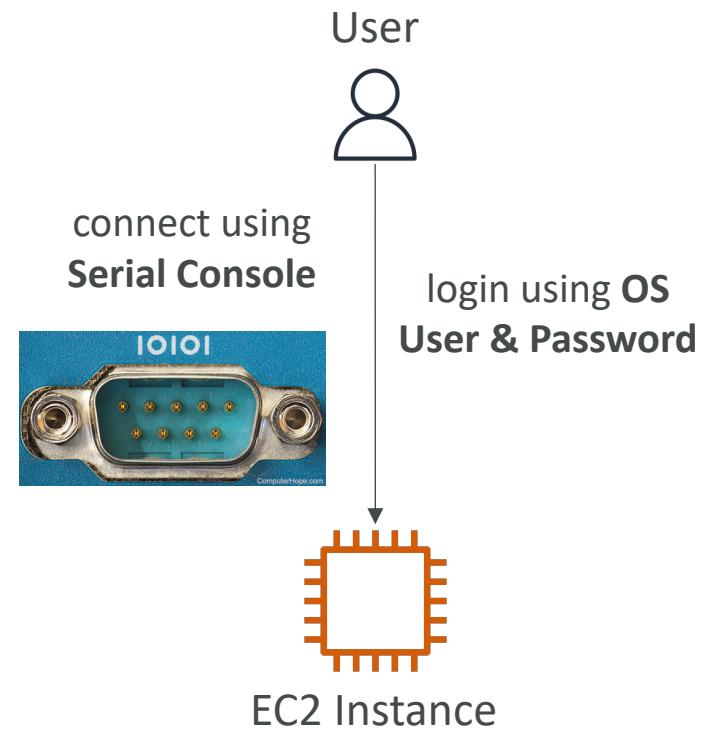
# EC2 Instance Connect

## Browser Based SSH - Explanation



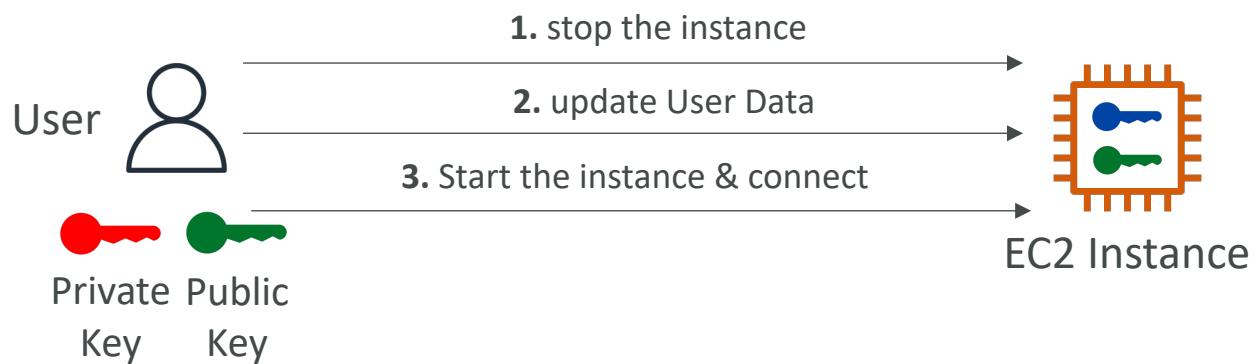
# EC2 Serial Console – Explanation

- Use cases: troubleshoot boot, troubleshoot network configuration, analyze reboot issues...
- Directly access your EC2 instance's serial port (as if keyboard and monitor are directly attached to the EC2 instance)
- Does NOT require any network capabilities
- Use with supported Nitro-based EC2 instances
- **Must setup OS User and Password**
- Only one active session per EC2 instance
- Disabled by default (enable at AWS account level)



# Connect to Linux EC2 Instance with a Lost SSH Key Pair – Using EC2 User Data

- Create a new Key Pair; then copy the public key
- Stop the instance, update the EC2 User Data (cloud-config format)
- Start the instance and connect with the private key
- Delete the EC2 User Data



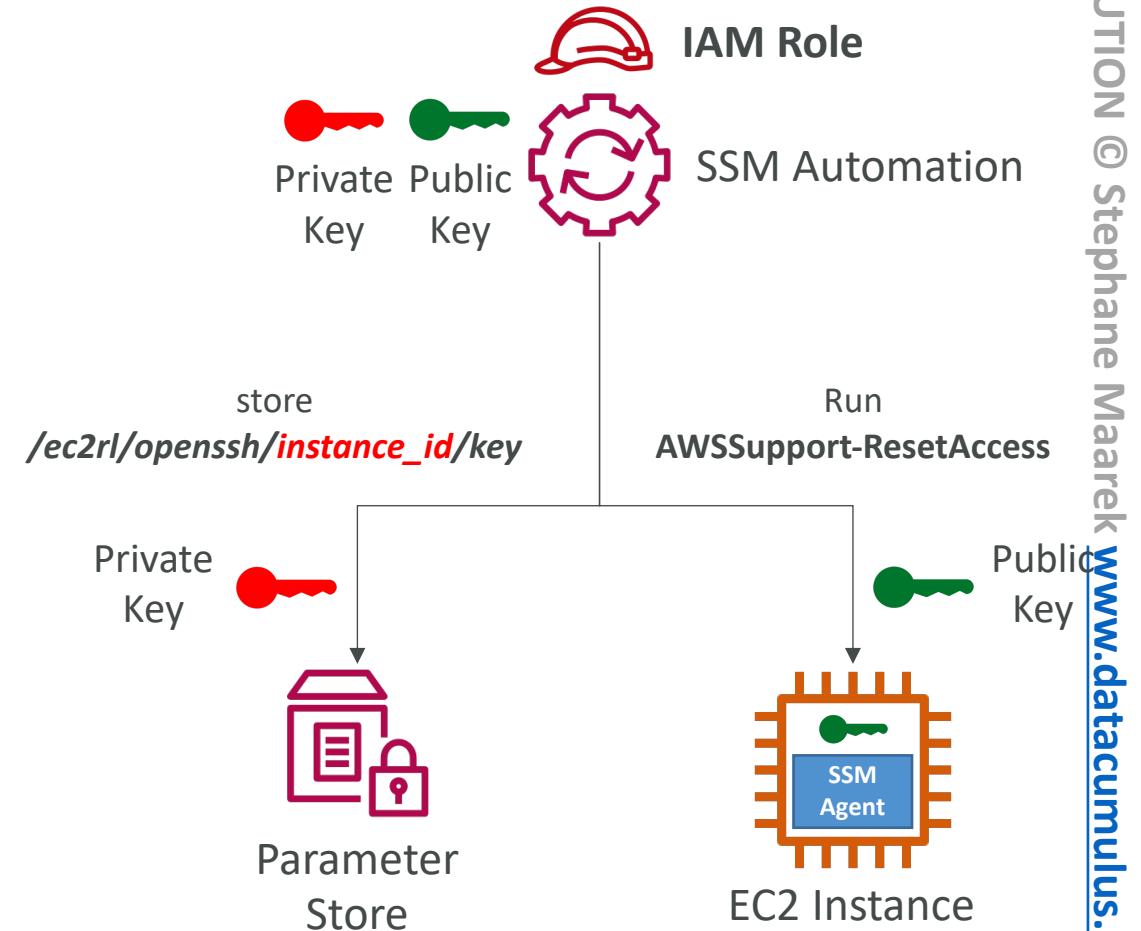
**Note:** Doesn't override the existing public keys

## EC2 User Data

```
Content-Type: multipart/mixed; boundary="--/"  
MIME-version: 1.0  
  
--/  
Content-Type: text/cloud-config; charset="us-ascii"  
MIME-Version: 1.0  
Content-Transfer-Encoding: 7bit  
Content-Disposition: attachment; filename="cloud-config.txt"  
  
#cloud-config  
cloud_final_modules:  
- [users-groups, once]  
users:  
- name: ec2-user  
  ssh-authorized-keys:  
  - ssh-rsa AAAAB3NzaC1yc2EAAAQABAAAAgQC3kfTYXvPi2+...
```

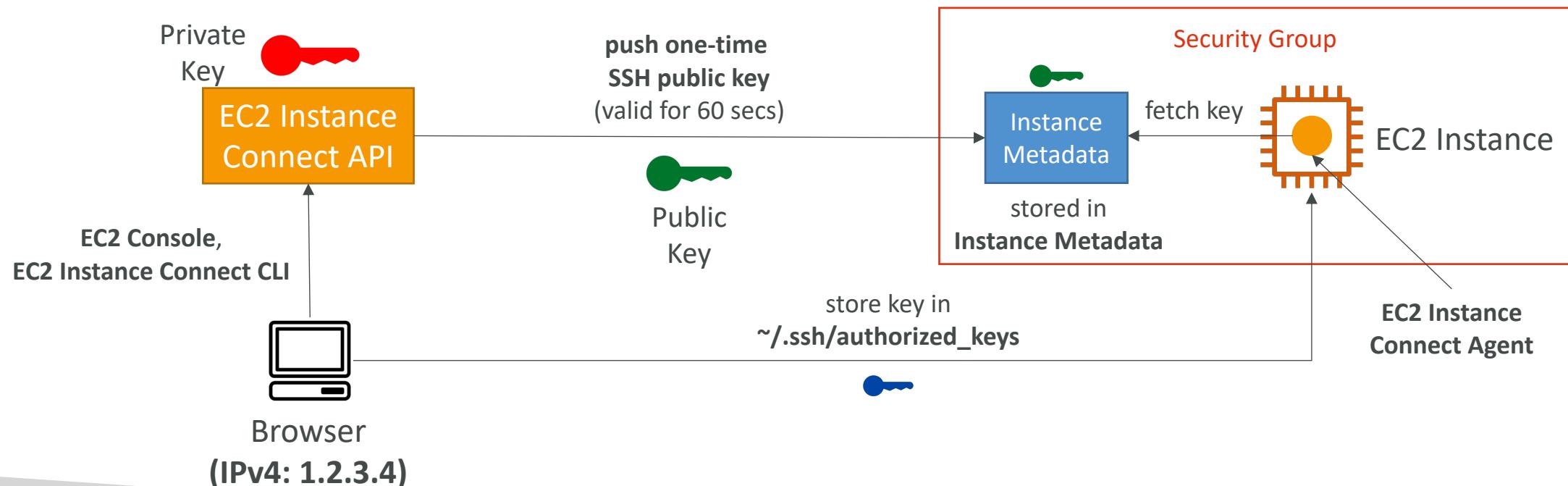
# Connect to Linux EC2 Instance with a Lost SSH Key Pair – Using Systems Manager

- Use `AWSSupport-ResetAccess` Automation Document
- Will create and apply a new key pair
- Works for both Linux and Windows
- The private key stored encrypted in SSM Parameter Store  
`/ec2rl/openssh/instance_id/key`
- Must have SSM Agent installed



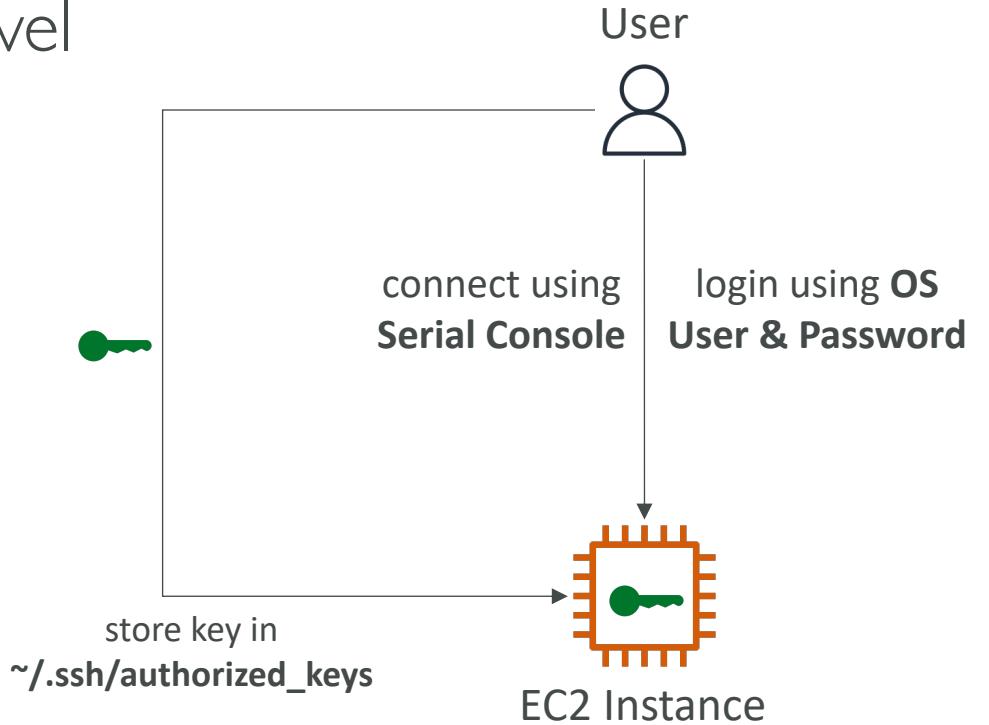
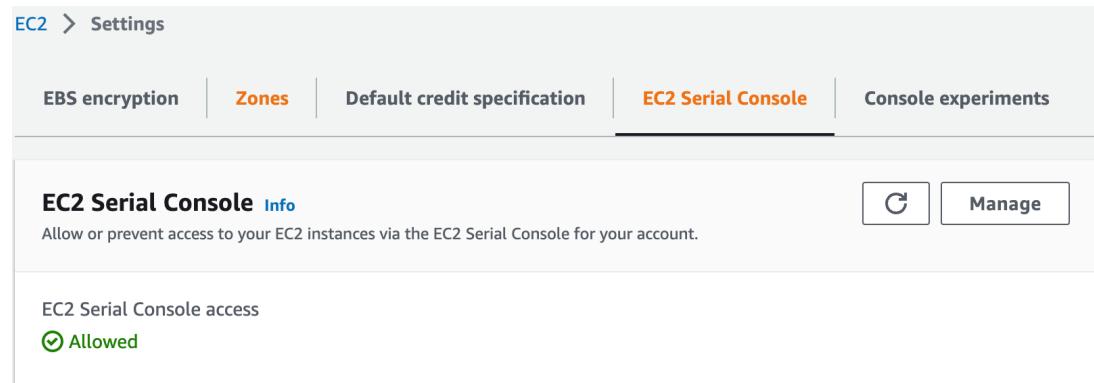
# Connect to Linux EC2 Instance with a Lost SSH Key Pair – Using EC2 Instance Connect

- EC2 Instance Connect agent must be installed (already installed on Amazon Linux 2 and Ubuntu 16.04 or later)
- Connect using EC2 Instance Connect temporary session
- Store permanent new public SSH key into `~/.ssh/authorized_keys`



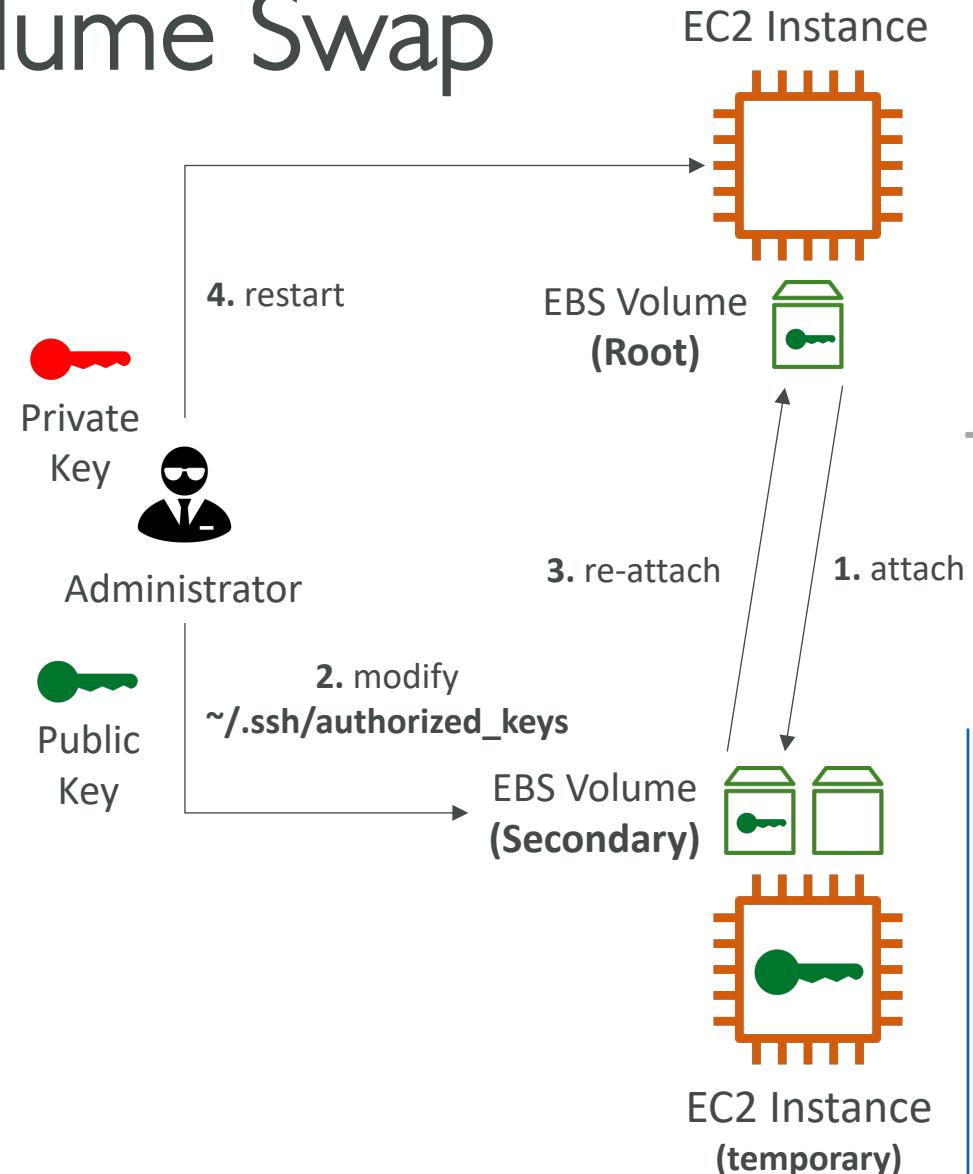
# Connect to Linux EC2 Instance with a Lost SSH Key Pair – Using EC2 Serial Console

- Connect to your instance without a working network connection
- Used with supported Nitro-based EC2 instances
- Must be enabled at the AWS account level



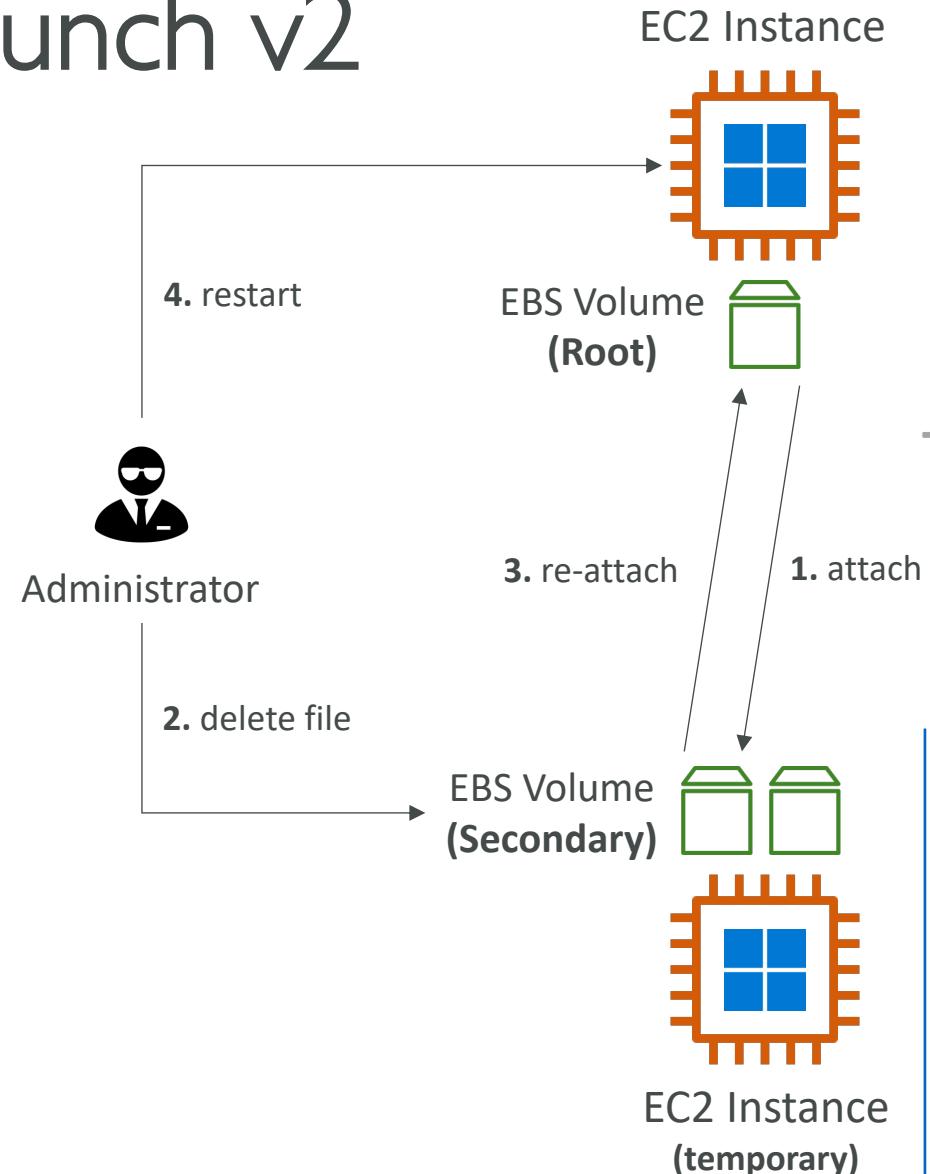
# Connect to Linux EC2 Instance with a Lost SSH Key Pair – Using EBS Volume Swap

- Create a new Key Pair
- Stop the original EC2 instance
- Detach the EBS root volume
- Attach the EBS volume to a temporary EC2 instance as a secondary volume
- Add the new public key to `~/.ssh/authorized_keys` on the volume
- Re-attach the volume to the original instance, then restart the instance



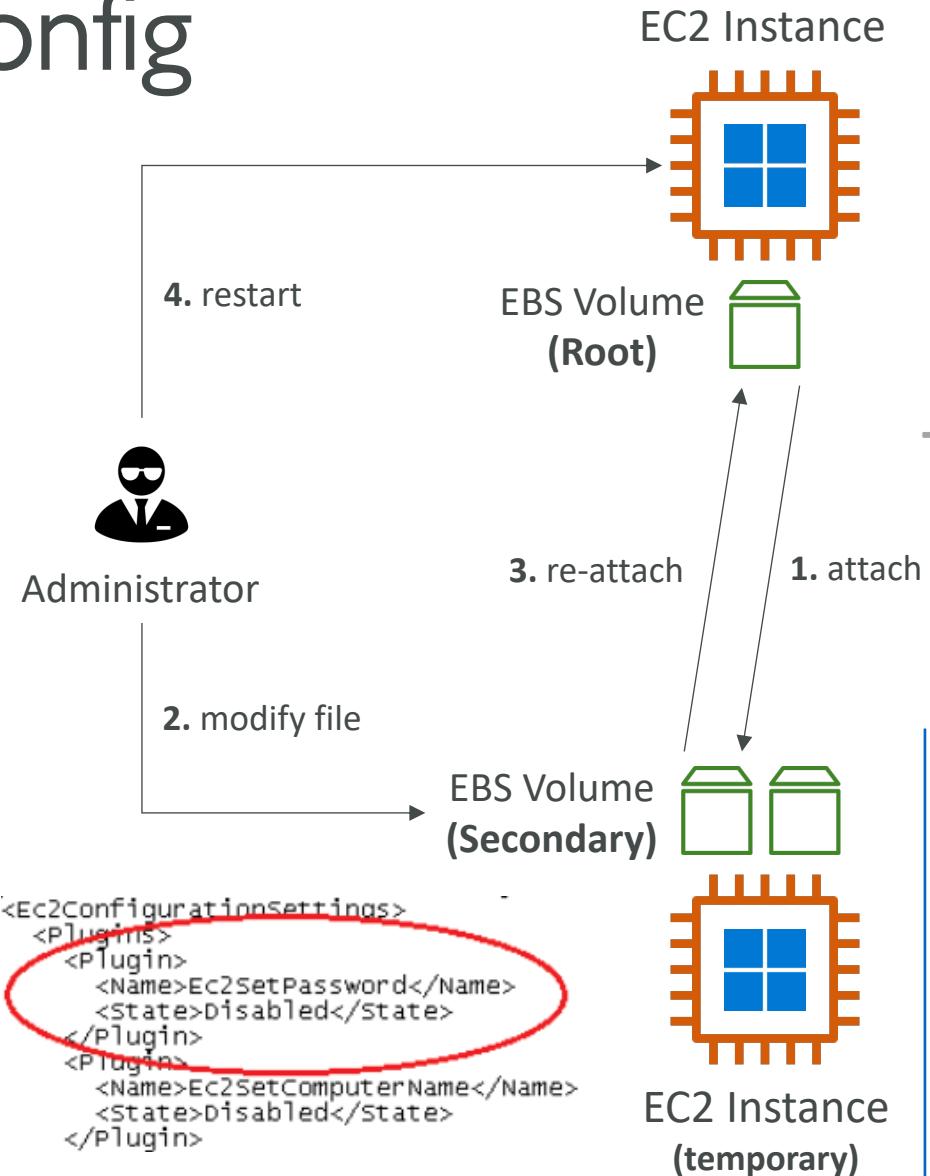
# Connect to Windows EC2 Instance with a Lost Password – Using EC2Launch v2

- Verify EC2Launch v2 service is running (Windows AMIs with the EC2Launch v2 service)
- Detach the EBS root volume
- Attach the volume to a temporary instance as a secondary volume
- Delete file  
`%ProgramData%\\Amazon\\EC2Launch\\state\\run-once`
- Re-attach the volume to the original instance, then restart the instance, you will be able to set a new password



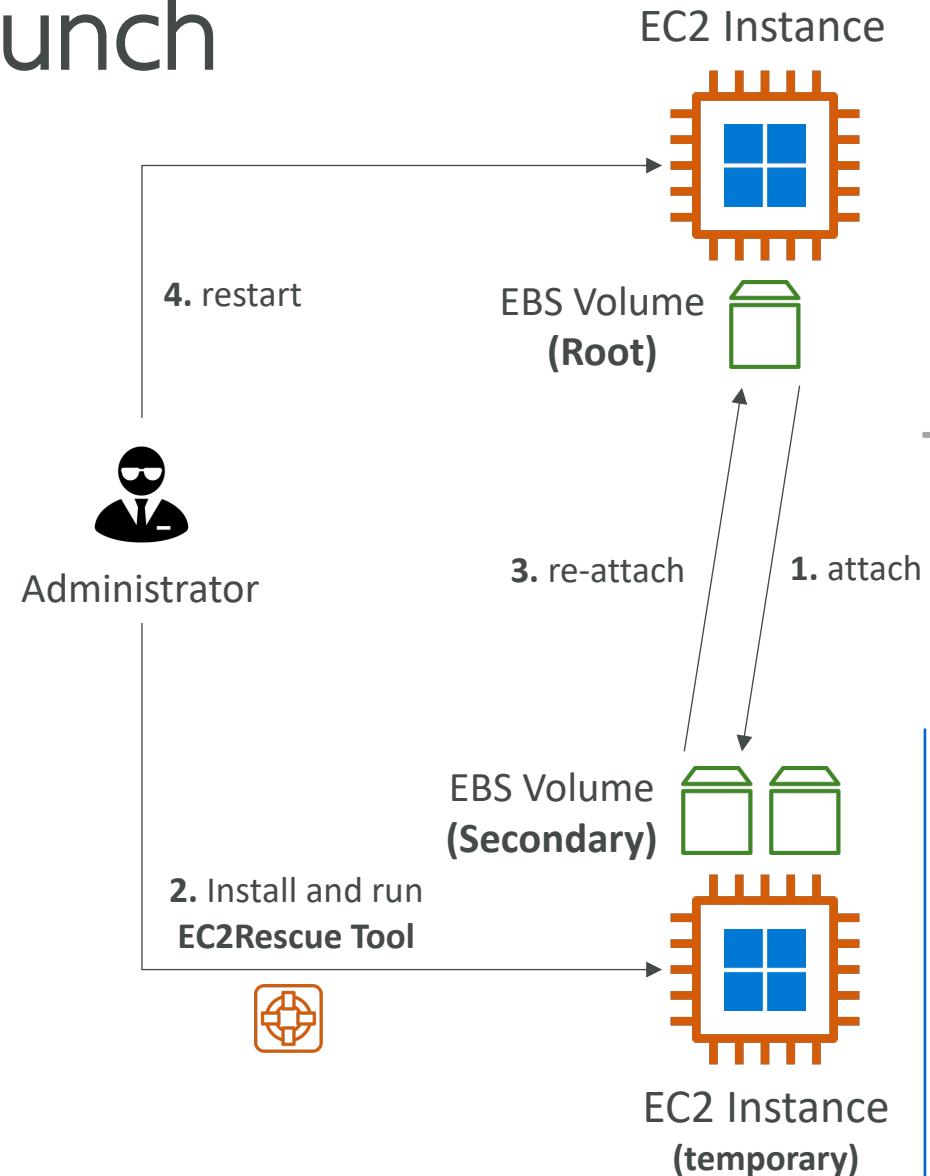
# Connect to Windows EC2 Instance with a Lost Password – Using EC2Config

- Verify EC2Config service is running
- Windows AMIs before Windows Server 2016
- Detach the EBS root volume
- Attach the volume to a temporary instance as a secondary volume
- Modify file `\Program Files\Amazon\Ec2ConfigService\Settings\config.xml`
- Set `EC2SetPassword` to *Enabled*
- Reattach the volume to the original instance, then restart the instance



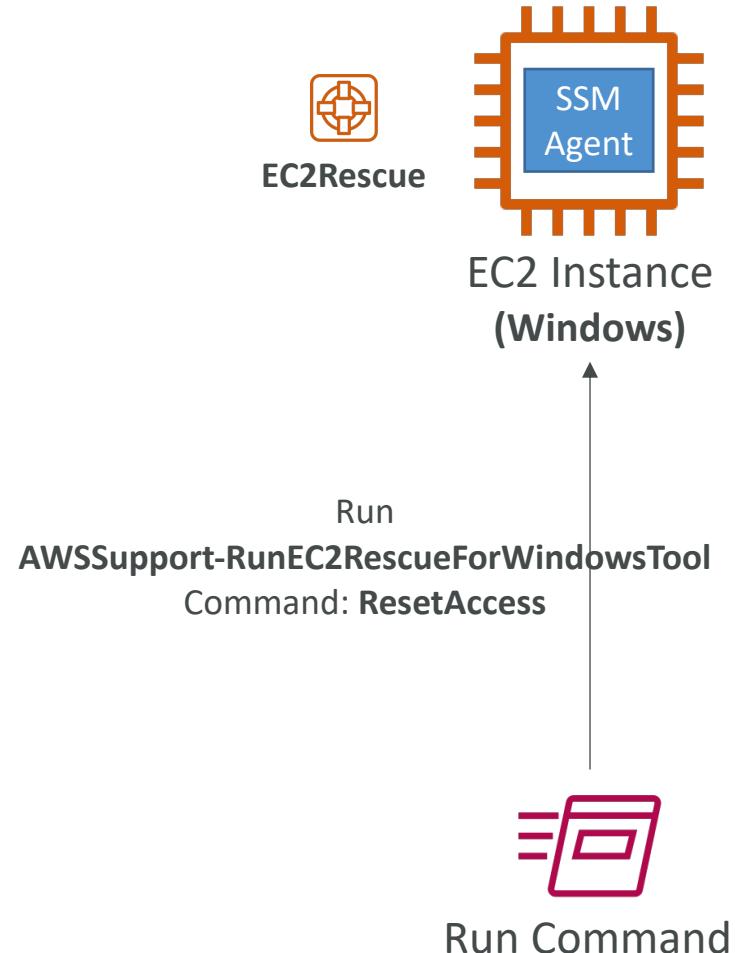
# Connect to Windows EC2 Instance with a Lost Password – Using EC2Launch

- Windows Server 2016 and later AMIs that doesn't include EC2Launch v2
- Detach the EBS root volume
- Attach the volume to a temporary instance as a secondary volume
- Download and install EC2Rescue Tool for Windows Server
- Select Offline Instance Option -> Diagnose and Rescue -> Reset Administrator Password
- Reattach the volume to the original instance, then restart the instance



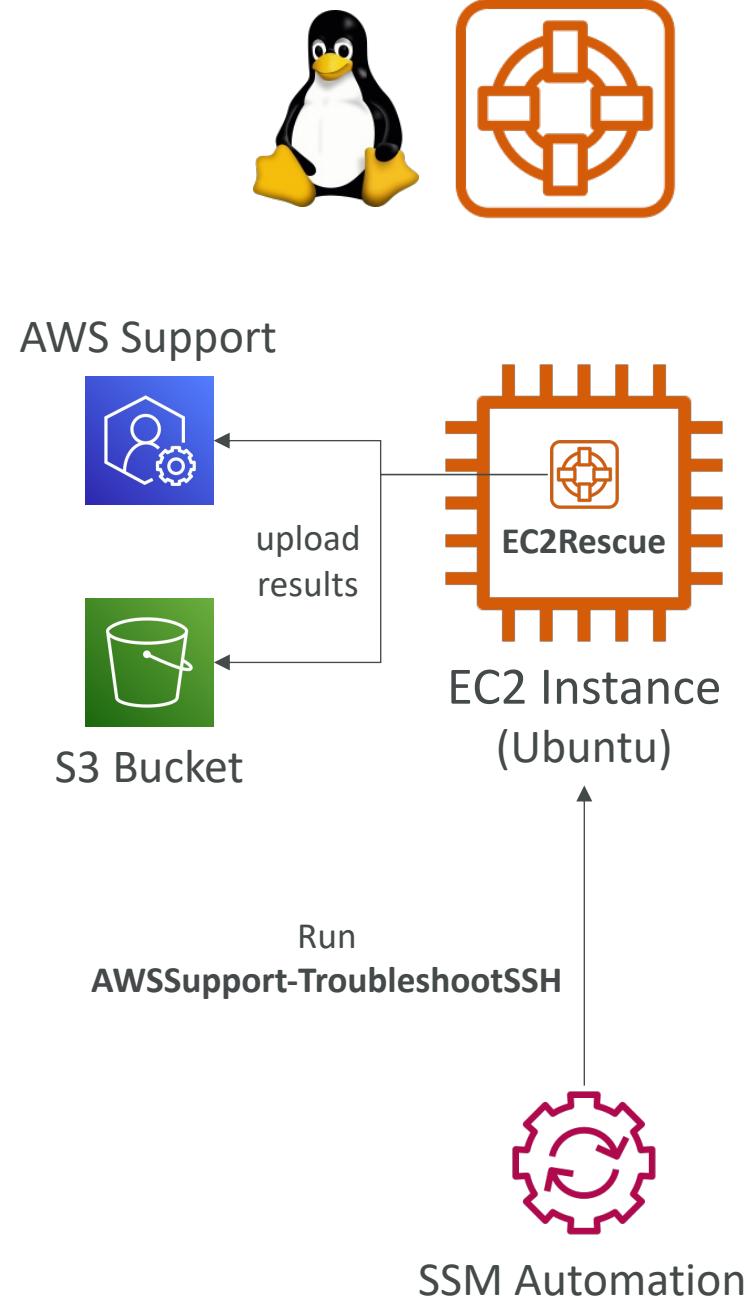
# Connect to Windows EC2 Instance with a Lost Password – Using Systems Manager

- Must have SSM Agent installed
- Method 1
  - Use AWSSupport-RunEC2RescueForWindowsTool Run Command Document
  - Install and run EC2Rescue Tool for Windows Server
  - Command is set to ResetAccess
- Method 2
  - Use AWSSupport-ResetAccess Automation Document
  - Works for both Linux and Windows
- Method 3
  - Manually AWS-RunPowerShellScript Run Command Document
  - Command: `net user Administrator Password@123`



# EC2Rescue Tool for Linux

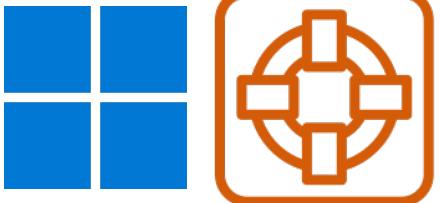
- Diagnose and troubleshoot common issues
- Gather syslog logs, diagnose problematic kernel parameters, diagnose common OpenSSH issues, ...
- Supports over 100 modules
- Amazon Linux 2, Ubuntu, RHEL, SUSE Linux
- Install manually or using **AWS Support-TroubleshootSSH** Automation Document
  - Installs the tool and tries to fix issues with SSH connections to the instance
- Upload the results directly to AWS Support or an S3 bucket



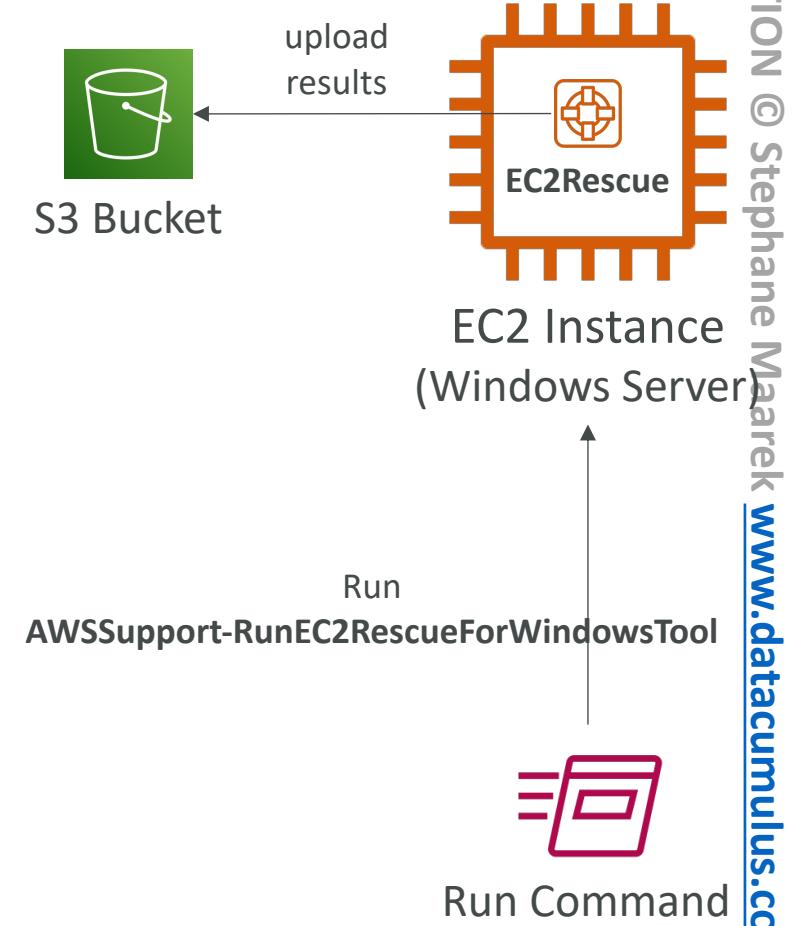
# EC2Rescue Tool for Linux – Use Cases

- Collect System Utilization Reports
  - vmstate, iostat, mpstat, ...
- Collect Logs and Details
  - syslog, dmesg, application error logs, and SSM logs
- Detect System Problems
  - Asymmetric routing or duplicate root device labels
- Automatically Remediate System Problems
  - Correcting OpenSSH file permissions
  - Disabling known problematic kernel problems
- You can create your own custom module

# EC2Rescue Tool for Windows Server



- Diagnose and troubleshoot common issues
- Collect log files, troubleshoot issues, provide suggestions, ...
- Supports 2 modules (data collector, analyzer)
- Windows Server 2008 R2 or later
- Install manually or using **AWSSupport-AWSSupport-RunEC2RescueForWindowsTool**  
Run Command Document
  - Commands: CollectLogs, FixAll, ResetAccess
- Use **AWSSupport-ExecuteEC2Rescue** Automation Document to troubleshoot connectivity issues
- Upload the results directly to an S3 bucket



# EC2Rescue Tool for Windows Server – Use Cases

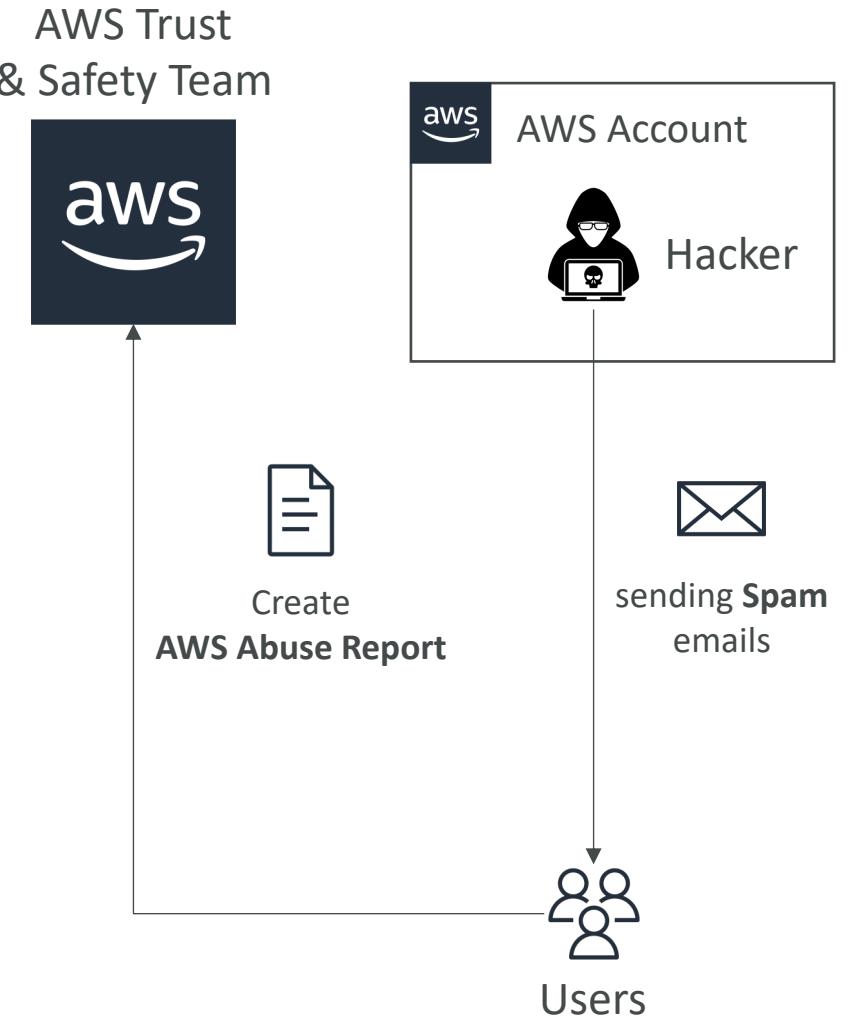
- Instance Connectivity Issues
  - Firewall, RDP, or network interface configuration
- OS Boot Issues
  - Blue screen or stop error, a boot loop, or a corrupted registry
- Gather OS Logs and Configuration Files
  - If you need advanced log analysis and troubleshooting
- Common OS Issues
  - Disk signature collision
- Perform a restore

# AWS Acceptable Use Policy (AUP)

- Governs your use of the services offered by AWS
- You may not use for:
  - Illegal or fraudulent activity
  - Violate the rights of others
  - Threaten, terrorism, violence, or other serious harm
  - Child sexual abuse content or activity
  - Violate the security, integrity or availability for other networks and computers
  - Distribute, publish or facilitate the unsolicited mass emails (e.g., spams)
- AWS will remove or disable any content that violates this policy
- <https://aws.amazon.com/aup/>

# AWS Abuse Report

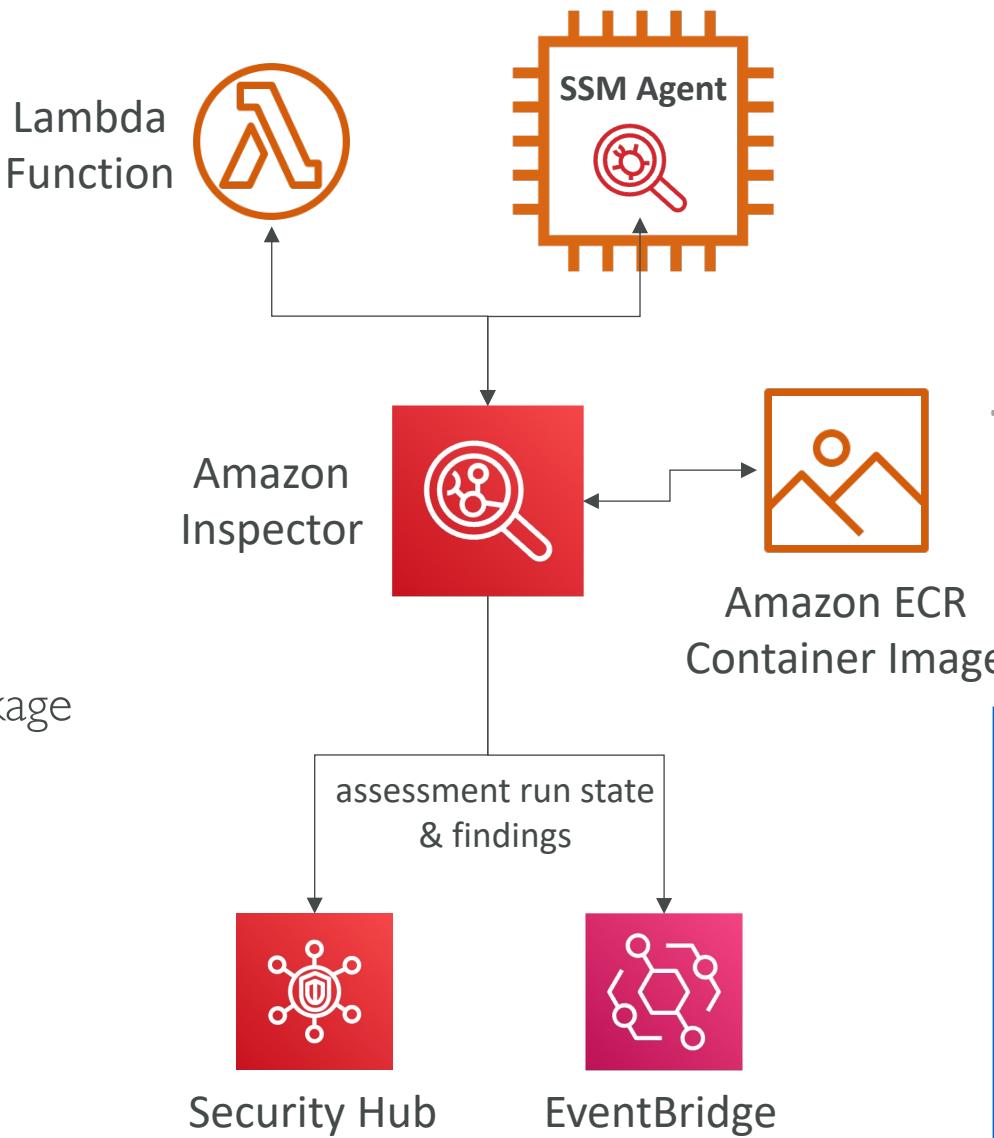
- When you suspect that AWS resources are used for abusive or illegal purposes
- You can create an AWS Abuse Report
- Examples: spam, port scanning, DDoS attacks, intrusion attempts, hosting prohibited content, distributing malware
- Contact **AWS Trust & Safety Team** with details (e.g., logs, email headers, ...)
- If you receive an email that your AWS resources are used for illegal activity:
  - Respond to the email and explain how you're preventing this
  - If you don't respond within 24 hours, AWS might suspend your AWS account



# Domain 2 – Logging and Monitoring

# Amazon Inspector

- Automated Security Assessments
- For EC2 instances
  - Leveraging the AWS System Manager (SSM) agent
  - Analyze against unintended network accessibility
  - Analyze the running OS against known vulnerabilities
- For Container Images push to Amazon ECR
  - Assessment of Container Images as they are pushed
- For Lambda Functions
  - Identifies software vulnerabilities in function code and package dependencies
  - Assessment of functions as they are deployed
- Reporting & integration with AWS Security Hub
- Send findings to Amazon Event Bridge



# What does Amazon Inspector evaluate?



- Remember: only for EC2 instances, Container Images & Lambda functions
- Continuous scanning of the infrastructure, only when needed
- Package vulnerabilities (EC2, ECR & Lambda) – database of CVE
- Network reachability (EC2)
- A risk score is associated with all vulnerabilities for prioritization

# Logging in AWS for security and compliance

- To help compliance requirements, AWS provides many service-specific security and audit logs
- Service Logs include:
  - CloudTrail trails - trace all API calls
  - Config Rules - for config & compliance over time
  - CloudWatch Logs - for full data retention
  - VPC Flow Logs - IP traffic within your VPC
  - ELB Access Logs - metadata of requests made to your load balancers
  - CloudFront Logs - web distribution access logs
  - WAF Logs - full logging of all requests analyzed by the service
- Logs can be analyzed using AWS Athena if they're stored in S3
- You should encrypt logs in S3, control access using IAM & Bucket Policies, MFA
- Move Logs to Glacier for cost savings
- Read whitepaper if interested at:  
[https://d0.awsstatic.com/whitepapers/compliance/AWS\\_Security\\_at\\_Scale\\_Logging\\_in\\_AWS\\_Whitepaper.pdf](https://d0.awsstatic.com/whitepapers/compliance/AWS_Security_at_Scale_Logging_in_AWS_Whitepaper.pdf)

# AWS Systems Manager Overview



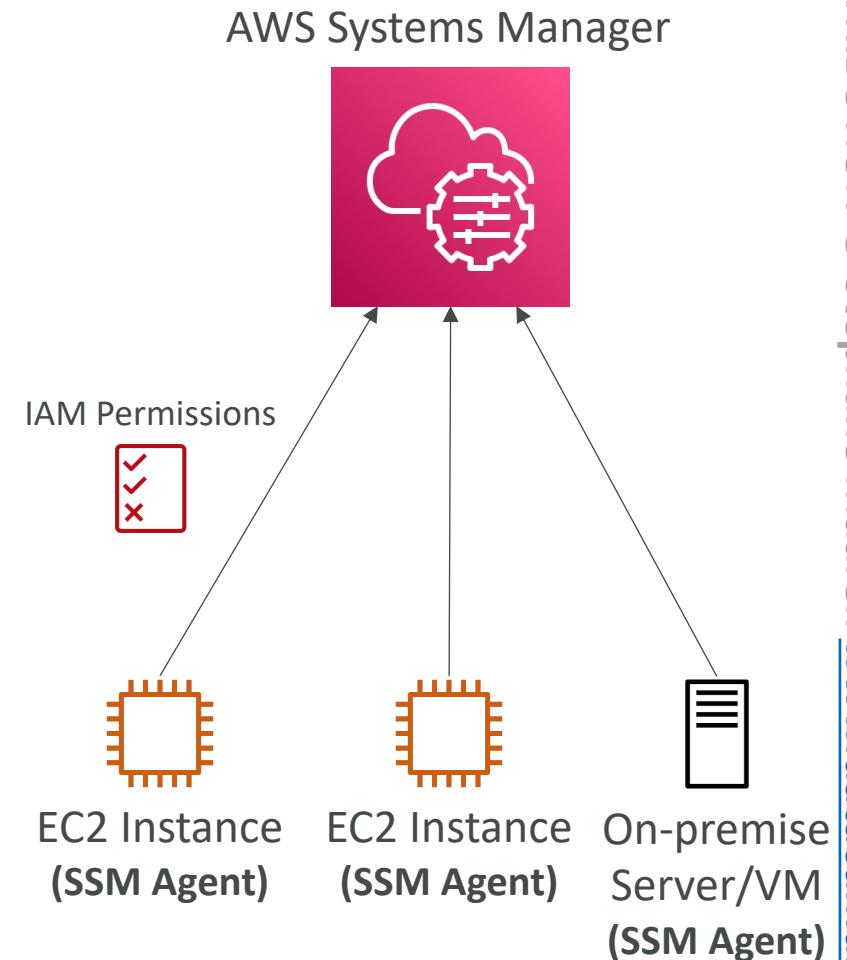
- Helps you manage your **EC2** and **On-Premises** systems at scale
- Get operational insights about the state of your infrastructure
- Easily detect problems
- **Patching automation for enhanced compliance**
- Works for both Windows and Linux OS
- Integrated with CloudWatch metrics / dashboards
- Integrated with AWS Config
- Free service

# AWS Systems Manager Features

- Resource Groups
- Operations Management
  - Explorer
  - OpsCenter
  - CloudWatch Dashboard
  - PHD
  - Incident Manager
- Shared Resources
  - Documents
- Change Management
  - Change Manager
  - Automation
  - Change Calendar
  - Maintenance Windows
- Application Management
  - Application Manager
  - AppConfig
  - Parameter Store
- Node Management
  - Fleet Manager
  - Compliance
  - Inventory
  - Hybrid Activations
  - Session Manager
  - Run Command
  - State Manager
  - Patch Manager
  - Distributer

# How Systems Manager works

- We need to install the SSM agent onto the systems we control
- Installed by default on Amazon Linux 2 AMI & some Ubuntu AMI
- If an instance can't be controlled with SSM, it's probably an issue with the SSM agent!
- Make sure the EC2 instances have a proper IAM role to allow SSM actions



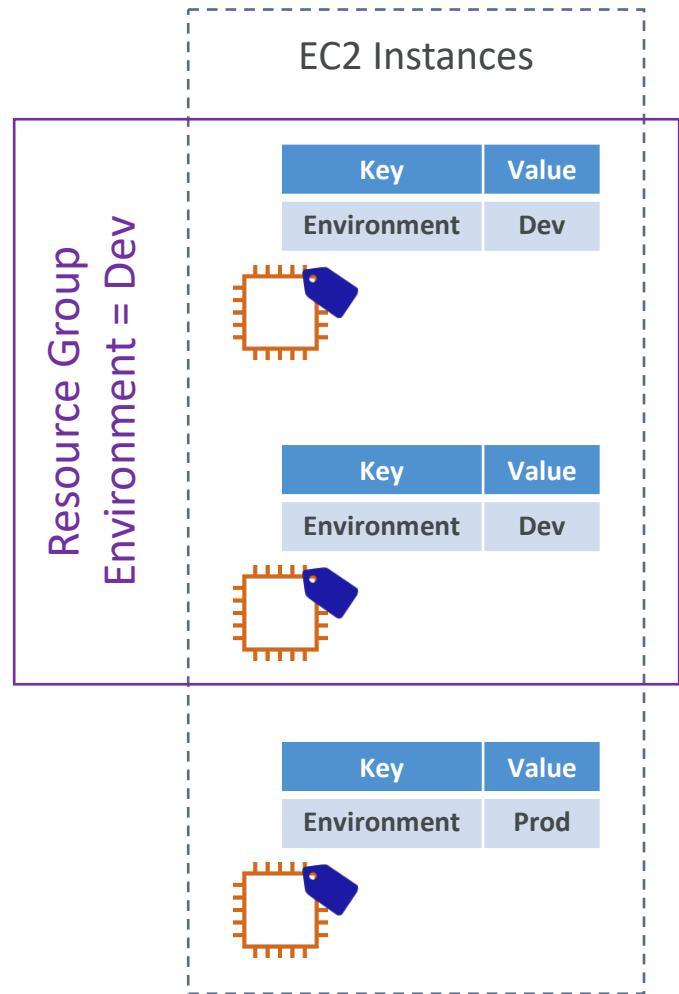
# AWS Tags



- You can add text key-value pairs called Tags to many AWS resources
- Commonly used in EC2
- Free naming, common tags are Name, Environment, Team ...
- They're used for
  - Resource grouping
  - Automation
  - Cost allocation
- Better to have too many tags than too few!

# Resource Groups

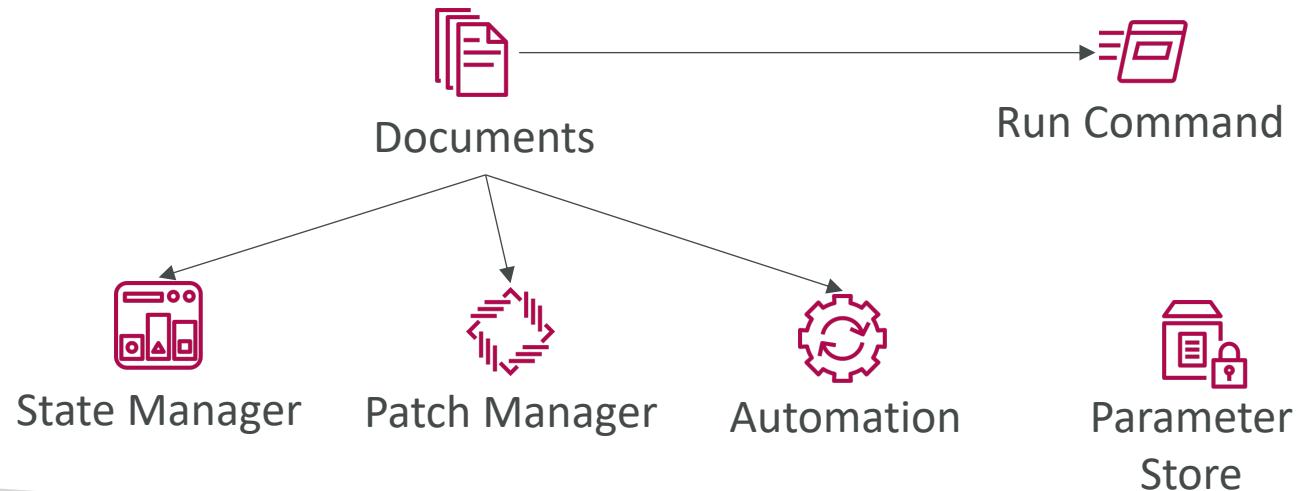
- Create, view or manage logical group of resources thanks to **tags**
- Allows creation of logical groups of resources such as
  - Applications
  - Different layers of an application stack
  - Production versus development environments
- Regional service
- Works with EC2, S3, DynamoDB, Lambda, etc...



# SSM – Documents

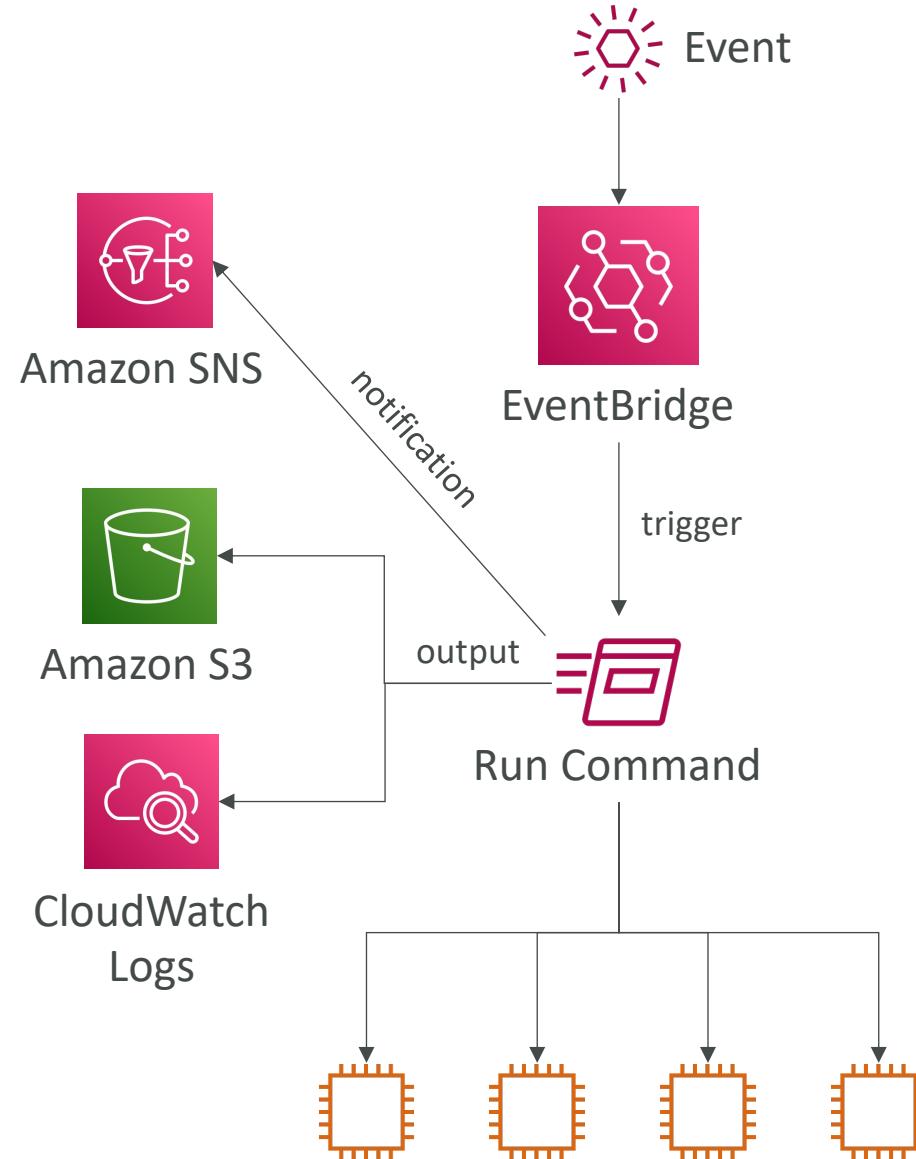
- Documents can be in JSON or YAML
- You define parameters
- You define actions
- Many documents already exist in AWS

```
---  
schemaVersion: '2.2'  
description: State Manager Bootstrap Example  
parameters: {}  
mainSteps:  
  - action: aws:runShellScript  
    name: configureServer  
    inputs:  
      runCommand:  
        - sudo yum install -y httpd  
        - sudo yum --enablerepo=epel install -y clamav
```



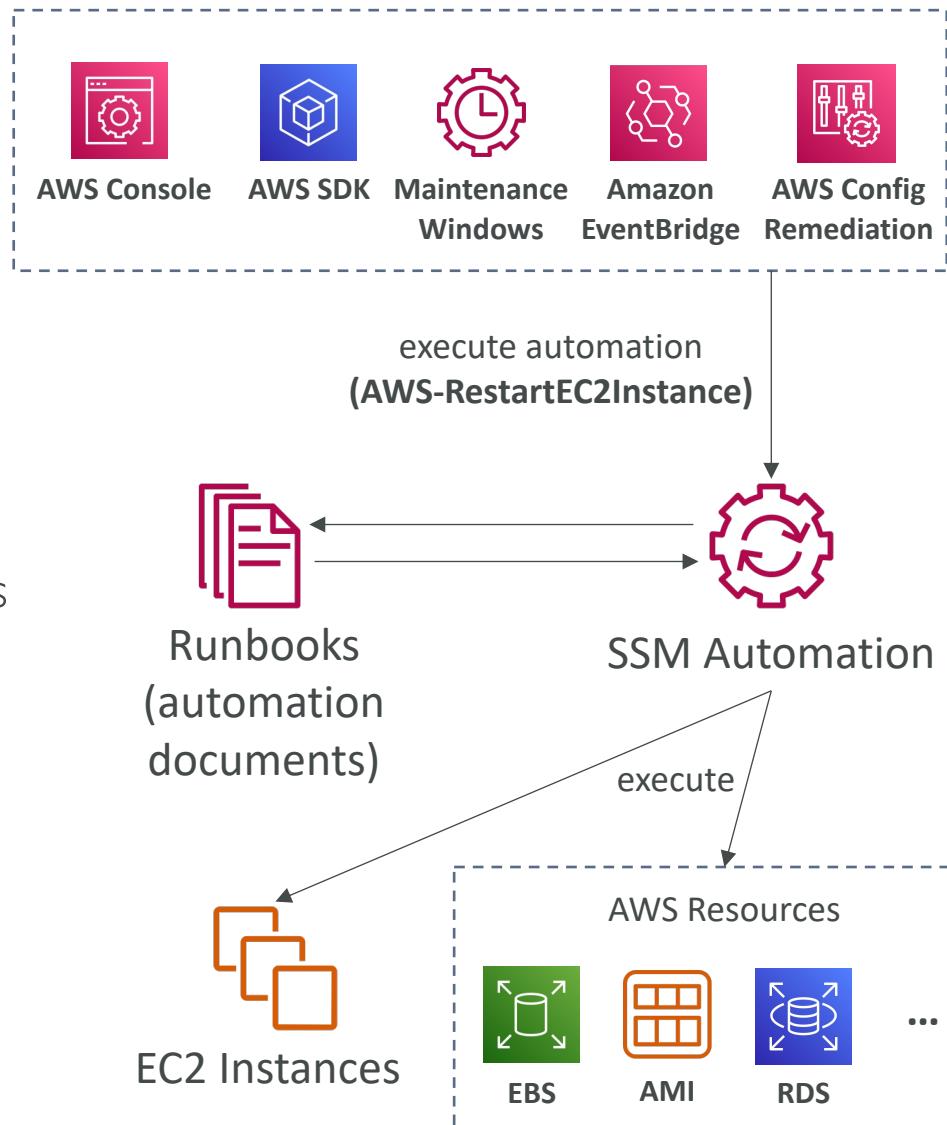
# SSM – Run Command

- Execute a document (= script) or just run a command
- Run command across multiple instances (using resource groups)
- Rate Control / Error Control
- Integrated with IAM & CloudTrail
- No need for SSH
- Command Output can be shown in the Console, sent to S3 bucket or CloudWatch Logs
- Send notifications to SNS about command statuses (In progress, Success, Failed, ...)
- Can be invoked using EventBridge



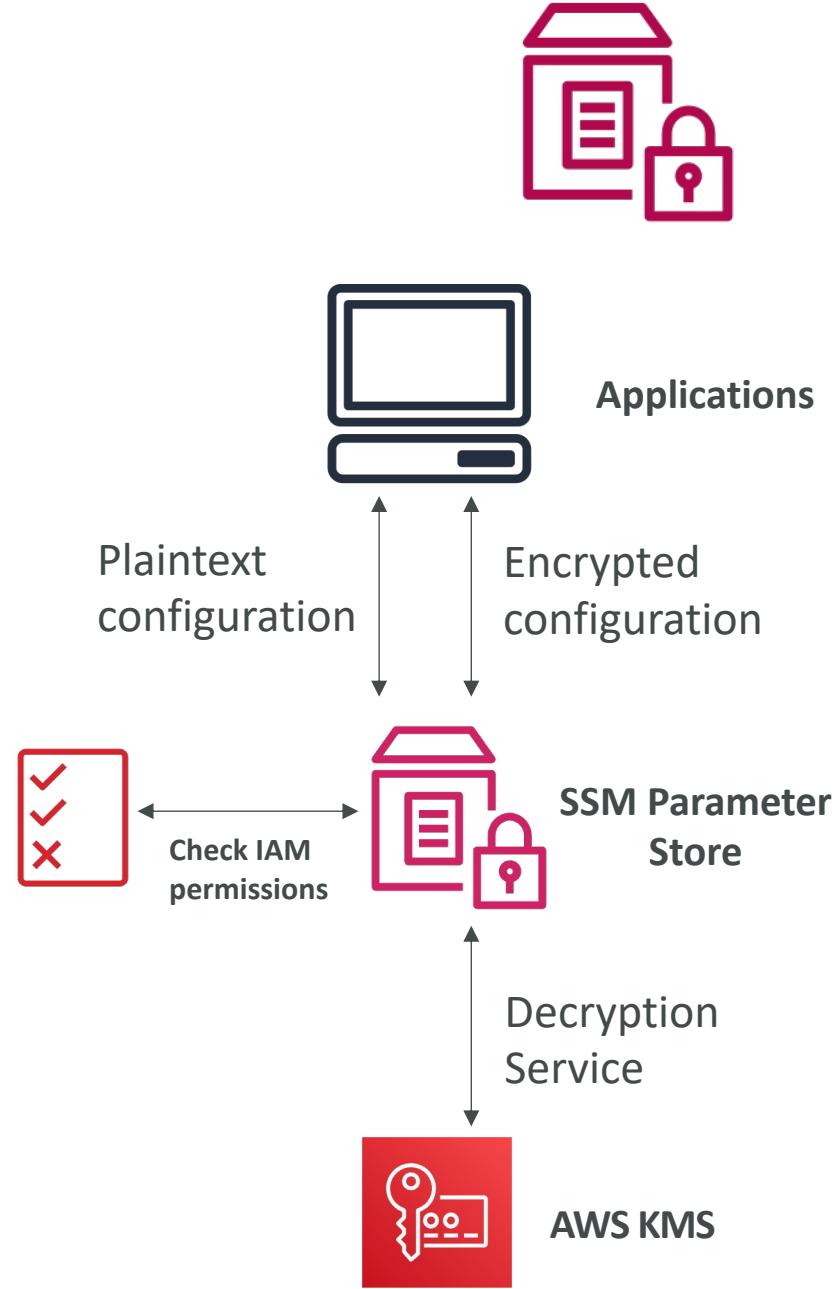
# SSM - Automation

- Simplifies common maintenance and deployment tasks of EC2 instances and other AWS resources
- Example: restart instances, create an AMI, EBS snapshot
- **Automation Runbook**
  - SSM Documents of type Automation
  - Defines actions preformed on your EC2 instances or AWS resources
  - Pre-defined runbooks (AWS) or create custom runbooks
- Can be triggered
  - Manually using AWS Console, AWS CLI or SDK
  - By Amazon EventBridge
  - On a schedule using Maintenance Windows
  - By AWS Config for rules remediations



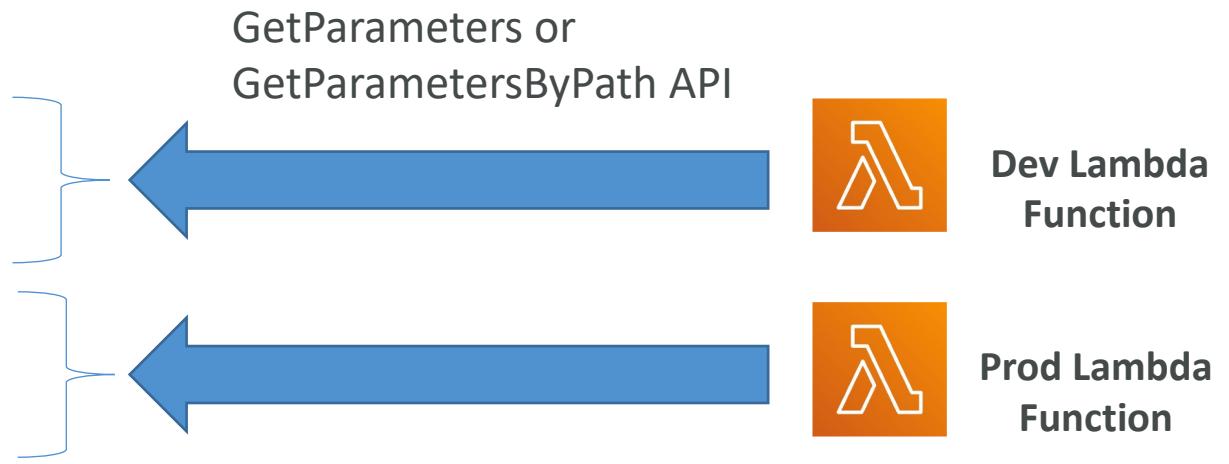
# SSM Parameter Store

- Secure storage for configuration and secrets
- Optional Seamless Encryption using KMS
- Serverless, scalable, durable, easy SDK
- Version tracking of configurations / secrets
- Security through IAM
- Notifications with Amazon EventBridge
- Integration with CloudFormation



# SSM Parameter Store Hierarchy

- /my-department/
  - my-app/
    - dev/
      - db-url
      - db-password
    - prod/
      - db-url
      - db-password
  - other-app/
  - /other-department/
  - /aws/reference/secretsmanager/secret\_ID\_in\_Secrets\_Manager
  - /aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86\_64-gp2 (public)



# Standard and advanced parameter tiers

	Standard	Advanced
Total number of parameters allowed (per AWS account and Region)	10,000	100,000
Maximum size of a parameter value	4 KB	8 KB
Parameter policies available	No	Yes
Cost	No additional charge	Charges apply
Storage Pricing	Free	\$0.05 per advanced parameter per month

# Parameters Policies (for advanced parameters)

- Allow to assign a TTL to a parameter (expiration date) to force updating or deleting sensitive data such as passwords
- Can assign multiple policies at a time

**Expiration (to delete a parameter)**

```
{  
  "Type": "Expiration",  
  "Version": "1.0",  
  "Attributes": {  
    "Timestamp": "2020-12-02T21:34:33.000Z"  
  }  
}
```

**ExpirationNotification (EventBridge)**

```
{  
  "Type": "ExpirationNotification",  
  "Version": "1.0",  
  "Attributes": {  
    "Before": "15",  
    "Unit": "Days"  
  }  
}
```

**NoChangeNotification (EventBridge)**

```
{  
  "Type": "NoChangeNotification",  
  "Version": "1.0",  
  "Attributes": {  
    "After": "20",  
    "Unit": "Days"  
  }  
}
```



# SSM – Inventory

- Collect metadata from your managed instances (EC2/On-premises)
- Metadata includes installed software, OS drivers, configurations, installed updates, running services ...
- View data in AWS Console or store in S3 and query and analyze using Athena and QuickSight
- Specify metadata collection interval (minutes, hours, days)
- Query data from multiple AWS accounts and regions
- Create Custom Inventory for your custom metadata (e.g., rack location of each managed instance)

# SSM – State Manager



- Automate the process of keeping your managed instances (EC2/On-premises) in a state that you define
- Use cases: bootstrap instances with software, patch OS/software updates on a schedule ...
- **State Manager Association:**
  - Defines the state that you want to maintain to your managed instances
  - Example: port 22 must be closed, antivirus must be installed ...
  - Specify a schedule when this configuration is applied
- Uses SSM Documents to create an Association (e.g., SSM Document to configure CW Agent)



# SSM – Patch Manager

- Automates the process of patching managed instances
- OS updates, applications updates, security updates, ...
- Supports both EC2 instances and on-premises servers
- Supports Linux, macOS, and Windows
- Patch on-demand or on a schedule using **Maintenance Windows**
- Scan instances and generate patch compliance report (missing patches)
- Patch compliance report can be sent to S3



# SSM – Patch Manager

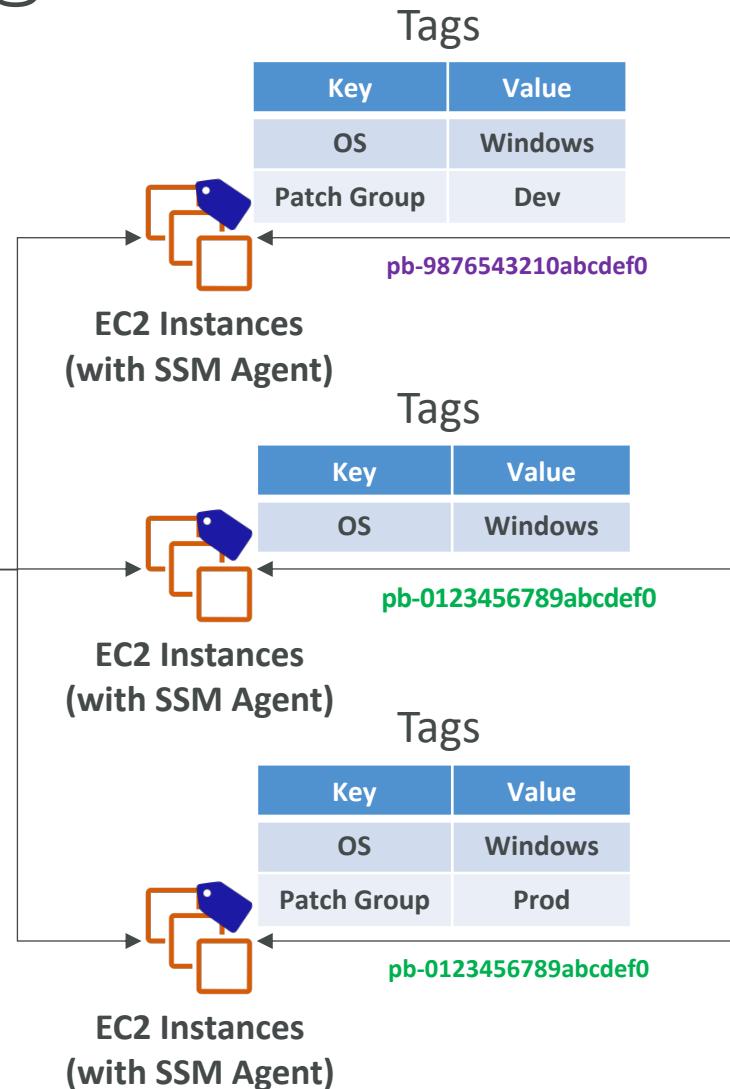
- **Patch Baseline**

- Defines which patches should and shouldn't be installed on your instances
- Ability to create custom Patch Baselines (specify approved/rejected patches)
- Patches can be auto-approved within days of their release
- By default, install only critical patches and patches related to security

- **Patch Group**

- Associate a set of instances with a specific Patch Baseline
- Example: create Patch Groups for different environments (dev, test, prod)
- Instances should be defined with the tag key **Patch Group**
- An instance can only be in one Patch Group
- Patch Group can be registered with only one Patch Baseline

# SSM – Patch Manager



Patch Baselines		
Patch Baseline ID	Patch Group	Default
pb-0123456789abcdef0	Default	Yes
pb-9876543210abcdef0	Dev	No

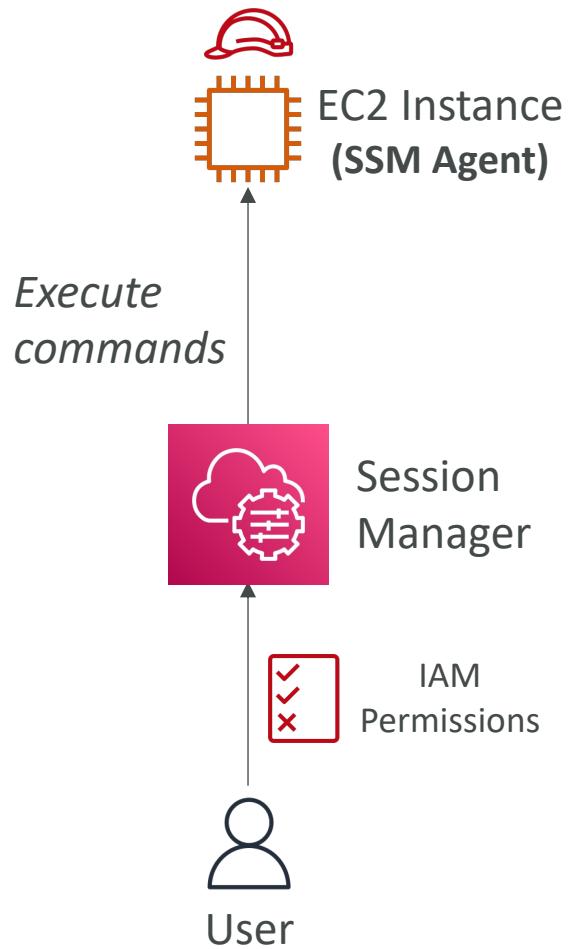
# SSM – Maintenance Windows



- Defines a schedule for when to perform actions on your instances
- Example: OS patching, updating drivers, installing software, ...
- Maintenance Window contains
  - Schedule
  - Duration
  - Set of registered instances
  - Set of registered tasks

# SSM – Session Manager

- Allows you to start a secure shell on your EC2 and on-premises servers
- Access through AWS Console, AWS CLI, or Session Manager SDK
- Does not need SSH access, bastion hosts, or SSH keys
- Supports Linux, macOS, and Windows
- Log connections to your instances and executed commands
- Session log data can be sent to S3 or CloudWatch Logs
- CloudTrail can intercept **StartSession** events



# SSM – Session Manager

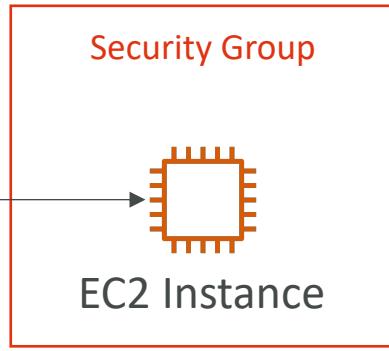
- IAM Permissions
  - Control which users/groups can access Session Manager and which instances
  - Use tags to restrict access to only specific EC2 instances
  - Access SSM + write to S3 + write to CloudWatch
- Optionally, you can restrict commands a user can run in a session

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Action": "ssm:StartSession",  
      "Resource": "arn:aws:ec2:us-east-1:123456789012:instance/*",  
      "Condition": {  
        "StringLike": {  
          "ssm:resourceTag/Environment": ["Dev"]  
        }  
      }  
    }  
  ]  
}
```

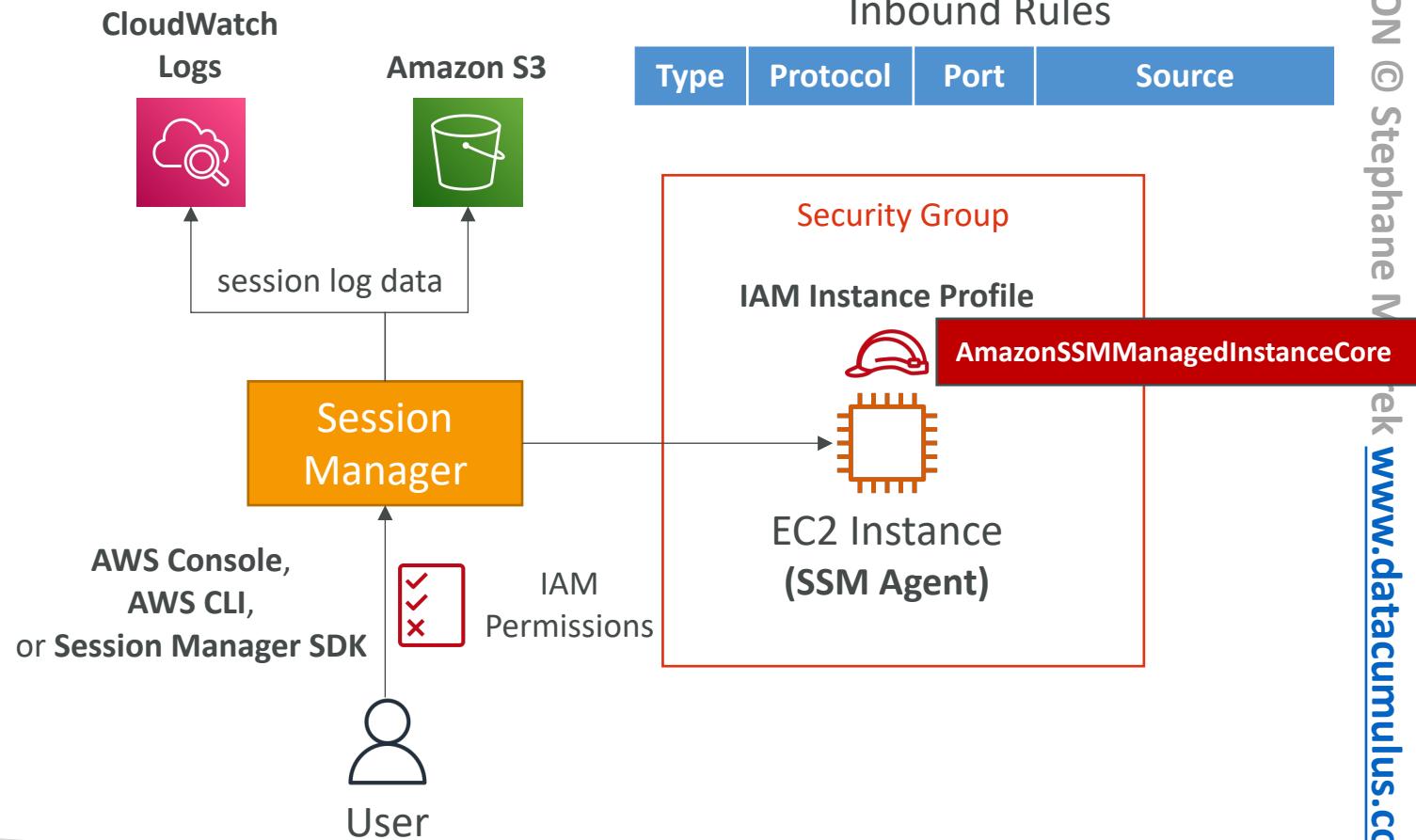
# SSH vs. SSM Session Manager

## Connect using SSH

Inbound Rules			
Type	Protocol	Port	Source
SSH	TCP	22	1.2.3.4/32

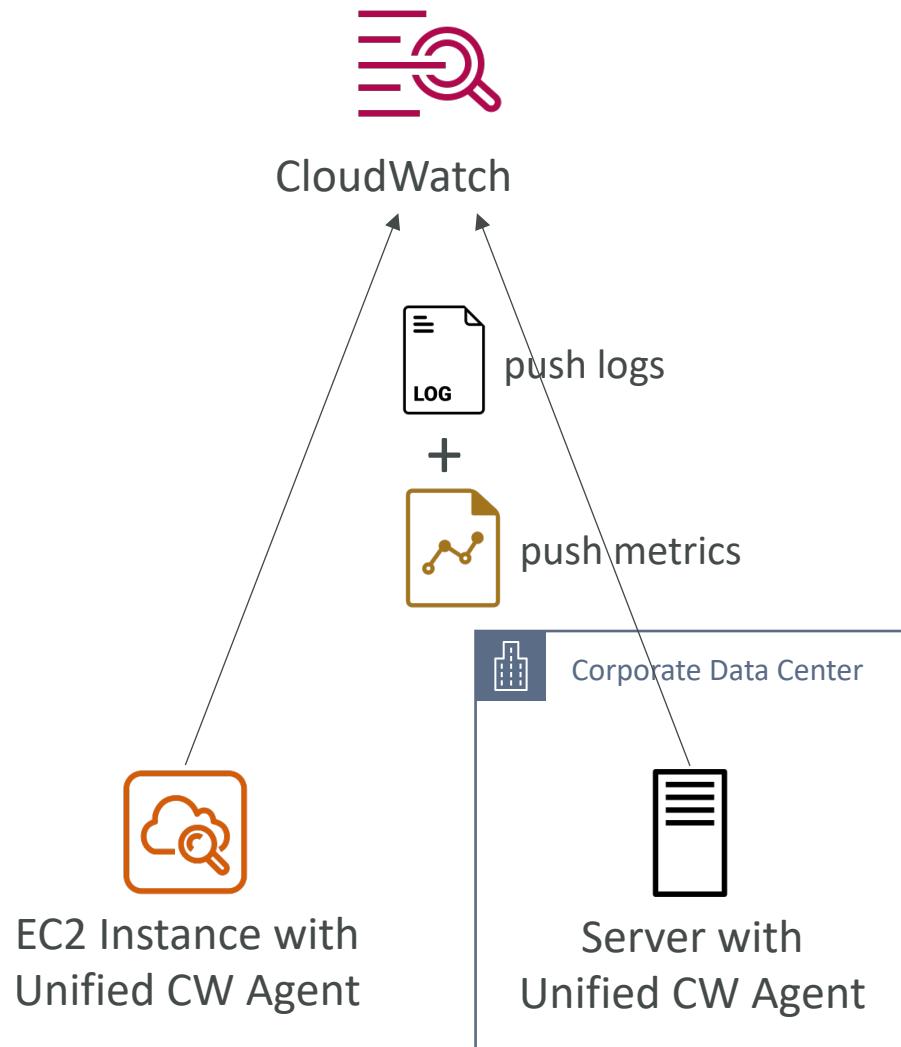


## Connect using SSM Session Manager



# Unified CloudWatch Agent

- For virtual servers (EC2 instances, on-premises servers, ...)
- Collect additional system-level metrics such as RAM, processes, used disk space, etc.
- Collect logs to send to CloudWatch Logs
  - No logs from inside your EC2 instance will be sent to CloudWatch Logs without using an agent
- Centralized configuration using SSM Parameter Store
- Make sure IAM permissions are correct
- Default namespace for metrics collected by the Unified CloudWatch agent is **CWAgent** (can be configured/changed)



# Unified CloudWatch Agent – procstat Plugin

- Collect metrics and monitor system utilization of individual processes
- Supports both Linux and Windows servers
- Example: amount of time the process uses CPU, amount of memory the process uses, ...
- Select which processes to monitor by
  - `pid_file`: name of process identification number (PID) files they create
  - `exe`: process name that match string you specify (RegEx)
  - `pattern`: command lines used to start the processes (RegEx)
- Metrics collected by procstat plugin begins with “`procstat`” prefix (e.g., `procstat_cpu_time`, `procstat_cpu_usage`, ...)

# Unified CloudWatch Agent - Troubleshooting

- CloudWatch Agent Fails to Start
  - Might be an issue with the configuration file
  - Check configuration file logs at `/opt/aws/amazon-cloudwatch-agent/logs/configuration-validation.log`
- Can't Find Metrics Collected by the CloudWatch Agent
  - Check you're using the correct namespace (default: CWAgent)
  - Check the configuration file `amazon-cloudwatch-agent.json`

```
"agent": {  
    "metrics_collection_interval": 60,  
    "region": "us-west-1",  
    "logfile": "/opt/aws/amazon-cloudwatch-agent/logs/amazon-cloudwatch-agent.log",  
    "debug": false,  
    "run_as_user": "cwagent"  
}
```

*amazon-cloudwatch-agent.json* Sample

# Unified CloudWatch Agent - Troubleshooting

- **CloudWatch Agent Not Pushing Log Events**

- Update to the latest CloudWatch Agent version
- Test connectivity to the CloudWatch Logs endpoint `logs.<region>.amazonaws.com` (check SGs, NACLs, ...)
- Review account, region, and log group configurations
- Check IAM Permissions
- Verify the system time on the instance is correctly configured

```
[ "logs:CreateLogGroup",  
  "logs:CreateLogStream",  
  "logs:PutLogEvents",  
  "logs:DescribeLogStreams" ]
```

Custom Policy

+  [CloudWatchAgentAdminPolicy](#)

+  [CloudWatchAgentServerPolicy](#)

Managed Policies  
(recommended)

- Check CloudWatch Agent logs at `/opt/aws/amazon-cloudwatch-agent/logs/amazon-cloudwatch-agent.log`



# CloudWatch Logs

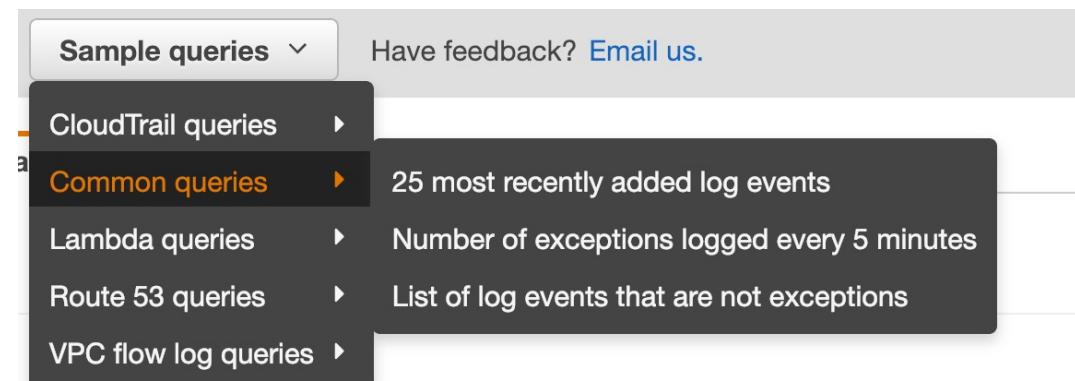
- **Log groups:** arbitrary name, usually representing an application
- **Log stream:** instances within application / log files / containers
- Can define log expiration policies (never expire, 30 days, etc..)
- **CloudWatch Logs can send logs to:**
  - Amazon S3 (exports)
  - Kinesis Data Streams
  - Kinesis Data Firehose
  - AWS Lambda
  - OpenSearch

# CloudWatch Logs - Sources

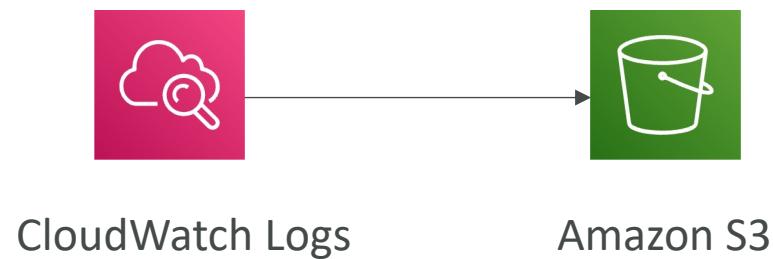
- SDK, CloudWatch Logs Agent, CloudWatch Unified Agent
- Elastic Beanstalk: collection of logs from application
- ECS: collection from containers
- AWS Lambda: collection from function logs
- VPC Flow Logs: VPC specific logs
- API Gateway
- CloudTrail based on filter
- Route53: Log DNS queries

# CloudWatch Logs Metric Filter & Insights

- CloudWatch Logs can use filter expressions
  - For example, find a specific IP inside of a log
  - Or count occurrences of “ERROR” in your logs
- Metric filters can be used to trigger CloudWatch alarms
- CloudWatch Logs Insights can be used to query logs and add queries to CloudWatch Dashboards

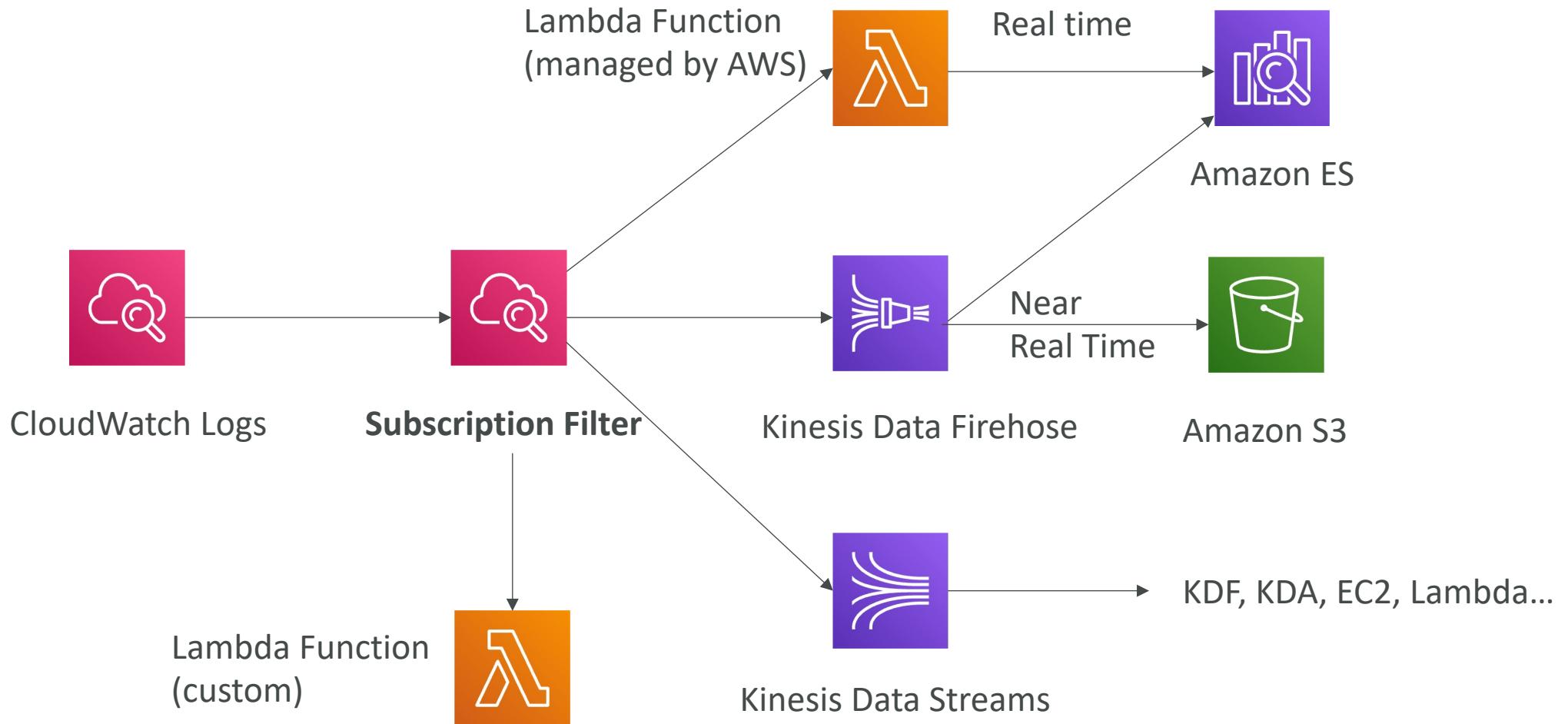


# CloudWatch Logs – S3 Export

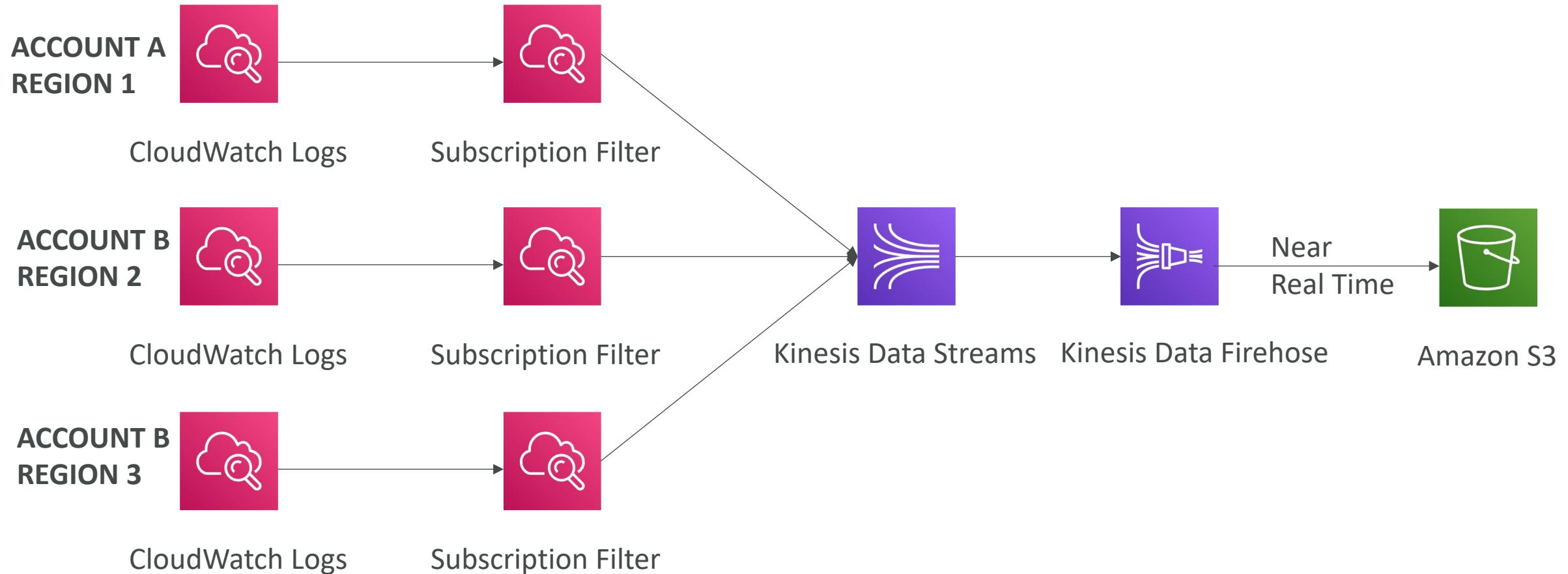


- Log data can take up to 12 hours to become available for export
- The API call is `CreateExportTask`
- Not near-real time or real-time... use Logs Subscriptions instead

# CloudWatch Logs Subscriptions



# CloudWatch Logs Aggregation Multi-Account & Multi Region



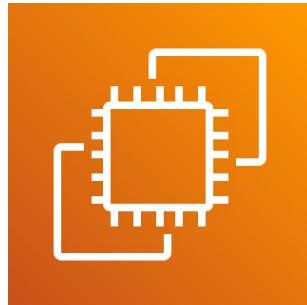
# CloudWatch Alarms



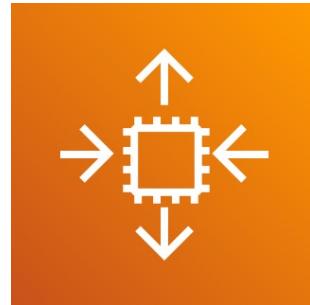
- Alarms are used to trigger notifications for any metric
- Various options (sampling, %, max, min, etc...)
- Alarm States:
  - OK
  - INSUFFICIENT\_DATA
  - ALARM
- Period:
  - Length of time in seconds to evaluate the metric
  - High resolution custom metrics: 10 sec, 30 sec or multiples of 60 sec

# CloudWatch Alarm Targets

- Stop, Terminate, Reboot, or Recover an EC2 Instance
- Trigger Auto Scaling Action
- Send notification to SNS (from which you can do pretty much anything)



Amazon EC2



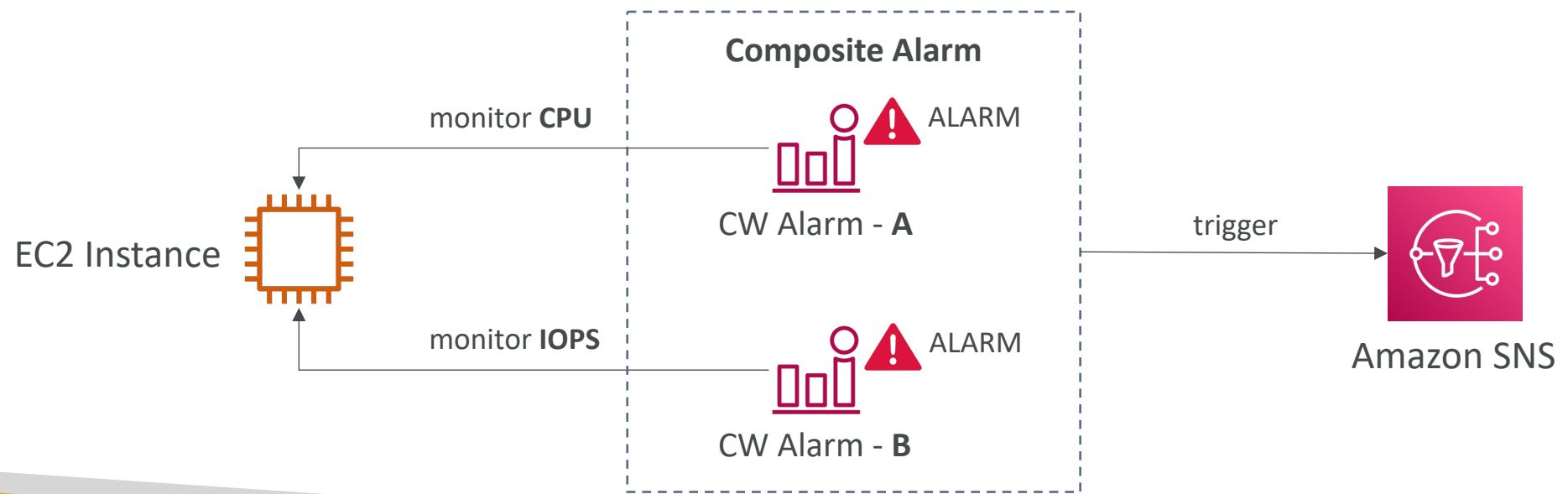
EC2 Auto Scaling



Amazon SNS

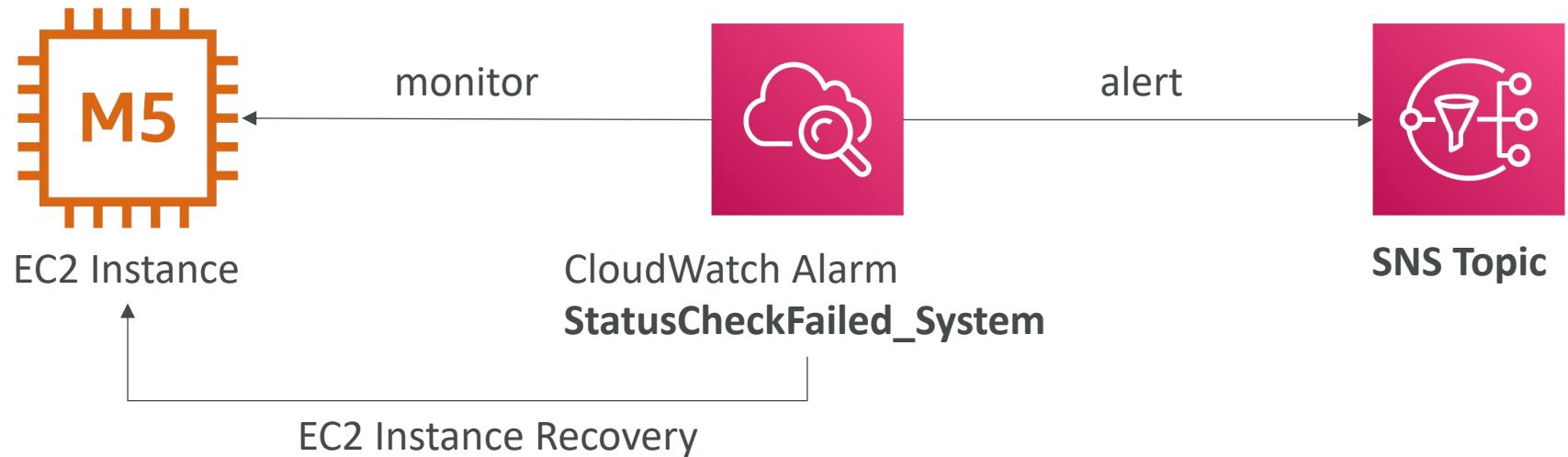
# CloudWatch Alarms – Composite Alarms

- CloudWatch Alarms are on a single metric
- Composite Alarms are monitoring the states of multiple other alarms
- AND and OR conditions
- Helpful to reduce “alarm noise” by creating complex composite alarms



# EC2 Instance Recovery

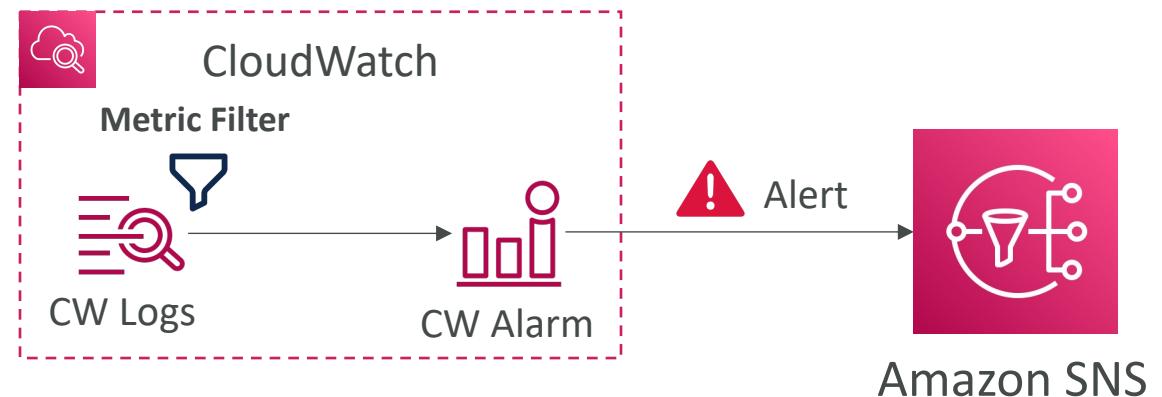
- Status Check:
  - Instance status = check the EC2 VM
  - System status = check the underlying hardware



- Recovery: Same Private, Public, Elastic IP, metadata, placement group

# CloudWatch Alarm: good to know

- Alarms can be created based on CloudWatch Logs Metrics Filters

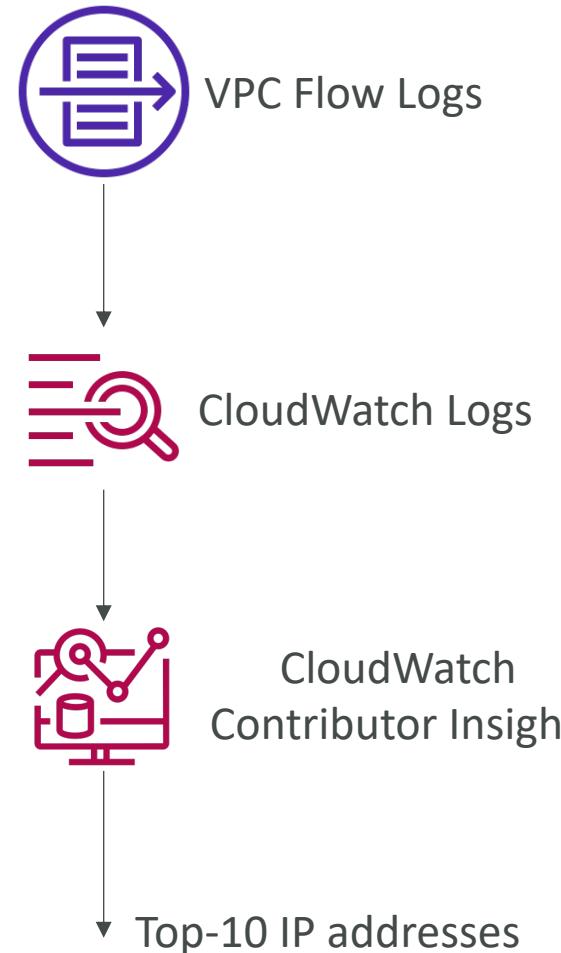


- To test alarms and notifications, set the alarm state to Alarm using CLI  
`aws cloudwatch set-alarm-state --alarm-name "myalarm" --state-value ALARM --state-reason "testing purposes"`

# CloudWatch – Contributor Insights



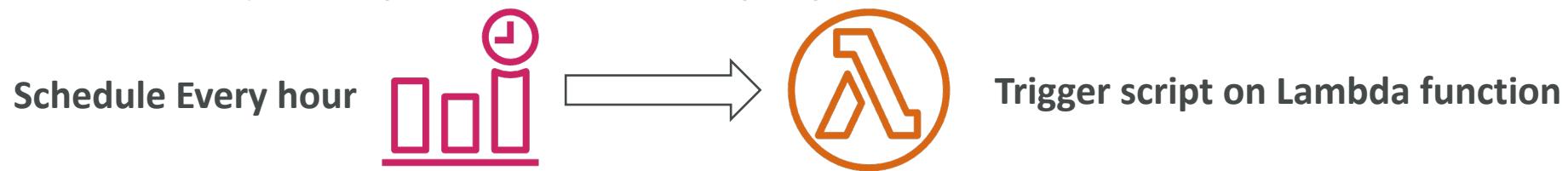
- Analyze log data and create time series that display contributor data
- Helps you find top talkers and understand who/what is impacting system performance
- Example: find bad hosts, identify the heaviest network users, find the URLs that generate the most errors
- Works for any AWS-generated logs (VPC, DNS, etc..)
- Built-in rules created by AWS (leverages your CW Logs) or build your own rules



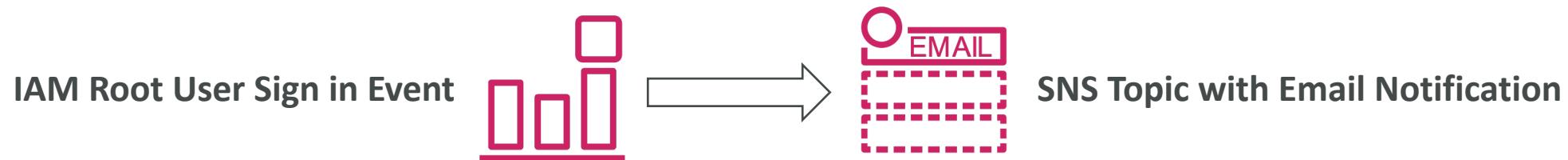
# Amazon EventBridge (formerly CloudWatch Events)



- Schedule: Cron jobs (scheduled scripts)

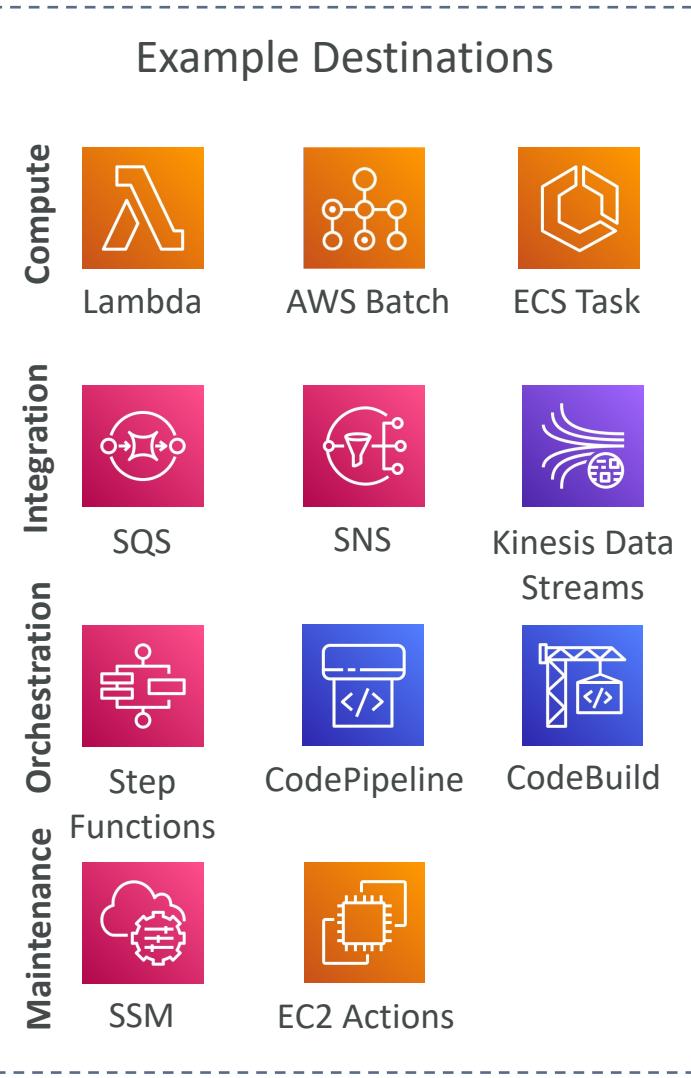
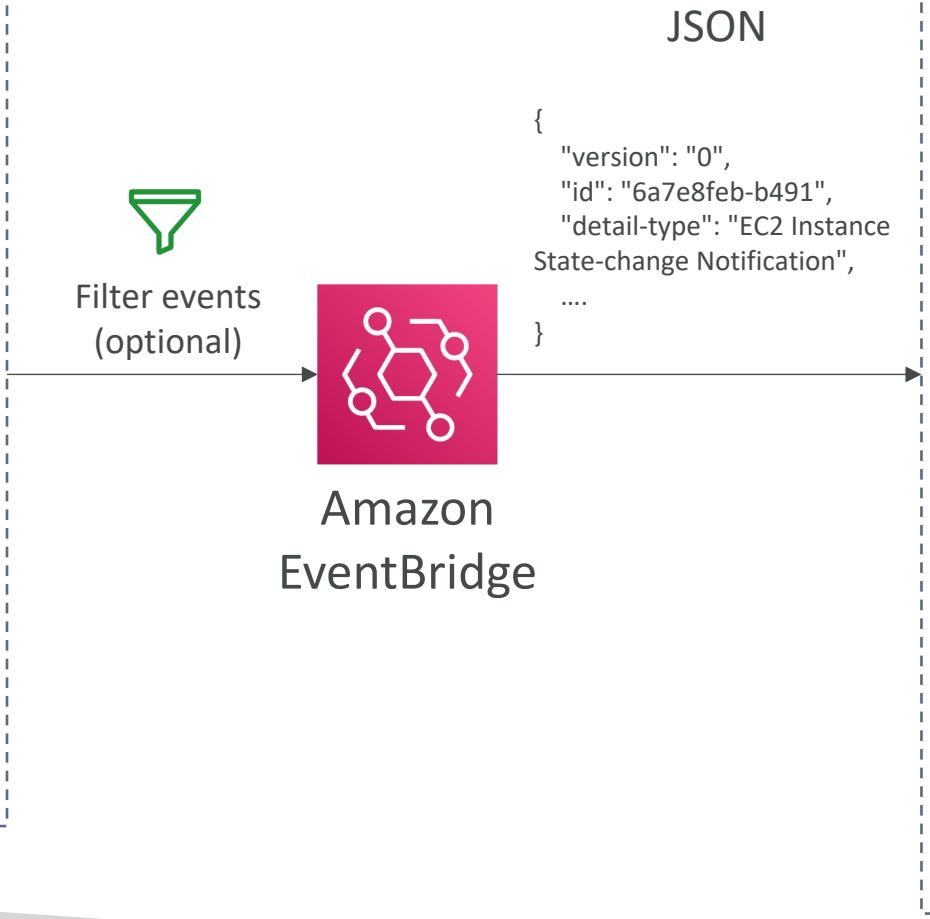
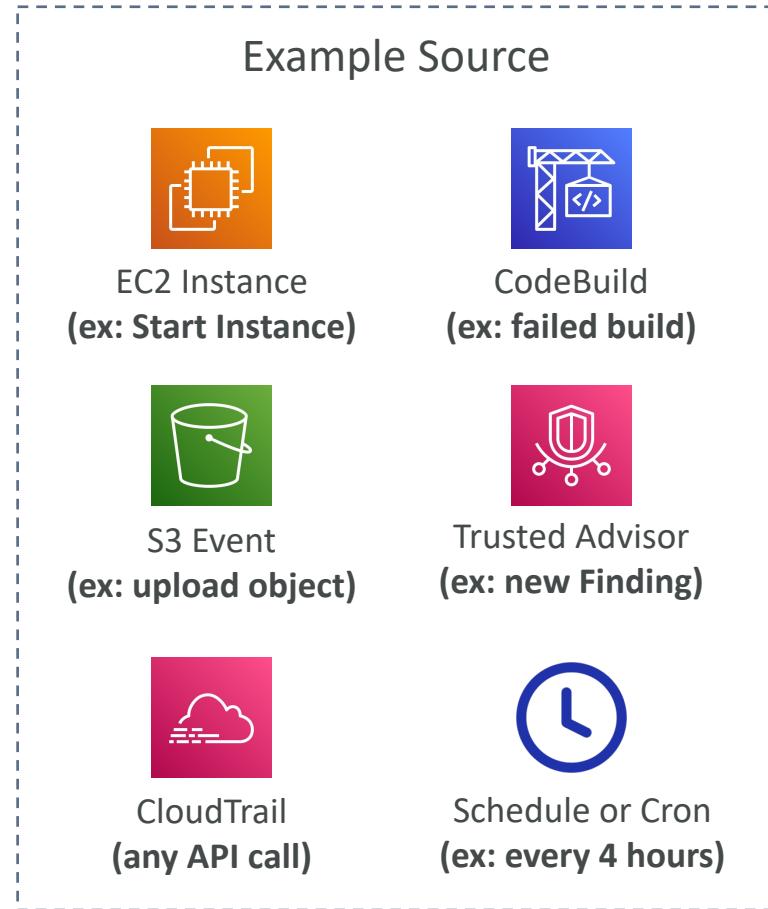


- Event Pattern: Event rules to react to a service doing something

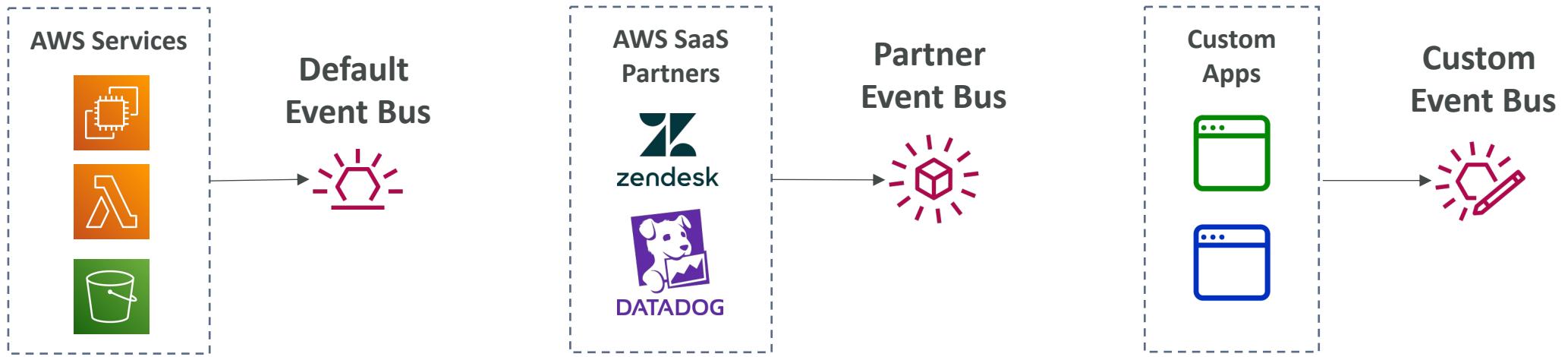


- Trigger Lambda functions, send SQS/SNS messages...

# Amazon EventBridge Rules



# Amazon EventBridge



- Event buses can be accessed by other AWS accounts using Resource-based Policies
- You can archive events (all/filter) sent to an event bus (indefinitely or set period)
- Ability to replay archived events

# Amazon EventBridge – Schema Registry

- EventBridge can analyze the events in your bus and infer the **schema**
- The **Schema Registry** allows you to generate code for your application, that will know in advance how data is structured in the event bus
- Schema can be versioned

The screenshot shows the AWS Schema Registry interface. At the top, it displays the ARN: `aws.codepipeline@CodePipelineActionExecutionStateChange`. Below this, the **Schema details** section provides the following information:

Schema name	Last modified	Schema ARN
<code>aws.codepipeline@CodePipelineActionExecutionStateChange</code>	Dec 1, 2019, 12:11 AM GMT	-
Description		Schema registry aws.events
Schema for event type <code>CodePipelineActionExecutionStateChange</code> , published by AWS service <code>aws.codepipeline</code>		Number of versions 1
		Schema type OpenAPI 3.0

Below the details, the **Version 1** section is shown, created on Dec 1, 2019, 12:11 AM GMT. It includes an **Action** dropdown and a **Download code bindings** button. The schema definition is displayed as follows:

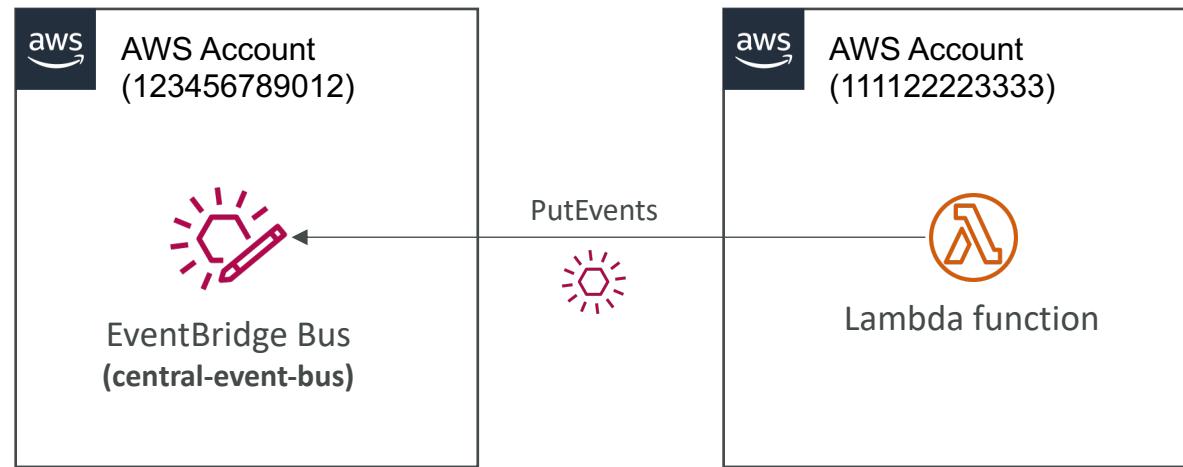
```
1 {
2   "openapi": "3.0.0",
3   "info": {
4     "version": "1.0.0",
5     "title": "CodePipelineActionExecutionStateChange"
6   },
7   "paths": {},
8   "components": {
9     "schemas": {
10       "AWSEvent": {
```

# Amazon EventBridge – Resource-based Policy

- Manage permissions for a specific Event Bus
- Example: allow/deny events from another AWS account or AWS region
- Use case: aggregate all events from your AWS Organization in a single AWS account or AWS region

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": "events:PutEvents",  
            "Principal": { "AWS": "111122223333" },  
            "Resource": "arn:aws:events:us-east-1:123456789012:  
event-bus/central-event-bus"  
        }  
    ]  
}
```

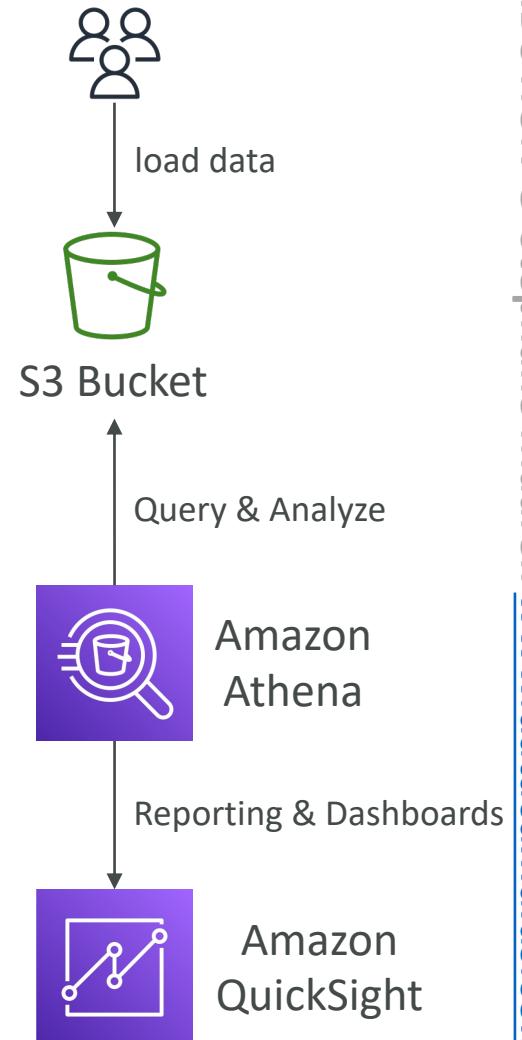
Allow **events** from another AWS account



# Amazon Athena



- Serverless query service to analyze data stored in Amazon S3
- Uses standard SQL language to query the files (built on Presto)
- Supports CSV, JSON, ORC, Avro, and Parquet
- Pricing: \$5.00 per TB of data scanned
- Commonly used with Amazon Quicksight for reporting/dashboards
- **Use cases:** Business intelligence / analytics / reporting, analyze & query VPC Flow Logs, ELB Logs, CloudTrail trails, etc...
- **Exam Tip:** analyze data in S3 using serverless SQL, use Athena

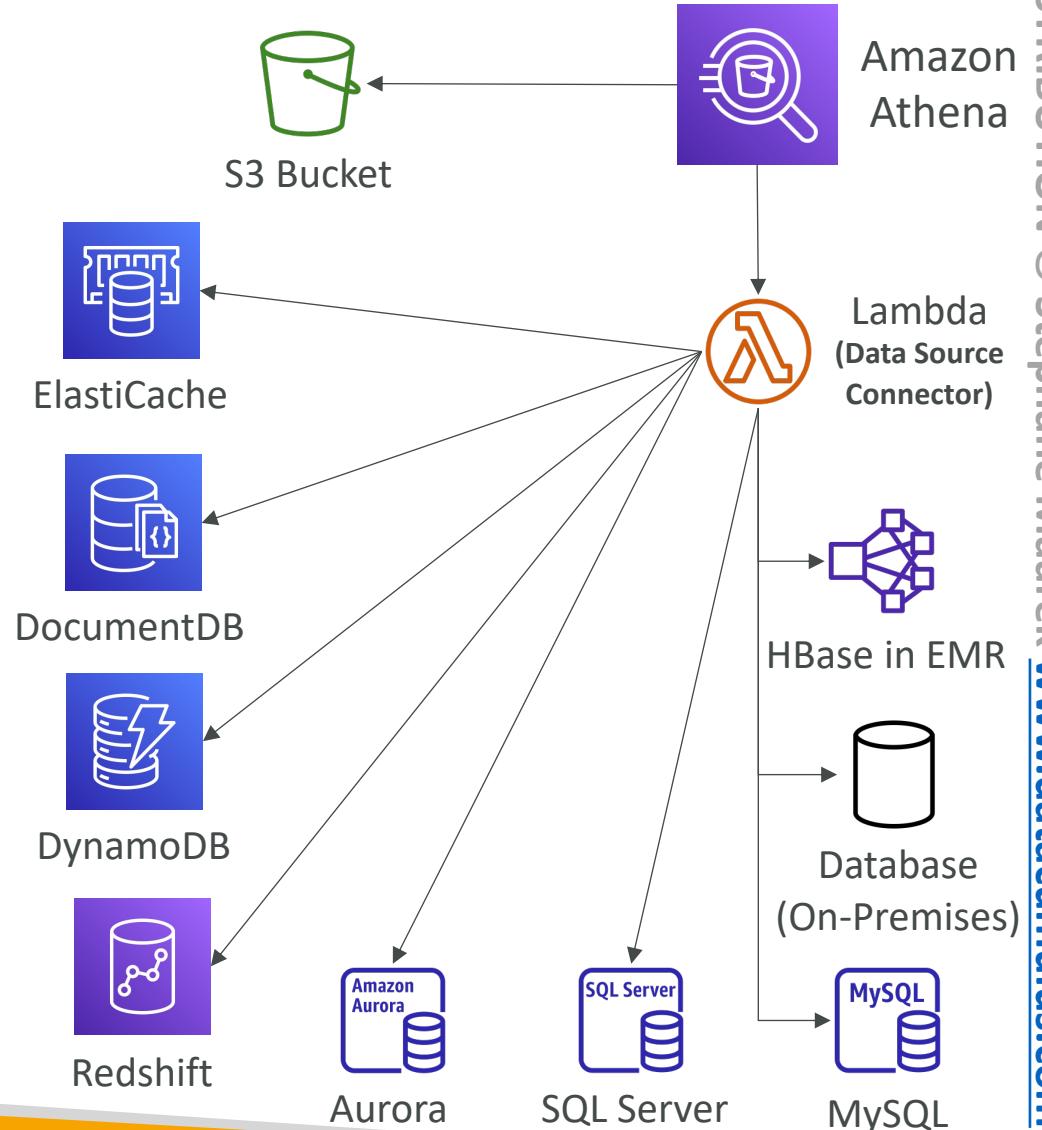


# Amazon Athena – Performance Improvement

- Use **columnar data** for cost-savings (less scan)
  - Apache Parquet or ORC is recommended
  - Huge performance improvement
  - Use Glue to convert your data your Parquet or ORC
- **Compress data** for smaller retrievals (bzip2, gzip, lz4, snappy, zlip, zstd...)
- **Partition** datasets in S3 for easy querying on virtual columns
  - s3://yourBucket/pathToTable  
  / <PARTITION\_COLUMN\_NAME>=<VALUE>  
  / <PARTITION\_COLUMN\_NAME>=<VALUE>  
  / <PARTITION\_COLUMN\_NAME>=<VALUE>  
  /etc...
  - Example: s3://athena-examples/flight/parquet/year=1991/month=1/day=1/
- Use **larger files** (> 128 MB) to minimize overhead

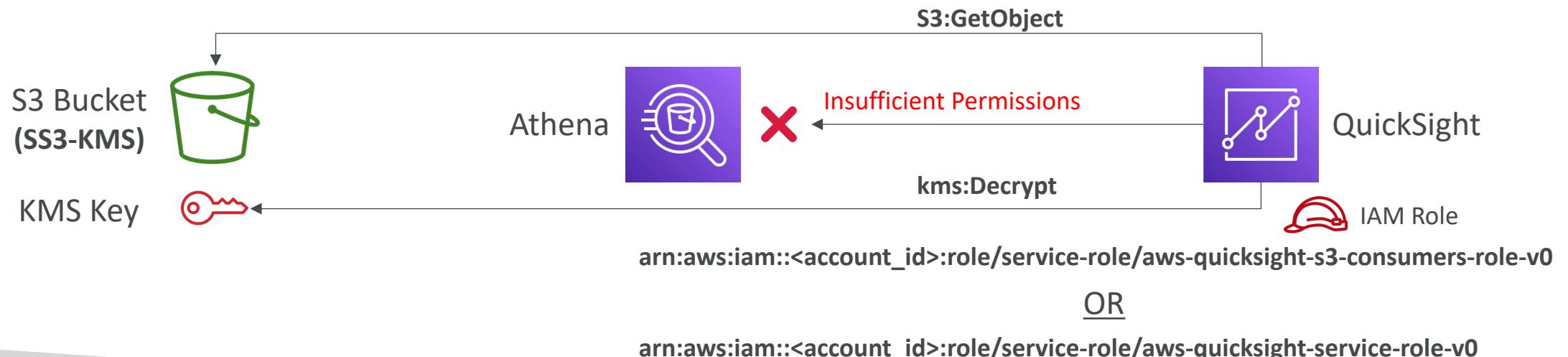
# Amazon Athena – Federated Query

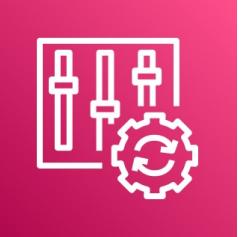
- Allows you to run SQL queries across data stored in relational, non-relational, object, and custom data sources (AWS or on-premises)
- Uses Data Source Connectors that run on AWS Lambda to run Federated Queries (e.g., CloudWatch Logs, DynamoDB, RDS, ...)
- Store the results back in Amazon S3



# Amazon Athena – Troubleshooting

- Insufficient Permissions When using Athena with QuickSight
  - Validate QuickSight can access S3 buckets used by Athena
  - If the data in the S3 buckets is encrypted using AWS KMS key (SSE-KMS), then QuickSight IAM role must be granted access to decrypt with the key (`kms:Decrypt`)
    - `arn:aws:iam::<account_id>:role/service-role/aws-quicksight-s3-consumers-role-v0` (Default)
    - `arn:aws:iam::<account_id>:role/service-role/aws-quicksight-service-role-v0`





# AWS Config

- Helps with auditing and recording **compliance** of your AWS resources
- Helps record configurations and changes over time
- Questions that can be solved by AWS Config:
  - Is there unrestricted SSH access to my security groups?
  - Do my buckets have any public access?
  - How has my ALB configuration changed over time?
- You can receive alerts (SNS notifications) for any changes
- AWS Config is a per-region service
- Can be aggregated across regions and accounts
- Possibility of storing the configuration data into S3 (analyzed by Athena)

# Config Rules

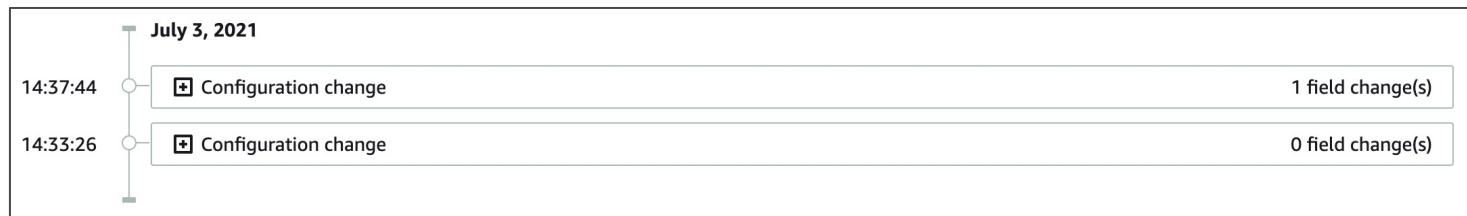
- Can use AWS managed config rules (over 75)
- Can make custom config rules (must be defined in AWS Lambda)
  - Ex: evaluate if each EBS disk is of type gp2
  - Ex: evaluate if each EC2 instance is t2.micro
- Rules can be evaluated / triggered:
  - For each config change
  - And / or: at regular time intervals
- AWS Config Rules does not prevent actions from happening (no deny)
- Pricing: no free tier, \$0.003 per configuration item recorded per region, \$0.001 per config rule evaluation per region

# AWS Config Resource

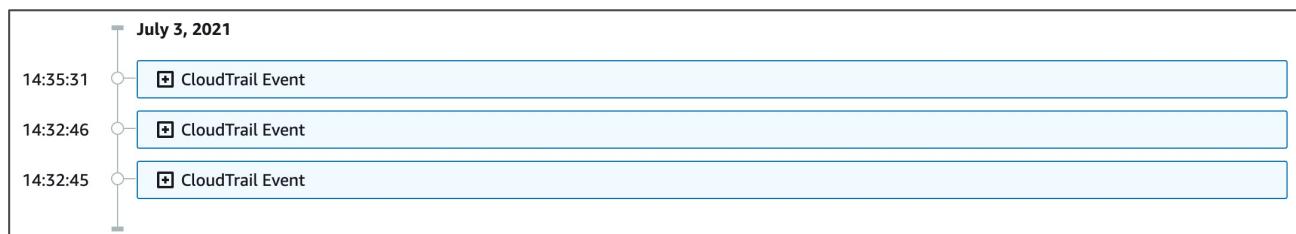
- View compliance of a resource over time

<input type="radio"/> sg-077b425b1649da83e	EC2 SecurityGroup	 Compliant
<input type="radio"/> sg-0831434f1876c0c74	EC2 SecurityGroup	 Noncompliant
<input type="radio"/> sg-09f10ed254d464f30	EC2 SecurityGroup	 Compliant

- View configuration of a resource over time

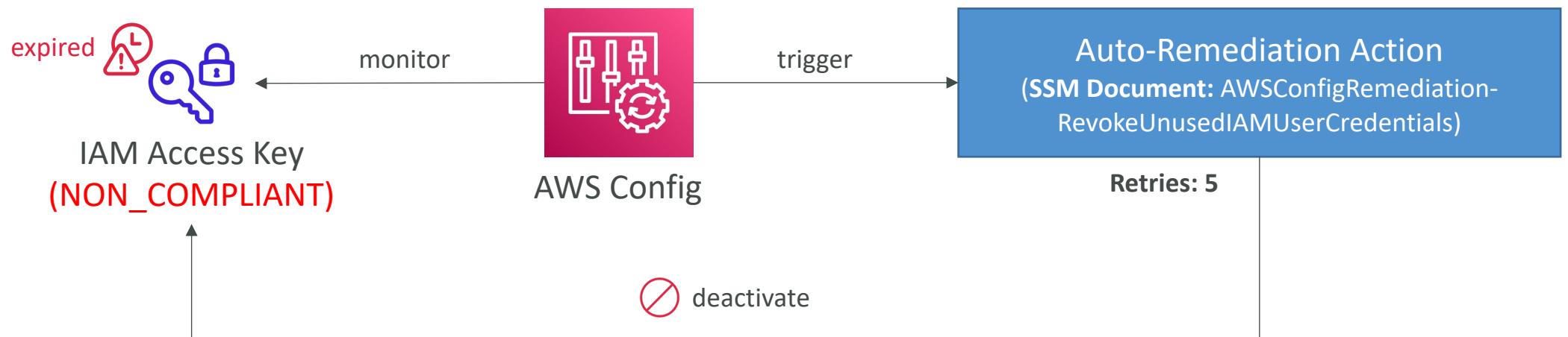


- View CloudTrail API calls of a resource over time



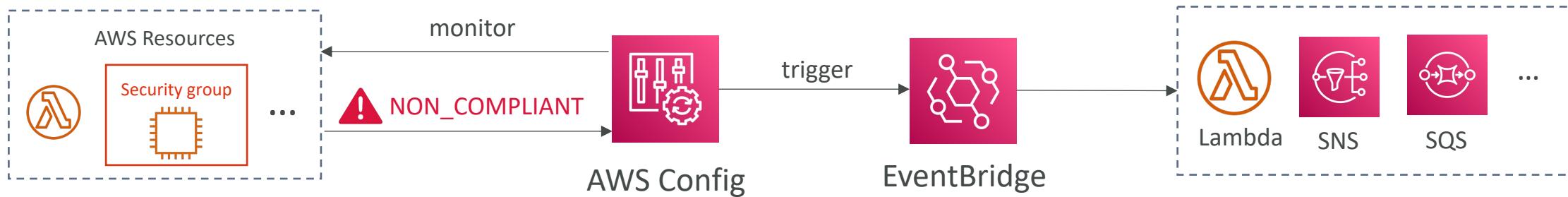
# Config Rules – Remediations

- Automate remediation of non-compliant resources using SSM Automation Documents
- Use AWS-Managed Automation Documents or create custom Automation Documents
  - Tip: you can create custom Automation Documents that invokes Lambda function
- You can set **Remediation Retries** if the resource is still non-compliant after auto-remediation

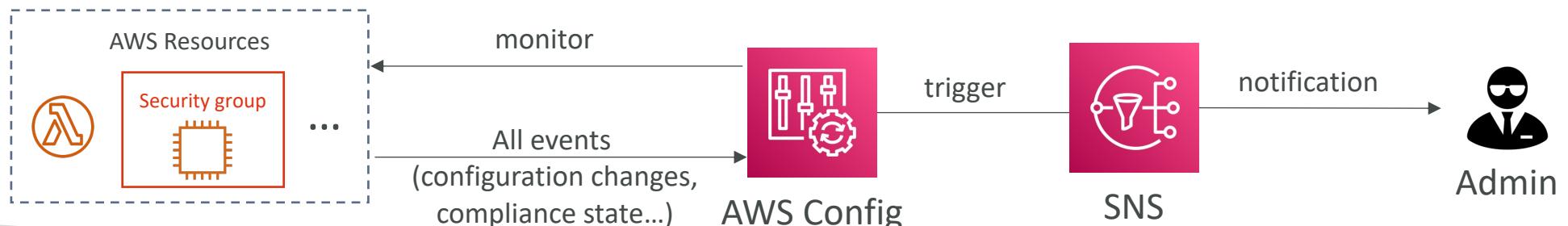


# Config Rules – Notifications

- Use EventBridge to trigger notifications when AWS resources are non-compliant

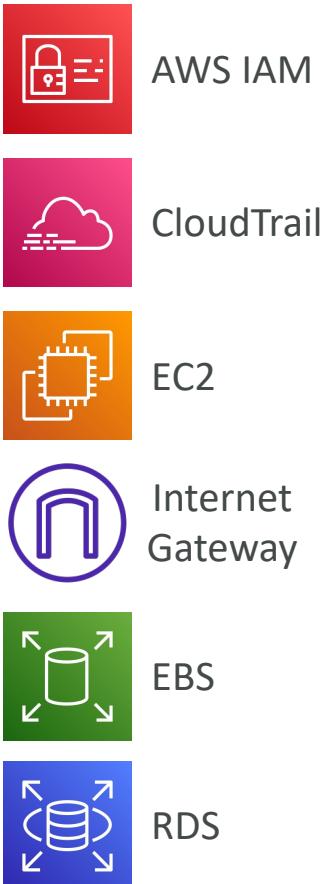


- Ability to send configuration changes and compliance state notifications to SNS (all events – use SNS Filtering or filter at client-side)

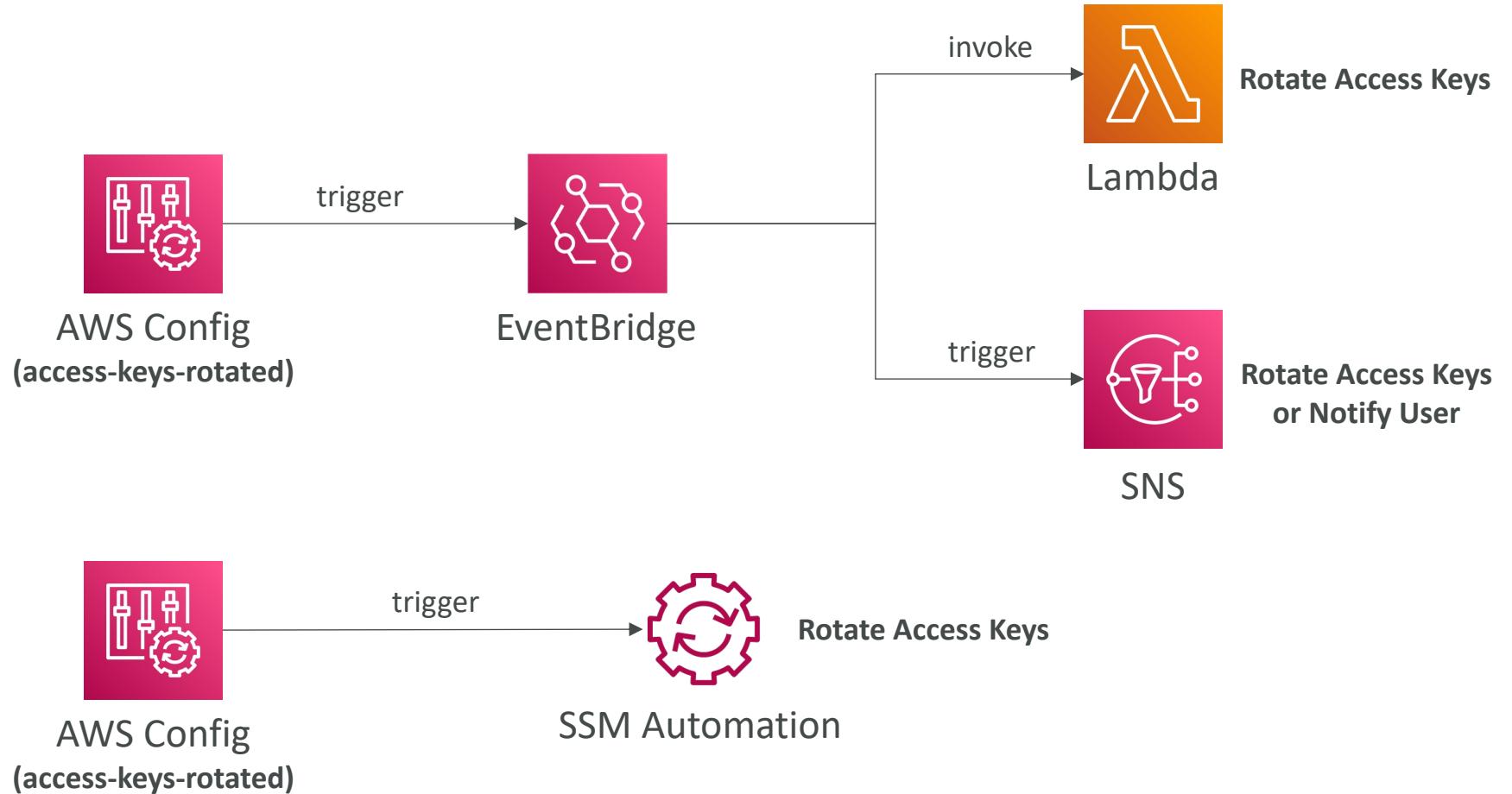


# AWS Config – Use Cases

- Audit IAM Policies
- Detect if CloudTrail has been disabled
- Detect if EC2 instances are created with unapproved AMIs
- Detect if Security Groups are open to the public
- Detect if Internet Gateway is added to unauthorized VPC
- Detect if EBS volumes are encrypted
- Detect if RDS databases are public



# AWS Config – Example: Expired IAM Keys





# Trusted Advisor

- No need to install anything – high level AWS account assessment
- Analyze your AWS accounts and provides recommendation:

**Cost Optimization**



**Performance**



**Security**



**Fault Tolerance**



**Service Limits**



- Core Checks and recommendations – all customers
- Can enable weekly email notification from the console
- Full Trusted Advisor – Available for **Business & Enterprise** support plans
  - Ability to set CloudWatch alarms when reaching limits
  - Programmatic Access using AWS Support API



# Trusted Advisor Checks Examples

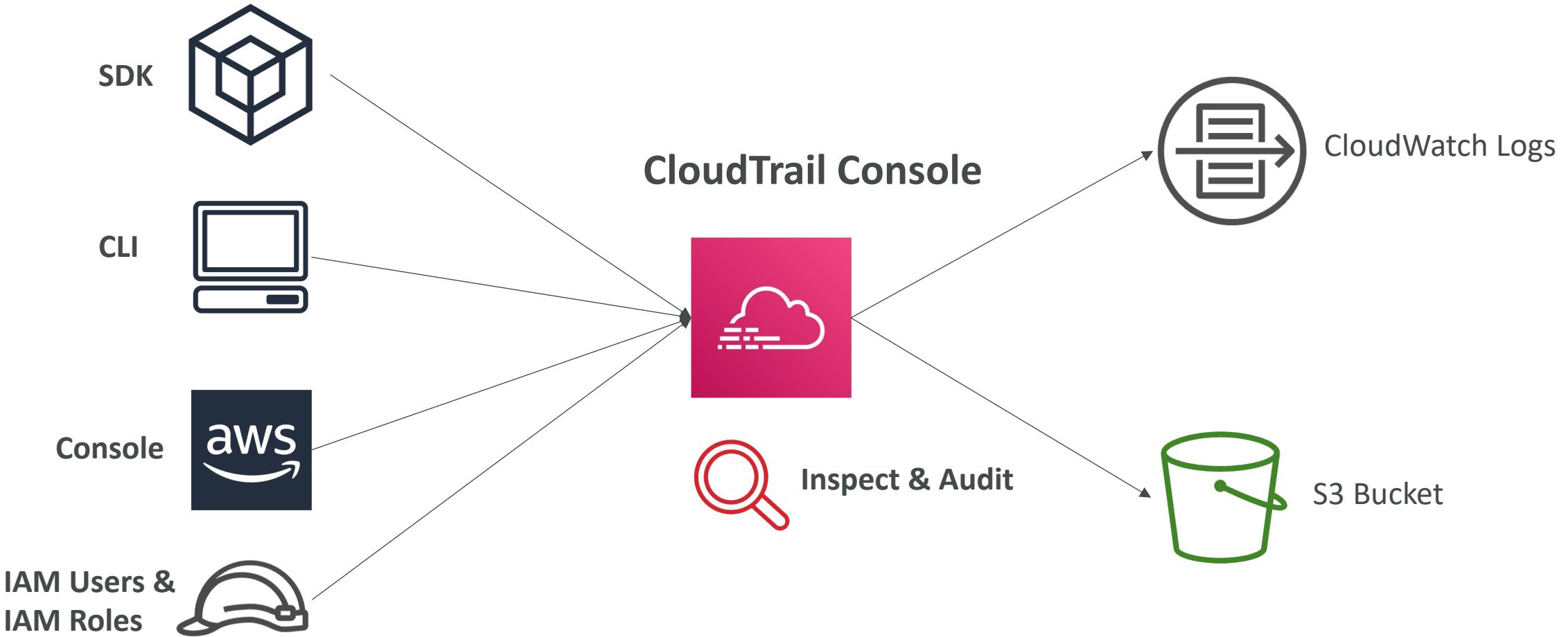
- Cost Optimization:
  - low utilization EC2 instances, idle load balancers, under-utilized EBS volumes...
  - Reserved instances & savings plans optimizations,
- Performance:
  - High utilization EC2 instances, CloudFront CDN optimizations
  - EC2 to EBS throughput optimizations, Alias records recommendations
- Security:
  - MFA enabled on Root Account, IAM key rotation, exposed Access Keys
  - S3 Bucket Permissions for public access, security groups with unrestricted ports
- Fault Tolerance:
  - EBS snapshots age, Availability Zone Balance
  - ASG Multi-AZ, RDS Multi-AZ, ELB configuration...
- Service Limits



# AWS CloudTrail

- Provides governance, compliance and audit for your AWS Account
- CloudTrail is enabled by default!
- Get an history of events / API calls made within your AWS Account by:
  - Console
  - SDK
  - CLI
  - AWS Services
- Can put logs from CloudTrail into CloudWatch Logs or S3
- A trail can be applied to All Regions (default) or a single Region.
- If a resource is deleted in AWS, investigate CloudTrail first!

# CloudTrail Diagram





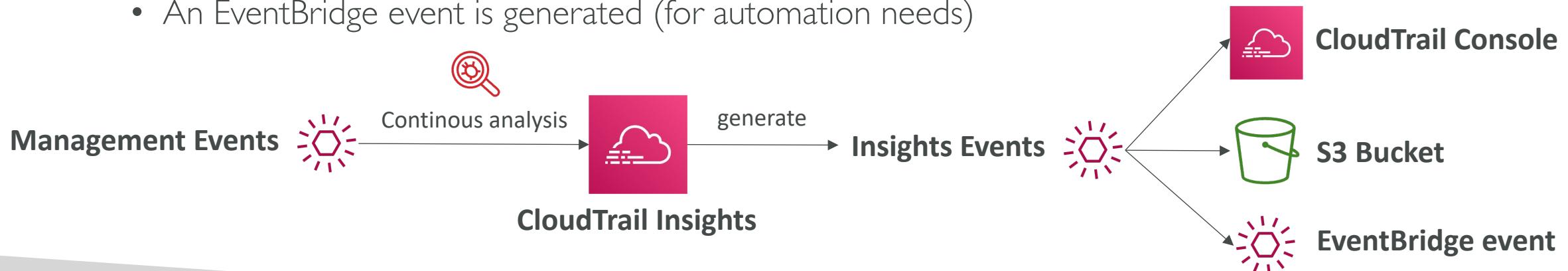
# CloudTrail Events

- Management Events:
  - Operations that are performed on resources in your AWS account
  - Examples:
    - Configuring security (IAM `AttachRolePolicy`)
    - Configuring rules for routing data (Amazon EC2 `CreateSubnet`)
    - Setting up logging (AWS CloudTrail `CreateTrail`)
  - By default, trails are configured to log management events.
  - Can separate Read Events (that don't modify resources) from Write Events (that may modify resources)
- Data Events:
  - By default, data events are not logged (because high volume operations)
  - Amazon S3 object-level activity (ex: `GetObject`, `DeleteObject`, `PutObject`): can separate Read and Write Events
  - AWS Lambda function execution activity (the `Invoke` API)
- CloudTrail Insights Events:
  - See next slide ☺



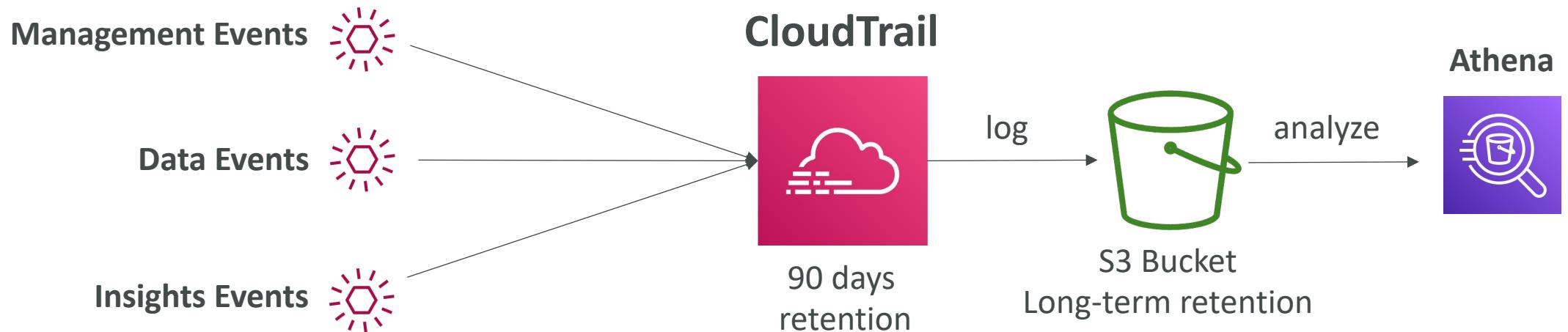
# CloudTrail Insights

- Enable CloudTrail Insights to detect unusual activity in your account:
  - inaccurate resource provisioning
  - hitting service limits
  - Bursts of AWS IAM actions
  - Gaps in periodic maintenance activity
- CloudTrail Insights analyzes normal management events to create a baseline
- And then continuously analyzes write events to detect unusual patterns
  - Anomalies appear in the CloudTrail console
  - Event is sent to Amazon S3
  - An EventBridge event is generated (for automation needs)

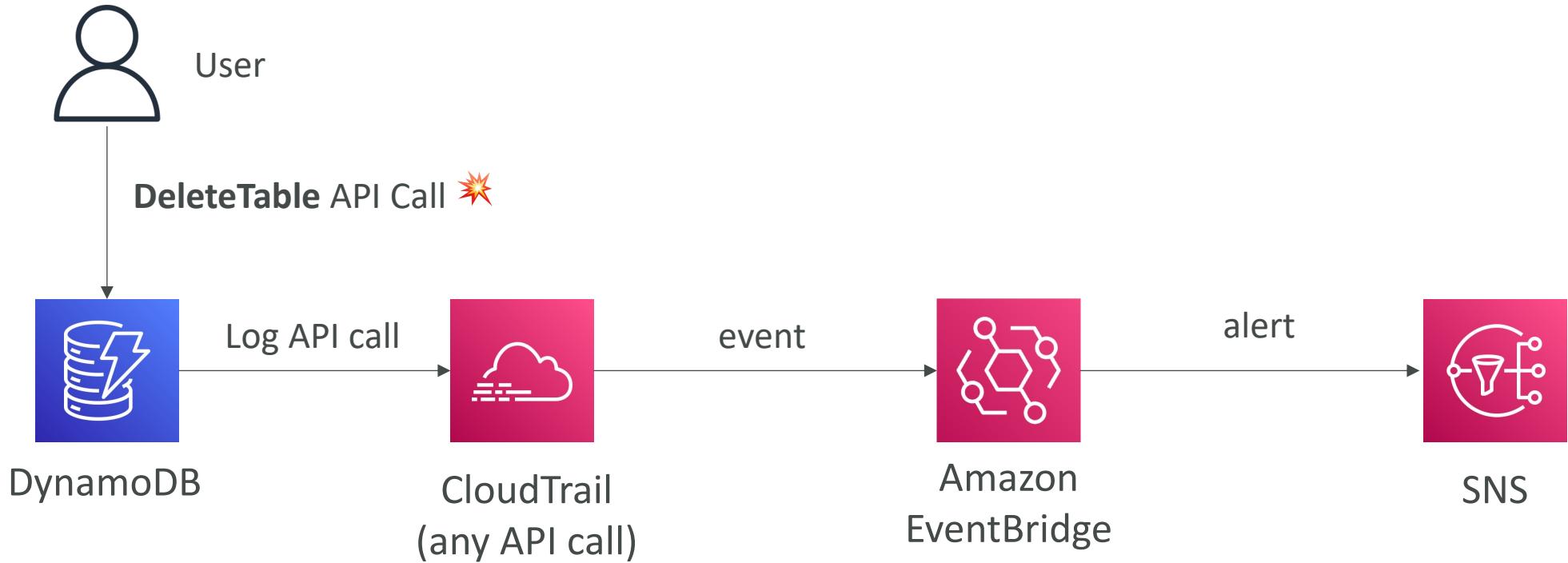


# CloudTrail Events Retention

- Events are stored for 90 days in CloudTrail
- To keep events beyond this period, log them to S3 and use Athena

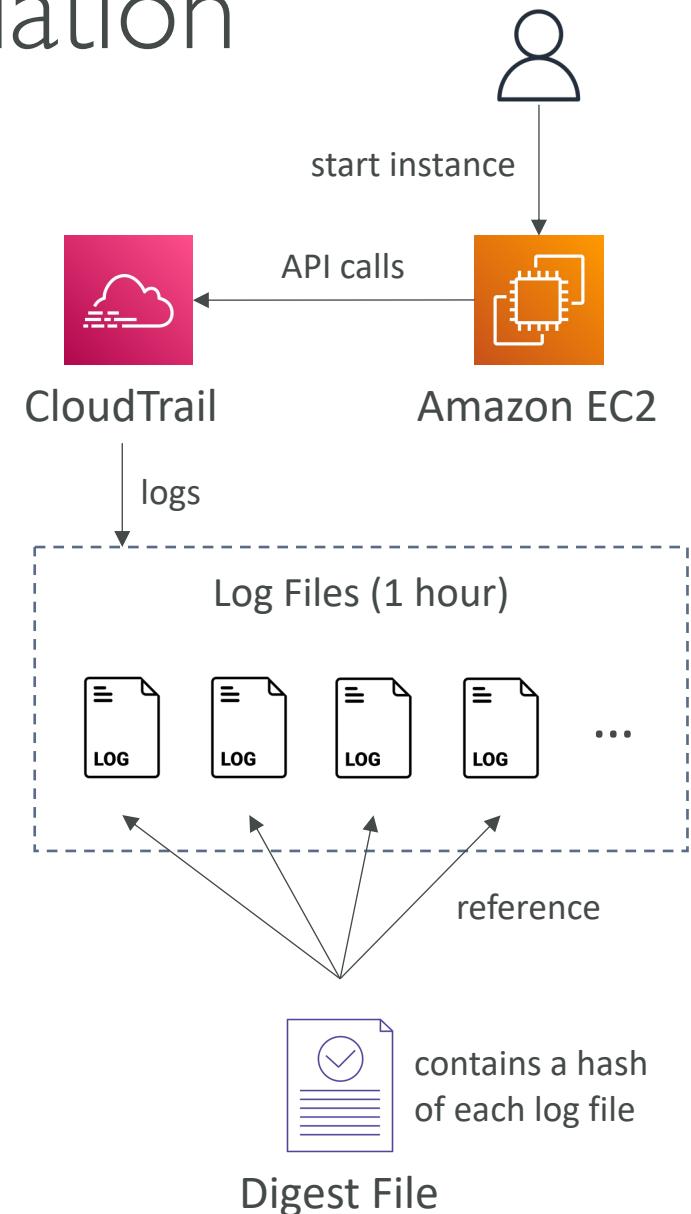


# Amazon EventBridge – Intercept API Calls

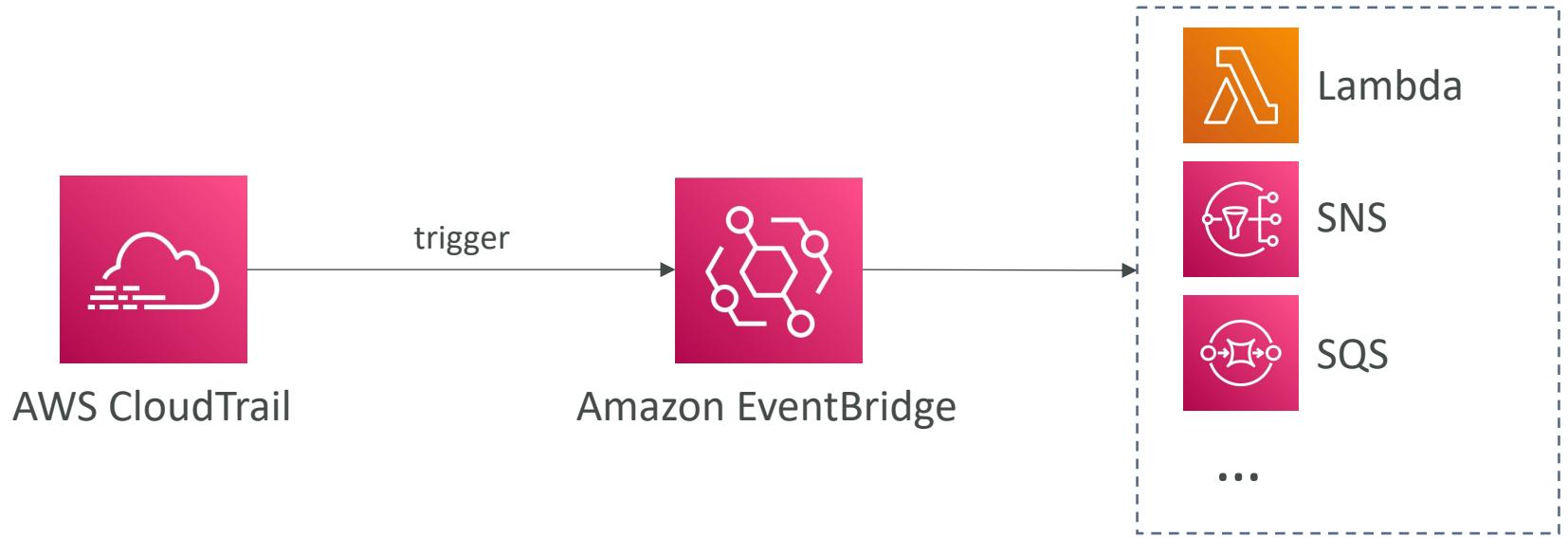


# CloudTrail – Log File Integrity Validation

- **Digest Files:**
  - References the log files for the last hour and contains a hash of each
  - Stored in the same S3 bucket as log files (different folder)
- Helps you determine whether a log file was modified/deleted after CloudTrail delivered it
- Hashing using SHA-256, Digital Signing using SHA-256 with RSA
- Protect the S3 bucket using bucket policy, versioning, MFA Delete protection, encryption, object lock
- Protect CloudTrail using IAM



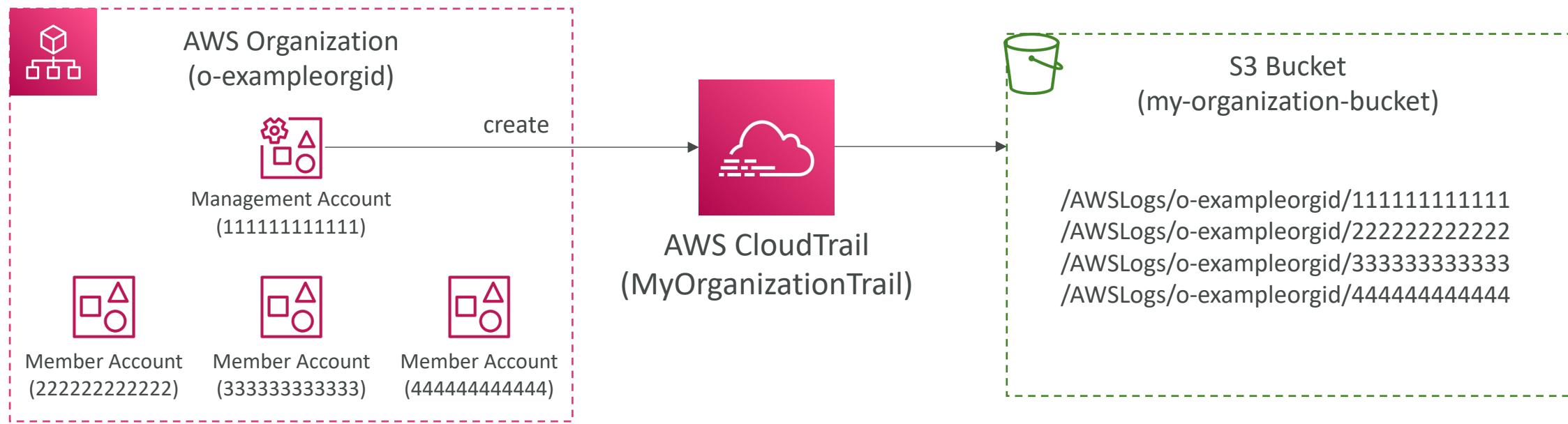
# CloudTrail – Integration with EventBridge



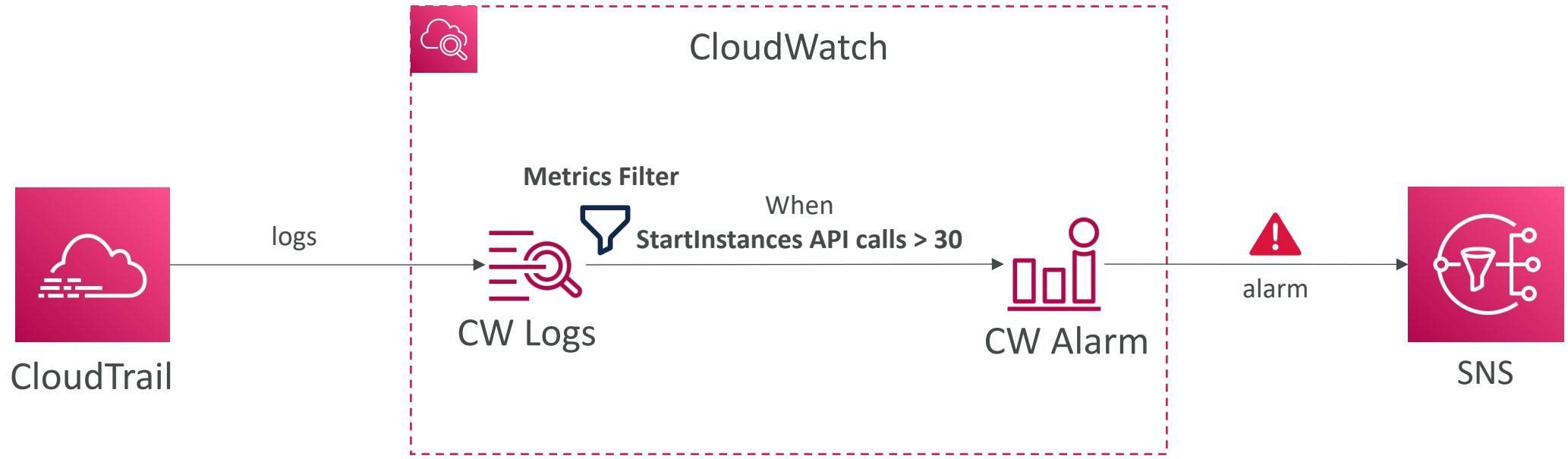
- Used to react to any API call being made in your account
- CloudTrail is not “real-time”:
  - Delivers an event within 15 minutes of an API call
  - Delivers log files to an S3 bucket every 5 minutes

# CloudTrail – Organizations Trails

- A trail that will log all events for all AWS accounts in an AWS Organization
- Log events for management and member accounts
- Trail with the same name will be created in every AWS account (IAM permissions)
- Member accounts can't remove or modify the organization trail (view only)



# CloudTrail to CloudWatch Metrics Filter



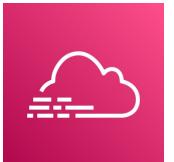
**Advantage: ability to be able to set thresholds for detection**

# Monitor Account Activity

- AWS Config Configuration History
  - Must have AWS Config Configuration Recorder on
- CloudTrail Event History
  - Search API history for past 90 days
  - Filter by resource name, resource type, event name, ...
  - Filter by IAM user, assumed IAM role session name, or AWS Access Key
- CloudWatch Logs Insights
  - Search API history beyond the past 90 days
  - CloudTrail Trail must be configured to send logs to CloudWatch Logs
- Athena Queries
  - Search API history beyond the past 90 days



AWS Config



CloudTrail

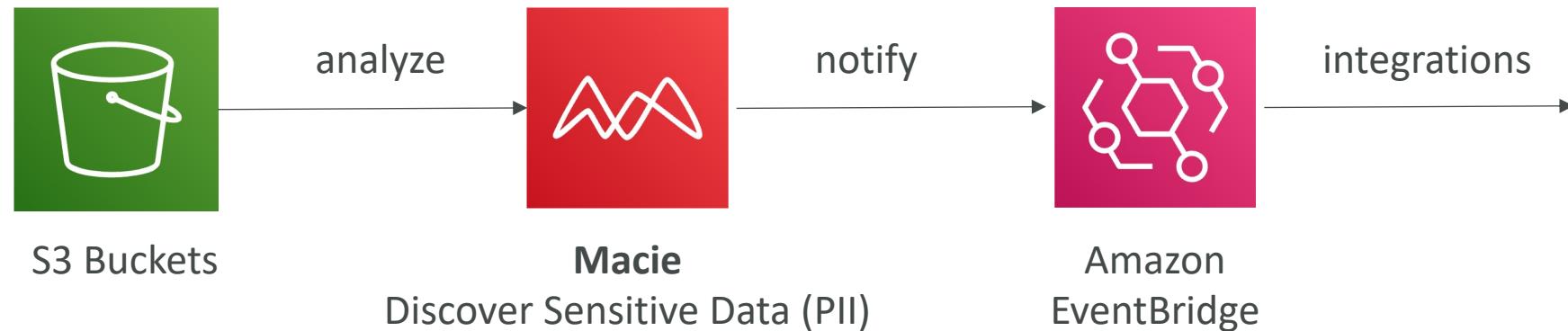
CloudWatch Logs  
Insights

Athena

# AWS Macie

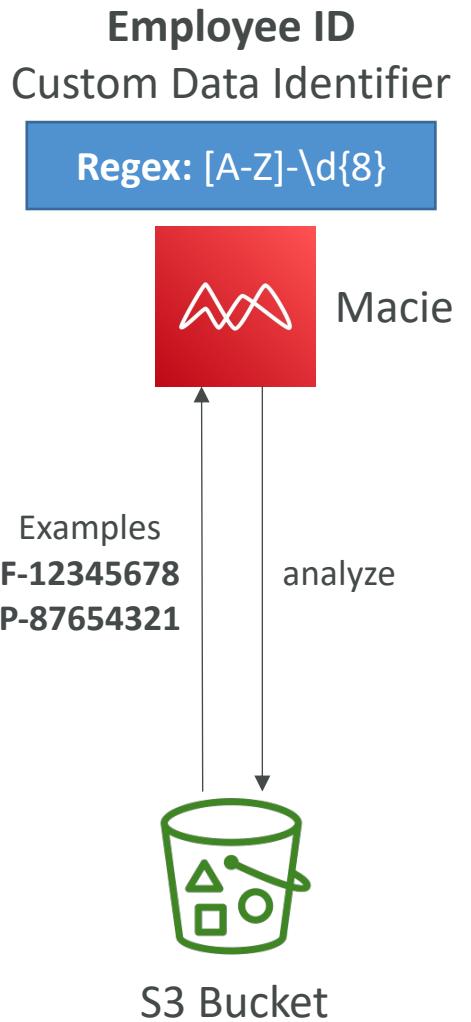


- Amazon Macie is a fully managed data security and data privacy service that uses machine learning and pattern matching to discover and protect your sensitive data in AWS.
- Macie helps identify and alert you to sensitive data, such as personally identifiable information (PII)



# AWS Macie – Data Identifiers

- Used to analyze and identify sensitive data in your S3 buckets
- **Managed Data Identifier**
  - A set of built-in criteria that are designed to detect specific type of sensitive data
  - Examples: credit cards numbers, AWS Credentials, bank accounts
- **Custom Data Identifier**
  - A set of criteria that you define to detect sensitive data
  - Regular expression, keywords, proximity rule
  - Examples: employee IDs, customer account numbers
- You can use **Allow Lists** to define a text pattern to ignore (e.g., public phone numbers)



# AWS Macie – Findings

- A report of a potential issue or sensitive data that Macie found
- Each finding has a severity rating, affected resource, datetime, ...
- **Sensitive Data Discovery Result**
  - A record that logs details about the analysis of an S3 object
  - Configure Macie to store the results in S3, then query using Athena
- **Suppression Rules** – set of attribute-based filter criteria to archive findings automatically
- Findings are stored for 90 days
- Review findings using AWS Console, EventBridge, Security Hub

# AWS Macie – Findings Types

- **Policy Findings**

- A detailed report of policy violation or issue with the security of S3 bucket
- Examples: default encryption is disabled, bucket is public, ...
- **Policy:IAMUser/S3BucketEncryptionDisabled, Policy:IAMUser/S3BucketPublic**
- Detect changes only after you enable Macie

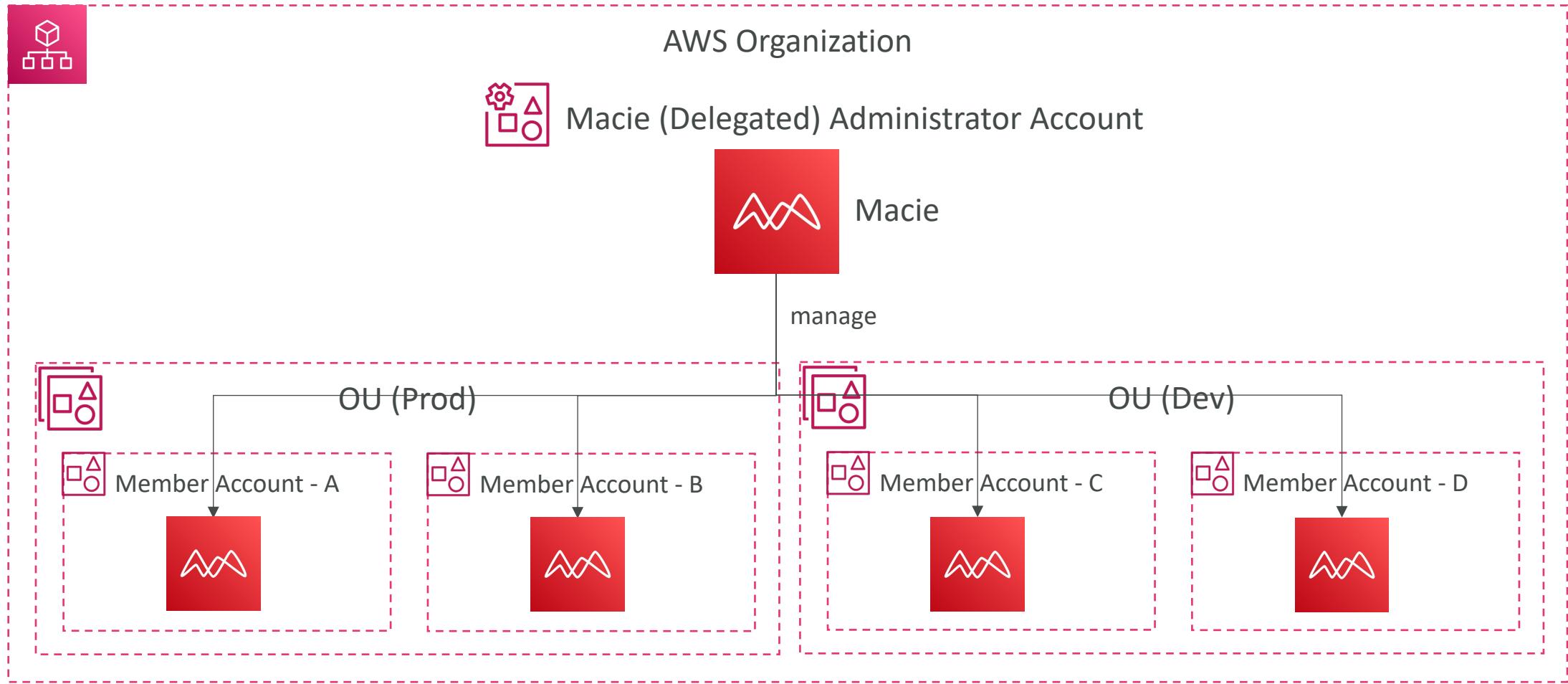
- **Sensitive Data Findings**

- A detailed report of sensitive data that's found in S3 buckets
- Examples: Credentials (private keys), Financial (credit card numbers), ...
- **SensitiveData:S3Object/Credentials, SensitiveData:S3Object/Financial**
- For Custom Data Identifier **SensitiveData:S3Object/CustomIdentifier**

# AWS Macie – Multi-Account Strategy

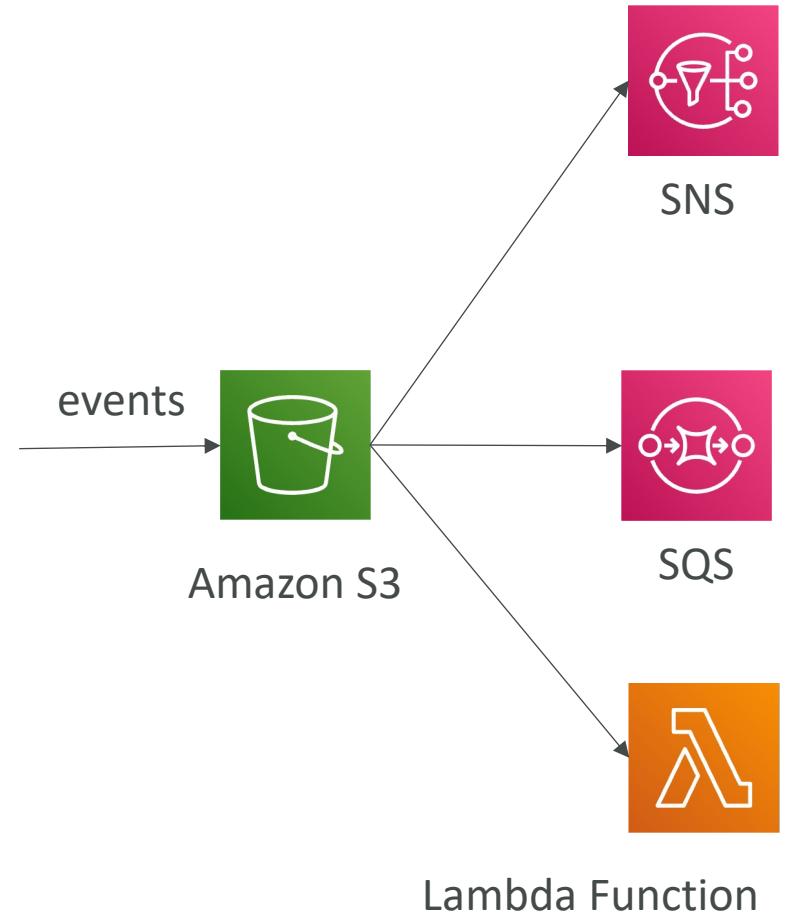
- You can manage multiple accounts in Macie
- Associate the Member accounts with the Administrator account
  - Through an AWS Organization
  - Sending invitation through Macie
  - Supports Delegated Administrator in an AWS Organization
- Administrator account can:
  - Add and remove member accounts
  - Have access to all S3 sensitive data and settings for all accounts
  - Manage Automated Sensitive Data Discovery and run Data Discovery jobs
  - Manage Data Identifiers and Findings

# AWS Macie – Multi-Account Strategy

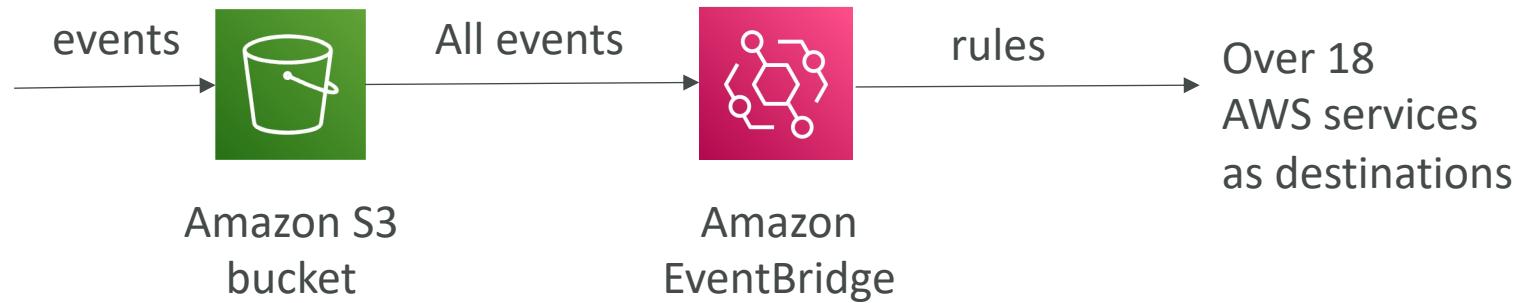


# S3 Event Notifications

- S3:ObjectCreated, S3:ObjectRemoved, S3:ObjectRestore, S3:Replication...
- Object name filtering possible (\*.jpg)
- Use case: generate thumbnails of images uploaded to S3
- Can create as many “S3 events” as desired
- S3 event notifications typically deliver events in seconds but can sometimes take a minute or longer



# S3 Event Notifications with Amazon EventBridge



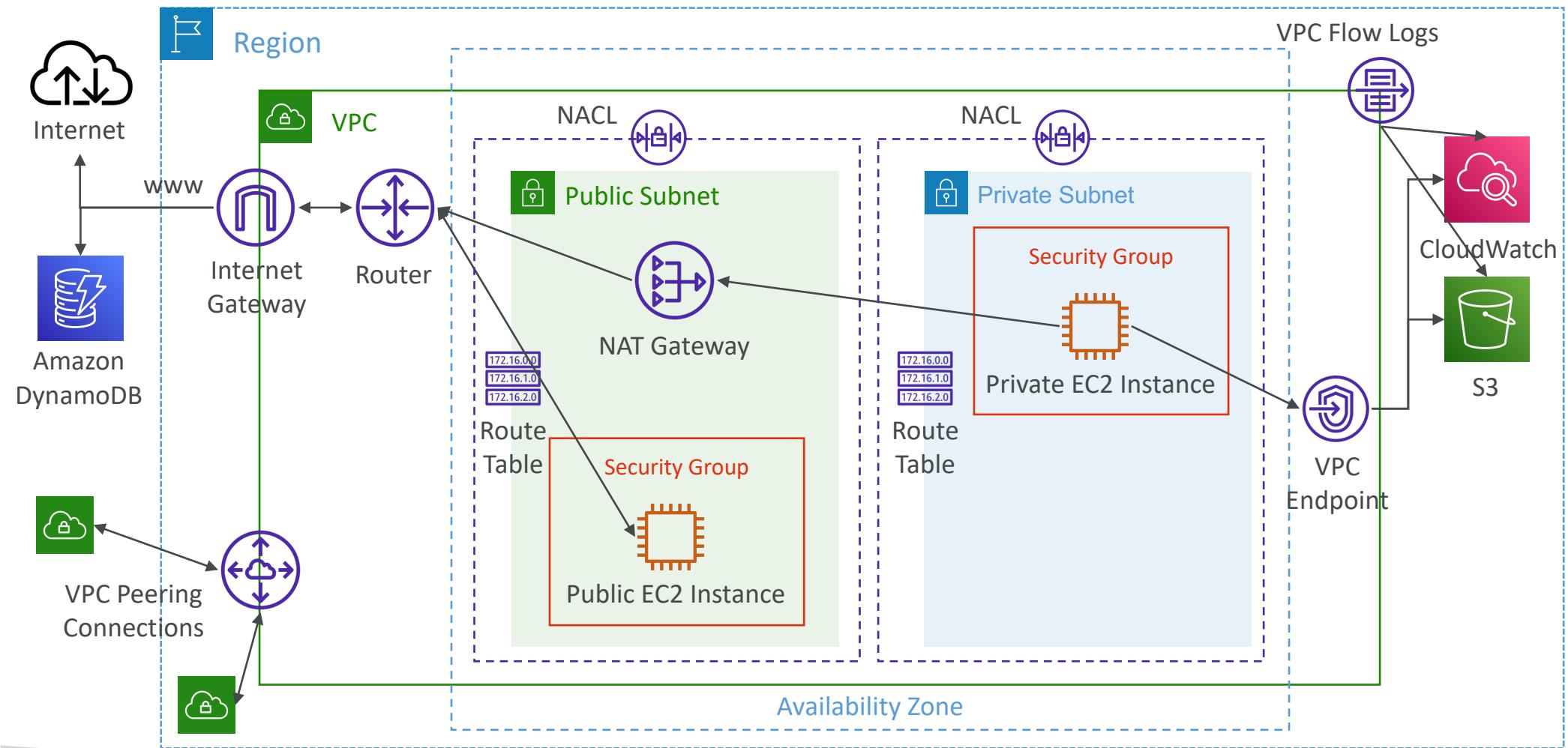
- Advanced filtering options with JSON rules (metadata, object size, name...)
- Multiple Destinations – ex Step Functions, Kinesis Streams / Firehose...
- EventBridge Capabilities – Archive, Replay Events, Reliable delivery



# VPC Flow Logs

- Capture information about IP traffic going into your interfaces:
  - VPC Flow Logs
  - Subnet Flow Logs
  - Elastic Network Interface (ENI) Flow Logs
- Helps to monitor & troubleshoot connectivity issues
- Flow logs data can go to S3 / CloudWatch Logs
- Captures network information from AWS managed interfaces too: ELB, RDS, ElastiCache, Redshift, WorkSpaces, NATGW, Transit Gateway...

# VPC Flow Logs



# VPC Flow Logs Syntax

version	interface-id	dstaddr	dstport	packets	start	action
2	123456789010	eni-1235b8ca123456789	172.31.16.139	172.31.16.21	20641	ACCEPT OK
2	123456789010	eni-1235b8ca123456789	172.31.9.69	172.31.9.12	49761	REJECT OK
account-id	srcaddr	srcport	protocol	bytes	end	log-status

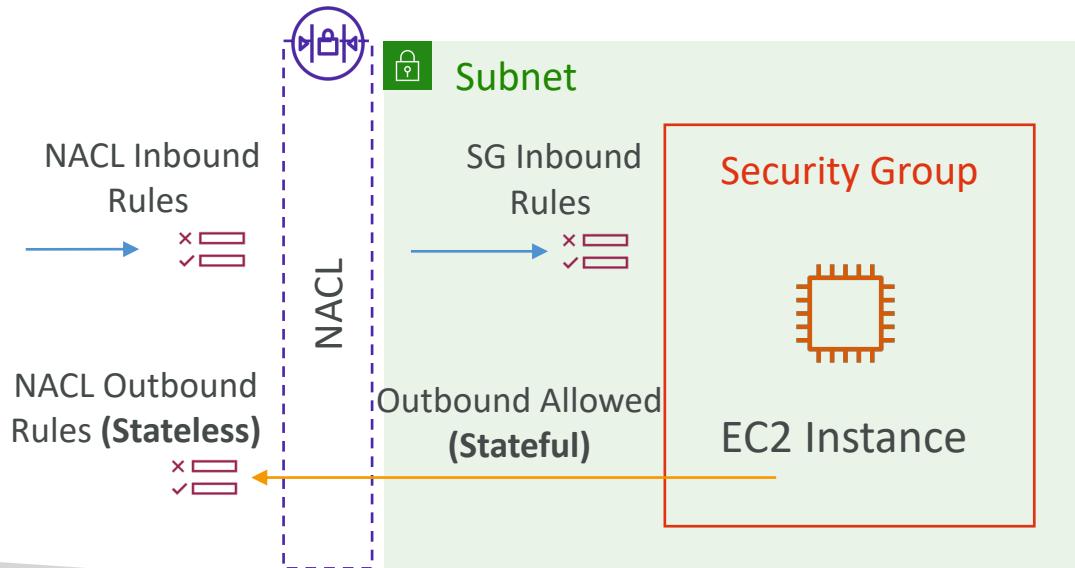
- srcaddr & dstaddr – help identify problematic IP
- srcport & dstport – help identify problematic ports
- Action – success or failure of the request due to Security Group / NACL
- Can be used for analytics on usage patterns, or malicious behavior
- Query VPC flow logs using Athena on S3 or CloudWatch Logs Insights
- Flow Logs examples: <https://docs.aws.amazon.com/vpc/latest/userguide/flow-logs-records-examples.html>

# VPC Flow Logs – Troubleshoot SG & NACL issues

Look at the “ACTION” field

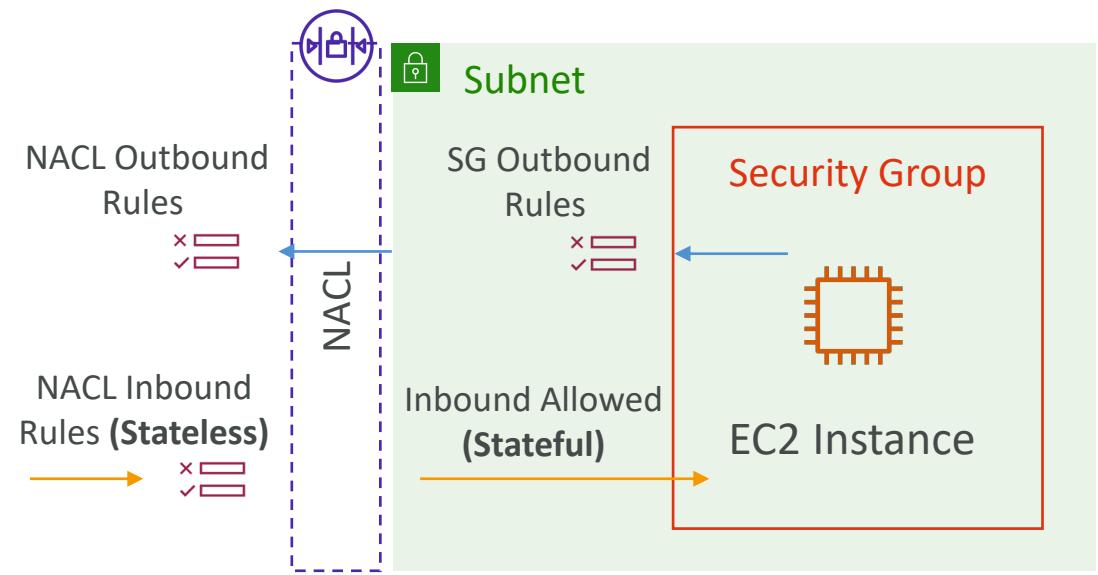
## Incoming Requests

- Inbound REJECT => NACL or SG
- Inbound ACCEPT, Outbound REJECT => NACL

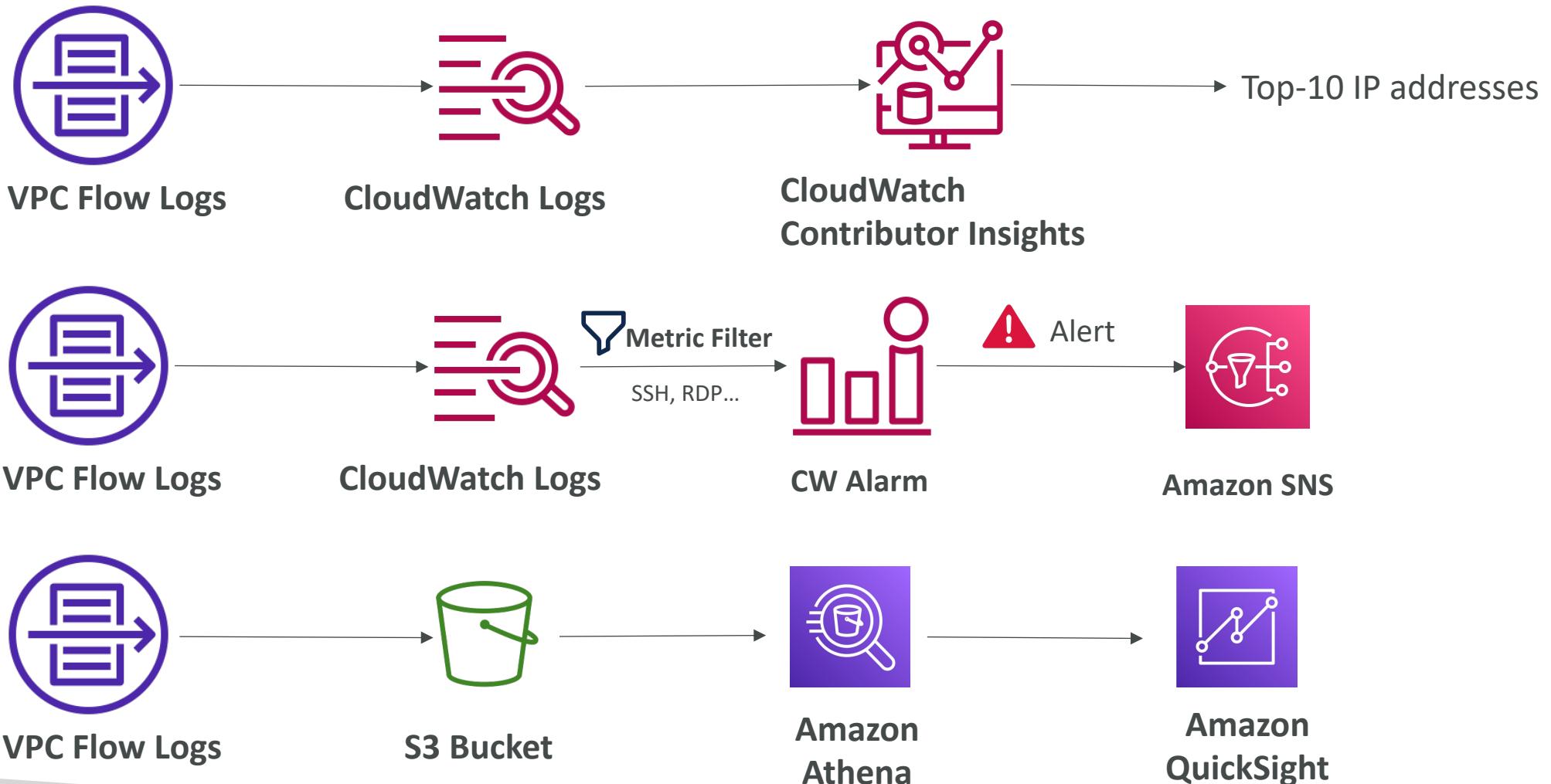


## Outgoing Requests

- Outbound REJECT => NACL or SG
- Outbound ACCEPT, Inbound REJECT => NACL



# VPC Flow Logs – Architectures

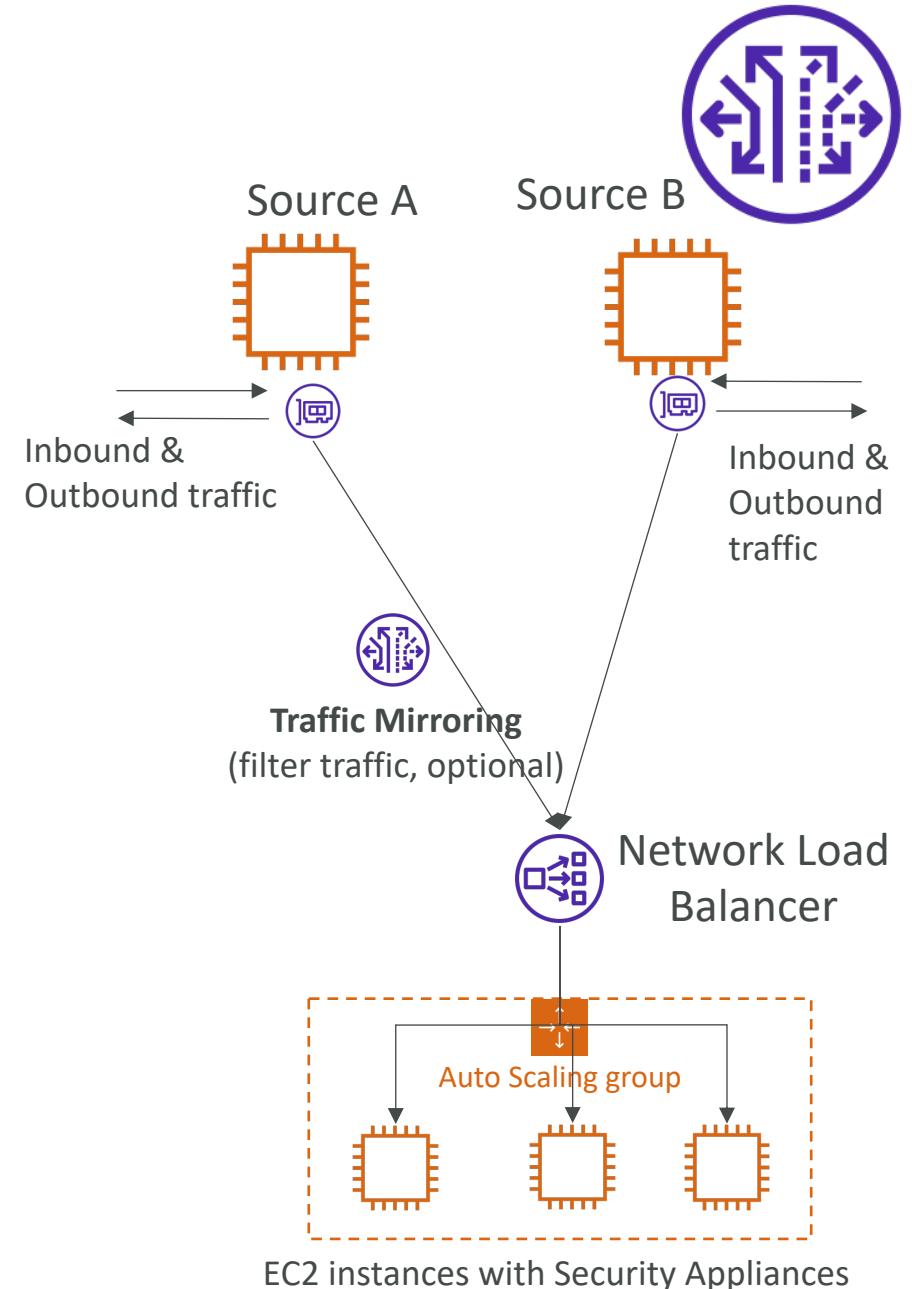


# VPC Flow Logs – Traffic not captured

- Traffic to Amazon DNS server (custom DNS server traffic is logged)
- Traffic for Amazon Windows license activation
- Traffic to and from 169.254.169.254 for EC2 instance metadata
- Traffic to and from 169.254.169.123 for Amazon Time Sync service
- DHCP traffic
- Mirrored traffic
- Traffic to the VPC router reserved IP address (e.g., 10.0.0.1)
- Traffic between VPC Endpoint ENI and Network Load Balancer ENI

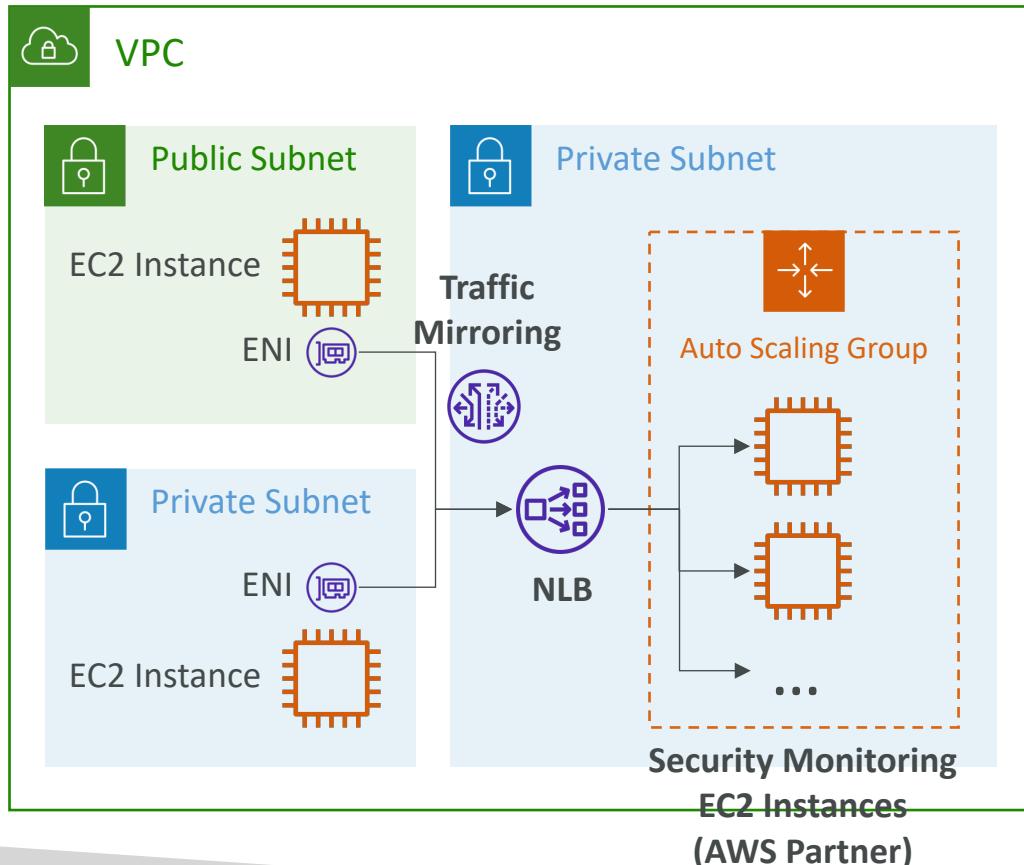
# VPC – Traffic Mirroring

- Allows you to capture and inspect network traffic in your VPC
- Route the traffic to security appliances that you manage
- Capture the traffic
  - From (Source) – ENIs
  - To (Targets) – an ENI or a Network Load Balancer
- Capture all packets or capture the packets of your interest (optionally, truncate packets)
- Source and Target can be in the same VPC or different VPCs (VPC Peering)
- Use cases: content inspection, threat monitoring, troubleshooting, ...

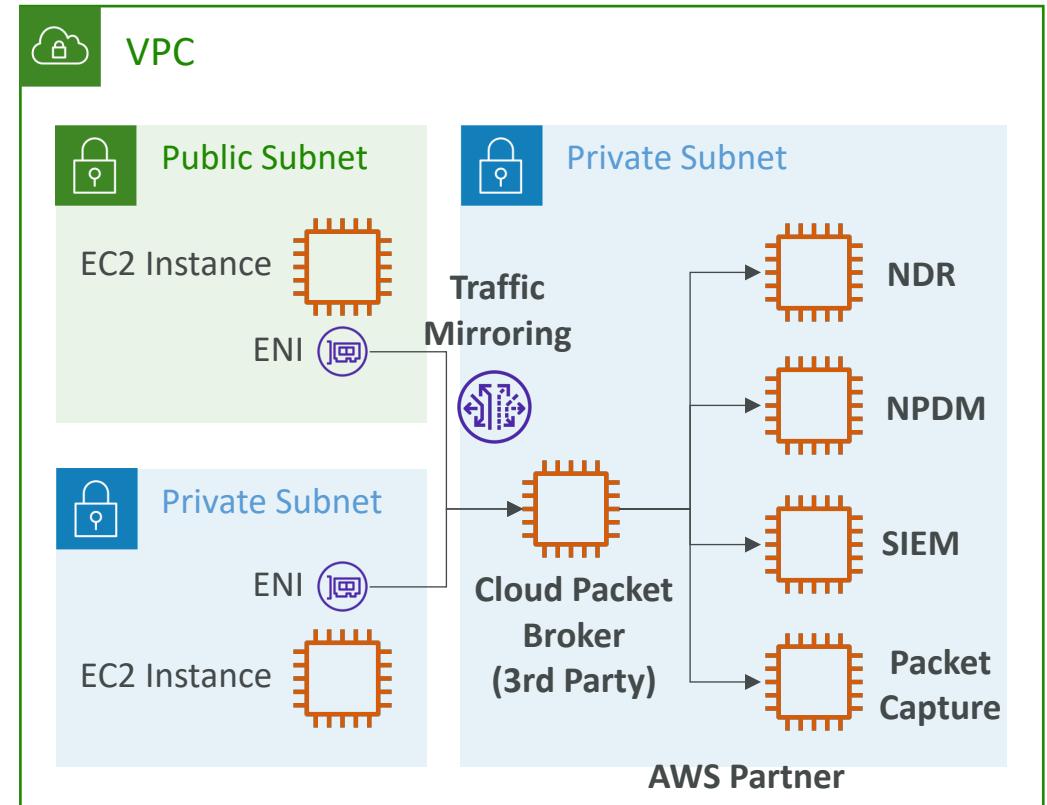


# VPC Traffic Mirroring – Architecture

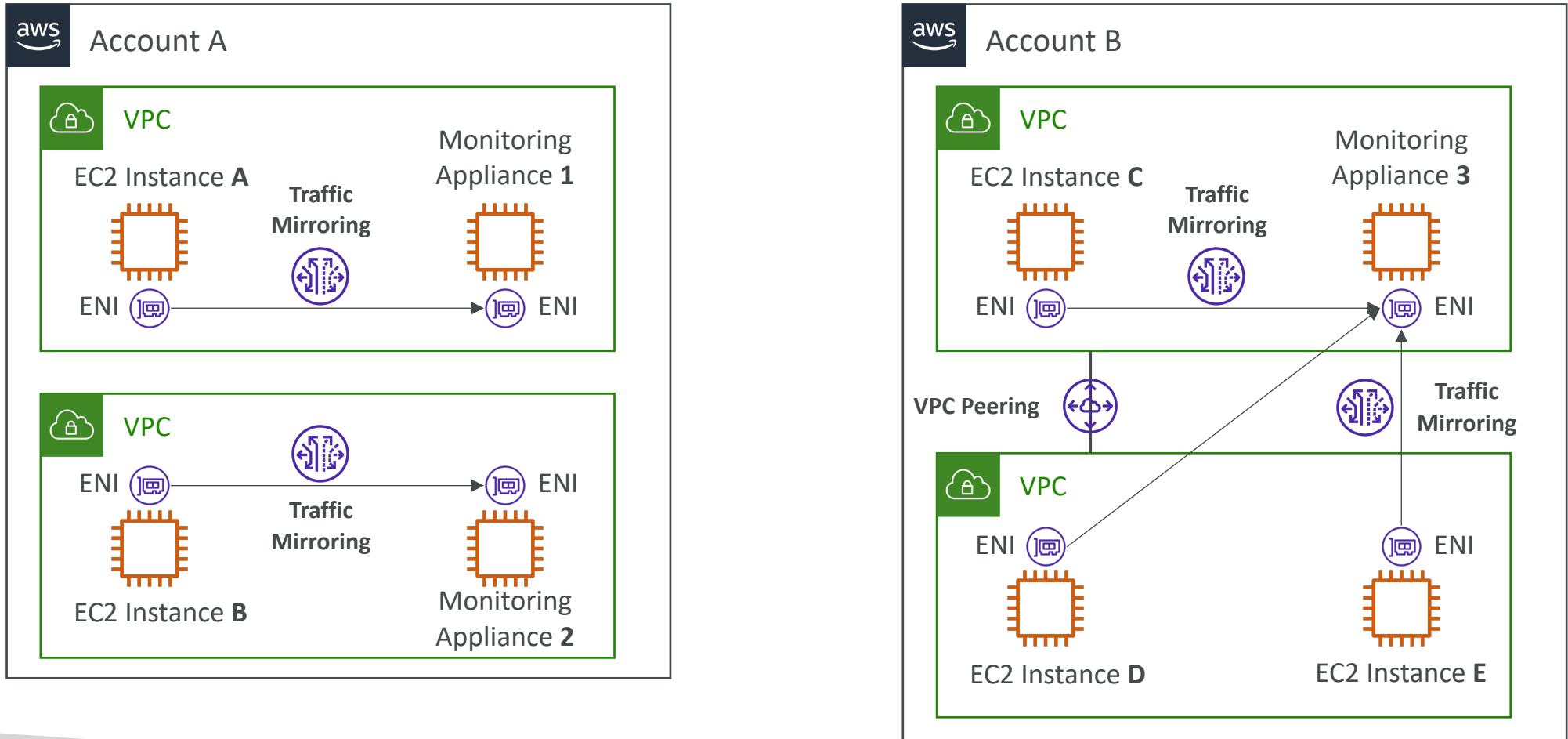
Traffic is distributed across EC2 instances  
in ASG (same security appliance)



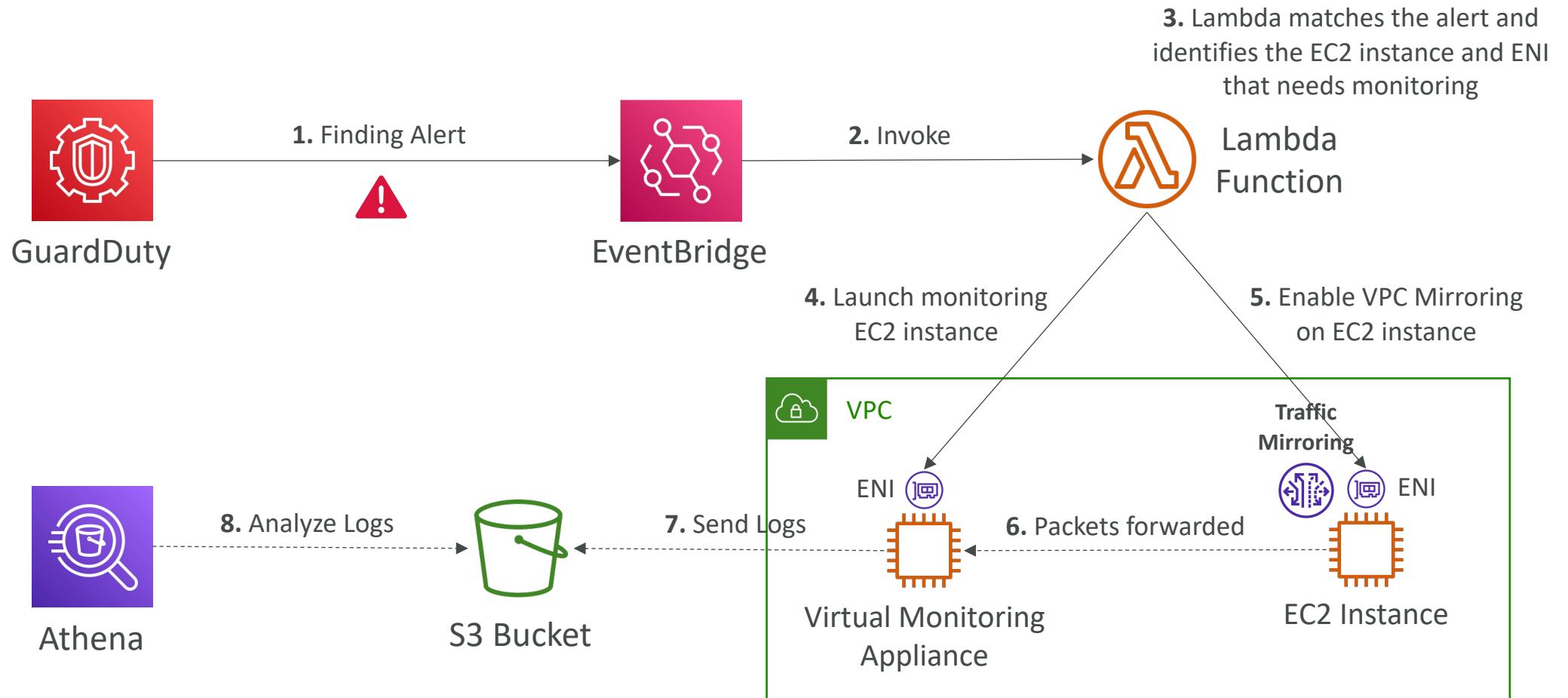
All Traffic is sent to multiple EC2 instances  
with different security appliances



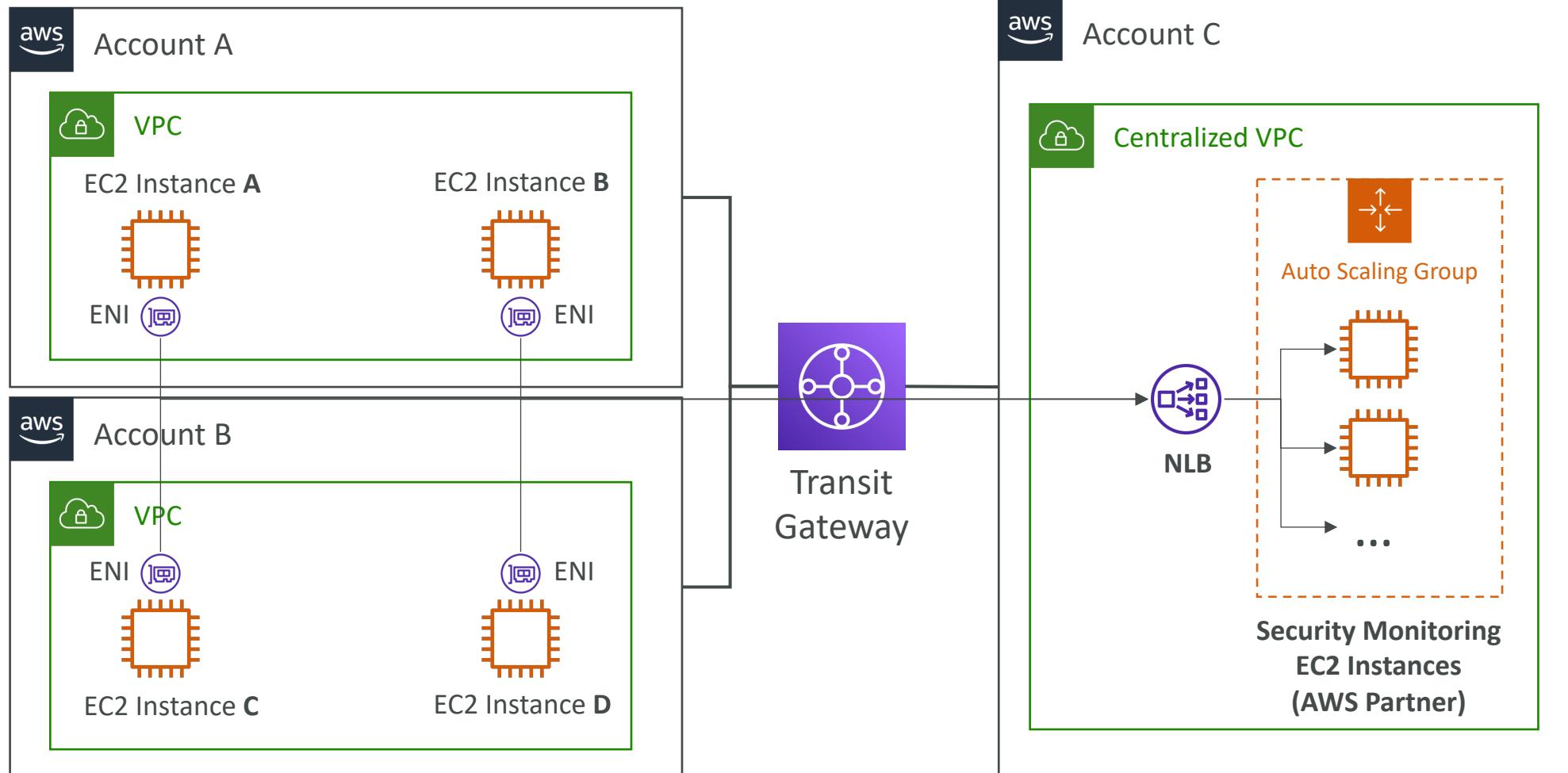
# VPC Traffic Mirroring – Distributed Deployment Model



# Automation of VPC Traffic Mirroring



# VPC Traffic Mirroring – Centralized Deployment Model



**Note:** higher data transfer costs

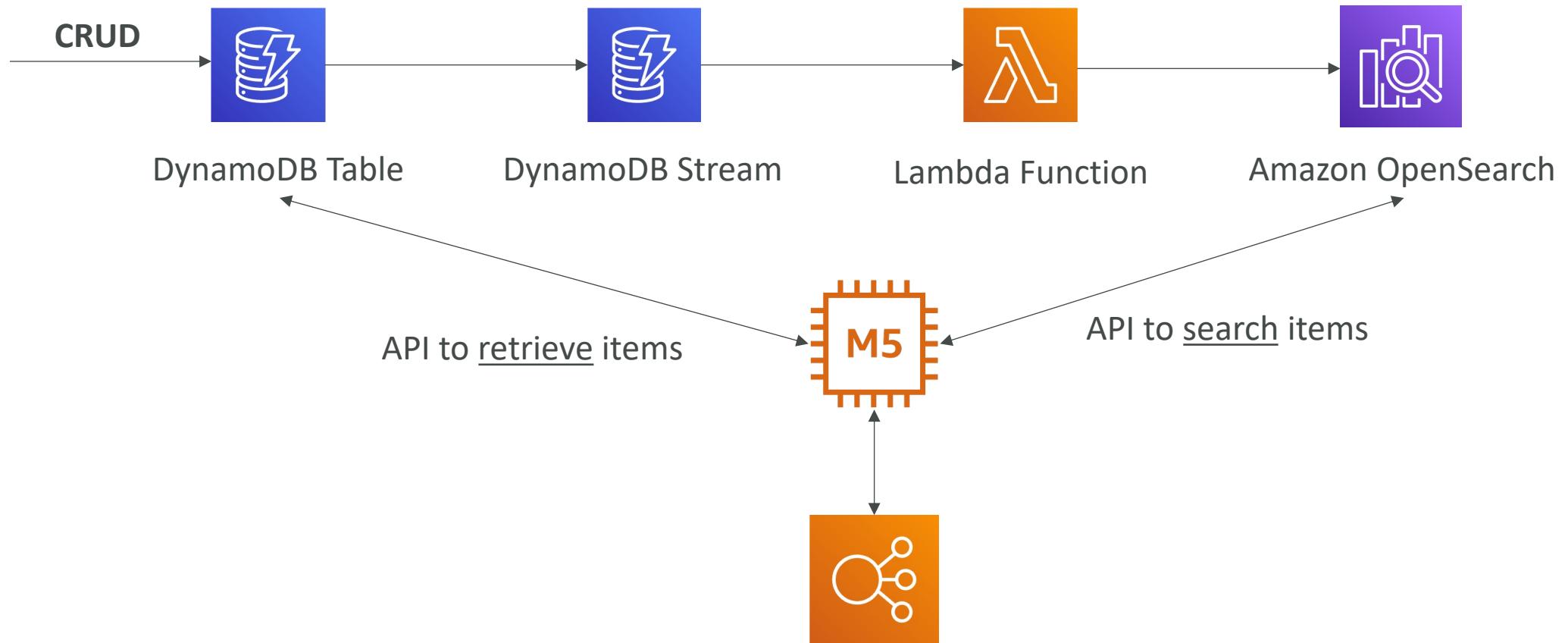
# Amazon OpenSearch Service



- Amazon OpenSearch is successor to Amazon ElasticSearch
- In DynamoDB, queries only exist by primary key or indexes...
- With OpenSearch, you can search any field, even partially matches
- It's common to use OpenSearch as a complement to another database
- OpenSearch requires a cluster of instances (not serverless)
- Does not support SQL (it has its own query language)
- Ingestion from Kinesis Data Firehose, AWS IoT, and CloudWatch Logs
- Security through Cognito & IAM, KMS encryption, TLS
- Comes with OpenSearch Dashboards (visualization)

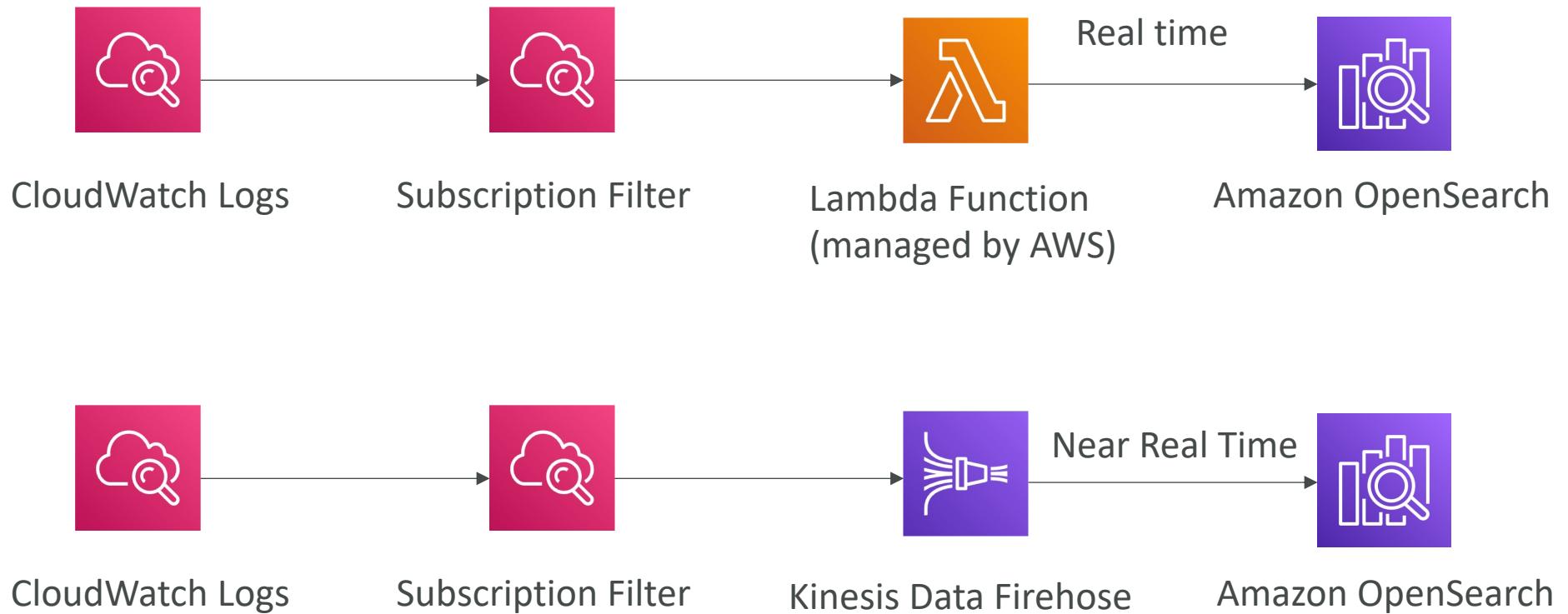
# OpenSearch patterns

## DynamoDB



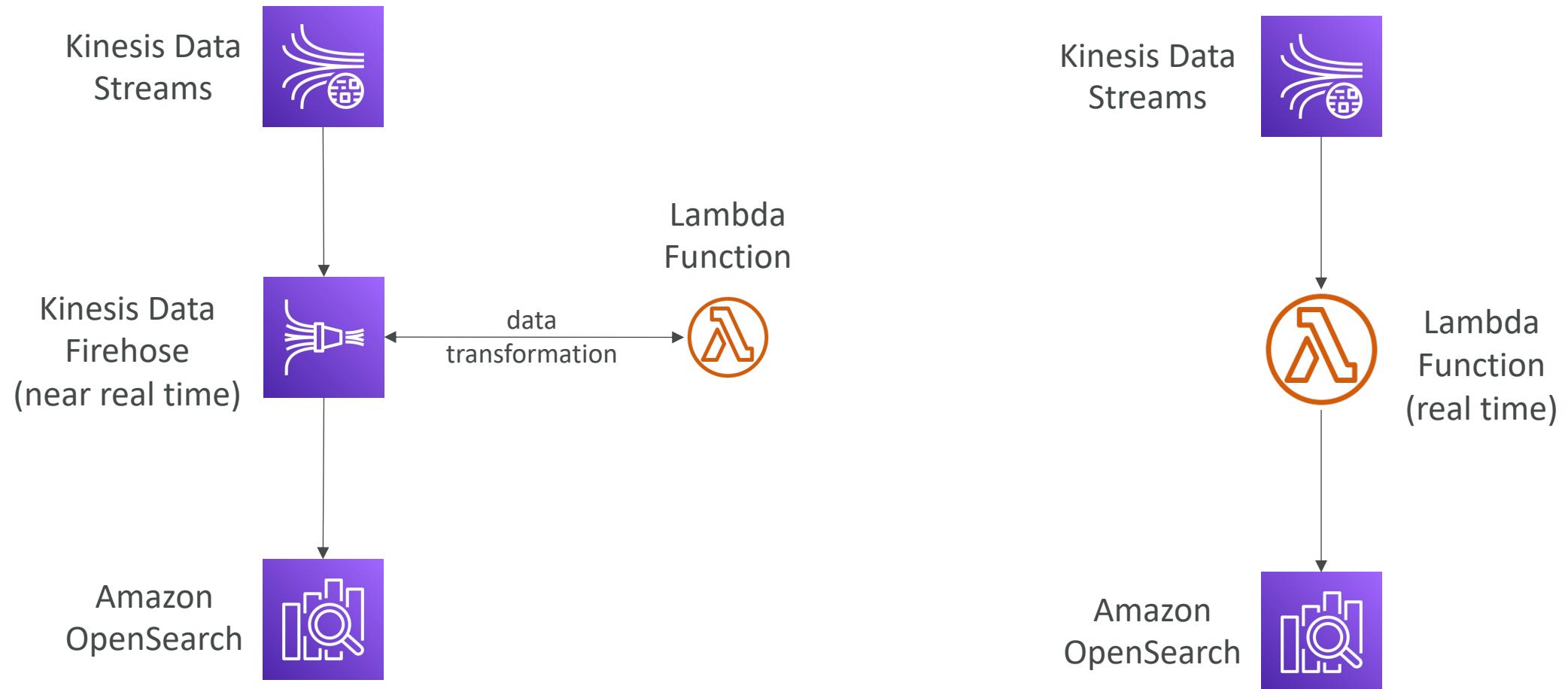
# OpenSearch patterns

## CloudWatch Logs



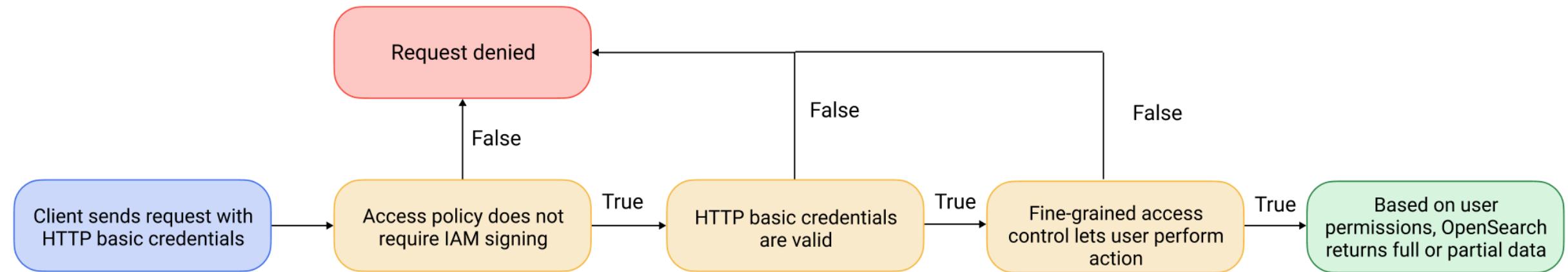
# OpenSearch patterns

## Kinesis Data Streams & Kinesis Data Firehose



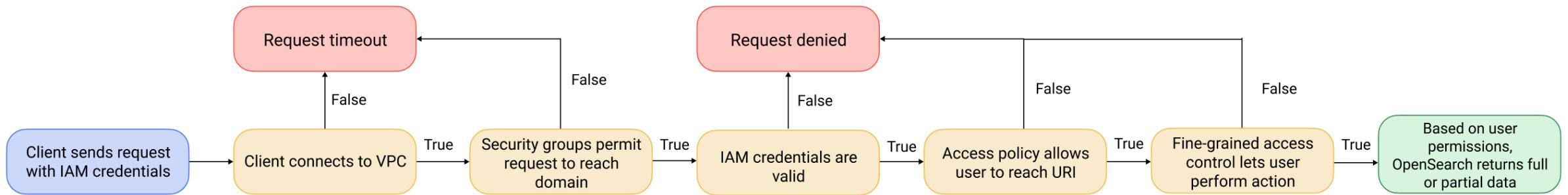
# OpenSearch – Public Access

- Accessible from the Internet with a public endpoint
- Restrict access using **Access Policies**, **Identity-based Policies**, and **IP-based Policies**



# OpenSearch – VPC Access

- Specify VPC, Subnets, Security Groups, and IAM Role
- VPC Endpoints and ENIs will be created (IAM Role)
- You need to use VPN, Transit Gateway, managed network, or proxy server to connect to the domain
- Restrict access using **Access Policies and Identity-based Policies**



# OpenSearch – Deploy in VPC

- Domain Access Policy – specify which actions a principal can perform on the domains subresources (e.g., indexes, APIs)

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Principal": {  
                "AWS": "arn:aws:iam::123456789012:user/test-user"  
            },  
            "Action": "es:*",  
            "Resource": "arn:aws:es:us-west-1:987654321098:domain/  
test-domain/*"  
        }  
    ]  
}
```

**Grant IAM user full access on all  
the OpenSearch domain sub-resources**

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Principal": {  
                "AWS": "arn:aws:iam::123456789012:role/power-user-role"  
            },  
            "Action": [  
                "es:ESHttpGet",  
                "es:ESHttpPut"  
            ],  
            "Resource": "arn:aws:es:us-west-1:987654321098:domain/test-  
domain/commerce-data/*"  
        }  
    ]  
}
```

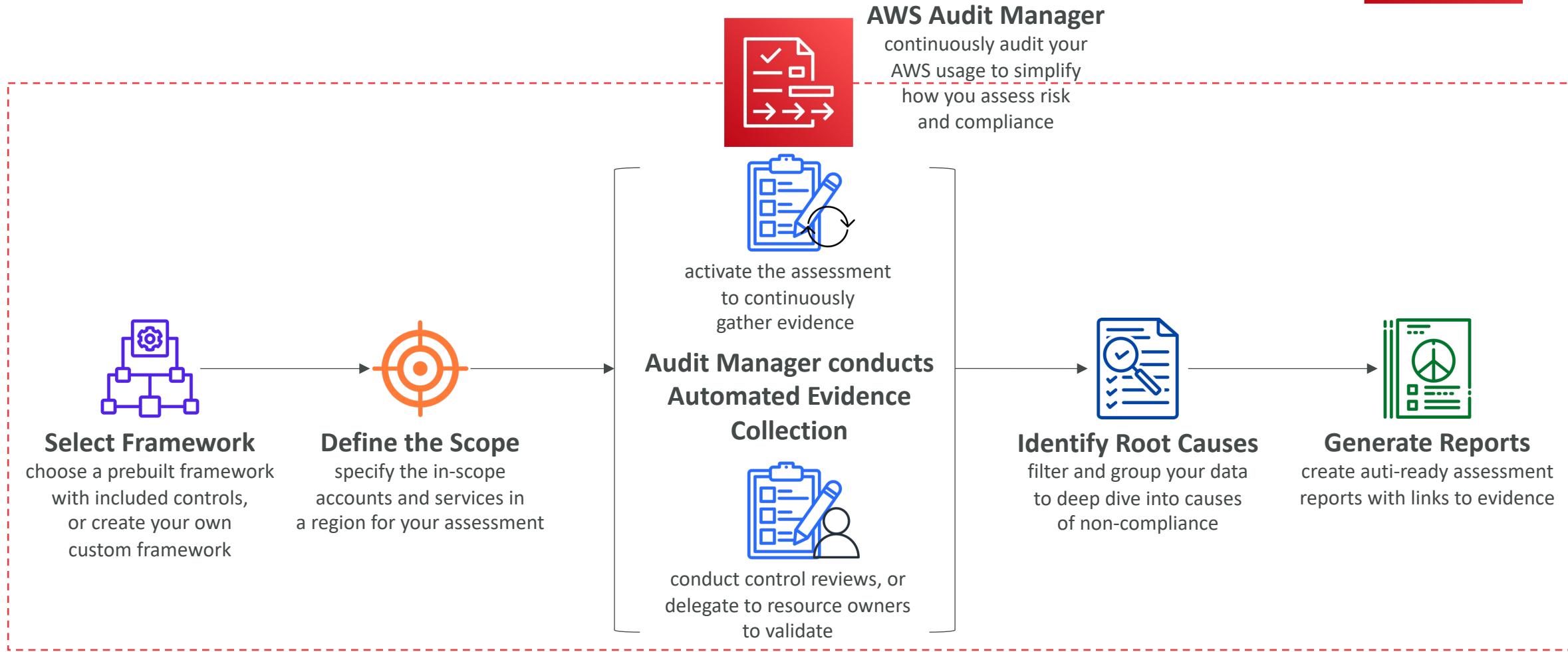
**Grant IAM role access on some methods  
to access the OpenSearch domain sub-resources**

# AWS Audit Manager



- Assess risk and compliance of your AWS workloads
- Continuously audit AWS services usage and prepare audits
- Prebuilt frameworks include:
  - CIS AWS Foundations Benchmark 1.2.0 & 1.3.0
  - General Data Protection Regulation (GDPR),
  - Health Insurance Portability and Accountability Act (HIPAA)
  - Payment Card Industry Data Security Standard (PCI DSS) v3.2.1
  - Service Organization Control 2 (SOC 2)
- Generates reports of compliance alongside evidence folders
- Integrates with Security Hub, Config, Control Tower, CloudTrail, License Manager
- Run over multi-account via integration with AWS Organizations

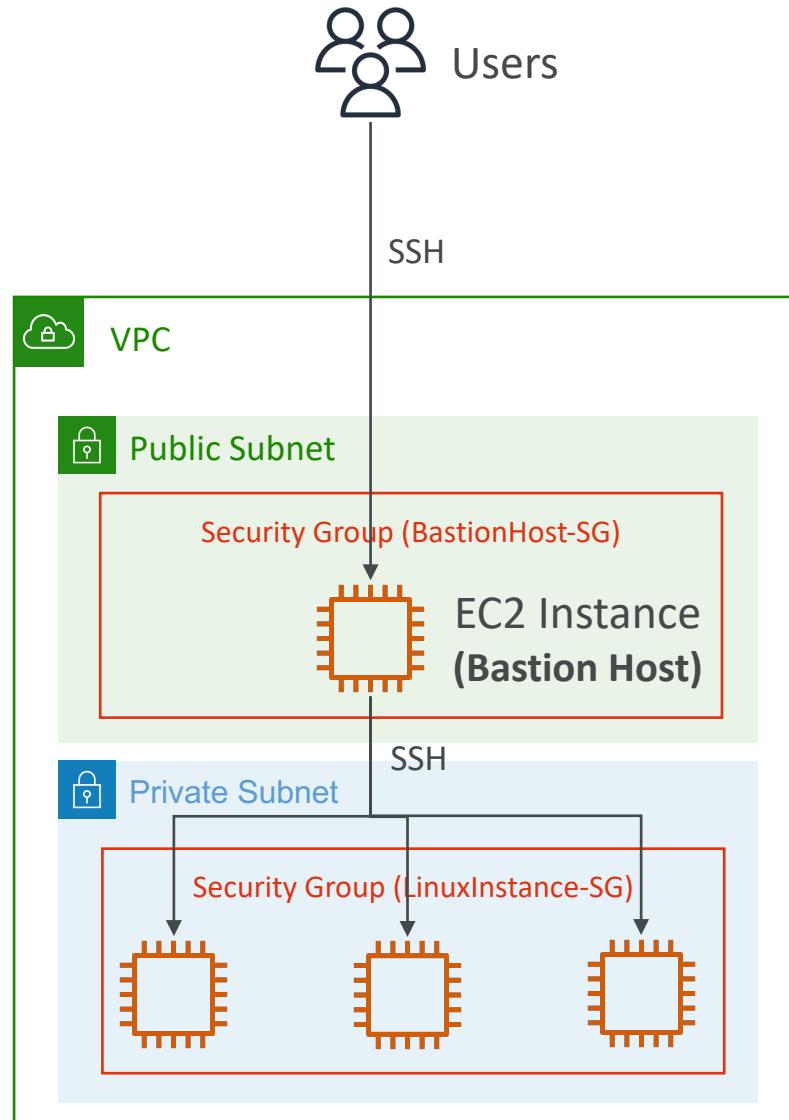
# AWS Audit Manager



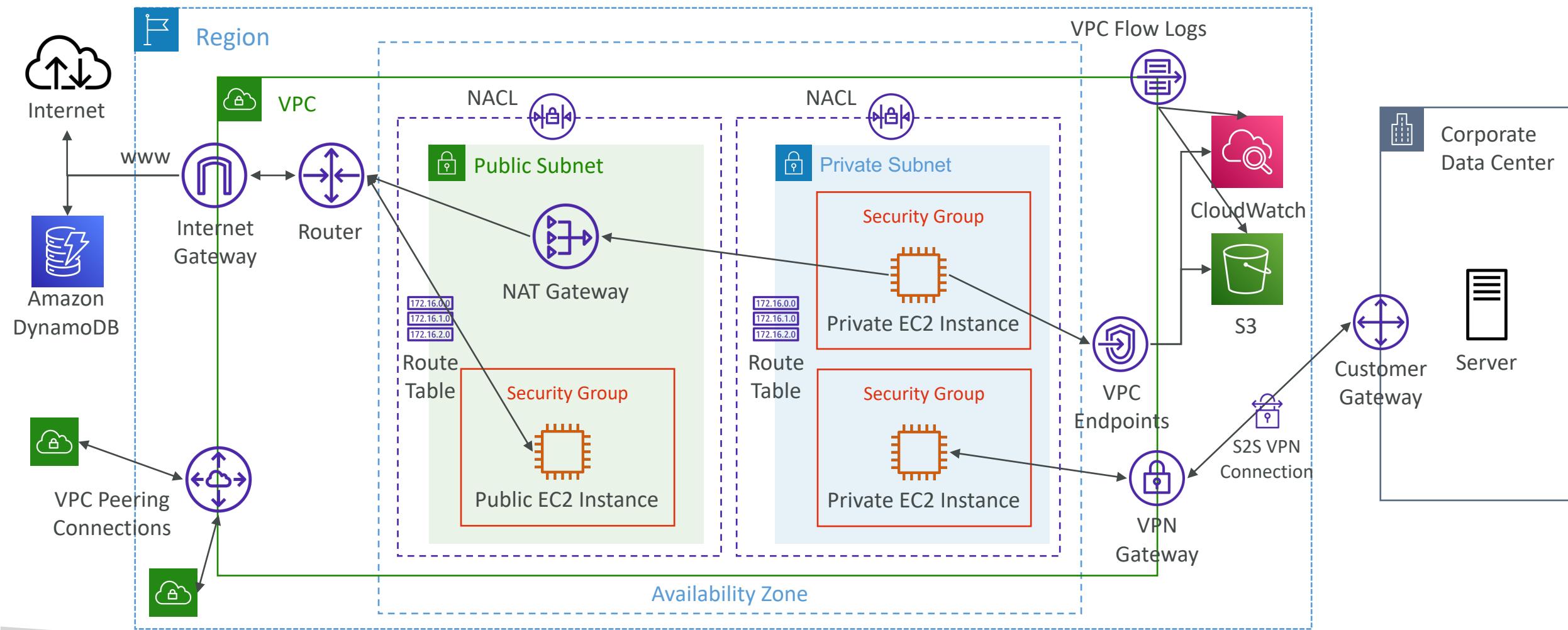
# Domain 3 – Infrastructure Security

# Bastion Hosts

- We can use a Bastion Host to SSH into our private EC2 instances
- The bastion is in the public subnet which is then connected to all other private subnets
- **Bastion Host security group must allow** inbound from the internet on port 22 from restricted CIDR, for example the public CIDR of your corporation
- **Security Group of the EC2 Instances** must allow the Security Group of the Bastion Host, or the private IP of the Bastion host



# AWS Site-to-Site VPN



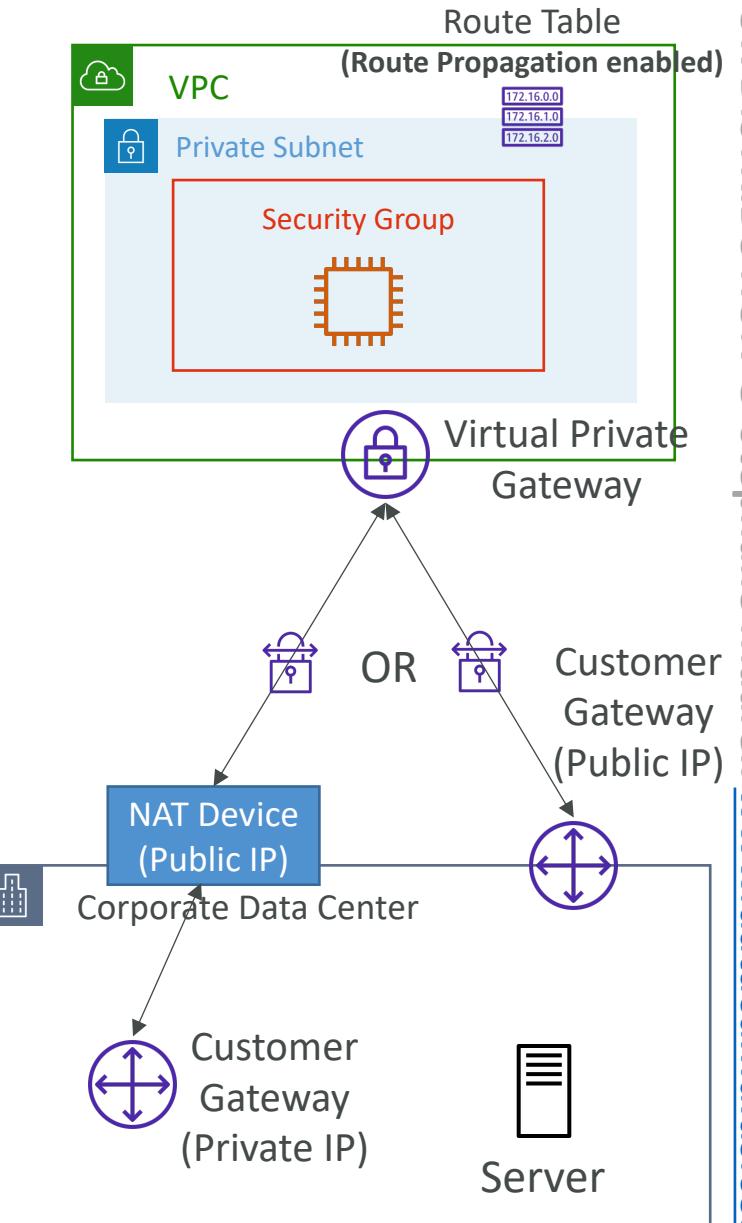
# AWS Site-to-Site VPN



- **Virtual Private Gateway (VGW)**
  - VPN concentrator on the AWS side of the VPN connection
  - VGW is created and attached to the VPC from which you want to create the Site-to-Site VPN connection
  - Possibility to customize the ASN (Autonomous System Number)
- **Customer Gateway (CGW)**
  - Software application or physical device on customer side of the VPN connection
  - <https://docs.aws.amazon.com/vpn/latest/s2svpn/your-cgw.html#DevicesTested>

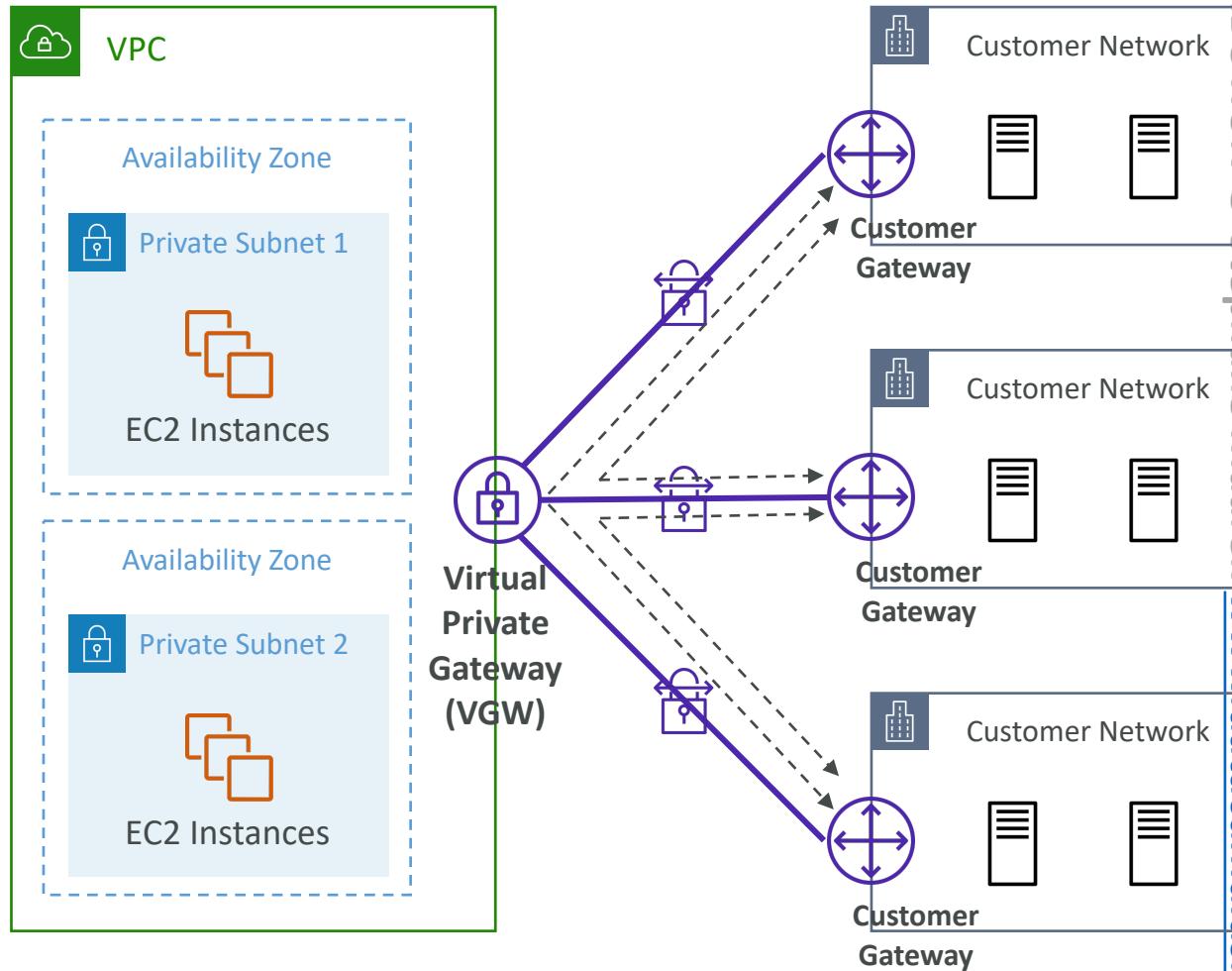
# Site-to-Site VPN Connections

- Customer Gateway Device (On-premises)
  - What IP address to use?
    - Public Internet-routable IP address for your Customer Gateway device
    - If it's behind a NAT device that's enabled for NAT traversal (NAT-T), use the public IP address of the NAT device
- Important step: enable Route Propagation for the Virtual Private Gateway in the route table that is associated with your subnets
- If you need to ping your EC2 instances from on-premises, make sure you add the ICMP protocol on the inbound of your security groups



# AWS VPN CloudHub

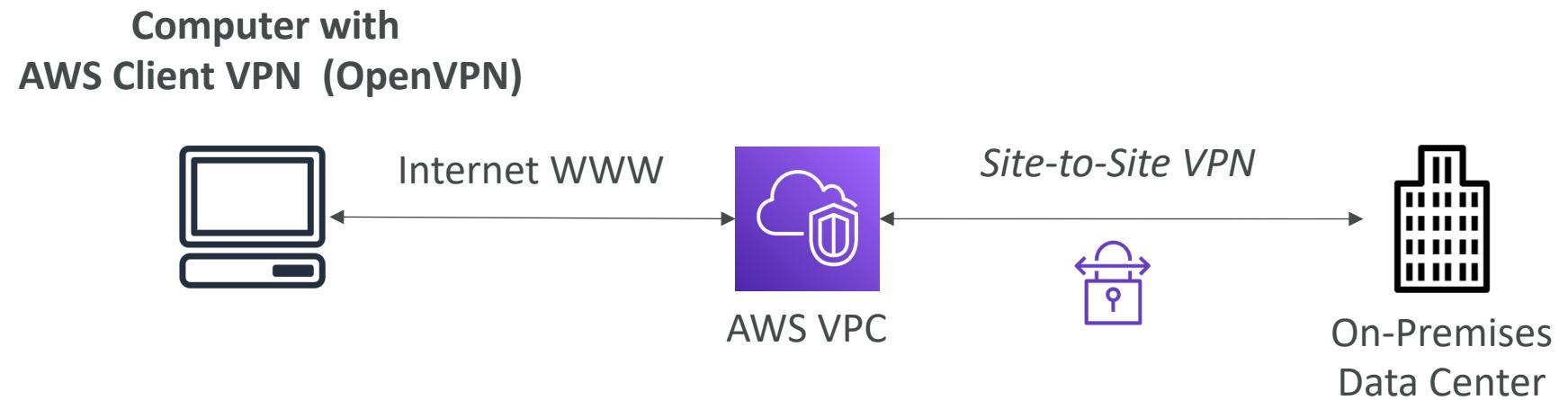
- Provide secure communication between multiple sites, if you have multiple VPN connections
- Low-cost hub-and-spoke model for primary or secondary network connectivity between different locations (VPN only)
- It's a VPN connection so it goes over the public Internet
- To set it up, connect multiple VPN connections on the same VGW, setup dynamic routing and configure route tables



# AWS Client VPN

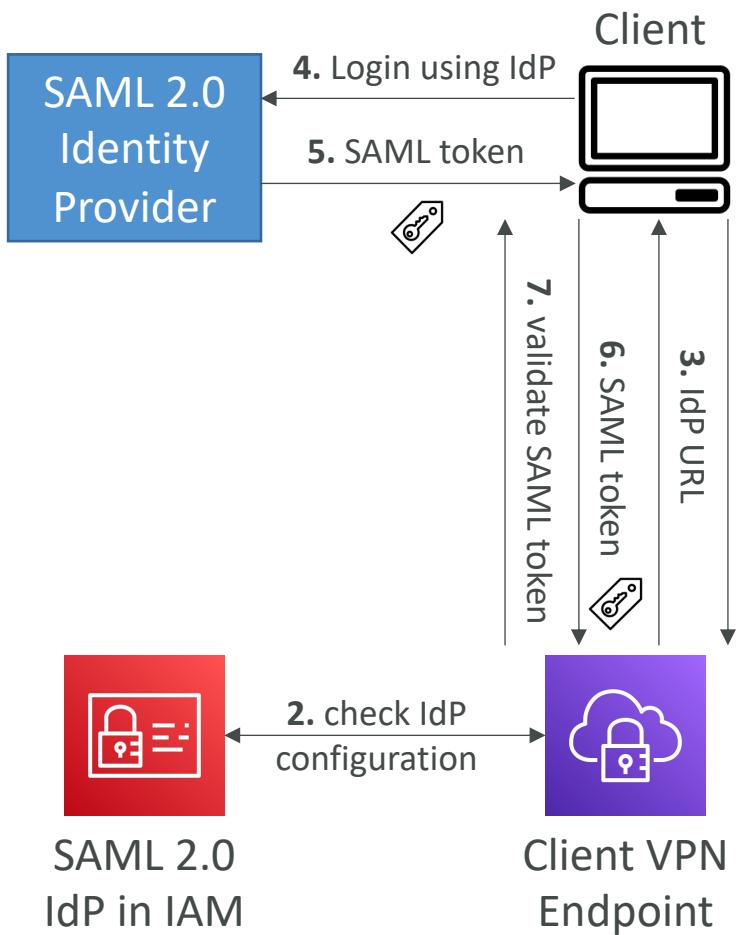


- Connect from your computer using OpenVPN to your private network in AWS and on-premises
- Allow you to connect to your EC2 instances over a private IP (just as if you were in the private VPC network)
- Goes over public Internet



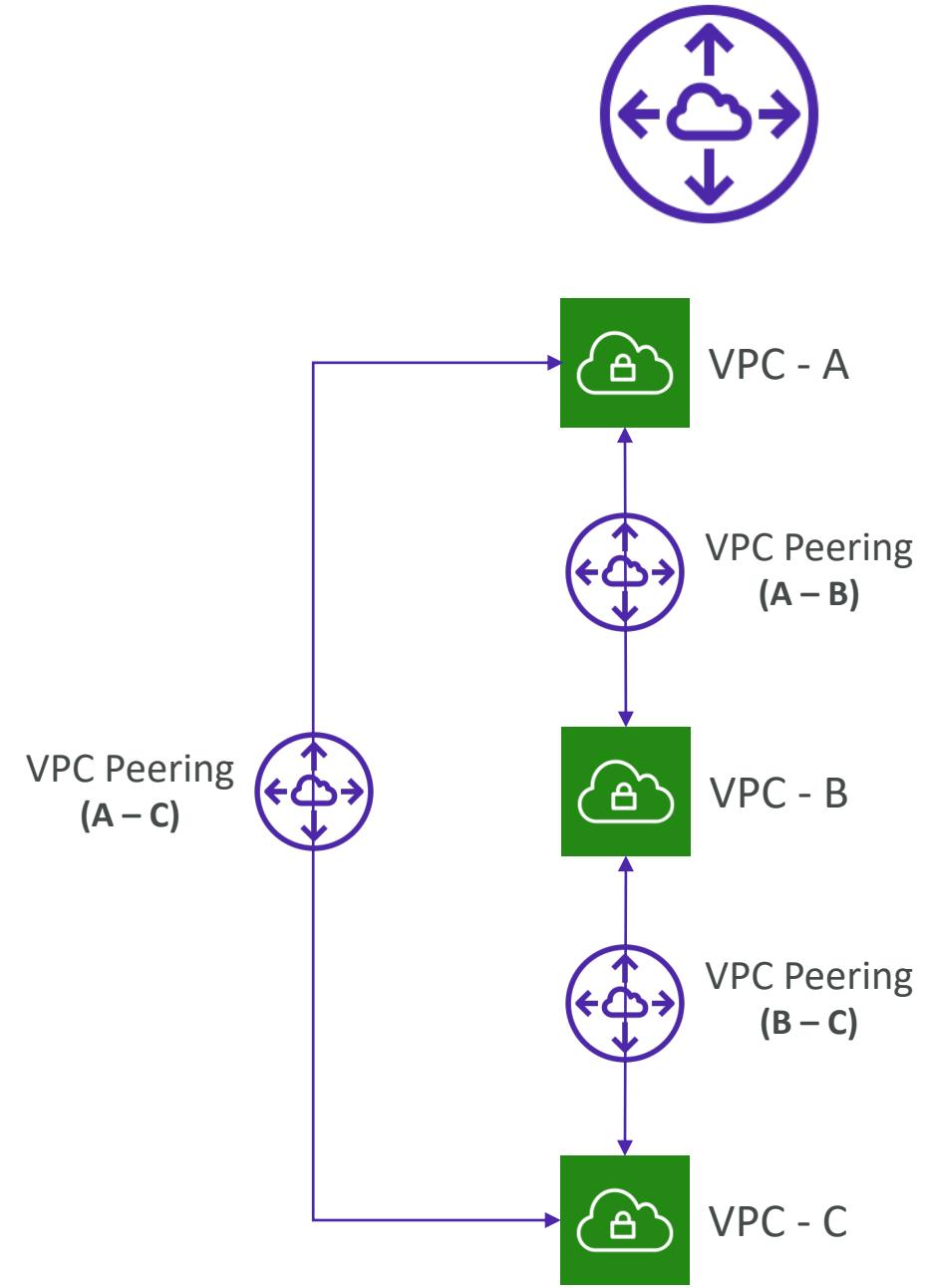
# Client VPN – Authentication Types

- Active Directory Authentication
  - Authenticate against Microsoft Active Directory (User-Based)
  - AWS Managed Microsoft AD or on-premises AD through AD Connector
  - Supports MFA
- Mutual Authentication
  - Use certificates to perform the authentication (Certificate-Based)
  - Must upload the server certificate to AWS Certificate Manager
  - One client certificate for each user (recommended)
- Single Sign-On (supports IAM Identity Center / AWS SSO)
  - Authenticate against SAML 2.0-based identity providers (User-based)
  - Establish trust relationship between AWS and the identity provider
  - Only one identity provider at a time



# VPC Peering

- Privately connect two VPCs using AWS' network
- Make them behave as if they were in the same network
- Must not have overlapping CIDRs
- VPC Peering connection is **NOT** transitive (must be established for each VPC that need to communicate with one another)
- You must update route tables in each VPC's subnets to ensure EC2 instances can communicate with each other



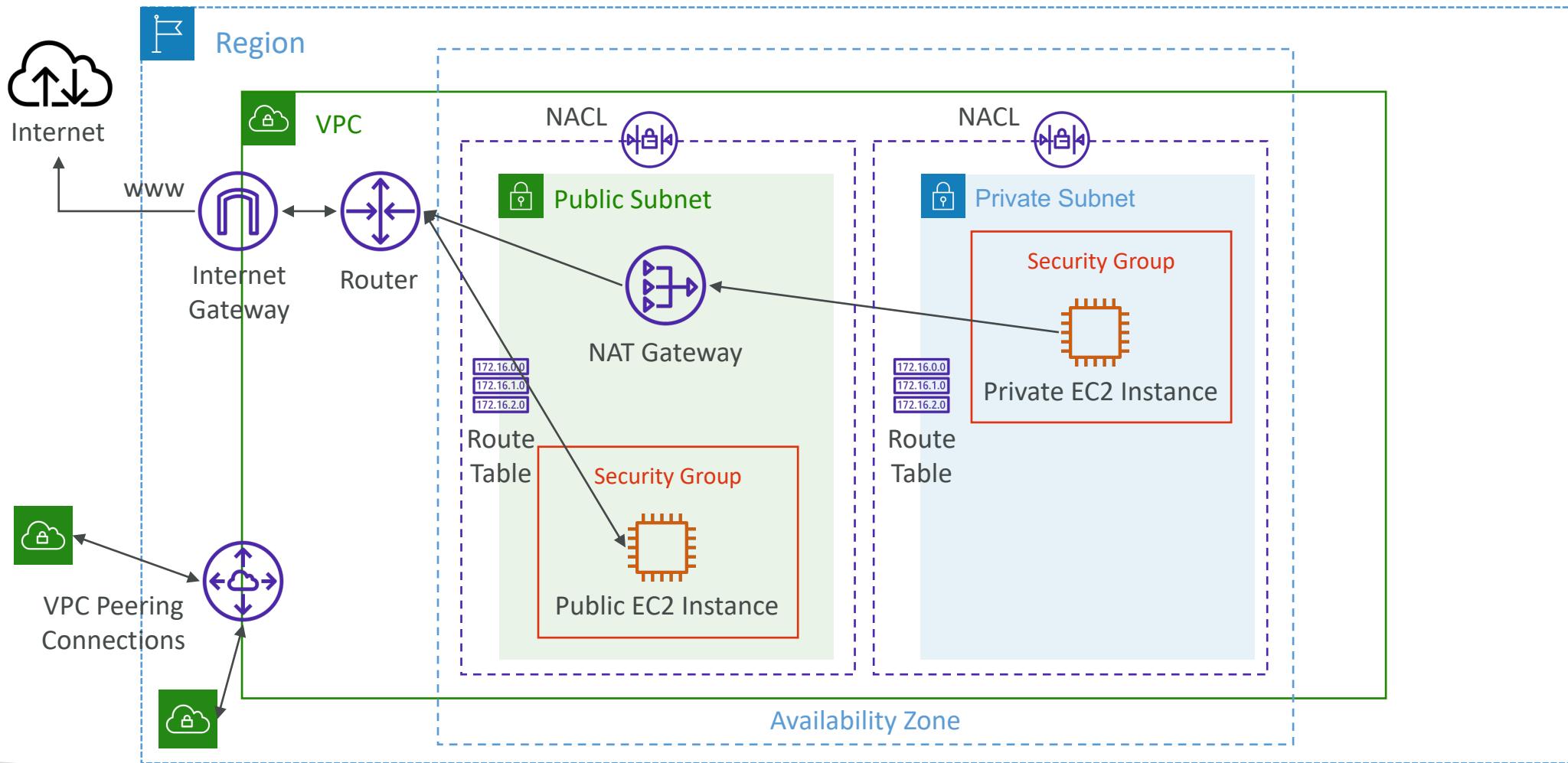
# VPC Peering – Good to know

- You can create VPC Peering connection between VPCs in different AWS accounts/regions
- You can reference a security group in a peered VPC (works cross accounts – same region)

Type	Protocol	Port range	Source
HTTP	TCP	80	sg-04991f9af3473b939 / default
HTTP	TCP	80	[REDACTED] / sg-027ad1f7865d4be76

Account ID

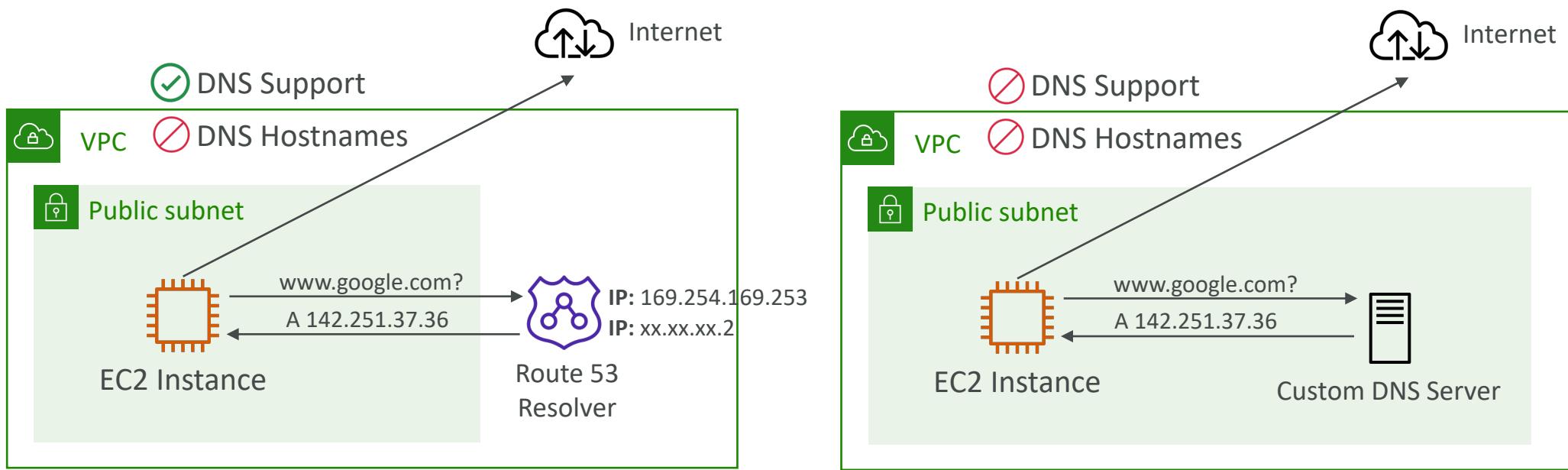
# VPC Peering



# DNS Resolution in VPC

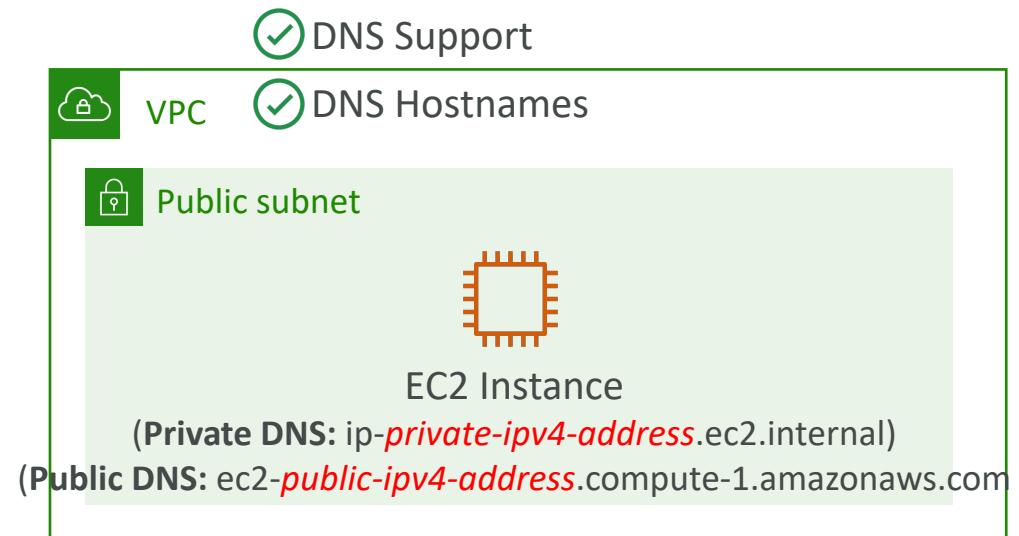
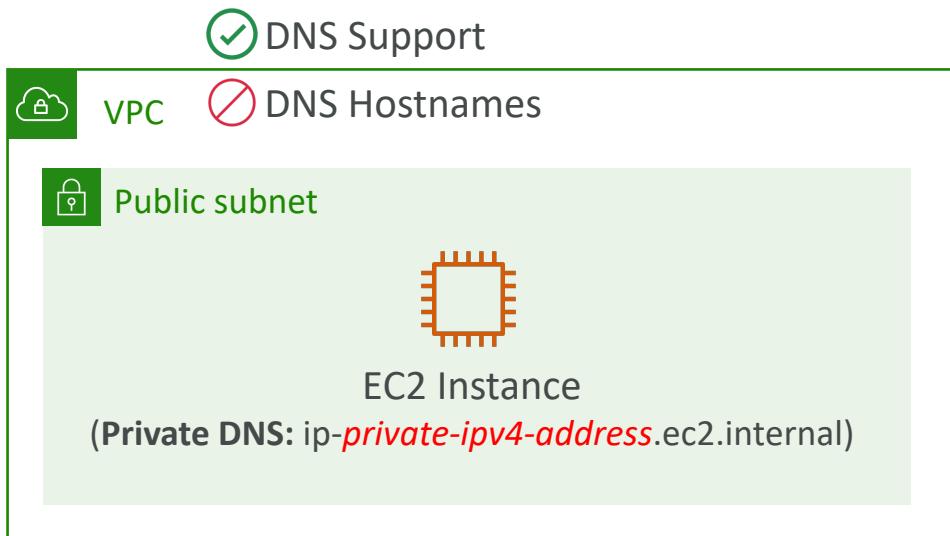
- DNS Resolution (`enableDnsSupport`)

- Decides if DNS resolution from Route 53 Resolver server is supported for the VPC
- True (default): it queries the Amazon Provider DNS Server at 169.254.169.253 or the reserved IP address at the base of the VPC IPv4 network range plus two (.2)



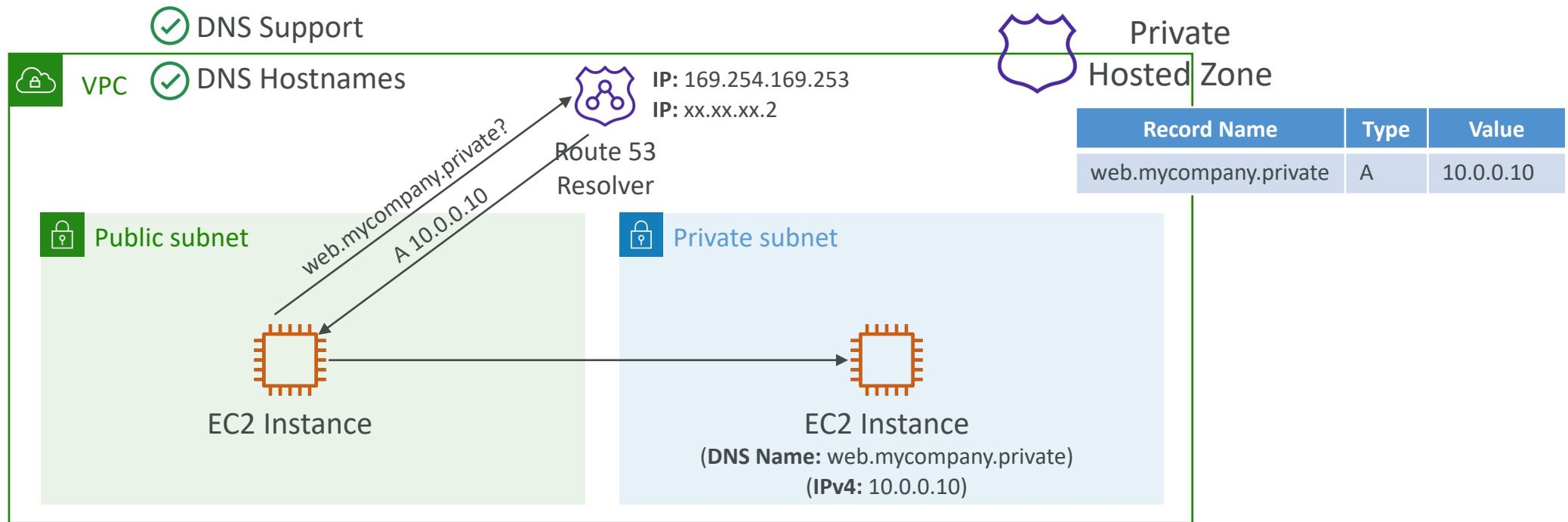
# DNS Resolution in VPC

- DNS Hostnames (enableDnsHostnames)
  - By default,
    - True => default VPC
    - False => newly created VPCs
  - Won't do anything unless enableDnsSupport=true
  - IfTrue, assigns public hostname to EC2 instance if it has a public IPv4



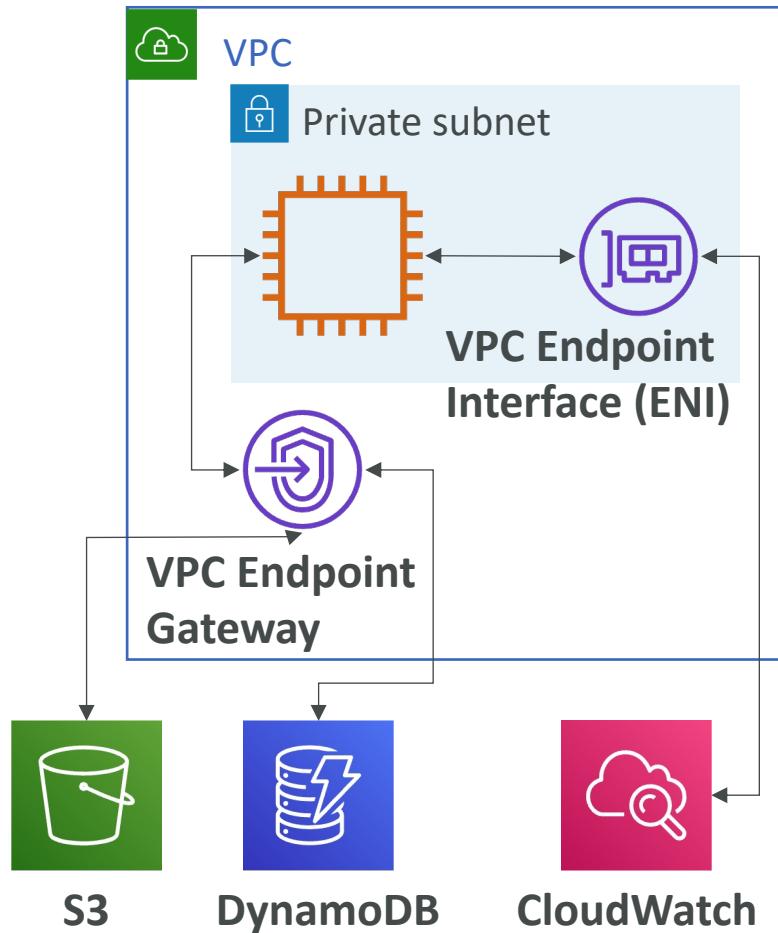
# DNS Resolution in VPC

- If you use custom DNS domain names in a Private Hosted Zone in Route 53, you must set both these attributes (enableDnsSupport & enableDnsHostname) to true



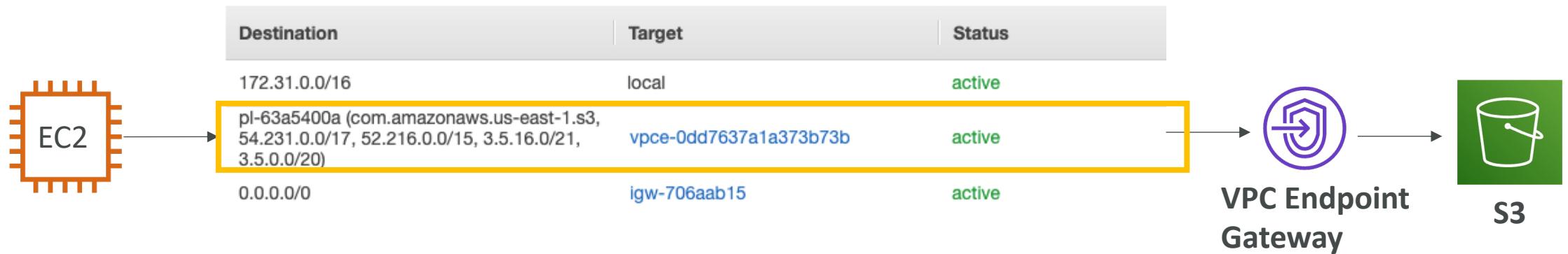
# VPC Endpoints

- Endpoints allow you to connect to AWS Services using a private network instead of the public www network
- They scale horizontally and are redundant
- No more IGW, NAT, etc... to access AWS Services
- VPC Endpoint Gateway (S3 & DynamoDB)
- VPC Endpoint Interface (all except DynamoDB)
- In case of issues:
  - Check DNS Setting Resolution in your VPC
  - Check Route Tables



# VPC Endpoint Gateway

- Only works for S3 and DynamoDB, must create one gateway per VPC
- Must update route tables entries (no security groups!)
- Gateway is defined at the VPC level

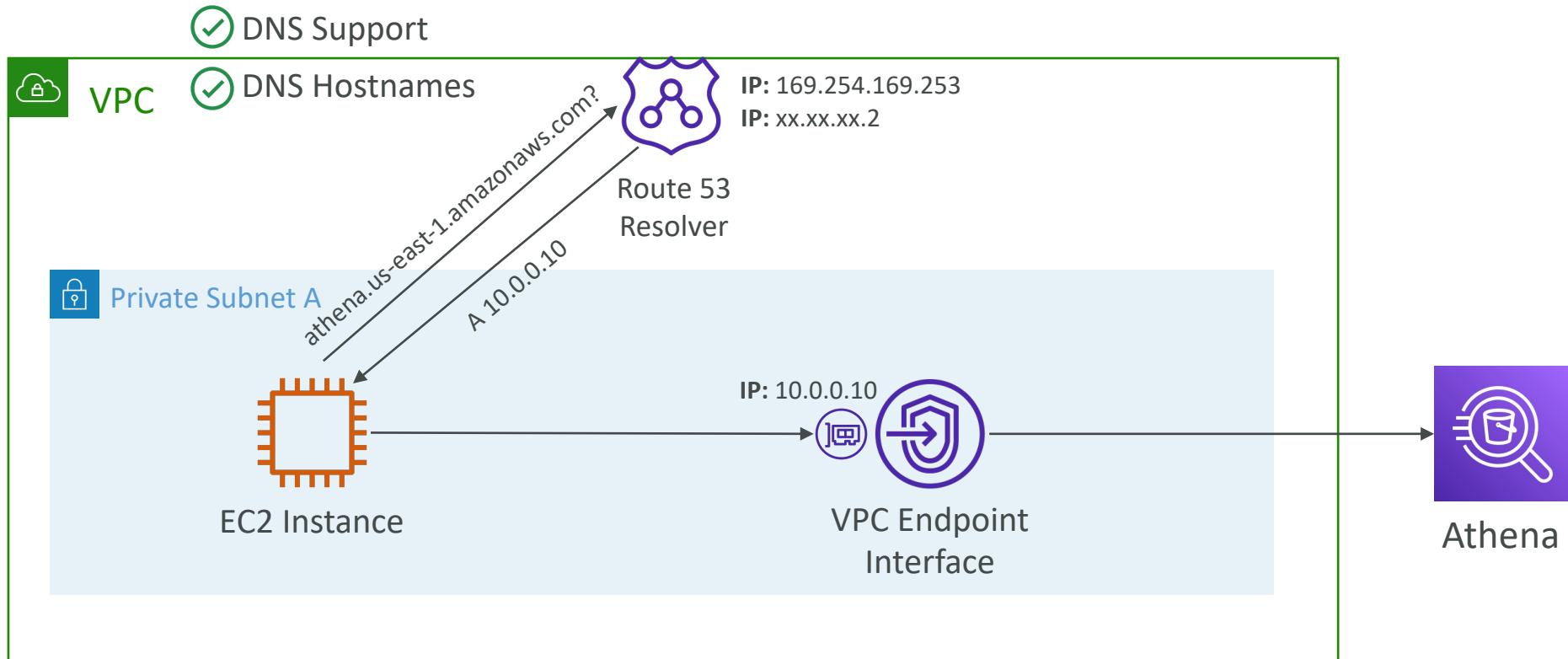


- DNS resolution must be enabled in the VPC
- The same public hostname for S3 can be used
- Gateway endpoint cannot be extended out of a VPC (VPN, DX, TGW, peering)

# VPC Endpoints Interface

- Provision an ENI that will have a private endpoint interface hostname
- Leverage Security Groups for security
- Private DNS (setting when you create the endpoint)
  - The public hostname of a service will resolve to the private Endpoint Interface hostname
  - VPC Setting: “Enable DNS hostnames” and “Enable DNS Support” must be ‘true’
  - Example for Athena:
    - vpce-0b7d2995e9dfe5418-mwrths3x.athena.us-east-1.vpce.amazonaws.com
    - vpce-0b7d2995e9dfe5418-mwrths3x-us-east-1a.athena.us-east-1.vpce.amazonaws.com
    - vpce-0b7d2995e9dfe5418-mwrths3x-us-east-1b.athena.us-east-1.vpce.amazonaws.com
    - athena.us-east-1.amazonaws.com (private DNS name)
- Interface can be accessed from Direct Connect and Site-to-Site VPN

# VPC Endpoints Interface

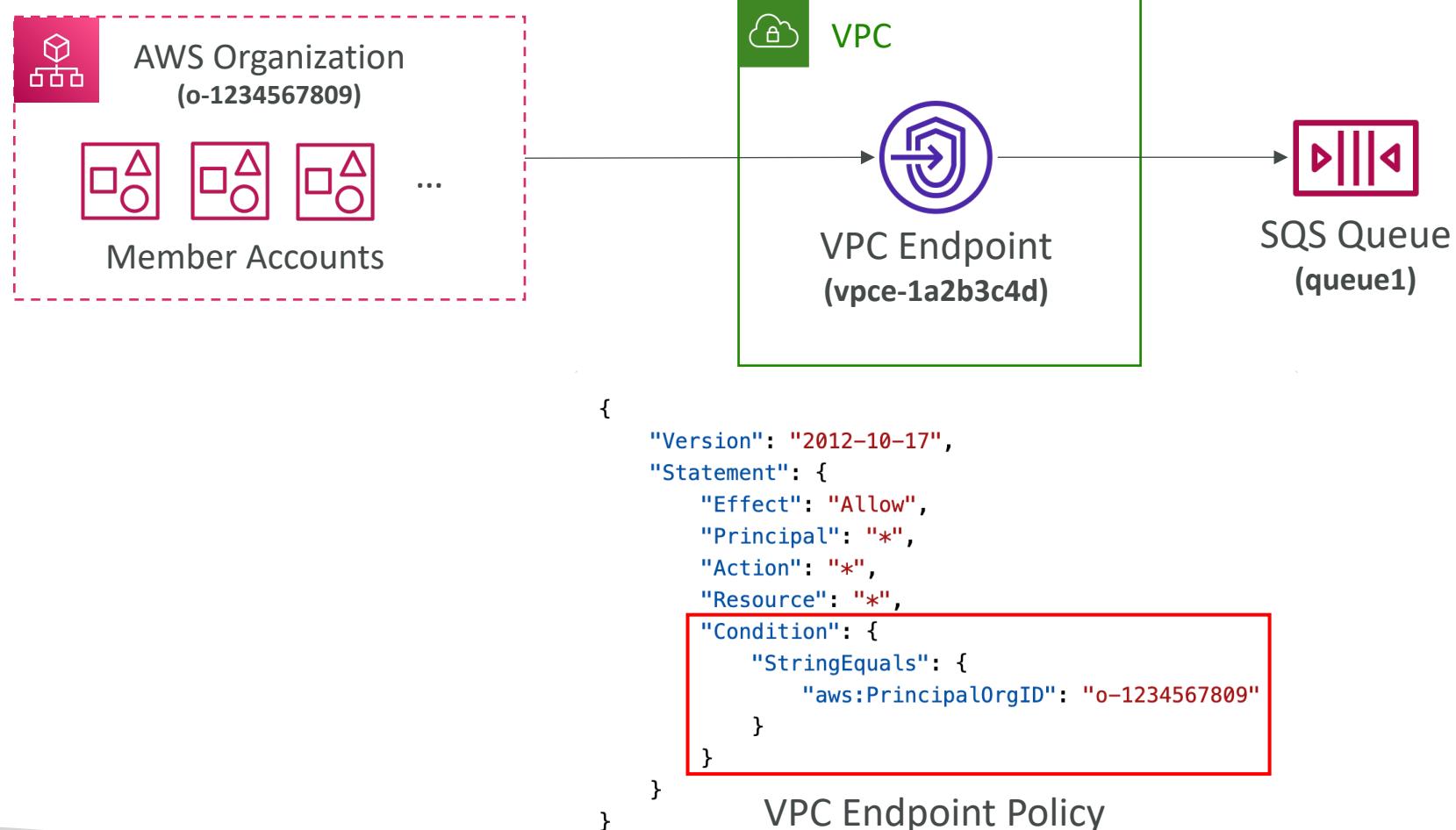


# VPC Endpoint Policy

- Controls which AWS principals (AWS accounts, IAM users, IAM Roles) can use the VPC Endpoint to access AWS services
- Can be attached to both **Interface Endpoint** and **Gateway Endpoint**
- Can restrict specific API calls on specific resources
- Doesn't override or replace Identity-based Policies or service-specific policies (e.g., S3 bucket policy)
- Note: can use `aws:PrincipalOrgId` to restrict access only within the Organization

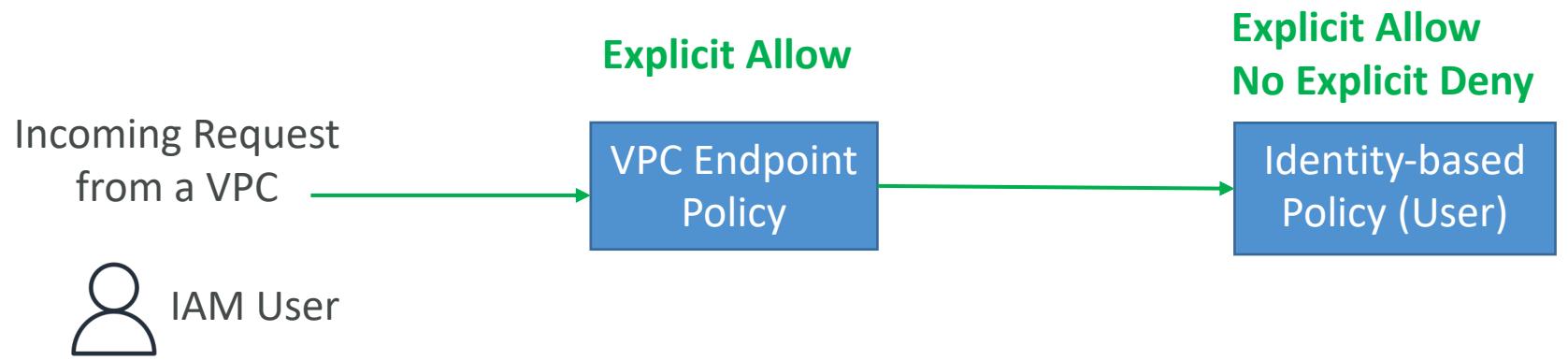
```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {"Effect": "Allow",  
     "Principal": "*",  
     "Action": [  
       "s3>ListBucket",  
       "s3GetObject",  
       "s3PutObject"  
     ],  
     "Resource": [  
       "arn:aws:s3:::bucket_name",  
       "arn:aws:s3:::bucket_name/*"  
     ],  
     "Condition": {  
       "ArnEquals": {  
         "aws:PrincipalArn": "arn:  
aws:iam::123456789012:role/test-role"  
       }  
     }  
  }  
}
```

# VPC Endpoint Policies – Diagram



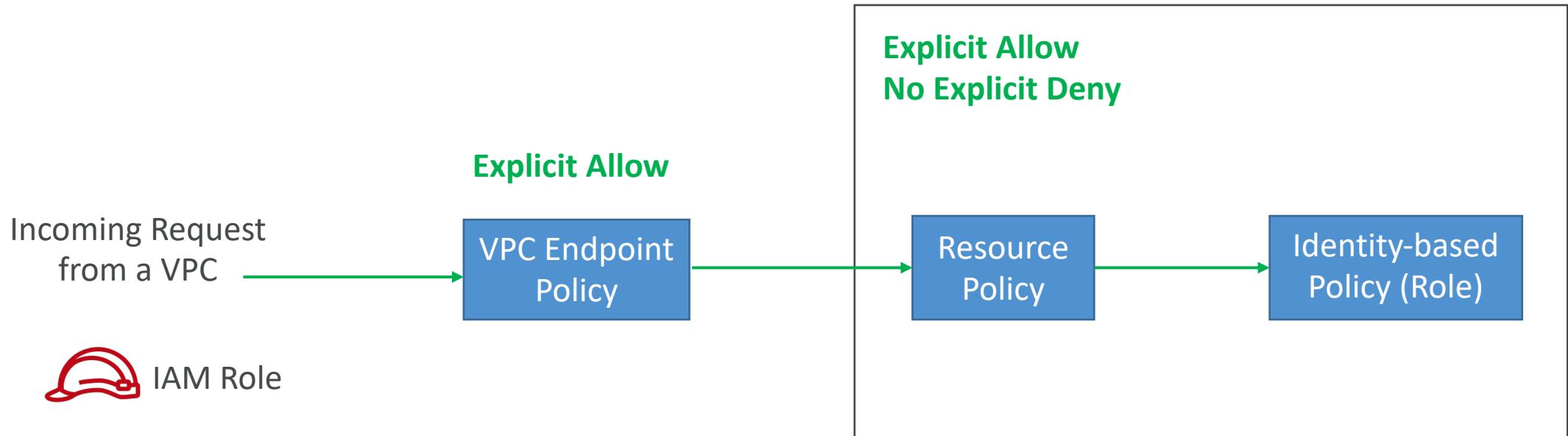
# VPC Endpoint Policy – Authorization logic

- VPC Endpoint Policy + Identity-based Policy

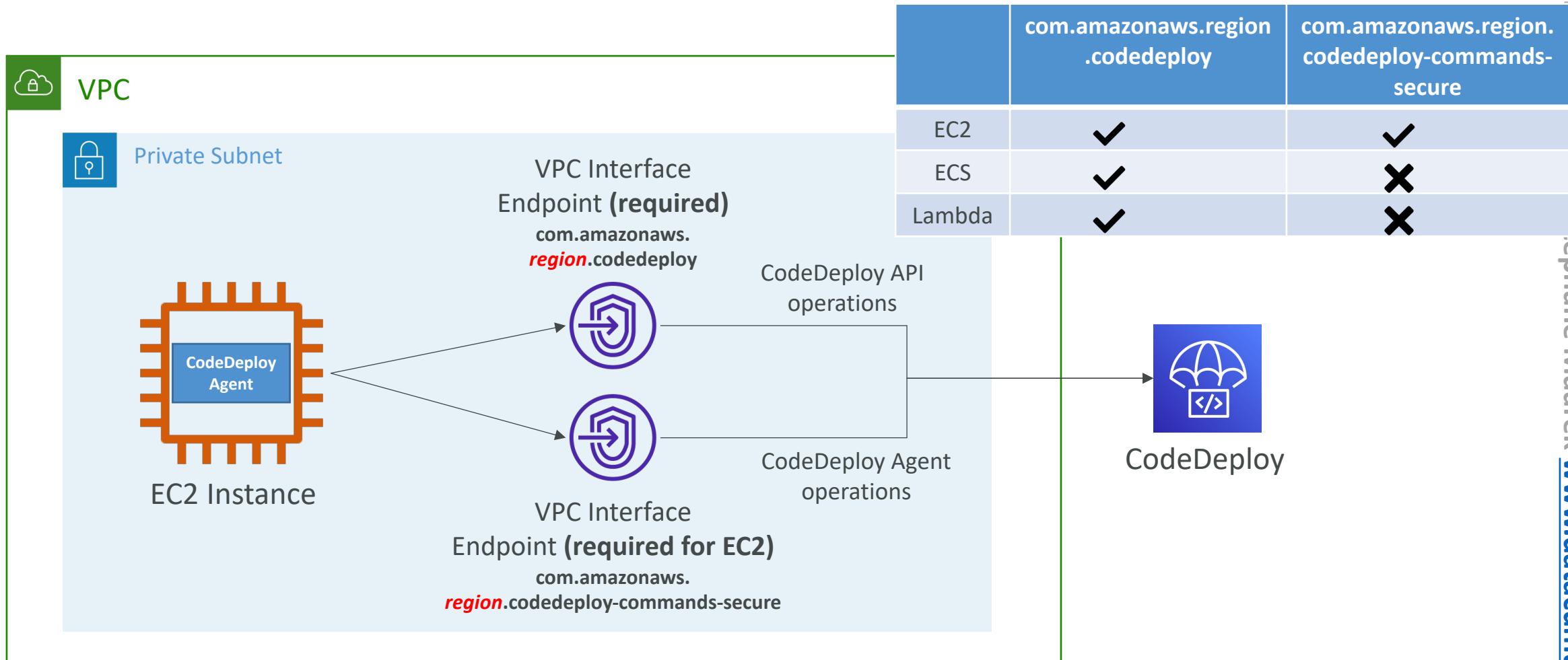


# VPC Endpoint Policy – Authorization logic

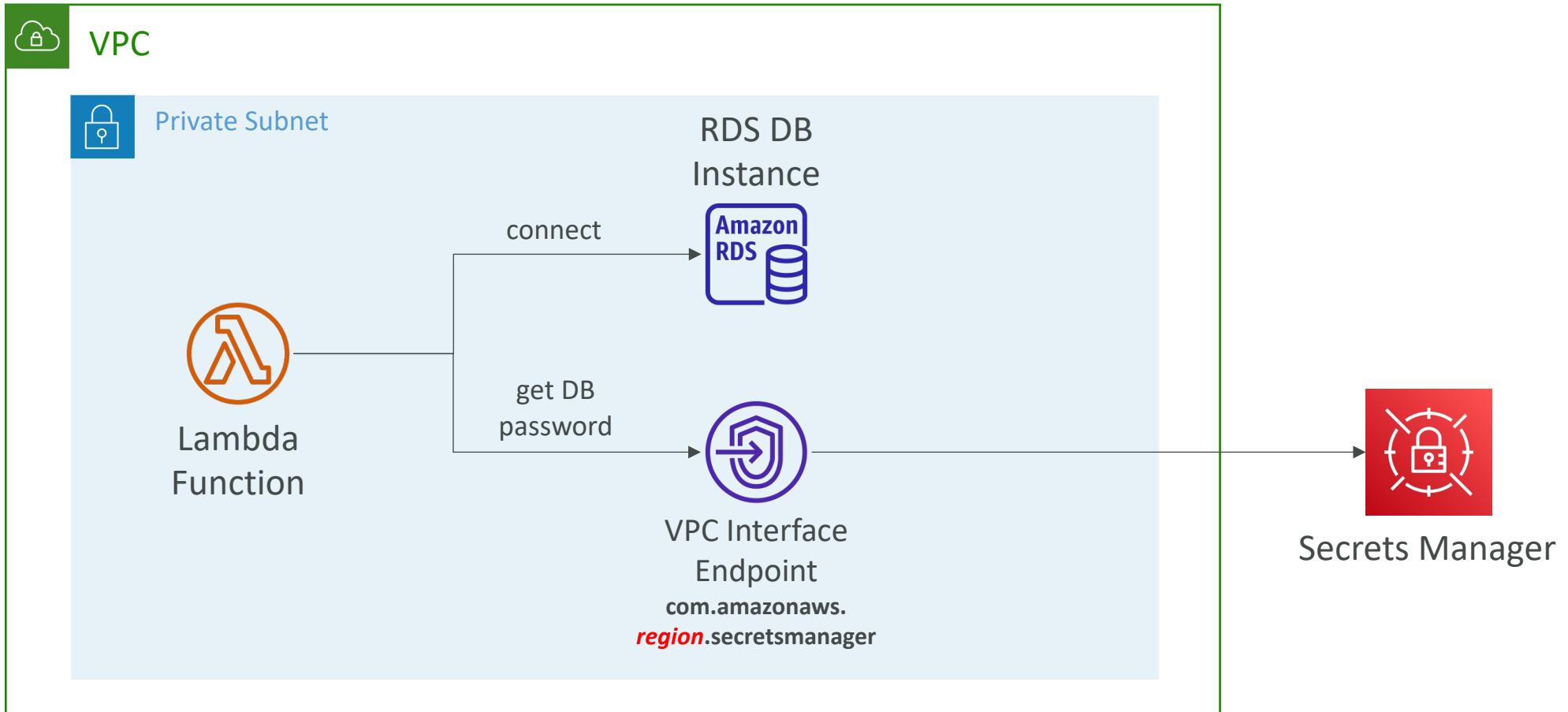
- VPC Endpoint Policy + Resource-based Policy + IAM Role



# VPC Endpoint – CodeDeploy

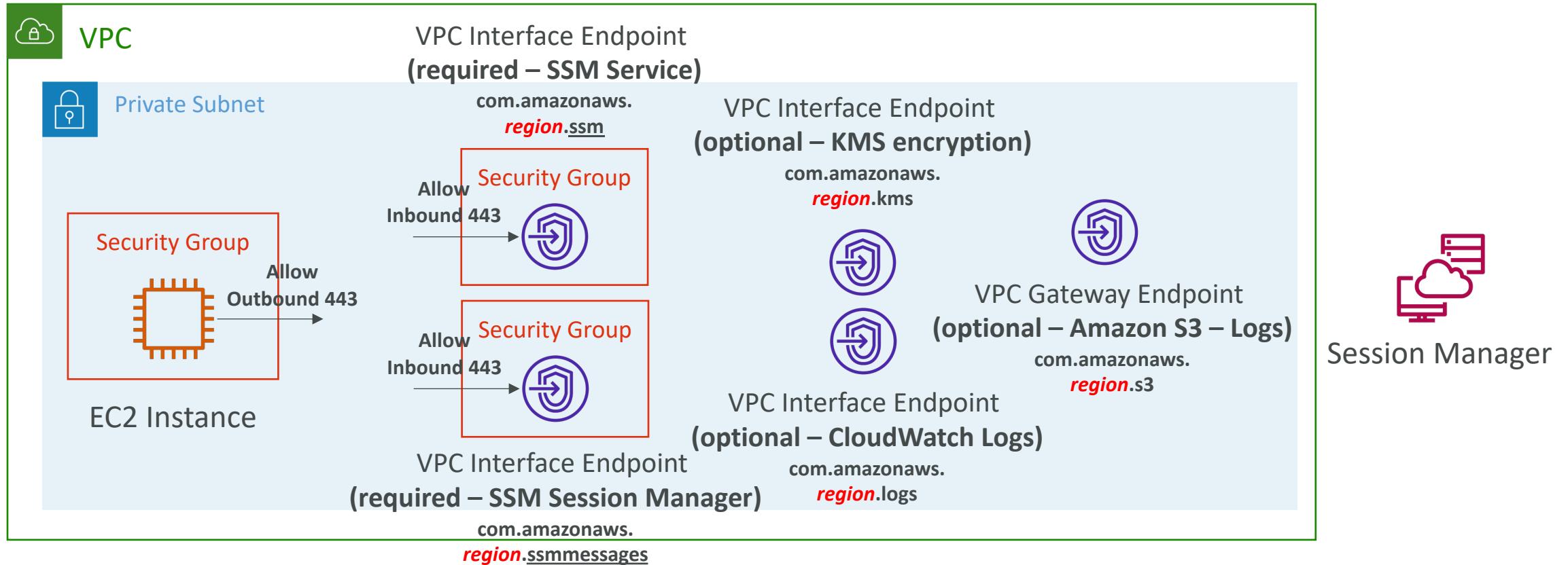


# VPC Endpoint – Secrets Manager

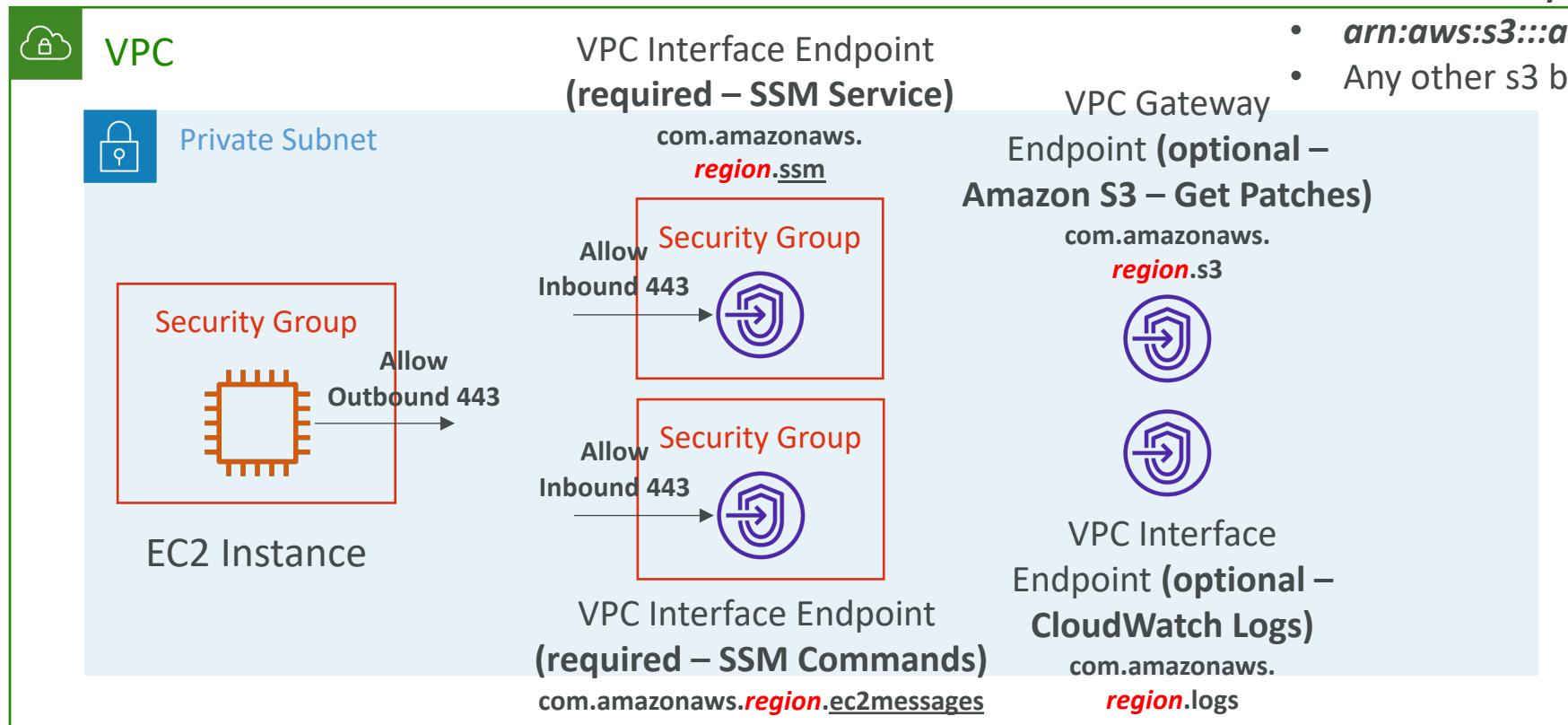


# VPC Endpoint – SSM Session Manager

- Connect to EC2 instances in private subnets, without Internet access



# VPC Endpoint – Patch Manager



**Note:** VPC endpoint policy must allow access to those S3 buckets:

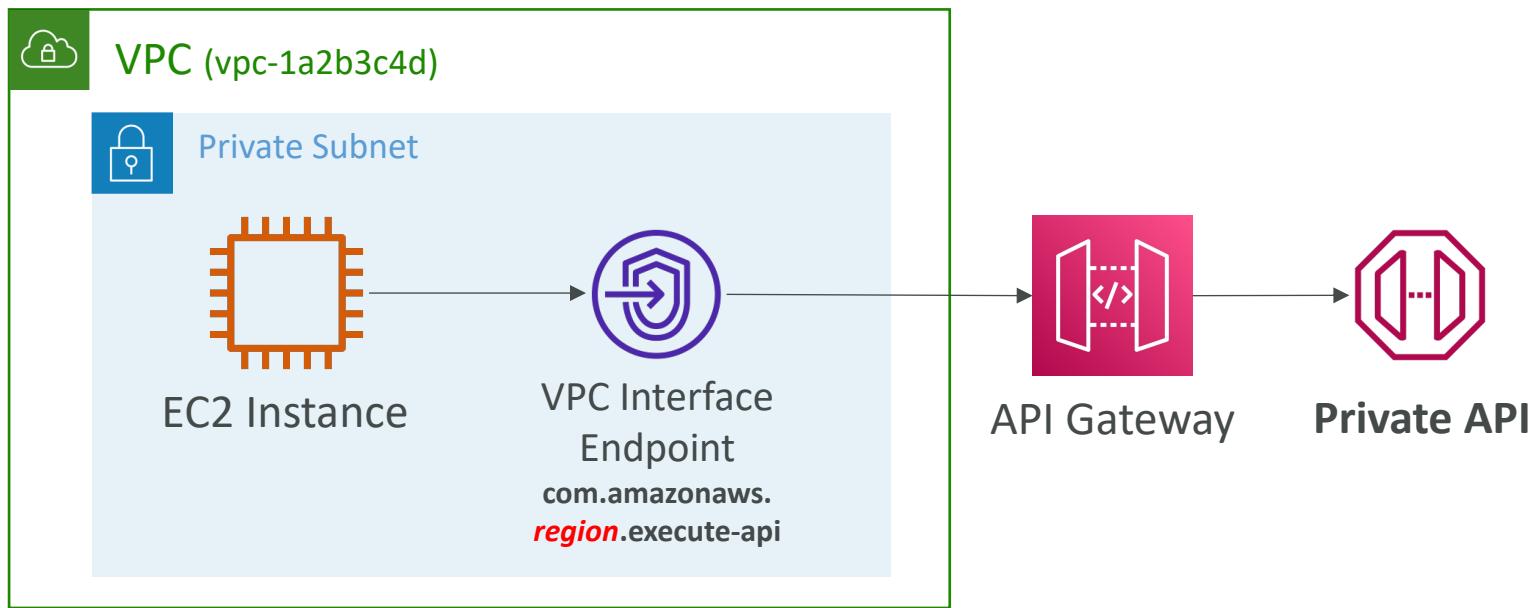
- *arn:aws:s3:::patch-baseline-snapshot-*region*/\**
- *arn:aws:s3:::aws-ssm-*region*/\**
- Any other s3 buckets required by SSM



**Note:** for S3 Gateway Endpoint, update route tables

# VPC Endpoint – API Gateway

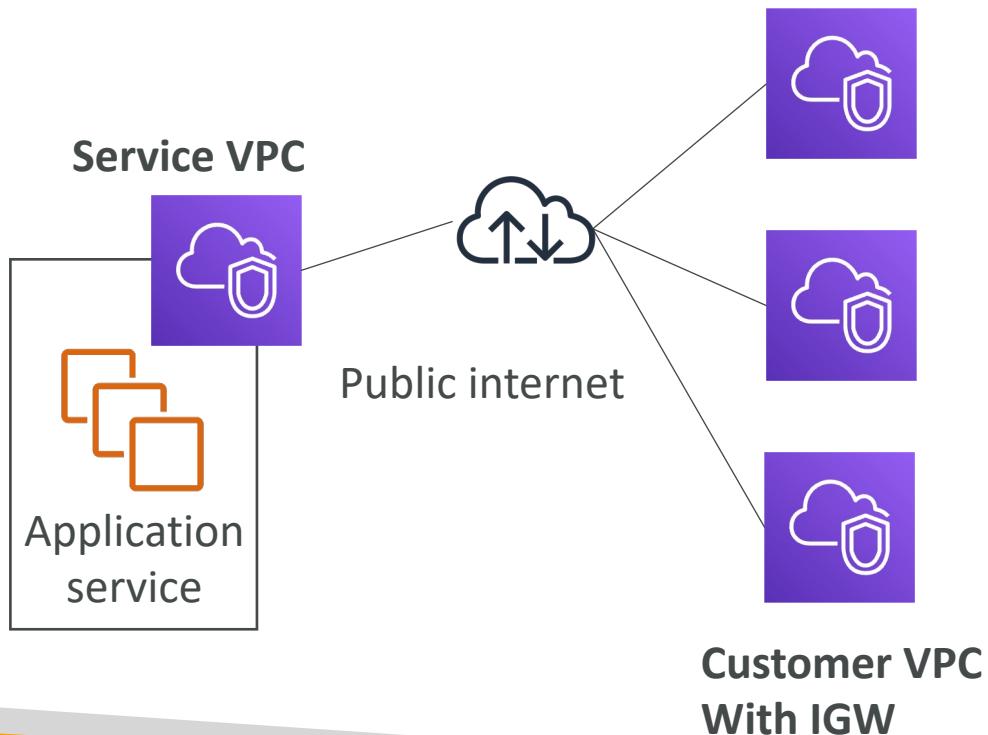
- Private REST APIs can only be accessed using VPC Interface Endpoint
  - VPC Endpoint Policies can be used together with API Gateway resource policies
  - Restrict access to your private APIs from VPC and VPC Endpoints using resource policies (`aws:SourceVpc` and `aws:SourceVpce`)

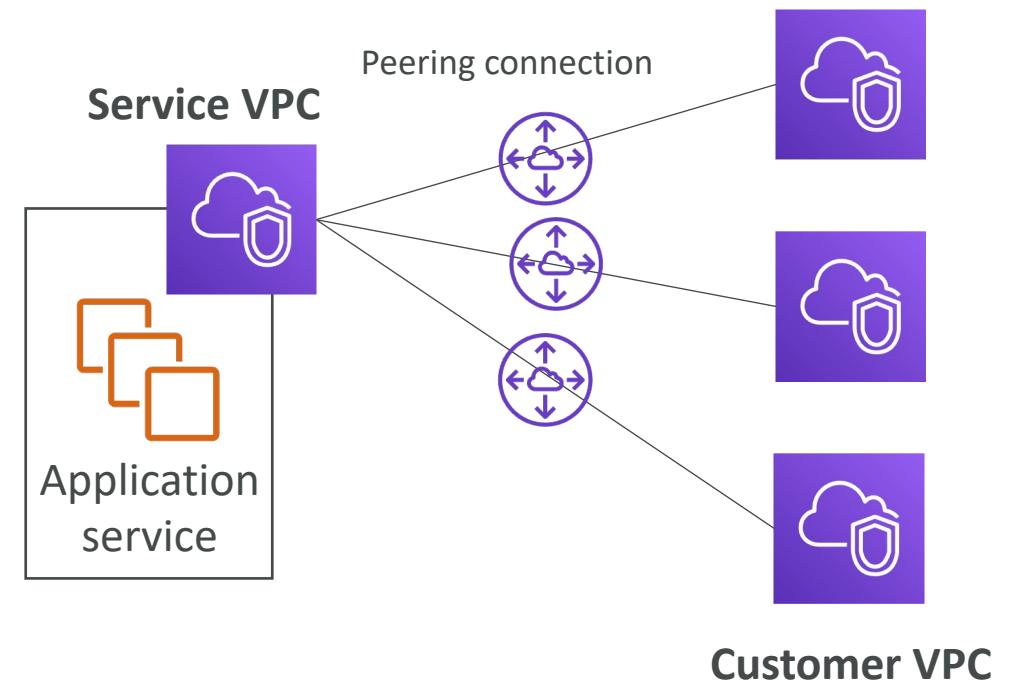


```
"Version": "2012-10-17",
"Statement": [
    {
        "Effect": "Allow",
        "Principal": "*",
        "Action": "execute-api:Invoke",
        "Resource": "execute-api:/"
    },
    {
        "Effect": "Deny",
        "Principal": "*",
        "Action": "execute-api:Invoke",
        "Resource": "execute-api:/",
        "Condition": {
            "StringNotEquals": {
                "aws:SourceVpc": "vpc-1a2b3c4d"
            }
        }
    }
]
```

# Exposing services in your VPC to other VPC

- Option 1: make it public
  - Goes through the public www
  - Tough to manage access
- Option 2: VPC peering
  - Must create many peering relations
  - Opens the **whole** network

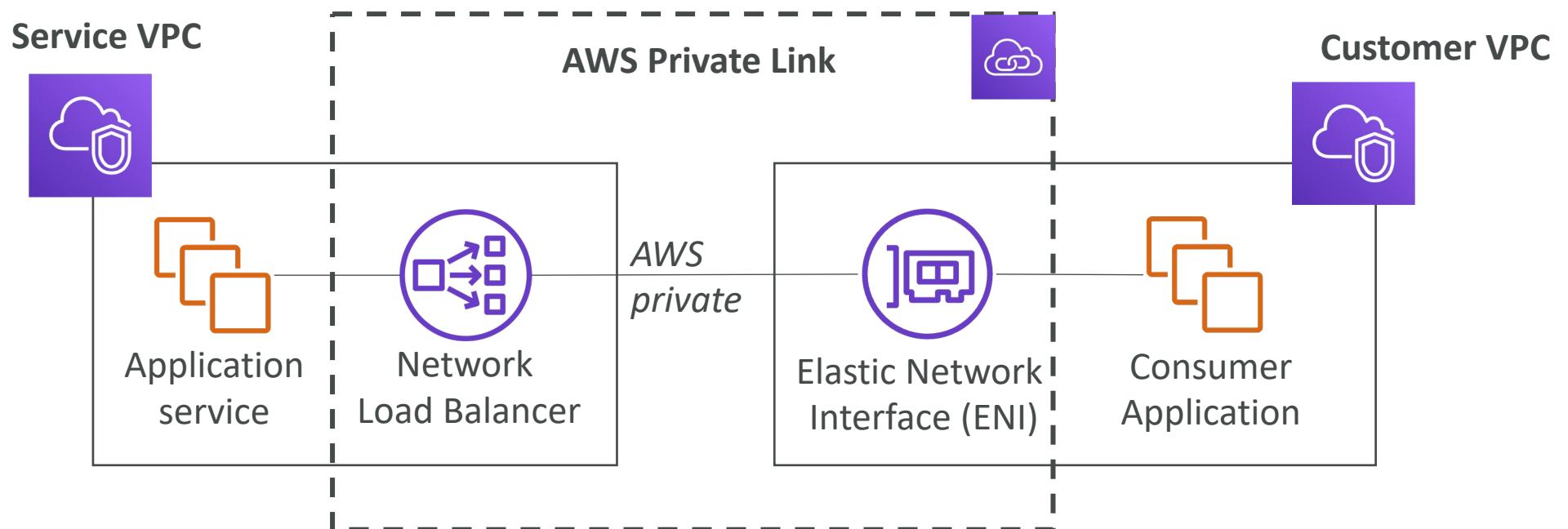


# AWS PrivateLink (VPC Endpoint Services)

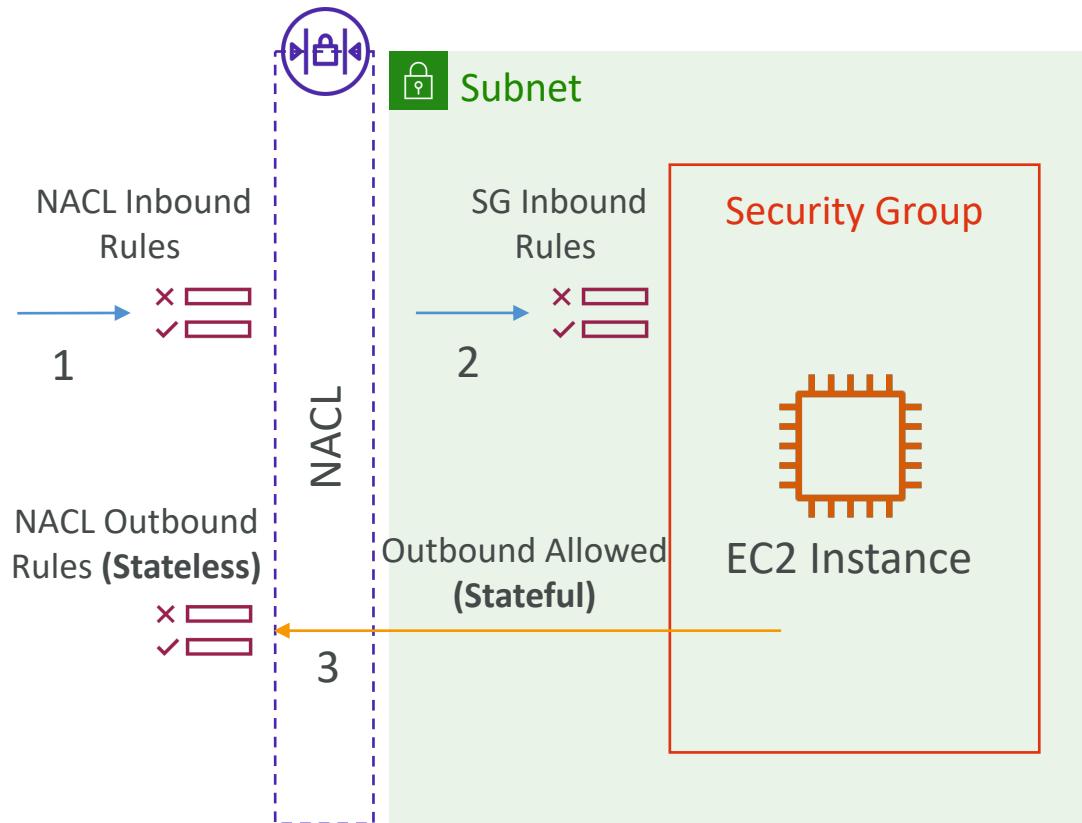


- Most secure & scalable way to expose a service to 1000s of VPC (own or other accounts)
- Does not require VPC peering, internet gateway, NAT, route tables...
- Requires a network load balancer (Service VPC) and ENI (Customer VPC) or GWLB
- If the NLB is in multiple AZ, and the ENIs in multiple AZ, the solution is fault tolerant!

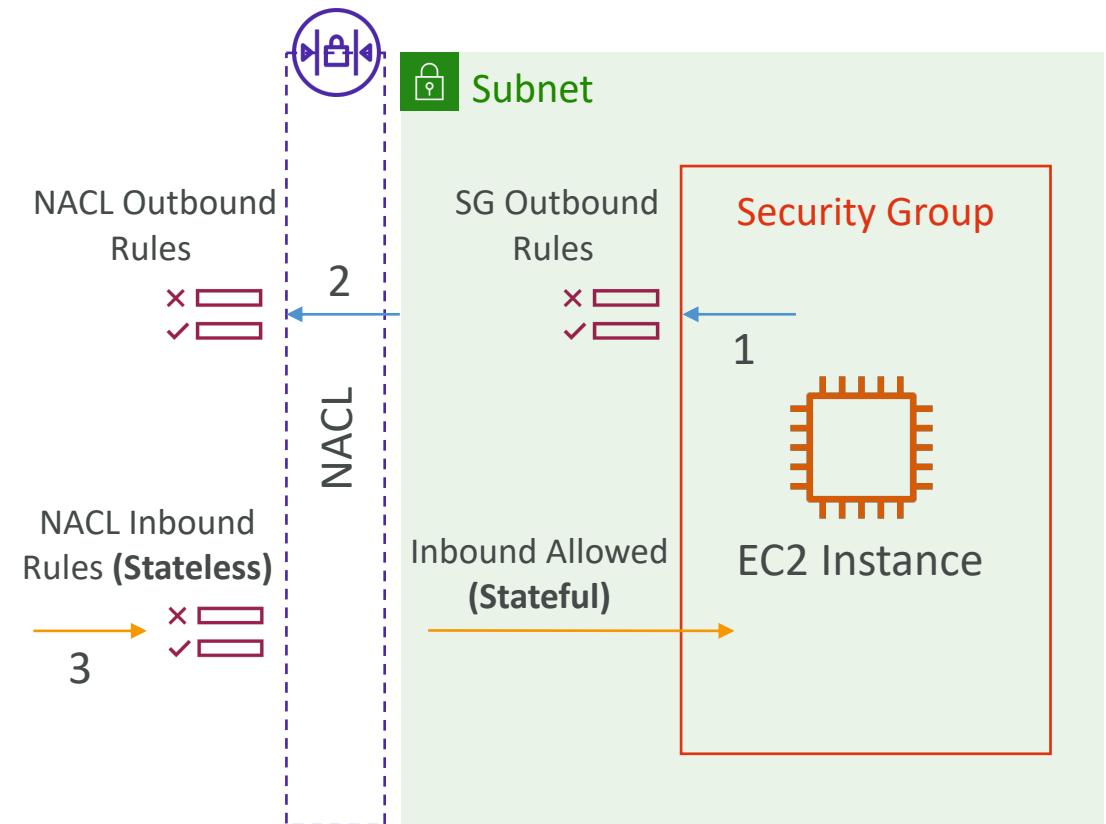


# Security Groups & NACLs

## Incoming Request



## Outgoing Request

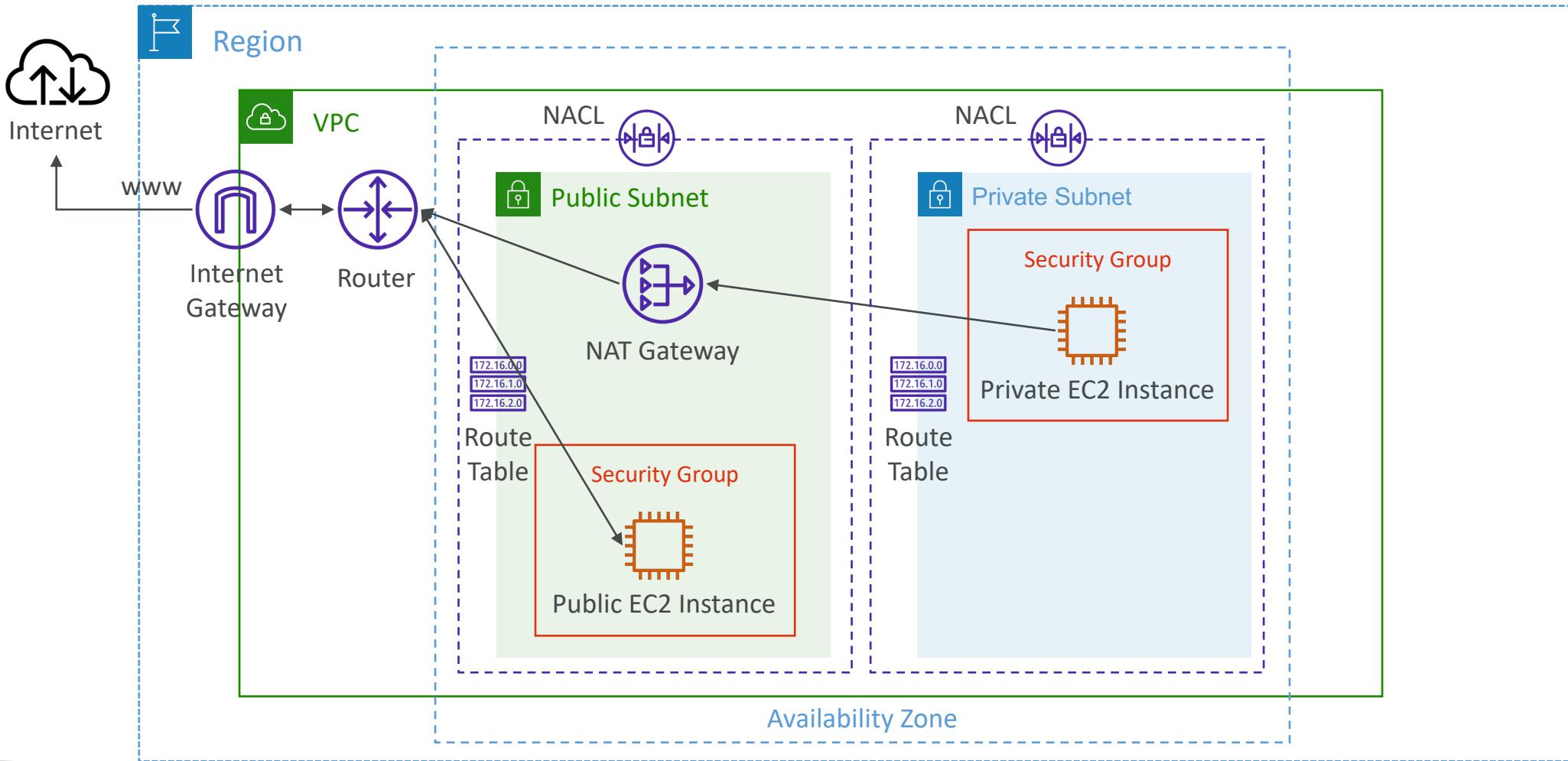




# Network Access Control List (NACL)

- NACL are like a firewall which control traffic from and to subnets
- One NACL per subnet, new subnets are assigned the Default NACL
- You define NACL Rules:
  - Rules have a number (1-32766), higher precedence with a lower number
  - First rule match will drive the decision
  - Example: if you define #100 ALLOW 10.0.0.10/32 and #200 DENY 10.0.0.10/32, the IP address will be allowed because 100 has a higher precedence over 200
  - The last rule is an asterisk (\*) and denies a request in case of no rule match
  - AWS recommends adding rules by increment of 100
- Newly created NACLs will deny everything
- NACL are a great way of blocking a specific IP address at the subnet level

# NACLs



# Default NACL

- Accepts everything inbound/outbound with the subnets it's associated with
- Do NOT modify the Default NACL, instead create custom NACLs



Default NACL for a VPC that supports IPv4

## Inbound Rules

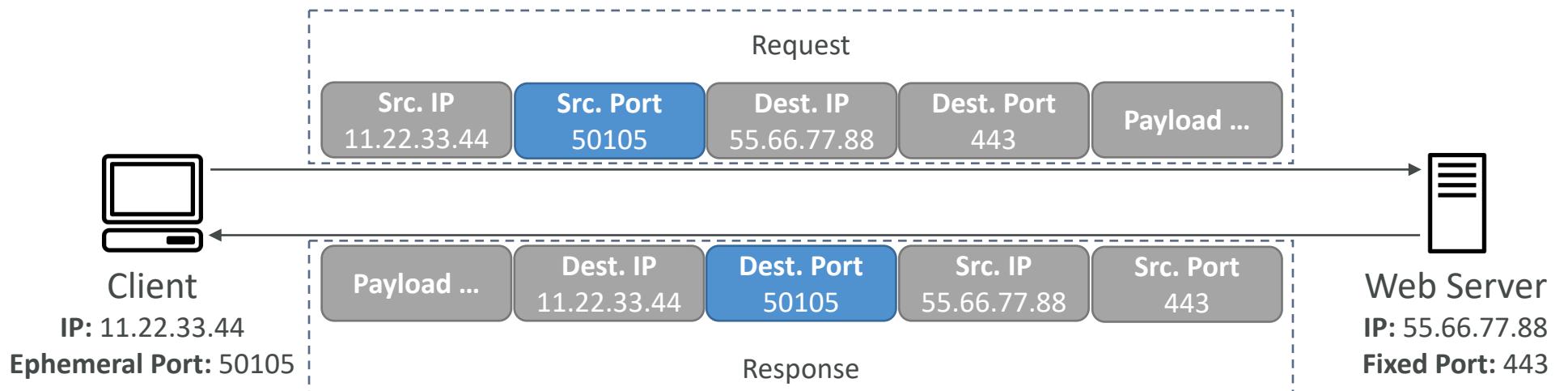
Rule #	Type	Protocol	Port Range	Source	Allow/Deny
100	All IPv4 Traffic	All	All	0.0.0.0/0	ALLOW
*	All IPv4 Traffic	All	All	0.0.0.0/0	DENY

## Outbound Rules

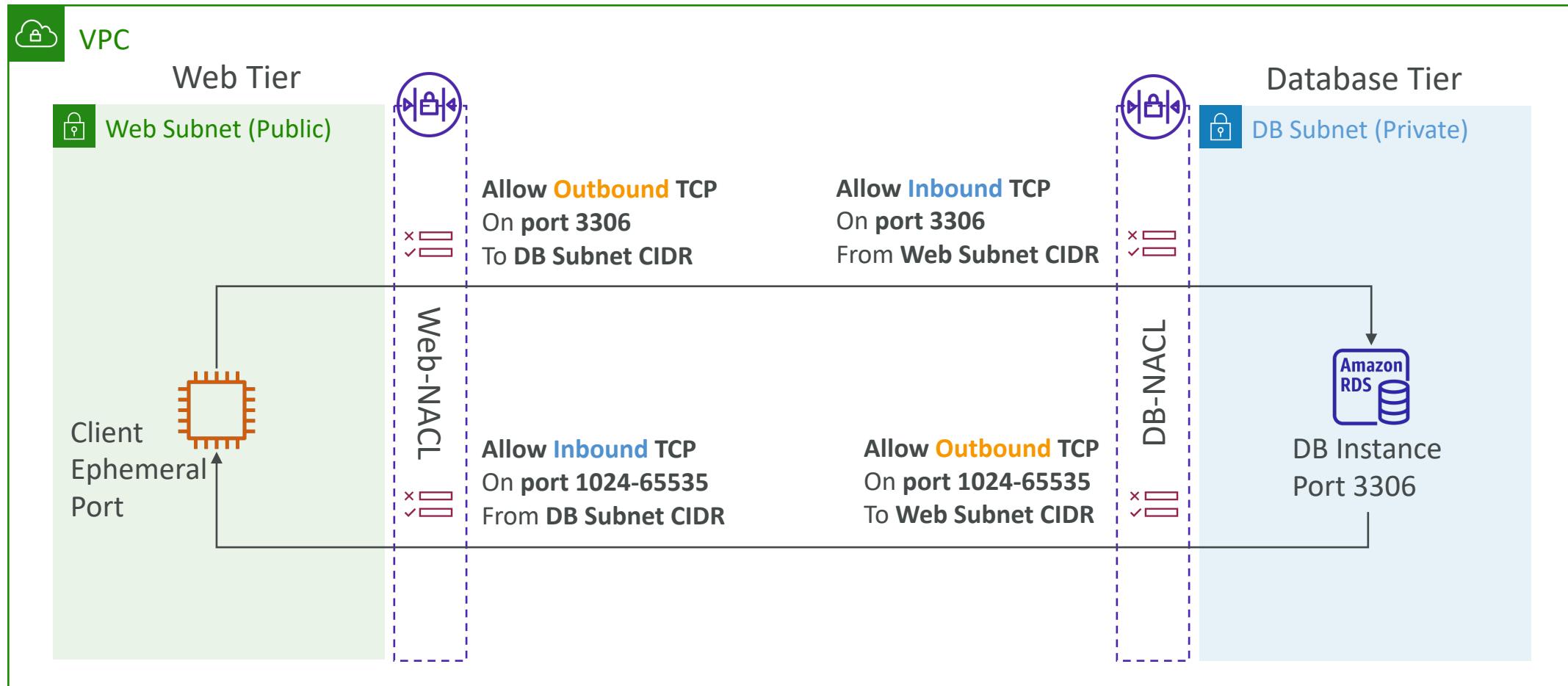
Rule #	Type	Protocol	Port Range	Destination	Allow/Deny
100	All IPv4 Traffic	All	All	0.0.0.0/0	ALLOW
*	All IPv4 Traffic	All	All	0.0.0.0/0	DENY

# Ephemeral Ports

- For any two endpoints to establish a connection, they must use ports
- Clients connect to a **defined port**, and expect a response on an **ephemeral port**
- Different Operating Systems use different port ranges, examples:
  - IANA & MS Windows 10 → 49152 – 65535
  - Many Linux Kernels → 32768 – 60999

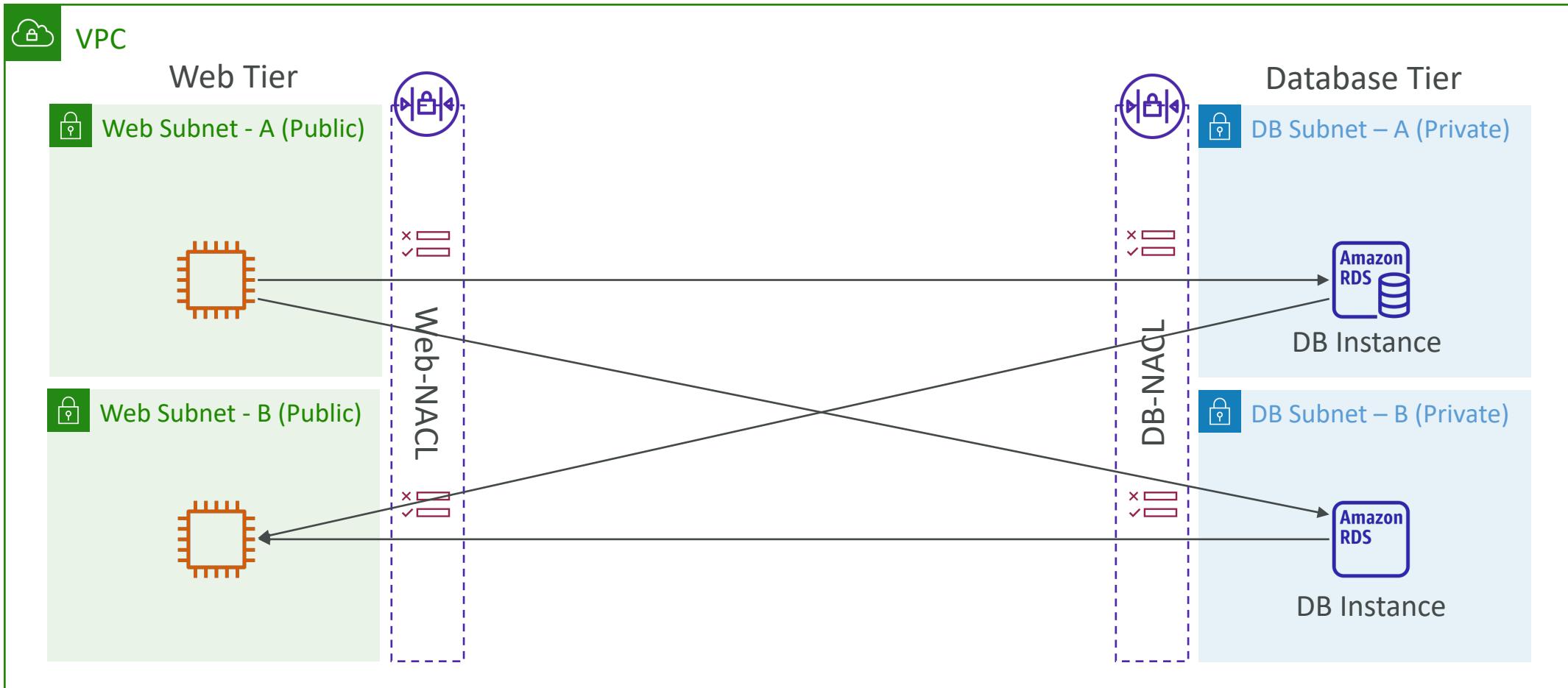


# NACL with Ephemeral Ports



<https://docs.aws.amazon.com/vpc/latest/userguide/vpc-network-acls.html#nacl-ephemeral-ports>

# Create NACL rules for each target subnets CIDR



# Security Group vs. NACLs

Security Group	NACL
Operates at the instance level	Operates at the subnet level
Supports allow rules only	Supports allow rules and deny rules
<b>Stateful:</b> return traffic is automatically allowed, regardless of any rules	<b>Stateless:</b> return traffic must be explicitly allowed by rules (think of ephemeral ports)
All rules are evaluated before deciding whether to allow traffic	Rules are evaluated in order (lowest to highest) when deciding whether to allow traffic, first match wins
Applies to an EC2 instance when specified by someone	Automatically applies to all EC2 instances in the subnet that it's associated with

NACL Examples: <https://docs.aws.amazon.com/vpc/latest/userguide/vpc-network-acls.html>

# Security Groups – Outbound Rules

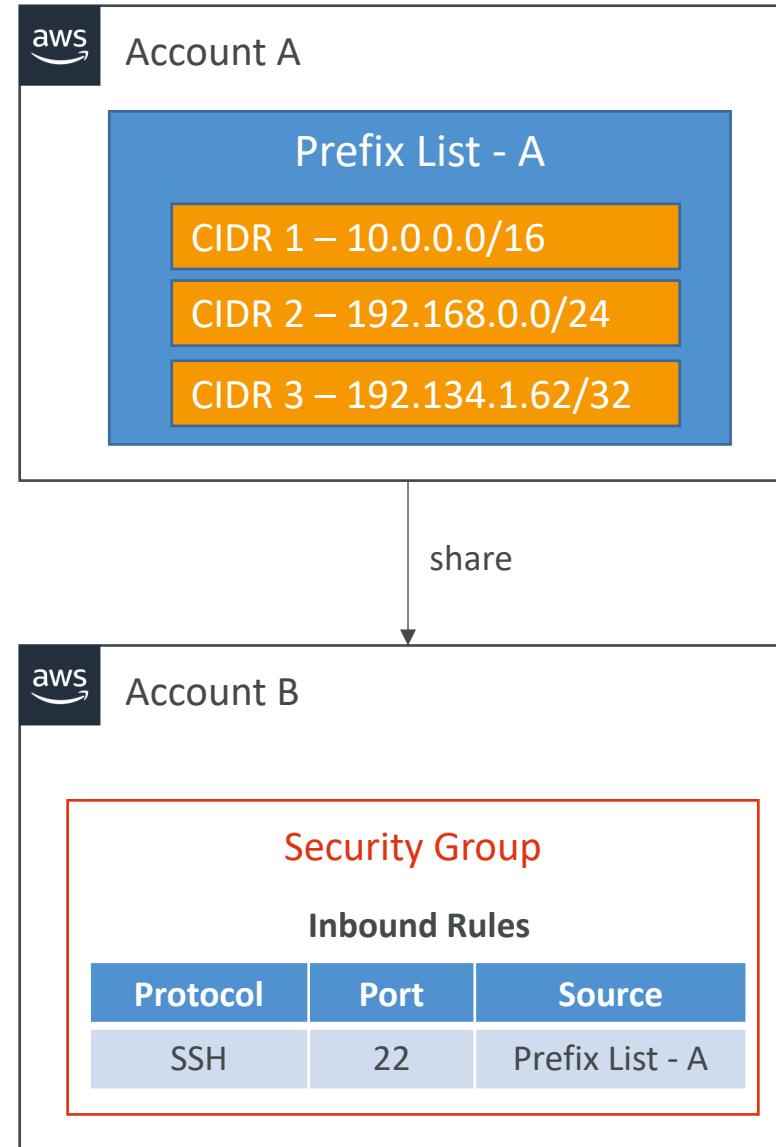
- Default is allowed 0.0.0.0/0 anywhere
- But we can remove and just allow specific prefixes

Outbound rules (1/1)									Manage tags	Edit outbound rules	
<input type="text"/> Filter security group rules								<	1	>	
<input checked="" type="checkbox"/>	Name ▾	Security group rule... ▾	IP version ▾	Type ▾	Protocol ▾	Port range ▾	Destination ▾	Description			
<input checked="" type="checkbox"/>	-	sgr-0ee382d4b7d379...	-	HTTPS	TCP	443	pl-63a5400a (com.amazonaws.us-east-1.s3)		-		

Allow Outbound traffic over port 443 to Amazon S3

# Managed Prefix List

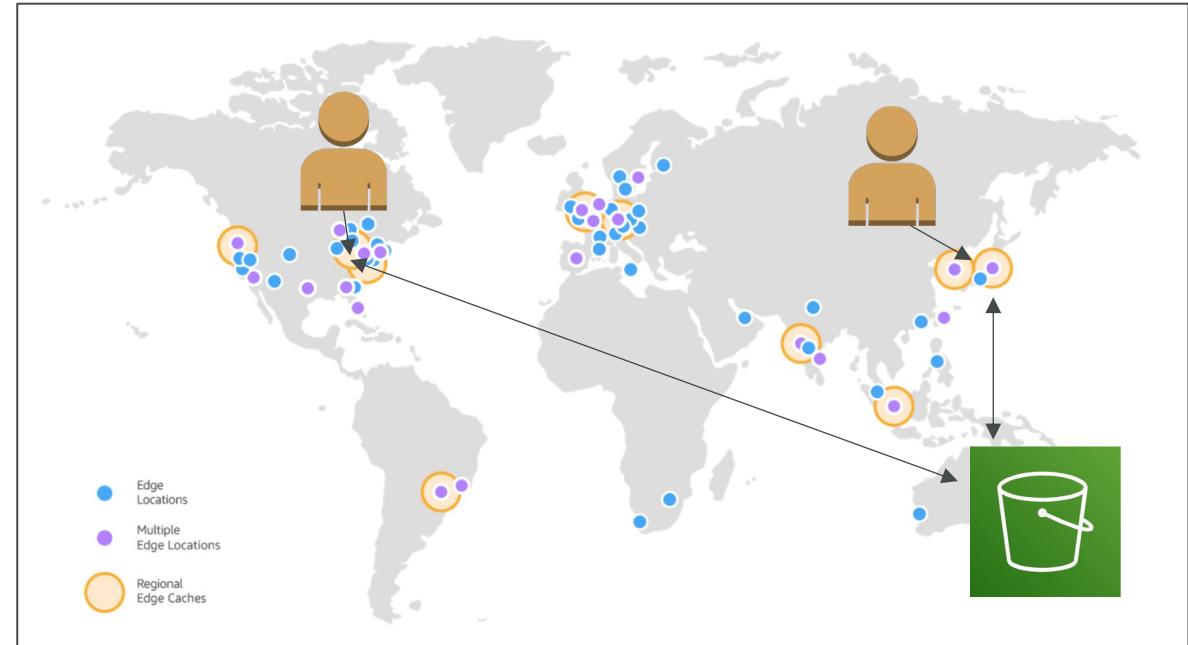
- A set of one or more CIDR blocks
- Makes it easier to configure and maintain Security Groups and Route Tables
- **Customer-Managed Prefix List**
  - Set of CIDRs that you define and managed by you
  - Can be shared with other AWS accounts or AWS Organization
  - Modify to update many Security Groups at once
- **AWS-Managed Prefix List**
  - Set of CIDRs for AWS services
  - You can't create, modify, share, or delete them
  - S3, CloudFront, DynamoDB, Ground Station...



# Amazon CloudFront



- Content Delivery Network (CDN)
- Improves read performance, content is cached at the edge
- Improves users experience
- 216 Point of Presence globally (edge locations)
- DDoS protection (because worldwide), integration with Shield, AWS Web Application Firewall

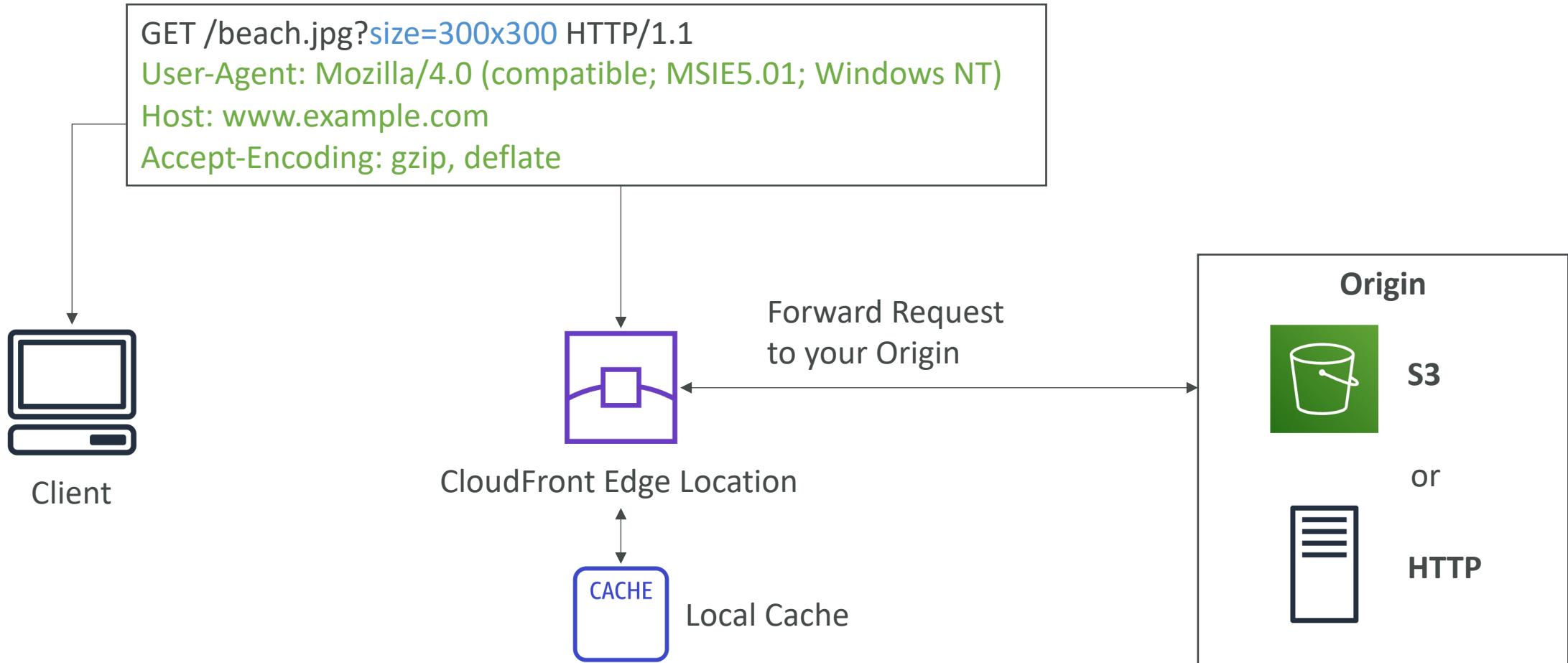


Source: <https://aws.amazon.com/cloudfront/features/?nc=sn&loc=2>

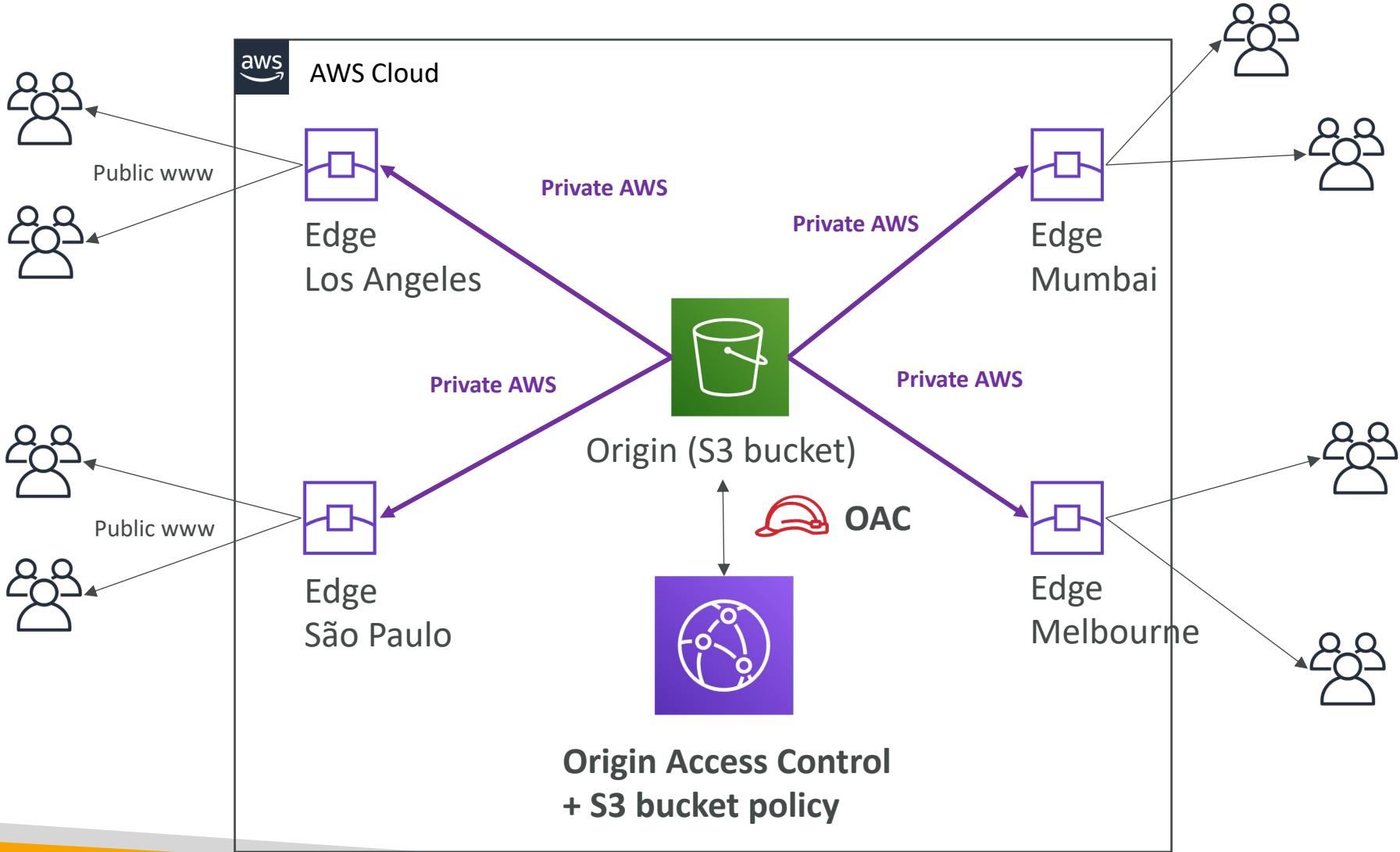
# CloudFront – Origins

- **S3 bucket**
  - For distributing files and caching them at the edge
  - Enhanced security with CloudFront Origin Access Control (OAC)
  - OAC is replacing Origin Access Identity (OAI)
  - CloudFront can be used as an ingress (to upload files to S3)
- **Custom Origin (HTTP)**
  - Application Load Balancer
  - EC2 instance
  - S3 website (must first enable the bucket as a static S3 website)
  - Any HTTP backend you want

# CloudFront at a high level



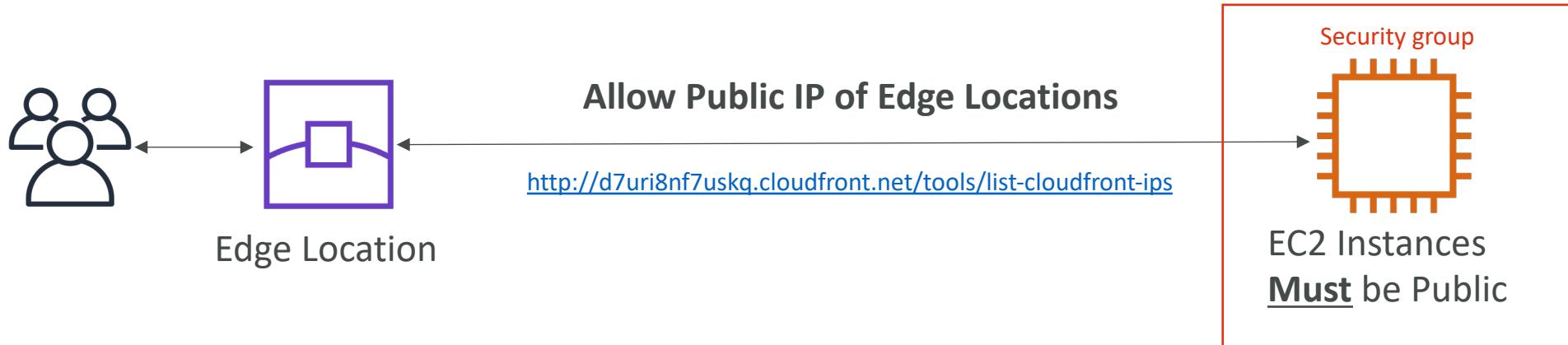
# CloudFront – S3 as an Origin



# CloudFront vs S3 Cross Region Replication

- CloudFront:
  - Global Edge network
  - Files are cached for a TTL (maybe a day)
  - Great for static content that must be available everywhere
- S3 Cross Region Replication:
  - Must be setup for each region you want replication to happen
  - Files are updated in near real-time
  - Read only
  - Great for dynamic content that needs to be available at low-latency in few regions

# CloudFront – ALB or EC2 as an origin



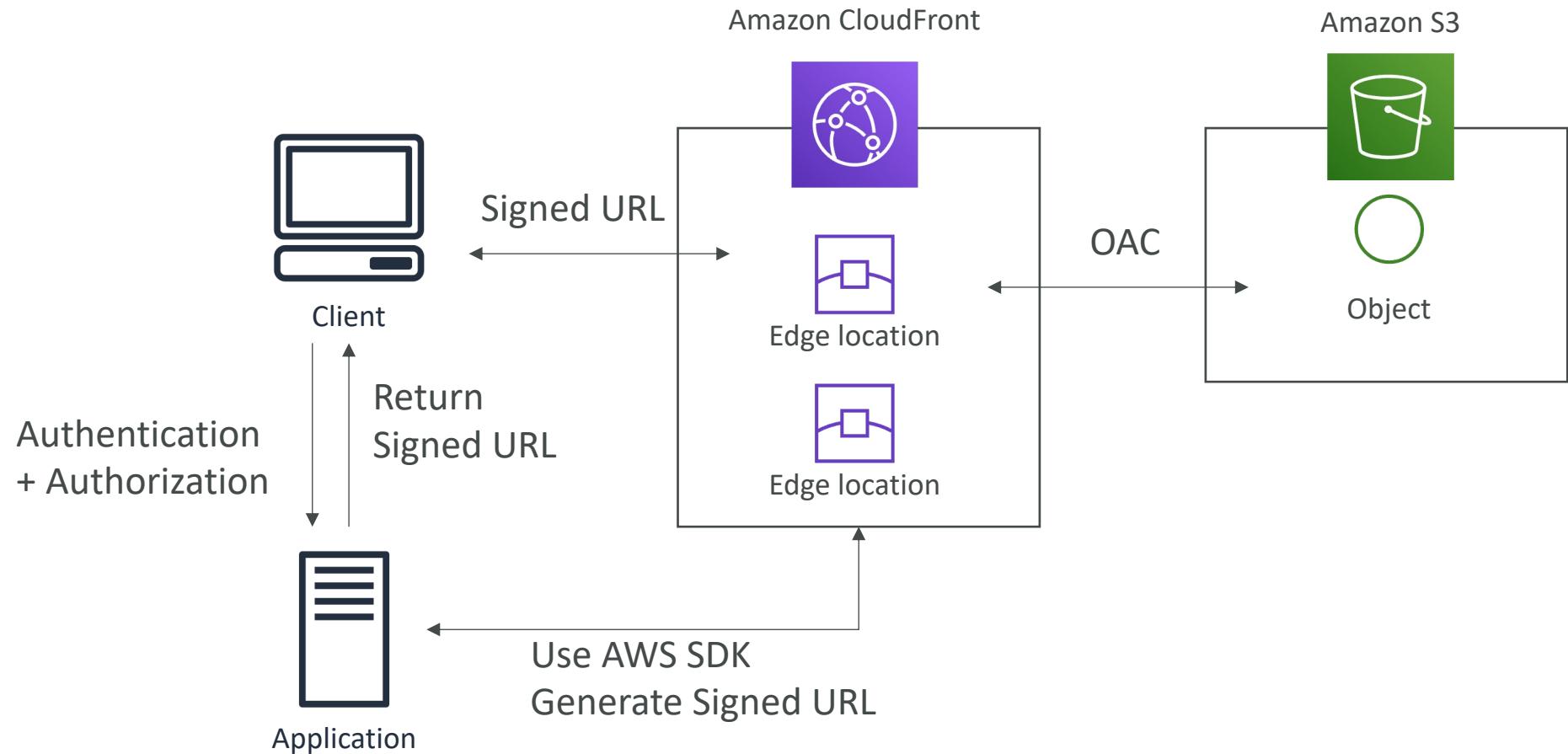
# CloudFront Geo Restriction

- You can restrict who can access your distribution
  - **Allowlist:** Allow your users to access your content only if they're in one of the countries on a list of approved countries.
  - **Blocklist:** Prevent your users from accessing your content if they're in one of the countries on a list of banned countries.
- The “country” is determined using a 3<sup>rd</sup> party Geo-IP database
- Use case: Copyright Laws to control access to content

# CloudFront Signed URL / Signed Cookies

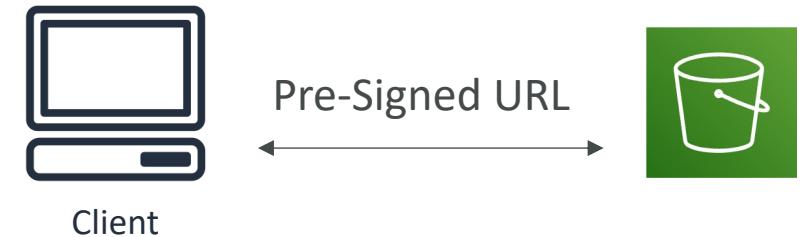
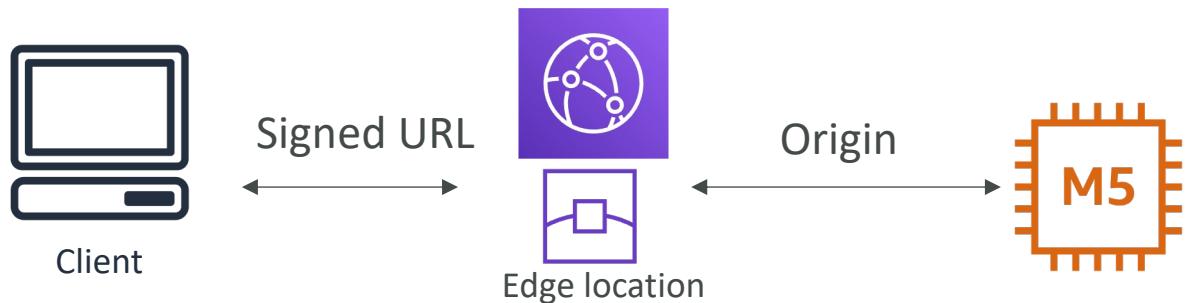
- You want to distribute paid shared content to premium users over the world
- To **Restrict Viewer Access**, we can create a CloudFront Signed URL / Cookie
- How long should the URL be valid for?
  - Shared content (movie, music): make it short (a few minutes)
  - Private content (private to the user): you can make it last for years
- Signed URL = access to individual files (one signed URL per file)
- Signed Cookies = access to multiple files (one signed cookie for many files)

# CloudFront Signed URL Diagram



# CloudFront Signed URL vs S3 Pre-Signed URL

- CloudFront Signed URL:
  - Allow access to a path, no matter the origin
  - Account wide key-pair, only the root can manage it
  - Can filter by IP, path, date, expiration
  - Can leverage caching features
- S3 Pre-Signed URL:
  - Issue a request as the person who pre-signed the URL
  - Uses the IAM key of the signing IAM principal
  - Limited lifetime

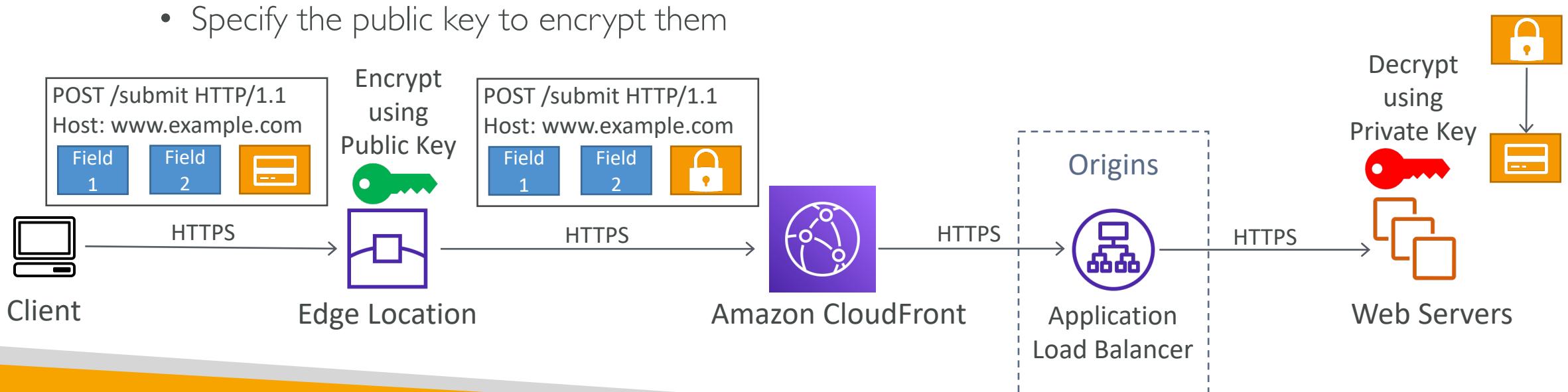


# CloudFront Signed URL Process

- Two types of signers:
  - Either a trusted key group (recommended)
    - Can leverage APIs to create and rotate keys (and IAM for API security)
  - An AWS Account that contains a CloudFront Key Pair
    - Need to manage keys using **the root account and the AWS console**
    - Not recommended because you shouldn't use the root account for this
- In your CloudFront distribution, create one or more **trusted key groups**
- You generate your own public / private key
  - The private key is used by your applications (e.g. EC2) to sign URLs
  - The public key (uploaded) is used by CloudFront to verify URLs

# CloudFront – Field Level Encryption

- Protect user sensitive information through application stack
- Adds an additional layer of security along with HTTPS
- Sensitive information encrypted at the edge close to user
- Uses asymmetric encryption
- Usage:
  - Specify set of fields in POST requests that you want to be encrypted (up to 10 fields)
  - Specify the public key to encrypt them



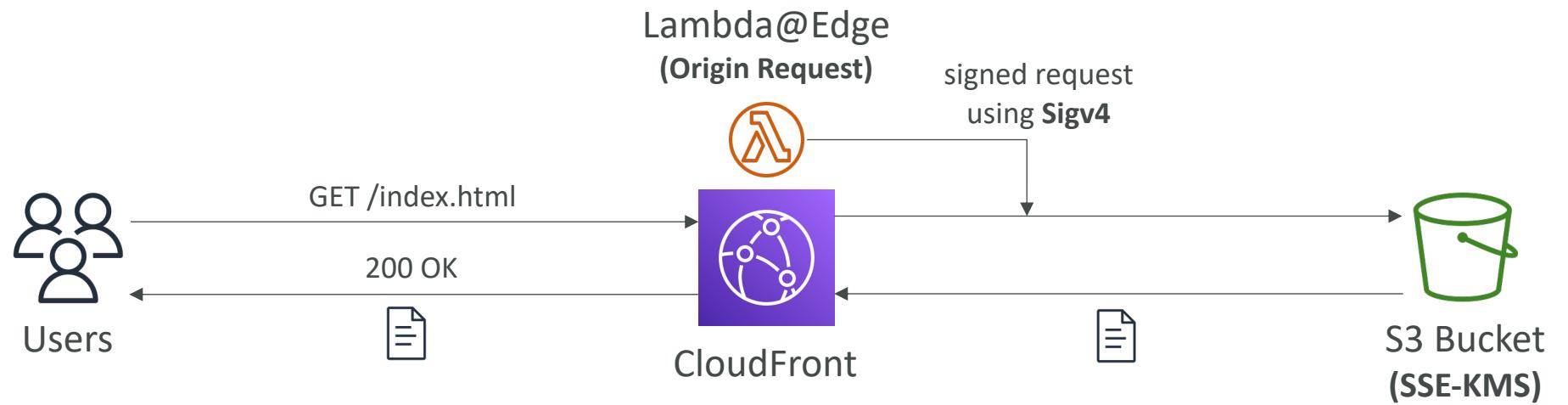
# Origin Access Control with SSE-KMS

- OAC supports SSE-KMS natively (as requests are signed with Sigv4)
- Add a statement to the KMS Key Policy to authorize the OAC

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Principal": {  
                "AWS": "arn:aws:iam::111122223333:root",  
                "Service": "cloudfront.amazonaws.com"  
            },  
            "Action": [  
                "kms:Decrypt",  
                "kms:Encrypt",  
                "kms:GenerateDataKey*"  
            ],  
            "Resource": "*",  
            "Condition": {  
                "StringEquals": {  
                    "AWS:SourceArn": "arn:aws:cloudfront::distribution/EDFDVBD6EXAMPLE"  
                }  
            }  
        }  
    ]  
}
```

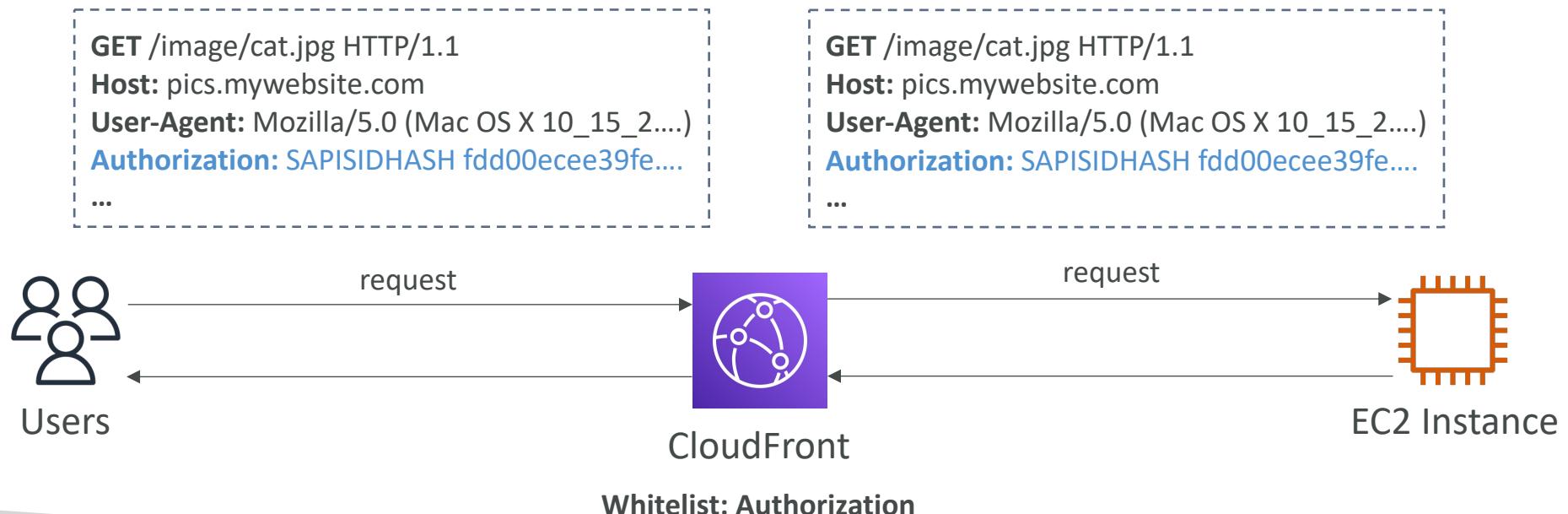
# Origin Access Identity with SSE-KMS

- OAI doesn't support SSE-KMS natively (only SSE-S3)
- Use **Lambda@Edge** to sign requests from CloudFront to S3
- Make sure to disable OAI for this to work



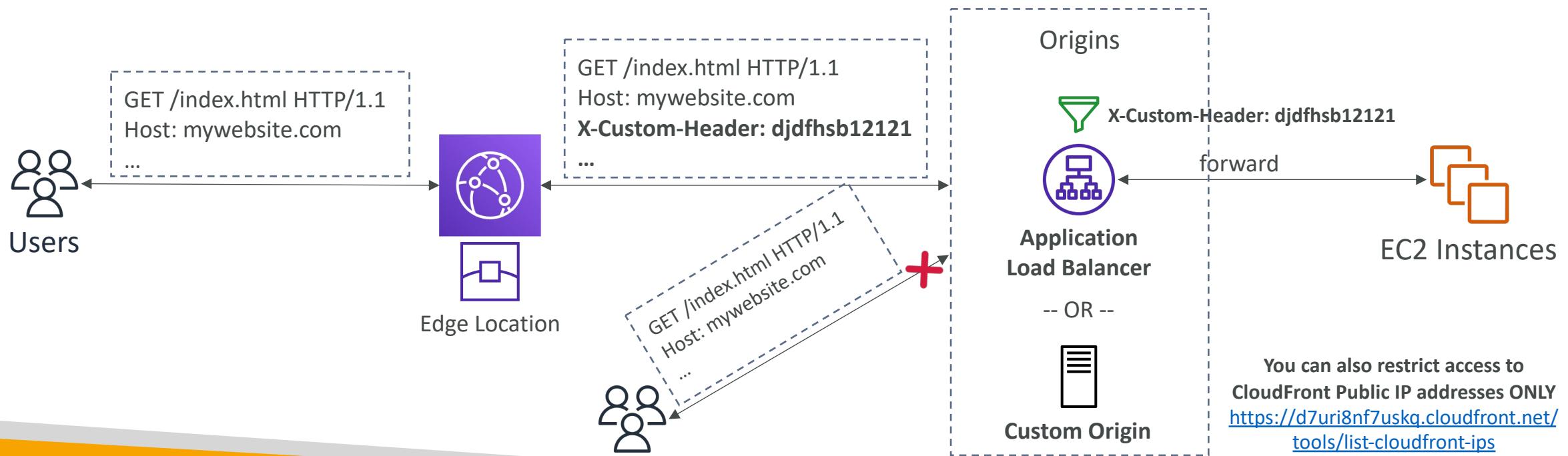
# CloudFront Authorization Header

- Configure CloudFront distribution to forward the Authorization header using Cache Policy
- Not supported for S3 Origins

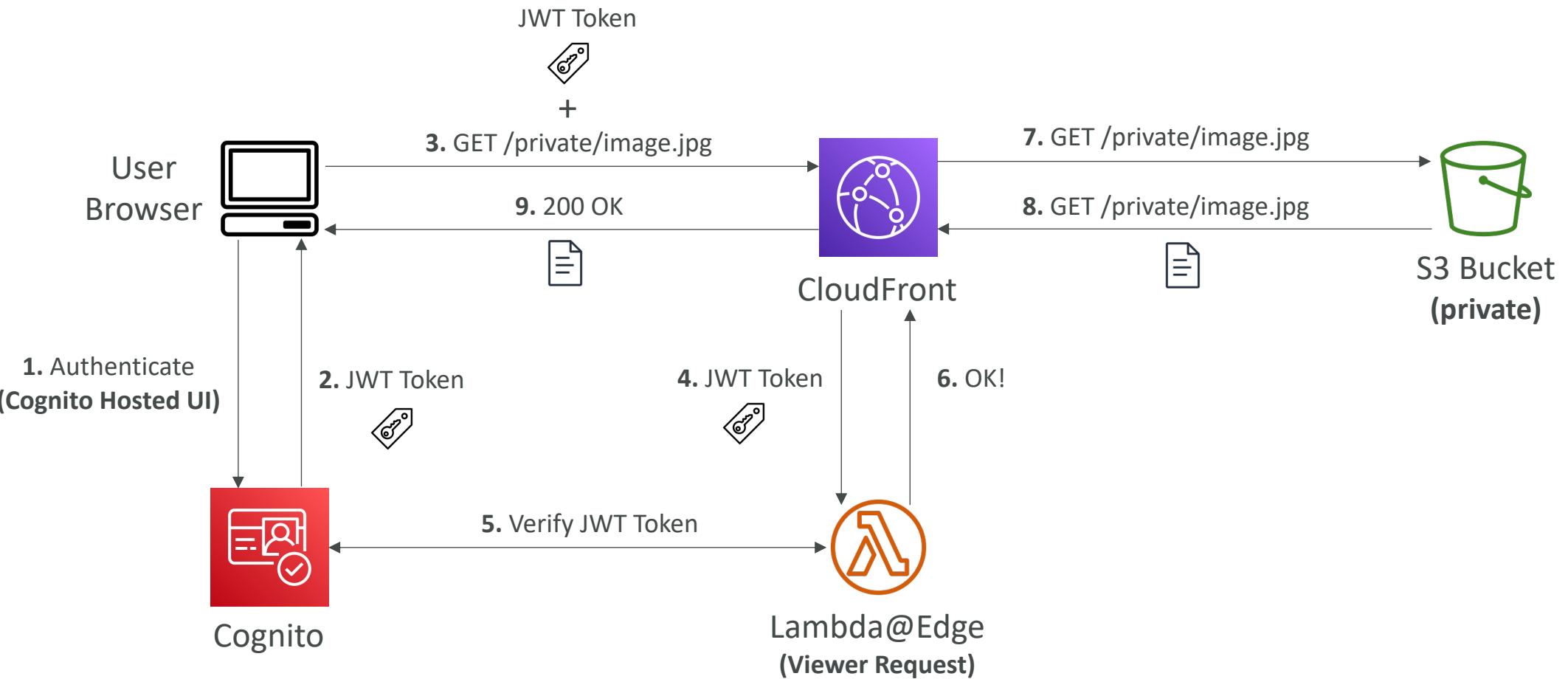


# CloudFront – Restrict access to ALB

- Prevent direct access to your ALB or Custom Origins (only access through CloudFront)
- First, configure CloudFront to add a **Custom HTTP Header** to requests it sends to the ALB
- Second, configure the ALB to only forward requests that contain that Custom HTTP Header
- Keep the custom header name and value secret!



# CloudFront – Integration with Cognito



# AWS WAF – Web Application Firewall



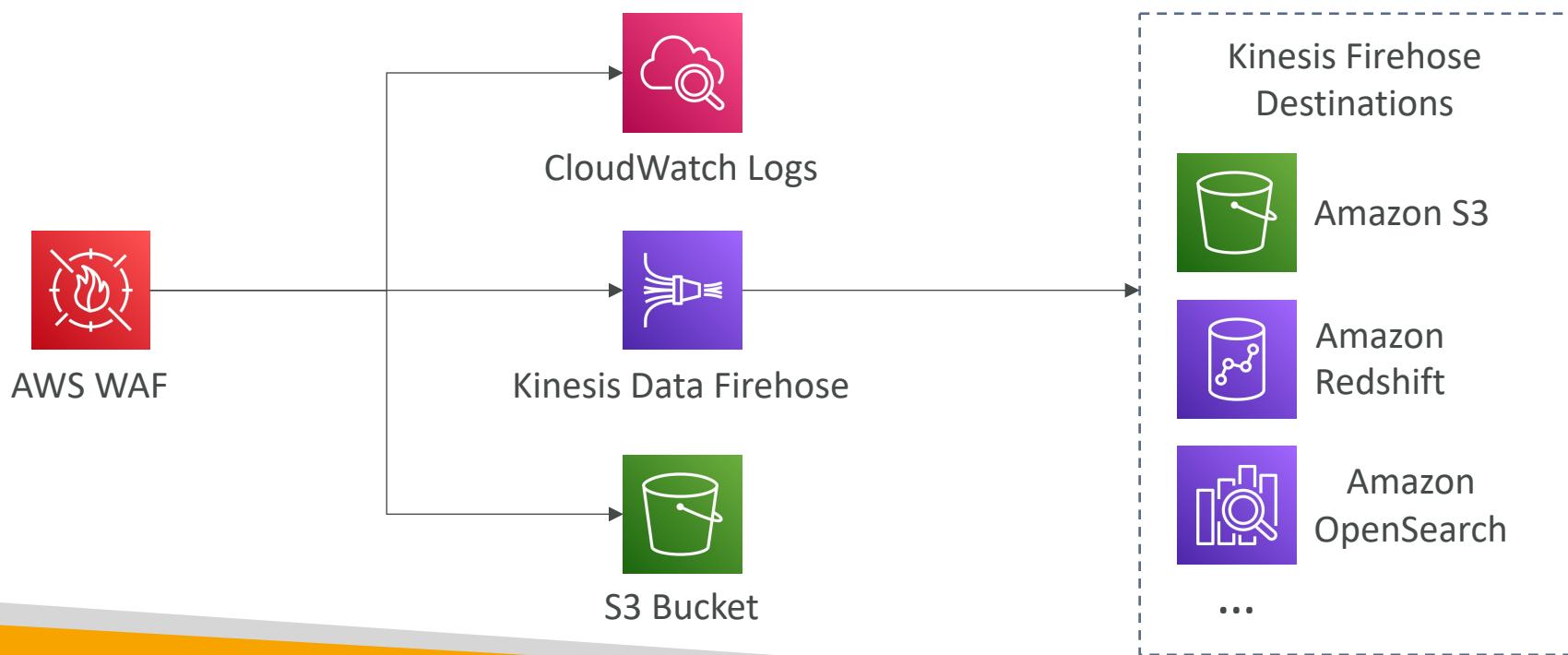
- Protects your web applications from common web exploits (Layer 7)
- Deploy on **Application Load Balancer** (localized rules)
- Deploy on **API Gateway** (rules running at the regional or edge level)
- Deploy on **CloudFront** (rules globally on edge locations)
  - Used to front other solutions: CLB, EC2 instances, custom origins, S3 websites
- Deploy on AppSync (protect your GraphQL APIs)
- WAF is not for DDoS protection
- Define Web ACL (Web Access Control List):
  - Rules can include IP addresses, HTTP headers, HTTP body, or URI strings
  - Protects from common attack - SQL injection and Cross-Site Scripting (XSS)
  - Size constraints, Geo match
  - Rate-based rules (to count occurrences of events)
- Rule Actions: Count | Allow | Block | CAPTCHA

# AWS WAF – Managed Rules

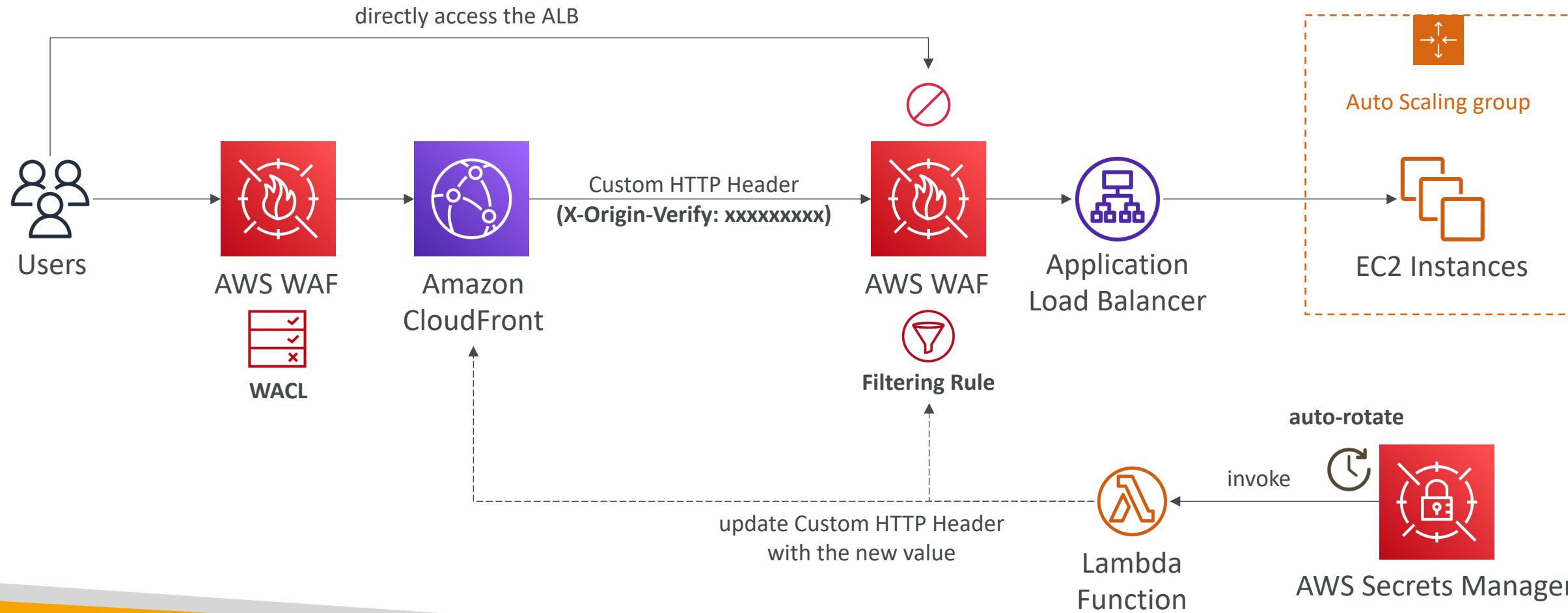
- Library of over 190 managed rules
- Ready-to-use rules that are managed by AWS and AWS Marketplace Sellers
- **Baseline Rule Groups** – general protection from common threats
  - AWSManagedRulesCommonRuleSet, AWSManagedRulesAdminProtectionRuleSet, ...
- **Use-case Specific Rule Groups** – protection for many AWS WAF use cases
  - AWSManagedRulesSQLiRuleSet, AWSManagedRulesWindowsRuleSet, AWSManagedRulesPHPRuleSet, AWSManagedRulesWordPressRuleSet, ...
- **IP Reputation Rule Groups** – block requests based on source (e.g., malicious IPs)
  - AWSManagedRulesAmazonIpReputationList, AWSManagedRulesAnonymousIpList
- **Bot Control Managed Rule Group** – block and manage requests from bots
  - AWSManagedRulesBotControlRuleSet

# WAF - Web ACL – Logging

- You can send your logs to an:
  - Amazon CloudWatch Logs log group – 5 MB per second
  - Amazon Simple Storage Service (Amazon S3) bucket – 5 minutes interval
  - Amazon Kinesis Data Firehose – limited by Firehose quotas



# Solution Architecture – Enhance CloudFront Origin Security with AWS WAF & AWS Secrets Manager



# AWS Shield: protect from DDoS attack



- DDoS: Distributed Denial of Service – many requests at the same time
- AWS Shield Standard:
  - Free service that is activated for every AWS customer
  - Provides protection from attacks such as SYN/UDP Floods, Reflection attacks and other layer 3/layer 4 attacks
- AWS Shield Advanced:
  - Optional DDoS mitigation service (\$3,000 per month per organization)
  - Protect against more sophisticated attack on [Amazon EC2](#), [Elastic Load Balancing \(ELB\)](#), [Amazon CloudFront](#), [AWS Global Accelerator](#), and [Route 53](#)
  - 24/7 access to AWS DDoS response team (DRP)
  - Protect against higher fees during usage spikes due to DDoS
  - Shield Advanced automatic application layer DDoS mitigation automatically creates, evaluates and deploys AWS WAF rules to mitigate layer 7 attacks

# AWS Firewall Manager



- Manage rules in all accounts of an AWS Organization
- Security policy: common set of security rules
  - WAF rules (Application Load Balancer, API Gateways, CloudFront)
  - AWS Shield Advanced (ALB, CLB, NLB, Elastic IP, CloudFront)
  - Security Groups for EC2, Application Load Balancer and ENI resources in VPC
  - AWS Network Firewall (VPC Level)
  - Amazon Route 53 Resolver DNS Firewall
  - Policies are created at the region level
- Rules are applied to new resources as they are created (good for compliance) across all and future accounts in your Organization

# WAF vs. Firewall Manager vs. Shield



AWS WAF



AWS Firewall Manager



AWS Shield

- WAF, Shield and Firewall Manager are used together for comprehensive protection
- Define your Web ACL rules in WAF
- For granular protection of your resources, WAF alone is the correct choice
- If you want to use AWS WAF across accounts, accelerate WAF configuration, automate the protection of new resources, use Firewall Manager with AWS WAF
- Shield Advanced adds additional features on top of AWS WAF, such as dedicated support from the Shield Response Team (SRT) and advanced reporting.
- If you're prone to frequent DDoS attacks, consider purchasing Shield Advanced

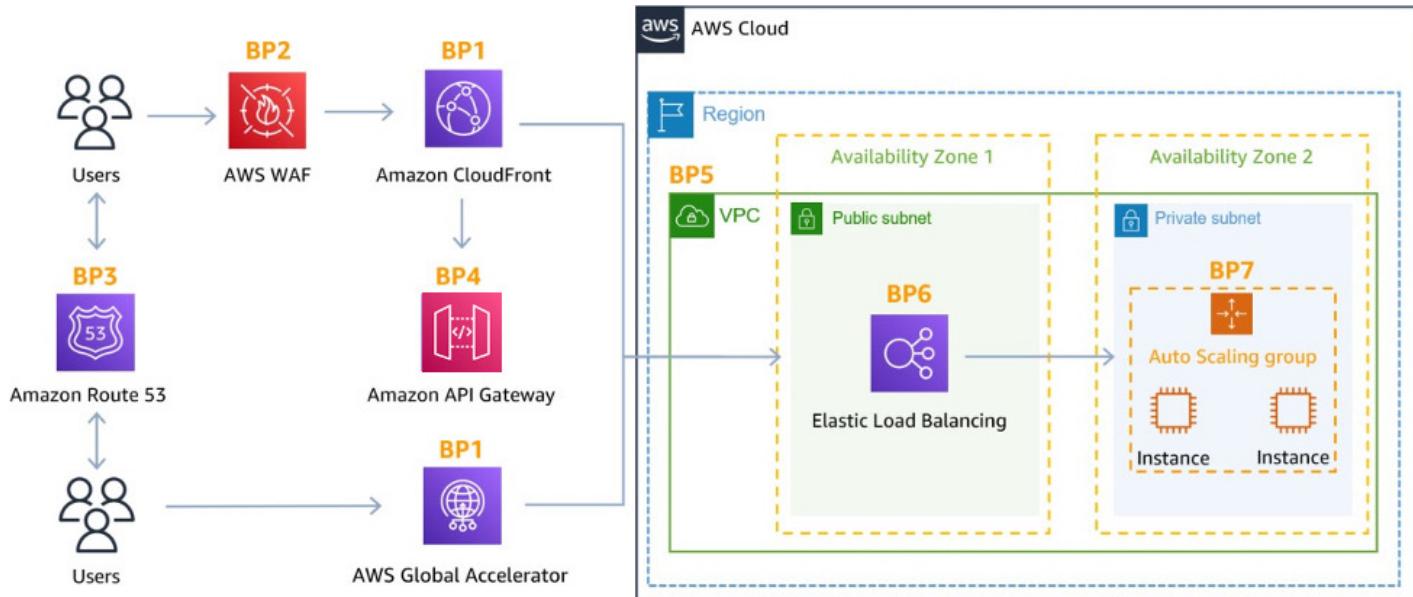
# Shield Advanced CloudWatch Metrics

- Helps you to detect if there's a DDoS attack happening
- **DDoSDetected** – indicates whether a DDoS event is happening for a specific resource
- **DDoSAttackBitsPerSecond** – number of bits per second during a DDoS event for a specific resource
- **DDoSAttackPacketsPerSecond** – number of packets per second during a DDoS event for a specific resource
- **DDoSAttackRequestsPerSecond** – number of requests per second during a DDoS event for a specific resource

# AWS Best Practices for DDoS Resiliency

## Edge Location Mitigation (BP1, BP3)

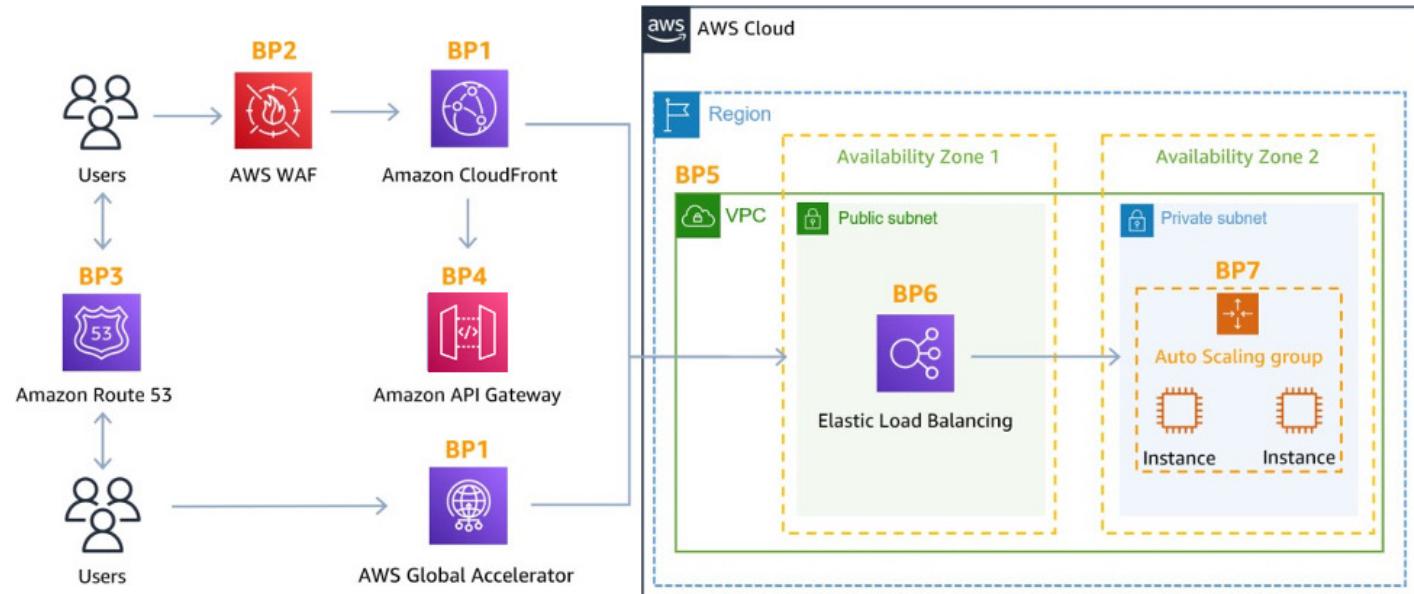
- **BP1 – CloudFront**
  - Web Application delivery at the edge
  - Protect from DDoS Common Attacks (SYN floods, UDP reflection...)
- **BP1 – Global Accelerator**
  - Access your application from the edge
  - Integration with Shield for DDoS protection
  - Helpful if your backend is not compatible with CloudFront
- **BP3 – Route 53**
  - Domain Name Resolution at the edge
  - DDoS Protection mechanism



# AWS Best Practices for DDoS Resiliency

## Best practices for DDoS mitigation

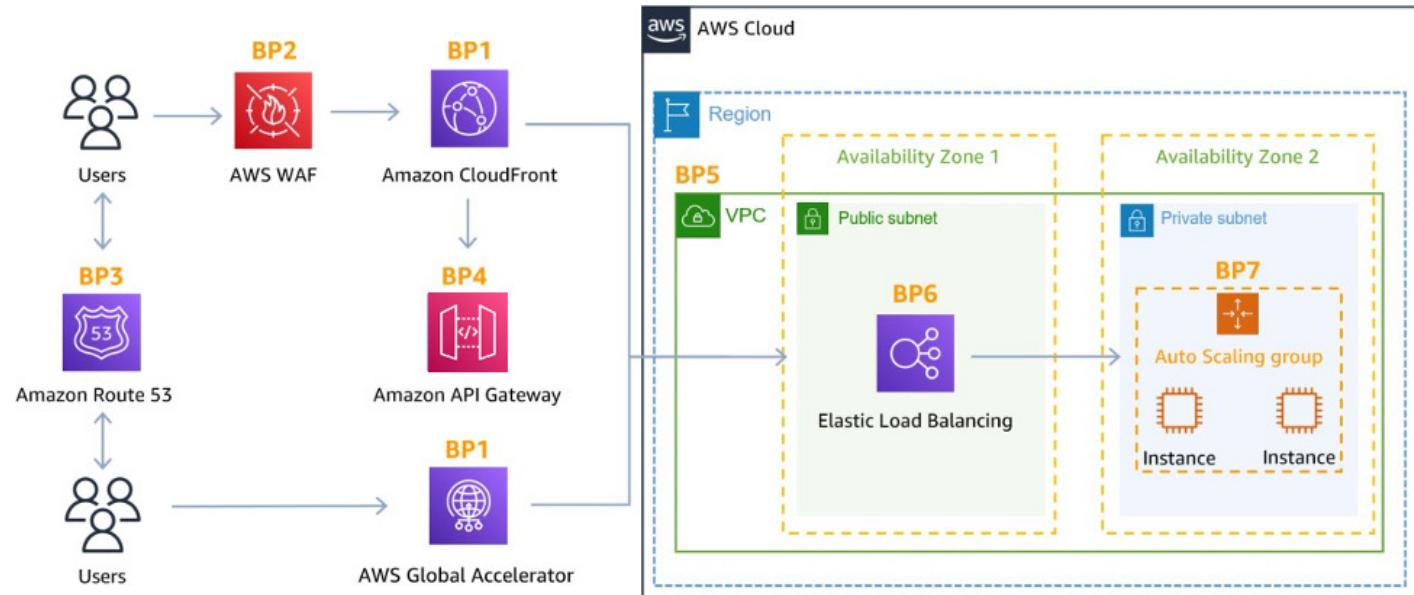
- Infrastructure layer defense (BP1, BP3, BP6)
  - Protect Amazon EC2 against high traffic
  - That includes using Global Accelerator, Route 53, CloudFront, Elastic Load Balancing
- Amazon EC2 with Auto Scaling (BP7)
  - Helps scale in case of sudden traffic surges including a flash crowd or a DDoS attack
- Elastic Load Balancing (BP6)
  - Elastic Load Balancing scales with the traffic increases and will distribute the traffic to many EC2 instances



# AWS Best Practices for DDoS Resiliency

## Application Layer Defense

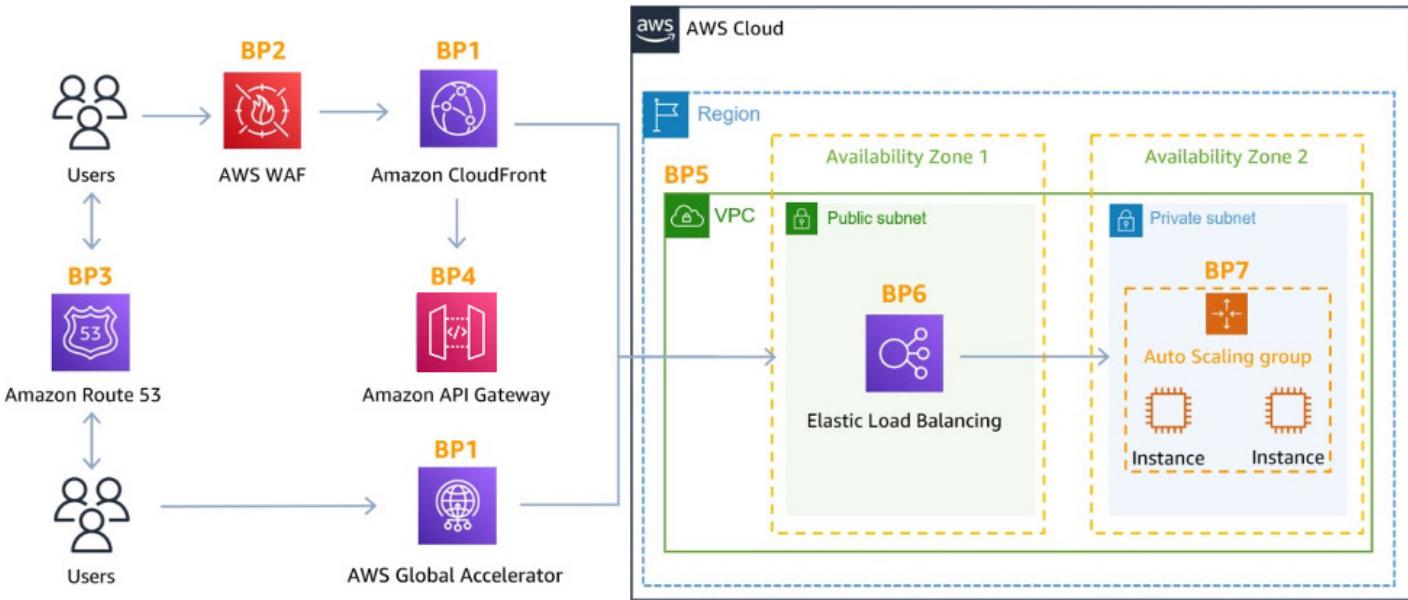
- Detect and filter malicious web requests (BP1, BP2)
  - CloudFront cache static content and serve it from edge locations, protecting your backend
  - AWS WAF is used on top of CloudFront and Application Load Balancer to filter and block requests based on request signatures
  - WAF rate-based rules can automatically block the IPs of bad actors
  - Use managed rules on WAF to block attacks based on IP reputation, or block anonymous IPs
  - CloudFront can block specific geographies
- Shield Advanced (BP1, BP2, BP6)
  - Shield Advanced automatic application layer DDoS mitigation automatically creates, evaluates and deploys AWS WAF rules to mitigate layer 7 attacks



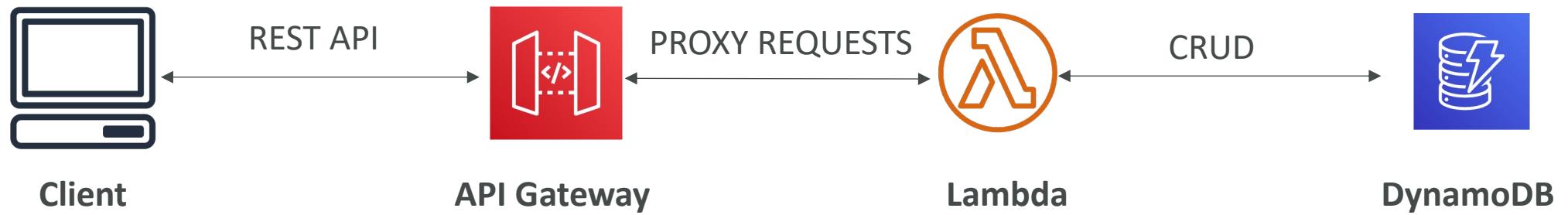
# AWS Best Practices for DDoS Resiliency

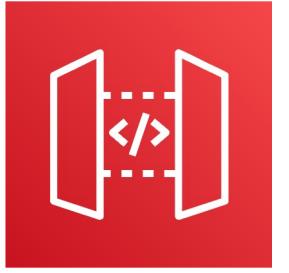
## Attack surface reduction

- Obfuscating AWS resources (BP1, BP4, BP6)
  - Using CloudFront, API Gateway, Elastic Load Balancing to hide your backend resources (Lambda functions, EC2 instances)
- Security groups and Network ACLs (BP5)
  - Use security groups and NACLs to filter traffic based on specific IP at the subnet or ENI-level
  - Elastic IP are protected by AWS Shield Advanced
- Protecting API endpoints (BP4)
  - Hide EC2, Lambda, elsewhere
  - Edge-optimized mode, or CloudFront + regional mode (more control for DDoS)
  - WAF + API Gateway: burst limits, headers filtering, use API keys



# Example: Building a Serverless API





# AWS API Gateway

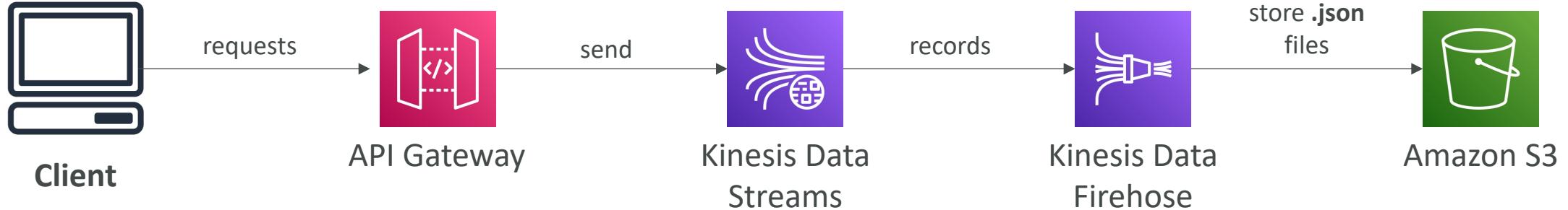
- AWS Lambda + API Gateway: No infrastructure to manage
- Support for the WebSocket Protocol
- Handle API versioning (v1, v2...)
- Handle different environments (dev, test, prod...)
- Handle security (Authentication and Authorization)
- Create API keys, handle request throttling
- Swagger / Open API import to quickly define APIs
- Transform and validate requests and responses
- Generate SDK and API specifications
- Cache API responses

# API Gateway – Integrations High Level

- Lambda Function
  - Invoke Lambda function
  - Easy way to expose REST API backed by AWS Lambda
- HTTP
  - Expose HTTP endpoints in the backend
  - Example: internal HTTP API on premise, Application Load Balancer...
  - Why? Add rate limiting, caching, user authentications, API keys, etc...
- AWS Service
  - Expose any AWS API through the API Gateway
  - Example: start an AWS Step Function workflow, post a message to SQS
  - Why? Add authentication, deploy publicly, rate control...

# API Gateway – AWS Service Integration

## Kinesis Data Streams example



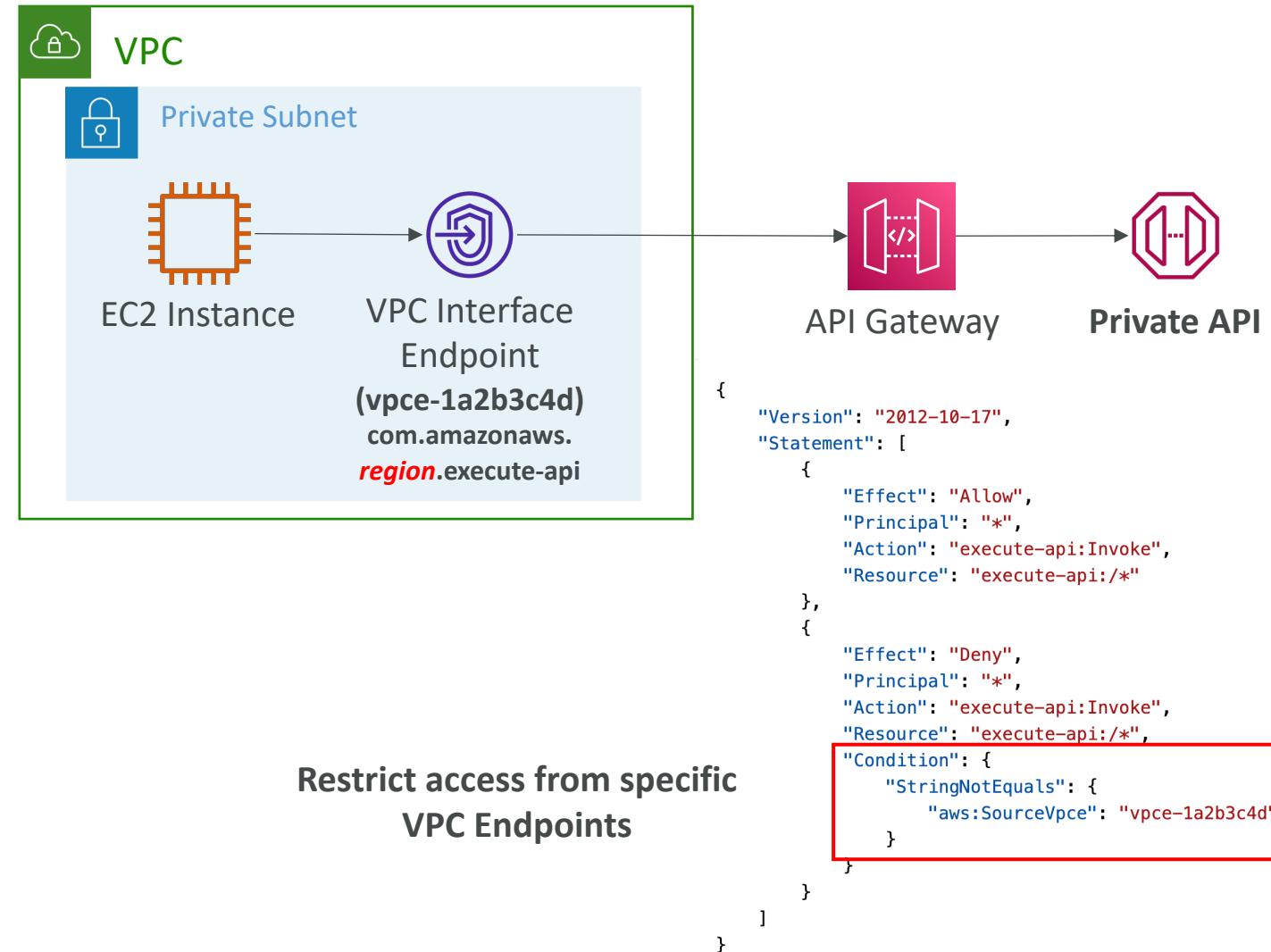
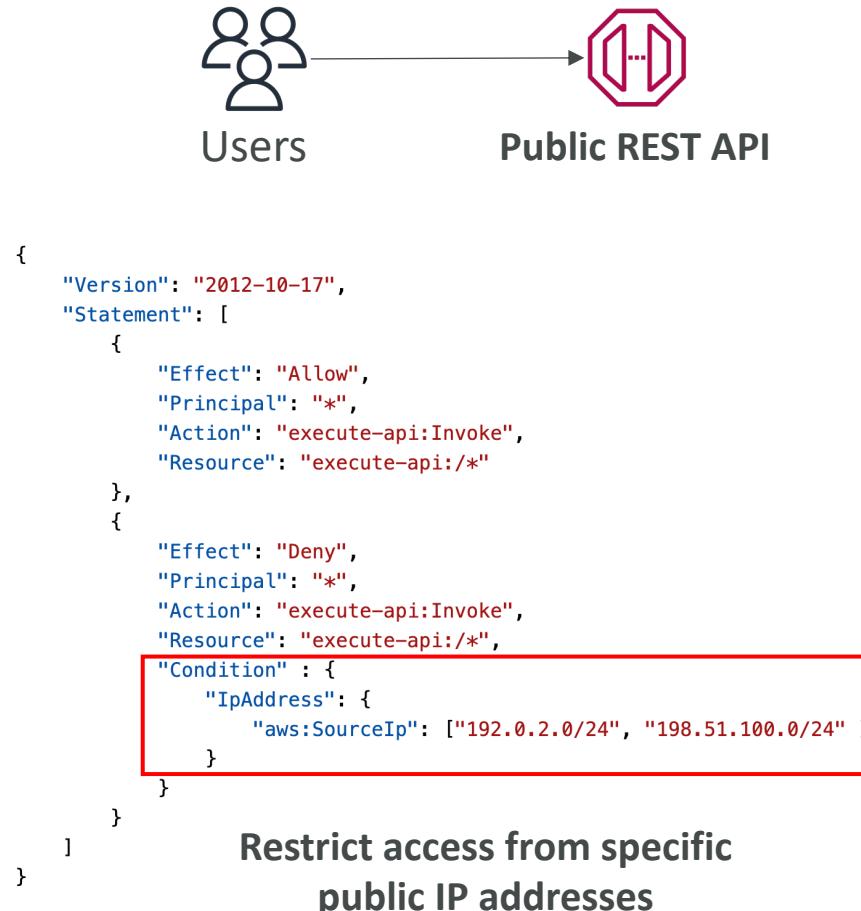
# API Gateway - Endpoint Types

- **Edge-Optimized (default):** For global clients
  - Requests are routed through the CloudFront Edge locations (improves latency)
  - The API Gateway still lives in only one region
- **Regional:**
  - For clients within the same region
  - Could manually combine with CloudFront (more control over the caching strategies and the distribution)
- **Private:**
  - Can only be accessed from your VPC using an interface VPC endpoint (ENI)
  - Use a resource policy to define access

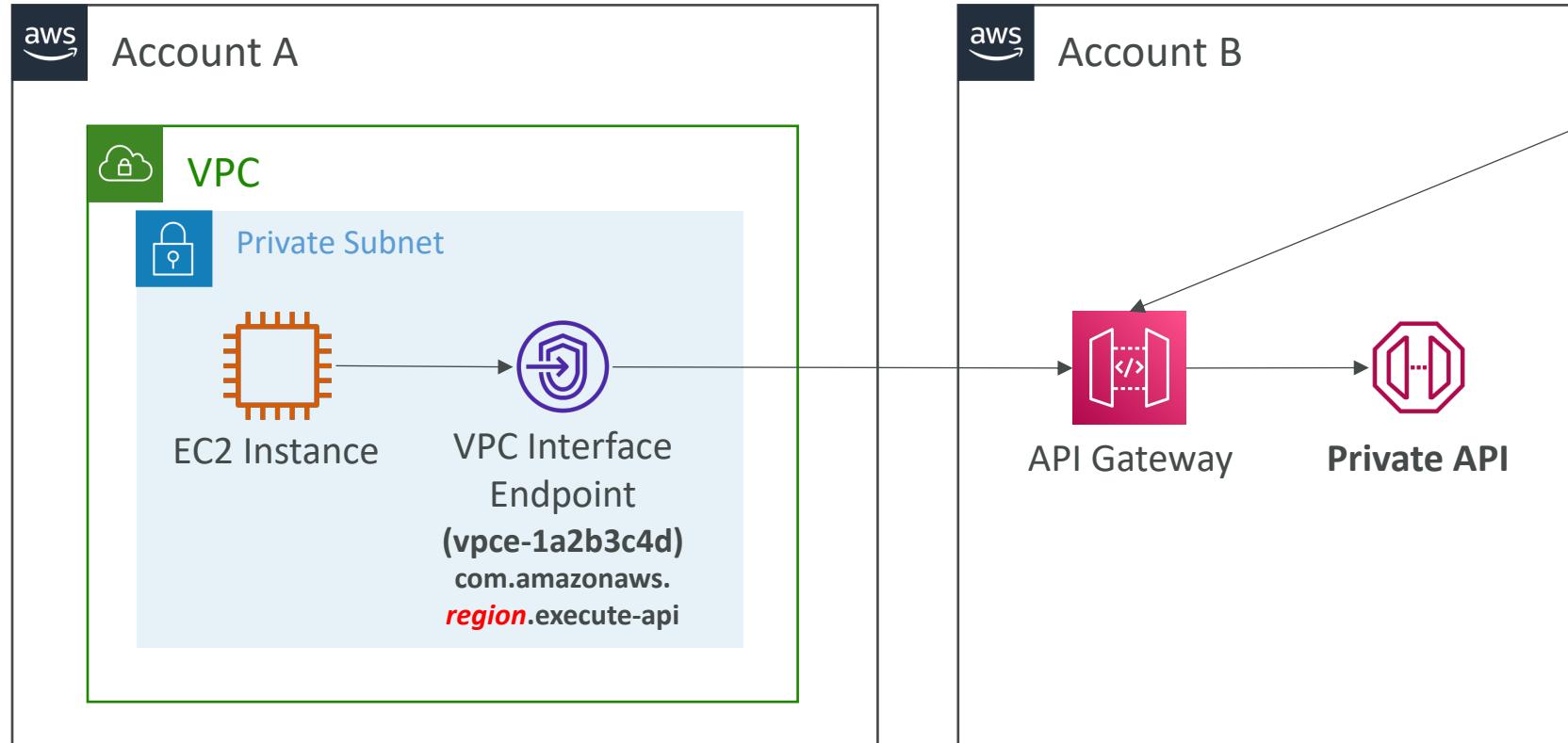
# API Gateway – Security

- User Authentication through
  - IAM Roles (useful for internal applications)
  - Cognito (identity for external users – example mobile users)
  - Custom Authorizer (your own logic)
- Custom Domain Name HTTPS security through integration with AWS Certificate Manager (ACM)
  - If using Edge-Optimized endpoint, then the certificate must be in **us-east-1**
  - If using Regional endpoint, the certificate must be in the API Gateway region
  - Must setup CNAME or A-alias record in Route 53

# API Gateway – Resource Policy



# API Gateway – Cross VPC Same-Region Access



No VPC peering needed!

## Resource Policy

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Principal": "*",  
            "Action": "execute-api:Invoke",  
            "Resource": "execute-api:/{*}"  
        },  
        {  
            "Effect": "Deny",  
            "Principal": "*",  
            "Action": "execute-api:Invoke",  
            "Resource": "execute-api:/{*}",  
            "Condition": {  
                "StringNotEquals": {  
                    "aws:SourceVpc": "vpce-1a2b3c4d"  
                }  
            }  
        }  
    ]  
}
```

# API Gateway – Throttling

- Account Limit
  - API Gateway throttles requests at 10,000 RPS across all APIs
  - Soft limit that can be increased upon request
- In case of throttling => 429 Too Many Requests (retryable error)
- Can set Stage limit & Method limits to improve performance
- Or you can define Usage Plans to throttle per customer
- Note: One API Gateway that is overloaded and not limited can cause the other APIs to be throttled



# AWS Artifact (not really a service)

- Portal that provides customers with on-demand access to AWS compliance documentation and AWS agreements
- **Artifact Reports** - Allows you to download AWS security and compliance documents from third-party auditors, like AWS ISO certifications, Payment Card Industry (PCI), and System and Organization Control (SOC) reports
- **Artifact Agreements** - Allows you to review, accept, and track the status of AWS agreements such as the Business Associate Addendum (BAA) or the Health Insurance Portability and Accountability Act (HIPAA) for an individual account or in your organization
- Can be used to support internal audit or compliance

# Route 53 – DNS Query Logging

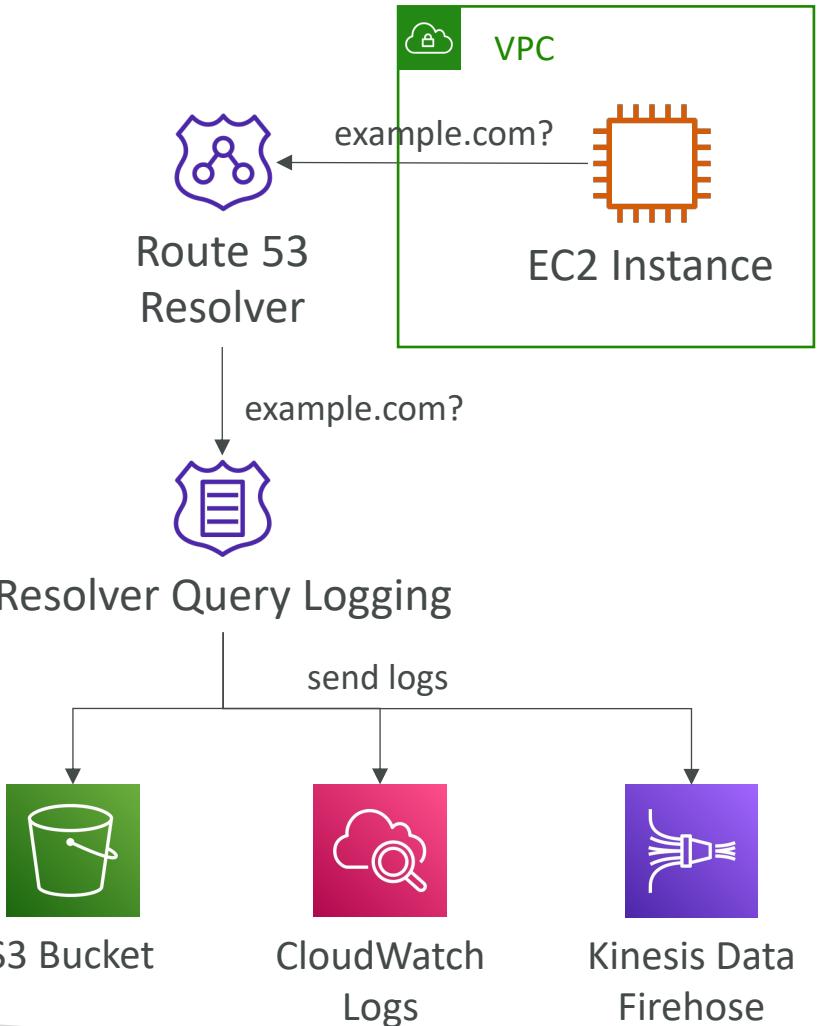
- Log information about public DNS queries Route 53 Resolver receives
- Only for Public Hosted Zones
- Logs are sent to CloudWatch Logs only

Log Format	Version	Hosted Zone ID	Query Type	Query Protocol	Resolver IP Address
1.0 2017-12-13T08:16:02.130Z	Z123412341234	example.com	A	NOERROR	UDP FRA6 192.168.1.1 -
1.0 2017-12-13T08:15:50.235Z	Z123412341234	example.com	AAAA	NOERROR	TCP IAD12 192.168.3.1 192.168.222.0/24
1.0 2017-12-13T08:16:03.983Z	Z123412341234	example.com	ANY	NOERROR	UDP FRA6 2001:db8::1234 2001:db8:abcd::/48
1.0 2017-12-13T08:15:50.342Z	Z123412341234	bad.example.com	A	NXDOMAIN	UDP IAD12 192.168.3.1 192.168.111.0/24
1.0 2017-12-13T08:16:05.744Z	Z123412341234	txt.example.com	TXT	NOERROR	UDP JFK5 192.168.1.2 -

# Route 53 – Resolver Query Logging



- Logs all DNS queries...
  - Made by resources within a VPC (EC2, Lambda, etc...)
  - From on-premises resources that are using Resolver Inbound Endpoints
  - Leveraging Resolvers Outbound Endpoints
  - Using Resolver DNS Firewall
- Can send logs to CloudWatch Logs, S3 bucket, or Kinesis Data Firehose
- Configurations can be shared with other AWS Accounts using AWS Resource Access Manager (AWS RAM)

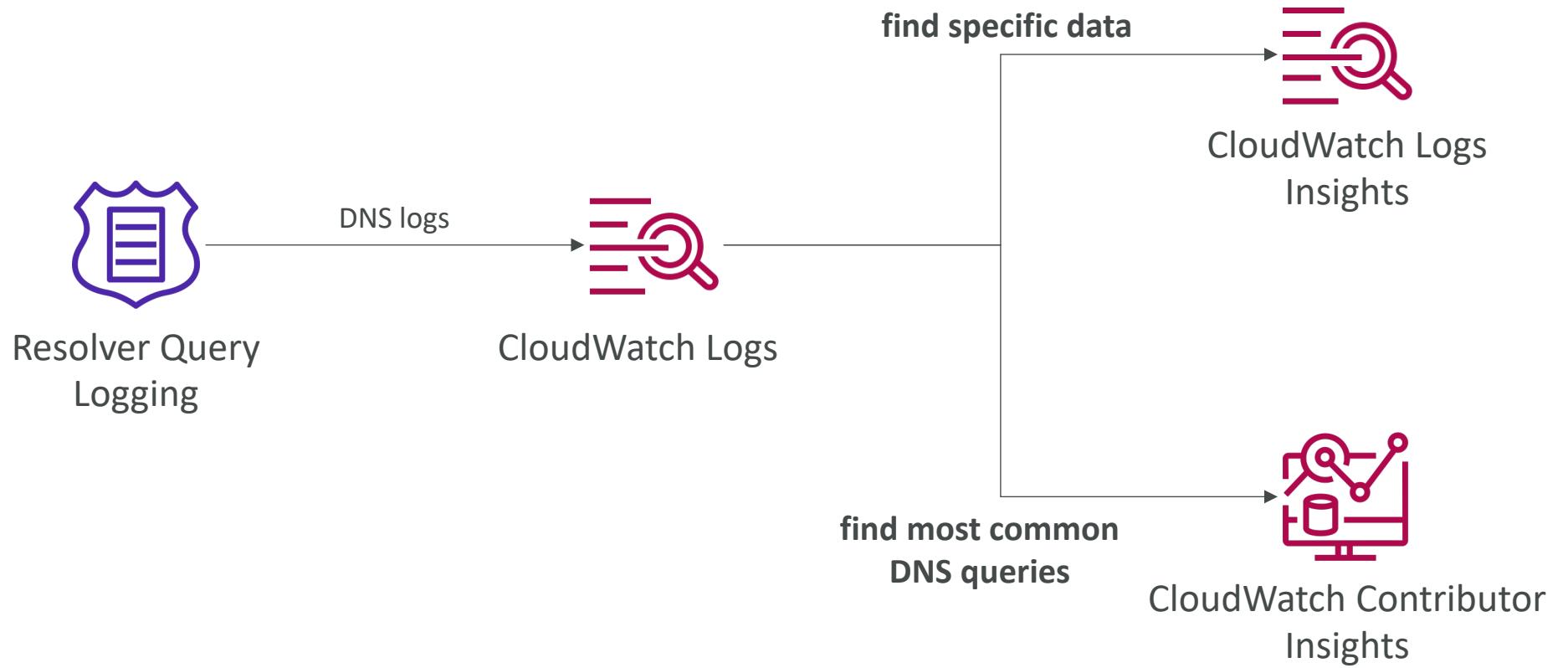




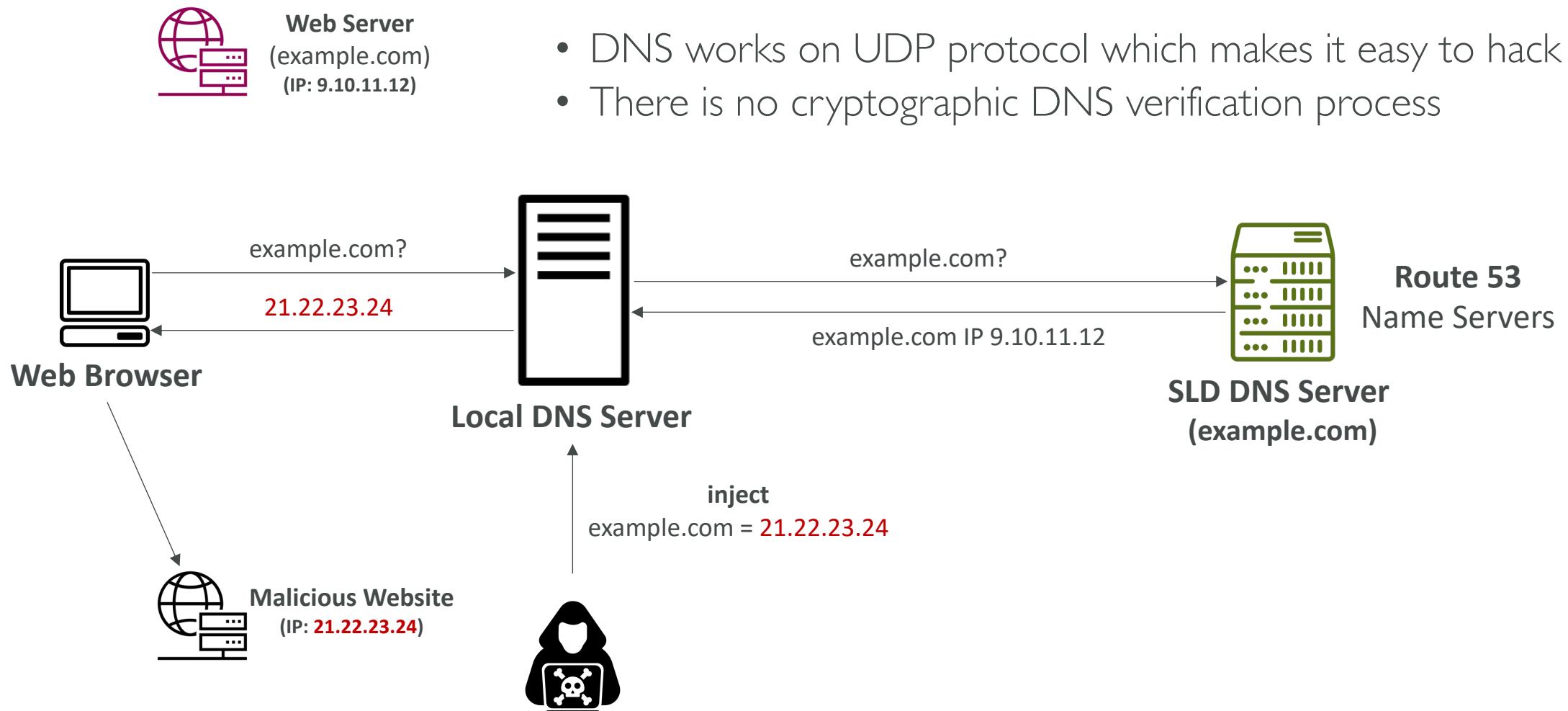
# Route 53 – Resolver Query Logging

```
{  
    "srcaddr": "4.5.64.102",  
    "vpc_id": "vpc-7example",  
    "answers": [  
        {  
            "Rdata": "203.0.113.9",  
            "Type": "PTR",  
            "Class": "IN"  
        }  
    ],  
    "firewall_rule_group_id": "rslvr-frg-01234567890abcdef",  
    "firewall_rule_action": "BLOCK",  
    "query_name": "15.3.4.32.in-addr.arpa.",  
    "firewall_domain_list_id": "rslvr-fdl-01234567890abcdef",  
    "query_class": "IN",  
    "srcids": {  
        "instance": "i-0d15cd0d3example"  
    },  
    "rcode": "NOERROR",  
    "query_type": "PTR",  
    "transport": "UDP",  
    "version": "1.100000",  
    "account_id": "111122223333",  
    "srcport": "56067",  
    "query_timestamp": "2021-02-04T17:51:55Z",  
    "region": "us-east-1"  
}
```

# Route 53 – Resolver Query Logging



# DNS Poisoning (Spoofing)



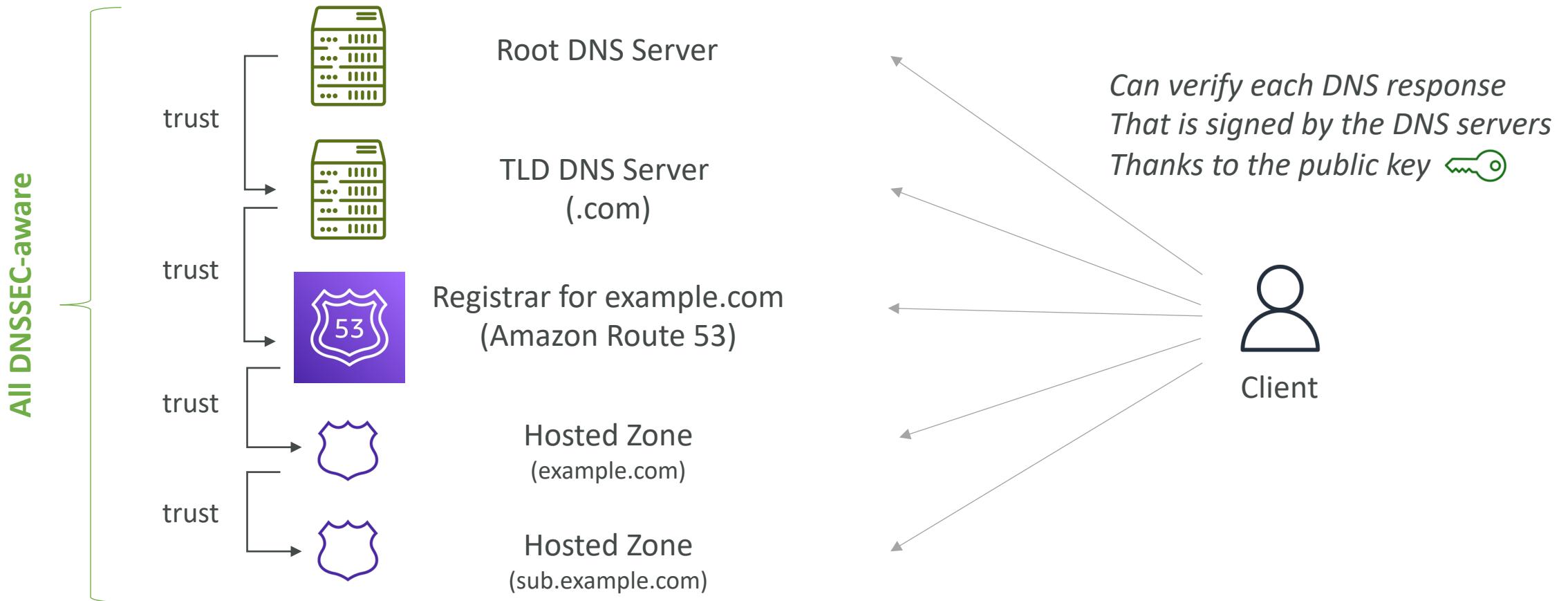
# Route 53 – DNS Security Extensions (DNSSEC)

- A protocol for securing DNS traffic, verifies DNS data integrity and origin
- Works only with **Public Hosted Zones**
- Route 53 supports both DNSSEC for Domain Registration and Signing
- **DNSSEC Signing**
  - Validate that a DNS response came from Route 53 and has not been tampered with
  - Route 53 cryptographically signs each record in the Hosted Zone
  - Two Keys:
    - Managed by you: Key-signing Key (KSK) – based on an asymmetric CMK in AWS KMS
    - Managed by AWS: Zone-signing Key (ZSK)
- When enabled, Route 53 enforces a TTL of one week for all records in the Hosted Zone (records that have TTL less than one week are not affected)

# Route 53 – Enable DNSSEC on a hosted zone

- Step 1 – Prepare for DNSSEC signing
  - Monitor zone availability (through customer feedback)
  - Lower TTL for records (recommended 1 hour)
  - Lower SOA minimum for 5 minutes
- Step 2 – Enable DNSSEC signing and create a KSK
  - Enable DNSSEC in Route 53 for your hosted zone (Console or CLI)
  - Make Route 53 create a KSK in the console and link it to a Customer managed CMK
- Step 3 – Establish chain of trust
  - Create a chain of trust between the hosted zone and the parent hosted zone
  - **By creating a Delegation Signer (DS) record in the parent zone**
  - It contains a hash of the public key used to sign DNS records
  - Your registrar can be Route 53 or a 3rd party registrar
- Step 4 – (good to have) Monitor for errors using CloudWatch Alarms
  - Create CloudWatch alarms for *DNSSECIInternalFailure* and *DNSSECKevSigningKeysNeedingAction*

# DNSSEC – Chain of Trust

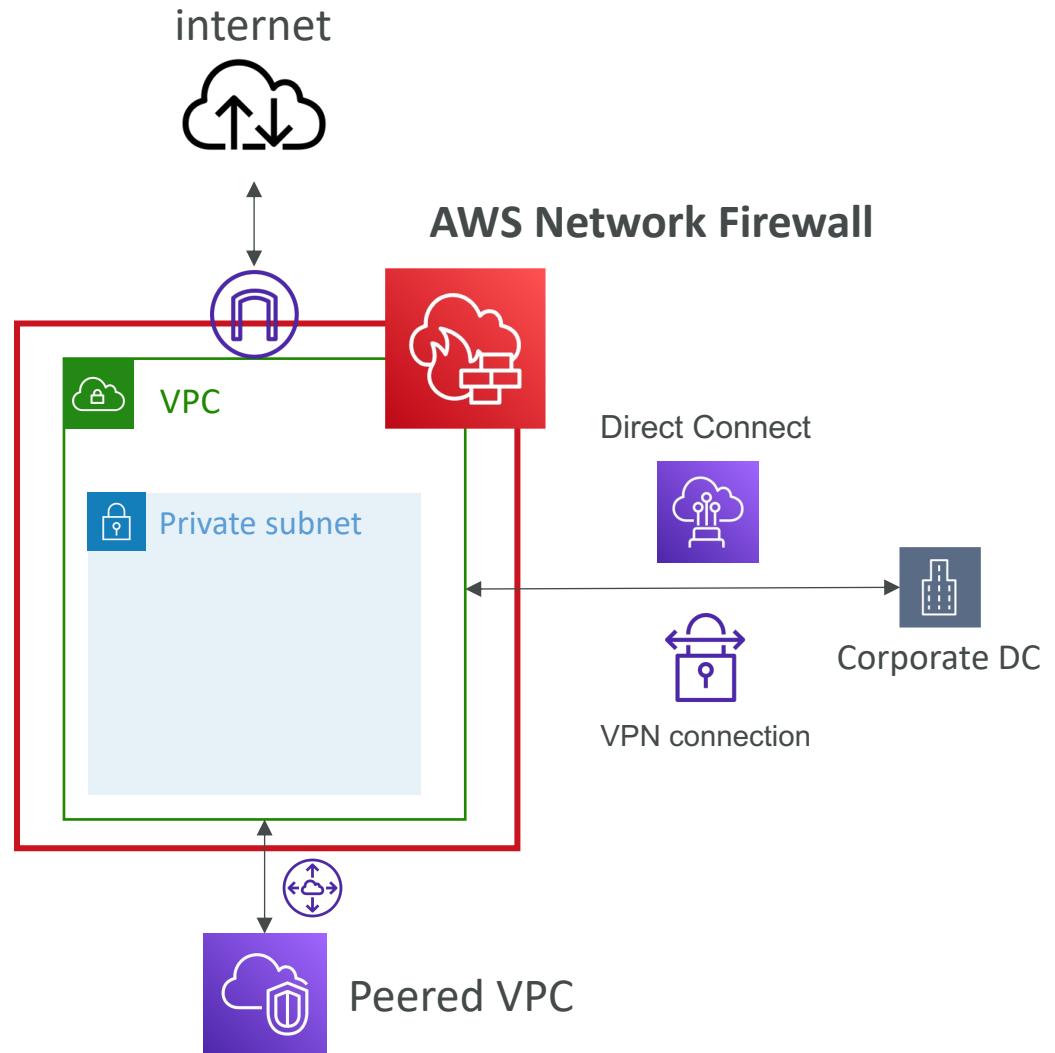


# Network Protection on AWS

- To protect network on AWS, we've seen
  - Network Access Control Lists (NACLs)
  - Amazon VPC security groups
  - AWS WAF (protect against malicious requests)
  - AWS Shield & AWS Shield Advanced
  - AWS Firewall Manager (to manage them across accounts)
- But what if we want to protect in a sophisticated way our entire VPC?

# AWS Network Firewall

- Protect your entire Amazon VPC
- From Layer 3 to Layer 7 protection
- Any direction, you can inspect
  - VPC to VPC traffic
  - Outbound to internet
  - Inbound from internet
  - To / from Direct Connect & Site-to-Site VPN
- Internally, the AWS Network Firewall uses the AWS Gateway Load Balancer
- Rules can be centrally managed cross-account by AWS Firewall Manager to apply to many VPCs

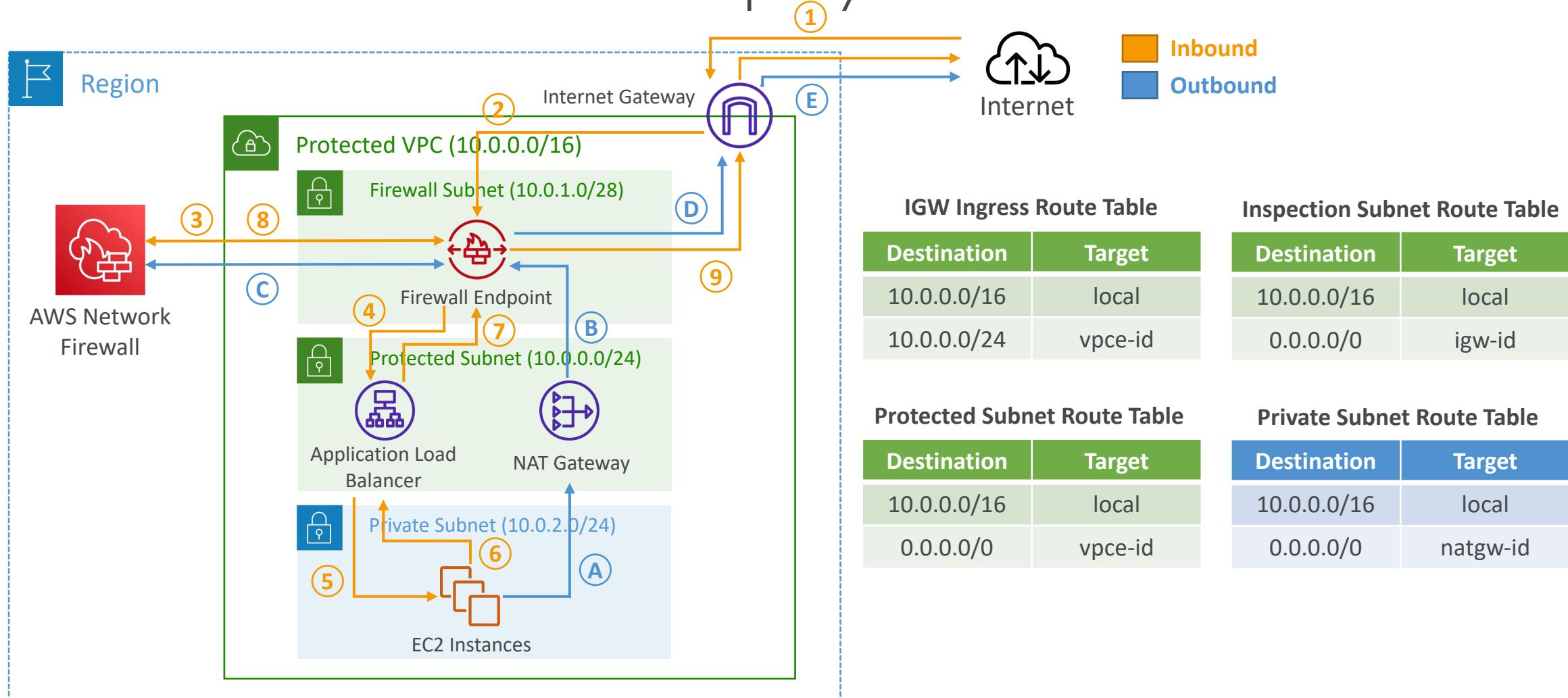




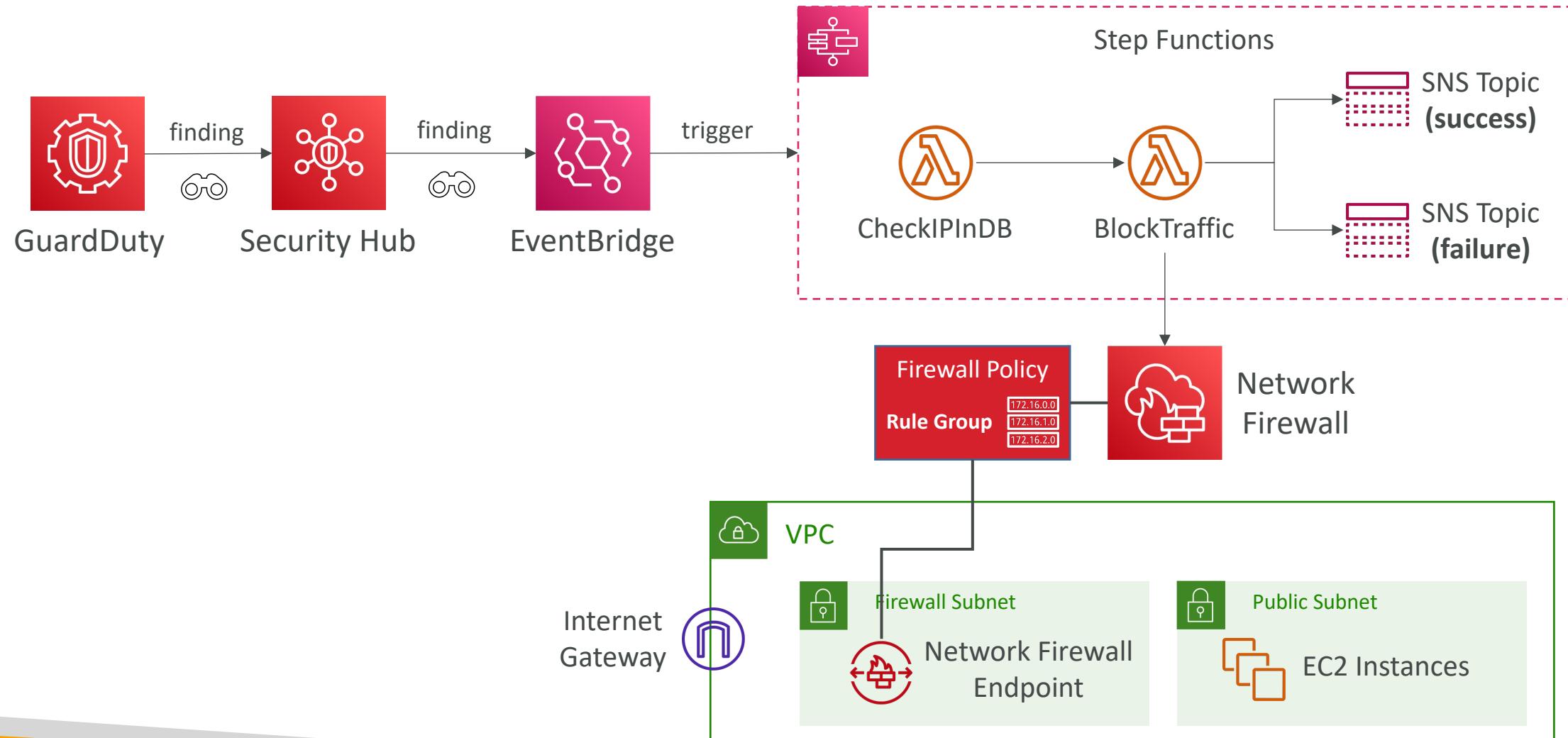
# Network Firewall – Fine Grained Controls

- Supports 1000s of rules
  - IP & port - example: 10,000s of IPs filtering
  - Protocol – example: block the SMB protocol for outbound communications
  - Stateful domain list rule groups: only allow outbound traffic to \*.mycorp.com or third-party software repo
  - General pattern matching using regex
- **Traffic filtering:** Allow, drop, or alert for the traffic that matches the rules
- Active flow inspection to protect against network threats with intrusion-prevention capabilities (like Gateway Load Balancer, but all managed by AWS)
- Send logs of rule matches to Amazon S3, CloudWatch Logs, Kinesis Data Firehose

# Network Firewall – Deployment Architectures

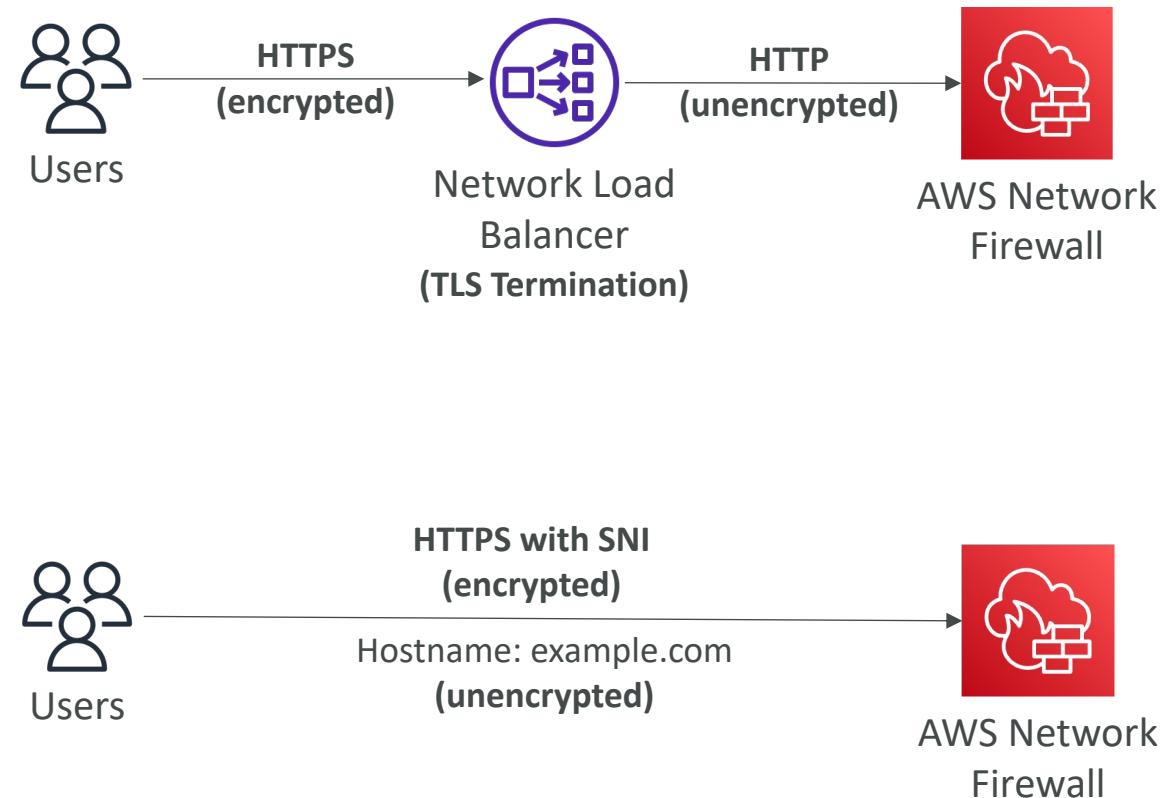


# Network Firewall – Deployment Architectures



# Network Firewall – Encrypted Traffic

- AWS Network Firewall does not currently support **deep packet inspection** for encrypted traffic
- You can first decrypt the traffic using a Network Load Balancer
- If using HTTPS with SNI, you can block based on the SNI Hostname Content (which is unencrypted)



# Amazon Simple Email Service (Amazon SES)

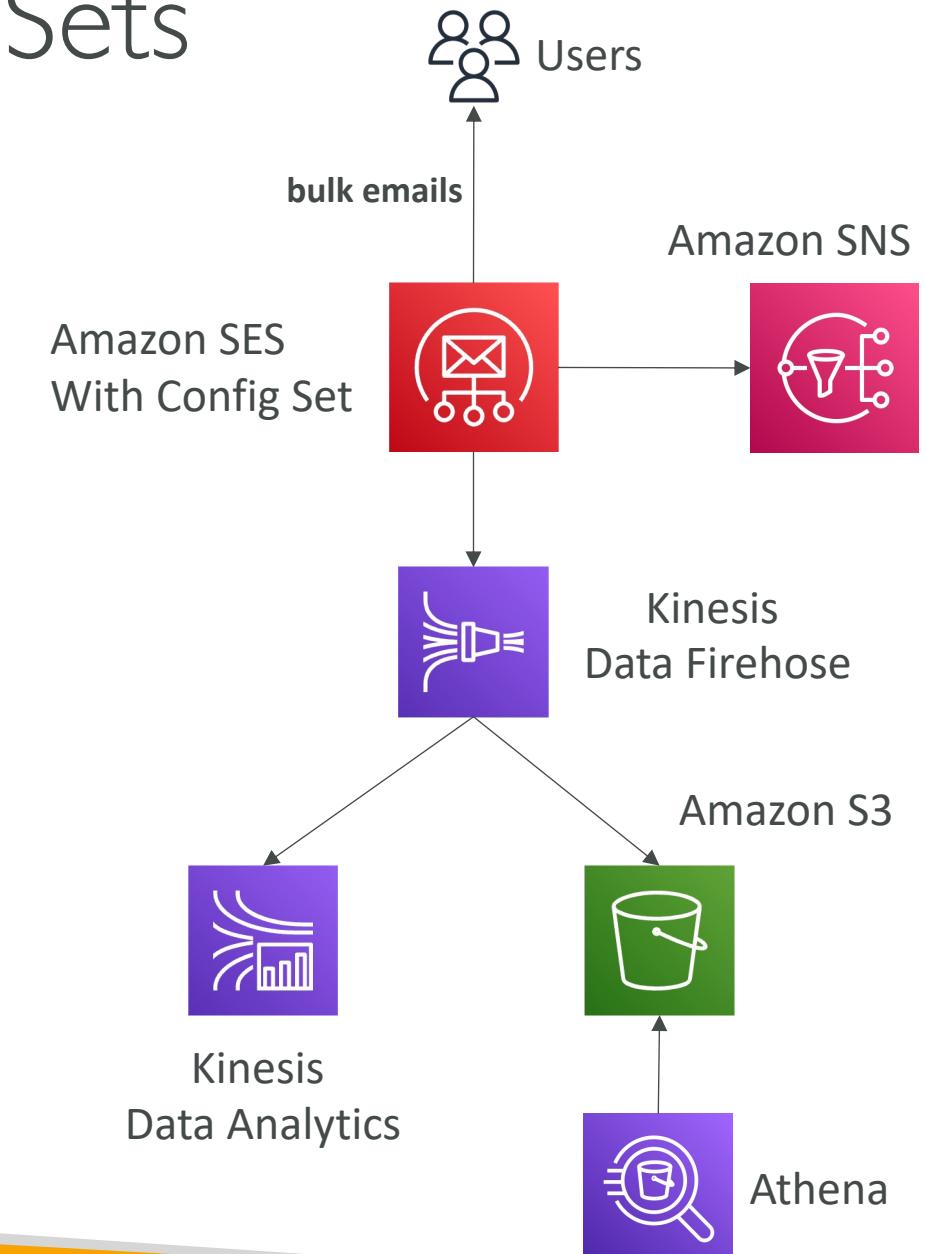


- Fully managed service to send emails securely, globally and at scale
- Allows inbound/outbound emails
- Reputation dashboard, performance insights, anti-spam feedback
- Provides statistics such as email deliveries, bounces, feedback loop results, email open
- Supports DomainKeys Identified Mail (DKIM) and Sender Policy Framework (SPF)
- Flexible IP deployment: shared, dedicated, and customer-owned IPs
- Send emails using your application using AWS Console, APIs, or SMTP
- Use cases: transactional, marketing and bulk email communications



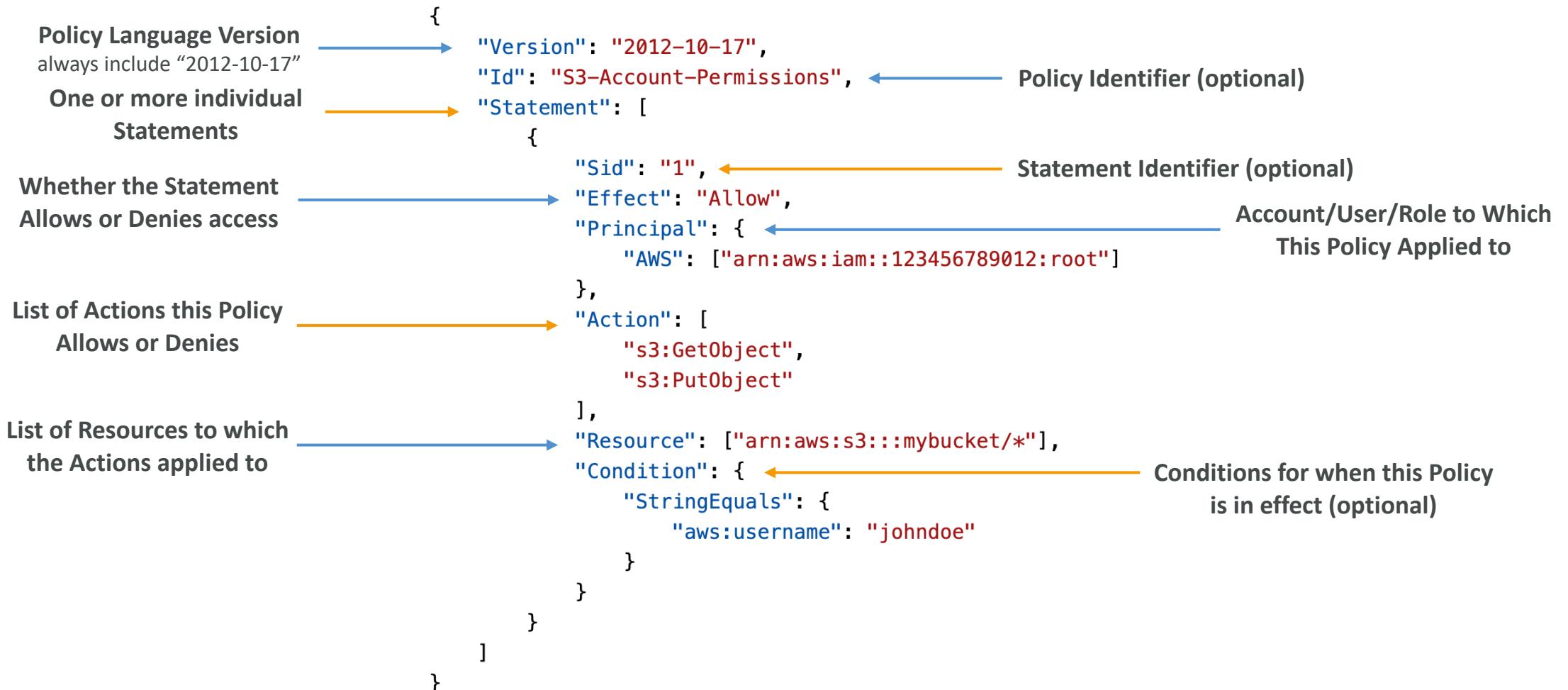
# Amazon SES – Configuration Sets

- Configuration sets help you customize and analyze your email send events
- Event destinations:
  - Kinesis Data Firehose: receives metrics (numbers of sends, deliveries, opens, clicks, bounces, and complaints) for each email
  - SNS: for immediate feedback on bounce and complaint information
- **IP pool management:** use IP pools to send particular types of emails



# Domain 4 – Identity and Access Management

# IAM Policies Structure



# IAM Policy – NotAction with Allow

- Provide access to all the actions in an AWS service, except for the actions specified in **NotAction**
- Use with the **Resource** element to provide scope for the policy, limiting the allowed actions to the actions that can be performed on the specified resource

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "NotAction": "iam:*",  
      "Resource": "*"  
    }  
  ]  
}  
  
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "NotAction": "s3>DeleteBucket",  
      "Resource": "arn:aws:s3:::*"  
    }  
  ]  
}
```

# IAM Policy – NotAction with Deny

- Use the **NotAction** element in a statement with "Effect": "Deny" to deny access to all the listed resources except for the actions specified in the **NotAction** element
- This combination does not allow the listed items, but instead explicitly denies the actions not listed
- You must still allow explicitly actions that you want to allow

```
"Version": "2012-10-17",
"Statement": [
  {
    "Effect": "Deny",
    "NotAction": "iam:*",
    "Resource": "*",
    "Condition": {
      "BoolIfExists": {
        "aws:MultiFactorAuthPresent": "false"
      }
    }
  }
]
```

# IAM Policy – Restrict to One Region (NotAction)

Deny everything outside of eu-central-1

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Deny",  
            "NotAction": [  
                "cloudfront:*",  
                "iam:*",  
                "route53:*",  
                "support:*"  
            ],  
            "Resource": "*",  
            "Condition": {  
                "StringNotEquals": {  
                    "aws:RequestedRegion": ["eu-central-1"]  
                }  
            }  
        }  
    ]  
}
```

Deny everything outside of eu-central-1  
except Amazon S3

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Deny",  
            "NotAction": ["s3:*"],  
            "Resource": "*",  
            "Condition": {  
                "StringNotEquals": {  
                    "aws:RequestedRegion": ["eu-central-1"]  
                }  
            }  
        }  
    ]  
}
```

# (Action | Not Action) & (Allow | Deny)

		<b>Allow</b>	<b>Deny</b>
<b>Action</b>	<pre>{   "Version": "2012-10-17",   "Statement": [     {       "Effect": "Allow",       "Action": ["iam:*"],       "Resource": "*"     }   ] }</pre> <p>Allow IAM</p>	<pre>{   "Version": "2012-10-17",   "Statement": [     {       "Effect": "Deny",       "Action": ["iam:*"],       "Resource": "*"     }   ] }</pre> <p>Deny IAM</p>	
<b>NotAction</b>	<pre>{   "Version": "2012-10-17",   "Statement": [     {       "Effect": "Allow",       "NotAction": ["iam:*"],       "Resource": "*"     }   ] }</pre> <p>Allow anything not IAM</p>	<pre>{   "Version": "2012-10-17",   "Statement": [     {       "Effect": "Deny",       "NotAction": ["iam:*"],       "Resource": "*"     }   ] }</pre> <p>Deny anything not IAM</p>	

# Principal Options in IAM Policies

- AWS Account and Root User

```
"Principal": { "AWS": "123456789012" }  
"Principal": { "AWS": "arn:aws:iam::123456789012:root" }
```

- IAM Roles

```
"Principal": { "AWS": "arn:aws:iam::123456789012:role/role-name" }
```

- IAM Role Sessions

```
"Principal": { "AWS": "arn:aws:sts::123456789012:assumed-role/role-name/role-session-name" }  
"Principal": { "Federated": "cognito-identity.amazonaws.com" }  
"Principal": { "Federated": "arn:aws:iam::123456789012:saml-provider/provider-name" }
```

# Principal Options in IAM Policies

- IAM Users

```
"Principal": { "AWS": "arn:aws:iam::123456789012:user/user-name" }
```

- Federated User Sessions

```
"Principal": { "AWS": "arn:aws:sts::123456789012:federated-user/user-name" }
```

- AWS Services

```
"Principal": {  
    "Service": [  
        "ecs.amazonaws.com",  
        "elasticloadbalancing.amazonaws.com"  
    ]  
}
```

- All Principals

```
"Principal": "*"  
"Principal": { "AWS": "*" }
```

# IAM Condition – Condition Operators

- `StringEquals` / `StringNotEquals` – Case Sensitive, Exact Matching

```
"StringEquals": {  
    "aws:PrincipalTag/job-category": "iamuser-admin"  
}
```

- `StringLike` / `StringNotLike` – Case Sensitive, optional Partial Matching using \*, ?

```
"StringLike": {  
    "s3:prefix": ["", "home/*/data/", "home/${aws:username}/"]  
}
```

- `DateEquals`, `DateLessThan...`

```
"DateGreaterThan": {  
    "aws:TokenIssueTime": "2020-01-01T00:00:01Z"  
}
```

- `ArnLike` / `ArnNotLike` – used specifically for ARN

# IAM Condition – Condition Operators

- Bool – check for Boolean value

```
"Bool": {  
    "aws:SecureTransport": "false"  
}
```

- IpAddress / NotIpAddress (CIDR format)

- Resolves to the IP address that the request originates from
- Public IP only – IpAddress does not apply to requests through VPC Endpoints

```
"IpAddress": {  
    "aws:SourceIp": "203.0.113.0/24"  
}
```

# IAM Conditions – RequestedRegion

- The AWS region of the request
- Used to restrict specific actions in specific AWS regions

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Action": [  
        "ec2:Stop*",  
        "ec2:Terminate*"  
      ],  
      "Resource": "*",  
      "Condition": {  
        "StringEquals": {  
          "aws:RequestedRegion": [  
            "eu-west-1",  
            "eu-west-2",  
            "eu-west-3"  
          ]  
        }  
      }  
    }  
  ]  
}
```

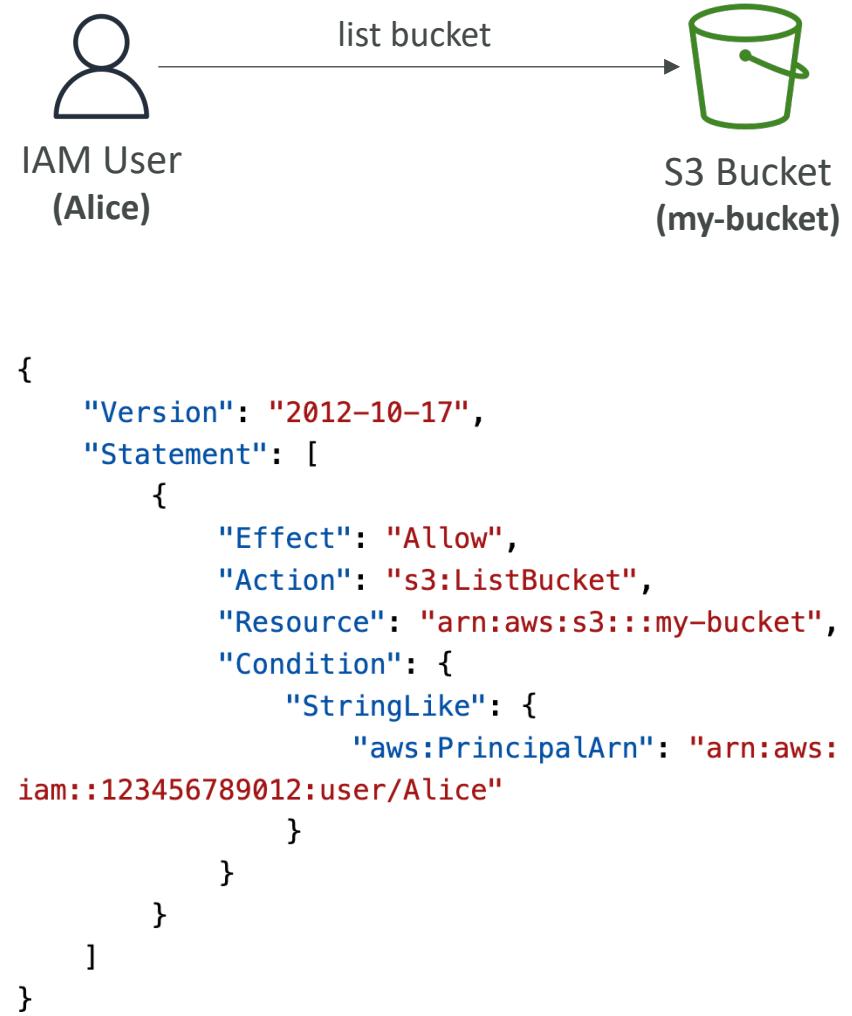
# IAM Conditions – RequestedRegion

- When using a global AWS service (e.g., IAM, CloudFront, Route53, Support), the AWS region is always us-east-1
- Work around using NotAction and Deny

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Deny",  
      "NotAction": [  
        "cloudfront:*",  
        "iam:*",  
        "route53:*",  
        "support:*"  
      ],  
      "Resource": "*",  
      "Condition": {  
        "StringNotEquals": {  
          "aws:RequestedRegion": [  
            "eu-central-1",  
            "eu-west-1",  
            "eu-west-2",  
            "eu-west-3"  
          ]  
        }  
      }  
    }  
  ]  
}
```

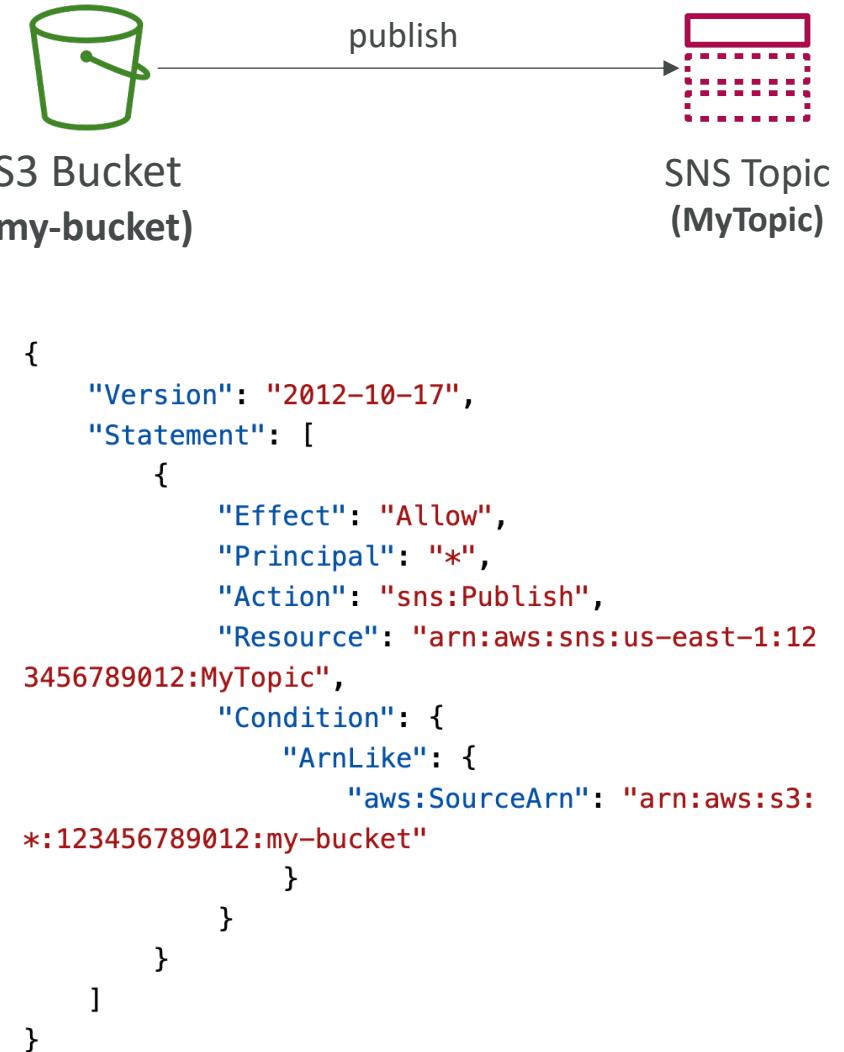
# IAM Conditions – PrincipalArn

- Compare the ARN of the principal that made the request with the ARN specified in the policy
- Note: For IAM roles, the request context returns the ARN of the role, not the ARN of the user that assumed the role
- The following types of Principals are allowed:
  - IAM role arn:aws:iam::123456789012:role/role-name
  - IAM user arn:aws:iam::123456789012:user/user-name
  - AWS Root User arn:aws:iam::123456789012:root
  - AWS STS federated user session



# IAM Conditions – SourceArn

- Compare the ARN of the resource making a service-to-service request with the ARN that you specify in the policy
- This key is included in the request context only if accessing a resource triggers an AWS service to call another service on behalf of the resource owner
- Example: an S3 bucket update triggers an SNS topic `sns:Publish` API, the policy set the value of the condition key to the ARN of the S3 bucket



# IAM Conditions – CalledVia

- Look at the AWS service that made the request on behalf of the IAM User or Role
- Contains an ordered list of each AWS service in the chain that made requests on the principal's behalf
- Supports:
  - athena.amazonaws.com
  - cloudformation.amazonaws.com
  - dynamodb.amazonaws.com
  - kms.amazonaws.com



```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Deny",  
      "Principal": "*",  
      "Action": "s3:Get*",  
      "Resource": "arn:aws:s3:::bucket/*",  
      "Condition": {  
        "StringNotEquals": {  
          "aws:CalledVia": "athena.amazonaws.com"  
        }  
      }  
    }  
  ]  
}
```

# IAM Conditions – IP & VPC Conditions

- **aws:SourceIp**
  - Public requester IP (e.g., public EC2 IP if coming from EC2)
  - Not present in requests through VPC Endpoints
- **aws:VpcSourceIp** – requester IP through VPC Endpoints (private IP)
- **aws:SourceVpce** – restrict access to a specific VPC Endpoint
- **aws:SourceVpc**
  - Restrict to a specific VPC ID
  - Request must be made through a VPC Endpoint
- Common to use these conditions with S3 Bucket Policies

# IAM Conditions – IP & VPC Conditions

## Restrict Access from Public IP Addresses

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

## Restrict Access from Private IP Addresses (through a VPC Endpoint)

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Deny",  
      "Action": "*",  
      "Resource": "*",  
      "Condition": {  
        "NotIpAddress": {  
          "aws:VpcSourceIp": ["192.0.2.0/24", "198.51.100.0/24"]  
        }  
      }  
    }  
  ]  
}
```

# IAM Conditions – ResourceTag & PrincipalTag

- Controls access to AWS Resources using tags
- **aws:ResourceTag** – tags that exist on AWS Resources
  - Sometimes you will see ec2:ResourceTag (service-specific)
- **aws:PrincipalTag** – tags that exist on the IAM user or IAM role making the request

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": ["ec2:startInstances", "ec2:StopInstances"],  
            "Resource": "arn:aws:ec2:us-east-1:123456789012:instance/*",  
            "Condition": {  
                "StringEquals": {  
                    "ec2:ResourceTag/Project": "DataAnalytics",  
                    "aws:PrincipalTag/Department": "Data"  
                }  
            }  
        }  
    ]  
}
```

# IAM Permission Boundaries

- IAM Permission Boundaries are supported for users and roles (not groups)
- Advanced feature to use a managed policy to set the maximum permissions an IAM entity can get.

**Example:**

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "s3:*",  
                "cloudwatch:*",  
                "ec2:*"  
            ],  
            "Resource": "*"  
        }  
    ]  
}
```



**IAM Permission Boundary**

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

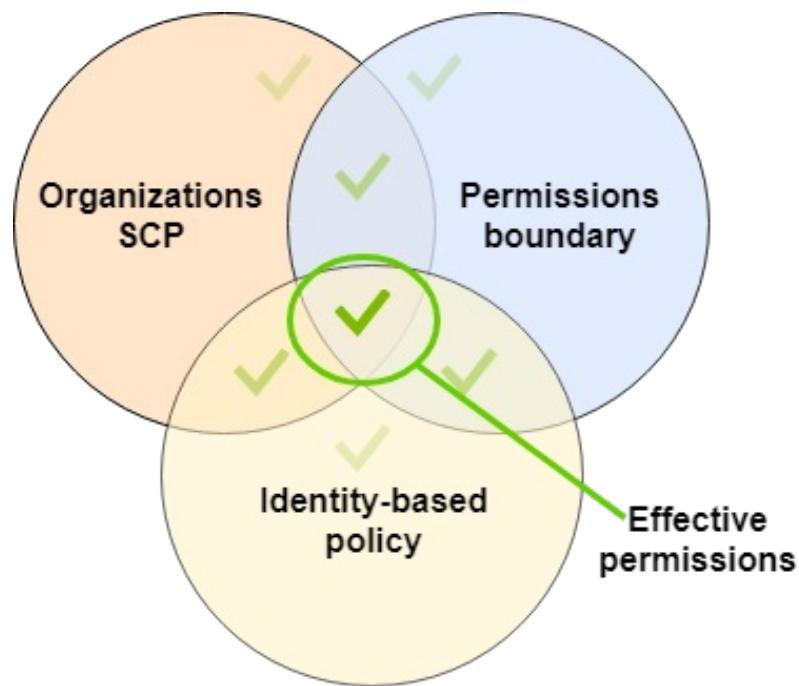
**IAM Permissions  
Through IAM Policy**



**No Permissions**

# IAM Permission Boundaries

- Can be used in combinations of AWS Organizations SCP



[https://docs.aws.amazon.com/IAM/latest/UserGuide/access\\_policies\\_boundaries.html](https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies_boundaries.html)

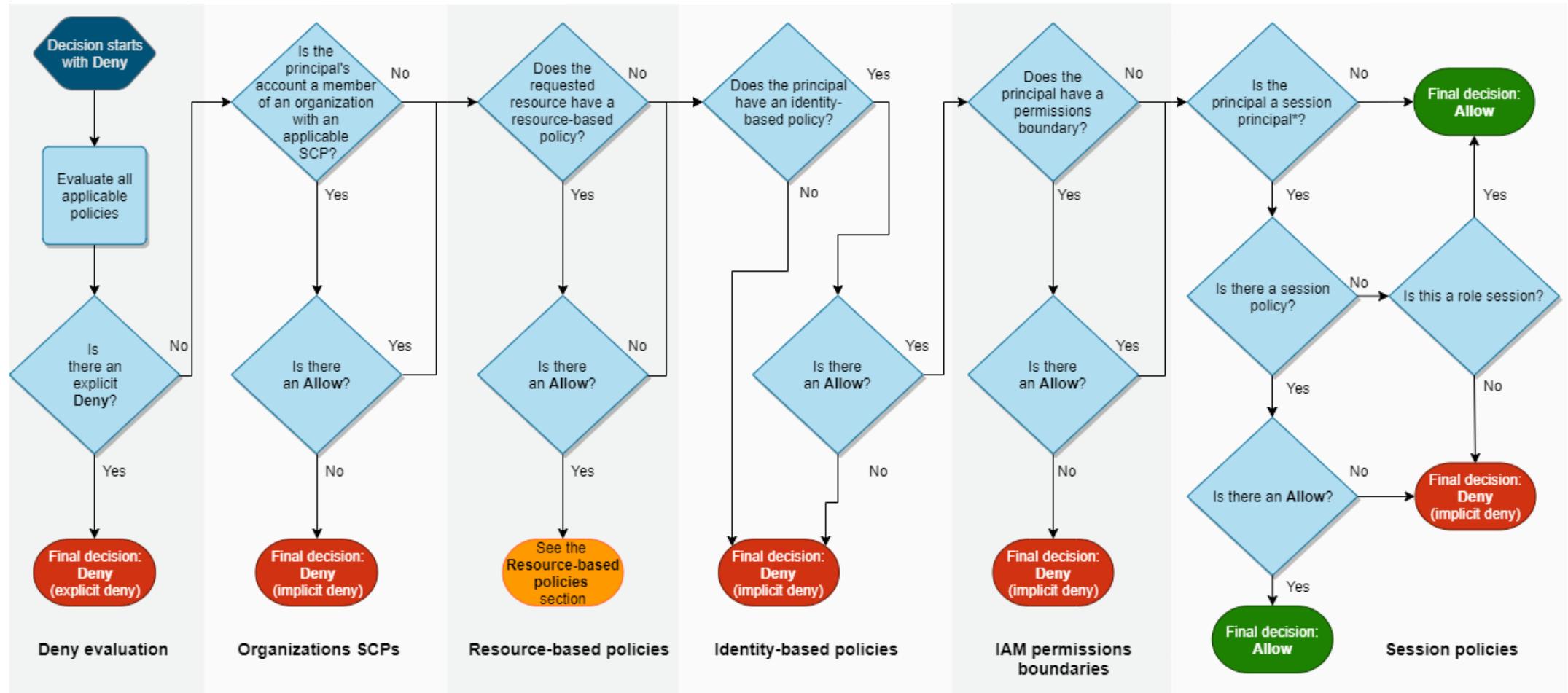
## Use cases

- Delegate responsibilities to non administrators within their permission boundaries, for example create new IAM users
- Allow developers to self-assign policies and manage their own permissions, while making sure they can't "escalate" their privileges (= make themselves admin)
- Useful to restrict one specific user (instead of a whole account using Organizations & SCP)

# IAM Policy – Simplified Evaluation Logic (Allow/Deny)

1. By default, all requests are implicitly denied except for the AWS account root user, which has full access.
2. An explicit allow in an identity-based or resource-based policy overrides the default in 1)
3. If a permissions boundary, Organizations SCP, or session policy is present, an explicit allow is used to limit actions. Anything not explicitly allowed is an implicit deny and may override the decision in 2)
4. An explicit deny in any policy overrides any allows

# IAM Policy Evaluation Logic



\*A session principal is either a role session or an IAM federated user session.

[https://docs.aws.amazon.com/IAM/latest/UserGuide/reference\\_policies\\_evaluation-logic.html](https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_evaluation-logic.html)

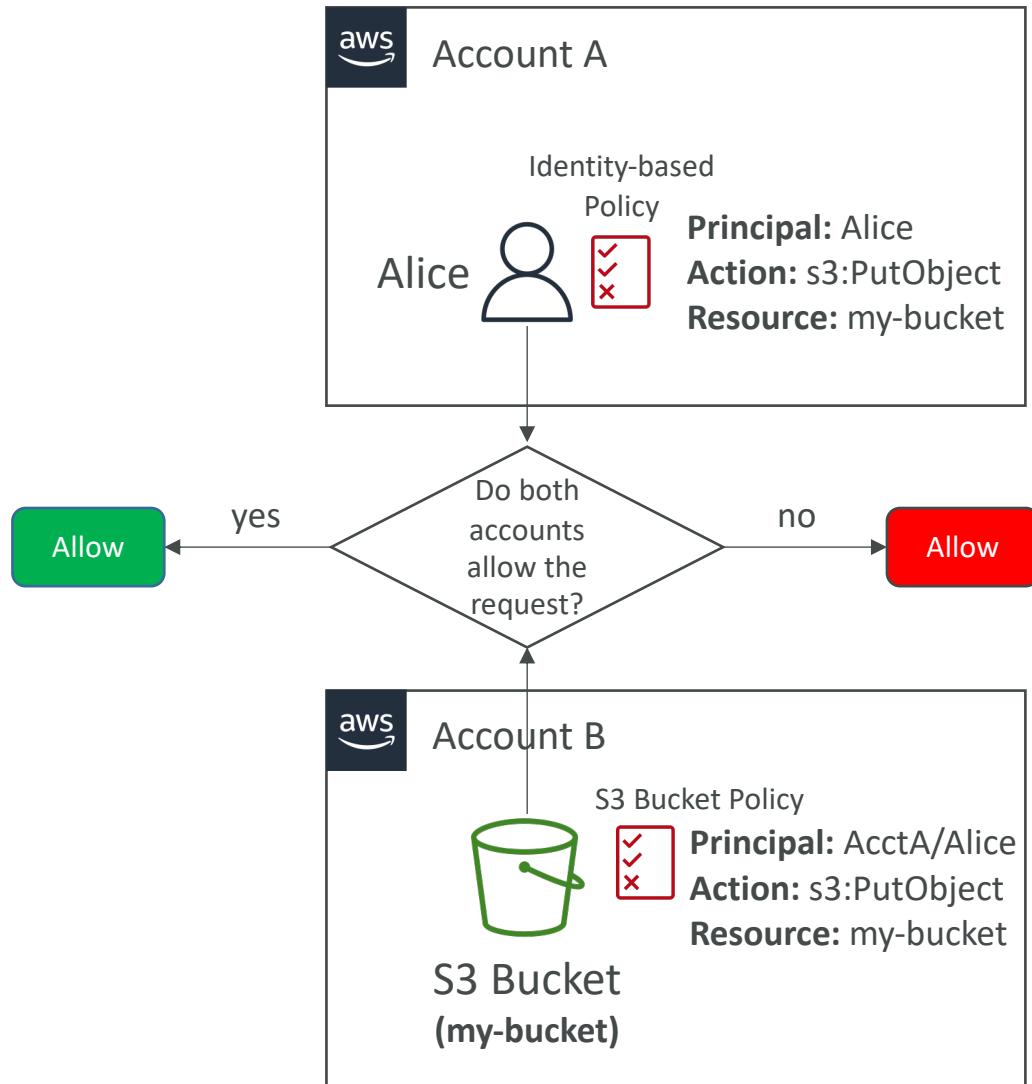
# Example IAM Policy

- Can you perform sqs>CreateQueue?
- Can you perform sqs>DeleteQueue?
- Can you perform ec2:DescribeInstances?

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

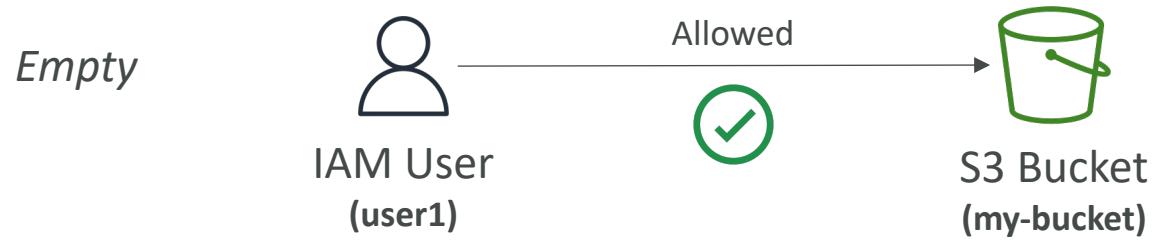
# Cross-Account Access Policy Evaluation Logic

- Request from Account A to Account B:
- The requester in Account A must have an identity-based policy
- That policy must allow the requester to make a request to the resource in Account B
- The resource-based policy in Account B must allow the requester in Account A to access the resource



# IAM Policy + Resource Policy

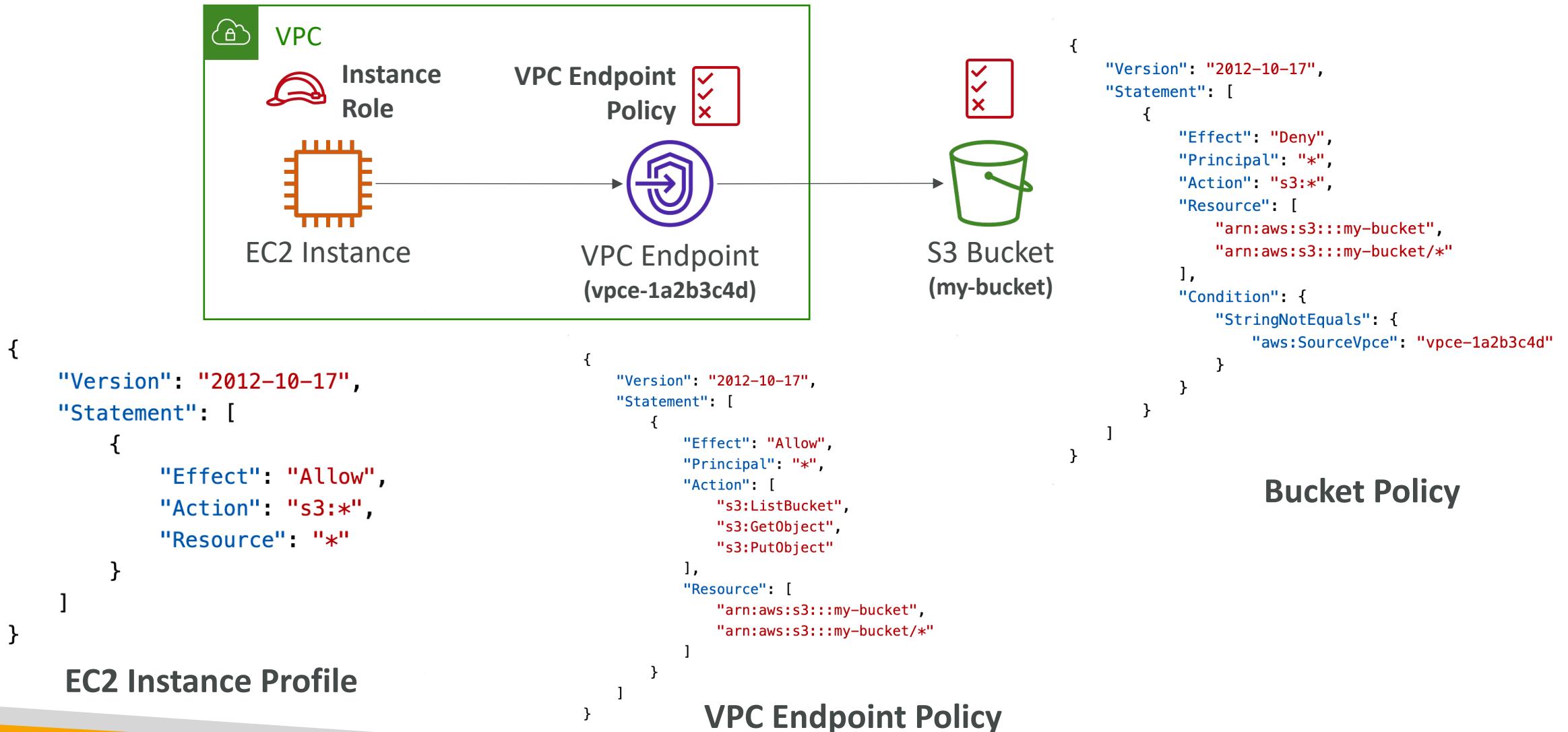
## IAM Policy



## Resource Policy (S3 Bucket Policy)

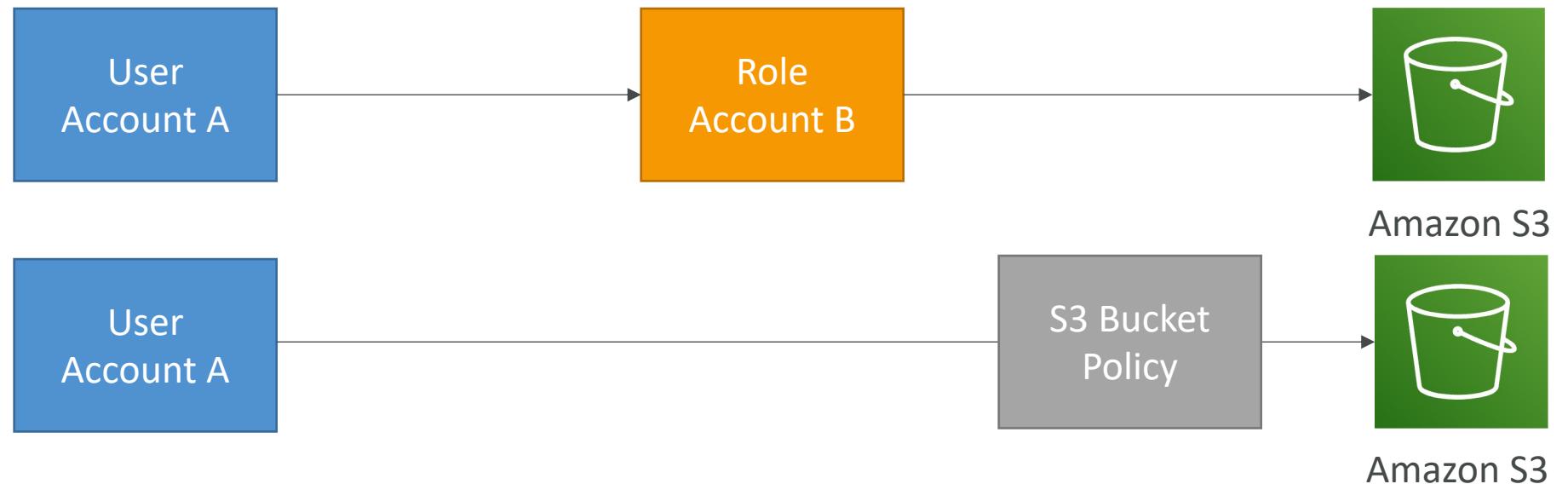
```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Principal": {  
        "AWS": "arn:aws:iam::123456789012:user/user1"  
      },  
      "Action": [  
        "s3>ListBucket",  
        "s3GetObject"  
      ],  
      "Resource": [  
        "arn:aws:s3:::my-bucket",  
        "arn:aws:s3:::my-bucket/*"  
      ]  
    }  
  ]  
}
```

# IAM Policy + VPC Endpoint Policy + Resource Policy



# IAM Roles vs Resource Based Policies

- Cross account:
  - attaching a resource-based policy to a resource (example: S3 bucket policy)
  - OR using a role as a proxy



# IAM Roles vs Resource-Based Policies

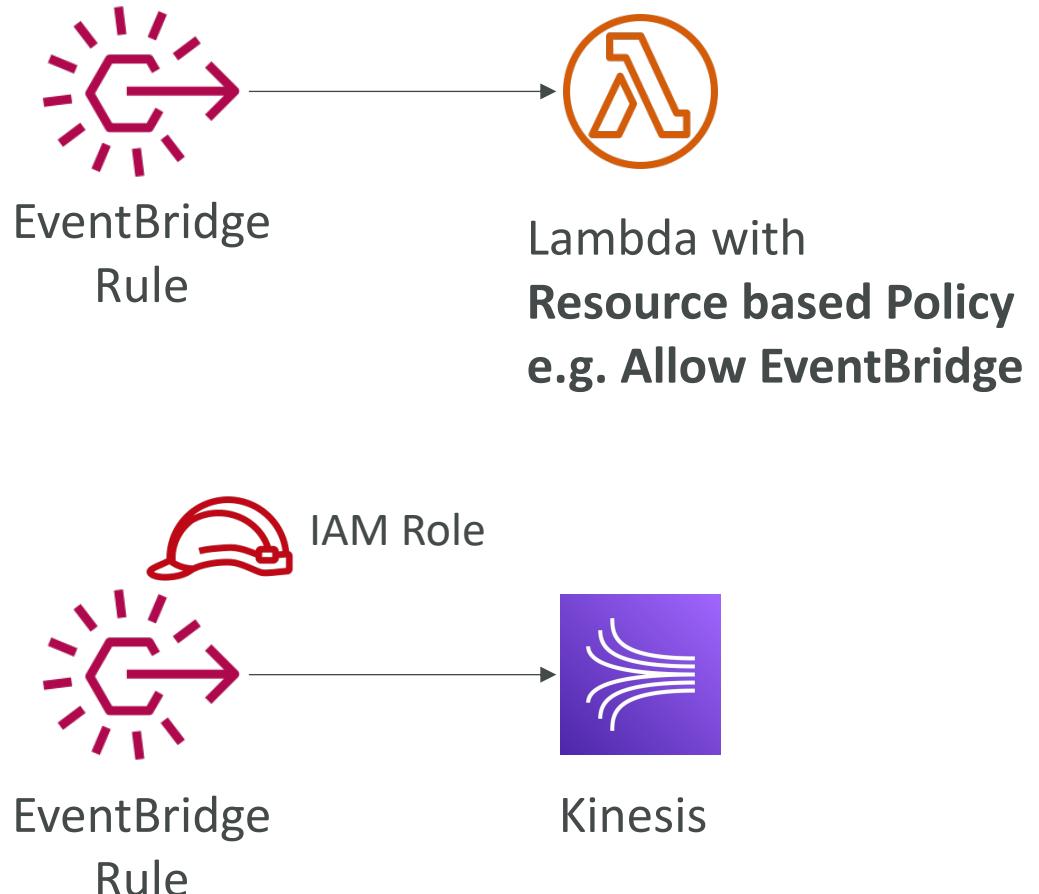
- When you assume a role (user, application or service), you give up your original permissions and take the permissions assigned to the role
- When using a resource-based policy, the principal doesn't have to give up his permissions
- Example: User in account A needs to scan a DynamoDB table in Account A and dump it in an S3 bucket in Account B.
- Supported by: Amazon S3 buckets, SNS topics, SQS queues, etc...

# IAM Roles vs Resource-Based Policies

- IAM Roles:
  - Helpful to give temporary permissions for a specific task
  - Allow a user/application to perform many actions in a different account
  - Permissions expire over time
- Resource-based policies:
  - Used to control access to specific resources (resource-centric view)
  - Allow cross-account access
  - Permanent authorization (as long as it exists in the resource-based policy)

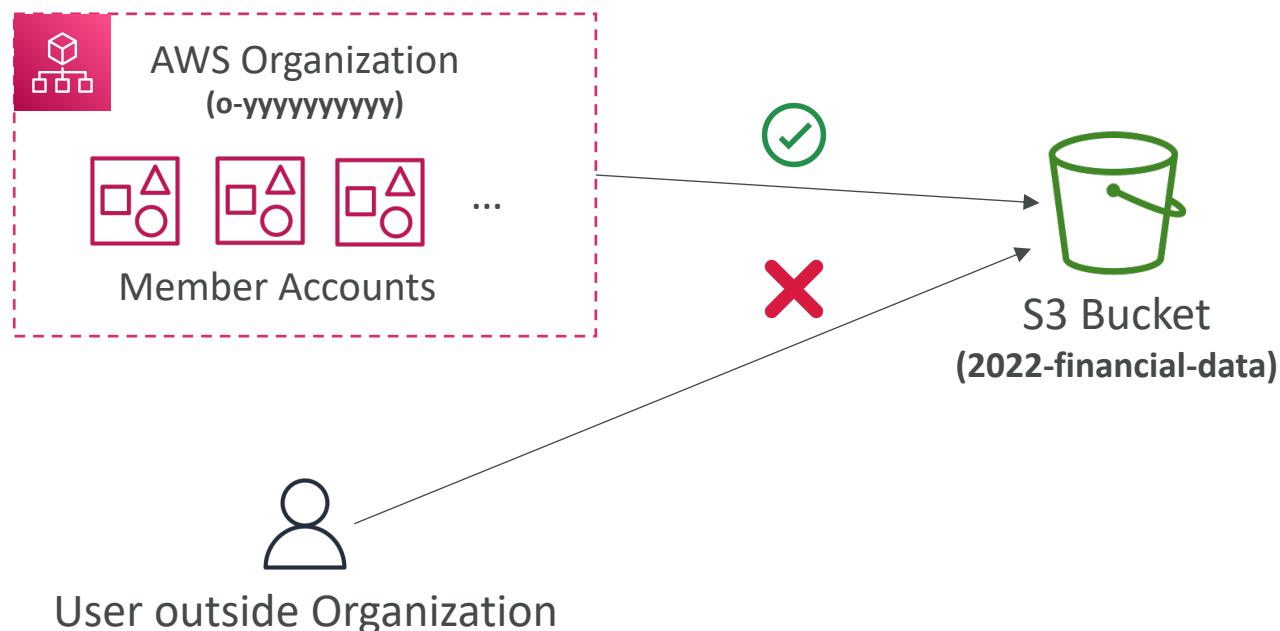
# Amazon EventBridge – Security

- When a rule runs, it needs permissions on the target
- Resource-based policy: Lambda, SNS, SQS, CloudWatch Logs, API Gateway...
- IAM role: Kinesis stream, Systems Manager Run Command, ECS task...



# Resource Policies & aws:PrincipalOrgID

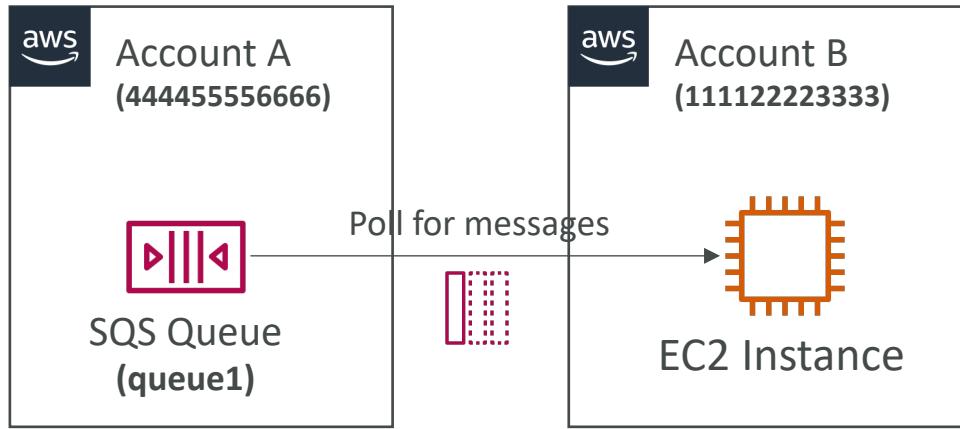
- aws:PrincipalOrgID can be used in any resource policies to restrict access to accounts that are member of an AWS Organization



```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Action": ["s3:PutObject", "s3:GetObject"],  
      "Resource": "arn:aws:s3:::2022-financial-data/*",  
      "Condition": {  
        "StringEquals": {  
          "aws:PrincipalOrgID": ["o-yyyyyyyyyy"]  
        }  
      }  
    }  
  ]  
}
```

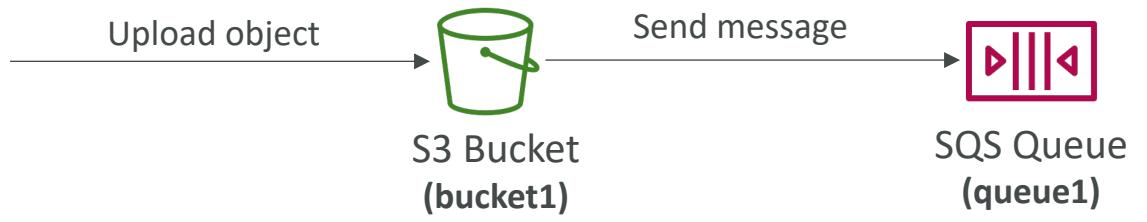
# SQS Queue Access Policy

## Cross Account Access



```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": { "AWS": ["111122223333"] },
      "Action": "sns:ReceiveMessage",
      "Resource": "arn:aws:sns:us-east-1:111122223333:queue1"
    }
  ]
}
```

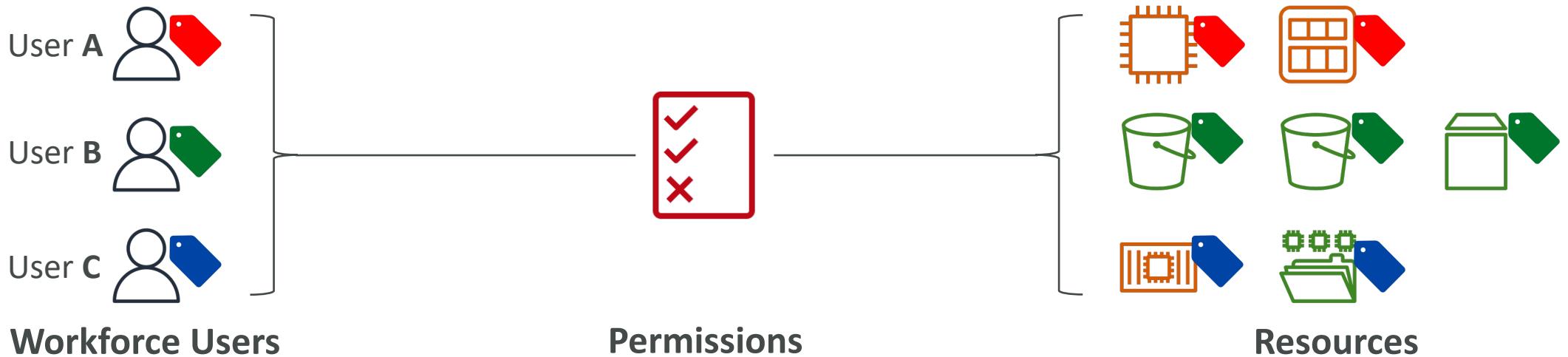
## Publish S3 Event Notifications To SQS Queue



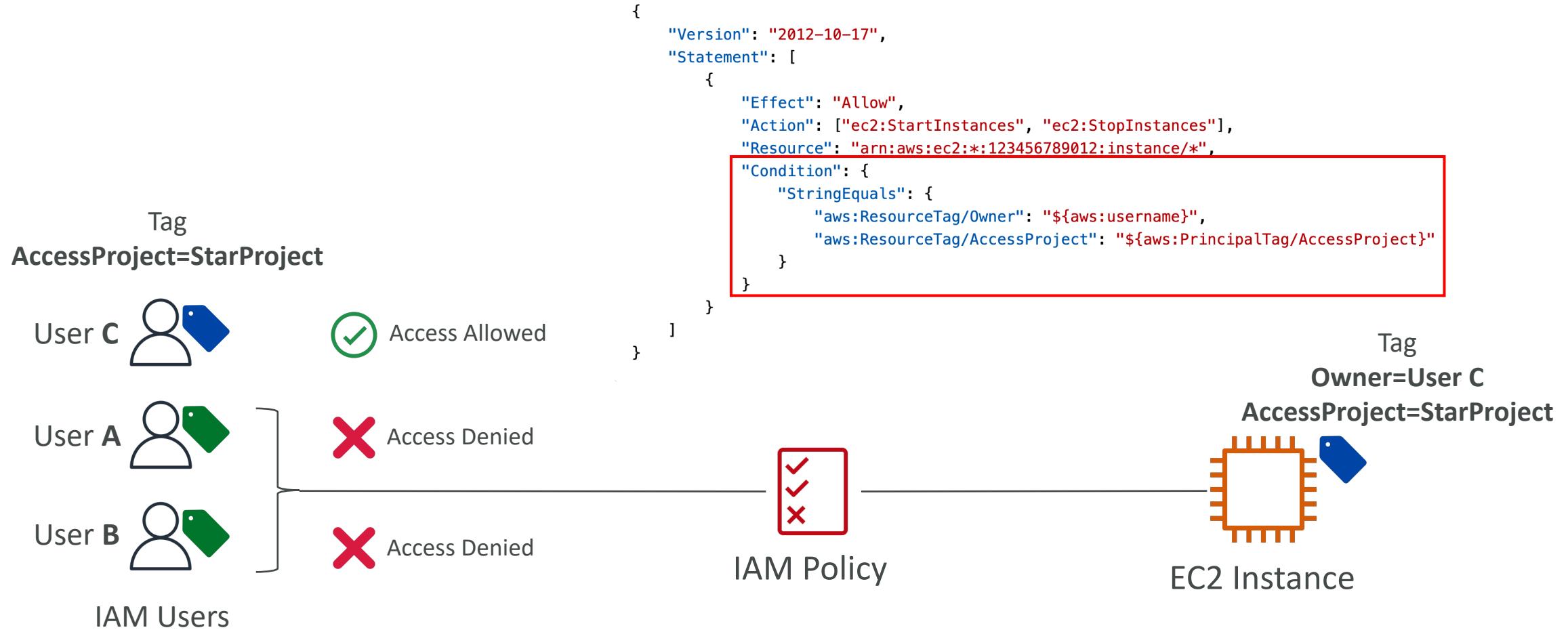
```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": { "AWS": "*" },
      "Action": "sns:Publish",
      "Resource": "arn:aws:sns:us-east-1:444455556666:queue1",
      "Condition": {
        "ArnLike": {
          "aws:SourceArn": "arn:aws:s3:::*::bucket1"
        },
        "StringEquals": {
          "aws:SourceAccount": "<bucket1_owner_account_id>"
        }
      }
    }
  ]
}
```

# ABAC – Attribute-Based Access Control

- Defines fine-grained permissions based on user attributes
- Example: department, job role, team name, ...
- Instead of creating IAM roles for every team, use ABAC to group attributes to identify which resources a set of users can access
- Allow operations when the principal's tags matches the resource tag
- Helpful in rapidly-growing environments



# ABAC – Example



# ABAC vs. RBAC

- **Role-Based Access Control (RBAC)**
  - Defines fine-grained permissions based on user role or job function
  - Example: Administrator, DB Admins, DevOps, ...
  - Create different policies for different job functions
  - Disadvantage: must update policies when resources are added
- **Attribute-Based Access Control (ABAC) – Advantages**
  - Scale permissions easily (no need to update policies when new resources added)
  - Permissions automatically granted based on attributes
  - Require fewer policies (you don't create different policies for different job functions)
  - Ability to use users' attributes from corporate directory  
(e.g., SAML 2.0-based IdP or Web IdP)

# Multi Factor Authentication - MFA



- Users have access to your account and can possibly change configurations or delete resources in your AWS account
- You want to protect your Root Accounts and IAM users
- MFA = password you know + security device you own



- Main benefit of MFA:  
if a password is stolen or hacked, the account is not compromised

# MFA devices options in AWS

## Virtual MFA device



Google Authenticator  
(phone only)

Support for multiple tokens on a single device.



Authy  
(multi-device)

## Universal 2nd Factor (U2F) Security Key



YubiKey by Yubico (3<sup>rd</sup> party)

Support for multiple root and IAM users using a single security key

# MFA devices options in AWS

## Hardware Key Fob MFA Device



Provided by Gemalto (3<sup>rd</sup> party)

## Hardware Key Fob MFA Device for AWS GovCloud (US)



Provided by SurePassID (3<sup>rd</sup> party)

# Amazon S3 – MFA Delete

- **MFA (Multi-Factor Authentication)** – force users to generate a code on a device (usually a mobile phone or hardware) before doing important operations on S3
- MFA will be required to:
  - Permanently delete an object version
  - Suspend Versioning on the bucket
- MFA won't be required to:
  - Enable Versioning
  - List deleted versions
- To use MFA Delete, Versioning must be enabled on the bucket
- Only the bucket owner (root account) can enable/disable MFA Delete



Google Authenticator



MFA Hardware Device

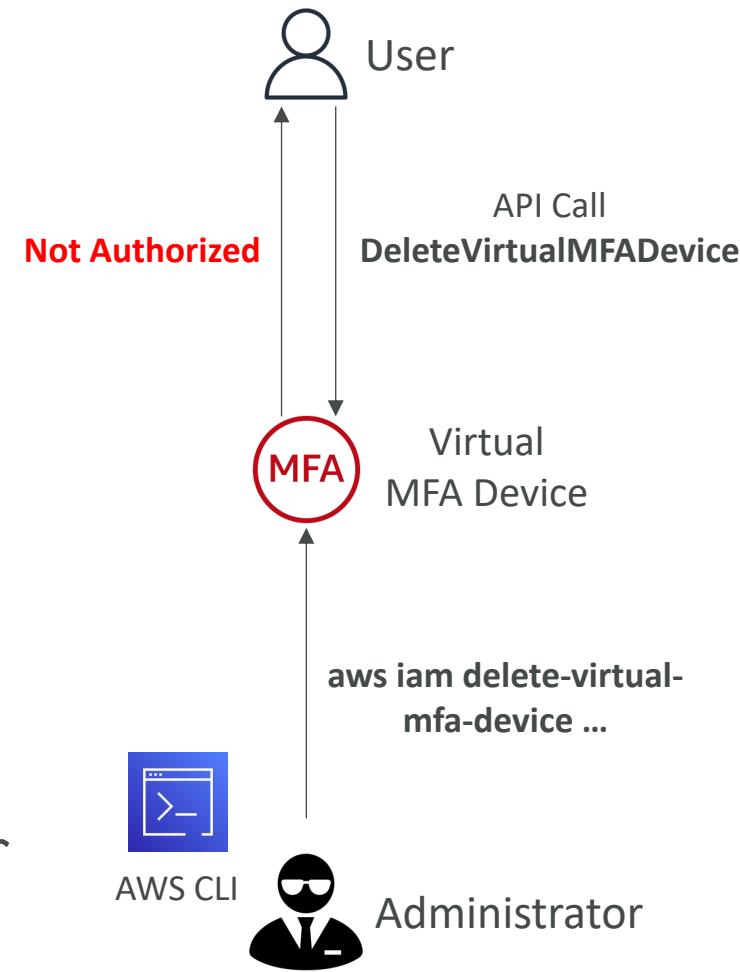
# IAM Conditions – MultiFactorAuthPresent

- Restrict access to AWS services for users not authenticated using MFA
- aws:MultiFactorAuthPresent
- Compatible with the AWS Console and the AWS CLI

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": "ec2:*",  
            "Resource": "*"  
        },  
        {  
            "Effect": "Deny",  
            "Action": ["ec2:StopInstances", "ec2:TerminateInstances"],  
            "Resource": "*",  
            "Condition": {  
                "BoolIfExists": {  
                    "aws:MultiFactorAuthPresent": false  
                }  
            }  
        }  
    ]  
}
```

# Not Authorized to Perform iam:DeleteVirtualMFADevice

- This error happens even the user has the correct IAM permissions
- This happens if someone began assigning a virtual MFA device to a user and then cancelled the process
  - E.g. created an MFA device for the user but never activates it
  - Must delete the existing MFA device then associate a new device
  - AWS recommends a policy that allows a user to delete their own virtual MFA device only if they are authenticated using MFA
- To fix the issue, the administrator must use the AWS CLI or AWS API to remove the existing but deactivated device



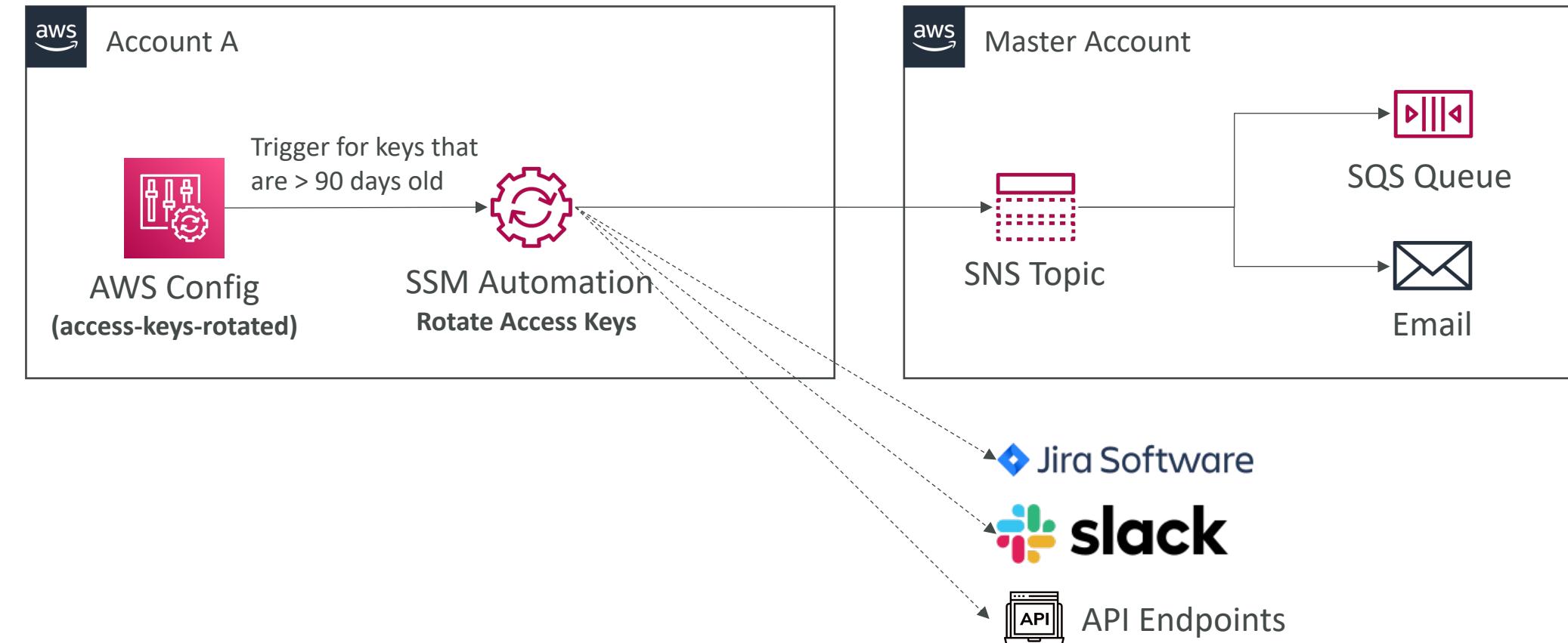
# IAM Credentials Report

- IAM users and the status of their passwords, access keys, and MFA devices
- Download using IAM Console, AWS API, AWS CLI, or AWS SDK
- Helps in auditing and compliance
- Generated as often as once every 4 hours

1	user	arn	user_creation_time	password_enabled	password_last_used	password_last_changed
2	<root_account>	arn:aws:iam::123456789012:root	2019-04-08T12:35:51Z	not_supported	2023-01-02T08:37:35Z	not_supported
3	alice	arn:aws:iam::123456789012:user/alice	2021-03-25T20:54:31Z	TRUE	2021-04-18T17:34:05Z	2021-03-25T20:54:37+00
4	bob	arn:aws:iam::123456789012:user/bob	2020-08-10T19:12:51Z	TRUE	2022-02-02T08:20:54Z	2020-08-10T19:12:56+00
5	admin	arn:aws:iam::123456789012:user/admin	2020-03-28T08:35:11Z	TRUE	2022-03-12T10:28:51Z	2020-03-28T08:35:15+00

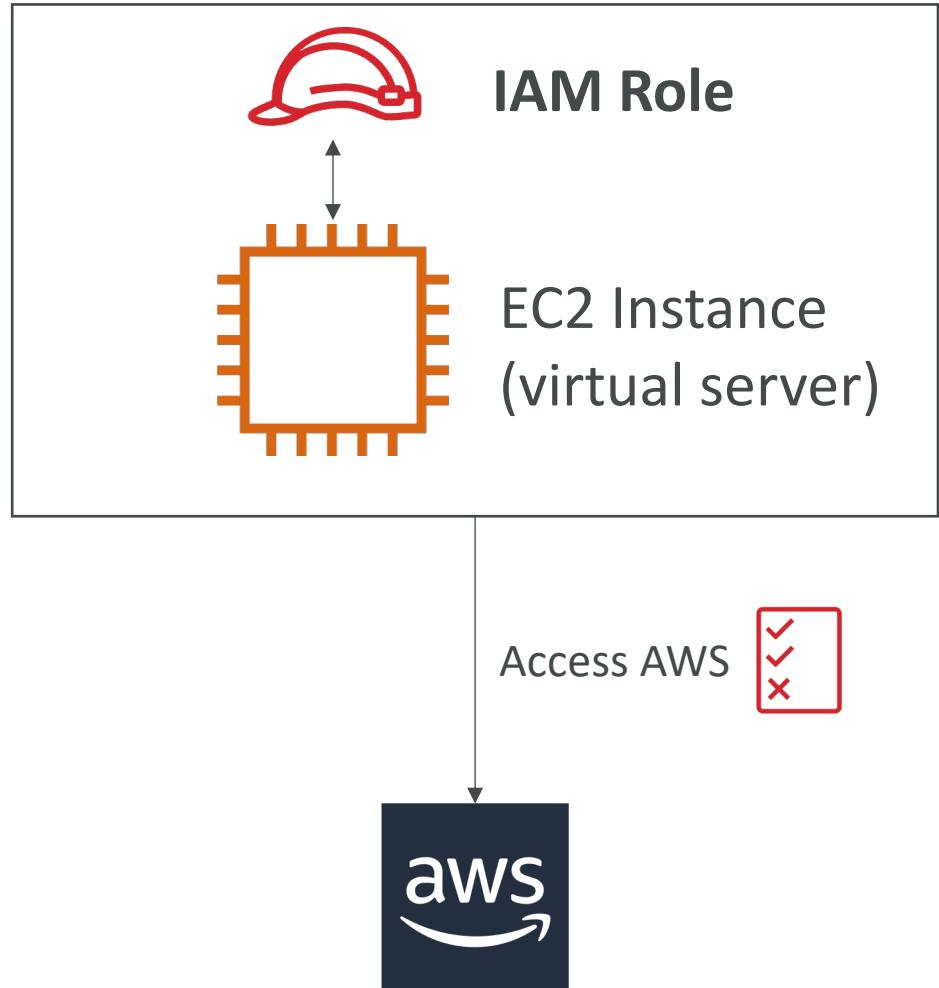
1	user	password_next_rotation	mfa_active	access_key_1_active	access_key_1_last_rotated	access_key_1_last_used	access_key_1_last_used_timestamp	access_key_2_active	access_key_2_last_rotated	access_key_2_last_used	access_key_2_last_used_timestamp
2	<root_account>	not_supported	TRUE	FALSE	N/A	N/A	N/A	N/A	N/A	N/A	FAILED
3	alice	N/A	FALSE	FALSE	N/A	N/A	N/A	N/A	N/A	N/A	FAILED
4	bob	N/A	TRUE	FALSE	2020-10-16T18:24:16+00:00	2022-03-27T11:05:01Z	us-east-1	ec2	TRUE	TRUE	TRUNCATED
5	admin	N/A	TRUE	FALSE	N/A	N/A	N/A	N/A	N/A	N/A	FAILED

# Managing Aged Access Keys through AWS Config Remediations



# IAM Roles for Services

- Some AWS service will need to perform actions on your behalf
- To do so, we will assign permissions to AWS services with IAM Roles
- Common roles:
  - EC2 Instance Roles
  - Lambda Function Roles
  - Roles for CloudFormation



# Delegate Passing Permissions to AWS Services

- You can grant users permissions to pass an IAM role to an AWS service
- Ensure that only approved users can configure an AWS service with an IAM role that grants permissions
- Grant **iam:PassRole** permission to the user's IAM user, role, or group
- **PassRole** is not an API call (no CloudTrail logs generated)
  - Review CloudTrail log that created or modified the resource receiving the IAM role

```
{  
  "Version": "2012-10-17",  
  "Statement": [{  
    "Effect": "Allow",  
    "Action": [  
      "iam:GetRole",  
      "iam:PassRole"  
    ],  
    "Resource": "arn:aws:iam::  
123456789012:role/EC2-roles-for-*"  
  }]  
}
```

**Allow User to Pass Only  
Approved Roles**

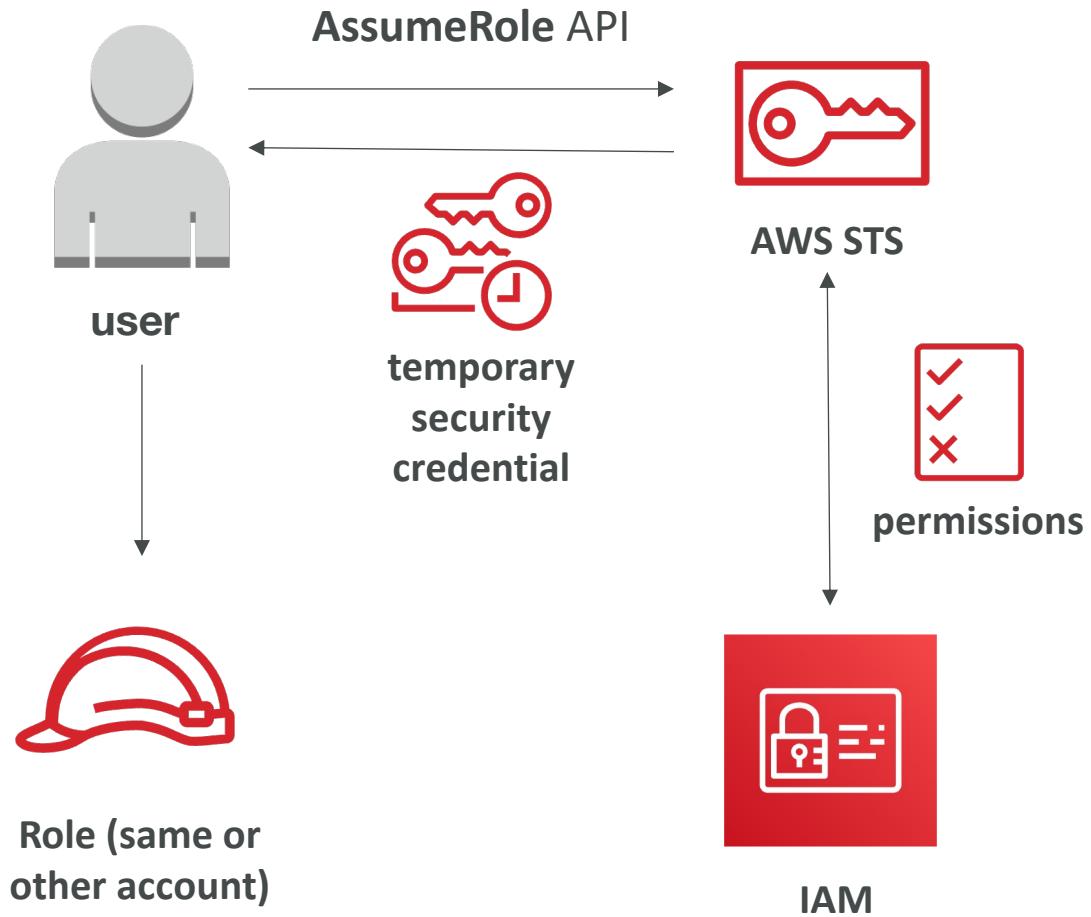


# AWS STS – Security Token Service

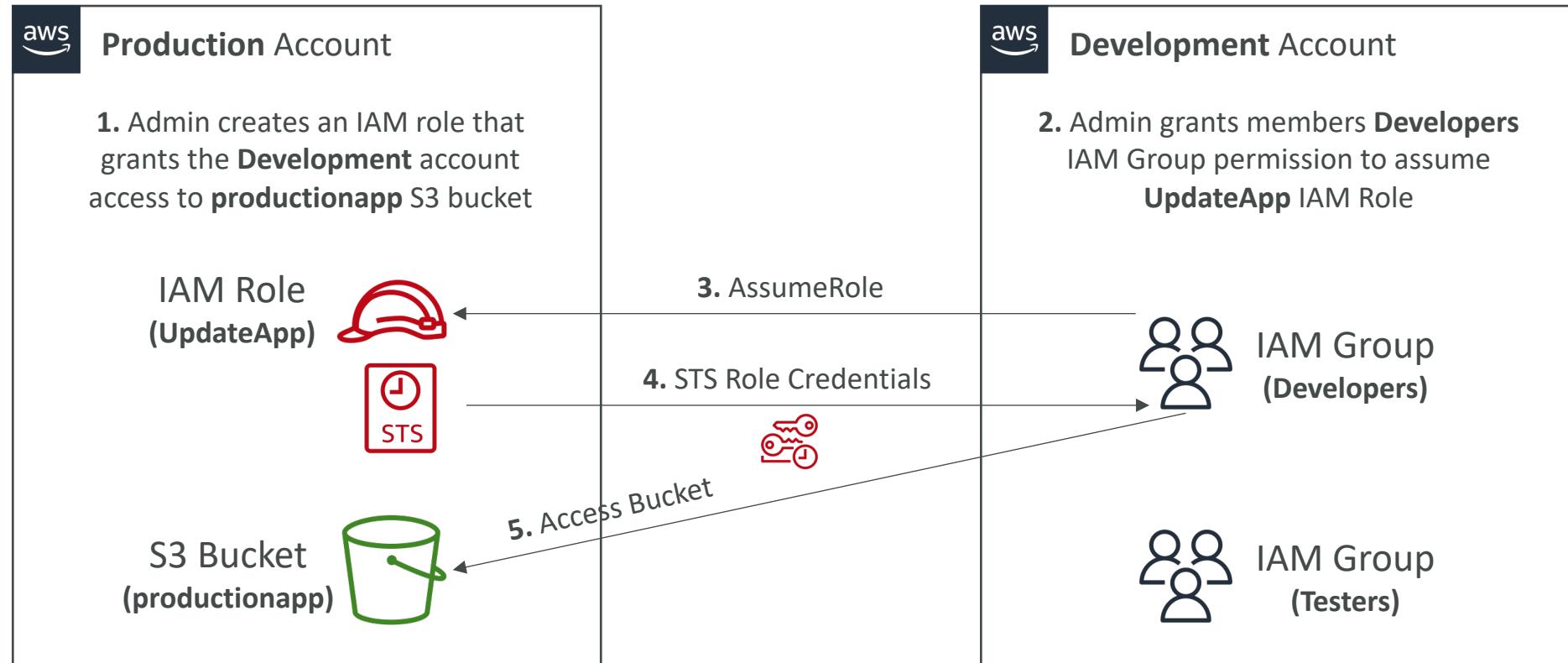
- Allows to grant limited and temporary access to AWS resources.
- Token is valid for up to one hour (must be refreshed)
- **AssumeRole**
  - Within your own account: for enhanced security
  - Cross Account Access: assume role in target account to perform actions there
- **AssumeRoleWithSAML**
  - return credentials for users logged with SAML
- **AssumeRoleWithWebIdentity**
  - return creds for users logged with an IdP (Facebook Login, Google Login, OIDC compatible...)
  - AWS recommends against using this, and using **Cognito** instead
- **GetSessionToken**
  - for MFA, from a user or AWS account root user

# Using STS to Assume a Role

- Define an IAM Role within your account or cross-account
- Define which principals can access this IAM Role
- Use AWS STS (Security Token Service) to retrieve credentials and impersonate the IAM Role you have access to (`AssumeRole API`)
- Temporary credentials can be valid between 15 minutes to 1 hour



# Cross-Account Access with STS



# STS – Version 1 vs. Version 2

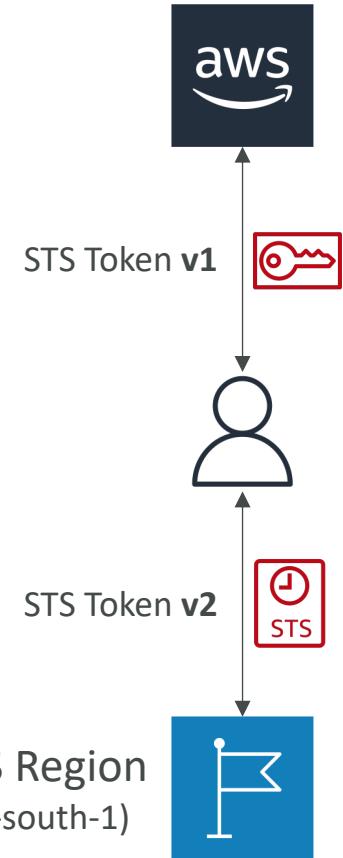
- **STS Version 1**

- By default, STS is available as a global single endpoint  
<https://sts.amazonaws.com>
- Only support AWS Regions that are enabled by default
- Option to enable “All Regions”

- **STS Version 2**

- Version 1 tokens do not work for new AWS Regions (e.g., me-south-1)
- Regional STS endpoints is available in all AWS Regions
- Reduce latency, built-in redundancy, increase session token validity
- STS Session Tokens from regional endpoints (STS Version 2) are valid in all AWS Regions

STS Global Endpoint  
<https://sts.amazonaws.com>



STS Regional Endpoint  
<https://sts.me-south-1.amazonaws.com>

# STS – Version 1 vs. Version 2

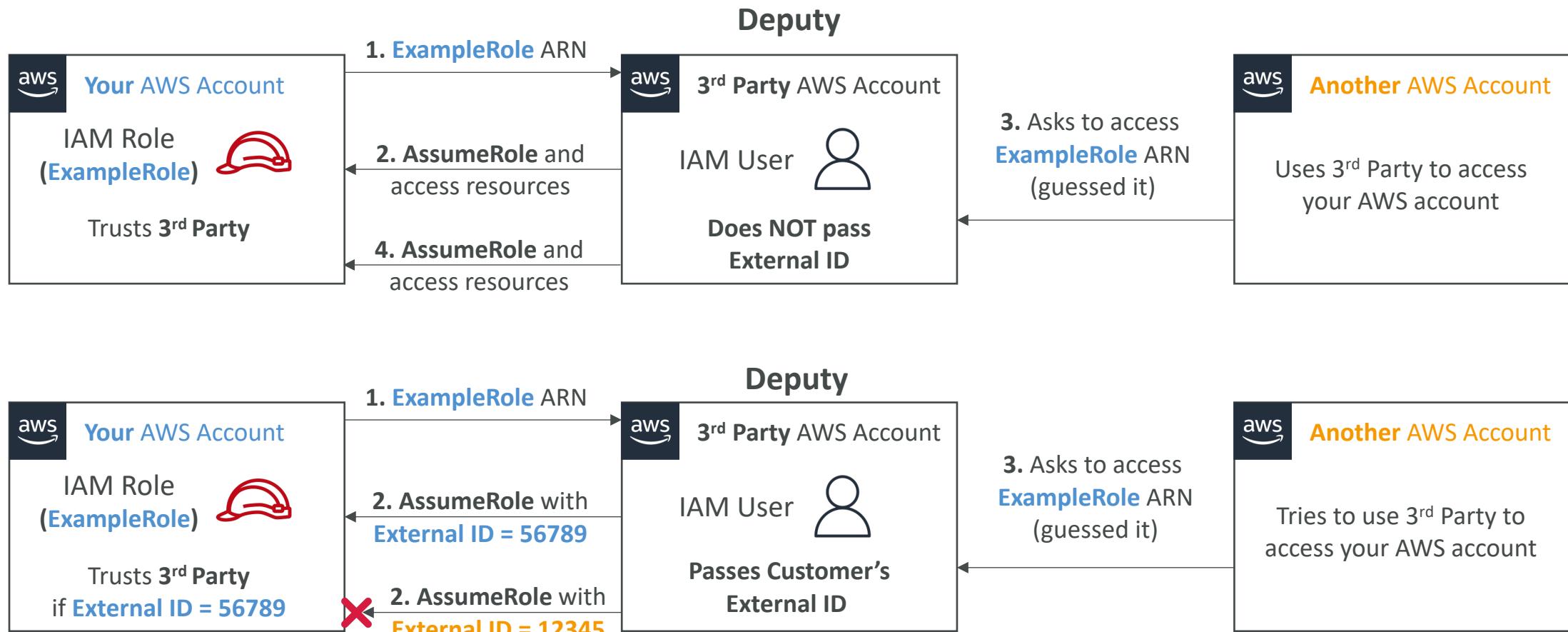
- Error: "An error occurred (AuthFailure) when calling the DescribeInstances operation: AWS was not able to validate the provided access credentials."
- To solve, two options:
  1. Use the Regional STS Endpoint (any region) which will return STS Tokens Version 2. Use the closest regional endpoint for lowest latency
  2. By default, the AWS STS calls to the STS global endpoint issues session tokens which are of Version 1 (Default AWS Regions). You can configure STS global endpoint to issue STS Tokens Version 2 (All AWS Regions)

# Granting Access to 3<sup>rd</sup> Party using External ID

- External ID is a piece of data that can be passed to AssumeRole API
- Allowing the IAM role to be assumed only if a certain value is present (External ID)
- It solves the Confused Deputy problem
- Prevent any other customer from tricking 3<sup>rd</sup> party into unwittingly accessing your resources

```
"Version": "2012-10-17",
"Statement": [
    {
        "Effect": "Allow",
        "Principal": { "AWS": "3rd Party Account ID" },
        "Action": "AssumeRole",
        "Condition": {
            "StringEquals": { "sts:ExternalId": "56789" }
        }
    }
]
```

# Granting Access to 3<sup>rd</sup> Party using External ID



Generated External ID = **56789** (generated by the deputy)

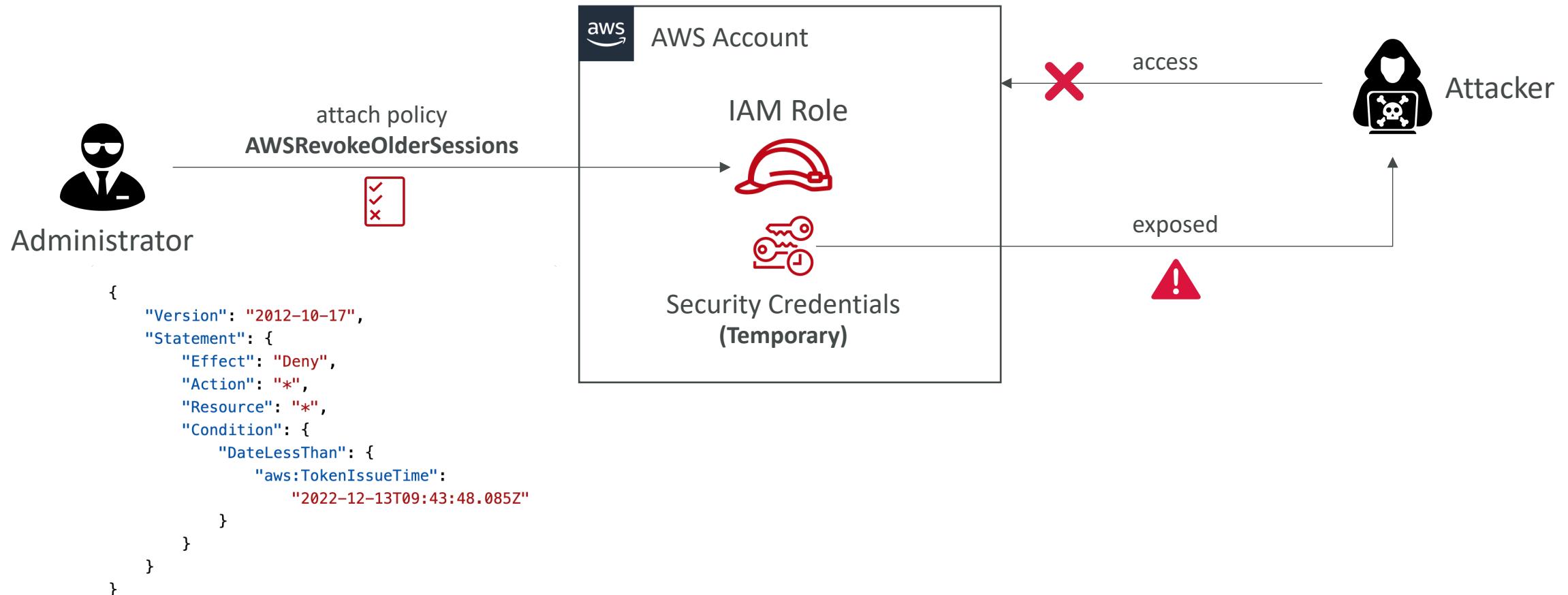
Generated External ID = **12345**

# Revoking IAM Role Temporary Credentials

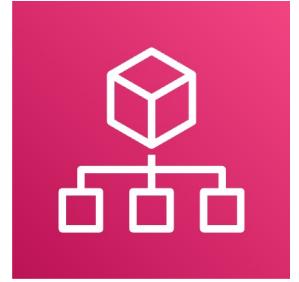
- Users usually have a long session duration time (e.g., 12 hours)
- If credentials are exposed, they can be used for the duration of the session
  - Immediately revoke all permissions to the IAM role's credentials issued before a certain time
  - AWS attaches a new inline IAM policy to the IAM role that denies all permissions (forces users to reauthenticate) if the token is too old
  - Doesn't affect users who assumes the IAM role after you revoke sessions (don't worry about deleting the policy)

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Deny",  
            "Action": "*",  
            "Resource": "*",  
            "Condition": {  
                "DateLessThan": "  
                    aws:TokenIssueTime":  
                    "2022-12-13T09:43:48.085Z"  
            }  
        }  
    ]  
}  
  
AWSRevokeOlderSessions Policy
```

# Revoking IAM Role Temporary Credentials

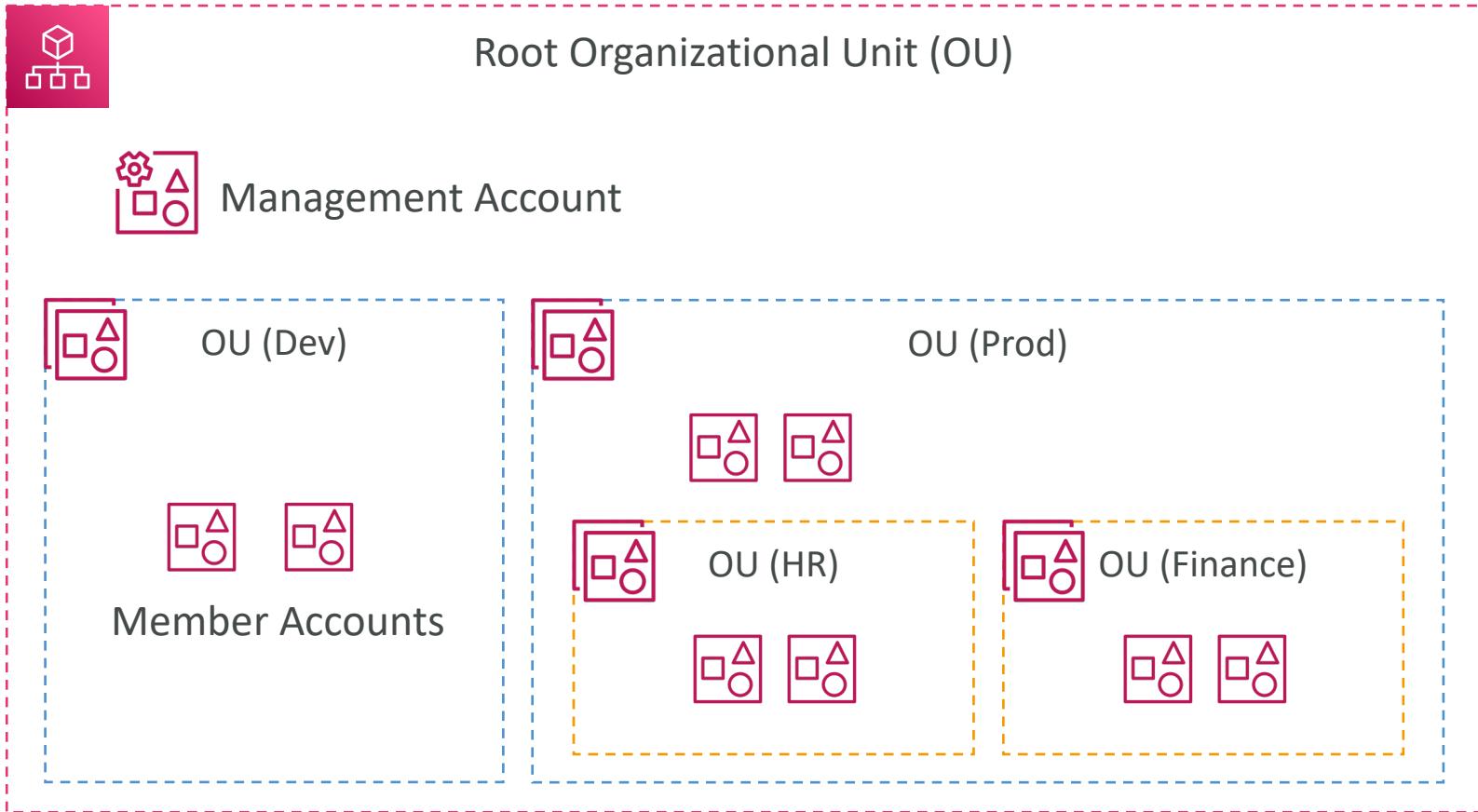


# AWS Organizations



- Global service
- Allows to manage multiple AWS accounts
- The main account is the management account
- Other accounts are member accounts
- Member accounts can only be part of one organization
- Consolidated Billing across all accounts - single payment method
- Pricing benefits from aggregated usage (volume discount for EC2, S3...)
- Shared reserved instances and Savings Plans discounts across accounts
- API is available to automate AWS account creation

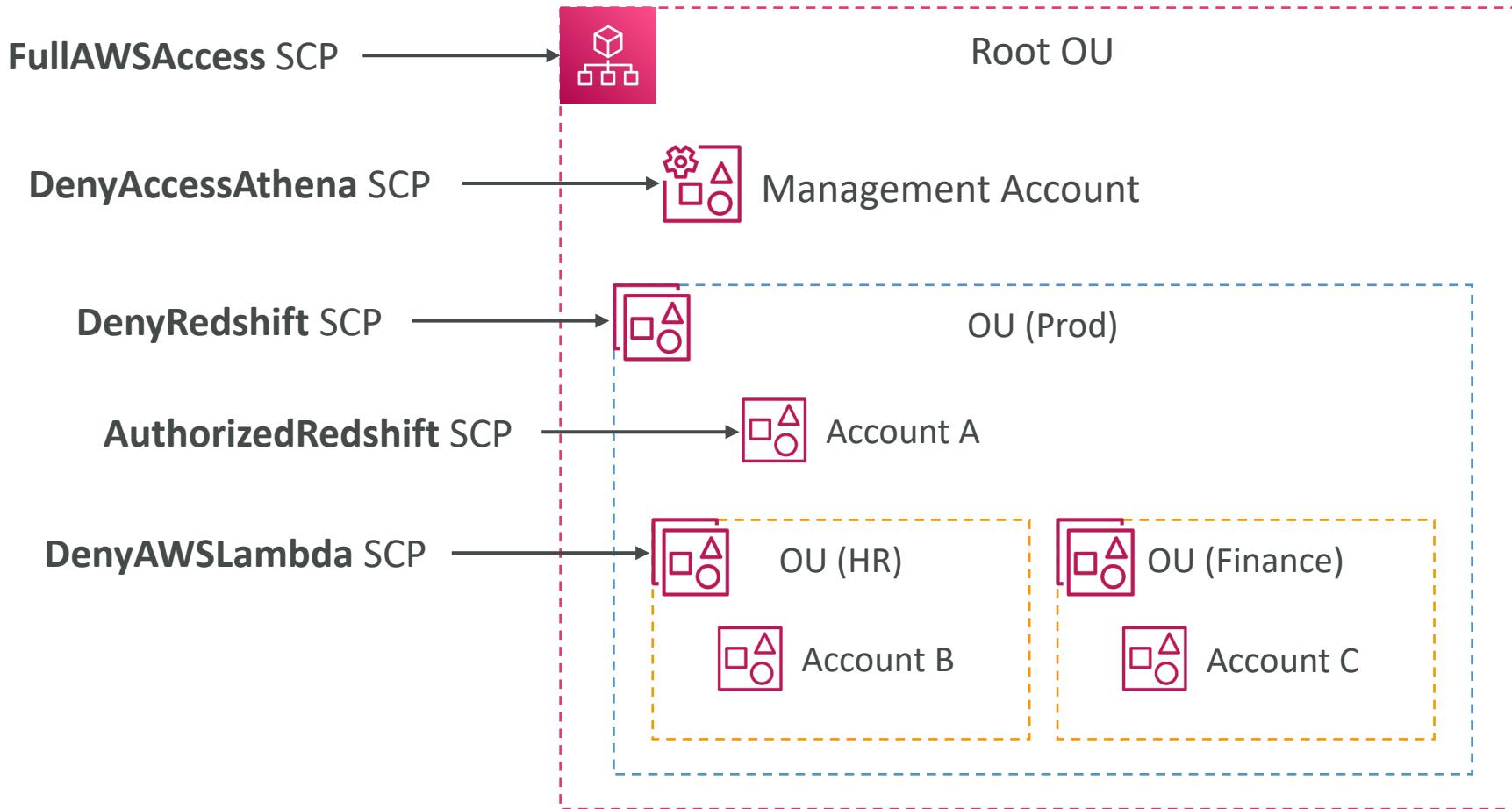
# AWS Organizations



# AWS Organizations

- Advantages
  - Multi Account vs One Account Multi VPC
  - Use tagging standards for billing purposes
  - Enable CloudTrail on all accounts, send logs to central S3 account
  - Send CloudWatch Logs to central logging account
  - Establish Cross Account Roles for Admin purposes
- Security: Service Control Policies (SCP)
  - IAM policies applied to OU or Accounts to restrict Users and Roles
  - They do not apply to the management account (full admin power)
  - Must have an explicit allow (does not allow anything by default – like IAM)

# SCP Hierarchy



- Management Account
  - Can do anything
  - (no SCP apply)
- Account A
  - Can do anything
  - EXCEPT access Redshift (explicit Deny from OU)
- Account B
  - Can do anything
  - EXCEPT access Redshift (explicit Deny from Prod OU)
  - EXCEPT access Lambda (explicit Deny from HR OU)
- Account C
  - Can do anything
  - EXCEPT access Redshift (explicit Deny from Prod OU)

# SCP Examples

## Blocklist and Allowlist strategies

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "AllowsAllActions",  
            "Effect": "Allow",  
            "Action": "*",  
            "Resource": "*"  
        },  
        {  
            "Sid": "DenyDynamoDB",  
            "Effect": "Deny",  
            "Action": "dynamodb:*",  
            "Resource": "*"  
        }  
    ]  
}
```

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "ec2:*",  
                "cloudwatch:/*"  
            ],  
            "Resource": "*"  
        }  
    ]  
}
```

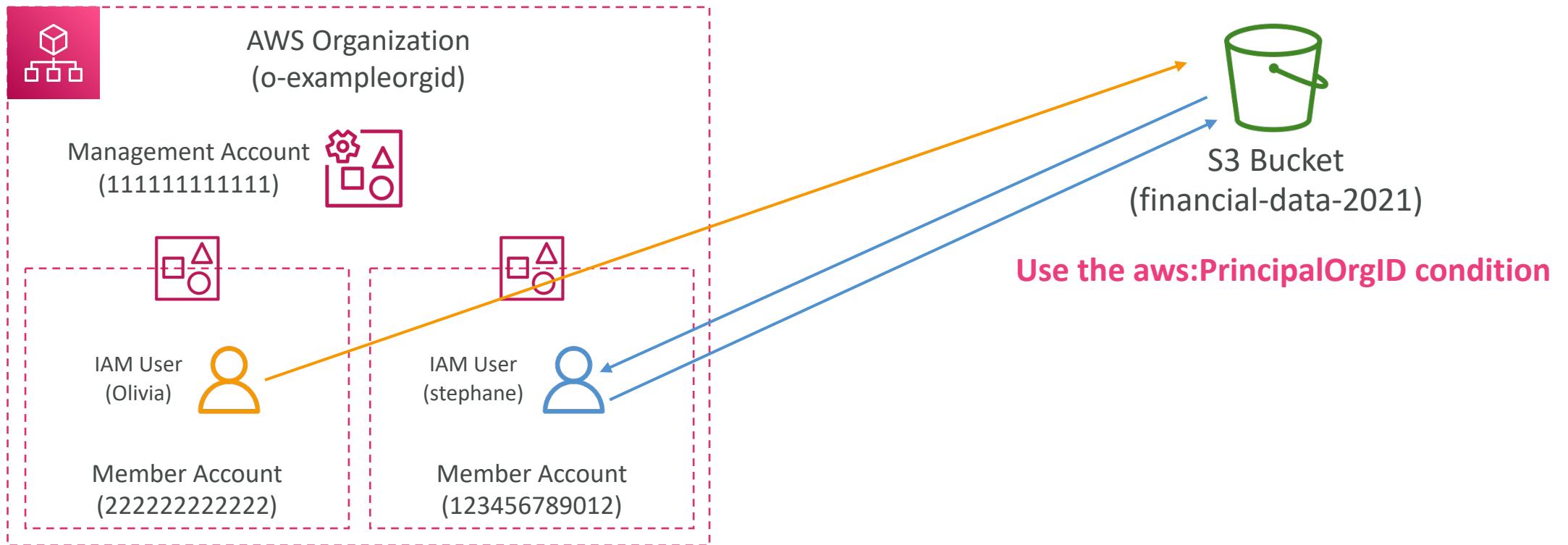
More examples: [https://docs.aws.amazon.com/organizations/latest/userguide/orgs\\_manage\\_policies\\_example-scps.html](https://docs.aws.amazon.com/organizations/latest/userguide/orgs_manage_policies_example-scps.html)

# AWS Organizations – Reserved Instances

- For billing purposes, the consolidated billing feature of AWS Organizations treats all the accounts in the organization as one account.
- This means that **all accounts** in the organization can receive the hourly cost benefit of Reserved Instances that are purchased by **any other account**.
- **The payer account (master account) of an organization** can turn off Reserved Instance (RI) discount and Savings Plans discount sharing for any accounts in that organization, including the payer account
- This means that RIs and Savings Plans discounts aren't shared between any accounts that have sharing turned off.
- To share an RI or Savings Plans discount with an account, both accounts must have sharing turned on.

# AWS Organizations – IAM Policies

- Use `aws:PrincipalOrgID` condition key in your resource-based policies to restrict access to IAM principals from accounts in an AWS Organization



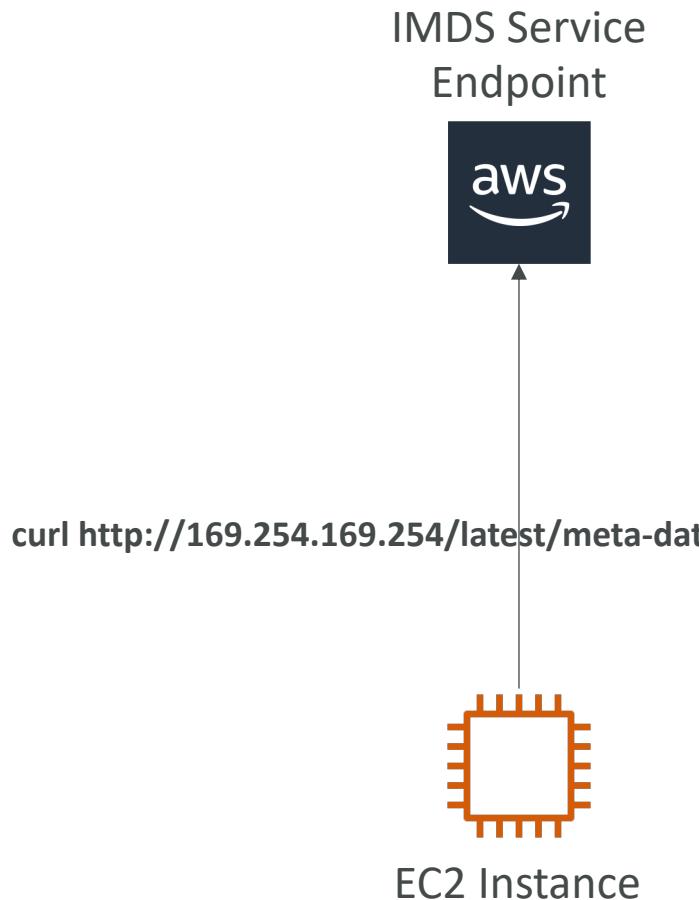
# AWS Organizations – Tag Policies

- Helps you standardize tags across resources in an AWS Organization
- Ensure consistent tags, audit tagged resources, maintain proper resources categorization, ...
- You define tag keys and their allowed values
- Helps with AWS Cost Allocation Tags and Attribute-based Access Control
- Prevent any non-compliant tagging operations on specified services and resources (has no effect on resources without tags)
- Generate a report that lists all tagged/non-compliant resources
- Use CloudWatch Events to monitor non-compliant tags

```
{  
  "tags": {  
    "costcenter": {  
      "tag_key": {  
        "@@assign": "CostCenter"  
      },  
      "tag_value": {  
        "@@assign": ["100", "200"]  
      },  
      "enforced_for": {  
        "@@assign": ["secretsmanager:*"]  
      }  
    }  
  }  
}
```

# AWS EC2 Instance Metadata Service (IMDS)

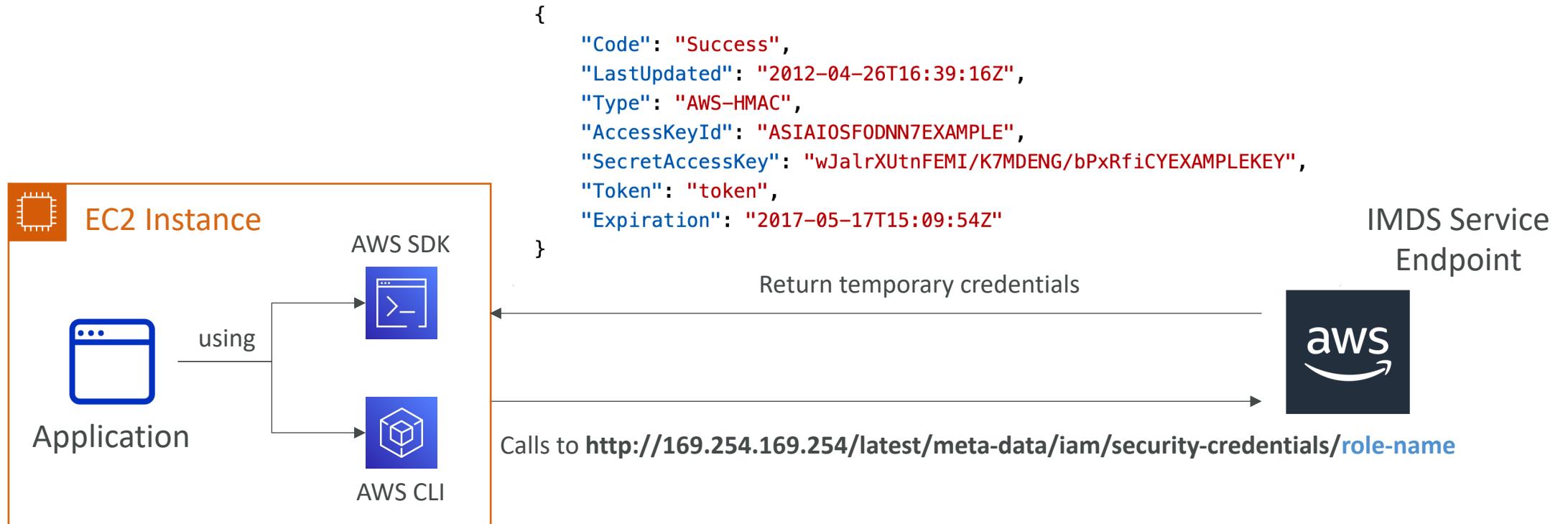
- Information about an EC2 instance (e.g., hostname, instance type, network settings, ...)
- Can be accessed from within the EC2 instance itself by making a request to the EC2 metadata service endpoint <http://169.254.169.254/latest/meta-data>
- Can be accessed using EC2 API or CLI tools (e.g., curl or wget)
- Metadata is stored in key-value pairs
- Useful for automating tasks such as setting up an instance's hostname, configuring networking, or installing software



# AWS EC2 Instance Metadata - Example

- ami-id, block-device-mapping/, instance-id, instance-type, network/hostname, local-hostname, local-ipv4, public-hostname, public-ipv4
- iam – InstanceProfileArn, InstanceId
- iam/security-credentials/role-name – temporary credentials for the role attached to your instance
- placement/ – launch Region, launch AZ, placement group name...
- security-groups – names of security groups
- tags/instance – tags attached to the instance

# EC2 Instance Role – How it works

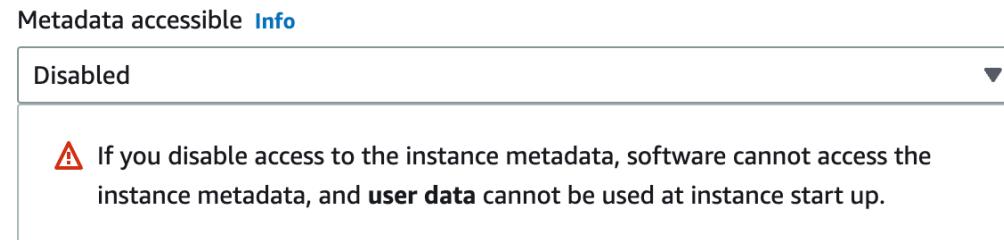


# EC2 Instance Metadata – Restrict Access

- You can use local firewall rules to disable access for some or all processes
  - `iptables` for Linux, PF or IPFW for FreeBSD

```
$ sudo iptables --append OUTPUT --proto tcp --destination 169.254.169.254 \
--match owner --uid-owner apache --jump REJECT
```

- Turn off access using AWS Console or AWS CLI (`HttpEndpoint=disabled`)



# IMDSv2 vs. IMDSv1

- IMDSv1 is accessing <http://169.254.169.254/latest/meta-data> directly
- IMDSv2 is more secure and is done in two steps:
  - I. Get Session Token (limited validity) – using headers & PUT

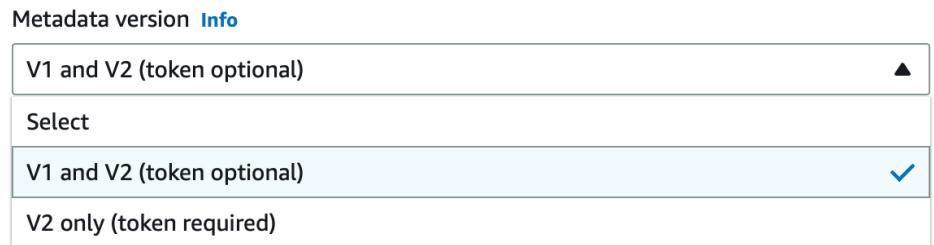
```
$ TOKEN=`curl -X PUT "http://169.254.169.254/latest/api/token"  
-H "X-aws-ec2-metadata-token-ttl-seconds: 21600" `
```

2. Use Session Token in IMDSv2 calls – using headers

```
$ curl http://169.254.169.254/latest/meta-data/profile  
-H "X-aws-ec2-metadata-token: $TOKEN"
```

# Requiring the usage of IMDSv2

- Both IMDSv1 and IMDSv2 are available (enabled by default)
- The CloudWatch Metric **MetadataNoToken** provide information on how much IMDSv1 is used
- You can force Metadata Version 2 at Instance Launch using either:
  - AWS console
  - AWS CLI “**HttpTokens: required**”
- You can require IMDSv2 when registering an AMI: `--imds-support v2.0`



# Require EC2 Role Credentials to be retrieved from IMDSv2

- AWS credentials provided by the IMDS now include an `ec2:RoleDelivery` IAM context key
  - 1.0 for IMDSv1
  - 2.0 for IMDS v2
- Attach this policy to the IAM Role of the EC2 Instance
- Or attach it to an S3 bucket to only require IMDSv2 when API calls are made by an IAM role
- Or attach it as an SCP in your account

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Deny",  
      "Action": "*",  
      "Resource": "*",  
      "Condition": {  
        "NumericLessThan": {  
          "ec2:RoleDelivery": "2.0"  
        }  
      }  
    }  
  ]  
}
```

# IAM Policy or SCP to force IMDSv2

Prevent the launch of an EC2 instance using old instance metadata (IMDSv1)

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Deny",  
            "Action": "ec2:RunInstances",  
            "Resource": "arn:aws:ec2:*::instance/*",  
            "Condition": {  
                "StringNotEquals": {  
                    "ec2:MetadataHttpTokens": "required"  
                }  
            }  
        }  
    ]  
}
```

Prevent modifying a running EC2 instance using **ModifyInstanceMetadataOptions** API to re-enable old instance metadata (IMDSv1)

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Deny",  
            "Action": "ec2:ModifyInstanceMetadataOptions",  
            "Resource": "*",  
            "Condition": {  
                "StringNotLike": {  
                    "aws:PrincipalARN": "arn:aws:iam::*:role/ec2-imds-admins"  
                }  
            }  
        }  
    ]  
}
```

# How Authorization Works in Amazon S3

- **User Context**

- Is the IAM principal authorized by the parent AWS account (IAM policy) ?
- If the parent owns the bucket or object, then bucket policy/ACL or object ACL is evaluated
- If the parent owns the bucket/object, it can grant permissions to its IAM principals using Identity-Based Policy or Resource-Based Policy

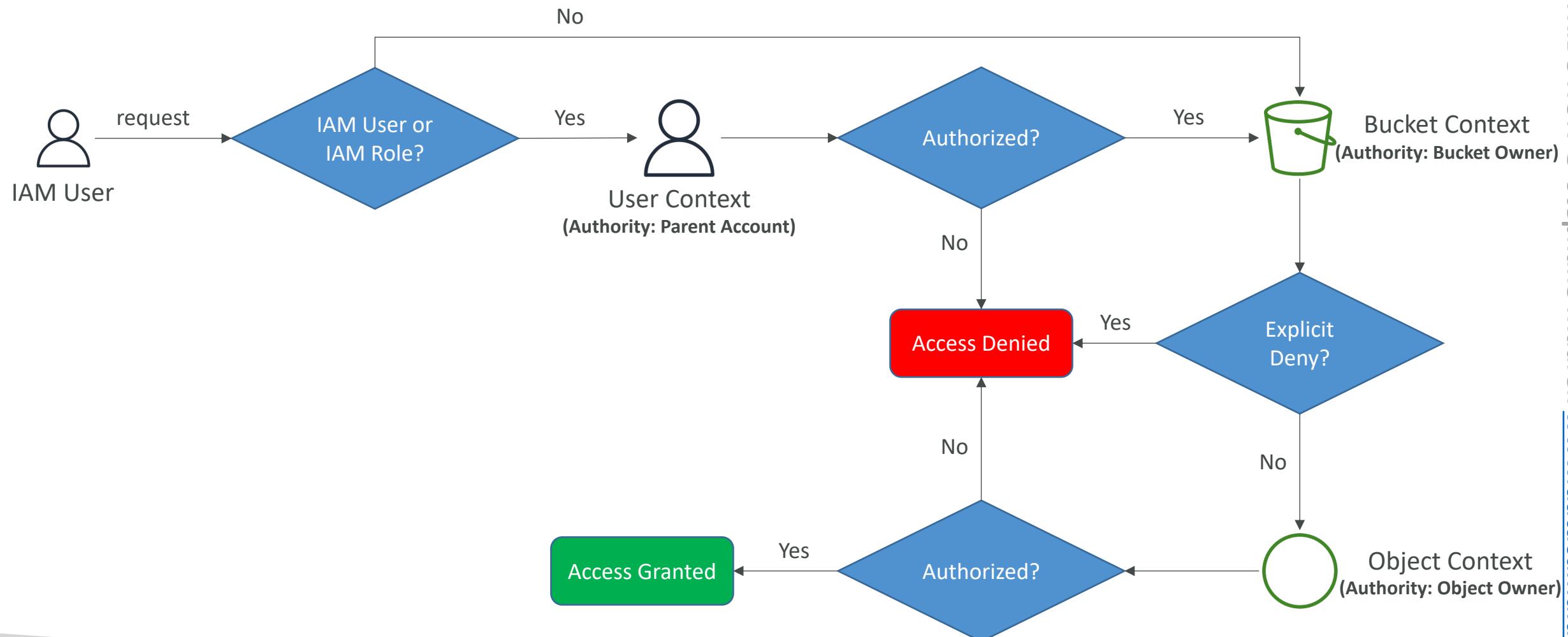
- **Bucket Context**

- Evaluates the policies of the AWS account that owns the bucket (check for Explicit Deny)

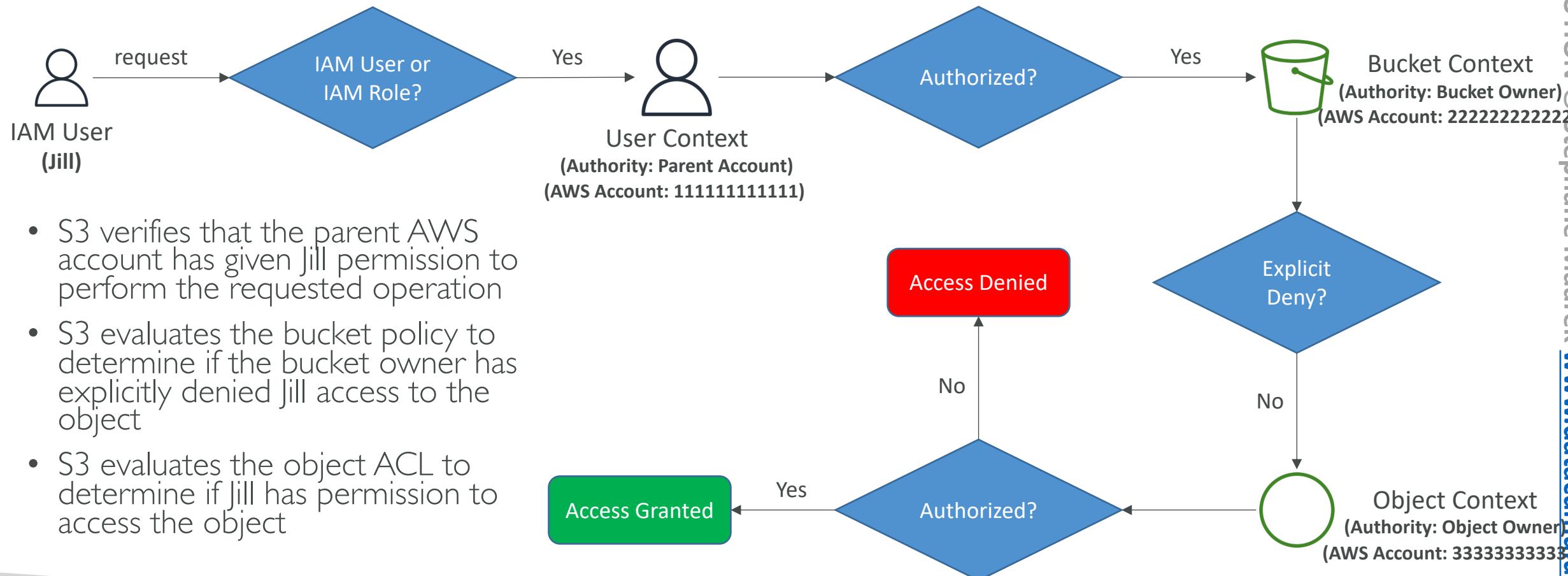
- **Object Context**

- Requester must have permission from the object owner (using Object ACL)
- If bucket owner = object owner, then access granted using Bucket Policy
- If bucket owner != object owner, then access granted using object owner ACL
- If you want to own all objects in your bucket and only use Bucket Policy and IAM-Based Policies to grant access, **enable Bucket Owner Enforced Setting for Object Ownership**
  - Then Bucket and objects ACLs can't be edited and are no longer considered for access

# How Authorization Works in Amazon S3



# Object Operation Request example: 3 accounts



# Bucket Operations vs Object Operations

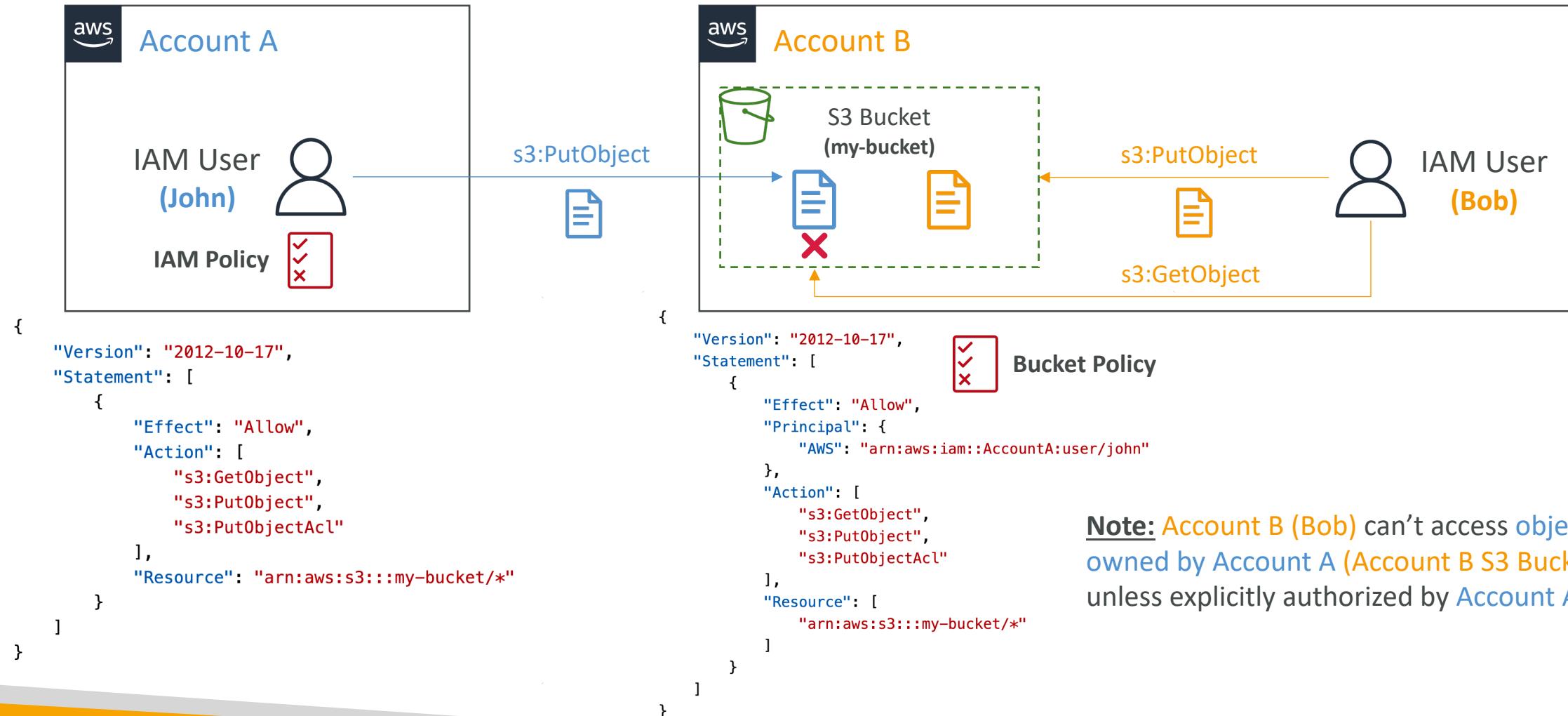
- s3>ListBucket permission applies to  
arn:aws:s3:::test
- => bucket level permission
- s3GetObject, s3PutObject,  
s3DeleteObject applies to  
arn:awn:s3:::test/\*
- => object level permission

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Action": ["s3>ListBucket"],  
      "Resource": "arn:aws:s3:::test"  
    },  
    {  
      "Effect": "Allow",  
      "Action": [  
        "s3PutObject",  
        "s3GetObject",  
        "s3DeleteObject"  
      ],  
      "Resource": "arn:aws:s3:::test/*"  
    }  
  ]  
}
```

# Cross-Account Access to Objects in S3 Buckets

- Use one of the following to grant cross-account access to S3 objects:
- IAM Policies and S3 Bucket Policy
- IAM Policies and Access Control Lists (ACLs)
  - ACLs only works if Bucket Owner Enforced setting = Disabled
  - By default, all newly created buckets have Bucket Owner Enforced = Enabled
- Cross-Account IAM Roles

# IAM Policies and Resource-Based Bucket Policies



# Object Permissions in Cross-Account Setting

- Account A user can make sure it gives up object ownership by granting the object ownership to the Account B administrator
- Using ACL: with adding condition that requests include **ACL-specific headers**, that either:
  - Grant full permissions explicitly (ex: x-amz-grant-full-control)
  - Use Canned ACL (see next)
- Using S3 Object Ownership (bucket-level setting to disable ACLs)

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Principal": {  
                "AWS": "arn:aws:iam::AccountB-ID:user/AccountBadmin"  
            },  
            "Action": "s3:PutObject",  
            "Resource": "arn:aws:s3:::awsexamplebucket1/*",  
            "Condition": {  
                "StringEquals": {  
                    "s3:x-amz-grant-full-control": "id=AccountA-CanonicalUserID"  
                }  
            }  
        },  
        {  
            "Effect": "Deny",  
            "Principal": {  
                "AWS": "arn:aws:iam::AccountB-ID:user/AccountBadmin"  
            },  
            "Action": "s3:PutObject",  
            "Resource": "arn:aws:s3:::awsexamplebucket1/*",  
            "Condition": {  
                "StringNotEquals": {  
                    "s3:x-amz-grant-full-control": "id=AccountA-CanonicalUserID"  
                }  
            }  
        }  
    ]  
}
```

# Canned ACL

## Object Permissions in Cross-Account Setting

- You can require the **x-amz-acl** header with a Canned ACL granting full control permission to the bucket owner
- To require the **x-amz-acl** header in the request, specify the **s3:x-amz-acl** condition key

```
"Condition": {  
    "StringEquals": {  
        "s3:x-amz-acl": "bucket-owner-full-control"  
    }  
}
```

# S3 Canned ACL – Deep Dive

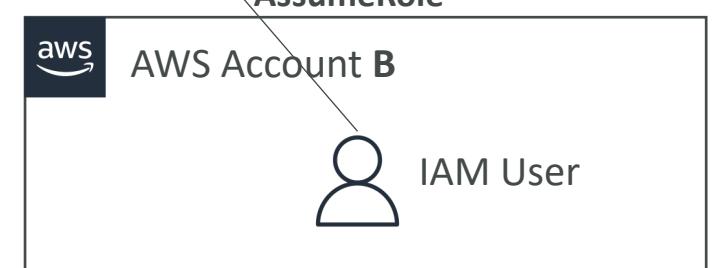
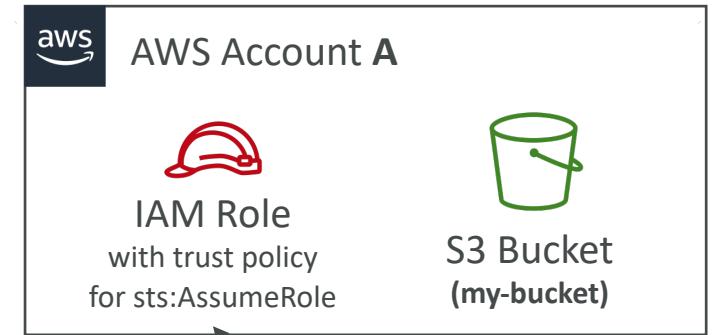
- Canned ACL are “Shortcuts ACLs”
- ACL are NOT recommended (& disabled by default since Apr 2023)
- By default, enable “Object Ownership” so that only Bucket Policies, IAM Policies, SCP, and VPC Endpoint Policies control access to your S3 Bucket Objects
- The bucket owner will own all the objects

Canned ACL	Applies to	Permissions Added to ACL
private	Bucket & Object	Owner gets FULL_CONTROL. No one else has access rights (default).
public-read	Bucket & Object	Owner gets FULL_CONTROL. AllUsers group gets READ access.
public-read-write	Bucket & Object	Owner gets FULL_CONTROL. AllUsers group gets READ and WRITE access. (not recommended)
aws-exec-read	Bucket & Object	Owner gets FULL_CONTROL. EC2 gets READaccess to GET an Amazon Machine Image (AMI).
authenticated-read	Bucket & Object	Owner gets FULL_CONTROL. AuthenticatedUsers group gets READ access.
bucket-owner-read	Object	Object owner gets FULL_CONTROL. Bucket owner gets READ access. (S3 ignores it if you specify it when creating a bucket)
bucket-owner-full-control	Object	Both the object owner and the bucket owner get FULL_CONTROL over the object. (S3 ignores it if you specify it when creating a bucket)
log-delivery-write	Bucket	LogDelivery group gets WRITE and READ ACPpermissions on the bucket.

# Cross-Account IAM Roles

- You can use cross-account IAM roles to centralize permission management when providing cross-account access to multiple services
- Example: access to S3 objects that are stored in multiple S3 buckets
- Bucket Policy is not required as the API calls to S3 come from within the account (through the assumed IAM role)

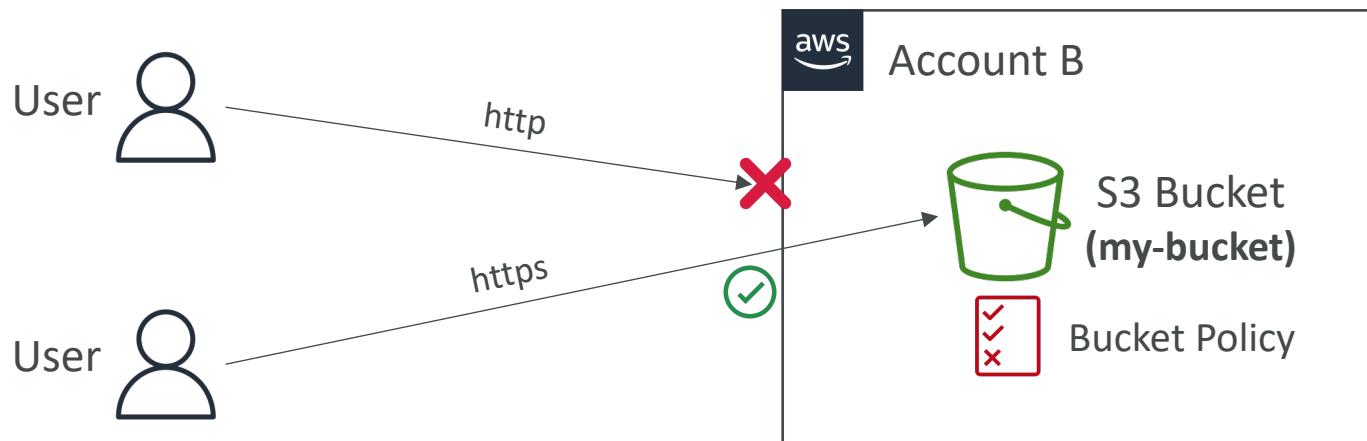
```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {"Effect": "Allow",  
     "Action": [  
       "s3:GetObject",  
       "s3:PutObject"  
     ],  
     "Resource": "arn:aws:s3:::my-bucket/*"  
   }  
}
```



```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {"Effect": "Allow",  
     "Action": "sts:AssumeRole",  
     "Resource": "arn:aws:iam::AccountA:role/AccountARole"  
   }  
}
```

# Sample Bucket Policies

## Force HTTPS In Flight



```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Deny",  
      "Principal": "*",  
      "Action": "s3:GetObject",  
      "Resource": "arn:aws:s3:::my-bucket/*",  
      "Condition": {  
        "Bool": {  
          "aws:SecureTransport": "false"  
        }  
      }  
    }  
  ]  
}
```

# S3 Bucket Policy – Restrict Public IP Address

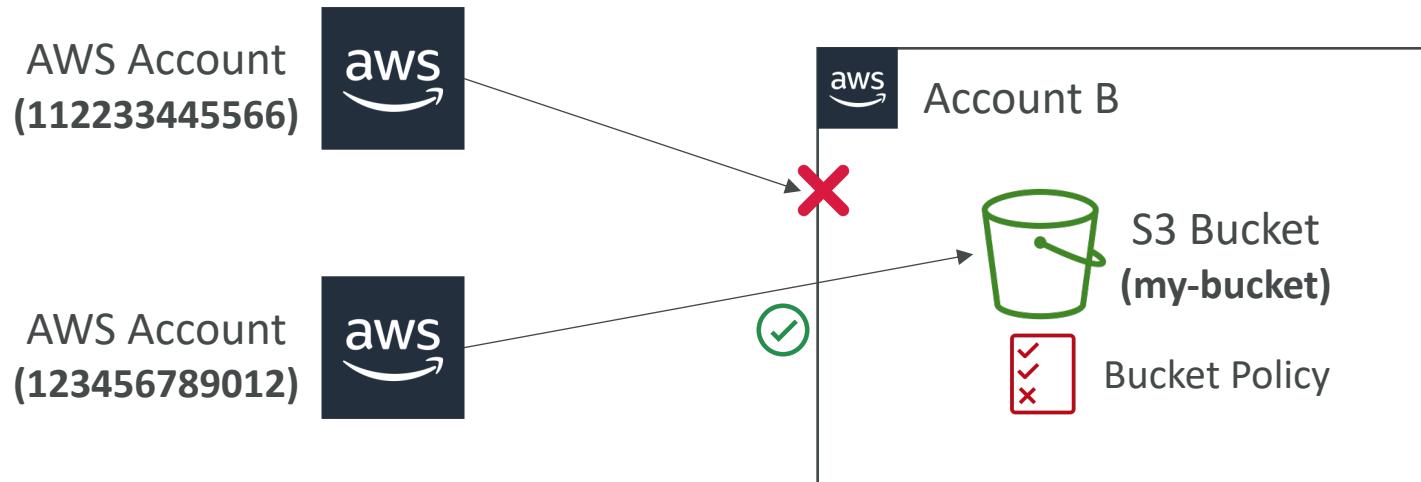


## Bucket Policy

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Deny",  
      "Principal": "*",  
      "Action": "s3:*",  
      "Resource": "arn:aws:s3:::my-bucket/*",  
      "Condition": {  
        "NotIpAddress": {  
          "aws:SourceIp": ["11.11.11.11/24"]  
        }  
      }  
    }  
  ]  
}
```

# S3 Bucket Policy – Restrict by User ID

Block traffic to the bucket unless request is from specified **users (same AWS account)**



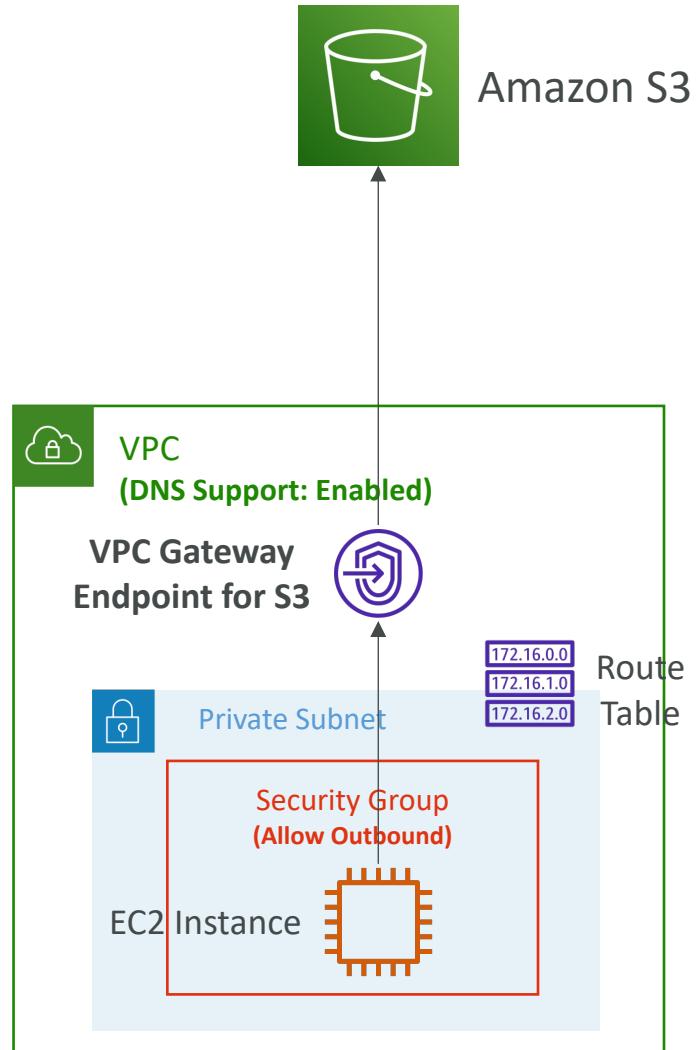
```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Deny",  
      "Principal": "*",  
      "Action": "s3:*",  
      "Resource": "arn:aws:s3:::my-bucket/*",  
      "Condition": {  
        "StringNotLike": {  
          "aws:userId": [  
            "RoleIDExample:*",  
            "UserIDExample",  
            "123456789012"  
          ]  
        }  
      }  
    }  
  ]  
}
```

Role ID  
User ID  
AWS Account ID (root user)

# VPC Gateway Endpoint for Amazon S3

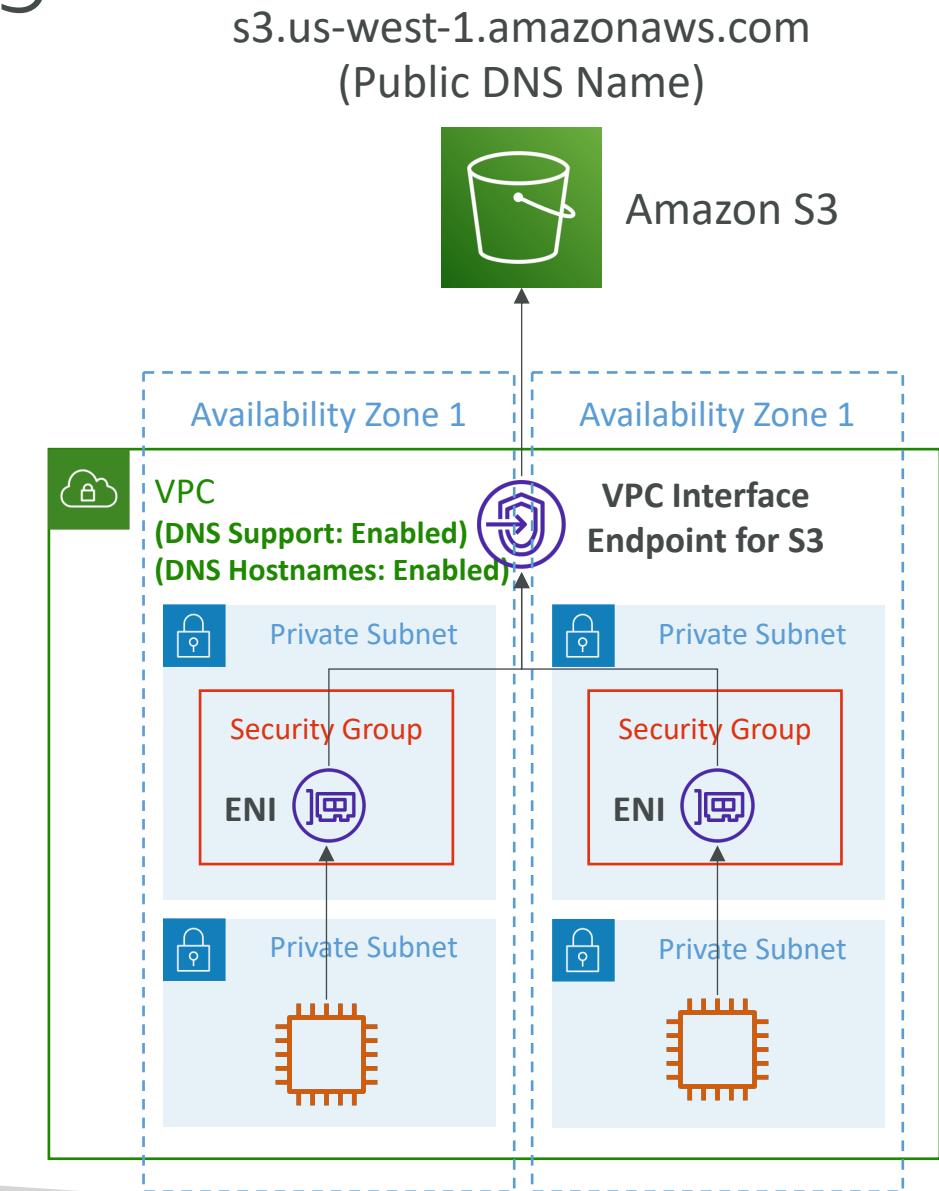
- No cost
- Only accessed by resources in the VPC where it's created
- Make sure “DNS Support” is Enabled
- Keep on using the public DNS of Amazon S3
- Make sure Outbound rules of SG of EC2 instance allows traffic to S3

Outbound rules (1/1)						
<input type="button" value="C"/> Manage tags <input type="button" value="Edit outbound rules"/>						
Name	Security group rule...	IP version	Type	Protocol	Port range	Destination
<input checked="" type="checkbox"/>	sgr-0ee382d4b7d379...	-	HTTPS	TCP	443	pl-63a5400a (com.amazonaws.us-east-1.s3) <input type="button" value="Edit"/>



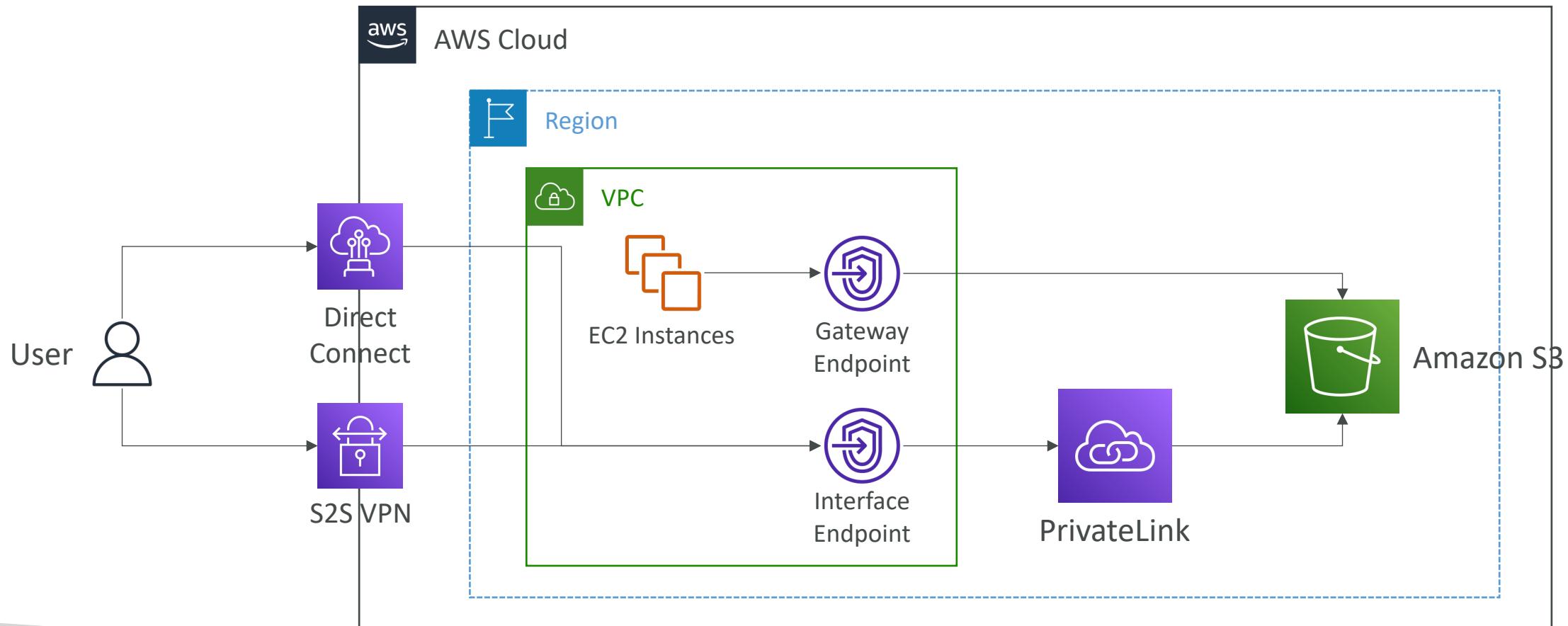
# VPC Interface Endpoint for S3

- ENI(s) are deployed in your Subnets (Security Groups can be attached to ENIs)
- Can access from on-premises (VPN or Direct Connect)
- Costs \$0.01 per hour per AZ
- The public hostname of a service will resolve to the private Endpoint Interface hostname
- Both VPC Setting “Enable DNS hostnames” and “Enable DNS Support” must be ‘true’
- No “Private DNS name” option for VPC Interface Endpoint for S3

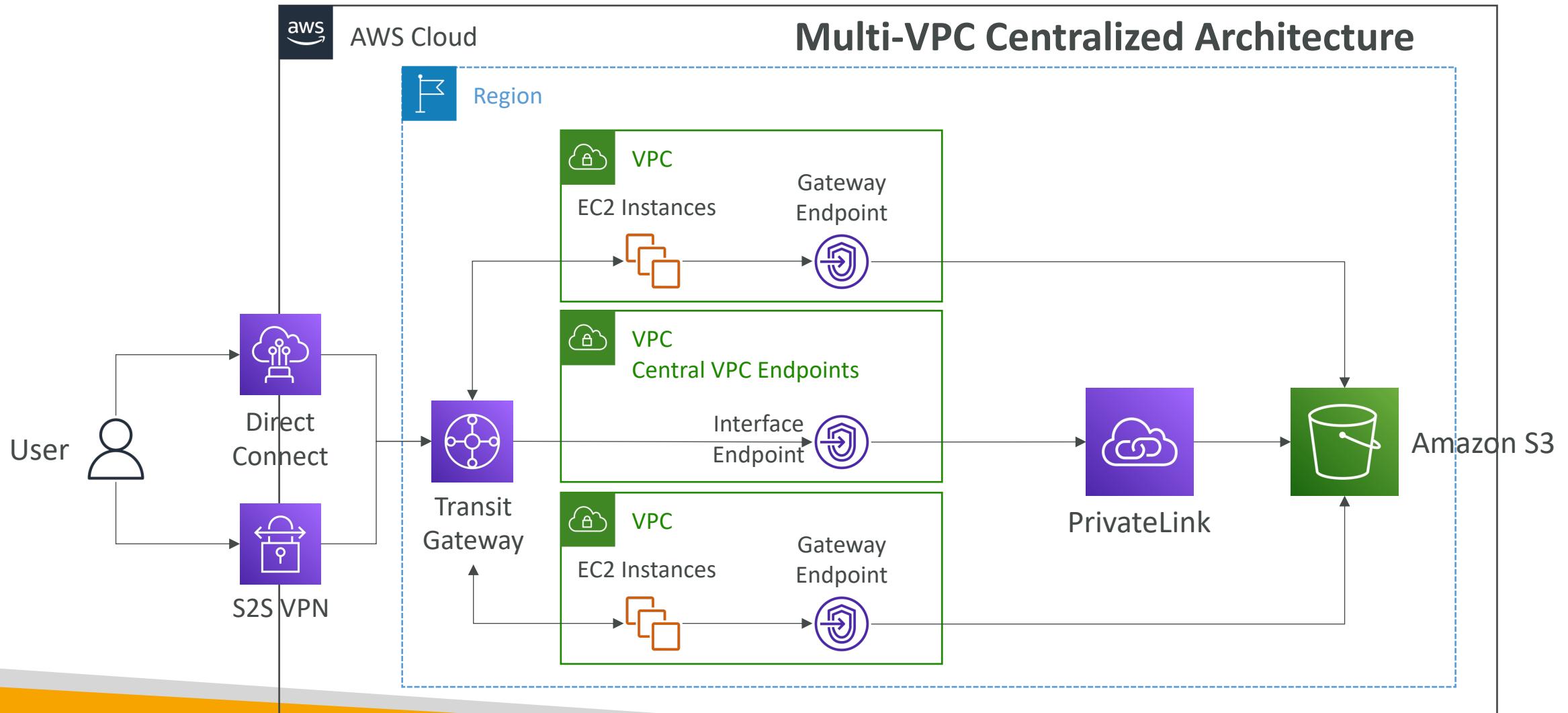


# Amazon S3 Security – VPC Endpoint Strategies

## Single VPC Architecture



# Amazon S3 Security – VPC Endpoint Strategies



# VPC Endpoint Restrictions

## aws:SourceVpc and aws:SourceVpce

Restrict Access from  
**specific VPCs**

```
{  
    "Effect": "Deny",  
    "Action": "s3:*",  
    "Principal": "*",  
    "Resource": [  
        "arn:aws:s3:::my-bucket",  
        "arn:aws:s3:::my-bucket/*"  
    ],  
    "Condition": {  
        "StringNotEquals": {  
            "aws:SourceVpc": [  
                "vpce-111bbb22"  
            ]  
        }  
    }  
}
```

Restrict Access from  
**Specific VPC Endpoints**

```
{  
    "Effect": "Deny",  
    "Action": "s3:*",  
    "Principal": "*",  
    "Resource": [  
        "arn:aws:s3:::my-bucket",  
        "arn:aws:s3:::my-bucket/*"  
    ],  
    "Condition": {  
        "StringNotEquals": {  
            "aws:SourceVpce": [  
                "vpce-1111111",  
                "vpce-2222222"  
            ]  
        }  
    }  
}
```

# IP Address Restrictions

## aws:SourceIp and aws:VpcSourceIp

Restrict Access from  
**Public IP Addresses**

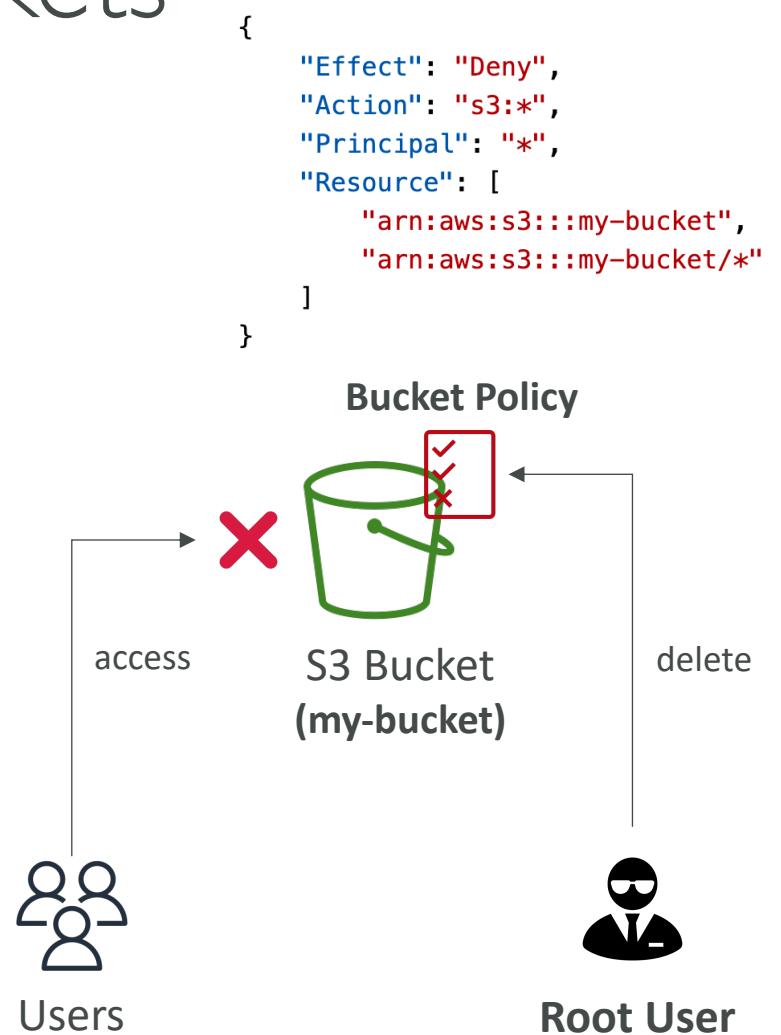
```
{  
    "Effect": "Deny",  
    "Action": "s3:*",  
    "Principal": "*",  
    "Resource": [  
        "arn:aws:s3:::my-bucket",  
        "arn:aws:s3:::my-bucket/*"  
    ],  
    "Condition": {  
        "NotIpAddress": {  
            "aws:SourceIp": [  
                "192.0.2.0/24",  
                "203.0.113.0/24"  
            ]  
        }  
    }  
}
```

Restrict Access from  
**Private IP Addresses**

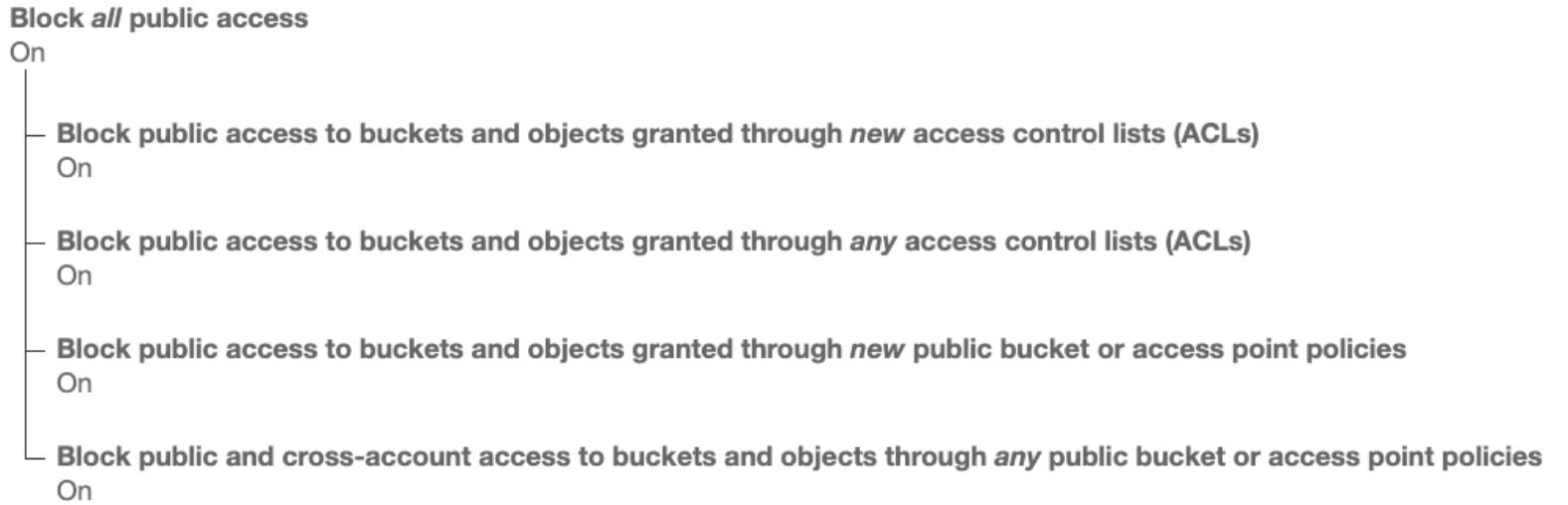
```
{  
    "Effect": "Deny",  
    "Action": "s3:*",  
    "Principal": "*",  
    "Resource": [  
        "arn:aws:s3:::my-bucket",  
        "arn:aws:s3:::my-bucket/*"  
    ],  
    "Condition": {  
        "NotIpAddress": {  
            "aws:VpcSourceIp": [  
                "10.1.1.1/32",  
                "172.1.1.1/32"  
            ]  
        }  
    }  
}
```

# Regain Access to Locked S3 Buckets

- If you incorrectly configured your S3 bucket policy to deny access to everyone (Deny s3:\*, Principal:\*)
- You must delete the S3 bucket policy using the AWS account root user
- Note: Deny statements in IAM policies do not affect the root account

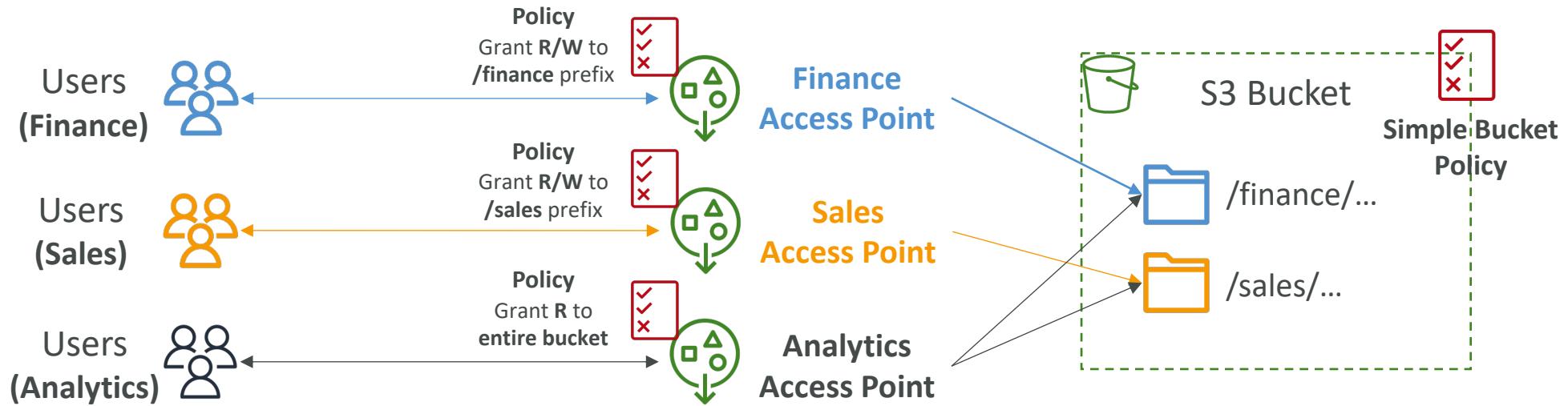


# Bucket settings for Block Public Access



- These settings were created to prevent company data leaks
- If you know your bucket should never be public, leave these on
- Can be set at the account level

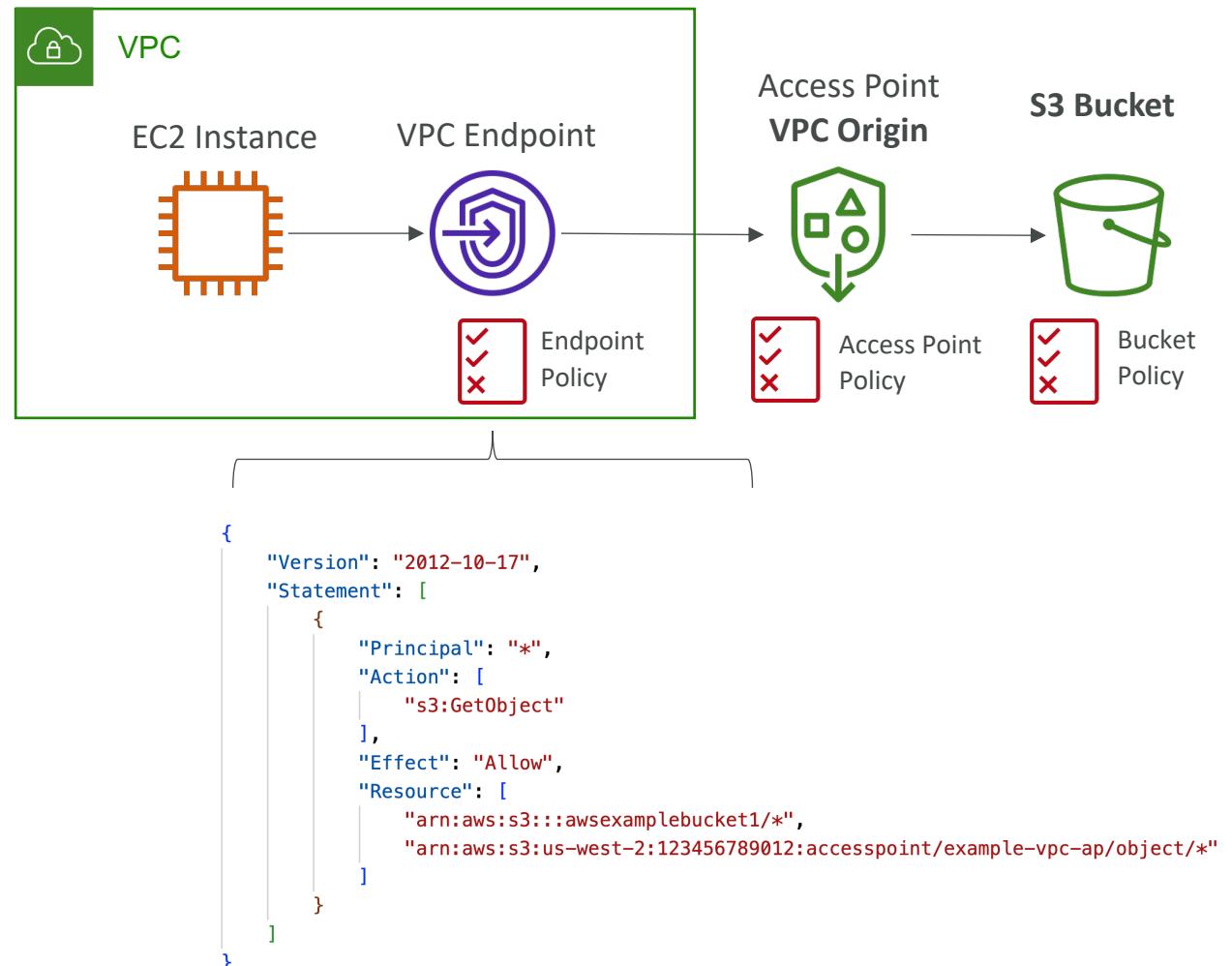
# S3 – Access Points



- Access Points simplify security management for S3 Buckets
- Each Access Point has:
  - its own DNS name (Internet Origin or VPC Origin)
  - an access point policy (similar to bucket policy) – manage security at scale

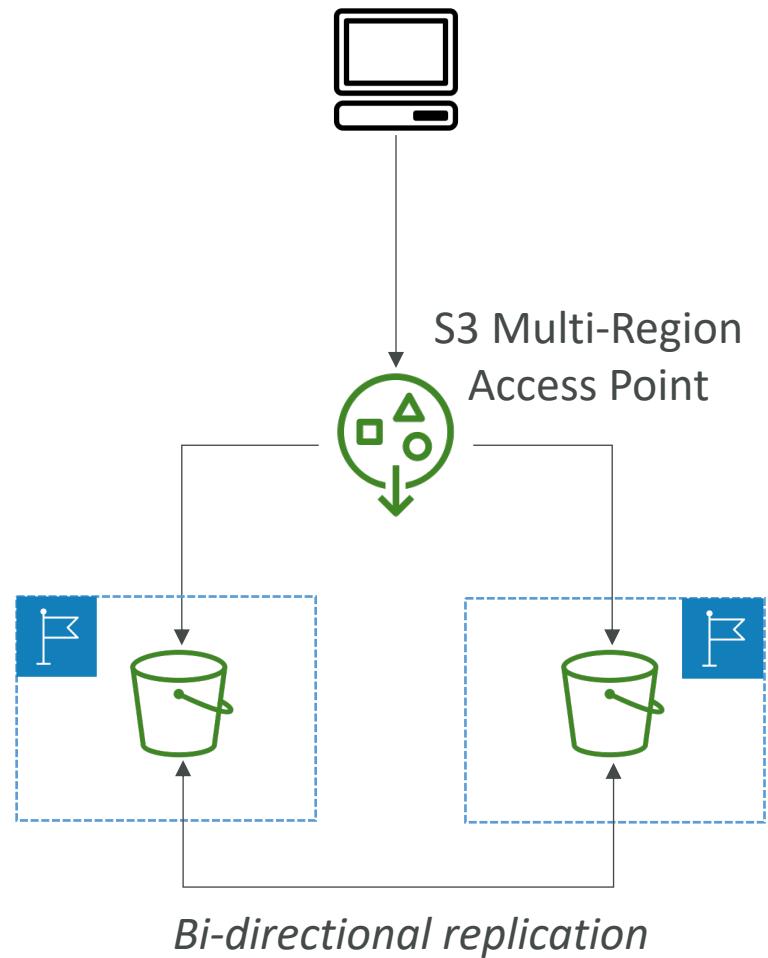
# S3 – Access Points – VPC Origin

- We can define the access point to be accessible only from within the VPC
- You must create a VPC Endpoint to access the Access Point (Gateway or Interface Endpoint)
- The VPC Endpoint Policy must allow access to the target bucket and Access Point

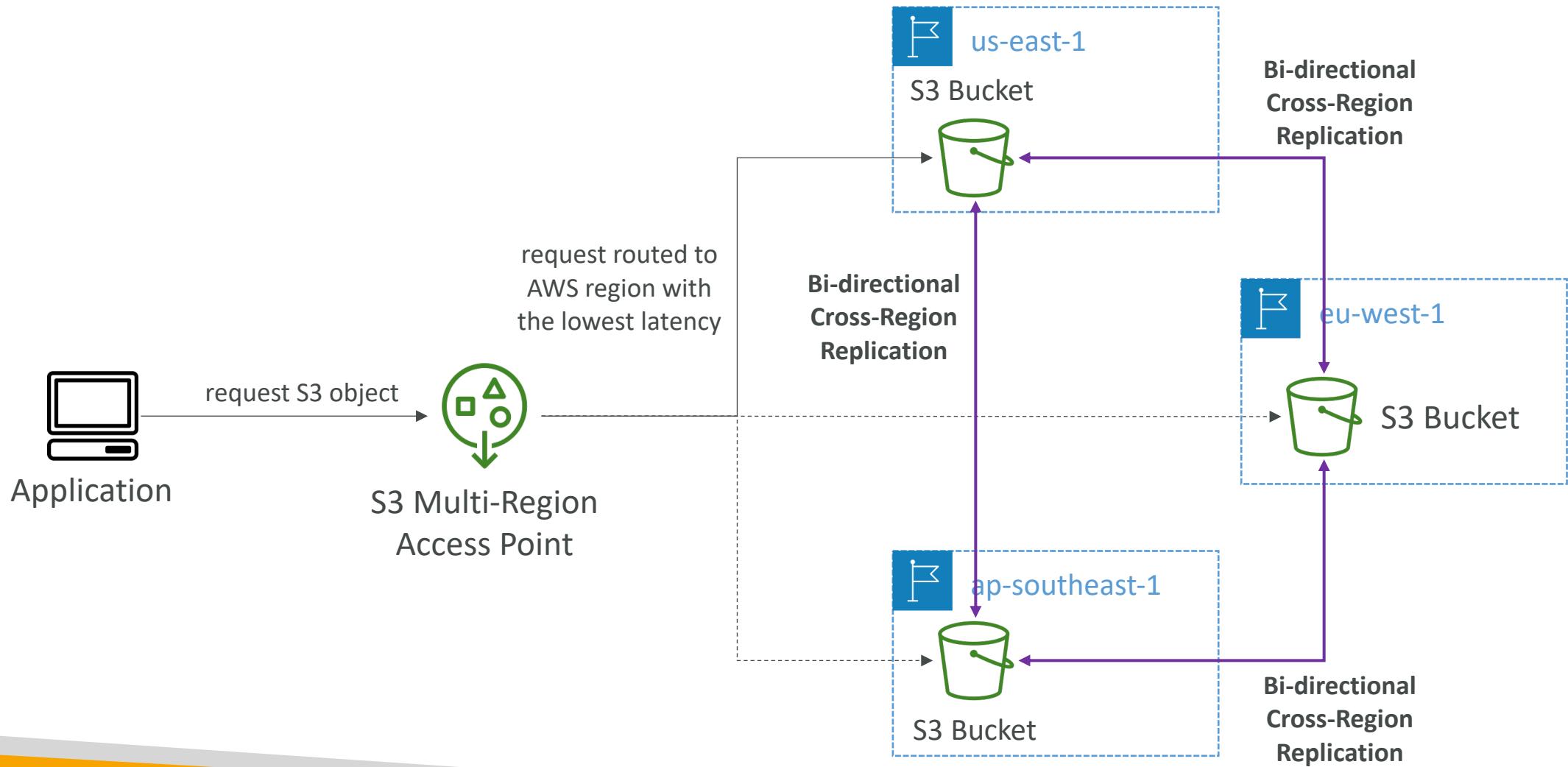


# S3 – Multi-Region Access Points

- Provide a global endpoint that spans S3 buckets in multiple AWS regions
- Dynamically route requests to the nearest S3 bucket (lowest latency)
- Bi-directional S3 bucket replication rules are created to keep data in sync across regions
- **Failover Controls** – allows you to shift requests across S3 buckets in different AWS regions within minutes (Active-Active or Active-Passive)

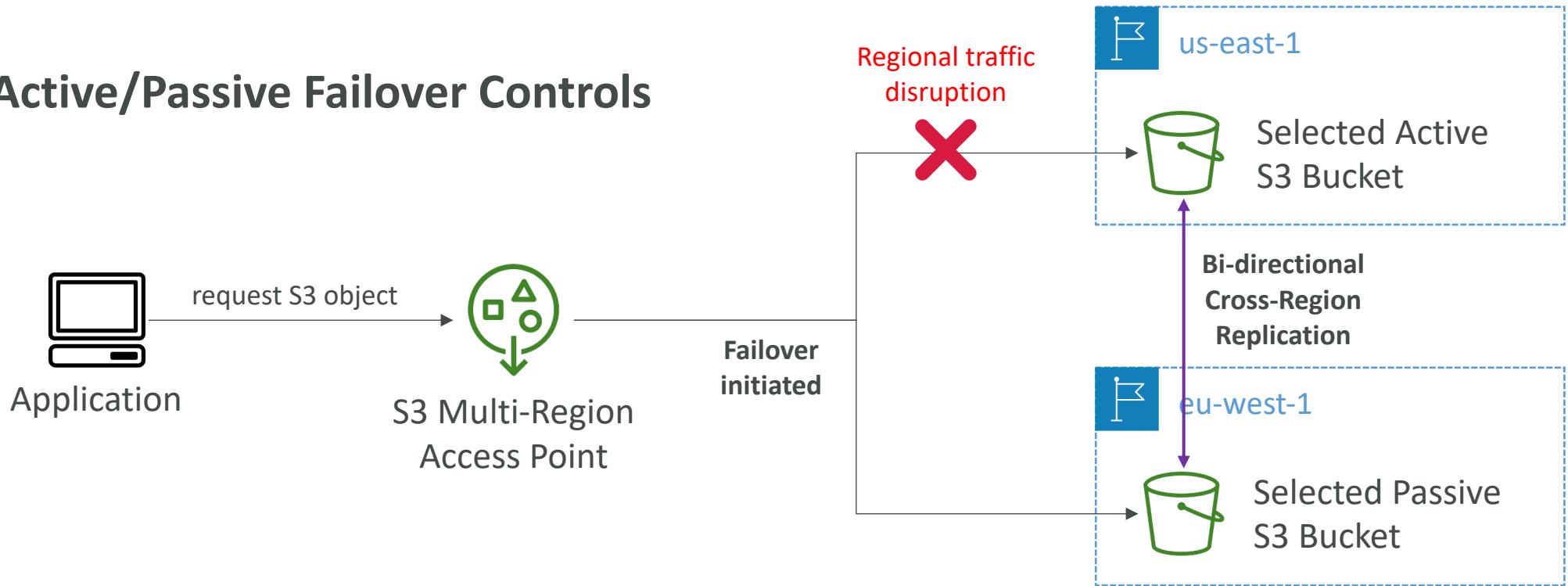


# S3 – Multi-Region Access Points



# Multi-Region Access Points – Failover Controls

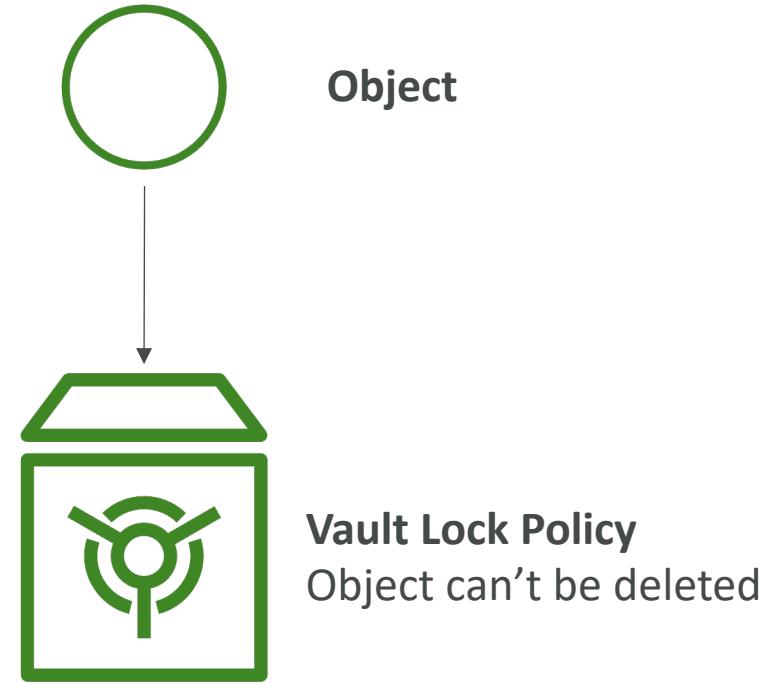
## Active/Passive Failover Controls



*Works with active/active or active/passive setups*

# S3 Glacier Vault Lock

- Adopt a WORM (Write Once Read Many) model
- Create a Vault Lock Policy
- Lock the policy for future edits (can no longer be changed or deleted)
- Helpful for compliance and data retention



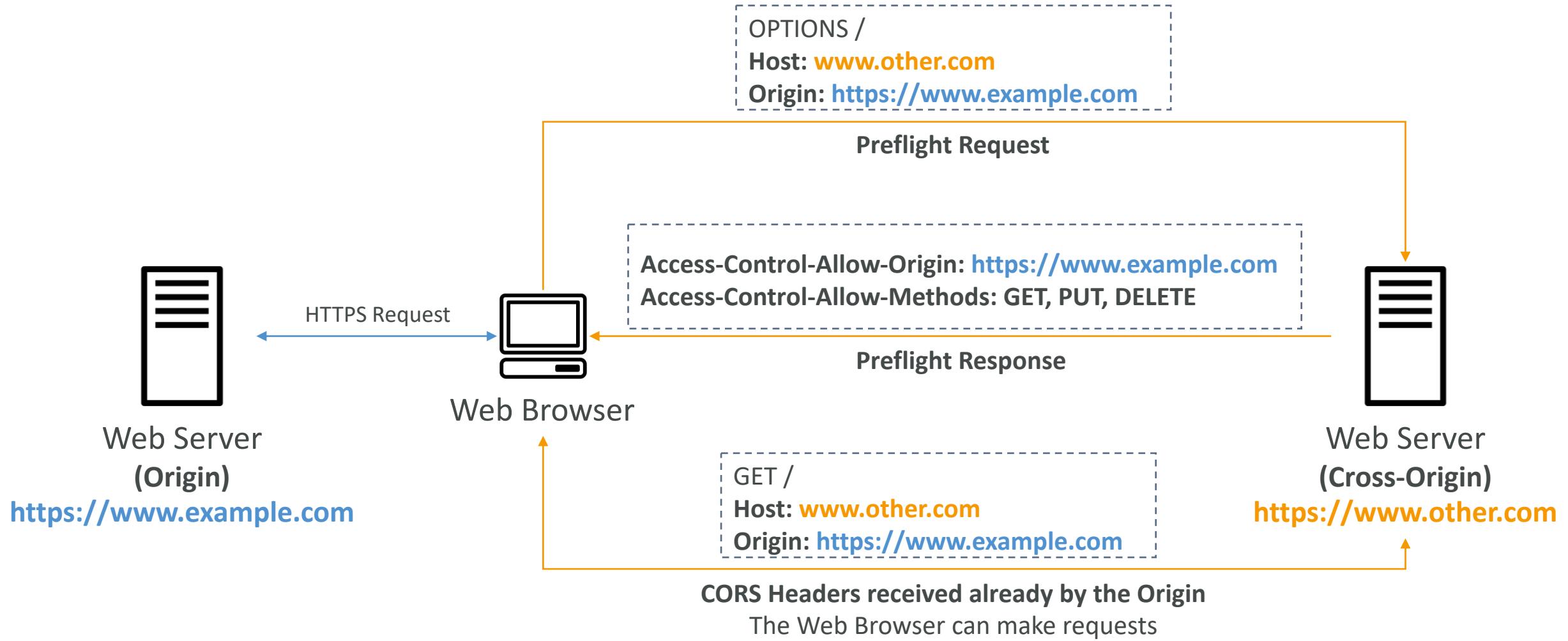
# S3 Object Lock (versioning must be enabled)

- Adopt a WORM (Write Once Read Many) model
- Block an object version deletion for a specified amount of time
- **Retention mode - Compliance:**
  - Object versions can't be overwritten or deleted by any user, including the root user
  - Objects retention modes can't be changed, and retention periods can't be shortened
- **Retention mode - Governance:**
  - Most users can't overwrite or delete an object version or alter its lock settings
  - Some users have special permissions to change the retention or delete the object
- **Retention Period:** protect the object for a fixed period, it can be extended
- **Legal Hold:**
  - protect the object indefinitely, independent from retention period
  - can be freely placed and removed using the `s3:PutObjectLegalHold` IAM permission

# What is CORS?

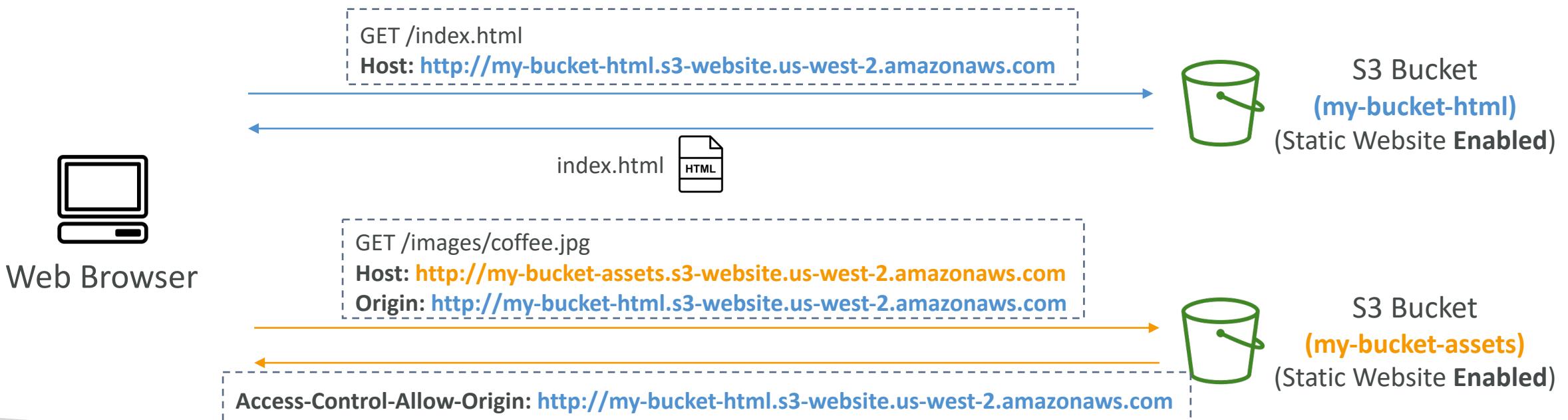
- Cross-Origin Resource Sharing (CORS)
- Origin = scheme (protocol) + host (domain) + port
  - example: <https://www.example.com> (implied port is 443 for HTTPS, 80 for HTTP)
- Web Browser based mechanism to allow requests to other origins while visiting the main origin
- Same origin: <http://example.com/app1> & <http://example.com/app2>
- Different origins: <http://www.example.com> & <http://other.example.com>
- The requests won't be fulfilled unless the other origin allows for the requests, using CORS Headers (example: Access-Control-Allow-Origin)

# What is CORS?



# Amazon S3 – CORS

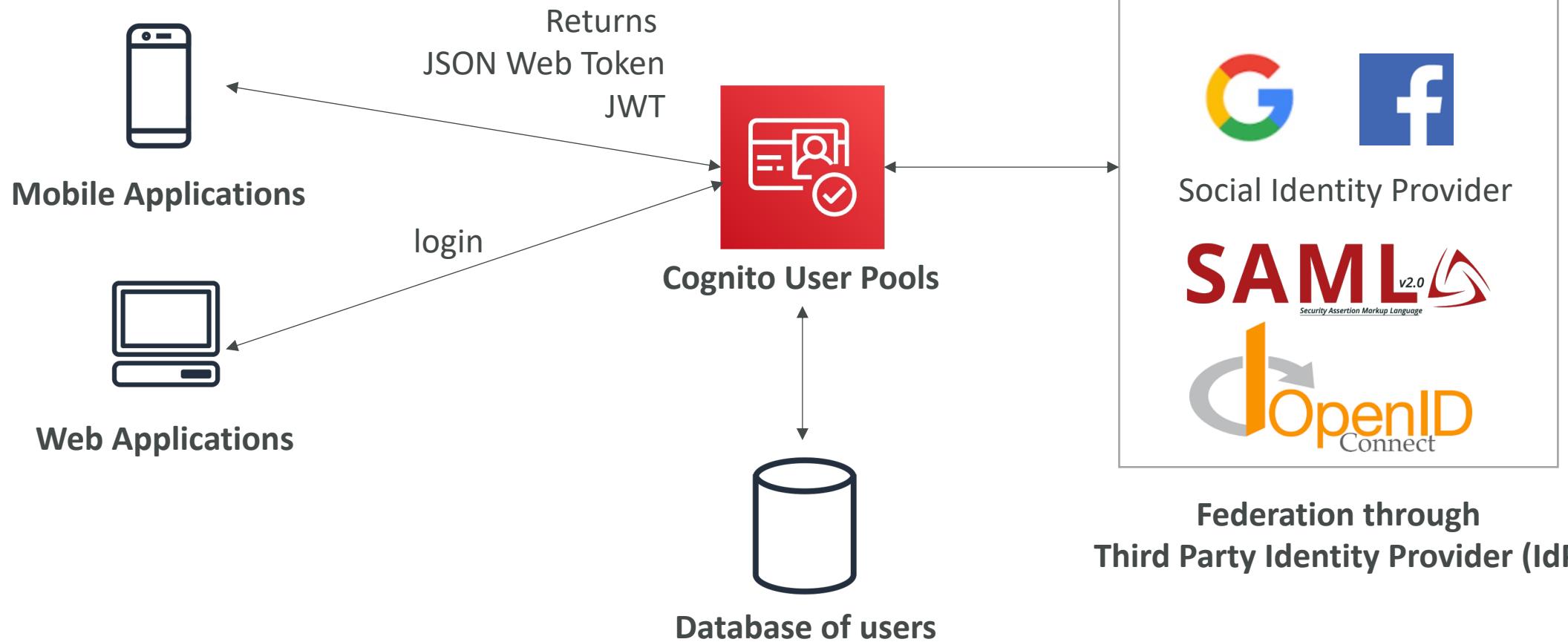
- If a client makes a cross-origin request on our S3 bucket, we need to enable the correct CORS headers
- It's a popular exam question
- You can allow for a specific origin or for \* (all origins)



# Cognito User Pools (CUP) – User Features

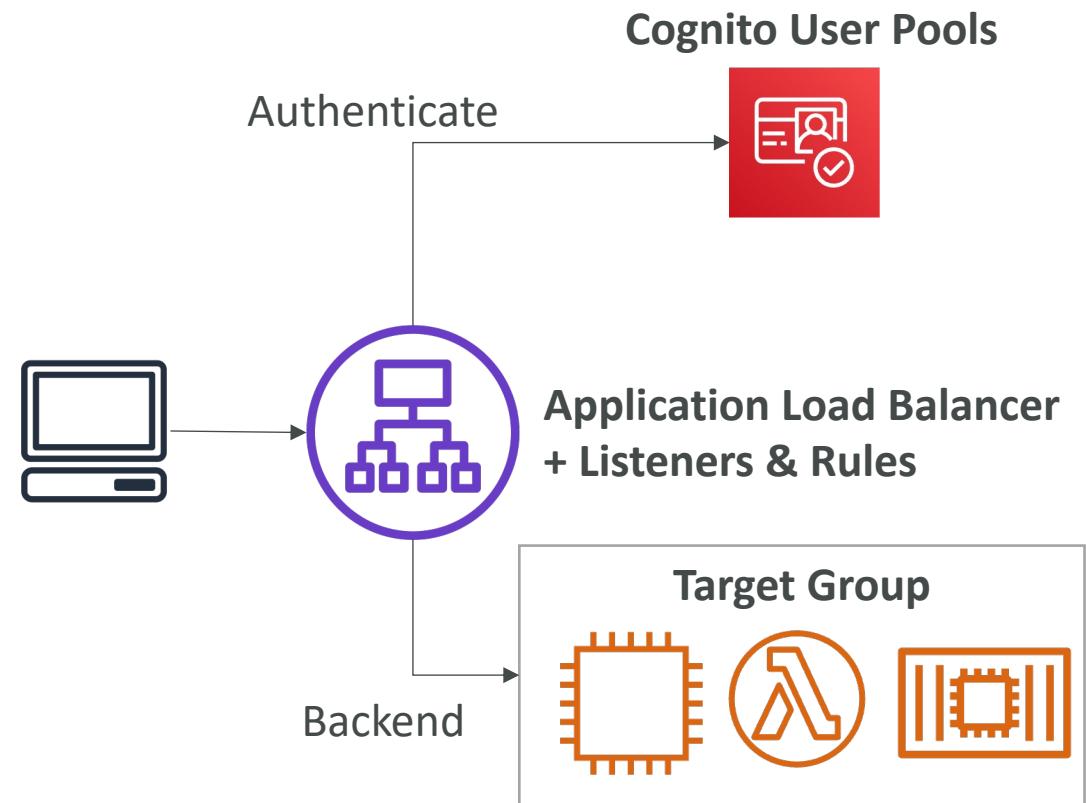
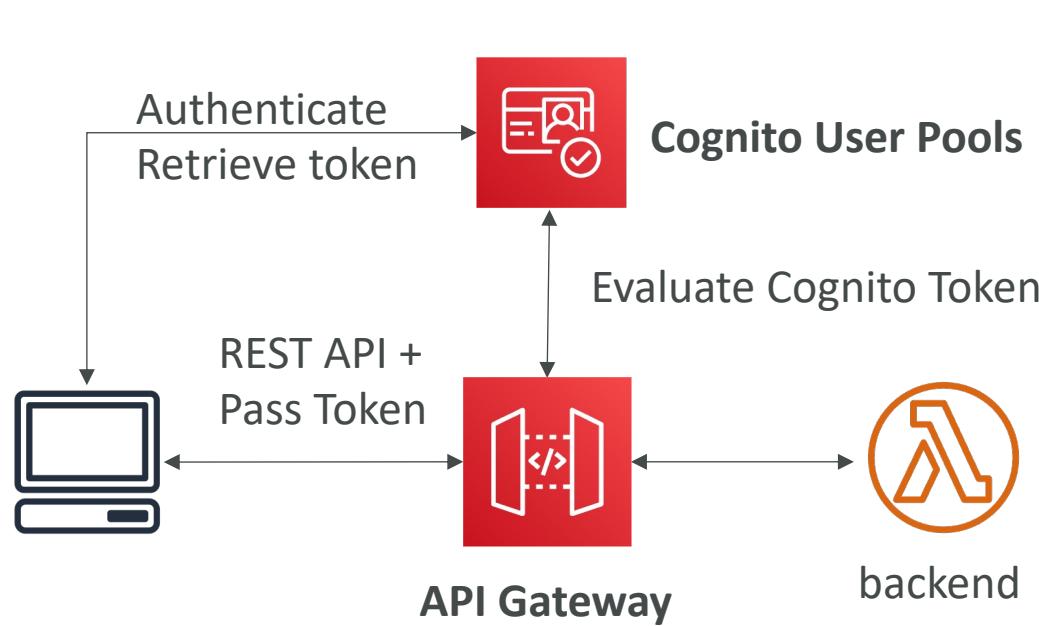
- Create a serverless database of user for your web & mobile apps
- Simple login: Username (or email) / password combination
- Password reset
- Email & Phone Number Verification
- Multi-factor authentication (MFA)
- Federated Identities: users from Facebook, Google, SAML...
- Feature: block users if their credentials are compromised elsewhere
- Login sends back a JSON Web Token (JWT)

# Cognito User Pools (CUP) – Diagram



# Cognito User Pools (CUP) - Integrations

- CUP integrates with API Gateway and Application Load Balancer



# Cognito User Pools – Lambda Triggers

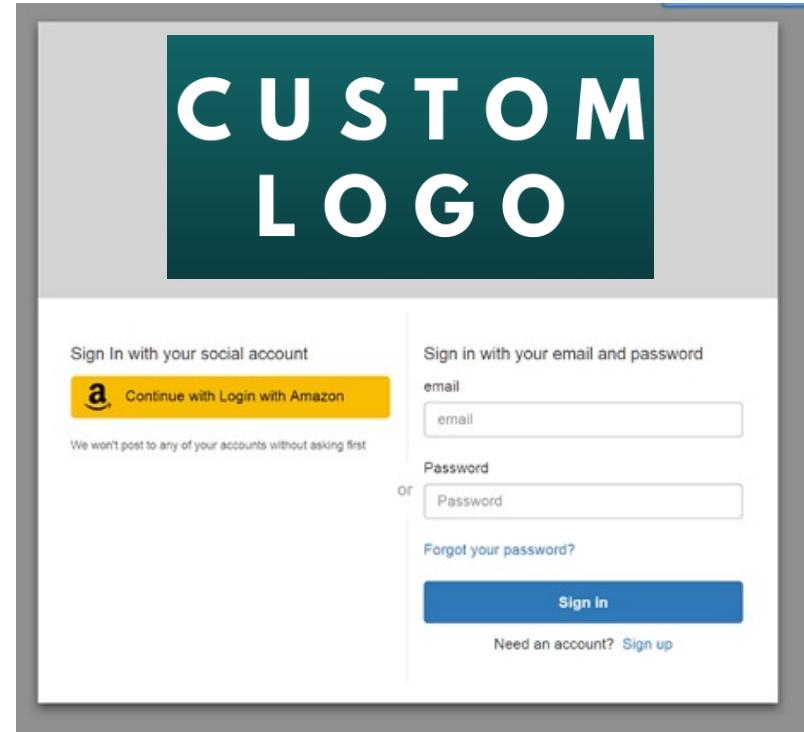
- CUP can invoke a Lambda function synchronously on these triggers:

User Pool Flow	Operation	Description
Authentication Events	Pre Authentication Lambda Trigger	Custom validation to accept or deny the sign-in request
	Post Authentication Lambda Trigger	Event logging for custom analytics
	Pre Token Generation Lambda Trigger	Augment or suppress token claims
Sign-Up	Pre Sign-up Lambda Trigger	Custom validation to accept or deny the sign-up request
	Post Confirmation Lambda Trigger	Custom welcome messages or event logging for custom analytics
	Migrate User Lambda Trigger	Migrate a user from an existing user directory to user pools
Messages	Custom Message Lambda Trigger	Advanced customization and localization of messages
Token Creation	Pre Token Generation Lambda Trigger	Add or remove attributes in Id tokens

<https://docs.aws.amazon.com/cognito/latest/developerguide/cognito-user-identity-pools-working-with-aws-lambda-triggers.html>

# Cognito User Pools – Hosted Authentication UI

- Cognito has a hosted authentication UI that you can add to your app to handle sign-up and sign-in workflows
- Using the hosted UI, you have a foundation for integration with social logins, OIDC or SAML
- Can customize with a custom logo and custom CSS

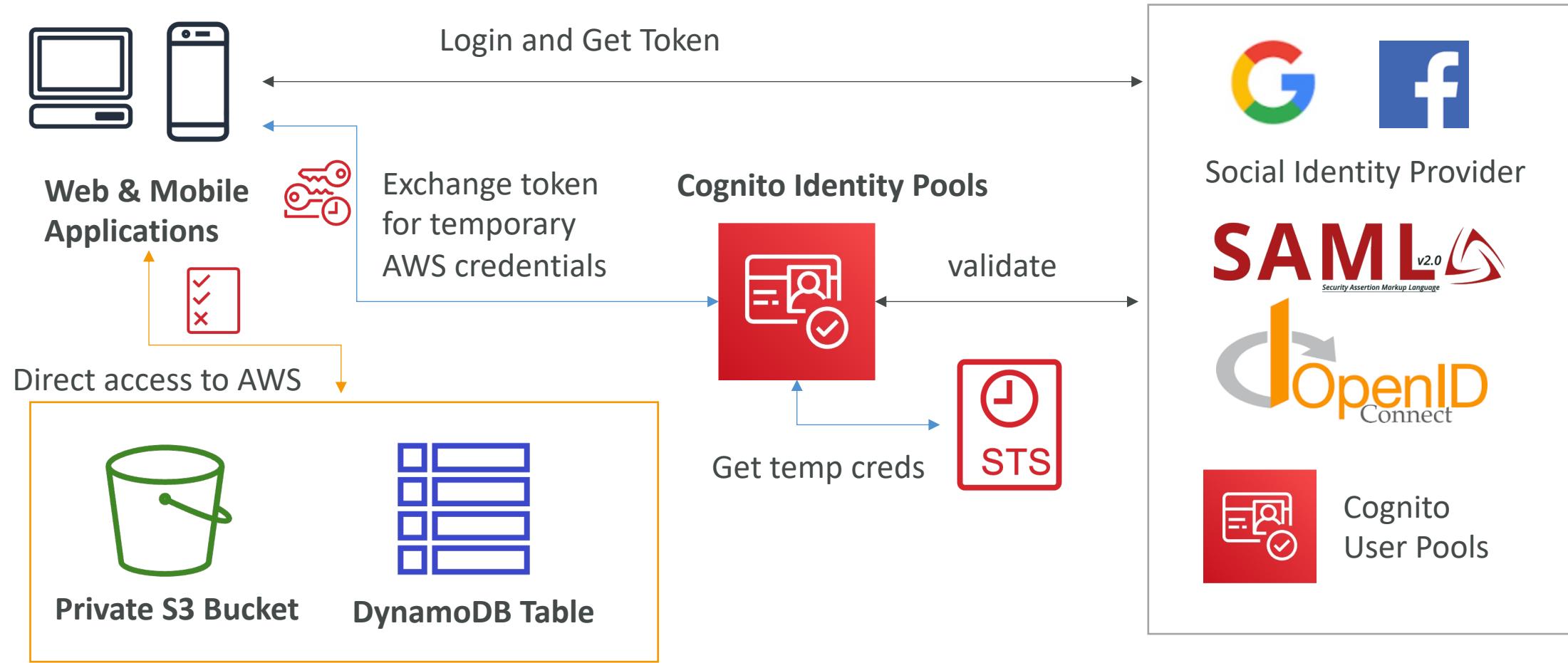


<https://aws.amazon.com/blogs/aws/launch-amazon-cognito-user-pools-general-availability-app-integration-and-federation/>

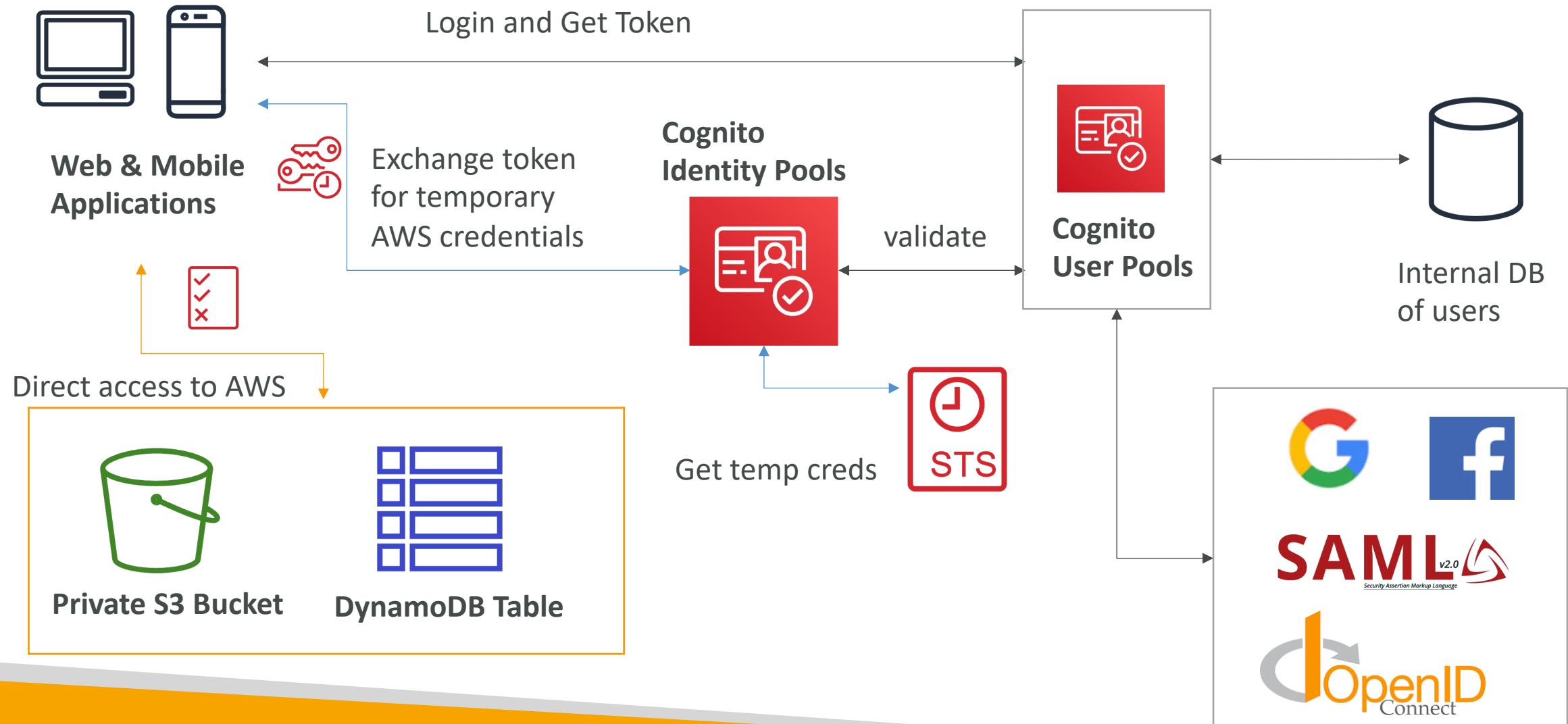
# Cognito Identity Pools (Federated Identities)

- Get identities for “users” so they obtain temporary AWS credentials
- Your identity pool (e.g identity source) can include:
  - Public Providers (Login with Amazon, Facebook, Google, Apple)
  - Users in an Amazon Cognito user pool
  - OpenID Connect Providers & SAML Identity Providers
  - Developer Authenticated Identities (custom login server)
  - Cognito Identity Pools allow for unauthenticated (guest) access
- Users can then access AWS services directly or through API Gateway
  - The IAM policies applied to the credentials are defined in Cognito
  - They can be customized based on the user\_id for fine grained control

# Cognito Identity Pools – Diagram



# Cognito Identity Pools – Diagram with CUP



# Cognito Identity Pools – IAM Roles

- Default IAM roles for authenticated and guest users
  - Define rules to choose the role for each user based on the user's ID
  - You can partition your users' access using **policy variables**
- 
- IAM credentials are obtained by Cognito Identity Pools through STS
  - The roles must have a “trust” policy of Cognito Identity Pools

# Cognito Identity Pools – Guest User example

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Action": [  
                "s3:GetObject"  
            ],  
            "Effect": "Allow",  
            "Resource": [  
                "arn:aws:s3:::mybucket/assets/my_picture.jpg"  
            ]  
        }  
    ]  
}
```

# Cognito Identity Pools – Policy variable on S3

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

# Cognito Identity Pools – DynamoDB

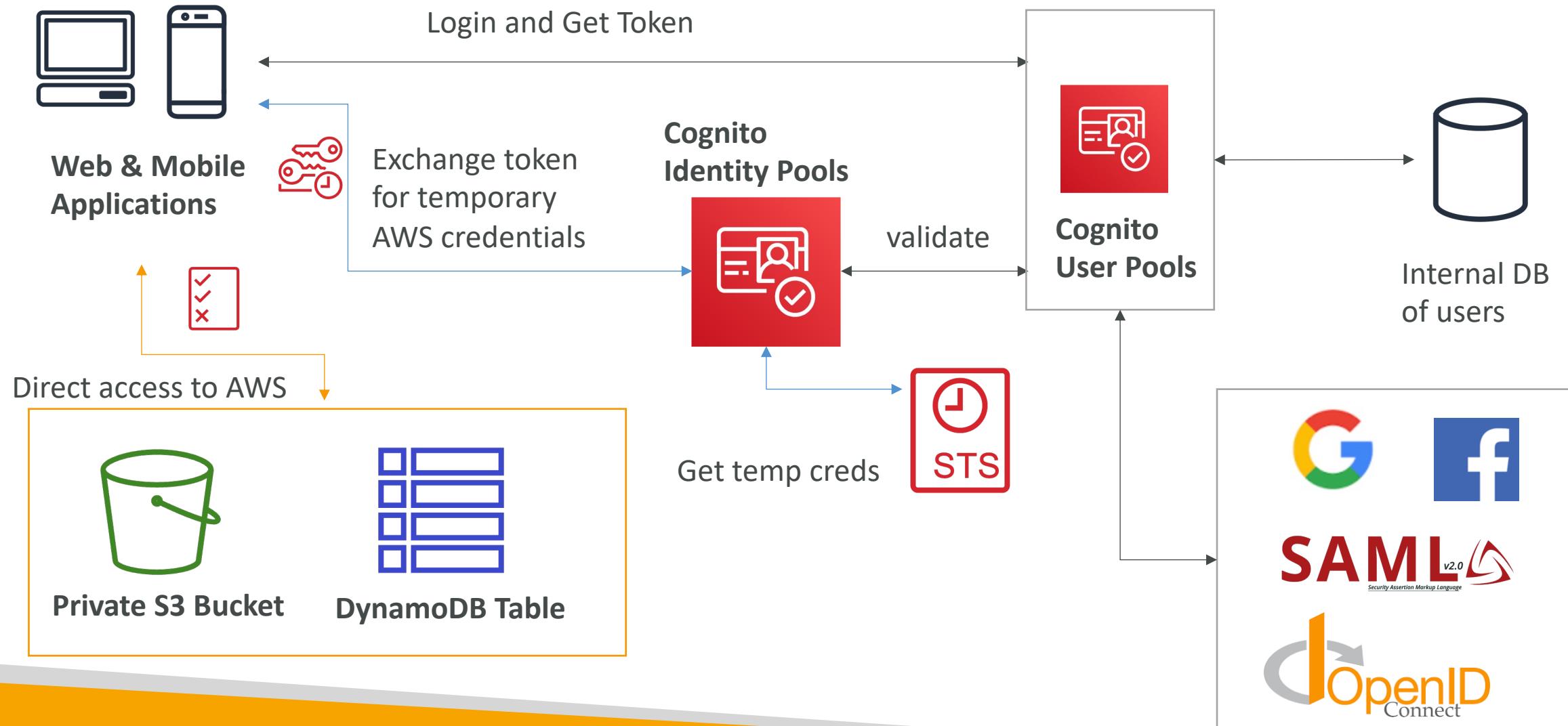
```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "dynamodb:GetItem", "dynamodb:BatchGetItem", "dynamodb:Query",  
                "dynamodb:PutItem", "dynamodb:UpdateItem", "dynamodb:DeleteItem",  
                "dynamodb:BatchWriteItem"  
            ],  
            "Resource": [  
                "arn:aws:dynamodb:us-west-2:123456789012:table/MyTable"  
            ],  
            "Condition": {  
                "ForAllValues:StringEquals": {  
                    "dynamodb:LeadingKeys": [  
                        "${cognito-identity.amazonaws.com:sub}"  
                    ]  
                }  
            }  
        }  
    ]  
}
```



# Cognito User Pools vs Identity Pools

- **Cognito User Pools:**
  - Database of users for your web and mobile application
  - Allows to federate logins through Public Social, OIDC, SAML...
  - Can customize the hosted UI for authentication (including the logo)]
  - Has triggers with AWS Lambda during the authentication flow
- **Cognito Identity Pools:**
  - Obtain AWS credentials for your users
  - Users can login through Public Social, OIDC, SAML & Cognito User Pools
  - Users can be unauthenticated (guests)
  - Users are mapped to IAM roles & policies, can leverage policy variables
- **CUP + CIP = manage user / password + access AWS services**

# Cognito Identity Pools – Diagram with CUP

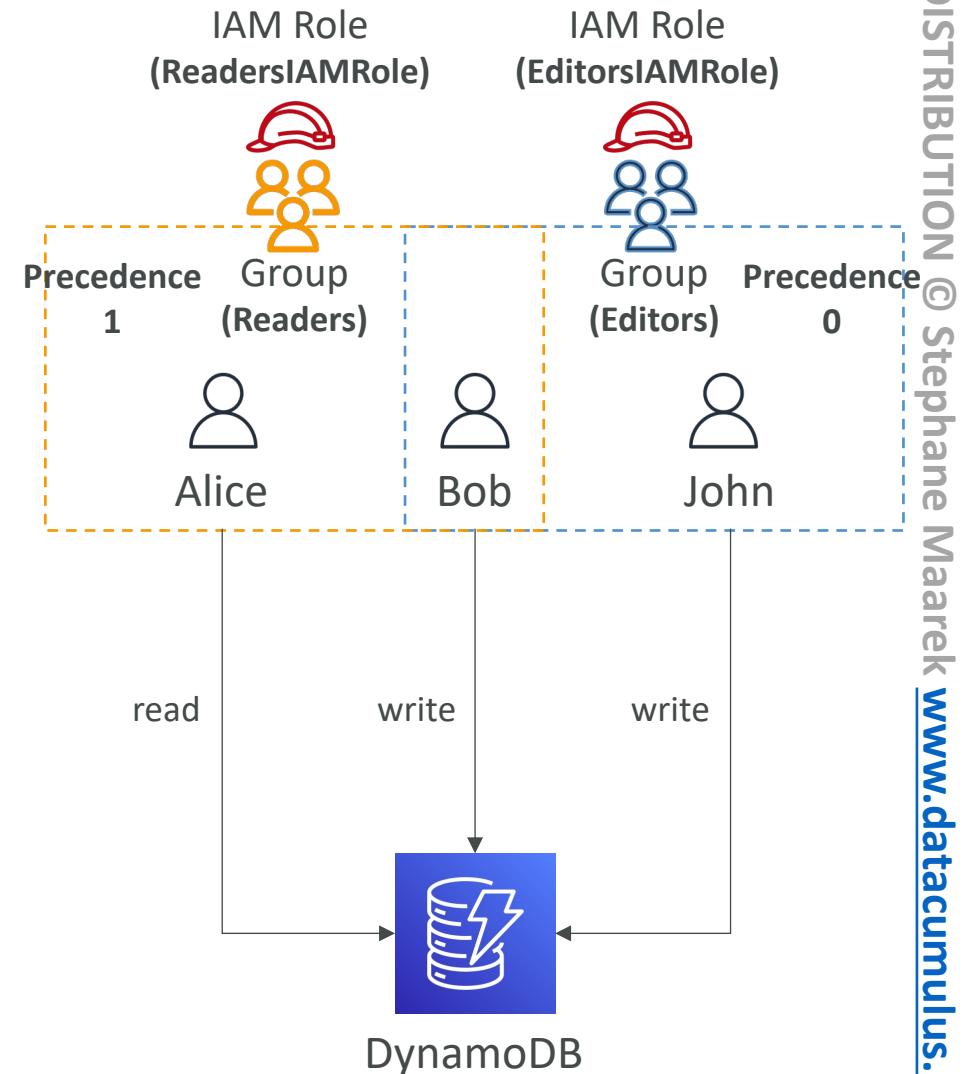


# Cognito Sync

- Deprecated – use AWS AppSync now
- Store preferences, configuration, state of app
- Cross device synchronization (any platform – iOS, Android, etc...)
- Offline capability (synchronization when back online)
- Store data in datasets (up to 1MB), up to 20 datasets to synchronize
- **Push Sync:** silently notify across all devices when identity data changes
- **Cognito Stream:** stream data from Cognito into Kinesis
- **Cognito Events:** execute Lambda functions in response to events

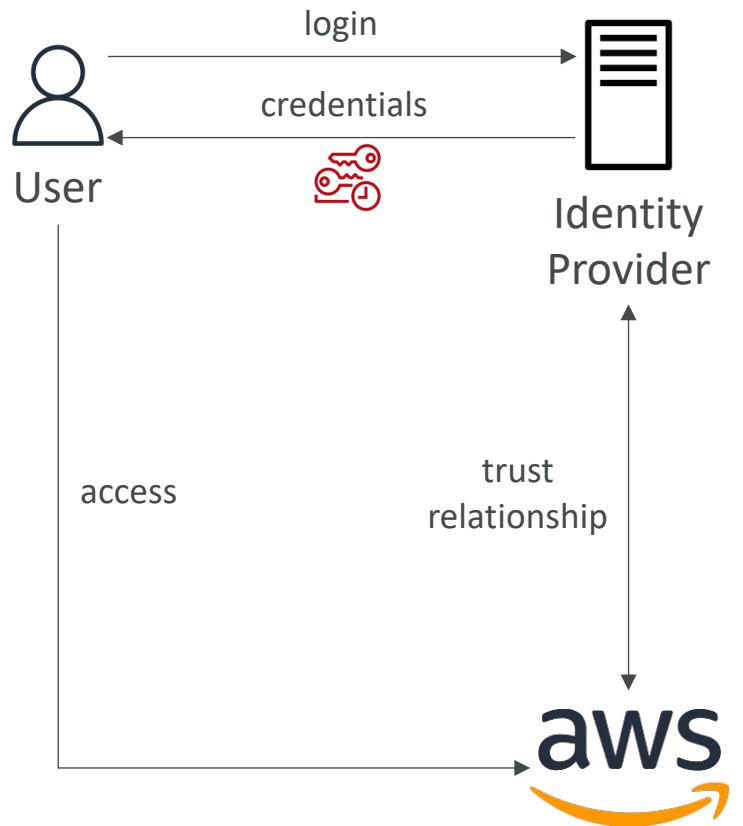
# Cognito User Pool Groups

- Collection of users in a logical group in Cognito User Pool
- Defines the permissions for users in the group by assigning IAM role to the group
- Users can be in multiple groups:
  - Assign precedence values to each group (lower will be chosen and its IAM role will be applied)
  - Choose from available IAM roles by specifying the IAM role ARN
- You can't create nested groups



# Identity Federation in AWS

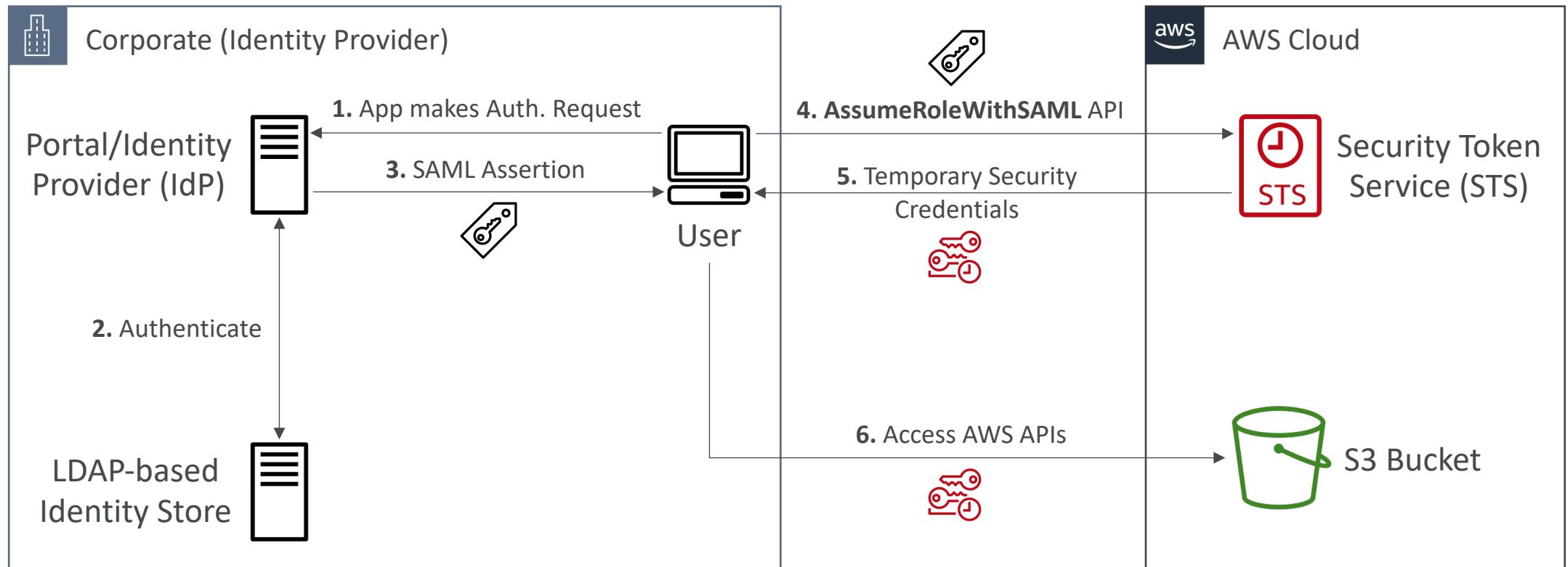
- Give users outside of AWS permissions to access AWS resources in your account
- You don't need to create IAM Users (user management is outside AWS)
- Use cases:
  - A corporate has its own identity system (e.g., Active Directory)
  - Web/Mobile application that needs access to AWS resources
- Identity Federation can have many flavors:
  - SAML 2.0
  - Custom Identity Broker
  - Web Identity Federation With(out) Amazon Cognito
  - Single Sign-On (SSO)



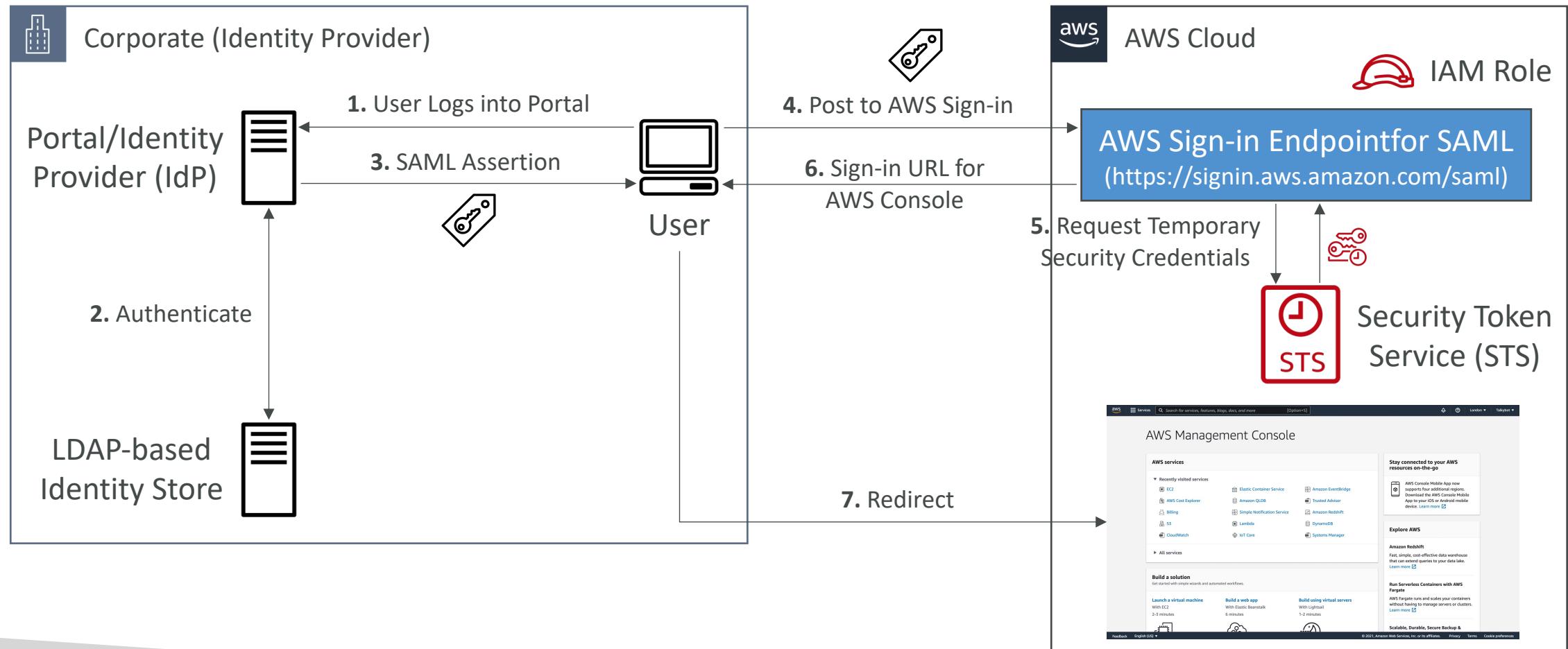
# SAML 2.0 Federation

- Security Assertion Markup Language 2.0 (SAML 2.0)
- Open standard used by many identity providers (e.g., ADFS)
  - Supports integration with Microsoft Active Directory Federations Services (ADFS)
  - Or any SAML 2.0–compatible IdPs with AWS
- Access to AWS Console, AWS CLI, or AWS API using temporary credentials
  - No need to create IAM Users for each of your employees
  - Need to setup a trust between AWS IAM and SAML 2.0 Identity Provider (both ways)
- Under-the-hood: Uses the STS API `AssumeRoleWithSAML`
- SAML 2.0 Federation is the “old way”, Amazon Single Sign-On (AWS SSO) Federation is the new managed and simpler way
  - <https://aws.amazon.com/blogs/security/enabling-federation-to-aws-using-windows-active-directory-adfs-and-saml-2-0/>

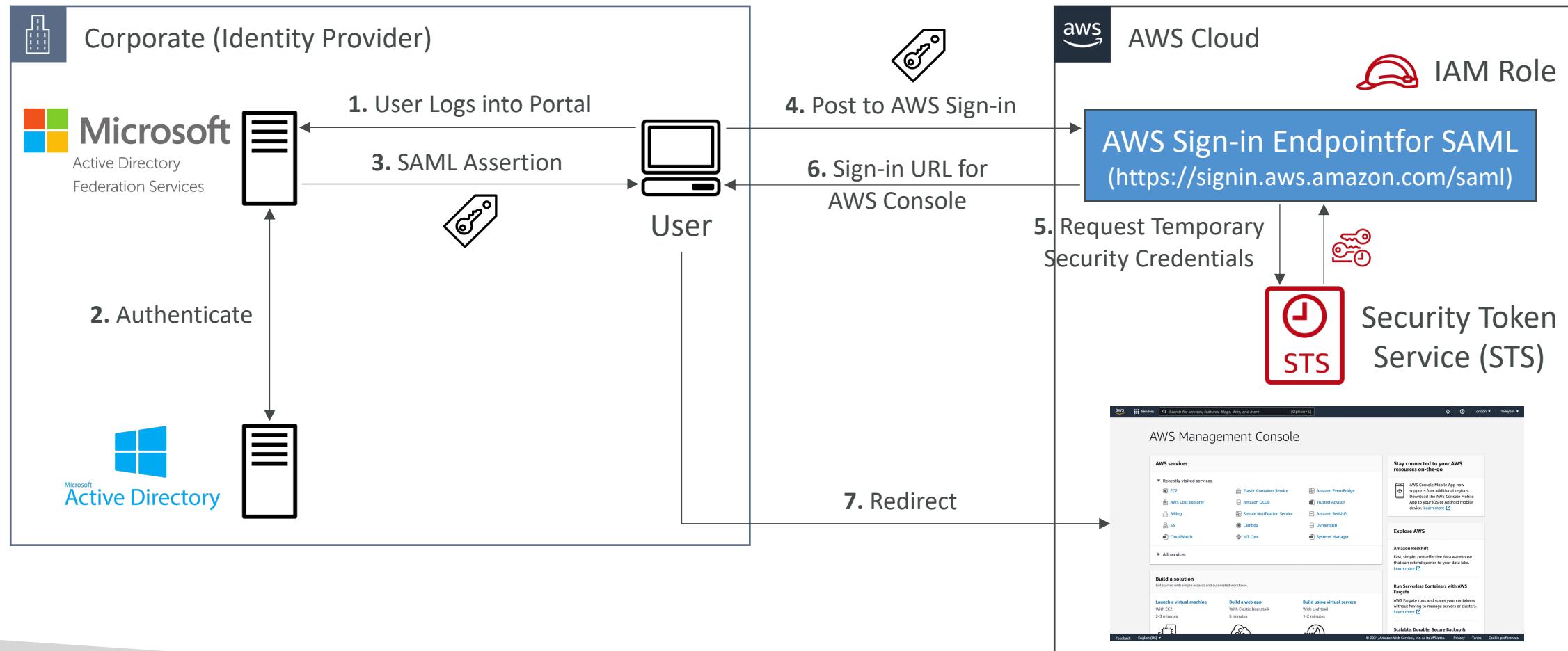
# SAML 2.0 Federation – AWS API Access



# SAML 2.0 Federation – AWS Console Access

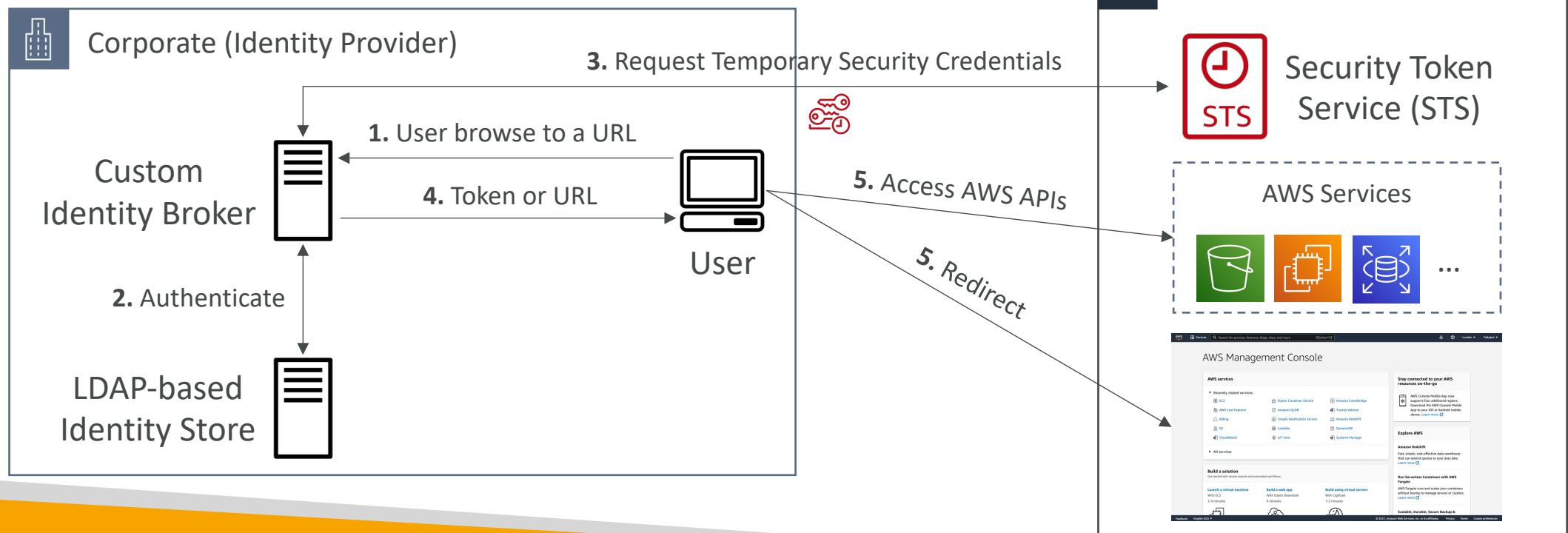


# SAML 2.0 Federation – Active Directory FS (ADFS)



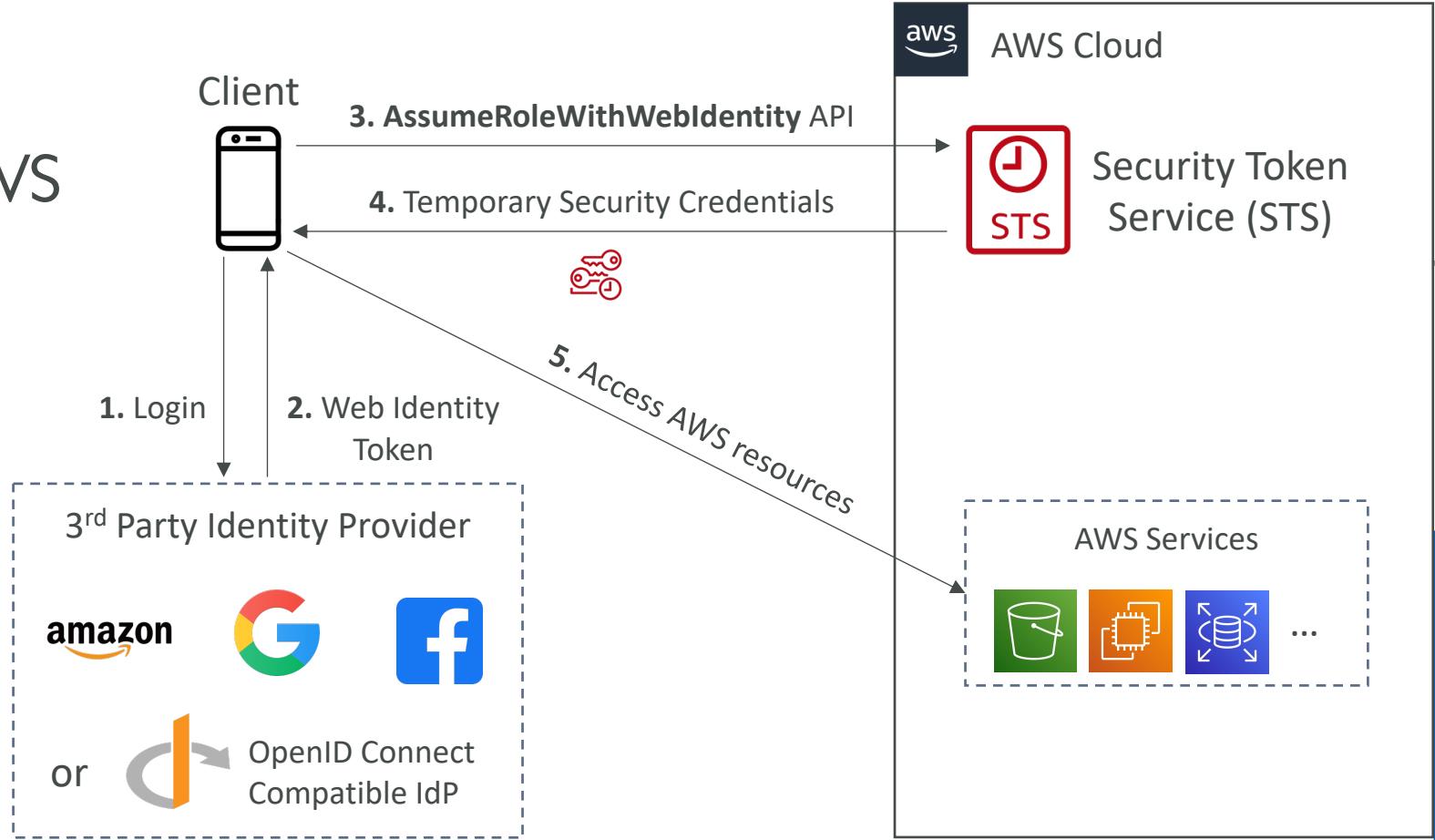
# Custom Identity Broker Application

- Use only if Identity Provider is **NOT** compatible with SAML 2.0
- The Identity Broker Authenticates users & requests temporary credentials from AWS
- The Identity Broker must determine the appropriate IAM Role
- Uses the STS API AssumeRole or GetFederationToken



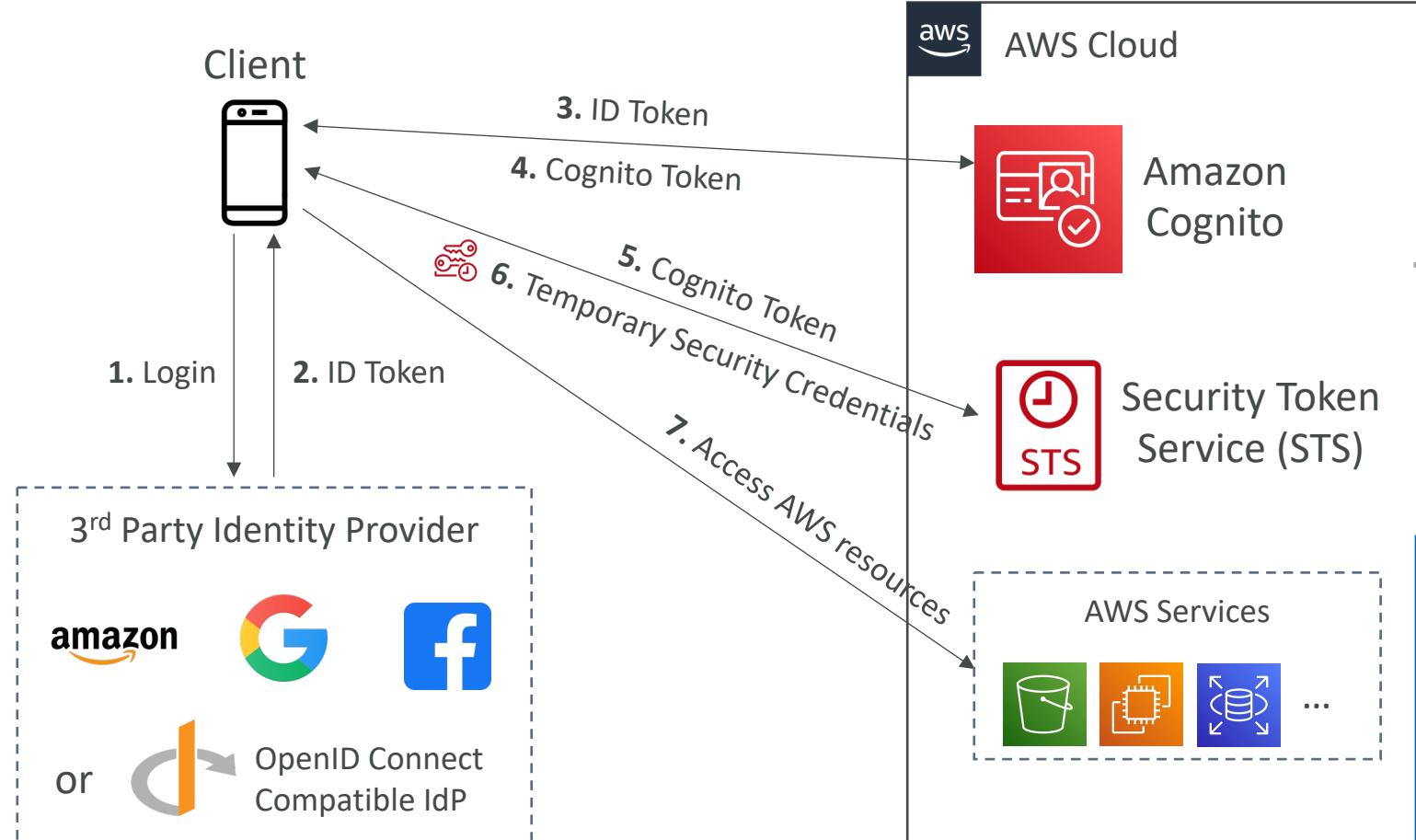
# Web Identity Federation – Without Cognito

- Not recommended by AWS  
– use Cognito instead



# Web Identity Federation – With Cognito

- Preferred over Web Identity Federation
  - Create IAM Roles using Cognito with the least privilege needed
  - Build trust between the OIDC IdP and AWS
- Cognito benefits:
  - Supports anonymous users
  - Supports MFA
  - Data Synchronization
- Cognito replaces a Token Vending Machine (TVM)



# Web Identity Federation – IAM Policy

- After being authenticated with Web Identity Federation, you can identify the user with an IAM policy variable

- Examples:

- cognito-identity.amazonaws.com:sub
- www.amazon.com:user\_id
- graph.facebook.com:id
- accounts.google.com:sub

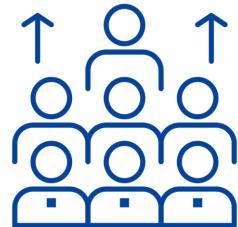
```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": "s3>ListBucket",  
            "Resource": "arn:aws:s3:::myBucket",  
            "Condition": {  
                "StringLike": {  
                    "s3:prefix": "Amazon/mynumbersgame/${www.amazon.com:user_id}/*"  
                }  
            }  
        }, {  
            "Effect": "Allow",  
            "Action": ["s3:GetObject", "s3:PutObject", "s3>DeleteObject"],  
            "Resource": [  
                "arn:aws:s3:::myBucket/Amazon/mynumbersgame/${www.amazon.com:user_id}",  
                "arn:aws:s3:::myBucket/Amazon/mynumbersgame/${www.amazon.com:user_id}/*"  
            ]  
        }  
    ]  
}
```

# AWS IAM Identity Center

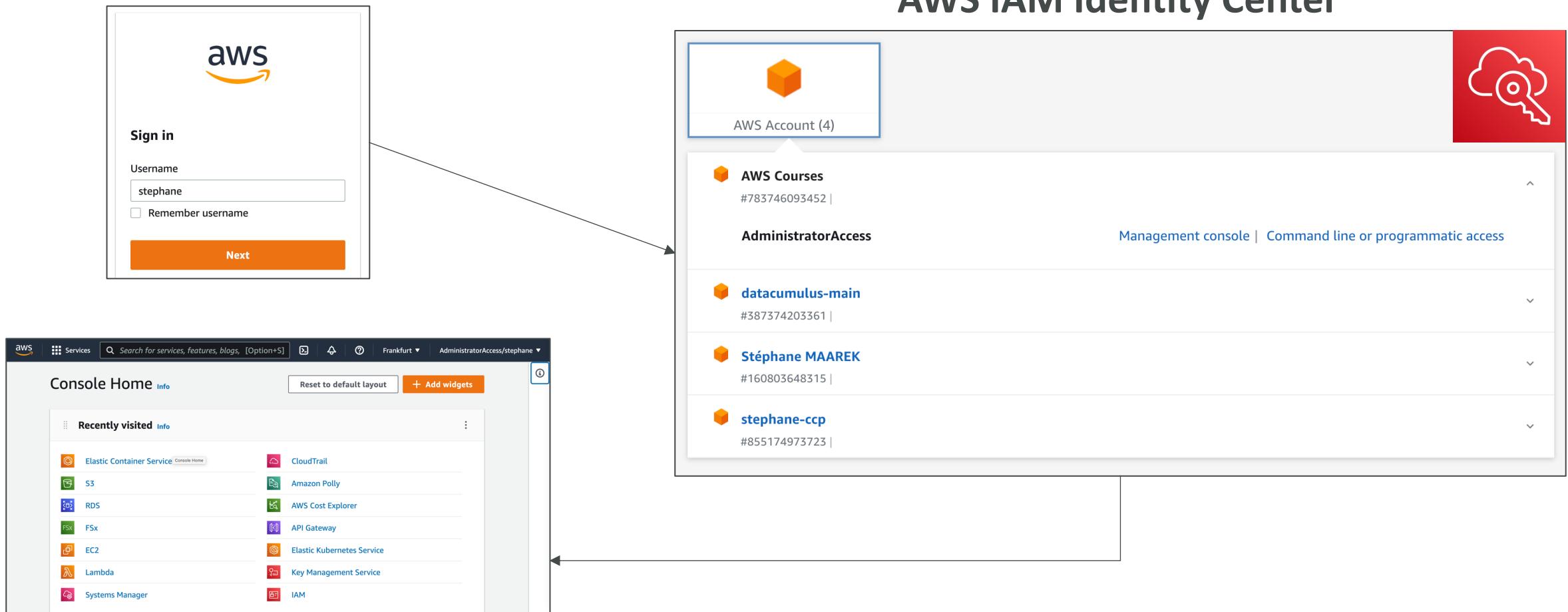
(successor to AWS Single Sign-On)



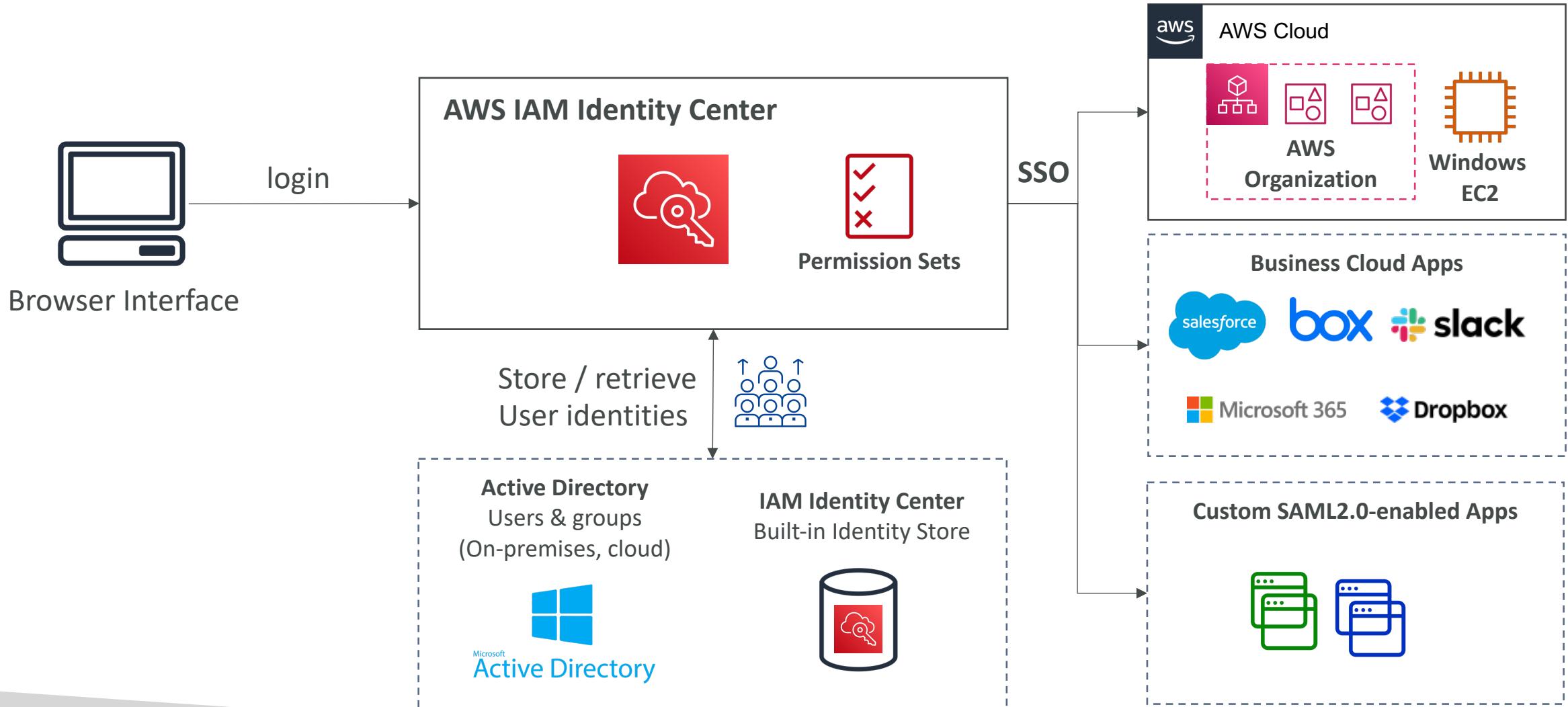
- One login (single sign-on) for all your
  - AWS accounts in AWS Organizations
  - Business cloud applications (e.g., Salesforce, Box, Microsoft 365, ...)
  - SAML2.0-enabled applications
  - EC2 Windows Instances
- Identity providers
  - Built-in identity store in IAM Identity Center
  - 3<sup>rd</sup> party: Active Directory (AD), OneLogin, Okta...



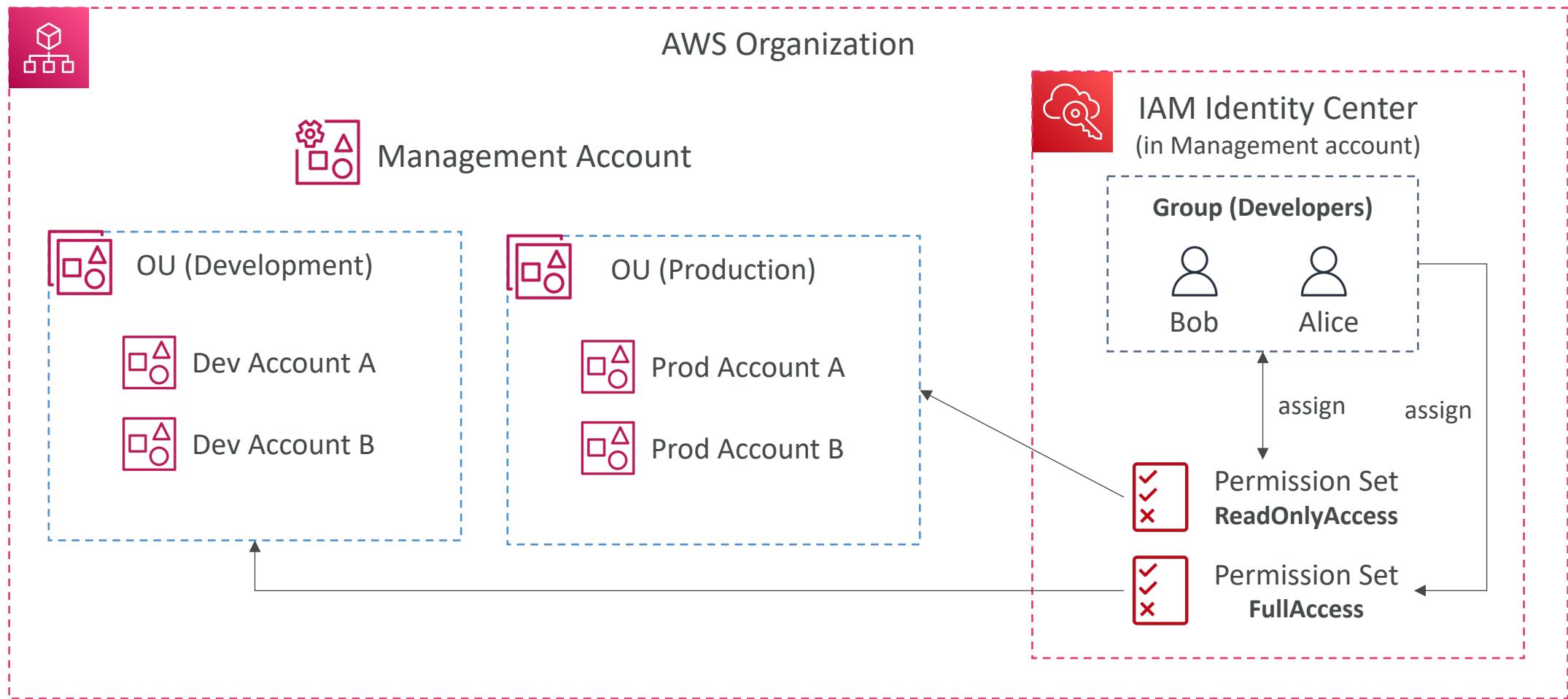
# AWS IAM Identity Center – Login Flow



# AWS IAM Identity Center



# IAM Identity Center

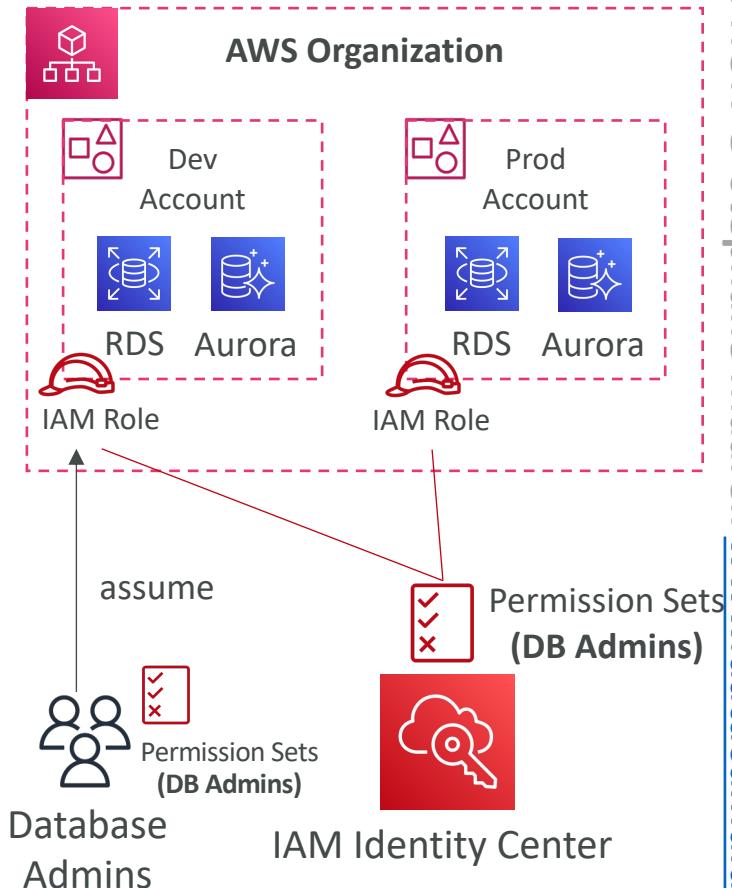


# AWS IAM Identity Center

## Fine-grained Permissions and Assignments

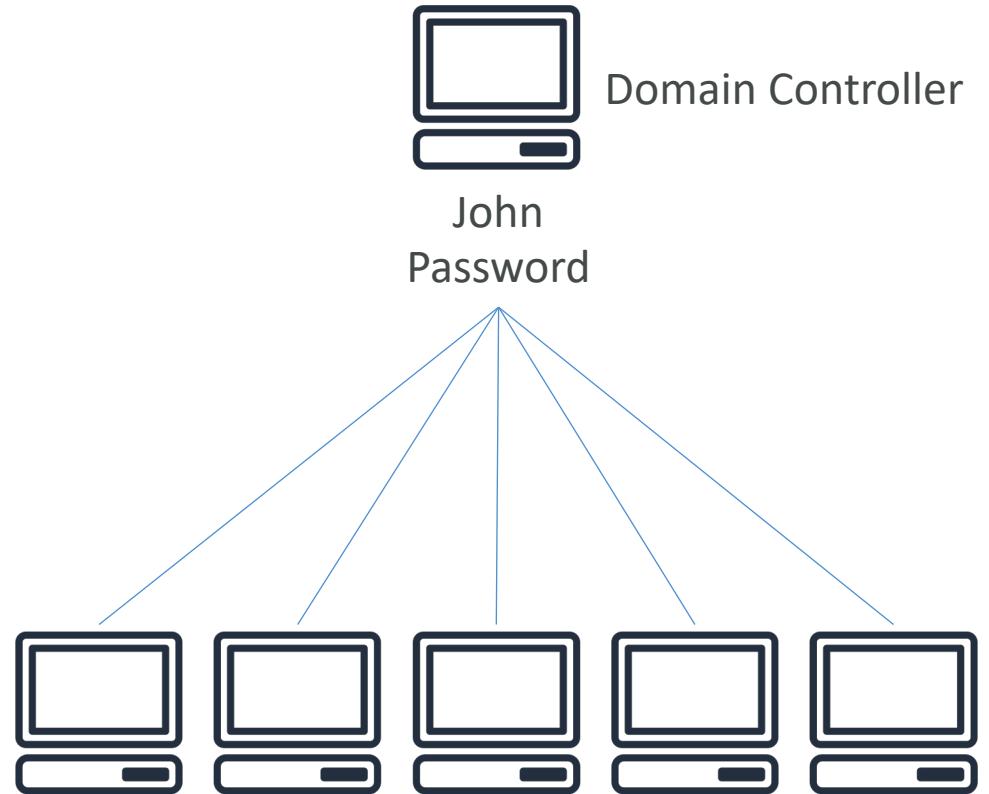


- Multi-Account Permissions
  - Manage access across AWS accounts in your AWS Organization
  - Permission Sets – a collection of one or more IAM Policies assigned to users and groups to define AWS access
- Application Assignments
  - SSO access to many SAML 2.0 business applications (Salesforce, Box, Microsoft 365, ...)
  - Provide required URLs, certificates, and metadata
- Attribute-Based Access Control (ABAC)
  - Fine-grained permissions based on users' attributes stored in IAM Identity Center Identity Store
  - Example: cost center, title, locale, ...
  - Use case: Define permissions once, then modify AWS access by changing the attributes



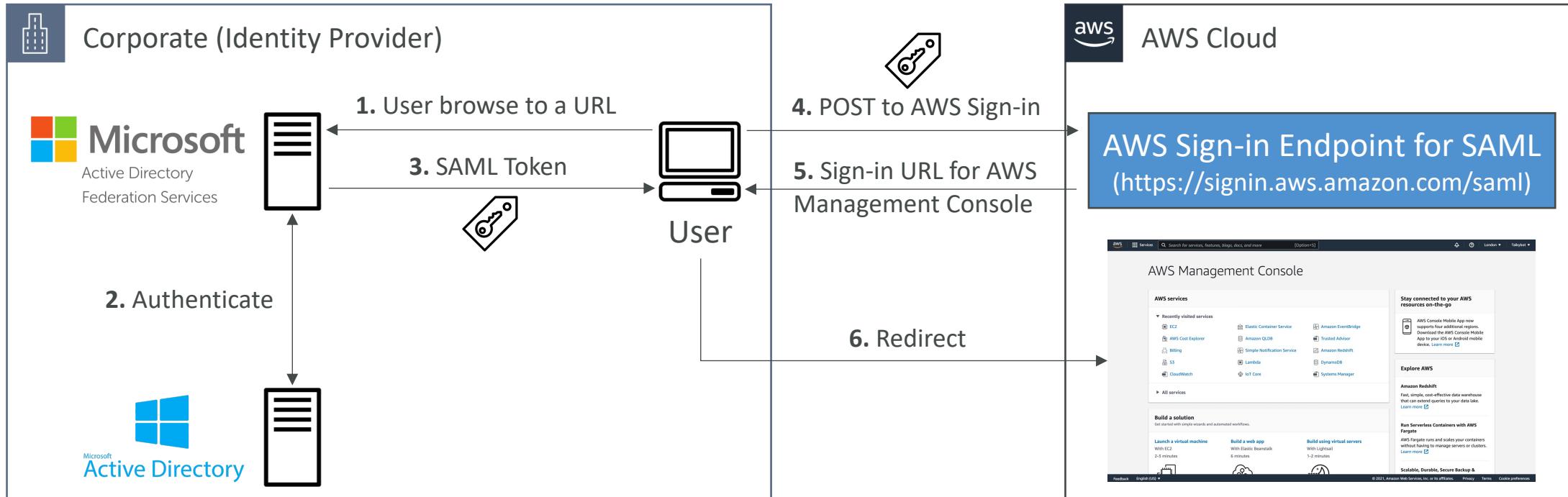
# What is Microsoft Active Directory (AD)?

- Found on any Windows Server with AD Domain Services
- Database of **objects**: User Accounts, Computers, Printers, File Shares, Security Groups
- Centralized security management, create account, assign permissions
- Objects are organized in **trees**
- A group of trees is a **forest**



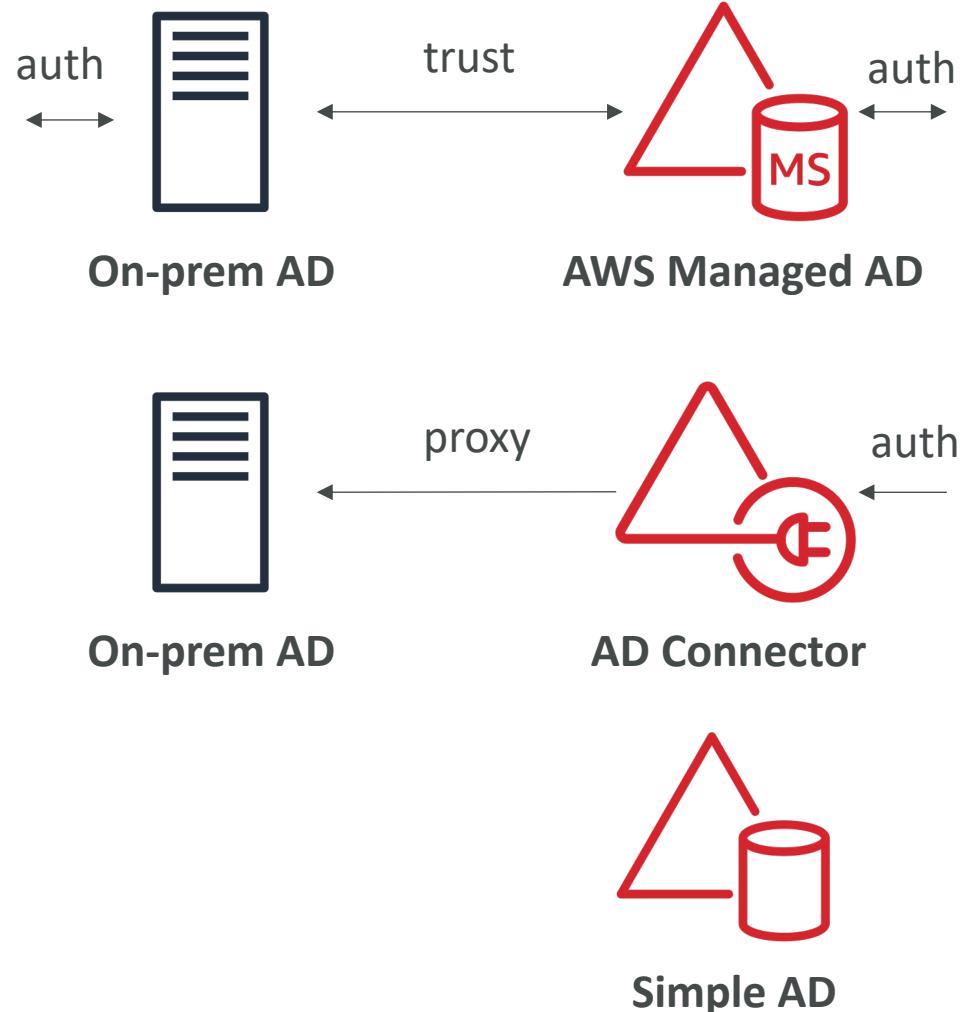
# What is ADFS (AD Federation Services)?

- ADFS provides Single Sign-On across applications
- SAML across 3<sup>rd</sup> party: AWS Console, Dropbox, Office365, etc...



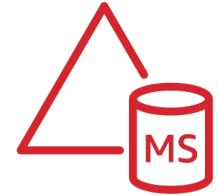
# AWS Directory Services

- AWS Managed Microsoft AD
  - Create your own AD in AWS, manage users locally, supports MFA
  - Establish “trust” connections with your on-premises AD
- AD Connector
  - Directory Gateway (proxy) to redirect to on-premises AD, supports MFA
  - Users are managed on the on-premises AD
- Simple AD
  - AD-compatible managed directory on AWS
  - Cannot be joined with on-premises AD

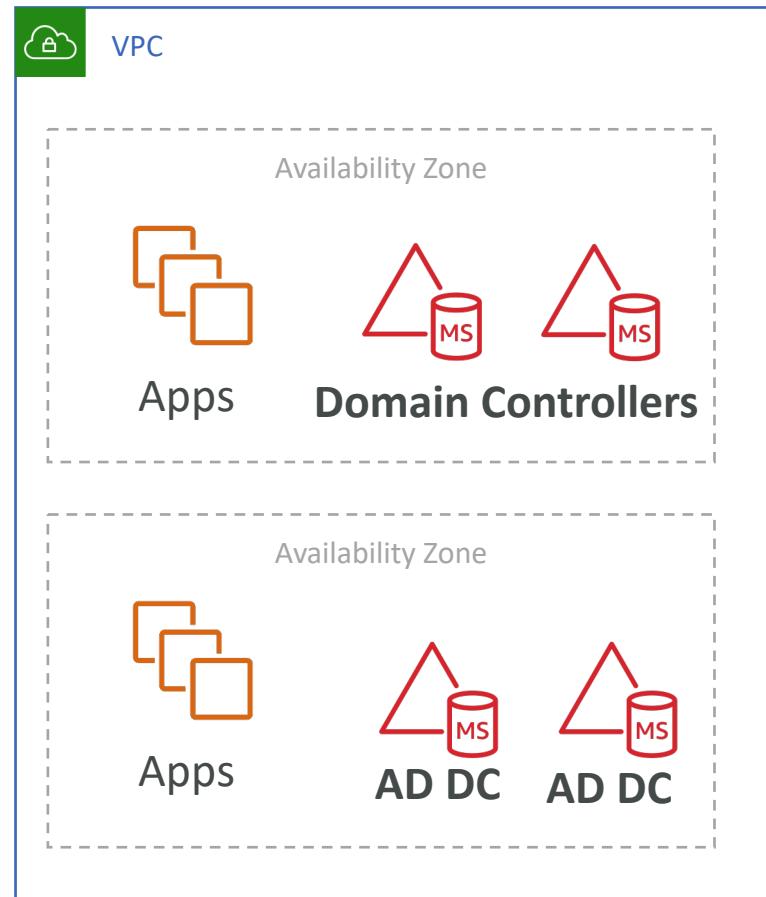


# AWS Directory Services

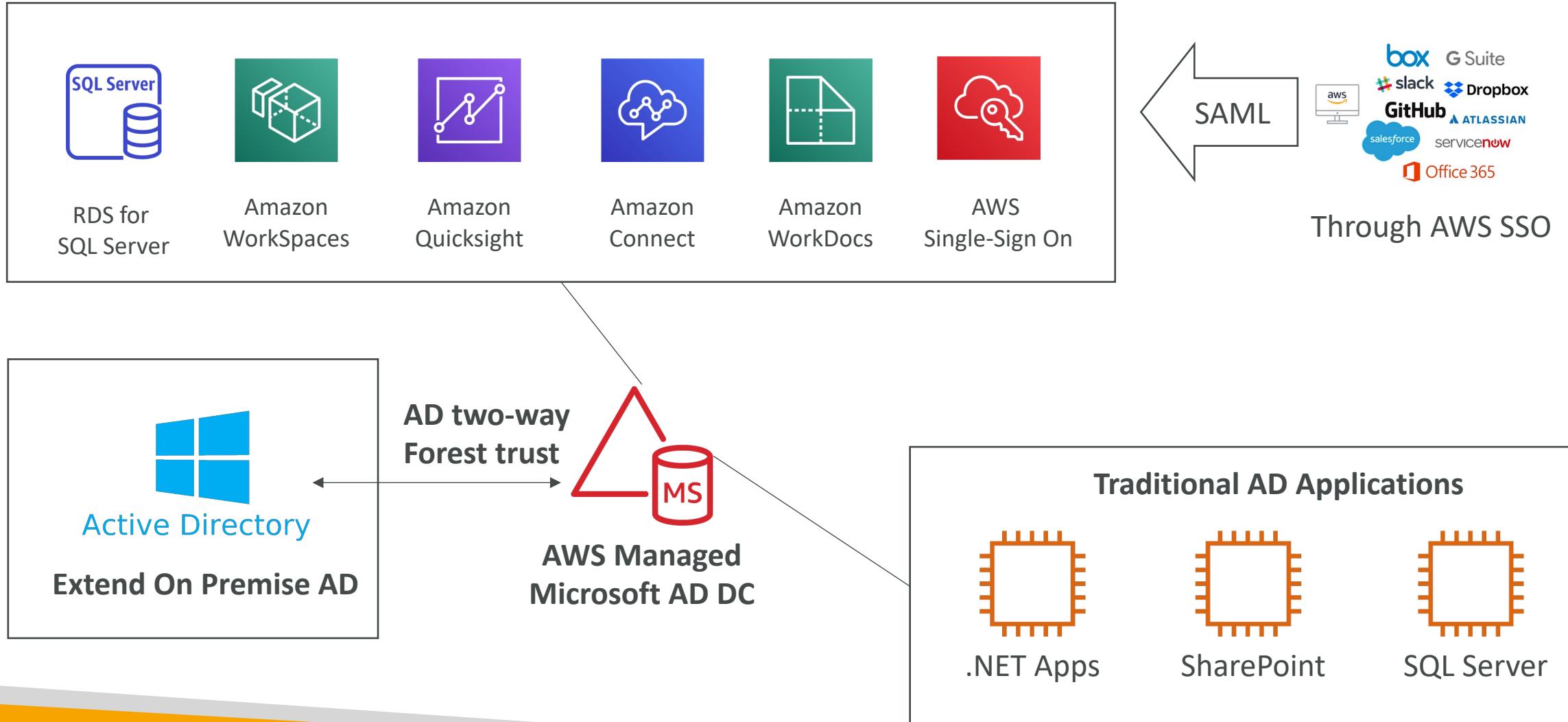
## AWS Managed Microsoft AD



- Managed Service: Microsoft AD in your AWS VPC
- EC2 Windows Instances:
  - EC2 Windows instances can join the domain and run traditional AD applications (sharepoint, etc)
  - Seamlessly Domain Join Amazon EC2 Instances from Multiple Accounts & VPCs
- Integrations:
  - RDS for SQL Server, AWS Workspaces, Quicksight...
  - AWS SSO to provide access to 3<sup>rd</sup> party applications
- Standalone repository in AWS or joined to on-premises AD
- Multi AZ deployment of AD in 2 AZ, # of DC (Domain Controllers) can be increased for scaling
- Automated backups
- Automated Multi-Region replication of your directory

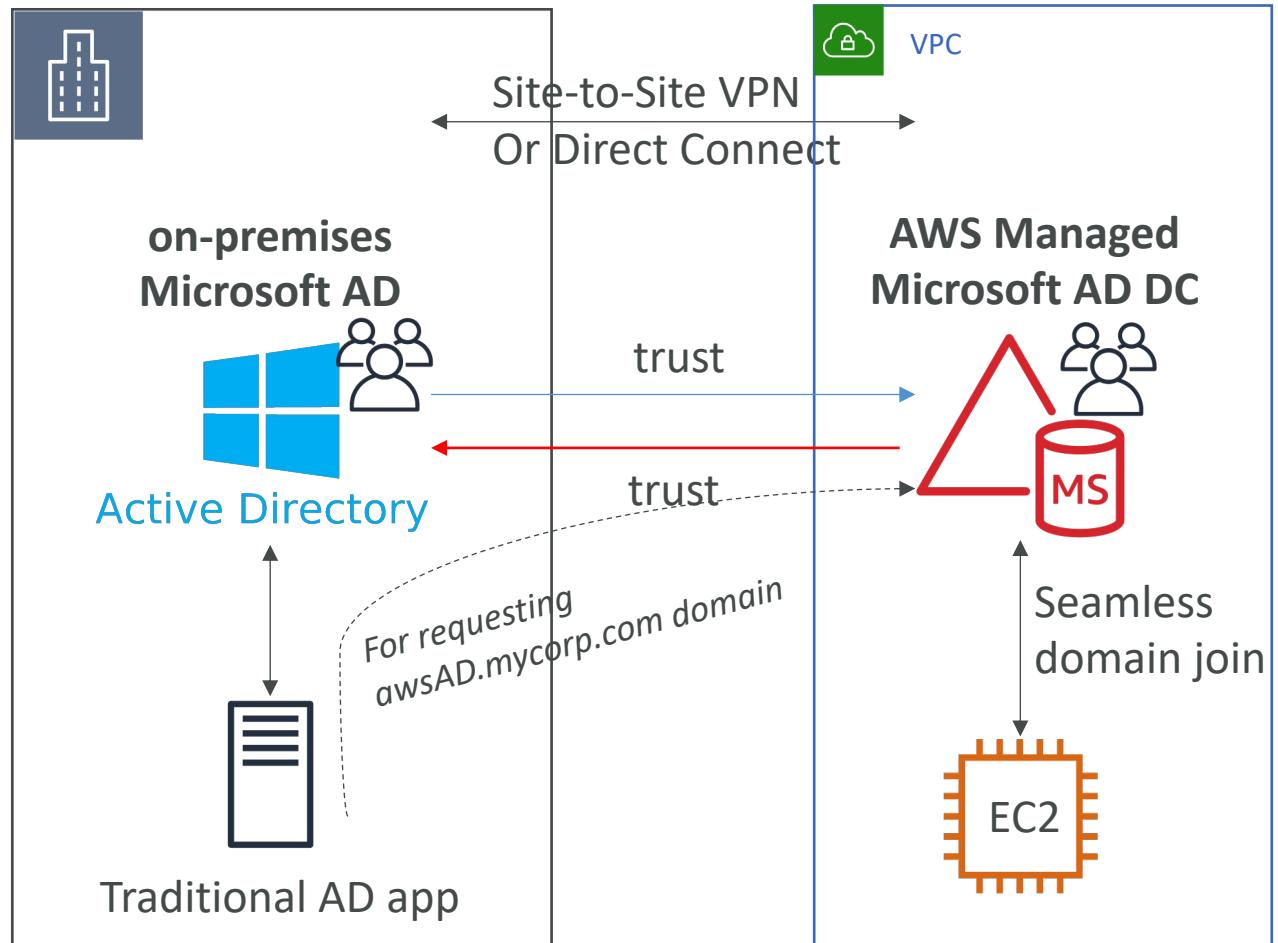


# AWS Microsoft Managed AD - Integrations



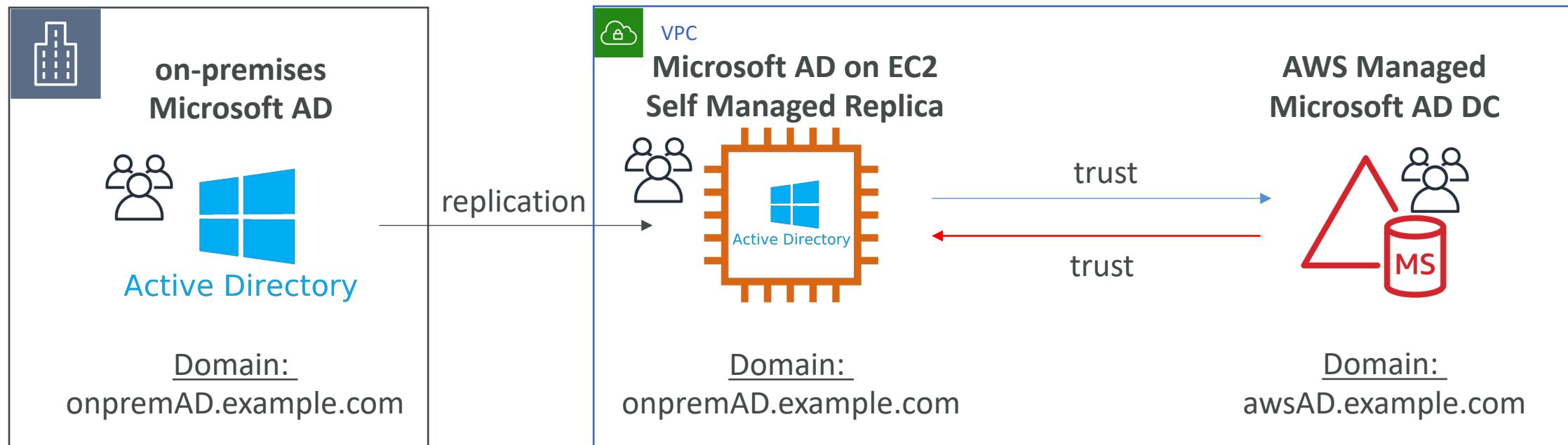
# Connect to on-premises AD

- Ability to connect your on-premises Active Directory to AWS Managed Microsoft AD
- Must establish a Direct Connect (DX) or VPN connection
- Can setup three kinds of forest trust:
  - One-way trust: AWS => on-premises
  - One-way trust: on-premises => AWS
  - Two-way forest trust: AWS ⇄ on-premises
- Forest trust is different than synchronization



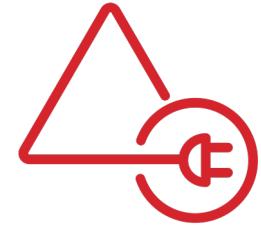
# Solution Architecture: Active Directory Replication

- You may want to create a replica of your AD on EC2 in the cloud to minimize latency of in case DX or VPN goes down
- Establish trust between the AWS Managed Microsoft AD and EC2

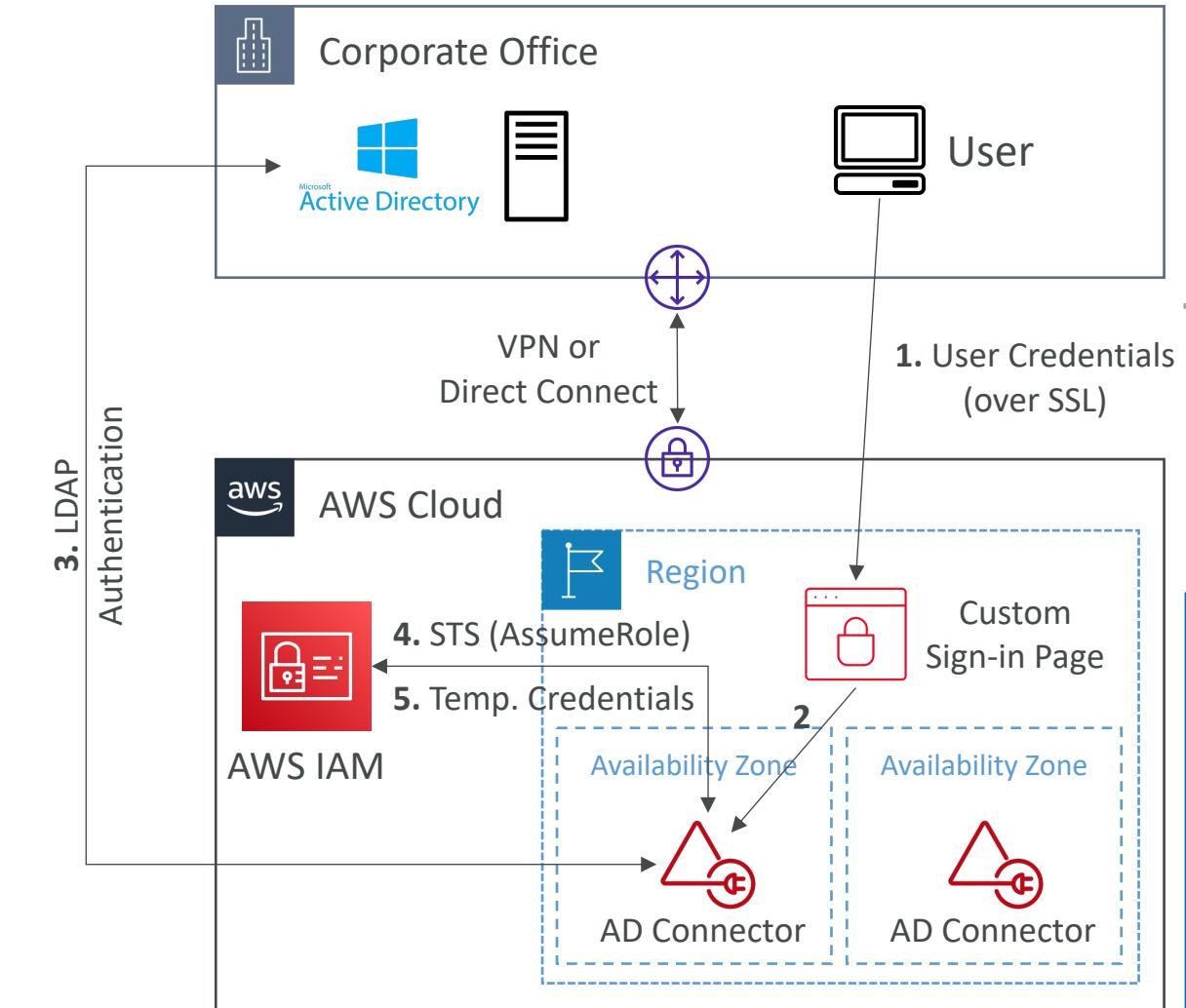


# AWS Directory Services

## AD Connector

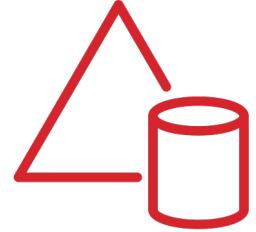


- AD Connector is a directory gateway to redirect directory requests to your on-premises Microsoft Active Directory
- No caching capability
- Manage users solely on-premises, no possibility of setting up a trust
- VPN or Direct Connect
- Doesn't work with SQL Server, doesn't do seamless joining, can't share directory



# AWS Directory Services

## Simple AD



- Simple AD is an inexpensive Active Directory–compatible service with the common directory features.
- Supports joining EC2 instances, manage users and groups
- Does not support MFA, RDS SQL server, AWS SSO
- Small: 500 users, large: 5000 users
- Powered by Samba 4, compatible with Microsoft AD
- lower cost, low scale, basic AD compatible, or LDAP compatibility
- No trust relationship

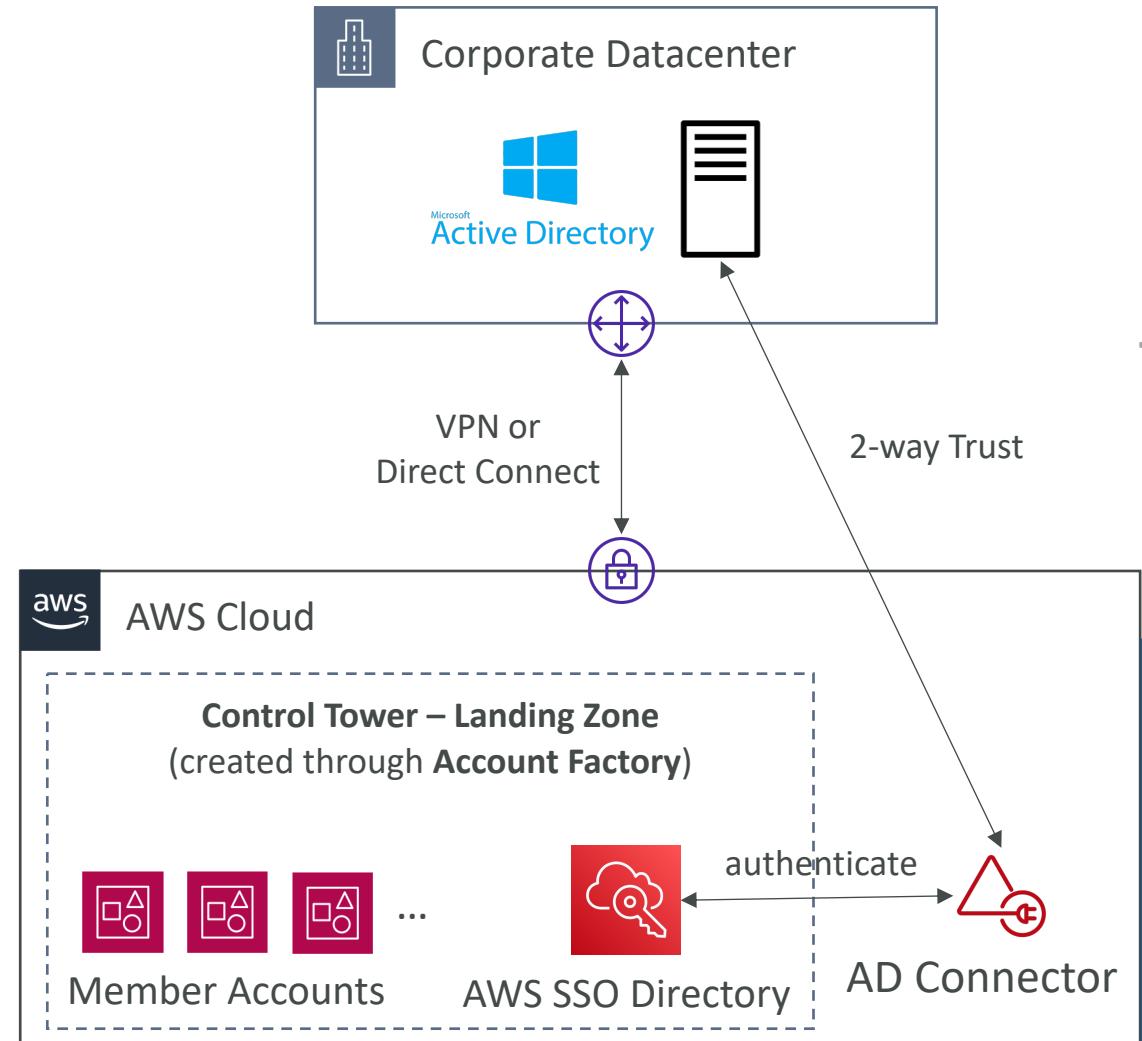
# AWS Control Tower



- Easy way to set up and govern a secure and compliant multi-account AWS environment based on best practices
- Benefits:
  - Automate the set up of your environment in a few clicks
  - Automate ongoing policy management using guardrails
  - Detect policy violations and remediate them
  - Monitor compliance through an interactive dashboard
- AWS Control Tower runs on top of AWS Organizations:
  - It automatically sets up AWS Organizations to organize accounts and implement SCPs (Service Control Policies)

# AWS Control Tower – Account Factory

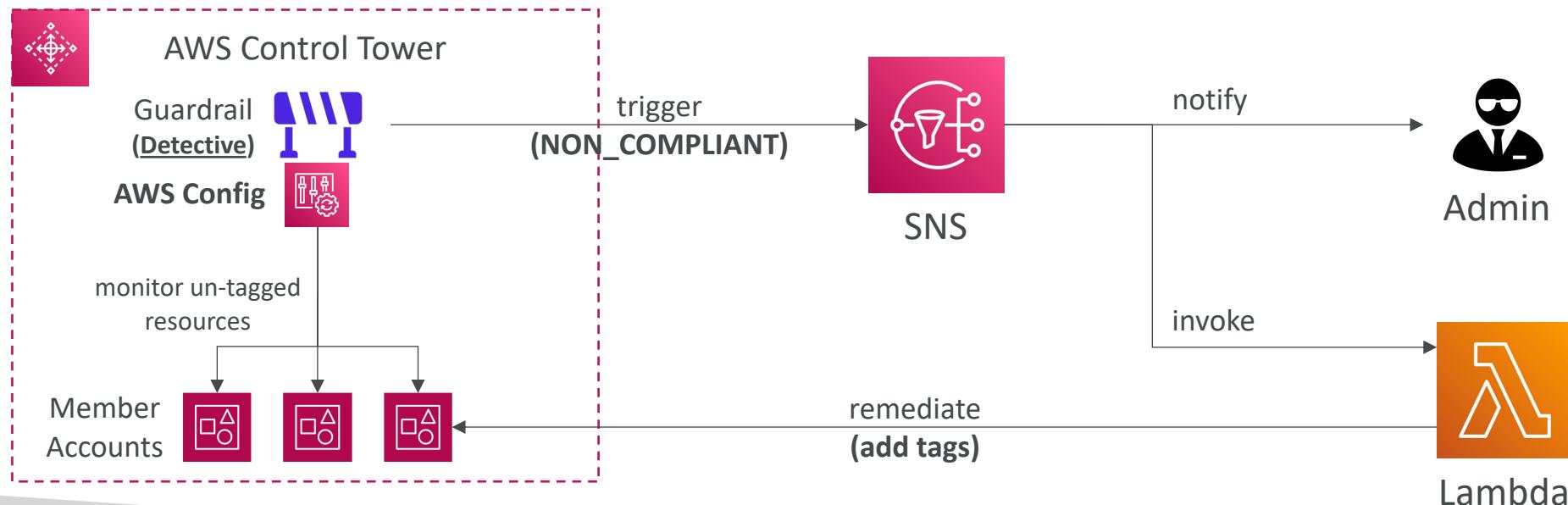
- Automates account provisioning and deployments
- Enables you to create pre-approved baselines and configuration options for AWS accounts in your organization (e.g., VPC default configuration, subnets, region, ...)
- Uses AWS Service Catalog to provision new AWS accounts



# AWS Control Tower – Detect and Remediate Policy Violations

- **Guardrail**

- Provides ongoing governance for your Control Tower environment (AWS Accounts)
- Preventive – using SCPs (e.g., Disallow Creation of Access Keys for the Root User)
- Detective – using AWS Config (e.g., Detect Whether MFA for the Root User is Enabled)
- Example: identify non-compliant resources (e.g., untagged resources)



# AWS Control Tower – Guardrails Levels

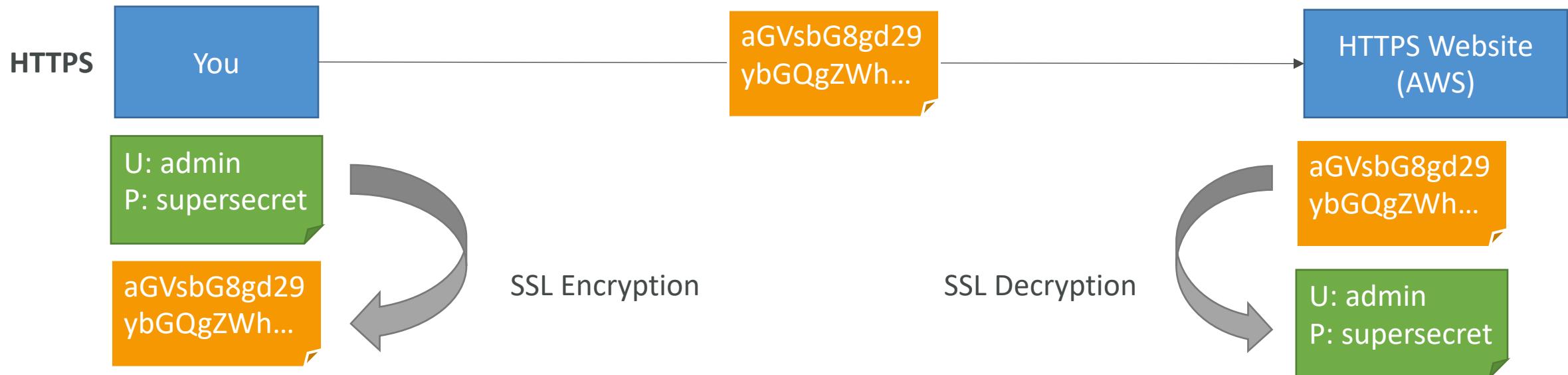
- **Mandatory**
  - Automatically enabled and enforced by AWS Control Tower
  - Example: Disallow public Read access to the Log Archive account
- **Strongly Recommended**
  - Based on AWS best practices (optional)
  - Example: Enable encryption for EBS volumes attached to EC2 instances
- **Elective**
  - Commonly used by enterprises (optional)
  - Example: Disallow delete actions without MFA in S3 buckets

# Domain 5 – Data Protection

# Why encryption?

## Encryption in flight (SSL)

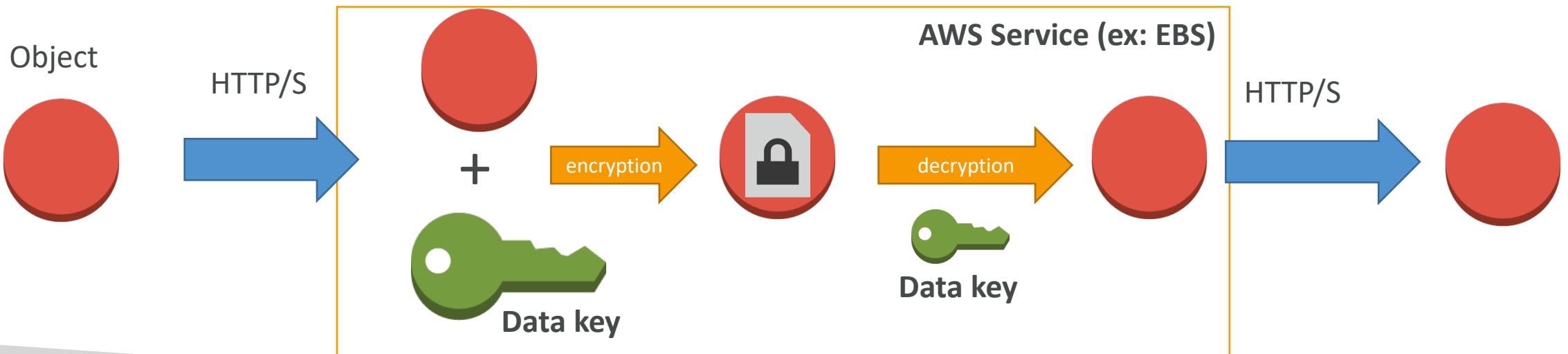
- Data is encrypted before sending and decrypted after receiving
- SSL certificates help with encryption (HTTPS)
- Encryption in flight ensures no MITM (man in the middle attack) can happen



# Why encryption?

## Server side encryption at rest

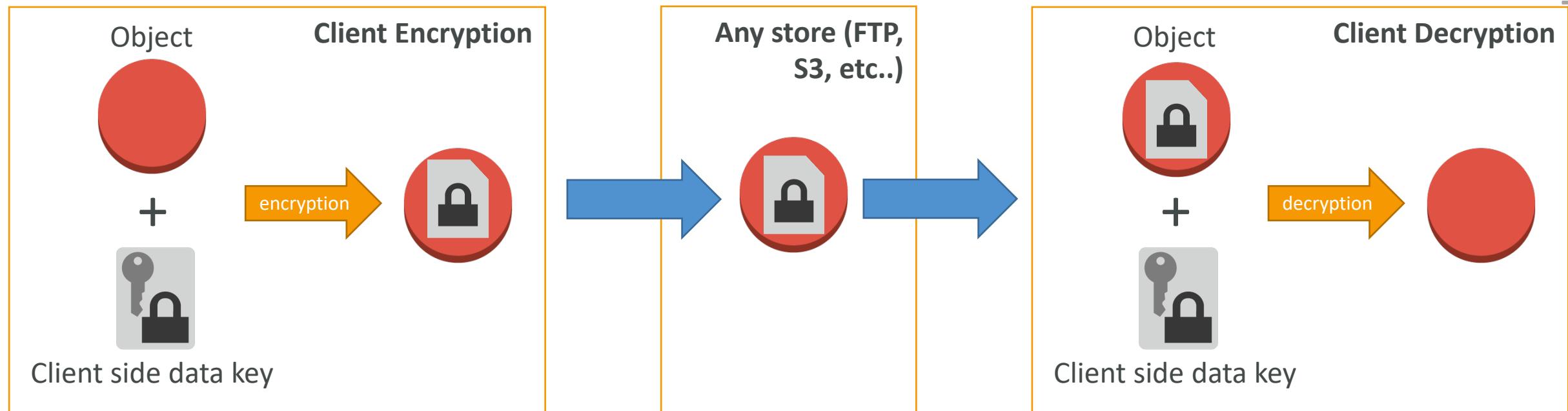
- Data is encrypted after being received by the server
- Data is decrypted before being sent
- It is stored in an encrypted form thanks to a key (usually a data key)
- The encryption / decryption keys must be managed somewhere and the server must have access to it



# Why encryption?

## Client side encryption

- Data is encrypted by the client and never decrypted by the server
- Data will be decrypted by a receiving client
- The server should not be able to decrypt the data
- Could leverage Envelope Encryption

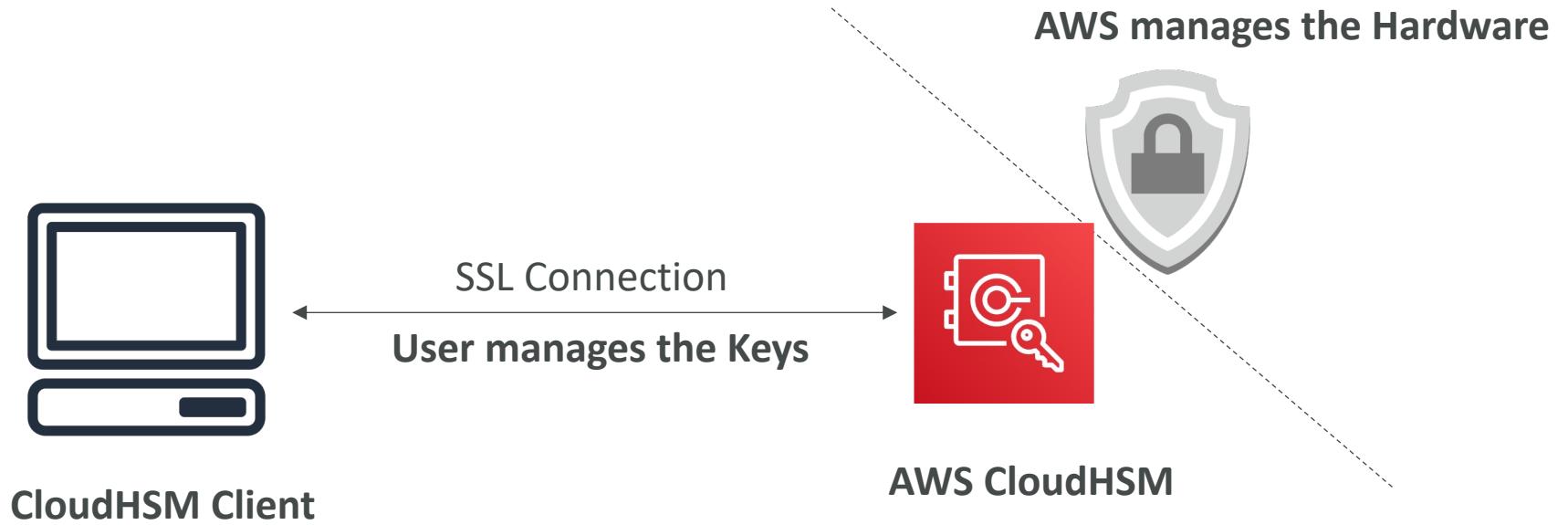


# CloudHSM



- KMS => AWS manages the software for encryption
- CloudHSM => AWS provisions encryption **hardware**
- Dedicated Hardware (HSM = Hardware Security Module)
- You manage your own encryption keys entirely (not AWS)
- HSM device is tamper resistant, FIPS 140-2 Level 3 compliance
- Supports both symmetric and **asymmetric** encryption (SSL/TLS keys)
- No free tier available
- Must use the CloudHSM Client Software
- Redshift supports CloudHSM for database encryption and key management
- **Good option to use with SSE-C encryption**

# CloudHSM Diagram



## IAM permissions:

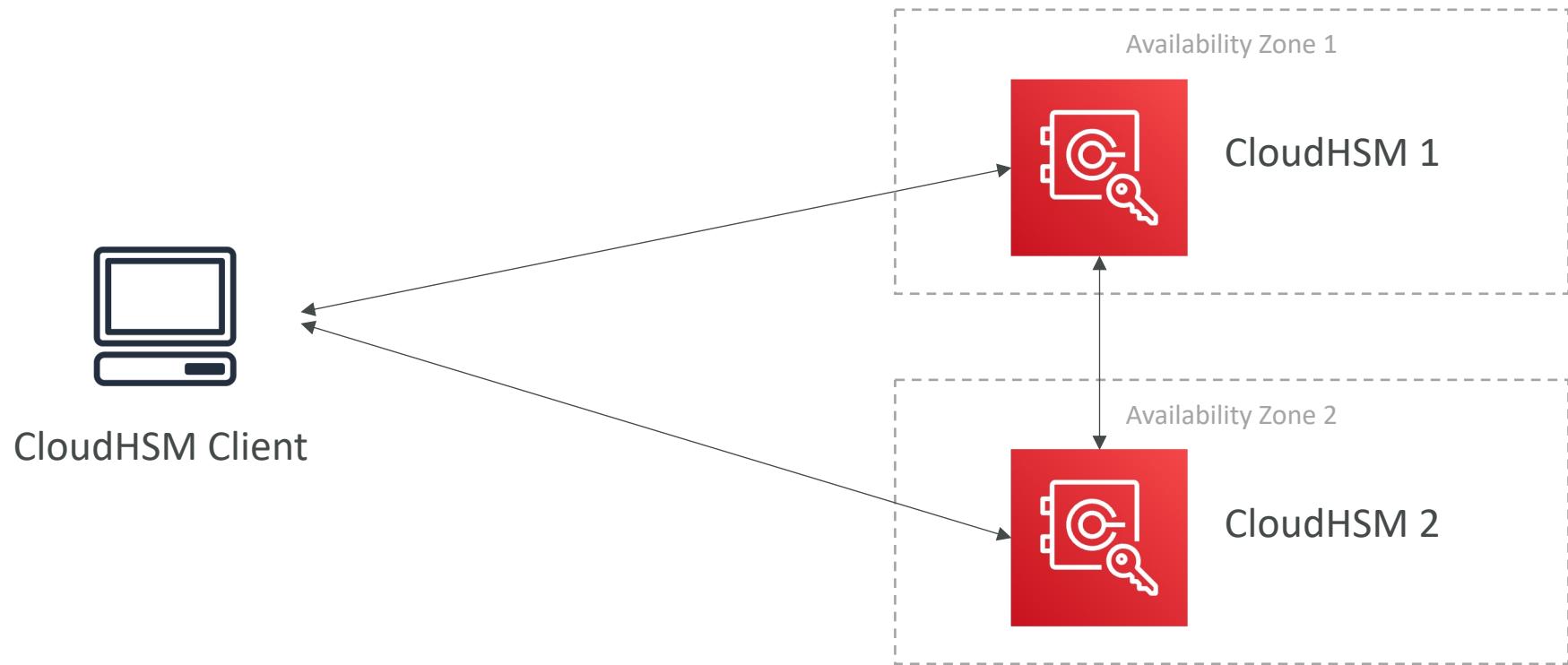
- CRUD an HSM Cluster

## CloudHSM Software:

- Manage the Keys
- Manage the Users

# CloudHSM – High Availability

- CloudHSM clusters are spread across Multi AZ (HA)
- Great for availability and durability



# CloudHSM vs. KMS

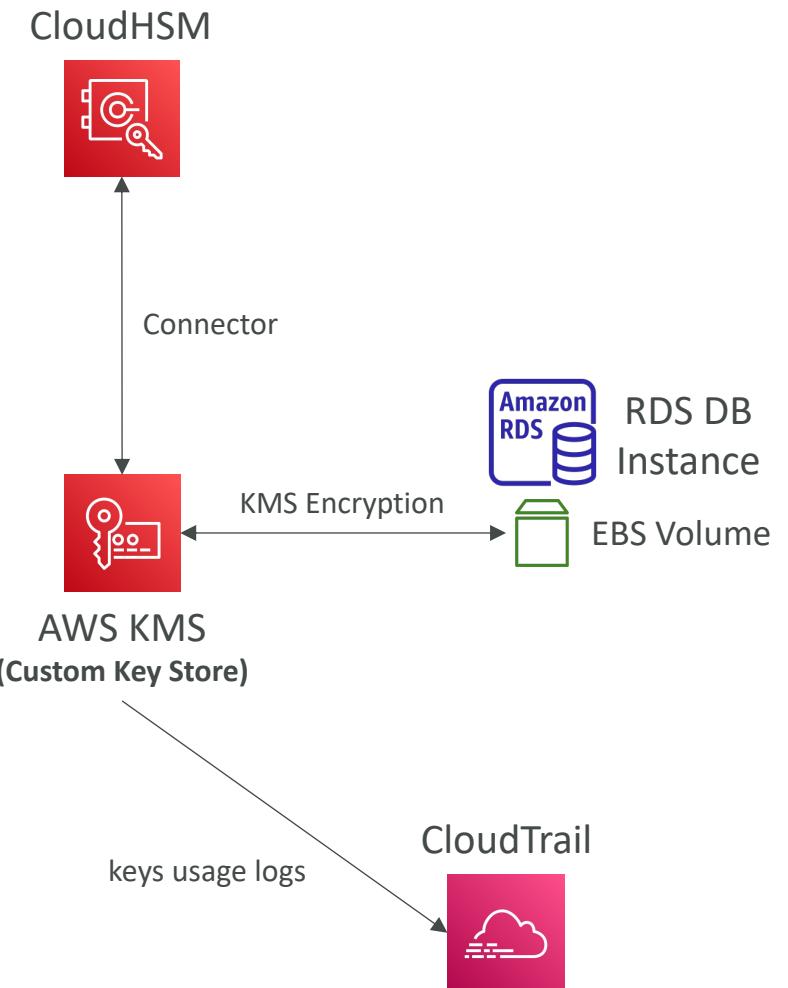
Feature	AWS KMS	AWS CloudHSM
Tenancy	Multi-Tenant	Single-Tenant
Standard	FIPS 140-2 Level 2	FIPS 140-2 Level 3
Master Keys	<ul style="list-style-type: none"><li>• AWS Owned Keys</li><li>• AWS Managed Keys</li><li>• Customer Managed KMS Keys</li></ul>	Customer Managed CMK
Key Types	<ul style="list-style-type: none"><li>• Symmetric</li><li>• Asymmetric</li><li>• Digital Signing</li></ul>	<ul style="list-style-type: none"><li>• Symmetric</li><li>• Asymmetric</li><li>• Digital Signing &amp; Hashing</li></ul>
Key Accessibility	Accessible in multiple AWS regions KMS Key Replication	<ul style="list-style-type: none"><li>• Deployed and managed in a VPC</li><li>• Can be shared across VPCs (VPC Peering)</li></ul>
Cryptographic Acceleration	None	<ul style="list-style-type: none"><li>• SSL/TLS Acceleration</li><li>• Oracle TDE Acceleration</li></ul>
Access & Authentication	AWS IAM	You create users and manage their permissions

# CloudHSM vs. KMS

Feature	AWS KMS	AWS CloudHSM
High Availability	AWS Managed Service	Add multiple HSMs over different AZs
Audit Capability	<ul style="list-style-type: none"><li>• CloudTrail</li><li>• CloudWatch</li></ul>	<ul style="list-style-type: none"><li>• CloudTrail</li><li>• CloudWatch</li><li>• MFA support</li></ul>
Free Tier	Yes	No

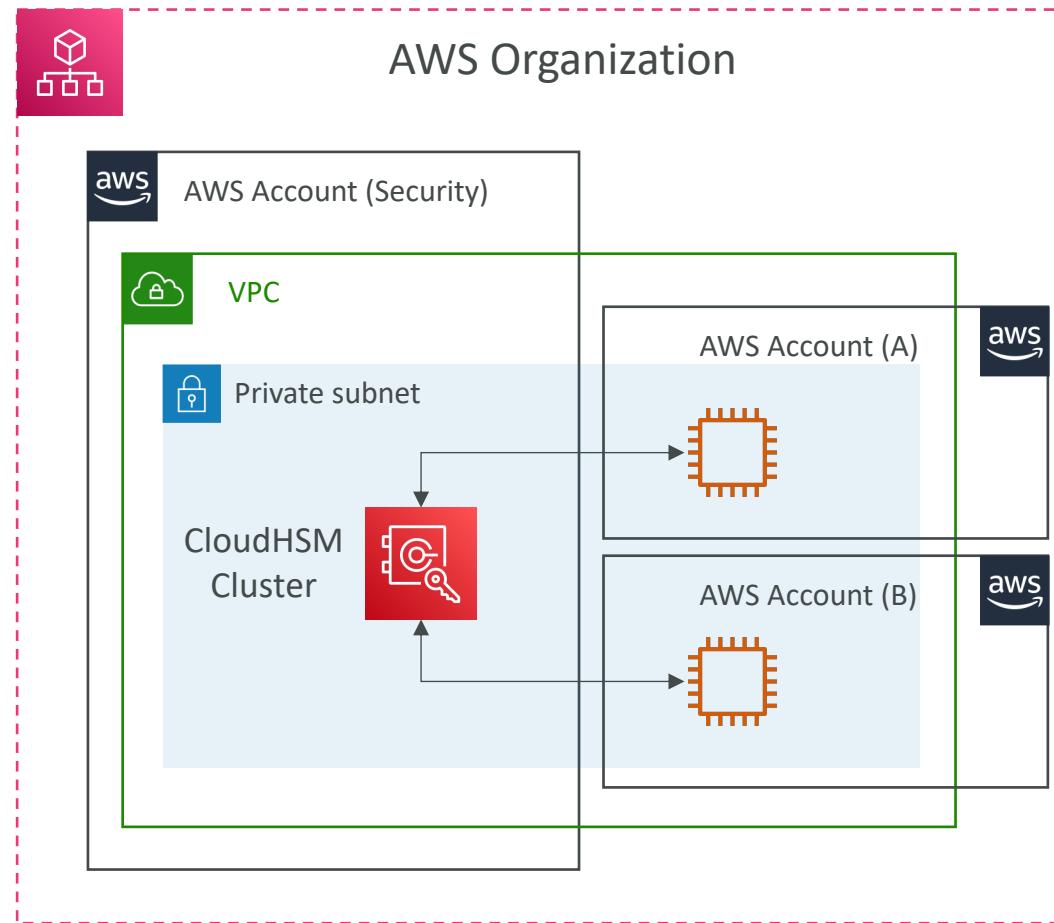
# CloudHSM – Integration with AWS and 3rd Party Services

- Integration with AWS Services
  - Through integration with AWS KMS
  - Configure KMS Custom Key Store with CloudHSM
  - Example: EBS, S3, RDS, ...
  - Supports RDS Oracle TDE (through KMS)
- Integration with 3<sup>rd</sup> Party Services
  - Allows creating and storing keys in CloudHSM
  - Use cases: SSL/TLS Offload, Windows Server Certificate Authority (CA), Oracle TDE, Microsoft SignTool, Java Keytool, ...



# CloudHSM – Sharing Cluster Across-Accounts

- You can share the private subnets a CloudHSM clusters resides in using AWS RAM
- You CANNOT share the CloudHSM cluster itself
- Share VPC Subnets with entire Organization, specific OUs, or AWS accounts
- Note: configure CloudHSM Security Group to allow traffic from clients



# AWS KMS (Key Management Service)



- Anytime you hear “encryption” for an AWS service, it’s most likely KMS
- Easy way to control access to your data, AWS manages keys for us
- Fully integrated with IAM for authorization
- Seamlessly integrated into:
  - Amazon EBS: encrypt volumes
  - Amazon S3: Server-side encryption of objects
  - Amazon Redshift: encryption of data
  - Amazon RDS: encryption of data
  - Amazon SSM: Parameter store
  - Etc...
- But you can also use the CLI / SDK

# KMS – KMS Key Types

- **Symmetric (AES-256 keys)**
  - First offering of KMS, single encryption key that is used to Encrypt and Decrypt
  - AWS services that are integrated with KMS use Symmetric KMS keys
  - Necessary for envelope encryption
  - You never get access to the KMS key unencrypted (must call KMS API to use)
- **Asymmetric (RSA & ECC key pairs)**
  - Public (Encrypt) and Private Key (Decrypt) pair
  - Used for Encrypt/Decrypt, or Sign/Verify operations
  - The public key is downloadable, but you can't access the Private Key unencrypted
  - Use case: encryption outside of AWS by users who can't call the KMS API

# Types of KMS Keys

- **Customer Managed Keys**
  - Create, manage and use, can enable or disable
  - Possibility of rotation policy (new key generated every year; old key preserved)
  - Can add a Key Policy (resource policy) & audit in CloudTrail
  - Leverage for envelope encryption
- **AWS Managed Keys**
  - Used by AWS service (aws/s3, aws/ebs, aws/redshift)
  - Managed by AWS (automatically rotated every 1 years)
  - View Key Policy & audit in CloudTrail
- **AWS Owned Keys**
  - Created and managed by AWS, use by some AWS services to protect your resources
  - Used in multiple AWS accounts, but they are not in your AWS account
  - You can't view, use, track, or audit

# Types of KMS Keys

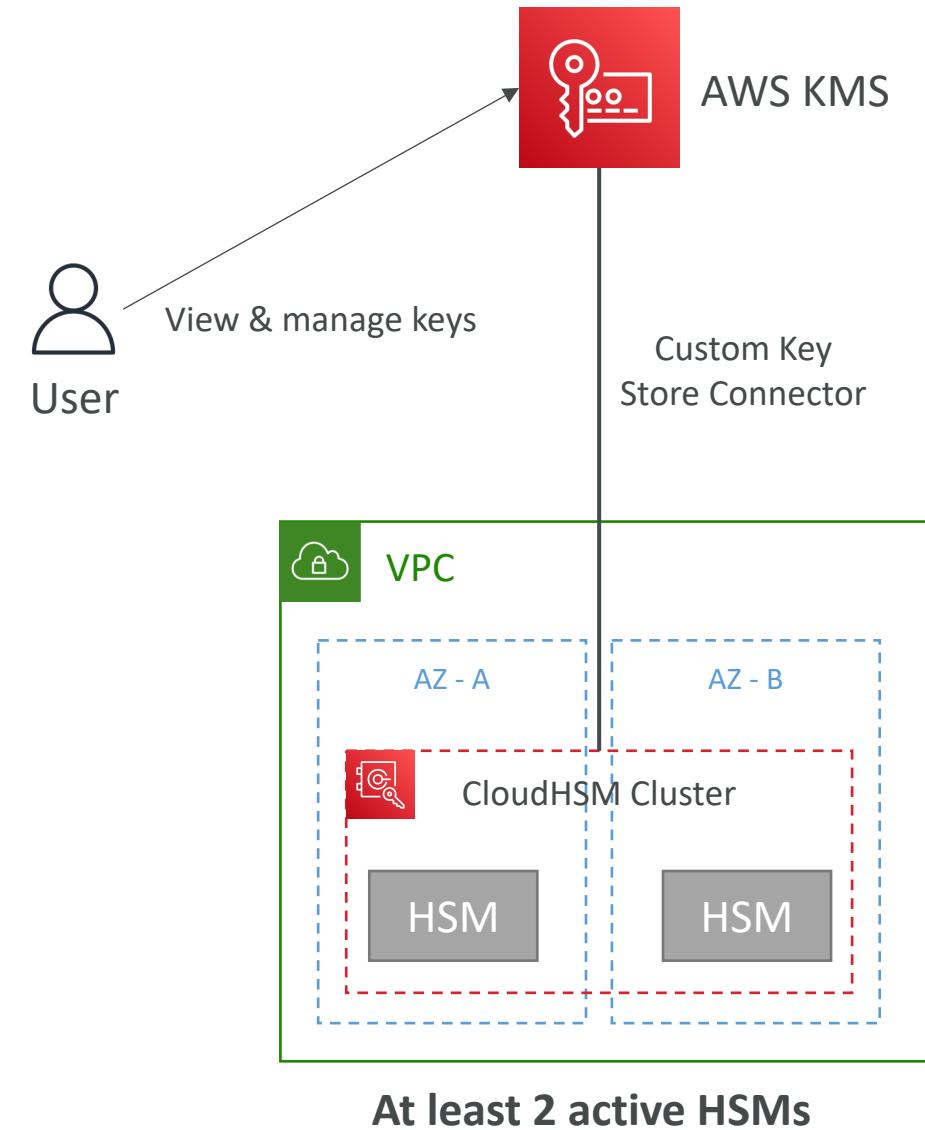
KMS Key	Customer Managed Key	AWS Managed Key	AWS Owned Key
Can view metadata?	✓	✓	✗
Can manage?	✓	✗	✗
Used only for my AWS account?	✓	✓	✗
Automatic Rotation	Optional (every 1 year)	Required (every 1 years)	Varies

# KMS Key Material Origin

- Identifies the source of the key material in the KMS key
- Can't be changed after creation
- **KMS (AWS\_KMS)** – default
  - AWS KMS creates and manages the key material in its own key store
- **External (EXTERNAL)**
  - You import the key material into the KMS key
  - You're responsible for securing and managing this key material outside of AWS
- **Custom Key Store (AWS\_CLOUDHSM)**
  - AWS KMS creates the key material in a custom key store (CloudHSM Cluster)

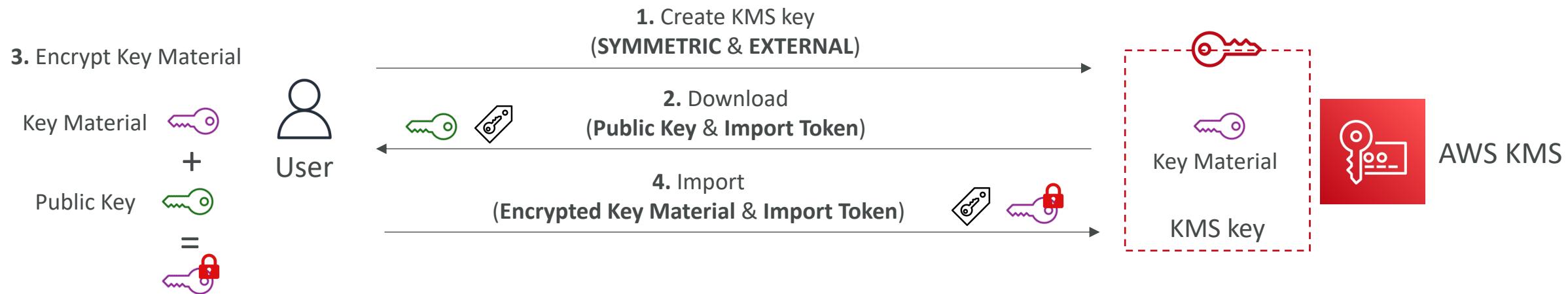
# KMS Key Source – Custom Key Store (CloudHSM)

- Integrate KMS with CloudHSM cluster as a Custom Key Store
- Key materials are stored in a CloudHSM cluster that you own and manage
- The cryptographic operations are performed in the HSMs
- Use cases:
  - You need direct control over the HSMs
  - KMS keys needs to be stored in a dedicated HSMs
  - HSMs must be validated at FIPS 140-2 Level 3 (KMS validated at FIPS 140-2 Level 2)

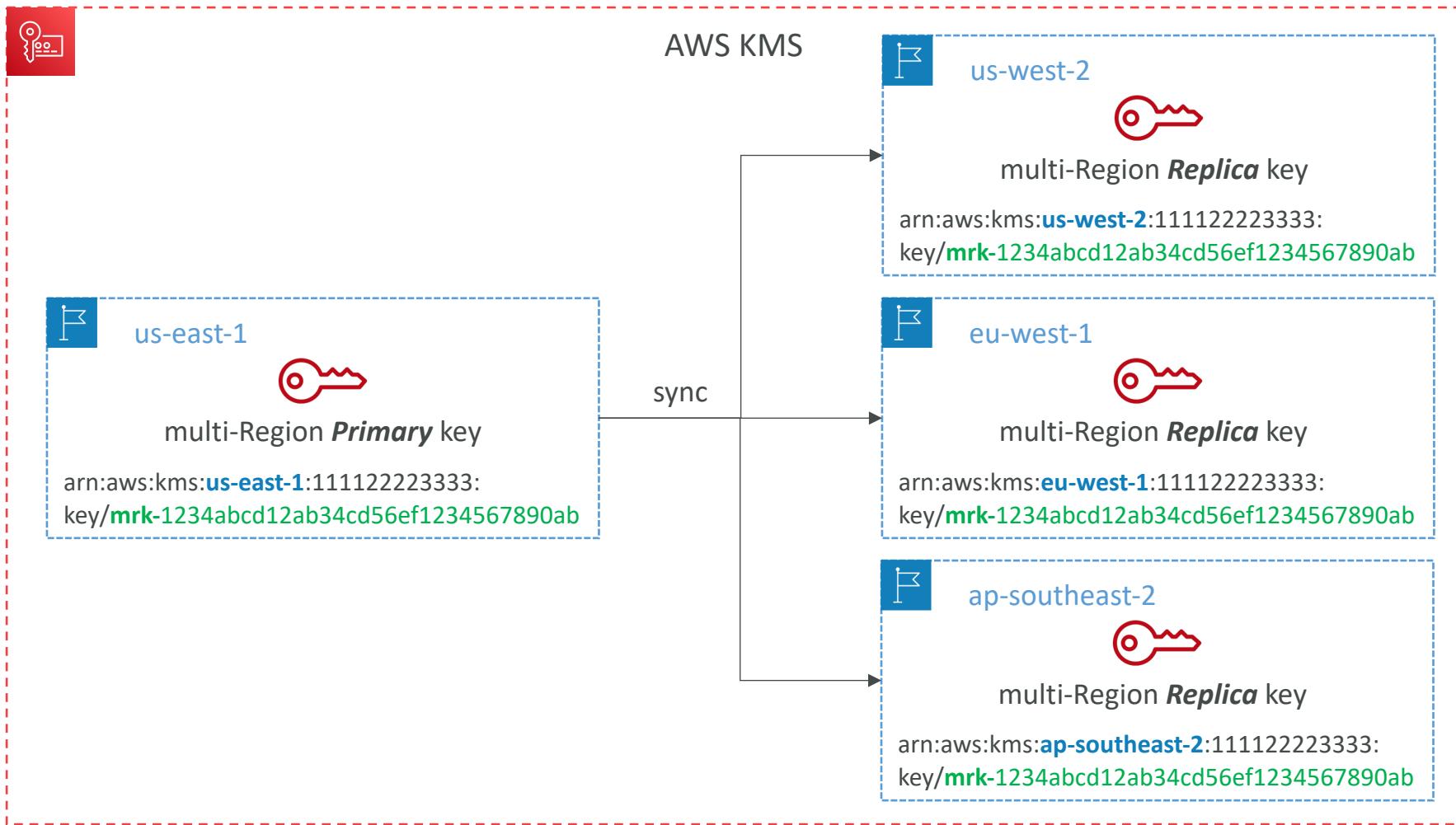


# KMS Key Source - External

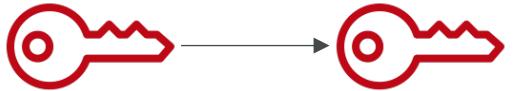
- Import your own key material into KMS key, Bring Your Own Key (BYOK)
- You're responsible for key material's security, availability, and durability outside of AWS
- Must be 256-bit **Symmetric** key (Asymmetric is NOT supported)
- Can't be used with Custom Key Store (CloudHSM)
- Manually rotate your KMS key (Automatic Key Rotation is NOT supported)



# KMS Multi-Region Keys



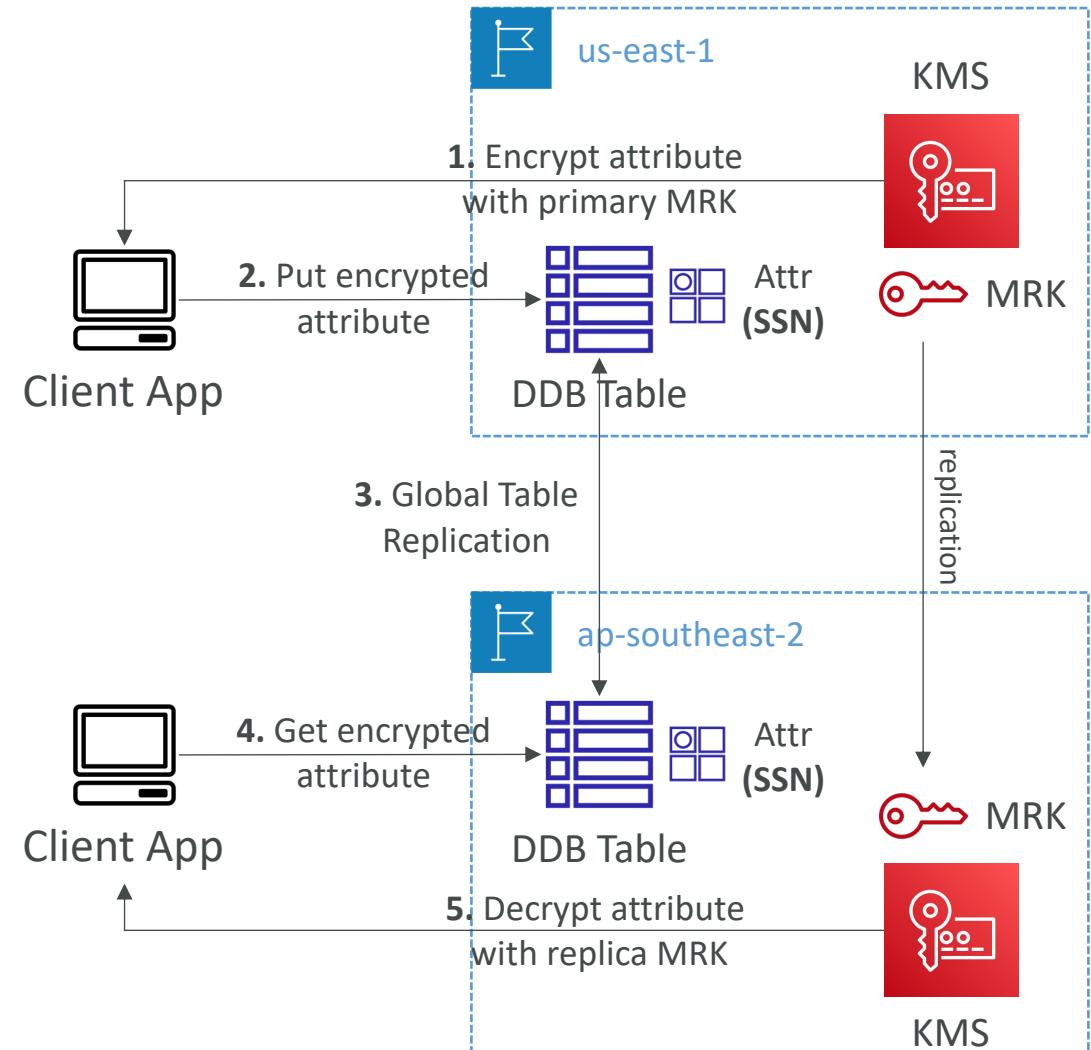
# KMS Multi-Region Keys



- Identical KMS keys in different AWS Regions that can be used interchangeably
- Multi-Region keys have the same key ID, key material, automatic rotation...
- Encrypt in one Region and decrypt in other Regions
- No need to re-encrypt or making cross-Region API calls
- KMS Multi-Region are NOT global (Primary + Replicas)
- Each Multi-Region key is managed **independently**
- **Use cases:** global client-side encryption, encryption on Global DynamoDB, Global Aurora

# DynamoDB Global Tables and KMS Multi-Region Keys Client-Side encryption

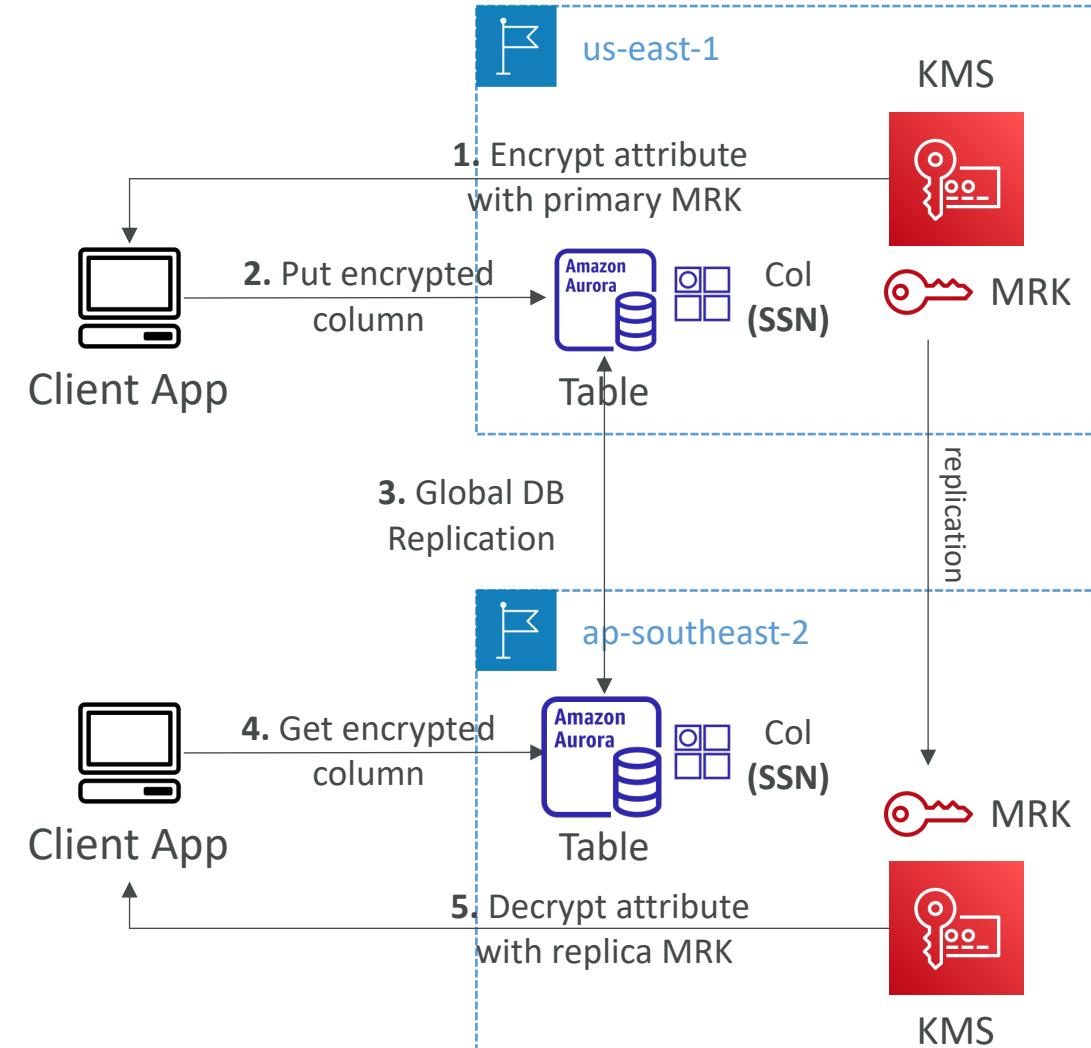
- We can encrypt specific attributes client-side in our DynamoDB table using the **Amazon DynamoDB Encryption Client**
- Combined with Global Tables, the client-side encrypted data is replicated to other regions
- If we use a multi-region key, replicated in the same region as the DynamoDB Global table, then clients in these regions can use low-latency API calls to KMS in their region to decrypt the data client-side
- Using client-side encryption we can protect specific fields and guarantee only decryption if the client has access to an API key



# Global Aurora and KMS Multi-Region Keys

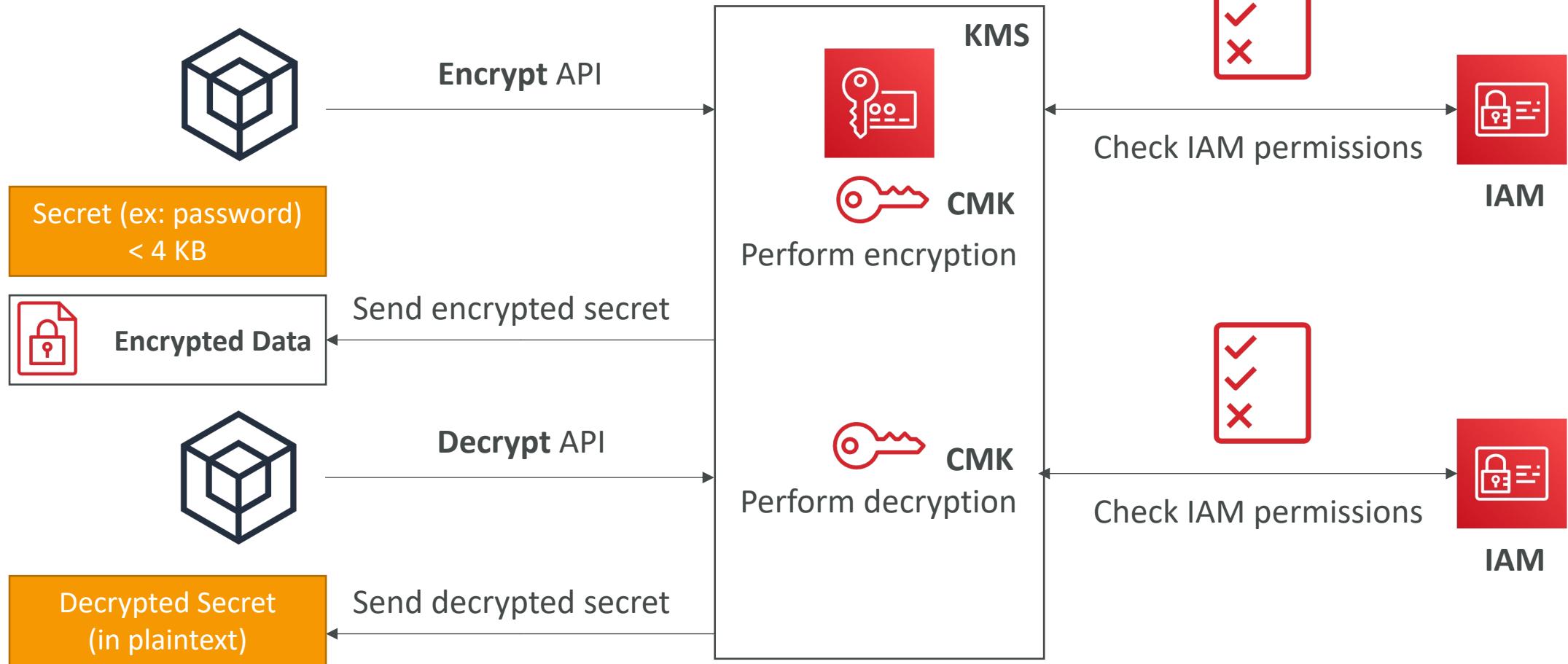
## Client-Side encryption

- We can encrypt specific attributes client-side in our Aurora table using the **AWS Encryption SDK**
- Combined with Aurora Global Tables, the client-side encrypted data is replicated to other regions
- If we use a multi-region key, replicated in the same region as the Global Aurora DB, then clients in these regions can use low-latency API calls to KMS in their region to decrypt the data client-side
- Using client-side encryption we can protect specific fields and guarantee only decryption if the client has access to an API key, we can **protect specific fields even from database admins**



# How does KMS work?

## API – Encrypt and Decrypt

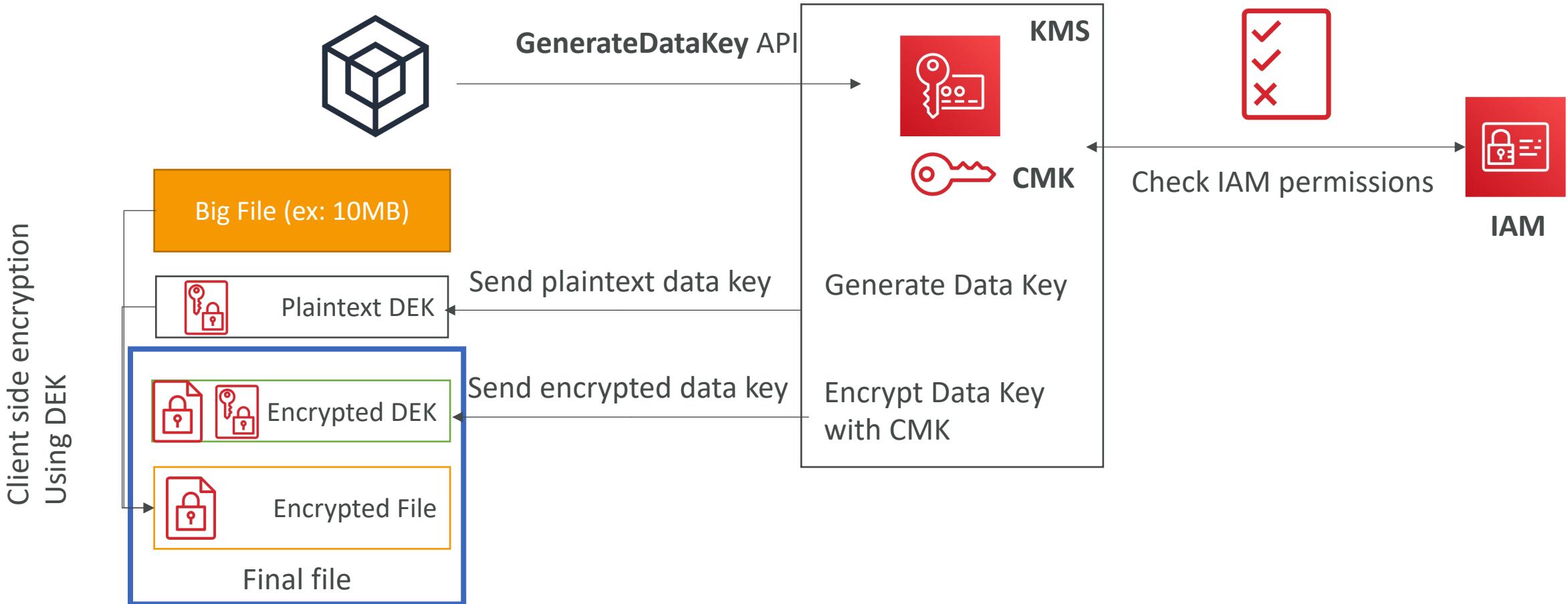


# Envelope Encryption

- KMS Encrypt API call has a limit of 4 KB
- If you want to encrypt >4 KB, we need to use Envelope Encryption
- The main API that will help us is the `GenerateDataKey` API
- For the exam: anything over 4 KB of data that needs to be encrypted must use the Envelope Encryption == `GenerateDataKey` API

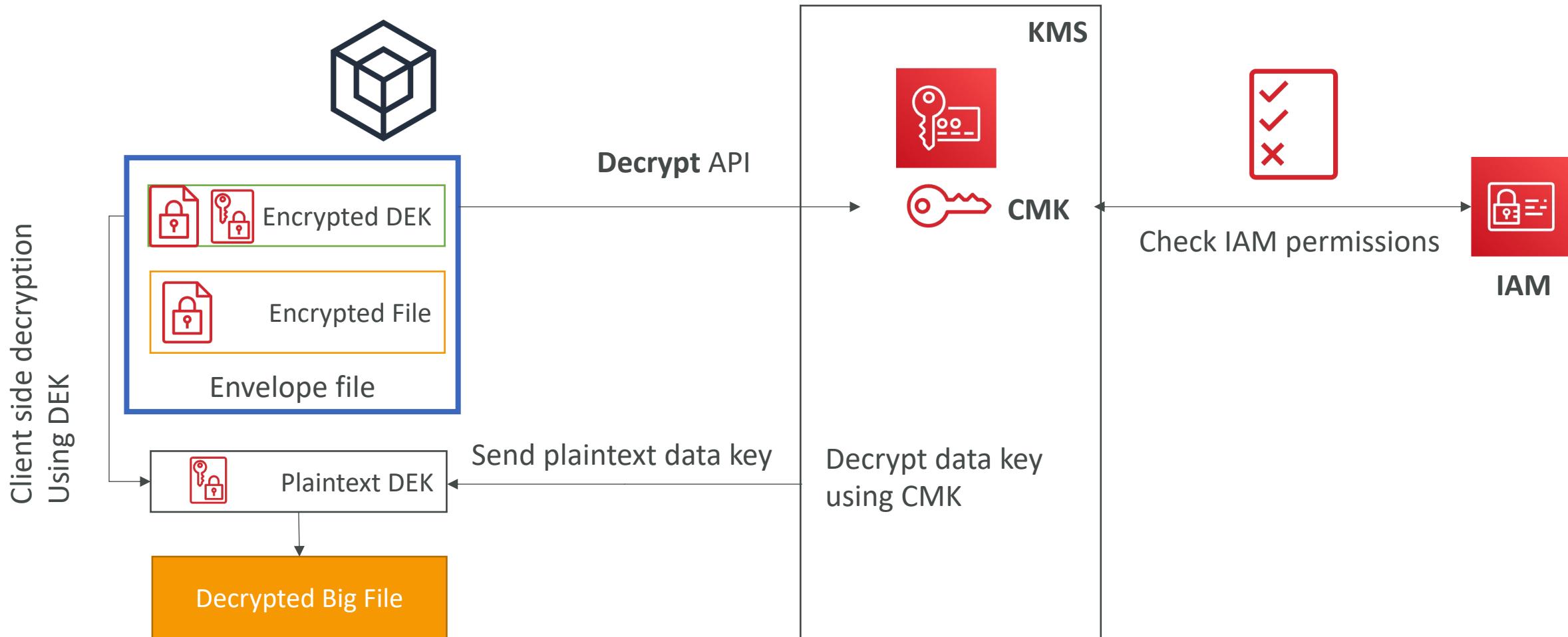
# Deep dive into Envelope Encryption

## GenerateDataKey API



# Deep dive into Envelope Encryption

## Decrypt envelope data





# Encryption SDK

- The AWS Encryption SDK implemented Envelope Encryption for us
  - The Encryption SDK also exists as a CLI tool we can install
  - Implementations for Java, Python, C, JavaScript
- 
- **Feature - Data Key Caching:**
    - re-use data keys instead of creating new ones for each encryption
    - Helps with reducing the number of calls to KMS with a security trade-off
    - Use LocalCryptoMaterialsCache (max age, max bytes, max number of messages)

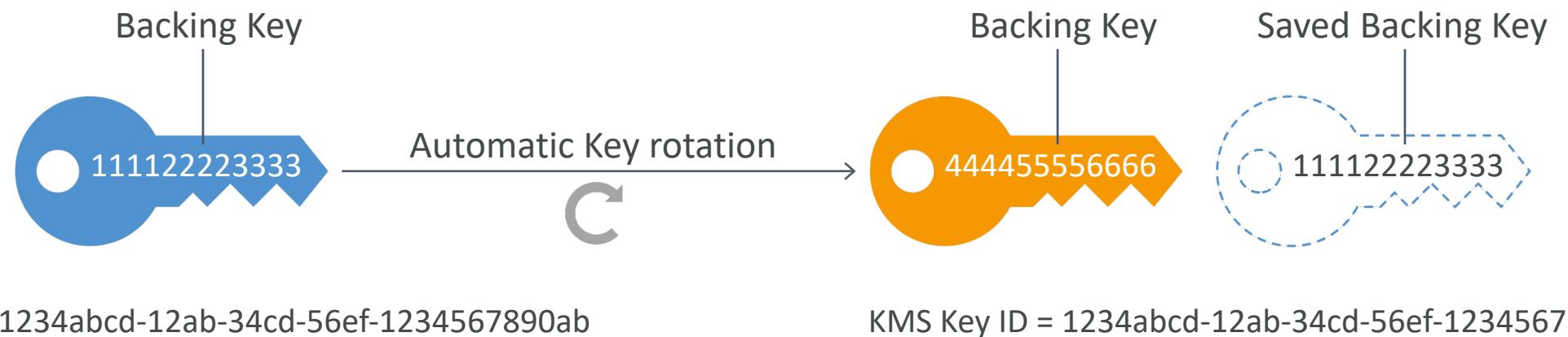
# KMS Symmetric – API Summary



- **Encrypt:** encrypt up to 4 KB of data through KMS
- **GenerateDataKey:** generates a unique symmetric data key (DEK)
  - returns a plaintext copy of the data key
  - AND a copy that is encrypted under the CMK that you specify
- **GenerateDataKeyWithoutPlaintext:**
  - Generate a DEK to use at some point (not immediately)
  - DEK that is encrypted under the CMK that you specify (must use Decrypt later)
- **Decrypt:** decrypt up to 4 KB of data (including Data Encryption Keys)
- **GenerateRandom:** Returns a random byte string

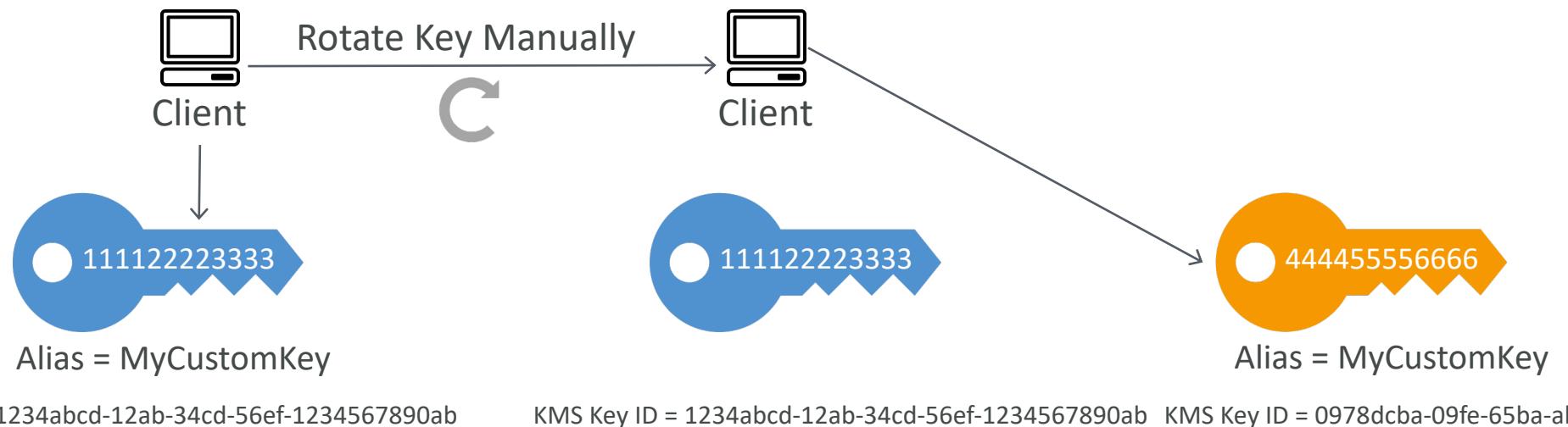
# KMS Automatic Key Rotation

- AWS-managed KMS Keys: automatically rotated every 1 year
- For Customer-Managed KMS Key
  - Automatic key rotation is optionally enabled, will happen every 1 year
  - Previous key is kept active so you can decrypt old data
  - New Key has the same KMS Key ID (only the backing key is changed)



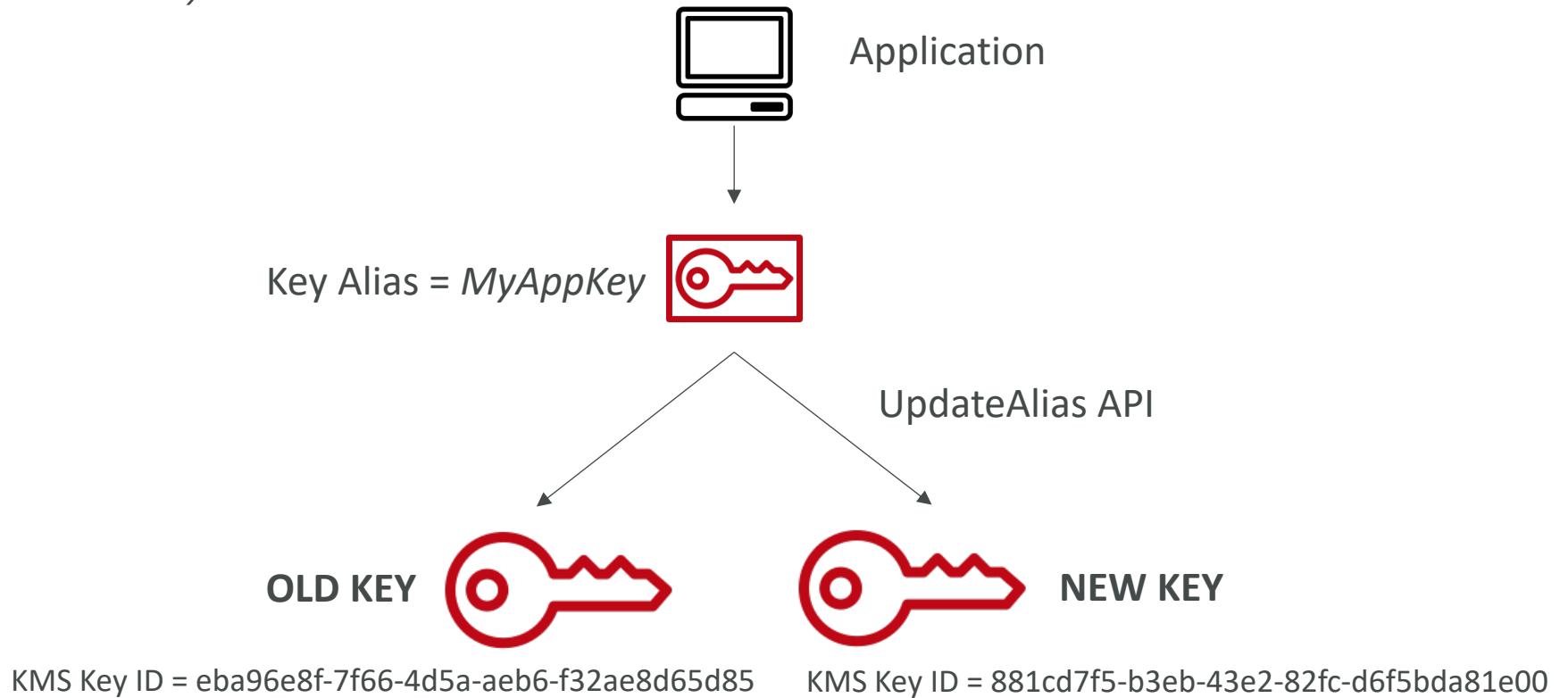
# KMS Manual Key Rotation (for Customer-Managed KMS Key and Imports)

- When you want to rotate key every 90 days, 180 days, etc...
- New Key has a different KMS Key ID
- Keep the previous key active so you can decrypt old data
- Better to use aliases in this case (to hide the change of key for the application)
- Good solution to rotate KMS Key that are not eligible for automatic rotation (like asymmetric CMK)



# KMS Alias Updating

- Better to use aliases in this case (to hide the change of key for the application)

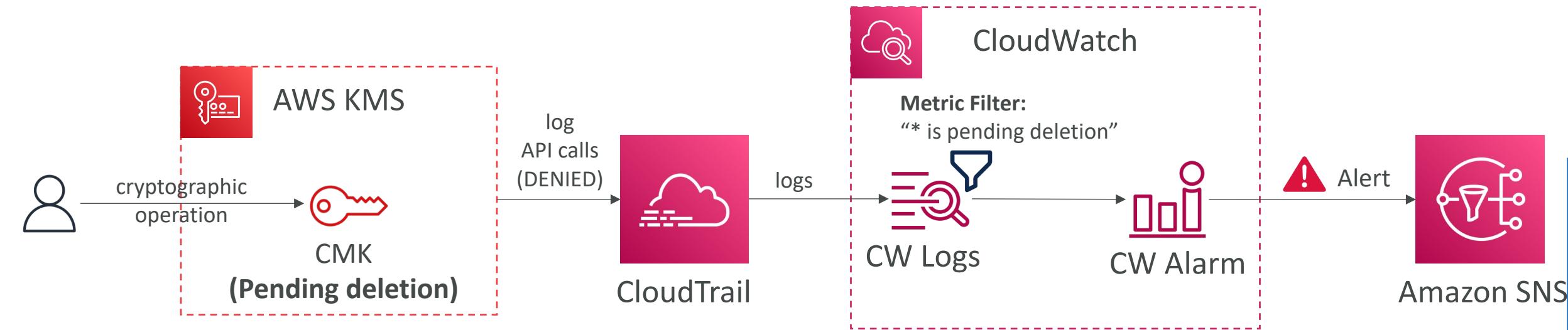


# KMS Key Deletion

- Generated Keys (from within KMS)
  - No expiration date
  - Cannot be deleted immediately, mandatory 7 to 30 days waiting period
    - You can cancel key deletion during the waiting period
    - During the waiting period, the KMS Key cannot be used for Encrypt / Decrypt
    - Everything will be deleted at the end of the waiting period
  - You may manually disable it immediately instead (to re-enable it later)
- Imported Keys:
  - You may set an expiration period on the Key
    - KMS will delete the key material
    - You can also delete the key material on demand
    - The metadata is kept so you can re-import in the future
  - You may manually disable it and schedule for deletion (everything is deleted)
- AWS managed keys or AWS owned keys cannot be deleted

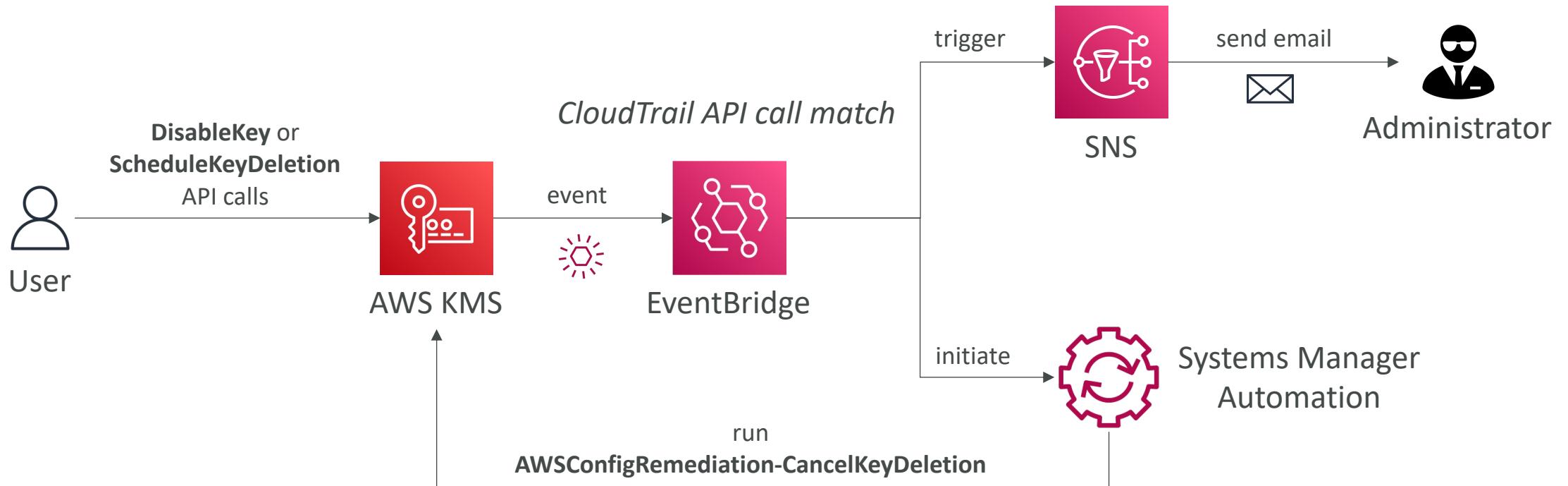
# KMS Key Deletion – CloudWatch Alarm

- Use CloudTrail, CloudWatch Logs, CloudWatch Alarms and SNS to be notified when someone tries to use a CMK that's "Pending deletion" in a cryptographic operation (Encrypt, Decrypt, ...)



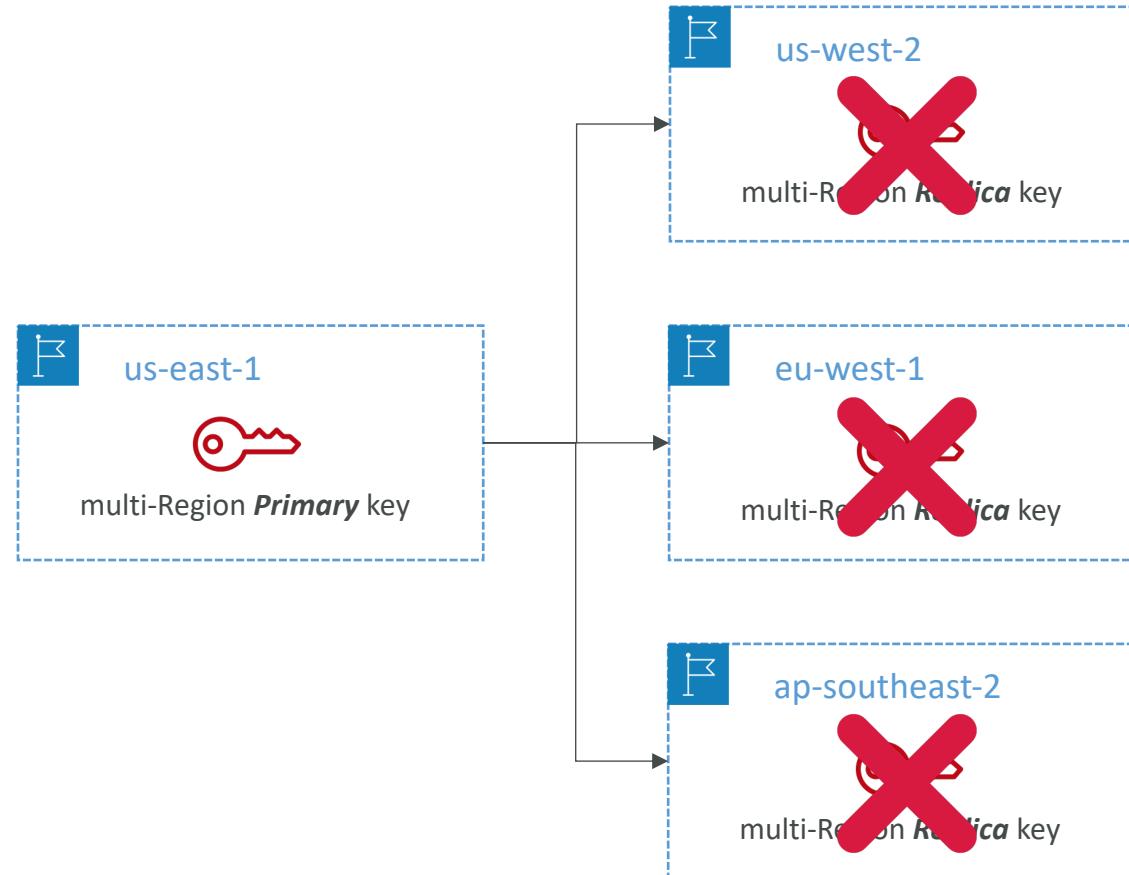
# KMS Key Deletion – Notifications

- To be notified of Keys being deleted or disabled
- Using CloudTrail + EventBridge

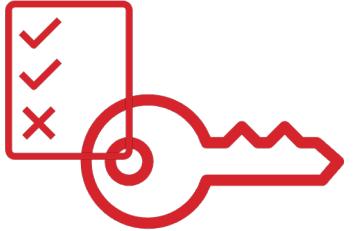


# KMS Multi Region Key Deletion

- **Deleting Replica Key**
  - Less risky, can always be re-created from the Primary Key (if it exists)
  - Must be scheduled (7 to 30 days)
- **Deleting Primary Key**
  - Cannot happen until all Replicas have been deleted
  - If you want to delete a Primary Key but keep Replicas, promote another one as Primary and then delete the “old Primary Key”
  - Must be scheduled (7 to 30 days after the replicas are deleted)



# KMS Key Policies



- Control access to KMS keys, “similar” to S3 bucket policies
- Difference: you cannot control access without them
- **Default KMS Key Policy:**
  - Created if you don't provide a specific KMS Key Policy
  - Complete access to the key to the root user = entire AWS account
- **Custom KMS Key Policy:**
  - Define users, roles that can access the KMS key
  - Define who can administer the key
  - Useful for cross-account access of your KMS key

# Default KMS Key Policy

- It gives the AWS account that owns the KMS key, full access to the KMS key
- KMS Key Policy does NOT automatically give permission to the account or any of its users
- Allows the account to use IAM policies to allow access to the KMS key, in addition to the key policy

## Default KMS Key Policy

```
{  
  "Effect": "Allow",  
  "Action": "kms:*",  
  "Principal": {  
    "AWS": "arn:aws:iam::123456789012:root"  
  },  
  "Resource": "*"  
}
```



# KMS Key Policies

- AWS Owned Keys
  - You cannot view or change the Key Policy
- AWS Managed Keys (e.g., aws/ebs)
  - You can view the Key Policy
  - You cannot change the Key Policy
- AWS Customer Managed Keys
  - You can view the Key Policy
  - You can edit the Key Policy

## Key Policy Comparison

	AWS Owned Keys	AWS Managed Keys	AWS Customer Managed Keys
View	✗	✓	✓
Edit	✗	✗	✓

```
{
  "Version": "2012-10-17",
  "Id": "auto-ebs-2",
  "Statement": [
    {
      "Sid": "Allow access through EBS for all principals in the account that are authorized to use EBS",
      "Effect": "Allow",
      "Principal": {
        "AWS": "*"
      },
      "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms>CreateGrant",
        "kms:DescribeKey"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "kms:CallerAccount": "123456789012",
          "kms:ViaService": "ec2.us-east-1.amazonaws.com"
        }
      }
    },
    {
      "Sid": "Allow direct access to key metadata to the account",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:root"
      },
      "Action": [
        "kms:Describe*",
        "kms:Get*",
        "kms>List*",
        "kms:RevokeGrant"
      ],
      "Resource": "*"
    }
  ]
}
```

Key Policy for AWS  
Managed Key (aws/ebs)

# Custom KMS Key Policy – Allow Admins

- KMS Key administrators have permissions to manage the KMS key
- KMS Key administrators cannot use the KMS Key Cryptographic Operations (Encrypt / Decrypt...)
- You can add IAM Users / Roles as KMS Key administrators

```
{  
  "Sid": "Allow access for Key Administrators",  
  "Effect": "Allow",  
  "Principal": {  
    "AWS": [  
      "arn:aws:iam::123456789012:user/KMSAdminUser",  
      "arn:aws:iam::123456789012:role/KMSAdminRole"  
    ]  
  },  
  "Action": [  
    "kms>Create*",  
    "kms>Describe*",  
    "kms>Enable*",  
    "kms>List*",  
    "kms>Put*",  
    "kms>Update*",  
    "kms>Revoke*",  
    "kms>Disable*",  
    "kms>Get*",  
    "kms>Delete*",  
    "kms>TagResource",  
    "kms>UntagResource",  
    "kms>ScheduleKeyDeletion",  
    "kms>CancelKeyDeletion"  
  ],  
  "Resource": "*"  
}
```

# Custom KMS Key Policy – Allow Users to Directly Use the KMS Key

- Allows IAM Users / Roles to use the KMS Key directly
- IAM Users / Roles don't need IAM Policies if the KMS Key is in the same account
  - The KMS Key explicitly authorizes the IAM Principal
- Alternative is Default KMS Key + IAM Policy

```
{  
  "Sid": "Allow use of the key",  
  "Effect": "Allow",  
  "Principal": {  
    "AWS": [  
      "arn:aws:iam::111122223333:user/ExampleUser",  
      "arn:aws:iam::111122223333:role/ExampleRole",  
      "arn:aws:iam::444455566666:root"  
    ]  
  },  
  "Action": [  
    "kms:Encrypt",  
    "kms:Decrypt",  
    "kms:ReEncrypt*",  
    "kms:GenerateDataKey*",  
    "kms:DescribeKey"  
  ],  
  "Resource": "*"  
}
```



# KMS Grants

- Allows **you** to grant access to specific AWS KMS keys to other AWS accounts and IAM Users / Roles within your AWS account
- Often used for temporary permissions
- Can be created for a variety of operations, including encrypt, decrypt, sign, and verify, as well as creating more grants
- Grants are for one KMS Key only, and one or more IAM Principal
- Once granted, a principal can perform any operation as specified in the Grant
- Grants do NOT expire automatically, you must delete them manually
- You don't need to change KMS Key Policy or IAM Policy

# Creating a KMS Key Grant

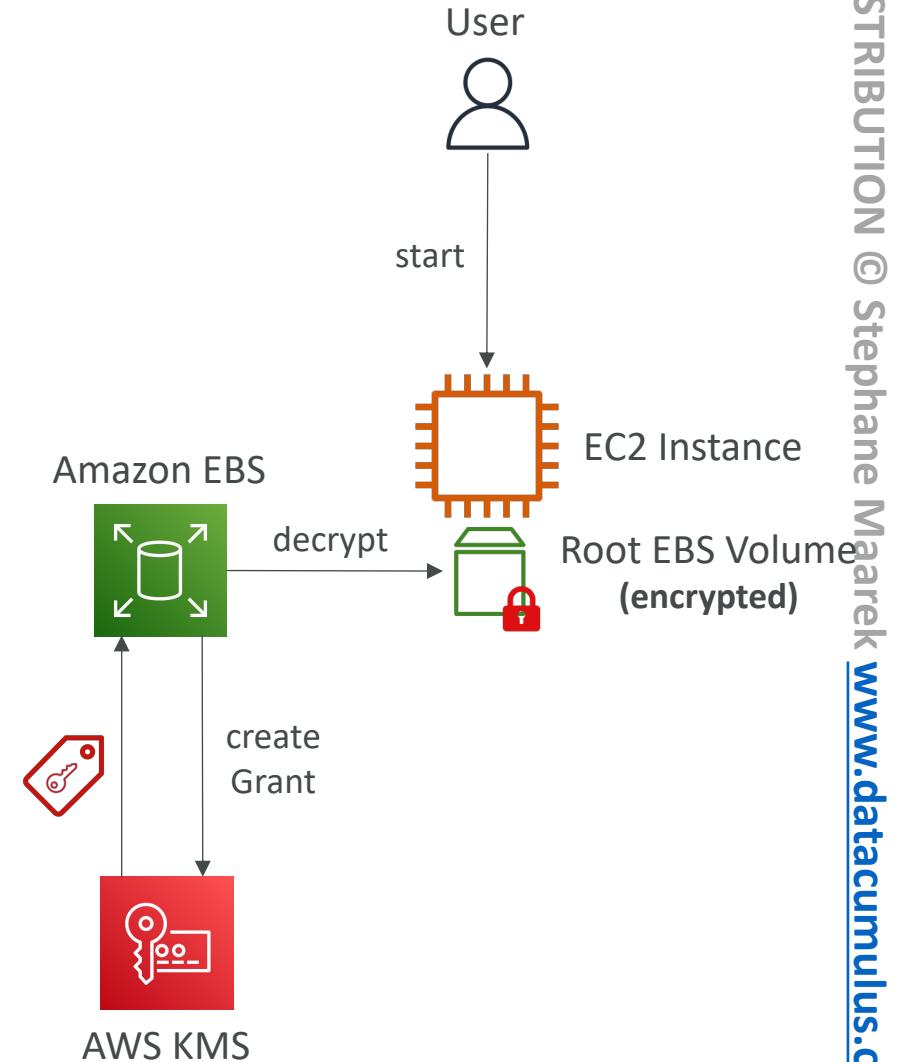
- Using AWS CLI

```
$ aws kms create-grant --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
  --grantee-principal arn:aws:iam::123456789012:user/ExampleUser \
  --operations Decrypt \
  --retiring-principal arn:aws:iam::123456789012:role/ExampleRole \
  --constraints EncryptionContextSubset={Department=IT}
```

- Make sure to delete a KMS Key Grant when you're done using it
- For now, there's no support in the AWS Console

# KMS Grants – AWS Service Usage

- Grants are commonly used by AWS services that integrate with AWS KMS to encrypt your data at rest
- The AWS service creates a **Grant on behalf of a user in the account**, uses its permissions, and retires the grant as soon as its task is complete
- Users must have the **CreateGrant** IAM permission



# Custom KMS Key Policy – Grants for AWS Services Use Cases

- Use this KMS Key Policy with:
  - Amazon EBS and Amazon EC2 to attach an encrypted EBS volume to an EC2 instance
  - Amazon Redshift to launch an encrypted cluster
  - AWS services integrated with AWS KMS that use grants to create, manage, or use encrypted resources with those services

```
{  
  "Sid": "Allow attachment of persistent resources",  
  "Effect": "Allow",  
  "Principal": {  
    "AWS": "arn:aws:iam::123456789012:user/ExampleUser"  
  },  
  "Action": [  
    "kms>CreateGrant",  
    "kms>ListGrants",  
    "kms>RevokeGrant"  
  ],  
  "Resource": "*",  
  "Condition": {  
    "Bool": {  
      "kms>GrantIsForAWSResource": true  
    }  
  }  
}
```

KMS Key Policy

# Troubleshooting: Can't Start an EC2 Instance with Encrypted EBS Volume

- Reasons:

- KMS key might be disabled or deleted
- EBS service might not have the required permissions to use KMS key

- Resolution:

- Make sure the KMS key is exists and enabled
- Make sure that EBS service has the require permissions to create KMS Grants in behalf of the specified principal
- **kms:GrantIsForAWSResource** – allows AWS services to create Grants (e.g., EBS)

```
{  
    "Effect": "Allow",  
    "Principal": {  
        "AWS": "arn:aws:iam::123456789012:user/ExampleUser"  
    },  
    "Action": "kms>CreateGrant",  
    "Resource": "*",  
    "Condition": {  
        "Bool": {  
            "kms:GrantIsForAWSResource": true  
        }  
    }  
}
```

# Condition Keys – kms:ViaService

- **kms:ViaService** – limits the use of a KMS key to requests from specified AWS services
- Example: the following key policy allow usage of the KMS Key through EC2 or RDS in us-west-2 on behalf of the ExampleUser
- IAM user must be authorized to use the KMS Key and Grant it to the AWS service
- Can be used with AWS Managed Keys (e.g., aws/ebs)

```
{  
    "Effect": "Allow",  
    "Principal": {  
        "AWS": "arn:aws:iam::123456789012:user/ExampleUser"  
    },  
    "Action": [  
        "kms:Encrypt",  
        "kms:Decrypt",  
        "kms:ReEncrypt*",  
        "kms:GenerateDataKey*",  
        "kms>CreateGrant",  
        "kms>ListGrants",  
        "kms:DescribeKey"  
    ],  
    "Resource": "*",  
    "Condition": {  
        "StringEquals": {  
            "kms:ViaService": [  
                "ec2.us-west-2.amazonaws.com",  
                "rds.us-west-2.amazonaws.com"  
            ]  
        }  
    }  
}
```

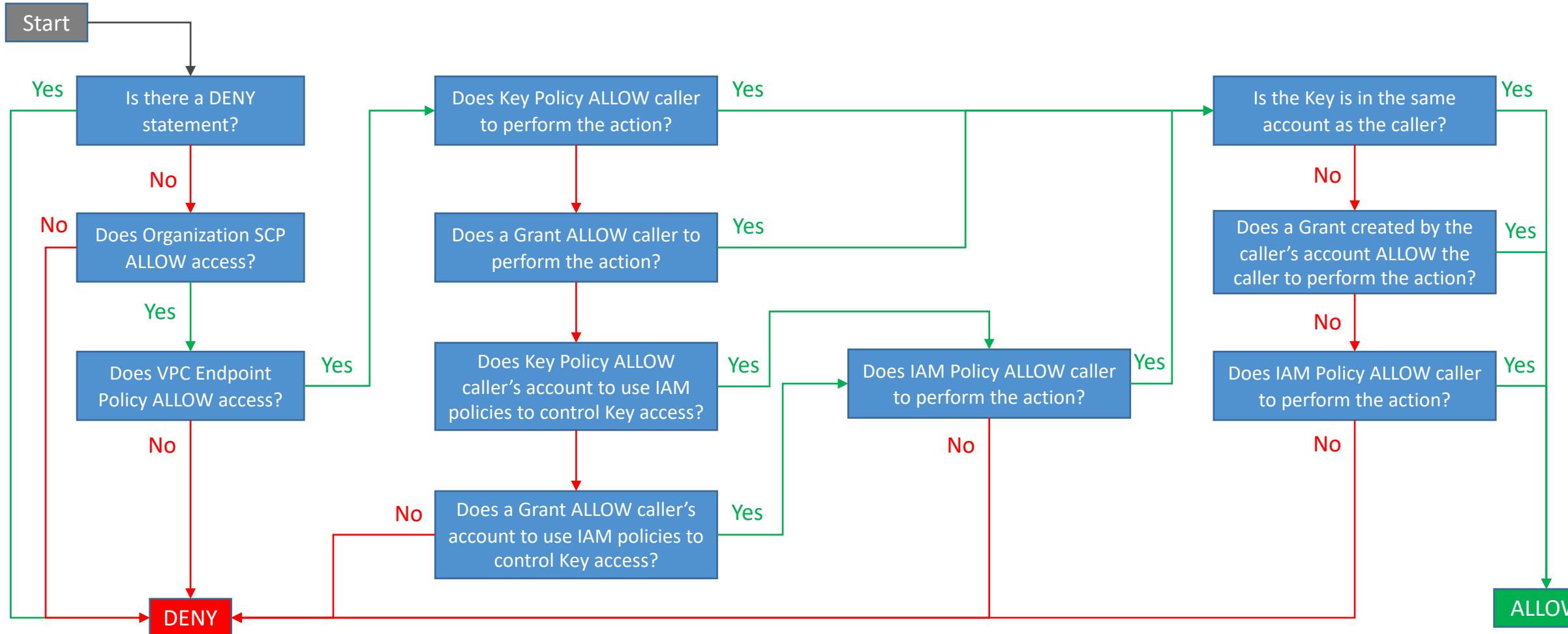
# Condition Keys – kms:CallerAccount

- kms:CallerAccount – Allow or deny access to all identities (IAM users and roles) in an AWS account
- Example: the following KMS Key policy is the Key policy for AWS Managed Key for Amazon EBS (aws/ebs)

```
{  
    "Effect": "Allow",  
    "Principal": {  
        "AWS": "*"  
    },  
    "Action": [  
        "kms:Encrypt",  
        "kms:Decrypt",  
        "kms:ReEncrypt*",  
        "kms:GenerateDataKey*",  
        "kms>CreateGrant",  
        "kms:DescribeKey"  
    ],  
    "Resource": "*",  
    "Condition": {  
        "StringEquals": {  
            "kms:CallerAccount": "123456789012",  
            "kms:ViaService": "ec2.us-west-2.amazonaws.com"  
        }  
    }  
}
```

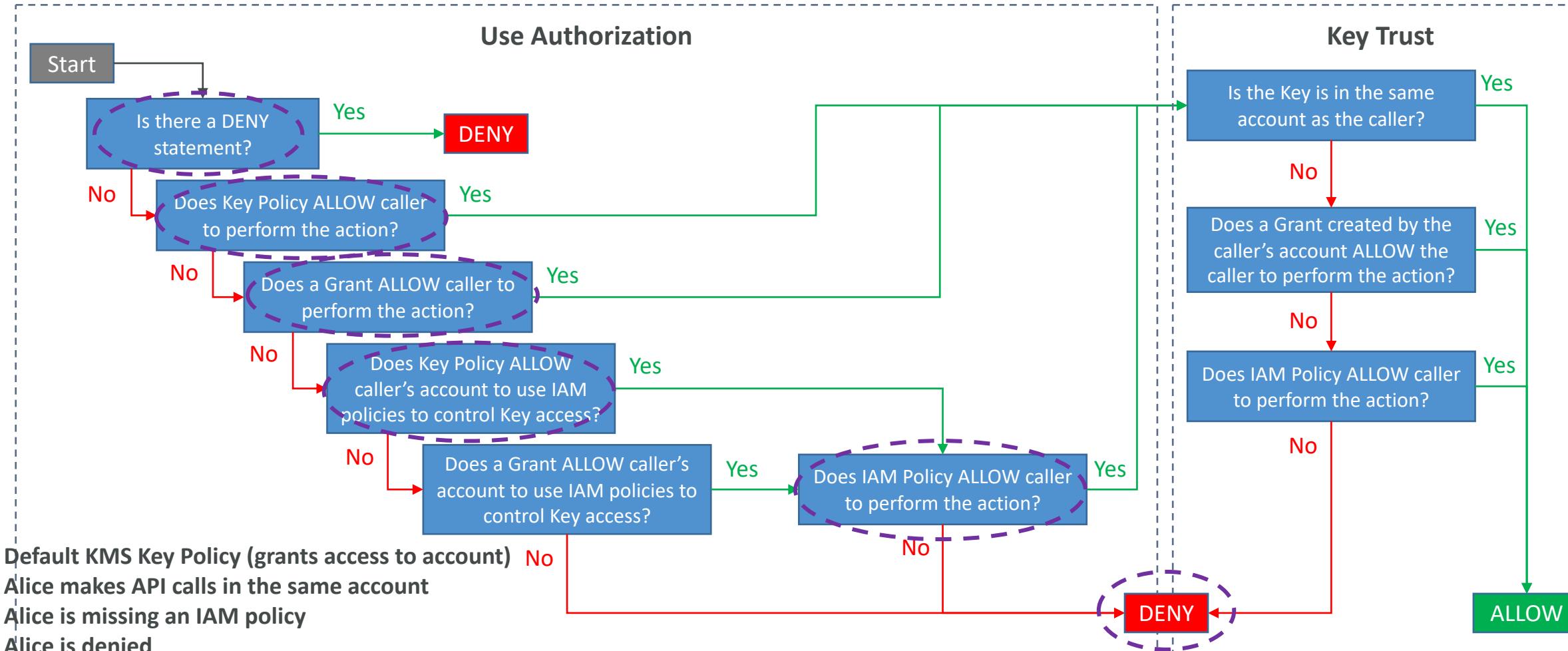
AWS Managed Key Policy  
for Amazon EBS (aws/ebs)

# KMS Key Authorization Process

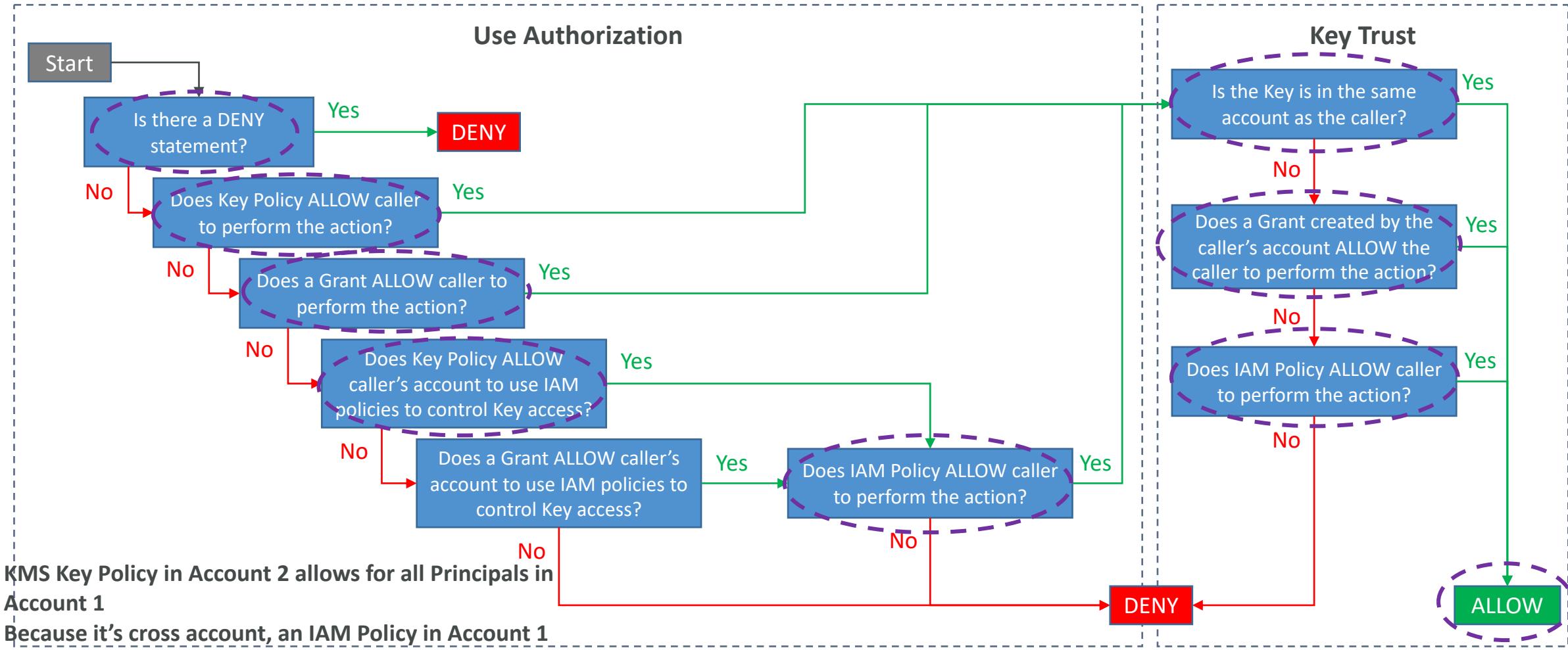


- Key Policy or Key Grants must have explicit Allow
- IAM Policies, SCP, and VPC Endpoint Policies

# KMS Key Authorization Process – Example I



# KMS Key Authorization Process – Example 2

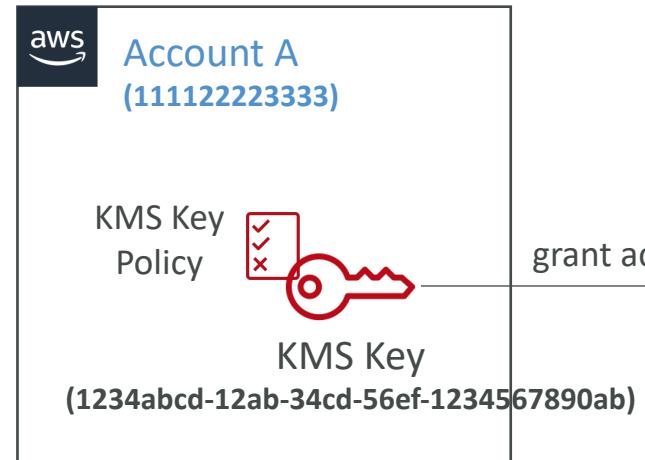


- KMS Key Policy in Account 2 allows for all Principals in Account 1
- Because it's cross account, an IAM Policy in Account 1 must be used to authorise usage of KMS Key in Account 2

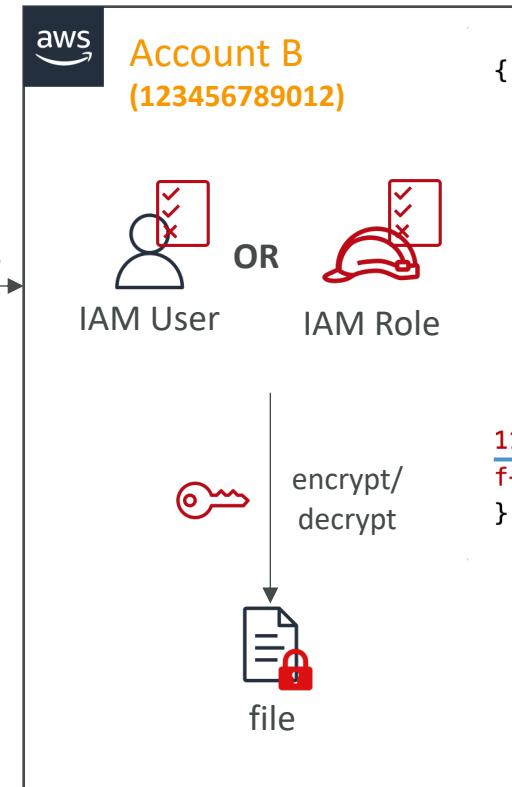
# KMS Key – Cross-Account Access

```
{  
  "Effect": "Allow",  
  "Principal": {  
    "AWS": "arn:aws:iam::123456789012:root"  
  },  
  "Action": [  
    "kms:Encrypt",  
    "kms:Decrypt",  
    "kms:ReEncrypt*",  
    "kms:GenerateDataKey*",  
    "kms:DescribeKey"  
  ],  
  "Resource": "*"  
}
```

**KMS Key Policy**



grant access



```
{  
  "Effect": "Allow",  
  "Action": [  
    "kms:Encrypt",  
    "kms:Decrypt",  
    "kms:ReEncrypt*",  
    "kms:GenerateDataKey*",  
    "kms:DescribeKey"  
  ],  
  "Resource": "arn:aws:kms:us-west-2:  
  111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"  
}
```

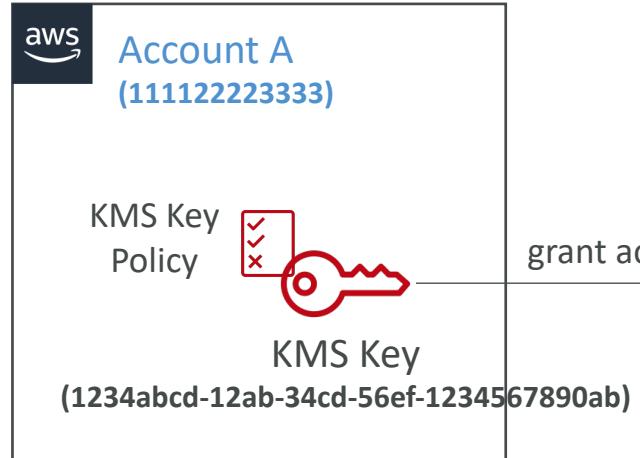
**IAM Policy**

# KMS Key – Cross-Account Access

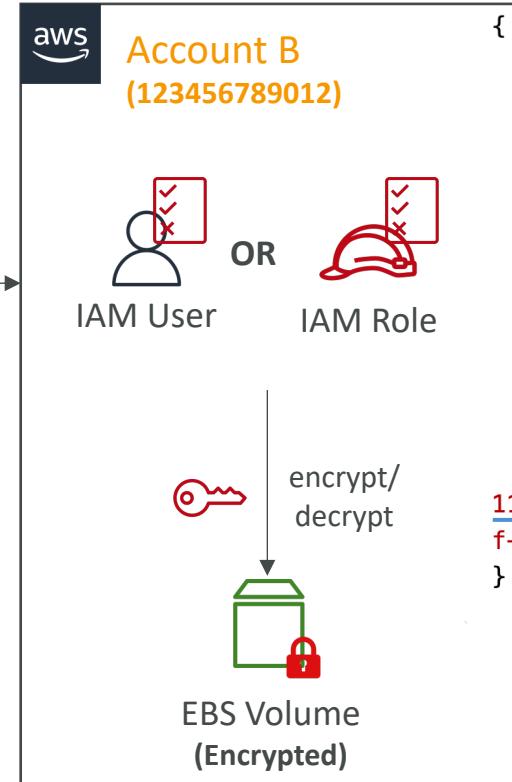
## Usage of External KMS Keys with AWS Services

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::123456789012:root"
  },
  "Action": [
    "kms:Encrypt",
    "kms:Decrypt",
    "kms:ReEncrypt*",
    "kms:GenerateDataKey*",
    "kms:DescribeKey",
    "kms>CreateGrant",
    "kms>ListGrants",
    "kms:RevokeGrant"
  ],
  "Resource": "*"
}
```

**KMS Key Policy**



grant access

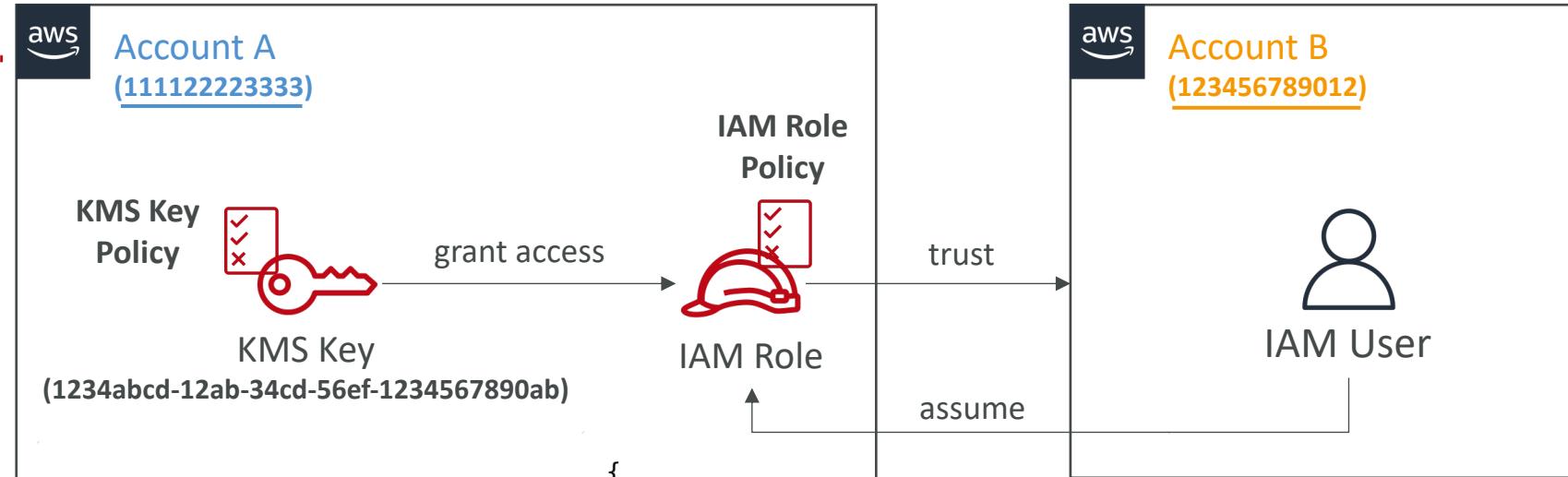


```
{
  "Effect": "Allow",
  "Action": [
    "kms:Encrypt",
    "kms:Decrypt",
    "kms:ReEncrypt*",
    "kms:GenerateDataKey*",
    "kms:DescribeKey",
    "kms>CreateGrant",
    "kms>ListGrants",
    "kms:RevokeGrant"
  ],
  "Resource": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
}
```

**IAM Policy**

# KMS Key – Cross-Account Access Through assuming an IAM Role

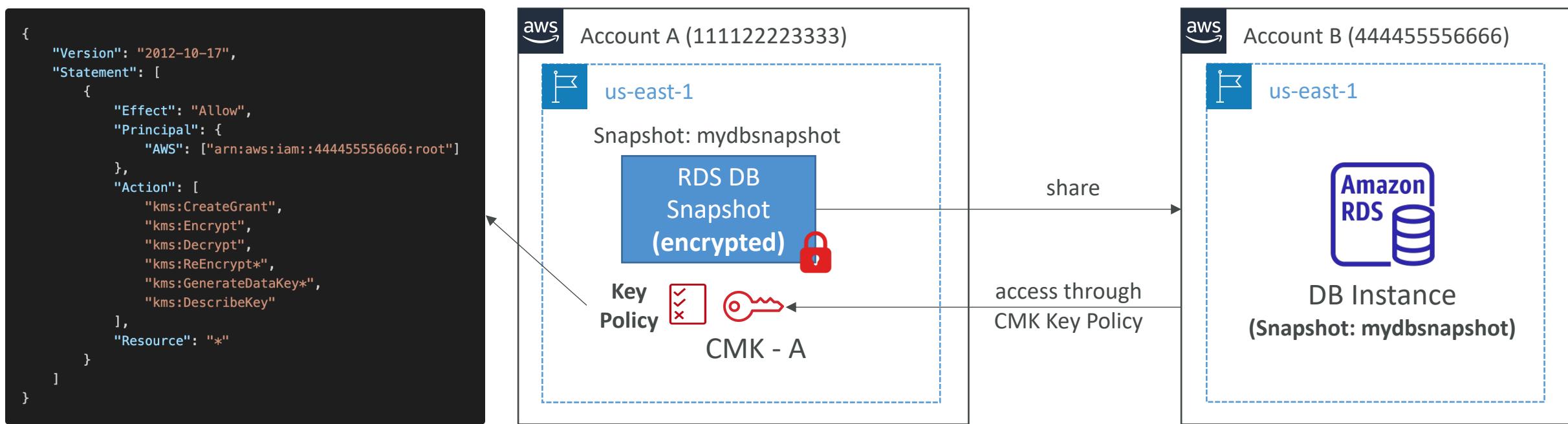
```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:root"
  },
  "Action": [
    "kms:Encrypt",
    "kms:Decrypt",
    "kms:ReEncrypt*",
    "kms:GenerateDataKey*",
    "kms:DescribeKey"
  ],
  "Resource": "*"
}
KMS Key Policy
```



```
"Effect": "Allow",
"Action": [
  "kms:Encrypt",
  "kms:Decrypt",
  "kms:ReEncrypt*",
  "kms:GenerateDataKey*",
  "kms:DescribeKey"
],
"Resource": "arn:aws:kms:us-west-2:
111122223333:key/1234abcd-12ab-34cd-56e
f-1234567890ab"
}
```

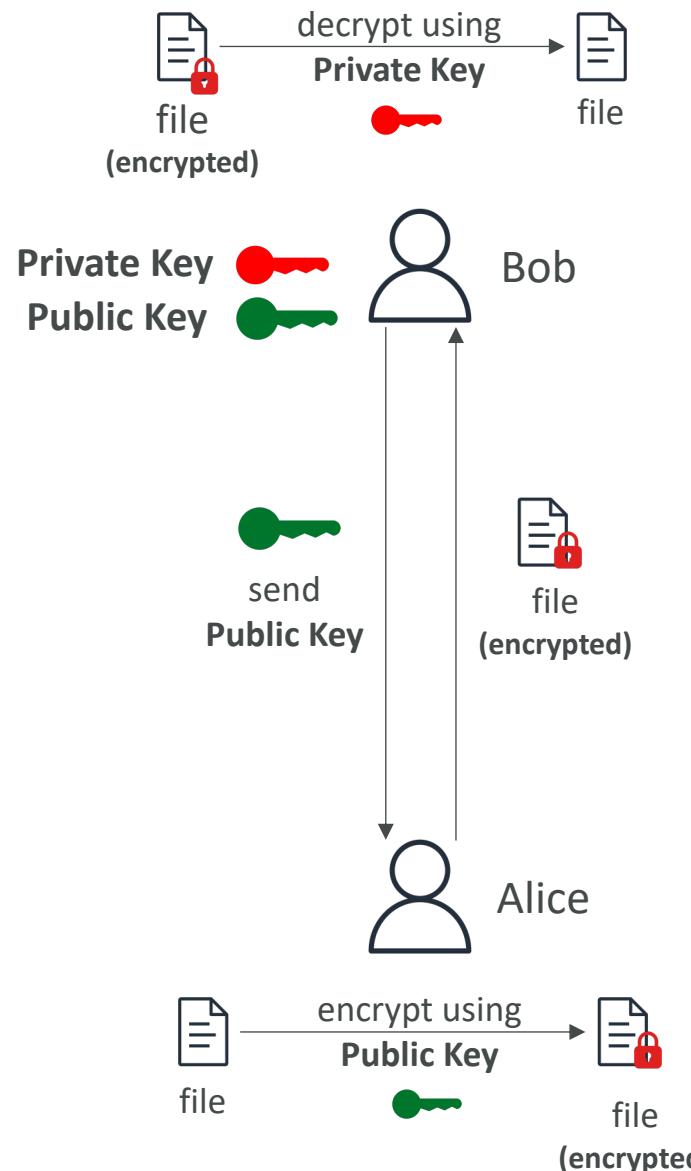
# Sharing KMS Encrypted RDS DB Snapshots

- You can share RDS DB snapshots encrypted with KMS CMK with other accounts, but must first share the KMS CMK with the target account using Key Policy



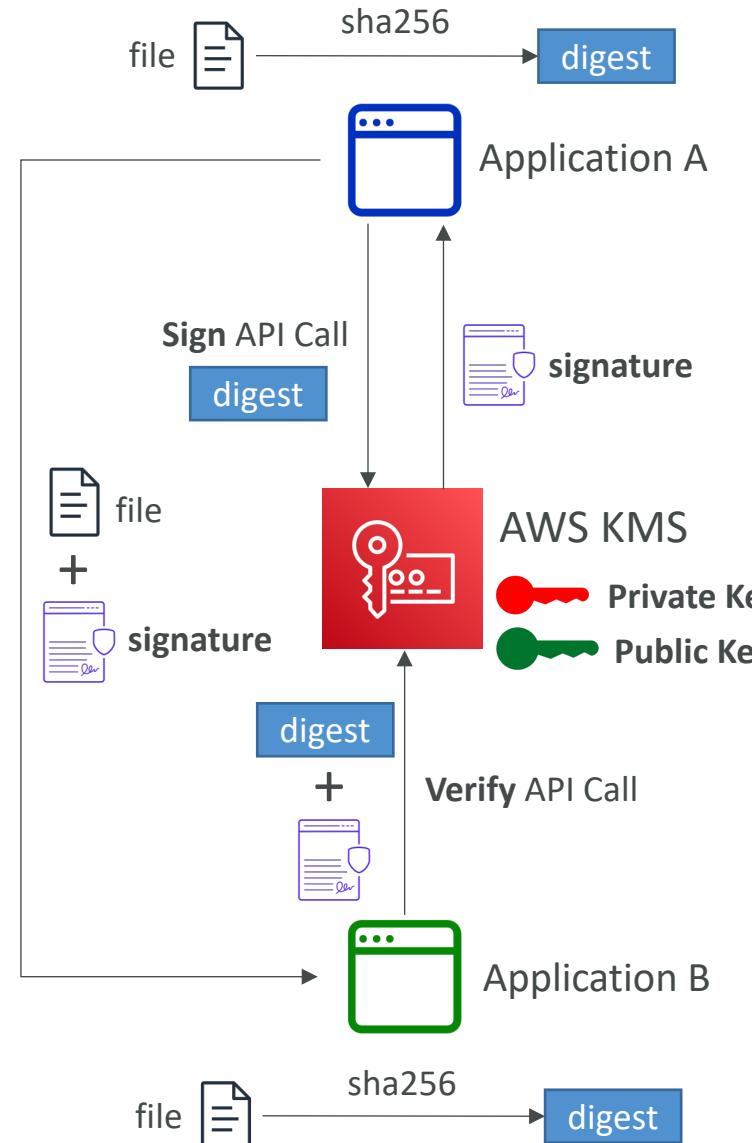
# Asymmetric Encryption

- An encryption process that uses a pair of related keys (public & private) to encrypt and decrypt data
- Public Key can be shared, Private Key must be kept secret
- KMS supports 3 types of asymmetric KMS keys:
  - RSA KMS Keys – encryption/decryption or signing/verification
  - Elliptic Curve (ECC) KMS Keys – signing and verification
  - SM2 KMS Keys (China Regions only) – encryption/decryption or signing/verification
- Private Keys never leaves AWS KMS unencrypted



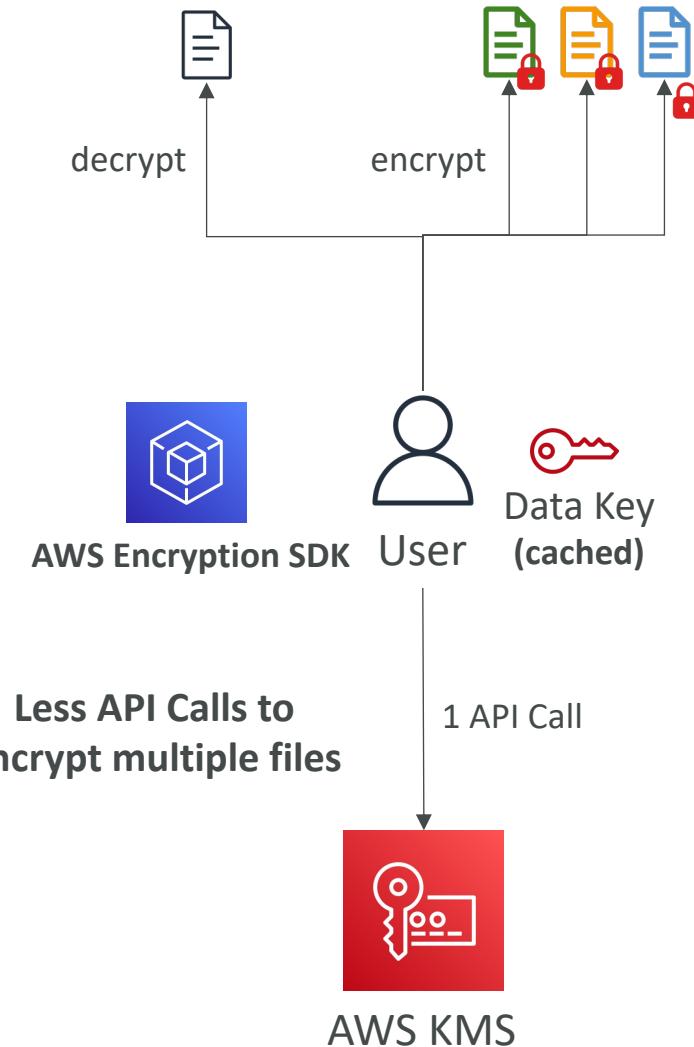
# Digital Signing with KMS Asymmetric

- Helps you verify the integrity of messages or files sent across different systems
- Verify that file has not been tampered with in transit
- Public Key used to verify the signature while Private Key used in the signing process
- Use cases: document e-signing, secure messaging, authentication/authorization tokens, trusted source code, e-commerce transactions, ...



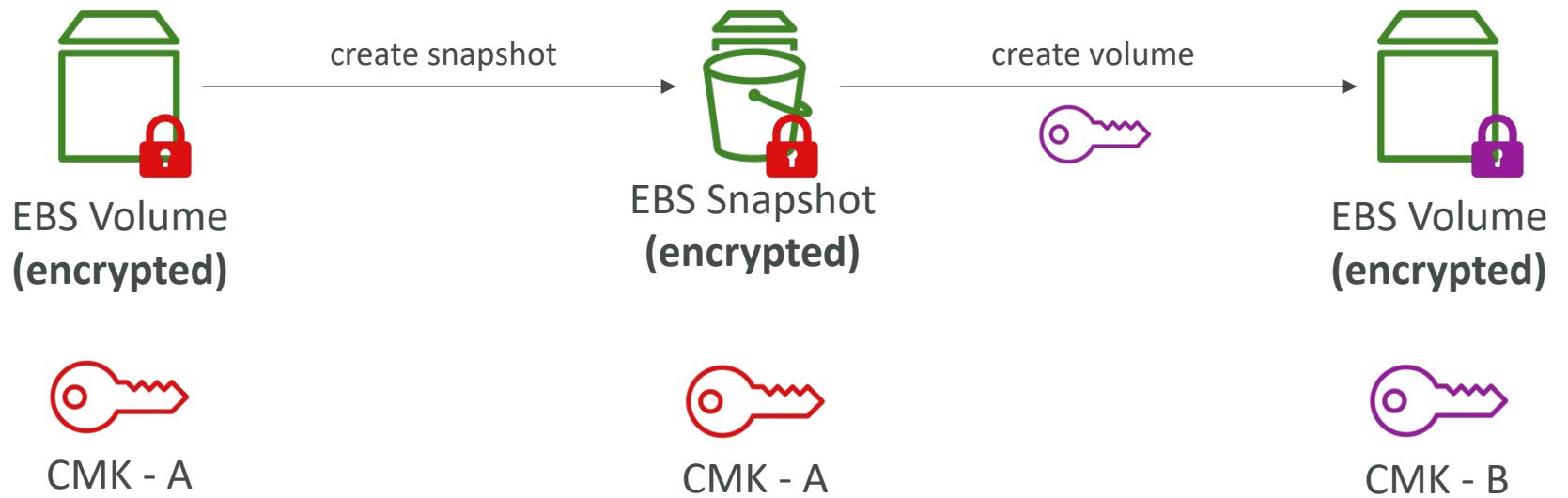
# KMS API Calls Limits & Data Key Caching

- When your application makes multiple API calls to KMS and you hit service limits (requests per second limit) ...
- **Data Key Caching** allows you to reuse data keys that protect your data
- Instead of generating a new data key for each encryption operation
- Reduce latency, improve throughput, reduce cost, stay within service limits, ...
- Implemented using AWS Encryption SDK
- Note: encryption best practices discourages reuse of data keys (tradeoff cost / security)

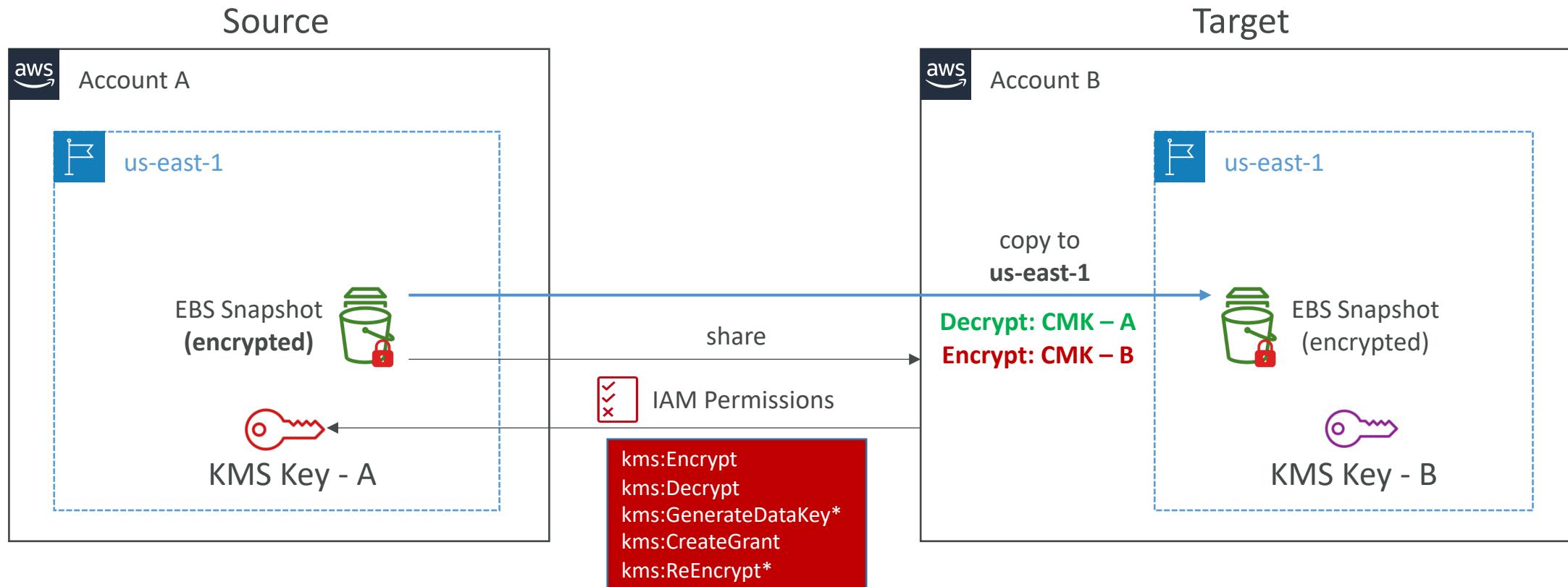


# Changing The KMS Key For An Encrypted EBS Volume

- You can't change the encryption keys used by an EBS volume
- Create an EBS snapshot and create a new EBS volume and specify the new KMS key



# Automate Cross-Account EBS KMS-Encrypted Snapshot Copies

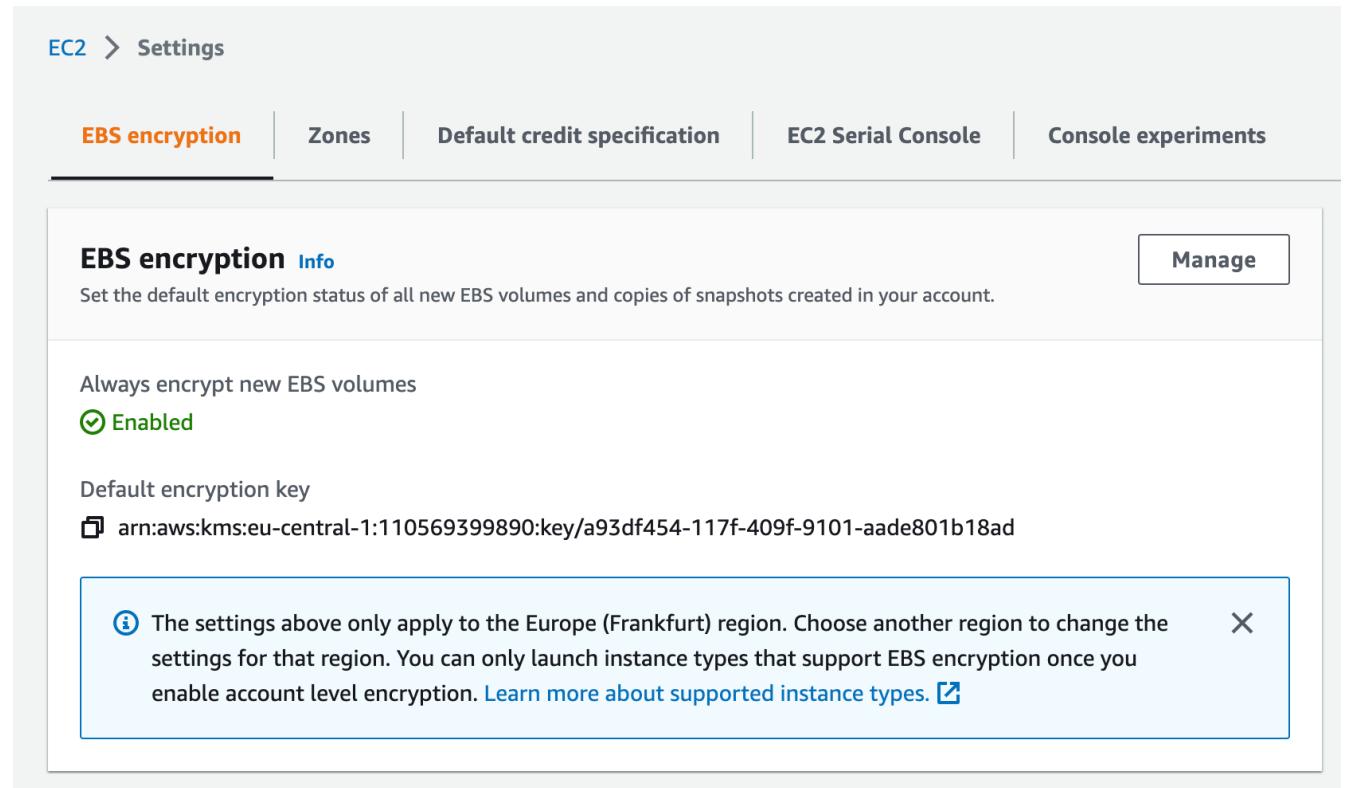


# KMS Key Policy in Target Account

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "kms:RevokeGrant",  
                "kms>CreateGrant",  
                "kms>ListGrants"  
            ],  
            "Resource": [  
                "arn:aws:kms:us-east-1:111111111111:key/  
1234abcd-12ab-34cd-56ef-1234567890ab",  
                "arn:aws:kms:us-east-1:222222222222:key/  
4567dcba-23ab-34cd-56ef-0987654321yz"  
            ],  
            "Condition": {  
                "Bool": {  
                    "kms:GrantIsForAWSResource": "true"  
                }  
            }  
        },  
        {  
            "Effect": "Allow",  
            "Action": [  
                "kms:Encrypt",  
                "kms:Decrypt",  
                "kms:ReEncrypt*",  
                "kms:GenerateDataKey*",  
                "kms:DescribeKey"  
            ],  
            "Resource": [  
                "arn:aws:kms:us-east-1:111111111111:key/  
1234abcd-12ab-34cd-56ef-1234567890ab",  
                "arn:aws:kms:us-east-1:222222222222:key/  
4567dcba-23ab-34cd-56ef-0987654321yz"  
            ]  
        }  
    ]  
}
```

# EBS Encryption – Account level setting

- New Amazon EBS volumes aren't encrypted by default
- There's an account-level setting to encrypt automatically new EBS volumes and Snapshots
- This setting needs to be enabled on a per-region basis



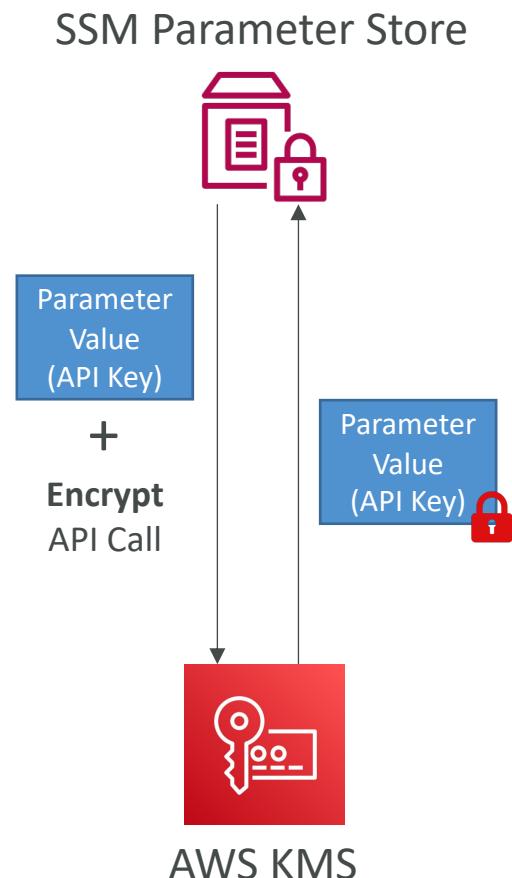
# ABAC with KMS

- Control access to your KMS Keys based on tags and aliases



# KMS with SSM Parameter Store

- SSM Parameter Store uses KMS to encrypt/decrypt parameter values of type **Secure String**
- Two types of Secure String Parameters:
  - Standard – all parameters encrypted using the same KMS key
  - Advanced – each parameter encrypted with a unique data key (Envelope Encryption)
- Specify the KMS key or use AWS Managed Key (`aws/ssm`)
- Works only with **Symmetric KMS Keys**
- Encryption process takes place in AWS KMS
- Note: you must have access to both the KMS key and the parameter in SSM Parameter Store



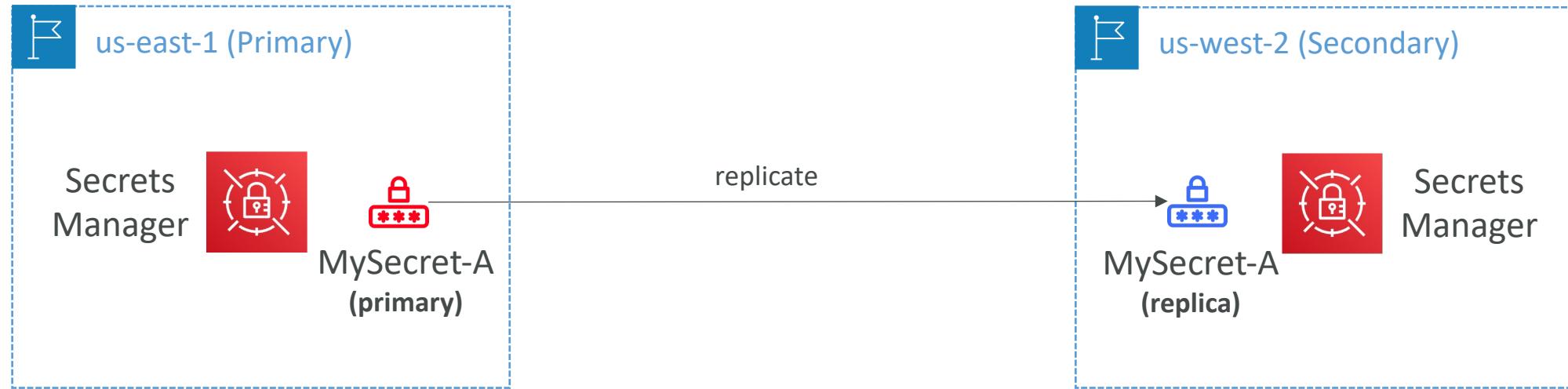
# AWS Secrets Manager



- Newer service, meant for storing secrets
- Capability to force **rotation of secrets** every X days
- Automate generation of secrets on rotation (uses Lambda)
- Integration with **Amazon RDS** (MySQL, PostgreSQL, Aurora)
- Secrets are encrypted using KMS
- Mostly meant for RDS integration

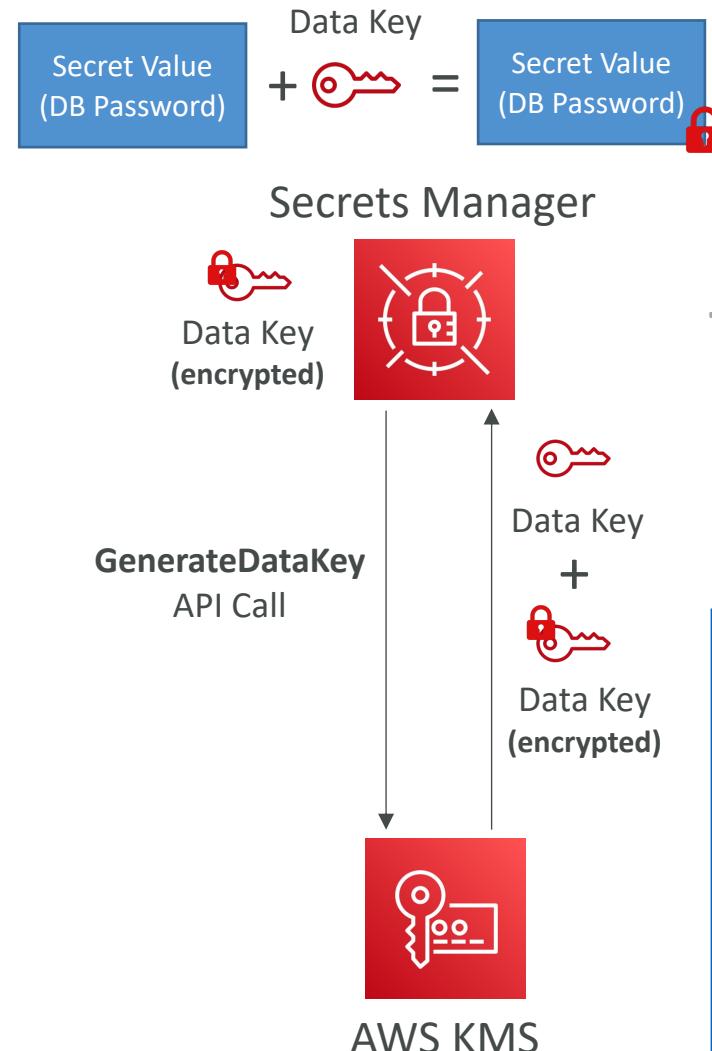
# AWS Secrets Manager – Multi-Region Secrets

- Replicate Secrets across multiple AWS Regions
- Secrets Manager keeps read replicas in sync with the primary Secret
- Ability to promote a read replica Secret to a standalone Secret
- Use cases: multi-region apps, disaster recovery strategies, multi-region DB...



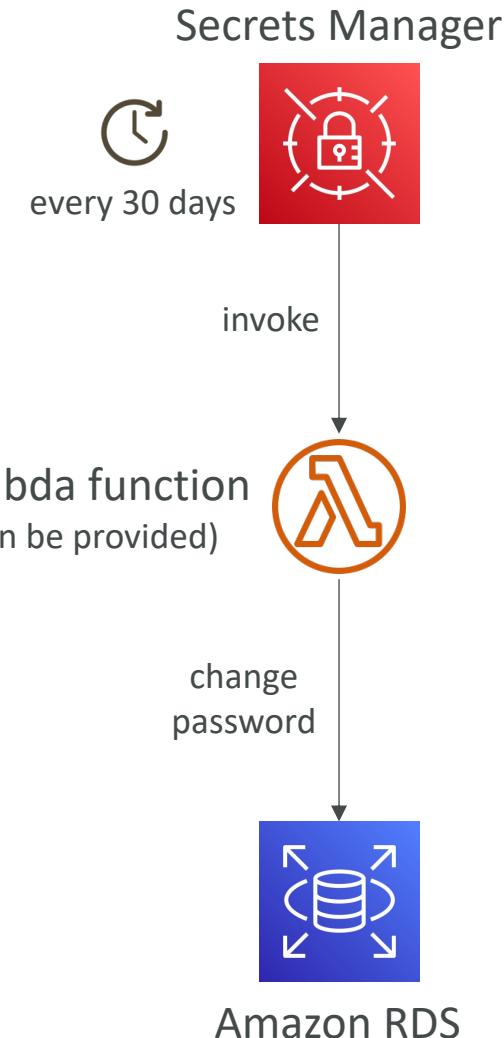
# KMS with Secrets Manager

- Secrets Manager uses KMS to encrypt/decrypt every version of every Secret value
- Each Secret value is encrypted with a unique data key (**Envelope Encryption**)
- Specify the KMS key or use AWS Managed Key (`aws/secretsmanager`)
- Works only with **Symmetric KMS Keys**
- Encryption process takes place in Secrets Manager
- Note: you must have access to both the KMS key and the Secret in Secrets Manager

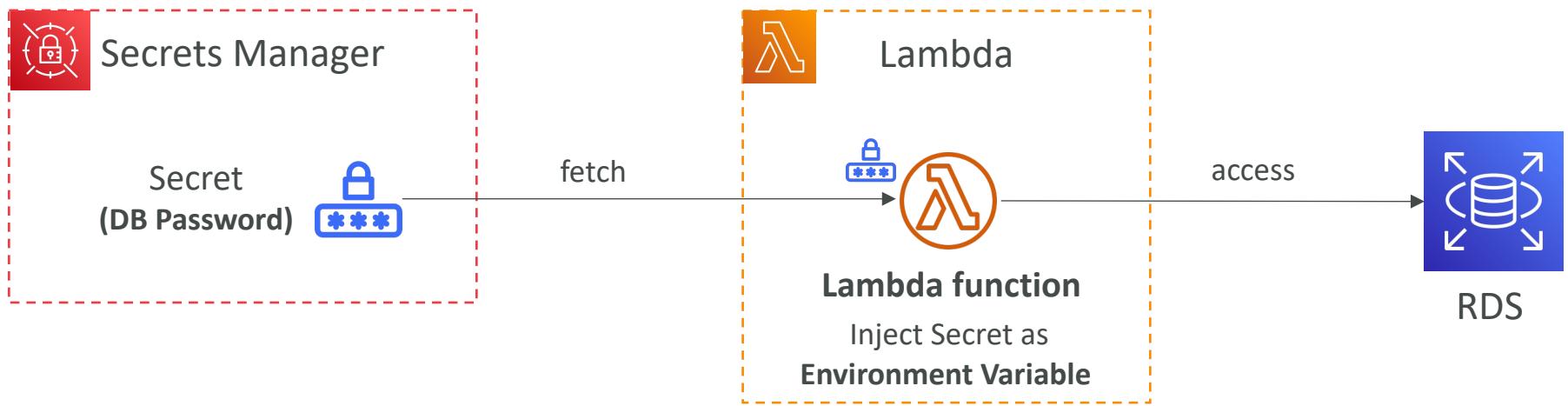


# Secrets Manager – Secrets Rotation

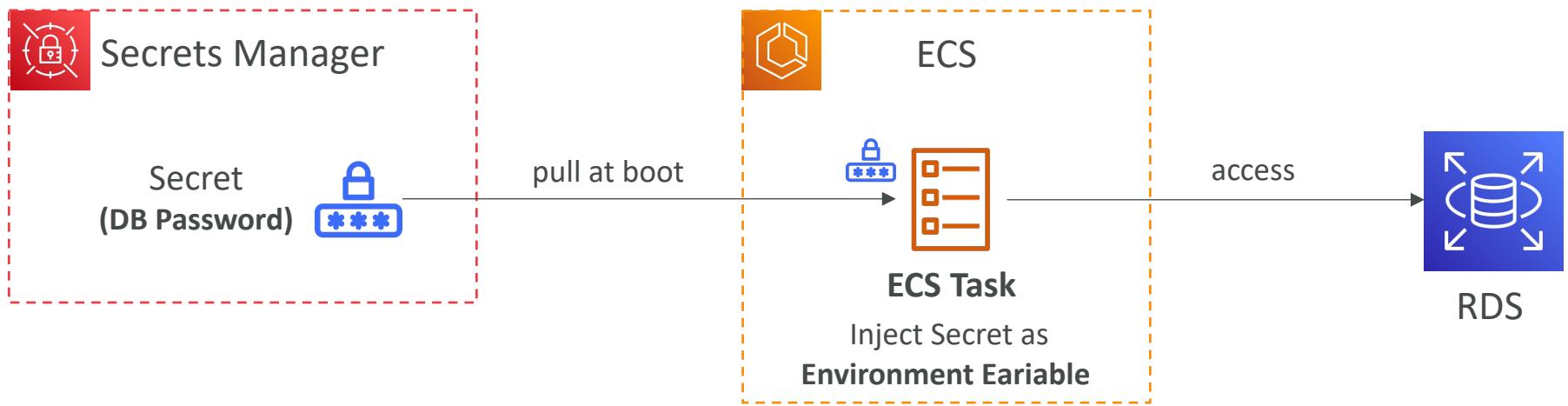
- Automatically and periodically updating a Secret
- Automated password rotation for integrated databases
- RDS, Redshift, DocumentDB, other databases...
- Credentials are changed in the Secret and the database
- Secrets Manager uses Lambda function to rotate Secrets
- When you Enable Secret Rotation, the Secret is rotated immediately
- Note: Lambda function must have access to both Secrets Manager and your database (if Lambda in VPC private subnet, use VPC Endpoints or NAT Gateway)



# Secrets Manager with Lambda integration



# Secrets Manager with ECS integration



# Secrets Manager – Resource Policy

- Specify who can access a Secret and what actions an IAM identity can perform
- Use cases:
  - Grant access to a single Secret for multiple users
  - Enforcing permissions (e.g., adding an explicit deny to a Secret)
  - Sharing a Secret between AWS accounts

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Action": "secretsmanager:*",  
      "Principal": {  
        "AWS": "arn:aws:iam::123456789012:user/Mary"  
      },  
      "Resource": "*"  
    }  
  ]  
}
```

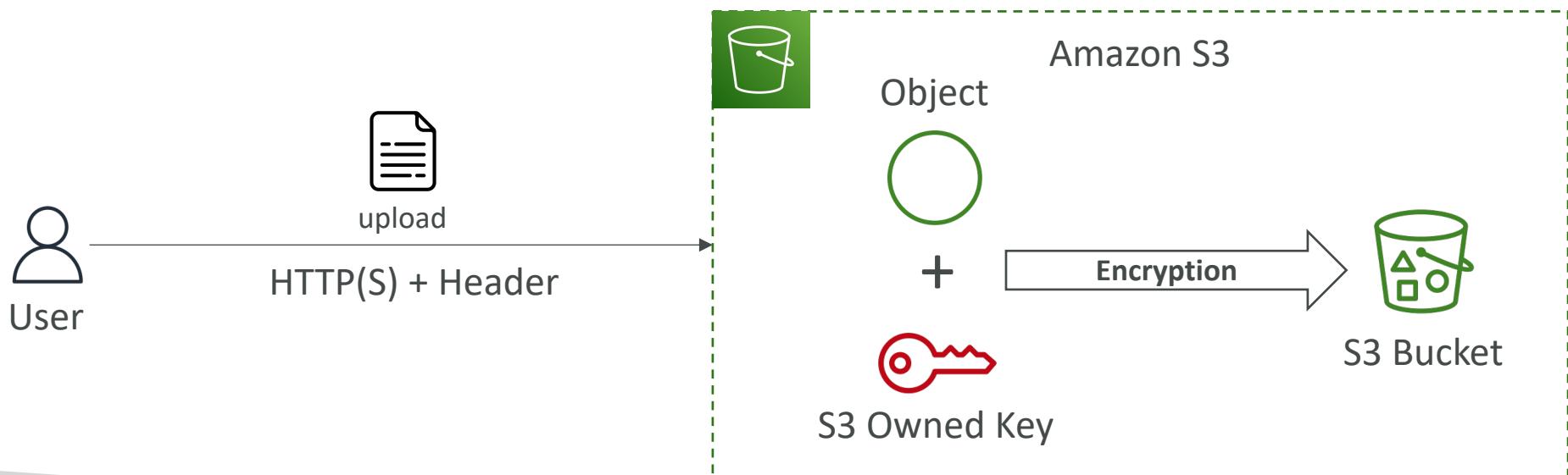


# Amazon S3 – Object Encryption

- You can encrypt objects in S3 buckets using one of 4 methods
- Server-Side Encryption (SSE)
  - Server-Side Encryption with Amazon S3-Managed Keys (SSE-S3) – Enabled by Default
    - Encrypts S3 objects using keys handled, managed, and owned by AWS
  - Server-Side Encryption with KMS Keys stored in AWS KMS (SSE-KMS)
    - Leverage AWS Key Management Service (AWS KMS) to manage encryption keys
  - Server-Side Encryption with Customer-Provided Keys (SSE-C)
    - When you want to manage your own encryption keys
- Client-Side Encryption
- It's important to understand which ones are for which situation for the exam

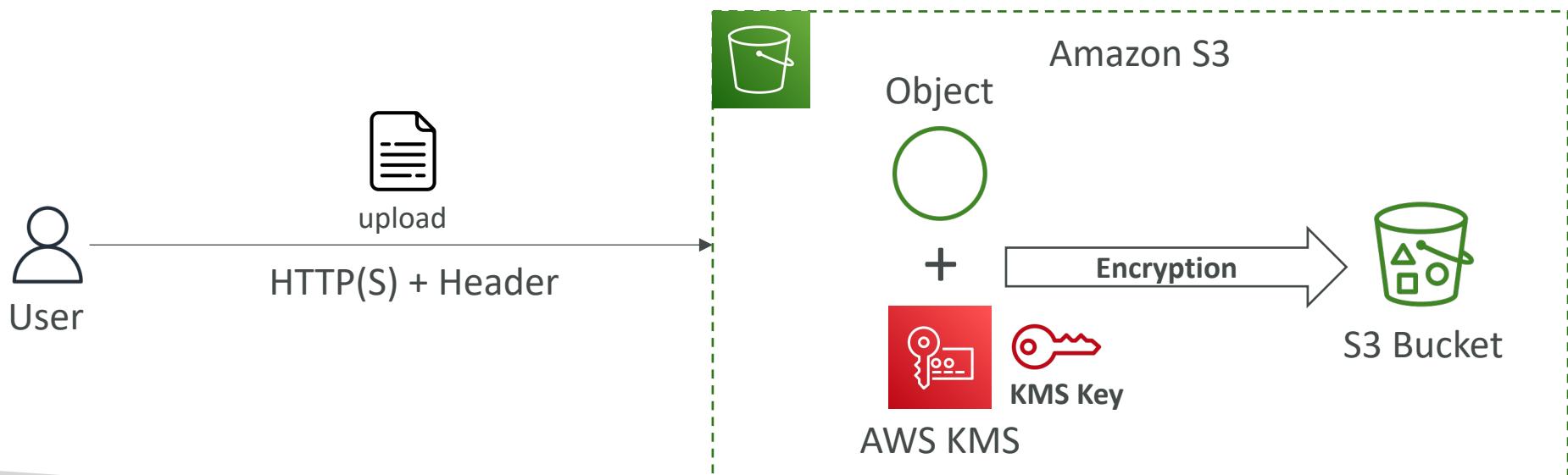
# Amazon S3 Encryption – SSE-S3

- Encryption using keys handled, managed, and owned by AWS
- Object is encrypted server-side
- Encryption type is AES-256
- Must set header "x-amz-server-side-encryption": "AES256"
- Enabled by default for new buckets & new objects



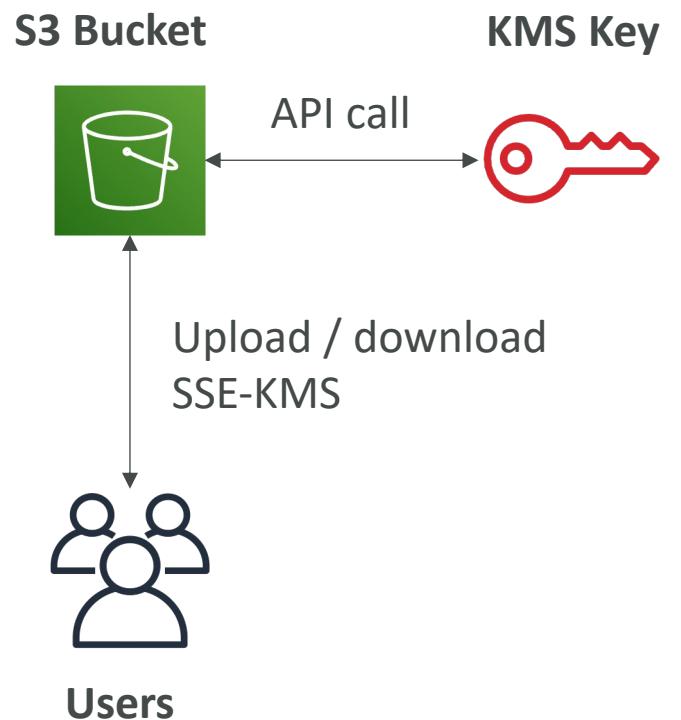
# Amazon S3 Encryption – SSE-KMS

- Encryption using keys handled and managed by AWS KMS (Key Management Service)
- KMS advantages: user control + audit key usage using CloudTrail
- Object is encrypted server side
- Must set header "x-amz-server-side-encryption": "aws:kms"



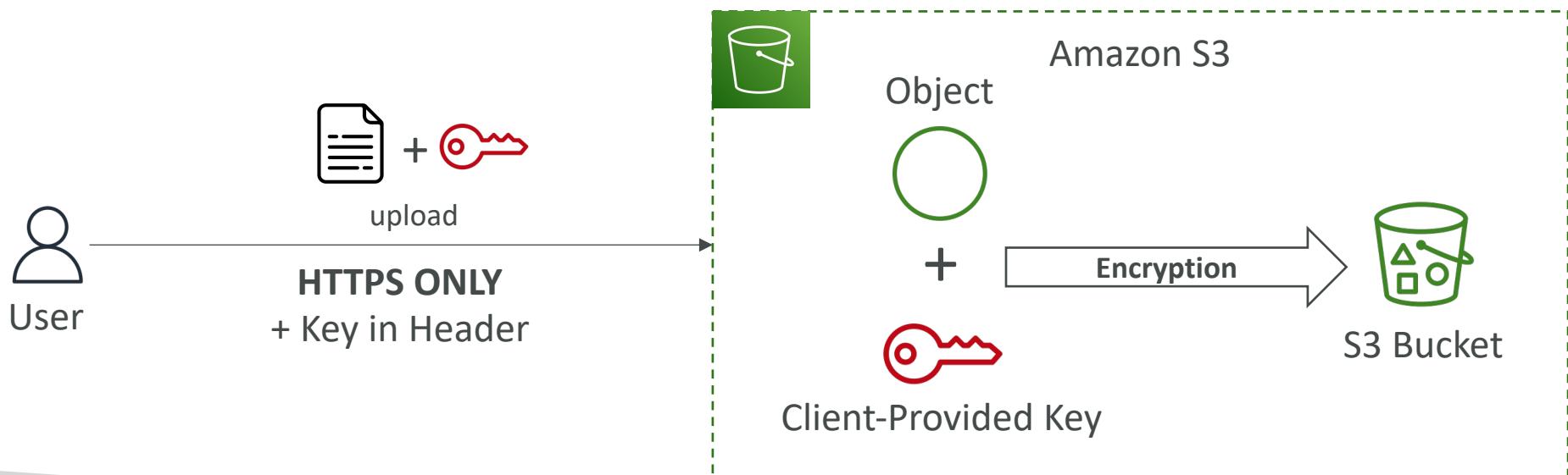
# SSE-KMS Limitation

- If you use SSE-KMS, you may be impacted by the KMS limits
- When you upload, it calls the **GenerateDataKey** KMS API
- When you download, it calls the **Decrypt** KMS API
- Count towards the KMS quota per second (5500, 10000, 30000 req/s based on region)
- You can request a quota increase using the Service Quotas Console



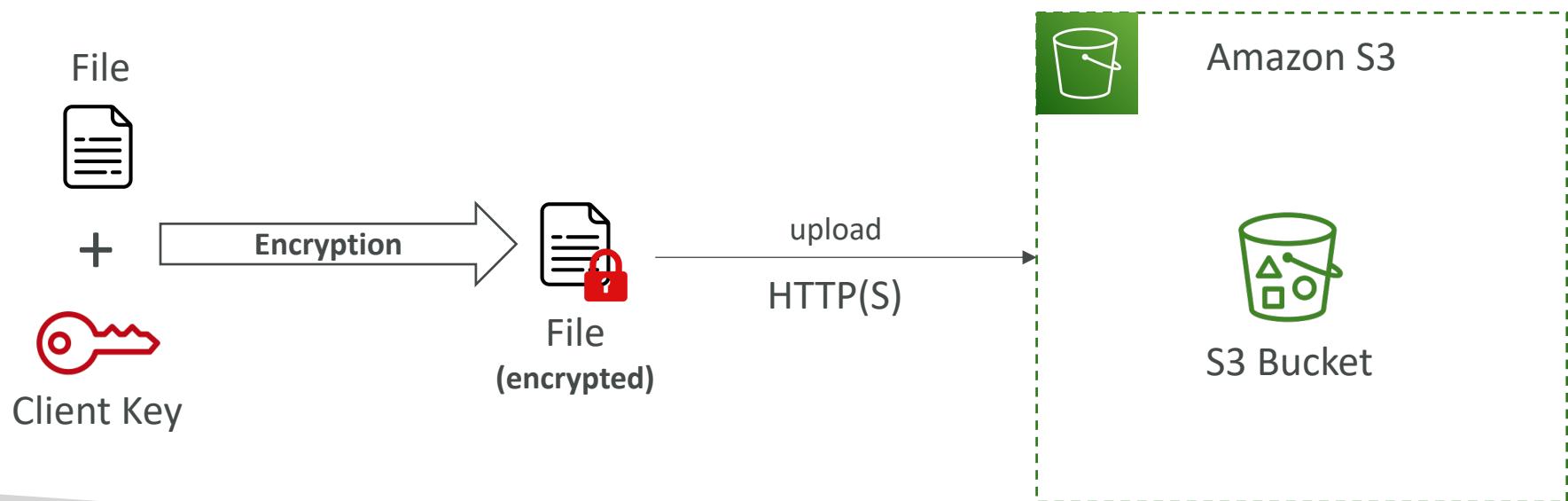
# Amazon S3 Encryption – SSE-C

- Server-Side Encryption using keys fully managed by the customer outside of AWS
- Amazon S3 does **NOT** store the encryption key you provide
- **HTTPS must be used**
- Encryption key must provided in HTTP headers, for every HTTP request made



# Amazon S3 Encryption – Client-Side Encryption

- Use client libraries such as [Amazon S3 Client-Side Encryption Library](#)
- Clients must encrypt data themselves before sending to Amazon S3
- Clients must decrypt data themselves when retrieving from Amazon S3
- Customer fully manages the keys and encryption cycle



# Amazon S3 – Encryption in transit (SSL/TLS)

- Encryption in flight is also called SSL/TLS
- Amazon S3 exposes two endpoints:
  - HTTP Endpoint – non encrypted
  - HTTPS Endpoint – encryption in flight
- HTTPS is recommended
- HTTPS is mandatory for SSE-C
- Most clients would use the HTTPS endpoint by default



# S3 Encryption for Objects

- SSE-S3: encrypts S3 objects using keys handled & managed by AWS
- SSE-KMS: leverage KMS to manage encryption keys
  - Key usage appears in CloudTrail
  - objects made public can never be read
  - On `s3:PutObject`, make the permission `kms:GenerateDataKey` is allowed
- SSE-C: when you want to manage your own encryption keys
- Client-Side Encryption
- Glacier: all data is AES-256 encrypted, key under AWS control

# Amazon S3 – Default Encryption vs. Bucket Policies

- SSE-S3 encryption is automatically applied to new objects stored in S3 bucket
- Optionally, you can “force encryption” using a bucket policy and refuse any API call to PUT an S3 object without encryption headers (SSE-KMS or SSE-C)

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Deny",  
            "Action": "s3:PutObject",  
            "Principal": "*",  
            "Resource": "arn:aws:s3:::my-bucket/*",  
            "Condition": {  
                "StringNotEquals": {  
                    "s3:x-amz-server-side-encryption": "aws:kms"  
                }  
            }  
        }  
    ]  
}
```

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Deny",  
            "Action": "s3:PutObject",  
            "Principal": "*",  
            "Resource": "arn:aws:s3:::my-bucket/*",  
            "Condition": {  
                "Null": {  
                    "s3:x-amz-server-side-encryption-customer-algorithm": "true"  
                }  
            }  
        }  
    ]  
}
```

- Note: Bucket Policies are evaluated before “Default Encryption”

# S3 Bucket Policies – Force HTTPS

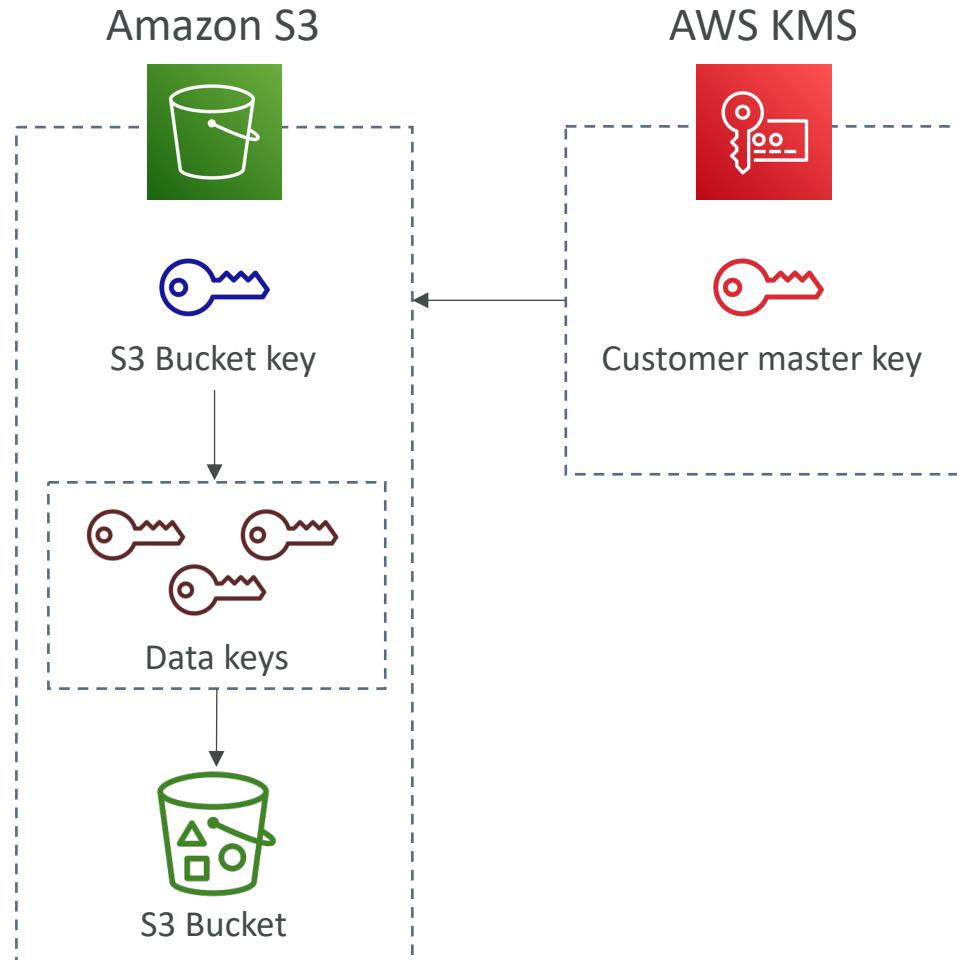
```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Deny",  
            "Action": "s3:*",  
            "Principal": "*",  
            "Resource": [  
                "arn:aws:s3:::my-bucket",  
                "arn:aws:s3:::my-bucket/*"  
            ],  
            "Condition": {  
                "Bool": {  
                    "aws:SecureTransport": "false"  
                }  
            }  
        }  
    ]  
}
```

# S3 Bucket Policies – Force SSE-KMS

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Deny",  
      "Principal": "*",  
      "Action": "s3:PutObject",  
      "Resource": "arn:aws:s3:::bucketname/*",  
      "Condition": {  
        "StringNotEquals": {  
          "s3:x-amz-server-side-encryption": "aws:kms"  
        }  
      }  
    },  
    {  
      "Effect": "Deny",  
      "Principal": "*",  
      "Action": "s3:PutObject",  
      "Resource": "arn:aws:s3:::samplebucketname/*",  
      "Condition": {  
        "StringNotLikeIfExists": {  
          "s3:x-amz-server-side-encryption-aws-kms-key-id": "arn:  
aws:kms:us-east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab"  
        }  
      }  
    }  
  ]  
}
```

# S3 Bucket Key for SSE-KMS Encryption

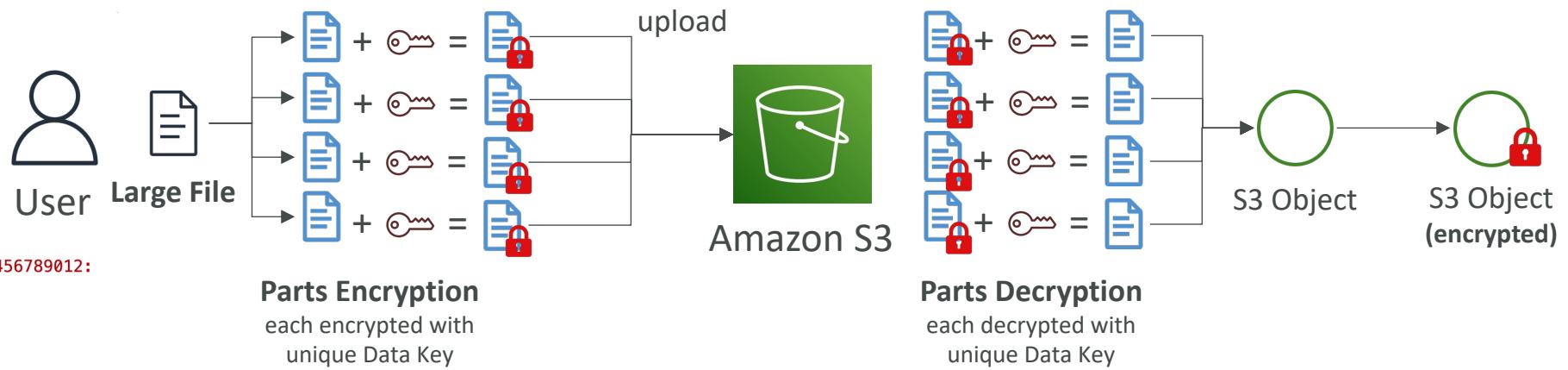
- New setting to decrease...
  - Number of API calls made to KMS from S3 by 99%
  - Costs of overall KMS encryption with Amazon S3 by 99%
- This leverages data keys
  - A “S3 bucket key” is generated
  - That key is used to encrypt KMS objects with new data keys
- You will see **less KMS CloudTrail events in CloudTrail**



# Large File Upload to S3 with KMS Key

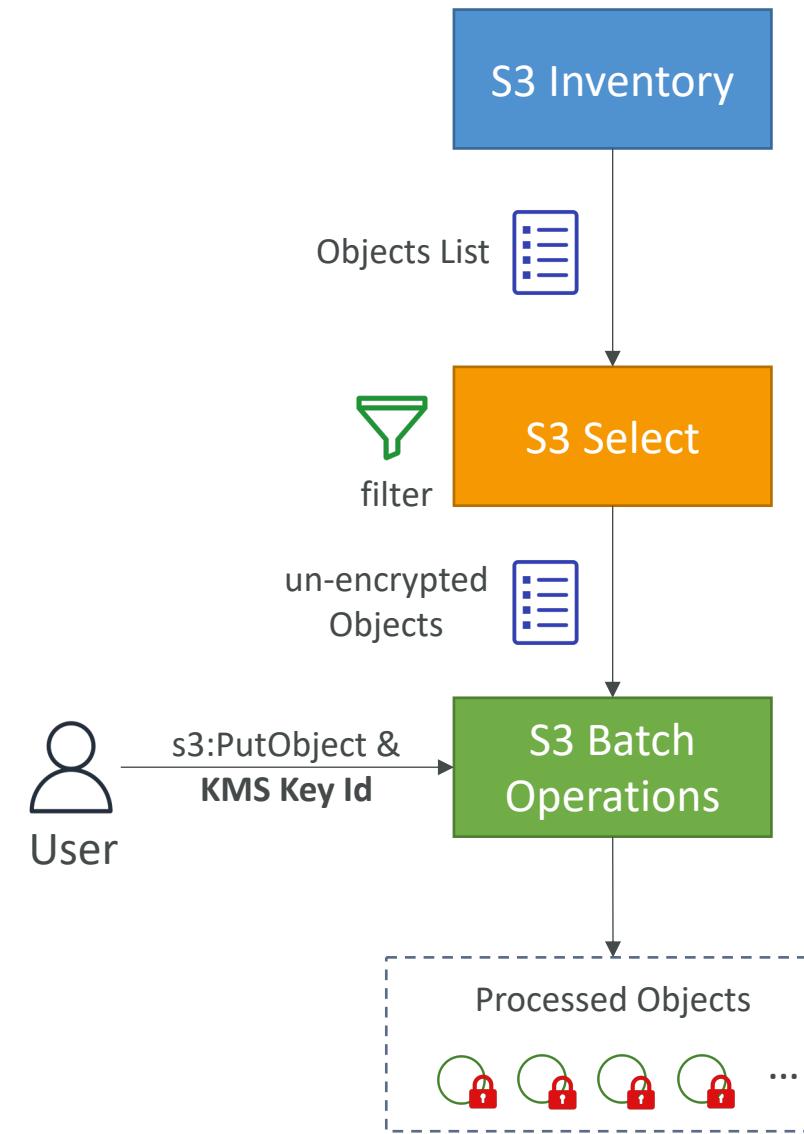
- When uploading a large file to S3 with KMS encryption, you'll have to use **S3 multi-part upload**
- You must have the following permissions to the KMS Key:
  - **kms:GenerateDataKey** – allows you to encrypt each object part with a unique Data Key
  - **kms:Decrypt** – decrypt object parts before they can be assembled, then re-encrypt them with the KMS Key

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {"Effect": "Allow",  
     "Action": [  
       "kms:Decrypt",  
       "kms:GenerateDataKey"  
     ],  
     "Resource": [  
       "arn:aws:kms:example-region-1:123456789012:  
key/111aa2bb-333c-4d44-5555-a111bb2c33dd"  
     ]  
  }]
```



# S3 Batch – Object Encryption

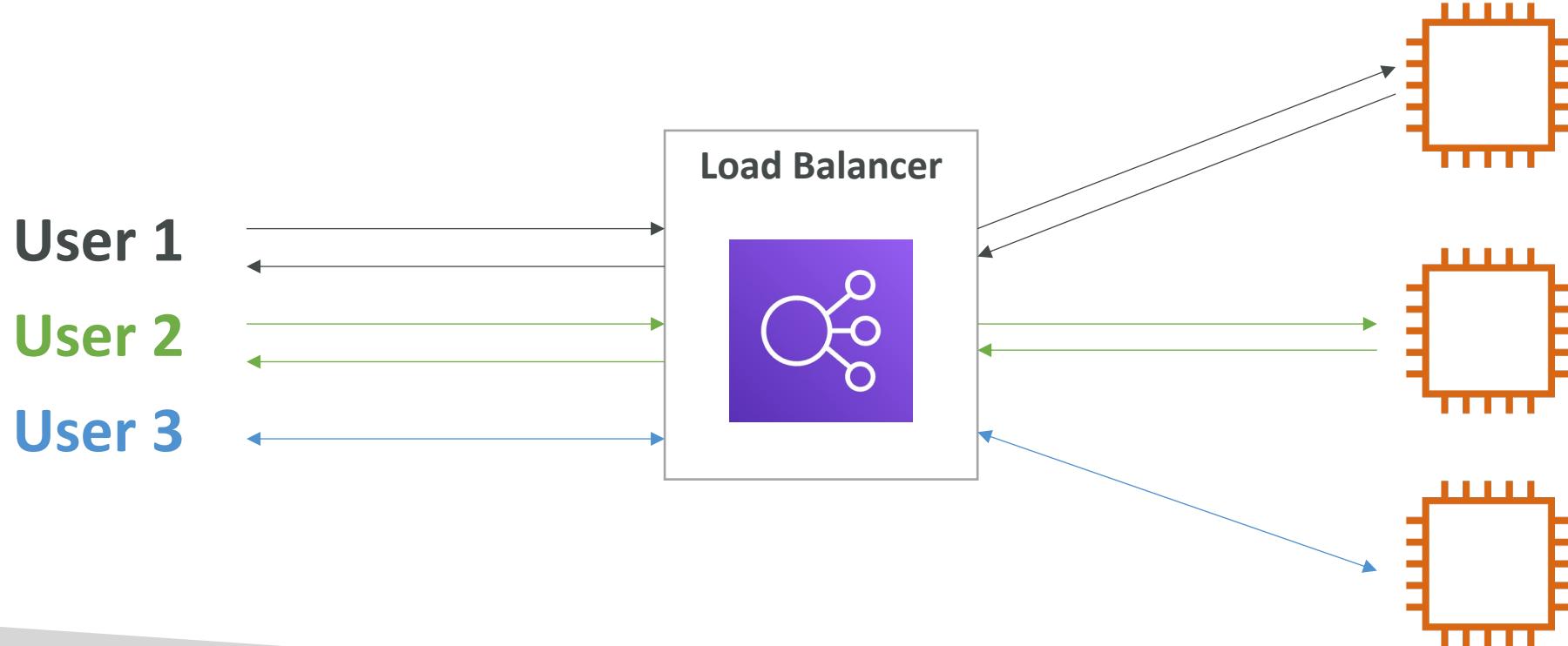
- S3 Batch – perform bulk operations on existing S3 objects with a single request (e.g., encrypt un-encrypted objects)
- S3 Inventory – to get the list of all objects and its associated metadata (select Encryption Status from optional fields)
- S3 Select or Athena – to filter and list only un-encrypted objects
- Note: S3 Batch Operations job must have access to the S3 bucket and the KMS Key



# What is load balancing?



- Load balancers are servers that forward internet traffic to multiple servers (EC2 Instances) downstream.



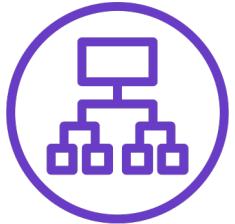
# Why use a load balancer?

- Spread load across multiple downstream instances
- Expose a single point of access (DNS) to your application
- Seamlessly handle failures of downstream instances
- Do regular health checks to your instances
- Provide SSL termination (HTTPS) for your websites
- High availability across zones

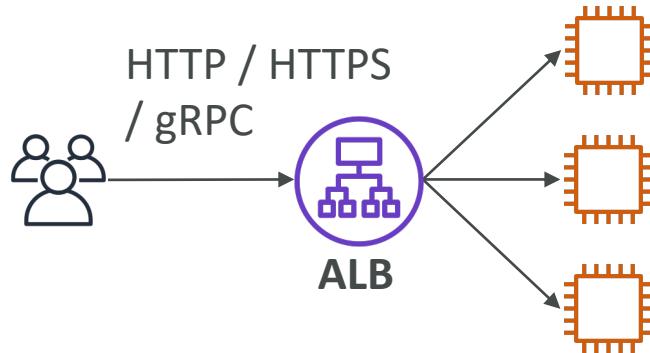
# Why use an Elastic Load Balancer?

- An ELB (Elastic Load Balancer) is a **managed load balancer**
  - AWS guarantees that it will be working
  - AWS takes care of upgrades, maintenance, high availability
  - AWS provides only a few configuration knobs
- It costs less to setup your own load balancer but it will be a lot more effort on your end (maintenance, integrations)
- 4 kinds of load balancers offered by AWS:
  - Application Load Balancer (HTTP / HTTPS only) – Layer 7
  - Network Load Balancer (ultra-high performance, allows for TCP) – Layer 4
  - Gateway Load Balancer – Layer 3
  - Classic Load Balancer (retired in 2023) – Layer 4 & 7

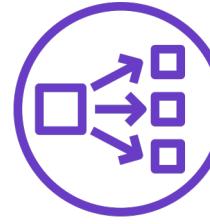
## Application Load Balancer



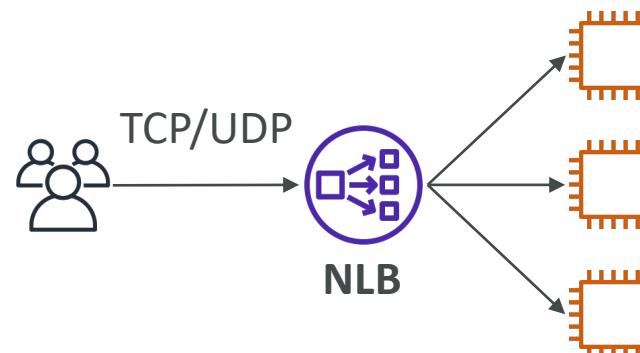
- HTTP / HTTPS / gRPC protocols (Layer 7)
- HTTP Routing features
- Static DNS (URL)



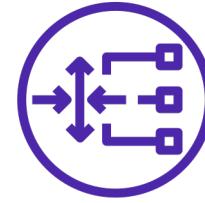
## Network Load Balancer



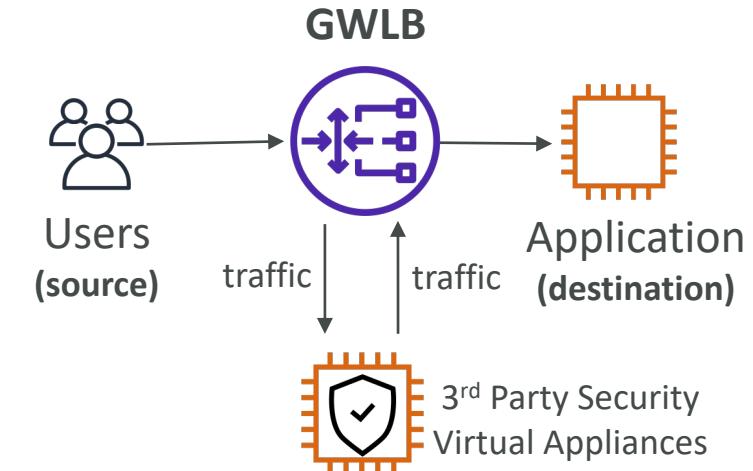
- TCP / UDP protocols (Layer 4)
- High Performance: millions of requests per seconds
- Static IP through Elastic IP

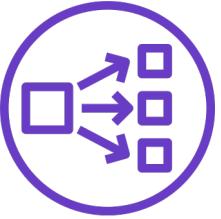


## Gateway Load Balancer



- GENEVE Protocol on IP Packets (Layer 3)
- Route Traffic to Firewalls that you manage on EC2 Instances
- Intrusion detection



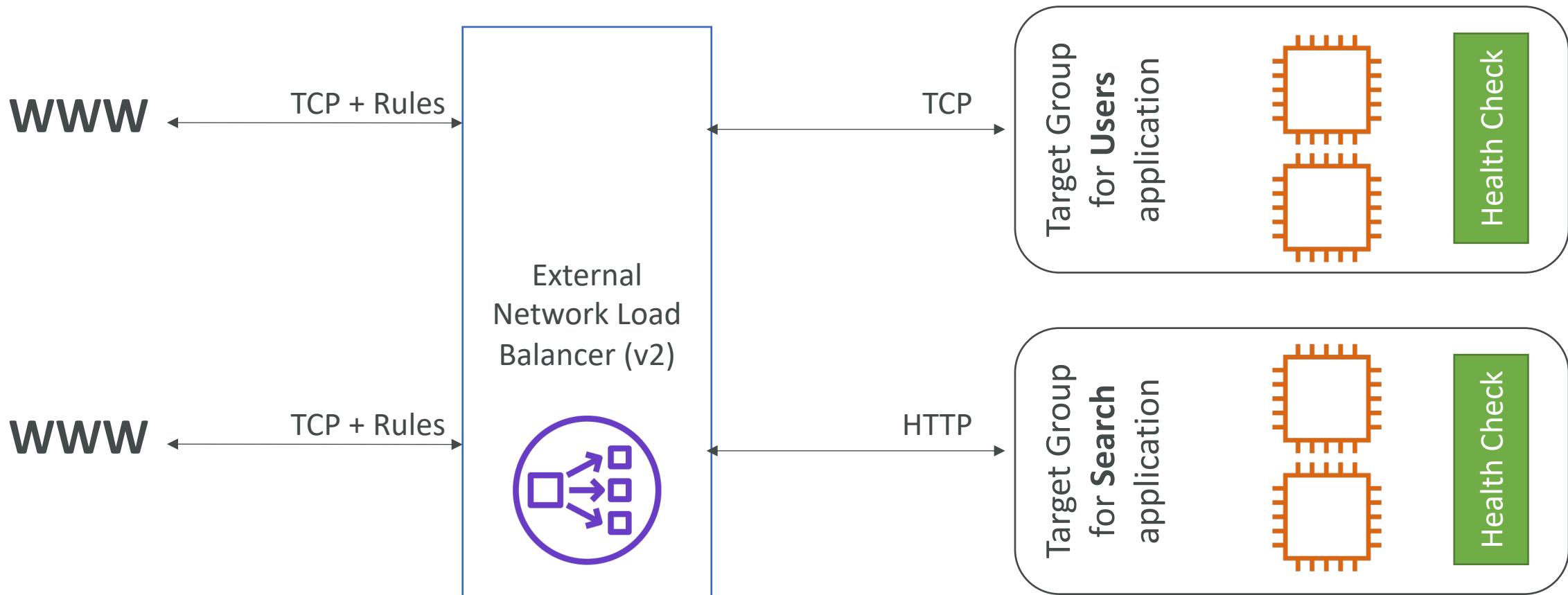


# Network Load Balancer (v2)

- Network load balancers (Layer 4) allow to:
  - Forward TCP & UDP traffic to your instances
  - Handle millions of requests per second
  - Less latency ~100 ms (vs 400 ms for ALB)
- NLB has one static IP per AZ, and supports assigning Elastic IP (helpful for whitelisting specific IP)
- NLB are used for extreme performance, TCP or UDP traffic
- Not included in the AWS free tier

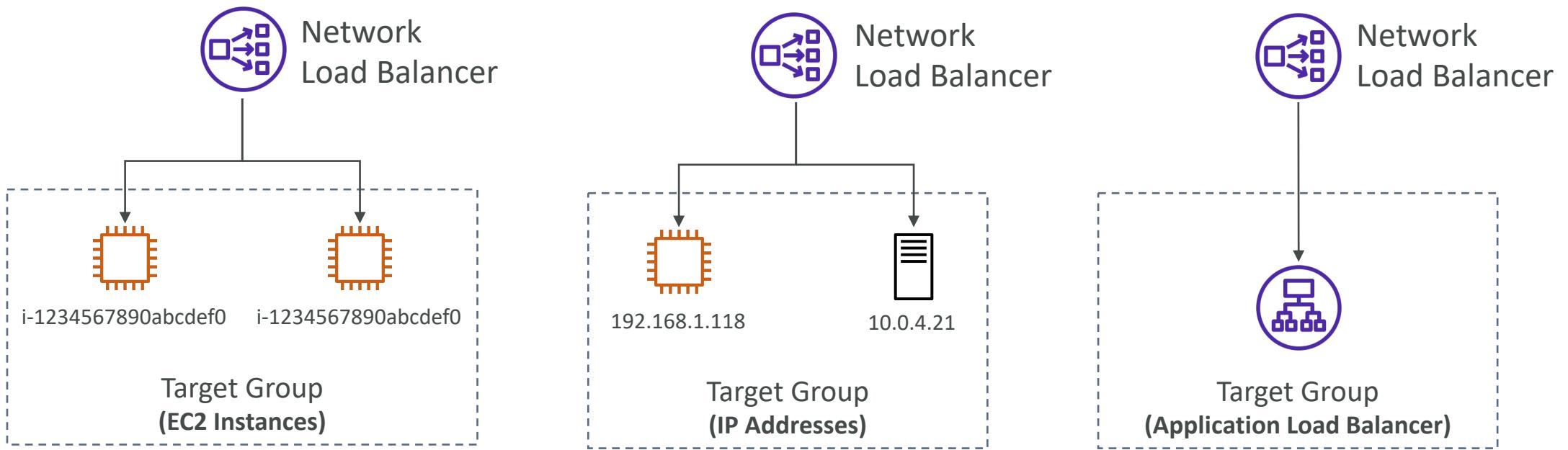
# Network Load Balancer (v2)

## TCP (Layer 4) Based Traffic



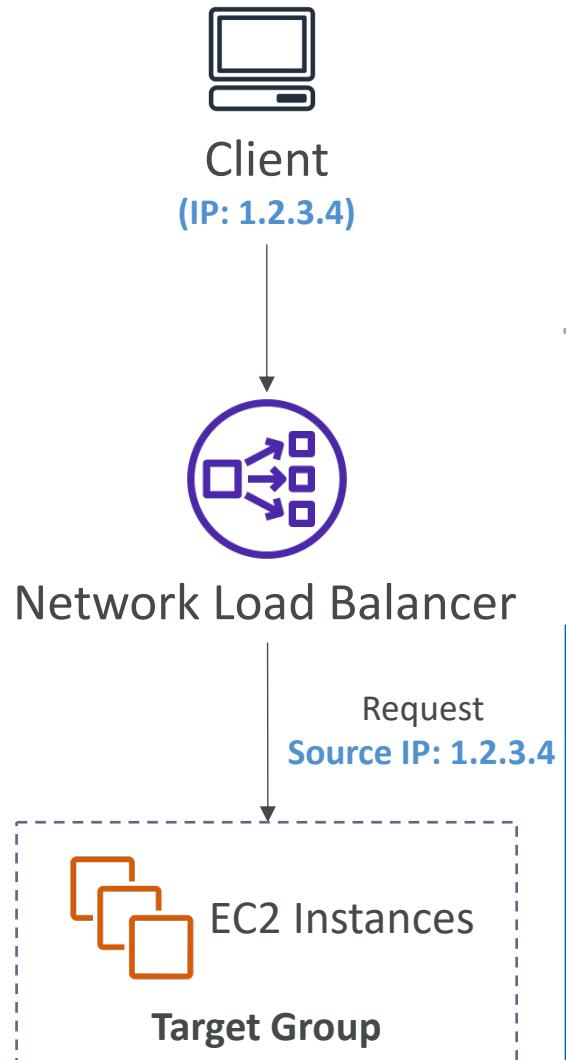
# Network Load Balancer – Target Groups

- EC2 instances
- IP Addresses – must be private IPs
- Application Load Balancer
- Health Checks support the TCP, HTTP and HTTPS Protocols



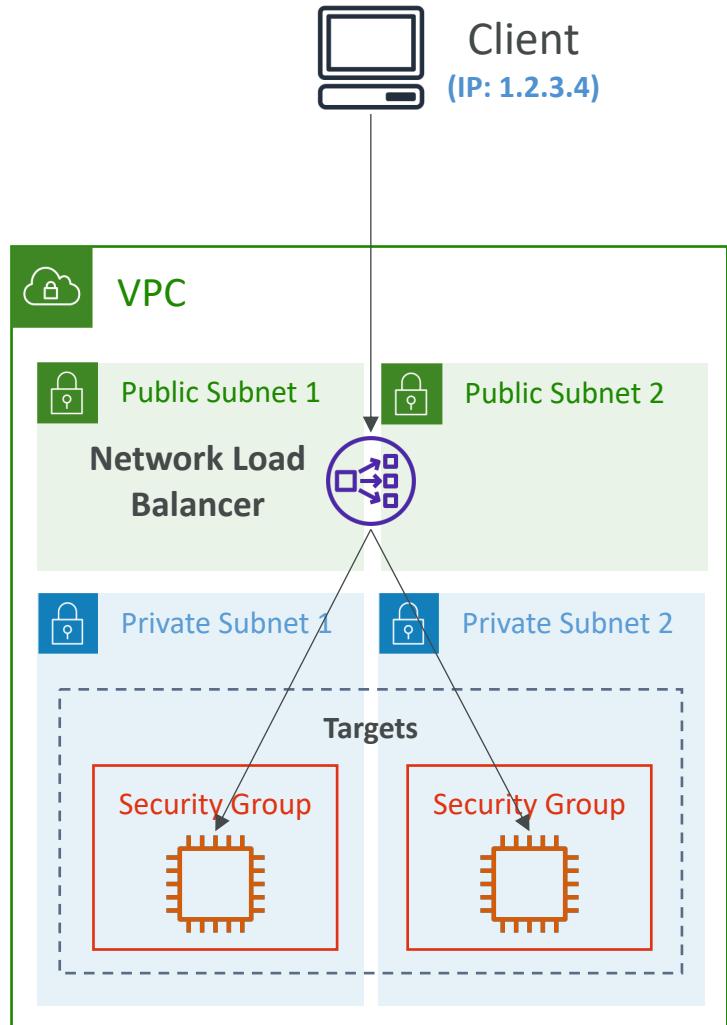
# Network Load Balancer – Client IP Preservation

- Client IP addresses are forwarded to Network Load Balancer targets
- When disabled, the private IP address of the Network Load Balancer becomes the client IP address for all incoming traffic
- Default Settings:
  - Targets by instance ID / ECS Tasks: **Enabled**
  - Targets by IP address TCP & TLS: **Disabled (default)**
  - Targets by IP address UDP & TCP\_UDP: **Enabled (default)**
- Note: Can't be disabled for both targets by instance ID and by IP address (UDP & TCP\_UDP)



# Network Load Balancer – Security Groups

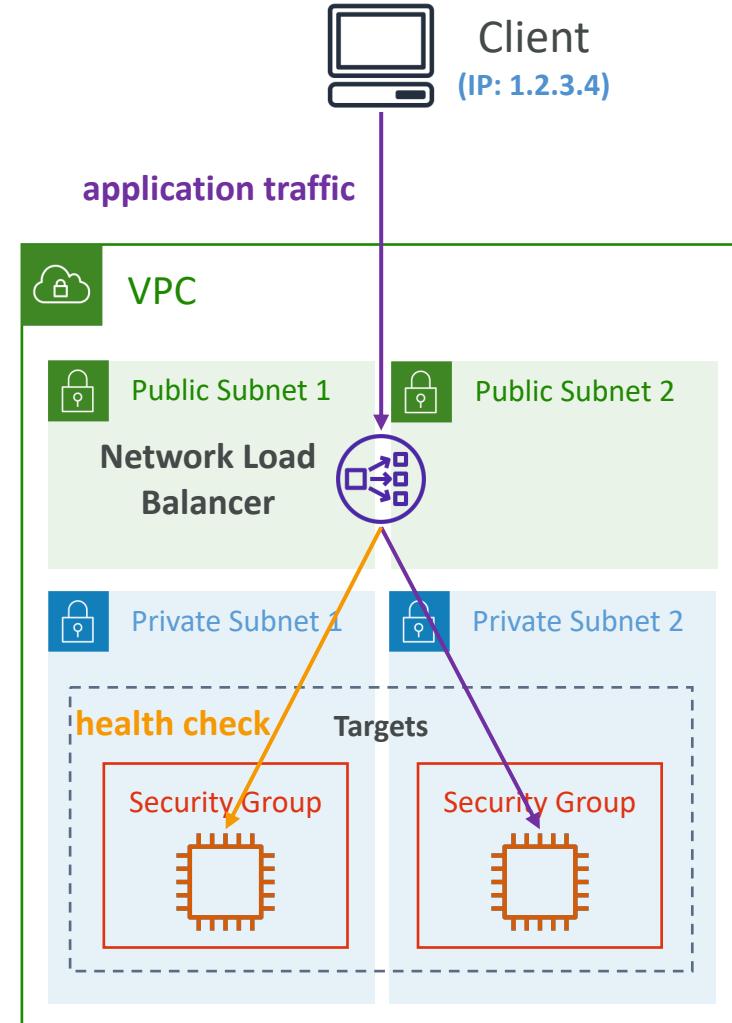
- Network Load Balancers do not have associated Security Groups
- Client's Security Groups can't be used as a source in the target's Security Groups
- Therefore, target's Security Groups must use the IP addresses of the clients to allow traffic



# NLB – Recommended Security Group Rules with Client IP Preservation Enabled

## Inbound Rules

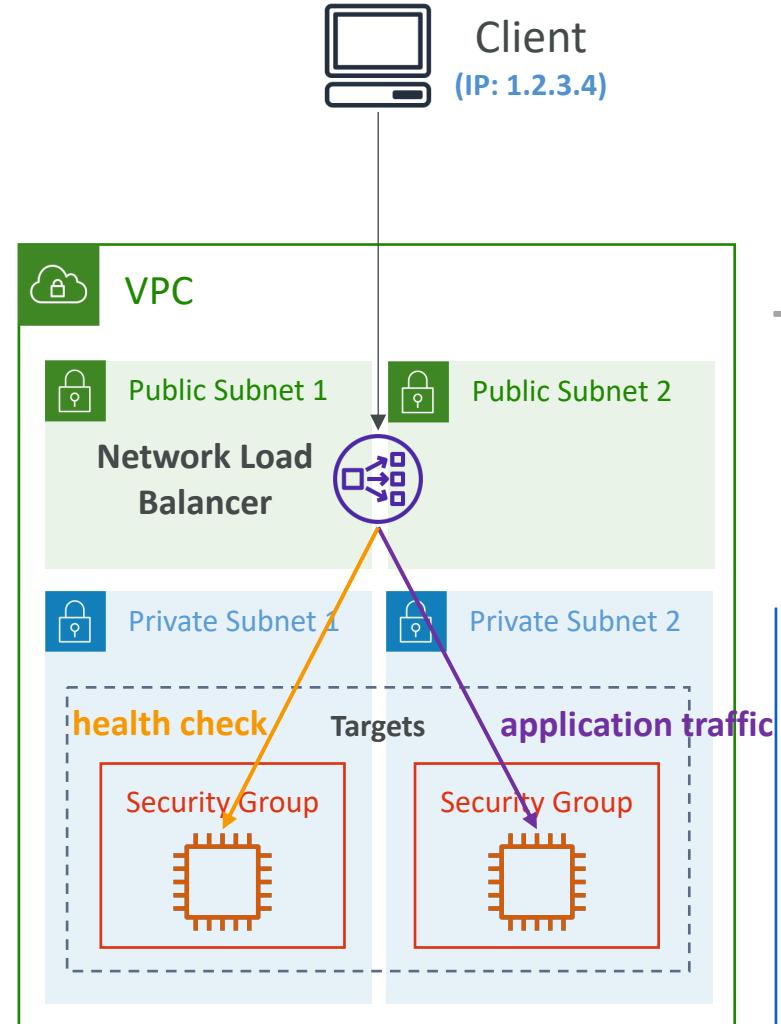
Source	Protocol	Port Range	Comment
Client IP / VPC CIDR	Listener	Listener	Allow traffic from your application, your network, Internet
VPC CIDR	Health Check	Health Check	Allow health check traffic from the Network Load Balancer



# NLB – Recommended Security Group Rules with Client IP Preservation Disabled

## Inbound Rules

Source	Protocol	Port Range	Comment
VPC CIDR	Listener	Listener	Allow traffic from Network Load Balancer VPC IP addresses
VPC CIDR	Health Check	Health Check	Allow health check traffic from the Network Load Balancer



# Network Load Balancer – Network ACL

## NLB Subnet NACLs

Source	Protocol	Port Range	Comment
Client IP / VPC CIDR	Listener	Listener	Allow traffic from Client and Internet
VPC CIDR	Health Check	1024 – 65535	Allow Health Check traffic

Inbound

Source	Protocol	Port Range	Comment
Client IP / VPC CIDR	Listener	Listener	Allow responses to Client and Internet
VPC CIDR	Health Check	Health Check	Allow Health Check traffic
VPC CIDR	Health Check	1024 – 65535	Allow Health Check traffic

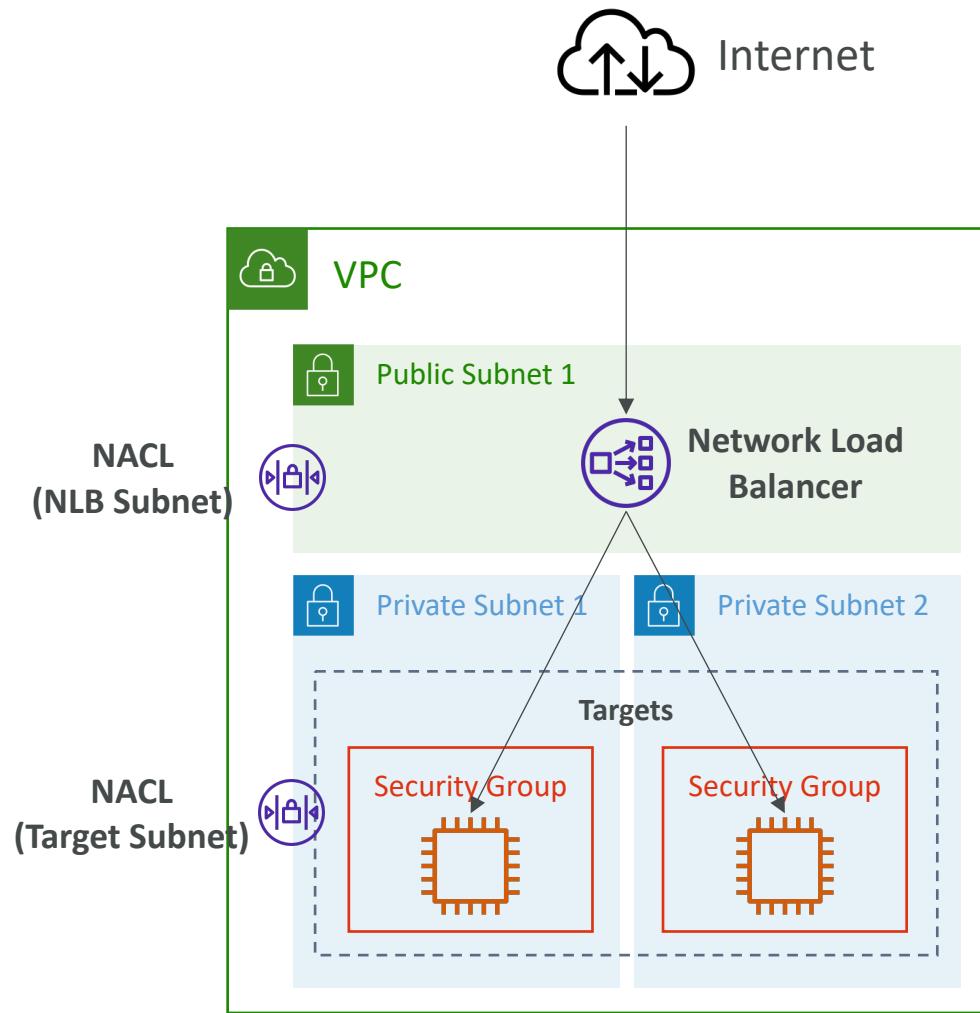
Outbound

## Targets Subnet NACLs

Source	Protocol	Port Range	Comment
Client IP / VPC CIDR	Listener	Listener	Allow traffic from Client and Internet
VPC CIDR	Health Check	Health Check	Allow Health Check traffic from NLB

Inbound

Source	Protocol	Port Range	Comment
Client IP / VPC CIDR	Listener	Listener	Allow responses to Client and Internet
VPC CIDR	Health Check	1024 – 65535	Allow Health Check traffic

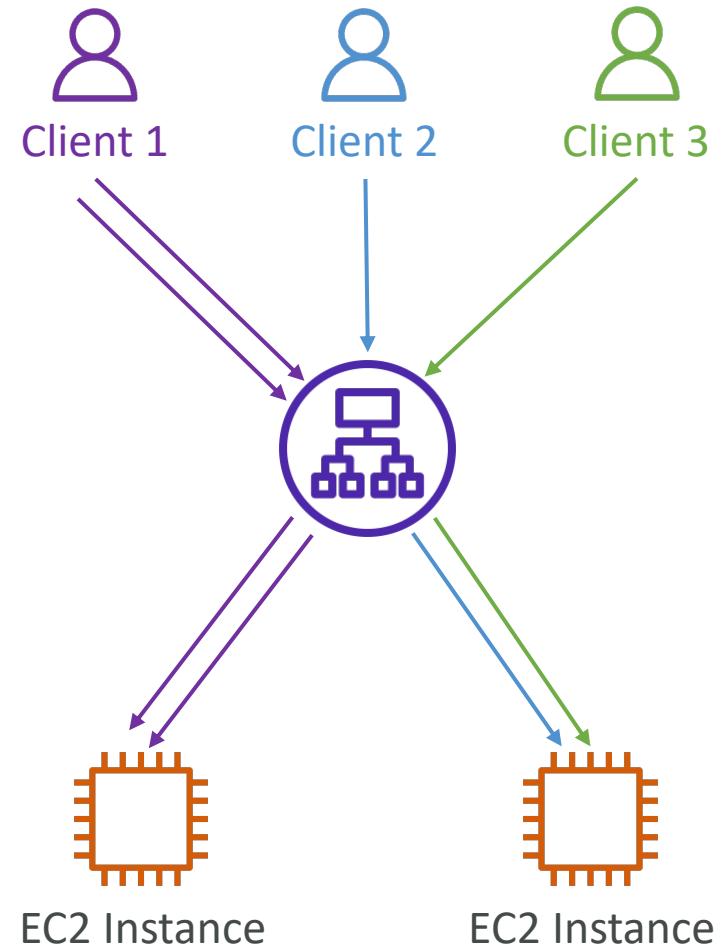


# Network Load Balancer – Troubleshooting

- Registered Target is NOT In Service
  - Health check can be failing
  - Target's Security Group does not allow traffic on health check port/protocol from the NLB
  - Target's Subnet NACL does not allow traffic from the NLB
- Requests are NOT Routed To Targets
  - Target's Security Group does not allow traffic
    - Must allow traffic from Client IP addresses (if Client IP Preservation enabled)
    - Must allow traffic from the NLB IP addresses (if Client IP Preservation disabled)
  - Target's Subnet NACL does not allow traffic (see before)
  - Targets are in an Availability Zone that is not enabled
  - EC2 instances is in a peered VPC
    - Then you must register EC2 instances by IP addresses, not by Instance ID

# Sticky Sessions (Session Affinity)

- It is possible to implement stickiness so that the same client is always redirected to the same instance behind a load balancer
- This works for Classic Load Balancers & Application Load Balancers
- The “cookie” used for stickiness has an expiration date you control
- Use case: make sure the user doesn’t lose his session data
- Enabling stickiness may bring imbalance to the load over the backend EC2 instances



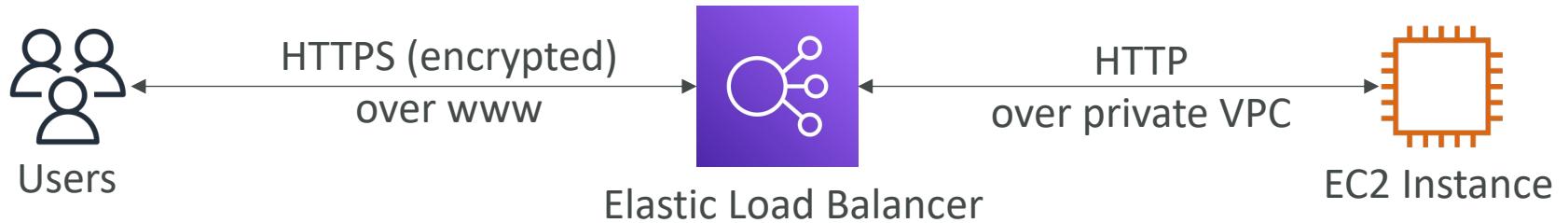
# Sticky Sessions – Cookie Names

- Application-based Cookies
  - Custom cookie
    - Generated by the target
    - Can include any custom attributes required by the application
    - Cookie name must be specified individually for each target group
    - Don't use **AWSALB**, **AWSALBAPP**, or **AWSALBTG** (reserved for use by the ELB)
  - Application cookie
    - Generated by the load balancer
    - Cookie name is **AWSALBAPP**
- Duration-based Cookies
  - Cookie generated by the load balancer
  - Cookie name is **AWSALB** for ALB, **AWSELB** for CLB

# SSL/TLS - Basics

- An SSL Certificate allows traffic between your clients and your load balancer to be encrypted in transit (in-flight encryption)
- SSL refers to Secure Socket Layer, used to encrypt connections
- TLS refers to Transport Layer Security, which is a newer version
- Nowadays, **TLS certificates are mainly used**, but people still refer as SSL
- Public SSL certificates are issued by Certificate Authorities (CA)
- Comodo, Symantec, GoDaddy, GlobalSign, DigiCert, Let's Encrypt, ...
- SSL certificates have an expiration date (you set) and must be renewed

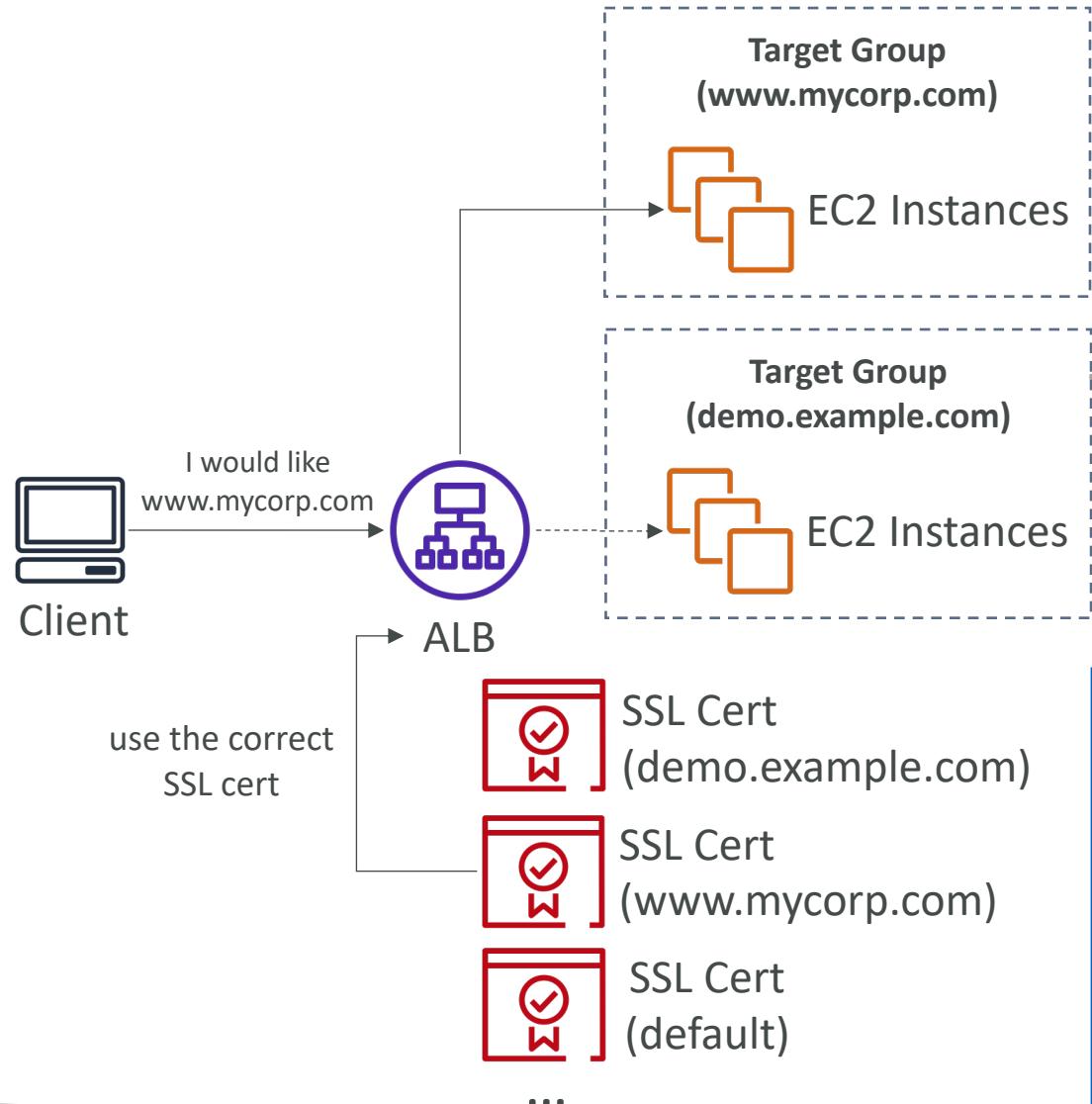
# Elastic Load Balancer – SSL Certificates



- The load balancer uses an X.509 certificate (SSL/TLS server certificate)
- You can manage certificates using ACM (AWS Certificate Manager)
- You can create/upload your own certificates alternatively
- HTTPS listener:
  - You must specify a default certificate
  - You can add an optional list of certs to support multiple domains
  - **Clients can use SNI (Server Name Indication) to specify the hostname they reach**
  - Ability to specify a Security Policy (for compliance, features, compatibility or security)

# SSL – Server Name Indication (SNI)

- SNI solves the problem of loading multiple SSL certificates onto one web server (to serve multiple websites)
- It's a “newer” protocol, and requires the client to indicate the hostname of the target hostname in the initial SSL handshake
- The server will then find the correct certificate, or return the default one
- Only works for ALB & NLB



# Elastic Load Balancers – SSL Certificates

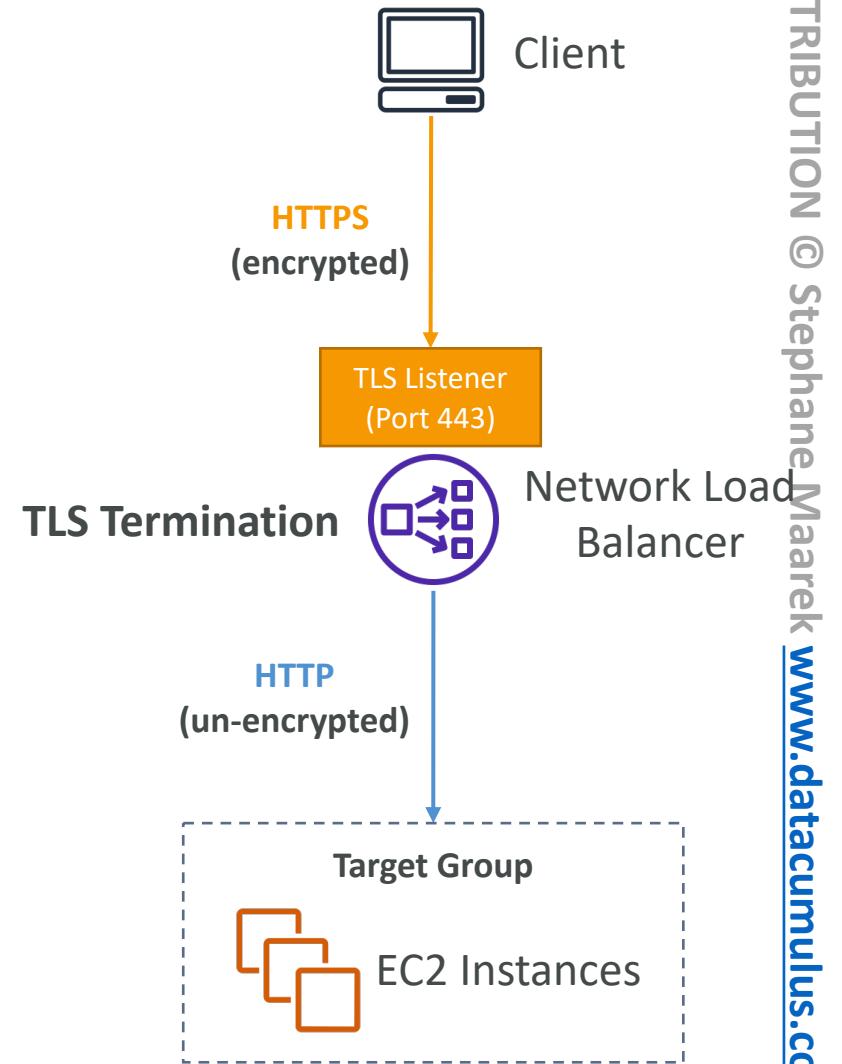
- **Classic Load Balancer**
  - Supports only one SSL certificate
  - The SSL certificate can have many Subject Alternate Name (SAN), but the SSL certificate must be changed anytime a SAN is added / edited / removed
  - Must use multiple CLB for multiple hostnames with multiple SSL certificates
  - Better to use ALB with Server Name Indication (SNI) if possible
- **Application Load Balancer**
  - Supports multiple listeners with multiple SSL certificates
  - Uses Server Name Indication (SNI) to make it work
- **Network Load Balancer**
  - Supports multiple listeners with multiple SSL certificates
  - Uses Server Name Indication (SNI) to make it work

# HTTPS/SSL Listener – Security Policy

- A combination of SSL protocols, SSL ciphers, and Server Order Preference option supported during SSL negotiations
- Predefined Security Policies (e.g., ELBSecurityPolicy-2016-08)
- For **ALB and NLB**
  - Frontend connections, you can use a predefined Security Policy
  - Backend connections, **ELBSecurityPolicy-2016-08** Security Policy is always used
- Use **ELBSecurityPolicy-TLS** policies
  - To meet compliance and security standards that require certain TLS protocol version
  - To support older versions of SSL/TLS (legacy clients)
- Use **ELBSecurityPolicy-FS** policies, if you require **Forward Secrecy**
  - Provides additional safeguards against the eavesdropping of encrypted data
  - Using a unique random session key

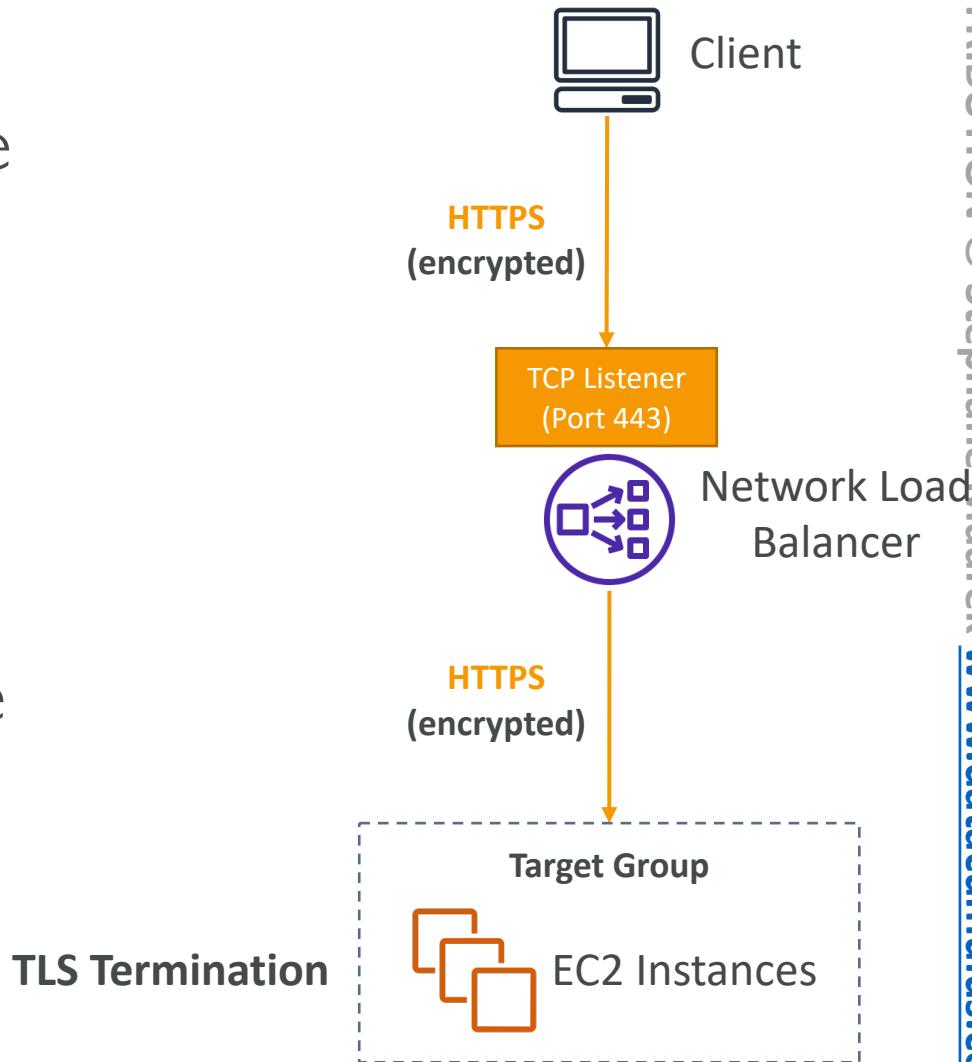
# Network Load balancer – SSL/TLS & HTTPS

- NLB uses a Server Certificate to terminate and decrypt the front-end connections before sending them to the targets
- To create a TLS listener, you must attach at least one Server Certificate from ACM on NLB
- If Client IP Preservation Enabled: both source IP address and port is presented to your backend servers, even when TLS is terminated at the NLB
- Network Load Balancers do not support TLS re-negotiation or mutual TLS authentication (mTLS)



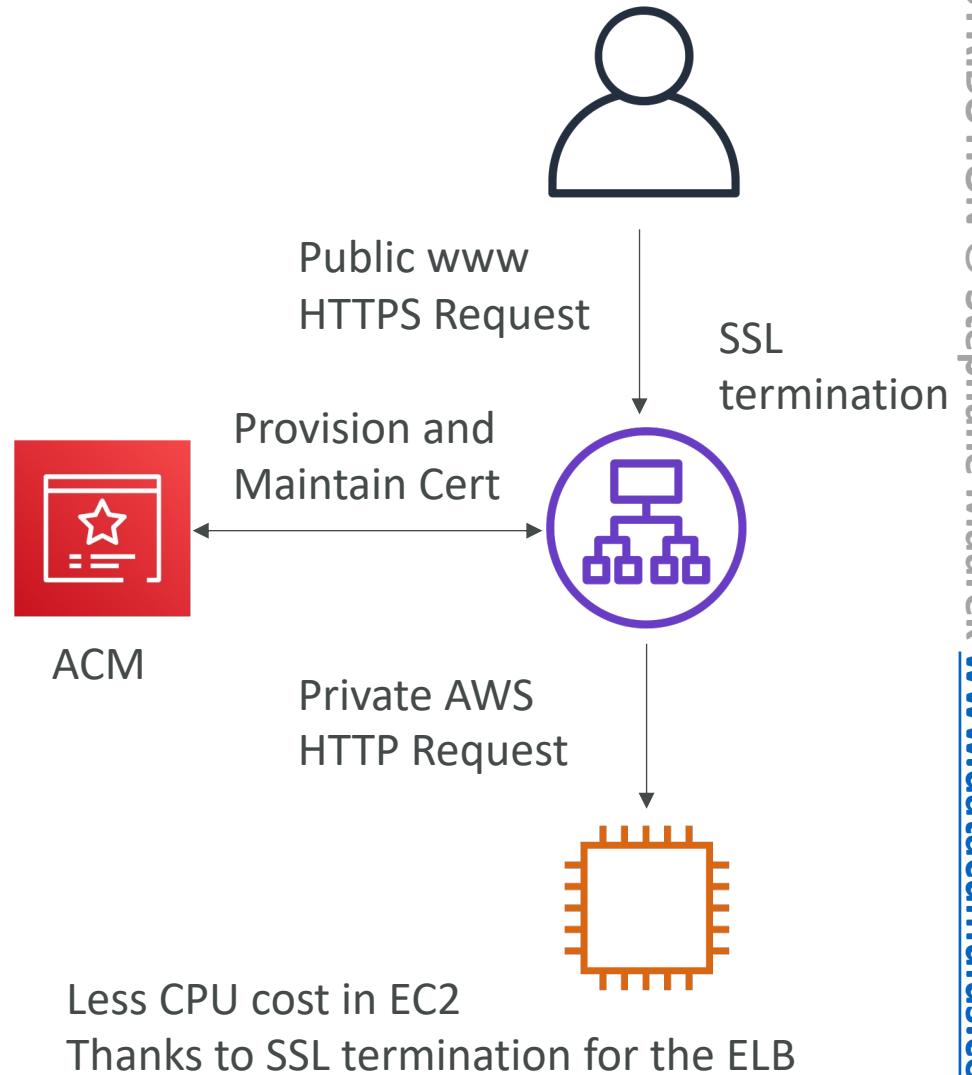
# Network Load balancer – SSL/TLS & HTTPS

- Use TCP Listener on port 443 to pass encrypted traffic to the targets without the NLB decrypting it
- Important: don't use a TLS Listener
- The targets must be able to decrypt the traffic
- TCP Listeners also support mTLS, because the NLB passes the request as is, and you can implement mTLS on the targets



# AWS Certificate Manager (ACM)

- To host public SSL certificates in AWS, you can:
  - Buy your own and upload them using the CLI
  - Have ACM provision and renew public SSL certificates for you (free of cost)
- ACM loads SSL certificates on the following integrations:
  - Load Balancers (including the ones created by EB)
  - CloudFront distributions
  - APIs on API Gateways
- SSL certificates is overall a pain to manually manage, so ACM is great to leverage in your AWS infrastructure!



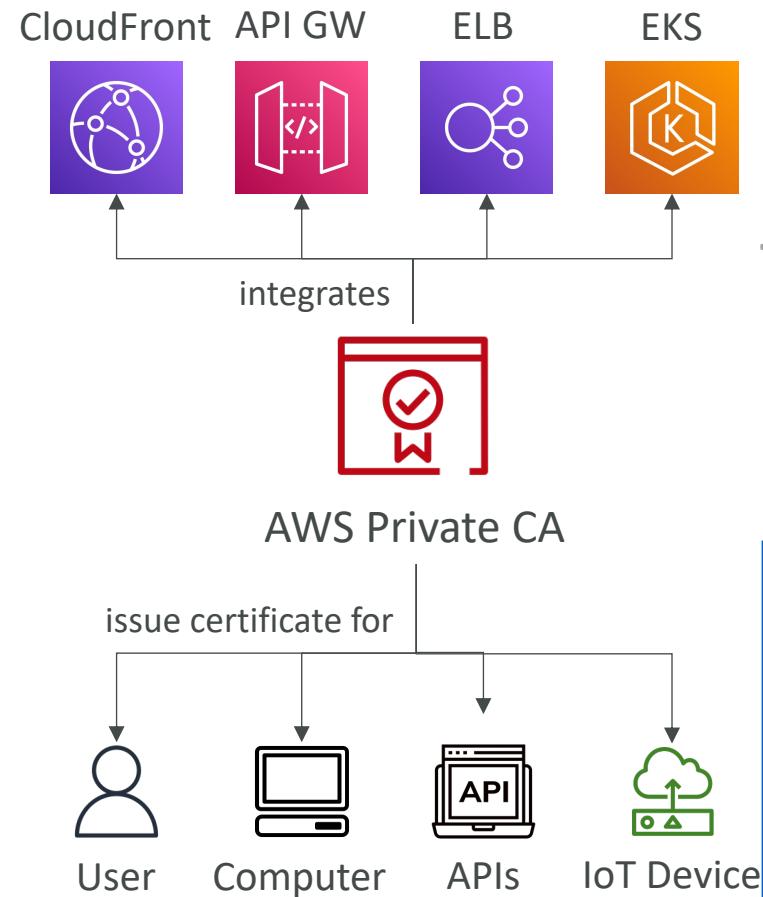
# ACM – Good to know

- Possibility of creating public certificates
  - Must verify public DNS
  - Must be issued by a trusted public certificate authority (CA)
- Possibility of creating private certificates
  - For your internal applications
  - You create your own private CA
  - Your applications must trust your private CA
- Certificate renewal:
  - Automatically done if generated provisioned by ACM
  - Any manually uploaded certificates must be renewed manually and re-uploaded
- ACM is a **regional** service
  - To use with a global application (multiple ALB for example), you need to issue an SSL certificate in each region where your application is deployed.
  - You cannot copy certs across regions

# AWS Private Certificate Authority (CA)

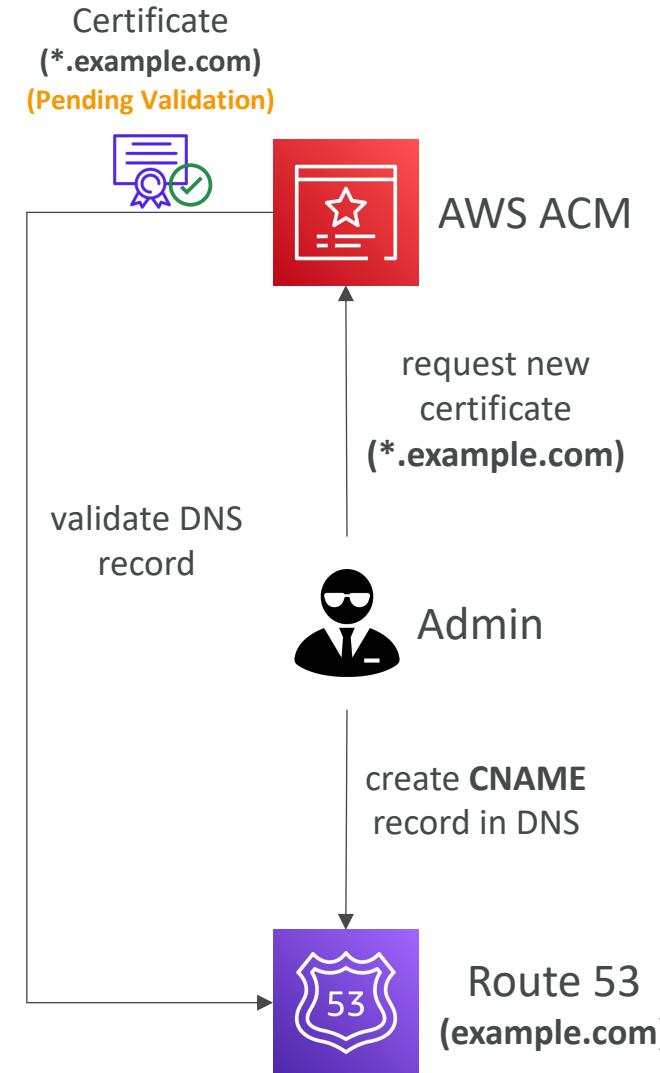


- Managed service allows you to create private Certificate Authorities (CA), including root and subordinaries CAs
- Can issue and deploy end-entity X.509 certificates
- Certificates are trusted only by your Organization (not the public Internet)
- Integrates with Amazon EKS with and any AWS service that is integrated with ACM
- Use cases:
  - Encrypted TLS communication, Cryptographically signing code
  - Authenticate users, computers, API endpoints, and IoT devices
  - Enterprise customers building a Public Key Infrastructure (PKI)



# ACM – Validation Techniques

- Before ACM issue a public certificate, you must prove that you own/control the domain
- **DNS Validation (recommended)**
  - Leverages a CNAME record created in DNS config (e.g., Route 53)
  - Preferred for automatic renewal purposes
  - Takes a few minutes to verify
- **Email Validation**
  - a Validation Email is sent to contact addresses in the WHOIS database
  - Takes a few minutes to verify
- **Note:** ACM Validation is NOT required for imported certificates or certificates signed by a Private CA



Type	Name	Value
CNAME	_a79865eb4cd1a6ab990a45779b4e0b96.example.com.	_424c7224e9b0146f9a8808af955727d0.acm-validations.aws.

# ACM – Automatic Renewal

- ACM Fails to Renew a DNS Validated Certificate
  - Most likely due to missing or inaccurate CNAME records in your DNS config.
  - You can try Email Validation (requires action by the Domain owner)
- ACM sends renewal notices 45 days before expiration
- Renewal emails sent to the Domain's WHOIS mailbox addresses
- Email contains a link that Domain owner can click for easy renewal
- ACM issues a renewed certificate with the same ARN



# ACM – Pending Validation – How to resolve?

- Resolution:

- Confirm CNAME record is added to the correct DNS config.

```
$ dig +short _a79865eb4cd1a6ab990a45779b4e0b96.example.com.
```

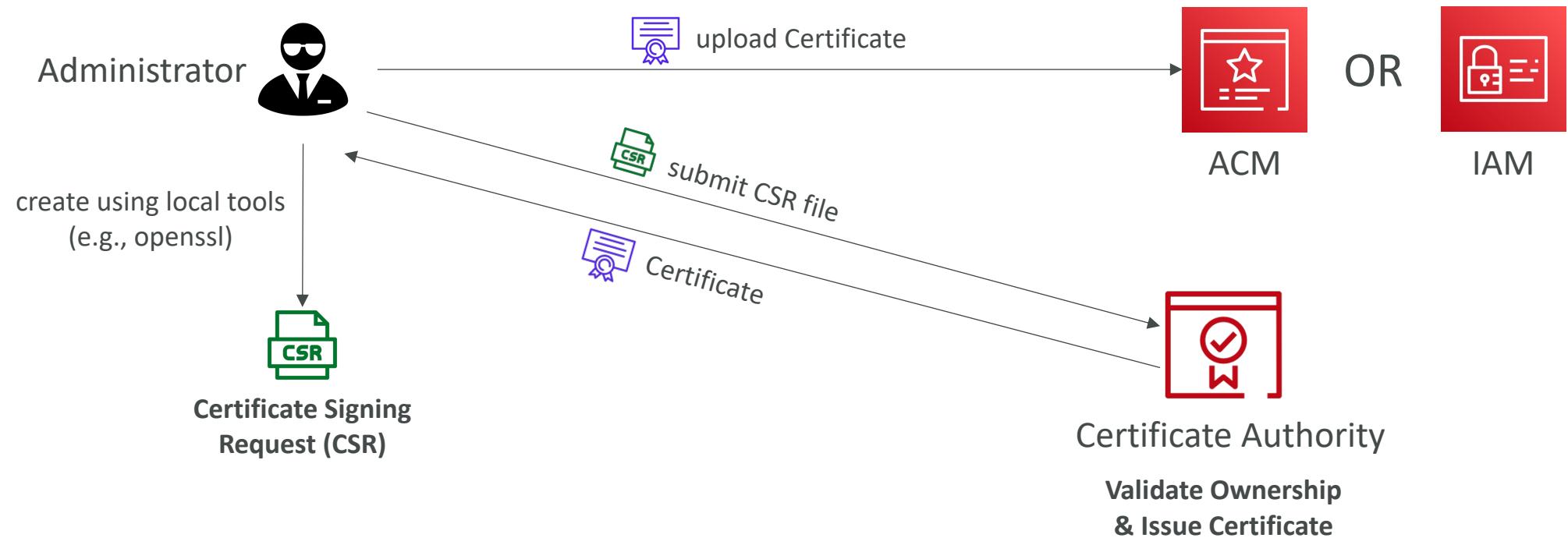
- Confirm CNAME record in your DNS config. contains no additional characters or has no missing characters
  - If your DNS Provider automatically adds the bare domain to the end of its DNS records, remove the bare domain from the DNS record name
    - \_a79865eb4cd1a6ab990a45779b4e0b96.example.com.example.com
  - If there're both CNAME and TXT records for the same domain name, then delete the TXT record

```
$ dig +short CNAME a79865eb4cd1a6ab990a45779b4e0b96.example.com.
```

```
$ dig TXT _a79865eb4cd1a6ab990a45779b4e0b96.example.com.
```

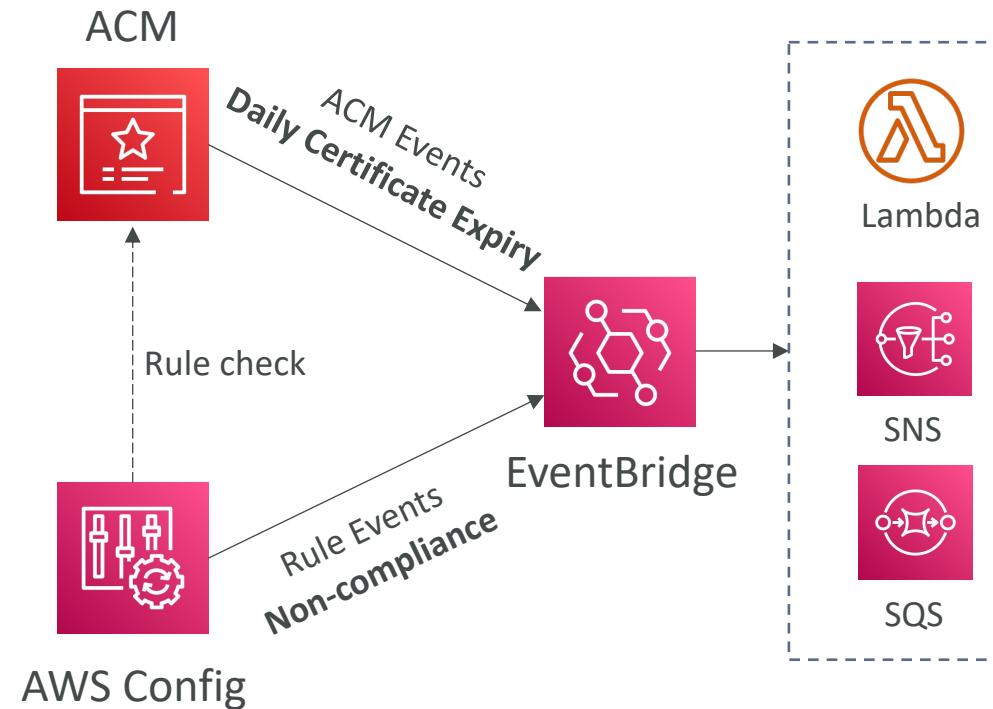
# Process to Manually Create a Certificate

- You can create a Certificate manually, then upload the Certificate to either ACM or IAM

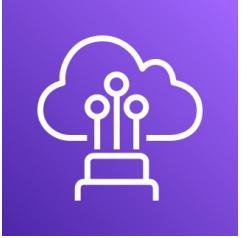


# ACM – Monitor Expired Imported Certificates

- ACM sends daily expiration events starting 45 days prior to expiration
  - The # of days can be configured
  - Events are appearing in EventBridge
- AWS Config has a managed rule named *acm-certificate-expiration-check* to check for expiring certificates (configurable number of days)



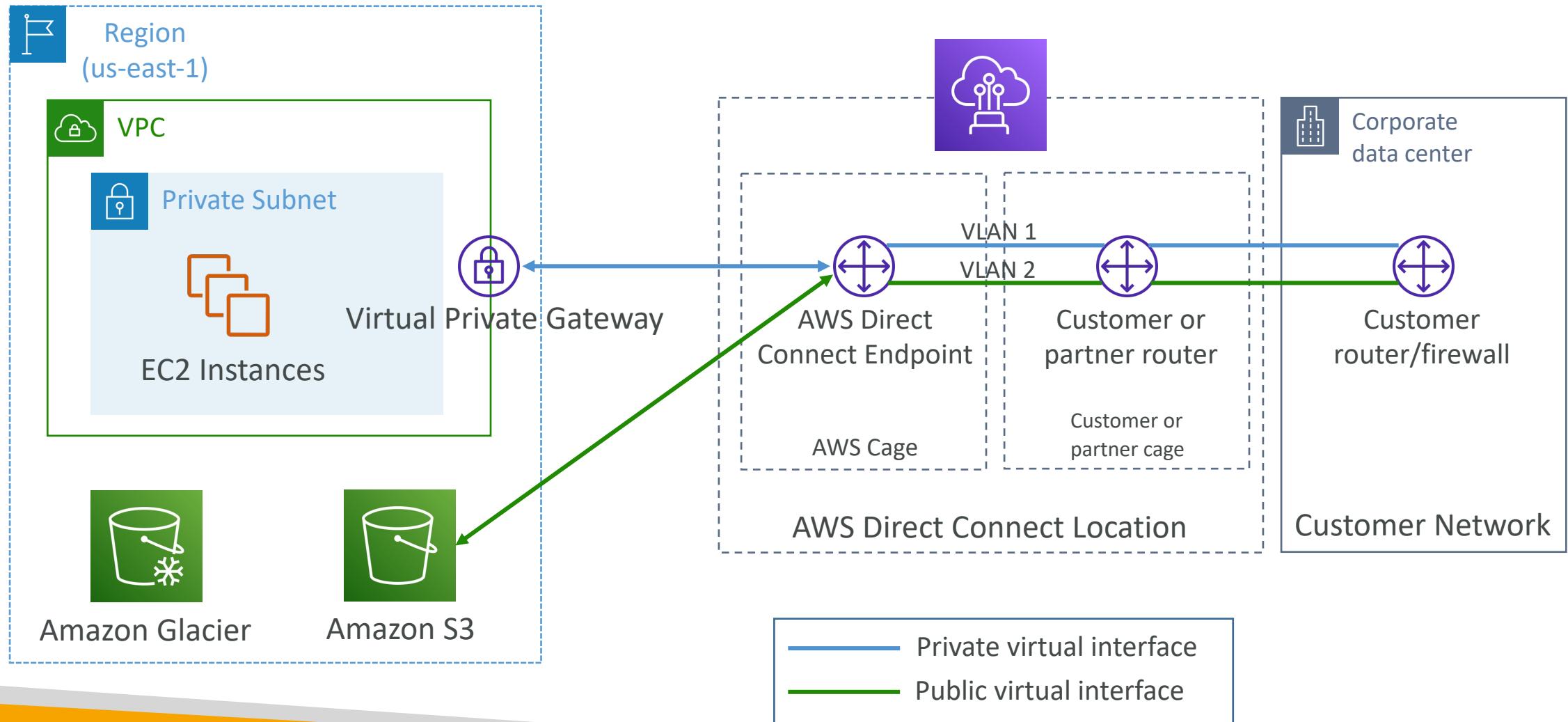
# Other Services



# Direct Connect (DX)

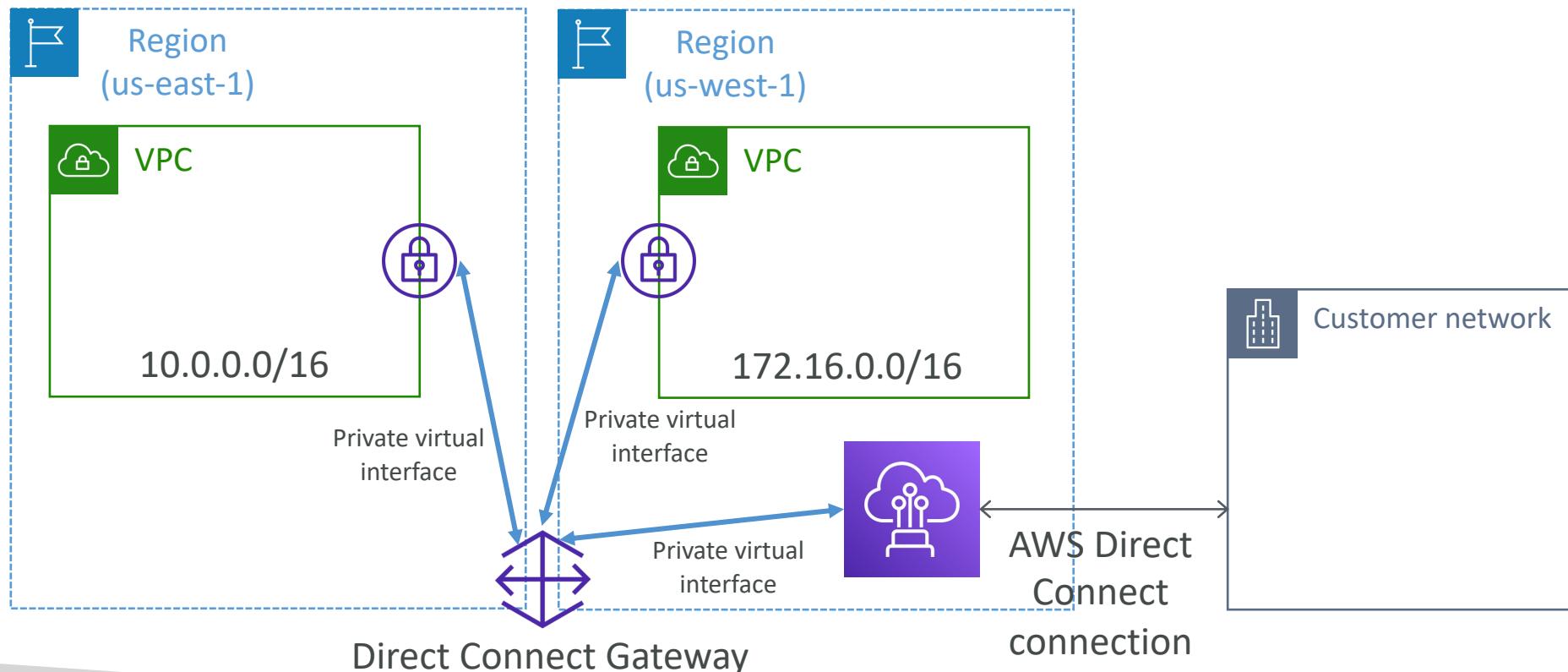
- Provides a dedicated private connection from a remote network to your VPC
- Dedicated connection must be setup between your DC and AWS Direct Connect locations
- You need to setup a Virtual Private Gateway on your VPC
- Access public resources (S3) and private (EC2) on same connection
- Use Cases:
  - Increase bandwidth throughput - working with large data sets – lower cost
  - More consistent network experience - applications using real-time data feeds
  - Hybrid Environments (on prem + cloud)
- Supports both IPv4 and IPv6

# Direct Connect Diagram



# Direct Connect Gateway

- If you want to setup a Direct Connect to one or more VPC in many different regions (same account), you must use a Direct Connect Gateway

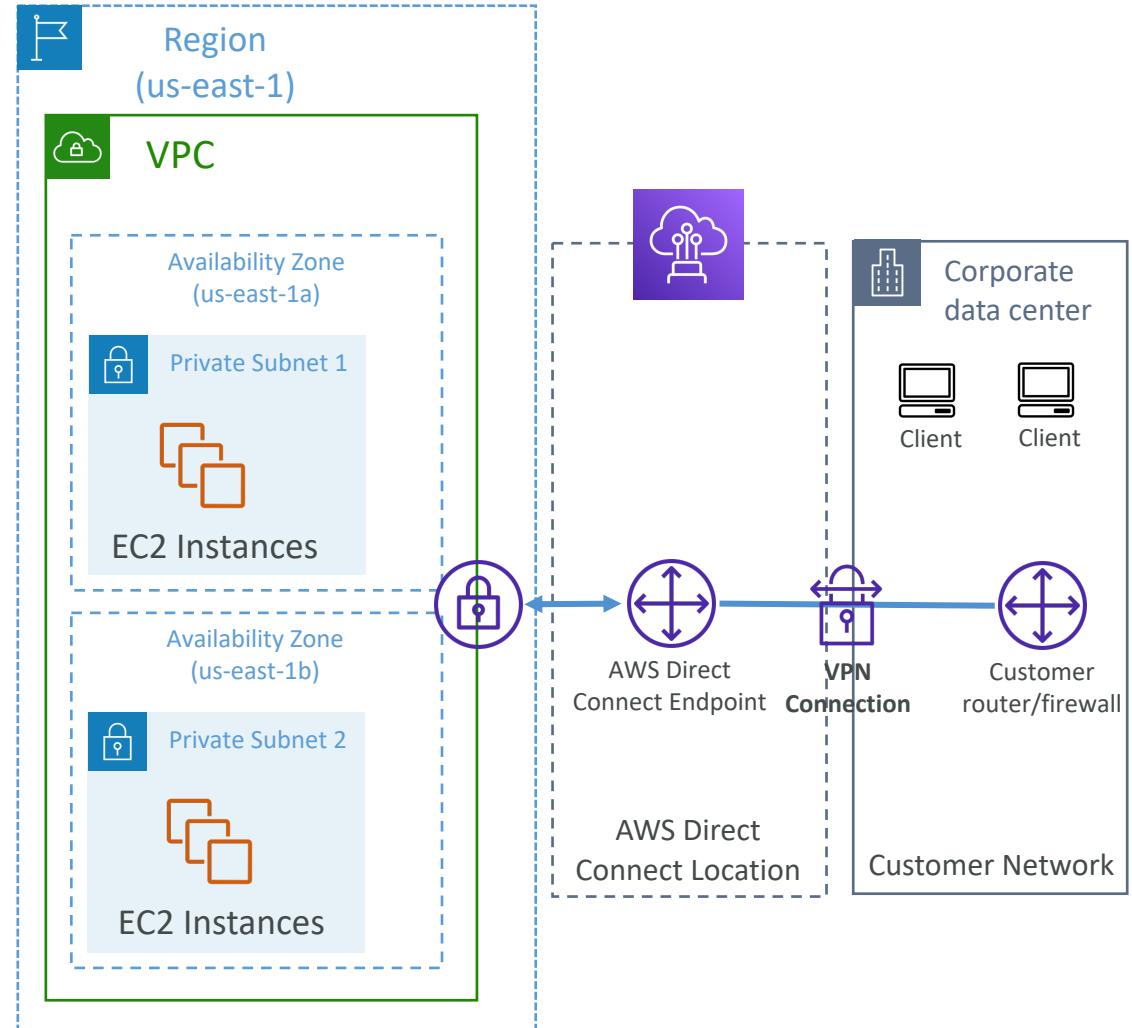


# Direct Connect – Connection Types

- **Dedicated Connections:** 1 Gbps, 10 Gbps and 100 Gbps capacity
  - Physical ethernet port dedicated to a customer
  - Request made to AWS first, then completed by AWS Direct Connect Partners
- **Hosted Connections:** 50Mbps, 500 Mbps, to 10 Gbps
  - Connection requests are made via AWS Direct Connect Partners
  - Capacity can be **added or removed on demand**
  - 1, 2, 5, 10 Gbps available at select AWS Direct Connect Partners
- Lead times are often longer than 1 month to establish a new connection

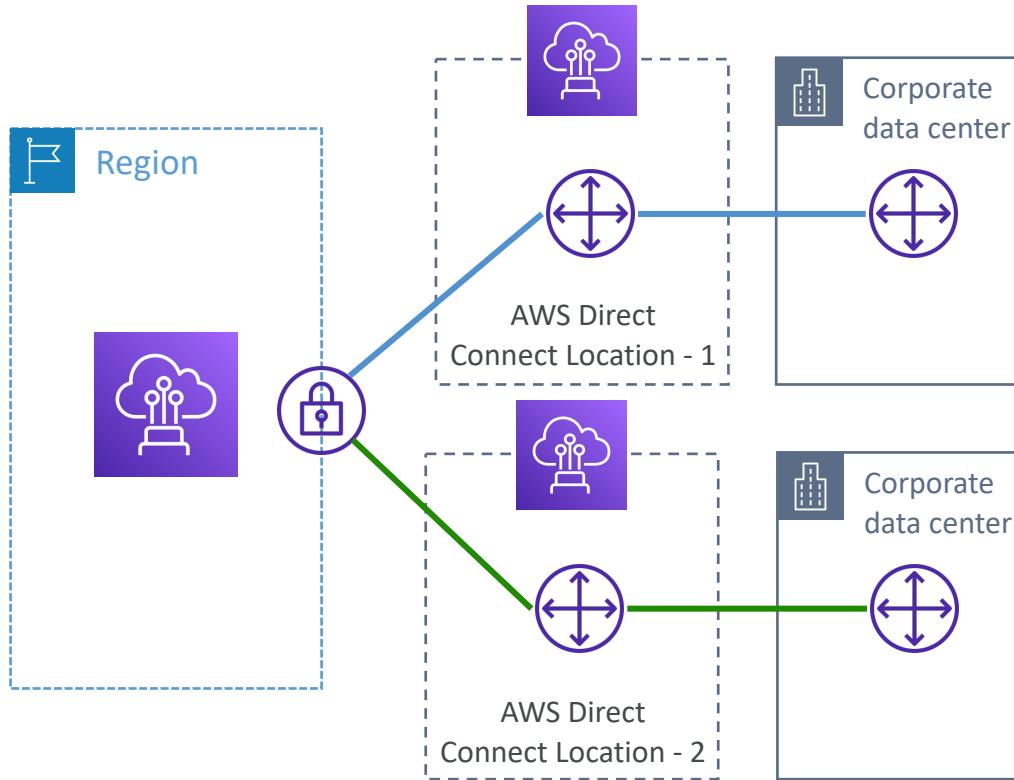
# Direct Connect – Encryption

- Data in transit is not encrypted but is private
- AWS Direct Connect + VPN provides an IPsec-encrypted private connection
- Good for an extra level of security, but slightly more complex to put in place



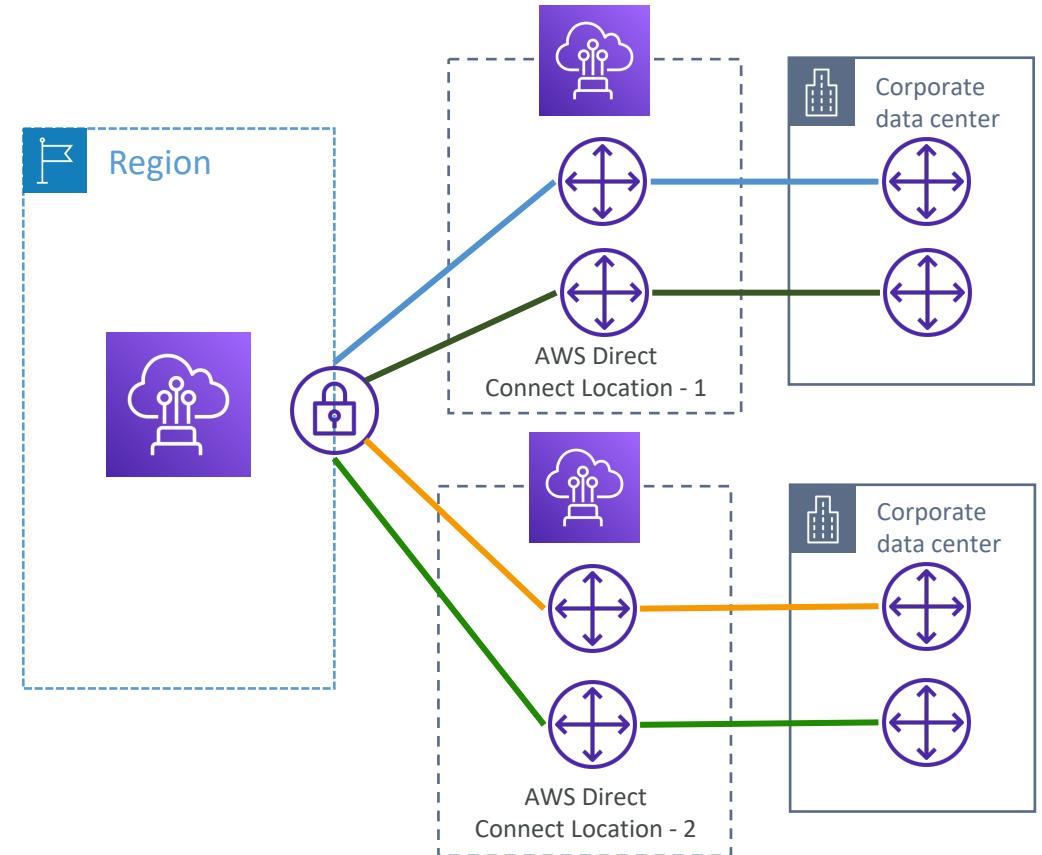
# Direct Connect - Resiliency

High Resiliency for Critical Workloads



One connection at multiple locations

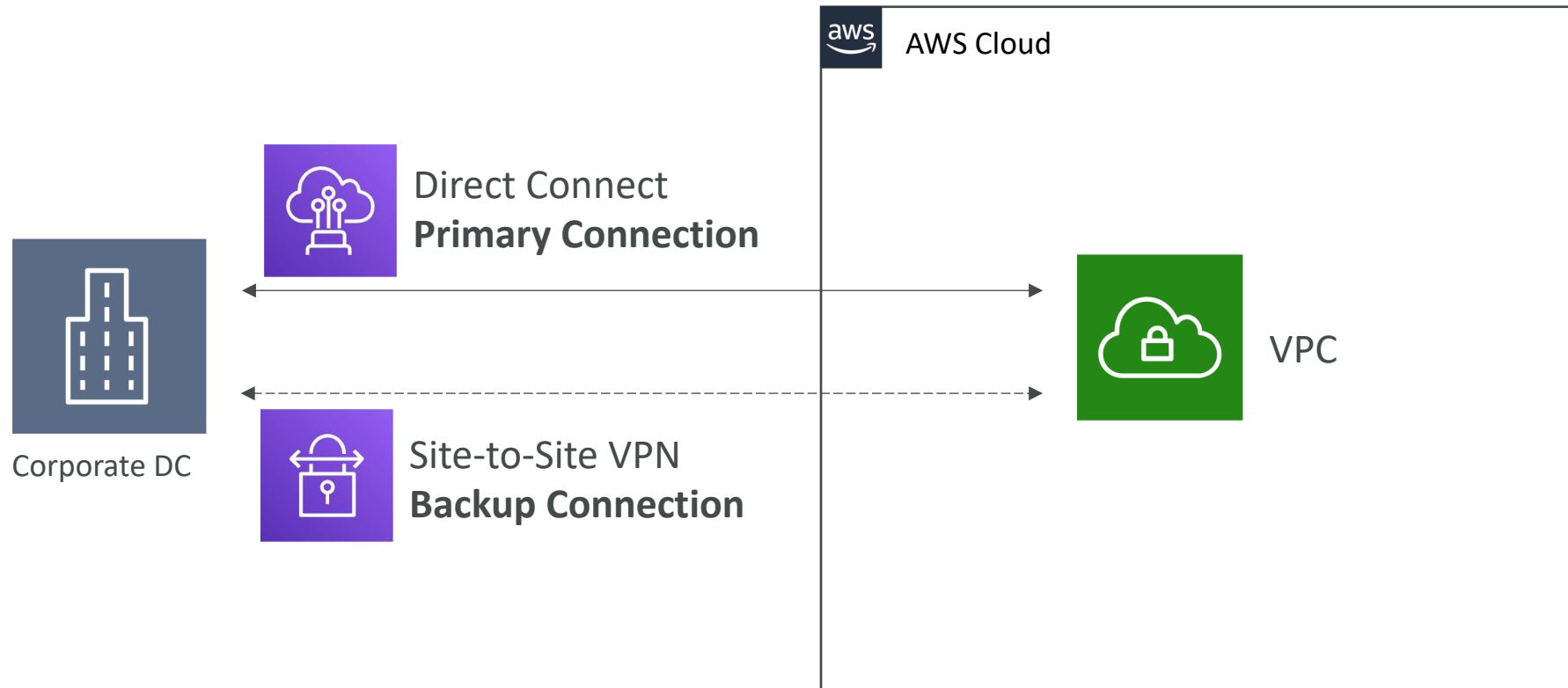
Maximum Resiliency for Critical Workloads



Maximum resilience is achieved by separate connections terminating on separate devices in more than one location.

# Site-to-Site VPN connection as a backup

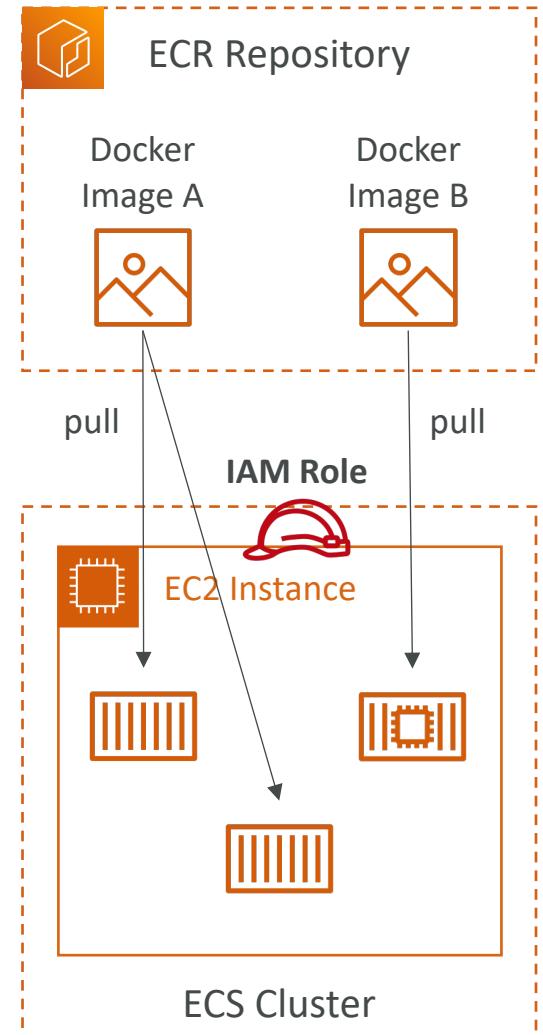
- In case Direct Connect fails, you can set up a backup Direct Connect connection (expensive), or a Site-to-Site VPN connection



# Amazon ECR

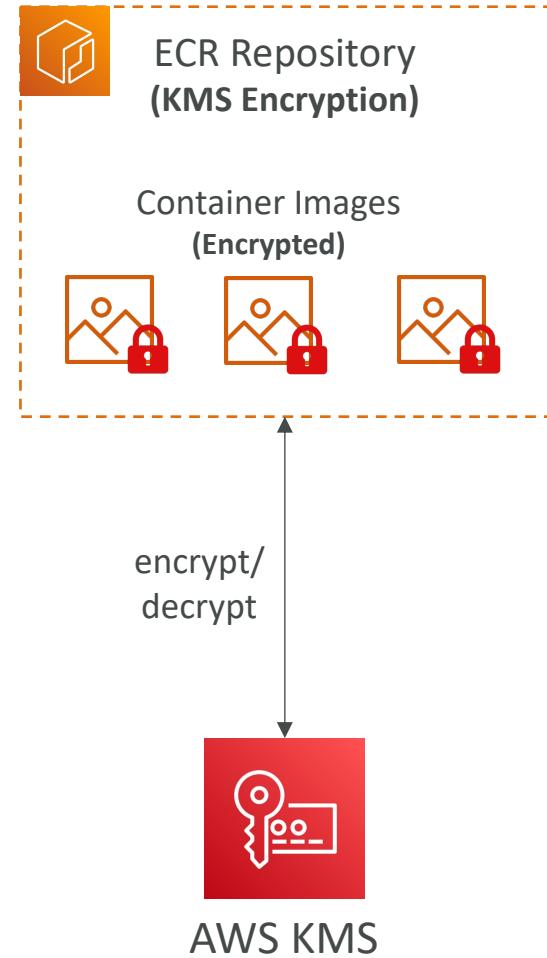


- ECR = Elastic Container Registry
- Store and manage Docker images on AWS
- Private and Public repository (Amazon ECR Public Gallery <https://gallery.ecr.aws>)
- Fully integrated with ECS, backed by Amazon S3
- Access is controlled through IAM (permission errors => policy)
- Supports image vulnerability scanning, versioning, image tags, image lifecycle, ...



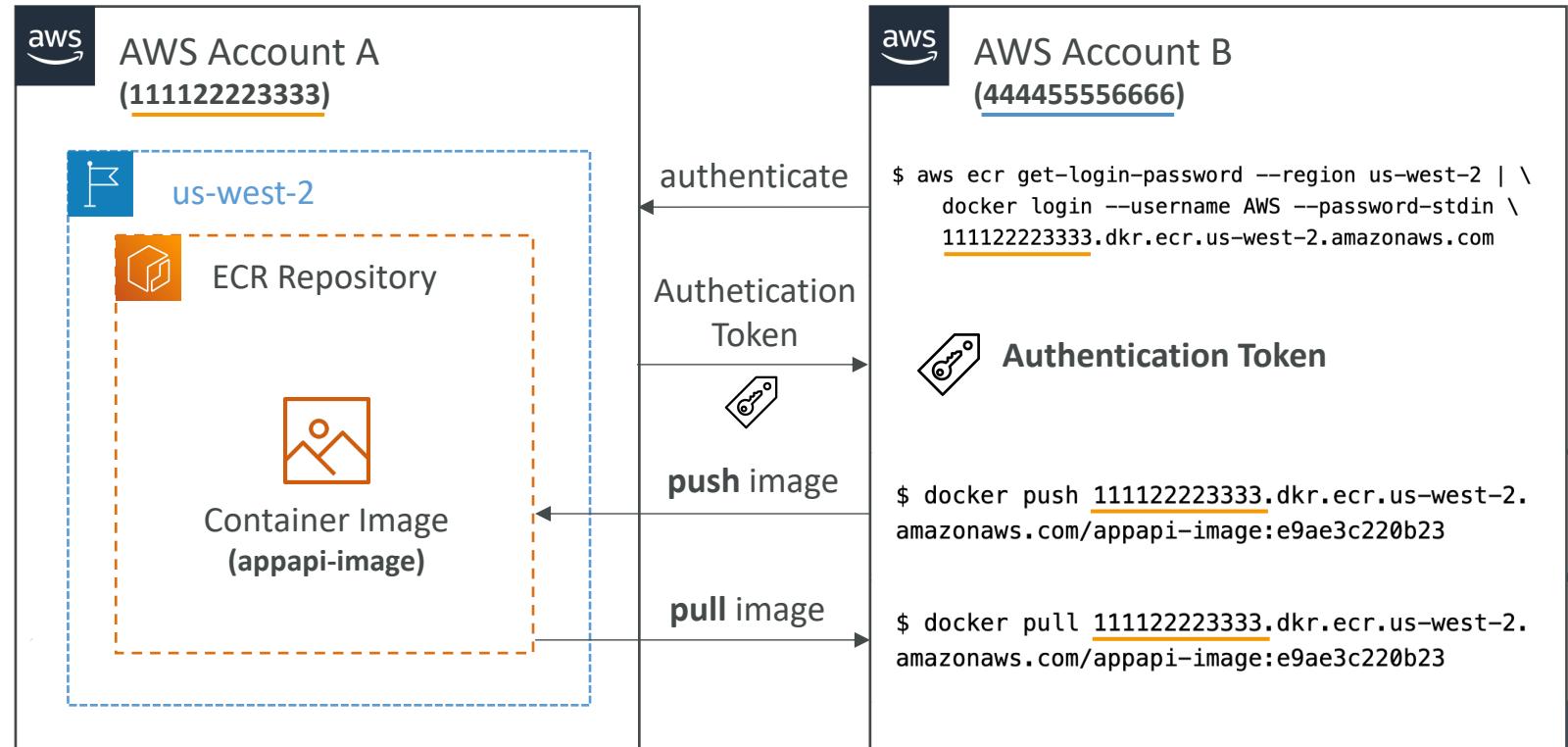
# Elastic Container Registry (ECR) with KMS

- ECR Repositories can be encrypted at rest using your KMS Keys
- Container Images are encrypted using a unique Data Encryption Key (**Envelope Encryption**)
- Can enable encryption only at Repository Creation
- ECR creates KMS Grant on your behalf to interact with KMS
- KMS Grant has **kms:DescribeKey**, **kms:Decrypt**, **kms:GenerateDataKey**, **kms:RetireGrant** (when you delete the repository)



# ECR – Cross-Account Access

```
{  
    "Version": "2008-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Principal": {  
                "AWS": "arn:aws:iam::444455556666:root"  
            },  
            "Action": [  
                "ecr:GetDownloadUrlForLayer",  
                "ecr:BatchGetImage",  
                "ecr:BatchCheckLayerAvailability",  
                "ecr:PutImage",  
                "ecr:InitiateLayerUpload",  
                "ecr:UploadLayerPart",  
                "ecr:CompleteLayerUpload"  
            ]  
        }  
    ]  
}  
  
Repository Policy
```

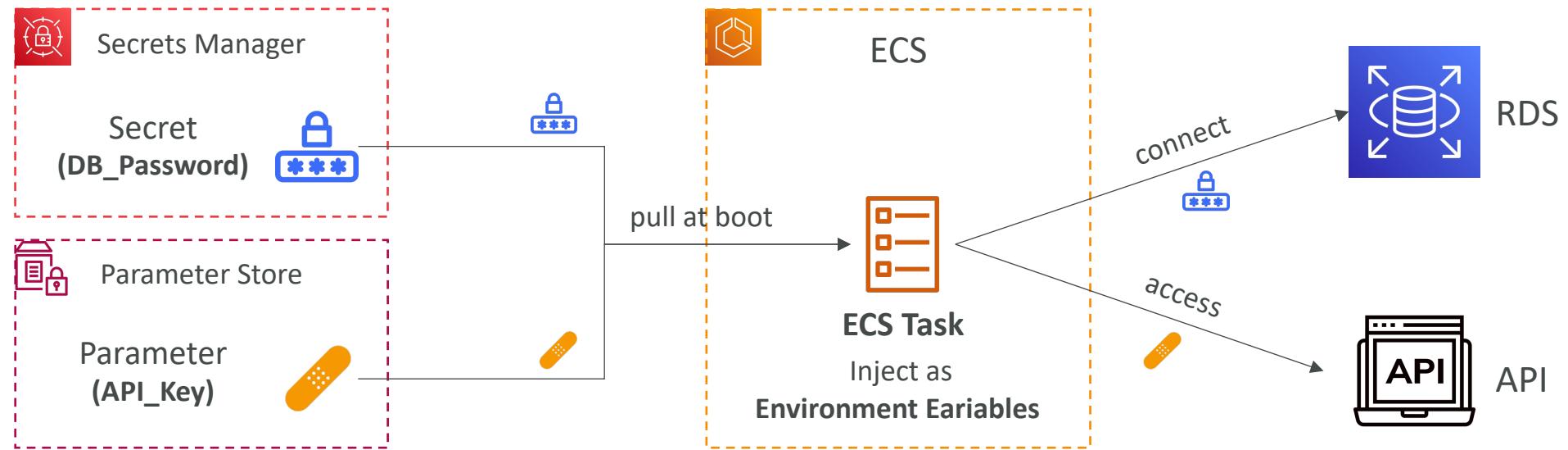


# ECR – Troubleshooting

- HTTP 403 (Forbidden) or “No Basic Auth Credentials” Error
- This error might happen from the `docker push` and `docker pull` commands even if you have successfully authenticated to Docker using `aws ecr get-login-password`
- Reasons:
  - You have authenticated to a different AWS region
  - You have authenticated to an ECR repository you don't have permissions for
  - Authenticated Token has expired (valid for 12 hours)

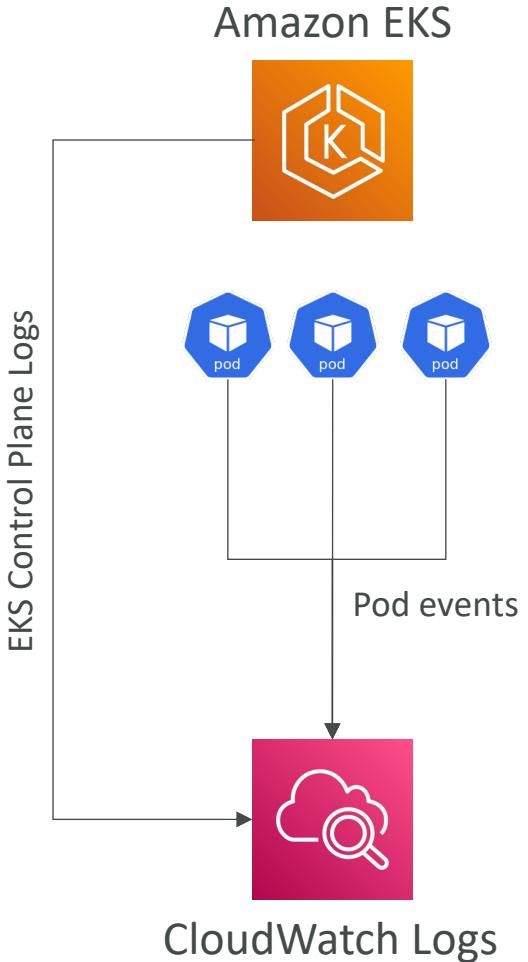
# ECS Integration with SSM Parameter Store & Secrets Manager

- ECS can inject sensitive data into your containers as environment variables
- Use SSM Parameter Store or Secrets Manager to store sensitive data
- Reference sensitive data in the container definition



# Amazon EKS – Logging

- Pod events, node events, ...
- Amazon EKS default event TTL is 60 minutes (can't be changed)
- To keep history of Kubernetes events beyond 60 minutes, send them to CloudWatch Logs
- **Amazon EKS Control Plane Logging**
  - Send EKS Control Plane logs to CloudWatch Logs
  - Audit logs, diagnostic logs, controller logs, ...
  - Ability to select the exact log types to send to CW Logs





# Lambda Execution Role (IAM Role)

- Grants the Lambda function permissions to AWS services / resources
- Sample managed policies for Lambda:
  - AWSLambdaBasicExecutionRole – Upload logs to CloudWatch.
  - AWSLambdaKinesisExecutionRole – Read from Kinesis
  - AWSLambdaDynamoDBExecutionRole – Read from DynamoDB Streams
  - AWSLambdaSQSQueueExecutionRole – Read from SQS
  - AWSLambdaVPCAccessExecutionRole – Deploy Lambda function in VPC
  - AWSXRayDaemonWriteAccess – Upload trace data to X-Ray.
- When you use an event source mapping to invoke your function, Lambda uses the execution role to read event data.
- Best practice: create one Lambda Execution Role per function

# Lambda Resource Based Policies

- Use resource-based policies to give other accounts and AWS services permission to use your Lambda resources
- Similar to S3 bucket policies for S3 bucket
- An IAM principal can access Lambda:
  - if the IAM policy attached to the principal authorizes it (e.g. user access)
  - OR if the resource-based policy authorizes (e.g. service access)
- When an AWS service like Amazon S3 calls your Lambda function, the resource-based policy gives it access.

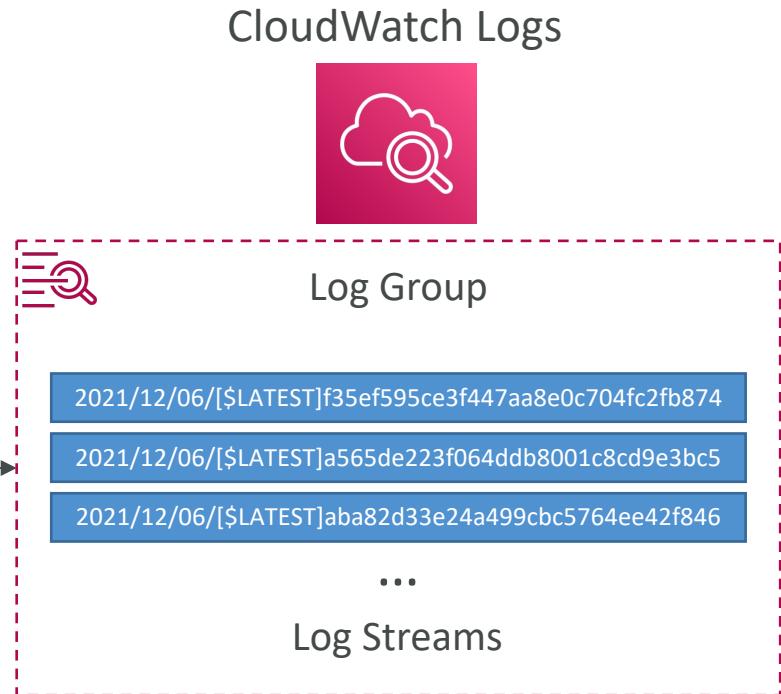
# Lambda Permissions to Write to CW Logs

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "logs:CreateLogGroup",  
                "logs:CreateLogStream",  
                "logs:PutLogEvents",  
                "logs:DescribeLogStreams"  
            ],  
            "Resource": [  
                "arn:aws:logs:*:*:  
            ]  
        }  
    ]  
}
```



Lambda Function

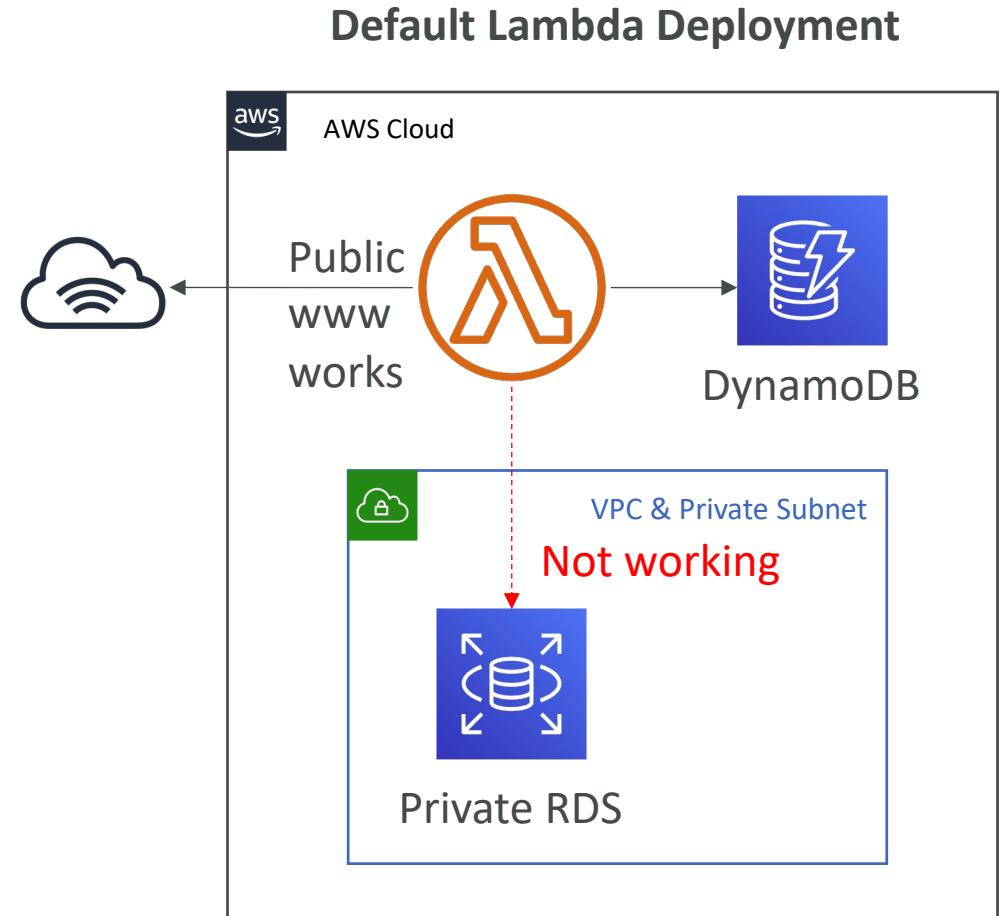
logs



Lambda Execution Role

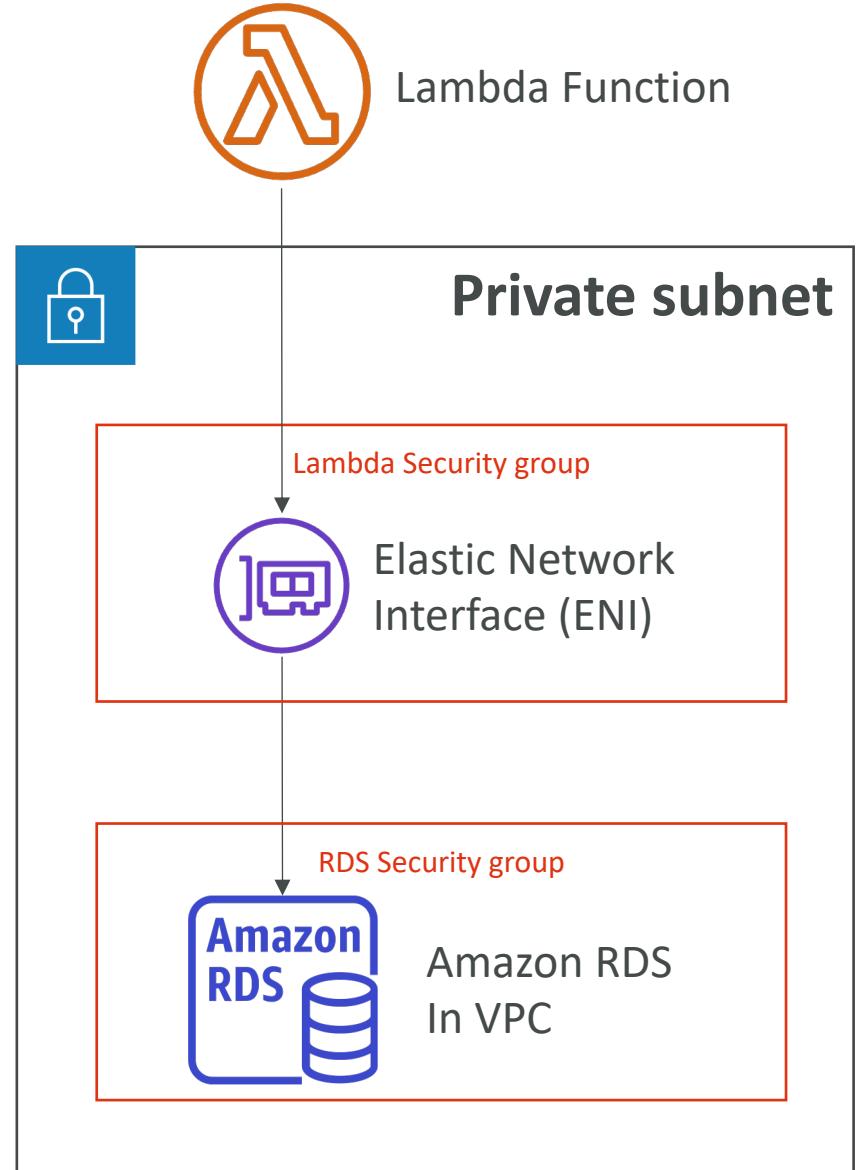
# Lambda by default

- By default, your Lambda function is launched outside your own VPC (in an AWS-owned VPC)
- Therefore it cannot access resources in your VPC (RDS, ElastiCache, internal ELB...)



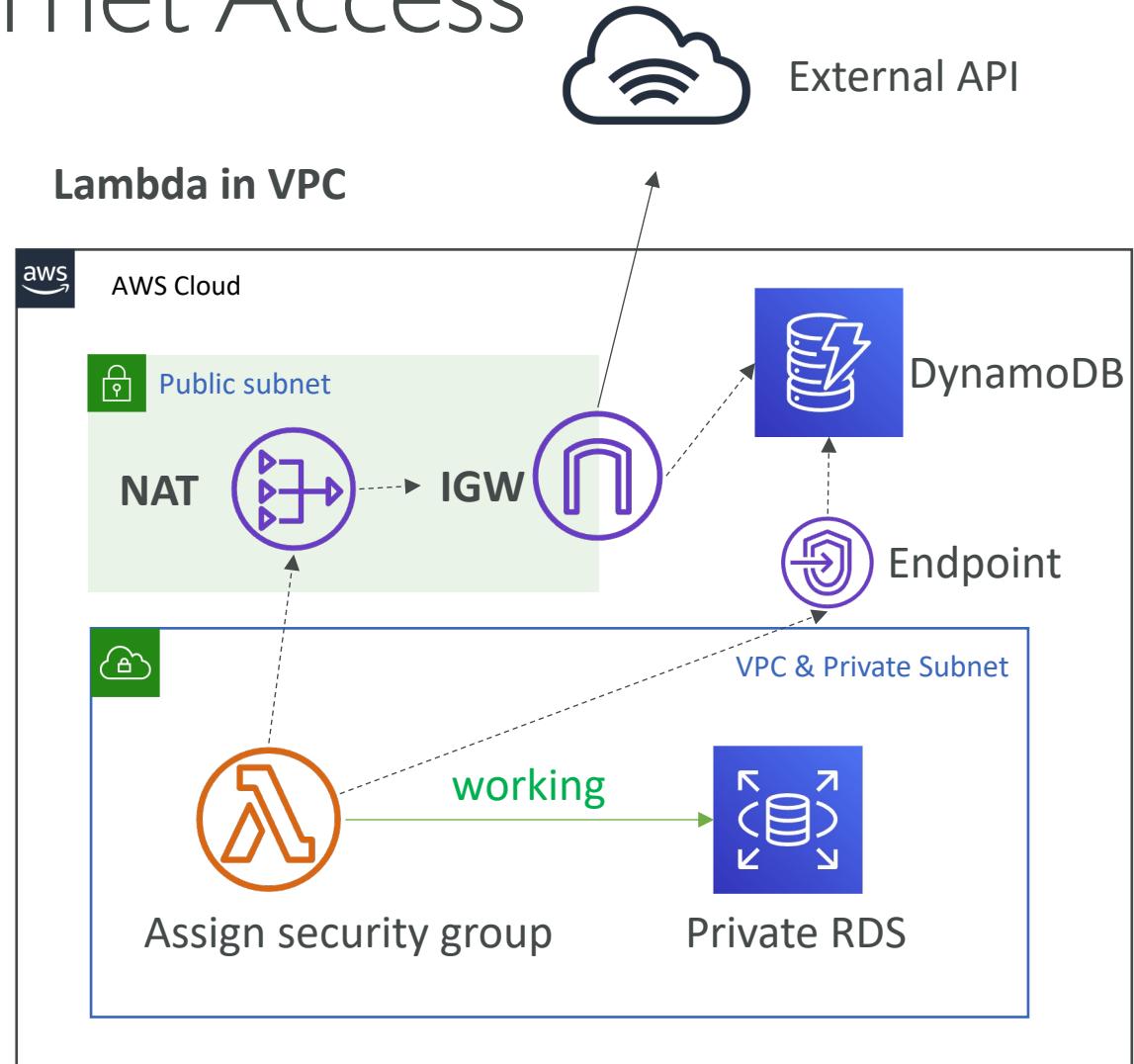
# Lambda in VPC

- You must define the VPC ID, the Subnets and the Security Groups
- Lambda will create an ENI (Elastic Network Interface) in your subnets
- `AWSLambdaVPCAccessExecutionRole`



# Lambda in VPC – Internet Access

- A Lambda function in your VPC does not have internet access
- Deploying a Lambda function in a public subnet does not give it internet access or a public IP
- Deploying a Lambda function in a private subnet gives it internet access if you have a **NAT Gateway / Instance**
- You can use **VPC endpoints** to privately access AWS services without a NAT

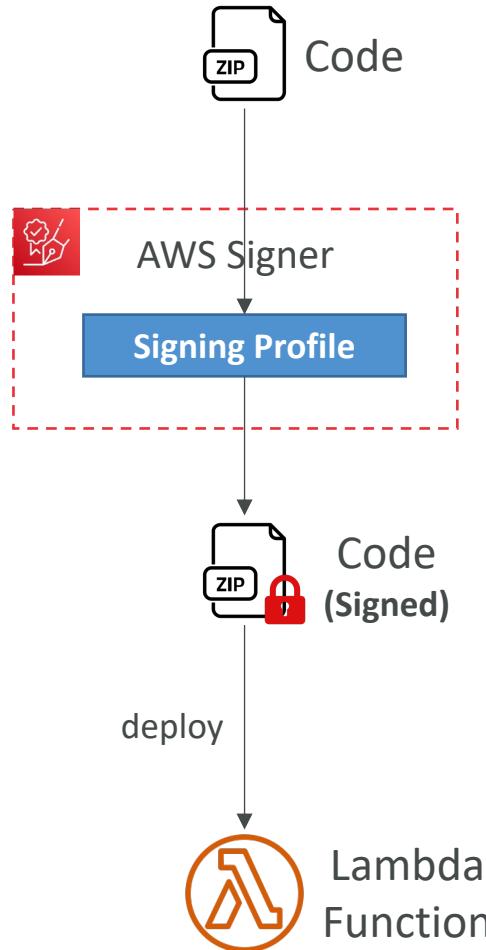


**Note:** Lambda - CloudWatch Logs works even without endpoint or NAT Gateway

# AWS Signer



- Fully managed Code-Signing service to ensure trust & integrity of your code
- Manages Code-Signing certificate public and private keys
- Code is validated against a digital signature to confirm that the code is unaltered and from a trusted publisher
- **Code-Signing for AWS Lambda**
  - Digitally sign code packages for Lambda
  - Enforce only trusted code runs in Lambda functions
  - Not supported for Container Lambda functions
- **Code-Signing for AWS IoT**
  - Sign code you create for AWS IoT and Amazon FreeRTOS
  - Integrated with ACM to generate/import certificates



# AWS Signer – Revoking Signing Profile

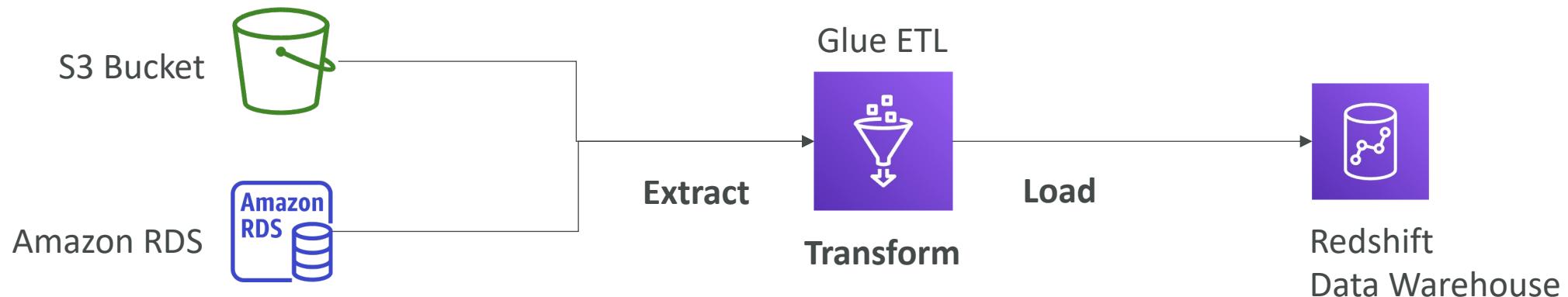
- For Lambda Functions, you can revoke the signature of Lambda deployment package (invalidate it)
- Causing it to fail Lambda signature checks
- Either revoke individual signatures using `RevokeSignature` API call or revoke a Signing Profile using `RevokeSigningProfile` API call
- Note: Use only in critical scenarios (irreversible)



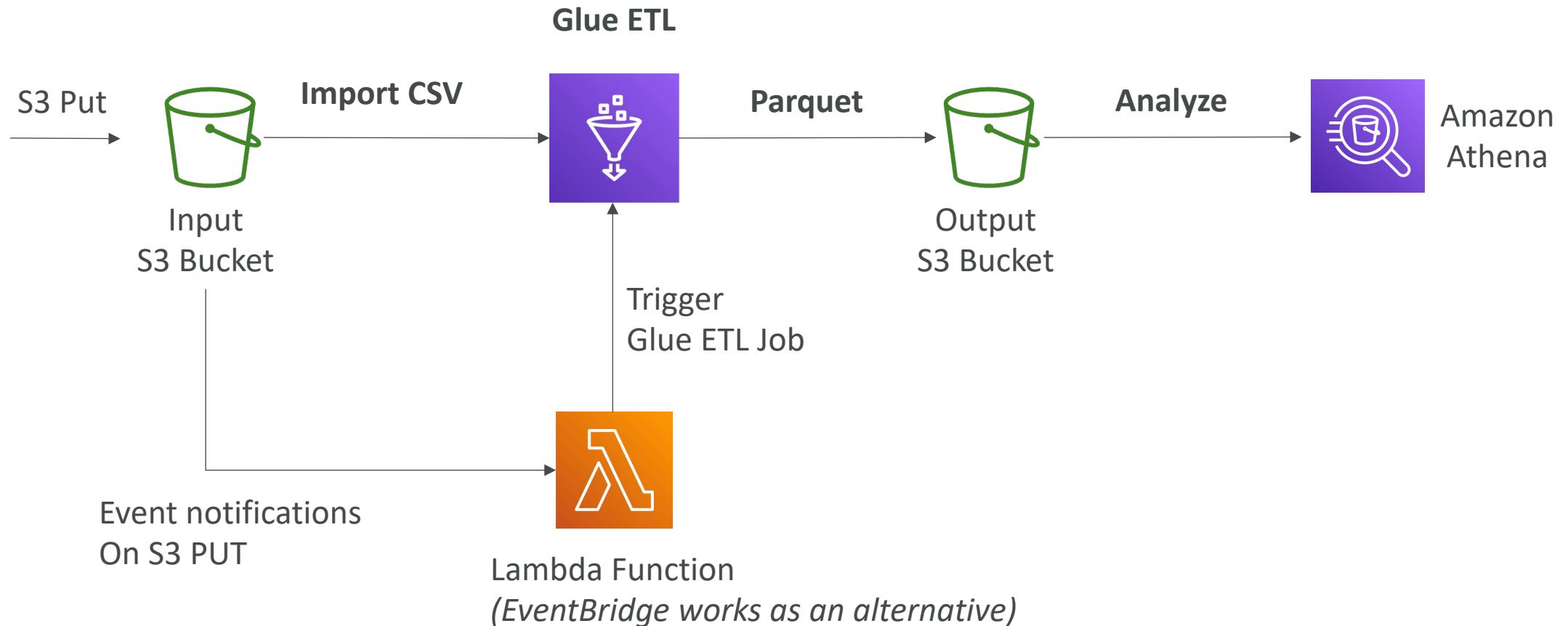
# AWS Glue



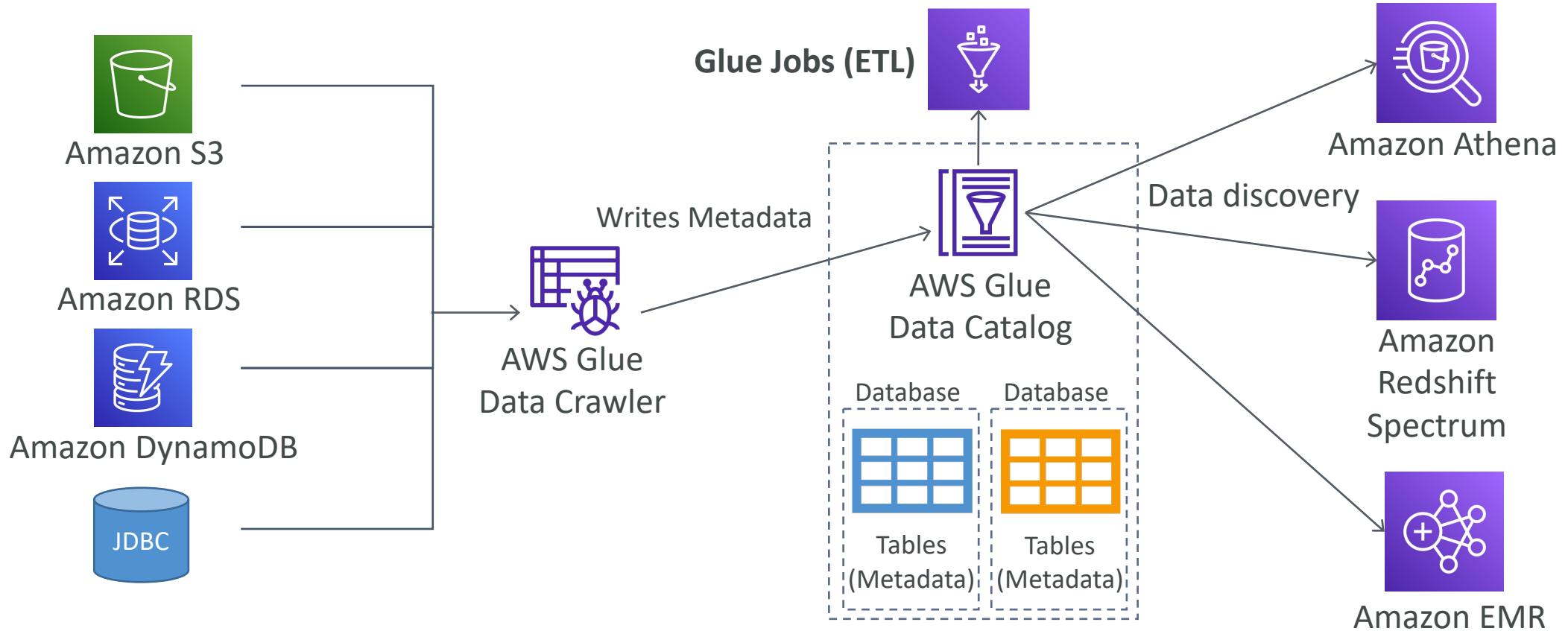
- Managed **extract, transform, and load (ETL)** service
- Useful to prepare and transform data for analytics
- Fully **serverless** service



# AWS Glue – Convert data into Parquet format



# Glue Data Catalog: catalog of datasets



# Glue – things to know at a high-level

- **Glue Job Bookmarks:** prevent re-processing old data
- **Glue Elastic Views:**
  - Combine and replicate data across multiple data stores using SQL
  - No custom code, Glue monitors for changes in the source data, serverless
  - Leverages a “virtual table” (materialized view)
- **Glue DataBrew:** clean and normalize data using pre-built transformation
- **Glue Studio:** new GUI to create, run and monitor ETL jobs in Glue
- **Glue Streaming ETL** (built on Apache Spark Structured Streaming): compatible with Kinesis Data Streaming, Kafka, MSK (managed Kafka)

# AWS Glue – Security

- Encryption at rest using KMS for databases, tables, Job Bookmarks, ...
- Encryption in transit using TLS
- Identity-Based Policies – grant access to Glue resources by attaching policies to IAM identities (user, group, role, service)
- Resource-Based Policies – policy attached to **Glue Data Catalog** to grant access to IAM identities (used for cross-account access)

```
"Version": "2012-10-17",
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "glue:GetTable",
      "glue:GetTables",
      "glue:GetDatabase",
      "glue:GetDatabases"
    ],
    "Resource": [
      "arn:aws:glue:us-west-2:123456789012:catalog",
      "arn:aws:glue:us-west-2:123456789012:database/db1",
      "arn:aws:glue:us-west-2:123456789012:table/db1/books"
    ]
  }
]
```

Identity-Based Policy

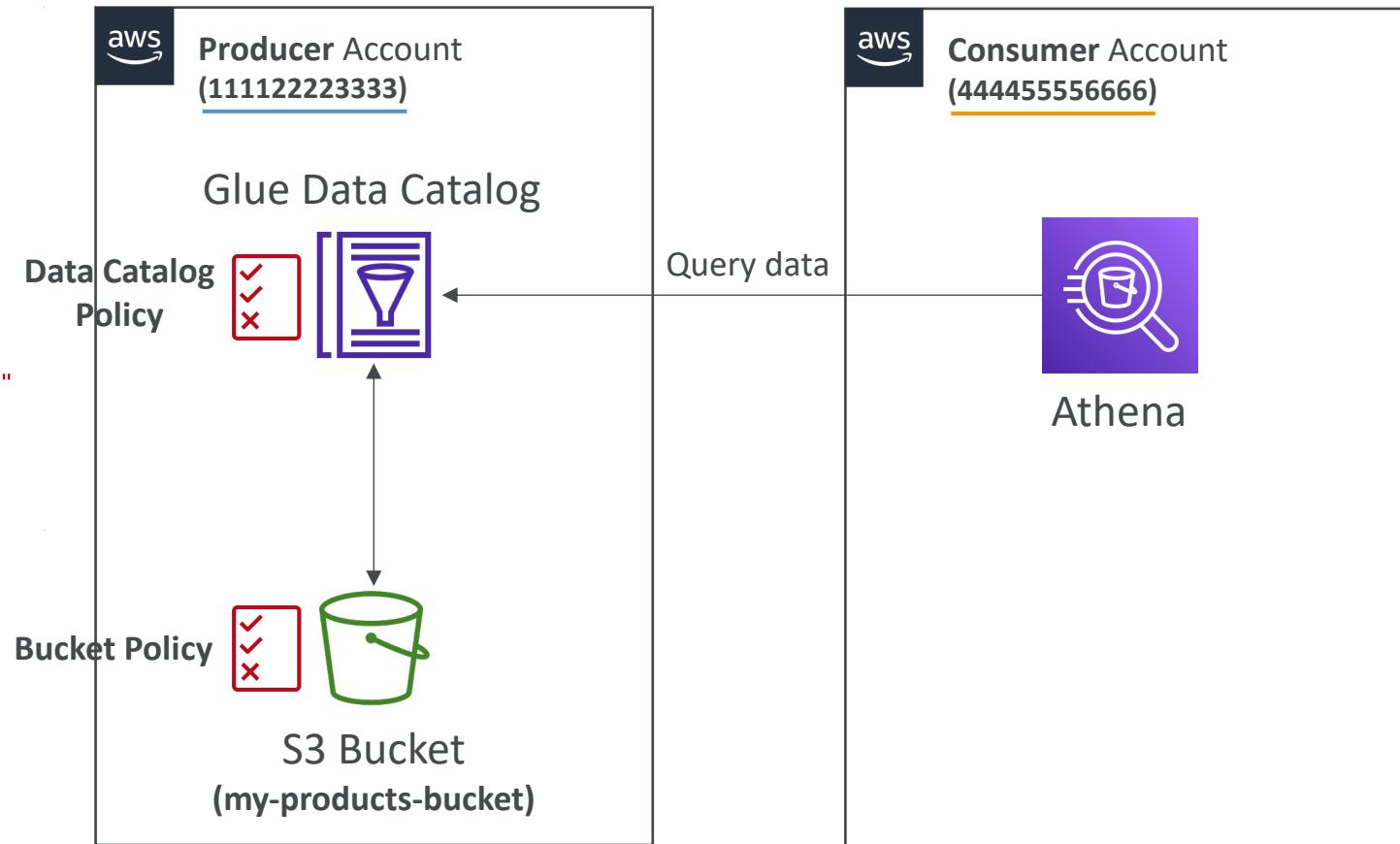
# AWS Glue – Centralized Data Catalog

```

{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::444455556666:root"
            },
            "Action": ["glue:GetDatabases*", "glue:GetTables*"],
            "Resource": [
                "arn:aws:glue:us-east-1:111122223333:catalog",
                "arn:aws:glue:us-east-1:111122223333:database/producer_database",
                "arn:aws:glue:us-east-1:111122223333:table/producer_database/orders"
            ]
        }
    ],
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::444455556666:root"
            },
            "Action": [
                "s3:GetObject",
                "s3>ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::my-products-bucket",
                "arn:aws:s3:::my-products-bucket/orders/*"
            ]
        }
    ]
}
  
```

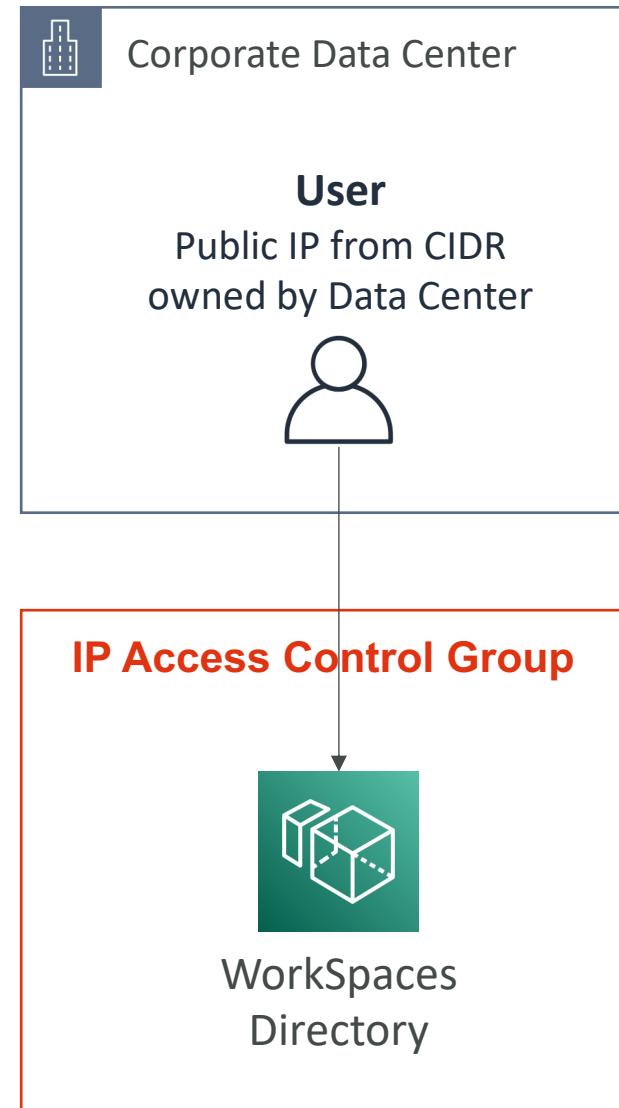
**Glue Data Catalog Policy**

**S3 bucket Policy**



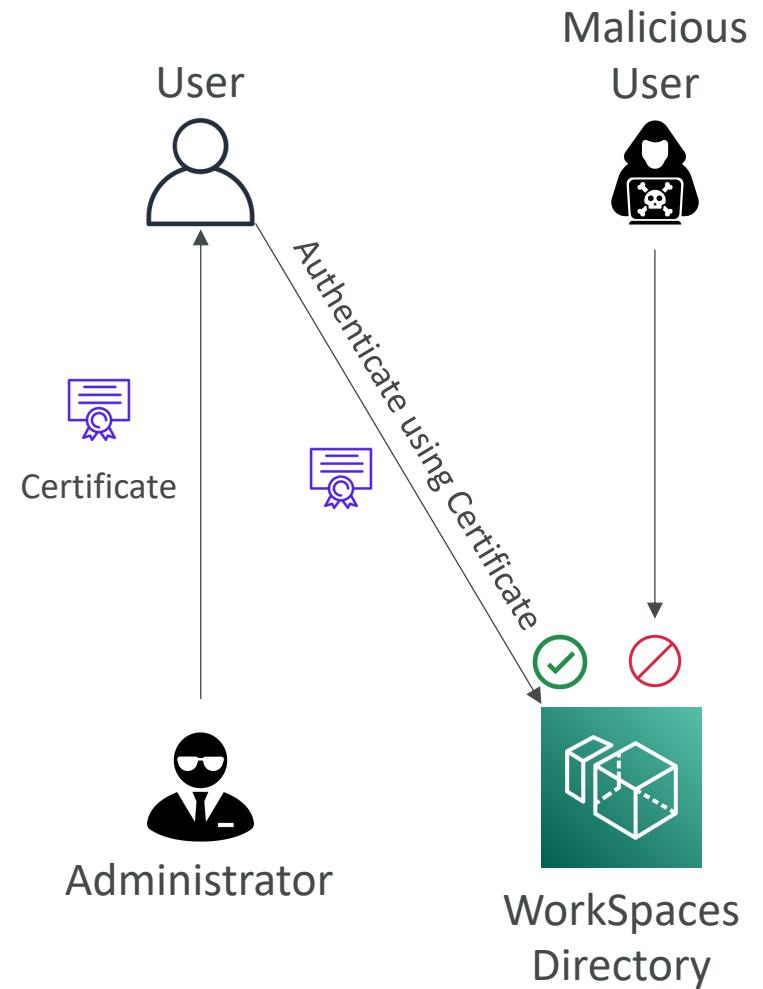
# Amazon WorkSpaces – Security IP Access Control Groups

- Similar to Security Groups but for WorkSpaces
- List of IP addresses / CIDR address ranges that users are authorized to connect from
- If users access WorkSpaces through VPN or NAT, the IP Access Control Group must authorize the public IP of these



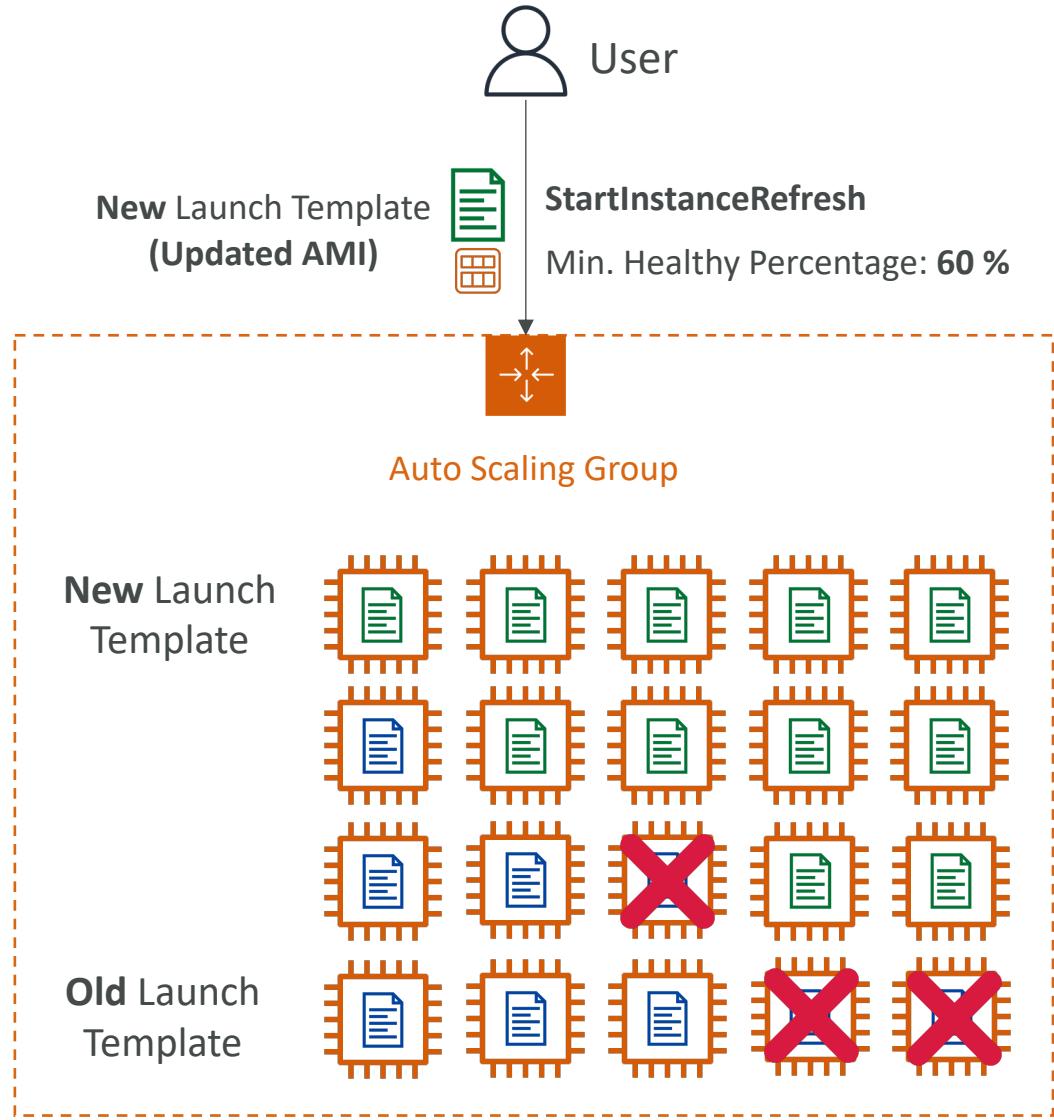
# Amazon WorkSpaces – Security Access Control Options and Trusted Devices

- Manage which client devices can access WorkSpaces
- WorkSpaces Certificate-based Authentication
  - Limit access only to trusted devices using Digital Certificates
  - Only Windows, MacOS, and Android clients



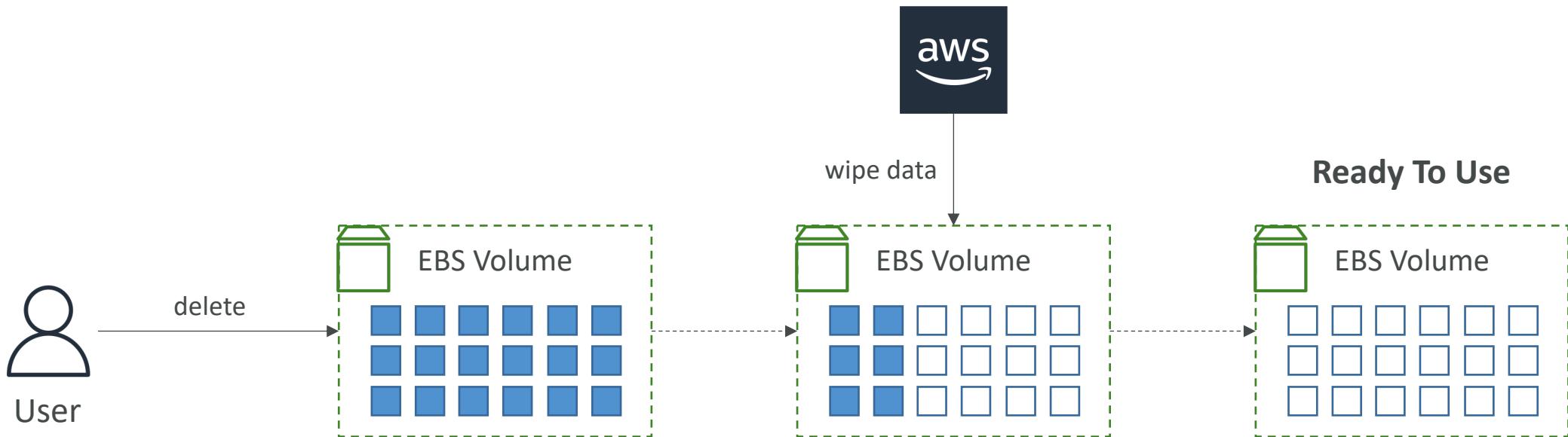
# Auto Scaling – Instance Refresh

- Goal: update launch template and then re-creating all EC2 instances
- For this we can use the native feature of Instance Refresh
- Setting of minimum healthy percentage
- Specify warm-up time (how long until the instance is ready to use)



# EBS Volumes – Data Wiping

- When you delete an EBS volume..
- The physical block storage used by deleted EBS volumes is overwritten with zeroes before it is allocated to a new volume
- Note: no need to manually wipe data





# AWS CloudShell

The screenshot shows the AWS CloudShell interface. At the top, there's a navigation bar with the AWS logo, a "Services" dropdown, a search bar ("Search for services, features, marketplace products, and docs [Alt+S]"), and "Select a Region" and "Support" dropdowns. Below the search bar, the title "AWS CloudShell" is displayed, along with an "Actions" dropdown and a gear icon. A sub-header "eu-west-1" is shown above the terminal area. The terminal itself has a dark background and displays the following text:  
Preparing your terminal...  
Try these commands to get started:  
aws help or aws <command> help or aws <command> --cli-auto-prompt  
[cloudshell-user@ip-1-2-3-4 ~]\$ █

At the bottom of the screen, there are links for "Feedback", "English (US) ▾", "Privacy Policy", and "Terms of Use". A copyright notice at the very bottom states: "© 2008 - 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved."

# AWS CloudShell



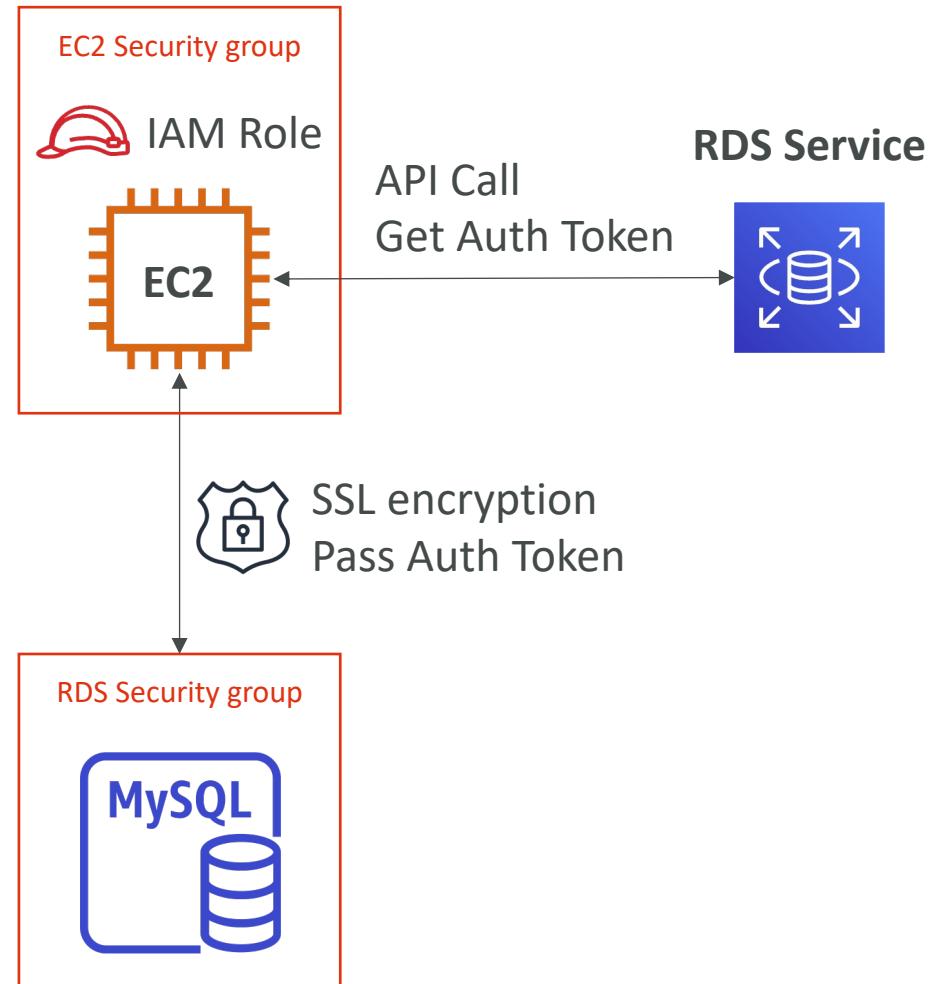
- Browser-based shell to manage your AWS resources
- Run AWS CLI commands & run scripts without leaving your browser
- Pre-installed common libraries and command-line tools:
  - AWS CLI, ECS CLI, SAM CLI, and AWS SDKs for Python and Node.js
  - Bash, Powershell, zsh, vi (editor), git, npm, pip, ...
- **Pre-configured AWS Credentials** (inherit from the logged in user)
- Persistent storage with up to 1GB per AWS region
- Runs on Amazon Linux 2
- **Can't access resources in a VPC (EC2, RDS, ...)**

# RDS & Aurora Security

- **At-rest encryption:**
  - Database master & replicas encryption using AWS KMS – must be defined as launch time
  - If the master is not encrypted, the read replicas cannot be encrypted
  - To encrypt an un-encrypted database, go through a DB snapshot & restore as encrypted
- **In-flight encryption:** TLS-ready by default, use the AWS TLS root certificates client-side
- **IAM Authentication:** IAM roles to connect to your database (instead of username/pw)
- **Security Groups:** Control Network access to your RDS / Aurora DB
- **No SSH available** except on RDS Custom
- **Audit Logs can be enabled** and sent to CloudWatch Logs for longer retention

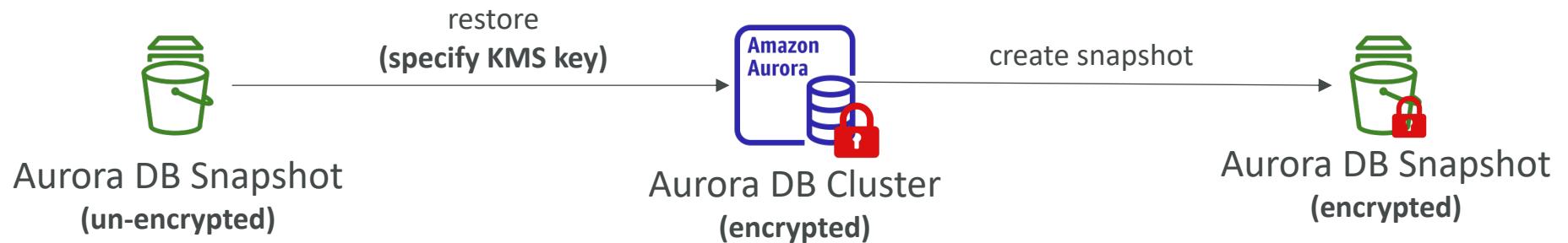
# RDS - IAM Authentication

- IAM database authentication works with MariaDB, MySQL and PostgreSQL
- You don't need a password, just an authentication token obtained through IAM & RDS API calls
- Auth token has a lifetime of 15 minutes
- Benefits:
  - Network in/out must be encrypted using SSL
  - IAM to centrally manage users instead of DB
  - Can leverage IAM Roles and EC2 Instance profiles for easy integration



# Aurora – Encrypt un-encrypted Snapshot

- When you encrypt Aurora cluster then all DB instances, snapshots, logs, and backups are all encrypted
- You can't create an encrypted snapshot of an un-encrypted DB cluster
- Note: you can't encrypt an un-encrypted snapshot while you copy the snapshot



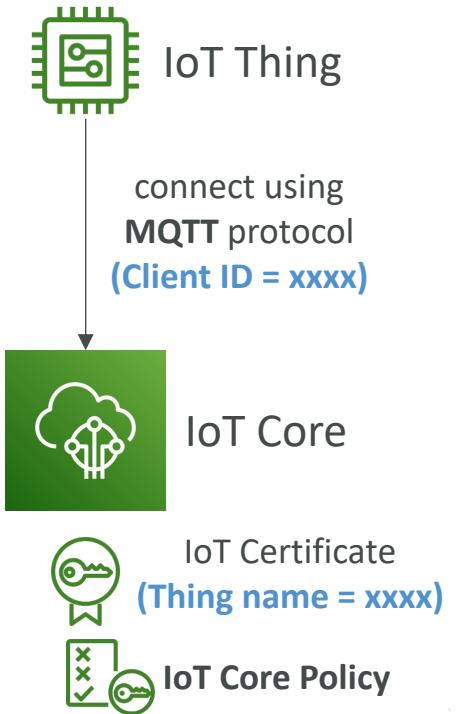
# AWS IoT Core – Security

- **IoT Core Policies**

- JSON documents allow you to control access to IoT Core Control Plane
- IoT Core message broker, send/receive MQTT messages...
- Attached to IoT Certificate or Cognito Identities (IoT Thing)
- Example: allow access to all MQTT topics or restrict access to a single topic

- **IoT Thing Policy Variable**

- Allows you to write IoT Core Policies with permissions based on IoT Thing properties (e.g., Thing name, Thing type, Thing attribute values, ...)
- Replaced when an IoT Thing connects to AWS IoT



```

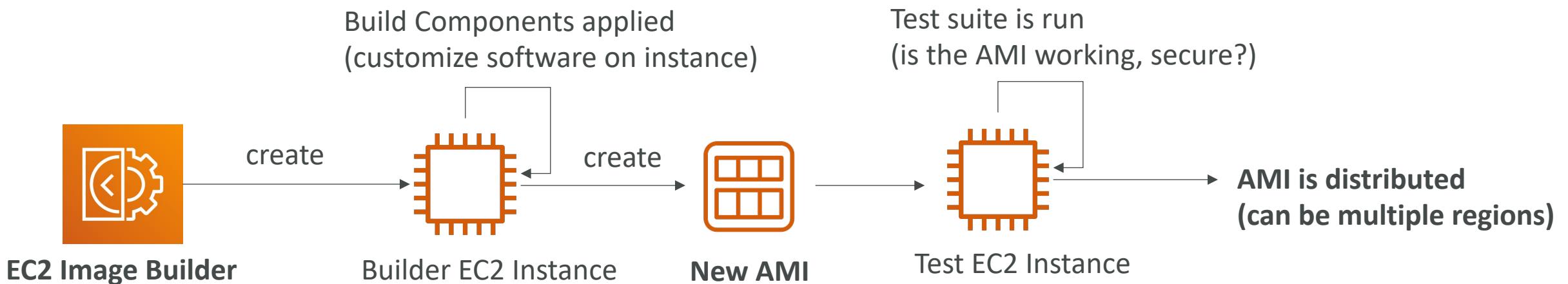
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iot:Connect",
      "Resource": [
        "arn:aws:iot:us-east-1:123456789012:  
client/${iot:Connection.Thing.ThingName}"
      ]
    }
  ]
}
  
```

requires that

**IoT Thing name = Client ID for MQTT connection**

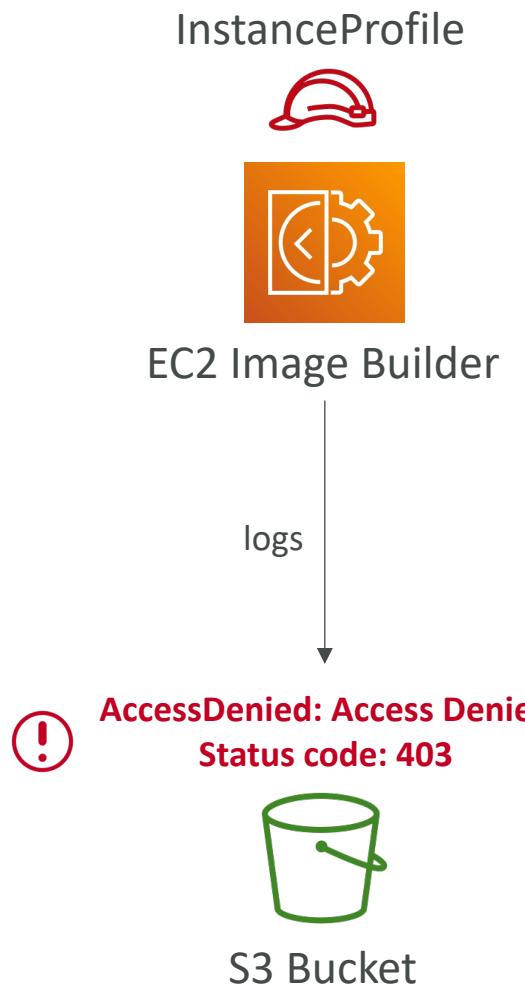
# EC2 Image Builder

- Used to automate the creation of Virtual Machines or container images
- => Automate the creation, maintain, validate and test **EC2 AMIs**
- Can be run on a schedule (weekly, whenever packages are updated, etc...)
- Free service (only pay for the underlying resources)



# EC2 Image Builder – Troubleshooting

- Error: “AccessDenied: Access Denied status code: 403”
- Reasons:
  - Instance profile Role doesn’t have the required permissions
  - Instance Profile Role is missing permissions required for logging to Amazon S3
- Resolution:
  - Make sure the following Managed Policies added to your Instance Profile Role:
    - AmazonSSMManagedInstanceCore, EC2InstanceProfileForImageBuilder, EC2InstanceProfileForImageBuilderECRContainerBuilds
  - Add a Policy to your Instance Profile Role that grants PutObject permissions to write to your S3 Bucket



# Amazon Redshift – Database Hierarchy

- **Superusers**

- Have the same permissions as DB owners for all databases
- Example: *admin* user which is created when you launch the Redshift Cluster
- You must be a superuser to create a superuser

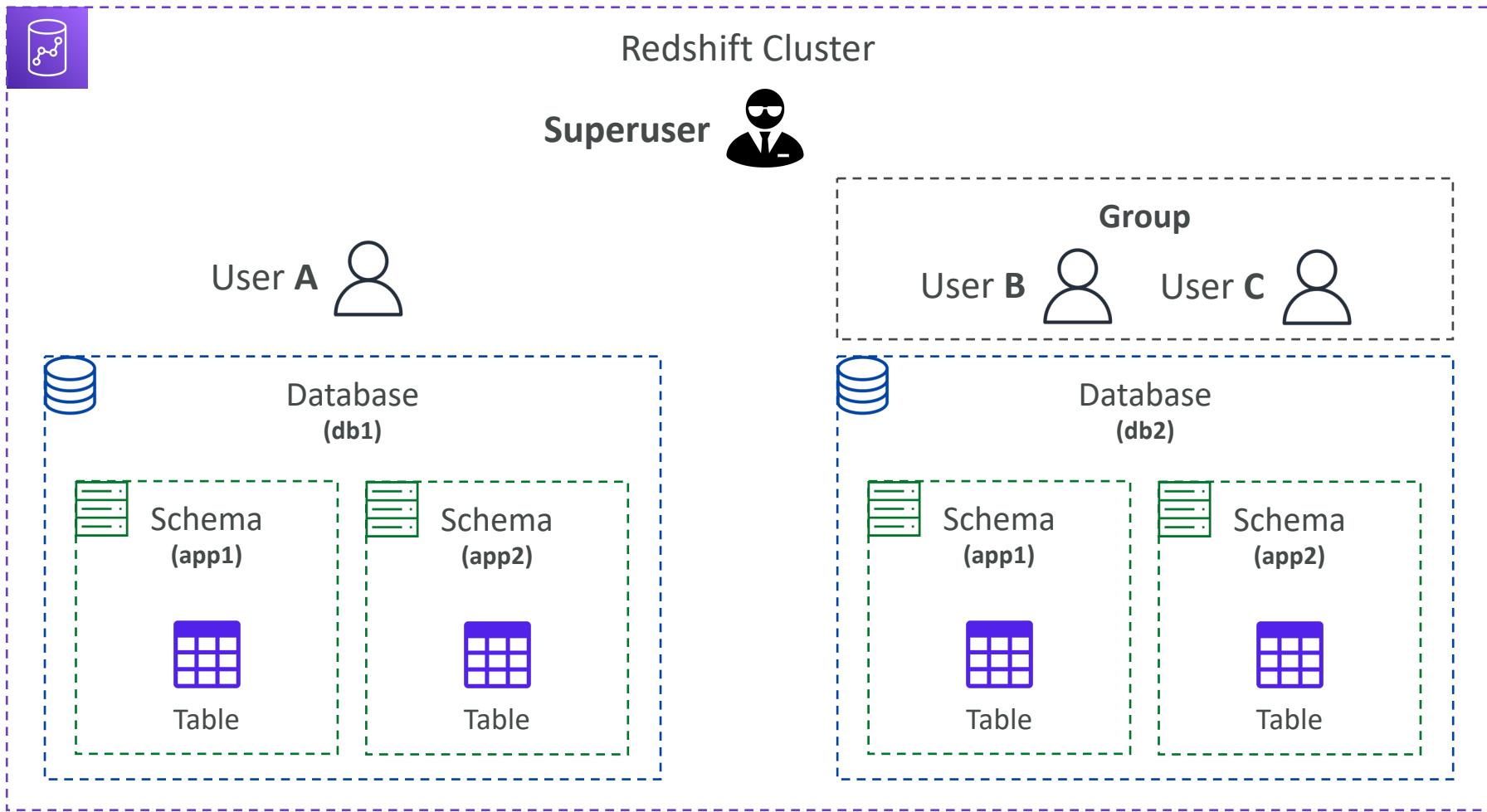
- **Users**

- Can only be created and dropped by a **superuser**
- Can own databases and database objects (e.g., tables)
- Can grant permissions on those objects to other DB Users, Groups, and Schemas
- User is granted permissions in two ways
  - Explicitly, by having those permissions assigned directly to the account
  - Implicitly, by being a member of a group that is granted permissions

# Amazon Redshift – Database Hierarchy

- **Groups**
  - Collections of Users that can be collectively assigned permissions
  - Good for streamlined security maintenance
- **Database**
  - Collection of one or more Schema
  - When a User create a database, the User becomes its owner
  - Superusers have the same permissions as database owners
- **Schema**
  - Collections of database tables and other database objects
  - Used to group database objects under a common name
  - Users can be granted access to a single schema or to multiple schemas

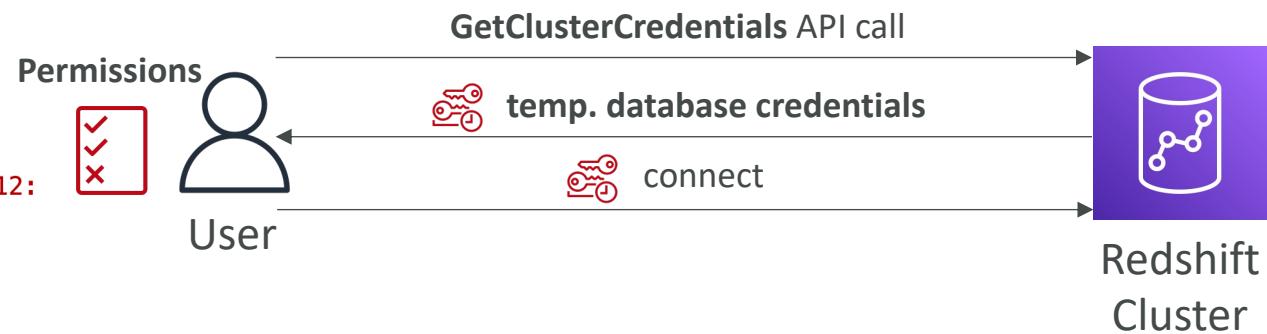
# Amazon Redshift – Database Hierarchy



# Amazon Redshift – in-DB Security

- Commonly, Users log in to database by providing a database username and password
- However, you don't have to maintain usernames and passwords in your Amazon Redshift database
  - You can configure your system to permit users to create database credentials and log in to the database based on their IAM credentials
  - Redshift provides the **GetClusterCredentials** API operation to generate temporary database user credentials

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {"Effect": "Allow",  
     "Action": "redshift:GetClusterCredentials",  
     "Resource": "arn:aws:redshift:us-west-2:123456789012:  
dbuser:examplecluster/temp_creds_user"}]  
}  
existing database user
```



# Amazon Redshift – in-DB Security

- When you create temporary user credentials for an existing database user, you can disable the user's password to force the user to log on with the temporary password
- Alternatively, you can use the **GetClusterCredentials Autocreate** option to automatically create a new database user each time you connect
- To create a read-only user, add a user to a group that only has **read-only privileges to the specified Schemas for a database**
  - Note that you will still have to initially manually specify all of the schema names & then subsequently modify the group for any new schemas that you may create

# Exam Preparation

# Quick note on Course Coverage

- There are many services you will find in questions that are **distractors**
- There are **over 200 AWS services**, and we can't cover them all
- I have covered all services that from my research and experience, people get questions for at the exam.
  
- If you see a service not covered by my course, but in someone else's practice exam, don't panic, I must have intentionally left it out
- If you see a service that is an answer at the exam but not covered in my course, please let me know:  
<https://links.datacumulus.com/aws-cert-feedback>

# How will the exam work?

- You'll have to register online at <https://www.aws.training/>
- Fee for the exam is 300 USD
- No notes are allowed, no pen is allowed, no speaking
- 65 questions will be asked in 170 minutes
- At the end you can optionally review all the questions / answers
  
- You will know few days later if you passed / failed the exams
- You will know the overall score at the same time
- You will not know which answers were right / wrong
- To pass you need a score of **at least 750 out of 1000**
- If you fail, you can retake the exam again 14 days later

# State of learning checkpoint

- Let's look how far we've gone on our learning journey
- <https://aws.amazon.com/certification/certified-security-specialty/>

# Congratulations!

# Congratulations!

- Congrats on finishing the course!
- I hope you will pass the exam without a hitch ☺
- If you haven't done so yet, I'd love a review from you!
- If you passed, I'll be more than happy to know I've helped
  - Post it in the Q&A to help & motivate other students. Share your tips!
  - Post it on LinkedIn and tag me!
- Overall, I hope you learned how to use AWS and that you will be a tremendously good AWS Security Specialist